

Computer Networks

Link State Routing

(§5.2.5, 5.6.6)



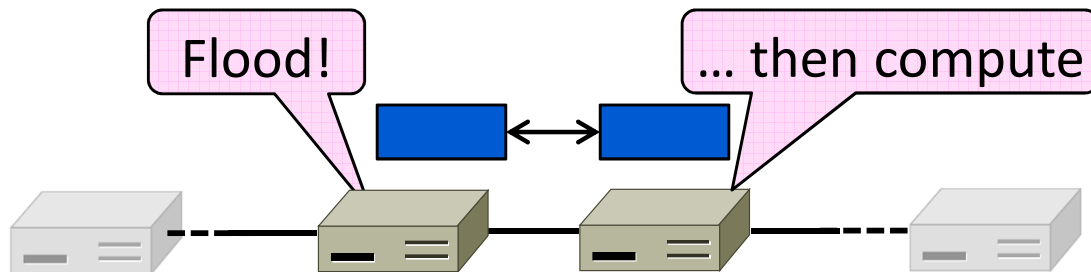
David Wetherall (djw@uw.edu)

Professor of Computer Science & Engineering


UNIVERSITY *of* WASHINGTON

Topic

- How to compute shortest paths in a distributed network
 - The Link-State (LS) approach







Link-State Routing

- One of two approaches to routing
 - Trades more computation than distance vector for better dynamics
-  Widely used in practice
 - Used in Internet/ARPANET from 1979
 - Modern networks use OSPF and IS-IS



Link-State Setting

Nodes compute their forwarding table in the same distributed setting as for distance vector:

1.  Nodes know only the cost to their neighbors; not the topology
2.  Nodes can talk only to their neighbors using messages
3.  All nodes run the same algorithm concurrently
4.  Nodes/links may fail, messages may be lost

Link-State Algorithm

Proceeds in two phases:

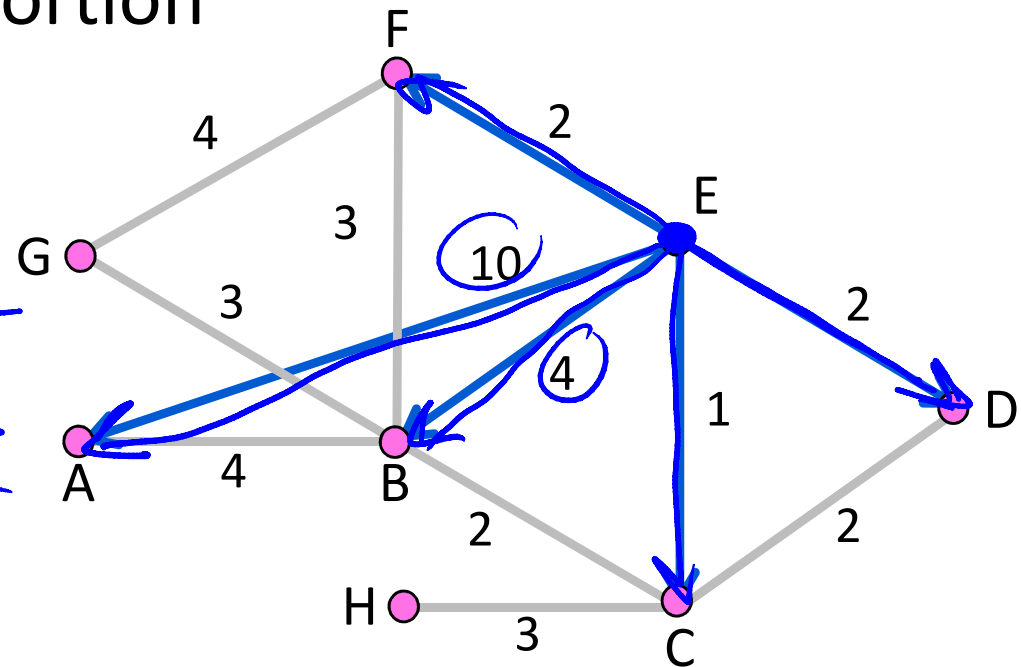
1.  Nodes flood topology in the form of link state packets
 - Each node learns full topology
2.  Each node computes its own forwarding table
 - By running Dijkstra (or equivalent)

