IPADS INSTITUTE OF PARALLEL AND DISTRIBUTED SYSTEMS

上海交通大學 SHANGHAI JIAO TONG UNIVERSITY

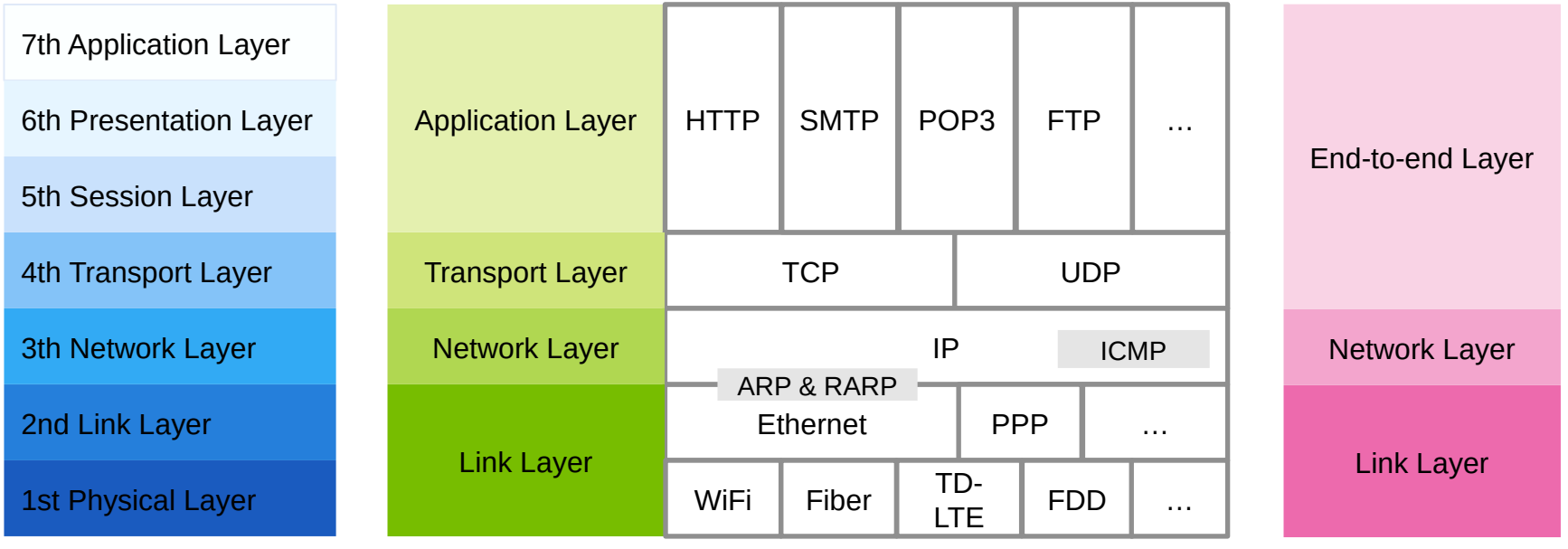
Network Layer
All about routing

IPADS, Shanghai Jiao Tong University

https://www.sjtu.edu.cn

Credits to Katrina LaCurts, MIT

Review: OSI, TCP/IP & Protocol Stack



Review: Simple Parity Check

2 bits -> 3 bits

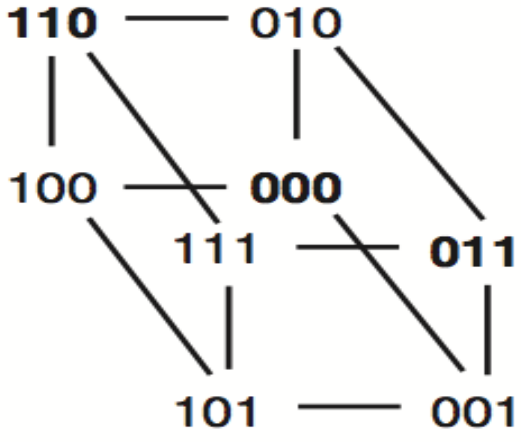
- Detect 1-bit errors
- 8 patterns total

Only 4 correct patterns

- 00 -> 000
- 11 -> 110
- 10 -> 101
- 01 -> 011

Hamming distance of this code is 2

- 1-bit flipping will cause incorrect pattern



NETWORK LAYER

IP: Best-effort Network

1. Best-effort network

- If it cannot dispatch, may discard a packet

2. Guaranteed-delivery network

- Also called store-and-forward network, no discarding data
- Work with complete messages rather than packets
- Use disk for buffering to handle peaks
- Tracks individual message to make sure none are lost

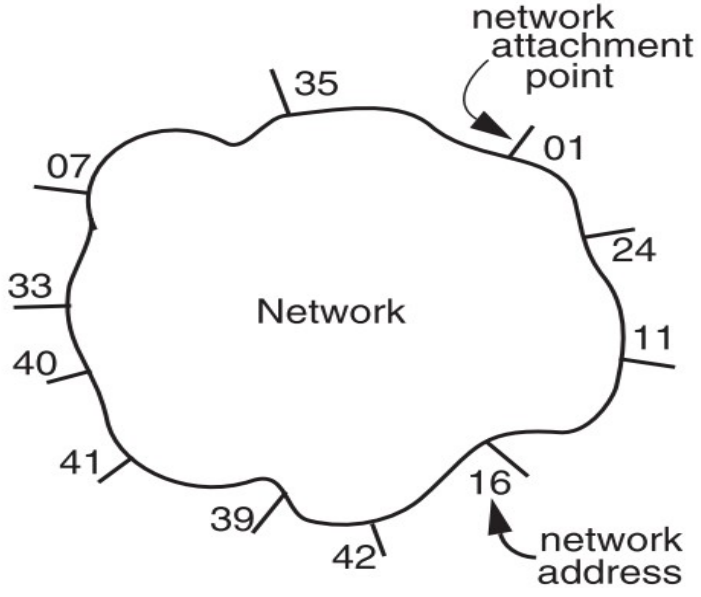
In real world

- No absolute guarantee
- Guaranteed-delivery: higher layer; best-effort: lower layer

The Network Layer

Addressing interface

- Network attachment points
- Network address
- Source & destination



NETWORK_SEND

- (segment_buffer, destnation, network_protocol, end_layer_protocol)

NETWORK_HANDLE

- (packet, network_protocol)

Managing the Forwarding Table: Routing

Routing (or path-finding)

- Constructing the tables

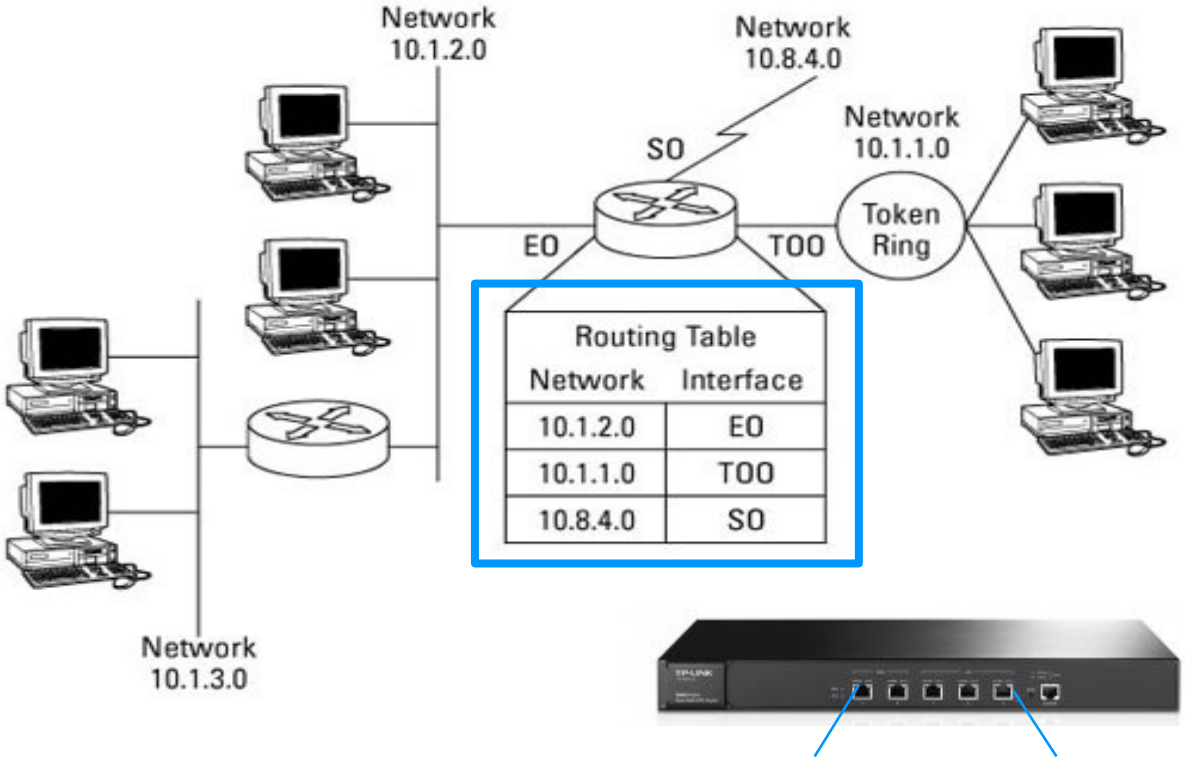
Impractical by hand

- Determining the best paths requires calculation
- Recalculating the table when links change
- Recalculating the table when link fails
- Adapt according to traffic congestion

Static routing vs. adaptive routing

- Adaptive routing requires exchange of info

IP Route Table



Control-plane VS. Data-plane

Control-plane

- Control the data flow by defining rules
- E.g., the routing algorithm

Data-plane

- Copies data according to the rules
- Performance critical
- E.g., the IP forwarding process

Routing

How to generate the routing table?

Goal of A Routing Protocol

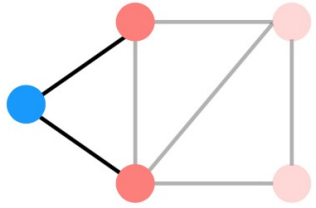
Allow each switch to know, for every node dst in the network, a route to dst

Allow each switch to know, for every node dst in the network, a minimum-cost route to dst

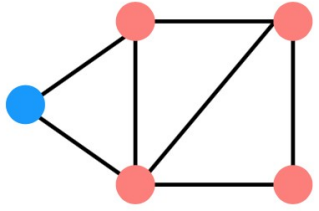
Build a routing table at each switch, such that routing_table[dst] contains a minimum-cost route to dst

Distributed Routing: 3 Steps in General

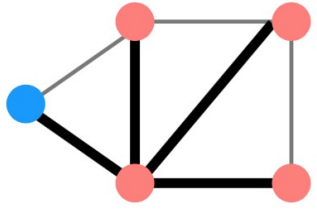
1. Nodes learn about their neighbors via the HELLO protocol



2. Nodes learn about other reachable nodes via advertisements



3. Nodes determine the minimum-cost routes (of the routes they know about)



Two Types of Routing Protocol

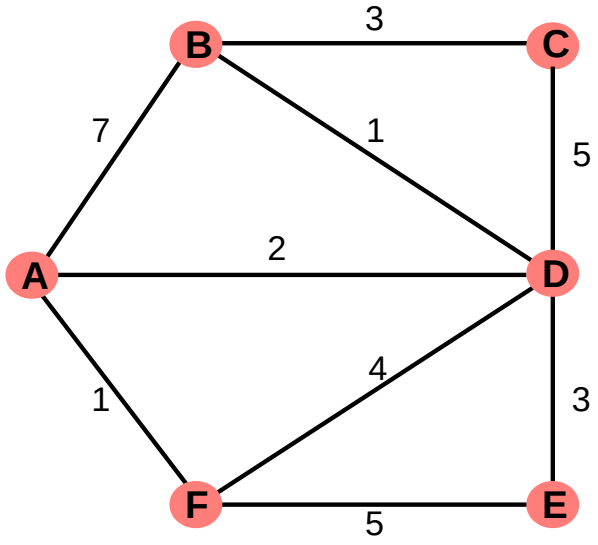
Protocol 1: Link-state

- A node's advertisements contain a list of its neighbors and its link costs to those nodes
- Nodes advertise to everyone their costs to their neighbors
  - via *flooding*
- Integrate using **Dijkstra's shortest path algorithm**

Protocol 2: Distance-vector

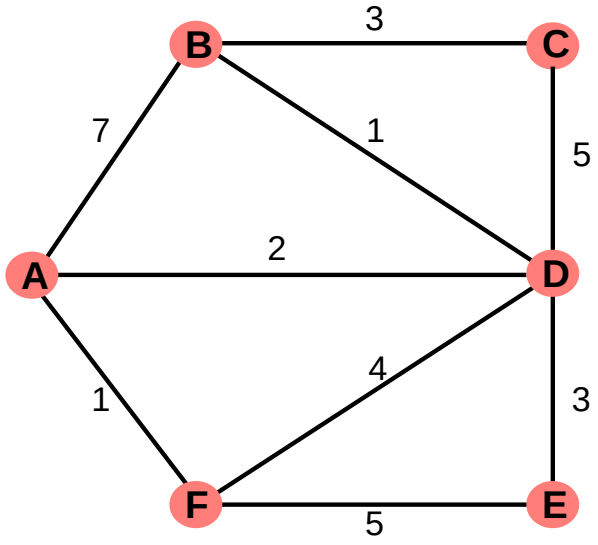
- Nodes advertise to neighbors with their cost to all known nodes
- Update routes via **Bellman-Ford** integration

link-state routing: disseminate full topology information so that nodes can run a shortest-path algorithm



link state

link-state routing: disseminate full topology information so that nodes can run a shortest-path algorithm

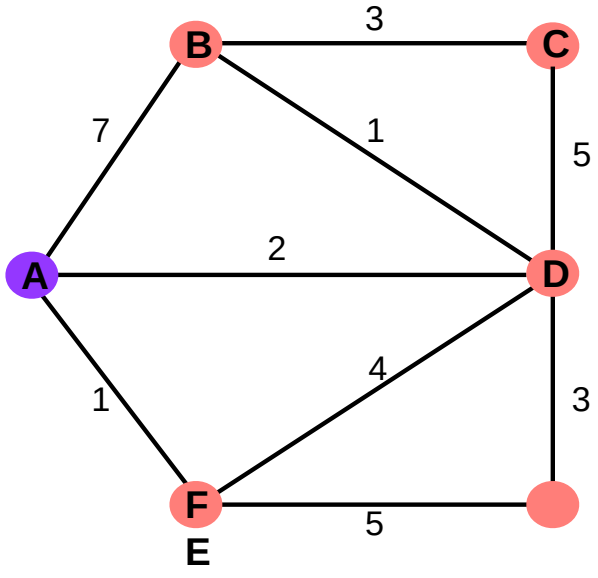


link state

what's in an advertisement

its **link costs** to each of its **neighbors**

link-state routing: disseminate full topology information so that nodes can run a shortest-path algorithm



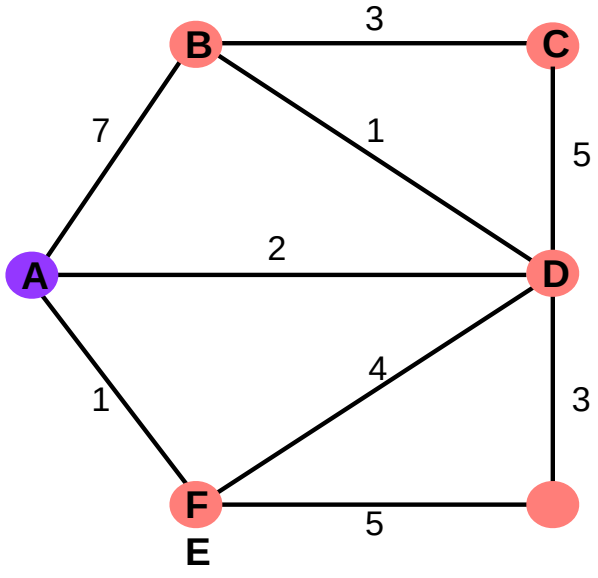
A's advertisement: [(B,7),(D,2),(F,1)]

link state

what's in an advertisement

its **link costs** to each of its **neighbors**

link-state routing: disseminate full topology information so that nodes can run a shortest-path algorithm



A's advertisement: [(B,7),(D,2),(F,1)]

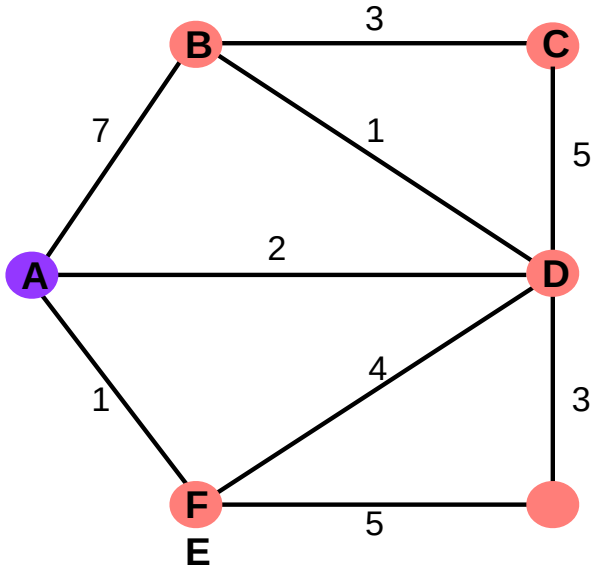
link state

what's in an advertisement

its **link costs** to each of its **neighbors**

who gets a node's advertisement

link-state routing: disseminate full topology information so that nodes can run a shortest-path algorithm



A's advertisement: [(B,7),(D,2),(F,1)]

link state

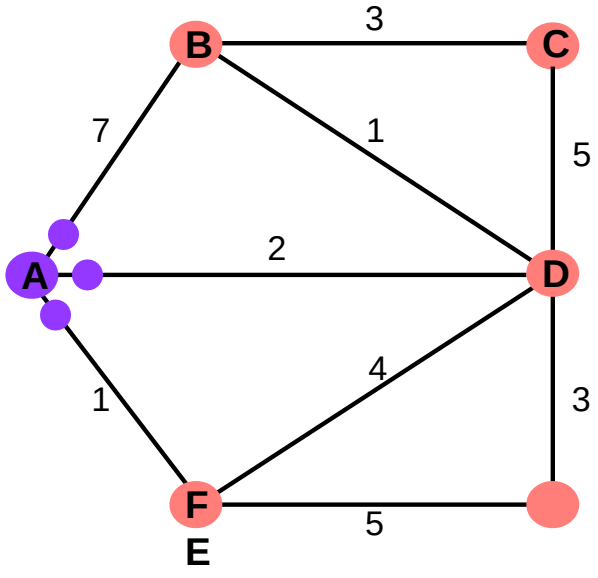
what's in an advertisement

its **link costs** to each of its **neighbors**

who gets a node's advertisement

effectively, **every other node** (via flooding)

link-state routing: disseminate full topology information so that nodes can run a shortest-path algorithm



A's advertisement: [(B,7),(D,2),(F,1)]

link state

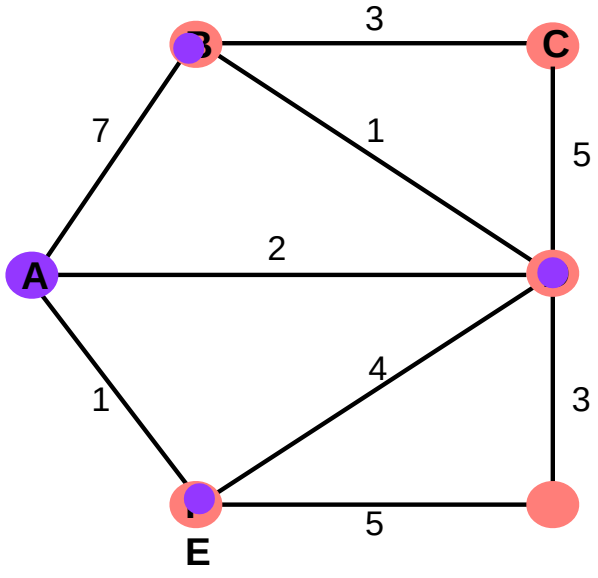
what's in an advertisement

its **link costs** to each of its **neighbors**

who gets a node's advertisement

effectively, **every other node** (via flooding)

link-state routing: disseminate full topology information so that nodes can run a shortest-path algorithm



A's advertisement: [(B,7),(D,2),(F,1)]

link state

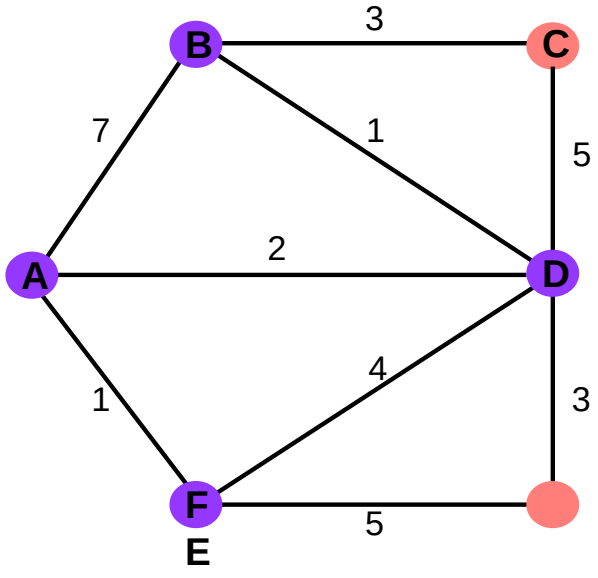
what's in an advertisement

its **link costs** to each of its **neighbors**

who gets a node's advertisement

effectively, **every other node** (via flooding)

link-state routing: disseminate full topology information so that nodes can run a shortest-path algorithm



