

CSE 222

Graduate Networking

Fall 2001

Lecture 7: Intra-domain Routing

Stefan Savage

The essence of routing



How do I get
there from **here**?

Overview

- Basics
- Intra-domain routing
 - ◆ Link state
 - ◆ Distance vector
- Inter-domain routing
 - ◆ Policy
- I'll be very surprised if we finish today...

What does a router do?

- **Forwarding**

- ◆ Move packet from input link to the appropriate output link
 - » “Next hop” only path to ultimate destination
- ◆ Purely local computation
- ◆ Must go be very fast (executed for ever packet)

- **Routing**

- ◆ Doing work so you’re sure that the “next hop” actually leads to the destination
- ◆ Distributed computation and communication
- ◆ Can go slower (only important when topology changes)

Kinds of forwarding

- Source routing
 - ◆ Complete path in packet
- Virtual circuits
 - ◆ Set up path out-of-band and store path identifier in routers
 - ◆ Local path identifier in packet
- **Global address lookup**
 - ◆ Router looks up address in forwarding table
 - ◆ Forwarding table contains (address, next-hop) tuples

Source routing

- Packet contains complete ordered path information
 - ◆ I.e. node A then D then X then J...
- Host computes path
- Router looks up next hop in packet header
- Strips next hop and forwards remaining packet

Source routing evaluation

- Strengths
 - ◆ Very fast to lookup next hop
 - ◆ Very flexible (every packet can take different path)
- Weaknesses
 - ◆ Host must know global topology and detect failures
 - ◆ Variable packet header up to max path
- In practice
 - ◆ Ad hoc networks (DSR)
 - ◆ Multicomputer (Paragon) and SAN networks (Myrinet)
 - ◆ Minimal Internet support (LSR, SSR)

Virtual circuits

- Setup path out-of-band
 - ◆ Enter (input id, output id, next hop) entry into each router on path
 - ◆ Provide initial local input id to sending host as path id
- Forwarding
 - ◆ Send packet with path id
 - ◆ Router looks up input, swaps for output, forwards on next hop
 - ◆ Repeat until reach destination

Virtual circuit evaluation

- Strengths
 - ◆ Table lookup for forwarding (why faster than IP lookup?)
 - ◆ Flexible (one path per flow)
- Weaknesses
 - ◆ Requires connection setup before can send
 - ◆ Complicated to deal with failures
- In practice
 - ◆ ATM: fixed VC identifiers and separate signaling code
 - ◆ MPLS: ATM meets the IP world (why? *traffic engineering*)

Global address lookup

- All addresses are globally known
- Host sends packet with destination address in header
- Router maintains forwarding table
 - ◆ (Address, next-hop) tuple
 - ◆ Lookup address, send packet to next-hop link
- Distributed routing protocol used to populate tables

Global address evaluation

- Strengths
 - ◆ Handles failures well; No path state, so any router can forward any packet
 - ◆ No connection setup required
- Weaknesses
 - ◆ Inflexible
 - » Usually all packets to destination follow same path
 - ◆ More state
 - » Must store information on all destinations even if never used
 - ◆ Forwarding lookup more expensive
 - » This is a whole lecture in itself; not now
- In practice
 - ◆ IP routing

Intra-domain routing

- Routing **within** a network/organization
- A **single** administrative domain

- Overall goals
 - ◆ Provide intra-network connectivity
 - ◆ Adapt quickly to failures or topology changes
 - ◆ **Optimize** use of network resources

- Problem statement
 - ◆ Network is a directed graph $G=(V,E)$
 - ◆ Routers are vertices, links are edges labeled with some metric
 - » For simplicity ignore hosts, they are part of each V
 - ◆ For each V , find the shortest path to every other V

Quick aside: host routing

- Generally, hosts are *single-homed*
 - ◆ Connected to a single network
- Don't need to understand topology
- Can simply have a *default* route
 - ◆ All non-local traffic sent to default next hop (a router)
 - ◆ Router maintains “default-free” forwarding table (or knows how to get to a router that does)

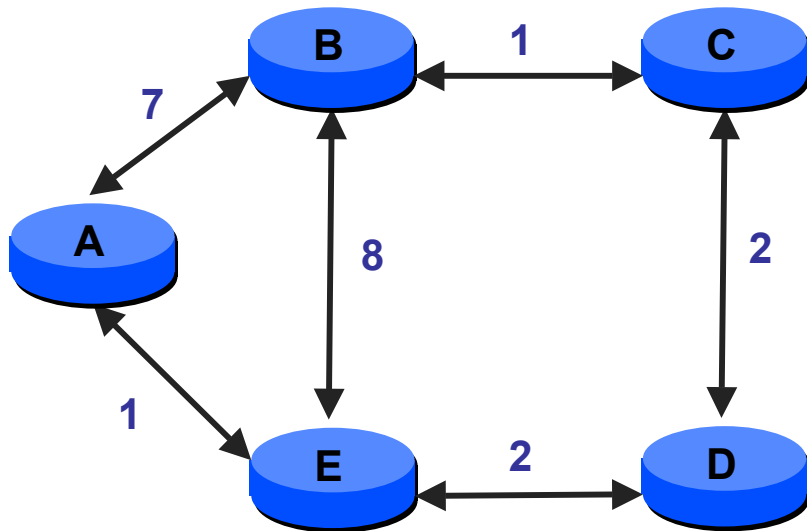
Three approaches

- **Static**
 - ◆ Type in the right answers and hope they are always true
- **Distance vector**
 - ◆ Tell your neighbors when you know about everyone
- **Link state**
 - ◆ Tell everyone what you know about your neighbors

