Midterm

Chapter 8, sections 1-4

- 8.1: Definition of automaton
- 8.2: Composition
  - Understand theorems, but I will not require you to state or prove them.
- 8.3: Fairness
  - Understand theorems
- 8.4: This is just a hint, you should read it

Section 8.5 pp 216-219

- Invariant assertions
- Safety properties
- Liveness properties

Section 8.6 pp 228-229

- Complexity measures – later today

Section 15.1.1

Section 15.2

- See also 4.1.2 for the algorithm idea

Sections 15.2-15.3

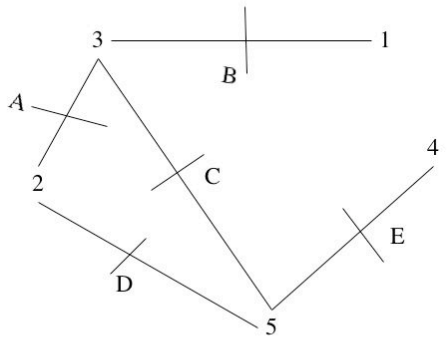
Lecture Notes & Homeworks

Spanning Tree

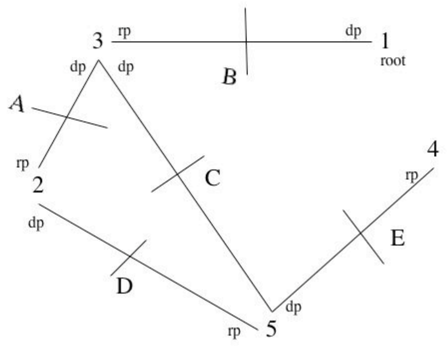
Note that BFS computes a spanning tree (the parent pointers identify the edges). How many edges are there in a spanning tree?

The Cisco STP uses this.

Example 1

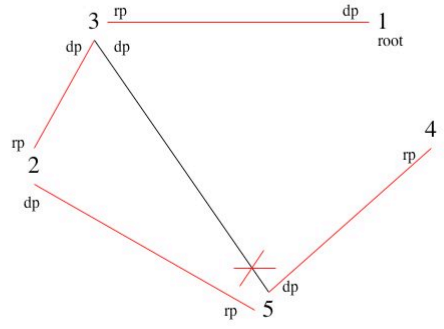


Example 1



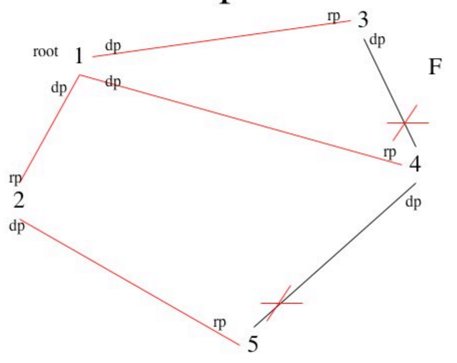
Assume ports are numbered clockwise starting at 12

Example 1



Assume ports are numbered clockwise starting at 12

Example 2



Assume ports are numbered clockwise starting at 12

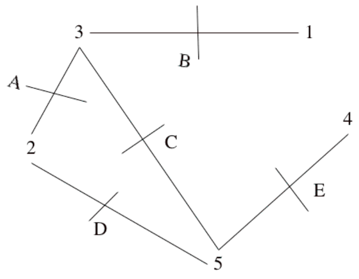
What does the Cisco STP compute

When the protocol stabilizes, the state should be as follows:

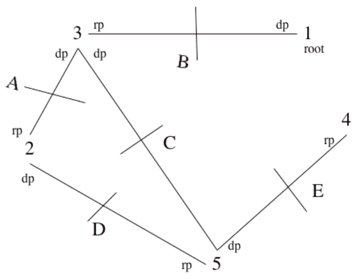
1) **Root bridge**: The process (switch) with the lowest MAC address (or lowest combined priority+MAC address) is the root. This uses a leader election algorithm.
2) **Root ports**: Each bridge has one root port. The root port on each bridge is the port of the bridge with the smallest distance from the root. If two ports are equidistant from the root, then the one going to the bridge with the lower MAC address is the root port. This uses a breadth-first search, if we assume rounds; however, if the network is asyncrhonous, it's more complicated.
3) **Designated ports**: Each network segment (connecting bridges) has a designated port. Messages put on that network segment are forwarded to the rest of the network through the designated port. The designated port is on the bridge closest to the root. If there is a tie, it is on the bridge with the lowest MAC address. If the bridge selected by this rule has multiple ports on a network, it is the port with the lowest id.

