CS 425: Computer Networks **Jan-Apr 2020**

Lecture 13: Computer Networks – February 14, 2020

Lecturer: Swaprava Nath Scribe(s): Abhishek Saini, Mayank Sharma, Mohit K. Singh, Pranjal P. Lal

In the last lecture, we discussed

1. 802.11 Protocol
2. Switching

In this lecture, we will discuss two algorithms used in switches named

1. Backward Learning Algorithm
2. Decentralized Spanning Tree Algorithm

13.1 Backward Learning Algorithm

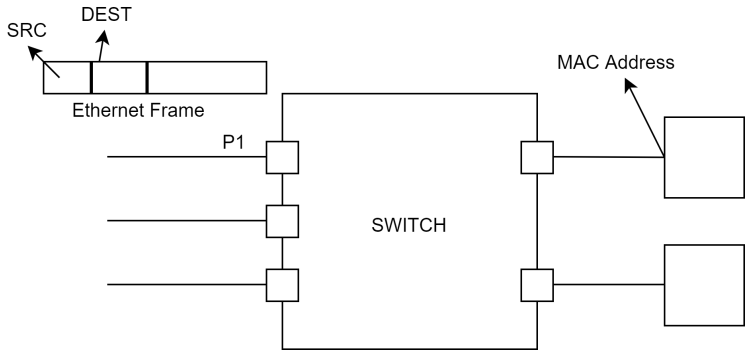

Figure 13.1: Switch

Question : How does the switch learn which port is connected to which host?
Observation : Port P_1 is connected to the source address

The Backward Learning Algorithm creates a mapping of ports and MAC addresses. The algorithm works in three steps

1. To fill the table, it looks at the source address of the received frame and creates a mapping from the source address to the port on which the frame is received
2. To forward the data to the correct destination, it forwards the data to the port to which the destination is connected
3. If the destination is not found in the mapping, the message is broadcast to all ports.

The mapping keeps on changing as devices are added or removed from different ports.

Example: Consider the switch given in the figure

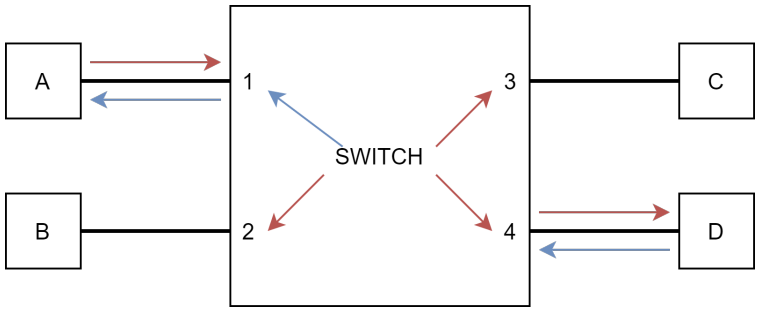

Figure 13.2: Example

When data is sent from A $\longrightarrow$ D (red arrow):

1. Data will get broadcast because the table is initially empty and port number of D is not known.
2. By looking at the source of the data packet the mapping (A,1), where A is the MAC address of the host, is added to the table.

Now, if data is sent from D $\longrightarrow$ A (blue arrow):

1. Since the mapping (A,1) is present, the switch knows that A is connected to port number 1 and the data packet will only be sent to port 1.

