Open-Source Prototyping of 5G-and-Beyond Wireless Systems

DESIGN DOCUMENT

Team Number: 15

Client: Professor Zhang

Advisers: Professor Zhang

Team Members: Anh To, Bradley Norman, Elias Zougmore, Haan Zilmer

Roles: Lead Algorithm Designer, Meeting Facilitator, Lead Testing Engineer, Report Manager

Sddec21-15@iastate.edu http://sddec21-15.sd.ece.iastate.edu

Revised: Final Version

Executive Summary

Development Standards & Practices Used

- IEEE and SESC software development standards
- We will constantly communicate for trouble shootings/clarifications
- Show up on time for group meetings and advisor meetings
- We strive to be open about using the open-source platform srsRAN
- Continuous Integration/ Continuous Development

Summary of Requirements

We need to research concepts of 5G and wireless networks. Specifically, how the srsLTE platform implements their 5G network. Research and implement scheduling algorithms in srsLTE and test its performance in an on-campus testbed.

Our advisors gave us four resource documents with different scheduling algorithms that we needed to understand to pick the one that we wanted to implement. We have decided to implement the USC algorithm.

During break, srsLTE code base was heavily modified and renamed srsRAN. Our team decided to take some time to relearn the code base and do more research on c++ since that is the language used for the code base.

The main goal is for us to learn more about 5G scheduling and the software needed to implement it as well as see if implementing the algorithm would improve the scheduling performance of srsRAN.

Applicable Courses from Iowa State University Curriculum

- SE 329
- CPRE/EE 185

New Skills/Knowledge acquired that was not taught in courses

- 5g infrastructure
- Scheduling algorithms
- srsRAN

Table of Contents

List of figures/tables/symbols/definitions

1 Introduction

1.1 Acknowledgement

We would like to acknowledge Professor Zhang for the research resources he has provided us with as well as the advice he provided us with for the project.

1.2 Problem and Project Statement

Problem statement: Advancements in 5G technology have led to an increase in demand for qualified engineers with the ability to develop and prototype advanced wireless solutions. 5G wireless networks are expected to enable not only Gbps mobile connectivity but also machine-type communications for smart agriculture, connected and automated vehicles, smart grid, Industry 4.0, and AR/VR. 5G wireless is projected to reach a market size of $250 billion by 2025, and it has been attracting significant investment from industry and government worldwide. Our project is in a research capacity, so while we will not be solving any specific problem, we will be looking into ways to improve the scheduling algorithm for 5G Systems.

Solution approach: Through this project, team members will get hands on experience with the development and implementation of advanced wireless 5G algorithms. As a part of the project, members will get to use platform technologies such as srsRAN, USRP software defined radios, and small-scale wireless testbeds. Our main work will be done in adjusting and rewriting the Scheduler algorithm in order to improve its functionality.

Project outputs: Experience with platform technologies and testbeds. Knowledge of advanced wireless 5G algorithms, and implementation of these algorithms through at scale wireless testbeds.

1.3 Operational Environment

Our project will operate in a software environment. We will have an open-source software platform (srsRAN) and a testbed containing two software defined radios and two NUC's with srsRAN installed.

1.4 Requirements

Functionality

- Ensures schedule efficiency and must utilize an efficient time allocation process.
- RAN and Mobile core unity ensures communication between base stations and the mobile core.

User Interface: Users of our product must be able to access and use the modified algorithm for academic purposes. For example, future students should be able to understand our code through clear and concise comments. Non-functional requirements

- Research done on 5G wireless Systems.
- Research done into srsRAN base code.
- Full Documentation on srsRAN code base

1.5 Intended Users and Uses

The product will be used for research purposes and other areas of academia.

Therefore, the users will be primarily researchers, educators, and students.

1.6 Assumptions and Limitations

- Assumptions:
  - Our research will be used by other researchers in the 5G systems field.
  - Our test environment will apply to real life situations.
- Limitations:
  - Time
  - Lack of information on the code base we are using
  - Lack of knowledge in the field of advanced wireless algorithms and technologies.
  - Number of people

1.7 Expected End Product and Deliverables

Deliverables: Wireless testbed setup, testbed implementation of an advanced 5G wireless algorithm.

- The team will be implementing a wireless algorithm on a small-scale wireless testbed to gain understanding of algorithm testing using wireless testbeds.

2 Project Plan

2.1 Task Decomposition

- Complete Research
  - Cover given reference materials on 5G
- Tools Setup
  - Set up GitLab

        - CI/CD setup ○

Website setup

        - Testbed setup

- Implement/refine algorithm using srsRAN
    - Begin with simple implementation of wireless algorithm.
    - Begin implementation of advanced wireless algorithms.
    - Analyze testbed results.
    - Refine the algorithm.
    - Retest
- Refine algorithm

2.2 Risks And Risk Management/Mitigation

The nature of the project is purely software based as we are going to be developing a scheduling algorithm. Therefore, our primary risk is code that does not work 100% of the time for all usage. Some of the errors that can cause code to not function as expected are anomalies of the algorithm or user input.

The probability of our code not fitting all scenarios is 1.00 as it is almost impossible for our algorithm to cover everything that 5g can be provided. An obvious mitigation of this is to include as much code testing as possible.

Cybersecurity is another factor involved in many software-based projects, however, since all code we will be using, and writing is open source and for academic purposes we will be sharing information freely with the public and have no need to be concerned with cybersecurity.

2.3 Project Proposed Milestones, Metrics, and Evaluation Criteria

Milestones:

1. Complete research
2. Finish tools setup
3. Implement simple code on platform
4. Prototype of algorithm
5. Finished algorithm

Progress metric will be time spent on the given task.

2.4 Project Timeline/Schedule

Figure 1: spring semester Ghant chart

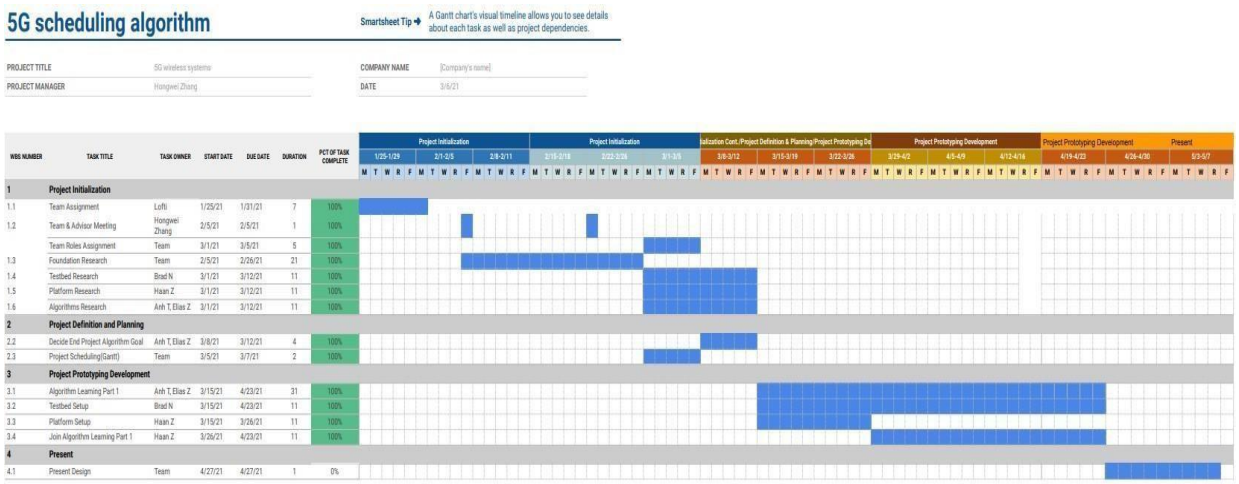
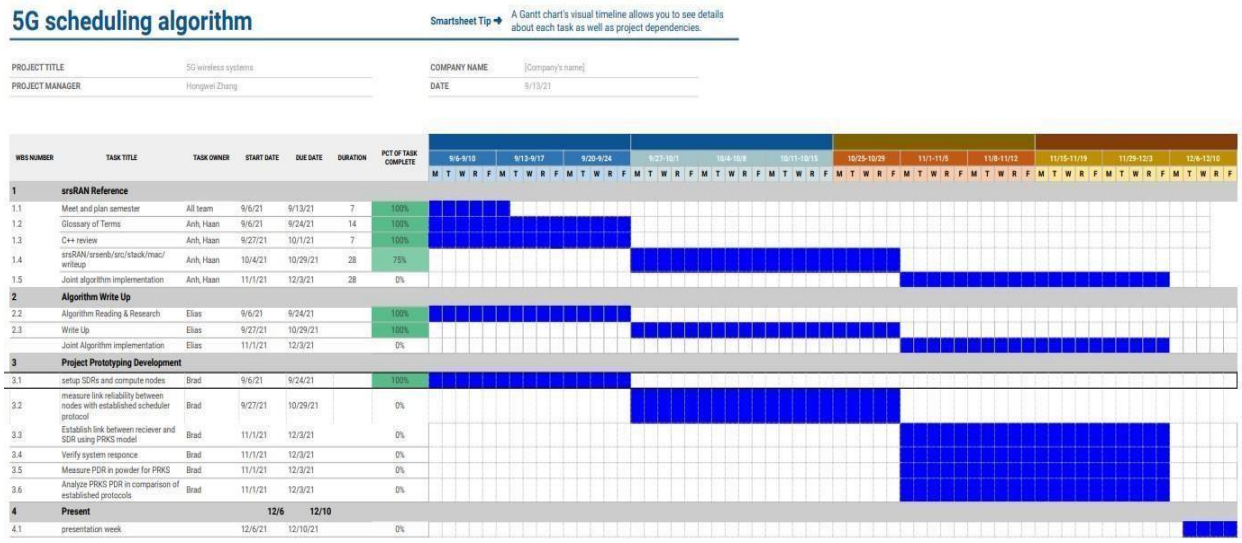


Figure 2: Fall semester Ghant chart



2.5 Project Tracking Procedures

GitHub, MS Team, Gantt Chart

2.6 Personnel Effort Requirements

Members	Anh To	Brad Norman	Haan Zilmer	Elias Zougmore
Time (Coding/Testing) Hours/Week	6hr	6hr	6hr	6hr

2.7 Other Resource Requirements

- Internet access
- Personal computers
- srsRAN
- Reference Materials
- wireless testbed
- UCS scheduling
- Physical SDRs

2.8 Financial Requirements

We have no hardware involved in our project that the university does not already own. The only hardware needed will be provided by our professor and is already a part of his research.

3 Design

3.1 Previous Work and Literature

There are similar products everywhere in 5G wireless systems as each system needs a scheduler in order to function. For example, IEEE put out an article about a scheduler used with time reversal theory and downlink user selection algorithm. We will also be following a paper on scheduling written in part by our professor for

base theory for our scheduler. Our project will differ from each individual theory proposed in the literature by combining ideas from each source.

3.2 Design Thinking

Throughout this project, team members will get hands-on experience with the implementation of advanced 5G algorithms. We decided to work on srsRAN source code through an on-campus testbed. This decision came up with implementing the static part first. We also thought about working on the mobile aspect of the transmission, but it will bring more interference between base stations. Working in the static part, we spent most of the time working on the algorithm implementation and working srsRAN. We proposed implementing the UCS algorithm. A Unified Cellular Scheduling (UCS) is a framework based on the Physical-Ratio-K (PRK) interference model that schedules uplink, downlink, and Device to Device (D2D) transmissions to ensure predictable communication. To ensure that a maximal set of non-interfering links are scheduled to transmit at each carrier and each time slot. To be more specific, the unified scheduler is based on the ONAMA TDMA scheduling algorithm which is adapted to traffic demand. Since The ONAMA algorithm is designed for single carrier wireless networks, it has been extended to consider the specific cellular network such as multiple carriers, base stations, user equipment's and traffic demands. We have planned out what we will do throughout the semester, and we have a Gantt chart that helps us follow up on what to do or when they are due.

3.3 PROPOSED DESIGN

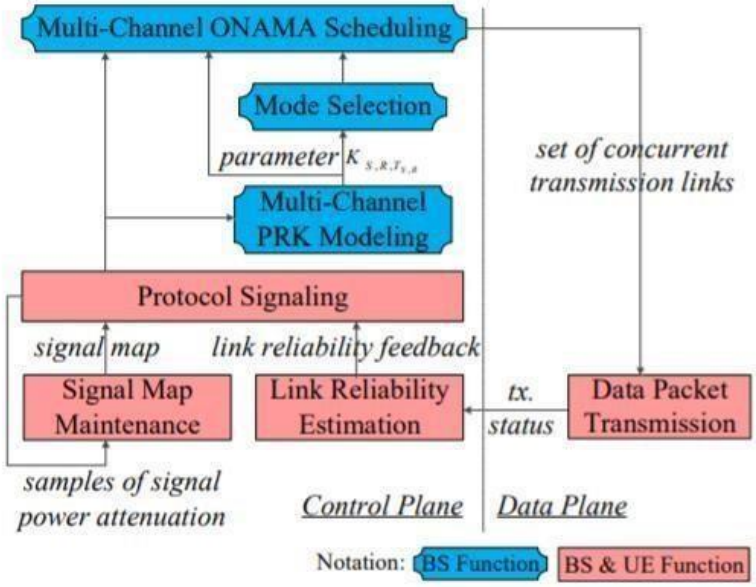


Figure 3. UCS architecture

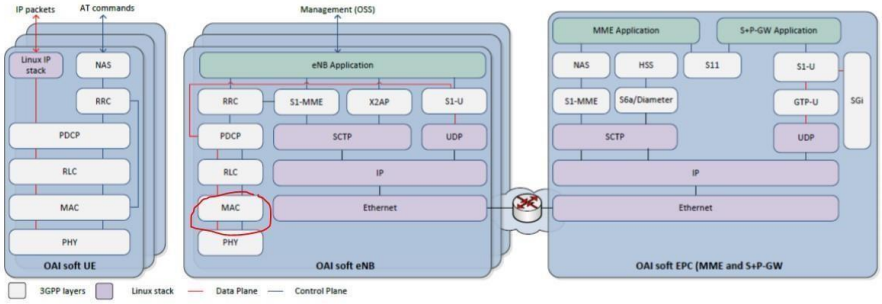


Figure 4. UCS implementation in MAC layer

In order to approach this problem, we are thinking about working on the static part first. We tried implementing the source code for the srsRAN mac so we can know what to modify or not. We mostly worked don the Unified scheduler algorithm which is in C++ language. The source code only works with Linux, so we are using a virtual machine to run the code.

The PRK model unifies the scheduling of uplinks, downlinks and D2D links in cellular networks and provides a unified framework about interferences related to wireless network scheduling. With the unified control signaling, the node

estimates the channel gain and uses signal map to record the estimated results. To have a reliable estimate, the user equipment is used to the minimum so it can share information (signal map, estimates) with its respective base stations. Then the base station collects the data needed to estimate the PRK model parameters. One K is maintained for each channel so a node can transmit the average channel gain between itself and neighbors. K should be properly selected to guarantee accuracy of interference control and avoid control signaling overhead. The PRK model decides which link can or cannot transmit concurrently so the task of mode selection for UE to UE is decided whether the two shall communicate with each other directly or through base stations.

3.4 Technology Considerations

We will be using two software defined radios, and two NUC compute nodes with srsRAN as our testbed for this project. The alternative would be utilizing an at scale testbed located in Salt Lake City called POWDER. See appendix 1, USRP hardware driver and srsRAN installation guide.

3.5 Development Process

We used an Agile development process. For our project we will need to have the ability to adapt our planning and code to the results of our tests. The waterfall method would not work for us as it does not allow us to adjust our algorithm to meet the test requirements, and the TDD method does not apply as our testing outcomes will be an analysis of the system functionality.

3.6 Design Plan

Our design will be different than most projects, as our project is solely research based. Therefore, our design will be focused on learning how our algorithm will function when compared to other algorithms. With that in mind the design plan for our scheduler algorithm will be testing and researching how the theory of uplink, downlink and Device to Device work in a unified manner to utilize the available wireless communication carriers.

4 Testing

The algorithm will be tested using two software defined radios with two compute nodes with srsRAN software suite installed. The default scheduling algorithms for srsRAN are proportional fair and round robin. In order to test our algorithm, the default scheduling algorithm will be altered, and a comparison between the two algorithms will be required. In order to achieve this, inter-channel interference will need to be simulated within the code base by deliberately inserting packet loss. The team determined that an iperf simulation would measure the required network parameters to verify performance.

4.1 Interface Testing

Interfacing with srsRAN is done through Linux's command lines in both the user equipment and base station equipment shells accessed through the NUC computer node. In order to test the functionality of our algorithm we will require an interface between at least one base station and one or more static or mobile user equipment, this will be done with software defined radios located in the testbed. In order to test the scheduling algorithm, the packet loss between the user equipment and the base station will be measured by measuring the packets which are transmitted from the IP address of the base station srs interface and comparing with the packets received at the IP address of the user equipment srs interface.

4.2 Acceptance Testing

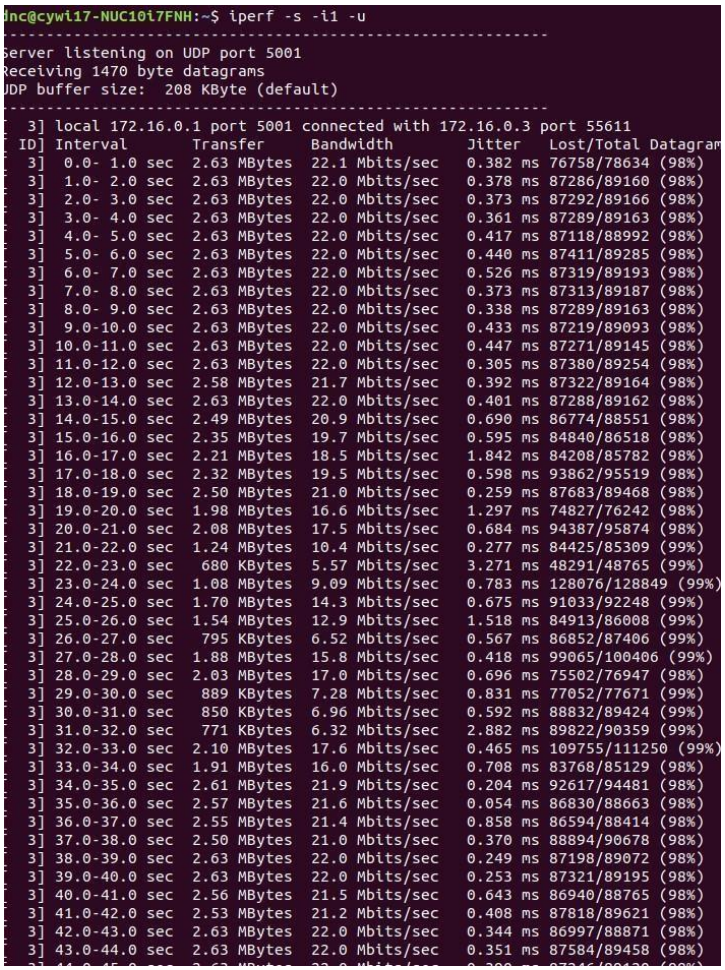
The algorithm’s functional requirement is that it performs the frequency division multiplexing for the User Equipment's within the experiment's domain. A failure of the algorithm will result in interference between communication channels, and packet loss greater than that of the default scheduling algorithm. The inability for the base station to interface with the User Equipment, slow data rates, mis allocation of resources between User Equipment and base station under test are also indications of failure.

4.3 Results

The results depicted in figure 5 are that of a UDP iperf simulation for the default srsRAN scheduling algorithm. The connection is made between the base station

and user equipment software defined as radios. Bandwidth and lost datagrams are measured at the srs interfaces of the SDR's.

Figure 5: UDP iperf simulation results



```
dnc@cywi17-NUC10i7FNH:~$ iperf -s -i1 -u
------------------------------------------------------------
Server listening on UDP port 5001
Receiving 1470 byte datagrams
UDP buffer size:  208 KByte (default)
------------------------------------------------------------
[  3] local 172.16.0.1 port 5001 connected with 172.16.0.3 port 55611
[ ID] Interval       Transfer     Bandwidth        Jitter   Lost/Total Datagram
[  3]  0.0- 1.0 sec  2.63 MBytes  22.1 Mbits/sec   0.382 ms 76758/78634 (98%)
[  3]  1.0- 2.0 sec  2.63 MBytes  22.0 Mbits/sec   0.378 ms 87286/89160 (98%)
[  3]  2.0- 3.0 sec  2.63 MBytes  22.0 Mbits/sec   0.373 ms 87292/89166 (98%)
[  3]  3.0- 4.0 sec  2.63 MBytes  22.0 Mbits/sec   0.361 ms 87289/89163 (98%)
[  3]  4.0- 5.0 sec  2.63 MBytes  22.0 Mbits/sec   0.417 ms 87118/88992 (98%)
[  3]  5.0- 6.0 sec  2.63 MBytes  22.0 Mbits/sec   0.440 ms 87411/89285 (98%)
[  3]  6.0- 7.0 sec  2.63 MBytes  22.0 Mbits/sec   0.526 ms 87319/89193 (98%)
[  3]  7.0- 8.0 sec  2.63 MBytes  22.0 Mbits/sec   0.373 ms 87313/89187 (98%)
[  3]  8.0- 9.0 sec  2.63 MBytes  22.0 Mbits/sec   0.338 ms 87289/89163 (98%)
[  3]  9.0-10.0 sec  2.63 MBytes  22.0 Mbits/sec   0.433 ms 87219/89093 (98%)
[  3] 10.0-11.0 sec  2.63 MBytes  22.0 Mbits/sec   0.447 ms 87271/89145 (98%)
[  3] 11.0-12.0 sec  2.63 MBytes  22.0 Mbits/sec   0.305 ms 87380/89254 (98%)
[  3] 12.0-13.0 sec  2.58 MBytes  21.7 Mbits/sec   0.392 ms 87322/89164 (98%)
[  3] 13.0-14.0 sec  2.63 MBytes  22.0 Mbits/sec   0.401 ms 87288/89162 (98%)
[  3] 14.0-15.0 sec  2.49 MBytes  20.9 Mbits/sec   0.690 ms 86774/88551 (98%)
[  3] 15.0-16.0 sec  2.35 MBytes  19.7 Mbits/sec   0.595 ms 84840/86518 (98%)
[  3] 16.0-17.0 sec  2.21 MBytes  18.5 Mbits/sec   1.842 ms 84208/85782 (98%)
[  3] 17.0-18.0 sec  2.32 MBytes  19.5 Mbits/sec   0.598 ms 93862/95519 (98%)
[  3] 18.0-19.0 sec  2.50 MBytes  21.0 Mbits/sec   0.259 ms 87683/89468 (98%)
[  3] 19.0-20.0 sec  1.98 MBytes  16.6 Mbits/sec   1.297 ms 74827/76242 (98%)
[  3] 20.0-21.0 sec  2.08 MBytes  17.5 Mbits/sec   0.684 ms 94387/95874 (98%)
[  3] 21.0-22.0 sec  1.24 MBytes  10.4 Mbits/sec   0.277 ms 84425/85309 (99%)
[  3] 22.0-23.0 sec   680 KBytes  5.57 Mbits/sec   3.271 ms 48291/48765 (99%)
[  3] 23.0-24.0 sec  1.08 MBytes  9.09 Mbits/sec   0.783 ms 128076/128849 (99%)
[  3] 24.0-25.0 sec  1.70 MBytes  14.3 Mbits/sec   0.675 ms 91033/92248 (99%)
[  3] 25.0-26.0 sec  1.54 MBytes  12.9 Mbits/sec   1.518 ms 84913/86008 (99%)
[  3] 26.0-27.0 sec   795 KBytes  6.52 Mbits/sec   0.567 ms 86852/87406 (99%)
[  3] 27.0-28.0 sec  1.88 MBytes  15.8 Mbits/sec   0.418 ms 99065/100406 (99%)
[  3] 28.0-29.0 sec  2.03 MBytes  17.0 Mbits/sec   0.696 ms 75502/76947 (98%)
[  3] 29.0-30.0 sec   889 KBytes  7.28 Mbits/sec   0.831 ms 77052/77671 (99%)
[  3] 30.0-31.0 sec   850 KBytes  6.96 Mbits/sec   0.592 ms 88832/89424 (99%)
[  3] 31.0-32.0 sec   771 KBytes  6.32 Mbits/sec   2.882 ms 89822/90359 (99%)
[  3] 32.0-33.0 sec  2.10 MBytes  17.6 Mbits/sec   0.465 ms 109755/111250 (99%)
[  3] 33.0-34.0 sec  1.91 MBytes  16.0 Mbits/sec   0.708 ms 83768/85129 (98%)
[  3] 34.0-35.0 sec  2.61 MBytes  21.9 Mbits/sec   0.204 ms 92617/94481 (98%)
[  3] 35.0-36.0 sec  2.57 MBytes  21.6 Mbits/sec   0.054 ms 86830/88663 (98%)
[  3] 36.0-37.0 sec  2.55 MBytes  21.4 Mbits/sec   0.858 ms 86594/88414 (98%)
[  3] 37.0-38.0 sec  2.50 MBytes  21.0 Mbits/sec   0.370 ms 88894/90678 (98%)
[  3] 38.0-39.0 sec  2.63 MBytes  22.0 Mbits/sec   0.249 ms 87198/89072 (98%)
[  3] 39.0-40.0 sec  2.63 MBytes  22.0 Mbits/sec   0.253 ms 87321/89195 (98%)
[  3] 40.0-41.0 sec  2.56 MBytes  21.5 Mbits/sec   0.643 ms 86940/88765 (98%)
[  3] 41.0-42.0 sec  2.53 MBytes  21.2 Mbits/sec   0.408 ms 87818/89621 (98%)
[  3] 42.0-43.0 sec  2.63 MBytes  22.0 Mbits/sec   0.344 ms 86997/88871 (98%)
[  3] 43.0-44.0 sec  2.63 MBytes  22.0 Mbits/sec   0.351 ms 87584/89458 (98%)
```

