Tutorial 5 - Spanning Trees

Gidon Rosalki

2025-11-27

Notice: If you find any mistakes, please open an issue at `https://github.com/robomarvin1501/notes_networking`

1 Spanning Tree

Until now we have worked on a network where we have a single, large, shared channel (think like a bus from computer architecture), into which every computer on the network connects. This shared channel was a single shared collision domain. As we have seen, as the number of computers increases, so too do the number of collisions, reducing the goodput significantly. To resolve this we generally split networks into one or more *broadcast domains*. A **broadcast domain** is the set of all nodes that receive each others layer 2 broadcast frames. Each node in the domain could reach every other node via broadcast messages. A **collision domain** is who can interfere with whom on the wire. It is important to note that from now, a broadcast domain will *not* be the same as a collision domain.

Splitting the network into larger physical topologies, split by switches. Consider the following image:

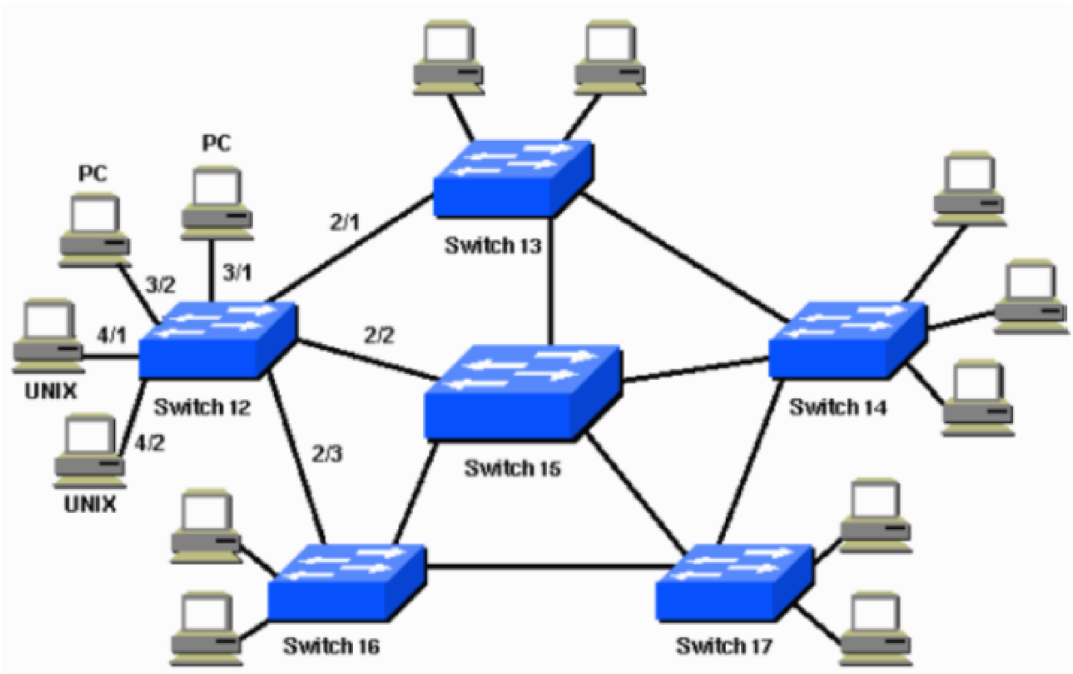

Figure 1: An example complex topology

So here, the collision domain is *only* between each switch, and either switch or node, across a single link, but every node connected to the same switch will be on the same broadcast domain.

There are 2 main ways to connect network elements, switches and hubs. Hubs are stupid. They are boxes, with many ports, and you connect a computer to each port. Whenever they receive a packet on a given port, they rebroadcast it on all other ports. They do not keep state, they do not cache messages to be sent when they will not collide. They are thus one very large collision domain.

On the other hand, switches will retain state, cache messages to avoid collisions, and move packets in a smarter way, such that packets only intended for one recipient will only go to that recipient. They thus split their connected components to many individual collision domains.

1.1 Switches

A switch allows any two connected nodes to communicate with each other, usually without collisions. It can store some packets in case of contention, and it learns the output ports upon which to send each incoming packet (based on previous incoming MAC address). When a node sends a frame, it contains the source, and destination MAC addresses. Whenever a switch receives a packet on one of its ports, it saves the pair (SRC MAC, IN PORT) in a table, and if there is an entry (DST MAC, PORT k) in the table, then it sends it through port k, unless k is the incoming port. If

it does not have that entry, then it sends it on all the ports (aside from the incoming one, since that computer clearly already has the packet).
Since we do not manually update the network topology on all connected devices every time we connect / disconnect a device, switch entries (the above table) have a timeout. If a certain MAC has not been seen for this timeout, then the entry is removed, since it may be assumed that the device has been disconnected.

