srsRAN Project

The srsRAN Project is a complete 5G RAN solution, featuring an ORAN-native CU/DU developed by SRS.

The solution includes a complete L1/2/3 implementation with minimal external dependencies. Portable across processor architectures, the software has been optimized for x86 and ARM. srsRAN follows the 3GPP 5G system architecture implementing the functional splits between distributed unit (DU) and centralized unit (CU). The CU is further disaggregated into control plane (CU-CP) and user-plane (CU-UP).

See the srsRAN Project for information, guides and project news.

Build instructions and user guides - srsRAN Project documentation.

Community announcements and support - Discussion board.

Features and roadmap - Features.

Build Preparation

Dependencies

- Build tools:
  - cmake: https://cmake.org/
- Mandatory requirements:
  - libfftw: https://www.fftw.org/
  - libsctp: https://github.com/sctp/lksctp-tools
  - yaml-cpp: https://github.com/jbeder/yaml-cpp
  - mbedTLS: https://www.trustedfirmware.org/projects/mbed-tls/
  - googletest: https://github.com/google/googletest/
- Optional requirements:
  - googletest: https://github.com/google/googletest/
    - You can enable test building by using the cmake option `-DBUILD_TESTING=On`. GoogleTest is only mandatory when building with tests.

You can install the build tools and mandatory requirements for some example distributions with the commands below:

Ubuntu 22.04

```
sudo apt-get install cmake make gcc g++ pkg-config libfftw3-dev libmbedtls-
dev libsctp-dev libyaml-cpp-dev
```

Fedora
Arch Linux

Split-8

For Split-8 configurations, either UHD or ZMQ is required for the fronthaul interface. Both drivers are linked below, please see their respective documentation for installation instructions.

- UHD: https://github.com/EttusResearch/uhd
- ZMQ: https://zeromq.org/

Split-7.2

For Split-7.2 configurations no extra 3rd-party dependencies are required, only those listed above.

Optionally, DPDK can be installed for high-bandwidth low-latency scenarios. For more information on this, please see this tutorial.

Build Instructions

Download and build srsRAN:

Vanilla Installation
First, clone the srsRAN Project repository:

```
    git clone https://github.com/srsRAN/srsRAN_Project.git
```

Then build the code-base:

```
    cd srsRAN_Project
    mkdir build
    cd build
    cmake ../
    make -j $(nproc)
```

You can now run the gNB from `srsRAN_Project/build/apps/gnb/`. If you wish to install the srsRAN Project gNB, you can use the following command:

```
    sudo make install
```

ZMQ Enabled Installation
DPDK Enabled Installation

PHY Tests

PHY layer tests use binary test vectors and are not built by default. To enable, see the docs.

Deploying srsRAN Project

srsRAN Project can be run in two ways:

- As a monolithic gNB (combined CU & DU)
- With a split CU and DU

For exact details on running srsRAN Project in any configuration, see the documentation.

For information on configuring and running srsRAN for various different use cases, check our tutorials.

Installation Guide

The following steps need to be taken in order to download and build srsRAN Project:

1. Install dependencies
2. Install RF driver (only required for Split 8 deployments)
3. Clone the repository
4. Build the codebase

Note

srsRAN Project requires a Linux-based OS, we recommend Ubuntu (22.04 or later).

Build Tools and Dependencies

srsRAN Project uses CMake and C++17. We recommend the following build tools:

- cmake
- gcc (v11.4.0 or later) **OR** Clang (v14.0.0 or later)

srsRAN Project has the following necessary dependencies:

- libfftw
- libsctp
- yaml-cpp
- mbedTLS
- googletest

You can install the required build tools and dependencies for various distributions as follows:

Ubuntu 22.04 (or later)FedoraArch Linux

```
sudo apt-get install cmake make gcc g++ pkg-config libfftw3-dev libmbedtls-dev
libsctp-dev libyaml-cpp-dev libgtest-dev
```

It is also recommended users install the following (although they are not required):

- Ccache: This will help to speed up re-compilation
- backward-cpp: This library helps to generate more informative backtraces in the stdout if an error occurs during runtime

---

RF-drivers

Note

UHD and/or ZMQ are only required for Split 8 deployments, if you are planning on using a Split 7.2 deployment you may skip this step.

srsRAN Project uses RF drivers to support different radio types. Currently, only UHD and ZMQ are supported:

- UHD (We recommended the LTS version of UHD, i.e. either 3.15 or 4.0.)
- ZMQ

Clone and Build

srsRAN Project can be built with certain features enabled or disabled. This is done during the build process by using CMake flags and/or by downloading third party dependencies prior to building the code. The following sections outline these various build options.

