Importing 5G UE code to Android

**We had a working 5G
UE code for Linux. Our task was to migrate the code to Android.**

- **Cross Compiling**
  cross-compiling is when we build the executable for a platform other than in which the compiler is running. Our code had been build for Linux platform. So we used a cross compiler that can build the executable for android.

  We used linux-arm-gnueabi-g++ cross compiler.

- **Openssl and boost libraries for the android platform:**

  Our code was using some libraries which were missing in the cross compiler so we had to find arm version of those libraries. OpenSSL, boost and pthread libraries needed to be downloaded. We used this particular installation guide for OpenSSL.

  https://stackoverflow.com/a/43314315

- **Root permissions were needed for some purposes in the android**
- **Linking the libraries statically:**

  When we compiled the code normally and ran the code on the android device, the program was not able to link those libraries dynamically so we linked many libraries statically (adding the .o files to the code).

- **Disabling the timer in the code**
  After statically linking the relevant libraries the code started to run but it was showing some error. After some trial and error, we found that it is showing those errors in the code of the Timer. So we removed that part from the code for now since the code wasn't using the Timer data for any decision-making purposes. We were also trying to figure out why the Timer code is not working properly.
- **Testing the setup**
  Our code was able to make a connection and register with the N3IWF and was also able to establish a session.

- **Ping from the N3IWF was not working**

Connecting the phone to N3IWF

Testing of the code on android UE

After building both CP (control plane) and DP (data plane) code using the cross compiler (It is already specified in the CMake file we only need to run CMake and makefile).

```
>>mkdir build

>>cd build

>>cmake ..

>>make
```

Connect the android phone to the system using the USB.

USB debugging option must be turned on before connecting to the system. (USB debugging option can be found in the developer's option in setting)

Check if the device is connected using

>>adb devices #it will show all the devices connected with USB debugging ON

Push both the executables to the rooted android phone using adb command

>>adb push "executable_name" /data/local/temp

This file will go to the temp folder of the rooted device.

Using

>>adb shell command we can access the device via terminal.

Then we go to the temp folder and run both the CP and the DP executables.

>>**cd /data/local/temp**

>>**./uenas <Unique-ID>**

>>**./greUE**

>>**echo "R" > tunUE** # For registration initiation

>>**echo "S" > tunUE** # For session initiation