Comprehensive Setup Guide for Establishing a Raspberry Pi Mesh Network with Hyperledger Fabric for STOFL

This guide provides detailed instructions for configuring a Raspberry Pi mesh network to support Hyperledger Fabric, tailored for STOFL. By following these steps, you'll learn how to prepare your Raspberry Pi devices, establish a mesh network among them, and deploy Hyperledger Fabric.

Table of Contents

Prerequisites

Before starting, ensure you have the following:

- 12 Raspberry Pi devices (1 designated as the gateway and 11 as peers, with scalability options)
- Corresponding SD cards for each Raspberry Pi
- An SD card reader-equipped computer
- Internet connectivity for software downloads, plus an Ethernet cable
- Terminal or command prompt acces

Flashing Raspberry Pi Devices

It is recommend using the Raspberry Pi Imager for device preparation. Select your Raspberry Pi model (e.g., Raspberry Pi 4), opt for the Debian Bullseye Legacy 64-bit full OS, and choose your SD card. Before proceeding to flash:

- Hostname configuration: `blocc-containerX`, X representing the device number (e.g., blocc-container1).
- Default username: `pi` with password `blocc-pi`.
- Wi-Fi configuration is crucial for internet access during setup.
- Enable SSH with password authentication for remote access.

Note: The Debian Bullseye image is preferred due to its compatibility with Docker and necessary dependencies for this guide. Using alternative versions like Bookworm will lead to compatibility issues.

Creating a Mesh Network

1. Basic Network Configuration

After flashing the Raspberry Pi devices, power on the Pi that is going to be your gateway Pi and connect it to your laptop using an ethernet cable. After that, establish an SSH connection using `ssh pi@hostname.local`, substituting "hostname" with the previously assigned hostname. This device should automatically connect to the internet, a configuration step completed during the flashing process. Confirm connectivity by executing `ping 8.8.8.8` and checking for zero packet loss. If Wi-Fi setup was overlooked or needs updating, utilize `raspi-config` to adjust settings accordingly. An Ethernet connection from your laptop to the Pi may also facilitate internet access if your laptop's sharing options are configured correctly. Note that after using `raspi-config`, a reboot is necessary (`sudo reboot n`).

2. Create the Gateway Pi

Acknowledging the foundational work at https://github.com/binnes/WiFiMeshRaspberryPi, this section guides you through configuring the Pi as a gateway for the mesh network.

batman-adv Configuration

1. Upon reboot, install the mesh network management utility, `batctl`, with `sudo apt-get install -y batctl`.

2. Create a startup script, `~/start-batman-adv.sh`, using your preferred text editor (e.g., `vim ~/start-batman-adv.sh`). The script should contain commands to add `wlan0` to batman-adv, configure `bat0`, and activate these interfaces.

```
#!/bin/bash
# batman-adv interface to use
sudo batctl if add wlan0
sudo ifconfig bat0 mtu 1468

# Tell batman-adv this is a gateway client
sudo batctl gw_mode client

# Activates batman-adv interfaces
sudo ifconfig wlan0 up
sudo ifconfig bat0 up
```

3. Make this script executable: `chmod +x ~/start-batman-adv.sh`.

4. Define the network interface for `wlan0` by creating `sudo vim /etc/network/interfaces.d/wlan0` and configuring it for ad-hoc mode with a specified channel and ESSID. These values must be consistent across all devices in the network.

```
auto wlan0
iface wlan0 inet manual
    wireless-channel 1
```

```
    wireless-essid blocc-mesh
    wireless-mode ad-hoc
```

5. Ensure `batman-adv` loads at boot (`echo 'batman-adv' | sudo tee --append /etc/modules`) and prevent DHCP from managing `wlan0` (`echo 'denyinterfaces wlan0' | sudo tee --append /etc/dhcpcd.conf`).

