UNIVERSIDADE FEDERAL DE GOIÁS
INSTITUTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

RÚBEN FRANÇA XAVIER

Integração do modelo de referência Multi-Access Edge Computing (MEC) com o Núcleo 5G

GOIÂNIA
2022

UNIVERSIDADE FEDERAL DE GOIÁS
INSTITUTO DE INFORMÁTICA

TERMO DE CIÊNCIA E DE AUTORIZAÇÃO (TECA) PARA DISPONIBILIZAR VERSÕES ELETRÔNICAS DE TESES

E DISSERTAÇÕES NA BIBLIOTECA DIGITAL DA UFG

Na qualidade de titular dos direitos de autor, autorizo a Universidade Federal de Goiás (UFG) a disponibilizar, gratuitamente, por meio da Biblioteca Digital de Teses e Dissertações (BDTD/UFG), regulamentada pela Resolução CEPEC nº 832/2007, sem ressarcimento dos direitos autorais, de acordo com a Lei 9.610/98, o documento conforme permissões assinaladas abaixo, para fins de leitura, impressão e/ou download, a título de divulgação da produção científica brasileira, a partir desta data.

O conteúdo das Teses e Dissertações disponibilizado na BDTD/UFG é de responsabilidade exclusiva do autor. Ao encaminhar o produto final, o autor(a) e o(a) orientador(a) firmam o compromisso de que o trabalho não contém nenhuma violação de quaisquer direitos autorais ou outro direito de terceiros.

1. Identificação do material bibliográfico

[x] Dissertação [ ] Tese [ ] Outro*:_____

*No caso de mestrado/doutorado profissional, indique o formato do Trabalho de Conclusão de Curso, permitido no documento de área, correspondente ao programa de pós-graduação, orientado pela legislação vigente da CAPES.

Exemplos: Estudo de caso ou Revisão sistemática ou outros formatos.

2. Nome completo do autor

Rúben França Xavier

3. Título do trabalho

Integração do modelo de referência Multi-access Edge Computing (MEC) com o Núcleo 5G

4. Informações de acesso ao documento (este campo deve ser preenchido pelo orientador)

Concorda com a liberação total do documento [ ] SIM [x] NÃO[1]

[1] Neste caso o documento será embargado por até um ano a partir da data de defesa. Após esse período, a possível disponibilização ocorrerá apenas mediante:
a) consulta ao(à) autor(a) e ao(à) orientador(a);
b) novo Termo de Ciência e de Autorização (TECA) assinado e inserido no arquivo da tese ou dissertação.
O documento não será disponibilizado durante o período de embargo.
Casos de embargo:
- Solicitação de registro de patente;
- Submissão de artigo em revista científica;
- Publicação como capítulo de livro;
- Publicação da dissertação/tese em livro.

Obs. Este termo deverá ser assinado no SEI pelo orientador e pelo autor.

Documento assinado eletronicamente por **Antonio Carlos De Oliveira Junior**, **Professor do Magistério Superior**, em 21/11/2022, às 12:48, conforme horário oficial de Brasília, com fundamento

no § 3º do art. 4º do Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **RUBEN FRANÇA XAVIER**, **Discente**, em 21/11/2022, às 14:06, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do Decreto nº 10.543, de 13 de novembro de 2020.

A autenticidade deste documento pode ser conferida no site https://sei.ufg.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0, informando o código verificador **3348060** e o código CRC **5CD0B7D8**.

Referência: Processo nº 23070.051400/2022-79 SEI nº 3348060

UNIVERSIDADE FEDERAL DE GOIÁS
INSTITUTO DE INFORMÁTICA

TERMO DE CIÊNCIA E DE AUTORIZAÇÃO (TECA) PARA DISPONIBILIZAR VERSÕES ELETRÔNICAS DE TESES E DISSERTAÇÕES NA BIBLIOTECA DIGITAL DA UFG

Na qualidade de titular dos direitos de autor, autorizo a Universidade Federal de Goiás (UFG) a disponibilizar, gratuitamente, por meio da Biblioteca Digital de Teses e Dissertações (BDTD/UFG), regulamentada pela Resolução CEPEC nº 832/2007, sem ressarcimento dos direitos autorais, de acordo com a Lei 9.610/98, o documento conforme permissões assinaladas abaixo, para fins de leitura, impressão e/ou download, a título de divulgação da produção científica brasileira, a partir desta data.

O conteúdo das Teses e Dissertações disponibilizado na BDTD/UFG é de responsabilidade exclusiva do autor. Ao encaminhar o produto final, o autor(a) e o(a) orientador(a) firmam o compromisso de que o trabalho não contém nenhuma violação de quaisquer direitos autorais ou outro direito de terceiros.

1. Identificação do material bibliográfico

[X] Dissertação [ ] Tese [ ] Outro*:_____

*No caso de mestrado/doutorado profissional, indique o formato do Trabalho de Conclusão de Curso, permitido no documento de área, correspondente ao programa de pós-graduação, orientado pela legislação vigente da CAPES.

Exemplos: Estudo de caso ou Revisão sistemática ou outros formatos.

2. Nome completo do autor

Rúben França Xavier

3. Título do trabalho

Integração do modelo de referência Multi-access Edge Computing (MEC) com o Núcleo 5G

4. Informações de acesso ao documento (este campo deve ser preenchido pelo orientador)

Concorda com a liberação total do documento [X] SIM [ ] NÃO[1]

[1] Neste caso o documento será embargado por até um ano a partir da data de defesa. Após esse período, a possível disponibilização ocorrerá apenas mediante:
a) consulta ao(à) autor(a) e ao(à) orientador(a);
b) novo Termo de Ciência e de Autorização (TECA) assinado e inserido no arquivo da tese ou dissertação.
O documento não será disponibilizado durante o período de embargo.
Casos de embargo:
- Solicitação de registro de patente;
- Submissão de artigo em revista científica;
- Publicação como capítulo de livro;
- Publicação da dissertação/tese em livro.

Obs. Este termo deverá ser assinado no SEI pelo orientador e pelo autor.

Documento assinado eletronicamente por **Rúben França Xavier**, **Discente**, em 05/11/2024, às 13:45, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **Antonio Carlos De Oliveira Junior**, **Professor do Magistério Superior**, em 05/11/2024, às 14:59, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do Decreto nº 10.543, de 13 de novembro de 2020.

A autenticidade deste documento pode ser conferida no site https://sei.ufg.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0, informando o código verificador **4946976** e o código CRC **A73308D9**.

Referência: Processo nº 23070.051400/2022-79 SEI nº 4946976

RÚBEN FRANÇA XAVIER

Integração do modelo de referência Multi-Access Edge Computing (MEC) com o Núcleo 5G

Dissertação apresentada ao Programa de Pós–Graduação em Ciência da Computação do Instituto de Informática da Universidade Federal de Goiás, como requisito para obtenção do título de Mestre em Ciência da Computação.

Área de concentração: Ciência da Computação.

Linha de Pesquisa: Sistemas de Computação.

Orientador: Professor Doutor Antonio Carlos de Oliveira Junior

Co-Orientador: Professor Doutor Leandro Alexandre Freitas

GOIÂNIA
2022

Ficha de identificação da obra elaborada pelo autor, através do
Programa de Geração Automática do Sistema de Bibliotecas da UFG.

Xavier, Rúben França
Integração do modelo de referência Multi-Access Edge Computing (MEC) com o Núcleo 5G [manuscrito] / Rúben França Xavier. - 2022.
LVI, 56 f.

Orientador: Prof. Dr. Antonio Carlos de Oliveira Junior; co orientador Dr. Leandro Alexandre Freitas.
Dissertação (Mestrado) - Universidade Federal de Goiás, Instituto de Informática (INF), Programa de Pós-Graduação em Ciência da Computação, Goiânia, 2022.
Bibliografia.
Inclui gráfico, tabelas, lista de figuras, lista de tabelas.

1. Integração. 2. MEC. 3. 5G. 4. 5GC. 5. Serviço. I. Junior, Antonio Carlos de Oliveira, orient. II. Título.

CDU 004

UNIVERSIDADE FEDERAL DE GOIÁS

INSTITUTO DE INFORMÁTICA

ATA DE DEFESA DE DISSERTAÇÃO

Ata nº **21** da sessão de Defesa de Dissertação de **Rúben França Xavier,** que confere o título de Mestre em Ciência da Computação, na área de concentração em Ciência da Computação.

Aos vinte e um dias do mês de outubro de dois mil e vinte e dois, a partir das catorze horas, na sala 150 do prédio do INF, realizou-se a sessão pública de Defesa de Dissertação **"Integração do modelo de referência Multi-access Edge Computing (MEC) com o Núcleo 5G"**. Os trabalhos foram instalados pelo Orientador, Professor Doutor Antonio Carlos de Oliveira Júnior (INF/UFG) com a participação dos demais membros da Banca Examinadora: Professor Doutor Leandro Alexandre Freitas (IFG/Inhumas), coorientador; Professor Doutor Augusto José Venâncio Neto (DIMAp/UFRN), membro titular externo; e Professor Doutor Fabricio Lira Figueiredo (CPqD, membro titular externo. A realização da banca ocorreu por meio de videoconferência. Durante a arguição os membros da banca não fizeram sugestão de alteração do título do trabalho. A Banca Examinadora reuniu-se em sessão secreta a fim de concluir o julgamento da Dissertação, tendo sido o candidato **aprovado** pelos seus membros. Proclamados os resultados pelo Professor Doutor Antonio Carlos de Oliveira Júnior, Presidente da Banca Examinadora, foram encerrados os trabalhos e, para constar, lavrou-se a presente ata que é assinada pelos Membros da Banca Examinadora, aos vinte e um dias do mês de outubro de dois mil e vinte e dois.

TÍTULO SUGERIDO PELA BANCA

Documento assinado eletronicamente por **Fabricio Lira Figueiredo**, **Usuário Externo**, em 21/10/2022, às 17:02, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **Leandro Alexandre Freitas**, **Usuário Externo**, em 21/10/2022, às 17:02, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **AUGUSTO JOSÉ VENÂNCIO NETO**, **Usuário Externo**, em 21/10/2022, às 17:03, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **Antonio Carlos De Oliveira Junior**, **Professor do Magistério Superior**, em 21/10/2022, às 17:03, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **RUBEN FRANÇA XAVIER**, **Discente**, em 22/10/2022, às 06:52, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do Decreto nº 10.543, de 13 de novembro de 2020.

A autenticidade deste documento pode ser conferida no site https://sei.ufg.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0, informando o código verificador **3207622** e o código CRC **770DD165**.

Referência: Processo nº 23070.051400/2022-79 SEI nº 3207622

Resumo

Xavier, Rúben França. **Integração do modelo de referência Multi-Access Edge Computing (MEC) com o Núcleo 5G**. GOIÂNIA, 2022. 58p. Dissertação de Mestrado. Programa de pós-graduação em Ciência da Computação, Instituto de Informática, Universidade Federal de Goiás.

Multi-access Edge Computing (MEC) é o conceito-chave para que possam ser desenvolvidas novas aplicações e serviços que levam benefícios da computação de borda para as redes e usuários. Com as aplicações e serviços na borda, ou seja, mais próximos dos usuários e dispositivos, será possível utilizar de recursos como ultra baixa latência, ampla largura de banda e consumo de recursos. Contudo, para que isso seja possível é necessário que seja feito integração entre framework MEC e core 5G. Para isso, este trabalho propõe a especificação de um serviço que estenderá a API *Multi-Access Traffic Steering* (MTS) como forma de ponte para ligação entre MEC e 5G Core. Foi usado o free5GC e um cluster Kubernetes para simular um ambiente fim a fim. O serviço proposto foi validado utilizando o ferramental supracitado, considerando cenários com grande volume de usuários. A validação demonstrou que o serviço resolve o problema apresentado, além de demonstrar a sua viabilidade em casos de uso.

Palavras–chave

Edge Computing, MEC, 5G, 5G Core, integração, API

Abstract

Xavier, Rúben França. **Integration of the Multi-Access Edge Computing (MEC) reference model with the 5G Core**. GOIÂNIA, 2022. 58p. MSc. Dissertation. Programa de pós-graduação em Ciência da Computação, Instituto de Informática, Universidade Federal de Goiás.

Multi-access Edge Computing (MEC) is the key concept for developing new applications and services that bring the benefits of edge computing to networks and users. With applications and services at the edge, that is, closer to users and devices, it will be possible to use features such as ultra low latency, high bandwidth and resource consumption. However, for this to be possible, it is necessary to integrate the MEC framework and the 5G core. For this, this work proposes the specification of a service that will extend the *Multi-Access Traffic Steering* (MTS) API as a bridge for the connection between MEC and 5G Core. Free5GC and a Kubernetes cluster were used to simulate an end-to-end environment. The proposed service was validated using the aforementioned tool, with end-to-end scenarios and also in scenarios with a large volume of users. The validation demonstrated that the service solves the presented problem, in addition to demonstrating its viability in use cases.

Keywords

Edge computing, 5G, 5G Core, MEC, API, integration

Sumário

Lista de Figuras **10**

Lista de Tabelas **11**

1 Introdução **12**
- 1.1 Introdução 12
- 1.2 Motivação 13
- 1.3 Justificativa 14
- 1.4 Objetivos 15
  - 1.4.1 Objetivo Geral 15
  - 1.4.2 Objetivos Específicos 15
- 1.5 Metodologia 16
- 1.6 Contribuições 16
- 1.7 Publicações 17
- 1.8 Organização do trabalho 17

2 Conceitos e Trabalhos Relacionados **18**
- 2.1 Conceitos Fundamentais 18
  - 2.1.1 Sistema 5G (5GS) 18
  - 2.1.2 5G Core (5GC) 19
    - Funções de Rede 19
    - Fatiamento de Rede 21
    - Interfaces de Rede 21
  - 2.1.3 Arquitetura de Referência MEC 21
- 2.2 Trabalhos Relacionados 22
  - 2.2.1 A new Approach to 5G and MEC Integration 23
  - 2.2.2 Toward slicing-enabled multi-access edge computing in 5G 23
  - 2.2.3 MEC in 5G Networks 24
  - 2.2.4 On O-RAN, MEC, SON and Network Slicing integration 25
  - 2.2.5 Comparação de Trabalhos Relacionados 26

3 Proposta de um Serviço para integração MEC e Núcleo 5G **29**

4 Implementação da API proposta **34**
- 4.1 Funcionamento 38
  - 4.1.1 MTS API 38
  - 4.1.2 Service Registry 39

5 Avaliação e Resultados **41**
5.1 Casos de uso candidatos 41
5.2 Descrição do ambiente 41
5.3 Métricas de avaliação 45
5.3.1 Validação da API 45
5.3.2 Validação Fim a Fim 46
5.4 Comunicação Fim a Fim 47
5.5 Experimentação da API proposta 48

6 Conclusão **52**

Referências Bibliográficas **54**

Lista de Figuras

2.1 Arquitetura do 5GC. Baseada em [46] 20
2.2 Arquitetura de referência MEC. Retirada de [15] 22
2.3 Arquitetura DASMO habilitada para integração MEC e sistema 5G. Retirada de [49] 23
2.4 Arquitetura de gerenciamento e orquestração para *Network Slice*. Baseada em [26] 24
2.5 Arquitetura de integração entre MEC e 5GS. Baseada em [25] 25
2.6 Visão geral de O-RAN integrado, 5GC-CP e MEC para dois domínios RIC I-near-RT. Retirado de [29]. 26

3.1 Descrição dos serviços das APIs de gerenciamento de tráfego. Retirado de [17] 30
3.2 Arquitetura geral da Proposta. 30
3.3 Diagrama apresentando métodos que serão implementados. 32

4.1 Funcionamento base da API proposta 39
4.2 Registro de uma aplicação no *Service Registry* 39

5.1 Estrutura de VMs utilizadas na validação. 43
5.2 Arquitetura do *core* free5GC. 44
5.3 Arquitetura geral de testes. 46
5.4 Troca de Mensagens capturadas usando WireShark na interface do M-UPF 47
5.5 Desempenho com 1 Usuário. 49
5.6 Desempenho com 10 Usuários. 50
5.7 Desempenho com 50 Usuários. 50
5.8 Desempenho com 100 Usuários. 51

Lista de Tabelas

2.1 Comparação da proposta deste trabalho com trabalhos relacionados. 27

5.1 KPIs utilizandos para comparação. 42

CAPÍTULO 1

Introdução

1.1 Introdução

Nos últimos anos a produção de dados ao redor do mundo cresceu significativamente em função do aumento considerável do número de conexões [48]. Tecnologias já apresentadas anteriormente, como a computação em nuvem, já não são mais suficientes para atender os requisitos de *Quality of Experience* (QoE) de aplicações que têm alta produção de dados e também necessitam de baixa latência [23]. Logo, micro *data centers* [21, 9], *cloudlets* [42] e *fog computing* [7] são apresentadas como possíveis soluções para determinados casos. O *Mobile Cloud Computing* (MCC) foi o primeiro paradigma da computação que possibilitou o desenvolvimento de aplicações mais complexas, a partir da combinação de *cloud computing*, *mobile computing* e redes sem fio [11].

As redes de dados para telefonia celular evoluiu consideravelmente desde a criação da 1º Geração (1G) na década de 80. O 1G, que para época era uma tecnologia avançada, permitia apenas a transmissão de voz, porém com o aumento do número de usuários sofreu com problemas de escalabilidade, qualidade de voz por exemplo [37, 24]. Na Terceira Geração (3G) a rede passa a ter capacidade de comunicação de dados para os dispositivos móveis, e além disso trouxe uma maior qualidade de voz e segurança [33, 24]. A evolução para a quarta geração (4G) traz consigo diferenças como a implementação total em ambiente IP, permitindo, além da transmissão de voz, a possibilidade do acesso a dados e arquivos multimídia [43]. Estas evoluções na rede permitiram que as aplicações também evoluíssem e assim se espera que o número de dispositivos que requerem ampla largura de banda cresça com o desenvolvimento, por exemplo, da internet das coisas (*Internet of Things* - IoT) [24].

A evolução das aplicações ao longo dos anos fizeram com que estas se tornassem exigentes quanto aos seus requisitos de rede. As redes de Quinta Geração (5G) surgem na expectativa de que possam atender estas aplicações [41]. Em comparação com as anteriores, o 5G, padronizado pela *3rd Generation Partnership Project* (3GPP), traz como uma de suas principais evoluções, apresentada na *Release* 15, uma nova arquitetura baseada em serviços e microsserviços chamada *Service-Based Architecture* (SBA) [2].

Esta arquitetura traz como característica principal a separação do plano de dados (*Data Plane* - DP) e plano de controle (*Control Plane* - CP).

Entretanto, a introdução das redes 5G trouxe consigo uma nova geração de serviços e aplicações que estão associadas a três serviços propostos na rede: *Enhanced Mobile Broadband* (eMBB), *Ultra Reliable Low Latency Communications* (URLLC) e *Massive Machine Type Communications* (mMTC) [6, 41]. O eMBB trata-se da maximização da taxa de transferência dos dados de forma que aplicações, como por exemplo, o *streaming* de vídeo e realidade aumentada e virtual se beneficiem de tal [6]. O URRLC engloba serviços que são altamente sensíveis a latência e necessitam ainda que haja um alto nível de confiabilidade na rede, por exemplo automação industrial, carros autônomos e cirurgias remotas [6]. O mMTC envolve um grande número de dispositivos (e.g. 1 milhão de dispositivos em um quilômetro quadrado) conectados na rede, com diferentes requisitos, como dispositivos de monitoramento, detecção e contagem.

Estas aplicações e serviços geram uma grande quantidade de dados, e por isso há a necessidade que a rede consiga manipular estes dados, de forma que estejam disponíveis para serem utilizados por essas aplicações, serviços e seus provedores. Logo, necessita-se que a rede entregue recursos como ultra baixa latência, ampla largura de banda e consumo de recursos [18]. Em casos deste tipo, a rede de dados não possui estrutura suficiente a atender os requisitos destas aplicações e serviços, logo, uma abordagem onde o processamento é feito na borda, ou seja, mais próximo do usuário como proposta de resolução deste problema, e é neste cenário que surge o *Multi-Access Edge Computing* (MEC) [45]. *Multi-Access Edge Computing*, antes conhecido como *Mobile Edge Computing*, é um conceito no qual os serviços são levados até a borda da rede, ou seja, mais próximo de onde os dados são produzidos [3]. A proposta do MEC é se tornar uma solução complementar junto a um núcleo de rede 5G, para que seja possível alcançar os requisitos de aplicações e serviços aderentes ao URLLC [46].

A *European Telecommunications Standards Institute* (ETSI) é a entidade responsável pelo desenvolvimento da padronização do MEC. Esta padronização acontece sob um *Institute Specification Group* (ISG) com o objetivo oferecer a aplicativos um ambiente que permita integração eficiente e contínua para aplicações em plataformas de múltiplos fornecedores [14].

1.2 Motivação

O acesso a rede de dados móveis está em crescente em todo o mundo. No Brasil, por exemplo, segundo a Agência Nacional de Telecomunicações (ANATEL), em fevereiro de 2021, foram 211,8 milhões de acessos à banda larga móvel, número que no mês seguinte saltou para 213,7 milhões [4]. Isso representa um salto na densidade de acessos

de 98,5 acessos/100hab. para 99,4 acessos/100hab [4]. Uma previsão feita pela Ericsson diz que em 2026 54% de todo o tráfego de dados móveis seja por meio de redes 5G [13]. Esta crescente obriga as operadoras a buscar melhorias para o usuário e também para sua infraestrutura de forma a atender a demanda existente.

Para isso, a rede 5G traz consigo evoluções em relação a gerações anteriores. Além de uma nova arquitetura, o 3GPP apresenta na *Release* 16 a expansão do sistema 5G (5G System - 5GS), como por exemplo serviços prioritários de multimídia, acesso 5G através de satélites, possibilidade de conexão sem fio e com fio ao 5G, automação de rede, entre outros [1]. A nova arquitetura do 5G é chamada de *Service Based Architecture* (SBA). A SBA é um modelo de desenvolvimento constituído por NFs que produzem e expõe serviços através de uma interface chamada *Service Based Interface* (SBI). Mesmo com este novo modelo, ainda são necessárias tecnologias que auxiliem o 5G à alcançar patamares mais avançados. É nesse ponto que se baseia a introdução do MEC.

As principais motivações desta dissertação estão amparadas nas vantagens que a utilização dos conceitos de MEC trazem para a rede. Minimização de congestionamento de tráfego, diminuição dos custos de transmissão de dados, redução da latência e da carga computacional são exemplos dessas vantagens [18] [22]. Além disto, no fato de que o progresso, no que tange à integração entre redes 5G e *frameworks* MEC, não só beneficiam a comunidade acadêmica, mas também que o produto final gerado por este trabalho abra possibilidade de testes de novas tecnologias que vão se beneficiar desta integração, tais como veículos autônomos, aplicações para *smart cities*, *healthcare*, aplicações de realidade aumentada/virtual.

Espera-se que alguns casos de uso dentro das redes 5G tornem-se dependentes dos conceitos e frameworks MEC para entregar serviços aos usuários finais [26]. Veículos autônomos, *smart cities*, *healthcare* e *Internet of Things* (IoT) são exemplos de casos de uso que são sensíveis a latência e terão ganhos significativos a partir do uso dos conceitos de MEC para seu desenvolvimento. Estes, por exemplo, se aplicam dentro do contexto de serviços URLLC [31].

1.3 Justificativa

Com o surgimento das redes 5G, diversas possibilidades foram abertas para implementação de aplicações e serviços na rede. Há uma expectativa de que aplicações como realidade virtual, realidade aumentada, veículos autônomos, *Tactile Internet* e cenários IoT de conectividade massiva sejam implementados [39]. Contudo, para que isto seja possível, passa-se a necessitar que a rede entregue mais recursos [18].

A partir das características da computação em nuvem, e do tempo de resposta obtido por meio de linhas de transmissões físicas, surge a necessidade de levar essas

aplicações para a borda da rede como possível solução para obtenção dos recursos necessários [49]. Isto por que aplicações executadas na borda, consequentemente estão mais próximas ao usuário, o que permite que a rede tenha vantagens apresentadas na Seção 1.2. Logo, no contexto do 5G, o MEC torna-se um componente essencial para sua composição, de forma a prover meios para que os objetivos propostos no seu desenvolvimento sejam atingidos [18]. Nesse sentido, a integração entre as redes 5G e *frameworks* MEC torna-se necessária.

A arquitetura de referência MEC foi definida pela ETSI, a qual incorpora o ETSI NFV MANO como complemento aos domínios de gerenciamento e orquestração [15]. O MEC definido como uma *Application Function* (AF) interage com o sistema 5G para, influenciar nas decisões de roteamento incluindo (re)seleção de *User Plane Function* (UPF), acessar a *Network Exposition Function* (NEF) para recursos e para interação com o controle das políticas de acesso [16]. Porém, para integrar MEC e 5G há obrigatoriedade que seja seguido este padrão, podendo ser implementado como um serviço implantado no MEC.

1.4 Objetivos

1.4.1 Objetivo Geral

Propor um modelo de integração *open-source* entre MEC e o Núcleo de Rede 5G, estruturado como serviço.

1.4.2 Objetivos Específicos

- Estudo sobre as especificações ETSI MEC e 3GPP 5G;
- Identificação dos principais elementos de integração entre o modelo de referência MEC e o Core 5G;
- Estudar sobre ferramentas que implementam o modelo de referência MEC, como por exemplo o Akraino EALTEdge;
- Projetar uma API de integração MEC e o Core 5G a partir dos modelos de referência definidos pela ETSI e pelo 3GPP;
- Validar a API MTS por meio de implementação;
- Validar o funcionamento da API utilizando aplicação genérica;
- Validar funcionamento da solução fim a fim;

1.5 Metodologia

Para alcançar os objetivos deste trabalho, o primeiro passo foi conhecer as especificidades das tecnologias MEC e 5G. Inicialmente, foram mapeados componentes de *software* da arquitetura de referência MEC [15], entendendo seus objetivos e características. Este mapeamento também foi feito em relação à arquitetura do 5GC [41].

Como núcleo de rede, escolheu-se o *core* 5G free5GC [35]. Esta escolha está amparada no fato do seu código ser *open source*, sua implementação é aderente ao padrão 3GPP e possuir uma comunidade ativa. Além disso, a existência do minicurso desenvolvido por Silva *et al.* (2021) reforça esta escolha visto que este minicurso utiliza o *core* my5G-core que é um *fork* do free5GC [46].

Para plataforma MEC foram feitas pesquisas para identificar plataformas MEC que pudessem ser utilizadas. Uma plataforma foi encontrada e foram feitas tentativas de validação de seu funcionamento. A falta de documentação e suporte impediram avanços neste sentido. Sendo assim, este trabalho utiliza um *cluster* Kubernetes de maneira que simule uma plataforma MEC.

A partir daí, buscou-se desenvolver o serviço o qual este trabalho propõe. Para isso, foram avaliados documentos técnicos os quais indicaram que este serviço é aderente aos objetivos da *Multi-access Traffic Steering* API (MTS API), logo, foi implementado utilizando esta como base. Além disso, foi desenvolvido um serviço de registro (*Service Registry* - SR) o qual é parte do ambiente de validação, além de uma aplicação teste.

A validação deste trabalho foi feita em três partes. A primeira parte validou o funcionamento individual do núcleo da rede, *cluster* e aplicações/serviços. A segunda parte a validação dos registros das aplicações e serviços do SR. Por fim, foram feitas validações do serviço, colheita e avaliação de resultados.

1.6 Contribuições

As contribuições deste trabalho podem ser descritas e resumidas da seguinte forma:

- **Integração entre MEC e 5GC**. Este trabalho mostra uma possibilidade existente de integração entre MEC e 5GC utilizando um serviço disponibilizado através de uma API.
- **Implementação de serviço de Integração**. Uma das possibilidades para que a integração entre MEC e 5G aconteça é através de um serviço. Este serviço pode ser disponibilizado por meio de uma API de comunicação.
- **Expansão da MTS API que possibilita o encaminhamento do tráfego para o 5GC**. A MTS API não possui como foco o envio do tráfego das aplicações para

o 5GC. Sendo assim, expande-se esta API com a criação de um novo serviço para seja possível tráfego MEC e rede 5G.

- **Validação da API em ambiente de testes**. A expansão da API foi validada em ambiente virtualizado, à partir de *deployments* de *framework* MEC simulado e do 5GC.

1.7 Publicações

- XAVIER, Rúben F.; OLIVEIRA-JR, Antonio; FREITAS, Leandro A. Uma discussão da integração entre MEC e 6G. In: **Anais do II Workshop de Redes 6G**. SBC, 2022. p. 13-18. [52]
- CUNHA, Kaíque MR et al. Uma abordagem sobre a integração da Computação de Borda Móvel e a Rede 5G para Internet das Coisas na Agricultura 4.0. In: **Anais da IX Escola Regional de Informática de Goiás**. SBC, 2021. p. 118-131. [10]

1.8 Organização do trabalho

As seções deste trabalho estão organizadas da seguinte forma: a Seção 2 apresenta os conceitos fundamentais do trabalho e trabalhos relacionados. A Seção 3 apresenta o serviço proposto por este trabalho. A Seção 4 mostra os detalhes sobre a implementação da API de comunicação. A Seção 5 apresenta e discute os resultados da validação da proposta. A Seção 6 apresenta a conclusão do trabalho. Por fim as referências.

CAPÍTULO 2

Conceitos e Trabalhos Relacionados

Este trabalho possui caráter científico, logo, a revisão sistemática torna-se obrigatória a fim de construir uma base de conhecimento para o desenvolvimento deste. Sendo assim, a base de conhecimento deste trabalho foi selecionada com base no método RBS *Roadmap* apresentado em [8]. Para apoiar a revisão e documentá-la da melhor forma possível utilizamos a ferramenta Parsifal, que é uma ferramenta online construída para auxiliar durante todo o processo [38].

A execução da pesquisa utilizando as *strings* de busca foi feita utilizando o Google Scholar. A partir dos resultados obtidos, selecionamos artigos de 3 (três) bases principais: IEEE *Digital Library*, ACM *Digital Library* e Springer Link.

2.1 Conceitos Fundamentais

2.1.1 Sistema 5G (5GS)

Os sistemas 5G vem sendo desenvolvidos com objetivo de permitir que haja a conexão entre UE e rede 5G de qualquer lugar [46]. Esse sistema é formado pelo núcleo de rede 5G, uma nova interface de rádio chamada 5G *New Radio* e na ponta da rede os equipamentos de usuário (*User Equipment* - UE) 5G [46].

A *Release* 16 do 3GPP traz um foco abrangente, com o objetivo de expandir o sistema 5G (*5G System* - 5GS). Essa expansão traz por exemplo acesso via satélite 5G, serviços para *Vehicle-to-everything* (V2X), convergência entre redes sem fio e cabeada para 5G, e demais [36]. Também são introduzidos recursos como NR em bandas não licenciadas, *Integrated Access Backhaul* (IAB), aprimoramento para *Multiple-Input Multiple-Out* (MIMO) massivo, fatiamento de recursos, novos recursos para URLLC e IoT Industrial, e suporte para redes não públicas (*Non-Public Networks* - NPNs) [36]. Estes recursos são tratados como cruciais para que as redes atendam aos requisitos necessários.

2.1.2 5G Core (5GC)

Cada vez mais a necessidade da conectividade de redes sem fio tem aumentado. Isto se dá pelo aumento da oferta de serviços de multimídia, e consequentemente, houve um aumento na demanda de tráfego de dados [5]. A Ericsson prevê que em 2026 a produção de dados mensal chegará à 226 exabytes (EB), um valor consideravelmente superior se comparado ao estimado para 2020 que era de 51EB por mês [13].

Desde 1991, com a adoção em larga escala das redes GSM ou também chamadas de segunda geração (2G), as redes evoluíram. O 3G possibilitou o acesso e a navegação na internet por meio de aparelhos celulares [41]. No entanto, com o surgimento do 4G em 2008, tivemos uma adoção abrangente da Banda larga móvel (*Mobile Broad Band* - MBB) [41].

Diferentemente das redes de gerações anteriores, o 5G possui uma arquitetura chamada *Service-Based Architecture* (SBA), a qual é baseada em serviços e microsserviços [2]. Com este novo modelo, o *core* 5G possui características como a separação entre o plano de dados (*Data Plane* - DP) e plano de controle (*Control Plane* - CP), virtualização de funções e fatiamento de rede (*Network Slicing* - NS) [46].

A construção da arquitetura sob o SBA tem o objetivo de se beneficiar das vantagens trazidas por ele como maior eficiência na manutenção e desenvolvimento, associação de microsserviços a recursos exclusivos e ciclos de vida independentes, e melhor escalabilidade com instanciação sob demanda de serviços [41]. A SBA é um modelo de desenvolvimento constituída por NFs que produzem e expõe serviços através de uma interface *Service Based Interface* (SBI). Na SBI os serviços são acessados por meio de APIs e uma combinação dos protocolos *Hypertext Transfer Protocol* (HTTP), *Representational State Transfer* (REST) e *JavaScript Object Notation* (JSON) [46].

O 5GC é parte do sistema 5G (5G *System* - 5GS) juntamente com uma nova interface de radio (*New Radio* - NR). Com base nos conceitos de computação em nuvem, o 5GS tem o seu núcleo orientado a serviços com a arquitetura SBA, além de suporte nativo a fatiamento de recursos de rede, virtualização e computação móvel de borda [19]. A arquitetura do 5GS se baseia em arquitetura não-autônoma (*Non-Standalone* - NSA) e autônoma (*Standalone* - SA). Na arquitetura NSA a rede e sua nova interface de rádio são utilizadas de maneira conjunta com a infraestrutura de um núcleo existente, ou seja, *Evolved Packet Core* (EPC). Já no SA a interface de rádio está conecta a nova arquitetura do núcleo de rede proposta para o 5G.

Funções de Rede

Os componentes de software do 5GC são construídos sob os conceito de *Network Function Virtualization* (NFV) e *Software Defined Networks* (SDN). Sendo assim, as

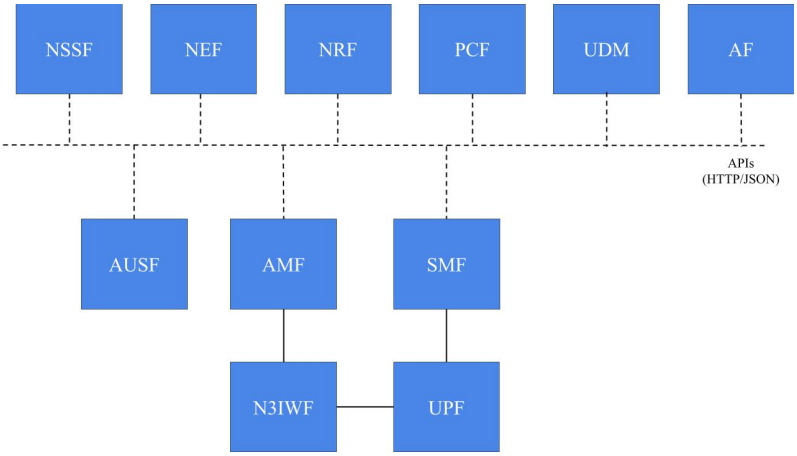

Figura 2.1: Arquitetura do 5GC. Baseada em [46]

funções que antes eram executadas por *hardwares* fechados passam a ser virtualizadas, e no contexto do 5G são virtualizados não só as funções de rede mas também a rede de transporte, acesso e *core* [32]. Ao discutir a associação MEC e 5GC, algumas funções tem participação direta.

O **User Plane Function** (UPF) é o ponto central do plano de usuário do *core* 5G. Esta NF é onde o Equipamento de Usuário (*User Equipment* - UE) irá se conectar a rede de dados (*Data Network* - DN) [41]. As funções do UPF são controladas pela *Session Management Function* (SMF). Além de criar a ponte entre UE e DN, o UPF também tem funções de roteamento, inspeção e encaminhamento de pacotes, gerenciamento de regras de QoS, entre outras [41].

A **Session Management Function** (SMF) tem como uma de suas funções o gerenciamento do UPF, configurando-o para prover a conexão entre UE e DN [46]. Esta configuração envolve o gerenciamento da sessão do usuário, alocação de endereço IP e as sessões IP PDU [41].

A **Application Management Function** (AMF) interage com a rede de dados e via UE através das interfaces N2 e N1 respectivamente [41]. A AMF permite sinalização segura (encriptada) com a UE, permitindo registro, autenticação e mobilidade na rede [41].

A **Network Exposure Function** (NEF) tem a capacidade de monitorar a rede 5G de forma a permitir que os eventos se tornem acessíveis a aplicações autorizadas e também as NFs [41]. Ela também permite que aplicações possam gerenciar regras de QoS existentes ou a inserção de novas, requerer prioridade, além da possibilidade de entrega de informações sobre o UE [41].

Fatiamento de Rede

O fatiamento de rede, do inglês *network slice* (NS), é uma das principais tecnologias dentro do ecossistema do 5G, permitindo a instanciação de redes lógicas sobre uma infraestrutura comum [41]. *Network slices*, em geral, tem como objetivo atender a um cliente definido e seus objetivos, ou seja, redes lógicas dedicadas, de forma que os recursos necessários sejam configurados e estejam disponíveis ali [41, 47].

Interfaces de Rede

As interfaces de rede aqui descritas são pontos de comunicação entre dois componentes de software, neste caso, do 5GC. Estas interfaces em sua grande maioria possuem padronizações específicas de funcionamento e protocolos. Dentre as diversas interfaces padronizadas e presentes no *core*, destaca-se 4, sendos estas $N3$, $N4$, $N6$ e $N9$. As demais interfaces estão fora do escopo deste trabalho.

- **N3** - Provê a comunicação de dados entre interface de rádio e UPF. Por esta interface os dados são encapsulados para o roteamento [41].
- **N4** - O SMF controla o UPF e isto é feito através da interface N4. Toda e qualquer comunicação feita entre ambos é feita por esta interface [41].
- **N6** - O UPF permite conexão com redes de dados externas utilizando a interface N6. Neste ponto os dados não são mais encapsulados, porém, existe a possibilidade com a utilização de uma rede virtual privada (*Virtual Private Networking* - VPN) [41].
- **N9** - No Core 5G UPFs podem ser implantados em série. Para conectá-los é utilizada a interface de rede N9. Mobilidade é um caso de uso que se beneficia disto [41].

2.1.3 Arquitetura de Referência MEC

O MEC, assim como o 5GC, possui uma entidade que o padroniza. A ETSI, entidade responsável, tem trabalhado constantemente na padronização e expansão do MEC. Constantemente são publicados documentos técnicos que discutem desde sua arquitetura até o desenvolvimento de serviços no MEC.

A arquitetura de referência MEC é a base para o desenvolvimento de qualquer *framework* MEC. Esta arquitetura está definida em [15] e pode ser vista na Figura 2.2. Na arquitetura é possível observar todos os componentes pertencentes a tal, como MEC *Platform* (MEP), MEC *Host*, MEC *Platform Manager* (MEPM).

Na arquitetura MEC, o MEC *Host* é o componente que abrange elementos importantes como MEP e a O Gerenciador da Infraestrutura de Virtualização (*Virtualization*

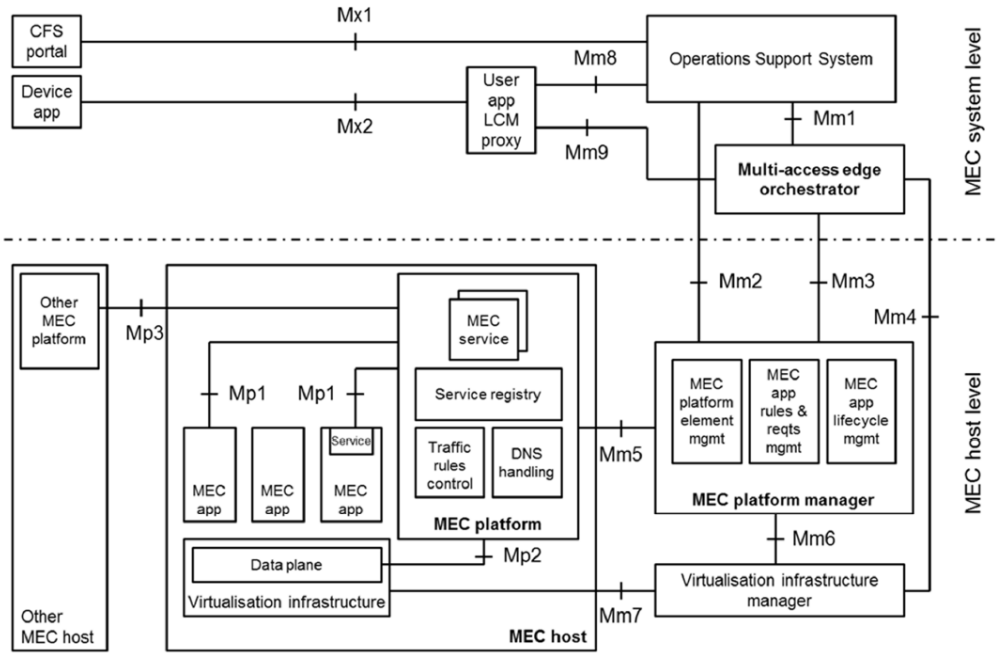

Figura 2.2: Arquitetura de referência MEC. Retirada de [15]

Infrastructure Manager - VIM), além das aplicações MEC (MEC app). O VIM é responsável por prover armazenamento, processamento e recursos de rede para as aplicações MEC, além disso, também é onde o plano de dados está localizado [15]. O MEP possui funções de gerenciamento com maior peso no que tange às aplicações. Entre suas funções está a responsabilidade de disponibilizar um ambiente no qual as aplicações tenham a possibilidade de descobrir, disponibilizar e consumir serviços, comunicação com o plano de dados no que tange a sua configuração a partir de regras de tráfego recebidas, hospedar os serviços MEC, servidor DNS, entre outros [15].

O MEPM possui funções de gerenciamento. Entre elas, gerenciar o ciclo de vida das aplicações e levar informações ao orquestrador de borda (*Multi-Access Edge Orchestrator* - MEO), gerenciar elementos do MEP, além de gerenciar regras e requisitos de aplicativos [15]. Entre as funções exercidas pelo MEO estão a manutenção de uma visão global sobre o sistema MEC baseada nos MEC *Hosts* implantados, serviços e recursos disponíveis e também na topologia geral, seleção de *hosts* MEC baseado em requisitos, instanciação, realocação (se disponível) e encerramento de aplicativo [15].

2.2 Trabalhos Relacionados

Na literatura, poucos trabalhos discutem a integração entre 5G e MEC propriamente dita. No geral, a discussão é feita de forma mais abrangente, tratando principalmente dos benefícios e/ou aplicações em cenários que estão fora do alcance e objetivo

deste trabalho. Abaixo são apresentados os artigos *A new Approach to 5G and MEC Integration* [49], *Toward slicing-enabled multi-access edge computing in 5G* [26], *MEC in 5G Networks* [25], *On O-RAN, MEC, SON and Network Slicing integration* [29].

2.2.1 A new Approach to 5G and MEC Integration

Uma arquitetura para integração entre MEC e redes 5G foi apresentada por Tomaszewski *et al.* (2020) [49]. A arquitetura proposta é apresentada na Figura 2.3. Esta arquitetura é baseada na arquitetura *Distributed Autonomous Slice Management and Orchestration* (DASMO) [28].

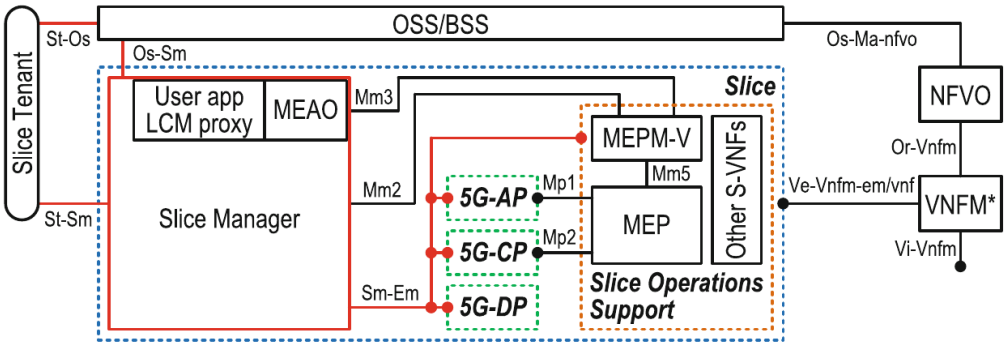

Figura 2.3: Arquitetura DASMO habilitada para integração MEC e sistema 5G. Retirada de [49]

A arquitetura DASMO se baseia em paradigma de domínio único, entretanto, é possível transformá-la em multi-domínio adicionando *Inter-Domain Operations Support* (IDOS) como parte do *Slice Operations Support* (SOS), atuando como um *gateway inter-slice* para troca de informações. Apesar do detalhamento deste trabalho sobre a arquitetura proposta e também seu embasamento nos padrões, não é apresentado nenhuma validação em termos de implementação. Dessa forma, o trabalho de Tomaszewski *et. al.* deixa uma lacuna no que tange à validação da arquitetura proposta em seu trabalho.

2.2.2 Toward slicing-enabled multi-access edge computing in 5G

Ksentini and Frangoudis (2020) propõem em seu trabalho uma arquitetura de gerenciamento e orquestração para *Network Slice* (NS) [26]. A arquitetura proposta por este trabalho pode ser vista na Figura 2.4. Sua definição utiliza como base conceitos como *Network Slice Instance* (NSI) e *Network Slice Subnet Instance* (NSSI). O NSI possui em sua estrutura NFs definidas com recursos lógicos e físicos correspondentes. Informações destas *Network Functions* (NF) são descritas a partir de *Network Slice Template* (NST) que serão utilizados para definição dos *slices*.

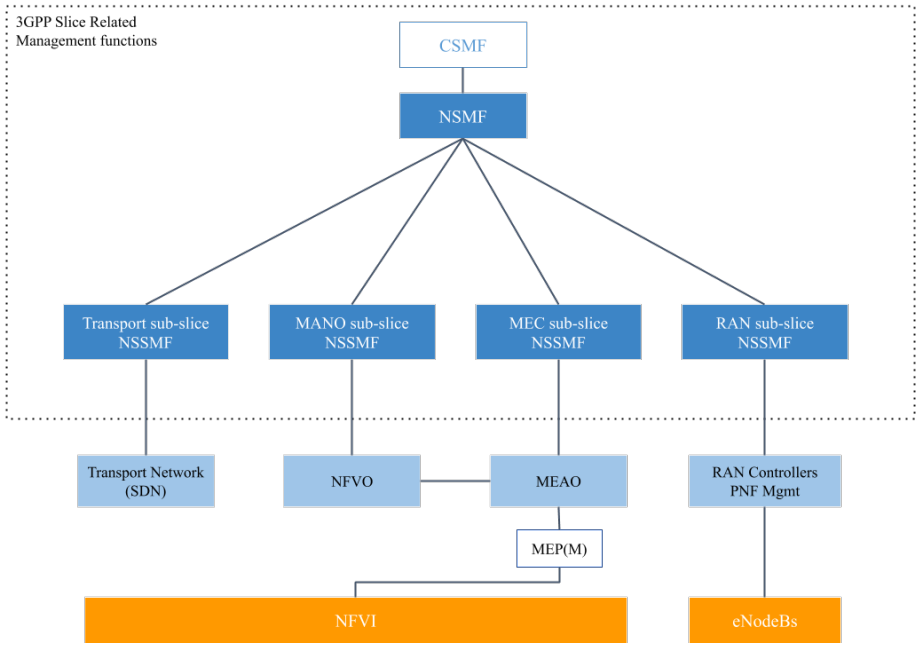

Figura 2.4: Arquitetura de gerenciamento e orquestração para *Network Slice*. Baseada em [26]

A contribuição deste trabalho está na divisão de uma fatia de rede completa, em partes menores, nas quais poderão ser feitos *deployments* como *Virtual Network Function* (VNF). Dentro desta divisão, são propostos dois modelos: *in-slice* e *multi-tenancy*. A diferença básica entre eles está na instanciação do *MEC Platform* (MEP). No modelo *in-slice*, o MEP é instanciado dentro da *slice* e no *multi-tenancy* já existe um MEP fora da *slice* que é compartilhado. Nesta arquitetura o MEP tem função de se comunicar com a rede 5G para que sejam criadas políticas de direcionamento de tráfego. Esta comunicação pode ser de forma direta (quando é um MEP confiável), ou seja, MEP se comunica diretamente com PCF do núcleo de rede, ou através da *Network Exposure Function* (NEF). O ambiente de validação desta proposta foi adaptado para utilização do OpenAirInterface (OAI). Para isso, a rede principal do OAI foi estendida para separação do CP e DP, e também para comunicação com MEP via *Mp2*. Este trabalho propõe uma nova arquitetura e tendo como foco principal permitir que a arquitetura MEC dê suporte para o *Network Slicing*.

2.2.3 MEC in 5G Networks

Kekki *et al.* (2018) define uma arquitetura de integração entre 5G *System* (5GS) e arquitetura MEC que pode ser vista na Figura 2.5 [25]. Nesta arquitetura, o orquestrador MEC pode interagir diretamente com a NEF, e em alguns casos diretamente com outras funções dentro do 5GS. O ponto central está na integração entre os dois sistemas

através da interface N6, a qual se liga ao UPF. Para o MEC, o UPF é visto de forma distribuída e configurável, e em alguns *deployments*, pode ser local tornando-se parte de sua arquitetura.

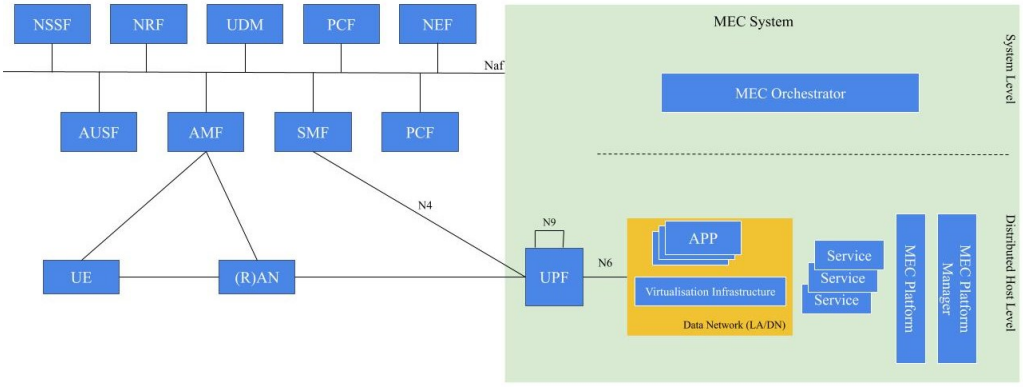

Figura 2.5: Arquitetura de integração entre MEC e 5GS. Baseada em [25]

2.2.4 On O-RAN, MEC, SON and Network Slicing integration

Kuklinski *et. al* [29] apresenta um novo conceito de integração entre a plataforma O-RAN, *Self-Organizing Network* (SON), MEC, e *Network Slice*. A partir da identificação das lacunas existentes em cada uma das tecnologias citadas, é proposta uma arquitetura a qual utiliza como base a arquitetura da plataforma O-RAN, com foco no *near-Real Time RAN Intelligent Controller* (near-RT RIC). A arquitetura proposta propõe uma alteração ao near-RT RIC, e as tecnologias MEC, SON e NS de forma a integrar todas as arquiteturas em um mesmo componente, chamado *Integrated* near-RT RIC (I-near-RT RIC), sendo assim, todas as tecnologias passam a ser implementadas em um mesmo *host*. A Figura 2.6 mostra a integração entre O-RAN, 5GC e MEC em domínios com dois I-near-RT RIC.

Especificamente ao MEC, sua arquitetura original é aproveitada, se tornando a solução para problemas identificados anteriormente como a cooperação entre near-RT RICs a partir do *Multi-Access Edge Application Orchestrator* (MEAO). As aplicações MEC são implementadas como xApps (aplicações da plataforma O-RAN). O *Multi-Access Edge* Platform (MEP) é decomposto e passa a ser chamado de O-MEP. Importante ressaltar que a integração do MEC com o 5GC acontece através da interface Mp2 como uma *Application Function* (AF). Dessa forma, o 5GC enxerga o MEC como *Network Application Function* (NAF). Este trabalho traz uma proposta apenas em nível conceitual. Sendo assim, não há validação de sua proposta em *testbed*.

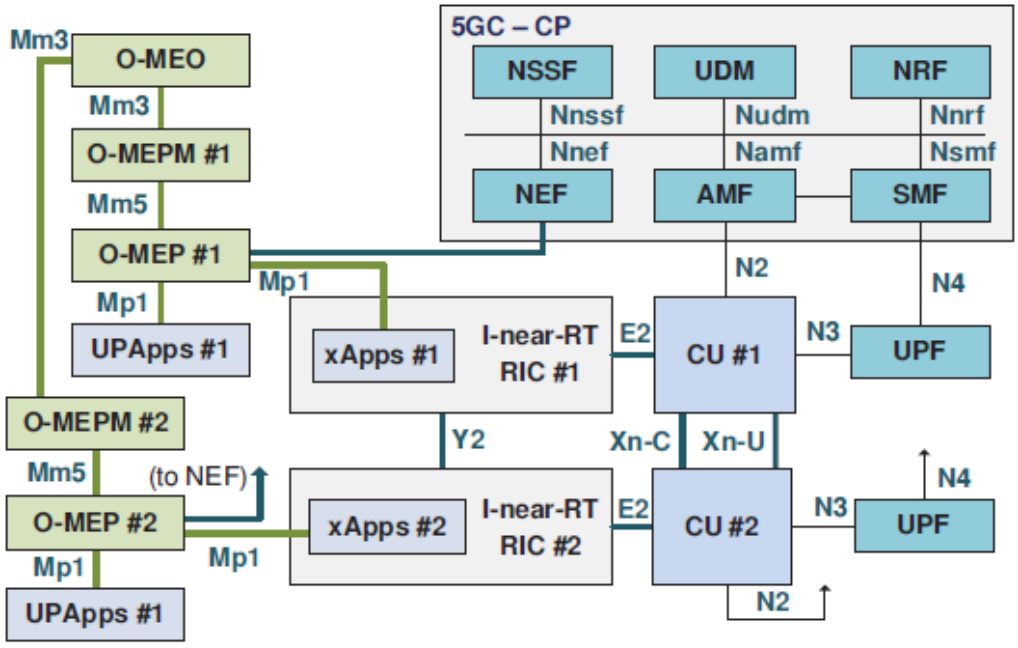

Figura 2.6: Visão geral de O-RAN integrado, 5GC-CP e MEC para dois domínios RIC I-near-RT. Retirado de [29].

2.2.5 Comparação de Trabalhos Relacionados

A Tabela 2.1 apresenta uma comparação dos principais trabalhos relacionados com a proposta deste trabalho. Esta comparação auxilia na percepção e mapeamento das características entres os trabalhos relacionados e este trabalho. Os critérios para comparação foram:

- **Proposta de Integração 5G-MEC** - Se o trabalho em questão discute a integração entre 5G e MEC;
- **Integração Através de Serviço** - Se a integração proposta no trabalho é feita através da implementação de um serviço;
- **Integração Através de Nova Arquitetura** - Se a integração entre 5G e MEC proposta inclui uma alteração nas arquiteturas, inclusão ou alteração de componentes pré-definidos.
- **Validação da Proposta** - Se a proposta apresentada foi validada com implementações, e podem ser observados os resultados desta validação.
- **Validação com 5GC SA** - Se existe validação, esta validação utiliza um núcleo de rede 5G.

O primeiro ponto analisado trata-se do tema central deste trabalho e que foi analisado também nos trabalhos supracitados. Fato é que em alguns deles não tratam apenas de integração do MEC com o 5G, mas também de outras tecnologias como SON

em [29]. Logo, todos os trabalhos trazidos para esta discussão apresentam propostas que envolvem a integração 5G-MEC, sendo então diferenciados por suas propostas.

Tabela 2.1: Comparação da proposta deste trabalho com trabalhos relacionados.

Trabalhos	**Proposta Integração 5G-MEC**	**Integração através de Serviço**	**Integração através de Nova Arquitetura**	**Validação da Proposta**	**Validação com 5GC SA**
[25] MEC in 5G Networks	X		X		
[49] A new Approach to 5G and MEC Integration	X		X		
[26] Toward slicing-enabled multi-access edge computing in 5G	X		X	X	
[29] On O-RAN, MEC, SON and Network Slicing integration	X		X		
Este Trabalho	X	X		X	X

Avaliou-se também como foi feita a integração nos trabalhos. Neste sentido, observou-se dois pontos: i) Se a integração foi feita através da implementação de um serviço MEC que possibilitasse a comunicação entre rede 5G e MEC; ii) Se a proposta apresentada nos trabalhos apresentam uma nova arquitetura que possibilita esta integração. Nesses pontos, este trabalho diferencia-se dos demais, trazendo como proposta um serviço. Os demais trabalhos propõem integração baseada em novas arquiteturas ou adaptação de arquiteturas já existentes. Em Tomaszewski *et al.* [49] a arquitetura DASMO foi base para a arquitetura proposta, de maneira que os conceitos base foram reutilizados. Neste trabalho, observa-se que aplicações, CP e DP do 5G, e componentes relacionados a gerenciamento de *slice* ocupam uma mesma fatia de rede. Além disso, MEP foi inserido dentro do que foi chamado de *Slice Operations Support*. Em [29] tomou-se por base o que se tem de documentação pública do O-RAN para se propor uma nova arquitetura,

com foco principal em alterar um de seus componentes (I-near-RT RIC). Além disso, a arquitetura proposta não se mantém apenas entre MEC e 5G, incorporando também NS e SON, sendo tratados como tecnologias parcialmente complementares e sobrepostas. Em [26] foi proposta uma nova arquitetura de orquestração e gerenciamento tendo como objetivo a possibilidade de se ter o MEC em 5G. Contudo, esta integração se dá no contexto de fatias de rede, além disso, o *core* de rede utilizado é adaptado para 5G. Em todos os trabalhos, independente de como isso é feito, o MEP se comunica diretamente com o CP do núcleo de rede através da interface MP2 como uma AF.

Por fim, foi observado se as propostas destes trabalhos foram validadas. Avalia-se dois pontos: i) Se a proposta foi validada de alguma forma; ii) Se a proposta validada, utiliza um core 5GC. Nesse sentido, vê-se que apenas 1 dos trabalhos relacionados teve sua proposta validada, sendo este [26]. Este trabalho foi validado em *testbed* utilizando diferentes máquinas como servidores de borda e também um *core* de rede OAI. Porém, ao observar este trabalho identifica-se componentes da arquitetura de núcleos de rede *Evolved Packet Core* (EPC) como MME e HSS. Sendo assim, apesar de validar a proposta, não utiliza um *core* de rede 5G SA. Este trabalho tem como seu foco de avaliação em como a arquitetura proposta consegue criar fatias de rede para MEC dentro do contexto do 5G que possibilitem a comunicação entre ambas tecnologias. Já em Tomaszewski *et al.* [49] e [29] não são apresentadas nenhuma validação da proposta arquitetural feita.

Sendo assim, baseado na observação dos trabalhos, vê-se que há um diferencial em nossa proposta tanto em como a integração é feita quanto na sua validação. A integração proposta se dará através da implementação de um serviço e será validada através de ambiente *testbed* utilizando um *core* de rede 5G SA. Além disso, é possível observar que ao propor uma nova arquitetura ou a alteração de uma existente apenas uma proposta foi validada em ambiente real, e as demais se restringiram no campo teórico e conceitual.

Kekki *et al.* (2018) discute que a possibilidade da implantação como uma aplicação que provê um serviço [25]. Esta possibilidade pode ser uma vantagem comercial levando em consideração o seu tempo para lançamento, e também que este serviço pode se tornar uma serviço de plataforma MEC à medida que a tecnologia e os modelos de negócios que a utilizem se estabilizem. Com isso, reforça-se aquilo que foi proposto de maneira que alimente o cenário acadêmico com uma proposta que diferencia-se das demais apresentadas.

CAPÍTULO 3

Proposta de um Serviço para integração MEC e Núcleo 5G

O MEC traz consigo diversas vantagens já apresentadas anteriormente para as redes 5G. Contudo, para que a integração entre MEC e 5G ocorra é necessário que seja especificado um componente de comunicação entre ambos. Este componente deve seguir as definições dos grupos que padronizam o *5G System* (5GS) e MEC. Kekki *et al.* (2018) discute que uma das possibilidades para integração é uma aplicação que provê um serviço ou que este mesmo serviço seja implantados juntamente aos serviços da própria plataforma sendo então parte dela [25]. A Figura 3.1 apresenta a implantação das APIs no ambiente MEC. A linha pontilhada se refere às duas possibilidades citadas anteriormente, sendo utilizado como exemplo as APIs BWM e MTS as quais serão discutidas no decorrer deste capítulo. Kekki *et al.* (2018) diz que a implantação como uma aplicação pode acelerar o processo de lançamento e implantação se comparado a outra. Isto porque ao implantar o serviço como parte da plataforma, toda sua estrutura e funcionalidade precisam estar atreladas aos componentes da mesma, trabalho que não é trivial. Sendo assim, uma API de comunicação que provê um serviço, implantada como uma aplicação, é um exemplo de como este serviço pode ser especificado. Logo, a proposta deste trabalho é especificação de um serviço que possibilite a integração entre MEC e 5GC, o qual seguirá a implantação como uma aplicação.

Este serviço deve permitir que uma aplicação cliente presente no UE se comunique com a sua respectiva aplicação servidor, isolada internamente na plataforma MEC, e vice-versa. Dessa forma, o serviço deve receber as informações do destino e o conteúdo que deve ser enviado e, tendo validado a existência destes, proceder com o encaminhamento do conteúdo. Após isto, o serviço deve aguardar uma resposta e devolvê-la para o seu requisitante. Este fluxo se repete em ambos sentidos. As informações sobre o destino devem ser enviadas no corpo da requisição para que o serviço possa processá-las de maneira que entenda como fazer este encaminhamento. Para que haja um nível de segurança, para usar este serviço será necessário um *token* que é gerado pelo SR ao verificar a existência de um serviço. Inicialmente a proposta deste trabalho não inclui o envio de

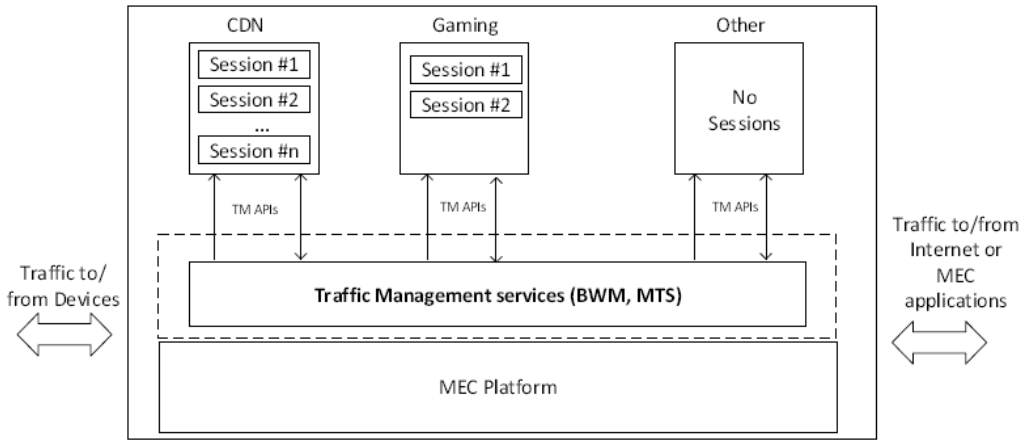

Figura 3.1: Descrição dos serviços das APIs de gerenciamento de tráfego. Retirado de [17]

arquivos de mídia, sendo esta uma das possibilidades de expansão.

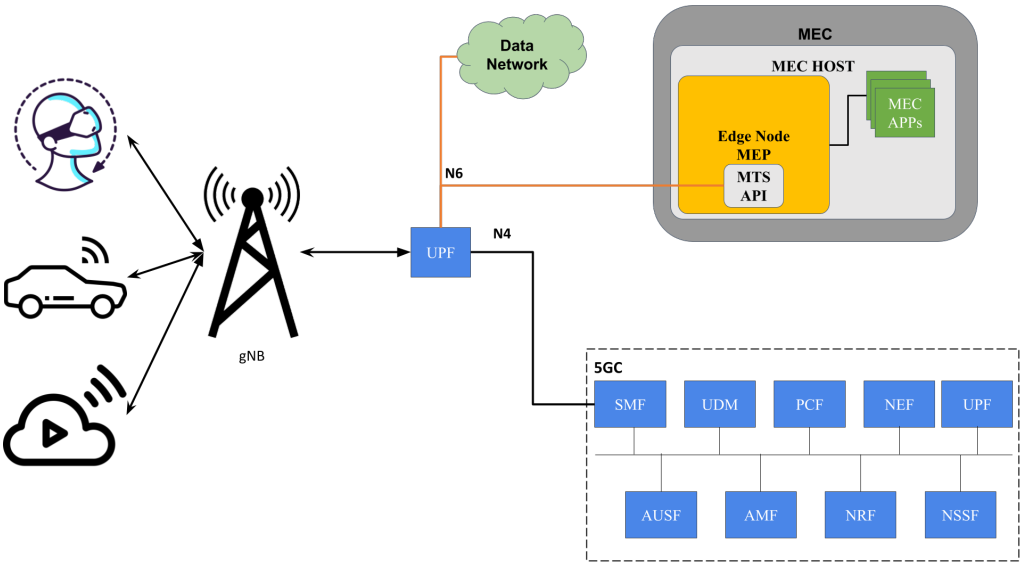

Figura 3.2: Arquitetura geral da Proposta.

A Figura 3.2 apresenta a arquitetura geral da proposta. Nesta figura é possível observar o serviço da MTS API no MEP estando este acessível através da interface *N6*. O MEP é o componente no qual o serviço estará sendo executado independente da maneira que é implantado (aplicação ou parte da plataforma). O MEP é parte do MEC HOST que abrange não só o MEP como também as aplicações MEC, fechando assim o escopo dos componentes MEC abrangidos por este trabalho. Também é possível visualizar a arquitetura do 5GC SA, conectado a um UPF através da interface *N4* que já foi definida anteriormente. Vê-se que o *core* possui 2 UPFs e apenas 1 SMF. Isso é possível no 5GC SA devido ao modelo de arquitetura SBA. Nessa figura é apresentada a interface de rádio gNB e também algumas figuras as quais representam possíveis aplicações que

se comunicam ao MEC. Salienta-se que comunicação intra core, ou seja, entre NFs, e também interface de rádio não estão inseridos no escopo deste trabalho.

