E6998-02: Internet Routing

Lecture 8
Link-State Routing

John Ioannidis

AT&T Labs – Research

ji+ir@cs.columbia.edu

Copyright © 2002 by John Ioannidis. All Rights Reserved.

Announcements

Lectures 1-5, 7-8 are available.

There will be no lecture on 10/3 (Inauguration).

Homework 3 will be assigned on 10/1.

- Relax.
- Think about your projects.

Link-State Routing

- Each router starts by knowing:
  - Its attached networks.
  - Its neighbors.
  - Cost to get to its neighbors.
- Each router creates a Link State Advertisement (LSA)
  - Also called a Link State Packet (LSP).
  - List of neighbors and cost to get to each one of them.
- Each router transmits its LSA to all other routers.
  - How?
- Each router receives LSAs from all other routers.
  - Puts together all the LSAs to form a complete graph.
- Each router can now compute routes to each destination.

Meet the Neighbors

- How does a router find who its neighbors are?
  - By manual configuration.
  - Automatically.
- Some variant of “My name is Bob and I’m a router.”
  - “hello” packet.
- On point-to-point links, there is only one neighbor.
  - No need for retransmissions if there is some other indication that the link was broken.
- On shared links, periodically multicast (or broadcast) the hello packet.
  - Not all physical neighbors need to become neighbors in the LS protocol sense.
  - We call this “establishing adjacencies”.

A New LSA Is Constructed:

- Periodically.
- When a new neighbor (new link) shows up.
- When a neighbor (link) goes away.
- When the cost of the link to a neighbor changes.

Disseminating LSAs

- How do we send information to routers that we don’t know how to reach?
  - This was not an issue in DV because we were only sending vectors to our immediate neighbors.
- A: By flooding.
  - Each LSA received is transmitted to all links but the one it came from.
- How to avoid exponential growth of packets?
  - Store each LSA (obviously).
  - If you get an LSA that you’ve already stored, ignore it and don’t propagate.
  - But it’s not that simple.

Getting the Correct LSA

- How do we know that the LSA we got was the one most recently sent out?
  - Packets take multiple paths.
  - Arrive out of order.
- Timestamps?
- Non-global timestamps can cause problems:
  - A corrupted packet appearing to be from the far future would cause valid LSAs to be ignored.
- Global timestamps could provide sanity checks:
  - If LSA from the distant future or past, just ignore.
  - How do we pick the sanity interval?
  - Require synchronized clocks.
    - Harder problem than LSA distribution.

Maybe Sequence Numbers?

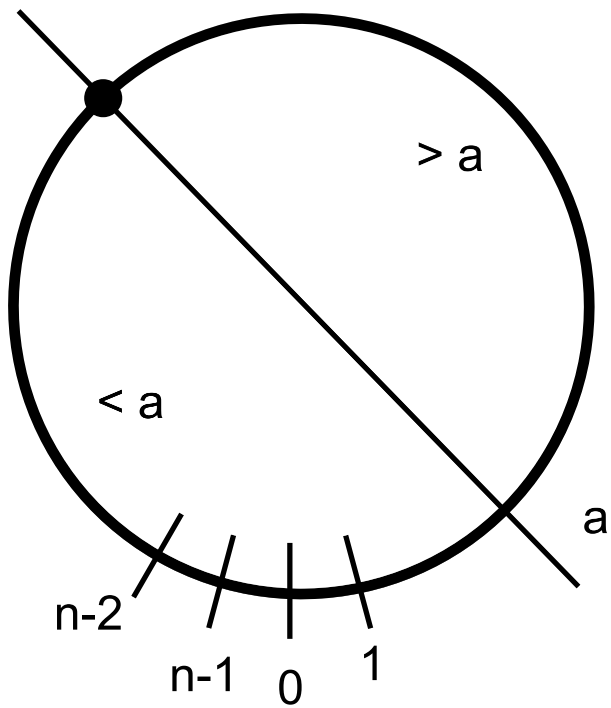
- Sequence numbers by themselves are somewhat equivalent to non-global timestamps.
- A large field guarantees uniqueness.
  - but suffers from the corruption problem.
- A small field can quickly wrap around.
  - What then?
  - Need a way to compare SNs in the presence of wrap-around.

Circular Sequence Number Space

- Space of size ***n*** (***n* = 2*^k*^**).
- Given two numbers ***a*** and ***b***, ***a*** is less than ***b*** if:

  $|a - b| \leq n/2$ and $a < b$, or

  $|a - b| > n/2$ and $a > b$.
- Pictorially:



Problems with Sequence Numbers

- This doesn’t solve the problem either:
- If a router crashes and starts at 0, it will start sending unusable packets again.
- Somewhat reduces, but does not eliminate the problem of corrupted packets.

Add an Age Field

- Use an age field in the LSA in addition to the SN.
- Routers decrement this field at a fixed rate.
- While age > 0, only LSAs with increasing SNs are accepted.
- When age reaches zero, an LSA is accepted regardless of its SN.

- Does this solve the problem?

ARPANET, ca. 1980

- LSAs contain source, s/n, age, list of neighbors.
- When router generates an LSA:
  - Sets the s/n to 1 more (mod n) than the previous LSA it had generated.
  - Sets age to max value.
- When router receives an LSA (not its own!):
  - Accepts it if s/n > stored s/n for the same source.
  - Accepts it always if age of stored LSA is zero.
  - Only propagates LSAs with non-zero age (of course).
- If a router does not hear back an LSA that it generated within 100ms, it assumes it was lost and retransmits.

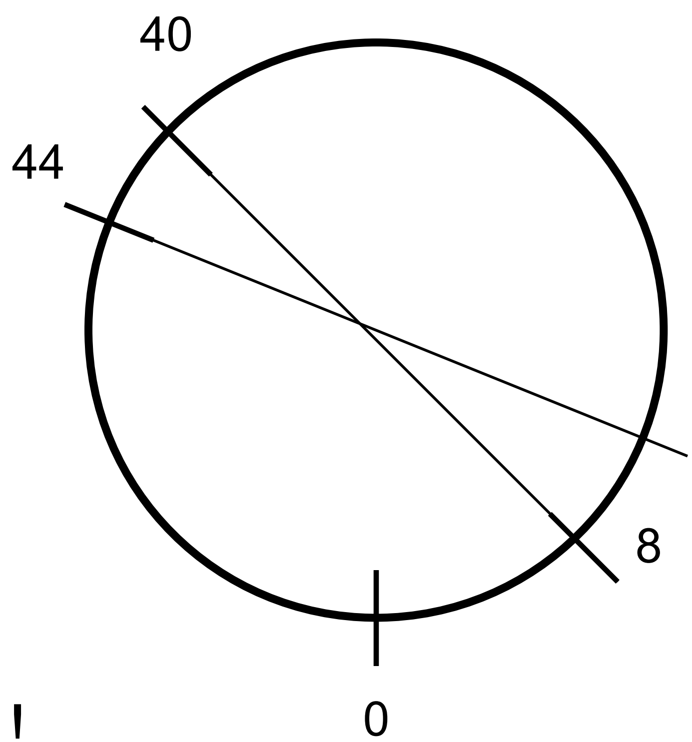
ARPANET, cont'd

- Sequence number is a 6 bit field.
- Age starts at 56 seconds, decrements every 8 seconds:
  - 3 bit field (56 = 7x8).
- Routers wait 90 seconds at startup (let everything age out).
- Routers generate a new LSA within 60 seconds of the previous one.

- Does it work?

ARPANET, October 27th, 1980

- ARPANET stops working.
- Three different LSAs from the same source are being continuously flooded:



- 8 < 40 < 44 < 8 !

No Way to Recover!

- LSAs arriving in the right order.
- LSAs being retransmitted in the same order.
- LSAs have no time to age out, since they are being replaced by “newer” LSAs.

- How did it happen?
  - Bit corruption: 44d=101100b, 40d=101000b, 8d=001000b!
- How did we recover?
  - Take down offending IMP.
  - Patch all IMPs to ignore (and not propagate) offending LSAs.
  - Fix offending IMP, bring back up.
  - Repatch all IMPs to accept all LSAs.

Requirements for LS Protocols

- LSAs must be distributed
  - correctly (unchanged), and
  - completely (to all routers).
- Otherwise different routers will have different maps of the network
  - and will compute different routes,
  - Possibly resulting in routing loops.
- LSA distribution must not cause unbounded creation of LSA packets.
  - Similar to TTL notion in IP.

- But as we just saw, these were not enough.

Requirements for LS Protocols, cont'd

- Self-stabilization:
  - Recover from corrupted packets.
  - Recover from defective equipment.
  - Recover from malicious attacks.
  - Do so in a reasonable amount of time.
- Efficiency:
  - Do not generate too many packets.
  - Do not consume too many router resources.
- Responsiveness:
  - Do not wait for a long time before you can start routing packets.

Improved LSA Distribution

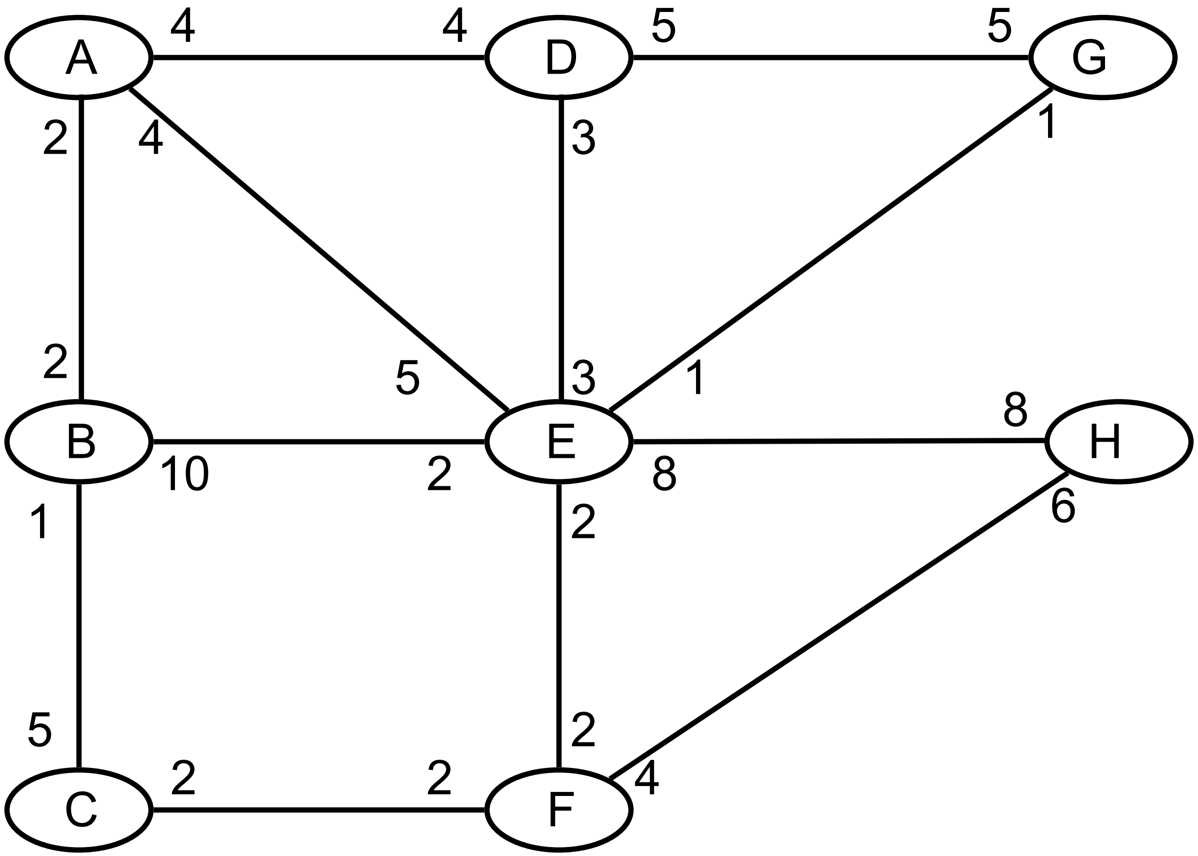
- Sequence number is a linear space.
  - No circular comparisons.
  - But has to be large enough.
- Age is set by the originating router (~1h).
  - Decremented by each router propagating it.
  - Further decremented as it sits in memory.
  - IS-IS decrements; OSPF increments.
- A received LSA is not immediately retransmitted.
  - Sits in per-link queues.
  - A new LSA arriving before an older one has made it out overwrites the older one.
  - Queues are scanned round-robin and LSAs are sent out one link at a time.

