**SRSRAN**

(Software Radio System Radio Access Network)



**Introduction to SRSRAN**

* The SRS-RAN project aims to provide an open-source implementation of the LTE(long term evolution) radio access network, allowing researchers, developers, and enthusiasts to experiment with and contribute to the development of LTE networks.
* It focuses on implementing LTE (4G) radio access networks through the innovative use of software-defined radio (SDR) technology.
* Srslte provide an end to end software solution open source for a 4g network.we include a full stack application for us enodeb and epc application.
* This is a Software written in c and c++ targetting X86 and arm processor and deployable and runnable on the laptop.
* Srslte name is updated to srs ran as we moved into 5g and the core and the focus on the ran itself and the enode b or gnobe b.
* Most trusted open-source software for mobile networks.
* Software Radio Systems (SRS:This refers to the organization or company that initiated the development of SRS-RAN. They specialize in software-defined radio solutions, and SRS-RAN is one of their projects.
* Radio Access Network (RAN): In telecommunications, a RAN is a part of a mobile telecommunication system that connects user devices (such as mobile phones) to the core network. It includes elements like base stations (eNodeB in LTE) and the radio frequency (RF) components responsible for wireless communication.
* Software-Defined Radio (SDR): SDR is a technology that uses software to implement the functionality of certain hardware components traditionally found in radio communication systems. This allows for more flexibility and programmability in wireless communication.

In the context of SRS-RAN, the combination of "Software Radio Systems" and "Radio Access Network" suggests that it is a software-based implementation of the LTE radio access network, utilizing the principles of software-defined radio.

SRS-RAN provides an open-source framework that allows users to set up and experiment with LTE networks in a software environment, making it a valuable tool for educational purposes, research, and development in the field of wireless communication. Users can customize and modify the software to simulate various network configurations, making it a versatile platform for learning and innovation.

**Set up SRSRAN**

SRS-RAN (Software Radio Systems Radio Access Network) is typically connected to hardware through the use of Software-Defined Radio (SDR) devices.

SDR is a technology that allows the flexible implementation of radio communication systems using software on a general-purpose computer, coupled with a hardware interface for radio frequency (RF) signal processing.

### Prerequisites:

1. Hardware:
   * Ensure that you have compatible SDR hardware. Commonly used devices include those from Ettus Research (USRP series), Lime Microsystems, or others.
2. Software Dependencies:
   * Install required software dependencies, including libraries and tools needed for building and running SRS-RAN. Refer to the official SRS-RAN documentation for specific requirements.

### Installation:

1. sudo apt-get install cmake make gcc g++ pkg-config libfftw3-dev libmbedtls-dev libsctp-dev libyaml-cpp-dev libgtest-dev
2. Clone the SRS-RAN Repository:
   * Clone the SRS-RAN repository from the official GitHub page:

git clone https://github.com/srsRAN/srsRAN\_Project.git

1. Install Dependencies:
   * Follow the instructions in the SRS-RAN documentation to install any additional dependencies required for your operating system.
2. Build SRS-RAN:
   * Navigate to the SRS-RAN directory and build the software:

cd srsRAN\_Project

* + mkdir build
  + cd build
  + cmake ../
  + make -j $(nproc)
  + make test -j $(nproc)
  + sudo make install
  + sudo add-apt-repository ppa:softwareradiosystems/srsran-project
  + sudo apt-get update
  + sudo apt-get install srsran-project -y
  + sudo gnb -c filename

## PHY testvectors

* + Sudo gedit tar -xzf phy\_testvectors.tar.gz -C /path\_to\_your\_local\_repository/srsgnb/
  + option(USE\_PHY\_TESTVECTORS "Enable testvector PHY tests" ON)

**Installing SDR Drivers:**

1. Ensure that the drivers for the selected SDR device are installed on your computer. These drivers facilitate communication between the SDR hardware and the software, allowing the SRS-RAN to interact with the hardware.

### Configuration:

1. Edit Configuration Files:
   * Modify the configuration files in the config directory to match your setup. Key configuration files include sib.conf, enb.conf, and rr.conf.
2. Configure SDR Hardware:
   * Specify the SDR hardware details in the configuration files. This includes information such as the device type, IP address, and other parameters relevant to your SDR hardware.

### Connecting to SDR Hardware:

1. Connect SDR Device:
   * Connect your SDR hardware to your computer and ensure it is properly recognized. Refer to the documentation for your specific SDR device for any additional setup steps.
2. Launch SRS-RAN:
   * Start the SRS-RAN software with the specified configuration:

./srsran\_enb

1. Monitor Console Output:
   * Monitor the console output for any error messages or status updates. This will provide information about the initialization of the SRS-RAN software.

### Testing:

1. Connect User Equipment (UE):
   * If desired, connect simulated User Equipment (UE) devices to the SRS-RAN setup. Modify the configuration files accordingly.
2. Experiment and Test:
   * Experiment with different LTE network configurations, test scenarios, and protocols using the SRS-RAN setup.