5 Implementation

For this semester, we used the implemented code of srsRAN (the well commented code) to continue implementation of the unified scheduler algorithm. We needed to understand the code better to some extent that it makes changing the code easier. The objective of the unified scheduler is to schedule data transmission so that most set of non-interfering links are scheduled to transmit carrier and time slot. The Physical-Ratio-k interference model, used for developing field deployable scheduling, is extended here for multiple channel settings while the UCS places the scheduling decisions at each base station and having the user equipment share state information with the right base station. The PRK model defines for each link in the set of links provided, an exclusion region around the receiver so the signal

can travel from each base station or user equipment's to the receiver as node with addresses. The algorithm was a little bit of a challenge because the actual code had no comments lines and asking for help was not much of the help since the code is an open source and the code kept changing much of the time. We did write the code with our example of nodes as links for transmission, but we were not able to test it with the actual code. What we were thinking was that someone can pick up where we left off and continue since we have the algorithm and the code, well documented.

6 Closing Material

6.1 Conclusion

Our main goal is adjusting and rewriting the Scheduler algorithm in order to improve its functionality. In order to do this, we implemented the source code of srsLTE and are now working on the testing part of the code implemented. We are using Powder and testbed for the testing. We tried connecting the user equipment to the base station to test the connectivity between them. From there, we will have the adjustment of the source code left to be done, we need to make it, so it meets our requirements.