Phase 1: Topology Dissemination

- Each node floods link state packet (LSP) that describes their portion of the topology

Node E's LSP
flooded to A, B,
C, D, and F

| Seq. # | |
|--------|----|
| A | 10 |
| B | 4 |
| C | 1 |
| D | 2 |
| F | 2 |

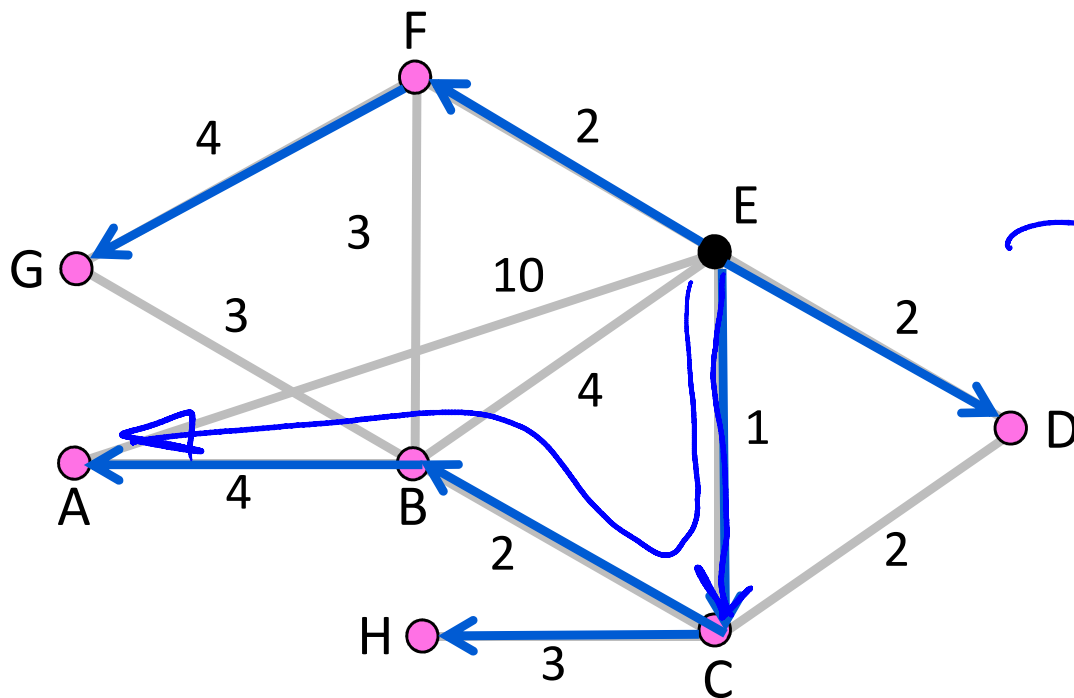


Phase 2: Route Computation

- Each node has full topology
 - By combining all LSPs
- Each node simply runs Dijkstra
 - Some replicated computation, but finds required routes directly
 - Compile forwarding table from sink/source tree
 - That's it folks!

Forwarding Table

Source Tree for E (from Dijkstra)

