

Lecture 14:

Link-state Routing

CSE 123: Computer Networks
Stefan Savage



This class



- New topic: **routing**



How do I get
there from
here?



Lecture 14 Overview

- Routing overview
- Intra vs. Inter-domain routing
- Link-state routing protocols



Router Tasks

- Forwarding
 - ◆ Move packet from input link to the appropriate output link
 - ◆ Purely local computation
 - ◆ Must go be very fast (executed for every packet)
- Routing
 - ◆ Make sure that the next hop actually leads to the destination
 - ◆ Global decisions; distributed computation and communication
 - ◆ Can go slower (only important when topology changes)



Forwarding Options

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



Source Routing

- Routing
 - ◆ Host computes path
 - » Must know global topology and detect failures
 - ◆ Packet contains complete ordered path information
 - » I.e. node A then D then X then J...
 - ◆ Requires variable length path header
- Forwarding
 - ◆ Router looks up next hop in packet header, strips it off and forwards remaining packet
 - » Very quick forwarding, no lookup required
- In practice
 - ◆ ad hoc networks (DSR), some HPC networks (Myrinet), and for debugging on the Internet (LSR,SSR)



Virtual Circuits

- Routing
 - ◆ Hosts sets up path out-of-band, requires connection setup
 - ◆ Write (input id, output id, next hop) into each router on path
 - ◆ Flexible (one path per flow)
- Forwarding
 - ◆ Send packet with path id
 - ◆ Router looks up input, swaps for output, forwards on next hop
 - ◆ Repeat until reach destination
 - ◆ Table lookup for forwarding (why faster than IP lookup?)
- In practice
 - ◆ ATM: fixed VC identifiers and separate signaling code
 - ◆ MPLS: ATM meets the IP world (why? *traffic engineering*)

Destination-based Forwarding



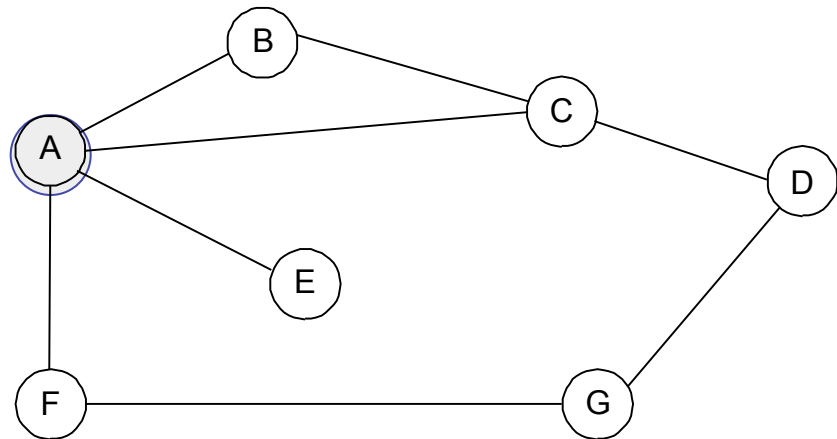
- Routing
 - ◆ All addresses are globally known
 - » No connection setup
 - ◆ Host sends packet with destination address in header
 - » No path state; only routers need to worry about failure
 - ◆ Distributed routing protocol used to routing tables
- Forwarding
 - ◆ Router looks up destination in table
 - » Must keep state proportional to destinations rather than connections
 - ◆ Lookup address, send packet to next-hop link
 - » All packets follow same path to destination
- In Practice: IP routing



Routing Tables

- The routing table at A, lists – *at a minimum* – the next hops for the different destinations