A's advertisement: [(B,7),(D,2),(F,1)]

link state

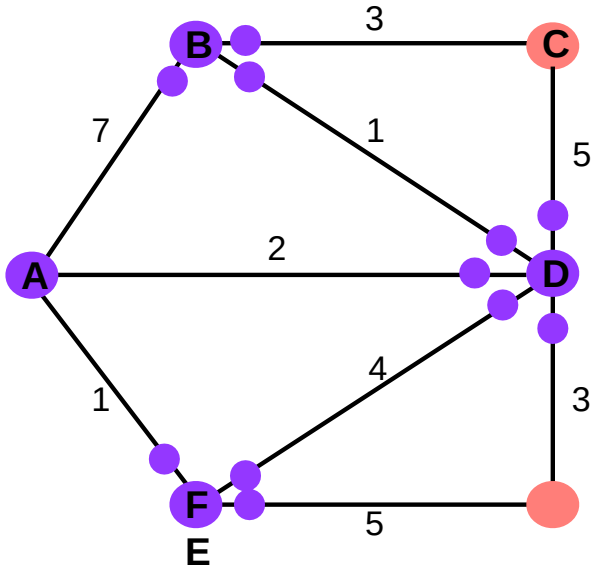
what's in an advertisement

its **link costs** to each of its **neighbors**

who gets a node's advertisement

effectively, **every other node** (via flooding)

link-state routing: disseminate full topology information so that nodes can run a shortest-path algorithm



A's advertisement: [(B,7),(D,2),(F,1)]

link state

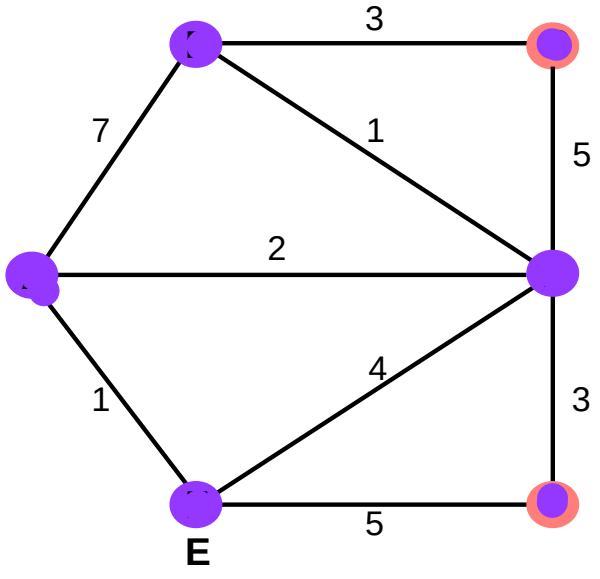
what's in an advertisement

its **link costs** to each of its **neighbors**

who gets a node's advertisement

effectively, **every other node** (via flooding)

link-state routing: disseminate full topology information so that nodes can run a shortest-path algorithm



A's advertisement: [(B,7),(D,2),(F,1)]

link state

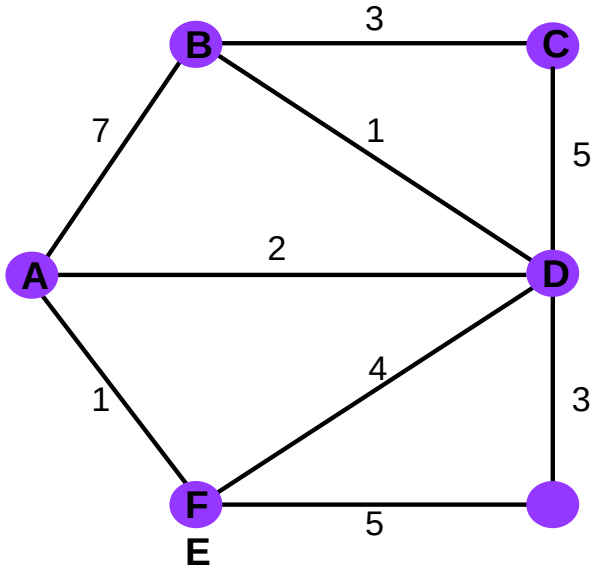
what's in an advertisement

its **link costs** to each of its **neighbors**

who gets a node's advertisement

effectively, **every other node** (via flooding)

link-state routing: disseminate full topology information so that nodes can run a shortest-path algorithm



A's advertisement: [(B,7),(D,2),(F,1)]

link state

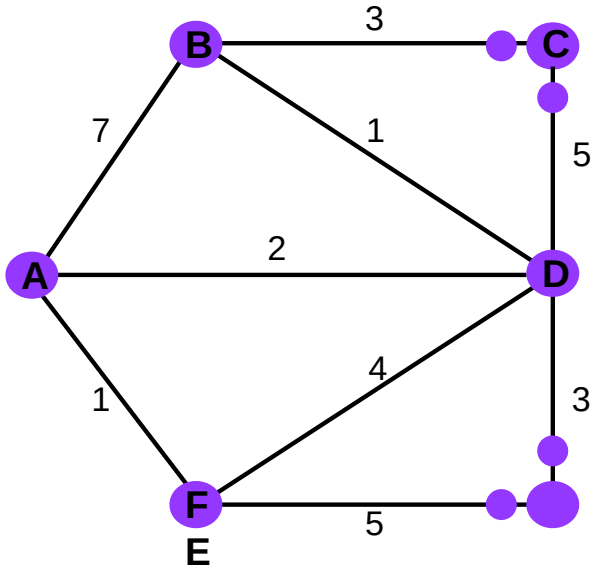
what's in an advertisement

its **link costs** to each of its **neighbors**

who gets a node's advertisement

effectively, **every other node** (via flooding)

link-state routing: disseminate full topology information so that nodes can run a shortest-path algorithm



A's advertisement: [(B,7),(D,2),(F,1)]

link state

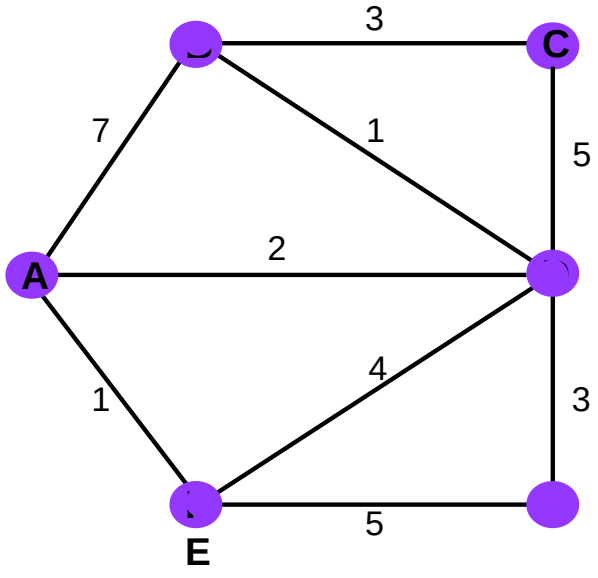
what's in an advertisement

its **link costs** to each of its **neighbors**

who gets a node's advertisement

effectively, **every other node** (via flooding)

link-state routing: disseminate full topology information so that nodes can run a shortest-path algorithm



A's advertisement: [(B,7),(D,2),(F,1)]

link state

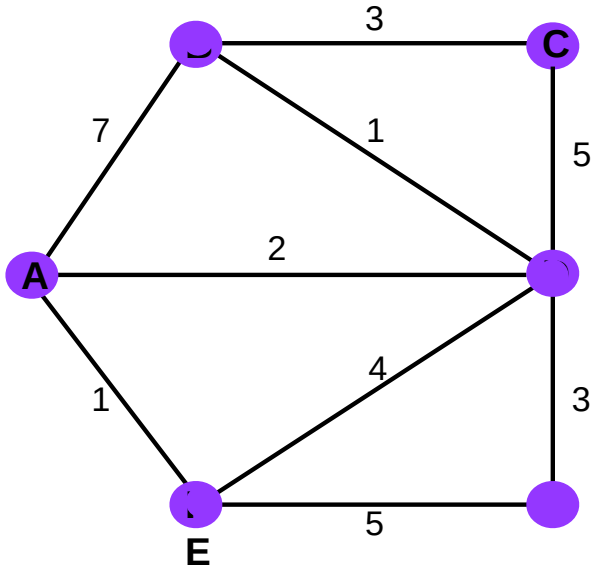
what's in an advertisement

its **link costs** to each of its **neighbors**

who gets a node's advertisement

effectively, **every other node** (via flooding)

link-state routing: disseminate full topology information so that nodes can run a shortest-path algorithm



A's advertisement: [(B,7),(D,2),(F,1)]

nodes keep track of which advertisements they've forwarded so that they don't re-forward them

link state

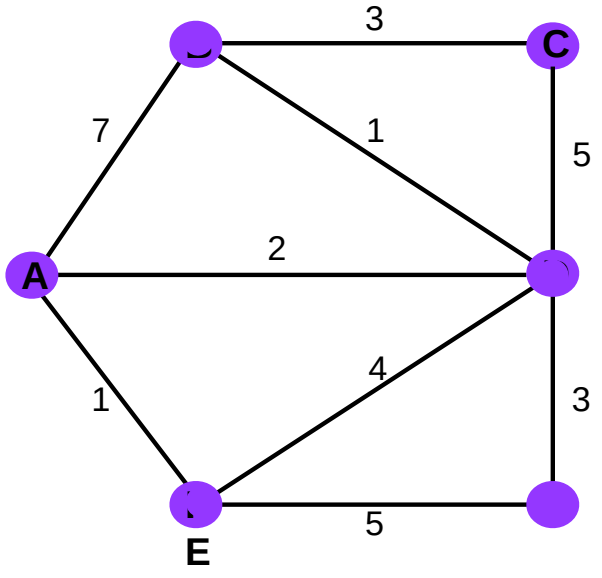
what's in an advertisement

its **link costs** to each of its **neighbors**

who gets a node's advertisement

effectively, **every other node** (via flooding)

link-state routing: disseminate full topology information so that nodes can run a shortest-path algorithm



A's advertisement: [(B,7),(D,2),(F,1)]

nodes keep track of which advertisements they've forwarded so that they don't re-forward them

they can also be a bit smarter about flooding, and not forward an advertisement back to the node that sent it

link state

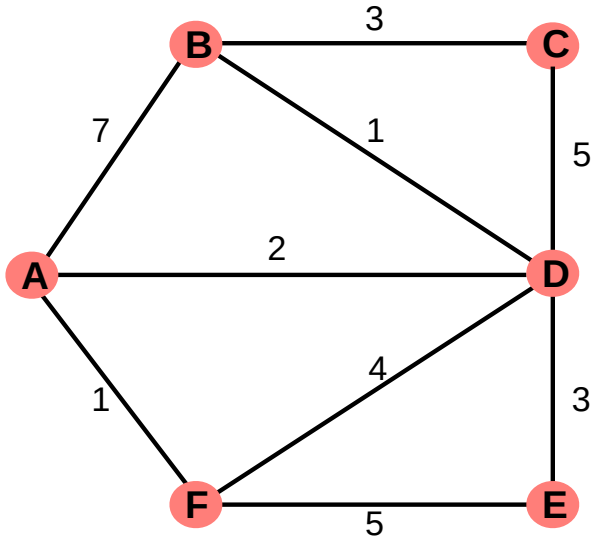
what's in an advertisement

its **link costs** to each of its **neighbors**

who gets a node's advertisement

effectively, **every other node** (via flooding)

link-state routing: disseminate full topology information so that nodes can run a shortest-path algorithm



nodes *integrate* advertisements by running Dijkstra's Algorithm

link state

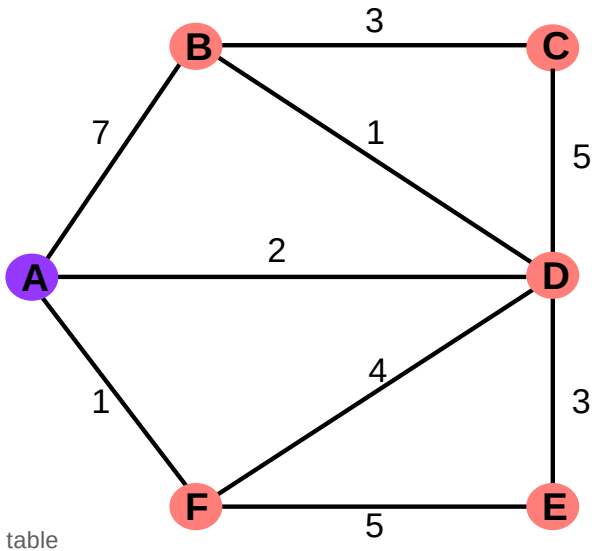
what's in an advertisement

its **link costs** to each of its **neighbors**

who gets a node's advertisement

effectively, **every other node** (via flooding)

link-state routing: disseminate full topology information so that nodes can run a shortest-path algorithm



A's routing table

dst	route	cost
B	A-B	7
C	?	∞
D	A-D	2
E	?	∞
F	A-F	1

link state

what's in an advertisement

its **link costs** to each of its **neighbors**

who gets a node's advertisement

effectively, **every other node** (via flooding)

Link-state Routing

Keep track of W, the set of nodes haven't processed yet

- Initially, W is all nodes in the network

Keep track of the current costs and routes to all of the nodes. Initially:

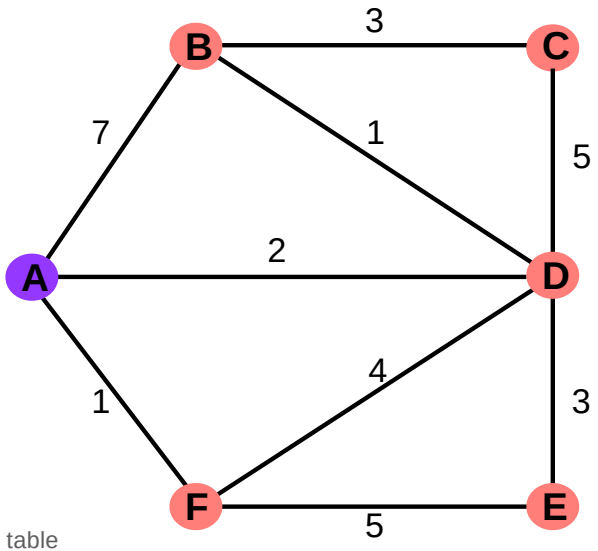
- routing_table[self] = Self; routing_table[anyone else] = ?
- cost_table[self] = 0; cost_table[anyone else] = infinity

While W is not empty:

- 1. u = the node in W we have the minimum cost to so far
- 2. Remove u from W
- 3. For every neighbor v of u:

```
d = cost_table[u] + cost(u, v)
if d < cost_table[v]
    cost_table[v] = d
    routing_table[v] = routing_table[u]
```

link-state routing: disseminate full topology information so that nodes can run a shortest-path algorithm



A's routing table

dst	route	cost
B	A-B	7
C	?	∞
D	A-D	2
E	?	∞
F	**A-F**	**1**

link state

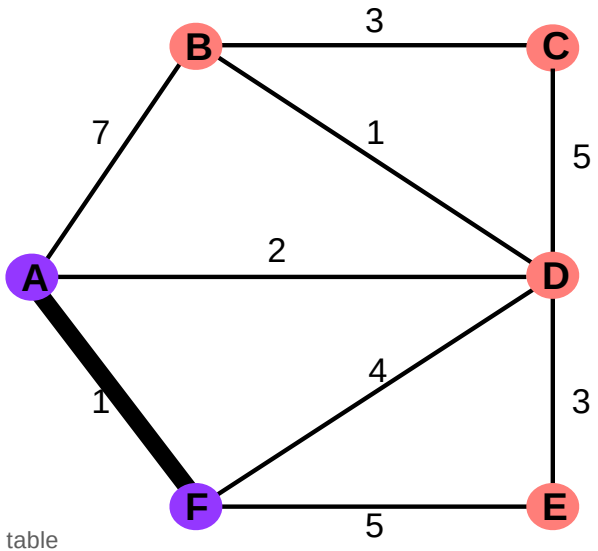
what's in an advertisement

its **link costs** to each of its **neighbors**

who gets a node's advertisement

effectively, **every other node** (via flooding)

link-state routing: disseminate full topology information so that nodes can run a shortest-path algorithm



A's routing table

dst	route	cost
B	A-B	7
C	?	∞
D	A-D	2
E	?	∞
F	**A-F**	**1**

link state

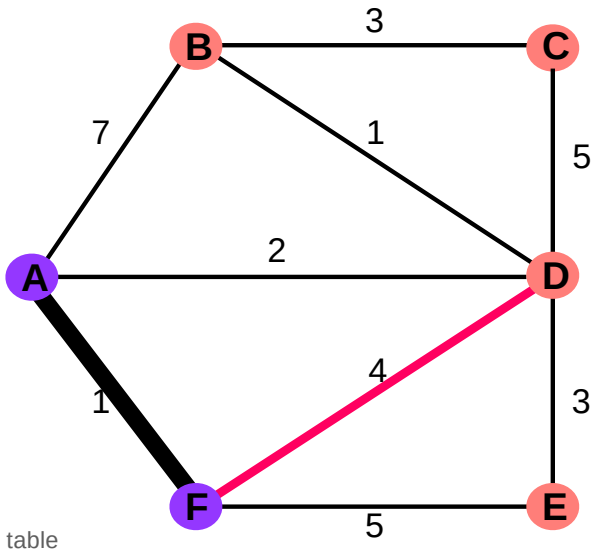
what's in an advertisement

its **link costs** to each of its **neighbors**

who gets a node's advertisement

effectively, **every other node** (via flooding)

link-state routing: disseminate full topology information so that nodes can run a shortest-path algorithm



A's routing table

dst	route	cost
B	A-B	7
C	?	∞
D	A-D	2
E	?	∞
F	**A-F**	**1**

link state

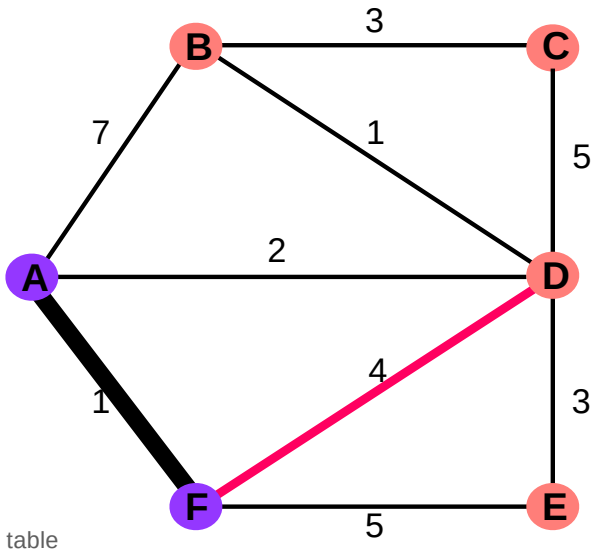
what's in an advertisement

its **link costs** to each of its **neighbors**

who gets a node's advertisement

effectively, **every other node** (via flooding)

link-state routing: disseminate full topology information so that nodes can run a shortest-path algorithm



A's routing table

dst	route	cost
B	A-B	7
C	?	∞
D	A-D	2
E	?	∞
F	**A-F**	**1**

F does not provide A with a better route to D

link state

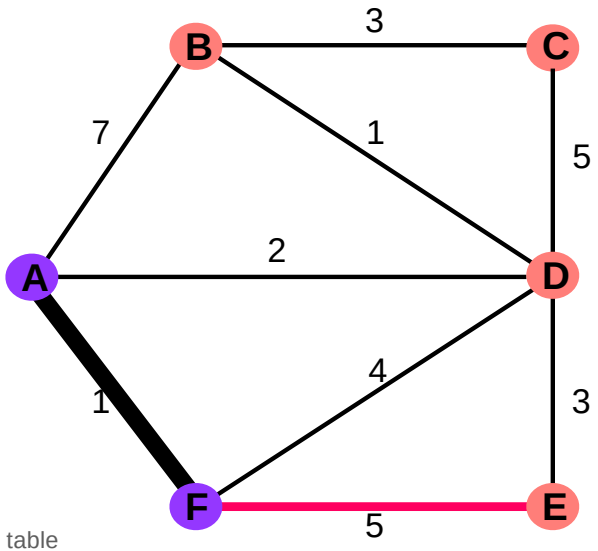
what's in an advertisement

its **link costs** to each of its **neighbors**

who gets a node's advertisement

effectively, **every other node** (via flooding)

link-state routing: disseminate full topology information so that nodes can run a shortest-path algorithm



A's routing table

dst	route	cost
B	A-B	7
C	?	∞
D	A-D	2
E	?	∞
F	**A-F**	**1**

link state

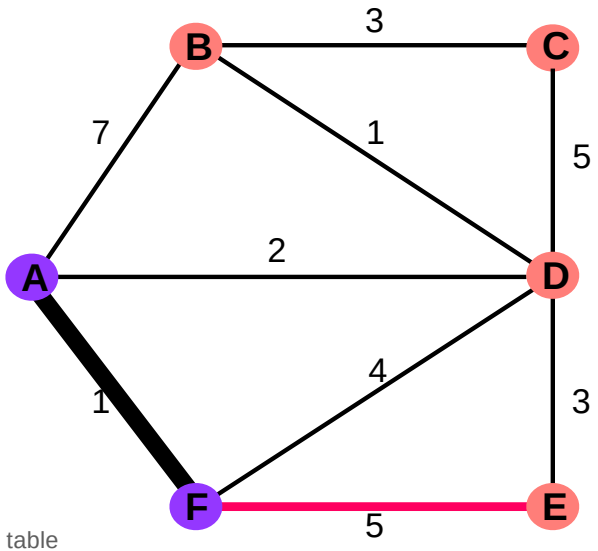
what's in an advertisement

its **link costs** to each of its **neighbors**

who gets a node's advertisement

effectively, **every other node** (via flooding)

link-state routing: disseminate full topology information so that nodes can run a shortest-path algorithm



