COMP 790-088
Networked and Distributed Systems

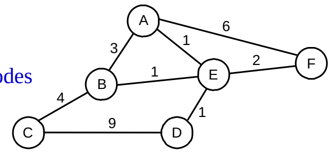
Link State Routing

Jasleen Kaur

September 21, 2009

Intra-domain Routing
Formulation

- Intra-domain routing ⟹ ~ 100 routers
- Given:
  - » **Graph**: where nodes are routers and edges are links
  - » **Cost**: associated with each link
- Find:
  - » **Lowest-cost path** between any two nodes
- Requirements:
  - » Self-healing, traffic-sensitive, scalable



Need dynamic, distributed algorithms!
Two classes: based on "distance-vector" and "link-state"

Link-state Routing
Basic Idea

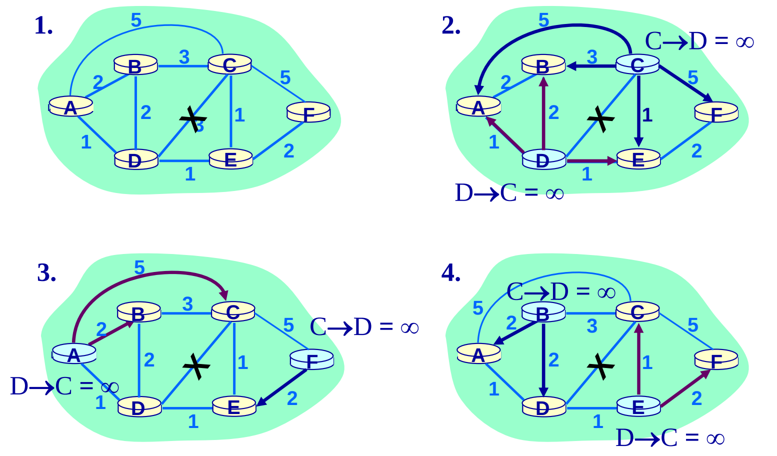
- Speed of convergence is key advantage of link-state routing
- Approach: if each node has complete info about all links, it can:
  - Build complete map of network
  - And compute shortest path to any node
- Two key mechanisms:
  - Reliable dissemination (of complete link-state of the network)
  - Calculation of routes (from the sum of accumulated link-state)

Link State Routing
Reliable Flooding

- On link-cost changes, and periodically, each node creates a link-state packet (LSP) that contains:
  - For enabling route-computation
    - ID of node that created it
    - List of directly-connected neighbors + cost of link to each
  - For ensuring reliability of flooding
    - Sequence number
    - TTL for this packet
- Transmission of LSPs between adjacent routers is made reliable
  - Using ACKs and retransmissions
- When K receives an LSP originated at Y, it stores it if:
  - Has no previous state (or has only smaller sequence number) from Y
    - If it stores, it also forwards to all neighbors (except one who forwarded LSP)

Link State Flooding Algorithm

Example

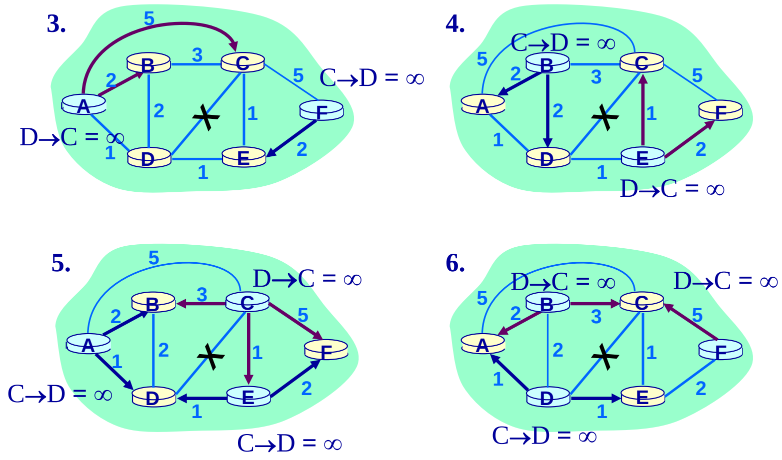


COMP 790-088

5

Link State Flooding Algorithm

Example (Continued)



COMP 790-088

6

Link State Routing

Dijsktra's Algorithm

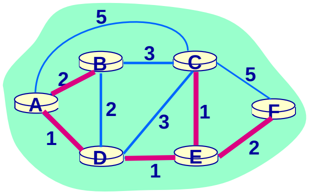
```
Initialization:
  N = {A}
  for all nodes v
    if v adjacent to A
      then D(v) = c(A,v)
      else D(v) = infinity

Loop
   find node w not in N such that D(w) is a minimum
   add node w to N
   update D(v) for all nodes v adjacent to w and not in N:
      D(v) = min( D(v), D(w) + c(w,v) )
    /* new cost to node v is either old cost to v or known
      shortest path cost to w plus cost from w to v  */
until all nodes in N
```

N is the set of nodes to which we have computed the minimum cost path
D(*x*) is the current minimum cost path to *x*
c(*n*,*m*) is the cost of the link from *n* to *m*

Link State Routing

Dijsktra's Algorithm: Example

Step	start N	D(B),p(B)	D(C),p(C)	D(D),p(D)	D(E),p(E)	D(F),p(F)
0	A	2,A	5,A	1,A	infinity	infinity

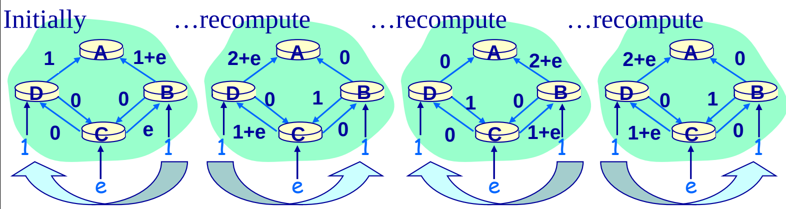


N is the set of nodes to which we have computed the minimum cost path
D(*x*) is the current minimum cost path to *x*
p(*x*) is the predecessor of *x* on the current minimum cost path to *x*

Link State Routing

Oscillating Routes

- "Route oscillations" are possible in link state algorithms
- Let the link cost equal the amount of carried traffic
  - » Assume the link cost is updated as traffic changes



Node	Path to A
D	A
C	B
B	A

Node	Path to A
D	A
C	D
B	C

Node	Path to A
D	C
C	B
B	A

Node	Path to A
D	A
C	D
B	C

Least Cost Path Computations

Link-state vs. Distance-vector Algorithms

- Message complexity:
  - » LS: With *N* nodes, *E* links, *O*(*N*x*E*) messages sent for flooding
  - » DV: Exchange between neighbors only (may trigger further exchanges)
    - ❖ Due to reliable flooding, LS considered to generate less traffic
- Speed of Convergence:
  - » LS: O(N^2) algorithm and *O*(*N*x*E*) messages
    - ❖ May have oscillations depending on choice of metric
  - » DV: Convergence time varies
    - ❖ Routing loops possible
    - ❖ Count-to-infinity problem
- Robustness: what happens if there are failures?
  - » LS: Node can advertise incorrect *link* cost

    Each node computes only its *own* table
  - » DV: Node can advertise incorrect *path* cost

    Each node's table used by others
    - ❖ *Errors propagate through network*