Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Análise do Tráfego de Controle de Funções Virtualizadas em Redes 5G

Éden de Lucas Castelar Vale Medeiros

Monografia apresentada como requisito parcial
para conclusão do Bacharelado em Ciência da Computação

Orientadora
Prof.a Dr.a Priscila América Solis M. Barreto

Brasília
2021

Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Análise do Tráfego de Controle de Funções Virtualizadas em Redes 5G

Éden de Lucas Castelar Vale Medeiros

Monografia apresentada como requisito parcial
para conclusão do Bacharelado em Ciência da Computação

Prof.a Dr.a Priscila América Solis M. Barreto (Orientadora)
CIC/UnB

Prof. Dr. Eduardo Adilo Pelinson Alchieri
CIC/UnB

Prof. Dr. João José Costa Gondim
CIC/UnB

Prof. Dr. Marcelo Grandi Mandelli
Coordenador do Bacharelado em Ciência da Computação

Brasília, 28 de maio de 2021

Dedicatória

Este trabalho é dedicado a minha família, por sempre terem me dado apoio e motivação na busca dos meus objetivos. Aos amigos que tive o prazer de conhecer em minha jornada acadêmica e que os desejo levar por toda a vida. Por fim, a minha amada namorada Cláudia A. Guimarães que sempre esteve ao meu lado nos momentos difíceis e sempre me motivou e me fez melhorar como pessoa.

Agradecimentos

Agradeço a minha orientadora Priscila Solis por ter me dado a oportunidade de realizar esse projeto e por me ajudar em toda essa trajetória e criação do tema. Ao mestrando Cristoffer Leite por compartilhar sua pesquisa e me apoiar na formulação da monografia. Ao Rafael Amaral pelos ensinamentos passados e ao projeto 5G-Range que tive o prazer de participar.

O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES), por meio do Acesso ao Portal de Periódicos.

Resumo

A virtualização de funções de rede é um elemento da arquitetura especificada pela ETSI para as redes 5G, em que se tem a substituição de dispositivos de *hardware* de alto custo por dispositivos baseados em *software*. Além disso, esses serviços virtualizados podem ser executados em servidores genéricos ao invés de hardware proprietário e especializado.

Nesse trabalho é apresentada a construção de um ambiente virtualizado que utiliza um conjunto de ferramentas disponíveis para a implementação das funcionalidades de virtualização na arquitetura 5G. Esse ambiente foi construído em conformidade com a documentação e especificações da ETSI. Posteriormente, esse ambiente foi validado mediante a comparação de resultados com um cenário VoIP desenvolvido em outro trabalho. Assim também, neste trabalho é apresentada uma análise detalhada do tráfego de controle na infraestrutura virtualizada. A análise também inclui a compreensão de quanto é afetado o volume de tráfego de controle ao adicionar nós de computação ou novas funções de rede. Os resultados obtidos mostram que o tráfego de controle nessa arquitetura é plausível de melhorias e otimização para futuros cenários e aplicações em 5G.

Palavras-chave: 5G, OpenStack, OSM, SBA, SDN, NFV, 5GC, computação em nuvem

Abstract

The virtualization of network functions is an element of the architecture specified by ETSI for 5G networks, in which there is the replacement of high-cost hardware devices with software-based devices. In addition, these virtualized services can be run on generic servers instead of proprietary and specialized hardware. In this work, the construction of a virtualized environment that uses a set of tools available for the implementation of virtualization functionalities in the 5G architecture is presented. This environment was built in accordance with ETSI documentation and specifications. Subsequently, this environment was validated by comparing the results with a VoIP scenario developed in another work. Also, this work presents a detailed analysis of the control traffic in the virtualized infrastructure. The analysis also includes an understanding of how much control traffic volume is affected when adding compute nodes or new network functions. The results obtained shows that the control traffic in this architecture is plausible for improvements and optimization for future scenarios and applications in 5G.

Keywords: 5G, OpenStack, OSM, SBA, SDN, NFV, 5GC, cloud computing

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas e Siglas

3GPP 3rd Generation Partnership Project.

4G Quarta geração de telefonia móvel.

5G Quinta geração de telefonia móvel.

5GC 5G Core Network.

AMF Access and Mobility Management Function.

AMQP Advanced Message Queuing Protocol.

API Application Programming Interface.

AUSF Authentication Server Function.

CAPEX Capital Expenditure.

COTS Commercial off-the-shelf.

CP Connection Point.

DHCP Dynamic Host Configuration Protocol.

DNS Domain Name System.

EM Element Management.

eMBB Enhanced Mobile Broadband.

ETSI European Telecommunications Standards Institute.

HTTP Hypertext Transfer Protocol.

IaaS Infrastructure as a Service.

IoT Internet of Things.

IP Internet Protocol.

ISG Industry Specification Groups.

ISP Internet Service Provider.

KVM Kernel-based Virtual Machine.

LTE Long-Term Evolution.

LXC Linux Containers.

MANO Management and Orchestration.

MIMO Multiple Input Multiple Output.

mMTC Massive Machine Type Communication.

NaaS Network as a Service.

NAT Network Address Translation.

NEF Network Exposure Function.

NF Network Function.

NFV Network Function Virtualization.

NFVI NFV Infrastructure.

NFVO NFV Orchestrator.

NR New Radio.

NRF Network Repository Function.

NSA Non-Standalone.

NSD Network Service Descriptor.

NSSF Network Slice Selection Function.

OPEX Operational Expenditure.

OSM Open Source MANO.

PaaS Plataform as a Service.

PCF Policy Control Function.

QEMU Quick Emulator.

QoS Quality of Service.

RAM Random-Access Memory.

REST Representational State Transfer.

RPC Remote Procedure Call.

RPi Raspberry Pi.

SA Standalone.

SaaS Software as a Service.

SBA Service-Based Architecture.

SBI Service-Based Interface.

SDN Software-Defined Network.

SIP Session Initiation Protocol.

SMF Session Management Function.

SOA Service-Oriented Architecture.

SSH Secure Shell.

UDM Unified Data Management.

UPF User Plane Function.

URLLC Ultra-Reliable Low Latency Communication.

VDU Virtual Deployment Unit.

VIM Virtualised Infrastructure Manager.

VL Virtual Link.

VM Virtual Machine.

VMM Virtual Machine Monitor.

VNC Virtual Network Computing.

VNF Virtual Network Function.

VNFD VNF Descriptor.

VNFM VNF Manager.

VoIP Voice Over IP.

VXLAN Virtual Extensible LAN.

Capítulo 1

Introdução

O desenvolvimento das redes móveis e sem fio da quinta geração (5G) progrediu em ritmo acelerado nos últimos cinco anos. O sistema 5G tem a ambição de responder à mais ampla gama de serviços e aplicações na história das comunicações móveis e sem fio [4]. Ao responder aos requisitos desses serviços e aplicativos, o sistema 5G visa fornecer uma plataforma flexível que permita explorar novos casos de negócios e modelos que integrem indústrias verticais, como a automotiva, manufatura, energia, tele saúde e entretenimento [5]. A proliferação de objetos e dispositivos conectados abrirá caminho para uma ampla gama de novos serviços e modelos de negócios associados, permitindo a automação em vários setores da indústria e mercados verticais (por exemplo, energia, tele saúde, cidades inteligentes, redes veiculares, fabricação industrial etc.). Espera-se que dispositivos de comunicação autônoma criarão tráfego móvel com características significativamente diferentes do tráfego que hoje tem origem predominantemente humano [6]. A coexistência de aplicativos centrados no ser humano e nas máquinas imporá requisitos de desempenho muito diversos dos que são observados hoje.

O paradigma de fatiamento da rede proposto nas redes 5G pretende satisfazer a demanda de setores verticais que solicitam serviços de telecomunicação dedicados, fornecendo descrições de requisitos de fatia de rede sob demanda que são "voltados ao cliente". O fatiamento da rede refere-se ao processo de dividir a rede física em várias redes virtuais, cada uma arquitetada e otimizada para um aplicativo/serviço específico. Uma fatia de rede é uma rede virtual que é criada no topo de uma rede física de tal forma que o arrendatário da fatia possa operar sua própria rede física dedicada [7].

Com um número maior de solicitações de clientes e fatias de rede, uma estrutura de gerenciamento e controle de rede móvel terá, portanto, que exibir um nível significativamente maior de automação para todo o gerenciamento do ciclo de vida das suas diversas instâncias [8]. Essa automação do ciclo de vida das diversas fatias deve ser suportada por uma arquitetura que possua funções e ferramentas que implementam procedimentos cog-

nitivos para todas as fases do ciclo de vida. Dois facilitadores tecnológicos fundamentais dessa arquitetura incluem a virtualização de funções de rede, bem como funções de rede programáveis, definidas por software e recursos de infraestrutura. Outros elementos-chave constituem procedimentos e protocolos eficientes de gerenciamento e orquestração. Por fim, algoritmos de análise de dados escaláveis e centrados em serviços que exploram fontes de dados de vários domínios, complementados com mecanismos de segurança confiáveis, abrirão caminho para a implantação de serviços de rede personalizados com funções de rede virtualizadas em uma infraestrutura comum. Procurando também usufruir de todas essas vantagens e diminuir custos ocasionados pela manutenção e expansão do modelo de negócio, o Instituto Europeu de Padrões de Telecomunicações (ETSI) especificou para a arquitetura 5G o conceito de funções de rede virtualizadas [9]. Essa arquitetura possibilita a utilização de *hardware* comum do mercado ao invés de *hardware* proprietário e especializado. Ainda mais, os seus casos de uso vão desde sua utilização em centros de processamento de dados até sua inserção no núcleo de rede das redes de telecomunicações.

Nesse contexto, a computação em nuvem possibilita a construção de infraestruturas complexas e, ao mesmo tempo, oferece a utilização otimizada dos recursos de computação sob demanda através da *Internet*. Elasticidade, disponibilidade, portabilidade, compartilhamento de recursos e monitoramento são atrativos que a fez se propagar pela maioria das áreas da tecnologia da informação.

1.0.1 Motivação e Objetivos do Trabalho

Um elemento importante e pouco abordado na literatura relacionada [10, 11, 12] é o comportamento do tráfego de controle, resultante dos processos de gerenciamento e orquestração da infraestrutura virtual. Nesse contexto, este trabalho tem como objetivo geral analisar o tráfego de controle em uma infraestrutura virtualizada para as redes 5G, conforme as especificações da ETSI. Os objetivos específicos deste trabalho são os seguintes:

- Estudar e compreender os elementos principais da rede 5G, principalmente na camada de rede e na implementação das funções de serviços;
- Estudar e implementar um protótipo simples do ambiente virtualizado para um cenário 5G, conforme as especificações do ETSI;
- Validar o protótipo com trabalhos relacionados e uma aplicação típica de um cenário 5G;

- Analisar o tráfego de controle na infraestrutura virtual para identificar e caracterizar possíveis otimizações nos processos de gerenciamento e orquestração das ferramentas utilizadas.

Para alcançar os objetivos anteriormente descritos, o presente trabalho foi estruturado em três capítulos: o Capítulo 2 apresenta a fundamentação teórica dos principais elementos utilizados na pesquisa. Assim também, neste capítulo são descritas as funcionalidades básicas das ferramentas utilizadas no ambiente. O Capítulo 3 apresenta o ambiente implementado, em que são descritas as suas funcionalidades e é feita a validação do ambiente a partir da comparação com outros trabalhos relacionados e similares. Assim também, este capítulo apresenta os resultados experimentais e descreve em detalhe a análise do tráfego de controle. Finalmente, o Capítulo 4 apresenta as Conclusões e Trabalhos Futuros, em que são feitas ponderações sobre o estudo e sugestões de trabalhos futuros.

Capítulo 2

Fundamentação Teórica

Este capítulo introduz os conceitos teóricos estudados e utilizados para desenvolver o presente trabalho. São abordados os conceitos de Computação em Nuvem e Ambientes Virtualizados, assim como Redes 5G. Também é apresentada uma descrição do conjunto de ferramentas disponíveis para a implementação das funcionalidades de virtualização na arquitetura 5G.

2.1 Computação em Nuvem e Ambientes Virtualizados

A virtualização é um conceito que vem sendo estudado desde 1960 e tem como funcionamento a criação de barreiras virtuais no qual softwares executam em ambientes isolados não havendo interferência entre si. Isso é um fator de extrema importância se considerado que há uma maior chance de erros ocorrerem a nível de software do que em hardware [1]. Um software problemático pode gerar instabilidade no sistema e interferir na execução de outros programas, o que pode ser alarmante levando-se em consideração que existem serviços que necessitam estar com uma alta disponibilidade ou são críticos para o modelo de negócio de uma organização.

Uma estratégia que as empresas utilizavam para contornar esse tipo de problema, era a aquisição de vários computadores no qual cada serviço executaria somente em uma máquina e, caso houvesse instabilidade em um software, apenas aquele serviço estaria comprometido. Tal prática, era considerada um sucesso por um ponto de vista pois o problema em si era resolvido, mas, ao mesmo tempo, era considerada dispendiosa tendo em vista que haveria um custo adicional para adquirir um novo computador caso fosse implementado um serviço adicional e, além disso, teria-se o custo de manutenção daquela crescente quantidade de computadores [13].

A virtualização, portanto, é uma técnica de ocultar as características físicas dos recursos computacionais (sejam eles de armazenamento, rede ou processamento) da perspectiva de outros sistemas, aplicações ou dos próprios usuários finais. Isso é obtido por meio da criação de uma camada entre o *hardware* e o sistema operacional e aplicações que estejam executando. Essa camada intermediária da virtualização é o monitor de máquina virtual (VMM), também conhecido como hipervisor. Ele tem como responsabilidade gerenciar os recursos que as máquinas virtuais utilizam. Ele garante a estabilidade do sistema como um todo e cria a ilusão de que a VM estaria executando em um computador físico e que possui total controle dos recursos daquele sistema.