## Build Tools and Dependencies:

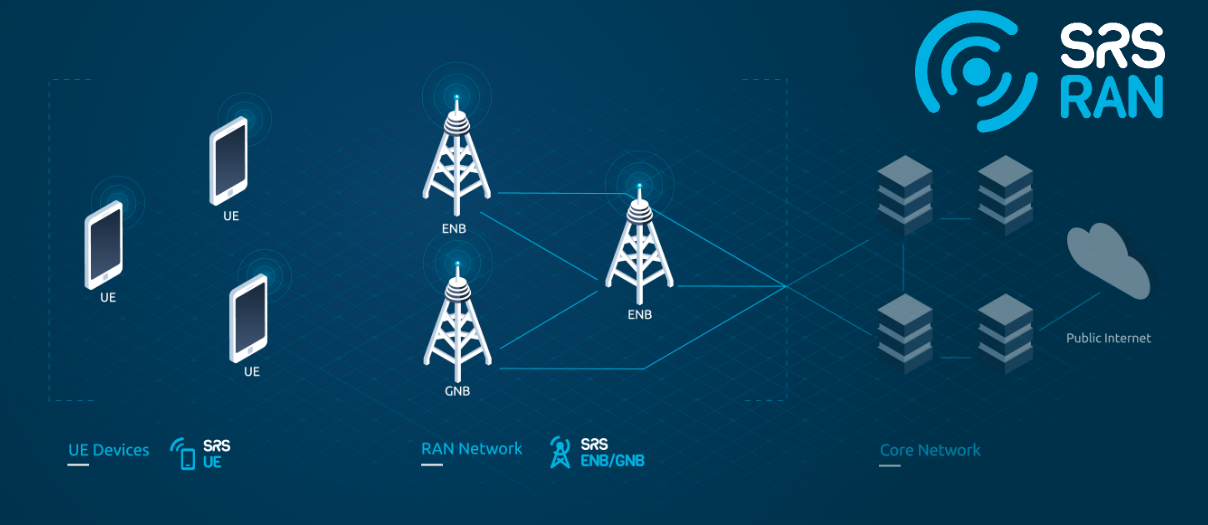
Tools:

* Cmake: A cross-platform build system used to configure and build the SRS-RAN project.
* Gcc:It is a compiler for various programming languages like C,C++.

Dependencies:

1. ibfftw (Fastest Fourier Transform in the West):
   * Purpose: libfftw is a library for computing discrete Fourier transforms (DFTs) efficiently.
   * Potential Use in SRS-RAN: It might be used for signal processing tasks related to the radio access network.
2. libsctp:
   * Purpose: libsctp is the implementation of the Stream Control Transmission Protocol (SCTP), a transport layer protocol.
   * Potential Use in SRS-RAN: It could be used for certain network communication tasks within the SRS-RAN system.
3. yaml-cpp:
   * Purpose: yaml-cpp is a YAML parser and emitter for C++.
   * Potential Use in SRS-RAN: It might be used for parsing and handling configuration files in the YAML format.
4. PolarSSL/mbedTLS:
   * Purpose: mbedTLS (formerly PolarSSL) is a cryptographic library that provides SSL/TLS support, cryptography algorithms, and secure communication protocols.
   * Potential Use in SRS-RAN: It could be used for implementing secure communication channels within the LTE network.
5. googletest:
   * Purpose: googletest is a testing framework for C++.
   * Potential Use in SRS-RAN: It may be used for writing and running unit tests to ensure the correctness of SRS-RAN components.

**SRSRAN ARCHITECTURE:**

****

**srsUE:** srsUE is a component in SRS-RAN that represents the User Equipment(UE) in the LTE network. It simulates the behavior of a mobile device,such as a smartphone or tablet,within the SRS-RAN environment.

The srsUE component allows user to simulate and experiment with the behavior of UE devices in the context of an LTE network.It is essential part of the SRS-RAN project for testing and research purpose.

**Srsenb:** It stands for Evolved NodeB,which is the nomenclature used in LTE for the base station or eNodeB.In srsran represents the LTE base station component.

The srsenb component is responsible for managing the radio resources,communicating with UEs(srsUE) , and facilitating the connection between Ues and the core network.It plays a central role in the SRS-RAN simulation.

**Srsepc:** It stands for Evolved Packet Core,which is the core network component in the LTE architecture.In SRS-RAN,srsepc represents the simulation of the LTE core network.

The srsepc component emulates the functionalities of the LTE core network,including the Mobility Management Entity(MME),Serving Gateway(SGW), and Packet Data Network Gateway(PGW).It manages the mobility of Ues and facilitates the transfer of data between the LTE networks and external networks.

**LTE Protocol Stack:**

SRS-RAN includes the LTE protocol stack,encompassing the different layers of the LTE communication model.This include the physical layer, MAC (Medium Access Control)layer,RLC(Radio Link Control) layer,PDCP (Packet Data Convergence Protocol)layer,and RRC(Radio Resource Control)layer.

**Importance of SRS-RAN:**

1. **Innovation in Telecommunication:**

SRS-RAN facilitates innovation by providing an open-source platform for the development of LTE (4G) radio access networks. It encourages experimentation and exploration of new ideas in the telecommunications field.

1. Research and Development:

Researchers and developers can use SRS-RAN to study and advance the state of LTE networks.It serves as a valuable tool for conducting experiments, testing protocols, and exploring novel approaches to wireless communication.

1. Education and Learning:

SRS-RAN is an educational resource that allows students and professionals to gain hands-on experience in telecommunications. It provides a practical environment for learning about LTE technology, software-defined radio, and network protocols.

1. Cost-Effective Prototyping:

As an open-source solution,SRS-RAN offers a cost-effective ways for individuals and organizations to prototype and experiment with LTE networks.This is especially beneficial for those with limited resources who want to explore and text telecommuication concepts.