Improved LSA Distribution, cont'd

- LSAs should get acknowledged.
  - Various ways of doing that, somewhat different per protocol.
- LSAs that have not been refreshed in some time should get flushed.
  - Again, this varies between protocols.

The Link State Database

- Consider this network (Doyle, p174) :



Each Router Advertises

- Its adjacent neighbors:
  - (router ID, neighbor ID, cost of link)
- Its directly attached stub (no neighbors) networks:
  - (router ID, network prefix, cost of link)

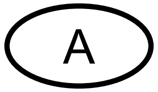
- E.g., E advertises:
  - (E, A, 5), (E, D, 3), (E, G, 1), (E, H, 8), (E, F, 2), (E, B, 2).
- By fitting the LSAs (“jigsaw puzzle routing”), each router forms the same topological database for the network.

Dijkstra’s SPF Algorithm

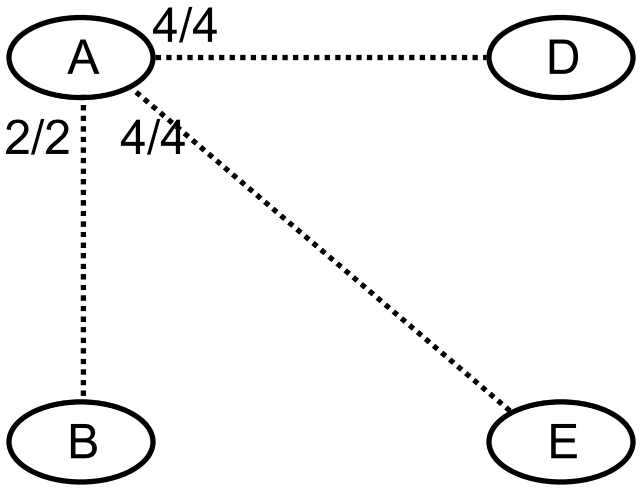
- (All routing algorithms strive to compute the shortest path!)
- Review this from your W4231 notes!
- The Tree Database.
  - Branches (links) definitely assigned to the tree.
  - When finished, this is the SPT.
- The Candidate Database.
  - Branches from which the next branch to go to Tree will be selected.
- The Link State Database.
  - Remaining (rejected or not considered) branches.
- Two sets of nodes:
  - Connected by branches in the Tree database.
  - Rest.

DSPF for Routers

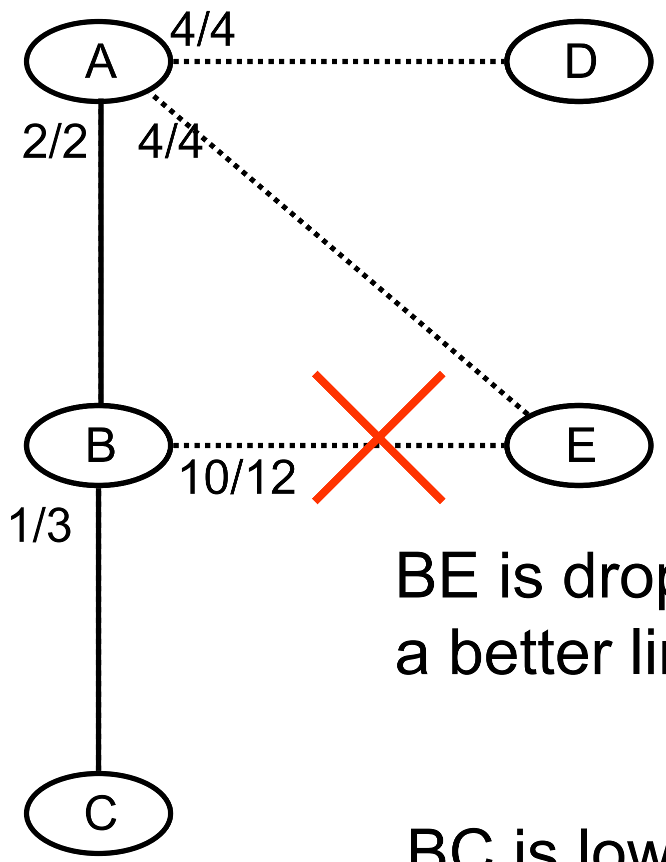
- From A's point of view.
- Initialize Tree by placing A as root.



- Links to all of A's neighbors are added to Candidate.
- Cost to root is computed.



- Lowest-cost link is added to tree.

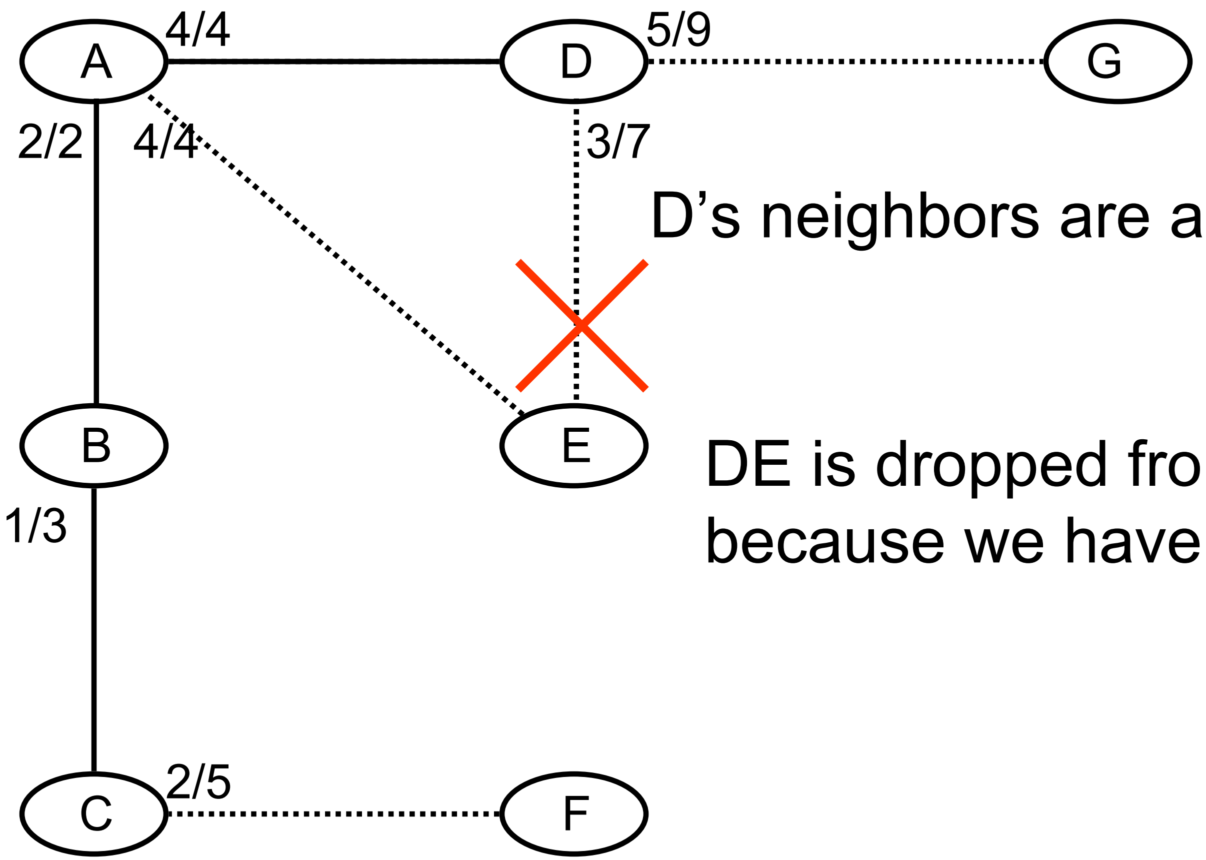


BE is dropped from candidate because we have a better link to E (AE)

BC is lowest cost in Candidate, and is added to tree

- C’s neighbors are added to Candidate.

