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

Gabriel Henrique Davanço Silva

Classificação de tráfego por classes de serviço no núcleo 5G

Blumenau
2022

Gabriel Henrique Davanço Silva

Classificação de tráfego por classes de serviço no núcleo 5G

Trabalho de Conclusão de Curso apresentado à Universidade Federal de Santa Catarina como parte dos requisitos necessários 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 e Automação e Computação

Blumenau
2022

Gabriel Henrique Davanço Silva

Classificação de tráfego por classes de serviço no núcleo 5G

Trabalho de Conclusão de Curso apresentado à Universidade Federal de Santa Catarina como requisito parcial para a obtenção do título de Engenheiro de Controle e Automação.

Comissão Examinadora

Documento assinado digitalmente
Adao Boava
Data: 01/08/2022 13:48:30-0300
CPF: 645.640.969-15
Verifique as assinaturas em https://v.ufsc.br

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

Documento assinado digitalmente
Ciro Andre Pitz
Data: 01/08/2022 13:51:06-0300
CPF: 055.728.089-38
Verifique as assinaturas em https://v.ufsc.br

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

Documento assinado digitalmente
Mauri Ferrandin
Data: 01/08/2022 16:53:00-0300
CPF: 018.580.869-73
Verifique as assinaturas em https://v.ufsc.br

Prof. Dr. Mauri Ferrandin
Universidade Federal de Santa Catarina

Blumenau, 1 de agosto de 2022

Agradecimentos

Agradeço aos meus pais que sempre me apoiaram e me deram suporte para que eu seguisse os caminhos que sonhava. Agradeço à minha esposa que sempre esteve ao meu lado nos melhores e piores momentos dos últimos anos. Agradeço aos meus professores que me transformaram através do conhecimento e aprendizado. Por fim, agradeço a todos que de alguma forma ajudaram para a realização deste trabalho, seja com uma dica ou conselho, seja com uma crítica ou com palavras de incentivo.

"Eu posso não ter ido para onde eu pretendia ir, mas eu acho que acabei terminando onde eu pretendia estar."

(Douglas Adams)

Resumo

Um dos objetivos do 5G é atender à uma extensa variedade de cenários como: latências fim-a-fim menores que 1 ms para carros autônomos e drones, velocidades de *downlink* na casa de Gbit/s para celulares e computadores pessoais e conectar milhões de sensores e equipamentos inteligentes simultaneamente com grande eficiência energética, tudo isto utilizando apenas uma infraestutura de rede. Com estas demandas diversas, surge o desafio de proporcionar uma boa experiência de acesso a todos estes equipamentos e usuários. A fim de atender essas diferentes demandas, o conceito de classes de serviço é criado. A divisão do tráfego nestas classes de serviço em conjunto com o uso de técnicas de fatiamento de rede e aplicação de QoS, aloca recursos de maneira mais eficiente no núcleo, aprimorando a qualidade da experiência dos usuários, mitigando os gargalos de rede. Neste contexto, propõe-se funções para classificar os pacotes que trafegam no core 5G nestes diferentes cenários em suas grades de serviço correspondentes. Utilizando um ambiente 5G e equipamentos de usuário simulados, três abordagens de classificação implementadas como funções em *python*, usam características dos cabeçalhos e área de dados dos pacotes para identificá-los como uma das três classes: eMBB, mMTC e URLLC. Com estas funções foi possível realizar uma classificação satisfatória obtendo 99% de precisão em alguns casos.

Palavras-Chave: Redes 5G, Classificação de tráfego, Classes de serviço, Free5gc Compose, UERANSIM

Abstract

A 5G objective is to support an extense variety of scenarios such as: end-to-end latency less than 1 ms for autonomous cars and drones, downlink speeds around Gbit/s for cell phones and personal computers and connect millions of sensors and smart devices simultaneously with great energy efficiency, all of this using only one network infrastructure. With these diverse demands, the challenge of providing a good access experience to all these equipment and users. In order to meet these different demands, the concept of service grades is created. The division of traffic into service grades alongside with the use of techniques such as network slicing and QoS enforcement, allocates resources more efficiently in the core, improving the quality of user experience, mitigating the network bottlenecks. In this context, functions are proposed to classify the packets travelling thru 5G core in this different scenarios in its corresponding service grades. Using simulated 5G environment and user equipment, three classification approaches, implemented as python functions, use packets headers and payload to identify them as one of the grades: eMBB, mmTC and URLLC. With these functions it was possible to perform a satisfactory classification, reaching more than 99% of accuracy in some cases.

Keywords: 5G, Traffic classification, Service grades, Free5gc Compose, UERANSIM

Lista de figuras

Lista de tabelas

Lista de Siglas e Abreviaturas

1G	*1st Generation*
2G	*2nd Generation*
2.5G	*2.5rd Generation*
2.75G	*2.75rd Generation*
3G	*3rd Generation*
3GPP	*3rd Generation Partnership Program*
4G	*4th Generation*
5G	*5th Generation*
5GC	*5G Core*
AMF	*Access Mobility Function*
AMP	*Advanced Mobile Phone*
API	*Application Programming Interface*
AR	*Augmented Reality*
ARP	*Address Resolution Protocol*
AUSF	*Authentication Server Function*
CDMA	*Code Division Multiple Access*
CLI	*Command Line Interface*
CP-OFDM	*Cycle Prefix-Orthogonal Frequency Division Multiplexing*
CPU	*Central Processing Unit*
DHCP	*Dynamic Host Configuration Protocol*
DJIUAV	*DJI Unmanned Aerial Vehicle*
DL	*Downlink*
DN	*Data Network*
DNN	*Data Network Name*
DNS	*Domain Name System*
EGPRS	*Enhanced Data Rates For GSM Evolution*
eMBB	*Enhanced Mobile Broad-Band*
eNB	*Enhanced Node B*
EPC	*Evolved Packet Core*
EPS	*Evolved Packet System*
E-UTRAN	*Evolved UMTS Terrestrial Radio Access Network*
FDMA	*Frequency Division Multiple Access*
FM	*Frequency Modulation*
FQDN	*Fully Qualified Domain Name*
FTP	*File Transfer Protocol*

GB	*Gigabyte*
gNB	*Next Generation Node Base*
GPL-2.0	*GNU General Public License v2.0*
GPL-3.0	*GNU General Public License v3.0*
GPRS	*General Packet Radio Service*
GSM	*Global System for Mobile Communications*
GTP-U	*General Packet Radio Service Tunneling Protocol User*
GUI	*Graphic User Interface*
HSPA+	*High Speed Packet Access Plus*
HTTP	*Hyper Text Transfer Protocol*
HTTPS	*Hyper Text Transfer Protocol Secure*
Hz	*Hertz*
IANA	*Internet Assigned Network Authority*
ICMP	*Internet Control Message Protocol*
IEEE	*Institute of Eletrical and Eletronic Engineers*
IMSI	*International Mobile Subscriber Identity*
IoT	*Internet of Things*
IP	*Internet Protocol*
km^2	*Kilômetro Quadrado*
ISO	*International Standards Organization*
LAN	*Local Area Network*
LTE	*Long Term Evolution*
M2M	*Machine to Machine*
MBB	*Mobile Broad Band*
mMTC	*Massive Machine Type Communication*
MQTT	*Message Queuing Telemetry Transport*
MTU	*Maximum Transmission Unit*
N3IWF	*Non-3GPP Inter-Working Function*
NaaS	*Network as a Service*
NAS	*Non-Access Stratum*
NAT	*Network Address Translation*
NCTU	*National Chiao Tung University*
NEF	*Network Exposure Function*
NF	*Network Function*
NFV	*Network Function Virtualization*
NGAP	*NG Application Protocol*
NG-RAN	*Next Generation RAN*
NR	*New Radio*
NRF	*Network Repository Function*

NSSF	*Network Slice Selection Function*
NWDAF	*Network Data Analysis Function*
MAC	*Media Access Control*
MIMO	*Multiple Input Multiple Output*
MTSO	*Mobile Telecommunications Switch Office*
MU-MIMO	*Multi-User MIMO*
OAM	*Operation, Administration and Management*
OFDM	*Orthogonal Frequency Division Multiplexing/Multiple Access*
OM	*Operations and Management*
OSI	*Open Systems Interconnection*
PCF	*Policy Control Function*
PDC	*Personal Digital Celular*
PDU	*Protocol Data Unit*
PFCP	*Packet Forward Control Protocol*
PLMN	*Public Land Mobile Network*
PSTN	*Public Switched Telephone Network*
QoE	*Quality of Experience*
QoS	*Quality of Service*
RAM	*Random Access Memory*
RAN	*Radio Acess Network*
RARP	*Reverse Address Resolution Protocol*
REST	*Representational State Transfer*
SC-FDMA	*Single-Carrier Frequency Division Multiple Access*
SCTP	*Stream Control Transmission Protocol*
SD	*Slice Differentiator*
SDN	*Software Defined Network*
SL	*Supervised Learning*
SMF	*Session Management Function*
SMS	*Short Message Service*
SMTP	*Simple Mailt Transfer Protocol*
S-NSSAI	*Single Network Slice Selection Assistance Information*
SSH	*Secure Shell*
SST	*Slice Type*
SU-MIMO	*Single User MIMO*
SUPI	*Subscription Permanent Identifier*
TDMA	*Time Division Multiple Access*
TD-SCDMA	*Time Division-Synchronous Code Division Multiple Access*
TEID	*Tunnel Endpoint Identifier*
UDM	*Unified Data Management*

UDP	*User Datagram Protocol*
UDR	*Unified Data Repository*
UE	*User Equipament*
UFSC	*Universidade Federal de Santa Catarina*
UL	*Uplink*
ULCL	*Uplink Classifier*
UMTS	*Universal Mobile Telecommunications System*
UPF	*User Plane Function*
URLLC	*Ultra Reliable Low Latency Communication*
VLAN	*Virtual LAN*
VM	*Virtual Machine*
VoIP	*Voice over IP*
VR	*Virtual Reality*
WCDMA	*Wideband CDMA*
WebUI	*Web User Interface*
WiMAX	*Worldwide Interoperability for Microwave Access*

Lista de Códigos Fontes

Sumário

1 Introdução

Conforme as redes móveis de comunicação evoluem, maiores se tornam os desafios para atender de maneira satisfatória as demandas dos usuários destas redes. Atingir um nível de qualidade de serviço exigido e proporcionar uma alta Qualidade de Experiência (QoE - *Quality of Experience*) para estes consumidores se torna não somente uma preocupação, mas uma fator fundamental para o projeto destas soluções.

Em um contexto onde estão presentes conceitos como Internet das coisas (IoT - *Internet of Things*), indústria 4.0, Realidade Virtual/Realidade Aumentada (VR/AR *Virtual Reality/Augmented Reality*) e *Big Data*, estes usuários passam a ter um sentido mais amplo, indo além dos equipamentos de comunicação convencionais como celulares, *tablets* e computadores, se estendendo para os objetos *smart* (eletrodomésticos, equipamentos de domótica, *wearables*, entre outros) passando por carros autônomos, sensores, atuadores e até mesmo redes elétricas inteligentes, ou *smart grids*, e satélites. Dentro desta perspectiva, uma das soluções que mais se mostra promissora é a das redes 5G [1, 2].

A quinta geração de redes móveis tem como características principais ser ágil, escalável e flexível, tornando possível projetar, provisionar e gerir arquiteturas de rede para diversas aplicações e finalidades [2]. Isto é obtido através da utilização de conceitos de virtualização como: Virtualização de funções de rede (NFV - *Network Function Virtualization*) e Redes definidas por software (SDN - *Software Defined Networks*) juntamente com o conceito de Rede como serviço (NaaS - *Network as a Service*).

1.1 Problematização

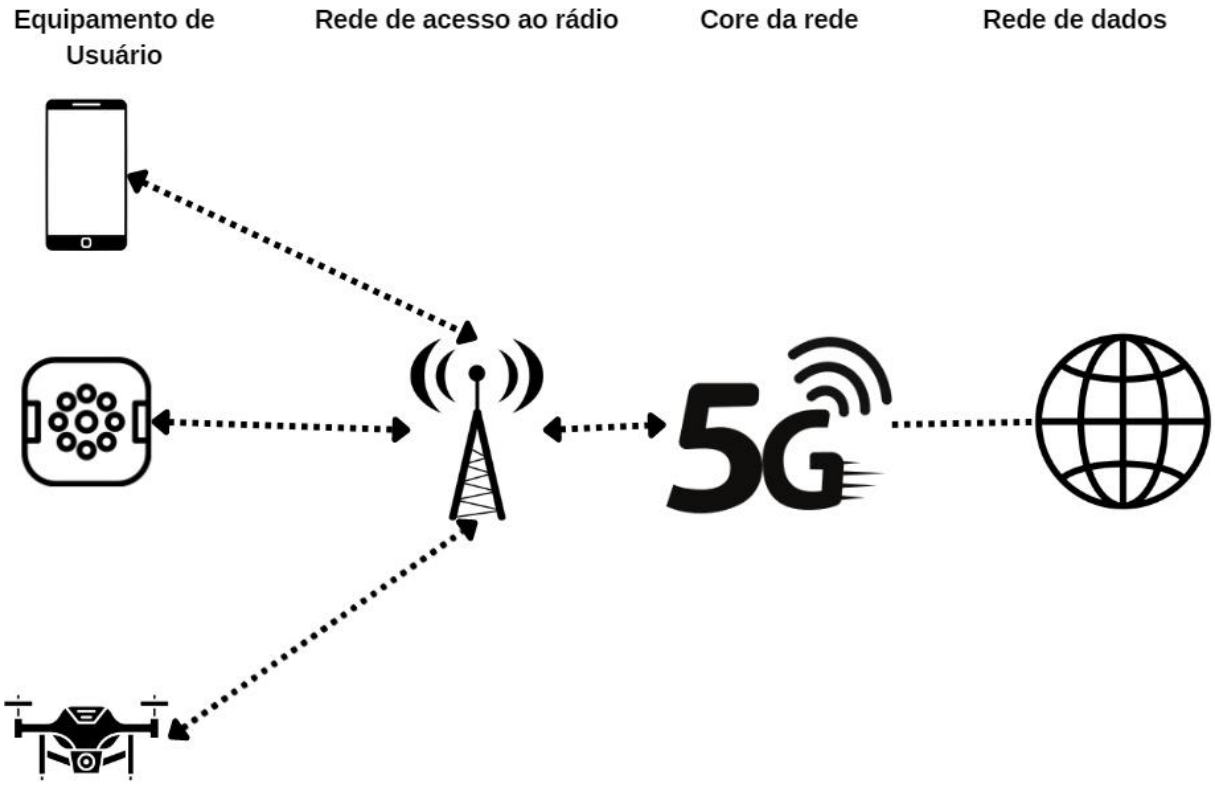
As redes 5G buscam diferenciar-se das gerações anteriores através da disponibilização de diferentes serviços. O Projeto de Parceria de 3ª Geração (3GPP - *3rd Generation Partnership Project*), por meio da Release 15 de 2019 [3], define três principais grades de serviço: Banda larga móvel aprimorada (eMBB - *enhanced Mobile Broad Band*), Comunicações massivas do tipo máquina (mMTC - *massive Machine Type Communications*) e Comunicações de baixa latência ultra-confiáveis (URLLC - *Ultra Reliable Low Latency Communications*).

Para o ganho de desempenho da rede se deseja que os recursos dos enlaces sejam alocados eficientemente de acordo com a quantidade, periodicidade e tipo do tráfego que circula pelos mesmos. Assim sendo, técnicas de classificação de tráfego, em associação com técnicas de aplicação de qualidade de serviço (QoS - *Quality of Service*), podem trazer ganhos de desempenho para arquiteturas de rede, principalmente através da mitigação, ou até mesmo ausência, de "gargalos"de rede.

Não há, no entanto, uma padronização para como classificar o tráfego e nem quais características deste tráfego devem pertencer a cada uma destas classes, o que torna o encaminhamento (por exemplo para diferentes enlaces ou até mesmo fatias de rede) e a posterior aplicação de qualidade de serviço uma tarefa complexa e ainda pouco aplicada em implementações práticas.

Um exemplo de aplicação que se beneficia da classificação do tráfego por classes de serviço é uma rede 5G como demonstrada na Figura 1, onde três usuários com demandas muito diferentes são servidos pelo mesmo conjunto de funções de rede dentro do mesmo núcleo, enquanto o smartphone necessita de alta taxa de dados de pico, o mesmo não possui uma necessidade de baixas latências, menores que 1 ms tal qual o drone, já o sensor precisa de eficiência energética ao acessar os serviços do *core*. Espera-se que o tipo de tráfego gerado por cada usuário tenha características e necessidades específicas, podendo assim serem não só classificados, como também encaminhados de maneira diferente pelo *core* até a rede de dados.

Figura 1 – Topologia de rede 5G com três Equipamentos de usuário



Fonte: O Autor

Ao longo do trabalho, esta topologia será utilizada como base para as implementações e aplicações desenvolvidas. Tendo esta arquitetura em mente, será escolhida a base de dados e serão projetadas as funções de classificação.

1.2 Objetivos gerais

Este trabalho tem como objetivo geral estudar, analisar e comparar métodos para classificação de tráfego em classes de serviço de redes 5G, além de utilizar ferramentas de simulação de redes 5G voltadas a técnicas de geração, captura e classificação de tráfego no core da rede.

1.3 Objetivos específicos

Os objetivos específicos deste trabalho podem ser listados como:

- Compreender de maneira profunda as redes 5G, suas partes constituintes, sistemas e serviços;
- Configuração, projeto e simulação de uma topologia de rede 5G, contendo o Equipamento de usuário (UE - *User Equipament*), Rede de acesso ao rádio (RAN - *Radio Access Network*) e integração de técnicas de classificação de tráfego no *core*;
- Análise as técnicas de QoS e QoE;

1.4 Estrutura do documento

O Capítulo 2 contém uma pesquisa sobre a pilha de protocolos TCP/IP, a história das redes sem fio, uma revisão sobre redes 5G contendo seus principais objetivos, componentes e serviços, bem como uma compilação de tecnologias de virtualização e as ferramentas utilizadas para simular o envio, captura e classificação de tráfego. O capítulo 3 apresenta a metodologia utilizada para a simulação e obtenção de dados, bem como explica o funcionamento das funções de classificação. Já o capítulo 4 apresenta e analisa os resultados obtidos a partir da simulação proposta. E, por fim, o capítulo 5 conclui e expõe algumas das melhorias e futuras pesquisas a serem realizadas.

2 Revisão de Literatura

Este capítulo visa apresentar os principais conceitos utilizados durante a elaboração deste trabalho, exibindo algumas definições e padronizações que nortearam a solução final. Primeiramente é apresentada a pilha de protocos TCP/IP, uma vez que os pacotes, seus cabeçalhos e tamanho da sua área de dados serão utilizados na implementação da classificação final. Além disso o capítulo sumariza a história das redes móveis de comunicação, os objetivos e componentes da rede 5G, algumas formas de virtualização bem como as ferramentas de simulação utilizadas ao longo do trabalho.

2.1 Pilha de protocolos TCP/IP

O TCP/IP representa um conjunto de protocolos que permite que diversos equipamentos que constituem uma rede possam comunicar-se entre si. É um protocolo estruturado por camadas na qual cada camada utiliza e presta serviços às camadas adjacentes. Cada camada apenas trata das informações que correspondem à sua função [4].

Esta arquitetura foi inspirada nos avanços e desafios experimentados pela sua antecessora ARPANET, uma rede experimental patrocinada pelo Departamento de Defesa dos Estados Unidos (DoD - *U.S. Departament of Defense*) que ligava centenas de universidades e instalações governamentais. Uma das exigências do DoD era que as conexões permanecessem intactas desde que as máquinas de origem e destino estivessem funcionais, mesmo que algumas máquinas ou linhas de transmissão entre elas tivessem sido retiradas de operação [5]. Além disso uma arquitetura flexível era necessária, uma vez que eram vastos os tipos de aplicação que se desejava oferecer.

Todos estes requisitos culminaram na escolha de uma rede do tipo comutação de pacotes, baseada em uma camda que faz a conexão entre rede, mas não baseada no estado da conexão em si. A pilha de protocolos consiste em cinco camadas: camada física, camada de enlace com a rede, camada de rede, camada de transporte e camada de aplicação [6].Cada uma delas é apresentada em mais detalhes a seguir e a sua disposição pode ser visualizada na fig:layers.png.

2.1.1 Camada física

A camada física possui a função de mover individualmente os *bits* dentro de um quadro de um nó para outro [6]. Esta camada descreve as características físicas da comunicação tais como a natureza do meio usado para a comunicação (cobre, fibra-óptica ou *links* de rádio) e todos os detalhes relacionados com os sinais (modulações, comprimentos de

Figura 2 – A pilha de protocolos TCP/IP

Fonte: Adaptado de [6]

onda, níveis de sinal, sincronizações, distâncias máximas, etc). Por estes aspectos, é fortementemente dependente da arquitetura e do meio utilizados [4, 5].

2.1.2 Camada de enlace com a rede

Os serviços da camada de enlace com a rede, assim como da camada física, variam conforme a arquitetura de rede utilizada. É a camada de abstração de *hardware* e devido à enorme variedade de tecnologias de rede possíveis, é uma camada não normalizada pelo modelo TCP/IP. É possível a interligação e interoperação com redes heterogêneas [4].

Para duas das arquiteturas mais comuns, *Ethernet* e os padrões *IEEE 802.11* (também conhecidos como Wi-Fi), um endereço físico denominado endereço MAC é entregue a cada um dos nós da rede. Este endereço é utilizado no processo de encapsulamento e desencapsulamento para encontrar o nó de destino corretamente.

2.1.3 Camada de rede

A camada de rede é reponsável por rotear datagramas de um *host* a outro. A maneira como este roteamento é feito é através de um endereço lógico denominado endereço IP [6].

Esta camada é responsável pelo endereçamento, roteamento e controle de envio e recepção dos dados. A comunicação é realizada por datagramas. O protocolo IP é não orientado à conexão, não garantindo que os pacotes IP cheguem ao seu destino nem garantindo a entrega na ordem em que foram enviados. O IP é o protocolo responsável por definir o caminho que um pacote de dados deverá percorrer desde o *host* de origem até

ao *host* destino, passando por uma ou várias redes onde poderá encontrar protocolos de conexão como o IP, o ICMP (*Internet Control Message Protocol*), responsável por encaminhar mensagens de controle na rede, o ARP (Address Resolution Protocol), responsável por mapear os endereços IPs da camada 3 em endereços de *hardware* na camada 2, e o RARP (Reverse Address Resolution Protocol), que realiza a função oposta ao ARP [4].

2.1.4 Camada de transporte

A camada de transporte é responsável por transportar mensagens entre o servidor e o cliente. Nesta camada, dois são os protocolos possíveis: TCP e UDP.

O protocolo TCP é orientado a conexão, ou seja possui mecanismos de controle para garantir a entrega de pacotes e controle de fluxo (por exemplo para sincronizar a velocidade do emissor/receptor). Além disso o protocolo TCP também divide as mensagens em segmentos mais curtos e possui controle de congestionamento.

O protocolo UDP, por sua vez, é não orientado a conexão e não confiável. No entanto estas podem ser características desejadas, principalmente se o tempo de resposta entre o emissor e o receptor (também chamado latência) precisar ser inferior à um valor pré estipulado.

2.1.5 Camada de aplicação

A camada de aplicação é a responsável pelas aplicações da rede. Esta camada inclui vários protocolos como (HTTP - *Hyper Text Transfer Protocol*) para páginas *web*, SMTP (*Simple Mail Transfer Protocol*) para serviços de *email*, FTP (*File Transfer Protocol*) para transferência de arquivos, entre outros [6]. Esta camada não possui um padrão comum para todas as aplicações, ou seja, consoante o serviço em questão irá depender também o protocolo que o vai atender [4].

A maioria dos protocolos de aplicação possui um endereço fixo para comunicação com o sistema operacional, também denominado porta. Para os mais utilizados e convencionais, estes endereços são padronizados pela IANA (*Internet Assigned Numbers Authority*), já para protocolos específicos as chamadas portas dinâmicas são utilizadas, podendo ter um valor máximo de 65535.

2.2 História das redes de comunicação móvel

Sistemas de comunicação móveis evoluíram vastamente desde a introdução da primeira geração analógica nos anos 1980. Desde então, cada sistema oferece novas funções e funcionalidades que superam as de gerações anteriores [1].

2.2.1 Primeira Geração de redes móveis - 1G

Datadas da década de 1980, as primeiras redes de comunicação móvel eram analógicas e eram voltadas, em sua grande maioria, para chamadas de voz. Os primeiros equipamentos eram pouco portáteis e pesavam alguns kilogramas.

Neste contexto, o sistema de telefone móvel avançado (AMPS - *Advanced Mobile Phone System*) foi o padrão de rede de acesso 1G mais amplamente implementado [7]. O núcleo é chamado de MTSO(*Mobile Telecommunications Switch Office*) e se interliga com a rede pública de telefonia comutada (PSTN - *Public Switched Telephone Network*) às estações base que realizam as conexões usando modulação por frequência (FM - *Frequency Modulation*) além de realizar o controle de acesso ao meio por FDMA (*Frequency Division Multiple Access*).

A primeira geração foi descontinuada praticamente em todos os lugares do mundo com a chegada do novo milênio, sendo uma exceção com relação às gerações atuais que têm se mantido operacionais de maneira paralela [1].

2.2.2 Segunda Geração de redes móveis - 2G

Lançada em 1991, a arquitetura 2G foi a primeira rede de comunicação móvel a ser adotada em larga escala. Seus principais focos eram chamadas de voz e mensagens de texto [2]. Foi a primeira geração a utilizar tecnologias digitais e eram consideradas mais estáveis, cobriam áreas maiores e tinham capacidade de comportar mais usuários que as redes predecessoras, a tecnologia de acesso era chamada GSM (*Global System for Mobile Communications*).

Além do GSM outros dois padões foram importantes no Brasil, IS-136 e o IS-95. O primeiro utilizava a largura do canal de 30 kHz e a tecnologia TDMA. Já o segundo utilizava a tecnologia de acesso CDMA e operava na faixa dos 800 MHz, onde a largura de cada canal era de $1,25$ MHz e a largura de banda de 25 MHz [8].

Além disso, essa geração proporcionou a 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 Acess*) e PDC (*Personal Digital Celular*), tecnologia que foi usada exclusivamente no Japão [7].

O GSM 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 de pacotes. Essa evolução foi finalizada com a chamada geração 2.75G, através da tecnologia EDGE (*Enhanced Data Rates For GSM Evolution*), que apresentou um padrão de melhoria das taxas de dados para o GSM [7].

Essas evoluções tecnológicas desde o GSM, passando pelo GPRS e pelo EDGE, capacitaram os UEs a terem conexão com a *internet*, utilização de serviços multimídia e

navegação por meio de *browsers*. Embora com modificações para atender o aumento da demanda de dados, o sistema 2G ainda se encontra operacional até os dias de hoje.

2.2.3 Terceira Geração de redes móveis - 3G

Lançada em 1999, a terceira geração de redes móveis deu aos usuários a possibilidade de navegar na *internet* e utilizar funcionalidades adicionais nos telefones celulares [2]. Os padrões utilizadas nessa geração são GSM, EGPRS (*Enhanced General Packet Radio Service*), sistema móvel universal de telecomunicações (UMTS - *Universal Mobile Telecommunications System*) e, além dos padrões citados, CDMA 2000, TD-SCDMA (*Time Division-Synchronous Code Division Multiple Access*) e o CDMA de banda larga (WCDMA - *Wideband CDMA*) foi amplamente adotado.

Esta geração conta com taxas de dados consideravelmente maiores que as antecessoras, chegando à $6Mbit/s$ com a tecnologia HSPA+ (*High Speed Packet Access Plus*) enquanto as redes 2G EDGE (*Enhanced Data Rates for GSM Evolution*) atingiam taxas de dados máxima teóricas de $130Kbit/s$ [9].

2.2.4 Quarta Geração de redes móveis - 4G

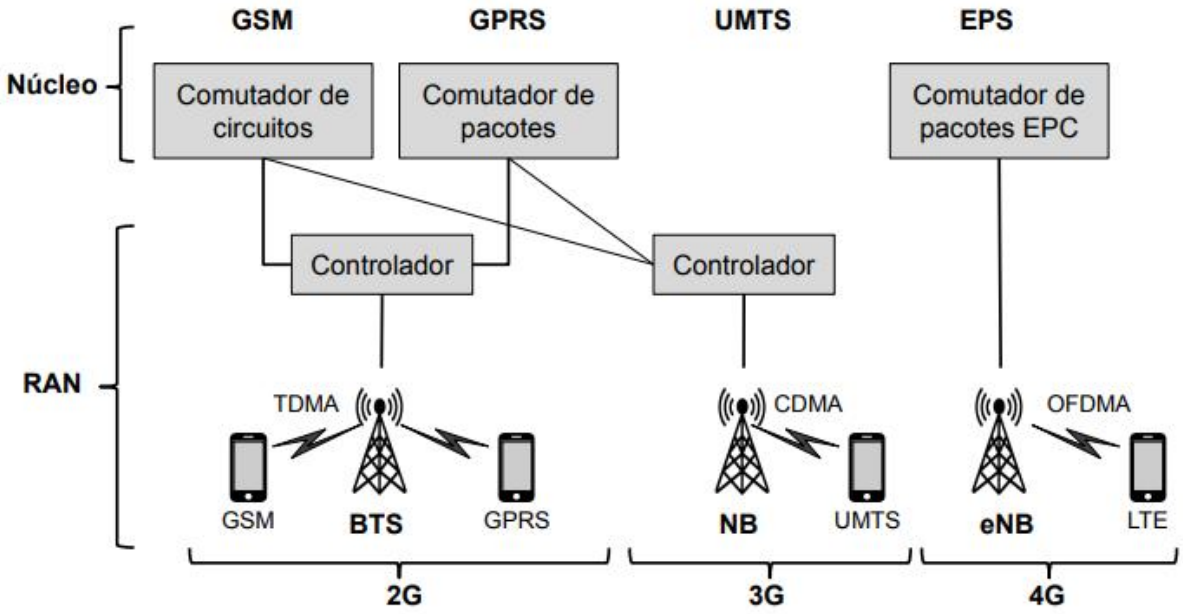
A rede 4G, lançada em 2010, revolucionou a utilização da *internet* na telefonia móvel. Dois padrões emergiram dos esforços globais para a padronização dessa rede, sendo o LTE (*Long Term Evolution*) desenvolvido pela 3GPP que define uma rede de acesso composta por estações rádiobase do tipo NodeB evoluída (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 (*Non-Access Stratum*).

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*) [7].

Outra mudança significativa nas redes 4G é que o tráfego de voz é do tipo Voz sobre IP (VoIP - Voice over IP) [10]. A Figura 3 sumariza as tecnologias utilizadas nas gerações

digitais (do 2G ao 4G) na parte da rede de acesso e no núcleo.

Figura 3 – Evolução arquitetural das redes móveis.



Fonte: Adaptado de [7]

2.3 Sistema 5G

A arquitetura do 5G consiste de duas partes: a nova tecnologia NG-RAN (*Next Generation RAN*) com suporte a *New Radio* (NR) e o *core* 5G (5GC). Ambos tendo mudado consideravelmente comparados a gerações anteriores [2]. Suas padronizações foram lideradas sobretudo pelas *releases* 15, 16 e 17 da 3GPP. Estas partes constituintes, bem como seus objetivos e algumas tecnologias relevantes são descritas as seguir.

2.3.1 Objetivos do 5G

As redes 5G surgem como a evolução das redes LTE no aspecto de proporcionarem maiores taxas de trasmissão de dados e menores latências fim-a-fim. Estas são características muito desejadas devido ao aumento significativo de equipamentos multimídia conectados e prevendo a continuidade dessa tendência nos próximos anos. A Tabela 1 mostra alguns dos principais requisitos desta nova tecnologia para lidar com estes desafios.

Mesmo sendo uma evolução natural das gerações antecessoras, o 5G se mostra distinto das mesmas em outros aspectos. Um dos mais importantes é a mudança de paradigma onde as operadoras, que até então tinham como principais clientes os usuários finais, podem passar a ter como clientes principais as indústrias. Isto representa uma mudança sem precedentes não somente tecnológica, mas no modelo de negócios [2].

Tabela 1 – Requisitos das redes 5G.

Métrica	Requisito
Pico da taxa de dados	Até 20 Gb/s DL, até 10 Gb/s UL
Taxas de transmissão médias observadas	Até 100Mb/s DL, até 50 Mb/s UL
Eficiência espectral	Até 30 bits/s Hz DL, até 15 bits/s Hz UL
Densidade de conexão	Até 1 milhão de equipamentos/km^2
Vida de bateria dos equipamentos	Mais que 10 anos
Mobilidade	Até 500 km/h
Latência de dados do usuário	1 ms para casos de uso na indústria, 4ms para MBB
Confiabilidade	Pelo menos 99,999%

Fonte: Adaptado de [2]

Outra mudança consideravel entre o 4G e o 5G é a criação de novas grades de serviço que visam contemplar os mais distintos cenários, aplicações e casos de uso. Os três principais são:

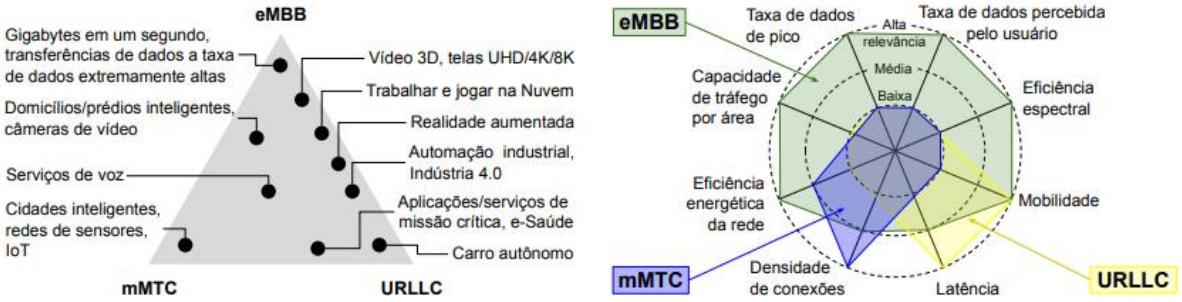
- eMBB que visa servir a centros metropolitanos densamente povoados com velocidades de downlink se aproximando de 1Gbit/s;
- mMTC habilita a comunicação máquina-máquina e aplicações de IoT que uma nova onda de clientes de redes sem fio podem vir a esperar de suas redes, tudo sem comprometer as outras classes de serviço;
- URLLC habilita as necessidades de comunicação críticas onde a largura de banda não é tão importante quanto a velocidade, especificamente uma latência fim-a-fim de $1ms$ ou menos;

A Figura 4 mostra alguns dos casos de uso de cada uma das três principais grades de serviço padronizadas nas redes 5G. É possível observar pela figura a), a esquerda, como alguns dos cenários não são definidos facilmente como pertencentes a apenas uma das três grades de serviço, possuindo características que podem ser uma combinação entre duas ou mais grades, como é o caso dos serviços de voz. Já no lado b) da figura, a direita, se apresentam as capacidades das três principais grades em termos dos requisitos da redes 5G.

O 5G apresenta a oportunidade para operadoras oferecerem conexão para residências em áreas onde fibra ótica é difícil ou muito cara de ser implementada. Evitando a necessidade de obras para instalação deste tipo de acesso que consumiriam muito tempo e recursos financeiros.

Embora seja um dos principais objetivos do 5G fornecer taxas de dados consideravelmente maiores em relação à gerações anteriores, um aspecto pelo menos tão importante quanto é a capacidade de gerenciar uma grande quantidade de equipamentos de IoT comunicando-se de maneira simultânea [1], bem como ser utilizada como alternativa à algumas arquiteturas de redes industriais.

Figura 4 – (a)Capacidades a serem ofertadas e (b) requisitos definidos para as redes 5G.



Fonte: Adaptado de [7]

2.3.2 Core

O *core* das redes 5G precisa ser capaz de suportar o novo paradigma da arquitetura baseada em serviços para redes modulares e precisa fornecer uma experiência consistente para usuários de redes de acesso 3GPP ou não 3GPP. Além disso precisa ser adaptável à tecnologias baseadas em nuvem, computação de borda e serviços de comunicação máquina-máquina para habilitar a utilização de carros autônomos e linhas de montagem robotizadas, por exemplo.

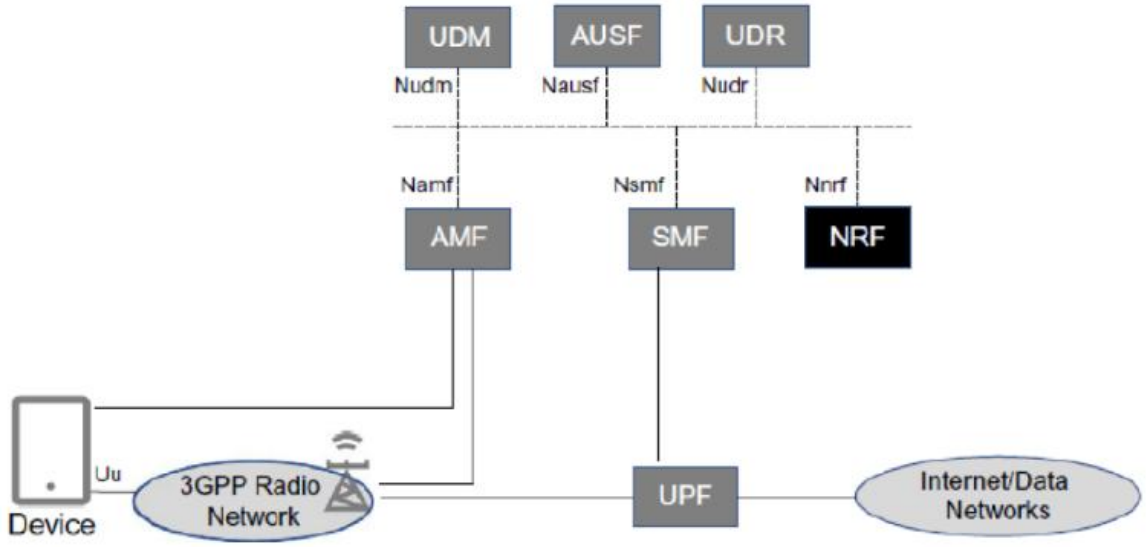
O 5GC traz consigo o conceito de interfaces baseadas em serviços. Isto significa que as funções de rede que incluem lógica e funcionalidade para processar fluxos de sinalização não são interconectadas através de interfaces ponto-a-ponto, ao invés disso expõem e tornam outros serviços disponíveis para outras funções de rede.

Para cada interação entre as funções de rede, uma delas age como consumidora de serviço e outra como produtora do seviço. A grande diferença no *core* 5G comparado com as gerações anteriores e suas arquiteturas tradicionais é o uso de interações baseadas em serviços entre as funções de rede [2].

As funções de rede (NFs - *Network Functions*) consumidoras podem localizar e contatar uma função produtora de serviço através do conceito de *Service Discovery*. Dentro da arquitetura 5G, este registro de consumidoras e produtoras é feito por uma função de rede dedicada denominada Função do Repositório de Rede (NRF - *Network Repository Function*). Uma outra maneira em que as funções de rede podem interagir é através do conceito de inscrição e notificação de serviços, desta forma é possível que uma função produtora de serviço possa enviar notificações para todos os consumidores quando algum critério pré-estabelecido é atingido. Esta estratégia evita que os consumidores enviem mensagens periódicas para requisitar o serviço, permitindo que o produtor apenas os notifique quando necessário.

Embora existam diversas NFs no núcleo 5G, as utilizadas durante o processo de simulação em adição à outras relevantes para este trabalho se encontram descritas a seguir. As funções de rede consideradas essenciais podem ser visuzalizadas na Figura 5.

Figura 5 – Funções de rede essencias do *core 5G*



Fonte: Adaptado de [2]

2.3.2.1 AMF

A função de acesso e mobilidade (AMF - *Access and Mobility Function*) interage com a rede de acesso via rádio e com os equipamentos através das interfaces N2 e N1 respectivamente. A AMF tem suporte à conexões de sinalização encriptadas com os aparelhos, permitindo que estes que registrem-se, sejam autenticados e mudem entre diferentes células de rádio na rede. Também permite alcançar e ativar equipamentos que estão em modo ocupado [2].

2.3.2.2 AUSF

A funcionalidade da função de autenticação de servidor (AUSF - *Authentication Server Function*) é consideravelmente limitada, mas muito importante. Fornece serviços para a autenticação de equipamentos específicos, neste processo utilizando credenciais criadas pela função de gerenciamento de dados (UDM - *Unified Data Management*). Além disso, a AUSF fornece serviços para a geração de material criptográfico para permitir atualizações de segurança das informações de rotas e outros parâmetros nos dispositivos [2].

2.3.2.3 NRF

É possível, utilizando a função de rede NRF, descobrir as funções de dados de usuário (UPFs - *User Plane Functions*) disponíveis na rede. Neste caso a função de gerenciamento de sessão (SMF - *Session Management Function*) pode questionar a NRF e a resposta recebida é uma lista de UPFs juntamente com algumas informações básicas sobre cada

uma das UPFs, como nome da rede de dados (DNN - *Data Network Name*) e fatia de rede que cada uma das UPF suporta [2].

Isto reduz a necessidade de informações pré-configuradas na SMF. Por outro lado, a informação que pode ser aprendida da NRF é bastante limitada e não contém, por exemplo, informações detalhadas sobre a topologia das UPFs, portanto, para casos de uso mais avançados, ter informação pré-configurada na SMF é a opção preferida.

2.3.2.4 NSSF

A função de seleção de fatia (NSSF - Network Slice Selection Function) tem o único papel de oferecer suporte a seleção de fatias de rede baseada na combinação de valores S-NSSAI (*Single - Network Slice Selection Assistance Information*) definidos na rede e requisitados por cada equipamento permitido na inscrição [2].

2.3.2.5 NWDAF

A função de análise de dados da rede (NWDAF - *Network Data Analysis Function*) coleta dados de outras funções de rede usando serviços que expõem eventos oferecidos por essas NFs. Também coleta dados de sistemas, operações e gerenciamento (O&M - *Operations and Management*) e dados relacionados a inscrição do repositório unificado de dados (UDR - *Unified Data Repository*). Os serviços oferecidos pela NWDAF podem, em teoria, ser consumidos por qualquer outra NF e até por aplicações externas. Consumidores primários da NWDAF são NSSF e a função de controle de políticas (PCF - *Policy Control Function*).

A análise feita pela NWDAF na coleta de dados pode, por exemplo, ser um resumo de dados históricos ou estatísticos ou uma tentativa de prever valores de dados futuros. Os dados analíticos podem ser usados por outras funções de rede para aplicar certas ações na rede, como selecionar uma modificação em uma fatia específica ou modificação de QoS para um serviço [2].

2.3.2.6 PCF

A função de controle de políticas (PCF) realiza controle de políticas relacionadas ou não a sessão de dados, isto é, a maneira como cada usuário específico pode acessar a rede. Um exemplo é através da restrição da área geográfica em que cada usuário pode se ligar ou mover dentro da rede mantendo conectividade. Também pode ser utilizada para definir qual tecnologia de acesso um usuário pode utilizar [2].

A PCF é construída a partir das interações com outras funções de rede como UDR e AMF. As políticas podem ser vistas como regras de como usuários, sessões e fluxos de dados devem ser controlados ou gerenciados, incluindo quais serviços são permitidos ou não, como a tarifação deve ser feita, qual qualidade de serviço aplicar, etc.

2.3.2.7 SMF

A SMF, ou função de gerenciamento de sessões, como o nome sugere, gerencia as sessões dos equipamentos. Isto inclui estabelecimento, modificação e liberação de sessões individuais e alocação de endereços IP por sessão. Além disso controla as diferentes UPFs na rede através da interface N4, este controle inclui configuração de desvio e direcionamento de tráfego na UPF para sessões individuais.

Ainda faz parte das funções da SMF as tarefas relacionadas a tarifação na rede, coletando seus próprios dados de tarifação e controlando a funcionalidade de tarifação na UPF [2].

2.3.2.8 UDM

A UDM gera os dados para autenticar os equipamentos da rede. Também autoriza o acesso de usuários específicos baseado em dados de inscrição. Isto pode significar, por exemplo, aplicar diferentes regras de acesso para usuários remotos e locais.

No caso de haver mais de uma instância de AMF e SMF na rede, a UDM mantêm o registro de qual instância está servindo qual equipamento em específico [2].

2.3.2.9 UDR

A UDR, o repositório de dados unificado, é a base de dados onde vários tipos de dados são armazenados. A função da UDR de armazenar e acessar dados é oferecido à outras funções de rede, especificamente UDM, PCF e,de maneira adicional, à função de exposição de rede (NEF - *Network Exposure Function*).

2.3.2.10 UPF

A função de plano de usuário, ou UPF, tem como tarefa principal processar e encaminhar dados de usuário. A funcionalidade da UPF é controlada pela SMF. A UPF é conectada com redes IP externas e age como um ponto de ancoragem estável para os equipamentos em relação a estas redes externas. A UPF realiza vários tipos de processamentos e encaminhamento de dados. Gerando relatórios de uso de dados para a SMF, que são depois incluídos nos relatórios de tarifação e outros [2].

Além de agir como um ponto de interconexão entre a sessão PDU e as redes de dados, as seguintes funções são oferecidas pelas instâncias UPF:

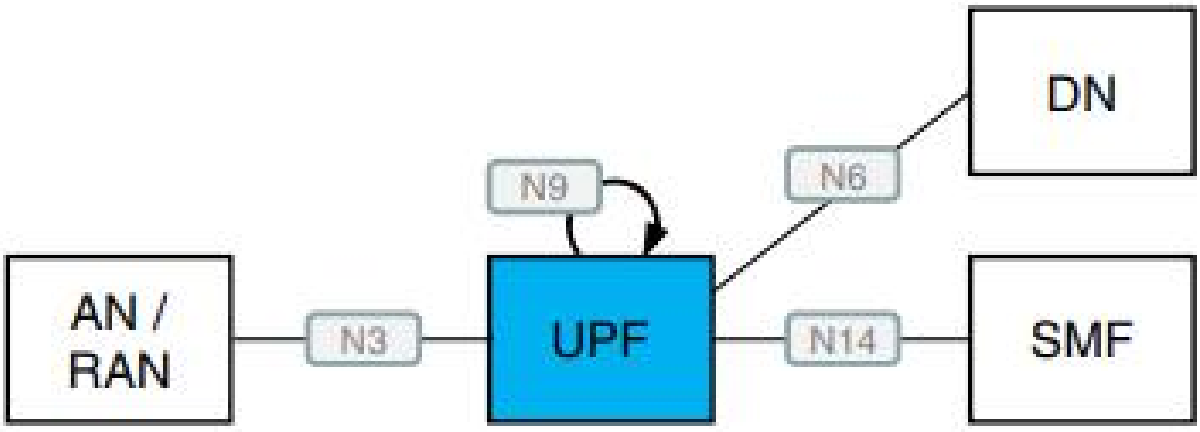
- detecção de tráfego incluindo detecção de aplicações;
- encaminhamento de tráfego baseado em regras definidas pela SMF;
- carregamento de dados e encaminhamento para garantir a integridade dos dados durante um *handover*;

- aplicação de QoS;
- relatório de uso de recursos (a UPF reporta o uso de de recursos para a SMF através da interface N4);
- relatório de eventos baseados em vários gatilhos e condições; e
- replicação do tráfego do plano de usuário para monitoramento [11].

A UPF pode aplicar QoS e marcação de pacotes em direção a rede de acesso via rádio ou em direção à redes de dados. Esta funcionalidade pode ser utilizada pela rede de transporte para lidar com cada pacote de maneira individual e com a devida prioridade em caso de congestionamento na rede.

O plano de usuário consiste na concatenação de várias partes. No ponto de vista da UE, é a primeira conexão com plano de usuário através da tecnologia de acesso utilizada (por exemplo NG-RAN). No ponto de vista da rede acesso é a conexão entre o core, via interface N3 ou N9, até a rede de dados (via N6) como é possível ver na Figura 6.

Figura 6 – Interfaces da UPF.



Fonte: Adaptado de [1]

2.3.3 Rede de acesso

A porção de rede de acesso via rádio de redes móveis consiste de diversas estações rádio-base, cada uma servindo transmissão e recepção de informação digital em uma ou diversas células, onde uma célula neste contexto se refere a menor parte de uma área geográfica total que a rede atua [2]. Estações base são localizadas em lugares cuidadosamente selecionados para otimizar a capacidade geral de cobertura dos serviços móveis. Isto significa que em áreas onde muitos usuários estão presentes, por exemplo no centro de uma cidade, a demanda precisa ser suprida posicionando estações base mais próximas

umas das outras e portanto permitindo mais (e menores) células; enquanto que nas áreas rurais, onde não são tantos os usuários presentes, as células normalmente são maiores para cobrir grandes áreas com o menor número de estações base possível. Um ponto a se observar é que mesmo em áreas urbanas são utilizadas células menores em conjunto com células maiores, ocorrendo sobreposições e intersecções a fim de atender usuários com alta e baixa mobilidade.

O projeto do core da rede 5G e da tecnologia de acesso ao rádio foi feita em paralelo e em cooperação. Um princípio chave do projeto da arquitetura do core 5G da 3GPP foi não porporcionar compatibilidade com versões de gerações anteriores de rede de acesso (GSM, WCDMA e LTE) ao invés disso uma nova tecnologia foi criada a New Radio (NR).

No 5G, as bandas de frequência suportadas variam de $450MHz$ até $6GHz$ (classificada como *Frequency Range 1, FR1, mid/low band* ou *sub 6 GHz bands*) e de 24,250 GHz até 52,600 GHz (classificadas como *Frequency Range 2, FR2, high band ou above 6 GHz bands*). A grande inovação são as altas frequências, também chamadas de *millimiter wave (mmWave)*, capazes de transferir uma grande quantidade de dados com uma latência muito baixa. No entanto, quanto maior a frequência de transmissão, menor é o alcance das ondas, ficando suscetíveis a perda de sinal causadas por objetos comuns, como portas e janelas, e até mesmo pela presença de indivíduos ou animais entre o emissor e o receptor [2].

2.3.3.1 NR

A 3GPP define uma entidade lógica denominada gNB. De fato, "gNB" é utilizada somente quando se referindo a estação rádio-base NR conectada a um *core* 5G. As estações base são interconectadas pela interface Xn, que consiste da parte de sinalização Xn-C e de transferência de dados Xn-U. Todas as estações base estão conectadas à uma ou mais AMFs e UPFs no core da rede pelas interfaces N2 e N3 [2].

Os dados de usuários são trasferidos entre estações base e UPFs utilizando redes IP. A pilha de protocolos completa é mostrada em Figura 7 e é a mesma para as interfaces Xn-U e N3. GTP-U (*General Packet Radio Service Tunneling Protocol User*) é um protocolo bem conhecido, uma vez que é utilizado em todas as gerações de sistemas móveis anteriores e porpociona um serviço de comunicação de dados confiável. GTP-U é utilizado por cima de uma pilha UDP/IP convencional que por sua vez é executada por cima de protolos de camada 2, tipicamente *Ethernet*.

A 3GPP escolheu a forma de onda CP-OFDM (Cycle Prefix-Orthogonal Frequency Division Multiplexing) para o 5G NR. A modulação por multiplexação por divisão de frequência ortogonal (OFDM) foi escolhida como base para o 5G por uma série de razões: ter eficiencia espectral tanto no DL (*Downlink*) quanto no UL (Uplink) para atender aos requisitos de altas taxas de dados, [...] utilização fluente de MIMO e [...] combinada com

Figura 7 – Pilha de protocolos da RAN 5G

Fonte: Adaptado de [2]

beamforming, OFDM contribui para compensar as perdas de propagação das ondas de rádio em bandas de alta frequência [2].

2.3.3.2 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 [2].

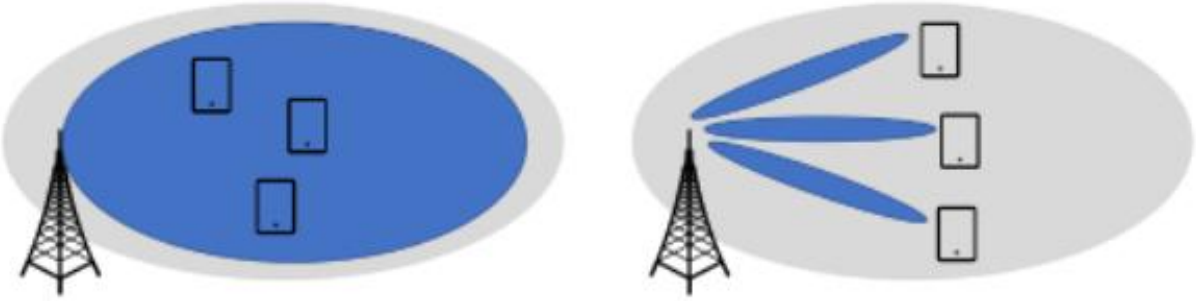
As técnicas de feixes múltiplos, onde há múltiplos feixes partindo da antena, cada um cobrindo uma menor porção da célula possuem feixes dinamicamente controláveis e direcionáveis, o que é usado para maximizar a performance através da otimização das características do link de rádio para conexão com cada dispositivo. Um exemplo destas técnicas pode ser vizualizado na Figura 8. A esquerda, se observa um exemplo de feixe único, cobrindo a célula por completo e a direita, um exemplo de *beamforming*.

2.3.3.3 MIMO

O princípio básico do MIMO é utilizar canais separados e não correlacionados entre as antenas do transmissor e do receptor a fim de transmitir diferentes fluxos de dados nos mesmos recursos físicos [11]. No lado do receptor, há uma combinação ou seleção do melhor sinal visando aumentar a qualidade do sinal. Os sistemas de rádio 5G tipicamente combinam essas duas técnicas.

MIMO de usuário único (SU-MIMO - *Single User MIMO*) significa transmitir duas ou mais cópias do mesmo fluxo de dados em direções ligeiramente diferentes usando *Beam-*

Figura 8 – Técnicas de feixe único e feixes múltiplos.

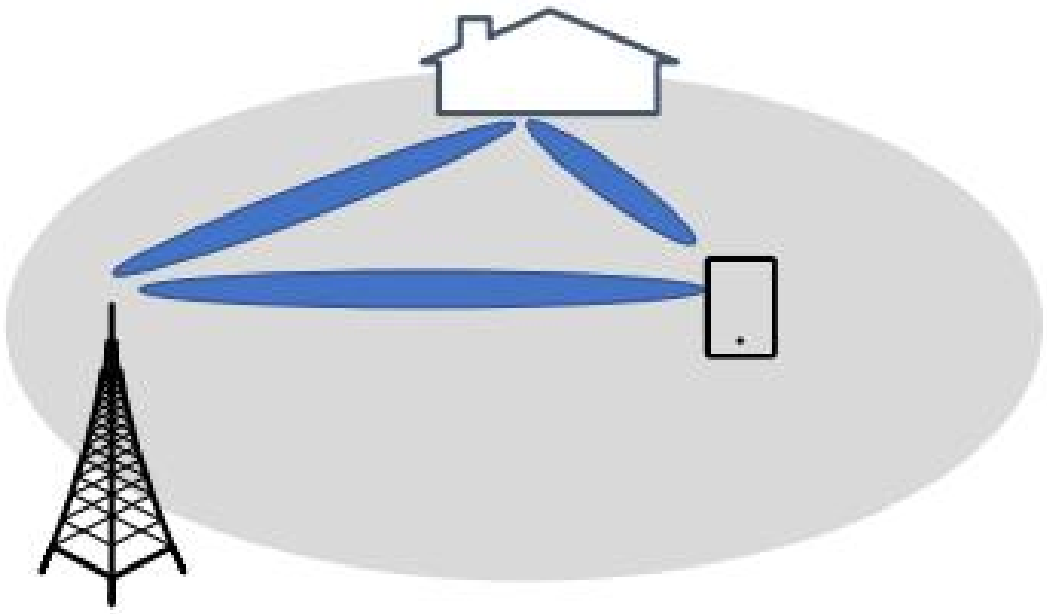


Fonte: Adaptado de [1]

forming, 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 9, o UE recebe um sinal da antena diretamente e outro sinal que é refletido do prédio, intensificando o sinal recebido final [2].

Figura 9 – MIMO de usuário único

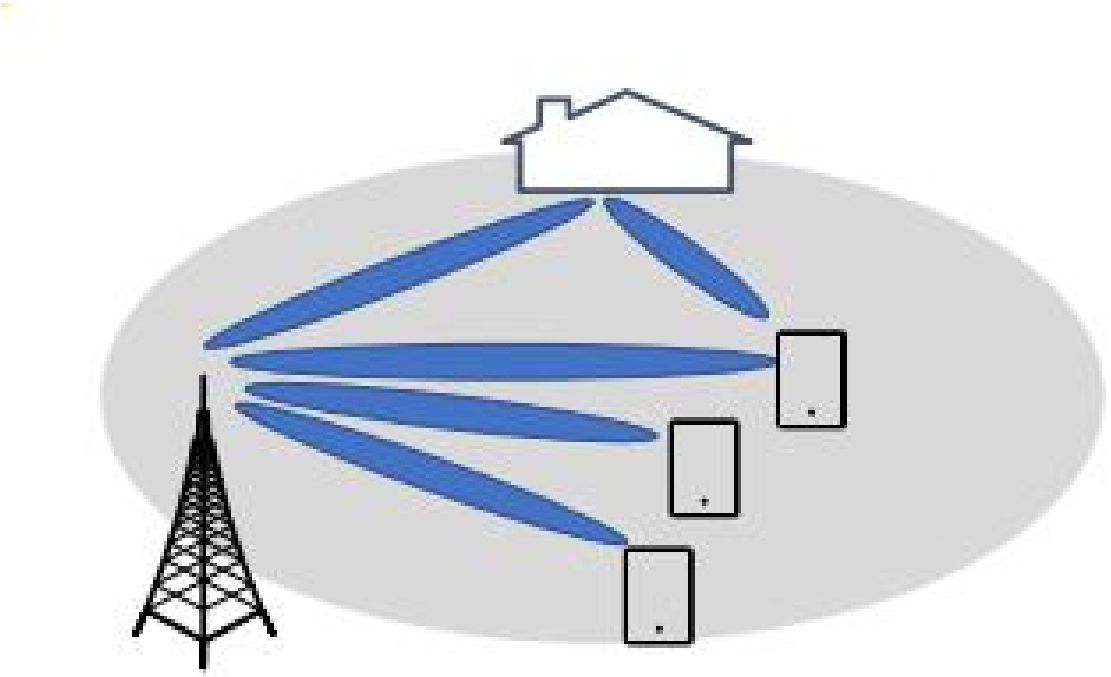


Fonte: Adaptado de [1]

Além do SU-MIMO há o MIMO multiusuário (MU-MIMO - *Multi User MIMO*), em que o objetivo não é otimizar o desempenho para um único usuário, mas sim obter um alto rendimento agregado para vários usuários [2]. Esta técnica é utilizada em situações de grande carga na rede, quando se deseja otimizar a 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 10.

Figura 10 – MIMO de usuários múltiplos



Fonte: Adaptado de [1]

2.3.4 Equipamento de usuário

Equipamento de usuário, no contexto de redes 5G, é uma terminologia geral que descreve qualquer sistema que possui conectivade com uma rede 5G. Sendo ele um equipamento estático, como um servidor em um *data center* ou um sensor em uma aplicação industrial, um sistema embarcado como em um smartphone ou drone, ou até mesmo uma aplicação virtualizada como máquinas virtuais, *containers* e servers virtuais.

2.3.5 Sessão PDU

Uma das tarefas principais do sistema 5G é prover ao UE conectividade com a rede de dados. A rede de dados pode ser a *internet*, uma rede de dados específica de operadora ou uma rede de dados dedicada vertical (como por exempleo de uma fábrica). As sessões PDUs (*Protocol Data Unit*) podem ser de dois tipos principais: as sessões PDU do tipo IP (*Internet Protocol*) e as sesões PDU do tipo ETH (*Ethernet*).

As sessões PDU do tipo IP oferecem um serviço de camada 3 incluindo a atribuição de endereços IP, filtragem e QoS baseada em fluxo IP[...]. As sessões PDU do tipo IP são esperadas por serem as sessões PDUs mais comumente utilizadas para conectividade com a internet comum [11].

A sessão PDU do tipo Ethernet oferece serviços de camada 2 incluindo controle de endereço MAC (*Media Access Control*) e/ou QoS baseado em VLAN (*Virtual LAN*), encaminhamento e filtragem de frames. As sessões PDU do tipo Ethernet são esperadas para serem utilizadas nas aplicações industriais, em serviços de LAN (*Local Area Network*) (como por exemplo quando o 5G é utilizado dentro de uma empresa).

Quando o UE solicita o estabelecimento de uma Sessão de PDU, esta 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) [2].

2.3.6 Fatiamento de rede

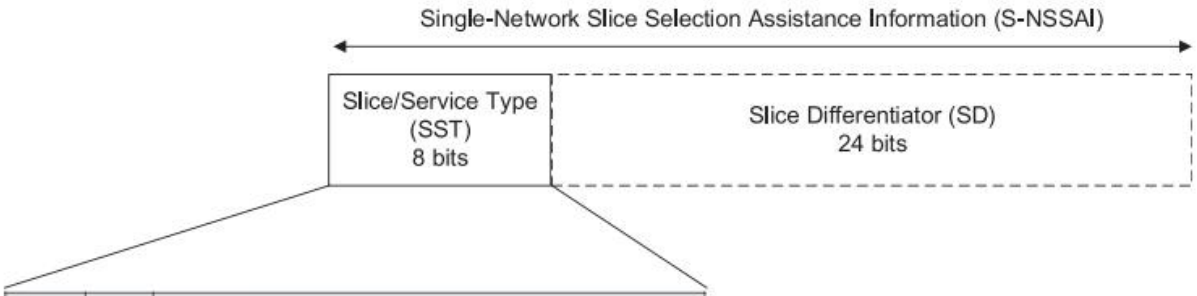
Redes tradicionais e sua abordagem onde uma única rede deve servir para todos os propósitos precisa ser adaptada a fim de que o grande número de casos de uso de implementação e várias aplicações possam ser suportadas. Portanto, ao invés de usar uma única rede monolítica servindo múltiplos propósitos, avanços tecnológicos como virtualização e SDN permitem construir redes lógicas dentro de uma camada de infraestrutura comum e compartilhada. Estas redes lógicas são denominadas fatias de rede [2].

A 3GPP define uma fatia de rede como sendo constituída por uma rede de rádio e um core de rede, onde algumas partes desses recursos podem ser compartilhados por várias fatias, enquanto algumas partes podem ser exclusivas de uma fatia em particular. Uma fatia de rede em particular é identificada por um parâmetro chamado S-NSSAI, consistindo por sua vez de dois sub-parâmetros: o tipo de fatia (SST - *Slice Type*) e o opcional diferenciador de fatia (SD - Slice Diferentiatior) [2].

O SD é utilizado para diferenciar entre múltiplas fatias do mesmo tipo. A parte SST é mandatória e indica o tipo de características da fatia de rede. O valor de intervalo do SST inclui é padronizada conforme 11.

O fatiamento de rede é um dos principais aspectos de um ambiente 5G completamente funcional e otimizado. Este fornece os meios para projetar e implementar sistemas de

Figura 11 – Formato do S-NSSAI



0		Reserved
1	eMBB	Network Slice suitable for handling of 5G enhanced Mobile Broadband
2	URLLC	Network Slice suitable for handling of ultra-reliable low latency communications
3	MIoT	Network Slice suitable for handling of massive IoT
4	V2X	Network Slice suitable for handling of V2X services
5–127		Reserved
128–255		Operator-specific

Fonte: Adaptado de [2]

comunicação customizados e integrar serviços de segmentos de negócios[1]. Uma fatia de rede pode ser projetada com os recursos necessários para atender a uma certo conjunto de especificações (como por exemplo sensores estacionários enviando poucos e infrequentes dados), enquanto outra fatia na mesma rede pode ser projetada para outro conjunto de especificações (como por exemplo *streaming* de video de alta qualidade) [11].

2.4 Virtualização

Tradicionalmente, os elementos funcionais do *core* das redes móveis são aplicações distribuídas que escalam horizontalmente e que são rodadas em *hardware* dedicado. O primeiro grande passo da virtualização foi migrar estas aplicações para a utilização de recursos virtualizados como máquinas virtuais (VM - *Virtual Machines*) e, posteriormente, *containers*.

Pelo fato da aplicação não estar mais restringida pelos recursos e capacidades de um equipamento físico, este passo permitiu uma flexibilidade muito maior na implementação destas aplicações. No caso específico de redes e telecomunicações, o uso destas técnicas permite que os projetos de rede possam ser realizados e implementados em escalas muito maiores ou muito menores.

Já os sistemas 5G têm como objetivo proporcionar um maior nível de abstração projetado para simplificar a gestão e operacionalização da rede. Além do mais, novos serviços precisarão ser implementados rapidamente conforme novos modelos de negócios emergem

e fazem com que operadoras mudem para redes programáveis e definidas por *software* [2]. As técnicas de virtualização utilizadas neste trabalho são apresentadas a seguir.

2.4.1 Máquinas Virtuais

A virtualização a nível de sistema permite à instâncias de um sistema operacional rodar simultaneamente por cima de um *hypervisor*. Um hypervisor é um parte de *software* que cria e roda máquinas virtuais. Existem dois tipos de *hypervisors*: *hypervisor* de paravirtualização (tipo 1) e hypervisor com emulação (tipo 2).

Um *hypervisor* de tipo 1 é um programa executado diretamente em uma plataforma de hardware que hospeda VMs vinculadas a sistemas operacionais que foram editados para que as instruções passadas a essas VMs sejam executadas diretamente nesta plataforma de *hardware*. Esta plataforma pode suportar sistemas operacionais convidados (*guests*) através de seus *drivers*. Os hypervisores clássicos nesta categoria incluem VMware vSphere, VMware ESX, Microsoft Hyper-V, KVM entre outros [12].

Já um *hypervisor* de tipo 2 é um programa executado na plataforma de *hardware* de sistemas operacionais nativos, ou seja sem realizar modificações no sistema nativo. Este sistema operacional, quando convidado pelo *hypervisor*, é executado no equipamento graças a um emulador para que o dispositivo subjacente leve em consideração todas as construções. Os sistemas operacionais convidados não estão cientes que são virtualizados, portanto não requerem nenhuma modificação, diferentemente do tipo 1. São exemplos de deste tipo: Microsoft Virtual PC, Oracle VM Virtual Box, VMware Player, QEMU entre outros.

2.4.2 Containers

Containers são outro modelo de virtualização, isolados uns dos outros e compartilhando os *kernels* do sistema operacional. Containers são amplamente utilizados em setores onde há a necessidade de otimizar recursos de *hardware* para se rodar múltiplas aplicações e para aumentar a flexibilidade e produtividade [2].

Containers são especialmente úteis para aplicações no setor de telecomunicações:

- onde baixa latência, resiliência e portabilidade são requisitos chave (por exemplo em ambientes de computação de borda);
- para a implementação de serviços de vida curta, em aplicações de implementação altamente ágil;
- em aprendizado de máquina ou inteligência artificial quando é útil dividir um problema em tarefas menores, é esperado, portanto, que containers irão auxiliar, em certa extensão, a automação;

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 [12].

2.4.3 Redes definidas por software

SDN se refere a arquitetura de rede que tem sido projetada para minimizar as limitações de hardware pela abstração de funções baixo nível. Ao invés disso, as redes definidas por software tornam possível executar estas funções de maneira baseada em software, com plano de controle centralizado via APIs (*Application Programming Interface*) [1].

Especificamente para o 5G, o conceito de SDN pode ser usado como uma estrutura que proporciona o funcionamento do plano de controle, otimizando assim a transmissão de dados. Alguns dos benefícios do conceito de SDN são a otimização da largura de banda e o aprimoramento da performance de latência na rede bem como o roteamento e reroteamento de dados, que acontecem praticamente em tempo real via SDN, o que diminui consideravelmente as interrupções e contribui para o desenvolvimento de seviços de alta disponibilidade que são desejados para aplicações de tempo crítico.

2.4.4 Virtualização de funções de rede

Juntamente com o conceito de SDN, a NFV tem um importante papel para porporcionar performance ótima no 5G. A principal tarefa da NFV é desacoplar o *software* do *hardware* pela aplicação de VMs ou Containeres. Os benefícios da utilização desse conceito são as características extremamente dinâmicas, então uma vez necessário, uma VM (ou *container*) é criado para a respectiva função automaticamente, o que resulta em economia de gastos uma vez que o operador de tal rede não precisa investir em um equipamento *standalone* que tipicamente é baseado em arquiteturas de *hardware* proprietárias [1]

Outro ponto a ser observado em NFV é que esta tecnologia permite a implementação do conceito de fatiamento de rede. O fatiamento de rede permite a divisão de diversas redes lógicas dentro de uma mesma rede baseada em uma única estrutura física.

Além disso, a flexibilidade e automação no instanciamento de funções de rede via *software* alavanca outras tecnologias de rede como as baseadas na nuvem. Como um benefício adicional de NFV, o funcionamento e gerenciamento de conjuntos de funções de rede melhora a escala de performance da rede [1]. Portanto torna-se uma alternativa viável e lógica utilizar conceitos de virtualização para projeto, instanciamento e gerenciamento de redes 5G.

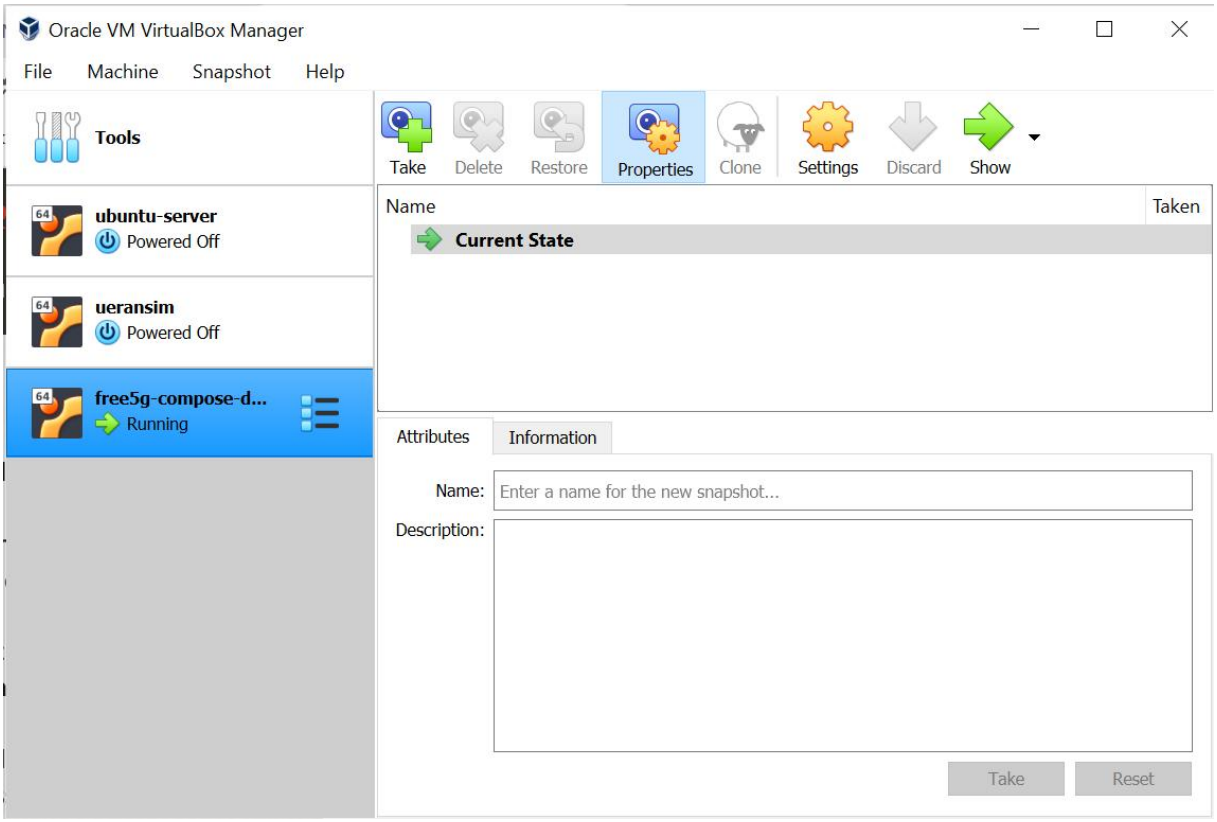
2.5 Ferramentas de simulação

Nas subseções seguintes são descritas as principais ferramentas utilizadas durante a etapa de simulação do trabalho.

2.5.1 Oracle VM VirtualBox

O *software* Oracle VM VirtualBox é o *software* de virtualização multi-plataforma de código aberto (*open source*) mais popular do mundo e permite com que desenvolvedores entreguem soluções de código mais rapidamente rodando múltiplos sistema operacionais em um único equipamento [13]. A aplicação do VirtualBox conta com uma interface gráfica intuitiva e funcional permitindo ao usuário configurar, clonar, restaurar e inicializar VMs de maneira rápida. Esta interface se encontra na Figura 12.

Figura 12 – Interface gráfica de usuário do Oracle VM VirtualBox



Fonte: O Autor

O pacote básico possui uma licença do tipo GPL-2.0 (General Public License v2.0), isto é, livre de taxas para utilização e comercialização.

2.5.2 Free5GC

O *Free5GC Compose* é uma versão em Docker Compose do *Free5GC*, que por sua vez é um projeto em código aberto para simulação de processos e aplicações do *core* 5G mantido pela Universidade Nacional Chiao Tung (NCTU - National Chiao Tung University), localizada em Hsinchu, Taiwan. O objetivo final do projeto *Free5GC* é implementar o core da rede 5G definido na Release 15 (R15) e posteriores. A licença do *Free5GC* é a

Apache 2.0, isto é, qualquer um pode usar o *Free5GC* para fins comerciais sem pagar taxas [14].

No estágio 3, o *Free5GC* apresenta um *core* 5G completamente funcional e suporta funcionalidades de operação, administração, e gerenciamento (OAM - *Operation, Administration and Management*) além de orquestração. Foram adicionadas as funções N3IWF e a funcionalidade de classificador de Uplink (ULCL - *UpLink Classifier*) [14], além das já suportadas funções: NSSF, NRF, UDM, PCF, AUSF, AMF, SMF, e UPF.

Além dos *containers* contendo cada uma das NFs isoladas, o *Free5GC Compose* possui um container com a implementação de uma interface *Web* (WebUI). A WebUI permite cadastrar os dispositvos de usuário no Core com uma série de configuraçãoes 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, endereço 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*) [15].

2.5.2.1 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 [16], o kernel GTP5G é responsável pelas camadas, como a camada de transporte. O protocolo de plano de usuário para tunelamento GPRS é identificado em cada nó com um identificador do terminal do túnel (TEID - *Tunnel Endpoint Identifier*), um endereço IP e um número de porta no Protocolo de Datagrama do Usuário (UDP *User Datagram Protocol*). Assim, o GTP-U é necessário para permitir o encaminhamento de pacotes entre entidades GTP-U, conforme a 3GPP TS 29.244 [17].

2.5.2.2 MongoDB

MongoDB é um *software open source* de banco de dados orientado a documentos e multiplataforma, escrito na linguagem C++. Classificado como um programa de banco de dados NoSQL, o MongoDB usa documentos semelhantes a JSON com esquemas. É desenvolvido pela MongoDB Inc. e publicado sob uma combinação das licensas GNU Affero General Public License e Licença Apache [18].

MongoDB é a base de dados do tipo não-relacional mais amplamente utilizada e é parte do *framework* de empresas como Bosch, Cisco, Humanitix e Toyota. A aplicação

MongoDB é rodada através de um container no free5gc-Compose como base de dados para as NFs implementadas.

2.5.3 UERANSIM

O UERANSIM 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 [19].

O projeto posssui três interfaces principais na perspectiva UE/RAN: interface de controle (entre RAN e AMF), interface de usuário (entre RAN e UPF) e interface de rádio (entre UE e RAN). O plano de controle tem duas interfaces: NAS (*Non-Access Stratum*), que está sob o controle da UE, e NGAP (*NG Application Protocol*), que está sob o controle da RAN [19].

2.5.4 Scapy

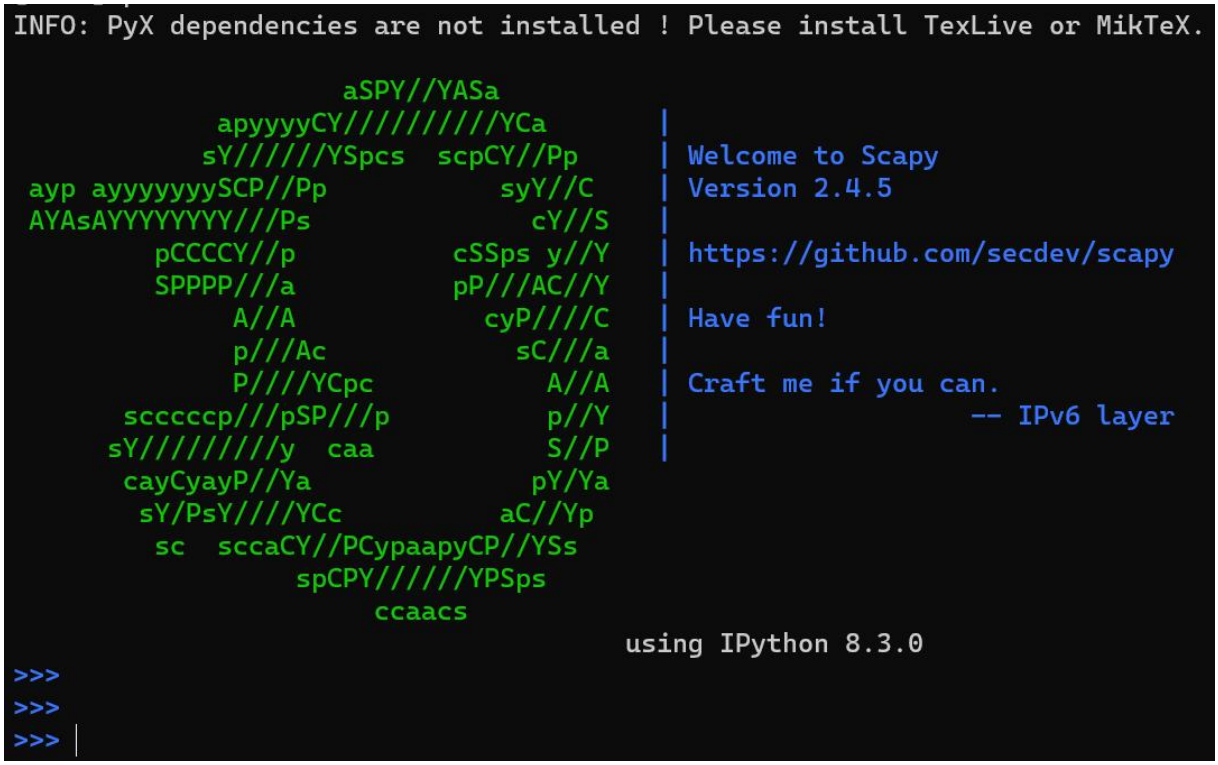
Scapy é um poderoso programa interativo de manipulação de pacotes. É capaz de forjar e decodificar pacotes de um grande número de protocolos, enviá-los, capturá-los, casar requisições e respostas, entre outros [20]. Uma das funcionalidades mais interessantes do Scapy é a capacidade de criar pacotes customizáveis, especificando as diferentes camadas e protocolos de rede, bem como alterar cada um dos campo dos cabeçalhos destes pacotes.

Essa criação e customização é facilitada com o uso da interface por linha de comando demonstrada pela Figura 13 e pela sintaxe simples e direta, uma vez que o Scapy é implementado em Python. Outra funcionalidade que auxilia na criação de fluxos de pacotes e a função *rdpcap* em que é possível ler arquivos de gravação de tráfego (.pcap), arquivos estes que podem ser gerados em capturas de ferramentas como Tcpdump e Wireshark.

Outra funcionalidade implementada pela biblioteca *scapy* que foi utilizada neste trabalho é o método *sniff*. Esta função é responsável por capturar pacotes e armazená-los em uma estrutura que pode, posteriormente ser manipulada e tratada utilizando *Python*.

No código fonte 2.1 é possível observar o commando sniff para a captura de cem pacotes, esta quantidade é configurada através do parâmetro *count*. O comando se utiliza de um filtro que seleciona somente pacotes cujo protocolo da camada de transporte é UDP (*User Data Porotocl*) e cujas portas são 67 ou 68, ambas referentes ao protocolo

Figura 13 – Interface de linha de comando Scapy para distribuições Linux



Fonte: O Autor

DHCP (*Dynamic Host Configuration Protocol*) e vindos do endereços IP de origem 8.8.8.8 através do parâmetro *filter*.

Código Fonte 2.1 – Exemplo de utilização do comando *sniff* da biblioteca *scapy*

```
1 pacotes = sniff(filter="src 8.8.8.8 udp port 67 or 68", iface="eth0", count
    =100)
```

Fonte: O Autor

Além disto este comando filtra qual interface de rede desejada para interceptar os pacotes, neste caso *eth0*, uma interface do tipo *Ethernet* padrão, presente em sistema operacionais Linux. Por fim, todos estes pacotes interceptados são armazenados em uma variável *pacotes* para posterior tratamento e manipulação.

3 Descrição e Configuração do Sistema de Simulação

A metodologia para a construção de um ambiente de simulação que realiza a classificação do tráfego no core da rede 5G em grades de serviço e as ferramentas e técnicas utilizadas para tal se encontram descritas neste capítulo. Se inicia por uma visão geral do ambiente de simulação, posteriormente se mostra as configurações nos simuladores free5GC-compose e UERANSIM e por fim são descritas as funções que realizam o envio, captura e classificação do tráfego, bem como a classificação considerada ideal para esta amostra de dados.

3.1 Ambiente de simulação

Para o ambiente de simulação foram criadas duas VMs no programa Virtual Box rodando em um sistema hospedeiro Windows 10 e suas configurações, em conjunto com as especificações do computador hospedeiro, são compiladas na Tabela 2. Em ambas VMs foram instalados sistemas operacionais Ubuntu Server 20.04 com uma Unidade de Processamento Central (CPU - Central Processing Unit) virtualizada de 1 núcleo e 30 GB de armazenamento de disco. A única diferença entre as VMs reside na quantidade de memória RAM (Random Access Memory): na VM1 são 2048 MB e na VM2 são 1024 MB.

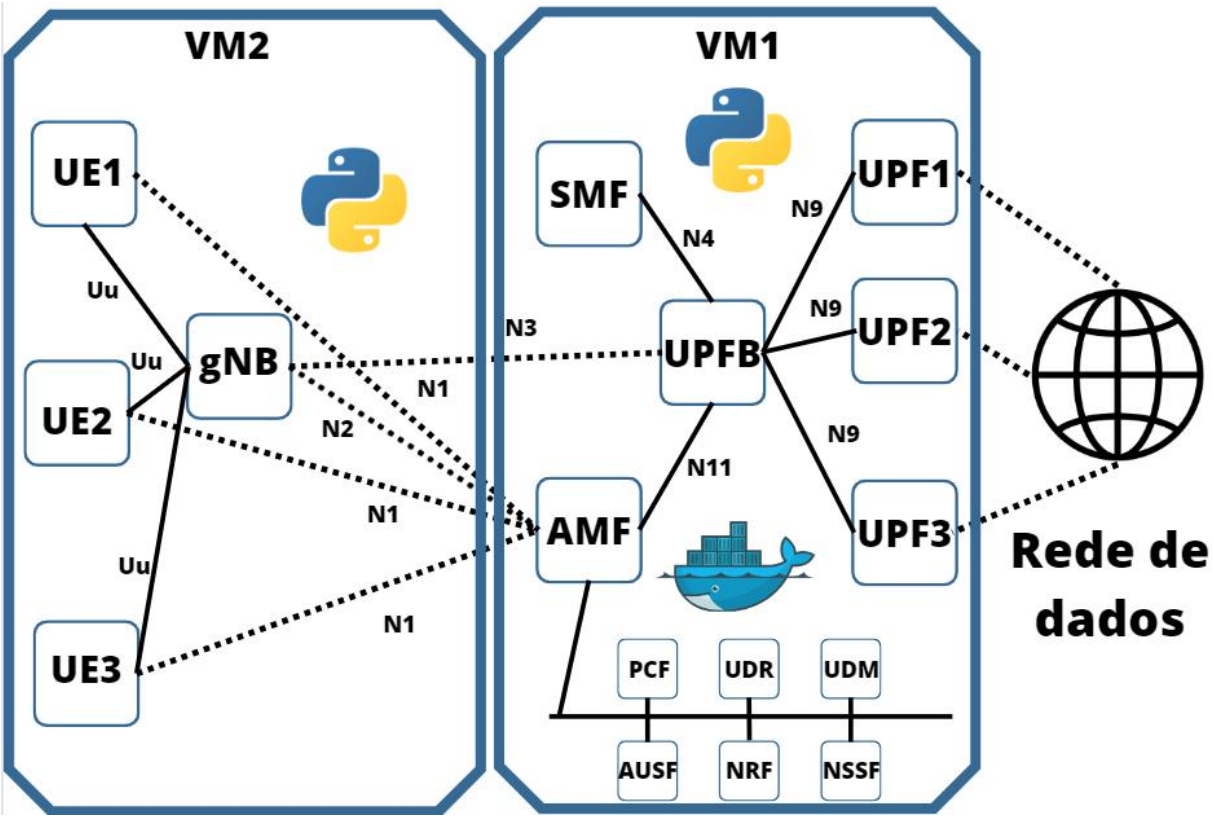
Tabela 2 – Configurações das VMs e PC hospedeiro.

Aplicação	Endereço IP	SO	Memória RAM	Núcleos	Armazenamento em disco
Free5GC Compose	192.168.56.120	Ubuntu Server 20.04	2048 MB	1 un	30 GB
UERANSIM	192.168.56.121	Ubuntu Server 20.04	1024 MB	1 un	30 GB
Computador Hospedeiro	192.168.100.8	Windows 10	8 GB	8 un	512 GB

Fonte: O Autor

A Figura 14 mostra as conexões entre as VMs e os programas instalados em cada uma. Como observado, a VM1 contêm o simulador free5GC, que por sua vez instancia todos os conteineres contendo as NFs do core da rede virtualizada bem como aplicações adicionais como a WebUI e a base de dados MongoDB. Além disso a VM 1 também possui a biblioteca Scapy para classificar o tráfego provindo dos UEs (VM2). Já a VM2 possui o UERANSIM, onde são criados três equipamentos de usuário (para simular o drone, o smartphone e o sensor na Figura 1) além da biblioteca Scapy para simular o tráfego enviado ao core (VM1).

Figura 14 – Ambiente de simulação utilizado.



Fonte: O Autor.

3.2 Criação das VMs e instalação de programas

A criação e configuração das VMs foi feita com base nos passos em [21] e [22]. Primeiramente se instalou no computador hospedeiro o programa Virtual Box e foi baixada imagem do Ubuntu Server. Posteriormente foram criadas as VMs no *software* de virtualização com base no quadro 2 contendo dois adaptadores de rede cada uma, sendo o primeiro deles do tipo tradução de endereço da rede (NAT - Network Address Translation) e o segundo do tipo *Host-Only Adapter*. Além disso a imagem do Ubuntu Server foi carregada para dentro das VMs e as mesmas foram iniciadas.

O código fonte 3.1 demonstra como mudar o *hostname* da VM1 para *free5gc-compose*. O mesmo foi feito para VM2, mudando-o para *UERANSIM*. Destaca-se ainda que as permissões de super usuário foram utilizadas com o comando *sudo*.

Código Fonte 3.1 – Modificação do hostname da VM1

```
1 $ sudo vi /etc/hostname
2 free5gc-compose
```

Fonte: O Autor

Além disto o nome de domínio absoluto (FQDN - *Fully Qualified Domain Name*) de ambas foi mudado para corresponder ao novos nomes, conforme código fonte 3.2 que

mostra o exemplo para a VM1. Nota-se que a configuração foi feita para o endereço IP 127.0.0.1, também chamado de endereço de *loopback*, um endereço IP que permite ao *host* direcionar tráfego para si mesmo.

Código Fonte 3.2 – Modificação do FQDN da VM1

```
1 $ sudo vi /etc/hosts
2 127.0.0.1 localhost
3 127.0.1.1 free5gc-compose
```

Fonte: O Autor

No passo seguinte, definiu-se endereços IPs estáticos para a interface de rede *Host-Only Adapter* (*enp0s8*) e a configuração utilizando o protocolo DHCP foi desabilitada nesta mesma interface conforme código fonte 3.3, onde é mostrado o exemplo para a VM1. Tendo definido o endereço IP estático na VM, foram utilizados os comandos *netplan try* e *netplan apply* para checar se o endereço IP é válido e para aplicar as configurações conforme código fonte 3.3. Ainda neste código fonte um processo de *reboot* foi realizado para confirmar as alterações configuradas.

Código Fonte 3.3 – Comandos para tornar estático os endereços IPs das VMs

```
1 \$ sudo vi / etc / netplan /00 - installer - config . yaml
2 network :
3  ethernets :
4    enp0s3 :
5       dhcp4 : true
6     enp0s8 :
7       dhcp4 : false
8       addresses : [192.168.56.120/24]
9   version : 2
10
11 \$ sudo netplan try
12 ...
13 Press Enter ...
14 ...
15 \$ sudo netplan apply
16 ...
17 \$ ifconfig
18 ...
19 \$ sudo shutdown -r now
```

Fonte: O Autor

Por fim, foram realizados os comandos *ping* e *ifconfig* conforme o código fonte 3.4. O comando ping é utilizado para inferir se a VM possui conectividade com a internet através do envio de pacotes ICMP a um endereço IP na rede, neste caso *google.com*. Já

o comando *ifconfig* lista os endereços IPs internos pertencentes às interfaces de rede da VM e, para que possa ser utilizado, primeiramente deve-se instalar o pacote *net-tools*.

Código Fonte 3.4 – Exemplos dos comandos ping e ifconfig

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

Fonte: O Autor

Tendo posse do endereço IP da VM1 (192.168.56.120) e da VM2 (192.168.56.121), é possível utilizar o protocolo SSH (Secure Shell), através de algum programa de *prompt* de comando no computador hospedeiro. Utilizando o acesso remoto é possível atualizar as VMs utilizando o código fonte 3.5, onde são feitos além do *update* e *upgrade*, um processo de *reboot* para confirmar as alterações e atualizações.

Código Fonte 3.5 – Comandos para realizar o *update, upgrade e reboot* da VM

```
1 $ sudo apt update
2 ...
3 $ sudo apt update
4 ...
5 $ sudo shutdown -r
6 ...
```

Fonte: O Autor

3.3 Configuração do free5GC compose

Primeiramente, acessando a VM1 via SSH, é clonado o repositório do *free5GC Compose* localizado em [15] e as aplicações básicas são instaladas conforme [21]. Outro módulo instalado para o funcionamento do *core* 5G simulado é o GTP5, cujo repositório, localizado em [23] também é clonado. Após clonados os dois repositórios necessários, ambas aplicações são instaladas e mais uma vez a VM é reinicializada.

No simulador Free5gc Compose cada NF é instanciada através de um container e possui sistema de nome de domínio (DNS - *Domain Name System*) de um endereço IP local privado associado. Assim, a primeira edição necessária foi no arquivo de configurações da SMF, onde o endereço IP fixo da VM1 foi utilizado para a interface N6. Originalmente é utilizado o DNS associado ao conteiner da UPFB, como pode ser visto na linha 1 de 3.6. Além disso, os nós da UPF1,UPF2 e UPF3 também foram definidos por seus nós limítrofes seguindo a topologia apresentada na Figura 1.

Código Fonte 3.6 – Trecho alterado do arquivo de configurações da SMF

```
  UPFB:  # the name of the node
      type: UPF # the type of the node (AN or UPF)
      node_id: upfb.free5gc.org # the IP/FQDN of N4 interface on this UPF (PFCP)
      sNssaiUpfInfos: # S-NSSAI information list for this UPF
        - sNssai: # S-NSSAI (Single Network Slice Selection Assistance Information)
            sst: 1 # Slice/Service Type (uinteger, range: 0~255)
            sd: 010203 # Slice Differentiator (3 bytes hex string, range: 000000~FFFFFF)
          dnnUpfInfoList: # DNN information list for this S-NSSAI
            - dnn: internet
              pools:
                - cidr: 60.60.0.0/16
      interfaces: # Interface list for this UPF
        - interfaceType: N3 # the type of the interface (N3 or N9)
          endpoints: # the IP address of this N3/N9 interface on this UPF
            - upfb.free5gc.org
          networkInstance: internet # Data Network Name (DNN)
    UPF1:
      type: UPF
      node_id: upf1.free5gc.org
      interfaces:
        - interfaceType: N9
          endpoints:
            - upf1.free5gc.org
          networkInstance: internet # Data Network Name (DNN)

    UPF2:
      type: UPF
      node_id: upf2.free5gc.org
      interfaces:
        - interfaceType: N9
          endpoints:
            - upf2.free5gc.org
          networkInstance: internet # Data Network Name (DNN)
    UPF3:
      type: UPF
      node_id: upf3.free5gc.org
      interfaces:
        - interfaceType: N9
          endpoints:
            - upf3.free5gc.org
          networkInstance: internet # Data Network Name (DNN)
  links: # the topology graph of userplane, A and B represent the two nodes of each link
```

```
43        - A: gNB
44          B: UPFB
45        - A: UPFB
46          B: UPF1
47        - A: UPFB
48          B: UPF2
49        - A: UPFB
50          B: UPF3
```

Fonte: O Autor

Para a configuração do 5GC no *free5GC Compose*, o segundo arquivo editado foi o arquivo de configuração do Docker Compose, nele o container *upf* teve seu nome atualizado para *upfb* e três novos conteineres foram instanciados (upf1, upf2 e upf3), adequando o arquivo a arquitetura mostrada na fig:topologia. É possível observar as mudanças no código fonte 3.7 onde os conteineres upfb e upf1 são mostrados nas linhas 21 a 37 e 2 a 17, respectivamente. Vale ressaltar que os conteineres upf2 e upf3 são cópias do upf1, tendo apenas o nome modificado.

Código Fonte 3.7 – Trecho alterado do arquivo de configuração dos *conteiners* do simulador para adição dos novos planos de usuário

```
1  services:
2    free5gc-upf-1:
3      container_name: upf1
4      build:
5        context: ./nf_upf
6        args:
7          DEBUG_TOOLS: "false"
8      command: bash -c "./upf-iptables.sh && ./free5gc-upfd -f ../config/
       upfcfg.yaml"
9      volumes:
10       - ./config/upfcfg1.yaml:/free5gc/config/upfcfg.yaml
11       - ./config/upf-iptables.sh:/free5gc/free5gc-upfd/upf-iptables.sh
12     cap_add:
13       - NET_ADMIN
14     networks:
15       privnet:
16         aliases:
17           - upf1.free5gc.org
18
19 ...
20
21   free5gc-upf-b:
22     container_name: upfb
23     build:
24       context: ./nf_upf
25       args:
```

```
26          DEBUG_TOOLS: "false"
27      command: bash -c "iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
        && ./free5gc-upfd -f ../config/upfcfg.yaml"
28      volumes:
29        - ./config/upfcfgb.yaml:/free5gc/config/upfcfg.yaml
30      cap_add:
31        - NET_ADMIN
32      ports:
33        - "2152:2152/udp"
34      networks:
35        privnet:
36          aliases:
37            - upfb.free5gc.org
38
39 ...
40
41  free5gc-amf:
42      container_name: amf
43      build:
44        context: ./nf_amf
45        args:
46          DEBUG_TOOLS: "false"
47      command: ./amf -amfcfg ../config/amfcfg.yaml
48      expose:
49        - "8000"
50      volumes:
51        - ./config/amfcfg.yaml:/free5gc/config/amfcfg.yaml
52      environment:
53        GIN_MODE: release
54      ports:
55        - "38412:38412/sctp"
56      networks:
57        privnet:
58          aliases:
59            - amf.free5gc.org
60      depends_on:
61        - free5gc-nrf
```

Fonte: O Autor

Ainda no mesmo arquivo, a porta 2152 utilizada pelo protocolos GTP-U e UDP para o estabelecimento do túnel de dados do usuário durante a sessão PDU e a porta 38412 utilizada pelo protocolo SCTP (*Stream Control Transmission Protocol*) foram abertas para a UPFB e para a AMF, respectivamente. As outras portas abertas, tais quais a porta 8000 referente ao protocolo HTTP, a porta 5000 utilizada pelo conteiner da WEBUI e a porta 27017 utilizada pelo mongoDB não foram modificadas. Outra observação é que o conteiner n3iwf não foi utilizado para esta simulação, por ser referente a função que

adequa padrões não 3GPP ao core da rede, o que foge do escopo deste trabalho.

No código fonte 3.7 é possível observar que as funções UPFs descritas implementam o arquivo *upf-iptables.sh*, um *shell script* que contêm algumas regras de configuração de *firewall*. Este arquivo pode ser visto em sua totalidade no código fonte 3.8. A primeira regra cria uma condição para pacotes que estão prestes a sair do core pela interface *eth0* para endereços IPs dinâmicos e a segunda cria uma condição para aceitar o encaminhanto dos pacotes vindos nesta interface.

Código Fonte 3.8 – Arquivo contendo regras de *firewall* para o roteamento de pacotes no *core*

```
1 #!/bin/bash
2 #
3 # Configure iptables in UPF
4 #
5 iptables -t nat -A POSTROUTING -o eth0  -j MASQUERADE
6 iptables -I FORWARD 1 -j ACCEPT
```

Fonte: O Autor

Realizadas todas estas alterações, o *core* simulado pode ser agora inicializado através do comando contido no código fonte 3.9. É possível visualizar que a aplicação da WEBUI é funcional e que, através dela, é possível adicionar e editar os UEs conectados ao *core*, bem como suas configurações de usuário tais quais: a rede móvel terrestre pública (PLMN - *Public Land Mobile Network*), o 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.

Código Fonte 3.9 – Comando para instanciar os *containers* do core

```
1 $ sudo docker-compose up
```

Fonte: O Autor

3.4 Configuração do UERANSIM

O simulador UERANSIM, assim como o free5gc Compose na VM1 teve o repositório, disponível em [19], clonado na VM2 e as aplicações básicas instaladas. O simulador possui alguns arquivos de configuração a fim de integrar-se a aplicações de core simulado, no entanto apenas dois foram modificados neste trabalho.

