Universidade Federal de Santa Catarina
Centro de Blumenau
Departamento de Engenharia de
Controle, Automação e Computação

Dener Kraus

Computação de borda para indústria utilizando a rede 5G

Blumenau
2021

Dener Kraus

Computação de borda para indústria utilizando a rede 5G

Trabalho de Conclusão de Curso de Graduação em Engenharia de Controle e Automação do Centro de Blumenau da Universidade Federal de Santa Catarina para a obtenção do título de Engenheiro de Controle e Automação.
Orientador: Prof. Dr. Adão Boava.
Coorientador: Eng. Christian Mailer.

Universidade Federal de Santa Catarina
Centro de Blumenau
Departamento de Engenharia de
Controle, Automação e Computação

Blumenau
2021

Ficha de identificação da obra elaborada pelo autor,
através do Programa de Geração Automática da Biblioteca Universitária da UFSC.

Kraus, Dener
Computação de borda para indústria utilizando a rede 5G / Dener Kraus ; orientador, Adão Boava, coorientador, Christian Mailer, 2021.
95 p.

Trabalho de Conclusão de Curso (graduação) - Universidade Federal de Santa Catarina, Campus Blumenau, Graduação em Engenharia de Controle e Automação, Blumenau, 2021.

Inclui referências.

1. Engenharia de Controle e Automação. 2. 5G. 3. Computação de borda. 4. Free5gc Compose e UERAMSIM. 5. Ping, tcpdump, iperf e netperf. I. Boava, Adão. II. Mailer, Christian. III. Universidade Federal de Santa Catarina. Graduação em Engenharia de Controle e Automação. IV. Título.

Dener Kraus

Computação de borda para indústria utilizando a rede 5G

Este Trabalho de Conclusão de Curso foi julgado adequado para obtenção do Título de Engenheiro de Controle e Automação e aprovado em sua forma final pelo Curso de Engenharia de Controle e Automação.

Comissão Examinadora

Documento assinado digitalmente
Adao Boava
Data: 30/09/2021 21:31:44-0300
CPF: 645.640.969-15
Verifique as assinaturas em https://v.ufsc.br

Prof. Dr. Adão Boava.
Orientador
Universidade Federal de Santa Catarina

Documento assinado digitalmente
Carlos Roberto Moratelli
Data: 01/10/2021 08:52:16-0300
CPF: 035.999.369-94
Verifique as assinaturas em https://v.ufsc.br

Prof. Dr. Carlos Roberto Moratelli.
Avaliador
Universidade Federal de Santa Catarina

Documento assinado digitalmente
Ciro Andre Pitz
Data: 01/10/2021 08:14:51-0300
CPF: 055.728.089-38
Verifique as assinaturas em https://v.ufsc.br

Prof. Dr. Ciro André Pitz.
Avaliador
Universidade Federal de Santa Catarina

Blumenau, 1 de outubro de 2021

Dedico este trabalho a todos os que me ajudaram ao longo desta caminhada e aos que se interessarem.

Agradecimentos

Agradeço ao meu pai, mãe, irmã e outros familiares por toda a infraestrutura, ajuda e carinho que me disponibilizaram durante toda a minha vida, sem a ajuda e motivação de todos para enfrentar os desafios, com certeza não teria chegado até aqui. Agradeço a Deus por estar me guiando neste caminho bonito e repleto de aventuras.

Agradeço ao meu professor orientador Dr. Adão Boava que me guiou nessa ultima etapa para concluir minha formação, dando muito suporte, conselhos e me motivando no desenvolvimento deste trabalho. Estendo os agradecimentos ao meu coorientador, colega de faculdade e de trabalho, o Engenheiro de Controle e Automação Christian Mailer, por toda a ajuda, contribuindo com dicas e conselhos. Agradeço também, ao meu outro amigo e colega de trabalho, o Engenheiro de Controle e Automação Gabriel Dinse por todo o suporte.

Agradeço a professora Dra. Janaína Gonçalvez Guimarães por ter me orientado ao longo de dois anos como estagiário no Laboratório de Circuitos e Sistemas Digitais – LABCID. Agradeço também, ao professor Dr. Leonardo Mejia Rincon que me guiou na minha Iniciação Científica - IC, ambos me transmitiram muito conhecimento e experiência nas áreas da eletrônica e da mecatrônica respectivamente.

Agradeço a todos os outros professores do curso de Engenharia de Controle e Automação por todo o conhecimento de qualidade que se esforçaram tanto em transmitir durante as aulas teóricas e práticas, promovendo uma das melhores formações do Brasil com nota máxima na avaliação do MEC.

"In real open source, you have the right to control your own destiny."
(Linus Torvalds)

Resumo

A computação de borda da 5° geração de rede móvel tem como alguns de seus requisitos a latência muito baixa e a confiabilidade muito alta para atender aplicações da indústria. Neste trabalho é utilizado um ambiente 5G simulado com o Free5GC Compose que contém as funções de rede (NFs -*Network Funtions*) implementadas em *containers* usando o Docker Compose. Neste ambiente é utilizado também o simulador de dispositivos de usuário e antena 5G, o UERANSIM. O dispositivo móvel simulado se conecta ao Core 5G a fim de acessar a rede de dados (DN - *Data Network*) em dois locais diferentes: rede de dados local (Borda) e a rede de dados central (*Data Center*) localizada mais distante. Por fim, é apresentado o desempenho em relação aos dois locais, através do monitoramento do tráfego de rede, para o estudo e entendimento dos conceitos da arquitetura 5G voltada para a computação de borda, a qual reduziu a latência de 160 ms para 1.3 ms na análise feita utilizando o protocolo ICMP.

Palavras-Chave: 5G, Computação de borda, Docker, Free5gc Compose, UERANSIM

Abstract

The 5th generation mobile network edge computing has as some of its requirements very low latency and very high reliability to satisfy industry applications. This work uses a 5G environment simulated with Free5GC Compose that contains the network functions (NFs) implemented in containers using Docker Compose. In this environment the simulator of user devices and 5G antenna, the UERANSIM, is also used. The simulated mobile device connects to the Core 5G in order to access the data network (DN) in two different locations: local data network (Edge) and the central data network (Data Center) located further away. Finally, the performance in relation to the two locations is presented, through the monitoring of network traffic, for the study and understanding of the concepts of the 5G architecture focused on edge computing, which reduced the latency from 160 ms to 1.3 ms in the analysis performed using the ICMP protocol.

Keywords: 5G, Edge Computing, Docker, Free5gc Compose, UERANSIM

Lista de Figuras

Lista de Tabelas

Lista de Quadros

Lista de Códigos Fonte

Lista de Abreviaturas e Siglas

1G	*1st Generation*
2G	*2nd Generation*
2.5G	*2.5rd Generation*
2.75G	*2.75rd Generation*
3G	*3rd Generation*
3GPP	*3rd Generation Partnership Project*
4G	*4th Generation*
5G	*5th Generation*
5GC	*Core 5G*
ABDI	*Agência Brasileira de Desenvolvimento Industrial*
AMF	*Access and Mobility Management Function*
AMPS	*Advanced Mobile Phone System*
Anatel	*Agência Nacional de Telecomunicações*
API	*Application Programming Interface*
ARIB	*Association of Radio Industries and Businesses*
ATIS	*Alliance for Telecommunications Industry Solutions*
AUSF	*Authentication Server Function*
BP	*Branch Point*
BR	*Brasil*
CCSA	*China Communications Standards Association*
CDMA	*Code Division Multiple Access*
CN	*China*
CPEs	*Customer-Premises Equipment*
CPU	*Central Processing Unit*
D2D	*Device to Device*
dBm	*Decibel miliwatt*
DE	*Deutsch*
DL	*Downlink*
DN	*Data Network*
DNN	*Data Network Name*
DNS	*Domain Name System*
DPA-MIMO	*Distributed Phased Arrays Based MIMO*
EC2	*Elastic Compute Cloud*
EGPRS	*Enhanced Data Rates For GSM Evolution*
ETSI	*European Telecommunications Standards Institute*

EUA	*United States of America*
E-UTRAN	*Evolved UMTS Terrestrial Radio Access Network*
FDD	*Frequency Division Duplex*
FDMA	*Frequency Division Multiple Access*
FQDN	*Fully Qualified Domain Name*
FN	*Finland*
FR	*France*
FR1	*Frequency Range 1*
FR2	*Frequency Range 2*
GB	*Gigabyte*
Gb/s	*Gigabit por segundo*
GHz	*Gigahertz*
gNB	*Next Generation Node Base*
GPL-3.0	*GNU General Public License v3.0*
GPRS	*General Packet Radio Services*
GSM	*Global System for Mobile Communications*
GSMA	*Global System for Mobile Communications Alliance*
GTP-U	*GPRS Tunneling Protocol User Plane*
HD	*High Definition*
HTTP	*Hypertext Transfer Protocol*
Hz	*Hertz*
ICMP	*Internet Control Message Protocol*
IMSI	*International Mobile Subscriber Identity*
IoT	*Internet of Things*
IP	*Internet Protocol*
IPsec	*IP Security Protocol*
ITU	*International Telecommunications Union*
I-UPF	*Intermediary - User Plane Function*
IPTV	*Internet Protocol Television*
JP	*Japan*
km^2	*Quilômetro quadrado*
Km/h	*Quilômetros por hora*
KR	*South Korea*
LTE	*Long Term Evolution*
MB	*Megabyte*
Mb/s	*Megabit por segundo*
ME	*Mobile Equipment*
MEC	*Multi-access Edge Computing*
MHz	*Megahertz*

MIMO	*Multiple Input Multiple Output*
MU-MIMO	*Multi User Multiple Input Multiple Output*
ms	*Milissegundo*
N3IWF	*Non-3GPP Inter Working Function*
NAT	*Network Address Translation*
NCTU	*National Chiao Tung University*
Netz-C	*Radio Telephone Network C*
NF	*Network Function*
NFV	*Network Function Virtualization*
NMT	*Nordic Mobile Telephone*
NR	*New Radio*
NRF	*Network Repository Function*
NSSF	*Network Slice Selection Function*
OFDMA	*Orthogonal Frequency Division Multiplexing / Multiple Access*
PCF	*Policy Control Function*
PDU	*Protocol Data Unit*
PFCP	*Packet Forwarding Control Protocol*
PLMN	*Public Land Mobile Network*
PSA	*PDU Session Anchor*
PSTN	*Public Switched Telephone Network*
QoS	*Quality of Service*
RAM	*Random Access Memory*
RAN	*Radio Access Network*
REST	*Representational State Transfer*
S-NSSAI	*Single Network Slice Selection Assistance Information*
SA	*Stand Alone*
SBA	*Service-Based Architecture*
SCTP	*Stream Control Transmission Protocol*
SDL	*Supplementary Downlink*
SDN	*Software Defined Networking*
SE	*Sweden*
SEP	*Standard-Essential Patent*
SMF	*Session Management Function*
SO	*Operational System*
SS	*Switching System*
SSC	*Service and Session Continuity*
SSD	*Solid-State Drive*
SU-MIMO	*Single User Multiple Input Multiple Output*
SUL	*Supplementary Up Link*

SUPI	*Subscription Permanent Identifier*
TCP	*Transmission Control Protocol*
TD-SCDMA	*Time Division-Synchronous Code Division Multiple Access*
TDD	*Time-Division Duplex*
TDMA	*Time-division multiple access*
TEID	*Tunnel Endpoint Identifier*
TSDSI	*Telecommunications Standards Development Society, India*
TSG	*Techniques Specifications Group*
TTA	*Telecommunications Technology Association*
TTC	*Toronto Transit Commission*
UDM	*Unified Data Management*
UDP	*User Datagram Protocol*
UDR	*Unified Data Repository*
UMTS	*Universal Mobile Telecommunication System*
UE	*User Equipment*
UFSC	*Universidade Federal de Santa Catarina*
UL	*Up Link*
UL CL	*Up Link Classifier*
UPF	*User Plane Function*
UPFB	*User Plane Function Branch*
US	*United States*
USIM	*User Services Identity Module*
VM	*Virtual Machine*
VMM	*Virtual Machine Monitor*
WEBUI	*User Grafic Interface*
WCDMA	*Wide Band Code Divison Multiple Access*
WiMAX	*Worldwide Interoperability for Microwave Access*

Lista de Símbolos

$	*Terminal do Ubuntu Server 20.04*
#	*Inicializa um comentário*
%	*Porcento*

Sumário

1 Introdução

A maneira de prestar serviços na área de telecomunicações móveis está passando por uma grande transformação. Esses serviços possuem requisitos que exigem novas arquiteturas de telecomunicação para garantir a qualidade dos serviços prestados. O projeto, desenvolvimento e implantação dessas arquiteturas precisam evoluir para atender de forma rápida e flexível esses requisitos rigorosos de comunicação. Para atender a essas exigências, existe uma grande expectativa nas redes móveis de quinta geração (5G). Com o 5G espera-se ser possível, por exemplo, a prestação de serviços para as indústrias inteligentes (Indústria 4.0), telemedicina, fazendas inteligentes, *Blockchain*, veículos autônomos e a Internet das Coisas (IoT - *Internet of Things*) [1].

A forte conectividade do 5G irá promover o fluxo total de dados e informações em uma gama mais ampla para a inteligência artificial, computação em servidores remotos (Cloud) e processamento de dados em servidores locais (computação de borda) em todos os campos econômicos da sociedade. A rede 5G pode transportar serviços de banda larga móvel de alto tráfego como realidade aumentada, vídeos de ultra-alta definição 4K/8K e atender de uma maneira mais eficiente à IoT em grande escala com base em comunicações do tipo dispositivo para dispositivo (D2D -*Device to Device*) em massa e fornecer melhor suporte na prestação de serviços que exigem baixa latência e alta confiabilidade [2]. Assim, o 5G aumentará consideravelmente a velocidade de troca de informações, comparado às gerações anteriores.

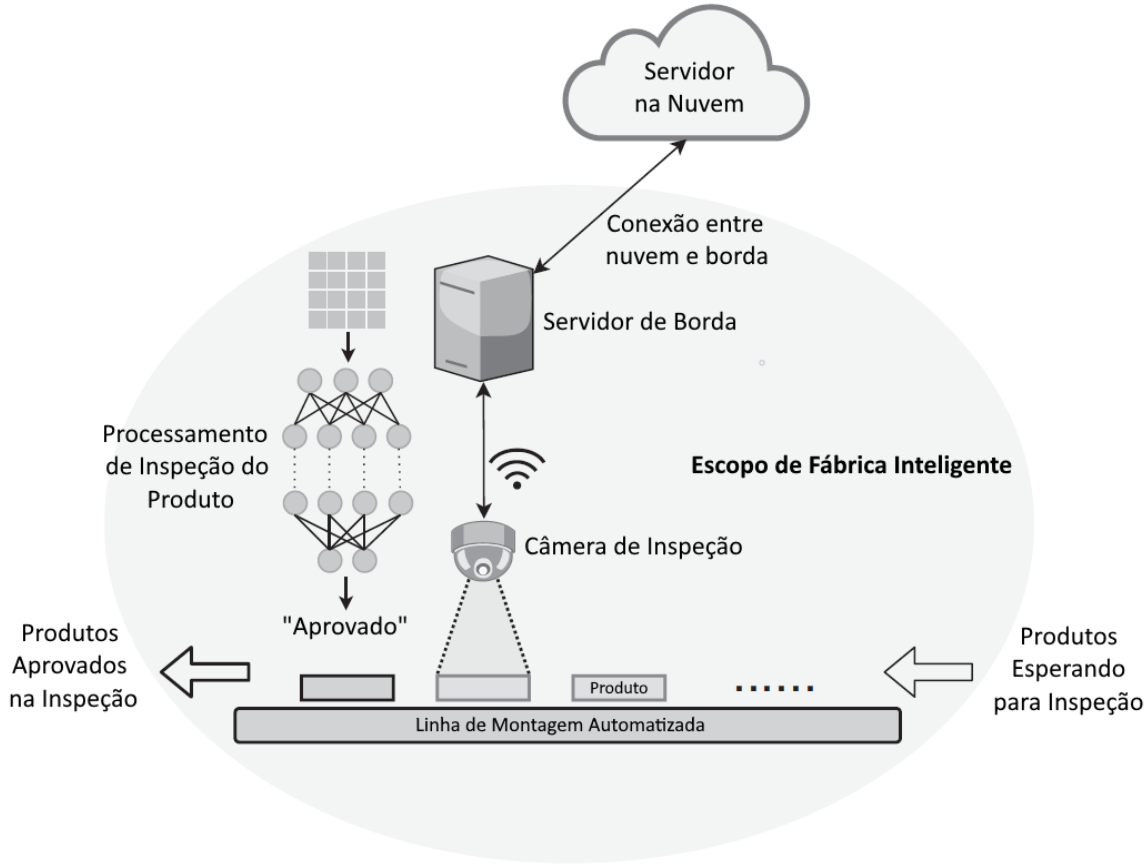
1.1 Problematização

Dispositivos de IoT tendem a ter um baixo custo, o que implica em um baixo poder de processamento de dados, assim, para muitas das aplicações os dados necessitam ser enviados para um servidor a fim de serem processados. Em alguns casos, a distância entre o dispositivo de IoT e o servidor torna isso inviável devido à barreiras físicas ou ao alto custo de implementação. Além disso, a administração de tantos protocolos de comunicação industrial diferentes torna difícil a integralização e unificação dos dados a serem processados.

Assim, para aliviar os gargalos de latência e com eficiência energética, o 5G disponibiliza novos serviços. A utilização dos novos serviços possibilita a solução do paradigma da computação de borda que baseia-se em uma rede de computação em multicamadas e que pode ser introduzida em fábricas inteligentes, sendo possível também com as novas funções de rede do Core 5G administrar e integralizar sinais não 5G, ou seja, não 3GPP (*3rd Generation Partnership Project*), como o WiFi (IEEE 802.11) e entre outros.

Considere o cenário apresentado na Figura 1, em que uma câmera de inspeção de superfície é um dispositivo 5G 3GPP, configurada em uma linha de montagem automatizada de uma fábrica inteligente fictícia. Os produtos estão fluindo na linha de montagem automatizada, aguardando a inspeção um por um. Quando um produto chega à posição certa, a superfície dele será capturada pela câmera de inspeção, que é um dispositivo IoT de baixo poder de processamento de dados. Então, as imagens vão ser carregadas para o servidor 5G na borda que fará a inferência e análise quase em tempo real [3]. Por fim, as imagens também podem ser transmitidas para o servidor em nuvem para futuras análises, armazenamento de *backup* e outras apurações que podem ser realizadas pelas funções de rede do Core 5G.

Figura 1 – Linha de montagem automatizada.



Fonte: Adaptado de [3].

O ganho obtido pelo usuário com o a utilização do sistema 5G, tal qual o da Figura 1, em relação as tecnologias atuais — como será visto no Capítulo 2 — tem como um de seus objetivos garantir comunicações com baixa latência ultra confiável (URLLC – *Ultra-Reliable Low-Latency Communication*), aumentando a confiabilidade do sistema e diminuindo a latência nas aplicações críticas da indústria. Com o Core 5G pode-se também integralizar os dispositivos Não-3GPP com os 3GPP, unificando os dados e usufruindo de todas as novas funções (serviços) do 5G que serão apresentados no Capítulo 2

e ao longo do trabalho, tendo ênfase na computação de borda 5G que pode ser utilizada no cenário da Figura 1. Assim, as novas funções de rede do 5G geram ganhos em desempenho, segurança, escalabilidade, unificação de protocolos industriais diferentes e entre outras vantagens.

1.2 Objetivo geral

Este trabalho tem como objetivo principal estudar e utilizar plataformas voltadas para a simulação e teste do sistema 5G aplicado para a computação de borda que pode ser utilizada, por exemplo, na linha de montagem automatizada apresentada na Figura 1.

1.2.1 Objetivos específicos

A seguir está descrito os objetivos planejados divididos em pequenas partes e metas:

1. Para a realização deste trabalho se faz necessário inicialmente um estudo aprofundado sobre todo o sistema 5G e seus serviços;
2. Configuração e simulação da plataforma para simulação do 5GC em uma arquitetura de computação de borda;
3. Simulação e configuração da plataforma de simulação de dispositivo 5G móvel e antena de uma rede de acesso de rádio (RAN - *Radio Access Network*);
4. Simulação completa do (UE - *User Equipment*) se conectando no servidor de borda e no servidor remoto através de uma conexão que deve ser estabelecido entre UE, antena, 5GC e servidores;
5. Estudo e utilização de ferramentas para a obtenção de dados referentes a desempenho de redes para a geração de gráficos.

1.3 Estrutura do documento

O Capítulo 2 contém uma pesquisa sobre patentes e mercado da computação de borda 5G, história das redes móveis, objetivos da 5° geração de redes móveis, o ambiente 5G e seus serviços, dispositivo de usuário e antena, bem como, a literatura sobre a computação de borda e virtualização. Os procedimentos e ferramentas foram descritos no Capítulo 3. No Capítulo 4, estão expostos os resultados obtidos e a análise do funcionamento de cada etapa do sistema. Finalmente, no Capítulo 5 é realizado uma avaliação geral sobre os resultados obtidos, impacto do sistema e futuras pesquisas que podem ser desenvolvidas.

2 Revisão de literatura

Neste capítulo é apresentado os conceitos fundamentais selecionados pelo autor, tendo como objetivo sanar as possíveis dúvidas dos leitores nos capítulos seguintes e aprofundar o próprio conhecimento do autor em relação aos assuntos aqui abordados, assim como, apresentar uma visão do mercado atual e pesquisas feitas referente a computação de borda 5G.

2.1 Patentes e mercado da computação de borda 5G

Na Tabela 1 é apresentado as principais empresas no depósito de patentes, declaração de patentes essenciais padrões (SEP - *Standard-Essential Patent*) e desenvolvimento de padrões relacionados com computação de borda 5G. Essas empresas são do ramo de computação em nuvem ou de setores de dados e *software*.

Tabela 1 – Relação empresas/desenvolvedores e número de patentes que descrevem tecnologias de computação de borda.

	Cessionário atual/ Desenvolvedor de padrões	**Pedidos de patente**	**Declaração SEP**	**Contribuições de padrões**
1	Huawei (CN)	821	138	862
2	Intel (US)	686	42	488
3	Nokia (FN)	576	87	439
4	SAS Institute (US)	426	0	0
5	Apple (US)	386	72	41
6	Samsung Electronics (KR)	287	16	536
7	Verizon (US)	196	0	50
8	Microsoft (US)	188	0	0
9	Cisco (US)	168	0	39
10	Ericsson (SE)	163	6	374
11	LG Electronics (KR)	160	33	144
12	NEC (JP)	158	3	55
13	Pure Storage (US)	155	0	0
14	IBM (US)	125	0	0
15	Siemens (DE)	120	0	30
16	Sony (JP)	119	0	66
17	AT&T (US)	99	0	130
18	ZTE (CN)	96	4	193

19	Qualcomm (US)	68	6	256
20	Tencent (CN)	64	0	117
21	Convida Wireless (US)	60	0	88
22	CATT Datang Mobile (CN)	55	2	0
23	China Mobile (CN)	54	0	206
24	Deutsche Telekom (DE)	47	0	64
25	InterDigital (US)	46	2	77
26	SoftBank (JP)	46	0	4
27	Orange (FR)	41	0	60
28	Hewlett Packard Enterprise (US)	39	0	19
29	ETRI (KR)	37	1	29
30	Fraunhofer (DE)	35	11	17
31	Robert Bosch (DE)	34	0	10
32	Sharp (JP)	30	2	0

Fonte: Adaptado de [4].

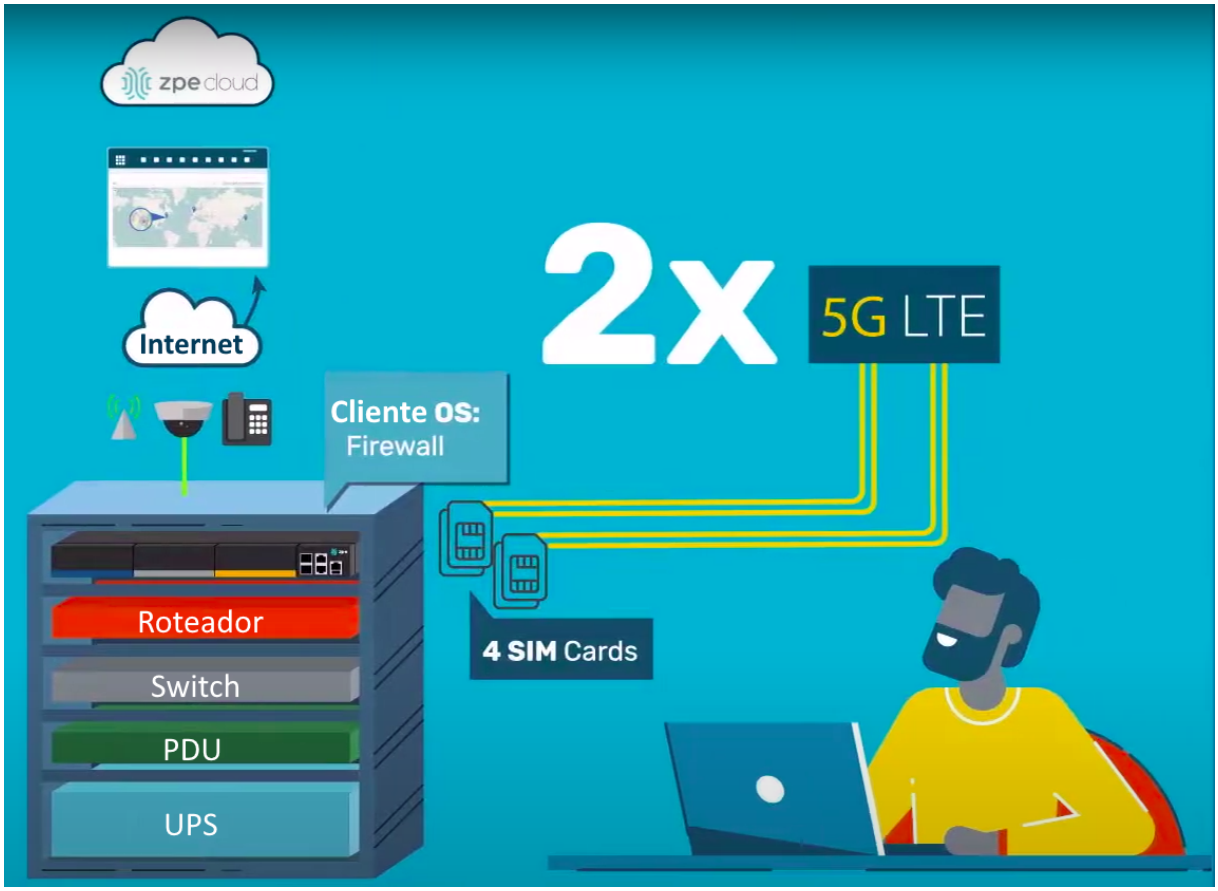
Fabricantes de chips, celulares e redes como Huawei (China), Intel (EUA), Nokia (Finlândia), Apple (EUA), Samsung Electronics (Coréia) e Ericsson (Suécia) contribuem fortemente para o desenvolvimento de padrões e, ao mesmo tempo, possuem grandes carteiras de patentes, algumas das quais são declaradas essenciais como padrão, isto é, uma patente que se torna um padrão que outros desenvolvedores ou empresas devem seguir. Além disso, muitas das operadoras de telecomunicações podem ser encontradas na lista dos principais proprietários de patentes e desenvolvedores de padrões, como Verizon (EUA), *AT&T* (EUA), China Mobile (China), Deutsche Telekom (Alemanha) e Orange (França).

