Tiago Schoeping Reinert

Análise e Caracterização do Tráfego na Rede de Controle de Nuvens Computacionais Baseadas em OpenStack Com Auxílio de um Sistema de Monitoramento

Joinville

2017

UNIVERSIDADE DO ESTADO DE SANTA CATARINA
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

Tiago Schoeping Reinert

ANÁLISE E CARACTERIZAÇÃO DO TRÁFEGO NA REDE DE CONTROLE DE NUVENS COMPUTACIONAIS BASEADAS EM OPENSTACK COM AUXÍLIO DE UM SISTEMA DE MONITORAMENTO

Trabalho de conclusão de curso submetido à Universidade do Estado de Santa Catarina como parte dos requisitos para a obtenção do grau de Bacharel em Ciência da Computação

Charles Christian Miers
Orientador

Joinville, Novembro de 2017

ANÁLISE E CARACTERIZAÇÃO DO TRÁFEGO NA REDE DE CONTROLE DE NUVENS COMPUTACIONAIS BASEADAS EM OPENSTACK COM AUXÍLIO DE UM SISTEMA DE MONITORAMENTO

Tiago Schoeping Reinert

Este Trabalho de Conclusão de Curso foi julgado adequado para a obtenção do título de Bacharel em Ciência da Computação e aprovado em sua forma final pelo Curso de Ciência da Computação Integral do CCT/UDESC.

Banca Examinadora

Charles Christian Miers - Doutor (orientador)

Rafael Rodrigues Obelheiro - Doutor

Mauricio Aronne Pillon - Doutor

Agradecimentos

Gostaria de agradecer o Laboratório de Pro cessamento Paralelo e Distribuído(LabP2D)/UDESC, por disponibilizar sua instalação e os recursos necessários para realização deste trabalho.

Resumo

Os softwares para gerenciamento de nuvens computacionais apresentam uma ampla adoção e possuem arquiteturas voltadas para escalabilidade e segurança dos dados. Dentre as opções de solução para nuvens computacionais privadas, o OpenStack é destacadamente a solução mais usual. Contudo, como qualquer solução de nuvem computacional, um dos aspectos considerados cruciais em qualquer nuvem é seu desempenho. Neste contexto, a maioria das pesquisas concentra-se na parte das nuvens visíveis aos usuários, relegando a um plano secundário as operações internas do provedor. O OpenStack utiliza um agrupamento de redes exclusivamente para tráfego de controle, que contém todas as tarefas administrativas e operacionais, denominado Domínio de Controle. Entretanto, pouco se conhece sobre a influência no comportamento do Domínio de Controle do OpenStack originária de eventos gerados pelo usuário (*i.e.,* alocação de *Virtual Machines* (VMs)) e de eventos periódicos (*i.e.,* atualização da lista de VMs ativas). Este trabalho tem como objetivo caracterizar o tráfego de controle em uma nuvem OpenStack com o auxílio de um sistema de monitoramento, objetivando identificar e compreender o comportamento da rede de controle, que está contida neste domínio. Para alcançar este objetivo, será realizada uma análise de métodos para medição e análise de tráfego, assim como uma análise sobre funcionamento do OpenStack voltada à sua arquitetura de funcionamento. Após, pretende-se projetar e implementar um sistema de monitoramento, que levantará informações sobre o tráfego na rede de controle periodicamente. A partir dos dados coletados será feita uma análise do tráfego, que será voltada para alguns serviços presentes no tráfego da rede de controle do OpenStack.

Palavras-chaves: Computação em nuvem, Caracterização de tráfego, Rede de controle, Sistema de monitoramento de tráfego, Análise de desempenho, OpenStack.

Abstract

Cloud computing management softwares are popular, having both deployment scalability and data security as their main characteristics. Among all available private cloud management softwares, OpenStack stands among the most popular ones. However, like in any other cloud management software, performance is considered one of the most important metrics. In this context, most researches focus only on analysing the cloud section visible to users, relinquishing the internal operations and behavior of the cloud provider. OpenStack has a network group specifically for management traffic, named Management Domain, which contains all the operational and administrative tasks of the cloud. However, there is a lack of information regarding how both user generated tasks (*i.e.,* VM initialization) and periodic tasks (*i.e.,* update active VM list) may impact the behavior of the Management Domain in OpenStack clouds. This work aims to understand the OpenStack management network behavior better, which is a network contained inside the Management Domain. We will achieve it by characterizing the management traffic through an analysis approach using the data provided by a network monitoring software. To reach the set goal, we start studying traffic measurement / analysis approaches, as well as OpenStack software focusing on its deployment architecture. Thus, we specify, design, and deploy a traffic monitoring system, which will be responsible for generating data about the traffic in the management network. Finally, the traffic analysis will focus on key OpenStack services, using the data generated by the traffic monitoring system as input.

Keywords: Cloud computing, Traffic characterization, Management network, Traffic monitoring system, Performance analysis, OpenStack.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas

AMQP *Advanced Message Queuing Protocol*

API *Application Programming Interface*

DDoS *Distributed Denial of Service*

IaaS *Infrastructure as a Service*

IDS *Intrusion Detection System*

KVM *Kernel-based Virtual Machine*

LDAP *Lightweight Directory Access Protocol*

NIST *National Institute of Standards and Technology*

PaaS *Platform as a Service*

RPC *Remote Procedure Call*

SaaS *Software as a Service*

SNMP *Simple Network Management Protocol*

VLAN *Virtual Local Area Network*

VM *Virtual Machine*

VPN *Virtual Private Network*

1 Introdução

Tanto questões de segurança como de análise de desempenho de uma nuvem computacional dependem da identificação de comportamentos e operações que ocorrem durante o seu uso. Percebe-se que há uma necessidade de identificar operações, muitas das quais são operações que ocorrem no nível da camada de rede e demandam análise e compreensão do que está trafegando e sua finalidade. Neste sentido, a caracterização de tráfego auxilia através do emprego de técnicas e métodos que possibilitam a coleta e identificação de forma sistematizada. Apesar de não haver uma definição formal ou método para caracterização de tráfego, este termo refere-se à minuciosa análise do tráfego de uma rede, buscando entender seu comportamento e também as consequências que derivam deste comportamento.

A caracterização de tráfego em nuvens computacionais ainda é incipiente em alguns aspectos enquanto em outros recai sobre questões já amplamente estudadas. Quando se trata de caracterização de tráfego de aplicações tradicionais (*e.g.,* servidor web) já existem trabalhos bem detalhados (ARLITT; WILLIAMSON, 1996; BRAUN; CLAFFY, 1995; GILL et al., 2007). Entretanto, quanto ao tráfego de controle torna-se escasso em função das especificidades de cada nuvem computacional.

Uma nuvem computacional privada opera na infraestrutura própria de uma organização, portanto, toda manutenção e cuidados com segurança são de responsabilidade dela. Em questão de uso, nuvens privadas são criadas para atender os propósitos da organização, e são acessíveis apenas por indivíduos habilitados pela organização, o que possibilita um grau maior de controle e segurança sobre os dados ali contidos (JADEJA; MODI, 2012).

Dentre os *softwares* de código aberto existentes para gerenciamento e virtualização de nuvens computacionais *Infrastructure as a Service* (IaaS), que permitem a criação de nuvens privadas, destaca-se o OpenStack[1] pela sua popularidade. Em relação à parte física da instalação de uma nuvem com OpenStack, a arquitetura possui três domínios com políticas de seguranças distintas: Domínio Público, Domínio de Controle e

[1] `https://www.openstack.org`

Domínio de Convidados (The OpenStack project, 2017g). O Domínio de Controle é uma das partes que define o desempenho de nuvens computacionais OpenStack, justamente por trafegar dados de funcionalidades essenciais ao funcionamento da nuvem. Consequentemente, o estudo do tráfego neste domínio envolve identificar possíveis fatores que limitem seu funcionamento e também identificar limitações intrínsecas do software.

Este trabalho pretende analisar e caracterizar o tráfego da rede neste domínio, levantando questões de desempenho presentes no **Domínio de Controle** com o auxílio de um sistema de monitoramento. Mais especificamente, este trabalho pretende analisar o desempenho na rede de controle do OpenStack sob diferentes aspectos: comportamento dos serviços durante instanciação de VM; classificar o tráfego recebido pelos nós controladores da nuvem; e analisar a presença e impacto de eventos periódicos. Para auxiliar o processo de caracterização do tráfego, será implementado um sistema de monitoramento de tráfego para alguns serviços do OpenStack.

Como estudo de caso, será realizado o monitoramento da rede de controle de uma nuvem computacional baseada em OpenStack. O tráfego coletado servirá para caracterizar o comportamento de certas funcionalidades do OpenStack na rede de controle, e como elas influenciam no desempenho desta rede. Em relação à este trabalho, os objetivos específicos são:

- Identificar e analisar métodos mais adequados à caracterização de tráfego, sob a ótica de nuvens computacionais;
- Identificar e analisar abordagens para o monitoramento de tráfego que possibilitem monitorar redes de controle em nuvens computacionais baseadas em OpenStack;
- Especificar uma ferramenta para monitoramento do tráfego da rede de controle em nuvens computacionais baseadas em OpenStack, para auxiliar na sua caracterização;
- Implementar um sistema de monitoramento para redes de controle em nuvens computacionais baseadas em OpenStack;
- Aplicar o sistema de monitoramento implementado em uma rede de controle de uma nuvem computacional baseada em OpenStack; e
- Analisar e caracterizar o comportamento de funcionalidades selecionadas no tráfego da rede, levantando questões de desempenho.

Foi adotada a pesquisa aplicada como método de pesquisa neste trabalho. Dessa forma, na primeira parte (TCC-I) do trabalho foi realizado levantamento bibliográfico, a análise de trabalhos relacionados, a definição da arquitetura de funcionamento do sistema de monitoramento e a definição dos experimentos que serão realizados. Na segunda parte (TCC-II) do trabalho será realizado um estudo de caso aplicando o sistema de monitoramento implementado na nuvem TCHE, localizada no LabP2D do CCT/UDESC, com o objetivo de coletar dados para realizar a caracterização do tráfego da rede de controle da nuvem em questão.

O trabalho organiza-se da seguinte forma. O Capítulo 2 apresenta a fundamentação teórica sobre o tema proposto, contendo uma introdução à computação em nuvem e virtualização, e conceitos básicos sobre software de gerenciamento de nuvem OpenStack. Após, são apresentadas abordagens para efetuar a caracterização de tráfego. Com todas as definições básicas feitas, é apresentado o problema abordado neste TCC, bem como trabalhos relacionados. O Capítulo 3 contém a proposta para realização da caracterização de tráfego. Para tal, explica-se em maior detalhes os serviços do OpenStack que este trabalho tem interesse ao caracterizar o tráfego. Com estas características definidas, detalha-se a proposta do sistema de monitoramento de tráfego e como será realizada a análise, e então define-se como serão realizados os experimentos na qual a proposta será aplicada. Por fim, o Capítulo 4 apresenta as considerações desta etapa (TCC-I) e estabelece as metas que serão desenvolvidas nos trabalhos futuros.

2 Fundamentação Teórica

O conceito de computação em nuvem (Seção 2.1) embora tenha vários aspectos amplamente conhecidos, normalmente possui as suas especificações desconhecidas para o grande público. Neste sentido, torna-se necessário entender os modelos de serviços, modelos de implantação e uma arquitetura de referência. A implantação de uma nuvem emprega tecnologias de virtualização, que apesar de não serem uma tecnologia recente, é parte comum à softwares gerenciadores de nuvem, e portanto, a compreensão do comportamento de uma nuvem computacional está diretamente ligado à entender sobre o processo de virtualização.

Uma das aplicações da computação em nuvem é a consolidação de servidores, que consiste na instalação de um software responsável por gerenciar nuvens computacionais, que no caso de nuvens IaaS um dos softwares mais populares é o OpenStack (Seção 2.2). O processo de consolidação deve levar em conta a arquitetura necessária para a instalação do OpenStack, ou seja, aprender sobre as funcionalidades oferecidas pelo software, como elas se comportam, e também como organizar a infraestrutura física é importante para garantir seu funcionamento. Para entender o comportamento de uma nuvem computacional, uma análise minuciosa dos seus componentes torna-se necessária. Sendo assim, para entender, por exemplo, o comportamento em uma das redes na qual o OpenStack opera, a caracterização de tráfego (Seção 2.3) é um processo que emprega técnicas que compreendem a medição do tráfego, ou seja, coleta do tráfego, e a posterior análise dele, baseando-se preferencialmente em abordagens já existentes para ambos.

Após conhecer os conceitos básicos sobre nuvens computacionais e como a efetuar a caracterização de tráfego em uma rede pertencente à esta nuvem, é possível estudar sobre um ambiente na qual esta técnica é aplicável (Seção 2.4). Por fim, para concretizar o processo de monitoramento e caracterização de tráfego em nuvens computacionais, torna-se necessário apresentar trabalhos com objetivos similares (Seção 2.5), exibindo exemplos de métodos, métricas e ferramentas empregadas no processo.

2.1 Computação em Nuvem

A computação em nuvem é um paradigma para utilização de infraestrutura computacional. Dentre as várias definições existentes sobre computação em nuvem, a definição do NIST (MELL; GRANCE, 2011) é a mais próxima de uma padronização e possui ampla adoção. O documento, por sua vez, define características intrínsecas a qualquer nuvem computacional e fornece algumas classificações relacionadas ao uso da tecnologia. De acordo com Mell e Grance (2011), a computação em nuvem é um modelo de computação que possibilita acesso conveniente, ubíquo e escalável à um conjunto de recursos computacionais (*e.g.*, rede, armazenamento, aplicações, plataformas e serviços) que podem ser rapidamente provisionados e liberados com o mínimo de gerenciamento ou de interação com o provedor de serviço.

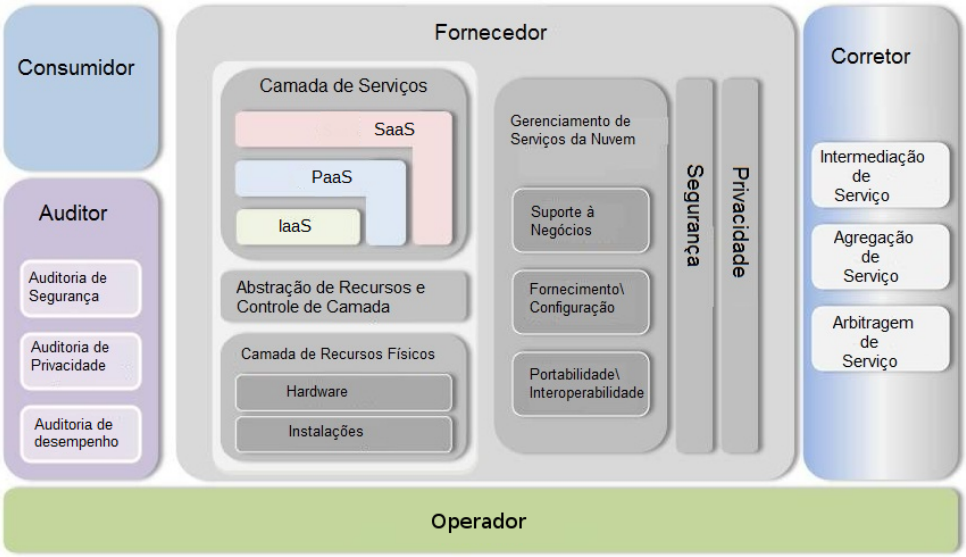
Seguindo a definição do NIST, a oferta de serviços de computação em nuvem são classificados em três categorias: *Infrastructure as a Service* (IaaS), *Platform as a Service* (PaaS), *Software as a Service* (SaaS). Estas categorias variam respectivamente, de mais próximo do hardware, com maior responsabilidade e controle do consumidor, até mais alto nível, na qual o serviço é ofertado como uma aplicação. O IaaS fornece ao consumidor a capacidade de provisionar processamento, armazenamento, rede e outros recursos computacionais na qual o consumidor é capaz de utilizar software, incluindo sistemas operacionais e consegue configurar alguns componentes de rede e armazenamento, e retém controle sobre o software na VM. No PaaS é fornecido ao consumidor uma plataforma para desenvolvimento, que contém um conjunto de linguagens e bibliotecas configuradas. O consumidor tem controle sobre as aplicações instaladas por ele, e controle parcial sobre configurações do ambiente no qual as aplicações foram instaladas. O modelo SaaS fornece ao consumidor acesso às aplicações executando sobre a infraestrutura da nuvem computacional, e tem controle apenas sobre configurações específicas de cada consumidor naquela aplicação.

Nuvens computacionais também são classificadas de acordo com o seu modelo de implantação, que define as políticas de uso dos recursos da infraestrutura da nuvem e por quem ela é gerenciada. De acordo com NIST, existem quatro modelos de implantação: nuvem privada, cuja infraestrutura da nuvem é de uso exclusivo de uma única organização, e pode é gerenciada por ela ou terceiros; Nuvem comunitária, na qual a infraestrutura da nuvem é de uso exclusivo de uma comunidade específica de consumidores, composta

por organizações com necessidades em comum, e é gerenciada por esta comunidade ou/e terceiros; Na nuvem pública a infraestrutura da nuvem é de uso aberto, e é gerenciada por uma empresa, organização governamental, organização acadêmica, ou uma combinação deles; A nuvem híbrida tem a infraestrutura composta por duas ou mais infraestruturas de nuvens distintas (pública, comunitária, privada), que continuam sendo entidades únicas, mas são interligadas por tecnologias que permitem a troca de dados e portabilidade de aplicações (*e.g.,* balanceamento de carga entre nuvens distintas).

Independente do modelo de implantação de uma nuvem computacional diversas partes estarão envolvidas, também chamadas de atores, que exercem diferentes atividades e funções dentro da nuvem (BOHN et al., 2011). A Figura 2.1 exibe um diagrama de alto nível criado pelo NIST, contendo os principais atores envolvidos junto de uma arquitetura de nuvem computacional.

Figura 2.1: Arquitetura de referência do NIST para computação em nuvem.



Fonte: (BOHN et al., 2011).

Segundo (BOHN et al., 2011), os principais atores envolvidos na computação em nuvem são definidos como:

- **Consumidor:** Uma pessoa ou organização que mantém uma relação de negócios com um *Provedor* de serviços de nuvem, na qual utiliza o serviço oferecido pelo provedor.
- **Provedor:** Uma pessoa, organização, ou entidade responsável por tornar o serviço de computação em nuvem disponível às partes interessadas. As atribuições do *Pro-*

vedor variam de acordo o tipo de serviço ofertado (IaaS, PaaS, SaaS), e são divididas com o *Consumidor* conforme o tipo de serviço contratado.

- **Auditor:** Uma pessoa ou entidade que conduz avaliações independentes de serviços de nuvem, e também de desempenho, segurança e operação de sistemas de informação presentes na implementação da nuvem.

- **Corretor:** Entidade que controla o uso, desempenho e distribuição de serviços em nuvem, e intermedeia relações entre *Provedor* e *Consumidor*. Este ator oferece um serviço que simplifica o acesso à diversos serviços de nuvem, como possibilitar, por exemplo, que o *Consumidor* realize a integração entre serviços de nuvem sem necessitar que tenha conhecimento aprofundado;

- **Operador:** Intermediário que provê conectividade e transporte de serviços de nuvem do *Provedor* até o *Consumidor*. Ou seja, um ator neste caso pode ser uma companhia que oferece serviços de telecomunicação ou que transporta fisicamente dispositivos de armazenamento.

Para alguns destes atores, como o provedor e o auditor, há interesse em saber detalhadamente sobre a operação do serviço de computação em nuvem em questão. Neste âmbito, a caracterização de tráfego oferece uma maneira de entender o que ocorre nas redes relacionadas à nuvem, permitindo entender mais detalhadamente o comportamento e desempenho da nuvem. Com estas informações em mãos, torna-se possível por exemplo, descobrir o quanto é possível expandir a nuvem com a infraestrutura de rede atual, e também a entender como componentes internos da nuvem interferem no seu funcionamento.

Dentro dos modelos de serviços ofertados, este trabalho foca no tipo IaaS. Neste sentido, usualmente os provedores IaaS fornecem serviços de virtualização, *e.g.,* VM e contêiner. Os serviços baseados em VM são os mais comumente ofertados e utilizados (NICODEMUS; BOERES; REBELLO, 2016). Segundo Jain e Paul (2013), a adoção de tecnologias de virtualização tem como um de seus principais motivos a facilidade de gerenciamento, obtida pela abstração dos recursos em função da virtualização. Ou seja, os recursos computacionais são ofertados através de uma interface uniforme padronizada, que independente do hardware sendo virtualizado, os dispositivos virtuais funcionam da mesma maneira, o que simplifica seu uso e controle pelo consumidor ou provedor da

nuvem.