So, we have a clever method for our switches to learn a topology, but this has its limitations, for example, when there is a loop in the network. When there is a loop, then the switches will be constantly relearning the origin port for packets, which is obviously *not good*. We do not want to only have one patch between 2 LANs, since having more than one path is important for redundancy. If a device / link fails, we do not want the entire network to fail as a result. However, the problems only begin when there is more than one *active* path between 2 LANs, so to fix loops we can:

1. Add a counter packets that is reduced at each switch (this counter acts as a lifetime)
2. Create a single path between any two LAN segments

We will try option 2: We would like to create a single active path between any 2 LAN segments, but we should account for the existence of multiple physical paths. To avoid loops, we will create an active topology of a tree, placed virtually on top of our physical network, such that if there is a failure, the tree can regrow a connection.

2 Spanning Tree Protocol

We ignore the hosts in the network while running this protocol (as in, the end computers, remember, switches are not considered hosts). After convergence (the tree is connected), we want:

- There to be a single root, acknowledged by all
- Each LAN segment has one **and only one** active link on the path to the root (through a designated switch)
- Each switch port not taking part in the active tree will be disabled (but listening for changes to allow healing disconnecting/failing parts of the tree)

We will assume (for this course) that each switch has its own unique ID (this should always be the case in real life, but sometimes some companies are [REDACTED]). We use special data frames called HELLO (or BPDUs - Bridge Protocol Data Units) which contain:

- Root ID (RID) - What this switch currently thinks is the root ID of the tree (initially, its own ID)
- The distance to root (DTR) / also known as the root path cost, which is the cost of the path from this switch, to the root
- The switch ID (SID) of this switch

Every node performs the following algorithm **independently** of other nodes:

- Send HELLO frames through all ports. If it gets a HELLO frame, update the RID and DTR if necessary
- If it was expecting a HELLO, but did not receive during a defined time period, compute a new RID

A node will select the switch with the lowest RID as its root. Based on a new HELLO packet, the switch will update its path to the root (i.e., which ports will remain active, and which ones will be disabled) using the following logic in descending order of priority:

1. Root ID (which root it believes in)
2. Shortest distance to the root
3. Lower SID
4. Lower port number (this is a tie breaker for when there are 2 physical links of the same cost to the same location)

If everything proceeds as it should (no failures), then every switch will have the same root port.

From this protocol, we have 3 port types:

- Root Port (RP) / Parent Port (PP): This connects a switch towards the root, is active, and there is only **one** per switch
- Forwarding port (FP) / Designated Port: A port that forwards traffic for its LAN segment, as in, a port that is connected to the PP for another switch (there is one designated port per segment)
- Blocked Port (BP): The switch will not forward any data from/to those ports. However, the switch will listen to new HELLO messages on those ports in order to potentially heal the network

RPs, and FPs will be part of the spanning tree. It is important to note that a HELLO packet never leaves the LAN segment on which it was transmitted, since it is only important for the two most directly connected switches.

Behold, an example of a spanning tree:

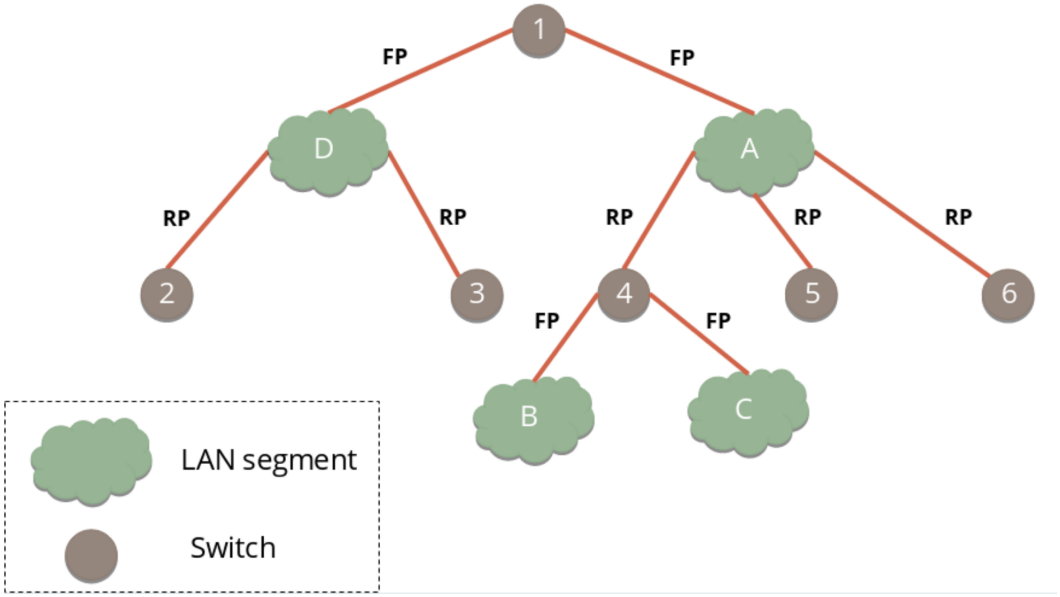

Figure 2: Spanning tree example

What happens if different links have different (known) bandwidths? What needs to change?

- We would prefer using high bandwidth links over low bandwidth ones
- We would need to change the algorithm into a weighted version
- Would need to select high bandwidth links before low bandwidth ones

This modification represents an interesting thought experiment, and a reasonable example of a modification that may be made for an exam question.

2.1 Example

Let us create the spanning tree for a given network (very common exam question). Consider the following network:

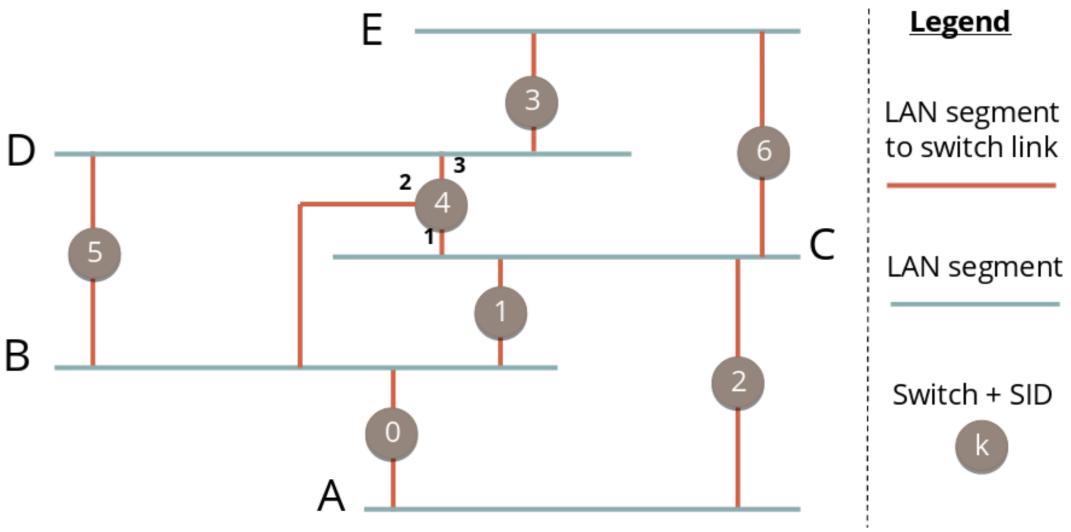

Figure 3: Example network

We begin with switch 0 sending a HELLO packet on segments A, and B. Switch 2 will see RID 0 (since 0 sent its own SID as the RID), and since this is lower than its SID (and currently RID) of 2, so it updates its RID to 0, its DTR to 1, and the root port to point towards switch 0.
This may also be seen in tabular form:

SID	RID	DTR
1	1	0
2	2	0
3	3	0
4	4	0
5	5	0
6	6	0

Table 1:

Everyone thinks that they are the root at the beginning. The above steps update the table

SID	RID	DTR
0	0	0
1	1	0
2	0	1
3	3	0
4	4	0
5	5	0
6	6	0

Table 2:

Following on from this, switches 1, 4, and 5 do the same as 2 did above:

SID	RID	DTR
0	0	0
1	0	1
2	0	1
3	3	0
4	0	1
5	0	1
6	6	0

Table 3:

So in order for segments A, and B to send to the root, they will send to switch 0. Those connections will now be the designated / forwarding port. Remember, the root port is where a **switch** sends towards the root, and the forwarding port is where a **LAN** sends towards a root.