Existem dois tipos de hipervisor, sendo eles o hipervisor tipo 1, também chamado de *Bare Metal* e o hipervisor tipo 2, também conhecido como *hosted*. No primeiro caso, o hipervisor executa logo acima da camada de hardware, possuindo assim acesso privilegiado a ele. Como exemplo desse tipo de hipervisor, têm-se o KVM [14], Xen [15] e o VMware ESXi [16]. No segundo caso, o hipervisor executa no espaço de usuário assim dependendo da disponibilidade dos serviços providos pelo sistema operacional hospedeiro. Têm-se como exemplo de hipervisor tipo 2 o Oracle VirtualBox [17] e VMware Workstation [18]. Os dois tipos de hipervisor são demonstrados na Figura 2.1.

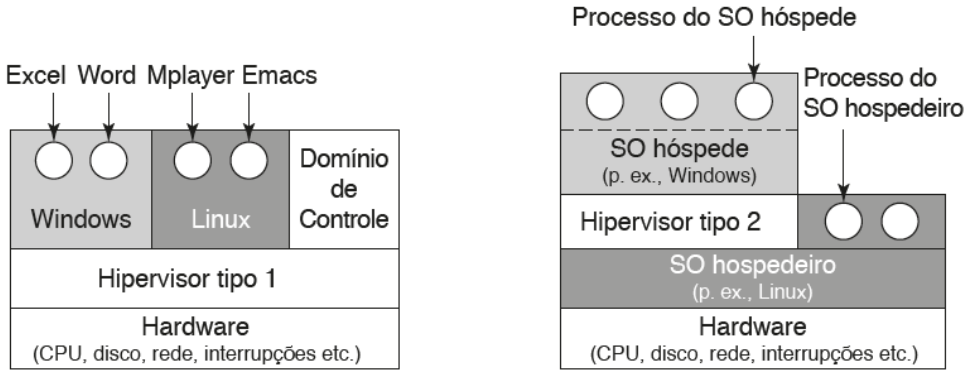


Figura 2.1: Tipos de hipervisor e sua respectiva localização [1].

Mesmo com a possibilidade de executar várias máquinas virtuais e, consequentemente, tendo-se uma menor quantidade de computadores físicos, o custo de manutenção ainda era considerável já que todos os serviços seriam executados na mesma máquina e, caso ocorresse uma falha de hardware ou software, todos os serviços ficariam indisponíveis. Além disso, era necessário a instalação integral de um sistema operacional por cada VM desenvolvida. Essa limitação impulsionou a disseminação da tecnologia de contêineres que é o agrupamento da aplicação e suas dependências de uma forma auto-contida podendo ser executado em qualquer plataforma que suporte virtualização de contêineres. A comparação entre esses dois tipos de virtualização é mostrado na Figura 2.2.

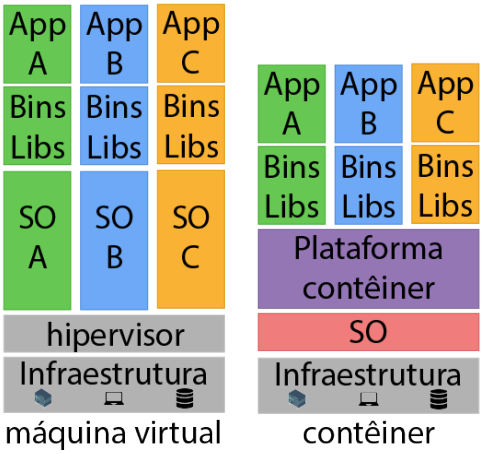


Figura 2.2: Comparação entre um ambiente com máquina virtual e contêiner.

A computação em nuvem é um conceito no qual os recursos são virtualizados, dinamicamente alocados e oferecidos como serviço pela Internet [19]. Empresas como Microsoft, Amazon e Google passaram a oferecer soluções de computação em nuvem como uma forma de aumentar a eficiência dos recursos subutilizados dos seus grandes centros de processamento de dados, *data centers*, e atender uma demanda de empresas menores que careciam de recurso computacional ou de armazenamento mas que não queriam gerir a própria infraestrutura [1].

A computação em nuvem pode ser do tipo pública, privada ou híbrida. No primeiro, ela é fornecida como serviço à terceiros pela *Internet*. No segundo, é usada somente pela organização sendo que sua infraestrutura pode ser local ou localizada em terceiros que façam o seu gerenciamento. E, no último, seria a junção dos dois primeiros tipos, em que algumas partes da nuvem seriam de uso interno da empresa e outras partes seriam dado o acesso ao público. Além disso, o fornecimento desse serviço possui três tipos de modelos: Infraestrutura como Serviço (IaaS), Plataforma como Serviço (PaaS) e Software como Serviço (SaaS). Com a popularização da computação em nuvem, outros serviços começaram a ser criados como, por exemplo, Rede como Serviço (NaaS).

Infraestrutura como serviço é quando o provedor da nuvem possibilita o controle total dos recursos ao usuário. O cliente consegue, por exemplo, efetuar a criação de uma rede virtual, hospedar suas próprias aplicações e fazer o provisionamento de processamento e armazenamento. Plataforma como serviço, por outro lado, já possui um escopo mais limitado do que o anterior. O provedor fornecerá os serviços que podem ser usados dentro da nuvem. Esses serviços, por exemplo, podem ser desde um banco de dados até o desenvolvimento e hospedagem de aplicações web. E, por último, o Software como serviço que é simplesmente a utilização de um software remotamente com a promessa de

alta disponibilidade pelo provedor. Na Figura 2.3 é explicitada a diferença entre esses tipos de modelo citados, por um lado o consumidor utilizando IaaS tem mais poder sobre o que pode ser feito na infraestrutura, mas em contrapartida ele ganha uma grande tarefa de fazer o gerenciamento do ambiente. Diferente do SaaS que tudo é gerenciado pelo provedor mas o consumidor fica a mercê das soluções que o provedor fornece.

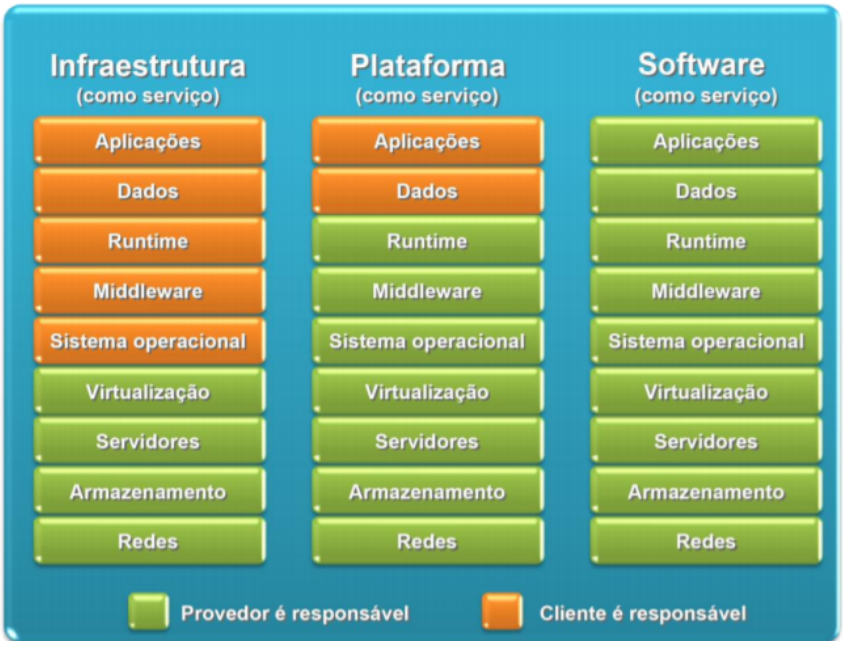


Figura 2.3: A diferença entre os três tipos de modelos de nuvem [2].

A computação em nuvem possibilita uma alta agilidade, conectividade, disponibilidade e o provisionamento automático dos recursos. Essas vantagens e entre outras proporcionam uma quebra de paradigma sobre como os serviços de hoje em dia devem ser modelados. Considerando a vasta quantidade de dispositivos que se conectam a *Internet* e que esse número tende a crescer de uma forma cada vez mais acelerada a cada ano [6]. É necessário que haja uma alta adaptabilidade dos serviços prestados para possibilitar uma fácil expansão do modelo de negócio e também de sua manutenção. Levando em conta esses fatores e a versatilidade de casos onde a computação em nuvem pode ser utilizada, ela conseguiu até alcançar as redes de telecomunicações em que ela é empregada na arquitetura proposta pela ETSI.

Nas seções seguintes serão apresentadas e descritas um conjunto de ferramentas de virtualização, orquestração e supervisão, que foram pesquisadas e aplicadas dentro do contexto deste trabalho.

2.2 OpenStack

OpenStack é uma solução de computação em nuvem criado em 2010 pela colaboração conjunta da NASA com a Rackspace [20]. Desde o seu início, já tinha-se como meta uma solução que atendesse os requisitos de uma nuvem pública ou privada independentemente do seu tamanho, que fosse simples de implementar e que pudesse ser escalável [21]. O OpenStack, desde sua criação, teve uma comunidade muito ativa sempre efetuando melhorias e adicionando funcionalidades de acordo com as tendências e necessidades do mercado. A ferramenta continua sendo uma tecnologia altamente relevante a ser utilizada em novas aplicações visto que muitos estudos ainda continuam sendo feitos como, por exemplo, proposta de algoritmo de alocação de VMs levando em conta o custo energético [22] e de análises de vulnerabilidades [23].

O OpenStack é uma solução que segue o modelo IaaS. Ele foi construído de forma totalmente modularizada, o que permite a adição ou remoção de componentes de acordo com o serviço que deseja-se alcançar. Além disso, esses componentes podem ser reutilizados e não precisam estar executando na mesma máquina uma vez que o desenvolvimento do OpenStack teve como mente a utilização da arquitetura orientada a serviços (SOA). O ambiente que será detalhado posteriormente nesse documento no capítulo 3 utiliza o OpenStack em sua configuração mínima, onde só é necessário a presença dos seguintes componentes: *Keystone*, *Glance*, *Placement*, *Nova* e *Neutron*. A arquitetura conceitual que é mostrada a interação entre esses componentes pode ser visto na Figura 2.4.

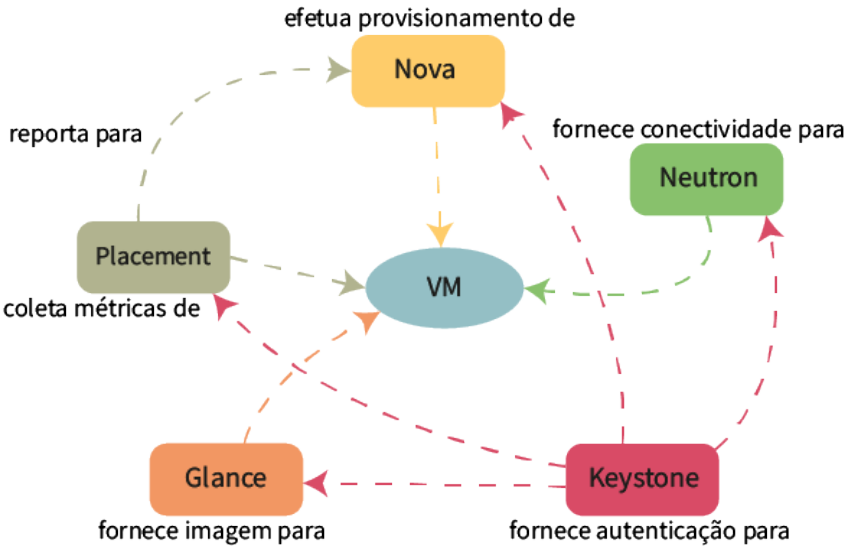


Figura 2.4: Arquitetura conceitual do OpenStack.

2.2.1 Keystone

O serviço de identidade, **keystone**, é um dos principais componentes do OpenStack. Ele é responsável pela autenticação e autorização em todo o ambiente virtualizado [24]. Todo serviço, seja ele interno ou não, e usuário que deseja se comunicar diretamente com algum dos serviços internos do OpenStack, deve primeiramente autenticar-se no keystone. Após isso, será gerado um *token* que internamente possui as permissões de acesso daquele usuário e que deverá ser passado ao serviço no qual deseja utilizar. Basicamente, o keystone possui um conjunto de estruturas que o ajuda a definir níveis de acesso a componentes e as suas relações. Essas estruturas são de: usuários, grupos (conjunto de usuários), projetos e domínios (agrupamento de projetos, usuários e grupos).

2.2.2 Glance

O serviço de imagem, **glance**, é responsável pelo gerenciamento das imagens das entidades virtualizadas sejam elas máquinas virtuais ou contêineres [25]. Todos os dados referentes a uma imagem são salvos em um banco de dados junto com seus respectivos metadados. Os metadados são informações adicionais que podem ajudar o OpenStack em decidir em qual hospedeiro a máquina virtual deve ser instanciada. Esses metadados podem conter informações desde a arquitetura do processador necessário para executar aquela imagem até o mínimo de memória secundária necessária para a correta execução da VM utilizando aquela imagem.

2.2.3 Placement

O serviço **placement** foi introduzido na versão Newton (2016) do OpenStack. Ele foi criado para desacoplar a parte responsável por gerenciar os recursos físicos e virtualizados do ambiente da parte de decisão sobre qual máquina hospedeira instanciar uma máquina virtual. O *placement* cria uma árvore onde cada nó é um recurso podendo ser qualitativo ou quantitativo. Uma mídia de armazenamento, por exemplo, é vista como um recurso, essa mídia pode ser do tipo disco rígido ou de estado sólido. Dessa maneira, isso seria uma qualidade desse recurso que seria um nó filho do nó que representa a mídia de armazenamento. Com isso, o *placement* consegue criar um inventário de recursos que estão disponíveis na infraestrutura. Essa árvore é atualizada frequentemente pelos nós de computação para que o servidor de controle possa escolher o melhor hospedeiro no qual a máquina virtual será instanciada segundo uma série de critérios que podem ser ajustados pelo administrador da nuvem de computação [26].

