Chapter 4: Network Layer

Three Types of Communications

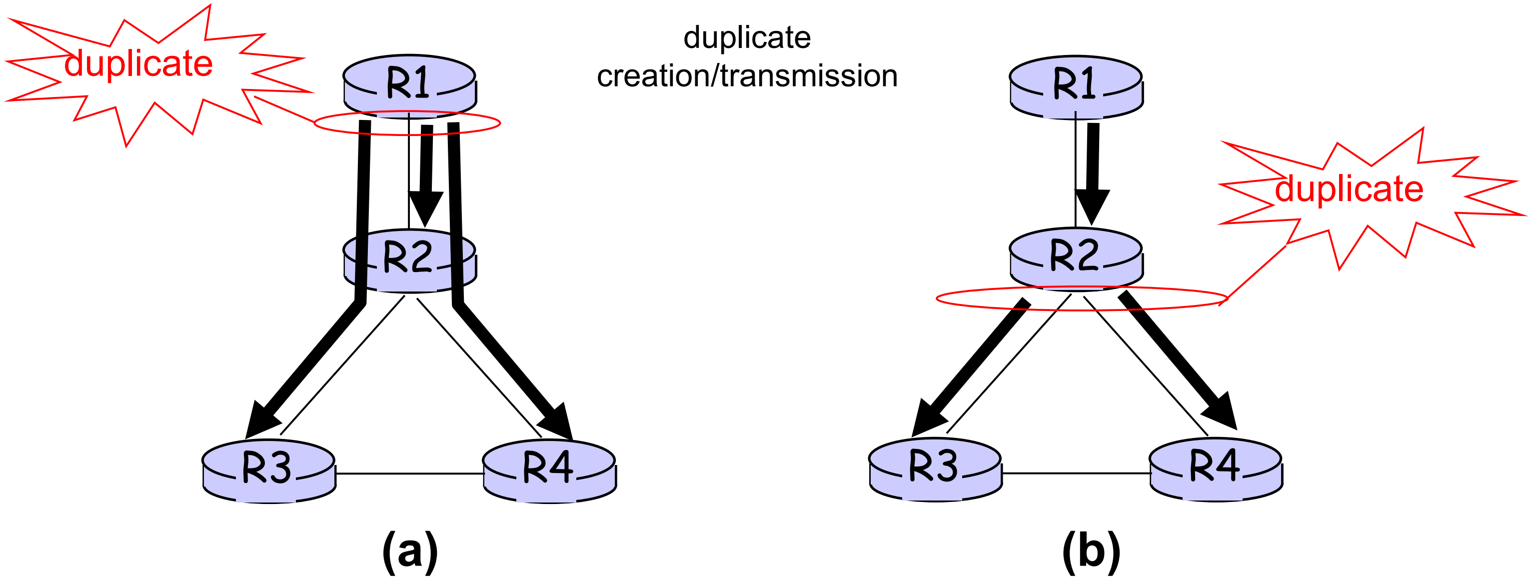
Instructor: HOU, Fen

2025

Three types of communications

- Unicast (one-to-one) 单播
  - Single source, single receiver
- Broadcast (one-to-all) 广播
  - Send same packet to all receivers
  - “all” is limited in some way such as in a LAN, subnet, or in a organization.
- Multicast (one-to-several) 多播
  - Send some packet to multiple receivers
  - Applications: bulk data transfer (e.g., software upgrade from the developer to users needing the upgrade), shared data applications (e.g., whiteboard), remote education (the transfer of the audio, video, and text of a live lecture to a set of distributed students) , teleconference, etc.

Multicast/Broadcast Routing

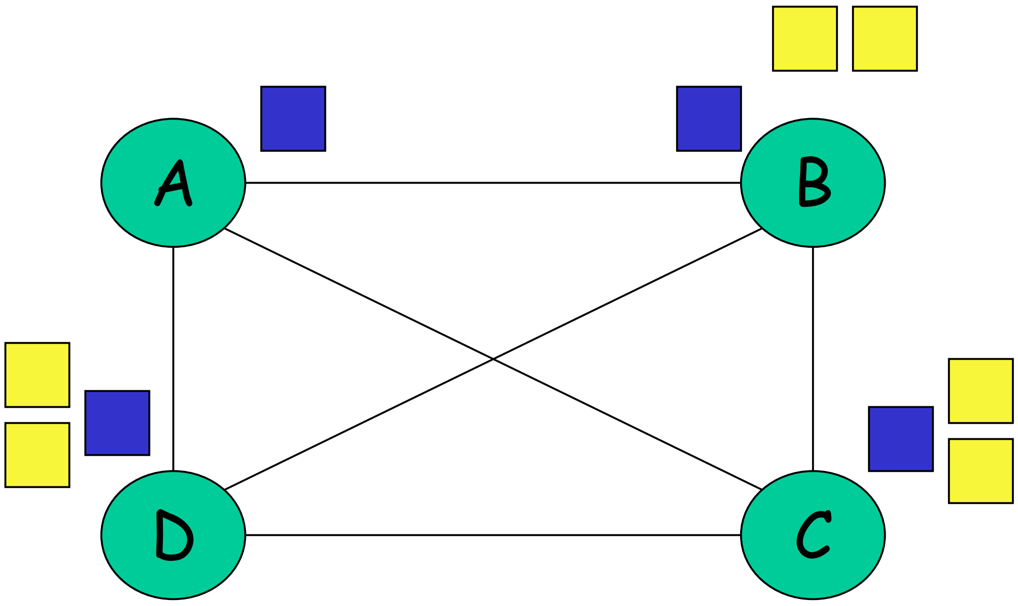


Source-duplication versus in-network duplication.
(a) source duplication, (b) in-network duplication

- Source duplication (N-way unicast)
  - Simple (advantage)
  - Inefficient (disadvantage)
  - Source may not know the addresses of all receivers (disadvantage)
- In-network duplication: make the copy at the network node

Multicast/Broadcast Routing

- Uncontrolled flooding: when node receives a broadcast packet, it sends copy to all neighbors except the neighbor from which it received the packet
    - Problems: cycles & broadcast storm



Multicast/Broadcast Routing

- Controlled flooding: node only broadcasts packets if it hasn't broadcast the same packet before
  - Sequence-number-controlled flooding
    - Source assigns increasing seq. no. to broadcast packets
    - Nodes keep track of packet id (source address and sequence number) already broadcasted
    - Discard packets with old seq. no.