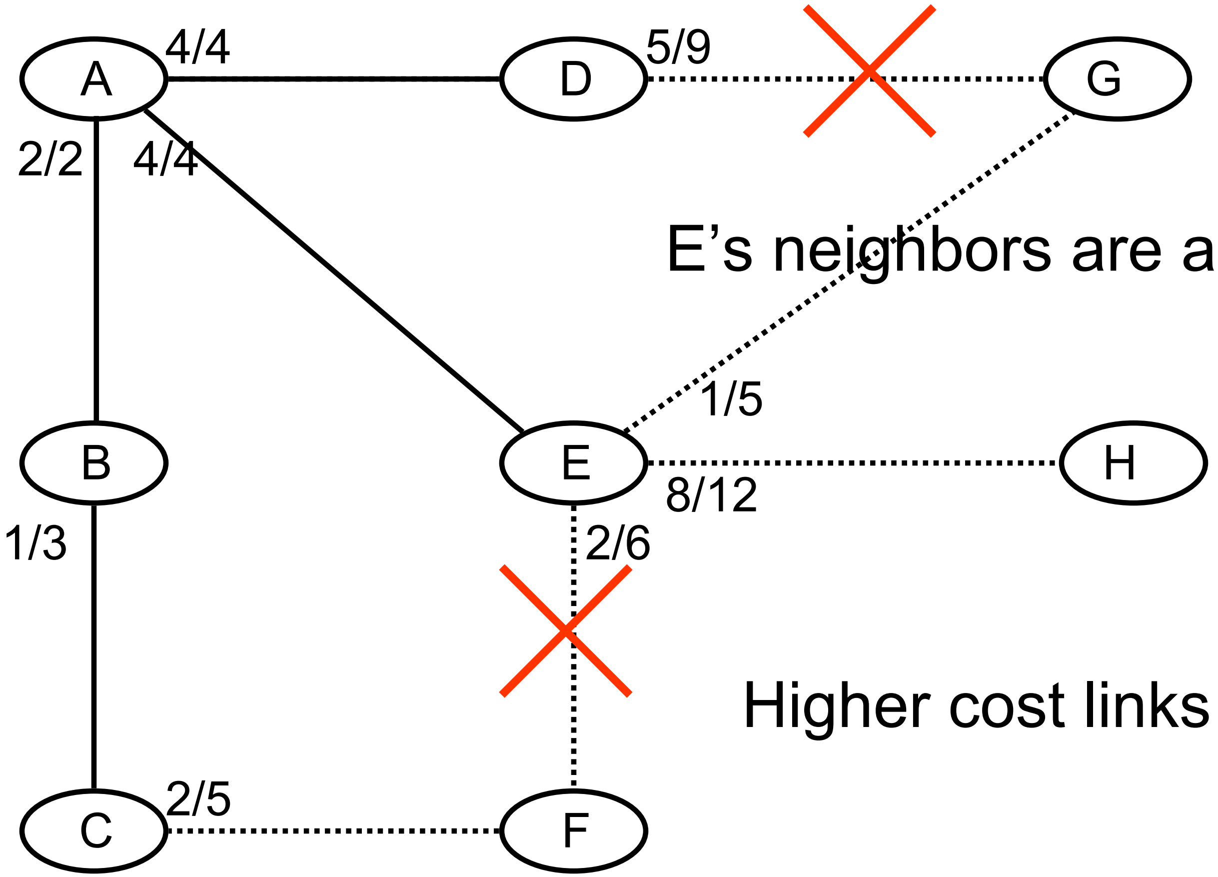
AD and AE are equal cost to A;
Pick AD, add to Tree



D’s neighbors are added to Candidates

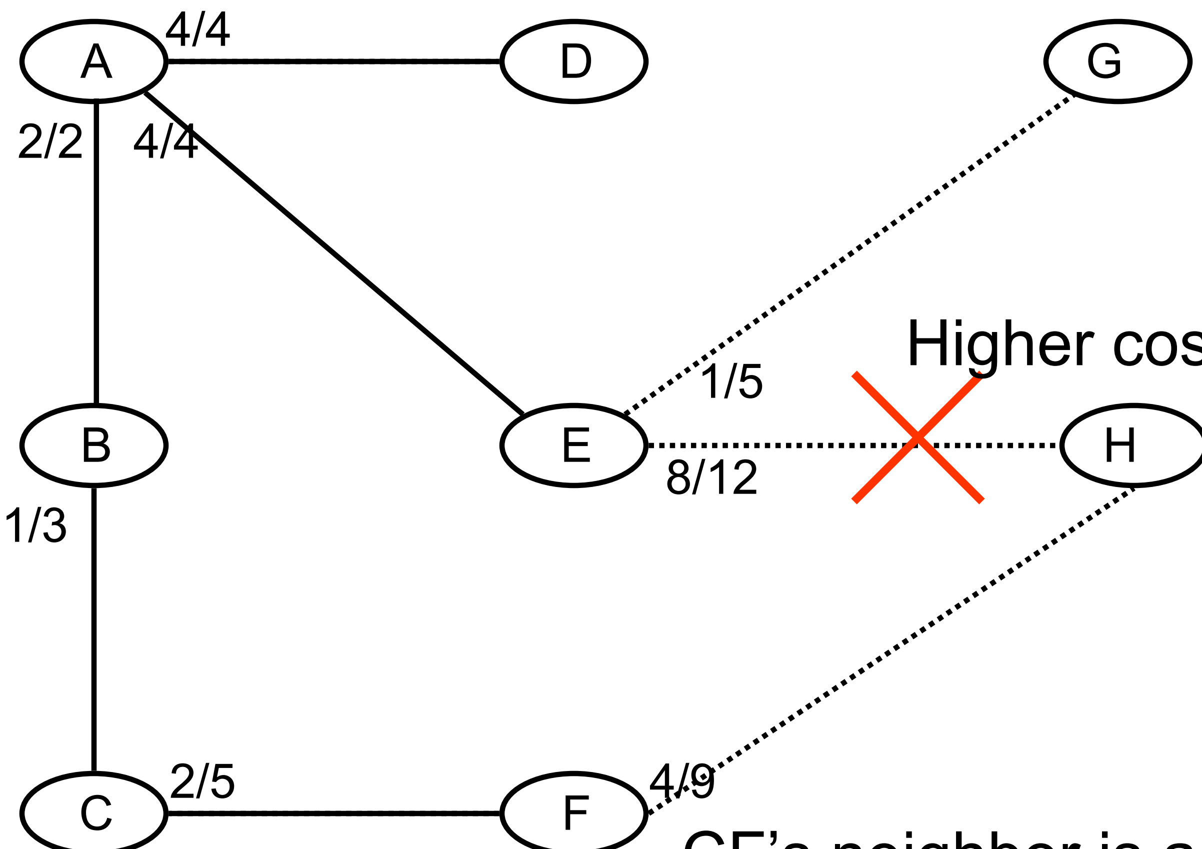
DE is dropped from candidate
because we have a better link to E

- AE is lowest cost, added to Tree



E’s neighbors are added to Candidates

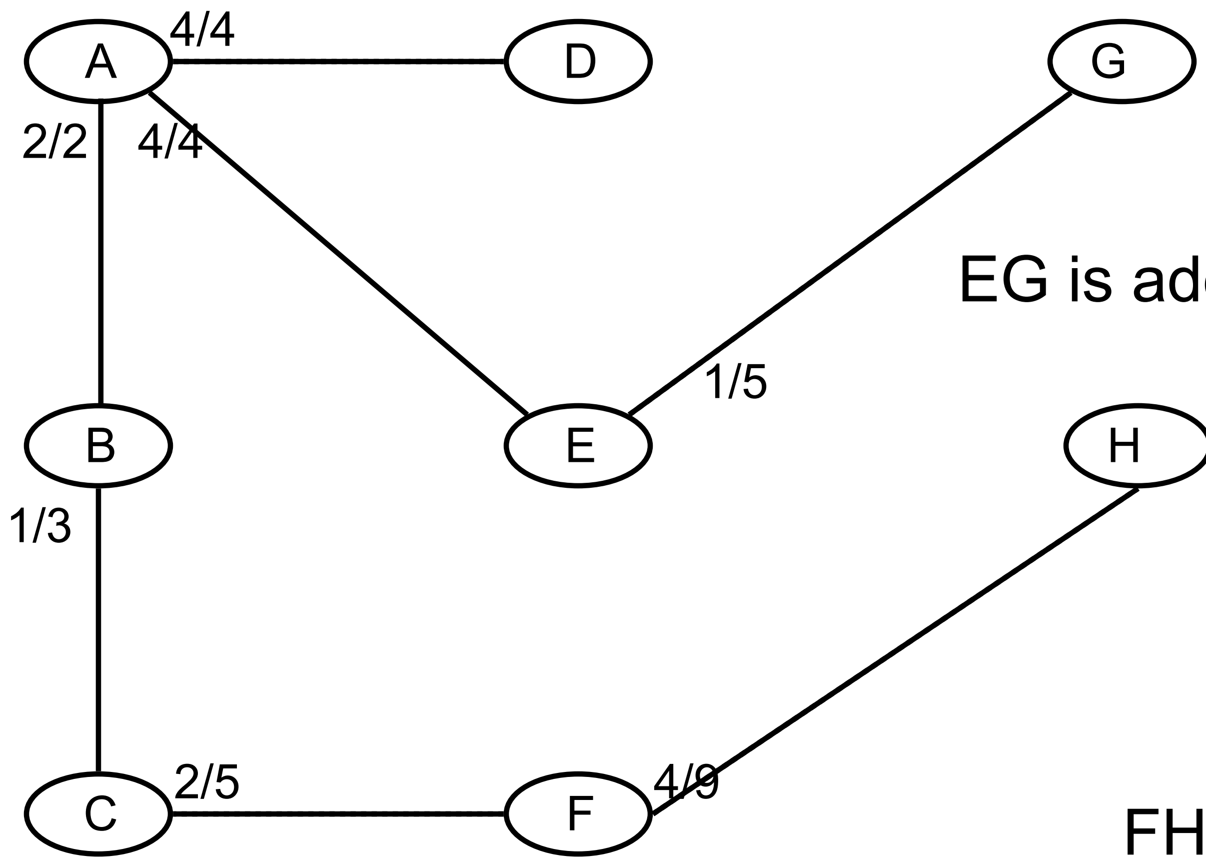
Higher cost links are dropped.



Higher cost links are dropped.

