



NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

**OPERATIONALIZING AN OPEN SOURCE 5G MOBILE
NETWORK THROUGH PROLIFERATED LOW EARTH
ORBIT SATELLITE NETWORKS**

by

Lucas M. Hoffer

September 2023

Thesis Advisor:

Co-Advisor:

Second Reader:

Chad A. Bollmann

James B. Michael

Darren J. Rogers

Approved for public release. Distribution is unlimited.

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE			<i>Form Approved OMB No. 0704-0188</i>	
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington, DC, 20503.				
1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE September 2023	3. REPORT TYPE AND DATES COVERED Master's thesis		
4. TITLE AND SUBTITLE OPERATIONALIZING AN OPEN SOURCE 5G MOBILE NETWORK THROUGH PROLIFERATED LOW EARTH ORBIT SATELLITE NETWORKS			5. FUNDING NUMBERS REK7F	
6. AUTHOR(S) Lucas M. Hoffer				
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000			8. PERFORMING ORGANIZATION REPORT NUMBER	
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) ONR-333, Arlington, VA, 22203-1995			10. SPONSORING / MONITORING AGENCY REPORT NUMBER	
11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.				
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release. Distribution is unlimited.			12b. DISTRIBUTION CODE A	
13. ABSTRACT (maximum 200 words) Fifth Generation New Radio (5G NR) provides for significant improvement over older mobile network technologies in terms of quality of service, particularly dependable allocation of traffic flows by service type and priority, much lower latency, and higher bandwidth. The DOD's 5G strategy and implementation plan treats 5G as a critical strategic technology and directs the adoption and integration of 5G NR into operational defense communication networks. With the addition of satellite networks (SatNets), 5G can also be leveraged for end-to-end communications in austere and expeditionary environments. This thesis documents a successful design, implementation, and test of a portable open source 5G network with commercial off-the-shelf (COTS) hardware that connects user equipment to a data network backhaul through low Earth orbit (LEO) SatNets which could be setup and utilized for communications in areas with no usable network connections.				
14. SUBJECT TERMS 5G, New Radio, NR, radio access network, RAN, 5G core, 5GC, backhaul, open source, srsRAN, Open5GS, Starlink, satellite network, SatNet, commercial off-the-shelf, COTS, throughput testbed			15. NUMBER OF PAGES 147	
			16. PRICE CODE	
17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT UU	

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. Z39-18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release. Distribution is unlimited.

**OPERATIONALIZING AN OPEN-SOURCE 5G MOBILE NETWORK
THROUGH PROLIFERATED LOW EARTH ORBIT SATELLITE NETWORKS**

Lucas M. Hoffer
Lieutenant Commander, United States Navy
BS, United States Naval Academy, 2014

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

from the

**NAVAL POSTGRADUATE SCHOOL
September 2023**

Approved by: Chad A. Bollmann
Advisor

James B. Michael
Co-Advisor

Darren J. Rogers
Second Reader

Gurminder Singh
Chair, Department of Computer Science

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

Fifth Generation New Radio (5G NR) provides for significant improvement over older mobile network technologies in terms of quality of service, particularly dependable allocation of traffic flows by service type and priority, much lower latency, and higher bandwidth. The DOD's 5G strategy and implementation plan treats 5G as a critical strategic technology and directs the adoption and integration of 5G NR into operational defense communication networks. With the addition of satellite networks (SatNets), 5G can also be leveraged for end-to-end communications in austere and expeditionary environments. This thesis documents a successful design, implementation, and test of a portable open-source 5G network with commercial off-the-shelf (COTS) hardware that connects user equipment to a data network backhaul through low Earth orbit (LEO) SatNets which could be set up and utilized for communications in areas with no usable network connections.

THIS PAGE INTENTIONALLY LEFT BLANK

Table of Contents

1	Introduction	1
1.1	Purpose	1
1.2	Related Work	2
1.3	Research Questions	4
1.4	Thesis Organization	4
2	5G Network Architecture and Functionality	5
2.1	Radio Access Network	5
2.2	Core Network.	12
2.3	User Equipment.	14
2.4	Backhaul	21
2.5	Summary	25
3	5G Cellular Network Testbed	27
3.1	Setup and Implementation.	27
3.2	Radio Access Network	30
3.3	5G Core	35
3.4	User Equipment.	45
3.5	Network Backhaul	54
3.6	Summary	55
4	Results	57
4.1	UE Device Connection	57
4.2	Optimization Testing	71
4.3	Throughput Testing	85
4.4	Application and Use Cases	95
4.5	Observed Network Limitations	102
4.6	Summary	104

5	Conclusions	105
5.1	Significant Contributions	105
5.2	Key Findings	106
5.3	Recommendations for Future Work	107
	 Appendix: Code Snippets	 109
A.1	Network Startup.	109
A.2	Software Defined Radio Dependencies.	110
	 List of References	 115
	 Initial Distribution List	 123

List of Figures

Figure 2.1	MIB and SIB signalling flow. Source [24].	7
Figure 2.2	Simplified random access procedure. Adapted from [22].	8
Figure 2.3	Detailed random access procedure. Adapted from [25].	9
Figure 2.4	RRC state machine and transitions. Source [23].	10
Figure 2.5	Simplified 5G SBA	13
Figure 2.6	Major 5G categorical use cases. Source [30].	15
Figure 2.7	SUCI breakdown. Source [35].	17
Figure 2.8	5G-GUTI. Source [37].	18
Figure 2.9	UE and network identity exchange. Source [38].	19
Figure 2.10	5G key hierarchy generation. Source [36].	20
Figure 2.11	PDU session. Source [39].	21
Figure 2.12	Types of NTN platforms. Source [41].	22
Figure 2.13	Typical 5G NTN satellite integration compared to terrestrial networks. Source [43].	24
Figure 3.1	5G network architecture	28
Figure 3.2	Successful RAN configuration with connection to 5GC	34
Figure 3.3	Open5GS component diagram. Source [54].	35
Figure 3.4	IP table verification	41
Figure 3.5	5GC services running properly as indicated by the operating system	42
Figure 3.6	AMF log file indicating NF registration and successful service startup	42
Figure 3.7	UDM web GUI	43

Figure 3.8	UDM through MongoDB	44
Figure 3.9	sysmocom ISIM. Source [33].	48
Figure 3.10	pySim programming options	50
Figure 3.11	Programming for USIM using pySim-prog.py	51
Figure 3.12	USIM file structure	52
Figure 3.13	Starlink coverage map as of June 2023. Source [44].	54
Figure 4.1	Screen capture of Samsung A22 service mode indicating capable 5G NR bands	59
Figure 4.2	Screen captures of Samsung A22 service mode and SIM card report showing successful network connection	60
Figure 4.3	AMF log extract showing successful Samsung A22 network connection	61
Figure 4.4	SMF log extract showing successful IP address assignment to the Samsung A22	61
Figure 4.5	UPF log extract showing successful UPF session establishment and subsequent IP address assignment to the Samsung A22	61
Figure 4.6	srsRAN gNB trace from a connected Samsung A22	62
Figure 4.7	Speedtest.net test indicating achieved Internet access on Samsung A22 while connected to our 5G network	63
Figure 4.8	Functional diagram of the Quectel modem	64
Figure 4.9	Quectel modem physical and virtual USB connections	66
Figure 4.10	Minicom interface used for changing serial device to USB2	67
Figure 4.11	Successful connection to modem through Minicom	67
Figure 4.12	Quectel modem network detection and connection of our 5G network	69
Figure 4.13	Quectel PDU session establishment	70
Figure 4.14	Registration rejection for unknown user	70

Figure 4.15	Physical testing setup showing the mounted directional antennas on a mobility cart, along with display, SDR, and NUC controller . . .	72
Figure 4.16	Channel used for proximity testing	73
Figure 4.17	MCS modulation order. Source [61].	74
Figure 4.18	Box plot comparison of gNB transmit gain vs received SNR at UE at 1 m	78
Figure 4.19	Box plot comparison of 59 dB, 69 dB, 79 dB gNB transmit gain vs received SNR at UE at 1 m	79
Figure 4.20	SNR Tukey testing results of transmit gain at 1 m	80
Figure 4.21	Box plot comparison of 59 dB, 69 dB, 79 dB gNB transmit gain vs received SNR at UE at 10m	81
Figure 4.22	Tukey testing results comparing 59 dB, 69 dB, 79 dB gNB transmit gain vs received SNR at UE at 10 m of transmit gain at 10 m . . .	81
Figure 4.23	Indoor testing comparison of distance vs RSRP at UE	84
Figure 4.24	Outdoor testing comparison of distance vs RSRP at UE	84
Figure 4.25	Network setup for throughput testing. Adapted from [8].	86
Figure 4.26	Box plot comparison of download speeds using curl and speedtest through SatNet backhaul	89
Figure 4.27	CDF comparison of download speeds using curl and speedtest through SatNet backhaul	90
Figure 4.28	Box plot of latency results from speedtest through SatNet backhaul	91
Figure 4.29	Box plot of upload speeds using speedtest through SatNet backhaul	92
Figure 4.30	CDF of upload speeds using speedtest through SatNet backhaul .	93
Figure 4.31	Comparison of throughput vs distance between UE and gNB . . .	94
Figure 4.32	ATAK public server connection	97
Figure 4.33	Publisher subscriber message queueing telemetry transport (MQTT) diagram. Adapted from [70]	99

Figure 4.34	Two Samsung A22s using myMQTT connected to test mosquitto broker over our 5G network. Subscribed message retrieval is shown on the left from the published message on the right.	100
-------------	--	-----

List of Tables

Table 2.1	3GPP frequency range definitions. Adapted from [26] and [27]. . .	11
Table 2.2	Starlink satellite constellation information	25
Table 3.1	SWAP-C of 5G network hardware components	29
Table 3.2	Software memory requirements	30
Table 3.3	Open5GS IP addressing	36
Table 3.4	Important services in UST. Adapted from [32].	47
Table 3.5	pySim programs	48
Table 4.1	Modem virtual communication ports	66
Table 4.2	Quectel modem AT commands	68
Table 4.3	Quectel modem commands to add an APN	69
Table 4.4	3GPP modulation order	74
Table 4.5	MCS testing network settings	75
Table 4.6	Comparison of MCS values vs received SNR at UE at 1m	76
Table 4.7	Transmit gain testing network settings	77
Table 4.8	Comparison of gNB transmit gain vs received SNR at UE at 1 m .	77
Table 4.9	Comparison of 59 dB, 69 dB, 79 dB gNB transmit gain vs received SNR at UE at 10 m	80
Table 4.10	Antenna and proximity network testing settings	82
Table 4.11	Indoor testing comparison of distance vs RSRP at UE	83
Table 4.12	Outdoor testing comparison of distance vs RSRP at UE	83
Table 4.13	Testing results of outdoor maximum operating distance	85

Table 4.14	Download speeds using curl and speedtest through SatNet backhaul	88
Table 4.15	Latency results from speedtest through SatNet backhaul	91
Table 4.16	Upload speeds using speedtest through SatNet backhaul	92
Table 4.17	srsRAN available bands. Adapted from [27].	103

List of Acronyms and Abbreviations

5G	Fifth Generation
5GC	5G core
5G-GUTI	5G global unique temporary identifier
5G-S-TMSI	5G serving-TMSI
ADF	application dedicated file
AGC	automatic gain control
AMF	access and mobility function
ARFCN	absolute radio frequency channel number
ATAK	Android team awareness kit/Android tactical assault kit
AUSF	authentication server function
BFT	blue force tracker
BW	bandwidth
COTS	commercial off-the-shelf
CORESET	control resource set
CK	ciphering key
DF	dedicated file
DL	downlink
DN	data network
DOD	Department of Defense

EF	elementary file
FR	frequency range
FPLMN	forbidden PLMN
GEO	geosynchronous
gNB	next generation nodeB
HPLMN	home public land mobile network
ICCID	universal integrated circuit card identifier
IK	integrity key
IMEI	international mobile equipment identifier
IMSI	international mobile subscriber identifier
IoT	internet of things
ISR	intelligence, surveillance, and reconnaissance
LEO	low earth orbit
MCC	mobile country code
MCS	modulation and coding scheme
MF	master file
MIB	master information block
MIMO	multiple-input multiple-output
mMTC	massive machine-type communications
MNC	mobile network code
MQTT	message queueing telemetry transport
MSIN	mobile subscriber identification number

MIB	master information block
NAI	network access identifier
NGO	non-governmental organization
NRF	network repository function
NSA	non-standalone
NTN	non-terrestrial network
NUC	next unit of computing
OP	operator code
OPc	operator code (derived)
PBCH	physical broadcast control channel
PCI	physical cell identity
PDCCH	physical downlink control channel
PDSCH	physical downlink shared channel
PDU	protocol data unit
p-LEO	proliferated-low earth orbit
PLMN	public land mobile network
PSS	primary synchronization signal
PRACH	physical random-access channel
PUCCH	physical uplink control channel
PUSCH	physical uplink shared channel
RAR	random-access response
RAN	radio access network

RF	radio frequency
RRC	radio resource control
RTT	round trip time
SA	standalone
SatNet	satellite network
SCS	subcarrier spacing
SDR	software defined radio
SEAF	security anchor function
SI	system information
SIB	system information block
SIB1	system information block one
SMF	session management function
SNR	signal-to-noise ratio
SQN	sequence number
SS	synchronization signal
SSS	secondary synchronization signal
SUPI	subscription permanent identifier
SUCI	subscription concealed identifier
SWAP-C	size, weight, power, and cost
TC-RNTI	temporary cell radio-network temporary identifier
3GPP	Third Generation Partnership Project
TMSI	temporary mobile subscriber identifier

TP	throughput
UDM	unified data management
UL	uplink
UE	user equipment
UICC	universal integrated circuit card
UPF	user plane function
USIM	Universal Subscriber Identity Module
UST	USIM service table

THIS PAGE INTENTIONALLY LEFT BLANK

Acknowledgments

I would like to thank the Computer Science (CS) department and faculty for their time and effort throughout my education here at NPS. Thank you to my advisor team of CAPT Chad Bollmann, Professor Bret Michael, and Mr. Darren Rogers. CAPT Bollmann, thank you for the Electrical and Computer Engineering (ECE) guidance and perspective throughout the thesis process and for challenging my thinking on 5G technology. Professor Michael, thank you for your support and providing validation to this work. Darren, thank you for the endless conversations and allowing me to use you as a sounding board.

Thank you Jake Snyder for the discussions, collaboration, and friendship throughout the thesis process as we conquered 5G together in our office.

Lastly and most importantly, I want to thank my wife, Lainie. I'm overwhelmed with gratitude for your support, encouragement, and love. You are an incredible mother to our two precious children, Lila and Lincoln, and seeing you love and care for them makes my heart so full. I am so proud of you and your work ethic and career ambitions, reminding me that we should always work at it with all our heart, as working for the Lord. We make a great team and I love you endlessly.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 1:

Introduction

The use of cellular networks for communication has become ubiquitous worldwide, with many use cases extending well beyond traditional phone calls and text messages between users. Furthermore, establishment and integration of Fifth Generation (5G) networks into commercial and military environments continues to expand in developed nations. In order to maximize efficiency of operations, 5G networks provide tethering to the Internet at large for vast amounts of electronic devices while also allowing for mobility and significant data speeds with almost immeasurable latency or signal delay. Leveraging these capabilities will prove vital in continuing to increase efficiency and productivity in many operational aspects, some yet to be discovered.

In this thesis, we seek to test the capabilities and limitations of a constructed open source 5G network by investigating radio frequency (RF) characteristics, testing signal strength in relation to network proximity, gathering information on maximum operating distances, and uncovering network throughput over a satellite network (SatNet) backhaul. Additionally, we look to provide a systematic approach to programming SIM cards and user equipment (UE) for prepared use on an open source network.

1.1 Purpose

According to the Department of Defense (DOD) 5G Strategy, "5G wireless and mobile communications is a critical strategic technology: those nations that master advanced communications technologies and ubiquitous connectivity will have a long-term economic and military advantage" [1]. One of the crucial DOD 5G lines of effort is to promote technology development by facilitating the advancement and adoption of 5G technology and identifying new uses for 5G systems by researching, developing, testing, and evaluating such capabilities [2]. As such, the DOD is investing heavily by experimenting with network technologies in both the radio access network (RAN) and 5G core (5GC), with emphasis on open architectures and virtualization alternatives [2]. In conjunction with expanding the use of 5G technologies is the further integration and utilization of SatNets into our communications

systems, specifically those in closer proximity to Earth in low earth orbit (LEO).

The Third Generation Partnership Project (3GPP) is an international organization that establishes mobile broadband standards and guidelines, but focuses mostly on cellular telecommunications technologies including radio access, core network, and service capabilities by producing releases that outline network features at given points in time to provide developers requirements and goals for progressing the mobile network standards. In early March of 2022, 3GPP issued Release 17, its newest global 5G standard, expanding system capabilities through increased spectrum usage to up through 71 GHz, enhancing massive multiple-input multiple-output (MIMO), and signal beamforming. More specifically related to this thesis, 5G support for satellite communications was introduced. Additionally, 3GPP has issued several technical reports (TR) [3]–[6] addressing feasibility solutions for using SatNets in overall 5G network design. Consequently, using SatNets in 5G networks continues to drive upcoming 3GPP releases for future requirements of mobile networks. Also, the Congressional Research Service recently issued a report [7] on the potential to leverage LEO satellites in order to address the broadband digital divide in rural and remote areas. As such, integration of SatNets into our communication networks should be seen as a critical complementary technology to 5G deployment.

This thesis researches emerging 5G technologies and their suitability for integrating proliferated-low earth orbit (p-LEO) constellations into mobile network architecture. We design, develop, and test a portable open source 5G network with commercial off-the-shelf (COTS) hardware that connects UEs to a network backhaul through p-LEO SatNets. Portions of this thesis were adapted from [8]. In particular, the setup and design of our network throughput testing were modeled after this paper.

1.2 Related Work

The related work for this thesis can be broken down into two separate categories: work related to open source software for 5G networks and work related to integrating SatNets into 5G networks.

A lot of research has been conducted on the idea of using open source software as an alternative to the large network providers. There have been other cellular testbeds created

and tested like in [9], where the authors investigated the computational efficiency and breakdown of CPU utilization while using srsLTE and srsUE with one virtual device connected to an instantiated LTE network. Computational memory requirements of virtual RANs were tested in [10] using srsRAN, showing that the CPU and memory utilization of the next generation nodeB (gNB) increases significantly for each additional connected user. Other researchers sought to compare different open source 5GC stacks [11], [12] by investigating and dissecting the available documentation provided for each project. An in-depth study of Open5GS was conducted in [13], where they document the 5GC setup while using virtual UE and RAN features to integrate with the Open5GS network. Using srsRAN to conduct network slicing is discussed in [14], but is used unilaterally without connection to any core components. Additionally, several NPS students have investigated utilizing open-source materials to construct and test 4/5G networks via a different open source software package called OpenAirInterface [15], [16]. In [17], the authors also used OpenAirInterface, but studied the vulnerabilities related to location services. Other research investigations have used 5G network-in-a-box solutions like in [18] to study the potential use of 5G networks for streaming intelligence, surveillance, and reconnaissance (ISR) video in unmanned aerial vehicle (UAV) missions.

As discussed in the preceding section, integration of SatNets is an important step towards realizing the full potential of 5G networks. Consequently, a multitude of studies discuss the integration of SatNets into 5G networks. [19] considers pushing edge computation capabilities directly to the satellite while [20] and [21] discuss the 3GPP standards for leveraging SatNets as a entry point into the 5G network space.

In both categories of related work, however, none provide proof of concept in constructing usable or operational 5G networks with srsRAN and Open5GS open source software and SatNets for backhaul communication. Further, the related research remains mostly theoretical with limited demonstration of functional network components with open source software. This thesis research is different in that it seeks to coalesce disparate open source software stacks of srsRAN and Open5GS with relatively cheap COTS hardware, while providing backhaul communication with commercially-available satellite Internet.

1.3 Research Questions

An objective of this research is to provide a proof of concept for establishing standalone 5G communications without the use of any organic commercial network infrastructure. The scope of this thesis is limited to implementing a hardware/software construction of a 5G mobile network to subsequently test successful end-to-end connection of user equipment through a SatNet backhaul. The idea is that a user or group of users could reasonably assemble an inexpensive and small-scale 5G network for use as a communications channel in environments that do not have any usable network connection. Consequently, our intended research questions are as follows:

1. Does the constructed network configuration establish end-to-end connections via satellite networks?
2. What are the bandwidth and latency limitations of utilizing satellite systems for backhaul 5G connectivity?
3. What are the hardware limitations for constructing the 5G mobile network?
4. What is the overall size, weight, power, and cost (SWaP-C) of the RAN, Core, and satellite terminal hardware?
5. What is the observed end-to-end throughput for the network?
6. What is the necessary proximity of end users to the RAN while utilizing different antenna hardware?
7. What is the overall data size of the entire software buildout? How many bytes and lines of code each for the RAN and Core components?

1.4 Thesis Organization

There are five chapters and two appendices in this thesis. Chapter 2 provides an overview of both 5G and satellite networks and discusses how the two can be integrated. The 5G cellular network testbed is outlined in Chapter 3 with subsequent results detailed in Chapter 4. The contributions, key findings, and considerations for future work are covered in Chapter 5. The appendix contains code snippets used in this thesis. Code was written in Python and Bash.

CHAPTER 2:

5G Network Architecture and Functionality

This chapter outlines the different facets and overall architectural design of 5G networks, including an introductory study of the components and their intercommunication. 5G networks are abundantly complex with extensive RF signalling conducted by the gNB cells and UEs, as well as a constant exchange of messages between UEs, RAN, and 5GC. The signalling and message exchange between network components is essential to a functioning and operational 5G network, as explained in this chapter. Material covered in this chapter directly relates to our network construction, methodology, and results covered in Chapters 3 and 4.

A 5G network can be separated into four unique elements: RAN, 5GC, UE, and network backhaul. We achieved connection to the mobile backhaul through various means, but ultimately attained access with the use of SatNets. The initial sections in this chapter detail each of the 5G network elements to provide a succinct but robust understanding of the associated functions and responsibilities.

2.1 Radio Access Network

A gNB is the physical implementation of the RAN, responsible for all RF-related operations of the 5G network including but not limited to UE connection establishment with admission control, radio resource management, and routing of control and user plane information to the access and mobility function (AMF) and user plane function (UPF) of the 5GC, respectively. The RAN connects to the 5GC through the NG-RAN (Next-Generation RAN) interface and uses the air interface for UE communication.

2.1.1 UE Access to a Network

There are two different aspects to gaining network access for UE. They must be able to both find the RAN in a given coverage area as well as gain initial access to that network by requesting connection setup in a procedure usually referred to as random access.

Cell Search

A cell is defined as a cellular network access point within a 5G network, described by periodic control signals known as blocks. The first step for a UE to gain network access is to look for synchronization signal (SS) blocks of active cells within its geographic area. These SS blocks, cooperatively with the physical broadcast control channel (PBCH), are periodically transmitted on the downlink (DL) from each cell [22]. The SS block assists with the initial synchronization, while the PBCH carries pertinent system information.

Within the SS block, the primary synchronization signal (PSS) is the first signal transmitted and thus the first signal searchable by the device. The device can then use the PSS to eliminate any frequency deviation between the cell and the device. As a result, the device can now synchronize to the secondary synchronization signal (SSS) and determine the physical cell identity (PCI) of the cell. There are 1008 PCIs in total, constructed from combinations of the ranges of the PSS (0-2) and SSS (0-335); that is, $3 \times 336 = 1008$.

The PBCH carries the master information block (MIB), which contains the minimum amount of information that the device needs for acquiring and connecting to the cell, namely system information block one (SIB1) [22]. The MIB transmits on the PBCH with a period of 80 ms and comprises the parameters needed to acquire and decode SIB1 from the cell [23]. System information (SI) is the common term for all the information that a device needs to connect and operate on a cell within a network. The system information is carried within various system information blocks (SIBs), with each containing different types of system information [22]. SIB1 consists of the imperative system information necessary to obtain entry to a particular cell and broadcasts every 160 ms to the entire cell area through a regularly scheduled physical downlink shared channel (PDSCH) transmission [22].

All of the remaining SIBs contain additional information, but nothing necessary to gain the initial access to a cell. Therefore, these SIBs can be transmitted on demand and by request of the UE, as seen in Figure 2.1, which shows a depiction of the signalling flow between a UE and a cell.

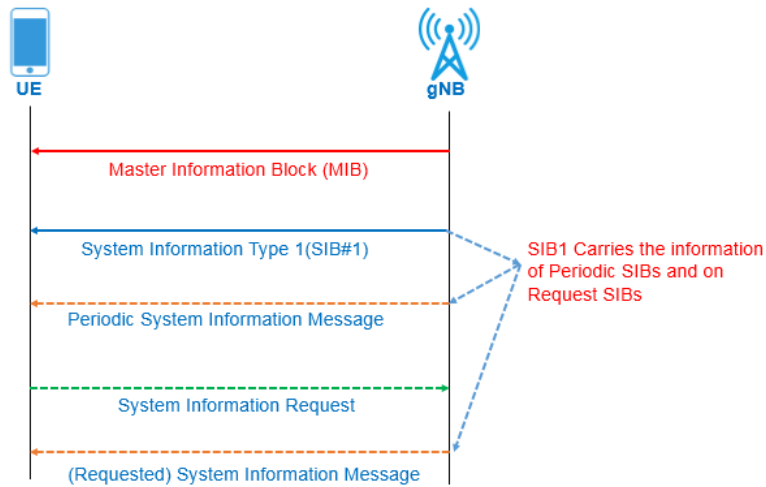


Figure 2.1. MIB and SIB signalling flow. Source [24].

As a reference, the list below (adapted from [22]) outlines some of the different physical channels associated with 5G:

PDSCH —Physical Downlink Shared Channel— main physical channel for data transmission from cell to UE as well as paging information, random-access response messages, and delivery of SI

PBCH —Physical Broadcast Control Channel— carries part of SI like the MIB, required by device to access cell and network

PDCCH —Physical Downlink Control Channel— used for downlink control information like scheduling decisions and required for retrieval of PDSCH and enabling transmission of PUSCH

PUSCH —Physical Uplink Shared Channel— uplink counterpart to PDSCH

PUCCH —Physical Uplink Control Channel— used by device to send acknowledgements that indicate successful receipt of downlink transport blocks to the gNB and to request resources to transmit uplink data through

PRACH —Physical Random-Access Channel— used for random access

Random Access

After finding a cell, the device can now proceed with gaining access through a random-access procedure, which is comprised of a four-step process. These steps are shown in Figure 2.2 with a detailed depiction of the messages between the UE and cell in Figure 2.3. The four steps are summarized below from [22]:

- ① UE transmits a preamble, called a physical random-access channel (PRACH)
- ② Network transmits a random-access response (RAR), which confirms receipt of the PRACH from the UE. Network also assigns a temporary identity to the UE, called a temporary cell radio-network temporary identifier (TC-RNTI). This communication occurs on the physical downlink control channel (PDCCH).
- ③ ④ "Device and network exchange of messages (uplink (UL) - message 3; DL - message 4) with the aim of resolving potential collisions due to simultaneous transmissions of the same preamble from multiple devices within the cell. If successful, message 4 also transitions the device to a connected state" [22].

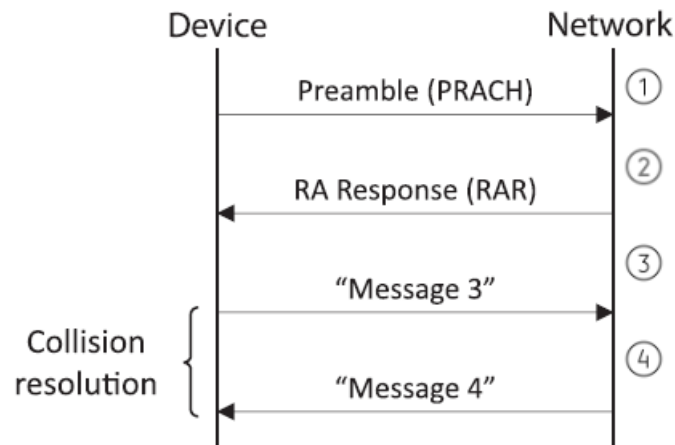


Figure 2.2. Simplified random access procedure. Adapted from [22].

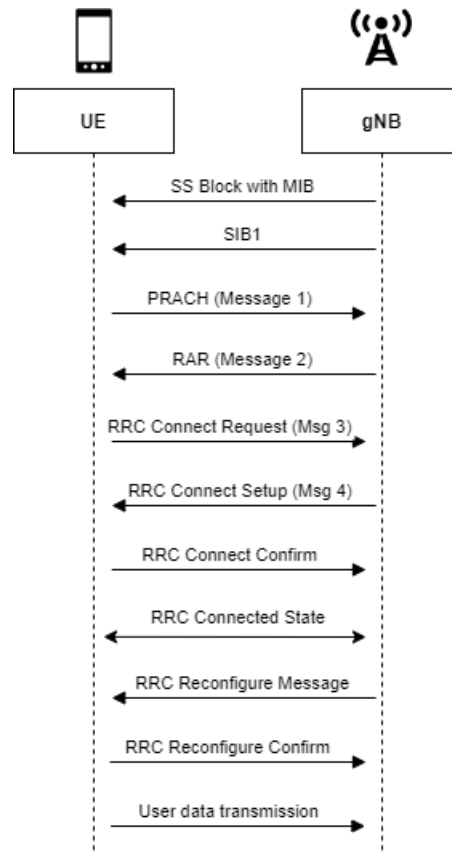


Figure 2.3. Detailed random access procedure. Adapted from [25].

After decoding the SS block and SIB1, the UE transmits a preamble to the cell for starting the initial access process. Once the preamble is received by the cell, the RAR is sent back, which contains a temporary identifier that is used for further communication between the device and the network. Upon receipt, the device will now adjust UL transmission timing and move on to the next step in the random access process. Messages 3 and 4 are used for collision resolution and ensure unique identifiers are used for each unique device. Message 3 sends a radio resource control (RRC) connection request, and message 4 will be the RRC connection setup. Additionally, another message will be sent by the UE to the gNB confirming RRC connection.

The RRC manages control plane message exchanges between the gNB and UE, and is a protocol utilized by the cell to establish link states for UEs. It functions as a state machine

with three possible states: RRC_CONNECTED, RR_INACTIVE, RRC_IDLE. Idle mode indicates that there is no RRC connection established, and inactive is a sort of dormant state with no active user plane communication between UE and gNB, but where RRC connection can be quickly resumed. The inactive feature expands on Fourth Generation (4G) functionality to provide low latency capabilities to allow for massive machine-type communications (mMTC) connections where devices are not always active, but rather report metrics sporadically. Figure 2.4 shows the different states and transitions.

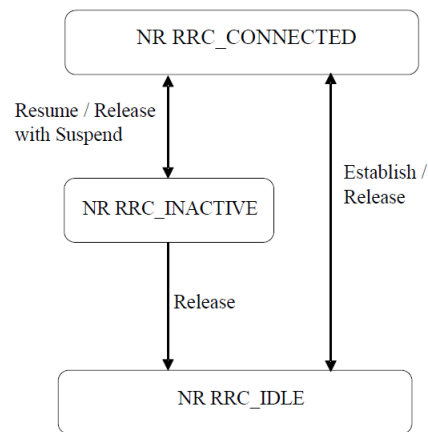


Figure 2.4. RRC state machine and transitions. Source [23].

When troubleshooting a potential faulty UE connection to a network, one approach is to investigate the random access procedure and associated four-step process in order to confine the problem. For example, when setting up our network, we uncovered that the UE had discovered the cell and that the network had sent a confirmation of the PRACH, however there was no exchange of messages 3 and 4. We were then able to debug and adjust the cell parameters accordingly to establish a successful connection. Further details about these efforts are discussed in Chapter 4.

2.1.2 Radio Frequency

5G is divided into two broad frequency bands, frequency ranges (FRs): FR1 and FR2, shown in Table 2.1. FR1 consists of both low and mid band, with frequencies reaching up to around 7.125 GHz. FR2 is generally considered as frequencies starting at around 20 GHz, reaching

through mm wave above 30 GHz and heading to 71 GHz, as released in the latest 3GPP Technical Standard (TS) for FR2 [26]. Of note, all bands also equate to an absolute radio frequency channel number (ARFCN), which is a unique identifier associated with each of the different carrier frequencies in a cellular network. We will use these ARFCNs when configuring the gNB settings of our network. Additionally, every ARFCN must match with a corresponding and appropriate control resource set (CORESET) value.

Table 2.1. 3GPP frequency range definitions. Adapted from [26] and [27].

Frequency Range Designation	Frequency Range
FR1	410 MHz - 7.125 GHz
FR2	24.25 GHz - 71 GHz

Low band transmits around 600-700 MHz to provide broad, longer-range coverage to large geographic areas [28]. It provides the most coverage, but lacks in speed with rates only reaching about 50 Mbps. This band is heavily utilized in rural areas and places with few users. Mid band typically operates in the 1.7 to 2.5 GHz range and offers a better balance between speed and coverage area [28]. Speeds of almost 1 Gbps could be reached, but usually hover somewhere in the middle of 100-900 Mbps. High band operates at above 24 GHz, rendering speeds of 10 Gbps possible, but more often, around 1 Gbps. However, coverage areas are significantly smaller, in the range of only a couple hundred meters, or even less for highly dense areas with any obstructions.

2.2 Core Network

The 5GC is responsible for all operations outside of the entry point functionality provided by the RF frontend of the RAN. Such functions include, but are not limited to, device management, authentication, network slicing, and setup of end-to-end connections. The core is split into two main parts: control plane and user plane. The user plane contains the user's data, or actual content of the communication, whereas the control plane consists of all the signalling traffic necessary to establish a connection between the UE and network.

The core itself can be thought of as a service-based architecture (SBA), meaning that the functionality and services provided are abstracted from individual pieces of hardware. The SBA is commodity-based and provides a modular framework where similar functions are outfitted from varying suppliers and vendors. 3GPP defines SBA as a set of interconnected network functions (NF), each with authorization to access each other's services [29]. In many cases, these core services are provided by virtual machines running software that can accomplish the different functions.

A simplified depiction of the 5GC as an SBA is rendered in Figure 2.5. Further details on certain functionality will be discussed in the user equipment section of this chapter where identity exchange and authentication are outlined. Also, networking links between components are discussed in Chapter 3 with specific reference to Open5GS and the IP addressing scheme used in our network.

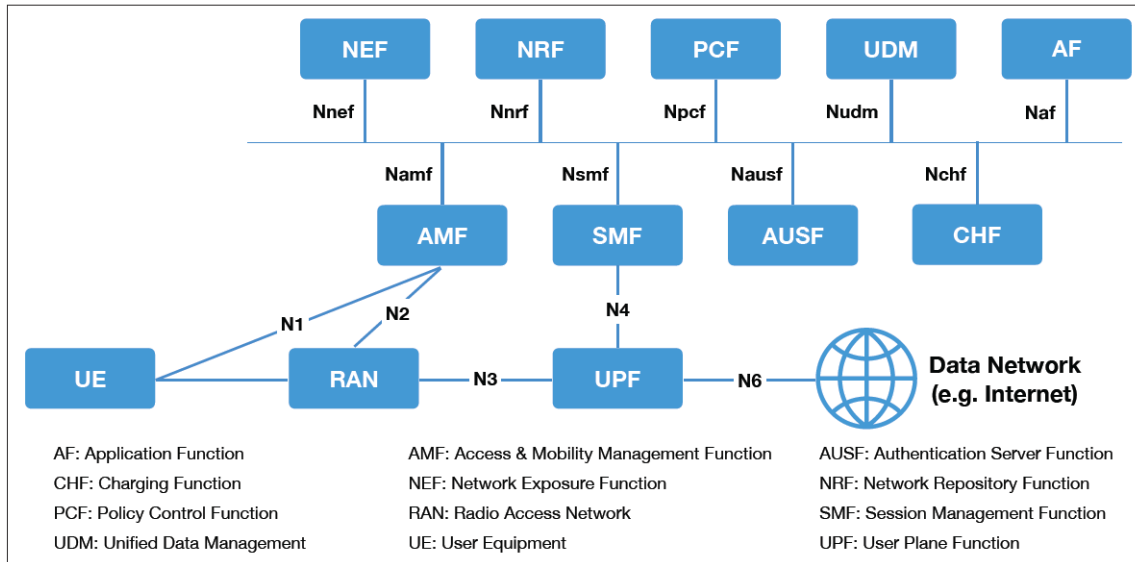


Figure 2.5. Simplified 5G SBA

The below list extracted from [22] contains a concise summary for each of the significant 5GC components related to this thesis:

AMF —Access and Mobility Management Function— signalling between UE and 5GC, mostly used for authentication and mobility management during network connections

AUSF —Authentication Server Function— authentication server for network access within the user’s home network

NRF —Network Repository Function— provides a way to discover other services by having control plane functions configured to register with the NRF

SEAF —Security Anchor Function— part of the serving network that acts as a liaison between the UE and home network during the authentication process

SMF —Session Management Function— session establishment/modification/release and IP address allocation for UEs

UDM —Unified Data Management— authentication credentials, storage and management of active subscriber registration, and access authorization based on subscription plans

UPF —User Plane Function— includes packet inspection/routing/forwarding/fil-

tering and QoS handling, but most importantly acts as a gateway between the 5G architecture and external network or Internet

2.3 User Equipment

User equipment can be generalized as any sort of communication node that connects to the 5G network. Some examples could include kitchen appliances, temperature sensors, security monitoring systems, health devices, vehicles, or cell phones. In reality, anything with "smart" prepended to the name can be associated as a UE for 5G networks.

5G brings communication functionality to a whole new level for UEs, and more broadly end users seeking to integrate all features of daily life. For end users, 5G vastly improves on 4G mobile telecommunications networks with substantial upgrades to network bandwidth (BW), data size, and connectivity with reduced latency. With these improvements come a vast array of new emerging uses of mobile networks such as remote medical surgeries, autonomous vehicle networks, smart logistics, augmented reality, and smart cities. The key upgrades of 5G technologies are described as three broad categories in [2] and summarized below. They are also depicted visually in Figure 2.6.

Enhanced Mobile Broadband (eMBB) - Increased network bandwidth (BW) speeds up to 10 Gbps for users of high mobility

Massive Machine-Type Communication (mMTC) - Increased data size networks with countless connected devices (1 Million/km²), which also includes anything within the internet of things (IoT).

Ultra Reliability and Low Latency Communication (URLLC) - Increased connectivity with reduced latency "supporting delivery of critical communications assured by ultra-reliable [(99.999% availability)] and low latency networks [(1ms latency)]" [2]

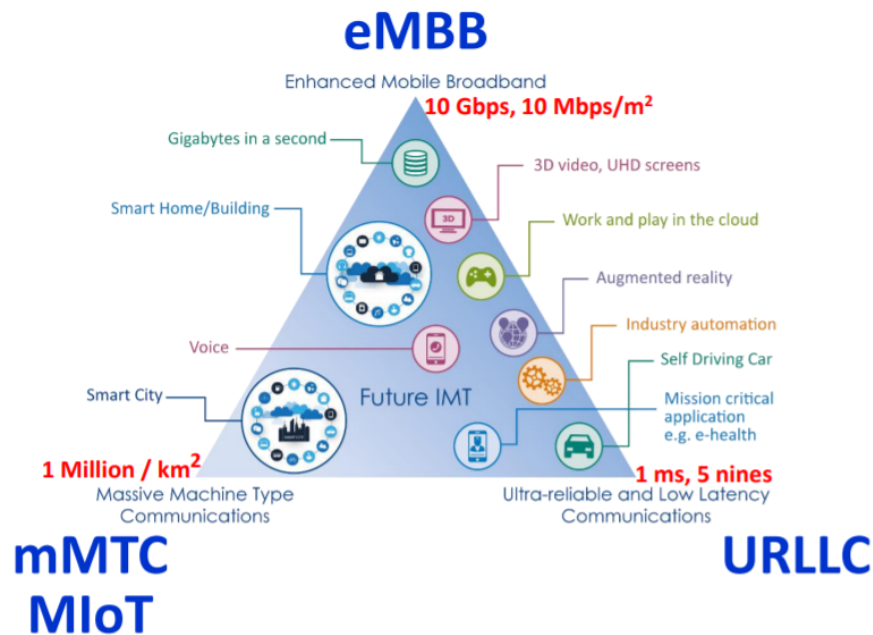


Figure 2.6. Major 5G categorical use cases. Source [30].

2.3.1 UE Identifiers

This subsection outlines the different identifiers associated with UE and further details the dual authentication performed by both the UE and network as described in the RAN and cell search section.

Universal Subscriber Identity Module (USIM)

The USIM is an application that resides on the universal integrated circuit card (UICC), which is the physical smart card that is inserted into a UE. The USIM contains a processor with electronically erasable programmable read only memory (EEPROM), RAM for program execution, ROM for the operating system, and cryptographic algorithms [31]. There also exists a hierarchical file system, which resides on the EEPROM and consists of three file types: master files (MFs), dedicated files (DFs), and elementary files (EFs). The MF is the root of the file system, containing DFs and EFs. EFs contain various types of formatted and

programmable data in a grouping subordinate to DFs [32]. There are hundreds of different EFs within the file structure, and particular ones will be discussed in Chapter 3 where we outline USIM changes necessary for proper configuration of our testbed.

The list below compiled from [33] contains a summary of important UE identifiers, all of which are programmable through manipulating their associated EF.

Universal integrated circuit card identifier (ICCID) - value assigned to the UICC hardware

International mobile subscriber identifier (IMSI) - comprised of the public land mobile network (PLMN), which is the combination of the mobile country code (MCC) and mobile network code (MNC), as well as the mobile subscriber identification number (MSIN):

MCC: identifies the Country origin of the subscriber

MNC: identifies the home network of the subscriber

MSIN: identifies the unique subscriber

Subscriber Key (K): secret key stored on the USIM and mirrored in the UDM

Operator code (derived) (OPc): operator code derived from a combination of K and operator code (OP) (common amongst USIMs of the same single operator)

Sequence number (SQN): arbitrary value mirrored in the UDM that is used to prevent replay attacks by incrementing after each authentication challenge

Forbidden PLMN (FPLMN): list of PLMNs not allowed for use by the subscriber

Tracking Area

In 5G cellular networks, a tracking area code (TAC) is a unique identifier assigned to a group of gNBs within a geographic area. The TAC in combination with the PLMN comprise the tracking area identifier (TAI), which assists UEs in determining connections to different gNBs and ensure transparent handoffs between cells.

Subscription permanent identifier (SUPI)

Every user, commonly referred to as a subscriber within a 3GPP mobile network, is issued a SUPI, which differs from the international mobile equipment identifier (IMEI) identifier directly associated with the hardware of the UE. In typical public 5G networks, the SUPI

simply equates to a user's assigned IMSI. Otherwise, an network access identifier (NAI) in the form of a string of characters, usually involving an email address, can be used for private networks to identify separate users.

Subscription concealed identifier (SUCI)

The SUCI is an identifier used to conceal the SUPI and provide privacy for users on the network. A breakdown of the specific components of the SUCI is shown in Figure 2.7, and summarized below from [34].

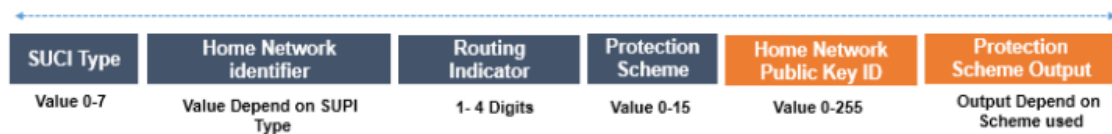


Figure 2.7. SUCI breakdown. Source [35].

SUCI Type: identifies the type of SUPI concealed in the SUCI:

- 0: IMSI
- 1: Network Access Identifier (NAI)
- 2-7: Values for future use

Home Network Identifier: identifies the home network of the subscriber, usually in the form of a PLMN. When an NAI is used for the SUPI, however, this identifier takes the form of a string of characters representing a domain name like user@mobile.com.

Routing Indicator: identifies a value between 1 to 4 digits for use by the home network operator for signal routing, inherently provisioned on the USIM.

Protection Scheme Identifier: represents a null or non-null scheme.

Home Network Public Identifier: identifies a public key provisioned by the home public land mobile network (HPLMN) and used to identify the key type for SUPI protection.

Protection Scheme Output: consists of a string of characters and represents the output of the public key protection scheme.

5G global unique temporary identifier (5G-GUTI)

The AMF provides an 80-bit 5G-GUTI to the UE as a temporary identifier while using either a 3GPP or non-3GPP network [29], as shown in Figure 2.8. It consists of the PLMN, AMF ID, and the 5G temporary mobile subscriber identifier (TMSI). The AMF may also re-issue the 5G-GUTI for several reasons upon receiving any of the following [36]:

1. Initial registration request or mobility registration update
2. Periodic registration update
3. UE response to a paging message

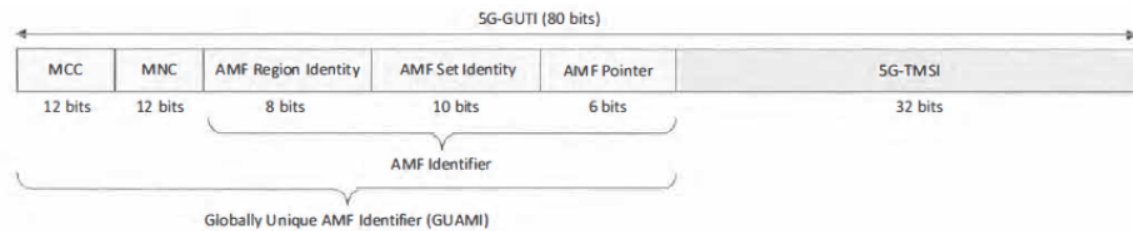


Figure 2.8. 5G-GUTI. Source [37].

Additionally, the 5G serving-TMSI (5G-S-TMSI) is a shortened form of the 5G-GUTI, providing for efficient use of radio signalling within the network during paging and service requests and takes the form of:

$$\langle 5G-S-TMSI \rangle := \langle AMF Set ID \rangle \langle AMF Pointer \rangle \langle 5G-TMSI \rangle [29]$$

2.3.2 UE and Network Identity Exchange

In order to register the UE on a live network an exchange of information between the UE and network must occur, identifying each side of the connection. Figure 2.9 shows the interchange of such messages starting with an initial registration request by the UE where the SUPI is encrypted to become the SUCI and then sent across the network via the gNB to the AMF.

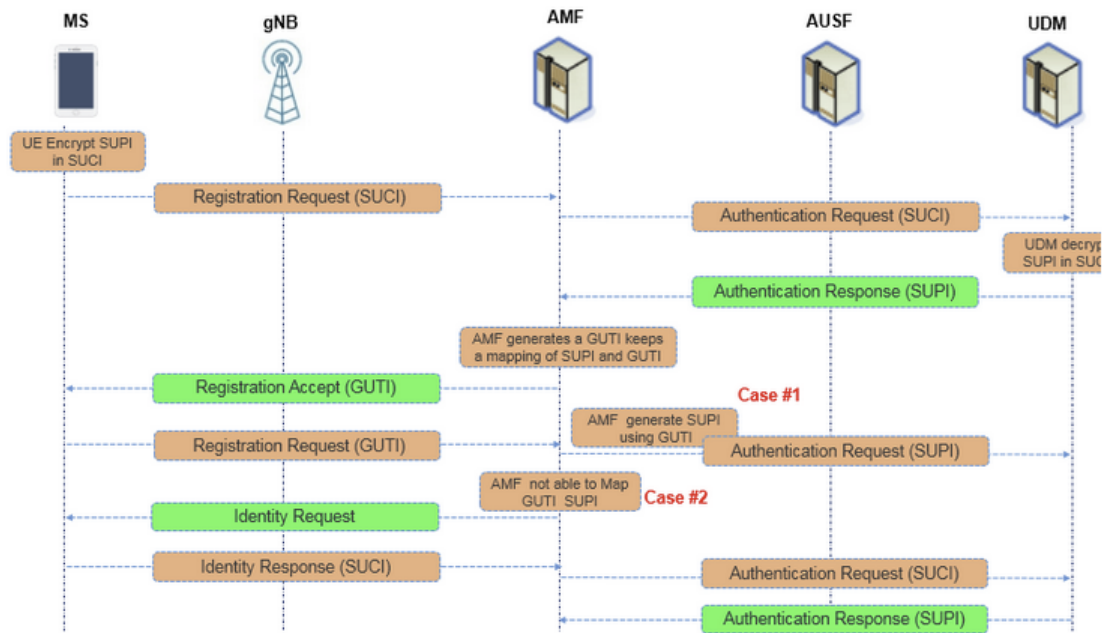


Figure 2.9. UE and network identity exchange. Source [38].

Following the UE's initial registration request, 5G-GUTI is assigned to the UE by the AMF after decrypting the SUCI to the SUPI via the AUSF and then verifying that it is a registered user within the UDM. In response, the UE sends back a registration request with the same 5G-GUTI to verify its use and to fully authenticate on the network. Once the authenticated connection is accepted and fully established, the UE can request a protocol data unit (PDU) session from the 5GC. The session management function (SMF) control plane function is responsible for establishing and maintaining all PDU sessions, and will assign the device an IP address so that it can access the UPF, external data network, and Internet.

Authentication

Mutual authentication between the UE and network is established through the processes of primary authentication and key agreement, which create keying material, K_{SEAF} , provided by the AUSF of the home network to the SEAF of the serving network [36]. Figure 2.10 shows a full key hierarchy derivation with distinctions between the only two 3GPP-accepted authentication methods for 5G: EAP-AKA and 5G AKA. 5G AKA advances the techniques of EPS-AKA by "providing the home network with proof of successful

authentication of the UE from the visited network" [36]. The keys are used for different functions within the network such as user plane traffic, control plane messages, RRC signalling, and communication between gNB and UE.

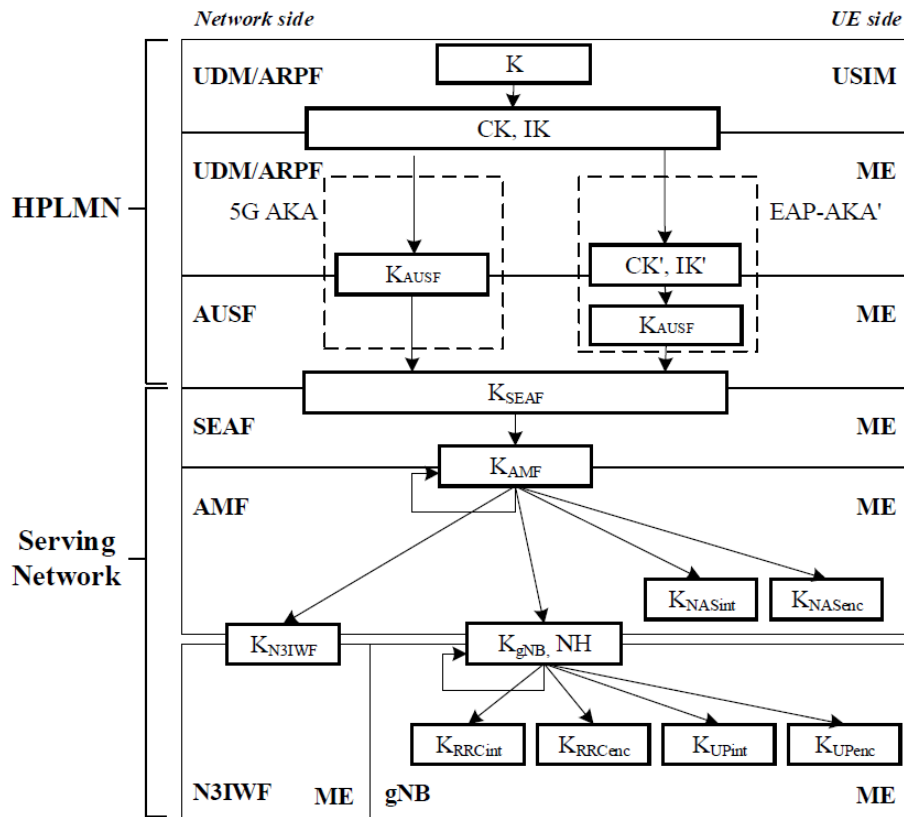


Figure 2.10. 5G key hierarchy generation. Source [36].

Stemming from the secret key (**K**), the integrity key (**IK**) and ciphering key (**CK**) are created through an algorithm that combines the **K** value along with a random number (**RAND**) generated by the AUSF [36]. Additionally, the UE and network exchange a series of messages to authenticate each other using the **RAND** and the stored **K** value. From there, the **CK** can be used for encryption and decryption of both control and user plane messages between the UE and 5GC, while the **IK** is used to maintain message integrity.

2.3.3 User Plane Traffic

Once the UE achieves registration on the network through authentication, the SMF coordinates a PDU session so that the user can gain access to the data network (DN) through the UPF from the gNB. A subsequent quality of service (QoS) flow for a specific activity is set up between the UE, gNB, UPF, and DN through the use of a General Packet Radio System (GPRS) Tunnelling Protocol User Plane (GTP-U). This creates a tunnel for data flow from the UE to the UPF through the gNB, as seen in Figure 2.11. During this process, the UE is also assigned an IP address from the SMF and can now access the data network and send or receive any IP-related traffic. Communication between the UPF and SMF is handled through the packet forwarding control protocol (PFCP).

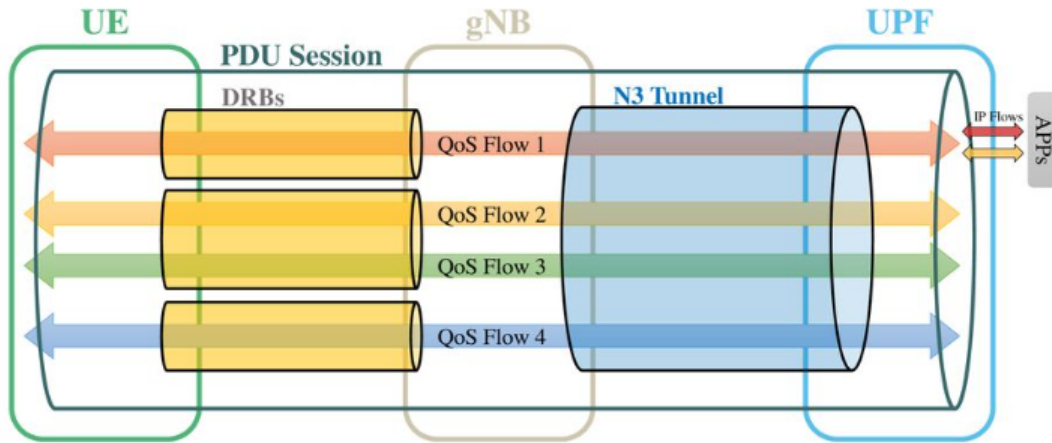


Figure 2.11. PDU session. Source [39].

2.4 Backhaul

Backhaul in a 5G network refers to the signal path from the 5GC to any other network, thereby providing external communication. This outside connection often generalizes to the Internet or cloud. Traditionally, fiber wireline extends directly from the core component hardware through various switches and routers until eventually reaching the Internet backbone. However, for the purposes of this thesis research and in our network design, we used a commercial satellite provider to establish our backhaul connection to the Internet. Details about our specific network setup are discussed in Chapter 3.

2.4.1 Satellite Networks

Satellite networks can be used to significantly expand coverage to end users, especially in remote or restricted areas where wireless networks remain uncovered. Leveraging satellites also provides for user mobility, redundancy, and scalability. Some examples include connecting hikers in national parks, first responders communicating in austere emergency situations, and business owners in rural communities maintaining supervision over remote work sites. As a network connection category or type, non-terrestrial networks (NTNs) consist of any non-terrestrial flying object which broadly includes satellites, unmanned aerial systems (UAS), balloons, blimps, manned aircraft, and more. More commonly though, SatNets are the key feature of NTNs.

Figure 2.12 describes the altitude ranges of different NTN orbit patterns. LEO orbits have a period of approximately 90 minutes and can cover hundreds of miles within their sensor field of view. In comparison, geosynchronous (GEO) orbits allow for constant line of sight observations within a very large footprint covering around 42% of the Earth's surface [40]. LEO satellites have the disadvantage of only being able to observe a certain ground user for a short period of time and require a constellation of satellites that are spaced evenly in several orbital planes in order to provide consistent coverage. However, a significant advantage of LEO orbits entails the proximity to ground users on Earth, drastically reducing round trip times (RTTs) and latency.

Platforms	Altitude range	Orbit	Typical beam footprint size
Low-Earth Orbit (LEO) satellite	300 – 1500 km	Circular around the earth	100 – 1000 km
Medium-Earth Orbit (MEO) satellite	7000 – 25000 km		100 – 1000 km
Geostationary Earth Orbit (GEO) satellite	35 786 km	notional station keeping position fixed in terms of elevation/azimuth with respect to a given earth point	200 – 3500 km
UAS platform (including HAPS)	8 – 50 km (20 km for HAPS)		5 - 200 km
High Elliptical Orbit (HEO) satellite	400 – 50000 km	Elliptical around the earth	200 – 3500 km

Figure 2.12. Types of NTN platforms. Source [41].

In the context of 5G networks, expectations of NTNs showcase the rationale behind why SatNets are expected to be seamlessly integrated into the existing 5G architecture. These expectations are summarized in the list below and adapted from [42]:

1. Assist the deployment of 5G service in remote locations that are not easily covered by terrestrial 5G networks.
2. Enhance service coverage capabilities through reduced vulnerability space vehicles that are more immune to physical attacks and natural disasters.
3. Provide a cost effective manner to upgrade network technologies in underserved areas.
4. Reinforce 5G service through redundancy of network access vectors.
5. Enable 5G network scalability by establishing multiple paths to the end user.

According to 3GPP Releases, SatNet configurations vary depending on the level of computation provided by the satellite. TR 38.811 [42] and TR 38.821 [41] describe these setups in two broad categories:

1. **Bent Pipe** - payload performs "radio frequency filtering, frequency conversion, and amplification" [42]
2. **Regenerative** - payload performs the same functions as a bent pipe, as well as, "demodulation/decoding, switch and/or routing, and coding/modulation, effectively equivalent to having base station functions (gNB)" [42]

In both cases though, the satellite connects the 5GC and end user through a radio frequency link called a service link [42]. Separately, a feeder link is a pathway that establishes connection to a ground station gateway and on to the data network or Internet [42]. Figure 2.13 shows a simplification of the network signal path for both terrestrial and traditional non-terrestrial 5G networks. More simply, a service link is where a UE would communicate directly with the SatNet for initial connection into the network and a feeder link is where the SatNet provides access to the data network and Internet. In contrast to the current 3GPP specifications, our network leveraged the satellite to serve as a feeder link instead of a service link, meaning that our SatNet usage provided only the backhaul connection instead of the entry point vector. Further details on our exact network design and approach are discussed in Chapter 3.

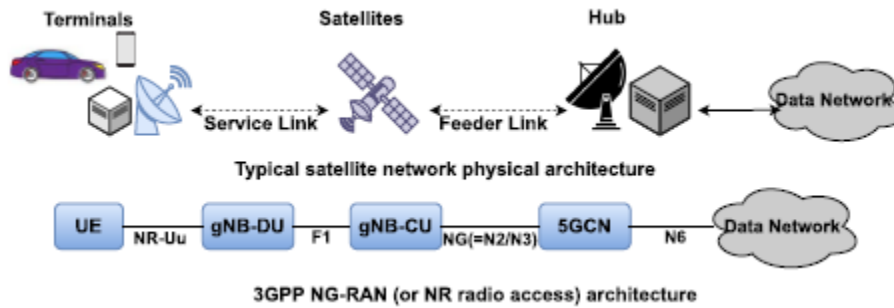


Figure 2.13. Typical 5G NTN satellite integration compared to terrestrial networks. Source [43].

Historically, most satellite Internet services come from only a handful of GEO satellites that orbit the Earth at around 35,000 km, chosen due to synchronization with the Earth’s rotation to enable continuous signal coverage. Consequently, the associated higher latency makes it nearly impossible to support any kind of streaming, gaming, video calls, or other high data rate activities [44]. LEO satellites allow for lower power requirements and shorter RTTs with latency as low as 20 ms, compared to GEO spacecraft with over 600 ms delays [44]. More recently, p-LEO spacecraft, which consist of thousands of small satellites in a LEO orbit that form one large constellation as a whole, have been developed and deployed to provide Internet services. Such endeavors have already become widely commercially available with companies and products such as OneWeb, SpaceX’s Starlink, Amazon’s Kuiper, and Telesat’s Lightspeed.

In this thesis we use SpaceX’s Starlink to achieve our Internet backhaul connection due to its lower cost, larger global coverage, and numerous integration with several other communications architectures within the DOD. Furthermore, in December of 2022, SpaceX announced a partner project called Starshield with the US government, identifying national security agencies as the primary customer [45]. Starshield will have three focus areas of earth observation, communications, and hosted payloads, all of which have additional encryption capabilities for data security [45].

As of September 2022, there were over 3,000 functioning satellites within the Starlink constellation with a planned deployment of around 12,000 and possible expansion to 42,000 [46]. According to the owner of Starlink, there might be close to 500,000 active users of

the satellite constellation in 32 different countries as of June 2022 [47]. Table 2.2 shows an approximation of several metrics associated with the Starlink constellation. Download speeds upward of 150 Mbps are advertised as possible with current deployments of the Starlink system.

Table 2.2. Starlink satellite constellation information

Altitude	Period	Latency	Speed	Orbital Planes	Spacecraft	Freq Bands
550 km	90 mins	20 ms	150 Mbps	72 planes	3000 satellites	Ku, Ka, E

2.5 Summary

5G networks are complex. This chapter gave a high-level overview of specified 5G components and their functions and responsibilities. This context supports our explanation of how we use open source software for both the RAN and 5GC, integration of UEs as network users, and SatNets for backhaul connectivity in our network design.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 3:

5G Cellular Network Testbed

This chapter describes the construction and assembly of our cellular network testbed and successful interconnection of the RAN, 5GC, UE, and backhaul components, creating a fully functional 5G standalone (SA) end-to-end network. A broad overview of the testbed is followed by a detailed depiction of each of the singular components.

3.1 Setup and Implementation

One of the main goals of this thesis involved building a working 5G network using open-source software and COTS hardware. We accomplished this by establishing each of the four 5G network components independently, and then merging their functionalities together. Our 5G network consisted of the following modules:

RAN – srsRAN software run on an Intel next unit of computing (NUC) connected to and controlling an Ettis Research B200 software defined radio (SDR), which served as the gNB

5G Core - Open5GS core run on Intel NUC

5G UEs - Quectel RM500Q-AE Modem and Samsung A22

Backhaul - Starlink

Figure 3.1 shows our overall network architecture design, indicating a bi-directional signal path between UE and the Internet. The Ettis Research universal software radio peripheral (USRP) B200 SDR, chosen for its compact size, signal quality, and relative low cost, provided the RAN RF frontend. It was programmed through srsRAN open-source software running on an Intel NUC small form factor desktop computer. Open5GS software supplied the network functionality for the 5GC, running as a separate software-based service on the Intel NUC. The backbone Internet connection was achieved with the use of commercial satellite Internet provided by Starlink. Two different UE devices and types were used to test the network including a raw 5G modem and one commercial cellular phone. The Sysmocom USIM facilitated access to our network for each of the UEs, supplying necessary subscriber information and authentication credentials.

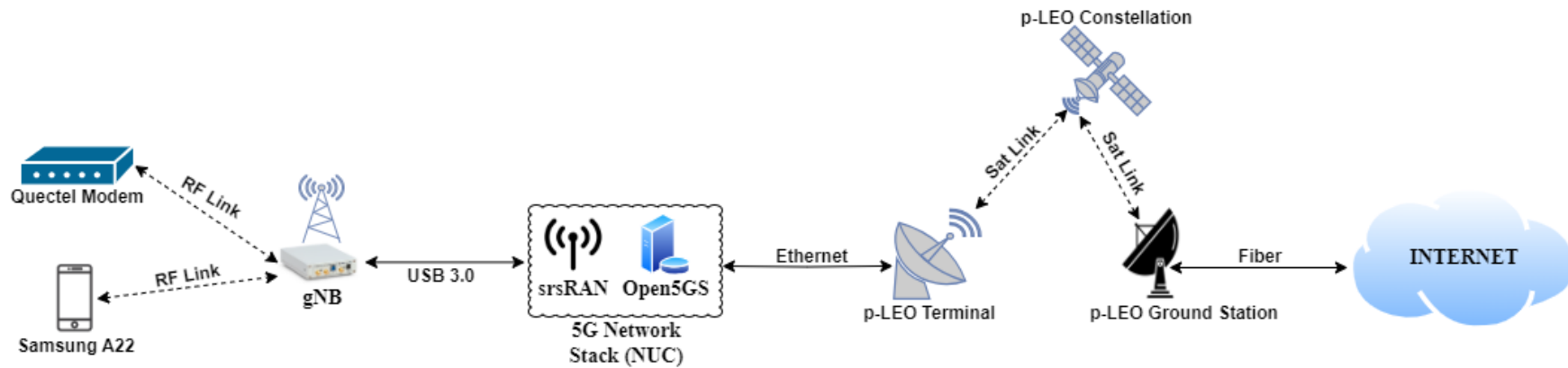


Figure 3.1. 5G network architecture

Specifications for main characteristics are included in the list below, with size, weight, power, and cost (SWAP-C) of the network hardware noted in Table 3.1.

Intel NUC – 64 GB of RAM, Intel Core i7 CPU 10th Generation, 6 Cores, 2.0 TB SSD, Ubuntu OS [48]

USRP B200 - RF coverage from 70 MHz to 6 GHz, GNU radio, C++, and Python APIs, USB 3.0, 56 MHz BW, USB bus powered, 1 TX and 1 RX at half or full duplex [49]

Starlink - 20 ms latency, up to 500 Mbps with premium service contract, IEEE 802.11a/b/g/n/ac standards, WPA2 [44]

Table 3.1. SWAP-C of 5G network hardware components

Equipment	Size(cm)	Weight(g)	Power(W)	Cost(\$)
Intel NUC	35.7 x 18.9 x 12	504	65	1200
B200	9.7 x 15.5 x 1.5	350	5	1175
Quectel Modem w/Module	11.9 x 6.9 x 2.1	245	3.7	499
Samsung A22	16.3 x 7.4 x 0.9	203	4	200
Starlink Dish	51.3 x 54.4 x 30.3	4200	62.5	599**
Starlink Router	18 x 25.4 x 6.5	1000	62.5	599**
Starlink Base	56.3 x 32.8 x 12.2	2900	N/A	599**
Amarisoft Omni Antenna	3 x 3 x 11.5	90	N/A	15
Lcom Omni Antenna	7.5 x 7.5 x 7.5	230	N/A	85
Egasky Directional Antenna	30 x 5 x 25	517	N/A	45
**sold as package				

In total, the amount of stored memory required for all the software associated with our constructed network is around 650 MB. Table 3.2 shows an approximation of physical memory necessary for each of the modules individually.

Table 3.2. Software memory requirements

Component	Memory (MB)
srsRAN	37
Open5GS	80
USRP	500
Dependencies	25

3.2 Radio Access Network

The RAN provides an RF entry point for the network and handles all the signalling and messaging necessary for UEs to search, find, connect, and send data. The open source software srsRAN, developed by Software Radio Systems, was used to create the RAN component of our network. srsRAN provides a full-stack software gNB solution for 5G SA, implemented in C/C++ and supporting a wide range of baseband hardware platforms including x86, ARM, and PowerPC [50]. srsRAN can also be integrated into mostly any generic SDR as long as the correct dependencies are downloaded on the controlling device. The software is available through a GitHub repository and updated at least quarterly to fix bugs and security flaws, as well as add additional functionality in line with 3GPP Releases. Listing 3.1 shows the code executed to download the srsRAN package.

Listing 3.1: srsRAN download

```
#Standard srsRAN packages will suffice with the
#USRP Hardware Driver (UHD) Software
sudo add-apt-repository ppa:softwareradiosystems/srsran
sudo apt-get update
sudo apt-get install srsran -y
```

In order for the SDR to function properly and take srsRAN software inputs from the controlling NUC computer, dependencies need to be properly downloaded. Listing 2 in the appendix shows the necessary command line execution to facilitate functionality for the UHD, GNU radio, field-programmable gate array (FPGA) software, as well as configuration

of USB settings to allow for handling of plug and unplug events.

Of note, when running the srsRAN software on the SDR, according to [51], the CPU should be set to performance mode to maximize allocated RAM and power output:

```
echo "performance" | sudo tee /sys/devices/system/cpu/  
cpu*/cpufreq/scaling_governor
```

Next, changes must be made to the srsRAN configuration files for the RAN to properly connect to the 5GC. Those changes are noted in Listing 3.2 with highlighting to emphasize on critical settings. The MCC and MNC must match those used by the 5GC and the UEs, particularly the USIM PLMN settings. Also, the *mme_addr* corresponds to the address associated with the NG Application Protocol (NGAP) used for control plane signalling between the RAN and AMF of the 5GC [52]. This address must match the correlated Open5GS setting, as seen in the 5GC section of this chapter. More information about the RF and scheduler settings are located in Chapter 4 under the RF considerations section, where testing and optimization of the RF gain and MCS values are conducted.

Several settings for the gNB must be adjusted as well. First, ensure that the 16-bit TAC matches that of the 5GC [34]. Additionally, within the gNB settings the ARFCN must fall within the band parameters and also have a corresponding CORESET [27]. CORESET values range between 0 and 11 and indicate several physical layer settings for control plane messages. More information about the ARFCN ranges for each frequency band can be found in Table 4.17.

Listing 3.2: srsRAN config changes

enb.conf

```
#MCC and MNC changed to desired settings  
#must match Open5GS and USIM PLMN settings  
#mme_addr set to allow RAN to connection  
#to AMF and UPF  
  
[enb]  
enb_id = 0x19B
```

```

mcc = 910
mnc = 70
mme_addr = 127.0.0.2
gtp_bind_addr = 127.0.1.1
slc_bind_addr = 127.0.1.1
slc_bind_port = 0
n_prb = 50
#tm = 4
#nof_ports = 2

[rf]
#transmit gain adjusted as desired
#maximum value of 79
tx_gain = 79
#rx_gain follows automatic gain control if not specified
#rx_gain = 60

[scheduler]
#need to adjust MCS for RF environment
#and different equipment, particularly SDR
nr_pdsch_mcs=10
nr_pusch_mcs=10

rr.conf
#add NR Cell to the cell list and delete LTE cells
#example shown of an NR cell operating at FR1 Band 3
#368500 – equates to Band 3 at 1865 MHz frequency
#TAC must match AMF in 5GC
#Used TAC of 1 for simplicity
#CORESET must be valid for particular ARFCN

nr_cell_list =

```

```
(  
    {  
        rf_port = 0;  
        cell_id = 1;  
        root_seq_idx = 1;  
        tac = 1;  
        pci = 500;  
        dl_arfcn = 368500;  
        coreset0_idx = 6;  
        band = 3;  
    }  
);
```

3.2.1 RAN Success

Figure 3.2 displays successful establishment of the RAN, where several messages indicate proper gNB function:

1. **"NG connection successful"** - RAN successfully connected to 5GC (note that 5GC must be running first)
2. **"Detected device: B200"** and **"RF Device 'UHD' successfully opened"** - srsRAN program successfully found, connected, and started using B200 SDR for RF transmission
3. **"eNodeB started"** - subsequent frequency information for UL and DL indicate gNB started successfully and began to broadcast MIB and SIB information

```
snuc@snuc-NUC:~$ sudo srsenb ~/Hoffer/configs/srsran/Band3/enb.conf
[sudo] password for snuc:
Active RF plugins: librsran_rf_uhd.so librsran_rf_zmq.so
Inactive RF plugins:
--- Software Radio Systems LTE eNodeB ---

Reading configuration file /home/snuc/Hoffer/configs/srsran/Band3/enb.conf...

Built in Release mode using commit f2dff0b7a on branch master.

Opening 1 channels in RF device=default with args=default
Supported RF device list: UHD zmq file
Trying to open RF device 'UHD'
NG connection successful
[INFO] [UHD] linux; GNU C++ version 9.4.0; Boost_107100; UHD_4.2.0.HEAD-0-g46a70d85
[INFO] [LOGGING] Fastpath logging disabled at runtime.
Opening USRP channels=1, args: type=b200,master_clock_rate=23.04e6
[INFO] [UHD RF] RF UHD Generic instance constructed
[INFO] [B200] Detected Device: B200
[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] Asking for clock rate 23.040000 MHz...
[INFO] [B200] Actually got clock rate 23.040000 MHz.
RF device 'UHD' successfully opened

==== eNodeB started ====
Type <t> to view trace
Setting frequency: DL=1842.5 Mhz, UL=1747.5 MHz for cc_idx=0 nof_prb=52
```

Figure 3.2. Successful RAN configuration with connection to 5GC

3.3 5G Core

The 5GC provides all the background services for the network, such as authentication, subscriber tracking, and communication with outside networks. Open5GS is a C-language open source implementation of the 5GC which consists of two core architecture segments: 4/5G non-standalone (NSA) and 5G SA [53], as depicted in Figure 3.3. The main difference between NSA and SA is that NSA utilizes 4G Core functionality in combination with 4/5G radio networks whereas in SA, a 5G radio is directly connected to a 5GC. Put simply, NSA is a combination of 4G and 5G networks and SA is exclusively 5G. Our network design used a 5G RAN and 5GC to provide 5G SA functionality.

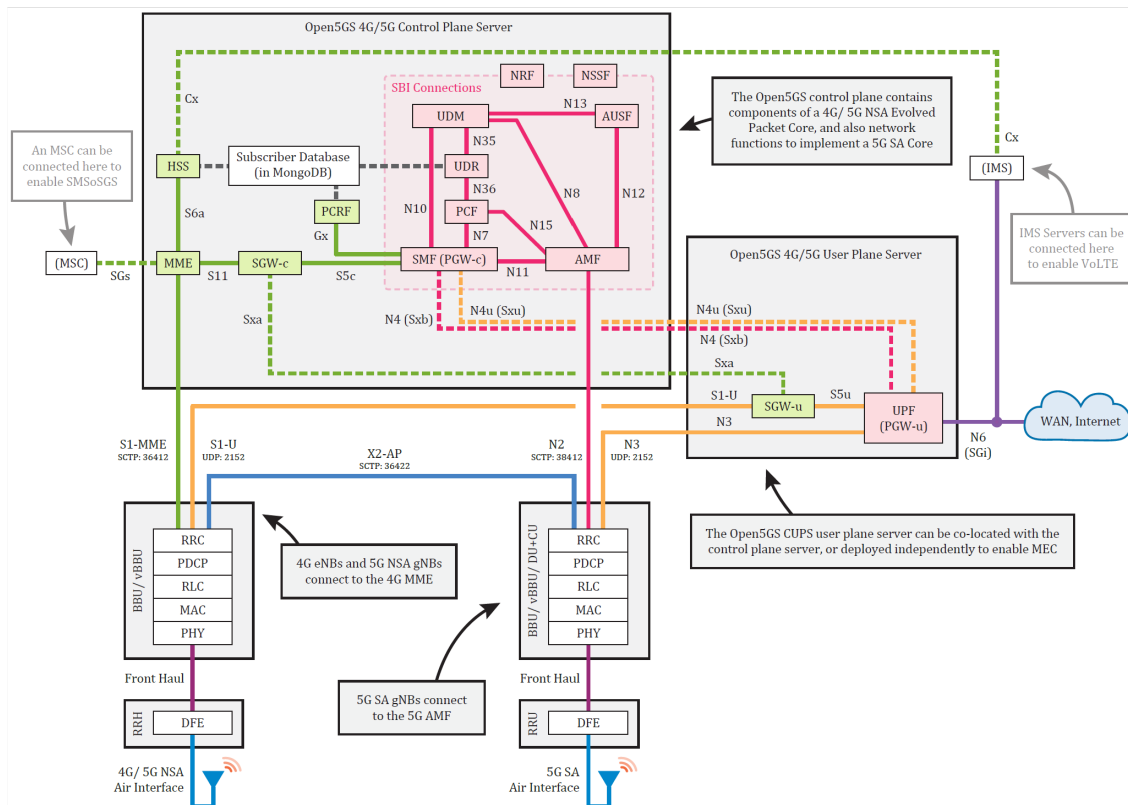


Figure 3.3. Open5GS component diagram. Source [54].

The Intel NUC provided the compute resources needed for the 5GC, instantiating each of the different services while using a local addressing scheme for intercommunication. By default, Open5GS settings enable each of the 5GC components to communicate with each

other by using the local loopback address space (127.0.0.0/8) [53]. Of note, inter-service communication between core services either occurs directly, or indirectly through a service communication proxy (SCP) which was introduced in Release 16 [29]. SCP allows the control plane to serve as the mediator of all signalling and control plane messages in the 5GC. As a result, the SCP can help solve difficulties related to prioritization, traffic routing, and load balancing by providing the ability to route traffic through alternative paths [55]. The most recent Open5GS update includes SCP functionality as the default. Table 3.3 shows the IP addressing used for the 5GC in relation to Figure 3.3.

Table 3.3. Open5GS IP addressing

Link Name	Component Connection	IP Address
Subscriber Database	UDR to MongoDB	127.0.0.1:3000
N2	gNB to AMF	127.0.0.5:38412
N3	gNB to UPF	127.0.0.7:2152
N4	SMF to UPF	127.0.0.7:8805
N6	UPF to Data Network	Public IP Address
SMF-sbi	SMF	127.0.0.4:7777
AMF-sbi	AMF	127.0.0.5:7777
NRF-sbi	NRF	127.0.0.10:7777
AUSF-sbi	AUSF	127.0.0.11:7777
UDM-sbi	UDM	127.0.0.12:7777
PCF-sbi	PCF	127.0.0.13:7777
NSSF-sbi	NSSF	127.0.0.14:7777
BSF-sbi	BSF	127.0.0.15:7777
UDR-sbi	UDR	127.0.0.20:7777
SCP-sbi	SCP	127.0.1.10:7777
*sbi - service-based interface		

3.3.1 Open5GS Download and Configuration

Listing 3.3 shows the code executed from [54] to download the publicly-available Open5GS software through their GitHub repository. It also shows how to setup the web GUI for the UDM core service so that we can more easily manage our subscribers through a graphical interface.

Listing 3.3: Open5GS download

```
sudo apt update
#repo downloads through ppa
sudo apt install software-properties-common
sudo add-apt-repository ppa:open5gs/latest
sudo apt update
sudo apt install open5gs

#installation of the web GUI for Open5GS:

sudo apt update
#download curl: command-line tool to
#transfer data to or from a server
sudo apt install curl
curl -fsSL https://deb.nodesource.com/setup_14.X
\ | sudo -E bash -
#need nodejs, then can download the Open5GS GUI
sudo apt install nodejs
curl -fsSL https://open5gs.org/open5gs/assets/webui/install
\ | sudo -E bash -

#Open5GS web GUI login information:
#http://localhost:3000
#user: admin
#password: 1423
```

Modifications must be made to the default Open5GS settings. A network interface for the

5GC must also be created, with subsequent manipulation of the IP tables. The changes include:

1. Verifying loopback addresses consistent with srsRAN configurations.
2. Adjusting PLMN, TAC, AMF ID, network identification, and integrity and enciphering security settings.
3. Establishing Open5GS internal network interfaces.
4. Adding subnet addressing for the UPF.
5. Enabling IP forwarding and creating NAT rules for communication between 5GC and the Internet-facing interface of the CPU.
6. Adding subscribers to the UDM.

Listing 3.4 shows the changes made to the AMF and UPF configuration files, highlighting indicates the critical features for successful operation of our testbed.

Listing 3.4: Open5GS configs

AMF

```
#configure desired PLMN – must match srsRAN
#verify ngap address to match srsRAN
#choose integrity and ciphering order preferences
#can change network name as desired
```

amf:

```
  sbi:
    - addr: 127.0.0.5
      port: 7777
  ngap:
    - addr: 127.0.0.2
  guami:
    - plmn_id:
        mcc: 901
        mnc: 70
      amf_id:
        region: 2
```

```

        set: 1
tai:
  - plmn_id:
      mcc: 901
      mnc: 70
      tac: 1
plmn_support:
  - plmn_id:
      mcc: 901
      mnc: 70
    s_nssai:
      - sst: 1
security:
  integrity_order: [NIA2, NIA1, NIA0]
  ciphering_order: [NEA0, NEA1, NEA2]
network_name:
  full: Open5GS
amf_name: open5gs-amf0

```

UPF

```

#verify pfcf and gtpu addresses match AMF settings
#update subnet addresses accordingly
upf:

```

```

  pfcf:
    - addr: 127.0.0.7
  gtpu:
    - addr: 127.0.0.2
  subnet:
    - addr: 10.45.0.1/16
    - addr: 2001:db8:cafe::1/48

```

3.3.2 Networking Configurations

Our UPF contains a subnet of 10.45.0.1/16 (IPv4) and 2001:db8:cafe::1/48 (IPv6), where UEs will be assigned an IP address once fully authenticated to the network and a subsequent PDU session is established by the SMF. The connected devices within the subnet need a way to communicate externally to the Internet through the 5GC, so a tunnel must be created for the Open5GS network interface (ogstun) permitting messages (packets) to travel out to the Internet. This can be done by enabling IP forwarding and then creating a NAT rule for the UPF, as outlined in [54] and seen in Listing 3.5.

Listing 3.5: 5GC network interface and IP Tables

```
#add interface for Open5GS network tunnel
ip tuntap add name ogstun mode tun
#add subnet addresses – must match UPF settings
ip addr add 10.45.0.0/16 dev ogstun
ip addr add 2001:db8:cafe::1/48 dev ogstun
ip link set ogstun up

#enable forwarding and add NAT rule for
#bridging the PGWU/UPF to the WAN
#allows network traffic from UPF subnet to reach Internet
sudo sysctl -w net.ipv4.conf.all.forwarding=1
sudo sysctl -w net.ipv6.conf.all.forwarding=1

#command for masquerading subnet
sudo iptables -t nat -A POSTROUTING -s
\ 10.45.0.0/16 ! -o ogstun -j MASQUERADE
sudo ip6tables -t nat -A POSTROUTING -s
\ 2001:db8:cafe::/48 ! -o ogstun -j MASQUERADE

#ensure that the packets in the ‘INPUT’ chain
#to the ‘ogstun’ interface are accepted
sudo iptables -I INPUT -i ogstun -j ACCEPT
```

```
#might need to disable firewall to allow packets through
#disable ufw
ufw disable
```

IP forwarding enables internal nodes within a firewall to communicate with each other, and the firewall needs to be configured for IP masquerading so that requests can be masked from internal nodes with the IP address of the forward-facing node. Furthermore, POSTROUTING allows the packets to be adjusted when leaving the external device and the MASQUERADE target specifies masking the internal private IP address with that of the external address. Verification of the modified network configurations can be seen by listing the IP table and NAT rules through `sudo iptables -L` and `sudo iptables -L -t nat` respectively, as seen in Figure 3.4.

```
snuc@snuc-NUC:~$ sudo iptables -L
[sudo] password for snuc:
Chain INPUT (policy ACCEPT)
target     prot opt source                destination
ACCEPT     all  --  anywhere              anywhere

Chain FORWARD (policy ACCEPT)
target     prot opt source                destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source                destination
snuc@snuc-NUC:~$ sudo iptables -L -t nat
Chain PREROUTING (policy ACCEPT)
target     prot opt source                destination

Chain INPUT (policy ACCEPT)
target     prot opt source                destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source                destination

Chain POSTROUTING (policy ACCEPT)
target     prot opt source                destination
MASQUERADE all  --  10.45.0.0/16          anywhere
```

Figure 3.4. IP table verification

3.3.3 Core Success

Figure 3.5 lists the running 5GC components after starting each of the services. You can utilize the command `ps aux | grep open5gs` to display all the processes running on your Linux system containing *open5gs*. Another way to verify that each of the 5GC services are running is to investigate the log files and looking for the *NF registered* message, indicating that the service successfully connected to the NRF. As an example, Figure 3.6 shows the AMF log file indicating NF registration after restarting the service.

```
snuc@snuc-NUC:~$ ps aux | grep open5gs
open5gs 468760 0.3 0.0 2777888 13524 ? Ssl 15:00 0:00 /usr/bin/open5gs-mmec -c /etc/open5gs/mme.yaml
open5gs 468764 0.0 0.0 259024 18280 ? Ssl 15:00 0:00 /usr/bin/open5gs-sgwcd -c /etc/open5gs/sgwc.yaml
open5gs 468772 0.5 0.0 3117180 39960 ? Ssl 15:00 0:00 /usr/bin/open5gs-smfd -c /etc/open5gs/smf.yaml
open5gs 468777 0.0 0.0 150504 17524 ? Ssl 15:00 0:00 /usr/bin/open5gs-amfd -c /etc/open5gs/amf.yaml
open5gs 468787 0.0 0.0 248040 16436 ? Ssl 15:00 0:00 /usr/bin/open5gs-sgwud -c /etc/open5gs/sgwu.yaml
open5gs 468825 0.0 0.0 266028 20724 ? Ssl 15:00 0:00 /usr/bin/open5gs-upfd -c /etc/open5gs/upf.yaml
open5gs 468829 0.4 0.0 2801692 16728 ? Ssl 15:00 0:00 /usr/bin/open5gs-hssd -c /etc/open5gs/hss.yaml
open5gs 468834 0.5 0.0 2800264 16908 ? Ssl 15:00 0:00 /usr/bin/open5gs-pcrfd -c /etc/open5gs/pcrf.yaml
open5gs 468859 0.0 0.0 130472 15604 ? Ssl 15:00 0:00 /usr/bin/open5gs-nrfd -c /etc/open5gs/nrf.yaml
open5gs 468890 0.0 0.0 132208 14792 ? Ssl 15:00 0:00 /usr/bin/open5gs-ausfd -c /etc/open5gs/ausf.yaml
open5gs 468940 0.0 0.0 132208 13304 ? Ssl 15:00 0:00 /usr/bin/open5gs-udmd -c /etc/open5gs/udm.yaml
open5gs 468945 0.0 0.0 182020 19944 ? Ssl 15:00 0:00 /usr/bin/open5gs-pcfd -c /etc/open5gs/pcf.yaml
open5gs 468967 0.0 0.0 130456 13264 ? Ssl 15:00 0:00 /usr/bin/open5gs-nssf -c /etc/open5gs/nssf.yaml
open5gs 468996 0.0 0.0 137780 13240 ? Ssl 15:00 0:00 /usr/bin/open5gs-bsfd -c /etc/open5gs/bsf.yaml
open5gs 469002 0.0 0.0 166420 18552 ? Ssl 15:00 0:00 /usr/bin/open5gs-udrd -c /etc/open5gs/udr.yaml
open5gs 469060 0.1 0.0 143260 16792 ? Ssl 15:00 0:00 /usr/bin/open5gs-scpd -c /etc/open5gs/scp.yaml
snuc 469066 0.0 0.0 9040 656 pts/5 S+ 15:00 0:00 grep --color=auto open5gs
```

Figure 3.5. 5GC services running properly as indicated by the operating system

```
File Edit Search View Tools Options Language Buffers Help
1 amf.log
Open5GS daemon v2.5.4
01/19 15:13:40.810: [app] INFO: Configuration: '/etc/open5gs/amf.yaml' (./lib/app/oqs-init.c:126)
01/19 15:13:40.810: [app] INFO: File Logging: '/var/log/open5gs/amf.log' (./lib/app/oqs-init.c:129)
01/19 15:13:40.813: [sbi] INFO: NF Service [namf-comm] (./lib/sbi/context.c:1401)
01/19 15:13:40.813: [sbi] INFO: nghttp2 server() [127.0.0.5]:7777 (./lib/sbi/nghttp2-server.c:150)
01/19 15:13:40.813: [amf] INFO: ngap server() [127.0.0.2]:38412 (./src/amf/ngap-sctp.c:61)
01/19 15:13:40.813: [sctp] INFO: AMF initialize...done (./src/amf/app.c:33)
01/19 15:13:40.814: [sbi] INFO: [e9586cc6-984e-41ed-a22b-ed275adfa9c] NF registered [Heartbeat:10s] (./lib/sbi/nf-sm.c:214)
```

Figure 3.6. AMF log file indicating NF registration and successful service startup

3.3.4 Adding Subscribers

The easiest way to add or remove subscribers from the 5GC involves using the web GUI, located at <http://localhost:3000>, and visually shown in Figure 3.7. It uses MongoDB

on the backend to maintain the record entries and can also be accessed directly through the command line with: `mongo > use open5gs`, as seen in Figure 3.8.

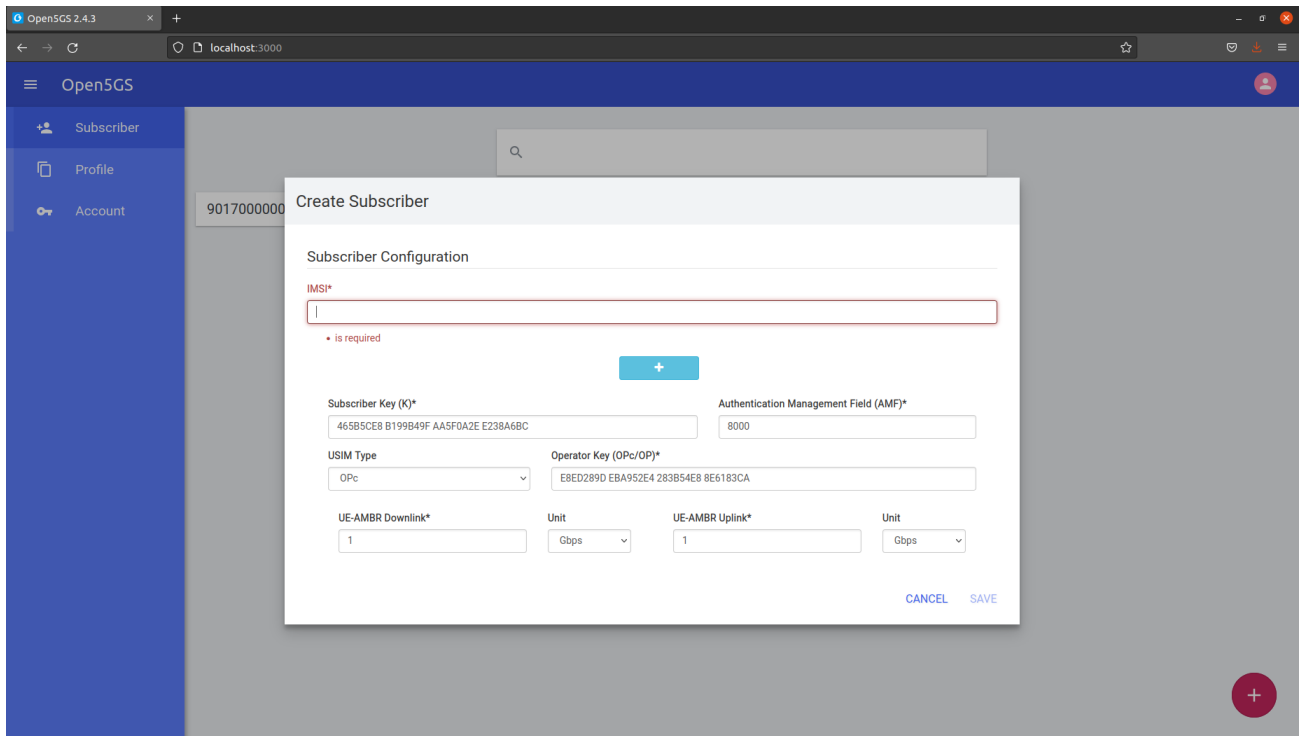


Figure 3.7. UDM web GUI

```

snuc@snuc-NUC:~$ mongo
MongoDB shell version v3.6.8
connecting to: mongodb://127.0.0.1:27017
Implicit session: session { "id" : UUID(
MongoDB server version: 3.6.8
Server has startup warnings:
2022-09-27T10:20:41.544-0500 I STORAGE
2022-09-27T10:20:41.544-0500 I STORAGE
2022-09-27T10:20:41.544-0500 I STORAGE
2022-09-27T10:20:42.270-0500 I CONTROL
2022-09-27T10:20:42.270-0500 I CONTROL
2022-09-27T10:20:42.270-0500 I CONTROL
2022-09-27T10:20:42.270-0500 I CONTROL
> use open5gs
switched to db open5gs
> db.stats()
{
  "db" : "open5gs",
  "collections" : 4,
  "views" : 0,
  "objects" : 7,
  "avgObjSize" : 670.2857142857143,
  "dataSize" : 4692,
  "storageSize" : 122880,
  "numExtents" : 0,
  "indexes" : 7,
  "indexSize" : 200704,
  "fsUsedSize" : 119584329728,
  "fsTotalSize" : 1968335929344,
  "ok" : 1
}
>

```

Figure 3.8. UDM through MongoDB

For simplicity, we primarily used the web GUI to add and remove subscribers. Once logged into the UDM on the localhost address, you can directly add a subscriber by pressing the *plus* button on the bottom right and then filling out the following requisite user data fields, ensuring they correspond to the subscriber information located on the USIM:

IMSI

OPc or OP value

Subscriber key (K)

Access Point Name (APN) settings*

Note that an APN is the name of the gateway between the 5G network and Internet. The APN name and type entered for that subscriber in the 5GC must match the one added to the UE. For transparency, we named our APN *Internet*, and then used the type *IPv4v6* so that it

could use either IPv4 or IPv6. Additionally, for 5G networks, APN is also known as a Data Network Name (DNN).

3.4 User Equipment

There were two distinct types of UEs tested on our network, with details for each shown in the list below. Specific actions performed on each of the devices to achieve network connection can be found in the UE device connection section of Chapter 4.

1. **Quectel RM500Q-AE modem** - 3/4/5G capable modem with maximum data rates of 2.1 Gbps downlink and 450 Mbps uplink on 5G SA operation. Manufactured in compliance with 3GPP Release 15 and operates on 5G SA 1, 2, 3, 5, 7, 8, 12, 20, 25, 28, 38, 40, 41, 48, 66, 71, 77, 78, 79 sub-6 GHz bands [56]
2. **Samsung A22 5G** - 2/3/4/5G capable commercially-available cellular phone able to operate on 5G SA 1, 3, 5, 7, 8, 28, 38, 40, 41, 78, 79 sub-6 GHz bands, according to the service mode statistics of RF supported bands as depicted in Figure 4.1.

3.4.1 USIM

In addition to the descriptions in Chapter 2, there are specific elements pertaining to USIMs that are required for UE connection and operation. This subsection outlines actions necessary to program the USIM so that the UE can search, decode, send random access messages to the RAN, and fully register on the network through the 5GC. Consequently, key values need to be configured on the USIM file structure that match those associated with the gNB and 5GC:

1. **IMSI and PLMN** - PLMN portion of the IMSI along with separate specific programming of the MCC and MNC. The modified PLMN will also assign a home network for the device to allow for automatic registration. Of note, an associated IMSI entry must be made in the UDM of the core.
2. **K** - secret key of the device used for authentication and message transmission.
3. **OPc** - combination of the OP and K keys used for the mutual authentication process.

Additionally, the USIM service table (UST) within the file structure must be properly programmed:

1. **UST** - Numerous important services are indicated and described in Table 3.4, but specific services that need to be activated or deactivated include:
 - (a) **Service 124** - Subscription identifier privacy support** - conceals the SUPI (IMSI in our network) by converting into the SUCI.
 - If activated, EF.SUCI_Calc_Info and EF.RoutingIndicator files must be present to facilitate the SUPI to SUCI conversion
 - If deactivated, SUPI will be sent in the clear on your network. For transparency of observing and digesting log files, as well as operability, we decided to deactivate Service 124.
 - (b) **Service 125 - SUCI Calculation by the USIM** - sysmocom-ISIM does not support on-SIM calculation, so it must be deactivated
2. **SUCI_Calc_Info** - if service 124 is activated, public keys corresponding to the network operator must be added to EF.SUCI_Calc_Info, as well as the UDM of the 5GC so that the device can encrypt the SUPI to the SUCI and the 5GC can subsequently decrypt
3. **Routing indicator** - used as part of the SUCI calculation and by default an empty file, has to be present if service 124 for use of SUPI is activated

**For our testing, in order to enable more transparent observation of the network control messaging, we deactivated Service 124 so that the SUPI was used in lieu of encrypting to the SUCI.

Table 3.4. Important services in UST. Adapted from [32].

Number	Service	Description
35	APN Control List (ACL)	List of allowed APNs
42	Operator controlled PLMN selector with Access Technology	Preferred PLMNs
43	HPLMN selector with access technology	Home network priorities
45	PLMN network name	Network name for registered PLMN
46	Operator PLMN list	Prioritized list of TAIs
71	Equivalent HPLMN	Establishment of multiple home networks
122	5GS Mobility Management Information	Facilitates core to user messaging
123	5G Security Parameters	Contains secure temporary key for 5G
124	Subscription identifier privacy support	Indicates support for use of SUPI
125	SUCI Calc Info	SUCI calculation on USIM or device
126	Unified Access Control (UAC) Access Identities support	List of multi-media services
129	5GS Operator PLMN list	TAI identities associated with service 45

For our USIM, we used the sysmoISIM-SJA2 as in seen in Figure 3.9, which is an ET-SI/3GPP standards-compliant SIM that allows for the programming of the IMSI, K, OPc, and any file within the UICC file system structure [33]. The manual for the sysmoUSIM contains extensive information about its capabilities and provides a reference for many programming operations [33].

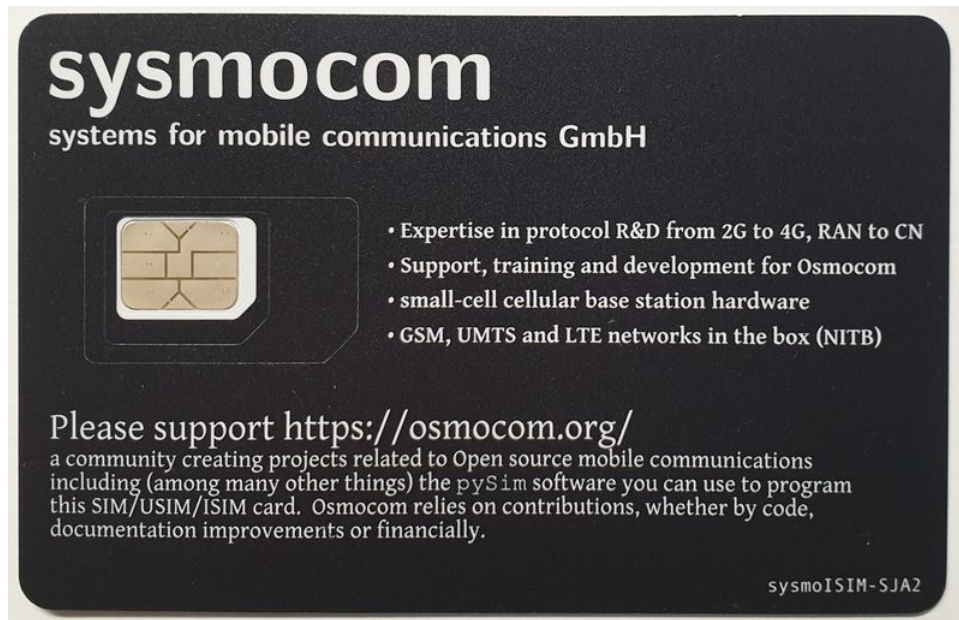


Figure 3.9. sysmocom ISIM. Source [33].

USIM Programming

There are several open source options for programming USIMs, but we decided to use a suite of Python files called pySim due to our familiarity with Python and reported robustness of the program packages. The programs are described in Table 3.5 and can all be run on the command line. Listing 3.6 shows the necessary steps to download the pySim module and the corresponding dependencies.

Table 3.5. pySim programs

pySim-read	Reads SIM and outputs commonly used fields
pySim-program	Easy programming of USIM for smaller subset of configurations like IMSI, K, OPc, PLMN, and MSISDN
pySim-shell	Enables read and write access to the USIM file system and administrative commands

Listing 3.6: pySim download

```
#download dependencies
#ensure python3 is already installed
```

```
sudo apt-get install pcscd pcsc-tools libccid  
\ libpcsc-lite-dev python-pyscard pyscard serial  
\ pytlv cmd2 == 1.5 jsonpath-ng construct bidict gsm0338  
  
#use git clone to grab pySim code from GitHub  
git clone https://gitea.osmocom.org/sim-card/pysim.git  
  
#then cd into pysim to run programs directly  
cd pysim
```

pySim-program can be used to quickly modify the USIM while maintaining the ability to configure numerous options as seen in Figure 3.10. It also ratchets through numerous EFs within the file structure and changes values as indicated by the command line inputs. For example, when you program an IMSI change through pySim-program, not only does EF.IMSI get modified, but also the associated PLMN in the EF.PLMNwACT, EF.HPLMNwACT, EF.OPLMNwACT, EF.EHPLMN, EF.OPL, and several other EFs within the ADF.USIM application dedicated file (ADF). Therefore, pySim-program ensures that each of the broader USIM parameters are correctly adjusted throughout the file system. However, pySim-shell has more capabilities and is still necessary to fine tune certain aspects of the USIM such as the UST.

```

snuc@snuc-NUC:/bin/pysim$ ./pySim-prog.py -help
Usage: pySim-prog.py [options]

Options:
-h, --help            show this help message and exit
-d DEV, --device=DEV  Serial Device for SIM access [default: /dev/ttyUSB0]
-b BAUD, --baud=BAUD  Baudrate used for SIM access [default: 9600]
-p PCSC, --pcsc-device=PCSC
                        Which PC/SC reader number for SIM access
--modem-device=DEV    Serial port of modem for Generic SIM Access (3GPP TS
                        27.007)
--modem-baud=BAUD     Baudrate used for modem's port [default: 115200]
--osmocon=PATH        Socket path for Calypso (e.g. Motorola C1XX) based
                        reader (via OsmocomBB)
-t TYPE, --type=TYPE  Card type (user -t list to view) [default: auto]
-T, --probe           Determine card type
-a PIN_ADM, --pin-adm=PIN_ADM
                        ADM PIN used for provisioning (overwrites default)
-A PIN_ADM_HEX, --pin-adm-hex=PIN_ADM_HEX
                        ADM PIN used for provisioning, as hex string (16
                        characters long)
-e, --erase           Erase beforehand [default: False]
-S SOURCE, --source=SOURCE
                        Data Source [default: cmdline]
-n NAME, --name=NAME  Operator name [default: Magic]
-c CC, --country=CC   Country code [default: 1]
-x MCC, --mcc=MCC     Mobile Country Code [default: 901]
-y MNC, --mnc=MNC     Mobile Network Code [default: 55]
--mnc-len=MNCLEN     Length of Mobile Network Code [default: 2]
-m SMSC, --smc=SMSC   SMSC number (Start with + for international no.)
                        [default: '00 + country code + 5555']
-M SMSPP, --smpp=SMSP Raw SMSP content in hex [default: auto from SMSC]
-s ID, --iccid=ID     Integrated Circuit Card ID
-i IMSI, --imsi=IMSI  International Mobile Subscriber Identity
--msisdn=MSISDN       Mobile Subscriber Integrated Services Digital Number
-k KI, --ki=KI        Ki (default is to randomize)
-o OPC, --opc=OPC     OPC (default is to randomize)
--op=OP              Set OP to derive OPC from OP and KI
--acc=ACC            Set ACC bits (Access Control Code). not all card types
                        are supported
--opmode=OPMODE       Set UE Operation Mode in EF.AD (Administrative Data)
--epdgid=EPDGID       Set Home Evolved Packet Data Gateway (ePDG)
                        Identifier. (Only FQDN format supported)
--epdgSelection=EPDGSELECTION
                        Set PLMN for ePDG Selection Information. (Only
                        Operator Identifier FQDN format supported)
--pcscf=PCSCF         Set Proxy Call Session Control Function (P-CSCF)
                        Address. (Only FQDN format supported)
--ims-hdomain=IMS_HDOMAIN
                        Set IMS Home Network Domain Name in FQDN format
--impi=IMPI           Set IMS private user identity
--impu=IMPU           Set IMS public user identity
--read-imsi           Read the IMSI from the CARD
--read-iccid          Read the ICCID from the CARD
-z STR, --secret=STR  Secret used for ICCID/IMSI autogen
-j NUM, --num=NUM     Card # used for ICCID/IMSI autogen
--batch              Enable batch mode [default: False]
--batch-state=FILE    Optional batch state file
--read-csv=FILE       Read parameters from CSV file rather than command line
--write-csv=FILE      Append generated parameters in CSV file
--write-hlr=FILE      Append generated parameters to OpenBSC HLR sqlite3
--dry-run            Perform a 'dry run', don't actually program the card
--card-handler=FILE   Use automatic card handling machine

```

Figure 3.10. pySim programming options

Figure 3.11 displays an example of USIM programming with the pySim-program script where the IMSI, PLMN, K, and OPc were changed. Of note when programming using pySim-program, the default setting randomizes the K and OPc values when overwriting anything on the USIM unless explicitly designated. You must also have the admin key, sometimes labeled as ADM, in order to modify any values on the USIM. This ADM value was provided by the manufacturer upon purchase of the USIM.

```
snuc@snuc-NUC:/bin/pysim$ sudo ./pySim-prog.py -p0 -t sysmoISIM-SJA2 -i 90170000005154
9 -x 901 -y 70 -s 8988211000000515497 -a 85613955 -o AAAAAABBBBBBCCCCDDDDDEEEEEFFF
-k 11112222333344445555666677778888
QStandardPaths: XDG_RUNTIME_DIR not set, defaulting to '/tmp/runtime-root'
Using PC/SC reader interface
Ready for Programming: Insert card now (or CTRL-C to cancel)
Generated card parameters :
> Name      : Magic
> SMSP      : e1fffffffffffffffffffff0581005155f5ffffffffffff000000
> ICCID     : 8988211000000515497
> MCC/MNC   : 901/70
> IMSI      : 901700000051549
> Ki        : 11112222333344445555666677778888
> OPC       : AAAAAABBBBBBCCCCDDDDDEEEEEFFF
> ACC       : None
> ADM1(hex) : 3835363133393535
> OPMODE    : None
Programming ...
Warning: Programming of the ICCID is not implemented for this type of card.
Programming successful: Remove card from reader
```

Figure 3.11. Programming for USIM using pySim-prog.py

As mentioned, pySim-shell allows for more extensive programming of the USIM. When using pySim-shell, the MF of the USIM file structure will be opened upon program execution. From there, you must verify the admin keys for the USIM by running `verify_adm X`, where the X represents ADM. Similar to using pySim-program, these admin keys must be verified to modify anything within the file structure.

Once you have verified the admin key, you are ready to start navigating through the file system and individually modifying particular EFs. Figure 3.12 displays a simplified version of the 5G USIM file structure, indicating particular EFs of importance that must be either configured or verified before a UE can register on our network.



Figure 3.12. USIM file structure

Many of the EFs are simply JSON files, so reading and updating them can be done directly on the command line. To navigate within the file structure, you just need to `/textttselect` the desired file. Listing 3.7 shows how to deactivate Service 124 within the UST, while Listings 3.8 and 3.9 show how to read and update the SUCI_Calc_Info and Routing_Indicator EFs. EF.SUCI_Calc_Info would have to be modified according to [57], to include a prioritized listing of the network public key or keys used for asymmetric encryption. These keys would then also have to be mirrored in the 5GC so that the network could subsequently decrypt the SUCI using their private key. A particular note about the routing indicator is that the four-digit value only uses the first two bytes of the four-byte EF.Routing_Indicator file [33].

Listing 3.7: UST activation and deactivation

```
#navigate to the file
> select ADF.USIM
> select EF.UST

#display the list of services and their associated status
> read_binary_decoded

#to deactivate service 124
> ust_service_deactivate 124
```

Listing 3.8: SUCI_Calc_Info

```
#navigate to the file
> select ADF.USIM
> select DF.5GS
> select EF.SUCI_Calc_Info

#read the file
> read_binary_decoded

#updating the file if activating Service 124 and
#encrypting SUPI to SUCI
#XXXX represents the public keys inputted as a JSON file
> update_binary_decoded '{XXXX}'
```

Listing 3.9: Routing_Indicator

```
#navigate to the file
> select ADF.USIM
> select DF.5GS
> select EF.Routing_Indicator

#read the file
> read_binary_decoded
```

```
#if you wanted to update the file with routing indicator 1
#keep in mind that routing indicator takes up only
#two bytes of total four byte file
> update_binary 0001ffff
```

3.5 Network Backhaul

Network backhaul provides access to the Internet in our 5G network. Once a UE is assigned an IP address and PDU session from the SMF, the UE can access the Internet from the UPF. There are several options for supplying network backhaul to an open source network to include, but not limited to ethernet, Wi-Fi, separate mobile hot spot, and SatNets. Only SatNets provide an Internet capability in austere signal environments with no organic network connectivity. Referring back to Figure 3.1, the backhaul of our 5G network was provided by Starlink's LEO satellite Internet service. A current coverage map of Starlink's available services can be seen in Figure 3.13.

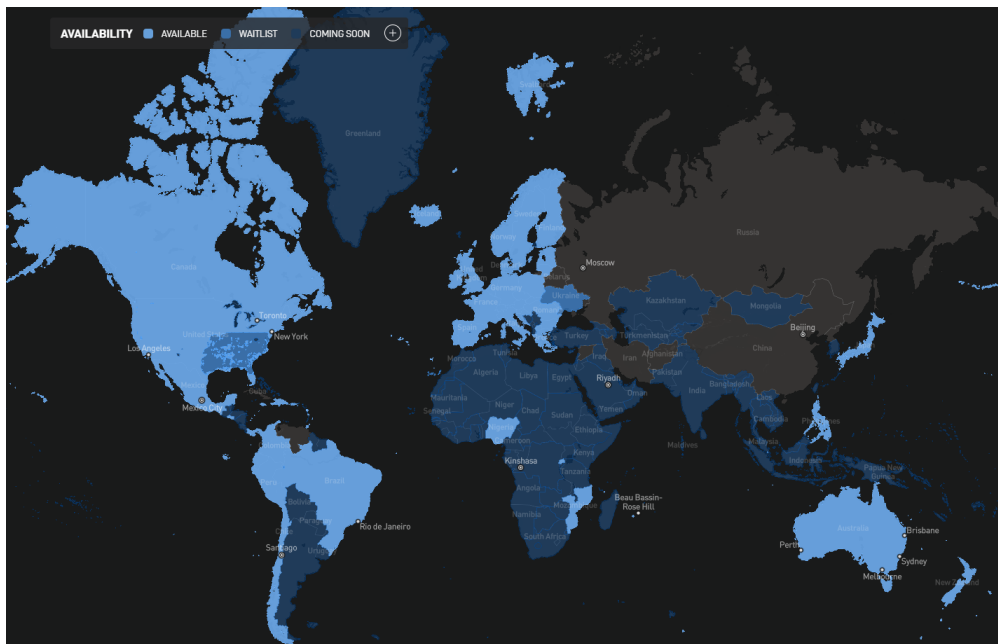


Figure 3.13. Starlink coverage map as of June 2023. Source [44].

3.5.1 Starlink Setup

The Starlink components necessary for integration into our network included simply the satellite dish, router, and power supply. Additionally, a separate mobile phone application called *Starlink* can be found on any device application store and is used to configure and calibrate the Starlink antenna. Once turned on, the Starlink system performs a scan to determine proper antenna angle for best connectivity. This process takes about 30 seconds to one minute, ending with the mechanical movement of the satellite dish from its original vertical starting position to a proper azimuth for optimal connection. Internet access can be verified within the mobile phone application after the dish maneuvers into position.

Once the dish moved into place and connected to the Starlink network, we connected our NUC to the router via an ethernet cable and port, as depicted in Figure 3.1. At this point, since our IP forwarding and masquerading were already setup, the UE connected to our 5G network was able to readily access the data network and Internet through the Starlink SatNet.

3.6 Summary

The successful configuration of our 5G network was described in this chapter. Software and hardware dependencies were managed and all four disparate parts of a 5G network were correctly deployed and interconnected. Next, Chapter 4 details our network testing and evaluation, attempting to answer our proposed research questions.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 4: Results

This chapter examines the results from our network implementation in Chapter 3. We answer our proposed research questions from Chapter 1 and investigate necessary UE and network modifications for successful connection. Additionally, we optimize network performance through available settings, conduct throughput testing with a satellite backhaul, determine network proximity for UEs, and describe and outline use cases where our network could be adopted.

4.1 UE Device Connection

One of the main goals of this thesis entailed establishing end-to-end device connection via our network and out through the Internet. Device connection from a UE out to the data network requires a succession of steps from UE searching of the network cell to communicating out of the network through the UPF. These steps were outlined in detail in Chapter 2 and 3, but a consolidated list consists of six broad and distinct events:

1. UE and RAN RACH procedure resulting in RRC connected
2. UE added to AMF
3. UE registration with mutual authentication between UE and 5GC
4. PDU session established by SMF
5. SMF assigns UE an IP address within the 10.45.0.0/16 subnet of our network
6. UE gains access to UPF through a new logical connection (UE > UPF > Data Network), establishing pipeline from UE to Internet

Each of these sequenced events can be verified through an inspection of the various 5GC log files, gNB traces, and the UE itself:

1. gNB trace from the srsRAN command line startup indicating a RACH procedure and individual RACH messages contained in the gNB log file
2. AMF log file
3. AMF log file

4. SMF log file
5. SMF log file and IP address allocation apparent on the UE
6. UPF log file and verified by Internet connection on UE

4.1.1 Samsung A22

To configure the Samsung A22, it was necessary to use the service mode under developer settings to be able to both search for our network and then successfully iterate through the registration process. We found that the device was not able to search for our network while stepping through multiple cellular technologies. As a result, we had to lock the phone to searching for only 5G networks in order to permit discovery of our network. Additionally, specific 5G NR bands had to be enabled that corresponded to the usable bands of our network, as seen in Table 4.17. The full listing of 5G-capable bands that are supported on the device can be seen on the bottom of Figure 4.1.

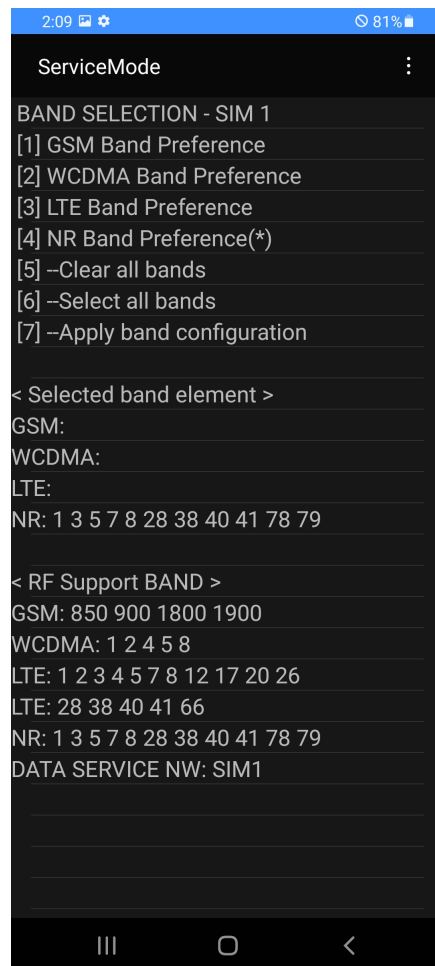


Figure 4.1. Screen capture of Samsung A22 service mode indicating capable 5G NR bands

To get to the service mode, dial *#0011# for the basic developer menu. Then, type "Q" and "0000" as input, successively, to get to the extended service mode menu. Once in service mode, we were able to lock the bands to 5G only and then select and implement all of the Samsung's capable bands as shown in Figure 4.1 and outlined in the step list below:

1. **Mode Locking** - SERVICE MENU > UE SETTING and INFO > SETTING > PROTOCOL > NAS > NETWORK CONTROL > PREF MODE SET > NEXT PAGE > NR ONLY
2. **Band Selection** - SERVICE MENU > UE SETTING and INFO > SETTING >

PROTOCOL > NAS > NETWORK CONTROL > BAND SELECTION > BAND SELECTION SIM 1 > NR BAND PREFERENCE > NR ALL

Connection Success

After those settings were changed, we were finally able to discover and subsequently connect to our open source 5G network. Figure 4.2 shows both the service mode report on the left and the SIM card report on the right, indicating a connection to our network.

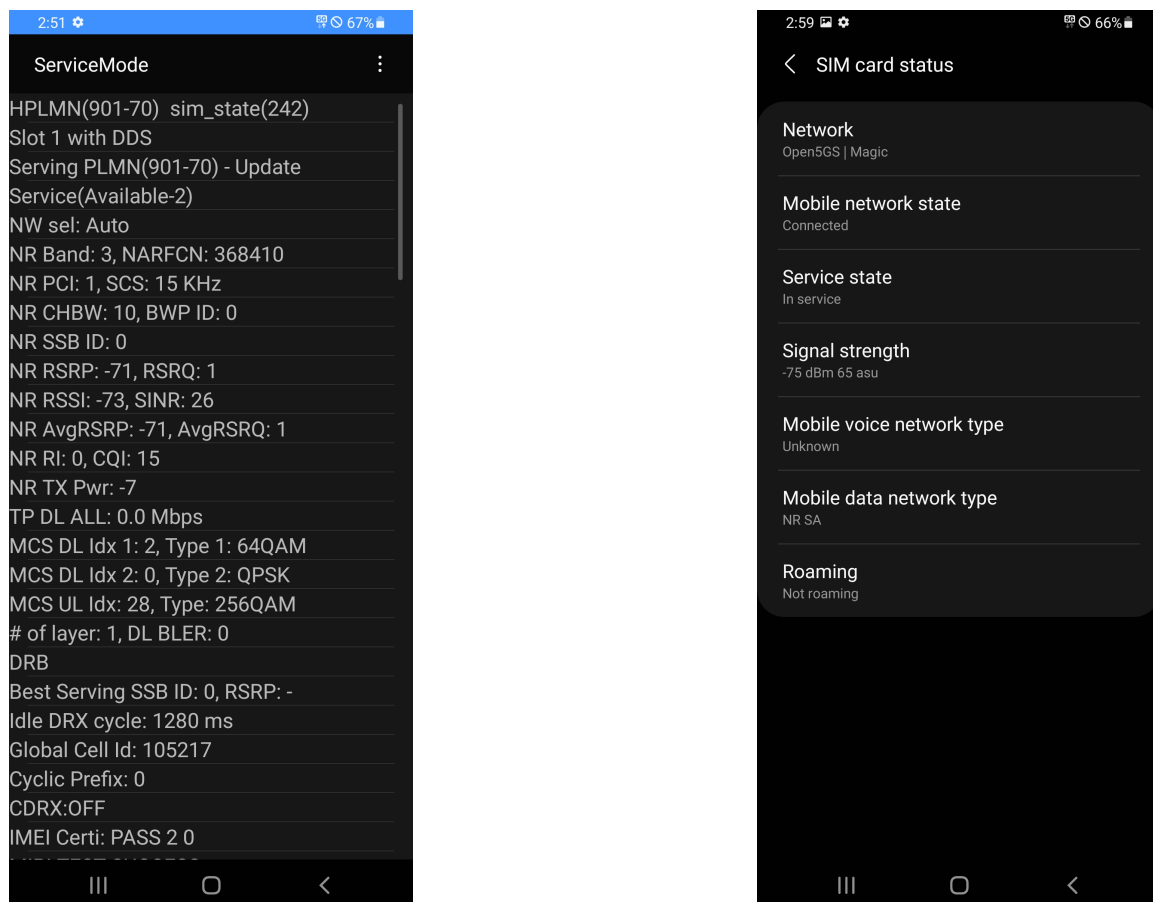


Figure 4.2. Screen captures of Samsung A22 service mode and SIM card report showing successful network connection

We can also use the 5GC log files to further verify successful UE connection to our network. Figure 4.3 shows portions of the AMF log file indicating successful device addition to the

5GC AMF component.

A PDU session was then established, including subsequent IP address assignment to the device by the SMF, as shown in Figure 4.4. Lastly, Figure 4.5 shows the portion of the UPF log file indicating a successful initiation of a UPF for our connected UE.

```
04/07 11:47:25.689: [amf] INFO: [Added] Number of AMF-UEs is now 1 (./src/amf/context.c:1340)
04/07 11:47:25.689: [gmm] INFO: Registration request (./src/amf/gmm-sm.c:135)
04/07 11:47:25.689: [gmm] INFO: [suci-0-901-70-0-0-0-0000051573] SUCI (./src/amf/gmm-handler.c:149)
04/07 11:47:26.131: [gmm] INFO: [imsi-901700000051573] Registration complete (./src/amf/gmm-sm.c:1134)
04/07 11:47:26.131: [amf] INFO: [imsi-901700000051573] Configuration update command (./src/amf/nas-path.c:431)
04/07 11:47:26.131: [gmm] INFO: UTC [2023-04-07T18:47:26] Timezone[0]/DST[0] (./src/amf/gmm-build.c:536)
04/07 11:47:26.131: [gmm] INFO: LOCAL [2023-04-07T11:47:26] Timezone[-25200]/DST[1] (./src/amf/gmm-build.c:541)
04/07 11:47:26.249: [amf] INFO: [Added] Number of AMF-Sessions is now 1 (./src/amf/context.c:2253)
04/07 11:47:26.249: [gmm] INFO: UE SUPI[imsi-901700000051573] DNN[internet] S_NSSAI[SST:1 SD:0xffff] (./src/amf/gmm-handler.c:1062)
```

Figure 4.3. AMF log extract showing successful Samsung A22 network connection

```
10/21 11:23:03.625: [smf] INFO: [Removed] Number of SMF-Sessions is now 0 (./src/smf/context.c:2980)
10/21 11:23:03.625: [smf] INFO: [Removed] Number of SMF-UEs is now 0 (./src/smf/context.c:955)
10/21 11:23:04.025: [smf] INFO: [Added] Number of SMF-UEs is now 1 (./src/smf/context.c:896)
10/21 11:23:04.025: [smf] INFO: [Added] Number of SMF-Sessions is now 1 (./src/smf/context.c:2973)
10/21 11:23:04.028: [smf] INFO: UE SUPI[imsi-901700000051573] DNN[internet] IPv4[10.45.0.3] IPv6[] (./src/smf/npcf-handler.c:495)
```

Figure 4.4. SMF log extract showing successful IP address assignment to the Samsung A22

```
10/21 11:15:43.443: [qtp] INFO: qtp_connect() [127.0.1.1]:2152 (./lib/qtp/path.c:60)
10/21 11:23:03.625: [upf] INFO: [Removed] Number of UPF-sessions is now 0 (./src/upf/context.c:212)
10/21 11:23:04.028: [upf] INFO: [Added] Number of UPF-Sessions is now 1 (./src/upf/context.c:178)
10/21 11:23:04.028: [upf] INFO: UE F-SEID[CP:0x2 UP:0x2] APN[internet] PDN-Type[1] IPv4[10.45.0.3] IPv6[] (./src/upf/context.c:397)
10/21 11:23:04.028: [upf] INFO: UE F-SEID[CP:0x2 UP:0x2] APN[internet] PDN-Type[1] IPv4[10.45.0.3] IPv6[] (./src/upf/context.c:397)
```

Figure 4.5. UPF log extract showing successful UPF session establishment and subsequent IP address assignment to the Samsung A22

Additionally, srsRAN provides a trace of the gNB, which includes several measurements and metrics that indicate network and UE performance from the DL and UL respectively. An example gNB trace from a connected Samsung A22 on the network can be seen in Figure 4.6

```
==== eNodeB started ====
Type <t> to view trace
Setting frequency: DL=2655.0 Mhz, UL=2535.0 Mhz for cc_idx=0 nof_prb=52
RACH: slot=9611, cc=0, preamble=3, offset=0, temp_crnti=0x4601
t
Enter t to stop trace.
```

-----DL-----													-----UL-----												
rat	pci	rn	rl	mcs	brate	ok	nok	(%)	pusch	pucch	phr	mcs	brate	ok	nok	(%)	bsr								
nr	0	4601	7	0	5	3.1M	667	0	0%	20.1	15.2	0	5	343k	67	0	0%	0.0							
nr	0	4601	9	0	5	2.5M	549	0	0%	16.5	16.4	0	5	374k	73	0	0%	0.0							
nr	0	4601	10	0	5	1.8M	399	0	0%	15.5	17.0	0	5	246k	48	0	0%	0.0							
nr	0	4601	8	0	5	2.3M	492	0	0%	15.9	17.0	0	5	261k	51	0	0%	0.0							
nr	0	4601	6	0	5	3.3M	718	0	0%	14.9	17.0	0	5	369k	72	0	0%	0.0							
nr	0	4601	6	0	5	3.9M	840	0	0%	14.0	16.9	0	5	307k	60	0	0%	0.0							
nr	0	4601	6	0	5	3.4M	743	0	0%	14.0	16.7	0	5	379k	74	0	0%	0.0							
nr	0	4601	9	0	5	3.0M	660	0	0%	14.4	16.4	0	5	368k	72	0	0%	0.0							
nr	0	4601	8	0	5	2.9M	640	0	0%	14.0	16.3	0	5	338k	66	0	0%	0.0							
nr	0	4601	9	0	5	2.4M	515	0	0%	13.8	15.9	0	5	266k	52	0	0%	0.0							
nr	0	4601	9	0	5	2.8M	611	0	0%	13.8	15.9	0	5	323k	63	0	0%	0.0							

-----DL-----													-----UL-----												
rat	pci	rn	rl	mcs	brate	ok	nok	(%)	pusch	pucch	phr	mcs	brate	ok	nok	(%)	bsr								
nr	0	4601	9	0	5	2.9M	626	0	0%	13.6	15.8	0	5	271k	53	0	0%	0.0							
nr	0	4601	8	0	5	2.6M	566	0	0%	13.9	15.4	0	5	338k	66	0	0%	0.0							
nr	0	4601	7	0	5	2.5M	541	0	0%	13.4	15.2	0	5	256k	50	0	0%	0.0							
nr	0	4601	7	0	5	3.2M	688	0	0%	12.9	15.4	0	5	629k	123	0	0%	0.0							
nr	0	4601	8	0	5	3.0M	644	0	0%	12.5	15.1	0	5	277k	54	0	0%	0.0							
nr	0	4601	8	0	5	2.8M	605	0	0%	9.3	16.8	0	5	507k	99	0	0%	0.0							
nr	0	4601	8	0	5	3.1M	667	0	0%	15.6	17.1	0	5	369k	72	0	0%	0.0							
nr	0	4601	7	0	5	2.8M	616	0	0%	17.1	17.0	0	5	471k	92	0	0%	0.0							
nr	0	4601	9	0	5	2.7M	591	0	0%	16.3	16.7	0	5	609k	119	0	0%	0.0							
nr	0	4601	7	0	5	3.0M	650	0	0%	16.7	16.6	0	5	333k	65	0	0%	0.0							
nr	0	4601	8	0	5	3.0M	656	0	0%	16.3	16.5	0	5	768k	150	0	0%	0.0							

-----DL-----													-----UL-----												
rat	pci	rn	rl	mcs	brate	ok	nok	(%)	pusch	pucch	phr	mcs	brate	ok	nok	(%)	bsr								
nr	0	4601	9	0	5	2.4M	529	0	0%	15.9	16.2	0	5	379k	74	0	0%	0.0							
nr	0	4601	11	0	5	2.4M	530	0	0%	14.2	16.0	0	5	292k	57	0	0%	0.0							
nr	0	4601	8	0	5	2.8M	613	0	0%	13.5	15.8	0	5	358k	70	0	0%	0.0							
nr	0	4601	10	0	5	2.3M	496	0	0%	13.6	15.5	0	5	338k	66	0	0%	0.0							
nr	0	4601	11	0	5	2.9M	627	0	0%	13.5	15.6	0	5	292k	57	0	0%	0.0							
nr	0	4601	9	0	5	2.8M	613	0	0%	12.7	15.3	0	5	353k	69	0	0%	0.0							
nr	0	4601	11	0	5	2.9M	640	0	0%	10.7	15.1	0	5	323k	63	0	0%	0.0							
nr	0	4601	8	0	5	2.3M	505	0	0%	11.7	14.9	0	5	261k	51	0	0%	0.0							
nr	0	4601	6	0	5	3.3M	709	0	0%	14.2	16.4	0	5	307k	60	0	0%	0.0							
nr	0	4601	10	0	5	2.4M	516	0	0%	14.9	16.9	0	5	425k	83	0	0%	0.0							
nr	0	4601	13	0	5	2.0M	440	0	0%	14.0	16.9	0	5	251k	49	0	0%	0.0							

Figure 4.6. srsRAN gNB trace from a connected Samsung A22

Internet Access

As discussed in Chapter 3, an APN must be added to the UE mobile device settings in order to access the Internet. This APN must exactly match the APN programmed in the 5GC UDR subscriber database for the particular USIM inserted into the mobile device. For the Samsung, you can add or edit current APNs under: *Settings > Connections > Mobile networks > Access Point Names*. We added a new APN and changed the following fields in accordance with our USIM settings in the UDR:

Name - *Internet*

APN - *Internet*

MCC - *901*

MNC - 70

APN Type - *default,Internet*

APN Protocol - *IPv4v6*

After adding the APN and toggling the mobile data connection, we did a simple test to confirm our Internet connection using *speedtest.net*, resulting in Figure 4.7. Of note, at the time, our network backhaul was provided by an AT&T wireless hotspot that supplied Internet to our testing office.

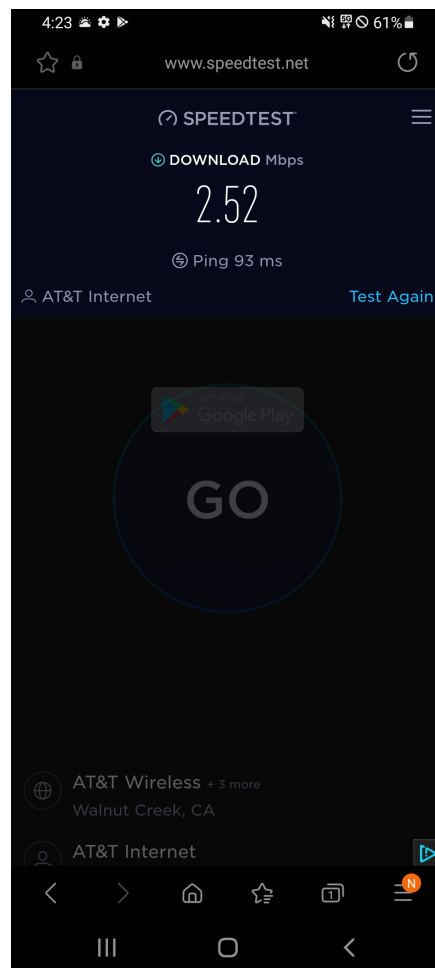


Figure 4.7. Speedtest.net test indicating achieved Internet access on Samsung A22 while connected to our 5G network

4.1.2 Quectel Modem

The Quectel RM500Q-AE modem was used as a white-box connection to our 5G network without any hardware or software restrictions imposed by telecommunications providers or cellular device manufacturers. Out of the box, the modem must be placed and physically integrated into a USB3.0 to M.2 evaluation board (modem adapter) [58]. The evaluation board enables communications of messages between a controller (e.g., CPU) and the modem itself through a serial or USB connection, as well as a slot for SIM card insertion. Figure 4.8 displays the functional diagram of the evaluation board.

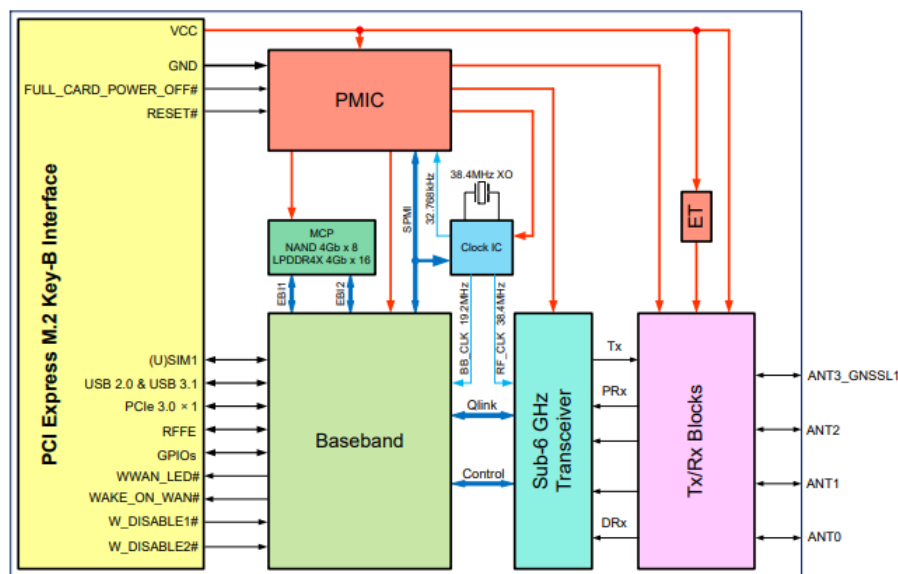


Figure 4.8. Functional diagram of the Quectel modem

In order to access the modem interface, both *Minicom* and *Android Debug Bridge (ADB)* can be used, which are command line tools that provide for serial port communication to a device. Both of these installations were done according to Listing 4.1, which includes some associated dependencies as well.

Listing 4.1: Quectel communication tools setup

Minicom

```
sudo apt update
sudo apt-get install minicom
```

Android Tools

```
sudo apt-get install android-tools-adb
\ android-tools-fastboot
#download dependencies:
sudo apt-get install gcc-arm-linux-gnueabi
\ g++-arm-linux-gnueabi gdb-arm-none-eabi
\ gcc-arm-none-eabi libc6-armel-cross
\ libc6-dev-armel-cross binutils-arm-linux-gnueabi
\ libncurses5-dev binwalk
```

Figure 4.9 shows the physical USB3.0 connection between the controlling laptop and the evaluation board, along with the virtual communication ports. There are four virtual communication ports between the Quectel modem and the host computer, setup in Linux as */dev/ttyUSB0* through */dev/ttyUSB3*, each communicating different types of information as seen in Table 4.1.

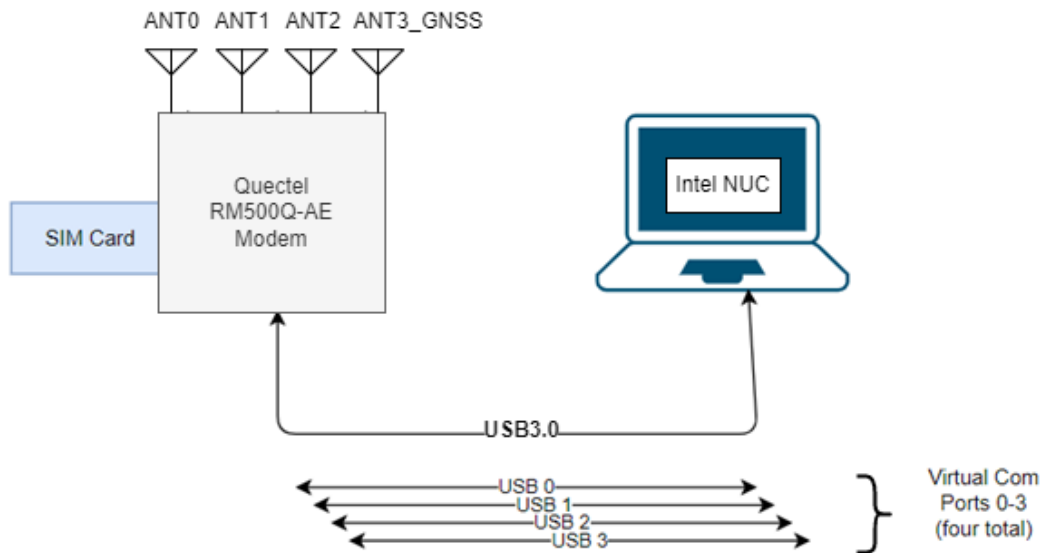


Figure 4.9. Quectel modem physical and virtual USB connections

Table 4.1. Modem virtual communication ports

Linux Port	Communication Type
/dev/ttyUSB0	Qualcomm DIAG data
/dev/ttyUSB1	GNSS location and positioning data
/dev/ttyUSB2	AT modem commands
/dev/ttyUSB3	Alternate for AT modem commands

We are concerned with virtual connection USB2, where AT (attention) commands can be passed to the modem from the host computer. AT commands comprise a set of operations that are standardized for most modems and are also known as the Hayes command set [59]. A full listing of AT commands specific to the Quectel RM500Q-AE modem can be found in [60]. As mentioned, we use *Minicom* as the interface to the

modem from our host computer by calling the program on the command line and using the `s` option for setting up the correct USB connection: `sudo minicom -s`. Next, we must change the serial connection to USB2 through the following sequence of commands: `Serial port setup > A(Serial Device) > "/dev/ttyUSB2"`, as seen in Figure 4.10. Successful connection to the modem through USB2 is shown in Figure 4.11, whereby we are ready to start executing AT commands.

```
+-----+
| A -   Serial Device       : /dev/modem |
| B - Lockfile Location    : /var/lock  |
| C -   Callin Program     :             |
| D -   Callout Program    :             |
| E -   Bps/Par/Bits       : 115200 8N1 |
| F - Hardware Flow Control : Yes       |
| G - Software Flow Control : No       |
|                                     |
|   Change which setting?             |
+-----+
| Screen and keyboard |
| Save setup as dfl   |
| Save setup as..    |
| Exit                |
| Exit from Minicom   |
+-----+
```

Figure 4.10. Minicom interface used for changing serial device to USB2

```
Welcome to minicom 2.7.1

OPTIONS: I18n
Compiled on Dec 23 2019, 02:06:26.
Port /dev/ttyUSB2, 12:00:16

Press CTRL-A Z for help on special keys
```

Figure 4.11. Successful connection to modem through Minicom

Once you are connected to the modem, you can run AT commands to retrieve responses from

the modem, configure settings, register to networks, and so on. When first connected, you should run AT+COPS=? to query for all available networks that the modem can receive and decode the SS Block and MIB. Then, you can run AT+QSCAN=2 to get network information specific to 5G. Ideally, if the HPLMN is set up correctly during the USIM programming, the modem should automatically attempt to connect to the network due to its presence in the preferred operator listing. Otherwise, you can use the AT+COPS= command to manually register to the network. You can also set the preferred network for subsequent automatic registration with: AT+CPOL=1,2,"90170". Table 4.2 displays a series of other useful AT commands used for 5G modem operations. You will also notice that each command begins with AT+ indicating that it is an attention command (AT) with an extension (+). The latest manual for the modem utilized in this thesis can be found in [60].

Table 4.2. Quectel modem AT commands

AT Command	Description
AT+CIMI	IMSI lookup
AT+COPS	Operator detection and selection
AT+C5GREG	5GS network registration status
AT+CSQ	Signal quality report
AT+QSRP	Reference Signal Received Power (RSRP) report
AT+CPOL	Preferred operator list
AT+COPN	Listing of known operators
AT+QNWINFO	Query currently connected network information
AT+QSCAN	Quick scan of nearby network cells
AT+QNWCFG	Configure and query network parameters
AT+QNWPRECFG	Configure network searching preferences

Connection Success

The modem was able to successfully register to our network, shown in Figure 4.12.

```

Welcome to minicom 2.7.1

OPTIONS: I18n
Compiled on Dec 23 2019, 02:06:26.
Port /dev/ttyUSB2, 10:11:05

Press CTRL-A Z for help on special keys

AT
OK
AT+CPOL?
+CPOL: 1,2,"90170",1,1,1,1,1

OK
AT+QNWINFO
+QNWINFO: "FDD NR5G","90170","NR5G BAND 3",368410

OK
AT+QSRP
+QSRP: -91,-77,-73,-82,NR5G

OK
AT+QENG="servingcell"
+QENG: "servingcell","NOCONN","NR5G-SA","FDD", 901,70,19B01,1,1,368410,3,1,-77,-11,28,0,-

```

Figure 4.12. Quectel modem network detection and connection of our 5G network

Internet Access

Much like the Samsung, the APN on the Quectel modem must also be configured to enable Internet connectivity. The following series of steps in Table 4.3 executes the commands required to add an APN.

Table 4.3. Quectel modem commands to add an APN

AT Command	Description
AT+CGATT=1	Activates packet domain service for PDU session establishment
AT+CGDCONT?	Queries list of APNs
AT+CGDCONT=1,"IP","Internet"	Add an APN in context 1, setup for IPv4 ("IP"), and name it "Internet"
AT+CGACT=1,1	Activate the created APN and reattach to network if disconnected
AT+CGPADDR=1	Used to show the IP address assigned for PDU session

More information about each of these commands and all the options for configuration can

be found in the Quectel AT Command Manual [60]. Figure 4.13 shows the establishment of a PDU session, with an IP address assigned to the Quectel modem. Of note, the other two APNs of IMS and SOS are default on the modem.

```
AT+CGATT=1
OK
AT+CGDCONT?
+CGDCONT: 1,"IP","internet","0.0.0.0",0,0,0,0,,,,,,,,,"",,,,0
+CGDCONT: 2,"IPV4V6","ims","0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0",0
+CGDCONT: 3,"IPV4V6","sos","0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0",0

OK
AT+CGACT=1,1
OK
AT+CGPADDR=1
+CGPADDR: 1,"10.45.0.2"
```

Figure 4.13. Quectel PDU session establishment

4.1.3 Unknown UE Registration Denial

As a result of the mutual authentication process within 5G, devices that are not registered with the network will not be able to connect. We tested this concept by attempting to register a UE containing a SIM with an ISMI that was not allocated in our UDR. Figure 4.14 shows the log entries when an unknown UE is rejected from connecting to the network. The UE's SUCI and associated SUPI are not part of the UDM, which leads to immediate registration rejection. Authentication does not even take place.

```
10/28 11:00:43.610: [amf] INFO: InitialUEMessage (./src/amf/ngap-handler.c:361)
10/28 11:00:43.610: [amf] INFO: [Added] Number of qNB-UEs is now 1 (./src/amf/context.c:2132)
10/28 11:00:43.610: [amf] INFO: Unknown UE by 5G-S TMSI[AMF ID:0x20000,M TMSI:0xcd05b492] (./src/amf/ngap-handler.c:431)
10/28 11:00:43.610: [amf] INFO: RAN UE NGAP ID[1] AMF UE NGAP ID[1] TAC[1] CellID[0x66c001] (./src/amf/ngap-handler.c:497)
10/28 11:00:43.610: [amf] INFO: [suci-0-901-70-0-0-0-0000051570] Unknown UE by SUCI (./src/amf/context.c:1443)
10/28 11:00:43.610: [amf] INFO: [Added] Number of AMF-UEs is now 1 (./src/amf/context.c:1239)
10/28 11:00:43.610: [qmm] INFO: Registration request (./src/amf/qmm-sm.c:135)
10/28 11:00:43.610: [qmm] INFO: [suci-0-901-70-0-0-0-0000051570] SUCI (./src/amf/qmm-handler.c:149)
10/28 11:00:43.610: [sbi] WARNING: Try to discover [nausf-auth] (./lib/sbi/path.c:114)
10/28 11:00:43.610: [sbi] INFO: [68320006-56e7-41ed-957a-7b2605191cb7] (NF-discover) NF registered (./lib/sbi/nrnf-handler.c:731)
10/28 11:00:43.610: [sbi] INFO: [68320006-56e7-41ed-957a-7b2605191cb7] (NF-discover) NF Profile updated (./lib/sbi/nrnf-handler.c:758)
10/28 11:00:43.611: [qmm] WARNING: [suci-0-901-70-0-0-0-0000051570] Cannot find SUCI [404] (./src/amf/qmm-sm.c:701)
10/28 11:00:43.611: [amf] WARNING: [suci-0-901-70-0-0-0-0000051570] Registration reject [11] (./src/amf/nas-path.c:163)
10/28 11:00:43.612: [amf] INFO: UE Context Release [Action:3] (./src/amf/ngap-handler.c:1406)
10/28 11:00:43.612: [amf] INFO: RAN UE NGAP ID[1] AMF UE NGAP ID[1] (./src/amf/ngap-handler.c:1407)
10/28 11:00:43.612: [amf] INFO: SUCI[suci-0-901-70-0-0-0-0000051570] (./src/amf/ngap-handler.c:1410)
10/28 11:00:43.612: [amf] INFO: [Removed] Number of qNB-UEs is now 0 (./src/amf/context.c:2139)
10/28 11:00:43.612: [amf] INFO: [Removed] Number of AMF-UEs is now 0 (./src/amf/context.c:1323)
```

Figure 4.14. Registration rejection for unknown user

4.2 Optimization Testing

This section illustrates how we optimized our network through signal quality testing in conjunction with manipulation of different gNB and SDR settings. Configuring the proper RF environment for the network proved crucial in obtaining cell search, registration, and subsequent network connection. The most important parameters or aspects included modulation and coding scheme (MCS), transmit and receive gain, and selection of antenna hardware. We decided to use Band 3 of FR1 for all of our testing due to widespread adoption of most UEs and consistent stability of connection for our Quectel Modem and Samsung A22 devices. Optimization of network settings needs to be considered for any use of a network like ours when employing operationally. RF environments can be vastly different for dissimilar locations, so testing should be conducted before operational use.

Figure 4.15 illustrates our physical testing setup, showing the mounted directional antennas on a mobility cart, along with display, SDR, and NUC controller. The channel used for proximity testing is also shown in Figure 4.16.



Figure 4.15. Physical testing setup showing the mounted directional antennas on a mobility cart, along with display, SDR, and NUC controller



Figure 4.16. Channel used for proximity testing

4.2.1 MCS Testing

MCS, which defines the quantity of informative bits that can be carried by one symbol or resource element, is an important setting that needs to be finely tuned to achieve a stable UE connection. The MCS setting is a 5-bit field ranging from 0 to 28 with an additional three values reserved for re-transmission due to errors in information within the PDSCH or PUSCH [61]. Each of the value schemes represent a particular combination of modulation

scheme and channel-coding rate. The combination can then be measured as a spectral efficiency, which equates to the amount of information bits per modulation symbol [22].

5G supports QPSK, 16-QAM, 64-QAM, and 256-QAM modulation schemes, indicated as the Modulation order (Q_m) parameter by 3GPP according to Table 4.4. These (Q_m) values correlate to certain MCS values according to Figure 4.17.

Table 4.4. 3GPP modulation order

(Q_m)	Modulation method
2	QPSK
4	16-QAM
6	64-QAM
8	256-QAM

MCS Index <i>l_{MCS}</i>	Modulation Order <i>Q_m</i>	Target code Rate <i>R</i> x [1024]	Spectral efficiency
0	2	120	0.2344
1	2	157	0.3066
2	2	193	0.3770
3	2	251	0.4902
4	2	308	0.6016
5	2	379	0.7402
6	2	449	0.8770
7	2	526	1.0273
8	2	602	1.1758
9	2	679	1.3262
10	4	340	1.3281
11	4	378	1.4766
12	4	434	1.6953
13	4	490	1.9141
14	4	553	2.1602
15	4	616	2.4063
16	4	658	2.5703
17	6	438	2.5664
18	6	466	2.7305
19	6	517	3.0293
20	6	567	3.3223
21	6	616	3.6094
22	6	666	3.9023
23	6	719	4.2129
24	6	772	4.5234
25	6	822	4.8164
26	6	873	5.1152
27	6	910	5.3320
28	6	948	5.5547
29	2	reserved	
30	4	reserved	
31	6	reserved	

Figure 4.17. MCS modulation order. Source [61].

