CS419: Computer Networks

Lecture 5, Part 3: Feb 21, 2005

Internet Routing:

Distance-vector (DV) and Path-vector (PV) scaling

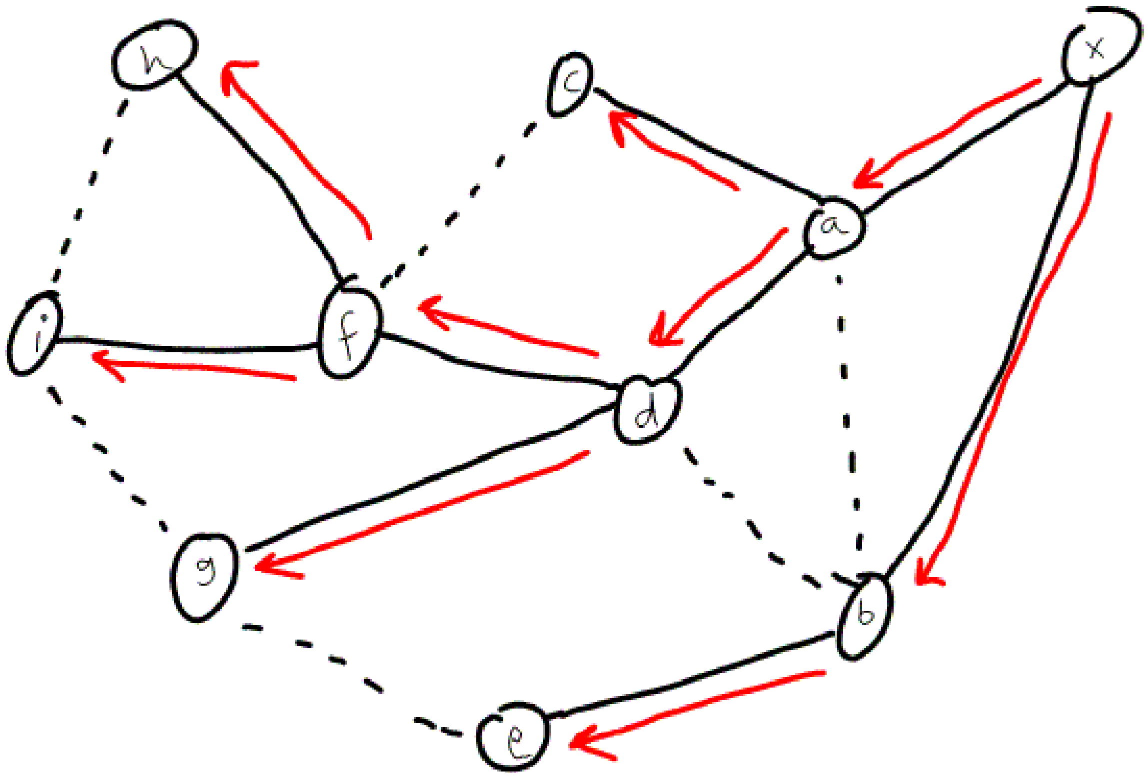
CS419

- DV scales as the number of destinations N
- Path-vector scales approx as N(1/2D), where D is the network diameter
  - Because paths are one average ½ the diameter
  - A single link change can still result in large updates
    - (all destinations for which there is a new path)
  - So overhead can vary depending on situation (unpredictable)

