Intra-Domain Routing

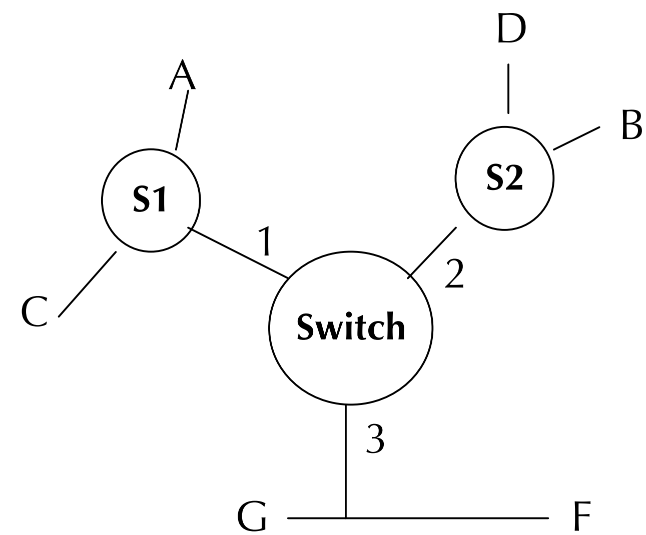
6.829

Jacob Strauss

September 14, 2006

Review: Learning Bridges (Switches)

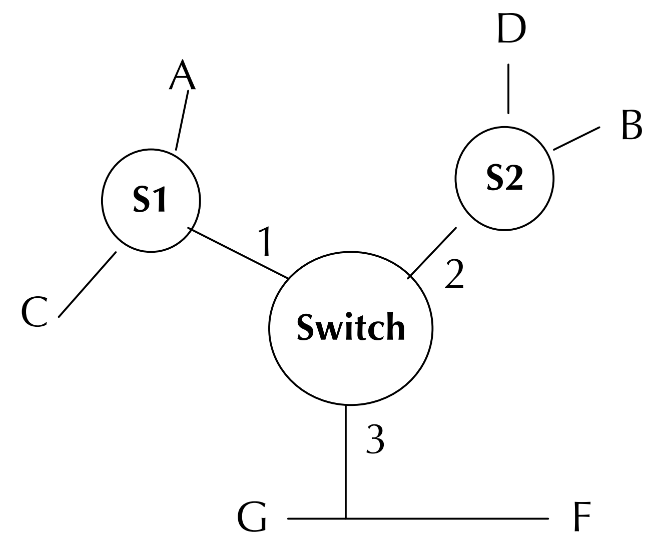
- Bridge builds a forwarding table
  - Destination -> Output port
  - Learned from incoming packets
- Forwarding:
  - For every packet, we need to look up the output port toward its destination
  - If address not found or broadcast flood to all but input port
  - Update forwarding table
- Loop Avoidance
  - Elect a root Bridge
  - Construct Spanning Tree to root



Destination	Port
A	1
B	2
C	1
D	2
F	3
G	3

Learning Bridge Scaling Problems

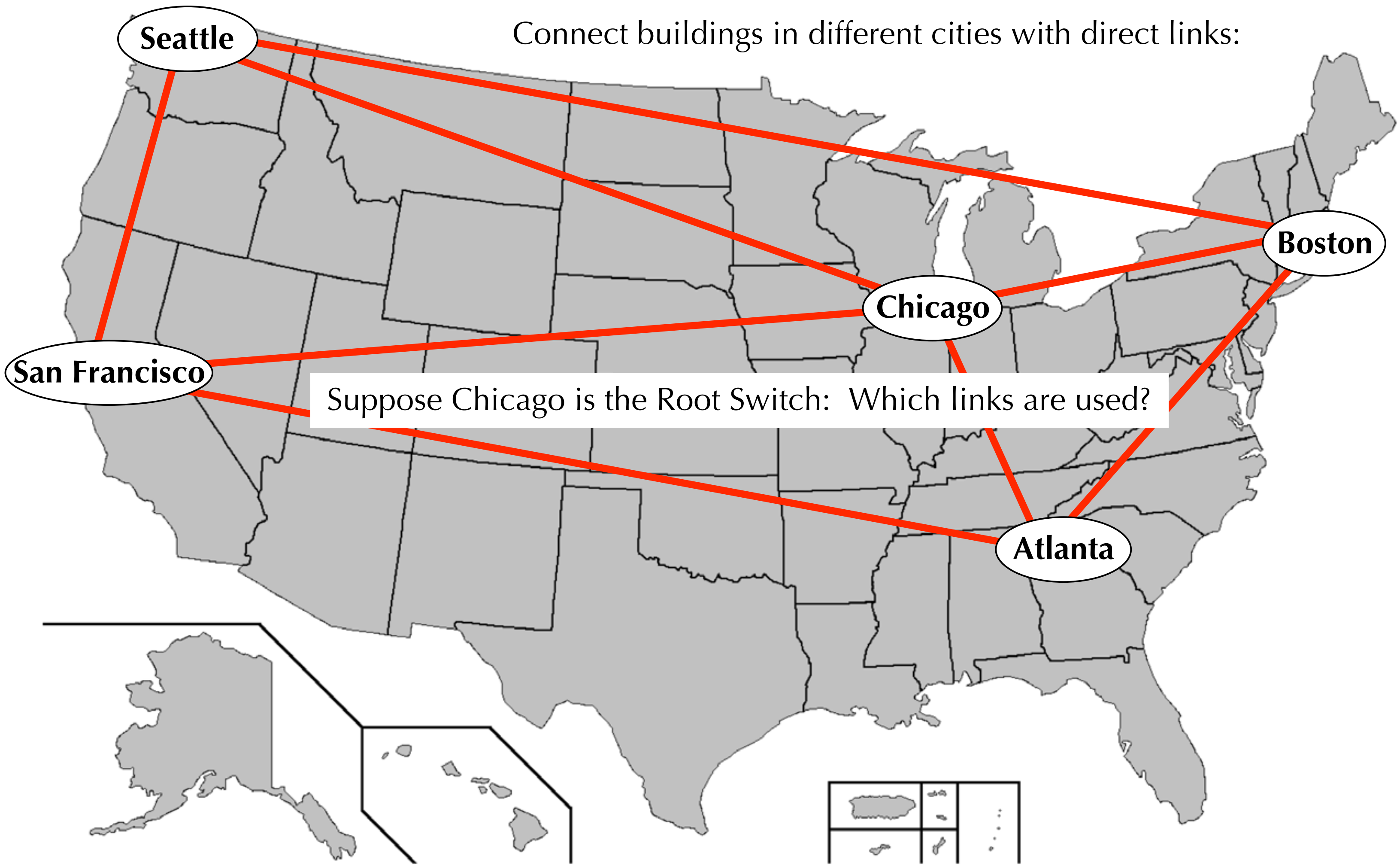
- Forwarding entry per destination
  - Large tables
  - Floods for unknown destinations
- Cannot mix physical network types
- Inefficient Routes
  - Concentrates traffic at a few switches
  - Not shortest path
  - Okay for short paths, not for long
  - Cannot use redundancy



