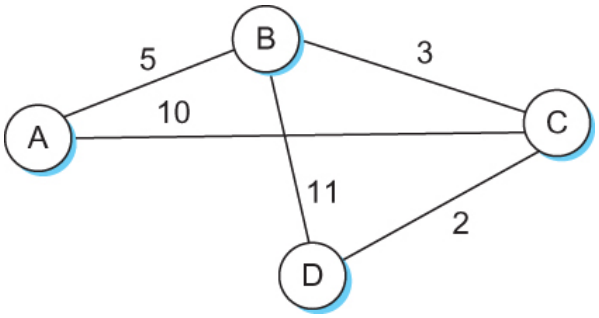
Link State Notes

- Each node(host) constructs a link state packet which contains the cost to each of its immediate neighbors
- Each node distributes its link state packet to all nodes in the network – uses reliable flooding
- Each node stores the most recent link state packet from every node
- A node uses link state packets that it has received to determine its routing table
- Use Forward Search Algorithm which is a realization of Dijkstra's Algorithm
- Storage at each node is large
- Open Shortest Path First (OSPF) uses Link State

Example of Forward Search Algorithm: Consider the following network



Consider node D, and that it has the Link State Packets from nodes A, B and C. The table below shows the link state packets in the form of **(Neighbor, Cost, Node).**

Link State Packets for All Nodes

Node	*Link State Packets*
A	**(B,5,B), (C,10,C)**
B	**(A,5,A), (C,3,C), (D,11,D)**
C	**(A,10,A), (B,3,B), (D,2,D)**
D	**(C,2,C), (B,11,B)**

Construct the routing table for Node D from the link state packets above. Using the Forward Search Algorithm, we will use the triplet form of **(Destination, cost, NextHop)** to represent the entries in our tentative and confirmed lists.

Step 1:
Start with the node that is creating the routing table – D in this case. It is entered on the confirmed list. Then all of the nodes that it can reach are added to the tentative list. To add these nodes, use the link state packet information for Node D

Confirmed:
(D, 0, -)

Tentative:
(B, 11, B), (C, 2, C)

Step 2:

Look at all entries on the tentative list and put the one with the lowest cost onto the confirmed list

Confirmed:	**Tentative:**
(D, 0, -), (C, 2, C)	**(B, 11, B)**

Step 3:
Now use the link state packets for the node just added to the confirmed list - in this case node C. Each link state packet for Node C is used to add more entries to the tentative list. If one of these entries has the same destination as an existing entry on the tentative list, keep the lowest cost entry and remove the other one.

To add these new entries, we combine the entry just added to the confirmed list with each of its link state packets.
Combining (C, 2, C) and (A, 10, C), we know C can reach A with a cost of 10 and next hop of A. Therefore at node D, A can be reached by using a next hop of C with a cost of 12. So (A, 12, C) is added to the tentative list
Combining (C, 2, C) and (B, 3, C) gives a tentative list entry of (B, 5, C)
The last Link state packet is for node D, but Node D (as is Node C) is already on the confirmed list, so ignore the link state packet for reaching node D

Confirmed:	**Tentative:**
(D, 0, -), (C, 2, C)	**(B, 11, B), (A, 12, C), (B, 5, C)**

There are two entries for reaching node B, so keep the lowest cost entry

Confirmed:	**Tentative:**
(D, 0, -), (C, 2, C)	**(A, 12, C), (B, 5, C)**

Step 4:
Pulling the lowest cost entry from the tentative list and moving it to the confirmed list

Confirmed:	**Tentative:**
(D, 0, -), (C, 2, C), (B, 5, C)	**(A, 12, C)**

Step 5:

Now use the link state packets for the node just added to the confirmed list - in this case node B. Each link state packet for Node B is used to create new entries for the tentative list. If one of these entries has the same destination as an existing entry on the tentative list, keep the lowest cost entry and remove the other one.

To add these new entries, we combine the entry just added to the confirmed list with each of its link state packets.

Combining (B, 5, C) and (A, 5, B), we know B can reach A with and additional cost of 5. Therefore node D can reach node A using a next hop of C with a total cost of 10. So (A, 10, C) is added to the tentative list

The other two link state packets for node B have destination nodes already on the confirmed list, so these link state packets are ignored.