Multicast/Broadcast Routing

- Reverse path forwarding (RPF):
  - A router forwards packet only if it arrives through the shortest path between this router and source.
  - Use unicast routing information: RPF need only know the neighbor on its unicast shortest path to the sender
  - Cannot avoid the reception of redundant broadcast packets

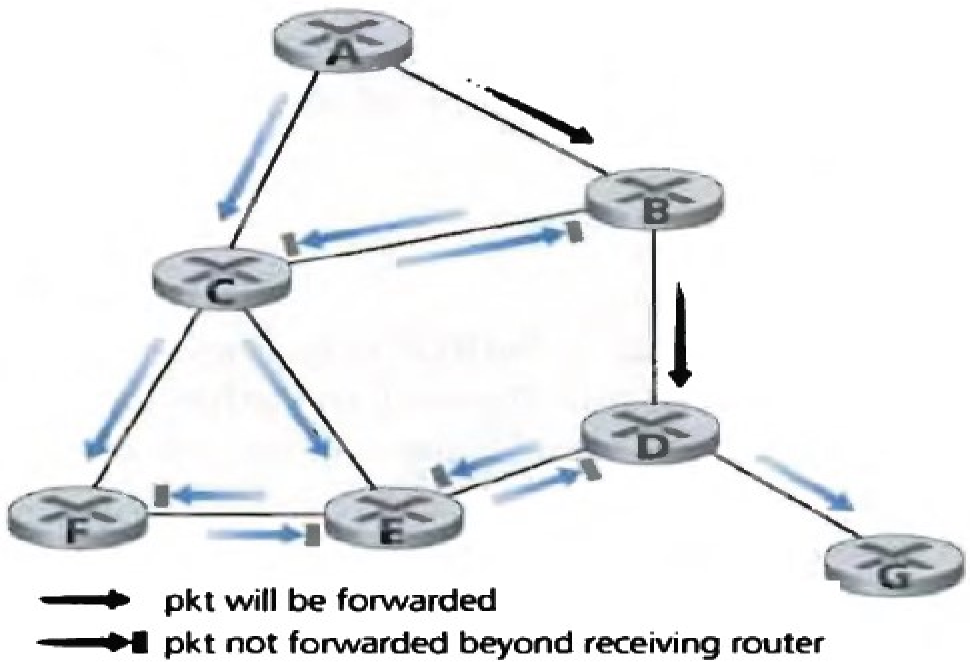
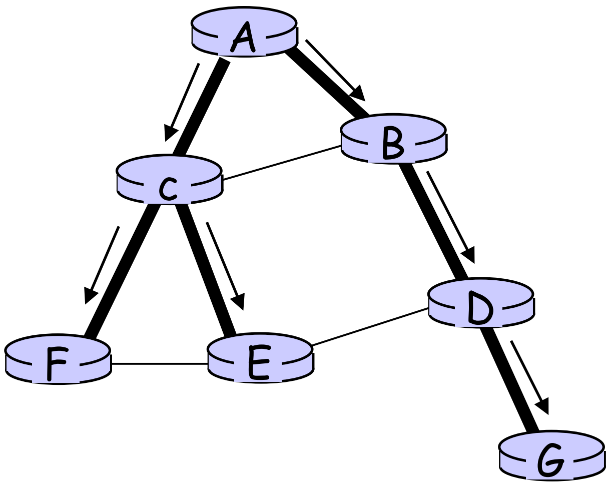


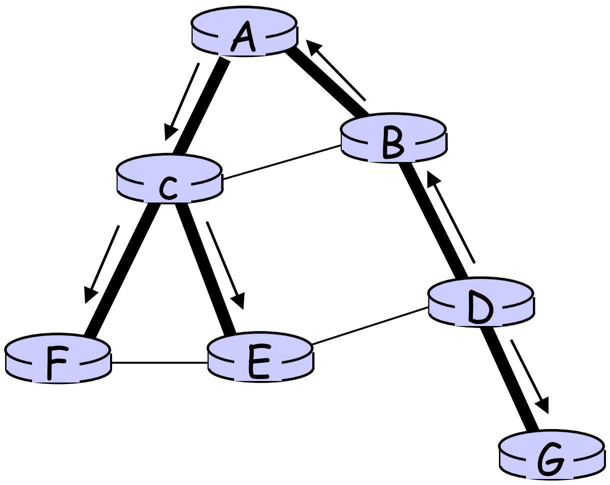
Figure 4.44 ♦ Reverse path forwarding

Spanning Tree

- Spanning tree approach
  - Objective: Each node exactly receive one copy. No redundant packets received by any node



(a) Broadcast initiated at A



(b) Broadcast initiated at D

Spanning Tree

- Terms:
  - Tree: in the graph theory, a tree is an undirected graph in which any two vertices are connected by exactly one simple path. Any connected graph without cycles is a tree.
  - A spanning tree of a graph is a tree consisting all nodes in a graph.
  - The cost of a tree: if each link has an associated cost, the cost of a tree is the sum of the link costs in this tree.
  - A minimum spanning tree: a spanning tree whose cost is the minimum of all of the graph's spanning trees.

