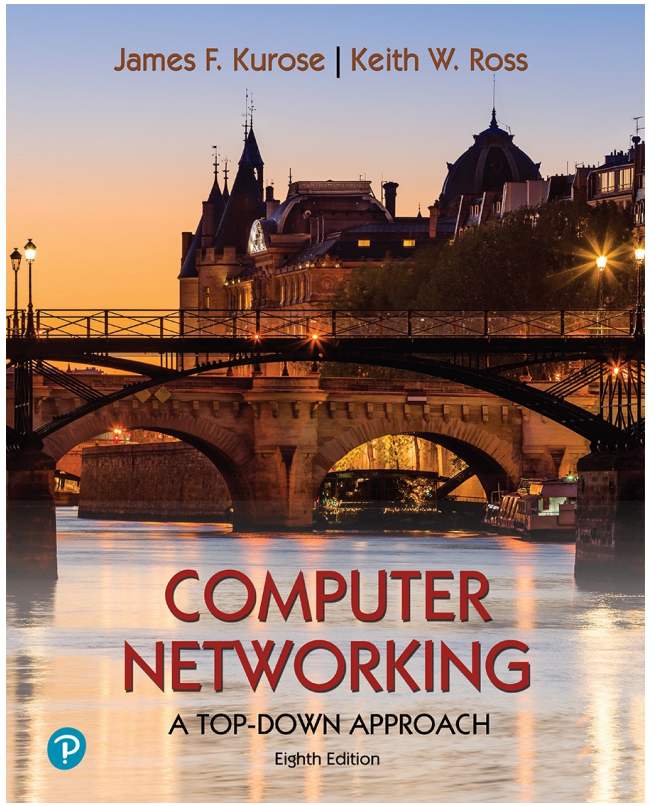
Chapter 5
Network Layer: Control Plane

Yaxiong Xie

Department of Computer Science and Engineering
University at Buffalo, SUNY



Computer Networking: A Top-Down Approach
8th edition
Jim Kurose, Keith Ross
Pearson, 2020

Adapted from the slides of the book's authors

Network layer control plane: our goals

- understand principles behind network control plane:
  - traditional routing algorithms
  - SDN controllers
  - network management, configuration
- instantiation, implementation in the Internet:
  - OSPF, BGP
  - OpenFlow, ODL and ONOS controllers
  - Internet Control Message Protocol: ICMP
  - SNMP, YANG/NETCONF

Network layer: "control plane" roadmap

- introduction
- routing protocols
  - link state
  - distance vector
- intra-ISP routing: OSPF
- routing among ISPs: BGP
- SDN control plane
- Internet Control Message Protocol

- network management, configuration
  - SNMP
  - NETCONF/YANG

Network-layer functions

- forwarding: move packets from router's input to appropriate router output

data plane

- routing: determine route taken by packets from source to destination

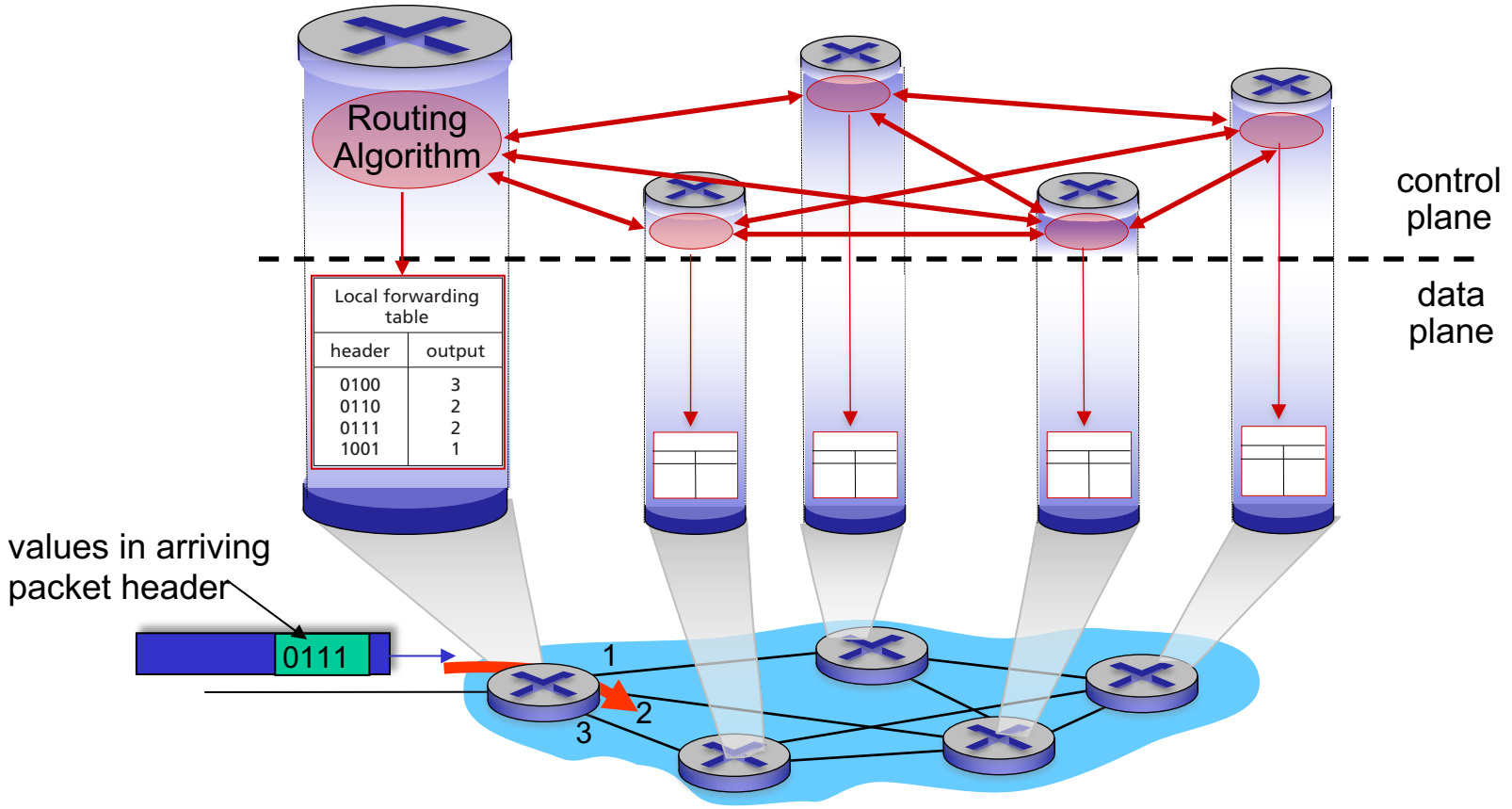
control plane

Two approaches to structuring network control plane:

- per-router control (traditional)
- logically centralized control (software defined networking)

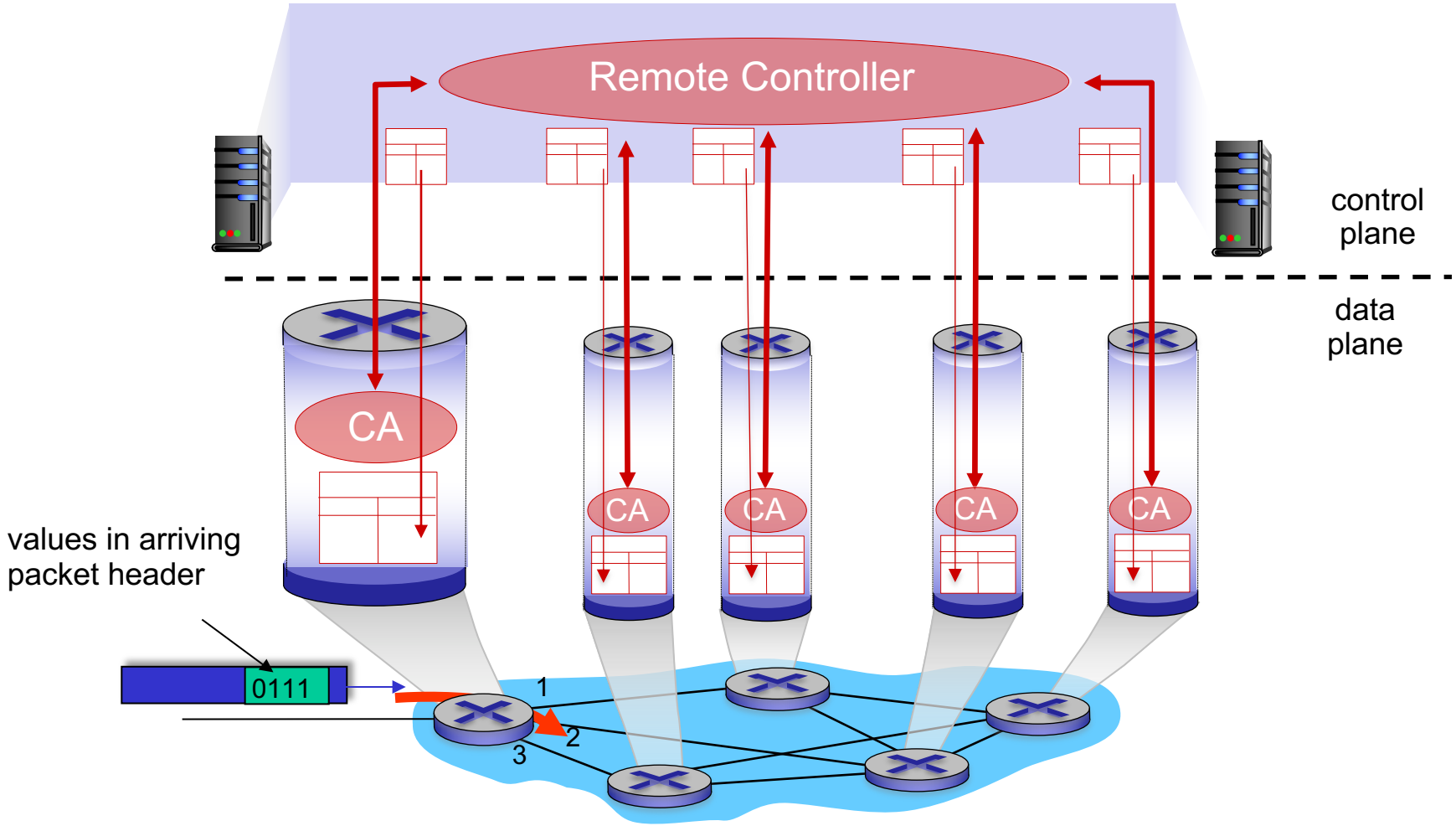
Per-router control plane

Individual routing algorithm components *in each and every router* interact in the control plane

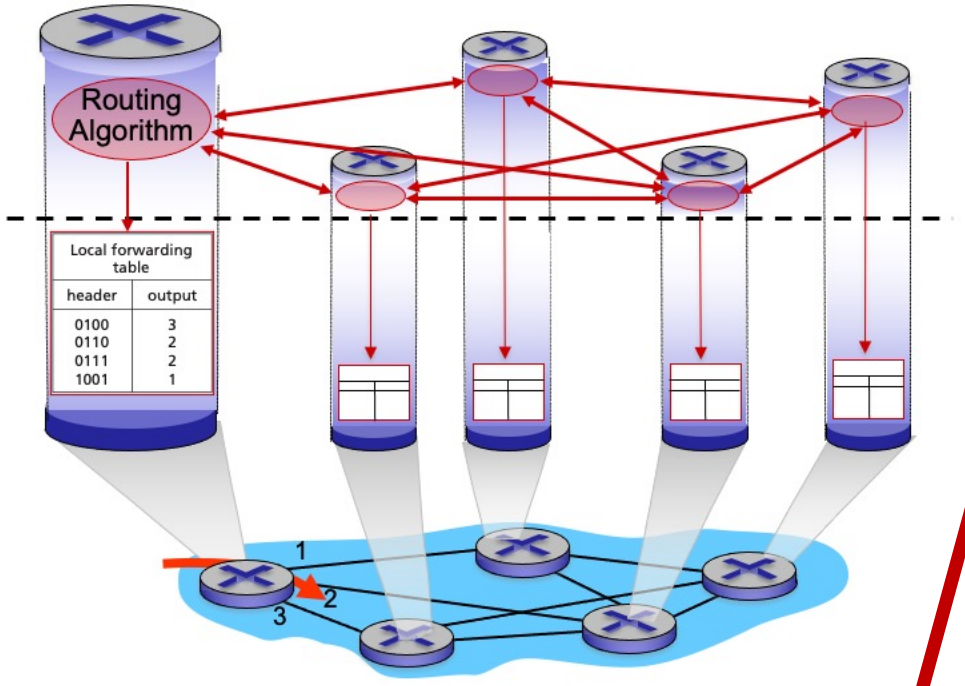


Software-Defined Networking (SDN) control plane

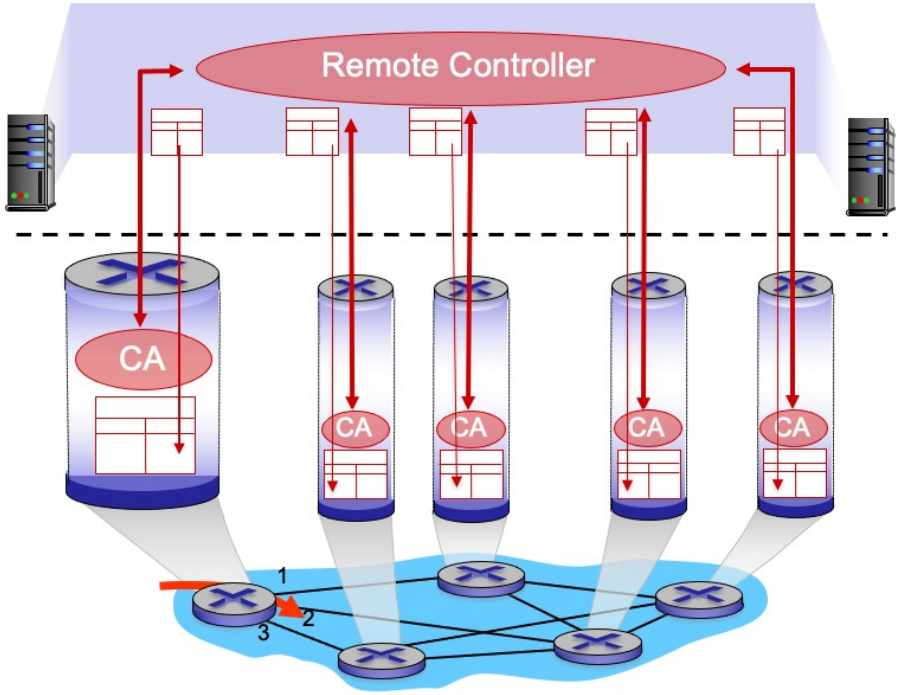
Remote controller computes, installs forwarding tables in routers



Per-router control plane



SDN control plane



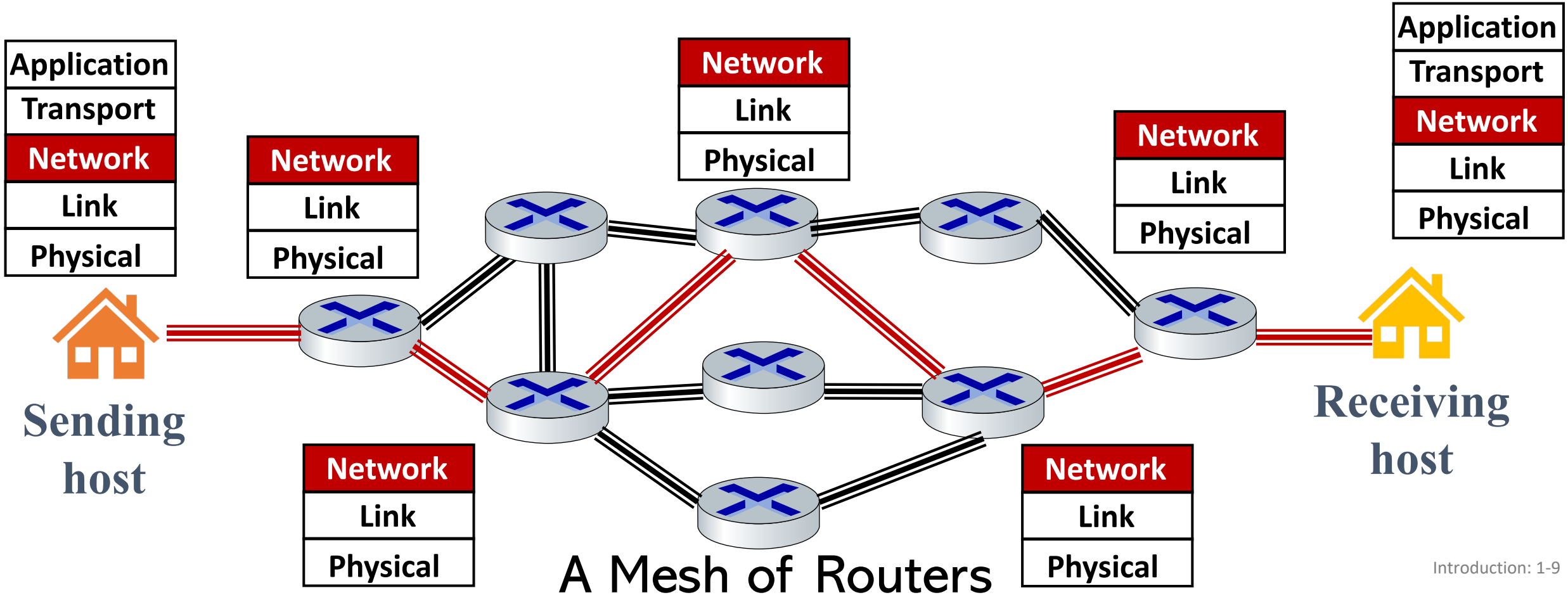
Network layer: “control plane” roadmap

- introduction
- **Per-Router** routing protocols
  - link state
  - distance vector
- intra-ISP routing: OSPF
- routing among ISPs: BGP
- SDN control plane
- Internet Control Message Protocol

- network management, configuration
  - SNMP
  - NETCONF/YANG

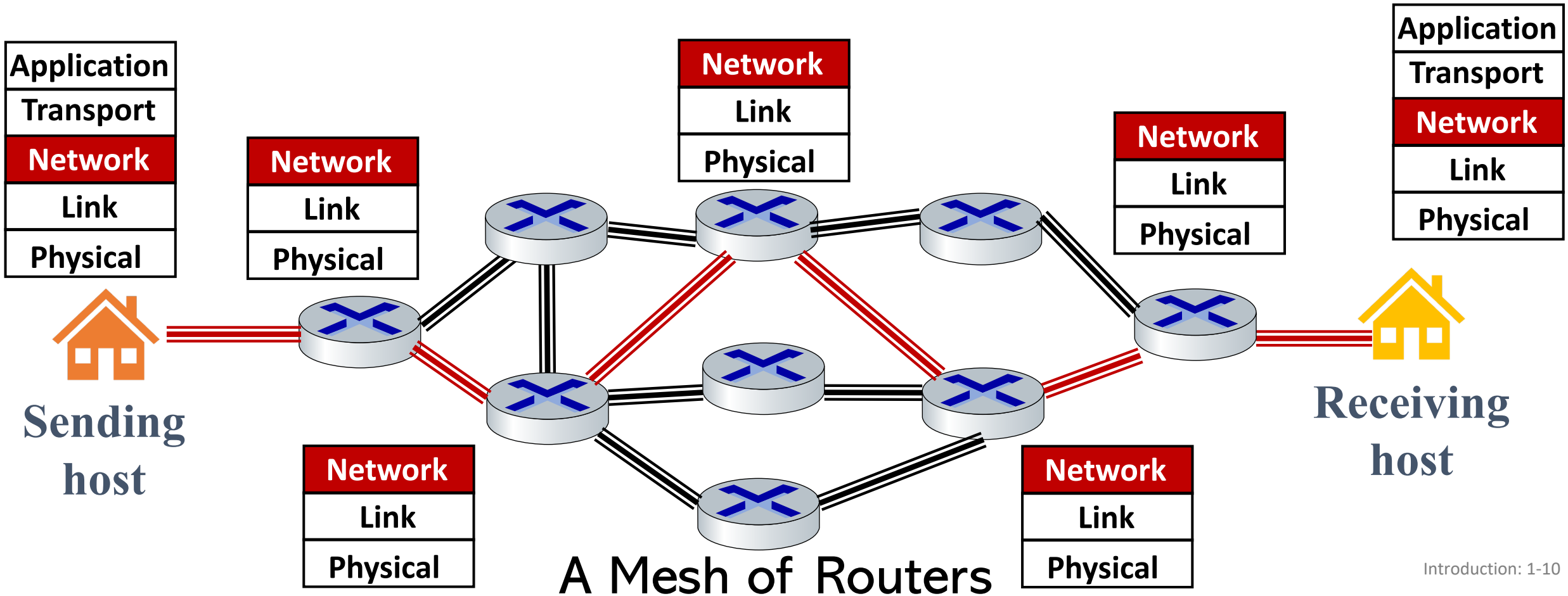
Goal of routing protocols

Determine "good" paths (equivalently, routes), from sending hosts to receiving host, through network of routers



Goal of routing protocols

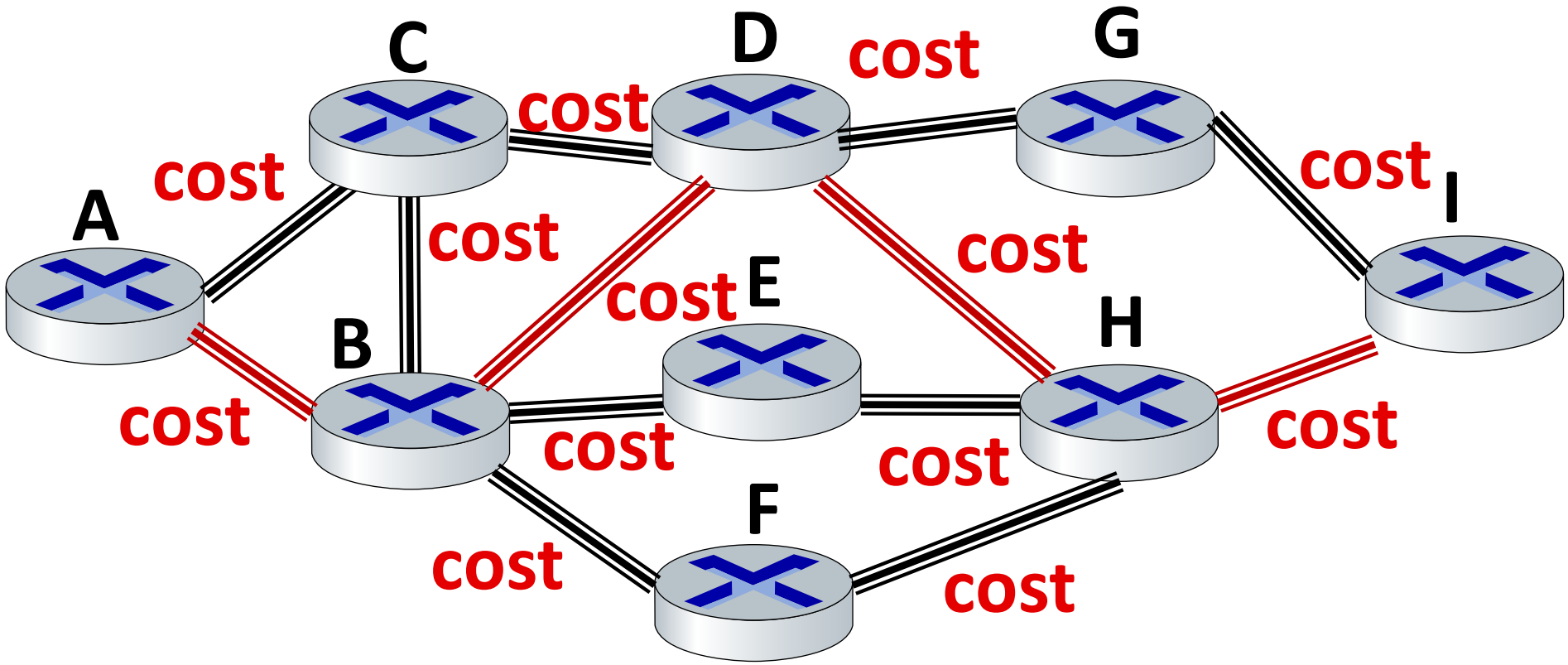
What is a "good" path?