Distance-path overhead example

CS419

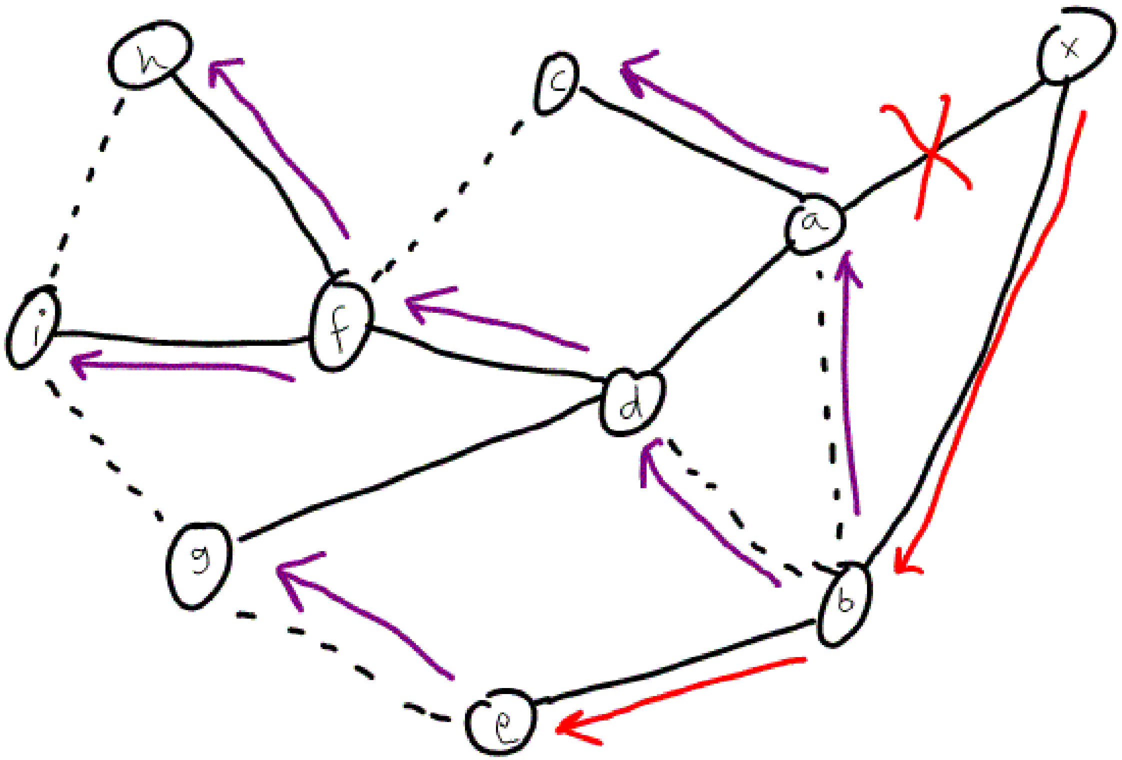
Node x doesn't "see" dashed links



What if link x–a goes down?

One link change can result in updates about many destinations

x has to "learn" all of these paths



Updates about 7 destinations!

Distance-vector problems

- As we saw, distance-vector (DV) routing algorithms, while simple, suffer from slow convergence
- Path-vector (PV) fixes most of this, but still has some unpredictability
- Link State pre-dates PV, is less flexible but has very fast convergence and predictable overheads
  - In wide use: OSPF

Link-State approach

CS419

- Like PV, LS works by providing more explicit information about the state of the network
  - In fact, *complete* information about the state of the network!
- Every node knows about every link
  - Internally contains a “map” of the complete network
- From this map, each node computes its next hops

LS RIB

CS419

	N_1	N_2	N_3
D_1	5	1	7
D_2	2	9	3
D_3	11	2	6
D_4	3	3	4

DV RIB

link	from	to	cost
L_1	D_1	D_3	6
L_2	D_1	D_4	12
L_3	D_2	D_4	8
L_4	D_2	D_3	2
⋮	⋮	⋮	⋮