2.1.1 Virtualização

A virtualização é uma das tecnologias que servem como alicerce às nuvens computacionais. Através da virtualização criam-se infraestruturas lógicas, incluindo topologias de redes e computadores, que funcionam sobre recursos físicos existentes. Assim, possibilita-se criar, por exemplo, vários computadores virtuais, ou seja, VMs, que executam em apenas um computador físico, e contam com característica de isolamento, sendo controladas por algum software responsável (*e.g.,* hipervisores, como: VirtualBox[1] e *Kernel-based Virtual Machine* (KVM)[2]).

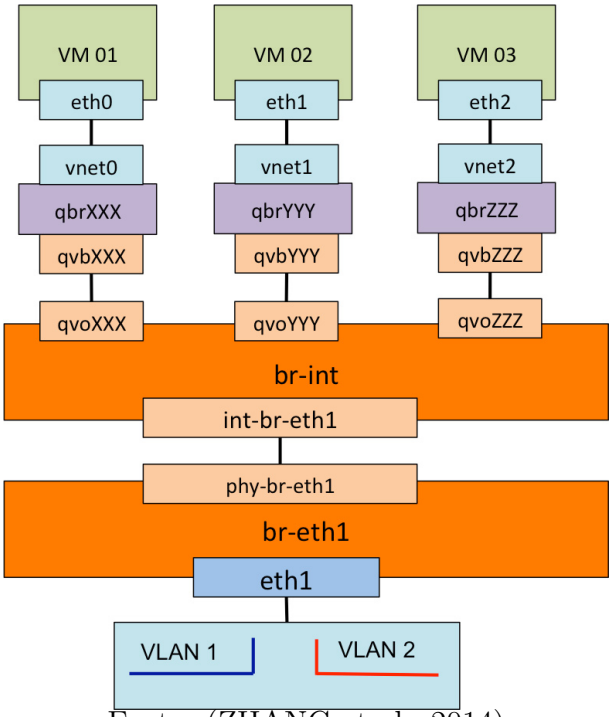
Segundo Tanenbaum e Wetherall (2010), a virtualização de redes computacionais pode ser feita através de uma *Virtual Local Area Network* (VLAN), por exemplo, na qual um *switch* com suporte a VLAN encaminha pacotes por suas diferentes portas em função da VLAN contida no cabeçalho de cada pacote, fornecendo a capacidade de isolamento entre computadores pertencentes à mesma rede física. Em relação a comunicação das VMs, tecnologias de virtualização de rede instaladas num computador, como a *Open vSwitch*[3] permite que VMs comuniquem-se através da Camada 2, independente da localização física da VM, e também provê isolamento na comunicação das várias VMs executando em um computador (CALLEGATI et al., 2014).

Contudo, a virtualização adiciona camadas de abstração sobre a infraestrutura física existente, tal como o computador, por exemplo, que deve redirecionar e endereçar cada pacote na sua rede virtual. Outro ponto levantado por Callegati et al. (2014) é referente ao monitoramento e controle de tráfego. O tráfego entre VMs no mesmo computador não são comutados para a rede física, logo, não é possível empregar dispositivos para monitorar e filtrar pacotes de um jeito que não comprometa o isolamento e segurança da infraestrutura virtualizada.

A Figura 2.2 ilustra a arquitetura de rede em um computador. Para um pacote originário da `vm01` alcançar a rede física (`vlan01`), é necessário que o pacote atravesse nove dispositivos virtuais no computador.

[1] `https://www.virtualbox.org`
[2] `https://www.linux-kvm.org`
[3] `http://openvswitch.org`

Figura 2.2: Redes virtualizadas em um computador.



Fonte: (ZHANG et al., 2014).

Segundo Zhang et al. (2014), dispositivos virtualizados *Open vSwitch* presentes na Figura 2.2 (`br-int, br-eth1`) comportam-se de maneira idêntica a *switches* físicos. Outros dispositivos virtualizados também estão presentes, como *hubs* (`qbrXXX, qbrYYY, qbrZZZ`), e interfaces virtualizadas (`qvbXXX, qvoXXX, int-br-eth1`, e `phy-br-eth1`). Ainda segundo Zhang et al. (2014), a complexidade envolvida na transição dos pacotes entre as interfaces virtualizadas é significativa, e dificulta o processo de *debugging* e correção de possíveis problemas.

Nuvens computacionais utilizam virtualização em todos os âmbitos discutidos simultaneamente, e necessitam de um software que gerencie o ciclo de vida e comunicação nestes ambientes. Softwares que gerenciam nuvens computacionais possuem arquiteturas complexas em função da quantidade de tarefas que são responsáveis, tal como: balancear a carga entre os hosts, configuração, alocação e destruição de instâncias de VMs, gerenciar as imagens nas quais as VMs são armazenadas, gerenciar a conectividade de rede das VMs, disponibilizar interfaces para gerenciamento externo de VMs, etc. A Seção 2.2 introduz os conceitos básicos sobre o software de gerenciamento de nuvem IaaS OpenStack[4], que compõe o ambiente no qual planeja-se realizar a caracterização de tráfego, conforme proposto neste trabalho. O OpenStack oferece serviços do tipo IaaS, podendo ser usado por um provedor para ofertar serviços tanto privados como públicos.

[4] `https://www.openstack.org`

2.2 OpenStack

O OpenStack foi criado em 2010, através de uma colaboração entre a Rackspace e Anso Labs (contratada da NASA), o sistema está atualmente na sua 16ª versão, identificada pelo codinome "Pike"(The OpenStack project, 2017l). **A versão do OpenStack abordada nesta seção é a Newton (2016)** para garantir que toda a discussão relacione-se com o ambiente no qual o trabalho está sendo realizado, que atualmente possui o OpenStack Newton instalado.

O OpenStack permite controlar uma infraestrutura composta por recursos de computação, armazenamento e rede em um *data center*. O OpenStack possui diversas formas de acesso, sendo a mais básica através da interface web (Horizon) e as mais versáteis como APIs que proporcionam acesso facilitado para os consumidores e provedores (The OpenStack project, 2017l).

A principal forma de acesso é pela API, que expõe funcionalidades da nuvem para serem acessadas por consumidores fisicamente distantes e por aplicações externas, permitindo por exemplo, a integração com outras nuvens. Segundo Corradi, Fanelli e Foschini (2014), o serviço de API do OpenStack é o seu *front-end*, pois recebe requisições dos consumidores e as transforma em ações na nuvem. As funcionalidades oferecidas desta maneira variam desde controle das VMs até autenticação. Mais especificamente, a API disponibiliza suas funcionalidades como *Web Services*, cujas requisições são transportadas como mensagens HTTP com seu *payload* em XML, que são interpretadas e convertidas para comandos na nuvem. Através deste estilo de *Web Service*, a API do OpenStack fornece compatibilidade com nuvens de empresas distintas, como Eucalyptus e Amazon (CORRADI; FANELLI; FOSCHINI, 2014).

A instalação do OpenStack é modular, feita através de serviços, que para maior efetividade, funcionam sobre uma arquitetura recomendada de organização da rede.

2.2.1 Serviços OpenStack

A modularidade é um diferencial, pois permite acoplar serviços para adicionar funcionalidades extras a uma instalação existente, como serviço de banco de dados (*e.g.,* Trove). De acordo com Litvinski e Gherbi (2013), esta arquitetura modular independente e com comunicação assíncrona foi criada para garantir a elasticidade e escalabilidade horizontal,

na qual serviços podem ser utilizados em qualquer instalação. A Tabela 2.1 contém alguns dos serviços de uma instalação de exemplo do OpenStack Newton, segundo a documentação do projeto (The OpenStack project, 2017g).

Tabela 2.1: Lista de serviços em uma instalação básica do OpenStack

Tipo de serviço	**Nome do serviço**	**Portas (TCP)**	**Descrição**
Dashboard	Horizon	80, 443	Disponibiliza uma interface gráfica web para interagir com serviços subjacentes do OpenStack (e.g., iniciar uma instância, atribuir endereços IP, e definir permissões de acesso)
Computação	Nova	8773 - 8775, 6080 - 6082	Gerencia o ciclo de vida de instâncias de processamento em um ambiente OpenStack. Responsabilidades incluem iniciação, escalonamento e desalocação de VMs de acordo com a demanda
Rede	Neutron	9696	Habilita o serviço de conectividade de rede para outros serviços OpenStack, como o Nova. Provê uma API para consumidores definirem as redes e suas particularidades
Armazenamento de objeto	Swift	6000 - 6002, 873	Armazena e recupera objetos contendo dados não estruturados através de sua API. Possui tolerância a falha através da replicação de dados numa arquitetura espalhada/horizontal
Armazenamento de bloco para VM	Cinder	8776, 3260	Disponibiliza armazenamento persistente em bloco para instâncias em execução
Identificação	Keystone	5000, 35357	Disponibiliza um serviço de autenticação e autorização para outros serviços do OpenStack. Inclui autenticação da API e descoberta de serviços
Imagem de VM	Glance	9292, 9191	Armazena e recupera as imagens utilizadas nas VMs. O serviço Nova utiliza este serviço durante o provisionamento de instância de VMs
Telemetria	Ceilometer	8777	Monitora e mede o uso da nuvem para cobrança, benchmark e outros fins estatísticos

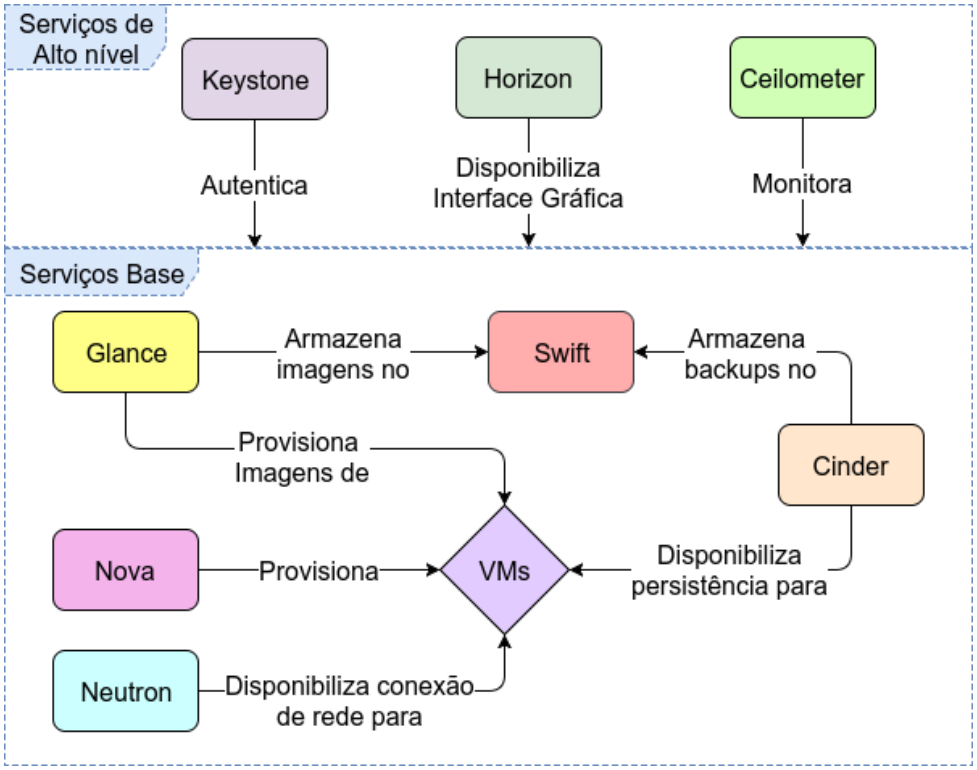
Fonte: Adaptado de (The OpenStack project, 2017g).

Na Tabela 2.1, o único serviço não presente nesta lista em relação à documentação é o Heat, que é um serviço opcional, responsável por orquestrar e sincronizar múltiplas nuvens. Para os serviços funcionarem corretamente as portas listadas nesta tabela, que utilizam TCP, não podem ser bloqueadas pelos *firewalls* pertencentes à infraestrutura da nuvem. Além destas portas, existem outras portas TCP de propósito geral no OpenStack, como a TCP/3306 (acesso ao MySQL), que é utilizada pela maioria dos serviços, e a porta TCP/5672, na qual o RabbitMQ, um servidor para troca de mensagens *Advanced Message Queuing Protocol* (AMQP) opera, e é responsável pela comunicação interna dos serviços Nova, Neutron e Cinder (explicado na Seção 3.1).

A Figura 2.3 ilustra como serviços básicos do OpenStack se relacionam. Nos serviços de alto nível, há relacionamento direto com múltiplos serviços, tanto para disponibilizar funcionalidades para outros serviços (*e.g.,* Keystone), quanto para utilizar ou acessar recursos de outros serviços (*e.g.,* Horizon e Ceilometer, respectivamente). O Keystone cuida da autenticação e autorização dentro da nuvem, que é consultado pelos serviços da nuvem antes de executar alguma ação solicitada pela API, por exemplo, com o objetivo de confirmar a identidade do solicitante. O objetivo do Horizon é disponibilizar uma

interface gráfica (*front-end*), que possibilita ao consumidor gerenciar, criar e iniciar suas instâncias e respectivas configurações (*e.g.,* rede, sistema operacional e características físicas (*flavor*)), que então são transformadas em chamadas de API e enviadas para outros serviços do OpenStack. O Ceilometer é o serviço que monitora e analisa o desempenho do OpenStack, disponibilizando métricas relacionadas a rede (*e.g.,* quantia de bytes enviado por cada interface, quantia de erros de envio e recebimento por interface), armazenamento (*e.g.,* tamanho de imagem, latência do armazenamento de instância), processamento (*e.g.,* uso de memória de instância, quantia de instruções executadas), entre outros, que podem ser empregadas para *benchmark* ou cobrança pelo uso dos recursos, por exemplo.

Figura 2.3: Arquitetura conceitual entre serviços do OpenStack



Fonte: Adaptado de (The OpenStack project, 2017g).

Em relação aos serviços base, segundo The OpenStack project (2017g), a maioria está relacionado com disponibilizar funcionalidades para instâncias (representado na Figura 2.3 pelo losângulo escrito VMs, na qual apenas o Swift não tem relação direta), e neste sentido, segundo Kumar et al. (2014), para manter a modularidade e consequentemente a escalabilidade, o OpenStack divide responsabilidades sobre múltiplos serviços, como por exemplo, o Glance armazenar no Swift as imagens de sistemas operacionais que disponibiliza. O Nova, por exemplo, utiliza múltiplos serviços durante certas tarefas, como no provisionamento de instância de VM, na qual requisita a imagem do sistema

operacional ao Glance, configura a rede da instância com o Neutron e disponibiliza armazenamento persistente para a instância através do Cinder.

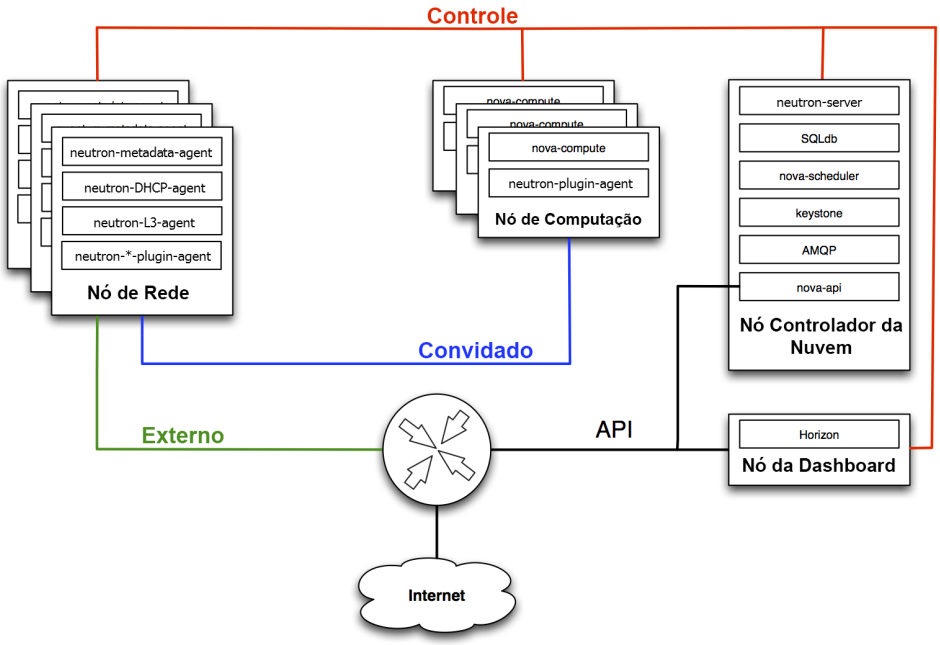
A instalação destes serviços nos servidores também pode seguir uma arquitetura sugerida. Segundo o The OpenStack project (2017g), uma instalação adequada do OpenStack contém vários computadores que hospedam serviços (hosts), que são divididos em pelo menos dois tipos: controle e computação. Ao longo deste trabalho, ao referenciar-se um host com papel específico em uma nuvem (*e.g.,* controle, computação), este será chamado por nó, seguido do seu papel (*e.g.,* nó de controle, nó de computação). Dentro de uma nuvem podem haver múltiplos hosts distribuídos entre computação e controle, na qual nós de controle executam serviços que gerenciam e dispõem recursos à nuvem, enquanto que nós de computação hospedam instâncias. Caso sejam adotados os serviços na Figura 2.3, por exemplo, todos os serviços, menos parte do serviço Nova ficariam hospedados em nós de controle. Isto ocorre pois de acordo com Corradi, Fanelli e Foschini (2014), o Nova divide-se em quatro componentes: API, computação, escalonador, e o gerenciador de rede, que foi substituído pelo Neutron. O serviço de API do Nova é responsável por disponibilizar acesso e controle às instâncias, tanto internamente quanto por acesso externo. O serviço de computação é instalado nos nós de computação, e é quem se comunica com o software gerenciador de VMs (*hypervisor*), também instalado em cada nó de computação. Dentre as responsabilidades do serviço de computação está gerenciar o ciclo de vida das instâncias e acessar métricas de desempenho e carga do *hypervisor*. O serviço de escalonamento recebe os pedidos de instância de VM pela API, e então realiza o escalonamento para decidir em qual nó de computação a instância será alocada.

É possível executar todos estes serviços em um mesmo host, contudo, não há necessidade, pois é possível empregar apenas um nó controlador para inúmeros nós de computação. Também podem ser adicionados nós de controle redundantes, que realizam as mesmas tarefas e estão sempre em sincronia, e garantem, portanto, o funcionamento ininterrupto da nuvem caso um dos nós de controle fique indisponível. Esta comunicação entre nós de controle e computação ocorre na rede de controle, contida no Domínio de Controle, apresentado na Seção 2.2.2. Considerando as diversas possibilidades de instalação de uma nuvem OpenStack, sua documentação define uma organização básica da rede, que possibilita escalar a nuvem e seus serviços.

2.2.2 Arquitetura de rede

Para realizar a comunicação entre os host que hospedam diferentes serviços são utilizadas redes físicas de acordo com infraestrutura recomendada, conforme ilustrado na Figura 2.4. Esta arquitetura possui três domínios com políticas de seguranças distintas: Domínio Público, Domínio de Controle e Domínio de Convidados (The OpenStack project, 2017g). Os três domínios englobam quatro redes distintas, cujas políticas de segurança das redes correspondem ao domínio na qual pertencem, e são divididas em – **Domínio Público**: rede externa e rede de API; **Domínio de Convidados**: rede de convidados; e **Domínio de Controle**: rede de controle e rede de armazenamento (opcional).

Figura 2.4: Arquitetura de rede física recomendada para o OpenStack



Fonte: Adaptado de (The OpenStack project, 2017c)

O **Domínio Público** envolve as redes responsáveis pela visibilidade da API da nuvem criada na Internet e do acesso à Internet pelas VMs, que correspondem respectivamente à rede externa e à rede de API. Em ambas as redes, todos os endereços IPs devem ser visíveis a partir da Internet para seu pleno funcionamento.

