NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

APPLIED CYBER OPERATIONS CAPSTONE REPORT

LEVERAGING OPENAIRINTERFACE AND SOFTWARE DEFINED RADIO TO ESTABLISH A LOW-COST 5G NON-STANDALONE ARCHITECTURE

by

Jean P. Jasmin

December 2021

Advisor: Chad A. Bollmann
Second Reader: Darren J. Rogers

Approved for public release. Distribution is unlimited.

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE	Form Approved OMB No. 0704-0188

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 December 2021	3. REPORT TYPE AND DATES COVERED Applied Cyber Operations Capstone Report

4. TITLE AND SUBTITLE LEVERAGING OPENAIRINTERFACE AND SOFTWARE DEFINED RADIO TO ESTABLISH A LOW-COST 5G NON-STANDALONE ARCHITECTURE	5. FUNDING NUMBERS
6. AUTHOR(S) Jean P. Jasmin	
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) N/A	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)

Commercial cellular service providers are at the forefront of the paradigm shift from 4G Long Term Evolution (LTE) to 5G New Radio (NR). The increase in throughput, provisioning of ultra-low latency, and greater reliability of 5G enable potential uses that no other wireless communication could support. The Department of Defense (DOD) is interested in 5G NR technologies, but the implementation of the architecture can be lengthy and costly. This capstone configured a 4G LTE network and a 5G non-standalone network using OpenAirInterface and software defined radios (SDR). Universal Subscriber Identity Module (USIM) cards were configured and introduced to user equipment and attached to the 4G LTE network. A gNodeB (gNB) was added to the 4G LTE network to establish the 5G non-standalone (NSA) network architecture (3GPP Option 3). The testbed developed in this research was able to connect the core to a commercial internet service provider and browse the internet using third-party applications. Our analysis educates future researchers on the challenges and lessons learned when implementing the OpenAirInterface 4G LTE and 5G NSA networks. This work also provides a better understanding of 4G LTE and 5G NSA OpenAirInterface software usability, flexibility, and scalability for potential use cases for the DOD.

14. SUBJECT TERMS 5G, OpenAirInterface, OAI, 4G LTE, Long-Term Evolution, EPC, Evolved Packet Core, eNB, eNodeB, Evolved Node B, gNB, gNodeB, SDR, software defined radio, NSA, non-standalone	15. NUMBER OF PAGES 133
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UU

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release. Distribution is unlimited.

LEVERAGING OPENAIRINTERFACE AND SOFTWARE DEFINED RADIO TO ESTABLISH A LOW-COST 5G NON-STANDALONE ARCHITECTURE

CPO Jean P. Jasmin (USN)

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN APPLIED CYBER OPERATIONS

from the

NAVAL POSTGRADUATE SCHOOL
December 2021

Reviewed by:

Chad A. Bollmann
Advisor

Darren J. Rogers
Second Reader

Accepted by:

Alex Bordetsky
Chair, Department of Information Sciences

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

Commercial cellular service providers are at the forefront of the paradigm shift from 4G Long Term Evolution (LTE) to 5G New Radio (NR). The increase in throughput, provisioning of ultra-low latency, and greater reliability of 5G enable potential uses that no other wireless communication could support. The Department of Defense (DOD) is interested in 5G NR technologies, but the implementation of the architecture can be lengthy and costly. This capstone configured a 4G LTE network and a 5G non-standalone network using OpenAirInterface and software defined radios (SDRs). Universal Subscriber Identity Module (USIM) cards were configured and introduced to user equipment and attached to the 4G LTE network. A gNodeB (gNB) was added to the 4G LTE network to establish the 5G non-standalone (NSA) network architecture (3GPP Option 3). The testbed developed in this research was able to connect the core to a commercial internet service provider and browse the internet using third-party applications. Our analysis educates future researchers on the challenges and lessons learned when implementing the OpenAirInterface 4G LTE and 5G NSA networks. This work also provides a better understanding of 4G LTE and 5G NSA OpenAirInterface software usability, flexibility, and scalability for potential use cases for the DOD.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

I. INTRODUCTION....1
- A. SCOPE....2
- B. RESEARCH OBJECTIVE....3
- C. ORGANIZATION OF CAPSTONE....4

II. BACKGROUND AND LITERATURE REVIEW....5
- A. BACKGROUND....5
  - 1. History of Advances in Cellular Technologies....5
  - 2. Air Interface Stack....7
  - 3. Architecture and Equipment....11
- B. LITERATURE REVIEW....14
  - 1. OpenAirInterface—4G LTE....15
  - 2. OpenAirInterface—5G NSA....15
  - 3. Comparison to Prior Work....17
  - 4. OpenAirInterface Competitors....17

III. METHODOLOGY....19
- A. USRP....19
- B. ENB/GNB....20
- C. EPC....20
- D. PROVISIONING UNIVERSAL SUBSCRIBER IDENTITY MODULE....21
- E. SIMULATION....24
  - 1. Network Configuration Setup....24
  - 2. Deploying the eNB / gNB....25
  - 3. Connecting the User Equipment....26

IV. RESULTS....31
- A. NETWORK CONFIGURATIONS....31
  - 1. Network Configuration 1: EPC, eNB, and COTS UE....32
  - 2. Network Configuration 2: EPC, eNB1, eNB2, and COTS UE....34
  - 3. Network Configuration 3: EPC, eNB, gNB, and COTS UE....36
- B. ANALYSIS....38
  - 1. OpenAirInterface Platform....38
  - 2. Evolved Packet Core....40

3. eNB 42
4. gNB 42
5. User Equipment 43
6. Network Performance 44
7. Network Mobility 46

V. CONCLUSIONS AND FUTURE WORK 49
A. CONCLUSIONS 49
B. FUTURE WORK 50

APPENDIX A. OPENAIRINTERFACE SOFTWARE INSTALLATION GUIDE 53
A. PREREQUISITE AND USRP SETUP 53
1. Low-Latency Kernel 53
2. VOLK Libraries 53
3. GNU Radio 54
4. USRP UHD 54
5. GNU Radio 57
B. ENB/GNB 58
1. eNB Installation 58
2. eNB Configuration 59
3. gNB Installation 62
4. gNB Configuration 63
C. EPC 65
1. Low-Latency Kernel 65
2. freeDiameter 66
3. Docker 66
4. openair-epc-fed 68
D. UPDATED EPC INSTALLATION 72
1. openair-epc-fed 72

APPENDIX B. OPENAIRINTERFACE SOFTWARE SIMULATION 75
A. SETTING UP BASH SCRIPTS 75
B. LAUNCHING THE EPC 75
1. Start the EPC Containers Script 75
2. Stop the EPC Containers Script 76
3. Recover the Logs, Configurations, and Traces 77
4. Restart EPC Containers script 78
C. OPTIONAL FLAGS FOR LAUNCHING THE ENB/GNB 78

APPENDIX C. INSTALLING DIAGPARSER AND MINICOM 81
A. INSTALLING DIAGPARSER AND MINICOM.....81
B. CONNECTING TO QUECTEL MODULE.....82
C. AT COMMANDS.....85

APPENDIX D. COMMAND LINE CHEAT SHEET 87
A. COMMON LINUX COMMAND LINE USED 87
B. USEFUL DOCKER COMMAND LINE USED.....89

APPENDIX E. LATENCY RESULTS 91

APPENDIX F. IPERF RESULTS 93
A. IPERF TEST FROM UE TO EPC.....93
B. IPERF TEST FROM UE TO ENB 93
C. IPERF TEST FROM UE TO GNB.....94

APPENDIX G. 4G AND 5G FREQUENCY OPERATING BAND CHARTS 95

APPENDIX H. 3GPP CORE NETWORKS RADIO-ACCESS TECHNOLOGY 101

SUPPLEMENTAL FILE STRUCTURE.....103
A. HSS FILE STRUCTURE.....103
B. MME FILE STRUCTURE 103
C. SPGW-C FILE STRUCTURE.....103
D. SPGW-U FILE STRUCTURE.....104

LIST OF REFERENCES.....105

INITIAL DISTRIBUTION LIST 111

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF FIGURES

Figure 1. The Architecture of the Air Interface Protocol Stack. Adapted from [10]. 8

Figure 2. RAN Protocol Architecture. Source [13]. 9

Figure 3. Example of LTE Data Flow. Source [13]. 11

Figure 4. 4G E-UTRA and 5G NR Dual Connectivity (3GPP Option 3a). Adapted from [15] 12

Figure 5. EPC Network Functions 13

Figure 6. OYEITIMES SIM Writer 22

Figure 7. Physical Architecture of the Simulation Lab 24

Figure 8. Quectel QNavigator V1.6.9—Home Tab 29

Figure 9. eNB, EPC host, and Docker Containers IP Scheme 31

Figure 10. Network Configuration 1 32

Figure 11. MME Log 33

Figure 12. MME Log Showing eNB Peer Connection 33

Figure 13. Snapshot of MME Log Showing eNB Peer Connection and UE Attached 34

Figure 14. Dual eNB and EPC Network Scheme 35

Figure 15. Snapshot of MME Log Showing Connected eNBs 35

Figure 16. Snapshot of MME Log Connected eNBs and Attached UEs 36

Figure 17. eNB, gNB, and EPC IP NSA Network Scheme 36

Figure 18. eNB View of gNB Peer Connection with eNB 37

Figure 19. gNB View of Peer Connection with eNB 38

Figure 20. MME Log SEGFAULT 44

Figure 21. enb.band7.tm1.25PRB.usrpb210 - eNB Configuration 60

Figure 22. enb.band7.tm1.25PRB.usrpb210 - MME Parameters and Network_Interfaces.....61

Figure 23. gnb.band78.tm1.fr1.106PRB.usrpb210—gNB Configuration 63

Figure 24. gnb.band78.tm1.fr1.106PRB.usrpb210—MME Parameters and Network_Interfaces.....64

Figure 25. Core Networks Deployment Models. Source [54].....101

LIST OF TABLES

Table 1. Description of USIM Fields Updated in OYEITIMES SIM Writer. Adapted from [33], [34], and [35]. 23

Table 2. OnePlus 6T APN Settings.....27

Table 3. Google Pixel 4a 5G APN Settings 28

Table 4. NPS GitHub.....41

Table 5. iPerf Test Results 45

Table 6. Ping Test Results (average) 46

Table 7. Changes Made in the eNB Configuration File [45]. 60

Table 8. enb.band7.tm1.25PRB.usrpb210—MME Parameters and X2 Settings.....61

Table 9. enb.band7.tm1.25PRB.usrpb210—Network Interfaces Settings.....62

Table 10. srpb210—gNB configuration.....63

Table 11. gnb.band78.tm1.fr1.106PRB.usrpb210—MME parameters and X2 Settings.....65

Table 12. gnb.band78.tm1.fr1.106PRB.usrpb210—Network Interface Settings 65

Table 13. Optional flags to Launch the eNB. Adapted from [46]. 78

Table 14. Optional flags to Launch the gNB. Adapted from [46]. 79

Table 15. AT Commands Used.....85

Table 16. Minicom Quick Command Reference 85

Table 17. Useful Linux Commands Used. Source [49]. 87

Table 18. Useful Docker Commands Used. Source [51]. 89

Table 19. Overall Ping Results between OnePlus 6T and Google Pixel 4a 5G.....91

Table 20. Ping Results 92

Table 21. Full iPerf Results from UE to EPC 93

Table 22. Full iPerf Results from UE to eNB 93

Table 23. iPerf Results from UE to gNB 94

Table 24. 4G LTE Operating Bands. Source [52].....95

Table 25. 5G NR FR1 Operating Bands. Source [53]. 97

LIST OF ACRONYMS AND ABBREVIATIONS

1G	First Generation
2G	Second Generation
3G	Third Generation
4G	Fourth Generation
5G	Fifth Generation
3GPP	Third-Generation Partnership Project
AM	Acknowledge Mode
AT	ATtention
CLI	Command Line Interface
CN	Core Network
COTS	Commercial Off-The-Shelf
CPU	Central Processing Unit
CUPS	Control and User Plane Separation
DOD	Department of Defense
DRB	Data Radio Bearer
DSP	Digital Signal Processing
eMBB	enhanced Mobile Broadband
EMM	EPS Mobility Management
eNB	Evolved NodeB
EN-DC	E-UTRAN and NR Dual Connectivity
EPC	Evolved Packet Core
EPS	Evolved Packet System
ESM	EPS Session Manager
E-UTRA	Evolved Universal Terrestrial Radio Access
FCC	Federal Communications Commission
FGPA	Field-Programmable Gateway Array
gNB	gNodeB
GPRS	General Packet Radio Services
GPSDO	GPS-disciplined, oven-controlled oscillator
GSM	Global System for Mobile Communications

GTP	GPRS Tunneling Protocol
GUI	Graphical User Interface
HSS	Home Subscriber Server
IM	Instant Messaging
IoT	Internet of Things
IP	Internet Protocol
ISP	Internet Service Provider
LTE	Long Term Evolution
MAC	Medium Access Control
MIMO	Multiple-Input Multiple-Output
MME	Mobility and Management Entity
MMS	Multimedia Message Service
mMTC	massive Machine Type Communications
mmW	Millimeter Wave
NAS	Non-Access Stratum
NB-IoT	Narrowband Internet-of-Things
NR	New Radio
NSA	Non-Standalone
OAI	OpenAirInterface
OS	Operating System
OSA	OpenAirInterface Software Alliance
PC	Personal Computers
PDCP	Packet Data Convergence Protocol
PDN	Packet Data Network
PDU	Packet Data Unit
PGW	Packet Data Network Gateway
PHY	Physical
PLMN	Public Land Mobile Network
QoS	Quality-of-Service
RAN	Radio Access Network
RF	Radio Frequency
RLC	Radio Link Control

RRC	Radio Resource Control
SA	Stand-Alone
SDR	Software-Defined Radio
SDU	Service Data Unit
SGW	Serving Gateway
SIM	Subscriber Identity Module
SIMD	Single Instructions, Multiple Data
SMA	SubMiniature version A
SME	Subject Matter Expert
SMS	Short Message Service
SN	Sequence Number
SPGW	Service Packet Gateway
SPGW-C	Service Packet Gateway Control Plane
SPGW-U	Service Packet Gateway User Plane
SRB	Signaling Radio Bearer
TCP	Transmission Control Protocol
UDM	Unified Data Management
UDP	User Datagram Protocol
UE	User Equipment
UHD	USRP Hardware Driver
UM	Un-Acknowledge Mode
UPF	User Plane Function
URLLC	Ultra-reliable, Low Latency Communications
USIM	Universal Subscriber Identity Module
USRP	Universal Software Radio Peripheral
VOLK	Vector-Optimized Library of Kernels
VoLTE	Voice over LTE

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGMENTS

I would like to thank the following people for their assistance, guidance, and insight throughout this process.

Dr. CDR Chad Bollmann: thank you for your time, support, confidence, and mentorship during this process.

Mr. Darren Rogers: thank you for the exchange of ideas, advice, and assistance when I experienced difficulties in this project.

Most importantly, I thank my wife for her patience and devotion to our family while I focused on this project, and my kids for their daily reminder of why I must remain focused. I would not have completed this project without your support.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

In recent years, service providers have focused on delivering 5G technologies throughout the world. 5G is the fifth generation of wireless mobile technology that improves upon 4G Long-Term Evolution (LTE) technology. However, 4G is still widely used today and is likely to co-exist with 5G. The rollout of 5G began when the 3rd Generation Partnership Project (3GPP)—a consortium that governs specifications regarding mobile telecommunication networks—introduced a technical report known as Release 15. 5G technology offers increased bandwidth, extremely low latency, ultra-reliability, and the ability to support a massive surge of mobile devices known as the Internet of Things (IoT).

5G can accommodate the increased demand for reliable, fast, and low-latency data communications. The 5G spectrum operates in the low, mid I and II, and the high-band spectrums. The low-band covers the frequency range from 70 Megahertz (MHz) to 1 Gigahertz (GHz) and can travel up to 30 kilometers (km). The mid-bands consist of two different parts, mid-band I and mid-band II. Mid-band I operates between 1 GHz and 2.6 GHz and can travel up to 15 km, while mid-band II operates between 3.5 GHz and 6 GHz range and travels up to 8 km. The low-band and mid-bands together are referred to as the sub-6 band. The high-band operates between 24 - 40 GHz and travels less than 1 km [1]. High-band is also referred to as millimeter Wave (mmW).

The 5G frequency bands are implemented in two frequency ranges: "frequency range 1" (FR1) and "frequency range 2" (FR2). FR1 falls within the mid-band II spectrum—3.3 – 4.2 GHz and 4.4 – 4/99 GHz. FR2 is within the high-band spectrum—24.25 – 29.5 GHz, 31.8 – 33 – 4 GHz, and 37 – 40 GHz [2]. The low-band, mid-band, and high-band spectrum each has advantages and disadvantages. The low-band spectrum offers higher distance and better propagation through objects. However, the high-band spectrum can support a higher data rate and lower latency, but it cannot propagate through buildings and trees. Though mid-band does not travel as far as the low-band, nor does it support the speed and latency that high-band offers, the mid-band spectrum offers higher speeds than low-band and greater distance than high-band.

Implementing a 5G network typically requires proprietary equipment only offered by large-scale vendors, which can be very expensive. An alternative approach is using software to replace hardware infrastructure. OpenAirInterface is an alternative low-cost, open-source initiative that provides a software implementation of cellular technology components: an eNodeB (eNB); a gNodeB (gNB); a user equipment (UE); and an evolved packet core (EPC) using computing platforms (x86) along with software-defined radios (SDR). This study aims to create a foundation for other research into the potential uses of 5G non-standalone (NSA) architecture at a low cost.

The purpose of this capstone is to facilitate future Department of Defense (DOD) research into operationally relevant uses of open-source software in 5G network implementations. The increased speed and ultra-low latency of 5G creates a broad spectrum of use that enables technology to communicate wirelessly. The DOD recognizes the benefits and potential uses of 5G networks.

As of October 2020, the DOD became a stakeholder in 5G cellular technologies when the DOD approved $600 million to install 5G on five U.S. military bases to revolutionize their current logistical operations [3]. However, other than the approval of $600 million to implement 5G, there has not been much research published on the cost to implement a network on DOD installations. The DOD's step toward modernizing its current cellular communications infrastructure shows its interest in the potential uses of 5G that could increase the efficiency of its daily operations. This capstone introduces open-source software and hardware for a 5G network; another goal is to identify whether vulnerabilities exist within the OpenAirInterface software, ultimately enhancing the DOD's ability to assess operationally relevant use cases.

A. SCOPE

This research seeks to document a potential implementation of the OpenAirInterface 4G/5G stack and our provisioning of 4G/5G user devices. This research also assesses the OpenAirInterface 5G NSA network architecture usability, flexibility, and security for potential DOD use. This capstone also establishes a physical testbed using the

4G LTE network and reconfigures the OpenAirInterface to support a 5G non-standalone network.

The first phase of this capstone establishes a 4G LTE network using the OpenAirInterface software stack. The architecture implements the 5th Generation Radio Technology Base Station (gNodeB) and the EPC via the OpenAirInterface software. Once the 4G LTE stack is established, the next step is to configure OpenAirInterface as a 5G NSA architecture, introduce a USIM, and test whether it can successfully access the 5G network. During this research phase, OpenAirInterface software logs are analyzed to ensure that each component can successfully communicate.

The second phase analyzes the OpenAirInterface to determine if there are known vulnerabilities present that are native to the traditional 5G architecture. With access to available cellular radios, this research can further inspect various aspects of the stack. This capstone can also use commercial cellular phones and other Quectel Modules to inspect different aspects/vectors for consideration when implementing 5G.

B. RESEARCH OBJECTIVE

This research explores the OpenAirInterface 5G NSA network architecture, which creates an environment where mobile devices can communicate, providing comprehensive insights into the different challenges of implementing the network. We establish reconfigurable 4G LTE and 5G NSA networks, using the OpenAirInterface software, to gain a better understanding of the virtualization of traditional cellular management systems. The goal is to determine whether OpenAirInterface can be leveraged to establish 5G cellular networks at a low cost.

This research is a first step to enabling the DOD to utilize open-source and inexpensive software to deploy a 5G architecture that could support service installations and integrate with a commercial backhaul. This research also aims to encourage further DOD research on using cellular technology and facilitate DOD use of the cellular spectrum for military purposes. Another objective of this study is to identify vulnerabilities within the OpenAirInterface stack, enabling the DOD to assess operationally relevant use cases.

C. ORGANIZATION OF CAPSTONE

Chapter II discusses important concepts of cellular technologies from the first generation to the fifth generation. Chapter II also reviews existing research literature that showcases different uses of the OpenAirInterface software. The literature review is organized to show the different implementations of the OpenAirInterface software stack. In Chapter III, we outline the design of the OpenAirInterface software stack and the simulation of the 4G LTE / 5G NSA network. Chapter III also covers the methodology used to establish the network used for this capstone. Chapter IV presents the results. Finally, Chapter V is an analysis and an overview of the research results and contains future work.

II. BACKGROUND AND LITERATURE REVIEW

This chapter discusses the history of cellular technology. The background includes key advances in cellular technology from the first generation through the fifth generation, the 4G LTE architecture, as pertinent to our work using the 5G NR NSA architecture. This chapter also summarizes the 4G LTE air interface stacks and critical protocols that enable traffic between the 4G network components. Finally, this chapter reviews different literature discussing various implementations of the OpenAirInterface software.

A. BACKGROUND

1. History of Advances in Cellular Technologies

The first-generation (1G) in cellular technology was introduced in the late 1970s and became popular in the early 1980s. 1G technology enabled the use of wireless mobile devices. However, 1G technology was only capable of basic voice communications based on analog technology called Advanced Mobile Phone System (AMPS), which operates in the 800 – 900 MHz frequency range. 1G traffic is multiplexed using frequency-division multiple access (FDMA). It could transfer data at a maximum speed of 9.6 kbps within the 150 MHz frequency range, described in [4].

The second-generation (2G) in cellular technology was first introduced in the late 1980s. 2G uses digital modulation instead of analog technology used in 1G. The standard used to deploy 2G was the Global System for Mobile Communications (GSM). 2G became known as 2.5G when the General Packet Radio Service (GPRS) was implemented into the system. GPRS is a packet-switching protocol that allows GSM-based devices to deliver data in other ways than short messages service (SMS). GPRS enabled 2G and 2.5G to improve spectrum utilization efficiency, data service, transmission quality, system capacity, and coverage. In addition to SMS, 2G also enabled fax and multimedia messaging service (MMS). 2G maximum data transfer rate is 43.2 kbps while 2.5G maximum data rate peaks at 384 Kbps, as shown in [5].

Approximately ten years later, the third-generation (3G) of mobile networks was introduced. 3G was the first mobile system capable of up to 2 Mbps of data transfer [5] and

supporting quality-of-service (QoS) controls. In addition, 3G improved voice quality and increased data transmission capacity. The increase in data transmission enabled wireless internet access, allowing faster web browsing, video streaming, and more security. The emergence of mobile devices widely accessing the internet manifested from these advances in smartphones and 3G cellular networks. The universal mobile telecommunication system (UMTS) radio access network (UTRAN) is the 3G cellular network based on the GSM standard. 3G cellular technology also included the wideband code division multiple access (WCDMA), which "increase [d] system capacity and communication quality" [5].

The first fourth generation (4G) network was launched in 2009. 4G enabled more robust network coverage, with higher data speeds than 3G offered. The evolved-UTRAN (E-UTRAN) in 4G improved on the 3G UTRAN system. The 4G cellular networks evolved into 4G LTE, which provides better mobile broadband support than its predecessor, 3G/ 3.5G, and 4G. It was the fastest developing system of mobile communications technology. The theoretical peak data rate for 4G LTE is up to 300 Mbps. Commercial 4G service providers offered data download speeds from 24 Mbps to 36 Mbps, while the upload speeds measure between 7 Mbps and 17 Mbps [6]. With the increase of throughput, customers could access the internet, instant message (IM), stream media, and make live video calls. A fact sheet by Pew Research Center shows that mobile devices started to become more popular around the time 4G LTE was introduced [7].

By the time fifth generation (5G) rolled out in 2019, it was advertised with the capabilities of enhanced mobile broadband (eMBB), ultra-reliable low latency (URLLC), and massive Machine Type Communications (mMTC). Those features enabled higher speeds and higher connection density than 4G LTE. 5G offers peak data rates scale up to 10 Gbps, latency as low as one millisecond, and 99.999% reliability [8]. The increase in 5G speeds and lower latency has presented new ways to take advantage of the cellular spectrum.

5G mobile broadband allows for a high data rate in dense coverage areas, such as military installations, university campuses, and sports stadiums. In addition, URLLC and mMTC enable uses of the cellular spectrum that could not be explored in 4G. For example, URLLC improved latency that could be used in a command and control (C2) situation

where timely information is required for critical decision making—such as giving the command to fire a missile from an unmanned aerial vehicle (UAV), unmanned ground vehicle (UGV), or unmanned surface vessel (USV). As the name implies, mMTC enables machine-to-machine communication such that multiple UAVs, UGVs, and USVs could communicate using the 5G cellular spectrum.

5G is still in the deployment phase within the commercial sector, where it is becoming more available through different parts of the United States. There are two approaches to the 5G deployment: non-standalone (NSA) and standalone (SA). NSA uses the current 4G LTE core to handle the user-plane functions, whereas SA uses all 5G NR equipment to handle both user-plane and control-plane functionalities of the 5G network. The testbed we developed as part of this research is the non-standalone deployment model.

2. Air Interface Stack

For 4G and 5G, the air interface protocol stack defines the radio transmission between the UE and the base station (eNodeB or gNodeB). According to the Telecommunications Engineer's Reference, air interface "provides the physical link between the mobile [device] and the network" [9]. The layers of the air interface protocol stack are the physical (PHY) layer, medium access control (MAC) layer, radio link control (RLC) layer, and the packet data convergence protocol (PDCP) layer. Figure 1 illustrates the architecture of the air interface protocol stack.

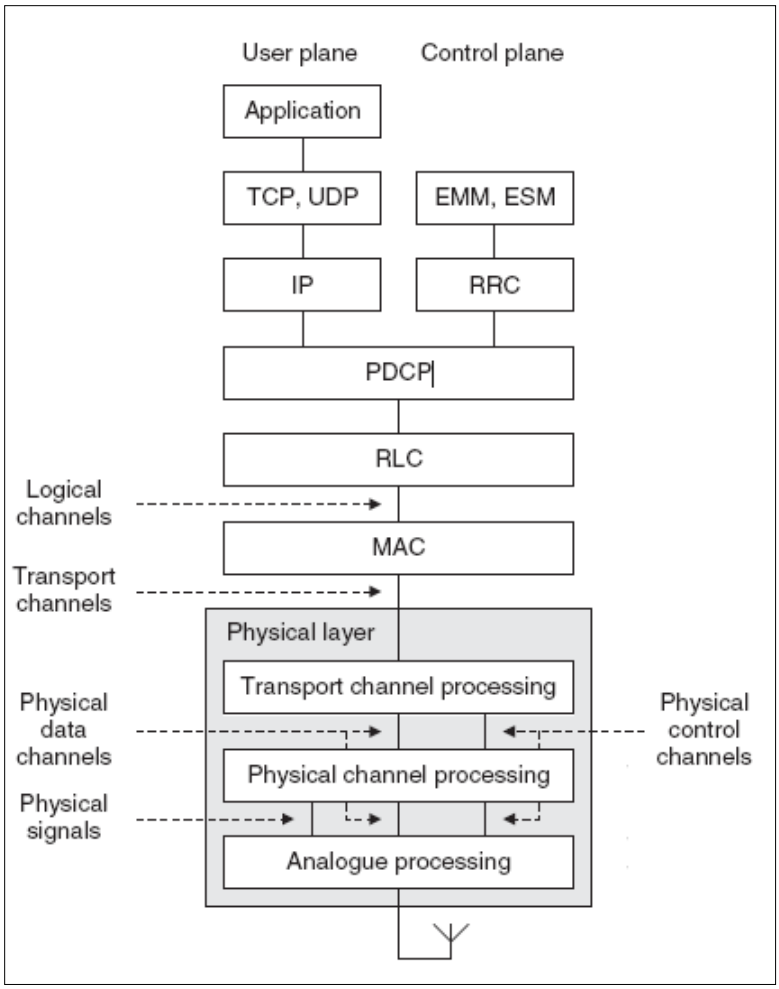


Figure 1. The Architecture of the Air Interface Protocol Stack. Adapted from [10].

a. Control Plane

The control plane manages signaling messages that are critical for UE connectivity. Control transmissions messages in the network originate from the Radio Resource Control (RRC) layer of the control plane or the Mobility Management Entity (MME). The RRC layer controls communications between the eNB and the UE. The MME is responsible for the initial paging procedures and authentication of UEs requesting access to the network. The RRC also transfers Non-Access Stratum (NAS) messages used to control communications between the UE and the MME [11]. The UE attachment to the MME is handled by a NAS protocol, the Evolved Packet System (EPS) mobility management

(EMM) protocol. The EPS Session Management (ESM) protocol supports establishing and managing the user data in the NAS. ESM protocol also defines the IP connectivity between a UE and a packet data network (PDN) [12]. Figure 2 illustrates the RAN protocol architecture. The EMM and ESM are handled by the NAS protocol and not depicted in Figure 2.

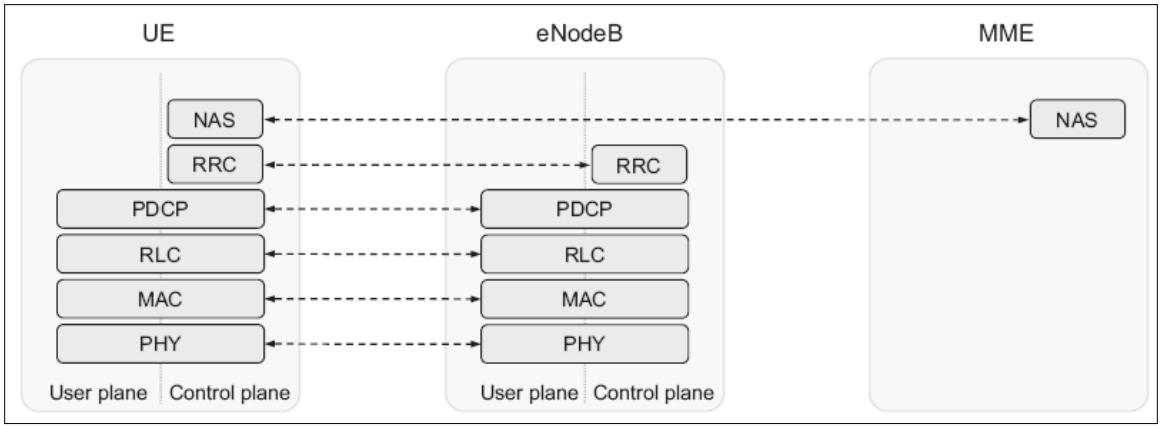


Figure 2. RAN Protocol Architecture. Source [13].

b. User Plane

The user plane manages the exchange of data packets—such as TCP, and UDP packets—produced and consumed by the users or applications. The user plane uses an IP/ UDP-based protocol, GPRS Tunneling Protocol (GTP), to transfer user data within the GSM and LTE networks. GTP is a group of communications protocols that are used for the transfer of GPRS packets within the 2G, 3G, 4G, and 5G standards.

c. PDCP Layer

One main function of the PDCP layer is to use header compression techniques to reduce the redundant IP header overhead within an IP flow. Other functions of the PCDP layer include the transfer of user data, ciphering/deciphering for the Data Radio Bearer (DRB) and Signaling Radio Bearer (SRB), integrity protection for the SRB, maintaining sequence numbers (SN) for SRB and DRB, and time-based SDU discard for SRB and DRB. In addition, the PDCP provides services to the RRC layer and the user plane layer.

d. RLC Layer

The radio link control layer sits above the MAC layer and below the PDCP layer. The RLC functions are performed by an RLC entity configured at the eNB and a peer entity configured at the UE (or vice versa.) Three RLC modes can be configured—Transport Mode (TM), Un-Acknowledge Mode (UM), and Acknowledge Mode (AM). The three modes are further explained in [14]. The RLC layer receives Service Data Units (SDU), frames the SDUs, segments and concatenates the SDUs, and then constructs Protocol Data Units (PDU). An SDU is a packet received by a layer and a PDU is a packet output of a layer. Then, the RLC layer adds an RLC header based on the mode and submits the PDU to the MAC layer.

e. MAC Layer

The MAC layer manages the upper layers' access to the PHY layer. As Figure 1 shows, the MAC layer is connected to the PHY layer through transport channels and the RLC layer through logical channels. The MAC layer determines and performs switching on the logical channels to the transport channels. The MAC layer also multiplexes and demultiplexes the data between both channels.

f. PHY Layer

The PHY layer comprises three parts: the transport channel processor, the physical channel processor, and the analog processor. The PHY layer interfaces with the MAC sublayer (layer 2) and the RLC (layer 3). The PHY offers data transport to the higher layers of the air interface stack.

g. Air Interface Stack Data Flow

Packets (SDU) generated by an application will be received by the IP layer. The IP layer transfers the PDU packets to the PDCP layer. The PDCP layer receives the SDU packets, performs header compression, adds PDCP header information to the packets, and transfers the PDU packets to the RLC layer. The RLC layer segments the SDU packets, adds the mode of operation, and sends the PDU to the MAC layer. The MAC layer receives the SDU packets, adds a MAC header and padding, multiplexes or demultiplexes the SDU

packets, and submits the PDU packets to the PHY layer, which transmits the packets. Figure 3 shows an example of LTE data flow.

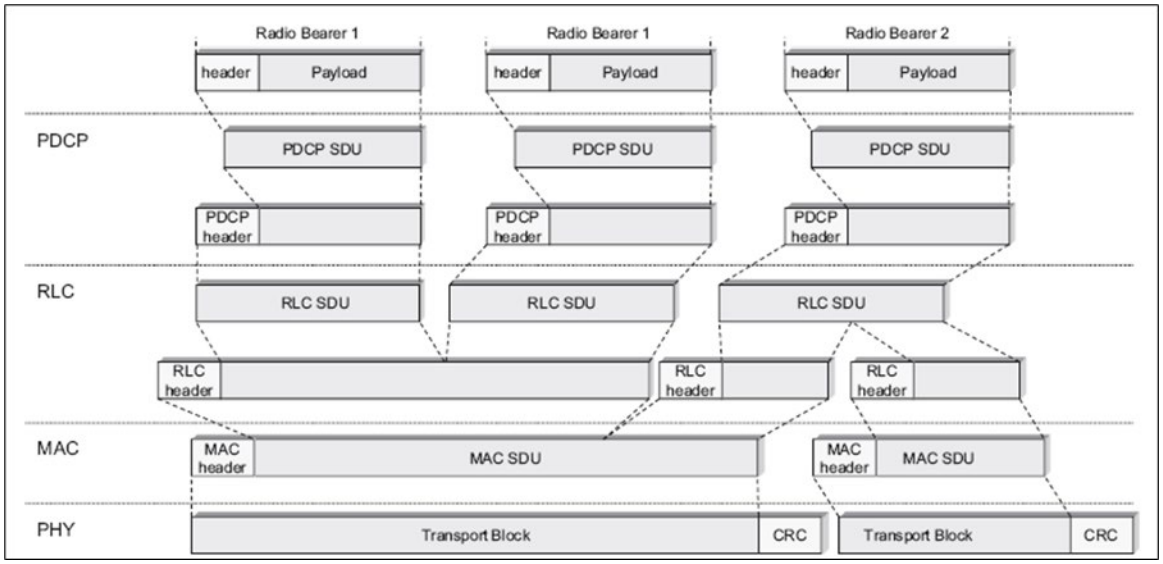


Figure 3. Example of LTE Data Flow. Source [13].

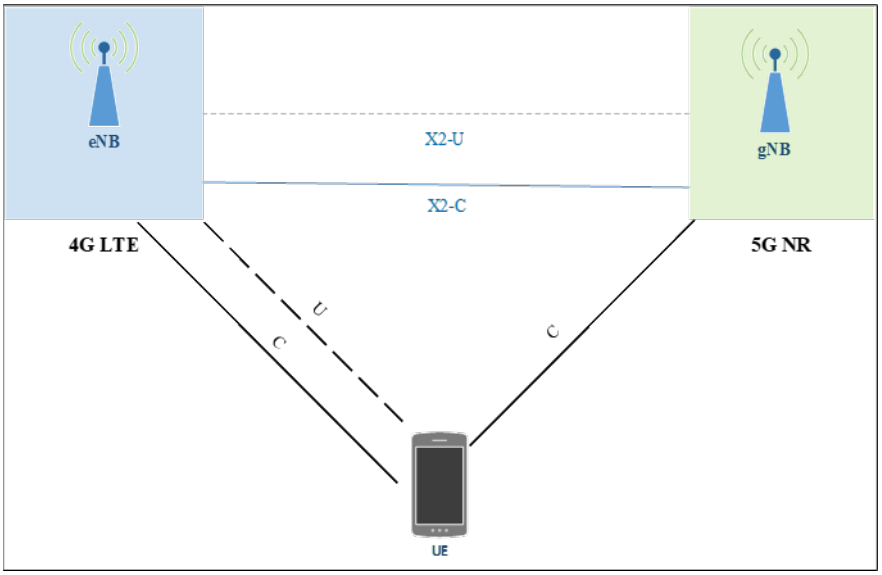
3. Architecture and Equipment

The deployment of 5G is outlined in the 3rd Generation Partnership Project (3GPP) Release 15, which provides a standardized protocol for implementing 5G networks.

a. Architecture and Deployment Models

3GPP Release 15 defines two 5G architecture deployment models: the Stand-Alone (SA) architecture and the NSA architecture. The SA architecture configures and operates the 5G new radio (NR) using only 5G components, whereas the NSA architecture relies on the existing 4G LTE core network. In the NSA architecture, the gNB connects to the eNB via the X2 interface. The eNB is the primary node, while the gNB is the secondary node. This capstone uses the 5G NSA deployment model (option 3). The 5G NSA architecture is the LTE Assisted "Architecture Option 3" [15], shown in Figure 4. 3GPP Option 3 architecture offers dual connectivity through 4G (E-UTRA) and 5G (NR) called E-UTRA

and NR Dual Connectivity (EN-DC) [15]. Other deployment models are outlined in Appendix H.



"c" indicates the control plane (dotted line), and "u" (solid line) indicates the user plane.

Figure 4. 4G E-UTRA and 5G NR Dual Connectivity (3GPP Option 3a). Adapted from [15].

b. Evolved Packet Core

The Evolved Packet Core is the principal component of the radio access network (RAN), with three main network functions to deliver the core of the air interface network. The components on an EPC are a Mobility Management Entity (MME), a Home Subscriber Server (HSS), and Serving Gateways (SGW) [16].

The Service Packet Gateway (SPGW) is separated into two network functions: SPGW-C and SPGW-U. The SPGW-C and SPGW-U are implemented via separated docker containers. The purpose of splitting SPGW functions is to separate the Control and User Plane (CUPS).

The SPGW-C is the control plane of the packet data network gateway that manages the control portion of the Serving Gateway (SGW) and the Packet Network Gateway

(PGW) [16]. In addition, the SPGW-C handles control requests originating from the MME via the S11 interface.

The SPGW-U is the packet data network gateway user plane that manages the user traffic between the PDN and the eNB. The SPGW-U communicates with the eNB via the S1-U interface. The user traffic is tunneled through the GTP, managed by the SPGW-C via the SXab interface.

The HSS stores and updates the database containing the user subscription information. In addition, the HSS is responsible for authentication, authorization, and mobility management functions and connects to the MME via the S6a interface [17].

The MME is the core control function of the EPC. The MME handles the UE access network and mobility as well as establishes a link for the UE. The MME also authenticates the authorization of the UE access to the service provider Public Land Mobile Network (PLMN) [5]. The MME connects to the eNB via the S1-C interface and to the SPGW-C via the S11 interface. Figure 5 illustrates each component of the EPC and the respective interface it uses for communication within the OpenAirInterface architecture.

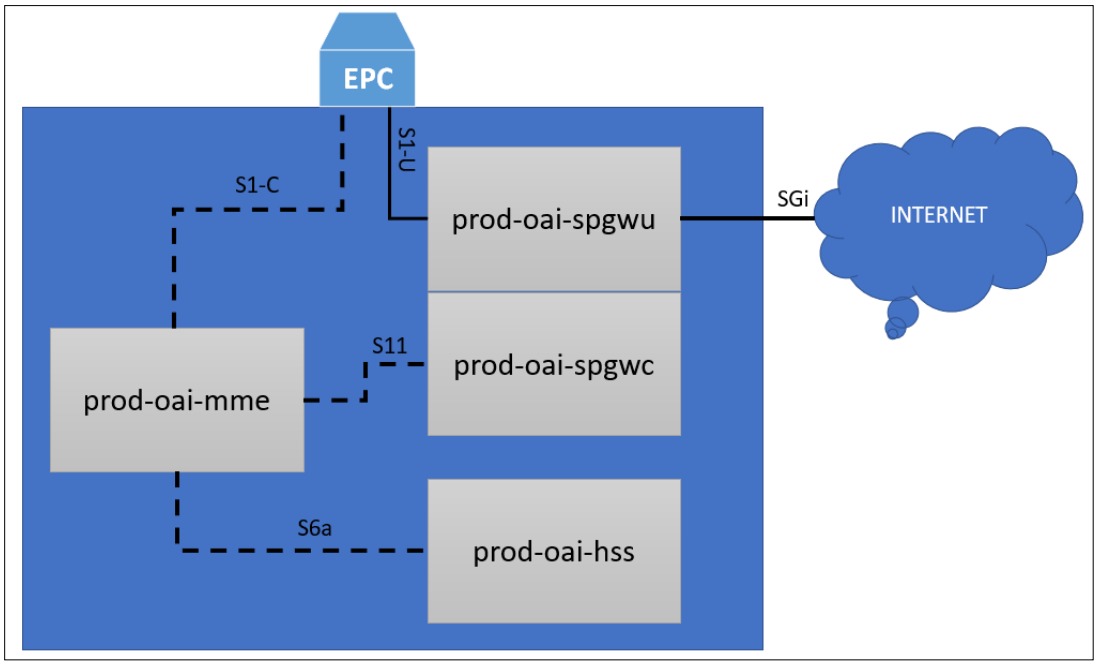


Figure 5. EPC Network Functions

c. eNodeB / gNodeB

The Evolved Node B (eNB) is the 4G LTE base station that creates mobile network coverage and serves LTE devices via the LTE CUPS. The gNodeB is the base station for 5G RAN. The gNB manages NR devices' control and user planes. The eNB connects to the EPC via the S1 interface. In a non-standalone architecture, the gNB connects to the eNB via the X2 interface. In a standalone architecture, the gNB connects to the 5GC via the NG interface.

d. OpenAirInterface

OpenAirInterface is an open-source software developed by EURECOM that implements 3GPP-compliant technology using software to create a 4G and 5G Radio Access Network (RAN) and Core Network (CN) technologies. OpenAirInterface was created by a community of developers worldwide, known as the OpenAirInterface Software Alliance (OSA). The software implements the components of the 4G/5G system and air interface protocol stack using software running on x86 processors and software-defined radios. OpenAirInterface enables researchers and developers the ability to establish and disestablish ad-hoc cellular networks on-demand.

e. Universal Software Radio Peripheral

The 4G/5G architecture uses the Universal Software Radio Peripheral (USRP) that connects to an x86 computing platform through a high-speed cable (USB 3.0). The software-defined radio contains a field-programmable gateway array (FPGA), an integrated circuit chip on the USRP, which enables post-manufacturing reconfigurations by the user. The FPGA is critical for digital signal processing (DSP) because it translates real signals to low-rate, complex, baseband signals for the OpenAirInterface application running on the host computer. The USRP provides the radio frequency (RF) capabilities of the OpenAirInterface architecture.

B. LITERATURE REVIEW

Since 2017, a consortium of developers has implemented various versions of an open-source 4G LTE cellular stack with commercial off-the-shelf (COTS) hardware.

OpenAirInterface released a 5G standalone architecture in early 2021 for implementation; however, it is still under heavy development. There has been various research exploring using the OpenAirInterface 5G NSA software stack. With the exception of [18], none of the literature provided detailed guidance to replicate the testbeds used during research. The lack of available documentation leads to new researchers experiencing the same obstacles when replicating the different testbeds presented in the literature, and the documentation in this work aims to remove some of those obstacles. We also discuss other 4G and 5G open-source consortia in this section.

1. OpenAirInterface—4G LTE

The OpenAirInterface software stack was implemented in [18] to create a 4G LTE architecture. This 4G testbed [18] consisted of an EPC, eNB, USRP, Quectel module, and two attenuators. The attenuators directed transmissions from the Quectel module to the Ettus B200 USRP rather than transmitting over-the-air. Lanoue was able to connect the UE to the internet using the 4G LTE architecture [18]. Internet connectivity was verified by successfully pinging Google's primary DNS server IP address (8.8.8.8) from the UE. The testbed included five different 4G LTE network setups that showcased the ease of configurability of OpenAirInterface.

The work performed in [19] presented an all-in-one LTE network deployment on a PC using OpenAirInterface. Nikaein et al. were able to show OpenAirInterface's interoperability with commercial LTE to transmit live-video streams. They were also able to demonstrate the reconfigurability of the OpenAirInterface platform. The researchers discussed whether the OpenAirInterface platform is a suitable LTE ecosystem and playground [19]. This work showed OpenAirInterface's reconfigurability and interoperability with commercial smartphones.

2. OpenAirInterface—5G NSA

The authors in [20] designed a prototype programmable network-in-a-box using OpenAirInterface and COTS devices. The architecture can support a 4G LTE network or a 5G NSA LTE assisted network (3GPP Option 3). The implemented testbed demonstrated a solution for an on-the-fly 4G LTE and 5G NSA network. They also measured the latency

and throughput performance of their network-in-a-box prototype. This research tested the latency of the 5G NSA and observed an average throughput of 22.21 Mbps for the 5G NSA mode. The maximum throughput achieved was 30.7 Mbps. The research was able to determine the throughput and latency using iPerf tests.

The testbed in [16] showcased the architecture of the OpenAirInterface software suite for both the NSA and the SA mode. In the NSA architecture, the 5G network uses an existing 4G network. The 5G NSA architecture accelerated the use of 5G capabilities without upgrading the EPC.

The researchers in [21] discuss the implementation of a 5G Narrowband Internet-of-Things (NB-IoT) using OpenAirInterface and LTEBox. "NB-IoT is a 3GPP standard defined to support mMTC service" [21]. LTEBox is Nokia's version of the EPC. The authors used LTEBox instead of OpenAirInterface as its EPC because OpenAirInterface EPC did not support NB-IoT at the time of their research. [21], detailed its use of COTS hardware and NB-IoT modules to forward sensing data to the internet via its implementation of the 5G network. The installation used three personal computers (PCs): one PC as the EPC, another PC as the eNB, and the third PC to interface with the user UE. They used two Quectel BC95 modules as their UE and included a USRP for signaling.

The author of [22] discussed industry professionals' security concerns and the overall architecture of the 5G standard. The paper encapsulates the research performed on different vulnerabilities that affect the confidentiality, integrity, and availability (CIA triad) of 5G cellular networks. The research identified multiple vulnerabilities that could affect different components of the CIA triad. Some of the vulnerabilities identified included spoofing attacks, Man-in-the-Middle attacks, and denial-of-service (DoS) attacks. In addition, the research explains that a Distributed DoS (DDoS) attack is possible through "overburdening the system with fake requests" [22]. This capstone achieves a form of internal DoS attack, described in Chapter IV, Section B. A spoofing attack is covered in [18]; however, additional vulnerabilities native to 4G and 5G is discussed in [22].

3. Comparison to Prior Work

This research implements a 4G and 5G NSA network using the OpenAirInterface software, two cellular phones, and a Quectel module connection to the network without using attenuators. The networks implemented in [19], [21], and [22] use a combination of different open-source software vice a unitary OpenAirInterface stack.

The UE used in [18] was an OpenAirInterface lte-uesoftmodem and the Quectel module EC20; the lte-uesoftmodem is an OpenAirInterface software version of a UE used to test network services. Additionally, the network configurations implemented in [18] use an attenuator between the SDR and the UE. Our testbed can also use different internet service providers (ISP) for backhaul services without major configuration changes.

Finally, in contrast with prior work, we document additional network configurations, device setup, and challenges experienced during the different network configurations implementation. The procedures used to create the 4G and 5G RAN are concatenated in the appendices. We aim for this capstone to be a single point of reference for future researchers who wish to recreate a 4G or 5G NSA network using OpenAirInterface.

4. OpenAirInterface Competitors

OpenBTS is an open-source application that implements a 3GPP-compliant air interface. OpenBTS uses USRP hardware and Asterisk software to create a low-cost GSM cellular network. Asterisk is an open-source framework that can build communication applications, specifically providing voice over IP (VoIP) capabilities. Any IP connection can serve as a backhaul for the OpenBTS software. The OpenBTS official wiki page states that OpenBTS is only capable of 2G and 3G and does not support inter-Radio Access Technology (RAT) mobility [23].

The FreedomFi approach to a low-cost 5G network differs from the other open-source approaches. FreedomFi sells an x86 proprietary appliance named FreedomFi gateway. The gateway is pre-loaded with an open-source software, Magma that enables the creation of a private LTE and 5G network for about $500 [24]. The FreedomFi gateway

appliance is not required to access the Magma Access Gateway, rather, it is a quick and inexpensive method to establish mobile core network.

Software Radio System RAN (srsRAN), also known as srsLTE, is an open-source 4G and 5G software suite. The software suite runs on off-the-shelf compute and RF hardware and implements an eNB, gNB, and a UE. The SRS UE is a full-stack software that is implemented in C/C++. The eNB can be used in a 4G LTE network and 5G NSA network. The gNB solution for a 5G SA network will be available in the second quarter of 2022 [25].

free5GRAN is a new open-source 5G development framework. It provides a 5G RAN architecture that works in SA mode; however, free5GRAN is still under active development, and is not fully stable. The target audience for free5GRAN are engineers and researchers for testing and development, and beginners to experiment with the main components of its 5G stack. free5GRAN provides an application programming interface (API), [26], to help users understand its code.

The open-source competitors provide different capabilities than OpenAirInterface. For example, OpenAirInterface is a software implementation of the EPC, eNB, and gNB; however, OpenBTS contains most of the GSM stack that enables mobility management, call control, and text messaging [27]. srsRAN is limited because it does not have an implementation of the EPC. Instead, srsRAN integrates with other open-source implementations of the EPC.

FreedomFi offers a hardware device with Magma installed. Magma is an open-source packet core project. Magma provides an alternative implementation of OpenAirInterface EPC. The Magma software provides an access gateway, orchestrator, and federation gateway. The access Gateway implements the EPC, PGW, and Authentication, Authorization, and Accounting (AAA) [28]. The orchestrator is a cloud solution that securely manages and monitors the wireless network. Finally, the federation gateway integrates 3GPP compliant mobile network operator (MNO) components.

III. METHODOLOGY

This chapter covers the specifications of the devices used and the installation of the software required for the USRP, eNB/gNB, and the EPC. This section also describes the procedures used to configure the OpenAirInterface architecture. Section A discusses the installation of the USRP. Section B contains the installation process of the OpenAirInterface eNB/gNB open stack network. Section C discusses the installation procedure used for the OpenAirInterface EPC software stack. Finally, Section D describes the procedures to implement and configure the required hardware and software components for the eNB/gNB OpenAirInterface described in [18].

The test environment in this capstone used HP ProBook 640 laptops with Intel® Core ™ i7-4712MQ processors running at 2.30GHz. The operating system (OS) used on all PCs was Ubuntu 18.04.1 LTS (Bionic Beaver). Three of the four PCs emulate the core components of the 4G and 5G networks. The first PC serves as the EPC, the second PC serves as the eNB, and the third PC serves as the gNB, which is also used to interact with the Quectel module in a single eNB LTE network configuration. The fourth PC is used to capture packets using Wireshark. This testbed used two Ettus B200 USRP SDR, connected to the eNB and gNB via USB. Specifications for the OpenAirInterface software require an Intel architecture-based PC with at least 4 CPU cores.

A. USRP

The Ettus B200 USRP is a low-cost experimental platform developed by Ettus Research Products that provides continuous frequency coverage from 70 MHz – 6 GHz with up to 56 MHz of real-time bandwidth [29]. The USRP supports the USRP Hardware Driver (UHD) software used in [18] and [21] to model GSM base stations. The software used to configure the Ettus B200 is called GNU Radio. GNU Radio is a free and open-source software development toolkit that provides virtual signal handling to employ an SDR [30]. The Ettus B200, along with the OpenAirInterface software (running on a laptop), is used to implement the eNB.

The minimum software recommended for use with the Ettus B200 USRP is Ubuntu 18.04 with a low-latency kernel. The low-latency kernel is optimized to achieve the lowest latency possible for the OpenAirInterface implementation. To determine if the eNB/gNB PC has the low-latency kernel, the command "uname -r" within Ubuntu's terminal can be executed. The result of the command will show the version number and the kernel header. The eNB/gNB PC kernel header for this experiment is "5.4.0-74-low-latency." The required steps to install and configure a low-latency kernel on Ubuntu 18.04 LTS are described below.

B. ENB/GNB

OpenAirInterface implements the 4G base station, the eNB, of the 4G/5G network design. The eNB is the 4G LTE network base station that performs radio-related network functions and creates the mobile network [31]. The eNB sends and receives radio transmissions to the mobile devices connected to it. The eNB is connected to the EPC through the S1 interface. In the 5G NSA architecture, the eNB performs the user plane and control plane functions for the 4G LTE network and the user plane function for the 5G network. The gNB is the base station that performs the radio-related functions for the 5G network. The gNB performs the control plane functions in the 5G NSA architecture. The steps used to install and configure the eNB and gNB are detailed in Appendix A, Section B. The prerequisite instructions covered in Appendix A, Section A are required before performing the steps in Section B.

C. EPC

The EPC is installed on Docker and deployed via docker containers. This capstone installed "docker-ce/bionic 5:20.10.7~3-0~ubuntu-bionic amd64," "docker-ce-cli/bionic 5:20.10.7~3-0~ubuntu-bionic amd64," and "docker-ce-rootless-extras/bionic 5:20.10.7~3-0~ubuntu-bionic amd64." The steps used to install and configure the EPC are detailed in Appendix A, Section C.

During this research, the EPC was installed two times. The first time, the eNB was successfully able to establish a peer connection with the EPC. The PC that hosted the EPC later experienced a system failure that was unable to be restored. The EPC was

installed a second time, referencing the master branch on OpenAirInterface's GitHub page. The installation procedures were different than the procedures used previously. The GitHub page advertises the installation and configuration as a "Simple Docker Deployment using the OAI Legacy MME" [32]. The new installation method did not require network-specific configuration as the previous method did. The new installation method took approximately 30 minutes, whereas the initial installation method took over three hours. The new installation was successfully installed without any errors; however, the eNB was unable to establish a peer connection with the EPC. After a considerable amount of troubleshooting, we focused on re-installing the EPC in the same way as the initial installation. The installation procedure used to install the EPC the second time is found in Appendix A, Section D.

D. PROVISIONING UNIVERSAL SUBSCRIBER IDENTITY MODULE

Universal Subscriber Identity Module (USIM) provisioning and reprovisioning were performed using the OYEITIMES SIM writer Ver 4.1.5. The OYEITIMES SIM writer program is publicly available for about $150 – $300; however, the OYEITIMES SIM writer program could be obtained by purchasing their SIM Card reader for about $30 – $50. The SIM writer can provision 2G SIM, 3G USIM, 4G USIM, and 5G Integrate SIM (iSIM) cards. In Figure 6, we highlight the configuration made within the OYEITIMES program to configure the USIM card. The other configurations that are not highlighted in Table 1 were the default configurations on the USIM card.

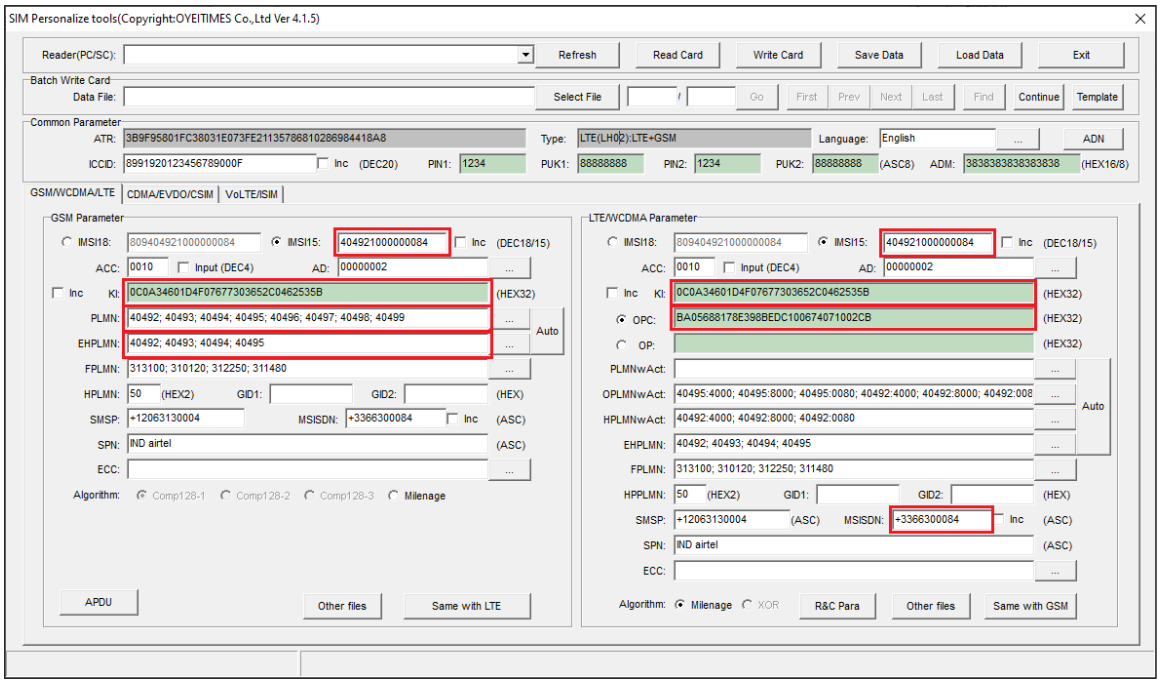


Figure 6. OYEITIMES SIM Writer

Table 1 shows the fields that were changed in this capstone. It also lists the non-abbreviated name, length, and a short description of the fields.

Table 1. Description of USIM Fields Updated in OYEITIMES SIM Writer. Adapted from [33], [34], and [35].

Field	Non-Abbreviated Form	Configuration	Length	Description
IMSI	International Mobile Subscriber Identity	404921000000084	15 bits binary	A unique identifier that identifies a subscriber within a mobile network.
Ki	Key Identification	0C0A34601D4F07677303652C0462535B	32-bits hexadecimal	Used to permit a subscriber to a mobile network.
OP	Operator Code	63BFA50EE6523365FF14C1F45F88737D	32-bits hexadecimal	OP is the same for all SIMs for an operator. Used with Ki to generate an OPc.
OPc	Operator Code (derived from OP)	BA05688178E398BEDC100674071002CB	32-bits hexadecimal	Computed key using the Rijndael Encryption algorithm OPc=Encypt-Algo(OP, Key)
PLMN	Public Land Mobile Network	40492; 40493; 40494; 404045; 40496; 40497; 40498; 40499	5 - 6 digits	A sequence of MCC and MNC that identifies the operators RAN
FPLMN	Forbidden PLMN	313100; 310120; 312250; 311480	5 - 6 digits	A list of forbidden PLMNs the subscriber is not allowed to access.
MSISDN	Mobile Station International Subscriber Directory Number	+3366300084	-	Programmable phone number of the subscriber.
SPN	Service Provider Name	IND airtel	-	Name of the network.

E. SIMULATION

This section describes the simulation of the 5G NSA network. This section explains the environment setup procedures, beginning with the network configuration setup, then running the EPC, connecting the eNB to the EPC, connecting the gNB to the eNB, and finally connecting UEs to the eNB and gNB. Lastly, this section covers tearing down the environment. Parts of this section were automated using multiple bash scripts. The installation methods discussed in Appendix A must be completed before performing any steps in this section.

1. Network Configuration Setup

Setting up the network architecture began with connecting the eNB to the EPC PCs using a network switch. The switch used in this lab is the LINKSYS LGS308. All parameters on the switch were left as default. Static IP addresses were assigned to each PC to ensure that the IP addresses remained the same throughout testing. A ping from the eNB host PC to the EPC host PC was performed to test connectivity between the two PCs. IP routes were then added from the eNB host machine to the docker virtual network bridge and tested by pinging the MME. The same series of steps were performed when the gNB was added to the network. Figure 7 depicts the network architecture.

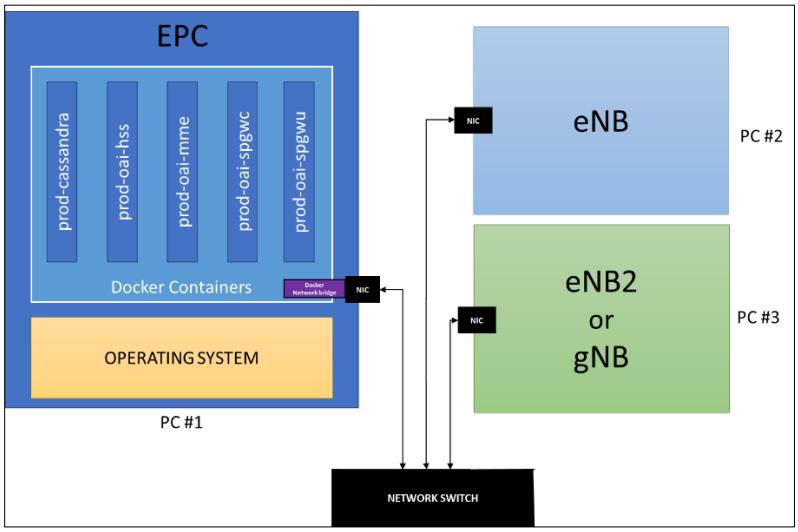


Figure 7. Physical Architecture of the Simulation Lab

In the network configurations that required a second eNB or a gNB, an external timing source was used to synchronize the Ettus B200 for coherent operations. The OctoClock CDA-2900-G was used for timing and frequency reference distribution. The OctoClock synchronizes the base stations to a common timing source that prevents time slippage of the additional eNB or the gNB. It distributes 10 MHz and 1 PPS signal generated from an internal GPS-disciplined, oven-controlled oscillator (GPSDO) that enables time-alignment and multi-channel systems to synchronize within ~50 ns [36].

The "PPS Out" port one on the OctoClock was connected to the eNB Ettus B200 USRP "PPS IN" port using a SubMiniature version A (SMA) cable. Port 2 on the OctoClock was used to connect to the eNB2/gNB Ettus B200 USRP "PPS IN" port. The "Primary Ref" switch was toggled to internal to use the GPSDO timing standard.

2. Deploying the eNB / gNB

The eNB is launched after all the containers on the EPC are running. The Docker commands listed in Appendix D, Section B were used on the EPC host machine to configure and update the containers. Before running the eNB, the routing tables on the EPC host machine were updated to ensure that the eNB host machine could ping the "prod-oai-mme," "prod-oai-spgwc," and the "prod-oai-spgwu-tiny" containers. Attempts to ping the prod-oai-hss were unsuccessful. The installation procedure used in this phase noted that pinging the HSS container "won't work" [37]. Although pinging the HSS container didn't work, it was still accessible using commands listed in Appendix D, Section B.

a. Launch the eNB

To launch the eNB, the following command was executed:

```
cd ~/openairinterface5g/cmake_targets/lte_build_oai/build
sudo -E ./lte-softmodem -O ~/openairinterface5g/ci-scripts/conf_files/
<configuration_file> | tee <log_file_output>
```

Generally, the eNB is launched after the EPC is running. The exception is to launch the eNB without the EPC using the "--noS1" flag. The "--noS1" flag enables the eNB to test connectivity to the UE without any support for the S1 interface. The "-E" flag ensures

that the environmental variables are preserved while running the eNB. The "-O" flag indicates the correct path and filename of the edited configuration file. The "tee <log_file_output>" command is a UNIX command that copies the standard input to the standard output and makes a copy to a file. Additional flags that could be used while executing the eNB can be found in Appendix B, Section C.

b. Launch the gNB

The gNB was executed using the command as shown:

```
cd ~/openairinterface5g/cmake_targets/ran_build/build
sudo ./nr-softmodem -O ~/openairinterface5g/ci-scripts/conf_files/ <configuration_file> -E | tee <log_file_output>
```

The gNB is launched after the eNB is running. The "-E" flag in the gNB is "required to enable a tri-quarter sampling rate when using the B2xx series USRP" [38] . The "-O" flag is used for the same purpose as the eNB. Additional flags that could be used while executing the gNB can be found in Appendix B, Section C.

3. Connecting the User Equipment

There were two types of UEs used in this research. The first is the EC20 Quectel Module. The second were COTS cellphones. This research used the OnePlus 6T and the Google Pixel 4a 5G cellular phones. The sections below detail the process used to send AT commands to the EC20 Quectel Module and collect air interface traffic. The use of QNavigator is introduced next. Lastly, this section describes introducing the OnePlus 6T and the Google Pixel 4a 5G to the 4G network configuration.

a. Android ONEPLUS 6T

One mobile device used was the OnePlus 6T. Most Android devices will have a similar configuration setup since they both use the Android OS. Our phone was running Android version 8.1.0. The One Plus 6T has two sim slots; however, only one was used for this research.

A new Access Point Name (APN) was created by navigating to settings and then selecting SIM & Network > SIM 1 (NPS 4G) > Access Point Names. The APN was then

added by clicking the "+" sign on the top right corner to add a new APN. The configuration settings shown in Table 2 are the APN settings used on the OnePlus 6T in this environment. After configuring and saving the new APN, the phone was connected to the 4G network.

Table 2. OnePlus 6T APN Settings

NAME	apn1.carrier.com
APN	apn1.carrier.com
Proxy	Not set
Port	Not set
Username	Not set
Password	Not set
Server	Not set
MMS proxy	Not set
MMS port	Not set
MCC	404
MNC	92
Authentication type	Not set
APN type	default
APN protocol	IPv4/IPv6
APN roaming protocol	IPv4/IPv6
Bearer	LTE
MVNO type	None

The OnePlus 6T was connected to the network by navigating to SIM & Network > SIM 1 (NPS 4G) > Network operators. The option to "Choose automatically" was turned off and the 4G "IND Airtel" RAN was selected.

b. Google pixel 4a 5G

The second device used in the lab environment was the Google Pixel 4a 5G. The setup was similar to the OnePlus, with a few variations to the settings navigation path. First, the APN was created on the Google Pixel 4a 5G by navigating to settings > Networking and Internet > Mobile network > Advanced > Access Point Names. Next, the hamburger menu on the top right was selected, followed by New APN. After configuring and saving the new APN, the phone was connected to the 4G network by navigating to settings > Networking and Internet > Mobile network > Advanced > Choose network and

selecting the 4G RAN network. The configuration settings shown in Table 3 are the APN settings used on the Google Pixel 4a 5G in this environment.

Table 3. Google Pixel 4a 5G APN Settings

Name	apn1.carrier.com
APN	apn1.carrier.com
Proxy	Not set
Port	Not set
Username	Not set
Password	Not set
Server	Not set
MMSC	Not set
MMS proxy	Not set
MMS port	Not set
MCC	404
MNC	92
Authentication type	Not set
APN type	default,mms,supl,hipri,fota,ims,cbs,xcap
APN protocol	IPv4/IPv6
APN roaming protocol	IPv4/IPv6
Bearer	Unspecified
MVNO type	None

c. Interacting with the Quectel Module Using QNavigator

QNavigator is a Windows-based application used to interact with the Quectel module. QNavigator offers a graphical user interface to connect to the modem and to execute AT commands. Figure 8 displays the home tab of the QNavigator software showing information that could be queried using AT commands. This research used QNavigator V1.6.9.1. QNavigator was used to test the modem connection to the network and test internet connectivity. Connectivity was verified by pinging Cloudflare's primary DNS IP address, and Google's primary DNS IP address. [18] discusses common commands to interact with the Quectel module using QNavigator.

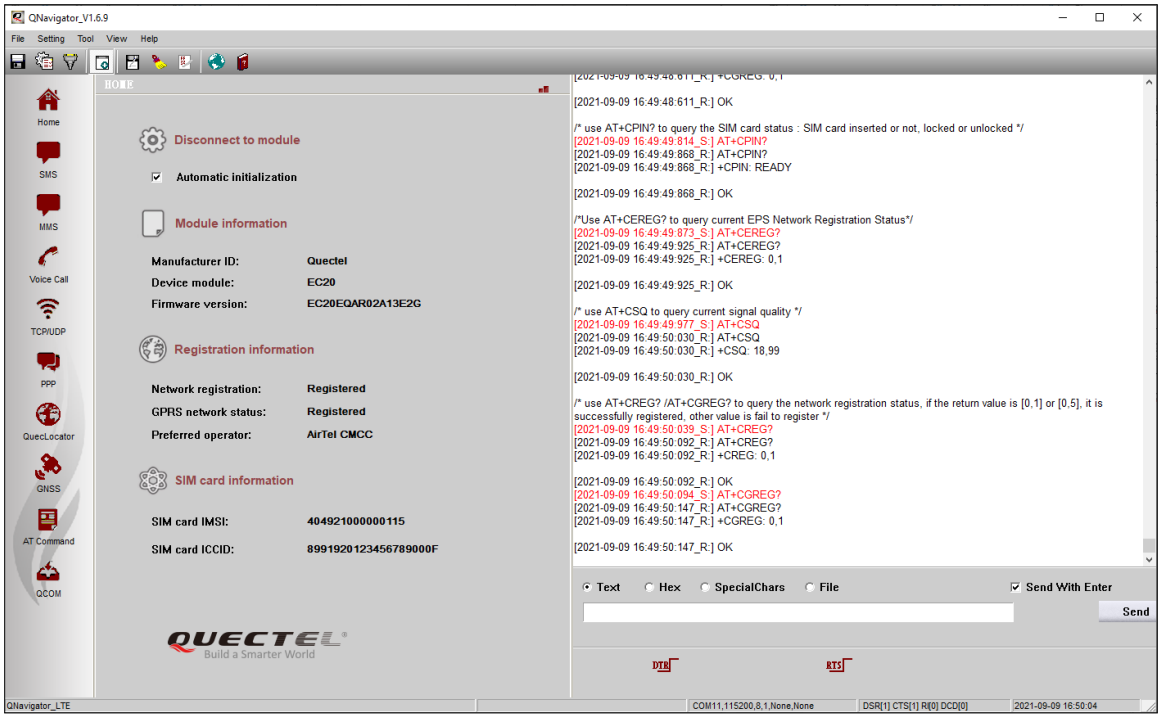


Figure 8. Quectel QNavigator V1.6.9—Home Tab

d. Collect Cellular Network Air Interface Traffic and Interface with the Quectel Module using Minicom.

Collecting cellular network air interface traffic was performed in two different ways—the first method required a several steps. To simplify the process, a script was created to launch Wireshark, launch minicom, and enter configurations to set up the minicom terminal. Minicom is a text-based modem control and terminal emulator connected to a device through GNU/Linux PC serial ports. A keyboard and mouse emulator, "xdotool," was installed on the host PC to configure Minicom using the script. The scripts were configured on a PC that was not the EPC or eNB. The Quectel module is not capable of connecting the 5G network; it is only used for the 4G network configuration. Since the PC that served as the gNB was not used in the 4G network configuration, the gNB PC was also used to configure and install DiagParser, Android Tools, and Minicom to send AT commands to the UE. "DiagParser" is used to decode the Qualcomm DIAG format. Android Tools enable the user to the command-line interface (CLI) with the Quectel Modem. The installation and configuration scripts are found in Appendix C,

Sections A, and B. The common AT commands used in this project are given in Appendix C, Section B; however, the supplemental includes a full list of potential AT commands.

The second method used to collect air interface traffic was Wireshark running on a fourth PC connected to the switch. We enabled port forwarding via ethernet from the ports on the switch that connected the EPC, eNB, and the gNB to the port on the switch to which the PC with Wireshark was connected. The benefit of using this method to capture packets is that it requires fewer steps and does not require configuration when reestablishing the network.

IV. RESULTS

This chapter describes the results of the 4G LTE and the 5G NR NSA network configuration using the OpenAirInterface software. We test three additional network configurations not showcased in [18]. Also, this chapter provides an analysis of the OpenAirInterface platform, different components of the network architecture, network performance, and the network's mobility.

A. NETWORK CONFIGURATIONS

This section discusses the different OpenAirInterface network configurations using three subsections to show the different configurations used during the experimentation phase of this research: 1. EPC and eNB network configuration; 2. EPC and dual eNB network configuration; and 3. EPC, eNB, and gNB network configuration. Figure 9 shows the IP address scheme used in "Network Configuration 1." "Network Configuration 2," and "Network Configuration 3" builds on the "Network Configuration 1" topology.

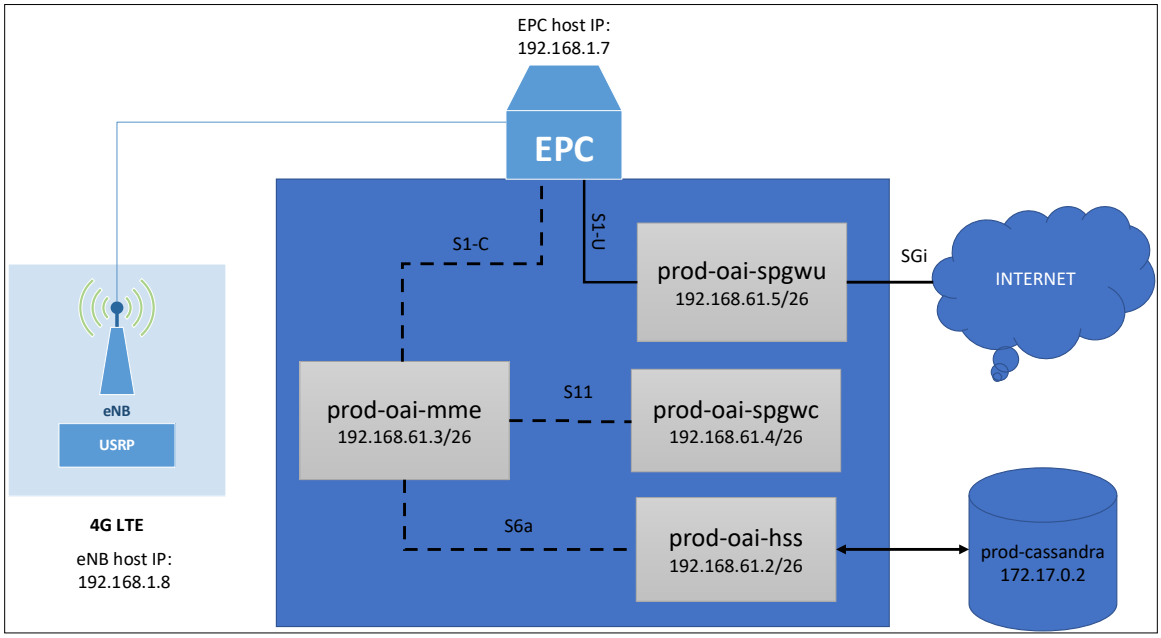


Figure 9. eNB, EPC host, and Docker Containers IP Scheme

1. Network Configuration 1: EPC, eNB, and COTS UE

After installing the OpenAirInterface software for the EPC and eNB, the two PCs were connected to a Linksys LGS308 switch, as shown in Figure 10. The EPC host machine was configured to have a static IP address of 192.168.1.7. The eNB host machine was configured to have a static IP address of 192.168.1.8. The EPC's IP address was used as the gateway to forward the OpenAirInterface network traffic to its respective container.

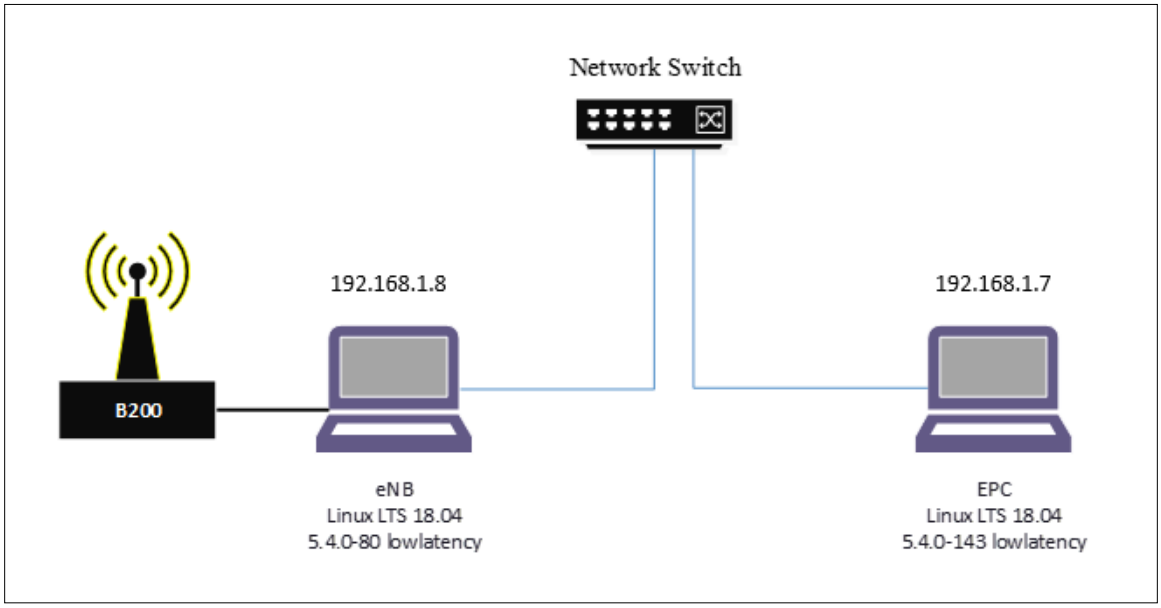


Figure 10. Network Configuration 1

The EPC was launched using the scripts in Appendix A Section B. The scripts launched the docker containers in the required order to run the EPC successfully: 1. "oai-hss:production"; 2. "oai-mme:production"; 3. "oai-spgwc:production"; and 4. oai-spgwu-tiny:production. The last command in the script to execute follows the mme_check_run.log. Following the "mme_check_run.log" shows verbose details of the processes that are running related to the OpenAirInterface cellular network. The "tail" flag within the script enables the contents of the "mme_check_run.log" to display and is updated every five seconds. Figure 11 depicts a snapshot of the "mme_check_run.log" log

that shows the connection status of eNB and attached UEs. The “mme_check_run.log” can be found within the MME docker container (“oai-mme:production”).

```
000294 00001:851655 7F3BD0FF9700 DEBUG S6A    /openair-mme/src/s6a/s6a_peer.c :0093    Diameter identity of MME: mme.openairinterface.org with length: 24
000295 00001:851728 7F3BD0FF9700 DEBUG S6A    /openair-mme/src/s6a/s6a_peer.c :0127    S6a peer connection attempt 1 / 8
000296 00001:851758 7F3BD0FF9700 DEBUG S6A    /openair-mme/src/s6a/s6a_peer.c :0141    Peer hss.openairinterface.org is now connected...
000297 00001:852011 7F3C290DC700 DEBUG SCTP   rc/sctp/sctp_primitives_server.c:0705    Received SCTP_INIT_MSG
000298 00001:852059 7F3C290DC700 DEBUG SCTP   rc/sctp/sctp_primitives_server.c:0333    Creating new listen socket on port 36412 with
000299 00001:852073 7F3C290DC700 DEBUG SCTP   rc/sctp/sctp_primitives_server.c:0336    ipv4 addresses:
000300 00001:852088 7F3C290DC700 DEBUG SCTP   rc/sctp/sctp_primitives_server.c:0344     - 192.168.61.3
000301 00001:852099 7F3C290DC700 DEBUG SCTP   rc/sctp/sctp_primitives_server.c:0349    ipv6 addresses:
000302 00010:791450 7F3C22FAB700 DEBUG MME-AP src/mme_app/mme_app_statistics.c:0039    ======================================= STATISTICS ==========================================

000303 00010:791533 7F3C22FAB700 DEBUG MME-AP src/mme_app/mme_app_statistics.c:0042                    |   Current Status| Added since last display|  Removed since last display |
000304 00010:791555 7F3C22FAB700 DEBUG MME-AP src/mme_app/mme_app_statistics.c:0048    Connected eNBs |          0      |              0              |              0              |
000305 00010:791569 7F3C22FAB700 DEBUG MME-AP src/mme_app/mme_app_statistics.c:0054    Attached UEs   |          0      |              0              |              0              |
000306 00010:791582 7F3C22FAB700 DEBUG MME-AP src/mme_app/mme_app_statistics.c:0060    Connected UEs  |          0      |              0              |              0              |
000307 00010:791594 7F3C22FAB700 DEBUG MME-AP src/mme_app/mme_app_statistics.c:0066    Default Bearers|          0      |              0              |              0              |
000308 00010:791605 7F3C22FAB700 DEBUG MME-AP src/mme_app/mme_app_statistics.c:0072    S1-U Bearers   |          0      |              0              |              0              |

000309 00010:791616 7F3C22FAB700 DEBUG MME-AP src/mme_app/mme_app_statistics.c:0075    ======================================= STATISTICS ==========================================
```

Figure 11. MME Log

The eNB is launched after all the docker containers on the EPC host are running. The command used to launch the eNB is shown in Chapter III.E.1.a. After the command to launch the eNB is executed in the Linux terminal, a new connection is displayed on the MME Log. Figure 12 displays the established peer connection as well as debugging information.

```
▯[0m[0;32m006735 02851:437858 7F3FECFF9700 DEBUG SCTP   rc/sctp/sctp_primitives_server.c:0469    Client association changed: 0
▯[0m[0;32m006736 02851:437941 7F3FECFF9700 DEBUG SCTP   enair-mme/src/sctp/sctp_common.c:0101    ----------------------
▯[0m[0;32m006737 02851:437996 7F3FECFF9700 DEBUG SCTP   enair-mme/src/sctp/sctp_common.c:0102    SCTP Status:
▯[0m[0;32m006738 02851:438019 7F3FECFF9700 DEBUG SCTP   enair-mme/src/sctp/sctp_common.c:0103    assoc id .....: 84
▯[0m[0;32m006739 02851:438035 7F3FECFF9700 DEBUG SCTP   enair-mme/src/sctp/sctp_common.c:0104    state ........: 4
▯[0m[0;32m006740 02851:438050 7F3FECFF9700 DEBUG SCTP   enair-mme/src/sctp/sctp_common.c:0105    instrms ......: 2
▯[0m[0;32m006741 02851:438066 7F3FECFF9700 DEBUG SCTP   enair-mme/src/sctp/sctp_common.c:0106    outstrms .....: 2
▯[0m[0;32m006742 02851:438080 7F3FECFF9700 DEBUG SCTP   enair-mme/src/sctp/sctp_common.c:0108    fragmentation : 1452
▯[0m[0;32m006743 02851:438095 7F3FECFF9700 DEBUG SCTP   enair-mme/src/sctp/sctp_common.c:0109    pending data .: 0
▯[0m[0;32m006744 02851:438110 7F3FECFF9700 DEBUG SCTP   enair-mme/src/sctp/sctp_common.c:0110    unack data ...: 0
▯[0m[0;32m006745 02851:438125 7F3FECFF9700 DEBUG SCTP   enair-mme/src/sctp/sctp_common.c:0111    rwnd .........: 106496
▯[0m[0;32m006746 02851:438139 7F3FECFF9700 DEBUG SCTP   enair-mme/src/sctp/sctp_common.c:0112    peer info     :
▯[0m[0;32m006747 02851:438154 7F3FECFF9700 DEBUG SCTP   enair-mme/src/sctp/sctp_common.c:0114        state ....: 2
▯[0m[0;32m006748 02851:438169 7F3FECFF9700 DEBUG SCTP   enair-mme/src/sctp/sctp_common.c:0116        cwnd .....: 4380
▯[0m[0;32m006749 02851:438184 7F3FECFF9700 DEBUG SCTP   enair-mme/src/sctp/sctp_common.c:0118        srtt .....: 0
▯[0m[0;32m006750 02851:438199 7F3FECFF9700 DEBUG SCTP   enair-mme/src/sctp/sctp_common.c:0120        rto ......: 3000
▯[0m[0;32m006751 02851:438213 7F3FECFF9700 DEBUG SCTP   enair-mme/src/sctp/sctp_common.c:0122        mtu ......: 1500
▯[0m[0;32m006752 02851:438228 7F3FECFF9700 DEBUG SCTP   enair-mme/src/sctp/sctp_common.c:0123    ----------------------
▯[0m[0;32m006753 02851:438243 7F3FECFF9700 DEBUG SCTP   rc/sctp/sctp_primitives_server.c:0479    New connection
▯[0m[0;32m006754 02851:438270 7F3FECFF9700 DEBUG SCTP   enair-mme/src/sctp/sctp_common.c:0205    ----------------------
▯[0m[0;32m006755 02851:438285 7F3FECFF9700 DEBUG SCTP   enair-mme/src/sctp/sctp_common.c:0206    Local addresses:
▯[0m[0;32m006756 02851:438304 7F3FECFF9700 DEBUG SCTP   enair-mme/src/sctp/sctp_common.c:0217        - [192.168.61.3]
▯[0m[0;32m006757 02851:438320 7F3FECFF9700 DEBUG SCTP   enair-mme/src/sctp/sctp_common.c:0234    ----------------------
▯[0m[0;32m006758 02851:438339 7F3FECFF9700 DEBUG SCTP   enair-mme/src/sctp/sctp_common.c:0151    ----------------------
▯[0m[0;32m006759 02851:438354 7F3FECFF9700 DEBUG SCTP   enair-mme/src/sctp/sctp_common.c:0152    Peer addresses:
▯[0m[0;32m006760 02851:438370 7F3FECFF9700 DEBUG SCTP   enair-mme/src/sctp/sctp_common.c:0163        - [192.168.1.8]
▯[0m[0;32m006761 02851:438385 7F3FECFF9700 DEBUG SCTP   enair-mme/src/sctp/sctp_common.c:0178    ----------------------
▯[0m[0;32m006762 02851:438433 7F3FECFF9700 DEBUG SCTP   rc/sctp/sctp_primitives_server.c:0554    SCTP RETURNING!!
▯[0m[0;32m006763 02851:438468 7F3FECFF9700 DEBUG SCTP   rc/sctp/sctp_primitives_server.c:0547    [84][50] Msg of length 59 received from port 36412, on stream 0, PPID 18
▯[0m[0;32m006764 02851:438494 7F3FECFF9700 DEBUG SCTP   rc/sctp/sctp_primitives_server.c:0554    SCTP RETURNING!!
▯[0m[0;32m006765 02851:438513 7F40597FA700 DEBUG S1AP   mme/src/s1ap/s1ap_mme_handlers.c:2826    Create eNB context for assoc_id: 84
▯[0m[0;32m006767 02851:438835 7F40597FA700 DEBUG S1AP   mme/src/s1ap/s1ap_mme_handlers.c:0361    S1-Setup-Request macroENB_ID.size 3 (should be 20)
▯[0m[0;32m006766 02851:438823 7F40597FA700 DEBUG S1AP   mme/src/s1ap/s1ap_mme_handlers.c:0321    New s1 setup request incoming from macro eNB id: 00e01▯[0m
▯[0;32m006768 02851:438878 7F40597FA700 DEBUG S1AP   mme/src/s1ap/s1ap_mme_handlers.c:0423    Adding eNB to the list of served eNBs
▯[0m[0;32m006769 02851:438897 7F40597FA700 DEBUG S1AP   mme/src/s1ap/s1ap_mme_handlers.c:0438    Adding eNB id 3585 to the list of served eNBs
▯[0m[0;32m006770 02851:439070 7F405AFFD700 DEBUG SCTP   rc/sctp/sctp_primitives_server.c:0283    [50][84] Sending buffer 0x7f3fcc034180 of 27 bytes on stream 0 with ppid 18
▯[0m[0;32m006771 02851:439234 7F405AFFD700 DEBUG SCTP   rc/sctp/sctp_primitives_server.c:0296    Successfully sent 27 bytes on stream 0
▯[0m
▯[0;32m006772 02860:478425 7F4058FF9700 DEBUG MME-AP src/mme_app/mme_app_statistics.c:0039    ======================================= STATISTICS ==========================================

▯[0m[0;32m006773 02860:478500 7F4058FF9700 DEBUG MME-AP src/mme_app/mme_app_statistics.c:0042                    |   Current Status| Added since last display|  Removed since last display |
▯[0m[0;32m006774 02860:478520 7F4058FF9700 DEBUG MME-AP src/mme_app/mme_app_statistics.c:0048    Connected eNBs |          1      |              1              |              0              |
▯[0m[0;32m006775 02860:478570 7F4058FF9700 DEBUG MME-AP src/mme_app/mme_app_statistics.c:0054    Attached UEs   |          0      |              0              |              0              |
▯[0m[0;32m006776 02860:478595 7F4058FF9700 DEBUG MME-AP src/mme_app/mme_app_statistics.c:0060    Connected UEs  |          0      |              0              |              0              |
▯[0m[0;32m006777 02860:478609 7F4058FF9700 DEBUG MME-AP src/mme_app/mme_app_statistics.c:0066    Default Bearers|          0      |              0              |              0              |
▯[0m[0;32m006778 02860:478622 7F4058FF9700 DEBUG MME-AP src/mme_app/mme_app_statistics.c:0072    S1-U Bearers   |          0      |              0              |              0              |

▯[0m[0;32m006779 02860:478635 7F4058FF9700 DEBUG MME-AP src/mme_app/mme_app_statistics.c:0075    ======================================= STATISTICS ==========================================
```

Figure 12. MME Log Showing eNB Peer Connection

The next step was to connect the UE, in this case, the OnePlus 6T, to the network. After a UE is introduced to the network, mme_check_run.log shows the number of "Connected UEs" and the number of "Attached UEs" in the "Current Status" column. The UE first connects to the eNB and then performs the LTE attach procedure. After the UE is attached to the EPC, the UE is in a state where the default EPS bearer enables IP connectivity for that UE [12]. After completing the attach procedure, the MME log is updated to display the "Connected eNBs," "Attached UEs," and "Connected UEs," as displayed in Figure 13.

```
===================================== STATISTICS ==========================================

                |   Current Status| Added since last display|  Removed since last display |
Connected eNBs  |          1      |              0             |             0              |
Attached UEs    |          1      |              0             |             0              |
Connected UEs   |          1      |              0             |             0              |
Default Bearers|          0      |              0             |             0              |
S1-U Bearers    |          0      |              0             |             0              |
```

Figure 13. Snapshot of MME Log Showing eNB Peer Connection and UE Attached

2. Network Configuration 2: EPC, eNB1, eNB2, and COTS UE

The dual eNB network configuration is similar to the single eNB network setup. The second eNB is connected using the same process as the first eNB. A configuration change is required to update the "eNB_ID" field on the template configuration file. If the MME identifies that it already has a peer connection with an eNB_ID, the MME will reject any additional eNB attempting to establish a peer connection with that eNB_ID. Additional configuration changes are outlined in Appendix A, Section B, Section 2.

The IP scheme of the dual eNB configuration tested in this research is the same as the single eNB IP scheme, apart from the second eNB. Figure 14 illustrates the IP scheme used in "Network Configuration 2." The IP addresses of the Docker containers remain the same, as shown in Figure 9.

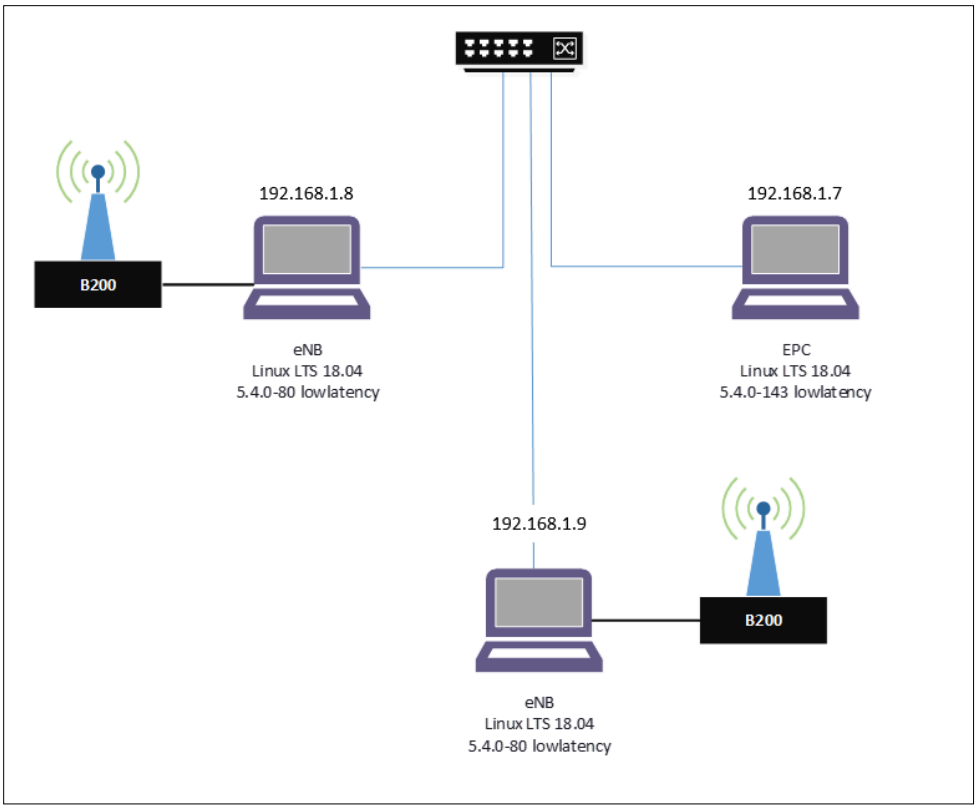


Figure 14. Dual eNB and EPC Network Scheme

The second EPC followed the same connection procedures depicted in Figure 11 and Figure 12. Figure 15 below is a snapshot of the MME log in a dual eNB configuration.

```
===================================== STATISTICS ==========================================

                |   Current Status| Added since last display|  Removed since last display |
Connected eNBs  |          2      |              1             |             0              |
Attached UEs    |          0      |              0             |             0              |
Connected UEs   |          0      |              0             |             0              |
Default Bearers|          0      |              0             |             0              |
S1-U Bearers    |          0      |              0             |             0              |

===================================== STATISTICS ==========================================
```

Figure 15. Snapshot of MME Log Showing Connected eNBs

After both eNBs established peer connectivity with the EPC, the Google Pixel 4a 5G was introduced to the network. The MME log shows the addition in the “Current Status” column in Figure 16. The “Added since last display” column display is also updated.

```
===================================== STATISTICS ==========================================

                |   Current Status| Added since last display|  Removed since last display |
Connected eNBs  |          2      |              0          |             0               |
Attached UEs    |          1      |              1          |             0               |
Connected UEs   |          1      |              1          |             0               |
Default Bearers|          0      |              0          |             0               |
S1-U Bearers    |          0      |              0          |             0               |

===================================== STATISTICS ==========================================
```

Figure 16. Snapshot of MME Log Connected eNBs and Attached UEs

3. Network Configuration 3: EPC, eNB, gNB, and COTS UE

Introducing the gNB to the single eNB network configuration was the next step in configuring the 5G NSA architecture. The procedures to establish the 5G NSA configuration are first to launch the EPC, then the eNB, and finally the gNB. Figure 17 shows the IP scheme used in "Network Configuration 3."

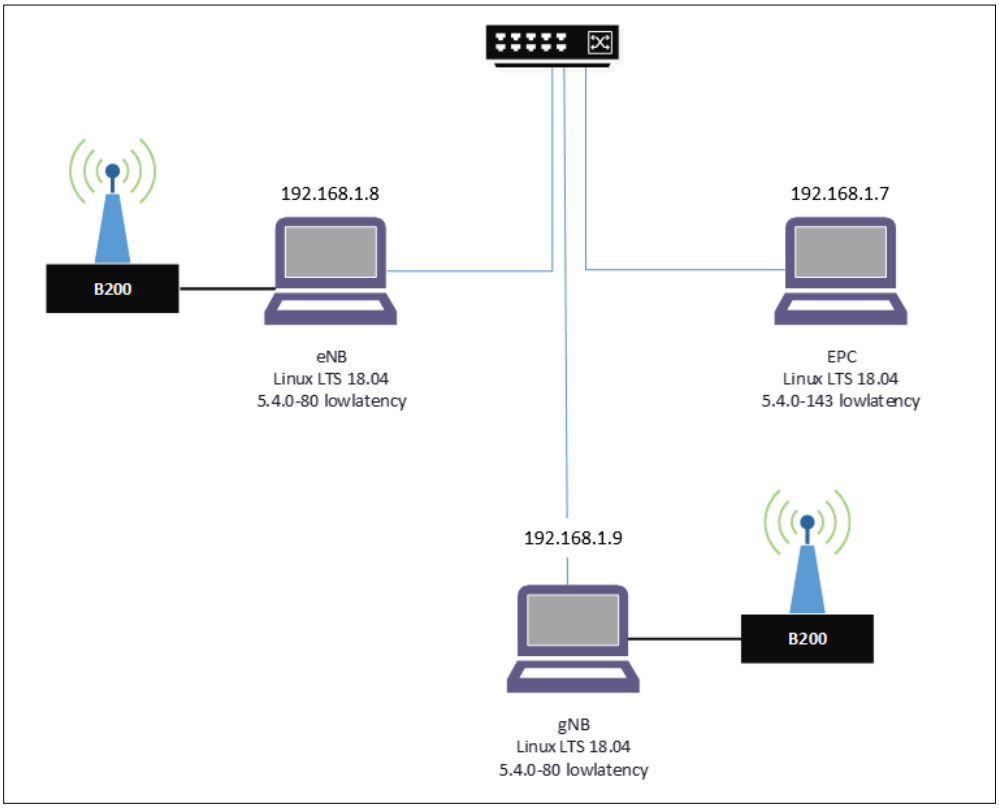


Figure 17. eNB, gNB, and EPC IP NSA Network Scheme

Unlike “Network Configuration 1,” in the 5G NSA architecture, the gNB establishes a peer connection with the eNB. When the gNB is connected to the eNB, it cannot be seen or viewed on the “mme_check_run.log log.” However, as shown in Figure 18, a successful connection can be observed on the eNB terminal where the command to execute the eNB was performed. As shown, the gNB establishes a connection to the eNB using the stream control transmission protocol (SCTP) and the S1 Application Protocol (S1AP). SCTP is designed to transport cellular signaling messages over IP networks. S1AP provides the control plane signaling between the E-UTRAN and the EPC. [39] [40] were referenced when running the gNB.

```
[0m[SCTP]   sctp_bindx SCTP_BINDX_ADD_ADDR socket bound to : 192.168.1.9
[0m[0m[SCTP]   Converted ipv4 address 192.168.61.3 to network type
[0m[0m[SCTP]   connectx assoc_id  5 in progress..., used 1 addresses
[0m[0m[SCTP]   Inserted new descriptor for sd 96 in list, nb elements 1, assoc_id 5
[0m[0m[SCTP]   Found data for descriptor 96
[0m[0m[SCTP]   Received notification for sd 96, type 32769
[0m[0m[SCTP]   Client association changed: 0
[0m[0m[SCTP]   ----------------------
[0m[0m[SCTP]   Peer addresses:
[0m[0m[SCTP]       - [192.168.61.3]
[0m[0m[SCTP]   ----------------------
[0m[0m[SCTP]   ----------------------
[0m[0m[SCTP]   SCTP Status:
[0m[0m[SCTP]   assoc id .....: 5
[0m[0m[SCTP]   state ........: 4
[0m[0m[SCTP]   instrms ......: 2
[0m[0m[SCTP]   outstrms .....: 2
[0m[0m[SCTP]   fragmentation : 1452
[0m[0m[SCTP]   pending data .: 0
[0m[0m[SCTP]   unack data ...: 0
[0m[0m[SCTP]   rwnd .........: 106496
[0m[0m[SCTP]   peer info     :
[0m[0m[SCTP]       state ....: 2
[0m[0m[SCTP]       cwnd .....: 4380
[0m[0m[SCTP]       srtt .....: 0
[0m[0m[SCTP]       rto ......: 3000
[0m[0m[SCTP]       mtu ......: 1500
[0m[0m[SCTP]   ----------------------
[0m[0m[SCTP]   Comm up notified for sd 96, assigned assoc_id 5
[0m[0m[S1AP]   3590 -> 00e060
[0m[0m[SCTP]   Successfully sent 59 bytes on stream 0 for assoc_id 5
[0m[0m[SCTP]   Found data for descriptor 96
[0m[0m[SCTP]   [5][96] Msg of length 27 received from port 36412, on stream 0, PPID 18
[0m[0m[S1AP]   servedGUMMEIs.list.count 1
[0m[0m[S1AP]   servedPLMNs.list.count 1
[0m[0m[1;31m[S1AP]   S1AP_FIND_PROTOCOLIE_BY_ID: /home/labware/workarea/openairinterface5g-develop/openair3/S1AP/s1ap_eNB_handlers.c 465: ie is NULL
[0m[LOADER] library libcoding.so successfully loaded
```

Figure 18. eNB View of gNB Peer Connection with eNB

A successful connection can also be observed on the gNB terminal that was used to launch the gNB. The gNB uses the X2 Application Protocol (X2AP) to handle UE mobility signaling between the gNB and eNB and uses SCTP to establish a peer connection with the eNB. The communication to establish the 5G NSA network can be viewed in Figure 19.

```
[X2AP]   Registered new eNB[0] and home eNB id 3584
[GNB_APP]   [gNB 0] gNB_app_register for instance 0
[X2AP]   eNB[0] eNB id 3584 acting as a listner (server)
[SCTP]   Received SCTP_INIT_MSG_MULTI_REQ
[SCTP]   Creating new listen socket on port 36422 with
[SCTP]   ipv4 addresses:
[SCTP]          - 192.168.1.9
[PHY]   gNB 0 configured
[MAC]   Getting Total PRACH info from index 98 (col 1 ) absoluteFrequencyPointA 640000 mu 1 frame_type 1 start_symbol 0 N_t_slot 3 N_dur 4 N_RA_sfn = 1
[NR_MAC]   Total available RO 3, num of active SSB 1: unused RO = 0 association_period 1 N_RA_sfn 1 total_prach_occasions_per_config_period 3
[X2AP]   eNB[0] eNB id 3584 acting as an initiator (client)
[NR_RRC]   [FRAME 00000][gNB][MOD 00][RNTI 0] gNB:OPENAIR NR RRC IN....
[SCTP]   Converted ipv4 address 192.168.1.8 to network type
[SCTP]   sctp_connectx SUCCESS, used 1 addresses assoc_id 39
[SCTP]   Inserted new descriptor for sd 96 in list, nb elements 2, assoc_id 39
[SCTP]   Found data for descriptor 96
[SCTP]   Received notification for sd 96, type 32769
[SCTP]   Client association changed: 0
[SCTP]   ----------------------
[SCTP]   Peer addresses:
[SCTP]       - [192.168.1.8]
[SCTP]   ----------------------
[SCTP]   ----------------------
[GTPU]   Initializing UDP for local address 192.168.1.9 with port 2152
[SCTP]   SCTP Status:
[SCTP]   assoc id .....: 39
[SCTP]   state ........: 4
[SCTP]   instrms ......: 32
[SCTP]   outstrms .....: 32
[SCTP]   fragmentation : 1452
[SCTP]   pending data .: 0
[SCTP]   unack data ...: 0
[SCTP]   rwnd .........: 106496
[SCTP]   peer info     :
[SCTP]       state ....: 2
[SCTP]       cwnd .....: 4380
[SCTP]       srtt .....: 0
[SCTP]       rto ......: 3000
[SCTP]       mtu ......: 1500
[SCTP]   ----------------------
[SCTP]   Comm up notified for sd 96, assigned assoc_id 39
here comes the tree (instance 0):
```

Figure 19. gNB View of Peer Connection with eNB

B. ANALYSIS

This section examines each component of the OpenAirInterface software stack used in this capstone. The analysis performed in this section is derived from the various challenges experienced during this research's installation and simulation phases.

1. OpenAirInterface Platform

The OpenAirInterface website states that it is "the fastest-growing community and software asset in 5G wireless" [41]. The OpenAirInterface Alliance has official GitHub pages where the master repository is available for developers and researchers. The alliance allows researchers and developers to branch the master repository and submit a pull request so that the researchers and developers can contribute towards identifying and removing bugs that are present within the software. The bugs can be found and fixed by researchers and developers in their respective branches. However, the bug can still exist in the master branch of the software until the updated branch has been verified and merged with the

master branch or the master branch has been updated to reflect the changes made in the developer's branch.

The OpenAirInterface publishes a charter with project timelines for adding functionality to the software. However, the timelines are an approximation and unreliable as they serve as more of a goal date than the expected date of the OpenAirInterface software update. There does not appear to be an official due date as to when the software will be updated.

The procedure to install the software varies depending on the branch installed. The master branch contains elaborate procedures that are simple to follow for the installation; however, there were required dependencies that were not listed in the prerequisite portion of their installation page. For example, a dependency required before installing the eNB is freeDiameter, which implements the Diameter base protocol that provides authentication, authorization, and accounting. The freeDiameter protocol is used to enable authentication, authorization, and accounting (AAA) for mobile IPv4 applications within the OpenAirInterface 4G network [31].

The implementation of the OpenAirInterface software stack could be streamlined by maintaining a working repository that is properly configured. However, the initial configuration, changing configurations, maintenance, and troubleshooting of the networks require telecommunications systems and computer networking subject matter experts (SMEs) who can understand each component of the program in depth. In addition, implementing the OpenAirInterface software requires an intermediate level of knowledge to navigate Linux OS through CLI and a basic understanding of manipulating the OS' network configurations.

The SMEs must also have an in-depth foundation of cellular protocols. The verbose details shown on the MME log show the communications occurring between the different components and which protocol is being used. When the program fails, the issues are printed on the screen. The protocol displayed on the MME log should be the first thing investigated after a failure because it will help narrow down the root cause of the failure. The protocols are associated with a particular air interface layer that is usually correlated with a specific component.

The OpenAirInterface software is written in C, and the program is found within multiple C files. Usually, when an error occurs, it is echoed on the MME log along with the full path, filename, and line that the error occurred. Therefore, the SMEs must understand how to navigate the different files, read the C program, and resolve errors. Though the C files are lightly commented, they give context to where errors are derived from—which is useful for troubleshooting.

Lastly, the OpenAirInterface eNB branch used in the test phase does not detect when stack smashing occurs. Stack smashing is a buffer overflow that happens when the input exceeds capacity buffer capacity [42]. As mentioned in [42], stack smashing occurs when a UE attempts to connect to the eNB and immediately releases the connection, which does not complete the RRC handshake. As a result, the eNB is left in a half-open state while constantly attempting to complete the connection, which exhausts the eNB resources. Any attempt to introduce the UE back to the network fails because the buffer overflow has already begun. After approximately a minute, the eNB crashes and requires a re-launch to resume normal operations. In srsRAN, the stack smashing is detected, and the backtrace, file containing debug information, is saved, and the program terminates.

We were able to replicate the eNB stack smashing with high success. We observed the eNB stack smashing consistently when the Google Pixel 4a 5G was connected to the network and Voice over LTE (VoLTE) was enabled. VoLTE is a high-speed wireless standard where voice calls are transmitted using LTE networks. The eNB stack smashing does not occur as often when the OnePlus 6T is the only device connected to the network. The only way identified to resolve an eNB stack smashing occurrence was to re-launch the eNB. An eNB stack smashing can be considered as an internal DoS.

2. Evolved Packet Core

The procedures to install the EPC components were easy to follow; however, some installation commands required configuration parameters that are network specific, and some were default parameters. Some instructions were clear on which parameters required network specificity, while other commands were vague. The developer branches from the master repository were referenced during the installation as other branches had better installation and

configuration instructions than the master branch. The developer branch contains recent commits and is updated about once every week.

As mentioned in Chapter III, Section C, we experienced an unrestorable failure of the EPC host machine. The second attempt to install the EPC on a new machine was unsuccessful because the master branch updated the installation procedures and docker containers. The master branch is updated once every two to three months and is subjected to the continuous integration process. The master branch was committed July 28th; however, is it not known if a failed peer connection from the eNB to the EPC was caused by a bug within the program or a feature not yet added to the updated master branch EPC. The periodic updates added features and changed methodologies of the OpenAirInterface software, making it difficult for researchers and developers to reproduce network configurations and test them against known potential vulnerabilities. The process to install the EPC via this method is outlined in Appendix A, Section C.

To restore our testbed to operation, the EPC was restored from a cloned hard drive with the same version of the EPC and all the dependencies originally installed. A private GitHub page was then created to upload the EPC, eNB, and gNB repositories to provide a backup solution that is easily accessible should any PCs experience another unrestorable failure. The private GitHub page ensures that the repositories are not updated. The username and registered email of the GitHub page are provided in Table 4.

Table 4. NPS GitHub

Username	Email Address
NPS-OAI-LAB	nps.oai.lab@gmail.com

There were times when the EPC host PC was rebooted and the eNB could not establish a peer connection. It was caused because the PC flushes all non-persistent routes in its routing table at reboot. A solution was to create a script that clears all routes in the routing table and adds only the IP routes required for testing. These steps were performed on the EPC, eNB, and gNB. Once the IP routes were corrected, the EPC host, eNB, and gNB could ping the docker containers shown in Figure 9.

3. eNB

The eNB installation presented similar challenges to the EPC. Configuring and installing "Network Configuration 1," as shown in Figure 10, was the first step to constructing a 5G NSA architecture. The rationale for this method is to first establish the 4G LTE network before installing and configuring the gNB. However, configuring "Network Configuration 1" was difficult because of the complexity of the software. The lack of reliable installation documentation and instruction for the master repository caused unnecessary troubleshooting. A few different critical dependencies needed to install the eNB were not covered in the GitHub master repository. The GitHub page assumed that the installer had prior knowledge of the dependent software needed to install the eNB without errors. All software critical to the installation of the eNB is covered in Chapter III and Appendix A.

4. gNB

The procedure to install and configure the gNB was much simpler than that for the eNB. After the installation commands were executed, the configuration files were updated. The configuration that required changing was different than that of the eNB configuration file. The tracking_area_code, plmn_list, mme_ip_address, and enable_x2 changed to "yes" had to match the eNB configurations. The different configuration that the gNB file required set is the "target_enb_x2_ip_address." Once all configurations were updated, the gNB had no issues establishing peer connection to the eNB.

The Google Pixel 4a 5G is the only phone on hand capable of accessing the 5G NSA network. However, connecting the Pixel to the gNB was inconsistent. Since the eNB was running simultaneously with the gNB, the phone often connected only to the eNB rather than the gNB although the gNB is part of the network. Often, the control plane functions were not handled by the gNB. Although the 5G NSA network was connected, we observed the phone connect to the gNB only one time. Every other attempt to connect the phone to the gNB failed. It is not clear why the phone failed to establish connectivity with the gNB. While troubleshooting, the gNB was rebooted, which deleted the stats printed on the terminal used to launch the gNB.

5. User Equipment

The Quectel Module was the first device that was tested and was able to access the network. Interfacing with the device was initially performed using minicom; however, this research migrated to using QNavigator because its GUI is advantageous compared to the command line intensive program. QNavigator also has a functionality that lists AT commands and proper syntax that can be executed by clicking on the AT command.

The second device we attempted to connect to the 4G network was the Google Nexus 6P. Unfortunately, the Nexus could not find the "IND Airtel" network operator, and therefore, it was unable to connect. However, the Nexus 6P was able to find T-Mobile, AT&T, Sprint, Verizon, and FirstNet commercial networks, which are local networks specific to the geographic region. The phone was rebooted several times, a new APN was created, and the SIM card was swapped with a known good SIM card; however, the Nexus was still unable to find the 4G LTE network. We discontinued efforts to connect the Nexus 6P after we identified that the OnePlus was able to find and connect to the "IND Airtel" network.

Connecting the OnePlus 6T and the Google Pixel 4a 5G to the network was as uncomplicated as the Quectel Module. First, a new APN was created and configured on each phone, as discussed in Chapter III, Section E. The network connection bar on the OnePlus and the Google Pixel 4a 5G displayed the 4G LTE icon on the settings bar. Next, the phones were assigned IP addresses on the 12.1.1.0/24 network as configured in the spgwc-cfg.sh file located within the SPGW-C container home directory. Ping tests to Google's DNS primary server were successful from the OnePlus 6T, Google Pixel 4a 5G, and the Quectel Module.

Although the phones were connected to the network and able to ping the DNS servers on the public internet, they could not browse the web using fully qualified domain names (FQDN). The error received on the Google Chrome web browser indicating that the issue was DNS-related. We validated that the issue was DNS-related by testing that the phones were able to browse using IP addresses. We tested using IP addresses for the Naval Postgraduate School's Webpage, 52.42.202.22, and Cloudflare's IP address, 1.1.1.1.

The phones were then disconnected from the 4G network and connected to Wi-Fi to download "Network Tools—DNS Changer," an application found on the Google Play Store

that can switch primary and secondary DNS servers. After re-establishing the connection to the OpenAirInterface cellular network, the DNS was switched to Google's DNS server. The phone was able to ping and web browse when the DNS Changer application was enabled. The performance of the phones is discussed in section B.5 of this chapter.

The next phase was to introduce the Pixel to the 5G NSA network. As mentioned in section 3.B of this chapter, connecting the Pixel to the gNB was inconsistent. When the Pixel is connected to the gNB, it can be viewed on the terminal; however, the phone did not display the 5G icon on its settings bar. It is likely because only the control plane was connected to the gNB, while the user plane was still connected to the eNB (or 4G LTE network). The network icon on the phone displayed a fill bar with the letter "R" above it. The "R" indicates that the phone is roaming, even though roaming was not enabled.

When roaming was enabled on the Pixel, it causes the MME to fail and raise "SEGFAULT," as shown in Figure 20. Although the MME failed, the phones seemingly remained connected to the network, but no services were available. The MME was restored by restarting each container within the MME. We tested roaming on the OnePlus 6T in order to troubleshoot, and the MME did not fail and continued operating normally.

```
Assertion (8 >= pco_dst->num_protocol_or_container_id) failed!
In copy_protocol_configuration_options() /openair-mme/src/common/3gpp_24.008_sm_ies.c:167
Invalid number of protocol_or_container_id 9
 Will Intentionaly raise SEGFAULT to be catched by GDB!
Obtained 8 stack frames.
./bin/oai_mme(display_backtrace+0x28) [0x55fe0aaff021]
./bin/oai_mme(copy_protocol_configuration_options+0x11a) [0x55fe0ab0c096]
./bin/oai_mme(esm_recv_pdn_connectivity_request+0xbe4) [0x55fe0aaee4e0]
./bin/oai_mme(_esm_sap_recv+0x2ca) [0x55fe0aab1ca0]
./bin/oai_mme(nas_esm_proc_data_ind+0x7e) [0x55fe0aaaf9a7]
./bin/oai_mme(+0x115691) [0x55fe0a9dc691]
/lib/x86_64-linux-gnu/libpthread.so.0(+0x76db) [0x7f1c8f8466db]
/lib/x86_64-linux-gnu/libc.so.6(clone+0x3f) [0x7f1c8d9ff71f]
005107 00638:964113 7F1C82FFD700 DEBUG UDP    /src/udp/udp_primitives_server.c:0122    Received 1 events
```

Figure 20. MME Log SEGFAULT

6. Network Performance

Network performance tests were performed to measure latency and throughput. Latency was measured using pings, and throughput was tested using iPerf. iPerf is a tool to

measure maximum achievable bandwidth on IP networks. Pings were conducted using a third-party application available on Google Store, the Hurricane Electric (HE.Net) Network Tools. This third-party application was chosen because it implemented both pings and iPerf along with other network diagnostic tools.

We performed iPerf tests from the Google Pixel 4a 5G to the EPC, eNB, and gNB, respectively. The iPerf test can measure the throughput between the UE and the base station. The EPC, eNB, and gNB were the server for each test, and the Pixel was the client in the test. Five iPerf2 tests were performed from the UE to the eNB and the EPC. The outcome of the iPerf tests with the eNB as the server resulted in an average bandwidth of 19.97 Mbits/sec.

We also attempted to measure 5G NSA network performance. The iPerf test conducted from the UE to the EPC as the server resulted in an average bandwidth of 20.76 Mbits/sec. The iPerf tests attempted on the gNB caused eNB to experience stack smashing most of the time when an iPerf test was performed. We gathered three successful iPerf tests with the gNB as the server out of more than 15 attempts. Table 5 shows the bandwidth of 1MB iPerf tests from the Google Pixel to the EPC and eNB in LTE mode and the gNB in NSA mode. Appendix F lists the iPerf test performed in this research.

The gNB performance results were significantly lower than expected. The cause of the low throughput could be due to the eNB stack smashing even during a successful iPerf test; however, we were not able to confirm that the eNB stack smashing is the cause of the low throughput. Appendix F lists the full iPerf results from the EPC, eNB, and gNB.

Table 5. iPerf Test Results

Server	UE Min Bandwidth	UE Avg Bandwidth	UE Max Bandwidth	Command Used on Server
iPerf from UE to EPC LTE mode	962 Kbits/sec	20.76 Mbits/sec	29.8 Mbits/sec	iperf -s 192.168.1.7
iPerf from UE to eNB LTE mode	5.15 Mbits/sec	19.97 Mbits/sec	32.9 Mbits/sec	iperf -s 192.168.1.8
iPerf from UE to gNB NSA mode	1.33 Kbits/sec	71.11 Kbits/sec	126 Kbits/sec	iperf -s 192.168.1.9

We measured latency for "Network Configuration 1" by pinging Google's, Cloudflare's, and AT&T's DNS servers using the HE.Net application. One thousand pings were performed from the OnePlus 6T and the Pixel 4a 5G to each DNS server. The average round-trip-time (RTT) for pings performed by the OnePlus 6T is 38.72 ms with a 0.013% packet loss rate. The average RTT for pings performed by the Google Pixel 4a 5G was 42.11 ms with a 0.011% packet loss rate. Table 6 shows the average of the result from the ping tests conducted in this research. Appendix E shows the results of all pings performed from each phone.

Table 6. Ping Test Results (average)

OnePlus 6T				
Success	**Loss rate**	**Avg RTT**	**Best RTT**	**Worst RTT**
0.987	0.013	38.72	27.0	76.87

Google Pixel				
Success	**Loss rate**	**Avg RTT**	**Best RTT**	**Worst RTT**
0.989	0.011	42.114	29.6	81.3

7. Network Mobility

One of the benefits of the OpenAirInterface network implementation is that it has a relatively small footprint. The small footprint enables the testing environment to be disestablished and re-established without making extensive configuration changes, if any. In our case, the only required configuration changes involve updating IP routes and ensuring the EPC can access the internet. The testing environment was moved from NPS' Center for Cyber Warfare (CCW) to an offsite testing environment. The lab was able to be disestablished and re-established within minutes. IP routes were updated using bash scripts.

While setup in the CCW, the phones experience an intermittent connection to the MME. We witnessed the "MME Log" displaying the UE completing the attach procedure and immediately terminating the UE attach connection. Fortunately, the cycle of the UE connecting to the network and disconnecting from the network did not cause the eNB to

crash because the UE completed the RRC handshake, which did not leave the eNB in a half-open state.

It seems as if the cellular connection quality between eNB and the two UEs was better in the off-site test lab than the quality within the CCW. The phones did not disconnect intermittently, and web browsing was successfully tested for two hours. The only successful iPerf test was conducted in the offsite environment. After the testing environment was returned to the CCW, the issues experienced before the move recurred. The testbed was moved back to the offsite test lab a second time, and the issues experienced in the CCW didn't reappear. Future researchers could leverage a spectrum analyzer to determine whether there are frequencies around the CCW that could potentially conflict with the LTE band 7 and NR band n78—the frequency band used during this capstone. 3GPP LTE and NR operating bands are listed in Appendix G.

THIS PAGE INTENTIONALLY LEFT BLANK

V. CONCLUSIONS AND FUTURE WORK

A. CONCLUSIONS

This capstone provides a detailed guide to implementing a 4G / 5G NSA network using the OpenAirInterface software. Also, it demonstrates the potential ability to establish scalable, inexpensive, rapidly deployable, and reconfigurable cellular networks using OpenAirInterface. Three network configurations and various tools that could be used have been discussed in this research, offering future researchers the ability to recreate, reconfigure, and troubleshoot the networks presented. The different network configurations presented in this work add to the network configurations analyzed in [18]. In addition, this research did not use the attenuators used in [18], proving that the Ettus B200 SDRs could handle air interface traffic at high speeds.

Although OpenAirInterface has been under development in the past few years, the software is still unstable. Furthermore, we assess that the target audience for the OpenAirInterface software is researchers and developers. Therefore, the software's usability cannot be evaluated until a stable release of the program is available.

We have tested the flexibility of the OpenAirInterface software by adding an eNB to "Network Configuration 1." We also proved the software's flexibility by adding a gNB to "Network Configuration 1." The addition of the eNB or gNB proves that the program does not require major configuration changes to change operating modes.

Last, we were able to showcase the eNB crashing due to eNB stack smashing. The eNB stack smashing is a form of an internal DoS attack. The DoS attack could be caused intentionally or unintentionally. Other potential vulnerabilities native to 4G and 5G were not tested, and it remains unknown whether the vulnerabilities exist within the OpenAirInterface software.

We conclude that the OpenAirInterface software is operational but not ready to be adopted for commercial use or implemented within the DOD. The software is still undergoing heavy development and significant changes. Though it is not known when the stable software will be released, continued research using the current software

implementation is valuable to understanding the potential use cases within the DOD once stable software is available.

B. FUTURE WORK

One direction for future work involves isolating and resolving issues related to the MME. For instance, future researchers could investigate the cause of the MME failure when roaming is enabled on the Pixel. The MME is susceptible to an internal DoS attack from within the network. Also, future researchers could determine if the cause of the MME segment fault is only caused by the Google Pixel 4a 5G or a bug within the OpenAirInterface software.

The new process to implement the 4G LTE and 5G NSA architecture has been streamlined; however, different configurations must be examined. Continued research in the 4G LTE and 5G NSA networks would enable the DOD to move one step closer to realizing a cellular network that is owned, governed, and managed by the DOD. Also, additional research in the new installation process could evaluate whether some of the challenges experienced in this research still exist. For example, further research could determine if the new implementation methods handle the eNB crashing due to stack smashing. There is a potential that the OpenAirInterface implemented software solutions in the eNB manage and half-open connections caused by UEs.

Furthermore, to truly realize the full capabilities that 5G offers, research in the SA architecture could prove beneficial for future use cases where eMBB, URLLC, high-speed wireless services, or mMTC are required in military operations. For example, a military application would be to use the 5G network to pass data using the Android Team Awareness Kit Military (ATAK-MIL) application in littoral operations. ATAK-MIL is a government off-the-shelf android app that enables blue and red force tracking on a map overlay, messaging, and critical information sharing that could be used in a tactical environment. Through ATAK, Commanders could maintain a near real-time situational awareness of an operation and deliver critical mission updates to nodes within the network. In the example mentioned above, the 5G network could also be used to enable node-to-node communication.

Once a stable, end-to-end implementation of the 4G and 5G cellular network software is available, continued research could test the addition of multiple UEs and the stability of the 5G NSA software stack. We were not able to fully test and analyze all primary capabilities of mobile phones on the NSA network. To deploy the software or similar networks, research would be required to determine the additional equipment needed and compatibility to expand the cellular coverage range and ultimately scale the network to include multiple cores and base stations.

Finally, one of the utilities used for writing SIM cards was OYEITIMES SIM Writer. The application had functionality that could not be updated because the fields were greyed out. Within the directory of the files, there was a file titled "GRSIMWrite.grsp," which contains all the programmable fields in the OYEITIMES application. Future work could explore how the application writes to SIM cards and determine if the fields could be updated on the ".grsp" files to enable the fields to be written on the SIM.

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX A. OPENAIRINTERFACE SOFTWARE INSTALLATION GUIDE

This appendix details the prerequisite software and installation steps discussed in Chapter III. This appendix is arranged into four sections—Section A describes the installation procedures to install the USRP. The installation processes to install the eNB and gNB are described in Section B. Finally, the methods to install the EPC are covered in Section C. Section D provides the new procedures used to install the EPC after recovering from the host failure.

A. PREREQUISITE AND USRP SETUP

1. Low-Latency Kernel

1. Navigate to the Ubuntu packages webpage.

https://packages.ubuntu.com/bionic/linux-signed-lowlatency-hwe-18.04/download

2. Select the security.ubuntu.com/ubuntu link.

3. A dialog box will appear. Click Ok.

4. The linux-signed-low-latency-hwe-18.04 installation box will appear.

Click Install.

5. After the installation is completed, reboot the machine.

6. Verify kernel installation.

uname -a

2. VOLK Libraries

Vector-Optimized Library of Kernels (VOLK) is a library used to give GNU Radio a boost in performance in signal processing. The Volk framework adds single Instructions, Multiple Data (SIMD) functionality that simultaneously processes multiple data inputs with a single instruction. VOLK is a set of functions that act as a vector multiplier for complex floats [43]. The instructions to install VOLK radio is derived from [44].

Install VOLK

1. Clone the VOLK repository

```
git clone --recursive https://github.com/gnuradio/volk.git
```

2. Clone VOLK submodules

```
git submodule update --init –recursive
```

3. Build Volk

```
cd volk
mkdir build
cd build
cmake -DCMAKE_BUILD_TYPE=Release -DPYTHON_EXECUTABLE=/usr/bin/python3 .//
make
make test
sudo make install
```

4. Link the system's cache shared library.

```
sudo ldconfig
```

3. GNU Radio

GNU Radio is an open-source software toolkit that performs signal processing using software rather than using embedded circuits in hardware radios. The GNU Radio software supports the required signal processing via that the open-air-interface stack needs to operate.

4. USRP UHD

1. Update and install the required dependencies.

```
sudo apt-get update
```

```
sudo apt-get -y install git swig cmake doxygen build-essential libboost-all-dev libtool libusb-1.0-0 libusb-1.0-0-dev libudev-dev libncurses5-dev libfftw3-bin libfftw3-dev libfftw3-doc libcppunit-1.14-0 libcppunit-dev libcppunit-doc ncurses-bin cpufrequtils python-numpy python-numpy-doc python-numpy-dbg python-scipy python-docutils qt4-bin-dbg qt4-default qt4-doc libqt4-dev libqt4-dev-bin python-qt4 python-qt4-dbg python-qt4-dev python-qt4-doc python-qt4-doc libqwt6abi1 libfftw3-bin libfftw3-dev libfftw3-doc ncurses-bin libncurses5 libncurses5-dev libncurses5-dbg libfontconfig1-dev libxrender-dev libpulse-dev swig g++ automake autoconf libtool python-dev libfftw3-dev libcppunit-dev libboost-all-dev libusb-dev libusb-1.0-0-dev fort77 libsdl1.2-dev python-wxgtk3.0 git libqt4-dev python-numpy ccache python-opengl libgsl-dev python-cheetah python-mako python-lxml doxygen qt4-default qt4-dev-tools libusb-1.0-0-dev libqwtplot3d-qt5-dev pyqt4-dev-tools python-qwt5-qt4 cmake git wget libxi-dev gtk2-engines-pixbuf r-base-dev python-tk liborc-0.4-0 liborc-0.4-dev libasound2-dev python-gtk2 libzmq3-dev libzmq5 python-requests python-sphinx libcomedi-dev python-zmq libqwt-dev libqwt6abi1 python-six libgps-dev libgps23 gpsd gpsd-clients python-gps python-setuptools
```

2. Build and install the UHD from the source code.

```
cd $HOME

mkdir workarea

cd workarea
```

3. Clone the repository.

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

cd uhd
```

4. List the UHD releases and checkout the desired UHD.

```
git tag -l

git checkout v3.14.0.0
```

5. Create a build folder.

```
cd host

mkdir build

cd build

cmake ../
```

```
make
```

6. Run basic tests.

```
make test
```

7. Install the UHD

```
sudo make install
```

8. Link the system's cache shared library.

```
sudo ldconfig
```

9. Ensure that the LD_LIBRARY_PATH environment is defined under the folder that the UHD is installed.

```
export LD_LIBRARY_PATH=/usr/local/lib
```

10. If the library is already defined, run the following command.

```
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/local/lib
```

11. Download the UHD FPGA Images.

```
sudo uhd_images_downloader
```

12. Write the "XG" FPGA image to the USRP B200.

```
uhd_image_loader --args "type=x300,fpga=XG"
```

13. Change the socket buffer sizes.

```
sudo sysctl -w net.core.rmem_max=33554432
sudo sysctl -w net.core.wmem_max=33554432
```

14. Connect the USRP and verify the UHD installation.

```
uhd_find_devices
uhd_usrp_probe
```

15. Configure the USB

```
cd $HOME/workarea/uhd/host/utils
```

```
sudo cp uhd-usrp.rules /etc/udev/rules.d/
sudo udevadm control --reload-rules
sudo udevadm trigger
```

16. Configure thread scheduling

```
sudo groupadd usrp
sudo usermod -aG usrp $USER
```

17. Configure USB to enable non-root users to 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
```

5. GNU Radio

1. Clone the repository

```
cd /home/labware/workarea
git clone --recursive https://github.com/gnuradio/gnuradio
```

2. Checkout GNU Radio release

```
cd gnuradio
checkout branch
git checkout v3.7.13.4
```

3. Update submodules

```
git submodule update --init --recursive
```

4. Create a folder within the repository, invoke CMake, build GNU Radio

```
mkdir build
cd build
```

```
cmake ../
make
```

5. Run basic tests.

```
make test
```

6. Install GNU Radio using the default prefix.

```
sudo make install
```

7. Link the system's cache shared library.

```
sudo ldconfig
```

8. Verify that GNU Radio is installed.

```
gnuradio-config-info --version
gnuradio-config-info --prefix
gnuradio-config-info --enabled-components
```

9. Run the GNU Radio

```
gunradio-companion
```

B. ENB/GNB

1. eNB Installation

1. Retrieve the latest version of the `master` branch and build the eNB

```
mkdir -p ~/openairinterface5g
cd ~/openairinterface5g/
git checkout master
source oaienv
cd cmake_targets
./build_oai -w USRP –eNB –build-lib
```

2. Edit the conf file by replacing all `CI_*IP_ADDR` pattern with the correct IP address.

```
source oaienv

cd ~/openairinterface5g/ci-scripts/conf_files

cp enb.band7.tm1.25PRB.usrpb200.conf my-enb.band7.tm1.25PRB.usrpb200.conf

vi my-enb.band7.tm1.25PRB.usrpb210.conf

cd ~/openairinterface5g/cmake_targets
```

3. Starting the eNB.

```
sudo -E ./lte_build_oai/build/lte-softmodem -O ~/openairinterface5g/ci-scripts/conf_files/ci-enb.band7.tm1.25PRB.usrpb210.conf --nokrnmod 1 --noS1 --eNBs.[0].rrc_inactivity_threshold 0 2>&1 | tee ENB.log
```

4. Running the eNB.

```
cd cmake_targets

sudo -E ./lte_build_oai/build/lte-softmodem -O ~/openairinterface5g/ci-scripts/conf_files/ci-enb.band7.tm1.25PRB.usrpb210.conf --nokrnmod 0 --noS1 --eNBs.[0].rrc_inactivity_threshold 0 2>&1 | tee ENB.log
```

2. eNB Configuration

The eNB requires a configuration file to set the RAN parameters to establish a peer connection to the EPC. The “enb.band7.tm1.25PRB.usrpb210.conf” file was used for this capstone. There are a few lines that are network-specific and are required to be updated. Figure 21 is a screenshot of the eNB configuration file. The configuration file shows an updated eNB_ID, tracking area code, and plmn_list (MCC and MNC updated).

```
eNBs =
(
 {
    ////////// Identification parameters:
    eNB_ID    =  0xe01;

    cell_type =  "CELL_MACRO_ENB";

    eNB_name  =  "eNB-Eurecom-LTEBox";

    // Tracking area code, 0x0000 and 0xfffe are reserved values
    tracking_area_code = 5;
    plmn_list = ( { mcc = 404; mnc = 92; mnc_length = 2; } );
```

Figure 21. enb.band7.tm1.25PRB.usrpb210 - eNB Configuration

Table 7 summarizes the updated fields in the eNB configuration file and a brief description of the field.

Table 7. Changes Made in the eNB Configuration File [45].

Option	Update	Description
eNB_ID	0xe01	It is not required for single eNB architecture. However, in a multiple eNB architecture, the eNB_ID will conflict on the EPC and only one will be allowed to establish peer connectivity.
Tracking area code	5	Internationally unique identifier for a tracking area within the PLMN
MCC	404	MCC is used along with the International Mobile Subscriber Identity (IMSI) to identify the region from which the mobile subscriber belongs.
MNC	92	MNC uniquely identifies the home PLMN to the user on the mobile device display when connected to the network.

Figure 22 is a screenshot of additional configuration changes made on the eNB configuration file.

```
    ////////// MME parameters:
    mme_ip_address      = ( { ipv4       = "192.168.61.3";
                              ipv6       = "192:168:30::17";
                              active     = "yes";
                              preference = "ipv4";
                            }
                          );

    enable_measurement_reports = "yes";

    ///X2
    enable_x2 = "yes";
    t_reloc_prep      = 1000;      /* unit: millisecond */
    tx2_reloc_overall = 2000;      /* unit: millisecond */

    NETWORK_INTERFACES :
    {
        ENB_INTERFACE_NAME_FOR_S1_MME            = "eth0";
        ENB_IPV4_ADDRESS_FOR_S1_MME              = "192.168.1.8";
        ENB_INTERFACE_NAME_FOR_S1U               = "enp0s25";
        ENB_IPV4_ADDRESS_FOR_S1U                 = "192.168.1.8";
        ENB_PORT_FOR_S1U                         = 2152; # Spec 2152
        ENB_IPV4_ADDRESS_FOR_X2C                 = "192.168.1.8";
        ENB_PORT_FOR_X2C                         = 36422; # Spec 36422
    };
  }
);
```

Figure 22. enb.band7.tm1.25PRB.usrpb210 - MME Parameters and Network_Interfaces

Table 8 shows configuration changes made in the MME parameters and X2 settings section of the eNB configuration file. The IP address shown in Table 8 is the default IP address of the MME; however, the docker network IP address could be changed. If changes to the Docker network IP address were made, this section would need to be updated.

Table 8. enb.band7.tm1.25PRB.usrpb210—MME Parameters and X2 Settings

MME parameters	
Setting	**Update**
ipv4	“192.168.61.3”
X2 Settings	
Setting	**Update**
enable_x2	“yes”

Table 9 shows the Network Interfaces configuration updates made on the eNB configuration file. The IP addresses listed is the IP address of the eNB host PC. The eNB host machine was given a static IP address of 192.168.1.8.

Table 9. enb.band7.tm1.25PRB.usrpb210—Network Interfaces Settings

Network Interfaces	
Setting	**Update**
ENB_IPV4_ADDRESS_FOR_S1_MME	"192.168.1.8"
ENB_IPV4_ADDRESS_FOR_S1U	"192.168.1.8"
ENB_IPV4_ADDRESS_FOR_X2C	"192.168.1.8"

3. gNB Installation

1. Download the repository from GitLab

https://gitlab.eurecom.fr/oai/openairinterface5g/-/tree/develop

2. Build and install the gNB.

```
cd <oai installation directory>/openairinterface5g-develop
source oaienv
cd cmake_targets/
./build_oai -I -w USRP –gNB
```

3. Update the MME section and the Network Interfaces section of the gNB configuration file with the IP address of the EPC.

```
cd <oai installation directory>/openairinterface5g-develop/ci-scipts/conf_files
gedit <filename.conf>
```

4. Update the MME parameters section of the gNb configuration file with the IP address of the EPC. Update the X2 Network Interfaces section of the configuration file with the IP address of the eNB host / gNB host.

```
cd <oai installation directory>/openairinterface5g-develop/ci-scipts/conf_files
```

gedit <filename.conf>

4. gNB Configuration

The gNB requires a configuration file to set the RAN parameters to establish a peer connection to the eNB. The gnb.band78.tm1.fr1.106PRB.usrpb210 file was used for this capstone. In addition, there are a few lines that are network-specific and are required to be updated. Figure 23 is a screenshot of the gNB configuration file. The configuration file shows an updated gNB_ID, tracking area code, and plmn_list to match the configuration made on the eNB configuration file.

```
gNBs =
(
 {
    ////////// Identification parameters:
    gNB_ID    =  0xe01;

    cell_type =  "CELL_MACRO_GNB";
    gNB_name  =  "gNB-Eurecom-5GNRBox";

    // Tracking area code, 0x0000 and 0xfffe are reserved values
    tracking_area_code  =  5;
    plmn_list = ({mcc = 404; mnc = 92; mnc_length = 2;});
```

Figure 23. gnb.band78.tm1.fr1.106PRB.usrpb210—gNB Configuration

Table 10 summarizes the updated fields in the gNB configuration file and a brief description of the field.

Table 10. srpb210—gNB configuration

Option	Update	Description
gNB_ID	0xe01	Not required for single geNB architecture.
Tracking area code	5	Internationally unique identifier for a tracking area within the PLMN
MCC	404	MCC is used along with the International Mobile Subscriber Identity (IMSI) to identify

Option	Update	Description
		the region from which the mobile subscriber belongs.
MNC	92	MNC uniquely identifies the home PLMN to the user on the mobile device display when connected to the network.

Figure 24 is a screenshot of additional configuration changes made on the gNB configuration file.

```
    ////////// MME parameters:
    mme_ip_address      = ( { ipv4       = "192.168.61.3";
                              ipv6       = "192:168:30::17";
                              port       = 36412 ;
                              active     = "yes";
                              preference = "ipv4";
                            }
                          );

    ///X2
    enable_x2 = "yes";
    t_reloc_prep      = 1000;      /* unit: millisecond */
    tx2_reloc_overall = 2000;      /* unit: millisecond */
    t_dc_prep         = 1000;      /* unit: millisecond */
    t_dc_overall      = 2000;      /* unit: millisecond */
    target_enb_x2_ip_address      = (
                                      { ipv4       = "192.168.1.8";
                                        ipv6       = "192:168:30::17";
                                        preference = "ipv4";
                                      }
                                     );

    NETWORK_INTERFACES :
    {

        GNB_INTERFACE_NAME_FOR_S1_MME            = "eth0";
        GNB_IPV4_ADDRESS_FOR_S1_MME              = "192.168.1.9";
        GNB_INTERFACE_NAME_FOR_S1U               = "eth0";
        GNB_IPV4_ADDRESS_FOR_S1U                 = "192.168.1.9";
        GNB_PORT_FOR_S1U                         = 2152; # Spec 2152
        GNB_IPV4_ADDRESS_FOR_X2C                 = "192.168.1.9";
        GNB_PORT_FOR_X2C                         = 36422; # Spec 36422
    };
  }
);
```

Figure 24. gnb.band78.tm1.fr1.106PRB.usrpb210—MME Parameters and Network_Interfaces

Table 11 shows configuration changes made in the MME parameters and X2 settings section of the gNB configuration file. The IP address shown in Table 11 is the default IP address of the MME that is in the eNB configuration file.

Table 11. gnb.band78.tm1.fr1.106PRB.usrpb210—MME parameters and X2 Settings

MME parameters	
Setting	**Update**
ipv4	"192.168.61.3"
X2 Settings	
Setting	**Update**
enable_x2	"yes"

Table 12 shows the Network Interfaces configuration updates made on the gNB configuration file. The IP addresses listed is the IP address of the gNB host PC. The eNB host machine was given a static IP address of 192.168.1.9.

Table 12. gnb.band78.tm1.fr1.106PRB.usrpb210—Network Interface Settings

Network Interfaces	
Setting	**Update**
GNB_IPV4_ADDRESS_FOR_S1_MME	"192.168.1.9"
GNB_IPV4_ADDRESS_FOR_S1U	"192.168.1.9"
GNB_IPV4_ADDRESS_FOR_X2C	"192.168.1.9"

C. EPC

1. Low-Latency Kernel

1. Navigate to the Ubuntu packages webpage.

https://packages.ubuntu.com/bionic/linux-signed-lowlatency-hwe-18.04/download

2. Select the security.ubuntu.com/ubuntu link.

3. A dialog box will appear. Click Ok.

4. The linux-signed-low-latency-hwe-18.04 installation box will appear.

Click Install.

5. After the installation is completed, reboot the machine.

reboot

6. Verify kernel successfully installed.

uname -a

2. freeDiameter

1. Install the required dependencies for building the source for freeDiameter

sudo apt-get -y install mercurial cmake make gcc g++ bison flex libsctp28-dev libgnutls-dev libgcrypt20-dev libidn11-dev ssl-cert debhelper fakeroot libpq-dev libmysqlclient-dev libxml2-dev swig python-dev

2. Clone the freeDiameter repository

cd

hg clone http://www.freediameter.net/hg/freeDiameter

cd freeDiameter

3. Configure and generate Makefiles

mkdir fDbuild

cd fDbuild

cmake ../freeDiameter

4. Edit (optional) and compile files

make edit_cache

make

3. Docker

1. Uninstall old version of docker

```
sudo apt-get remove docker docker-engine docker.io containerd runc
```

2. Set up the repository

```
sudo apt-get update

sudo apt-get install

sudo apt-transport-https \

ca-certificates \

curl \

gnupg \

lsb-release
```

3. Add Docker's GPG key.

```
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /usr/share/keyrings/docker-archive-keyring.gpg
```

4. Set up the stable repository

```
echo "deb [arch=amd64 signed-by=/usr/share/keyrings/docker-archive-keyring.gpg] https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
```

5. Install Docker Engine (use this step to install the latest version of the Docker engine).

```
sudo apt-get update

sudo apt-get install docker-ce docker-ce-cli containerd.io
```

6. List and install a specific version of the Docker Engine (use this step to install a previously released version).

```
apt-cache madison docker-ce

sudo apt-get install docker-ce=<VERSION_STRING> docker-ce-cli=<VERSION_STRING> containerd.io
```

7. Verify that Docker Engine is correctly installed

```
sudo docker run hello-world
```

9. Delete the hello-world image (optional)

```
docker image rm hello-world --force
```

8. Check the currently installed version of Docker Engine

```
apt list | grep docker-ce
```

4. openair-epc-fed

1. Clone the openair-epc-fed repository

```
cd
git clone https://github.com/OPENAIRINTERFACE/openair-epc-fed.git
cd openair-epc-fed
```

2. Check out the latest GIT repository

```
git checkout master
git pull origin master
```

3. Resync the sub-modules

```
./scripts/syncComponents.sh
```

4. Log into docker and pull the docker images.

```
docker login [enter your docker credentials when prompted]
docker pull Ubuntu:bionic
docker pull cassandra:2.1
docker logout
```

5. Enable forwarding from the Docker containers to the outside world

```
sudo sysctl net.ipv4.conf.all.forwarding=1
sudo iptables -P FORWARD ACCEPT
```

6. change the IP range of the default docker bridge network. If the daemon.json file does not exist, create it and add the bridge IP. If the daemon.json file does exist, verify the IP address of the bridge to your network.

```
gedit /etc/docker/daemon.json
```

7. Add the network information

```
{
    "bip": "Bridge_IP_ADDRESS"
}
```

8. Restart the docker daemon and verify that the bridge IP has been updated.

```
sudo service docker stop
sudo service docker start
docker network inspect bridge name
```

5. Build the HSS, MME, SPGW-C, and SPGW-U images without the EURECOM proxy option.

```
docker build --target oai-hss --tag oai-hss:production --file component/oai-hss/docker/Dockerfile.ubuntu18.04 component/oai-hss
docker build --target oai-mme --tag oai-mme:production --file component/oai-mme/docker/Dockerfile.ubuntu18.04 component/oai-mme
docker build --target oai-spgwc --tag oai-spgwc:production --file component/oai-spgwc/docker/Dockerfile.ubuntu18.04 component/oai-spgwc
docker build --target oai-spgwu-tiny --tag oai-spgwu-tiny:production --file component/oai-spgwu-tiny/docker/Dockerfile.ubuntu18.04 component/oai-spgwu-tiny
```

6. Remove untagged and images not referenced by any container.

```
docker image prune --force
```

7. Verify the containers built were successful

```
docker image ls
```

8. Create docker public network

```
docker network create --attachable --subnet 192.168.61.0/26 --ip-range 192.168.61.0/26 prod-oai-public-net

docker run --name prod-cassandra -d -e CASSANDRA_CLUSTER_NAME="OAI HSS Cluster" -e CASSANDRA_ENDPOINT_SNITCH=GossipingPropertyFileSnitch cassandra:2.1

docker network connect prod-oai-public-net prod-oai-hss

docker run --privileged --name prod-oai-mme --network prod-oai-public-net -d --entrypoint /bin/bash oai-mme:production -c "sleep infinity"

docker run --privileged --name prod-oai-spgwc --network prod-oai-public-net -d --entrypoint /bin/bash oai-spgwc:production -c "sleep infinity"

docker run --privileged --name prod-oai-spgwu-tiny --network prod-oai-public-net -d --entrypoint /bin/bash oai-spgwu-tiny:production -c "sleep infinity"
```

9. Configure the Cassandra, HSS, MME, SPGW-C, and SPGW-U containers.

9.1. Cassandra

```
docker cp component/oai-hss/src/hss_rel14/db/oai_db.cql prod-cassandra:/home

docker exec -it prod-cassandra /bin/bash -c "nodetool status"

Cassandra_IP=`docker inspect --format="{{range .NetworkSettings.Networks}}{{.IPAddress}}{{end}}" prod-cassandra`

docker exec -it prod-cassandra /bin/bash -c "cqlsh --file /home/oai_db.cql ${Cassandra_IP}"
```

9.2. HSS

```
HSS_IP=`docker exec -it prod-oai-hss /bin/bash -c "ifconfig eth1 | grep inet" | sed -f ./ci-scripts/convertIpAddrFromIfconfig.sed` \ python3 component/oai-hss/ci-scripts/generateConfigFiles.py --kind=HSS

--cassandra=${Cassandra_IP} --hss_s6a=${HSS_IP} --apn1=apn1.carrier.com \

--apn2=NPS4G.apn.epc.mnc092.mcc404.3gppnetwork.org --users=200 \

--imsi=320230100000001 --ltek=0c0a34601d4f07677303652c0462535b \

--op=63bfa50ee6523365ff14c1f45f88737d  --nb_mmes=1 --from_docker_file \

docker cp ./hss-cfg.sh prod-oai-hss:/openair-hss/scripts

docker exec -it prod-oai-hss /bin/bash -c "cd /openair-hss/scripts && chmod 777 hss-cfg.sh && ./hss-cfg.sh"
```

9.3 MME

```
MME_IP=`docker inspect --format="{{range .NetworkSettings.Networks}}{{.IPAddress}}{{end}}" prod-oai-mme`
```

```
SPGW0_IP=`docker inspect --format="{{range .NetworkSettings.Networks}}{{.IPAddress}}{{end}}" prod-oai-spgwc` \ python3 component/oai-mme/ci-scripts/generateConfigFiles.py --kind=MME \
--hss_s6a=${HSS_IP} --mme_s6a=${MME_IP} \
--mme_s1c_IP=${MME_IP} --mme_s1c_name=eth0 \
--mme_s10_IP=${MME_IP} --mme_s10_name=eth0 \
--mme_s11_IP=${MME_IP} --mme_s11_name=eth0
--spgwc_s11_IP=${SPGW0_IP} \
--mcc=404 --mnc=92 --tac_list="5 6 7" --from_docker_file
docker cp ./mme-cfg.sh prod-oai-mme:/openair-mme/scripts
docker exec -it prod-oai-mme /bin/bash -c "cd /openair-mme/scripts && chmod 777 mme-cfg.sh && ./mme-cfg.sh"
```

9.4 SPGW-C

```
python3 component/oai-spgwc/ci-scripts/generateConfigFiles.py --kind=SPGW-C \
--s11c=eth0 --sxc=eth0 --apn=apn1.carrier.com --dns1_ip=172.20.20.12 --dns2_ip=8.8.8.8 --from_docker_file
docker cp ./spgwc-cfg.sh prod-oai-spgwc:/openair-spgwc
docker exec -it prod-oai-spgwc /bin/bash -c "cd /openair-spgwc && chmod 777 spgwc-cfg.sh && ./spgwc-cfg.sh"
```

9.5 SPGW-U

```
python3 component/oai-spgwu-tiny/ci-scripts/generateConfigFiles.py --kind=SPGW-U \
--sxc_ip_addr=${SPGW0_IP} --sxu=eth0 --s1u=eth0 --from_docker_file
docker cp ./spgwu-cfg.sh prod-oai-spgwu-tiny:/openair-spgwu-tiny
docker exec -it prod-oai-spgwu-tiny /bin/bash -c "cd /openair-spgwu-tiny && chmod 777 spgwu-cfg.sh && ./spgwu-cfg.sh"
docker inspect --format="{{range .NetworkSettings.Networks}}{{.IPAddress}}{{end}}" prod-cassandra
docker inspect --format="{{range .NetworkSettings.Networks}}{{.IPAddress}}{{end}}" prod-oai-hss
docker inspect --format="{{range .NetworkSettings.Networks}}{{.IPAddress}}{{end}}" prod-oai-mme
docker inspect --format="{{range .NetworkSettings.Networks}}{{.IPAddress}}{{end}}" prod-oai-spgwc
```

```
docker inspect --format="{{range .NetworkSettings.Networks}}{{.IPAddress}}{{end}}"
prod-oai-spgwu-tiny
```

D. UPDATED EPC INSTALLATION

This section provides the new methods to install the EPC. To install, replace Section C in this appendix with this section.

1. openair-epc-fed

1. Pull the images from Docker Hub

```
docker login

docker pull ubuntu: bionic

docker pull cassandra:2.1

docker pull rdefosseoai/oai-hss

docker pull rdefosseoai/oai-mme

docker pull rdefosseoai/oai-spgwc

docker pull rdefosseoai/oai-spgwu-tiny
```

2. re-tag the images for docker-compose

```
docker image tag rdefosseoai/oai-hss:latest oai-hss:production

docker image tag rdefosseoai/oai-mme:latest oai-mme:production

docker image tag rdefosseoai/oai-spgwc:latest oai-spgwc:production

docker image tag rdefosseoai/oai-spgwu-tiny:latest oai-spgwu-tiny:production
```

3. remove the re-tagged images (optional)

```
docker image rm rdefosseoai/oai-hss

docker image rm rdefosseoai/oai-mme

docker image rm rdefosseoai/oai-spgwc

docker image rm rdefosseoai/oai-spgwu-tiny
```

4. Network Configuration

```
sudo sysctl net.ipv4.conf.all.forwarding=1
sudo iptables -P FORWARD ACCEPT
```

5. Manually configuring IP range (optional)

```
cd /etc/docker/
sudo touch daemon.json
sudo cat >> daemon.json
```

6. Enter the IPv4 address and CIDR notation into the daemon.json file.

```
{
        "bip": "ipv4/CIDR"
}
ctrl+ c
```

7. Restart the docker service, verify it is running, and verify the network configurations committed.

```
sudo service docker restart
service docker status
docker network inspect bridge
```

8. Clone the repository

```
git clone --branch v1.1.2 https://github.com/OPENAIRINTERFACE/openair-epc-fed.git
cd openair-epc-fed
git checkout -f v1.1.2
```

9. Syncrhonize all git submodules

```
./scripts/syncComponentsLegacy.sh
```

10. Initialize the Cassandra DB

```
cd docker-compose/oai-mme-legacy
docker-compose up -d db_init
```

11. Follow database logs. Before continuing beyond this step, a connection must be made. An “OK” should appear to indicate a connection to the SQL server is successful. Because connection errors occur before receiving the “OK,” the following command may be entered multiple times.

```
docker logs prod-db-init –follow
```

12. Remove the database init container because it is not needed anymore.

```
docker rm -f prod-db-init
```

APPENDIX B. OPENAIRINTERFACE SOFTWARE SIMULATION

This appendix summarizes the steps used to create different bash scripts during simulation. Scripts are used to run a series of commands; any command that could be executed from the terminal can be used in a script. First, this appendix discusses creating bash scripts for running the environment. Using scripts speeds up and streamlines launching the EPC as well as tearing it down. Using bash scripts makes simulating the test environment easier and faster. Rather than running multiple commands, a single command could be executed to run the script.

A. SETTING UP BASH SCRIPTS

1. Open a text editor and save it with a ".sh" file extension in a location where the shell can find it. The first line should have only **#!/bin/bash** to interpret the file as an executable.

2. The permission of the newly created script will need to be changed to allow the script to be executable.

```
chmod +x [filename]
```

B. LAUNCHING THE EPC

1. Start the EPC Containers Script

1. Create a script file named "launch_4g_containers.sh." The script will launch all 4G docker containers.

```
#!/bin/bash
cd ~/openair-epc-fed
# Starting the network functions with 'tshark'. Launching network functions with 'tshark' is not necessary; however, it is helpful for debugging issues.
docker exec -d prod-oai-hss /bin/bash -c "nohup tshark -i eth0 -i eth1 -w /tmp/hss_check_run.pcap 2>&1 > /dev/null"
docker exec -d prod-oai-mme /bin/bash -c "nohup tshark -i eth0 -i lo:s10 -w /tmp/mme_check_run.pcap 2>&1 > /dev/null"
```

```
docker exec -d prod-oai-spgwc /bin/bash -c "nohup tshark -i eth0 -i lo:p5c -i lo:s5c -w /tmp/spgwc_check_run.pcap 2>&1 > /dev/null"

docker exec -d prod-oai-spgwu-tiny /bin/bash -c "nohup tshark -i eth0 -w /tmp/spgwu_check_run.pcap 2>&1 > /dev/null"

docker exec -d prod-oai-hss /bin/bash -c "nohup ./bin/oai_hss -j ./etc/hss_rel14.json --reloadkey true > hss_check_run.log 2>&1"

sleep 2


# Launching each container. The order in which they are launched matters. The order should be prod-oai-hss, prod-oai-mme, prod-oai-spgwc, and prod-oai-spgwu-tiny.

docker exec -d prod-oai-mme /bin/bash -c "nohup ./bin/oai_mme -c ./etc/mme.conf > mme_check_run.log 2>&1"

sleep 2

docker exec -d prod-oai-spgwc /bin/bash -c "nohup ./bin/oai_spgwc -o -c ./etc/spgw_c.conf > spgwc_check_run.log 2>&1"

sleep 2

docker exec -d prod-oai-spgwu-tiny /bin/bash -c "nohup ./bin/oai_spgwu -o -c ./etc/spgw_u.conf > spgwu_check_run.log 2>&1"

docker exec -it prod-oai-mme tail -f mme_check_run.log
```

2. Stop the EPC Containers Script

1. Create a script file named "stop_4g_containers.sh." The script will stop all 4G docker containers, recover the log files, configurations, and traces, then zip the files.

```
!#/bin/bash

cd ~/openair-epc-fed

# Stopping each container and tcpdumb.

docker exec -it prod-oai-hss /bin/bash -c "killall --signal SIGINT oai_hss tshark tcpdump"

docker exec -it prod-oai-mme /bin/bash -c "killall --signal SIGINT oai_mme tshark tcpdump"

docker exec -it prod-oai-spgwc /bin/bash -c "killall --signal SIGINT oai_spgwc tshark tcpdump"

docker exec -it prod-oai-spgwu-tiny /bin/bash -c "killall --signal SIGINT oai_spgwu tshark tcpdump"

sleep 10
```

```
docker exec -it prod-oai-hss /bin/bash -c "killall --signal SIGKILL oai_hss tshark tcpdump"
docker exec -it prod-oai-mme /bin/bash -c "killall --signal SIGKILL oai_mme tshark tcpdump"
docker exec -it prod-oai-spgwc /bin/bash -c "killall --signal SIGKILL oai_spgwc tshark tcpdump"
docker exec -it prod-oai-spgwu-tiny /bin/bash -c "killall --signal SIGKILL oai_spgwu tshark tcpdump"
```

3. Recover the Logs, Configurations, and Traces

```
rm -Rf archives
mkdir -p archives/oai-hss-cfg archives/oai-mme-cfg archives/oai-spgwc-cfg archives/oai-spgwu-cfg

# Retrieve the modified configuration files.
docker cp prod-oai-hss:/openair-hss/etc/. archives/oai-hss-cfg
docker cp prod-oai-mme:/openair-mme/etc/. archives/oai-mme-cfg
docker cp prod-oai-spgwc:/openair-spgwc/etc/. archives/oai-spgwc-cfg
docker cp prod-oai-spgwu-tiny:/openair-spgwu-tiny/etc/. archives/oai-spgwu-cfg

# Retrieve the modified logs.
docker cp prod-oai-hss:/openair-hss/hss_check_run.log archives
docker cp prod-oai-mme:/openair-mme/mme_check_run.log archives
docker cp prod-oai-spgwc:/openair-spgwc/spgwc_check_run.log archives
docker cp prod-oai-spgwu-tiny:/openair-spgwu-tiny/spgwu_check_run.log archives

# Retrieve the packet captures (PCAP).
docker cp prod-oai-hss:/tmp/hss_check_run.pcap archives
docker cp prod-oai-mme:/tmp/mme_check_run.pcap archives
docker cp prod-oai-spgwc:/tmp/spgwc_check_run.pcap archives
docker cp prod-oai-spgwu-tiny:/tmp/spgwu_check_run.pcap archives

# Create a zip file for the data.
```

```
zip -r -qq docker_files.zip archives
```

4. Restart EPC Containers script

1. Create a script file named "restart_4g_containers.sh." The script will stop all 4G docker containers.

```
!#/bin/bash
docker container restart prod-cassandra prod-oai-hss prod-oai-mme prod-oai-spgwc prod-oai-spgwu-tiny
```

C. OPTIONAL FLAGS FOR LAUNCHING THE ENB/GNB

Running the eNB and the gNB is covered in Chapter III. This section lists the different flags that could be used to launch the eNB and gNB. Table 13 lists optional flags that could be used to launch the eNB.

Table 13. Optional flags to Launch the eNB. Adapted from [46].

eNB	
Flags	**Uses**
--basicsim	enables the eNB to execute by emulating the radio head (USRP device). It allows connecting the OAI UE and the OAI eNodeB through a network interface carrying the time-domain samples, getting rid of over the air unpredictable perturbations.
--noS1	used to inject/receive user-plane traffic over a virtual IP Point-to-Point interface, also known as a TUN interface.
--nokrnomod 0	used to enforce kernel modules instead of tun.
--nokrnomod 1	used to enforce the preferred and supported tunnel interface

Table 14 lists optional flags that could be used to launch the gNB and a brief description of their use.

Table 14. Optional flags to Launch the gNB. Adapted from [46].

gNB	
Flags	**Uses**
--ue-fo-compensation	enables the frequency offset compensation at the UE. This command is useful when running over the air or without an external clock/time source
--usrp-args	sed to identify the USRP and set some basic parameters (like the clock source)
--clock-source	sets the clock source to internal or external
--phy-test flag	used to run the UE without the support of eNB or EPC in NSA mode

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX C. INSTALLING DIAGPARSER AND MINICOM

A. INSTALLING DIAGPARSER AND MINICOM

1. Download Wireshark from their official website (https://www.wireshark.org) and install it on a Linux 16.04 LTS PC or higher. Quectel has also been tested on Windows Subsystem for Linux (WSL).

2. Install Diag Parser.

```
sudo apt-get update
sudo apt-get install libgnutls28-dev make gcc g++ libtalloc-dev autoconf automake make debhelper debscripts libtool build-essential pkg-config libpsclite-dev -y
```

3. Install Diag Parser

```
git clone git://github.com/moiji-mobile/diag-parser
cd diag-parser
. /build/build_local.sh
```

4. Install Minicom

```
sudo apt-get install minicom -y
```

5. Install Android Tools

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

6. Install NetCat.

```
sudo apt-get install netcat
```

B. CONNECTING TO QUECTEL MODULE

1. Install xdotool

```
sudo apt update

sudo apt install xdotool -y
```

2. Create a script to collect traffic. This script will launch five separate terminal tabs—each performing a different function.

```
#!/bin/bash


# Terminal 1—Start monitoring traffic on port 4729.

gnome-terminal --tab

xdotool key Control+Page_Down

cd

xdotool key KP_Enter

xdotool type nc\ -u\ -l\ -p\ 4729

xdotool key KP_Enter

xdotool sleep 10


# Terminal 2 - Launch Wireshark from CLI and monitor port 4729.

gnome-terminal --tab

xdotool key Control+Page_Down

xdotool type cd\ ~/Documents/capstone

xdotool key KP_Enter

xdotool type ./1terminal.sh

xdotool key KP_Enter
```

```
xdotool type sudo\ wireshark\ -i\ lo\ -f"""port\ 4729"""\ -k

xdotool key KP_Enter

xdotool type [password]

xdotool key KP_Enter

xdotool sleep 15


# Terminal 3 - Monitor DiagParser traffic

gnome-terminal --tab

xdotool key Control+Page_Down

xdotool type sudo\ minicom\ -s

xdotool key KP_Enter

xdotool type [password]

xdotool key KP_Enter

xdotool sleep 3


# Terminal 4—Use DiagParser to decode Qualcomm DIAG format from the Quectel Module.

gnome-terminal --tab

xdotool key Control+Page_Down

xdotool type cd\ ~/diag-parser

xdotool key KP_Enter

xdotool type sudo\ ./diag_parser\ -g\ 127.0.0.1\ -i\ /dev/ttyUSB0\ -v

xdotool key KP_Enter

xdotool type [password]

xdotool key KP_Enter
```

```
xdotool sleep 3

# Terminal 5 - Send commands to the modem.
gnome-terminal --tab
xdotool key Control+Page_Down
xdotool type sudo\ minicom\ -s
xdotool key KP_Enter
xdotool type [password]
xdotool key KP_Enter
```

3. The next steps are performed manually on the Linux terminal's Tabs 3 and 5. First, the minicom virtual port will need to be changed to /dev/ttyUSB2 to send ATtention (AT) commands from the modem. AT commands are a set of instructions used to control the modem.

3.a. Type "sudo minicom-s" in the terminal and enter the password if prompted.

3.b. Use the keyboard to select "serial port setup."

3.c. Select Serial Device by typing "A."

3.d. Change "/dev/tty8" to "/dev/ttyUSB2".

3.e. Press "enter" on the keyboard twice.

3.f. Select "Exit."

After completing step 3, AT Commands were used to reconfigure the Quectel modem or query the modem for information. Alternatively, a simpler method of sending AT commands to the Quectel modem is to use QNavigator.

C. AT COMMANDS

Table 15 list the common AT commands that were used when interacting with the Quectel Module using minicom. A comprehensive list of AT Commands can be found at [47].

Table 15. AT Commands Used

AT Commands		
Commands	**Description**	**Output**
ATE1	display user input.	echoes [user input]
AT+QCCID	show ICCID	+QCCID: 8991920123456789000F
AT+QOPN	displays operator names	[list operators]
AT+QNWINFO	Query Network Information	+QNWINFO: “FDD LTE,” “311480,” “LTE BAND 5,” 2561
AT+CGREG	Network Registration Status	OK
AT+GMI	request manufacturer information	Quectel
AT+GMM	request TA model identification	EC20
AT+GMR	request TA revision identification software release	EC20EQAR02A13E2G

Table 16 is a quick reference of the minicom command. More about minicom can be found by performing the manual command (e.g., man minicom). The man page can be found online at [48].

Table 16. Minicom Quick Command Reference

Minicom	
Commands	**Description**
Ctrl+a, c	clears the screen
Ctrl+a, e	local echo on/off
Ctrl+a, i	cursor key mode
Ctrl+a, j	suspend minicom
Ctrl+a, m	initialize modem

Minicom	
Commands	**Description**
Ctrl+a, o	configure minicom modem
Ctrl+a, q	quits minicom
Ctrl+a, s	send files
Ctrl+a, u	add carriage return
Ctrl+a, y	paste files
Ctrl+a, z	brings up minicom command summary

APPENDIX D. COMMAND LINE CHEAT SHEET

This appendix lists various Linux and Docker commands we used in the methodology and simulation phase of this capstone. First, the commonly used Linux command is listed in Section 1, the Docker commands are listed in Section 2.

A. COMMON LINUX COMMAND LINE USED

This section covers useful Linux commands for this capstone, as shown in Table 17. The commands covered in this section are not inclusive of the commands used during this project and covers a list of common commands and commands that make debugging and troubleshooting easier. More information from each command can be found from its respective Linux manual page. To find a manual page for a command enter the command followed by "man." To launch the manual page for the "man" command, enter "man man" in the Linux terminal.

Table 17. Useful Linux Commands Used. Source [49].

Commands	Description	Example Use on Capstone
apt list	filter list available apt packages for a specific package.	apt list | grep [package_name]
cat	concatenate files to output (default is screen). The command is used to read the data in a file.	cat [filename]
brctl	managed bridged virtual networks. delbr option deletes the network	brctl delbr [virtual_network_name]
chmod	modify file permission. -x switch the file to execution permission.	chmod +x [filename]
df	displays information of device name, total blocks, total disk space, used disk space, available disk space, and mount point one a file system. -h displays the information in a human-readable format [50].	df -h

Commands	Description	Example Use on Capstone
du	displays information on disk usage. --max-depth-N prints the entire directory if it is N or fewer levels below the command line argument. -h displays the information in a human-readable format. du redirects the output of the command to more. more print the information one screen at a time.	--max-depth=[N] -h | more
gedit	Launch the gedit text editor.	gedit [filename] &
grep	command-line utility for searching for plain text. - is recursive. -n is the line number. -w is to match the whole word. -e is the pattern used to search.	grep -rnw “[path]” -e “[string]”
ifconfig	View and configure the kernel-resident network interfaces.	ifconfig
ip link	configure the virtual networks on the host machine. The set option establishes the virtual bridge up or down. The del option deletes the virtual network	sudo ip link set [virtual_link_name] [up or down] sudo ip link del [virtual_link_name]
ip route	list all of the route entries in the kernel.	sudo ip route
ip route add	add a route to the destination ipv4 network via the local gateway.	sudo ip route add [ip4_address] via gateway [GW_ipv4_address]
ip route del	delete a route for destination network via interface device.	sudo ip route del [ipv4_address] via [local_ipv4_interface_address] dev enp0s25
ip route flush	removes all of the current routes in the routing table.	sudo ip route flush table main
ls	display a list of files and folders in a given directory. The -1 option list each file and folder on a new line.	ls [option]
newgrp	update group ID during a login session. prevent having to log off after updating the group with the current user logged in.	newgrp [group name]
ping	used for troubleshooting network connections. -c [N] sends N number of ping requests to the destination.	ping -c [N] [ipv4_address]
ps -a	displays a list of processes running on the system. -A displays all	ps -A

Commands	Description	Example Use on Capstone
	processes currently running on the system.	
pwd	print a copy of the working directory	pwd
reboot	restarts the host machine.	reboot
route	display and alter the routing table.	route
sudo	used to run a Linux command with elevated privileges.	sudo [command]
	add group to the sudoers list	group add [group name]
	add a user to a group. -a appends a user to a group. -G modifies additional user groups.	usermod -aG [groupname] [username]
service	starts, stop, or restart a particular service.	sudo service [service_name] start sudo service [service_name] stop sudo service [service_name] restart
sleep	suspend the program execution for a specified amount of time	sleep [time]
top	displays a dynamic list of a real-time view of processes currently running on the system.	top

B. USEFUL DOCKER COMMAND LINE USED

This section covers useful docker commands for this capstone, as detailed in Table 18. The commands covered in this section are not inclusive of the commands used during this project and cover a list of common commands and commands that make debugging and troubleshooting easier.

Table 18. Useful Docker Commands Used. Source [51].

Commands		Description	Example
docker container	inspect	display detailed information about one or more containers	docker container inspect [options]
	prune	remove all unused containers	docker container prune --force
	restart	restart one or more container	docker container restart [options]
	rm	remove one or more containers	docker container rm [options]

Commands		Description	Example
	stats	display a live stream of container(s) resource usage statistics	docker container stats [container name]
	start	start one or more stopped containers	docker container start [options]
	stop	stop one or more running containers	docker container stop [options]
docker image	ls	list images	docker image ls
	prune	remove all unused images	docker image prune --force
	rm	remove one or more images	docker image rm [options] --force
docker	exec	enables command line interface within the docker container	docker exec -it [containter name] /bin/bash
		view status of cassandra database	docker exec -it prod-cassandra /bin/ bash -c “nodetool status”
	help	detailed information about a specific docker command	docker [option] --help
docker network		connect a container to a network	docker network connect [options] network container
	disconnect	disconnect a container from a network	docker network disconnect [options] network container
	inspect	display detailed information about one or more networks	docker network inspect [options]
	ls	list networks	docker network ls [options]
docker network	prune	remove all unused networks	docker network prune --force
	rm	remove one or more networks	docker network rm [options] --force
docker	rm	remove one or more containers	docker rm [options] --force
docker	ps	list containers	docker ps [options]
docker	version	show docker version information	docker version

Docker - Uninstall	
Commands	**Description**
dpkg -l | grep -i docker	identify which package you have installed
sudo apt-get purge -y docker-engine docker docker.io docker-ce docker-ce-cli sudo apt-get autoremove -y --purge docker-engine docker docker.io docker-ce	remove images, containers, volumes, or user-created configuration files
sudo rm -rf /var/lib/docker /etc/docker sudo rm /etc/apparmor.d/docker sudo groupdel docker sudo rm -rf /var/run/docker.sock	remove docker from the system completely

APPENDIX E. LATENCY RESULTS

This appendix shows the results from the ping tests performed from the OnePlus 6T and the Google Pixel 4a 5G while connected to the OpenAirInterface 4G LTE network.

Table 19. Overall Ping Results between OnePlus 6T and Google Pixel 4a 5G

OnePlus 6T				
Success	**Loss rate**	**Avg RTT**	**Best RTT**	**Worst RTT**
0.987	0.013	38.72	27.0	76.87

Google Pixel				
Success	**Loss rate**	**Avg RTT**	**Best RTT**	**Worst RTT**
0.989	0.011	42.114	29.6	81.3

Table 20. Ping Results

OnePlus 6T					
Ping Test to Google DNS Server (8.8.8.8)					
Success	**Loss rate**	**Avg RTT**	**Best RTT**	**Worst RTT**	**std. Dev**
95%	5%	38.94	29.6	52.9	4.85
99%	1%	37.32	28.2	65	7.2
99%	1%	37.58	28.7	57.5	5.98
100%	0%	39.91	27.4	79.9	9.38
97%	3%	39	29	58.8	5.15
98%	2%	37.8	28.2	58.3	5.62
99%	1%	39.46	29.1	59.8	6.26
100%	0%	38.8	27.9	68.8	7.06
98%	2%	38.31	29.6	66.4	6.45
95%	5%	38.48	28.9	59.7	5.11
Ping Test to Cloudfare (1.1.1.1)					
Success	**Loss rate**	**Avg RTT**	**Best RTT**	**Worst RTT**	**std. Dev**
98%	2%	40.47	29	69.7	6.97
97%	3%	40.64	29.1	69.3	7.64
99%	1%	39.28	29.1	58.3	5.55
100%	0%	40.64	30.5	69.2	7.75
99%	1%	41.52	29.5	65.4	7.57
97%	3%	38.95	30.2	52.3	5.2
99%	1%	40.05	29.6	68.1	7.65
100%	0%	38.87	29.1	60.6	6.47
100%	0%	37.72	28.7	56.1	5.8
99%	1%	42.22	29.4	68.1	9.23
Ping Test to ATT Primary DNS Server (68.94.156.1)					
Success	**Loss rate**	**Avg RTT**	**Best RTT**	**Worst RTT**	**std. Dev**
95%	2%	37.9	27	72.3	6.31
100%	0%	36.92	27.3	54.2	4.52
100%	0%	37.63	29	65.2	6.79
100%	0%	36.39	28.5	56.6	4.64
100%	0%	37.27	28.1	64.7	5.24
98%	2%	39.01	28.1	67.9	28.1
97%	3%	37.12	27.9	76.2	7.03
100%	0%	37.03	27.8	57.8	5.75
100%	0%	38.22	30.4	53.9	4.22
100%	0%	38.17	28	73.3	7.99
99%	1%	37.566	28.21	64.21	8.059

Ping Test to Google Pixel					
Google DNS Server (8.8.8.8)					
Success	**Loss rate**	**Avg RTT**	**Best RTT**	**Worst RTT**	**std. Dev**
97%	3%	40.82	30.9	51.8	3.89
99%	1%	41.28	32	67.4	5.56
100%	0%	42.22	32.4	73.9	4.94
96%	4%	42.26	32.2	70.2	5.84
99%	1%	42.19	29.8	78.9	7.91
100%	0%	41.97	29.8	74.1	6.61
100%	0%	43.85	29.6	78.7	8.51
100%	0%	43.71	30.6	78.6	8.38
100%	0%	42.13	31.2	67.4	5.84
97%	3%	41.89	30.9	68	5.36
Ping Test to Cloudfare (1.1.1.1)					
Success	**Loss rate**	**Avg RTT**	**Best RTT**	**Worst RTT**	**std. Dev**
98%	2%	42.85	31.9	59.4	4.94
100%	0%	43.02	32.8	65.5	5.57
100%	0%	42.71	32.5	71.9	6.1
100%	0%	42.59	32	58.1	4.84
100%	0%	41.34	30.9	51.3	4.05
99%	1%	42.09	31.1	57	4.6
97%	3%	41.98	31.8	50.5	4.06
100%	0%	41.35	32	49	3.84
100%	0%	43.18	33.2	76.1	5.52
100%	0%	42.26	31.4	69.8	5.66
Ping Test to ATT Primary DNS Server (68.94.156.1)					
Success	**Loss rate**	**Avg RTT**	**Best RTT**	**Worst RTT**	**std. Dev**
97%	3%	41.88	31.1	55.1	4.54
97%	3%	41.74	29.8	59.2	4.68
100%	0%	41.65	30	74.1	7.29
98%	2%	41.64	30.5	75.3	7.41
99%	1%	42.4	31.4	70.3	6.89
100%	0%	39.66	29.6	46.8	3.56
97%	3%	41.45	31.9	54.2	4.42
97%	3%	42.12	32.3	68	5.81
100%	0%	42.69	30.8	81.3	8.82
100%	0%	42.5	30.5	72.8	7.25
99%	2%	41.773	30.79	65.71	6.067

APPENDIX F. IPERF RESULTS

A. IPERF TEST FROM UE TO EPC

```
------------------------------------------------------------
Server listening on TCP port 5001
Binding to local address 192.168.1.7
TCP window size:  128 KByte (default)
------------------------------------------------------------
```

Table 21. Full iPerf Results from UE to EPC

Server				UE
[ID]	Interval	Transfer	Bandwidth	Bandwidth
[4]	0.0- 9.9 sec	1.00 MBytes	850 Kbits/sec	962 Kbits/sec
[4]	0.0-46.2 sec	1.00 MBytes	182 Kbits/sec	24.9 Mbits/sec
[4]	0.0- 1.1 sec	1.00 MBytes	7.43 Mbits/sec	22.6 Mbits/sec
[4]	0.0-164.4 sec	1.00 MBytes	7.16 Kbits/sec	25.5 Mbits/sec
[4]	0.0- 1.1 sec	1.00 MBytes	7.47 Mbits/sec	29.8 Mbits/sec

B. IPERF TEST FROM UE TO ENB

```
------------------------------------------------------------
Server listening on TCP port 5001
Binding to local address 192.168.1.8
TCP window size:  128 KByte (default)
------------------------------------------------------------
```

Table 22. Full iPerf Results from UE to eNB

Server				UE
[ID]	Interval	Transfer	Bandwidth	Bandwidth
[4]	0.0- 1.1 sec	1.00 MBytes	7.48 Mbits/sec	30.8 Mbits/sec
[4]	0.0- 2.5 sec	1.00 MBytes	3.30 Mbits/sec	5.15 Mbits/sec
[4]	0.0-18.0 sec	1.00 MBytes	465 Kbits/sec	5.7 Mbits/sec
[4]	0.0- 1.3 sec	1.00 MBytes	6.29 Mbits/sec	25.3 Mbits/sec
[4]	0.0-85.6 sec	1.00 MBytes	98.0 Kbits/sec	32.9 Mbits/sec

C. IPERF TEST FROM UE TO GNB

The iPerf tests from the Google Pixel 4a 5G to the gNB cause the eNB to experience stack smashing. Even when the iPerf tests were completed, the iPerf result on the server (gNB) did not display the server's bandwidth.

```
------------------------------------------------------------
Server listening on TCP port 5001
TCP window size:  128 KByte (default)
------------------------------------------------------------
```

Table 23. iPerf Results from UE to gNB

Server				**UE**
[ID]	Interval	Transfer	Bandwidth	Bandwidth
[3]	0.0 - 96.8 sec	1 Mbytes	-	86 Kbits/sec
[3]	0.0 - 63.3 sec	1 Mbytes	-	1.33 Kbits/sec
[3]	0.0 - 66.7 sec	1 Mbytes	-	126 Kbits/sec

APPENDIX G. 4G AND 5G FREQUENCY OPERATING BAND CHARTS

Table 24. 4G LTE Operating Bands. Source [52].