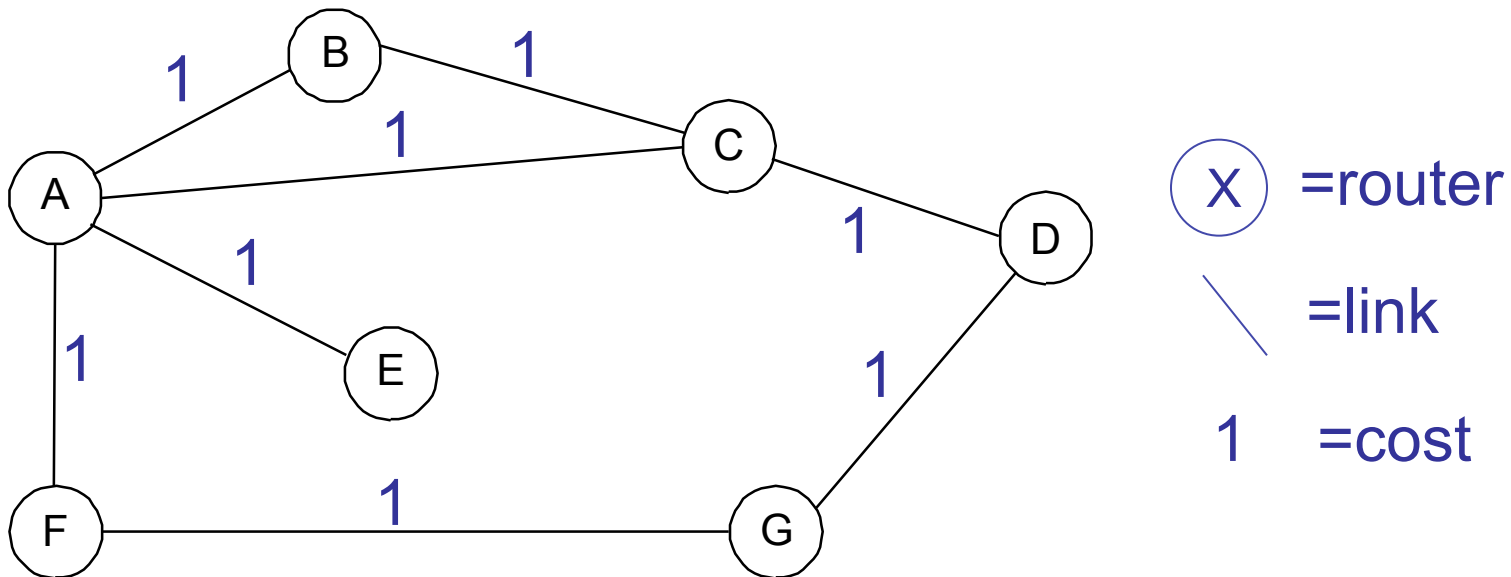
Dest	Next Hop
B	B
C	C
D	C
E	E
F	F
G	F





Routing on a Graph

- Essentially a graph theory problem
 - ◆ Network is a directed graph; routers are vertices
- Find “best” path between every pair of vertices
 - ◆ In the simplest case, best path is the shortest path





Routing Challenges

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



Intra-domain Routing

- Routing within a network/organization
 - ◆ A single administrative domain
 - ◆ The administrator can set edge costs
- Overall goals
 - ◆ Provide intra-network connectivity
 - ◆ Adapt quickly to failures or topology changes
 - ◆ Optimize use of network resources
- Non-goals
 - ◆ Extreme scalability
 - ◆ Lying, and/or disagreements about edge costs



Basic Approaches

- Static
 - ◆ Type in the right answers and hope they are always true
 - ◆ ...So far
- Link state
 - ◆ Tell everyone what you know about your neighbors
 - ◆ Today's lecture!
- Distance vector
 - ◆ Tell your neighbors when you know about everyone
 - ◆ Next time...



