

Lecture 16:

More Intra-domain Routing

CSE 123: Computer Networks

Aaron Schulman





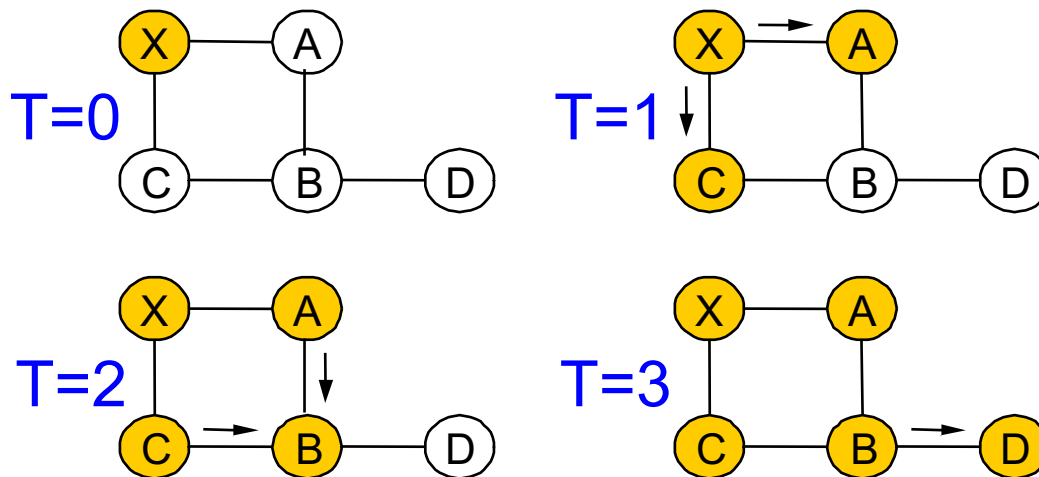
Lecture 16 Overview

- Wrap up discussion of link state routing
 - What can go wrong and how to deal with it
- Intro to Distance vector routing

Remember: Flooding Example



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



- Entire process repeats with LSPs for A, B, C, ...
 - ◆ Actually in runs in parallel



Making Things 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
 - ◆ Put a Time To Live (TTL) timer on each LSP
 - ◆ TTL in every LSP, decremented periodically (second) 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



Route 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

Link State Convergence Delay

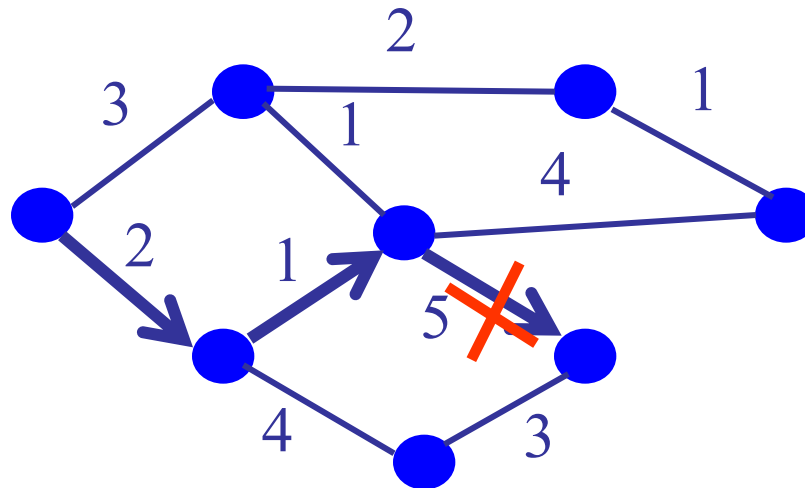


- Sources of convergence delay
 - ◆ Detection latency
 - ◆ Flooding of link-state information
 - ◆ Shortest-path computation
 - ◆ Creating the forwarding table
- Poor performance during convergence period
 - ◆ Lost packets due to black holes and TTL expiry
 - ◆ Looping packets consuming resources
 -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