CF's neighbor is added to Candidates

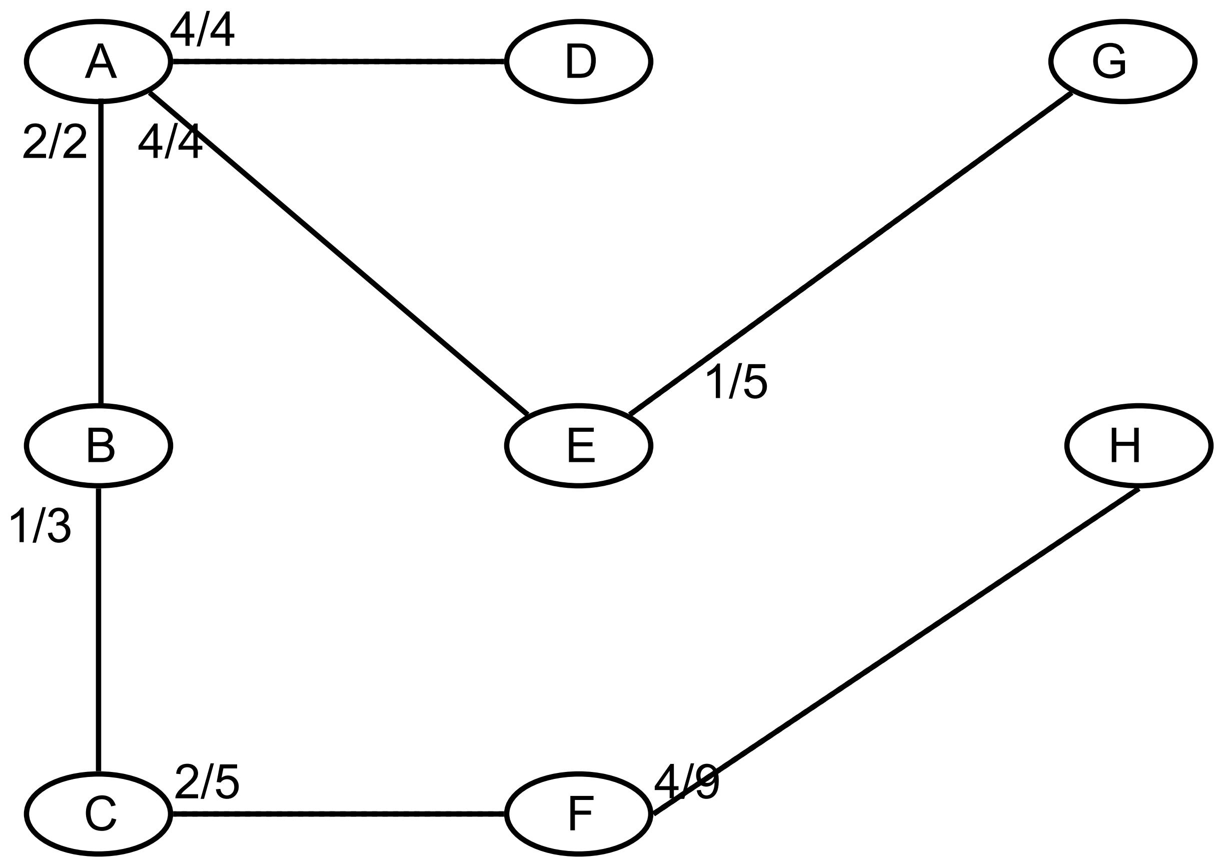
CF is added to the tree



EG is added to the tree

FH is added to the tree

The Shortest Path Tree, from A

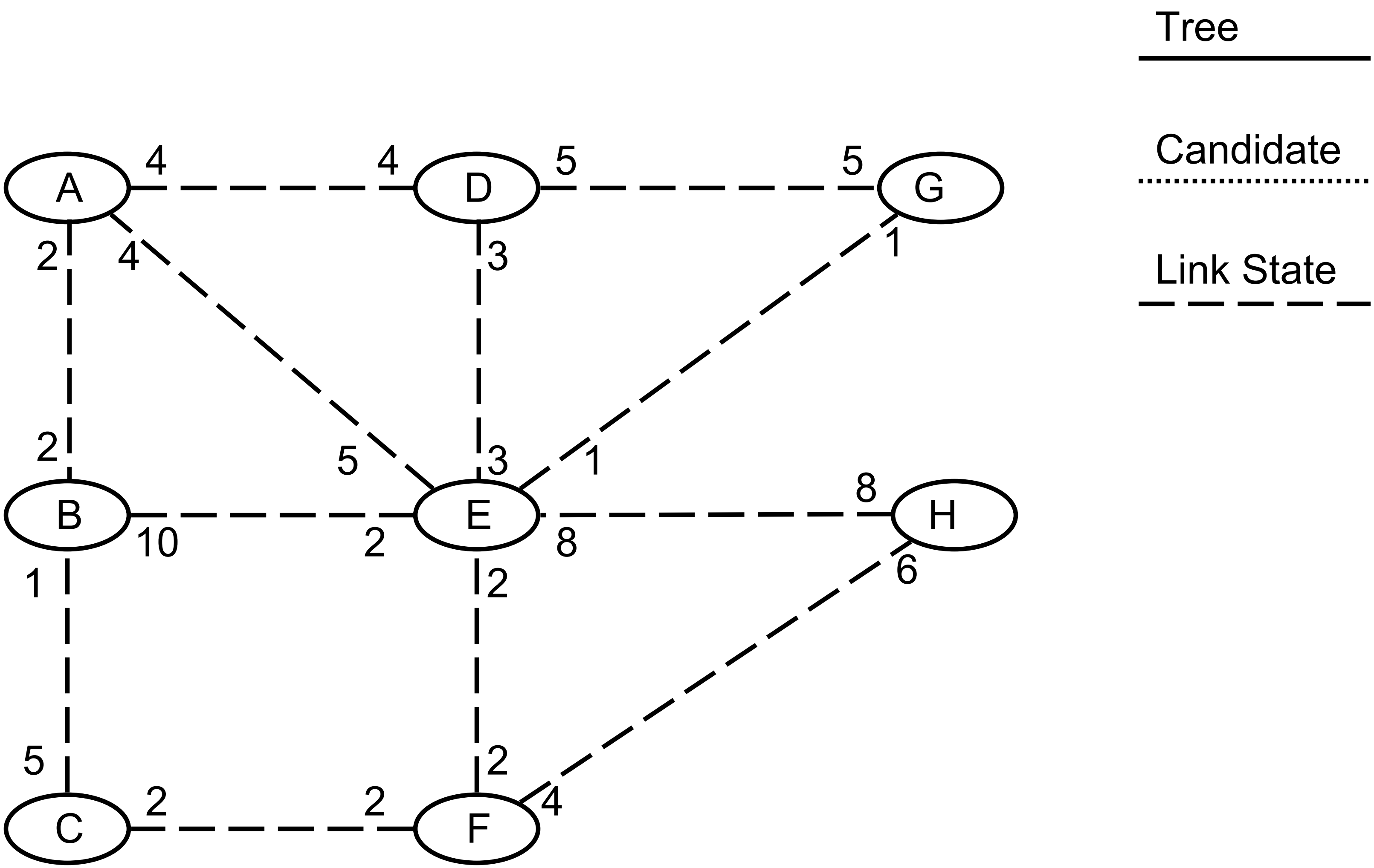


A’s forwarding table

Destination	Next hop
A	self
B	B
C	B
D	D
E	E
F	B
G	E
H	B

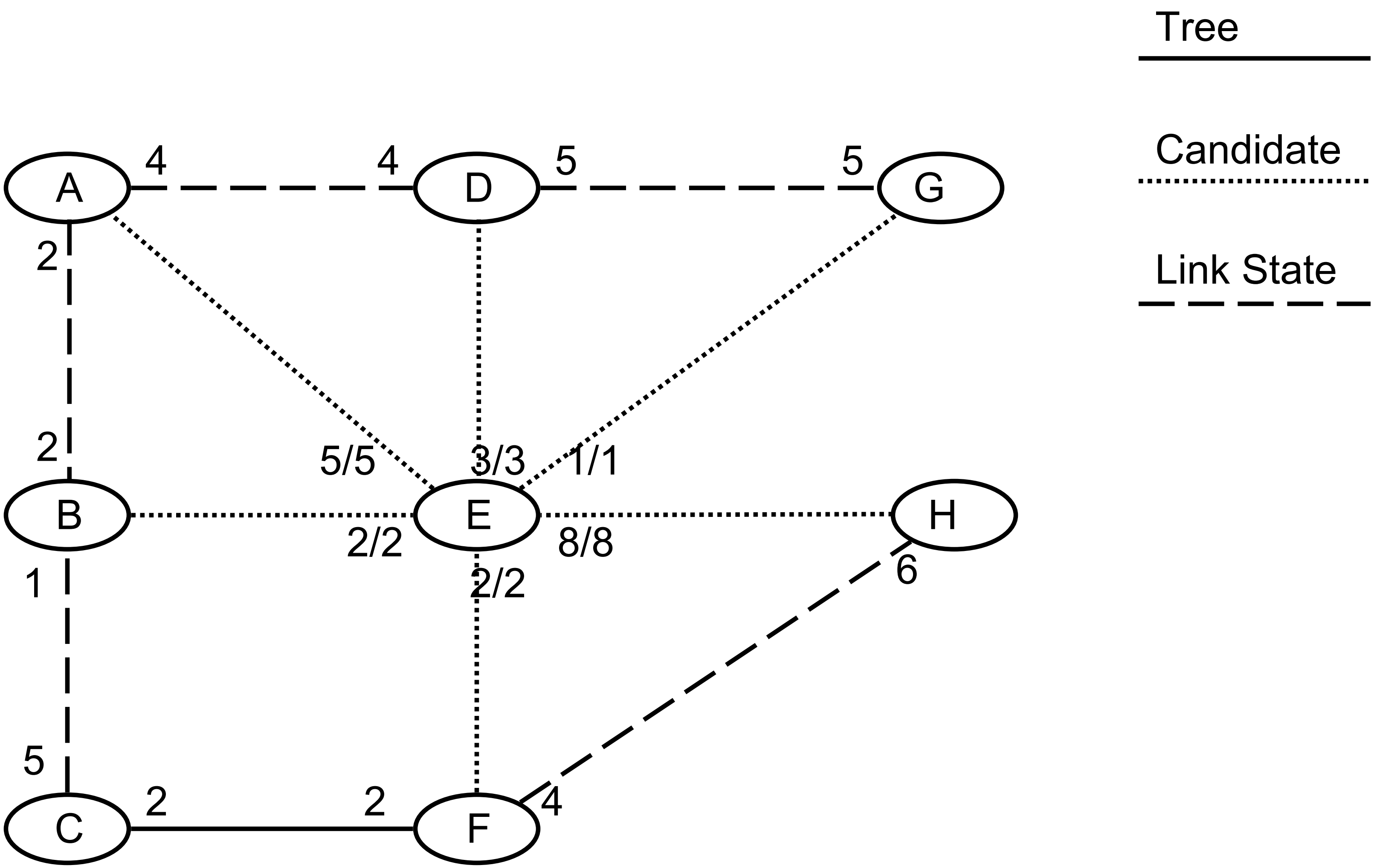
- To send a packet to G, hand it to E.