Os líderes de tecnologia listados na Tabela 1 são de extrema importância para o sucesso da computação de borda, pois têm desenvolvido dispositivos, chips, redes, aplicativos, serviços, sensores e padrões de conectividade para realizar os primeiros casos de uso da computação de borda. Prevê-se que o mercado global da computação de borda em nuvem crescerá para $US\$12$ bilhões ainda em 2021 [4]. Supõe-se que, até 2023, cerca de 70% das empresas estarão realizando parte de seus processamentos de dados usando a computação de borda 5G [4]. Assim, o interesse por uma parcela desse mercado é propagado entre várias empresas do ramo de desenvolvimento de tecnologia.

Uma dessas empresas é a ZPE Systems que é sediada na cidade de Fremont no estado da Califórnia (EUA) e possui uma de suas filiais em Blumenau no estado de Santa Catarina (BR). Essa empresa produz equipamentos para *data center* que contém a tecnologia 5G embarcada e, para isso, tem parceiros como a Intel e Cisco, listados na Tabela 1.

Na Figura 2 é apresentado uma arquitetura de *data center*, no qual encontra-se em seu topo (acima do roteador e de cor preta) um dos modelos de Nodegrid [5] dessa empresa para a administração e controle dos equipamentos. Conforme a mesma figura, ele suporta 2 SIM Cards 5G e 2 SIM Cards 4G LTE (*Long-Term Evolution*) que são suficientes para garantir a redundância exigida para este tipo de equipamento, podendo acessá-lo pela internet utilizando a plataforma ZPE Cloud de forma segura e eficiente [6]. Com isto, pode-se entender através deste exemplo o tipo de infraestrutura que espera-se encontrar para os servidores de borda ou na nuvem.

Figura 2 – Evitando o tempo ocioso com uma infraestrutura de borda melhor.



Fonte: Adaptado de [6].

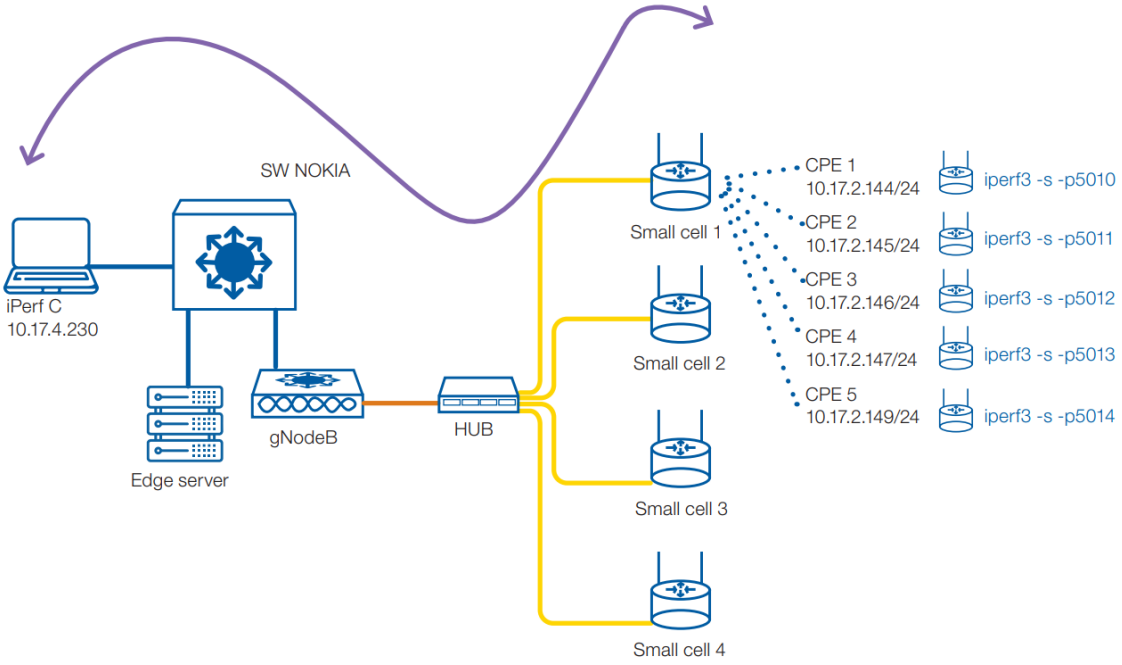
Outra empresa também interessada na computação de borda 5G é a WEG que tem criado parcerias com empresas da Tabela 1, como: Nokia e Qualcomm, além da já existente cooperação com a Agência Brasileira de Desenvolvimento Industrial (ABDI), a Agência Nacional de Telecomunicações (Anatel) e a Claro. O Open Lab 5G - V2COM da WEG em uma das fábricas, localizada na cidade de Jaraguá do Sul no estado de Santa Catarina (BR) completou testes práticos de conectividade à rede 5G. Foram realizados testes para avaliação de desempenho e a convivência de dispositivos e antenas com a tecnologia 5G em ambiente real para reunir informações sobre faixas de frequência, latência, potência e

outras características necessárias às aplicações industriais [7].

De acordo com o relatório de resultados preliminares do Open Lab 5G - V2COM [8], foram obtidos dados experimentais utilizando a ferramenta *iPerf* [9], com transmissão de dados tanto TCP como em UDP ativando tráfego em cinco premissas de equipamentos de usuário (CPEs - *Customer-premises equipment*) parados e a uma distância de aproximadamente cinco metros de uma *Small Cell*: rede privada independente da Nokia na banda n78 (3,5 GHz), faixa de 3,7 GHz a 3,8 GHz, operando com largura de canal de 100 MHz, no modo SA (*Stand Alone*) e TDD (*Time-Division Duplex*) com distribuição 3:7 e saída de potência (*Effective Isotropic Radiated Power - E.I.R.P.*) de 23 dBm.

Um dos testes do relatório de resultados preliminares do Open Lab 5G - V2COM [8] é o de *Throughput*, latência e perda de pacotes versus a distância entre a *Small Cell* e um dos CPEs. Tais testes foram realizados posicionando fisicamente os CPEs a diferentes distâncias da *Small Cell* 4 da rede privativa independente – SA representada na Figura 3, ou seja, apenas uma *Small Cell* ligada e as outras três desligadas.

Figura 3 – Topologia do teste da rede privativa independente – SA.

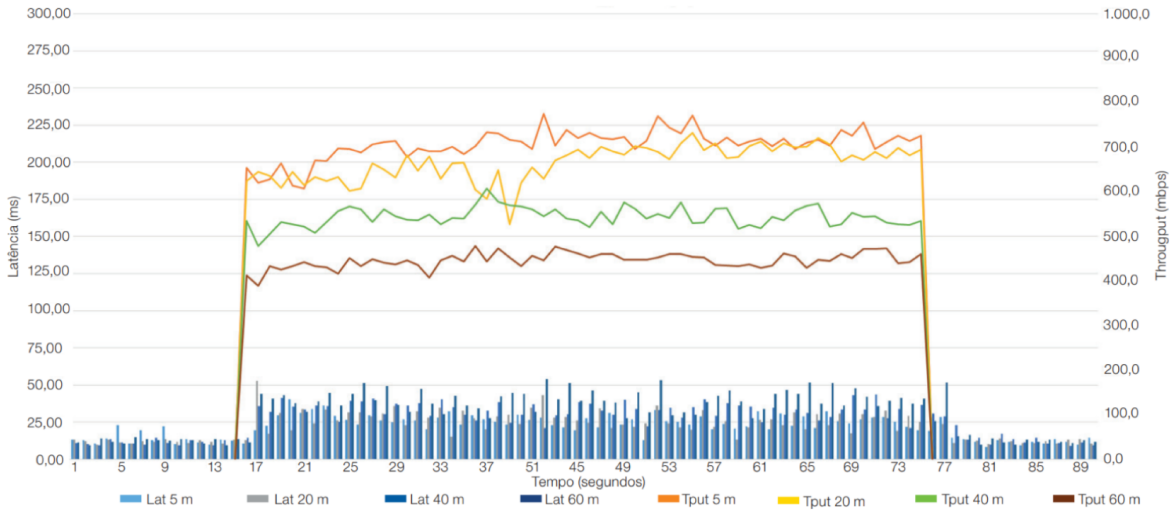


Fonte: Adaptado de [8].

Foram realizados testes utilizando-se a ferramenta *iPerf* com transmissão de dados em TCP para esta topologia em específico. As verificações foram feitas com o CPE parado às distâncias de 5, 20, 40 e 60 metros da *Small Cell*. Os valores de *Throughput* máximo, médio e mínimo obtidos durante o tempo de um minuto de medição são apresentados nas Figuras 4 e 5. Também são apresentados nas mesmas figuras, os valores de latência médios obtidos durante quinze segundos antes do início do envio dos pacotes de dados,

durante o um minuto em que são enviados os pacotes de dados e durante quinze segundos depois do envio dos pacotes de dados. Todas essas medições são apresentadas para cada uma das quatro distâncias em um mesmo gráfico para melhor visualização e comparação. São apresentados também, para as diferentes distâncias, os percentuais de redução de *Throughput* obtidos com relação ao *Throughput* a 5 metros.

Figura 4 – Downlink – Latência vs Throughput máximo (TCP).

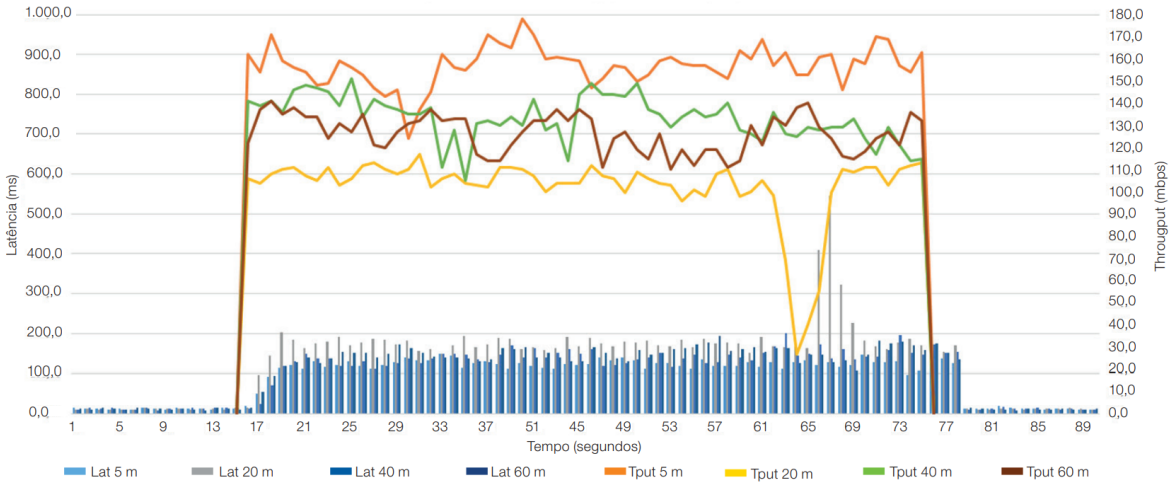


Lat 5 m Média rtt (ms)	
Antes	13,93
Durante	25,76
Depois	13,41
Lat 20 m Média rtt (ms)	
Antes	11,92
Durante	27,09
Depois	13,81
Lat 40 m Média rtt (ms)	
Antes	11,28
Durante	32,95
Depois	15,21
Lat 60 m Média rtt (ms)	
Antes	12,32
Durante	39,64
Depois	15,96

Tput 5 m (Mbps)		
Med	706,5	0%
Mín	607,0	
Máx	775,0	
Tput 20 m (Mbps)		
Med	665,7	-5,8%
Mín	527,0	
Máx	733,0	
Tput 40 m (Mbps)		
Med	544,0	-23,0%
Mín	478,0	
Máx	607,0	
Tput 60 m (Mbps)		
Med	446,3	-36,8%
Mín	390,0	
Máx	478,0	

Fonte: Adaptado de [8].

Figura 5 – Uplink – Latência vs Throughput máximo (TCP).



Lat 5 m Média rtt (ms)	
Antes	12,26
Durante	119,73
Depois	35,86
Lat 20 m Média rtt (ms)	
Antes	11,57
Durante	183,07
Depois	42,09
Lat 40 m Média rtt (ms)	
Antes	12,54
Durante	136,46
Depois	41,41
Lat 60 m Média rtt (ms)	
Antes	11,69
Durante	144,70
Depois	39,56

Tput 5 m (Mbps)		
Med	157,0	0%
Mín	124,0	
Máx	178,0	
Tput 20 m (Mbps)		
Med	102,4	-34,7%
Mín	26,5	
Máx	117,0	
Tput 40 m (Mbps)		
Med	133,3	-15,1%
Mín	105,0	
Máx	151,0	
Tput 60 m (Mbps)		
Med	126,4	-19,5%
Mín	110,0	
Máx	141,0	

Fonte: Adaptado de [8].

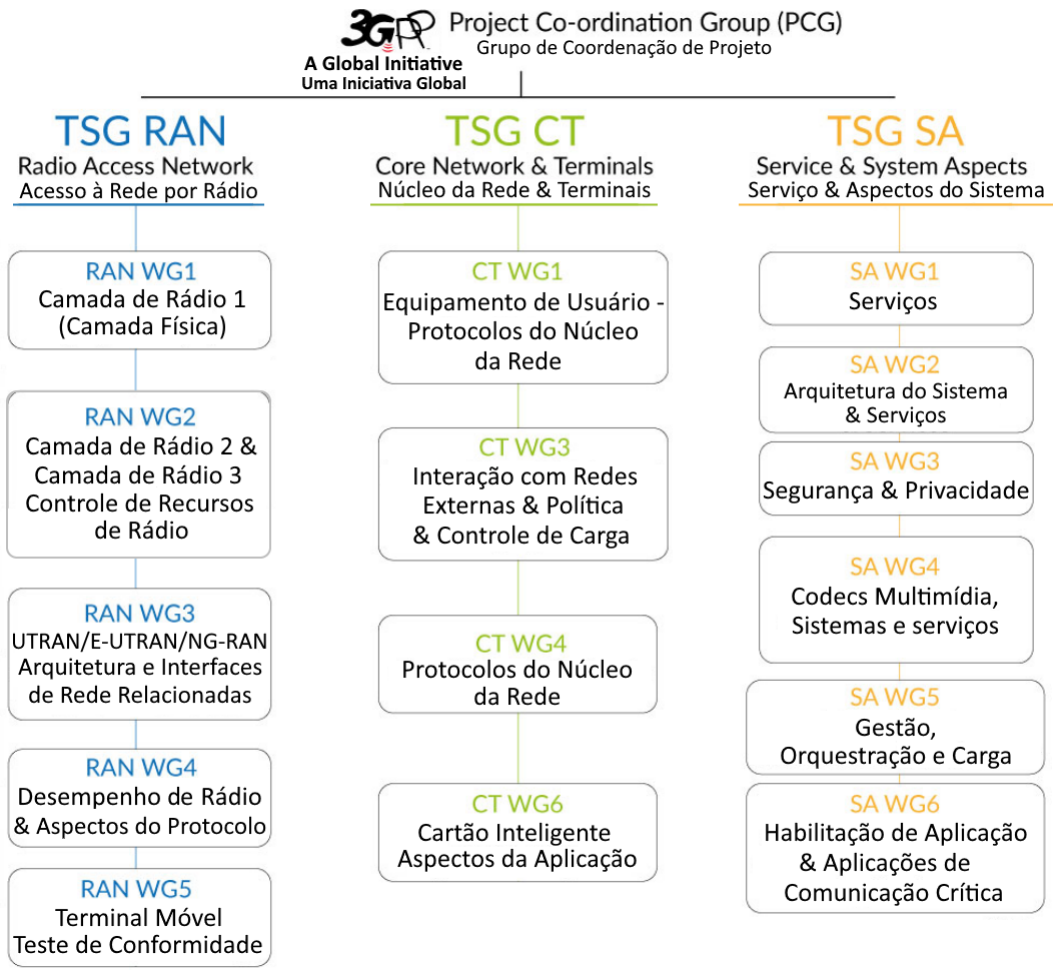
Neste pesquisa realizada pela WEG foram obtidos dados de grande valor para este trabalho. Dados que serão utilizados a título de comparação com os resultados que espera-se obter nesta obra.

2.2 3GPP

O Projeto de Parceria de 3^a Geração (3GPP) reúne 7 organizações de desenvolvimento de padrões de telecomunicações: Associação de Indústrias e Empresas de Rádio (ARIB - Association of Radio Industries and Businesses) do Japão, Aliança para Soluções do Setor de Telecomunicações (ATIS - *Alliance for Telecommunications Industry Solutions*) da América do Norte, Associação de Padrões de Comunicações da China (CCSA - *China Communications Standards Association*), Instituto Europeu de Normas de Tele-

comunicações (ETSI - *European Telecommunications Standards Institute*), Sociedade de Desenvolvimento de Padrões de Telecomunicações, India (TSDSI - *Telecommunications Standards Development Society, India*), Associação de Tecnologia de Telecomunicações (TTA -*Telecommunications Technology Association*) da Coreia do Sul e a Comissão de Trânsito de Toronto (TTC - *Toronto Transit Commission*) do Canada. Conhecidos como "parceiros organizacionais", eles fornecem a seus membros um ambiente estável para produzir os relatórios e especificações que definem as tecnologias 3GPP [10]. O 3GPP define especificações para um sistema móvel completo, incluindo aspectos relativos aos terminais, redes de acesso de rádio, redes principais, e partes da rede de serviços. Órgãos de normatização em cada região do mundo têm autorização para receber os resultados do 3GPP e publicá-los na sua região, como normas ou recomendações formais. As especificações do 3GPP estão estruturadas em versões. Normalmente, as discussões de tecnologias do 3GPP referem-se à funcionalidades em uma ou outra versão [11]. E todas as novas versões são compatíveis com as versões anteriores. A seguir, na Figura 6 está representado os três grupos de especificações técnicas (TSG - *Technical Specification Group*) do 3GPP.

Figura 6 – Os três TSGs do 3GPP.



Fonte: Adaptado de [10].

2.3 Marcos históricos das redes móveis

Historicamente as redes de comunicação móveis tiveram início durante a década de 1980 com a primeira geração 1G, sistema analógico pensado somente para chamadas por voz [12]. Os primeiros celulares possuíam tamanho e peso significativos (tipicamente de vários quilogramas) com o auricular separado, conforme ilustrado na Figura 7, na qual é apresentado o modelo NMT-450 da Motorola.

Figura 7 – Motorola NMT-450.

Fonte: Adaptado de [13].

As tecnologias dessa geração são: telefone móvel nórdico (NMT - *Nordic Mobile Telephone*) que foi o primeiro sistema de telefone celular totalmente automático, o telefone por rede de rádio do tipo C (Netz-C - *Radio Telephone Network C*) e o sistema de telefone móvel avançado (AMPS - *Advanced Mobile Phone System*). Nesse contexto, o AMPS foi o padrão de rede de acesso 1G mais amplamente implementado. Para esta rede de acesso e núcleo desta arquitetura o MTSO (*Mobile Telecommunications Switch Office*) é considerado como núcleo, pois realiza tarefas de controle de *handover*, registro e autenticação de UEs, roteamento e gestão de ligações telefônicas (*paging*, *handoff*, etc.). Além disso, o MTSO interliga a PSTN (*Public Switched Telephone Network*) ou SS (*Switching System*) às múltiplas BTSs (*Base Transceiver Station*) [14] que por sua vez realizam as conexões com UEs através de canais analógicos em modulação por frequência (FM - *Frequency Modulation*) e acesso múltiplo aos poucos canais disponíveis via FDMA (*Frequency Division Multiple Access*) em um espectro licenciado [15].

2.3.1 Segunda Geração - 2G

Aproximados 10 anos depois, durante a década de 1990, surgiu a rede 2G com grandes avanços. O primeiro avanço foi a substituição do sistema de comunicação analógico pelo digital, integração de serviços de mensagens curtas de texto (SMS - *Short Message Service*) e de outros dados entre dispositivos, utilização do múltiplo acesso por divisão de tempo (TDMA - *Time Division Multiple Access*), múltiplo acesso por divisão de código (CDMA - *Code Division Multiple Access*) e do PDC (*Personal Digital Cellular*), tecnologia que foi usada exclusivamente no Japão. Ressalta-se que o IS-95 (*Interim Standard 95*) foi a primeira tecnologia de celular digital baseada em CDMA e o IS-136 (*Interim Standard 136*) é conhecido como TDMA [14]. O Sistema Global de Comunicações Móveis (GSM - *Global System for Mobile Communications*) é a primeira versão implementada, seguido pela 2.5G, o qual utiliza o padrão de Serviços Gerais de Pacotes por Rádio (GPRS - *General Packet Radio Service*), introduzindo o suporte à comutação por pacotes. Essa evolução foi finalizada com a chamada geração 2.75G, através da tecnologia EDGE (Enhanced Date Rates For GSM Evolution), que apresentou um padrão de melhoria nas taxas de dados para o GSM [14].

O 2G tem canais de comunicação compactos utilizando multiplexadores, fazendo com que os dados possam ser transmitidos pelo mesmo canal para muitos usuários [15]. Para o 2.5G, a tecnologia GPRS trouxe novos grandes avanços, como a transmissão de pacotes de dados, suportando conexão com a internet, serviços de multimídia e navegação por *web browsers* através do UE [15]. Toda a família do sistema 2G ainda está operacional hoje em dia, sendo que importantes evoluções foram implementadas para dar suporte ao aumento da demanda por serviços de dados.

2.3.2 Terceira Geração - 3G

Em seguida veio a rede 3G no ano de 2004, tendo como principal melhoria a banda larga. Essa rede também tornou possível a portabilidade e conectividade universal [15]. As tecnologias utilizadas nessa geração são: CDMA 2000, GSM, EGPRS (*Enhanced Data Rates For GSM Evolution*), sistema móvel universal de telecomunicação (UMTS - *Universal Mobile Telecommunication System*), TD-SCDMA (*Time Division-Synchronous Code Division Multiple Access*) e, além dos padrões citados, o CDMA de banda larga (WCDMA - *Wideband CDMA*) foi amplamente adotado [14].

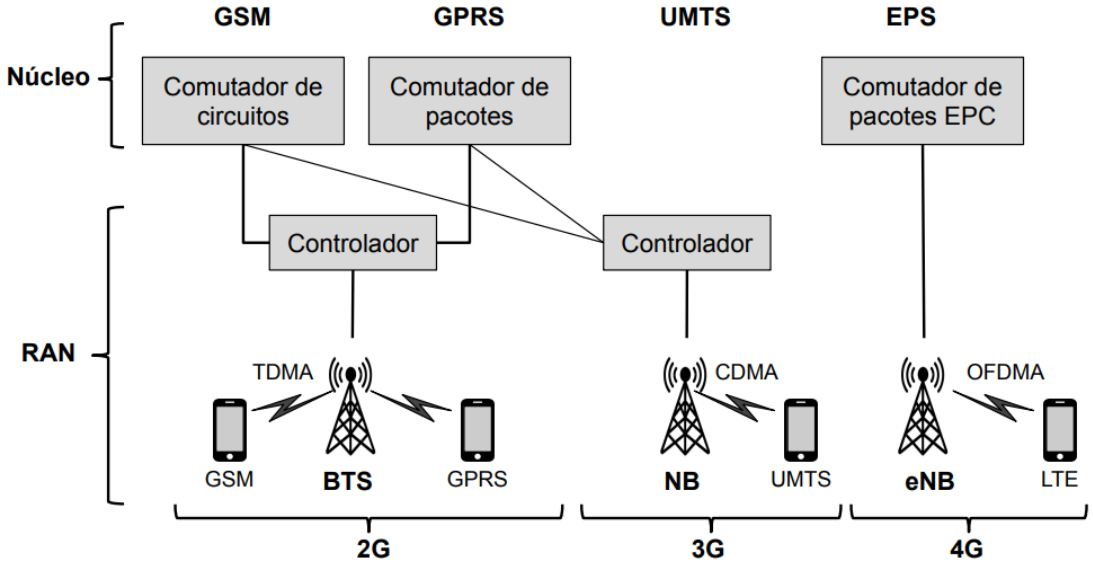
2.3.3 Quarta Geração - 4G

A rede 4G, lançada em 2010, revolucionou a utilização da Internet na telefonia móvel. Dois padrões emergiram dos esforços globais para a padronização dessa rede, sendo o LTE desenvolvido pela 3GPP que define uma rede de acesso composta por estações rádio

base do tipo NodeB evoluída (eNB - *evolved NodeB*), as quais são interligadas entre si e com o núcleo EPC (*Evolved Packet Core*). Este sistema é denominado de protocolo EPS (Evolved Packet System) que fornece procedimentos para o controle da mobilidade quando o UE utiliza a Rede de Acesso Rádio Terrestre UMTS Evoluída (E-UTRAN). Ele também fornece controle de segurança para os protocolos NAS[14].