2.2.4 Nova

O serviço de computação, **nova**, é responsável pelo gerenciamento das máquinas virtuais [27]. Ele consegue se comunicar com uma variedade de hipervisors para garantir uma maior interoperabilidade em uma estrutura possivelmente heterogênea. O nova alcança esse objetivo por meio de uma biblioteca auxiliar chamade de *libvirt* que o possibilita usar uma diversidade de hipervisors como: QEMU, LXC, VirtualBox, VMware ESX e Microsoft Hyper-V. A tarefa de gerenciar uma VM em um ambiente de nuvem é uma tarefa complexa, por isso, o nova é dividido em subcomponentes.

1. **Nova-api**: recebe e executa ações relacionadas ao gerenciamento das máquinas virtuais podendo ser requisitado a lista de servidores virtuais executando ou a sua situação por exemplo;
2. **Nova-api-metadata**: é um componente que possibilita que uma VM consiga receber informações extras por meio de uma requisição REST API.
3. **Nova-compute**: é a parte principal do nova que se comunica com o hipervisor da máquina hospedeira. Esse componente estará tanto no servidor principal quanto no nó de computação.
4. **Nova-scheduler**: é responsável por decidir em qual nó de computação deve ser instanciado a máquina virtual. Para ser feita essa decisão, são levados em consideração uma coleção de filtros em que é analisado as características da máquina hospedeira, da imagem e das configurações da VM. Esses filtros podem ser modificados e até receber pesos que alteram a sua prioridade durante o processo.
5. **Nova-conductor**: informações que são pertinentes aos nós de computação não são acessadas diretamente ao banco de dados por eles. Essa intermediação é feita por esse componente.

2.2.5 Neutron

O serviço de rede, **neutron**, é o que fornece conectividade a máquina virtual. Após a VM ter sido instanciada, o neutron cria as *interfaces* virtuais necessárias e as conecta as redes virtualizadas no qual aquele nó de computação foi configurado. Essas redes virtuais podem ser internas da infraestrutura ou possibilitar que a VM tenha acesso a Internet. Além disso, o neutron é bastante flexível sendo possível criar várias redes virtuais com as mais diferentes topologias e tudo sendo abstraído da rede física. O serviço consegue criar dispositivos virtuais que agem tanto na camada de rede quanto na camada de enlace de dados [28].

2.2.6 Mensagens de Controle

As mensagens de controle do OpenStack são transmitidas tanto por HTTP quanto por AMQP (*Advanced Message Queuing Protocol*). O primeiro protocolo é mais utilizado na autenticação, por meio do REST API, dos serviços com o keystone. Por meio dele é feito o gerenciamento das imagens armazenadas pelo serviço glance e a atualização da árvore de recursos do serviço placement. Algumas outras funcionalidades também ocorrem por meio do HTTP como o gerenciamento dos componentes do OpenStack.

O AMQP é o meio de comunicação por onde ocorre a maioria das mensagens entre os serviços do OpenStack. É por meio desse protocolo que é feito a verificação da situação dos serviços (*health check*) e a coordenação dos componentes que compõem a infraestrutura. É possível utilizar como intermediador de mensagens (*message broker*) o RabbitMQ, Qpid ou ZeroMQ. Eles recebem as mensagens, gerada pelos produtores, e garantem que as mensagens sejam entregues ao destinatário correto, chamado de consumidor. Para que ocorra o envio e recebimento das mensagens, é necessário antes efetuar a criação de entidades que são responsáveis pelo armazenamento e encaminhamento delas, sendo essas entidades: *queue*, *exchange*, *routing-key* e *binding-key*. Além disso, é necessário interligar o *exchange* com a fila (*queue*), processo chamado de *binding*, já que a mensagem primeiro é encaminhada ao *exchange* e a partir daí é colocada na fila. A Figura 2.5 demonstra a interação entre esses objetos.

1. **Routing-key**: é o tópico da mensagem. O *exchange*, dependendo do tipo, define para qual fila deve ser encaminhado a mensagem por meio desse atríbuto.

2. **Queue**: é o local em que as mensagens ficarão armazenadas até que seja confirmado o recebimento pelo destinatário. No ambiente utilizado, essa fila de mensagens ficará no servidor de controle. O AMQP utiliza o modelo de publicar-assinar (*publish-subscribe*). Dessa maneira, apenas os destinatários que tenham se cadastrado para mensagens daquela fila serão notificados.

3. **Exchange**: é a parte que recebe a mensagem e decide para qual fila aquela mensagem deve ser encaminhada, o despacho da mensagem depende do tipo de *exchange*. O tipo de *exchange direct* apenas verifica se coincide o *routing-key* da mensagem com o *binding-key* da fila. O *exchange fanout* ignora o *routing-key* da mensagem e a transmite para todas as filas em que aquele exchange está interligado. O *exchange topic* é parecido com o *direct* na questão da verificação, mas nesse caso a coincidência será parcial.

4. **Binding-key**: são as regras de roteamento que o *exchange* utiliza para a distribuição das mensagens para as filas. O *binding* pode ser visto como uma 3-upla composta

por (*Queue*, *Exchange*, *Routing-key*) sendo o *Routing-key* opcional. Dessa maneira, uma fila é atrelada a um *Exchange* e recebe mensagens encaminhadas por ele.

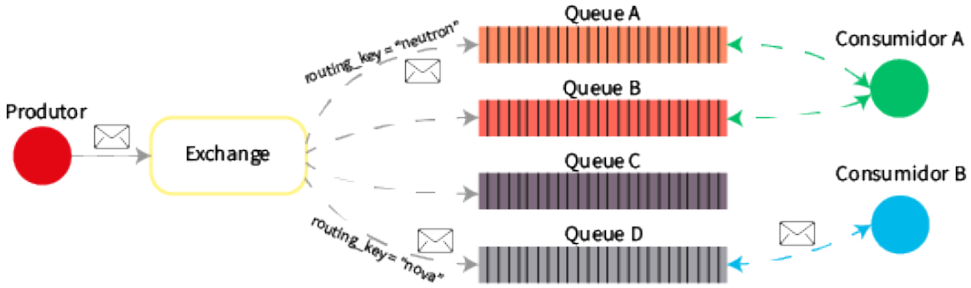


Figura 2.5: Demonstração da relação entre *exchange*, *queue* e a trajetória feita pelas mensagens.

2.3 Redes 5G

A quinta geração do padrão de tecnologia dos sistemas de telecomunicações (5G) tem como propósito atender uma sociedade móvel e totalmente conectada oferecendo altas taxas de velocidade, ampla cobertura, baixa latência e a habilidade de suportar vários dispositivos conectados [8]. Além disso, é a primeira geração das redes móveis que tem como objetivo suprir as necessidades de uma indústria vertical. Diferente das gerações móveis anteriores, não há uma competição entre organizações de padrão trabalhando em uma solução para o 5G. A 3GPP (*3rd Generation Partnership Project*) é o consórcio que está definindo as especificações técnicas para as redes 5G e que tiveram seu inicio em 2016. Para garantir uma transição gradual entre as gerações móveis pelos operadores de rede móvel, a 3GPP especificou dois tipos de implantações que podem ser feitas, sendo elas: autônoma (SA) e não-autônoma (NSA). O modelo NSA permite a reutilização da infraestrutura da geração antiga (4G) utilizando o 5G NR (*New Radio*). Enquanto que o modelo SA tem como meta ser o objetivo final da transição onde não haverá nenhuma dependência da infraestrutura do 5G com gerações de redes móveis anteriores, nele será possível alcançar o potencial proposto pelas redes 5G e a utilização do 5GC (5G Core).

O 5G NR, a nova tecnologia de acesso por rádio a ser utilizada, é o componente que ditará quais serviços serão suportados pelas redes 5G. Tendo em mente uma grande variedade de serviços a serem suportados, a 3GPP os separou em três grandes cenários, sendo eles: banda larga móvel aprimorada (*Enhanced Mobile Broadband* - eMBB), comunicação de baixa latência ultraconfiável (*Ultra-Reliable Low Latency Communication* - URLLC) e comunicação do tipo de máquina massiva (*Massive Machine Type Communication* - mMTC). Na primeira parte da transição NSA, foi dado foco ao eMBB. Esse

cenário concentra-se em atingir altas taxas de velocidade com uma moderada latência. Para atingir esse objetivo, é necessário utilizar tecnologias que utilizem um espectro de frequência diferente, por exemplo onda milimétrica, e um conjunto de antenas de transmissão e recepção que possibilitem múltiplas entradas e múltiplas saídas (MIMO).

Dessa maneira, tem-se como objetivo alcançar altos picos de *downlink* de 20 Gbps diferente da geração anterior 4G LTE que tinha uma taxa máxima de 1 Gbps. No cenário URLLC, o foco será garantir baixas latências de 1 ms ou menos para serviços sensíveis ao tempo como de carros autônomos, cirurgias feitas a longa distância e dispositivos da área de saúde como medidores de pulsação. O cenário mMTC tem como função prover conectividade para bilhões de dispositivos, esse caso em específico é voltado para a Internet das Coisas (IoT) onde se tem vários equipamentos, como eletrodomésticos, que se conectam à Internet. Levando em consideração todos esses três cenários, suportar todos em uma única infraestrutura seria uma tarefa muito complexa, por causa disso o 5G utiliza um conceito de fatia de rede (*Network Slice*). *Network Slice* é a execução de múltiplos serviços lógicos, nesse caso das redes de telecomunicações, onde cada um estará isolado, mas compartilhando uma mesma infraestrutura [8]. Na Figura 2.6 há um exemplo de *Network Slice* onde duas fatias de rede compartilham uma mesma infraestrutura física. Essas fatias, por exemplo, podem ser de organizações diferentes que estão usando a infraestrutura de terceiros ou uma mesma empresa separando logicamente a rede dos seus serviços.

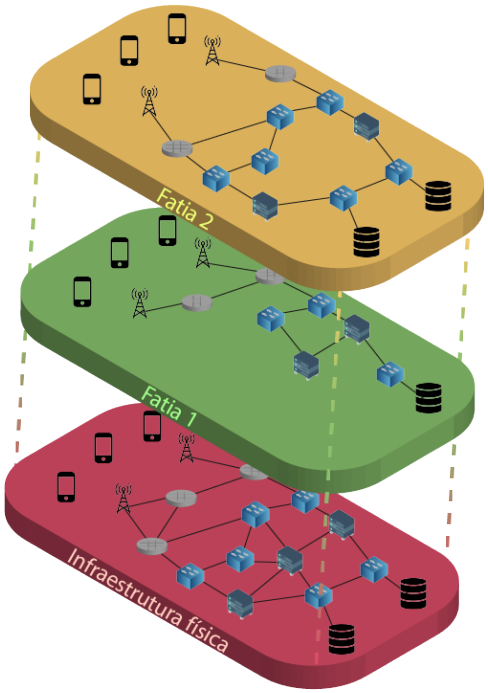


Figura 2.6: Exemplo de fatia de rede (*Network Slice)*.

O 5GC, é a arquitetura núcleo do 5G que tem como projeto ser uma solução nativa em computação em nuvem. Ela se apoia na arquitetura baseada em serviço (SBA) [7] em que cada função de rede (NF) é uma entidade reutilizável e independente que se comunica por meio de um barramento em comum (SBI) utilizando um protocolo em comum (HTTP/2) no qual juntas oferecem um serviço. Essas funções de rede, que podem ser vista na Figura 2.7 interagem em uma arquitetura de Provedor-Consumidor. Essa arquitetura possibilita uma composição de recursos tanto físicos quanto virtuais. Para alcançar esse objetivo e virtualização dessas funções de rede, o núcleo de rede 5GC tem como pilares tecnologias como SDN (*Software-Defined Network*) e NFV (*Network Function Virtualization*).

Alguns elementos funcionais mostrados na Figura 2.7 que compõem a arquitetura 5GC SBA são próprios das redes 5G como a função seletora de fatia de rede (NSSF) que seleciona a fatia de rede no qual o dispositivo irá acessar, a função de exposição de rede (NEF) que expõe serviços e funcionalidades para o núcleo da rede e a função de repositório de rede (NRF) que proporciona uma centralização das informações de cada elemento da rede e assim permite que as funções de redes descubram os outros elementos funcionais pertencentes a mesma rede. A função de gerenciamento de acesso e mobilidade (AMF) comunica-se com vários elementos do núcleo da rede como a função de autenticação (AUSF), a entidade que gerencia a inscrição do dispositivo (UDM), a função de controle de políticas e cobranças (PCF), que efetua o monitoramento das funções do plano de controle, e a de gerenciamento de sessão do usuário (SMF). E, por último, a função de plano do usuário (UPF) é responsável pelo roteamento e encaminhamento do pacote e asseguramento da qualidade do serviço (QoS) fornecido.