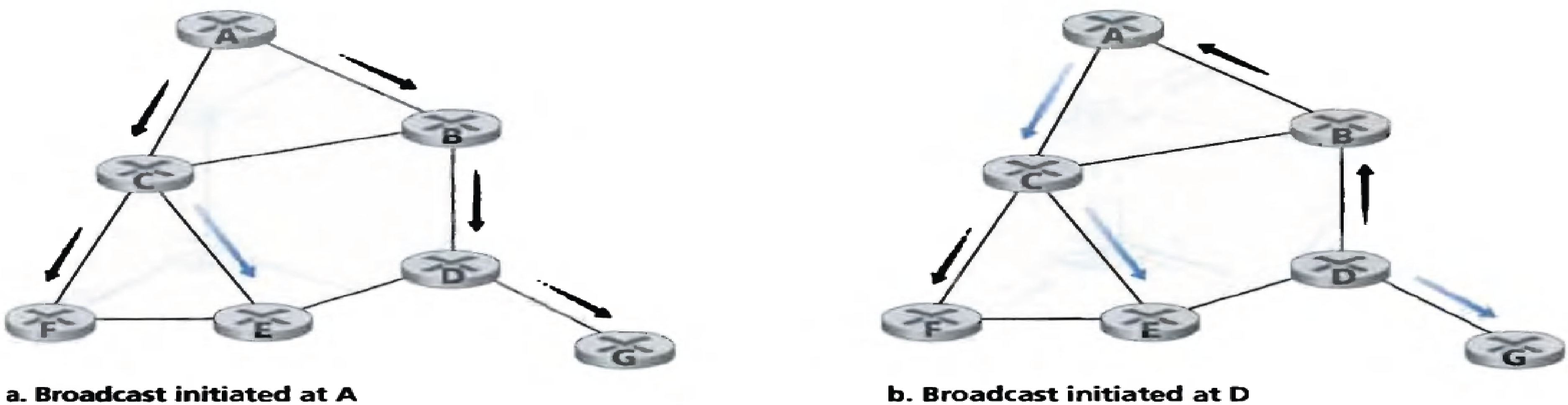


Figure 4.45 ♦ Broadcast along a spanning tree

Prim's Algorithm

- It is an algorithm used to find the minimum spinning tree.

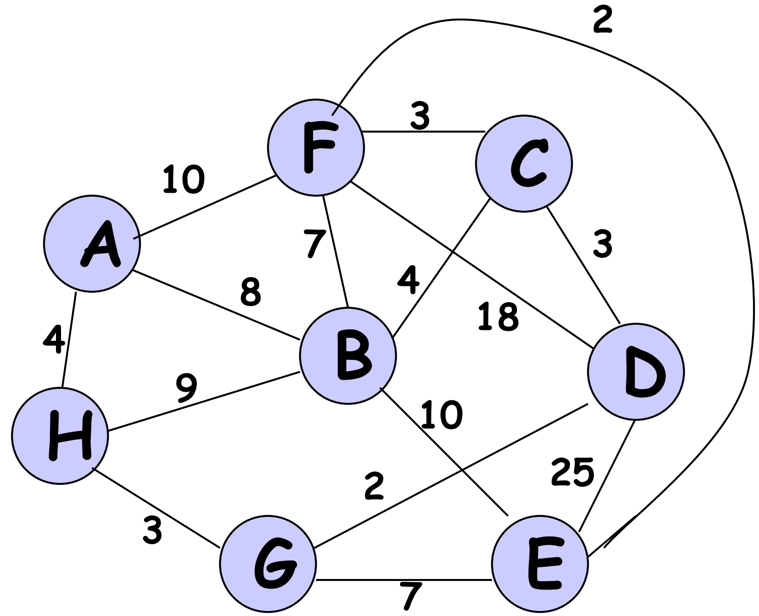
Prim's Algorithm

```
Initialization:
  T= {u}
  for all nodes v
    if v is neighbor to u
        then D(v) = c(u,v)
    else D(v) = ∞
 Loop
   find a node w without the set T and with the minimum edge weight D(w)
  add w to T
  update D(v) for each node v which is the neighboring nodes of w and
    is not in the Set T using the following rule:
        D(v) = min{ D(v), c(w,v) }
    /* new edge weight to v is either old edge weight to v or known
    shortest edge  weight to v */
   For nodes that are not adjacent to w, we don't need to
   do any update
 until |T|=N   /*all nodes are in T*/
```

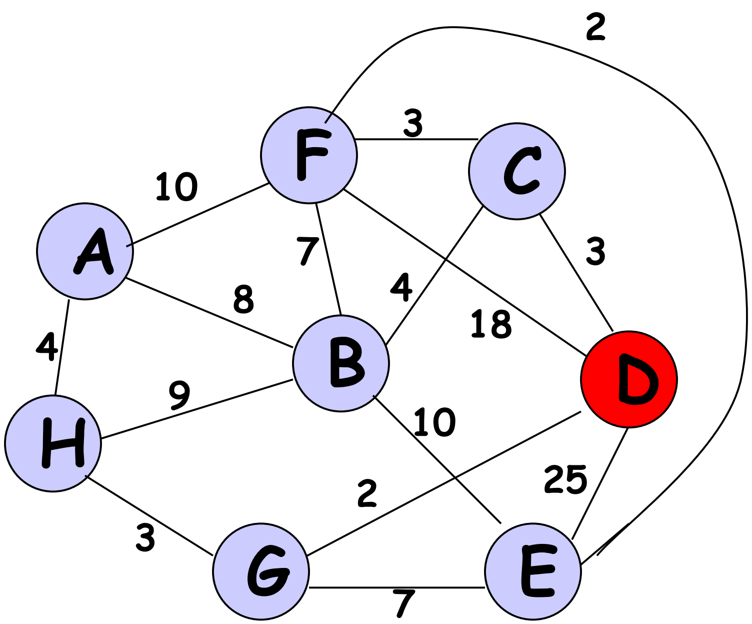
- Iterative process: after n iterations, a minimum spinning tree is formed.

Prim's Algorithm

- Problem: Given a connected, undirected, weighted graph G with n vertices, Please find a spinning tree of G starting from root vertex D using Prim's algorithm.