From E’s perspective, step 0



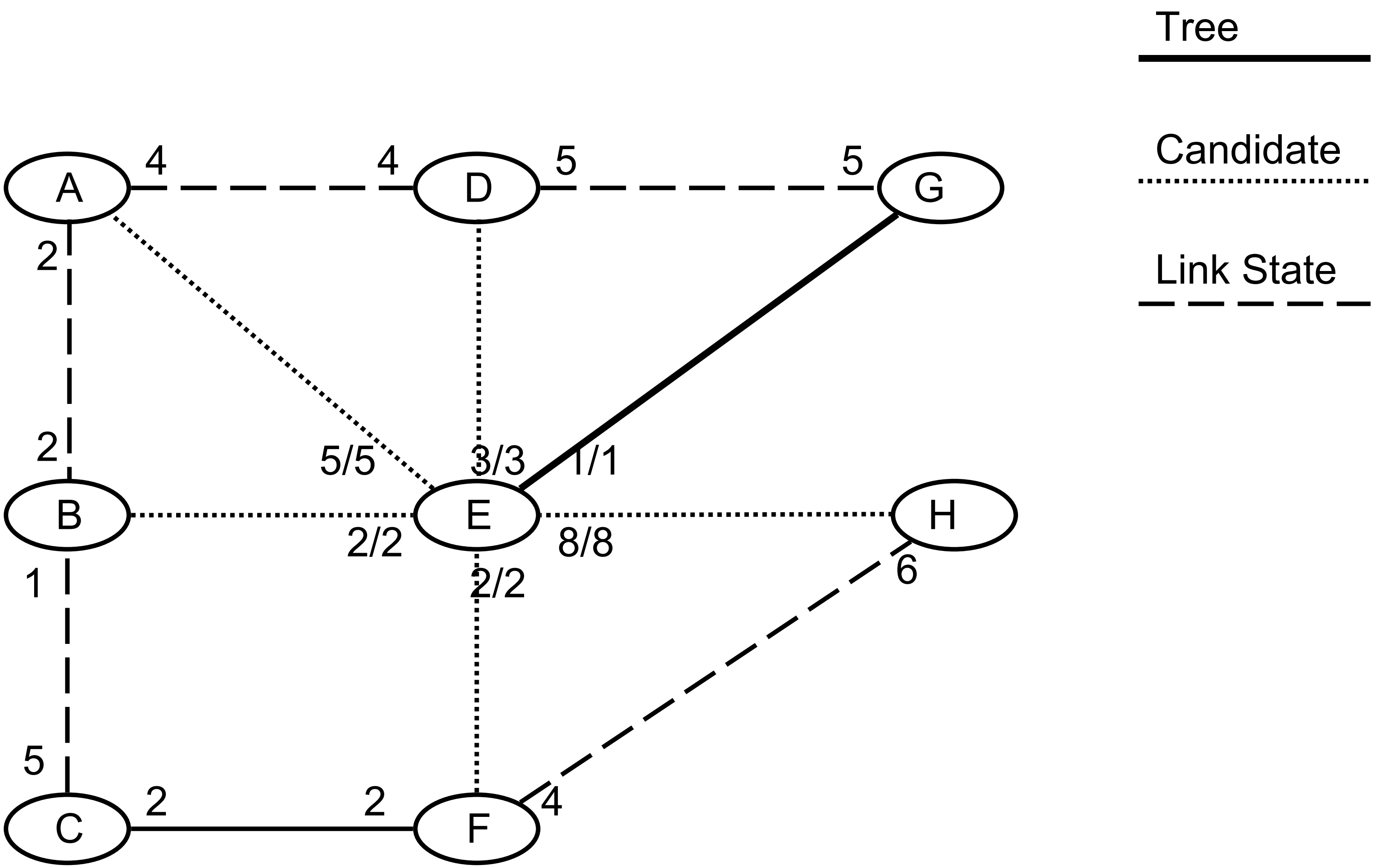
All links start in the LS database

From E’s perspective, steps 1 & 2



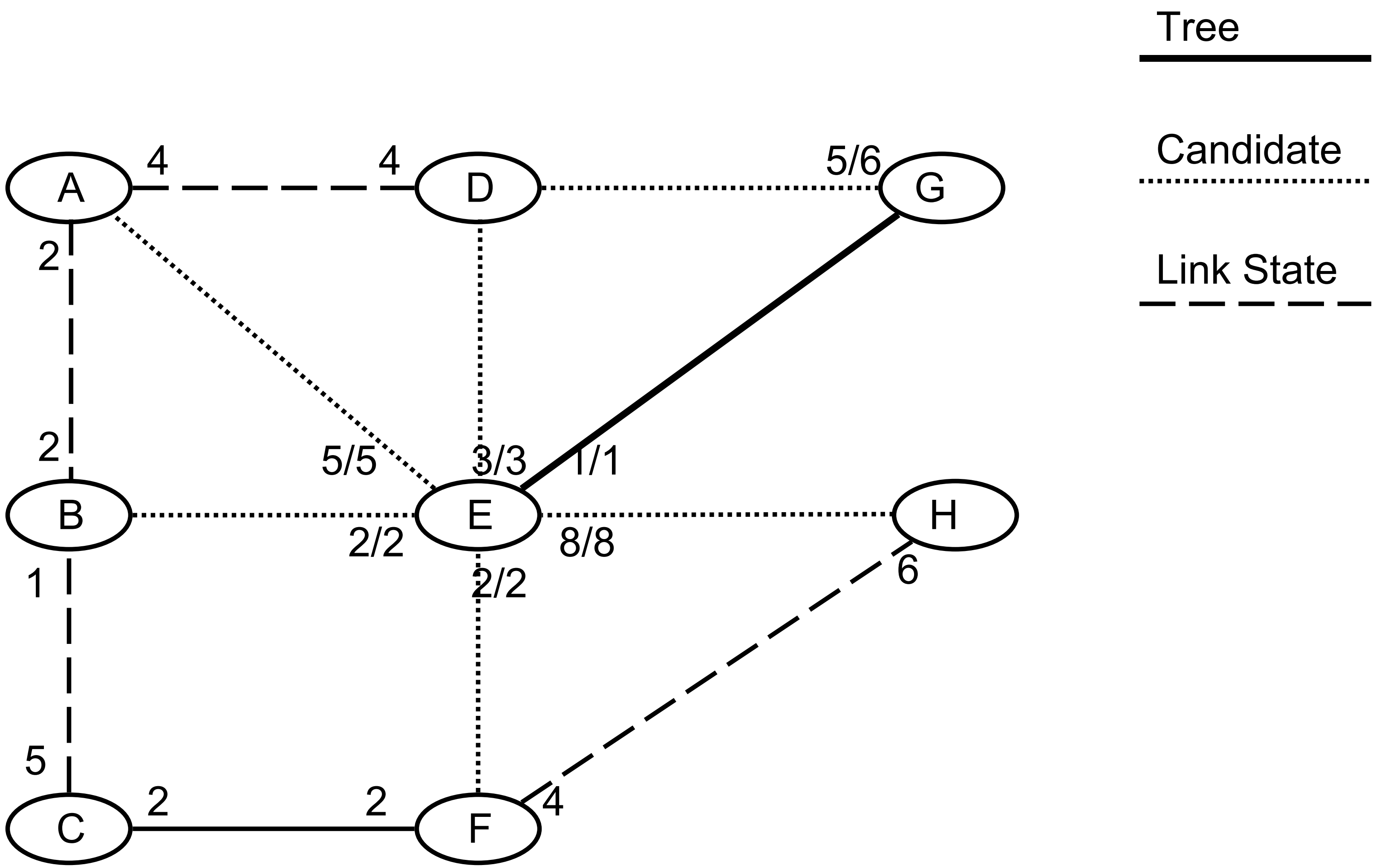
E’s neighbors become candidates; root costs computed.

From E’s perspective, step 3



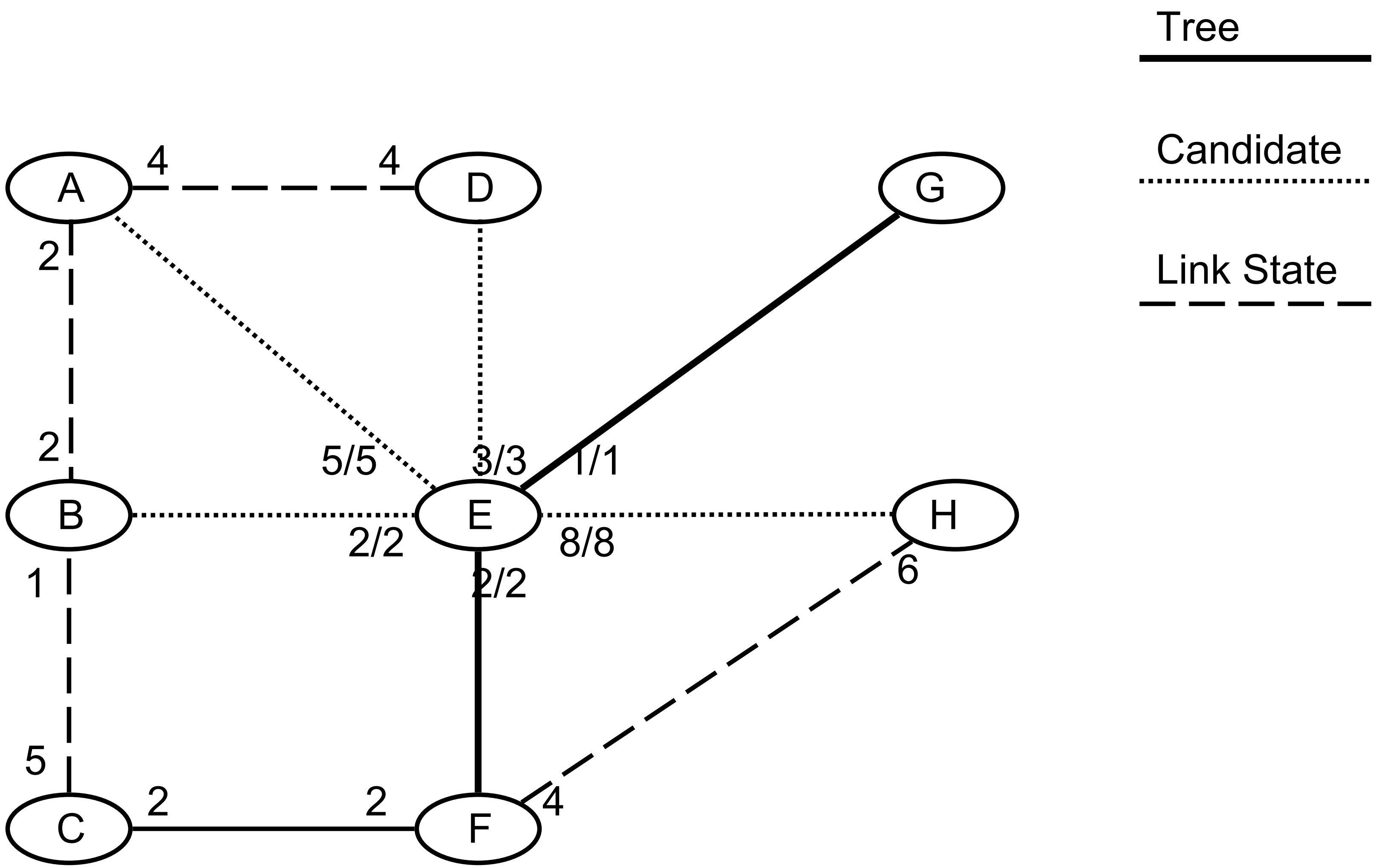
Cheapest candidate added to tree.