Goal of routing protocols

What is a "good" path? **Path with the smallest cost**

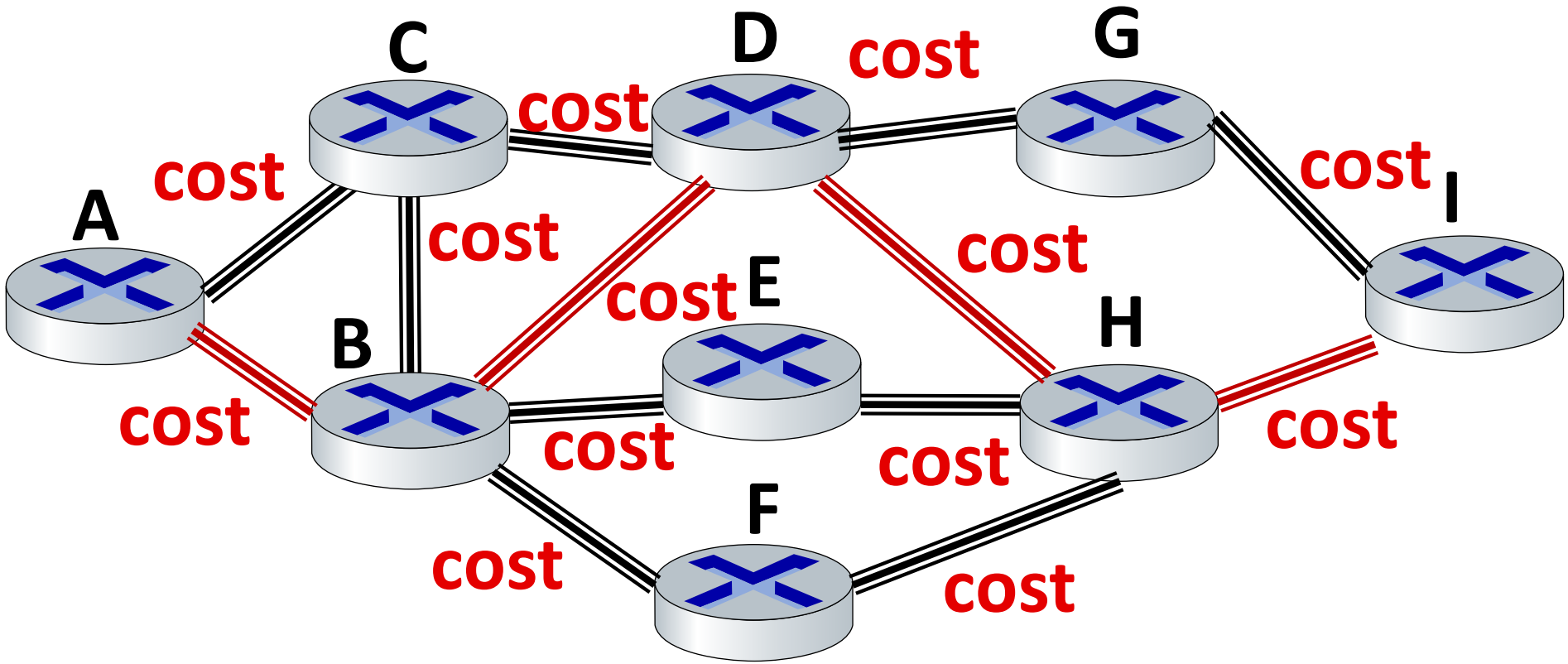
cost = 1 **-> Path with the smallest number of hops**



Goal of routing protocols

What is a "good" path? **Path with the smallest cost**

cost = $delay$ **-> Path with the lowest latency**



Goal of routing protocols

What is a "good" path? **Path with the smallest cost**

Example: A -> I

A->C->D->G->I

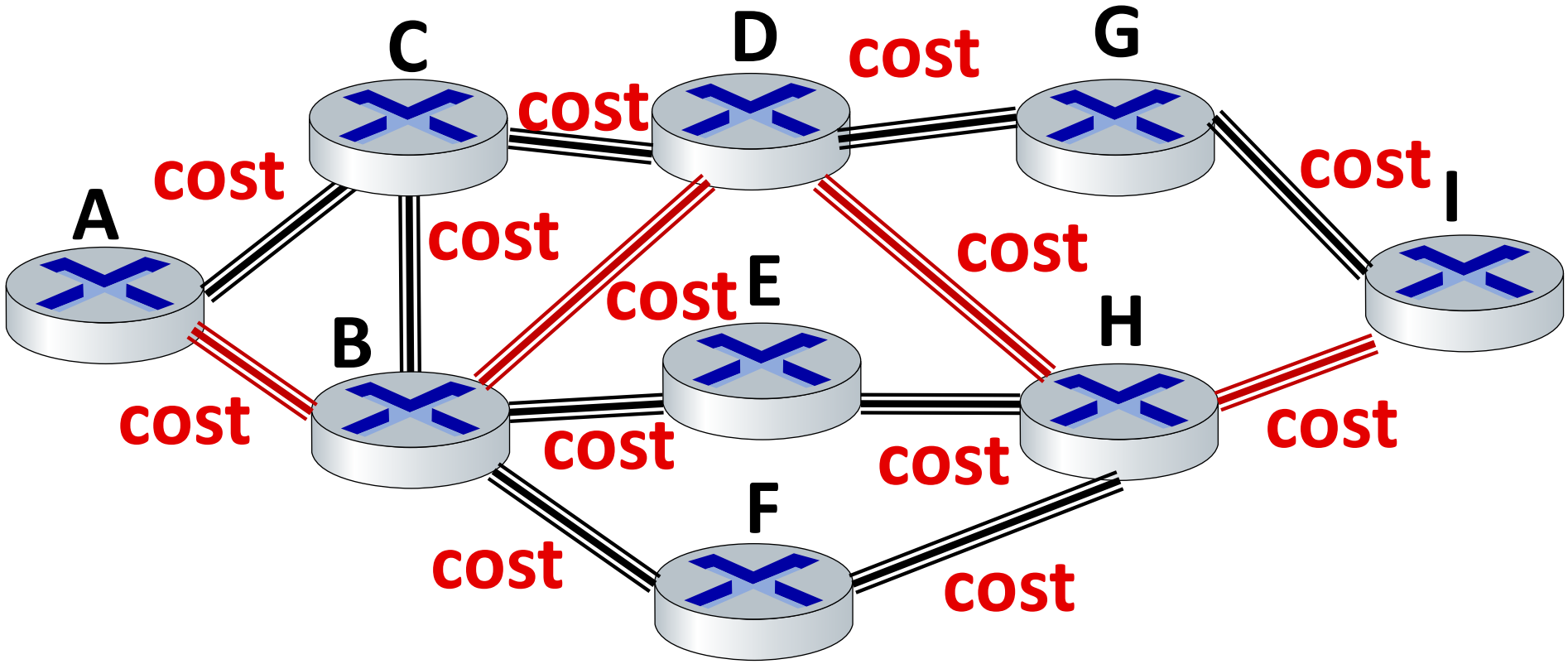
A->B->D->G->I

A->B->E->H->I

A->C->B->E->H->I

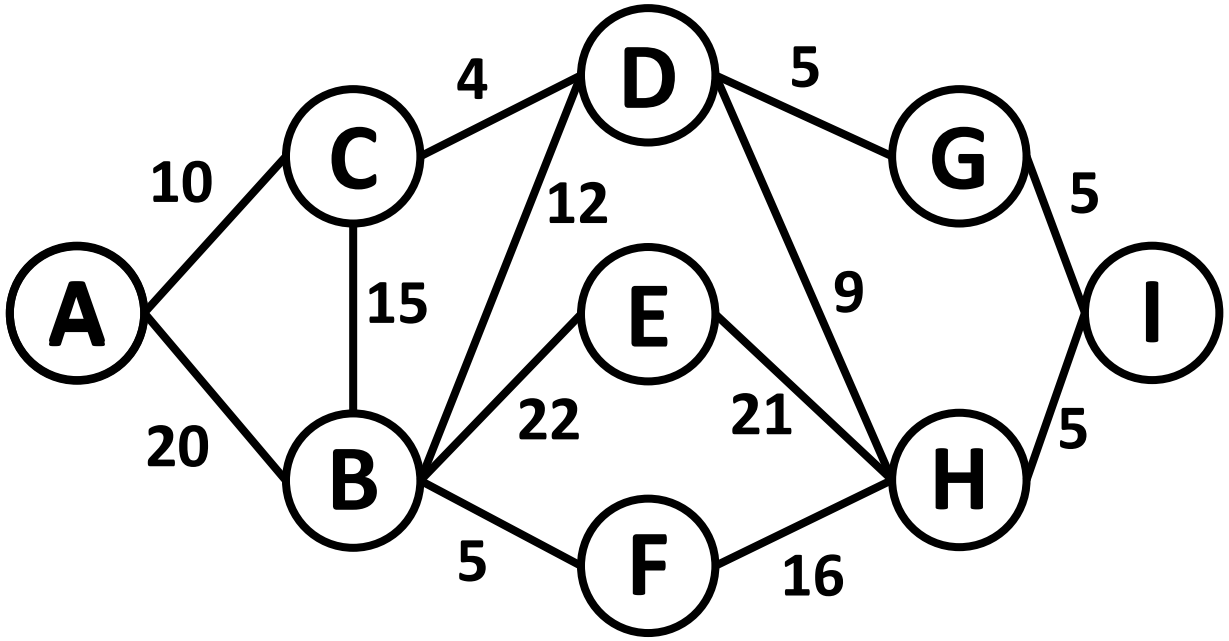
A->B->F->H->I

A->B->D->H->I



Goal of routing protocols

What is a "good" path?



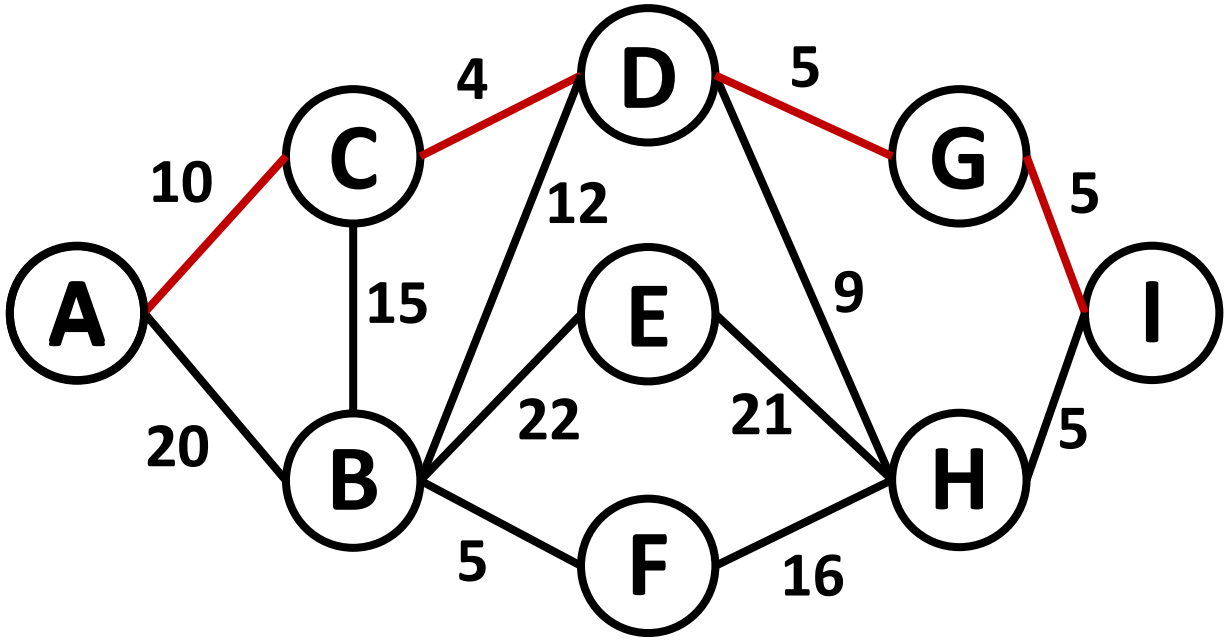
Path with the smallest cost

Mathematical Problem:
Find the shortest path in this graph

Example: Router A:
Find the shortest path to all the other routers inside the network

Path to Routing Tables

What is a "good" path?



Path with the smallest cost

Router A->I: **A->C->D->G->I**

IP Address Range	Interface
200.23.16.0/23	B
200.23.18.0/23	B
IP of Router I	C

Routing Challenges

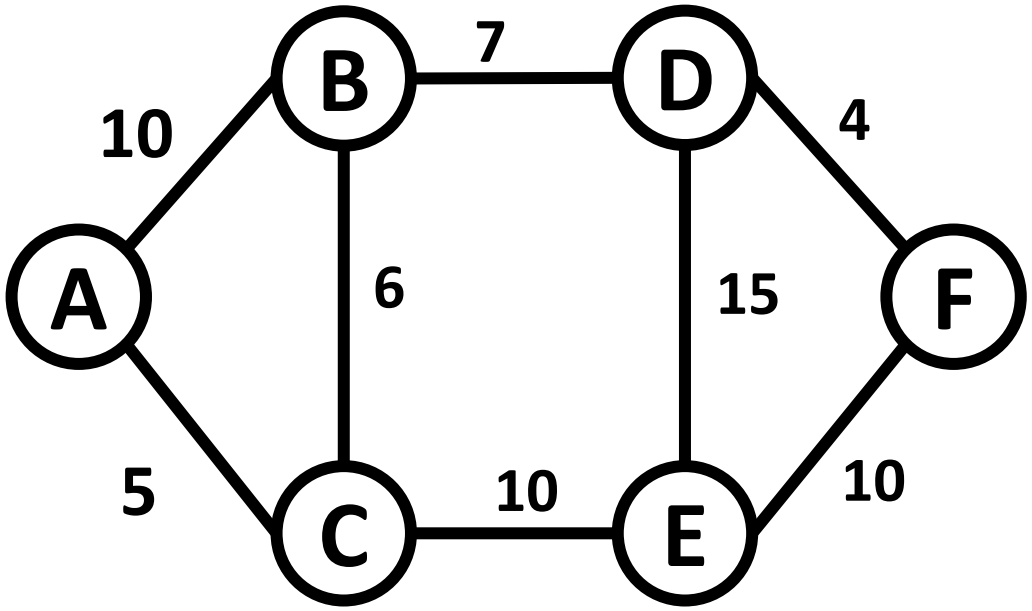
- How to choose the best path
  - Defining “best” can be tricky
- How to scale to millions of users?
  - Minimizing control messages and routing table size
- How to adapt to failures or changes?
  - Node and link failures, plus message loss

Network layer: "control plane" roadmap

- introduction
- **Per-Router** routing protocols
  - link state
  - distance vector
- intra-ISP routing: OSPF
- routing among ISPs: BGP
- SDN control plane
- Internet Control Message Protocol

- network management, configuration
  - SNMP
  - NETCONF/YANG

High-Level Idea: A Graph Theory Problem



Topology of the whole network

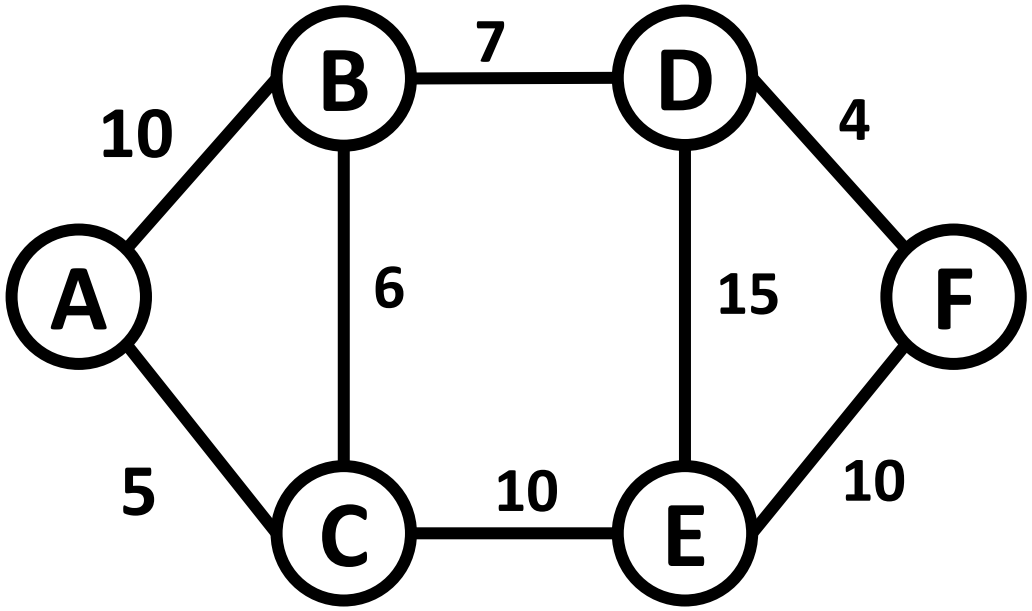
Suppose

- Calculating routing for node A
- Node A know the full topology of the whole network
  - Which can be described as a graph

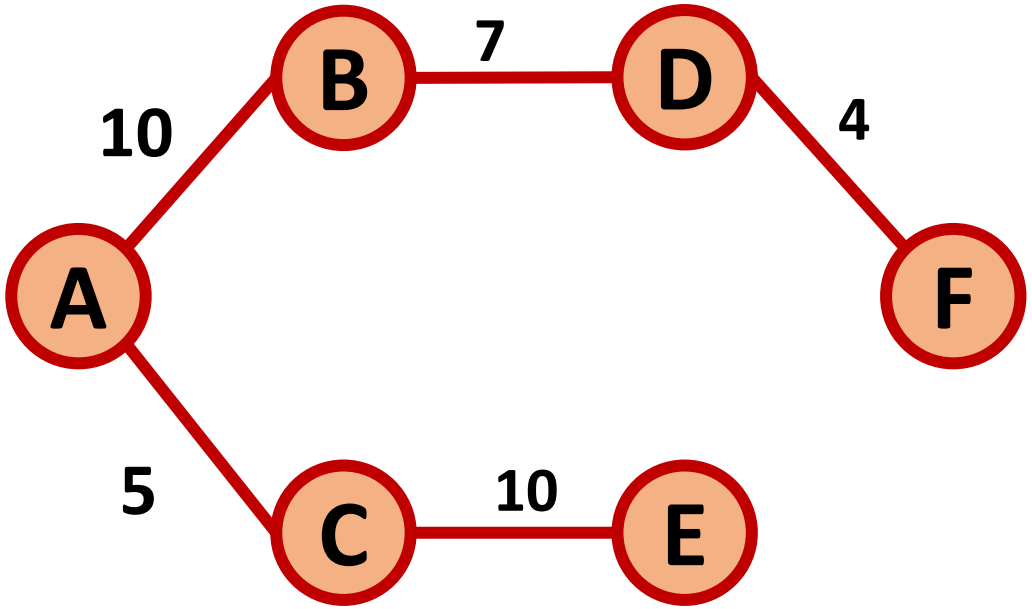
Problem: Finding the shortest path from node A to all other nodes

Mathematically, this problem can be solved by graph theory!

High-Level Idea: A Graph Theory Problem

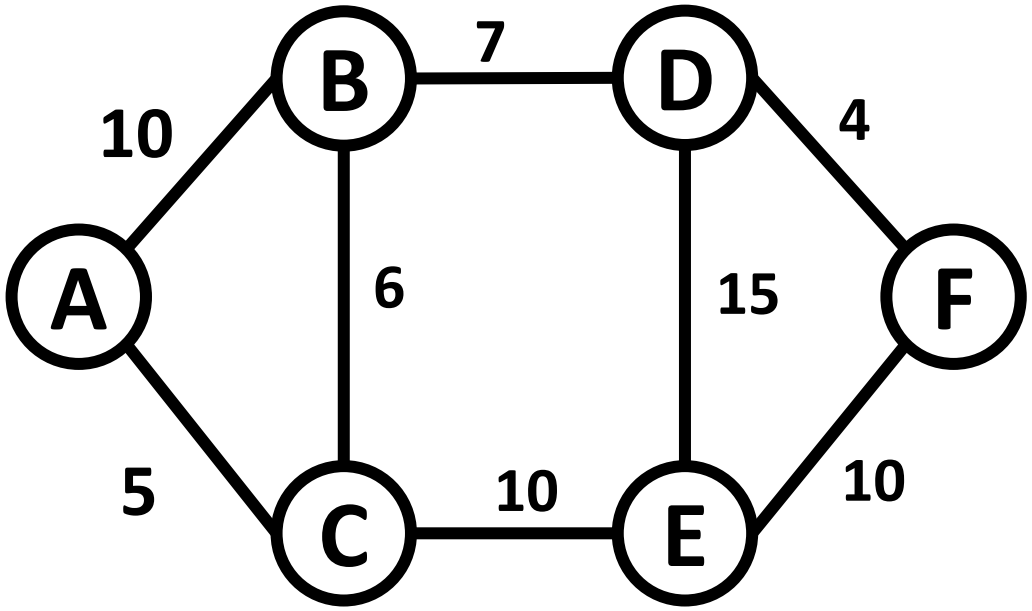


Topology of the whole network



Mathematically, this problem can be solved by graph theory!

High-Level Idea: A Graph Theory Problem

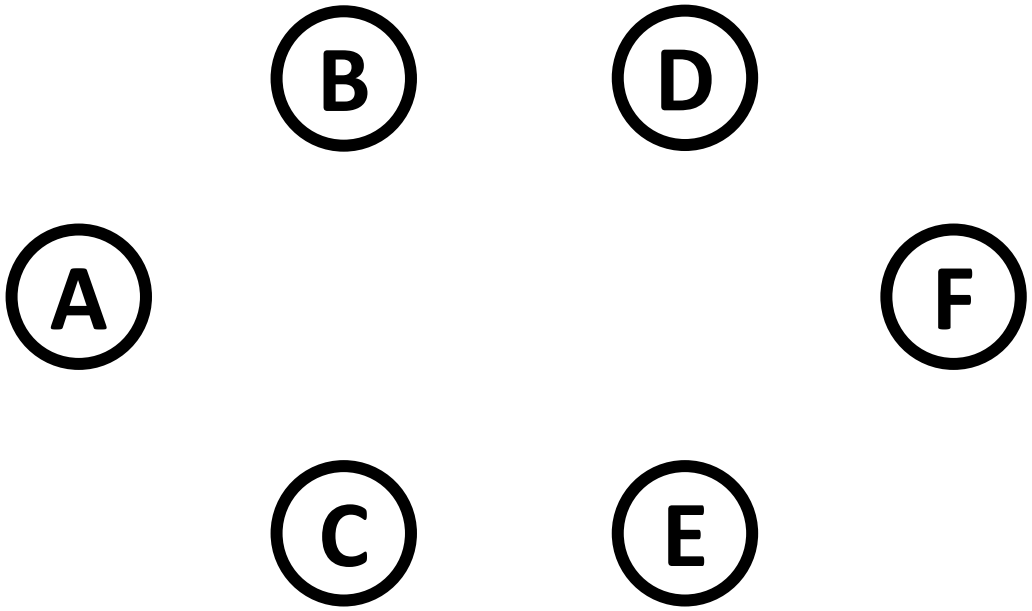


Topology of the whole network

How does node A get the whole topology of network?

