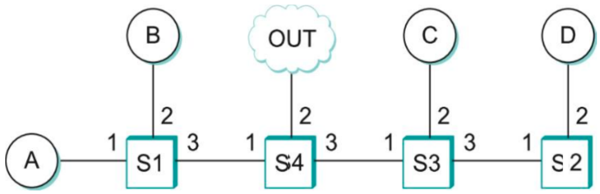
1. Give forwarding tables for switches S1-S4 in the figure below. Each switch should have a "default" routing entry, chosen to forward packets with unrecognized destination addresses toward OUT. Any specific-destination table entries duplicated by the default entry should then be eliminated.



S1

A	1
B	2
Default	3

S2

D	2
Default	1

S3

C	2
D	3
Default	1

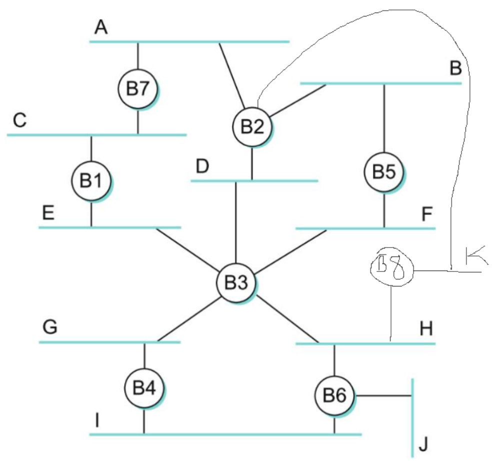
S4

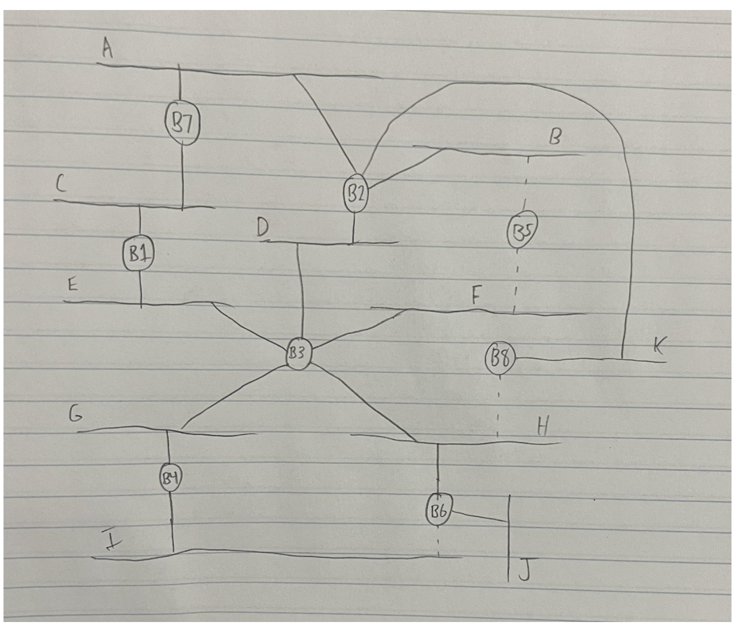
A	1
B	1
C	3
D	3
Default	2

2. The virtual circuit mechanism described in class assumes that each link is point-to-point. Extend the forwarding algorithm to work in the case that links are both point-to-point and shared-media connections such as Ethernet.

In addition to storing the port number in the forwarding table, we will have to store the mac address of the given host. This is because one port is no longer uniquely tied to a single host as each link could have multiple hosts tapped into it. We would also need a method to share the link fairly among the hosts that are tapped into the shared media connection.

3. Given the extended LAN shown below, indicate which ports are not selected by the spanning tree algorithm.





Dotted lines indicate non selected ports

4. Suppose the packet fragments below all pass through another router onto a link with an MTU of 350 bytes, not counting the link header. Show the fragments produced. If the packet were originally fragmented for this MTU, how many fragments would be produced?

Because the MTU is not evenly divisible by 8 due to the offset, we will have to make our largest packet size be the closest smaller number to 350 that is evenly divisible by 8. We would then have size = 344 and the next offset be 43 (because 344/8=43).