Confirmed:	**Tentative:**
(D, 0, -), (C, 2, C), (B, 5, C)	**(A, 12, C), (A, 10, C)**

There are two entries for reaching node A, so keep the lowest cost entry

Confirmed:	**Tentative:**
(D, 0, -), (C, 2, C), (B, 5, C)	**(A, 10, C)**

Step 6:
Only one entry on the tentative list, so add it to the confirmed list

Confirmed:	**Tentative:**
(D, 0, -), (C, 2, C), (B, 5, C), (A, 10, C)	

Now use the link state packets for the node just added to the confirmed list - in this case node A. Each of the link state packets for Node A are used to create new entries for the tentative list. All the link state packets for node A have destination nodes already on the list, so no new entries are put on the tentative list.

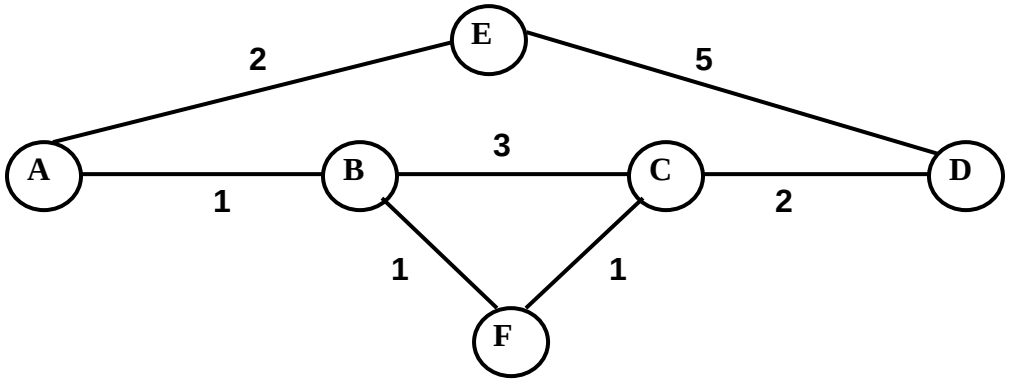
Since the tentative list is empty, the algorithm is complete

The routing table is then written directly from the confirmed list. All of the destinations are shown, the cost to that destination and the next hop to take to reach the destination.

Node D Routing Table

Destination	*Cost*	*NextHop*
A	**10**	**C**
B	**5**	**C**
C	**2**	**C**

In this example, the link state packets will be listed from each node, and will use the format of (Neighbor, Cost) – this format is slightly different from the above example



The following table shows the link state packets that each node is transmitting.

Link State Packets (LSP) for each Node					
LSP node A	**LSP Node B**	**LSP Node C**	**LSP Node D**	**LSP Node E**	**LSP Node F**
(B, 1)	**(A, 1)**	**(B, 3)**	**(C, 2)**	**(A, 2)**	**(B, 1)**
(E, 2)	**(C, 3)**	**(F, 1)**	**(E, 5)**	**(D, 5)**	**(C, 1)**
	(F, 1)	**(D, 2)**			

Find the shortest path spanning tree for node A, and create the routing table for node A from this tree. Using the Forward Search Algorithm, we will use the triplet form of **(Destination, cost, NextHop)** to represent the entries in our tentative and confirmed lists.

Step 1:
Start with the node that is creating the routing table – A in this case. It is entered on the confirmed list. Then all of the nodes that it can reach are added to the tentative list. To add these nodes, use the link state packet information for Node A

Confirmed:
(A, 0, -)

Tentative:
(B, 1, B), (E, 2, E)

Step 2:
Look at all entries on the tentative list and put the one with the lowest cost onto the confirmed list

Confirmed:
(A, 0, -), (B, 1, B)

Tentative:
(E, 2, E)

Step 3:
Now use the link state packets for the node just added to the confirmed list - in this case node B. Each link state packet for Node B is used to add more entries to the tentative list. If one of these entries has the same destination as an existing entry on the tentative list, keep the lowest cost entry and remove the other one.

To add these new entries, we combine the entry just added to the confirmed list with each of its link state packets.