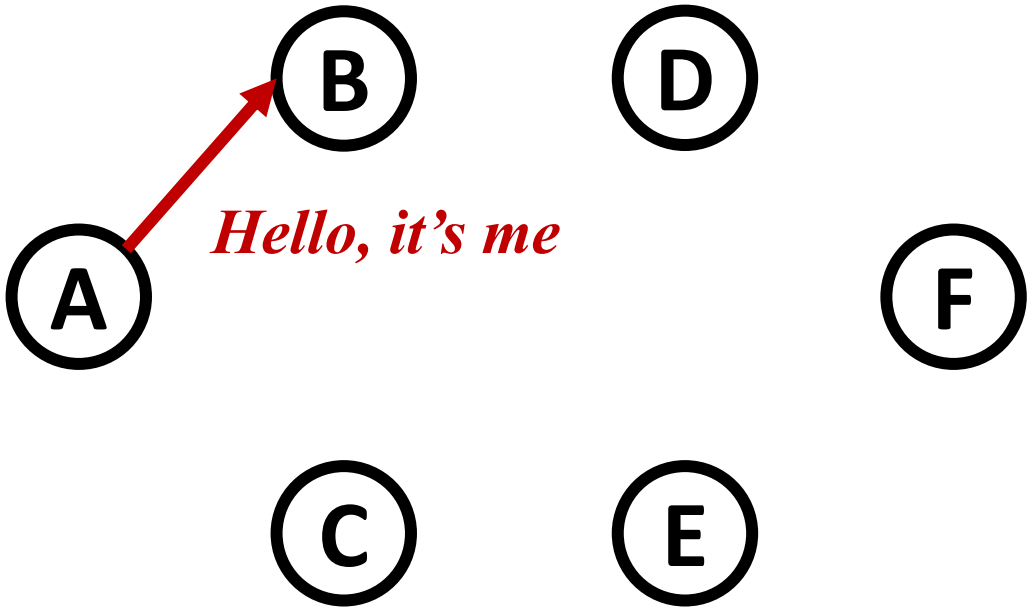
Mathematically, this problem can be solved by graph theory!

Obtain the Topology: Discover your Neighbor



Topology of the whole network

Obtain the Topology: Discover your Neighbor

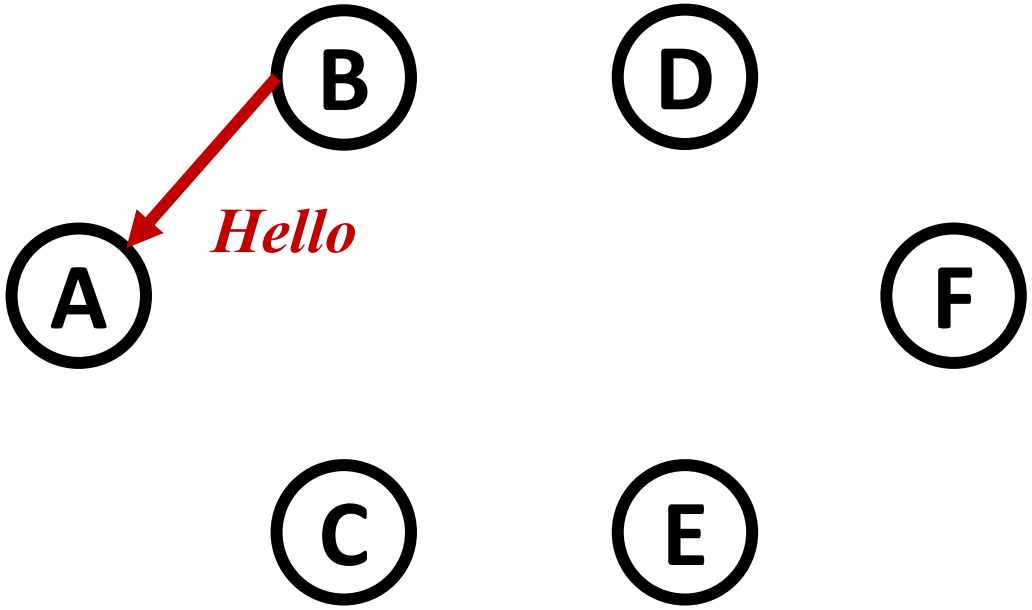


Topology of the whole network

Node A sends a ***hello*** message to Node B

If Node B is the neighbor of Node A and Node B is running the same routing protocol

Obtain the Topology: Discover your Neighbor



Topology of the whole network

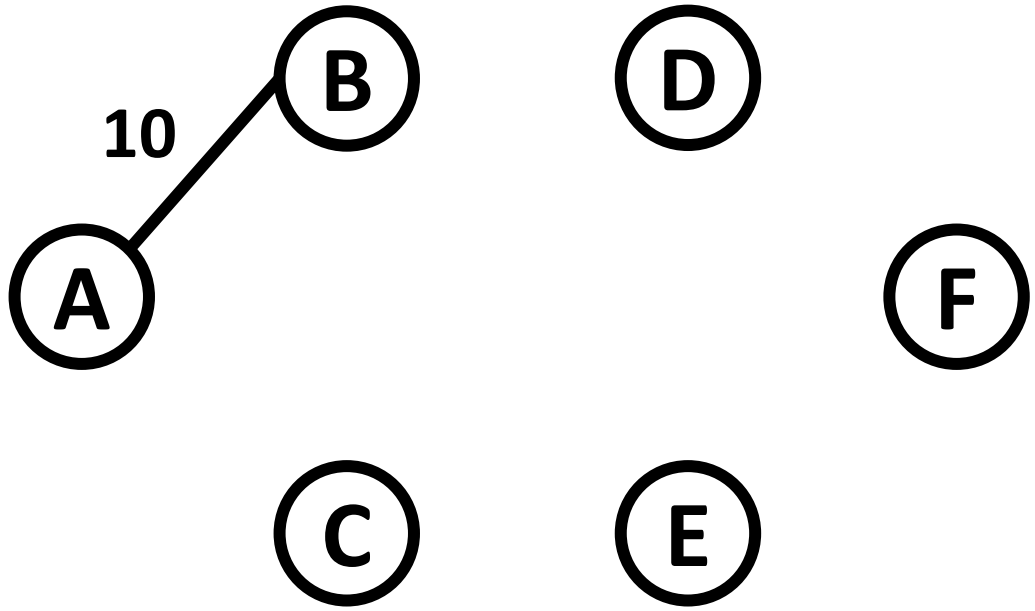
Node A sends a ***hello*** message to Node B

If Node B is the neighbor of Node A and Node B is running the same routing protocol

Node B sends a ***hello*** message back to Node A

Neighbor discovery is usually done by exchanging hello messages on directly connected links

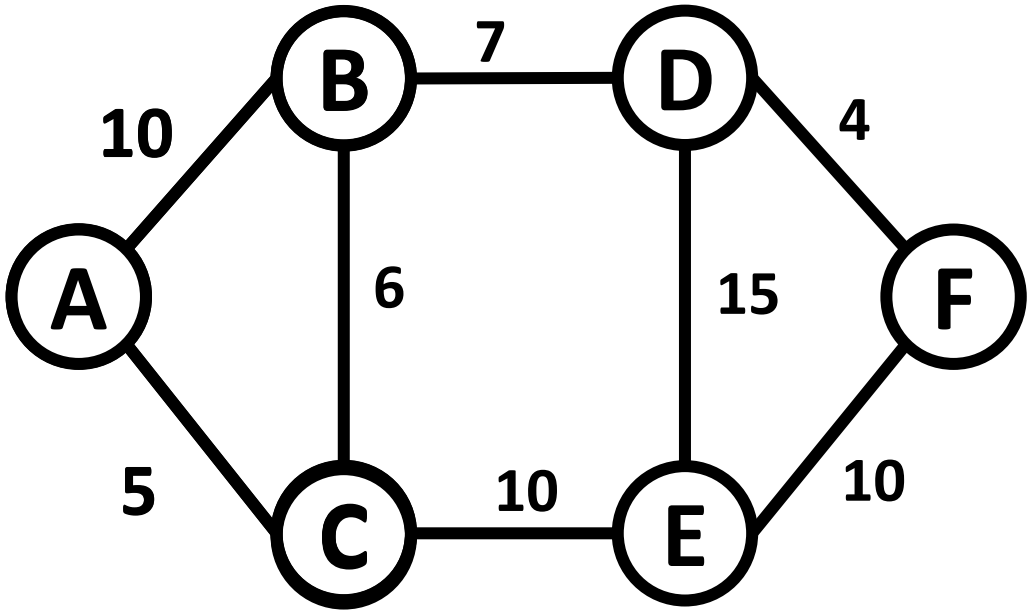
Obtain the Topology: Defining the Cost



Topology of the whole network

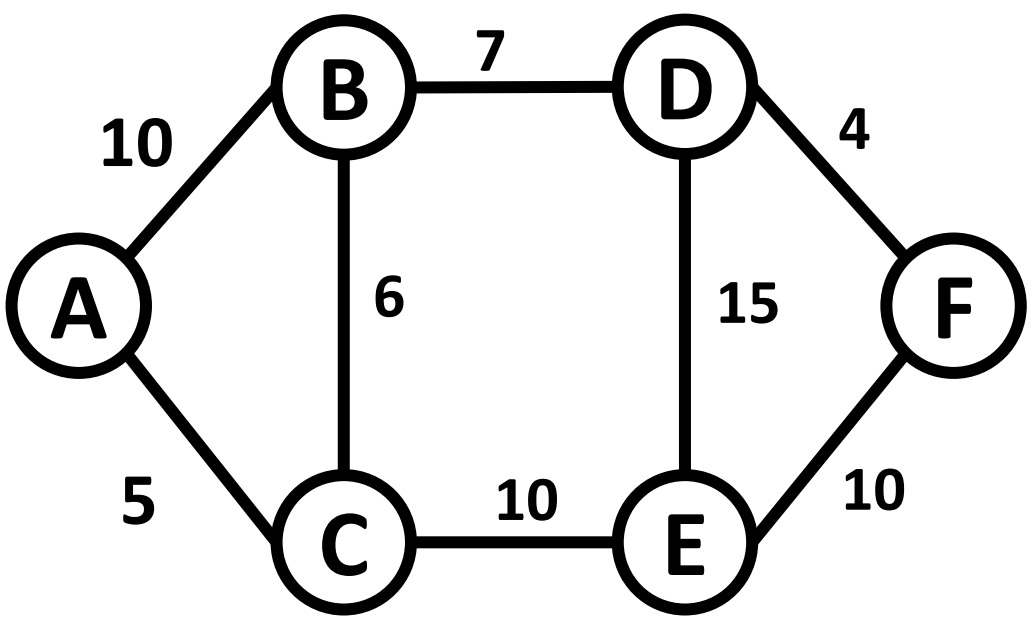
- Manually configured by operator
  - Cost from A->B is 10
- Estimated by routers based on bandwidth, latency, loss

Obtain the Topology: Link State

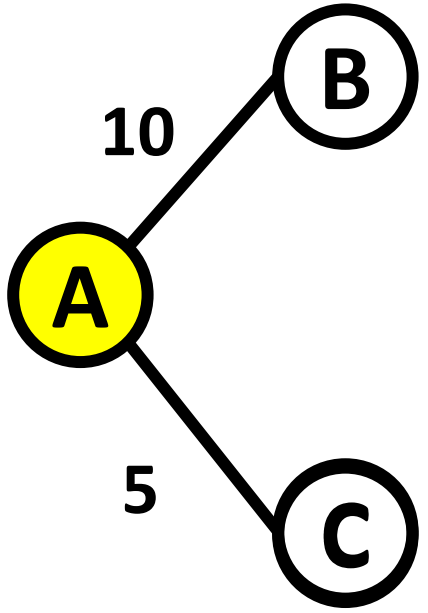


Topology of the whole network

Obtain the Topology: Link State



Topology of the whole network

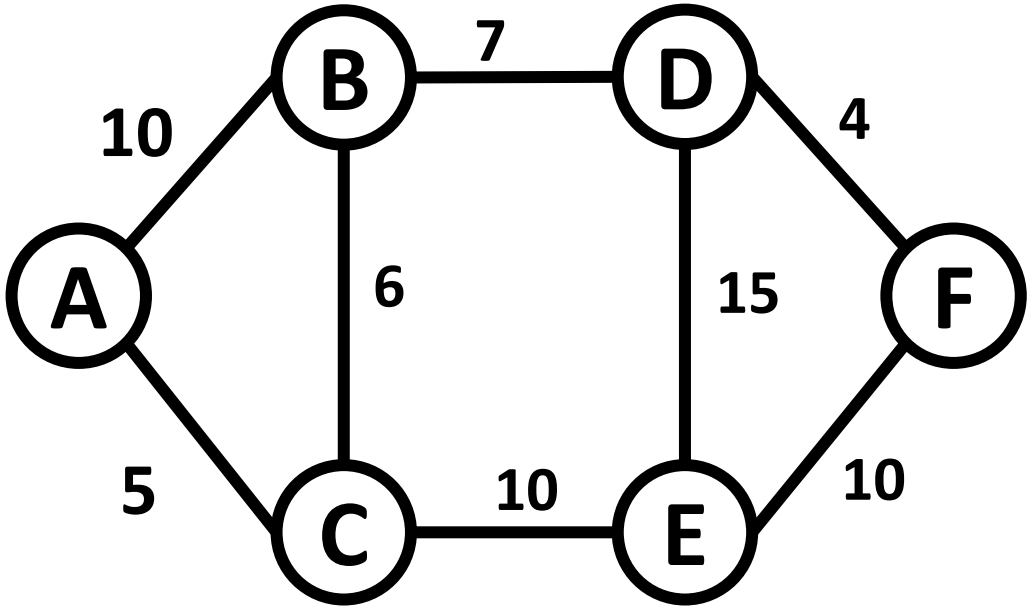


To B, Cost 10

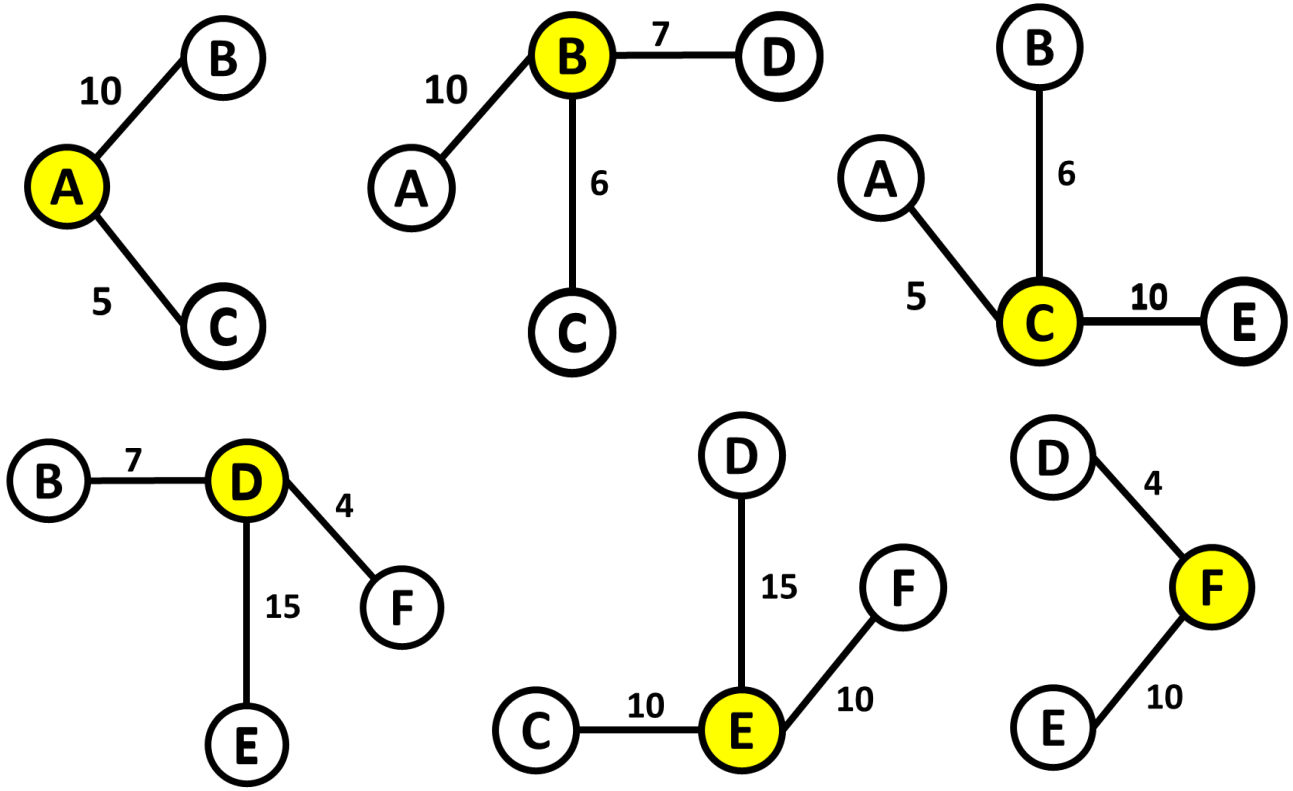
To C, Cost 5

Link State of Node A

Obtain the Topology: Link State



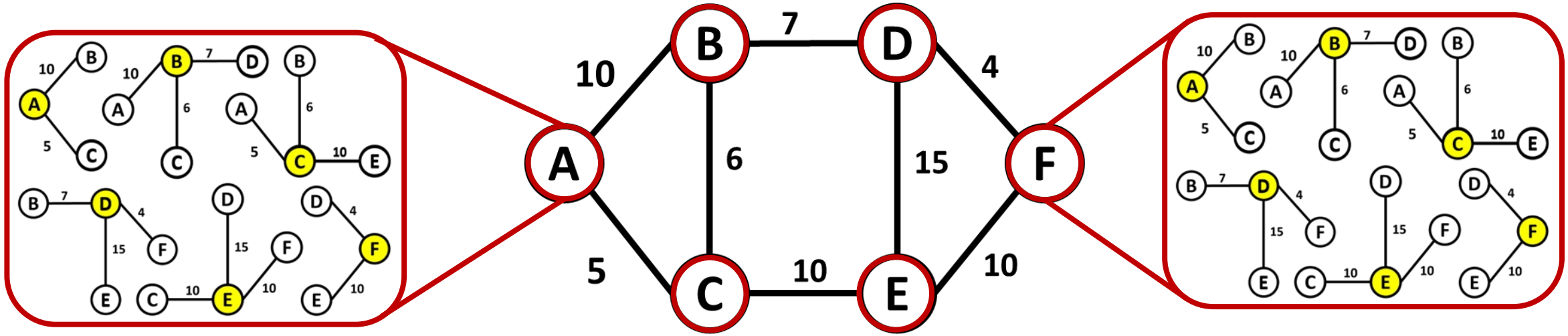
Topology of the whole network



Link State of All Nodes

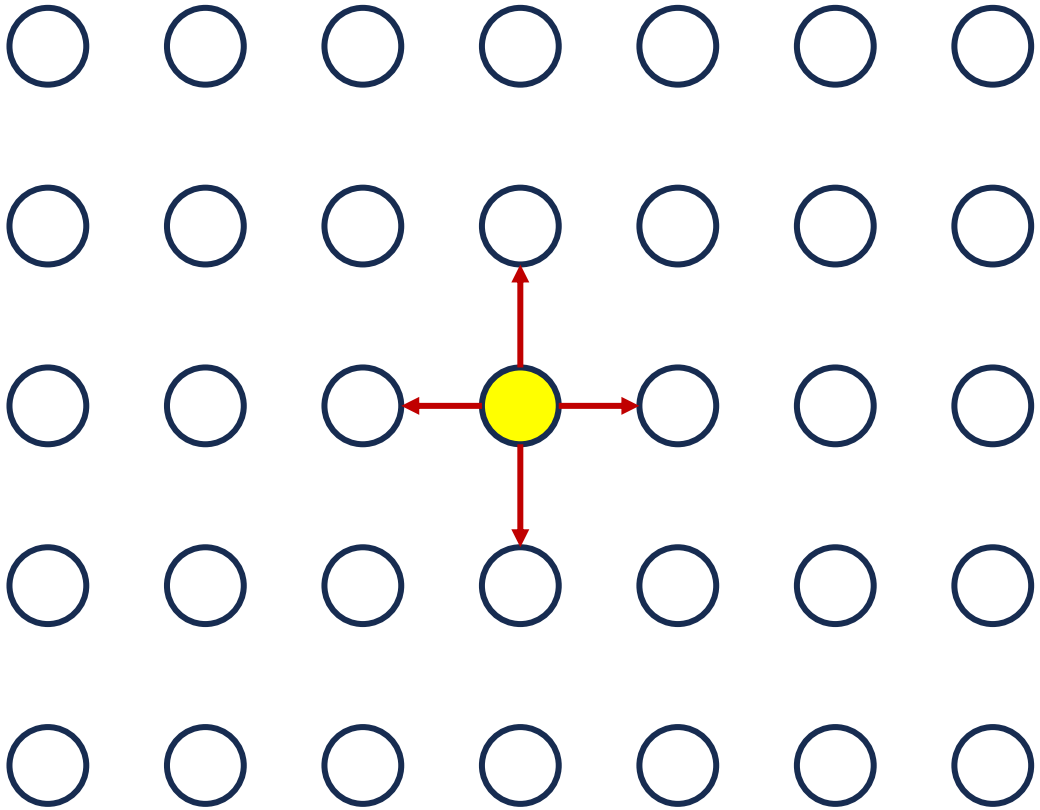
Combine the link states of all nodes, we can reconstruct the topology of the whole network!