Examples

Example 1

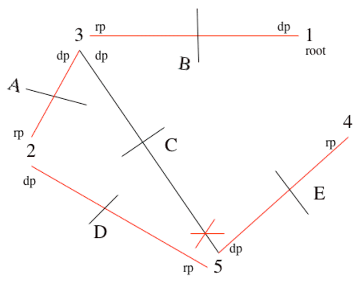


Example 1



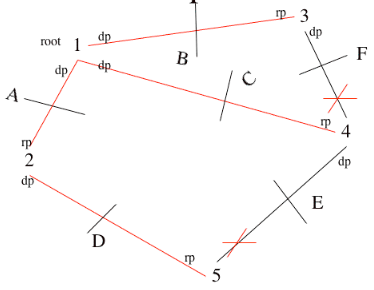
Assume ports are numbered clockwise starting at 12

Example 1



Assume ports are numbered clockwise starting at 12

Example 2



Assume ports are numbered clockwise starting at 12

Cisco Version

All processes send a BPDU (Basic Protocol Data Unit) at each round (actually, default is every 2 seconds – but we will describe this as a synchronous algorithm, running in rounds). The BPDU contains the id (MAC address) of the sending process, the id of the process it thinks is the root, and the distance from the sending process to its presumed root.

id:Int
id:Int
cost:Int

When a process receives a BPDU, it compares the ID of the root (designated by the neighbor that sent the BPDU) to the local value for the ID of the root. If the new root ID is lower, it replaces its local root ID with the new one and adds one to the cost in the incoming BPDU and makes that its cost to the root. If it receives different BPDU's having different roots, it uses the "best," i.e., the one with the lowest root ID and the lowest distance (if root Ids are the same).
Finally it designates the port to which the sending neighbor is connected as the root port. (Effectively choosing its parent)

Note that this is essentially BFS.

Designated ports are then chosen: for each network it is on, the bridge checks incoming BPDUs. If its own cost is lowest, or if it is tied with another bridge on cost and its id is smaller, it is the designated bridge, and its port on that network segment is the designated port. If it has multiple ports on the network segment, it chooses the one with the lowest id.

Complexity Analysis

Synchronous

Algorithm	**Rounds**	**Messages**	**Reference**
Basic leader	n	O(n^2)	Section 3.3, p 31
Halting leader	2n	O(n^2)	“
Best case message complexity	5n	O(n log n)	Section 3.4, pp 34-35
Breadth-first search	diam	|E|	Section 4.2, p 60
Terminating breadth-first search	2*diam	O(|E|)	Section 4.2, p 60

Asynchronous

ℓ : upper bound for each task

d : upper bound on time to deliver oldest message in the channel

Algorithm	**Time**	**Messages**	**Reference**
*Basic leader	O(n(ℓ+d))	O(n^2)	Section 15.1.1, pp 479-480
Halting leader	O(n(ℓ+d))	O(n^2)	Section 15.1.1, pp 479-480
Best case message complexity	O(n(ℓ+d))	O(n log n)	Sections 15.1.2-3
*Breadth-first search	diam*(ℓ+d)+ ℓ	O(|E|)	Section 15.3, p. 498
*Terminating breadth-first search	O(n*(ℓ+d)+ ℓ)	O(|E|)	Section 15.3, p. 498

The only ones with anything tricky in the reasoning are starred. For the basic leader algorithm, the problem is establishing that message queuing doesn't increase the time. There's an interesting and sophisticated inductive proof on pp 480-481.

The first tricky thing about breadth-first search is that the spanning tree built in the asynchronous case may not be the breadth-first tree. Since a node uses the first search message it receives to determine its parent, and since messages on a long path may go very fast, the path of parent points may be simple path including all nodes in the tree. In this case, the length of the path is n-1.

However, no matter how long the path of parent pointers is, the bounds on task time and channel time mean that messages traveling on the shortest (but not necessarily fastest) path arrive within time $O(diam*(\ell+d)+ \ell)$. Thus, no matter who the last node reached chooses as its parent, the first message it receives has to arrive within time $O(diam*(\ell+d)+ \ell)$.

However, if the nodes send messages back up the tree, instead of going fast they may go slowly, ie, $O(n*(\ell+d)+ \ell)$ because they take n steps.

Midterm