Link-state Routing

- 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
 - ◆ Global information allows optimal route computation
 - ◆ Straightforward to implement and verify



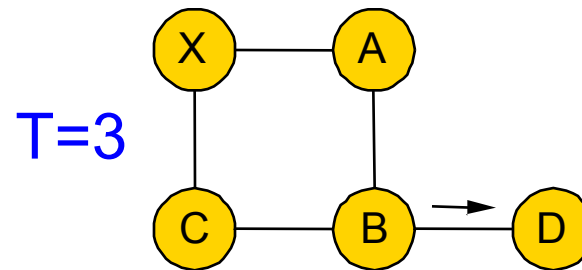
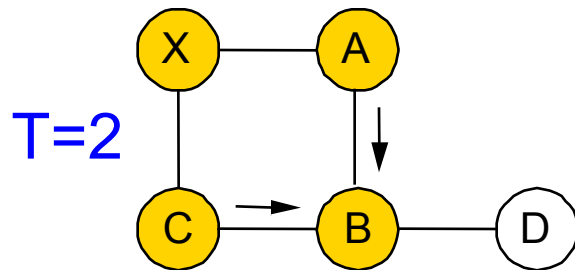
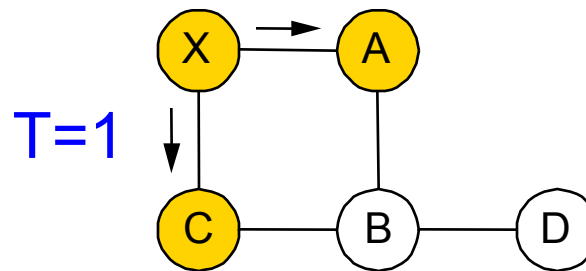
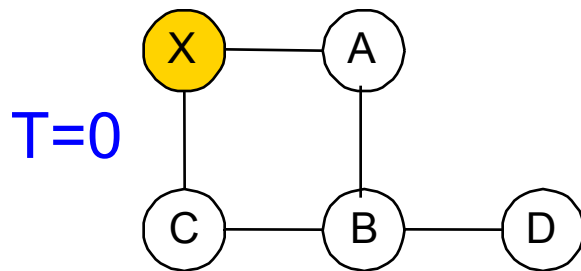
Broadcasting Link State

- Reliable flooding
 - ◆ Each router transmits a **Link State Packet** (LSP) on all links
 - ◆ A neighboring router forwards out all links except incoming
 - » Keep a copy locally; don't forward previously-seen LSPs
- Challenges
 - ◆ Packet loss
 - ◆ Out-of-order arrival
- Solutions
 - ◆ Acknowledgments and retransmissions
 - ◆ Sequence numbers
 - ◆ Time-to-live for each packet



Flooding Example

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



Dijkstra's Shortest Path Tree (SPT)



- So you have all of these LSPs. Now what?
- Graph algorithm for single-source shortest path tree (find best route to all nodes)

