Lecture 6

Gidon Rosalki

2025-11-23

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

1 Interconnecting Broadcast Domains

1.1 STP

We need to solve the following, in order:

1. How to find a root switch?
2. How to compute a spanning tree of the switches?
3. How to compute a spanning tree of broadcast domains?

This protocol is part of a family of protocols, called self stabilising protocols. The objective of these is to stabilise the network to some form, which will not have it constantly changing. In our context, we want every switch to use the same tree.

1.1.1 Choosing a root switch

Assume each switch has a root identifier. Each switch remembers the lowest ID that it has seen so far, and periodically floods its root ID to all its neighbours. When it receives a flooded ID from its neighbours, it updates its root ID if necessary.

1.1.2 Compute a ST given a root

The idea is that each node finds its shortest path to the root. At each node, output the parent pointer, and distance. This is done through the distributed Bellman Ford algorithm:
Assume that there is a unique root node s. Each node will, periodically, tell all of its neighbours what is its *distance* from s. They can tell since at s, $dist\,(s) = 0$. At node v:

$$dist_v = \min_{u:(v,u)\in E} \{dist_u + 1\}$$

Mark the neighbour with the lowest distance as the parent.

Bellman Ford has some important properties:

- It works for any assignment of **non negative** link weights $w\,(u, v)$
- It works when the system operates asynchronously
- It works regardless of the initial distances

To compute the spanning tree of LAN segments, we assume that we are given a spanning tree of the switches. The idea is that each broadcast domain has at least one switch attached, but only *one* of them should forward packets. We choose the switch closest to the root, and break ties by switch ID. If then they are still tied, by the interface / port ID. Switches all listen to all distances announcements on each port. They mark the port as "designated port" **if and only if** the switch in question is the best on that port's associated broadcast domain.

Definition 1.1 (Link cost)**.** *The **link cost** is a cost associated with each port on each switch ("weight" of connection to broadcast domain). We default to 1. This cost can be aiether manually, or automatically assigned, and can be used to alter the path to the root switch.*

Definition 1.2 (Root port)**.** *Each non-root switch has a **root port**, which is the port on the path towards the root switch.*

I think this would be better called parent port, it is not the root of the tree, but rather the parent of this specific node in the tree. Consider it a parent pointer in a linked list. The root port is part of the lowest cost path towards the root switch of the tree. If port costs are equal on a switch, the port with the lowest ID becomes root port.

So from this, we see that the spanning tree has a few requirements:

- Each switch has a unique identifier
- There is a unique port identifier for all ports, on all switches

So the algorithm for the spanning tree is as follows:

- Keep sending a single message, which contains this switches' root ID, its cost to root, and its ID
- Save my_root_ID to be the smallest seen root_id so far, and save the lowest cost to root to be the smallest transmitted root_cost, plus the link cost

In short, every node publishes to its immediate connections who it is, how far it is from its root, and its ID. Every node receives this, and updates (if necessary) its parent switch to be the node with the shortest path to the root switch.

Only the root, and designated ports are active for data forwarding. All the other ports are in the blocking state, and do not forward packets. If a switch has no designated port, then it does not forward anything. This includes on the root port.
All ports always send BPDU (Bridge Protocol Data Unit, the above combination of data) messages, in order to update the tree regularly.

2 IP Networks - Interconnecting LANs

2.1 Introduction to IP addressing

We will mostly consider IPv4 in this lecture. In this scheme, an IP address is a 32 bit (split across 4 values) identifier for a host, router, and interface. An *interface* is the connection between a host / router, and the physical link. A router typically has multiple interfaces, where a host typically only has one or two interfaces. They each will have IP addresses associated with each interface. An example would be 192.168.1.178. The IP address is split into the subnets, and host, where the subnets are indicated by the higher order bits, and the host by the lower order bits. So in our example, our host 178 is on the subnet 192.168.1. The router can have many subnets, across different ports. Traffic may be passed within a single subnet trivially, but to communicate across subnets, one must involve the router.

For this kind of addressing, we use CIDR: Classless InterDomain Routing. The subnet portion of the address is of an arbitrary length, with an address format a.b.c.d/x. Here, x is the number of bits in the subnet portion of the address. In the above example of an IP address, it would be given as 192.168.1.178/24, indicating that the first 24 bits indicate the subnet, and the last 8 (the number 178) indicates the host. This x is also called the subnet mask, because that is the size of the bitwise mask we apply to get the host.

So this builds into why the internet is not just a single, huge LAN. Each switch stores a table of every MAC it knows, which given the size of the internet, would not be manageable. Instead, we use **hierarchical addressing**. This way, each ISP has a block of IP addresses, advertising to the rest of the world that any datagram whose x first bits match its blocks should be sent to it, and then handles routing from the rest of the internet into its IP addresses itself. An organisation may well rent a large block of these from an ISP. Should an organisation renting from an ISP change ISP, then the new ISP will begin advertising this address as well, rather than the organisation changing all its IPs. The method used here is longest prefix matching, and everyone routes towards the IP that matches the longest part of the prefix. ISPs get assigned blocks of IP addresses by ICANN (Internet Corporation for Assigned Names and Numbers), which handles allocating IP addresses, DNS, and assigns domains.

The IP also handles demultiplexing, where it needs to be decided where to send the packet in the end user. There is a field in the packet header called the "upper level" field. If it's TCP, then it is sent to the process handling TCP packets. Similarly for UDP, ICMP, and so on. There is a similar demultiplexing for layers 2 and 3, using the type field in Ethernet, and between layers 4 and 5 with ports. There are 2^{16} ports, allowing a lot of choice.

2.2 DHCP

One needs to *acquire* an IP address from somewhere. This can be hard-coded into a system, by its administrator, but this is liable to cause errors. What if it changes networks / subnets? What if someone else is already using this IP address? The IP will no longer be valid. To resolve this we have **DHCP**: Dynamic Host Configuration Protocol. This dynamically gets IP addresses from serves, and behaves "plug and play", like adding more switches to a network.

2.2.1 Overview

The goal is to allow a host to dynamically obtain an IP address from the network when it joins. This IP address is leased to the host, since the IP may be handed to *any* host. It can renew its lease on this address later, should it still be connected. This allows reusing addresses, since addresses should only be held when the device is connected, and active.

Upon connection, the host broadcasts the DHCP discover message. The DHCP server responds with an offer. The host requests an IP address with a "DHCP request", and the server sends it an address using a "DHCP ack" .