No **Domínio de Convidados** encontra-se a rede responsável pela comunicação entre as VMs. Os consumidores do serviço de oferta de VMs são referenciados como convidados por não possuírem vínculo direto com a administração da nuvem, logo, não há garantias sobre quem estes convidados são e quais os seus objetivos. Portanto, tecnologias de virtualização da rede são empregadas em cada nó de computação para isolar o tráfego de cada VM e impedir que elas sejam capazes de interferir ou visualizar

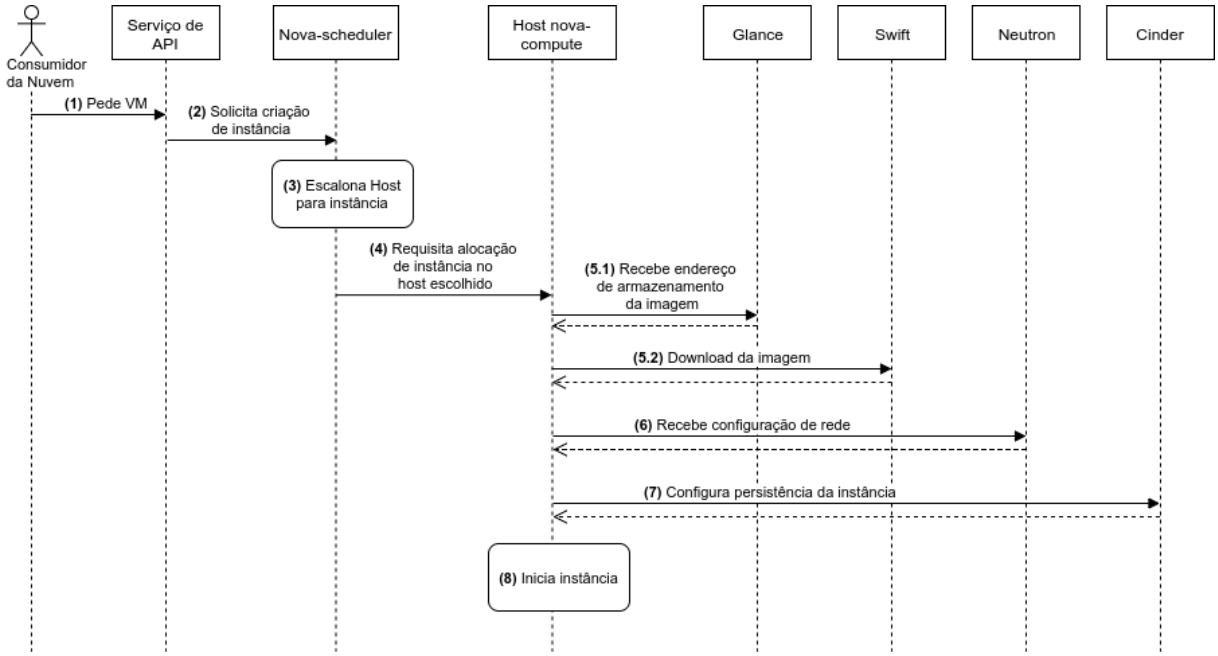
o tráfego para as outras VMs no mesmo host. Mais especificamente, o serviço Neutron é responsável pela virtualização de rede de convidado, na qual permite *multitenancy* através de *network namespaces*. *Multitenancy* é uma arquitetura de software que permite isolar um grupos de consumidores da nuvem (*tenant* ou *project* em versões mais recentes do OpenStack) de maneira que eles compartilhem o acesso a certos recursos, cada qual com privilégios específicos. Ou seja, é possível permitir que certas VMs criem redes virtuais entre si como se fizessem parte da mesma rede física, e portanto, podem comunicar-se diretamente, isso tudo de acordo com a configuração dos consumidores. Segundo (DENTON, 2016), um *network namespace* é definido como uma cópia lógica de uma rede, com regras de roteamento e *firewall* próprias e dispositivos de interface de rede virtuais. O Neutron provê DHCP e roteamento próprio para cada *namespace* e permite que a rede de um *project*/*tenant* conecte-se com o Domínio Público (Internet) adicionando um roteador virtual à sua rede.

O **Domínio de Controle** contém a rede responsável pelo tráfego de controle e de armazenamento do OpenStack, que corresponde ao tráfego gerado pela comunicação entre os seus serviços. Esta rede é um ambiente acessível apenas pelo software que gerencia a nuvem, e apenas os administradores da nuvem tem a capacidade de acessá-la em condições normais, dispensando a necessidade de isolamento através da virtualização. Neste sentido, a virtualização de rede está presente apenas nos nós de computação, que virtualizam todas as interfaces de rede em conjunto com o *hypervisor*. Tanto os nós controladores da nuvem quanto os de computação têm interfaces físicas que ligam-se à rede de controle, e portanto, a documentação do OpenStack recomenda que todos os nós pertencentes à nuvem possuam duas interfaces de rede cada. Múltiplas interfaces de rede física permitem conectividade à rede de controle e a outras redes conforme o caso, como a rede de convidados no caso de nós de computação, ou a rede pública no host responsável por gerenciar a rede das VMs. Portanto, uma VM em execução só poderá acessar a rede de controle caso consiga executar um ataque de elevação de privilégio e tenha acesso às interfaces de rede físicas do host na qual a VM em questão está alocada (KRUTZ; VINES, 2010).

Para exemplificar o funcionamento da rede de controle, a Figura 2.5 ilustra o processo de criação de instâncias do OpenStack do ponto de vista dos serviços, em alto nível. A comunicação entre os serviços é realizada através da rede de controle, ou seja, todas as setas na figura representam tráfego gerado na rede durante o processo.

Opcionalmente, é possível criar uma rede de transferência de arquivos, separada da rede de controle, mas contida no Domínio de Controle, com o objetivo de otimizar o uso de banda para serviços relacionados com transferência de arquivos (Swift, Cinder), que podem ser responsáveis por boa parte do tráfego de controle.

Figura 2.5: Diagrama de sequência em alto nível: Criação de instância no OpenStack



Fonte: O próprio autor.

1. O processo de criação de instância inicia com um pedido de VM por um consumidor da nuvem através da interface gráfica no Horizon, que realiza o pedido na API, ou diretamente pela API. Para simplificar o diagrama, o serviço Keystone foi omitido, e sua responsabilidade está relacionada com esta etapa **(1)**, cuja API autentica o consumidor da nuvem para garantir que ele possui as permissões necessárias;

2. A API então comunica o serviço Nova-scheduler, responsável pelo escalonamento;

3. O componente Nova-scheduler, que executa em um nó de controle define em qual nó de computação a instância será executada;

4. Após realizar o escalonamento, repassa o pedido para o nó de computação escolhido. O nó de computação recebe o pedido de instância, que contém informações como imagem a ser utilizada na VM (*flavor*), e as características da VM (*e.g.,* memória, processamento, rede), e inicia o processo de configuração da instância de VM;

5. O processo para recuperar a imagem do sistema operacional a ser utilizada na VM é divido em dois passos;

   5.1. O primeiro passo corresponde à buscar o endereço de armazenamento da imagem no Glance; e

   5.2. Então, no segundo passo, recupera do Swift o dado não estruturado que contém a imagem correspondente. Após recuperada a imagem, os passos posteriores no processo são responsáveis por configurar o ambiente na qual a VM executará, e a ordem que ocorre pode não ser a mesma deste diagrama de sequência;

6. A configuração da rede da VM é fornecido pelo Neutron, que define as configurações de acordo com o *tenant* e realiza as modificações necessárias na rede.

7. A última configuração necessária é a persistência de dados da VM, que é fornecido pelo Cinder, serviço responsável pelo armazenamento de bloco.

8. Com todas as configurações definidas, o passo final é a inicialização da VM pelo nó de computação que hospedará a instância.

Nota-se que o OpenStack possui vários serviços que podem estar dispostos em hosts distintos. Nesse sentido, o tráfego do Domínio Público reflete o tráfego normal das aplicações hospedadas. Em contrapartida, o tráfego do Domínio de Controle e Domínio de Convidado são transações internas da nuvem, às quais não são visíveis a perspectiva do consumidor. Ou seja, esse tráfego possui características intrínsecas à solução de nuvem e à sua arquitetura de implantação. O foco deste trabalho é caracterizar o tráfego do Domínio de Controle para levantar aspectos operacionais e de desempenho de uma nuvem OpenStack.

2.3 Caracterização de Tráfego

A medição / caracterização do tráfego é uma tarefa empregada para entender e resolver problemas relacionados com o desempenho em redes de computadores (DAINOTTI; PESCAPE; VENTRE, 2006). Pretende-se ter a caracterização de tráfego neste trabalho como fim, ou seja, a rede caracterizada será o produto da aplicação de técnicas que analisam o tráfego e de análises posteriores dos dados gerados. O processo de caracterização

de tráfego é descrito nesta seção em duas etapas: medição de tráfego/levantamento de dados e a análise do tráfego obtido.

2.3.1 Medição de Tráfego

Durante a medição de tráfego, são capturados os dados que trafegam pela rede da qual pretende-se caracterizar o tráfego. Na captura dos dados são utilizadas ferramentas (*e.g.,* tcpdump) que realizam a captura dos dados através de duas abordagens: medição ativa e medição passiva. Após a captura, o tráfego então é abstraído em um certo nível de agregação (*e.g.,* fluxo, pacote e bytes), variando de acordo com as capacidades da ferramenta de monitoramento e a técnica de análise empregada (DAINOTTI; PESCAPE; VENTRE, 2006).

Tipo de Medição

Atualmente existem duas abordagens para realização de medição do tráfego de rede: medição ativa e medição passiva (VILELA, 2006; WILLIAMSON, 2001). A medição ativa injeta tráfego na rede e mede o seu desempenho baseado no que foi injetado. Este tipo de medição é considerado intrusivo, pois o processo gera tráfego, que influencia no funcionamento da rede, e consequentemente na medição. Desta forma, medições ativas requerem cuidados na utilização, pois deve ser levado em consideração o tráfego gerado pela medição na etapa de análise do tráfego, posteriormente. Ao medir o comportamento de filas em um roteador, por exemplo, o processo de medição pode alterar o comportamento da fila e influenciar nos resultados (VILELA, 2006). De acordo com Williamson (2001), algumas ferramentas para medição ativa são: `ping`, `traceroute` e `pathchar`. O `ping` fornece a latência na rede até um certo destino, `traceroute` fornece o caminho pelo qual um pacote foi roteado até o seu destino, e o `pathchar` estima o uso de banda e latência dos nós até um destino.

Por outro lado, a medição de tráfego passiva é usada para observar e armazenar o tráfego de pacotes em uma rede sem injetar qualquer tráfego na rede durante o processo de medição. Ou seja, a medição de tráfego é não intrusiva (WILLIAMSON, 2001). Neste sentido, a medição passiva necessita que sejam criados pontos de monitoramento por onde deseja-se coletar o tráfego, seja ele por hardware específico, ou dispositivos que pertençam a topologia e disponham desta funcionalidade (VILELA, 2006). O protocolo *Simple*

Network Management Protocol (SNMP), presente em vários dispositivos de rede permite o monitoramento de variáveis relacionadas ao tráfego (*e.g.,* utilização de banda, perda de pacote) e do dispositivo (*e.g.,* uso de processador e memória) (GROSSGLAUSER; REXFORD, 2002). Ainda segundo Grossglauser e Rexford (2002), estes dados disponibilizados pelos dispositivos através do SNMP são acessados e armazenados por um software que monitora e gerencia a rede. Contudo, estes dados não relevam detalhes suficientes para detectar a origem de um problema na rede (*e.g., Distributed Denial of Service* (DDoS)), necessitando a coleta de tráfego para obter-se maiores detalhes. A coleta de tráfego em dispositivos de rede pode ser realizada utilizando protocolos implementados com este fim, que dentre as ferramentas existentes destacam-se: NetFlow e sFlow. A ferramenta NetFlow, presente em roteadores da marca Cisco disponibiliza a medição passiva de tráfego TCP e UDP, agregando-o em fluxos (VILELA, 2006). sFlow foi criado através de um consórcio entre várias empresas (*e.g.,* HP, Hitachi, Brocade, etc), com o objetivo de criar uma ferramenta escalável de coleta de tráfego. A ferramenta sFlow funciona de maneira similar à Netflow, agregando os pacotes em fluxos, mas aplicando obrigatoriamente a técnica de amostragem sistemática, apresentada na Seção 2.3.1, com o objetivo de alcançar escalabilidade com monitoramento em tempo real (sFlow Project, 2003).

Nível de agregação do tráfego

O nível de agregação do tráfego define a granularidade dos dados a serem trabalhados, ou seja, o quão detalhada é a representação do tráfego. Portanto, uma caracterização de tráfego com alto nível de granularidade (fino), por exemplo, exigirá uma quantidade maior de armazenamento e processamento posteriormente, mas com a vantagem de representar o tráfego com alta fidelidade.

Contudo, deve-se levar em conta a abordagem que será utilizada na etapa de análise do tráfego ao decidir a granularidade, pois os dados coletados servirão como entrada para abordagem escolhida. Neste sentido, por exemplo, algumas técnicas de análise utilizam apenas informações contidas no cabeçalho dos pacotes, descartando-se o *payload*. Ao empregar técnicas deste tipo não há ganho em representar o tráfego com uma granularidade fina, em função da alta quantidade de armazenamento e tempo de processamento necessários. Segundo a taxonomia de Khalife, Hajjar e Diaz-Verdejo (2014), o tráfego é comumente agregado nos seguintes níveis:

- **Pacote:** Todos os pacotes que trafegam na rede são armazenados, e é um nível de agregação com granularidade fina, ou seja, com maior quantidade de dados. Neste nível, exemplos de dados coletados são IP de origem e destino, tamanho do pacotes, números do protocolo e *payload*. Ferramentas como TCPdump[5] e SNORT[6] são capazes de fazer captura neste nível (CALLADO et al., 2009).

- **Fluxo:** Agregação de pacotes em um único objeto, obtido através da aplicação de uma regra de agregação durante a medição do tráfego, como por exemplo, agrupar pacotes com mesma porta e IP de origem e destino, e portanto, possui granularidade maior que o pacote. Os dados coletados nesse nível são: fluxos por unidade de tempo, tamanho do fluxo, duração do fluxo (CALLADO et al., 2009). Ferramentas como Cisco NetFlow e sFlow são comumente utilizadas para coleta neste nível.

- **Host:** Agregação com granularidade grossa, na qual um host é um objeto, representado pelo seu respectivo endereço IP. Neste nível, é associado o uso de uma aplicação a um host, e portanto a quantidade de dados é limitada. Um exemplo é o Blinc, criado por Karagiannis, Papagiannaki e Faloutsos (2005), que analisa as interações do host num alto nível, como o seu papel na rede baseado na quantidade de portas (atua como provedor de algum serviço) e sua interação com outros host (popularidade na rede).

Dentre os níveis de agregação de tráfego apresentados, segundo Khalife, Hajjar e Diaz-Verdejo (2014), a agregação por fluxo foi o mais adotado nos trabalhos analisados. Isto pode estar relacionado com a similaridade nas informações disponíveis nos níveis de fluxo e pacote, e também, em certas situações a análise com granularidade fina é impraticável por conta da quantidade de tráfego gerado (*e.g., data centers* e *backbones*), cuja utilização de fluxos é viável.

De acordo com Callado et al. (2009), durante o processo de medição de tráfego é possível compactar a quantidade de dados gerados com qualquer granularidade através do emprego de técnicas de amostragem. Ainda segundo Callado et al. (2009), esta fase opcional é disponibilizada pelo processo de amostragem sistemática em alguns roteadores da Cisco na funcionalidade de coleta de pacotes. A amostragem sistemática, representada pela expressão $\frac{1}{n}$, significa que nestes casos haverá a coleta apenas do 1º pacote a cada

[5] `http://www.tcpdump.org`
[6] `https://www.snort.org`

n pacotes. Apesar de parecer questionável quando aplicado a situações com grandes variações, um dos estudos levantados pelo autor indicou que não há uma perda de precisão relevante no processo, e a amostragem permitiu uma significativa redução do uso do processador e rede pelo roteador.

2.3.2 Análise de Tráfego

A etapa posterior à medição do tráfego é a de análise do tráfego, em que todo o tráfego coletado é analisado. Não foi identificado um método específico para realizar a caracterização de tráfego. Portanto, nesta etapa são apresentadas técnicas de classificação com maior abrangência, que podem ser utilizadas, por exemplo, para classificação e análise de tráfego em tempo real. As técnicas de classificação categorizam o tráfego em função da correspondência de características presentes no tráfego obtido com suas respectivas classificações. Segundo a taxonomia de Khalife, Hajjar e Diaz-Verdejo (2014), as técnicas de classificação são divididas em inspeção de conteúdo, análise de características do tráfego ou abordagens híbridas entre elas.

As primeiras técnicas a surgirem baseavam-se apenas na inspeção do conteúdo de pacote, em que comparam o conteúdo do tráfego na rede com amostras já classificadas. Um arquivo de vídeo sendo transmitido na rede, por exemplo, pode ser detectado pela presença de uma sequência de bytes característicos, ou seja, sua assinatura (DHARMAPURIKAR et al., 2004). A verificação de assinatura é efetuada através da busca por sequências de bytes idênticas, ou a avaliação com expressões regulares. Contudo, limitações com questões de privacidade, a popularização da criptografia e a quantidade de processamento necessário limitam sua efetividade (PARK; HONG; WON, 2011).

A análise de características de tráfego é complementar à inspeção do conteúdo, na qual classifica o tráfego através das informações do cabeçalho e metadados relacionados ao tráfego analisado. Essa análise também é conhecida como classificação às escuras, e resulta na criação de perfis de tráfego e a sua consequente análise estatística. As técnicas de análise de características de tráfego utilizam abordagens gráficas ou estatísticas, que subdividem-se em modelos estatísticos simples ou com auxílio de *machine learning*. As principais técnicas que analisam características do tráfego são:

- **Gráfica:** Técnicas gráficas, como em Jr., Turkett e Fulp (2009), ilustram as inte-

rações entre os computadores em uma rede através de grafos baseados no tráfego analisado. Nesta representação os nós são os hosts envolvidos e as arestas a interação entre eles. Esta modelagem baseia-se na ideia que hosts envolvidos na mesma aplicação apresentam padrões que revelam tal comportamento, ao mesmo passo que a utilização de grafos facilita a visualização destes padrões.

- **Modelagem estatística simples:** Técnicas nesta categoria analisam atributos genéricos do tráfego e suas respectivas correlações estatísticas para identificar as aplicações executando no tráfego analisado. O princípio base desta abordagem é de que o tráfego na camada de rede possui propriedades estatísticas (*e.g.,* distribuição do tamanho dos pacotes, tempo entre seus recebimentos) que são únicos para certas classes de aplicações, permitindo sua distinção (NGUYEN; ARMITAGE, 2008). A utilização de heurísticas, por exemplo, estende esta abordagem aplicando um conjunto de regras baseados nas informações processadas (KARAGIANNIS; PAPAGIANNAKI; FALOUTSOS, 2005). Outra extensão é com a análise de perfil, que avalia o nível de heterogeneidade de um comportamento em relação ao perfil de tráfego comum àquela rede (XU; ZHANG; BHATTACHARYYA, 2005).

- **Estatística com machine learning:** De acordo com Williams, Zander e Armitage (2006), algoritmos de *machine learning* comparam tráfego com os mesmos conjuntos de características, procurando semelhanças para agrupá-los em *clusters*. As duas principais abordagens na caracterização de tráfego são *machine learning* não supervisionado e *machine learning* supervisionado. Técnicas com *machine learning* supervisionado fornecem um conjunto de tráfego pré classificado, na qual o algoritmo analisa e treina para descobrir as prováveis correlações. Então, é aplicado a sua análise no conjunto de tráfego a ser classificado, que deve possuir as mesmas características observáveis do conjunto de treino. No *machine learning* não supervisionado é dado como entrada um conjunto de tráfego sem nenhuma classificação prévia, o algoritmo representa estes tráfegos de alguma forma, com vetores, por exemplo, e utiliza algum método de comparação, como a distância Euclidiana para verificar sua similaridade e agrupá-los (WILLIAMS; ZANDER; ARMITAGE, 2006).

Em relação às abordagens de análise apresentadas, a de análise estatística simples possui maior aplicabilidade, pois qualquer ambiente que gere métricas quantitativas pode empregá-lo. Por exemplo, dos trabalhos relacionados apresentados na Seção 2.5, três

deles utilizam a análise estatística simples (SCIAMMARELLA et al., 2016; WANG; NG, 2010; VENZANO; MICHIARDI, 2013), enquanto apenas um aplica a abordagem gráfica (SHARMA et al., 2015). Para a caracterização proposta neste TCC, a análise estatística simples inicialmente apresenta maior versatilidade, podendo ser mais facilmente adequada conforme define-se os detalhes da caracterização do tráfego. Assim, como as métricas em sua maioria relacionam-se com desempenho, na fase de análise do tráfego, esta técnica inicialmente possui o potencial de fornecer bons resultados ao mesmo passo que possui baixa complexidade.