Transient Disruptions

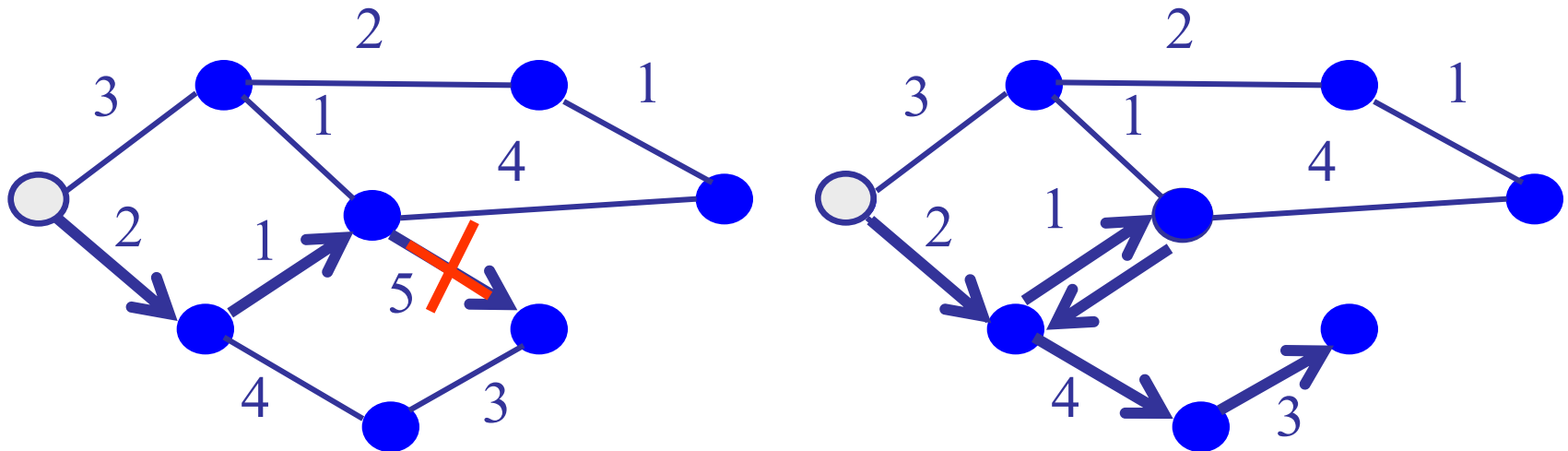
- Detection delay
 - ◆ A router does not detect a failed link immediately
 - ◆ ... and forwards data packets into a **black hole**
 - ◆ Depends on timeout for detecting lost LSPs





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





Reducing Delay

- Faster detection
 - ◆ Smaller LSP 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



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



Link State evaluation

- Strengths
 - ◆ Loop-free as long as LS databases have converged
 - » 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 (large database)