$S \leftarrow \{\}$

$Q \leftarrow \langle \text{remaining nodes keyed by distance} \rangle$

While $Q \neq \{\}$

$u \leftarrow \text{extract-min}(Q)$ *$u = \text{node with lowest cost}$*

$S \leftarrow S \text{ plus } \{u\}$ **$\leftarrow u \text{ is done}$**

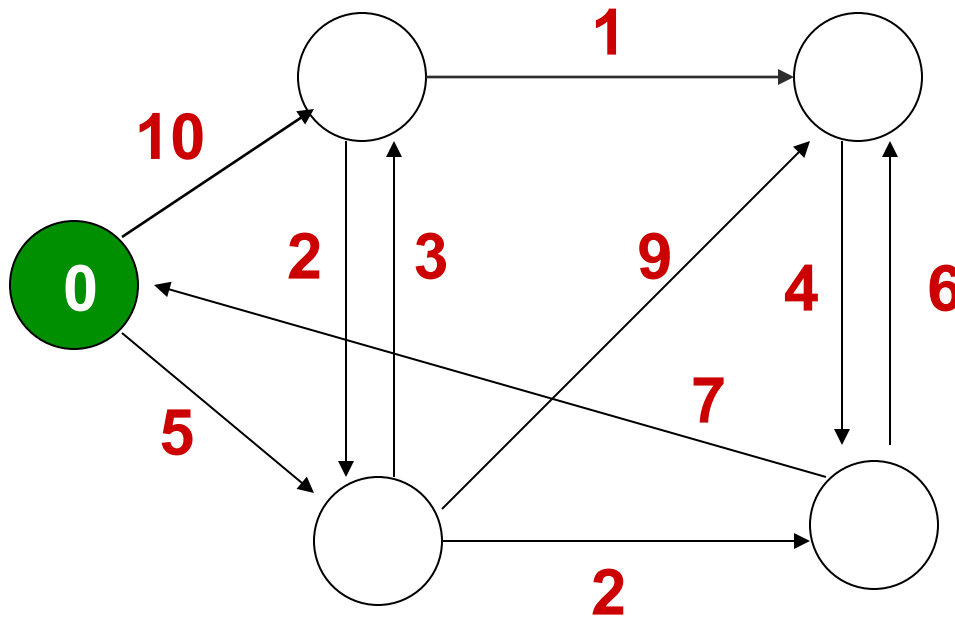
Within Q :

for each node v adjacent to u

“relax” the cost of v *is it cheaper to go through u ?*



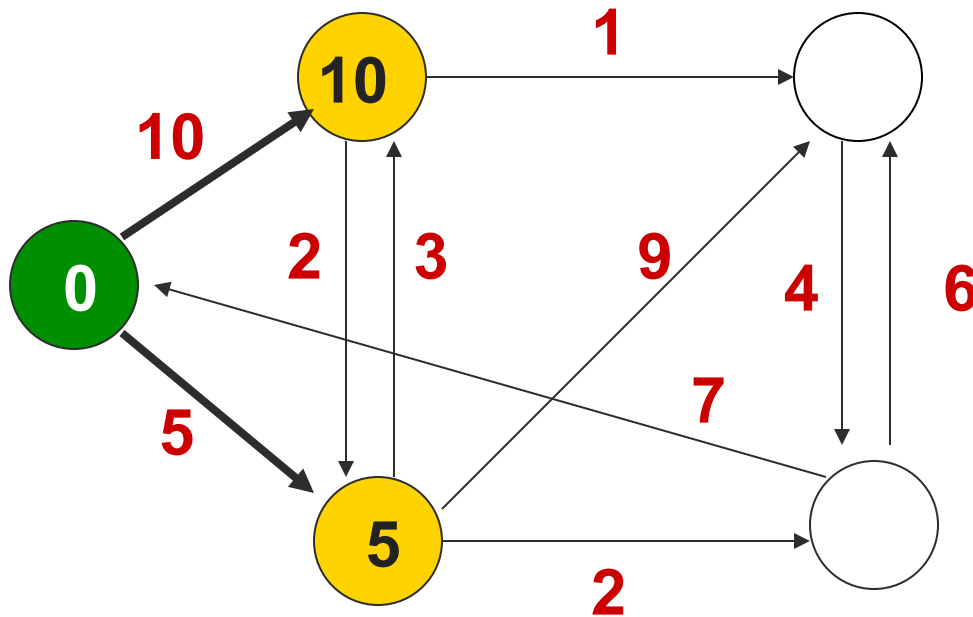
Dijkstra Example – Step 1



- Green nodes are “confirmed”
- Yellow nodes are “tentative”
- We can add ourselves to “confirmed”