No geral, as abordagens de análise apresentadas (estatística simples, gráfica, estatística com *machine learning*) têm foco na análise das características presentes no tráfego, na qual empregam características comuns a qualquer tipo de tráfego, ou seja, aplicáveis para analisar qualquer tráfego. Contudo, em certas situações, é possível utilizar informações relacionadas à rede na qual planeja-se caracterizar o tráfego para diminuir a complexidade na etapa de medição e análise de tráfego. Informações sobre protocolos de aplicação em execução na rede, por exemplo, facilitam a identificação do comportamento do tráfego, e a identificar qual parte dele fornecerá resultados relevantes. Neste sentido, sua aplicabilidade é limitada a redes cujo comportamento seja previsível. Segundo Karagiannis et al. (2004), por exemplo, monitorar aplicações apenas em função das portas e protocolos padrões não apresenta resultados confiáveis, por conta da capacidade destas aplicações de utilizarem diferentes portas a cada execução e empregarem protocolos próprios. Por outro lado, em nuvens computacionais OpenStack por exemplo, é possível mapear as portas TCP que originam o tráfego no Domínio de Controle aos serviços OpenStack que as originam, conforme apresentado na Tabela 2.1. Isto é possível, pois conforme afirmado anteriormente, o comportamento do tráfego no Domínio de Controle é previsível, no sentido de que não há como um dos host pertencentes à nuvem subitamente passar a utilizar apenas protocolos com criptografia (*i.e., torrent*), tentando proteger o destino e origem dos pacotes. Deste modo, o tráfego no Domínio de Convidado não possui esta mesma previsibilidade. Exemplificando, cujo consumidor da nuvem pode executar alguma aplicação própria, que ignora padronização de portas ou utilize algum protocolo com criptografia.

O processo de caracterização de tráfego é suscetível ao ambiente, ou seja, a rede na qual planeja-se caracterizar. Portanto, conhecimento prévio do ambiente é essencial, e a sua definição detalhada pode ajudar na definição de estratégias que diminuam a

complexidade das etapas envolvidas no processo de caracterização ou até da execução das ferramentas envolvidas.

2.4 Definição do Problema

A realização da caracterização de tráfego é uma atividade que possui etapas genéricas comuns, mas cuja aplicação pode mudar de acordo com o escopo, ou seja: devem ser levantadas informações sobre o ambiente para a definição das ferramentas e métodos. Neste sentido, ao caracterizar o tráfego de uma rede, deve-se analisar as possíveis aplicações e protocolos que operam na rede, e a partir de suas características definir como realizar a caracterização. Por exemplo, no processo de medição de tráfego, a escolha entre medição passiva ou ativa pode ser em função do requisito de baixa latência da rede de uma certa aplicação, na qual o uso de medição passiva é mais interessante por não injetar tráfego na rede. Contudo, ao empregar a medição passiva, deve-se analisar a topologia da rede, e decidir como alocar os pontos de monitoramento de maneira a cobrir o tráfego de interesse. Numa topologia *Fat-Tree*, por exemplo, estão presentes múltiplos níveis de *switches*, e apenas pacotes de *broadcast* alcançam toda a rede. Para alcançar todo o tráfego desta rede, tornam-se necessários *switches* com funcionalidades que permitam copiar ou gravar o tráfego que passa por eles, ou como outra alternativa, monitorar o tráfego nas interfaces de rede dos dispositivos conectados (ZHANG; MOORE, 2007).

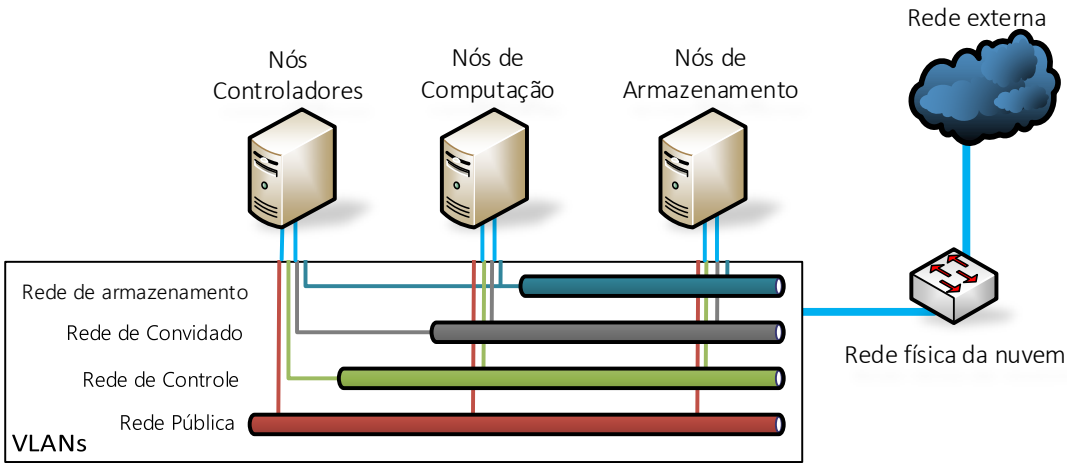
Ao levar esta questão para nuvens OpenStack, dois locais da nuvem apresentam potencial para caracterização de tráfego: Domínio de Convidado, e Domínio de Controle. Em função das características de funcionamento, cada uma deve ter a caracterização abordada de maneira diferente. No Domínio de Convidado, por exemplo, existem múltiplas redes virtualizadas, logo, medir o tráfego a partir de uma VM permite coletar apenas o tráfego pertencente àquela rede virtualizada, que é composta por VMs do mesmo *tenant/project* (DENTON, 2016). Para coletar este tráfego na sua totalidade, a medição de tráfego deve ser realizada em um nível mais baixo, que não seja afetado pela virtualização, como em *switches* físicos ou no nível do *hypervisor*. Outro ponto importante é em relação à privacidade do tráfego, que deve ser mantida a mais alta possível para os consumidores da nuvem, ou seja, o conteúdo do tráfego não deve ser coletado ao medir o tráfego deste domínio. Em função destes requisitos de privacidades imposta na medição de tráfego do Domínio de Convidados, por exemplo, devem ser empregadas apenas técnicas voltadas

para características do tráfego na fase de análise. Logo, as fases na caracterização de tráfego não são independentes, e cada uma pode apresentar seus próprios desafios.

Em relação ao Domínio de Controle, o seu funcionamento é mais simples por não haver alterações constantes na sua topologia durante a execução (*i.e.,* criação de nova rede virtual, inserção de *vm* na rede virtual criada). Um cenário que a virtualização pode ser empregada nestas redes ocorre caso elas operem na mesma rede física que outro domínio (*i.e.,* Domínio de Convidado, Domínio Público), e torna-se necessário isolar o tráfego delas através de alguma tecnologia como VLANs. A VLAN isola o tráfego da rede de controle, mas não acrescenta complexidade à rede, ou seja, mesmo que virtualizada, não é perceptível a diferença aos dispositivos presentes nela (TANENBAUM; WETHERALL, 2010). Contudo, devem ser tomados cuidados em relação ao desempenho do Domínio de Controle, que afeta o funcionamento de toda a nuvem em função dos serviços do OpenStack que são executados na mesma. Isto está diretamente relacionado com o objetivo deste trabalho, que é analisar o Domínio de Controle, e portanto, busca-se alcançar a representação mais fiel possível do tráfego dela.

Um exemplo de implementação de nuvem OpenStack é ilustrado na Figura 2.6, na qual o tráfego emprega VLANs para isolar o tráfego de cada uma das suas redes: rede de armazenamento, rede de controle, rede de convidados e rede pública. Ao todo, a nuvem em questão possui seis hosts: dois operam como nós de armazenamento, outro dois operam como nós de controle, e por fim, dois hosts funcionam como nós de computação.

Figura 2.6: Topologia de uma implementação de nuvem



Fonte: O próprio autor.

Cada um destes hosts estão interligados através de duas interfaces de rede física (em azul claro na figura), e portanto, o uso de VLAN é necessário para isolar as quatro redes, que compartilham o meio físico. Na parte de software, a utilização dos recursos destes hosts é feito pelo OpenStack, que disponibiliza suas funcionalidades para o consumidor através dos vários serviços instalados. Neste sentido, este trabalho foca na caracterização voltada para os serviços mais populares, segundo The OpenStack project (2017l): Glance, Keystone, Cinder, Swift, Nova e Neutron. Em relação ao tráfego gerado por estes serviços, este trabalho planeja caracterizar aspectos do tráfego de controle que relacionem-se com o seu comportamento e desempenho.

Dentre os cenários de análise, planeja-se observar, por exemplo, o impacto de eventos periódicos no tráfego de controle, como no caso do serviço Nova solicitar a lista de VMs ativas. Análises de comportamento geral também serão realizadas, que inclui, por exemplo, uma análise longitudinal, classificando o uso de banda dos serviços executando na nuvem. Outro caso previsto é a caracterização do tráfego em tarefas específicas na nuvem, como o ciclo de vida de uma instância de VM (*e.g.,* criação de uma instância, ilustrado na Figura 2.5). Estas são expectativas gerais do que espera-se analisar, mas conforme a análise dos dados obtidos for feita é possível que outras conclusões surjam. Um dos passos iniciais neste TCC, a análise de trabalhos com escopo semelhante tem sua importância neste processo, na qual analisa-se estudos que utilizam métricas e métodos aplicáveis neste trabalho.

2.5 Trabalhos Relacionados

Para nortear o desenvolvimento da abordagem de medição e análise de tráfego proposta neste trabalho, esta seção apresenta outros trabalhos que têm o escopo ou objetivo similar, na qual monitoram e/ou caracterizam o tráfego em nuvens computacionais. Ao apresentar os trabalhos busca-se explicar brevemente o contexto e objetivos deles, e então aprofundar em características, métricas e ferramentas que auxiliem na definição da proposta para caracterização de tráfego. Após expor cada trabalho individualmente, a seção 2.5.5 apresenta uma discussão e comparação entre as abordagens contidas nos trabalhos.

2.5.1 Venzano e Michiardi (2013)

O trabalho de Venzano e Michiardi (2013) investiga o impacto da virtualização no desempenho de comunicação em rede de aplicações com grande volume de dados em uma nuvem privada OpenStack. Para monitorar o tráfego gerado, desenvolveram a ferramenta de monitoramento OSMeF (OpenStack *Measurement Framework*), que executa sobre a ferramenta `nuttcp` para coletar métricas como vazão TCP e uso de CPU. Em relação aos testes, o Hadoop é a aplicação distribuída escolhida em função da intensa troca de mensagens. A aplicação foi alocada em um host para controlá-la e em outras dois hosts para realizar o processamento, além destes, outro host ficou responsável por hospedar serviços do OpenStack.

Neste trabalho foram realizadas duas caracterizações: **(I)** desempenho da interface de *Loopback* em VMs; e **(II)** comparação da comunicação entre VMs no mesmo nó e em nós diferentes. A caracterização **(I)** realizou-se diretamente na VM, e verificou-se o desempenho da interface com diferentes quantidades de processadores. A caracterização **(II)** foi realizada no Domínio de Convidado, que contém a rede física na qual as aplicações em execução nas VMs têm acesso e comunicam-se. Não é explícito onde que a ferramenta de monitoramento (OSMeF) executa nos testes, mas em **(II)**, pelas métricas coletadas (*i.e.,* uso de CPU), e a abordagem passiva de medição, é necessário monitorar cada host que executa a aplicação distribuída. Outra característica é que a ferramenta necessita executar à nível de *hypervisor* para acessar diretamente a interface física de rede do nó. O processo de medição de tráfego neste trabalho utiliza diretamente a maioria das métricas coletadas (*i.e.,* uso de CPU, vazão TCP), e apenas no caso da métrica "Índice de justiça de Jain" (JAIN; CHIU; WR, 1984) houve cálculo pós coleta. Dentre os resultados obtidos destacam-se: há menor desempenho em interfaces *Loopback* num ambiente virtualizado; e a comunicação entre VMs em um mesmo host possui melhor qualidade do que quando são distribuídas entre múltiplos hosts.

2.5.2 Sciammarella et al. (2016)

O trabalho de Sciammarella et al. (2016) caracteriza o tráfego do Domínio de Controle de uma nuvem OpenStack geodistribuída, buscando entender o impacto da comunicação entre os nós controladores da nuvem e os nós de processamento em diferentes cenários. O ambiente de testes simula uma nuvem OpenStack distribuída entre universidades e

centros de pesquisa. Na realização de um dos testes e sua respectiva medição de tráfego foi utilizada a ferramenta Rally[7], que possibilita realizar *benchmark* do OpenStack em diferentes cenários. Como resultado, este trabalho buscou analisar como a execução de instâncias e outros serviços impacta na rede de controle.

Foram realizados três testes neste trabalho: **(I)** impacto do número de servidores de armazenamento e processamento, **(II)** impacto do número de VMs por servidor de armazenamento e processamento, e **(III)** impacto da criação e destruição de múltiplas VMs. Nos três testes a caracterização de tráfego utilizou medição de tráfego passiva para realizar a coleta de tráfego, que focaram em analisar o uso de banda. O teste **(I)** analisou o *middleware* de comunicação RabbitMQ, e o MySQL, que são utilizados em múltiplos serviços, e separou o uso de banda de cada um em função do tempo decorrido. Similarmente, no teste **(II)**, analisou-se o uso de banda no Domínio de Controle após a criação de quantidades variadas de VM dentro de um ambiente controlado, sem categorizar as aplicações. Não foi especificada a ferramenta de monitoramento empregada nos testes **(I)** e **(II)**. O teste **(III)** utilizou a ferramenta Rally, que permite realizar os testes e coletar os dados de maneira automatizada, executando situações que simulam diferentes cargas na nuvem. Para realizar os testes, o Rally comunica-se com a nuvem OpenStack através da API, realizando ações na nuvem conforme definido no *script* sendo executado. Em relação à coleta de dados, não é definida de maneira clara na documentação da ferramenta como ela ocorre. Para o teste **(III)**, o *script* definido foi criação e destruição de múltiplas instâncias, que então coletou o tráfego gerado pela tarefa, buscando entender o uso de banda resultante em função das requisições. Não há como confirmar se houve uso exclusivo da ferramenta Rally na etapa de medição de tráfego, que é algo plausível levando em conta a similaridade das métricas analisadas.

2.5.3 Wang e Ng (2010)

O trabalho de Wang e Ng (2010) caracteriza o tráfego de rede entre VMs com um e dois processadores virtuais hospedados na *Amazon Elastic Cloud Computing* (EC2). O objetivo é identificar o impacto da virtualização no desempenho da rede, com alguns objetivos específicos, como verificar se há compartilhamento de processador entre duas VMs de apenas um processador virtual cada, e o impacto resultante na comunicação.

[7] `https://wiki.openstack.org/wiki/Rally`

Realizaram-se dois testes que avaliaram o uso de processador e características de rede como: vazão TCP e UDP, atraso na transmissão, e perda de pacote. Nos dois testes foram empregadas as mesma ferramentas: `ping` para calcular o atraso na comunicação, ferramentas autorais para vazão TCP e UDP, e a ferramenta Badabing (SOMMERS et al., 2005) para estimar a perda de pacote correspondente. Como resultado, concluiu-se que há, de fato, compartilhamento de processador nas VMs mais simples ofertadas pelo serviço Amazon EC2, e que este compartilhamento de processador resulta em instabilidade na transmissão de dados pela rede, variando rapidamente entre 1GB/s e zero GB/s.

Em relação aos testes, ambos analisaram as mesmas métricas, mas sob perspectivas diferentes, em dois teste: teste espacial, e teste temporal. O teste espacial realizou as medições em 750 VMs de 1 processador e 150 VMs de 2 processadores, no total, com duração aproximada de 10 segundos cada. O teste temporal efetuou as medições em 6 VMs de 1 processador e 3 VMs de 2 processadores ao longo de 150 horas cada. Em relação às ferramentas empregadas, todas utilizam medição ativa de tráfego, ou seja, fizeram a medição com base nos pacotes criados pela próprias ferramentas. As ferramentas autorais para medição de TCP e UDP funcionam de maneira similar, executando duas instâncias das ferramentas em VMs diferentes, e calculando as métricas resultantes com base na comunicação entre elas. A ferramenta Badabing é executada em duas VMs, e a cada 5ms realiza o envio de pacotes e cálculo da métrica de perda de pacote.

2.5.4 Sharma et al. (2014)

O trabalho de Sharma et al. (2015) desenvolveu o Hansel, um sistema de detecção de falhas para nuvens baseadas em OpenStack. Este sistema monitora o Domínio de Controle, buscando identificar padrões na comunicação entre serviços que caracterizem a ocorrência de falhas. Durante a execução, o Hansel analisa cada mensagem enviada pelos serviços através de *Remote Procedure Call* (RPC) e REST, buscando algum identificador único pertencente àquela cadeia de eventos. A partir do identificador cria-se um grafo direcionado, que corresponde à uma cadeia de eventos gerado por uma requisição REST. Mensagens RPC são agregadas definidas como auto transições no nó mais recente do grafo, que corresponde à mensagem REST mais recente. Esta agregação de mensagens RPC é necessário por sua quantidade ser maior do que de mensagens REST, e também ao fato de que mensagens REST necessitarem de menos processamento para identificar

o par no qual pertencem. Falhas são diagnosticados ao identificar uma mensagem REST na cadeia de eventos que contém algum código de erro. Ao identificá-la, retorna-se então ao início do grafo para determinar a sequência de eventos que levaram à falha.

A arquitetura de instalação divide o sistema de monitoramento Hansel em duas partes: agentes monitores e serviço de análise central. Para ter acesso a todas as mensagens emitidas, os agentes monitores executam em todos nós de controle e de computação pertencentes à nuvem em questão. Estes agentes monitores foram implementados sobre a ferramenta Bro (PAXSON, 1999), que através da API, chamada Broccoli, possibilitou o *parse* das mensagens do OpenStack em Python. Estes agentes monitores realizam a medição passiva de tráfego, enviando para o serviço de análise central o contexto extraído das mensagens. O contexto é composto pelo identificador da mensagem, que pode ser extraído do UUID ou do identificador atribuído pelo OpenStack, e de outros metadados como: origem e destino da mensagem, protocolo (*e.g.,* REST ou RPC), e o conteúdo da mensagem. A geração dos identificadores é atribuída aos agentes monitores para diminuir a carga no serviço de análise central, que efetua o resto do processamento (*e.g.,* geração dos grafos, identificação de falha). Segundo os autores, o Hansel não depende de um formato específico de mensagem nos protocolos REST e RPC, e portanto pode ser aplicado em outras versões do OpenStack Juno, que foi a versão testada. Durante os testes realizados, o sistema de monitoramento funcionou adequadamente com a transmissão de até 1.600 mensagens REST e RPC por segundo no tráfego do Domínio de Controle.

2.5.5 Análise e comparação dos trabalhos relacionados

Esta seção expõe uma análise sobre os trabalhos relacionados apresentados. Em específico, são apontadas características relevantes a cada trabalho, e após é feita uma comparação entre estas características.

Para evidenciar as características comparadas, a Tabela 2.2 e Tabela 2.3 apresentam os mesmos trabalhos, mas com diferentes características avaliadas. Na Tabela 2.2 são apresentadas três características, na qual a primeira delas define o objetivo final daquele trabalho. A segunda característica apresentada na tabela define qual domínio, conforme apresentado na Seção 2.2.2, este trabalho analisa. A terceira e última característica desta tabela define o ambiente de computação em nuvem utilizado no trabalho. A Tabela 2.3 apresenta outras três características, cuja primeira característica define qual

o tipo de medição que as ferramentas utilizadas empregam. A segunda característica define as métricas utilizadas para caracterizar o tráfego ou avaliar o sistema de monitoramento implementado. A terceira e última característica apresentada nesta tabela são as ferramentas produzidas ou empregadas no trabalho para alcançar o seu objetivo.

Tabela 2.2: Análise comparativa dos trabslhos relacionados: objetivo do trabalho, rede alvo, e ambiente

Trabalho	**Objetivo do trabalho**	**Domínio alvo**	**Ambiente**
Venzano e Michiardi (2013)	Caracterização de tráfego	Domínio de Convidado	OpenStack
Sciammarella et al. (2016)	Caracterização de tráfego	Domínio de Controle	OpenStack
Wang e Ng (2010)	Caracterização de tráfego	Domínio de Convidado	Amazon EC2
Sharma et al. (2014)	Monitoração de tráfego	Domínio de Controle	OpenStack

Fonte: O próprio autor.

Na Tabela 2.2, dos quatro trabalhos analisados, três realizam a caracterização de tráfego de nuvens computacionais como objetivo final, e utilizam ferramentas para medição de tráfego como meio para atingir este objetivo. Um trabalho desenvolveu um sistema para monitoramento de tráfego, na qual realiza a busca por falhas como objetivo final. Esta diferença nos objetivos reflete na aplicação de métricas, que no caso dos trabalhos que caracterizam o tráfego, as métricas são empregadas no processo de caracterização, enquanto que no sistema de monitoramento, as métricas avaliam o desempenho do sistema. Este TCC realizará os dois objetivos, na qual monitorará algumas características da rede em tempo real, e então, fará uma caracterização do tráfego com uma abordagem longitudinal.