Destination	Port
A	1
B	2
C	1
D	2
F	3
G	3

Bridge Scaling Problems

Connect buildings in different cities with direct links:



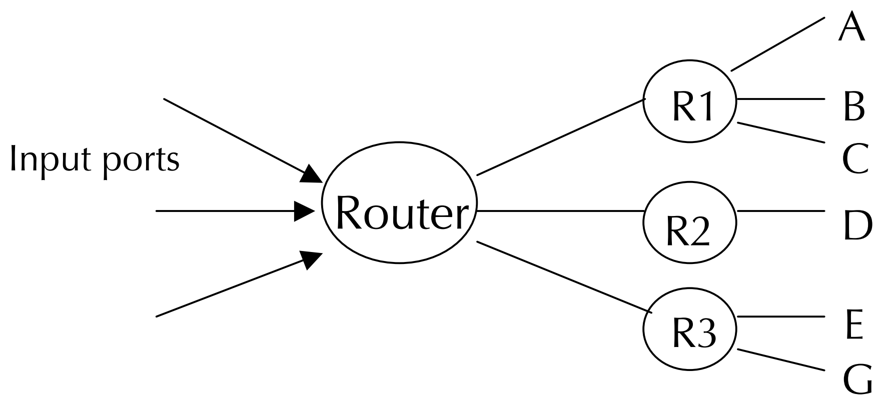
Suppose Chicago is the Root Switch: Which links are used?

Bridge Scaling Problems

All packets go through Chicago switch -- not shortest path

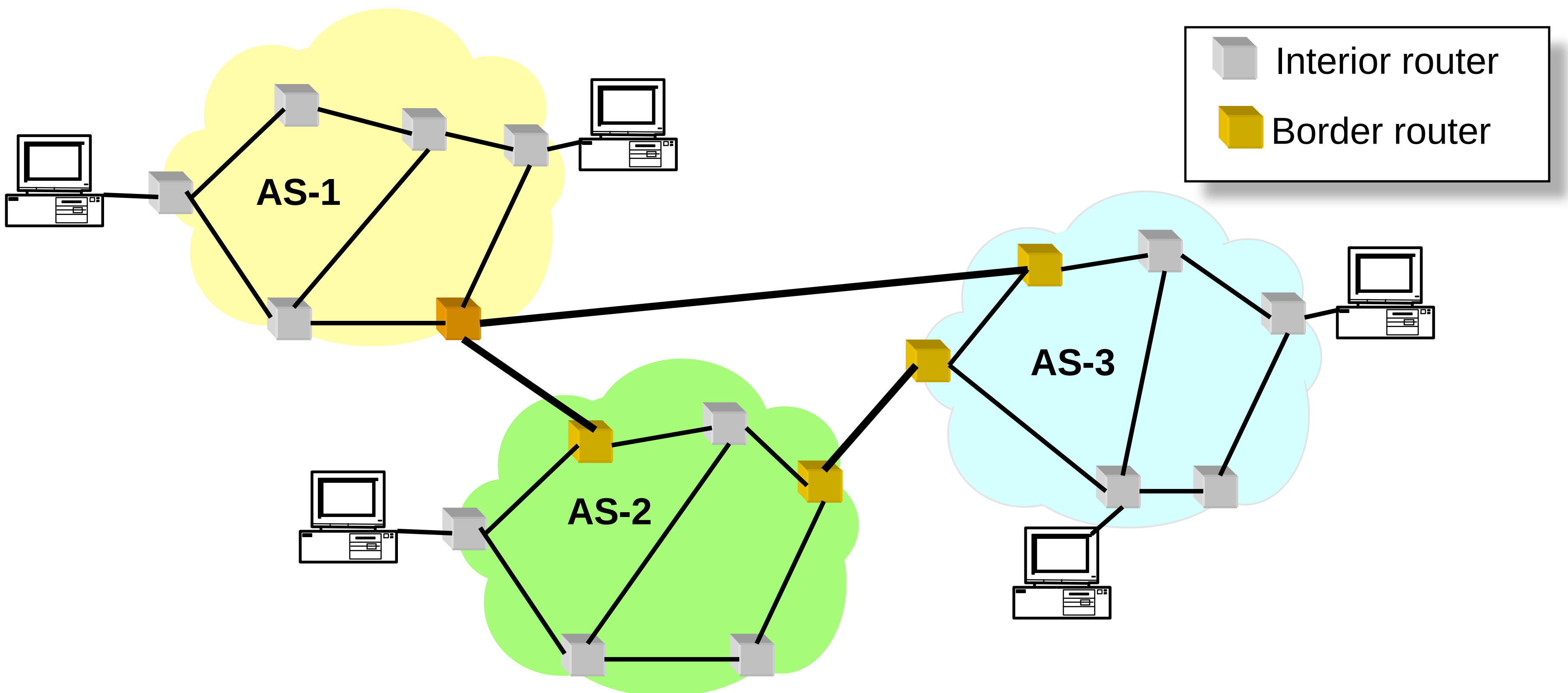
Add a layer over Ethernet: IP & Routing

- Add a new protocol over physical layer
  - No longer tied to Ethernet
- Hierarchal Addressing
  - All addresses in Boston start with 18.1.x.x
    Chicago start with 18.2.x.x
  - Forwarding tables stay small with fewer updates
- Separate Routing from Forwarding
  - Routing is finding the path
  - Forwarding is action of sending the packet to the next-hop toward its destination
- Each router has a forwarding table
  - Forwarding tables are created by a routing protocol