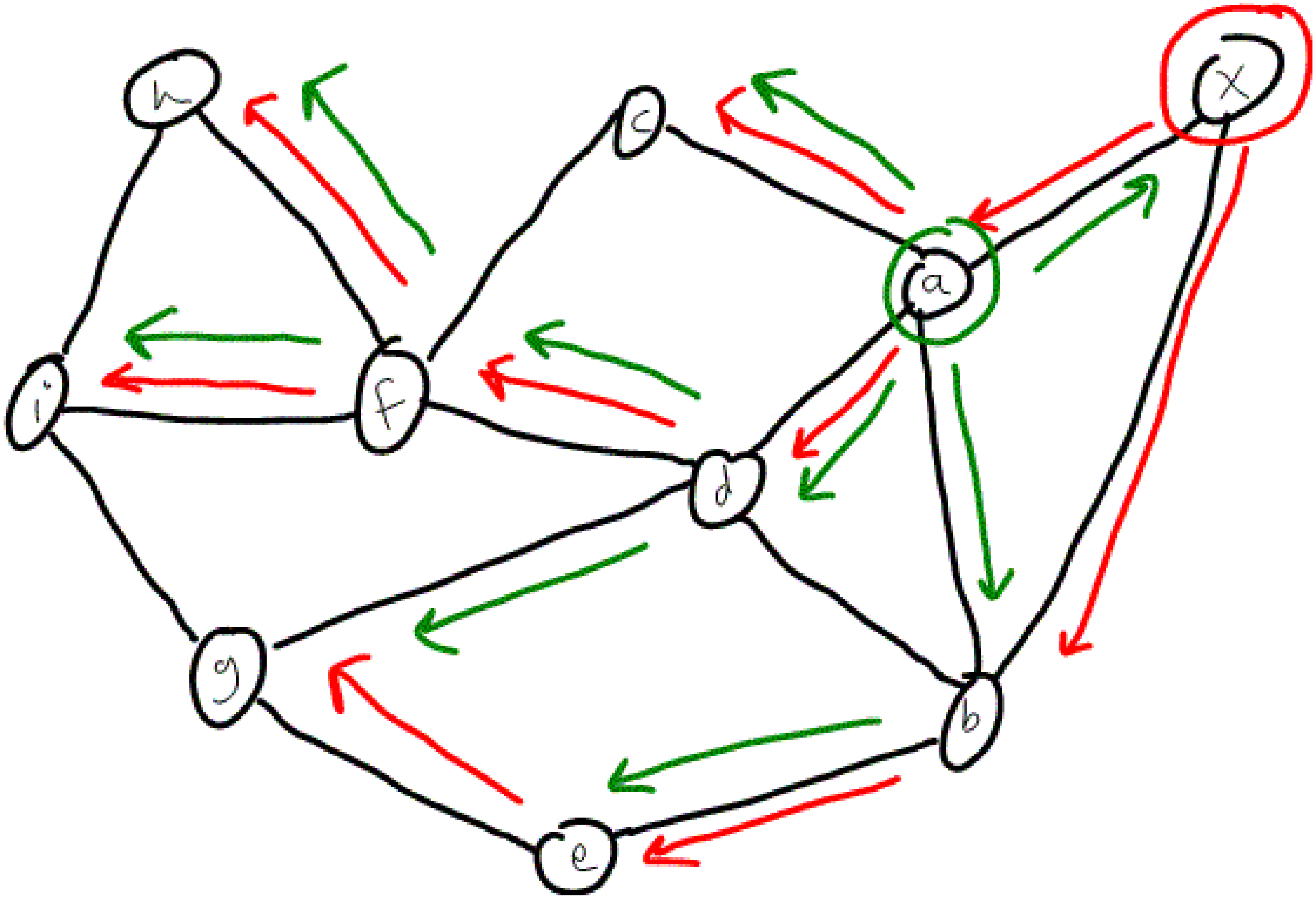
LS RIB

Link State Operation

- Each node floods the status of all of its links to every other node
  - This creates the RIB
- Each node generates its FIB by running a shortest-path spanning tree algorithm with itself as the root

Shortest paths overlap

CS419



Flooding

- Each node periodically floods a *Link State Update* (LSU) to all nodes
  - Or immediately if a link changed
- LSU contains:
  - List of all the node’s links and costs
  - A sequence number (to determine which LSU is the most recent
  - A hop count

Flooding algorithm (simplified)

CS419

- Each node stores the latest LSU seq num (SNs) received for all nodes
- When a node originates an LSU, it increments the SN
- When an LSU is received, if the received SNr is “newer than” SNs, then:
  - Record information in LSU
  - Send LSU to all neighbors
  - Set SNs = SNr
- Otherwise, ignore the LSU

Sequence number initialization and wrap around

CS419

- This is far trickier than you'd think…
- Imagine an 8-bit unsigned sequence number (0 <= SN <= 0xff)
- Say SNs = 0xf0, and SNr = 0x0f
- Is the received LSU newer or older than the stored one?

