UNIVERSIDADE ESTADUAL PAULISTA
"JÚLIO DE MESQUITA FILHO"
Instituto de Ciência e Tecnologia
de Sorocaba

JOÃO GUSTAVO RODRIGUES DE OLIVEIRA

ESTUDO SOBRE REDES DE TELEFONIA 4G UTILIZANDO RÁDIO DEFINIDO POR SOFTWARE: SOFTWARE DEFINED RADIO - SDR

Sorocaba – SP

2022

JOÃO GUSTAVO RODRIGUES DE OLIVEIRA

ESTUDO SOBRE REDES DE TELEFONIA 4G UTILIZANDO RÁDIO DEFINIDO POR SOFTWARE: SOFTWARE DEFINED RADIO - SDR

Projeto Final de Curso apresentado ao Instituto de Ciência e Tecnologia de Sorocaba, Universidade Estadual Paulista (UNESP), como parte dos requisitos para obtenção do grau de Bacharel em Engenharia de Controle e Automação.

Orientador: Prof. Dr. Everson Martins

Sorocaba – SP

2022

Oliveira, João
Estudo sobre redes de telefonia 4G utilizando rádio definido por software: Software defined radio - SDR / João Oliveira. -- Sorocaba, 2022
99 p.

Trabalho de conclusão de curso (Bacharelado - Engenharia de Controle e Automação) - Universidade Estadual Paulista (Unesp),

1. Rádio definido por software. 2. LTE. 3. Virtualização. 4. Contêineres. I. Título.

Sistema de geração automática de fichas catalográficas da Unesp. Biblioteca do Instituto de Ciência e Tecnologia, Sorocaba. Dados fornecidos pelo autor(a).

UNIVERSIDADE ESTADUAL PAULISTA
"JÚLIO DE MESQUITA FILHO"
Instituto de Ciência e Tecnologia
Câmpus de Sorocaba

ESTUDO SOBRE REDES DE TELEFONIA 4G UTILIZANDO RÁDIO DEFINIDO POR SOFTWARE: SOFTWARE DEFINED RADIO - SDR

JOÃO GUSTAVO RODRIGUES DE OLIVEIRA

ESTE TRABALHO DE GRADUAÇÃO FOI JULGADO ADEQUADO COMO PARTE DO REQUISITO PARA A OBTENÇÃO DO GRAU DE **BACHAREL EM ENGENHARIA DE CONTROLE E AUTOMAÇÃO**

Prof. Dr. Mauricio Becerra Vargas

Coordenador

BANCA EXAMINADORA:

Prof. Dr. Everson Martins
Orientador/UNESP-Campus de Sorocaba

Prof. Dr. Galdenoro Botura Junior

UNESP- Campus de Sorocaba

Prof. Dr. Eduardo Verri Liberado

UNESP – Campus de Sorocaba

Agosto de 2022

AGRADECIMENTOS

Gostaria de dedicar um agradecimento especial a minha família e minha companheira Joelle que sempre estiveram presentes me proporcionando apoio, possibilitando que eu pudesse continuar focado e empenhado em meus estudos. Agradeço também ao Instituto de ciência e tecnologia de Sorocaba (UNESP) por toda a estrutura e pelos anos de ensinamento que me foram passados. Um agradecimento especial ao FIT instituto de tecnologia, por proporcionar a infraestrutura necessária para a realização do projeto em questão, cedendo os equipamentos necessários e o auxílio para a realização dos testes. Por fim, agradeço ao professor Everson Martins pelo auxílio prestado durante todo o processo de implementação desse projeto.

OLIVEIRA, J. G. R. **Estudo Sobre Redes De Telefonia 4G Utilizando Rádio Definido por Software: Software Defined Radio – SDR.** 2022. 99 f. Trabalho de Conclusão de Curso (Graduação em Engenharia de Controle e Automação) – Instituto de Ciência e Tecnologia de Sorocaba, UNESP – Universidade Estadual Paulista, Sorocaba, 2022.

RESUMO

As tecnologias de comunicação móvel estão presentes em todo o planeta, fazendo parte do dia a dia de nossa sociedade. Ao longo dos últimos 20 anos, a maneira como nos comunicamos mudou drasticamente, isso se deve principalmente ao desenvolvimento dessas tecnologias. O desenvolvimento da comunicação móvel progrediu de maneira muito veloz nos últimos anos, proporcionando ao usuário não somente a capacidade de se comunicar por meio de ligações de voz, mas também trocar mensagens de texto e navegar na internet de maneira rápida e responsiva. Há diversas "gerações" de tecnologias móvel, sendo que cada uma delas trouxe algum tipo de avanço para o sistema de comunicação móvel. Dessa forma, o intuito desse estudo é traçar uma sequência lógica sobre a evolução das tecnologias móvel e focar principalmente na quarta geração ou 4G LTE. Tendo em vista que, mesmo com o avanço e a promessa de altas velocidades e baixa latência das tecnologias 5G NR, o 4G ainda será a tecnologia predominante por mais alguns anos, e algumas de suas características serão utilizadas em algum grau para o 5G. Para tal, será utilizado de um Software Defined Radio, equipamento esse capaz, entre outras coisas, de emular redes de comunicação móvel 4G. Também serão utilizados softwares de código aberto para emular um core 4G, um eNodeB e será utilizado um software para gravação de SIM cards, visando a criação de uma rede privada 4G operacional.

Palavras-chave: Redes móveis de comunicação. tecnologia 4G LTE. Rádio Definido por Software. Virtualização. Contêineres.

OLIVEIRA, J. G. R. **Study of 4G Technology Using Software Defined Radio – SDR** 2022. 99 p. Course Conclusion Paper (Graduation in Control and Automation Engineering) – Institute of Science and Technology of Sorocaba, UNESP – Universidade Estadual Paulista, Sorocaba, 2022

ABSTRACT

Mobile communication technologies are present all over the world, being part of the daily life of our society. Over the last 20 years, the way we communicate has changed dramatically, this is mainly due to the development of these technologies. The mobile communication technologies have progressed in a very advanced way, making possible to the users not only the ability to communicate through voice calls, but also exchange messages through SMS and the access to the internet in a fast and responsive way. We have several "generations" of mobile technologies, and each of them brought advancement to the mobile communication system. Thus, the purpose of this study is to trace a sequential logic on the evolution of mobile technologies and focus mainly on the fourth generation or 4G LTE. Considering that, even with the advancement of 5G NR and its features and technology promises, 4G will still be a predominant technology for a couple of years, and some of its features will be used to some degree on 5G. To this end, a Software Defined Radio will be used, being an equipment capable, out of other features, of emulating 4G mobile communication networks. Also, will be implemented open source softwares for a 4G core, eNodeB and will be used a software for recording sim cards, aiming for a private 4G network.

Keywords: Mobile Communication Networks. 4G Technology LTE. Software Defined Radio. Virtualization. Containers.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

1G	- Tecnologia de telefonia móvel de primeira geração
2G	- Tecnologia de telefonia móvel de segunda geração
3G	- Tecnologia de telefonia móvel de terceira geração
3GPP	- *3rd Generation Partnership Project*
4G	- Tecnologia de telefonia móvel de quarta geração
5G	- Tecnologia de telefonia móvel de quinta geração
APN	- *Access Point Name*
AS	- *Access Stratum*
CSN	- *Circuit switched network*
CP	- *Control Plane*
COTS	- *Commercial off-the-shelf*
CPRI	- *Common public radio interface*
CSFB	- *Circuit switch fall back*
DC	- *Direct current*
DFT	- *Discrete Fourier transform*
EARFCN	- *Absolute radio-frequency channel number*
EDGE	- *Enhanced Data Rates for GSM Evolution*
EMM	- Evolved packet system *Mobility Management*
eNodeB	- Nó B (estação base)
EPC	- *Evolved packet core*
EPS	- *Evolved packet system*
ESM	- Evolved packet system *Mobility Management*
E-UTRA	- *Evolved universal terrestrial radio access*
E-UTRAN	- *Evolved universal terrestrial radio access network*
FDD	- *Frequency division duplex*
FFT	- *Fast Fourier transform*
FPGA	- *Field Programmable Gate Arrays*
GPIO	- *General Purpose Input/Output*
GPRS	- *General Packet Radio Service*

GPS	- Sistema de posicionamento global
GPSDO	- GPS disciplined oscillator
HSDPA	- *High-Speed Downlink Packet Access*
HSS	- *Home Subscriber Storage*
I-CSCF	- *Interrogator Call Session Control Function*
IP	- *Internet protocol*
ISI	- *Inter Symbol Interference*
IFFT	- *Inverse Fast Fourier transform*
IMPU	- *IP Multimedia Private Identity*
IMPI	- *IP Multimedia Public Identity*
IMS	- *IP Multimedia Subsystem*
IMSI	- *International mobile subscriber identity*
JDK	- *Java Development Kit*
K	- *Secret Key*
LTE	- *Long term Evolution*
NBR	- Norma Brasileira
NodeB	- Nó B (estação base)
MAC	- *Medium access control*
MCC	- *Mobile Country Code*
MCS	- *Modulation and Coding Scheme*
MIMO	- *Multiple input multiple output*
MME	- *Mobility Management Entity*
MMS	- *Multimedia Messaging Service*
MNC	- *Mobile Network Code*
MSISDN	- *Mobile Station Integrated Services Digital Network*
NAS	- *Non Access Stratum*
NSA	- *Not Stand Alone*
OFDM	- *Orthogonal Frequency Division Multiplexing*
OFDM	- *Orthogonal Frequency Division Multiple Access*
OPC	- *Operator Key*

P-CSCF	- *Proxy Call Session Control Function*
PCRF	- *Policy and Charging Rules Function*
PDCP	- *Packet Data Convergence Protocol*
PDN	- *Packet Data Network*
PDU	- *Protocol Data Unit*
PGW	- *Packet Data Network Gateway*
PLMN	- *Public Land Mobile Network*
PSTN	- *Public Switched Telephone Network*
QAM	- *Quadrature Amplitude Modulation*
QCI	- *Quality of service Identifier*
QoS	- *Quality of service*
RAN	- *Radio access network*
RLC	- *Radio link control*
RNC	- *Radio network controller*
RTP	- *Real Time Protocol*
RRH	- *Remote radio head*
S-CSCF	- *Serving Call Session Control Function*
SC-FDMA	- *Single-carrier FDMA*
SA	- *Stand Alone*
SDR	- *Software defined radio*
SGW	- *Serving Gateway*
SIM	- *Subscriber Identity Module*
SIP	- *Session Initiation Protocol*
SISO	- *Single Input Single Output*
SMF	- *Session Management function*
SMS	- *Short Message Service*
TCP	- *Transmission Control Protocol*
TDD	- *Time division duplex*
UE	- *User equipment*
UHD	- *USRP Hardware Driver*

UMB	- *Ultra Mobile Broadband*
UMTS	- *Universal mobile telephone system*
UNESP	- Universidade Estadual Paulista
UP	- *User Plane*
UPF	- *User Plane function*
USRP	- *Universal Software Radio Peripheral*
UTRA	- *Universal terrestrial radio access*
UTRAN	- *Universal terrestrial radio access network*
VoLTE	- *Voice over LTE*
VoNR	- *Voice over New Radio*
W-CDMA	- *Wide-Band Code-Division Multiple Access*
WiMax	- *Worldwide Interoperability for Microwave Access*

Sumário

1 INTRODUÇÃO

1.1 JUSTIFICATIVA

A tecnologia de comunicação móvel de quarta geração (4G), está presente em quase todos os municípios brasileiros, alcançando cerca de 88, 63% da população brasileira. Porém, na perspectiva de cobertura total do território nacional, a cobertura é de aproximadamente 11,09%, implicando em uma vasta região sem cobertura desse serviço (TELESINTESE, 2022) Dessa forma, é seguro afirmar que o 4G ainda é uma tecnologia essencial, segundo (GSMA, 2022) o 4G é responsável por 43% do total de conexões celulares no mundo, e continuará predominante na américa latina, onde estima-se que em 2025 ainda representará 67% do total de conexões (BRECHAZERO, 2022).

O termo LTE (Long Term Evolution) foi cunhado pela 3GPP (3rd Generation Partnership Project), sendo esse um órgão de muita relevância para o desenvolvimento das telecomunicações, responsável por protocolar as definições e o funcionamento das redes de telecomunicações, desde o desenvolvimento da tecnologia de comunicação móvel de terceira geração (3G) (TELECO, 2022). Dessa forma, para um maior entendimento tanto do LTE como do das tecnologias de comunicação móvel, é importante traçar um paralelo sobre o desenvolvimento das redes de telefonia como um todo.

A tecnologia de comunicação móvel de primeira geração (1G) surgiu em meados dos anos 80, e trouxe ao usuário o acesso aos serviços de voz sem fio pela primeira vez, dando início ao mercado de telecomunicações. Baseava-se principalmente em um sistema analógico por comutação de circuitos (do inglês, circuit switched network - CSN), que basicamente consiste no estabelecimento de uma ligação entre dois “nós” através do chaveamento de circuitos nas malhas. As frequências de operação nesse sistema estavam na casa de 800 a 900 MHz, sendo a capacidade do canal limitada a 30 KHz, possuindo uma qualidade de serviço muito baixa se comparada às novas versões de tecnologia de comunicação móvel que estariam por vir. Porém devido a essas características, o 1G possuía uma infraestrutura de rede mais simples de ser implementada (ARSHAD; KASHIF; QUERSHI; 2019).

Dentro das tecnologias de comunicação móvel de segunda geração (2G) foram desenvolvidas diferentes versões, sendo o Sistema Global para Comunicações Móveis (do inglês, Global System for Mobile Communications – GSM) a primeira implementação que providenciou ao usuário o acesso tanto a voz como ao serviço de troca de mensagens curtas (do inglês, Short Message Service - SMS). A próxima versão do 2G foi o Serviço de Rádio de Pacote Geral (do inglês, General Packet Radio Services – GPRS), também conhecido como

2.5G que providenciava velocidades de acesso a dados de até 150 Kbps. Por fim, em meados dos anos 2000 surge o EDGE (Enhanced Data Rates for GSM Evolution) ou 2.75G, que incrementou a taxa de acesso a dados para cerca de 384 Kbps (YADAV, 2017).

A tecnologia de comunicação móvel de terceira geração (3G) surgiu para aprimorar os serviços de voz e acesso a dados. No 3G houve um salto na qualidade dos serviços (do inglês, Quality of Service - QoS) oferecidos, oferecendo até 2 Mbps de taxa de acesso à Internet, o que se demonstrou bem superior aos 384 Kbps oferecidos pelo 2.75G EDGE. Foram introduzidas variações para o 3G como o 3.5G (HSDPA – High Speed Downlink Packet Access) que oferece velocidades de até 14,4 Mbps para o downlink e 5,8 Mbps para o uplink, valores consideravelmente maiores do que os oferecidos no 3G padrão (YADAV, 2017). Por fim, o 3.75G (HSUPA – High Speed Uplink Packet Access) incrementou a velocidade do uplink e diminuiu o atraso entre uplink e downlink (ARSHAD; KASHIF; QUERSHI; 2019).

O 4G surgiu com a ideia de oferecer alta taxa de dados, com velocidades de até 100 Mbps, oferecendo também serviços como MMS (Multimedia Messaging Service). Parte da infraestrutura de rede do 4G é baseada no protocolo TCP/IP (*Transmission Control Protocol)*, simplificando a implementação se comparado ao 3G e podendo oferecer, ao trabalhar em conjunto com outras funcionalidades, a alta taxa de transmissão de dados (ALNAAS et al., 2018). Atualmente com a internet desempenhando um papel crucial, o conceito oferecido pelo 4G de trafegar em alta velocidade, é de extrema importância.

Neste contexto, a utilização do 4G para redes privadas vem se demonstrando de grande relevância, sendo que segundo (TELESINTESE, 2022) “existem atualmente no planeta cerca de mil redes privadas LTE funcionando em diferentes empresas, com uma grande variedade de usos”. Uma rede privada LTE consiste basicamente em uma rede 4G customizada para o uso de um cliente específico, geralmente associado a uma área restrita como uma empresa ou setor rural (TOPIN, 2022). O crescimento na utilização dessa tecnologia ainda continuará onde em 2026 estima-se que o número de redes privadas baseadas no LTE ou 5G será de aproximadamente 13.500 (MOTOROLA, 2022). Dessa forma, o estudo e implementação de redes 4G é relevante para o cenário brasileiro e mundial, sendo que os testes que serão apresentados no trabalho em questão, irão demonstrar a complexidade e as funcionalidades principais de uma rede privada 4G.

1.2 OBJETIVOS

O principal objetivo do trabalho em questão é emular uma rede privada 4G utilizando um rádio definido por software (SDR), e nessa rede emulada conectar celulares comerciais para

realizar testes de navegação na internet, velocidade de navegação e ligações de voz. Ao demonstrar a implementação e funcionamento da rede, este projeto visa avaliar a complexidade e a relevância de uma rede LTE privada. Outro ponto que será levantado é a utilização das técnicas de virtualização e conteinerização no campo das telecomunicações, demonstrando a instalação de um ambiente com esses conceitos para a validação dessas tecnologias.

2 REVISÃO DE LITERATURA

Esse trabalho tem como base a emulação de uma rede LTE privada utilizando um rádio definido por software (SDR), onde para os componentes de uma rede LTE são utilizados softwares de código aberto. Levando esse aspecto em consideração, Tavares et al. (2019) demonstra a emulação de uma rede LTE usando um SDR do modelo ettus USRP B210, realizando análises de tráfego (ETTUS, 2022). No trabalho de Tavares et al. (2019), foi utilizado o software Open Air Interface (OAI), que se trata de uma solução completa para a arquitetura LTE, possuindo os componentes EPC (Evolved Packet Core) e eNodeB (Evolved Node B) (OPEN AIR INTERFACE, 2022). Os testes se focaram justamente na análise de tráfego da comunicação entre os componentes EPC e eNodeB (TAVARES et al., 2019). No artigo de Curpen et al. (2019) é apresentado as funcionalidades de um SDR para a criação de uma rede privada LTE, utilizando para a infraestrutura da rede LTE o software Open-LTE. São realizados testes focados na análise do procedimento de conexão de um celular comercial a rede, sem realização de testes de desempenho, pois o foco do artigo era justamente demonstrar esse procedimento de conexão em uma rede LTE emulada por um SDR (CURPEN et al. ,2019).

No artigo de Tran et al. (2017) tem-se a descrição da relevância da tecnologia Cloud Radio Access Network (C-RAN), focando na criação de uma arquitetura LTE em ambientes virtualizados. O ambiente de testes implementado consiste no software OAI, sendo utilizado para a arquitetura da rede LTE (EPC e eNodeB), assim como para emular um UE, onde foram utilizados dois SDRs B210 para emular a rede e o UE. No trabalho de Nóvoa et al. (2020) é descrita a construção de uma rede LTE utilizando um SDR B210, com a utilização do software OAI para os componentes da rede LTE, assim como do software FlexRAN para realizar a funcionalidade de slicing da RAN (Radio access network). O slicing é uma ferramenta introduzida no 5G para melhor alocar os recursos da rede para cada tipo de usuário (ERICSSON, 2022). No artigo de Nóvoa et al. (2020), essa funcionalidade foi utilizada para limitar os recursos da RAN para um usuário em específico, no caso um celular comercial (Samsung Galaxy S4), demonstrando as funcionalidades do software FlexRAN e o desempenho da rede em diferentes tipos de slicing aplicados.

Uma das diferenças entre os artigos citados anteriormente e este projeto final de curso, está no software utilizado para a eNodeB, que para este trabalho se trata do (SRSLTE, 2022). Dessa forma, no trabalho de (MAQSOOD, 2019) é utilizado o srsLTE como solução completa da rede LTE (eNodeB e EPC), assim como para a emulação do UE, sendo utilizado os SDRs USRP X300 e LimeSDR para a emulação da rede LTE e do UE. São criados diversos modelos

de testes, utilizando outros equipamentos para emular componentes da rede (CMW500) e testar as suas funcionalidades, sendo que para o propósito deste projeto final de curso, somente os testes utilizando os SDRs serão abordados. No trabalho de (MAQSOOD, 2019) são realizados testes de desempenho e tráfego no downlink, sendo estes testes comparados entre os SDRs utilizados e os valores esperados teoricamente.

3 REVISÃO CONCEITUAL

3.1 TECNOLOGIAS DE COMUNICAÇÃO MÓVEL DE QUARTA GERAÇÃO 4G (LTE)

O desenvolvimento do 4G foi focado na entrega de altas taxas de dados através da utilização de novas tecnologias, tanto para a infraestrutura da rede, como na maneira como o sinal era gerado e transmitido. A associação entre 4G e LTE é automática, sendo os dois termos associados ao mesmo tipo de tecnologia. Entretanto, no início do desenvolvimento do 4G surgiram diferentes métodos de padronização para essa tecnologia, sendo um deles o LTE. O LTE acabou por se tornar o método mais utilizado para a implementação de redes 4G, devido ao desempenho, facilidade e interoperabilidade oferecidos, se comparado às outras formas de padronização (UMB - Ultra Mobile Broadband, WiMax - Worldwide Interoperability for Microwave Access) (3GPP, 2022). Segundo (MORAY, 2013) "o LTE foi um projeto desenvolvido pela 3GPP, o objetivo dele era determinar o long term Evolution (LTE) da universal mobile telephone system (UMTS). O UMTS também foi um projeto da 3GPP que estudou diversas tecnologias de multiplexação e acesso, antes de definir a utilização do Wide-Band Code-Division Multiple Access (W-CDMA) para a utilização na radio access network (RAN). Atualmente os termos W-CDMA e o UMTS são inseparáveis, porém esse não era o caso antes da tecnologia ser escolhida." A RAN do UMTS possui dois principais componentes, o universal terrestrial radio access (UTRA) que em conjunto com o user equipamento (UE) representa a interface de ar dessa arquitetura. O segundo componente da RAN no UMTS é o universal terrestrial radio access network (UTRAN) que consiste na estação base ou NodeB e do radio network controller (RNC) (MORAY, 2013).

Como o LTE foi desenvolvido para suceder essa tecnologia, sua infraestrutura possui uma nomenclatura similar ao empregado ao UMTS, como o eNodeB (evolved Node B), E-UTRA (evolved universal terrestrial radio access) e E-UTRAN (evolved universal terrestrial radio access network), que desempenham papéis similares aos desempenhados pelos seus pares no UMTS (MORAY, 2013).Uma das principais mudanças, no entanto, está na parte do core do sistema, com o desenvolvimento de um core voltado a comunicação via IP, deixando de lado o conceito de serviços com arquitetura CSN, sendo necessária toda uma reestruturação do funcionamento desses componentes. Essa reestruturação foi uma das principais responsáveis pelo ganho de desempenho alcançado pelas redes LTE do ponto de vista de velocidade para navegação na internet (3GPP, 2022). A release 8 da 3GPP, lançada oficialmente em 2008, é uma das bases para a tecnologia LTE, onde todas as especificações para as redes do protocolo estão descritas (3GPP, 2022). O desenvolvimento das redes LTE continuaram por pelo menos

mais 8 anos, até o release 14 da 3GPP, onde as funcionalidades desta tecnologia são demonstradas no seu desempenho máximo, e foi dado início à introdução do 5G (tecnologia de comunicação móvel de quinta geração) (ELETRONIC-NOTES, 2022).

3.1.1 INTERFACE DE AR PARA O LTE: FDD e TDD

Para o LTE tem-se as seguintes técnicas de duplexação: o time division duplex (TDD) e o frequency division duplex (FDD) (AVIAT, 2019). Segundo (AVIAT, 2019) "é essencial que a transmissão do sinal de rádio ocorra de maneira simultânea para que os dados possam fluir de maneira ininterrupta tanto no downlink como para o uplink" para tal, utiliza-se dessas técnicas de duplexação.

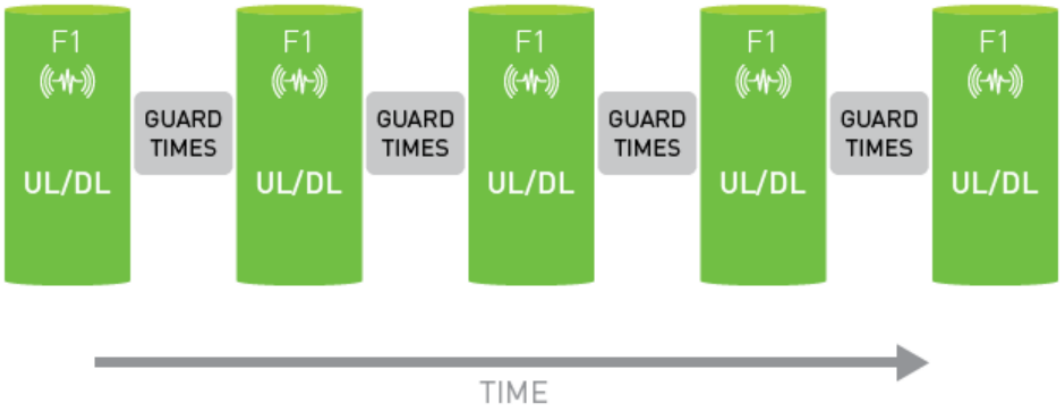
O FDD transmite em duas frequências diferentes, utilizando uma banda de guarda para separar a parte do downlink e uplink, sendo por esse motivo também uma técnica mais cara de ser utilizada, já que, será necessário a compra de duas frequências para a transmissão do sinal, e pelo fato do mercado ter utilizado mais dessa técnica, resultando em um número limitado de frequências a serem compradas (CARVALHO, 2013). Porém, como o FDD é amplamente utilizado por uma questão de compatibilidade com redes anteriores, os equipamentos para ele tendem a ser mais baratos, podendo amortecer um pouco os gastos da necessidade de compra de outra frequência (CARVALHO, 2013).

No TDD ambos downlink e uplink são transmitidos na mesma frequência separados pelo tempo, sendo necessário o emprego de bandas de guarda para separar cada par de downlink e uplink transmitidos, podendo resultar em uma perda de eficiência na latência (AVIAT, 2019). A Figura 1 apresenta um exemplo do espectro de transmissão em FDD e a Figura 2 do TDD:

Figura 1 - Transmissão de pacotes utilizando o FDD.

Fonte: AVIAT, 2019.

Figura 2 - Transmissão de pacotes utilizando o TDD.



Fonte: AVIAT, 2019.

Na tabela 1 está representada as especificações para as frequências a serem trabalhadas com essas técnicas de duplexação (MORAY, 2013):

Tabela 1 - Frequências utilizadas para o LTE.