Destination	Next-hop
A -- C	R1
D	R2
E	R3
G	R3

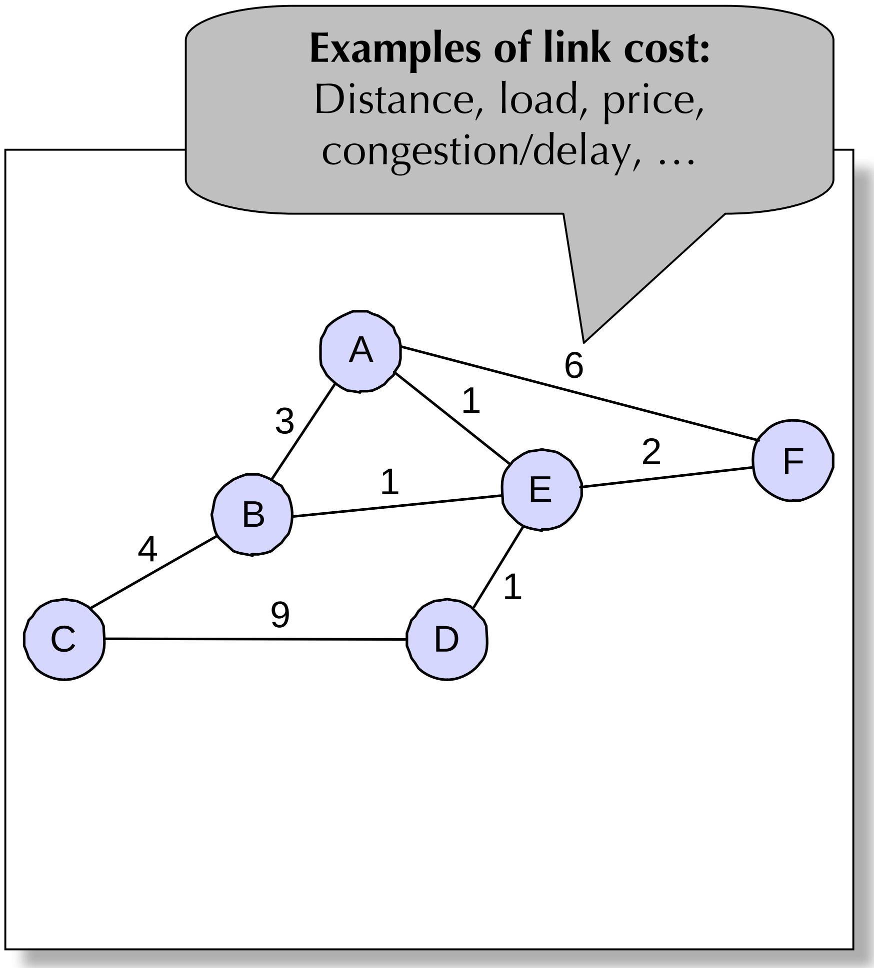
Picture of the Internet



- Internet: A collection of Autonomous Systems (AS)
  - Defined by control, not geography
- Routing:
  - Intra-Domain Routing (this lecture)
  - Inter-Domain Routing (BGP: next lecture)

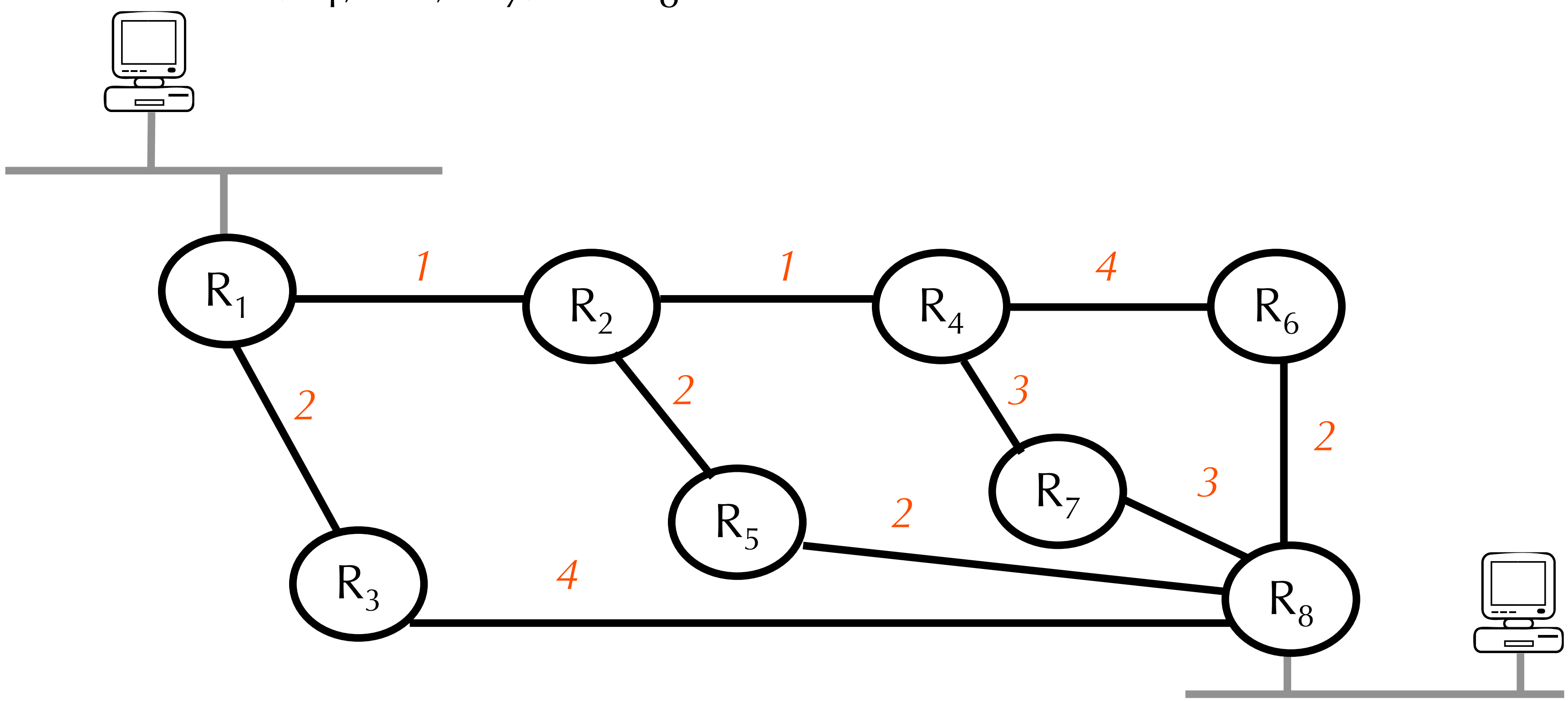
Factors Affecting Routing

- Routing algorithms view the network as a graph
  - Intra-domain routing: nodes are routers
  - Inter-domain routing: nodes are ASes
- Problem: find lowest cost path between two nodes (Shortest Path)
- Factors
  - Semi-dynamic topology (deal with link failures)
  - Dynamic load
  - Policy