A's routing table

dst	route	cost
B	A-B	7
C	?	∞
D	A-D	2
E	A-F	6
F	**A-F**	**1**

= the cost from A to F **+** the cost from F to E

link state

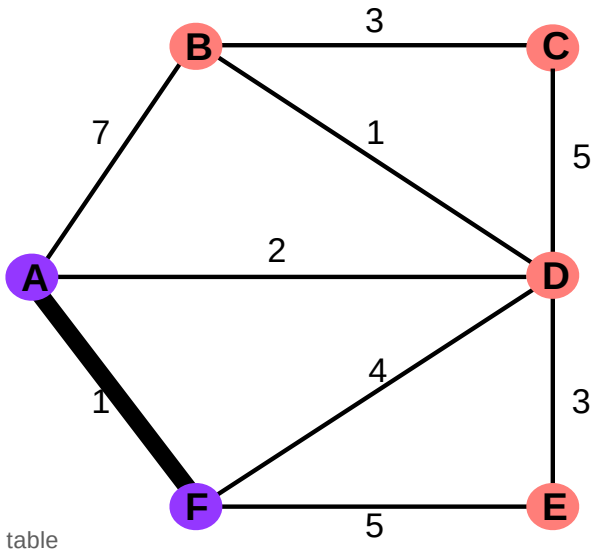
what's in an advertisement

its **link costs** to each of its **neighbors**

who gets a node's advertisement

effectively, **every other node** (via flooding)

link-state routing: disseminate full topology information so that nodes can run a shortest-path algorithm



A's routing table

dst	route	cost
B	A-B	7
C	?	∞
D	**A-D**	**2**
E	A-F	6
F	**A-F**	**1**

link state

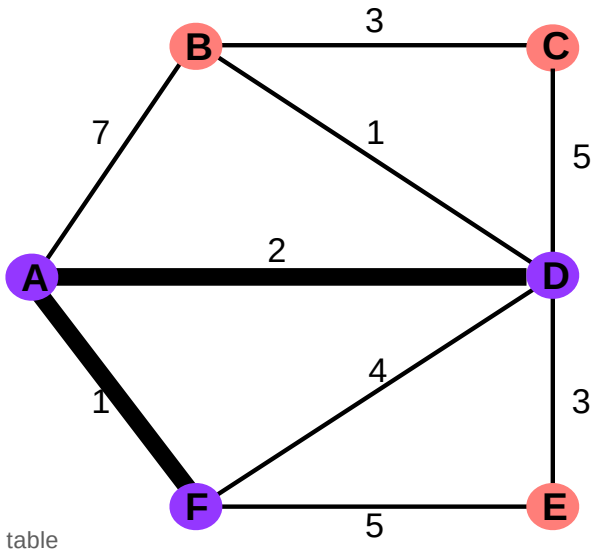
what's in an advertisement

its **link costs** to each of its **neighbors**

who gets a node's advertisement

effectively, **every other node** (via flooding)

link-state routing: disseminate full topology information so that nodes can run a shortest-path algorithm



A's routing table

dst	route	cost
B	A-B	7
C	?	∞
D	**A-D**	**2**
E	A-F	6
F	**A-F**	**1**

link state

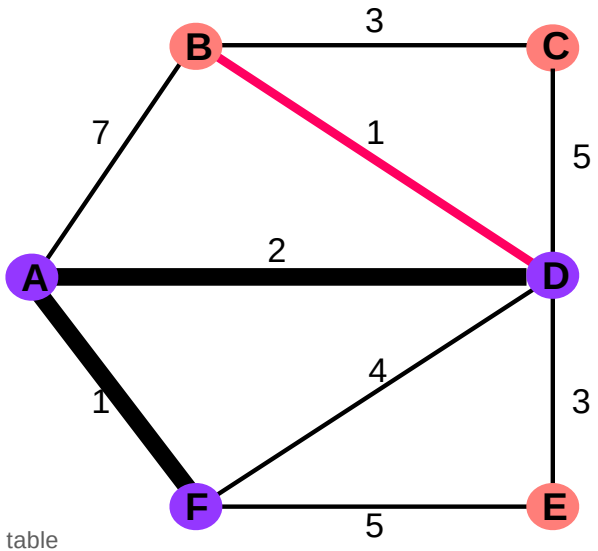
what's in an advertisement

its **link costs** to each of its **neighbors**

who gets a node's advertisement

effectively, **every other node** (via flooding)

link-state routing: disseminate full topology information so that nodes can run a shortest-path algorithm



A's routing table

dst	route	cost
B	A-B	7
C	?	∞
D	**A-D**	**2**
E	A-F	6
F	**A-F**	**1**

link state

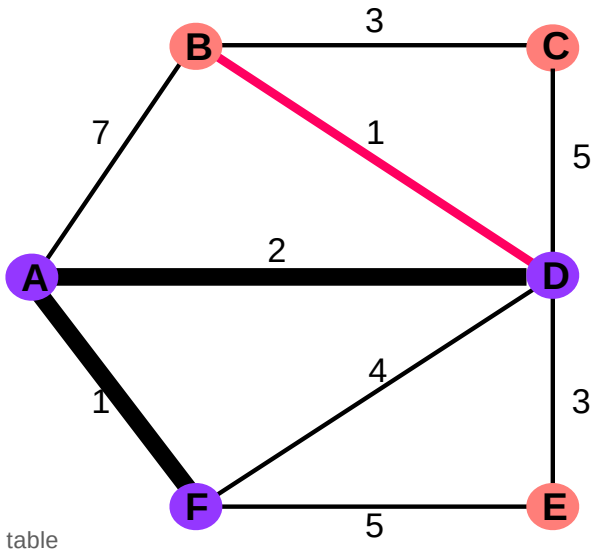
what's in an advertisement

its **link costs** to each of its **neighbors**

who gets a node's advertisement

effectively, **every other node** (via flooding)

link-state routing: disseminate full topology information so that nodes can run a shortest-path algorithm



A's routing table

dst	route	cost
B	A-B	3
C	?	∞
D	**A-D**	**2**
E	A-F	6
F	**A-F**	**1**

link state

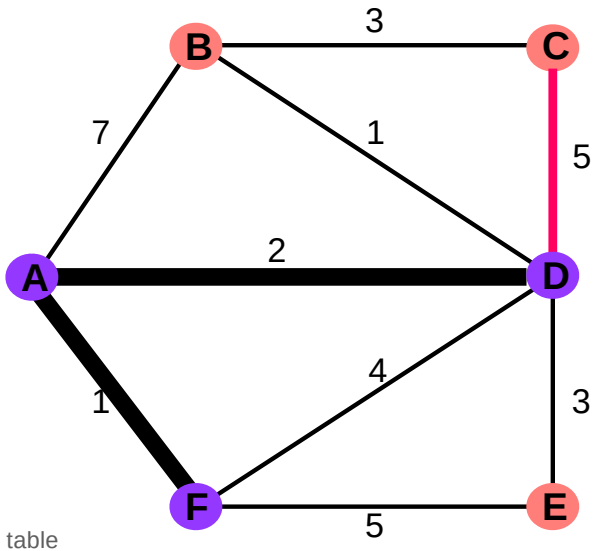
what's in an advertisement

its **link costs** to each of its **neighbors**

who gets a node's advertisement

effectively, **every other node** (via flooding)

link-state routing: disseminate full topology information so that nodes can run a shortest-path algorithm



A's routing table

dst	route	cost
B	A-B	3
C	?	∞
D	**A-D**	**2**
E	A-F	6
F	**A-F**	**1**

link state

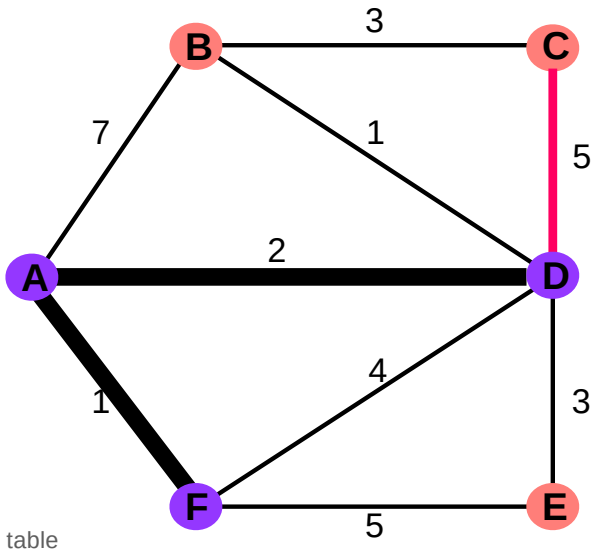
what's in an advertisement

its **link costs** to each of its **neighbors**

who gets a node's advertisement

effectively, **every other node** (via flooding)

link-state routing: disseminate full topology information so that nodes can run a shortest-path algorithm



A's routing table

dst	route	cost
B	A-B	3
C	A-D	7
D	**A-D**	**2**
E	A-F	6
F	**A-F**	**1**

link state

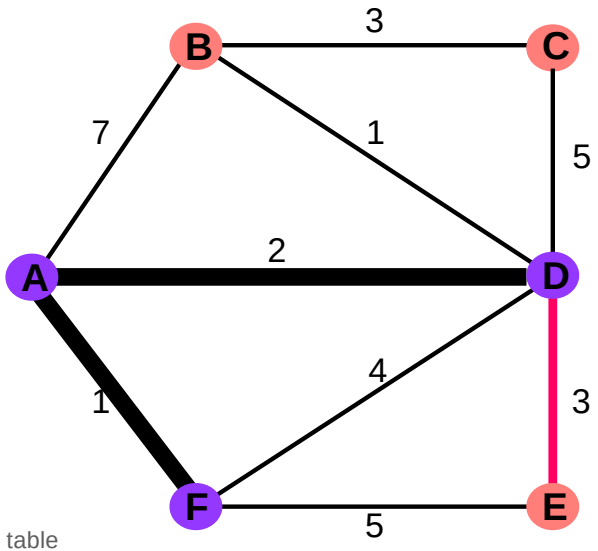
what's in an advertisement

its **link costs** to each of its **neighbors**

who gets a node's advertisement

effectively, **every other node** (via flooding)

link-state routing: disseminate full topology information so that nodes can run a shortest-path algorithm



A's routing table

dst	route	cost
B	A-B	3
C	A-D	7
D	**A-D**	**2**
E	A-F	6
F	**A-F**	**1**

link state

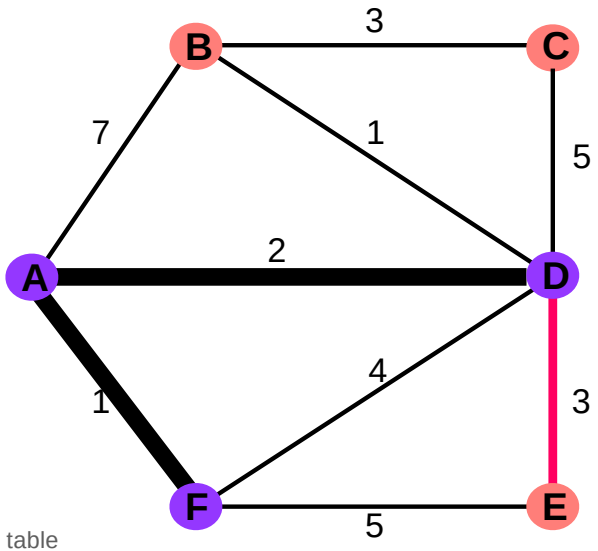
what's in an advertisement

its **link costs** to each of its **neighbors**

who gets a node's advertisement

effectively, **every other node** (via flooding)

link-state routing: disseminate full topology information so that nodes can run a shortest-path algorithm



A's routing table

dst	route	cost
B	A-B	3
C	A-D	7
D	**A-D**	**2**
E	A-D	5
F	**A-F**	**1**

link state

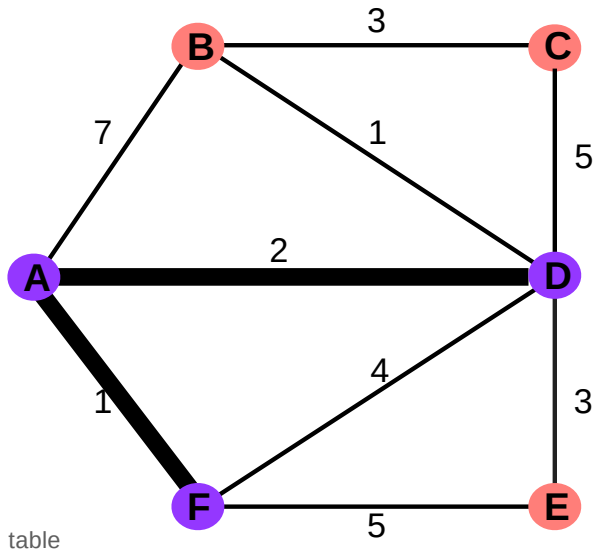
what's in an advertisement

its **link costs** to each of its **neighbors**

who gets a node's advertisement

effectively, **every other node** (via flooding)

link-state routing: disseminate full topology information so that nodes can run a shortest-path algorithm



A's routing table

dst	route	cost
B	A-B	3
C	A-D	7
D	**A-D**	**2**
E	A-D	5
F	**A-F**	**1**

we don't need to "visit" F; we already know the shortest path to it

link state

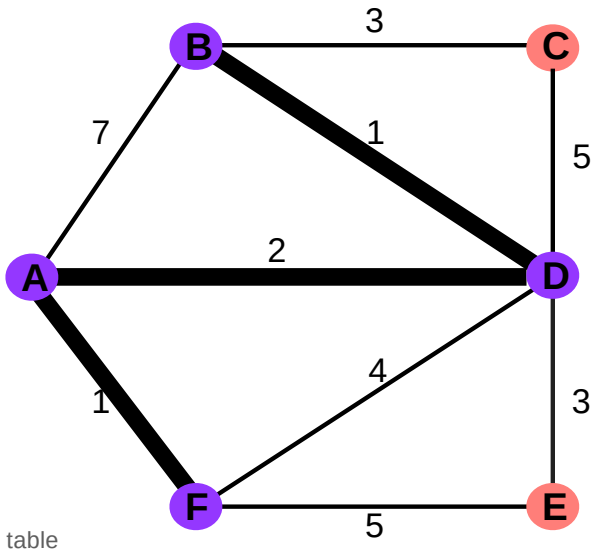
what's in an advertisement

its **link costs** to each of its **neighbors**

who gets a node's advertisement

effectively, **every other node** (via flooding)

link-state routing: disseminate full topology information so that nodes can run a shortest-path algorithm



A's routing table

dst	route	cost
B	**A-B**	**3**
C	A-D	7
D	**A-D**	**2**
E	A-D	5
F	**A-F**	**1**

link state

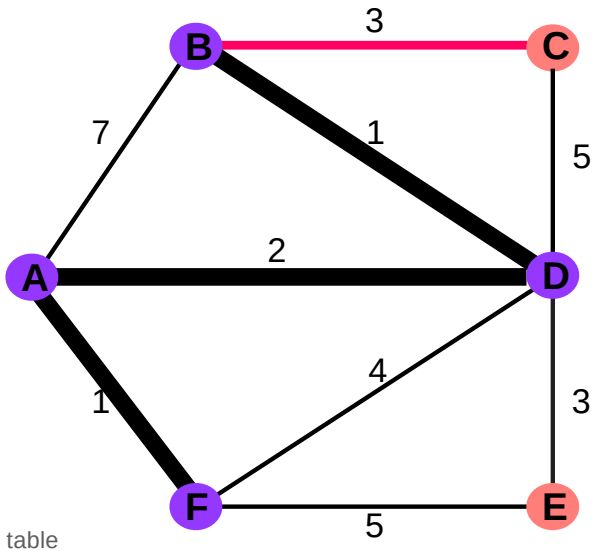
what's in an advertisement

its **link costs** to each of its **neighbors**

who gets a node's advertisement

effectively, **every other node** (via flooding)

link-state routing: disseminate full topology information so that nodes can run a shortest-path algorithm



A's routing table

dst	route	cost
B	**A-B**	**3**
C	A-D	7
D	**A-D**	**2**
E	A-D	5
F	**A-F**	**1**

link state

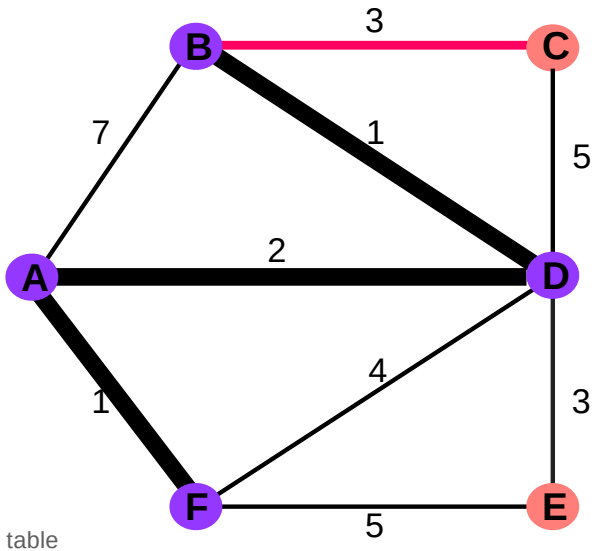
what's in an advertisement

its **link costs** to each of its **neighbors**

who gets a node's advertisement

effectively, **every other node** (via flooding)

link-state routing: disseminate full topology information so that nodes can run a shortest-path algorithm



A's routing table

dst	route	cost
B	**A-B**	**3**
C	A-D	6
D	**A-D**	**2**
E	A-D	5
F	**A-F**	**1**

notice that A's *route* doesn't change, but the cost needs to update
(and the actual path of the packets from A to C has changed)

link state

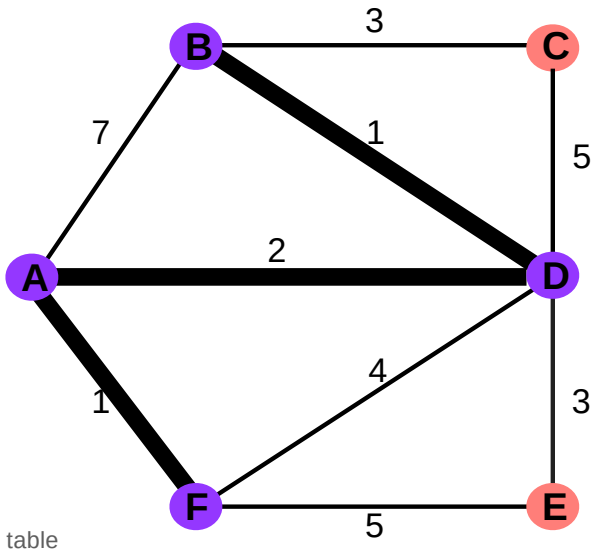
what's in an advertisement

its **link costs** to each of its **neighbors**

who gets a node's advertisement

effectively, **every other node** (via flooding)

link-state routing: disseminate full topology information so that nodes can run a shortest-path algorithm



A's routing table

dst	route	cost
B	**A-B**	**3**
C	A-D	6
D	**A-D**	**2**
E	A-D	5
F	**A-F**	**1**

link state

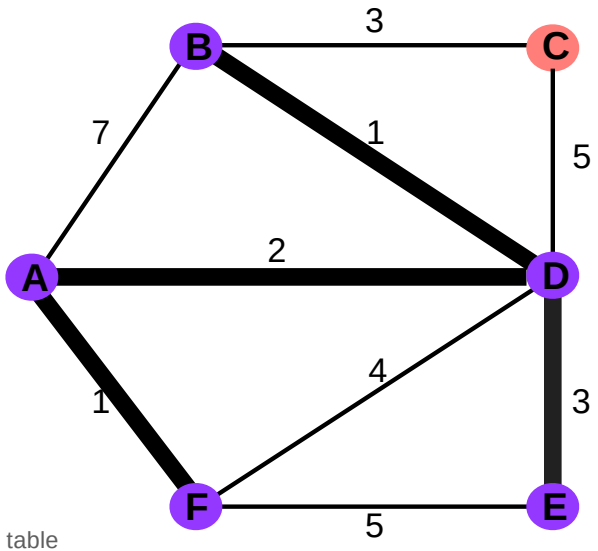
what's in an advertisement

its **link costs** to each of its **neighbors**

who gets a node's advertisement

effectively, **every other node** (via flooding)

link-state routing: disseminate full topology information so that nodes can run a shortest-path algorithm



A's routing table

dst	route	cost
B	**A-B**	**3**
C	A-D	6
D	**A-D**	**2**
E	**A-D**	**5**
F	**A-F**	**1**

link state

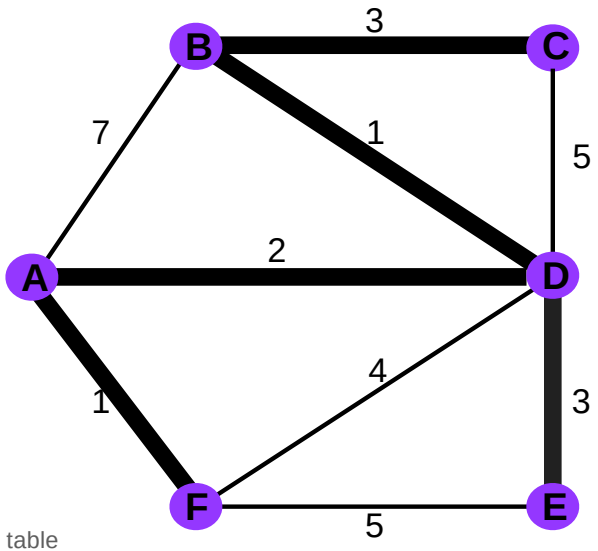
what's in an advertisement

its **link costs** to each of its **neighbors**

who gets a node's advertisement

effectively, **every other node** (via flooding)

link-state routing: disseminate full topology information so that nodes can run a shortest-path algorithm



A's routing table

dst	route	cost
B	A-B	3
C	A-D	6
D	A-D	2
E	A-D	5
F	A-F	1

link state

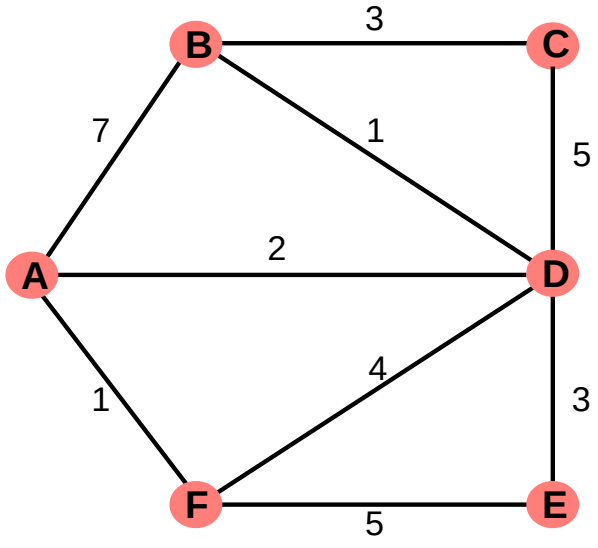
what's in an advertisement

its **link costs** to each of its **neighbors**

who gets a node's advertisement

effectively, **every other node** (via flooding)

link-state routing: disseminate full topology information so that nodes can run a shortest-path algorithm



link state

what's in an advertisement

its **link costs** to each of its **neighbors**

who gets a node's advertisement

effectively, **every other node** (via flooding)

what happens when things fail?

flooding makes link-state routing very resilient to failure

what limits scale?

the **overhead** of flooding

Distance-vector Routing

In link-state, nodes calculate full shortest paths. But actually they only need the route (first-hop) to a destination

Advertisement format: Each node's advertisement is a list of all the nodes it knows about, and its current costs to those nodes

- Initially, this advertisement is just [(self, 0)].