From E's perspective, steps 4 & 5



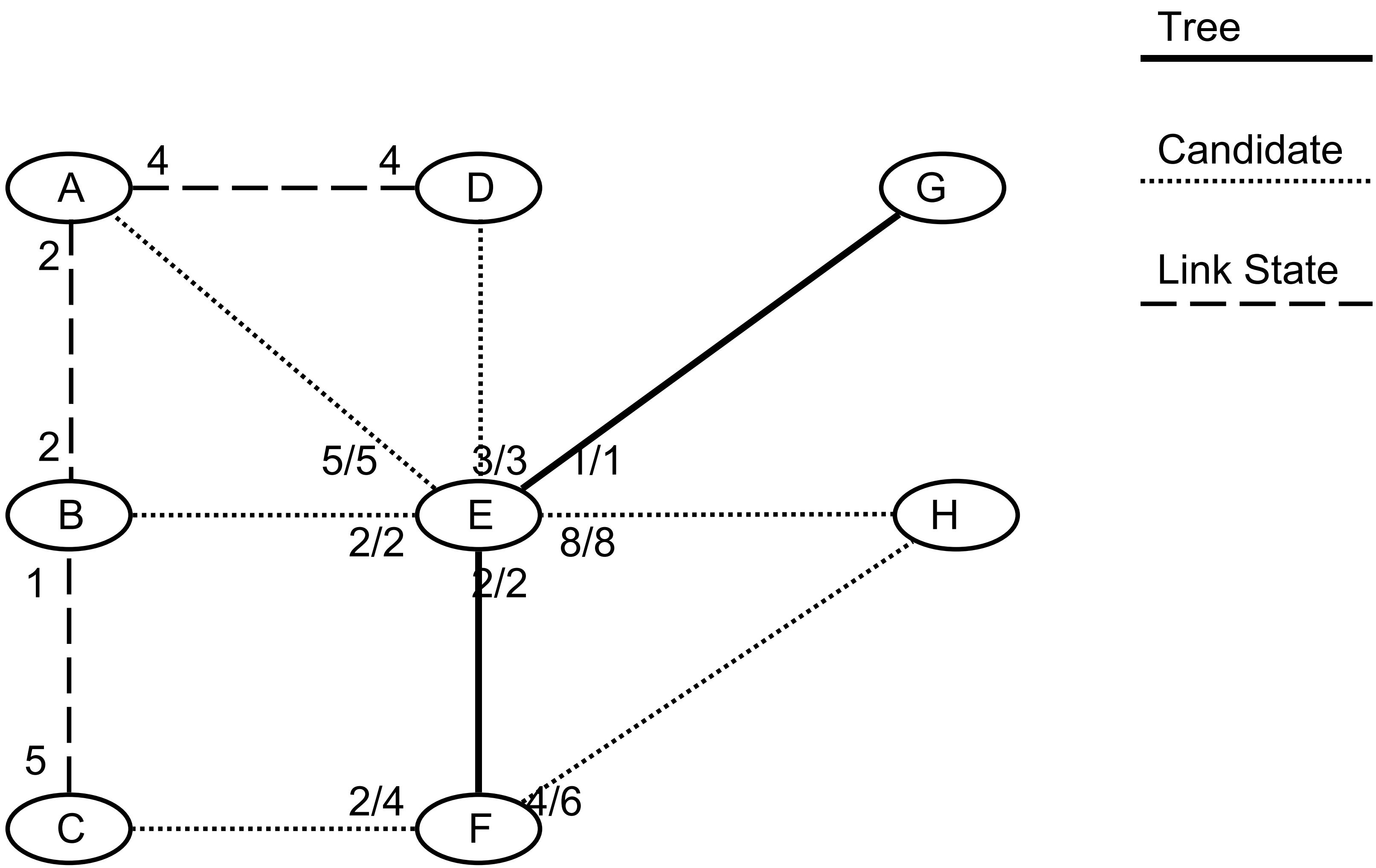
G's neighbor becomes candidate; root costs computed.

From E's perspective, steps 6 & 7



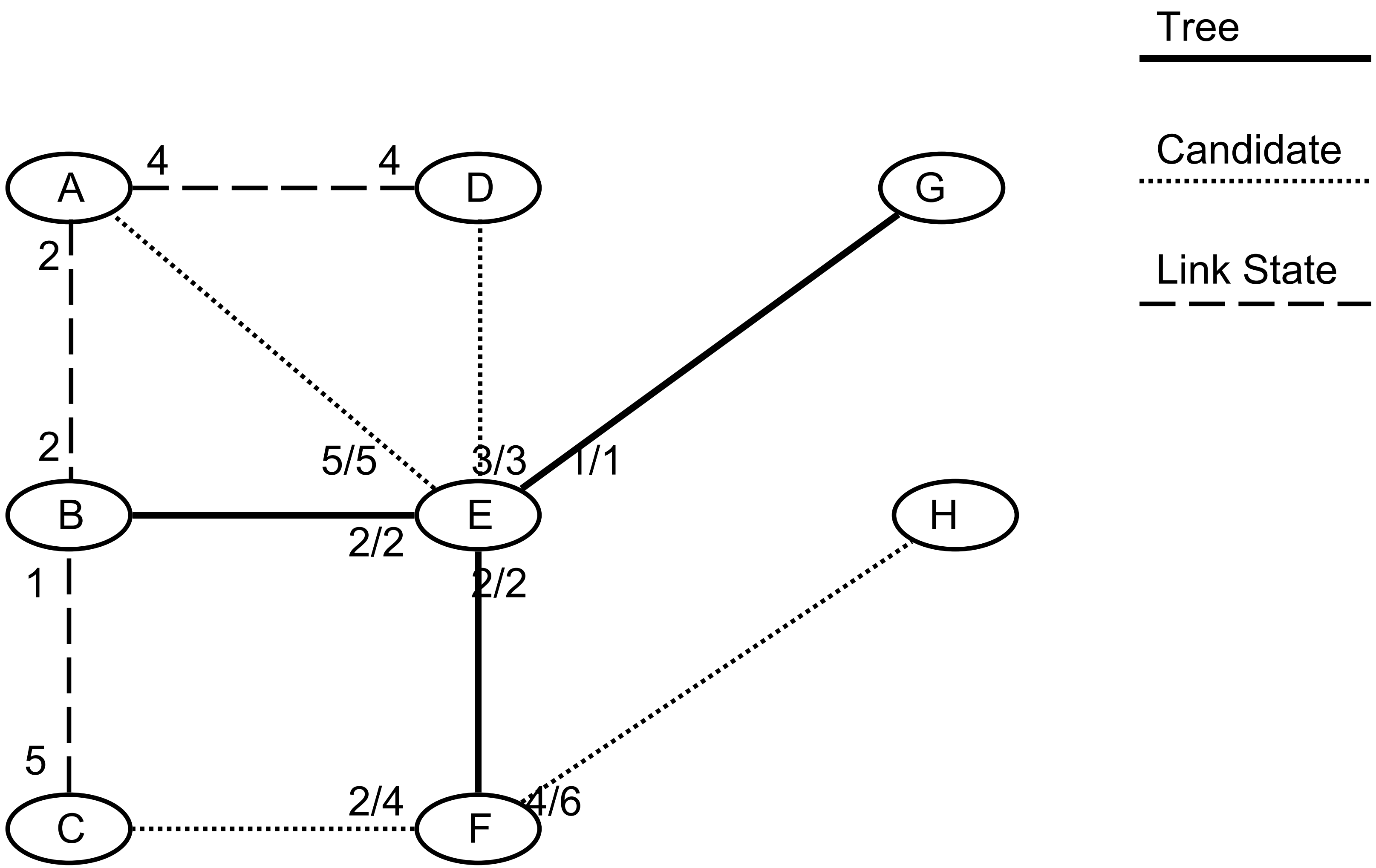
GD removed (ED better path to D); EF (cheapest) added to tree.

From E’s perspective, steps 8 & 9



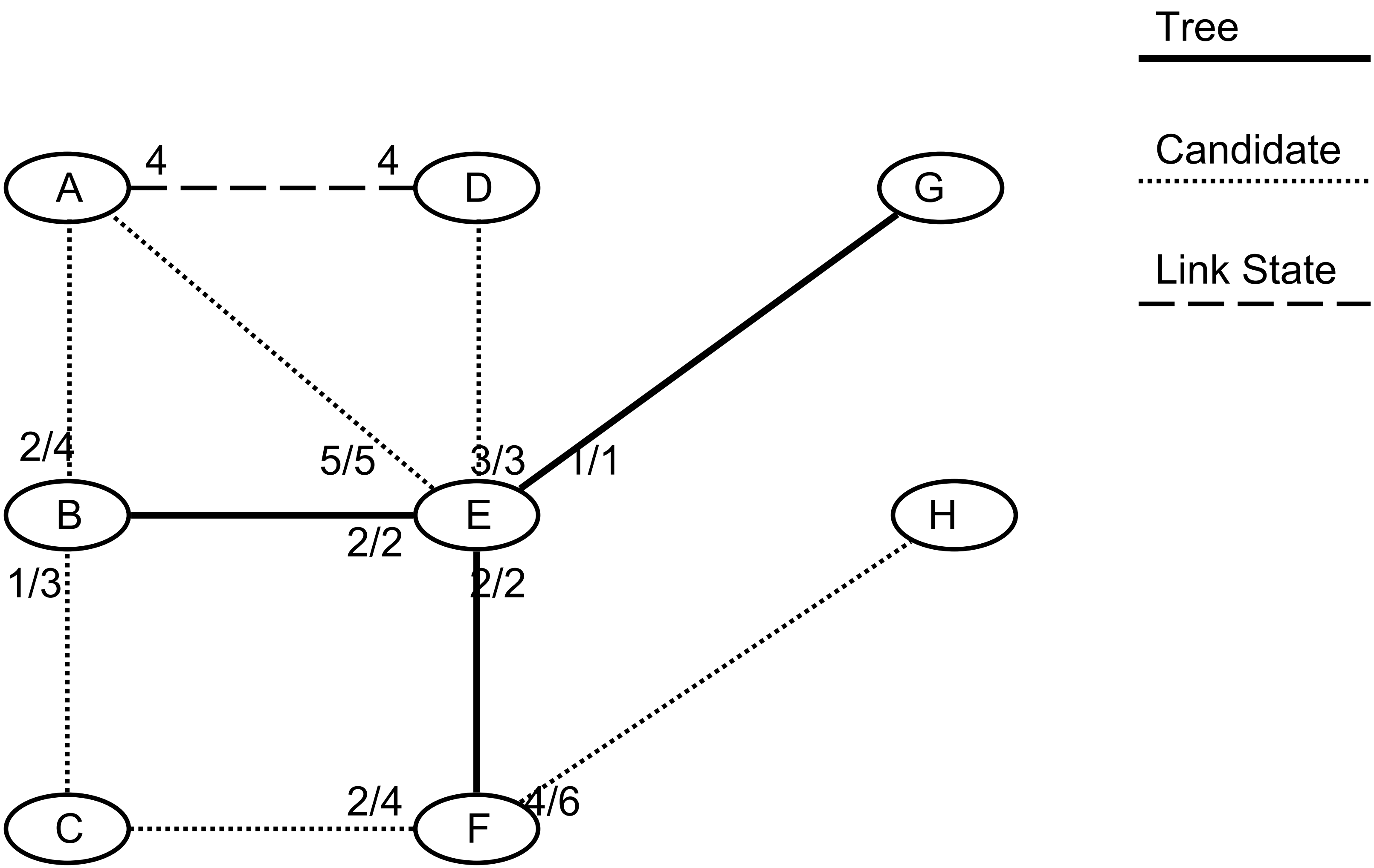
F’s neighbors become candidates; root costs computed