13.1.1 Backward Learning with Multiple Switches

Example: Consider the following example which contains two switches and a hub

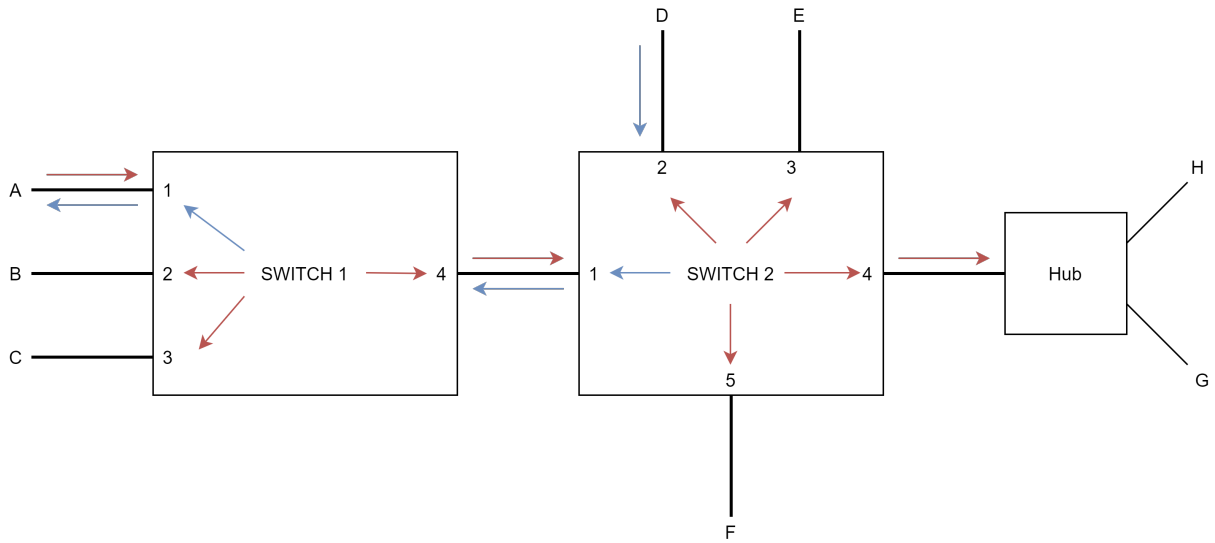

Figure 13.3: Multiple Switches

When data is sent from A $\longrightarrow$ D (red arrow):

1. Data will get broadcast by Switch 1 because its mapping table is initially empty and port number of D is not known.
2. Switch 1 will add the mapping (A,1) to its table
3. Switch 2 will receive the data packet from port 4 and will add the mapping (A,1) to its own mapping table
4. Since switch 2 does not have a mapping for destination D, it will again broadcast the packet to all ports and data will be received by D

In this process Switch 1 has added the mapping (A,1) to its table and Switch 2 has added the mapping (A,) to its table. Hence, if a data packet is now sent from D $\longrightarrow$ A (blue arrow):

1. Switch 2 will send the packet to port 1 as mapping for A is present in the table
2. Switch 1 will receive the packet from port 4 and will forward it to port 1

Limitations: The assumption in this method is that there will not be any loops in the network. Consider the example in fig 4, where a message needs to be sent from node A of the CS department to node D of the ME department. For simplicity, we assume the departments to be single switches.

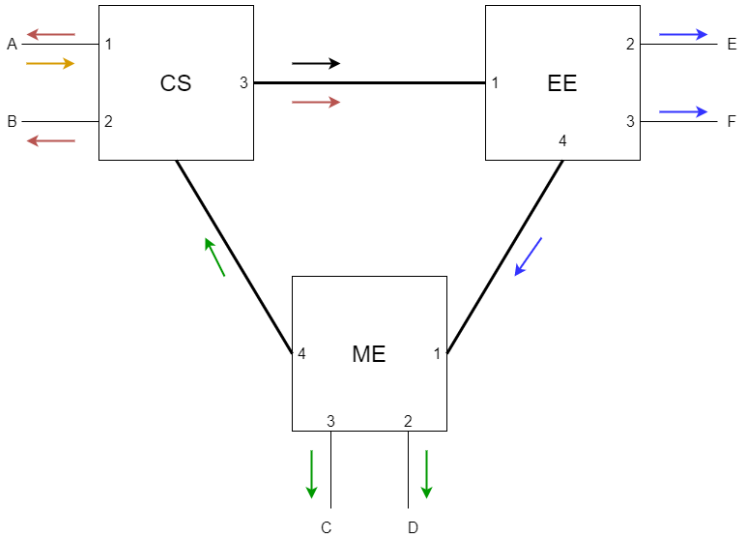

Figure 13.4: Switches in Loop

The flow will be as follows:

1. The signal from A travels to CS switch. (Orange arrow)
2. Since the CS switch does not know the address of D, the signal is broadcast to both EE and ME switches. (Black arrow, only one is shown to avoid cluttering)
3. The EE switch again broadcasts it to all connected nodes namely, E, F and the ME switch (Blue arrows).

4. The ME switch does the same and message reaches D. However, since the ME switch does not know D's address beforehand, it also broadcasts the signal to CS. (Green arrows)
5. The CS switch still does not know the address of D and re-broadcasts (Red arrows).
6. This forms a cycle and eventually exhausts the bandwidth.

Question : Why loops exist?

- For Redundancy/Reliability
- Human Errors, due to limited info.

Solution to this problem: Collectively find a spanning tree in the network.

Since there can be multiple spanning trees, we need to find a unique spanning tree. Also, algorithms such as Kruskal's can not be used because there is no centralized plan for the network. Hence, the challenge is to find a unique spanning tree in decentralized manner and use it for all further transmissions. One of the algorithms to achieve this goal is explained in the next section.

13.2 Decentralised Spanning Tree Algorithm, Perlman 1985

The idea is to let the switches use their local networks and through information exchange converge to one of the many possible spanning trees. The main features of the algorithm are as follows:

1. Each switch initially believes itself to be the root.
2. Each switch sends periodic updates to its neighbours containing its own address, root's address and hop distance to root.