Nodes who receive an advertisement: Node X's advertisement will be received only by its neighbors

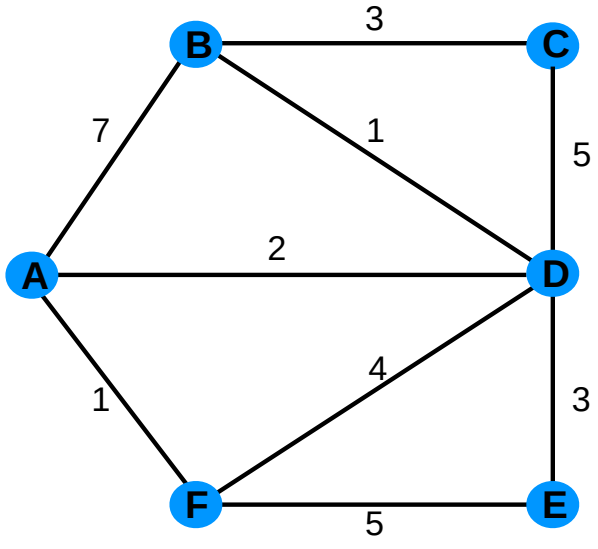
Integrate step: When node X receives an advertisement from its neighbor Y, this advertisement will be a list of [(dst, cost)] pairs. Each cost represents Y's cost to dst

Distance-vector Routing

For each (dst, cost) in the advertisement, X needs to check for two things:

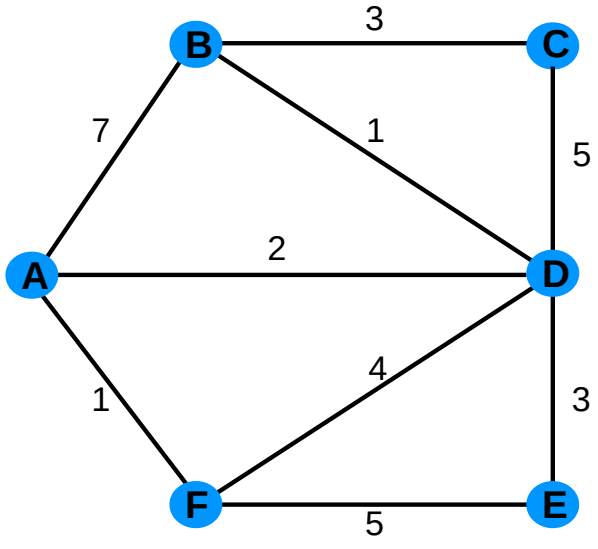
- If X is already using Y to get to dst, update the cost information (remember, costs can change!)
- If X is not using Y to get to dst, see if Y could provide a better path; if so, update the routing and cost information

distance-vector routing: disseminate information about the current *costs* to each node, rather than the actual topology



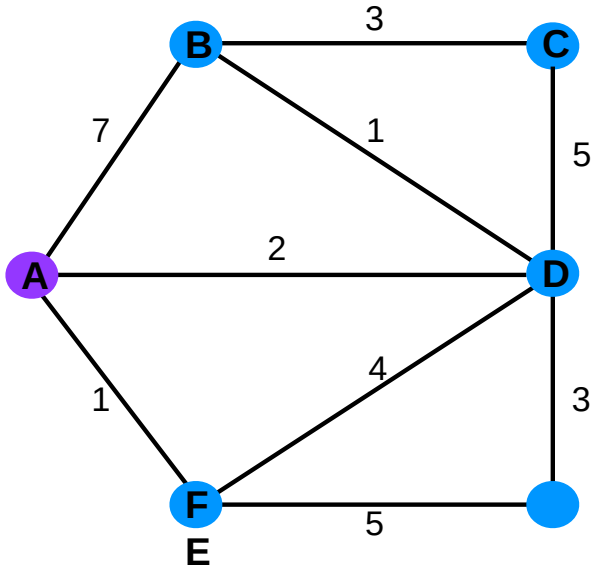
link state	distance vector
what's in an advertisement	
its **link costs** to each of its **neighbors**	
who gets a node's advertisement	
effectively, **every other node** (via flooding)	
what happens when things fail?	
flooding makes link-state routing very resilient to failure	
what limits scale?	
the **overhead** of flooding	

distance-vector routing: disseminate information about the current *costs* to each node, rather than the actual topology



link state	distance vector
what's in an advertisement	
its **link costs** to each of its **neighbors**	its **current costs** to **every node it's aware of**
who gets a node's advertisement	
effectively, **every other node** (via flooding)	
what happens when things fail?	
flooding makes link-state routing very resilient to failure	
what limits scale?	
the **overhead** of flooding	

distance-vector routing: disseminate information about the current *costs* to each node, rather than the actual topology



A's *first* advertisement: [(B,7), (D,2),(F,1)]

A could also include (A,0) here

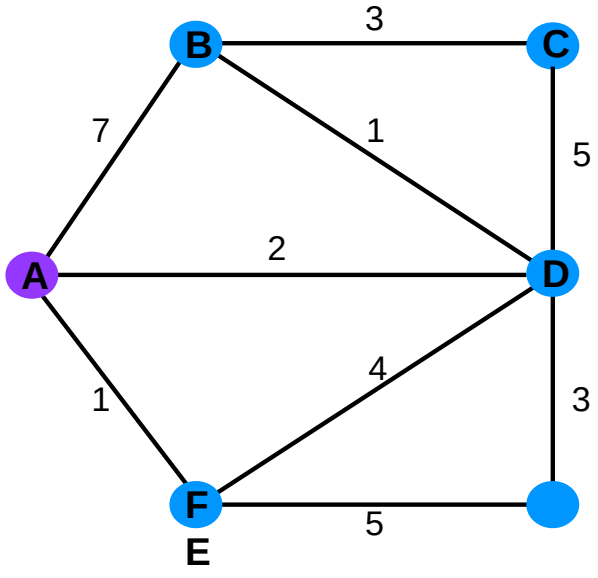
A's routing table

dst	route	cost
B	A-B	7
D	A-D	2
F	A-F	1

A's advertisement reflects its routing table, and right now, A only knows about its neighbors

link state	distance vector
what's in an advertisement	
its **link costs** to each of its **neighbors**	its **current costs** to **every node it's aware of**
who gets a node's advertisement	
effectively, **every other node** (via flooding)	
what happens when things fail?	
flooding makes link-state routing very resilient to failure	
what limits scale?	
the **overhead** of flooding	

distance-vector routing: disseminate information about the current *costs* to each node, rather than the actual topology



A's *first* advertisement: [(B,7), (D,2),(F,1)]

A could also include (A,0) here

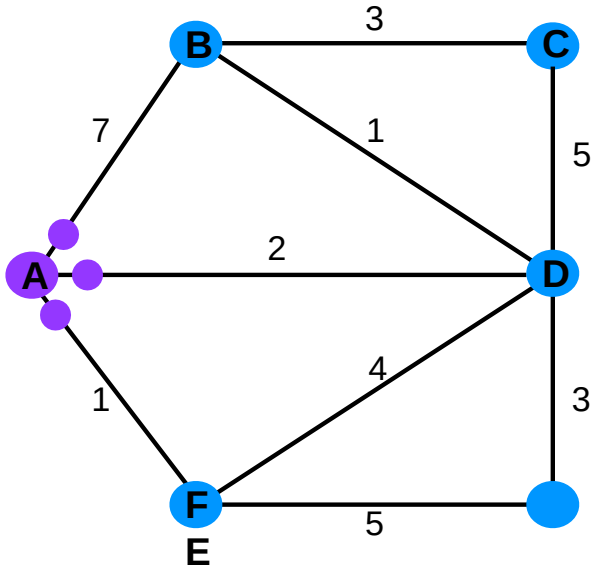
A's routing table

dst	route	cost
B	A-B	7
D	A-D	2
F	A-F	1

A's advertisement reflects its routing table, and right now, A only knows about its neighbors

link state	distance vector
what's in an advertisement	
its **link costs** to each of its **neighbors**	its **current costs** to **every node it's aware of**
who gets a node's advertisement	
effectively, **every other node** (via flooding)	only its **neighbors**
what happens when things fail?	
flooding makes link-state routing very resilient to failure	
what limits scale?	
the **overhead** of flooding	

distance-vector routing: disseminate information about the current *costs* to each node, rather than the actual topology



A's *first* advertisement: [(B,7), (D,2),(F,1)]

A could also include (A,0) here

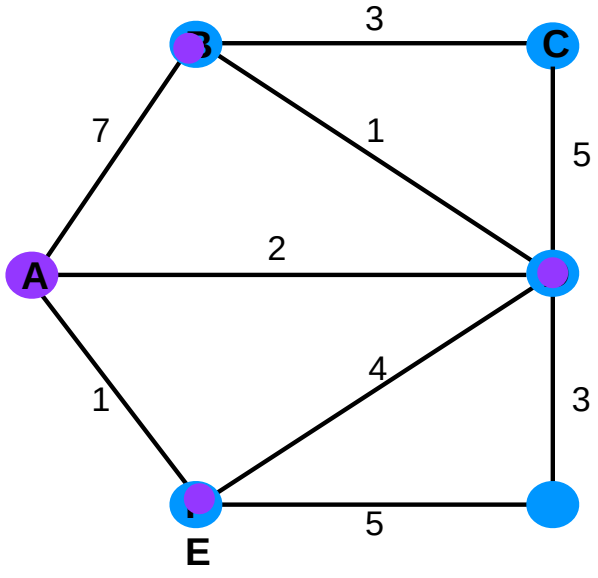
A's routing table

dst	route	cost
B	A-B	7
D	A-D	2
F	A-F	1

A's advertisement reflects its routing table, and right now, A only knows about its neighbors

	link state	distance vector
what's in an advertisement	its **link costs** to each of its **neighbors**	its **current costs** to **every node it's aware of**
who gets a node's advertisement	effectively, **every other node** (via flooding)	only its **neighbors**
what happens when things fail?	flooding makes link-state routing very resilient to failure	
what limits scale?	the **overhead** of flooding	

distance-vector routing: disseminate information about the current *costs* to each node, rather than the actual topology



A's *first* advertisement: [(B,7),(D,2),(F,1)]

A could also include (A,0) here

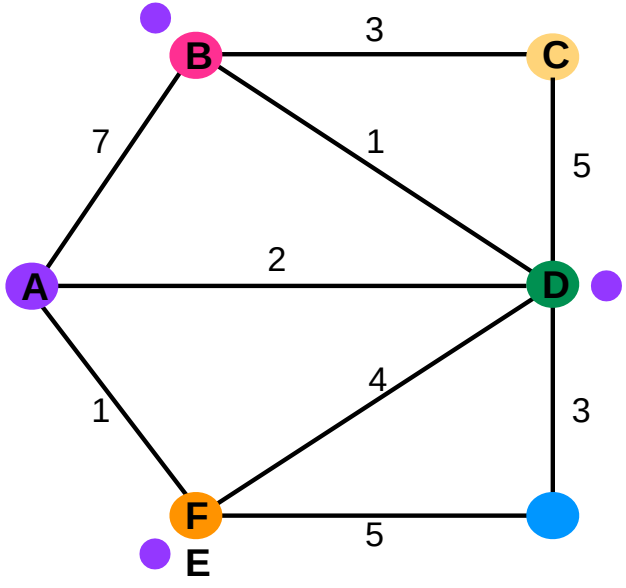
A's routing table

dst	route	cost
B	A-B	7
D	A-D	2
F	A-F	1

A's advertisement reflects its routing table, and right now, A only knows about its neighbors

	link state	distance vector
what's in an advertisement	its **link costs** to each of its **neighbors**	its **current costs** to **every node it's aware of**
who gets a node's advertisement	effectively, **every other node** (via flooding)	only its **neighbors**
what happens when things fail?	flooding makes link-state routing very resilient to failure	
what limits scale?	the **overhead** of flooding	

distance-vector routing: disseminate information about the current *costs* to each node, rather than the actual topology



A's *first* advertisement: [(B,7),(D,2),(F,1)]

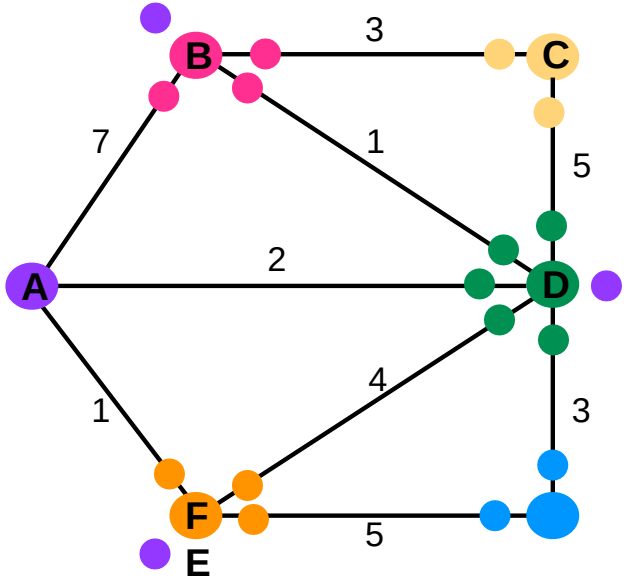
A's routing table

dst	route	cost
B	A-B	7
D	A-D	2
F	A-F	1

A's neighbors **do not** forward A's advertisements; they *do* send advertisements of their own to A

link state	distance vector
what's in an advertisement	
its **link costs** to each of its **neighbors**	its **current costs** to **every node it's aware of**
who gets a node's advertisement	
effectively, **every other node** (via flooding)	only its **neighbors**
what happens when things fail?	
flooding makes link-state routing very resilient to failure	
what limits scale?	
the **overhead** of flooding	

distance-vector routing: disseminate information about the current *costs* to each node, rather than the actual topology



A's *first* advertisement: [(B,7),(D,2),(F,1)]

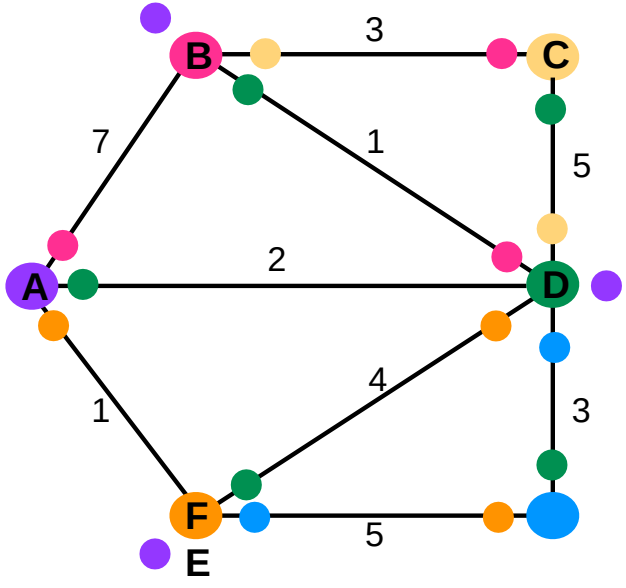
A's routing table

dst	route	cost
B	A-B	7
D	A-D	2
F	A-F	1

A's neighbors **do not** forward A's advertisements; they *do* send advertisements of their own to A

link state	distance vector
what's in an advertisement	
its **link costs** to each of its **neighbors**	its **current costs** to **every node it's aware of**
who gets a node's advertisement	
effectively, **every other node** (via flooding)	only its **neighbors**
what happens when things fail?	
flooding makes link-state routing very resilient to failure	
what limits scale?	
the **overhead** of flooding	

distance-vector routing: disseminate information about the current *costs* to each node, rather than the actual topology



A's *first* advertisement: [(B,7),(D,2),(F,1)]

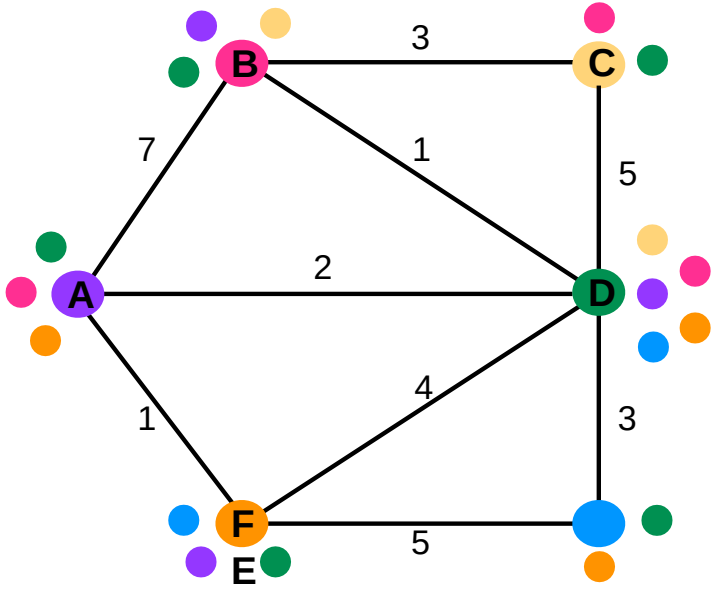
A's routing table

dst	route	cost
B	A-B	7
D	A-D	2
F	A-F	1

A's neighbors **do not** forward A's advertisements; they *do* send advertisements of their own to A

link state	distance vector
what's in an advertisement	
its **link costs** to each of its **neighbors**	its **current costs** to **every node it's aware of**
who gets a node's advertisement	
effectively, **every other node** (via flooding)	only its **neighbors**
what happens when things fail?	
flooding makes link-state routing very resilient to failure	
what limits scale?	
the **overhead** of flooding	

distance-vector routing: disseminate information about the current *costs* to each node, rather than the actual topology



A's *first* advertisement: [(B,7),(D,2),(F,1)]

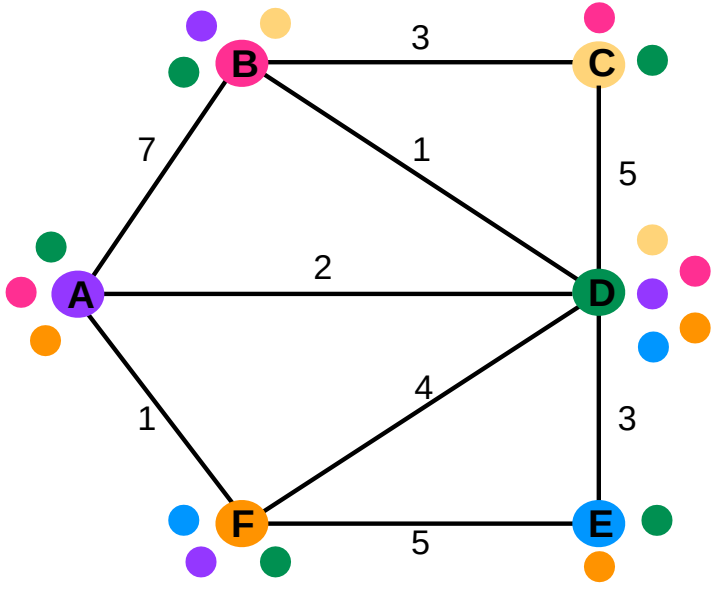
A's routing table

dst	route	cost
B	A-B	7
D	A-D	2
F	A-F	1

A's neighbors **do not** forward A's advertisements; they *do* send advertisements of their own to A

link state	distance vector
what's in an advertisement	
its **link costs** to each of its **neighbors**	its **current costs** to **every node it's aware of**
who gets a node's advertisement	
effectively, **every other node** (via flooding)	only its **neighbors**
what happens when things fail?	
flooding makes link-state routing very resilient to failure	
what limits scale?	
the **overhead** of flooding	

distance-vector routing: disseminate information about the current *costs* to each node, rather than the actual topology



A's routing table

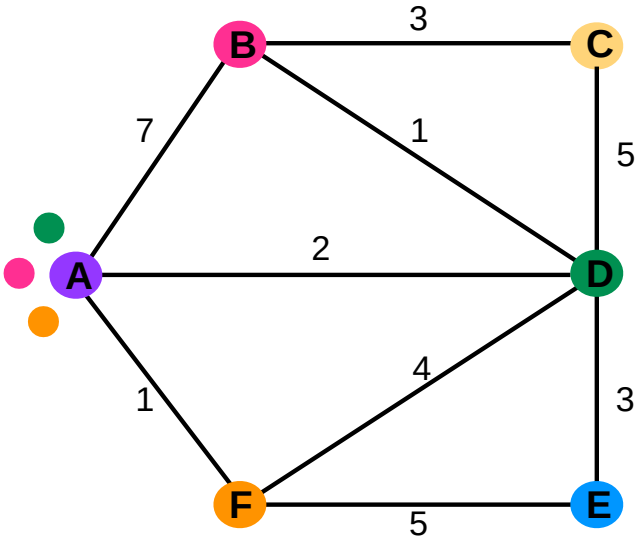
dst	route	cost
B	A-B	7
D	A-D	2
F	A-F	1

B's *first* adv: [(A,7), (C,3), (D,1)]
D's *first* adv: [(B,1), (C,5), (E,3), (F,4)(A,2)]
F's *first* adv: [(A,1), (D,4), (E,5)]

A receives advertisements from B, D, and F

link state	distance vector
what's in an advertisement	
its **link costs** to each of its **neighbors**	its **current costs** to **every node it's aware of**
who gets a node's advertisement	
effectively, **every other node** (via flooding)	only its **neighbors**
what happens when things fail?	
flooding makes link-state routing very resilient to failure	
what limits scale?	
the **overhead** of flooding	

distance-vector routing: disseminate information about the current *costs* to each node, rather than the actual topology



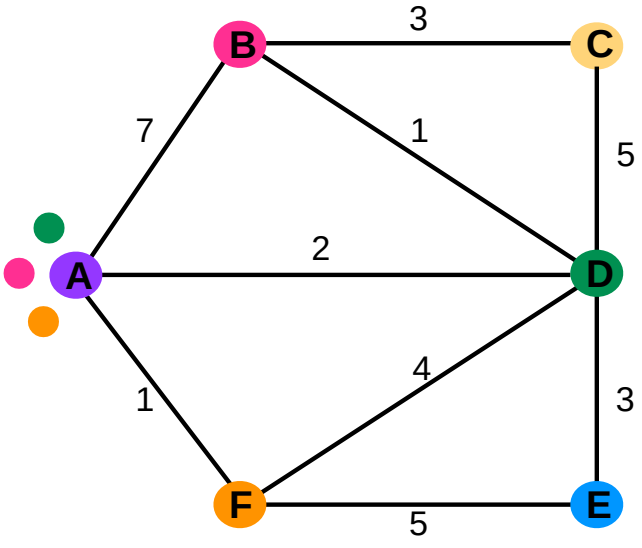
A's routing table

dst	route	cost
B	A-B	7
D	A-D	2
F	A-F	1

B's *first* adv: [(A,7), (C,3), (D,1)]

link state	distance vector
what's in an advertisement	
its **link costs** to each of its **neighbors**	its **current costs** to **every node it's aware of**
who gets a node's advertisement	
effectively, **every other node** (via flooding)	only its **neighbors**
what happens when things fail?	
flooding makes link-state routing very resilient to failure	
what limits scale?	
the **overhead** of flooding	

distance-vector routing: disseminate information about the current *costs* to each node, rather than the actual topology



A's routing table

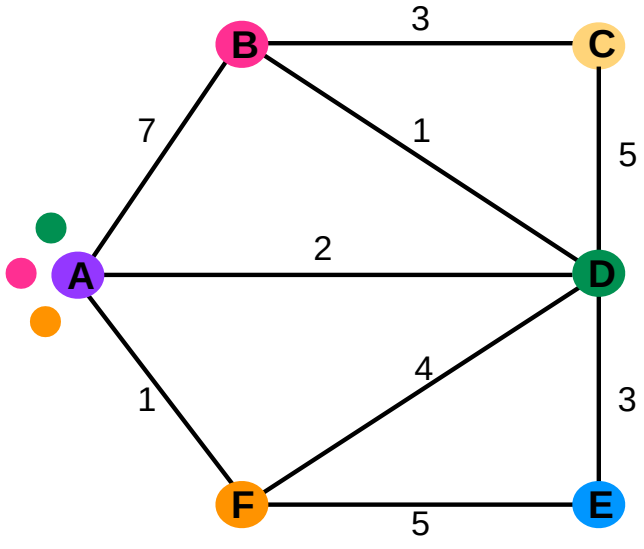
dst	route	cost
B	A-B	7
C	A-B	10
D	A-D	2
F	A-F	1

B's *first* adv: [(A,7), (C,3), (D,1)]

A's cost to B + B's cost to C

link state	distance vector
what's in an advertisement	
its **link costs** to each of its **neighbors**	its **current costs** to **every node it's aware of**
who gets a node's advertisement	
effectively, **every other node** (via flooding)	only its **neighbors**
what happens when things fail?	
flooding makes link-state routing very resilient to failure	
what limits scale?	
the **overhead** of flooding	

distance-vector routing: disseminate information about the current *costs* to each node, rather than the actual topology



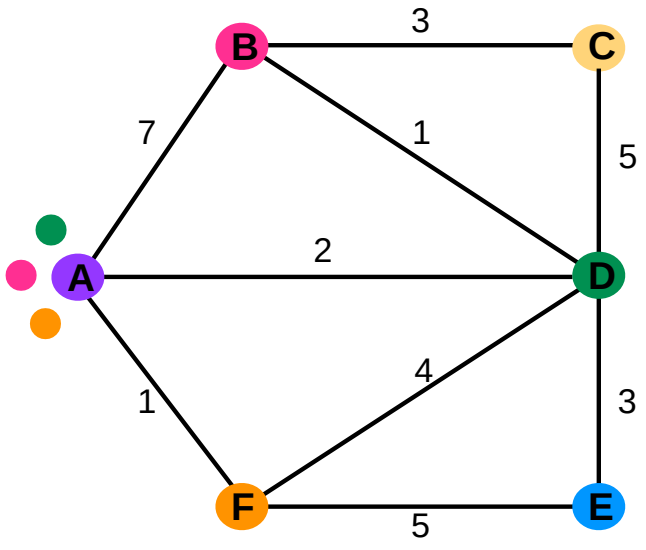
A's routing table

dst	route	cost
B	A-B	7
C	A-B	10
D	A-D	2
F	A-F	1

B's *first* adv: [(A,7), (C,3), (D,1)]
D's *first* adv: [(B,1), (C,5), (E,3), (F,4)(A,2)]
F's *first* adv: [(A,1), (D,4), (E,5)]

link state	distance vector
what's in an advertisement	
its **link costs** to each of its **neighbors**	its **current costs** to **every node it's aware of**
who gets a node's advertisement	
effectively, **every other node** (via flooding)	only its **neighbors**
what happens when things fail?	
flooding makes link-state routing very resilient to failure	
what limits scale?	
the **overhead** of flooding	

distance-vector routing: disseminate information about the current *costs* to each node, rather than the actual topology



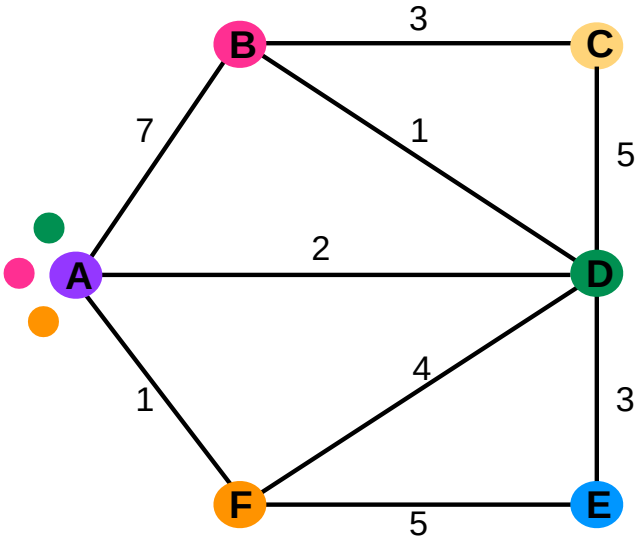
A's routing table

dst	route	cost
B	A-B	7
C	A-B	10
D	A-D	2
F	A-F	1

D's *first* adv: [(B,1), (C,5), (E,3), (F,4)(A,2)]

link state	distance vector
what's in an advertisement	
its **link costs** to each of its **neighbors**	its **current costs** to **every node it's aware of**
who gets a node's advertisement	
effectively, **every other node** (via flooding)	only its **neighbors**
what happens when things fail?	
flooding makes link-state routing very resilient to failure	
what limits scale?	
the **overhead** of flooding	

distance-vector routing: disseminate information about the current *costs* to each node, rather than the actual topology



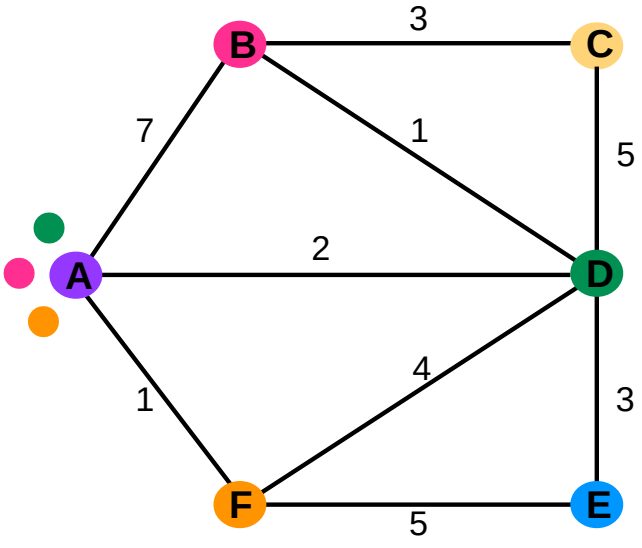
A's routing table

dst	route	cost
B	A-D	3
C	A-B	10
D	A-D	2
F	A-F	1

D's *first* adv: [(B,1), (C,5), (E,3), (F,4)(A,2)]

link state	distance vector
what's in an advertisement	
its **link costs** to each of its **neighbors**	its **current costs** to **every node it's aware of**
who gets a node's advertisement	
effectively, **every other node** (via flooding)	only its **neighbors**
what happens when things fail?	
flooding makes link-state routing very resilient to failure	
what limits scale?	
the **overhead** of flooding	

distance-vector routing: disseminate information about the current *costs* to each node, rather than the actual topology



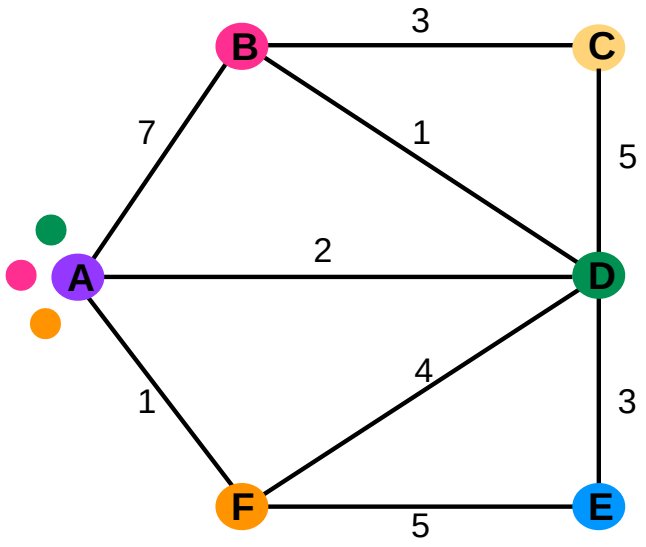
A's routing table

dst	route	cost
B	A-D	3
C	A-D	7
D	A-D	2
F	A-F	1

D's *first* adv: [(B,1), (C,5), (E,3), (F,4)(A,2)]

link state	distance vector
what's in an advertisement	
its **link costs** to each of its **neighbors**	its **current costs** to **every node it's aware of**
who gets a node's advertisement	
effectively, **every other node** (via flooding)	only its **neighbors**
what happens when things fail?	
flooding makes link-state routing very resilient to failure	
what limits scale?	
the **overhead** of flooding	

distance-vector routing: disseminate information about the current *costs* to each node, rather than the actual topology



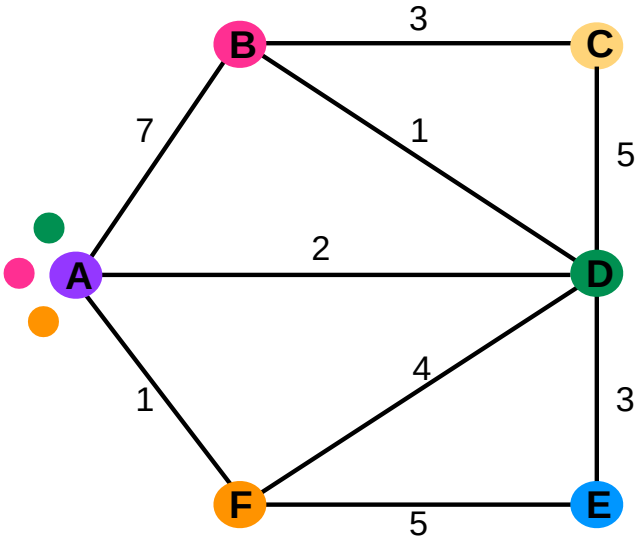
A's routing table

dst	route	cost
B	A-D	3
C	A-D	7
D	A-D	2
E	A-D	5
F	A-F	1

D's *first* adv: [(B,1), (C,5), (E,3), (F,4)(A,2)]

link state	distance vector
what's in an advertisement	
its **link costs** to each of its **neighbors**	its **current costs** to **every node it's aware of**
who gets a node's advertisement	
effectively, **every other node** (via flooding)	only its **neighbors**
what happens when things fail?	
flooding makes link-state routing very resilient to failure	
what limits scale?	
the **overhead** of flooding	

distance-vector routing: disseminate information about the current *costs* to each node, rather than the actual topology



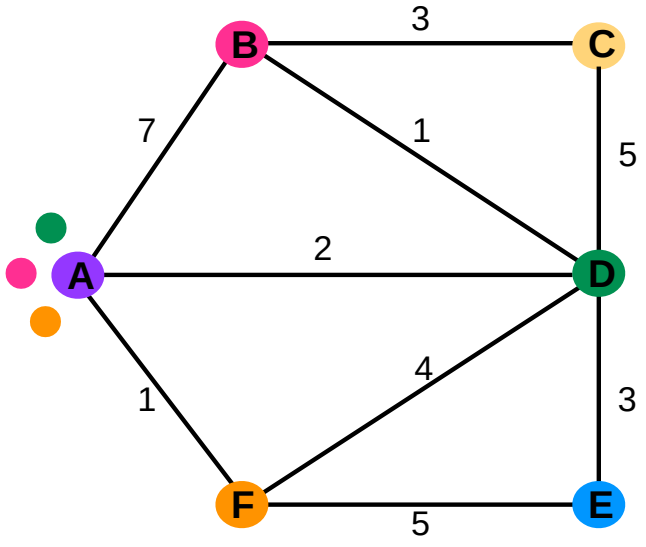
A's routing table

dst	route	cost
B	A-D	3
C	A-D	7
D	A-D	2
E	A-D	5
F	A-F	1

B's *first* adv: [(A,7), (C,3), (D,1)]
D's *first* adv: [(B,1), (C,5), (E,3), (F,4)(A,2)]
F's *first* adv: [(A,1), (D,4), (E,5)]

link state	distance vector
what's in an advertisement	
its **link costs** to each of its **neighbors**	its **current costs** to **every node it's aware of**
who gets a node's advertisement	
effectively, **every other node** (via flooding)	only its **neighbors**
what happens when things fail?	
flooding makes link-state routing very resilient to failure	
what limits scale?	
the **overhead** of flooding	

distance-vector routing: disseminate information about the current *costs* to each node, rather than the actual topology



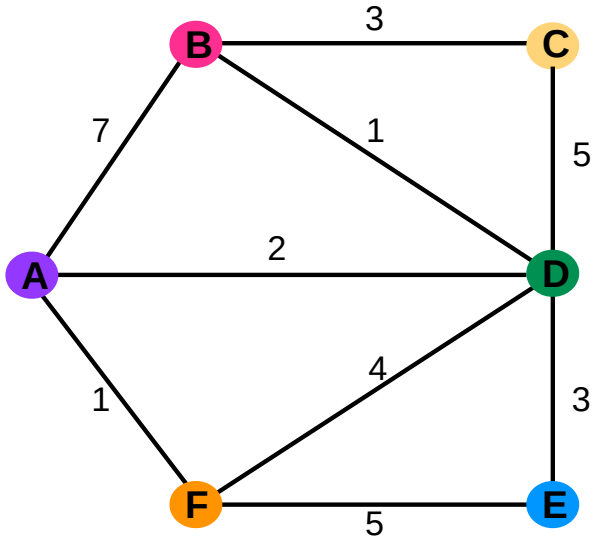
A's routing table

dst	route	cost
B	A-D	3
C	A-D	7
D	A-D	2
E	A-D	5
F	A-F	1

F's *first* adv: [(A,1), (D,4), (E,5)]

link state	distance vector
what's in an advertisement	
its **link costs** to each of its **neighbors**	its **current costs** to **every node it's aware of**
who gets a node's advertisement	
effectively, **every other node** (via flooding)	only its **neighbors**
what happens when things fail?	
flooding makes link-state routing very resilient to failure	
what limits scale?	
the **overhead** of flooding	

distance-vector routing: disseminate information about the current *costs* to each node, rather than the actual topology



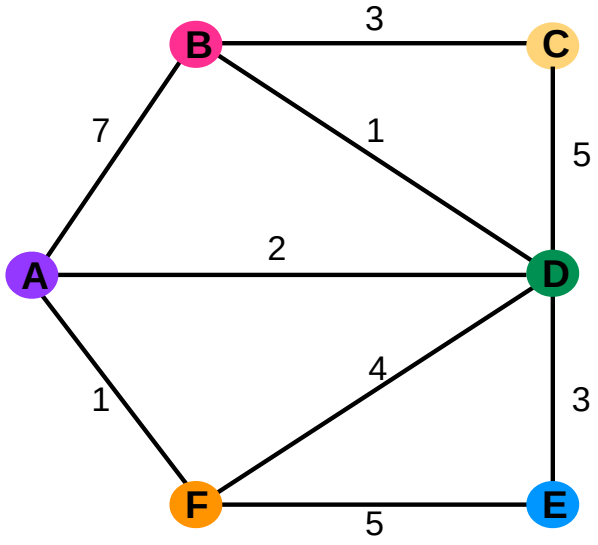
A's routing table

dst	route	cost
B	A-D	3
C	A-D	7
D	A-D	2
E	A-D	5
F	A-F	1

this is A's routing table after one round of advertisements; note that it does not have knowledge of the min-cost route to C yet

link state	distance vector
what's in an advertisement	
its **link costs** to each of its **neighbors**	its **current costs** to **every node it's aware of**
who gets a node's advertisement	
effectively, **every other node** (via flooding)	only its **neighbors**
what happens when things fail?	
flooding makes link-state routing very resilient to failure	
what limits scale?	
the **overhead** of flooding	

distance-vector routing: disseminate information about the current *costs* to each node, rather than the actual topology



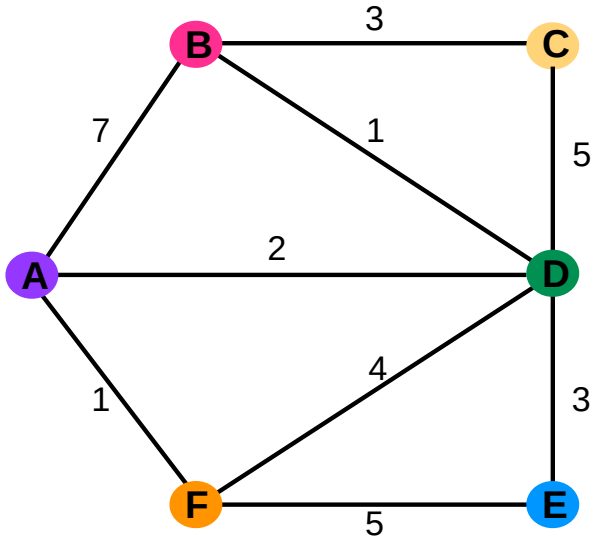
A's routing table

dst	route	cost
B	A-D	3
C	A-D	7
D	A-D	2
E	A-D	5
F	A-F	1

A's *second* adv:
[(B,3), (C,7), (D,2), (E,5), (F,1)]

link state	distance vector
what's in an advertisement	
its **link costs** to each of its **neighbors**	its **current costs** to **every node it's aware of**
who gets a node's advertisement	
effectively, **every other node** (via flooding)	only its **neighbors**
what happens when things fail?	
flooding makes link-state routing very resilient to failure	
what limits scale?	
the **overhead** of flooding	

distance-vector routing: disseminate information about the current *costs* to each node, rather than the actual topology



A's routing table

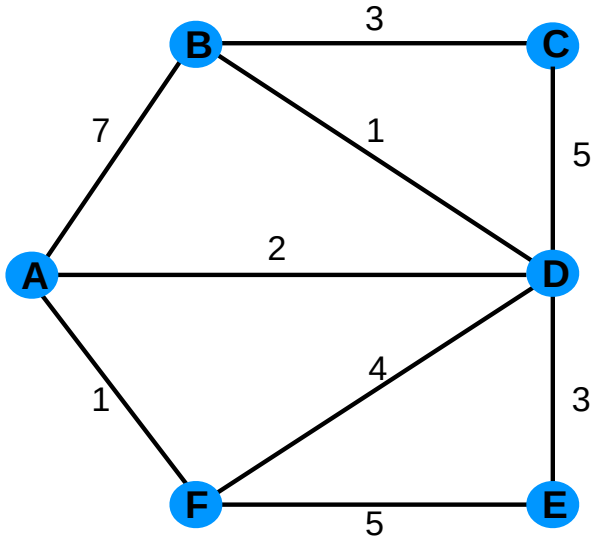
dst	route	cost
B	A-D	3
C	A-D	7
D	A-D	2
E	A-D	5
F	A-F	1

A's *second* adv:
[(B,3), (C,7), (D,2), (E,5), (F,1)]

A will learn about the correct min-cost path to C in the next round of advertisements; try that out for yourself!

link state	distance vector
what's in an advertisement	
its **link costs** to each of its **neighbors**	its **current costs** to **every node it's aware of**
who gets a node's advertisement	
effectively, **every other node** (via flooding)	only its **neighbors**
what happens when things fail?	
flooding makes link-state routing very resilient to failure	
what limits scale?	
the **overhead** of flooding	

distance-vector routing: disseminate information about the current *costs* to each node, rather than the actual topology



link state	distance vector
what's in an advertisement	
its **link costs** to each of its **neighbors**	its **current costs** to **every node it's aware of**
who gets a node's advertisement	
effectively, **every other node** (via flooding)	only its **neighbors**
what happens when things fail?	
flooding makes link-state routing very resilient to failure	failures can be complicated because of timing
what limits scale?	
the **overhead** of flooding	failure handling

Problem of INFINITY

When a node A has no route to destination B, it will advertise a cost of INFINITY to B

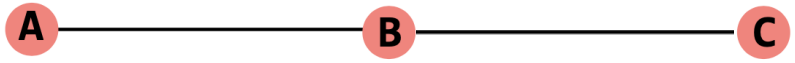
- A cost of INFINITY B is interpreted as there being no route to B
- So INFINITY must be larger than the longest path in the network

But because the order in which advertisements are sent matters, sometimes nodes can incorrectly think there's a route when there isn't one

This can last for up to INFINITY steps (usually 16 or 32)

INFINITY

A sends advertisements at t=0, 10, 20,..; B sends advertisements at t=5, 15, 25,..

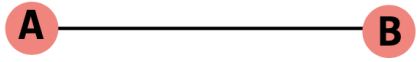


A:	Self, 0	A:	B->A, 1
B:	A->B, 1	B:	Self, 0
C:	A->B, 2	C:	B->C, 1

t=9: B<->C fails

INFINITY

A sends advertisements at t=0, 10, 20,..; B sends advertisements at t=5, 15, 25,..



A: Self, 0	A: B->A, 1
B: A->B, 1	B: Self, 0
C: A->B, 2	C: None, inf
A: Self, 0	A: B->A, 1
B: A->B, 1	B: Self, 0
C: A->B, 2	C: B->A, 3 (2+1)
A: Self, 0	A: B->A, 1
B: A->B, 1	B: Self, 0
C: A->B, 4	C: B->A, 3
A: Self, 0	A: B->A, 1
B: A->B, 1	B: Self, 0
C: A->B, 4	C: B->A, 5

C

t=9: B<->C fails

t=10: B receives the following advertisement from A:
[(A,0),(B,1),(C,2)]

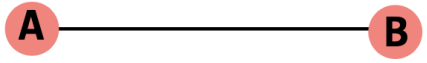
t=15: A receives the following advertisement from B:
[(A,1),(B,0),(C,3)]

t=20: B receives the following advertisement from A:
[(A,0),(B,1),(C,4)]

continues until both costs to C are INFINITY

Split Horizon

A sends advertisements at t=0, 10, 20,..; B sends advertisements at t=5, 15, 25,..




```
A: Self, 0    A: B->A, 1
B: A->B, 1    B: Self, 0
C: A->B, 2    C: None, inf

A: Self, 0    A: B->A, 1
B: A->B, 1    B: Self, 0
C: A->B, 2    C: None, inf

A: Self, 0    A: B->A, 1
B: A->B, 1    B: Self, 0
C: None, inf  C: None, inf
```

C

`t=9: B<->C` fails

`t=10:` `B` receives the following advertisement from `A`: `[(A,0)]`

`t=15:` `A` receives the following advertisement from `B`: `[(B,0),(C,inf)]`

split horizon takes care of this particular case

Link-State Summary

Pros:

- Fast convergence

Cons:

- flooding is costly: 2 x #Node x #Line advertisements

Only good for small networks

Distance Vector

Pros:

- Low overhead: 2x #Line advertisements

Cons:

- Convergence time is proportional to longest path
- The infinity problem

Only good for small networks

How to Scale the Routing?

3 Ways to Scale

Path-vector Routing

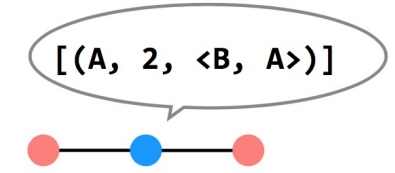
- Advertisements include the path, to better detect routing loops

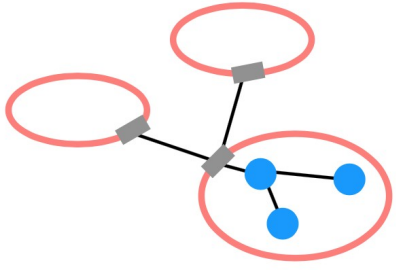
Hierarchy of Routing

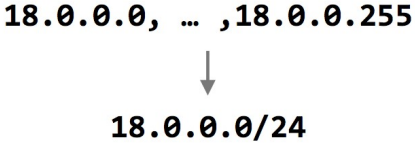
- Route between **regions**, and then within a region

Topological Addressing

- Assign addresses in contiguous blocks to make advertisements smaller







Path Vector Exchange

Each participant maintains a path vector

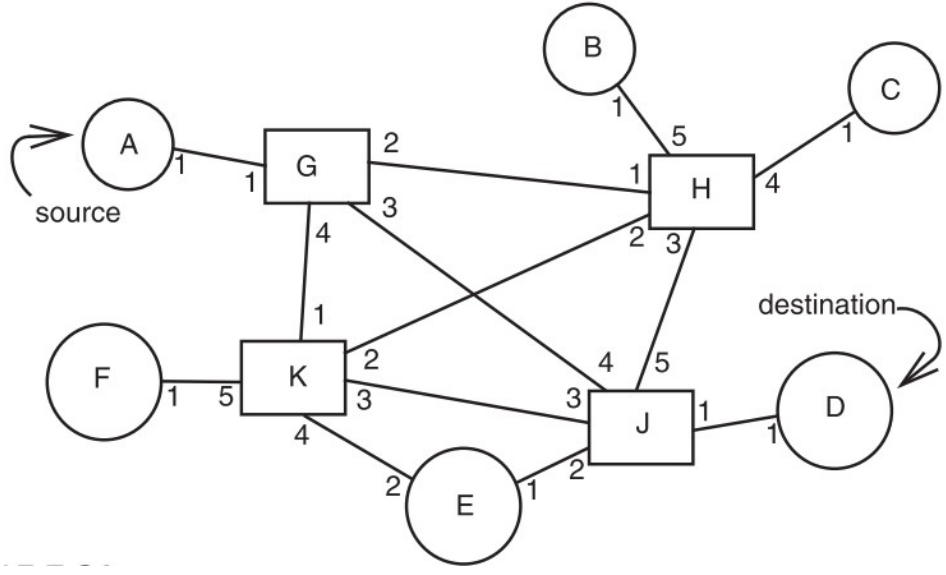
- A complete path to some destination
- E.g., zero-length path to itself
- Gradually learns about other paths
- Construct a new forwarding table from its new path vector

Algorithm

- Advertising
- Path selection

to	path
G	< >

Path Vector Exchange



A

destination	link
A	end-layer
all other	1

G

destination	link
A	1
B	2
C	2
D	3
E	4
F	4
G	end-layer
H	2
J	3
K	4

Need coordination to ensure no loop

From A, via link 1

to	path
A	< >

From H, via link 2:

to	path
H	< >

From J, via link 3:

to	path
J	< >

From K, via link 4:

to	path
K	< >

path vector

to	path
A	<A>
G	< >
H	<H>
J	<J>
K	<K>

forwarding table

to	link
A	1
G	end-layer
H	2
J	3
K	4

path vector

to	path
A	<A>
B	<H, B>
C	<H, C>
D	<J, D>
E	<J, E>
F	<K, F>
G	< >
H	<H>
J	<J>
K	<K>

forwarding table

to	link
A	1
B	2
C	2
D	3
E	3
F	4
G	end-layer
H	2
J	3
K	4

From A, via link 1

to	path
A	< >
G	<G>

From H, via link 2:

to	path
B	<B>
C	<C>
G	<G>
H	< >
J	<J>
K	<K>

From J, via link 3:

to	path
D	<D>
E	<E>
G	<G>
H	<H>
J	< >
K	<K>

From K, via link 4:

to	path
E	<E>
F	<F>
G	<G>
H	<H>
J	<J>
K	< >

Path Vector Exchange

Better for scaling

- Like Distance-Vector, but include the full path in the routing advertisements
- Overhead increases (advertises are larger), but convergence time decreases (avoid counting to infinity)
- Overhead is still lower than Link-State

Questions on Path Vector

How do we avoid permanent loops?

- When a node updates its paths, it never accepts a path that has itself

What happens when a node hears multiple paths to the same destination?

- It picks the better path

What happens if the graph changes?

- Algorithm deals well with new links
- To deal with links that go down, each router should discard any path that a neighbor stops advertising

Hierarchical Address Assignment & Routing

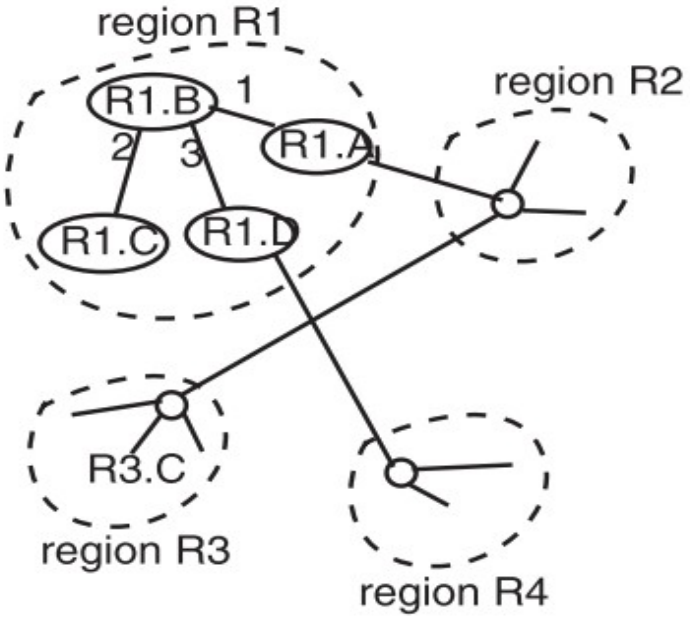
Two problems of the path vector implementation

- Every attachment point must have a unique address
- The path vector grows in size with the number of attachment points

Hierarchy for better scalability

- Two parts of network address: region & station, e.g., "11,75"
- Regions correspond to the set of closely-connected entities
- E.g., region-11 has only 1 entry in other region routers' table

Hierarchical Address Assignment & Routing



forwarding table in R1.B

region forwarding section

to	link
R1	local
R2	1
R3	1
R4	3

local forwarding section

to	link
R1.A	1
R1.B	end-layer
R1.C	2
R1.D	3

Region is also as known as **AS**: Autonomous System

Hierarchical Address Assignment & Routing

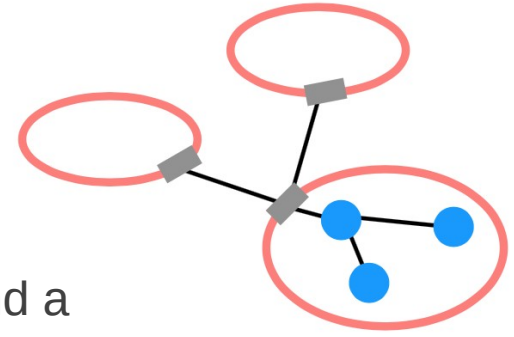
Problems introduced by hierarchy: more complex

- Binding address with location
  - Has to change address after changing location
- Paths may no longer be the shortest possible
  - Algorithm has less detailed information

More about hierarchy

- Can extend to more levels
- Different places can have different levels

Routing Hierarchy



Across Region

- Use one routing protocol to route **across regions**, and a different protocol to route **within regions**
- Implies that there are devices on the edge of each region that can "translate" between or "speak" both protocols

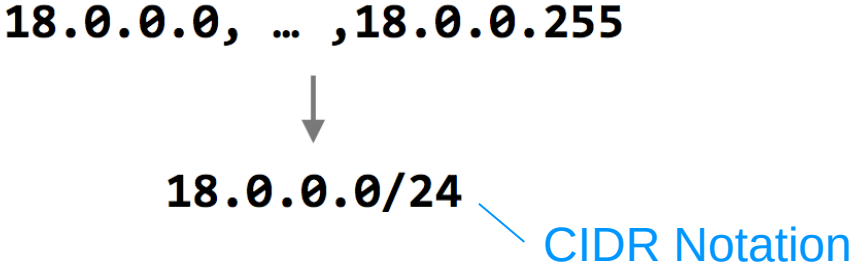
BGP is the path-vector protocol used across regions

- Border Gateway Protocol

Topological Addressing

Further reduce the routing table

- Despite being between regions, BGP still routes to IP addresses (e.g., to *18.0.0.1*, not to *region-3*)
- Addresses are given to regions in **contiguous blocks**, so that they can be specified succinctly via a particular notation ("CIDR" notation)
  - CIDR: Classless Inter Domain Routing
- Keeps advertisements small



Data Plane: Packet Forwarding

Network Layer Interface

```
structure packet
    bit_string source
    bit_string destination
    bit_string end_protocol
    bit_string payload
```

```
1 procedure NETWORK_SEND (segment_buffer, destination,
2                         net_protocol, end_protocol)
3     packet instance outgoing_packet
4     outgoing_packet.payload ← segment_buffer
5     outgoing_packet.end_protocol ← end_protocol
6     outgoing_packet.source ← MY_NETWORK_ADDRESS
7     outgoing_packet.destination ← destination
8     NETWORK_HANDLE (outgoing_packet, net_protocol)

9 procedure NETWORK_HANDLE (net_packet, net_protocol)
10    packet instance net_packet
11    if net_packet.destination != MY_NETWORK_ADDRESS then
12        next_hop ← LOOKUP (net_packet.destination,
forwarding_table)
13        LINK_SEND (net_packet, next_hop, link_protocol,
net_protocol)
14    else
15        GIVE_TO_END_LAYER (net_packet.payload,
```

Forwarding an IP Packet

Lookup packet's destination in forwarding table

- If known, find the corresponding outgoing link
- If unknown, drop packet

Decrement TTL (Time To Live)

- Drop packet if TTL is zero

Update header checksum

Forward packet to outgoing port

Transmit packet onto link

Data-plane Case Study: Intel's DPDK

DPDK: Data Plane Developmen

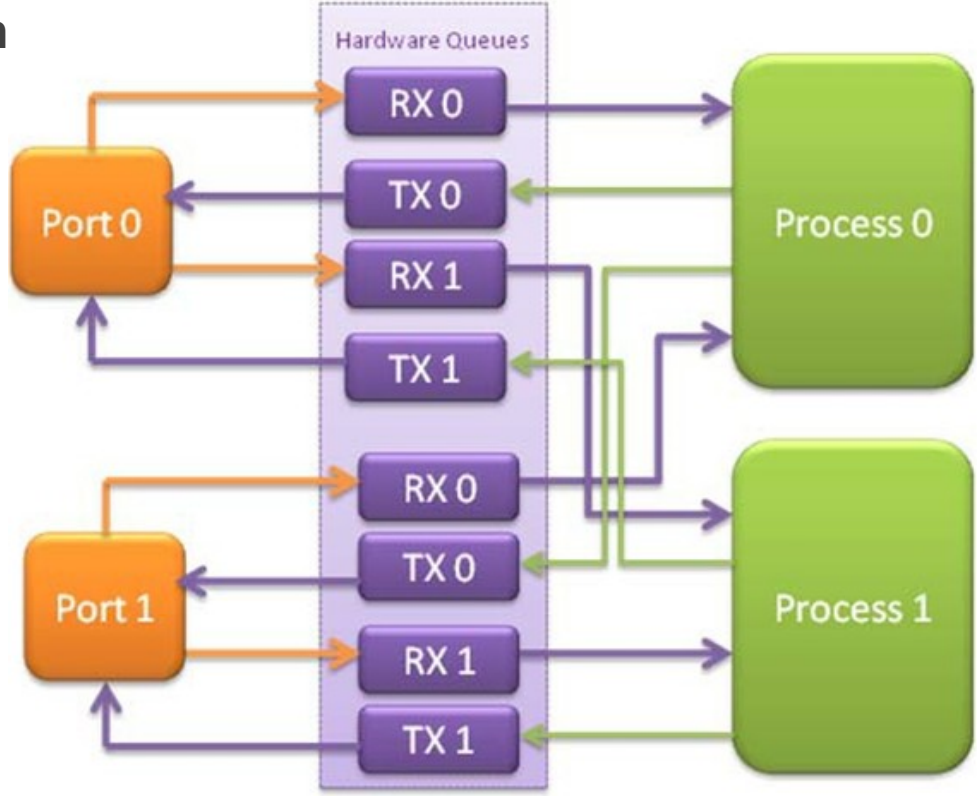
- Bypass kernel

Network card

- Has several ports
- A port has RX/TX

Processor

- Read packets from RX
  - Polling
- Find output port
- Write packets to TX



RX for receiving

TX for sending

NAT

NAT (Network Address Translation)

Private network

- Public routers do not accept routes to network 10 (e.g., 10.8.8.8)

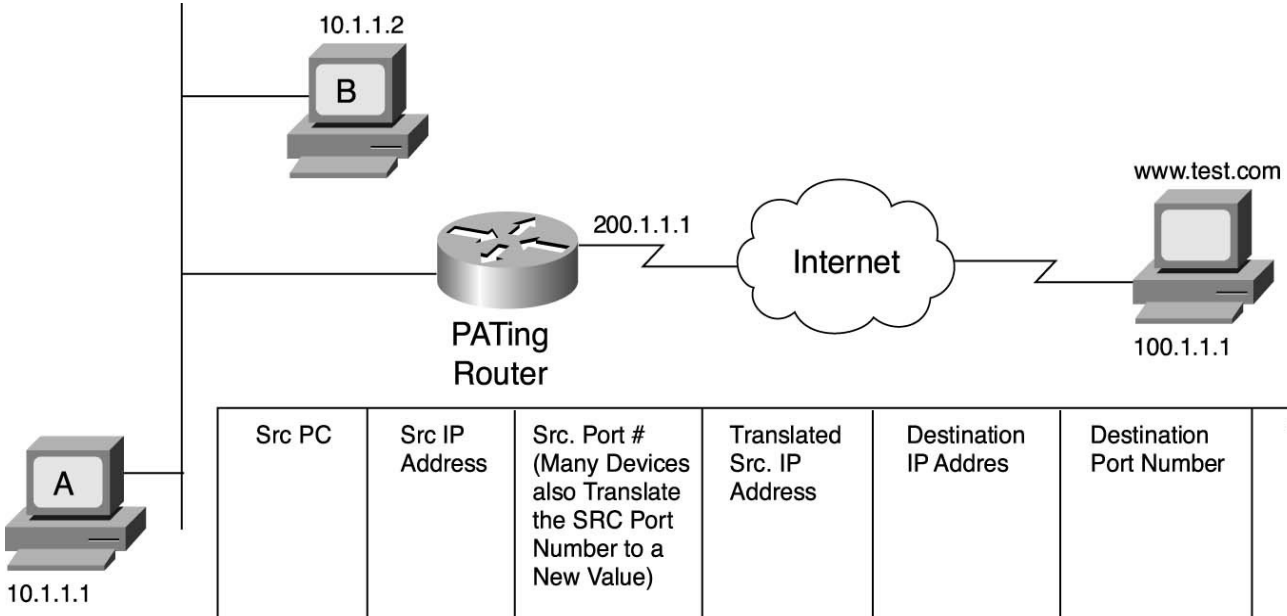
NAT router: bridge the private networks

- Router between private & public network
- Send: modify source address to temp public address
- Receive: modify back by looking mapping table

Limitations

- Some end-to-end protocols place address in payloads
- The translator may become the bottleneck
- What if two private network merge?

NAT



PAT Table

Src PC	Src IP Address	Src. Port # (Many Devices also Translate the SRC Port Number to a New Value)	Translated Src. IP Address	Destination IP Addres	Destination Port Number	Translated Destination IP Address
A	10.1.1.1	3759	200.1.1.1	100.1.1.1	80	100.1.1.1
B	10.1.1.2	2119	200.1.1.1	100.1.1.1	80	100.1.1.1

CASE: Ethernet Mapping

Mapping Internet to Ethernet

Case Study: Mapping Internet to Ethernet

Listen-before-sending rule, collision

Ethernet: CSMA/CD

- Carrier Sense Multiple Access with Collision Detection

Ethernet type

- Experimental Ethernet, 3 mpbs
- Standard Ethernet, 10 mbps
- Fast Ethernet, 100 mbps
- Gigabit Ethernet, 1000 mbps

Overview of Ethernet

A half duplex Ethernet

- The max propagation time is less than the 576 bit times, the shortest allowable packet
- So that two parties can detect a collision together
- If collision: wait random first time, exponential backoff if repeat

A full duplex & point-to-point Ethernet

- No collisions & the max length of the link is determined by the physical medium

leader	destination	source	type	data	checksum
64 bits	48 bits	48 bits	16 bits	368 to 12,000 bits	32 bits

Difference between Hub and Switch

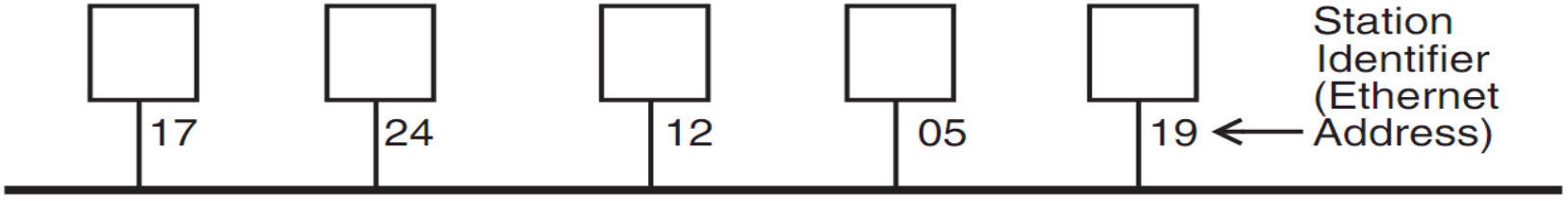
Hub

- A frame is "broadcast" to every one of its ports
- A 10/100Mbps hub must share its bandwidth with each port

Switch

- Keeps a record of the MAC addresses of all the devices
- A 10/100Mbps switch will allocate a full 10/100Mbps to each of its ports

Broadcast Aspects of Ethernet



Broadcast network

- Every frame is delivered to every station
- (Compare with forwarding network)

ETHERNET_SEND

- Pass the call along to the link layer

ETHERNET_HANDLE

- Simple, can even be implemented in hardware

Broadcast Aspects of Ethernet

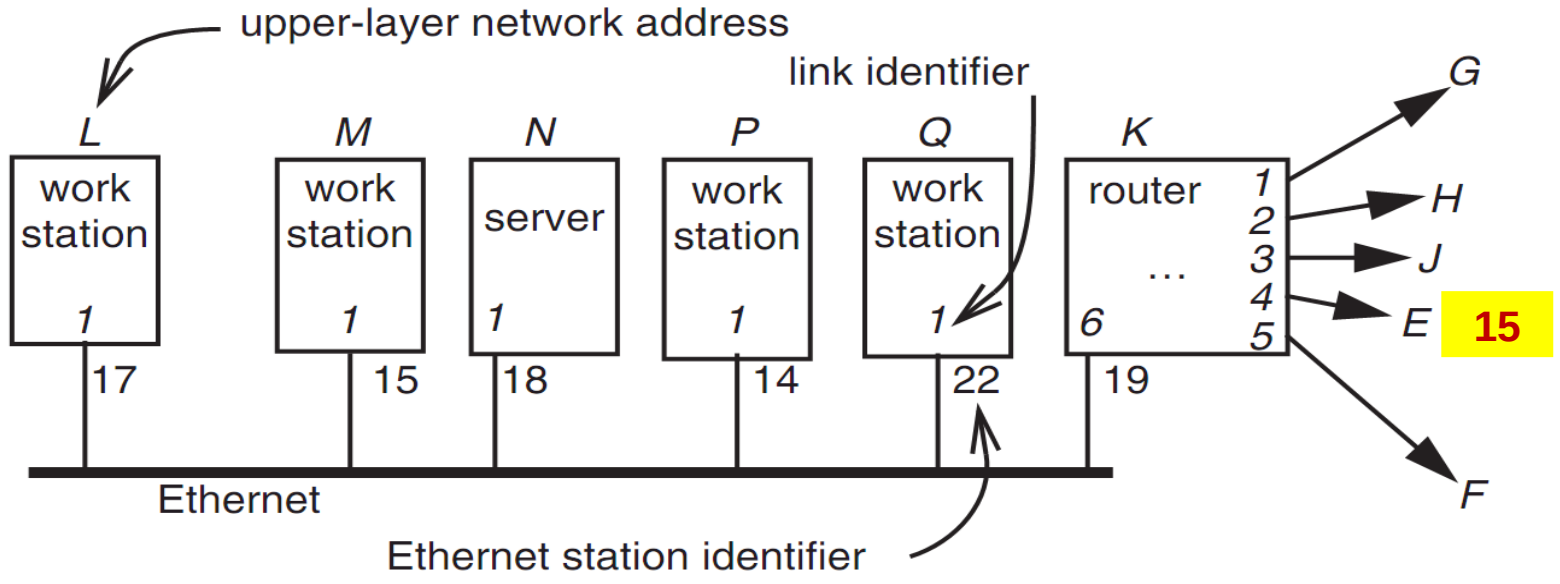
```
procedure ETHERNET_HANDLE (net_packet, length)
  destination ← net_packet.target_id
  if destination = my_station_id

    then
    GIVE_TO_END_LAYER (net_packet.data,
                       net_packet.end_protocol,
                       net_packet.source_id)
  else
    ignore packet
```

no need to do any forwarding

Broadcast Aspects of Ethernet

```
procedure ETHERNET_HANDLE (net_packet, length)
  destination ← net_packet.target_id
  if destination = my_station_id
      or destination = BROADCAST_ID
    then
    GIVE_TO_END_LAYER (net_packet.data,
                       net_packet.end_protocol,
                       net_packet.source_id)
  else
    ignore packet
```

Layer Mapping: Attach Ethernet to Forwarding Network



L sends a RPC to N by sending to station 18 of link 1

L sends a RPC to E by sending to K, E may have 15 as address, as well as M

Layer Mapping

The Internet network layer

- **NETWORK_SEND** (data, length, RPC, INTERNET, N)
- **NETWORK_SEND** (data, length, RPC, ENET, 18)

L must maintain a table

internet address	Ethernet/ station
M	enet/15
N	enet/18
P	enet/14
Q	enet/22
K	enet/19
E	enet/19

ARP (Address Resolution Protocol)

NETWORK_SEND ("where is M?", 11, ARP, ENET, BROADCAST)

NETWORK_SEND ("M is at station 15", 18, ARP, ENET, BROADCAST)

L asks E's Ethernet address, E does not hear the Ethernet broadcast, but the router at station 19 does, and it sends a suitable ARP response instead

Manage forwarding table as a cache

internet address	Ethernet/ station
M	enet/15

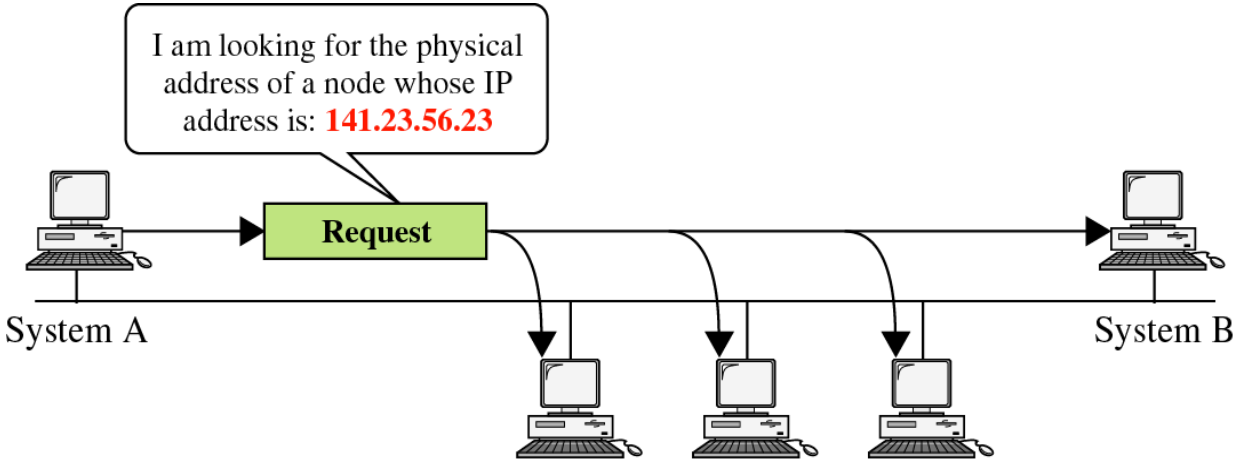
internet address	Ethernet/ station
M	enet/15
E	enet/19

ARP & RARP Protocol

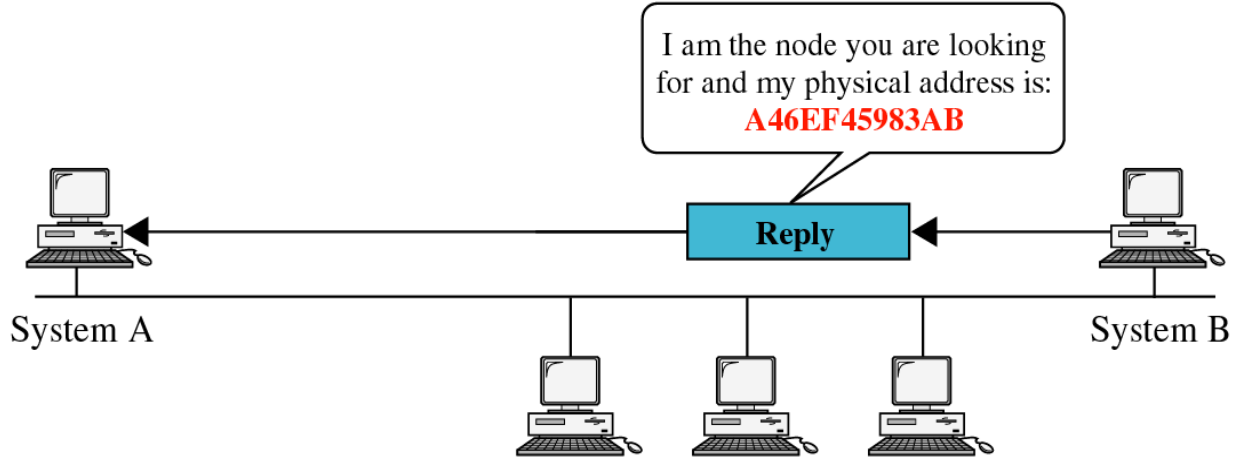
Hardware Type (16 bits)		Protocol Type (16 bits)
HA Length (8 bits)	PA Length (8 bits)	Operation (16 bits)
Sender Hardware Address (Octets 0-3)		
Sender Hardware Address (Octets 4-5)		Sender Protocol Address (Octets 0-1)
Sender Protocol Address (Octets 2-3)		Target Hardware Address (Octets 0-1)
Target Hardware Address (Octets 2-5)		
Target Protocol Address (Octets 0-3)		

Name mapping: IP address <-> MAC address

ARP & RARP



a. ARP request is broadcast



b. ARP reply is unicast

Network Topology

Take SJTU network for example

- Subnet: usually like 192.168.0.2 or 10.0.0.2
- Gateway: usually like 192.168.0.1
  - Get the global IP address: 202.120.40.82
  - A gateway usually has two (or more) IP address
- Proxy: get proxy's address
  - E.g., 106.185.46.164 (Japan)

Network Topology

How to Use socket to Access www.baidu.com ?

You code as if your PC connect directly with Baidu

- Call connect() with Baidu's IP address

But how does the system find next hop?

Putting All Together

App: I want to send a packet to Baidu, here is the packet with Baidu's IP in its header as target IP, and client's IP as source IP (Node-C)

OS: I don't know how to get to Baidu, I'll just send it to the router (gateway). But I cannot change the source IP of the packet, so I'll just change the MAC target address of the packet to the router's MAC address

Putting All Together

The router-1 (gateway): I get a packet with my MAC as target address. Is it my IP? No… So I'll just forward it to next hop, by changing the target MAC address to next hop's MAC address (NAT: change source IP and source port as well)

Router-2: I connect directly to Baidu, I'll just change the target MAC address to Baidu

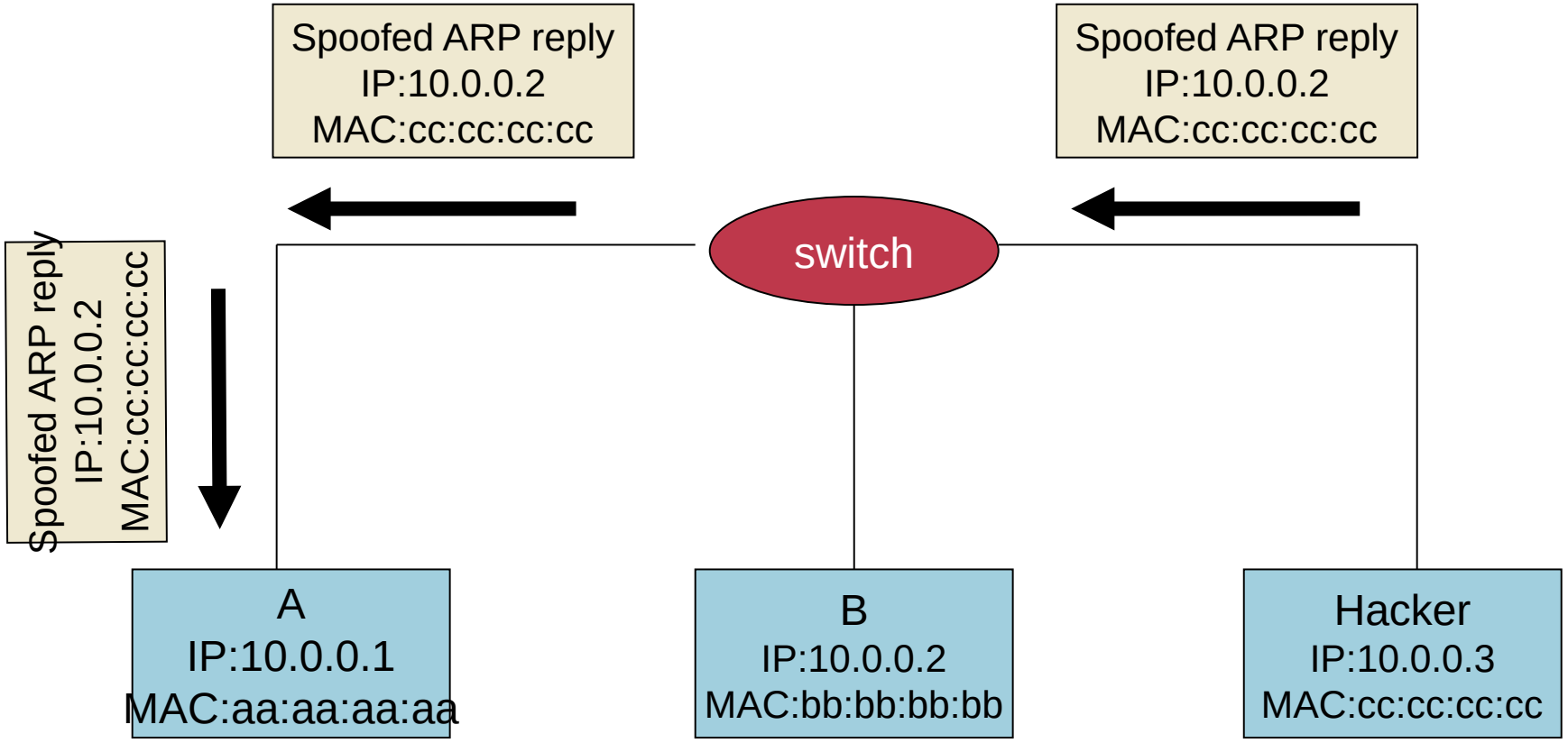
ARP Spoofing

Construct spoofed ARP replies

A target computer could be convinced to send frames destined for computer A to instead go to computer B

Computer A will have no idea that this redirection took place

This process of updating a target computer's ARP cache is referred to as "ARP poisoning"

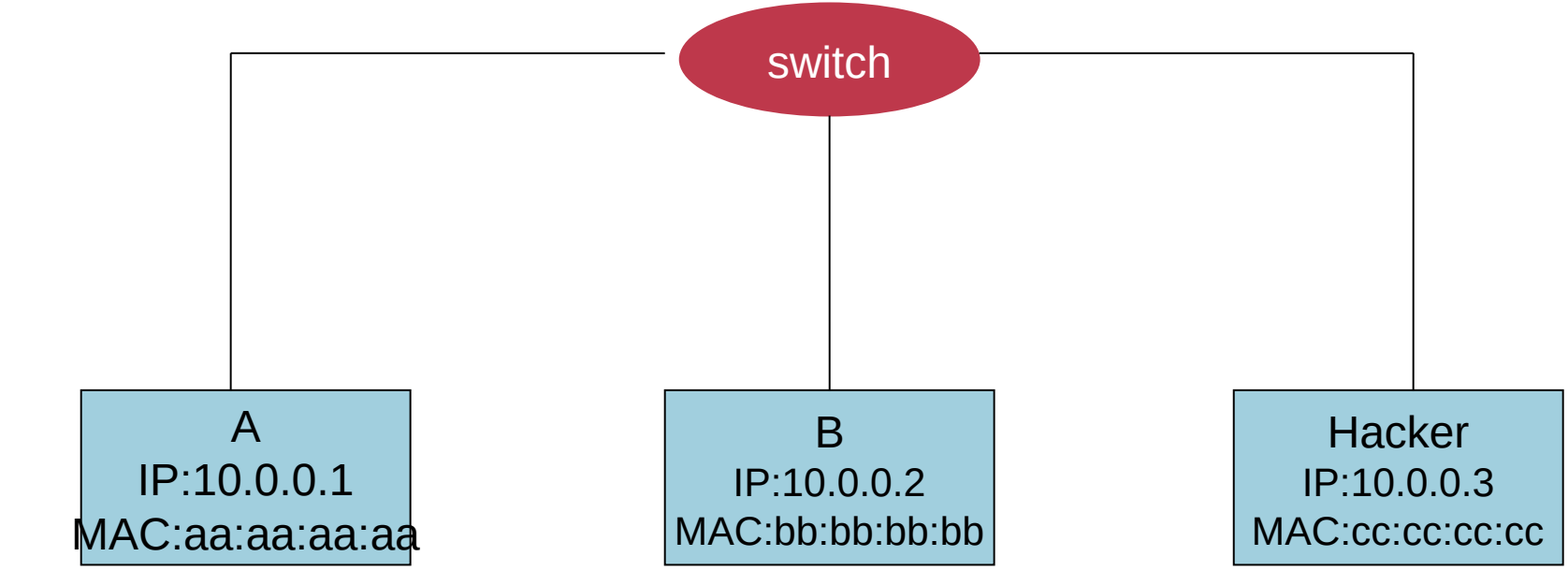


ARP cache

IP	MAC
10.0.0.2	bb:bb:bb:bb

ARP cache

IP	MAC
10.0.0.1	aa:aa:aa:aa



A's cache is poisoned

ARP cache

IP	MAC
10.0.0.2	cc:cc:cc:cc

ARP cache

IP	MAC
10.0.0.1	aa:aa:aa:aa

ARP Spoofing

Now all the packets that A intends to send to B will go to the hacker's machine

Cache entry would expire, so it needs to be updated by sending the ARP reply again

- How often?
- depends on the particular system
- Usually every 40s should be sufficient

In addition the hacker may not want his Ethernet driver talk too much

- Accomplish with ifconfig -arp

Man-in-the-Middle Attack

A hacker inserts his computer between the communications path of two target computers

The hacker will forward frames between the two target computers so communications are not interrupted

E.g., Hunt, Ettercap, etc.