Continuando a interpretação da Tabela 2.2, os domínios definidos na coluna "Domínio alvo" remetem à Seção 2.2.2, que por conta da definição genérica, também podem ser aplicados em outros sistemas de gerenciamento de nuvem computacional. Contudo, não há como ter permissão de acesso ao Domínio de Controle em nuvens públicas sem explorar algum tipo de falha no sistema, logo, a análise deste domínio está restrita a nuvens privadas, como as construídas com o OpenStack. A análise do Domínio de Convidado é possível em qualquer nuvem IaaS que o consumidor tenha acesso a VMs e possa executar ferramentas para medir o tráfego na sua rede virtualizada conforme as métricas escolhidas. Neste sentido, no caso de trabalhos no ambiente Amazon EC2, não é possível monitorar e posteriormente caracterizar o tráfego do Domínio de Controle.

Tabela 2.3: Análise comparativa dos trabalhos relacionados: tipo de medição, métricas, e ferramentas

Trabalho	Tipo de medição	Métricas	Ferramentas
Venzano e Michiardi (2013)	Passiva	Uso de CPU; Vazão TCP; Índice de Justiça de Jain	Autoral (implementado sobre Nuttcp)
Sciammarella et al. (2016)	Passiva	Uso de banda por serviço; Uso de banda da rede	Rally; Desconhecida
Wang e Ng (2010)	Ativa	Vazão TCP, Vazão UDP; Atraso; Perda de pacote	Badabing; Ping; Autoral
Sharma et al. (2014)	Passiva	Uso de CPU; Uso de memória	Hansel (implementado sobre Bro)

Fonte: O próprio autor.

Na Tabela 2.3, em ambos os tipos de trabalho foram empregadas ferramentas para realizar a medição do tráfego, pois independente do objetivo, seja caracterização ou apenas monitoramento, é necessário coletar o tráfego. Para realizar o processo de medição de maneira mais rápida e eficiente todos os trabalhos utilizaram ferramentas já conhecidas ou realizaram suas próprias implementações sobre ferramentas já existentes. Em relação às ferramentas prontas, o Badabing e o Rally foram aplicados diretamente na etapa de medição de tráfego, na qual o Badabing foca apenas na métrica de perda de pacote, e o Rally disponibiliza várias métricas, que são escolhidas de acordo com o escopo do trabalho.

Implementações em (SHARMA et al., 2015; VENZANO; MICHIARDI, 2013) foram realizadas sobre duas ferramentas, respectivamente: Bro e `Nuttcp`. O `Nuttcp` foi utilizado para dispensar a preocupação com a lógica em mais baixo nível, relacionado com a coleta de métricas do uso de CPU e interface de rede física. A ferramenta Bro oferece uma plataforma, e que ao usar sua API, é possível enviar e receber mensagens através do seu protocolo, além de poder acessar várias outras funcionalidades relacionadas com o monitoramento do tráfego de rede. O trabalho (SCIAMMARELLA et al., 2016) não descreveu a ferramenta empregada na realização de dois cenários durante a caracterização de tráfego, na qual envolviam analisar o uso de banda de certas aplicações e o uso de banda geral da rede, respectivamente.

As métricas escolhidas nos trabalhos, apesar de semelhantes, são aplicadas em contextos diferentes, e neste sentido impossibilita uma comparação direta. O trabalho de Venzano e Michiardi (2013) focou no contexto de aplicações distribuídas, e avaliou como o local na qual as VMs estão alocadas influenciam no desempenho resultante. Em relação à Wang e Ng (2010), a análise é mais elementar, cujo objetivo é descobrir qual a qualidade dos recursos de rede disponibilizados conforme o serviço contratado da EC2. As métricas adotadas por Sciammarella et al. (2016) avaliam qual o uso de recurso de rede de uma

nuvem conforme a escala da nuvem. Com os valores base definidos, é possível estimar até onde uma nuvem pode crescer e oferecer um serviço de qualidade. As métricas adotadas em Sharma et al. (2015) avaliam o uso de recursos pelo sistema de monitoramento, cujo um valor alto, por exemplo, aponta que monitorar a rede pode não ser viável, e valores baixos indicam sua viabilidade.

Este TCC pretende caracterizar o comportamento do Domínio de Controle, e portanto, parte da análise será voltada para métricas de desempenho. A outra parte da análise pretende avaliar comportamentos que ocorrem no Domínio de Controle, mas que talvez não sejam documentados pelo OpenStack, e portanto, não podem ser expressos em métricas.

2.6 Considerações do capítulo

Este capítulo conceituou computação em nuvem, o sistema de gerenciamento de nuvem OpenStack e abordagens existentes na caracterização de tráfego. Após apresentar sobre computação em nuvem detalha-se sobre como é realizada a virtualização e qual a sua influência no funcionamento de recursos computacionais. Então, após uma introdução sobre o OpenStack, são descritos alguns serviços que fornecem funcionalidades para a nuvem e a arquitetura de rede recomendada em uma instalação. A última parte da conceitualização expõe uma maneira de abordar a caracterização de tráfego, dividindo-a em duas etapas: medição de tráfego e análise de tráfego.

As seções após a conceitualização descrevem sobre o escopo do problema abordado neste TCC, relacionando-os com os conceitos apresentados. A primeira seção relacionada ao escopo do trabalho define o problema a ser tratado nesse TCC, na qual exemplifica-o aplicando alguns dos conceitos apresentados. Então, para fundamentar melhor o escopo deste TCC, são analisados características e ferramentas contidas em trabalhos relacionados, cuja análise auxilia na criação de uma proposta.

3 Proposta para caracterização de tráfego

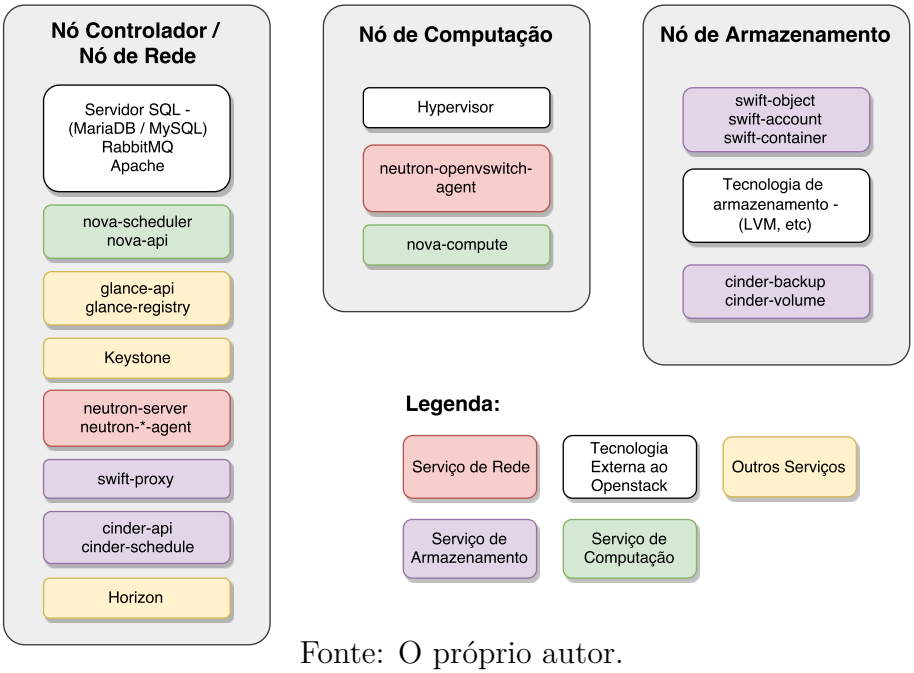
Para realizar a caracterização de tráfego em uma rede é necessário coletar/medir o tráfego para posterior análise, conforme descrito na Seção 2.3. Na fase de medição de tráfego é possível aplicar ferramentas existentes (*e.g.,* `Tcpdump`, Bro, `ping`) para recolher os dados que serão utilizados na fase de análise. Contudo, por serem ferramentas amplamente difundidas e com propósito geral, estas geram dados genéricos (*e.g.,* pacote inteiro), que não captam apenas os aspectos desejados para posterior análise. Sendo assim, a criação de uma ferramenta de monitoramento possibilita recolher apenas os dados pertinentes a análise em questão.

Contudo, ao tratar-se da caracterização envolvendo um software complexo como o OpenStack, é necessário conhecer pelo menos a arquitetura de funcionamento dos serviços envolvidos no processo de caracterização de tráfego, conforme apresentado na Seção 3.1. Pois conhecendo o funcionamento dos serviços do OpenStack é possível entender como capturar o tráfego de interesse, quais são os seus aspectos analisáveis, e quais as melhores abordagens a adotar (Seção 3.2). Tendo estas informações em mãos, pode-se definir os requisitos para a criação do sistema de monitoramento, conforme exibido na Seção 3.3. A partir dos requisitos e das observações sobre o ambiente do OpenStack, na Seção 3.4 é estabelecida a arquitetura do sistema de monitoramento. Algumas informações geradas pelo sistema de monitoramento podem ser aplicadas diretamente para entender o tráfego, permitindo sua análise em tempo real. Contudo, certas informações necessitam de análises minuciosas, cujo processo, por exemplo, precisa que sejam feitas comparações com outras informações coletadas a fim de gerar dados significativos. A fase seguinte à coleta realizada pelo sistema de monitoramento é a de análise de tráfego, cuja abordagem para o problema proposto é definida na Seção 3.5. Sendo explicado o que é feito com os dados coletados, e qual o tipo de informação resultante da análise. Por fim, define-se na Seção 3.6 como será aplicado o sistema de monitoramento, na qual explica-se os cenários de aplicação e os detalhes referentes à realização dos experimentos.

3.1 Funcionamento do OpenStack

Conforme apresentado na Seção 2.2, a instalação do OpenStack pode distribuir os serviços entre vários hosts. Segundo a arquitetura sugerida na documentação oficial (The OpenStack project, 2017g), são atribuídas responsabilidades específicas para cada host da nuvem, classificados em: armazenamento, computação, controle, e de rede. Estas categorias são definidas conforme os serviços executados no host, e dependendo do tamanho da nuvem, um host pode pertencer a múltiplas categorias, sendo responsável por controle e armazenamento, por exemplo. No caso de nuvens maiores esta centralização é evitada por questões de desempenho. A Figura 3.1 apresenta uma arquitetura de instalação conceitual do OpenStack, na qual um dos hosts opera como nó de controle da nuvem e nó de rede simultaneamente. Esta nomenclatura segue a utilizada na documentação do OpenStack, na qual ao referenciar-se às atribuições do host, coloca-se a palavra nó na frente, referenciando um host com serviços relacionados ao Neutron, por exemplo, como nó de rede.

Figura 3.1: Arquitetura conceitual de instalação de componentes do OpenStack



Fonte: O próprio autor.

Para explicitar melhor o papel de cada uma das categorias de host no tráfego de controle, esta seção apresenta brevemente como funcionam alguns serviços que executam nestes hosts, com foco na suas respectivas arquiteturas. Os serviços apresentados nesta seção são os considerados no monitoramento e posterior análise do tráfego, conforme definido na Tabela 2.1, com exceção do Horizon, que é basicamente uma interface web que acessa as APIs dos outros serviços, e não será abordado na explicação. Esta limitação de

serviços abordados visa simplificar o processo de caracterização, e segue a recomendação de serviços populares em uma instalação do OpenStack, segundo (The OpenStack project, 2017l).

Nem todos os serviços do OpenStack têm a mesma complexidade, e neste sentido, alguns serviços como Nova e Neutron destacam-se pela complexidade. Portanto, nesta seção alguns serviços mais complexos são abordados em mais detalhes do que outros com arquitetura simples (*e.g.,* Keystone, Swift). Além dos serviços, também é apresentada a ferramenta RabbitMQ, que possui papel importante na comunicação entre componentes pertencentes a certos serviços. Esta seção inicia com o Nova, um dos serviços principais no OpenStack, que gerencia o ciclo de vida das VMs e se relaciona com vários dos serviços do OpenStack para dispor suas funcionalidades (Figura 2.3).

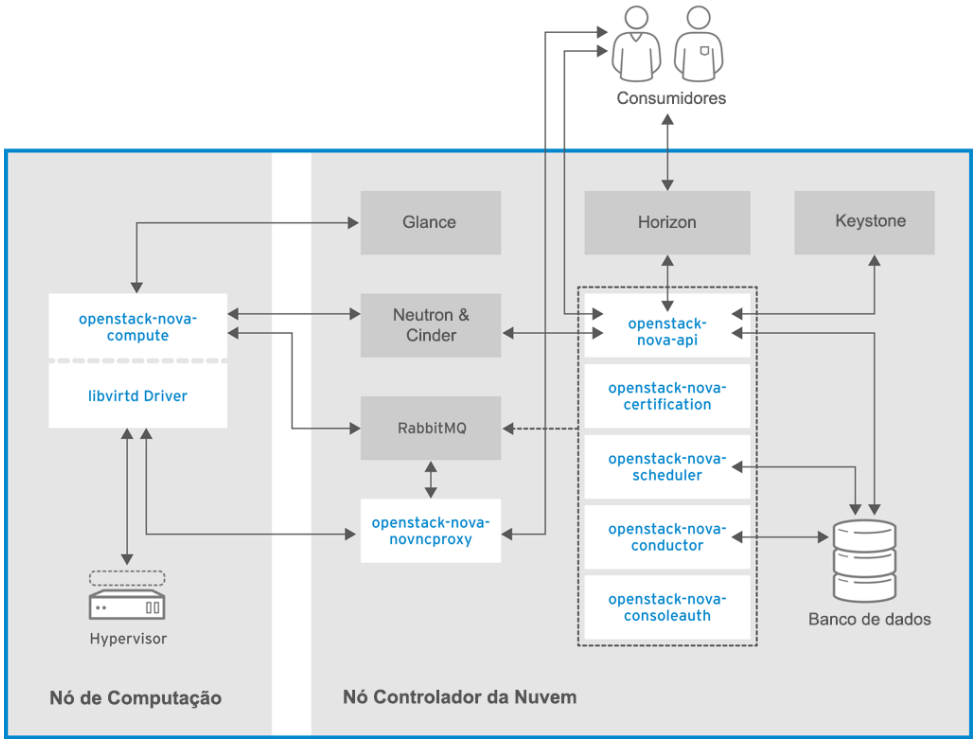
3.1.1 Nova

O Nova é o serviço de computação do OpenStack, o qual gerencia os nós de computação existentes na nuvem, controlando os *hypervisors* instalados nestes nós, e consequentemente, as VMs em execução. Versões anteriores do OpenStack também incluem componentes relacionados à configuração de rede no Nova (`nova-network`), que disponibiliza um modelo de gerenciamento de rede mais limitado quando comparado ao Neutron, e portanto foi descontinuado. Segundo Red Hat (2017), as versão recentes do Nova utilizam uma arquitetura que divide-se em diversos componentes, conforme ilustrado na Figura 3.2.

Apesar dos vários componentes pertencentes ao serviço Nova (Figura 3.2), este trabalho foca em alguns componentes principais: API, escalonamento, computação, banco de dados, e *message broker* (AMQP Server).

Conforme apresentado na Seção 2.2, a API do Nova (`nova-api`) é responsável por disponibilizar acesso e tratar requisições para o serviço Nova. O `nova-api` em sua essência é um *Web Service* implementado sobre o protocolo REST, e gerencia autorização e funções básicas de controle, na qual implementa modelos de API compatíveis como o da Amazon e Rackspace. Existem diferentes versões da REST API do Nova, que introduzem novas funcionalidades ou padronizações, o OpenStack Newton por exemplo, que é a versão que este TCC irá trabalhar, tem a REST API do Nova na versão 2.38 (The OpenStack project, 2017f). Toda requisição recebida pelo `nova-api` é avaliada para verificar se os

Figura 3.2: Arquitetura dos componentes pertencentes ao serviço Nova



Fonte: Adaptado de (Red Hat, 2017).

recursos referenciados na requisição estão disponíveis, e caso estiverem disponíveis, a requisição é enviada ao *message broker*, que é um servidor *Advanced Message Queuing Protocol* (AMQP), e disponibiliza a mensagem para os recursos referenciados acessarem (The OpenStack project, 2017h). O *message broker* é um *middleware* com o protocolo AMQP utilizado para comunicação interna entre componentes de certos serviços, que no caso do Nova por exemplo, se aplica aos componentes `nova-scheduler` e `nova-compute`, e também intermedeia o acesso ao banco de dados pelo `nova-compute`.

Durante a troca de mensagens, o *message broker* também atualiza o banco de dados do Nova, que mantém informações sobre o estado corrente da nuvem, como por exemplo, o números de VMs executando em cada nó de computação da nuvem (Red Hat, 2017). Este banco de dados é implementado em MySQL e é acessível pelos componentes do Nova, o qual centraliza as informações que os componentes necessitam durante sua execução. Após receber a mensagem contendo o pedido de VM pelo *message broker*, o componente responsável pelo escalonamento, chamado `nova-scheduler` define qual nó de computação hospedará uma nova instância de VM. Para definir o nó de computação é consultado o banco de dados do Nova, que contém informações aplicáveis no processo de filtragem para encontrar um nó de computação disponível para hospedar a VM. Neste sentido, é possível fornecer pesos para as métricas, ajudando na escolha do nó de com-

putação, como também é possível deixar a escolha aleatória entre os nós de computação filtrados no processo, que correspondem aos nós de computação podendo hospedar aquela VM (The OpenStack project, 2017h).

No caso de uma nuvem de pequeno porte, os componentes do Nova exibidos até aqui executam em um mesmo nó de controle, pois não necessitam de grande quantidade de processamento. Em contraste, o componente `nova-compute` executa em cada host que contém um *hypervisor*, ou seja, um nó de computação, e é responsável por gerenciá-los. Como todos os outros componentes do Nova que usam AMQP, o `nova-compute` em execução verifica periodicamente por tarefas enfileiradas para ele, e realiza-as conforme solicitado (*e.g.,* criar instância, finalizar instância, ler saída do *console*). Se outros serviços que dispõem funcionalidades extras às VMs estiverem instalados (*e.g.,* Cinder, Glance, Neutron), cada `nova-compute` realiza a comunicação direta com estes serviços.

3.1.2 Cinder

Para disponibilizar armazenamento persistente às VMs, o serviço Cinder permite acoplar volumes/blocos de armazenamento nas VMs em execução. Além de disponibilizar armazenamento para VMs, o Cinder também fornece a habilidade de criar *snapshots* dos volumes de armazenamento, que copia o conteúdo do volume, e possibilita gerar novos volumes contendo os dados deste *snapshot*. Mesmo sendo mais simples que o Nova, o Cinder também emprega múltiplos componentes para realizar suas funções, que podem ser distribuídos entre diferentes hosts. Alguns componentes com comportamento similar aos componentes do Nova estão presentes: escalonador, banco de dados, API e *message broker*; em que difere-se nos componentes responsáveis pelo armazenamento: `cinder-backup` e `cinder-volume`.

O papel do `cinder-api`, componente que gerencia a API do Cinder, é disponibilizar uma interface REST para que os outros serviços do OpenStack, como o Nova, acessem suas funcionalidades (Red Hat, 2017). A REST API do Cinder encontra-se na v3.15 no OpenStack Newton (The OpenStack project, 2017f). Após recebido o pedido pelo `cinder-api`, ele é enviado ao *middleware* responsável pela comunicação entre os componentes, baseado no protocolo AMQP, e que não necessita ser de uso exclusivo do Cinder.

O banco de dados, baseado em MySQL, tem propósito similar ao banco de

dados no Nova: compartilhar informações relacionadas ao estado atual do serviço para os componentes, na qual também pode usar o mesmo servidor de banco de dados. O escalonador `cinder-scheduler` decide onde armazenar volumes e backups criados, cuja funcionalidade é disponibilizada pelos componentes `cinder-volume` e `cinder-backup`, que podem ser hospedados em múltiplos hosts, servindo como nós de armazenamento. Segundo Red Hat (2017), estes dois componentes podem usar várias soluções de armazenamento (*e.g.,* Ceph, LVM, NFS), e em especial o `cinder-backup` é capaz de armazenar seus backups no Serviço de armazenamento Swift.

3.1.3 Swift

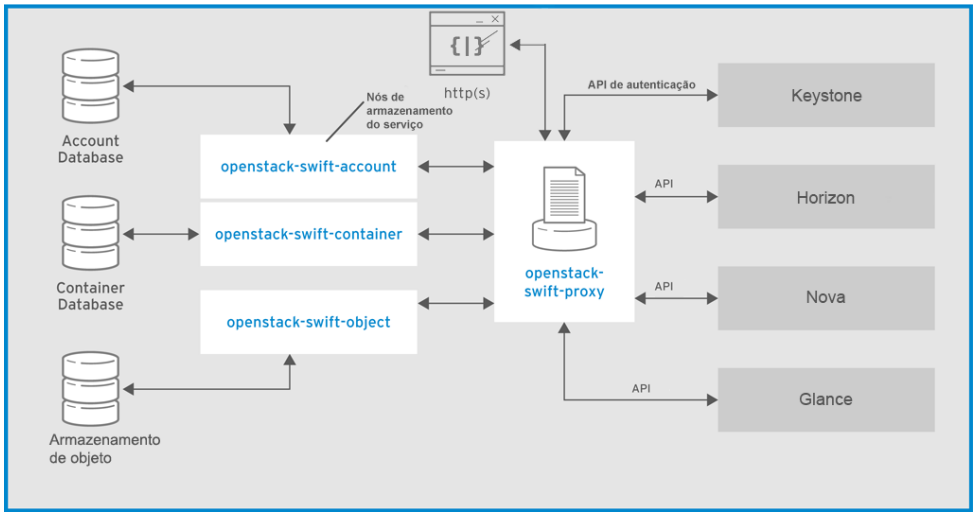
O Swift é outro serviço do OpenStack responsável pelo armazenamento, mas com o foco em armazenar dados estáticos de grande volume (*e.g.,* vídeos, imagens, imagens de VM). O acesso aos dados é feito com o um pedido HTTP (`Get, Put, Delete`) para o `swift-proxy`, que é o componente do Swift responsável pela API, e autenticação no serviço, que no OpenStack Newton encontra-se na v1 (The OpenStack project, 2017f). Ao receber uma requisição é consultado a localização do objeto nos outros componentes (`swift-account`, e `swift-container`, respectivamente), roteando então a requisição de acordo com as informações consultadas, conforme ilustrado na Figura 3.3.

O Swift tem dois componentes que gerenciam acesso a listagem de informações: o `swift-account`, que gerencia a lista de contêineres de armazenamento no banco de dados, e o `swift-container`, cuja tarefa é gerenciar a lista de objetos contidos nos contêineres, também armazenado no banco de dados. Para gerenciar os objetos armazenados, o componente `swift-object` armazena, recupera e exclui objetos conforme requisitado (The OpenStack project, 2017j). A parte de autenticação do `swift-proxy` comunica-se com o serviço Keystone, que fornece uma API que centraliza o processo de autenticação do OpenStack.

3.1.4 Keystone

O serviço Keystone fornece autenticação e autorização aos serviços do Openstack. Este serviço possibilita a autenticação através de múltiplos mecanismos, como por exemplo, credencial com usuário e senha, ou através de *tokens* (Red Hat, 2017). Desta maneira, é

Figura 3.3: Arquitetura dos componentes pertencentes ao serviço Swift



Fonte: Adaptado de (Red Hat, 2017).

possível combinar estes mecanismos no uso do Keystone, no qual um pedido de autenticação é validado através de usuário e senha, e então, o serviço pode retornar um *token* ao solicitante, que é utilizado para a autenticação posteriormente (The OpenStack project, 2017k).

A arquitetura de funcionamento do Keystone é relativamente simples, possuindo apenas um componente principal, que é responsável pela API do serviço, e encontra-se na v3 no OpenStack Newton. A API do Keystone armazena as identidades, autorizações, catálogos de serviços e *tokens* no MySQL (MariaDB) por padrão, podendo-se adicionar outras tecnologias para cuidar de partes específicas do serviços. Neste sentido, a tecnologia *Lightweight Directory Access Protocol* (LDAP) pode ser empregada no gerenciamento de autenticações e *memcache* ou *Redis* para gerenciar a persistência dos *tokens* (Red Hat, 2017). Após autenticado o usuário (consumidor ou administrador), ele é capaz de acessar o catálogo fornecido pelo Keystone, cuja funcionalidade é armazenar os serviços em funcionamento no Openstack, e informar ao usuário quais recursos ele pode acessar.

3.1.5 Glance

O serviço Glance funciona como um registro de imagens de VMs acessível pelos consumidores da nuvem. É possível que os consumidores adicionem novas imagens, desde que estas sejam em um dos formatos aceitos pelo serviço, que segundo Red Hat (2017), inclui `iso`, VMs do VirtualBox (`vdi`), VMware (`vmdk`) e Amazon EC2 (`ami`). Outra alternativa é gerar um *snapshot* a partir de uma das VMs do consumidor, que pode servir como *backup*

ou imagem para iniciar outras VMs.

Em relação à sua arquitetura, o Glance é dividido nos seguintes componentes principais (The OpenStack project, 2017d): `glance-api`, `glance-registry`, repositório de armazenamento e banco de dados (MySQL/MariaDB). A API do Glance (`glance-api`) gerencia todos os pedidos para recuperação e armazenamento de imagens, que no OpenStack Newton tem a API v2 (The OpenStack project, 2017f). Enquanto os pedidos são processados, o componente `glance-registry` acessa o banco de dados instalado para buscar informações da imagem em questão (*e.g.,* metadados da imagem, local de armazenamento). As imagens registradas neste serviço podem ser armazenadas no serviço Swift ou através de outras tecnologias como RADOS Block Service, VMware datastore, HTTP, ou até como arquivo no próprio sistema de arquivos do host do Glance.

3.1.6 Neutron

Conforme apresentado na Seção 2.2.2, uma das funcionalidades do serviço Neutron é fornecer rede às VMs dos consumidores, incluindo a possibilidade de configurar a rede em questão, sub-redes e os seus roteadores (Red Hat, 2017). Outras funcionalidades adicionais também podem ser oferecidas para a rede criada em um *project/tenant*, como DHCP (`neutron-dhcp-agent`), *Firewall* ou *Virtual Private Networks* (VPNs).

Comparado com os outros serviços do OpenStack, o Neutron é um dos mais complexos, podendo envolver diversas tecnologias de rede e possibilidades de configuração (DENTON, 2016). No geral, o Neutron tem os seguintes componentes: *Network agents*, que inclui os componentes `neutron-dhcp-agent`, `neutron-openvswitch-agent` e `neutron-l3-agent`; `neutron-server` e *message broker* (The OpenStack project, 2017e). Os *Network agents* executam em todos os hosts de uma nuvem OpenStack, e são responsável por cuidar, por exemplo, da configuração das VMs contidas no host e também de serviços e *plugins* em execução no Neutron, como o Open vSwitch (`neutron-openvswitch-agent`). O Open vSwitch é uma das tecnologias que pode ser empregada na virtualização das interfaces de rede físicas dos hosts, e permite criar VLANs para isolar o tráfego de cada rede. Assim, é possível que o tráfego de dois domínios diferentes trafeguem no mesmo meio físico com isolamento.

O componente `neutron-ml2` (Modular Layer 2) é obrigatório no Neutron, e controla a Camada 2 da rede, ou seja, encaminha pacote apenas dentro das redes criadas

pelos consumidores. A nível estrutural, o `neutron-ml2` funciona como um *framework* que possibilita a criação de *plugins* que relacionam-se com a Camada 2 da rede, como o TaaS (*Tap-as-a-Service*)[1], por exemplo. O TaaS possibilita a criação de espelhamentos de portas das redes dos consumidores da nuvem, que tornam-se acessíveis através de portas virtuais criadas pelo TaaS no Open vSwitch, cuja aplicação inclui, por exemplo, o monitoramento do tráfego com um *Intrusion Detection System* (IDS). Para fornecer acesso à rede externa um *L3 Network agent* (`neutron-l3-agent`) deve executar no nó de rede da nuvem, e responsabilizar-se pelo roteamento dos pacotes para fora da rede do consumidor, que corresponde à função do roteador virtual inserido pelo consumidor na sua rede (Red Hat, 2017). O agente DHCP (`neutron-dhcp-agent`) é outro agente que executa no nó de rede, e fornece o serviço de DHCP para as redes dos consumidores.

Por fim, o componente `neutron-server` é responsável pela API do serviço, que age similar aos outros serviços, na qual envia as mensagens para o *message broker*, que encaminha-as para os outros agentes e *plugins* do Neutron. Através da sua API os consumidores definem as configurações das suas redes, e o administradores definem as tecnologias de redes empregadas nos serviços do Neutron (DENTON, 2016). A API do Neutron encontra-se na v2 atualmente, e é a mesma utilizada no OpenStack Newton, na qual baseia-se em um aprimoramento da API do Quantum (serviço de rede descontinuado) na v1.1 (The OpenStack project, 2017f).

3.1.7 RabbitMQ

Conforme apresentando nos serviços Neutron, Cinder e Nova, é necessário haver um servidor responsável pela comunicação interna destes serviços, ou seja, entre os seus componentes. O OpenStack emprega o protocolo AMQP para realizar esta comunicação interna, cujas mensagens são criadas no padrão do protocolo *Remote Procedure Call* (RPC), que é implementado pelo projeto Oslo. Neste sentido, o servidor AMQP (*e.g.,* RabbitMQ, Qpid, ZeroMQ), também chamado de *message broker* apenas transporta a mensagem, seguindo o padrão *publish/subscribe*. A utilização deste *middleware* para comunicação tem como objetivo (The OpenStack project, 2017b): desacoplar o host de origem e destino da mensagem; completo assincronismo entre os hosts (não é necessário que o destino esteja disponível ao enviar mensagem); balanceamento entre chamadas remotas (mensagem é

[1] `https://github.com/openstack/tap-as-a-service`

recebida pelo primeiro host disponível a acessar o RabbitMQ).

Neste meio de comunicação exitem dois tipos de chamadas: *RPC Call* e *RPC Cast* (The OpenStack project, 2017b; The OpenStack project, 2017i). No *RPC Call*, é enviada uma mensagem do host, e então cria-se um *Consumer* neste host, que escutará a resposta da execução remota, sem efetuar bloqueio no processo de espera. Para saber a quem cada resposta pertence, um campo `msg_id` é criado ao enviar a primeira mensagem, e assim o *Publisher* sabe que aquela resposta é para ele. No *RPC Cast*, o método é invocado remotamente, mas sem escutar a resposta de quem o executou. O envio pelo *Publisher* e recebimento de mensagens pelo *Consumer* é separado por tópico, e só os hosts com certo componente ou serviço em execução recebe mensagens de certo tópico. A arquitetura do OpenStack não diferencia os hosts conectados ao RabbitMQ, contudo, é possível verificar uma diferença em comportamento neles, na qual alguns hosts enviam mais mensagens, enquanto outros ficam responsáveis por receber e processar as mensagens. Este tipo de comportamento pode ser verificado ao comparar hosts que executam o `nova-api` e o `nova-compute`, por exemplo, em que a API repassa tarefas a serem realizadas para outros componentes do Nova, como o `nova-compute`, que fica à escutar por novas tarefas que deva executar.

3.1.8 Considerações OpenStack

Conhecidos alguns serviços do OpenStack, o próximo passo para a criação do sistema de monitoramento é analisar aspectos específicos de nuvens OpenStack de maneira que auxilie a definir o funcionamento do sistema de monitoramento proposto. Ou seja, objetivo da análise do ambiente de caracterização é definir algumas diretrizes úteis posteriormente: qual é o tráfego de interesse, quais são algumas das características observáveis neste tráfego de interesse, e quais são os locais da rede com maior concentração deste tráfego.

3.2 Ambiente de caracterização

A Seção 3.1 apresenta a arquitetura de alguns serviços do OpenStack incluídos na caracterizar de tráfego deste TCC. Sendo assim, a partir de suas características de funcionamento podem ser definidas estratégias para criar o sistema de monitoramento responsável por medir o tráfego e o analisar (Seção 2.3). O propósito desta seção é exibir características do

OpenStack apresentadas neste TCC que podem ser exploradas na construção do sistema de monitoramento.

As APIs dos serviços do OpenStack são responsáveis por expor suas funcionalidades aos consumidores da nuvem e a outros serviços. Neste sentido, toda ação realizada na nuvem por algum consumidor inicia-se através da comunicação com a API. Após a realização de um pedido através da API, o serviço correspondente retorna uma mensagem informando sobre o *status* da ação realizada (*e.g.,* sucesso, falha, informações à respeito). Logo, é possível descobrir se o consumidor foi capaz de realizar certa ação ao olhar a resposta recebida. Sendo assim, monitorar as mensagens que passam pelas APIs em busca de comportamentos de interesse é uma maneira de entender o que ocorre na nuvem. A Figura 3.4 apresenta uma resposta ao realizar uma consulta para a API do Nova (`nova-compute`) solicitando a lista de volumes fornecidos pelo Cinder que estão acoplados em uma certa VM.

Figura 3.4: Exemplo de resposta recebida ao consultar volumes de armazenamento acoplados a uma instância do Nova


```
{
    "volumeAttachments": [
        {
            "device": "/dev/sdd",
            "id": "a26887c6-c47b-4654-abb5-dfadf7d3f803",
            "serverId": "4d8c3732-a248-40ed-bebc-539a6ffd25c0",
            "volumeId": "a26887c6-c47b-4654-abb5-dfadf7d3f803"
        },
        {
            "device": "/dev/sdc",
            "id": "a26887c6-c47b-4654-abb5-dfadf7d3f804",
            "serverId": "4d8c3732-a248-40ed-bebc-539a6ffd25c0",
            "volumeId": "a26887c6-c47b-4654-abb5-dfadf7d3f804"
        }
    ]
}
```


Fonte: (The OpenStack project, 2017a).

A resposta segue o padrão REST, e é formatado com JSON, na qual pode-se aplicar ferramentas de *parse* ou análise com expressão regular para extrair informações. No caso de falha, por exemplo, esta mensagem no protocolo HTTP poderia retornar algum código de erro, como o código 401 (`unathorized`) caso o solicitante não tenha permissão para acessar a informação (The OpenStack project, 2017a).

A partir do monitoramento da API é possível visualizar a requisição de uma ação na nuvem e a resposta resultante do sistema. Ou seja, enxerga-se a nuvem como uma caixa preta, observando a entrada e a saída resultante da API. Para aumentar a quantidade de informações disponíveis, é possível observar o conteúdo desta "caixa preta" através do monitoramento da comunicação entre componentes de um serviço.

Serviços complexos como o Nova, Neutron e Cinder possuem componentes distribuídos pela nuvem, e a responsabilidade da comunicação entre estes componentes fica a cargo do *middleware* de comunicação (RabbitMQ). Conforme apresentado na Seção 3.1, quando um componente de algum serviço como o Nova enviar uma mensagem RPC para o RabbitMQ, a mensagem é recebida por um componente com interesse nela (tópico da mensagem). Contudo, uma mensagem enviada para o RabbitMQ pode esperar certo tempo até ser recebida por algum destino, pois algum dos destinos em potencial deve verificar se há alguma mensagem de interesse disponível para receber do *middleware* de comunicação. Sendo assim, ao monitorar as mensagens **saindo** do RabbitMQ pode-se ter certeza que aquelas mensagens serão processadas logo em seguida pelo destino.

O sistema de monitoramento de (SHARMA et al., 2015) analisa o tráfego RPC e REST, com o objetivo de organizá-lo numa sequência de mensagens que representa um evento na nuvem. A sequência é criada ao encontrar várias mensagens com o mesmo identificador, que segundo (SHARMA et al., 2015), o OpenStack utiliza para controle interno. Portanto, é possível relacionar requisições dos consumidores da nuvem e as subsequentes comunicações entre os componentes para executá-las. Porém, uma dificuldade encontrada por (SHARMA et al., 2015) ao analisar mensagens RPC é a ausência de uma maneira simples de indicar a presença de erro. Uma estratégia de uso para empregar ambos os protocolos pode consistir da análise de mensagens REST em direção à API, e então de acordo com o tipo de ação, caso houver necessidade de acompanhar o processo detalhadamente, pode-se analisar o tráfego RPC gerado em função desta requisição REST.

Serviços do OpenStack executam por padrão na rede de controle com as portas definidas segundo a Tabela 2.1, e não mudam durante a execução. Logo, é possível associar cada pacote de controle na nuvem ao serviço do OpenStack de destino através da porta de destino definida no cabeçalho do pacote. Mesmo este método de classificação sendo simples, com ele é possível realizar uma medição que classifica, por exemplo, os serviços do OpenStack que recebem mais mensagens. Contudo, para garantir que a classificação apresente resultados significativos é necessário isolar o tráfego de armazenamento.

A justificativa é que a transferência de uma imagem de VMs ou de um volume de armazenamento, por exemplo, é feita com vários pacotes, podendo impactar significativamente a medição realizada.

Em relação ao processo de coleta de tráfego, ou seja, na fase de medição do tráfego da rede de controle, é necessário que a abordagem meça o tráfego gerado pelo OpenStack. Com isso em mente, a medição ativa possui pouca aplicabilidade, pois o princípio desta abordagem é justamente depender apenas do tráfego gerado pelo próprio software de monitoramento, descartando a necessidade de conhecer características da rede (*e.g.,* topologia, comportamento) cuja medição será realizada. Portanto, no caso da rede de controle de uma nuvem OpenStack, por exemplo, a medição de tráfego passiva é a melhor escolha, visto que diferente da abordagem ativa, a medição passiva coleta todo o tráfego no ponto de medição, que na rede de controle é gerado apenas pelo OpenStack e tecnologias relacionadas.

Escolhida a abordagem de medição passiva, o próximo passo é decidir sobre o ponto de medição, que pode ser distribuído em vários pontos da rede dependendo da necessidade. Considerando o que foi apresentado sobre o tráfego RPC e REST no começo desta seção, por exemplo, deve-se escolher um ou mais pontos de coleta de maneira que a medição cubra o máximo de tráfego de interesse quanto possível (*e.g.,* mensagens em RPC e REST). Neste sentido, conforme ilustrado na Figura 3.1, os serviços do OpenStack e outras tecnologias relacionadas com as APIs e comunicação interna de serviços (*e.g.,* RabbitMQ) são distribuídas entre nós de rede e de controle da nuvem. Logo, os nós de rede e de controle de uma nuvem OpenStack são potencialmente bons locais para posicionar a parte responsável pela medição de tráfego no sistema de monitoramento.

3.3 Especificação de requisitos

Com o objetivo de solucionar o problema de caracterização de tráfego, abordado na Seção 2.4, esta seção inicia a proposta de solução apresentando alguns requisitos que devem ser considerados na proposta. Estes requisitos estão associados à primeira parte do processo de caracterização de tráfego proposto, que é a medição de tráfego, e que será de responsabilidade do sistema de monitoramento. Sendo assim, existem pré-requisitos para possibilitar que o sistema de monitoramento realize a medição de tráfego corretamente:

- Ter acesso ao tráfego que passa pelas interfaces de rede, físicas e virtuais, da nuvem monitorada;
- Um dispositivo de monitoramento necessita estar conectado à rede de controle da nuvem que pertence; e
- Existir um banco de dados de dados acessível para armazenar os dados gerados.

Os requisitos são abstrações de funcionalidades que monitoram aspectos observáveis na rede de controle. Estes aspectos monitorados serão consultados na fase de análise do tráfego. Os requisitos que devem possibilitar a medição do tráfego da rede de controle, a fim de gerar informações sobre aspectos observáveis do tráfego são divididos em requisitos funcionais e requisitos não funcionais:

Requisitos funcionais:

- **RF1.** Contabilizar o consumo de tráfego na rede de controle dos serviços em execução;
- **RF2.** Armazenar a comunicação entre os componentes dos serviços em execução;
- **RF3.** Armazenar as requisições recebidas pelas APIs, originárias tanto de consumidores quanto de outros serviços; e
- **RF4.** Os dados devem ser detalhados, de maneira que eles revelem tarefas responsáveis por variações na nuvem.

Requisitos não funcionais:

- **RNF1.** Operar independente do OpenStack, não sendo atrelado a uma implementação de nuvem apenas; e
- **RNF2.** O sistema de monitoramento não pode impactar no desempenho da nuvem sendo analisada.

