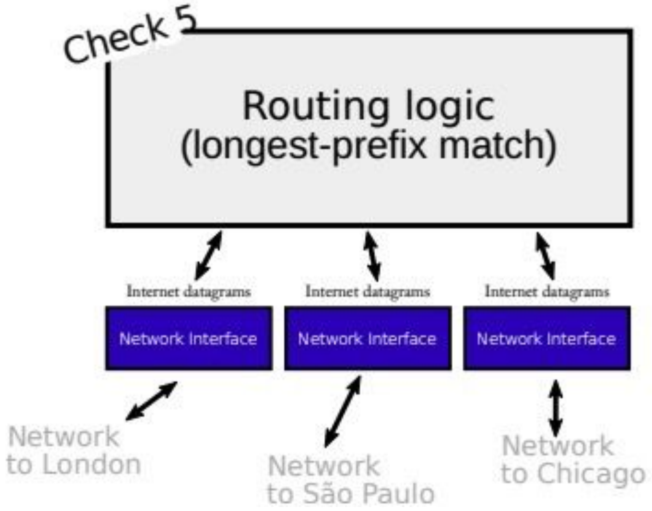
Routing

Previously:



- Routing tables tell computers where to send packets to their intended destination
- How do we build a routing table?

Exercise: Send Packets

- Each person has two cards
  - 1 has your router ID
  - Other note is your datagram (source: your router ID, some destination)
- Each person should end up with their corresponding destination datagram
- Can only pass notecards to your neighbors

General Approaches

Flooding: Send the packet across every link

Source Routing: source host controls the whole path

Distributed Algorithms: Each node makes a decision based on knowledge of the topology

- Bellman-Ford
- Dijkstra's

Flooding

Pros:

- + Does not require a routing table
- + Naturally will arrive at the destination using the shortest path

Cons:

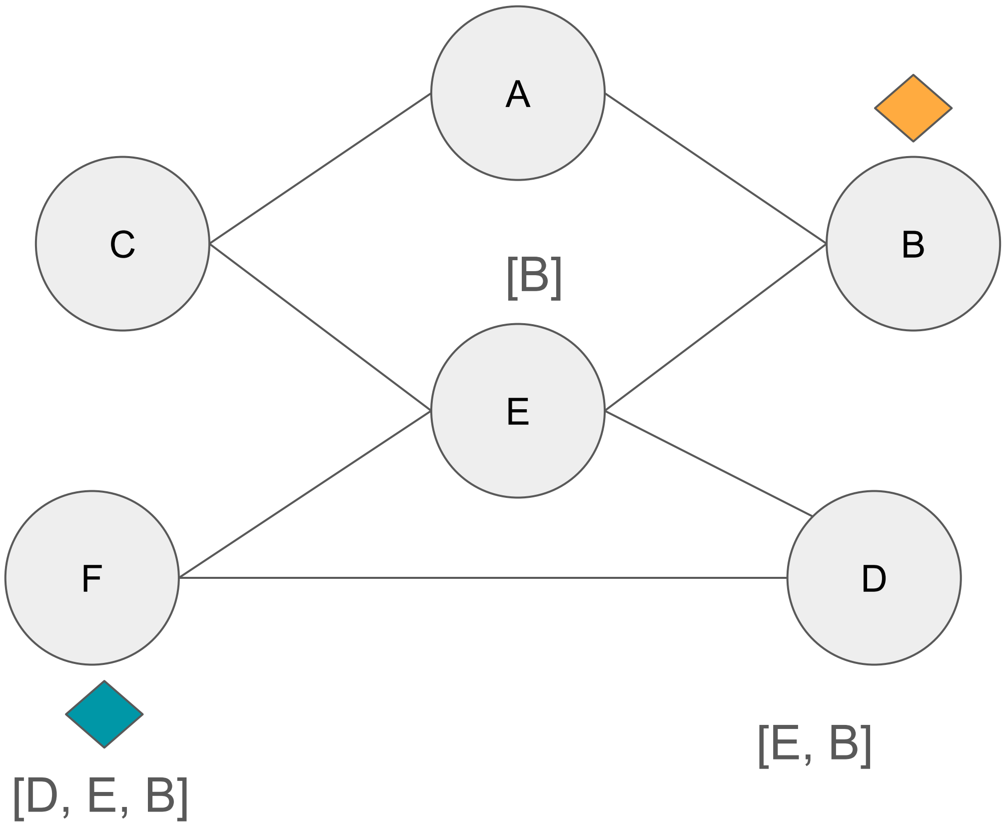
- - Inefficient
- - Possible Infinite Loops (need TTL)
- - Sending excess information

Source Routing

- “Bird’s eye view” source router already knows the intended path
- At each step, use information already provided to send to next hop

Cons:

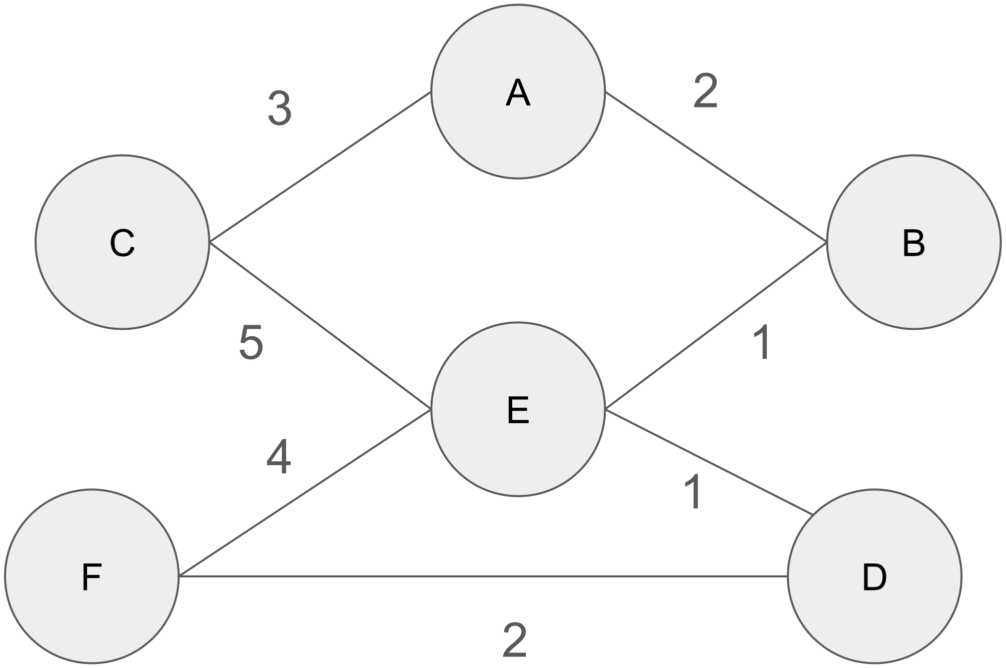
- Reveals the network topology
- Larger overhead



Distributed Routing Algorithms: Bellman Ford

1. Initialize distance for each node as infinity (and itself as 0)
2. Assume nodes know the cost for every directly connected node
3. After T seconds (ex: 30), send update to every neighboring node about current distance to each node
4. If newly advertised cost is less than what is currently in routing table, update
5. Go back to step 3

Example: Bellman Ford, Cost to F

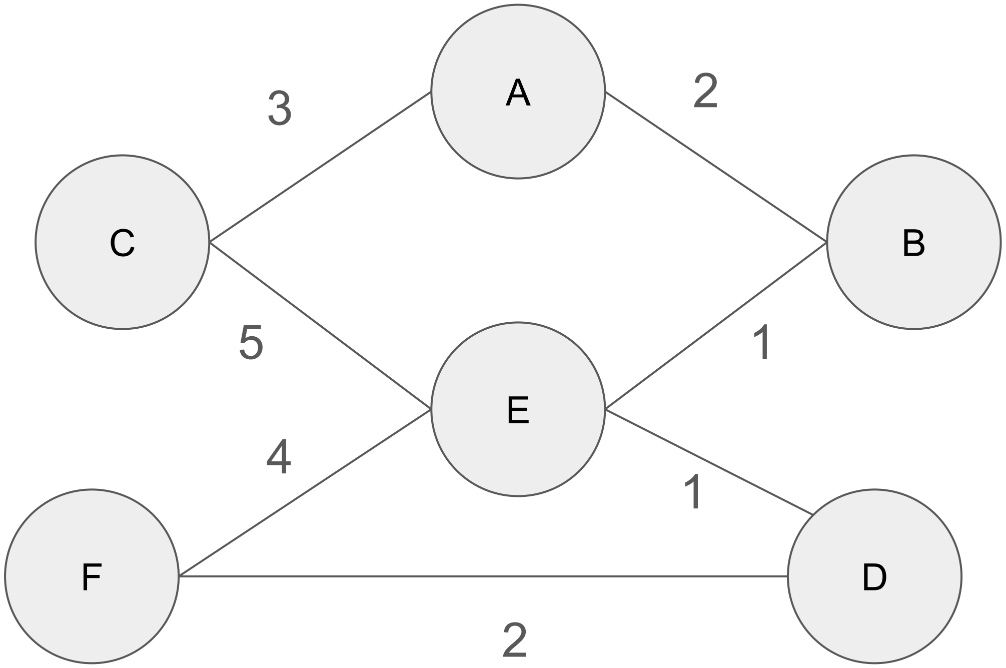


A	B	C	D	E
∞	∞	∞	∞	∞
∞	∞	∞	2	4
∞	5, E	9, E	2	3, D
7, B	4, E	8, E	2	3, D
6, B	4, E	8, E	2	3, D
6, B	4, E	8, E	2	3, D

Distributed Routing: Dijkstra's

1. Mark all nodes as unvisited, and set cost for current node as 0
2. Find current set of all unvisited neighbors from previously added nodes
3. Add the node to visited set that is the shortest known cost (will not have to check that node again)
4. Repeat from 2 until all nodes are visited

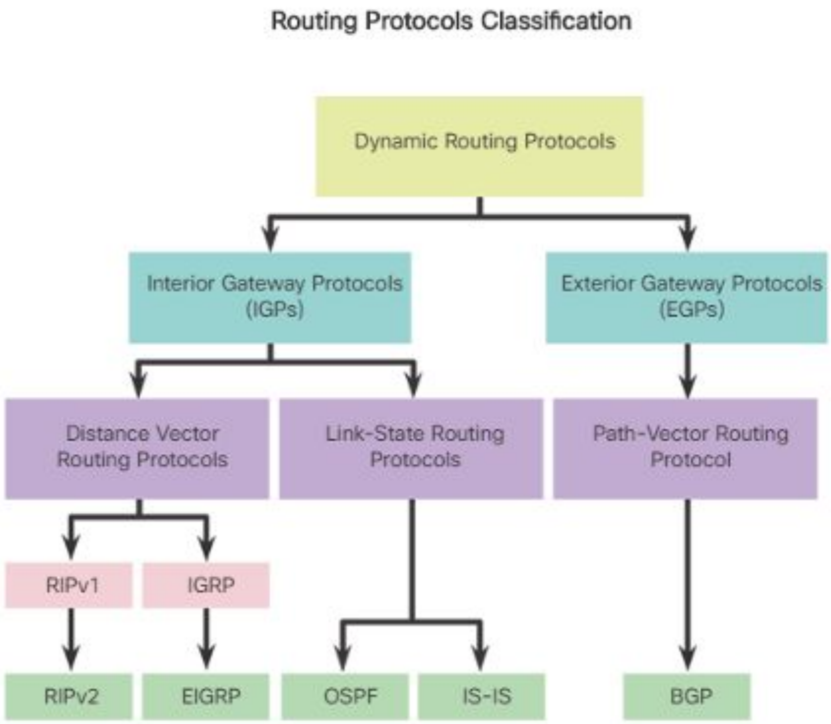
Example: Dijkstra's, Best path from B to C



Shortest Paths	Candidates	Added
B	A, E	E
BE (1)	A, D, F, C	D
BED (2)	A, F, C	A
BED (2) BA (2)	F, C	F
BEDF (4) BA (2)	C	C
BEDF (4) BAC (5)	NONE	NONE

Routing Protocols

- **Distance Vector**: sends their entire routing table to directly connected neighbors
- **Link State**: Every router shares information about their neighbors, then independently calculate the best path to each destination
- **Path-Vector**: contains the destination network, next router, and path to reach the destination

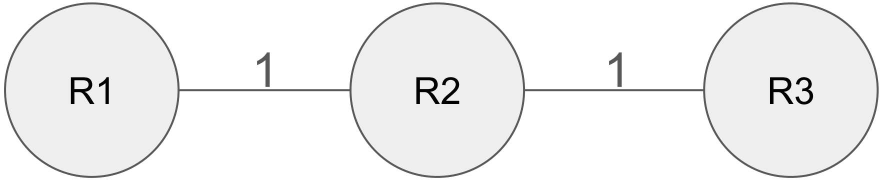


RIP: Routing Information Protocol

- Distance vector with hop count as the metric
- Uses the Bellman Ford Algorithm
- Continuously looks for updates, each node only knows about it’s neighbor

RIP

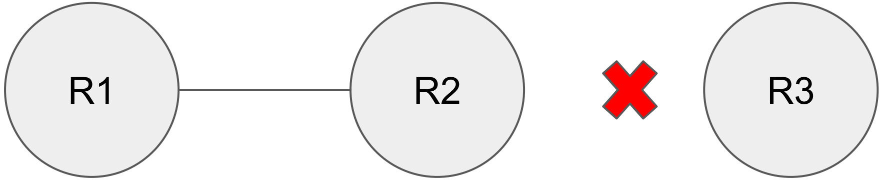
- What happens when a link is cut?



R1	R2
2, R2	1, direct

RIP

- What happens when a link is cut?
- (RIP has a hop count limit of 15)



R1	R2
2, R2	1, direct
3, R2	2, R1
4, R2	3, R2
∞	∞

OSPF: Open Shortest Path First

- Link state
- Only updates when there is a change in the topology
- Can use different metrics (link rate, congestion, etc)
- Each router calculates Dijkstra's Algorithm independently
- Each node has a full picture of the network

RIP vs OSPF

RIP	OSPF
Bellman Ford Can be slow to converge Gets updates from neighbors Uses hop count as a metric Continuously sends updates Fully distributed Better in smaller networks	Dijkstra's Faster Converge Requires knowledge of full topology Uses link rate as a metric Updates only on topology changes Calculates shortest path independently Can be used in large networks