Band number	Uplink		Downlink		Bandwidth	Duplex spacing	Gap	Duplex mode
	Low	High	Low	High				
1	1920	1980	2110	2170	60	190	130	FDD
2	1850	1910	1930	1990	60	80	20	FDD
3	1710	1785	1805	1880	75	95	20	FDD
4	1710	1755	2110	2155	45	400	355	FDD
5	824	849	869	894	25	45	20	FDD
6	830	840	875	885	10	35	35	FDD
7	2500	2570	2620	2690	70	120	50	FDD
8	880	915	925	960	35	45	10	FDD
9	1749.9	1784.9	1844.9	1879.9	35	95	60	FDD
10	1710	1770	2110	2170	60	400	340	FDD
11	1427.9	1452.9	1475.9	1500.9	20	48	23	FDD
12	698	716	728	746	18	30	12	FDD
13[1]	777	787	746	756	10	−31	21	FDD
14[1]	788	798	758	768	10	−30	20	FDD
15[2]	1900	1920	2600	2620	20	700	680	FDD
16[2]	2010	2025	2585	2600	15	575	560	FDD
17	704	716	734	746	12	30	18	FDD
18	815	830	860	875	15	45	30	FDD
19	830	845	875	890	15	45	30	FDD
20	832	862	791	821	30	−41	11	FDD
21	1447.9	1462.9	1495.9	1510.9	15	48	33	FDD
22	3410	3490	3510	3590	80	100	20	FDD
23	2000	2020	2180	2200	20	180	160	FDD
24	1626.5	1660.5	1525	1559	34	−101.5	67.5	FDD
25	1850	1915	1930	1995	65	80	15	FDD
26	814	849	859	894	35	45	10	FDD
27	807	824	852	869	17	45	28	FDD
28	703	748	758	803	45	55	10	FDD
33	1900	1920	1900	1920	20	0	0	TDD
34	2010	2025	2010	2025	15	0	0	TDD
35	1850	1910	1850	1910	60	0	0	TDD
36	1930	1990	1930	1990	60	0	0	TDD
37	1910	1930	1910	1930	20	0	0	TDD
38	2570	2620	2570	2620	50	0	0	TDD
39	1880	1920	1880	1920	40	0	0	TDD
40	2300	2400	2300	2400	100	0	0	TDD
41	2496	2690	2496	2690	194	0	0	TDD
42	3400	3600	3400	3600	200	0	0	TDD
43	3600	3800	3600	3800	200	0	0	TDD
44	703	803	703	803	100	0	0	TDD

Fonte: MORAY, 2013 – Modificado.

3.1.2 INTERFACE DE AR PARA O LTE: LARGURA DE BANDA

A largura de banda pode ser interpretada como um conjunto de frequências transmitidas em um canal específico, variando o tamanho que ocupa no espectro, dependendo da quantidade de frequências transmitidas. Nesse sentido o LTE trabalha com larguras de banda específicas, e devido a utilização da técnica de multiplexação OFDM (Orthogonal Frequency Division Multiplexing), tem-se a possibilidade de utilização de diferentes larguras de banda sem alterar as propriedades da camada física de transmissão (MORAY, 2013). Com essa característica de trabalhar com múltiplas bandas, há uma maior flexibilidade de implantação da solução, o LTE trabalha com larguras de banda de 1.4 MHz, 3 MHz, 5 MHz, 10 MHz, 15 MHz, e 20 MHz (MORAY, 2013). A Tabela 2 ilustra a combinação de bandas e largura de bandas possíveis para o LTE:

Tabela 2 - Larguras de banda disponíveis para o LTE.

E-UTRA band	Channel bandwidth					
	1.4 MHz	3 MHz	5 MHz	10 MHz	15 MHz	20 MHz
1			Yes	Yes	Yes	Yes
2	Yes	Yes	Yes	Yes	Yes	Yes
3	Yes	Yes	Yes	Yes	Yes	Yes
4	Yes	Yes	Yes	Yes	Yes	Yes
5	Yes	Yes	Yes	Yes		
6			Yes	Yes		
7			Yes	Yes	Yes	Yes
8	Yes	Yes	Yes	Yes		
9			Yes	Yes	Yes	Yes
10			Yes	Yes	Yes	Yes
11			Yes	Yes		
12	Yes	Yes	Yes	Yes		
13			Yes	Yes		
14			Yes	Yes		
17			Yes	Yes		
18			Yes	Yes	Yes	
19			Yes	Yes	Yes	
20			Yes	Yes	Yes	Yes
21			Yes	Yes	Yes	
22			Yes	Yes	Yes	Yes
23	Yes		Yes	Yes		
24			Yes	Yes		
25	Yes	Yes	Yes	Yes	Yes	Yes
26	Yes	Yes	Yes	Yes	Yes	
27	Yes	Yes	Yes	Yes		
28		Yes	Yes	Yes	Yes	Yes
33			Yes	Yes	Yes	Yes
34			Yes	Yes	Yes	
35	Yes	Yes	Yes	Yes	Yes	Yes
36	Yes	Yes	Yes	Yes	Yes	Yes
37			Yes	Yes	Yes	Yes
38			Yes	Yes		
39			Yes	Yes	Yes	Yes
40				Yes	Yes	Yes
41			Yes	Yes	Yes	Yes
42			Yes	Yes	Yes	Yes
43			Yes	Yes	Yes	Yes
44			Yes	Yes	Yes	Yes

Fonte: MORAY, 2013 – Modificado.

Nem todas as combinações demonstradas na tabela 2 podem ser utilizadas em cenários reais, devido a limitação de desempenho (MORAY, 2013). A tabela 3 tem-se a associação entre subportadoras para o uplink e downlink, resource blocks e largura de banda:

Tabela 3 - Associação de recursos por largura de banda LTE

Bandwidth	Resource Blocks	Subcarriers (downlink)	Subcarriers (uplink)
1.4 MHz	6	73	72
3 MHz	15	181	180
5 MHz	25	301	300
10 MHz	50	601	600
15 MHz	75	901	900
20 MHz	100	1201	1200

Fonte: KEYSIGHT, 2021 – Modificado.

Os resource blocks são (KEYSIGHT, 2021) "a menor unidade que pode ser alocada para o usuário. O resource block tem 180 KHz de largura na frequência e ocupa 1 slot no tempo. No eixo da frequência, os resource blocks tem 12 x 15 kHz subportadoras ou 24 x 7.5 kHz subportadoras. O número de subportadoras usadas por resource block para a maioria dos canais é 12."

3.1.3 TÉCNICAS DE MULTIPLEXAÇÃO E ACESSO: OFDMA E SC-FDMA

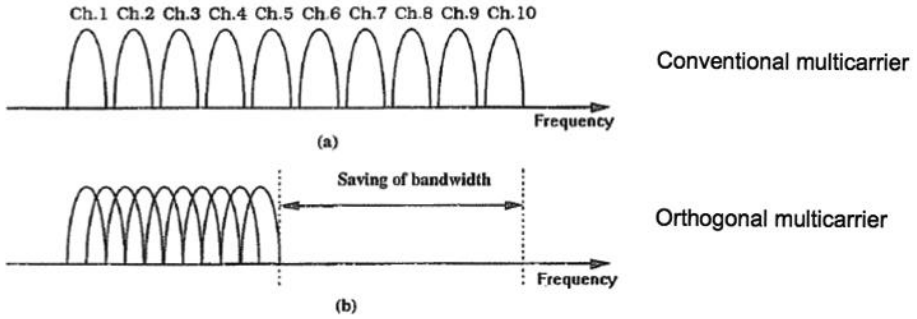
Para o desenvolvimento das redes LTE, um dos principais aspectos para os ganhos de desempenho foi o avanço nas técnicas de multiplexação, sendo uma dessas o OFDM. O OFDM (MORAY, 2013) "é um esquema de multiplexação digital baseado em um grande número de subportadoras, transmitidas de maneira pouco espaçadas para carregar dados. Cada subportadora é modulada em um formato de taxa de símbolos baixo, com o formato de modulação de amplitude em quadratura (do inglês, Quadrature Amplitude Modulation - QAM). A combinação dessas inúmeras subportadoras providência a transmissão de dados numa taxa similar aos métodos convencionais de transmissão. O OFDM é amplamente utilizado em aplicações como para os sinais de televisão e áudio digital e na transmissão de redes sem fio para acesso à internet."

O OFDM não é uma técnica que surgiu a pouco tempo, sendo que o primeiro desenvolvimento data de meados de 1957, quando o projeto kineplex do governo dos Estados Unidos da América propôs pela primeira vez o sistema para a transmissão com múltiplas subportadoras (MORAY, 2013). Segundo (MORAY, 2013) "o OFDM foi considerado como um candidato a tecnologia de multiplexação para o UMTS, porém foi decidido por utilizar o

W-CDMA. Isso se deve ao fato de na época, o consumo de recursos de hardware para a implementação dessa técnica ser muito elevado se comparado ao W-CDMA". Com o avanço das capacidades de hardware a possibilidade utilização do OFDMA (Orthogonal Frequency Division Multiple Access) foi se tornando possível (NAVITA; AMANDEEP, 2016)

O OFDMA é uma técnica de multiplexação e acesso baseado no OFDM, que visa a alta flexibilidade, facilidade de equalização e utilização optimizada do espectro de sinal transmitido (HUA et al., 2013). Essas características são alcançadas pois nessa técnica todo o espectro da banda é dividido em múltiplos subportadoras de maneira ortogonal, permitindo um melhor aproveitamento da banda (ANANDTECH, 2011). Na Figura 3 está representado um exemplo de como o método de ortogonalização oferecido pelo OFDMA proporcionou um aumento de eficiência na utilização da largura de banda:

Figura 3 - Utilização da banda no OFDMA.



Fonte: ANANDTECH, 2011.

Como pode-se observar pela Figura 3, em um modelo de multiplexação convencional (FDM – Frequency Division Multiplex), há um espaçamento entre os subportadoras, também conhecido como banda de guarda, para garantir que um sinal não se sobreponha ao outro. Já no OFDMA essas subportadoras são agrupadas via ortogonalização, o que resulta em um melhor aproveitamento do espectro de banda (ANANDTECH, 2011). Para realizar essa funcionalidade, o sinal a ser transmitido é dividido em múltiplos subportadoras, a fim de diminuir o impacto do atraso na transmissão do sinal. Para cada símbolo a ser transmitido, após a modulação, é realizada uma transformada inversa de Fourier (do inglês, Inverse Fast Fourier transform – IFFT) visando adequá-lo ao domínio do tempo e para o agrupamento via ortogonalização, dessa forma eliminando quase toda a banda de guarda para a transmissão do sinal. Como o agrupamento desses sinais pode gerar uma interferência entre símbolos (do inglês, Inter Symbol Interference - ISI) é utilizado um prefixo cíclico (do inglês, cyclic prefix) a fim de evitar o ISI (HUA et al., 2013). Após esse processo os símbolos são associados ao canal. Para o lado da recepção do sinal, é realizado o procedimento inverso ao descrito anteriormente, com a retirada

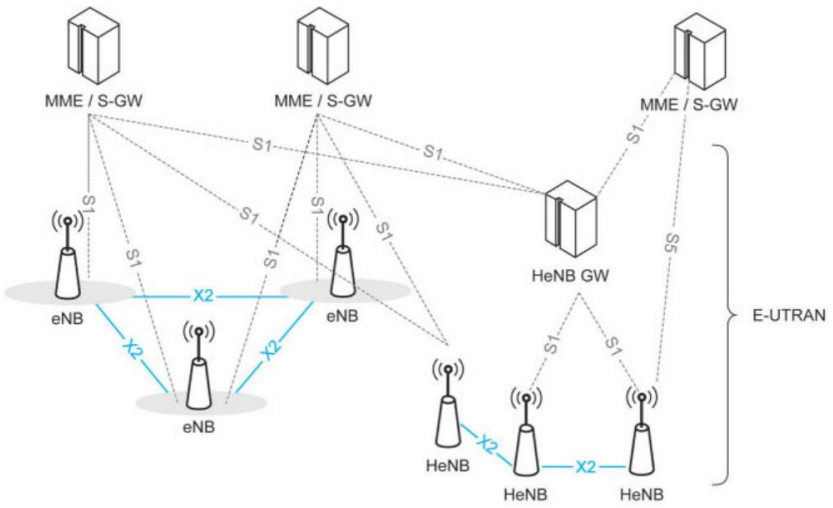
do prefixo cíclico e posteriormente a realização de uma transformada de Fourier (do inglês, Fast Fourier transform – FFT) para transpor o sinal novamente ao domínio da frequência (NAVITA; AMANDEEP, 2016). O OFDMA é utilizado somente para o downlink, e devido a presença de um alto pico de potência média para o sinal transmitido, tornando a utilização dessa ferramenta por parte do UE pouco eficiente, o que resultou na utilização do SC-FDMA (Single-carrier FDMA) para o uplink (REIS, 2022).

O SC-FDMA é semelhante ao OFDMA, contendo a diferença de ser mais eficiente do ponto de vista de consumo de potência, e pelo fato do mesmo não ser uma técnica para a transmissão de múltiplas portadoras. Nesse método é utilizado de uma transformada discreta de Fourier (do inglês, Discrete Fourier transform – DFT) antes de ser realizada uma IFFT no símbolo, dessa forma cada subportadora conterá parte de cada símbolo após a IFFT (REIS, 2022).

3.1.4 ARQUITETURA DO LTE: E-UTRAN

A topologia total de um sistema LTE é chamado de EPS (Evolved packet system), sendo que esse sistema pode ser subdividido em E-UTRAN e EPC (Evolved packet core). O E-UTRAN executa as funções de gerenciamento do rádio e consiste basicamente em uma eNodeB e do UE, sendo que a eNodeB providencia as funcionalidades de plano de controle e usuário ao usuário (3GPP, 2012). A Figura 4 se trata da arquitetura do E-UTRAN, demonstrando as interfaces de comunicação entre os seus componentes e a comunicação com o EPC (3GPP, 2012):

Figura 4 - Arquitetura do E-UTRAN

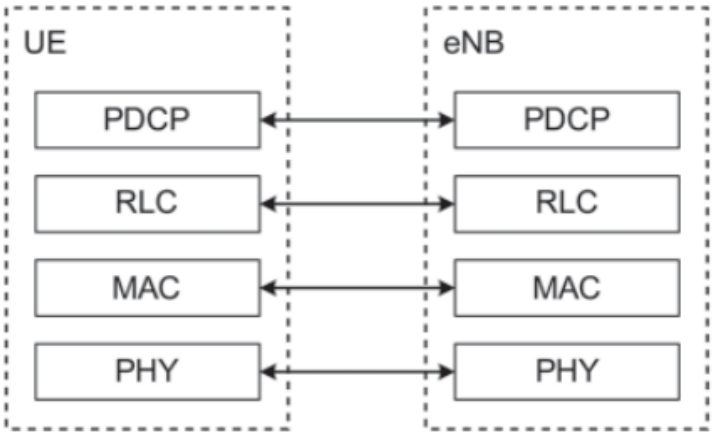


Fonte: MORAY, 2013.

O principal componente do E-UTRAN é o eNodeB. O eNodeB é responsável por diversas funcionalidades, entre as principais estão (TELECO, 2022): o controle de acesso do usuário aos serviços da rede, atribuição do uplink e downlink ao usuário, estabelecimento da interface S1U que em resumo estabelece o uplink entre usuário e a rede, estabelecimento do plano de controle (do inglês, control plane – CP) entre usuário e EPC (interface S1-MME), compressão de dados via protocolo de convergência de dados (do inglês, Packet Data Convergence Protocol -PDCP) entre outras funcionalidades.

É importante falar também sobre o (MORAY, 2013) “access stratum (AS) que contém as diretrizes responsáveis pelo acesso a interface de rádio e o controle da atividade de conexão entre UE e rádio. O AS é dividido em plano de controle e usuário. O plano de usuário é majoritariamente encarregado de transportar dados, como por exemplo pacotes IP pelo AS. No plano de controle é realizado o gerenciamento das conexões entre UE e a rede.” O diagrama do plano de usuário para o AS está representado na Figura 5:

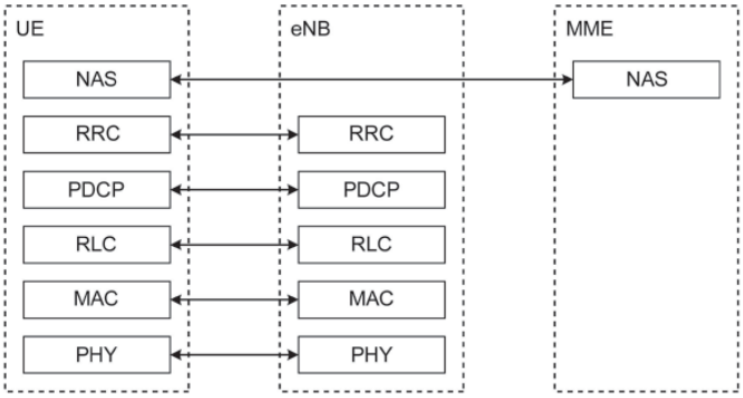
Figura 5 - Pilha de protocolo para o plano de usuário AS



Fonte: MORAY, 2013.

O plano de controle do AS para o PDCP, RLC (Radio link control) e MAC (Medium access control) se comporta da mesma maneira que o plano de usuário, porém as informações transportadas não são pacotes IP para dados, mas sim mensagens de controle geradas no RRC (Radio Resource Control), que podem conter mensagens de non access stratum (NAS) (MORAY, 2013). A pilha de protocolo para o plano de controle está ilustrada na Figura 6:

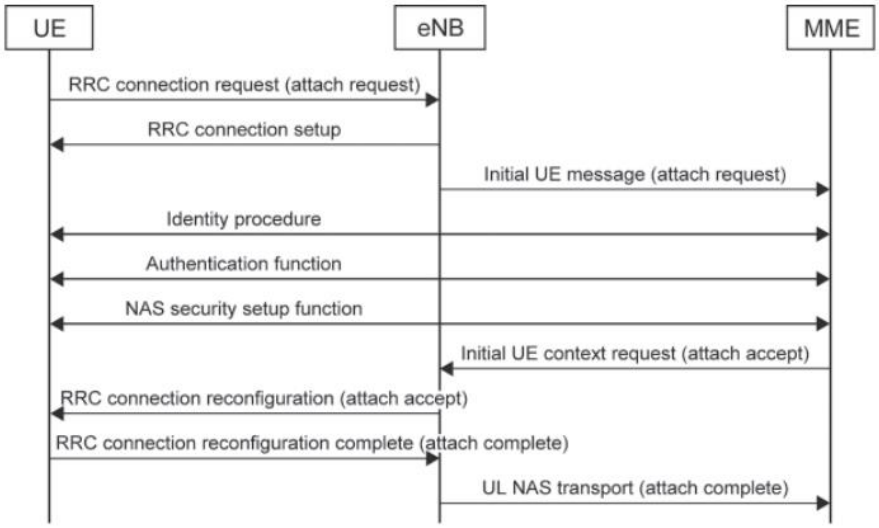
Figura 6 - Pilha de protocolo para o plano de controle AS



Fonte: MORAY, 2013.

Na Figura 7 está ilustrado o procedimento simplificado da conexão de um UE a rede LTE, levando em consideração alguns dos aspectos tratados na pilha de protocolo do access stratum:

Figura 7 - Attach de um UE na rede LTE



Fonte: MORAY, 2013.

Outro ponto interessante a ser comentado é sobre as interfaces de comunicação entre eNodeB e rádio. A eNodeB consiste basicamente no conjunto entre a unidade de banda base (do inglês, baseband unit – BBU) e o rádio (RRH), sendo que esses dois elementos estão conectados via interface FrontHaul. O protocolo de comunicação da interface FrontHaul é o CPRI (Common Public Radio interface) (TECHTARGET, 2021). A Figura 8 apresenta um diagrama da comunicação BBU e rádio via FrontHaul, e a comunicação entre eNodeB e EPC, que também pode ser chamada de BackHaul:

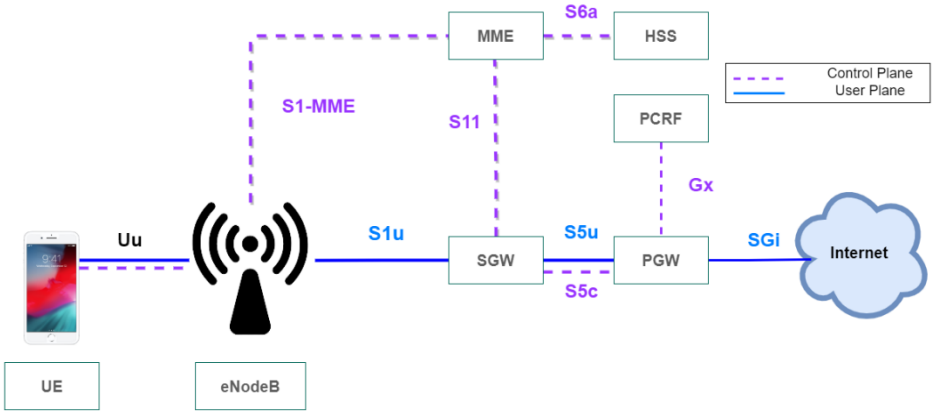
Figura 8 - Interface de comunicação FrontHaul e BackHaul

Fonte: Autoria Própria

3.1.5 ARQUITETURA DO LTE: EPC

O Evolved Packet Core (EPC) é um dos principais componentes de uma rede LTE, sendo responsável por muitas das funcionalidades de plano de controle e roteamento para o acesso à Internet na rede (TELECO, 2022). Na Figura 9 apresenta-se um esquemático dos componentes presentes em um EPC, a interconexão com o UE e eNodeB e as interfaces de comunicação entre eles:

Figura 9 - Arquitetura do EPC



Fonte: Autoria Própria.

O MME (Mobility Management Entity) é o componente mais importante do EPC, sendo responsável por gerenciar toda a parte de plano de controle, executando as funcionalidades de controle de mobilidade (EMM) e controle de seção (ESM). Outras funcionalidades importantes do MME são (TELECO, 2022): a definição do estabelecimento de túneis de comunicação, sinalização NAS e de protocolos de segurança e autenticação, definição do SGW (Serving Gateway) e PGW (Packet Gateway) que serão utilizados para o acesso do usuário a rede, entre outras funcionalidades.

Pela interface S1-MME é realizado o gerenciamento do plano de controle entre o EPC e E-UTRAN, ocorrendo as primeiras comunicações e o início dos protocolos de segurança e autenticação. Com a interface S6a entre MME e HSS (Home Subscriber Storage), são trocadas informações sobre as chaves de segurança do usuário, para que o MME possa determinar se os dados fornecidos pelo eNodeB sobre o usuário são válidos para o acesso à rede. A interface S11 atua no plano de controle entre MME e SGW, para determinar qual SGW e PGW será utilizado

para o estabelecimento da comunicação com o E-UTRAN. O HSS atua como o banco de dados da rede LTE, armazenando os parâmetros de acesso do usuário, e através da comunicação com o MME, determina se o usuário pode ou não acessar a rede. O SGW atua como ponto de entrada para o plano de usuário, executando funções de roteamento para o acesso do usuário a rede de dados através da interface S1U. Atua também, na seleção do PGW a ser utilizado pela interface S5, sendo dividida entre plano de controle e usuário, tanto para a comunicação e determinação de funcionalidades, como para o roteamento e acesso a dados (TELECO, 2022). O PGW atua na comunicação com a internet, realizando o roteamento do usuário para o acesso a dados via interface SGi, podendo atuar também para o acesso a outros serviços (MORAY, 2013). Por fim, o PCRF (Policy and Charging Rules Function) atua para definir regras de tráfego e tarifação, se comunicando com o PGW via interface Gx justamente para a determinação dessas funcionalidades. Na tabela 3 está apresentada a relação entre cada uma dessas interfaces, o protocolo de comunicação e suas funcionalidades (TELECO, 2022):

Tabela 4 - Interfaces e protocolos de comunicação EPC

INTERFACE	**PROTOCOLO**	**DESCRIÇÃO**
LTE-Uu	E-UTRA	É a interface para o CP e UP entre UE e eNodeB. A conexão de sinalização sobre a LTE-Uu é a *RRC* representada por *Signaling Radio Bearers* (SRBs).
X2	X2-AP (*CP*) GTP-U (*UP*)	A X2 é uma interface para o CP e UP entre dois eNBs. O protocolo X2-AP é utilizado no CP e um túnel GTP-U é utilizado para o UP.
S1-U	GTP-U (UP)	A Interface S1-U é utilizada para o UP entre a rede e-UTRAN (eNB) e o S-GW. Essa interface utiliza um protocolo GTP-U.
S1-MME	S1-AP (*CP*)	A Interface S1-MME é utilizada para o *CP* entre a rede e-UTRAN (eNB) e o MME. Essa interface utiliza um protocolo GTP-C.
S11	GTP-C (CP)	Interface para o *CP* entre um MME e um S-GW. Ele fornece um túnel GTP-C por usuário.
S5	GTP-C (*Control Plane*) GTP-U (*User Plane*)	É uma interface definida entre um S-GW e um P-GW para o *CP* e *UP*. A interface S5 proporciona dois túneis, um GTP-C para o *CP* e outro, GTP-U para UP.

S6a	Diameter	Interface para o CP entre um HSS e MME. Faz as trocas de assinaturas de usuário e informações de autenticação.
Gx	Diameter	Interface entre PCRF e o P-GW (*CP)*
SGi	IP	Interface entre P-GW e o PDN (Packet data Network).

Fonte: TELECO, 2022 – Modificado.