We have not met our final goal as there were problems that we could not foresee such as the code base changing over break. Another problem that we have seen coming is that we do not have experience of working with srsLTE/srsRAN. Despite these problems, we have been able to do the initial set ups for srsRAN, a prototype of the proposed scheduling algorithm, and multiple write ups for srsRAN.

6.2 References

Z. Wu, B. Wang, C. Jiang and K. J. R. Liu, "Downlink MAC Scheduler for 5G Communications with Spatial Focusing Effects," in *IEEE Transactions on Wireless Communications*, vol. 16, no. 6, pp. 3968-3980, June 2017, doi: 10.1109/TWC.2017.2690432.

Yuwei Xie, Hongwei Zhang, Pengfei Ren, "Unified Scheduling for Predictable Communication Reliability in Cellular Networks with D2D Links," https://www.ece.iastate.edu/~hongwei/group/publications/UCS.pdf

6.3 APPENDICES

1. **USRP Hardware Driver (UHD) and srsRAN Installation Guide**

Environment Setup

Run the following in the virtual machine and UE computer to Disable Frequency Scaling before you install UHD and srsRAN.

sudo apt-get install cpufrequtils sudo gedit

/etc/init.d/cpufrequtils

change:GOVERNOR="performance" sudo systemctl

disable ondemand sudo

/etc/init.d/cpufrequtils restart

UHD Installation

Reference:
https://kb.ettus.com/Building_and_Installing_the_USRP_OpenSource_Toolchain_(UHD_and_GNU_Radio)_on_Linux

On Ubuntu 18.04 system 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 ncursesbin cpufrequtils python-numpy python-numpy-doc python-numpy-dbg pythonscipy python-docutils qt4-bin-dbg qt4-default qt4-doc libqt4-dev

libqt4devbin python-qt4 python-qt4-dbg python-qt4-dev python-qt4-doc python-qt4doc 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 libasound2dev python-gtk2 libzmq3-dev libzmq5 python-requests python-sphinx libcomedidev python-zmq libqwt-dev libqwt6abi1 python-six libgps-dev libgps23 gpsd gpsdclients python-gps python-setuptools

########## In home directory ##############

git clone https://github.com/EttusResearch/uhd

cd uhd git checkout v3.15.0.0 cd host mkdir build

cd build cmake ../ make make test sudo make

install sudo ldconfig

srsRAN Installation

Reference:
https://docs.srslte.com/en/latest/general/source/1_installation.html#packageinstallation

sudo apt-get install build-essential cmake libfftw3-dev libmbedtls-dev libboostprogram-options-dev libconfig++-dev libsctp-dev

#####In home directory######

git clone https://github.com/srsRAN/srsRAN.git cd

srsRAN mkdir build cd build cmake ../ make make

test sudo make install sudo srsran_install_configs.sh

user

After installation, use “sudo srsEPC” and then “sudo srsENB” on the desktop that has srsENB and srsEPC. Then use “sudo srsUE” on the UE desktop. This start the UE and ENB connection.

Use “iperf -c –il –u" on the ENB desktop and “iperf -c ipaddress_UEdesktop –i1 –t60 –u –b 1000M” on the UE desktop to see connection strength between UE and ENB.

2. Other Considerations

After summer break we came back to our code base which had undergone a complete overhaul. So much so that the name was changed from srsLTE to srsRAN. With this overhaul we ran into many problems. The first being had to relearn an entirely new code base. This put us back about a month and a half in our project due to the lack of documentation and resources provided by srsRAN. While our advisor was very knowledgeable about 5g wireless systems, he does not have any firsthand knowledge of the code base we are using. This made it so that we had no place to go look for answers on how the code base functioned. We also struggled greatly with the small size of our group. While most teams are made up of 6 students, we have only 4. This meant that we had to have a single person perform tasks that were meant for two people on multiple occasions just to try and keep up with our project timeline. This of course was at times not possible with our schedules and ended up putting us back further on our project timeline. With all these setbacks we were unable to complete our original project goal, however, since this is a recurring project, we have built resources for future groups so that they do not run into the same problems we did.

3. srsRAN Source code

https://github.com/srsran/srsRAN