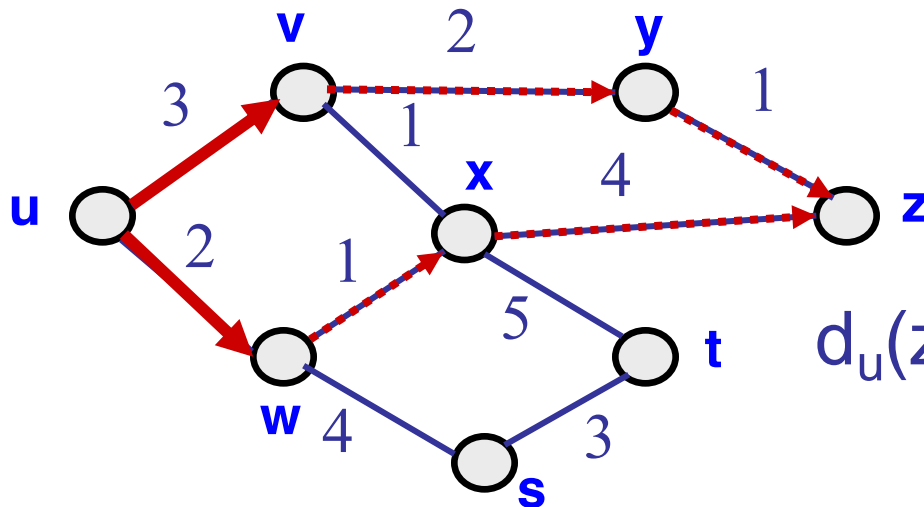
Distance vector algorithm

- Base assumption
 - ◆ Each router knows its **own address** and the cost to reach each of its **directly connected neighbors**
- Bellman-Ford algorithm
 - ◆ Distributed route computation using **only neighbor router info**
 - ◆ **No flooding like in link-state routing**
- Mitigating loops
 - ◆ Split horizon and poison reverse



Bellman-Ford Algorithm

- Define distances at each node X
 - ♦ $d_x(y)$ = cost of least-cost *path* from X to Y
- Update distances based on neighbors
 - ♦ $d_x(y) = \min \{c(x,v) + d_v(y)\}$ over all neighbors V



$$d_u(z) = \min\{c(u,v) + d_v(z), c(u,w) + d_w(z)\}$$



Distance Vector Algorithm

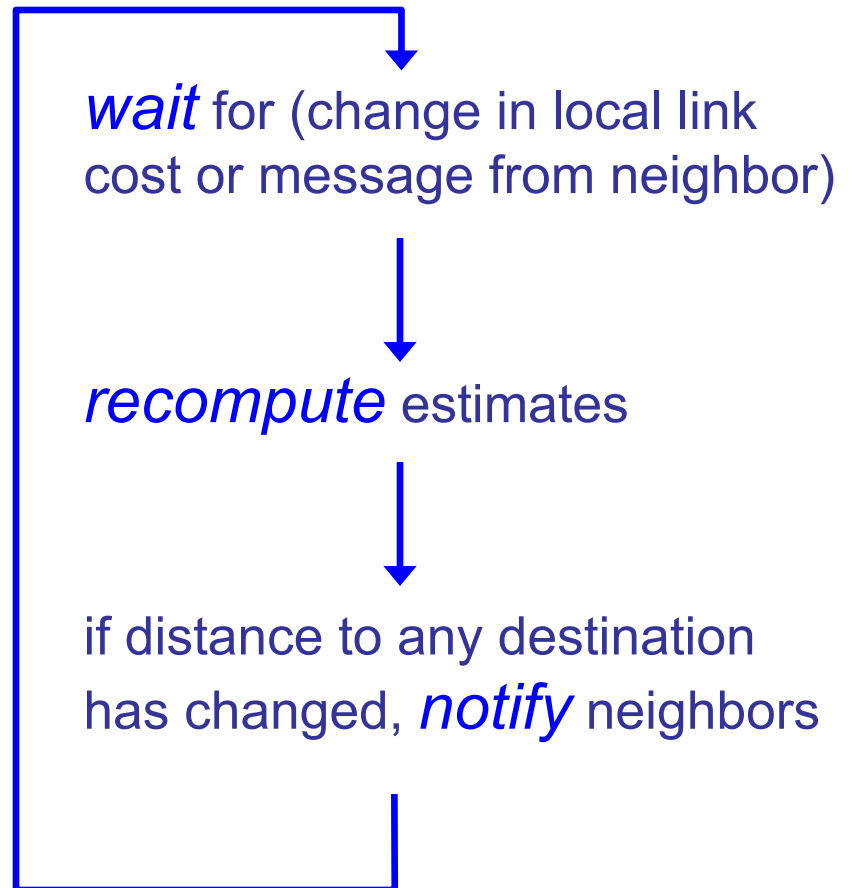
Iterative, asynchronous: each local iteration caused by:

- Local link cost change
- Distance vector update message from neighbor

Distributed:

- Each node notifies neighbors when its DV changes
- Neighbors then notify their neighbors if necessary

Each node:





For next time...

- Project 3 out Tonight
- Read Chapter 4.1 in P&D (internetwork routing)