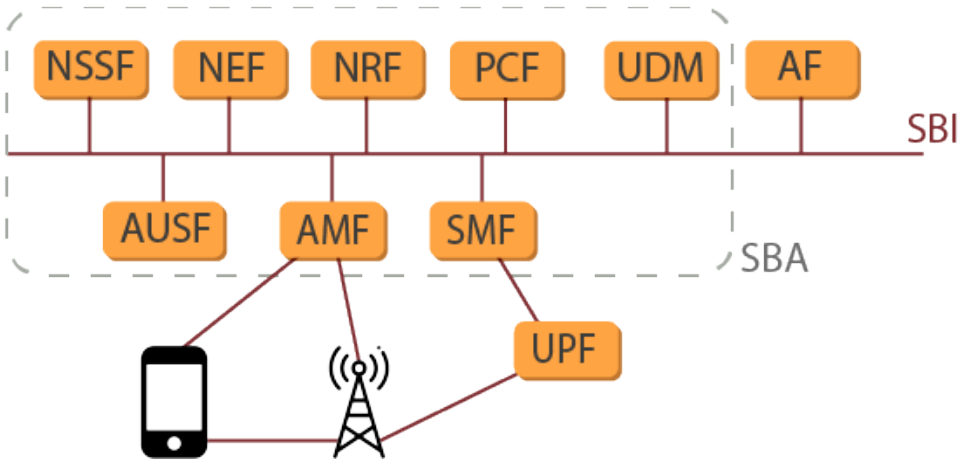
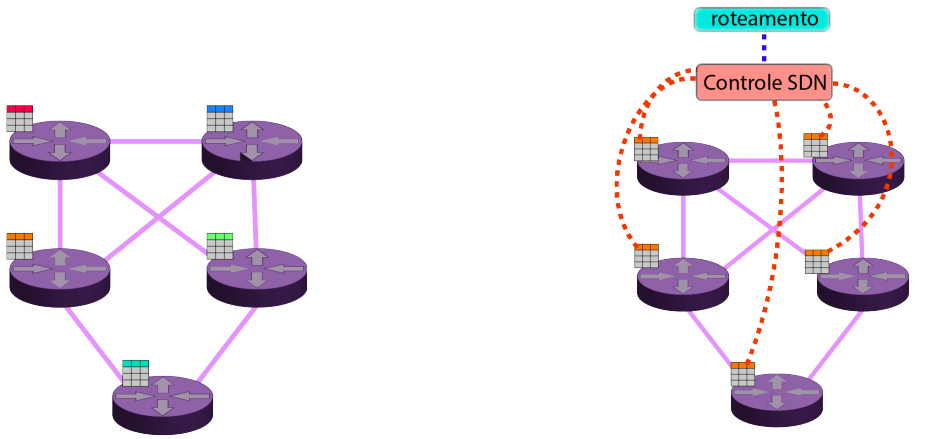


Figura 2.7: Arquitetura 5G SBA.

2.3.1 Rede definida por software

Rede definida por software (SDN) é um paradigma no qual há uma separação entre a parte de controle e a parte de dados de uma rede de computadores [29]. Essa prática se tornou atrativa pois torna a rede facilmente programável podendo assim adaptar-se rapidamente dependendo da quantidade de tráfego ou eventuais falhas dentro da rede por exemplo. Enquanto que no modelo tradicional cada dispositivo possui uma entidade de controle que define as regras de roteamento e as decisões a serem efetuadas, na SDN há uma entidade de controle centralizada que possui uma visão global sobre a rede, podendo assim modificar o comportamento dos dispositivos da rede de acordo com as instruções ou necessidades da aplicação de rede.

A entidade de controle possui duas *interfaces*: *southbound* e *northbound*. A *interface southbound* (em vermelho na Figura 2.8) é por onde ocorre a parte de controle da entidade controladora com os outros dispositivos. O protocolo mais comumente usado nessa parte, e também o primeiro a ser proposto, é o OpenFlow [29]. A *interface northbound* (em azul na Figura 2.8), por outro lado, é a *interface* no qual a entidade controladora é controlada. Essa *interface* tem como propósito que as aplicações da rede como roteamento, controle de acesso e balanceador de carrega controlem o *SDN Controller* e assim, por consequência, mudem o comportamento dos dispositivos da rede para alcançar o comportamento desejado. É importante notar que a utilização de SDN possibilita a virtualização da parte de controle, ou seja, os dispositivos controlados continuam sendo físicos. Além disso. esses dispositivos são especializados para aquele uso, dessa maneira, gera-se um custo adicional na aquisição já que se trata de um hardware especializado.



Rede Tradicional

Rede definida por Software

Figura 2.8: Comparação entre rede tradicional e SDN

2.3.2 Rede virtualizada de funções

A rede atual de computadores é composta por dispositivos proprietários e especializados, também referidos como *middle-boxes*, que juntos fornecem os serviços de rede. Esse modelo de negócio é bastante custoso para as operadoras de rede já que a alta necessidade em atualizar suas estruturas seguindo as novas tendências, e também a expansão das mesmas, exige um alto gasto em custos de capital (CAPEX) e elevados custos operacionais (OPEX). Dessa maneira, uma forma de diminuir esses custos é a utilização de hardware comum presente no mercado (COTS) já que o alto preço desses *middle-boxes* se deve justamente por serem uma solução personalizada. Diante desse impasse e com outros setores da área de tecnologia da informação se beneficiando das vantagens oferecidas pela computação em nuvem, em 2012 foi publicado pela ETSI [9] uma proposta de arquitetura titulada de rede virtualizada de funções procurando minimizar esses gastos citados e, ao mesmo tempo, aumentando a flexibilidade da rede e minimizando o tempo necessário para que os operadores de rede amadurecessem sua infraestrutura.

Rede virtualizada de funções (NFV) é a separação de serviços de rede dos dispositivos físicos, *middle-boxes*, que os fornecem. As grandes provedoras de serviço Internet (ISP) e operadores de telecomunicações como *Deutsche Telekom* e *British Telecom* formaram um grupo de especificações industriais (ISG) dentro da ETSI e definiram quais são as especificações de um arquitetura NFV em 2012 [30] como uma forma de padronizar e solucionar os altos gastos causados pelo uso de dispositivos proprietários. A especificação da arquitetura VNF [30] visa garantir uma interoperabilidade entre as soluções que seguem as diretivas da especificação e a definição de algumas nomenclaturas do domínio como, por exemplo, VNF. VNF é o nome dado as funções de rede que antes eram executadas a nível de hardware e passaram a ser executadas a nível de software. Essas VNFs podem exercer funções desde *firewall* e NAT até de balanceadores de carga de uma forma bastante flexibilizada já que seria apenas necessário instanciar uma VNF em um ponto na rede.

A Figura 2.11 é uma visão de alto nível da arquitetura ETSI NFV onde apenas são mostrados os componentes principais. A infraestrutura da rede virtualizada de funções (NFVI) é responsável pela gerência e administração dos recursos físicos e fornecimento dos recursos virtuais para as VNFs. O EM refere-se ao componente que controlará a VNF. O gerenciador da infraestrutura virtualizada (VIM) é responsável por intermediar a comunicação com a infraestrutura virtualizada, enquanto que o VNFM é responsável pelo gerenciamento do ciclo de vida das VNFs. E, finalmente, o orquestrador NFV (NFVO) é responsável por agrupar as VNFs e garantir que elas consigam se comunicar entre si. A parte da direita da Figura 2.11 é o gerenciador e orquestrador NFV MANO e engloba todos os componentes que permitirão o funcionamento da arquitetura NFV. Nesse trabalho, será utilizado o Open Source MANO (OSM) que terá como responsabilidade essa função. Ele

se trata de um arcabouço que segue as diretivas, e também é hospedado, pela ETSI.

No OSM há um conceito de descritor de VNF (VNFD), basicamente é um arquivo informando qual a entidade virtualizada (VDU) e as suas conexões externas (CP). Um exemplo de VNFD pode ser visto na Figura 2.9, o CP é conectado na *interface* da entidade, essas *interfaces* serão os pontos que serão externalizados por essa VNF quando ela for usado em um serviço. Uma definição de serviço de rede (NSD) é composto por VNFDs e *links* virtuais que as conectam (VL). Uma vez definido um serviço, ele pode ser executado no OSM e ele encarregará a NFVI de executar e garantir a conectividade entre as VNFs. Um exemplo de NSD pode ser visto na Figura 2.10, esse exemplo será utilizado como cenário no capítulo de Resultados Experimentais quando for variado a quantidade de VNFs presentes na infraestrutura.

Exemplo de um arquivo VNFD para a criação de VNF dentro do OSM e sua demonstração na Figura 2.9.

```
vnfd:vnfd-catalog:
  vnfd:
  - connection-point:
    - name: vnf_control
      type: VPORT
    id: zeus_vnf
    mgmt-interface:
      cp: vnf_control
    name: zeus_vnf
    short-name: zeus_vnf
    vdu:
    - count: '1'
      id: zeus_vnf
      image: alpine-x86_64
      interface:
      - external-connection-point-ref: vnf_control
        name: eth_hztp
        type: EXTERNAL
        virtual-interface:
          type: VIRTIO
      - external-connection-point-ref: vnf_data
        name: eth_dwrt
        type: EXTERNAL
        virtual-interface:
          type: VIRTIO
      name: zeus_vnf
```

```
      vm-flavor:
        memory-mb: '64'
        storage-gb: '0'
        vcpu-count: '1'
    version: '1.0'
```

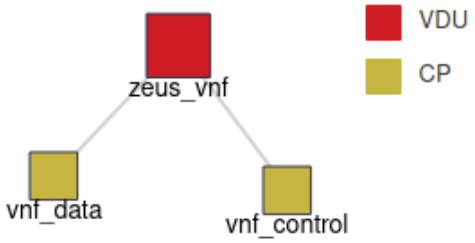


Figura 2.9: Exemplo de uma VNF.

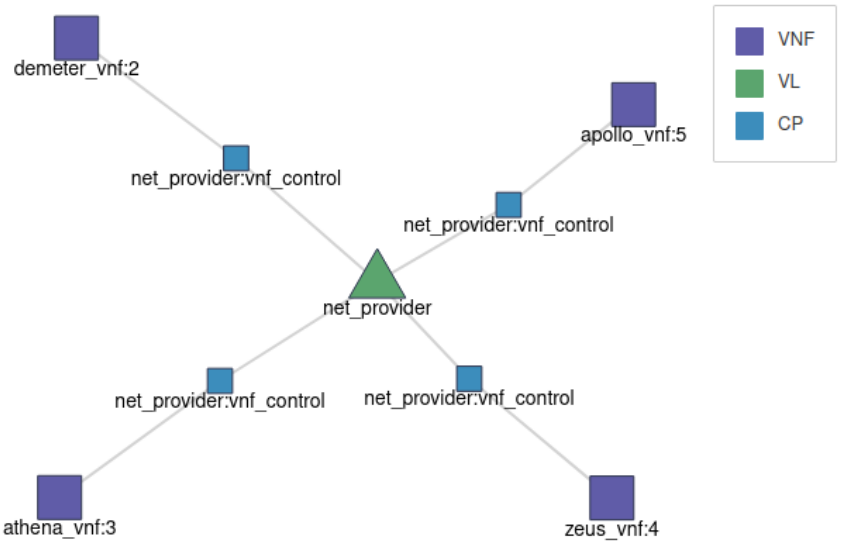


Figura 2.10: Exemplo de um serviço de rede composto por várias VNFs.

2.4 Resumo do capítulo

Nesse capítulo foi apresentado os conceitos no qual as ferramentas utilizadas no ambiente se apoiam. Além disso, foi introduzido o conceito de funções virtualizadas de rede, sendo esse um dos temas principais desse documento. Para isso, foi necessário detalhar outros conceitos no qual NFV se apoia como virtualização e computação em nuvem. Além

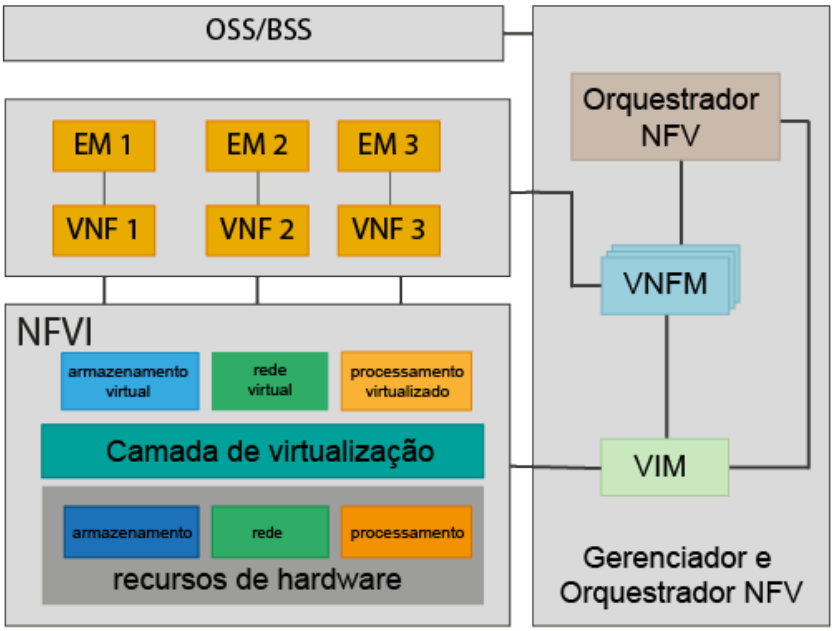


Figura 2.11: Arquitetura ETSI NFV em alto nível.

disso, foi descrito e explicado o funcionamento das ferramentas que são utilizadas no ambiente. Lembrando que o OpenStack é uma solução em nuvem complexa que possibilita sua utilização em diversos cenários, a configuração utilizada no ambiente será com a quantidade mínima de componentes necessários para o seu correto funcionamento.

Capítulo 3

Ambiente Implementado e Resultados Experimentais

Nesse capítulo será apresentado o ambiente virtualizado construído conforme as especificações da ETSI. Sera mostrado sua composição e a configuração das máquinas utilizadas. Também o capítulo apresenta a validação do ambiente utilizando um cenário similar ao proposto em outro trabalho. Finalmente, o capítulo apresenta uma análise do conteúdo das mensagens de controle transmitidas pela infraestrutura e suas respectivas frequências e serão feitas considerações sobre os cenários propostos e sobre os resultados obtidos.

3.1 Ambiente de Implementação

O ambiente de implementação foi definido a partir do uso de dispositivos simples e de baixo custo, a saber:

1. 1 notebook com um processador i7-8565U com 16GB de RAM e 1TB de armazenamento utilizando o sistema operacional Ubuntu 21.04 *Impish Indri*.
2. 1 Raspberry Pi (RPi) 4 Modelo B com 4GB de RAM e 32GB de armazenamento com o sistema operacional Ubuntu 18.04 LTS *Bionic Beaver*.
3. 2 notebooks com processador i7-4500U com 8GB de RAM e 1TB de armazenamento com o sistema operacional Ubuntu 18.04 LTS *Bionic Beaver*.