O segundo padrão é o IEEE 802.16, também chamado de WiMAX (*Worldwide Interoperability for Microwave Access*), desenvolvido por um grupo de trabalho do IEEE (*Institute of Electrical and Electronics Engineers*). Conforme sugerido pelo consórcio de seus fabricantes, ele foi motivado pelo enorme sucesso do IEEE 802.11, também chamado de WiFi, utilizado para a comunicação sem fio em redes locais. Entretanto, o WiMAX teve como objetivo a operação em comunicações sem fio fixas de alta velocidade de dados a fim de satisfazer as necessidades do 4G. Os dois padrões se mostraram equivalentes em termos de desempenho e tecnologia, baseados em acesso múltiplo via multiplexação ortogonal por divisão de frequência (OFDMA - *Orthogonal Frequency Division Multiplexing / Multiple Access*) e SC-FDMA (*Single-Carrier Frequency Division Multiple Access*) [14]. Dessa forma, o WiMAX assumiu um importante papel como tecnologia de acesso sem fio de banda larga fixa, enquanto que o LTE se tornou o padrão universal para a interface aérea do 4G. As velocidades máximas e médias dessa rede são aproximadamente 10 vezes maior que a do 3G, possibilitando chamadas de video em alta definição (HD - *High Definition*). Na Figura 8 é apresentado uma visão geral da evolução das redes móveis 2G até a rede 4G.

Figura 8 – Evolução arquitetural das redes móveis.



Fonte: Adaptado de [14].

No Quadro 1 é apresentado de forma resumida a evolução das gerações de redes móveis,

apresentando características importantes sobre cada uma delas, como por exemplo as velocidades de transferência de dados para cada geração.

Quadro 1 – Evolução das gerações e características importantes.

	1G	2G	3G	4G
Implementação	≈ 1984	≈ 1991	≈ 1999	≈ 2009
Serviços	Voz analógica	Voz digital SMS	Pacotes de dados de alta capacidade	Comunicação IP
Taxa de dados	1.9 kbps	14.4 kbps	384 kbps	200 Mbps
Multiplexação	FDMA	TDMA, CDMA	TDMA, CDMA	OFDMA, SC-FDMA
Rede de acesso	AMPS	GSM, PDC, IS-95, IS-136, EDGE, GPRS	CDMA 2000, UMTS, TD-SCDMA,WCDMA	LTE, WiMAX
Núcleo da rede	PSTN	PSTN - SS	PSTN - SS, rede de pacotes	EPC
3GPP	-	-	Lançamento 1999	Lançamento 8

Fonte: Adaptado de [14].

Ressalta-se que a 3GPP, conforme a última linha da Quadro 1 teve sua inicialização em dezembro de 1998 com o objetivo de desenvolver uma especificação para um sistema de telefonia móvel 3G baseado no sistema 2G GSM, daí o nome 3GPP [10].

2.4 Objetivos do 5G

A fim de atender às expectativas e necessidades identificadas no mercado e na indústria para casos de uso existentes e novos, uma série de itens sobre as características do serviço foram definidos para servir como objetivo de *design* e de especificações do 5G [16].

Para dar suporte a essa ampliação do modelo de negócio das empresas de telecomunicações, três requisitos principais foram definidos:

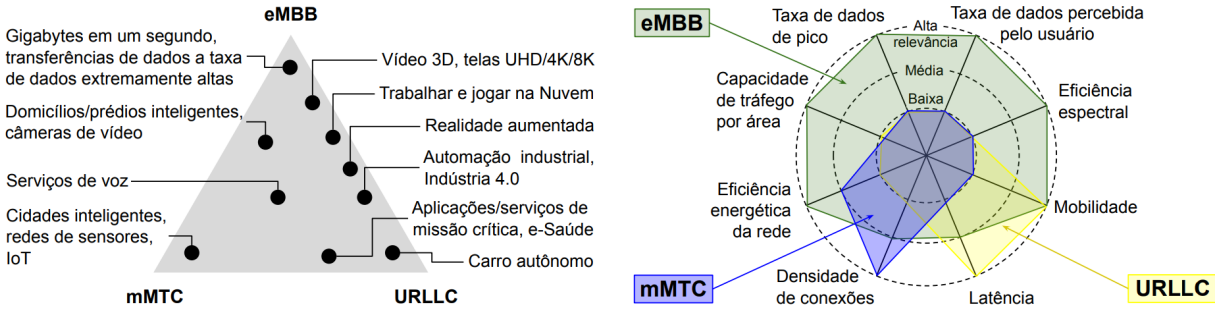
1. Banda larga móvel aprimorada (eMBB – *Enhanced Mobile Broadband*), referente aos serviços de banda larga móvel que atendam às necessidades de lidar de forma eficiente com volumes de dados muito grandes e crescentes na rede, otimizando a sua capacidade além de fornecer uma experiência de usuário aprimorada em partes maiores da rede;

2. Comunicações massivas de tipo de máquina (mMTC – *Massive Machine Type Communications*), casos de uso que visam um grande número de dispositivos pequenos ou baratos, suportando aplicativos da Internet das Coisas. Aqui, os requisitos incluem, alta eficiência energética para otimizar a vida útil da bateria desses dispositivos e alta densidade de conexão para poder atender a um grande número de dispositivos, mesmo em uma área geográfica limitada;

3. O (URLLC) para aplicações críticas da indústria e para os negócios, alguns dos requisitos mais importantes são latência muito baixa e confiabilidade muito alta.

Esses cenários são apresentados pela Figura 9 (a) que foi introduzida pela ITU (International Telecommunication Union) [17], em 2015, como parte da visão do que seriam redes 5G. Essa imagem, assim como a Figura 9 (b) que apresenta as capacidades principais de cada cenário, são muito utilizadas para tentar resumir algumas das propriedades mais relevantes de redes 5G.

Figura 9 – Requisitos definidos para as redes 5G (à esquerda) e as capacidades a serem ofertadas nesses cenários (à direita).



(a) Cenários 5G (b) Capacidades dos cenários

Fonte: Adaptado de [14].

Os requisitos serviram como entrada para os estudos técnicos propiciados pelo 3GPP, a partir dos quais eles foram formulados no relatório técnico [18]. Ressalta-se que este trabalho focará no item (3) dos objetivos do 5G, referente a aplicações críticas da indústria e para os negócios.

No Quadro 2 é apresentado um resumo com valores numéricos de alguns dos requisitos de serviço 5G mais importantes. Como os requisitos dependem do caso de uso e são bastante diversos, isso significa que a tecnologia de rádio do 5G precisava ser projetada de maneira flexível de modo que uma ampla variedade de casos de uso pudesse ser suportada de forma eficiente.

Quadro 2 – Requisitos de serviço para o 5G. **DL**=*Down Link* e **UL**=*Up Link*.

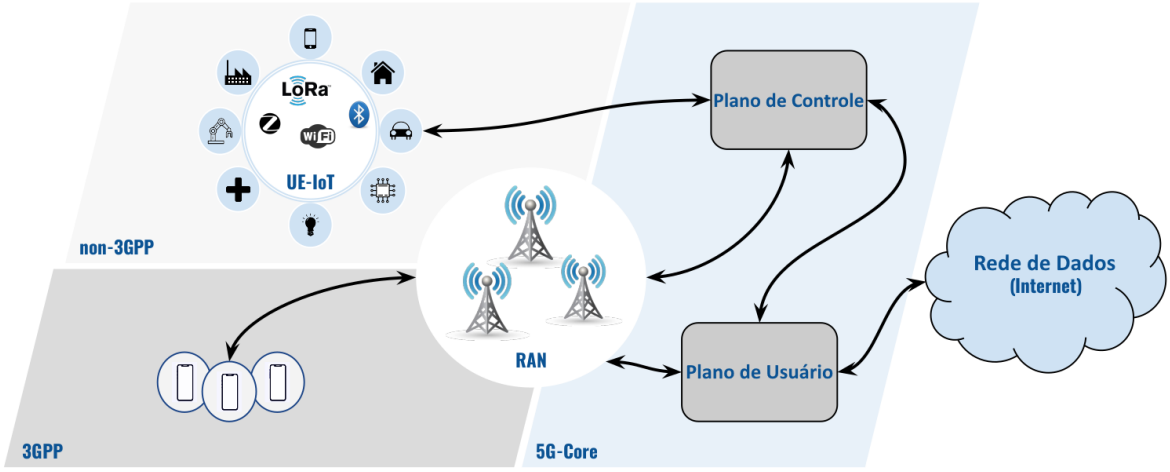
	Serviços 5G	**Requisitos**
1	Velocidade de pico	Até 20 Gb/s DL, até 10 Gb/s UL
2	Velocidade média	Até 100 Mb/s DL, até 50 Mb/s UL
3	Eficiência espectral	Até 30 $bits/s/Hz$ DL, até 15 $bits/s/Hz$ UL
4	Densidade de conexões	Até 1 milhão de dispositivos/km^2
5	Vida útil da bateria dos dispositivos	Mais de 10 anos
6	Mobilidade	Até 500 km/h
7	Latência de dados do usuário	$1ms$ para casos de uso da indústria, $4ms$ para banda larga móvel
8	Confiabilidade	Pelo menos $99,999\%$

Fonte: Adaptado de [16].

2.5 Ambiente 5G

Para uma melhor compreensão e ambientação do leitor, a Figura 10 apresenta um cenário para sistemas 5G com uma rede de acesso 3GPP e seus UE, conectados em antenas (RAN) e uma rede de acesso não-3GPP com seus UE-IoT. Essas redes estão conectadas ao núcleo 5G (*Core* 5G - 5GC), utilizando os planos de controle (funções de rede do 5GC para administração da rede não relacionadas aos usuários) e de usuário (funções de rede do 5GC para administração e controle dos dados dos usuários), para interconectar na rede de dados - principalmente a Internet.

Figura 10 – Cenário de um ambiente 5G.



Fonte: Adaptado de [1].

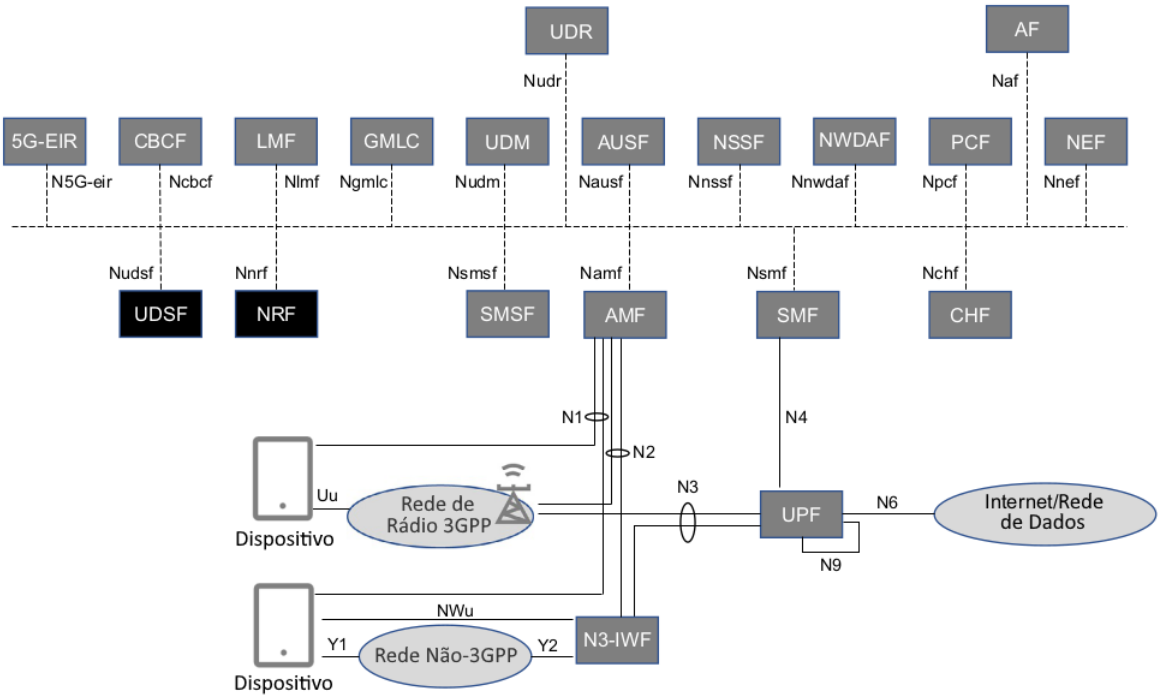
O 5G foi desenvolvido com o suporte das mais recentes tecnologias, tais como, redes definidas por software (SDN - *Software Defined Networks*), virtualização das NFs (NFV - *Network Function Virtualization*), computação em nuvem (*Cloud Computing*), arquitetura baseada em serviços (SBA - *Service-Based Architecture*) e conteinerização.

Desta forma, é possível observar que há a adoção de funcionalidades desenvolvidas em software, em vez de, integradas ao hardware, sendo isso que orienta o desenvolvimento das redes de comunicação e é o caminho natural de evolução. [1].

O ambiente 5G é composto por uma série de NFs do 5GC conforme a parte superior da Figura 11, que apresenta uma perspectiva de um dos possíveis cenários da arquitetura 5G em relação ao plano de usuário. A parte inferior é composta pelo equipamento do usuário, antena, função de plano do usuário (UPF - *User Plan Function*) e a DN da arquitetura. O equipamento do usuário conecta-se em uma antena por sinal de rádio que é padronizado pela 3GPP ou sinais não padronizados pela 3GPP como por exemplo o sinal Wi-Fi. Após a conexão do equipamento do usuário com a antena ou com um sinal não-3GPP, haverá a conexão com o 5GC através da AMF (que será explicado mais a frente interagindo)

principalmente com a SMF que também será explicado mais a frente junto as outras NFs do 5GC.

Figura 11 – Sistema 5G em detalhe com as NFs.



Fonte: Adaptado de [16].

2.5.1 *Core 5G*

O 5GC usa interações baseadas em NFs como serviço. Isso significa que cada NF oferece um ou mais serviços para outras NFs [16]. Na arquitetura 5GC, esses serviços são disponibilizados em interfaces de função de rede conectadas na arquitetura SBA.

Na prática, isso significa que a funcionalidade que possui suporte em uma NF específica é disponibilizada e acessível por meio de uma interface de programação de aplicativo (API - *Application Programming Interface*). Deve-se observar que esta arquitetura se aplica apenas à funcionalidade de sinalização, não à transferência de dados do usuário.

A seguir serão apresentados descrições referentes as NFs consideradas mais relevantes para este trabalho.

2.5.1.1 PCF

Uma peça central do núcleo 5G é a **função de políticas de controle** (*Policy Control Function* - PCF) que interage com várias outras NFs. É essa função que gera as políticas de controle para as funcionalidades relacionadas com o gerenciamento das sessões para o acesso e mobilidade, para a seleção de acesso dos UEs e a seleção de sessões (PDU -

Protocol Data Unit). Além disso, a PCF provê suporte para as transferências em *background* [1]. As funcionalidades da PCF, são portanto, o controle de políticas relacionadas ao usuário, serviços de um usuário e às sessões de dados que podem ser usadas para definir quais tecnologias de acesso de rádio um usuário pode utilizar.

2.5.1.2 AMF

A AMF é a **função de gerenciamento de acesso e mobilidade** que atua no estabelecimento de conexão entre o UE e o 5GC. Essa ação aciona um conjunto de procedimentos para identificar o UE, fornecendo uma estrutura de segurança que oferece um canal de transporte de mensagens. A AMF interage com a rede de rádio e os dispositivos por meio de sinalização nas interfaces N2 e N1 respectivamente. As conexões para todas as outras NFs são gerenciadas via interfaces baseadas em serviços.

A AMF está envolvida na maioria dos fluxos de uma rede 5G. Ela suporta conexões de sinalização criptografadas para dispositivos, permitindo que estes se registrem, sejam autenticados e possam se mover entre diferentes células de rádio na rede. A AMF também oferece suporte para alcançar e ativar dispositivos que estão no modo inativo [1].

2.5.1.3 SMF

A SMF é a **função de gerenciamento de sessão** que tem a responsabilidade de configurar a conectividade do UE para a DN, bem como gerenciar o Plano de Usuário para essa conectividade. Isso inclui estabelecimento, modificação e liberação de sessões individuais e alocação de endereços IP (*Internet Protocol*) por sessão. A SMF se comunica indiretamente com os dispositivos do usuário final, através do qual a AMF encaminha mensagens relacionadas à sessão entre os dispositivos e a SMF [1].

A SMF interage com outras NFs por meio da produção e consumo de serviços em sua interface baseada em serviço, mas também seleciona e controla as diferentes UPFs pela interface de rede N4. Este controle inclui configuração da direção de tráfego e fiscalização de tráfego na UPF para sessões individuais. Além disso, a SMF tem um papel fundamental para todas as funcionalidades relacionadas ao carregamento na rede. Ela coleta seus próprios dados de carregamento e controla a funcionalidade de carregamento na UPF. A SMF suporta a funcionalidade de carregamento offline e online. Além disso, a SMF interage com a PCF para controle de política de sessões do usuário [16].

2.5.1.4 UPF

A UPF é a **função de plano do usuário** e tem como principal tarefa processar e encaminhar os dados do usuário. A funcionalidade da UPF é controlada a partir da SMF. Ela se conecta com redes IP externas e atua como um ponto de ancoragem de IP para os dispositivos em direção a redes externas, escondendo a mobilidade. A UPF realiza vários

tipos de processamento dos dados encaminhados. Ela gera relatórios de uso do tráfego para a SMF que os inclui em relatórios de cobrança para outras NFs [1].

A UPF também pode aplicar "inspeções de pacotes", analisando o conteúdo dos pacotes de dados do usuário para uso como entrada para decisões de política ou como base para o relatórios de uso de tráfego. Ela também é executada em várias políticas de rede ou de usuário, por exemplo, impondo portas e aplicando diferentes limitações de tráfego de dados. Quando um dispositivo está em estado inativo e não pode ser acessado imediatamente pela rede, qualquer tráfego enviado para este dispositivo é armazenado em um *buffer* pela UPF que aciona por exemplo uma página da rede para forçar o dispositivo a voltar ao estado conectado e receber seus dados. A UPF também pode aplicar qualidade de serviço (QoS) de pacotes para a rede de rádio ou para redes externas. Isso pode ser usado pela rede de transporte para tratar cada pacote com a prioridade certa em caso de congestionamento na rede[16].

2.5.1.5 UDR

A UDR é o **repositório de dados unificado**, ou seja, é o banco de dados onde vários tipos de dados são armazenados. Os dados importantes são, naturalmente, os dados de assinatura e os dados que definem vários tipos de políticas de rede ou de usuário. O uso da UDR para armazenar e acessar dados é oferecido como serviço para outras NFs, especificamente a UDM e a PCF. [16].

2.5.1.6 NRF

Para entender a função NRF é preciso primeiramente entender que quando duas NFs se comunicam pela arquitetura 5G, elas assumem duas funções diferentes. A NF que envia a solicitação é o consumidor de serviço, enquanto que a NF que oferece um serviço e dispara alguma ação com base na solicitação é o produtor de serviço [16]. Após a conclusão da ação solicitada, o produtor do serviço responde ao consumidor do serviço.

A parte crítica desse conceito é o mecanismo de como o consumidor de serviço pode localizar e contatar um produtor de serviço que pode fornecer o serviço solicitado. A solução é baseada no conceito de *service discovery*. A descoberta de serviços depende de uma função bem conhecida na rede que controla todos os produtores de serviços disponíveis e quais serviços eles oferecem. Isso é feito por meio de cada produtor de serviço, por exemplo, uma NF 3GPP, como o PCF, registra que seus serviços estão disponíveis.

Na arquitetura 5GC, esse registro é feito em uma NF dedicada, que é chamada de **função de repositório de rede** (NRF). Este conceito permite que a NRF acompanhe todos os serviços disponíveis de todas as NFs na própria rede. Isso também significa que cada NF individual precisa ser configurada com o endereço de um ou mais NRFs, mas não precisa e não deve ter endereços para todas as outras NFs configuradas [16].

2.5.1.7 UDM

A UDM é a **função de gerenciamento de dados unificados**. Ela atua como um *front-end* para os dados de assinatura do usuário armazenados na UDR e executa várias funções a pedido da AMF. A UDM gera os dados de autenticação usados para autenticar dispositivos, autorizando o acesso para usuários específicos com base nos dados de assinatura. Caso haja mais de uma instância AMF e SMF na rede, a UDM controla qual instância está atendendo a qual dispositivo específico [1].

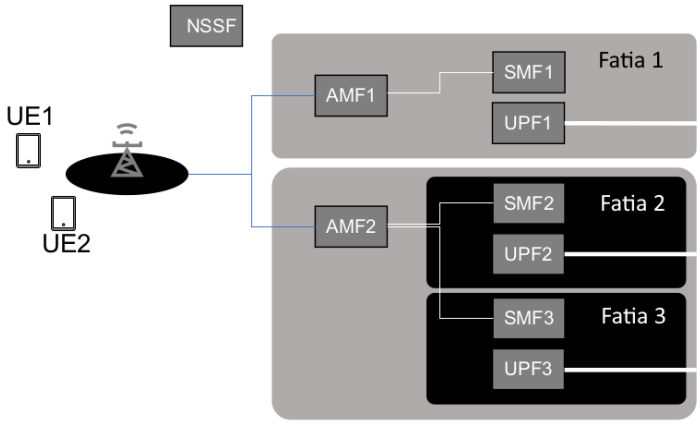
2.5.1.8 AUSF

A AUSF é a **função de servidor de autenticação** que apesar de ser bastante limitada, é muito importante. Ela fornece o serviço de autenticação de um dispositivo específico, em que o processa utilizando as credenciais de autenticação criadas pela UDM [1]. Além disso, a AUSF fornece serviços para a geração de material criptográfico para permitir a segurança, atualizações de informações de forma segura e outros parâmetros do equipamento do usuário [16].

2.5.1.9 NSSF

A NSSF é a **função de seleção de fatia de rede**. Ela tem como única função apoiar a seleção de fatias de rede com base em uma combinação de valores para informações de assistência de seleção de fatia de rede única (S-NSSAI - *Single – Network Slice Selection Assistance Information*) que define a fatia desejada, permitindo-as na assinatura [19]. Na Figura 12 é apresentado um exemplo de fatiamento de rede simplificado, onde o dispositivo 1 (UE1) se conecta à fatia 1 que consiste em uma AMF, SMF e UPF dedicadas. O dispositivo 2 (UE2) se conecta simultaneamente à fatia 2 e à fatia 3, cada uma delas contendo uma SMF e uma UPF, mas ambas sendo servidas por uma AMF2 comum [16].

Figura 12 – Fatiamento de rede.

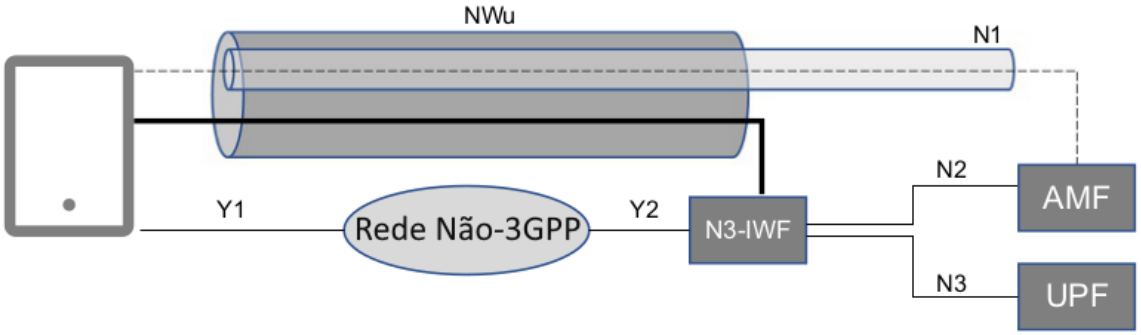


Fonte: Adaptado de [16].

2.5.1.10 N3IWF

A N3IWF é a **função de interoperação não 3GPP** que atua como um *gateway* para a rede móvel e o ponto de conexão para dispositivos que obtêm acesso em uma rede de acesso não 3GPP. Assim, uma vez que se trata de acesso não confiável, a arquitetura não especifica como uma rede de acesso não 3GPP é conectada à arquitetura 5GC, em vez disso, especifica como os dispositivos que utilizam qualquer acesso não 3GPP, que não é confiável, se conectam ao 5GC usando a N3IWF [16]. O tráfego para esses dispositivos poderia, em teoria, ser roteado entre a rede de acesso não confiável e a rede móvel até a Internet. A Figura 13 ilustra a configuração.

Figura 13 – Acesso não 3GPP na UPF e AMF.



Fonte: Adaptado de [16].

Primeiramente, o dispositivo se conecta à rede não 3GPP, após ser autorizado, ele recebe acesso e um endereço IP. Essa conexão é referida como Y1 na imagem, porém, como e quando isso é feito está fora do controle da operadora e nem é especificado pelo 3GPP. Y1 é normalmente uma interface WiFi. O dispositivo seleciona um N3IWF e se conecta a ele usando o serviço de acesso IP oferecido pela rede não 3GPP. Essa conexão é conhecida como Y2 e também não é especificada pelo 3GPP. Conforme explicado acima, a conexão Y2 pode muito bem ser a Internet pública. Em seguida, um túnel de protocolo de segurança de IP (IPsec - *IP Security Protocol*) é estabelecido entre o dispositivo e o N3IWF, por meio do qual tanto a sinalização quanto o tráfego de dados entre o dispositivo e a rede móvel podem ser encaminhados. O N3IWF seleciona uma AMF que em quase todos os casos deve ser a mesma AMF que já está servindo o dispositivo em um acesso 3GPP, se for o caso, uma interface N2 é estabelecida entre N3IWF e a AMF selecionada. Em seguida, uma interface N1 é estabelecida entre o dispositivo e a AMF [16].