- The first entry is for node A which is already on the confirmed list, so this link state is ignored.
- The next entry is (C, 3), so combining (B, 1, B) and (C, 3), we determine that node A can reach node C with a cost of 4 with next hop of node B. Therefore (C, 4, B) is added to the tentative list.
- The next entry is (F, 1), so combining (B, 1, B) and (F, 1) gives a tentative list entry of (F, 2, B) which is added to the tentative list

Confirmed: **Tentative:**
(A, 0, -), (B, 1, B) **(E, 2, E), (C, 4, B), (F, 2, B)**

Look at all entries on the tentative list and put the one with the lowest cost onto the confirmed list. Here we can pick two different entries, so for this example, select E

Confirmed: **Tentative:**
(A, 0, -), (B, 1, B), (E, 2, E) **(C, 4, B), (F, 2, B)**

Step 4:
Now use the link state packets for the node just added to the confirmed list - in this case node E. Each link state packet for Node E is used to add more entries to the tentative list. If one of these entries has the same destination as an existing entry on the tentative list, keep the lowest cost entry and remove the other one.

To add these new entries, we combine the entry just added to the confirmed list with each of its link state packets.

- The first entry is for node A which is already on the confirmed list, so this link state is ignored.
- The next entry is (D, 5), so combining (E, 2, E) and (D, 5), we determine that node A can reach node D with a cost of 2 + 5 = 7 with next hop of node E. Therefore (D, 7, E) is added to the tentative list.

Confirmed: **Tentative:**
(A, 0, -), (B, 1, B), (E, 2, E) **(C, 4, B), (F, 2, B), (D, 7, E)**

Look at all entries on the tentative list and put the one with the lowest cost onto the confirmed list.

Confirmed: **Tentative:**
(A, 0, -), (B, 1, B), (E, 2, E), (F, 2, B) **(C, 4, B), (D, 7, E)**

Step 5:
Now use the link state packets for the node just added to the confirmed list - in this case node F. Each link state packet for Node F is used to add more entries to the tentative list. To add these

new entries, we combine the entry just added to the confirmed list with each of its link state packets.

- The first entry is for node B which is already on the confirmed list, so this link state is ignored.
- The next entry is (C, 1), so combining (F, 2, B) and (C, 1), we determine that node A can reach node C with a cost of 2 + 1 = 3 with next hop of node B. Therefore (C, 3, B) is added to the tentative list.

Confirmed:	**Tentative:**
(A, 0, -), (B, 1, B), (E, 2, E), (F, 2, B)	**(C, 4, B), (D, 7, E), (C, 3, B)**

There are two entries to reach node C, so keeping the lowest cost entry removes (C, 4, B) from the tentative list. Look at all the remaining entries on the tentative list and put the one with the lowest cost onto the confirmed list.

Confirmed:	**Tentative:**
(A, 0, -), (B, 1, B), (E, 2, E), (F, 2, B), (C, 3, B)	**(D, 7, E)**

Step 6:

Now use the link state packets for the node just added to the confirmed list - in this case node C. Each link state packet for Node C is used to add more entries to the tentative list. To add these new entries, we combine the entry just added to the confirmed list with each of its link state packets.

- The first entry is for node B which is already on the confirmed list, so this link state is ignored.
- The next entry is for node F which is already on the confirmed list, so this link state is ignored
- The next entry is (D, 2), so combining (C, 3, B) and (D, 2), we determine that node A can reach node D with a cost of 2 + 3 = 5 with next hop of node B. Therefore (D, 5, B) is added to the tentative list.

Confirmed:	**Tentative:**
(A, 0, -), (B, 1, B), (E, 2, E), (F, 2, B), (C, 3, B)	**(D, 7, E), (D, 5, B)**

There are two entries to reach node D, so keeping the lowest cost entry removes (D, 7, E) from the tentative list. Look at all the remaining entries on the tentative list and put the one with the lowest cost onto the confirmed list.

Confirmed:	**Tentative:**
(A, 0, -), (B, 1, B), (E, 2, E), (F, 2, B), (C, 3, B), (D, 5, B)	**----**

Step 7:

Now use the link state packets for the node just added to the confirmed list - in this case node D. Each link state packet for Node D has a destination node that is already on the confirmed

list, so there are no entries put on the tentative list. The tentative list is empty, so the algorithm is complete. The resulting routing table for node A is:

Node A Routing Table

Destination	*Cost*	*NextHop*
B	1	B
C	3	B
D	5	B
E	2	E
F	2	B