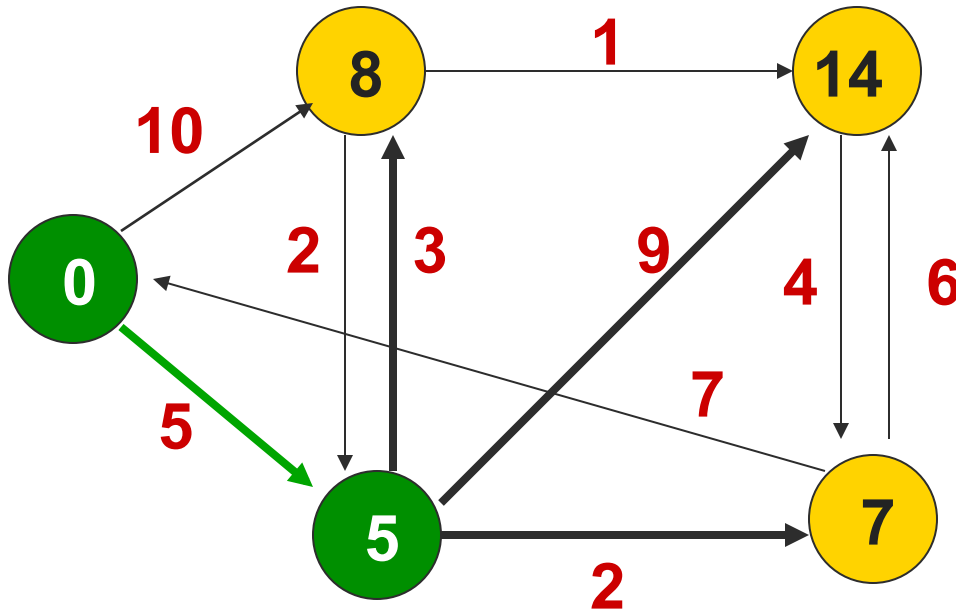
Example – Step 2



- Green nodes are “confirmed”
- Yellow nodes are “tentative”
- First look at neighbors
- “5” is cheaper than “10”
- We can confirm path with cost “5”



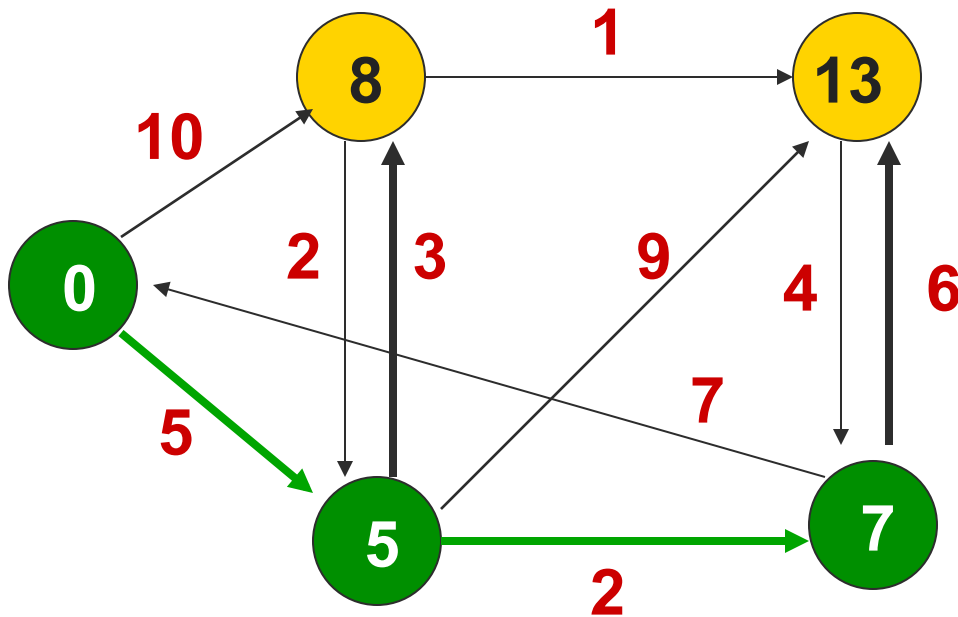
Example – Step 3



- Green nodes are “confirmed”
- Yellow nodes are “tentative”
- Update costs
- Look at 5’s neighbors
- 7 is cheapest
- We can confirm path with cost 7



