OAI Hands-On: RAN

Robert Schmidt

June 13, 2021

Outline

Preparation for RAN training session

The RAN Repository

Hands-On

Installation of dependencies and compilation

- This takes some time, so do it now!
- Dependency installation is needed only once

```
cd
git clone https://gitlab.eurecom.fr/oai/openairinterface5g.git
cd openairinterface5g/cmake_targets
git checkout develop                        # important!
./build_oai -I                              # install dependencies
./build_oai --ninja --gNB --nrUE -w SIMU    # compile gNB and nrUE
./build_oai -I -w USRP                      # for USRP
./build_oai --ninja --gNB --nrUE -w USRP    # for USRP
```

What do we cover in the RAN Hands-On?

- Some code repository explanations
- Training:
  - Set-up of end-to-end 5G/NR SA setup with RFsimulator
  - How to inject traffic in 5G/NR tunnel
  - Use of the scope + basic channel modelling
  - Connection of multiple UEs
  - How to build docker containers of the RAN
  - Individually, also covered tomorrow: How to start with the DU-CU/F1 split
  - Individually: how to set up with USRP, COTS UE
- No use of hardware SDRs

About the repository

- `https://gitlab.eurecom.fr/oai/openairinterface5g`
- Work happens in the `develop` branch
- Usually one integration branch per week, tagged in the format `YYYY-wWW`, e.g., `2022.w24`
- `master` for a known stable version

How to contribute

- Anyone can contribute!
- You have to sign a Contributor License Agreement
- Contributions go through
  - Peer review on Gitlab
  - Continuous Integration Build and Testing

Repository structure

- `openair1`: 3GPP LTE Rel-10/12 PHY, NR Rel-15 PHY
- `openair2`: 3GPP LTE Rel-10 MAC/RLC/PDCP/RRC/X2AP, NR Rel-15 MAC/RLC/PDCP/SDAP/RRC/X2AP (SA/NSA)
- `openair3`: 3GPP LTE Rel-10 S1AP/GTP
- Deep dive:
  - Where is NR PDSCH modulation? Called in `nr_generate_pdsch()`
  - Where is the NR PDSCH/DLSCH scheduler? See `gNB_dlsch_ulsch_scheduler()`
  - Where is the NR RRC Reconfiguration message sent? See `rrc_gNB_generate_dedicatedRRCReconfiguration()`
  - Where is the PDSCH simulation? See `dlschsim.c`

Repository structure

- `targets`: LTE executables (`RT/USER/`), SDR drivers (`ARCH`), configuration files (`PROJECTS`)
- `executables`: NR executables
- `cmake_targets`: everything related to compilation, build artifacts in `ran_build/build`
- `doc`: some documentation
- `ci-scripts`: everything related to continuous integration/testing, configuration files
- `common`: common code, generic libraries (Tpool, logging, configuration modules, ...)
- `charts/docker/openshift`: for building images
- Questions?

About the RFsimulator

- Why don't we use real radio?
- The RFsimulator *simulates* a radio device – it is a virtual SDR device
- Easier to set up, yet interchangeable with any other radio
- Large number of participants, no interference
- Allows the use of channel models

How to build

- Use of the `build_oai` script
  - Is a wrapper for `cmake`
  - Some useful options: `-h`, `--eNB`, `--gNB`, `--UE`, `--nrUE`, `-c`, `--ninja`, `--sanitize-address`, `-g`, `-w`, `-P`/`--physical_simulators`, ...
- By default, build artifacts are in `cmake_targets/ran_build/build` (`ran_build/` configurable, see `-d` switch)
- To rebuild more quickly, issue

```
ninja nr-uesoftmodem nr-softmodem coding dfts ldpc_optim8seg ldpc_optim
      ldpc_orig ldpc oai_eth_transpro params_libconfig rfsimulator nrscope
      params_libconfig telnetsrv
```

- Also interesting: `lte-softmodem`, `lte-uesoftmodem`

Basic end-to-end setup

- Start the core:

```
docker-compose -f docker-compose-5gcn-basic.yaml up -d
docker-compose -f docker-compose-5gcn-basic.yaml ps -a
```

- Start Wireshark with capture filter `sctp`
- Start the gNB and UE in a second terminal:

```
cd ~/openairinterface5g/cmake_targets/ran_build/build
sudo -E RFSIMULATOR=server ./nr-softmodem --rfsim --sa
      -O ~/oai_hand_on_session/gnb.sa.band78.fr1.106PRB.usrpb210.conf
cd ~/openairinterface5g/cmake_targets/ran_build/build
sudo -E RFSIMULATOR=127.0.0.1 ./nr-uesoftmodem -r 106
      --numerology 1 --band 78 -C 3619200000 --rfsim --sa --nokrnmod
      -O ~/oai_hand_on_session/ue.conf
```

How to inject traffic?

- One terminal in the host, the other in the docker container `demo-oai-ext-dn`

```
docker exec -it demo-oai-ext-dn bash
```

- Check the UE's IP address: interface `oaitun_ue1` with `ip address`
- Ping:

```
ping -I oaitun_ue1 12.1.1.1 # from host, "UL"
ping <UE IP address>        # from container, "DL"
```

- iperf:

```
iperf -sui1 -B <UE IP address>              # from host
iperf -uc <UE IP address> -i1 -t10 -b1M     # from container, DL traffic
```

Scope

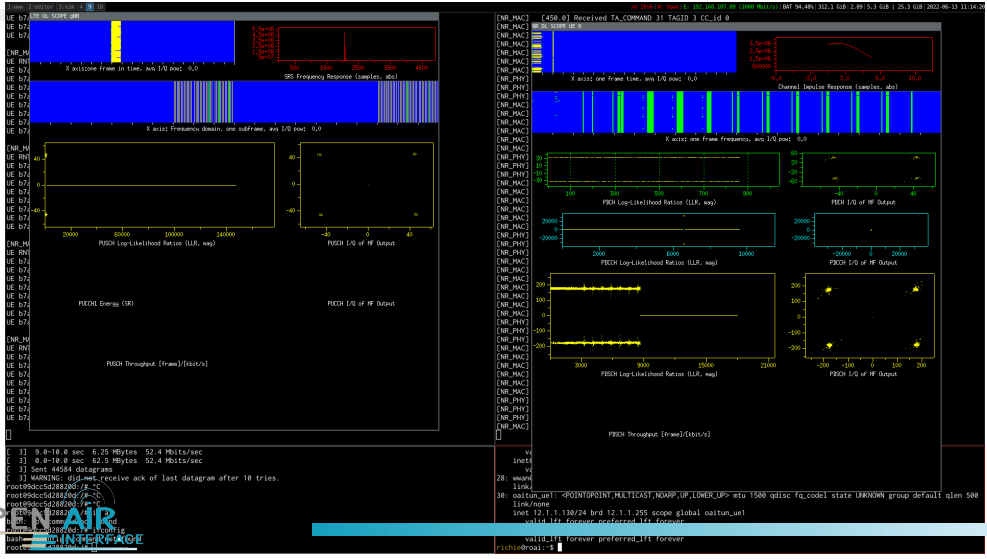
- Build scope manually in build directory: `ninja nrscope` (or pass the `--build-lib nrscope` option to `build_oai`)
- Run `nr-softmodem`/`nr-uesoftmodem` with `-d` switch
- If you get this error:

```
In fl_initialize() [flresource.c:995]: 5G-gNB-scope: Cant open display :0
In fl_bgn_form() [forms.c:347]: Missing or failed call of fl_initialize()
```

  Then you have to allow root to open the X display

```
xhost +si:localuser:root
```