Vanilla InstallationSplit 7.2 Only ConfigurationSplit 8 Only ConfigurationZMQ Enabled Installation

First, clone srsRAN Project repository:

```
git clone https://github.com/srsRAN/srsRAN_Project.git
```

Then build the code-base:

```
cd srsRAN_Project
mkdir build
cd build
cmake ../
make -j $(nproc)
make test -j $(nproc)
```

You can now run the gNB from `srsRAN_Project/build/apps/gnb/`. If you wish to install srsRAN Project, you can use the following command:

```
sudo make install
```

The Running srsRAN section of the documentation further discusses how to configure and run the gNB application.

Packages

srsRAN Project is available to download directly from packages for various linux distributions. Users looking for a simple installation who do not wish to edit the source code should use the package installation.

UbuntuArch Linux

Ubuntu users can download srsRAN Project packages using the following commands:

```
sudo add-apt-repository ppa:softwareradiosystems/srsran-project
sudo apt-get update
sudo apt-get install srsran-project -y
```

This will install the latest version of srsRAN Project from git.

When installed from packages, example configs for srsRAN Project can be found in `/usr/share/srsran`. For info on these config files, see here

The application can then be run using:

```
sudo gnb -c <config file>
```

PHY testvectors

A number of PHY tests are based on MATLAB generated testvectors. By default, those tests are disabled. The following steps are required to enable them:

1. Download the latest PHY testvector set.
2. Extract the PHY testvectors to their location within the srsRAN working directory:

```
tar -xf phy_testvectors.tar -C /path_to_your_local_repository/srsRAN_Project
```

3. Enable the use of the PHY testvectors by regenerating the CMake build system:

```
cmake -B build -DUSE_PHY_TESTVECTORS=ON
```

4. Rebuild srsRAN Project.

Running srsRAN Project

Note

This guide outlines running srsRAN Project applications in a Split 8 deployment with a USRP, for Split 7.2 deployments see here.

Baseline Requirements

To successfully run an end-to-end network srsRAN Project applications you will need the following:

- A PC with a Linux based OS (Ubuntu 22.04 or later)
- A USRP device
- srsRAN Project (see the Installation Guide)
- A 3rd-party 5G core (we recommend Open5GS)
- A 3rd-party 5G UE

Recommended:

- External clock source

If you plan to connect the gNB to a COTS UE we recommend that you use an external clock source such as an Octoclock or GPSDO that is compatible with your RF-frontend, as the on-board clock within the USRP may not be accurate enough to enable a connection with the UE. This is discussed further in the relevant tutorial.

System Preparation

Before running any of srsRAN Project applications, we recommend tuning your system for best performance. We provide a script to configure known performance parameters:

- srsRAN performance script

The script does the following:

1. Sets the scaling governor to performance
2. Disables DRM KMS polling
3. Tunes network buffers (Ethernet based USRPs only)

Run the script as follows from the main project folder:

```
sudo ./scripts/srsran_performance
```

Running srsRAN Project

srsGNB

If srsRAN Project has been installed using `sudo make install` or installed from packages then you will be able to run the gNB from anywhere on your machine.

If you have built srsRAN Project from source and have not installed it, then you can run the gNB from: `/srsRAN_Project/build/apps/gnb`. In this folder you will find the gNB application binary.

Run the gNB as follows, passing the YAML configuration file:

```
sudo ./gnb -c gnb_rf_b200_tdd_n78_10mhz.yml
```

Run the gNB with `sudo` to ensure threads are configured with the correct priority.

Example configuration files can be found in the `configs/` folder in srsRAN Project codebase. For more information on the configuration files and the available parameters see the configuration reference.