Em cada um dos dispositivos foi instalado a versão *Train* do OpenStack lançada em 2019. Tanto o OpenStack quanto o OSM foram configurados utilizando os guias de instalação fornecidos por cada um dos arcabouços [31, 32]. Todos os dispositivos estão conectados a um roteador com uma conexão cabeada e uma conexão sem fio. Na conexão sem fio ocorrerá o tráfego de controle da infraestrutura enquanto que na conexão

cabeada ocorrerá o tráfego de dados que será por onde as máquinas virtuais do ambiente virtualizado se comunicarão entre si. Para que o ambiente consiga ser demonstrado de forma clara e, que posteriormente seja fácil referenciar, foi definido um nome para cada máquina. A Figura 3.1 mostra a organização das máquinas com seus respectivos IPs e identificadores.

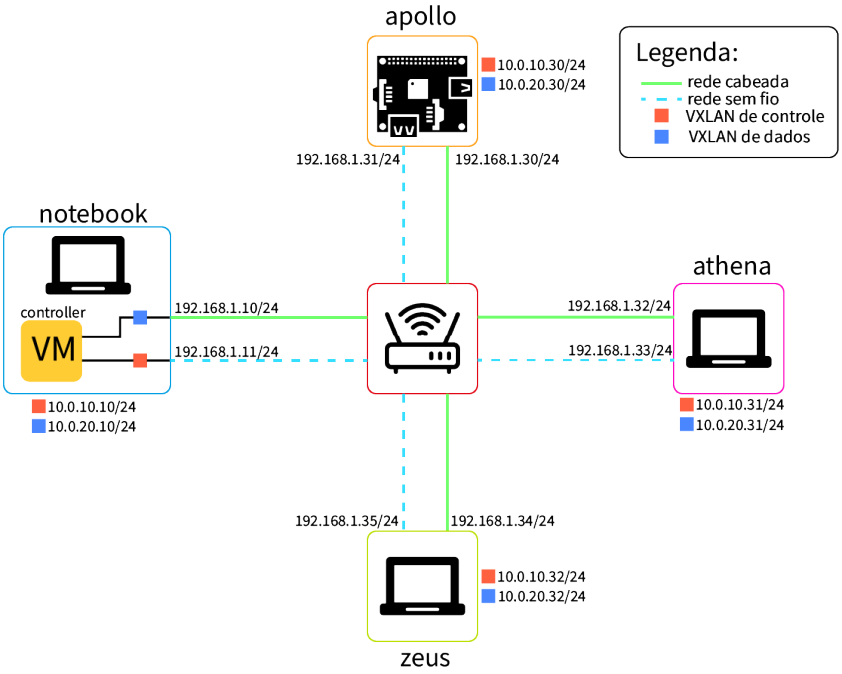


Figura 3.1: Topologia do ambiente utilizado.

Conforme ilustrado na Figura 3.1, o *controller* é executado como uma máquina virtual. Essa VM é conectada a duas VXLANs que estão presentes em todos os outros nós de computação. As VXLANs que possuem a mesma cor, estarão utilizando o mesmo VXLAN *ID*. Dessa maneira, dispositivos que conectem a essas VXLANs estarão em um mesmo barramento virtual isolado.

Os serviços do OpenStack citados no capítulo de Fundamentação Teórica estão todos executando no *controller* enquanto que apenas os serviços Nova e Neutron que executam nos nós de computação. Por se tratar de uma infraestrutura heterogênea, foi necessário criar imagens para a arquitetura ARM64 por causa do Raspberry Pi 4 e imagens com a arquitetura x86-64 para os outros nós de computação. Além disso, para se ter um maior controle sobre onde uma máquina virtual deve ser executada quando utilizado o OSM, foi determinado uma zona pra cada nó de computação. Dessa maneira, VMs que estejam na zona athena executaram no nó de computação athena, VMs que estejam na zona apollo executaram no nó de computação apollo e assim por diante. A versão do OSM utilizado é a 6, esse componente executa no servidor de controle.

Na Tabela 3.1 é mostrado os serviços do neutron em cada nó de computação e no servidor de controle com suas respectivas situações. Foi utilizado o *neutron-linuxbridge-agent* como mecanismo para conectar as *VM*s. Dessa maneira, toda máquina virtual quando criada é conectada a uma *bridge* virtual criada pelo neutron. Esse dispositivo virtualizado é conectado a uma rede virtual seguindo o mapeamento definido nas configurações da infraestrutura. Algumas colunas da Tabela 3.1 foram omitidas para caber no documento.

Tipo de Agente	Hospedeiro	Situação	Binário
Linux bridge agent	apollo	UP	neutron-linuxbridge-agent
DHCP agent	controller	UP	neutron-dhcp-agent
Linux bridge agent	athena	UP	neutron-linuxbridge-agent
Linux bridge agent	zeus	UP	neutron-linuxbridge-agent
Linux bridge agent	controller	UP	neutron-linuxbridge-agent
Metadata agent	controller	UP	neutron-metadata-agent
L3 agent	controller	UP	neutron-l3-agent

Tabela 3.1: Resultado da requisição sobre os agentes do neutron nos nós de computação e servidor central

Na Tabela 3.2 é mostrado os serviços do nova em cada nó de computação e no servidor de controle com suas respectivas situações. Como falado anteriormente, cada nó de computação foi atribuído a uma região para se ter uma maior controle sobre onde a máquina virtual deveria executar. No ambiente utilizado cada região possui apenas um nó de computação, mas na verdade é possível ter um conjunto de nós de computação em uma mesma região. Isso visa facilitar na organização do ambiente em setores ou departamentos de uma organização ou até regiões geográficas. Algumas colunas da Tabela 3.2 foram omitidas para caber no documento.

Binário	Hospedeiro	Zona	Situação
nova-scheduler	controller	internal	up
nova-conductor	controller	internal	up
nova-compute	athena	athena_zone	up
nova-compute	apollo	apollo_zone	up
nova-compute	zeus	zeus_zone	up

Tabela 3.2: Resultado da requisição sobre os agentes do nova nos nós de computação e servidor central

3.2 Validação do Ambiente

Após a configuração do ambiente, para se ter uma maior confiança de que não houve nenhum equivoco na configuração do ambiente, efetuou-se a comparação com outro trabalho que possui um ambiente virtualizado semelhante. No trabalho de Cristoffer Leite [3], a infraestrutura é composta por quatro Raspberry Pi (Figura 3.2) enquanto que nesse trabalho proposto é utilizado três nós de computação sendo apenas um sendo Raspberry Pi. São utilizadas as mesmas ferramentas OpenStack e OSM, porém no trabalho citado as versões utilizadas são mais antigas. Os ambientes em si possuem características semelhantes em termos funcionais, onde o papel do OpenStack está centrado em abstrair as nuancias do OSM.

No trabalho citado foi proposto um cenário VoIP, típico nas redes 5G para os cenários de baixa latência ultra confiável (URLLC). Na Figura 3.3 é possível visualizar o serviço de rede utilizado no trabalho, com as redes virtualizadas (uma para controle e outra para dados) e as quatro VNFs que possibilitam a execução desse tipo de serviço. Segundo o trabalho, foi utilizado uma carga de trabalho sintética VoIP composta por pacotes de 20 Bytes enviados com um período de amostra de 20 ms onde o tempo entre cada pacote foi de 0,02 segundos usando uma modelagem de distribuição normal com um desvio padrão de 0,0038 segundos. Além disso, foi utilizado o *Codec* G.729 para chamadas VoIP considerando uma ligação com duração de 180 segundos.

No ambiente criado, foi utilizado uma carga de trabalho sintética VoIP [33] com características semelhantes a da proposta no trabalho de Cristoffer. Foi utilizado também o *Codec* G.729 para chamadas VoIP considerando uma ligação com duração de 180 segundos, a carga é composta por pacotes de 20 Bytes enviados em um período de amostra de 20 ms em que o tempo médio entre cada pacote é de 0,02 segundos usando uma modelagem de distribuição normal com um desvio padrão de 0,0038 segundos. O serviço de rede utilizado como cenário VoIP pode ser visto na Figura 3.5. Esse serviço possui características semelhantes ao do desenvolvido por Cristoffer. Foi utilizado o servidor SIP Kamailio [34] em uma VNF, o Dnsmasq [35] para agir como servidor DHCP e DNS para os dispositivos que se associavam com os pontos de acesso. E, finalmente, uma VNF preparada para executar no apollo (Raspberry Pi) e outra para athena (notebook) que atuariam como pontos de acesso.

O resultado obtido no trabalho citado pode ser visto na Figura 3.4, enquanto que o resultado do ambiente virtualizado da Figura 3.1 pode ser visto na Figura 3.6. No gráfico é possível perceber que os resultados são similares quando comparados os picos de tráfego de controle, assim como a periodicidade de 60 segundos das mensagens. No trabalho citado obteve-se uma média de vazão de dados de 30.08 kbps com pico de 32.85 kbps. Os resultados obtidos neste trabalho apresentam uma média de vazão 23.90 kbps com pico

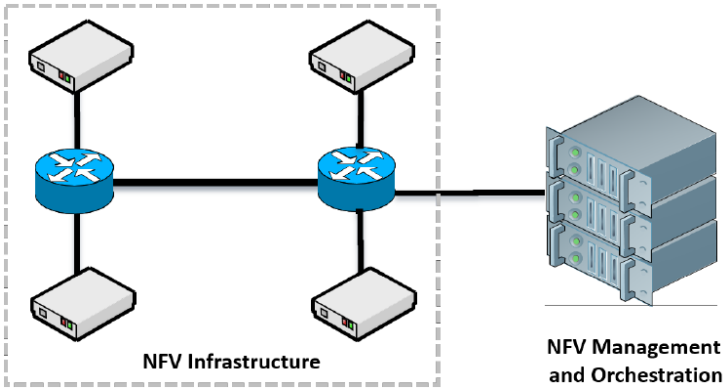


Figura 3.2: Infraestrutura utilizada no trabalho de Cristoffer Leite [3].

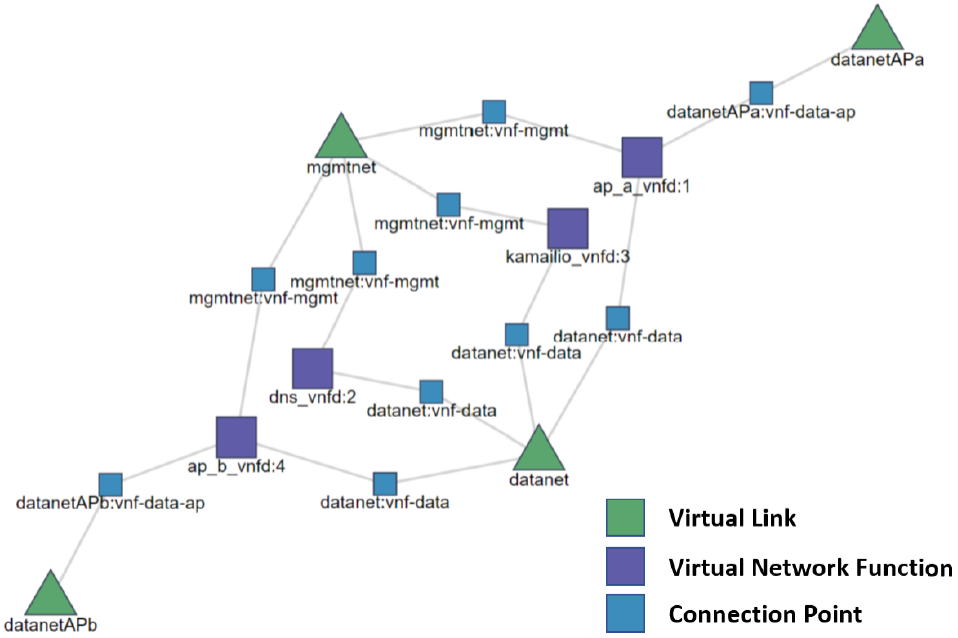


Figura 3.3: Cenário VoIP utilizado no trabalho de Cristoffer Leite [3].

de 26.93 kbps. A diferença entre os valores se deve pela natureza da carga de trabalho utilizada já que não foi a mesma, embora elas possuam as mesmas características. A carga de trabalho utilizada possui uma taxa média de envios de pacote a cada 0,02 segundos com o menor valor sendo 0,0039 segundos e o maior valor sendo 0,0337 segundos. Os resultados semelhantes mostram que o ambiente está funcionando corretamente e permite executar aplicações simples de cenários típicos do 5G.

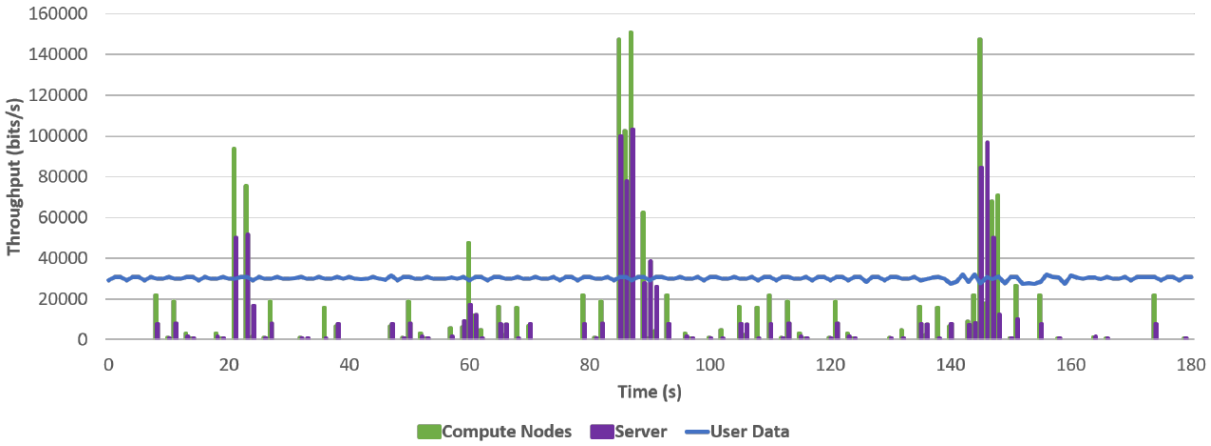


Figura 3.4: Resultados obtidos no trabalho de Cristoffer Leite [3] no cenário VoIP.