Scope: Screenshot



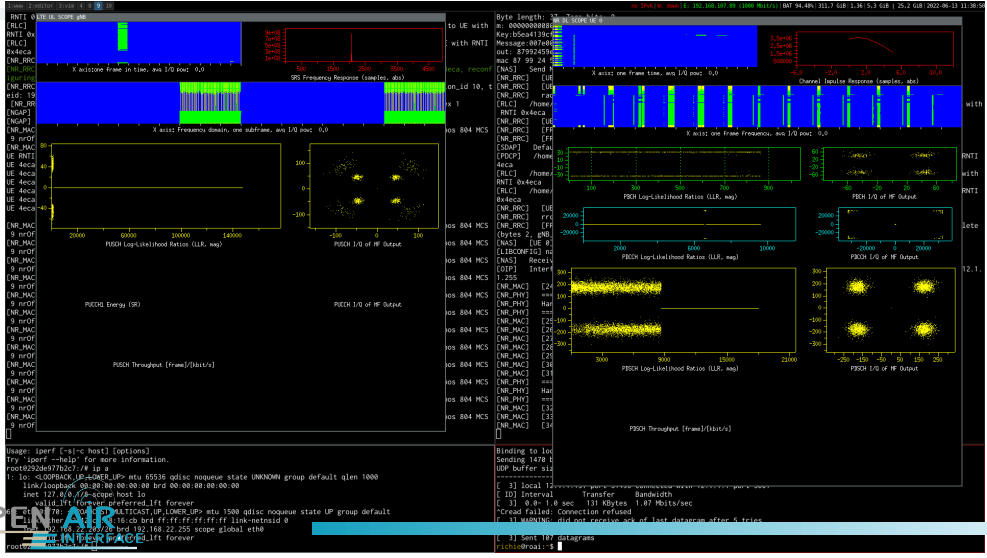
Channel Model

- We will simply add Gaussian noise
- Uncomment the last line in both the gNB and UE config file:

```
@include "channelmod_rfsimu.conf"
```

- Add the options `--rfsimulator.options chanmod --rfsimulator.modelname AWGN` to the gNB and UE command lines and start
- More info: `targets/ARCH/rfsimulator/README.md`, channel models defined in `openair1/SIMULATION/TOOLS/random_channel.c`

Channel Model: Screenshot



Dynamically modifying Channel Model Parameters using Telnet

- Make sure the telnet shared library is compiled: `ninja telnetsrv`
- Start the `nr-softmodem`/`nr-uesoftmodem` with parameter `--telnetsrv`
- New terminal: connect to gNB/UE: `telnet 127.0.0.1 9090`
  - Use `help` to show available commands
  - Use `channelmod show current` to show current channel model configuration
  - Use `channelmod help` to show available parameters to change
  - Example: change noise using `channelmod modify 1 noise_power_dB -5`, observe in scope!
- More information: `common/utils/telnetsrv/DOC/telnetusage.md`

Multi-UEs

- Create namespace for each UE to prevent interface name clash
- Follow instructions from this link
- Replace accordingly:
  - UE1: `ueNameSpace1`, `v-eth1`, `10.201.1.1/24`
  - UE2: `ueNameSpace2`, `v-eth2`, `10.202.1.1/24`
  - `enp0s31f6` with your internet network iface
- Start first UE as

```
ip netns exec ueNameSpace1 bash
sudo -E RFSIMULATOR=10.201.1.1 ./nr-uesoftmodem ...
```

- Check in Wireshark/UE output that everything is ok

Multi-UEs: Contd.

- Second UE *with different IMSI*

```
ip netns exec ueNameSpace2 bash
sudo -E RFSIMULATOR=10.202.1.1 ./nr-uesoftmodem ...
      --uicc0.imsi 208950000000032
```

- Exercise: Check in Wireshark that both are set up
- Exercise: Create traffic from/to both UEs

How to create a docker image?

- Creating a docker image is a 3-step process (due to CI specificities, see also `docker/README.md`):
  1. `ran-base` for dependencies (shared image)
  2. `ran-build` for compiling all targets (shared image)
  3. Per-target (eNB, gNB, nrUE, lteUE, ...) images
- First, build the shared images:

```
docker build --target ran-base --tag ran-base:latest
      --file docker/Dockerfile.base.ubuntu18 .
docker build --target ran-build --tag ran-build:latest
      --file docker/Dockerfile.build.ubuntu18 .
```

- Then, build the target images, e.g., gNB and nrUE (as used in the CN session):

```
docker build --target oai-gnb --tag oai-gnb:latest
      --file docker/Dockerfile.gNB.ubuntu18 .
```

CU-DU split

- Use config files from repo: `gNB_SA_CU.conf` and `gNB_SA_DU.conf` under `ci-scripts/conf_files/`
- Change:
  - `mnc` to 95
  - First `sst`/`sd` to 1/1, second to 2/2
  - `amf_ip_address.[0].ipv4` to 192.168.22.196
  - `NETWORK_INTERFACES.GNB_IPV4_ADDRESS_FOR_NG_AMF` and `NETWORK_INTERFACES.GNB_IPV4_ADDRESS_FOR_NGU` to 192.168.22.193/24
- Start Wireshark with capture filter `sctp`
- Start the gNB as follows, then the UE as before:

```
sudo -E RFSIMULATOR=server ./nr-softmodem --rfsim --sa -O gNB_SA_CU.conf
sudo -E RFSIMULATOR=server ./nr-softmodem --rfsim --sa -O gNB_SA_DU.conf
```