3.1.5 ARQUITETURA DO LTE: VOLTE E CORE IMS

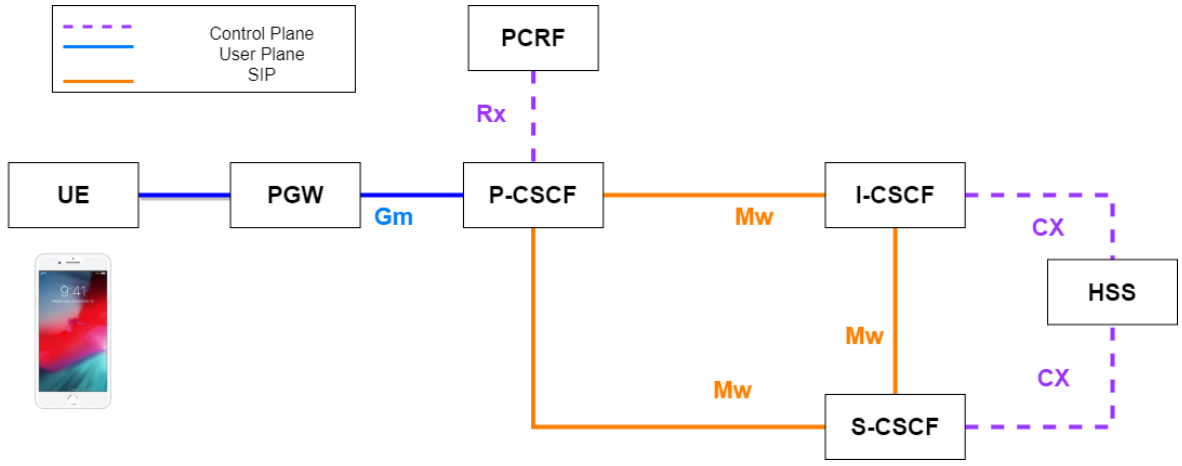
O LTE foi criado buscando entregar uma maior taxa de dados, tendo um enfoque muito claro no acesso à internet. Devido a esse aspecto, os serviços de voz foram em um primeiro momento deixado um pouco de lado, sendo utilizado o CSFB (circuit switch fall back) para a realização de ligações, que consiste basicamente em utilizar da infraestrutura do 2G ou 3G para a realização das ligações. Essa limitação se deve ao fato de o LTE ser baseado em serviços com comunicação via IP, não podendo em um primeiro momento utilizar dessa infraestrutura para a realização das ligações (TELECOMTUTORIAL, 2018).

Para contornar esse problema, foram feitas iniciativas a fim de desenvolver uma tecnologia para os serviços de voz voltada a infraestrutura do LTE, com a comunicação entre esses serviços sendo realizada via IP, o VoLTE (Voice over LTE) (TELECOMTUTORIAL, 2018). Para criar essa funcionalidade (3GLTEINFO, 2022) “em 2009 12 das maiores companhias do ramo de telecomunicações, colaboraram para propor uma solução para suportar o serviço de voz sobre redes LTE. A solução seria baseada na arquitetura IMS (IP Multimedia Subsystem) e a iniciativa foi chamada de One Voice.” Após a iniciativa one voice, em meados de 2010 foi inicializada pela GSMA o processo de padronização desses serviços, naquilo que ficou conhecido como IR 92, que define os padrões para a utilização do core IMS no VoLTE (GSMA, 2020).

O core IMS se tornou a base para a implementação do VoLTE, sendo o serviço IMS originalmente desenvolvido em meados de 1999 por uma empresa chamada 3G.IP. Essa iniciativa foi incorporada posteriormente pela 3GPP, como tentativa de utilização de serviços de mídia para o UMTS. O IMS é baseado em arquitetura de protocolo IP, providenciando serviços de voz assim como de transporte de dados (QADEER et al., 2009).

Para simplificar o entendimento do core IMS, pode-se dizer que um dos seus principais componentes é o servidor SIP (Session Initiation Protocol). O servidor SIP, para o core IMS, consiste basicamente em três componentes o P-CSCF (Proxy Call Session Control Function), I-CSCF (Interrogator Call Session Control Function) e S-CSCF (Serving Call Session Control Function). Esses componentes irão registrar os usuários, sendo parte do plano de controle e sinalização. Serão responsáveis também, por configurar as seções de mídia ou de carga para a realização das ligações. O servidor SIP é responsável pela iniciação de sessão, pela interrupção da seção e controle de seção. A Figura 10 apresenta uma arquitetura simplificada da comunicação entre EPC e core IMS (TELECOMTUTORIAL, 2020):

Figura 10 - Arquitetura simplificada do core IMS



Fonte: Autoria Própria.

O P-CSCF funciona como porta de entrada para o core IMS, sendo realizada uma comunicação de ponta a ponta entre esse componente e o UE. Algumas de suas funções são (QADEER et al., 2009):

1. Autorizar os recursos da portadora para o nível de QoS apropriado
2. Monitoramento
3. Realizar o planejamento para ligações de emergência
4. Compressão de dados
5. Identificar o I-CSCF adequado

O I-CSCF é responsável por requisitar os dados das chaves de segurança do UE ao HSS pela interface Cx, e repassa a requisição para o S-CSCF. Algumas de suas funções são (QADEER et al., 2009):

1. Selecionar um S-CSCF apropriado para a seção do UE
2. Efetuar parte do registro SIP
3. Gerar dados sobre os recursos alocados pelo UE

4. Atua como um gateway de interfuncionamento de ocultação de topologia

O S-CSCF é responsável por realizar o controle de sessão do usuário dentro da rede SIP, sendo considerado o nó central do plano de sinalização. Algumas de suas funções são (QADEER et al., 2009):

1. Providencia os serviços de roteamento
2. Gerência e inspeciona cada sinalização, sendo o nó central do plano de sinalização

De maneira simplificada, as principais vantagens de utilizar o VoLTE são a possibilidade de realização de ligações em alta definição (HD Calling), prover uma arquitetura voltada a comunicação via IP entre componentes, melhor desempenho de bateria e dispensar a necessidade de utilização do CSFB (TELECOMTUTORIAL, 2018). Um dos aspectos que garantem o funcionamento do VoLTE mesmo com o possível congestionamento da rede, é a maneira como os túneis são criados (TELECOMTUTORIAL, 2020). Dessa forma, são associados QCIs (Quality of service Identifier) apropriados para cada tipo de serviço, garantindo prioridade para as funcionalidades que requerem maior atenção. Na tabela 5 estão presentes os QCIs de uma rede LTE (RFWIRELESS, 2012):

Tabela 5 -QCIs para redes LTE

LTE QCI	Resource Type	Priority	Packet Delay Budget	Packet Error Loss Rate	Example Services
QCI-1	GBR	2	100ms	10^{-2}	Conversational voice
QCI-2	GBR	4	150ms	10^{-3}	live streaming of conversational voice
QCI-3	GBR	3	50ms	10^{-3}	Real time gaming
QCI-4	GBR	1	300ms	10^{-6}	Non conversational video (Buffered streaming)
QCI-5	Non-GBR	1	100ms	10^{-6}	IMS signalling
QCI-6	Non-GBR	6	300ms	10^{-6}	Video (buffered streaming), TCP based applications
QCI-7	Non-GBR	7	100ms	10^{-3}	Voice, video (live streaming), interactive gaming

QCI-8	Non-GBR	8	300ms	10^{-6}	Voice, video (live streaming), interactive gaming
QCI-9	Non-GBR	9	300ms	10^{-6}	Video (Buffered streaming), TCP based applications

Fonte: RFWIRELESS, 2012 – Modificado.

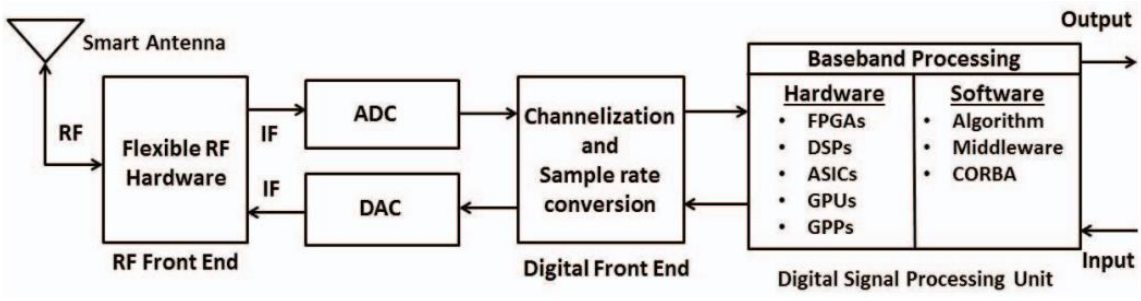
Para o funcionamento adequado dos serviços de voz no VoLTE, utiliza-se do QCI 5 para a sinalização e 1 para o estabelecimento do túnel de serviço de voz.

3.2 RÁDIO DEFINIDO POR SOFTWARE: SDR

Um dos principais objetivos do projeto em questão é avaliar as funcionalidades de um rádio definido por software (SDR). Para a criação da rede LTE emulada, a utilização desse equipamento é indispensável. Os SDRs estão presentes no mercado há pelo menos 30 anos, sendo ainda uma tecnologia em constante evolução e utilizada em diversos cenários. A definição mais genérica de um SDR é que ele se trata de um rádio onde (NATIONAL INSTRUMENTS, 2022) “algumas ou todas as funcionalidades da camada física de transmissão são executadas via software”.

Outro aspecto muito importante dos SDRs é a sua flexibilidade, podendo geralmente trabalhar em diversas frequências diferentes, com diferentes esquemas de modulação e demodulação de sinal, possibilitando a criação de cenários distintos. Devido a essas características flexíveis, os SDRs são extremamente eficazes para cenários de testes ou desenvolvimento, já que, ao contrário do que ocorria anteriormente, onde para o teste com múltiplos cenários era necessário desenvolver rádios para cada um dos casos, o SDR permite a centralização em um equipamento de múltiplas possibilidades de emulação de sistemas de rádio. A Figura 11 apresenta a arquitetura interna de um SDR (SINHA; VERMA; KUMAR, 2016):

Figura 11 - Arquitetura Interna de um SDR

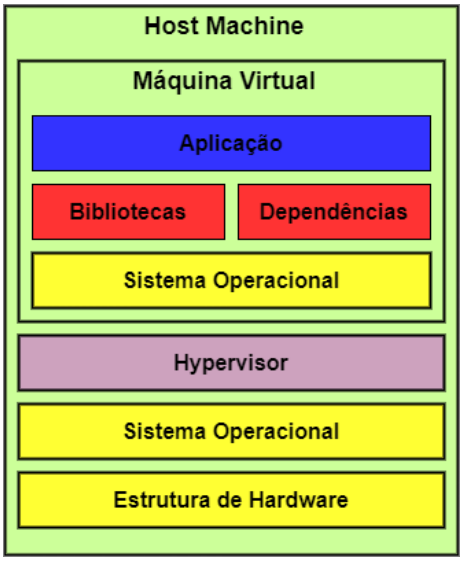


Fonte: SINHA, VERMA E KUMAR, 2016.

3.4 CONTÊINERES: DOCKER E DOCKER-COMPOSE

Outras funcionalidades que ultimamente estão sendo utilizadas no campo das telecomunicações, são as técnicas de virtualização e conteinerização. Uma máquina virtual consiste basicamente da virtualização da estrutura de uma máquina física, onde todos os componentes presentes em um servidor padrão, como kernel e sistema operacional, são criados virtualmente, sendo alocados a máquina virtual uma parte dos recursos da máquina hospedeira. Uma máquina virtual possibilita a criação de ambientes isolados, contendo geralmente gigabytes de armazenamento para comportar as funcionalidades a serem executadas em sua estrutura. A arquitetura de uma máquina virtual está ilustrada na Figura 12, onde pode-se observar que toda a arquitetura de uma máquina física é criada (REDHAT, 2022).

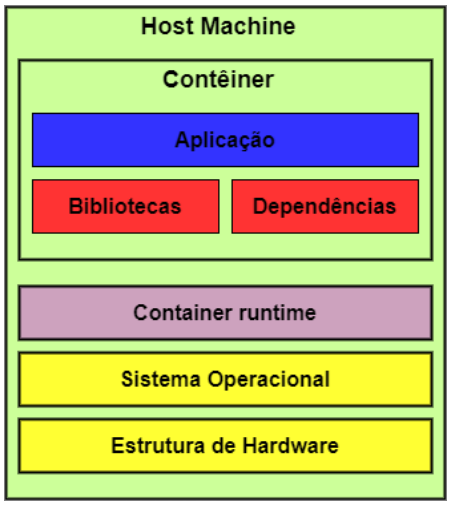
Figura 12 - Arquitetura de uma máquina virtual



Fonte: Autoria Própria.

Outra ferramenta de virtualização utilizada são os contêineres. Os contêineres são geralmente desenvolvidos para uma aplicação específica, contendo megabytes de armazenamento, sendo mais leves e rápidos do que uma máquina virtual. Isso se deve ao fato de eles compartilham do mesmo kernel do host (máquina hospedeira). A Figura 13 representa uma aplicação executada em contêiner (REDHAT, 2022):

Figura 13 - Arquitetura de contêiner



Fonte: Autoria Própria.

Uma das técnicas de conteinerização mais populares utilizadas é o Docker, sendo (DOCKER, 2022) “uma plataforma de código aberto que permite o desenvolvimento, envio e execução de aplicações. O Docker permite a separação das aplicações da infraestrutura de hardware do hospedeiro, para a entrega rápida das aplicações. Utilizando das vantagens do Docker para envio, teste e desenvolvimento rápido, é possível reduzir consideravelmente o atraso entre o desenvolvimento da aplicação e a execução dela em ambiente de produção.”

Ao se trabalhar com aplicações com múltiplos contêineres, a utilização somente das ferramentas do Docker podem acabar se tornando consideravelmente custosas. Uma forma de facilitar esse cenário é utilizar o Docker-compose. O Docker compose é (DOCKER, 2022) “uma ferramenta para definir e executar aplicações com múltiplos contêineres utilizando Docker. Com o compose, pode-se utilizar um arquivo YAML para configurar os serviços da aplicação. Então, com um único comando, é criado e iniciado todos os serviços configurados para a aplicação.” Dessa forma, visando incrementar o desempenho do sistema e facilitar a implementação dos softwares necessários, a ferramenta Docker e Docker-compose serão utilizados em um dos cenários, para a instalação dos softwares a serem utilizados.

4 SOFTWARES E HARDWARES UTILIZADOS

4.1 HARDWARE UTILIZADO PARA A ENODEB: USRP B210

O USRP B210 é um SDR desenvolvido pela ettus research, sendo esse equipamento (ETTUS, 2022) “capaz de transmitir em frequências de 70MHz até 6GHz, contendo uma FPGA Spartan6 da Xilinx e conectividade via USB 3.0”. O B210 contém as seguintes especificações (ETTUS, 2022):

- 2 TX & 2 RX, Half or Full Duplex
- Capacidade para 2x2 MIMO
- Uma FPGA Xilinx Spartan 6 XC6SLX150
- Até 56 MHz de banda em SISO 1x1
- Até 30.72 MHz de banda em MIMO 2x2
- Contém uma fonte DC
- Capacidade para GPIO

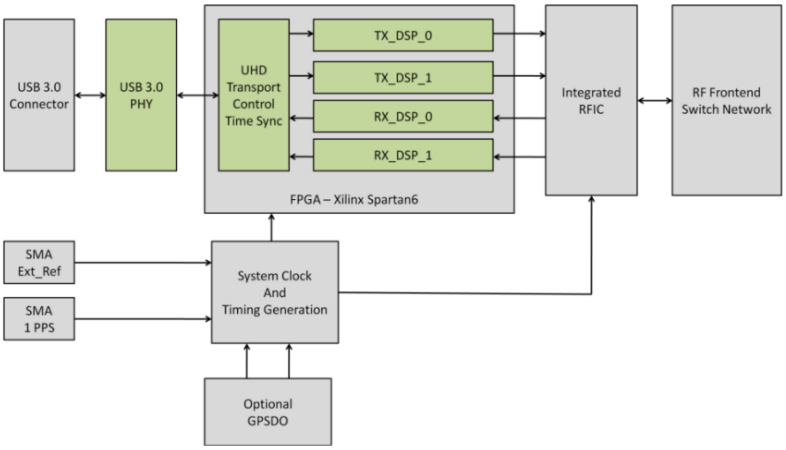
A Figura 14 demonstra o chipset do equipamento:

Figura 14 - USRP B210.

Fonte: ETTUS, 2022 – Modificado.

Esse equipamento permite a emulação de diversas frequências de rádio diferentes, podendo emular redes de telefonia, Wi-Fi, FM entre outras. Para o trabalho em questão, o B210 foi utilizado para emular uma rede de telefonia móvel 4G LTE, emulando mais especificamente a parte de rádio e eNodeB, em conjunto com o software srsRAN. Na Figura 15 está presente o diagrama de blocos representando a arquitetura interna do B210 (ETTUS, 2022):

Figura 15 – Arquitetura interna USRP B210.



Fonte: ETTUS, 2022 – Modificado.

Para garantir um maior sincronismo na transmissão do sinal da rede, foi utilizado um modelo de GPSDO (GPS disciplined oscillator), a Board Mounted GPSDO (TCXO) recomendado para o USRP B200/B210. O GPSDO é um é um oscilador de cristal controlado e disciplinado por GPS (Sistema de posicionamento global), no qual possui versões compatíveis com o USRP B200 e B210. O GPSDO fornece uma referência de alta precisão de 10 MHz e 1 sinal por segundo. Isso permite que os desenvolvedores criem sistemas que atendam a aplicativos com requisitos de sincronização rígidos ou alinhamento de tempo global em 50 ns (ETTUS, 2022).

4.1.1 SOFTWARE UTILIZADO PARA A ENODEB: SRSRAN

O srsRAN (ou srsLTE) é uma solução em código aberto, oferecendo as principais funcionalidades de uma rede de telefonia, como a parte de software para emulação de uma eNodeB, UE e EPC. Para o projeto em questão, foi utilizado em conjunto com o SDR B210 o software srsENB. O srsENB é (SRSRAN, 2022) “uma solução de software voltada para redes 4G LTE e 5G NSA (Not stand Alone). Foi desenvolvido utilizando como base o código C/C ++, podendo ser utilizado em diversas arquiteturas de hardware, como x86, ARM e PowerPC. Sendo executado em sistemas Linux o srsENB pode ser utilizado com qualquer software de EPC que suporte as os protocolos S1AP e GTP-U”.

Abaixo apresenta-se algumas das funcionalidades do srsENB (SRSRAN, 2022):

- É alinhado com release 10 da 3GPP com funcionalidades operacionais até a release 15
- Suporte ao 5G NR para ambos 5G NSA e SA (Stand Alone)
- Configuração para FDD

- Larguras de banda de: 1.4, 3, 5, 10, 15 e 20 MHz
- Possibilidade de captura de pacotes da camada física (MAC)
- Logs de erros detalhados
- Suporte a utilização de SDRs
- Interfaces padrão S1AP e GTP-U para a comunicação com o EPC
- Criptografia do User-plane

A instalação do srsRAN para o cenário em uma máquina física, está presente no apêndice A.

4.2 SOFTWARE UTILIZADO PARA O EPC: OPEN5GS

O Open5gs é uma implantação de EPC focada no cenário do 5G, tendo como base a linguagem C e sendo uma plataforma de código aberto (OPEN5GS, 2022). Na estrutura do Open5gs encontram-se todas as funcionalidades de em um core LTE padrão, onde cada micro serviço da aplicação representa um componente do EPC, seguindo as especificações da 3GPP. Para as redes 4G/5G NSA a estrutura do core contém os seguintes serviços (OPEN5GS, 2022):

- MME - Mobility Management Entity
- HSS - Home Subscriber Server
- PCRF - Policy and Charging Rules Function
- SGWC - Serving Gateway Control Plane
- SGWU - Serving Gateway User Plane
- PGWC/SMF - Packet Gateway Control Plane
- PGWU/UPF - Packet Gateway User Plane

Pode-se observar pela descrição dos serviços acima, que há uma divisão entre plano de controle e usuário para os componentes que trabalham com ambos os planos de comunicação. Essa divisão visa facilitar e isolar planos de comunicação buscando maior eficiência, sem fugir dos padrões estabelecidos pela 3GPP. Outro aspecto interessante está na nomenclatura utilizada para o PGW, que foi dividido em UPF (User Plane function) e SMF (Session Management function), sendo esses serviços presentes no core 5G, apesar da nomenclatura não há diferença nas funcionalidades de um PGW padrão. Algumas das funcionalidades do Open5GS são (OPEN5GS, 2022):

1. Segue os padrões estabelecidos pela release 16 da 3GPP
2. Utiliza dos algoritmos de encriptação AES, Snow3G, ZUC
3. Suporte para IPV6
4. Possibilidade de estabelecimento de múltiplas seções PDU

5. Possibilidade de execução de Handover
6. Possibilidade de utilização de CSFB e SMS
7. Suporte ao VoLTE e VoNR

A implementação do Open5gs para o cenário em uma máquina física, está presente no apêndice B.

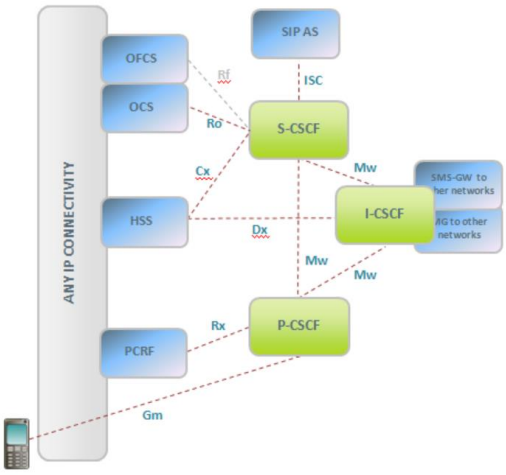
4.2.1 SOFTWARE UTILIZADO PARA O CORE IMS: KAMAILIO

O Kamailio é um software de código aberto que contempla as funcionalidades de um servidor SIP padrão (KAMAILIO, 2022). Para o projeto em questão, as funcionalidades do Kamailio foram empregadas na construção de um core IMS, sendo que diversos módulos para o desenvolvimento desta funcionalidade foram atribuídos à ferramenta. Algumas das funcionalidades do Kamailio para o desenvolvimento do IMS, estão listadas a seguir (KAMAILIO, 2013):

1. Suporte a autenticação via interface Cx e Dx pelo protocolo Diameter, com possibilidade de integração com HSS de outros desenvolvedores
2. Configuração de inicialização de sessão para o IMS com roteamento entre os serviços CSCFs
3. Integração da interface Rx para comunicação com o PCRF de diferentes desenvolvedores

Com essas funcionalidades e o emprego dos módulos necessários para o funcionamento do core IMS, é possível integralizar o Kamailio como servidor IMS, sendo criado os serviços padrão do CSCF e conectando o core IMS ao Open5GS, proporcionando a realização de ligações VoLTE. Na Figura 16 está representada a arquitetura proposta pelos desenvolvedores do Kamailio para o IMS (KAMAILIO, 2013):

Figura 16 – Arquitetura do Kamailio IMS



Fonte: Kamailio, 2013 – Modificado.

A instalação e configuração do Kamailio e dos seus softwares auxiliares, para o cenário em uma máquina física, está presente nos apêndices C, D e E.

4.3 SIM CARDS: SYSMOCOM, PYSIM E COIMS

Para o acesso dos celulares na rede emulada pelo SDR foram utilizados os SIM Cards da fabricante osmocom o sysmocom ISIM-SJA2 (SYSMOCOM, 2022). O sysmocom ISIM-SJA2 está ilustrado na Figura 17:

Figura 17 - SIM card Utilizado

Fonte: Autoria Própria.

Foi escolhido esse modelo de SIM Card pela capacidade de customização do mesmo, já que é possível alterar os índices de segurança através do software pysim, e da chave de segurança fornecida no momento da compra. Para um entendimento sobre o funcionamento dos SIM Cards, é importante falar sobre os parâmetros IMSI (International mobile subscriber identity), OPC (Operator Key) e K (Secret Key). O IMSI é parâmetro utilizado pela rede LTE para a identificação do usuário, sendo composto de 15 ou 16 caracteres, onde os 5 primeiros apresentam o PLMN (Public Land Mobile Network) da rede. O PLMN é composto pelo MCC (Mobile Country Code) que contêm 3 caracteres e representa o país ao qual a rede pertence (no Brasil esse valor é 724) e o MNC (Mobile Network Code), que representa a operadora ou rede utilizada (TELECO, 2022). Dessa forma os outros caracteres do IMSI são preenchidos conforme a operadora ou quem realizou a gravação desejar. O OPC e o K são chaves de segurança verificadas quando o UE realiza a tentativa de conexão à rede, todos esses parâmetros são armazenados no HSS. O Pysim é um programa em Python utilizado para ler, programar e navegar em certos campos ou parâmetros dos SIM Cards programáveis (OSMOCOM, 2022). Foi através desse software que os SIM Cards foram customizados.

Para a funcionalidade do VoLTE existem 3 chaves de segurança que são utilizadas para identificar se o usuário pode acessar o core IMS e realizar ligações: o IMPI (IP Multimedia Public Identity), IMPU (IP Multimedia Private Identity) e MSISDN (Mobile Station Integrated Services Digital Network. O IMPI é uma identidade global alocada pelo HSS contendo as informações de domínio da operadora a ser conectada, já o IMPU serve como número de

telefone para a comunicação do core IMS, sendo associado ao protocolo SIP para a realização das ligações) (TELECOMTUTORIAL, 2020). O MSISDN é o número reconhecido pelo UE para realização das ligações (TELECO, 2022). O CoIMS é um software utilizado para substituir as configurações de IMS e forçar a habilitação do VoLTE usando privilégios de operadora. Isso é necessário para celulares Android que normalmente vem com certas operadoras credenciadas para o VoLTE. Através desse desbloqueio, é possível forçar as configurações de IMS e consequentemente realizar o registro no core IMS e acesso ao serviço VoLTE (HERLESUPREETH, 2020). Na Figura 18 está representado o leitor utilizado para leitura e gravação dos SIM cards:

Figura 18 - Leitor para gravação dos SIM cards

Fonte: Autoria Própria.

A instalação dos softwares de gravação de SIM Cards (CoIMS e pysim) está presente no apêndice F, já o método utilizado para a gravação dos SIM Cards, está presente no apêndice G.

5 TESTES COM UMA REDE 4G LTE EMULADA POR SDR

Foram utilizados 4 celulares para a realização dos testes, sendo eles dos modelos: Motorola G 5G, iPhone 8, iPhone 12 mini e Samsung j7 prime. Todos os modelos possuem suporte ao VoLTE e trabalham em redes 4G LTE. Primeiramente para os aparelhos Android (Motorola 5 5G e Samsung j7 prime), foi necessário configurar a APN (Access Point Name*)* correta para o acesso aos serviços da rede emulada. O método para o cadastramento dos dados dos usuários na rede, está presente no apêndice H.

Para os iPhones, foi constatado que não era necessário alterar as configurações de APN uma vez que sem a inserção das APNs os celulares se conectaram à rede normalmente, tanto para o acesso à internet como para o VoLTE. No caso do Motorola, não foi possível realizar as configurações de APN para VoLTE, mesmo após o desbloqueio do SIM card pelo software CoIMS, isso se deve ao fato de esse modelo só aceitar essas configurações quando estando em uma rede comercial (OPEN5GS, 2021). Seria possível gravar os SIM cards com os dados de uma rede comercial, alterando o MCC e MNC, mas como para os testes em questão, a utilização dos outros três aparelhos supriam as necessidades para os testes VoLTE, essas configurações não foram efetuadas. Dessa forma, o Motorola 5 5G foi utilizado somente para os testes de navegação na internet.

Foram realizados testes em dois cenários distintos, o primeiro onde toda a instalação foi realizada em uma máquina física, e os softwares instalados foram funcionavam como serviços do Linux, sem nenhum tipo de virtualização. Esse cenário está representado na Figura 19:

Figura 19 - Cenário Implementado em um notebook pessoal

Fonte: Autoria Própria

O segundo cenário foi criado utilizando uma máquina virtual instalada em um servidor comercial, e os softwares para a arquitetura LTE foram instalados utilizando contêineres com orquestrador Docker-compose. A arquitetura é semelhante à presente no cenário com a máquina física, com o SDR conectado à porta USB 3.0 do servidor. A Figura 20 demonstra o SDR conectado ao servidor:

Figura 20 - Cenário Implementado em uma VM com Docker

Fonte: Autoria Própria

Para mais detalhes sobre a implementação do cenário utilizando Docker e Docker-compose, basta consultar o apêndice I.

5.1 TESTES DE CONECTIVIDADE NA REDE

Para executar o eNodeB e iniciar a transmissão da rede, no cenário em uma máquina física, deve-se conectar o SDR via cabo USB na porta 3.0 e então executar o comando:

$ sudo srsenb

Dessa forma será iniciado o processo de transmissão do sinal da rede. Nas Figuras 21 e 22, estão os logs do eNodeB e MME respectivamente, indicando o início da transmissão do sinal e o estabelecimento da interface S1-MME:

Figura 21 - Logs do eNodeB indicando a transmissão do sinal (máquina física)

```
Opening 1 channels in RF device=default with args=default
Supported RF device list: UHD file
Trying to open RF device 'UHD'
[INFO] [UHD] linux; GNU C++ version 7.5.0; Boost_106501; UHD_4.2.0.HEAD-0-g46a70d85
[INFO] [LOGGING] Fastpath logging disabled at runtime.
Opening USRP channels=1, args: type=b200,master_clock_rate=23.04e6
[INFO] [UHD RF] RF UHD Generic instance constructed
[INFO] [B200] Detected Device: B210
[INFO] [B200] Operating over USB 3.
[INFO] [B200] Detecting internal GPSDO....
[INFO] [GPS] Found an internal GPSDO: GPSTCXO, Firmware Rev 0.929b
[INFO] [B200] Initialize CODEC control...
[INFO] [B200] Initialize Radio control...
[INFO] [B200] Performing register loopback test...
[INFO] [B200] Register loopback test passed
[INFO] [B200] Performing register loopback test...
[INFO] [B200] Register loopback test passed
[INFO] [B200] Asking for clock rate 23.040000 MHz...
[INFO] [B200] Actually got clock rate 23.040000 MHz.
RF device 'UHD' successfully opened

==== eNodeB started ===
Type <t> to view trace
Setting frequency: DL=2150.0 Mhz, UL=1960.0 MHz for cc_idx=0 nof_prb=25
t
Enter t to stop trace.
```

Fonte: Autoria Própria

Figura 22 - Logs do MME indicando o estabelecimento da interface S1-MME (máquina física)

```
[mme] INFO: eNB-S1 accepted[192.168.56.1]:40373 in s1_path module (../src/mme/s1ap-sctp.c:114)
[mme] INFO: eNB-S1 accepted[192.168.56.1] in master_sm module (../src/mme/mme-sm.c:152)
[mme] INFO: [Added] Number of eNBs is now 1 (../src/mme/mme-context.c:1838)
[mme] INFO: eNB-N2[192.168.56.1] max_num_of_ostreams : 30 (../src/mme/mme-sm.c:194)
[mme] INFO: eNB-S1[192.168.56.1] connection refused!!! (../src/mme/mme-sm.c:210)
[mme] INFO: [Removed] Number of eNBs is now 0 (../src/mme/mme-context.c:1873)
[mme] INFO: eNB-S1 accepted[192.168.56.1]:59256 in s1_path module (../src/mme/s1ap-sctp.c:114)
[mme] INFO: eNB-S1 accepted[192.168.56.1] in master_sm module (../src/mme/mme-sm.c:152)
[mme] INFO: [Added] Number of eNBs is now 1 (../src/mme/mme-context.c:1838)
[mme] INFO: eNB-N2[192.168.56.1] max_num_of_ostreams : 30 (../src/mme/mme-sm.c:194)
```

Fonte: Autoria Própria

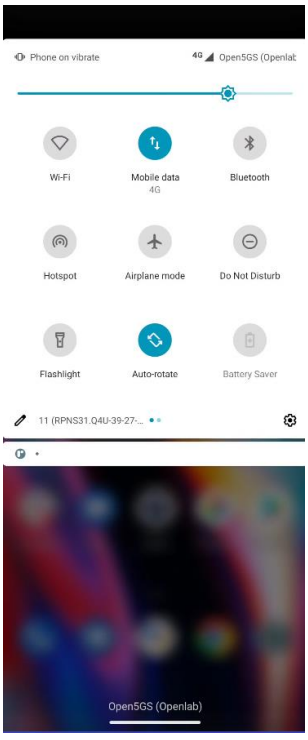
Na Figura 23 está representado o log do MME após a conexão do celular com IMSI=00101000000082, e na Figura 24 um print do celular conectado na rede gerada:

Figura 23 - Conexão do UE efetuada no MME

```
[app] INFO: Configuration: '/etc/open5gs/mme.yaml' (../lib/app/ogs-init.c:126)
[app] INFO: File Logging: '/var/log/open5gs/mme.log' (../lib/app/ogs-init.c:129)
[gtp] INFO: gtp_server() [127.0.0.2]:2123 (../lib/gtp/path.c:30)
[gtp] INFO: gtp_connect() [127.0.0.3]:2123 (../lib/gtp/path.c:60)
[mme] INFO: s1ap_server() [192.168.56.111]:36412 (../src/mme/s1ap-sctp.c:62)
[sctp] INFO: MME initialize...done (../src/mme/app-init.c:33)
[diam] INFO: CONNECTED TO 'hss.localdomain' (SCTP,soc#16): (../lib/diameter/common/logger.c:108)
[diam] INFO: CONNECTED TO 'hss.localdomain' (SCTP,soc#8): (../lib/diameter/common/logger.c:108)
[mme] INFO: eNB-S1 accepted[192.168.56.1]:40640 in s1_path module (../src/mme/s1ap-sctp.c:114)
[mme] INFO: eNB-S1 accepted[192.168.56.1] in master_sm module (../src/mme/mme-sm.c:152)
[mme] INFO: [Added] Number of eNBs is now 1 (../src/mme/mme-context.c:1838)
[mme] INFO: eNB-N2[192.168.56.1] max_num_of_ostreams : 30 (../src/mme/mme-sm.c:194)
[mme] INFO: InitialUEMessage (../src/mme/s1ap-handler.c:223)
[mme] INFO: [Added] Number of eNB-UEs is now 1 (../src/mme/mme-context.c:3491)
[mme] INFO:     ENB_UE_S1AP_ID[1] MME_UE_S1AP_ID[1] TAC[7] CellID[0x19b01] (../src/mme/s1ap-handler.c:363)
[mme] INFO: [001010000000082] Unknown UE by IMSI (../src/mme/mme-context.c:2479)
[mme] INFO: [Added] Number of MME-UEs is now 1 (../src/mme/mme-context.c:2309)
[emm] INFO: [] Attach request (../src/mme/emm-sm.c:203)
[emm] INFO:     IMSI[001010000000082] (../src/mme/emm-handler.c:188)
[mme] INFO: [Added] Number of MME-Sessions is now 1 (../src/mme/mme-context.c:3505)
[emm] INFO: [001010000000082] Attach complete (../src/mme/emm-sm.c:1018)
```

Fonte: Autoria Própria

Figura 24 - Conexão do UE na rede gerada



Fonte: Autoria Própria

No cenário utilizando o Docker-compose, para iniciar a transmissão, basta executar os comandos a seguir para garantir que o EPC, core IMS e eNodeB estejam sendo executados corretamente:

$ docker-compose up -d

$ docker-compose -f srsenb.yaml up -d && docker attach srsenb

É importante esperar que o core IMS e EPC estejam funcionando corretamente antes de executar o eNodeB. Os logs de conexão entre eNodeB e EPC são semelhantes ao observado no cenário em uma máquina física, bastando executar o comando a seguir para efetuar a leitura dos logs.:

$ docker logs <nome-do-componente>

A Figura 25 ilustra os logs do sgwc:

Figura 25 - Logs SGW-C (Docker)

```
fit@testnetwork:~$ docker logs sgwc
Deploying component: 'sgwc-1'
Open5GS daemon v2.4.9-25-g444e182

08/18 19:30:52.942: [app] INFO: Configuration: '/open5gs/install/etc/open5gs/sgwc.yaml' (../lib/app/ogs-init.c:126)
08/18 19:30:52.942: [app] INFO: File Logging: '/open5gs/install/var/log/open5gs/sgwc.log' (../lib/app/ogs-init.c:129)
08/18 19:30:52.953: [gtp] INFO: gtp_server() [172.22.0.5]:2123 (../lib/gtp/path.c:30)
08/18 19:30:52.953: [pfcp] INFO: pfcp_server() [172.22.0.5]:8805 (../lib/pfcp/path.c:30)
08/18 19:30:52.953: [pfcp] INFO: ogs_pfcp_connect() [172.22.0.6]:8805 (../lib/pfcp/path.c:61)
08/18 19:30:52.954: [app] INFO: SGW-C initialize...done (../src/sgwc/app.c:31)
08/18 19:30:52.954: [sgwc] INFO: PFCP associated (../src/sgwc/pfcp-sm.c:172)
```

Fonte: Autoria Própria

A Figura 24 apresenta o celular realizando a conexão na rede gerada, sendo semelhante o estado para ambas as implementações.

5.1.1 TESTE DE VELOCIDADE NA REDE

Foram realizados testes de velocidade com diferentes larguras de banda para o LTE, mais especificamente as larguras de 5 MHz, 10 MHz e 15 MHz para os dois cenários apresentados. Para selecionar a largura de banda desejada basta alterar o número de PRB na configuração do eNodeB, no arquivo enb.conf, como mostra a Figura 26:

Figura 26 - Configuração do eNodeB para 5 MHz de largura de banda

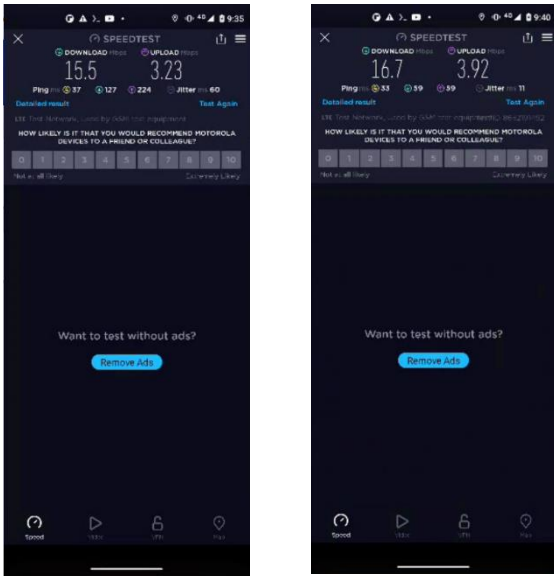
```
#############################################################
[enb]
enb_id = 0x19B
mcc = 001
mnc = 01
mme_addr = 127.0.1.100
gtp_bind_addr = 127.0.1.1
s1c_bind_addr = 127.0.1.1
n_prb = 25
#tm = 4
#nof_ports = 2

#############################################################
```

Fonte: Autoria Própria

Como mostrado na Tabela 3 existe uma associação entre largura de banda e PRB, no caso da Figura 26 a largura de banda é 5 MHz. Portanto, para trabalhar com diferentes larguras de banda utilizando o software srsENB, é necessário alterar o parâmetro de números de PRBs.

Nas Figuras 27, 28 e 29 estão presentes os testes de velocidade realizados no Motorola 5 5G, para ambos os cenários, utilizando as larguras de banda supracitadas respectivamente:

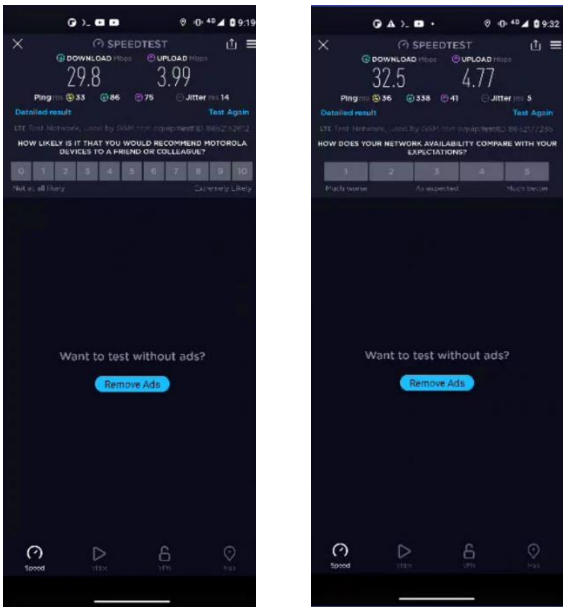
Figura 27 - Teste de velocidade na banda de 5 MHz: cenário (a) máquina física e (b) Docker



(a) (b)

Fonte: Autoria Própria

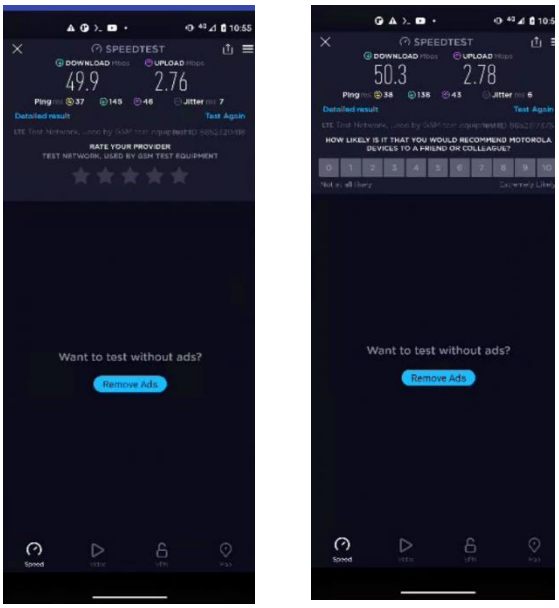
Figura 28 - Teste de velocidade na banda de 10 MHz: cenário (a) máquina física e (b) Docker



(a) (b)

Fonte: Autoria Própria

Figura 29 - Teste de velocidade na banda de 15 MHz: cenário (a) máquina física e (b) Docker



(a) (b)

Fonte: Autoria Própria

Os testes foram realizados próximo ao SDR, dessa forma considerou-se que o esquema de modulação utilizado foi o de 64 QAM. Na tabela 6 estão presentes os valores obtidos nos testes, os esperados teoricamente (ANISIMOFF, 2022) e os obtidos por (MAQSOOD, 2019) considerando o esquema de modulação como 64 QAM:

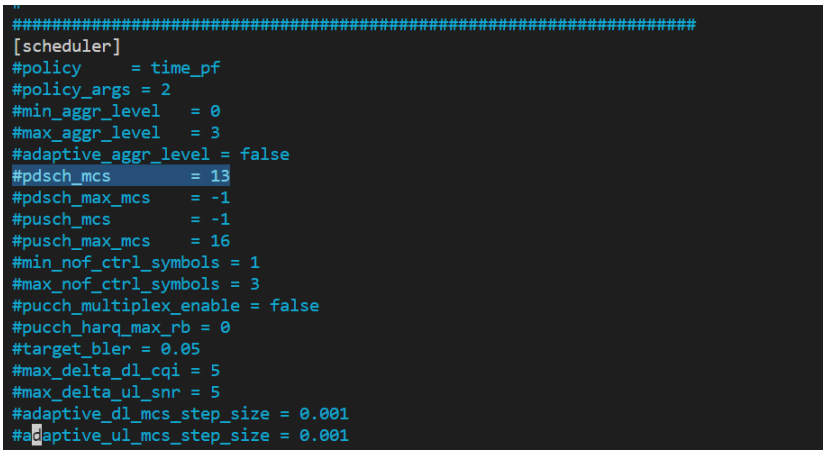
Tabela 6 – Comparação das velocidades obtidas

Largura de banda	***Velocidades Testes (downlink)***	***Velocidades por (MAQSOOD, 2019) (downlink)***	***Velocidades teóricas (ANISIMOFF, 2022) (downlink)***
5MHz	15,5 e 16,7 Mbps	13,8 Mbps	18,336 Mbps
10MHz	29,8 e 32,5 Mbps	30,8 Mbps	36,669 Mbps
15MHz	49,9 e 50,3 Mbps	45,8 Mbps	55,06 Mbps

Fonte: ANISIMOFF; MASQOOS, 2022, 2019 – Modificado

É possível selecionar um esquema de modulação específico para a realização dos testes nas configurações de eNodeB do software srsENB, no arquivo *enb.conf* como demonstrado na Figura 30:

Figura 30 - Configurações de MCS para seleção de modulação



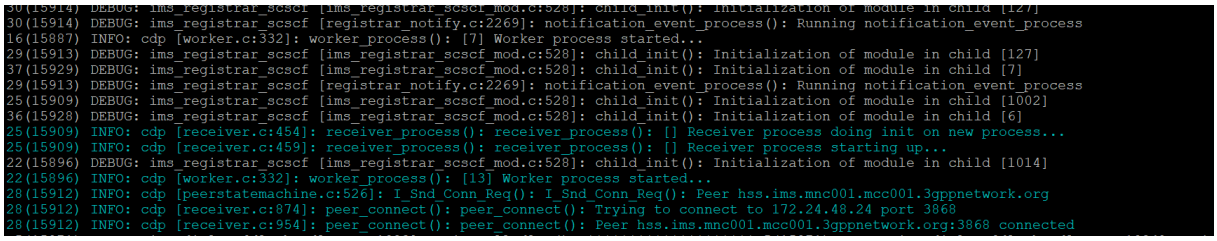
Fonte: Autoria Própria

Porém os testes com um MCS específico ou resultaram em erros de transmissão, ou não obtiveram os valores esperados para aquele tipo de modulação.

5.1.2 TESTE DE LIGAÇÃO VOLTE NA REDE

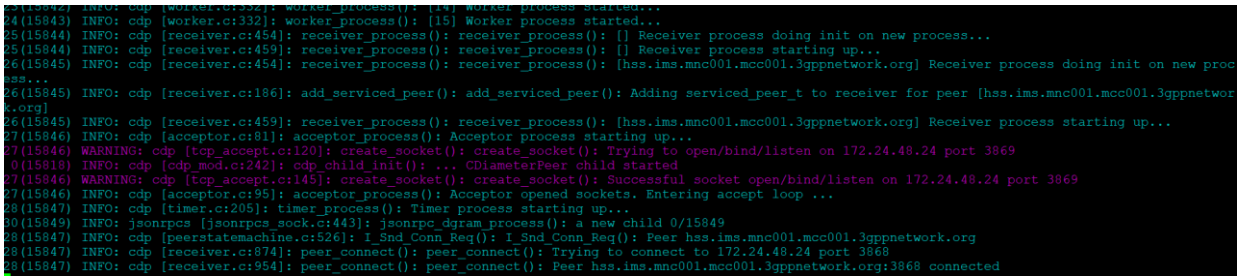
Para os testes VoLTE em ambos os cenários, foram utilizados três modelos de celular: o iPhone 12, iPhone 8 e Samsung j7 prime. No primeiro teste de ligação, foram utilizados dos UEs o Iphone 8 e o Samsung j7 prime. Para o iPhone 8 foi utilizado o sim card com o IMSI=001010000000082 e MSISDN=5515900082 e para o j7prime o IMSI=001010000000087 e MSISDN=5515900087. As figuras 31, 32 e 33 demonstram os logs dos serviços do Kamailio, demonstrando o seu funcionamento correto:

Figura 31 - Log demonstrando o funcionamento do S-CSCF (máquina física)



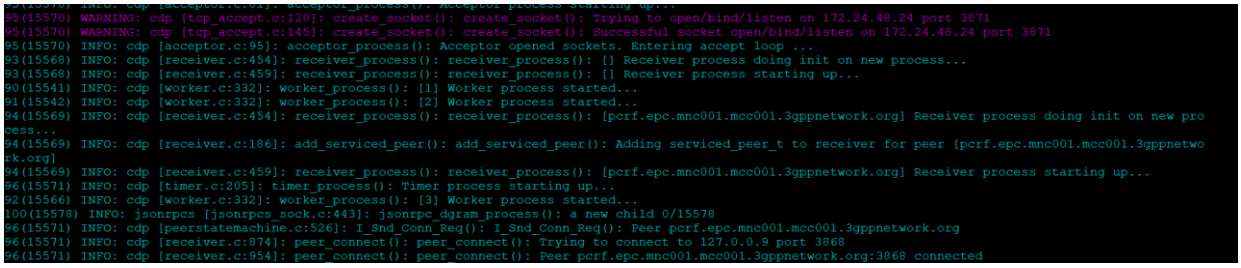
Fonte: Autoria Própria

Figura 32 - Log demonstrando o funcionamento do I-CSCF (máquina física)



Fonte: Autoria Própria

Figura 33 - Log demonstrando o funcionamento do P-CSCF (máquina física)



Fonte: Autoria Própria

Com essas configurações pode-se realizar a conexão dos UEs nos serviços do core IMS. As Figuras 34 e 35 apresentam o log do FHoSS demonstrando que os UEs se conectaram ao core IMS, no cenário com a máquina física:

Figura 34 - Attach do j7 prime na rede VoLTE no FHoSS (máquina física)

Fonte: Autoria Própria

Figura 35 - Attach do iPhone 8 na rede VoLTE no FHoSS (máquina física)ss

Fonte: Autoria Própria

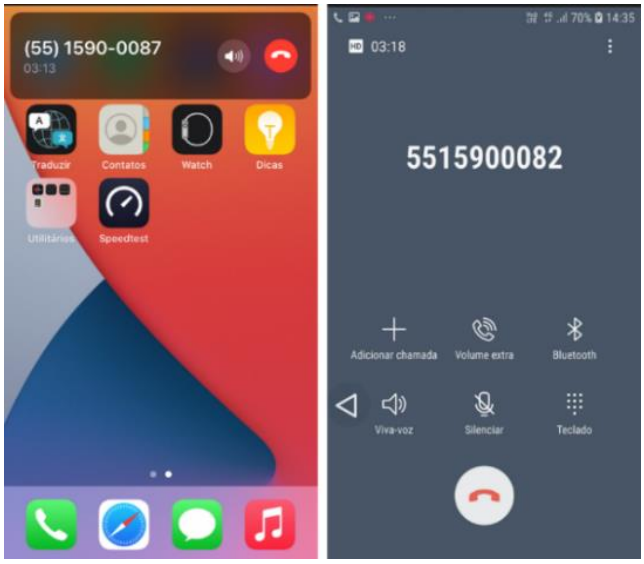
É importante salientar que os logs para ambos os ambientes são semelhantes, bastando executar o comando descrito na seção 5.1 para observar os logs para o cenário com o Docker. No j7 prime pode-se observar o símbolo VoLTE no canto superior da tela ao se conectar no core IMS, a Figura 36 apresenta uma imagem demonstrando essa conexão:

Figura 36 - Attach do j7 prime na rede VoLTE (máquina física)

Fonte: Autoria Própria

Na Figura 37 apresenta a ligação entre iPhone 8 e j7 prime sendo realizada:

Figura 37 - Ligação VoLTE entre iPhone 8 e j7 prime (máquina física)



Fonte: Autoria Própria

Foi realizado um teste de ligação entre os iPhones 8 e 12 mini, seguindo os mesmos procedimentos para o teste anterior no cenário com o Docker. A Figura 38 ilustra a ligação entre os iPhones:

Figura 38 - Ligação VoLTE entre iPhone 8 e iPhone 12 mini (Docker)

Fonte: Autoria Própria

Um ponto importante a ser levantado é que no cenário com a máquina física, foram enfrentados problemas quanto a realização das ligações ao utilizar os celulares Android. Isso se deve ao fato de os mesmos não identificarem a rede como apta para a realização das ligações VoLTE, devido a um parâmetro conhecido como *precondition*. Resumidamente durante o processo de ligação entre dois usuários, se o parâmetro em questão estiver habilitado, são trocadas informações entre os celulares sobre as condições adequadas para a realização da ligação, levando em consideração aspectos relacionados à rede e qualidade de sinal (NETMANIAS, 2022). Aparentemente com o cenário em uma máquina física os celulares Android identificavam que a rede não condizia com os requisitos necessários para a realização da ligação, dessa forma foi necessário desabilitar essa funcionalidade para efetuar a ligação, utilizando do software CoIMS, que possui também um aplicativo para Android para desabilitar alguns parâmetros de checagem da rede. Para desabilitar esse parâmetro foi consultado o fórum dos desenvolvedores do Open5gs e Kamailio IMS, sendo encontrado como resolver o problema em questão (HERLESUPREETH, 2022).

Não foi observado esse problema no cenário com o Docker, provavelmente devido a essa implementação estar mais atualizada quanto aos arquivos de configuração do Kamailio IMS, como pode-se observar ao comparar os arquivos de configuração do P-CSCF dos dois repositórios (HERLESUPREETH, 2020, 2022).

6 CONCLUSÕES

As tecnologias de comunicação móvel de quarta geração continuarão desempenhando um papel crucial na forma como nos comunicamos. Levando esse ponto em consideração, a criação desse cenário 4G, visou demonstrar a complexidade envolvida nas configurações de uma rede privada 4G e analisar as suas funcionalidades. Com o cenário em questão, foi possível usufruir das principais funcionalidades de uma rede de telefonia 4G, como navegar na internet e realizar ligações de voz.

Através do projeto em questão, por toda a implementação ser feita utilizando softwares de código aberto, foi possível pôr em prática e descrever o funcionamento de uma rede móvel de telefonia de maneira mais detalhada, compreendendo algumas das funcionalidades do protocolo LTE. Foram criados dois cenários distintos, um utilizando uma máquina física, e o outro utilizando de técnicas de virtualização e conteinerização com as ferramentas KVM e Docker respectivamente. Do ponto de vista de implementação, a diferença entre os dois cenários foi notável, onde a instalação por contêineres se demonstrou muito mais simples de ser executada. Outra característica para a utilização de contêineres é justamente a portabilidade entre ambientes, bastando o servidor onde o software será instalado contemplar o mesmo kernel da aplicação ou contêiner, devido a esse fato essas técnicas se demonstram extremamente importantes no meio das telecomunicações. Um ponto positivo da instalação realizada diretamente na máquina física é a curva de aprendizado adquirida, onde devido a necessidade de parametrização manual de diversos componentes, foi possível aprimorar os conhecimentos de maneira mais clara sobre as funcionalidades de uma rede LTE. No caso do cenário utilizando o Docker esse aspecto não é observado, já que, basta modificar um arquivo para que todo o ambiente seja parametrizado.

Com as redes já implantadas foi dado início aos testes de desempenho, onde para o uso padrão, ou seja, navegar na internet de maneira ininterrupta, não houve problemas. Os testes para a mudança de modulação de sinal se demonstraram ineficazes, sendo possível a realização de mais testes nesse quesito, a fim obter um maior controle na utilização da rede em questão. Os testes realizados neste projeto utilizaram de celulares comerciais de diferentes fabricantes, o que para a premissa de criação de um ambiente de testes é de extrema importância. Dessa forma, um aspecto importante de ser comentado é a instabilidade enfrentada em alguns modelos quando não utilizada a função de GPS do sistema, no caso do celular Samsung j7 prime, sem a utilização do GPS o mesmo não se conectava a rede de maneira adequada. Não foram encontradas diferenças notáveis entre os dois cenários quanto aos testes de velocidade de acesso

à Internet. Os valores obtidos nos testes de velocidade, se comparados aos esperados teoricamente (ANISIMOFF, 2022) e os valores práticos obtidos em (MAQSOOD, 2019), ficaram dentro de uma faixa aceitável, demonstrando que a rede estava funcionando adequadamente.

Para os testes VoLTE para ambos os ambientes, um dos primeiros problemas enfrentados foi no cadastramento dos usuários ao core IMS. Para os iPhones, não foi enfrentado qualquer tipo de problema quanto a tentativa de conexão, não sendo necessário configurar APNs específicas no UE para a conexão ao core IMS. Isso se deve ao fato desses modelos de celulares já estarem configurados a se conectar ao core IMS de uma rede de testes de maneira automática, criando a APN para a conexão. Para os celulares Android foram enfrentados alguns problemas quanto a realização dessa conexão, principalmente no caso do celular Motorola G 5G, onde mesmo com as configurações desbloqueadas no SIM Card pelo software CoIMS, não foi possível se conectar ao core IMS, configurado para uma rede de testes. Com o Samsung j7 prime, ao desbloquear os SIM Cards e configurar as APNs necessárias, não foram enfrentados problemas para a conexão. Um ponto importante a ser levantado é que no cenário com a máquina física, foram enfrentados problemas quanto a realização das ligações ao utilizar os celulares Android. Isso se deve ao fato de os mesmos não identificarem a rede como apta para a realização das ligações VoLTE, devido a um parâmetro conhecido como precondition. Aparentemente com o cenário em uma máquina física, os celulares Android identificavam que a rede não condizia com os requisitos necessários para a realização da ligação, dessa forma foi necessário desabilitar essa funcionalidade para efetuar a ligação, utilizando do software CoIMS, após essa modificação foi possível realizar as ligações. Para o cenário com o Docker essas modificações não foram necessárias.

As ligações realizadas apresentaram alguns problemas esporádicos de conexão, onde por alguns instantes não era possível realizar uma ligação, e por algumas vezes a ligação efetuada estava sem som ou até mesmo era interrompida de maneira abrupta. Provavelmente isso se deve às limitações de hardware e software enfrentadas na rede em questão, sendo possível também futuramente realizar uma análise mais detalhada a fim de melhorar esses aspectos de conexão.

Levando em consideração o crescimento na utilização de redes privadas LTE e 5G, a utilização do SDR e softwares de código aberto descritos neste projeto, demonstrou que é possível entregar as funcionalidades de uma rede LTE utilizando uma arquitetura de hardware mais simplificada. É válido salientar que para as redes privadas comerciais, a infraestrutura de hardware utilizada, como servidores e rádio por exemplo, é mais robusta do que a empregada

neste projeto. Contudo, é possível traçar um paralelo entre as funcionalidades descritas nesta arquitetura e as empregadas no mercado de redes privadas, principalmente em soluções que utilizam softwares de código aberto (SDXCENTRAL, 2021; FIRECELL, 2021; ALLBESMART, 2022). Essas soluções de rede privada com softwares de código aberto, possuem uma comunidade de desenvolvedores ativa que realiza testes utilizando SDRs, para solucionar possíveis problemas nos softwares utilizados (ONF, 2022).

Em suma, o objetivo de estudo e testes do funcionamento de uma rede privada LTE emulada por SDR, foi alcançado. Pela premissa de utilizar softwares de código aberto, que até certo ponto tem suas configurações transparentes ao desenvolvedor, foi possível entender as funcionalidades mais intrínsecas ao protocolo LTE e manipulá-los a fim de criar a rede emulada da maneira mais customizável possível. Os resultados de velocidade de navegação na internet e realização de ligações VoLTE foram satisfatórios, e como a rede em questão tem a finalidade de testes, o seu desempenho supriu as necessidades do projeto em questão.

REFERÊNCIAS

3GPP, **The Evolved Packet Core,** 2022. Disponível em: https://www.3gpp.org/technologies/keywords-acronyms/100-the-evolved-packet-core. Acesso em: 20 mar. de 2022.

3GPP, **Releases,** 2022. Disponível em: https://www.3gpp.org/specifications/67-releases. Acesso em: 22 mar. de 2022.

3GPP, **LTE**, 2015. Disponível em: https://www.3gpp.org/technologies/keywords-acronyms/98-lte. Acesso em: 29 abr. de 2022.

3GPP**, 3GPP TS 36.300 version 11.3.0 Release**, 2012. Disponível em: https://www.etsi.org/deliver/etsi_ts/136300_136399/136300/11.03.00_60/ts_136300v110300p.pdf. Acesso em: 18 abr. de 2022.

3GLTEINFO, **VoLTE history timeline**, 2016. Disponível em: https://www.3glteinfo.com/volte-history-timeline. Acesso em: 19 jan. de 2022.

ALLBESMART, **5G OPEN-SOURCE**, 2022. Disponível em: https://www.allbesmart.pt/serv_software_defined_5g.php. Acesso em: 5 ago. de 2022.

ANANDTECH, **verizon-4g-lte-two-datacards-wifi-hotspot-massively-reviewed**, 2011. Disponível em: https://www.anandtech.com/show/4289/verizon-4g-lte-two-datacards-wifi-hotspot-massively-reviewed/2. Acesso em: 26 mar. de 2022.

ANISIMOFF, **lte throughput calculator**, 2022. Disponível em: http://anisimoff.org/eng/lte_throughput_calculator.html. Acesso em: 24 jun. de 2022.

Arshad Q. K. Ud Din,Kashif A. U and Quershi I. M., "A Review on the Evolution of Cellular Technologies," 2019 16th International Bhurban Conference on Applied Sciences and Technology (IBCAST), 2019, pp. 989-993, doi: 10.1109/IBCAST.2019.8667173.

AVIAT, **Understanding FDD vs. TDD Microwave Systems,** 2019. Disponível em: https://blog.aviatnetworks.com/technology/understanding-fdd-vs-tdd-microwave-systems. Acesso em: 25 mar. de 2022.

BRECHAZERO, **Na busca pela 5G, não podemos ignorar a importância da 4G**, 2019. Disponível em: https://brechazero.com.br/na-busca-pela-5g-nao-podemos-ignorar-a-importancia-da-4g/ Acesso em: 27 abr. de 2022.

CARVALHO M. R., **Diferenças entre LTE-FDD e LTE-TDD,** 2013. Disponível em: https://www.ricardomcarvalho.pt/blog/diferencas-entre-os-modos-lte-fdd-e-lte-tdd/. Acesso em: 25 mar. de 2022.

Curpen R., Fernoaga V., Robu D. and Sandu F., "Open-LTE Call Emulator in Software Defined Radio," 2019 18th RoEduNet Conference: Networking in Education and Research (RoEduNet), 2019, pp. 1-6, doi: 10.1109/ROEDUNET.2019.8909662.

DOCKER, **Docker overview,** 2022. Disponível em: https://docs.docker.com/get-started/overview/. Acesso em: 19 jan. de 2022.

DOCKER, **Overview of Docker Compose,** 2022. Disponível em: https://docs.docker.com/compose/. Acesso em: 19 jan. de 2022.

ELETRONIC-NOTES, **3GPP 3GPP Specification Release Numbers,** 2022. Disponível em: https://www.electronics-notes.com/articles/connectivity/3gpp/standards-releases.php. Acesso em: 24 mar. de 2022.

ERICSSON, **Network slicing**, 2022. Disponível em: https://www.ericsson.com/en/network-slicing. Acesso em: 30 jun. de 2022.

ETTUS, **USRP B200/B210 Product Overview**, 2022. Disponível em: https://www.ettus.com/wp-content/uploads/2019/01/b200-b210_spec_sheet.pdf. Acesso em: 23 jan. de 2022.

ETTUS, **Internal GPSDO Application Notes (USRP-B2x0 Models)**, 2022. Disponível em: https://files.ettus.com/manual/page_gpsdo_b2x0.html. Acesso em: 23 jan. de 2022.

ETTUS, **Building and Installing the USRP Open-Source Toolchain (UHD and GNU Radio) on Linux**, 2022. Disponível em: https://kb.ettus.com/Building_and_Installing_the_USRP_Open-Source_Toolchain_(UHD_and_GNU_Radio)_on_Linux. Acesso em: 5 fev. de 2022.

FIRECELL, **OAI for Private 5G**, 2021. Disponível em: https://firecell.io/oai-for-private-5g/. Acesso em: 6 ago. de 2022.

GSMA, **The Mobile Economy**, 2021. Disponível em: https://www.gsma.com/mobileeconomy/wpcontent/uploads/2021/11/GSMA_ME_LATAM_2021.pdf. Acesso em: 27 abr. de 2022.

GSMA, **IMS Profile for Voice and SMS**, 2020. Disponível em: https://www.gsma.com/newsroom/wp-content/uploads/IR.92-v15.0-4.pdf. Acesso em: 19 jan. de 2022.

Hua M., Ren B., Wang M., Zou J., Yang C. and . Liu T, "Performance Analysis of OFDMA and SC-FDMA Multiple Access Techniques for Next Generation Wireless Communications," 2013 IEEE 77th Vehicular Technology Conference (VTC Spring), 2013, pp. 1-4, doi: 10.1109/VTCSpring.2013.6692696.

HERLESUPREETH, **CoIMS_Wiki**, 2020. Disponível em: https://github.com/herlesupreeth/CoIMS_Wiki. Acesso em: 29 jan. de 2022.

HERLESUPREETH, **Kamailio IMS Config**, 2020. Disponível em: https://github.com/herlesupreeth/Kamailio_IMS_Config. Acesso em: 7 fev. de 2022.

HERLESUPREETH, **Kamailio IMS Config Issues 6**, 2022. Disponível em: https://github.com/herlesupreeth/Kamailio_IMS_Config/issues/6. Acesso em: 15 jun. de 2022.

HERLESUPREETH, **Docker Open5gs**, 2022. Disponível em: https://github.com/herlesupreeth/docker_open5gs. Acesso em: 8 fev. de 2022.

HERLESUPREETH, **FHoSS**, 2022. Disponível em: https://github.com/herlesupreeth/FHoSS. Acesso em: 8 fev. de 2022.

HERLESUPREETH, **CoIMS Wiki**, 2022. Disponível em: https://github.com/herlesupreeth/CoIMS_Wiki. Acesso em: 19 jan. de 2022.

HERLESUPREETH, **Docker Open5gs**, 2022. Disponível em: https://github.com/herlesupreeth/docker_open5gs. Acesso em: 24 fev. de 2022.

KAMAILIO, **Welcome To Kamailio – The Open Source SIP Server**, 2022. Disponível em: https://www.kamailio.org/w/. Acesso em: 28 jan. de 2022.

KAMAILIO, **What is the IP Multimedia Subsystem (IMS) in Kamailio 4.0?**, 2013. Disponível em: https://www.kamailio.org/w/2013/05/ims-kamailio/. Acesso em: 28 jan. de 2022.

KEYSIGHT, **LTE Physical Layer Overview**, 2021. Disponível em: https://rfmw.em.keysight.com/wireless/helpfiles/89600b/webhelp/subsystems/lte/content/lte_overview.htm. Acesso em: 26 mar. de 2022.

LINUX KVM, **Kernel Virtual Machine**, 2022. Disponível em: https://www.linux-kvm.org/page/Main_Page. Acesso em: 22 fev. de 2022.

Maqsood B., 'Implementation and performance analysis of software defined radio (SDR) based LTE platform for truck connectivity application', Dissertation, 2019.

Mohammed Alnaas, Elmabruk Laias, Saleh Alghol, Hosian Akeel, "An Overview of the Development of Mobile Wireless Communication Technologies," American Journal of Computer Science and Engineering (AMCSE), vol. 5, Issue 2, pp. 22-29, April 9, 2018.

Moray Rumney, **LTE and the Evolution to 4G Wireless: Design and Measurement Challenges, 2nd Edition**. 2nd ed. Chichester, West Sussex, U.K: Wiley, 2013.

MOTOROLA, **WHAT IS PRIVATE LTE?**, 2022. Disponível em: https://www.motorolasolutions.com/en_us/solutions/what-is-private-lte.html. Acesso em: 16 jun. de 2022.

Navita and Amandeep, "Performance analysis of OFDMA, MIMO and SC-FDMA technology in 4G LTE networks," 2016 6th International Conference - Cloud System and Big Data Engineering (Confluence), 2016, pp. 554-558, doi: 10.1109/CONFLUENCE.2016.7508181.

NATIONAL INSTRUMENTS, **Software Defined Radio: Past, Present, and Future**, 2022. Disponível em: https://www.ni.com/pt-br/innovations/white-papers/17/software-defined-radio--past--present--and-future.html. Acesso em: 23 jan. de 2022.

NETMANIAS, **Dedicated Bearer setup in LTE and impact on VoLTE Precondition**, 2022. Disponível em: https://netmanias.com/en/post/blog/11789/lte-volte/dedicated-bearer-setup-in-lte-and-impact-on-volte-precondition. Acesso em: 8 jun. de 2022.

Nóvoa, Lucas & Tavares, Virgínia & Nahum, Cleverson & Batista, Pedro & Klautau, Aldebaro. (2020). RAN Slicing using OpenAirInterface and FlexRAN in a Virtualized Scenario. 10.14209/SBRT.2020.1570657929.

ONF SD-RAN **Hardware Installation**, 2022. Disponível em: https://docs.sd-ran.org/master/sdran-in-a-box/docs/HW_Installation_intro.html. Acesso em: 5 jan. de 2022.

OPEN5GS, **About**, 2022. Disponível em: https://open5gs.org/open5gs/about/. Acesso em: 28 jan. de 2022.

OPEN5GS, **Quickstart**, 2022. Disponível em: https://open5gs.org/open5gs/docs/guide/01-quickstart/. Acesso em: 28 jan. de 2022.

OPEN5GS, **Issues 752**, 2021. Disponível em: https://github.com/open5gs/open5gs/issues/752. Acesso em: 19 jan. de 2022.

OPEN5GS, **VoLTE setup**, 2022. Disponível em: https://open5gs.org/open5gs/docs/tutorial/02-VoLTE-setup/. Acesso em: 5 fev. de 2022.

OPEN AIR INTERFACE, **Projects**, 2022. Disponível em: https://openairinterface.org/projects/. Acesso em: 30 jun. de 2022.

ORACLE, **Java SE 7 Archive Downloads**, 2022. Disponível em: https://www.oracle.com/java/technologies/javase/javase7-archive-downloads.html. Acesso em: 8 fev. de 2022.

OSMOCOM,**pysim** , 2022. Disponível em: https://github.com/osmocom/pysim. Acesso em: 19 jan. de 2022.

Qadeer M. A., Khan A. H., Ansari J. A. and Waheed S., "IMS Network Architecture," *2009 International Conference on Future Computer and Communication*, 2009, pp. 329-333, doi: 10.1109/ICFCC.2009.106.

REDHAT, **Containers e máquinas virtuais (VMs),** 2022. Disponível em: https://www.redhat.com/pt-br/topics/containers/containers-vs-vms. Acesso em: 19 jan. de 2022

REIS R., **4G - LTE/LTE-A Coursework for Computer Networks II**, 2014. Disponível em: https://www.gta.ufrj.br/ensino/eel879/trabalhos_vf_2014_2/rafaelreis/ofdma_scfdma.html. Acesso em: 26 mar. de 2022.

RFWIRELESS, **LTE QoS Quality of Service, class identifier(QCI), QoS in LTE**, 2012. Disponível em: https://www.rfwireless-world.com/Tutorials/LTE-QoS.html. Acesso em: 22 jan. de 2022.

SDXCENTRAL, **ONF Exposes Three Branches of Private Network Project**, 2021. Disponível em: https://www.sdxcentral.com/articles/news/onf-exposes-three-branches-of-private-network-project/2021/06/. Acesso em: 5 ago. de 2022.

SIGNAL BOOSTERS, **lte-vs-4g-vs-5g-whats-the-difference**, 2022. Disponível em: https://www.signalboosters.com/blog/lte-vs-4g-vs-5g-whats-the-difference/. Acesso em: 29 abr. de 2022.

Sinha D., Verma A. K. and Kumar S., "Software defined radio: Operation, challenges and possible solutions," 2016 10th International Conference on Intelligent Systems and Control (ISCO), 2016, pp. 1-5, doi: 10.1109/ISCO.2016.7727079.

SIPWISE, **RTPengine**, 2020. Disponível em: https://github.com/sipwise/rtpengine. Acesso em: 7 fev.de 2022.

SRSLTE, **The srsLTE project is evolving**, 2022. Disponível em: https://www.srslte.com/srslte-srsran. Acesso em: 30 jan. de 2022.

SRS, **SRS ENB**, 2022. Disponível em: https://www.srs.io/products/#SRS-ENB. Acesso em: 25 jan. de 2022.

SRS, **Installation Guide**, 2022. Disponível em:https://docs.srsran.com/en/latest/general/source/1_installation.html. Acesso em: 19 jan. de 2022.

SRSRAN,**srsRANFeatures**, 2019. Disponível em: https://docs.srsran.com/en/latest/feature_list.html#srsenb. Acesso em: 25 jan. de 2022.

SYSMOCOM, **sysmousim-manual**, 2022. Disponível em: https://www.sysmocom.de/manuals/sysmousim-manual.pdf. Acesso em: 28 jan. de 2022.

Tavares, Virgínia & Pinto, Lucas & Couto, Gabriel & Klautau, Aldebaro & Müller, Francisco. (2019). Emulação de Rede LTE Usando OpenAirInterface e Análise de Tráfego Via Protocolo de Rede. 10.14209/sbrt.2019.1570559016.

TECHTARGET, **Definition FrontHaul**, 2021. Disponível em: https://www.techtarget.com/searchmobilecomputing/definition/fronthaul. Acesso em: 19 abr. de 2022.

TECMINT, **How to Create Virtual Machines in KVM Using Virt-Manager**, 2022. Disponível em: https://www.tecmint.com/create-virtual-machines-in-kvm-using-virt-manager/. Acesso em: 20 fev. de 2022.

TELECO, **HSPA e WiMax Móvel I: Tecnologias**, 2022. Disponível em: https://www.teleco.com.br/tutoriais/tutorialhspawimax1/pagina_2.asp. Acesso em: 28 abr. de 2022.

TELECO, **Redes LTE I: Protocolos e Divisão da Rede de Acesso**, 2022. Disponível em: https://www.teleco.com.br/tutoriais/tutorialredeslte/pagina_3.asp. Acesso em: 18 abr. de 2022.

TELECO, **Rede LTE: Rede e Mobilidade**, 2022. Disponível em: https://www.teleco.com.br/tutoriais/tutorialltemcp/pagina_2.asp. Acesso em: 19 abr. de 2022.

TELECO, **Redes LTE: Interfaces e Protocolos**, 2022. Disponível em: https://www.teleco.com.br/tutoriais/tutorialredeslteident/pagina_2.asp. Acesso em: 19 abr. de 2022.

TELECO, **Rede GSM: Conceitos e Geografia da Rede**, 2022. Disponível em: https://www.teleco.com.br/tutoriais/tutorialredegsm/pagina_2.asp. Acesso em: 29 abr. de 2022.

TELECOMTUTORIAL, **What is VoLTE – IMS Overview, Basics & Fundamentals**, 2018. Disponível em: https://www.telecomtutorial.info/post/what-is-volte-ims-overview-basics-fundamentals. Acesso em: 19 jan. de 2022.

TELECOMTUTORIAL, **VoLTE Interfaces Protocols & IMS Stack**, 2020. Disponível em: https://www.telecomtutorial.info/post/volte-interfaces-protocols-ims-stacks. Acesso em: 20 jan. de 2022.

TELECOMTUTORIAL, **VoLTE SIP IMS registration Call Flow Procedure & Default Vs Dedicated Bearer in LTE**, 2020. Disponível em: https://www.telecomtutorial.info/post/volte-sip-ims-registration-call-flow-procedure-default-vs-dedicated-bearer-in-lte. Acesso em: 21 jan. de 2022.

TELECOMTUTORIAL, **VoLTE IMS Architecture**, 2020. Disponível em: https://www.telecomtutorial.info/post/volte-ims-architecture. Acesso em: 29 jan. de 2022.

TELESINTESE, **ANATEL: APENAS 14% DO TERRITÓRIO BRASILEIRO TEM COBERTURA 3G E 4G**, 2020. Disponível em: https://www.telesintese.com.br/anatel-apenas-14-do-territorio-brasileiro-tem-cobertura-3g-e-4gs. Acesso em: 14 abr. de 2022.

TELESINTESE, **REDES PRIVADAS LTE/5G AINDA SÃO POUCAS NO GLOBO**, 2022. Disponível em: https://www.telesintese.com.br/redes-privadas-lte-5g-ainda-sao-poucas-no-globo/. Acesso em: 14 jun. de 2022.

TOPIN, **According-to-berg-insight-the-deployment-of-private-lte-and-5g-networks-will-increase-by-up-to-10-times-in-the-next-five-years**, 2022. Disponível em: https://topin.io/en/news/according-to-berg-insight-the-deployment-of-private-lte-and-5g-networks-will-increase-by-up-to-10-times-in-the-next-five-years. Acesso em: 15 jun. de 2022.

Tran T. X., Younis A. and Pompili D., "Understanding the Computational Requirements of Virtualized Baseband Units Using a Programmable Cloud Radio Access Network Testbed," 2017 IEEE International Conference on Autonomic Computing (ICAC), 2017, pp. 221-226, doi: 10.1109/ICAC.2017.42.

UBUNTU RELEASES, **Ubuntu 20.04.4 LTS (Focal Fossa)**, 2022. Disponível em: https://releases.ubuntu.com/focal/ Acesso em: 20 fev. de 2022.

YADAV R., " Challenges and Evolution of Next generation Wireless Communication," IMECS, vol. 2, March 2017.

APÊNDICE A – IMPLEMENTAÇÃO E CONFIGURAÇÃO DO SRSRAN

O primeiro passo para a integração do software srsRAN com o SDR, foi realizar a instalação do driver para o SDR da Ettus research, chamado de UHD (ETTUS, 2022). Para a instalação do UHD foi necessário executar os comandos a seguir:

$ sudo add-apt-repository ppa:ettusresearch/uhd && sudo apt update && apt -y install libuhd-dev libuhd4.2.0 uhd-host

Ao executar esses comandos será adicionado os pacotes para a instalação do UHD e será feita a sua instalação. Para evitar a perda do caminho para o driver foi adicionado o caminho para a biblioteca no arquivo *$HOME/.bashrc* , adicionando ao final do arquivo as seguintes linhas:

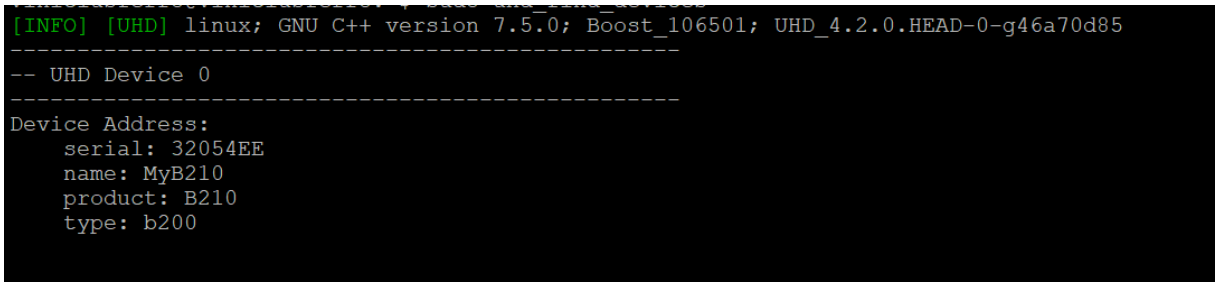
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/local/lib

Após o UHD instalado foi necessário realizar o download das imagens do FPGA. Para isso, foi necessário garantir que o SDR estivesse conectado ao computador, utilizando cabo USB na porta USB 3.0, e então executar o comando:

$ sudo uhd_images_downloader

Para verificar se o UHD foi instalado corretamente e na versão desejada, basta digitar *$ sudo uhd_find_devices* como demonstrado na figura 39:

Figura 39 - Saída para comando sudo uhd_find_devices



Fonte: Autoria Própria.

Com a instalação do UHD finalizada, foi realizada a instalação e configuração do srsRAN.