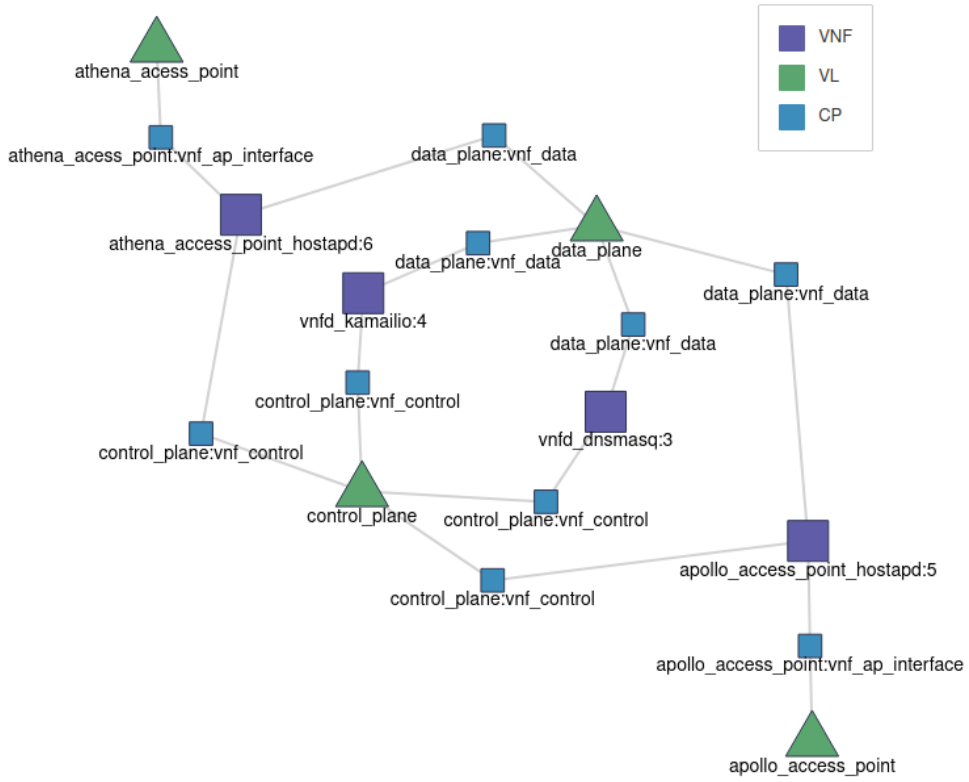


Figura 3.5: Cenário VoIP utilizado no ambiente e seu resultado na Figura 3.6.

3.3 Experimentos

As mensagens de controle são de fundamental importância no ambiente virtualizado, pois é por meio desta funcionalidade que o servidor de controle consegue descobrir ou ser notificado quando um nó de computação parou de responder ou não possui mais recursos. Na pesquisa realizada neste trabalho, não foi possível encontrar na documentação fornecida

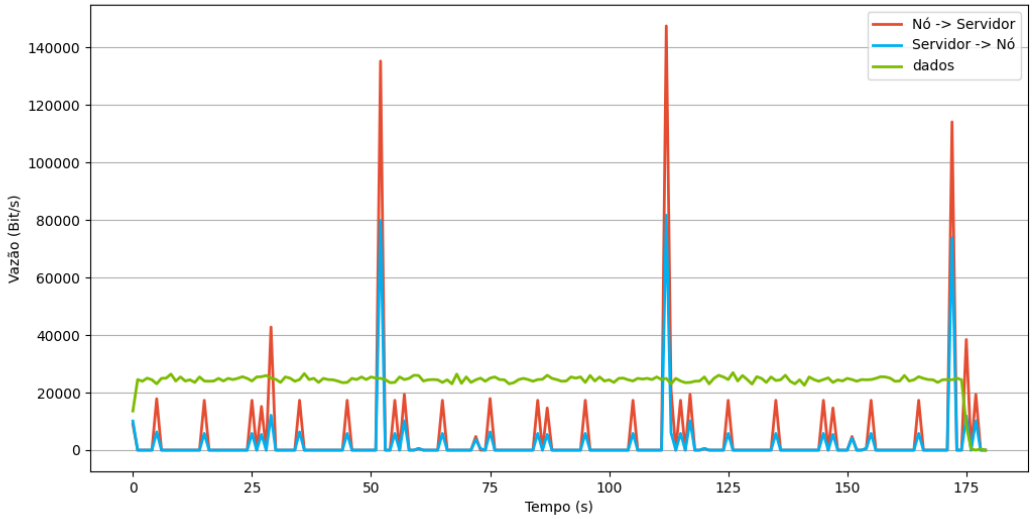


Figura 3.6: Resultado obtido em um cenário VoIP com uma ligação de 180 segundos.

pelo OpenStack um detalhe maior sobre as mensagens de controle e sua periodicidade. A informação técnica encontrada se restringe a que essas mensagens são chamadas remotas de procedimento (RPC).

Para melhor explorar e investigar a dinâmica das mensagens de controle no ambiente implementado, foram propostos dois cenários simples para capturar e compreender melhor os tipos de mensagens possíveis e mais dois cenários para verificar o impacto dessas mensagens no tráfego de controle, sendo eles respectivamente:

- **Cenário 1**: ambiente com um único nó de computação mas nenhuma VNF em execução;
- **Cenário 2**: ambiente com nó de computação e uma VNF executando.
- **Cenário 3**: variação da quantidade de VNFs.
- **Cenário 4**: variação da quantidade de nós de computação.

Para garantir que o máximo de mensagens fossem capturadas e também para facilitar na plotagem dos gráficos dos cenários acima, todos os serviços nova e neutron dos nós de computação eram parados. A captura era iniciada e, após isso, os serviços retomavam a sua execução. Isso era feito pois o inicio da comunicação do nó de computação com o servidor de controle era sempre iniciado pelo nó. Dessa maneira, todas as mensagens seriam capturadas já que não haveria tráfego de controle ocorrendo anteriormente. Essa ação de parar os serviços e depois continuar, causou um pico inicial maior na captura

e, consequentemente, nos gráficos. Isso se deve porque quando a comunicação do nó de computação e servidor de controle se inicia, vários parâmetros de comunicação como o nome da fila, o *exchange* a ser usado e outros parâmetros são definidos por causa da utilização do protocolo AMQP. Após as análises dos resultados dos quatro cenários, será feita uma tentativa de diminuir o tráfego de controle da infraestrutura ao alterar as configurações do ambiente virtualizado.

Como o foco do trabalho é sobre as mensagens de controle, a captura ocorrerá apenas no barramento sem fio da infraestrutura já que o barramento com fio sucede-se o tráfego de dados entre as máquinas virtuais. Além disso, as capturas terão uma duração de 200 segundos já que foi possível perceber na Figura 3.6 picos a cada 60 segundos, então esse período de captura aparenta ser um tempo razoável para a coleta de dados. E, finalmente, todas as capturas foram efetuadas na *interface* do servidor de controle. Na próxima seção são mostrados os resultados obtidos nos cenários propostos e feitas suas respectivas análises.

3.4 Resultados Experimentais

3.4.1 Cenário 1

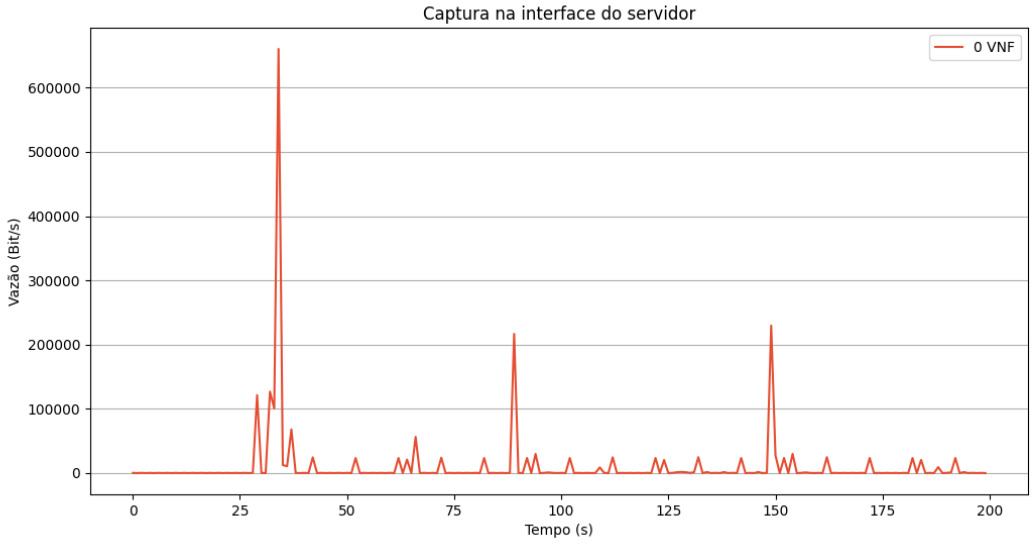


Figura 3.7: Tráfego de controle em um cenário com um nó de computação mas nenhuma VNF em execução.

Na Tabela 3.3 são resumidas as mensagens de controle que foram capturadas nesse cenário. É possível perceber que há uma grande quantidade de mensagens com ocorrências

Mensagem	Request frame (B)	Response frame (B)	Periodicidade (s)
get_compute_host	1.620	940	≈ 60
get_host	2.487 - 2.513	2.133 - 3.835	≈ 120
get_host_and_node	2.575	3.162	≈ 120
get_all_by_host	1.651	4.396 - 4.398	≈ 60
get_progress_by_node	1.572	540	≈ 60
sync_instance_info	1.318	-	≈ 120
save	2.013 - 2.993	507 - 547	≈ 10
refresh	3.032	421	≈ 60
get_by_uuid	2.492	3.640	≈ 60
get_by_instance_uuid	1.599	1.438	≈ 60
report_state	1.679 - 1.991	396 - 398	10-20

Tabela 3.3: Mensagens encontradas em um cenário com um nó de computação e nenhuma VNF executando

a cada 60 segundos, isso explica os picos observados na Figura 3.6 e Figura 3.7. Como explicado anteriormente, o pico por volta dos 30 segundos se deve pela inicialização da comunicação dos serviços nova e neutron no nó de computação.

Na Tabela 3.3, a função de cada mensagem de controle pôde ser determinada pelo seu conteúdo. As mensagens de *get_by_compute_host* até *get_all_by_host* são bastante similares, retornam informações sobre um nó de computação como a quantidade de memória de armazenamento disponível, arquitetura do processador e a utilização de outros recursos. A diferença entre estas mensagens é sobre a riqueza de detalhes fornecidos, da primeira mensagem até a mensagem *get_all_by_host* a quantidade de detalhes vai aumentando gradativamente. Sendo a última mensagem a maior de todas com um *frame* de tamanho médio de 4.396 Bytes. A mensagem *get_progress_by_host_node* refere-se a uma checagem se precisa ser feito uma migração em tempo de execução. Isso se deve porque o OpenStack permite que uma máquina virtual troque de hospedeiro sem interromper a sua execução. As outras mensagens são referentes ao serviço neutron, é por elas que é feito a sincronização entre as *interfaces* virtuais criadas para garantir conectividade as máquinas virtuais em uma mesma rede virtual. E, além disso, a mensagem *report_state* é o serviço neutron reportando sua situação como foi mostrado na Tabela 3.1.

Um exemplo da mensagem *get_all_by_host* que foi capturada nesse cenário pode ser vista abaixo. Pelo seu contéudo, é possível constatar que foi feito uma requisição sobre informações do nó de computação apollo (Raspberry Pi). Na mensagem consta a memória RAM disponível, a quantidade de máquinas virtuais que estão executando e o próprio IP da parte de controle do nó de computação. Essa é a maior mensagem que foi capturada, como pode ser visto na Tabela 3.3, e por causa disso ela foi mostrada de forma parcial.

```
{ "nova_object.namespace":"nova",
  "nova_object.data":{
     "objects":[
        {
            "nova_object.data":{
            "host":"apollo",
              "vcpus":4,
              "memory_mb":3791,
              "local_gb":28,
              "vcpus_used":2,
            "hypervisor_type":"QEMU",
              "hypervisor_hostname":"apollo",
              "free_ram_mb":2255,
              "free_disk_gb":28,
              "running_vms":2,
              "cpu_info":{
                 "arch":"aarch64",
            "host_ip":"10.0.10.30" }}}]}}
```


3.4.2 Cenário 2

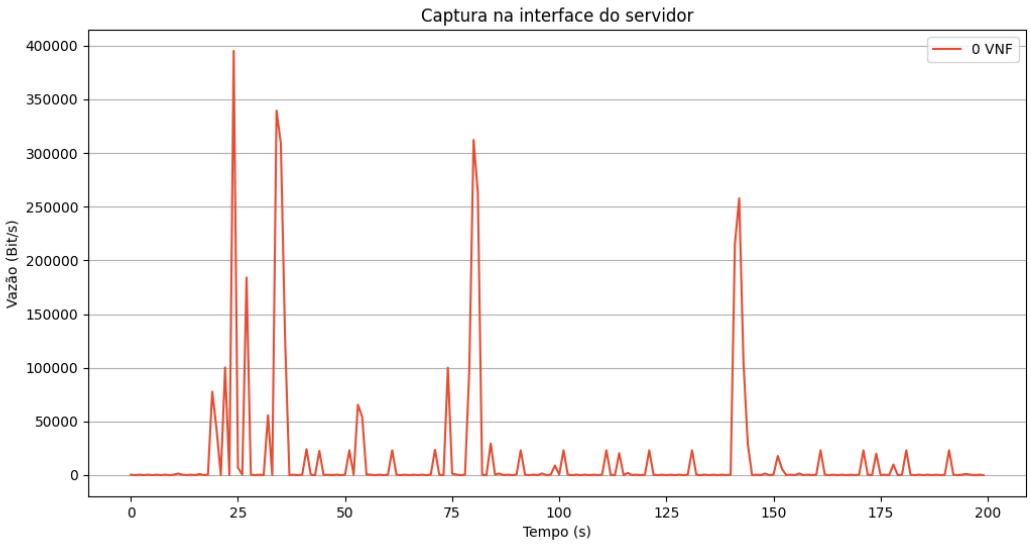


Figura 3.8: Tráfego de controle em um cenário com um nó de computação e uma VNF executando.