Sequence number initialization and wrap around

CS419

- When SN reaches max value, it will wrap around to 0
  - Thus, at some point, SN=0 is “newer than” SN=0xFFFF
- SNs = 0xf0, and SNr = 0x0f
- Probably SNr is newer, but you can’t be sure
  - Maybe there is some error that caused a router to send an old SN

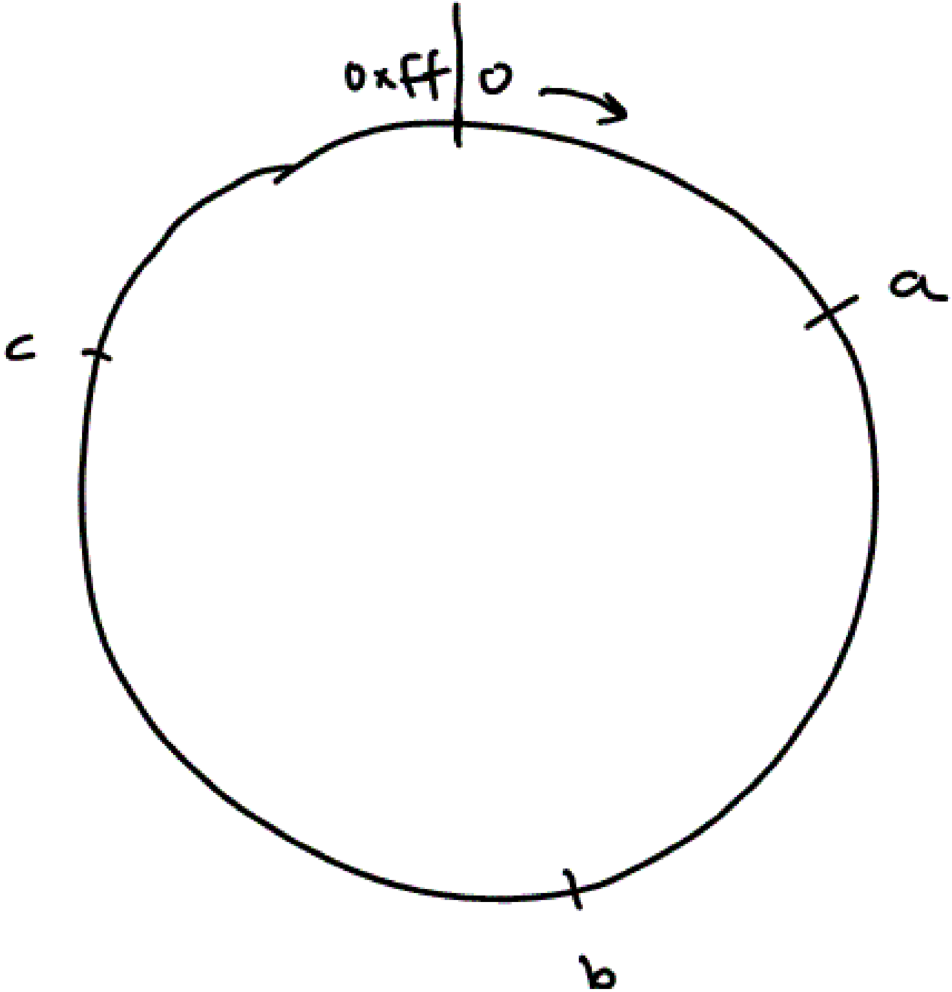
Approach number 1: circular seq num space

CS419

- To compare two numbers a and b
- Divide seq space in half at a
- If b is in clockwise half, then b is newer, else a is newer
- Router must save its own SN in non-volatile memory (disk)
- When router restarts, initialize own SN to latest saved value + 1