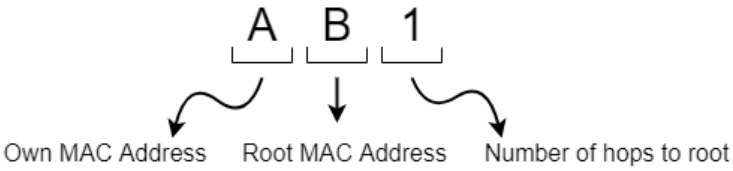

Figure 13.5: Periodic updates to neighbour nodes

3. Each switch listens to its neighbour's messages and updates root and path information as per these rules.
   - The roots with shorter distances are favoured.
   - All ties are settled by favouring the root with the lowest MAC address.
   - In case of multiple connections between two switches the port with the lower port ID is chosen.

Claim Each node running this algorithm in a distributed manner converges to a unique spanning tree.

Example: Consider the following network for better understanding of the algorithm. The colors in messages correspond the root chosen in it,(ex. B**A**0 has color of **A**.), where the letters A-F represent the MAC addresses of switches in lexical order.

In the first round (fig 13.6), all the nodes consider themselves to be the root and communicate the same to their neighbours.

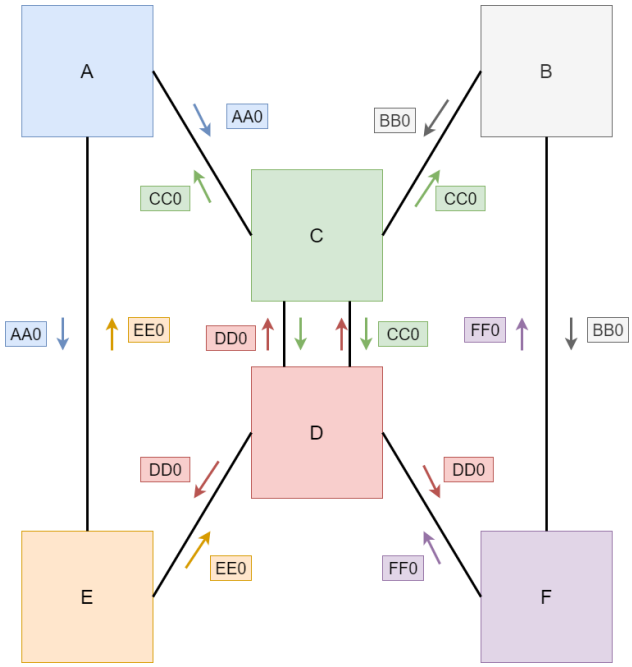

Figure 13.6: DST Algorithm: First Round

After receiving the messages from neighbours in the first round, switches C and E favour A to be the root as it has a lower MAC address. The same happens for D and F, which choose C and B respectively as their root. In case of D, the port number with lower port ID is chosen for connection to C. The switches send this updated information in the next round.

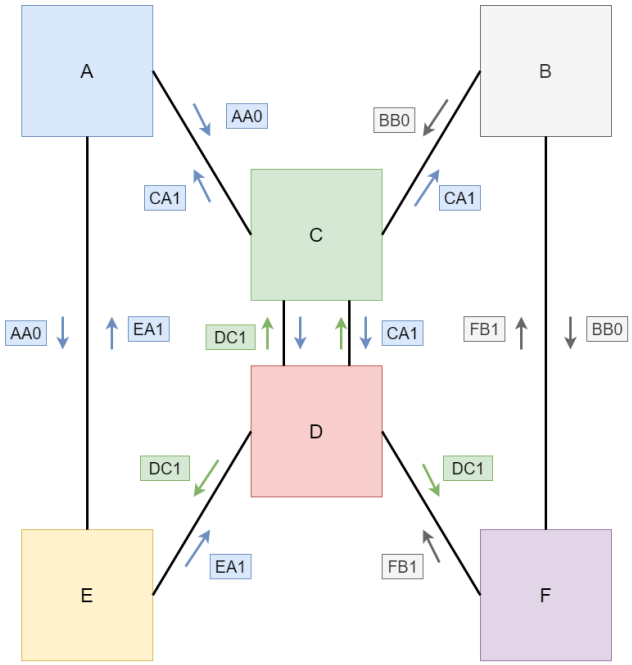

Figure 13.7: DST Algorithm: Second Round

After the second round (fig 13.7), B receives the messages CA1 and FB1 from C and F respectively and favours A as the root as it has lower MAC address. Similarly, D also favours A as the root, D also favours connection to A via C as C has lower MAC address than E.

