Tech Note

Running Afero Hub Software™ on Separate Hosts

December 15, 2017 – Version 1.1

1. Introduction

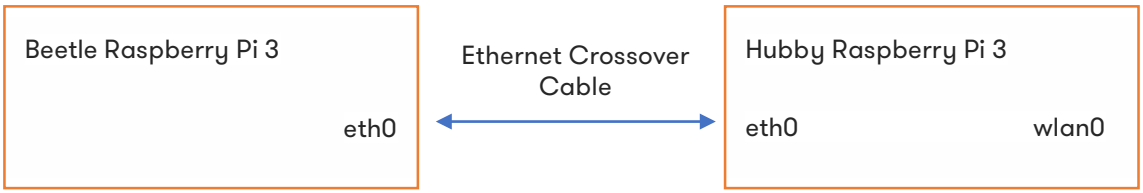
The two primary components of the Afero Hub Software, “Beetle” and “Hubby”, can be operated on two separate hosts. This configuration can be useful for testing since it allows shaping of network traffic between Beetle and Hubby. The technique could also be used to support Afero devices in a location without internet access, by running Beetle within Bluetooth range of the Afero devices, while running an associated instance of Hubby in a location where internet access is available.

- **Hubby (hubby)** - Software daemon that provides communication between Afero powered devices and the internet (via the Linux system’s internet connectivity).
- **Beetle (beetle)** - Software daemon that provides communication between the Afero Hub Software daemon and BlueZ, the Linux Bluetooth stack.

2. Approach

- Run Beetle on one host and Hubby on another host.
- Limit beetle←→hubby traffic to a private 10.x.x.x network connection on eth0 between the two hosts.
- Use **tc** and **wondershaper** to shape traffic on the eth0 interface to test beetle/hubby interaction.
- All hubby external traffic will traverse the Wi-Fi network on wlan0.

3. Diagram



4. Requirements

- Two Raspberry Pi 3 systems running the latest Raspbian "Stretch" OS from https://www.raspberrypi.org/downloads/raspbian/ (Raspbian Stretch Lite recommended).
- An Ethernet crossover cable, or two Ethernet cables and a private Ethernet switch, to connect the Pis.

5. Step-by-Step Guide

1. Connect both Raspberry Pis to a Wi-Fi network so they are accessible for set-up purposes.
2. Connect the two Pis via Ethernet ports using the crossover cable or via an Ethernet switch.

   Note: It is important the Ethernet connection between the Pis is completely isolated from any other network.
3. Decide which Pi is going to be the Beetle host and which will be the Hubby host.
4. On the Beetle host:

   a. Run:

   ```
   sudo vi /etc/network/interfaces
   ```

   b. Remove the line that reads "iface eth0 inet manual" and add these lines in its place:

   ```
   auto eth0
   iface eth0 inet static
   address 10.10.10.10
   netmask 255.255.255.0
   ```

c. Run the commands:

```
sudo ifdown eth0
sudo ifup eth0
```

d. Verify eth0 has the right IP address with:

```
ifconfig eth0
```

5. On the Hubby host, do the same except select a different address for the eth0 port, such as 10.10.10.11.
6. Verify that you can communicate over this private link by running `ping 10.10.10.11` from the Beetle host and `ping 10.10.10.10` from the Hubby host. Ensure this link is working before proceeding.
7. Set up the Hubby host to run the Afero Developer Hub Software per the instructions on the Afero Developer Portal.
8. On the Beetle host, perform all the steps as you would to install the Developer Hub Software but instead of installing `afero-hub`, install only the `afero-ble` package.
   a. Refer to the Afero Developer Hub Setup instructions on the Afero Developer Portal.
   b. The Beetle host will now be running Beetle ONLY. We will configure Beetle and Hubby to communicate with each other in the next steps.
9. Stop the services on the Hubby host by typing:

```
sudo systemctl stop hubby
```

10. Configure the Hubby host:
    a. Run:

```
sudo vi /lib/system/system/hubby.service
```

    b. Remove the lines “Requires=beetle.service” and “After=beetle.service”.
    c. Change the “ExecStart” line to read “ExecStart=/usr/bin/hubby –h 10.10.10.10 –p 6969”.
    d. Save this file, then run the commands:

```
sudo systemctl daemon-reload
sudo systemctl disable beetle
sudo systemctl restart hubby
```

11. Configure the Beetle host:
    a. Run:

```
sudo vi /lib/system/system/beetle.service
```

    b. Change the “ExecStart” line to read “ExecStart=/usr/bin/beetle –A 10.10.10.10”.
    c. Save this file, then run the commands:

```
sudo systemctl daemon-reload
sudo systemctl restart beetle
```


12. Confirm operation by checking messages in `/var/log/syslog`:
    a. On the Hubby host, type:

       ```
       grep BEETLE_HOST /var/log/syslog
       ```

       and verify that the last entry in the log shows:

       ```
       BEETLE_HOST as 10.10.10.10.
       ```

    b. On the Beetle host, type:

       ```
       grep 6969 /var/log/syslog
       ```

       and verify that the last entry in the log shows a Beetle message:

       ```
       listening on 10.10.10.10:6969
       ```

Notes:

- If you are simply separating BLE and Hub functionality across your network, you can ignore the rest of this document covering latency and bandwidth testing.
- If you wish to separate these hosts across a Wi-Fi network, you can remove the Ethernet cable and configure Beetle and Hubby to use the wlan0 network addresses in the `hubby.service` and `beetle.service` files. Remember, both files' configuration needs the IP address of the Beetle host.

6. Adding Latency

If you wish to test Hub functionality under non-ideal network configurations, you can add latency to the hubby/beetle connection with the command:

```
sudo tc qdisc add dev eth0 root netem delay 100ms
```

This example adds 100ms of latency (50ms outbound packet delay, 50ms inbound packet delay). You can remove this throttle with:

```
tc qdisc del dev eth0 root netem
```

These commands will not survive a reboot and can be entered from either the Beetle or Hubby hosts.

7. Verifying Latency

Use `ping 10.10.10.10` from the Hubby host (or `ping 10.10.10.11` from the Beetle host) to watch the ping latency times as you alter them with tc:

```
64 bytes from 10.10.10.11: icmp_seq=70 ttl=64 time=200 ms
64 bytes from 10.10.10.11: icmp_seq=71 ttl=64 time=200 ms
64 bytes from 10.10.10.11: icmp_seq=72 ttl=64 time=200 ms
64 bytes from 10.10.10.11: icmp_seq=73 ttl=64 time=0.262 ms
64 bytes from 10.10.10.11: icmp_seq=74 ttl=64 time=0.269 ms
64 bytes from 10.10.10.11: icmp_seq=75 ttl=64 time=0.275 ms
64 bytes from 10.10.10.11: icmp_seq=76 ttl=64 time=100 ms
64 bytes from 10.10.10.11: icmp_seq=77 ttl=64 time=100 ms
64 bytes from 10.10.10.11: icmp_seq=78 ttl=64 time=100 ms
64 bytes from 10.10.10.11: icmp_seq=79 ttl=64 time=100 ms
64 bytes from 10.10.10.11: icmp_seq=80 ttl=64 time=0.206 ms
```

8. Limiting Bandwidth

tc can be used to limit bandwidth as well as set network latency. The syntax is a little hostile, so we recommend a utility called **wondershaper**, which uses the same mechanism as **tc**.

- To install it type:

  ```
  apt-get install wondershaper
  ```

- Then run it with:

  ```
  sudo wondershaper eth0 <downloadkbps> <uploadkbps>
  ```

- Reset the limits with:

  ```
  sudo wondershaper clear eth0
  ```

9. Verifying Bandwidth

To verify the bandwidth restrictions, use "`iperf`". To install it, run:

```
sudo apt-get install iperf
```

on BOTH systems, beetle and hubby. Then on the beetle side run:

```
iperf –s
```

On the hubby side run:

```
iperf -c 10.10.10.10
```

`iperf` will tell you the effective bandwidth between the two systems. Unthrottled Ethernet between two hosts should be around 90-95 MBits/sec.

`iperf` only runs a 10-second test by default so for slower bandwidth connections run a longer and more reliable test by adding the "`-t <seconds>`" option to `iperf` (such as "`-t 60`" to run a full minute test).

9.1 Examples

9.1.1 Unthrottled

```
$ iperf -c 10.10.10.10
------------------------------------------------------------
Client connecting to 10.10.10.10, TCP port 5001
TCP window size: 43.8 KByte (default)
------------------------------------------------------------
[ 3] local 10.10.10.11 port 45519 connected with 10.10.10.10 port 5001
[ ID] Interval Transfer Bandwidth
[ 3] 0.0-10.0 sec 112 MBytes 94.2 Mbits/sec
```

9.1.2 After “wondershaper eth0 100 100”

```
$ iperf -c 10.10.10.10
------------------------------------------------------------
Client connecting to 10.10.10.10, TCP port 5001
TCP window size: 43.8 KByte (default)
------------------------------------------------------------
[ 3] local 10.10.10.11 port 45521 connected with 10.10.10.10 port 5001
[ ID] Interval Transfer Bandwidth
[ 3] 0.0-22.9 sec 384 KBytes 137 Kbits/sec
```

10. Adding Latency and Limiting Bandwidth

To limit bandwidth and latency at the same time, set the latency limitation on the Beetle host with **tc** and set the bandwidth limitation with **wondershaper** on the Hubby host.

Limiting the beetle←→hubby connection to 9kps (old modem speed) and adding 500ms of latency on top of the network limitation still allows hubby/beetle to work. Not very quickly, but it does work.