2.5.2 Antena 5G

Para atender alguns dos requisitos de capacidade muito alta e altas taxas de dados para os serviços 5G, é necessário utilizar dois conceitos técnicos denominados de múltiplas entradas e múltiplas saídas (MIMO - *Multiple Input and Multiple Output*) e de distribuição de sinais (*Beamforming*). Essas tecnologias podem ser implementadas em redes LTE (4G), mas a nova tecnologia de rádio (NR - *New Radio*) tem funcionalidades mais extensas, incluindo suporte para manipulação de dispositivos que estão em modo ocioso. Isso significa que a sinalização durante a pesquisa de UEs e de solicitações de acesso, pode utilizar *Beamforming* e MIMO [16].

2.5.2.1 Novo Rádio do 5G

Todas as gerações de sistemas móveis digitais definidos pelo 3GPP desde a década de 1990, GSM (2G), WCDMA (3G), LTE (4G) até o NR (5G) suportam conceitos básicos de transmissões digitais para muitos dispositivos em um sistema celular, mas cada geração faz isso com diferentes soluções técnicas, resultando em diferenças na capacidade e características do serviço. Com o NR deve ser possível implantar a transmissão de sinal de rádio em uma faixa muito ampla de bandas de frequência de 450 MHz até 52 GHz. Isso é uma faixa que nenhuma tecnologia de acesso de rádio anterior (2G, 3G ou 4G) suportou [16].

As faixas de frequência são divididas em duas partes:

- FR1 - Faixa de frequência 1, variando de 450 MHz a 6 GHz e normalmente referida como “banda média/baixa”;
- FR2 - Faixa de frequência 2, variando de 24 GHz a 52 GHz e normalmente referida como “banda alta” ou “onda milimétrica” (onda mm).

A seguir é apresentado o significado de algumas das siglas que serão utilizadas mais a frente na Tabela 2:

- TDD, significa que tanto o dispositivo quanto a estação base usam as mesmas frequências ao transmitir, mas são sincronizados para usar diferentes intervalos de tempo para evitar interferência. Isso é normalmente configurado com uma capacidade estática dividida entre o tráfego em DL e em UL, mas pode ser opcionalmente ajustado dinamicamente em células dedicadas onde isso ajuda a otimizar o desempenho.
- FDD (*Frequency-Division Duplex*), significa que o dispositivo e a estação base usam frequências diferentes em suas transmissões. O FDD é suportado apenas nas bandas médias/baixas, não nas bandas altas nas quais o TDD é sempre usado. Esta é uma

consequência da situação regulatória, ou seja, das regras que devem ser seguidas pelos titulares das licenças de espectro.

- SUL e SDL são (*Supplementary Uplink*) e (*Supplementary Downlink*), respectivamente e são bandas usadas para complementar outras bandas, a fim de melhorar a capacidade total e a cobertura do sistema [16].

Assim, na Tabela 2 é apresentado as bandas de frequência suportadas na FR1, onde pode-se notar uma grande quantidade de bandas de frequência com suporte para o uso do NR. E na Tabela 3 uma lista mais curta de bandas de frequência suportadas na FR2. Ressalta-se que o NR suporta os modos duplex TDD e FDD.

Tabela 2 – Bandas de frequência do NR suportadas na FR1.

Bandas de frequência	**Frequências no Up Link**	**Frequências no Down Link**	**Modo Duplex**
n1	1920-1980 MHz	2110-2170 MHz	FDD
n2	1850-1910 MHz	1930-1990 MHz	FDD
n3	1710-1785 MHz	1805-1880 MHz	FDD
n5	824-849 MHz	869-894 MHz	FDD
n7	2500-2570 MHz	2620-2690 MHz	FDD
n8	880-915 MHz	925-960 MHz	FDD
n12	699-716 MHz	729-746 MHz	FDD
n20	832-862 MHz	791-821 MHz	FDD
n25	1850-1915 MHz	1930-1995 MHz	FDD
n28	703-748 MHz	758-803 MHz	FDD
n34	2010-2025 MHz	2010-2025 MHz	TDD
n38	2570-2620 MHz	2570-2620 MHz	TDD
n39	1880-1920 MHz	1880-1920 MHz	TDD
n40	2300-2400 MHz	2300-2400 MHz	TDD
n41	2496-2690 MHz	2496-2690 MHz	TDD
n50	1432-1517 MHz	1432-1517 MHz	TDD
n51	1427-1432 MHz	1427-1432 MHz	TDD
n66	1710-1780 MHz	2110-2200 MHz	FDD
n70	1695-1710 MHz	1995-2020 MHz	FDD
n71	663-698 MHz	617-652 MHz	FDD
n74	1427-1470 MHz	1475-1518 MHz	FDD
n75	———————-	1432-1517 MHz	SDL
n76	———————-	1427-1432 MHz	SDL
n77	3.3-4.2 GHz	3.3-4.2 MHz	TDD
n78	3.3-3.8 GHz	3.3-3.8 MHz	TDD

n79	4.4-5.0 GHz	4.4-5.0 MHz	TDD
n80	1710-1785 MHz	———————-	SUL
n81	880-915 MHz	———————-	SUL
n82	832-862 MHz	———————-	SUL
n83	703-748 MHz	———————-	SUL
n84	1920-1980 MHz	———————-	SUL
n86	1710-1780 MHz	———————-	SUL

Fonte: Adaptado de [16].

Tabela 3 – Bandas de frequência do NR suportadas na FR2.

Bandas de frequência	**Frequências no Up Link**	**Frequências no Down Link**	**Modo Duplex**
n257	26.5-29.5 GHz	26.5-29.5 GHz	TDD
n258	24.25-27.5 GHz	24.25-27.5 GHz	TDD
n260	37-40 GHz	37-40 GHz	TDD
n261	27.5-28.35 GHz	27.5-28.35 GHz	TDD

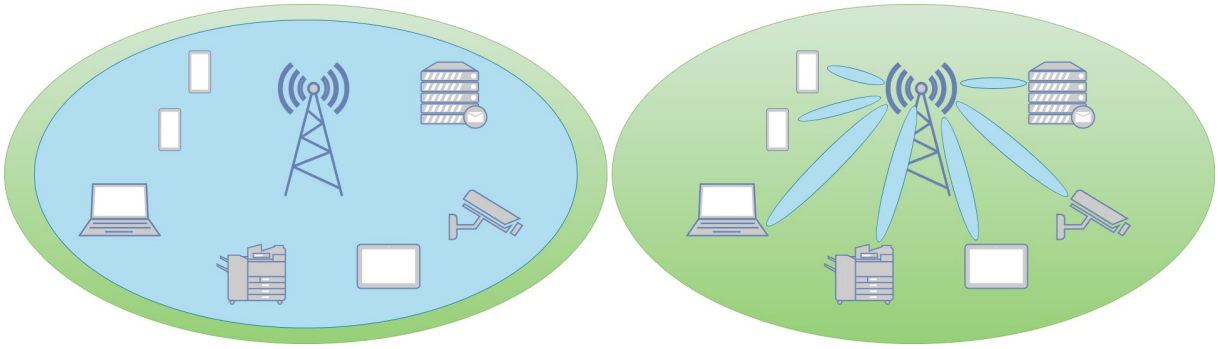
Fonte: Adaptado de [16].

2.5.2.2 *Beamforming*

Beamforming significa que a maior parte da energia transmitida do transmissor é direcionada para o receptor pretendido, em vez de ser espalhada por toda a célula como acontece no feixe único. O receptor também escuta, principalmente, os sinais de rádio vindos da direção do transmissor. Isso melhora a relação sinal-ruído-mais-interferência que é crucial para obter uma maior taxa de transferência de dados. Deve-se notar que em uma implantação típica o suporte para formação de feixe na direção de recepção é mais comum na estação base do que no dispositivo [16].

A técnica de feixe múltiplo (*multi-beam*) significa que existem vários feixes de antena, cada um cobrindo uma parte menor da célula. Esses feixes são dinamicamente controláveis e direcionáveis, o que é usado para maximizar o desempenho por meio da otimização das características na conexão de rádio com cada dispositivo. A Figura 14 ilustra os conceitos de feixe único à esquerda, onde todo o sinal de rádio é disperso pela célula e multi-feixe à direita onde o sinal é focado nos equipamentos do usuário.

Figura 14 – Feixe único e múltiplos feixes.



Fonte: Adaptado de [16].

2.5.2.3 MIMO

MIMO é uma técnica em que o mesmo conteúdo é transmitido simultaneamente na mesma frequência, mas em mais de um caminho de propagação, usando várias antenas ou usando técnicas de formação de feixes. O receptor combina ou seleciona o melhor dos diferentes sinais que recebe para aumentar a potência do sinal recebido. Os sistemas de rádio 5G geralmente combinam essas duas técnicas (MIMO com o *Beamforming*).

MIMO de usuário único (SU-MIMO) significa transmitir duas ou mais cópias do mesmo fluxo de dados em direções ligeiramente diferentes usando *Beamforming*, assim, os sinais de rádio sofrerão alguma perda de energia à medida que passam por vários tipos de materiais, como vidro, madeira, etc. Os sinais serão refletidos, por exemplo, em carros e edifícios localizados entre o transmissor e o receptor.

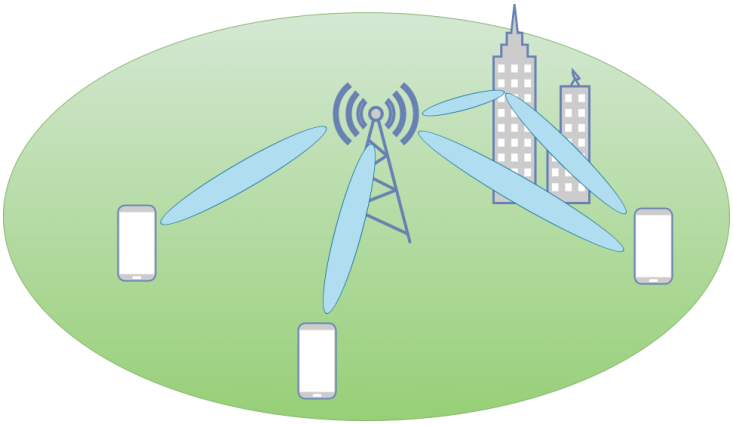
Combinando vários sinais, o receptor irá, portanto, atingir uma relação sinal-ruído-mais-interferência mais alta e, portanto, uma maior taxa de transferência de dados conforme ilustrado na Figura 15, o UE recebe um sinal da antena diretamente e outro sinal que é refletido do prédio, intensificando o sinal recebido final [16].

Figura 15 – Modo SU-MIMO em que o UE recebe sinal de rádio refletido e direto dentro da célula.

Fonte: Adaptado de [16].

Além do SU-MIMO há o MIMO multiusuário (MU-MIMO), em que o objetivo não é otimizar o desempenho para um único usuário, mas sim obter um alto rendimento agregado para vários usuários [16]. Isso é necessário quando a carga está alta na rede e há uma necessidade de otimizar o uso da capacidade geral. Assim, o *Beamforming* é utilizado para se comunicar simultaneamente com dois ou mais usuários nas mesmas frequências, além disso, os usuários precisam estar em diferentes partes da célula para permitir que diferentes feixes de rádio sejam usados, conforme a Figura 16.

Figura 16 – Modo MU-MIMO em que os UEs recebem sinal de rádio refletido e direto dentro da célula.



Fonte: Adaptado de [16].

A alocação das camadas MIMO e a direção dos feixes são continuamente adaptadas à situação e ao uso na célula. Isso não pode ser estático, pois os canais de rádio mudam constantemente conforme os dispositivos e outros objetos, por exemplo, carros se movendo na célula. Para conseguir isso, a estação base e os dispositivos fazem estimativas frequentes das características do canal de rádio que serão usadas pelas estações base para controlar o uso de MIMO e a formação de novos feixes [16].

2.5.3 Equipamento do Usuário

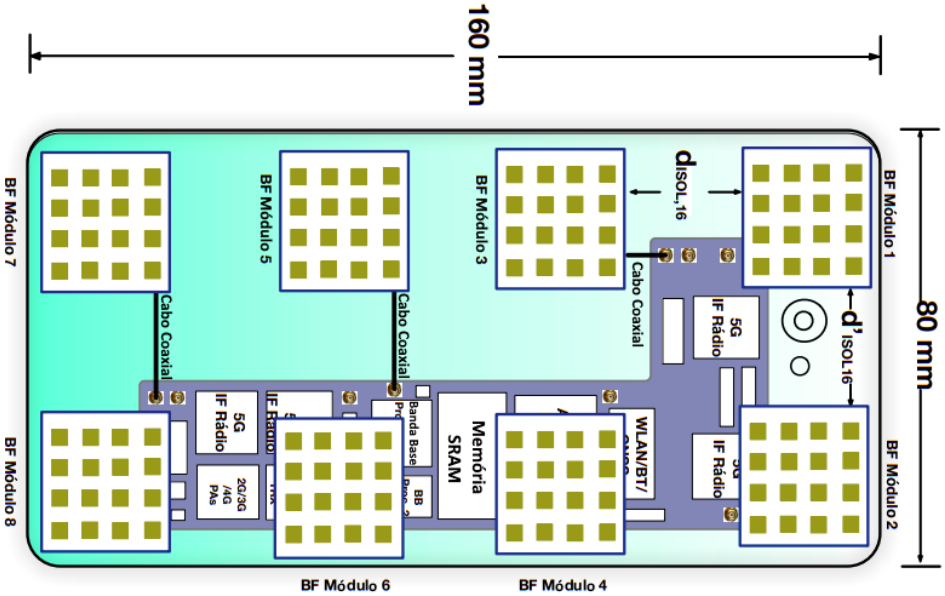
O equipamento do usuário abrange uma variedade de tipos de equipamentos com diferentes níveis de funcionalidade. Estes tipos de equipamentos são referidos como equipamentos de usuário (terminais) e também podem ser compatíveis com uma ou mais interfaces de acesso existentes (fixas ou de rádio). O equipamento do usuário pode incluir um cartão inteligente removível (*SIM Card*) que pode ser usado em diferentes tipos de equipamentos do usuário. Assim, o equipamento do usuário é subdividido em Equipamento Móvel (ME - *Mobile Equipment*) e Módulo de Identidade de Serviços do Usuário (USIM) [20].

Equipamento do usuário é um dispositivo que permite o acesso do usuário aos serviços da rede. Para efeitos das especificações 3GPP, a interface entre o UE e a rede é a

interface de rádio. O ME pode ainda ser subdividido em vários componentes, mostrando a conectividade entre vários grupos funcionais. Esses grupos podem ser implementados em um ou mais dispositivos de *hardware* [20].

A arquitetura de *hardware* interna de equipamentos do usuário são patenteadas e não divulgadas tão facilmente. Assim, para ilustrar uma visão mais detalhada de um dos tipos de equipamento do usuário, segue na Figura 17, um protótipo de celular 5G. Conforme o artigo [21], pode-se notar a distribuição das antenas (BF Módulos) do UE para a implementação de um sistema de matrizes de fase distribuídas baseadas em MIMO (DPA-MIMO - *distributed phased arrays based MIMO*) proposto em [21].

Figura 17 – DPA-MIMO em um aparelho de telefone móvel com $N_{ANT} = 16$.



Fonte: Adaptado de [21].

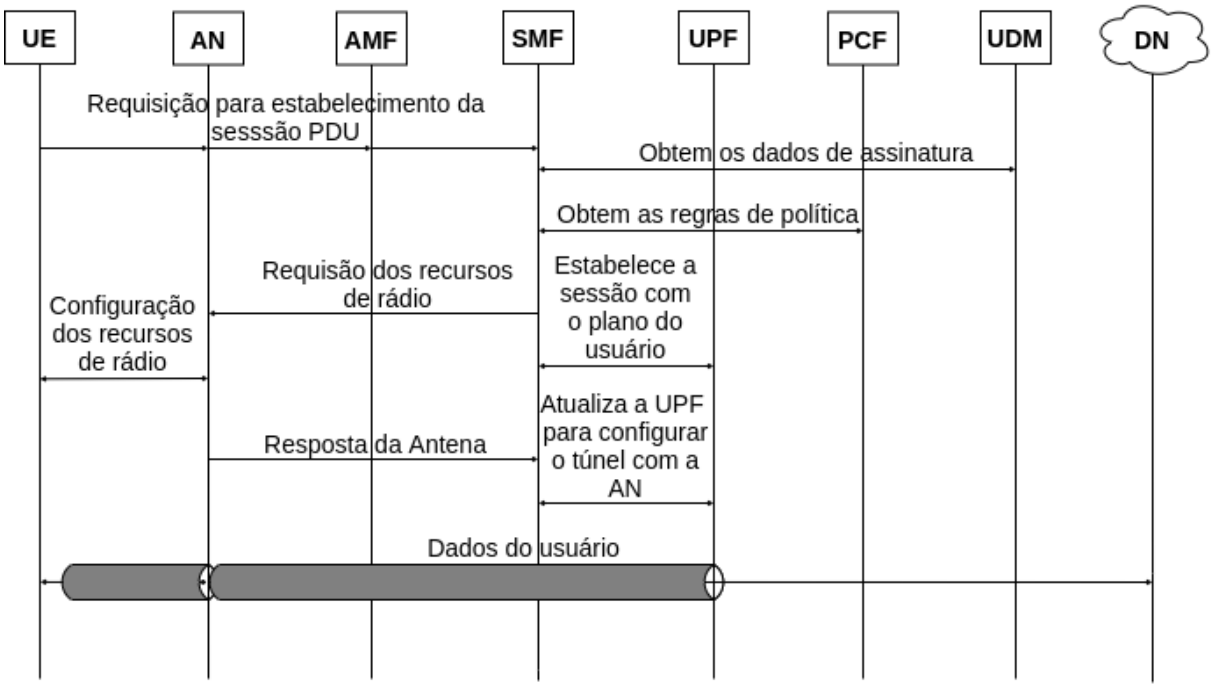
2.5.4 Sessão PDU

Para se conectar a DN, o UE solicita o estabelecimento de uma Sessão PDU, que fornece uma associação entre o UE e uma DN específica. Assim, a sessão PDU está diretamente relacionada com uma das principais tarefas do sistema 5G que é fornecer ao UE conectividade de dados em uma DN. A DN poderia ser a Internet, uma DN específica da operadora ou uma DN dedicada de uma fábrica.

Quando o UE solicita o estabelecimento de uma Sessão de PDU, ela fornece um nome de rede de dados (DNN) que informa a rede principal 5G qual a DN que o UE deseja se conectar. Os DNN usados em uma rede podem ser específicos da operadora ou de uma rede privada (Intranet) [16].

Na Figura 18 é apresentado a sequência de chamadas de estabelecimento da sessão de PDU, destacando as principais NFs envolvidas, bem como as etapas executadas no processo. Durante o estabelecimento da sessão PDU, a conexão no plano de usuário entre o UE e a DN é ativada. A conexão do plano de usuário fornece transporte de unidades de dados de protocolo (PDUs). Assim, PDU é um protocolo básico para o usuário final, que é transportado pela sessão PDU e depende do tipo de sessão PDU. PDUs podem ser, por exemplo, pacotes IP ou quadros Ethernet [16].

Figura 18 – Diagrama do procedimento de estabelecimento da sessão PDU.



Fonte: Adaptado de [16].

No Quadro 3 é descrito os parâmetros adicionais que descrevem as propriedades de uma sessão de PDU, apresentando algumas das mais importantes propriedades de uma sessão PDU. Os parâmetros da sessão PDU são determinados no momento do estabelecimento da sessão PDU e não mudam durante o tempo de vida dela. Além disso, também existem várias "propriedades" adicionais da sessão PDU que podem ser alteradas durante a sua vida útil [16].

Quadro 3 – Principais propriedades que caracterizam uma sessão PDU

	Propriedade da sessão PDU	**Descrição**
1	Identificador de Sessão PDU	Um identificador da sessão de PDU no UE e na rede.
2	Identificador de fatia (S-NSSAI)	Refere-se à fatia da rede em que a sessão PDU é estabelecida.
3	Nome da rede de dados (DNN)	Nome da DN que corresponde a qual sessão PDU fornece conectividade.
4	Tipo de sessão PDU	O tipo de protocolo básico do usuário final transportado pela sessão PDU. Pode ser IPv4, IPv6, IPv4 / IPv6 de pilha dupla, Ethernet ou Não estruturado.
5	Modo de continuidade de serviço e sessão (SSC)	Refere-se à longevidade do ponto de ancoragem do plano do usuário do Sessão PDU, se pode ser realocada ou não.
6	Segurança do plano do usuário Informação de aplicação	Informações que indicam se a cifragem do plano de usuário e/ou a proteção de integridade do plano em uso deve ser ativada para então gerar a sessão PDU.

Fonte: Adaptado de [16]

2.6 Computação de borda

A computação de borda trata de deixar os serviços mais próximos do local onde serão entregues. Os serviços aqui incluem potência de computação e memória necessária para, por exemplo, rodar uma requisição de um aplicativo. A computação de borda, portanto, visa trazer os aplicativos, dados e poder de computação (serviços) que se encontram longe em pontos centralizados (central de dados) para locais mais próximos do usuário (como centrais de dados distribuídas). O objetivo é atingir uma latência mais baixa e reduzir os custos de transmissão e tráfego no Core da rede. Aplicativos que usam grandes volumes de dados e/ou requerem tempos de resposta curtos, por exemplo, jogos de realidade virtual em tempo real, inspeção de qualidade por vídeo na indústria 4.0, carros autônomos, cidades inteligentes etc., são alguns dos candidatos que podem se beneficiar da computação de borda [16].

A 3GPP não especifica nenhuma solução ou arquitetura especial para computação de borda, em vez disso, a 3GPP define várias ferramentas gerais que podem ser usadas para fornecer um eficiente caminho para o usuário. Essas ferramentas não são específicas para computação de borda, mas podem ser usadas como facilitadores na sua implantação [20].

2.6.1 Seleção da UPF pela SMF

A SMF é responsável pela seleção da UPF. Os detalhes de como isso é feito não são padronizados e dependem de vários aspectos, por exemplo, aspectos de implantação relacionados à topologia de rede das UPFs implantadas, bem como, os requisitos do serviço que será entregue.

Quando a SMF faz a seleção de uma UPF, um pré-requisito é que a própria SMF esteja ciente de quais UPFs estão disponíveis em suas respectivas configurações, como recursos da UPF, sequência de carregamento no caso de mais de uma UPF, etc. Uma das formas é que a SMF pode ser configurada com as UPFs disponíveis. Essa configuração pode incluir informações relacionadas à topologia para que a SMF esteja ciente sobre a localização da UPF e de que forma as UPFs estão conectados. Isso permite que a SMF selecione UPFs adequadas, por exemplo, dependendo da localização do UE.

Uma vez que a SMF sabe sobre a(s) UPF(s) disponíveis e há uma necessidade da SMF selecionar uma ou mais UPFs para uma sessão de PDU, como exemplo, no estabelecimento de sessão de PDU ou em algum evento de mobilidade, a SMF pode levar diferentes informações em consideração para selecionar uma UPF. Os detalhes não são padronizados, mas deixados para implementação e configuração do operador [16]. Algumas dessas informações são recebidas da UPF, outras são recebidas da AMF, enquanto que algumas podem ser pré-configuradas na SMF, como as informações relacionadas à topologia do plano do usuário e terminações do plano do usuário.

2.6.2 Formas de classificação de tráfego para a DN

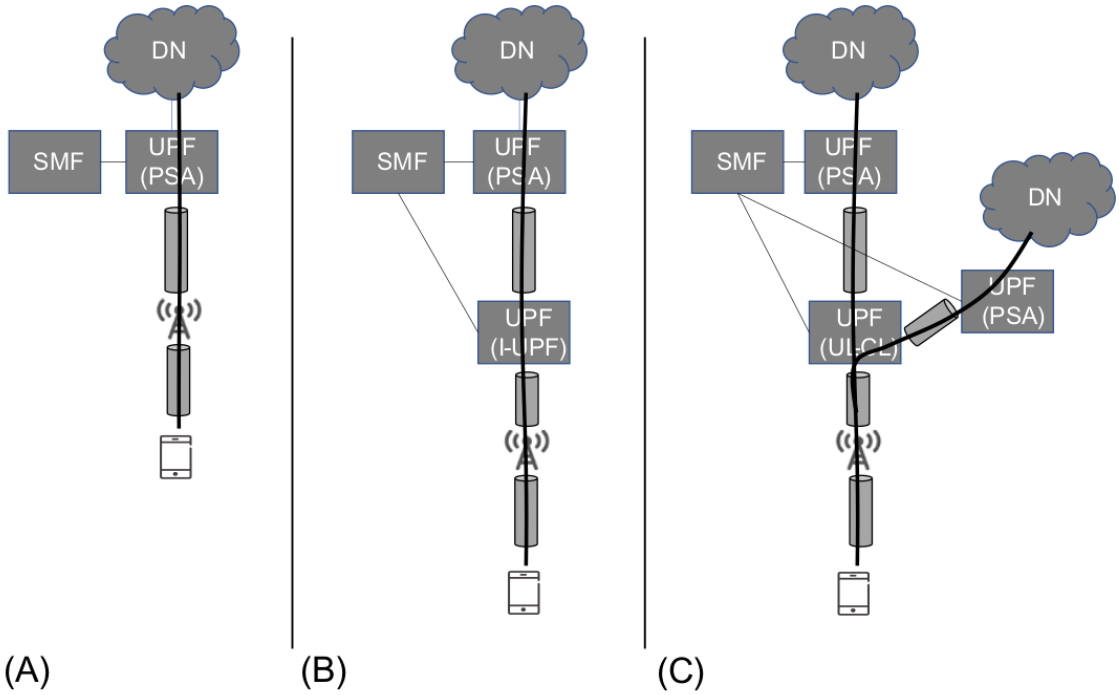
Uma sessão de PDU tem no caso mais simples uma única sessão PDU âncora (PSA - *PDU Session Anchor*) denominada de PSA UPF e, portanto, uma única interface N6 para a DN [16]. Mas uma sessão de PDU também pode ter mais do que uma PSA UPF e, portanto, várias interfaces N6 para uma DN conforme a Figura 19.

1. PSA UPF: Esta é a UPF que faz a conexão com a DN através da interface N6.

2. UPF intermediária (I-UPF): Esta é a UPF que é inserida no caminho do plano do usuário entre a AN e a PSA UPF. Ela encaminha o tráfego entre a AN e o PSA UPF.

3. UPF com classificador de *Up-Link* (UL-CL) ou ponto de ramificação (BP): Esta é uma UPF que está "bifurcando" o tráfego para uma sessão de PDU na conexão ascendente e "mesclando" caminhos Up-Link descendente, fazendo funções relacionadas a QoS.

Figura 19 – Configurações de UPF.



A: PSA único. **B**: PSA + I-UPF. **C**: UL-CL + 2 PSAs.
Fonte: Adaptado de [16]

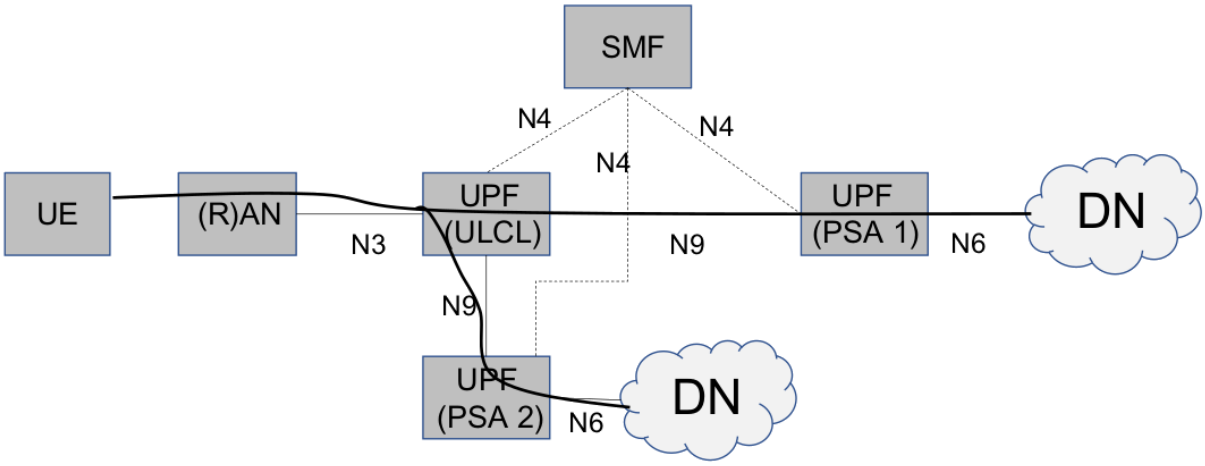
Essa última opção apresentada na Figura 19 pode ser usada para rotear seletivamente o tráfego do plano do usuário para diferentes interfaces N6, por exemplo, rotear de uma PSA UPF com interface N6 para um site periférico local e outra PSA UPF com interface N6 para um *data center* Remoto [16]. Essa funcionalidade é de suma importância para este trabalho, pois pode ser usada em aplicações da computação de borda 5G.

2.6.2.1 Classificação de *Up-Link*

Classificação de Up-Link é uma funcionalidade que é suportada por uma UPF onde a UPF desvia parte do tráfego para uma PSA UPF diferente (local) conforme é apresentado na Figura 20. O UL CL fornece encaminhamento de tráfego de ligação ascendente para diferentes Âncoras de Sessão PDU e desvia o tráfego de ligação descendente para o UE, isto é, o desvio do tráfego de diferentes Âncoras de Sessão PDU na ligação para o UE. O UL CL desvia o tráfego com base nas regras de detecção e encaminhamento de tráfego, fazendo uso de filtros de tráfego fornecidos pela SMF. Assim, o UL CL aplica as regras de filtragem, por exemplo, para examinar o endereço IP de destino dos pacotes IP da conexão ascendente enviados pelo UE e determina como o pacote deve ser encaminhado. A UPF que suporta um UL CL também pode ser controlada pela SMF para oferecer suporte à medição de tráfego e aplicar cobranças. O uso do UL CL se aplica a Sessões de PDU do

tipo IPv4 ou IPv6 ou IPv4v6 ou Ethernet, de modo que a SMF possa fornecer filtros de tráfego [16].

Figura 20 – Acesso local a DN usando o UL CL.



Fonte: Adaptado de [16]

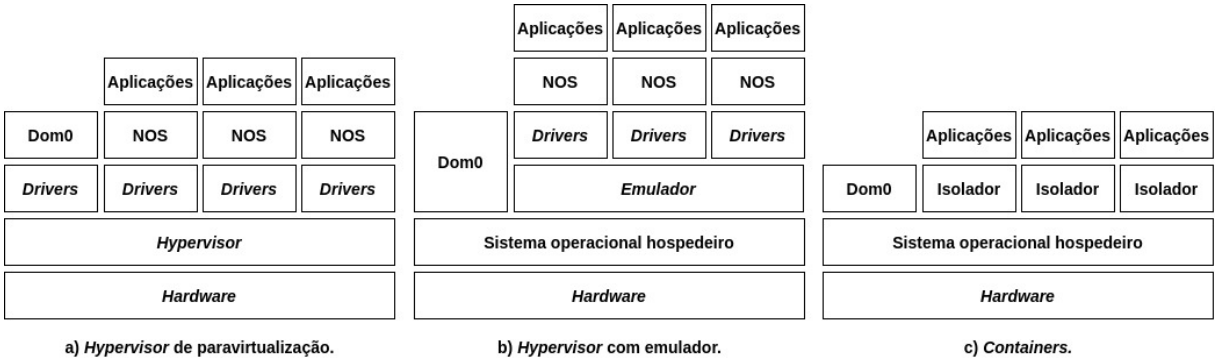
Quando a SMF decide desviar o tráfego, ela insere um UL CL no caminho de dados e uma PSA adicional. Isso pode ser feito a qualquer momento durante a vida útil de uma sessão de PDU. A PSA adicional pode ser colocada na mesma UPF que o UL CL ou pode ser uma UPF autônoma. Quando a SMF determina que o UL CL não é mais necessário, ele pode ser removido do caminho de dados pela SMF [16].

O UE desconhece o desvio de tráfego por parte do UL CL e não participa na inserção e na remoção do UL CL. A solução com o UL CL, portanto, não requer nenhuma funcionalidade específica do UE.

2.7 Virtualização

Existem várias maneiras de colocar máquinas virtuais em equipamentos físicos e podem ser classificadas em três categorias amplas, conforme apresentado na Figura 21. Os dois primeiros itens da Figura 21 correspondem a *hypervisors* e o terceiro item da mesma figura corresponde a *containers* [22]. Na Figura 21 item (a), trata de um *hypervisor* tipo 1 que executa sem auxílio de um SO. Ele pode ou não ser paravirtualizado. Tipicamente, é um misto entre virtualização completa e paravirtualização. Na Figura 21 item (b) é apresentado um *hypervisor* tipo 2. Ele não necessariamente exige emulação, mas que para este caso foi considerado o pior caso a fim de comparação. E por último o item (c) da Figura 21, pode-se notar que é implementado de forma mais simplificada com menos camadas de *software* os *containers*.

Figura 21 – Tipos de arquitetura com *hypervisor*, utilizando apenas *containers*.



Fonte: Adaptado de [22].

O *hypervisor* é um monitor da máquina virtual (VMM) que é frequentemente de código aberto. Os *hypervisors* operam em plataformas de hardware padrão. Além do VMM executado diretamente no hardware físico, a arquitetura geralmente compreende vários domínios executados simultaneamente. Esses domínios executam máquinas virtuais isoladas umas das outras. Cada máquina virtual pode ter seu próprio sistema operacional e aplicativos. O VMM controla o acesso ao hardware de vários domínios e gerencia o compartilhamento dos recursos entre os diferentes domínios. Assim, uma das principais tarefas do VMM é isolar as diferentes máquinas virtuais para que a execução de uma máquina virtual não afete o desempenho das outras [22].

Todos os *drivers* periféricos são mantidos em um domínio isolado específico para eles, conhecido como domínio zero (dom0) que oferece suporte físico confiável e eficaz. O dom0 possui privilégios especiais em comparação a outros domínios, conhecidos como domínios do usuário (domU) e, por exemplo, possui acesso irrestrito ao hardware da máquina física. Os domínios do usuário têm *drivers* virtuais e funcionam como se tivessem acesso direto ao hardware, no entanto, na realidade esses *drivers* virtuais se comunicam com o dom0 para acessar o hardware físico [22].

2.7.1 *Hypervisor* de paravirtualização

Um *hypervisor* de paravirtualização, ou *hypervisor* tipo 1, é um programa executado diretamente em uma plataforma de *hardware* que hospeda máquinas virtuais vinculadas a sistemas operacionais que foram modificados para que as instruções das máquinas virtuais sejam executadas diretamente em uma plataforma de hardware. Esta plataforma é capaz de oferecer suporte a sistemas operacionais convidados com seus *drivers*. Os *hypervisors* clássicos nesta categoria incluem Citrix Xen Server (código aberto), VMware vSphere, VMware ESX, Microsoft Hyper-V Server, Bare Metal e KVM (código aberto) [22].

2.7.2 *Hypervisor* com emulador

A segunda categoria de *hypervisor*, ou *hypervisor* tipo 2, é um programa executado na plataforma de hardware de sistemas operacionais nativos, ou seja, sem realizar nenhuma modificação no sistema nativo. O sistema operacional nativo, quando convidado pelo *hypervisor*, é executado no dispositivo graças a um emulador para que o dispositivo subjacente leve em consideração todas as construções. Os sistemas operacionais convidados não estão cientes que eles são virtualizados, portanto, não requerem nenhuma modificação, ao contrário da paravirtualização. Exemplos deste tipo de virtualização incluem Microsoft Virtual PC, Microsoft Virtual Server, Parallels Desktop, Parallels Server, Oracle VM Virtual Box (gratuito), VMware Fusion, VMware Player, VMware Server, VMware Workstation e QEMU (open source) [22].

2.7.3 Containers

O terceiro tipo deixa para trás os sistemas de *hypervisor* anteriores, executando várias máquinas simultaneamente como *containers*. Nesse caso, falamos de um isolador que é um programa que isola a execução dos aplicativos em um ambiente, chamado de contexto, ou mesmo as zonas de execução. Assim, o isolador é capaz de executar o mesmo aplicativo várias vezes em um modo de várias instâncias. Essa solução tem um desempenho muito bom, pois não causa sobrecarga, porém os ambientes são mais difíceis de isolar.

Em resumo, essa última solução facilita a execução das aplicações nas zonas de execução. Nesta categoria, pode-se citar os exemplos do Linux-Vserver, chroot, BSD Jail e Open VZ e a maioria das soluções de contêiner, como Docker [22].

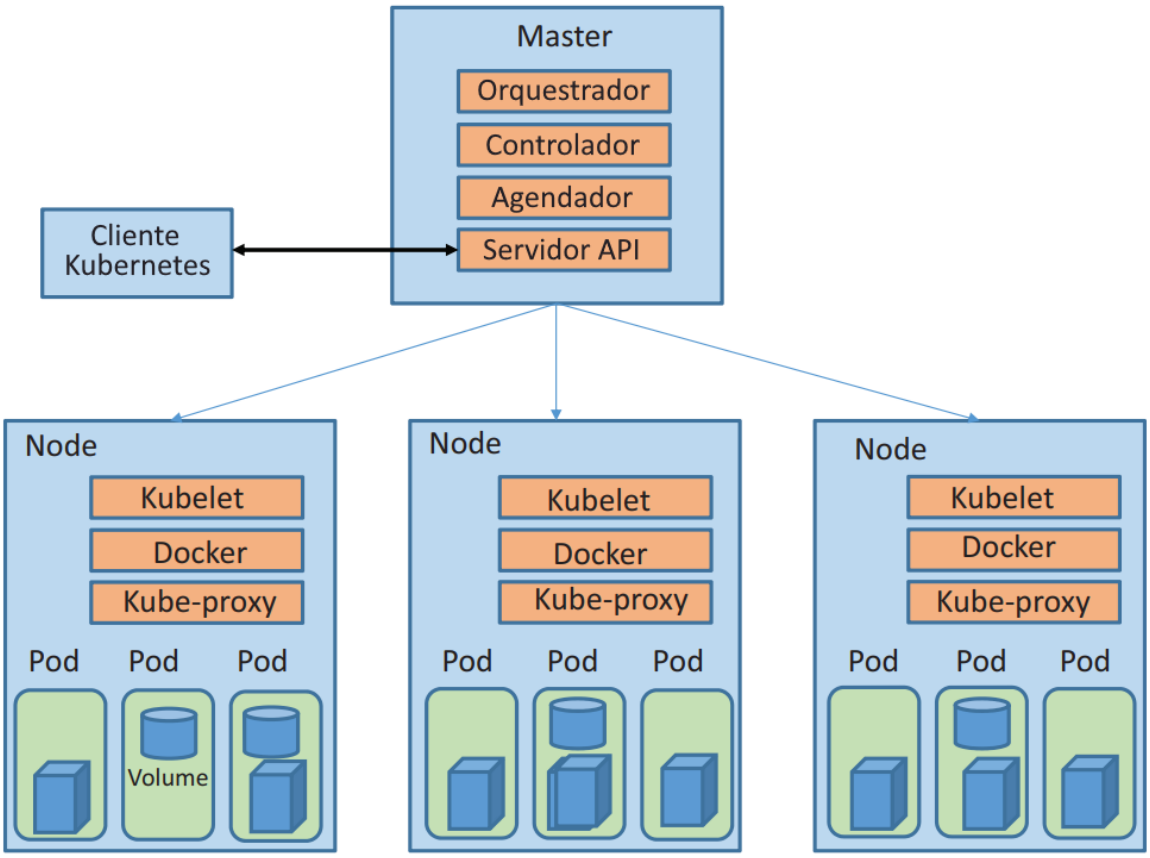
2.7.4 Orquestração com Kubernetes

Kubernetes (K8s) é um sistema de código aberto que permite a implantação, aumento e gerenciamento de aplicativos em *containers*. Essa solução foi criada pela primeira vez pelo Google, que a entregou à Cloud Native Computing Foundation. Esta plataforma permite a automação da implantação, aumento e implementação de *containers* de aplicativos em *clusters* e servidores. Esse software de código aberto funciona com uma ampla gama de tecnologias de *container*, como Docker, por exemplo [22].

A arquitetura do Kubernetes é mostrada na Figura 22. No servidor principal (Master) da Figura 22, encontra-se o orquestrador (Orchestrator), responsável pelo gerenciamento dos Pods. Pods são *containers* ou um grupo de *containers* hospedados por servidores que pertencem a um *cluster* de *hardware*.

O objetivo do programador é compartilhar a carga de trabalho nos servidores, gerenciando assim a execução dos Pods da melhor maneira possível. Por fim, o Kubelet é responsável pelo estado de execução de cada servidor.

Figura 22 – Arquitetura do Kubernetes.



Fonte: Adaptado de [22].

2.8 *Free5GC Compose*

O *Free5GC Compose* [23] é uma versão em Docker Compose da 3° versão do *Free5GC* [24]. O *Free5GC* é mantido pela Universidade Nacional Chiao Tung (NCTU - *National Chiao Tung University*), de Taiwan, bem como, é um projeto de código aberto com licença Apache 2.0 para redes móveis 5G. O objetivo final do *Free5GC* é implementar as NFs do 5GC definidas pela Versão 15 (R15) da 3GPP em diante. Atualmente, o *Free5GC* está na 3° versão que foi entregue em Abril de 2020, disponibilizando novos recursos para suportar serviços como o protocolo de rede que transmite sinais de televisão via internet (IPTV - *Internet Protocol Television*). Assim, o primeiro recurso disponibilizado no estágio 3 é o de operação completa do 5GC com o 5G Orchestrator para poder operar, administrar e gerenciar as NFs [24].

Foram disponibilizados na 3° versão do *Free5GC* os recursos de acesso para dispositivos não 3GPP (N3IWF) e o classificador de Up-link (UL CL). As NFs da 3° versão do *Free5GC* são as seguintes: NSSF, NRF, UDM, PCF, AUSF, AMF, SMF, UPF, WEBUI, N3IWF e o ULCL [24]. A interface gráfica na Web (WEBUI - *User Grafic Interface*), é uma interface

Web disponibilizada pelo *Free5GC* e que foi implementada em um Docker Container no *Free5GC Compose*, assim como, as outras NFs da 3° versão do *Free5GC* também foram implementadas em Containers [23]. A WEBUI permite cadastrar os dispositivos de usuário no Core com uma série de configurações como: Identidade Internacional de Assinante Móvel (IMSI - *International Mobile Subscriber Identity*), S-NSSAI, DNN e configurações de QoS. Podendo através da WEBUI também, visualizar informações da conexão como consumo de dados em tempo real, IP do assinante e entre outras funções.

A plataforma *Free5GC Compose* utiliza o Protocolo de Transferência de HiperTexto (HTTP - *HyperText Transfer Protocol*) para implementar a arquitetura de representação do estado de Transferência (REST - *Representational State Transfer*) [23]. Além disso, o *Free5GC Compose* permite o gerenciamento das NFs do 5GC em containers de forma centralizada, possibilitando que se configure um conjunto de containers através de uma única configuração e que se controle os estados de todos os containers com um único comando.

Para o gerenciamento e controle dos pacotes de dados em nível de camada de transporte, o *Free5GC Compose* necessita do GTP5G pré-instalado.

2.8.1 GTP5G

GTP5G é um módulo de kernel Linux personalizado para lidar com pacotes de dados através do protocolo de controle de encaminhamento de pacotes (PFCP - *Packet Forwarding Control Protocol*). Segundo a 3GPP TS 29.281 [25], o kernel GTP5G é responsável pelas camadas de baixo nível, como a camada de transporte. O protocolo de plano de usuário para tunelamento GPRS (GTP-U - *GPRS tunneling protocol user plane*) é identificado em cada nó com um identificador do terminal do tunnel (TEID - *Tunnel Endpoint Identifier*), um endereço IP e um número de porta UDP. Assim, o GTP-U é necessário para permitir o encaminhamento de pacotes entre entidades GTP-U, conforme a 3GPP TS 29.244 [26].

2.9 *UERANSIM*

O *UERANSIM* [27] é um simulador implementado em código aberto para o UE e a RAN, a qual, também é chamada de próxima geração de estação base (gNB - *Next Generation Node Base*) do 5G. Ele pode ser usado para testar a rede principal 5G e para estudo do sistema 5G.

Além disso, ele foi desenvolvido para atender da Release 15 em diante da 3GPP e tem a licença publica geral GNU v3.0 (GPL-3.0 - *GNU General Public License v3.0*). O UE e a gNB do *UERANSIM* são funcionais e prontos para uso. O *UERANSIM* apresenta a

primeira e única implementação do UE e da gNB 5G de código aberto do mundo. Algumas partes desse software estão com patente pendente [27].

2.10 *Ping*, *Tcpdump*, *iPerf* e *Netperf*

O comando *Ping* utiliza o protocolo de mensagens de controle da Internet (ICMP - *Internet Control Message Protocol*) para testar a conectividade entre equipamentos. Além disso, ele é utilizado para fins de teste, medição e administração de uma rede. Por padrão, o número de bytes de dados enviado pelo *datagrama Echo Request* é 56, mas aumenta para 64 bytes de dados ICMP quando combinado com os 8 bytes de dados do cabeçalho ICMP [28]. No Código Fonte 2.1, segue um exemplo de utilização do comando *Ping* para o IP do Google.

Código Fonte 2.1 – Exemplo de utilização do comando "*Ping*".

```
1 $ ping 8.8.8.8
```

Fonte: O Autor.

O *Tcpdump* é baseado na *libpcap*, uma poderosa API para a captura de pacotes de rede durante o tráfego do pacote. Assim, o *Tcpdump* mostra as conexões estabelecidas e o tráfego formado [29], além disso, pode-se filtrar de acordo com o protocolo desejado, por exemplo, o ICMP, protocolo de pacote de dados do usuário (UDP - *User Datagram Protocol*) e o protocolo de controle de transmissão (TCP - *Transmission Control Protocol*). No Código Fonte 2.2, segue um exemplo de utilização do comando *Tcpdump*, destinado a ficar escutando o tráfego em uma interface de rede específica sem a resolução do DNS e nem de portas, acelerando a exibição dos resultados na tela.

Código Fonte 2.2 – Exemplo utilização do comando "*Tcpdump*".

```
1 $ sudo tcpdump -n -i <interface>
```

Fonte: O Autor.

O *iPerf* é uma programa desenvolvido em três versões, *iPerf*, *iPerf2* e o *iPerf3* para medir a largura de banda entre dois computadores em uma rede local ou na Internet e conta em uma de suas versões, o *iPerf2*, a partir da versão 2.0.14a, a possibilidade de realizar medições de latência [30]. Ele é executado através da linha de comando e tem diversos argumentos para serem configurados e então obter os dados de interesse.

Ao se deparar com um programa cliente-servidor, é necessário executar dois *iPerfs*, um *iPerf* como servidor, por exemplo, em uma VM dentro de um computador e outro *iPerf* como cliente em outro computador, para então, verificar a velocidade entre o *iPerf* cliente e o *iPerf* servidor. Essa ferramenta permite ajustar vários parâmetros relacionados com os protocolos TCP, UDP e o protocolo de controle de transmissão (SCTP - *Stream Control Transmission Protocol*) [31].

No Código Fonte 2.3, segue um exemplo de utilização do comando *iPerf3* para um teste de desempenho específico entre um servidor e um cliente/servidor disponibilizado gratuitamente pelo *iPerf3* [32] para testes com DNS: *iPerf*.scottlinux.com, escutando na porta 5201, localizado na Califórnia EUA. Assim, o comando apresentado no Código Fonte 2.3 é executado como cliente *iPerf3*.

Código Fonte 2.3 – Exemplo de utilização do comando "*iPerf3*".

```
1 $ iperf3 -c iperf.scottlinux.com -p 5201
```

Fonte: O Autor.

O *Netperf* é uma ferramenta para mensurar a latência, *Jitter*, perda de pacotes e *Throutput* utilizando os protocolos UDP e TCP, bem como, é possível utilizar o protocolo SCTP, caso esteja habilitado na instalação [33]. Atualmente, a última versão do *Netperf* disponível é o *Netperf-2.7.0* [34]. Assim como o *iPerf*, o *Netperf* é um programa cliente-servidor e é necessário executar dois comandos do *Netperf*, o Netserver como servidor, e executar o comando *Netperf* em outro computador ou VM como cliente, para então, verificar os dados de interesse entre o cliente e o servidor pré-determinados pelos parâmetros utilizados pelo operador. No Código Fonte 2.4 é apresentado um exemplo de como inicializar o servidor do *Netperf*, através da utilização do comando *netserver*.

Código Fonte 2.4 – Exemplo de utilização do comando "netserver".

```
1 $ netserver
```

Fonte: O Autor.

No Código Fonte 2.5 é apresentado um exemplo de como inicializar o cliente do *Netperf*, através da utilização do comando *Netperf* e utilizando o -H para identificar o IP do servidor, -l para determinar o tempo de duração do teste e -t para declarar o tipo de protocolo utilizado, onde o protocolo TCP_STREAM do *Netperf* é utilizado para medições de Throutput.

Código Fonte 2.5 – Exemplo de utilização do comando "*Netperf*".

```
1 $ netperf -H <IP do servidor> -l 60 -t TCP_STREAM
```

Fonte: O Autor.

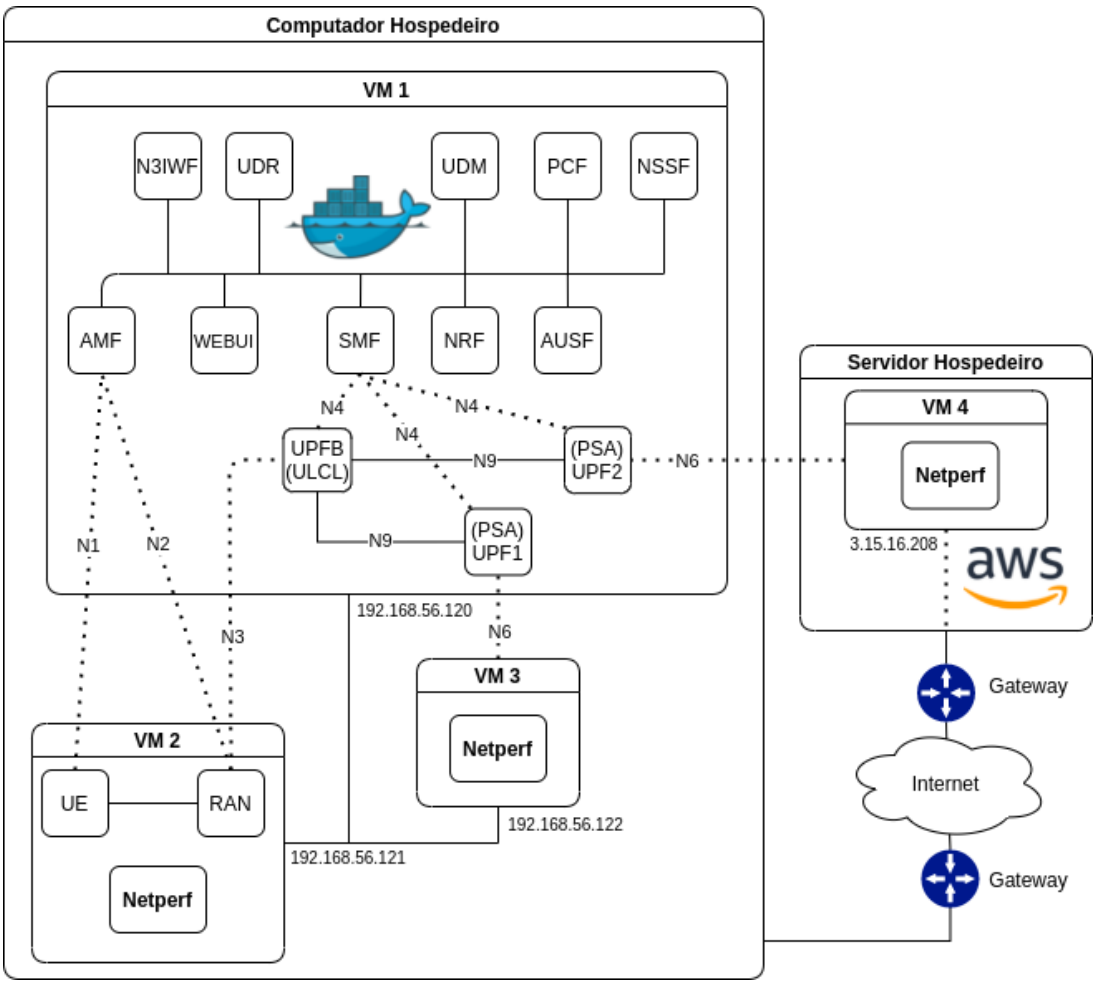
3 Metodologia

Este capítulo tem o intuito de apresentar a metodologia utilizada pelo autor através das ferramentas e conhecimentos necessários para a elaboração de um ambiente de teste virtual de computação de borda 5G.

3.1 Diagrama do ambiente de simulação de computação de borda 5G

Na Figura 23 é apresentado o 5GC instalado e configurado na VM 1. Na VM 2 foi instalado o *UERANSIM*, que é ilustrado através das siglas UE e RAN, bem como, o *Netperf* para ser utilizado como cliente. Nas VMs 3 e 4 encontra-se instalado o Netperf para ser utilizado como servidor de borda e servidor remoto respectivamente.

Figura 23 – Ambiente configurado para a simulação de computação de borda 5G.



Fonte: O Autor.

3.2 Criação das VMs

Para a simulação do ambiente de teste virtual de computação de borda 5G é necessário primeiramente, a criação de VMs. Optou-se em utilizar o VirtualBox (6.1.26) [35], por ser um software gratuito e de uso intuitivo. O VirtualBox foi instalado em um computador hospedeiro, que tem um processador intel i7 de 5° geração com 4 núcleos de processamento, 8 GB de memória de acesso aleatório (Random Access Memory - RAM), 256 GB de armazenamento em disco de estado sólido (SSD - *Solid-State Drive*) e um sistema operacional (SO) Ubuntu Desktop 20.04.

Após a instalação do VirtualBox, foi criada uma VM com o SO Ubuntu Server 20.04 [36], contendo 1 núcleo de processamento, 30 GB de armazenamento e configurada com 2 adaptadores de rede, o primeiro do tipo tradução do endereço da rede (NAT - *Network Address Translation*) e o segundo do tipo apenas adaptador do hospedeiro (*Host-only Adapter*).

Após a criação da VM, algumas verificações importantes foram feitas, por exemplo, o acesso a rede externa utilizando o comando *Ping*. Para a verificação do IP atual da VM utilizou-se o comando $ifconfig$, conforme o Código Fonte 3.1. Vale ressaltar que antes de executar o comando $ifconfig$, deve-se instalar primeiramente um pacote de aplicativos $net-tools$.

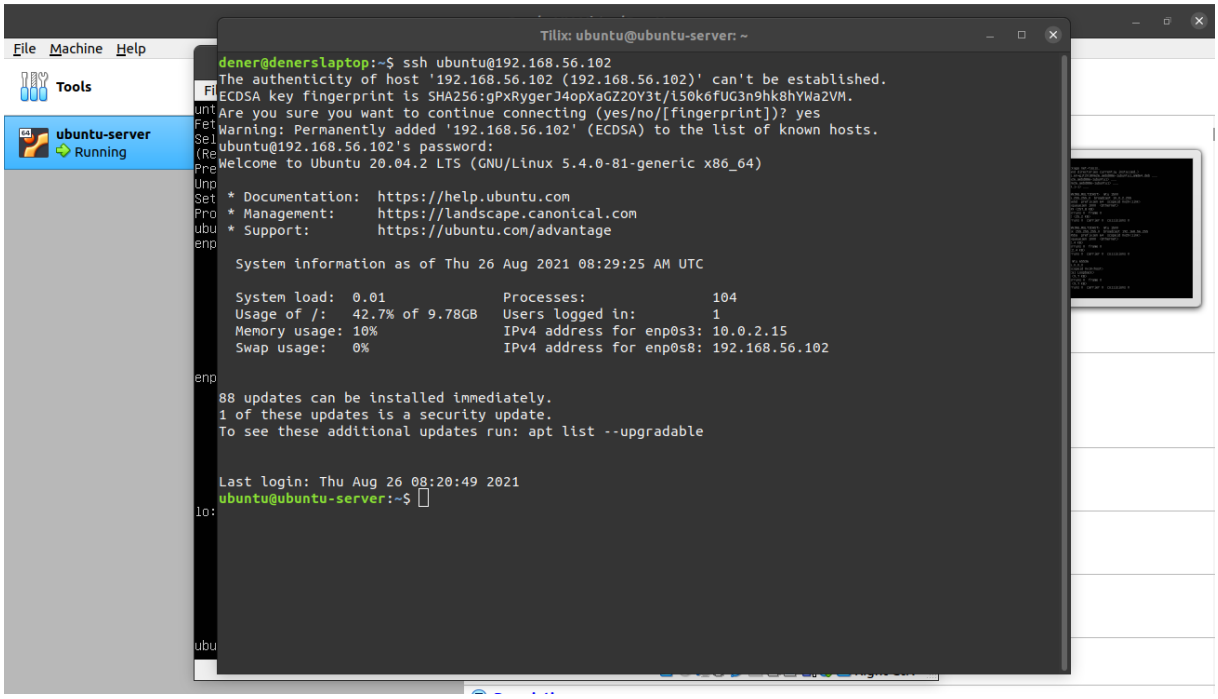
Código Fonte 3.1 – Comandos "*Ping*" e "ifconfig".

```
1 $ ping google.com
2 ...
3 $ sudo apt install net-tools
4 $ ifconfig
5 ...
```

Fonte: O Autor.

Com o IP da VM obtido através do comando $ifconfig$, pode-se acessar a VM utilizando o comando *ssh* (*Secure Shell*) via terminal de comando, conforme a Figura 24.

Figura 24 – Acesso da VM via SSH.



Fonte: O Autor.

Em seguida após acessar a VM, ela foi atualizada conforme o Código Fonte 3.2, no qual está listado os comandos utilizados para realizar o *update* e em seguida o *upgrade*. Por fim, é utilizado o comando *shutdown* para desligar a VM, conforme o Código Fonte 3.2.

Código Fonte 3.2 – Comandos *Update* e *Upgrade* aplicados na VM.

```
1 $ sudo apt update
2 ...
3 $ sudo apt upgrade
4 ...
5 $ sudo shutdown -P now
6 ...
```

Fonte: O Autor.

Para a instalação e configuração do *Free5gc Compose*, *UERANSIM* e Servidor *Netperf* de borda que serão explicados com mais detalhes nas próximas seções, é necessário clonar a VM criada e configurada anteriormente.

A ferramenta de clonagem do VirtualBox foi utilizada. Assim, foi criado uma VM para o *Free5gc Compose*, outra para o *UERANSIM* e uma para o Servidor *Netperf* de borda, em seguida, elas foram iniciadas e configuradas, conforme os Códigos Fonte 3.3, 3.4, 3.5 e 3.6.

A VM clonada ainda tem o nome da antiga VM. Assim, no Código Fonte 3.3 é apresentado o comando para editar o arquivo *hostname*, e então renomear a VM.

Código Fonte 3.3 – Alteração do hostname.

```
$ sudo nano /etc/hostname  # ou sudo vi /etc/hostname
free5gc-compose
```

Fonte: O Autor.

Em seguida, foi feita a alteração do nome de domínio completamente qualificado (FQDN - *Fully Qualified Domain Name*) para "free5gc-compose", conforme o Código Fonte 3.4.

Código Fonte 3.4 – Alteração do FQDN em hosts.

```
$ sudo nano /etc/hosts
127.0.0.1 localhost
127.0.1.1 free5gc-compose
...
```

Fonte: O Autor.

Para cada VM, foi desabilitado o "dhcp4" e adicionado um IPv4 estático, por exemplo, "192.168.56.120" com CIDR "/24" para a VM do *Free5gc Compose*, conforme o Código Fonte 3.5.

Código Fonte 3.5 – Desabilitar dhcp4 e adicionar IPv4 estático.

```
$ sudo nano /etc/netplan/00-installer-config.yaml
network:
  ethernets:
    enp0s3:
      dhcp4: true
    enp0s8:
      dhcp4: false
      addresses: [192.168.56.120/24]
  version: 2
```

Fonte: O Autor.

Após isso, foi checado se o IPv4 estático é válido com o comando *netplan try* e em seguida utilizado o *netplan apply* caso seja um IPv4 válido pelo primeiro comando. Por fim, foi utilizado o comando *ifconfig* para verificar se foi alterado o IPv4 e, ainda, a VM foi reiniciada utilizando o comando *shutdown* junto as *flags -r now*, conforme o Código Fonte 3.6.

Código Fonte 3.6 – Comandos "netplan try" e "netplan apply".

```
$ sudo netplan try
...
```

```
Press Enter ...
...
$ sudo netplan apply
...
$ ifconfig
...
$ sudo shutdown -r now
```

Fonte: O Autor.

Em seguida foi repetindo o processo de clonagem da primeira VM para as outras duas VMs de acordo com as configurações do Quadro 4.

A quarta VM, necessária para simular o Servidor *Netperf* Remoto, foi criada na plataforma da *Amazon Web Services* - AWS, uma plataforma na nuvem que disponibiliza de forma gratuita e limitada alguns de seus serviços para testes ou em versões pagas de acordo com a necessidade do usuário. Para este trabalho, optou-se em utilizar a modalidade gratuita da plataforma AWS referente ao serviço de criação de VMs, o *Elastic Compute Cloud* (EC2) [37], com limitações de escolha do SO e de recursos de hardware devido a modalidade escolhida. Foi criado uma VM com o máximo de recursos liberados por essa modalidade que pode ser utilizada por até um ano e que tem as especificações apresentadas no Quadro 4 a VM 4.

Quadro 4 – Configurações das VMs utilizadas no ambiente de computação de borda 5G.

VM	**Aplicação**	**VM IP/FQDN ou DNS**	**SO**	**Memória RAM**	**Núcleos**	**Armazenamento em disco**
1	**Free5GC Compose**	192.168.56.120/ free5gc-compose	Ubuntu Server 20.04	2048 MB	1 un.	30 GB
2	***UERANSIM*** **e** **Cliente *Netperf***	192.168.56.121/ ueransin-netperf-cliente	Ubuntu Server 20.04	1024 MB	1 un.	30 GB
3	**Servidor *Netperf*** **Borda**	192.168.56.122/ netperf-servidor-borda	Ubuntu Server 20.04	1024 MB	1 un.	30 GB
4	**Servidor *Netperf*** **Remoto na AWS**	3.15.16.208/ ec2-3-15-16-208.us-east-2.compute.amazonaws.com	Ubuntu Server 20.04	1024 MB	1 un.	30 GB

Fonte: O Autor.

O acesso da VM 4 é feito via comando ssh, utilizando um arquivo de chave privada disponibilizado pela Amazon. Tal acesso é permitido com a chave privada no computador em que vai acessar a VM que está na AWS. No Código Fonte 3.7 é apresentado como foi feito o primeiro acesso da VM 4.

Código Fonte 3.7 – Accesso da VM 4 remota.

```
$ ssh -i "<caminho do arquivo '.pem'>" ubuntu@3.15.16.208
Welcome to Ubuntu 20.04 ...
```

Fonte: O Autor.

3.3 Utilização do *Free5gc Compose*

Para a simulação dos serviços do 5GC, escolheu-se o programa *Free5gc Compose*, o qual foi instalado a partir do repositório do github [23] na VM 1. Além disso, para o correto funcionamento das NFs do *Free5gc Compose*, o módulo de *kernel* GTP5G foi instalado.

Cada *container* das NFs implementado no *Free5gc Compose* tem pré configurado um DNS de um IP local privado para facilitar a comunicação das NFs e facilitar na identificação de cada NF durante a configuração. Assim, realizou-se a verificação dos DNSs, bem como foi alterado o DNS da UPFB referente a interface N6 localizado no arquivo de configurações da SMF do *Free5gc Compose* para o IP da VM 1.

No passo seguinte, adicionou-se comandos no arquivo de configuração do Docker Compose, conforme o Código Fonte 3.8 para realiza a tradução dos endereços que passam pela UPFB e para habilitar o roteamento de pacotes, apresentados nas linhas 11 e 12. A porta 2152 na linha 20 do Código Fonte 3.8 foi aberta para a UPFB e a porta 38412 foi aberta para a AMF. As portas 2152 e 38412 são utilizadas pelos protocolos de transporte UDP e SCTP, respectivamente e o protocolo de encapsulamento GTP-U utiliza-se da porta 2152 e da pilha UDP/IP para o estabelecimento do túnel e transporte de dados do usuário durante a Sessão PDU. Além disso, para a NRF, AMF, AUSF, NSSF, PFC, SMF, UDM e a UDR, optou-se em manter a porta 8000 referente ao protocolo HTTP, bem como, para a WEBUI manteve-se a porta 5000.

Código Fonte 3.8 – Trecho do arquivo de executação das FNs.

```
$sudo nano docker-compose.yaml
...
free5gc-upf-b:
    container_name: upfb
    build:
      context: ./nf_upf
      args:
        DEBUG_TOOLS: "false"
    command: |
            /bin/bash -c "
            sysctl -w net.ipv4.ip_forward=1
            iptables -t nat -A POSTROUTING -o eth0 -j
  MASQUERADE
```

```
13              ./free5gc-upfd -f ../config/upfcfg.yaml
14              "
15      volumes:
16        - ./config/upfcfgb.yaml:/free5gc/config/upfcfg.yaml
17      cap_add:
18        - NET_ADMIN
19      ports:
20        - "2152:2152/udp"
21      networks:
22        privnet:
23          aliases:
24            - upfb.free5gc.org
25 ...
```

Fonte: O Autor.

Para a UPF1 e UPF2 não foi necessário abrir nenhuma porta e as mesmas configurações apresentadas na UPFB foram reutilizadas para elas, porém, adicionou-se um comando a mais que é apresentado no Código Fonte 3.9. Este comando é utilizado para aceitar o roteamento de todos os pacotes que passam pela UPF1 e UPF2.

Código Fonte 3.9 – Comando iptable para roteamento de pacotes.

```
1 iptables -I FORWARD 1 -j ACCEPT
```

Fonte: O Autor.

Ressalta-se que a NRF, UDM, UDR e a WEBUI utilizam o MongoDB para o armazenamento de dados, sendo ele um banco de dados orientado a documentos, diferente dos bancos de dados tradicionais que seguem o modelo relacional. Para o MongoDB, também optou-se em manter aberto a porta pré-configurada 27017. Como também, optou-se por desabilitar os recursos disponibilizados pela N3IWF. Essa NF, que serve como interface para as redes não 3GPP, como por exemplo, Wi-Fi, não será utilizada pois o escopo desse trabalho limita-se a rede 5G, especificamente para a computação de borda.

3.3.1 Configuração do UL CL

Para configurar o UL CL, o *Free5gc* disponibiliza um arquivo de configuração com o nome *uerounting*, conforme o Código Fonte 3.10. Nele foi possível configurar o SUPI do UE com o IP do destino que se deseja ativar o UL CL, por exemplo, quando o usuário tentar efetuar um *Ping* para o IP 3.15.16.208 os pacotes vão ser redirecionados para a UPF1.

Código Fonte 3.10 – Arquivo de configuração das rotas.

```
$ sudo vi config/uerouting.yaml
info:
  version: 1.0.0
  description: Routing information for UE
ueRoutingInfo:
  - SUPI: imsi-208930000000003
      - DestinationIP: 3.15.16.208
        UPF: !!seq
          - BranchingUPF
          - AnchorUPF1
```

Fonte: O Autor.

Por fim, para a UPF1 saber o que fazer com os pacotes recebidos que tenham o IP 3.15.16.208 em seu cabeçario, mais um comando iptables foi adicionado nas configurações do Docker Compose referentes ao container da UPF1, conforme o Código Fonte 3.11. Este comando tem como objetivo trocar o endereço de destino (3.15.16.208) dos pacotes para um novo endereço o (192.168.56.122).

Código Fonte 3.11 – Comando para tradução de IPs.

```
iptables -t nat -A PREROUTING -d 3.15.16.208 -j DNAT --to
   192.168.56.122
```

Fonte: O Autor.

Após as configurações, pode-se inicializar o Core através do comando *sudo docker-compose up* e espera-se que não aconteça nenhuma falha. O comando *sudo docker-compose down* pode ser utilizado para remover os containers das NFs após suspender o Core, pressionando o comando (ctrl+c). Em seguida, pode ser verificado na WEBUI as informações de identificação do equipamento de usuário no Core: Rede móvel terrestre pública (PLMN - *Public Land Mobile Network*), Identificador Permanente de Assinatura ou Identidade do assinante móvel internacional (SUPI - *Subscription Permanent Identifier*)/IMSI, método de autenticação, chave K, tipo de código de operadora, código de operadora, entre outras configurações. Optou-se em manter todas as configurações padrões, por exemplo, o SUPI:208930000000003.

3.4 Utilização do *UERANSIM*

Para simular o registro e tráfego de dados de um dispositivo de usuário no Core foi utilizado o *UERANSIM*, que primeiramente foi instalado a partir do repositório disponibilizado no github [27]. Em seguida foi alterado o IP de configuração referente ao local

da AMF, no arquivo de configuração referente ao gNB do *UERANSIM* adicionando neste arquivo o IP da VM 1 e no mesmo arquivo de configuração foi adicionado o IP da VM 2, para definir o local do UE e do gNB em uma única VM.

Outras configurações foram comparadas entre o arquivo de configuração do UE e gNB do *UERANSIM* com as configurações da WEBUI do *Free5gc Compose*.

Após as devidas verificações, a plataforma *UERANSIM* está pronta para realizar o registro de um UE simulado no 5GC, através de um acesso via rádio também simulado, inicializando uma sessão PDU através de um túnel entre UE e UPF. Assim, é esperado que o acesso via rádio simulado se comunique efetivamente com o Core e permita a criação da interface N1 que liga o UE diretamente a AMF, interface N2 que liga o RAN e a AMF e a interface N3 que liga a UPFB e o RAN.

3.5 *iPerf* ou *Netperf*

Como nem todas as ferramentas são implementadas e configuradas da mesma forma, ferramentas diferentes podem retornar resultados diferentes. Assim, segundo a Google Cloud e pesquisadores da AT&T, o *Netperf* retorna a melhor resposta para a medição de latência [38], desse modo, optou-se em utilizá-lo em vez do *iPerf*. Outro motivo é que o *iPerf* além de necessitar de uma instalação personalizada [30], também necessita realizar a sincronia de *clock* entre cliente e servidor, a fim de obter uma medição de latência coerente com a realidade devido ao modo em que foi implementado.

No Código Fonte 3.12 e Código Fonte 3.13 encontram-se os comandos utilizados na tentativa de mensurar a latência utilizando o *iPerf*. Como pode-se ver no Código Fonte 3.13 é apresentado valores que não condizem com a realidade.

Código Fonte 3.12 – Comando do *iPerf* utilizado no cliente.

```
1 $ iperf --bind 60.60.0.1 -c 192.168.56.122 -p 5001 -i 1 -e --
    trip-times
2 ...
```

Fonte: O Autor.

Código Fonte 3.13 – Comando do *iPerf* utilizado nos servidores.

```
1 $ iperf -s -e -i 1
2 ...
3 [ ID] Interval        Transfer    Bandwidth    Burst Latency
    avg/min/max/stdev ...
4 [  1] 0.00-1.00 sec   479 MBytes  4.02 Gbits/sec
    -903.226/-908.975/-882.718/14.811 ms ...
5 ...
```

Fonte: O Autor.

3.6 Análises com o *Ping*, *Tcpdump* e *Netperf*

Durante toda a elaboração do ambiente virtual, uma ferramenta muito utilizada é o comando *Ping*, pois com ele pode-se verificar se uma configuração de rede foi efetuada da forma correta com um simples *Ping*, por exemplo, entre as VMs ou de uma VM para a Internet.

Em paralelo ao comando *Ping*, a utilização do comando *Tcpdump* é indispensável, devido a praticidade e desempenho do programa em verificar se as trajetórias realizadas pelos pacotes de dados estão corretas ou não. Por exemplo, pode-se verificar com o comando *Tcpdump* se os pacotes de dados do protocolo ICMP gerados pelo comando *Ping* estavam passando através do túnel gerado pela sessão PDU. O mesmo procedimento pode ser feito com o comando *Tcpdump* para verificar o caminho percorrido pelos pacotes de dados dos protocolos UDP ou TCP gerados pelo comando *Netperf*. Além disso, pode-se verificar se os pacotes estavam passando pela UPF1 quando desejado ou passando pela UPF2 de acordo com as configurações de rotas pré-estabelecidas no Core.

Para a medição de desempenho foi instalado o programa *Netperf-2.7.0* [34] com a opção –enable-spin ativada na VM 2, VM 3 e VM 4. Na VM 2, o *Netperf* é utilizado como Cliente e na VM 3 e VM 4 ele é utilizado como Servidor. Assim, após estabelecer a sessão PDU entre o UE na VM 2 com a VM 3 ou entre VM 2 e VM 4, pode-se realizar as medições de latência, velocidade da transferência de arquivos (Throughput), perda de pacotes e *Jitter* para, por exemplo, os protocolos UDP e TCP.

4 Resultados

Neste capítulo são apresentados os resultados obtidos através de Imagens, Gráficos, Códigos Fontes e Quadros com o intuito de esclarecer detalhes sobre a topologia utilizada para a simulação de computação de borda 5G.

4.1 Carregamento do Core 5G e acesso na WEBUI

Após todo o preparo do ambiente de simulação de computação de borda 5G, executou-se na VM, o Core através do comando *docker-compose up*, conforme Código Fonte 4.1.

Ressalta-se que inicialmente o Core foi carregado sem as configurações do UL CL para a obtenção de dados referentes aos protocolos ICMP, TCP e UDP. Em uma segunda execução do Core foi habilitado o UL CL conforme explicado na Subseção 3.3.1 para comparar os dados obtidos com os dados obtidos sem o UL CL.

Código Fonte 4.1 – Carregamento das NFs.

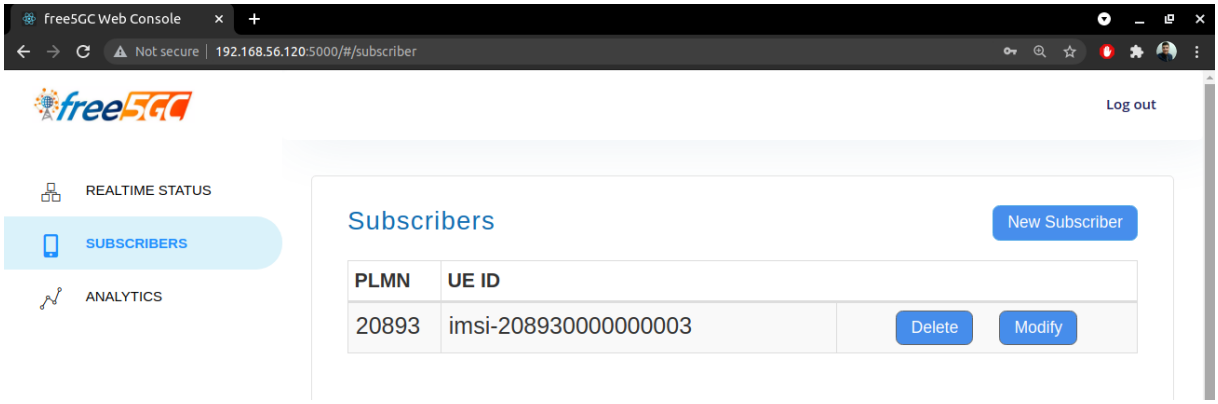
```
1  $ sudo docker-compose up
2  Creating network "free5gc-compose_privnet" with the default
       driver
3  Creating upf2    ... done
4  Creating mongodb ... done
5  Creating upf1    ... done
6  Creating upfb    ... done
7  Creating webui   ... done
8  Creating nrf     ... done
9  Creating amf     ... done
10 Creating udr     ... done
11 Creating ausf    ... done
12 Creating nssf    ... done
13 Creating pcf     ... done
14 Creating udm     ... done
15 Creating smf     ... done
16 Attaching to upf2, mongodb, upf1, upfb, nrf, webui, nssf, pcf
       , udr, ausf, amf, smf, udm
17 ...
```

Fonte: O Autor.

Em sequência, logo após o término do carregamento de todas as NFs do Core, pode-se acessar a WEBUI e cadastrar um novo usuário. Ressalta-se que optou-se em não alterar

nenhuma das configurações padrões pré-configuradas na ficha de inscrição da WEBUI. Na Figura 25 é apresentado o UE já cadastrado na WEBUI do *Free5GC* implementada em *container* no *Free5GC Compose*.

Figura 25 – WEBUI do *Free5GC* implementada em *container* no *Free5GC Compose*.



Fonte: O Autor.

4.2 Estabelecimento da sessão PDU

Após o cadastramento do UE no Core hospedado na VM 1 através da WEBUI, é realizado na VM 2, o estabelecimento da conexão entre o Core e a antena, mais especificamente entre a AMF e o gNB através do protocolo SCTP conforme o Código Fonte 4.2.

Código Fonte 4.2 – Carregando o simulador gNB.

```
1 $ build/nr-gnb -c config/free5gc-gnb.yaml
2 UERANSIM v3.2.0
3 [2021-09-12 11:12:01.133] [sctp] [info] Trying to establish
    SCTP connection... (192.168.56.120:38412)
4 [2021-09-12 11:12:01.138] [sctp] [info] SCTP connection
    established (192.168.56.120:38412)
5 ...
6 [2021-09-12 11:12:01.147] [ngap] [info] NG Setup procedure is
     successful
```

Fonte: O Autor.

