ns-3 GSoC 5G NR example requirements

This document provides initial goals and requirements for the 5G NR example tutorial requirements project described in ns-3's 2023 GSoC Project Ideas wiki. If selected to this project, a student will work with the ns-3 mentors to further develop, illustrate, visualize, and document the operations of an initial NR program.

Background and Motivation

The CTTC LENA 5G NR module is maintained as a third-party module for ns-3. The code has been under development for more than a decade, and in many aspects, closely follows the 3GPP specifications for 5G NR (while heavily abstracting other aspects). Because of closely following this specification, there are many configuration aspects to building 5G NR simulations, and the ability to extract data from the simulator or to visualize and understand what is going on, as well as to clearly understand what parts of the standard are implemented exactly or abstracted, is difficult. We want to reduce the learning curve for newcomers to this module.

The ITU has published a specification (M.2412) of how to perform calibration of system-level simulations for IMT-2020. CTTC performed calibration of the NR module against this specification, and wrote a paper about it. This paper and M.2412 provide more background on system-level simulations (ns-3 is an example of a system-level simulator), but that example program (which can be found in the nr module directory nr/examples/3gpp-outdoor-calibration), is too large to be an initial tutorial-level introduction to 5G NR. However, many 5G ns-3 users may want to step from the initial tutorial that we work on in this project to a large-scale example like the calibration program, so it would be nice to connect these programs in some way.

If you are interested to run the calibration program on a small scale (only 1 ring with 7 sites and 21 cells) and explore the output, try this command:

```
./ns3 run "cttc-nr-3gpp-calibration-user --technology=NR
--configurationType=calibrationConf --nrConfigurationScenario=RuralA
--ueNumPergNb=5 --operationMode=TDD --numRings=1 --basicTraces=1
--appGenerationTime=0.5"
```

Instead of this calibration example, we will focus on improving the documentation and output of the existing tutorial example, cttc-nr-demo.cc, which can be found in the "examples" directory of the nr module.

Current User Experience with cttc-nr-demo.cc

Users new to 5G LENA will typically read the manual, and from there be pointed to the cttc-nr-demo.cc example, which is documented as follows.

3.1.12 cttc-nr-demo.cc

The program `examples/cttc-nr-demo` is recommended as a tutorial to the use of the ns-3 NR module. In this example, the user can understand the basics to successfully configure a full NR simulation with end-to-end data transmission.

Firstly, the example creates the network deployment using the GridScenario helper. By default, the deployment consists of a single gNB and two UEs, but the user can provide a different number of gNBs and UEs per gNB.

The operation mode is set to TDD. The user can provide a TDD pattern as input to the simulation; otherwise the simulation will assume by default that downlink and uplink transmissions can occur in the same slot.