1. Customization and Adaptability:

The open nature of SRS-RAN allows users to customize and adapt the software to specific requirements. This flexibility is crucial for researchers and developers who need to tailor LTE networks to unique scenarios or experiment with new features.

**Applications of srs-ran:**

1. LTE Network Planing:

SRS-RAN can be used for planning and simulating LTE networks. It helps in understanding network behavior, optimizing configurations, and assessing the feasibility of deploying LTE in different scenarios.

1. Protocal Optimization:

Researchers can utilize SRS-RAN to optimize LTE protocols, improving efficiency, security, and performance. This is valuable for enhancing the overall reliability of LTE networks.

1. Security Analysis:

The open-source nature of SRS-RAN facilitates security research. Users can analyze the security aspects of LTE networks, identify vulnerabilities, and propose improvements to enhance the overall security posture.

1. Community Networks:

SRS-RAN can be employed in the creation of community LTE networks. This is particularly relevant in areas with limited telecommunications infrastructure, offering a low-cost alternative for providing connectivity.

1. Wireless Experiments and Demos:

SRS-RAN is suitable for creating wireless experiments and demonstrations. It allows users to showcase and test various aspects of LTE technology, making it useful for educational events, workshops, and technology showcases.

1. Software-Defined Radio Exploration:

SRS-RAN leverages software-defined radio techniques, making it a valuable platform for exploring SDR technology. Users can experiment with different radio configurations and gain insights into the principles of software-defined radi0.

OPEN5GS Installations:

* sudo apt-get install libzmq3-dev
* git clone https://github.com/zeromq/libzmq.git
* cd libzmq
* ./autogen.sh
* ./configure
* make
* sudo make install
* sudo ldconfig
* git clone https://github.com/zeromq/czmq.git
* cd czmq
* ./autogen.sh
* ./configure
* make
* sudo make install
* sudo ldconfig
* git clone https://github.com/srsran/srsRAN\_Project.git
* cd srsRAN\_Project
* mkdir build
* cd build
* cmake ../ -DENABLE\_EXPORT=ON -DENABLE\_ZEROMQ=O

Pay extra attention to the cmake console output. Make sure you read the following line:

...

* + -- FINDING ZEROMQ.
  + -- Checking for module 'ZeroMQ'
  + -- No package 'ZeroMQ' found
  + -- Found libZEROMQ: /usr/local/include, /usr/local/lib/libzmq.so
  + ...
* make -j`nproc`

sudo apt update

sudo apt install -y mongodb-org

sudo systemctl start mongod **(if** '/usr/bin/mongod' is not running**)**

sudo systemctl enable mongod **(**ensure to automatically start it on system boot**)**

sudo ip tuntap add name ogstun mode tun

sudo ip addr add 10.45.0.1/16 dev ogstun

sudo ip addr add 2001:db8:cafe::1/48 dev ogstun

sudo ip link set ogstun up

sudo apt install python3-pip python3-setuptools python3-wheel ninja-build build-essential flex bison git cmake libsctp-dev libgnutls28-dev libgcrypt-dev libssl-dev libidn11-dev libmongoc-dev libbson-dev libyaml-dev libnghttp2-dev libmicrohttpd-dev libcurl4-gnutls-dev libnghttp2-dev libtins-dev libtalloc-dev meson

git clone https://github.com/open5gs/open5gs

cd open5gs

meson build --prefix**=**`pwd`/install

ninja -C build

cd build

meson test -v

cd build

ninja install

cd ../

Lan to Lan

Both sender and receiver PC has same IP adresses

Sender Laptop Open terminal: python -mSimpleHTTPServer

Reciver Laptop Open browser: ipadresses:8000/

* for python installing

sudo apt-get install python

sudo apt-get install python2

**USRP INSTALLATION**

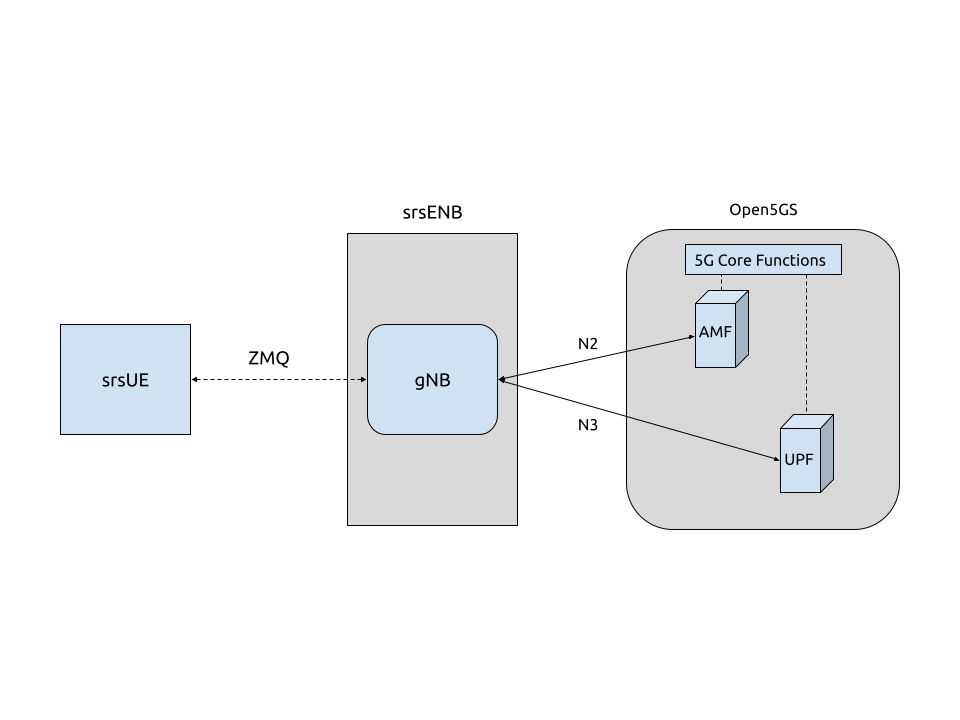
**1)Hardware Setup:**

* Connect the USRP B1200 to your host computer using the appropriate interface (USB or Ethernet).
* Ensure that the device is properly powered.