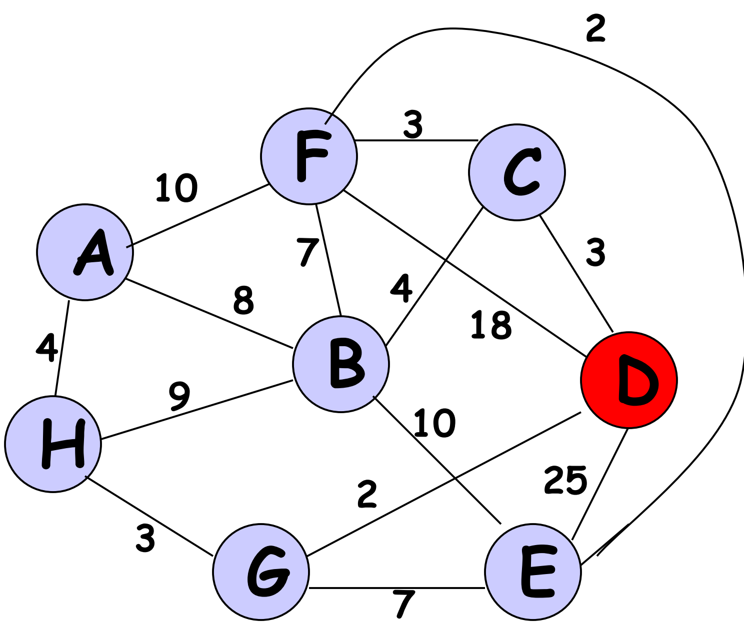
Prim's Algorithm: 1st iteration-initialization



Start with any node, say D

	T	D (v)	Path (v)
A		∞	–
B		∞	–
C		∞	–
D	{D}	0	–
E		∞	–
F		∞	–
G		∞	–
H		∞	–

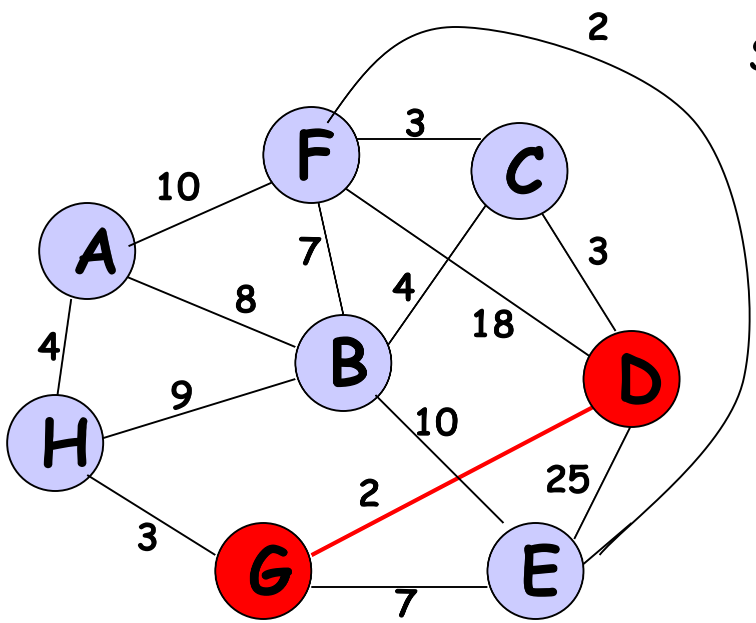
Prim's Algorithm: 1st iteration-initialization



Update distances of adjacent, unselected nodes

	T	*D (v)*	*Path (v)*
A		∞	–
B		∞	–
C		3	D-C
D	{D}	0	–
E		25	D-E
F		18	D-F
G		2	D-G
H		∞	–