6. Make sure the startup script gets called by editing file **/etc/rc.local** as root user, e.g. `sudo vim /etc/rc.local` and insert:

   ```
   /home/pi/start-batman-adv.sh &
   ```

   before the last line: **exit 0**

7. At this point, do not shut down the Pi.

Enabling the Gateway

The gateway will be the DHCP server for the mesh network. DHCP is the service that provides network configuration to devices on a network. As the mesh network is a separate network from the home/office network, the DHCP service will provide devices with network configuration for the TCP network that runs over the mesh.

We will configure a subnet for the mesh network with the following attributes:

Attribute	Value
Subnet	192.168.199.0/24
Subnet Mask	255.255.255.0
Gateway IP Address	192.168.199.1
Lower IP Range	192.168.199.2
Upper IP Range	192.168.199.99

> This is the subnet that will be running on the `bat0` interface of all connected nodes.

1. Install DHCP services: `sudo apt-get install -y dnsmasq`.

2. Configure the DHCP server in `/etc/dnsmasq.conf` by:

   - `sudo vi /etc/dnsmasq.conf`

   and add the following lines to the end of the file:

   ```
   interface=bat0
   dhcp-range=192.168.199.2,192.168.199.99,255.255.255.0,12h
   ```

3. Change the startup file to add the routing rules to forward mesh traffic to the home/office network and do the Network Address Translation on the reply. Set the node as a mesh gateway and also configure the gateway interface IP address. To do this update the **start-batman-adv.sh** file and change the content to:

```
#!/bin/bash
# batman-adv interface to use
sudo batctl if add wlan0
sudo ifconfig bat0 mtu 1468

# Tell batman-adv this is an internet gateway
sudo batctl gw_mode server

# Enable port forwarding
sudo sysctl -w net.ipv4.ip_forward=1
sudo iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
sudo iptables -A FORWARD -i eth0 -o bat0 -m conntrack --ctstate
RELATED,ESTABLISHED -j ACCEPT
sudo iptables -A FORWARD -i bat0 -o eth0 -j ACCEPT

# Activates batman-adv interfaces
sudo ifconfig wlan0 up
sudo ifconfig bat0 up
sudo ifconfig bat0 192.168.199.1/24
```

4. Power down the gateway Pi: `sudo shutdown -h now`.

Verifying Gateway Configuration

Ensuring the operational integrity of your mesh network involves a sequence of verification steps. Begin with your laptop connected to your primary home or office network.

1. Establish an SSH connection to the gateway Pi:

   ```
   ssh pi@hostname.local
   ```

   Substitute "hostname" with the actual hostname of your gateway Pi.

2. Execute the `ifconfig` command to inspect the network interfaces on the gateway Pi. You should observe:

   - The `eth0` interface with an IP address from your home/office network.
   - The `bat0` interface with the IP address 192.168.199.1.
   - The `wlan0` interface without an assigned IP address, indicating its role in the mesh network.

3. Use the `iwconfig` command to review wireless interfaces. You should see:

```
wlan0   IEEE 802.11  ESSID:"blocc-mesh"
        Mode:Ad-Hoc  Frequency:2.462 GHz  Cell: 3A:BC:74:3B:A1:D9
        Tx-Power=31 dBm
```

```
          Retry short limit:7   RTS thr:off   Fragment thr:off
          Power Management:on
```

Notable details include the ESSID "blocc-mesh" and the Ad-Hoc mode, crucial for mesh networking.

4. Display mesh interfaces participating in the network with `sudo batctl if`. The `wlan0` interface should be marked as active, indicating its participation in the mesh.

5. Use `sudo batctl n` to list neighboring mesh nodes visible to your gateway. This command verifies the mesh's operational status and node connectivity.

6. For a more intuitive display of mesh nodes, map MAC addresses to hostnames by creating `/etc/bat-hosts`. Include the MAC address and a descriptive hostname for each node, as shown:

   ```
   b8:27:eb:8e:ec:6c   blocc-container1
   b8:27:eb:bd:4d:e5   blocc-container2
   b8:27:eb:01:d4:bb   blocc-container3
   ```

   Subsequently, `sudo batctl n` will display neighbor nodes with identifiable hostnames, enhancing network monitoring and management.

3. Expanding the Mesh Network to Additional Pis

Integrating additional Raspberry Pis into the mesh network involves a streamlined process, utilizing pre-defined scripts for seamless configuration.

1. Securely SSH into each Raspberry Pi.
2. Clone the configuration repository: `git clone https://github.com/alfredomusumeci/blocc-pi-setup.git`.
3. Navigate inside the folder and execute `./pi-setup.sh` to install preferred text editors (e.g., vim), though this step is optional.
4. Run `./mesh-setup.sh IP_SUFFIX=01` for each Pi, adjusting `IP_SUFFIX` to match the specific Pi's identifier. Note the importance of leading zeros for numbers less than 10.
5. Verify the `start-batman-adv.sh` script to ensure the IP address configuration aligns with your network setup. Manual adjustments may be necessary to correct any discrepancies. Note that this is an important step as sometimes the start-batman-adv.sh script will have an error in that its environment variable is not being replaced by its value, the cause is uncertain. If that happens, proceed to modify the file and save changes.
6. Reboot the Pi (`sudo reboot n`). Upon reboot, the Pi will automatically join the mesh network.

Validation steps include SSHing into the gateway Pi and then to other Pis within the network, ensuring each device can access the internet through the gateway if it is active, and using `sudo batctl n` to visualize network connectivity among the Pis.

Installing Hyperledger Fabric

Once all Pis are integrated into the mesh network, the next phase involves deploying Hyperledger Fabric. This process utilizes a script within the blocc-pi-setup directory named `fabric-setup.sh`. This script not only retrieves the latest Hyperledger Fabric binaries but also configures the necessary files to properly configure the blockchain network. Accurate execution of this script is crucial.

For a full network with all 11 devices, execute `./fabric-setup.sh --container=1 --all`, adjusting the `--container` flag to match the specific Pi device (e.g., `blocc-container4` would use `--container=4`). If your setup includes a subset of the total devices, specify those devices using `./fabric-setup.sh --container=3 --specify=[3,6,10]` for `blocc-container3`, `blocc-container6`, and `blocc-container10`. Misconfiguration at this stage will impede successful transaction commitments, as the system expects participation from a predefined set of organizations.

Spinning Up and Utilizing the Fabric Network for STOFL

With Hyperledger Fabric installed, focus shifts to initiating the Fabric network and leveraging it for transaction validation (e.g., sensor readings).

1. Access the Fabric directory: `cd $HOME/fabric`.
2. Prior to starting or after shutting down Fabric, perform the following critical steps:
    - Terminate any running containers: `docker kill $(docker ps -q)`.
    - Remove the organizations directory, if it exists: `sudo rm -rf organizations`.
    - Similarly, delete the channel-artifacts directory, then recreate it `sudo rm -rf channel-artefacts; mkdir channel-artefacts`.
    - Erase the `peer` and `orderer` directories, identifiable by their lengthy, hostname-derived names.
3. On a single Pi, generate cryptographic materials: `cryptogen generate --config=$HOME/fabric/crypto-config.yaml --output=$HOME/fabric/organizations`.
4. Distribute these materials to each Pi in the network using `scp -r organizations pi@blocc-container1:/$HOME/fabric/organizations`, altering the target container number as necessary. The reason behind that is that the cryptogen tool creates private keys for each organization (i.e. for each pi) and these need to match for every Pi otherwise each Pi will hold different crypto material.
5. Within the script directory, `cd scripts`, initiate the orderer joining process for each channel by running `./join_orderer_to_channel channel_number`, iterating through each Pi with the appropriate channel number, i.e. blocc-container1 will have its orderer join ONLY channel 1 and so on.
6. Execute `./join_peer_to_channel -channel_number` on every Pi for each channel, ensuring comprehensive network participation, i.e. blocc-container1 will run the script 11 times, each time with channel number varifying from 1 to 11. The other Pis will do the same.
7. Install the sensor chaincode on every channel across all Pis using `./install_sensor_chaincode channel_number`, observing the script's feedback to confirm successful deployment. Similarly to before, run the script 11 times on each pi for all channel numbers.
8. Lastly, run the script `./commit_chaincode_def.sh` on each Pi, specifying the container and either using `--all` for a full network or `--specify=[3,6,9]` for a partial setup. This script, similarly to step 5, needs to only be run once per Pi. Hence, I would do ./commit_chaincode_def --container=1 --all in container 1, then move onto blocc-container2 (Pi 2) and substitue --container=2, etc. Note that, if you have a different configuration than all 11 pis, then instead of passing the flag --all you should pass --specify=[3,6,9] (replace 3,6,9 accordingly). Failure to do so will results in errors.