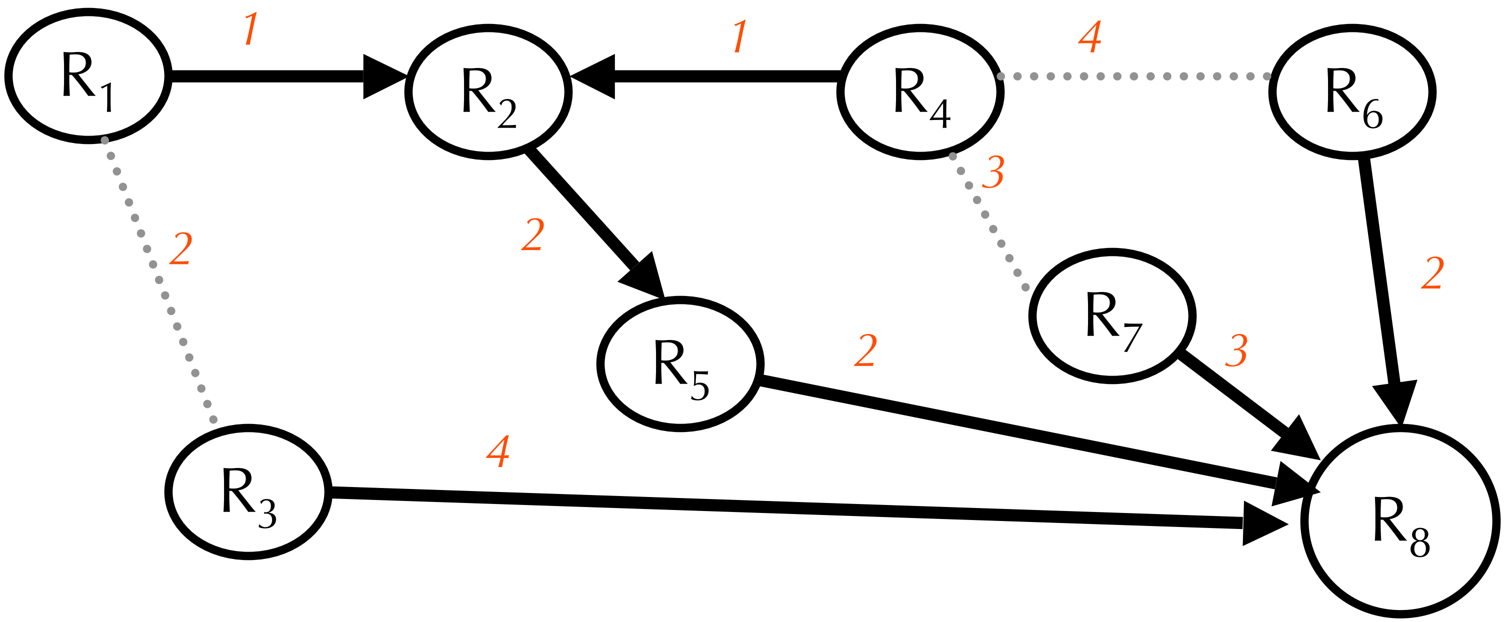


Problem: Shortest Path Routing

<u>Objective:</u> Determine the route from each router (R_1, …, R_7) to R_8 that minimizes the cost.



Solution is simple by inspection... (in this case)



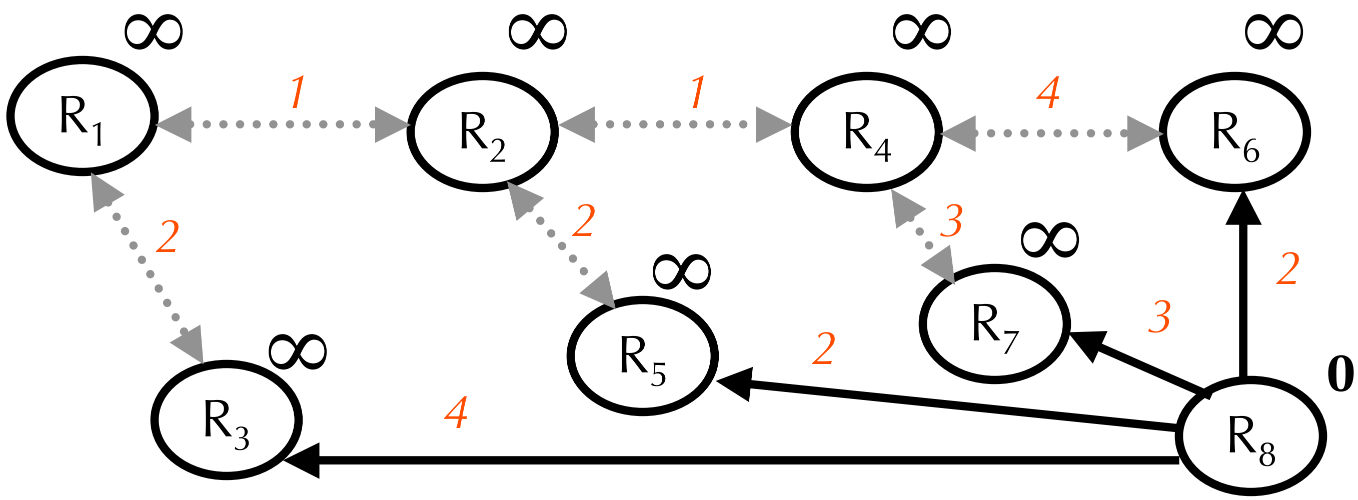
The shortest paths from all sources to a destination (e.g., R_8) is the spanning tree routed at that destination.

Two Main Approaches

- Distance Vector Protocols
  - E.g., RIP (Routing Information Protocol)
  - Based on Distributed Bellman-Ford Algorithm
- Link State Protocols
  - E.g., OSPF (Open Shortest Path First)
  - Based on Dijkstra Algorithm

Technique1: Distributed Bellman-Ford Algorithm

Example



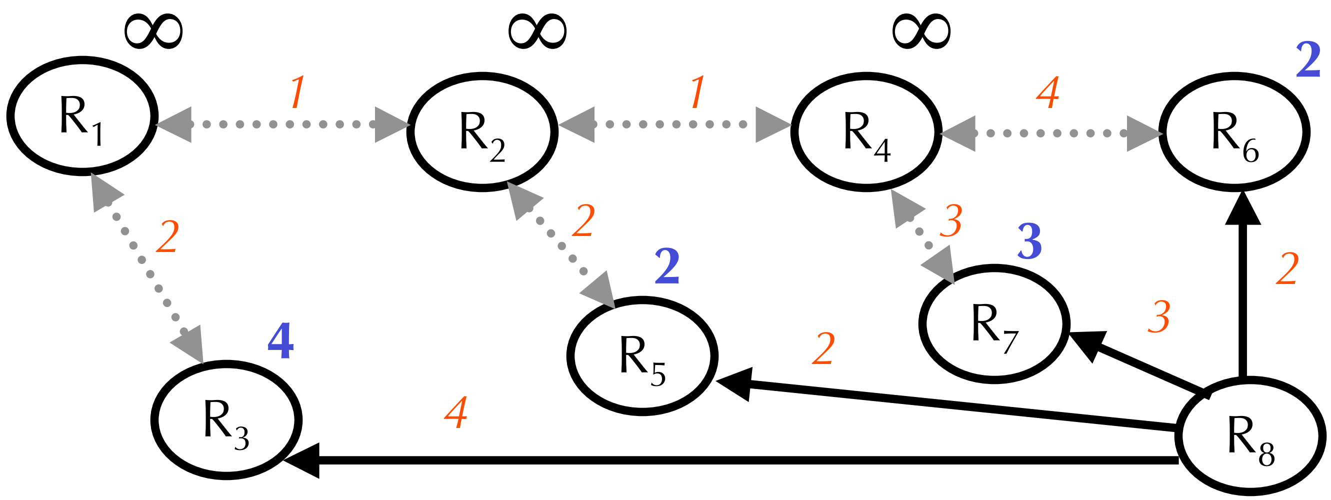
Each router keeps track of next hop to destination, cost to destination

Initial State: All routers except R8 set their route cost to ∞. R8 sets its route cost to 0.

Technique1: Distributed Bellman-Ford Algorithm

Example

R_1	Inf
R_2	Inf
R_3	4, R_8
R_4	Inf
R_5	2, R_8
R_6	2, R_8
R_7	3, R_8



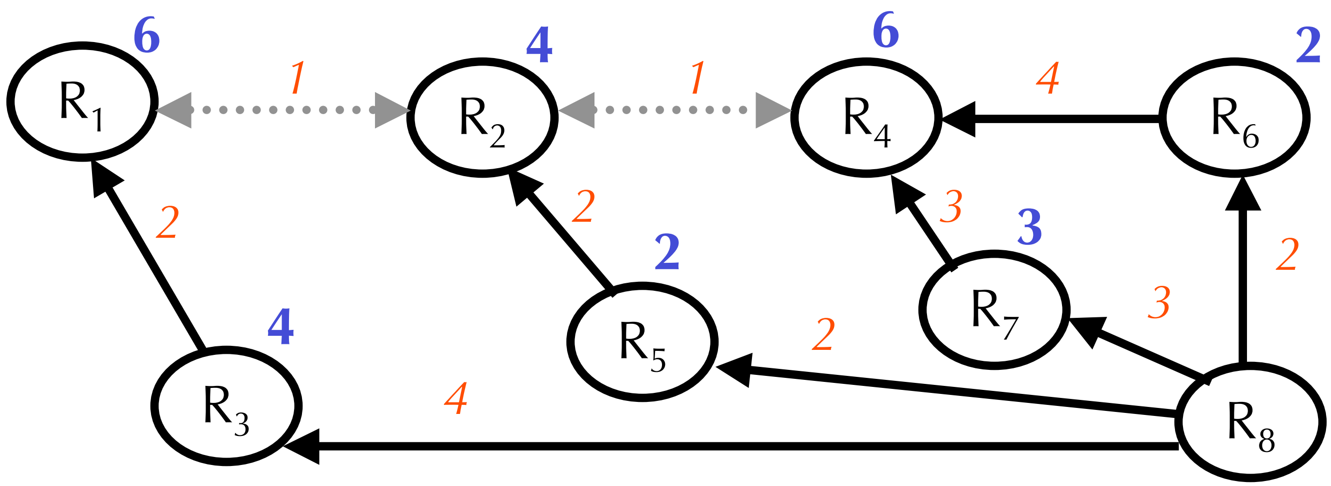
- Every T seconds, each Router tells its neighbors its route cost to R8
- Each router updates its cost as *min(current cost, received cost + link cost)*
- Set next hop to the source of the lowest cost message

Routing tables have both the next-hop and the cost

Technique1: Distributed Bellman-Ford Algorithm

Example

R_1	6, R_3
R_2	4, R_5
R_3	4, R_8
R_4	6, R_7
R_5	2, R_8
R_6	2, R_8
R_7	3, R_8

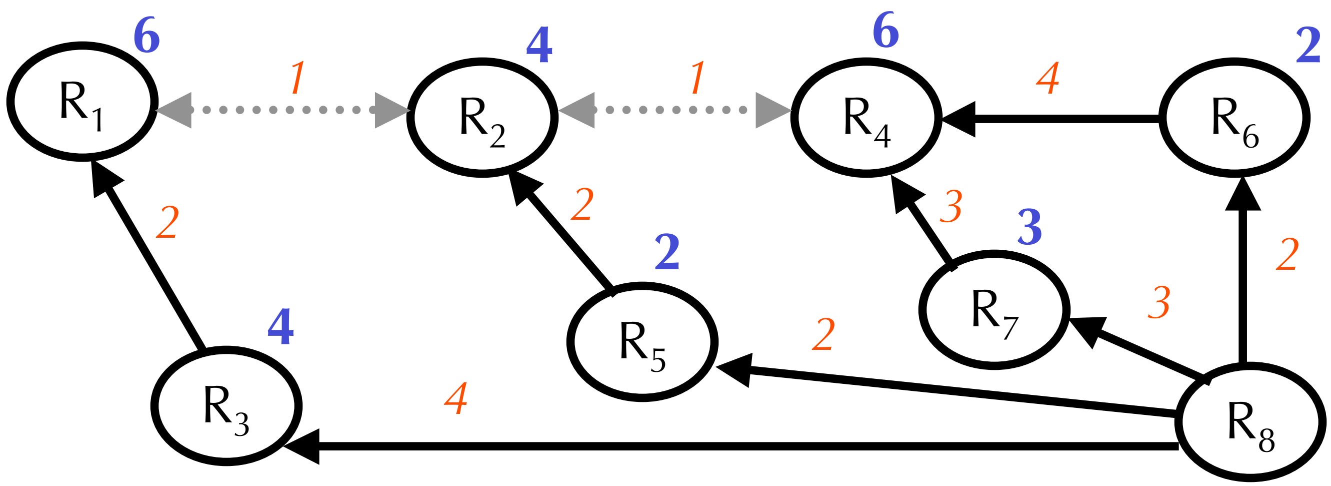


Repeat until no costs change

Technique1: Distributed Bellman-Ford Algorithm

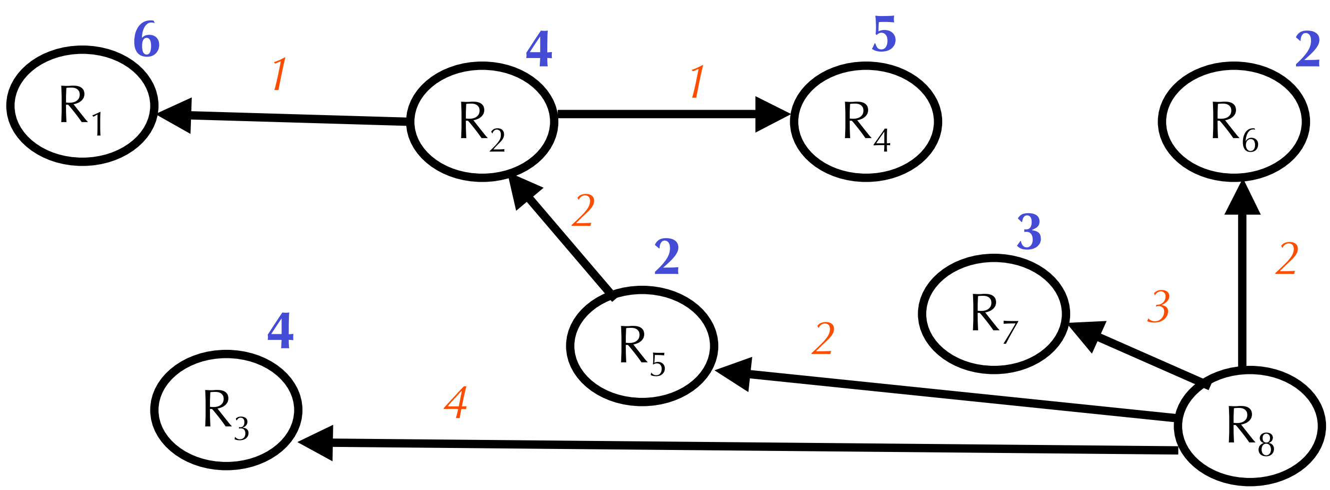
Example

R_1	6, R_3
R_2	4, R_5
R_3	4, R_8
R_4	6, R_7
R_5	2, R_8
R_6	2, R_8
R_7	3, R_8



Solution

R_1	5, R_2
R_2	4, R_5
R_3	4, R_8
R_4	5, R_2
R_5	2, R_8
R_6	2, R_8
R_7	3, R_8



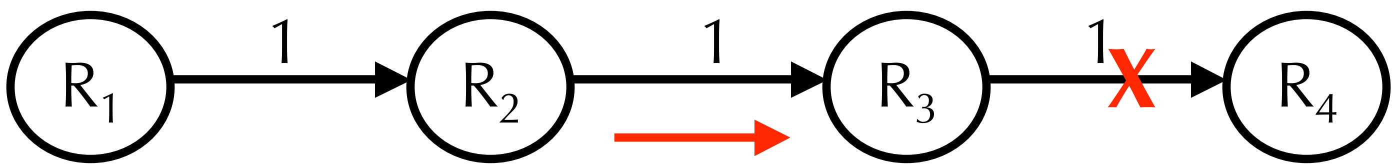
Distributed Bellman-Ford Algorithm

Questions:

1. How long will the algorithm take to stabilize?
2. How do we know that the algorithm always converges?
3. What happens when link costs change, or when routers/links fail?

A Problem with Bellman-Ford

"Bad news travels slowly"



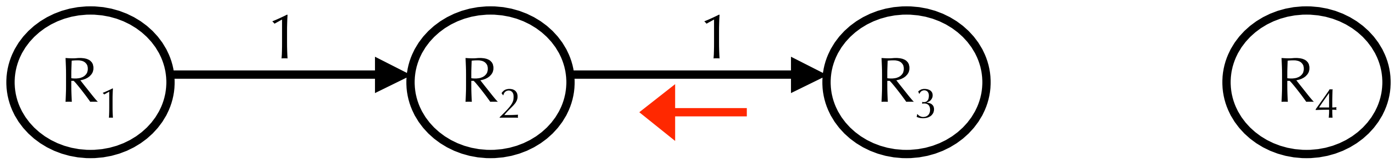
Consider the calculation of distances to R_4:

Time	R_1	R_2	R_3
0	3,R_2	2,R_3	1, R_4
1	3,R_2	2,R_3	**3,R_2**
2			
3			
4			

R_3 -- R_4 fails

A Problem with Bellman-Ford

“Bad news travels slowly”



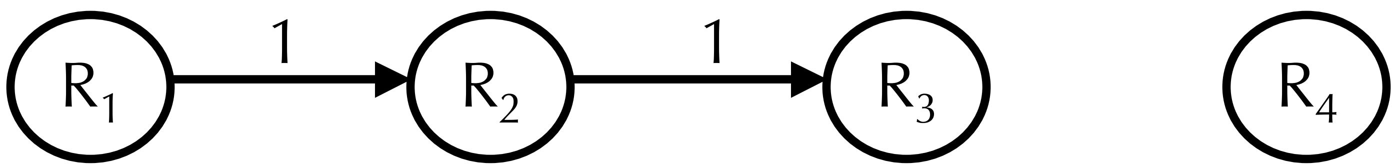
Consider the calculation of distances to R_4:

Time	R_1	R_2	R_3
0	3,R_2	2,R_3	1, R_4
1	3,R_2	2,R_3	**3,R_2**
2	3,R_2	**4,R_3**	3,R_2
3	**5,R_2**	4,R_3	**5,R_2**
4	5,R_2	**6,R_3**	5,R_2

… “Counting to infinity” …

R_3-R_4 fails

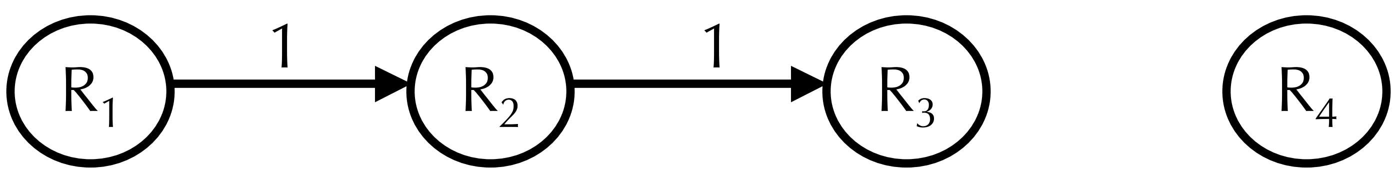
How are These Loops Caused?



- Observation 1:
  - R3's metric increases
- Observation 2:
  - R2 picks R3 as next hop to R4
  - But, the implicit path from R2 to R4 includes itself

Solutions to Counting to Infinity

- Set infinity = "some small integer" (e.g. 16). Stop when count = 16.
- Split Horizon: Because R_2 received lowest cost path from R_3, it does not advertise cost to R_3
- Split-horizon with poison reverse: R_2 advertises infinity to R_3



Comments on Bellman-Ford

- Asynchronous
- Works when some costs (i.e., weights) are negative, as long as there is no negative cost cycle.
  - *Why?*
- The graph may be directed (not in the distributed case)
- Small messages, small state at each router
  - No router has a complete image of the graph

Two Main Approaches

- Distance Vector Protocols
  - E.g., RIP (Routing Information Protocol)
  - Based on Distributed Bellman-Ford Algorithm
- Link State Protocols
  - E.g., OSPF (Open Shortest Path First)
  - Based on Dijkstra Algorithm

Link State Routing

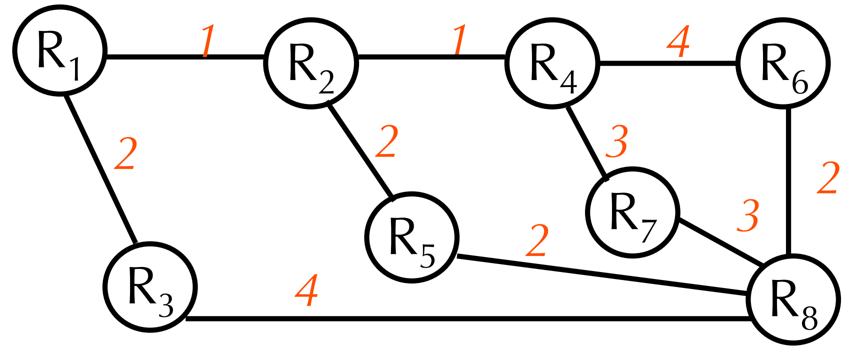
- Start condition
  - Each node assumed to know state of links to its neighbors
- Phase 1
  - Each node broadcasts its state to all other nodes
  - Reliable flooding mechanism
- Phase 2
  - Each node locally computes shortest paths to all other nodes from global state
  - Dijkstra's shortest path tree (SPT) algorithm

Phase 1: Link State Packets (LSPs)

- Periodically, each node creates a link state packet containing:
  - Node ID
  - List of neighbors and link cost
  - Sequence number
  - Time to live (TTL)
  - Node outputs LSP on all its links
- When a router receives a LSP from node
  - Keep most recent packet from each source
  - Forward to other routers
- All routers learn complete graph

Phase 2:
Dijkstra's Shortest Path First Algorithm

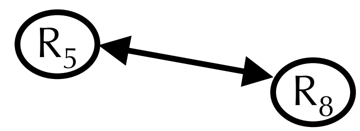
- Assumptions:
  - Costs are positive
  - Each router has the complete graph. *Is it scalable?*
- For each source, finds spanning tree routed on source router.



Dijkstra's Key Idea:
At each step, consider nodes with edges to nodes in set S; Pick the next closest node to destination and move it to S; update distances from destination

Step 1: $S = \{R_8\}$, $C = \{R_3, R_5, R_7, R_6\}$

Step 2: $S = \{R_8, R_5\}$, $C = \{R_3, R_7, R_6, R_2\}$

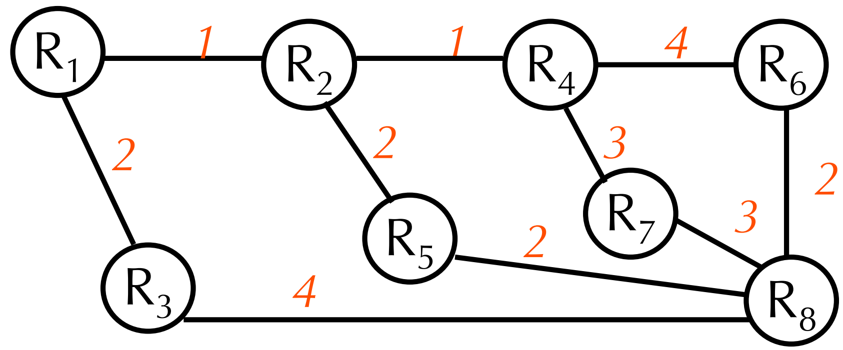


Set S: nodes where shortest path to destination is already known

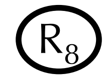
Set C: all nodes with direct edges to any node in S

Dijkstra's Key Idea:

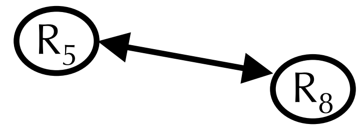
At each step, consider nodes with edges to nodes in set S; Pick the next closest node to destination and move it to S; update distances from destination



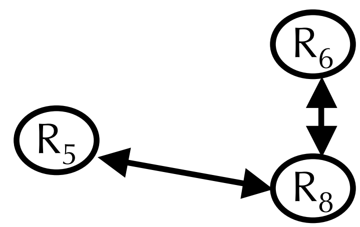
Step 1: $S = \{R_8\}$, $C = \{R_3, R_5, R_7, R_6\}$