Mensagem	Request frame (B)	Response frame (B)	Periodicidade (s)
get_info_devices	1.308	452	1x
ping	1.393	444	1x
get_by_filters	1.658	1.497	1x
get_devices_details	1.375	1.026	1x
update_device_up	1.364	391	1x
pull	1.375	1.909	1x
clean_expired_console	1.527	391	$\approx$ 300

Tabela 3.4: Mensagens a mais encontradas em um cenário com um nó de computação e uma VNF executando

Na Tabela 3.4 são as mensagens a mais que foram encontradas nesse cenário. As mensagens da Tabela 3.3 também ocorreram nesse segundo cenário, mas para não ficar repetitivo, já que não foi possível perceber uma diferença no conteúdo ou na periodicidade, não foram colocadas. A mensagem *ping* é apenas uma checagem pra ver se a VNF conseguiu ser instanciada e está executando. A mensagem *get_by_filters* permite que o servidor de controle consiga filtrar informações por meio de filtros que são definidos na requisição, na captura foi percebido que estava sendo verificado se era necessário fazer migração de alguma máquina virtual para outro hospedeiro. As mensagens *security_group_info_for_devices*, *get_devices_details_list* e *update_device_up* são referentes a *bridge* que é criada e no qual a VM se conecta. Nas mensagens estava sendo verificado a situação da *bridge* e em qual rede virtualizada ela estava conectada. Além disso, era verificado as definições de políticas de segurança da *interface* e quais tipos de acesso podiam ser feitos, a porta 22 do SSH não é permitida por padrão por exemplo. A mensagem *pull* é refere-se a sincronização do tunelamento VXLAN já que o ambiente permite a criação de redes internas sem conexão com a *Internet*. E, finalmente, a mensagem *clean_expired_console_auths* que como o próprio nome já diz, remove consoles de autenticação que já tenham expirado, eles são utilizados para acessar a *VM* por meio do VNC (Virtual Network Computing).

Um exemplo da mensagem *refresh* que foi capturada nesse cenário pode ser vista abaixo. A mensagem é sobre o nó de computação apollo, no qual está sendo verificado as configurações da rede virtualizada *provider* em que as VMs se conectam. A *interface tapb7069647-60* é por onde a máquina virtual com o IP 10.0.20.124 se conectará na *bridge brq53345909-07*. Todas as máquinas virtuais que forem criadas no nó de computação apollo e forem conectadas na rede virtualizada *provider*, serão conectadas nessa *bridge* e o serviço neutron criará uma *interface* virtualizada para cada VM.

```
1 { "network_info": {
2     "network": {
```

```
        "id": "53345909-075c-46d4-8afb-1d0509771426",
        "bridge": "brq53345909-07",
        "label": "provider",
        "subnets": [
          {
            "cidr": "10.0.20.0/24",
            "gateway": {
              "address": "10.0.20.1",
              "ips": [
                {
                  "address": "10.0.20.124",
                  "type": "fixed",
                  "version": 4
                          "meta": {
                    "dhcp_server": "10.0.20.100"
                  },
                  "devname": "tapb7069647-60" }]}}]}}
```

3.4.3 Cenário 3

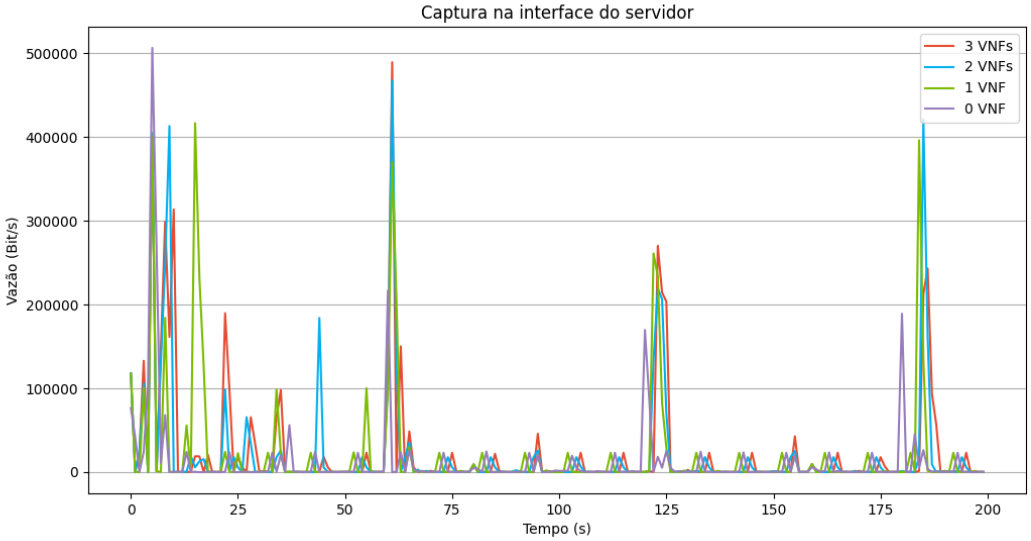


Figura 3.9: Cenário com a variação do número de VNFs.

Na Figura 3.9 é o resultado da variação da quantidade de VNFs no ambiente virtualizado. Pelo gráfico, não é possível perceber uma diferença significativa sobre o seu impacto

no tráfego de controle. A média da vazão na captura realizada com a quantidade de 0 VNF, 1 VNF, 2 VNFs e 3 VNF são, respectivamente, 11 kbps, 18 kbps, 18 kbps e 20 kbps. Isso é condizente com o que vendo observado nas mensagens de controle, as informações referentes as VNFs possuem uma pequena contribuição, apenas constando informações como o nome da VNF e a sua situação atual por exemplo. Dessa maneira, essa análise inicial leva a crer que a quantidade de VNF não afetam significativamente o tráfego de controle.

3.4.4 Cenário 4

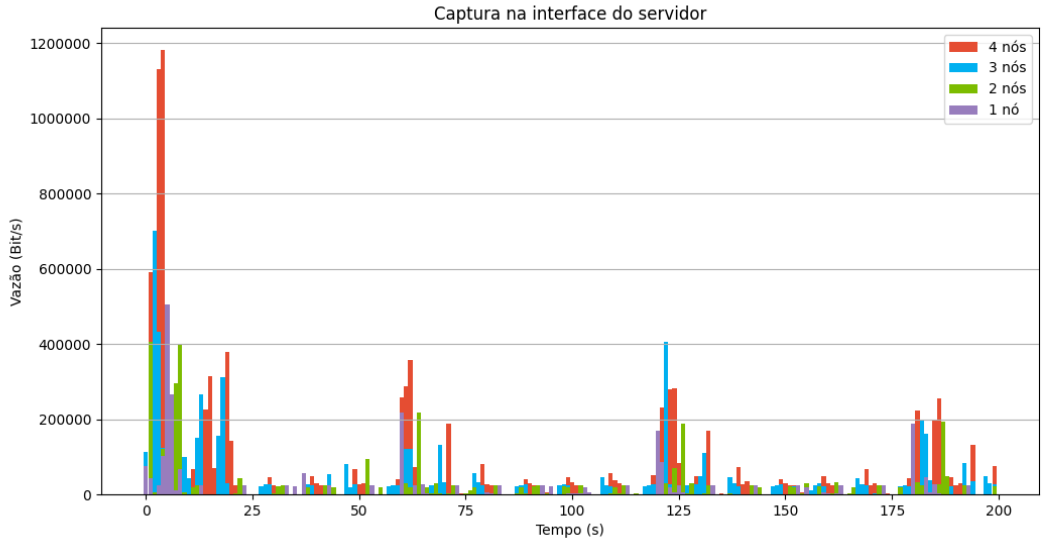


Figura 3.10: Cenário com a variação do número de nós de computação.

Na Figura 3.10 é mostrado o resultado obtido pela variação da quantidade de nós de computação na infraestrutura. Pelo gráfico, é possível perceber uma pequena proporção à medida que é adicionado nós de computação. A média da vazão na captura realizada com 1 nó, 2 nós, 3 nós e 4 nós são, respectivamente, 11 kbps, 16 kbps, 31 kbps e 46 kbps. Isso é condizente com o que vem sendo observado nas mensagens de controle pois a maioria das informações são referentes ao nós de computação como, por exemplo, a arquitetura do processador, quantidade de memória RAM e quantidade de memória de armazenamento. Além do mais, quanto maior a quantidade de nós de computação maior será a quantidade de ocorrência das mensagens das Tabelas 3.3 e 3.4.

3.4.5 Otimização do Tráfego de Controle

Na Figura 3.11 é mostrado a relação entre o *payload* da mensagem e os atributos de uso interno. Nas mensagens de requisição, ou seja, do nó de computação para o servidor de controle o conteúdo da mensagem tem uma contribuição muito pequena no tamanho total. Sendo os atributos internos do AMQP o grande responsável pelo tamanho da mensagens. Além disso, muitos desses atributos são nulos ou vazios e mesmo assim são enviados. Um fator curioso é que em algumas mensagens foi possível verificar o próprio nó de computação requisitando informações sobre si mesmo ao servidor. Esse comportamento é o esperado pois há uma checagem sobre quais informações o servidor de controle sabe sobre o nó de computação para que assim o nó reporte suas novas informações caso seja necessário.

Tendo sido observado o envio de atributos vazios ou nulos dentro da mensagem foi pensado que também poderia estar ocorrendo a repetição de outros atributos entre envios consecutivos de um mesmo tipo de mensagem. Para isso, algumas mensagens foram selecionadas e foi observado a parcela de atributos que eram repetidos e não repetidos utilizando o cenário da Figura 3.8, os resultados podem ser visto na Figura 3.12. Com essa comparação, fica claro que uma grande parcela do tráfego de controle se trata na verdade de mensagens repetidas e há espaço para uma certa otimização.

Levando em consideração o aumento do tráfego gerado pela adição de um nó de computação na infraestrutura e a quantidade de campos repetidos nas mensagens de controle, foi iniciada a próxima, e última, etapa do experimento que era elaborar considerações sobre estudar possíveis modificações que podem ser feitas na infraestrutura afim de diminuir o tráfego de controle gerado. A documentação do OpenStack é genérica nesse sentido, pois não mostrou a correlação entre a modificação de uma variável e sua consequência no ambiente. Depois de um estudo e observação no comportamento das mensagens de controle, foi possível verificar duas variáveis de configuração que possuem uma significativa contribuição para diminuir o tráfego de controle. Uma dessas variáveis é referente à frequência no qual o nó de computação reporta a situação de seus recursos para o servidor de controle e a outra é a utilização do algoritmo de compressão gzip nas mensagens de controle.

Na primeira variável o nó de computação só reportará sua situação caso ocorra um evento que mude o seu estado, por exemplo uma VNF começar a executar naquele nó. A segunda variável é a aplicação do algoritmo de compressão gzip nas mensagens. Havia também a opção de utilizar o algoritmo bz2, mas o mesmo se encontrava como experimental e, com isso, não foi possível testar essa possibilidade. Os resultados obtidos podem ser conferidos na Figura 3.13. Mesmo havendo uma diminuição nó tráfego de controle, o resultado não foi tão satisfatório. Como pode ser visto na Figura 3.12, foi analisado algumas mensagens considerando o cenário de uma VNF executando no cenário Figura 3.8,

Figura 3.11: Comparação entre o *payload* da mensagem e atributos de uso interno.

e foi possível perceber uma grande repetição dos atributos de uma mesma mensagem ao longo do tempo da execução do cenário. Levando isso em consideração, a compressão com o gzip estaria apenas aliviando o problema e não o solucionando já que a natureza da repetição das mensagens de controle é temporal e não espacial, ou seja, há uma pequena diferença entre mensagens consecutivas do mesmo tipo.

3.5 Resumo do capítulo

Nesse capítulo foi apresentada a descrição do ambiente virtualizado que foi construído como resultado da compreensão do padrão ETSI e suas ferramentas. Após isso, foi feito uma comparação com a infraestrutura utilizada em outro trabalho no qual tinha-se características semelhantes. Logo após, foi avaliado nesse ambiente o comportamento de uma aplicação VoIP e os seus resultados foram comparados com um trabalho similar. Dessa maneira, foi possível validar a infraestrutura construída neste trabalho. É importante

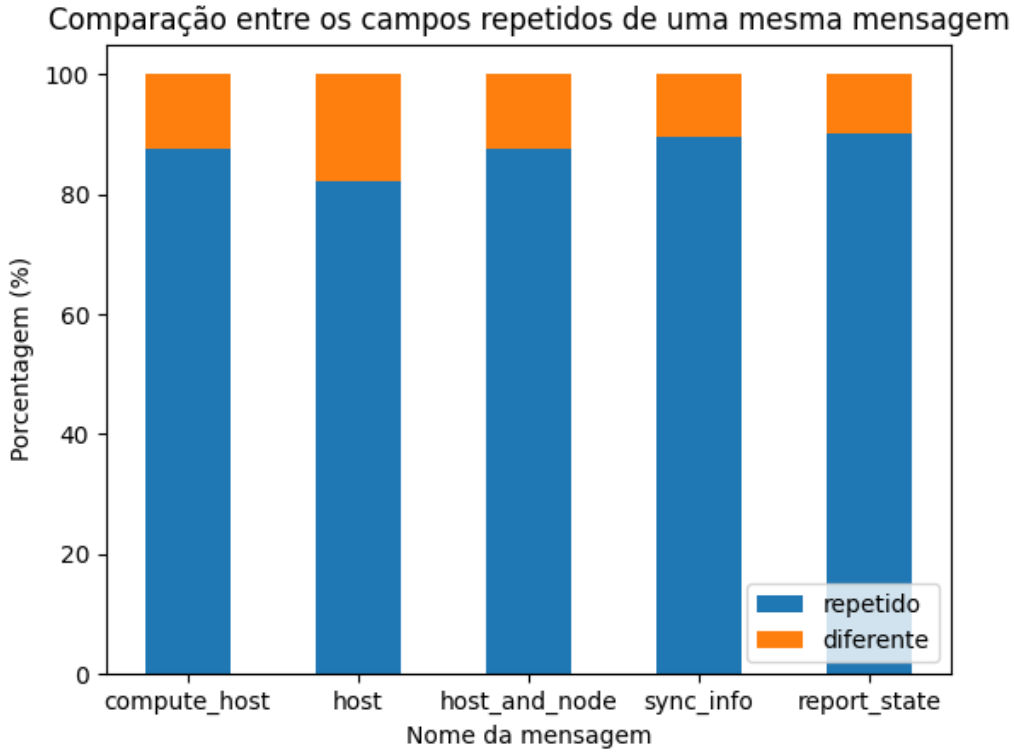


Figura 3.12: Comparação entre os campos repetidos de uma mesma mensagem do cenário da Figura 3.8 ao longo da captura.

destacar que as aplicações VoIP são consideradas parte do cenário URLLC, típico das redes 5G.

Após a validação e ter uma maior confiança da corretude do ambiente desenvolvido, foi feita uma análise do tráfego de controle da infraestrutura. Essa etapa foi composta por identificar as mensagens em si e entender a sua finalidade. Além disso, também foi possível entender o efeito que a variação da quantidade de nós de computação e a quantidade de VNFs acarretariam no tráfego de controle.

Finalmente, o experimentou avaliou a possível diminuição do tráfego de controle da infraestrutura ao alterar as configurações do ambiente virtualizado. Foi possível alcançar uma quantidade de tráfego de controle menor, mas que não é ainda satisfatória já que não foi possível eliminar as mensagens repetidas enviadas em um curto prazo de tempo. Essa grande quantidade de conteúdo repetido entre as mensagens pode ser um item de interesse em um futuro procedimento de otimização do tráfego de controle.

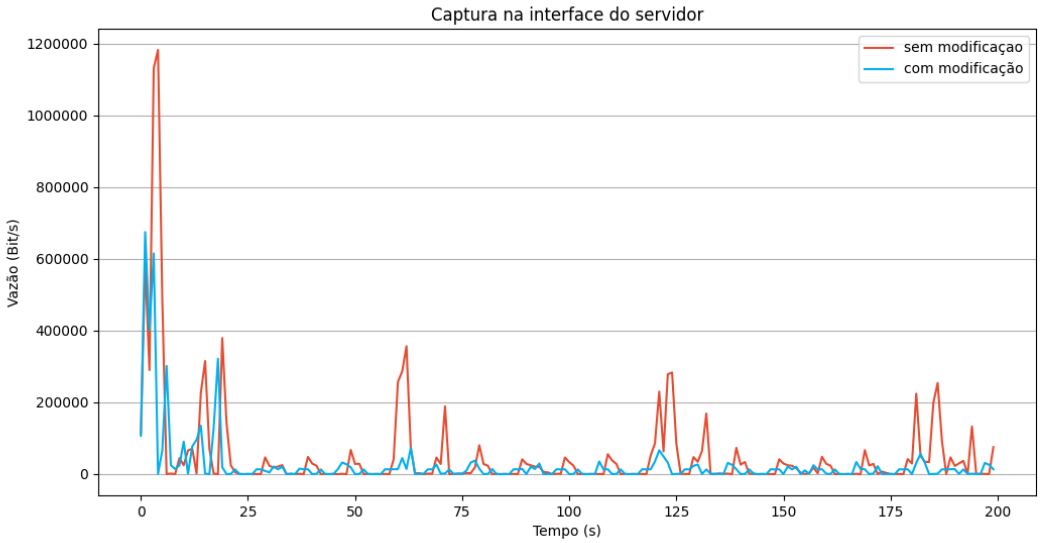


Figura 3.13: Comparação entre um ambiente com algoritmo de compressão gzip nas mensagens de controle e alteração na frequência das mensagens com um sem nenhuma modificação.

Capítulo 4

Conclusões e Trabalhos Futuros

Openstack é uma solução em nuvem modularizada e completa podendo ser adaptada para utilização em diversos cenários. Pode-se concluir que por causa da sua versatilidade de utilização em diversos cenários, fica esclarecido porque continua sendo objeto de vários estudos em diversas áreas. Além disso, a sua versatilidade é tanta que graças a isso foi possível integra-lá a outra ferramenta utilizada nesse projeto o OSM. No trabalho desenvolvido nesse documento, ambas foram fortes aliadas que possibilitaram a criação do ambiente virtualizado para redes 5G.

No desenvolvimento do trabalho foi possível aprender mais sobre o padrão de tecnologia móvel 5G e as inovações que estão sendo propostas no seu núcleo de rede. Para tal, foi implementado e validado um ambiente virtualizado que segue as especificações técnicas da ETSI em relação a arquitetura NFV MANO. Nesse ambiente o objetivo se concentrou em analisar o tráfego de controle, conforme é variada a quantidade de nós de computação e funções de rede. No ambiente foram avaliadas funções de rede virtualizada e foi possível recriar um cenário VoIP, sendo esse um caso tipico das redes 5G, em dispositivos não especializados ou proprietários.

Os resultados experimentais do trabalho permitem concluir que o tráfego de controle gerado pela infraestrutura possui espaço para melhorias já que boa parte do conteúdo dessas mensagens é repetido ou redundante ao longo do tempo. Ainda mais, foi avaliado de que é possível efetuar ajustes no ambiente para que menos tráfego possa ser gerado. Entretanto, esse ajuste deve ser melhor explorado e avaliado em diversos cenários já que a documentação disponível não permite entender a correlação entre a mudança de uma variável e sua consequência na infraestrutura.

4.1 Trabalhos futuros

Como opções de trabalhos futuros, uma possibilidade interessante é a criação de uma ferramenta que se integre à infraestrutura e possa efetuar uma compressão inteligente de dados do tráfego de controle. Ainda mais, considerando que há uma grande quantidade de informações repetidas ao decorrer de mensagens consecutivas, cria-se uma situação ideal para a utilização de algoritmos de compressão delta [36]. A quantidade de dados a ser reduzida pode ser muito significativa, além de viabilizar a integração de outras tecnologias aos cenários 5G, por exemplo, antenas de baixa potência e grande cobertura como LoRa e Sigfox.

Outra opção de trabalho futuro também pode ser a automatização na identificação e adaptação dos nós de computação em relação ao período das mensagens. Como foi avaliado neste trabalho, o OpenStack na configuração padrão possui uma incidência de mensagens em tempos múltiplos de 60 segundos. Isso ocasiona grandes picos no tráfego de rede que poderia ser evitado se houvesse uma organização entre os nós de computação de perceber uma grande quantidade de mensagens ocorrendo e assim retardar o envio da mensagem para não congestionar o canal.

Referências

[1] Andrew S. Tanenbaum, Herbert Bos: *Sistemas Operacionais Modernos.* Pearson Education, 4ª edição, 2016, ISBN 9788543018188. ix, 4, 5, 6

[2] Amazon: *Aquisição de soluções em nuvem: melhores práticas para clientes do setor público*, março 2015. https://forum.ibgp.net.br/wp-content/uploads/2017/05/Aquisi%C3%A7%C3%A3o_de_solu%C3%A7%C3%B5es_em_nuvem_melhores_pr%C3%A1ticas_para_clientes_do_setor_....pdf, acesso em 2021-05-19. ix, 7

[3] SILVA, Cristoffer Leite da: *A performance evaluation model for network function virtualisation on 5g networks*, 2019. https://repositorio.unb.br/handle/10482/38106, acesso em 2021-05-01. ix, 23, 24, 25

[4] Morgado, António, Kazi Mohammed Saidul Huq, Shahid Mumtaz e Jonathan Rodriguez: *A survey of 5g technologies: regulatory, standardization and industrial perspectives.* Digital Communications and Networks, 4(2):87–97, 2018, ISSN 2352-8648. https://www.sciencedirect.com/science/article/pii/S2352864817302584. 1

[5] Vora, Lopa J.: *Evolution of mobile generation technology: 1g to 5g and review of upcoming wireless technology 5g*, 2015. https://ijmter.com/papers/volume-2/issue-10/evolution-of-mobile-generation-technology-1g-to-5g-and-review-of-5g.pdf, acesso em 2021-05-20. 1

[6] Intelligence, GSMA: *The mobile economy 2018*, 2018. https://www.gsma.com/asia-pacific/resources/the-mobile-economy-report-2018/, acesso em 2021-05-20. 1, 7

[7] Brown, Gabriel: *Service-based architecture for 5g core networks*, novembro 2017. https://www-file.huawei.com/-/media/corporate/pdf/white%20paper/heavy-reading-whitepaper--service-oriented-5g-core-networks.pdf, acesso em 2021-05-06. 1, 14

[8] Redana, Simone, Ömer Bulakci, Christian Mannweiler, Laurent Gallo, Apostolos Kousaridas, David Navrátil, Anna Tzanakaki, Jesús Gutiérrez, Holger Karl, Peer Hasselmeyer, Anastasius Gavras, Stephanie Parker e Edward Mutafungwa: *5G PPP Architecture Working Group - View on 5G Architecture, Version 3.0*, junho 2019. https://doi.org/10.5281/zenodo.3265031, acesso em 2021-05-08. 1, 12, 13

[9] Paper, NFV White: *Network functions virtualisation: An introduction, benefits, enablers, challenges & call for action. issue 1.* outubro 2012. 2, 16

[10] Sanchez-Aguero, Victor, Ivan Vidal, Francisco Valera, Borja Nogales, Luciano Leonel Mendes, Wheberth Damascena Dias e Alexandre Carvalho Ferreira: *Deploying an nfv-based experimentation scenario for 5g solutions in underserved areas*. Sensors, 21(5), 2021, ISSN 1424-8220. https://www.mdpi.com/1424-8220/21/5/1897. 2

[11] Giannopoulos, Dimitris, Christos Tranoris, Panagiotis Papaioannou e Spyros Denazis: *Monitoring as a service over a 5g network slice*. abril 2021. 2

[12] Nogales, Borja, Victor Sanchez-Aguero, Ivan Vidal e Francisco Valera: *Adaptable and automated small uav deployments via virtualization*. Sensors, 18(12), 2018, ISSN 1424-8220. https://www.mdpi.com/1424-8220/18/12/4116. 2

[13] Rose, Robert J.: *Survey of system virtualization techniques*. 2004. 4

[14] RedHat, 2011. https://www.linux-kvm.org/page/Main_Page, acesso em 2021-05-23. 5

[15] 2003. https://xenproject.org/, acesso em 2021-05-23. 5

[16] 2001. https://www.vmware.com/products/esxi-and-esx.html, acesso em 2021-05-23. 5

[17] 2007. https://www.virtualbox.org/, acesso em 2021-05-23. 5

[18] 1999. https://www.vmware.com/br/products/workstation-pro.html, acesso em 2021-05-23. 5

[19] Kumar, Rakesh, Neha Gupta, Shilpi Charu, Kanishk Jain e Sunil Jangir: *Open source solution for cloud computing platform using openstack*. maio 2014. 6

[20] William, Bryan: *Openstack cloud computing platform*. https://www.nasa.gov/offices/oct/40-years-of-nasa-spinoff/openstack-cloud-computing-platform, acesso em 2021-04-27. 8

[21] *Introduction: A bit of openstack history*. https://docs.openstack.org/project-team-guide/introduction.html, acesso em 2021-04-27. 8

[22] F. F. Moges, S. L. Abebe: *Energy-aware vm placement algorithms for the openstack neat consolidation framework*. Journal of Cloud Computing, janeiro 2019. https://doi.org/10.1186/s13677-019-0126-y, acesso em 2021-05-10. 8

[23] Elia, Ivano Alessandro, Nuno Antunes, Nuno Laranjeiro e Marco Vieira: *An analysis of openstack vulnerabilities*. Em *2017 13th European Dependable Computing Conference (EDCC)*, páginas 129–134, 2017. 8

[24] *Keystone architecture*. https://docs.openstack.org/keystone/victoria/getting-started/architecture.html, acesso em 2021-04-27. 9

[25] *Glance basic architecture*. https://docs.openstack.org/glance/train/contributor/architecture.html, acesso em 2021-04-27. 9

[26] *Placement architecture.* https://docs.openstack.org/placement/train/contributor/architecture.html, acesso em 2021-04-27. 9

[27] *Compute service overview.* https://docs.openstack.org/nova/train/install/overview.html, acesso em 2021-04-27. 10

[28] *Networking architecture.* https://docs.openstack.org/security-guide/networking/architecture.html, acesso em 2021-04-27. 10

[29] Ramos, Fernando, Diego Kreutz e Paulo Veríssimo: *Software-defined networks: On the road to the softwarization of networking.* 28:6–13, maio 2015. 15

[30] ETSI: *Network functions virtualisation – introductory white paper.* https://portal.etsi.org/NFV/NFV_White_Paper.pdf, acesso em 2021-05-01. 16

[31] *Openstack installation guide.* https://docs.openstack.org/install-guide, acesso em 2021-05-03. 20

[32] *Installing osm.* https://osm.etsi.org/docs/user-guide/01-quickstart.html#installing-osm, acesso em 2021-05-03. 20

[33] Rafael Amaral, Gabriel Ferreira: *Eros-5 - gerador sintético de carga de trabalho para ambientes 5g*, 2021. https://github.com/notopoloko/Eros, acesso em 2021-05-22. 23

[34] 2001. https://www.kamailio.org/w/, acesso em 2021-05-23. 23

[35] Kelley, Simon, 2001. https://thekelleys.org.uk/dnsmasq/doc.html, acesso em 2021-05-23. 23

[36] Suh, Young Soo: *Send-on-delta sensor data transmission with a linear predictor.* Sensors, 7(4):537–547, 2007, ISSN 1424-8220. https://www.mdpi.com/1424-8220/7/4/537. 38