O primeiro deles, o arquivo de configurações relacionado ao simulador da Antena 5G foi alterado conforme o código fonte 3.10. O endereço IP estático da VM2 foi configurado como endereço IP da simulação do link de rádio bem como para as interfaces N2 e N3. Neste arquivo ainda foi inserido o endereço IP estático da VM1 como endereço da AMF, corroborando mais uma vez com o diagrama da Figura 1.

Código Fonte 3.10 – Arquivo para a configuração da antena 5G simulada

```
mcc: '208'          # Mobile Country Code value
mnc: '93'           # Mobile Network Code value (2 or 3 digits)

nci: '0x000000010'  # NR Cell Identity (36-bit)
idLength: 32        # NR gNB ID length in bits [22...32]
tac: 1              # Tracking Area Code

linkIp: 192.168.56.121   # gNB's local IP address for Radio Link Simulation (Usually same with local IP)
ngapIp: 192.168.56.121   # gNB's local IP address for N2 Interface (Usually same with local IP)
gtpIp: 192.168.56.121    # gNB's local IP address for N3 Interface (Usually same with local IP)

# List of AMF address information
amfConfigs:
  - address: 192.168.56.120
    port: 38412

# List of supported S-NSSAIs by this gNB
slices:
  - sst: 0x1
    sd: 0x010203

# Indicates whether or not SCTP stream number errors should be ignored.
ignoreStreamIds: true
```

Fonte: O Autor

O segundo arquivo a ser editado corresponde ao arquivo de configurações do UE conforme código fonte 3.11. No campo que especifica a antena que fará a conexão com o UE foi colocado o endereço IP fixo da VM2 e, para o UE1, todos os outros parâmetros de inscrição foram mantidos. Como são três os equipamentos de usuário conectados ao *core* nesta simulação, nos outros dois arquivos de equipamento de usuários (UE2 e UE3), apenas o valor do SUPI foi modificado.

Código Fonte 3.11 – Arquivo para a configuração do equipamento de usuário 1

```
# IMSI number of the UE. IMSI = [MCC|MNC|MSISDN] (In total 15 digits)
supi: 'imsi-208930000000003'
# Mobile Country Code value of HPLMN
mcc: '208'
# Mobile Network Code value of HPLMN (2 or 3 digits)
mnc: '93'

# Permanent subscription key
```

```
key: '8baf473f2f8fd09487cccbd7097c6862'
# Operator code (OP or OPC) of the UE
op: '8e27b6af0e692e750f32667a3b14605d'
# This value specifies the OP type and it can be either 'OP' or 'OPC'
opType: 'OPC'
# Authentication Management Field (AMF) value
amf: '8000'
# IMEI number of the device. It is used if no SUPI is provided
imei: '356938035643803'
# IMEISV number of the device. It is used if no SUPI and IMEI is provided
imeiSv: '4370816125816151'

# List of gNB IP addresses for Radio Link Simulation
gnbSearchList:
  - 192.168.56.121

# UAC Access Identities Configuration
uacAic:
  mps: false
  mcs: false

# UAC Access Control Class
uacAcc:
  normalClass: 0
  class11: false
  class12: false
  class13: false
  class14: false
  class15: false

# Initial PDU sessions to be established
sessions:
  - type: 'IPv4'
    apn: 'internet'
    slice:
      sst: 0x01
      sd: 0x010203

# Configured NSSAI for this UE by HPLMN
configured-nssai:
  - sst: 0x01
    sd: 0x010203

# Default Configured NSSAI for this UE
default-nssai:
  - sst: 1
    sd: 1
```

```
56
57 # Supported integrity algorithms by this UE
58 integrity:
59   IA1: true
60   IA2: true
61   IA3: true
62
63 # Supported encryption algorithms by this UE
64 ciphering:
65   EA1: true
66   EA2: true
67   EA3: true
68
69 # Integrity protection maximum data rate for user plane
70 integrityMaxRate:
71   uplink: 'full'
72   downlink: 'full'
```

Fonte: O Autor

Após feitas estas configurações, a antena 5G simulada e os três equipamentos de usuário podem ser inicializados usando os comandos do código fonte 3.12 em terminais diferentes. Ao executar estes comandos após a inicialização do *core* 5G simulado na VM1 (conforme código fonte 3.9), é possível observar a criação de três túneis para tráfego de dados de usuário (*uesimtun0*, *uesimtun1* e *uesimtun2*), criando três interfaces N1 que ligam cada um dos UEs com a AMF, uma interface N2 que liga a antena a AMF e a interface N3 que liga a RAN com a UPFB no lado do *core*.

Código Fonte 3.12 – Comandos para inicializar as aplicações da antena e dos equipamentos de usuário simulados

```
1 # inicia a antena
2 sudo build/nr-gnb -c config/free5gc-gnb.yaml
3 # inicia UEs
4 sudo build/nr-ue -c config/free5gc-ue1.yaml
5 sudo build/nr-ue -c config/free5gc-ue2.yaml
6 sudo build/nr-ue -c config/free5gc-ue3.yaml
```

Fonte: O Autor

3.5 Função para envio de tráfego

Na VM2, utilizada para a simulação dos equipamentos de usuário e rede de acesso ao rádio, o código fonte 3.13 foi utilizado para realizar a instalação completa da biblioteca *Scapy*. Biblioteca esta escolhida por ser de utilização intuitiva e por ser escrita na lingua-

gem de programação *python*, já pré-instalada em sistemas operacionais Ubuntu. Para a instalação foi utilizado o gerenciador de pacotes *pip*, nativo do *python*.

Código Fonte 3.13 – Commando para a instalação completa das aplicações da biblioteca *scapy*

```
1 $ sudo python3 -m pip install --pre scapy[complete]
```

Fonte: O Autor

Além de contar com uma interface por linhas de comando (CLI - *Command Line Interface*) que permite a criação de pacotes de dados costumizáveis, esta biblioteca conta com uma função denominada *rdpcap* que permite fazer a leitura de arquivos do tipo *pcap*. Esta extensão de arquivos é comumente gerada por programas de captura de tráfego como *tcpdump* e *wireshark*. Esta função se mostrou muito útil para esta simulação, possibilitando obter arquivos de captura de tráfego reais como base de dados para a classificação. Estes arquivos podem ser encontrados como exemplos em repositórios de programas de captura, como os disponibilizados de maneira aberta em [24].

Três destes arquivos são utilizados como base para a criação da função *send_traffic.py* mostrada no código fonte 3.14. Nela são lidos três arquivos *pcaps*:

- *tcp-ethereal-file1.trace*: uma comunicação utilizando majoritariamente pacotes do protocolo HTTP com vários segmentos TCP, representando um tráfego com características da grade de serviço eMBB.
- *mqtt_packets_tcpdump.pcap*: uma captura do protocolo MQTT (MQ Telemetry Transport), muito utilizado para comunicações do tipo máquina a máquina ou mMTC.
- *djiuav.pacp*: uma captura de mais de 25000 pacotes contendo protocolos de controle de um *drone*, tráfego este que possui características da grade URLLC.

Utilizando laços de iteração em conjunto com a biblioteca *scapy* é possível alterar o endereço IP de destino destes pacotes para o endereço da VM1 e o endereço fonte para o endereço IP corresponde à cada túnel de dados criado para cada equipamento de usuário. Pontua-se também que apenas os 3999 primeiros pacotes foram utilizados no caso do tráfego do drone, uma vez que estes eram os pacotes que continham mensagens do protocolo DJIUAV (DJI Unmmaned Aerial Vehicle), o tráfego subsequente tem uma predominância de pacotes de transmissão de vídeo, com mais características *broadband*.

Código Fonte 3.14 – Função em python para envio de tráfego através dos túneis de usuário

```
1 #! /usr/bin/python3
2 from scapy.all import *
3 def send_traffic():
```

```
4     #embb
5     http = rdpcap("/home/ubuntu/pcaps/embb/tcp-ethereal-file1.trace")
6     for i in range(len(http)):
7         http[i].payload.src="60.60.0.1"
8     #mmtc
9     mqtt = rdpcap("/home/ubuntu/pcaps/mmtc/mqtt_packets_tcpdump.pcap")
10    for i in range(len(mqtt)):
11        mqtt[i].payload.src="60.60.0.2"
12    #urllc
13    drone = rdpcap("/home/ubuntu/pcaps/urllc/djiuav.pcap")
14    drone = drone[0:4000]
15    for i in range(len(drone)):
16        drone[i].payload.src="60.60.0.3"
17    all_data = http + mqtt + drone
18    print(f"numero de pacotes: {len(all_data)}")
19    traff = all_data[0]*len(all_data)
20    for i in range(len(traff)):
21        traff[i] = all_data[i].payload
22        traff[i].dst="192.168.56.120"
23        send(traff[i])
24        print(f"sending packet {i}")
25    print(all_data)
```

Fonte: O Autor

Tendo definida esta função, basta executá-la na VM2, mantendo a execução da antena simulada e dos equipamentos de usuário conforme 3.12 e o *core* na VM1 conforme 3.9, que o tráfego simulado será gerado e enviado à VM1.

3.6 Script para captura e classificação de tráfego

Enquanto a VM2, hospedeira do UERANSIM, envia tráfego simulado para o 5GC, é necessário capturá-lo, para, posteriormente classificá-lo. Esta é uma tarefa feita na VM1 em duas etapas: primeiramente se intercepta o tráfego utilizando o comando *sniff* e o conteúdo interceptado é redirecionado à uma variável *capture*; posteriormente se classifica o conteúdo desta variável utilizando scripts em linguagem *python*.

3.6.1 Captura utilizando *sniff*

Após simular o tráfego na VM2 é preciso interceptar o tráfego proveniente dos túneis de usuário, criados pela interação dos simuladores UERANSIM e free5gc-compose, na VM1. Para isto foi utilizado o comando *sniff* dentro de um laço *for* na VM1 conforme código fonte 3.15.

Código Fonte 3.15 – Laço de iteração com o comando *sniff* para capturar tráfego vindo dos túneis de usuário

```
1 for i in range(drone_len+mqtt_len+http_len):
2     capture = sniff(filter="src 60.60.0.1 or src 60.60.0.2 or src 60.60.0.3
    ", iface="enp0s8", count=1)
3     print(f"Packet {i} captured")
```

Fonte: O Autor

No código fonte 3.15 é possível observar o commando sniff para a captura de um único pacote através do parâmetro *count*. O comando se utiliza de um filtro que seleciona somente pacotes vindos dos endereços IP de origem correspondentes aos túneis de dados de usuário através do parâmetro *filter*.

Além disto este comando filtra qual a interface de rede desejada para interceptar os pacotes, neste caso *enp0s8*. Por fim, todos estes pacotes interceptados são armazenados em uma variável *capture* para posterior tratamento e manipulação.

É válido ressaltar que as variáveis *drone_len, mqtt_len+http_len* são correspondentes a quantidade de pacotes de dados de cada um dos arquivos *.pcap* utilizados na simulação. Além disso o comando *print* é utilizada apenas para auxiliar no acompanhamento da captura dos pacotes dinamicamente, mostrando na tela do terminal qual pacote foi capturado e será, posteriormente, classificado.

3.6.2 Classificação usando *scapy*

Apropriando-se da mesma biblioteca *scapy* utilizada para simular o tráfego na VM2, é possível classificá-lo na VM1. Algumas abordagens podem ser seguidas e optou-se por três.

Na primeira e na segunda, simples condições do tipo *if/else* definem se o pacote em questão deve ser tratado como pertencente à grade de serviço eMBB, mMTC ou URLLC. Na terceira, mais parâmetros são analisados dentro do pacote e um vetor de pesos define a relevância de cada uma destas características para a classificação em cada uma das grades de serviço.

Embora as primeiras sejam de mais simples entendimento, estas podem não ser tão eficientes com amostras de dados mais complexas. A terceira abordagem é muito mais customizável e generalizável que as duas primeiras, porém requer um refinamento na definição dos índices do vetor.

3.6.2.1 Abordagem 1

No código fonte 3.16 é possível identificar a primeira abordagem, onde apenas duas características do pacote são utilizadas para a classificação: o protocolo da camada de transporte e o tamanho do *payload*, ou carga útil. Neste primeiro algoritmo se considera

que o tráfego do tipo URLLC se utiliza somente do protocolo UDP na camada de transporte, critério este assumido devido ao requisito de latência de dados de usuário menores ou iguais à $1ms$ e ao fato de que o UDP é um protocolo não-confiável e não orientado à conexão, ou seja, a utilização deste protocolo diminui a latência fim-a-fim devido a não possuir mecanismos de controle de recebimento e não retransmitir pacotes.

Código Fonte 3.16 – Função *classify_packets_v1.py*

```
#! /usr/bin/python3
from scapy.all import *
def classify_packets(traff, min_load=120):
    URLLC=MMTC=EMBB=0
    if traff[0].payload.proto==6:
        if len(traff[0].payload.load)<=min_load:
            MMTC+=1
            print("Packet classified as mMTC")
        else:
            EMBB+=1
            print("Packet classified as eMBB")
    else:
        URLLC+=1
        print("Packet classified as URLLC")
    return EMBB,MMTC,URLLC
```

Fonte: O Autor

Já os pacotes que possuem o protocolo TCP em sua camada de transporte precisam de um parâmetro adicional para a diferenciação entre as classes eMBB e mMTC e este parâmetro é o tamanho do *payload* do pacote. Quando o tamanho do *payload* é inferior ou igual a um limite configurável (denominado *min_load* no código fonte 3.16) este pacote é classificado como mMTC. Isto se deve ao tráfego máquina a máquina (M2M - *Machine to Machine*) apresentar, normalmente, pacotes onde a área de dados possui poucos *bytes* e, portanto, uma carga útil menor [25, 26]

Por fim os pacotes cujo protocolo na camada de transporte é TCP e que são superiores à *min_load* são classificados como eMBB. Esta classificação, apesar de abrangente, se mostra coerente com as características de tráfego desta classe de serviço, uma vez que esta será a grade direcionada a usuários convencionais, que representam a maior parte dos usuários das redes 5G.

3.6.2.2 Abordagem 2

No código fonte 3.17 é possível identifcar a segunda abordagem, alguns critérios são avaliados para classificação, sendo eles: os protocolos da camada de transporte, as portas de origem e destino e o tamanho do *payload*. Alguns valores podem ser configurados neste função, são eles:

- *min_load*: o limite de carga útil mínima que dtermina se o *payload* é considerado pequeno,
- *embb_ports*: um vetor de valores inteiros com os números de portas referentes aos protocolos de aplicação que se deseja associar à grade de serviço eMBB,
- *urllc_ports*: um vetor de valores inteiros com os números de portas referentes ao protocolos de aplicação que se deseja associar à grade de serviço URLLC,
- *mmtc_ports*: um vetor de valores inteiros com os números de portas referentes ao protocolos de aplicação que se deseja associar à grade de serviço mMTC.

Código Fonte 3.17 – Função *classify_packets_v2.py*

```
#! /usr/bin/python3
from scapy.all import *

def classify_packets(traff,min_load=120, embb_ports=[80,443], urllc_ports=[67, 68], mmtc_ports=[]):
    URLLC=MMTC=EMBB=0
    #protocolo da camada de transporte
    if traff[0].payload.proto==6:
        proto="TCP"
    elif traff[0].payload.proto==17:
        proto="UDP"

    #portas de destino
    if traff[0].dport in embb_ports:
        dstport = "embb"
    elif traff[0].dport in urllc_ports:
        dstport = "urllc"
    elif traff[0].dport in mmtc_ports:
        dstport = "mmtc"
    else:
        dstport="df"
    #portas de origem
    if traff[0].sport in embb_ports:
        srcport = "embb"
    elif traff[0].sport in urllc_ports:
        srcport = "urllc"
    elif traff[0].sport in mmtc_ports:
        srcport = "mmtc"
    else:
        srcport="df"

    #tamanho do payload
    if len(traff[0].payload.load)<=min_load:
```

```
33          payload = "pequeno"
34      else:
35          payload = "comum"
36
37      #Classificar em grades de servico:
38
39      if srcport == "embb" or dstport == "embb":
40          EMBB+=1
41          print("Packet classified as eMBB (by port)")
42      elif srcport == "mmtc" or dstport == "mmtc":
43          MMTC+=1
44          print("Packet classified as mMTC (by port)")
45      elif srcport == "urllc" or dstport == "urllc":
46          URLLC+=1
47          print("Packet classified as URLLC (by port)")
48      else:
49          if proto == "TCP":
50              if payload == "pequeno":
51                  MMTC+=1
52                  print("Packet classified as mMTC (by TCP + small payload)")
53              else:
54                  EMBB+=1
55                  print("Packet classified as eMBB (by TCP)")
56          else:
57              URLLC+=1
58              print("Packet classified as URLLC (by UDP)")
59      return EMBB,MMTC,URLLC
```

Fonte: O Autor

Alguns exemplos de números de portas referentes à protocos de aplicação comuns podem ser vistos como valores padrão em 3.17. As portas 80 e 443 são utilizadas pelos protocolos HTTP e HTTPS (*Hypertext Transfer Protocol Secure*) comuns de navegação em páginas *web*, portanto associadas à classe eMBB neste exemplo, já as portas 67 e 68 são utilizada pelo protocolo DHCP [27], apenas como exemplo, associadas à URLLC por utilizarem UDP na camada de transporte.

Primeiramente se avalia algumas características presentes no pacote para a posterior classificação. Avalia-se o protocolo da camada de transporte como UDP ou TCP, avalia-se as portas de origem e destino e se as mesmas estão presentes nos vetores passados como parâmetro e por fim se classifica o tamanho do payload como pequeno ou comum.

A classificação se dá considerando que, caso o valor da porta de destino ou origem do pacote esteja no vetor *embb_ports*, este é classificado como eMBB; a mesma lógica é aplicada para caso o número da porta esteja nos vetores *urllc_ports* ou *mmtc_ports*, sendo classificados como URLLC e mMTC, respectivamente.

Como última condição deste estrutura de decisão *if/else*, todos os pacotes que não

atenderem às condições anteriores são classificados conforme a abordagem 1 como no código fonte 3.16, tendo como critério o protocolo da camada de transporte e o tamanho do *payload*.

3.6.2.3 Abordagem 3

E por fim, no código fonte 3.18, é possível identificar a terceira abordagem, baseada no conceito de vetor de pesos customizável. Assim como na segunda abordagem, os mesmos critérios são utilizados para a classificação, no entanto os parâmetros a serem passados para a função são distintos.

Para esta terceira abordagem, além dos valores *min_load* e dos vetores *embb_ports, urllc_ports* e *mmtc_ports*, um novo parâmetro é configurável nesta função: o vetor w, que possui elementos que determinam os pesos de influencia para cada um dos critérios de classificação. O significado dos índices do vetor w é dado da seguinte maneira:

0. : determina a influência do protocolo da camada de transporte ser TCP para a classificação do pacote como eMBB

1. : determina a influência do protocolo da camada de transporte ser TCP para a classificação do pacote como mMTC

2. : determina a influência do protocolo da camada de transporte ser TCP para a classificação do pacote como URLLC

3. : determina a influência do protocolo da camada de transporte ser UDP para a classificação do pacote como eMBB

4. : determina a influência do protocolo da camada de transporte ser UDP para a classificação do pacote como mMTC

5. : determina a influência do protocolo da camada de transporte ser UDP para a classificação do pacote como URLLC

6. : determina a influência do número da porta de origem para a classificação do pacote como eMBB

7. : determina a influência do número da porta de origem para a classificação do pacote como mMTC

8. : determina a influência do número da porta de origem para a classificação do pacote como URLLC

9. : determina a influência do número da porta de destino para a classificação do pacote como eMBB

10. : determina a influência do número da porta de destino para a classificação do pacote como mMTC

11. : determina a influência do número da porta de destino para a classificação do pacote como URLLC

12. : determina a influência do tamanho do *payload* ser considerado pequeno para a classificação do pacote como eMBB

13. : determina a influência do tamanho do *payload* ser considerado pequeno para a classificação do pacote como mMTC

14. : determina a influência do tamanho do *payload* ser considerado pequeno para a classificação do pacote como URLLC.