Offset = 0
Size = 344 bytes
m=1, ident=x

Offset = 43
Size = 168 bytes
m=1, ident=x

Offset = 64
Size = 344 bytes
m=1, ident=x

Offset = 107
Size = 168 bytes
m=1, ident=x

Offset = 128
Size = 344 bytes
m=1, ident=x

Offset = 192
Size = 32 bytes
m=0, ident=x

If the packet were originally fragmented for this MTU, how many fragments would be produced?

Once again, the MTU is effectively 344, not 350 because of the limitations imposed by the offset.

Num fragments = ceiling((512+512+376)/344) = 5

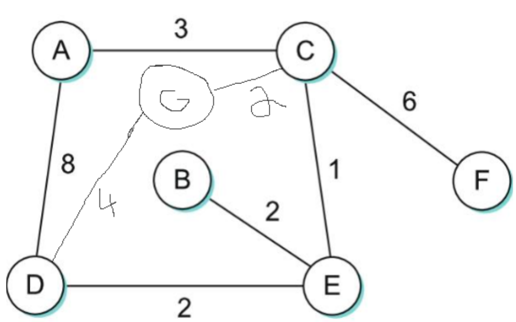
5. Ch 3 #43 Suppose an IP implementation adheres literally to the following algorithm on receipt of a packet, P, destined for IP address D:

a) Assuming you do not already have D in the cache, If a bunch of packets destined for D come in at once you will spam a bunch of ARP requests because the first one will not have come back yet. This will waste network resources. To fix this you could have a flag set for if there is an in transit ARP packet for the packet before you try to send another one.

b) if ([Ethernet address for D is in ARP cache])
        [send P]
    else if ([D ARP is in transit])
        [put P into a queue until the response comes back]
    else
        [send out an ARP query for D]

c) This depends on how many hosts there are and how big the cache is. If the cache is small compared to the number of hosts then this will make it so that most packets are dropped. The only packets that will get through are the ones where they come shortly after another packet was heading to the same destination.

6. For the network below, show how the (a) RIP and (b) link-state algorithm builds the routing table for node D.



a)
D table - first round

Destination	Cost	NextHop
A	8	A
B	∞	-
C	∞	-
E	2	E

F	∞	-
G	4	G

Second round
E tells D it can get to B with cost 2
E tells D it can get to C with cost 1

Destination	Cost	NextHop
A	8	A
B	4	E
C	3	E
E	2	E
F	∞	-
G	4	G

Third Round
E will tell D it can get to F with cost 7

Destination	Cost	NextHop
A	8	A
B	4	E
C	3	E
E	2	E
F	9	E
G	4	G

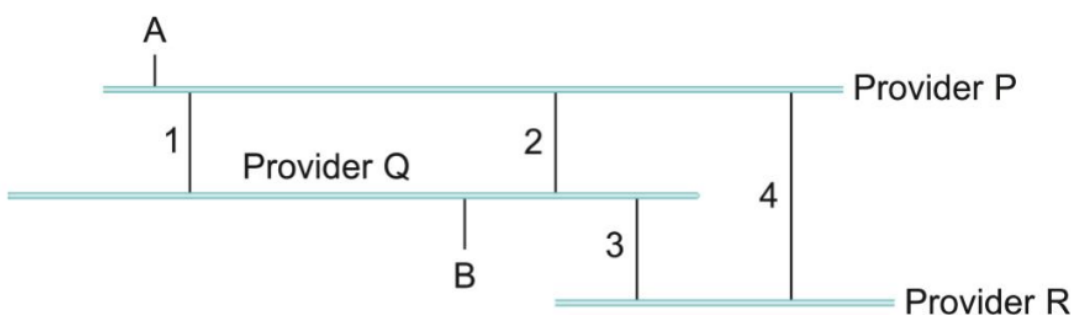
b)

Link State for D

Step	Confirmed	Tentative	Comments
1	(D,0,-)		Start off with just D itself.