Circular seq num space

CS419



b newer than a

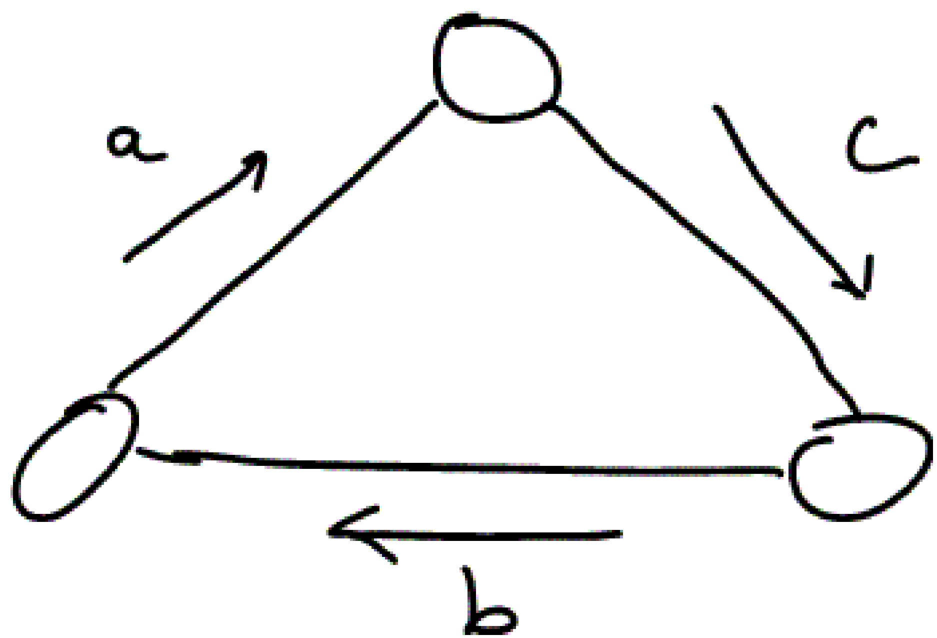
c newer than b

a newer than c

One problem with circular seq num space

CS419

- These SLU's would flood forever…
  - Or until the hop count expired
  - This apparently happened in the ARPANET



Approach number 2: Huge linear seq num space

CS419

- 64-bit sequence number space, no wrap-around
- Store own SN in non-volatile memory, init from most recent SN + 1
- When max value reached (2^{64}-1), crash!!!
- At 100 LSU/sec, takes 6 billion years to hit max (i.e. never crash)

Problem with huge linear seq num space

CS419

- Try explaining it to customers…
- Non-volatile storage must be very reliable
  - Disk, for instance, is not that reliable
- If the SN is lost, router must be restarted as a different router (i.e. with a different identity)