3.4 Proposta de sistema de monitoramento

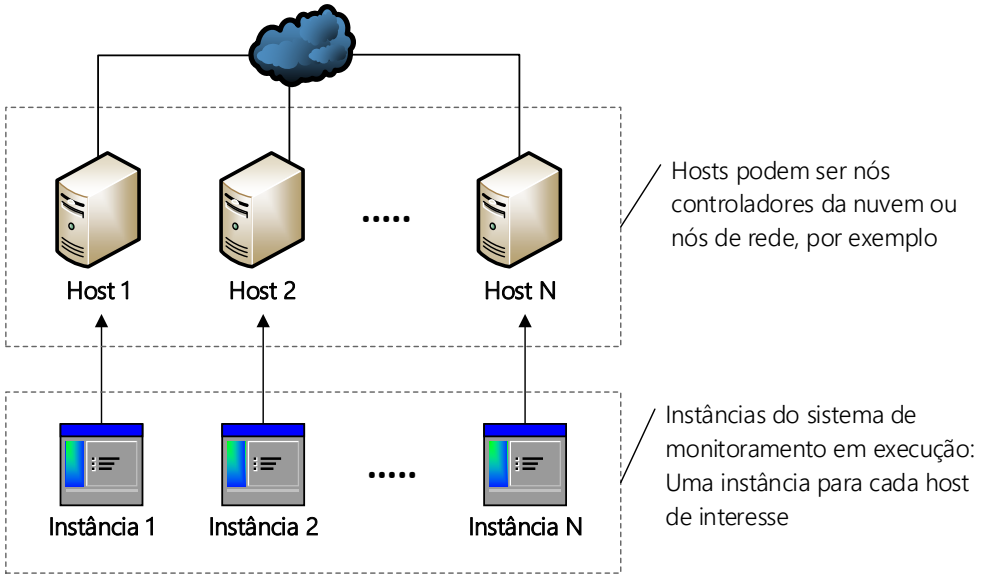
Definido o tráfego de interesse, como obter este tráfego (Seção 3.2), e os requisitos do sistema de monitoramento (Seção 3.3), o próximo passo é estipular a arquitetura do

sistema de monitoramento que auxiliará na caracterização de tráfego da rede de controle. A princípio, este sistema de monitoramento terá três funcionalidades, que são baseadas nos requisitos funcionais especificados:

- **F1.** Analisar o consumo de banda na rede de controle, classificando em função do serviço que receberá o tráfego (RF1);
- **F2.** Registrar requisições nas APIs dos serviços, que originam tanto de outros serviços como de consumidores (RF3); e
- **F3.** Registrar transações internas da nuvem, que trafegam pelo *middleware* de comunicação (RabbitMQ) (RF2).

As funcionalidades devem coletar o tráfego de controle através da medição passiva. Portanto, instâncias do sistema de monitoramento executarão em cada um dos hosts de interesse (pontos de medição escolhidos explicado na Seção 3.2), conforme ilustrado na Figura 3.5.

Figura 3.5: Arquitetura de instalação do sistema de monitoramento



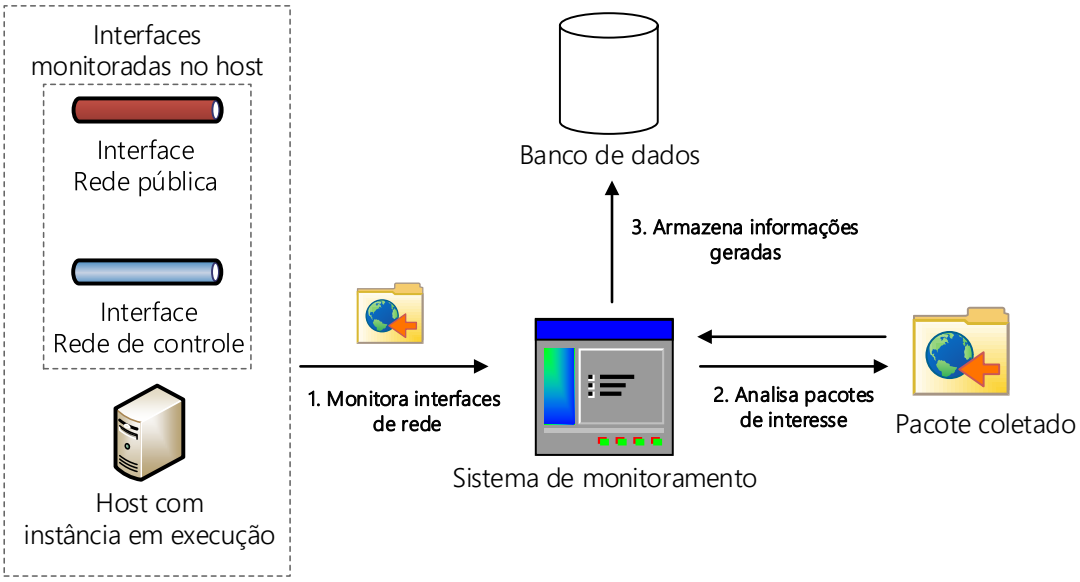
Fonte: O próprio autor.

Apesar de nem todos os hosts hospedarem APIs dos serviços (F2) ou o RabbitMQ (F3), pode haver interesse em monitorar tal host caso o tráfego de controle de algum serviço de interesse origine ou vá para ele (F1). A decisão se o pacote está relacionado com algum serviço de interesse é baseada na porta de destino, seguindo as portas por padrão na Tabela 2.1. Para realizar a coleta de tráfego, este sistema de monitoramento

será desenvolvido em uma linguagem de alto nível que forneça métodos relacionados à biblioteca `libpcap`. Esta biblioteca disponibiliza meios para sistemas UNIX capturarem tráfego em suas interfaces de rede, sendo elas virtuais ou físicas. Das linguagens possíveis, a preferência é para Python, por dispor de diversas bibliotecas e funcionalidades que simplificam o desenvolvimento, mas podendo mudar caso desempenho torne-se um fator crítico, seguindo o requisito RNF2.

Abordando o comportamento do sistema em alto nível é possível abstrair comportamentos comuns às três funcionalidades, conforme ilustrado na Figura 3.6.

Figura 3.6: Arquitetura do sistema de monitoramento em alto nível



Fonte: O próprio autor.

O sistema de monitoramento executará em um host de interesse, coletando tráfego nas interfaces de rede de acordo com as funcionalidades do sistema de monitoramento em execução naquele host, respeitando o requisito RNF1 (passo 1 na Figura 3.6). Caso um host não hospede APIs de serviços, ou o *middleware* de comunicação, por exemplo, é possível executar o sistema de monitoramento apenas para monitorar o tráfego na interface da rede de controle, pois é por onde passa o tráfego dos serviços de interesse, que é registrado pela F1. Apesar de existirem outros níveis de agregação de tráfego, neste sistema de monitoramento é necessário realizar a medição a nível de pacote. A agregação a nível de fluxo, por exemplo, apesar de popular, omite detalhes relevantes para este sistema de monitoramento, como o conteúdo do pacote. Outro motivo é que apenas um pacote pode ser o suficiente para provocar mudanças no comportamento da nuvem, tal como uma requisição de criação de instância. A agregação de tráfego com uma

granularidade mais grossa perde tais detalhes, que devem ser levados em consideração no monitoramento, de acordo com o requisito RF4.

Os pacotes coletados são filtrados conforme a funcionalidade em execução no sistema de monitoramento daquele host, que nos três casos verificam a porta de destino (passo 2 na Figura 3.6). Na F1, qualquer pacote que tenha a porta de destino pertencente a uma das portas contida na Tabela 2.1, por padrão, será aceito. Em relação à F2, apenas pacotes destinados às portas na qual as APIs executam serão aceitos. Para a F3, os pacotes destinados à porta na qual o RabbitMQ executa serão aceitos. Após, cada funcionalidade realiza seu processamento utilizado informações extraídas dos pacotes filtrados, posteriormente gravando no banco de dados escolhido, que é acessado por todas as instâncias em execução para gravar as informações geradas (passo 3 na Figura 3.6). Em relação aos dados armazenados, cada funcionalidade terá sua própria estrutura de armazenamento no banco de dados, que no caso de banco de dados relacionais (*i.e.,* MySQL, PostgreSQL), é o equivalente a uma ou mais tabelas para cada funcionalidade. O restante desta seção detalhará o funcionamento das três funcionalidades de monitoramento, na qual F2 e F3 serão agrupadas na explicação por conta da sua semelhança no funcionamento.

3.4.1 F1: Classificação consumo de banda, rede de controle

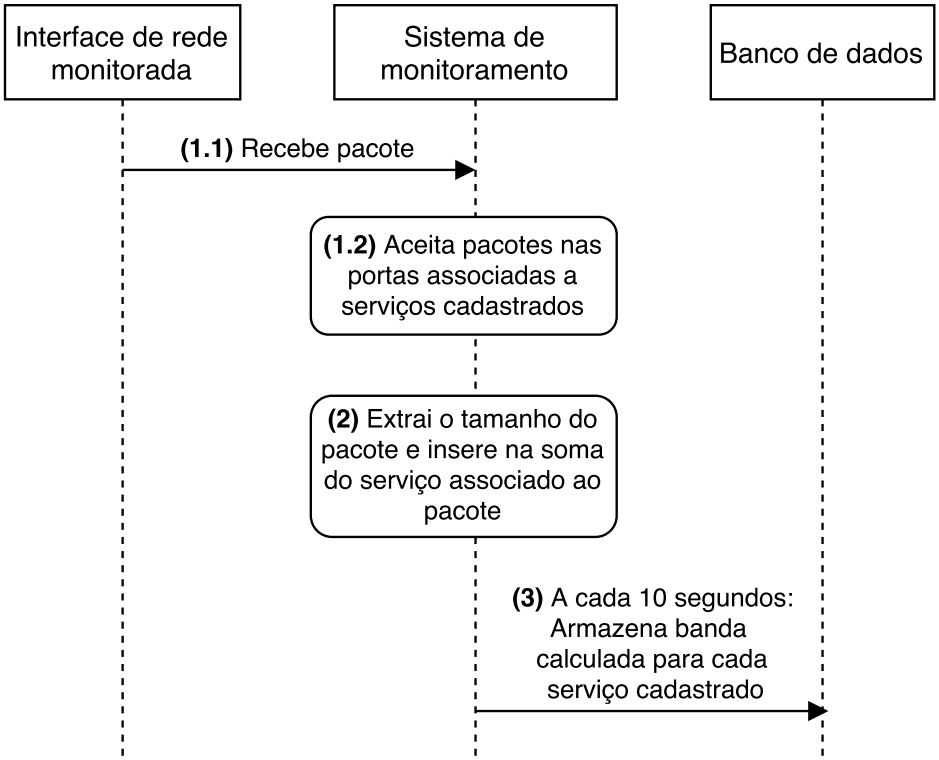
Esta funcionalidade tem como objetivo auxiliar na análise longitudinal de tráfego na rede de controle. Contudo, será considerado apenas o tráfego gerado pelos serviços delimitados na Tabela 2.1 para simplificar o escopo deste TCC. O diagrama de sequência na Figura 3.7 ilustra os passos de funcionamento desta funcionalidade do sistema de monitoramento.

Os passos que esta funcionalidade executa são:

1. Esta funcionalidade monitora apenas a interface da rede de controle, buscando todo tráfego que tenha como destino uma das portas associadas aos serviços em execução naquele host. As portas também são utilizadas para criar uma classificação de serviços, cuja porta de destino serve para decidir em qual serviço aquele pacote está associado;

2. Após definido o serviço destino daquele pacote, é extraído o tamanho daquele pacote, que é somado a um contador de tráfego para aquele serviço. Assim, por exemplo,

Figura 3.7: Diagrama de sequência: monitoramento de banda consumida pelos serviços



Fonte: O próprio autor.

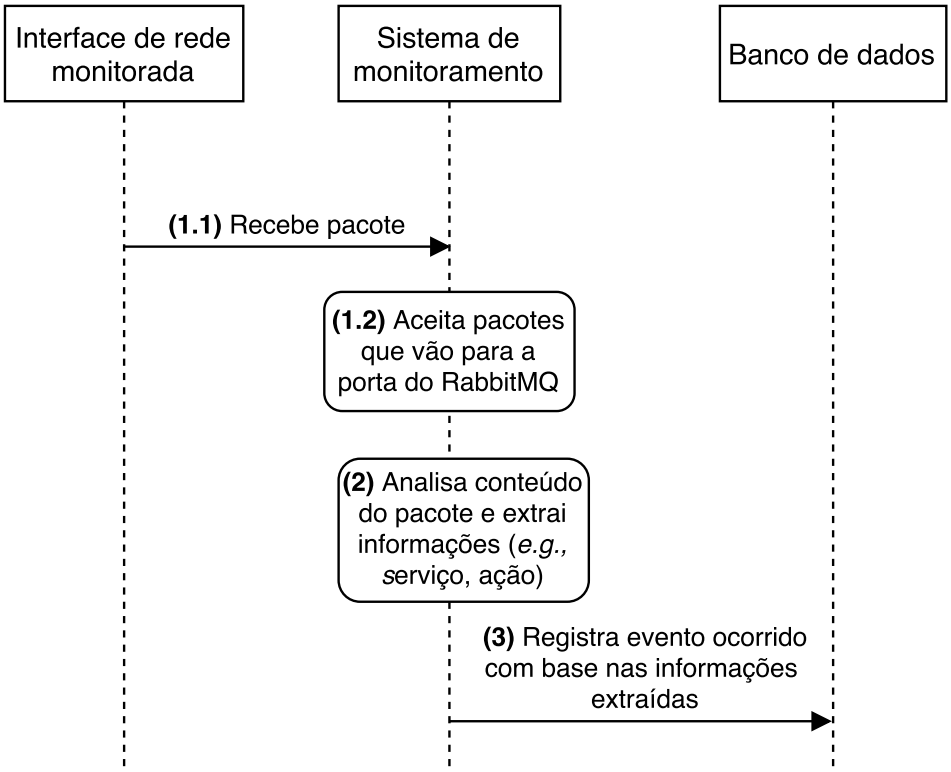
todos os pacotes recebidos por certo serviço ao longo de um tempo têm seu tamanho somado, e então esta soma é dividida pelo período de tempo estabelecido (*e.g.,* 5s, 10s). Ou seja, ao somar todo o tráfego recebido ao longo de um período, e depois dividir esta soma pelo período, é possível estabelecer uma aproximação da quantidade de banda utilizada (em KB/s) para o serviço em questão naquele período. Esta técnica é feita para todos os serviços considerados nesta análise, cada qual tendo seu próprio contador; e

3. Para cada uma destas somas é feita uma entrada no banco de dados, contendo o serviço e o cálculo de banda usada pelo serviço. No caso de oito serviços monitorados com um período de 10s, por exemplo, a cada 10s serão realizadas oito inserções, em que cada uma contém a quantidade de banda destinada ao seu serviço naqueles 10s. Após realizada a inserção o contador de cada serviço volta para zero, e faz o processo de inserção no banco de dados após 10s novamente.

3.4.2 F2 e F3: Cadastrar eventos detectados na API dos serviços e no *middleware* de comunicação

Ambas F2 e F3 comportam-se de maneira similar, variando apenas a porta de destino observada e o conteúdo/protocolo dos pacotes analisados. A Figura 3.8 ilustra um diagrama de sequência para F3, que monitora as transações internas da nuvem passando pelo *middleware* de comunicação. Para gerar informações significativas deve-se relacionar os dados gerados por estas funcionalidades com outras, sendo possível relacionar as duas, por exemplo, e criar uma sequência detalhada de ações na nuvem a partir de uma requisição, conforme feito por Sharma et al. (2015).

Figura 3.8: Diagrama de sequência: monitoramento de transações internas da nuvem



Fonte: O próprio autor.

As ações nestas funcionalidades ocorrem da seguinte forma, tomando o diagrama de sequência da F3 (Figura 3.8) como base:

1. A F2 monitora a interface da rede pública, enquanto a F3 monitora a interface da rede de controle. Ao receber pacotes, ambas as funcionalidades olham na porta de destino para decidir se aceitam os pacotes, na qual a F2 verifica se a porta de destino é de uma API que o host hospeda; e no caso da F3, é verificado se a porta de destino é a mesma que a do *middleware* de comunicação que o host hospeda; e

2. A princípio ambas as funcionalidades armazenarão as mesmas informações, mesmo que os protocolos de comunicação analisados sejam diferentes (HTTP na F2, e RPC na F3). Ou seja, para ambos as principais informações a armazenar são: serviço de origem, serviço de destino, IP do host e identificador da ação;

3. Por fim, armazena-se os dados de cada funcionalidade em suas respectivas tabelas.

Conforme definido, através da implementação deste sistema de monitoramento será possível gerar os dados necessários para realizar a caracterização de tráfego proposta para a rede de controle. A F1 é capaz de gerar informações relevantes em tempo real, possibilitando, por exemplo, verificar a variação do uso de banda pelos serviços ao longo de um período determinado. Diferentemente, as funcionalidades F2 e F3 geram dados que necessitam de uma análise mais minuciosa, realizada na etapa posterior à coleta, especificada na Seção 3.5.

3.5 Análise de tráfego

A etapa de análise de tráfego, que será aplicada nos dados gerados pelo sistema de monitoramento (definido na Seção 3.4) busca gerar informações que ajudem a compreender melhor o comportamento do tráfego analisado. Esta análise terá escopo limitado, no sentido de considerar apenas os serviços contidos na implementação mais popular de nuvem OpenStack, segundo The OpenStack project (2017g): Nova, Neutron, Cinder, Swift, Keystone e Glance. A proposta atual pretende caracterizar três ângulos diferentes na rede de controle:

- **Caracterização de funcionalidades da nuvem:** Busca entender como a execução de tarefas pelo consumidor influencia no funcionamento da rede de Controle da nuvem. A proposta inicial tem foco no ciclo de vida de VMs do OpenStack (*e.g.,* criação de VM, iniciação de VM, pausar VM em execução).

- **Consumo de banda por serviços:** Esta análise longitudinal tem como objetivo entender quais dos serviços considerados na análise mais geram tráfego na rede de Controle.

- **Influência de eventos periódicos no comportamento da nuvem:** Listar alguns dos eventos periódicos gerados pelos serviços da nuvem, e verificar qual o

impacto gerado por eles na rede de controle.

A **caracterização de funcionalidades da nuvem** utilizará informações geradas pelas F2 e F3, definidas no sistema de monitoramento. Estas funcionalidades geram dados referentes à comunicação com as APIs da nuvem, e da comunicação interna dos serviços, feita através do RabbitMQ. Sendo assim, é possível criar uma sequência que mostra a trajetória do tráfego durante a realização de uma tarefa na nuvem a partir de uma requisição recebida pela API. Ou seja, a partir de uma requisição de um consumidor (*e.g.,* criação de VM), cria-se um grafo direcionado que mostra quais mensagens foram enviadas para quais serviços à fim de concretizar a tarefa em questão. Segundo (SHARMA et al., 2015) é possível gerar esta sequência de eventos, mas ainda não foi realizada uma análise profunda para verificar se realmente existe alguma variável que pode ser utilizada para interligar estas mensagens diretamente.

A análise de **consumo de banda por serviços** baseia-se na F1 do sistema de monitoramento, que contabiliza o tráfego gerado por cada um dos serviços considerados na análise. A F1 do sistema de monitoramento deve gerar dados que podem ser interpretados diretamente, possibilitando a realização desta análise em tempo real. Sendo assim, é possível por exemplo, acessar os dados gerados por esta funcionalidade definindo o período de início e de fim da análise, na qual gera um gráfico mostrando qual a porcentagem de tráfego destinado a cada serviço.

Por fim, a análise da **influência de eventos periódicos no comportamento da nuvem** deve se basear principalmente nos dados gerados pela F1 do sistema de monitoramento, na qual buscará grandes variações no recebimento de tráfego dos serviços em curtos períodos de tempo. Baseando-se nestas variações serão consultados os dados gerados pelas F2 e F3 naquele período, que constará quais serviços enviaram as mensagens em questão. Ou seja, como produto final este ângulo de caracterização busca mostrar quem executou aquele evento periódico, qual a tarefa em questão, e se há impacto significativo. Definida a abordagem utilizada para a realização da caracterização de tráfego, o passo final é explicar como será realizado o experimento que aplicará a proposta definida neste capítulo.

3.6 Plano de testes

O plano de testes visa definir os experimentos a serem executados, os quais aplicarão a proposta de caracterização de tráfego. A finalidade destes experimentos é validar o sistema de monitoramento desenvolvido e a abordagem para análise do tráfego. No total serão realizados três experimentos, divididos em dois cenários diferentes, sendo que dois experimentos serão realizados em um dos cenários, e o outro cenário será usado no experimento restante.

- **Cenário 1:** Uma nuvem OpenStack com ambiente controlado, na qual toda interação com ela será originária dos experimentos executados; e
- **Cenário 2:** Uma nuvem OpenStack em ambiente de produção, com consumidores utilizando-a.