From E’s perspective, steps 10 & 11



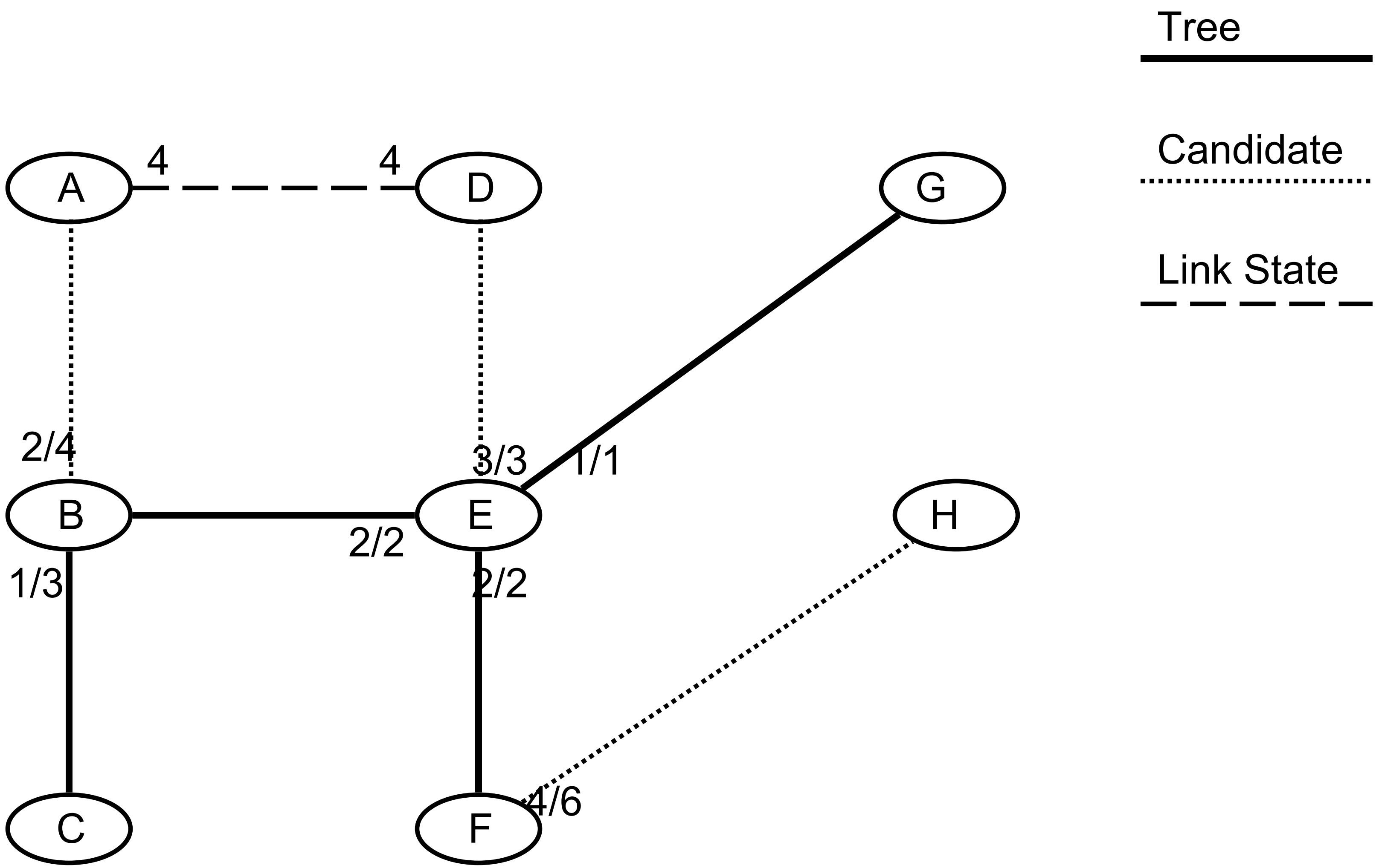
EH removed (EFH better); EB (chepest) added to tree.

From E’s perspective, steps 12 & 13



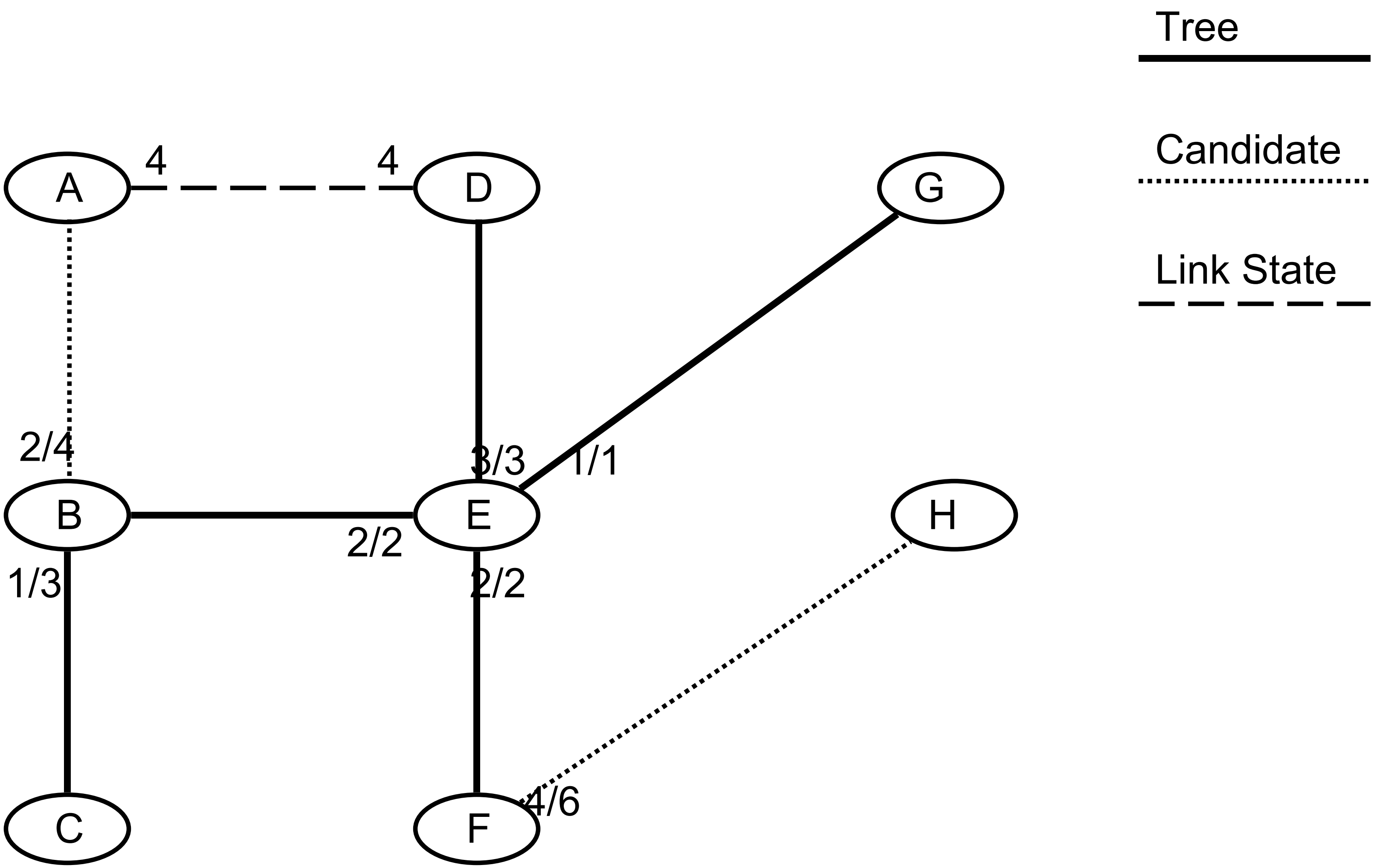
B’s neighbors become candidates; root costs computed.