2	(D,0,-)	(A,8,A), (G,4,G), (E,2,E)	Add Ds neightbors to the tentative group.
3	(D,0,-), (E,2,E)	(A,8,A), (G,4,G)	Pick the smallest cost node and add it to the confirmed group.
4	(D,0,-), (E,2,E)	(A,8,A), (G,4,G), (B,4,E), (C,3,E)	Add all of Es neighbors to the tentative stack if they are not already there or better than the current cost.
5	(D,0,-), (E,2,E), (C,3,E)	(A,8,A), (G,4,G), (B,4,E)	Pick the smallest cost node and add it to the confirmed group.
6	(D,0,-), (E,2,E), (C,3,E)	(G,4,G), (B,4,E), (A,6,E), (F,9,E)	Add all of Cs neighbors to the tentative stack if they are not already there or better than the current cost.
7	(D,0,-), (E,2,E), (C,3,E), (G,4,G)	(B,4,E), (A,6,E), (F,9,E)	Pop G off tentative priority queue since its least cost
8	(D,0,-), (E,2,E), (C,3,E), (G,4,G), (B,4,E)	(A,6,E), (F,9,E)	Pop B off tentative priority queue since its least cost
9	(D,0,-), (E,2,E), (C,3,E), (G,4,G), (B,4,E), (A,6,E)	(F,9,E)	Pop A off tentative priority queue since its least cost
10	(D,0,-), (E,2,E), (C,3,E), (G,4,G), (B,4,E), (A,6,E), (F,9,E)		Pop F off tentative priority queue since its least cost

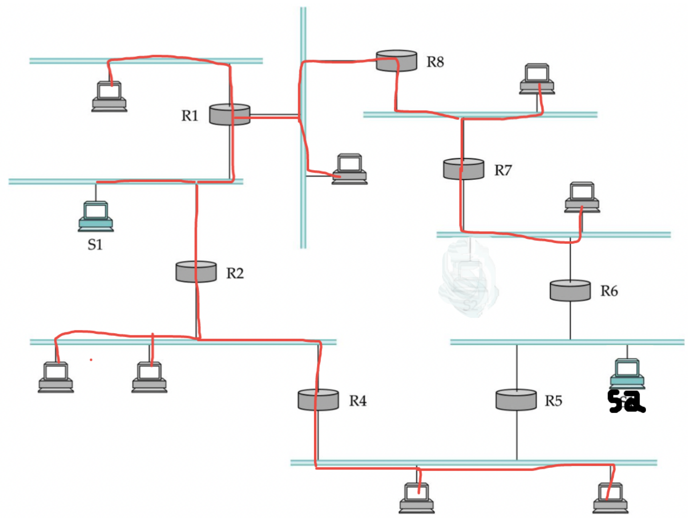
7.



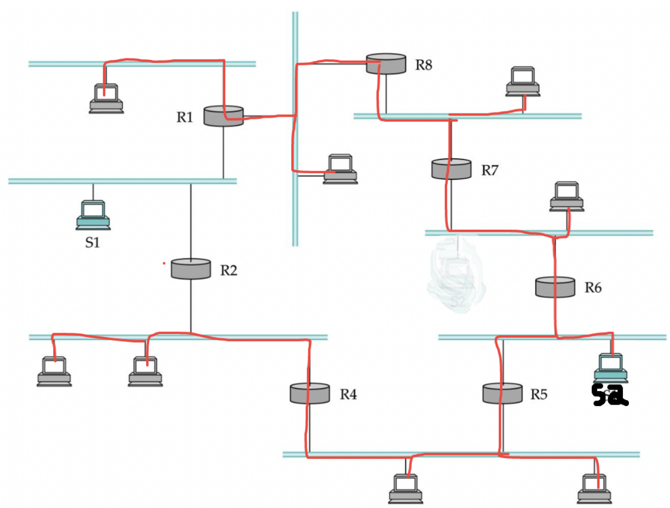
a) There are 3 different paths to get from provider Q to provider P. It could go through just 1, just 2, or 3 and then 4.
b) A to B will take link 1. B to A will take link 2.

c) It could adopt the policy that it must take the closest link to the destination (assuming that it is possible for Q to know where the links are compared to the destination). An alternative is that Q adopts the policy that outbound traffic is routed to the farthest link to the destination's provider (assuming that there are no other links not shown in the diagram above on Q).
d) It could adopt the policy that it prefers all outbound traffic to go through R if possible.

8.



a)



b)