Obtain the Topology: Link State



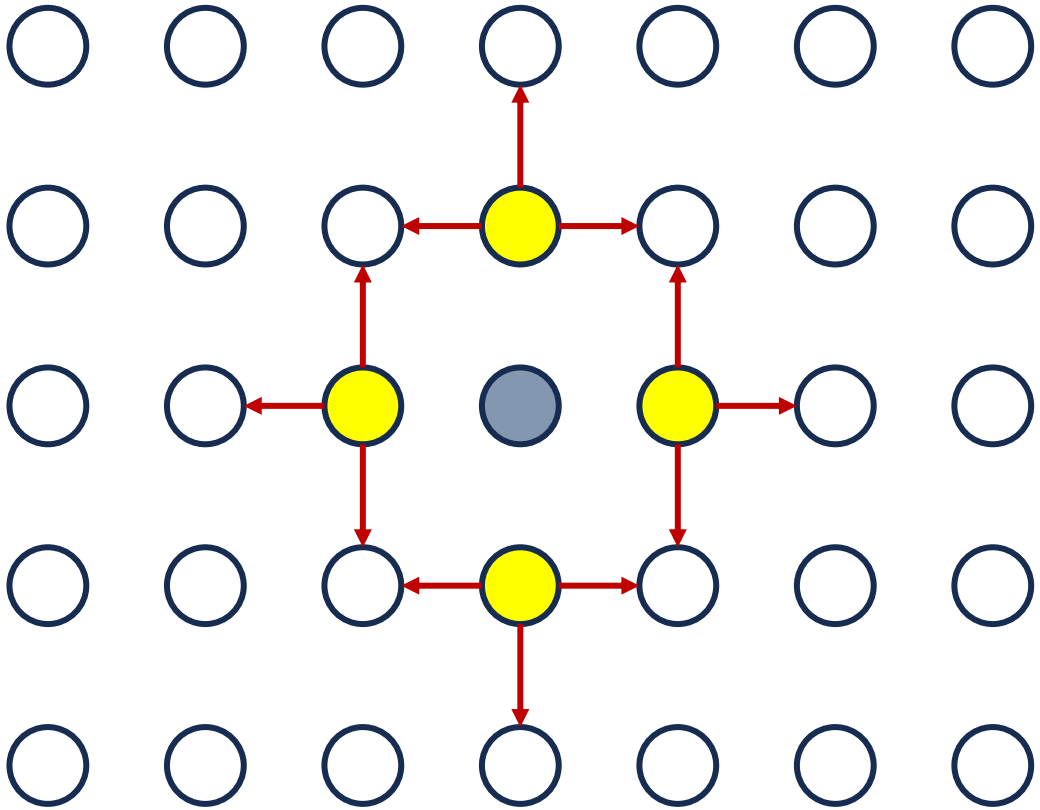
Each node needs to collect the link states of all other nodes before it can reconstruct the topology of the network

Obtain the Topology: Link State Flooding



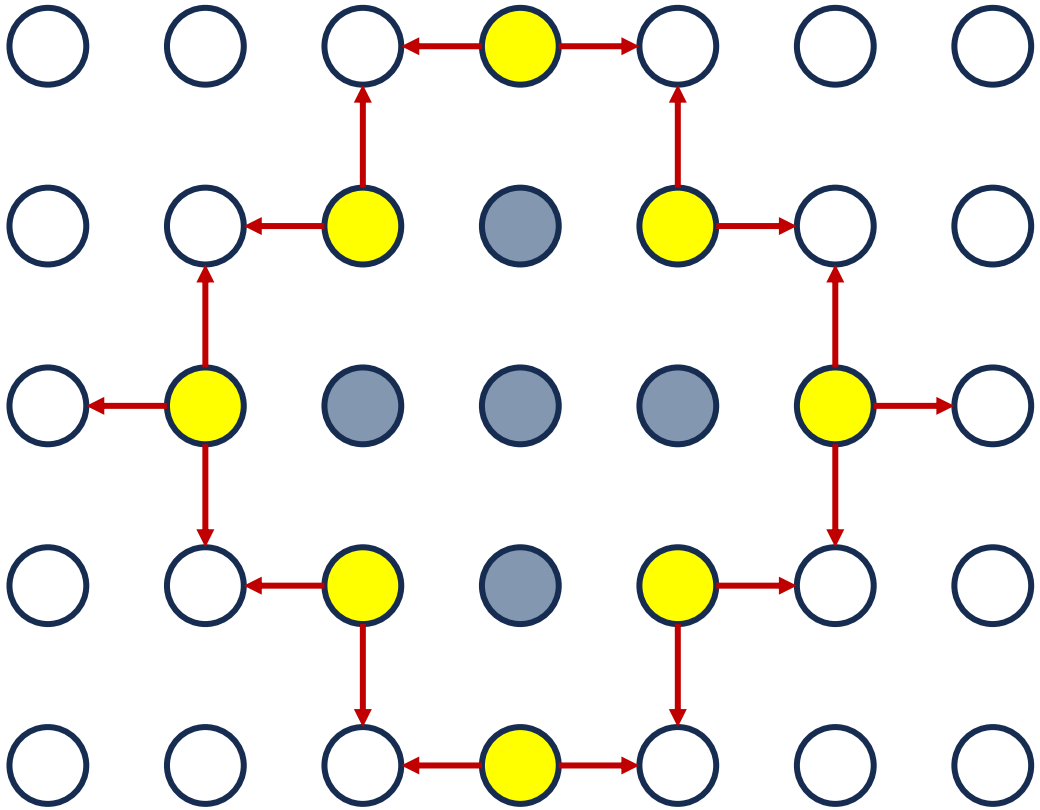
Example: one node floods one message to the whole network

Obtain the Topology: Link State Flooding



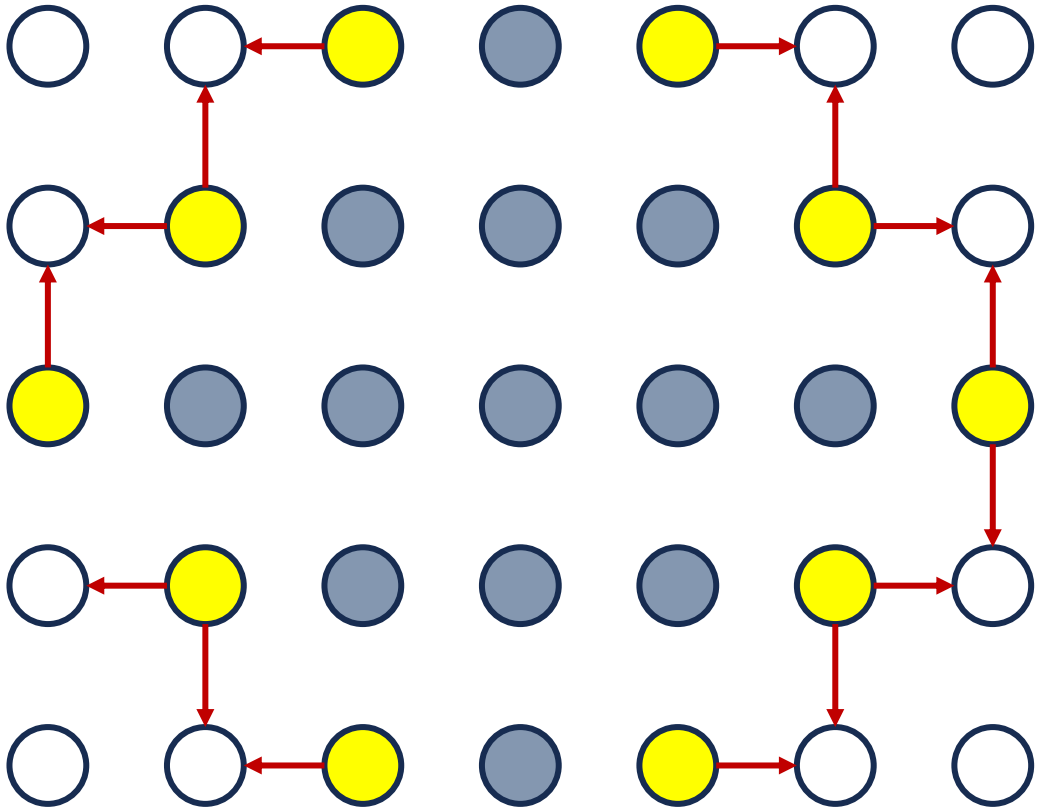
Example: one node floods one message to the whole network

Obtain the Topology: Link State Flooding



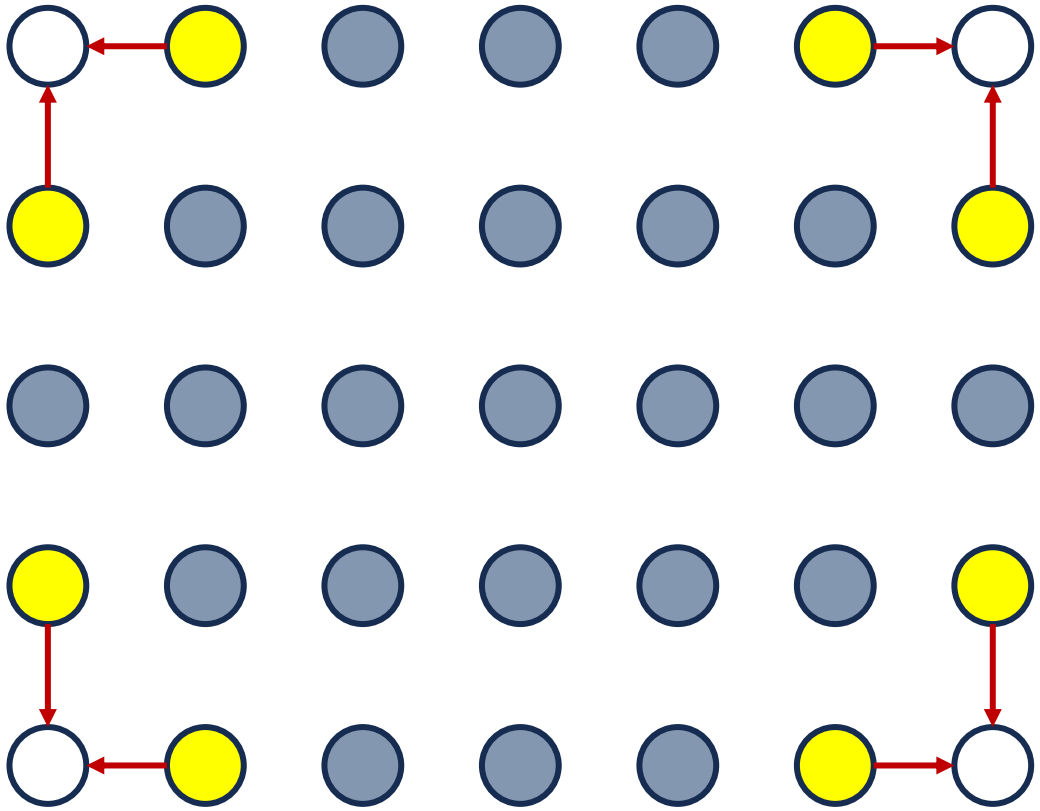
Example: one node floods one message to the whole network

Obtain the Topology: Link State Flooding



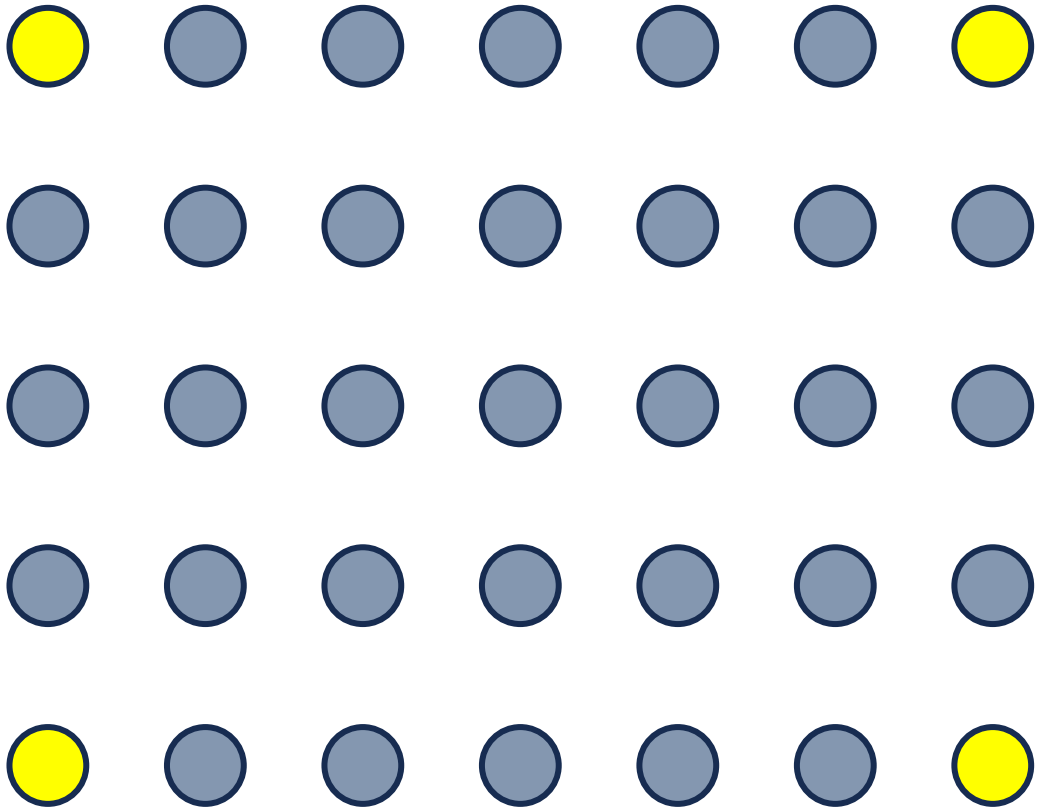
Example: one node floods one message to the whole network

Obtain the Topology: Link State Flooding



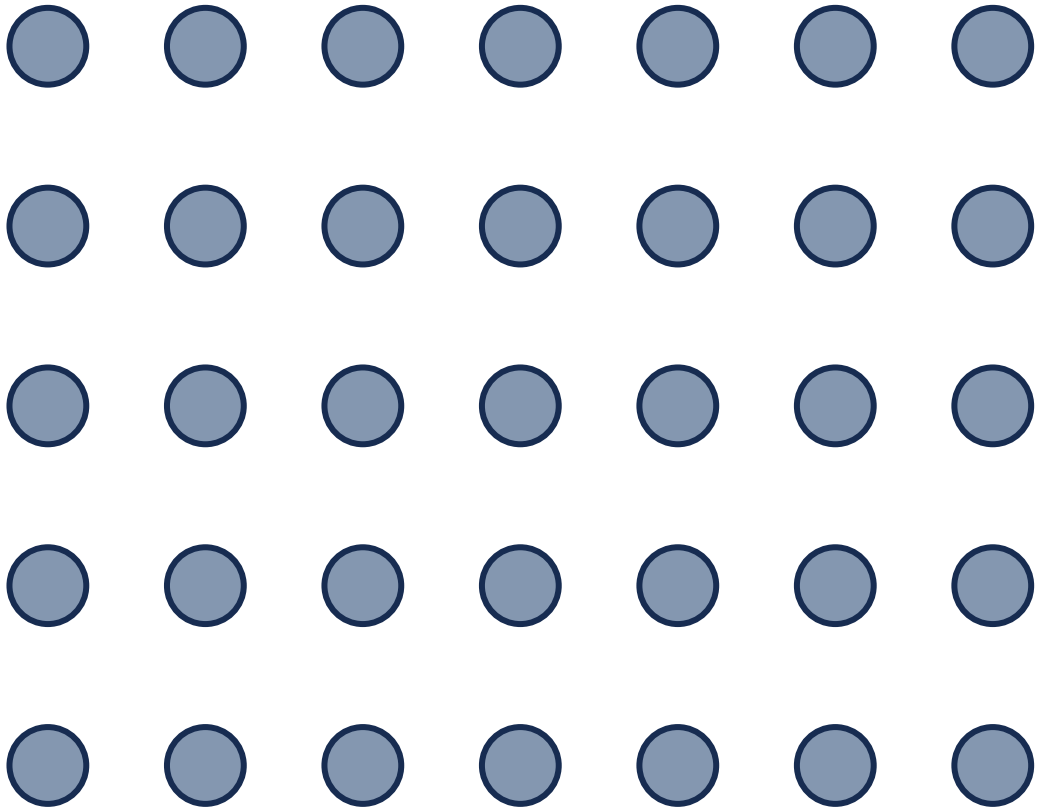
Example: one node floods one message to the whole network

Obtain the Topology: Link State Flooding



Example: one node floods one message to the whole network

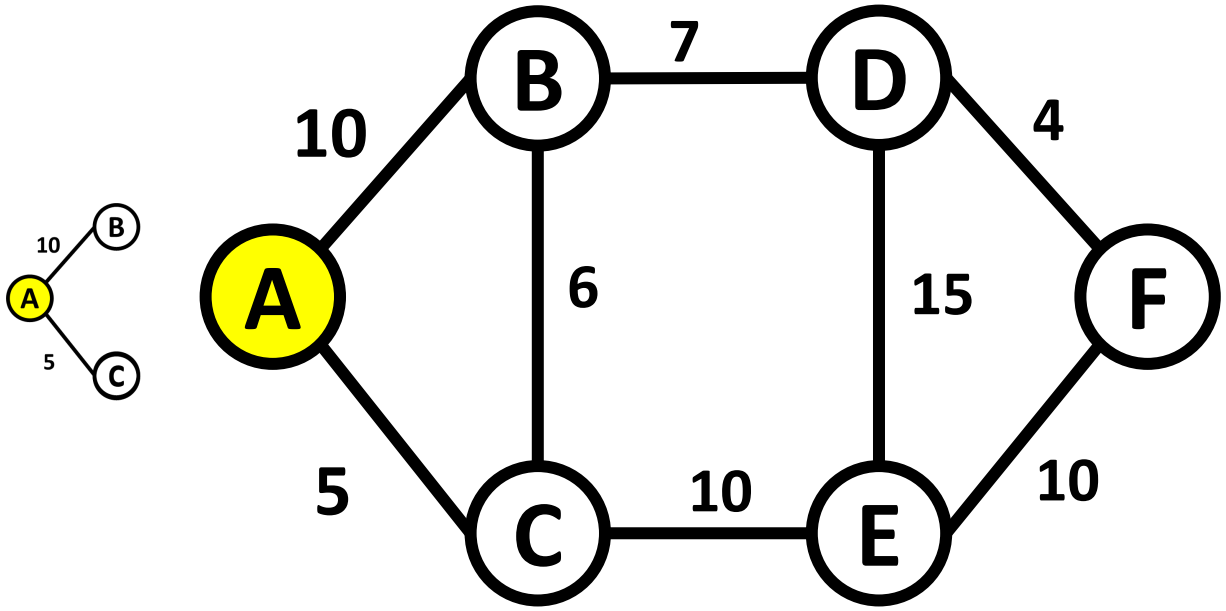
Obtain the Topology: Link State Flooding



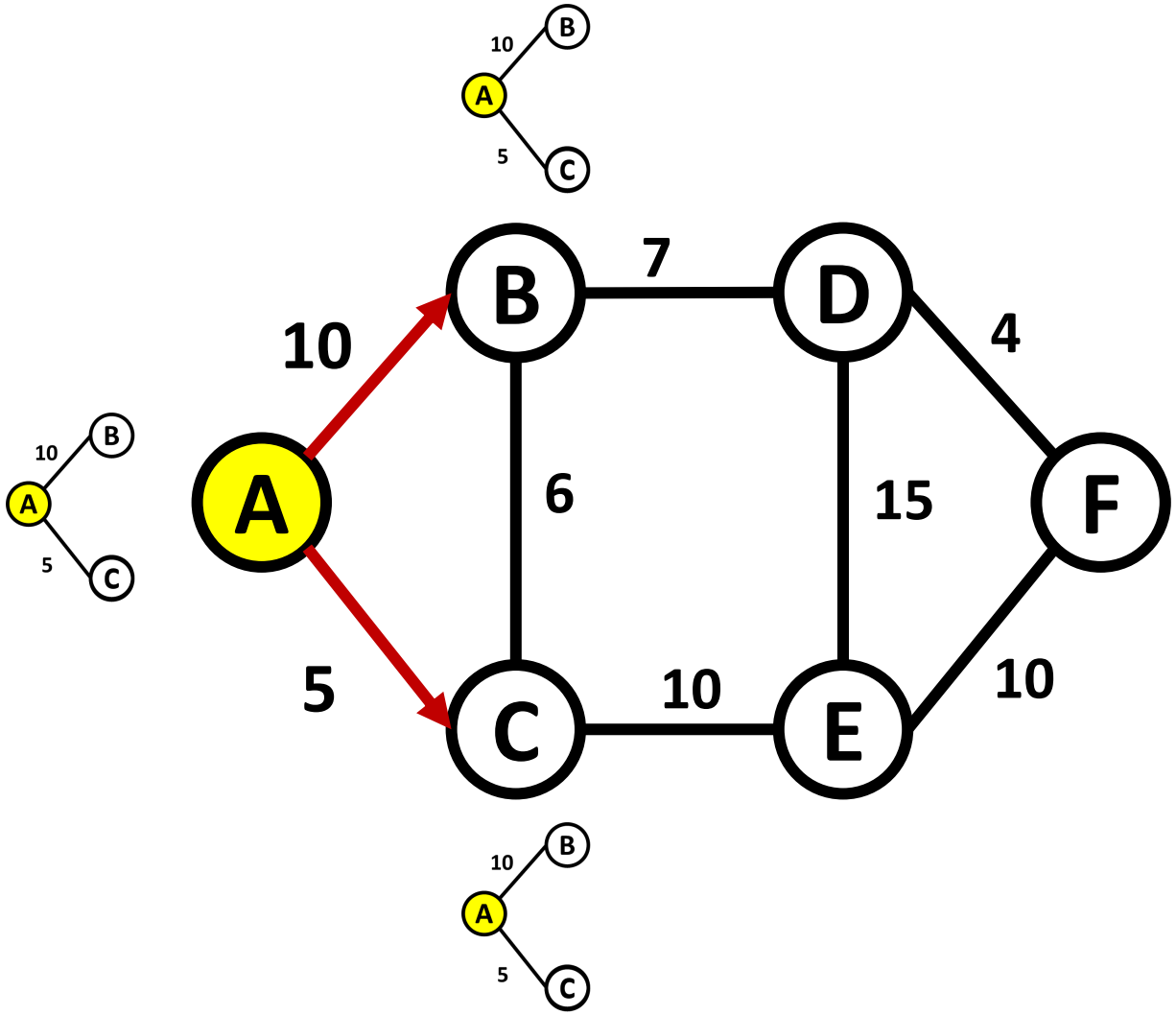
Example: one node floods one message to the whole network

Flooding finished!

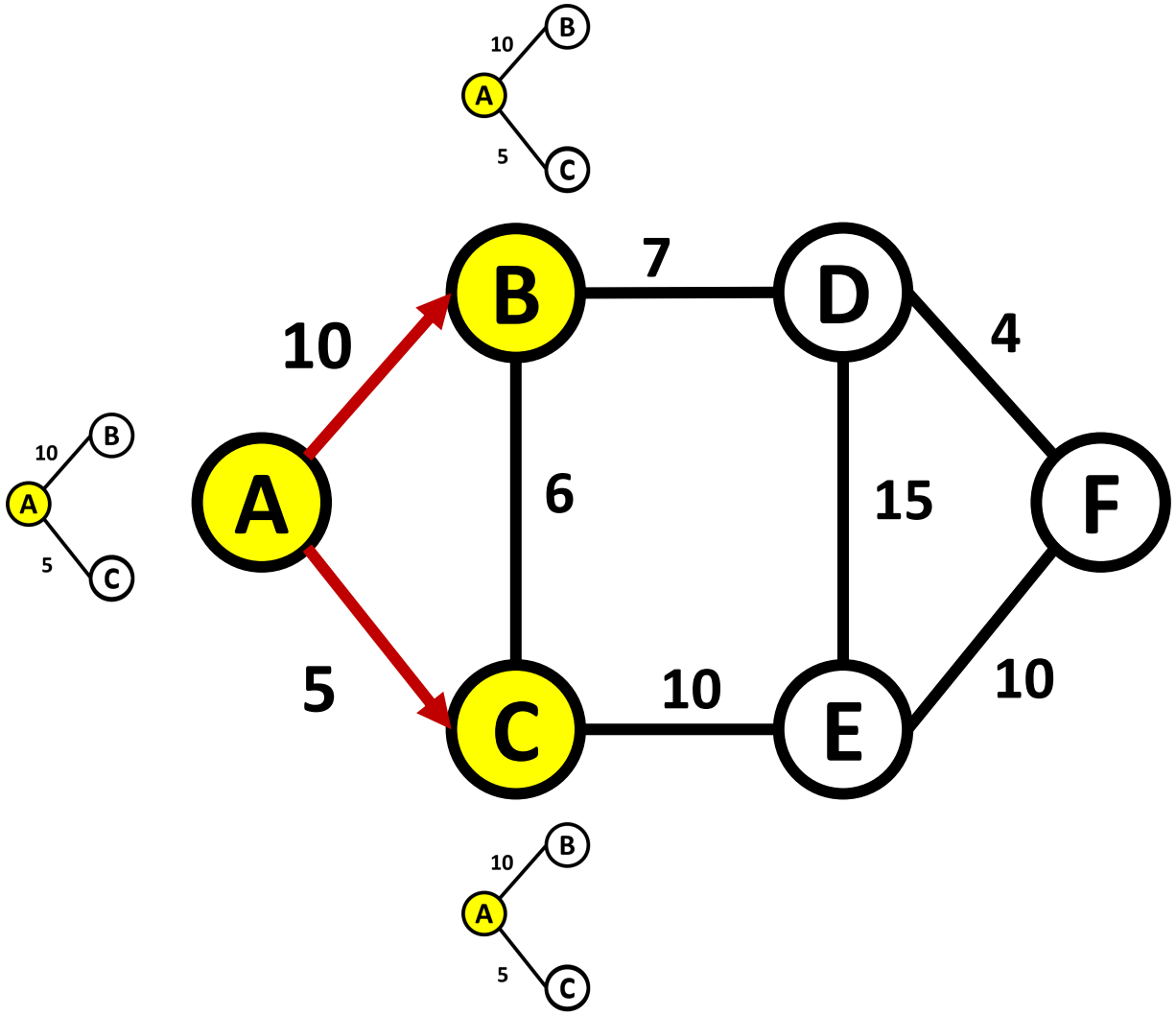
Obtain the Topology: Link State Flooding