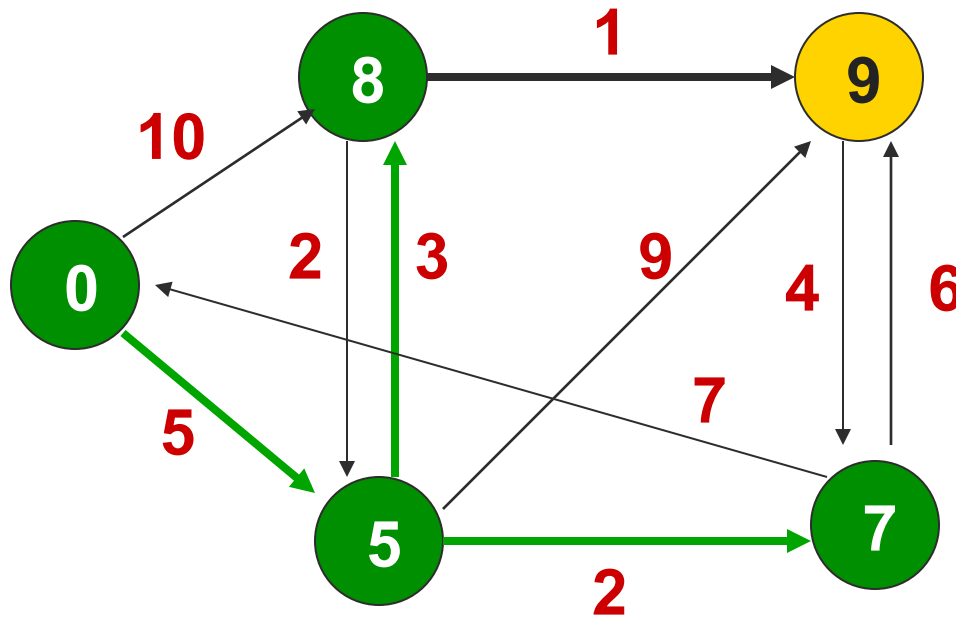
Example – Step 4



- Green nodes are “confirmed”
- Yellow nodes are “tentative”
- Update costs
- 7 has no new neighbors
- 8 is cheapest
- We can confirm 8