Approach number 3: lollipop shaped seq num space

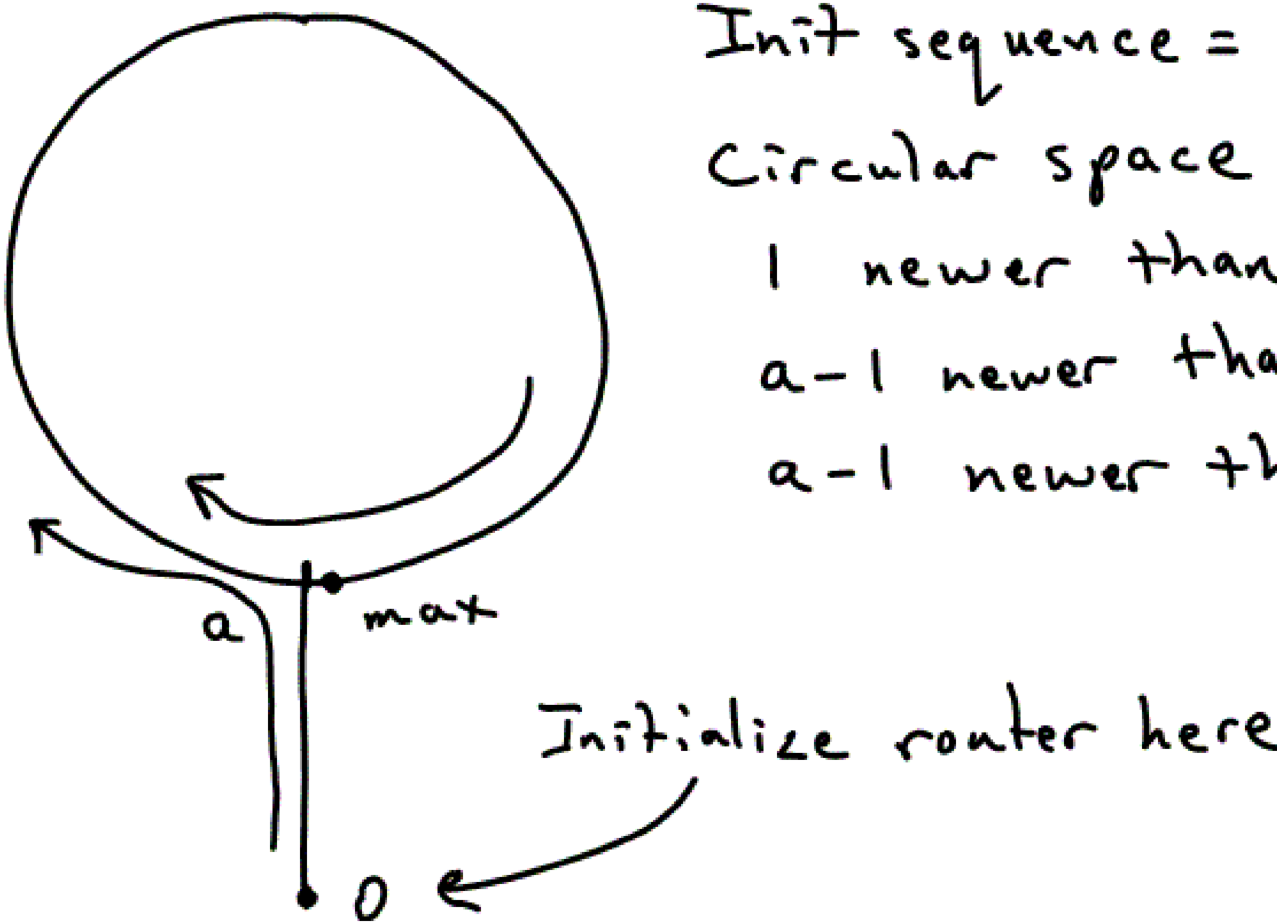
Init sequence = 0 → a-1

Circular space a → max

1 newer than 0

a-1 newer than a-2

a-1 newer than a



Problems with lollipop shaped seq num space

CS419

- Same a < b < c < a problem
  - Though this is mitigated by hop count in LSU
- If router restarts before SN >= a, then no new LSUs will be recognized until new SN reaches old high-water
  - But routers with bugs may often restart shortly after startup
- This approach in V1 of OSPF

Approach 4: Linear space with LSU flush

- Used by OSPF V2
- Extra bit in LSU used to indicate that last LSU should be flushed
- When router restarts, it flushes max SN, then sends initial LSU with SN=0
  - Likewise, if SN wraps, flush max SN before wrap
- Problem would occur if flush not received by all nodes
  - But OSPF flood is quite reliable (LSUs are ACK'd)

Shortest path calculation

CS419

- After any change in the network, the shortest path algorithm is run on the “graph” to calculate the next hops for the FIB
- Attributed to Dijkstra
- All routers must run exactly the same algorithm
  - So that they calculate consistent shortest paths

Shortest path algorithm (1/2)

CS419

- Maintain 2 lists, confirmed and tentative
  - Each entry has <dest, cost, nexthop>
- To initialize, add self to confirmed
- In each round of the algorithm:
  - One dest is moved from tentative to confirmed
  - Zero or more dests are moved into tentative

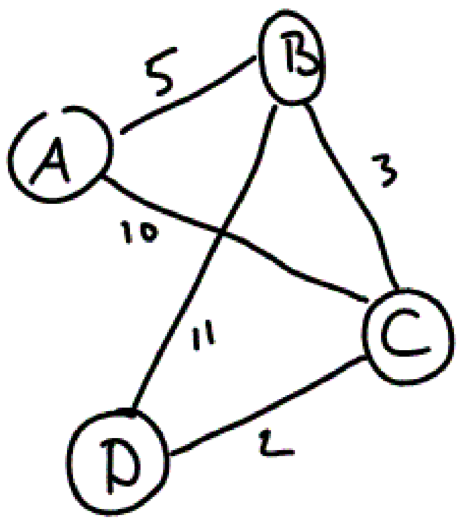
Shortest path algorithm (2/2)

CS419

- *next* = node just moved into confirmed
  - Calculate costs to all of *next*'s neighbors (as next_cost + link_cost)
    - Add neighbor to tentative if not there
    - Change entry in tentative if new cost is lower
- Move node with lowest cost from tentative to confirmed
- Repeat until tentative is empty

Example

CS419



(dest, cost, nexthop)