This has left us with the following network:

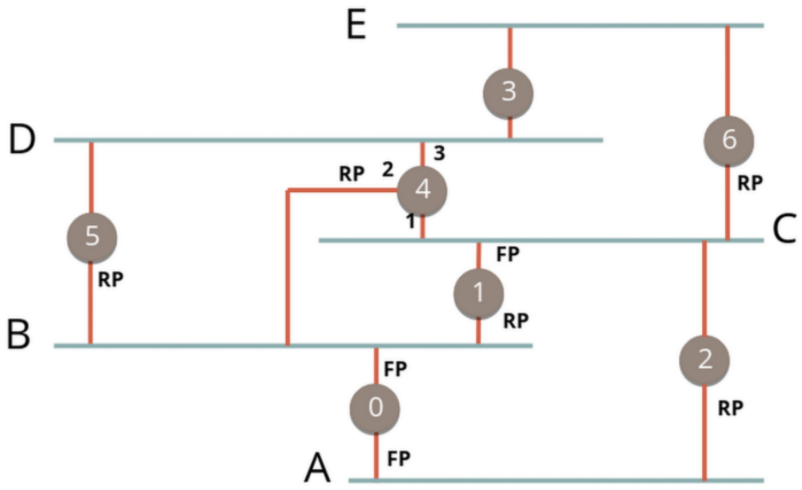

Figure 4:

Now, switches 1, 2, 4, and 6 send HELLO packets on C (in reality, all LANs would have these packets sent on them in parallel, but it is easier to calculate 1 at a time). Switches 1, 2, and 4 have already learned that 0 is the root, with a cost of 1, so they send on C (RID=0, DTR=1, SID $\in \{1, 2, 4\}$). So now, 6 updates its RID from 6 to 0, and DTR to 2:

SID	RID	DTR
0	0	0
1	0	1
2	0	1
3	3	0
4	0	1
5	0	1
6	0	2

Table 4:

On LAN C, to pick the designated port it evaluates the cost to root, the switch ID, and the port ID. Since switches 1, 2, and 4 tie with the lowest RID, we move to the next level which is the SID. SID 1 is the lowest, so switch 1 is the forwarding port for LAN C.

Now, switches 3, 4, and 5 send HELLO on LAN D. 3 updates its RID to 0, and DTR to 2:

SID	RID	DTR
0	0	0
1	0	1
2	0	1
3	0	2
4	0	1
5	0	1
6	0	2

Table 5:

As a result, switch 4 is the designated on D. It is designated instead of 5, despite them having the same DTR, because 4 has a lower SID.

Now, switches 3 and 6 send HELLO on LAN E. There are no switch updates to happen, aside from switch 3 being the designated on LAN E, since its SID is lower than that of switch 6.

We can now disable all the links that do not contain a Root Port, or a Forwarding Port, and then all the switches that do not connect at least 2 LANs (as in, the switch needs at least 1 Forwarding Port, and at least 1 Root Port). We are now left with the final (virtual) network topology tree:

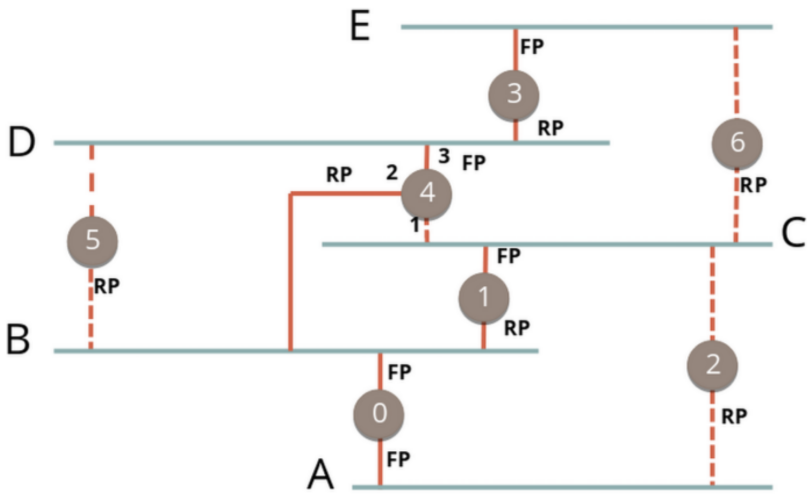

Figure 5:

We may also try an alternative method, which is less demonstrative of the realities, but is easier for us humans. Let us begin just with the LANs:

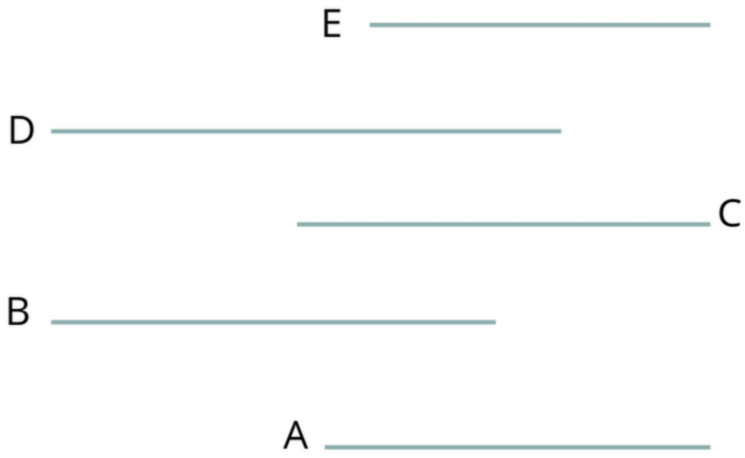

Figure 6:

We now add the root switch, and its connections. In this case, switch 0, with FPs from both A, and B. We may now go to the next unconnected LAN (C), and find its lowest DTR. Since it is connected to 0 through both switches 1, and 2, both with a DTR of 1, we connect in switch 1 with the corresponding FP, and RP, and disconnect 2:

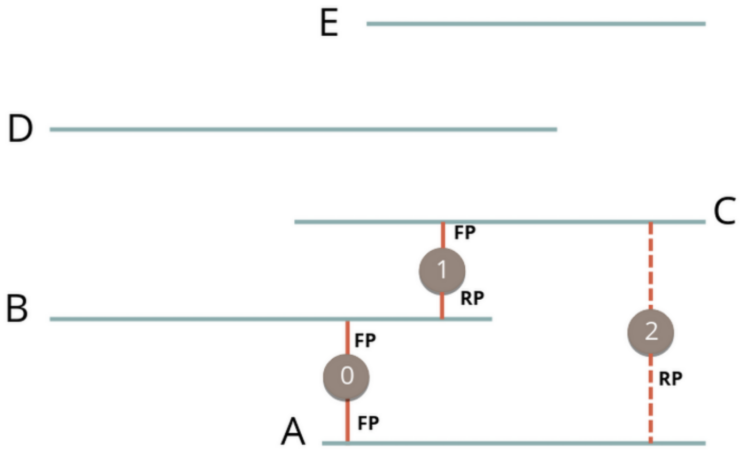

Figure 7:

We may repeat this this by D, and then E by the same path, and will wind up with the same result. Either method may be used in an exam / homework, depending on what the individual finds the easiest.

Let us now consider, what happens if switch 0 crashes?

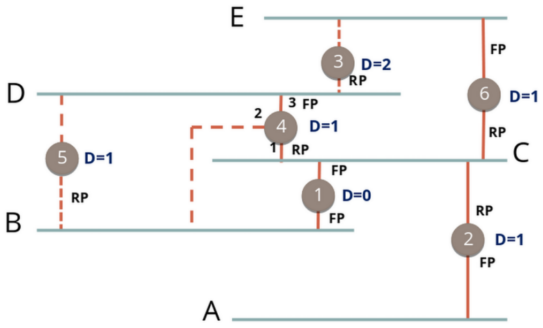

Figure 8:

Now, switch 1 will become the root of the tree, switch 2 replaces switch 0 on LAN A, switch 6 replaces 3 on LAN E, and switch 4 now uses port 1 as its RP, instead of port 2, since it is the same distance, but a lower port number.

If instead switch 1 crashes, then switch 2 will replace it, as no HELLO was observed on LAN C. These kinds of changes are also common questions, and the best way to resolve them is by intuitive understanding, probably gained form doing lots of examples.

3 Questions

3.1 Part 1

The network is composed from:

- LAN segments (horizontal lines): marked with capital letters
- End-nodes (triangles): marked with small letters
- Switches (squares): marked with numbers (those are their unique IDs)

MAC addresses of end nodes are named by their ID (e.g., station a: MAC(a)). MAC addresses of nodes that are connected to more than one LAN are marked by the node's ID and the PORT number (e.g., MAC(b,3)).