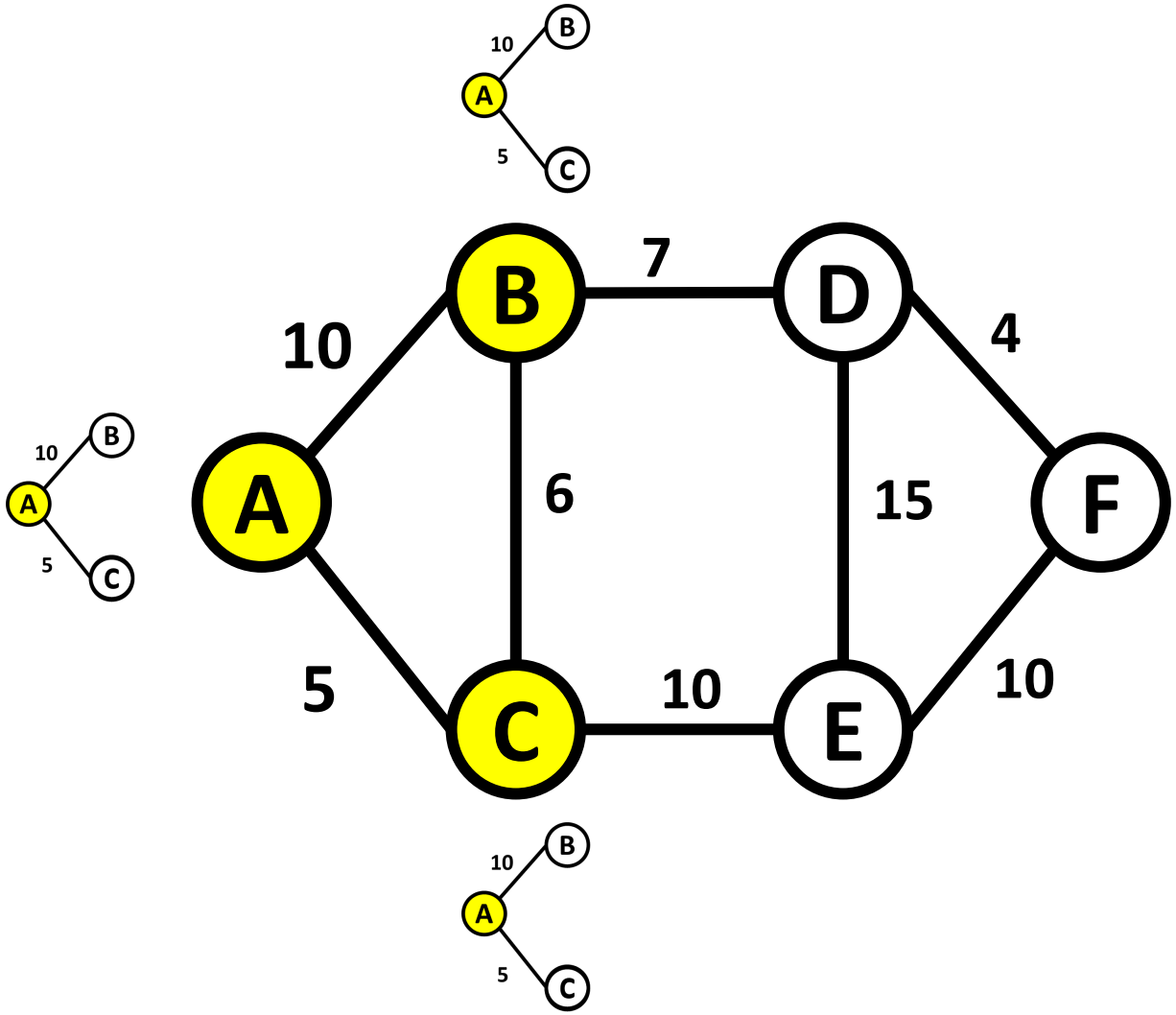
Obtain the Topology: Link State Flooding



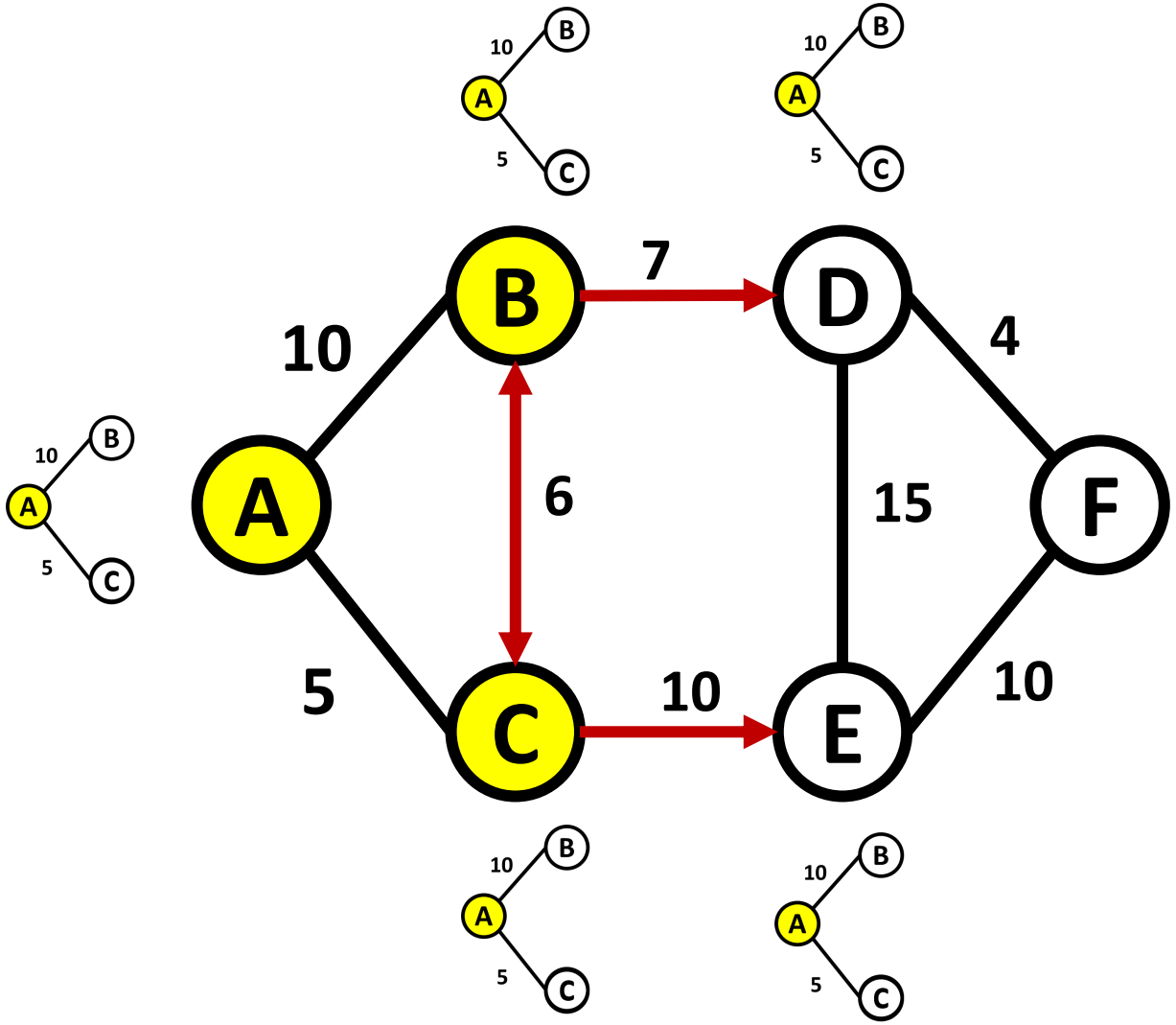
Obtain the Topology: Link State Flooding



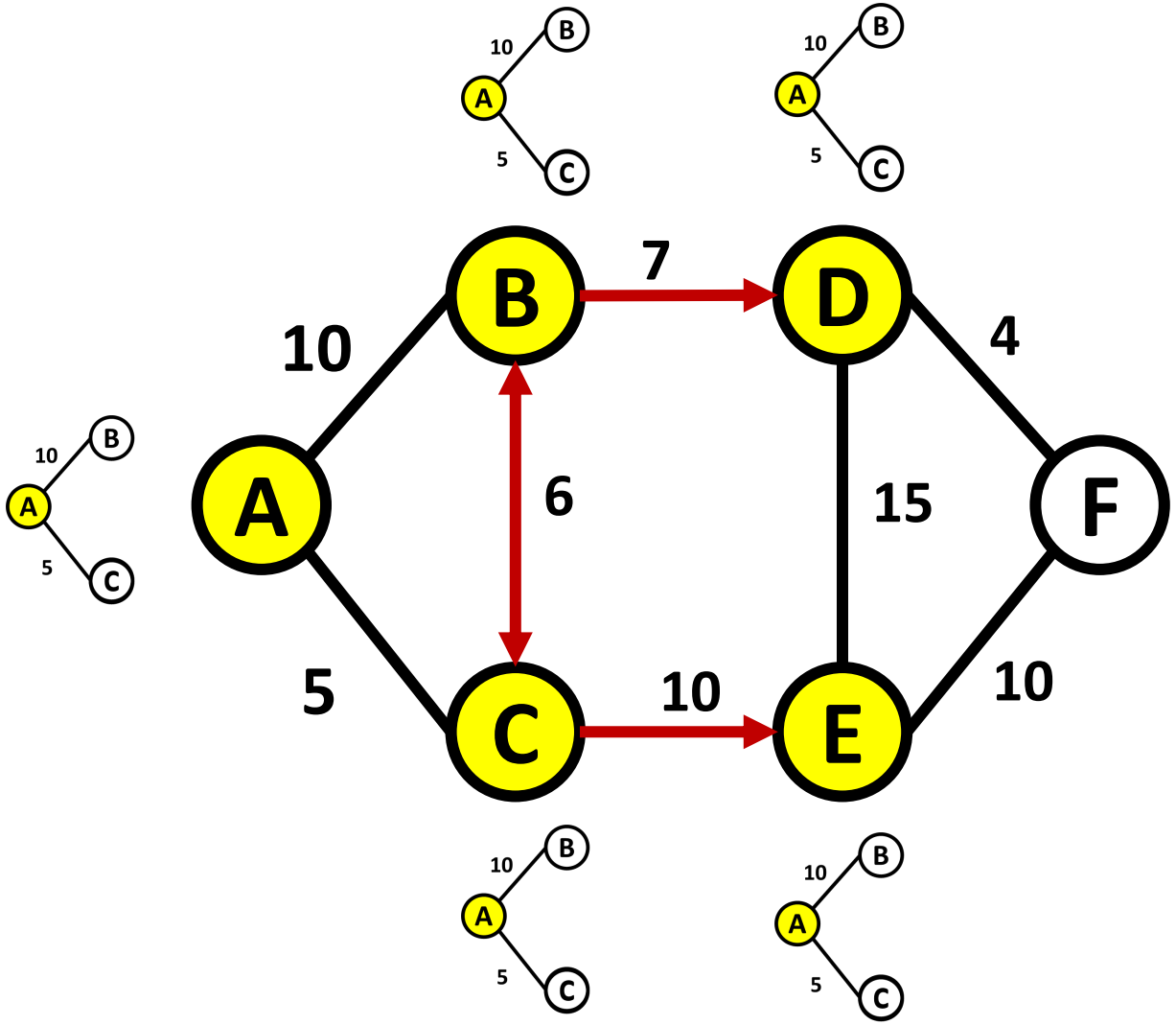
Obtain the Topology: Link State Flooding



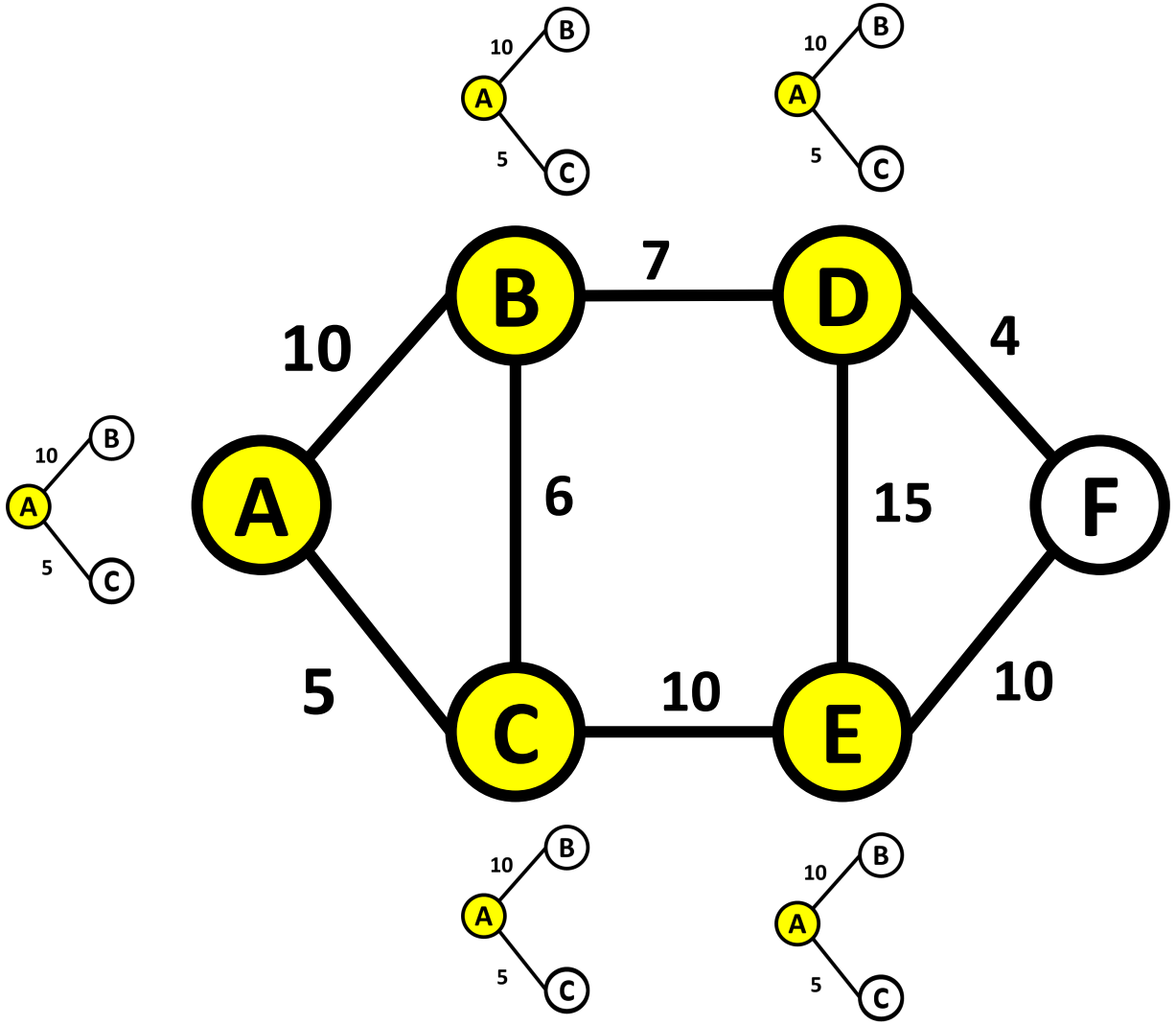
Obtain the Topology: Link State Flooding



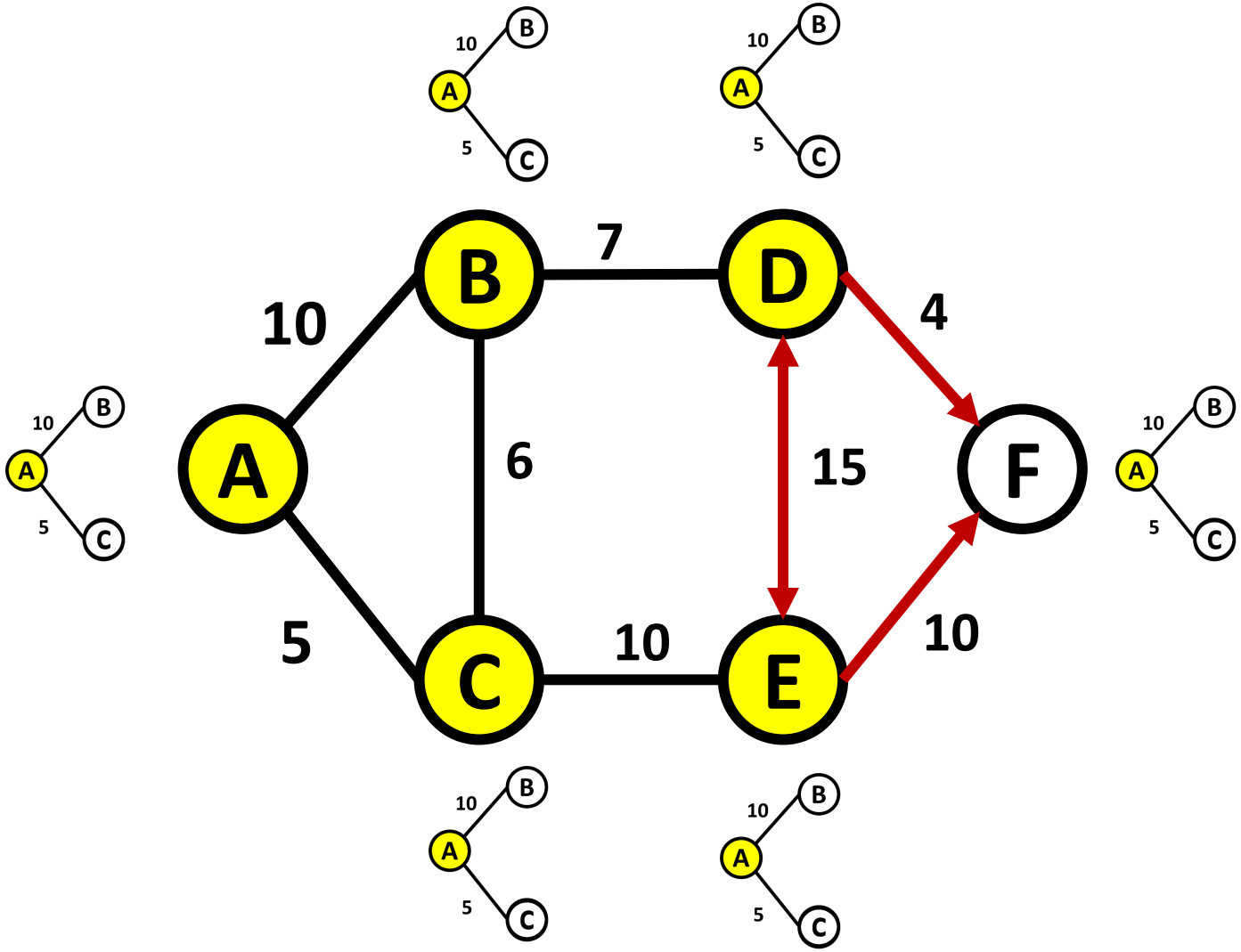
Obtain the Topology: Link State Flooding



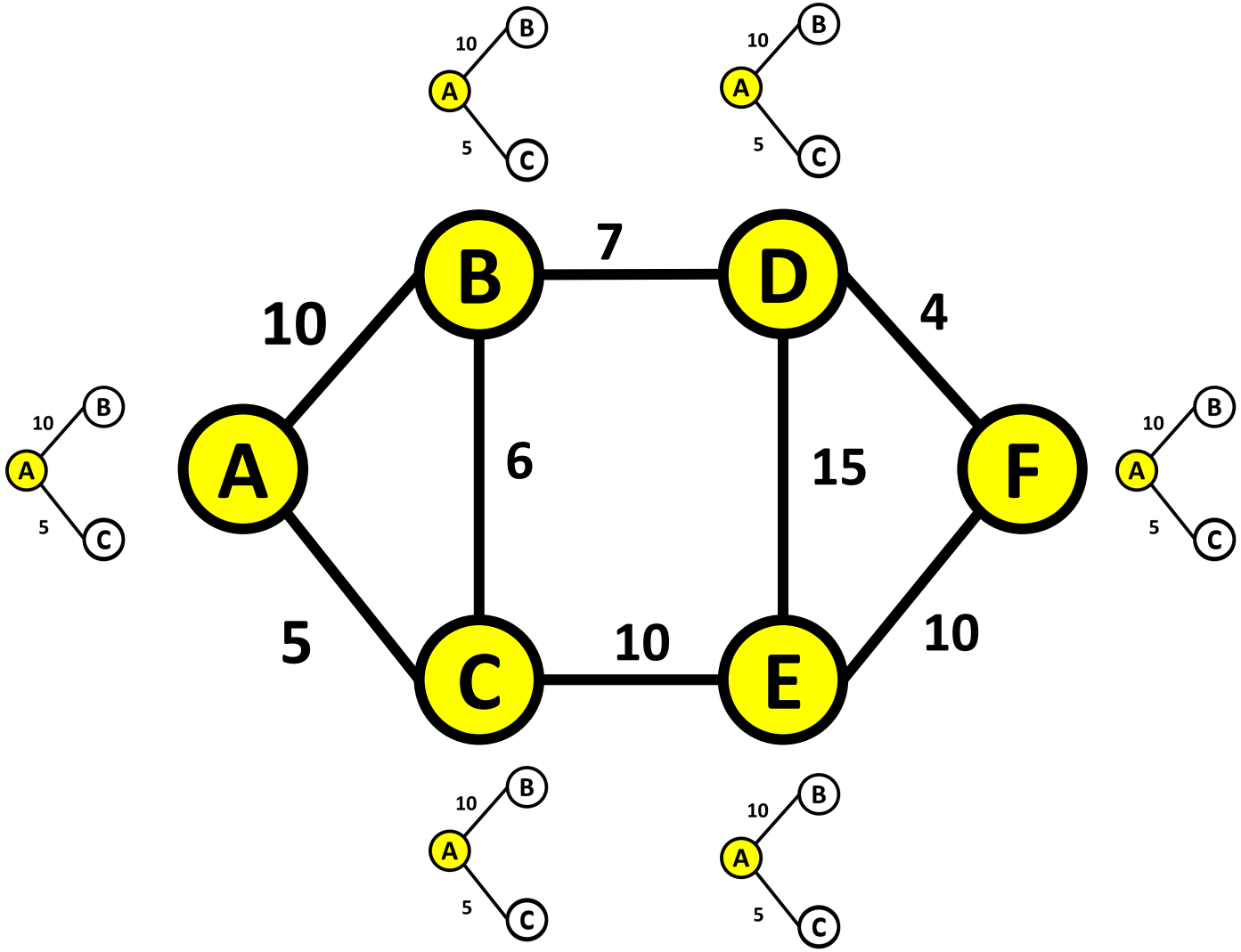
Obtain the Topology: Link State Flooding