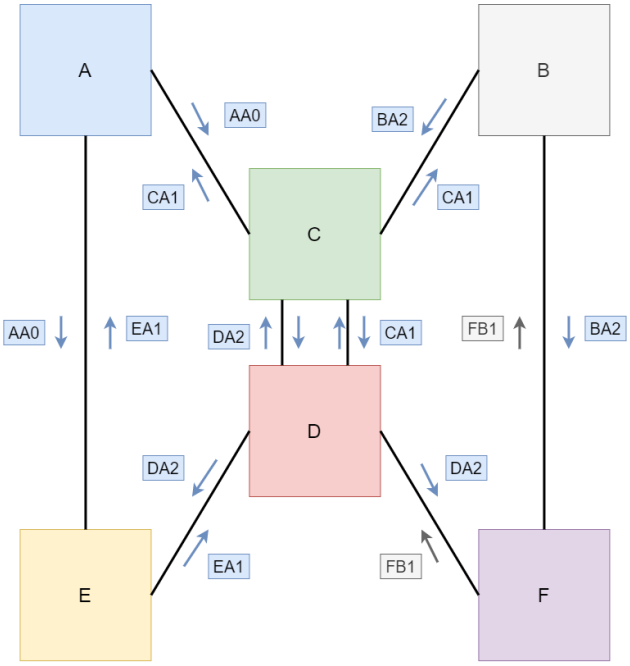

Figure 13.8: DST Algorithm: Third Round

After the third round (fig 13.8) F finally updates its root to A and all the switches converge to a common root and hence a unique spanning tree (fig 13.10) is obtained.

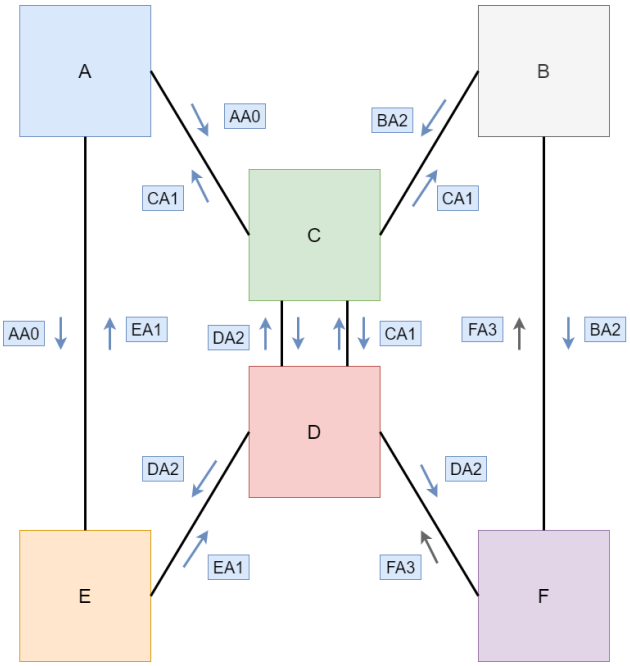

Figure 13.9: DST Algorithm: Fourth Round

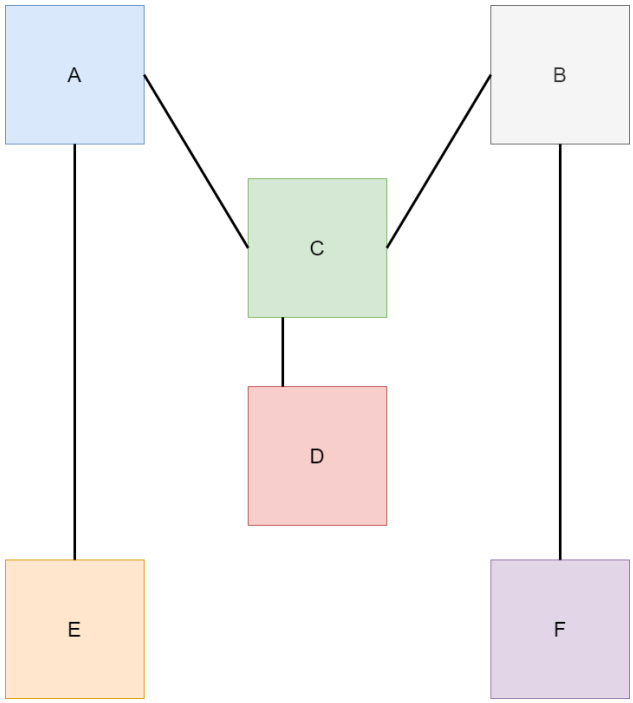

Figure 13.10: DST Algorithm: Obtained Spanning Tree

Limitations The obtained spanning tree is not efficient in terms of data transmission. For example, data transfer from D $\longrightarrow$ F requires transmission along D $\longrightarrow$ C $\longrightarrow$ B $\longrightarrow$ F which could be accomplished by making use of direct connection D $\longrightarrow$ F.

References

[CN5] Tanenbaum, Andrew S. and Wetherall, David J., Computer Networks, 5th Ed. Prentice Hall Press, 2010

[WIKI] `https://en.wikipedia.org/wiki/Spanning_Tree_Protocol`