Na Figura 3.3 é apresentado um diagrama de classes no qual são apresentados os componentes de software os quais fazem parte deste trabalho. Para o serviço proposto serão implementadas duas funções: *postData* e *makeRequest*. A função *postData* é a principal. Sua função será receber as informações presentes no corpo da requisição. A partir disto, deverá verificar a presença do endereço IP de destino, método HTTP que deverá ser utilizado e os dados da requisição. O Modelo de Dados 3.1 apresenta a especificação de como estas informações devem chegar à API.

Modelo de Dados 3.1: Modelo de dados recebido pela MTS API.

```
1 {
2     "service_ip": "<service_ip_com_endPoint>",
3     "method": <method>,
4     "data": {
5         <app_data>
6     }
7 }
```

Ao obter todas estas informações, é chamada a função *makeRequest*. Por sua vez esta função é responsável por encaminhar os dados fazendo uma nova requisição para o destino. Para isso irá utilizar as informações extraídas na função anterior. A função *makeRequest* fará então uma nova requisição ao destinatário para enviar as informações e aguardará pela resposta. Esta resposta é repassada de volta para *postData* que devolve ao UE. Além destes dois métodos, o serviço possui uma função para se registrar no SR, chamada *makeRegistry*. Esta função possui todas as informações sobre o serviço como IP, porta, e *end-points*. Estas informações são então enviadas ao SR efetuando um registro, e então, aplicações interessadas podem verificar e receber estas informações. O modelo de dados especificado será apresentado ainda nesta seção.

Ainda na Figura 3.3 é possível observar o *Service Registry*. Este serviço será o ponto de registro dos serviços e aplicações MEC disponíveis. Todas as aplicações e serviços MEC existentes deverão se registrar com endereço IP, porta, e também seus *end-points*. É necessário que haja no mínimo 1 *end-point*. O Modelo de Dados 3.2 apresenta a especificação de como estas informações deverão estar estruturadas. Pós-registro, o UE poderá verificar se o serviço desejado está disponível. Caso esteja, são repassadas todas as informações necessárias para comunicação. Estas funcionalidades serão implementadas nas funções *handle_registry_data*, *handle_specific_data*, *handle_get*, *handle_put*, *handle_post*, *handle_delete*, *encodeToken*. A função *handle_registry_data* é responsável por receber e verificar as informações da aplicação/serviço. Caso estejam corretas, é chamada a função *handle_post* que armazena estas informações para uso posterior.

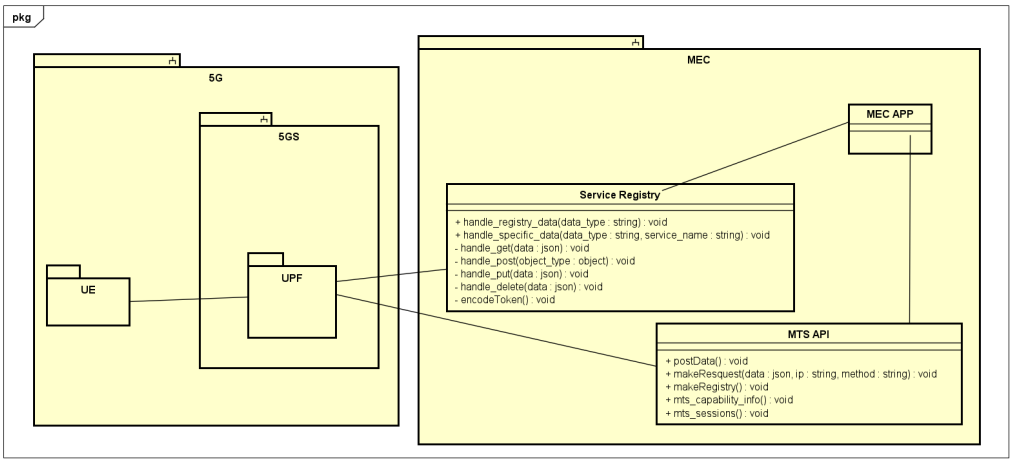

Figura 3.3: Diagrama apresentando métodos que serão implementados.

A função *handle_specific_data* é que permite acesso a informações sobre aplicações específicas e também chama as demais funções supracitadas. A função *handle_get* é de extrema importância neste contexto visto que é esta que, quando um UE verificar a existência de um serviço, e este existir, invocará o método *encodeToken* para que, juntamente com as informações da aplicação/serviço, seja enviado também um *token* para utilização da MTS API. As funções *handle_put* e *handle_delete* são funções que podem também ser chamadas pela *handle_specific_data* para alteração e exclusão, respectivamente, das informações armazenadas.

Para este trabalho será seguido a ideia de que o serviço será implantado como uma aplicação. Porém, este mesmo serviço poderia ser implantado como parte da plataforma MEC caso desejado. Nesse caso, o serviço também estaria atrelado ao MEP e disponível para que comunicação através da interface *N6*. Esta aplicação será desenvolvida como uma API REST. A implantação como aplicação não possui necessidade de comunicação com o CP do núcleo de rede 5G, se atendo especificamente ao DP e, consequentemente, a interface *N6*. Para que então haja comunicação, as informações trafegarão através de requisições HTTP, e, em específico, o serviço receberá requisições utilizando o método *POST*.

Modelo de Dados 3.2: Modelo de dados para registro.

```
{
    "service_name": "<service_name>",
    "service_ip": "<service_ip_com_endPoint>",
    "service_port": "<service_port>",
    "service_endPoint_Post": "<endPoint_Post>",
    "service_endPoint_Get": "<endPoint_Get>",
    "service_endPoint_Put": "<endPoint_Put>",
    "service_endPoint_Delete": "<endPoint_Delete>"
}
```

O documento técnico que apresenta a arquitetura de referência do MEC também apresenta duas APIs que ofertam serviços de gerenciamento de tráfego [15]. BWM é a API que, quando disponível, cuidará da alocação de banda de ou para aplicações, e também a priorização de certos tráfegos na rede. A API MTS, quando disponível, é responsável pelo gerenciamento do tráfego no que tange ao direcionamento, divisão e duplicação através de múltiplas redes de acesso de forma a atender os requisitos de rede solicitados pelas aplicações [17]. Por entender que os objetivos da API e do serviço já descrito são parcialmente compatíveis, propõe-se a expansão da MTS API de forma que ela também ofereça o serviço de encaminhamento de tráfego para a rede de dados externa ao MEC, neste caso o 5GC a partir de um novo serviço por ela ofertado. Por isto, são apresentados três métodos, além dos quais já foram apresentados, na MTS API mostrada na Figura 3.3, sendo eles *mts_capability* e *mts_sesssions*. Estes métodos são descritos documentados e descritos em [17]. A função *mts_capability* está atrelada a um *end-point* o qual tem por objetivo permitir o acesso à informações sobre a capacidade da MTS API como o tipo de acesso à rede. Já a função *mts_sesssions* possui atribuições relacionadas à sessão na API, chamada MTS *Session*. Uma MTS *Session* possui elementos para configurações da rede como latência máxima, nível de prioridade de tráfego e *throughput* mínimo. Sendo assim, esta cria e gerencia as seções requisitadas por aplicações para que, a partir das informações repassadas, seja feito o gerenciamento da rede para aquela aplicação.

CAPÍTULO 4

Implementação da API proposta

Para alcançar os objetivos deste trabalhos foi implementada uma API com duas funções já descritas no Capítulo 3. O desenvolvimento utilizou linguagem Python utilizando *framework* Flask. O micro *framework* Flask, é muito utilizado para desenvolvimento de aplicações *web* por ser simples, rápido e robusto [20]. Seu funcionamento se dá por um *end-point*, o qual recebe e encaminha informações.

As funções implementadas estão atreladas a um *end-point* que recebe requisições HTTP. A função principal recebe requisições somente através do método *POST*. Sendo assim, toda e qualquer requisição que não utiliza este método é rejeitada. Ao receber a requisição, a API invocará o método *postData*. Esta função pode ser vista no Código 4.1. Este método validará o conteúdo mínimo necessário para que os dados sejam encaminhados. As 2 linhas iniciais são: 1) A definição do endereço atrelado à função e o método que utilizará; 2) Um *decorator* implementado pela biblioteca JWT para proteger um *end-point* com requisitando um *token* JWT. A validação do conteúdo necessário pode ser observada na linha 6, sendo então validada a presença do IP de destino e também é método HTTP que será utilizado para encaminhar as informações, que caso estejam presentes são colocados em variáveis separadas. A presença dos dados é validado na linha 9 a fim de que não seja um parâmetro obrigatório em caso de utilização dos demais métodos HTTP os quais não os requerem. Caso haja dados a chamada da função *makeRequest* é feita com o parâmetro *data*, caso não haja é feita sem. As linhas 14 - 20 fazem o tratamento da resposta baseado na resposta da função *makeRequest*, sendo o código 201 utilizado para avaliar o sucesso da requisição e, em caso de erro, o código 404 é enviado na resposta ao cliente. Caso algum conteúdo necessário esteja faltando para que a requisição de encaminhamento, é enviada uma mensagem de erro com o código 404 junto com o texto *"Service IP and Method Requested"*, sendo este a condição presente nas linhas 21-23. A linha 24 devolve ao cliente a resposta seja ela qual for.

Código 4.1: Função postData implementada na API

```
1 @dataM.route('/', methods=['POST'])
2 @jwt_required()
3 def postData():
```

```
4       try:
5           request_info = request.get_json()
6           if request_info["service_ip"] and request_info["method"]:
7               service_ip = request_info["service_ip"]
8               method = request_info["method"]
9               if request_info["data"]:
10                  data = request_info["data"]
11                  response = makeRequest(data, service_ip, method)
12              else:
13                  response = makeRequest(service_ip, method)
14              if response.status_code == 201:
15                  response =
16                   Flask.response_class(response=response.content,
17                    status=201)
18              else:
19                  response =
20                   Flask.response_class(response=response, status=404)
21          else:
22              response = Flask.response_class(
23                response="Service␣IP␣and␣Method␣Requested", status=404)
24          return response
25      except Exception as e:
26          return e, 500
```

Para encaminhar as informações para o seu destino, a API executará uma nova requisição HTTP. Para isto a função *makeRequest* é chamada, recebendo como parâmetro as informações citadas. Esta função pode ser vista no Código 4.2. As linhas 3, 10, 13, e 17 verificam qual método deverá ser utilizado para a requisição. Caso os métodos sejam *POST* ou *PUT* os dados são necessários para que a requisição seja feita, sendo transformados em formato *JSON* nas linhas 7 e 15, tendo como diferença que no método *POST* é montado também um *header*. Aos demais métodos não é necessário dados na requisição, tornando então o parâmetro *data* não obrigatório, e, por isso, no cabeçalho da função o parâmetro recebe um valor vazio como padrão. A requisição é então feita utilizando o método adequado e, após fazer a requisição, a função recebe a resposta, devolve a *postData* que por fim devolve ao cliente.

Deve-se ressaltar que para o propósito deste trabalho não foram implementados outros métodos que compõe a MTS API. Isto por que estes métodos não eram necessários para alcançar o objetivo do serviço proposto.

Código 4.2: Função makeRequest implementada na API

```
1   def makeRequest (ip, method, data=""):
2       try:
3           if method == "POST":
4               header = {
```

```
5               'Content-Type': 'application/json',
6               }
7               data_json = json.dumps(data)
8               response = requests.post(url=ip, data=data_json,
9                 headers=header)
10              return response
11          if method == "GET":
12              response = requests.get(ip)
13              return response
14          if method == "PUT":
15              data_json = json.dumps(data)
16              response = requests.put(ip, data=data)
17              return response
18          if method == "DELETE":
19              response = requests.put(ip)
20              return response
21      except Exception as e:
22          return e, 500
```

Para o SR implementamos o mínimo necessário para se adequar à aquilo que se esperava. Para isso foram implementadas algumas funções. A primeira delas é a função *handle_registry_data* que é responsável pelo recebimento das informações das aplicações e serviços que desejam se registrar. Esta função pode ser vista no Código 4.3. A função recebe como parâmetro o tipo de dados, neste caso será *registry*. Não são implementados outros modelos, porém, em trabalhos futuros podem ser implementados outros. O conteúdo da requisição é então resgatado na linha 4 e validado chamado a função *validate_content* para que seja validado o conteúdo necessário para registro. Esta função pode ser vista no Código 4.4. Então é chamada a função *handle_post* que salva as informações no banco de dados. Logo após o registro, é gerado um *token* para esta aplicação/serviço na função *encodeToken* utilizando biblioteca JWT.

Código 4.3: Função makeRequest implementada na API

```
1  @service_registry.route('', methods=['POST'])
2  def handle_registry_data(data_type):
3      try:
4          rc = request.get_json()
5          result_msg, result = validate_content(rc)
6          if result:
7              obj_type = getattr(sys.modules[__name__], data_type)
8              result = handle_post(obj_type)
9              token, encode_status = encodeToken()
10             if encode_status == 201:
11                 response =
12                     Flask.response_class(response=token, status=201)
```

```
13                  return response
14              else:
15                  response = "Token generation failed"
16                  response =
17                     Flask.response_class(response=token, status=400)
18                  return response
19          else:
20              response =
21                 Flask.response_class(response=result_msg, status=406)
22              return response
23      except Exception as e:
24          return e
```

Código 4.4: Função validate_content implementada no Service Registry.

```
1  def validate_content(rc):
2      if rc["service_name"]:
3          if rc["service_ip"]:
4              if rc["service_port"]:
5                  if rc["service_endPoint_Post"] or
6                     rc["service_endPoint_Get"] or
7                        rc["service_endPoint_Put"] or
8                           rc["service_endPoint_Delete"]:
9                      return "Validate Complete",True
10                 else:
11                     return "Missing Service EndPoint (min. 1)"
12             else:
13                 return "Missing Service Port", False
14         else:
15             return "Missing Service IP", False
16     else:
17         return "Missing Service Name", False
```

No Service Registry também foi implementado a função *handle_specific_data*. Esta função tem por objetivo permitir que os dados armazenados no banco de dados sejam não só alterados como também recuperados. Os métodos *PUT* e *DELETE* permitem ao cliente alterar os registros. Já com a utilização do método *GET* o cliente pode obter as informações armazenadas. Neste caso específico, o UE acessará com o método *GET*. Ao obter os dados, será chamada a mesma função já apresentada para que seja gerado um *token* ao usuário. Este token será utilizado para que possa utilizar o serviço de integração desenvolvido. As funções *handle_put*, *handle_delete* e *handle_get* assumem respectivamente as funções supracitadas.

Código 4.5: Função handle_specific_data implementada no Service Registry.

```
1  @service_registry.route('/<service_name>',
```

```
2           methods=['GET', 'PUT', 'DELETE'])
3 def handle_specific_data(data_type, service_name):
4     obj_type = getattr(sys.modules[__name__], data_type)
5     data = obj_type.objects(service_name=service_name).first()
6     if not data: return jsonify({'error': 'data not found'})
7     if request.method == 'GET': return handle_get(data)
8     elif request.method == 'PUT': return handle_put(data)
9     elif request.method == 'DELETE': return handle_delete(data)
```

4.1 Funcionamento

4.1.1 MTS API

O serviço proposto por este trabalho, implementado na MTS API possui um *end-point* o qual é responsável por possibilitar a troca de informações entre dois interessados. Este serviço recebe uma requisição HTTP do tipo *POST*, a qual em seu corpo deve conter três informações: i) O endereço final pra onde as informações deve ser repassadas; ii) O método que deve ser utilizado para acessar aquele endereço; iii) E as informações que o cliente deseja enviar. Deve-se sempre atentar para o modelo de dados utilizado que é preestabelecido e deve ser seguido. Isto porque a API recebe as informações do corpo da requisição em formato JSON, logo a identificação de seu conteúdo é feito a partir de sua respectiva chave. Este modelo foi apresentado no Capítulo 3, no Código3.1.

Ao receber as informações, o método *postData* é invocado. Este método é responsável por identificar o endereço do cliente final, o método HTTP utilizado. Isto por que para enviar as informações a API faz uma nova requisição, e para utiliza o endereço que lhe é passado e também o método. Para fazer esta nova requisição, o método *makeRequest* é invocado, recebendo separadamente IP destino e dados que devem ser enviados como parâmetros. Ao fazer esta nova requisição a API recebe também a resposta dada pelo servidor e encaminha esta resposta até o cliente. Este fluxo pode ser observado na Figura 4.1.

Este serviço será exposto através de uma porta externa, chamada *NodePort* utilizando o serviço Kubernetes Kube-proxy. Neste caso, o Kube-proxy expõe uma porta associada ao serviço usando NAT [27]. Como é possível observar na Figura 4.1, espera-se que o cliente verifique a disponibilidade do serviço de interesse no SR e, caso disponível, receba todas as informações referentes ao mesmo. Sendo assim, o modelo de dados esperado pela MTS API contém o endereço IP completo, ou seja, contendo IP, porta e *end-point* de destino. Este modelo de dados pode ser observado abaixo:

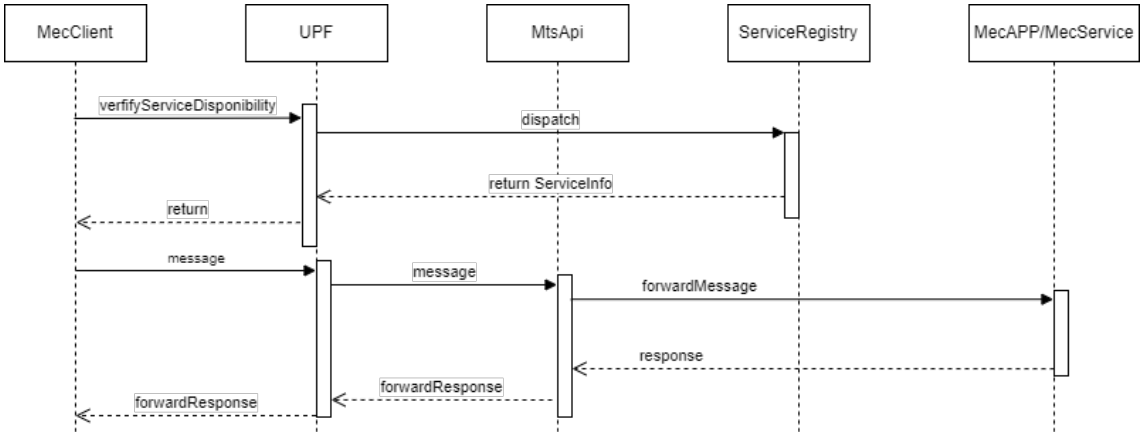

Figura 4.1: Funcionamento base da API proposta

4.1.2 Service Registry

O SR tem função receber informações sobre as aplicações e serviços disponíveis naquele servidor e possibilitar que aplicações clientes interessadas recebam essas informações para conseguir acessá-las ou saber endereçar seus dados. Para que esta função seja cumprida o SR utiliza banco de dados NoSQL MongoDB [34]

Para que uma aplicação MEC possa estar disponível para seus interessados é necessário que esta se registre no SR. O serviço de SR espera que a aplicação envie informações como seu endereço de IP, porta e seus *end-points*. É sempre esperado que a aplicação defina ao menos um *end-point*. Todo o fluxo de registro pode ser visto na Figura 4.2. Ao receber as informações o SR valida as informações recebidas e, caso todas as informações necessárias estejam presentes, as salva no banco dados. O modelo de dados esperado para registro no *Service Registry* foi definido e apresentado no Capítulo 3, no Código3.2.

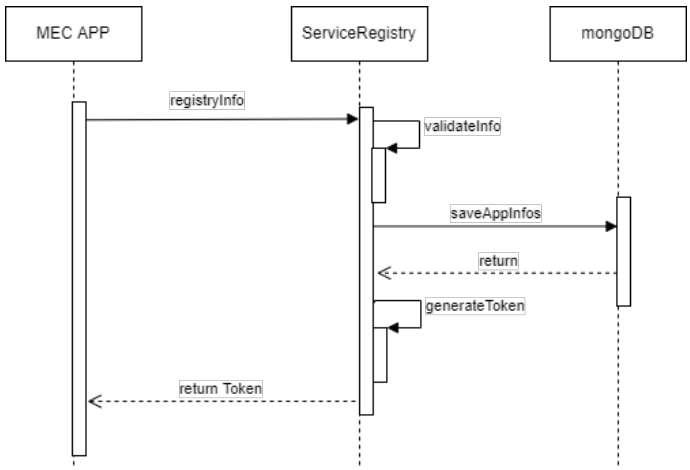

Figura 4.2: Registro de uma aplicação no *Service Registry*

O SR é um serviço que também estará exposto externamente, sendo assim, poderá ser acessado pelo cliente. O acesso do cliente se dará para verificação da existência do serviço de seu interesse, para isso, passará pra um *end-point* exclusivo para pesquisas e, caso exista aquele serviço registrado será retornado um JSON contendo todas as informações daquele serviço. Além das informações sobre a aplicação de interesse, também será enviado um *token* para autenticação na MTS API. Este token é gerado utilizando a biblioteca JWT do Python e para isso utiliza uma chave definida diretamente no código. Esta chave pode ser alterada de acordo com a necessidade, utilizando outras chaves. O fluxo de comunicação pode ser observado na Figura 4.1.

CAPÍTULO 5

Avaliação e Resultados

Este capítulo tem como objetivo apresentar a validação da proposta deste trabalho. Para isso, serão discutidos casos de usos que se beneficiam do MEC, o ambiente de validação e por fim os resultados. Os resultados da validação serão comparados com características de aplicações que se beneficiam do MEC com o objetivo de discutir a viabilidade da solução.

5.1 Casos de uso candidatos

Para avaliação dos resultados do serviço proposto por este trabalho serão utilizados Indicadores Chave de Performance (*Key Performance Indicators* - KPIs) descritos na literatura. Para isso, o cenário de avaliação incluirá cenários de *e-health* e também de IoT.

Para cenários da área da saúde, serão observados casos de cirurgia remota, consulta remota, suporte de paramédicos, eventos de saúde críticos, sinais vitais e *tags* de localização. Para cada um destes cenários, são descritos na literatura KPIs mínimos que são necessários para a execução da aplicação. Como base, utilizaremos KPIs apresentados em Chang *et al.* (2021) e Wijethilaka e Liyanage (2021) [51, 53]. Já em cenários de internet das coisas, são analisados características para cenários de casas inteligentes, IoT vestível (*Wearable* IoT - WIoT) e também cenários em fazendas, e para isso serão utilizadas características descritas em Porambage *et al.* (2021) [40].

Em todos estes casos, são apresentados diversos KPIs, como latência, número de dispositivos, velocidade de *downlink* e *uplink*. Porém, serão tomados como base para comparação apenas a latência. Eventualmente, poderão ser utilizadas outras informações de acordo com a necessidade. Esse KPIs são apresentados na Tabela 5.1.

5.2 Descrição do ambiente

Para que seja executado o que se foi proposto, projetou-se um ambiente para validação da proposta. Buscou-se um *framework* MEC e também um 5GC para que fosse

Tabela 5.1: KPIs utilizandos para comparação.

Aplicação	**Latência**
Smart Home	1ms - 1000s
Cirurgia Remota	$\leq 200ms$
Consulta Remota	1ms - 100s
Suporte de Paramédicos	200ms
Evento Crítico de Saúde	200ms
Fazendas Inteligentes	Várias Horas
Patchs de Sinais Vitais	1000ms
Tags Localizáveis	1000ms

possível avaliar e validar o produto final deste trabalho. Ao buscar uma plataforma MEC aderente, foram encontradas soluções, contudo, estas não ofereciam o devido suporte para o que foi planejado. Sendo assim, optou-se pela utilização de solução em contêineres, utilizando como solução de gerenciamento o Kubernetes. O Kubernetes é uma plataforma de código aberto, que executa o gerenciamento de cargas de trabalho e serviços presentes em contêineres [27].

Como solução para 5GC, escolheu-se o free5GC. Sua documentação indica uma execução em *bare-metal*, a qual após avaliação, decidiu-se por mudança na abordagem. Esta mudança se deu pela complexidade para execução do *core* como um todo. Para este trabalho, decidiu-se pela conteinerização das funções de rede do *core* free5GC. Sendo assim, todo o *core* passa a ser executado em contêineres Docker. Docker é uma plataforma aberta que permite a execução de aplicativos empacotados em contêineres, nos quais possuem tudo o que é necessário para sua execução, tornando-os levemente isolados do restante [12]. Contudo para execução do *core*, mesmo conteinerizado, há a necessidade que o módulo GTPU esteja instalado. Este módulo é utilizado pelo UPF e só pode ser instalado em ambiente Linux com *kernel* versão 5.4.

Após os estudos sobre as ferramentas, foram executados os *deployments* das mesmas. Todos os *deployments* feitos neste trabalho foram executados em máquinas virtuais (*Virtual Machine* - VM) hospedadas em um servidor dedicado. Este servidor está hospedado no Instituto de Informática da Universidade Federal de Goiás. A escolha por VMs se deu pela possibilidade de acesso remoto, visto que este trabalho foi desenvolvido em tempos de pandemia e, também, pela maior disponibilidade de recursos do que máquinas físicas.

Cada ferramenta possui em suas máquinas características de hardware únicas já definidas em sua documentação. A Figura 5.1 apresenta a estrutura das VMs utilizadas. Sendo 3 máquinas, a VM 1 será utilizada para *deploy* do free5GC. Esta máquina possui 8GB de memória RAM, 100GB de disco, 4vCPUs e sistema operacional Ubuntu *Live Server* 18.04 LTS. As máquinas 2 e 3 possuem 8 GB de memória RAM, 120GB de disco, 4

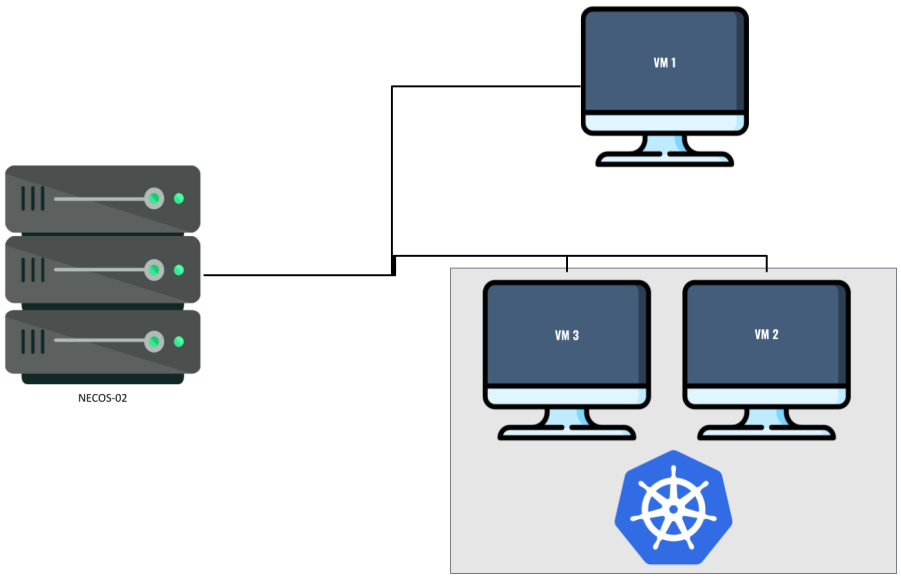

Figura 5.1: Estrutura de VMs utilizadas na validação.

vCPUs, com sistema operacional Ubuntu *Live Server* 18.04 LTS. Além disso, as máquinas 2 e 3 formam um cluster Kubernetes. O cluster Kubernetes neste cenário será utilizado para simular uma plataforma MEC, pois, entre outras razões, é possível associar as ideias de gerenciamento de ciclo de vida, manutenção de aplicação e alta disponibilidade de serviços presentes no MEC.

Com as máquinas devidamente configuradas, definiu-se a arquitetura arquitetura do free5GC utilizada para validação deste trabalho. Esta arquitetura utiliza uma arquitetura com 3 UPFs e é apresentada na Figura 5.3. Os três UPFs utilizados são *Branching* UPF (B-UPF), MEC UPF (M-UPF) e Internet UPF (I-UPF). É possível observar na Figura 5.3 que o B-UPF não está associado a nenhum *slice*, isto por que ele não possuirá interfaces do tipo $N6$, tendo apenas $N3$, $N4$ e $N9$, logo, sua função será apenas intermediar a conexão entre UE e DN.

Para compor o ambiente de validação da proposta, foi desenvolvida uma aplicação, simulando uma aplicação MEC. Esta aplicação tem por objetivo possibilitar simular um ambiente fim a fim, ou seja, tráfego partindo do UE até a aplicação MEC passando pela API proposta. Também será desenvolvido um *Service Registry* (SR) o qual irá permitir que as aplicações que desejem usar a MTS API recebam um *token* de registro para utilização dos serviços por ela ofertados.

Após o desenvolvimento de todas estas aplicações e serviços, passa-se para o *deployment* em ambiente de produção. Para execução em ambiente de produção é necessário que o servidor web que execute as aplicações mude. Sendo assim, em função da utilização da linguagem Python, dentre as listas de servidores de produção disponíveis, escolhemos o Waitress. O servidor Waitress é um servidor de produção para Python que

permite executar aplicações em Unix e Windows sob Python 3.7+, suporta HTTP/1.0 e HTTP/1.1 [50]. Esta escolha não possui motivo específico sendo possível a execuçãoutilizando qualquer servidor de produção para linguagem Python.

A execução dos *deployments* das aplicações e serviços que estarão no cluster, devem ser feitas de formar coordenada sendo o *service-registry* executado primeiro. Isso se dá devido a necessidade de inserção manual do endereço IP e porta internos do cluster, que estarão associados ao *service-registry*, nas aplicações que se registrarão neste serviço. A parti daí, o *deploy* de qualquer aplicação que se registrará no SR poderá ser executado. Sendo assim, executamos o deploy das aplicações sem ordem específica utlizando endereço IP, porta e end-point de registro.

Com isso, é possível passar para a fase de validação. A fase de validação é divida em 3. A primeira fase consiste na validação de todo o ambiente de testes, garantindo que free5GC, MTS-API e aplicação de testes estão sendo executados de forma correta. A segunda fase se dará no registro dos serviços fornecido pela API e pela aplicação de testes no *Service Registry*, de forma que suas informações estejam acessível pelas aplicações interessadas. A terceira e última fase, é a execução dos testes, colheita e avaliação de resultados.

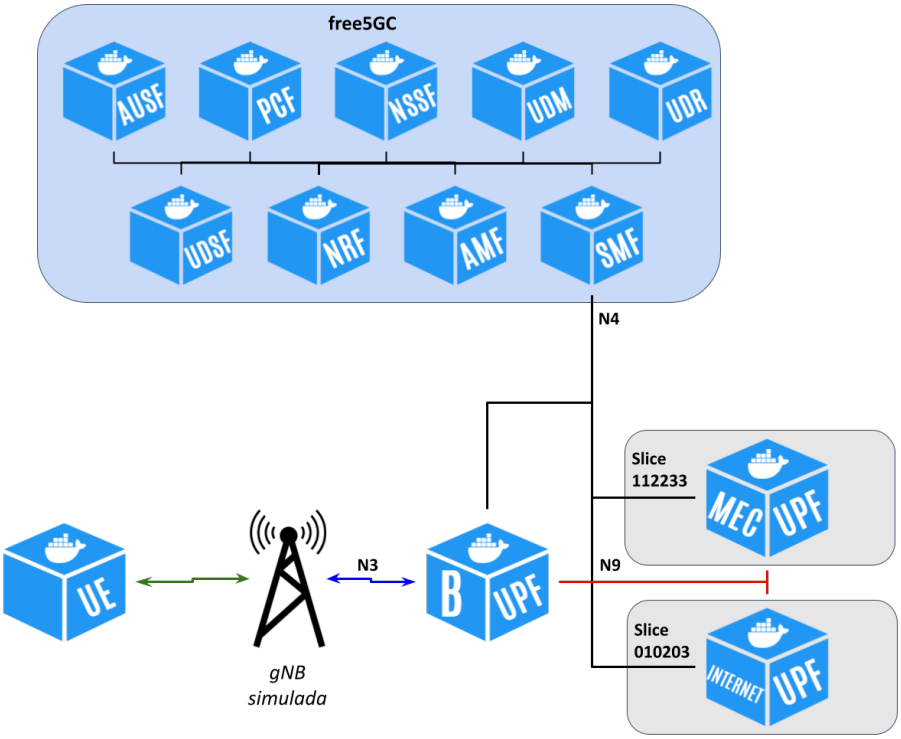

Figura 5.2: Arquitetura do *core* free5GC.

5.3 Métricas de avaliação

5.3.1 Validação da API

Para compor a validação será utilizada uma aplicação simples que faz cálculo de Índice de Massa Corporal (IMC). Esta aplicação recebe 2 informações sendo elas altura e peso, retornando o resultado e dizendo qual o resultado. Esta aplicação recebe requisições do tipo *POST*. É importante salientar que esta requisição poderia ser feita a qualquer aplicação que estivesse sendo executada no cluster e registrada no *service-registry*. Sendo assim, o uso desta aplicação se mantém apenas no objetivo de validar a API proposta.

Para avaliarmos a capacidade da API, será utilizada a ferramenta Locust [30]. A ferramenta proporciona a execução de requisições HTTP a um determinado IP e *end-point* daquele IP. Além disso, também possibilita simular número variados de usuários fazendo requisições simultaneamente para tal.