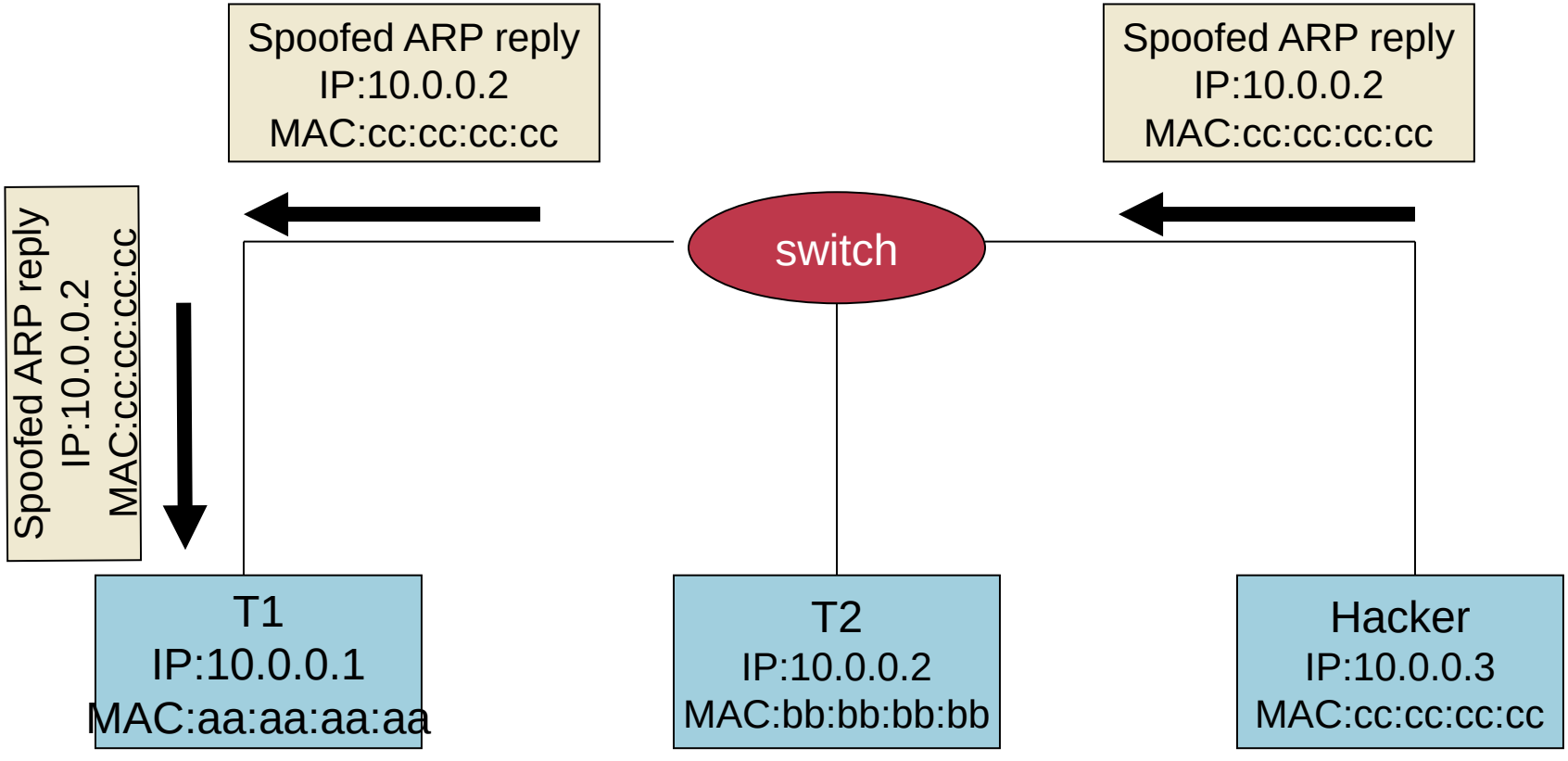
- Can be obtained easily in many web archives

Man-in-the-Middle Attack

The attack is performed as follows:

- Suppose X is the hacker's computer
- T1 and T2 are the targets

1. X poisons the ARP cache of T1 and T2
2. T1 associates T2's IP with X's MAC
3. T2 associates T1's IP with X's MAC
4. All of T1 and T2's traffic will then go to X first, instead of directly to each other

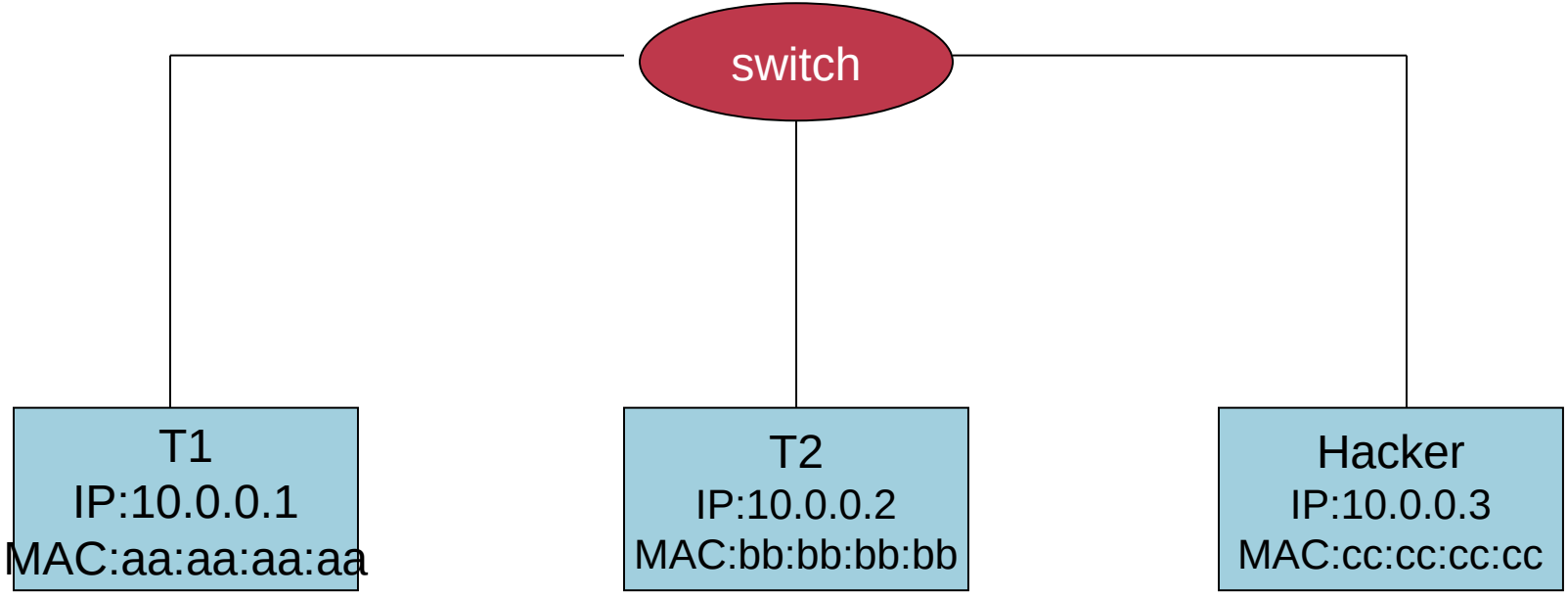


ARP cache

IP	MAC
10.0.0.2	bb:bb:bb:bb

ARP cache

IP	MAC
10.0.0.1	aa:aa:aa:aa



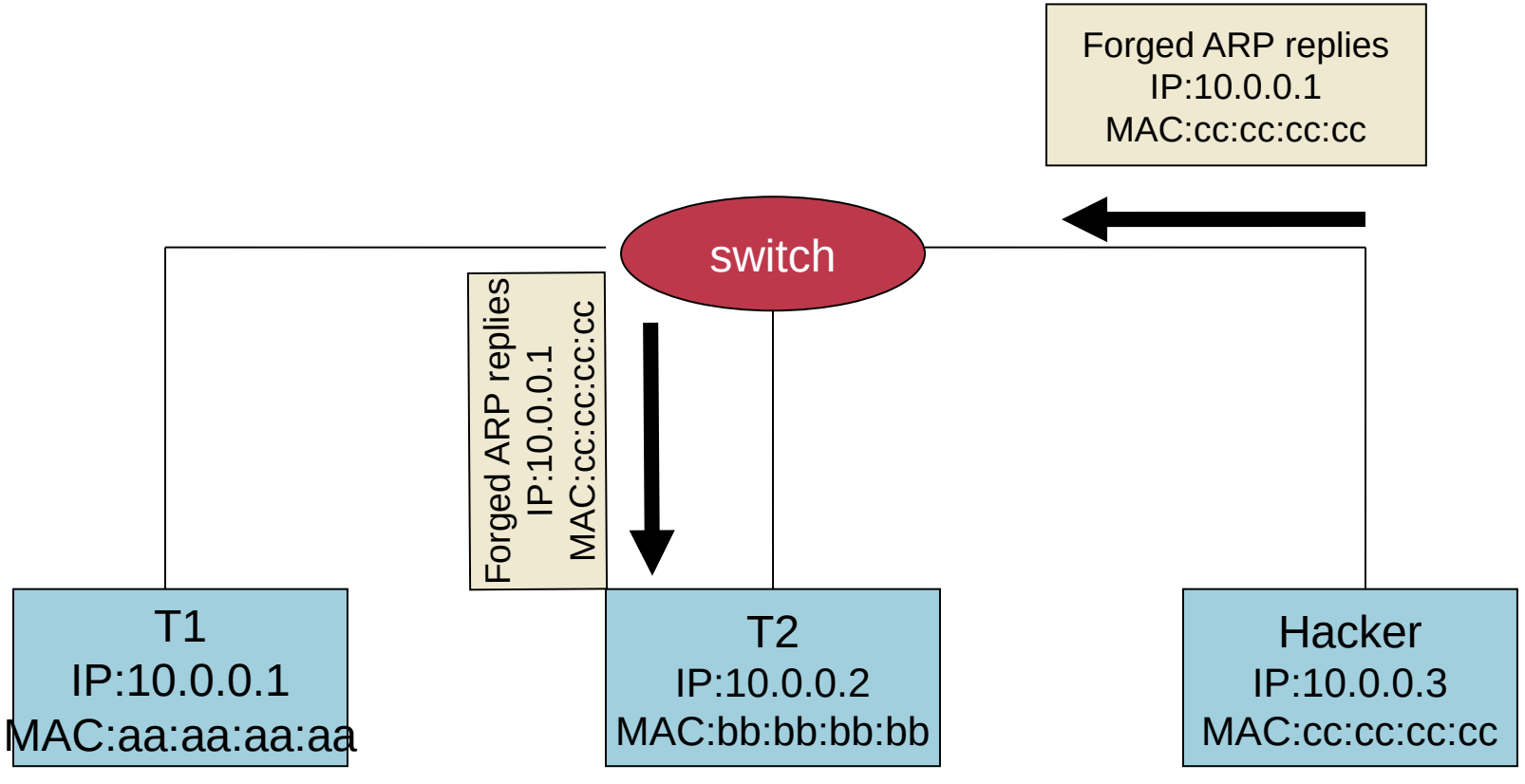
ARP cache

IP	MAC
10.0.0.2	cc:cc:cc:cc

ARP cache

IP	MAC
10.0.0.1	aa:aa:aa:aa

T1's cache is poisoned

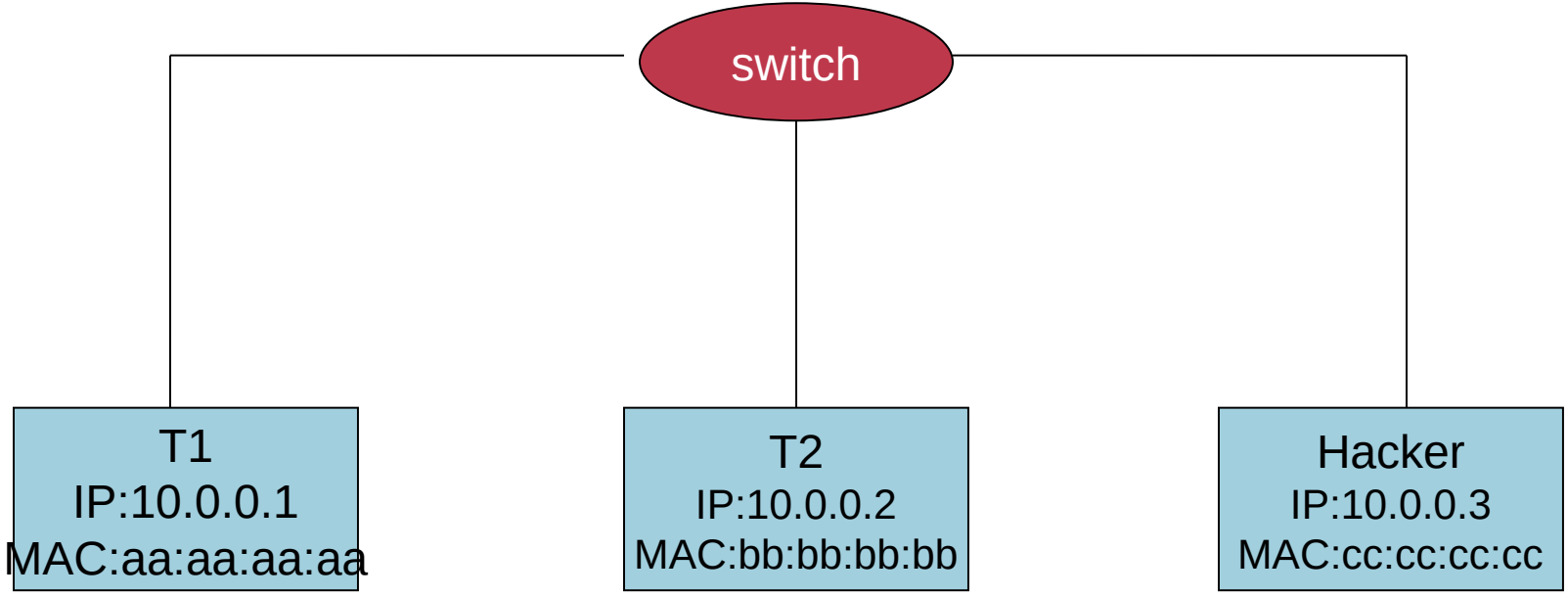


ARP cache

IP	MAC
10.0.0.2	cc:cc:cc:cc

ARP cache

IP	MAC
10.0.0.1	aa:aa:aa:aa



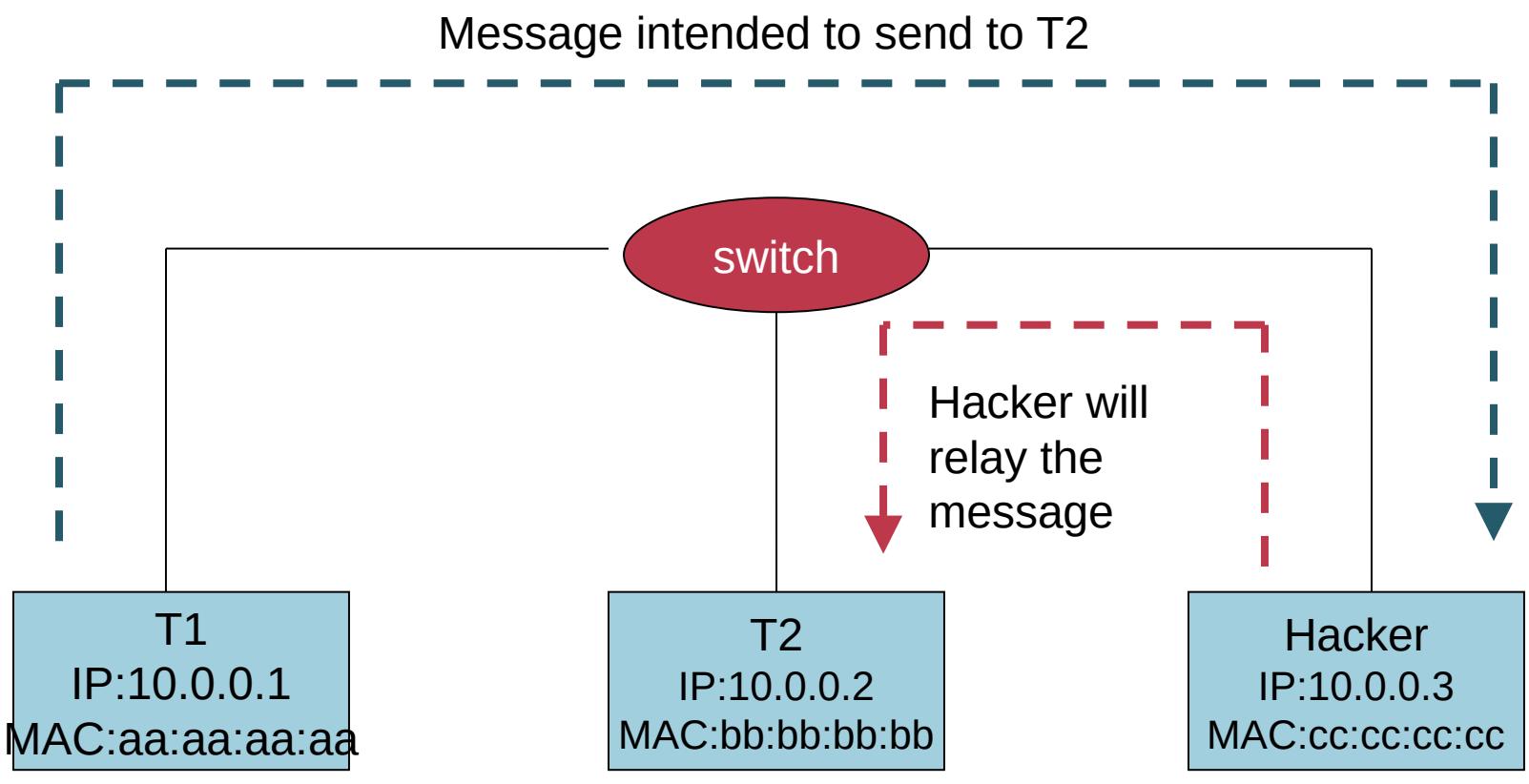
ARP cache

IP	MAC
10.0.0.2	cc:cc:cc:cc

ARP cache

IP	MAC
10.0.0.1	cc:cc:cc:cc

T2's cache is poisoned

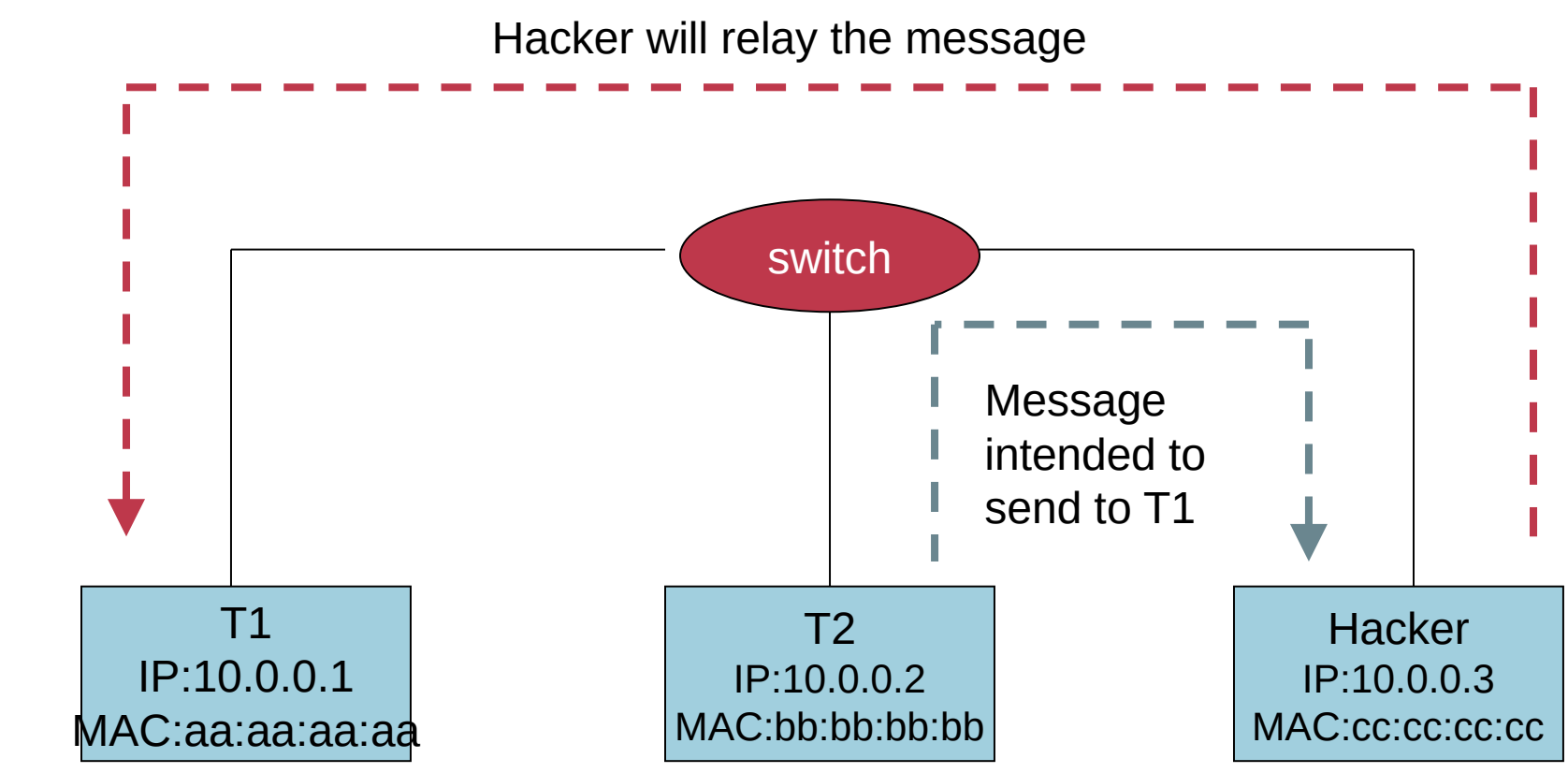


ARP cache

IP	MAC
10.0.0.2	cc:cc:cc:cc

ARP cache

IP	MAC
10.0.0.1	cc:cc:cc:cc



ARP cache

IP	MAC
10.0.0.2	cc:cc:cc:cc

ARP cache

IP	MAC
10.0.0.1	cc:cc:cc:cc

Defenses against ARP Spoofing

No Universal defense (!)

Use static ARP entries

- Cannot be updated; Spoofed ARP replies are ignored
- ARP table needs a static entry for each machine on the network
- Large overhead
  - Deploying these tables; Keep the table up-to-date

Arpwatch

- A free UNIX program which listens for ARP replies on a network
- Build a table of IP/MAC associations and store it in a file
- When a MAC/IP pair changes (flip-flop), an email is sent to an administrator