Run the STP algorithm when the dotted lines (connected to d) are disconnected. Mark the roots, disabled ports / switches, and parent (root) ports, if they exist.

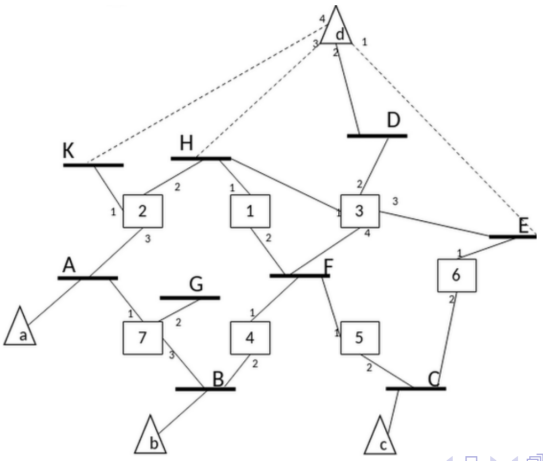

Figure 9:

I am not going to include here the step by step drawing of a diagram, because frankly, drawing in LaTeX is quite a lot of effort. I will instead describe the steps, and you may draw your own diagram as you follow along. The final answer will be included below. If we begin from the LANs, let us pick the root switch to be 1 (it has the lowest ID), which is between LANs H and F. If we now connect D, then switch 3 will be connected to the root through LAN H, since this is the lowest port ID. To now connect in LAN 3, we see that it is also connected through switch E, since this is its shortest path to the root. LAN K is simply connected through switch 2, to LAN H, and A will also connect through switch 2. LAN G only has one connection, to switch 7, so it is connected there. Switch 7 will now connect to LAN A (and thus switch 2). LAN C will connect to switch 5, and finally LAN B will connect to switch 4, since this is its shortest path to the root. So below is the resultant network topology:

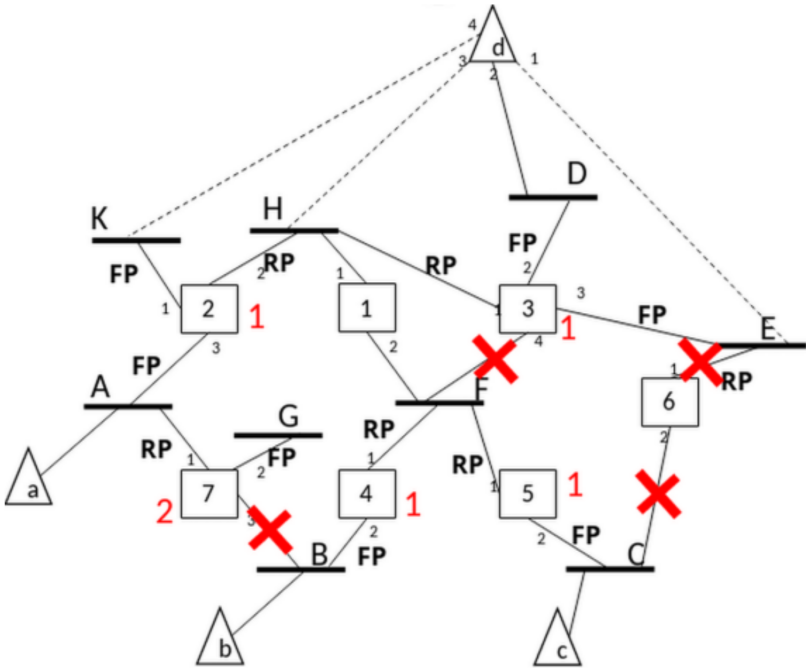

Figure 10:

3.2 Part 2

Should we run the STP algorithm again, but this time where the dotted lines are connected, then there will be no change.

3.3 Part 3

Assume that the network has been working for a long time and that all end-nodes have sent a message to all other end-nodes over all ports.

Fill in the switching table of switch 1:

MAC Address	Port
MAC(a)	1
MAC(b)	2
MAC(c)	2
MAC(d, 1)	1
MAC(d, 2)	1
MAC(d, 3)	1
MAC(d, 4)	1

Table 6:

Fill in the switching table of switch 2:

MAC Address	Port
MAC(a)	3
MAC(b)	2
MAC(c)	2
MAC(d, 1)	2
MAC(d, 2)	2
MAC(d, 3)	2
MAC(d, 4)	1

Table 7: