Exploring OpenBTS: Lab Write-up

In this lab session, we explored the functionalities of OpenBTS, an open-source GSM base station, and its integration with mobile devices. The objective was to understand the setup process, customize the base station, establish a connection with a mobile devices, and confirm connectivity by sending an sms message.

This lab takes place inside the OpenBTS virtual machin on ialab.dsu.edu

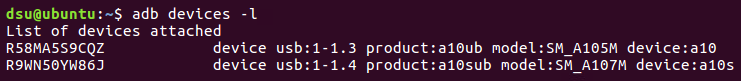
# Set Up:

Since the remainder of the lab relies on the presence of the radio device, I began the lab by executing the **lsusb** command to confirm the connection status of the OpenBTS radio.

A screenshot of a computer

Description automatically generated

Next, I executed the **adb** devices command in the terminal in order to view connected Android devices and obtain their serial numbers.



By selecting a device and extracting the serial number (R9WN50YW86J), I initiated the connection with the phone by executing the command 'scrcpy -s R9WN50YW86J'. This command allowed me to mirror the phone's screen onto my desktop, which enabled access and control of the device's interface and functionalities directly from my computer.

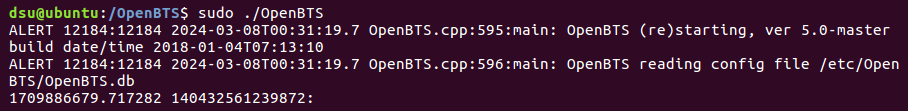
A screenshot of a computer code

Description automatically generated

A screenshot of a cell phone

Description automatically generated

Next, I launched OpenBTS by navigating to the appropriate directory and executing the `**sudo ./OpenBTS/**` command. This interface serves as the platform for executing various commands, including configuration changes, which are processed by OpenBTS to customize and manage the behavior of the base station. [1]



A close up of a screen

Description automatically generated

# Customization

Once inside the BTS command line, I modified the Mobile Network Code (MNC) key using the command **config GSM.Identity.MNC 17,** which resulted in the network being identified as 00117.



Next, I set the short name for the network to "DontBlink" using the command **config GSM.Identity.ShortName DontBltnk**. Assigning a unique short name to the network is important as it facilitates easy identification within the connected mobile device, especially as it moves within the coverage area. [1]



Before connecting the device, I ensured that Open Registration was enabled within the OpenBTS system. This step is important as it allows devices to join the network and use its services even if they have not been assigned a phone number yet. While some of these services are limited until they complete a setup process, devices setup in this manner can receive text messages. This ensures that devices can stay connected and receive important communications while they complete the setup process to fully integrate into the network. [1] Since this feature is disabled by default, I enabled it using the command 'config Control.LUR.OpenRegistration .\*.'

A computer screen shot of a program

Description automatically generated

# Connect to Device

After these initial configurations, I attempted to connect to the mobile device. However, my assigned network identity (00117) did not show up as expected. To resolve this issue, I first adjusted the Absolute Radio Frequency Channel Number (ARFCN). It is important for each tower to have its own unique frequency because if two nearby towers use the same frequency, their signals can interfere with each other and cause disruptions - similar to trying to listen to two radio stations broadcasting on the same frequency simultaneously. [1] Therefore, to resolve any possible network connectivity issues, I set the ARFCN to channel 10 by using the command ‘config GSM.Radio.C0 10.’

A black background with white text

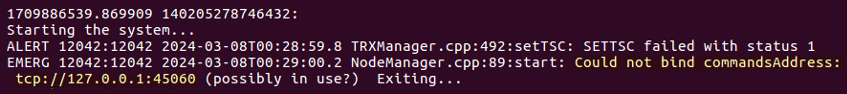
Description automatically generated

Next, I updated the gain settings within OpenBTS by setting it to the suggested value of ‘10’ by using the command rxgain 10. This adjustment helps ensure that the signals transmitted and received by OpenBTS are set to an appropriate level in order to achieve clear and reliable communication. [1] Therefore, by lowering the gain from "47" to "10", I aimed to reduces the sensitivity of the BTS receiver to external signals, and make it less susceptible to interference.

A black background with white text

Description automatically generated

After restarting to apply these changes, I encountered another challenge as OpenBTS failed to connect due to a binding issue on port 405060. To address this issue, I executed the command sudo netstat -tulnlp to identify the process ID (PID) responsible for occupying the port. After locating the associated OpenBTS process (PID 11636), I was able to terminate the process and free up the necessary resources. These steps resolved the binding issue and enabled OpenBTS to successfully initialize and reestablish its required connections.





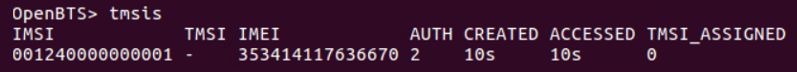
Finally, I was able to restart OpenBTS and establish a connection to the mobile device. By accessing the Network Operators menu and manually scanning for available networks, I identified my network by its short name, "DontBlink." After selecting this option, the mobile device successfully connected to OpenBTS.

A screenshot of a phone

Description automatically generated

# Connection Verification and Communication

To confirm the success of the connection, I executed the command **tmsis**. The output revealed only one device, which made it easy to match the International Mobile Equipment Identity (IMEI) with the new device.



However, To ensure accuracy, I also accessed the 'About' section of the Galaxy A10s to retrieve the IMEI number listed on the device.

A screenshot of a phone number

Description automatically generated

As expected, the IMEI values matched, verifying the IMEI number as 353414117636676. Therefore,

With these matching IMEI values, I can confidently assert that the corresponding International Mobile Subscriber Identity (IMSI) listed in the 'tmsis' output is the IMSI for our new device.

Additionally, the default Open Registration Message, managed by Control.LUR.OpenRegistration.Message, welcomes new devices to the network by broadcasting their IMSI. This message serves as an additional confirmation of the unique identifier, and can be observed through the SMS messages on the device. However, it is crucial to emphasize that broadcasting the IMSI via a message is highly unusual and specific to this lab environment.

A screenshot of a phone

Description automatically generated

Lastly, I utilized the previously discovered IMSI to send a message and test the communication capabilities of the system. Using the 'sendsms' command followed by the IMSI, I initiated the message transmission process within OpenBTS.