15. : determina a influência do tamanho do *payload* ser considerado comum para a classificação do pacote como eMBB

16. : determina a influência do tamanho do *payload* ser considerado comum para a classificação do pacote como mMTC

17. : determina a influência do tamanho do *payload* ser considerado comum para a classificação do pacote como URLLC.

Configurando estes pesos, cada um dos pacotes terá três coeficientes, um para cada grade de serviço (eMBB, mMTC e URLLC) que serão calculados com base nas características que apresenta em seus cabeçalhos e área de dados. Por fim, verifica-se qual dos três coeficientes é maior e se atribui àquele pacote a classificação da grade de serviço correspondente.

A estrutura *if/else* que classifica o tráfego final na classe de serviço correspondente tem como lógica considerar mMTC apenas os pacotes que possuírem esse índice maior que os índices eMBB e URRLC ao mesmo tempo. Já para os pacotes URRLC, basta que possuam um coeficiente maior ou igual a mMTC e obrigatoriamente maior que eMBB. Os pacotes eMBB, por serem considerados a classe mais abrangente, precisam apenas possuir o coeficiente eMBB maior ou igual a mMTC e maior ou igual a URLLC. Esta implementação pode ser vista em 3.18.

Código Fonte 3.18 – Função *classify_packets_v3.py*

```
1 #! /usr/bin/python3
2 from scapy.all import *
3 def classify_packets(traff, min_load=120,embb_ports=[80,443],
    urllc_ports=[67, 68], mmtc_ports=[], w =
    [0,0,0,0.1,0.3,0.7,0,0,0,0,0,0,0.1,0.9,0.3,0,0,0]):
4
```

```
5     URLLC=MMTC=EMBB=0
6     urllc=mmtc=embb=0
7
8     #protocolo da camada de transporte
9     if traff[0].payload.proto==6:
10        embb+=w[1]
11        mmtc+=w[2]
12        urllc+=w[3]
13    elif traff[0].payload.proto==17:
14        embb+=w[4]
15        mmtc+=w[5]
16        urllc+=w[6]
17
18    #portas de destino
19    if traff[0].dport in embb_ports:
20        embb+=w[10]
21    elif traff[0].dport in urllc_ports:
22        mmtc+=w[11]
23    elif traff[0].dport in mmtc_ports:
24        urllc+=w[12]
25    else:
26        embb+=w[10]
27
28    #portas de origem
29    if traff[0].sport in embb_ports:
30        embb+=w[13]
31    elif traff[0].sport in urllc_ports:
32        mmtc+=w[14]
33    elif traff[0].sport in mmtc_ports:
34        urllc+=w[15]
35
36    #tamanho do payload
37    if len(traff[0].payload.load)<=min_load:
38        embb+=w[19]
39        mmtc+=w[18]
40        urllc+=w[17]
41    else:
42        embb+=w[17]
43        mmtc+=w[18]
44        urllc+=w[19]
45
46    # CLASSIFICACAO:
47    if mmtc>=embb and mmtc>=urllc:
48        MMTC+=1
49        print("Packet classified as mMTC")
50    elif urllc>=embb:
51        URLLC+=1
```

```
52          print("Packet classified as URLLC")
53      else:
54          EMBB+=1
55          print("Packet classified as eMBB")
56      urllc=embb=mmtc=0
57      return EMBB,MMTC,URLLC
```

Fonte: O Autor

Assim sendo, é possível atribuir um valor entre 0 e 1 para cada um destes índices e desta maneira priorizar a influência de certos critérios sobre outros no momento da classificação. Um exemplo é o vetor w demonstrado em 3.18, onde os índices $3, 4$ e 5 possuem os valores $0.1, 0.3$ e 0.7 respectivamente, indicando que, uma vez que o pacote a ser classificado utilize o protocolo UDP na camada de transporte, há uma influência sete vezes maior para sua classificação final como URLLC e três vezes mais para mMTC em relação à sua classificação final como eMBB para este critério. Da mesma maneira os índices $12, 13$ e 14 que possuem os valores $0.1, 0.9$ e 0.1 respectivamente, indicando que o tamanho da carga útil do pacote ser considerada comum é uma característica nove vezes mais significativa para classificá-lo como mMTC do que este mesmo critério é significativo para classificá-lo como eMBB ou URLLC.

Esta abordagem é especialmente útil caso se tenha um conhecimento prévio das características do tráfego a ser classificado, podendo assim se ajustar os pesos conforme as particularidades do mesmo. Ou ainda no caso da integração desta função com uma técnica de aprendizagem supervisionada (SL - *Supervised Learning*), onde os pesos do vetor w são ajustados conforme o sistema "aprende" através de interações com uma base de testes categorizada e obtem resultados mais próximos aos ideais.

3.7 Classificação ideal

Com a finalidade de se obter resultados comparativos das três abordagens estrutradas anteriormente, é necessário criar uma classificação ideal. Em um contexto de *Network as a Service*, imagina-se um cenário onde as operadoras que oferem o serviço 5G criem classes de assinatura (ou planos) que reflitam de maneira aproximada as classes de serviço padronizadas. Sendo assim, o tráfego de determinados aparelhos pertenceria à um enlace com mais ou menos recursos a depender do plano contratado.

Em outras palavras, imagina-se um cenário onde uma operadora possa criar, como exemplo, um plano denominado Convencional (equivalente a eMBB) com maior largura de banda e velocidade de conexão à preços mais acessíveis que um plano denominado Premium (equivalente a URLLC) que deve entregar menores latências e maior confiabilidade, porém com valores mais altos. E ainda um plano Industria/SmartHome (equivalente a mMTC) onde os valores da assinatura podem variar conforme o número de equipamentos

em que se deseja conectar, disponibilizando categorias para usuários que querem conectar poucas dezenas de devices *smart* em uma residência, até empresas que desejam conectar milhões de equipamentos em uma grande unidade. Cada um destes planos possuirá enlaces separados com os recursos apropriados para atender aos requisitos demandados e realizará uma comparação do endereço IP do usuário para verificar em qual classe encaixá-lo.

Pensando nestes cenários, decidiu-se considerar a classificação ideal adotando como único critério o tipo de equipamento de usuário gerando aquele tráfego. Assim sendo, todo os pacotes vindos do *smartphone* pelo túnel *uesimtun0* devem ser classificados como eMBB, todos os pacotes gerados pelo sensor no túnel *uesimtun1* devem ser classificados como mMTC e, por fim, todos os pacotes enviados pelo drone via *uesimtun2* devem ser classificados como URLLC.

Feitas estas considerações, o resultado ideal para o algoritmo de classificação para os três arquivos *pcap* utilizados é: 218 eMBB, 19 mMTC e 3999 URLLC. Isto representa 5.1463%, 0.4485% e 94.4051% dos dados, respectivamente.

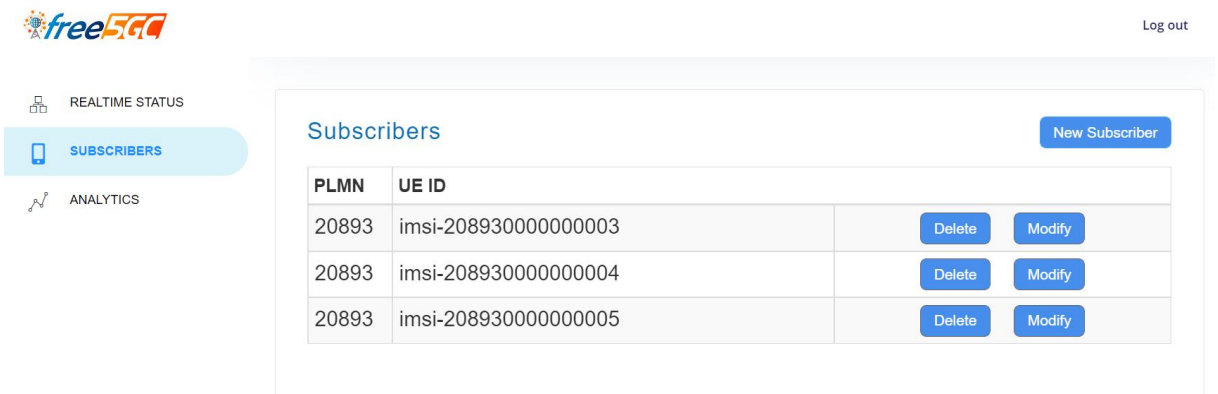
4 Resultados

Este capítulo detalha os resultados obtidos após realização de simulações utilizando os três algoritmos de classificação. Os resultados são dsicutidos e posteriormente comparados aos ideais.

4.1 Instanciamento do Core 5G e acesso à WebUI

Para verificar o correto funcionamento do *core* simulado através do *free5gc-compose* e dos equipamentos de usuário simulados com o *UERANSIM*, a aplicação de WebUI foi testada como é possível visualizar em 15. Ao acessar a interface é possível observar três usuários conectados ao core e verificar que os três SUPIs são os mesmos configurados conforme 3.11.

Figura 15 – Interface de usuário *Web* com três usuários cadastrados



Fonte: O Autor

Posteriormente se executou os comandos de 4.1 para verificar que os três UEs possuem acesso a *internet* através do 5GC simulado.

Código Fonte 4.1 – Execução do comando *ping* através dos três túneis de usuário

```
1 $ ping google.com -I uesimtun0
2 $ ping google.com -I uesimtun1
3 $ ping google.com -I uesimtun2
```

Fonte: O Autor

4.2 Função de classificação com abordagem 1

Uma vez que vários são os critérios utilizados para a classificação dos pacotes, como tamanho do *payload*, protocolo da camada de transporte, portas de origem e portas de

destino, a estratégia adotada para os testes é isolar a influência de cada um destes critérios e depois usar os valores que resultam em classificações mais próximas das ideais como fixos nos testes posteriores. Isto é feito gradualmente utilizando as funções das três abordagens desenvolvidas.

A versão 1 do algoritmo de classificação possui apenas um parâmetro configurável: a carga mínima *min_load*. Este parâmetro é relevante para definir quais pacotes que implementam TCP na camada de transporte serão classificados como mMTC (caso sejam menores ou iguais a *min_load*) ou eMBB (caso sejam maiores que *min_load*).Desta forma, a escolha deste parâmetro é crucial para os resultados encontrados pela função em *python*.

Para escolher os valores de *min_load* primeiramente se utilizou o comando *ifconfig* para obter o valor referente ao máxima unidade de transmissão (MTU - *Maximum Trasmission Unit*) dos túneis de usuário *uesimtun0*, *uesimtun1* e *uesimtun2*. O MTU é o valor máximo de dados que um pacote da camada 2 pode carregar sem que sofra fragmentação [6] e é utilizado como base para entender qual a porcentagem da área de dados útil que um pacote pode ocupar para ser considerado mMTC. O valor encontrado para o MTU dos três túneis foi de 1400 *bytes*.

Em posse desta informação, três testes foram realizados: *min_load*=5% do mtu (70 *bytes*), *min_load*=1% do mtu (14 *bytes*), *min_load*=0,5% do mtu (7 *bytes*). Ao final dos testes com estes três valores de *min_load*, procura-se o que melhor classifica os pacotes do tipo mMTC, ou seja, o valor que mais se aproxima do resultado ideal de 19 pacotes.

4.2.1 Teste com *min_load*=5%mtu

O primeiro teste realizado utilizou um valor de 70 *bytes* como carga útil mínima e obteve como resultado 615 pacotes classificados como eMBB, 671 pacotes classificados como mMTC e 2950 como URLLC. Isto implica $14,5184\%$, $15,8404\%$ e $73,7684\%$ dos pacotes classificados como eMBB, mMTC e URLLC, respectivamente.

Este resultado, que se mostra distante do ideal, é justificado por algumas particularidades dos dados utilizados, entre eles o baixo *payload* dos pacotes URLLC contidos no arquivo de tráfego do drone e a presença de pacotes TCP neste mesmo arquivo. Devido a lógica empregada pela função da versão 1, os pacotes com estas características ao invés de serem classificados como URLLC, foram considerados mMTC e eMBB, respectivamente.

Com esta análise, decidiu-se diminuir o valor de *min_load* para minimizar o problema dos baixos *payloads* do arquivo de tráfego do drone e, desta forma, tornar mais restritiva a classificação para a grade mMTC.

4.2.2 Teste com *min_load*=1%mtu

O teste seguinte do algoritmo da versão 1 utilizou o valor de 14 *bytes* como *min_load* e o total dos dados foi classificado como: 649 eMBB, 637 mMTC e 2950 URLLC, respec-

tivamente. As porcentagens de cada grade para a amostra total foram de $15,3211\%$ para eMBB, $15,0378\%$ para mMTC e $73,7684\%$ para URLLC.

Como esperado houve uma queda no número total e, consequentemente, na porcentagem total, de pacotes classificados como mMTC para este teste. Como reflexo natural, os pacotes eMBB aumentaram e os URLLC se mantiveram estáveis, uma vez que a implementação desta função diferencia eMBB e mMTC somente pelo tamanho da área de dados útil.

Embora o valor de *min_load* tenha sido cinco vezes menor em relação ao teste anterior, é possível ver como os resultados obtidos não refletem linearmente esta diminuição, diminuído em relação à porcentagem ideal. No entanto, estes testes mostram a tendência de melhora na classificação quanto menor o valor de *min_load*.

Desta forma decidiu-se, para os próximos testes, reduzir ainda mais este valor.

4.2.3 Teste com *min_load*=0.5%mtu

Para o terceiro teste, o valor de mínima carga útil para diferenciar pacotes eMBB de mMTC, ou *min_load*, foi ajustado para 7 *bytes* e o valor encontrado como classificação final foi: 1057 eMBB, 229 mMTC e 2950 URLLC. Isto significa porcentagens de $24,9528\%$ para eMBB, $5,4060\%$ para mMTC e $73,7684\%$ para URLLC nos pacotes classificados.

Mostrando mais uma vez a não-linearidade do problema, o valor de *min_load* foi reduzido pela metade enquanto a porcentagem de pacotes classificados como mMTC foi reduzida para aproximadamente um terço em relação aos testes anteriores. Embora o valor encontrado de $5,4060\%$ ainda seja mais de dez vezes maior que a ideal para mMTC, este resultado mostra a continuidade da tendência de diminuição de *min_load*.

Outros valores foram testados para *min_load*, diminuido o valor *byte* a *byte*, no entanto os resultados não foram significativamente melhores. Em alguns casos, o *min_load* tournou-se pequeno a ponto de se tornar muito restritivo e classificar de maneira imprecisa os dados.

4.2.4 Compilação e análise dos resultados para versão 1

Os resultados dos testes realizados com a versão 1 da função se encontram na Tabela 3, nele estão presentes além dos números de pacotes classificados em grades de serviço, suas respectivas porcentagens em relação ao total. A última linha do quadro detalha os resultados considerados ideais e suas porcentagens a fim de comparação.

Embora o valor de 7 *bytes* pareça restritivo, sobretudo quando comparado ao número de *bytes* médio encontrado em [25], onde foram realizados testes com 28 aparelhos de IoT. Este valor demostra como os equipamentos, protocolos e, acima de tudo, aplicações são variadas neste tipo de comunicação. Enquanto algumas câmeras ou lâmpadas inteligentes tem maior probabilidade de possuir um pacote com *payload* de algumas centenas de

bytes, o sensor utilizado neste trabalho possui um *payload* médio de menos de 13 *bytes*, comprovando mais uma vez como a classificação de tráfego utilizando características do pacote pode sofrer variações consideráveis conforme as amostras de capturas de pacotes utilizadas.

Os resultados da Tabela 3 expõem uma relação de compensação entre as grades eMBB e mMTC utilizando a versão 1: quanto menor o valor de *min_load*, melhor esta abordagem se torna para filtrar pacotes do tipo mMTC, porém permite com que mais pacotes sejam enquadrados na grade eMBB. Além disso é possível perceber que a classificação de tráfego utilizando somente a variação do valor de *min_load* e o protocolo da camada de transporte são insuficientes para determinar a grade de serviço dos pacotes de maneira satisfatória para esta amostra, sendo necessários mais critérios.

Assim sendo, os testes continuaram com a função da versão 2, que possui mais critérios para classificação.

4.3 Função de classificação com abordagem 2

Para a versão 2 do algoritmo de classificação, quatro são os parâmetros configuráveis: o *min_load*, tal qual na versão 1, e três vetores com números de portas referentes às classes de serviço: *embb_ports*, *mmtc_ports* e *urllc_ports*. Embora várias combinações de números de portas para cada uma das classes e valores de carga mínima possam ser combinados, escolheu-se analisar o comportamento dos valores das portas de origem e destino isoladamente no resultado final da classificação, e para tal o valor de *min_load* manteve-se fixo e igual a 7 *bytes*. Esta escolha se deu pela relação já conhecida dos testes da versão 1, em que é possível comprovar que quanto menor *min_load*, melhor se classifica mMTC, porém mais tráfego é considerado como eMBB.

A fim de avaliar o impacto das portas de origem e destino na classificação final por grade de serviço de maneira isolada, três testes foram executados: um teste configurando a porta 80 (protocolo HTTP) para a grade eMBB e deixando os outros vetores vazios, um teste configurando a porta 1883 (protocolo MQTT) para a grade mMTC enquanto os outros vetores foram deixados vazios e um teste configurando a porta 2001 (prtocolo DJIUAV) para a grade URLLC da mesma forma. Um quarto e último teste avalia a

Tabela 3 – Resultados para os testes versão 1 da função de classificação.

Teste	eMBB	%eMBB	mMTC	% mMTC	URLLC	% URLLC
min_load=70bytes	615	14, 5184	671	15, 8404	2950	73, 7684
min_load=14bytes	649	15, 3211	637	15, 0378	2950	73, 7684
min_load=7bytes	1057	24, 9528	229	5, 4060	2950	73, 7684
Ideal	218	5, 1463	19	0, 4485	3999	94.4051

Fonte: O Autor

influência quando portas conhecidas são configuradas para as três grades simultaneamente.

4.3.1 Teste com porta HTTP classificada como eMBB

O primeiro teste realizado para a função de classificação versão 2, faz com que todos os pacotes cuja porta de origem ou de destino é igual a 80, seja classificado como eMBB. O resultado desta simulação encontrou 1141 pacotes classificados como eMBB, 145 como mMTC e 2950 como URLLC. As porcentagens totais da amostra resultam em $26,9594\%$ para eMBB, $3,423\%$ para mMTC e $73,7684\%$ para URLLC.

Demonstrando a continuidade da tendência de baixos valores de *min_load*, mais pacotes são enquadrados em eMBB, menos em mMTC e os URLLC mantêm-se estáveis. Uma informação interessante ao se comparar o primeiro teste da versão 1 com este é de que 96 novas amostras foram consideradas eMBB por conta da escolha da porta 80 como parâmetro para classificação. Estes resultados atestam a efetividade da escolha da porta 80 como um critério para a classficação nas grades de serviço.

4.3.2 Teste com porta MQTT classificada como mMTC

A porta 1883 é padronizada para as comunicações entre equipamentos que utilizam o protocolo MQTT [27]. O resultado do teste utilizando-a como porta para classificação da grade mMTC foi de 1051 pacotes eMBB, 235 pacotes mMTC, além dos mesmos 2950 pacotes URRLC. Resultando em $24,8111\%$ para eMBB, $5,5477\%$ para mMTC e $73,7684\%$ para URLLC.

Estes resultados se mostram muito parecidos com os obtidos no terceiro teste da versão 1, onde 6 pacotes que antes eram eMBB foram classificados na versão 2 como mMTC devido a inserção da porta 1883 no vetor *mmtc_ports*. Uma razão pela qual estes resultados se assemelham tanto é que o tráfego mMTC, para a amostra específica deste trabalho, é bem caracterizável pelo seu baixo *payload*. Ou seja, com exceção dos já mencionados 6 pacotes, as amostras que agora são classificados devido a terem a porta 1883 como origem ou destino já eram classificados anteriormente por ter uma carga útil menor ou igual a 7 *bytes*.