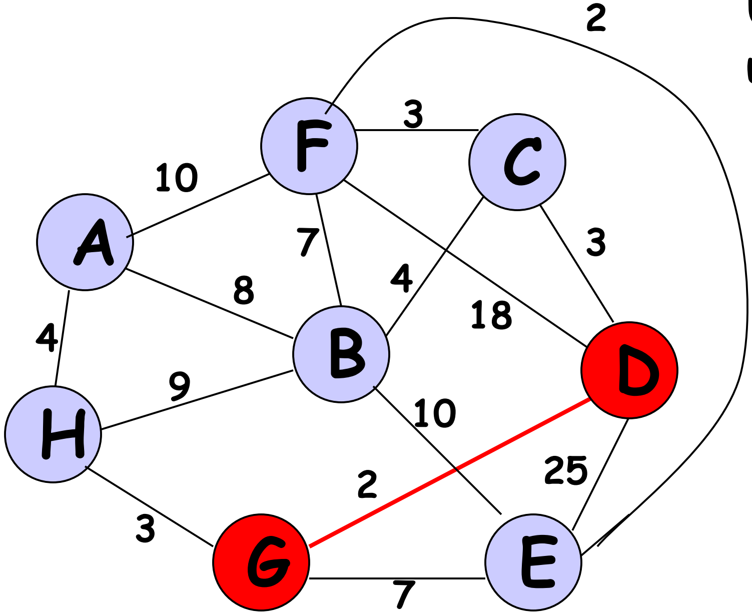
Prim's Algorithm: 2nd iteration



Select node with minimum distance

	T	*D (v)*	*Path (v)*
A		∞	–
B		∞	–
C		3	D-C
D	{D}	0	–
E		25	D-E
F		18	D-F
G	{D,G}	2	D-G
H		∞	–

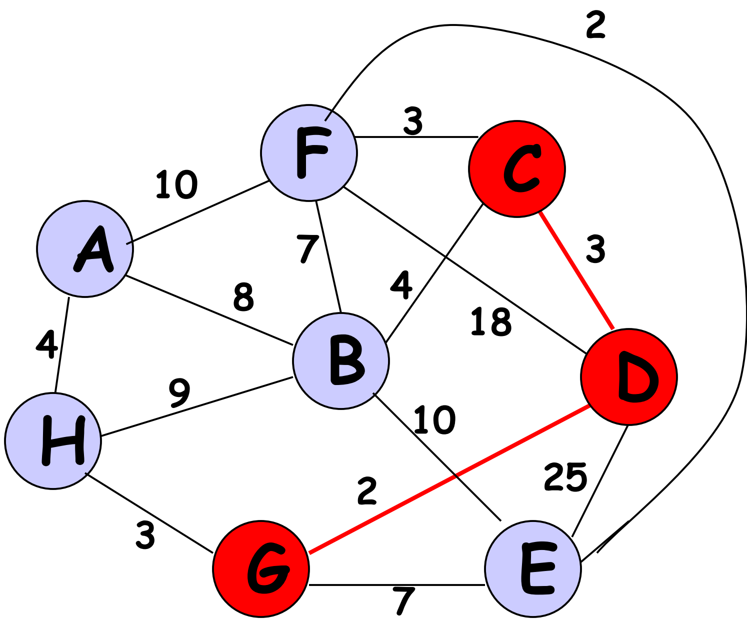
Prim's Algorithm: 2nd iteration



Update distances of adjacent, unselected nodes

	T	D (v)	Path (v)
A		∞	–
B		∞	–
C		3	D-C
D	{D}	0	–
E		7	G-E
F		18	D-F
G	{D,G}	2	D-G
H		3	G-H

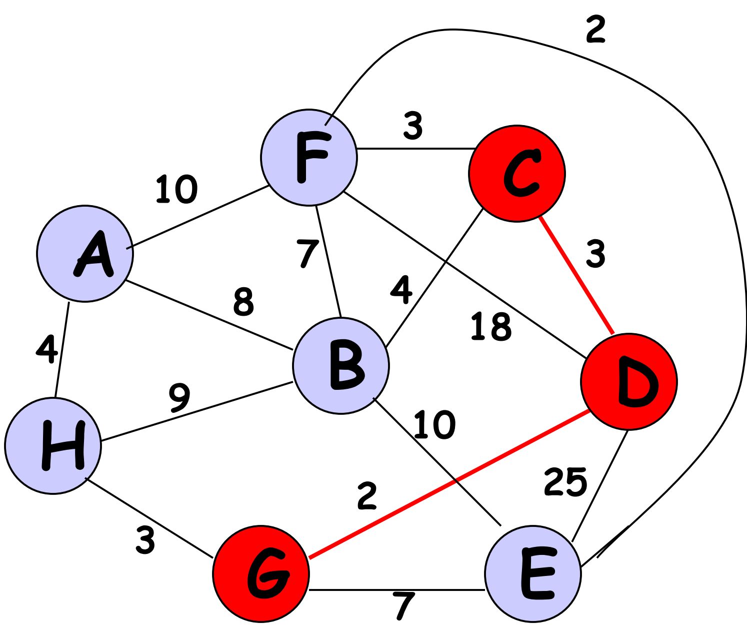
Prim's Algorithm: 3rd iteration



Select node with minimum distance

	T	*D (v)*	*Path (v)*
A		∞	–
B		∞	–
C	{D,G,C}	3	D-C
D	{D}	0	–
E		7	G-E
F		18	D-F
G	{D,G}	2	D-G
H		3	G-H

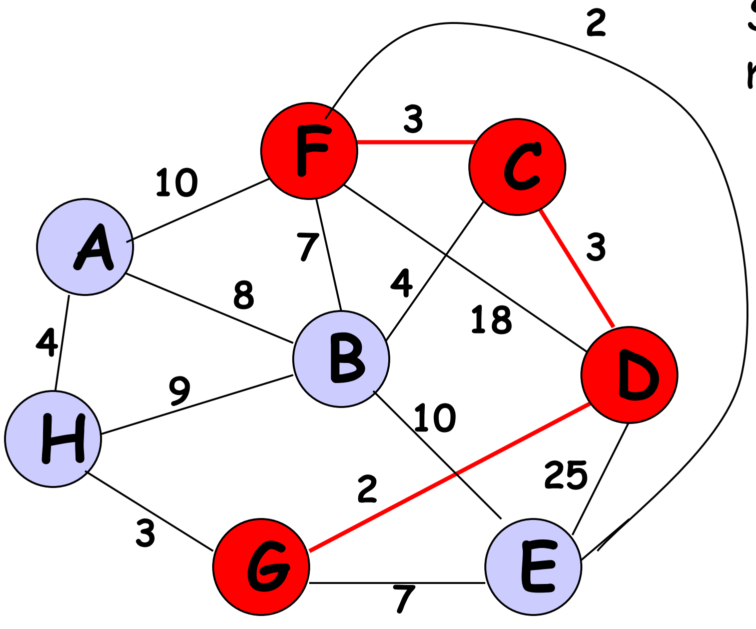
Prim's Algorithm: 3rd iteration



Update distances of adjacent, unselected nodes

	T	*D (v)*	*Path (v)*
A		∞	–
B		4	C-B
C	{D,G,C}	3	D-C
D	{D}	0	–
E		7	G-E
F		3	C-F
G	{D,G}	2	D-G
H		3	G-H

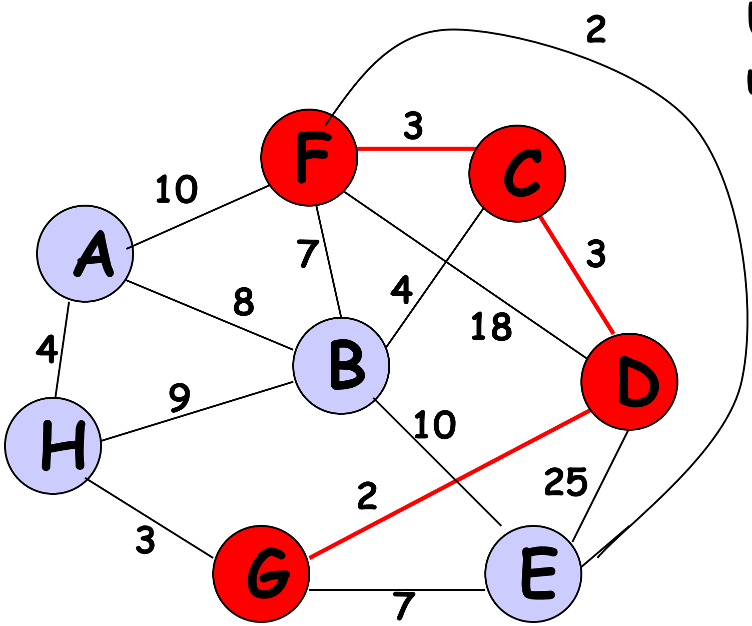