Em seguida é realizado o estabelecimento da sessão PDU, na qual o simulador do UE pré-configurado no Core inicia o túnel *uesimtun0* na VM 2, conforme o Código Fonte 4.3.

Código Fonte 4.3 – Carregando o simulador UE.

```
1 $ sudo build/nr-ue -c config/free5gc-ue.yaml
2 UERANSIM v3.2.0
```

```
3 ...
4 [2021-09-12 11:12:44.048] [nas] [info] PDU Session
    establishment is successful PSI[1]
5 [2021-09-12 11:12:44.090] [app] [info] Connection setup to
    PDU session[1] is successful, TUN interface[uesimtun0,
    60.60.0.1] is up.
```

Fonte: O Autor.

Como pode-se observar no Código Fonte 4.3, a Sessão PDU foi estabelecida com sucesso e o túnel *uesimtun0* recebeu o IP 60.60.0.1.

Com o comando *docker logs*, pode-se observar o comportamento do Core e as operações por ele realizadas durante o carregamento de forma individual para cada NF, sendo possível verificar mensagens de erro caso ocorram. No Código Fonte 4.4 é apresentado de forma parcial os logs da SMF, no qual é possível observar que houve o carregamento correto das configurações pré-estabelecidas em relação as rotas e que a SMF identificou e configurou a UPFB, UPF1 e UPF2.

Código Fonte 4.4 – Comando docker logs smf.

```
1 $ sudo docker logs smf
2 2021-09-13T10:31:49Z [INFO][SMF][App] SMF version:
3   free5GC version: v3.0.5
4   build time:      2021-07-17T23:06:07Z
5   commit hash:     04c01ec5
6   commit time:     2021-01-30T17:01:30Z
7   go version:      go1.14.4 linux/amd64
8 ...
9 2021-09-13T10:31:49Z [INFO][SMF][PFCP] UPF(10.100.200.2)[
    internet] setup association
10 2021-09-13T10:31:49Z [INFO][SMF][PFCP] UPF(10.100.200.5)[
    internet] setup association
11 2021-09-13T10:31:49Z [INFO][SMF][PFCP] UPF(10.100.200.4)[
    internet] setup association
```

Fonte: O Autor.

4.3 Dados obtidos utilizando ICMP

Com o estabelecimento da Sessão PDU feito, pode-se realizar testes de desempenho. Mas antes disso, na VM 2 pode-se observar o tráfego através do túnel *uesimtun0*, utilizando a ferramenta *Tcpdump* que foi configurada para capturar apenas o protocolo ICMP e para obter os pacotes que passam apenas através da interface *uesimtun0*, bem como,

para não resolver nomes (imprimir apenas IPs de origem e destino), conforme o Código Fonte 4.5, no qual é apresentado o tráfego de pacotes gerado entre cliente e servidor de borda. Ressalta-se que para obter o tráfego entre cliente e servidor remoto foi utilizado o mesmo comando.

Código Fonte 4.5 – Comando *Tcpdump* para o protocolo icmp utilizado.

```
1 $ sudo tcpdump -n -i uesimtun0 icmp
2 listening on uesimtun0, link-type RAW (Raw IP), capture size
    262144 bytes
3 12:20:43.853974 IP 60.60.0.1 > 192.168.56.122: ICMP echo
    request, id 21, seq 1, length 64
4 12:20:43.855571 IP 192.168.56.122 > 60.60.0.1: ICMP echo
    reply, id 21, seq 1, length 64
5 ...
```

Fonte: O Autor.

Para confirmar que o tráfego além de estar passando pelo túnel ele está passando pelo Core, foi utilizado o comando *Tcpdump* para obter o tráfego na VM 1 e assim, poder verificar o comportamento dos pacotes passando pelo Core, conforme o Código Fonte 4.6.

Código Fonte 4.6 – Comando *Tcpdump* para o protocolo icmp usado na VM 1 com o UL CL habilitado.

```
1 $ sudo tcpdump -n -i any icmp
2 listening on any, link-type LINUX_SLL (Linux cooked v1),
    capture size 262144 bytes
3 13:47:57.958896 IP 10.100.200.5 > 192.168.56.122: ICMP echo
    request, id 6, seq 1, length 64
4 13:47:57.958896 IP 10.100.200.5 > 192.168.56.122: ICMP echo
    request, id 6, seq 1, length 64
5 13:47:57.958913 IP 192.168.56.120 > 192.168.56.122: ICMP echo
     request, id 6, seq 1, length 64
6 13:47:57.959446 IP 192.168.56.122 > 192.168.56.120: ICMP echo
     reply, id 6, seq 1, length 64
7 13:47:57.959461 IP 192.168.56.122 > 10.100.200.5: ICMP echo
    reply, id 6, seq 1, length 64
8 13:47:57.959465 IP 192.168.56.122 > 10.100.200.5: ICMP echo
    reply, id 6, seq 1, length 64
9 ...
```

Fonte: O Autor.

Assim, nota-se que o Core (192.168.56.120) devido a configuração da interface N6 adicionada na UPFB (10.100.200.5) no arquivo de configuração da SMF, realiza uma requisição para a VM 2 (192.168.56.122) e em seguida a VM 2 responde ao Core.

Em seguida foi utilizado a ferramenta *Ping* com o objetivo de coletar os dados de latência, *Jitter*, perda de pacotes e verificar o se houve estresse de *hardware*, observando-se o consumo de CPU na VM2 durante a execução do *Ping*. O comando *Ping* foi configurado com um intervalo de 10 ms, duração de 1 minuto (6000 pacotes), ou seja, um pacote a cada 10ms, enviando os pacotes através do túnel *uesimtun0*, conforme Código Fonte 4.7 entre o cliente e o servidor de borda com o UL CL desligado. Ressalta-se que desta forma pode-se exigir um maior esforço de *hardware* e, assim, tornar uma comparação justa [38] com os dados obtidos mais a frente, utilizando os protocolos UDP e TCP nos gráficos que serão gerados.

Código Fonte 4.7 – Comando *Ping* entre o cliente e o servidor de borda.

```
1 $ sudo ping 192.168.56.122 -I uesimtun0 -c 6000 -i 0.01
2 PING 192.168.56.122 (192.168.56.122) from 60.60.0.1 uesimtun0
    : 56(84) bytes of data.
3 64 bytes from 192.168.56.122: icmp_seq=1 ttl=62 time=1.25 ms
4 64 bytes from 192.168.56.122: icmp_seq=2 ttl=62 time=1.12 ms
5 ...
6 64 bytes from 192.168.56.122: icmp_seq=5999 ttl=62 time=0.763
     ms
7 64 bytes from 192.168.56.122: icmp_seq=6000 ttl=62 time=1.98
    ms
8
9 --- 192.168.56.122 ping statistics ---
10 6000 packets transmitted, 6000 received, 0% packet loss, time
     63193ms
11 rtt min/avg/max/mdev = 0.671/1.325/4.328/0.442 ms
```

Fonte: O Autor.

A partir do comando *Ping* executado, pode-se perceber que não houve perda de pacotes, a latência média foi de 1.325 ms com um *Jitter* de 0.442 ms e o uso de CPU médio observado durante a execução do comando foi de 86% pelo processo *Ping* na VM 2.

Ainda na VM 2, realizou-se o comando *Ping* entre o cliente e o servidor remoto, conforme o Código Fonte 4.8.

Código Fonte 4.8 – Comando *Ping* entre o cliente e o servidor remoto.

```
1 $ sudo ping 3.15.16.208 -I uesimtun0 -c 6000 -i 0.01
2 PING 3.15.16.208 (3.15.16.208) from 60.60.0.1 uesimtun0:
    56(84) bytes of data.
```

```
64 bytes from 3.15.16.208: icmp_seq=1 ttl=59 time=163 ms
64 bytes from 3.15.16.208: icmp_seq=2 ttl=59 time=166 ms
...
64 bytes from 3.15.16.208: icmp_seq=5999 ttl=59 time=167 ms
64 bytes from 3.15.16.208: icmp_seq=6000 ttl=59 time=165 ms

--- 3.15.16.208 ping statistics ---
6000 packets transmitted, 6000 received, 0% packet loss, time
    84258ms
rtt min/avg/max/mdev = 152.982/162.626/196.101/2.971 ms, pipe
    15
```

Fonte: O Autor.

Neste caso observou-se que também não houve perda de pacotes, tendo uma latência média de 162.626 ms, *Jitter* de 2.971 ms e o uso de CPU médio observado durante a execução do comando foi de 4% pelo processo *Ping* na VM 2.

4.3.1 Dados obtidos utilizando ICMP com o UL CL habilitado

Para a obtenção dos dados utilizando ICMP com o UL CL habilitado foi feita as devidas configurações conforme apresentado na seção 3.3.1 e em seguida foi utilizado a ferramenta *Tcpdump* para analisar se o tráfego está passando pelo túnel *uesimtun0*, conforme apresentado no Código Fonte 4.10.

Código Fonte 4.9 – Comando *Tcpdump* para o protocolo icmp escutando na VM 2 com o UL CL habilitado.

```
$ sudo tcpdump -n -i uesimtun0 icmp
listening on uesimtun0, link-type RAW (Raw IP), capture size
    262144 bytes
13:48:01.388690 IP 60.60.0.1 > 192.168.56.122: ICMP echo
    request, id 6, seq 1, length 64
13:48:01.390941 IP 192.168.56.122 > 60.60.0.1: ICMP echo
    reply, id 6, seq 1, length 64
...
13:48:01.398695 IP 60.60.0.1 > 192.168.56.122: ICMP echo
    request, id 6, seq 2, length 64
13:48:01.400346 IP 192.168.56.122 > 60.60.0.1: ICMP echo
    reply, id 6, seq 2, length 64
```

Fonte: O Autor.

Para confirmar que o tráfego além de estar passando pelo túnel ele está passando pelo Core, pode-se também obter o tráfego na VM 1 para verificar o comportamento dos pacotes passando pelo Core, conforme o Código Fonte 4.10.

Código Fonte 4.10 – Comando *Tcpdump* para o protocolo ICMP usado na VM 1 com o UL CL habilitado.

```
1 $ sudo tcpdump -n -i any icmp
2 listening on any, link-type LINUX_SLL (Linux cooked v1),
    capture size 262144 bytes
3 13:47:57.958896 IP 10.100.200.5 > 192.168.56.122: ICMP echo
    request, id 6, seq 1, length 64
4 13:47:57.958896 IP 10.100.200.5 > 192.168.56.122: ICMP echo
    request, id 6, seq 1, length 64
5 13:47:57.958913 IP 192.168.56.120 > 192.168.56.122: ICMP echo
     request, id 6, seq 1, length 64
6 13:47:57.959446 IP 192.168.56.122 > 192.168.56.120: ICMP echo
     reply, id 6, seq 1, length 64
7 13:47:57.959461 IP 192.168.56.122 > 10.100.200.5: ICMP echo
    reply, id 6, seq 1, length 64
8 13:47:57.959465 IP 192.168.56.122 > 10.100.200.5: ICMP echo
    reply, id 6, seq 1, length 64
9 ...
```

Fonte: O Autor.

Assim, nota-se que o Core (192.168.56.120) de igual forma observado com o UL CL desabilitado, ele realiza uma requisição para a VM 2 (192.168.56.122) e em seguida a VM 2 responde ao Core com o UL CL habilitado. Para uma análise mais completa foi observado o tráfego na VM 1 referente aos pacotes UDP também, conforme o Código Fonte 4.11.

Código Fonte 4.11 – Comando *Tcpdump* para o protocolo UDP usado na VM 1 com o UL CL habilitado.

```
1 $ sudo tcpdump -n -i any udp
2 listening on any, link-type LINUX_SLL (Linux cooked v1),
    capture size 262144 bytes
3 15:05:53.824068 IP 192.168.56.121.2152 > 192.168.56.120.2152:
     UDP, length 100
4 15:05:53.824116 IP 192.168.56.121.2152 > 10.100.200.5.2152:
    UDP, length 100
5 15:05:53.824125 IP 192.168.56.121.2152 > 10.100.200.5.2152:
    UDP, length 100
```

```
15:05:53.824176 IP 192.168.56.120.2152 > 10.100.200.2.2152:
    UDP, length 100
15:05:53.824197 IP 192.168.56.120.2152 > 10.100.200.2.2152:
    UDP, length 100
15:05:53.824662 IP 10.100.200.2.2152 > 10.100.200.5.2152: UDP
    , length 92
15:05:53.824690 IP 10.100.200.2.2152 > 10.100.200.5.2152: UDP
    , length 92
15:05:53.824708 IP 10.100.200.5.2152 > 192.168.56.121.2152:
    UDP, length 92
15:05:53.824708 IP 10.100.200.5.2152 > 192.168.56.121.2152:
    UDP, length 92
15:05:53.824721 IP 192.168.56.120.2152 > 192.168.56.121.2152:
     UDP, length 92
```

Fonte: O Autor.

Neste último comando *Tcpdump* utilizado na análise para ICMP, pode-se oberservar a comunicação feita entre o Core e a antena para realizar a transmissão dos pacotes ICMP. Assim, a VM 2 (192.168.56.121) se comunica com o Core (192.168.56.120) pela interface N6 e também com a UPFB pelas interfaces N3 e N9. Em seguida, o Core estabelece a comunicação com a UPF1 (10.100.200.2) que responde para a UPFB (10.100.200.5) e a UPFB junto ao Core respondem ao "cliente" (VM 2).

Por fim, foi obtido os dados com a ferramenta *Ping* com o UL CL habilitado, conforme o Código Fonte 4.12.

Código Fonte 4.12 – Comando *Ping* entre o cliente e o servidor remoto.

```
$ sudo ping 3.15.16.208 -I uesimtun0 -c 6000 -i 0.01
PING 3.15.16.208 (3.15.16.208) from 60.60.0.1 uesimtun0:
    56(84) bytes of data.
64 bytes from 3.15.16.208: icmp_seq=1 ttl=62 time=1.45 ms
64 bytes from 3.15.16.208: icmp_seq=2 ttl=62 time=1.61 ms
...
64 bytes from 3.15.16.208: icmp_seq=5999 ttl=62 time=1.76 ms
64 bytes from 3.15.16.208: icmp_seq=6000 ttl=62 time=1.48 ms

--- 3.15.16.208 ping statistics ---
6000 packets transmitted, 6000 received, 0% packet loss, time
    60111ms
rtt min/avg/max/mdev = 0.805/1.317/2.962/0.458 ms
```

Fonte: O Autor.

Neste ultimo comando *Ping* executado, pode-se perceber que não houve perda de pacotes, a latência média foi de 1.317 ms com um *Jitter* de 0.458 ms e o uso de CPU médio observado durante a execução do comando foi de 87% pelo processo *Ping* na VM 2.

4.4 Dados obtidos utilizando UDP

Nesta Seção é apresentado os dados utilizando o *Netperf* para mensurar a latência, *Jitter*, perca de pacotes e *Throughput* com o protocolo UDP. No Código Fonte 4.13 é apresentado como foi inicializado o *netserver*. Este comando serve tanto para UDP como para TCP e ele foi utilizado na VM 3 e na VM 4, servidor de borda e servidor remoto respectivamente. Ressalta-se que para a obtenção destes dados a função UL CL foi desabilitada.

Código Fonte 4.13 – Comando utilizado para iniciar o servidor *Netperf*.

```
1 $ sudo netserver
```

Fonte: O Autor.

Em seguida é executado o comando *Tcpdump*, conforme o Código Fonte 4.14 na VM 2 para o tráfego de pacotes UDP que passam na interface *uesimtun0* entre cliente e servidores.

Código Fonte 4.14 – Comando *Tcpdump* utilizado para UDP.

```
1 $ sudo tcpdump -n -i uesimtun0 udp
2 listening on uesimtun0, link-type RAW (Raw IP), capture size
    262144 bytes
3 14:28:43.436240 IP 60.60.0.1.36489 > 192.168.56.122.59884:
    UDP, length 1
4 14:28:43.437613 IP 192.168.56.122.59884 > 60.60.0.1.36489:
    UDP, length 1
5 ...
```

Fonte: O Autor.

Ressalta-se que os pacotes também passaram pelo Core, onde foi realizado a mesma analise com a ferramenta *Tcpdump* na VM 1 conforme apresentado na Seção 4.3.

Em seguida, na VM 2 é executado o comando *Netperf*, conforme o Código Fonte 4.15. Neste comando é definido o IP do servidor (192.168.56.122), IP da interface *uesimtun0* que será transmitido os pacotes (60.60.0.1), quantidade de pacotes enviados (6000), tipo de protocolo (UDP_RR), tamanho dos pacotes (1024 bytes) e intervalo de tempo de envio entre pacotes (10 ms).

Código Fonte 4.15 – *Netperf* entre o cliente e o servidor de borda UDP.

```
1 $ netperf -H 192.168.56.122 -L 60.60.0.1 -l 60 -t UDP_RR -w
    10ms -b 1 -v 2 -- -O min_latency,mean_latency,max_latency,
    stddev_latency,transaction_rate
2 MIGRATED UDP REQUEST/RESPONSE TEST from 60.60.0.1 () port 0
    AF_INET to 192.168.56.122 () port 0 AF_INET : spin
    interval : first burst 0
3 ^CMinimum      Mean          Maximum      Stddev
    Transaction
4 Latency      Latency       Latency      Latency       Rate
5 Microseconds Microseconds  Microseconds Microseconds  Tran/s
6
7 1042         1415.18       38301        528.63        99.941
```

Fonte: O Autor.

Para o UDP, na conexão entre cliente e servidor de borda, obteve-se uma latência média de 1.415 ms e um *Jitter* de 0.528 ms com o uso de CPU da VM 2 em 87.5% pelo processo *Netperf*.

No Código Fonte 4.16 é apresentado os resultados de latência entre cliente e servidor remoto, em que a única alteração feita no comando é a do IP do servidor remoto (3.15.16.208).

Código Fonte 4.16 – *Netperf* entre o cliente e o servidor remoto UDP.

```
1 $ netperf -H 3.15.16.208 -L 60.60.0.1 -l 60 -t UDP_RR  -w 10
    ms -b 1 -v 2 -- -O min_latency,mean_latency,max_latency,
    stddev_latency,transaction_rate
2 MIGRATED UDP REQUEST/RESPONSE TEST from 60.60.0.1 () port 0
    AF_INET to 3.15.16.208 () port 0 AF_INET : spin interval :
     first burst 0
3 ^CMinimum      Mean          Maximum      Stddev
    Transaction
4 Latency      Latency       Latency      Latency       Rate
5 Microseconds Microseconds  Microseconds Microseconds  Tran/s
6
7 166524       171436.50     186989       232.50        0.024
```

Fonte: O Autor.

A latência média obtida entre cliente e o servidor remoto foi de 171.436 ms e não foi observado o uso excessivo de CPU pelo processo *Netperf* tanto na VM 2 como na VM 4.

O último dado coletado em relação ao UDP é o *Throughput* que também foi obtido a partir do *Netperf*, conforme o Código Fonte 4.17. No comando apresentado é identificado o IP do servidor que neste caso é o servidor de borda (192.168.56.122), em seguida o IP do túnel (60.60.0.1), o tempo de duração do teste (5 s), tipo de protocolo utilizado (UDP_STREAM), formato desejado para apresentar o resultado do *Throughput* (Mb/s) e o tamanho da mensagem (1400 MB/s).

Código Fonte 4.17 – *Throughput* entre cliente e servidor de borda UDP.

```
1 netperf -H 192.168.56.122 -L 60.60.0.1 -l 5 -t UDP_STREAM -f
    m -- -R 1 -m 1400
2 MIGRATED UDP STREAM TEST from 60.60.0.1 () port 0 AF_INET to
    192.168.56.122 () port 0 AF_INET : spin interval
3 Socket  Message  Elapsed      Messages
4 Size    Size     Time         Okay Errors   Throughput
5 bytes   bytes    secs            #      #   10^6bits/sec
6
7 212992    1400   5.00       221428      0     495.97
8 212992           5.00        50197            112.44
```

Fonte: O Autor.

O *Throughput* médio obtido entre o cliente e o servidor de borda foi de 112.44 Mb/s recebido pelo servidor local e foi observado o uso excessivo de CPU, tendo um pico de 160% pelos processos *nr-ue* e *nr-gnb* na VM 2, causando em algumas vezes falhas e perda do estabelecimento da sessão PDU. Em seguida, foi utilizado o mesmo comando do caso anterior, alterando apenas o IP do servidor de destino para mensurar o *Throughput* entre cliente e servidor remoto, conforme Código Fonte 4.18.

Código Fonte 4.18 – *Throughput* entre cliente e servidor remoto UDP.

```
1 $ netperf -H 3.15.16.208 -L 60.60.0.1 -l 5 -t UDP_STREAM -f m
    -- -R 1 -m 1400
2 MIGRATED UDP STREAM TEST from 60.60.0.1 () port 0 AF_INET to
    3.15.16.208 () port 0 AF_INET : spin interval
3 Socket  Message  Elapsed      Messages
4 Size    Size     Time         Okay Errors   Throughput
5 bytes   bytes    secs            #      #   10^6bits/sec
6
7 212992    1400   5.00       164537      0     368.54
8 212992           5.00        19400             43.45
```

Fonte: O Autor.

O *Throughput* médio obtido entre cliente e o servidor remoto foi de 43.45 Mb/s recebidos pelo servidor remoto e também foi observado o uso excessivo de CPU, tendo um pico de 140% pelos processos *nr-ue* e *nr-gnb* na VM 2, causando novamente em algumas vezes falhas e perda do estabelecimento da sessão PDU.

4.5 Dados obtidos utilizando TCP

Nesta Seção é apresentado os dados obtidos através do *Netperf* para mensurar a latência, *Jitter*, perca de pacotes e *Throughput* com o protocolo TCP. O comando para inicializar os servidores de borda e remoto foi o mesmo utilizado para o UDP. Este comando foi utilizado na VM 3 e na VM 4, servidor de borda e servidor remoto respectivamente.

Em seguida é executado o comando *Tcpdump* na VM 2 para ficar escutando o tráfego de pacotes TCP que passam na interface *uesimtun0*, conforme o Código Fonte 4.19. Ressalta-se que o mesmo comando foi utilizado para obter o tráfego entre cliente e servidor remoto.

Código Fonte 4.19 – Comando *Tcpdump* para TCP.

```
1 $ sudo tcpdump -n -i uesimtun0 tcp
2 17:34:05.947059 IP 60.60.0.1.35261 > 3.15.16.208.40779: Flags
    [.], ack 1, win 65280, length 0
3 17:34:05.948069 IP 60.60.0.1.35261 > 3.15.16.208.40779: Flags
    [P.], seq 1:2, ack 1, win 65280, length 1
4 17:34:05.950509 IP 3.15.16.208.40779 > 60.60.0.1.35261: Flags
    [.], ack 2, win 65535, length 0
5 17:34:06.118970 IP 3.15.16.208.40779 > 60.60.0.1.35261: Flags
    [P.], seq 1:2, ack 2, win 65535, length 1
6 ...
```

Fonte: O Autor.

Ressalta-se que os pacotes também passaram pelo Core, onde foi realizado a mesma analise com a ferramenta *Tcpdump* na VM 1 conforme apresentado na Seção 4.3.

Em seguida, na VM 2 também é executado o comando *Netperf*, conforme o Código Fonte 4.20. Neste comando em relação ao utilizado para o UDP, foi alterado apenas o tipo de protocolo que desta vez é utilizado o TCP_RR.

Código Fonte 4.20 – *Netperf* entre cliente e o servidor de borda TCP.

```
1 $ netperf -H 192.168.56.122 -L 60.60.0.1 -l 60 -t TCP_RR -w
    10ms -b 1 -v 2 -- -O min_latency,mean_latency,max_latency,
    stddev_latency,transaction_rate
2 MIGRATED TCP REQUEST/RESPONSE TEST from 60.60.0.1 () port 0
    AF_INET to 192.168.56.122 () port 0 AF_INET : spin
    interval : first burst 0
```

```
3 Minimum      Mean          Maximum      Stddev
     Transaction
4 Latency      Latency       Latency      Latency      Rate
5 Microseconds Microseconds  Microseconds Microseconds Tran/s
6
7 852          1562.48       29694        752.60       99.896
```

Fonte: O Autor.

Para o TCP na conexão entre cliente e servidor de borda, obteve-se uma latência média de 1.562 ms, um *Jitter* de 0.752 ms e o uso máximo de CPU observado foi de 85% pelo processo *Netperf* na VM 2.

No Código Fonte 4.21 é apresentado os resultados de latência entre cliente e servidor remoto utilizando o TCP.

Código Fonte 4.21 – Netperf entre o cliente e o servidor remoto TCP.

```
1 $ netperf -H 3.15.16.208 -L 60.60.0.1 -l 60 -t TCP_RR -w 10ms
     -b 1 -v 2 -- -O min_latency,mean_latency,max_latency,
    stddev_latency,transaction_rate
2 MIGRATED TCP REQUEST/RESPONSE TEST from 60.60.0.1 () port 0
    AF_INET to 3.15.16.208 () port 0 AF_INET : spin interval :
     first burst 0
3 ^CMinimum      Mean          Maximum      Stddev
     Transaction
4 Latency      Latency       Latency      Latency      Rate
5 Microseconds Microseconds  Microseconds Microseconds Tran/s
6
7 172906       180904.15     381814       29353.54     5.497
```

Fonte: O Autor.

Para este caso entre cliente e servidor remoto, obteve-se uma latência média de 180.904 ms e um *Jitter* de 29.353 ms. O uso máximo de CPU observado foi de 4% pelo processo *Netperf* na VM 2.

No Código Fonte 4.22 é apresentado o *Throughput* médio obtido entre o cliente e o servidor de borda, utilizando o protocolo TCP. O comando é similar ao utilizado para mensurar o *Throughput* médio com o protocolo UDP, alterando apenas o tipo de protocolo para TCP_STREAM.