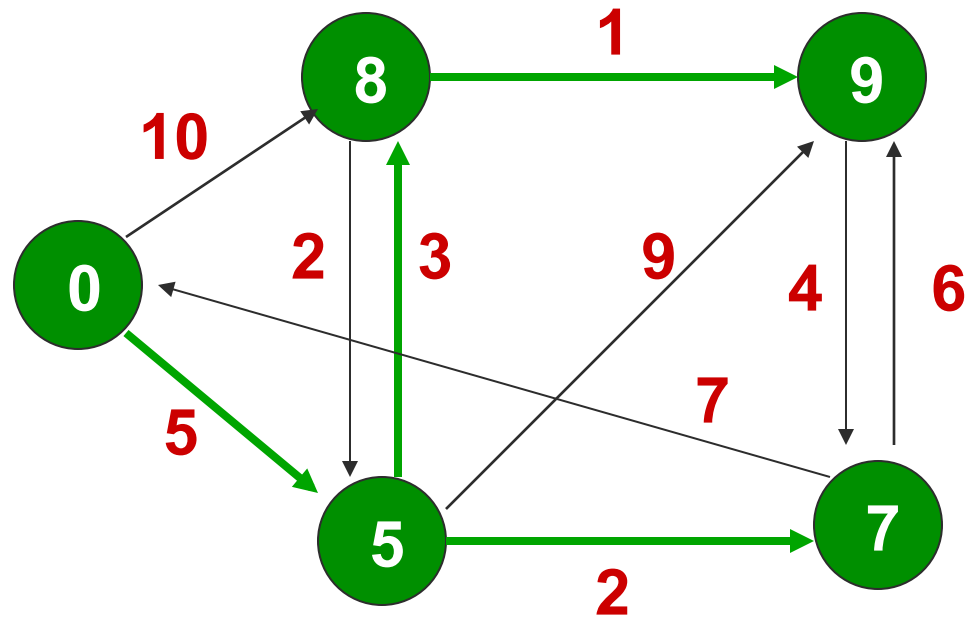


Example – Step 5



- Green nodes are “confirmed”
- Yellow nodes are “tentative”
- Update costs
- No new neighbors
- 9 is cheapest
- We can path with cost 9

Example – Done



Making Something Disappear



- Need to remove failed/old links from topology
 - ◆ LSPs carry sequence numbers to distinguish new from old
 - ◆ Routers only accept (and forward) the “newest” LSP
 - ◆ Send a new LSP with cost infinity to signal a link down
- But also need to remove entire routers
 - ◆ TTL in every LSP, decremented periodically by each router
 - ◆ When TTL = 0, purge the LSP and flood the network with an LSP with TTL 0 to tell everyone else to do the same



When to Flood?

- Triggered by a topology change
 - ◆ Link or node failure/recovery or
 - ◆ Configuration change like updated link metric
 - ◆ Converges quickly, but can cause flood of updates
- Periodically
 - ◆ Typically (say) every 30 minutes
 - ◆ Corrects for possible corruption of the data
 - ◆ Limits the rate of updates, but also failure recovery



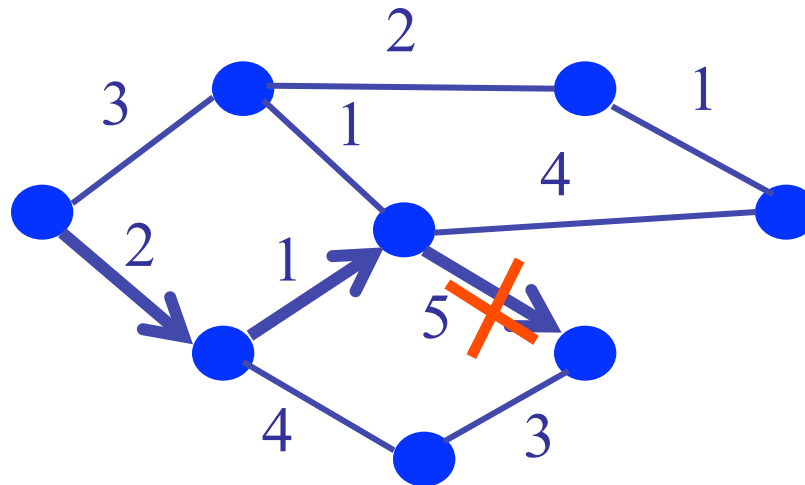
Convergence

- Getting consistent routing information to all nodes
 - ◆ E.g., all nodes having the same link-state database
 - ◆ Until routing protocol converges, strange things happen...
- Consistent forwarding after convergence
 - ◆ All nodes have the same link-state database
 - ◆ All nodes forward packets on shortest paths
 - ◆ The next router on the path forwards to the next hop