Prim's Algorithm: 4th iteration



Select node with minimum distance

	T	*D (v)*	*Path (v)*
A		∞	–
B		4	C-B
C	{D,G,C}	3	D-C
D	{D}	0	–
E		7	G-E
F	{D,G,C,F}	3	C-F
G	{D,G}	2	D-G
H		3	G-H

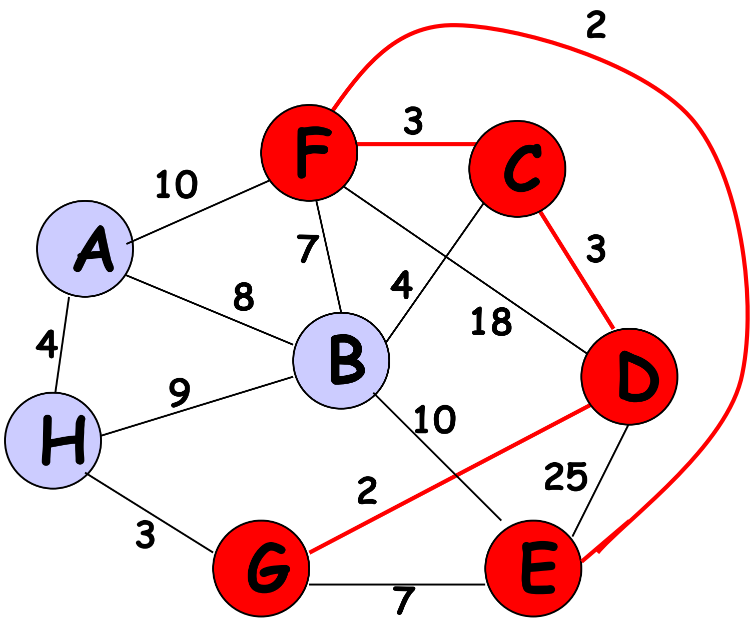
Prim's Algorithm: 4th iteration



Update distances of adjacent, unselected nodes

	T	*D (v)*	*Path (v)*
A		10	F-A
B		4	C-B
C	{D,G,C}	3	D-C
D	{D}	0	–
E		2	F-E
F	{D,G,C,F}	3	C-F
G	{D,G}	2	D-G
H		3	G-H

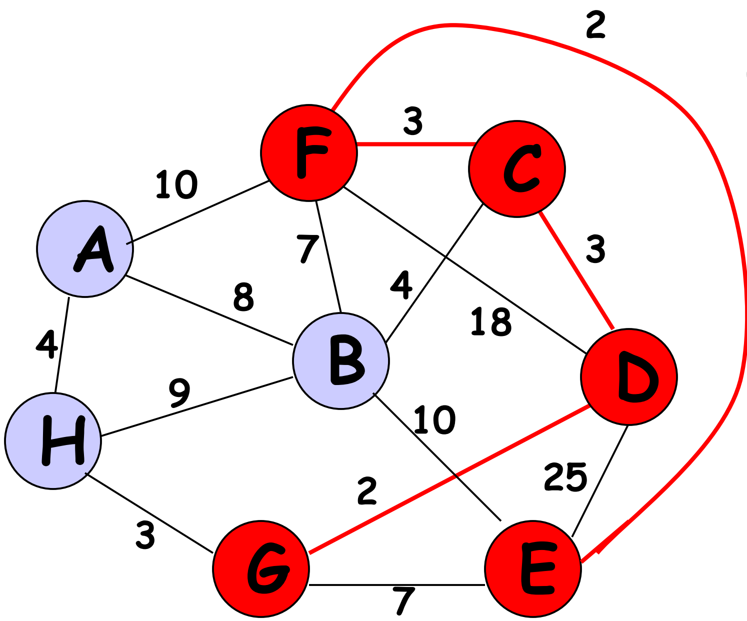
Prim's Algorithm: 5th iteration



Select node with minimum distance

	T	*D (v)*	*Path (v)*
A		10	F-A
B		4	C-B
C	{D,G,C}	3	D-C
D	{D}	0	–
E	{D,G,C,F,E}	2	F-E
F	{D,G,C,F}	3	C-F
G	{D,G}	2	D-G
H		3	G-H

Prim's Algorithm: 5th iteration

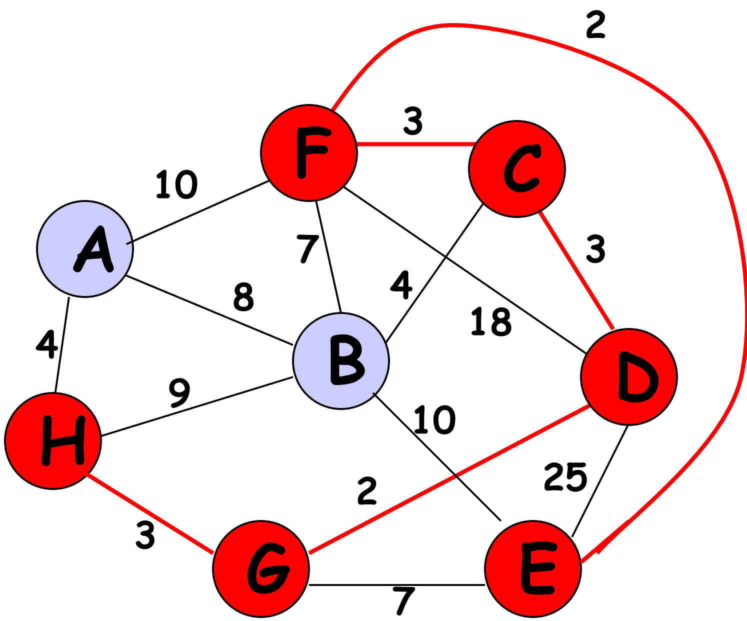


Update distances of adjacent, unselected nodes

	T	D (v)	Path (v)
A		10	F-A
B		4	C-B
C	{D,G,C}	3	D-C
D	{D}	0	–
E	{D,G,C, F,E}	2	F-E
F	{D,G,C, F}	3	C-F
G	{D,G}	2	D-G
H		3	G-H