**2)Software Installation**

* **Driver Installation:**
* Install the UHD (USRP Hardware Driver) software, which is necessary for communicating with and controlling the USRP device. You can find the UHD software on the Ettus Research or GNU Radio websites.
* sudo apt-get update
* sudo apt-get -y install git swig cmake doxygen build-essential libboost-all-dev libtool libusb-1.0-0 libusb-1.0-0-dev libudev-dev libncurses5-dev libfftw3-bin libfftw3-dev libfftw3-doc libcppunit-1.14-0 libcppunit-dev libcppunit-doc ncurses-bin cpufrequtils python-numpy python-numpy-doc python-numpy-dbg python-scipy python-docutils qt4-bin-dbg qt4-default qt4-doc libqt4-dev libqt4-dev-bin python-qt4 python-qt4-dbg python-qt4-dev python-qt4-doc python-qt4-doc libqwt6abi1 libfftw3-bin libfftw3-dev libfftw3-doc ncurses-bin libncurses5 libncurses5-dev libncurses5-dbg libfontconfig1-dev libxrender-dev libpulse-dev swig g++ automake autoconf libtool python-dev libfftw3-dev libcppunit-dev libboost-all-dev libusb-dev libusb-1.0-0-dev fort77 libsdl1.2-dev python-wxgtk3.0 git libqt4-dev python-numpy ccache python-opengl libgsl-dev python-cheetah python-mako python-lxml doxygen qt4-default qt4-dev-tools libusb-1.0-0-dev libqwtplot3d-qt5-dev pyqt4-dev-tools python-qwt5-qt4 cmake git wget libxi-dev gtk2-engines-pixbuf r-base-dev python-tk liborc-0.4-0 liborc-0.4-dev libasound2-dev python-gtk2 libzmq3-dev libzmq5 python-requests python-sphinx libcomedi-dev python-zmq libqwt-dev libqwt6abi1 python-six libgps-dev libgps23 gpsd gpsd-clients python-gps python-setuptools
* cd $HOME
* mkdir workarea
* cd workarea
* git clone https://github.com/EttusResearch/uhd
* cd uhd
* make
* make test
* sudo make install
* sudo ldconfig
* sudo uhd\_images\_downloader
* uhd\_find\_devices.

# 5G SA End-to-End



* cd srsran
* sudo apt-get install libzmq3-dev

1. git clone https://github.com/zeromq/libzmq.git

* cd libzmq
* ./autogen.sh
* ./configure
* make
* sudo make install
* sudo ldconfig
* cd ..

2.git clone https://github.com/zeromq/czmq.git

* cd czmq
* ./autogen.sh
* ./configure
* make
* sudo make install
* sudo ldconfig
* cd ..

3.git clone https://github.com/srsRAN/srsRAN\_4G.git

* cd srsRAN\_4G
* mkdir build
* cd build
* cmake ../
* make

Put extra attention in the cmake console output. Make sure you read the following line:

...

-- FINDING ZEROMQ.

-- Checking for module 'ZeroMQ'

-- No package 'ZeroMQ' found

-- Found libZEROMQ: /usr/local/include, /usr/local/lib/libzmq.so

...

### **Network Namespace Creation:**

Here we are created a network namespace for ue that name is ue1

* sudo ip netns add ue1

for checking that namesapce is created or not

* sudo ip netns list

### Configuration

We need to download some config files:

1. ue.conf
2. rr.conf
3. amf.conf
4. amf.yaml
5. upf.yaml

Need to modify config files:

* In ue.conf

To enable ZMQ the following is added to the UE config file:

[rf]

freq\_offset = 0

tx\_gain = 80

srate = 11.52e6

device\_name = zmq

device\_args = tx\_port=tcp://\*:2001,rx\_port=tcp://localhost:2000,id=ue,base\_srate=11.52e6

FOR network namespace config:

[gw]

netns = ue1

NR Features

[rat.eutra]

dl\_earfcn = 2850

nof\_carriers = 0

[rat.nr]

bands = 3,78

nof\_carriers = 1

[rrc]

release = 15

USIM Credential

[usim]

mode = soft

algo = milenage

opc = 63BFA50EE6523365FF14C1F45F88737D

k = 00112233445566778899aabbccddeeff

imsi = 901700123456780

imei = 353490069873319

APN

[nas]

apn = srsapn

apn\_protocol = ipv4

* enb.conf

Setting PLMN and MME

[enb]

enb\_id = 0x19B

mcc = 901

mnc = 70

mme\_addr = 127.0.0.2

gtp\_bind\_addr = 127.0.1.1

s1c\_bind\_addr = 127.0.1.1

s1c\_bind\_port = 0

n\_prb = 50

ZMQ

[rf]

rx\_gain = 40

tx\_gain = 80

# Example for ZMQ-based operation with TCP transport for I/Q samples

device\_name = zmq

device\_args = fail\_on\_disconnect=true,tx\_port=tcp://\*:2000,rx\_port=tcp://localhost:2001,id=enb,base\_srate=11.52e6

