Your node in a mesh network

Connecting to your Raspberry Pi

1. Verify that the Raspberry Pi has the SD card in place. Power on the device and wait for a solid green light with a flashing red light.

2. Connect your computer's WiFi Client to the Raspberry Pi's WiFi Access Point:

   - SSID: `<hostname>`
   - Password: `password`

3. Once your computer is connected to the WiFi Access Point, you can access the Raspberry Pi via a Secure Shell (SSH):

   - Host: `<hostname>.local` (or `10.0.0.1`)
   - Username: `root`
   - Password: `root`

   On **macOS or Linux** using **Terminal**, enter `ssh root@example.local` followed by the password `root`.

   On **Windows** using the **PuTTY** graphical interface, enter the `Host` and select `SSH` then click `Open` to initiate an SSH session. Enter the `Username` and `Password` when prompted.

You are now connected to and able to run commands from your Raspberry Pi, but unlike last week, your Raspberry Pi is currently not connected to all the other ones in the room. Find a partner and we will manually get the two Raspberry Pis to talk over an ethernet cable. For the rest of this activity, we will use `example` to represent one node, and `example2` to represent the other.

Making a wired ethernet link

1. The Raspberry Pi has one single ethernet port for wired networking. In the operating system, it is represented by `eth0`. Look at the `eth0` ethernet interface using:

   ```
   root@example:~# networkctl status eth0
   ```

   You should see `State: no-carrier (configuring)`. This is because the ethernet port is not connected to anything. We need to build a road between the two Raspberry Pis.

2. Connect the two nodes with an ethernet cable, then run the same command again. You will find that the state has changed to `State: degraded (configuring)`. Even though we have a road (i.e. ethernet cable), there is no street addresses (i.e. IP addresses) or road signs for directions (i.e. routes).

3. We will now assign IP addresses to the `eth0` interface on the two nodes. It is important that they have different addresses for the same reason two houses should not have the same street address. Let us assign `192.168.0.1` to `example`, and `192.168.0.2` to `example2`:

   ```
   root@example:~# ip addr add 192.168.0.1/24 dev eth0
   ```

   root@example2:~# `ip addr add 192.168.0.2/24 dev eth0`

   We are assigning the IP address `192.168.0.1` to the ethernet interface of `example`, and the `/24` tells the device to route all IP traffic to and from `192.168.0.X` addresses through the `eth0` interface with that IP address. On `example2`, the command does the same thing except the traffic is routed through its `eth0` interface with IP address `192.168.0.2`.

4. Check the status of `eth0` again and you should find that its state has changed to `State: routable (configuring)` and its newly assigned IP address will also be printed.

5. Check the route to ensure the road signs are properly registered:

   root@example:~# `ip route`

   One of the lines in the output should look like `192.168.0.0/24 dev eth0 proto kernel scope link src 192.168.0.1`, which means IP traffic to and from `192.168.0.X` addresses will be routed through the `eth0` network interface with the IP address `192.168.0.1`. There are other network interfaces on your Raspberry Pi, such as the one wirelessly connecting your computer. You can see them all by running:

   root@example:~# `networkctl status`

6. Now let's try to ping each other over this new pathway you just built!

   root@example:~# `ping 192.168.0.2`

   Or, using mDNS:

   root@example:~# `ping example2.local`

   Hit `Ctrl + C` to stop the ping.

We have confirmed the two Raspberry Pis can reach each other. Let's try to send human-readable messages over that wire.

Communicating over the wire

1. We will use the tool netcat to send plaintext messages to one another. On `example`, start listening on port 80:

   root@example:~# `nc -l -p 80`

   Then, connect to it from `example2` and start typing messages:

   root@example2:~# `nc example.local 80`
   root@example2:~# `Hello can you hear me?`

   Hit `Ctrl + C` to stop netcat.

2. Instead of writing each other in human-readable plaintext, we can send other messages over the wire, such as a language your web browser understands, called the Hypertext Transfer Protocol (HTTP). On `example`, we will run a minimal webserver that can respond to HTTP messages. Look at the script `start-webserver.sh` and run it:

```
root@example:~# cat ~/scripts/start-webserver.sh
root@example:~# sh ~/scripts/start-webserver.sh
```

From `example2`, send a HTTP request to it:

```
root@example2:~# curl example.local
```

Observe the response from the webserver. When you are ready to move on, hit `Ctrl + C` to stop the server.

Our applications talk in many protocols, such as HTTP and SSH (how you are connected to the Raspberry Pi), but they all run on top of IP infrastructure. This means when we build a mesh network, we just need to ensure it talks the Internet Protocol, as we have just done, and all applications will run on the mesh network without modifications. Now we know our Netflix and YouTube will (theoretically) work seamlessly over a mesh network, let's try to extend our two-node IP network to include one more node :)