From E's perspective, steps 14, 15 & 16



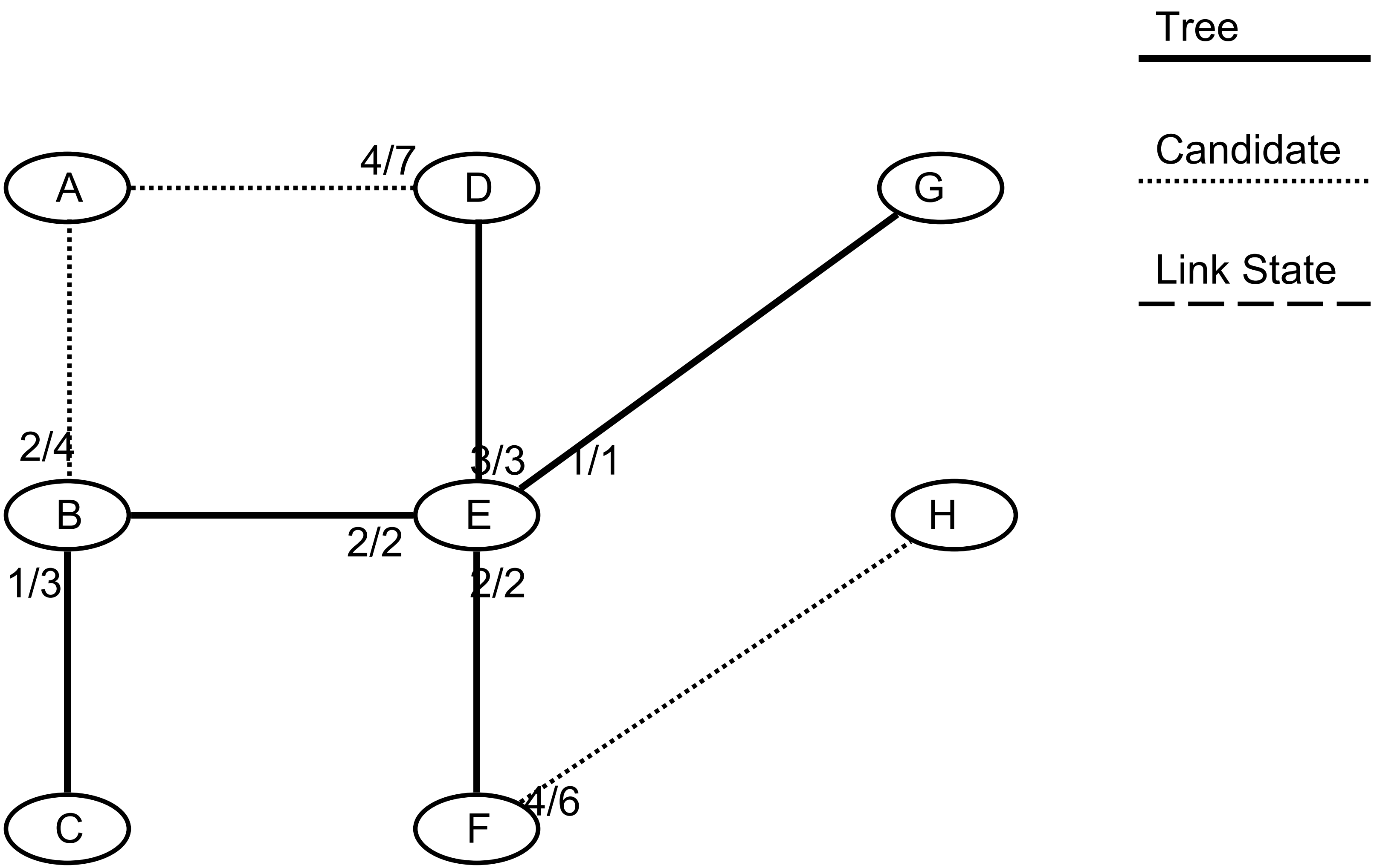
FC&EA removed (EBC&EBA cheaper); BC added to tree.

From E's perspective, step 17



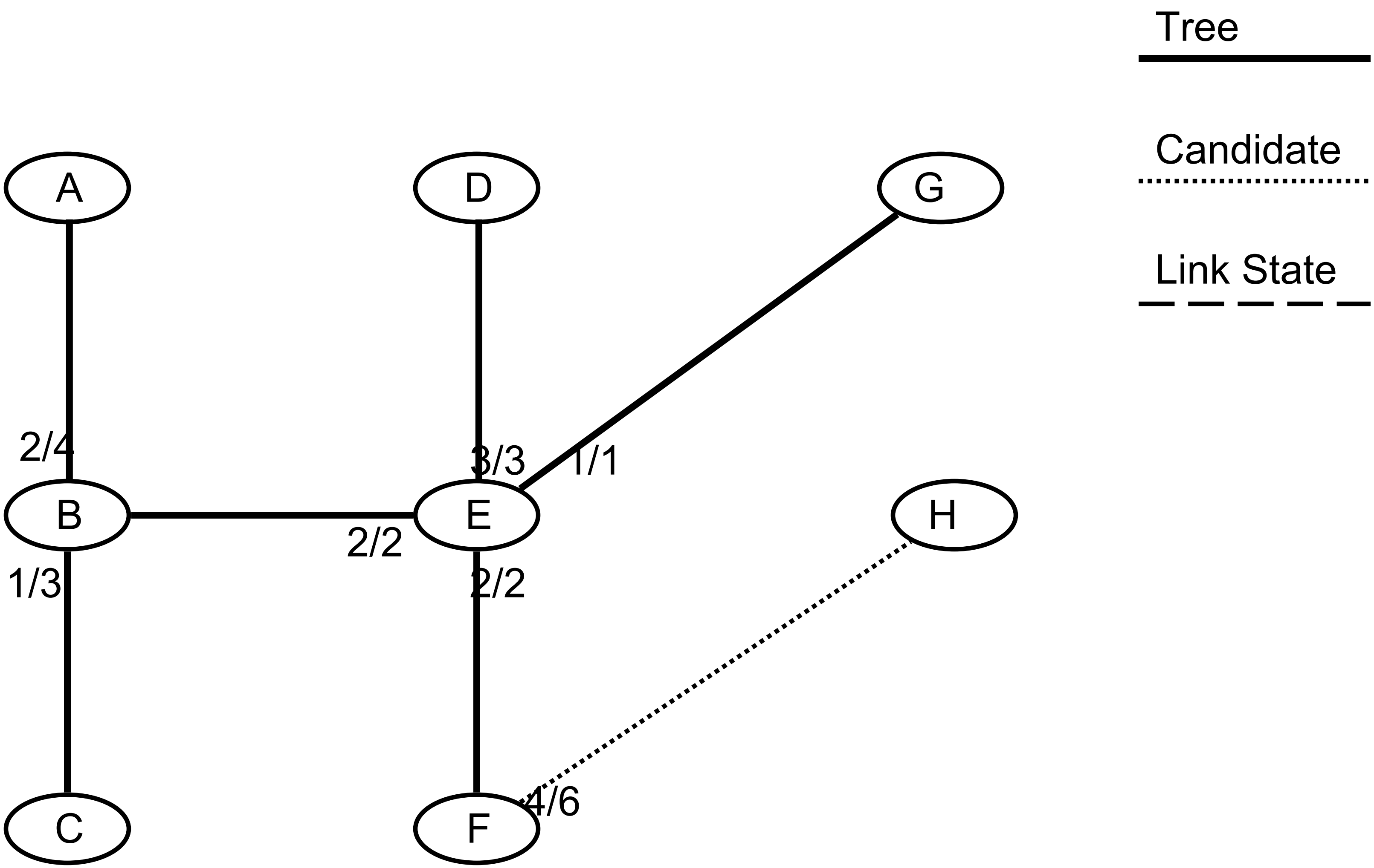
No neighbors to add. ED (cheapest) added to tree.

From E’s perspective, steps 18 & 19



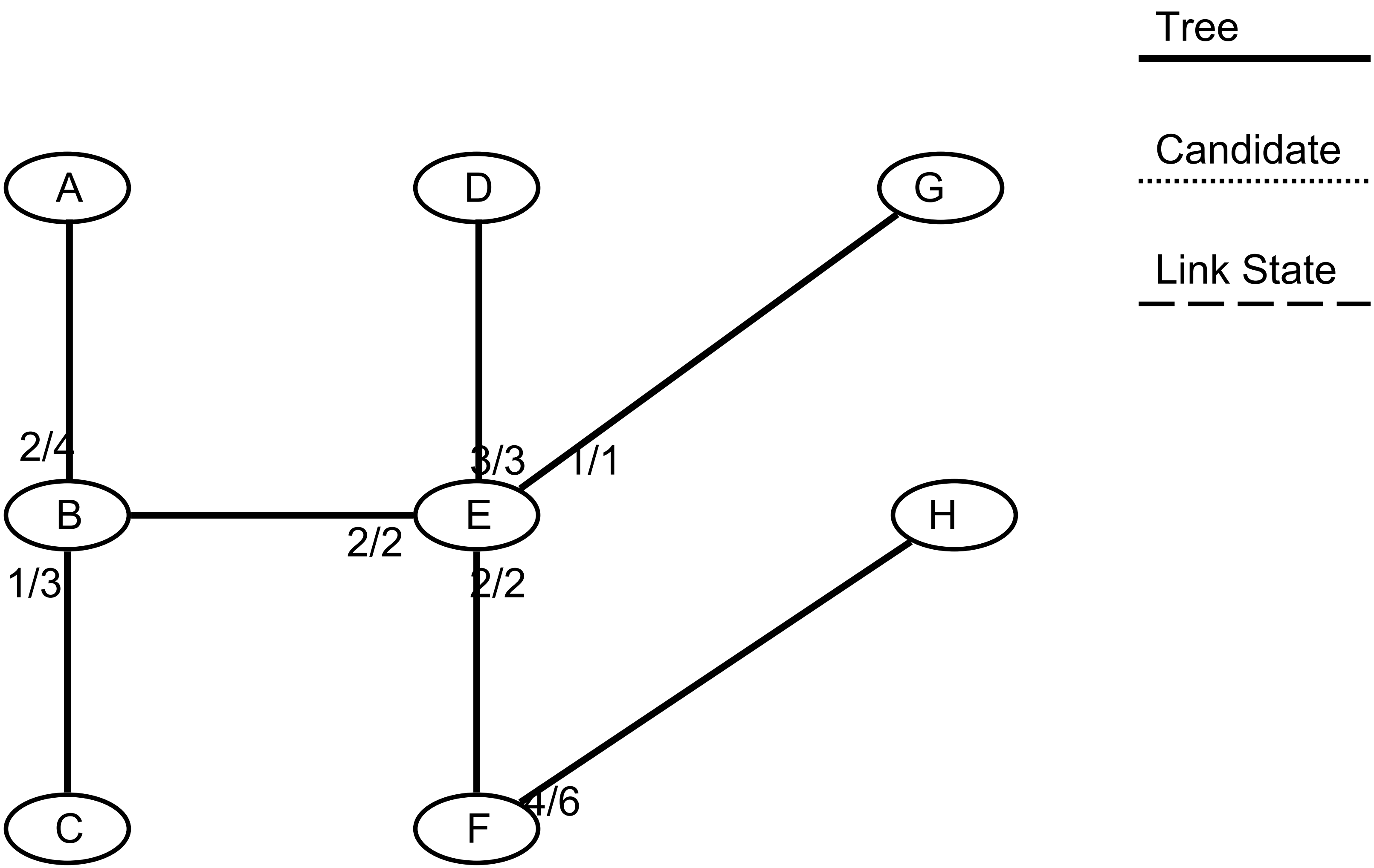
D’s neighbor added to candidates; root costs computed.

From E's perspective, steps 20 & 21



DA removed (EBA cheaper). BA (cheapest) added to tree.

From E's perspective, last step. Done!



FH added to tree

E’s forwarding table

Destination	Next hop
A	B
B	B
C	B
D	D
E	self
F	F
G	G
H	F

- G is directly attached, just give him the packet.

What about the return path?

- Work out G’s tree.
- Next hop for A is E.
- At E, next hop for A is B.
- Work out B’s tree.
- At B, next hop for A is A.

- So the route is not symmetric.
- This is a very common thing.
- This is a very good thing.
- Playing with Link costs allows us to do dictate how traffic flows.
- This is an example of ***Traffic Engineering***.