Peer-to-peer file sharing on a mesh

Connecting to your Raspberry Pi

1. Verify that the Raspberry Pi has its WiFi adapter(s) properly connected and the SD card is 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.

4. Find a Raspberry Pi near you and try to ping it to verify that the mesh network interface is working, replacing `example` and `example2` with the `hostname` of your node and the nearby node, respectively:

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

    Hit `Ctrl + C` to stop the ping.

You are now a mesh node operator running commands from your pet Raspberry Pi, which is connected to the rest of the mesh network we just formed in this room! For the rest of this activity, we will continue to use `example` to represent your node, and `example2` to represent another node near you. Be sure to replace these with the name of your own pet Raspberry Pi and that of its friend.

Running the InterPlanetary File System (IPFS)

1. Start IPFS in the background:

    root@example:~# `docker load --input ~/docker/tomeshnet-ipfs-0.1.tar`
    root@example:~# `docker run --name ipfs --network host --detach tomeshnet/ipfs:0.1`

    With these two commands, we have used a tool called `docker` (which we will not go into detail for now) to load all the files required to run IPFS, then told the Raspberry Pi to run IPFS in the background. You won't see much visual output, but next we will start interacting with this application.

2. Initiate an interactive session and write a message to the peer-to-peer filesystem:

```
root@example:~# docker exec -it ipfs sh
/ # echo "Hello World" | ipfs add
```

With the first command, we navigated to an interactive shell with access to the running IPFS application, and the second command added the text `Hello World` to the peer-to-peer filesystem.

3. Copy the content address from the `ipfs add` output, which is uniquely derived from your `Hello World` text. To read the content back, use `ipfs cat` followed by pasting the content address you copied:

   ```
   / # ipfs cat QmWATWQ7fVPP2EFGu71UkfnqhYXDYH566qy47CnJDgvs8u
   ```

4. Next, you can go to your browser and access the IPFS content you just added! You can use a URL formatted like this to access IPFS content from the browser of your computer:

   ```
   http://<hostname>.local:8080/ipfs/<ipfs-content-address>
   ```

   For example:

   ```
   http://example.local:8080/ipfs/QmWATWQ7fVPP2EFGu71UkfnqhYXDYH566qy47CnJDg
   ```

   Then you should see `Hello World` in your browser window. This works because each node is running an ipfs-to-http gateway that can be accessed from clients connected to the Access Point of the Raspberry Pi.

5. Try to echo other text and read it back in the shell and display it in your browser

You can actually add any type of file, such as photos and videos, to IPFS, and regardless of the file size you can address (and verify!) them with a short content address like this. So far we have created and fetched content from the IPFS running on our pet Raspberry Pi, but we have not shared any file across the mesh network. Let's do that!

Forming an IPFS content network

1. In order to form a network, you need two nodes to know about each other. So let's start with finding your identity in the network and sharing it with another node:

   ```
   / # ipfs id
   ```

   You will see a bunch of lines as output. Note the address that starts with `/ip6/fc` and looks something like this:

   ```
   /ip6/fcb0:3f14:ebc8:1f7b:a1ce:bd44:a410:5049/tcp/4001/ipfs/QmXLWSa1AbLJfi
   ```

   The `/ip6/fc00::/8` is your mesh node's unique IPv6 address (we will cover IPv6 addresses in the next workshop), and the part after `/ipfs/` is your IPFS node ID. You can think of the IPv6 as the street address to your Raspberry Pi, where one will find a live IPFS node, and the IPFS node ID uniquely identifies that IPFS node among all IPFS nodes in the world. Copy this line to a note pad and we will paste it into another node. With these two pieces of information it can both find and identify your IPFS node.

2. Type `exit` once to leave the IPFS session for now. Find a nearby node, ask the operator for permission to access their pet Raspberry Pi via SSH and confirm that they already have IPFS running. Now from the session of your Raspberry Pi, initiate an SSH session to the nearby node:

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

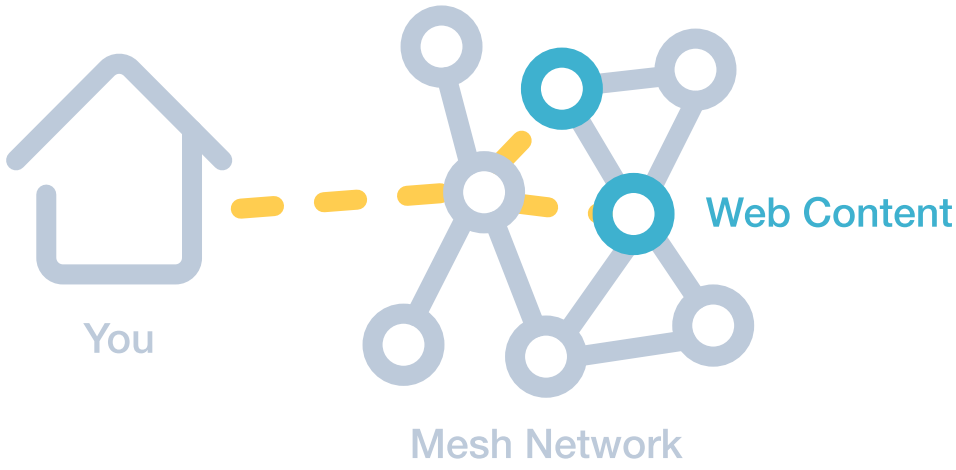
3. Yes we are in two levels of SSH sessions, but that's ok :) Now we initiate an interactive session to IPFS on `example2`, then add your address to tell it about your node:

   ```
   root@example2:~# docker exec -it ipfs sh
   / # ipfs bootstrap add
   /ip6/fcb0:3f14:ebc8:1f7b:a1ce:bd44:a410:5049/tcp/4001/ipfs/QmXLWSa1AbLJfi\
   ```

   Now your nodes are peered to form an IPFS network, which means `example` can access any content in the network accessible to `example2`, and vice versa. It doesn't take many peerings to connect all the nodes in this room.

4. Now let's add some content on `example2`, then type `exit` twice to return to `example`, and try to read that text back on `example` and from your browser. It may take a while after you first establish peering, but eventually the content will become available across the network.

To recap, we started the IPFS application and ipfs-to-http gateway in a docker container running on a Raspberry Pi, then peered two nodes to create an IPFS content-addressing network. Each mesh node is now also an IPFS node that publishes and fetches content within our local mesh network.



Can you identify where the pet Raspberry Pi system you just built fit in the above diagram of a mesh network?