Transient Disruptions

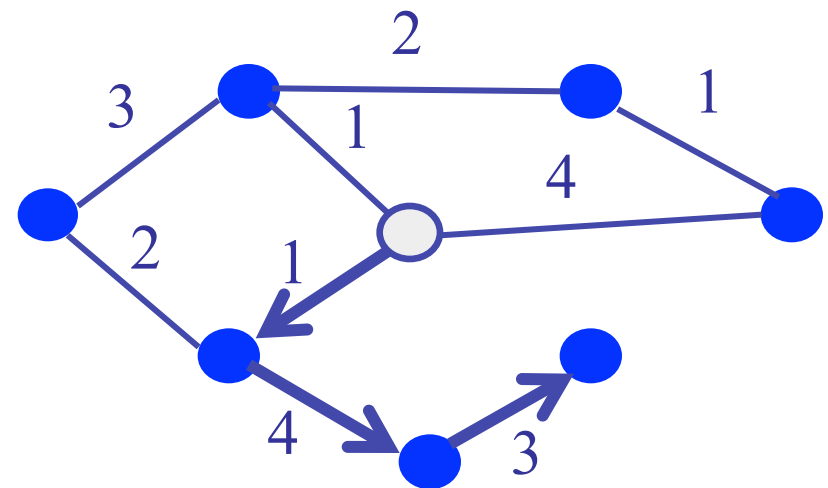
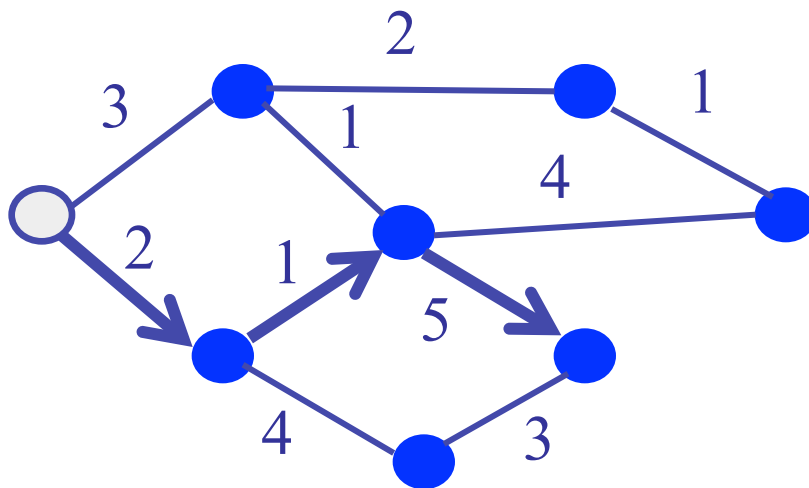
- Detection delay
 - ♦ A node does not detect a failed link immediately
 - ♦ ... and forwards data packets into a **black hole**
 - ♦ Depends on timeout for detecting lost hellos





Transient Disruptions

- Inconsistent link-state database
 - ◆ Some routers know about failure before others
 - ◆ The shortest paths are no longer consistent
 - ◆ Can cause transient forwarding loops





Convergence Delay

- Sources of convergence delay
 - ◆ Detection latency
 - ◆ Flooding of link-state information
 - ◆ Shortest-path computation
 - ◆ Creating the forwarding table
- Performance during convergence period
 - ◆ Lost packets due to blackholes and TTL expiry
 - ◆ Looping packets consuming resources
 - ◆ Out-of-order packets reaching the destination
- Very bad for VoIP, online gaming, and video



Reducing Delay

- Faster detection
 - ◆ Smaller hello timers
 - ◆ Link-layer technologies that can detect failures
- Faster flooding
 - ◆ Flooding immediately
 - ◆ Sending link-state packets with high-priority
- Faster computation
 - ◆ Faster processors on the routers
 - ◆ Incremental Dijkstra's algorithm
- Faster forwarding-table update
 - ◆ Data structures supporting incremental updates



Link State evaluation

- Strengths
 - ◆ Loop free as long as LS database's are consistent
 - » Can have transient routing loops – shouldn't last long
 - ◆ Messages are small
 - ◆ Converges quickly
 - ◆ Guaranteed to converge
- Weaknesses
 - ◆ Must flood data across entire network (scalability?)
 - ◆ Must maintain state for entire topology (database)



Real Link-state Protocols

- OSPF (Open Shortest Path First) and IS-IS
 - ◆ Most widely used intra-domain routing protocols
 - ◆ 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
 - » “Border” router pretends to be directly connected to all routers in an area (answers for them)
 - ◆ Load balancing: Multiple equal cost routes



For next time...

- Midterm