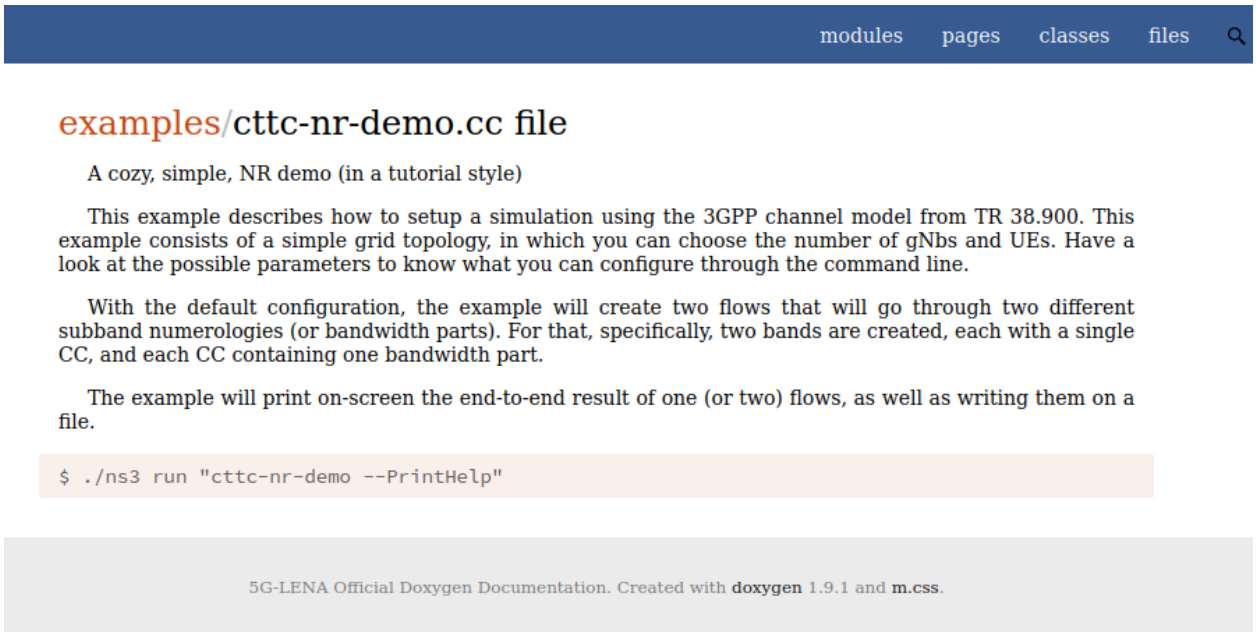
The example performs inter-band Carrier Aggregation of two CC, and each CC has one BWP occupying the whole CC bandwidth.

It is possible to set different configurations for each CC such as the numerology, the transmission power or the TDD pattern. In addition, each gNB can also have a different configuration of its CCs.

The UE can be configure to transmit two traffic flows simultaneously. Each flow is mapped to a single CC, so the total UE traffic can be aggregated.

The complete details of the simulation script are provided in https://cttc-lena.gitlab.io/nr/html/cttc-nr-demo_8cc.html

The Doxygen for this program is as follows:



Users can run this example (also with command-line arguments--not illustrated here):

```
./ns3 run cttc-nr-demo
```

and examine the following Flow Monitor output (which is stored also in a file called 'default'):

```
Flow 1 (1.0.0.2:49153 -> 7.0.0.2:1234) proto UDP
  Tx Packets: 6000
```

```
  Tx Bytes:   768000
  TxOffered:  10.240000 Mbps
  Rx Bytes:   767744
  Throughput: 10.236587 Mbps
  Mean delay:  0.271518 ms
  Mean jitter:  0.030006 ms
  Rx Packets: 5998
Flow 2 (1.0.0.2:49154 -> 7.0.0.3:1235) proto UDP
  Tx Packets: 6000
  Tx Bytes:   7680000
  TxOffered:  102.400000 Mbps
  Rx Bytes:   7671040
  Throughput: 102.280533 Mbps
  Mean delay:  0.835065 ms
  Mean jitter:  0.119991 ms
  Rx Packets: 5993

  Mean flow throughput: 56.258560
  Mean flow delay: 0.553292
```

Desired User Experience for cttc-nr-demo.cc

The cttc-nr-demo.cc program should have its own self-contained tutorial that walks through the following:

1) a step-by-step review/annotation of the program code and also the helpers that it calls
2) documentation of the typical 'path of a packet' through the data plane of the NR (RAN and EPC) stack, and guidance on how it can be traced to find out where a given packet is located at a particular instant in time.
3) guidance on how to manage the configuration parameters of the simulation
4) documentation of the available tracing (from the NR/LTE models) that can be hooked, possibly with some sample trace sinks. We will likely need to add some new trace sources to best meet our needs with this example.

Some things that are marked as /* TODO */ should be completed. For instance, there is a comment that some kind of plot to depict the layout should be added, and that some of the ns-3 code should be encapsulated in helper methods. Plan to work on those unfinished items.

A richer set of output data and plots (or even visualizations and animations) should be enabled, possibly as command-line options, and possibly as supporting plot scripts.