4.3.3 Teste com porta DJIUAV classificada como URLLC

A porta 2001, diferentemente das anteriores não é padronizada, porém é comumente utilizada pelo protocolo DJIUAV de controle de veículos aéreos não tripulados. Ao fornecer esta porta como parâmetro, se espera que o número de pacotes classificados como URLLC aumente, se aproximando dos desejados 3999.

Os resultados obtidos foram 197 pacotes classificados como eMBB, 108 classificados como mMTC e 3931 como URLLC, mais próximos aos ideais. Em termos de porcentagem

total dos dados se encontra $4,6506\%$ para eMBB, $2,5496\%$ para mMTC e $92,7998\%$ para URLLC.

Houve, devido a alteração do vetor *urllc_ports*, um ganho de quase 1000 pacotes classificados como URLLC, no entanto um reflexo desta alteração foi a diminuição drástica de pacotes eMBB, chegando inclusive a serem menores que os ideais 218.

4.3.4 Teste com portas dos protocolos classificadas simultaneamente

Por fim, um teste combinando todas as portas dos principais protocolos utilizados por cada um dos UEs foi realizado a fim de entender se o conhecimento prévio sobre as principais portas de origem e destino utilizadas pelos equipamentos da rede pode facilitar a classificação do tráfego. A classificação dos pacotes foi: 274 eMBB, 31 mMTC e 3931 URLLC, ou em porcentagem $6,4683\%$ para eMBB, $0,7318\%$ para mMTC e $92,7998\%$ para URLLC.

Os resultados apresentados utilizando as portas conhecidas se mostrarm satisfatórios, aproximando-se da classificação ideal. Os poucos pacotes que permaneceram classificados erroneamente se devem a particularidades da amostra escolhida, principalmente do tráfego do drone, onde alguns pacotes TCP estão presentes e são, a depender do tamanho do *payload* classificados como eMBB ou mMTC.

4.3.5 Compilação e análise dos resultados para versão 2

Os resultados dos testes realizados na função da versão 2 estão compilados na Tabela 4. Assim como no quadro Tabela 3, os pacotes totais e percentuais de cada classe de serviço, além dos resultados ideais estão presentes.

Tabela 4 – Resultados para os testes versão 2 da função de classificação.

Portas	eMBB	%eMBB	mMTC	% mMTC	URLLC	% URLLC
80	1141	$26,9594$	145	$3,423$	2950	$73,7684$
1883	1051	$24,8111$	235	$5,5477$	2950	$73,7684$
2001	197	$4,6506$	108	$2,5496$	3931	$92,7998$
$80, 1883, 2001$	274	$6,4683$	31	$0,7318$	3931	$92,7998$
`Ideal`	218	$5,1463$	19	$0,4485$	3999	$94,4051$

Fonte: O Autor

O que se percebe pela tabela é que a utilização da informação das portas de origem e destino, quando disponível, deve ser usada para aumentar a acurácia e eficiência do algoritmo. Percebe-se também o impacto das portas utilizadas simultaneamente para a melhora dos resultados.

Alguns poucos pacotes ainda não conseguiram ser classificados com a grade de serviço ideal e para tentar solucionar tais problemas a versão 3 utiliza-se de uma abordagem muito mais customizável.

4.4 Função de classificação com abordagem 3

A abordagem 3, ao invés de implementar uma série de estruturas do tipo *if/else* sobre as características dos cabeçalhos e área de dados dos pacotes a serem classificados, utiliza um mecanismo de vetor de pesos. Este vetor é denominado w e possui 18 parâmetros configuráveis, sendo seus índices de 0 a 17.

Este comprimento se dá pela multiplicação dos seis critérios utilizados pelas três classes de serviço às quais se deseja classificar. Os seis critérios são: a influência do protocolo TCP na classificação, a influência do protocolo UDP para a classificação, a influência da porta de origem, a influência da porta de destino, a influência do tamanho do *payload* ser menor que *min_load* e a influência do tamanho do *payload* ser maior que *min_load*.

Para os testes com a versão 3 do algoritmo de classificação, algumas considerações foram feitas baseadas nos testes das versões anteriores. Decidiu-se usar, além de um valor fixo para a carga útil mínima de 7 *bytes* conforme os testes da abordagem 1, as portas conhecidas dos equipamentos de usuário da rede conforme testes com a bordagem 2. Desta forma os vetores são dados por: *embb_ports* = [80], *mmtc_ports* = [1883] e *urllc_ports* = [2001].

Desta forma, gerou-se o vetor base para os testes com a versão 3, vetor esse que servirá como ponto de partida e de onde pequenas alterações foram realizadas gradualmente. Desta maneira é possível considerar somente o impacto dos pesos mudados pelos índices e como os mesmos alteram a classificação final. O vetor base é dado por

$$w_b = [0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0] \tag{4.1}$$

onde é possível perceber que os índices de 6 a 11, que representam a influência das portas de origem e destino para cada uma das classes de serviço, estão com o coeficiente máximo. Além disso o índice 13 que regula a influência de um baixo *payload* para a classificação do pacote em mMTC também tem o valor máximo. Estas escolhas de pesos se justificam pelos bons resultados encontrados ao utilizar as portas já citadas e pela classe mMTC ser bem classificável por seu baixo *payload*.

O resultado obtido utilizando apenas estes valores base foi de: 3223 pacotes eMBB, 24 pacotes mMTC e 989 URLLC. O que resulta em uma porcentagem total de $76,0859\%$ eMBB, $0,5668\%$ mMTC e $23,3475\%$ URLLC.

O que se percebe ao utilizar este vetor base no primeiro teste é que apenas as portas de destino e o valor *min_load*, por si sós, não são capazes de classificar o tráfego. São

necessárias as informações de protocolo da camada de transporte e tamanho do *payload* quando este é maior que a carga útil mínima.

Ao observar as porcentagens dos resultados dos pacotes, nota-se que muitos deles que deveriam ser classificados como URLLC foram classificados como eMBB. E isto se justifica pelos pacotes vindos do drone que não tem como porta de origem ou destino 2001. Estes pacotes, sobre a influência dos pesos de w_b acabam por não ter características marcantes associadas à nenhuma das três classes e sendo classificados na mais abrangente delas, o eMBB.

4.4.1 Influência do protocolo da camada de transporte

A fim de solucionar o problema de vários pacotes URLLC serem classificados como eMBB pela função da abordagem 3, o critério do protocolo da camada de transporte UDP foi considerado. Ao se utilizar o valor 1 para o índice 5 e mantendo zerados os índices 3 e 4, todos os pacotes cujo protocolo é UDP terão influência somente para classificação como URLLC.

Outro índice a ser alterado no vetor base diz respeito à características particulares dos dados utilizados. Conforme já observado no primeiro teste desta abordagem, a amostra de tráfego do drone possui alguns pacotes TCP que podem ser confundidos com a grade eMBB, desta forma o valor $0,5$ foi adicionado ao índice 2, que regula a influência do protocolo TCP para a classificação de pacotes URLLC.

Com estas alterações o novo vetor, denominado w_1 é dado por:

$$w_1 = [0, 0, 0, 5, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0] \tag{4.2}$$

e obteve um resultado de 290 pacotes da classe eMBB, 30 mMTC, e 3916 URLLC. Em porcentagens, $6,8461\%$ para eMBB, $0,7082\%$ para mMTC e $92,4221\%$ para URLLC na amostra total dos dados.

Com estas duas alterações é possível obter resultados muito mais próximos do ideal. A partir dos pesos que aumentaram a influência do coeficiente da classe URLLC, é possível como quase todos os pacotes eMBB classificados erroneamente por w_b, são corretamente classificados como URLLC por w_1. Para os poucos restantes, o tamanho da carga útil é analisada.

4.4.2 Influência do tamanho do *payload*

Embora o vetor w_1 tenha mostrado resultados próximos aos ideais em termos de números exatos e porcentagens, alguns pacotes URLLC ainda são classificados como eMBB devido ao tamanho do *payload*. A fim de minimizar este problema, o peso da influência do tamanho do *payload* para a classificação na grade URLLC, representado pelo índice 17 foi alterado para 0.5, resultando em

$$w_2 = [0, 0, 0, 5, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 5] \tag{4.3}$$

.

O resultado encontrado mostra 218 pacotes eMBB, 24 mMTC e 3994 URLLC com porcentagens de $5,1464\%$, $0,5666\%$ e $94,2634\%$ para eMBB, mMTC e URLLC, respectivamente.

Com este novo vetor que prioriza a classificação dos pacotes cuja área útil de dados é maior que *min_load* como URLLC, é possível ver que os desejados 218 pacotes eMBB foram alcançados. Há ainda 5 pacotes classificados de maneira errônea, onde a função falha em diferenciar as grades mMTC e URLLC.

Estes cinco pacotes são casos particulares na amostra de tráfego do drone, onde, pacotes de confirmação (ACK) do processo de *3-way handsahke* do protocolo TCP, cujo *payload* é zero, não possuem 2001 como porta de origem nem destino. Diante dessas características e com os valores de pesos de w_2, estes são classificados como mMTC. Outros testes tentaram aumentar o valor do índice 17, porém, com isso, pacote eMBB passaram a ser classificados como URLLC, invertendo o problema.

4.4.3 Compilação e análise dos resultados para versão 3

Os resultados dos testes realizados com a versão 3 da função em *python* se encontram na Tabela 5, nele estão presentes além dos números de pacotes classificados como cada uma das grades de serviço e suas respectivas porcentagens em relação ao total, os resultados considerados ideais.

Tabela 5 – Resultados para os testes versão 3 da função de classificação.

Vetor	eMBB	%eMBB	mMTC	% mMTC	URLLC	% URLLC
w_b	3223	$76,0859$	24	$0,5668$	989	$23,3475$
w_1	290	$6,8461$	30	$0,7082$	3916	$92,4221$
w_2	218	$5,1463$	24	$0,5666$	3994	94.2634
`Ideal`	218	$5,1463$	19	$0,4485$	3999	$94,4051$

Fonte: O Autor

Os resultados agregados no quadro 5 demonstram como a abordagem 3 e seu mecanismo de vetor de pesos pode aumentar a eficiência na classificação de amostras de dados com características particulares. Embora 5 dos 4236 pacotes totais não tenham sido classificado corretamente, este resultado se mostra satisfatório, atingindo 100% de acurácia para eMBB, $79,1564\%$ para mMTC e $99,845\%$ para URLLC no último teste com a versão 3.

Vale ressaltar que os pesos contidos nos índices de w_2 não podem ser generalizados e servem somente para esta amostra de dados específica. Para outros dados de tráfego

outras combinações de valores que reflitam os dados devem ser usados. Esta era a proposta quando este mecanismo de vetor de pesos foi desenvolvida, uma maneira mais customizada de gerar a classificação a depender da amostra disponível e suas particularidades.

Ao final desta abordagem é possível ver com apenas 5 dos 4236 pacotes foram classificados incorretamente, obtendo um resultado de $99,8820\%$ de acurácia na classificação total. A Tabela 6 sumariza os melhores resultados obtidos com cada abordagem, além de apresentar a diferença percentual entre estes resultados e o resultado ideal.

Tabela 6 – Compilação dos melhores resultados obtidos com cada abordagem.

Abordagem	eMBB	%eMBB	mMTC	% mMTC	URLLC	% URLLC
1	1057	$24,9528$	229	$5,40604$	2950	$73,7684$
2	274	$6,4683$	31	$0,7318$	3931	$92,7998$
3	218	$5,1463$	24	$0,5666$	3994	$94,2634$
`Ideal`	218	$5,1463$	19	$0,4485$	3999	$94,4051$

Fonte: O Autor

5 Conclusões

Este trabalho resultou em um relevante estudo sobre a classificação de tráfego em um contexto de redes 5G, especialmente para casos onde o *core* precisa atender diferentes serviços através de classes de serviço. Além disso foi possível observar como os sistemas 5G são capazes de operar e se comportar em ambientes simulados, compreendendo como esta tecnologia pode impactar as soluções de rede em um futuro a curto e médio prazos. O simulador *free5gc* com a sua implementação em *conteiners* se mostrou muito versátil e adaptável, sendo possível simular a topologia desejada através de configurações e comandos simples. Já o simulador UERANSIM foi capaz de estabelecer a conexão dos equipamentos simulados também de maneira intuitiva e prática.

As funções de envio, captura e classificação também se mostraram funcionais e obtiveram resultados relevantes ao apontar qual a classe de serviço mais apropriada para cada pacote. Embora a amostra de dados escolhida possua suas particularidades e exceções, grande parte dos pacotes foi classificado de maneira correta após a definição de valores coerentes.

A linguagem *python* também se mostrou muito intuitiva e simples de ser usada, apresentando uma sintaxe direta e visual. No entanto, por ser uma linguagem de alto nível, muitas vezes exigiu alto processamento e, somada a algumas limitações do hardware hospedeito, apresentou lentidão na classificação dos pacotes. Estas limitações devem ser consideradas, principalmente em situações de criticidade, como nas aplicações da grade URLLC.

Os resultados obtidos, embora não possam ser generalizados para todas as aplicações devido a complexidade, se mostraram satisfatórios e importantes, obtendo bons percentuais de acurácia. Destaca-se este trabalho como material de estudo para interessados em encaminhamento de pacotes em enlaces dentro do sistema 5G e para possíveis categorizações de tráfego para posterior aplicação de QoS.

Com relação aos objetivos propostos, este trabalho os cumpriu de maneira satisfatória. Foi possível compreender de maneira mais detalhada o funcionamento e aplicação de redes 5G, além de entender os benefícios e limitações da classificação do tráfego nestas redes. Com as simulações realizadas o autor conseguiu evidenciar a influência de diferentes aspectos para o correto funcionamento da solução proposta, sendo possível explicar os resultados obtidos e suas especificidades.

5.1 Pesquisas futuras

Para pesquisas futuras, é sugerido utilizar as soluções propostas neste trabalho em um ambiente 5G real, contendo elementos de *hardware* e *software* mais próximos dos utilizados na prática. Outra sugestão é o aprimoramento das funções de classificação, tornando-as adequadas a classificação também em aplicações críticas e em tempo real.

Para as funções de classificação descritas, mais testes com diferentes bases de dados podem ser realizados. Podendo assim, validar as três soluções para diferentes cenários de maneira mais assertiva.

Uma outra pesquisa bastante relevante é a integração das funções de classificação, sobretudo da abordagem 3, com técnicas de aprendizagem de máquina supervisionada. Em um cenário ideal, amostras de tráfego treinariam a rede para que os pesos dos índices do vetor w sejam determinados automaticamente e de maneira otimizada tendo em vista todos os desafios exigidos neste contexto. O emprego destas técnicas, em conjunto com os paradigmas de virtualização, se mostra muito promissor para a amplificação e popularização do 5G.

Referências Bibliográficas

1 PENTTINEN, J. T. J. *5G explained: Security and deployment of advanced mobile communications*. [S.l.]: John Wiley Sons Ltd, 2019. ISBN 9781119275688. 18, 23, 24, 27, 32, 35, 36, 38, 40

2 ROMMER, S. et al. *5G core networks: Power digitalization*. London: Academic Press, 2020. 502 p. ISBN 9780081030097. 18, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39

3 3GPP. *3GPP TR 21.915 version 15.0.0 Release 15*. Disponível em: <https://www.3gpp.org/release-15>. Acesso em: 31 jul. 2022. 18

4 FACULDADE DE ENGENHARIADA UNIVERSIDADE DO PORTO. *TCP/IP*. Disponível em: <https://paginas.fe.up.pt/~mrs01003/TCP_IP.htm>. Acesso em: 12 jul. 2022. 21, 22, 23

5 TANENBAUM, A. S.; WETHERALL, D. *Computer networks, 5th Edition*. [S.l.]: Pearson, 2011. ISBN 0132553171. 21, 22

6 KUROSE, J. F.; ROSS, K. W. *Computer Networking: A Top-Down Approach*. 7. ed. Boston, MA: Pearson, 2016. ISBN 978-0-13-359414-0. 21, 22, 23, 68

7 BOTH, C. B. et al. *Soft5G+: explorando a softwarização nas redes 5G*. [S.l.]: XXXVIII Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos, 2020. 24, 25, 26, 28

8 SANTOS, R. D. L. *1.2 – Evolução histórica*. Disponível em: <https://www.gta.ufrj.br/ensino/eel879/trabalhos_vf_2008_2/ricardo/1_2.html>. Acesso em: 31 jul. 2022. 24

9 TELECO. *Tecnologia 3G*. 2017. Disponível em: <https://www.teleco.com.br/3g_tecnologia.asp>. Acesso em: 12 jul. 2022. 25

10 TELECO. *Tecnologia 4G*. 2017. Disponível em: <https://www.teleco.com.br/4g_tecnologia.asp>. Acesso em: 12 jul. 2022. 25

11 CHANDRAMOULI, D.; LIEBHART, R.; PIRSKANEN, J. *5G for the connected world*. [S.l.]: John Wiley Sons Ltd, 2019. 32, 34, 37, 38

12 PUJOLLE, G. *Software networks: Virtualization, SDN, 5G and security*. 2. ed. [S.l.]: John Wiley Sons Ltd, 2020. ISBN 9781119694724. 39, 40

13 ORACLE. *Download VirtualBox*. Disponível em: <https://www.oracle.com/virtualization/virtualbox/>. Acesso em: 23 mar. 2022. 41

14 NATIONAL CHIAO TUNG UNIVERSITY AND NATIONAL CHUNG CHENG UNIVERSITY. *free5GC Link The Word!* 2019. Disponível em: <https://www.free5gc.org/>. Acesso em: 14 abr. 2022. 42

15 NATIONAL CHIAO TUNG UNIVERSITY. *Free5GC Compose*. Disponível em: <https://github.com/free5gc/free5gc-compose>. Acesso em: 26 jun. 2022. 42, 48

16 3GPP. *General Packet Radio System (GPRS) Tunneling Protocol User Plane (GTPv1-U)*. Disponível em: <https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=1699>. Acesso em: 1 jul. 2022. 42

17 3GPP. *Interface between the Control Plane and the User Plane nodes*. Disponível em: <https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3111>. Acesso em: 30 mai. 2022. 42

18 MONGODB INC. *MongoDB*. Disponível em: <https://www.mongodb.com/>. Acesso em: 12 jul. 2022. 42

19 GüNGöR, A. *UERANSIM 5G SOLUTIONS*. Disponível em: <https://github.com/aligungr/UERANSIM>. Acesso em: 12 jul. 2022. 43, 52

20 BIONDI, P. *Scapy: Packet Crafting for Python2 and Python3*. 2021. Disponível em: <https://scapy.net>. Acesso em: 12 jul. 2022. 43

21 BOAVA, A.; MAILER, C.; KRAUS, D. *Arquitetura de Serviços e Computação de Borda nas redes 5G para o desenvolvimento de redes virtuais privadas sobre o CORE 5G*. 2021. 46, 48

22 KRAUS, D. *Computação de borda para indústria utilizando a rede 5G*. Monografia — Unversidade Federal de Santa Catarina, 2021. 46

23 CHANG, Y. W. *Linux kernel module 5G GTP-U*. Disponível em: <https://github.com/PrinzOwO/gtp5g>. Acesso em: 12 jul. 2022. 48

24 WIRESHARK FOUNDATION. *Wireshark Sample Captures*. Disponível em: <https://gitlab.com/wireshark/wireshark/-/wikis/SampleCaptures>. Acesso em: 12 jul. 2022. 56

25 SIVANATHAN, A. et al. Classifying iot devices in smart environments using network traffic characteristics. *IEEE Transactions on Mobile Computing*, v. 18, n. 8, p. 1745–1759, 2019. 59, 69

26 BULASHENKO, A. et al. New traffic model of m2m technology in 5g wireless sensor networks. In: *2020 IEEE 2nd International Conference on Advanced Trends in Information Theory (ATIT)*. [S.l.: s.n.], 2020. p. 125–131. 59

27 IANA. *Service Name and Transport Protocol Port Number Registry*. Disponível em: <https://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.xhtml>. Acesso em: 12 jul. 2022. 61, 71