Table entries unchanged

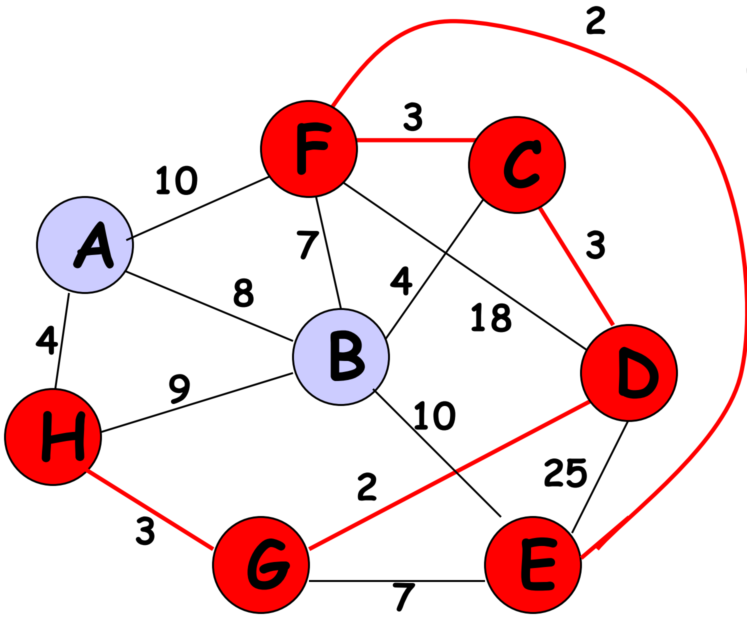
Prim's Algorithm: 6^{th} iteration



Select node with minimum distance

	T	D (v)	Path (v)
A		10	F-A
B		4	C-B
C	{D,G,C}	3	D-C
D	{D}	0	–
E	{D,G,C,F ,E}	2	F-E
F	{D,G,C,F}	3	C-F
G	{D,G}	2	D-G
H	{D,G,C,F ,E,H}	3	G-H

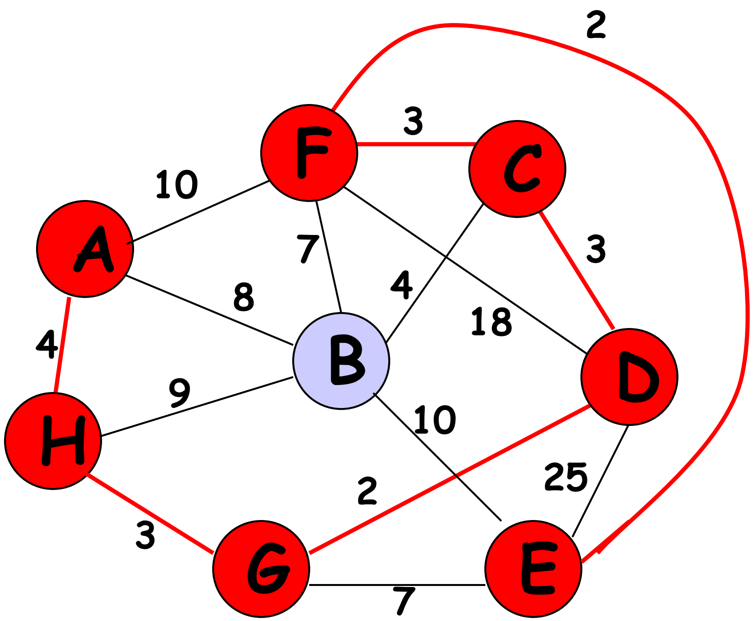
Prim's Algorithm: 6th iteration



Update distances of adjacent, unselected nodes

	T	*D (v)*	*Path (v)*
A		4	H-A
B		4	C-B
C	{D,G,C}	3	D-C
D	{D}	0	–
E	{D,G,C, F,E}	2	F-E
F	{D,G,C, F}	3	C-F
G	{D,G}	2	D-G
H	{D,G,C, F,E,H}	3	G-H

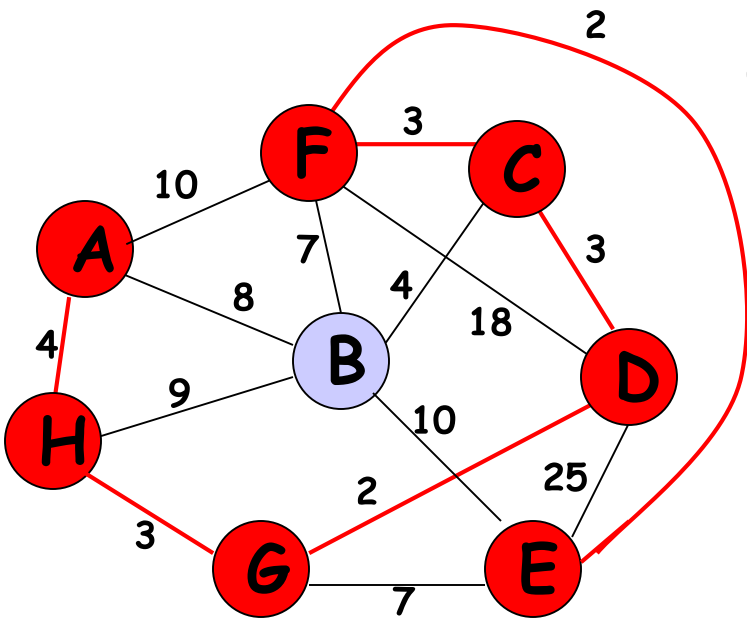
Prim's Algorithm: 7^{th} iteration



Select node with minimum distance

	T	*D (v)*	*Path (v)*
A	{D,G,C,F ,E,H,A}	4	H-A
B		4	C-B
C	{D,G,C}	3	D-C
D	{D}	0	–
E	{D,G,C,F ,E}	2	F-E
F	{D,G,C,F}	3	C-F
G	{D,G}	2	D-G
H	{D,G,C,F ,E,H}	3	G-H