Código Fonte 4.22 – *Throughput* entre o cliente e servidor de borda TCP.

```
1 $ netperf -H 192.168.56.122 -L 60.60.0.1 -l 5 -t TCP_STREAM -
    f m -- -R 1 -m 1400
```

```
2 MIGRATED TCP STREAM TEST from 60.60.0.1 () port 0 AF_INET to
    192.168.56.122 () port 0 AF_INET : spin interval
3 Recv   Send    Send
4 Socket Socket  Message  Elapsed
5 Size   Size    Size     Time     Throughput
6 bytes  bytes   bytes    secs.    10^6bits/sec
7
8 131072  16384   1400    5.10       100.53
```

Fonte: O Autor.

Neste caso, entre cliente e servidor de borda, obteve-se um *Throughput* médio de 100.53 Mb/s. O uso de CPU médio observado durante a execução do comando foi de 110% pelo processo nr-gnb e 46% pelo nr-ue na VM 2.

Em seguida foi utilizado o mesmo comando do caso anterior, alterando apenas o IP do servidor de destino, para mensurar o *Throughput* médio entre cliente e servidor remoto, conforme o Código Fonte 4.23.

Código Fonte 4.23 – *Throughput* entre o cliente e o servidor remoto TCP.

```
1 $ netperf -H 3.15.16.208 -L 60.60.0.1 -l 60 -t TCP_STREAM -f
    m -- -R 1 -m 1400
2 MIGRATED TCP STREAM TEST from 60.60.0.1 () port 0 AF_INET to
    3.15.16.208 () port 0 AF_INET : spin interval
3 Recv   Send    Send
4 Socket Socket  Message  Elapsed
5 Size   Size    Size     Time     Throughput
6 bytes  bytes   bytes    secs.    10^6bits/sec
7
8 131072  16384   1400    60.75       32.27
```

Fonte: O Autor.

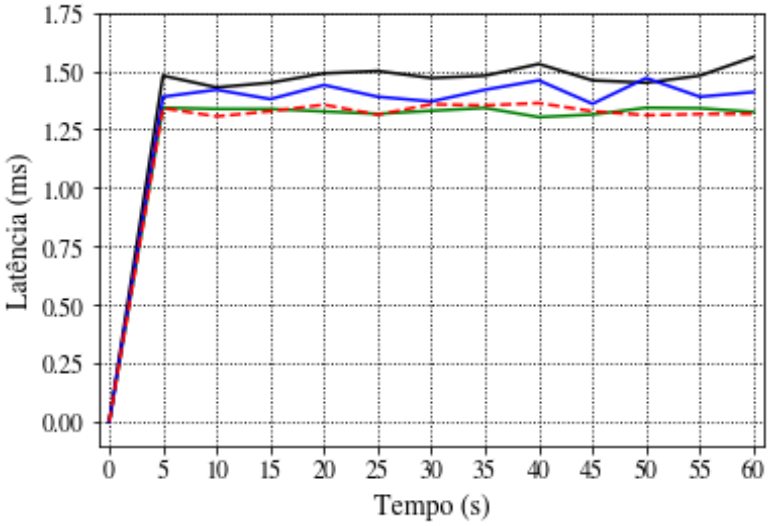
Neste ultimo caso analisado para TCP entre cliente e servidor remoto, obteve-se um *Throughput* médio de 32.27 Mb/s. O uso de CPU médio observado durante a execução do comando foi de 48% pelo processo nr-gnb e 18% pelo nr-ue na VM 2.

4.6 Gráficos dos dados obtidos

Para ilustrar e apresentar uma comparação dos resultados, segue na Figura 26, as latências obtidas através dos protocolos ICMP, UDP e TCP entre cliente e servidor de borda, utilizando ainda, o protocolo ICMP com o ULCL habilitado. Os dados foram obtidos durante uma leitura de 60 segundos, capturando os valores médios a cada 5

segundos, por exemplo, o valor médio entre o intervalo de 0 a 5 segundos em seguida de 0 a 10 segundos, etc.

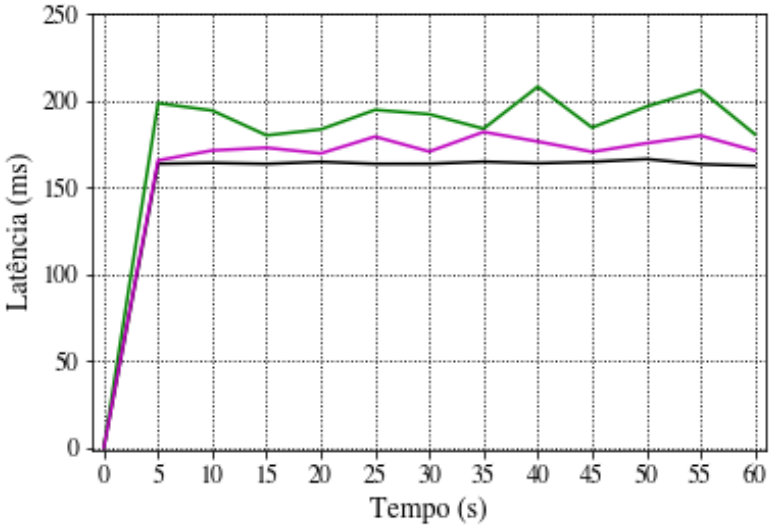
Figura 26 – Gráfico das latências obtidas entre cliente e servidor de borda, onde a cor **verde=ICMP sem ULCL**; Cor **azul=UDP**; Cor **preta=TCP** e Cor **vermelha=ICMP com ULCL**.



Fonte: O Autor.

Outro gráfico gerado foi o das latências obtidas utilizando os protocolos ICMP, UDP e TCP entre cliente e servidor remoto, conforme a Figura 27.

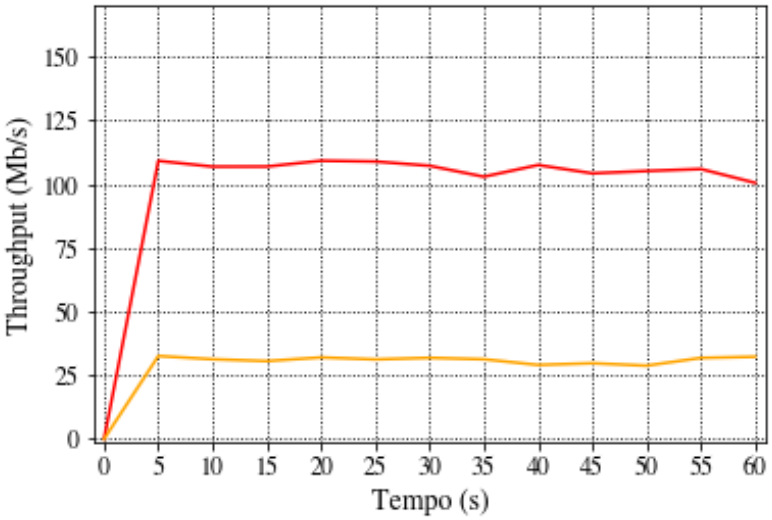
Figura 27 – Gráfico das latências obtidas entre cliente e servidor remoto, onde a cor **preta=ICMP**; Cor **verde=TCP** e Cor **magenta=UDP**.



Fonte: O Autor.

Para o último gráfico gerado é apresentado os *throughput* obtidos utilizando o protocolo TCP entre cliente, servidores de borda e remoto, conforme o Figura 28.

Figura 28 – Gráfico dos *Throughputs* obtidos utilizando o TCP entre cliente e servidores, onde a Cor **vermelho=Servidor de borda**; Cor **amarelo=Servidor remoto**.



Fonte: O Autor.

Na Tabela 4 é apresentado de forma resumida o resultado médio das latências e *Throughputs* obtidos para os protocolos ICMP, UDP e TCP.

Tabela 4 – Comparação dos resultados obtidos.

Origem e Destino dos pacotes de dados	**Latência ICMP (ms)**	**Latência UDP (ms)**	**Latência TCP (ms)**	***Throughput* UDP (Mb/s)**	***Throughput* TCP (Mb/s)**
Cliente e Servidor Borda	1.32	1.41	1.56	112.44	100.53
Cliente e Servidor Remoto	162.62	171.43	180.90	43.45	32.27
Cliente e Servidor Remoto (UL CL)	1.31	-	-	-	-

Fonte: O Autor.

Assim, com os resultados obtidos na implementação do UL CL, conforme apresentado na Tabela 4, pode-se observar que não foram obtidos dados para os protocolos UDP e TCP já que é esperado que os dados sejam os mesmos dos obtidos entre o cliente e servidor de borda.

5 Conclusões

Este trabalho resultou em um importante estudo sobre o funcionamento da arquitetura 5G, especificamente para a computação de borda. Através do Core, pode-se observar como as grandes operadoras de telecomunicações utilizam as NFs para administrar suas redes, bem como, o fluxo de dados que pode ser controlado por regras de QoS sobre seus clientes.

A utilização do *Free5GC Compose* facilitou a administração do 5GC e por ter comandos simples, pôde-se inicializar, parar e obter logs das NFs, demonstrando a versatilidade e a agilidade que traz a utilização das NFs implementadas em *containers* de fácil escalabilidade. O *UERANSIM* cumpriu o papel de estabelecer a Sessão PDU com o *Free5GC Compose* e de simular a conexão do cliente, tendo configurações simples e intuitivas.

O UL CL foi implementado de forma funcional, conseguindo classificar os pacotes de dados que contém em seu cabeçalho o IP do servidor remoto, conforme configurado no arquivo de roteamento do *Free5GC Compose*. Ele fez com que a latência, *Throughput*, perca de pacotes e *Jitter* obtidos entre cliente e servidor remoto fossem os mesmos se o cliente tentasse enviar pacotes de dados para o Servidor de borda. Dessa forma, pode-se concluir que a latência e os outros dados obtidos na utilização deste serviço podem ser considerados os mesmos dos obtidos entre o cliente e o servidor de Borda.

O *Netperf* apresentou-se uma ferramenta muito versátil que foi capaz de mensurar a latência, *Jitter*, *Throughput* e perda de pacotes. No entanto, devido as limitações de *hardware* do computador hospedeiro das VMs foi observado uma alta utilização de processamento que pode ter impactado nos resultados principalmente da VM 2 que hospedou o *UERANSIM*, bem como, na obtenção dos dados de *Throughput* para o protocolo UDP. Assim, optou-se em não construir um gráfico a partir os dados obtidos utilizando esse protocolo. Outro fator gerador de incertezas é a rede privada onde foram realizados os testes, principalmente na utilização dela para a comunicação com o servidor remoto por ter restrições impostas pela operadora na ocupação do tráfego da rede, isso pode ter interferido nos resultados.

Os dados obtidos em relação ao desempenho do sistema 5G apresentam diferenças em relação ao esperado, pois o que realmente gera o melhor desempenho da rede 5G é o hardware combinado ao software. Por mais que os dados não tenham sido obtidos a partir de um sistema real de computação de borda 5G, pode-se provar a grande vantagem natural que esta arquitetura tem em relação aos sistemas de centralização de dados, notando-se uma diferença de até quatro vezes maior no desempenho da comunicação entre o cliente e o servidor de borda em relação ao servidor remoto. Isso ocorre porque, a computação de borda puxa os recursos da nuvem que estão no núcleo do sistema de rede remoto para as bordas da rede, ou seja, para perto dos dispositivos finais, permitindo uma baixa latência,

baixo *Jitter* e alto *Throughput*, bem como, garantindo um serviço com alta eficiência energética.

Neste trabalho, a borda do sistema foi considerada em um servidor local muito próximo do servidor em que foi hospedado o Core, tendo uma distância considerada igual a zero. Além disso, o servidor remoto foi hospedado na AWS, o que gerou uma grande diferença na distância, ficando clara a vantagem do Servidor de Borda. Pode-se concluir também que a computação de borda 5G é capaz de atender a grande maioria das aplicações da indústria 4.0 e de suportar aplicações com alto consumo de banda ao mesmo tempo.

Os resultados obtidos pela WEG que foram apresentados na Seção 2.1 demonstraram a diferença impactante que o hardware do 5G proporciona, pois em uma distância de 5 metros entre uma Small Cell e CPEs, os resultados obtidos de *Throughput* foram maiores que os resultados obtidos neste trabalho.

5.1 Pesquisas futuras

Sugere-se implementar um ambiente 5G real, semelhante ao feito pela WEG, tendo foco na simulação de computação de borda com equipamentos de *hardware* físicos para o UE, RAN, Core e Servidores, próprios para a arquitetura 5G. Além disso, recomenda-se utilizar as mesmas ferramentas apresentadas neste trabalho para a obtenção dos dados dessa implementação. Assim será possível realizar comparações de desempenho com os dados que foram obtidos neste trabalho.

Pode-se também, expandir a arquitetura de computação de borda 5G com mais UPFs e UEs físicos para estudar o comportamento e influência gerada em um local com grande quantidade de dispositivos conectados ao 5GC a fim de demostrar a escalabilidade do sistema.

Também recomenda-se realizar uma pesquisa aprofundada sobre a ferramenta *iptables* que foi utilizada para configurar as regras do protocolo de internet IPv4 presentes na tabela de filtragem de pacotes. Além disso, é possível capacitar essa tabela ao adicionar novas regras para filtrar os pacotes dos protocolos UDP e TCP, além do ICMP que já foi adicionado a fim de poder disponibilizar o serviço de UL CL para todos os tipos de pacotes de dados.

Referências Bibliográficas

1 Gurjão, E. ***Livro de Minicursos SBRT 2020***. il. [S.l.]: Instituto Federal de Ensino, Ciência e Tecnologia da Paraíba – IFPB, 2020. 142 p. ISBN 978-65-87572-23-9. 22, 37, 39, 40, 41

2 LI, Z.; WANG, X.; ZHANG, T. ***5G+: How 5G Change the Society***. 1st ed.. ed. [S.l.]: Springer Singapore and Springer, 2021. ISBN 9789811568183. 22

3 YANG, Y. et al. ***Intelligent IoT for the Digital World: Incorporating 5G Communications and Fog/Edge Computing Technologies***. 1. ed. [S.l.]: Wiley, 2021. ISBN 9781119593546. 23

4 Pohlmann, T. ***Who is leading the 5G patent race for edge computing?*** Acesso em: 26/08/2021. Disponível em:<https://www.managingip.com/article/b1rznbcc4dsk23/who-is-leading-the-5g-patent-race-for-edge-computing>. 26

5 ZPE Systems. ***Products***. Acesso em: 04/09/2021. Disponível em:<https://www.zpesystems.com/products/>. 27

6 ZPE Systems. ***3 Ways Your Critical Remote Infrastructure Is Costing You***. Acesso em: 26/08/2021. Disponível em:<https://www.zpesystems.com/3-ways-your-critical-remote-infrastructure-is-costing-you/>. 27

7 WEG. ***WEG completa testes práticos de conectividade à rede 5G***. Acesso em: 26/08/2021. Disponível em:<https://www.weg.net/institutional/BR/pt/news/produtos-e-solucoes/weg-completa-testes-praticos-de-conectividade-a-rede-5g>. 28

8 WEG-V2COM OPEN LAB. ***RESULTADOS PRELIMINARES WEG-V2COM OPEN LAB 5G***. Acesso em: 04/09/2021. Disponível em:<https://sei.anatel.gov.br/sei/modulos/pesquisa/md_pesq_documento_consulta_externa.php?eEP-wqk1skrd8hSlk5Z3rN4EVg9uLJqrLYJw_9INcO6fX6o9bVPoiTHX_HKDp8z4jNp1Hsw31wuTQX8J-fqjddyWo1pe5dZRrEvwZXjQvETUCBSxxyrrpuXwu\EBod27a>. 28, 29, 30

9 iPerf. ***iPerf - The ultimate speed test tool for TCP, UDP and SCTP***. Acesso em: 07/09/2021. Disponível em:<https://iperf.fr/>. 28

10 3GPP. ***About 3GPP***. Acesso em: 26/08/2021. Disponível em:<https://www.3gpp.org/about-3gpp>. 31, 35

11 TELECO. ***HSPA e WiMax Móvel I: Tecnologias***. Acesso em: 26/08/2021. Disponível em:<https://www.teleco.com.br/tutoriais/tutorialhspawimax1/pagina_2.asp>. 31

12 PENTTINEN, J. T. J. ***5G Explained: Security and Deployment of Advanced Mobile Communications***. [S.l.]: John Wiley & Sons Ltd, 2019. ISBN 9781119275688. 32

13 Internet with a Brain. ***Motorola-nmt-450-phone***. Acesso em: 26/08/2021. Disponível em:<https://www.web3.lu/motorola-nmt-450-phone/>. 32

14 Both, C. et al. ***Soft5G+: explorando a softwarização nas redes 5G***. [S.l.]: XXXVIII Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos, 2020. 49 p. 32, 33, 34, 35, 36

15 Sudeep, T. ***Blockchain for 5G-Enabled IoT: The new wave for Industrial Automation***. 1st ed. 2021. ed. [S.l.]: Springer, 2021. ISBN 9783030674892. 32, 33

16 ROMMER, S. et al. ***5G Core Networks: Powering Digitalization***. London: Academic Press, 2020. 502 p. ISBN 978-0081030097. 35, 36, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53

17 ITU-R. ***Minimum requirements related to technical performance for IMT-2020 radio interface(s)***. Acesso em: 26/08/2021. Disponível em:<https://www.itu.int/pub/R-REP-M.2410-2017>. 36

18 3GPP. ***Study on scenarios and requirements for next generation access technologies (Release 14)***. Acesso em: 26/08/2021. Disponível em:<https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=2996>. 36

19 Mailer, C. ***Plataforma de CORE 5G em nuvem para disponibilização de funções de rede como serviço***. Acesso em: 16/09/2021. 54 p. Disponível em:<https://repositorio.ufsc.br/bitstream/handle/123456789/209624/TCC_20201_ChristianMailer.pdf?sequence=1&isAllowed=y>. 41

20 3GPP. ***Universal Mobile Telecommunications System (UMTS); LTE; General UMTS Architecture (3GPP TS 23.101 version 8.0.0 Release 8)***. Acesso em: 26/08/2021. Disponível em:<https://www.etsi.org/deliver/etsi_ts/123100_123199/123101/08.00.00_60/ts_123101v080000p.pdf>. 47, 48, 50

21 HUO, Y.; DONG, X.; XU, W. **5G Cellular User Equipment: From Theory to Practical Hardware Design**. 2017. 48

22 Pujolle, G. ***Software Networks: Virtualization, SDN, 5G, and Security***. 2nd ed. ed. [S.l.]: John Wiley & Sons, Incorporated, 2020. ISBN 9781119694724,1119694728,9781786304582. 53, 54, 55, 56

23 National Chiao Tung University. ***Free5GC Compose***. Acesso em: 26/08/2021. Disponível em:<https://github.com/free5gc/free5gc-compose>. 56, 57, 65

24 National Chiao Tung University and National Chung Cheng University. ***free5GC Link The World!*** Acesso em: 26/08/2021. Disponível em:<https://www.free5gc.org/>. 56

25 3GPP. ***General Packet Radio System (GPRS) Tunnelling Protocol User Plane (GTPv1-U)***. Acesso em: 26/08/2021. Disponível em:<https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=1699>. 57

26 3GPP. ***Interface between the Control Plane and the User Plane nodes***. Acesso em: 26/08/2021. Disponível em:<https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3111>. 57

27 GÜNGÖR, A. ***UERANSIM 5G SOLOTIONS***. Acesso em: 26/08/2021. Disponível em:<https://github.com/aligungr/UERANSIM>. 57, 58, 67

28 Guia Linux. ***PING***. Acesso em: 05/09/2021. Disponível em:<https://guialinux.uniriotec.br/ping/>. 58

29 Eriberto, J. ***ANÁLISE DE TRÁFEGO EM REDES TCP/IP COM TCPDUMP E WINDUMP***. Acesso em: 05/09/2021. Disponível em:<http://eriberto.pro.br/files/guia_tcpdump.pdf>. 58

30 iPerf. ***A tool that measures network performance of TCP/UDP including latency***. Acesso em: 13/09/2021. Disponível em:<https://sourceforge.net/projects/iperf2/files/>. 58, 68

31 Mills, M. ***iperf3: Tutorial para medir a velocidade entre dois dispositivos LAN e WiFi***. Acesso em: 05/09/2021. Disponível em:<https://itigic.com/pt/iperf3-tutorial-to-measure-speed-between-two-lan-and-wifi-devices/>. 58

32 iPerf3. ***Public iPerf3 servers***. Acesso em: 05/09/2021. Disponível em:<https://iperf.fr/iperf-servers.php>. 59

33 Netperf. ***Netperf Manual***. Acesso em: 13/09/2021. Disponível em:<https://hewlettpackard.github.io/netperf/doc/netperf.html#TCP_005fSTREAM>. 59

34 Netperf. ***Netperf Releases***. Acesso em: 13/09/2021. Disponível em:<https://github.com/HewlettPackard/netperf/releases>. 59, 69

35 ORACLE. ***Download VirtualBox***. Acesso em: 26/08/2021. Disponível em:<https://www.virtualbox.org/wiki/Downloads>. 61

36 CANONICAL. ***Get Ubuntu Server***. Acesso em: 26/08/2021. Disponível em:<https://ubuntu.com/download/server>. 61

37 Amazon. ***Amazon EC2***. Acesso em: 07/09/2021. Disponível em:<https://aws.amazon.com/ec2/>. 64

38 PHANEKHAM, D.; JONES, R. ***What a trip! Measuring network latency in the cloud***. Acesso em: 13/09/2021. Disponível em:<https://cloud.google.com/blog/products/networking/using-netperf-and-ping-to-measure-network-latency>. 68, 74

APÊNDICE A – Código em python para a geração dos gráficos

No Código Fonte A.1 segue o código em python utilizado para a geração dos gráficos.

Código Fonte A.1 – Código em python para geração dos gráficos.

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Sun Sep 12 21:57:06 2021

@author: Dener Kraus
"""
#-------------------------------------------------------------
#Importação de módulos/bibliotecas/pacotes
import matplotlib as mplt
import matplotlib.pyplot as plt
import numpy as np
import pickle

def ajusta_posicao_axes(axes,offset):
    box = axes.get_position()
    axes.set_position([box.x0+offset[0], box.y0+offset[1],
                       box.width+offset[2], box.height+offset
   [3]])
# -------------------------------------------------------------
# Configurações gerais das figuras
mplt.rc('lines', linewidth=1.5)
mplt.rc('grid', linestyle=":", color='black', linewidth=0.9)

mplt.rc('ytick', labelsize=12)
mplt.rc('xtick', labelsize=12)

mplt.rcParams['mathtext.fontset'] = 'stix'
mplt.rcParams['font.family'] = 'STIXGeneral'
# -------------------------------------------------------------
# Lê o arquivo ".txt" que contêm os dados obtidos
```

```
31 file = open('dados_tcc.txt','r')
32 T = []
33 LPL = []
34 LPR = []
35 LPU = []
36 LTL = []
37 LTR = []
38 TTL = []
39 TTR = []
40 LUL = []
41 LUR = []
42 for line in file:
43     splitLine = line.split("\t")
44     T.append(float(splitLine[0]))
45     LPL.append(float(splitLine[1]))
46     LPR.append(float(splitLine[2]))
47     LPU.append(float(splitLine[3]))
48     LTL.append(float(splitLine[4]))
49     LTR.append(float(splitLine[5]))
50     TTL.append(float(splitLine[6]))
51     TTR.append(float(splitLine[7]))
52     LUL.append(float(splitLine[8]))
53     LUR.append(float(splitLine[9]))
54
55 file.close()
56 # ------------------------------------------------------------
57 # Gerar as figuras
58 # Figura 1
59 fig1, ax1 = plt.subplots(figsize=(15/2.54, 10/2.54));
60 curve1 = ax1.plot(T, LPL,label='LPL', color='g')
61 curve2 = ax1.plot(T, LTL,label='LTL', color='k')
62 curve3 = ax1.plot(T, LUL,label='LUL', color='b')
63 curve3 = ax1.plot(T, LPU,label='LUL', color='r', linestyle='
     dashed')
64 ax1.set_ylabel("Latência (ms)", color='k',size=14)
65 ax1.set_xlabel("Tempo (s)", color='k',size=14)
66 ax1.grid(color='k')
67 ax1.xaxis.grid(color='k')
68 plt.xlim(-1, 61)
```

```
69 ax1.set_ylim(-0.1, 1.75)
70 ax1.set_xticks([0, 5, 10, 15, 20, 25, 30, 35, 40, 45, 50, 55,
       60])
71 plt.show()
72
73 # Figura 2
74 fig2, ax1 = plt.subplots(figsize=(15/2.54, 10/2.54));
75 curve4 = ax1.plot(T, LPR,label='LPR', color='k')
76 curve5 = ax1.plot(T, LTR,label='LTR', color='g')
77 curve6 = ax1.plot(T, LUR,label='LUR', color='m')
78 ax1.set_ylabel("Latência (ms)", color='k',size=14)
79 ax1.set_xlabel("Tempo (s)", color='k',size=14)
80 ax1.grid(color='k')
81 ax1.xaxis.grid(color='k')
82 plt.xlim(-1, 61)
83 ax1.set_ylim(-1, 251)
84 ax1.set_xticks([0, 5, 10, 15, 20, 25, 30, 35, 40, 45, 50, 55,
       60])
85 plt.show()
86
87 # Figure 3
88 fig4, ax1 = plt.subplots(figsize=(15/2.54, 10/2.54));
89 curve3 = ax1.plot(T, TTL,label='TTL', color='r')
90 curve4 = ax1.plot(T, TTR,label='TTR', color='orange')
91 ax1.set_ylabel("Throughput (Mb/s)", color='k',size=14)
92 ax1.set_xlabel("Tempo (s)", color='k',size=14)
93 ax1.grid(color='k')
94 ax1.xaxis.grid(color='k')
95 plt.xlim(-1, 61)
96 ax1.set_ylim(-1, 170.5)
97 ax1.set_xticks([0, 5, 10, 15, 20, 25, 30, 35, 40, 45, 50, 55,
       60])
98 plt.show()
```

Fonte: O Autor.