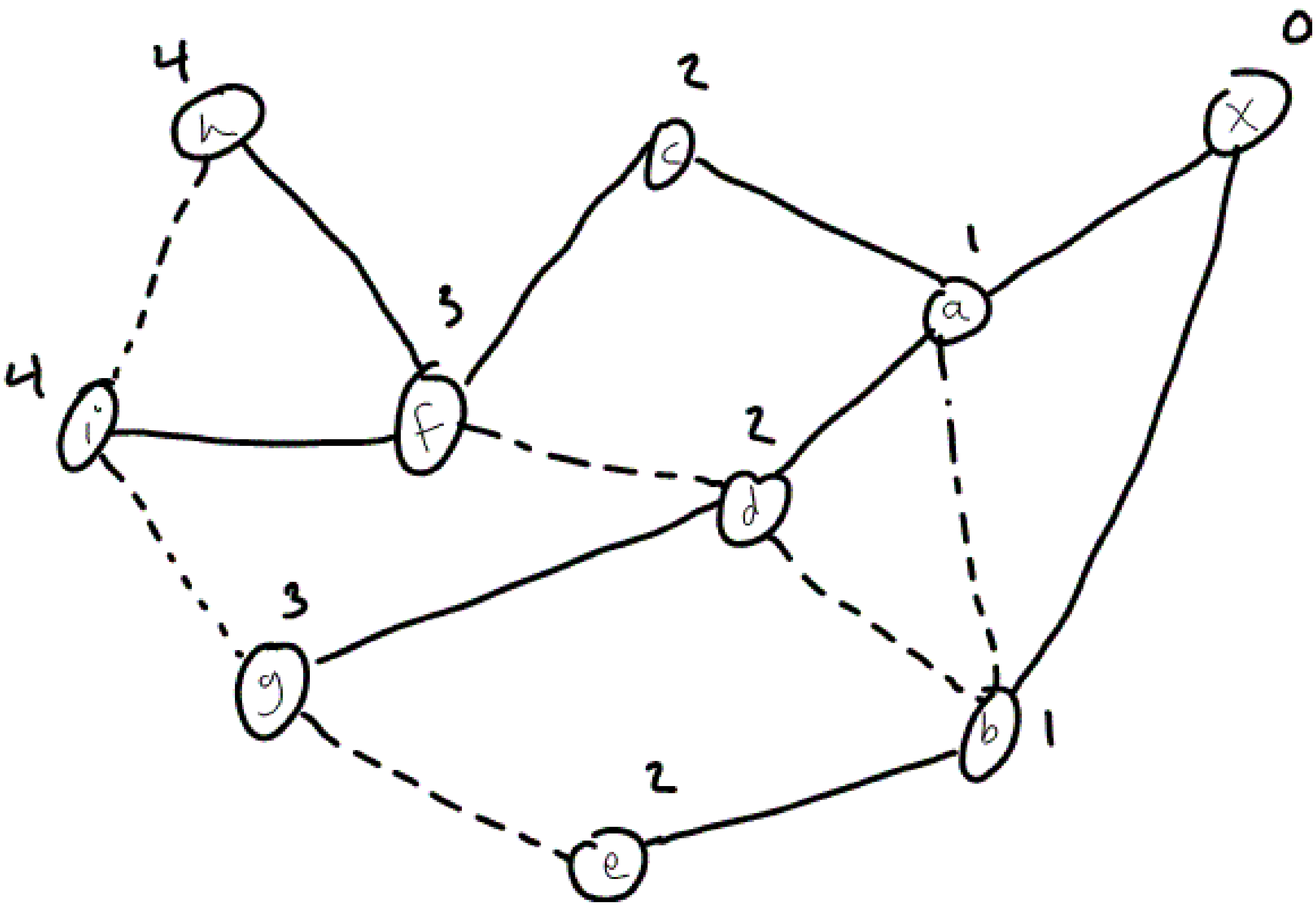
confirmed	tentative
(D, 0, -)	

Shortest path algorithm optimizations

CS419

- Finding the lowest-cost node in the tentative list is expensive
  - Maintain bins for different ranges of cost
  - Only need to search lowest-cost non-empty bin
- Maintain full tree (as predecessor nodes)
  - If non-tree link increases, do nothing
  - In other cases, can pre-populate confirmed and tentative lists

Example



Routing update packet priority

CS419

- Routing updates should have higher priority than data packets
  - So that they get through during congested periods
- But routing updates should be rate limited
  - So that an erroneous flood of updates doesn't starve the network
  - Nodes rate limit their neighbors as well as themselves