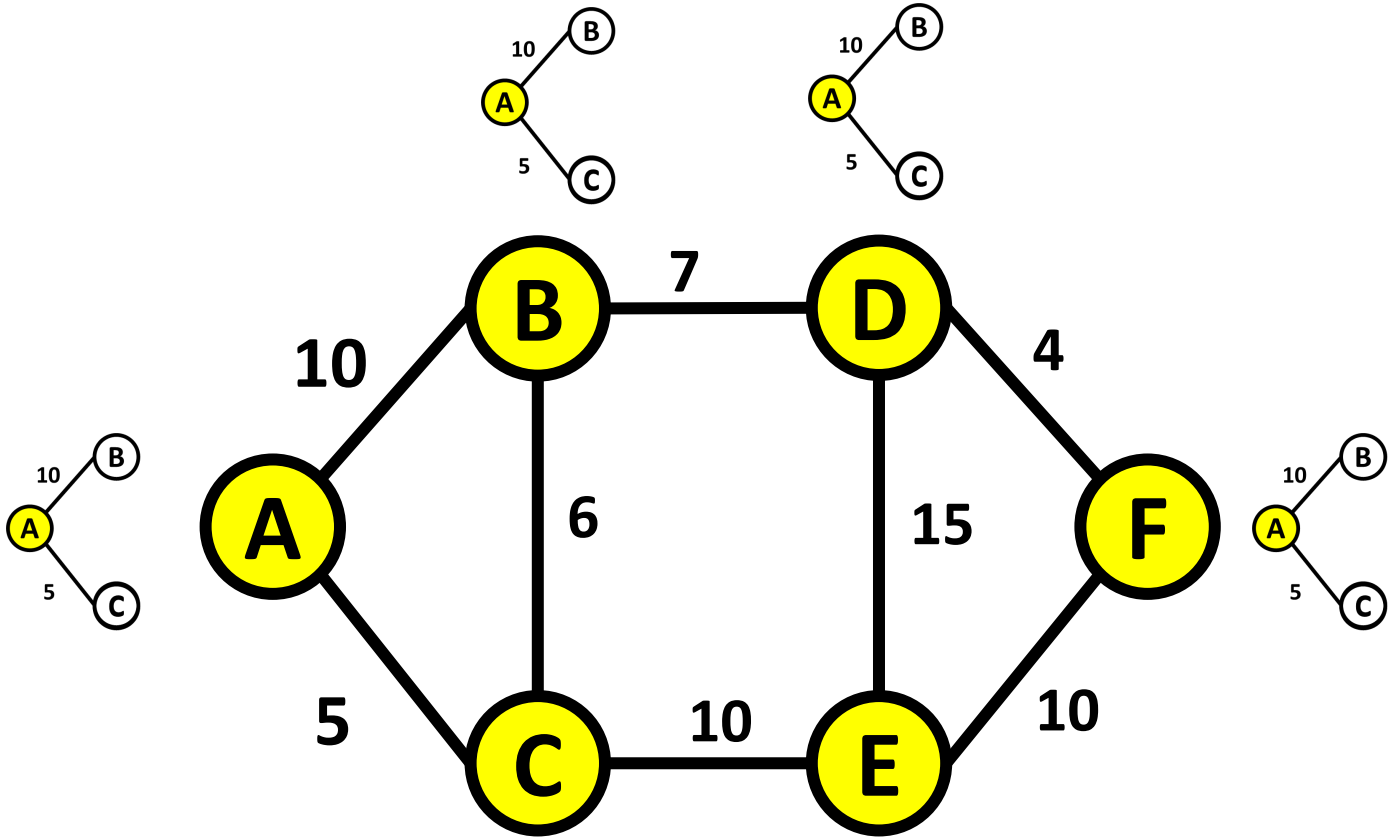
Obtain the Topology: Link State Flooding



Obtain the Topology: Link State Flooding



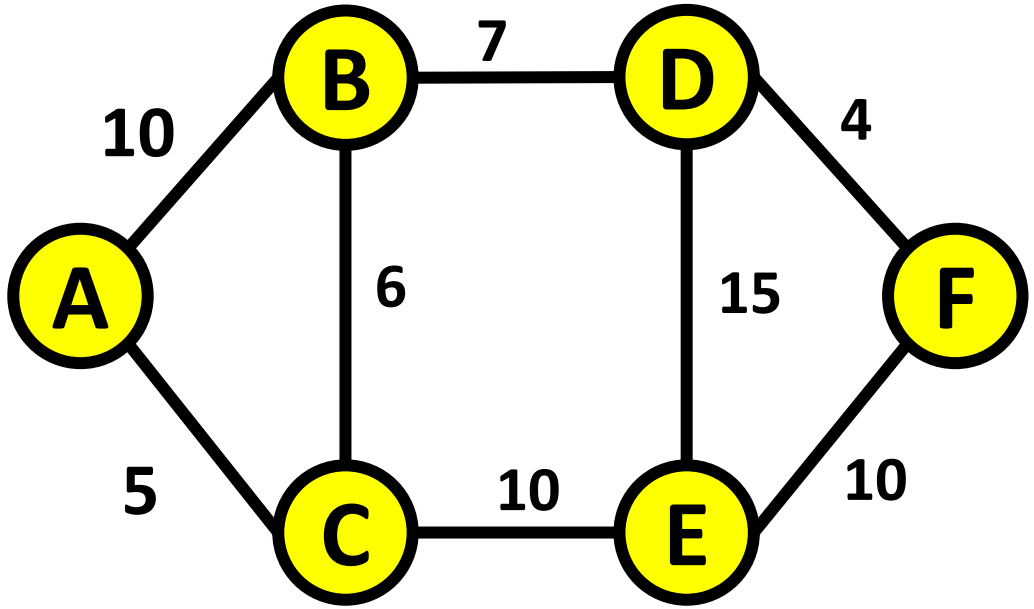
Obtain the Topology: Link State Flooding



Ideally, every node floods correctly and gets the same topology

Link State Flooding in real world

Link-state routing is elegant in theory, but real networks introduce loss, delay, duplicates, and topology changes.



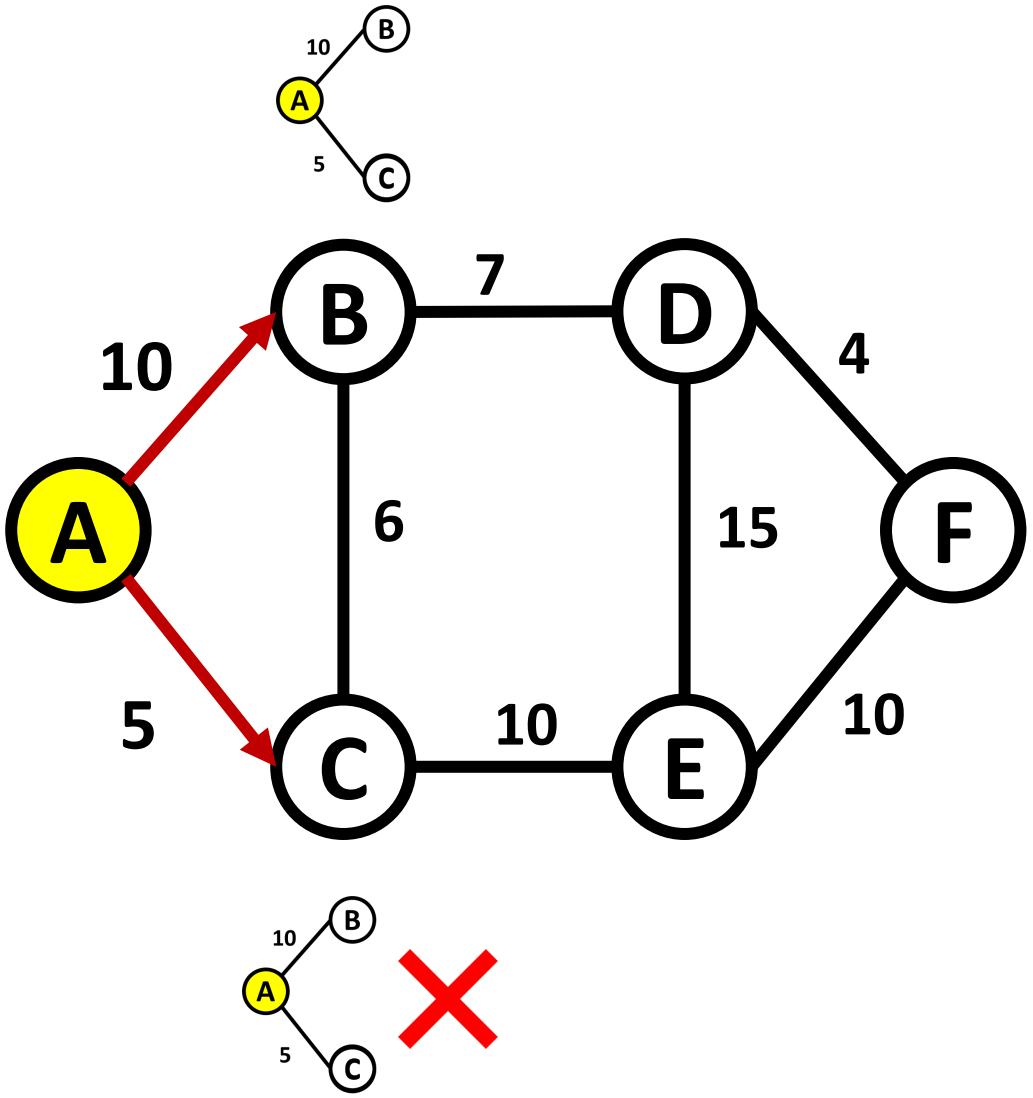
Ideally

- Every flooded packet is delivered
- No corruption
- No delay-related confusion
- No concurrent topology changes

In practice

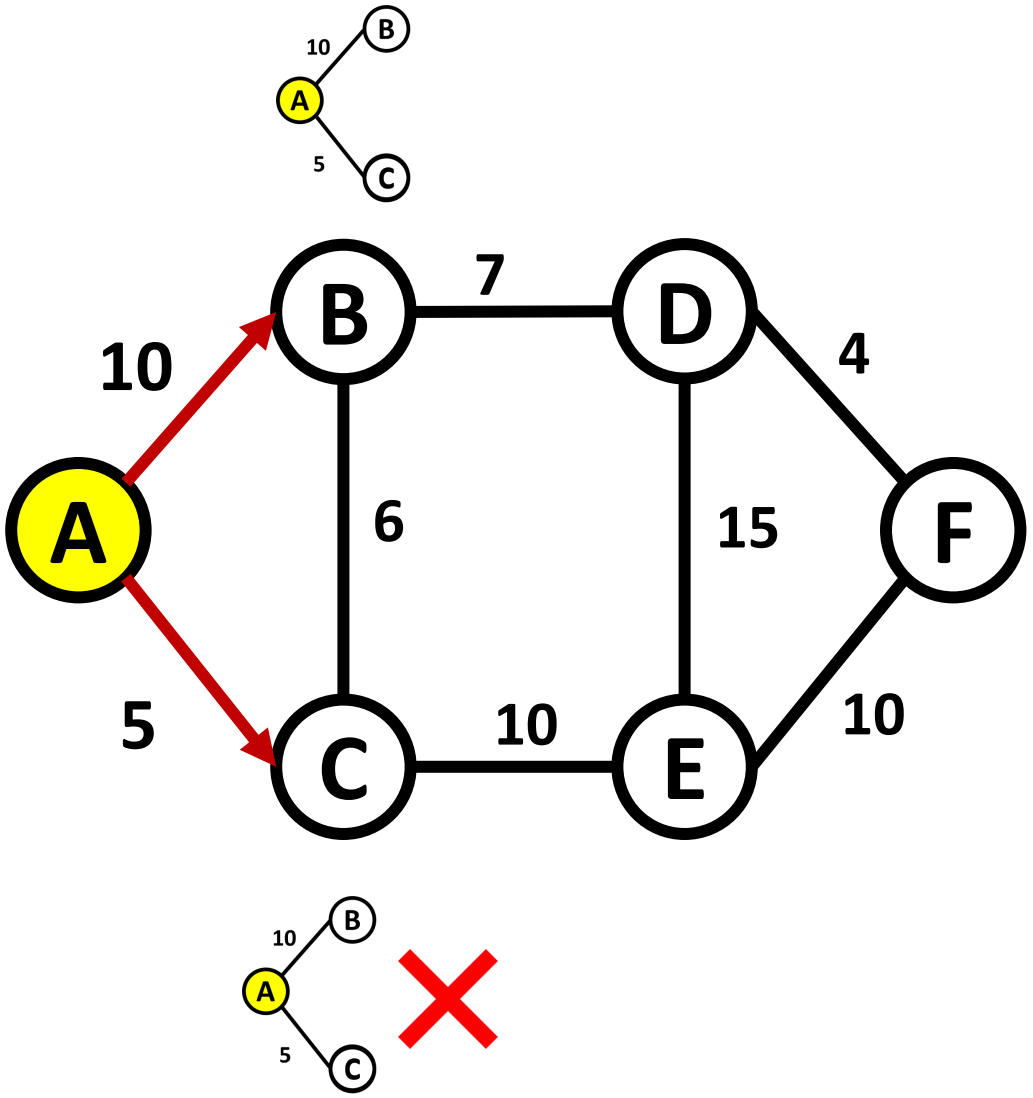
- Packets may be lost
- Updates may arrive late
- Links may fail while updates are still propagating

Link State Flooding: Packet Loss



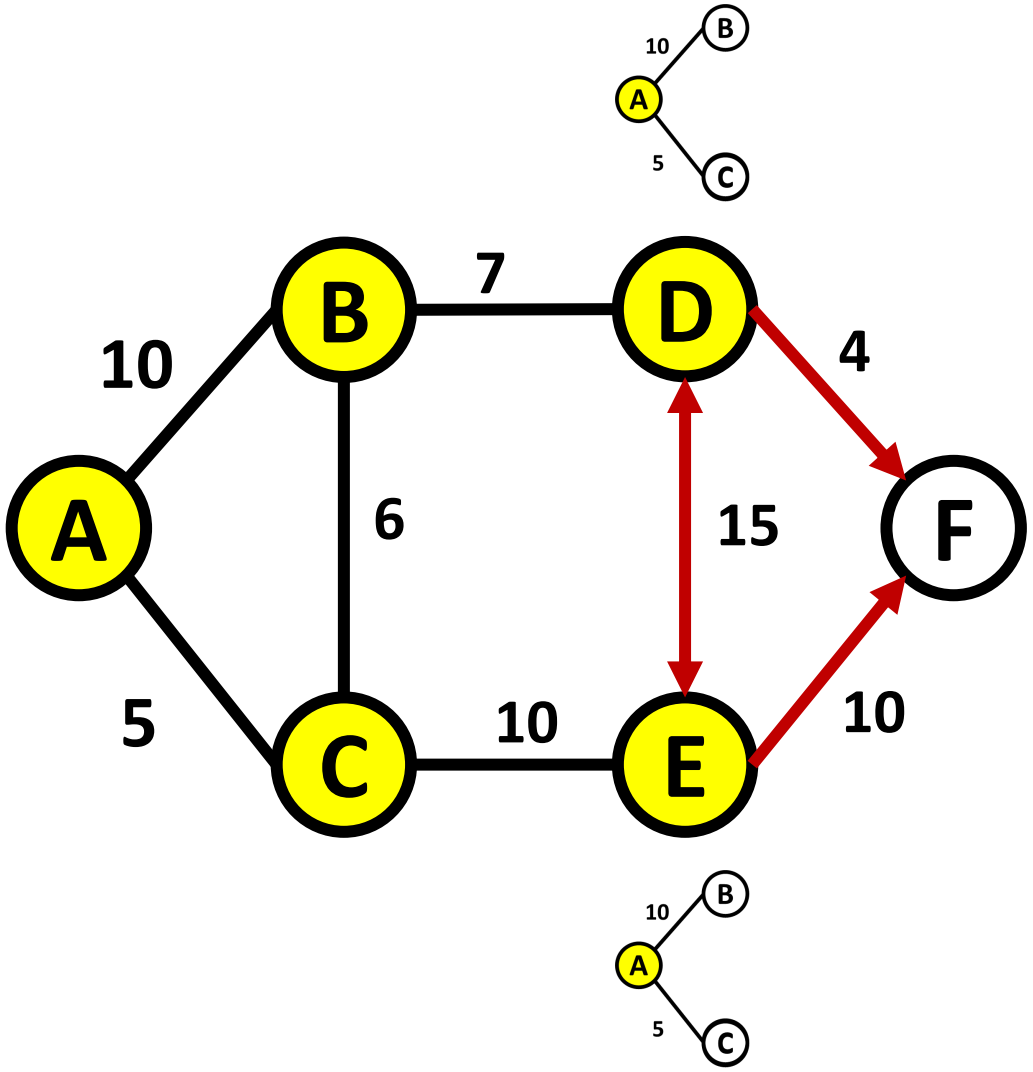
- If one router misses an update:
  - Some routers install the new topology
  - Others keep the old topology
  - Topology become inconsistent
- Possible effects:
  - Inconsistent routes
  - Temporary loops
  - Blackholes
  - Dropped packets

Link State Flooding: Packet Loss



- Flooding cannot be purely "send once and hope for the best."
- The protocol needs a way to ensure neighbors eventually receive the update.
- Reliable flooding:
  - explicit acknowledgment
  - request/retransmission mechanism

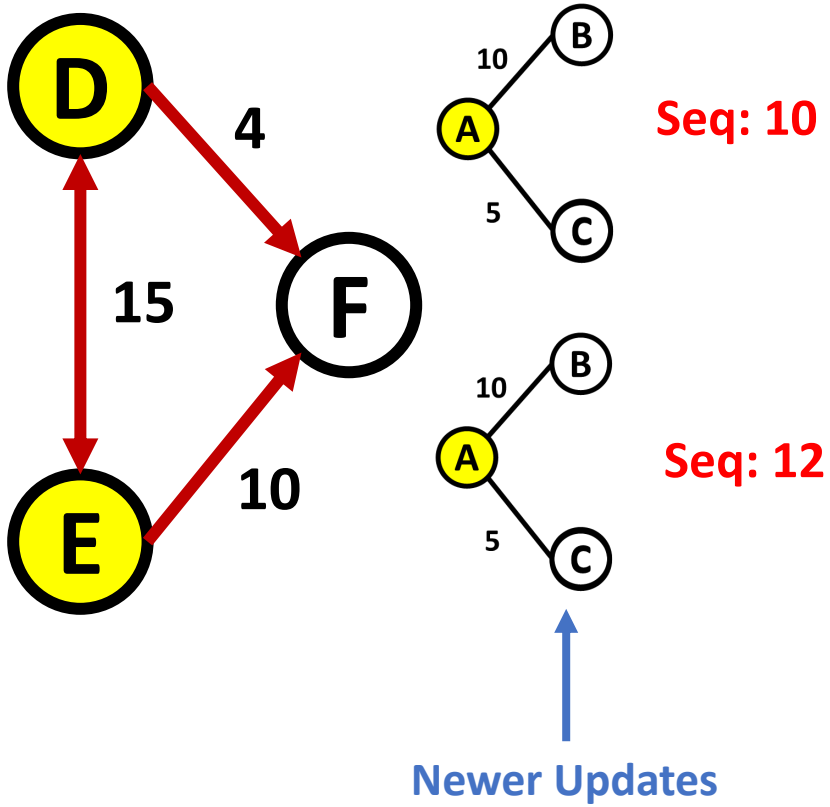
Link State Flooding: Duplicate Packets



Flooding Naturally Creates Duplicates

- Even if routers do not send an update back on the incoming link, the same update can still arrive along multiple paths.
- the protocol must answer:
  - Is this advertisement new?
  - Is it just a duplicate?

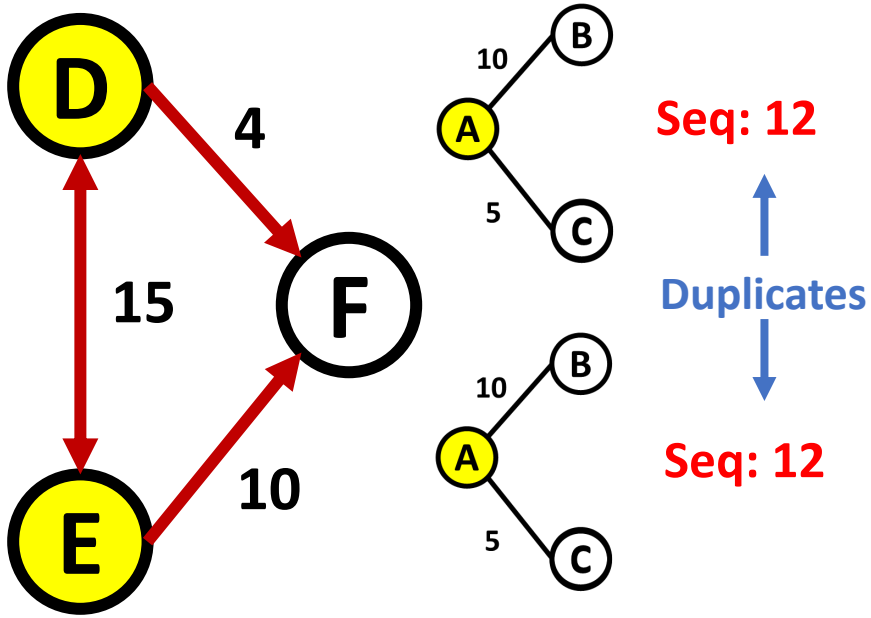
Link State Flooding: Duplicate Packets



Solution: Sequence Numbers

- Each new advertisement from the same router gets a larger sequence number.
- For a given originator:
  - Larger sequence number = newer information

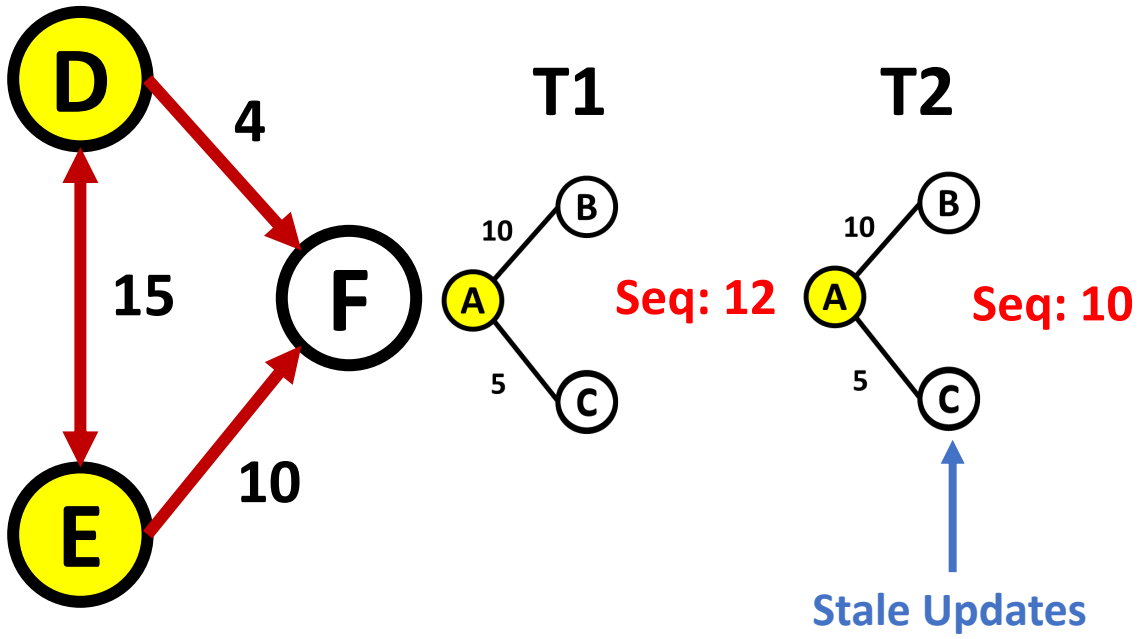
Link State Flooding: Duplicate Packets



Solution: Sequence Numbers

- Each new advertisement from the same router gets a larger sequence number.
- For a given originator:
  - Larger sequence number = newer information
  - equal sequence number = duplicate

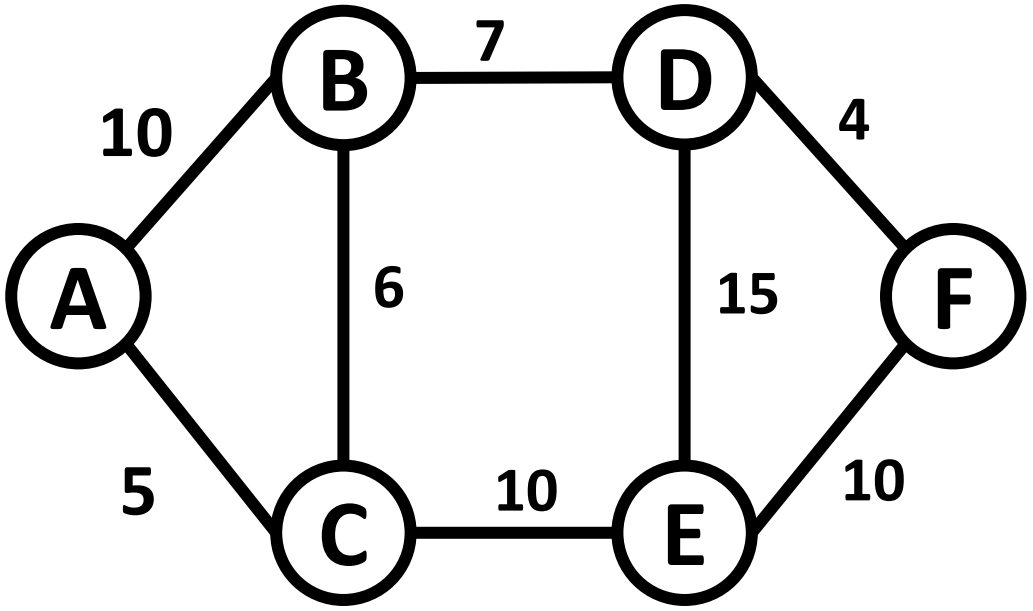
Link State Flooding: Duplicate Packets



Solution: Sequence Numbers

- Each new advertisement from the same router gets a larger sequence number.
- For a given originator:
  - Larger sequence number = newer information
  - equal sequence number = duplicate
  - smaller sequence number = stale information