* rr.conf

5G NR

nr\_cell\_list =

(

{

rf\_port = 0;

cell\_id = 1;

root\_seq\_idx = 1;

tac = 7;

pci = 500;

dl\_arfcn = 368500;

coreset0\_idx = 6;

band = 3;

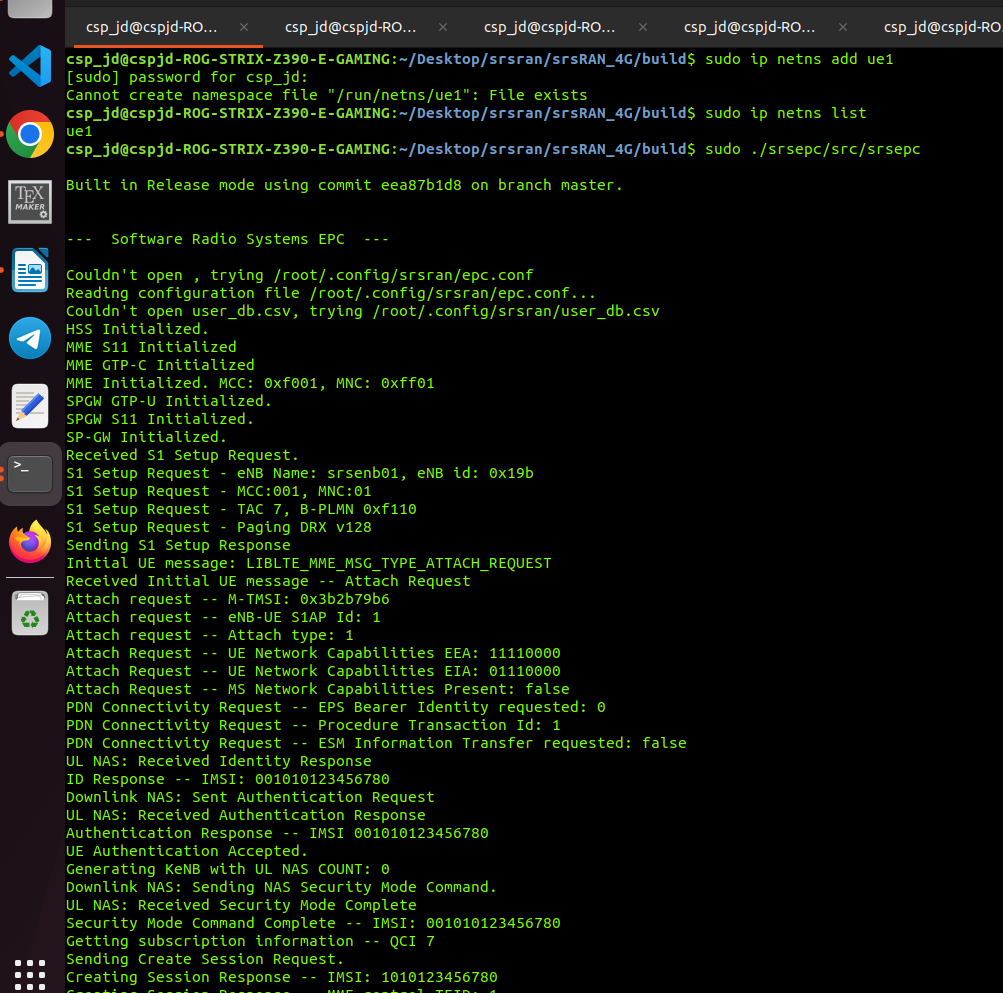
}

);

### Running the EPC:

* sudo ./srsepc/src/srsepc

Output:



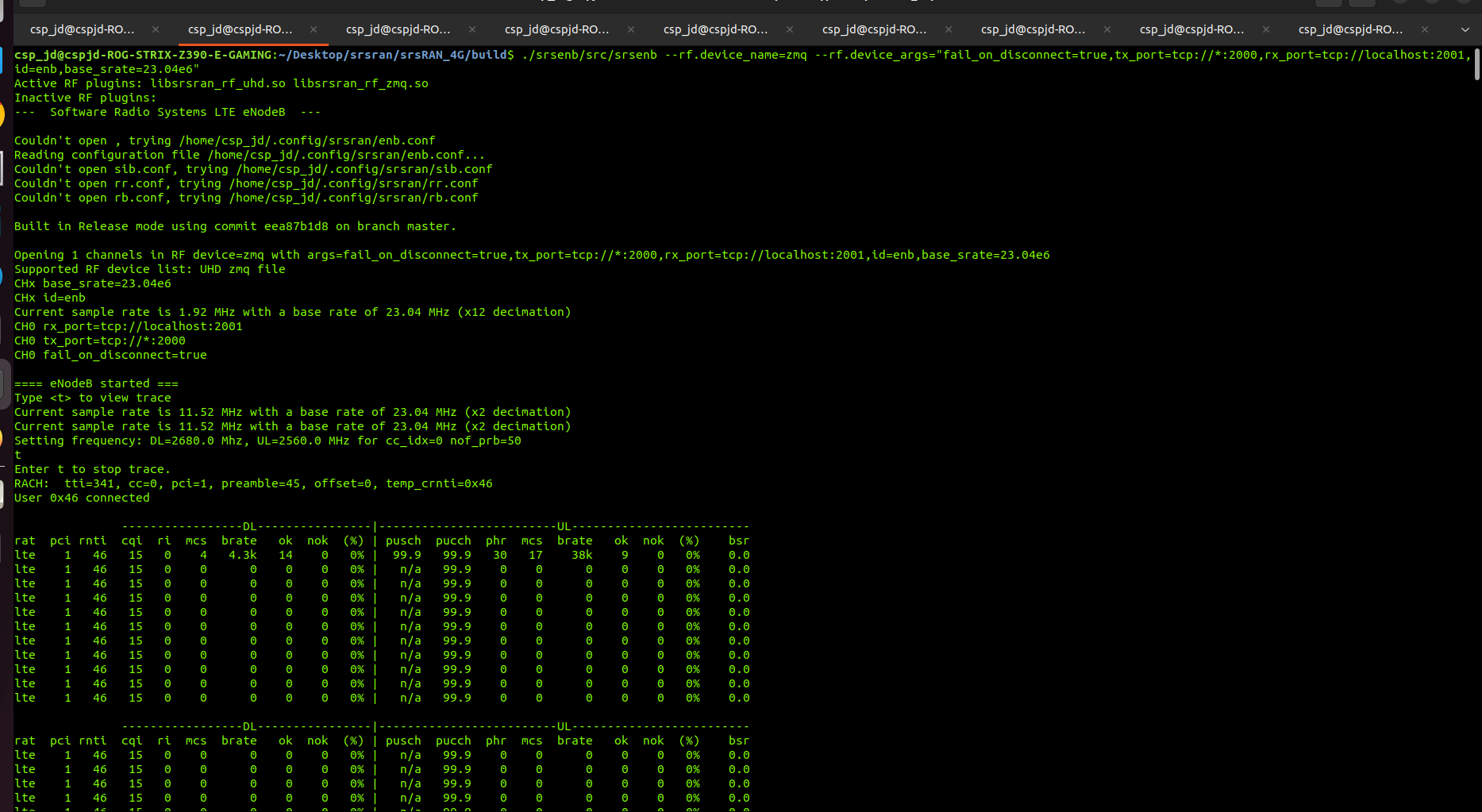
### Running the eNodeB

* ./srsenb/src/srsenb --rf.device\_name=zmq –rf.device\_args="fail\_on\_disconnect=true,tx\_port=tcp://\*:2000,rx\_port=tcp://localhost:2001,id=enb,base\_srate=23.04e6"

if above command is not working then use :

* sudo ./srsenb/src/srsenb --rf.device\_name=zmq –rf.device\_args="fail\_on\_disconnect=true,tx\_port=tcp://\*:2000,rx\_port=tcp://localhost:2001,id=enb,base\_srate=23.04e6"

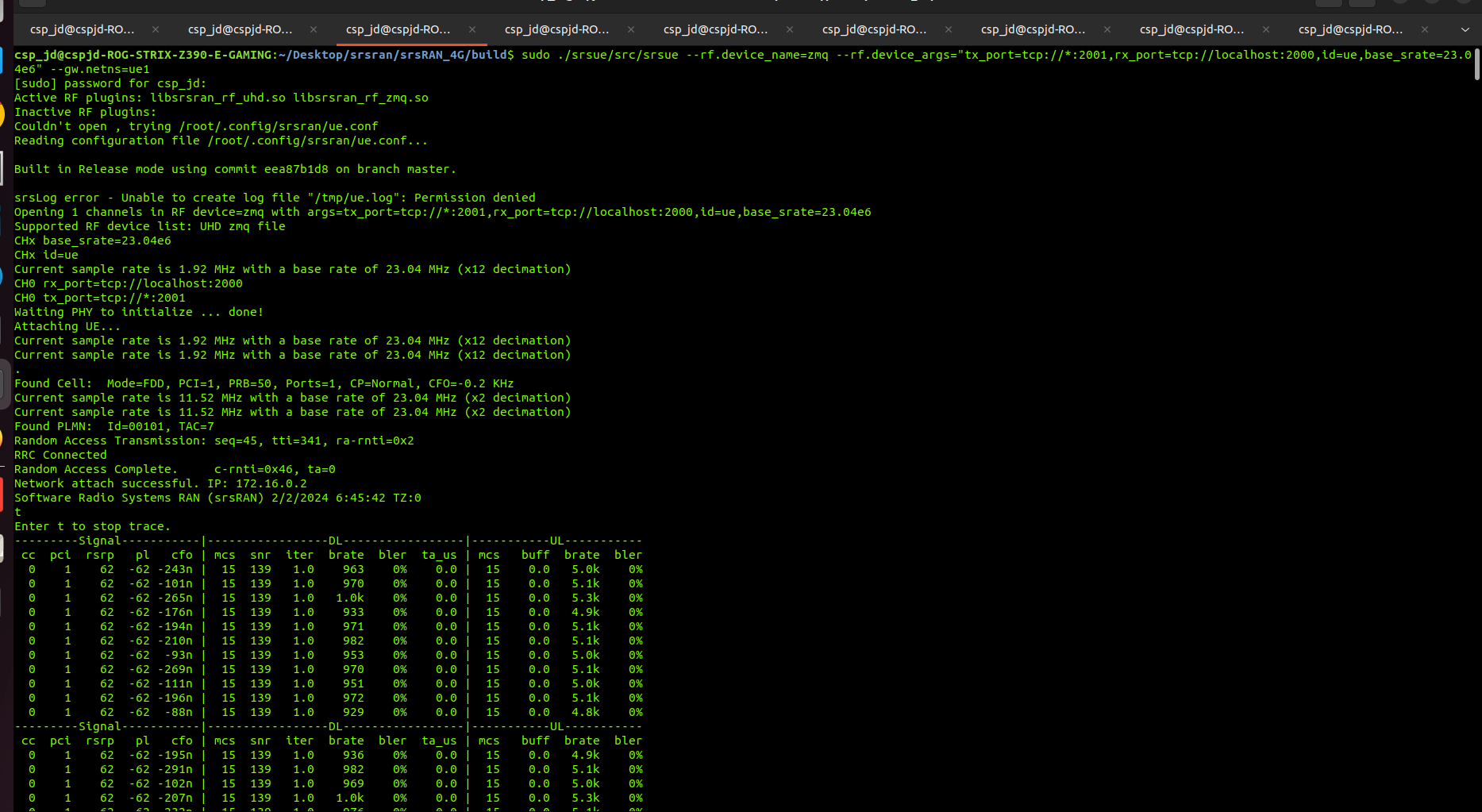
Output:



### Running the UE

* sudo ./srsue/src/srsue --rf.device\_name=zmq --rf.device\_args="tx\_port=tcp://\*:2001,rx\_port=tcp://localhost:2000,id=ue,base\_srate=23.04e6" –gw.netns=ue1

Output:



### Traffic Generation

To exchange traffic in the downlink direction, i.e. from the the EPC, just run ping or iperf as usual on the command line, e.g.:

* ping 172.17.0.1

In order to generate traffic in the uplink direction it is important to run the ping command in the UE’s network namespace.

* sudo ip netns exec ue1 ping 172.16.0.1

### iPerf3

iPerf3 is a tool primarily used for measuring the TCP and UDP network performance by generating traffic between two computers. It allows you to assess the bandwidth, latency, and other network-related metrics between a client and a server.

It is important to start the server first, and then the client.

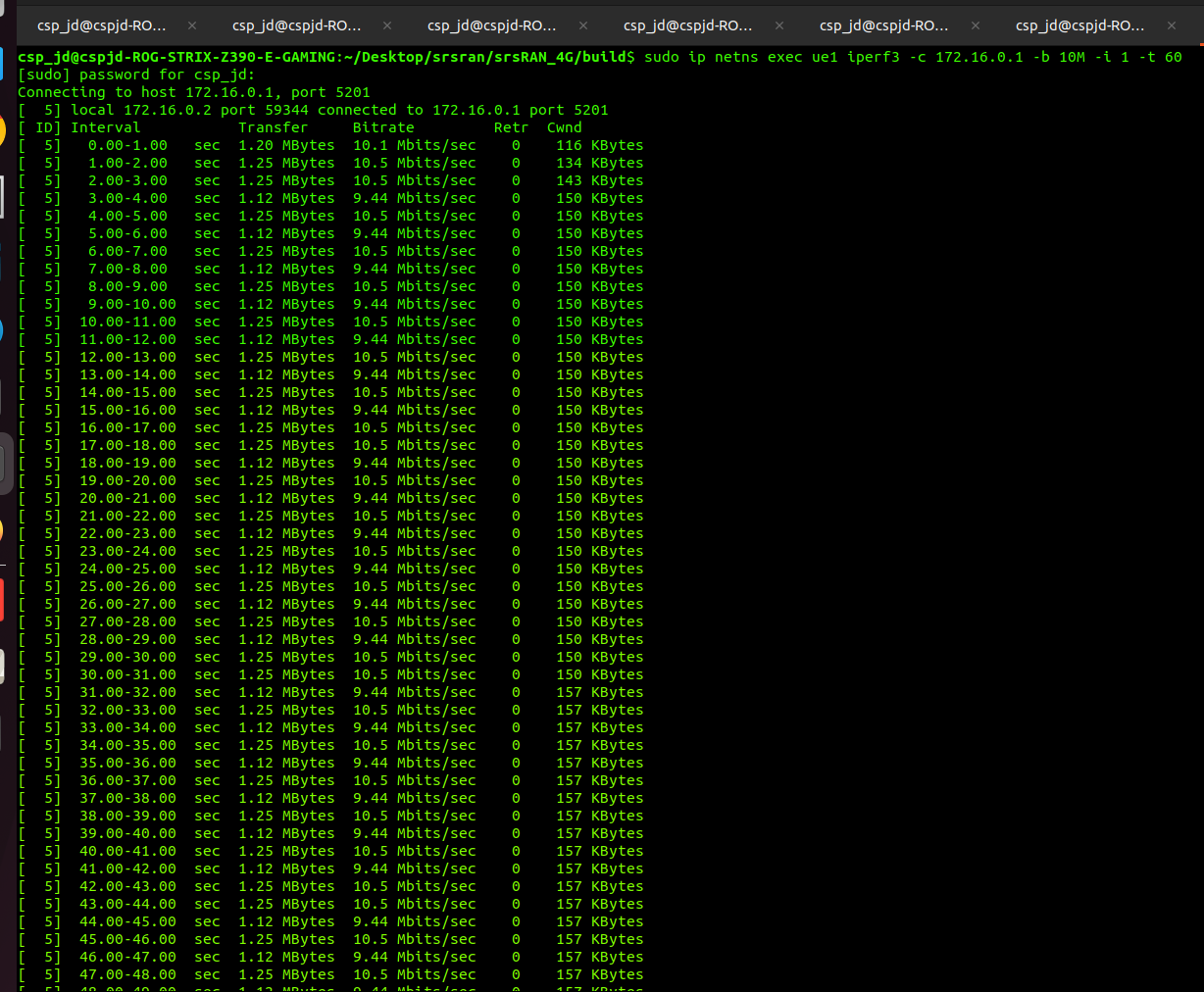
Network-side: Start the iperf server

* iperf3 -s -i 1

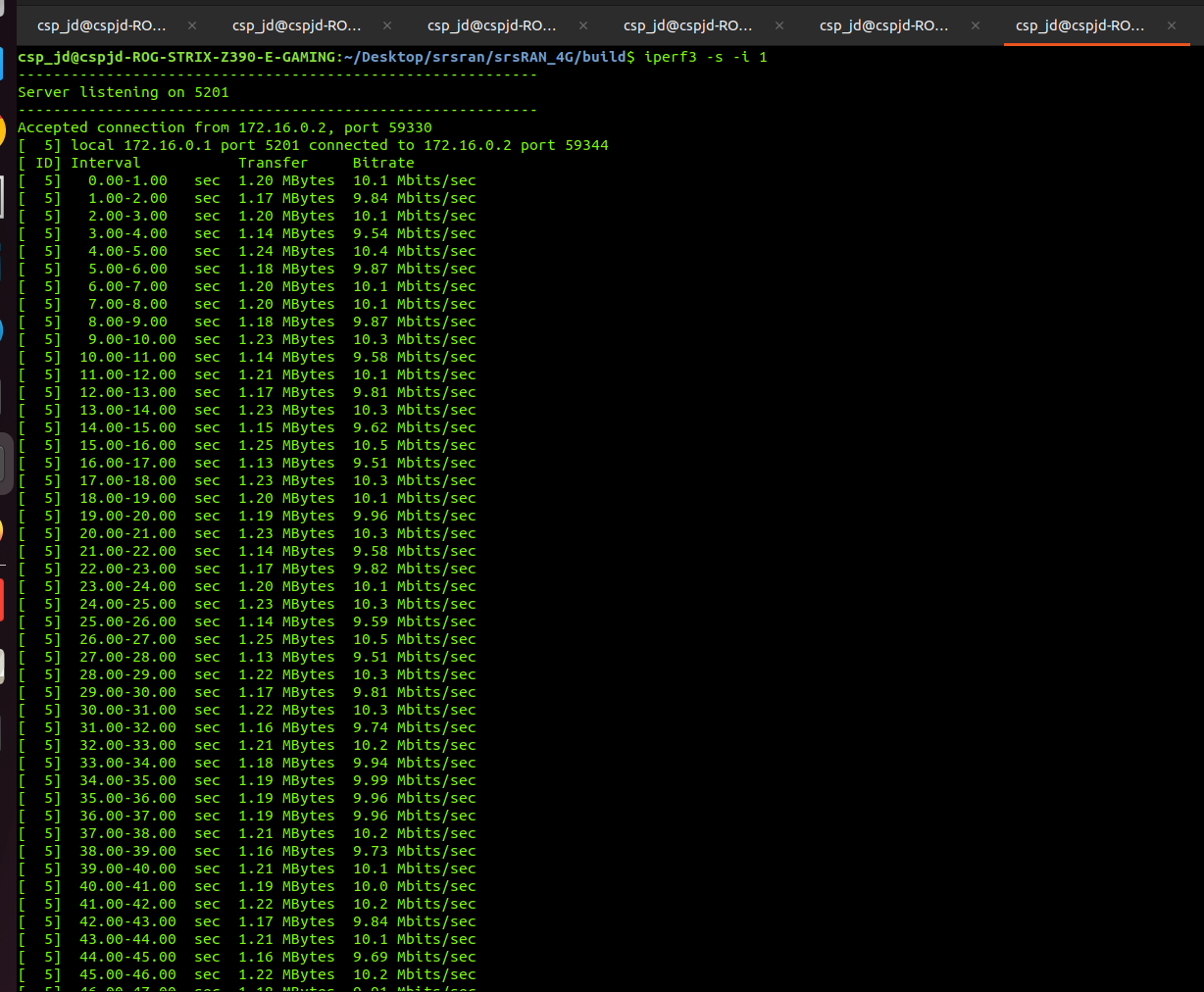
UE side:

* sudo ip netns exec ue1 iperf3 -c 10.45.0.1 -b 10M -i 1 -t 60

Outputs:

Client:

Server:



After all these steps

1.enb terminal enter t for enb trace.

2.ue terminal also enter t for ue trace.

**Wireshark:**

Wireshark is a network packet analyzer. A network packet analyzer presents captured packet data in as much detail as possible.

* sudo wireshark

if wireshark is not installed then :

* sudo apt-get install wireshark
* sudo usermod -a -G wireshark csp\_jd

if you got error like: couldn't run /usr/bin/dumpcap in child process:permission denied

* sudo chmod +x /usr/bin/dumpcap

then again run sudo wireshark

* sudo wireshark
* select interface name
* you will find packets transfermation