Logically, a larger MCS results from a better link quality and consequently permits a higher number of useful bits to be transmitted. For our network, srsRAN defaults to using a maximum MCS value of 28 with no ability for the radio resource scheduler to dynamically adapt to live RF conditions [50]. In typical 5G networks, MCS is a dynamic value determined from the UE's channel quality indicator (CQI), so that the gNB can vary the MCS depending on the quality of signal with respect to UE. Lack of dynamic MCS adjustment in the srsRAN software consequently provided the motivation behind our MCS testing. Prior to changing the default setting from 28 to that of lower values, we had difficulty with stable UE connection to the network, which lead to our MCS investigation.

We conducted testing by initializing our network with different MCS values while leaving all other network settings the same according to Table 4.5. We used a Samsung A22 placed one meter away from our network, starting the test by turning on the device and leaving for 60 seconds. We administered each test five separate times to provide some variability of the RF environment, resulting in 300 data points as the UE reports its network signal status once every second in service mode. We then recorded signal-to-noise ratio (SNR) values and the number of dropped packets from the UE to the network indicated by the gNB network trace on the NUC.

Table 4.5. MCS testing network settings

Band	ARFCN	DL (MHz)	UL (MHz)	PLMN	Transmit Gain	gNB Receive Gain
3	368500	1842.5	1747.5	90170	79	AGC

Table 4.6 show our MCS testing results, indicating the median SNR and number of dropped packets at each MCS level setting. For our equipment and positioning, MCS of 10 significantly outperformed the other values and packets start dropping at an MCS of greater than 20. Additionally, an SNR value of zero indicating a dropped signal between the gNB and UE occurred in every test except for 10, giving us reason to use the median instead of the mean to depict our results. Additionally, UEs had notable difficulty even connecting to the network at all with MCS values greater than 10. It is acknowledged that these results are not as expected, especially given the testing proximity of only 1m. Even with COTS hardware and open source software, we would expect to be able to take full advantage of 5G high

data rates and have similar SNR values with minimal dropped packets for the highest MCS values.

Table 4.6. Comparison of MCS values vs received SNR at UE at 1m

MCS	SNR Median (dB)	Dropped Packets
1	15.7	0
5	12.7	0
10	24.6	0
15	17.4	0
20	14.6	18
25	14.5	143
28	14.3	200

4.2.2 Transmit Gain Testing

gNB transmit gain substantially affects signal quality and facilitates UE capability to search for and connect to the network. IEEE defines gain as the radiation efficiency in combination with the directionality. In particular, transmit gain describes how well the antenna transforms the input power into the form of radio waves headed in a certain direction [62]. Conversely, receive gain describes how well the receiving antenna transforms the arriving radio wave from a certain direction into power. Another important aspect of receive gain is a concept called automatic gain control (AGC), which automatically adjusts the gain in the receiver based on signal strength. AGC primarily attenuates strong signals and amplifies weak signals in order to produce received signals within an optimal SNR. Power gain is usually measured and expressed as a logarithmic dB and for every increase of 3 dB, the overall power output is doubled [63]. For our system, AGC functionality was available and thus used for our receive gain settings, allowing the gNB to automatically tune its settings according to the incoming messages from the UE.

We conducted testing by initializing the gNB with different transmit gain values while leaving all other network settings the same according to Table 4.7. We used a Samsung A22 placed one meter away from our network, starting the test by turning on the device

and leaving for 60 seconds. We conducted each test five separate times to provide some variability of the RF environment, resulting in 300 data points as the UE reports its network signal status once every second in service mode. We then recorded SNR values and the number of dropped packets from the UE to the network indicated by the gNB network trace on the NUC.

Table 4.7. Transmit gain testing network settings

Band	ARFCN	DL (MHz)	UL (MHz)	PLMN	MCS	gNB Receive Gain
3	368500	1842.5	1747.5	90170	10	AGC

We found that using a transmit gain below 39 dB did not allow for the UE to discover the gNB. Table 4.8 and boxplots in Figure 4.18 show our transmit gain testing results in terms of average and standard deviation (SD) of the SNR.

Table 4.8. Comparison of gNB transmit gain vs received SNR at UE at 1 m

Transmit Gain	SNR Mean (dB)	SNR SD (dB)
39	5.510	3.271
49	7.415	3.525
59	26.741	1.078
69	23.098	0.662
79	24.612	0.787

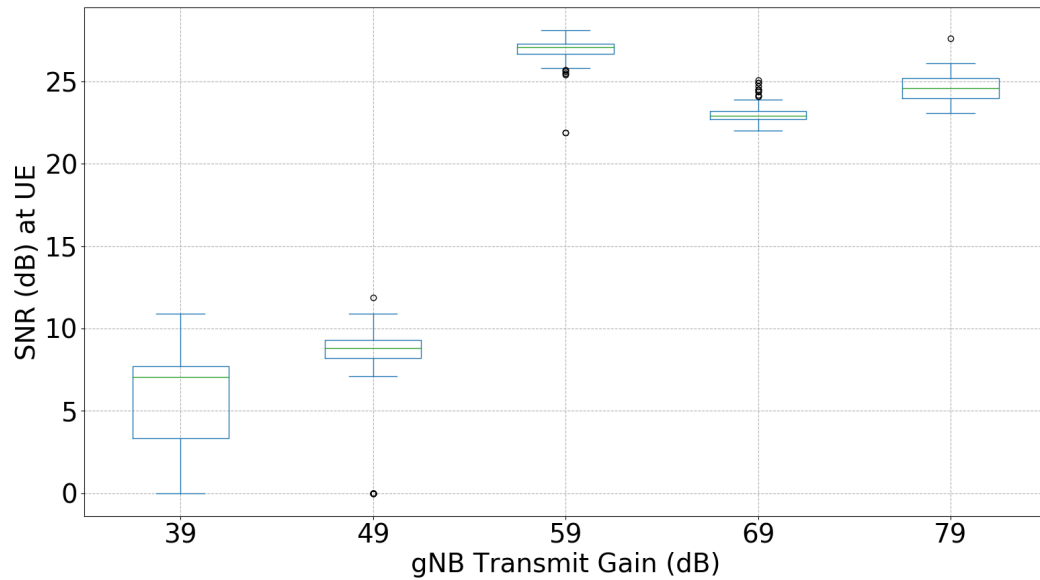


Figure 4.18. Box plot comparison of gNB transmit gain vs received SNR at UE at 1 m

It is visually apparent that transmit gain settings of 39 dB and 49 dB produced much worse results, so we scoped our results to only include 59 dB, 69 dB, and 79 dB. The updated results are shown by boxplots in Figure 4.19.

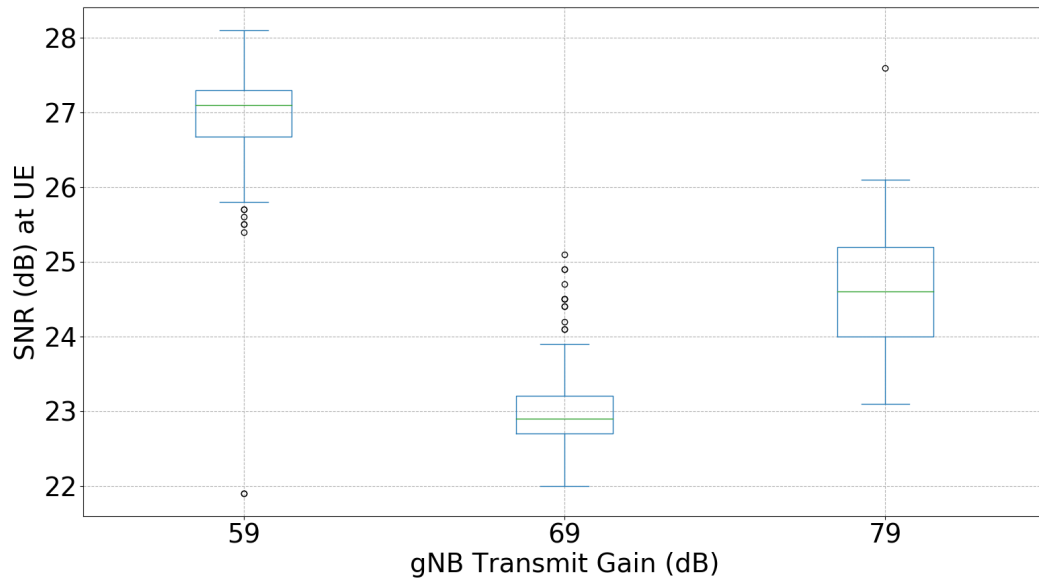


Figure 4.19. Box plot comparison of 59 dB, 69 dB, 79 dB gNB transmit gain vs received SNR at UE at 1 m

Once you remove the poor performing gain settings of 39 dB and 49 dB, it is more clear to see that 59 dB transmit gain performed best at the short distance of 1 m. However, we expected a more linear increasing result from 59 dB to 79 dB of gain, but believe that there was an inconsistency introduced by excess noise from the gNB transmission at such close proximity.

We then performed a one-way ANOVA test to verify that there was a statistically significant difference between transmit gain settings of 59 dB, 69 dB, and 79 dB. This resulted in a p-value of $2.229e-79$, which is $< .05$, and thus, rejected the null hypothesis that the mean SNR value is equal across all three transmit gain settings. As a result, a subsequent pairwise Tukey test was performed with the results shown in Figure 4.20, deducing that a transmit gain setting of 59 dB achieved statistically significant results in comparison to 69 dB and 79 dB. Additionally, 79 dB performed statistically better than 69 dB. However, it is important to reiterate that our testing was conducted with the UE placed at 1 m, so excess gain can contribute to additional noise and thus a lower SNR performance. It can

also present discrepancies in results thus confirming that gain settings should be optimized for varying use cases.

```

Multiple Comparison of Means - Tukey HSD, FWER=0.05
=====
group1 group2 meandiff p-adj lower upper reject
-----
Gain_59 Gain_69 -3.6427 0.001 -3.955 -3.3303 True
Gain_59 Gain_79 -2.1283 0.001 -2.4319 -1.8248 True
Gain_69 Gain_79 1.5143 0.001 1.2466 1.7821 True
-----

```

Figure 4.20. SNR Tukey testing results of transmit gain at 1 m

In consideration of our gain testing results using a trivial UE distance of one meter, we repeated the same transmit gain test at a distance of 10 m to highlight our assumption that a higher gain will yield a better signal connection result with a greater distance. Table 4.9 and Figure 4.21 report our results, showing that, as expected, using the largest transmit gain of 79 dB facilitates the best network connection.

We again performed a one-way ANOVA test to determine if there was a statistically significant difference between transmit gain settings of 59 dB, 69 dB, and 79 dB at 10 m. This resulted in a p-value of 6.285e-26, which is $< .05$, and thus, rejected the null hypothesis that the mean SNR value is equal across all three transmit gain settings. As a result, a subsequent pairwise Tukey test was performed with the results shown in Figure 4.22, indicating a statistically significant increase in performance from 59 dB to 79 dB.

Table 4.9. Comparison of 59 dB, 69 dB, 79 dB gNB transmit gain vs received SNR at UE at 10 m

Transmit Gain	SNR Mean (dB)	SNR SD (dB)
59	12.355	1.404
69	19.065	2.850
79	22.903	3.402

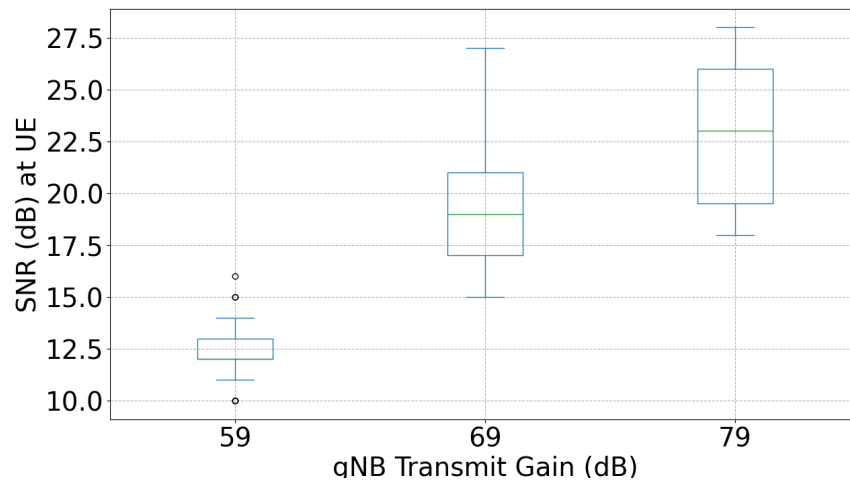


Figure 4.21. Box plot comparison of 59 dB, 69 dB, 79 dB gNB transmit gain vs received SNR at UE at 10m

```

Multiple Comparison of Means - Tukey HSD, FWER=0.05
=====
group1 group2 meandiff p-adj lower upper reject
-----
Gain_59 Gain_69 6.7097 0.0 5.0561 8.3633 True
Gain_59 Gain_79 10.5484 0.0 8.8948 12.202 True
Gain_69 Gain_79 3.8387 0.0 2.1851 5.4923 True
=====

```

Figure 4.22. Tukey testing results comparing 59 dB, 69 dB, 79 dB gNB transmit gain vs received SNR at UE at 10 m of transmit gain at 10 m

4.2.3 Antenna and Proximity Testing

Our next set of testing involved using different antennae to evaluate antenna effect on received SNR and associated performance with increased distance between the gNB and UE. We conducted our testing by using the Samsung A22 placed at distance intervals of 1, 10, 20, 30, 40, 50, and 100 meters while iterating through three different antennae and keeping all other network settings consistent according to Table 4.10.

Table 4.10. Antenna and proximity network testing settings

Band	ARFCN	DL (MHz)	UL (MHz)	PLMN	MCS	Transmit Gain	gNB Receive Gain
3	368500	1842.5	1747.5	90170	10	79	AGC

We used two different omni-directional and one directional antenna for these tests, with specifications according to the list below:

1. **Lcom omni-directional** - 3 dBi gain magnetic base with 10 foot high performance low-loss cable. Frequency coverage for 700 MHz, 850 MHz, AWS and PCS bands [64]
2. **Amarisoft omni-directional** - 500 MHz to 6 GHz RF coverage
3. **Egasky directional** - Ranges of 698-960 MHz with 8.5 dBi gain, 1710-2675 MHz and 3300-3700 MHz with 9 dBi gain

Of note, for all other tests we used the Amarisoft omni-directional antenna unless otherwise indicated.

We administered the test in two different environments: one inside down a corridor hallway, and the other outside with no obstructions between the gNB and UE. We conducted each test for 60 seconds at every distance and then repeated five times and recorded the mean Reference Signal Received Power (RSRP) as indicated by the Samsung A22 in service mode. RSRP is the average power level of the received signal from the gNB [61]. We decided to use the RSRP due to large standard deviation results from using the UE reported SNR value as well as inconsistent results when investigating the median.

For 3GPP 5G networks, reported RSRP values range from -44 dBm to -144 dBm (the closer to 0, the better the signal strength) [61]. According to Quectel modem guides and other commercial cellular providers, any RSRP value better than -90 dBm is generally considered a strong signal.

Testing results for both tests can be seen in Tables 4.11 and 4.12 with associated Figures 4.23 and 4.24. Each of the antennas facilitated UE connection up to our testing threshold of 100 m for both indoor and outdoor. We saw a slightly better performance indoors, especially for the omni-directional antennas, most likely due to signal multi-path in the concrete hallway

environment. As expected, the directional antenna performed significantly better than either of the omni-directional antennas. However, as distance increases, the margin for error on receiving the signal from a directional antenna decreases thus making it more difficult to maintain exact line of sight necessary for transmission and receipt of signal. Another takeaway is that we only conducted the test by moving the UE in a straight line directly backwards from the direction of the directional antenna. For operational use, this setup would not be acceptable for any mobility associated with the end user.

Table 4.11. Indoor testing comparison of distance vs RSRP at UE

Distance(m)	Lcom RSRP (dBm)	Amarisoft RSRP (dBm)	Directional RSRP (dBm)
1	-63	-60	-59
10	-78	-78	-70
20	-81	-80	-77
30	-83	-85	-78
40	-87	-88	-80
50	-97	-94	-82
100	-102	-97	-96

Table 4.12. Outdoor testing comparison of distance vs RSRP at UE

Distance(m)	Lcom RSRP (dBm)	Amarisoft RSRP (dBm)	Directional RSRP (dBm)
1	-64	-62	-58
10	-88	-85	-78
20	-94	-91	-79
30	-96	-93	-80
40	-98	-94	-85
50	-99	-97	-89
100	-103	-99	-95

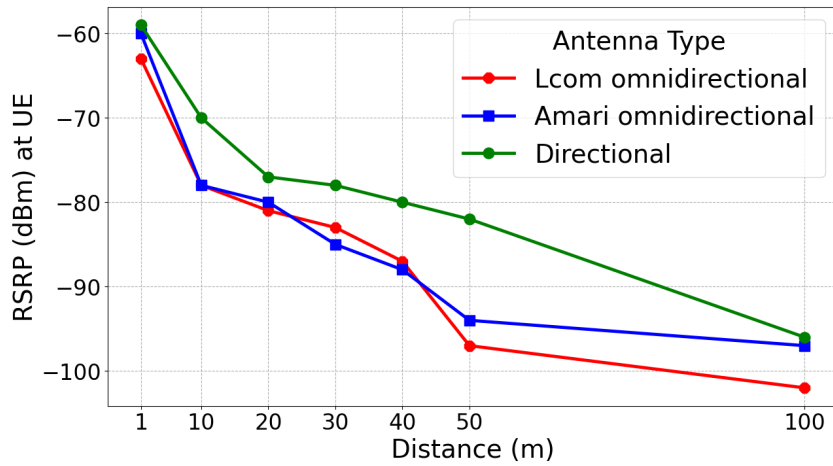


Figure 4.23. Indoor testing comparison of distance vs RSRP at UE

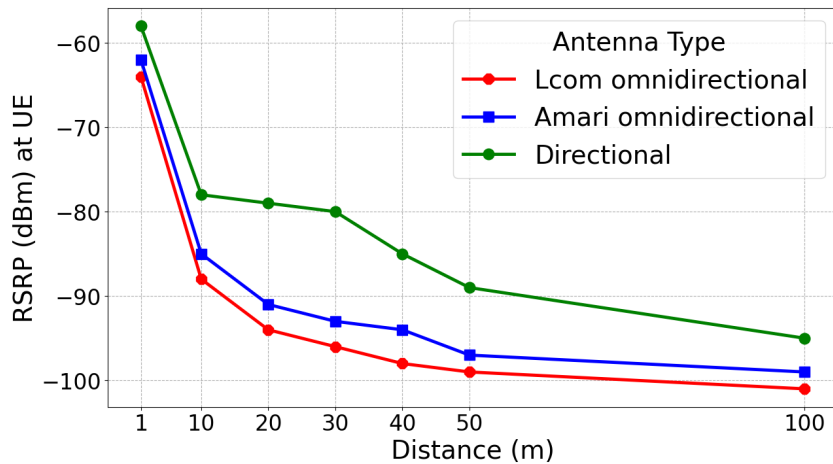


Figure 4.24. Outdoor testing comparison of distance vs RSRP at UE

Particular to our outdoor setup, since we were able to experiment with greater distances between gNB and UE, we conducted a separate test to discover maximum operating ranges for UEs on the network for each of the different antennas. We started by connecting the Samsung A22 to our network in proximity and then walking in a straight line directly away from our network setup until losing connection. Connection loss was indicated by

both a disconnection from the network on the UE as well as a UE disconnection message on the gNB trace. We then walked back toward the gNB until we were able to regain a consistent connection for at least 30 seconds. We repeated the test ten times and extracted the average. Results of our outdoor maximum range test are provided in Table 4.13. Again, not surprisingly, the directional antenna significantly outperformed each of the omni directional.

Table 4.13. Testing results of outdoor maximum operating distance

Antenna	Maximum Distance (m)
Amari	198
Lcom	206
Directional	366

4.3 Throughput Testing

Part of outlining the capabilities of any communications network involves understanding the achieved throughput (TP) for end users operating on the network. We tested and extracted both the download and upload speeds of our 5G network stack using a similar setup and methodology as [8]. However, in this case, we tailored the experiment to investigate speeds associated with downloading and uploading files on the Internet to measure our network TP. Our TP testing setup is shown in Figure 4.25, similar to our overall network design in Figure 3.1, but with particular illustration of the Quectel modem utilization.

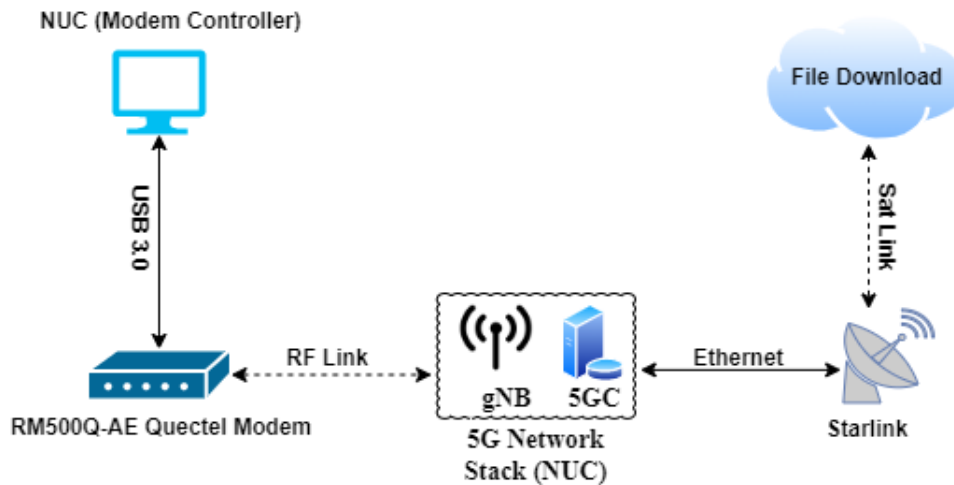


Figure 4.25. Network setup for throughput testing. Adapted from [8].

For our TP testing, the Quectel RM500Q-AE modem served as the UE connected to our network. With root shell access to the modem, we were able to use Android Debug Bridge (ADB) for communication to transfer the *curl* and *Speedtest* statically compiled binaries to use as command line tools in our testing process [8]. With respect to the methodology in [8], we were able to optimize the testing process by solely using the *curl* command, which is a Linux-based command line tool used to transfer data from or to a server [65]. The *time_total* option within *curl* extracts the total time of the *curl* operation in seconds, with precision down to the microsecond. Consequently, there was no need to use *tshark* to start and stop captures for extraction of the download times for each test. Once connected, the modem was used to access and download files from *www.testfiledownload.com*, which hosts zipped files of various sizes that we used to subsequently extract download speeds based on time to complete. In addition to *curl*, we were able to conduct TP testing using *Speedtest*, which is a command line tool to test Internet bandwidth by downloading and uploading data to *speedtest.net* servers.

4.3.1 Modem Configuration

We used ADB to send and retrieve data between the modem and NUC, however some additional configuration apart from the overall modem setup conducted in Chapter 3 was necessary. First, we needed to get root shell access to the modem in order to change the USB

settings. Root access is obtained by inputting the ADBKEY password for the associated ID of the modem. You can retrieve the modem ID by running `AT+QADBKEY?`. Then, assuming you have the corresponding ADBKEY password for that ID provided by Quectel, you can input that key with `AT+ADBKEY=<Quectel-provided ADB password>`. Once logged in with root shell access, the USB configuration settings need to be modified to enable ADB connection on the USB port: `AT+QCFG:"usbcfg",<vendorID><productID><diag><nmea><at-port><modem><rmnet><adb><uac>`. The vendor and product ID can be found by running `lsusb` on Linux to retrieve information on plugged in devices. So, the exact command we ran to enable USB debugging for ADB was:

`AT+QCFG="usbcfg",0x2c7c,0x0800,1,1,1,1,1,1,0`, where the "1s" enable that particular feature.

In addition to the changes on the modem itself, a udev file, which is used by Linux for device events, must be added to the controlling device (NUC) with the corresponding modem information. The udev file contains a USB configuration for each type of device you want to use for development. On Linux devices, the udev file for Android devices connected through USB is located at `/etc/udev/rules.d/51-android.rules`. Additional information about inputting the specific device information into that udev file can be found at [66]. Once the file is added and the udev service is restarted, you should see the modem listed as a connected ADB device when running the `adb devices` command.

4.3.2 Testing Process

We used both the *curl* and *speedtest* command line tools to test the TP of our network as each had static binaries available for the ARMv7 32-bit architecture of the Quectel modem. The *curl* command was used to initiate a file download from an online server to the modem device. We used the *www.testfiledownload* site, which hosts various sizes of dummy files containing random data, ranging from 100KB to 10GB, to test and assess our download speed. We conducted three tests, requesting file sizes of 1MB, 5MB, and 50MB to record download time through our 5G connection. The Quectel modem was placed one meter from our gNB during this testing process to simulate ideal conditions and provide a maximum TP result. Each of the tests were run a set number of times for each file size and repeated twice to account for variability in network performance. We then calculated the average download speed from the total time of each *curl* event.

Speedtest works by canvassing its 15,000+ global testing servers to find the optimum download server based on fastest ping time, thus providing the maximal potential of the TP speed calculation [67]. After identifying an optimal server, *Speedtest* functions much like our *curl* experiment in that it uses a TCP connection to download a file from that Speedtest server. We leveraged some of the available options within *Speedtest*, most notably the *-simple* option, which only uses a single TCP connection instead of multiple and more closely resembles a typical file download or transfer. We performed two series of tests using *Speedtest* to extract both the upload and download speeds, one using the *Speedtest* algorithm-selected optimized server and another in which we sampled all available servers for each test. Each of the two tests were also run a set number of times to account for variability in network performance.

4.3.3 Results

Our numerical results for download speeds of both *curl* and *Speedtest* tests are displayed in Table 4.14 and graphically depicted as box plots and a combined distribution function (CDF) in Figure 4.26 and Figure 4.27 respectively. Of note, *Speedtest* calculates the download rate by retrieving the maximum instantaneous rate at any point in time over the download process, whereas our speed calculations derived from *curl* indicate the average download speed over the entire downloading process including the setup, download, and tear down of the TCP connection.

Table 4.14. Download speeds using curl and speedtest through SatNet back-haul

Testing Type	Mean (Mbps)	SD (Mbps)	Median (Mbps)	Maximum (Mbps)	Minimum (Mbps)
Curl 1 MB File Download	2.22	0.56	2.24	3.50	1.07
Curl 5 MB File Download	6.49	5.03	4.31	19.89	1.50
Curl 50 MB File Download	9.45	5.39	7.21	26.96	2.55
Speedtest Optimum Server	14.89	6.54	13.32	35.22	5.03
Speedtest Sampled Servers	14.97	7.18	13.59	44.36	2.68

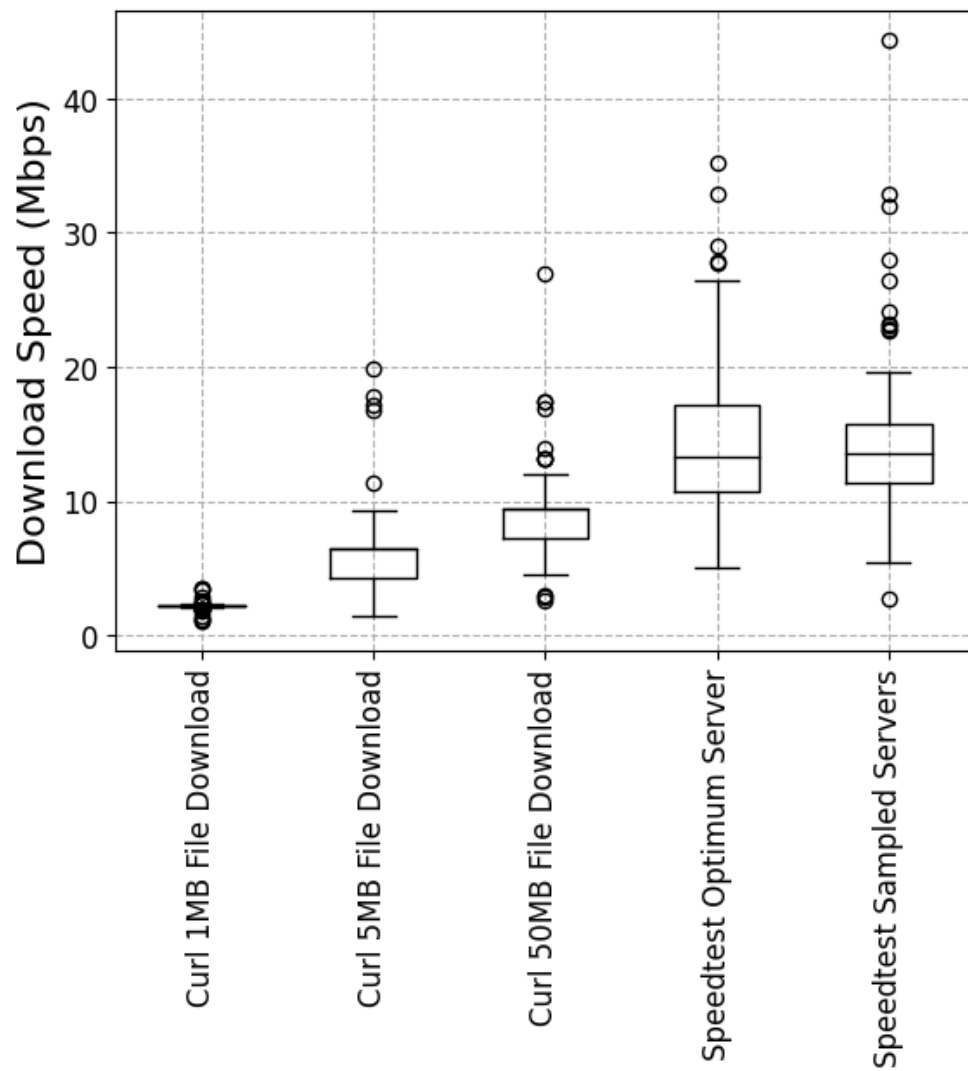


Figure 4.26. Box plot comparison of download speeds using curl and speedtest through SatNet backhaul

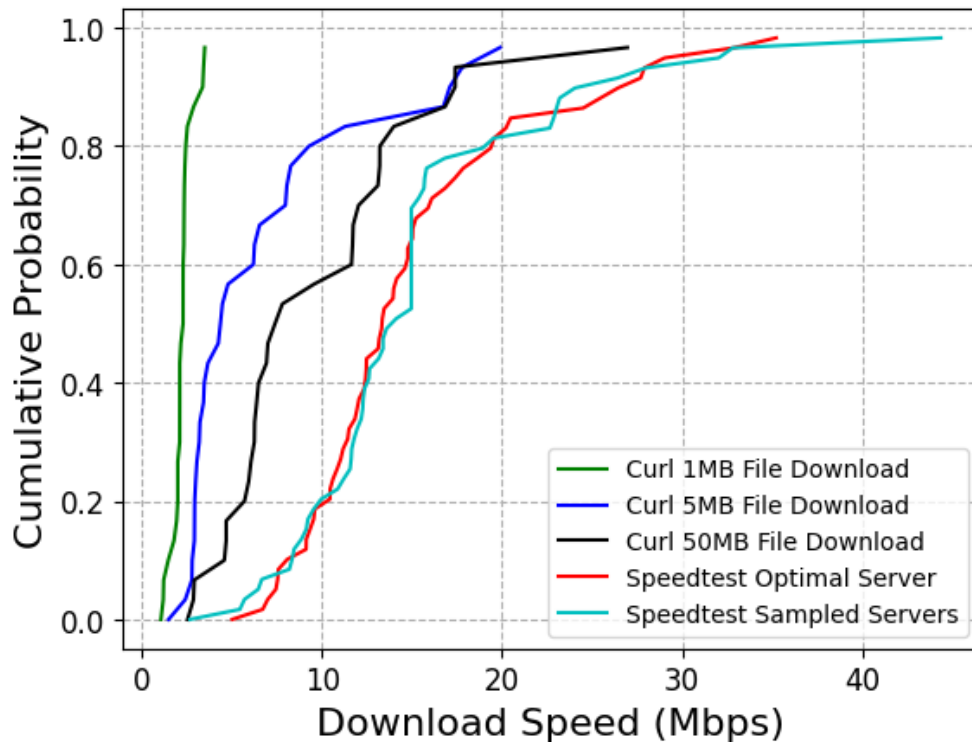


Figure 4.27. CDF comparison of download speeds using curl and speedtest through SatNet backhaul

For the largest file download size in the *curl* test, our network saw an average download rate of 9.45 Mbps and reached peaked speeds close to 27 Mbps. Also, our download results seemingly improved for each increase in file size most likely due to operation of the TCP protocol. TCP slow-start congestion control occurs for each of the downloads, meaning that the larger file size and thus longer download time meant that the TCP flows could operate longer at their maximum sustainable TP speeds. Also, ramp up speed is directly correlated with latency, which is slower for SatNet connections, but fairly reasonable given the LEO orbit of Starlink.

For our *Speedtest* results, we saw an average of almost 15 Mbps for both the optimal and sampled servers. Of note, the *Speedtest* command line tool only allows for usage of the top 10 local servers for connection testing, giving reason for why the optimal and sampled server results are so close in performance. Still, the sampled server tests provide a more

accurate depiction of TP speeds considering an end user would be accessing numerous different sites and server locations on the Internet. Also, lowest latency and ping time do not always correlate to the fastest download speeds, as seen by the observed higher maximum download rate of 45 Mbps with the sampled servers compared to only 35 Mbps with the optimal server. Our latency results for both *Speedtest* tests are shown in Table 4.15 and Figure 4.28, indicating a better performance from the optimal server, as expected.

Table 4.15. Latency results from speedtest through SatNet backhaul

Server	Mean (ms)	SD (ms)	Median (ms)	Maximum (ms)	Minimum (ms)
Optimal	64.80	21.99	62.15	177.72	31.98
Sampled	73.03	32.95	66.24	223.48	34.12

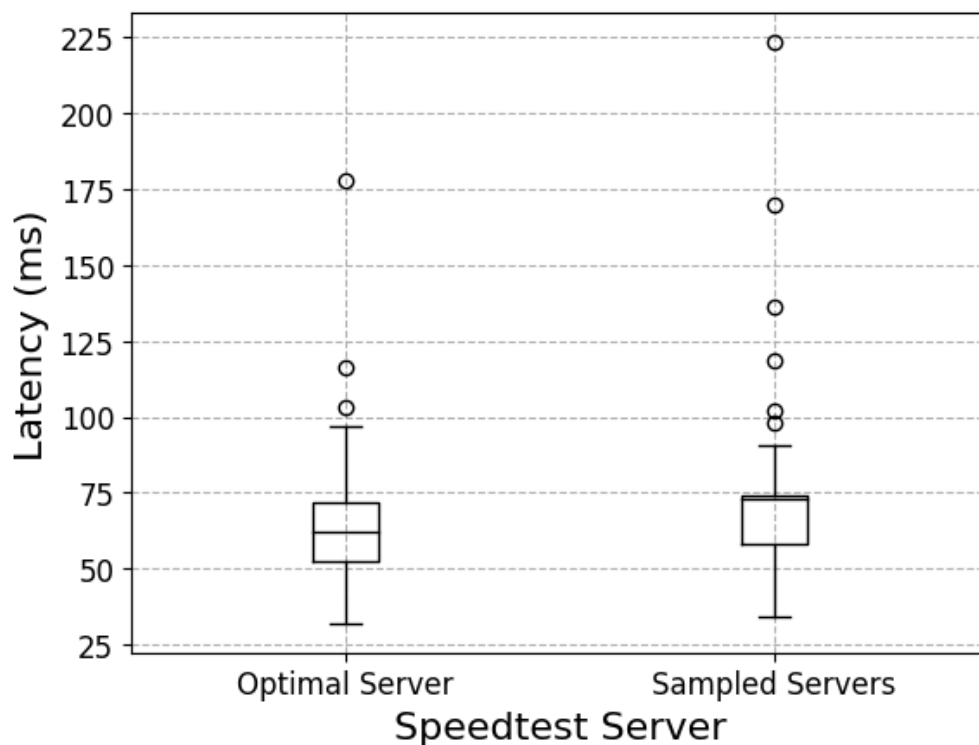


Figure 4.28. Box plot of latency results from speedtest through SatNet backhaul

Our numerical results for upload speeds of *Speedtest* are displayed in Table 4.16 and graphically depicted as a box plot and CDF in Figure 4.29 and Figure 4.30 respectively.

Table 4.16. Upload speeds using speedtest through SatNet backhaul

Testing Type	Mean (Mbps)	SD (Mbps)	Median (Mbps)	Maximum (Mbps)	Minimum (Mbps)
Speedtest Optimal Server	6.82	2.24	6.66	12.05	2.46
Speedtest Sampled Servers	6.87	1.78	6.87	11.73	4.07

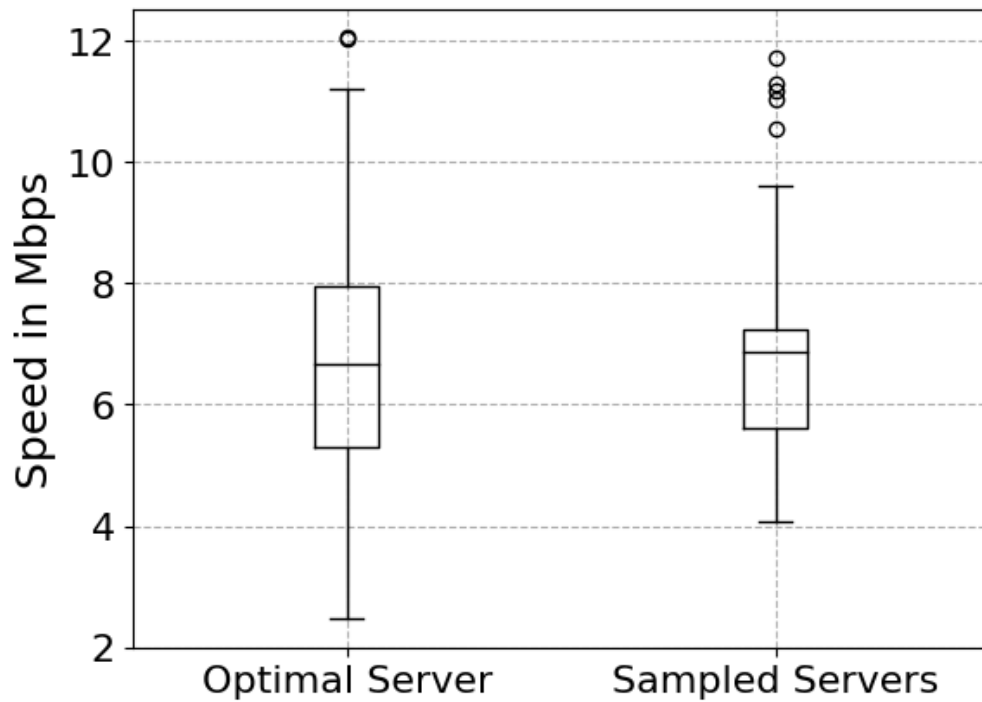


Figure 4.29. Box plot of upload speeds using speedtest through SatNet backhaul

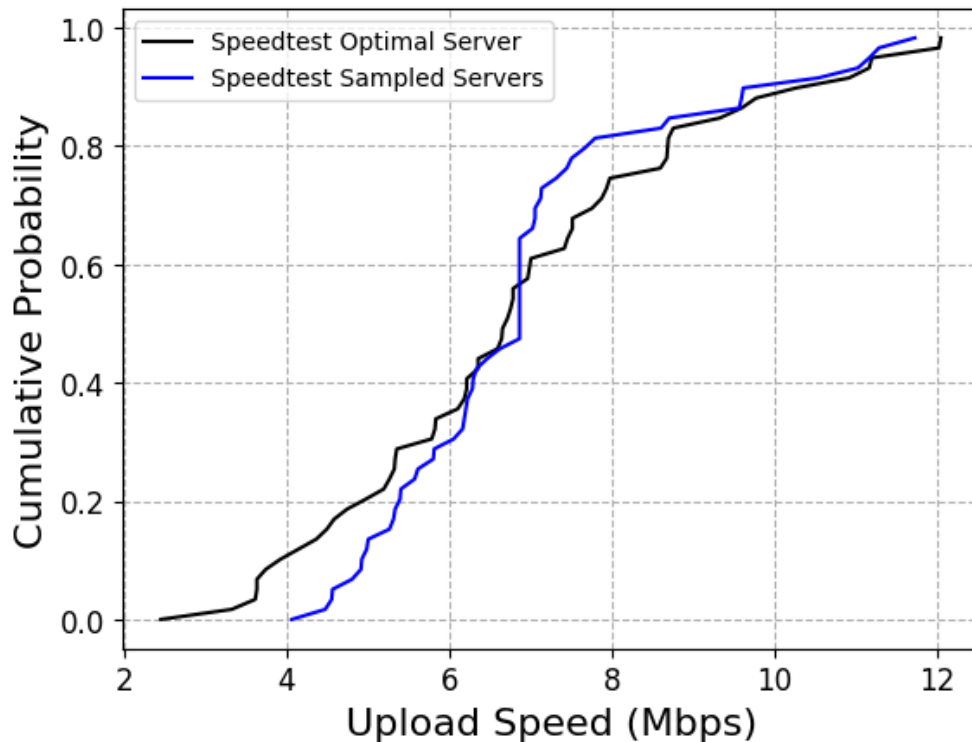


Figure 4.30. CDF of upload speeds using speedtest through SatNet backhaul

Much like the *Speedtest* server considerations for the download calculations, the upload tests for both the optimal and sampled servers were similar due to restrictions of the command line tool, only allowing the use of the closest 10 servers by ping time. Our network achieved average upload rates of almost 7 Mbps and a maximum of 12 Mbps. Anecdotally, streaming door cameras like Google Nest or Ring use between 2 to 4 Mbps of upload speed for their higher definition video streaming.

Throughput Compared to Proximity

We also completed an additional test to examine the data throughput rate as the UE distance to the network increased. This test examined similar aspects to the antenna and proximity test, but added in the condition of observed download rates. In the original proximity test, only control plane messages and RSRP values were considered for the extraction of maximum achieved operating distance alongside the signal quality. We compared download

rates at increasing distances to retrieve metrics associated with the operational capacity of the network. Of note, we used an omni-directional antenna to mimic how the 5G network would be leveraged in an operational communications environment.

We performed the same throughput test with the 50 MB *curl* file download, but at distance intervals of 1, 10, 20, 30, 40, 50, and 100 meters. Of note, a single 50 MB file download did not complete within 15 minutes for the 100 m test, meaning the download rate was less than 0.44 Mbps. The results are summarized in Figure 4.31, showing a significant decrease in overall throughput capability as distance increased.

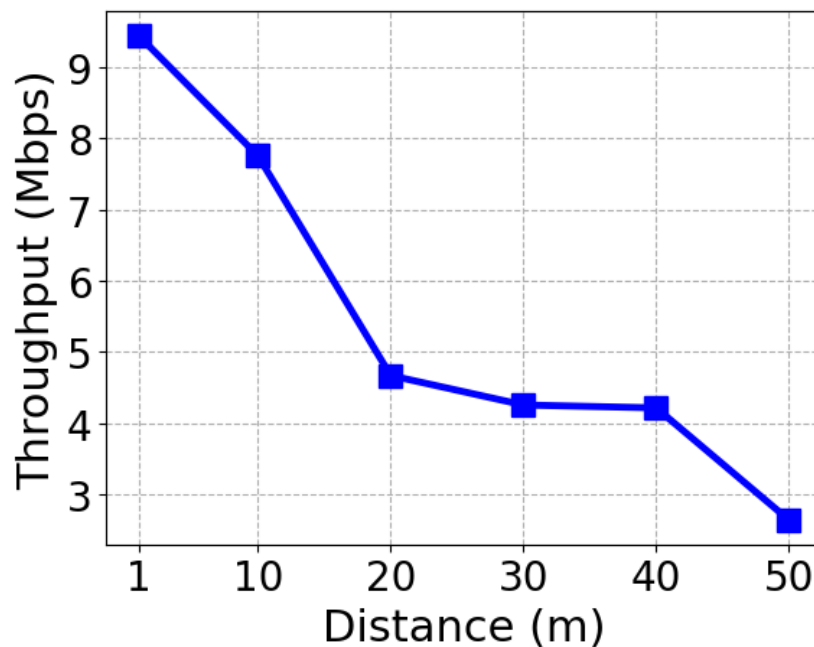


Figure 4.31. Comparison of throughput vs distance between UE and gNB

Even though our previous proximity testing revealed that the UE could maintain a connection with the network at up to 200 m through the control plane, this test indicates that the TP drastically decreases as the distance between the UE and network increases. Nonetheless, a seemingly unusable data connection starts to occur around 100 m, when the download rate decreases to much less than 1 Mbps.

4.4 Application and Use Cases

The ability to construct a 5G network from the ground up using only open source software and easily-procured COTS hardware represents an invaluable capability for numerous use cases. Such use cases could include creating on-demand networks for disaster relief efforts, small-team tactical or mobility operations, non-governmental organization (NGO) endeavors, underserved rural areas, and degraded maritime environments. A more generalized enumeration of use cases is provided in the following list:

1. **Emergency response** - quick setup for emergency first responders and other support personnel
2. **Military operations** - reliable communication for military units operating in remote or hostile environments, or as a blue force tracker (BFT)
3. **Rural areas** - coverage where traditional cellular networks are not available, but small congregations of people live
4. **Remote areas** - coverage in isolated areas such as oil rigs, mines, forward operating bases (FOBs), and the middle of the ocean for maritime platforms
5. **Temporary events** - coverage at temporary events such as concerts, festivals, fairs, and sports events
6. **IoT** - connect and manage IoT devices, such as sensors and other connected devices, in areas where traditional cellular networks are not available

Additionally, combining satellite connectivity with these 5G network architectures significantly improves the capabilities of the private network. Consequently, future requirements that mobile networks include the use of SatNets continues to drive upcoming 3GPP Releases. Integrating SatNets provides pervasive and near-worldwide access, especially important in areas with no viable network connections. SatNets can also provide consistent coverage to extremely dynamic and fast-moving users like airplanes and trains.

As a more basic use case, 5G networks can be leveraged to send messages between connected users. Unfortunately, Open5GS does not come equipped with an IP multimedia subsystem (IMS) server, so multimedia messaging directly between devices on the network is not possible. IMS servers provide an avenue for short message service (SMS), voice calls over IP, and video calling. Due to the lack of an IMS server, we chose to test device-to-device communication through the network using two IP-based solutions, Android team awareness

kit/Android tactical assault kit (ATAK) and message queueing telemetry transport (MQTT). These applications exemplify two of many potential ways to use our 5G network for user-user communications. We used the entirety of our network to test out these applications, including the data network component through our Starlink backhaul, demonstrating an end-to-end connection of UEs while served by our 5G network.

4.4.1 ATAK

ATAK is a widely used Android application for situational awareness and communication amongst the U.S. military as well as civilian users [68]. Other uses of this application include, but are not limited to, mission planning, geospatial referencing, full motion video (FMV), battlespace awareness, and geolocation of friendly forces. The civilian version of ATAK, sometimes referred to as CivTAK, is available for free both on the Android Play Store, as well as on github as an Android Packet Kit (APK). We downloaded the application from the Play Store while connected to our 5G network and proceeded to navigate to the server settings.

There exists a public server for CivTAK called FreeTAKServer, so we used this server to test our end-to-end device connection. The public server settings are as follows:

Server Address: 204.48.30.216

Streaming Protocol: TCP

Server Port: 8087

Figure 4.32 shows a successful connection to the ATAK public server, where messages can be sent between devices using a created chatroom, location updates of other users can be retrieved, and status reports can be posted.

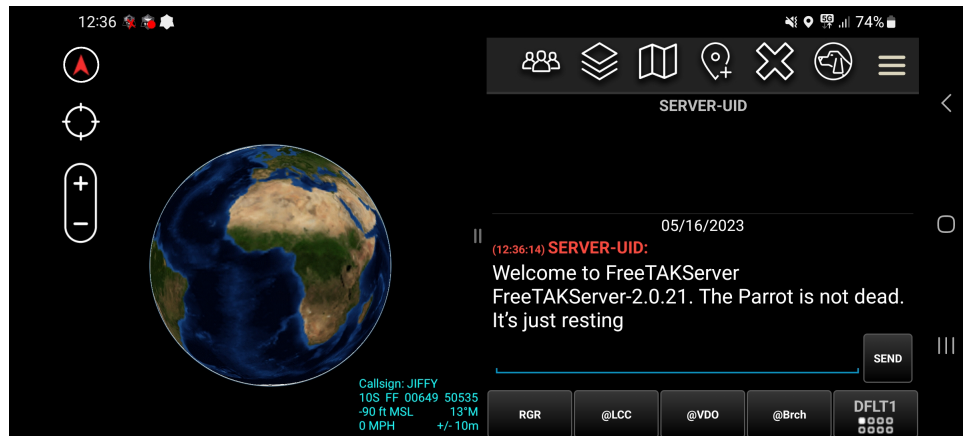


Figure 4.32. ATAK public server connection

Local Connection

We also tested functionality of ATAK when operating on our network locally in an ad hoc manner, meaning no connection to the data network. However, we were not able to successfully connect our UEs to the local server over the 5G connection. We did test the functionality of our ATAK server by successfully connecting through a common router shared by the server device (NUC).

FreeTAKServer is built using Python3 and is used to establish your own ATAK server, which integrates with CivTAK [69]. The FreeTAKServer github page contains detailed documentation, so we followed the installation guide at [69] to get our server up and running. Listing 4.2 shows the crucial changes necessary for some of the configuration files.

Listing 4.2: FreeTAKServer configuration changes

```
#In MainConfig.py
#change the following IPs to local machine:
"DataPackageServiceDefaultIP":
    {"default": _"192.168.1.55", "type": str},
"UserConnectionIP":
    {"default": _"192.168.1.55", "type": str},
```

```
#In config.py:
#192.168.1.55 is the wlan interface of our compute device
#Change IP that will be used to connect with the FTS API
IP = '192.168.1.55'
#webmap IP
WEBMAPIP = "192.168.1.55"
#the public IP your server is exposing
APPIP = '0.0.0.0'
```

Once fully installed, you need to start the server by running `sudo python3 -m FreeTAKServer.controllers.services.FTS`. You can also run the UI by changing into the UI directory at `/usr/local/lib/python3.8/dist-packages/FreeTAKServer-UI` and executing the python3 program with `python3 run.py`.

Then, you can access the UI in the web browser: `127.0.0.1:5000` with the default username and password:

```
username: admin
password: password
```

4.4.2 MQTT

MQTT is a lightweight messaging protocol mainly used by small IoT sensors and mobile devices with specific applications in limited networks with high latency or unreliable networks [70]. Some specific examples of integrating MQTT for UE communication on a 5G network include as a BFT for user location tracking and mapping, report sensor readings to a control center, and as a channel to pass other small scale data or information.

The protocol uses a publisher/subscriber (PubSub) model, based on the principle of publishing messages and subscribing to topics. So, clients can subscribe to topics that are relevant and subsequently receive messages that are published to those topics. These same clients can also publish messages to topics, making those available to any other subscribers to that topic. More simply, messages are published to topics and then clients can subscribe to the same topic to receive those messages. These messages are all handled by a broker to which both the publisher and subscriber have access. A visual depiction of the PubSub model is

shown in Figure 4.33.

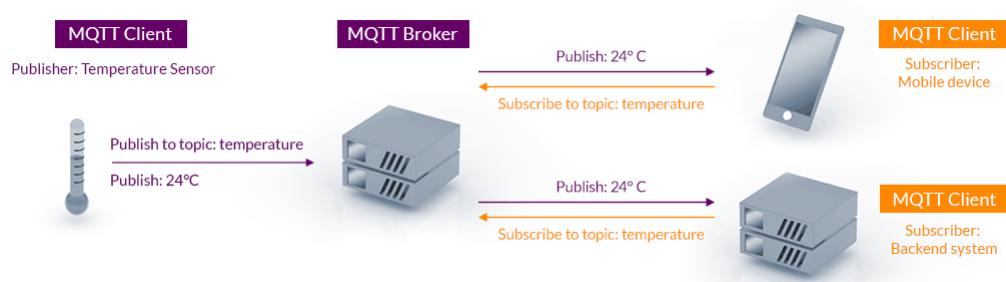


Figure 4.33. Publisher subscriber MQTT diagram. Adapted from [70]

For our testing, we used a free Android application from the Play Store called *MyMQTT*. The application provides a minimalistic user interface (UI) that allows the following inputs for broker connection:

1. Host address of broker
2. Port number
3. Toggle for SSL connection
4. Username and password if credentials used

After successful connection to a broker, you can now subscribe and/or publish to topics. In our test, we connected two Samsung devices to the same test broker at *test.mosquitto.org* and subscribed to a topic called *Open5GS_mqtt_test*. Subsequently, we published messages to that same topic and used it as a communication medium for our two devices as seen in Figure 4.34. Of note, we used the most basic setup of the devices and did not enable any additional TLS or credential requirements, while also using the test mosquitto broker online that anyone could see.

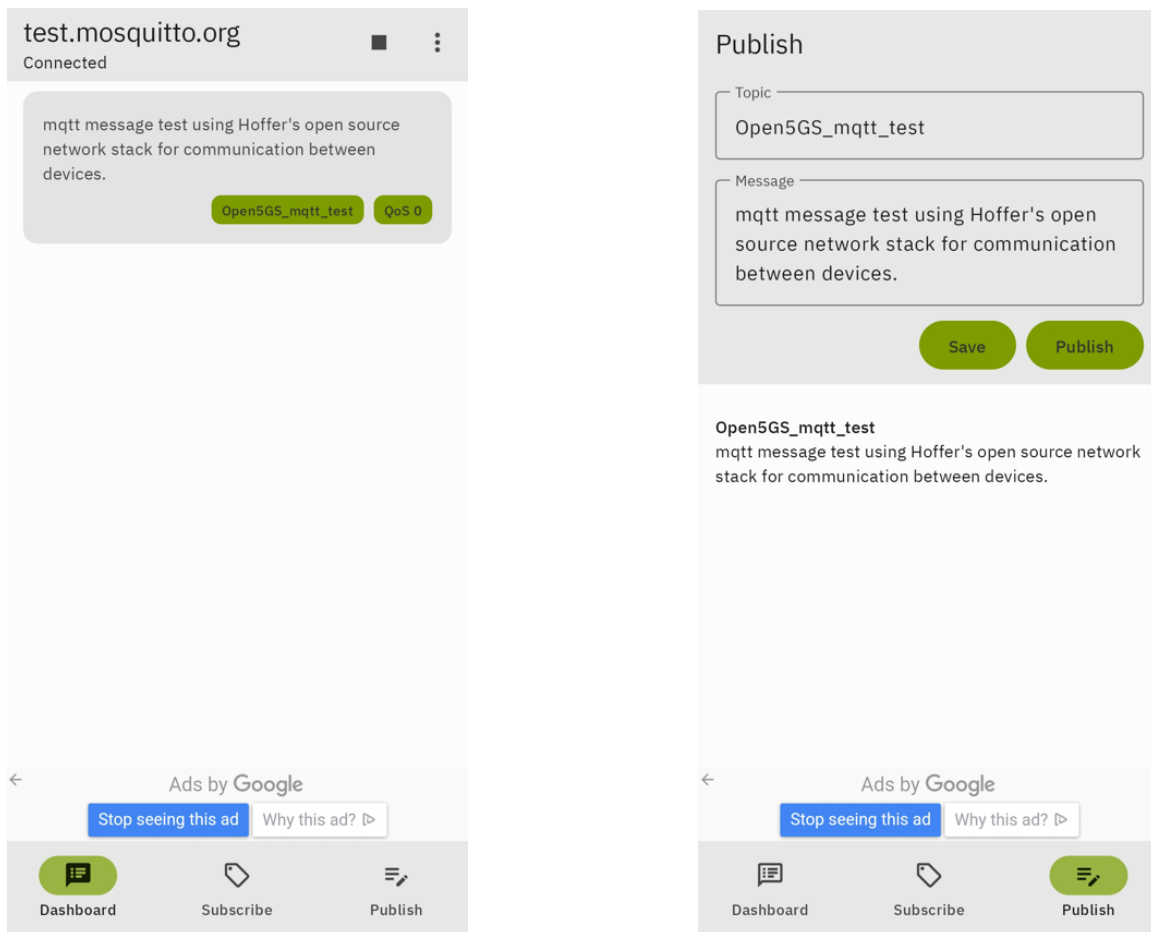


Figure 4.34. Two Samsung A22s using myMQTT connected to test mosquitto broker over our 5G network. Subscribed message retrieval is shown on the left from the published message on the right.

Local Connection

We also tested functionality of MQTT when operating on our network locally in an ad hoc manner, meaning no connection to the data network. However, we were not able to successfully connect our UEs to the local server over the 5G connection. We tested the functionality of our MQTT server by successfully connecting through a common router shared by the server device (NUC).

The mosquitto MQTT broker was also used when setting up the local MQTT broker, with

a Linux repository built specifically for it. The mosquitto broker can be downloaded by adding the repository and installing on the command line. Then, we set up a default listener on port 1883 with a specified protocol of mqtt, allowing for anonymous connections from clients with no associated username. These instructions were derived from [71] and are shown in listing 4.3.

Listing 4.3: Local mosquitto broker setup

```
#download and install mosquitto
sudo apt-add-repository ppa:mosquitto-dev/mosquitto-ppa
sudo apt update
sudo apt install mosquitto

#edit mosquitto config file
    persistence true
    persistence_location /home/snuc/mosquitto/data/

    log_dest stdout
    log_dest file /var/log/mosquitto/mosquitto.log

    connection_messages true
    log_timestamp true

#default listener
    listener 1883
    protocol mqtt

#no authentication, allow all connections
    allow_anonymous true
```

We used the same myMQTT android application to attempt connection from the Samsung UE to our local machine at *192.168.1.55* on port 1883. The local mosquitto broker will listen

to all connections on port 1883 incoming to the device. We were not able to successfully achieve connection to the local broker and deduce that the created PDU session tunnel between the UE and data network does not allow for communication of internal IP address space, but rather only out to the public Internet.

4.5 Observed Network Limitations

We have shown that our network establishes end-to-end connection, allowing for connected devices to reach through the Internet to communicate with external UEs. However, an apparent consideration with respect to srsRAN is that it only supports FDD bands with a 15 kHz subcarrier spacing (SCS), limiting the available bands to a subset of FR1 as shown in Table 4.17, and completely excluding any bands in FR2. Additionally, we were not able to instantiate the network on each of these bands due to a reported ARFCN and CORESET mismatch. In fact, we were only able to operate our network on bands 3, 7, and 20 resulting in the inability of some of the tested commercial phones to connect on our network. In addition to the UEs identified in this Chapter, we attempted to connect a OnePlus Nord N200 and Google Pixel 5, but they were not capable of using any of the limited network bands. The limited subset of usable bands with 5G SA FR1 significantly limits the potential hardware capable of operating on our network. Careful inspection and planning must be conducted in advance to ensure procured UE hardware will function on this limited subset of bands.

Table 4.17. srsRAN available bands. Adapted from [27].

Band	UL Frequency (MHz)	DL Frequency (MHz)	UL ARFCN Range	DL ARFCN Range	Successful Deployment
1	1920-1980	2110-2170	384000-396000	422000-434000	
2	1850-1910	1930-1990	370000-382000	386000-398000	
3	1710-1785	1805-1880	342000-382000	361000-376000	✓
5	824-849	869-894	164800-169800	1738000-178800	
7	2500-2570	2620-2690	500000-514000	524000-538000	✓
8	880-915	925-960	176000-183000	185000-192000	
20	832-862	791-821	166400-172400	158200-164200	✓
28	703-748	758-803	140600-149600	151600-160600	
66	1710-1780	2110-2200	342000-356000	422000-440000	
70	1695-1710	1995-2020	33900-342000	399000-404000	
71	663-698	617-652	132600-139600	123400-130400	
74	1427-1470	1475-1518	285400-294000	295000-303600	

As mentioned in Section 4.4, Open5GS does not have a native IMS server, so there is no automatic mechanism for providing device-device connections on our 5G network. We showed how IP-based applications can be used in place of IMS server support on the 5G network, but that solution requires traversing out to the Internet to exchange messages in the cloud instead of communicating in an ad hoc manner just through the local 5G network stack.

With respect to using SatNets for backhaul, there are obvious throughput limitations when using the network for bandwidth-intensive applications like video streaming or large file upload or downloads. However, SatNet coverage does extend to many worldwide locations as seen in the coverage map from Figure 3.13, while giving an end user much less speed when compared using broadband, fixed wireless, or traditional cellular networks.

In terms of network connection for the tested UEs, our 5G network often had issues with simply achieving device connection. Sometimes lack of connection occurred with no indication in the log files of any failed events, with other instances accompanied with errors that are not documented or attributed to a root cause. Additionally, if there were any network disturbances, the UE would not automatically reconnect to the network, but would rather require a complete device restart or toggle of airplane mode (cellular off and on) to reestablish connection. For example, if we connected a device to our network, then turned

our network off and back on again, the UE would not reconnect without intervention.

There are also some other network challenges more generally associated with using open source networks running on COTS hardware:

1. **Limited feature set:** open source solutions may not include all the features and functionalities of a commercial RAN solution
2. **Scalability:** open source solutions may not be able to handle a high number of users or devices
3. **Interoperability:** open source solutions may have issues intercommunicating with other mobile networks, along with compatibility issues with different UEs
4. **Security:** open source solutions may not have the same level of security as commercial RAN solutions
5. **Complexity:** open source projects require more technical knowledge and expertise to set up, configure, and maintain compared to a commercial RAN solution
6. **Limited frequency band support:** open source solutions may not support all frequency bands
7. **Limited hardware support:** open source solutions may not be compatible with all types of hardware devices
8. **Lack of purpose-built hardware:** commercial mobile network operators deploy specific hardware to maximize network efficiency, especially with respect to antenna selection and power emissions

4.6 Summary

The methodology and network setup in Chapter 3 were tested in this chapter. We demonstrated our capability to connect UEs through our network design presented in Chapter 3 to provide a mechanism for external communication. We also discussed the steps taken to facilitate connection and use of two different UEs. Additionally, we showcased how to best optimize network settings to achieve maximal results. Lastly, we investigated potential concerns and limitations associated with using our network operationally.

CHAPTER 5:

Conclusions

Even though 5G networks have been developed and deployed in various areas around the globe, their widespread employment remains limited. Utilizing a self-employed 5G network can permit communications in locations or events where the organic infrastructure is outdated or not usable. Our work in this thesis provided a proof-of-concept demonstration that such a network could be constructed from open source software and COTS hardware, establishing end-to-end communications through a SatNet backhaul.

5.1 Significant Contributions

The most significant contribution of this thesis is the successful design, construction, and testing of an open source 5G network. The ability to leverage an open standard software should enable earlier integration into existing communications architectures. This thesis outlined a systematic approach and step-by-step process to connect a recent, common, COTS UEs to an open source network. Furthermore, this thesis provides an understanding of the potential challenges associated with using open source implementations of 5G networks, and enumerates some of the corresponding capabilities and limitations. We also showed that hardware selection, particularly antennae, had significant effects on performance. In total, our contributions are summarized in the following list:

1. We successfully design, construct, and test an open source 5G network stack that enables end-to-end UE connections through SatNets.
2. We provide a systematic approach to configuring a 5G network, and illustrate a step-by-step process of how to configure unique commercial devices to operate on our network and achieve Internet connectivity.
3. We include detailed instructions on programming SIM cards capable of enabling network connection for multiple UEs.
4. We document our network optimization testing procedures showing the necessary RF settings and conditions to support device connections.
5. We illustrate necessary network proximity by comparing signal strength to increasing

- UE distances.
6. We conduct throughput testing measurements demonstrating achieved download and upload rates through our SatNet backhaul.
 7. We provide a methodology with associated scripts used for network testing that can be re-used with different network configurations and diversity of hardware and software.
 8. We explore several use cases of our 5G network for device communication and discuss other potential solutions for intra-network communication without the use of an IMS server.

5.2 Key Findings

Throughout the thesis process, we made several key findings. We discovered that an end-to-end 3GPP-compliant 5G network can be solely comprised of open-source components including all software and hardware, but that extensive SIM card and UE programming are necessary for connectivity within the 5G network. We also uncovered significant operating distances for devices connected to our 5G network, including more than 360 m when using directional antennas and up to 200 m with omni-directional. However, when considering the throughput in the data plane instead of just the control plane messages, useful ranges are limited to around 50 m, as download speeds decrease to below 0.5 Mbps at distances greater than 100 m.

A Quectel modem can be leveraged using bash scripts to test network performance in numerous configurations and additionally provide a network baseline before integrating commercial cellular devices. In addition, SatNet backhaul achieves maximal instantaneous download speeds at 1 m of up to 45 Mbps, however sustained download rates average closer to 10 Mbps with larger downloads. Similarly, upload speeds reach a maximal instantaneous speed at 1 m of 12 Mbps, with averages near 7 Mbps.

Consequently, our constructed network can be leveraged for numerous applications depending on the specific download and upload requirements. For example, use cases requiring video streaming services might not want find this current network configuration suitable, but IoT devices could certainly send short status updates or messages, even at further distances.

5.3 Recommendations for Future Work

Our work confirmed the viability of interconnecting disparate COTS hardware, software, and modules in order to create a portable 5G network that leveraged SatNets for access to the data network. The majority of research, however, focused on establishing end-to-end connection through the network. Consequently, there are numerous avenues for expanding our network and enriching our results with additional tests.

For expansion of the network, adding an IMS server to enable Voice over NR (VoNR) IP features should be the first consideration. Integration of an IMS server would eliminate the need for any IP-based applications for transmission and allow for direct communication of different devices connected to the network. Some other network-related modules could also be explored, such as a separate user plane server segregated from the control plane to establish a MEC-like aspect of the 5G network as suggested in [53], or instantiating the network on multiple computers by splitting the RAN and 5GC functions to then test the ability for multiple gNBs to leverage a single core. Other open source 5GC networks, such as SD-Core, could also be used and tested.

Enrichment of network testing can be accomplished by integrating more devices with additional hardware diversity. To do this, additional research should be conducted in producing functional ARFCN and CORESET pairs that successfully instantiate the network on different FR1 bands other than 3, 7, and 20. Another way to conduct further testing could include a study on cell reselection and handover by adding another proximal 5G network to investigate associated UE actions when interacting with multiple gNBs.

Further probing is necessary to figure out UE connection to either the ATAK or MQTT local server. The servers were properly established and functional within this thesis, but routing issues that prevented UE connection to either of the internally-bound servers. We deduce that the data tunnel created from the PDU session only gives UE access to the data network from the UPF.

With respect to security, SUPI concealment with use of the SUCI should be investigated, including the activation of UST service 124 for use of SUCI in network authentication. Subsequent addition of cryptographic keys within the SIM and 5GC are necessary for proper encryption and decryption. Additionally, efforts should be made to add security to

our IP-based use cases by activating TLS for those connections.

Lastly and potentially most importantly, as of the writing of this thesis, srsRAN has launched a new 5G-specific effort called *srsRAN Project* [72], which is compliant up to 3GPP Rel 17.4. This new project is a more robust and documented version of the 5G-upgraded srsLTE suite that was used for the research in this thesis. Work should be performed to compare the new project with the results achieved in this thesis and investigate any additional capabilities or limitations.

APPENDIX: Code Snippets

This appendix includes code snippets used for this thesis, including network start up procedures for the RAN and 5GC, and SDR download procedures.

A.1 Network Startup

Although the 5GC services will automatically startup during the CPU boot sequence, best practice consists of restarting the services before invoking the commands to launch the SDR and instantiate the gNB. The CPU should also be set to performance mode, according to [51], allowing the program to use as much memory as necessary to optimize performance. Listing 1 shows a simple shell script used to accomplish our network startup sequence. The `systemctl stop` and `start` commands for all the Open5GS 5GC functions are taken from [53].

Listing 1: Network startup script

```
#!/bin/bash

echo "Restarting Core Services"
sudo systemctl stop open5gs-mmef
sudo systemctl stop open5gs-sgwcd
sudo systemctl stop open5gs-smfd
sudo systemctl stop open5gs-amfd
sudo systemctl stop open5gs-sgwud
sudo systemctl stop open5gs-upfd
sudo systemctl stop open5gs-hssd
sudo systemctl stop open5gs-pcrfd
sudo systemctl stop open5gs-nrfd
sudo systemctl stop open5gs-ausfd
sudo systemctl stop open5gs-udmd
sudo systemctl stop open5gs-pcfd
sudo systemctl stop open5gs-nssfd
sudo systemctl stop open5gs-bsfd
```

```

sudo systemctl stop open5gs-udrd
sudo systemctl stop open5gs-webui
sudo systemctl stop open5gs-scpd

sudo systemctl restart open5gs-scpd
sudo systemctl restart open5gs-nrfd
sudo systemctl restart open5gs-mmed
sudo systemctl restart open5gs-sgwcd
sudo systemctl restart open5gs-smfd
sudo systemctl restart open5gs-amfd
sudo systemctl restart open5gs-sgwud
sudo systemctl restart open5gs-upfd
sudo systemctl restart open5gs-hssd
sudo systemctl restart open5gs-pcrfd
sudo systemctl restart open5gs-ausfd
sudo systemctl restart open5gs-udmd
sudo systemctl restart open5gs-pcfd
sudo systemctl restart open5gs-nssfd
sudo systemctl restart open5gs-bsfd
sudo systemctl restart open5gs-udrd
sudo systemctl restart open5gs-webui

echo "gNB startup sequence:"
echo "Putting CPU in performance mode"
echo "performance" | sudo tee /sys/devices/system/
cpu/cpu*/cpufreq/scaling_governor
echo "starting gNB"
sudo srsenb /home/snuc/Hoffer/configs/srsran/Band7/enb.conf

```

A.2 Software Defined Radio Dependencies

Listing 2 shows the series of code executed to properly download the UHD software for the USRP SDR and all the required dependencies. The code for Listing 2 was directly taken

from [73].

Listing 2: SDR download

```
#UHD dependencies
sudo apt-get -y install autoconf automake build-essential
ccache cmake cpufrequtils doxygen ethtool fort77
g++ gir1.2-gtk-3.0 git gobject-introspection
gpsd gpsd-clients inetutils-tools libasound2-dev
libboost-all-dev libcomedi-dev libcppunit-dev libfftw3-bin
libfftw3-dev libfftw3-doc libfontconfig1-dev libgmp-dev
libgps-dev libgsl-dev liblog4cpp5-dev libncurses5
libncurses5-dev libpulse-dev libqt5opengl5-dev
libqwt-qt5-dev libsdl1.2-dev libtool libudev-dev
libusb-1.0-0 libusb-1.0-0-dev libusb-dev libxi-dev
libxrender-dev libzmq3-dev libzmq5 ncurses-bin
python3-cheetah python3-click python3-click-plugins
python3-click-threading python3-dev python3-docutils
python3-gi python3-gi-cairo python3-gps python3-lxml
python3-mako python3-numpy python3-numpy-dbg
python3-opengl python3-pyqt5 python3-requests
python3-scipy python3-setuptools python3-six
python3-sphinx python3-yaml python3-zmq
python3-ruamel.yaml swig wget

#make a folder to hold the repository.
cd $HOME
mkdir uhd_lmh
cd uhd_lmh

#clone the repository and change directory
git clone https://github.com/EttusResearch/uhd
cd uhd
```

```
#checkout the desired UHD version
#you can get a full listing of tagged releases:
git tag -l

#check it out
git checkout release_003_011_000_001
git checkout v4.2.0.0

#create a build folder within the repository
cd host
mkdir build
cd build

#invoke CMake
cmake ..

#build UHD
make

#test to verify successful build process
make test

#install UHD
#default install UHD under the /usr/local/lib folder
#note: you need to run as root b/c folder permissions
sudo make install

#update the system's shared library cache
sudo ldconfig

#make sure that the LD_LIBRARY_PATH environment variable
#is defined and includes the folder where UHD installed
```

```

#most commonly, you #can add the line below to
#the end of your $HOME/.bashrc file:
export LD_LIBRARY_PATH=/usr/local/lib

#FPGA Downloads:
#download the UHD FPGA Images for this installation
sudo uhd_images_downloader
#try below if above command doesnt work
sudo usr/lib/udh/utils/uhd_images_downloader.py

#GNU Installation
sudo apt dist-update && sudo apt-get update
sudo apt install gnuradio

#Configuring USB
#On Linux, udev handles USB plug and unplug events.
#The following commands install a udev rule so that
#non-root users may access the device
cd $HOME/workarea/uhd/host/utils
sudo cp uhd-usrp.rules /etc/udev/rules.d/
sudo udevadm control --reload-rules
sudo udevadm trigger

#Connect the USRP
#UHD and GNU Radio now complete
#At this point, connect the USRP to the host computer.
#If the interface is USB, then open a terminal window
lsusb
#you should see the USRP listed on the USB bus with a
#VID of 2500 and PID of 0020 for the B200
#Also try running:
uhd_find_devices
uhd_usrp_probe

```

THIS PAGE INTENTIONALLY LEFT BLANK

List of References

- [1] DOD, “DoD 5G strategy,” Washington DC, USA, May 2020. Available: https://www.cto.mil/wp-content/uploads/2020/05/DoD_5G_Strategy_May_2020.pdf
- [2] DOD, “DOD 5G strategy implementation plan,” Washington DC, USA, Dec. 2020. Available: <https://www.cto.mil/wp-content/uploads/2020/12/DOD-5G-Strategy-Implementation-Plan.pdf>
- [3] 3GPP, “Study on New Radio (NR) to support non-terrestrial networks (NTN),” 3GPP, Tech. Rep. 38.821, 2023. Available: https://www.3gpp.org/ftp//Specs/archive/38_series/38.821/38821-920.zip
- [4] 3GPP, “Study on 5G System Support for Satellite Access and Backhaul,” 3GPP, Tech. Rep. 22.822, 2023. Available: https://www.3gpp.org/ftp//Specs/archive/22_series/22.822/22822-920.zip
- [5] 3GPP, “Study on Scenarios and Requirements for 5G Integration with Satellite Networks,” 3GPP, Tech. Rep. 22.824, 2023. Available: https://www.3gpp.org/ftp//Specs/archive/22_series/22.824/22824-920.zip
- [6] 3GPP, “Study on Interworking between Non-Terrestrial Networks (NTN) and Terrestrial Networks,” 3GPP, Tech. Rep. 22.825, 2023. Available: https://www.3gpp.org/ftp//Specs/archive/22_series/22.825/22825-920.zip
- [7] Congressional Research Service, “Low earth orbit satellites: Potential to address the broadband digital divide,” CRS Report No. R46896, Washington, DC: USA, August 31, 2021. Available: <https://crsreports.congress.gov/product/pdf/R/R46896>
- [8] J. Snyder, L. Hoffer, B. Martin, D. Rogers, and V. Kanth, “Comparing 5g network latency utilizing native security algorithms,” in *2023 Joint European Conference on Networks and Communications & 6G Summit (EuCNC/6G Summit)*, 2023, pp. 532–537.
- [9] I. Gomez-Migueluez, A. Garcia-Saavedra, P. D. Sutton, P. Serrano, C. Cano, and D. J. Leith, “srsLTE: An open-source platform for LTE evolution and experimentation,” *CoRR*, vol. abs/1602.04629, 2016. Available: <http://arxiv.org/abs/1602.04629>
- [10] S. Pramanik, A. Ksentini, and C. Fabiana Chiasserini, “Characterizing the computational and memory requirements of virtual rans,” in *2022 17th Wireless On-Demand Network Systems and Services Conference (WONS)*, 2022, pp. 1–8.

- [11] F. J. D. S. Neto, E. Amatucci, N. A. Nassif, and P. A. M. Farias, "Analysis for comparison of framework for 5G core implementation," in *2021 International Conference on Information Science and Communications Technologies (ICISCT)*. IEEE, 2021, pp. 1–5.
- [12] M. Kim, K. Park, J. Park, Y. Kim, J. Lee, and D. Moon, "Analysis of current 5G open-source projects," *Electronics and Telecommunications Trends*, vol. 36, no. 2, pp. 83–92, 2021.
- [13] C. S. Choudhari, R. Patil, and S. Saraf, "Deployment of 5G core for 5G private networks," in *2022 International Conference on Industry 4.0 Technology (I4Tech)*, 2022, pp. 1–6.
- [14] D. Johnson, D. Maas, and J. Van Der Merwe, "Open source ran slicing on powder: A top-to-bottom o-ran use case," in *Proceedings of the 19th Annual International Conference on Mobile Systems, Applications, and Services (MobiSys '21)*. New York, NY, USA: Association for Computing Machinery, 2021, p. 507–508. Available: <https://doi.org/10.1145/3458864.3466912>
- [15] J. P. Jasmin, "Leveraging openairinterface and software defined radio to establish a low-cost 5G non-standalone architecture," M.S. Thesis, Dept of Electrical Engineering, NPS, Monterey, CA, USA, December 2021. Available: <http://hdl.handle.net/10945/68767>
- [16] J. T. Mitchell, "Implementation of tactical open source 5G mobile networks," M.S. Thesis, Dept of Electrical Engineering, NPS, Monterey, CA, USA, September 2022. Available: <https://hdl.handle.net/10945/71080>
- [17] M. J. Lanoue, "Spoofed Networks: Exploitation of GNSS Security Vulnerability in 4G and 5G Mobile Networks," M.S. Thesis, Dept of Electrical Engineering, NPS, Monterey, CA, USA, June 2021. Available: <https://hdl.handle.net/10945/67451>
- [18] M. C. Leviton, "Usage of 5G in uav missions for isr," M.S. Thesis, Dept of Electrical Engineering, NPS, Monterey, CA, USA, September 2022. Available: <https://hdl.handle.net/10945/71075>
- [19] D. Lioprasitis, A. Priooulos, G. Gardikis, S. Pantazis, S. Costicoglou, A. Perentos, E. Hadjioannou, M. Georgiades, A. Phinikarides, A. Fornés-Leal, R. Gonzalez-Usach, C. E. Palau, and M. Esteve, "Satellite edge computing for 5G rural applications," in *2021 IEEE International Mediterranean Conference on Communications and Networking (MeditCom)*, 2021, pp. 1–2.

- [20] T. Darwish, G. K. Kurt, H. Yanikomeroglu, M. Bellemare, and G. Lamontagne, "LEO satellites in 5G and beyond networks: A review from a standardization perspective," *IEEE access*, vol. 10, pp. 35 040–35 060, 2022.
- [21] M. Y. Abdelsadek, H. Yanikomeroglu, and G. K. Kurt, "Future ultra-dense leo satellite networks: A cell-free massive mimo approach," in *2021 IEEE International Conference on Communications Workshops (ICC Workshops)*, 2021, pp. 1–6.
- [22] E. Dahlman, S. Parkvall, and J. Skold, *5G NR: The Next Generation Wireless Access Technology*. London, United Kingdom: Elsevier Science, 2020.
- [23] 3GPP, "NR; radio resource control (rrc); protocol specification," 3GPP, Technical Specification (TS) 38.331, 07 2022, version 17.1.0. Available: <https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3197>
- [24] Techplayon, "5G Master Information Block (NR-MIB)," Aug. 2018. Available: <http://www.techplayon.com/5G-nr-master-information-block-nr-mib/>
- [25] Pradeep, "LTE initial attach procedure in picture," Nov. 2013. Available: <https://ltebasics.wordpress.com/2013/11/08/lte-initial-attach-procedure-in-picture/>
- [26] 3GPP, "NR; user equipment (ue) radio transmission and reception; part 2: Range 2 standalone," 3GPP, Technical Specification (TS) 38.101-2, 06 2022, version 17.6.0. Available: <https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3284>
- [27] 3GPP, "NR; user equipment (ue) radio transmission and reception; part 1: Range 1 standalone," 3GPP, Technical Specification (TS) 38.101-1, 06 2022, version 17.6.0. Available: <https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3283>
- [28] Celona, "5G Bands Explained: How They Work & Why They Matter," October 14, 2020. Available: <https://www.celona.io/5G-lan/5G-bands>
- [29] 3GPP, "System architecture for the 5G system (5GS)," 3GPP, Technical Specification (TS) 23.501, 06 2022, version 17.5.0. Available: <https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3144>
- [30] ITU, "M.2083 : IMT Vision - "Framework and overall objectives of the future development of IMT for 2020 and beyond"," September 29, 2015. Available: <https://www.itu.int/rec/R-REC-M.2083-0-201509-I/en>
- [31] W. A. Jansen and A. Delaitre, "Reference material for assessing forensic sim tools," in *2007 41st Annual IEEE International Carnahan Conference on Security Technology*. IEEE, 2007, pp. 227–234.

- [32] 3GPP, “Characteristics of the universal subscriber identity module (usim) application,” 3GPP, Technical Specification (TS) 31.102, 09 2022, version 17.7.0. Available: <https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=1803>
- [33] sysmocom, “Cellular Network Infrastructure,” December 1, 2013. Available: <https://www.sysmocom.de/manuals/sysmocom-sim-manual.pdf>
- [34] 3GPP, “Numbering, addressing and identification,” 3GPP, Technical Specification (TS) 23.003, 09 2022, version 17.7.0. Available: <https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=729>
- [35] Paperbun, “5G-NR UE Identities (5G Identifiers),” September 7, 2021. Available: <https://paperbun.org/5G-nr-ue-identities-5G-identifiers/>
- [36] 3GPP, “Security architecture and procedures for 5G system,” 3GPP, Technical Specification (TS) 33.501, 09 2022, version 17.7.0. Available: <https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3169>
- [37] D. Kumar, “5G(NR)-GUTI, SUPI, SUCI,” February 21, 2021. Available: <https://www.5Gfundamental.com/2021/02/5G-guti-supi-suci.html>
- [38] Techplayon, “5G identifiers supi and suci,” November 6, 2019. Available: <https://www.techplayon.com/5G-identifiers-supi-and-suci/>
- [39] 3GPP, “NR and NG-RAN overall description;,” 3GPP, Technical Specification (TS) 38.300, 09 2022, version 17.2.0. Available: <https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3191>
- [40] *Space Operations*, JP-3-14 CH 1, Joint Chiefs of Staff, Washington, DC, USA, 2020 [Online]. Available: https://www.jcs.mil/Portals/36/Documents/Doctrine/pubs/jp3_14ch1.pdf
- [41] 3GPP, “Solutions for NR to support non-terrestrial networks (NTN),” 3GPP, Technical Specification (TS) 38.821, 05 2021, version 16.1.0. Available: <https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3525>
- [42] 3GPP, “Study on New Radio (NR) to support non-terrestrial networks,” 3GPP, Technical Specification (TS) 38.811, 09 2020, version 15.4.0. Available: <https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3234>

- [43] T. Darwish, G. K. Kurt, H. Yanikomeroglu, M. Bellemare, and G. Lamontagne, "LEO satellites in 5G and beyond networks: A review from a standardization perspective," *IEEE Access*, vol. 10, pp. 35 040–35 060, 2022.
- [44] SpaceX, "Starlink," August 17, 2022. Available: <https://www.starlink.com>
- [45] M. Wall, "SpaceX reveals 'Starshield' satellite project for national security use," Dec. 2022. Available: <https://www.space.com/spacex-starshield-satellite-internet-military-starlink>
- [46] J. McDowell, "Jonathan's Space Report | Space Statistics," September 28, 2022. Available: <https://planet4589.org/space/stats/star/starstats.html>
- [47] E. Musk, "Deck from spacex all-hands update talk i gave last week," June 5, 2022. Available: <https://twitter.com/elonmusk/status/1533408313894912001>
- [48] Intel(R), "Intel(r) nuc," September 1, 2022, <https://www.intel.com/content/www/us/en/products/details/nuc.html>.
- [49] EttisResearch, "USRP Hardware Driver and USRP Manual: USRP B2x0 Series," Accessed: January 10, 2023. Available: https://files.ettus.com/manual/page_usrp_b200.html
- [50] SRS, "Software radio systems: Trusted open software," Accessed: January 10, 2022. Available: <https://www.srs.io/>
- [51] Linux, "CPU Performance Scaling — The Linux Kernel documentation." Available: <https://www.kernel.org/doc/html/v4.14/admin-guide/pm/cpufreq.html>
- [52] 3GPP, "NG-RAN; pdu session user plane protocol," 3GPP, Technical Specification (TS) 38.415, 04 2022, version 17.0.0. Available: <https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3440>
- [53] S. Lee, "Open5gs is a c-language implementation of 5G core and EPC, i.e. the core network of NR/LTE network (release-16)," Accessed: April 15, 2023. Available: <https://open5Gs.org/open5Gs/>
- [54] S. Lee, "Open5gs quickstart guide," August 23, 2023. Available: <https://open5gs.org/open5gs/docs/guide/01-quickstart/>
- [55] 3GPP, "Specification of the 3GPP confidentiality and integrity algorithms EEA3 & EIA3; document 2: Zuc specification," 3GPP, Specification 35.222, 03 2022, version 17.0.0. Available: <https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=2400>

- [56] Quectel, “Quectel rm500q-ae iot/embb-optimized 5G sub-6 ghz m.2 module,” Accessed: July 14, 2022, https://www.quectel.com/wp-content/uploads/2021/03/Quectel_RM500Q-AE_5G_Specification_V1.1.pdf.
- [57] 3GPP, “Usim application test specification,” 3GPP, Technical Specification (TS) 31.121, 10 2022, version 17.0.0. Available: <https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=1814>
- [58] Haven Technologies, “Usb3 to m.2 key b 4G 5G modem adapter enclosure with sim card slot – v7 – sierra quectel fibcom simcom 5G supported,” Accessed: July 10, 2022. Available: <https://thewirelesshaven.com/shop/mini-pcie-m2-adapters/modem-enclosure/usb3-m2-4g-5g-modem-adapter-v7/>
- [59] 3GPP, “AT command set for User Equipment (UE),” 3GPP, Technical Specification (TS) 27.007, 09 2022, version 18.0.0. Available: <https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=1515>
- [60] Quectel, “Rg50xq&rm5xxq series at commands manual,” August 9, 2021, https://www.quectel.com/wp-content/uploads/2021/05/Quectel_RG50xQRM5xxQ_Series_AT_Commands_Manual_V1.2.pdf.
- [61] 3GPP, “NR; physical layer procedures for data,” 3GPP, Technical Specification (TS) 38.214, 09 2022, version 17.3.0. Available: <https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3216>
- [62] IEEE, “IEEE standard for definitions of terms for antennas,” *IEEE Std 145-2013 (Revision of IEEE Std 145-1993)*, pp. 1–50, 2014.
- [63] G. White and G. Louie, *The Audio Dictionary: Third Edition, Revised and Expanded*. University of Washington Press, 2005. Available: <https://books.google.com/books?id=DulVm8t88QkC>
- [64] Lcom, “Cellular/WiFi Multi-Band 3 dBi Specifications,” Accessed: January 13, 2023. Available: <https://www.l-com.com/wireless-antenna-cellular-wifi-multi-band-3-dbi-black-omni-antenna-w-magnetic-base-rp-sma-plug-connector>
- [65] Linux, “curl - How To Use,” August 26, 2023. Available: <https://curl.se/docs/manpage.html>
- [66] ArchLinux, “Android Debug Bridge - ArchWiki,” Accessed: March 13, 2023. Available: https://wiki.archlinux.org/title/Android_Debug_Bridge
- [67] Speedtest, “About Speedtest,” Accessed: August 3, 2023. Available: <https://www.speedtest.net/about>

- [68] ATAK, “CivTAK / ATAK,” Accessed: March 6 2023. Available: <https://www.civtak.org/>
- [69] FreeTAK Team, “Linux - FreeTAKServer Documentation,” Accessed: January 18, 2023. Available: https://freetakteam.github.io/FreeTAKServer-User-Docs/Installation/Linux/1_Install/
- [70] MQTT, “MQTT - The Standard for IoT Messaging,” Accessed: January 18, 2018. Available: <https://mqtt.org/>
- [71] R. Light, “mosquitto.conf man page,” Accessed: January 19, 2022. Available: <https://mosquitto.org/man/mosquitto-conf-5.html>
- [72] srsRAN Project Team, “5G - srsRAN Project,” Accessed: June 10, 2023. Available: <https://www.srsran.com/>
- [73] EttisResearch, “USRP Hardware Driver and USRP Manual: Building and Installing UHD from source.” Available: https://files.ettus.com/manual/page_build_guide.html

THIS PAGE INTENTIONALLY LEFT BLANK

Initial Distribution List

1. Defense Technical Information Center
Ft. Belvoir, Virginia
2. Dudley Knox Library
Naval Postgraduate School
Monterey, California



DUDLEY KNOX LIBRARY

NAVAL POSTGRADUATE SCHOOL

WWW.NPS.EDU

WHERE SCIENCE MEETS THE ART OF WARFARE