Distance Vector routing

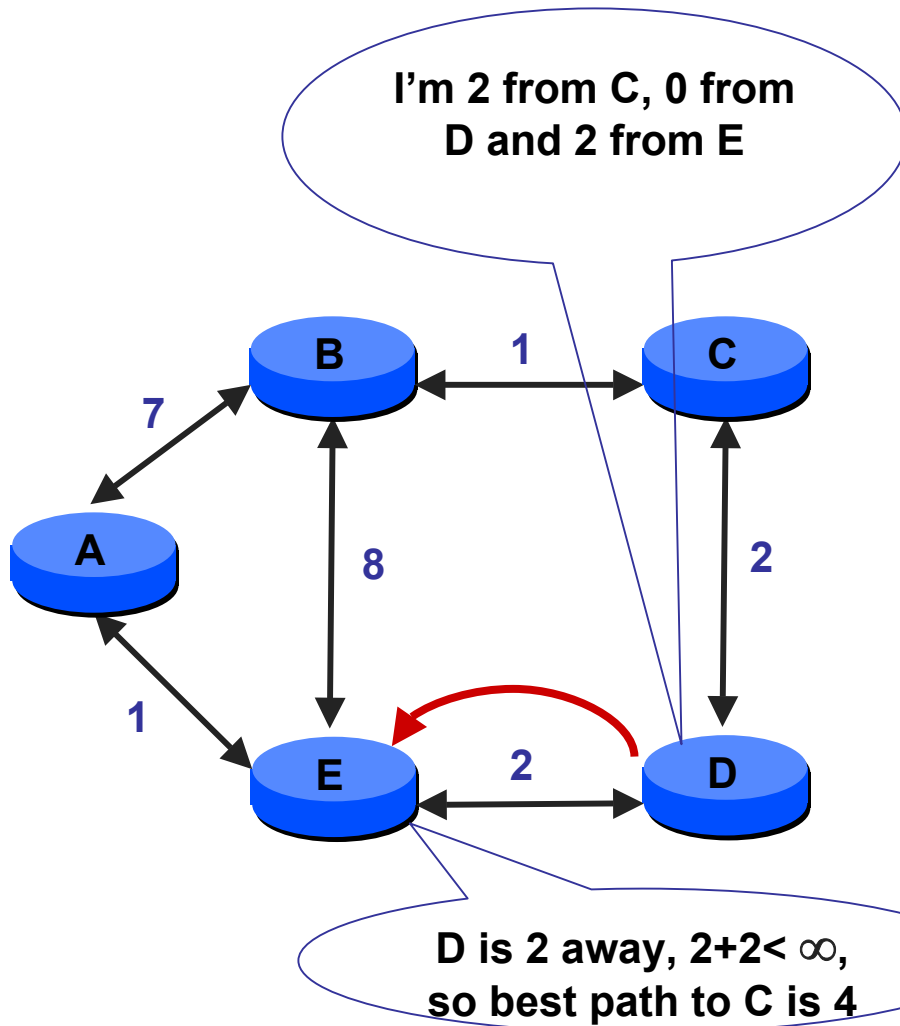
- Assume
 - ◆ Each router knows own address & cost to reach neighbors
- Goal
 - ◆ Calculate routing table containing next-hop information for every destination at each router
- **Distributed Bellman-Ford algorithm**
 - ◆ Each router maintains a vector of costs to all destinations
 - » Initialize neighbors with known cost, others with infinity
 - ◆ Periodically send copy of distance vector to neighbors
 - ◆ On reception of a vector
 - » If neighbor's path to a destination is shorter, switch to it

Initial conditions



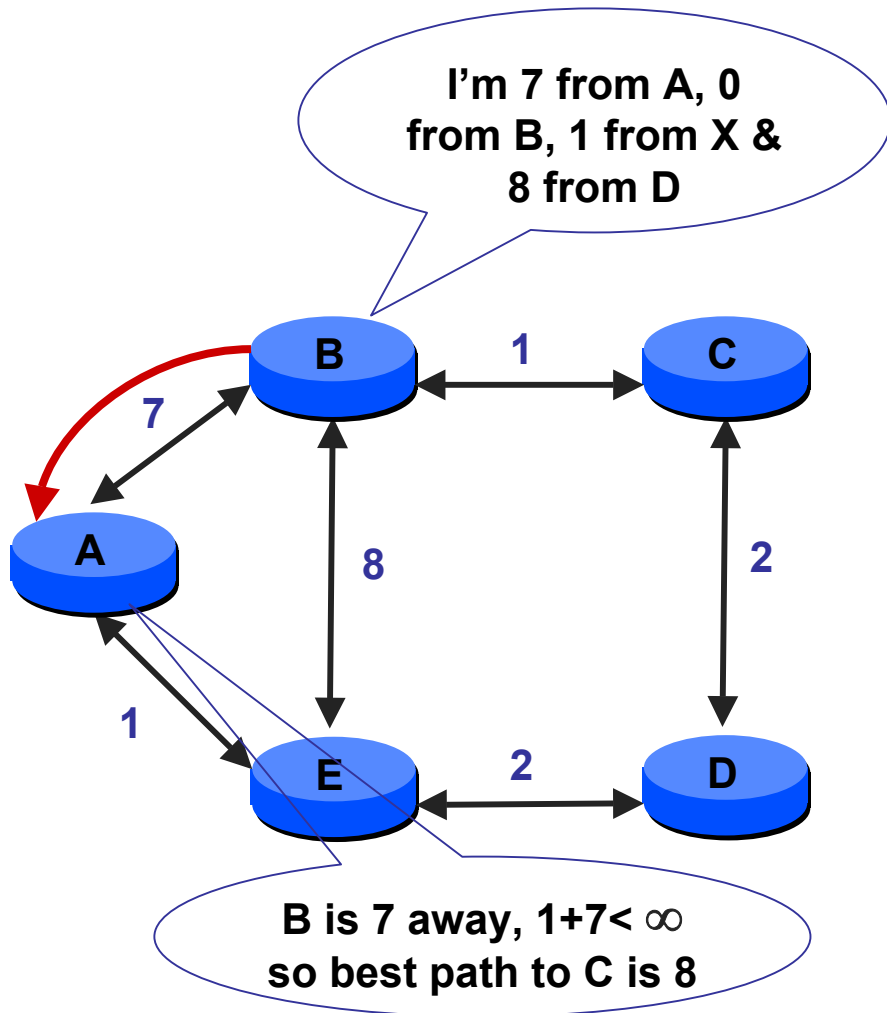
Info at node	Distance to Node				
	A	B	C	D	E
A	0	7	∞	∞	1
B	7	0	1	∞	8
C	∞	1	0	2	∞
D	∞	∞	2	0	2
E	1	8	∞	2	0

E receives D's vector



Info at node	Distance to Node				
	A	B	C	D	E
A	0	7	∞	∞	1
B	7	0	1	∞	8
C	∞	1	0	2	∞
D	∞	∞	2	0	2
E	1	8	4	2	0

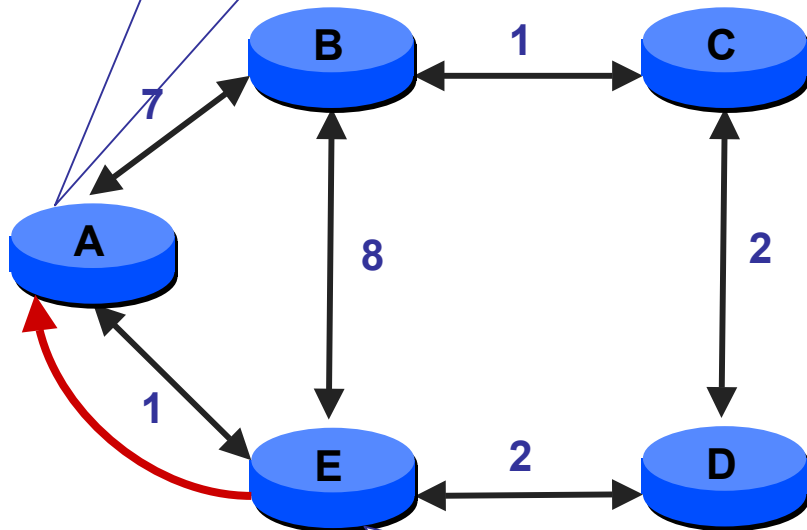
A receives B's vector



Info at node	Distance to Node				
	A	B	C	D	E
A	0	7	8	∞	1
B	7	0	1	∞	8
C	∞	1	0	2	∞
D	∞	∞	2	0	2
E	1	8	4	2	0

A receives E's vector

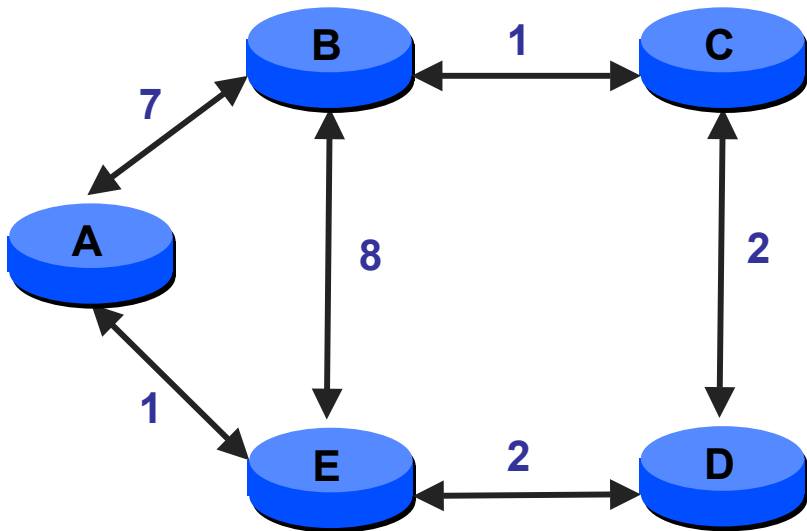
E is 1 away, $4+1 < 8$ so
 C is 5 away, $1+2 < \infty$
 so D is 3 away



I'm 1 from A, 8
 from B, 4 from C, 2
 from D & 0 from E

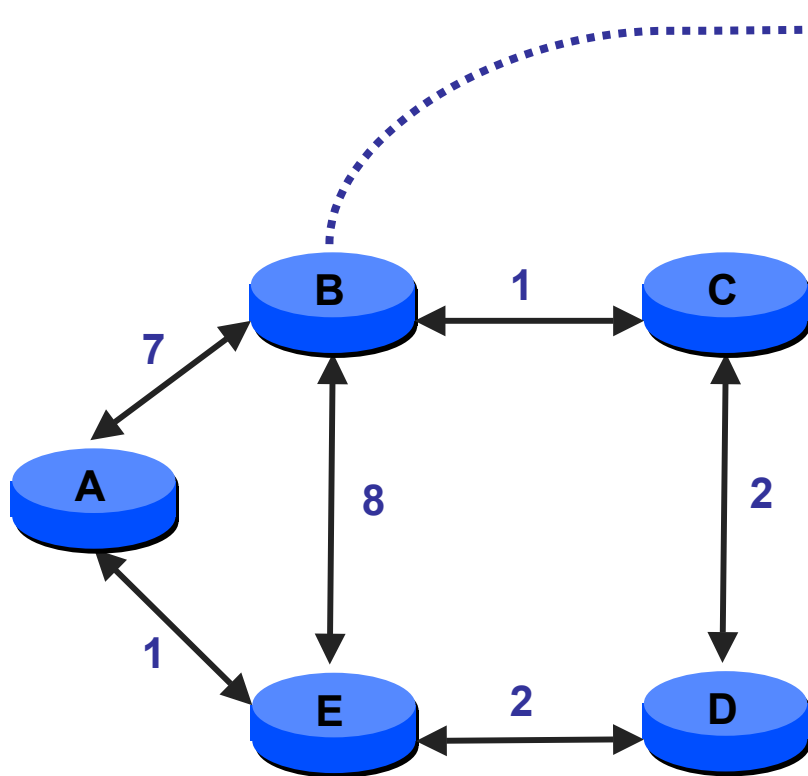
Info at node	Distance to Node				
	A	B	C	D	E
A	0	7	5	3	1
B	7	0	1	∞	8
C	∞	1	0	2	∞
D	∞	∞	2	0	2
E	1	8	4	2	0

Final state



Info at node	Distance to Node				
	A	B	C	D	E
A	0	6	5	3	1
B	6	0	1	3	5
C	5	1	0	2	4
D	3	3	2	0	2
E	1	5	4	2	0

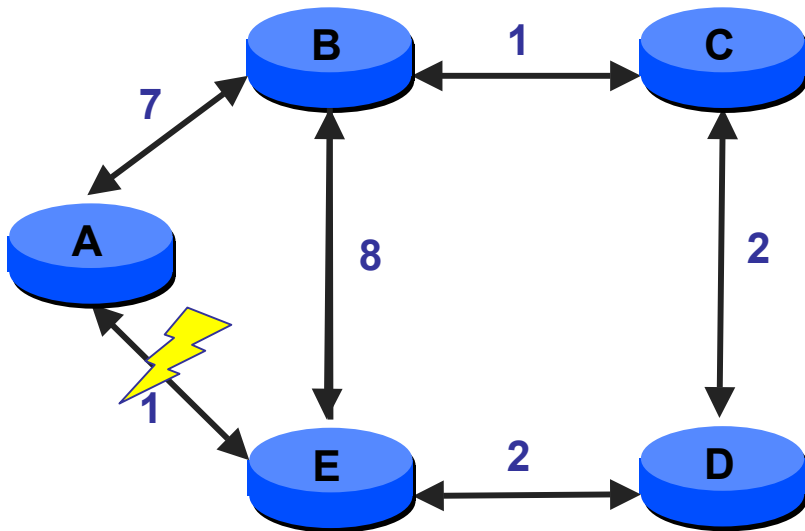
View from a node (B)



Dest	Next hop			
	A	A	E	C
A	0	7	9	6
B	6	0	13	2
C	5	12	12	1
D	3	10	10	3
E	1	8	8	5

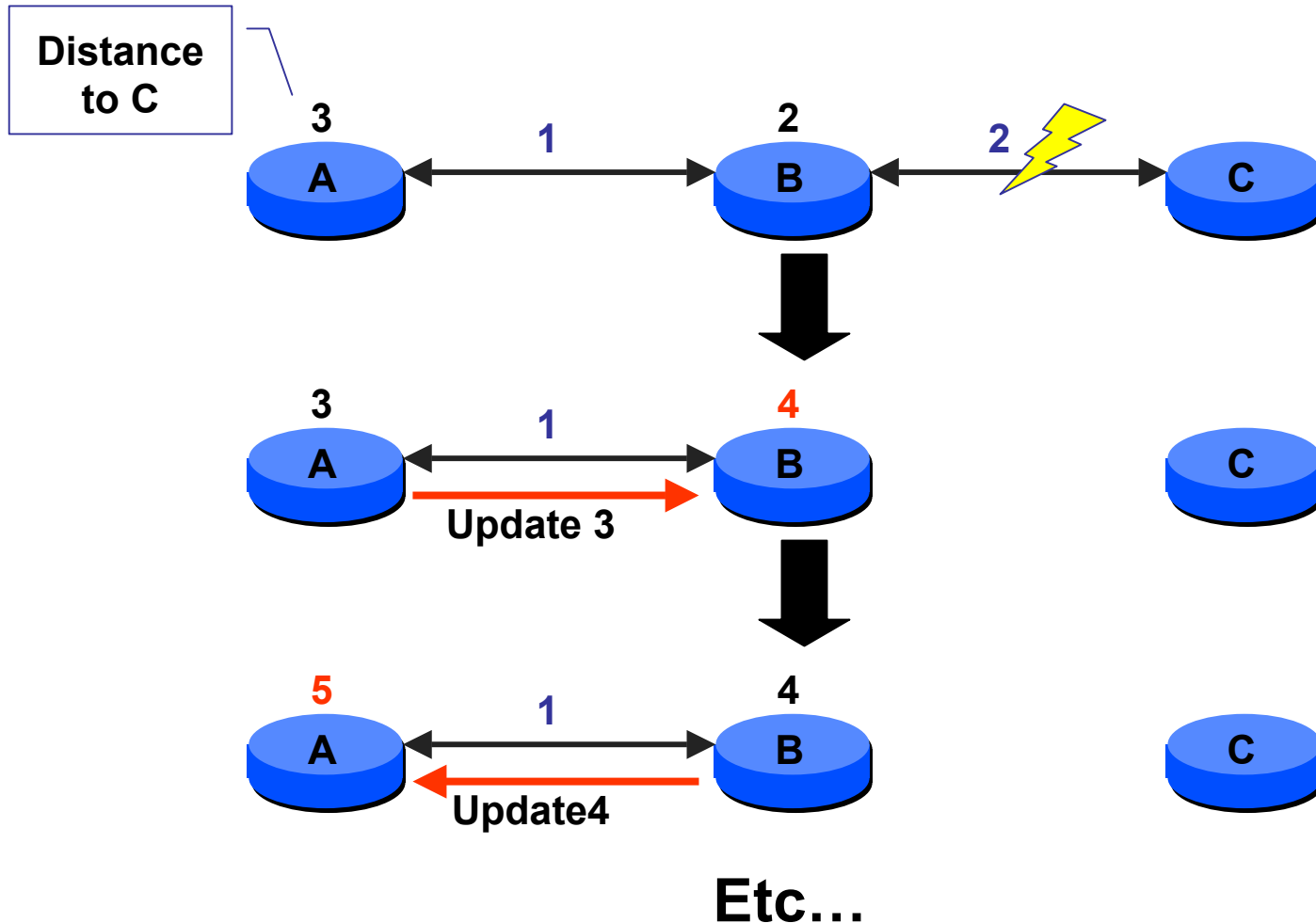
Link failure

- A marks distance to E as x, and tells B
- E marks distance to A as x, and tells B and D
- B and D recompute routes and tell C, E and E
- etc... until converge



Info at node	Distance to Node				
	A	B	C	D	E
A	0	7	8	10	12
B	7	0	1	3	5
C	8	1	0	2	4
D	3	3	2	0	2
E	12	5	4	10	0

Problems: *Count to Infinity*



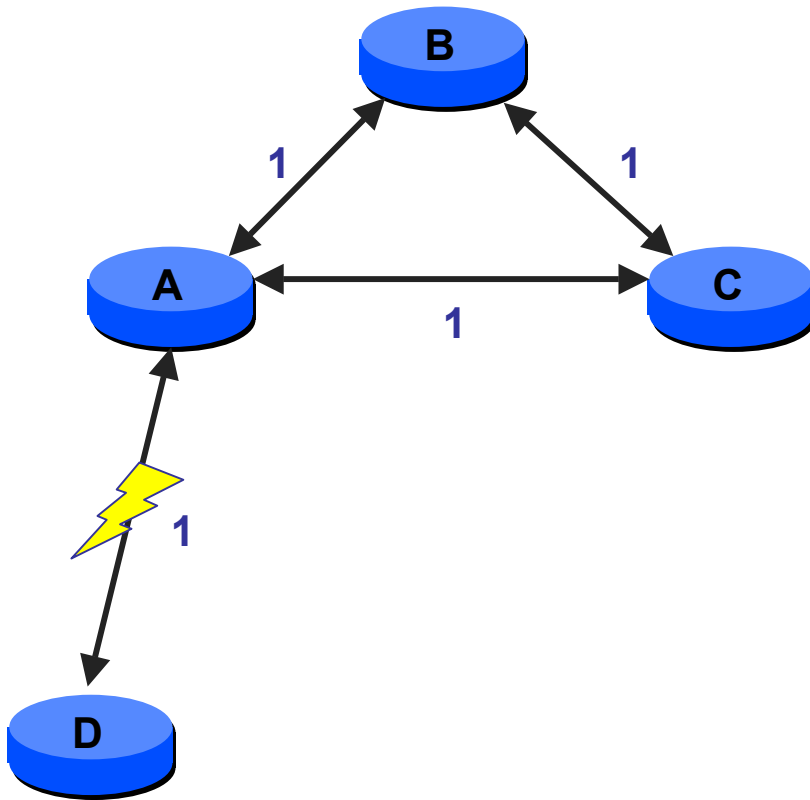
Why?

- Updates don't contain enough information
- Can't totally order bad news above good news
- B's accepts A's path to C that is *implicitly* through B!
- Aside: this also causes delays in convergence

Solutions

- Hold downs
 - ◆ As metric increases, delay propagating information
 - ◆ Limitation: Delays convergence
- Split horizon
 - ◆ Never advertise a destination through its next hop
 - » A doesn't advertise C to B
 - ◆ Poison reverse: Send negative information when advertising a destination through its next hop
 - » A advertises C to B with a metric of ∞
 - ◆ Limitation: Only works for "loop"s of size 2
- Loop avoidance
 - ◆ Full path information in route advertisement
 - ◆ Explicit queries for loops (e.g. DUAL)

How split horizon/pv fails



- A tells B & C that D is unreachable
- B tells C that D is unreachable
- B tells A that D is reachable with cost=3 (since route is through C, split horizon doesn't apply)
- A tells C that D is reachable through A (cost=4)
- Etc...

Other issues

- When to send route updates?
- Periodically
 - ◆ Limits granularity of failure recovery
 - ◆ Global synchronization can cause packet loss
- Jittered
 - ◆ Random offset from periodic deals with synchronization problem
- Triggered
 - ◆ Send updates immediately when metric changes
 - ◆ Converges more quickly, but causes flood of packets

Distance Vector summary

- Strengths
 - ◆ Doesn't need much state (only neighbors)
 - ◆ Doesn't put much overhead on network
- Weaknesses
 - ◆ Loop problems in large or complex networks
 - ◆ Must send large messages
 - ◆ Convergence is very slow in worst case
 - » Much better with triggered updates

Distance Vector in practice

- RIP
 - ◆ Small infinity (RIPv1, inf=16)
 - ◆ Split horizon/poison reverse
 - ◆ Jittered 30 second periodic updates
 - ◆ Triggered updates on failure
 - ◆ Metric is hop count
- EIGRP (Cisco proprietary)
 - ◆ Uses DUAL algorithm to avoid loops at all times
 - ◆ Keeps track of alternate loop-free next hops; explicit queries for loop-free paths otherwise
- BGP
 - ◆ Full path information to avoid loops

Link State routing

- Same goal, different approach
- Two phases
 - ◆ **Reliable flooding**
 - » Tell **all** routers what you know about your **local** topology
 - ◆ **Path calculation** (Dijkstra's algorithm)
 - » Each router computes best path over **complete** network
- Motivation
 - ◆ Using DV, routers only have local information, making it difficult to decide what to do when there are changes
 - ◆ With LS, faster convergence and better stability (hopefully)
 - ◆ More complex

Reliable flooding

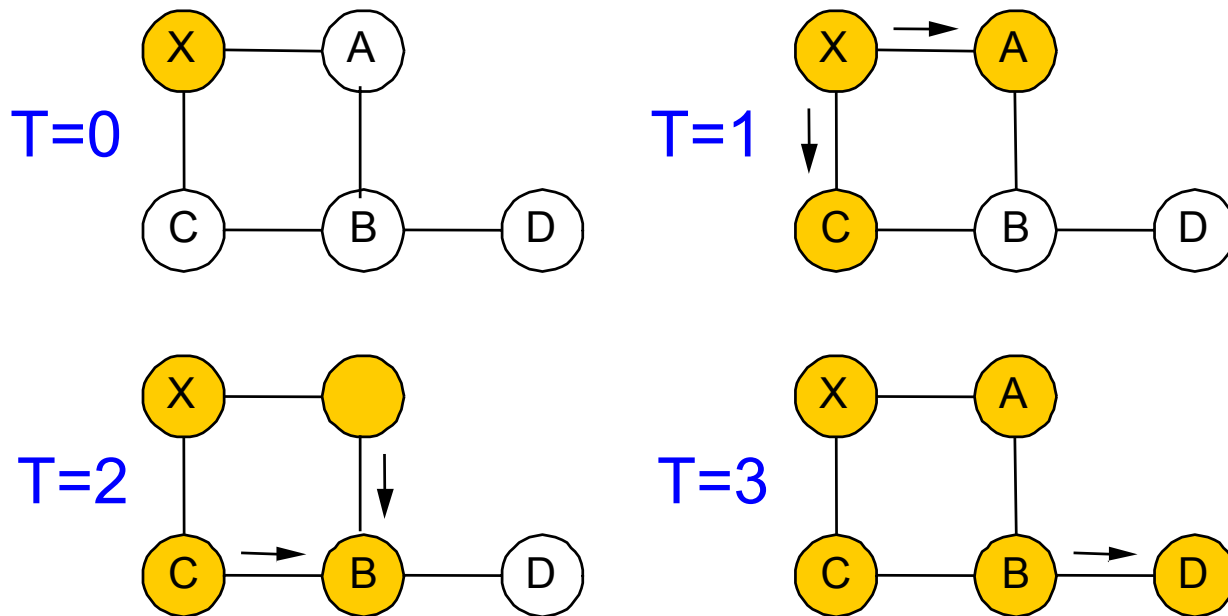
- Goal: tell everyone what you know about local topology
- Periodically send link state packets (LSPs) on **all** links
 - ◆ LSP contains [node, neighbors, costs, sequence number]
- If node X receives an LSP from node Y over link Q
 - ◆ If it is the “newest” LSP from Y that X has seen then save it in local link state database & forward LSP on all links **except** Q
 - ◆ Otherwise drop LSP
- Use explicit ACKs and retransmits to make flooding reliable
- Each LSP will travel at most once over each link

Reliable flooding challenges

- When link/router fails need to remove old data...how?
 - ◆ LSPs carry sequence numbers to distinguish new from old
 - ◆ Send a new LSP with cost infinity to signal a link down
- What happens when a router fails and restarts?
 - ◆ What sequence # should it use? Don't want data ignored
 - ◆ One option: Age LSPs and send with cost 0 to purge
 - ◆ Router can listen at startup to learn right sequence #
- What happens if the network is partitioned and heals?
 - ◆ Different LS databases must be synchronized
 - ◆ Use version #s

Flooding example

- LSP generated by X at T=0
- Nodes become orange as they receive it



Dijkstra's Shortest Path Tree (SPT) algorithm

- Graph algorithm for single-source shortest path tree

```
S ← {}
```

```
Q ← <all nodes keyed by distance>
```

```
While Q != {}
```

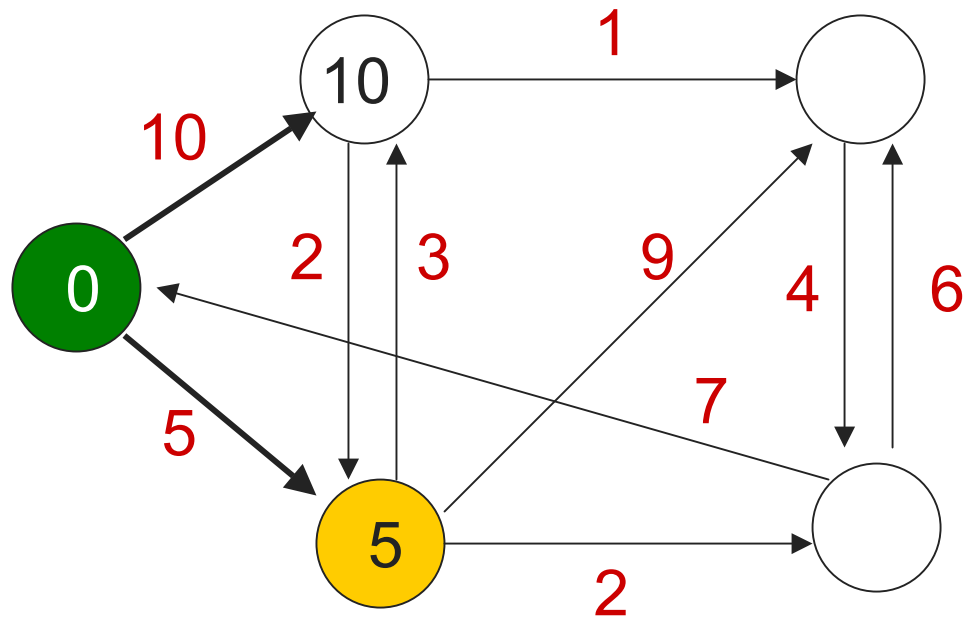
```
    u ← extract-min(Q)
```

```
    S ← S plus {u}
```

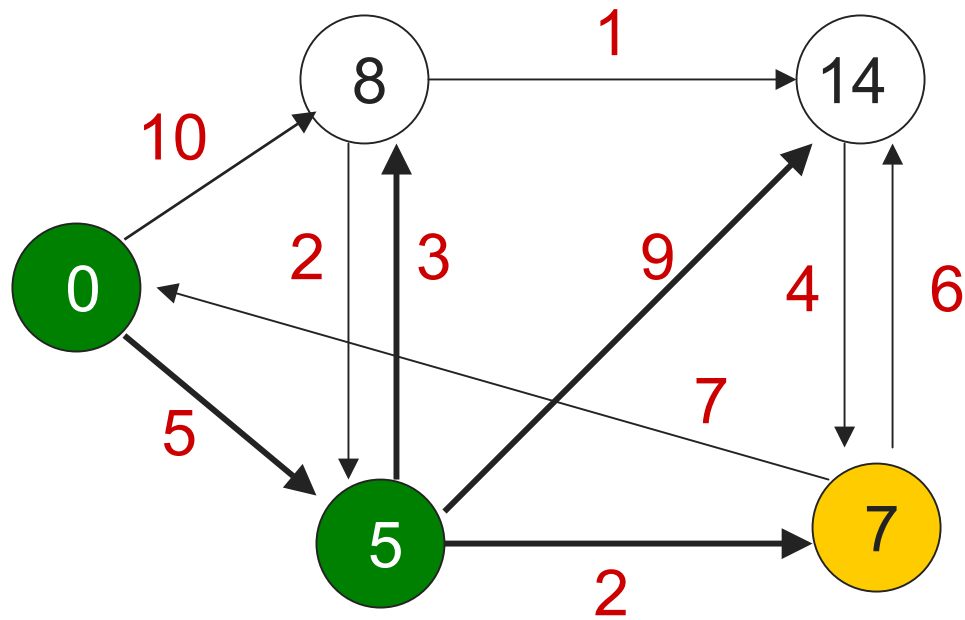
```
    for each node v adjacent to u  
        “relax” the cost of v
```