Serão executados 3 cenários. Cada cenário terá uma quantidade definida de usuários, sendo elas, 1, 10, 50, 100 de maneira que seja simulado a capacidade da API de atender a requisições concorrentes. Em comum, em cada cenário será observado resultados ao final de um período de tempo de execução continuada, ou seja, 5, 10, 30 e 60 segundos. O objetivo final é observar o tempo de resposta da API com as variações já relatadas acima. Para todos os conjuntos de testes será colhido o tempo médio de resposta e também a quantidade média de requisições.

Além disso, serão avaliados também a execução do servidor de produção com 4 e 8 *threads*. Com estes resultados será possível observar se há ou não melhora de desempenho e, a partir disto, discutir a viabilidade da API em cenários maiores.

Com o objetivo de se ter uma maior confiabilidade, os mesmos conjuntos de testes serão executados 10 vezes. Sendo assim, o tempo final será dado pelo tempo de resposta médio das 10 execuções de cada conjunto. Para que os resultados possam ser devidamente avaliados, serão calculados os intervalos de confiança, sendo este de 95%. O cálculo de intervalo de confiança se dá pela equação:

$$\overline{x} \pm Z \cdot \frac{\sigma}{\sqrt{n}}$$

Nesta equação $\overline{x}$ é a média dos tempos de resposta das 10 execuções. Z é o valor da distribuição normal, que para 95% é 1.96. σ é o desvio padrão dos tempos de resposta e n é o tamanho da amostra. O intervalo superior é dado pelo resultado da soma e o inferior da subtração.

5.3.2 Validação Fim a Fim

A validação fim a fim utiliza o cenário apresentado na Figura 5.3. A representação apresentada nesta figura é a parte interna das VMs, ou seja, o que está sendo executado em cada uma delas. Destacado em verde, o núcleo da rede 5G (free5GC) e UE virtual utilizado para testes, ambos estão sendo executados na VM 1 apresentada na Seção 5.2. Destacado em vermelho claro, sendo executado no cluster temos a MTS-API, o *Service Registry* e também uma aplicação MEC simples apenas para utilização nos testes. Estas aplicações e serviços estão sendo executadas nas VMs 1 e 2.

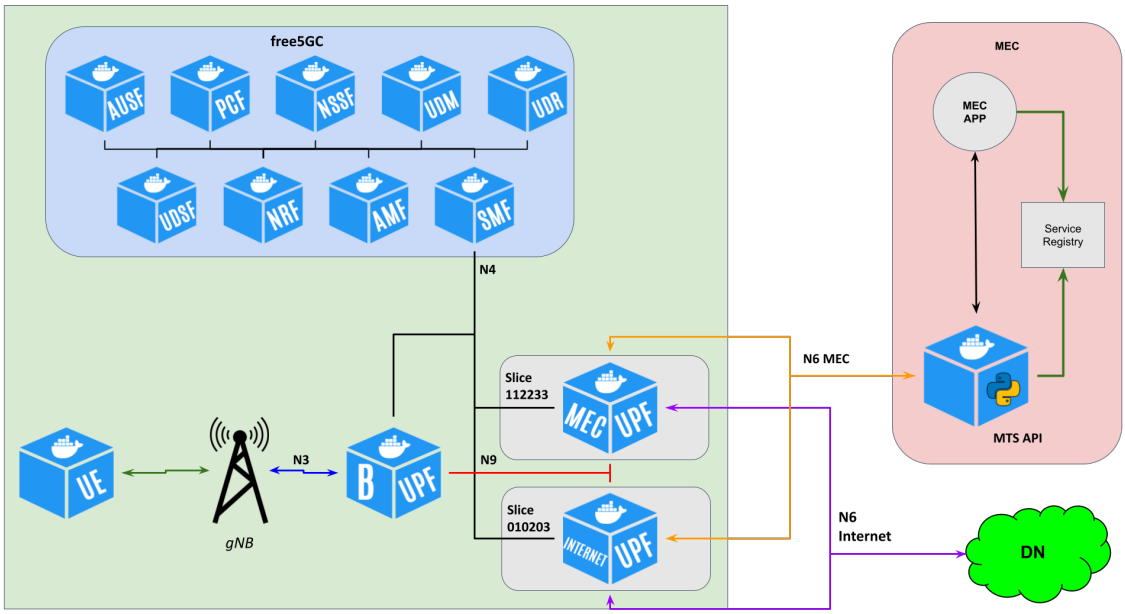

Figura 5.3: Arquitetura geral de testes.

Ao observar o cenário apresentado na Figura 5.3, M-UPF e I-UPF estão em *slices* de rede diferentes, sendo assim, quando o UE se conectar a rede 5G e seu *slice* for definido, todo o tráfego de rede passará por aquele UPF. As redes lógicas criadas podem ser utilizadas para personalização de soluções para cenários e serviços distintos [44]. Logo, neste cenário os *slices* foram divididos de forma a permitir melhor compreensão, ou seja, quando o UE desejar se comunicar com a plataforma MEC, utilizará o *slice* 112233 e consequentemente M-UPF, e quando desejar utilizar apenas a internet utilizará o *slice* 010203. Salienta-se que esta separação existe apenas para testes, ambos UPFs podem se conectar à *N6* MEC e *N6* Internet.

Com o UE virtual devidamente conectado à rede 5G, deve-se observar que a interface de rede conectada ao UPF se chama *uetun*1, logo, os testes que partem do UE devem sempre utilizar esta interface. Sendo assim, para este teste será feita uma requisição

utilizando a biblioteca Curl do UE até a aplicação MEC devida. O conteúdo da mensagem será o mesmo utilizado nos testes anteriores.

Este teste tem por objetivo mostrar que existe a possibilidade de comunicação fim a fim, sendo assim, serão observados os seguintes aspectos: i) A resposta da requisição, se foi ou não a esperada; ii) o tempo de resposta até da verificação até o primeiro envio, e; iii) o tempo de resposta das requisições; iv) Os pacotes capturados pelo Wireshark monitorando a interface do UPF. Para o itens 1 e 4 serão feitas apenas 1 requisição entendendo que as informações observadas não serão alteradas. Porém, as demais serão consideradas a média de tempo para 30 execuções. Para os tempos que serão discutidos também serão calculados os intervalos de confiança.

5.4 Comunicação Fim a Fim

A Validação Fim a Fim tem por objetivo demonstrar se houve comunicação entre UE e MEC-APP com tráfego passando pelo UPF e MTS-API. Dessa maneira, observaremos inicialmente a troca de mensagens entre UPF e MTS-API. A Figura 5.4 apresenta uma captura de imagem da janela da ferramenta WireShark. Nesta figura é possível identificar a mensagem que sai do UPF (IP 20.0.0.2) com destino à MTS-API (IP 10.16.10.26) na linha 4 e a resposta na linha 7.

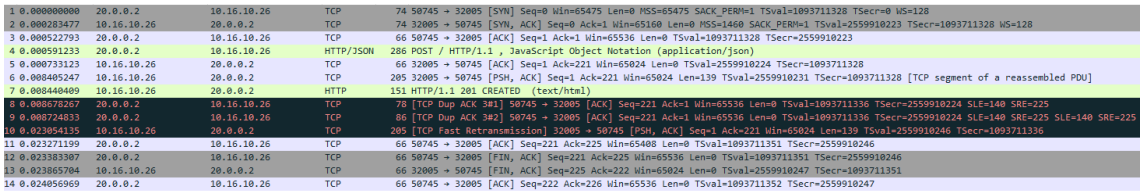

Figura 5.4: Troca de Mensagens capturadas usando WireShark na interface do M-UPF

Além disso, foi observado que o tempo de resposta médio para as requisições é de $31ms$, com um desvio padrão de 1.71. O resultado do desvio padrão demonstra que a variação do tempo de resposta das 30 execuções não foi grande, sendo próximos à média. Para verificar se há um serviço desejado e receber um *token*, o tempo de resposta no UE é de $26ms$, com um desvio padrão de 5.21 demonstrando uma variação maior do tempo de resposta. Se observado o tempo total do processo, da verificação da existência do serviço, envio da mensagem, até o cliente receber a primeira resposta leva $57ms$, com um desvio padrão de 5.63.

Estes resultados viabilizam a utilização desta solução em casos de uso de como *e-health*. Zhang *et al.* (2021) apresenta KPIs para alguns cenários no âmbito da saúde. Dessa forma, identifica-se que esta solução seria viável para cenários como suporte paramédico,

eventos críticos de saúde (não exemplificados no trabalho) [53]. Além disso, também poderia ser viável a sua aplicação em cenários que utilização geolocalização e *patch* de sinal vital para permitir o monitoramento a qualquer hora e lugar. Este cenário aceita tempo de resposta de até $1000ms$ [53].

5.5 Experimentação da API proposta

A experimentação da proposta leva em consideração as métricas descritas na Seção 5.3. Sendo assim, esta seção tem por objetivo observar os resultados coletados, e discuti-los. Esta avaliação levará em conta os casos de uso apresentados na Seção 5.1 e seus KPIs descritos na literatura. A variação da quantidade de usuários permitirá que seja observado qual seria o comportamento do serviço em cenários onde há a concorrência pelo mesmo recurso. Entende-se que este é uma cenário experimental, sendo que em cenários reais poderão existir números de usuários ainda maiores.

A Figura 5.5 apresenta um gráfico com o resultado do execução com apenas 1 usuário. No gráfico é possível observar que para 4 *threads* o tempo de resposta se manteve sempre em $6ms$, porém, com 8 *threads* obteve-se um tempo de resposta ligeiramente maior, sendo este de $7ms$, mas descendo para $6ms$ quando observado o cenário com 60 segundos de execução e, consequentemente, um número maior de requisições. Além disso, no cenário com 8 *threads* é possível observar que a margem de erro chegou a 0 em cenários com 30 segundos de execução. Esta é uma diferença significativa se comparado com a execução com 4 *threads*. O tempo de resposta obtido neste cenário se comparado à KPIs apresentados em Porambage *et al* (2018) se adéquam, por exemplo, a cenários de *Smart Home*, os quais aceitam latência de até 1000s, ou até de jogos no qual o a latência deve ser menor ou igual a $10ms$ [40].

O cenário com 10 usuários apresenta resultados com latência maiores se comparado ao cenário anterior. Isso porque, ao ser colocado 10 usuários requisitando o mesmo recurso de forma concorrente, há uma tendência de que o tempo de resposta aumente. Esta tendência se comprova ao comparamos os resultados do cenário com 1 usuário e este com 10 apresentados na Figura 5.6. Com 10 usuários, o tempo de resposta chegou a $58ms$ com 4 *threads* e $47ms$ com 8 *threads*. Mais uma vez, é possível observar que o cenário com maior número de requisições resultou em um intervalo de confiança menor do que em anteriores. Ao levar em consideração KPIs apresentados em Zhang *et al.* (2021) para cenários de *e-health*, observa-se que estes resultados viabilizam a utilização desta solução para aplicações de suporte paramédico, ou até eventos críticos de saúde os quais aceitam latência de até 200ms [53].

Com 50 e 100 usuários os resultados apresentaram tempos significativamente maiores se comparado aos anteriores. As Figuras 5.7 e 5.8, apresentam os resultados para

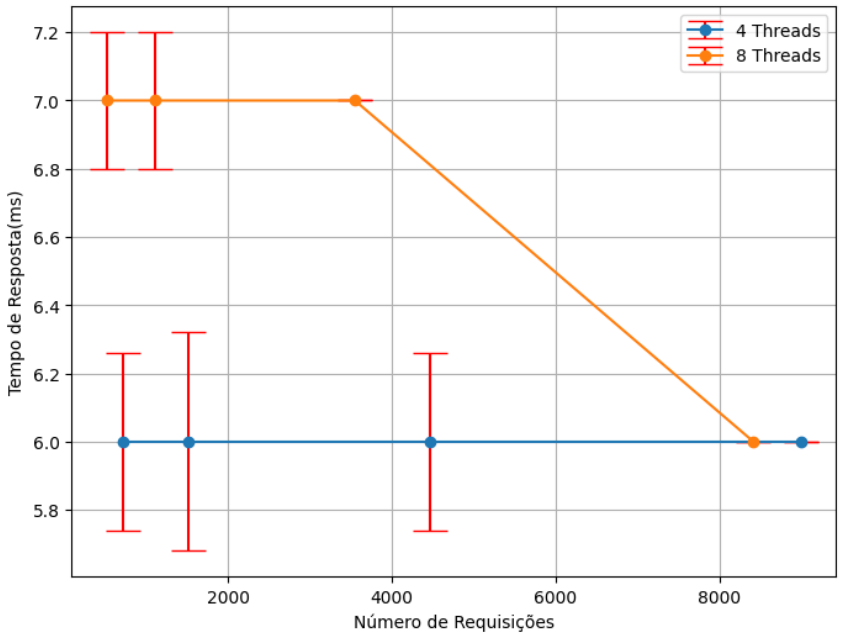

Figura 5.5: Desempenho com 1 Usuário.

50 e 100 usuários respectivamente. Estes gráficos retratam tempos médios de resposta extremamente altos, chegando a $520ms$ com 50 usuários e $1700ms$ com 100 usuários. Porém, ao aumentar o poder de processamento para 8 *threads*, observou-se um tempo de resposta mais baixo, chegando a $250ms$ e $590ms$ respectivamente. Em casos mais ajustados, ou seja, com intervalos de confiança menores, observa-se que o tempo de resposta foi de $270ms$ para 50 usuários e $720ms$ para 100 usuários.

Com os resultados para 50 usuários, é possível utilizar a solução principalmente em cenários de *Smart Farming* para transmissão de dados de um sensor ou até *tags* de localização, no qual a latência poderia chegar a $300ms$ e $1000ms$ respectivamente [51, 53]. Cenários com 100 usuários tornam-se cenários mais restritos em função do tempo de resposta apresentado. Mesmo assim, é possível identificar que cenários como *Smart Home* (até $1000s$), *tag* de localização ($1000ms$), consulta remota ($100s$) é plausível a utilização deste serviço [40, 53].

No geral, observa-se que a solução apresentou resultados esperados. Os tempos de resposta coletados demonstram que a solução é viável em alguns casos de uso de cenários distintos. Além disso, com a evolução das aplicações e também da rede, novos casos de uso pode surgir e novas demandas podem ser necessárias inclusive para os casos de uso utilizados como referência. Logo, a longo prazo torna-se necessário a revisão da viabilidade desta solução para novos cenários/necessidades.

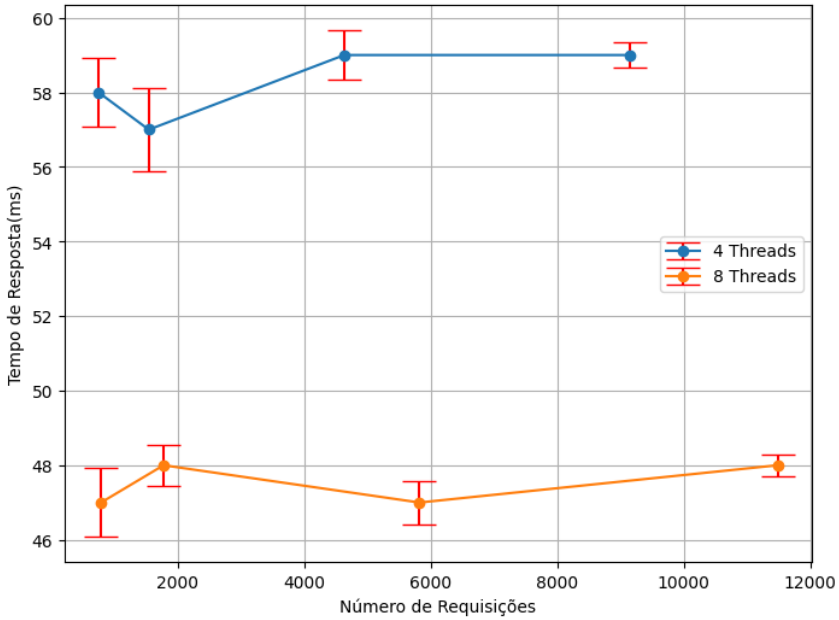

Figura 5.6: Desempenho com 10 Usuários.

