Computer Networks

Tutorial 5:

Learning Switch Tables
and Spanning Tree Algorithm

Scope of This Tutorial

- Learning switch tables
- Spanning tree algorithm

Learning Switch Tables – an Example

Suppose we have Ethernet switches S1 through S3 arranged as below. Each switch uses the learning algorithm. All forwarding tables are initially empty.

```
S1 — S2 — S3 — D
|    |    |
A    B    C
```

The communication scenario is as follows.

- A sends to B ⇒ S1, S2, S3 broadcast the frame and learn A address
- B replies to A ⇒ S1 & S2 forward the frame and learn B address
- C sends to B ⇒ S3 broadcasts the frame, S2 forwards it, S3 & S2 learn C address
- C sends to D ⇒ S1, S2, S3 broadcast the frame, S1 learns C address

The result is:

switch	known destinations
S1	A, B, C
S2	A, B, C
S3	A, C

Exercise 1

Suppose we have Ethernet switches S1 through S3 arranged as below. Each switch uses the learning algorithm. All forwarding tables are initially empty.

```
                B
                |
               S4
                |
A  —  S1  —  S2  —  S3  —  C
                |
                D
```

The communication scenario is as follows.

- A sends to D
- D sends to A
- A sends to B
- B sends to D

What are the contents of forwarding tables?

Spanning Tree Algorithm

The goal is to disable redundant (cyclical) paths

If the outage should partition the network into two pieces, both pieces will build spanning trees

Every switch has an ID – its smallest Ethernet address
Every interface is numbered
The switches first elect a root node – the one with the smallest ID

Switches send periodically *bridge protocol data units* (BPDUs) – hosts do not send them!
to the Ethernet multicast address 01:80:c2:00:00:00
BPDU contains: sender's ID, supposed root ID, path cost to that root
If a switch sees a new root candidate, it sends BPDUs on all interfaces

Once this process has stabilized, each switch knows

- its own path to the root
- which of its ports any neighbouring switches will be using to reach the root
- for each port, its directly connected neighbouring switches

Next every switch prunes its interfaces

1. It enables the port via which it reaches the root
2. It enables any of its ports that neighbouring switches use to reach the root
3. If a remaining port connects to a segment to which other “segment-neighbour” switches connect as well, the port is enabled if the switch has the minimum cost to the root among those segment-neighbours, or, if a tie, the smallest ID among those neighbours, or, if two ports are tied, the port with the smaller ID.
4. If a port has no directly connected switch-neighbours, it presumably connects to a host or segment, and the port is enabled.

What would happen if switches without STP are connected in a redundant topology?

Example

There is a network:

```
S1 — S2 — S3
|     |     |
S4 — S5 — S6
```

What is the spanning tree? Assume that ID of switch S_i is ‘i’.

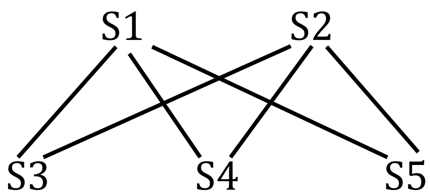
Solution:

S1 has the lowest ID ⇒ it is the root
S2 & S4 enable interfaces to the root – rule 1
S1 enables interfaces to them – rule 2
S3 has unique lower cost to S2 – rule 3
S5 has to paths of equal costs, lower ID belongs to S2 – rule 3
the same for S6, which selects S3 – rule 3

```
S1 — S2 — S3
|     |     |
S4    S5    S6
```

Exercise 2

A. There is a network:



What is the spanning tree? Assume that ID of switch S_i is 'i'.

Solution:

B. Do the same but assuming S4 has ID 0, and so will be the root.