Link State Flooding: Topology Change



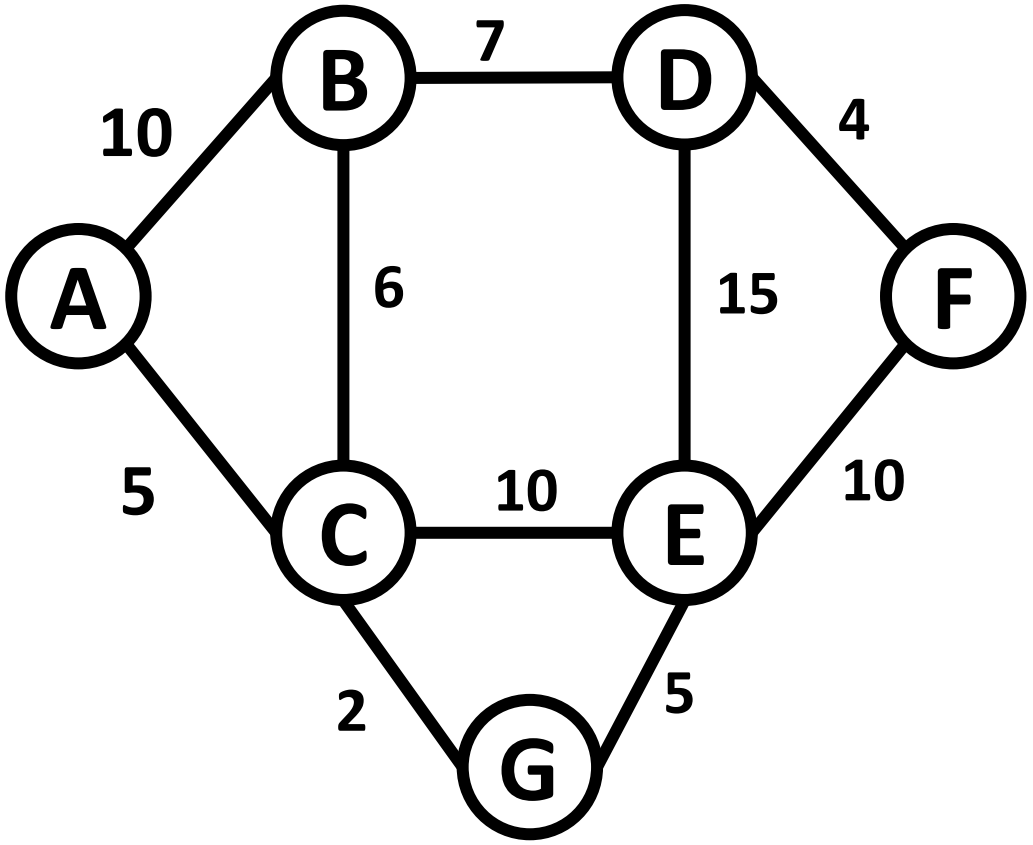
Link Change1: adding a new link

Link Change2: deleting a new link

Link Change3: cost of a link changes

- Unified rule for all link changes
  - Update local link-state record
  - Create a newer version
  - Flood the new advertisement
  - Neighbors install it
  - Routers recompute shortest paths

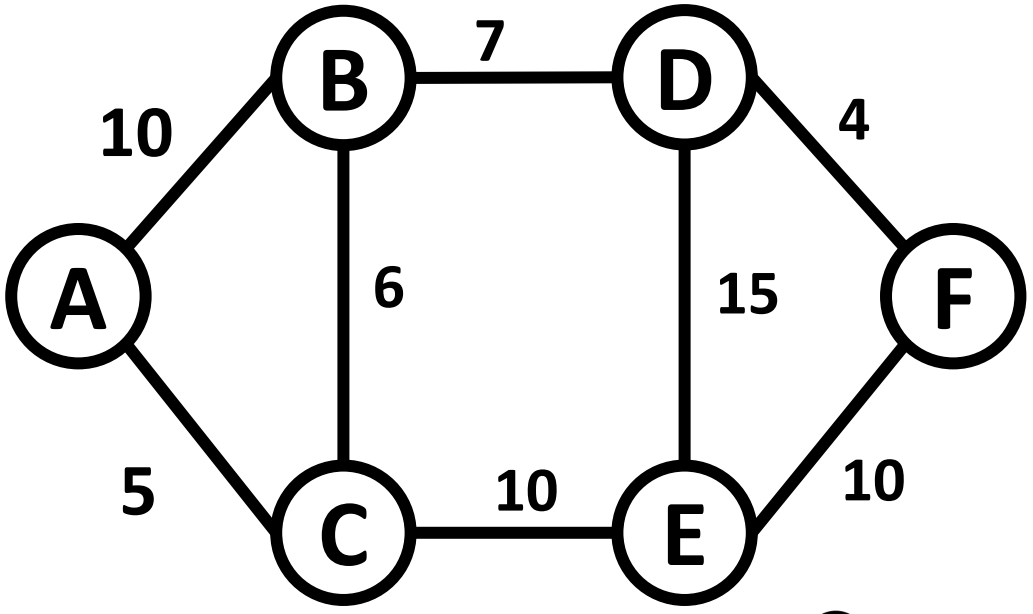
Link State Flooding: Topology Change

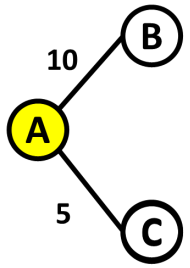


New Node Joins the Network

- It will detects its neighbors
- Its neighbors will also detect the node
- Messages about the new node will be flooded into the network

Link State Flooding: Topology Change





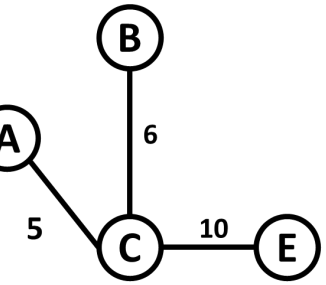
Seq: 12

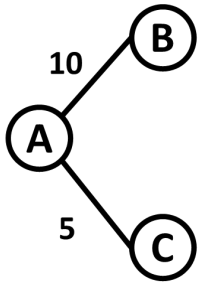
Age: 1200 s

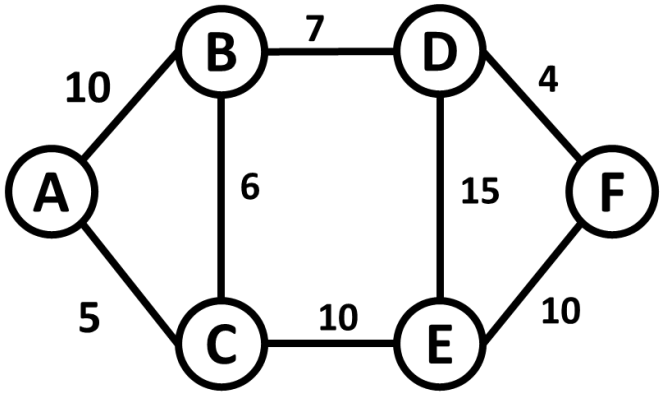
A router may fail, shut down, or become isolated

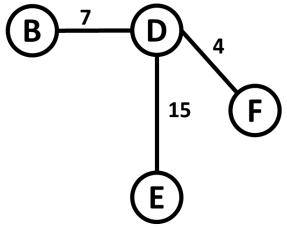
- Its neighbors will detect the node's disappear and flood that into the network
- What about the messages that flooded by the disappeared node?
- Each link state has an age field
  - The link stage becomes stale if the age exceeds the threshold

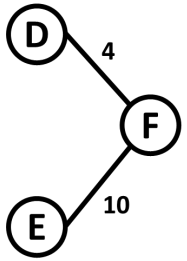
Obtain the Topology: Defining the Cost

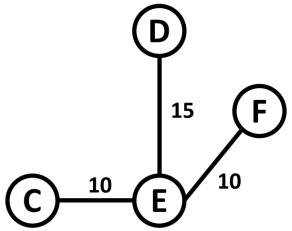


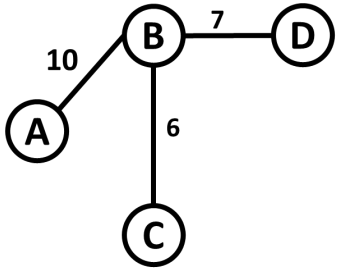


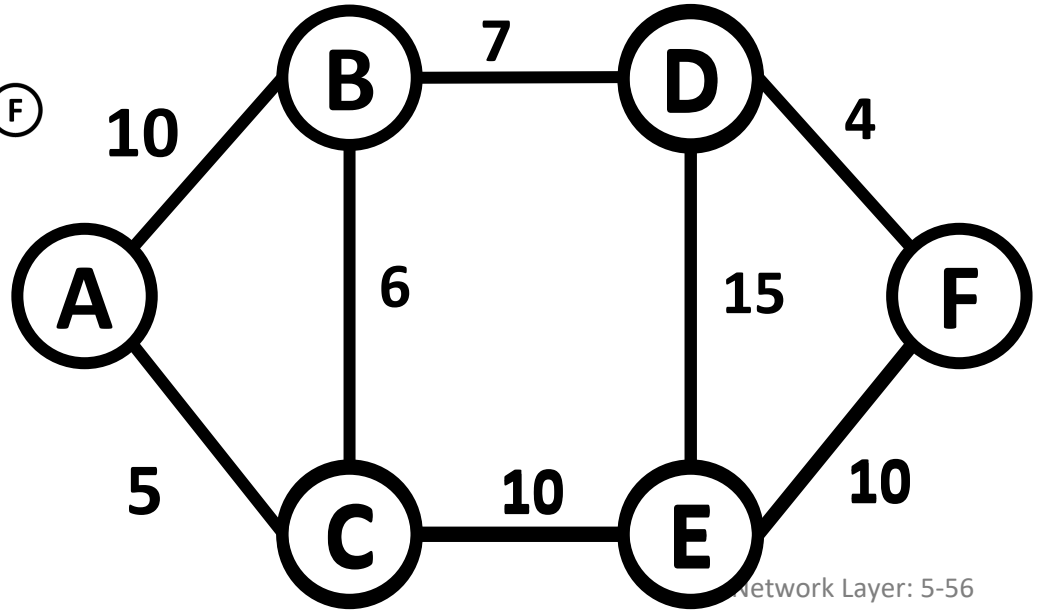




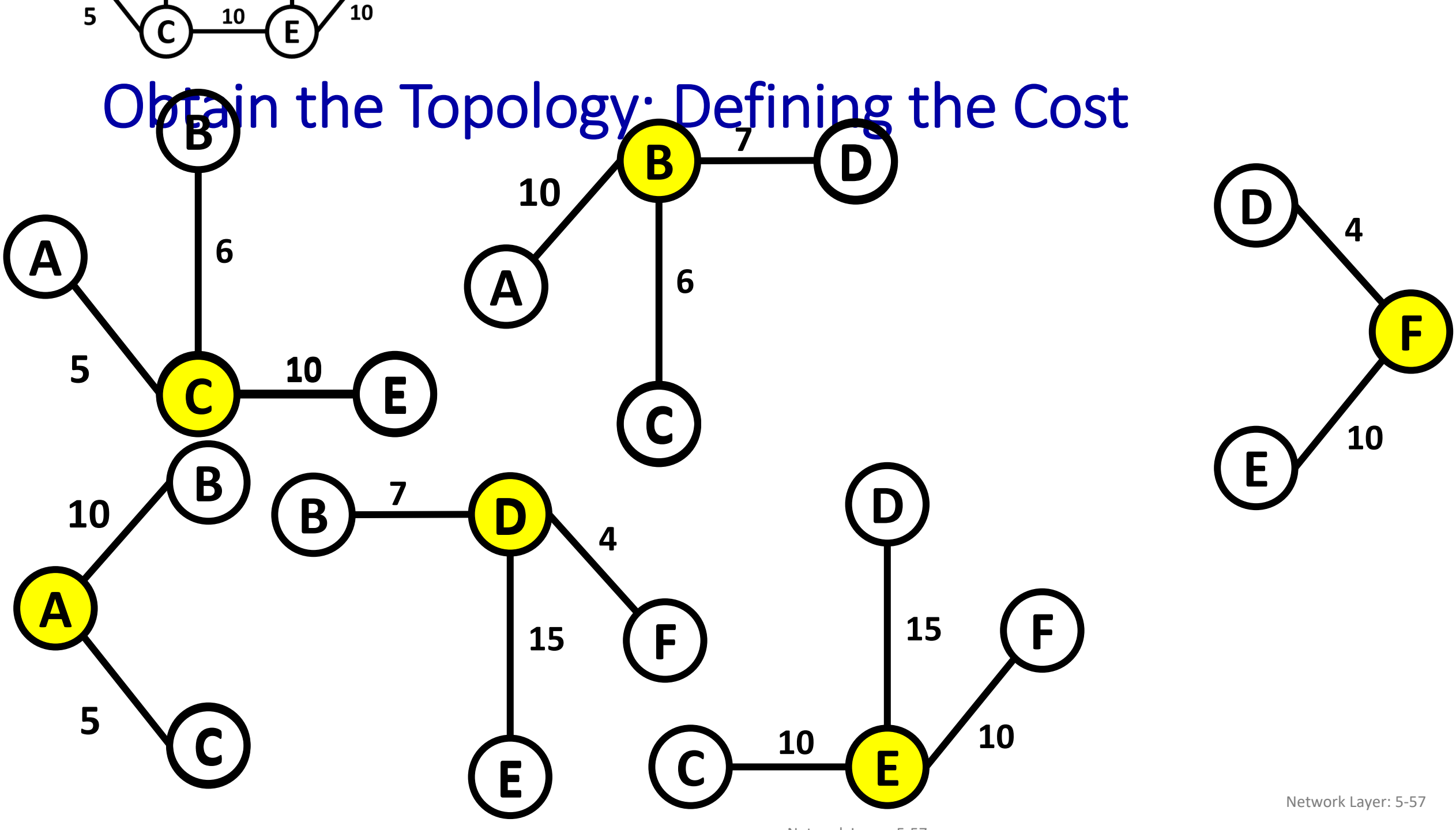








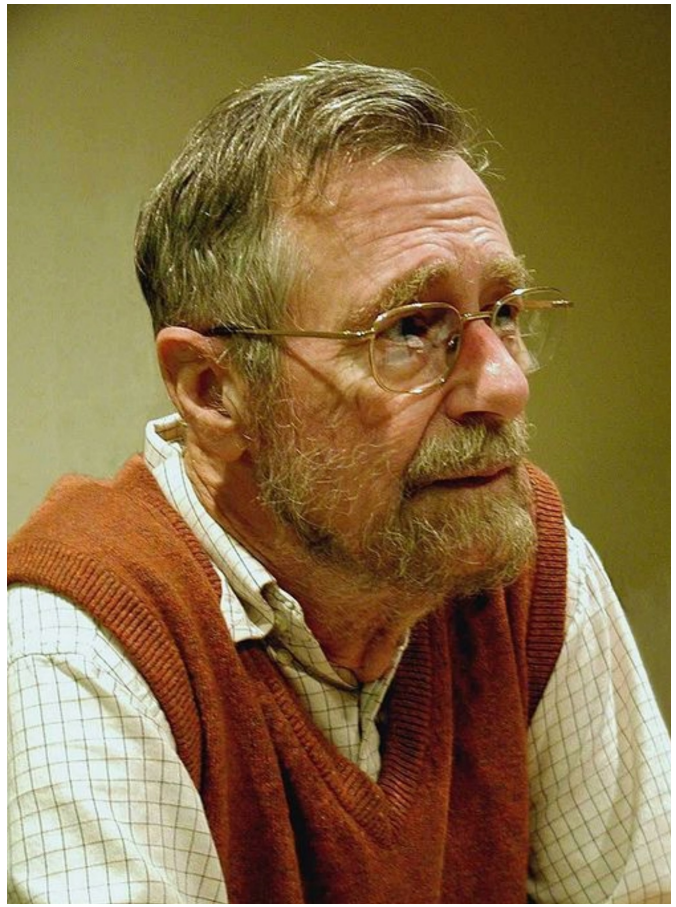
Obtain the Topology: Defining the Cost



Dijkstra's link-state routing algorithm

- Famous computer scientist
  - Programming languages
  - Distributed algorithms
  - Program verification

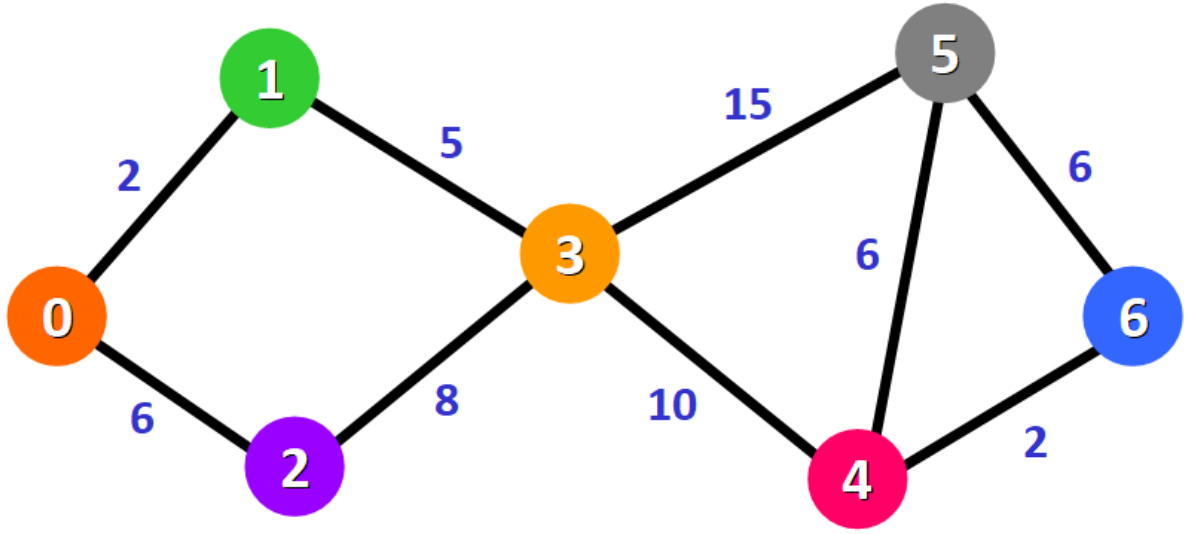
- Dijkstra's algorithm, 1969
  - Single-source shortest paths, given network with non-negative link costs



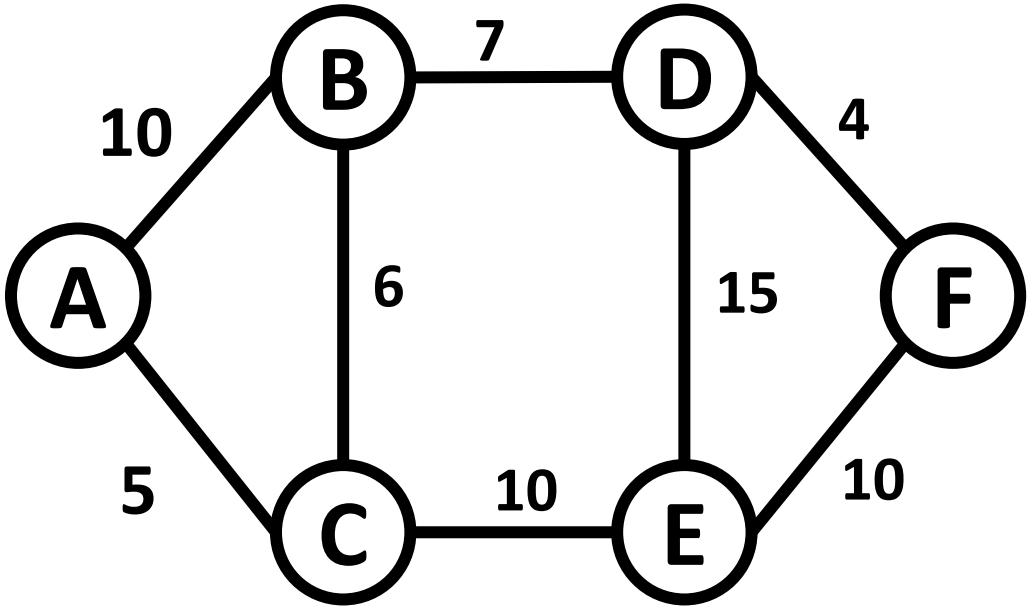
Edsger W. Dijkstra (1930-2002)

Dijkstra's link-state routing algorithm

- centralized: network topology, link costs known to *all* nodes
  - accomplished via "link state broadcast"
  - all nodes have same info
- computes least cost paths from one node ("source") to all other nodes
  - gives *forwarding table* for that node
- iterative: after *k* iterations, know least cost path to *k* destinations