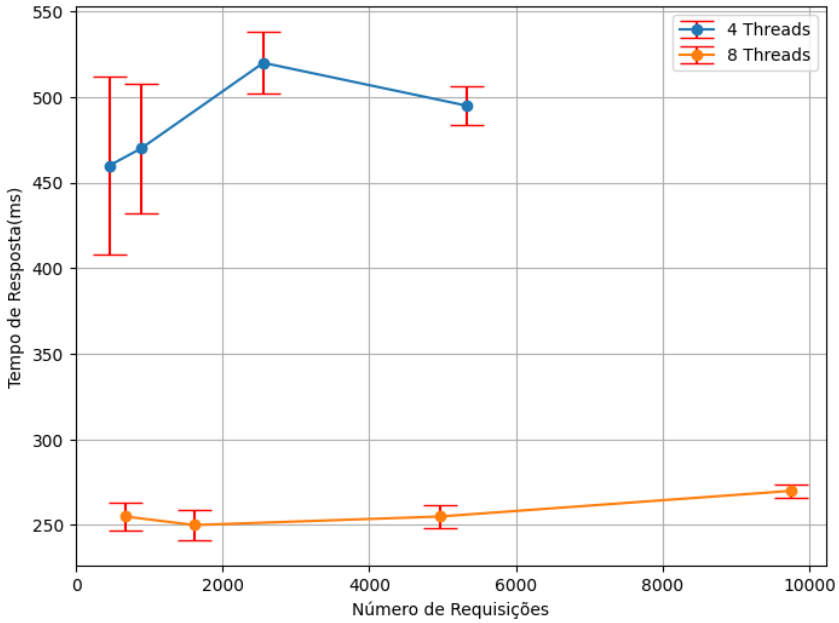

Figura 5.7: Desempenho com 50 Usuários.

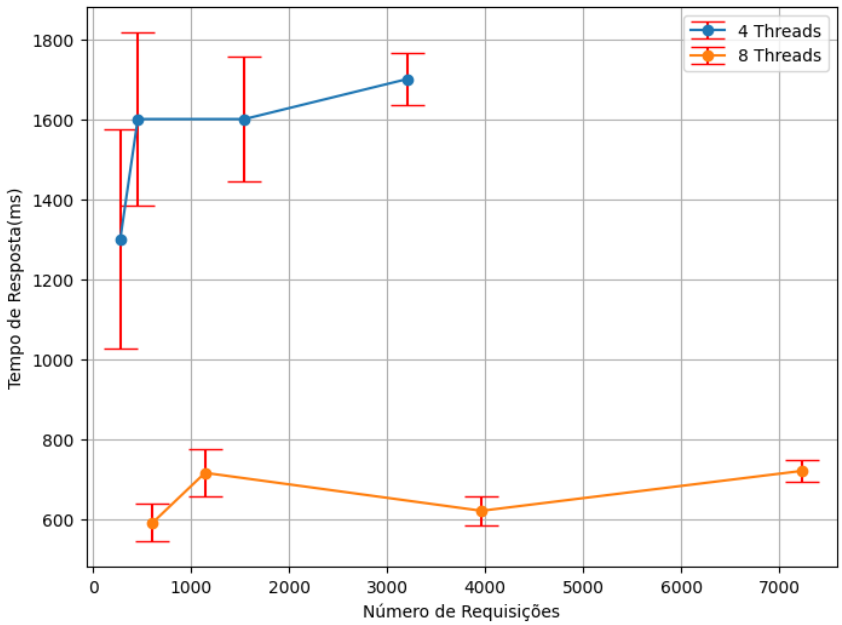

Figura 5.8: Desempenho com 100 Usuários.

CAPÍTULO 6

Conclusão

A partir dos trabalhos relacionados, é possível analisar que esta integração é ainda pouco discutida na comunidade acadêmica e cientifica. É observável que nos trabalhos existentes são propostos meios para esta integração sobretudo através de novas arquiteturas. Contudo existem lacunas deixadas por estes trabalhos no que tange principalmente à implementação e validação daquilo que é proposto.

Sendo assim, esta dissertação apresenta uma implementação de um serviço MEC que possibilita a integração entre redes 5G e MEC. Este serviço foi proposto como uma extensão a MTS API baseado na sua definição pela ETSI, entendendo que seu objetivo e o objetivo do serviço proposto são compatíveis. Esta proposta foi validada em ambiente *testbed* virtualizado, com a utilização de um núcleo de rede 5G *standalone*, com UE e gNB simulados, com um *cluster* Kubernetes assumindo a posição de plataforma MEC. Além disso, foram apresentados fundamentos das redes 5G e também do MEC além uma visão sobre o que se é apresentados nos documentos técnicos publicados pelos órgãos de padronização.

A partir dos resultados da validação, é possível afirmar que é possível obter comunicação entre UE e aplicação a partir do serviço proposto. Além disso avaliou-se também o desempenho desta API com base no tempo de resposta visto que MEC abrange principalmente aplicações tipo URLLC.

É notável que a discussão sobre as redes 5G cresceram nos últimos anos tanto no cenário acadêmico quanto comercial. Os seus benefícios, os quais já foram apresentados neste trabalho, são os grandes gatilhos para esses aumento. Sendo assim, entende-se que esta solução pode subsidiar outros trabalhos que estão por vir. Posteriormente esta solução pode ser comparada com soluções já discutidas na literatura gerando outros trabalhos e artigos. Também podem ser discutidos métodos para que esta solução seja melhorada, de forma que melhore o seu tempo de resposta e sejam alcançados de maneira constante o tempo de resposta mais baixo apresentado na validação.

Por fim, conclui-se que o objetivo geral deste trabalho foi alcançado e que é possível a integração 5G-MEC através da utilização de um serviço implantado como uma aplicação. Além disso, a solução especificada apresentou resultados satisfatórios nos

cenários especificados e avaliados por este trabalho.

Para trabalhos futuros pode ser implementada a possibilidade de envio de arquivos de mídia. Além disso, pode ser desenvolvido um componente extra pelo qual possibilite a comunicação em nível de CP do 5GC para inferir regras e requisitar informações sobre a rede. Também podem ser feitas validações para comparar a solução aqui especificada com soluções descritas nos trabalhos relacionados, de maneira que os parâmetros de avaliação sejam equiparados. Além disso, validações como cenários com maiores níveis de stress em cenários concorrentes podem ser importantes para avaliar os resultados. Por fim, a implantação deste serviço em uma plataforma MEC real, para possibilitar a avaliação de seu funcionamento em ambientes mais próximos aos reais.

Referências Bibliográficas

[1] 3GPP. **3rd generation partnership project (3gpp).** https://www.https://www.3gpp.org/, 2022. Acessado em: 26-09-2022.

[2] 3RD GENERATION PARTNERSHIP (3GPP). **Tr21.915 v15.9.0, 3rd generation partnership project**. https://www.3gpp.org/ftp/Specs/archive/21_series/21.915/, 2018. Acessado em: 06-05-2021.

[3] ABBAS, N.; ZHANG, Y.; TAHERKORDI, A.; SKEIE, T. **Mobile edge computing: A survey**. *IEEE Internet of Things Journal*, 5(1):450–465, 2017.

[4] AGÊNCIA NACIONAL DE TELECOMUNICAÇÕES. **Painel de dados**. https://informacoes.anatel.gov.br/paineis/acessos/telefonia-movel, 2021. Acessado em 07-05-2021.

[5] AL-FALAHY, N.; ALANI, O. Y. **Technologies for 5g networks: Challenges and opportunities**. *IT Professional*, 19(1):12–20, 2017.

[6] AMGOUNE, H.; MAZRI, T. **Comparison between different 5g architectures for a better integration of these services and proposal of an improved architecture**. In: *Proceedings of the 4th International Conference on Smart City Applications*, p. 1–7, 2019.

[7] BONOMI, F.; MILITO, R.; ZHU, J.; ADDEPALLI, S. **Fog computing and its role in the internet of things**. In: *Proceedings of the first edition of the MCC workshop on Mobile cloud computing*, p. 13–16, 2012.

[8] CONFORTO, E. C.; AMARAL, D. C.; SILVA, S. D. **Roteiro para revisão bibliográfica sistemática: aplicação no desenvolvimento de produtos e gerenciamento de projetos**. *Trabalho apresentado*, 8, 2011.

[9] CUERVO, E.; BALASUBRAMANIAN, A.; CHO, D.-K.; WOLMAN, A.; SAROIU, S.; CHANDRA, R.; BAHL, P. **Maui: making smartphones last longer with code offload**. In: *Proceedings of the 8th international conference on Mobile systems, applications, and services*, p. 49–62, 2010.

[10] CUNHA, K.; XAVIER, R.; MOREIRA, W.; FREITAS, L.; OLIVEIRA-JR, A. **Uma abordagem sobre a integração da computação de borda móvel e a rede 5g para internet das coisas na agricultura 4.0**. In: *Anais da IX Escola Regional de Informática de Goiás*, p. 118–131, Porto Alegre, RS, Brasil, 2021. SBC.

[11] DINH, H. T.; LEE, C.; NIYATO, D.; WANG, P. **A survey of mobile cloud computing: architecture, applications, and approaches**. *Wireless communications and mobile computing*, 13(18):1587–1611, 2013.

[12] DOCKER. **Docker**. `https://docs.docker.com/get-started/overview/`, 2022. (Accessed on 14/09/2022).

[13] ERICSSON. **Ericsson mobility report**. 2020.

[14] ETSI, A. **Industry specification group (isg) on multi-access edge computing (mec)**. `https://www.etsi.org/committee/1425-mec`, 2021. Acessado em: 06-05-2021.

[15] ETSI, I. **Multi-access edge computing (mec); framework and reference architecture**. Technical report, ETSI, 2020.

[16] ETSI, I. **Multi-access edge computing (mec); mec 5g integration**. Technical report, 2020, 2020.

[17] ETSI, I. **Multi-access edge computing (mec); traffic management apis**. Technical report, 2020, 2020.

[18] FILALI, A.; ABOUAOMAR, A.; CHERKAOUI, S.; KOBBANE, A.; GUIZANI, M. **Multi-access edge computing: A survey**. *IEEE Access*, 8:197017–197046, 2020.

[19] FOUKAS, X.; PATOUNAS, G.; ELMOKASHFI, A.; MARINA, M. K. **Network slicing in 5g: Survey and challenges**. *IEEE Communications Magazine*, 55(5):94–100, 2017.

[20] GEEKHUNTER. **Flask: o que é e como codar com esse micro framework python**. `https://blog.geekhunter.com.br/flask-framework-python/`, 2020. Acessado em: 21-02-2020.

[21] GREENBERG, A.; HAMILTON, J.; MALTZ, D. A.; PATEL, P. **The cost of a cloud: research problems in data center networks**, 2008.

[22] IWAI, T.; NAKAO, A. **Demystifying myths of mec: rethinking and exploring benefits of multi-access/mobile edge computing**. In: *2018 IEEE 7th International Conference on Cloud Networking (CloudNet)*, p. 1–4. IEEE, 2018.

[23] KAUR, H. **A survey on 5g standardization for edge computing and internet of things**. In: *Proc. 15th Adv. Int. Conf. Telecommun.*, p. 21–26, 2019.

[24] KAZMI, S. A.; KHAN, L. U.; TRAN, N. H.; HONG, C. S. **Network slicing for 5G and beyond networks**, volume 1. Springer, 2019.

[25] KEKKI, S.; FEATHERSTONE, W.; FANG, Y.; KUURE, P.; LI, A.; RANJAN, A.; PURKAYASTHA, D.; JIANGPING, F.; FRYDMAN, D.; VERIN, G.; OTHERS. **Mec in 5g networks**. *ETSI white paper*, 28:1–28, 2018.

[26] KSENTINI, A.; FRANGOUDIS, P. A. **Toward slicing-enabled multi-access edge computing in 5g**. *IEEE Network*, 34(2):99–105, 2020.

[27] KUBERNETES. **Kubernetes**. `https://kubernetes.io/pt-br/docs/`, 2021. (Accessed on 08/29/2022).

[28] KUKLIŃSKI, S.; TOMASZEWSKI, L. **Dasmo: A scalable approach to network slices management and orchestration**. In: *NOMS 2018-2018 IEEE/IFIP Network Operations and Management Symposium*, p. 1–6. IEEE, 2018.

[29] KUKLIŃSKI, S.; TOMASZEWSKI, L.; KOŁAKOWSKI, R. **On o-ran, mec, son and network slicing integration**. In: *2020 IEEE Globecom Workshops (GC Wkshps*, p. 1–6. IEEE, 2020.

[30] LOCUST. **Locust - a modern load testing framework**. `https://locust.io/`, 2022. (Accessed on 09/02/2022).

[31] MA, H.; LI, S.; ZHANG, E.; LV, Z.; HU, J.; WEI, X. **Cooperative autonomous driving oriented mec-aided 5g-v2x: Prototype system design, field tests and ai-based optimization tools**. *IEEE Access*, 8:54288–54302, 2020.

[32] MAMUSHIANE, L.; DLAMINI, S. **Leveraging sdn/nfv as key stepping stones to the 5g era in emerging markets**. In: *2017 Global Wireless Summit (GWS)*, p. 23–27. IEEE, 2017.

[33] MARTINS, J. S. **A (r) evolução das redes sem fio**. 2001.

[34] MONGODB. **Mongodb**. `https://www.mongodb.com/pt-br`, 2022. (Accessed on 08/29/2022).

[35] NATIONAL CHIAO TUNG UNIVERSITY. **free5gc**. `https://www.free5gc.org/`, 2021. Acessado em 01-06-2021.

[36] NETWORK, E. U. T. R. A. **3rd generation partnership project; technical specification group services and system aspects; general packet radio service (gprs) enhancements for evolved universal terrestrial radio access network (e-utran) access**. *EUTRA Network*, 2011.

[37] NIKOLOFSKI, D. R. F. **A quarta geração das redes sem fio: benefícios e evolução**. 2011.

[38] PARSIFAL. **About parsifal**. `https://parsif.al/about/`, 2021. Acessado em 22-05-2021.

[39] PHAM, Q.-V.; FANG, F.; HA, V. N.; PIRAN, M. J.; LE, M.; LE, L. B.; HWANG, W.-J.; DING, Z. **A survey of multi-access edge computing in 5g and beyond: Fundamentals, technology integration, and state-of-the-art**. *IEEE Access*, 8:116974–117017, 2020.

[40] PORAMBAGE, P.; OKWUIBE, J.; LIYANAGE, M.; YLIANTTILA, M.; TALEB, T. **Survey on multi-access edge computing for internet of things realization**. *IEEE Communications Surveys & Tutorials*, 20(4):2961–2991, 2018.

[41] ROMMER, S.; HEDMAN, P.; OLSSON, M.; FRID, L.; SULTANA, S.; MULLIGAN, C. **5G Core Networks: Powering Digitalization**. Elsevier, 2019.

[42] SATYANARAYANAN, M.; BAHL, P.; CACERES, R.; DAVIES, N. **The case for vm-based cloudlets in mobile computing**. *IEEE pervasive Computing*, 8(4):14–23, 2009.

[43] SGANZERLA, A. R.; HENRIQUE, L. R.; DE, A. **Estudo comparativo entre as redes 3g e 4g**, 2010.

[44] SHAH, S. D. A.; GREGORY, M. A.; LI, S. **Cloud-native network slicing using software defined networking based multi-access edge computing: A survey**. *IEEE Access*, 9:10903–10924, 2021.

[45] SHI, W.; CAO, J.; ZHANG, Q.; LI, Y.; XU, L. **Edge computing: Vision and challenges**. *IEEE internet of things journal*, 3(5):637–646, 2016.

[46] SILVA, J. P. L.; NERY, S. W. L.; SILVA, R.; OLIVEIRA-JR, A. C.; CARDOSO, K. V.; BOTH, C. B. **Entendendo o núcleo 5g na prática, através de uma implementação de código aberto**. In: *SBRT 2020: Livro de Minicursos*, chapter 1. Instituto Federal de Ensino, Ciência e Tecnologia da Paraíba – IFPB, Florianópolis-SC, 2021.

[47] TALEB, T.; AFOLABI, I.; SAMDANIS, K.; YOUSAF, F. Z. **On multi-domain network slicing orchestration architecture and federated resource control**. *IEEE Network*, 33(5):242–252, 2019.

[48] TANAKA, H.; YOSHIDA, M.; MORI, K.; TAKAHASHI, N. **Multi-access edge computing: A survey**. *Journal of Information Processing*, 26:87–97, 2018.

[49] TOMASZEWSKI, L.; KUKLIŃSKI, S.; KOŁAKOWSKI, R. **A new approach to 5g and mec integration**. In: *IFIP International Conference on Artificial Intelligence Applications and Innovations*, p. 15–24. Springer, 2020.

[50] WAITRESS. **Waitress wsgi server**. `https://pypi.org/project/waitress/`, 2022. (Accessed on 18/09/2022).

[51] WIJETHILAKA, S.; LIYANAGE, M. **Survey on network slicing for internet of things realization in 5g networks**. *IEEE Communications Surveys & Tutorials*, 23(2):957–994, 2021.

[52] XAVIER, R.; OLIVEIRA-JR, A.; FREITAS, L. **Uma discussao da integracao entre mec e 6g**. In: *Anais do II Workshop de Redes 6G*, p. 13–18, Porto Alegre, RS, Brasil, 2022. SBC.

[53] ZHANG, D.; RODRIGUES, J. J.; ZHAI, Y.; SATO, T. **Design and implementation of 5g e-health systems: Technologies, use cases, and future challenges**. *IEEE Communications Magazine*, 59(9):80–85, 2021.