9. Finally you're ready for sending sensor readings. If you want to send a transaction manually, then within one of the Pi (this will identify the channel you're sending it to) substitue the below and run accoridngly:

```
export
OSN_TLS_CA_ROOT_CERT=/home/pi/fabric/organizations/ordererOrganizations/co
ntainer3.blocc.doc.ic.ac.uk/tlsca/tlsca.container3.blocc.doc.ic.ac.uk-
cert.pem
export
PEER_CONTAINER10_CA_ROOT_CERT=/home/pi/fabric/organizations/peerOrganizati
ons/container10.blocc.doc.ic.ac.uk/peers/peer0.container10.blocc.doc.ic.ac
.uk/tls/ca.crt
export
PEER_CONTAINER6_CA_ROOT_CERT=/home/pi/fabric/organizations/peerOrganizatio
ns/container6.blocc.doc.ic.ac.uk/peers/peer0.container6.blocc.doc.ic.ac.uk
/tls/ca.crt

peer chaincode invoke -o blocc-container3:7050 --tls --cafile
${OSN_TLS_CA_ROOT_CERT} -C channel3 -n sensor_chaincode --peerAddresses
blocc-container6:7051 --tlsRootCertFiles ${PEER_CONTAINER6_CA_ROOT_CERT} -
-peerAddresses blocc-container10:7051 --tlsRootCertFiles
${PEER_CONTAINER10_CA_ROOT_CERT} -c '{"Function":"addReading","Args":
["0.1", "0.3", "300"]}'
```

You need to specify as many peerAddresses and tlsRootCertFiles as many Pis you have in your configuration.

Alternatively, it is possible to run the frontend/backend application which will keep sending transactions every 5 seconds (work in progress). 10. To bring down the network run `./container.sh down`, and repeat steps from 2.

Hyperledger Explorer

Each Pi also comes with a folder detailing its configuration for Hyperledger Explorer, a basic UI interface developed by the Hyperledger team to visualize your blocks and transactions over the network. In order for you to see that in your own laptop, you should ssh into your gateway with tunnelling enabled and allowing the explorer port (8080) to be tunnelled to a port of your choosing: `ssh -L 8080:blocc-container3.local:8080 pi@blocc-gateway.local` (in this case, also 8080). The command shows that explorer is running on blocc-container3 and I am tunnelling that through blocc-gateway to my own laptop. I can therefore open a browser in my laptop to localhost:8080 and visualize the explorer interface. Of course, in order for this to work before even ssh-ing, Hyperledger Explorer needs to be running, just cd into fabric/explorer and then run `docker-compose -f explorer.yaml up -d`, then run `docker-compose -f explorer.yaml down -v to` bring it down (this will stop the localhost UI).

Troubleshooting

NOTE

- When in a mesh, it is not possible to ssh into any pi unless: a. the ethernet cable to the desired pi we are trying to ssh into is connected to the laptop
- It is possible to ssh to the gateway pi even if the usb c cable is connected to the power source as long as the ethernet cable is connected to the laptop
- Even if the gateway is off, the pis are still able to see each other in the mesh