The tutorial program should be connected, somehow, to the more extensive 3GPP calibration program introduced above. Perhaps figures and data that can be generated on that (larger scale) calibration program should also be enabled for the tutorial program. Perhaps these two

programs should use similar helper methods and configuration management methods. Default configuration can be aligned to the extent possible. That is, we want to ensure that the cttc-nr-demo.cc program is a stepping-stone to a more sophisticated calibrated example.

GSoC code deliverables

Most code changes are expected to be generated to modify cttc-nr-demo.cc, and possibly to introduce new helper functions or supplementary scripts (possibly plotting scripts). Log statements in the module may be extended or improved, and new trace sources may be added. In addition, a new Sphinx document dedicated to this example, including new figures, should be generated.

Merge requests for the project will be generated towards the CTTC LENA OpenSim repositories here.

How to apply for this project

Working on an example program is a bit open-ended-- unlike a project where there is a specific goal to code and test a specific algorithm, in this kind of project, we will work more in an 'agile' or 'continual' mode of software development, where we do not plan out the full project schedule in detail, week-by-week, up front, but instead, we review where we are at from time to time (e.g., weekly) and set a weekly development milestone, review it, generate a merge request, and move on to defining the next milestone.

So, in our application template, where we are asking for 'Approach' and 'Plan', we would like to hear from you as to your assessment of what could be improved with the user experience with this program, and what kind of things would you like to prioritize. For instance, can you suggest how we could compress blocks of code into much simpler constructs, while preserving existing functionality? The present file is 695 lines long-- if we could cut it down to, say, 300 lines, what might it look like (preserving all existing functionality)?

We are not interested in a week-by-week plan that states, for example, that in week 8, you will work on something very specific-- we know that such a detailed schedule will not be possible to plan yet. Instead, we would like to see a list of things that you think could be improved, possibly with some sample APIs sketched out, and to have a look at the 3GPP calibration example also and describe how the demo program might be morphed into a very simplified version of the calibration example (and whether there are ideas about simplifying the calibration example without changing output).

Patch requirement

The purpose of the patch requirement is to demonstrate to the reviewers that you can code at a productive level. There is no specific requirement, but we can suggest a few possibilities below (note, you only have to pick one, not all of these suggestions):

1) In the cttc-nr-demo.cc program, this comment exists:

```
/*
* TODO: Add a print, or a plot, that shows the scenario.
*/
```

Write some code to dump node locations to an output file, and then write a script (such as gnuplot or matplotlib) to create a plot of locations, and submit the plotting script and the diff of the code changes to cttc-nr-demo.cc.

2) The cttc-nr-demo.cc program only prints the output of the IP flow monitor observation of the two traffic flows. The two flows originate packets from the UdpClient nodes (the 'remoteHost' in the scenario) and packets are received on the UEs.

There are some reported delay values for the two flows, as observed at the IP layer:

```
Mean delay:  0.271518 ms
Mean jitter:  0.030006 ms

Mean delay:  0.835065 ms
Mean jitter:  0.119991 ms
```

However, it is not that straightforward to get similar values out of the NR stack. For the patch requirement, please add a trace source to the NrGnbNetDevice to trace all packets arriving at the DoSend() method. The trace should be named 'Send'. It should be hooked to the following traced callback that you can add to nr-gnb-net-device.h:

```
TracedCallback<Ptr<const Packet>> m_sendTrace;
```

For guidance, look at how the 'MacTx' trace of CsmaNetDevice is implemented.

If you get this far and want to go further, add a trace sink function to cttc-nr-demo.cc (or two functions if want to separately handle the two flows) and hook it to this new trace, and have the trace sink print out something interesting with the data that it collects (such as the number of packets that it observed, which should be 6000 for each flow).

3) Provide a patch to fix some open issue in the ns-3-dev tracker

In your application, provide a URL to a GitLab or GitHub branch, or commit, or snippet or Gist that points to your code.