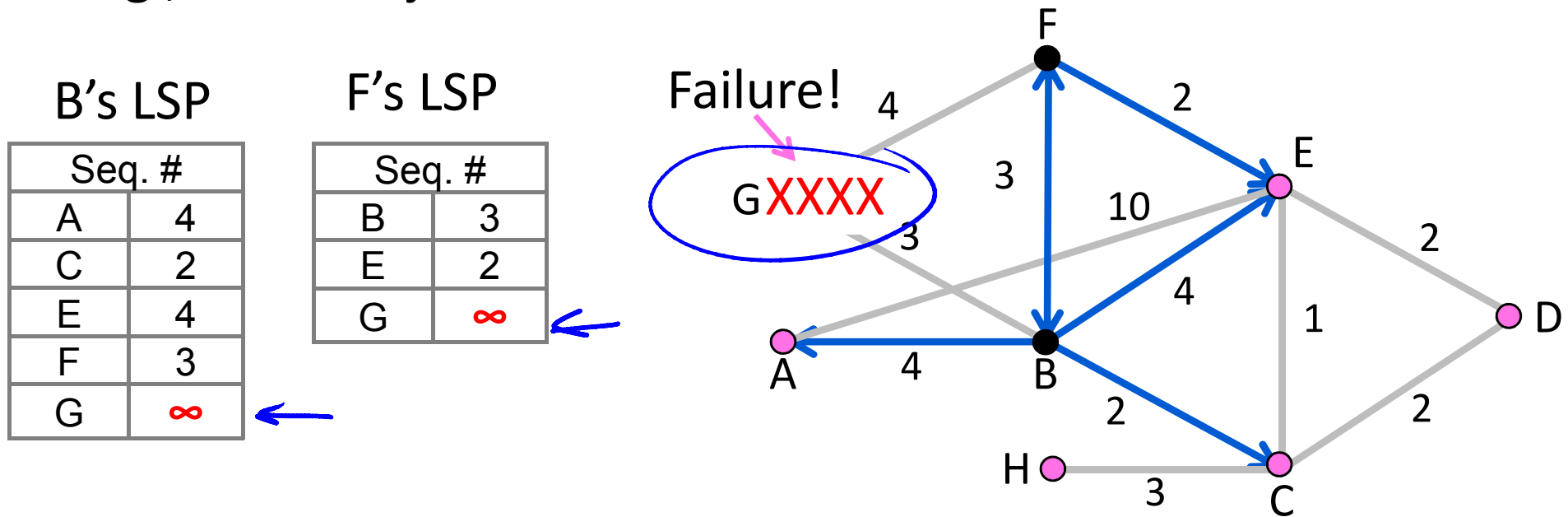


E's Forwarding Table

| To | Next |
|----|------|
| A | C |
| B | C |
| C | C |
| D | D |
| E | -- |
| F | F |
| G | F |
| H | C |

Handling Changes

- On change, flood updated LSPs, and re-compute routes
 - E.g., nodes adjacent to failed link or node initiate



Handling Changes (2)

- Link failure
 - Both nodes notice, send updated LSPs
 - Link is removed from topology
- Node failure
 - All neighbors notice a link has failed
 - Failed node can't update its own LSP
 - But it is OK: all links to node removed

Handling Changes (3)

- Addition of a link or node
 - Add LSP of new node to topology
 - Old LSPs are updated with new link
- Additions are the easy case ...

Link-State Complications

- Things that can go wrong:
 - Seq. number reaches max, or is corrupted
 - Node crashes and loses seq. number
 - Network partitions then heals
- Strategy:
 - Include age on LSPs and forget old information that is not refreshed
- Much of the complexity is due to handling corner cases (as usual!)

DV/LS Comparison

| Goal | Distance Vector | Link-State |
|------------------|-----------------------------|-----------------------------|
| Correctness | Distributed Bellman-Ford | Replicated Dijkstra |
| Efficient paths | Approx. with shortest paths | Approx. with shortest paths |
| Fair paths | Approx. with shortest paths | Approx. with shortest paths |
| Fast convergence | Slow – many exchanges | Fast – flood and compute |
| Scalability | Excellent – storage/compute | Moderate – storage/compute |

IS-IS and OSPF Protocols

- Widely used in large enterprise and ISP networks
 - IS-IS = Intermediate System to Intermediate System
 - OSPF = Open Shortest Path First
- Link-state protocol with many added features
 - E.g., “Areas” for scalability

END

© 2013 D. Wetherall

Slide material from: TANENBAUM, ANDREW S.; WETHERALL, DAVID J., COMPUTER NETWORKS, 5th Edition, © 2011.
Electronically reproduced by permission of Pearson Education, Inc., Upper Saddle River, New Jersey