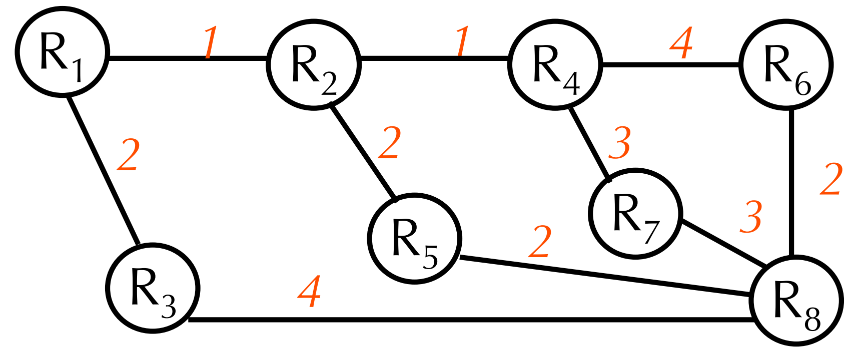


Step 2: $S = \{R_8, R_5\}$, $C = \{R_3, R_7, R_6, R_2\}$



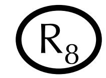
Step 3: $S = \{R_8, R_5, R_6\}$, $C = \{R_3, R_7, R_2, R_4\}$



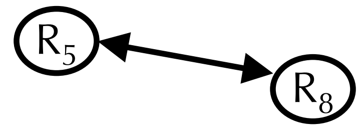


Dijkstra's Key Idea:
At each step, consider nodes with edges to nodes in set S; Pick the next closest node to destination and move it to S; update distances from destination

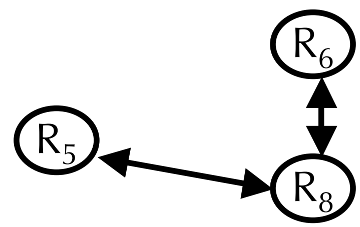
Step 1: S = $\{R_8\}$, C = $\{R_3, R_5, R_7, R_6\}$



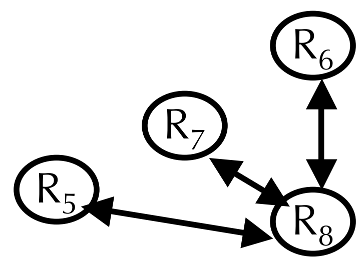
Step 2: S = $\{R_8, R_5\}$, C = $\{R_3, R_7, R_6, R_2\}$



Step 3: S = $\{R_8, R_5, R_6\}$, C = $\{R_3, R_7, R_2, R_4\}$



Step 4: S = $\{R_8, R_5, R_6, R_7\}$, C = $\{R_3, R_2, R_4\}$

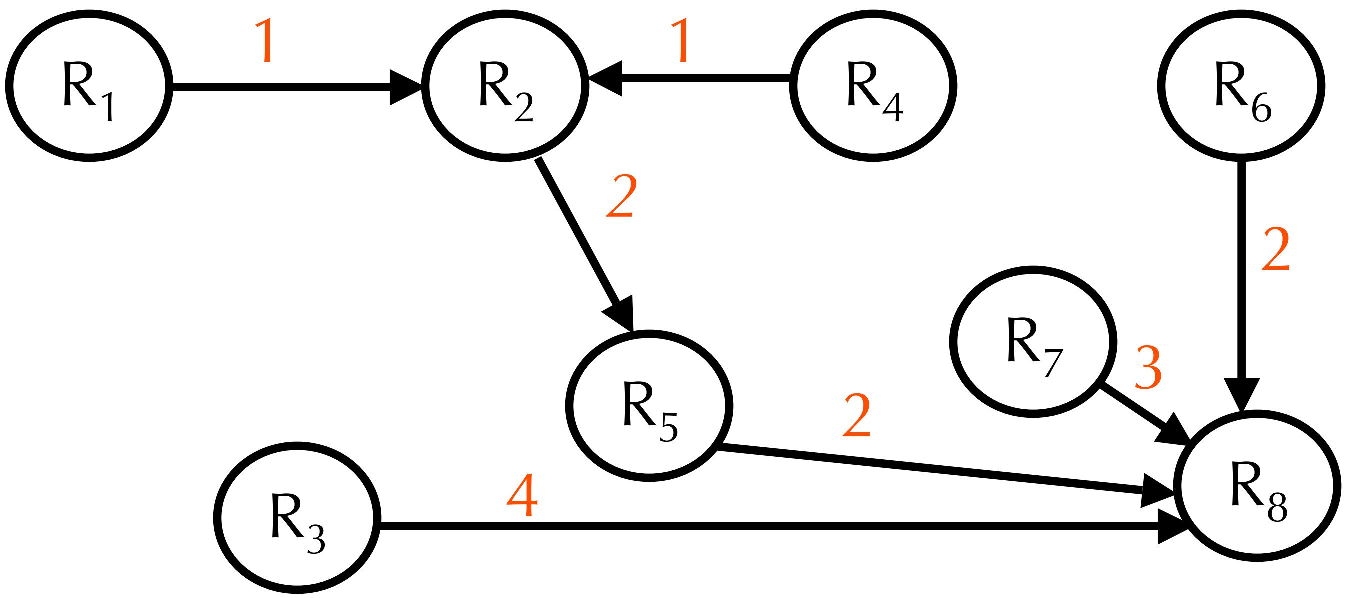


And so on…

Dijkstra's SPF Algorithm

Step 8: $S = \{R_8, R_5, R_6, R_7, R_2, R_1, R_4\}$,

$C = \{\}$.



OSPF optimizations

- Don’t send updates to all other routers
  - Elect a root router, send updates there
  - Root broadcasts link database to all routers
- Areas
  - Run routing algorithm separately in each area
  - Graph not propagated to other areas
  - Reduce state needed on each router
    - Operator needs to assign routers to areas

Summary: LS vs. DV

- Message size
  - Small in Link State (only state to neighbors)
  - Large in Distance Vector (costs to all destinations)
- Convergence speed
  - LS: faster – done once topology disseminated
- Space requirements
  - LS maintains entire topology
  - DV maintains only neighbor state
- Robustness:
  - Can be made robust since sources are aware of alternate paths
  - Incorrect calculation can spread to entire network

Summary: LS vs. DV

- Bottom line: no clear winner,
- Link State more prevalent in intra-domain routing
  - Protocol details
- (inter-domain uses BGP which is based on DV)