When running, the gNB should generate the following console output:

```
Available radio types: uhd.

--== srsRAN gNB (commit 77be7d339) ==--

[INFO] [UHD] linux; GNU C++ version 9.4.0; Boost_107100; UHD_4.2.0.HEAD-0-g197cdc4f
Making USRP object with args 'type=b200'
Cell pci=1, bw=10 MHz, dl_arfcn=632628 (n78), dl_freq=3489.42 MHz,
dl_ssb_arfcn=632640, ul_freq=3489.42 MHz

==== gNodeB started ===
Type <t> to view trace
```

Entering `t` will enable the console trace, see here for more details.

Configuration parameters can also be passed on the command line. To see the list of options, use:

```
./gnb --help
```

srsCU

If srsRAN Project has been installed using `sudo make install` or installed from packages then you will be able to run srsCU from anywhere on your machine.

If you have built srsRAN Project from source and have not installed it, then you can run srsCU from: `/srsRAN_Project/build/apps/cu`. In this folder you will find the srsCU application binary.

Run srsCU as follows, passing the YAML configuration file:

```
sudo ./srsCU -c cu.yml
```

Run srsCU with `sudo` to ensure threads are configured with the correct priority.

Example configuration files can be found in the `configs/` folder in srsRAN Project codebase. For more information on the configuration files and the available parameters see the configuration reference.

When running, srsCU should generate the following console output:

```
N2: Connection to AMF on 127.0.1.100:38412 completed
F1-C: Listening for new connections on 127.0.10.1:38471...

==== CU started ===
Type <h> to view help
```

Configuration parameters can also be passed on the command line. To see the list of options, use:

```
./srscu --help
```

srsDU

If srsRAN Project has been installed using `sudo make install` or installed from packages then you will be able to run srsDU from anywhere on your machine.

If you have built srsRAN Project from source and have not installed it, then you can run srsDU from: `/srsRAN_Project/build/apps/du`. In this folder you will find the srsDU application binary.

Run srsDU as follows, passing the YAML configuration file:

```
sudo ./srsDU -c du.yml
```

Run srsDU with `sudo` to ensure threads are configured with the correct priority.

Example configuration files can be found in the `configs/` folder in srsRAN Project codebase. For more information on the configuration files and the available parameters see the configuration reference.

When running, srsDU should generate the following console output:

```
Cell pci=1, bw=20 MHz, 1T1R, dl_arfcn=650000 (n78), dl_freq=3750.0 MHz,
dl_ssb_arfcn=649632, ul_freq=3750.0 MHz

Available radio types: uhd and zmq.
[INFO] [UHD] linux; GNU C++ version 9.3.0; Boost_107100; UHD_4.0.0.0-666-g676c3a37
```

```
[INFO] [LOGGING] Fastpath logging disabled at runtime.
Making USRP object with args 'type=b200,num_recv_frames=64,num_send_frames=64'
[INFO] [B200] Detected Device: B210
[INFO] [B200] Operating over USB 3.
[INFO] [B200] Initialize CODEC control...
[INFO] [B200] Initialize Radio control...
[INFO] [B200] Performing register loopback test...
[INFO] [B200] Register loopback test passed
[INFO] [B200] Performing register loopback test...
[INFO] [B200] Register loopback test passed
[INFO] [B200] Setting master clock rate selection to 'automatic'.
[INFO] [B200] Asking for clock rate 16.000000 MHz...
[INFO] [B200] Actually got clock rate 16.000000 MHz.
[INFO] [MULTI_USRP] Setting master clock rate selection to 'manual'.
[INFO] [B200] Asking for clock rate 23.040000 MHz...
[INFO] [B200] Actually got clock rate 23.040000 MHz.
F1-C: Connection to CU-CP on 127.0.10.1:38471 completed

==== DU started ===
Type <h> to view help
```

Entering `t` will enable the console trace, see here for more details.

Configuration parameters can also be passed on the command line. To see the list of options, use:

```
./srsdu --help
```

Grafana Metrics GUI

Note

The GrafanaGUI works with both the monolithic gNB and srsDU applications. The configuration changes required are the same for both applications.

srsRAN Project allows the reporting and visualization of the CU/DU metrics to a Grafana WebUI. This is done through the use of a Docker container that comes as standard with the srsRAN code base, located in the `docker/` folder. This container allows users to bring up the Grafana dashboard in a single command.