Primeiramente, para a instalação do software via pacote linux, foram executados os comandos a seguir (SRSRAN):

$ sudo add-apt-repository ppa:softwareradiosystems/srsran && sudo apt-get update && sudo apt-get install srsran -y

Após a instalação do srsRAN, para o funcionamento adequado da aplicação foi necessário configurar alguns parâmetros, presentes em sua pasta de configuração, que por padrão se encontra no diretório *$HOME*.config/srsran:

$ cd /root/.config/srsran

Nesse diretório encontram-se os seguintes arquivos: rr.conf, mbms.conf, ue.conf, epc.conf, user_db.csv, rb.conf e enb.conf. Para o caso de utilização do srsEPC e srsUE é válido configurar os arquivos epc.conf e ue.conf porém, no caso da implementação em questão, não será utilizada essas funcionalidades de EPC e emulação de UE do srsRAN. Dessa forma, os únicos arquivos que foram modificados são o enb.conf e o drb.conf. Para o caso do drb.conf, foi realizada a seguinte configuração, atribuindo o QCI 5 e 1 para o VoLTE:

Figura 40 – QCIs para o VoLTE arquivo rb.conf

```
{
  qci=1;
  pdcp_config = {
    discard_timer = 100;
    pdcp_sn_size = 12;
  }
  rlc_config = {
    ul_um = {
      sn_field_length = 10;
    };
    dl_um = {
      sn_field_length = 10;
      t_reordering    = 50;
    };
  };
  logical_channel_config = {
    priority = 2;
    prioritized_bit_rate   = -1;
    bucket_size_duration  = 100;
    log_chan_group = 1;
  };
},
  qci=5;
  pdcp_config = {
    discard_timer = -1;
    status_report_required = true;
  }
  rlc_config = {
    ul_am = {
      t_poll_retx = 80;
      poll_pdu = 128;
      poll_byte = 125;
      max_retx_thresh = 4;
    };
    dl_am = {
      t_reordering = 80;
      t_status_prohibit = 60;
    };
  };
  logical_channel_config = {
    priority = 11;
    prioritized_bit_rate   = -1;
    bucket_size_duration  = 100;
    log_chan_group = 2;
  };
},
{
```

Fonte: Autoria Própria.

Para o enb.conf, deve-se configurar o mn/mcc utilizado e o IP do MME do EPC (Open5gs). Dessa forma para o enb.conf foram realizadas as seguintes configurações, na Figura 41:

Figura 41 - Configuração do arquivo enb.conf

```
#####################################################################
# eNB configuration
#
# enb_id:               20-bit eNB identifier.
# mcc:                  Mobile Country Code
# mnc:                  Mobile Network Code
# mme_addr:             IP address of MME for S1 connnection
# gtp_bind_addr:        Local IP address to bind for GTP connection
# gtp_advertise_addr:   IP address of eNB to advertise for DL GTP-U Traffic
# s1c_bind_addr:        Local IP address to bind for S1AP connection
# n_prb:                Number of Physical Resource Blocks (6,15,25,50,75,100)
# tm:                   Transmission mode 1-4 (TM1 default)
# nof_ports:            Number of Tx ports (1 port default, set to 2 for TM2/3/4)
#
#####################################################################
[enb]
enb_id = 0x19B
mcc = 001
mnc = 01
mme_addr = 127.0.1.100
gtp_bind_addr = 127.0.1.1
s1c_bind_addr = 127.0.1.1
n_prb = 25
#tm = 4
#nof_ports = 2
```

Fonte: Autoria Própria.

APÊNDICE B – IMPLEMENTAÇÃO E CONFIGURAÇÃO DO OPEN5GS

Para a instalação o do Open5GS, no cenário com uma máquina física, foram realizados os passos necessários para a sua configuração e comunicação com o srsENB e Kamailio core IMS (OPEN5GS, 2022). Para o Ubuntu, existe a opção de instalação do Open5gs por pacotes, tornando o processo de build e instalação em si, muito mais simplificado. Primeiramente foi necessário instalar os pacotes a seguir:

$ sudo apt update

$ sudo apt install software-properties-common

$ sudo add-apt-repository ppa:open5gs/latest

$ sudo apt update

$ sudo apt install open5gs

Após a finalização da instalação desses pacotes, o Open5GS estará configurado na máquina, sendo necessário instalar o WebUI, que se trata da interface gráfica para cadastramento dos SIM Cards:

$ sudo apt update

$ sudo apt install curl

$ curl -fsSL https://deb.nodesource.com/setup_14.x | sudo -E bash -

$ sudo apt install nodejs

$ curl -fsSL https://open5gs.org/open5gs/assets/webui/install | sudo -E bash

Para facilitar a utilização da plataforma Open5gs, foi criado alguns scripts para otimizar tempo na hora da inicialização e checagem de status dos serviços. Dessa forma, abaixo estão ilustrados os scripts para reiniciar, coletar status e parar os serviços:

O script para reiniciar os serviços do Open5GS está presente na Figura 42:

Figura 42 - Script para reiniciar os serviços do Open5GS

```
#!/bin/bash
sudo systemctl restart open5gs-mmed
sudo systemctl restart open5gs-sgwcd
sudo systemctl restart open5gs-smfd
sudo systemctl restart open5gs-sgwud
sudo systemctl restart open5gs-upfd
sudo systemctl restart open5gs-hssd
sudo systemctl restart open5gs-pcrfd
sudo systemctl restart open5gs-webui
```

Fonte: Autoria Própria.

Na Figura 43 está representado o script para coletar o status dos serviços do Open5GS:

Figura 43 - Script para coletar o status dos serviços do Open5GS

```
#!/bin/bash
sudo systemctl status open5gs-mmed
sudo systemctl status open5gs-sgwcd
sudo systemctl status open5gs-smfd
sudo systemctl status open5gs-amfd
sudo systemctl status open5gs-sgwud
sudo systemctl status open5gs-upfd
sudo systemctl status open5gs-hssd
sudo systemctl status open5gs-pcrfd
sudo systemctl status open5gs-webui
```

Fonte: Autoria Própria.

O script para parar os serviços do Open5GS está ilustrado na Figura 44:

Figura 44 - Script para parar os serviços do Open5GS

```
#!/bin/bash
sudo systemctl stop open5gs-mmed
sudo systemctl stop open5gs-sgwcd
sudo systemctl stop open5gs-smfd
sudo systemctl stop open5gs-amfd
sudo systemctl stop open5gs-sgwud
sudo systemctl stop open5gs-upfd
sudo systemctl stop open5gs-hssd
sudo systemctl stop open5gs-pcrfd
sudo systemctl stop open5gs-webui
```

Fonte: Autoria Própria.

Outro script criado foi o de monitoramento dos logs de erro, abaixo tem-se um exemplo do script para a captura dos logs do MME:

#!/bin/bash

tail - f /var/log/open5gs/mme.log

Para os outros serviços bastou modificar o nome do arquivo .log. Para garantir a execução desses scripts é necessário dar permissão de execução para os arquivos, através do comando *chmod 777*. Para a configuração do Open5GS bastou configurar os arquivos dos serviços no diretório /etc/open5gs/. A instalação tanto dos softwares de eNodeB, EPC e core IMS foram realizadas na mesma máquina, dessa forma muitas das conexões se deram na interface interna de loopback (*lo*). Para a configuração do MME, foi modificado o arquivo mme.yaml como demonstrado na Figura 45:

Figura 45 - Configuração do mme.yaml

```
mme:
    freeDiameter: /etc/freeDiameter/mme.conf
    slap:
#       - addr: 127.0.0.2
      - addr: 127.0.1.100
    gtpc:
      - addr: 127.0.0.2
    gummei:
      plmn_id:
        mcc: 001
        mnc: 01
      mme_gid: 2
      mme_code: 1
    tai:
      plmn_id:
        mcc: 001
        mnc: 01
      tac: 7
    security:
        integrity_order : [ EIA2, EIA1, EIA0 ]
        ciphering_order : [ EEA0, EEA1, EEA2 ]
    network_name:
        full: OpenLab-X300
    mme_name: open5gs-mme0
```

Fonte: Autoria Própria.

Como observado na Figura 45, foi alterado o valor do IP para o S1-MME, que se trata da interface entre MME e eNodeB, esse IP deve estar de acordo com o indicado no enb.conf do srsENB.

Para o funcionamento do VoLTE são necessárias outras configurações, primeiramente deve-se mudar o realm dos serviços do Open5GS para o domínio que será utilizado, os arquivos que devem ser alterados estão no diretório /etc/freeDiameter, na Figura 46 está presente um exemplo da alteração necessária, para o arquivo mme.conf:

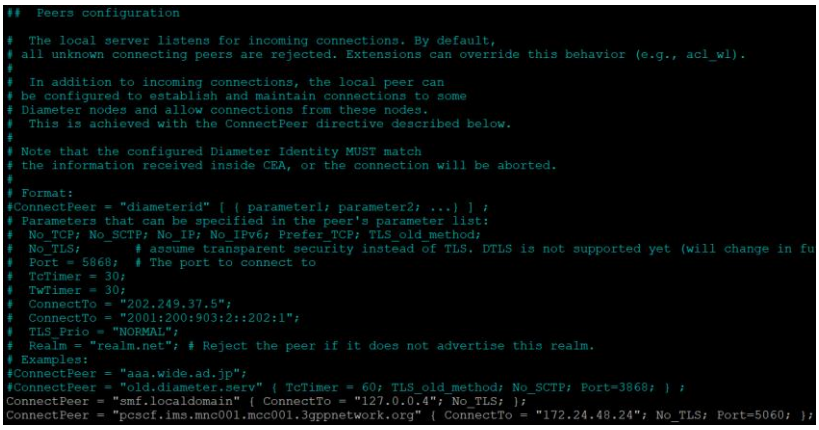
Figura 46 - Configuração de mme.conf

```
#############################################################
##  Peer identity and realm

# The Diameter Identity of this daemon.
# This must be a valid FQDN that resolves to the local host.
# Default: hostname's FQDN
#Identity = "aaa.koganei.freediameter.net";
Identity = "mme.localdomain";

# The Diameter Realm of this daemon.
# Default: the domain part of Identity (after the first dot).
#Realm = "koganei.freediameter.net";
Realm = "epc.mnc001.mcc001.3gppnetwork.org";
```

Fonte: Autoria Própria.

De forma similar foram alterados os mesmos campos nos serviços do HSS, SMF e PCRF, com a diferença de no PCRF ser necessário alterar também o seguinte campo, como mostra a Figura 47:

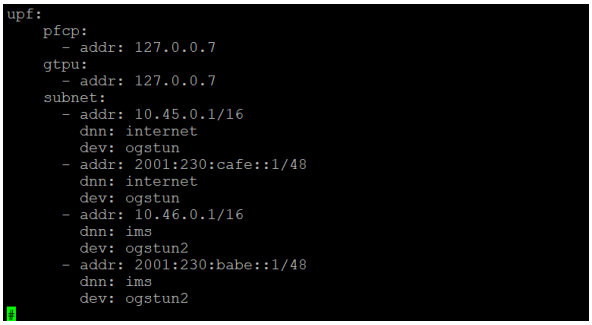
Figura 47 – Configuração do pcrf.conf



Fonte: Autoria Própria.

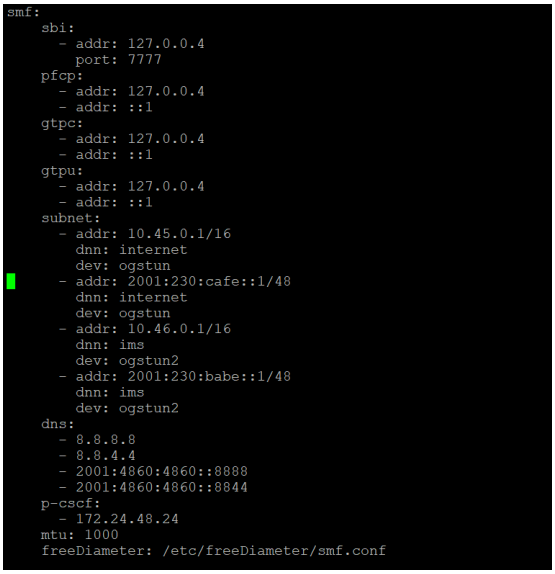
Essa configuração do PCRF é necessária para o estabelecimento da interface Rx. Foi necessário também configurar as APNs do VoLTE nos arquivos de configuração do smf e upf, no diretório /etc/open5gs:

Figura 48 – Configuração de apn para o UPF



Fonte: Autoria Própria.

Figura 49 - Configuração de apn para o SMF



Fonte: Autoria Própria

No smf.yaml foi adicionado também o IP do P-CSCF, que no caso se trata do IP da própria máquina. Ao adicionar o IP do P-CSCF no campo mostrado na Figura 49, será estabelecida a interface Gm. Com essas configurações, a conexão entre EPC e core IMS pode ser estabelecida, sendo necessário somente a criação dos túneis para acesso do usuário.

Como a configuração das rotas criadas são perdidas ao reiniciar o sistema, foi criado um script que deve ser executado antes de reiniciar os serviços do Open5gs e Kamailio. O script em questão está ilustrado na Figura 50:

Figura 50 - Script para a criação das rotas

```
#!/bin/bash
sudo sysctl -w net.ipv4.ip_forward=1
sudo sysctl -w net.ipv6.conf.all.forwarding=1
sudo iptables -t nat -A POSTROUTING -s 10.45.0.1/16 ! -o ogstun -j MASQUERADE
sudo ip6tables -t nat -A POSTROUTING -s 2001:230:cafe::/48 ! -o ogstun -j MASQUERADE
sudo iptables -t nat -A POSTROUTING -s 10.46.0.1/16 ! -o ogstun2 -j MASQUERADE
sudo ip6tables -t nat -A POSTROUTING -s 2001:230:babe::/48 ! -o ogstun2 -j MASQUERADE
sudo iptables -L -t nat
sudo iptables -L
```

Fonte: Autoria Própria

A Figura 51 apresenta a criação das rotas ao executar o script acima:

Figura 51 - Rotas criadas

```
net.ipv4.ip_forward = 1
net.ipv6.conf.all.forwarding = 1
Chain PREROUTING (policy ACCEPT)
target     prot opt source               destination

Chain INPUT (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination

Chain POSTROUTING (policy ACCEPT)
target     prot opt source               destination
MASQUERADE  all  --  10.45.0.0/16         anywhere
MASQUERADE  all  --  10.46.0.0/16         anywhere
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
rtpengine  udp  --  anywhere             anywhere

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination

Chain rtpengine (1 references)
target     prot opt source               destination
RTPENGINE  udp  --  anywhere             anywhere            RTPENGINE id:0
```

Fonte: Autoria Própria

APÊNDICE C – IMPLEMENTAÇÃO E CONFIGURAÇÃO DO KAMAILIO

Para o caso Kamailio não existe uma instalação simplificada por pacotes, dessa forma primeiramente foi necessário instalar as dependências a seguir (OPEN5GS, 2022):

*$ apt update && apt upgrade -y && apt install -y mysql-server tcpdump screen ntp ntpdate git-core dkms gcc flex bison libmysqlclient-dev make *
*libssl-dev libcurl4-openssl-dev libxml2-dev libpcre3-dev bash-completion g++ autoconf rtpproxy ipsec-tools libmnl-dev libsctp-dev libradcli-dev *
libradcli4

Alguns dos pacotes listados no guia (OPEN5GS, 2022), não foram possíveis de serem instalados via gerenciador de pacotes, isso se deve ao fato do ipsec-tools ter sido descontinuado para o Ubuntu 20.04, a alternativa para contornar esse problema foi instalar o pacote de maneira direta. O pacote foi baixado pelo link: http://archive.ubuntu.com/ubuntu/pool/universe/i/ipsec-tools/ipsec-tools_0.8.2+20140711-10build1_amd64.deb através do comando wget. Depois instalou-se o mesmo pelo comando:

$ sudo dpkg -i ipsec-tools_0.8.2+20140711-10build1_amd64.deb

Ao executar esse comando, é importante se atentar a arquitetura de Hardware presente em sua máquina (amd ou arm), para selecionar o pacote correto. Dessa forma, foi gerada a seguinte resposta para a instalação e verificação da presença do pacote ipsec-tools:

Figura 52 - Instalação do ipsec-tools

```
root@kamailio:/home/kamailio# sudo dpkg -i ipsec-tools_0.8.2+20140711-10build1_amd64.deb
Selecting previously unselected package ipsec-tools.
(Reading database ... 202417 files and directories currently installed.)
Preparing to unpack ipsec-tools_0.8.2+20140711-10build1_amd64.deb ...
Unpacking ipsec-tools (1:0.8.2+20140711-10build1) ...
Setting up ipsec-tools (1:0.8.2+20140711-10build1) ...
update-rc.d: warning: start and stop actions are no longer supported; falling back to defaults
Processing triggers for systemd (245.4-4ubuntu3.13) ...
Processing triggers for man-db (2.9.1-1) ...
root@kamailio:/home/kamailio# apt-get install ipsec-tools
Reading package lists... Done
Building dependency tree
Reading state information... Done
ipsec-tools is already the newest version (1:0.8.2+20140711-10build1).
The following package was automatically installed and is no longer required:
  libllvm11
Use 'sudo apt autoremove' to remove it.
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
```

Fonte: Autoria Própria

Com as dependências instaladas, bastou executar o clone do repositório e configurar o software para a 5.3:

$ mkdir -p /usr/local/src/

$ cd /usr/local/src/

$ git clone https://github.com/herlesupreeth/kamailio

$ cd kamailio

$ git checkout -b 5.3 origin/5.3

O próximo passo foi criar o arquivo onde as configurações e o software do Kamailio seriam implementados, além de configurar os módulos para o funcionamento do core IMS:

$ cd /usr/local/src/kamailio

$ make cfg

Agora basta trocar os dados do arquivo modules.lst, encontrado no diretório /usr/local/src/kamailio/src, para os sugeridos em (OPEN5GS, 2022) abaixo na Figura 53 tem-se o arquivo em questão:

Figura 53 - Configuração do arquivo modules.lst

```
# this file is autogenerated by make modules-cfg

# the list of sub-directories with modules
modules_dirs:=modules

# the list of module groups to compile
cfg_group_include=

# the list of extra modules to compile
include_modules= cdp cdp_avp db_mysql dialplan ims_auth ims_charging ims_dialog ims_diameter_server ims_icscf ims_ipsec_pcscf

# the list of static modules
static_modules=

# the list of modules to skip from compile list
skip_modules=

# the list of modules to exclude from compile list
exclude_modules= acc_json acc_radius app_java app_lua app_lua_sr app_mono app_perl app_python app_python3 app_ruby auth_epheme

modules_all= $(filter-out modules/CVS,$(wildcard modules/*))
modules_noinc= $(filter-out $(addprefix modules/, $(exclude_modules) $(static_modules)), $(modules_all))
modules= $(filter-out $(modules_noinc), $(addprefix modules/, $(include_modules) )) $(modules_noinc)
modules_configured:=1
```

Fonte: Autoria Própria

Com essas configurações foi possível compilar o software Kamailio, para tal bastou executar:

$ cd /usr/local/src/kamailio

$ export RADCLI=1

$ make Q=0 all | tee make_all.txt

$ make install | tee make_install.txt

$ ldconfig

Após a compilação do Kamailio e a sua instalação, foi necessário criar o banco de dados para cada um dos serviços do core IMS. Para a senha do banco basta pressionar a tecla Enter ou fornecer a senha configurada para ele:

$ cd /usr/local/src/kamailio/utils/kamctl/mysql

Para o P-CSCF:

$ mysql -u root -p pcscf < standard-create.sql

$ mysql -u root -p pcscf < presence-create.sql

$ mysql -u root -p pcscf < ims_usrloc_pcscf-create.sql

$ mysql -u root -p pcscf < ims_dialog-create.sql

Para o S-CSCF:

$ mysql -u root -p scscf < standard-create.sql

$ mysql -u root -p scscf < presence-create.sql

$ mysql -u root -p scscf < ims_usrloc_scscf-create.sql

$ mysql -u root -p scscf < ims_dialog-create.sql

$ mysql -u root -p scscf < ims_charging-create.sql

Para o I-CSCF:

$ cd /usr/local/src/kamailio/misc/examples/ims/icscf

$ mysql -u root -p icscf < icscf.sql

Agora basta criar os usuários para acessar os bancos de dados e garantir as permissões necessárias:

<mysql> CREATE USER 'pcscf'@'localhost' identified by 'heslo';

<mysql> CREATE USER 'icscf'@'localhost' identified by 'heslo';

<mysql> CREATE USER 'scscf'@'localhost' identified by 'heslo';

<mysql> CREATE USER 'provisioning'@'localhost' identified by 'provi';

<mysql> GRANT ALL PRIVILEGES ON pcscf. to pcscf@localhost;*

<mysql> GRANT ALL PRIVILEGES ON scscf. to scscf@localhost;*

<mysql> GRANT ALL PRIVILEGES ON icscf. to icscf@localhost;*

<mysql> GRANT ALL PRIVILEGES ON icscf. to provisioning@localhost;*

<mysql> CREATE USER 'pcscf'@'%' identified by 'heslo';

<mysql> CREATE USER 'icscf'@'%' identified by 'heslo';

<mysql> CREATE USER 'scscf'@'%' identified by 'heslo';

<mysql> CREATE USER 'provisioning'@'%' identified by 'provi';

<mysql> GRANT ALL PRIVILEGES ON pcscf. TO 'pcscf'@'%';*

<mysql> GRANT ALL PRIVILEGES ON scscf. TO 'scscf'@'%';*

<mysql> GRANT ALL PRIVILEGES ON icscf. TO 'icscf'@'%';*

<mysql> GRANT ALL PRIVILEGES ON icscf. TO 'provisioning'@'%';*

<mysql> FLUSH PRIVILEGES;

Após a inserção dos privilégios de usuário, foi agregado ao banco de dados do I-CSCF o domínio escolhido para ser trabalhado:

<mysql> use icscf;

<mysql> INSERT INTO `nds_trusted_domains` VALUES (1,'ims.mnc001.mcc001.3gppnetwork.org');

<mysql> INSERT INTO `s_cscf` VALUES (1,'First and only S-CSCF','sip:scscf.ims.mnc001.mcc001.3gppnetwork.org:6060');

<mysql> INSERT INTO `s_cscf_capabilities` VALUES (1,1,0),(2,1,1);

É importante salientar que esse domínio é somente para utilização interna, ou seja, não se trata de um domínio público que pode ser acessado externamente, mas sim um domínio para uso dos serviços presentes na máquina em que estão hospedados. Primeiramente foi necessário instalar o pacote bind9:

$ cd /etc

$ apt install -y bind9

Para configurar o domínio, primeiramente foi necessário acessar o diretório onde as configurações de rede ficarão armazenadas /etc/bind, após isso foi criado os arquivos ims.mnc001.mcc001.3gppnetwork.org e epc.mnc001.mcc001.3gppnetwork.org, para suprir os domínios do IMS e EPC respectivamente:

Figura 54 - Arquivo de configuração para o domínio IMS

```
$ORIGIN ims.mnc001.mcc001.3gppnetwork.org.
$TTL 1W
@                       1D IN SOA       localhost. root.localhost. (
                                        1               ; serial
                                        3H              ; refresh
                                        15M             ; retry
                                        1W              ; expiry
                                        1D )            ; minimum

                        1D IN NS        ns
ns                      1D IN A         172.24.48.50

pcscf                   1D IN A         172.24.48.50
_sip._udp.pcscf         1D SRV 0 0 5060 pcscf
_sip._tcp.pcscf         1D SRV 0 0 5060 pcscf

icscf                   1D IN A         172.24.48.50
_sip._udp               1D SRV 0 0 4060 icscf
_sip._tcp               1D SRV 0 0 4060 icscf

scscf                   1D IN A         172.24.48.50
_sip._udp.scscf         1D SRV 0 0 6060 scscf
_sip._tcp.scscf         1D SRV 0 0 6060 scscf

hss                     1D IN A         172.24.48.50
```

Fonte: Autoria Própria

O IP configurado corresponde ao mesmo da máquina sendo configurada. Agora basta criar outro domínio para o EPC:

Figura 55 - Arquivo de Configuração para o domínio EPC

```
$ORIGIN epc.mnc001.mcc001.3gppnetwork.org.
$TTL 1W
@                       1D IN SOA       localhost. root.localhost. (
                                        1               ; serial
                                        3H              ; refresh
                                        15M             ; retry
                                        1W              ; expiry
                                        1D )            ; minimum

                        1D IN NS        epcns
epcns                   1D IN A         172.24.48.50

pcrf                    1D IN A         127.0.0.9
```

Fonte: Autoria Própria

Após a criação desses arquivos de configuração dos domínios, foi necessário editar o arquivo /etc/bind/named.conf.local da seguinte forma:

Figura 56 - Configuração do arquivo named.conf.local

```
// Do any local configuration here
// Consider adding the 1918 zones here, if they are not used in your
// organization
//include "/etc/bind/zones.rfc1918";
zone "ims.mnc001.mcc001.3gppnetwork.org" {
        type master;
        file "/etc/bind/ims.mnc001.mcc001.3gppnetwork.org";
};
zone "epc.mnc001.mcc001.3gppnetwork.org" {
        type master;
        file "/etc/bind/epc.mnc001.mcc001.3gppnetwork.org";
};
```

Fonte: Autoria Própria

Foi necessário também editar o arquivo /etc/bind/ named.conf.options da seguinte forma:

Figura 57 - Configuração do arquivo named.conf.options

```
options {
        directory "/var/cache/bind";

        // If there is a firewall between you and nameservers you want
        // to talk to, you may need to fix the firewall to allow multiple
        // ports to talk.  See http://www.kb.cert.org/vuls/id/800113

        // If your ISP provided one or more IP addresses for stable
        // nameservers, you probably want to use them as forwarders.
        // Uncomment the following block, and insert the addresses replacing
        // the all-0's placeholder.

        //forwarders {
                // Put here the IP address of other DNS server which could be used if name cannot be resolved with DNS se
                //;
        //};

        //========================================================================
        // If BIND logs error messages about the root key being expired,
        // you will need to update your keys.  See https://www.isc.org/bind-keys
        //========================================================================
        dnssec-validation no;
        allow-query { any; };

        auth-nxdomain no;    # conform to RFC1035
        //listen-on-v6 { any; };
};
```

Fonte: Autoria Própria

Para efetivar as mudanças supracitadas, foi executado o seguinte comando:

$ systemctl restart bind9

Para garantir que as configurações de domínio se mantenham, foi necessário realizar uma configuração de IP estático via netplan. Para isso foi editado o arquivo */etc/neplan/01-network-manager-all.yaml*, sendo realizada a seguinte configuração:

Figura 58 - Arquivo de configuração netplan

```
network:
  version: 2
  renderer: networkd
  ethernets:
    wlp2s0:
      addresses:
        - 172.24.48.50/24
      gateway4: 172.24.50.254
      nameservers:
          search: [ims.mnc001.mcc001.3gppnetwork.org,epc.mnc001.mcc001.3gppnetwork.org]
          addresses: 172.24.48.50
```

Fonte: Autoria Própria

Com o arquivo configurado bastou aplicar essas configurações e resetar os serviços do resolv.conf:

$ netplan apply

$ ln -sf /run/systemd/resolve/resolv.conf /etc/resolv.conf

$ systemctl restart systemd-resolved.service

Com isso foi finalizada a configuração dos domínios internos para o ambiente. O próximo passo foi criar os arquivos de configuração para o Kamailio IMS. Foram utilizados os arquivos de configuração (HERLESUPREETH, 2020) apresentados no guia (OPEN5GS, 2022), sendo eles modificados de acordo com o ambiente em questão, dessa forma:

$ cd ~ && git clone https://github.com/herlesupreeth/Kamailio_IMS_Config

$ cd Kamailio_IMS_Config

$ cp -r kamailio_icscf /etc

$ cp -r kamailio_pcscf /etc

$ cp -r kamailio_scscf /etc

O próximo passo foi a criação de scripts para execução de cada seviço CSCF do Kamailio, abaixo está representado esse arquivo para o P-CSCF:

Figura 59 - Script de inicialização do P-CSCF

```
#!/bin/bash
mkdir -p /var/run/kamailio_pcscf
kamailio -f /etc/kamailio_pcscf/kamailio_pcscf.cfg -P /kamailio_pcscf.pid -DD -E
```

Fonte: Autoria Própria

Da mesma forma como foi feito para a leitura dos logs dos serviços do Open5gs, bastou modificar o nome do arquivo para o nome do serviço correto, e garantir a permissão de execução, tendo assim scripts para a execução de cada um dos serviços. O próximo passo foi alterar os IPs dentro dos arquivos de configuração de cada serviço, primeiramente para o P-CSCF:

$ cd /etc/kamailio_pcscf

Para esse serviço foi necessário configurar os arquivos /etc/kamailio_pcscf/pcscf.cfg e /etc/kamailio_pcscf/pcscf.xml. Para verificar os campos dos IPs a serem modificados bastou executar o comando grep -r “10.4.128.21”, este IP vem como default e o novo valor foi alterado para corresponder ao IP principal da máquina em questão (172.24.48.50). Além do IP, deve-se modificar o “alias” conforme o domínio definido anteriormente. Dessa forma, na Figura 60 estão os IPs a serem modificados, e nas figuras 61 e 62 os arquivos após as modificações:

Figura 60 - IPs a serem modificados

```
root@kamailio:/etc/kamailio_pcscf# grep -r "10.4"
pcscf.cfg:listen=udp:10.4.128.21:5060
pcscf.cfg:#listen=udp:10.4.128.21:5060 advertise 172.24.15.30:5060
pcscf.cfg:listen=tcp:10.4.128.21:5060
pcscf.cfg:#listen=tcp:10.4.128.21:5060 advertise 172.24.15.30:5060
pcscf.cfg:#!define IPSEC_LISTEN_ADDR "10.4.128.21"
pcscf.cfg:#!define RX_AF_SIGNALING_IP "10.4.128.21"
pcscf.xml:      <Acceptor port="3871" bind="10.4.128.21"/>
```

Fonte: Autoria Própria

Figura 61 - Arquivo pcscf.conf modificado

```
1  # IP-Adress for incoming SIP-Traffic, in the following format:
2
3  # SIP / UDP
4  listen=udp:172.24.48.50:5060
5  # Uncomment the below line only when UE is behind double NAT (e.g. VoIP calling over WiFi/ CN behind a NAT)
6  #listen=udp:10.4.128.21:5060 advertise 172.24.15.30:5060
7  # SIP / TCP
8  listen=tcp:172.24.48.50:5060
9  # Uncomment the below line only when UE is behind double NAT (e.g. VoIP calling over WiFi/ CN behind a NAT)
10 #listen=tcp:10.4.128.21:5060 advertise 172.24.15.30:5060
11 # SIP / TCP/TLS
12 #listen=tls:11.22.33.44:5061
13
14 # IPSEC / UDP
15 #!define IPSEC_LISTEN_ADDR "172.24.48.50"
16 #!define IPSEC_CLIENT_PORT 5100
17 #!define IPSEC_SERVER_PORT 6100
18 #!define IPSEC_MAX_CONN 10
19
20 # IP used in Rx_AAR_Register - IP of this P-CSCF, to be used in the flow for the AF-signaling
21 #!define RX_AF_SIGNALING_IP "172.24.48.50"
22 # Uncomment the below line only when UE is behind double NAT (e.g. VoIP calling over WiFi/ CN behind a NAT)
23 ##!define RX_AF_SIGNALING_IP "172.24.15.30"
```

Fonte: Autoria Própria

Figura 62 - Arquivo pcscf.xml modificado

```
<?xml version="1.0" encoding="UTF-8"?>
<DiameterPeer
        FQDN="pcscf.ims.mnc001.mcc001.3gppnetwork.org"
        Realm="ims.mnc001.mcc001.3gppnetwork.org"
        Vendor_Id="10415"
        Product_Name="CDiameterPeer"
        AcceptUnknownPeers="1"
        DropUnknownOnDisconnect="1"
        Tc="30"
        Workers="4"
        QueueLength="8"
        TransactionTimeout="5"
        SessionsHashSize="128"
        DefaultAuthSessionTimeout="3600"
        MaxAuthSessionTimeout="3600"
>
        <Peer FQDN="pcrf.epc.mnc001.mcc001.3gppnetwork.org" Realm="epc.mnc001.mcc001.3gppnetwork.org" port="3868"/>

        <Acceptor port="3871" bind="172.24.48.50"/>

        <Auth id="16777236" vendor="10415"/> <!-- 3GPP Rx -->
        <Auth id="16777236" vendor="0"/> <!-- 3GPP Rx -->

        <DefaultRoute FQDN="pcrf.epc.mnc001.mcc001.3gppnetwork.org" metric="10"/>

</DiameterPeer>
```

Fonte: Autoria Própria

Para os outros serviços do core IMS (S-CSCF e I-CSCF), foram realizados os mesmos procedimentos. Dessa forma as configurações para os serviços principais do core IMS estão finalizadas.

APÊNDICE D – IMPLEMENTAÇÃO E CONFIGURAÇÃO DO RTPENGINE

O RTPengine é um software que providencia as funcionalidades de RTP ao sistema, sendo utilizado como um software auxiliar ao Kamailio core IMS (SIPWISE, 2020). Primeiramente foi realizada a checagem das dependências necessárias para a construir a ferramenta (OPEN5GS, 2022):

$ export DEB_BUILD_PROFILES="pkg.ngcp-rtpengine.nobcg729"

$ apt install dpkg-dev

$ git clone https://github.com/sipwise/rtpengine

$ cd rtpengine && git checkout mr9.4

$ dpkg-checkbuilddeps

Diversos pacotes são instalados nesse comando, porém dois deles não possuem candidatos para instalação:

E: Package 'iptables-dev' has no installation candidate

E: Package 'libtirpc1' has no installation candidate

Dessa forma foi executado o comando acima sem esses dois pacotes, e posteriormente eles foram instalados pelo mesmo método utilizado com o ipsec-tools. Dessa forma para o iptables-dev:

$ wget http://archive.ubuntu.com/ubuntu/pool/main/i/iptables/iptables-dev_1.6.1-2ubuntu2_amd64.deb

$ apt-get install libxtables-dev

$ sudo dpkg -i iptables-dev_1.6.1-2ubuntu2_amd64.deb

Agora para o libtirpc1, o pacote libtirpc-common substitui o mesmo, não necessitando da instalação. Execute o comando abaixo para verificar se as dependências foram satisfeitas:

$ dpkg-checkbuilddeps

O mesmo não deve retornar nenhuma dependência. Foram executados os seguintes comandos para alocação e build do RTPEngine:

$ dpkg-buildpackage -uc -us

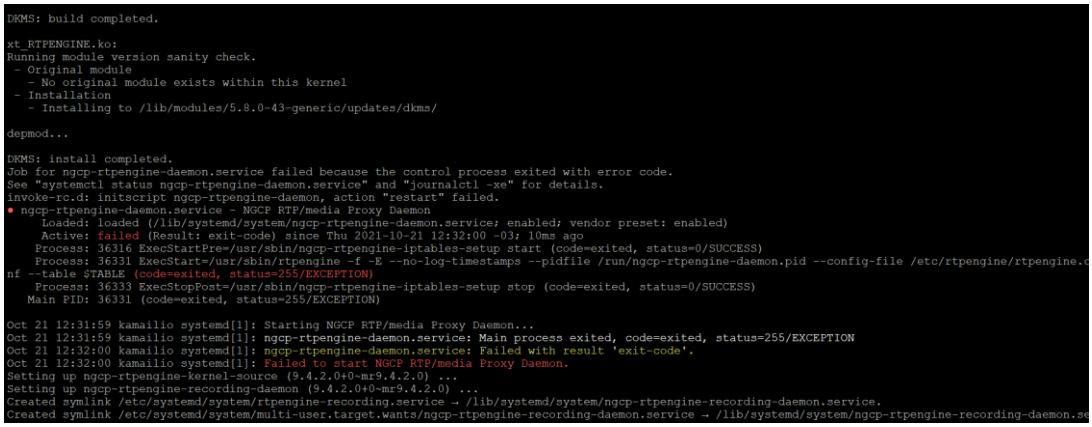
$ cd ..

*$ dpkg -i *.deb*

$ cp /etc/rtpengine/rtpengine.sample.conf /etc/rtpengine/rtpengine.conf

Ao executar os comandos acima, a ferramenta irá finalizar os processos com alguns erros, porém esses erros estão ligados a configuração dela, sendo importante se atentar somente se a build foi completada com sucesso:

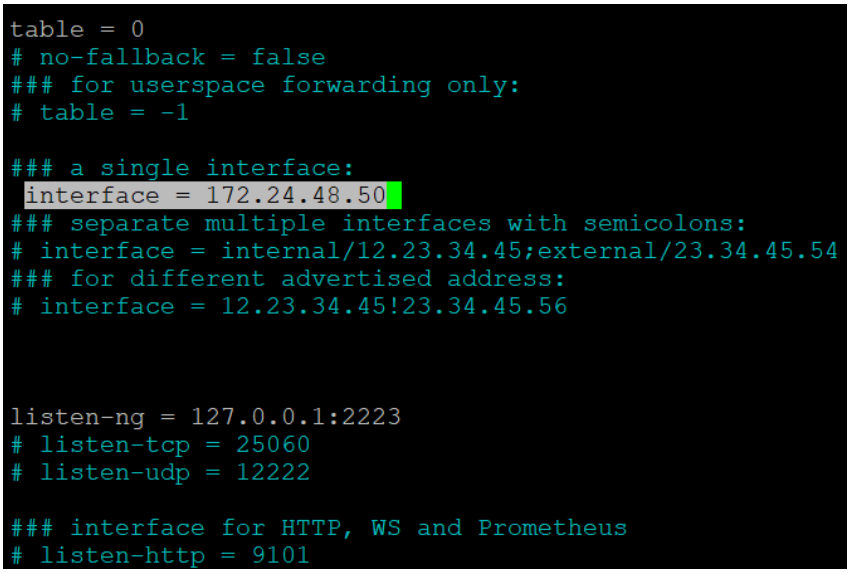
Figura 63 - Finalização da build do RTPengine



Fonte: Autoria Própria

Para que o serviço do RTPengine funcione corretamente é necessário configurar a interface correta para a conexão do mesmo, dessa forma na Figura 64 está ilustrada a configuração realizada:

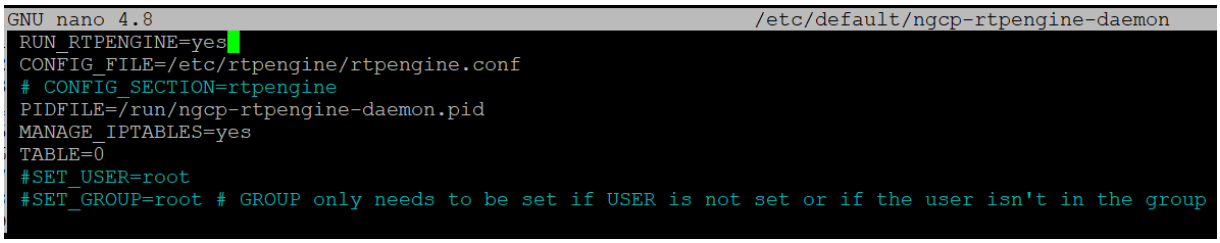
Figura 64 - Configuração da interface do RTPengine



Fonte: Autoria Própria

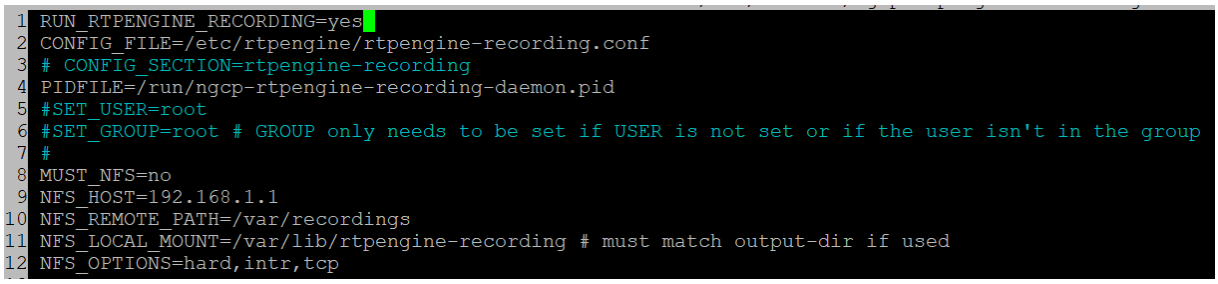
O arquivo da Figura 64 se encontra no diretório /etc/rtpengine/rtpengine.conf. Foi necessário também, editar os arquivos /etc/default/ngcp-rtpengine-daemon e /etc/default/ngcp-rtpengine-recording-daemon, como mostram as figuras 65 e 66 respectivamente:

Figura 65 - Arquivo ngcp-rtpengine-daemon configurado



Fonte: Autoria Própria

Figura 66 - Arquivo ngcp-rtpengine-recording-daemon configurado



Fonte: Autoria Própria

Com essas configurações realizadas, pode-se inicializar os serviços do RTPengine, executando os comandos a seguir:

$ cp /etc/rtpengine/rtpengine-recording.sample.conf /etc/rtpengine/rtpengine-recording.conf

$ mkdir /var/spool/rtpengine

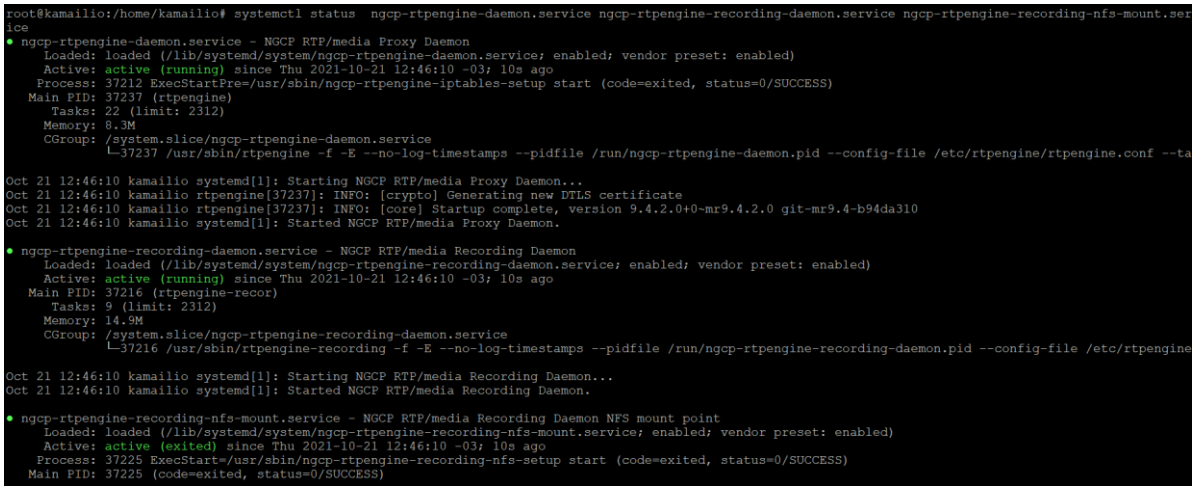
$ systemctl enable ngcp-rtpengine-daemon.service ngcp-rtpengine-recording-daemon.service ngcp-rtpengine-recording-nfs-mount.service

$ systemctl restart ngcp-rtpengine-daemon.service ngcp-rtpengine-recording-daemon.service ngcp-rtpengine-recording-nfs-mount.service

$ systemctl status ngcp-rtpengine-daemon.service ngcp-rtpengine-recording-daemon.service ngcp-rtpengine-recording-nfs-mount.service

A Figura 67 demonstra o funcionamento da ferramenta:

Figura 67 - Funcionamento do RTPengine



Fonte: Autoria Própria

APÊNDICE E – IMPLEMENTAÇÃO E CONFIGURAÇÃO DO FHOSS

Assim como o RTPengine o FHoSS se trata de uma aplicação utilizada de forma a auxiliar ao Kamailio, se tratando de um serviço de HSS usado para o cadastramento dos índices IMPI e IMPU dos usuários da rede (HERLESUPREETH, 2022). O software em questão é baseado no JDK, sendo necessário instalar o mesmo no sistema, para isso basta realizar o download presente em (ORACLE, 2022). Após a realização do download foi criado um diretório específico para o JDK e executado a sua instalação, através dos comandos a seguir:

$ mkdir -p /usr/lib/jvm/

$ tar -zxf jdk-7u79-linux-x64.tar.gz -C /usr/lib/jvm/

$ update-alternatives --install /usr/bin/java java /usr/lib/jvm/jdk1.7.0_79/bin/java 100

$ update-alternatives --install /usr/bin/javac javac /usr/lib/jvm/jdk1.7.0_79/bin/javac 100

O próximo passo foi realizar a instalação do apache-ant, para tal foi executado os comandos a seguir:

$ wget http://archive.apache.org/dist/ant/binaries/apache-ant-1.9.14-bin.tar.gz

$ tar xvfvz apache-ant-1.9.14-bin.tar.gz

$ mv apache-ant-1.9.14 /usr/local/

$ sh -c 'echo ANT_HOME=/usr/local/ >> /etc/environment'

$ ln -s /usr/local/apache-ant-1.9.14/bin/ant /usr/bin/ant

Após a instalação desses dois softwares, foi realizada a instalação do FHoSS, primeiramente foram executados os comandos abaixo:

$ mkdir /opt/OpenIMSCore

$ cd /opt/OpenIMSCore

$ git clone https://github.com/herlesupreeth/FHoSS

$ cd FHoSS

$ export JAVA_HOME="/usr/lib/jvm/jdk1.7.0_79"

$ export CLASSPATH="/usr/lib/jvm/jdk1.7.0_79/jre/lib/"

$ ant compile deploy | tee ant_compile_deploy.txt

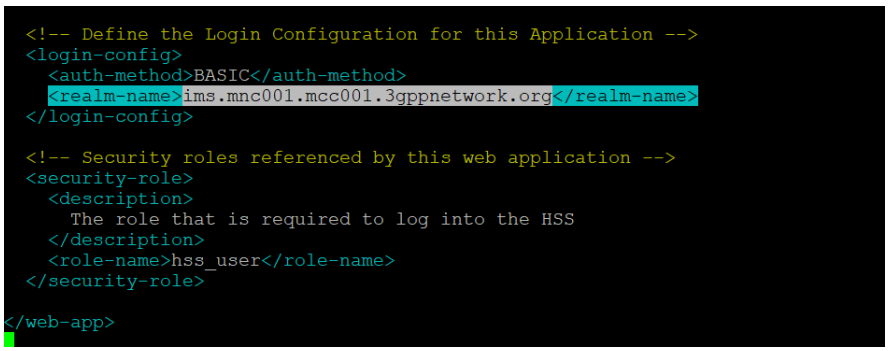
Com isso toda a instalação inicial do software pode ser finalizada. Após esse procedimento foi necessário configurar os parâmetros da aplicação para o funcionamento no cenário em questão, através de um script feito pelos próprios desenvolvedores do arquivo configurator.sh (OPEN5GS, 2022). A configuração para esse script está presente na Figura 68:

Figura 68 - Script configurator.sh

Fonte: Autoria Própria

Esse script deve ser executado no diretório /opt/OpenIMSCore/FHoSS/deploy, onde ao executá-lo será perguntado o domínio e IP que se deseja trabalhar, sendo inseridos o domínio criado para o IMS e o IP da interface padrão da máquina. Esse mesmo script foi executado em outros dois diretórios o /opt/OpenIMSCore/FHoSS/scripts e /opt/OpenIMSCore/FHoSS/config com o mesmo propósito citado anteriormente. Também foi necessário alterar o campo realm-name no arquivo /opt/OpenIMSCore/FHoSS/deploy/webapps/hss.web.console/WEB-INF/web.xml, como mostra a figura 69:

Figura 69 - Alteração no arquivo web.xml



Fonte: Autoria Própria

Foi realizada a mesma alteração no arquivo /opt/OpenIMSCore/FHoSS/src-web/WEB-INF/web.xml. Para a configuração do banco de dados da aplicação é necessário utilizar do arquivo /opt/OpenIMSCore/FHoSS/scripts/hss_db.sql, porém foi necessário excluir a última linha do script por conter comando antigos do mysql. Após a exclusão da linha, bastou executar os comandos abaixo:

$ cd /opt/OpenIMSCore

2$ mysql -u root -p hss_db < FHoSS/scripts/hss_db.sql

3$ mysql -u root -p hss_db < FHoSS/scripts/userdata.sql

Após a finalização da criação do banco, foi criado os usuários e garantido o acesso dos mesmos ao banco:

$ mysql

2<mysql> create user 'hss'@'localhost' identified by 'hss';

3<mysql> create user 'hss'@'%' identified by 'hss';

4<mysql> grant all privileges on hss_db. to hss@localhost;*

5<mysql> grant all privileges on hss_db. to hss@'%';*

6<mysql> FLUSH PRIVILEGES;

Com isso todas as configurações para o FHoSS estão finalizadas, bastando somente criar um script de inicialização do serviço:

Figura 70 - Script para inicialização do FHoSS

```
#!/bin/bash
# -----------------------------------------------------------------
# Include JAR Files
# -----------------------------------------------------------------

cd /opt/OpenIMSCore/FHoSS/deploy
JAVA_HOME="/usr/lib/jvm/jdk1.7.0_79"
CLASSPATH="/usr/lib/jvm/jdk1.7.0_79/jre/lib/"

echo "Building Classpath"
CLASSPATH=$CLASSPATH:log4j.properties:.
for i in lib/*.jar; do CLASSPATH="$i":"$CLASSPATH"; done
echo "Classpath is $CLASSPATH."

# -----------------------------------------------------------------
# Start-up
# -----------------------------------------------------------------

$JAVA_HOME/bin/java -cp $CLASSPATH de.fhg.fokus.hss.main.HSSContainer $1 $2 $3 $4 $5 $6 $7 $8 $9
```

Fonte: Autoria Própria

APÊNDICE F – INSTALAÇÃO DOS SOFTWARES PARA GRAVAÇÃO DE SIM CARDS

Para a instalação do pysim, primeiramente deve-se baixar o repositório (OSMOCOM, 2022):

$ git clone git://git.osmocom.org/pysim.git

Após baixar o repositório é necessário executar os comandos a seguir:

$ apt-get install python3-pyscard python3-serial python3-pip python3-yaml

pip3 install -r requirements.txt

Com isso o software pysim foi instalado com sucesso. A seguir estão os comandos para a instalação do CoIMS:

$ git clone https://github.com/herlesupreeth/CoIMS_Wiki

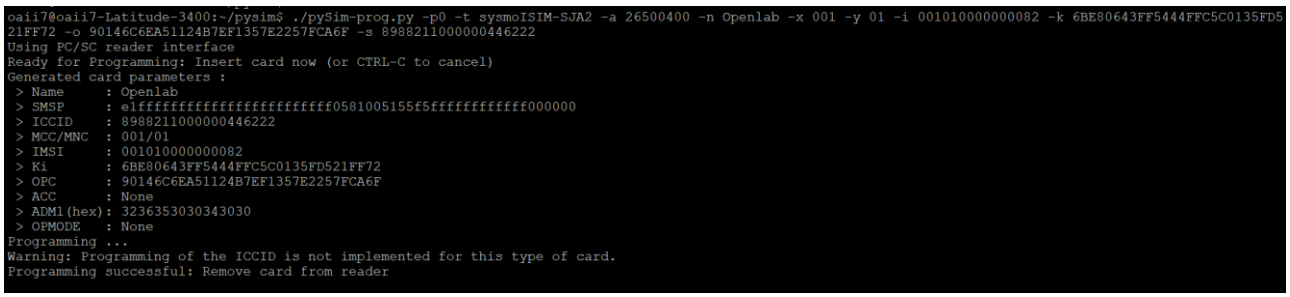
$ cd CoIMS_Wiki

$ alias gp="java -jar $PWD/gp.jar"o do CoIMS:

APÊNDICE G – GRAVAÇÃO DOS SIM CARDS

Para a gravação dos sim cards foi utilizado o leitor de sim cards ilustrado na Figura 18, conectado ao PC em que os softwares de gravação estavam instalados. A fabricante dos SIM cards, forneceu as chaves de administrador dos cartões para executar a customização dos mesmos. Foram gravados 4 SIM cards, com os seguintes dados: IMSI=001010000000082, IMSI=001010000000083, IMSI=001010000000086 e IMSI=001010000000087, todos configurados para uma rede de testes. Na Figura 71 tem-se um exemplo do comando executado para a gravação do sim card:

Figura 71 - Comando para a gravação dos SIM cards



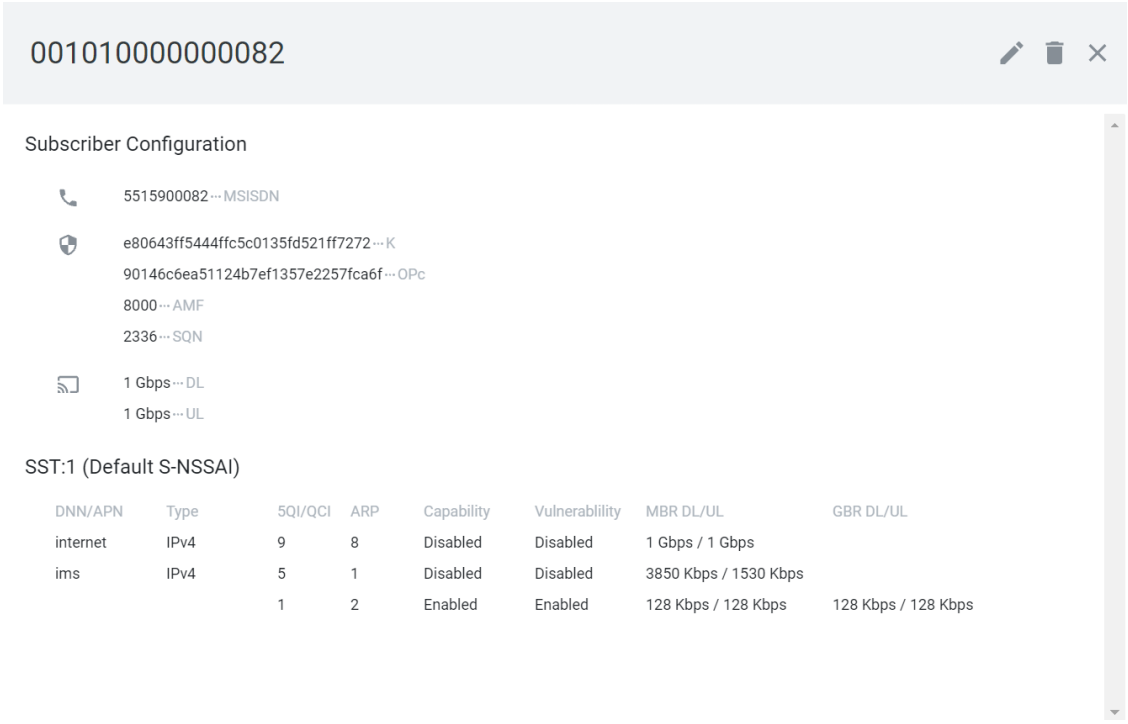
Fonte: Autoria Própria

O CoIMS foi utilizado para desbloquear algumas das funcionalidades do VoLTE, seguindo o guia providenciado no repositório do software (HERLESUPREETH, 2022).