E-UTRA Operating Band	Uplink (UL) operating band BS receive / UE transmit	Downlink (DL) operating band BS transmit / UE receive	Duplex Mode
1	1920 MHz—1980 MHz	2110 MHz—2170 MHz	FDD
2	1850 MHz—1910 MHz	1930 MHz—1990 MHz	FDD
3	1710 MHz—1785 MHz	1805 MHz—1880 MHz	FDD
4	1710 MHz—1755 MHz	2110 MHz—2155 MHz	FDD
5	824 MHz—849 MHz	869 MHz—894MHz	FDD
6[1]	830 MHz—840 MHz	875 MHz—885 MHz	FDD
7	2500 MHz—2570 MHz	2620 MHz—2690 MHz	FDD
8	880 MHz—915 MHz	925 MHz—960 MHz	FDD
9	1749.9 MHz—1784.9 MHz	1844.9 MHz—1879.9 MHz	FDD
10	1710 MHz—1770 MHz	2110 MHz—2170 MHz	FDD
11	1427.9 MHz—1447.9 MHz	1475.9 MHz—1495.9 MHz	FDD
12	699 MHz—716 MHz	729 MHz—746 MHz	FDD
13	777 MHz—787 MHz	746 MHz—756 MHz	FDD
14	788 MHz—798 MHz	758 MHz—768 MHz	FDD
15	Reserved	Reserved	FDD
16	Reserved	Reserved	FDD
17	704 MHz—716 MHz	734 MHz—746 MHz	FDD
18	815 MHz—830 MHz	860 MHz—875 MHz	FDD
19	830 MHz—845 MHz	875 MHz—890 MHz	FDD
20	832 MHz—862 MHz	791 MHz—821 MHz	FDD
21	1447.9 MHz—1462.9 MHz	1495.9 MHz—1510.9 MHz	FDD
22	3410 MHz—3490 MHz	3510 MHz—3590 MHz	FDD
23[1]	2000 MHz—2020 MHz	2180 MHz—2200 MHz	FDD
24	1626.5 MHz—1660.5 MHz	1525 MHz—1559 MHz	FDD
25	1850 MHz—1915 MHz	1930 MHz—1995 MHz	FDD
26	814 MHz—849 MHz	859 MHz—894 MHz	FDD
27	807 MHz—824 MHz	852 MHz—869 MHz	FDD
28	703 MHz—748 MHz	758 MHz—803 MHz	FDD
29	N/A	717 MHz—728 MHz	FDD[2]
30	2305 MHz—2315 MHz	2350 MHz—2360 MHz	FDD
31	452.5 MHz—457.5 MHz	462.5 MHz—467.5 MHz	FDD
32	N/A	1452 MHz—1496 MHz	FDD[2]
33	1900 MHz—1920 MHz	1900 MHz—1920 MHz	TDD
34	2010 MHz—2025 MHz	2010 MHz—2025 MH	TDD
35	1850 MHz—1910 MHz	1850 MHz—1910 MHz	TDD

36	1930 MHz—1990 MHz	1930 MHz—1990 MHz	TDD
37	1910 MHz—1930 MHz	1910 MHz—1930 MHz	TDD
38	2570 MHz—2620 MHz	2570 MHz—2620 MHz	TDD
39	1880 MHz—1920 MHz	1880 MHz—1920 MHz	TDD
40	2300 MHz—2400 MHz	2300 MHz—2400 MHz	TDD
41	2496 MHz 2690 MHz	2496 MHz 2690 MHz	TDD
42	3400 MHz—3600 MHz	3400 MHz—3600 MHz	TDD
43	3600 MHz—3800 MHz	3600 MHz—3800 MHz	TDD
44	703 MHz—803 MHz	703 MHz—803 MHz	TDD
45	1447 MHz—1467 MHz	1447 MHz—1467 MHz	TDD
46	5150 MHz—5925 MHz	5150 MHz—5925 MHz	TDD[8,9]
47	5855 MHz—5925 MHz	5855 MHz—5925 MHz	TDD
48	3550 MHz—3700 MHz	3550 MHz—3700 MHz	TDD
…			
64	Reserved		
65	1920 MHz—2010 MHz	2110 MHz—2200 MHz	FDD
66	1710 MHz—1780 MHz	2110 MHz—2200 MHz	FDD
67	N/A	738 MHz—758 MHz	FDD[2]
68	698 MHz—728 MHz	753 MHz—783 MHz	FDD
69	N/A	2570 MHz—2620 MHz	FDD
70	1695 MHz—1710 MHz	1995 MHz—2020 MHz	FDD[2]

NOTE 1:	Band 6, 23 is not applicable
NOTE 2:	Restricted to E-UTRA operation when carrier aggregation is configured. The downlink operating band is paired with the uplink operating band (external) of the carrier aggregation configuration that is supporting the configured Pcell.
NOTE 3:	A UE that complies with the E-UTRA Band 65 minimum requirements in this specification shall also comply with the E-UTRA Band 1 minimum requirements.
NOTE 4:	The range 2180–2200 MHz of the DL operating band is restricted to E-UTRA operation when carrier aggregation is configured.
NOTE 5:	A UE that supports E-UTRA Band 66 shall receive in the entire DL operating band
NOTE 6:	A UE that supports E-UTRA Band 66 and CA operation in any CA band shall also comply with the minimum requirements specified for the DL CA configurations CA_66B, CA_66C and CA_66A-66A
NOTE 7:	A UE that complies with the E-UTRA Band 66 minimum requirements in this specification shall also comply with the E-UTRA Band 4 minimum requirements.
NOTE 8:	This band is an unlicensed band restricted to licensed-assisted operation using Frame Structure Type 3
NOTE 9:	In this version of the specification, restricted to E-UTRA DL operation when carrier aggregation is configured.

NOTE 10:	The range 2010–2020 MHz of the DL operating band is restricted to E-UTRA operation when carrier aggregation is configured and TX-RX separation is 300 MHz The range 2005–2020 MHz of the DL operating band is restricted to E-UTRA operation when carrier aggregation is configured and TX-RX separation is 295 MHz

Table 25. 5G NR FR1 Operating Bands. Source [53].

NR Operating Band	Uplink (UL) operating band BS receive / UE transmit	Downlink (DL) operating band BS transmit / UE receive	Duplex Mode
n1	1920 MHz—1980 MHz	2110 MHz—2170 MHz	FDD
n2	1850 MHz—1910 MHz	1930 MHz—1990 MHz	FDD
n3	1710 MHz—1785 MHz	1805 MHz—1880 MHz	FDD
n5	824 MHz—849 MHz	869 MHz—894 MHz	FDD
n7	2500 MHz—2570 MHz	2620 MHz—2690 MHz	FDD
n8	880 MHz—915 MHz	925 MHz—960 MHz	FDD
n12	699 MHz—716 MHz	729 MHz—746 MHz	FDD
n14	788 MHz—798 MHz	758 MHz—768 MHz	FDD
n18	815 MHz—830 MHz	860 MHz—875 MHz	FDD
n20	832 MHz—862 MHz	791 MHz—821 MHz	FDD
n25	1850 MHz—1915 MHz	1930 MHz—1995 MHz	FDD
n26	814 MHz—849 MHz	859 MHz—894 MHz	FDD
n28	703 MHz—748 MHz	758 MHz—803 MHz	FDD
n29	N/A	717 MHz—728 MHz	SDL
n30[3]	2305 MHz—2315 MHz	2350 MHz—2360 MHz	FDD
n34	2010 MHz—2025 MHz	2010 MHz—2025 MHz	TDD
n38[10]	2570 MHz—2620 MHz	2570 MHz—2620 MHz	TDD
n39	1880 MHz—1920 MHz	1880 MHz—1920 MHz	TDD
n40	2300 MHz—2400 MHz	2300 MHz—2400 MHz	TDD
n41	2496 MHz—2690 MHz	2496 MHz—2690 MHz	TDD
n46	5150 MHz—5925 MHz	5150 MHz—5925 MHz	TDD[13]
n47[11]	5855 MHz—5925 MHz	5855 MHz—5925 MHz	TDD
n48	3550 MHz—3700 MHz	3550 MHz—3700 MHz	TDD
n50	1432 MHz—1517 MHz	1432 MHz—1517 MHz	TDD[1]
n51	1427 MHz—1432 MHz	1427 MHz—1432 MHz	TDD
n53	2483.5 MHz—2495 MHz	2483.5 MHz—2495 MHz	TDD
n65	1920 MHz—2010 MHz	2110 MHz—2200 MHz	FDD[4]
n66	1710 MHz—1780 MHz	2110 MHz—2200 MHz	FDD
n70	1695 MHz—1710 MHz	1995 MHz—2020 MHz	FDD
n71	663 MHz—698 MHz	617 MHz—652 MHz	FDD
n74	1427 MHz—1470 MHz	1475 MHz—1518 MHz	FDD
n75	N/A	1432 MHz—1517 MHz	SDL

n76	N/A	1427 MHz—1432 MHz	SDL
n77[12]	3300 MHz—4200 MHz	3300 MHz—4200 MHz	TDD
n78	3300 MHz—3800 MHz	3300 MHz—3800 MHz	TDD
n79	4400 MHz—5000 MHz	4400 MHz—5000 MHz	TDD
n80	1710 MHz—1785 MHz	N/A	SUL
n81	880 MHz—915 MHz	N/A	SUL
n82	832 MHz—862 MHz	N/A	SUL
n83	703 MHz—748 MHz	N/A	SUL
n84	1920 MHz—1980 MHz	N/A	SUL
n86	1710 MHz—1780 MHz	N/A	SUL
n89	824 MHz—849 MHz	N/A	SUL
n90	2496 MHz—2690 MHz	2496 MHz—2690 MHz	TDD[5]
n91	832 MHz—862 MHz	1427 MHz—1432 MHz	FDD[9]
n92	832 MHz—862 MHz	1432 MHz—1517 MHz	FDD[9]
n93	880 MHz—915 MHz	1427 MHz—1432 MHz	FDD[9]
n94	880 MHz—915 MHz	1432 MHz—1517 MHz	FDD[9]
n95[8]	2010 MHz—2025 MHz	N/A	SUL
n96[14]	5925 MHz—7125 MHz	5925 MHz—7125 MHz	TDD[13]

NOTE 1:	UE that complies with the NR Band n50 minimum requirements in this specification shall also comply with the NR Band n51 minimum requirements
NOTE 2:	UE that complies with the NR Band n75 minimum requirements in this specification shall also comply with the NR Band n76 minimum requirements.
NOTE 3:	Uplink transmission is not allowed at this band for UE with external vehicle mounted antennas.
NOTE 4:	A UE that complies with the NR Band n65 minimum requirements in this specification shall also comply with the NR Band n1 minimum requirements.
NOTE 5:	Unless otherwise stated, the applicability of requirements for Band n90 is in accordance with that for Band n41; a UE supporting Band n90 shall meet the requirements for Band n41. A UE supporting Band n90 shall also support band n41.
NOTE 6:	A UE that supports NR Band n66 shall receive in the entire DL operating band.
NOTE 7:	A UE that supports NR Band n66 and CA operation in any CA band shall also comply with the minimum requirements specified for the DL CA configurations CA_n66B and CA_n66(2A) in the current version of the specification.
NOTE 8:	This band is applicable in China only.

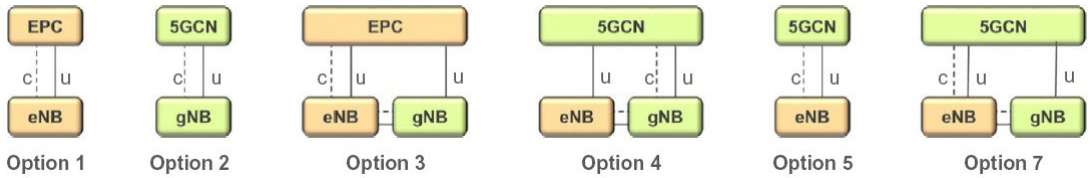
NOTE 9:	Variable duplex operation does not enable dynamic variable duplex configuration by the network, and is used such that DL and UL frequency

	ranges are supported independently in any valid frequency range for the band.
NOTE 10:	When this band is used for V2X SL service, the band is exclusively used for NR V2X in particular regions.
NOTE 11:	This band is unlicensed band used for V2X service. There is no expected network deployment in this band.
NOTE 12:	In the USA this band is restricted to 3450—3550 MHz and 3700—3980 MHz
NOTE 13:	This band is restricted to operation with shared spectrum channel access as defined in 37.213.
NOTE 14:	This band is applicable in the USA only subject to FCC Report and Order FCC 20- 51

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX H. 3GPP CORE NETWORKS RADIO-ACCESS TECHNOLOGY

This appendix outlines the different radio-access deployment technologies. The different 3GPP compliant deployment models are depicted in Figure 25.



"c" indicates the control plane (dotted line) and "u" (solid line) indicate the user plane.

Figure 25. Core Networks Deployment Models. Source [54].

- Option 1 is the LTE network consisting of the EPC as the core and eNB as the base station. "Network Configuration 1" and "Network Configuration 2" are Option 1.
- Option 2 consists of the 5G Core Network (5GCN) as the core and the gNB as the base station.
- Option 3 consists of the EPC as the core and the eNB and gNB as the base stations. "Network Configuration 3" is set up as Option 3. In Option 3, the eNB handles the control plane and user plane functions, and the gNB handles the user plane functions.
- Option 4 consists of the 5GCN as the core and the eNB and gNB as the base station. In Option 4, the eNB handles only user plane functions, and the gNB handles user plane and control plane functions.
- Option 5 consists of the 5GCN as the core and an eNB as the base station.

- Option 7 consists of the 5GCN as the core and the eNB and gNB as the base stations. In Option 7, the eNB handles the control and user plane functions, and the gNB handles only user plane functions.

SUPPLEMENTAL FILE STRUCTURE

This appendix provides lists the supplemental files provided with this research.

A. HSS FILE STRUCTURE

"acl.conf"	Configuration file for the peer whitelist extension.
"cacert.pem"	File containing the HSS CA certificate.
"hss.cert.pem"	File containing the HSS public key.
"hss_rel14.conf"	MySQL cassandra_Server_IP address configuration file.
"hss_rel14.json"	Data structure and objects for HSS configurations.
"hss_rel14_fd.conf"	Data structure and objects for freeDiameter configurations.
"oss.json"	Data structure and objects for logging.

B. MME FILE STRUCTURE

"mme.cacert.pem"	File containing the MME CA certificate.
"mme.cakey.pem"	File containing the MME private key.
"mme.cert.pem"	File containing the MME public key.
"mme.conf"	MME configuration file.
"mme.csr.pem"	File containing the MME certificate signing request.
"mme_fd.conf"	freeDiameter configuration for the MME.
"mme-cfg.sh"	MME configuration file.

C. SPGW-C FILE STRUCTURE

"spgwc-cfg.sh"	File containing the current SPGW-C configurations.

D. SPGW-U FILE STRUCTURE

"spgwu-cfg.sh" File containing the current SPGW-U configurations.

LIST OF REFERENCES

[1] M. Goldstein, "Next-generation wireless overview & outlook," *Slideshare*, Jul. 25, 2020. https://www.slideshare.net/markgirc/nextgeneration-wireless-overview-outlook-update-72520

[2] Anritsu, "Key technical aspects of 5G-NR," 2021. https://www.anritsu.eu/5G-Poster.html?utm_campaign=5G_poster&utm_medium=banner&utm_source=website&utm_content=hero_banner (accessed Nov. 03, 2021).

[3] U.S. Department of Defense, "DOD Announces $600 Million for 5G Experimentation and Testing at Five Installations," Oct. 08, 2020. https://www.defense.gov/News/Releases/Release/Article/2376743/dod-announces-600-million-for-5g-experimentation-and-testing-at-five-installati/

[4] N. Bhandari, S. Devra, and K. Singh, "Evolution of Cellular Network: From 1G to 5G," *International Journal of Engineering and Techniques*, vol. 3, no. 5, p. 9, Oct. 2017.

[5] M. Obaidat, F. Zarai, and P. Nicopolitidis, "Modeling and Simulation of Computer Networks and Systems - 1st Edition," *Elsevier*, Apr. 21, 2015. https://www.elsevier.com/books/modeling-and-simulation-of-computer-networks-and-systems/obaidat/978-0-12-800887-4

[6] Liane Cassavoy, "How Fast Is 4G LTE Wireless Service?," *Lifewire*, Feb. 11, 2021. https://www.lifewire.com/how-fast-is-4g-wireless-service-577566

[7] Pew Research Center, "Demographics of Mobile Device Ownership and Adoption in the United States," *Pew Research Center*, Apr. 07, 2021. https://www.pewresearch.org/internet/fact-sheet/mobile/?menuItem=011fca0d-9756-4f48-b352-d58f343696bf

[8] D. Chandramouli, Rainer Liebhart, and Juho Pirskanen, *5G for the Connected World*, First edition. Hoboken, NJ: JohnWiley & Sons Ltd, 2019.

[9] F. Mazda, *Telecommunications Engineer's Reference Book*. Oxford ; Boston: Butterworth-Heinemann, 1993.

[10] C. Cox, *An Introduction to LTE: LTE, LTE-Advanced, SAE, VoLTE and 4G Mobile Communications*, Second Edition. West Sussex, United Kingdom: John Wiley & Sons, Incorporated, 2012.

[11] 3GPP, "Non-Access Stratum (NAS)," *NAS*. https://www.3gpp.org/technologies/keywords-acronyms/96-nas (accessed Nov. 02, 2021).

[12] S. Yi, S. Chun, Y. Lee, S. Park, and S. Jung, *Radio Protocols for LTE and LTE-Advanced: Yi/Radio Protocols for LTE and LTE-Advanced*. Chichester, UK: John Wiley & Sons, Ltd, 2012. doi: 10.1002/9781118188545.

[13] E. Dahlman, S. Parkvall, and J. Sköld, *4G LTE/LTE-advanced for mobile broadband*, Second edition. Amsterdam ; New York: Elsevier, 2011. [Online]. Available: https://ebookcentral.proquest.com/lib/ebook- nps/ reader.action?docID=680868

[14] "LTE; Evolved Universal Terrestrial Radio Access (E-UTRA); Radio Link Control (RLC) protocol specification (3GPP TS 36.322 version 16.0.0 Release 16)." Accessed: Nov. 15, 2021. [Online]. Available: https://www.etsi.org/deliver/ etsi_ts/136300_136399/136322/16.00.00_60/ts_136322v160000p.pdf

[15] "Universal Mobile Telecommunications System (UMTS); LTE; 5G; Non-Access-Stratum (NAS) protocol for Evolved Packet System (EPS); Stage 3 (3GPP TS 24.301 version 15.8.0 Release 15)." Accessed: Nov. 15, 2021. [Online]. Available: https://www.etsi.org/deliver/etsi_ts/124300_124399/124301/ 15.08.00_60/ts_124301v150800p.pdf

[16] F. Kaltenberger, A. P. Silva, A. Gosain, L. Wang, and T.-T. Nguyen, "OpenAirInterface: Democratizing innovation in the 5G Era," *Comput. Netw.*, vol. 176, p. 107284, Jul. 2020, doi: 10.1016/j.comnet.2020.107284.

[17] The Plone Foundation, "End-to-end LTE/EPC network with OpenAirInterface (OAI) simulated eNB/UE and OAI's EPC — Emulab." https://wiki.emulab.net/ wiki/phantomnet/oepc-protected/oai-sim-epc (accessed Jul. 09, 2021).

[18] M. J. Lanoue, "Spoofed Networks: Exploitation of GNSS Security Vulnerability in 4G and 5G Mobile Networks," Thesis, Naval Postgraduate School, Monterey, CA, 2021. [Online]. Available: https://calhoun.nps.edu/bitstream/handle/10945/ 67451/ 21Jun_Lanoue_Matthew_Needs%20Supplemental.pdf?sequence=4&isAllowed=y

[19] N. Nikaein *et al.*, "Demo: OpenAirInterface: an open LTE network in a PC," in *Proceedings of the 20th annual international conference on Mobile computing and networking*, Maui Hawaii USA, Sep. 2014, pp. 305–308. doi: 10.1145/ 2639108.2641745.

[20] A. Aijaz, B. Holden, and F. Meng, "Open and Programmable 5G Network-in-a-Box: Technology Demonstration and Evaluation Results," *ArXiv210411074 Cs*, Apr. 2021, Accessed: Jul. 22, 2021. [Online]. Available: http://arxiv.org/abs/ 2104.11074

[21] C.-C. Chen, R.-G. Cheng, C.-Y. Ho, M. Kanj, B. Mongazon-Cazavet, and N. Nikaein, "Prototyping of Open Source NB-IoT Network," in *GLOBECOM 2020 - 2020 IEEE Global Communications Conference*, Taipei, Taiwan, Dec. 2020, pp. 1–5. doi: 10.1109/GLOBECOM42002.2020.9322398.

[22] S. Fonyi, "Overview of 5G Security and Vulnerabilities," *Cyber Def. Rev.*, vol. 5, no. 1, pp. 117–134, 2020.

[23] Range Networks, "OpenBTS-UMTS," 2014. http://openbts.org/w/index.php?title=OpenBTS-UMTS

[24] FreedomFi, "Order FreedomFi Gateway(s)," *FreedomFi*, 2021. https://freedomfi.com/product/

[25] Software Radio Systems, *Benefits of Choosing SRS*. Software Radio Systems, 2021. [Online]. Available: https://www.srs.io/

[26] doxygen, *free5GRAN*. Accessed: Nov. 14, 2021. [Online]. Available: https://free5g.github.io/free5GRAN-documentation/index.html

[27] Range Networks, "OpenBTS Application Suite Release 4.0 User Manual," Apr. 15, 2014. http://openbts.org/site/wp-content/uploads/2014/07/OpenBTS-4.0-Manual.pdf

[28] Magma, "Magma Introduction." https://magmacore.org/ (accessed Dec. 01, 2021).

[29] E. R. Brand a National Instruments, "USRP B200 USB Software Defined Radio (SDR)," *Ettus Research*. https://www.ettus.com/all-products/ub200-kit/ (accessed Jun. 07, 2021).

[30] "About GNU Radio · GNU Radio." Accessed: Jun. 07, 2021. [Online]. Available: https://www.gnuradio.org/about/

[31] A Ghayas, "What is the difference between Node B, eNodeB, and gNB? - Commsbrief," *COMMSBRIEF*. https://commsbrief.com/what-is-the-difference-between-node-b-enodeb-ng-enb-and-gnb/ (accessed Jun. 06, 2021).

[32] Openair-cn, *openair-epc-fed*. OpenAirInterface Software Alliance, 2021. [Online]. Available: https://github.com/OPENAIRINTERFACE/openair-epc-fed

[33] European Telecommunications Standards Institute, "Universal Mobile Telecommunications System (UMTS); LTE; 3G Security; Specification of the MILENAGE algorithm set: An example algorithm set for the 3GPP authentication and key generation functions f1, f1*, f2, f3, f4, f5 and f5*; Document 2: Algorithm specification (3GPP TS 35.206 version 16.0.0 Release 16)," Sophia Antipolis Cedex - France, ETSI TS 135 206 V16.0.0, Aug. 2020. [Online]. Available: https://www.etsi.org/deliver/etsi_ts/135200_135299/135206/16.00.00_60/ts_135206v160000p.pdf

[34] Diameter Protocol, "Diameter Protocol Explained: Usage of OP/OPc and Transport Key," *Diameter Protocol Explained*. https://diameter-protocol.blogspot.com/2013/06/usage-of-opopc-and-transport-key.html (accessed Oct. 11, 2021).

[35] S. Blacksays, "LTE Network Architecture," *Nick vs Networking*, May 13, 2019. https://nickvsnetworking.com/usim-basics/ (accessed Oct. 11, 2021).

[36] E. R. Brand a National Instruments, "OctoClock Clock Distribution Module with GPSDO," *Ettus Research*. https://www.ettus.com/all-products/octoclock-g/ (accessed Oct. 24, 2021).

[37] *OPENAIRINTERFACE/openair-epc-fed*. OpenAirInterface Software Alliance, 2021. Accessed: Oct. 17, 2021. [Online]. Available: https://github.com/OPENAIRINTERFACE/openair-epc-fed/blob/8b36c7f5cd80ff0a5e633d2a30e44beeb273907f/docs/CONFIGURE_NETWORKS.md

[38] Florian Kaltenberger, "TESTING_GNB_W_COTS_UE," *GitLab*, Jun. 2021. https://gitlab.eurecom.fr/oai/openairinterface5g/blob/develop/doc/TESTING_GNB_W_COTS_UE.md

[39] Openairinterface 5G Wireless Implementation, *Update SA documentation*. OpenAirInterface Software Alliance, 2021. [Online]. Available: https://gitlab.eurecom.fr/oai/openairinterface5g/-/blob/develop/doc/RUNMODEM.md

[40] Openairinterface 5G Wireless Implementation, *TESTING GNB W COTS UE*. Openairinterface 5G Wireless Implementation, 2021. Accessed: Oct. 16, 2021. [Online]. Available: https://gitlab.eurecom.fr/oai/openairinterface5g/-/blob/develop/doc/TESTING_GNB_W_COTS_UE.md

[41] OpenAirInterface 5G software alliance, "OpenAirInterface – 5G software alliance for democratising wireless innovation," *OpenAirInterface*. https://openairinterface.org/ (accessed Nov. 03, 2021).

[42] T. Byrd, V. Marojevic, and R. P. Jover, "CSAI: Open-Source Cellular Radio Access Network Security Analysis Instrument," *ArXiv190507617 Cs Eess*, May 2019, Accessed: Nov. 01, 2021. [Online]. Available: http://arxiv.org/abs/1905.07617

[43] Rondeau Research, "VOLK: Vector-Optimized Library of Kernels," Dec. 10, 2010. http://www.trondeau.com/blog/2010/12/11/volk-vector-optimized-library-of-kernels.html

[44] Michael Dickens, *Installing VOLK*. GNU Radio, 2021. [Online]. Available: https://github.com/gnuradio/volk/blob/178de94bace71c1abea1cb2f193c9194e8ba0454/README.md

[45] NickvsNetworking, "LTE (4G) – USIM Basics," *Nick vs Networking*, May 13, 2019. https://nickvsnetworking.com/usim-basics/

[46] Luis Pereira and Francois Taburet, "OpenAirInterface Functionalities," *GitLab*, Jun. 06, 2021. https://gitlab.eurecom.fr/oai/openairinterface5g/blob/develop/doc/RUNMODEM.md

[47] Quectel Wireless Solutions Co., Ltd., "EC25 & EC21 AT Commands Manual," Nov. 14, 2017. https://sixfab.com/wp-content/uploads/2018/09/Quectel_EC25EC21_AT_Commands_Manual_V1.2.pdf

[48] Miquel van Smoorenburg and Jukka Lahtinen, "Minicom - Linux manual page," Dec. 2013. https://man7.org/linux/man-pages/man1/minicom.1.html

[49] Michael Kerrisk, "The Linux man-pages project," *The Linux man-pages project*. https://www.kernel.org/doc/man-pages/ (accessed Nov. 25, 2021).

[50] Torbjorn Granlund, David MacKenzie, and Paul Eggert, "Linux Man Page - df," *DF*, Apr. 2020. https://linuxcommand.org/lc3_man_pages/df1.html

[51] Docker, "Docker Child Commands," *Docker Documentation*, 2021. https://docs.docker.com/engine/reference/commandline/docker/

[52] European Telecommunications Standards Institute, "LTE; Evolved Universal Terrestrial Radio Access (E-UTRA); User Equipment (UE) radio transmission and reception (3GPP TS 36.101 version 14.3.0 Release 14)6101v140300p.pdf," Sophia Antipolis Cedex - France, ETSI TS 136 101 V14.3.0, Apr. 2017. [Online]. Available: https://www.etsi.org/deliver/etsi_ts/136100_136199/136101/14.03.00_60/ts_136101v140300p.pdf

[53] European Telecommunications Standards Institute, “5G; NR; User Equipment (UE) radio transmission and reception; Part 1: Range 1 Standalone (3GPP TS 38.101-1 version 16.9.0 Release 16),” Sophia Antipolis Cedex - France, ETSI TS 138 101–1 V16.9.0, Oct. 2021. [Online]. Available: https://www.etsi.org/deliver/etsi_ts/138100_138199/13810101/16.09.00_60/ts_13810101v160900p.pdf

[54] E. Dahlman, S. Parkvall, and J. Sköld, *5G NR: the next generation wireless access technology*, Second edition. London San Diego, CA Cambridge, MA Oxford: Elsevier, Academic Press, 2021.

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California