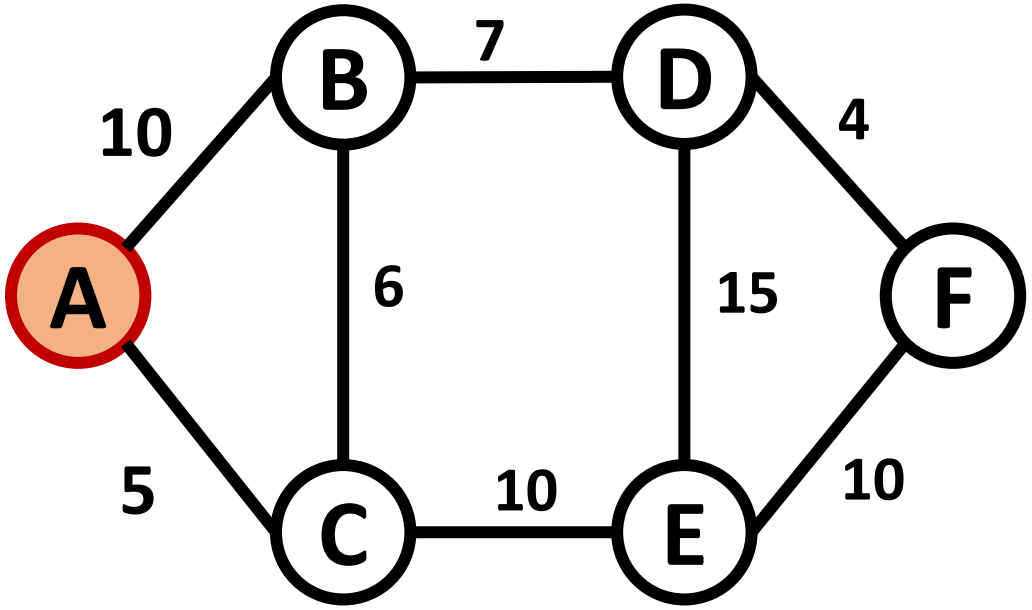
Dijkstra's link-state routing algorithm



Initialization

Visited Node	Shorted Distance
	∞
	∞
	∞
	∞
	∞
	∞

Dijkstra's link-state routing algorithm

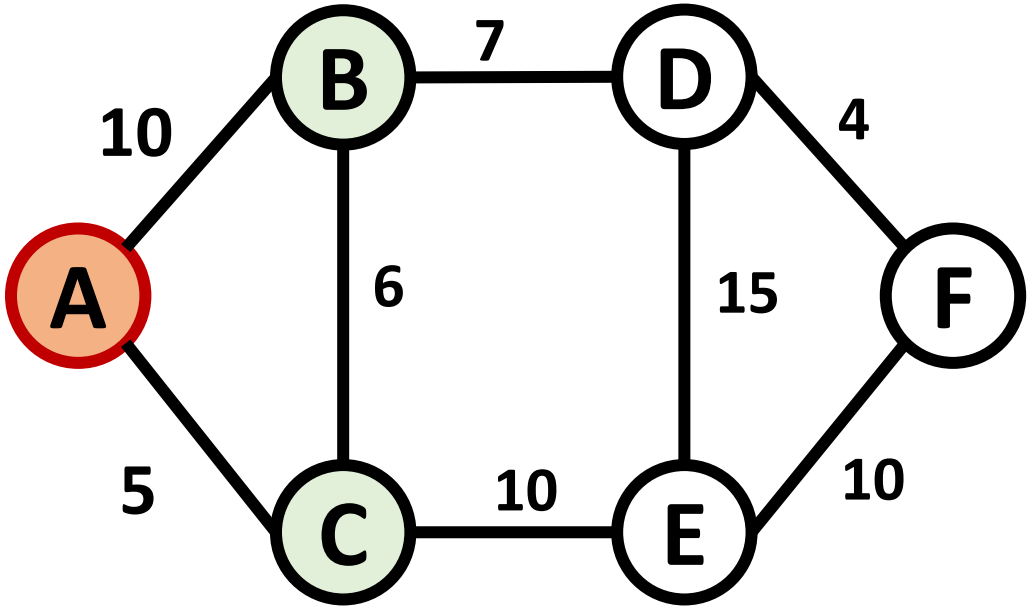


Initialization

Visited Node
A

Shorted Distance	
A	0
B	∞
C	∞
D	∞
E	∞
F	∞

Dijkstra's link-state routing algorithm



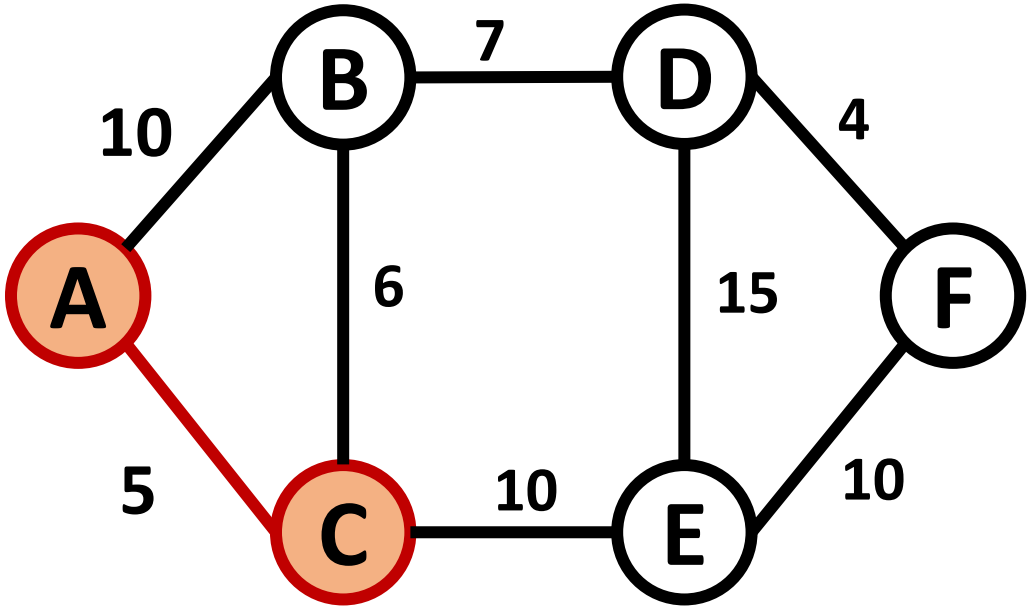
Initialization

Visited Node
A

Shorted Distance	
A	0
B	10
C	5
D	∞
E	∞
F	∞

Check the distance between the source and all its neighbors

Dijkstra's link-state routing algorithm



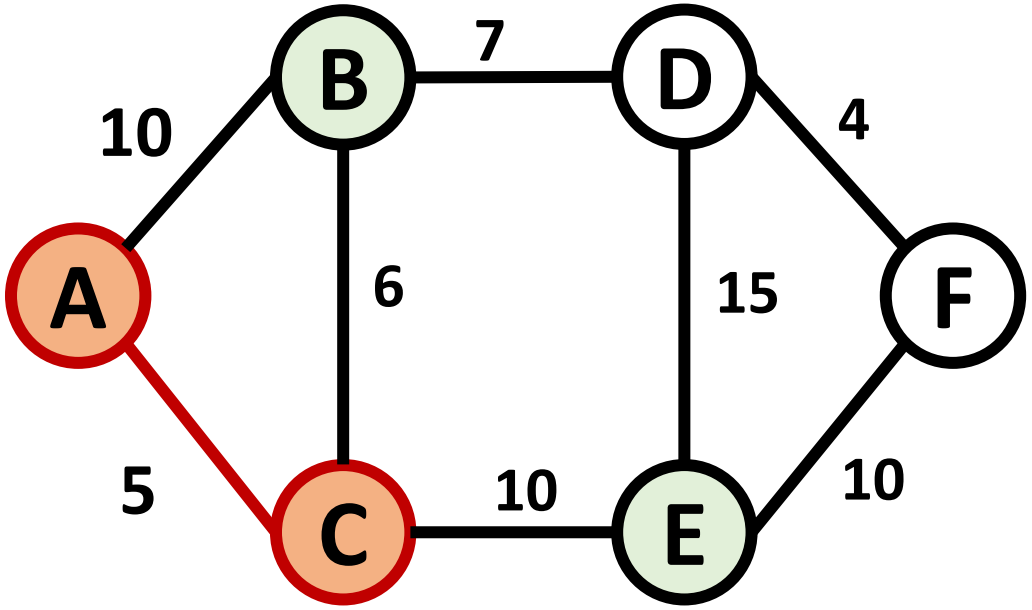
Initialization

Visited Node
A
C

Shorted Distance	
A	0
B	10
C	5
D	∞
E	∞
F	∞

Mark the selected neighbor (C) as visited

Dijkstra's link-state routing algorithm



A->B 10

A->C->B 11

A->C->E 15

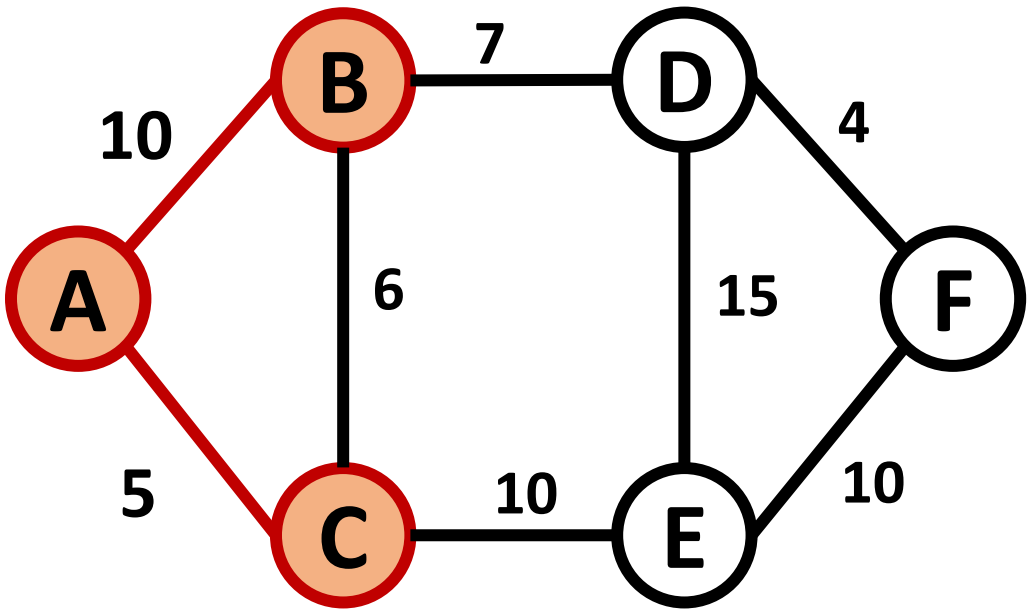
Initialization

Visited Node
A
C

Shorted Distance	
A	0
B	10
C	5
D	∞
E	∞
F	∞

Check the distance between source and all visited nodes' neighbors

Dijkstra's link-state routing algorithm



A->B 10

A->C->B 11

A->C->E 15

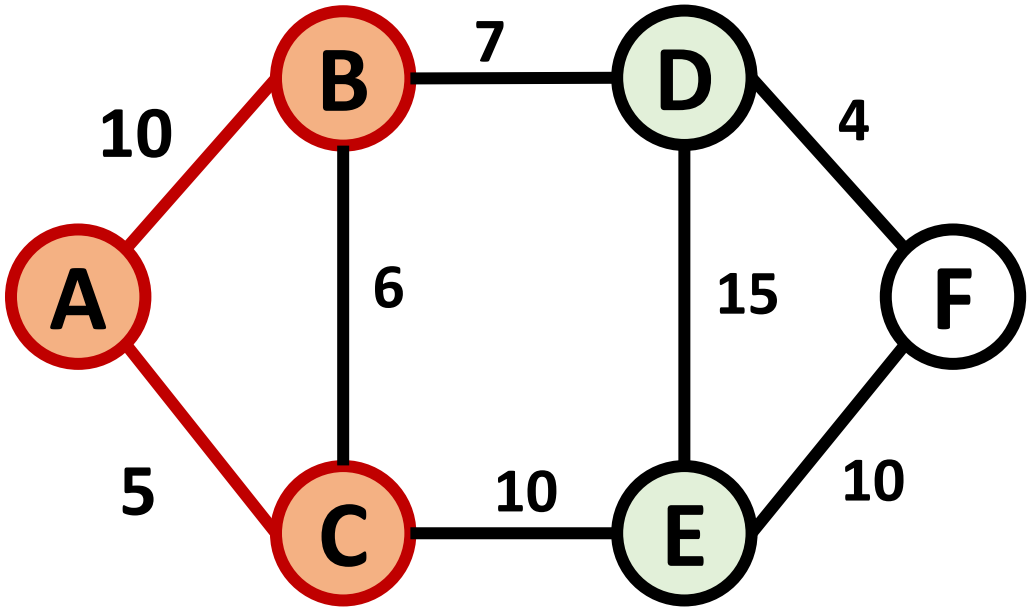
Initialization

Visited Node
A
B
C

Shorted Distance	
A	0
B	10
C	5
D	∞
E	∞
F	∞

Mark the selected neighbor (B) as visited

Dijkstra's link-state routing algorithm



A->B->D 17

A->C->E 15

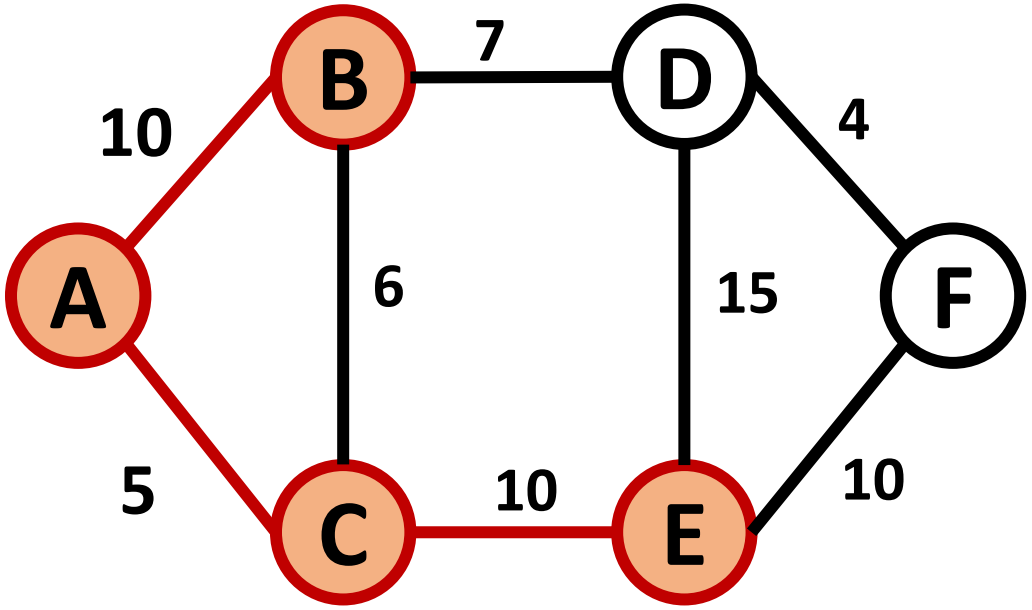
Initialization

Visited Node
A
B
C

Shorted Distance	
A	0
B	10
C	5
D	∞
E	∞
F	∞

Check the distance between source and all visited nodes' neighbors

Dijkstra's link-state routing algorithm



A->B->D 17

A->C->E 15

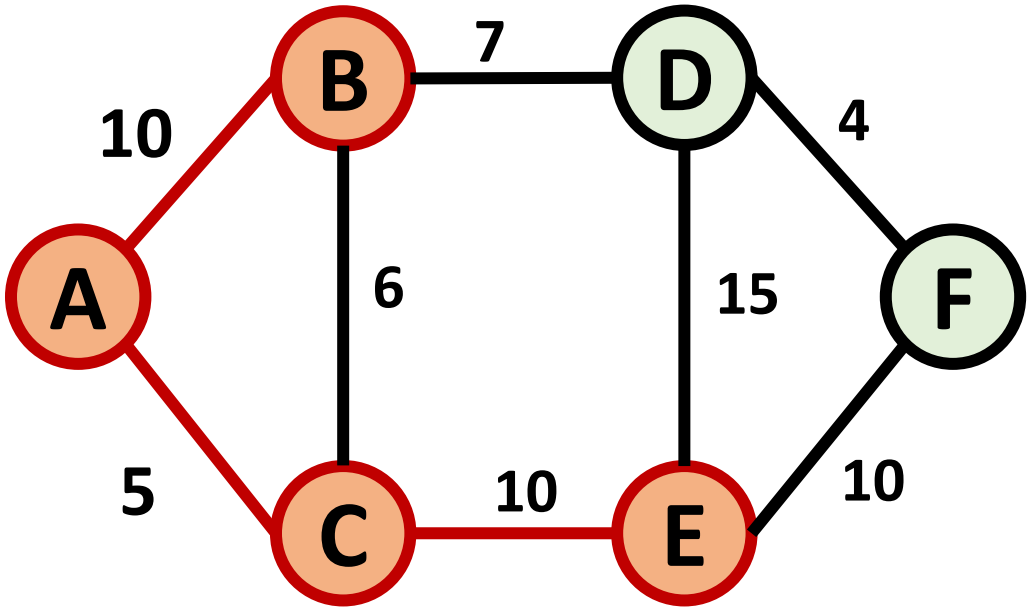
Initialization

Visited Node
A
B
C
E

Shorted Distance	
A	0
B	10
C	5
D	17
E	15
F	∞

Mark the selected neighbor (E) as visited

Dijkstra's link-state routing algorithm



A->B->D 17

A->C->E->D 30

A->C->E->F 25

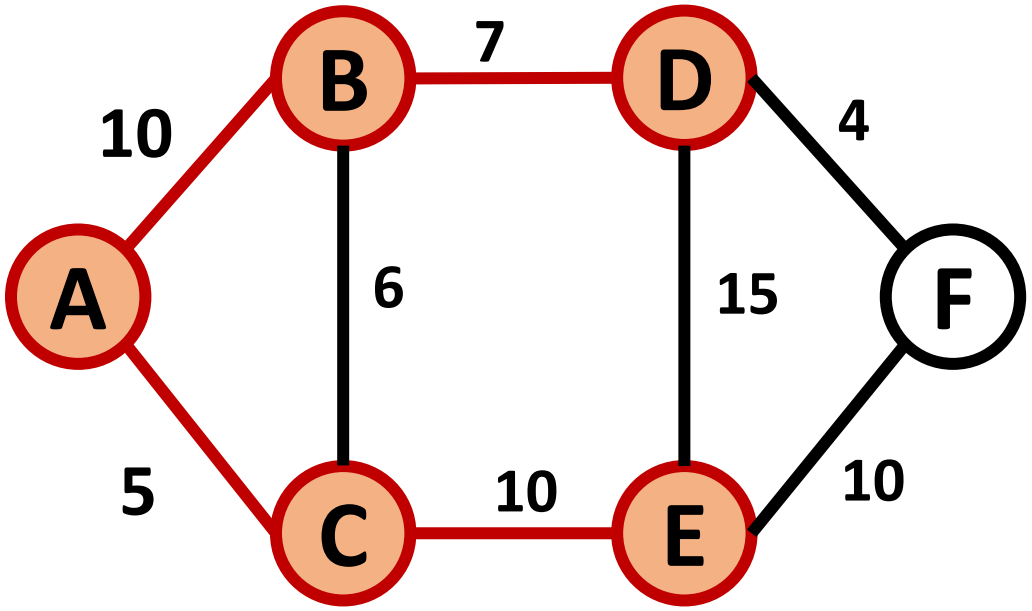
Initialization

Visited Node
A
B
C
E

Shorted Distance	
A	0
B	10
C	5
D	17
E	15
F	25

Check the distance between source and all visited nodes' neighbors

Dijkstra's link-state routing algorithm



A->B->D 17

A->C->E->D 30

A->C->E->F 25

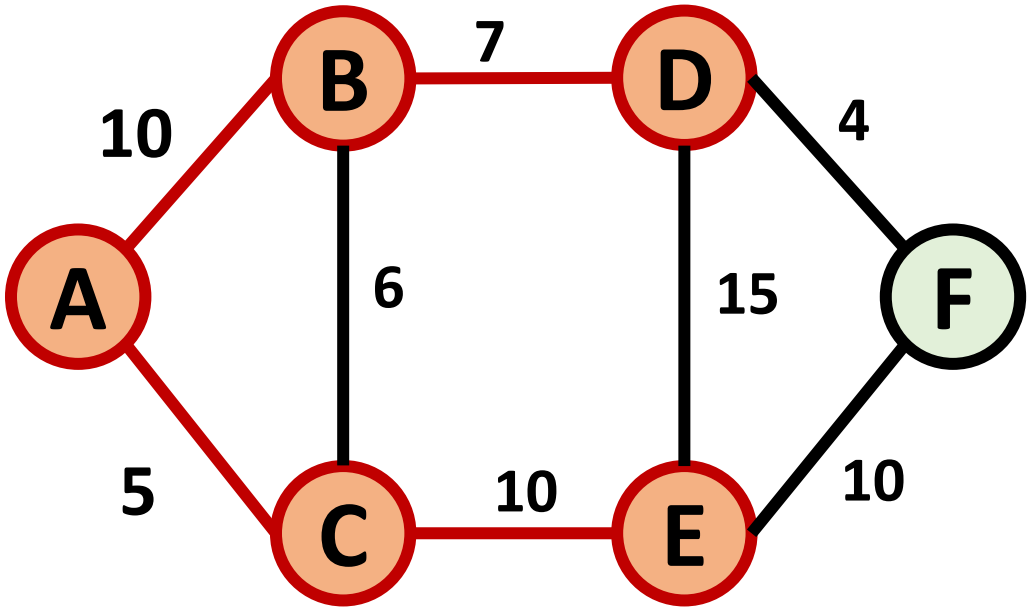
Initialization

Visited Node
A
B
C
D
E

Shorted Distance	
A	0
B	10
C	5
D	17
E	15
F	25

Mark the selected neighbor (D) as visited

Dijkstra's link-state routing algorithm



A->B->D->F 21

A->C->E->F 25

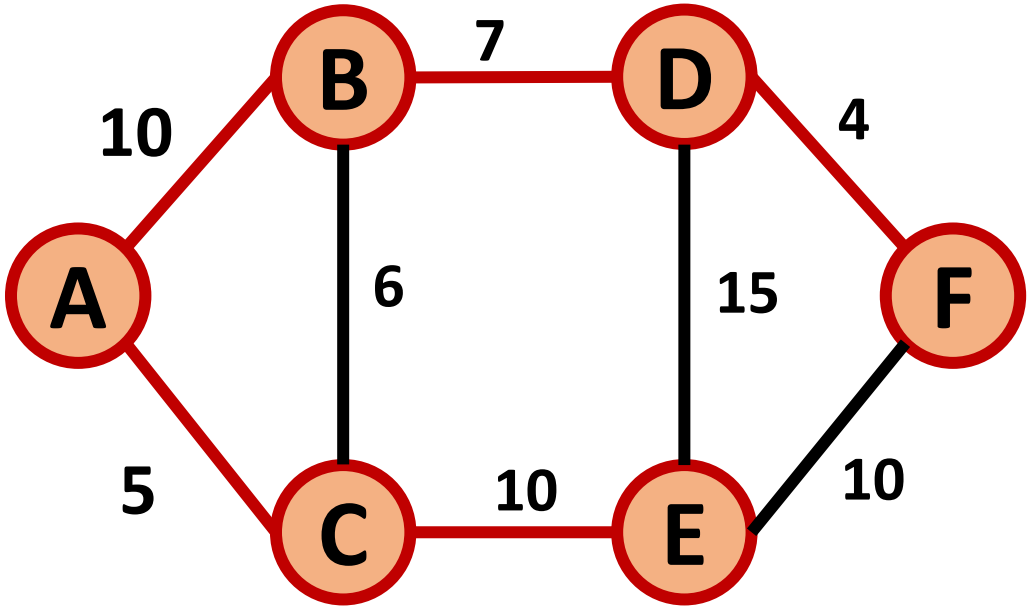
Initialization

Visited Node
A
B
C
D
E

Shorted Distance	
A	0
B	10
C	5
D	17
E	15
F	25

Check the distance between source and all visited nodes' neighbors

Dijkstra's link-state routing algorithm



A->B->D->F 21

A->C->E->F 25

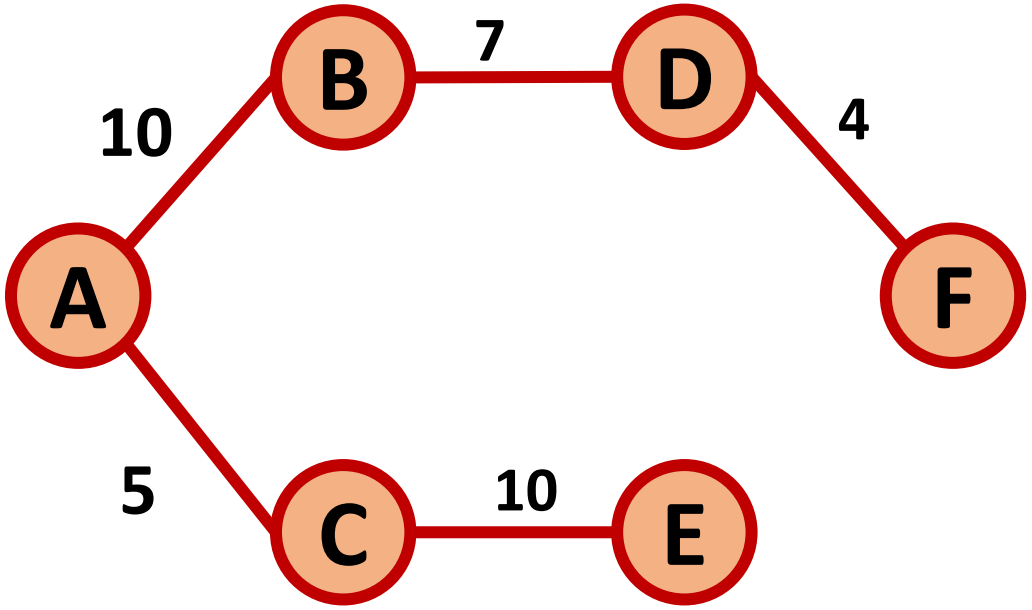
Initialization

Visited Node
A
B
C
D
E
F

Shorted Distance	
A	0
B	10
C	5
D	17
E	15
F	21

Mark the selected neighbor (F) as visited

Dijkstra's link-state routing algorithm: Result



Link	Next Hop	Overall Cost
A->B	B	10
A->C	C	5
A->D	B	17
A->E	C	15
A->F	B	21