APÊNDICE H – CADASTRAMENTO DOS DADOS DE USUÁRIO: WEBUI e FHOSS

Para que os usuários possam acessar os serviços fornecidos pela rede LTE, é necessário que eles tenham os seus dados gravados no banco de dados HSS. Tanto o WebUI como o FHoSS são softwares com interfaces gráficas que servem como porta de entrada para o desenvolvedor realizar o cadastro dos dados do usuário, sendo expostos respectivamente na porta 3000 e 8080. Na Figura 72 tem-se os dados gravados para o usuário com IMSI=001010000000082 pelo WebUI:

Figura 72 - Dados gravados no WebUI para o IMSI 001010000000082

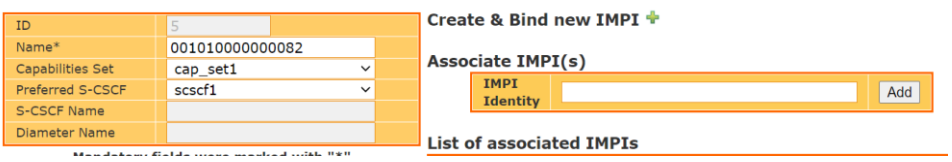


Fonte: Autoria Própria

É possível observar as configurações de IMSI, MSISDN e APN gravadas para o SIM card. Para os outros IMSIs foi realizado o mesmo procedimento.

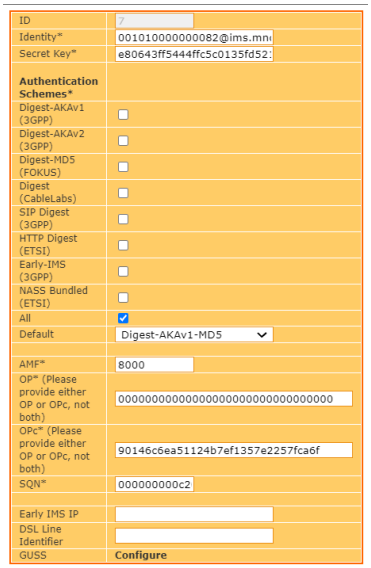
Para o FHoSS foram atribuídas as configurações de IMPI e IMPU dos usuários. A seguir as Figuras 73, 74, 75, 76 e 77 demonstram as configurações necessárias para a gravação desses parâmetros:

Figura 73 - Gravação do IMSI no FHoSS



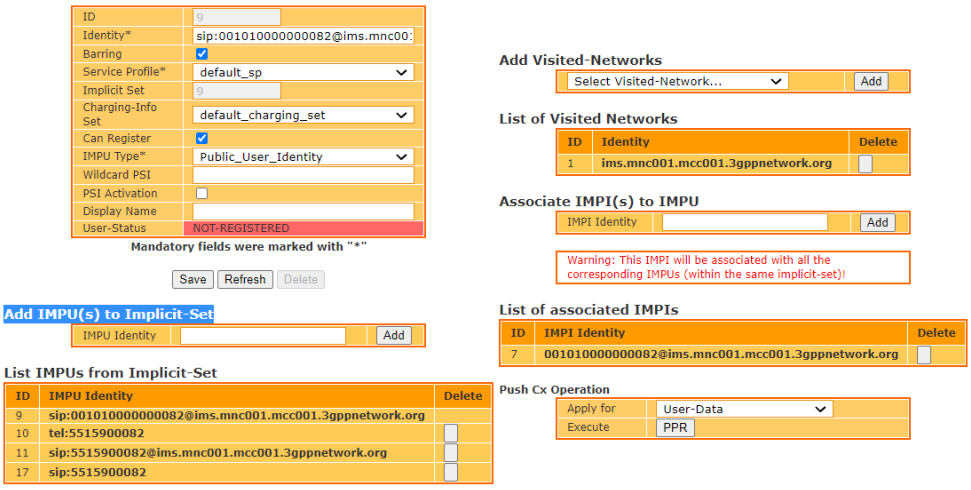
Fonte: Autoria Própria

Figura 74 - Configuração do IMPI no FHoSS



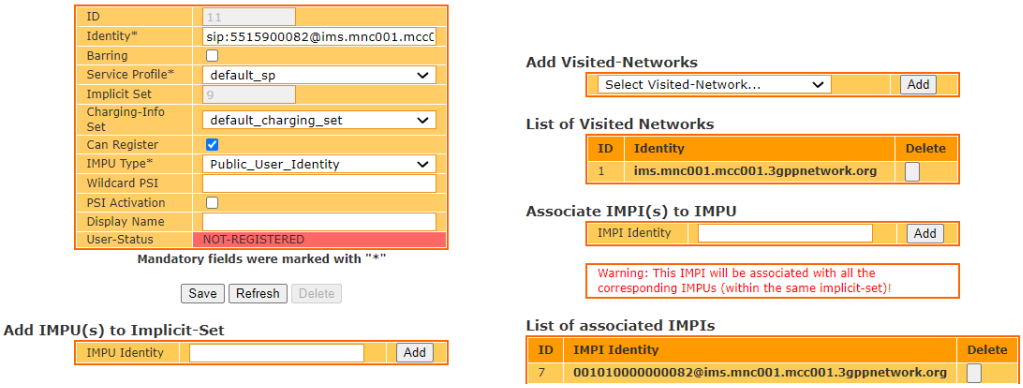
Fonte: Autoria Própria

Figura 75 – IMPU sip:001010000000082@ims.mnc001.mcc001.3gppnetwork.org



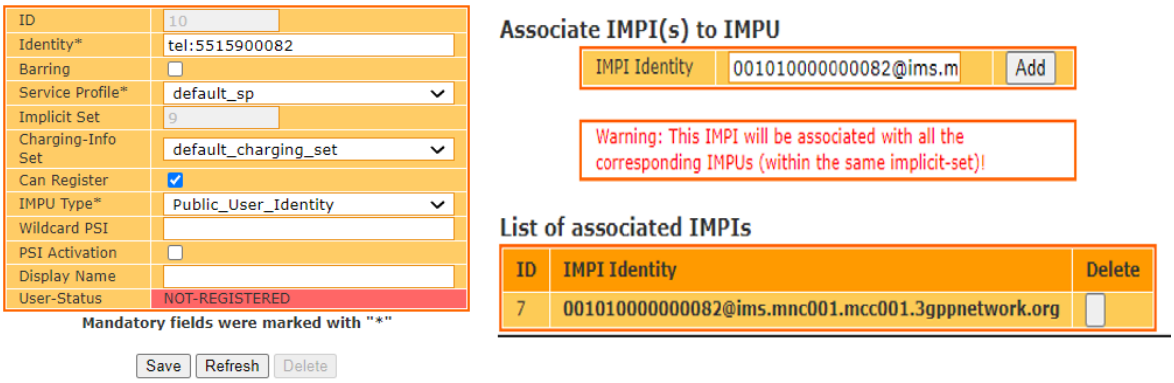
Fonte: Autoria Própria

Figura 76 - IMPU sip:5515900082@ims.mnc001.mcc001.3gppnetwork.org



Fonte: Autoria Própria

Figura 77 - IMPU tel:5515900082



Fonte: Autoria Própria

Para os outros IMSIs foi realizado o mesmo procedimento para a gravação dos dados.

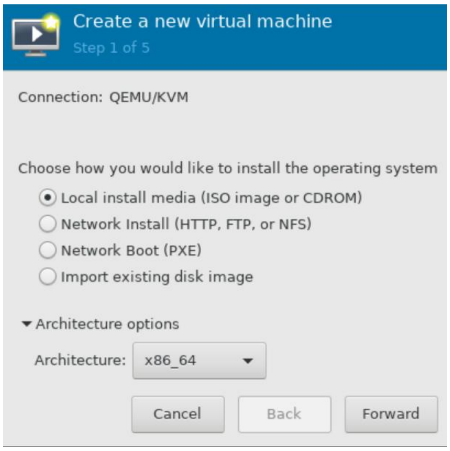
APÊNDICE I – IMPLEMENTAÇÃO UTILIZANDO DOCKER-COMPOSE EM UMA VM

Um outro método de instalação dos softwares para eNodeB, EPC e core IMS é a instalação via Docker e Docker-compose. Ao utilizar essas ferramentas de conteinerização, a implementação dos softwares citados nos apêndices acima pode ser feita de maneira mais simplificada e optimizada. O guia utilizado para a instalação foi o repositório (HERLESUPREETH, 2022) nele encontram-se todos os arquivos necessários para a criação do cenário em questão. Na implementação realizada diretamente na máquina física, foram enfrentadas diversas dificuldades para a sua adequação, e como supracitado, um dos propósitos do Docker é justamente facilitar essa transposição de aplicações para qualquer ambiente que possua o Docker, dessa forma esse foi um dos principais motivos para a realização desse tipo de implementação. Para isolar os testes com o cenário Docker, dos testes do cenário utilizando a máquina física, o ambiente em questão foi criado em uma máquina virtual (VM), criada pelo KVM em um servidor comercial. O KVM é um hypervisor que orquestra e cria máquinas virtuais conforme as configurações e necessidades do desenvolvedor (LINUX-KVM, 2022). Para instalar o KVM basta executar os comandos a seguir:

$ sudo apt -y install libvirt-clients libvirt-daemon qemu qemu-kvm

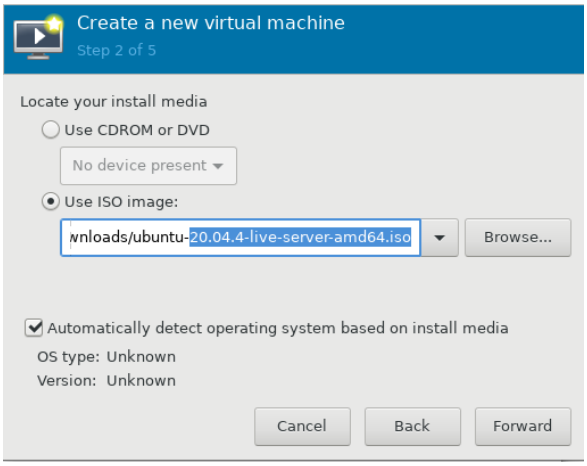
Após a instalação do KVM, foi criada a máquina virtual seguindo o guia (TECMINT, 2022), e utilizando a imagem do Ubuntu 20.04 (UBUNTU, 2022). Para abrir o layout para a criação da VM basta executar *$ sudo virt-manager*. As Figuras 78,79 e 80 representam alguns passos da criação da VM em questão:

Figura 78 - Passo 1 para a criação da VM utilizando o KVM



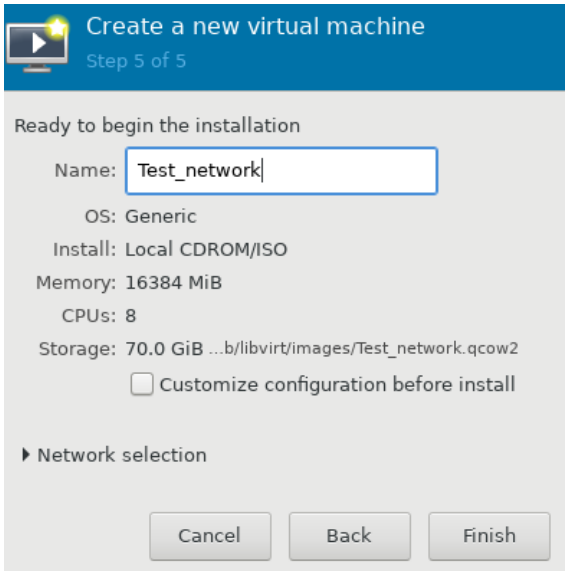
Fonte: Autoria Própria

Figura 79 - Passo 2 para a criação da VM utilizando o KVM



Fonte: Autoria Própria

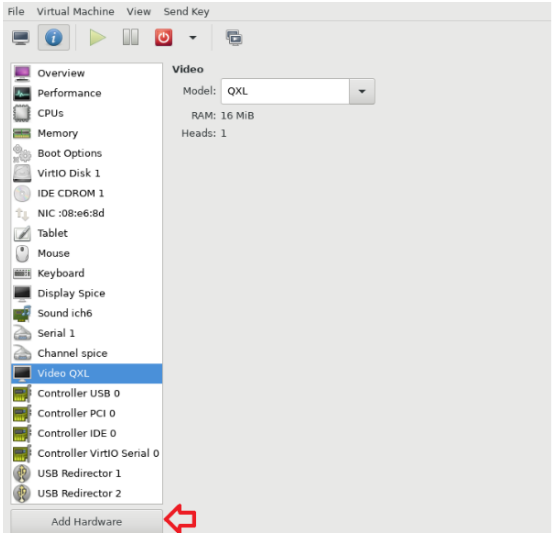
Figura 80 - Passo 3 para a criação da VM utilizando o KVM



Fonte: Autoria Própria

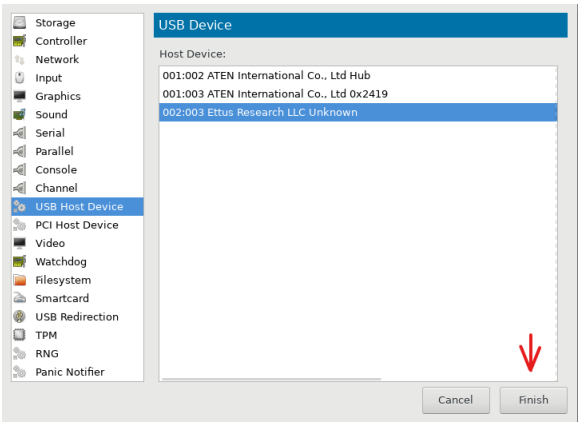
Para os outros passos basta seguir o procedimento padrão para a criação da VM. Outro procedimento necessário para a comunicação da VM com o SDR é adicionar o plugin de USB do SDR à VM pelo KVM. Para tal basta executar os passos demonstrados nas Figuras a seguir:

Figura 81 – Passo 1 para adicionar o plugin do SDR para a VM



Fonte: Autoria Própria

Figura 82 Passo 2 para adicionar o plugin do SDR para a VM



Fonte: Autoria Própria

A VM criada possui as mesmas capacidades de hardware do notebook utilizado no cenário em uma máquina física, a fim de manter certa similaridade entre os ambientes, com isso as configurações da VM estão finalizadas. Após a criação e configuração da VM, é necessário instalar o SSH para o acesso de maneira externa, executando o comando *sudo apt install ssh* pela interface gráfica do virt-manager.

Após verificar o IP da mesma (192.168.120.132) e acessar a máquina, foi necessário instalar as seguintes dependências:

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

$ sudo apt-get update

*$ sudo apt-get install *

*ca-certificates *

*curl *

*gnupg *

lsb-release

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

*$ echo *

*"deb [arch=$(dpkg --print-architecture) 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

$ sudo apt-get update

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

$ sudo apt-get install docker-compose

Com as dependências devidamente instaladas, pode-se partir para a configuração dos softwares em si. Primeiramente foi necessário baixar o repositório, e executar a construção da imagem a ser utilizada para o Open5gs (HERLESUPREETH, 2022).

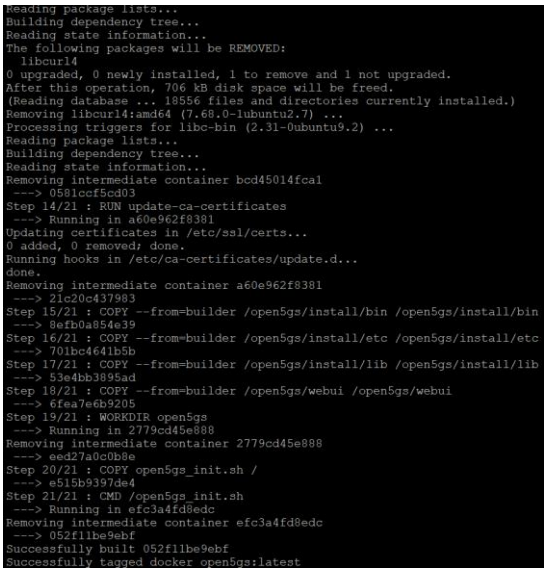
$ git clone https://github.com/herlesupreeth/docker_open5gs

$ cd docker_open5gs/base

$ docker build --no-cache --force-rm -t docker_open5gs .

A Figura 83 demonstra a finalização da construção da imagem do Open5gs:

Figura 83 - Construção da imagem do Open5gs



Fonte: Autoria Própria

O próximo passo foi realizar a construção da imagem para o Kamailio, dessa forma:

$ cd ../ims_base

$ docker build --no-cache --force-rm -t docker_kamailio .

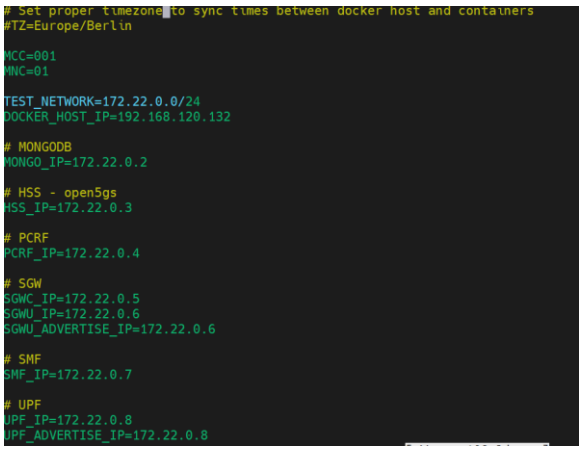
A Figura 84 ilustra a finalização da construção da imagem do Kamailio:

Figura 84 - Construção da imagem do Kamailio

Fonte: Autoria Própria

Com essas imagens construídas, pode-se dar início ao processo de inicialização dos contêineres, executando o Docker-compose. Porém, antes de executar o Docker-compose é necessário modificar o arquivo .env para que o mesmo esteja de acordo com o cenário que deseja ser instalado. Para o teste em questão foi utilizado o mnc/mcc=001/01 que corresponde a uma rede de testes, é necessário também configurar o IP do host, que no caso corresponde ao IP principal da VM (192.168.120.132). Dessa forma o arquivo .env deve ser modificado como mostrado na Figura 85:

Figura 85 - Configuração arquivo .env



Fonte: Autoria Própria

Com isso a instalação via Docker-compose pode ser executada, efetuando os comandos a seguir:

$ cd ..

$ set -a

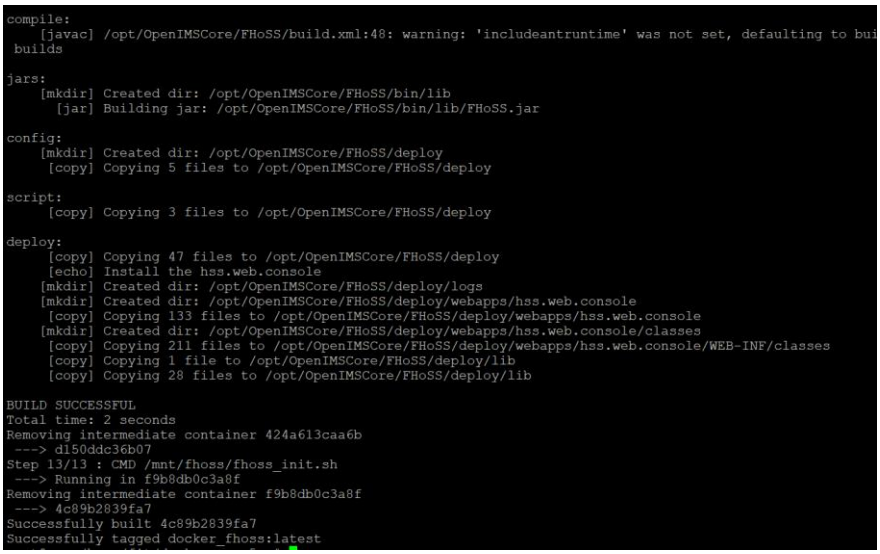
$ source .env

$ docker-compose build --no-cache

$ docker-compose up -d

Com essas simples modificações todas as interfaces e arquivos de configurações que para a implementação em uma máquina física eram modificados manualmente, são devidamente alterados e configurados, até mesmo as interfaces ogstun e ogstun2. Ao finalizar o processo de build, a seguinte mensagem deve aparecer:

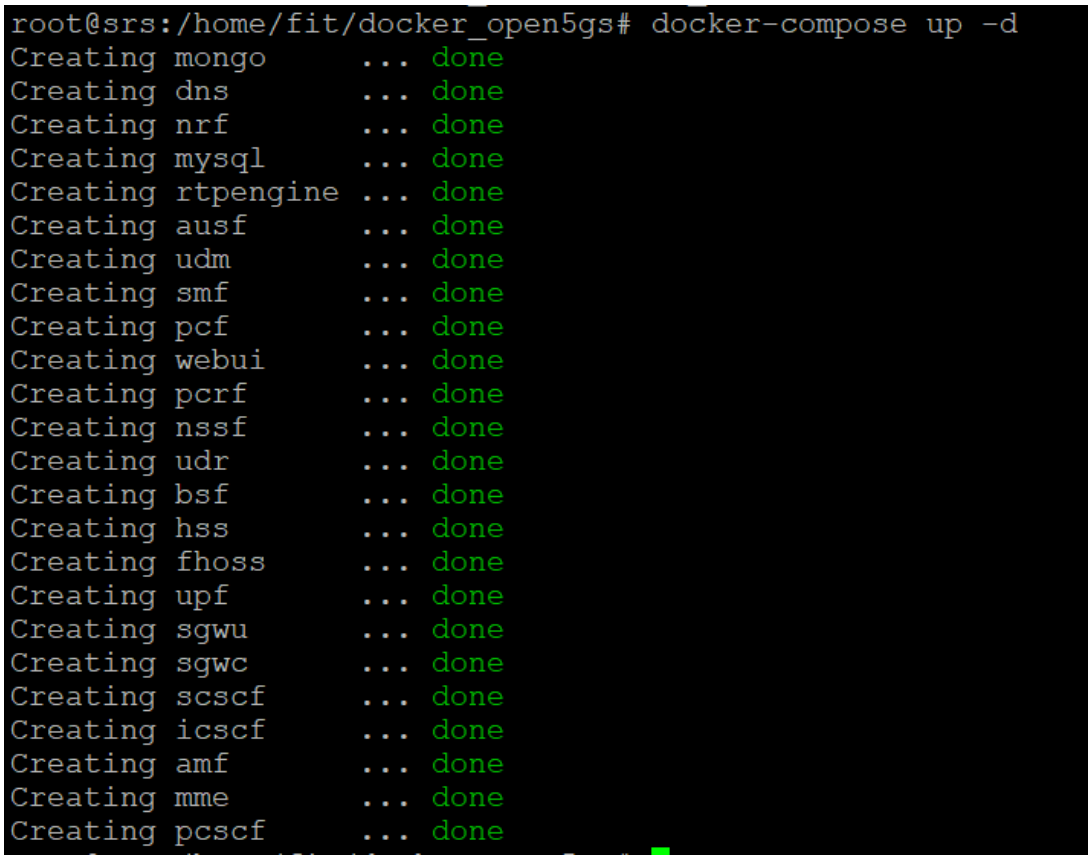
Figura 86 - Execução do docker-compose build –no-cache



Fonte: Autoria Própria

Ao executar o comando *$ docker-compose up -d* deve-se ter a seguinte resposta do sistema:

Figura 87 - Execução do comando docker-compose up -d



Fonte: Autoria Própria

Para o cadastramento dos usuários, bastou seguir os mesmos passos descritos no apêndice H deste documento. Para o eNodeB o procedimento é semelhante ao executado para o Kamailio e Open5gs, com a diferença de que para a execução do contêiner deve-se manter o SDR B210 conectado ao USB 3.0 do servidor, para que o software reconheça o SDR. Caso deseje-se modificar a banda a ser trabalhada, ganho em tx/rx ou qualquer outro parâmetro relacionado ao eNodeB basta modificar o arquivo *srslte/enb.conf*. Para o cenário em questão, nenhuma modificação foi necessária, bastando executar os comandos seguir para a criação da imagem e execução do contêiner do srsENB:

$ docker-compose -f srsenb.yaml build --no-cache

$ docker-compose -f srsenb.yaml up -d && docker attach srsenb