← u is done

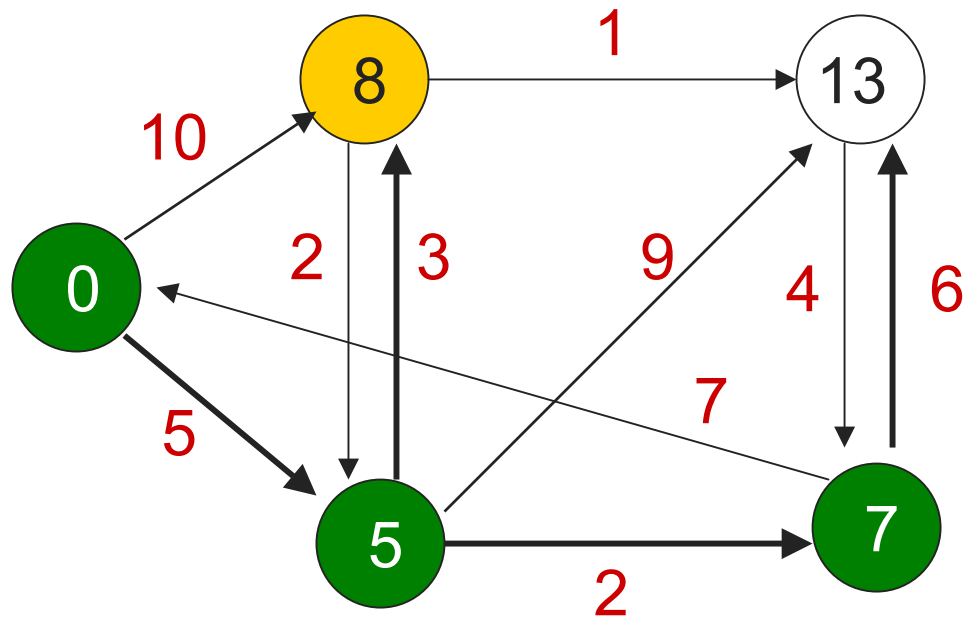
Example – Step 2



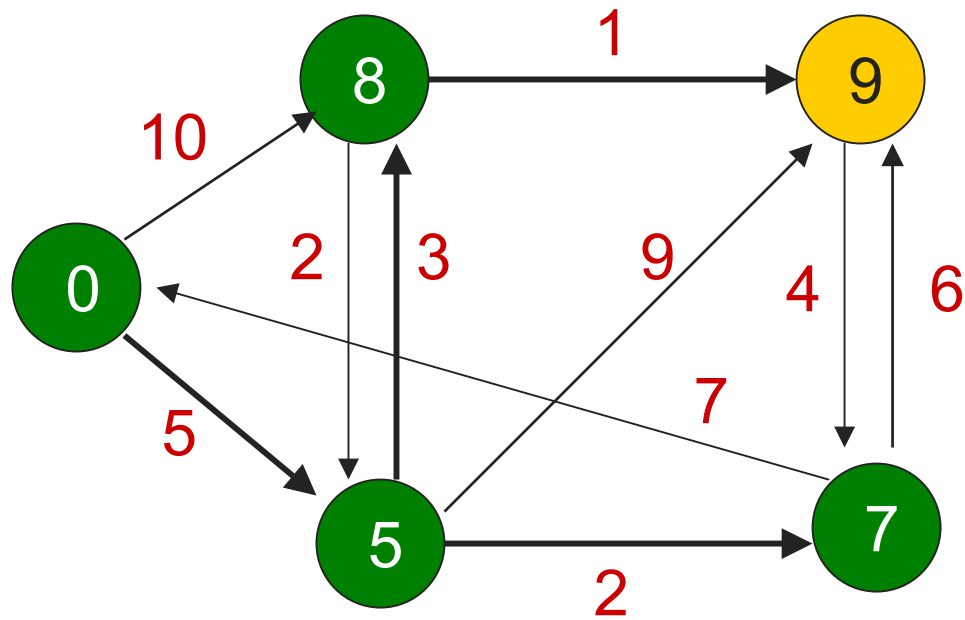
Example – Step 3



Example – Step 4



Example – Step 5



Link State evaluation

- Strengths
 - ◆ Loop free as long as LSDB's are consistent
 - » Can have transient routing loops
 - ◆ Messages are small
 - ◆ Converges quickly
- Weaknesses
 - ◆ Must flood data across entire network (scalability?)
 - ◆ Must maintain state for entire topology

Link State in practice

- OSPF (Open Shortest Path First) and IS-IS
 - ◆ Most widely used IGPs
 - ◆ Run by almost all ISPs and many large organizations
- Basic link state algorithm plus many features:
 - ◆ Authentication of routing messages
 - ◆ Extra hierarchy: Partition into routing areas
 - ◆ Load balancing: Multiple equal cost routes

Discussion

- How to pick metrics?
- How can you do load balancing?
- How does congestion impact routing?
- What if a router lies?
- What are the biggest scalability issues?