Further Reading:

- Grafana Docs

Configuration

To use the Grafana webUI, you will first need to have Docker installed on your system, you will also need to modify the gNB or DU configuration file to allow the reporting of the metrics to the necessary JSON format for use in the webUI.

Docker

Using the Docker Containers included with srsRAN requires `docker compose` to be installed on your system. You can read about `docker compose` here. There are multiple ways to install this, but the most basic way to do so is to install Docker Desktop. For installing Docker Desktop on linux, take a look at the Docker documentation.

Note

We recommend using a Docker Compose V2 or later.

srsRAN Project

To enable the correct reporting of metrics to the Grafana UI, the gNB or DU configuration files needs to be updated to allow the metrics to be output in the correct JSON format and then sent through a udp-socket to the metrics-server, where it can be parsed and displayed correctly by the GUI.

```
metrics:
    enable_json_metrics: true       # Enable reporting metrics in JSON format
    addr: 172.19.1.4                # Metrics-server IP
    port: 55555                     # Metrics-server Port
```

The `addr` and `port` values defined above mirror those set in the `docker-compose.yml` file found in the `/docker` folder. Any changes in these values must be kept consistent across both files.

Launching GUI

To launch the docker image for the Grafana UI, run the following command from the main folder containing srsRAN:

```
sudo docker compose -f docker/docker-compose.yml up grafana
```

The following output should be observed:

```
Creating network "docker_ran" with the default driver
Starting metrics_server ...
Starting metrics_server ... done
Creating grafana        ... done
```

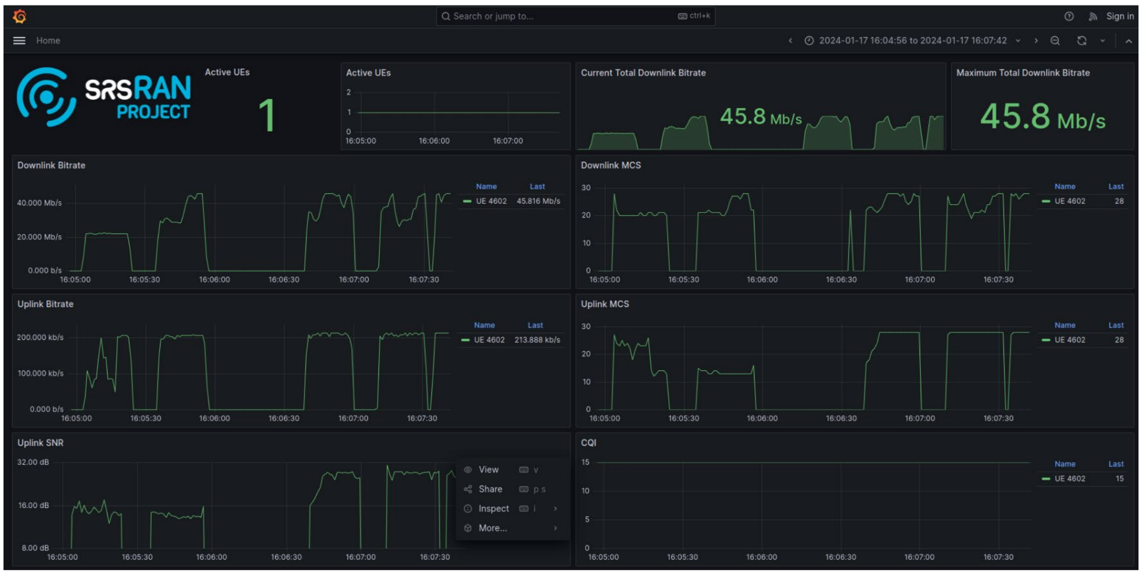
```
Attaching to grafana
```

Navigating to http://localhost:3300/ in your preferred web browser will allow you to view the UI.

You can then run srsRAN Project as normal. As the UE(s) connect to the network you will begin to see an output for each. These figures and graphics will update automatically during runtime, showing plots for each UE on the network.

GUI Output

A sample of the UI output can be seen here:



The above figure shows a single COTS UE connected to the network, with different traffic bursts of varying bandwidth being generated using iPerf. The cell bandwidth is 20 MHz.