Prim's Algorithm: 7^{th} iteration

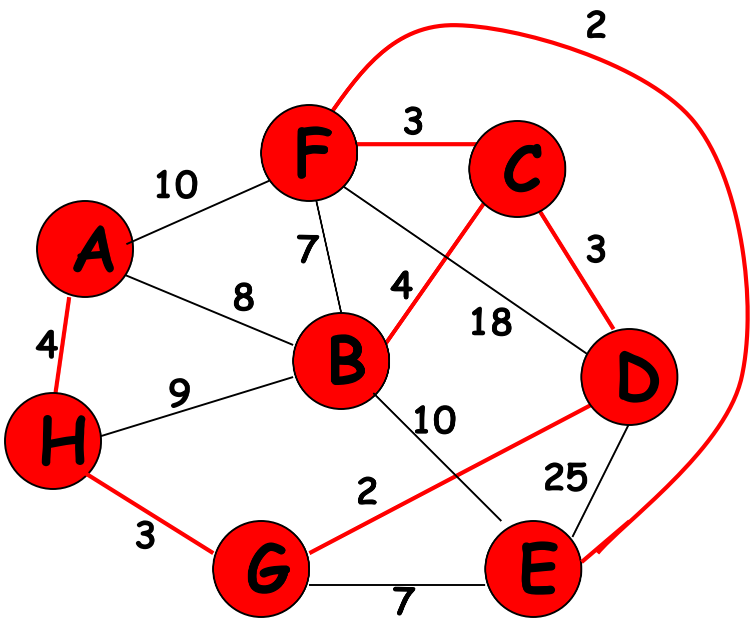


Update distances of adjacent, unselected nodes

	T	*D (v)*	*Path (v)*
A	{D,G,C,F ,E,H,A}	4	H-A
B	{D,G,C,F ,E,H,A, B}	4	C-B
C	{D,G,C}	3	D-C
D	{D}	0	–
E	{D,G,C,F ,E}	2	F-E
F	{D,G,C,F}	3	C-F
G	{D,G}	2	D-G
H	{D,G,C,F ,E,H}	3	G-H

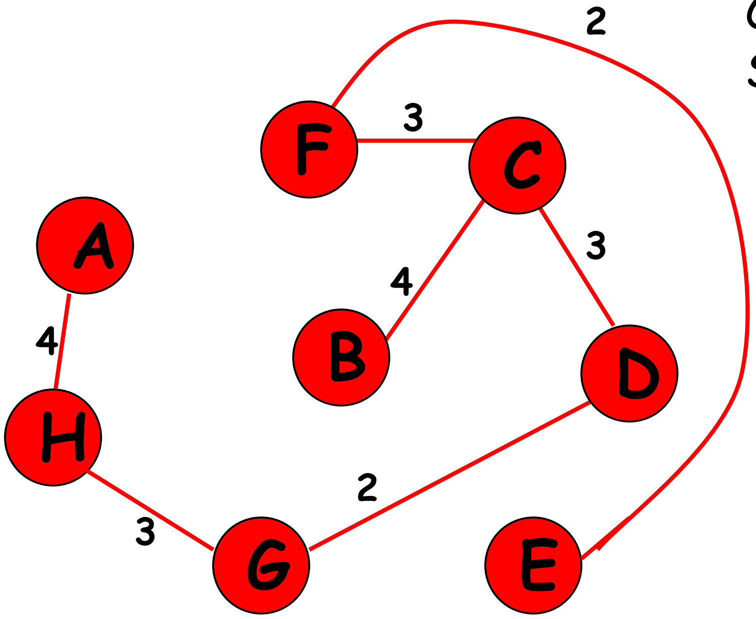
Table entries unchanged

Prim's Algorithm: 8th iteration



Select node with minimum distance

	T	*D (v)*	*Path (v)*
A	{D,G,C, F,E,H, A}	4	H-A
B	{D,G,C, F,E,H, A,B}	4	C-B
C	{D,G,C}	3	D-C
D	{D}	0	–
E	{D,G,C, F,E}	2	F-E
F	{D,G,C, F}	3	C-F
G	{D,G}	2	D-G
H	{D,G,C, F,E,H}	3	G-H

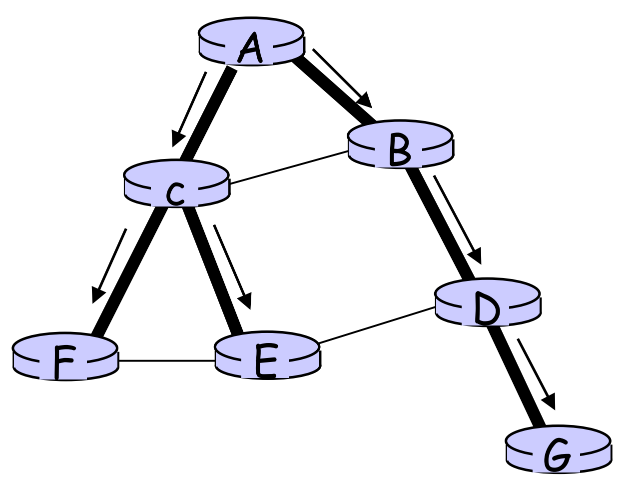


Cost of Minimum
Spanning Tree = $\Sigma\, d_v$ = 21

	T	*D (v)*	*Path (v)*
A	{D,G,C,F ,E,H,A}	4	H-A
B	{D,G,C,F ,E,H,A,B}	4	C-B
C	{D,G,C}	3	D-C
D	{D}	0	–
E	{D,G,C,F ,E}	2	F-E
F	{D,G,C,F}	3	C-F
G	{D,G}	2	D-G
H	{D,G,C,F ,E,H}	3	G-H

Spanning Tree

- Construct a spanning tree
- Nodes forward copies only along spanning tree
- Spanning tree approach
  - Avoid the redundant broadcast packets
  - The spanning tree can be used by any node to begin a broadcast



(a) Broadcast initiated at A **(b) Broadcast initiated at D**

Prim's Algorithm: Exercise

- Problem: Given a connected, undirected, weighted graph G with n vertices, find a spinning tree of G using Prim's algorithm.
- Start with any node, say node a