Additional ideas for this GSoC project from Biljana Bojovic

The following were suggested to me by Biljana as possible things to work on to improve the examples. Not all of these would necessarily be part of GSoC, but hopefully gives a flavor of the type of improvements we would like to make. I am copying her text below:

An example that configures just a single packet is cttc-3gpp-channel-simple-ran, and the test does the same and it checks the delay at each layer of the protocol stack is: nr-test-numerology-delay.cc, not sure if some of these two scripts could be useful for the lifecycle of a packet.

Not sure if there could be some simplification at the time of configuring the parameters of the simulation. E.g. if there could be some function that would add all the command line parameters that would directly access to the parameter and change it, e.g. ns3::NrUePhy::TxPower, so that for all the examples we could easily add all these NR direct parameter paths to the command line, so that when the user does ***PrintHelp***, they can easily and directly access to almost all the parameters from the NR protocol stack. It could do even without these being added to the command line, but I think that it would be easier for the user if he/she would see all of these paths when the user calls PrintHelp.

I think that we don't explain anywhere in the documentation that when you use hexagonal-grid-scenario-helper.cc, such as examples lena-lte-comparison and 3gpp-outdoor-calibration, that at the moment when is called the function CreateScenario that is generated the gnuplot file of the scenario by the function PlotHexagonalDeployment. So, the user would just need to run gnuplot and plot it, but maybe we could help the user by somehow giving these instructions to run gnuplot on a corresponding file to generate the .png/.pdf of the scenario. GridScenarioHelper lacks the doxygen documentation and it would be great if it would also have a function for plotting the scenario.

Another thing that I don't like in the examples and I did not have chance to work on it is that lena-lte-comparison and 3gpp-outdoor-calibration have some DB stats files in common.

E.g. The following files are identical in the two folders:

- flow-monitor-output-stats.cc
- flow-monitor-output-stats.h
- power-output-stats.cc
- power-output-stats.h
- rb-output-stats.cc
- rb-output-stats.h
- sinr-output-stats.cc
- sinr-output-stats.h
- slot-output-stats.cc
- slot-output-stats.h

These files should be "**git moved**" from lena-lte-comparison folder, where they were originally created first, into a new subfolder of examples, e.g. folder could be named stats or something similar. The idea of the folder would be to gather scripts/classes that help write typical examples statistics into the database. The same files would need to be removed from 3gpp-outdoor-calibration, and CMakeLists.txt should be accordingly updated.

Also common and identical functions from lena-lte-comparison/lena-v2-utils.cc and 3gpp-outdoor-calibration/cttc-nr-3gpp-calibration-utils-v2.cc, such as:

ReportSinrNr
ReportPowerNr
ReportSlotStatsNr
ReportRbStatsNr
ReportGnbRxDataNr

should be moved into a new common file that would be in examples/stats folder, because these functions are basically TraceSink functions whose role is to write the new trace into the database by using some of the stats classes from above, i.e., there are 5 functions for 5 different output stats classes.

In example lena-lte-comparison there are two almost similar .cc files (lena-lte-comparison-user.cc and lena-lte-comparison-campaign.cc), that were needed in past because of some compatibility with SEM, but I think that is time to remove one of them, they are basically at this point doing the same, they should be merged. Once they are merge, also the resulting example, e.g. lena-lte-comparison-user.cc could be merged with lena-lte-comparison.cc.

Once these traces are available to be used for all the examples of the NR module, then cttc-nr-demo could be extended with the possibility to export the results into the databases by using these DB stats classes.

Then, the plotting scripts could be created in the examples module sub-folder that would plot the results from these tables in the database, and also some other scripts should be added to print results from some of the NR traces files.

Finally, we received feedback regarding the complexity of the configuring 3GPP scenario parameter in NR examples. e.g., cttc-3gpp-channel-example, L99-L135 is too complicated for something that should basically be a 'string' parameter. Make it easier to create a default CcBwpCreator::SimpleOperationBandConf. Only specify bandwidth and frequency, use 1 CC and some default ("RMa"?) scenario.