Os experimentos visam verificar o funcionamento do sistema de monitoramento, e então analisar os dados gerados após a execução. Estes experimentos são:

- **Experimento 1:** Verificar o funcionamento do sistema de monitoramento, com o objetivo de avaliar a quantidade de dados gerados no monitoramento do **Cenário 1**. O sistema de monitoramento utilizará todos as suas funcionalidades, na qual instâncias executarão em todos os hosts de interesse por um período de sete dias. Durante este período, a nuvem deverá receber algumas requisições, que simularão atividades de consumidores, mas de forma esporádica. Deste modo, será avaliada a quantidade de dados gerados, com o objetivo de verificar, por exemplo, se o banco de dados utilizado é adequado no caso de monitorar uma nuvem em ambiente de produção.

- **Experimento 2:** Análise longitudinal do consumo de banda pelos serviços de uma nuvem em ambiente de produção (**Cenário 2**). A função que registra o consumo de banda pelos serviços do OpenStack executará por um período de 30 dias. Após, com os dados coletados será feita uma análise, com o objetivo de verificar a variação do consumo ao longo do período, identificando padrões de comportamento, por exemplo.

- **Experimento 3:** Caracterização do comportamento da rede de controle em uma nuvem em ambiente de produção (**Cenário 2**). O sistema de monitoramento utilizará todas as suas funcionalidades, na qual instâncias executarão em todos os hosts de interesse por um período de sete dias. Ao longo deste período, a nuvem em questão será usada pelos consumidores e o sistema de monitoramento irá gerar dados sobre o uso. Após, será feita uma caracterização de tráfego abordando dois ângulos de análise, definidos na Seção 3.5: influência de eventos periódicos no comportamento da nuvem, e caracterização de funcionalidades da nuvem.

Como comparativo, serão avaliados o tamanho do banco de dados nos experimentos 1 e 3, e verificar se há grande mudança na quantidade de dados gerados a partir do monitoramento. Ambos os cenários utilizarão uma instalação similar do OpenStack, na qual os serviços caracterizados serão os mesmos.

3.7 Considerações do Capítulo

Este capítulo apresentou a proposta de caracterização de tráfego para uma nuvem computacional OpenStack. A proposta baseia-se na análise da arquitetura de funcionamento de alguns serviços do OpenStack, o que possibilitou a definição de estratégias para utilizar no sistema de monitoramento, responsável pela primeira etapa: a medição de tráfego. Foram definidos requisitos para a criação deste sistema de monitoramento, que abordam suas funcionalidades, e também pré requisitos, que devem ser considerados na construção dele. O sistema de monitoramento em questão terá três funcionalidades que irão medir o tráfego e popular um banco de dados com as informações geradas a partir do tráfego avaliado. Então, a etapa posterior, de análise de tráfego utilizará estas informações para melhor entender o comportamento da rede de controle sobre três ângulos de análise diferentes. Após definir o que deve ser feito, a especificação do plano de testes determinou quais os experimentos que serão realizados, e também estabeleceu os cenários empregados nos experimentos.

4 Considerações & Próximos passos

Este TCC tem como objetivo levantar questões relacionadas ao desempenho no tráfego da rede de controle em nuvens OpenStack. Softwares distribuídos como o OpenStack podem apresentar comportamento complexos em função da grande quantidade de componentes e suas respectivas interações. Sendo assim, para entender melhor o que ocorre na rede de controle, este trabalho propõe a realização de uma análise e caracterização do tráfego dela, que conta com um sistema de monitoramento para levantar as informações.

O levantamento bibliográfico é um dos passos iniciais para alcançar este objetivo. Nesta etapa apresentou-se sobre conceitos de computação em nuvem, os atores envolvidos em uma nuvem computacional e como tecnologias de virtualização afetam o funcionamento de recursos computacionais e de rede. Assim como, foi realizada uma introdução ao OpenStack, em que abordou-se sua arquitetura de rede e expôs sobre o Domínio de Controle, explicando sua importância neste trabalho. Também foi apresentado o conceito de caracterização de tráfego, definindo-se uma maneira para realizá-la. Sobre estes conceitos então foi feita uma descrição do problema abordado neste TCC, na qual aplicaram-se alguns dos conceitos descritos anteriormente.

Após, foram identificados e analisados trabalhos com escopos similares, cujos objetivos são relacionados com caracterização de tráfego em nuvens computacionais ou o monitoramento das mesmas. Com base nesta análise foram identificadas algumas ferramentas e abordagens disponíveis para a proposta deste TCC, tal como o trabalho de (SHARMA et al., 2015), que criou um sistema de detecção de falhas em nuvens OpenStack. A proposta deste trabalho iniciou com um detalhamento da arquitetura de alguns serviços do OpenStack, especificando os componentes destes serviços e como são distribuídos em uma instalação. Consequentemente, foi realizado um detalhamento dos serviços analisados, buscando características que contribuíssem na construção do sistema de monitoramento, que é a primeira parte da proposta. O sistema de monitoramento foi definido primeiramente em função dos seus requisitos, que foram definidos em função dos dados necessários para a caracterização e do ambiente em que o sistema executará. Assim, o sistema de monitoramento seguiu os requisitos e aplicou as características levantadas a partir do detalhamento dos serviços, que usou para definir como realizar o monitoramento,

e qual o tráfego de interesse à monitorar. Além destas informações apresenta-se também como ocorre a coleta e análise do tráfego pelo sistema.

Na segunda parte da proposta apresentou-se como serão analisadas as informações obtidas do sistema de monitoramento, e quais serão os ângulos de análise adotados. Por fim, definiu-se como os experimentos que aplicam esta proposta ocorrerão, exibindo os cenários que serão utilizados nos experimentos e também detalhando como os experimentos ocorrerão. Esta caracterização de tráfego terá o escopo limitado ao conjunto de serviços presente na instalação mais popular, segundo The OpenStack project (2017l): Cinder, Neutron, Nova, Swift, Keystone e Glance. Na parte de análise, esta caracterização está definida para três dos múltiplos ângulos de caracterização possíveis. Sendo assim, é possível que novos ângulos de caracterização surjam no decorrer do TCC-II ao analisar as informações coletadas.

Para o TCC-II, será implementado o sistema de monitoramento. Após implementado, este será instalado e executará numa nuvem OpenStack conforme o plano de testes. Por fim, os dados coletados pelo sistema de monitoramento durante os testes serão analisados, a fim de gerar uma caracterização da rede de controle que levantará questões de desempenho presentes nela.

4.1 Cronograma

Até o presente momento, o cronograma proposto no Plano do TCC foi seguido conforme o previsto, na ordem e nos prazos estipulados. A etapas presentes nesta seção foram definidas no Plano de TCC para atingir os objetivos propostos.

4.2 Etapas realizadas

As etapas que já foram executadas:

1. **Formulação do plano de TCC**;
2. **Levantamento e fichamento das referências** – Pesquisa de fontes para o embasamento teórico do trabalho, com base nos objetivos específicos;

3. **Consolidação das referências** – Fundamentação necessária para compreender o objeto de trabalho e atingir o objetivo do TCC-I;

4. **Identificação e análise de métodos para caracterização de tráfego** – Identificar e analisar um método de caracterização de tráfego aplicável em nuvens computacionais;

5. **Identificação e análise de abordagens para sistemas de monitoramento de tráfego** – Levantamento de abordagens para sistemas de monitoramento de tráfego, analisando sua aplicabilidade no contexto de nuvens computacionais baseadas em OpenStack;

6. **Especificação de uma ferramenta para monitoramento do tráfego** – Especificar um sistema de monitoramento para rede de controle em nuvens computacionais OpenStack, que auxiliará na caracterização da rede de controle. Serão consideradas características específicas do OpenStack nesta etapa, como protocolos, arquitetura de implementação e funcionalidades deste sistema. Especificação do plano de testes.

7. **Escrita do TCC-I**.

4.3 Etapas a realizar

As seguintes etapas serão realizadas no TCC-II, sendo voltadas para a implementação do sistema de monitoramento, sua aplicação na nuvem escolhida e a análise dos dados obtidos.

8. **Implementar o sistema de monitoramento** – Implementar, segundo a **Especificação do sistema para monitoramento de tráfego**, o software que realizará o monitoramento da rede de controle da nuvem computacional;

9. **Experimentação e coleta de dados** – Será colocado em operação o sistema de monitoramento implementado, para testá-lo e coletar tráfego a fim de caracterizar a rede em questão;

10. **Caracterização e análise da rede monitorada** – Caracterizar e analisar o comportamento e funcionalidades presentes no tráfego da rede coletado através do sistema de monitoramento. Verificando então, a relevância e impacto do tráfego e funcionalidades do sistema no desempenho da rede.

11. **Escrita do TCC-II**.

4.4 Execução do cronograma

A Tabela 4.1 apresenta o cronograma definido no início do TCC-I. As etapas já concluídas são marcadas com X. As etapas em preto sem nenhuma marcação são as etapas a serem executadas.

Tabela 4.1: Cronograma das atividades para TCC-I e TCC-II

Etapas	**2017**					**2018**						
	A	S	O	N	D	J	F	M	A	M	J	J
1	X											
2	X	X										
3	X	X										
4		X	X									
5			X	X								
6				X								
7		X	X	X								
8					■	■	■	■				
9							■	■				
10								■	■	■		
11									■	■	■	

Fonte: O próprio autor.

Referências

ARLITT, M. F.; WILLIAMSON, C. L. Web server workload characterization: The search for invariants. *ACM SIGMETRICS Performance Evaluation Review*, ACM, v. 24, n. 1, p. 126–137, 1996.

BOHN, R. B. et al. Nist cloud computing reference architecture. In: *2011 IEEE World Congress on Services*. USA: CreateSpace Independent Publishing Platform, 2011. p. 594–596. ISSN 2378-3818.

BRAUN, H.-W.; CLAFFY, K. C. Web traffic characterization: an assessment of the impact of caching documents from ncsa's web server. *Computer Networks and ISDN systems*, Elsevier, v. 28, n. 1, p. 37–51, 1995.

CALLADO, A. et al. A survey on internet traffic identification. *IEEE Communications Surveys Tutorials*, IEEE, , v. 11, n. 3, p. 37–52, rd 2009. ISSN 1553-877X.

CALLEGATI, F. et al. Performance of network virtualization in cloud computing infrastructures: The openstack case. In: *2014 IEEE 3rd International Conference on Cloud Networking (CloudNet)*. Luxembourg, Luxembourg: IEEE, 2014. p. 132–137.

CORRADI, A.; FANELLI, M.; FOSCHINI, L. Vm consolidation: A real case based on openstack cloud. *Future Gener. Comput. Syst.*, Elsevier Science Publishers B. V., Amsterdam, The Netherlands, The Netherlands, v. 32, p. 118–127, mar. 2014. ISSN 0167-739X. Disponível em: <https://doi.org/10.1016/j.future.2012.05.012>.

DAINOTTI, A.; PESCAPE, A.; VENTRE, G. A packet-level characterization of network traffic. In: *2006 11th International Workshop on Computer-Aided Modeling, Analysis and Design of Communication Links and Networks*. Trento, Italy: IEEE, 2006. p. 38–45. ISSN 2378-4865.

DENTON, J. *Learning OpenStack Networking (Neutron) - Second Edition*. 2 edition. ed. Birmingham Mumbai: Packt Publishing - ebooks Account, 2016. ISBN 978-1-78528-772-5.

DHARMAPURIKAR, S. et al. Deep packet inspection using parallel bloom filters. *IEEE Micro*, v. 24, n. 1, p. 52–61, Jan 2004. ISSN 0272-1732.

GILL, P. et al. Youtube traffic characterization: A view from the edge. In: *Proceedings of the 7th ACM SIGCOMM Conference on Internet Measurement*. New York, NY, USA: ACM, 2007. (IMC '07), p. 15–28. ISBN 978-1-59593-908-1. Disponível em: <http://doi.acm.org/10.1145/1298306.1298310>.

GROSSGLAUSER, M.; REXFORD, J. Passive traffic measurement for ip operations. In: *IN THE INTERNET AS A LARGE-SCALE COMPLEX SYSTEM*. England: Oxford University Press, 2002. p. 2.

JADEJA, Y.; MODI, K. Cloud computing - concepts, architecture and challenges. In: *2012 International Conference on Computing, Electronics and Electrical Technologies (ICCEET)*. Kumaracoil, India: IEEE, 2012. p. 877–880.

JAIN, R.; CHIU, D. M.; WR, H. A quantitative measure of fairness and discrimination for resource allocation in shared computer systems. ACM, Hudson, MA, cs.NI/9809099, September 1984.

JAIN, R.; PAUL, S. Network virtualization and software defined networking for cloud computing: a survey. *IEEE Communications Magazine*, v. 51, n. 11, p. 24–31, November 2013. ISSN 0163-6804.

JR., E. G. A.; TURKETT, W. H.; FULP, E. W. Using network motifs to identify application protocols. In: *GLOBECOM 2009 - 2009 IEEE Global Telecommunications Conference*. Honolulu, HI, USA: IEEE, 2009. p. 1–7. ISSN 1930-529X.

KARAGIANNIS, T. et al. Transport layer identification of p2p traffic. In: *Proceedings of the 4th ACM SIGCOMM Conference on Internet Measurement*. New York, NY, USA: ACM, 2004. (IMC '04), p. 121–134. ISBN 1-58113-821-0. Disponível em: <http://doi.acm.org/10.1145/1028788.1028804>.

KARAGIANNIS, T.; PAPAGIANNAKI, K.; FALOUTSOS, M. Blinc: Multilevel traffic classification in the dark. *SIGCOMM Comput. Commun. Rev.*, ACM, New York, NY, USA, v. 35, n. 4, p. 229–240, ago. 2005. ISSN 0146-4833. Disponível em: <http://doi.acm.org/10.1145/1090191.1080119>.

KHALIFE, J.; HAJJAR, A.; DIAZ-VERDEJO, J. A multilevel taxonomy and requirements for an optimal traffic-classification model. *International Journal of Network Management*, v. 24, n. 2, p. 101–120, 2014. ISSN 1099-1190. NEM-13-0061.R1. Disponível em: <http://dx.doi.org/10.1002/nem.1855>.

KRUTZ, R. L.; VINES, R. D. *Cloud Security: A Comprehensive Guide to Secure Cloud Computing*. 10475 Crosspoint Boulevard, Indianapolis: Wiley Publishing, 2010. ISBN 0470589876, 9780470589878.

KUMAR, R. et al. Open source solution for cloud computing platform using openstack. *International Journal of Computer Science and Mobile Computing*, v. 3, n. 5, p. 89–98, 2014.

LITVINSKI, O.; GHERBI, A. Openstack scheduler evaluation using design of experiment approach. In: *16th IEEE International Symposium on Object/component/service-oriented Real-time distributed Computing (ISORC 2013)*. Paderborn, Germany: IEEE, 2013. p. 1–7. ISSN 1555-0885.

MELL, P. M.; GRANCE, T. *SP 800-145. The NIST Definition of Cloud Computing*. Gaithersburg, MD, United States, 2011.

NGUYEN, T. T. T.; ARMITAGE, G. A survey of techniques for internet traffic classification using machine learning. *IEEE Communications Surveys Tutorials*, v. 10, n. 4, p. 56–76, Fourth 2008. ISSN 1553-877X.

NICODEMUS, C. H. Z.; BOERES, C.; REBELLO, V. E. F. Impacto do uso de contêineres em um serviço de nuvem HTC. In: *XVII Simpósio em Sistemas Computacionais de Alto Desempenho*. Aracajú, Seará, Brazil: SBC, 2016. p. 146–157.

PARK, B.; HONG, J. W. K.; WON, Y. J. Toward fine-grained traffic classification. *IEEE Communications Magazine*, IEEE, , v. 49, n. 7, p. 104–111, July 2011. ISSN 0163-6804.

PAXSON, V. Bro: a system for detecting network intruders in real-time. *Computer Networks*, v. 31, n. 23, p. 2435 – 2463, 1999. ISSN 1389-1286. Disponível em: <http://www.sciencedirect.com/science/article/pii/S1389128699001127>.

Red Hat. *OpenStack 11 Architecture Guide: Components*. 2017. Disponível em: <https://access.redhat.com/documentation/en-us/red_hat_openstack_platform/11/html-/architecture_guide/components>.

SCIAMMARELLA, T. et al. Analysis of control traffic in a geo-distributed collaborative cloud. In: *2016 5th IEEE International Conference on Cloud Networking (Cloudnet)*. IEEE: Pisa, Italy, 2016. p. 224–229.

sFlow Project. *Traffic Monitoring using sFlow*. , 2003. Disponível em: <http://www-.sflow.org/sFlowOverview.pdf>.

SHARMA, D. et al. Hansel: Diagnosing faults in openstack. In: *Proceedings of the 11th ACM Conference on Emerging Networking Experiments and Technologies*. New York, NY, USA: ACM, 2015. (CoNEXT '15), p. 23:1–23:13. ISBN 978-1-4503-3412-9. Disponível em: <http://doi.acm.org/10.1145/2716281.2836108>.

SOMMERS, J. et al. Improving accuracy in end-to-end packet loss measurement. *SIGCOMM Comput. Commun. Rev.*, ACM, New York, NY, USA, v. 35, n. 4, p. 157–168, ago. 2005. ISSN 0146-4833. Disponível em: <http://doi.acm.org/10.1145/1090191-.1080111>.

TANENBAUM, A. S.; WETHERALL, D. J. *Computer Networks*. 5th. ed. Upper Saddle River, NJ, USA: Prentice Hall Press, 2010. ISBN 0132126958, 9780132126953.

The OpenStack project. *Compute (Nova) API*. 2017. Disponível em: <https://developer-.openstack.org/api-ref/compute/>.

The OpenStack project. *OpenStack Cinder Documentation: AMQP*. 2017. Disponível em: <https://docs.openstack.org/cinder/latest/contributor/rpc.html>.

The OpenStack project. *OpenStack Docs: Networking architecture*. 2017. Disponível em: <https://docs.openstack.org/security-guide/networking/architecture.html>.

The OpenStack project. *OpenStack Glance Installation*. 2017. Disponível em: <https://docs.openstack.org/glance/pike/install/index.html>.

The OpenStack project. *OpenStack Neutron Installation Guide*. 2017. Disponível em: <https://docs.openstack.org/neutron/pike/install/index.html>.

The OpenStack project. *OpenStack Newton API Documentation*. 2017. Disponível em: <https://developer.openstack.org/api-guide/quick-start/>.

The OpenStack project. *OpenStack Newton: Installation overview*. 2017. Disponível em: <https://docs.openstack.org/newton>.

The OpenStack project. *OpenStack Nova Administrator Guide*. 2017. Disponível em: <https://docs.openstack.org/nova/pike/admin/index.html>.

The OpenStack project. *OpenStack Nova Documentation: AMQP*. 2017. Disponível em: <https://docs.openstack.org/nova/latest/reference/rpc.html>.

The OpenStack project. *OpenStack Swift Administrator Guide*. 2017. Disponível em: <https://docs.openstack.org/swift/pike/admin/index.html>.

The OpenStack project. *OpenStack Swift Administrator Guide*. 2017. Disponível em: <https://docs.openstack.org/swift/pike/admin/index.html>.

The OpenStack project. *What Is: OpenStack*. 2017. Disponível em: <https://www-.openstack.org/software>.

VENZANO, D.; MICHIARDI, P. A measurement study of data-intensive network traffic patterns in a private cloud. In: *2013 IEEE/ACM 6th International Conference on Utility and Cloud Computing*. Dresden, Germany: IEEE, 2013. p. 476–481.

VILELA, G. S. *Caracterização de Tráfego Utilizando Classificação de Fluxos de Comunicação*. 2006.

WANG, G.; NG, T. S. E. The impact of virtualization on network performance of amazon ec2 data center. In: *Proceedings of the 29th Conference on Information Communications*. Piscataway, NJ, USA: IEEE Press, 2010. (INFOCOM'10), p. 1163–1171. ISBN 978-1-4244-5836-3. Disponível em: <http://dl.acm.org/citation.cfm?id=1833515.1833691>.

WILLIAMS, N.; ZANDER, S.; ARMITAGE, G. A preliminary performance comparison of five machine learning algorithms for practical ip traffic flow classification. *SIGCOMM Comput. Commun. Rev.*, ACM, New York, NY, USA, v. 36, n. 5, p. 5–16, out. 2006. ISSN 0146-4833. Disponível em: <http://doi.acm.org/10.1145/1163593.1163596>.

WILLIAMSON, C. Internet traffic measurement. *IEEE Internet Computing*, IEEE, v. 5, n. 6, p. 70–74, 2001.

XU, K.; ZHANG, Z.-L.; BHATTACHARYYA, S. Profiling internet backbone traffic: Behavior models and applications. *SIGCOMM Comput. Commun. Rev.*, ACM, New York, NY, USA, v. 35, n. 4, p. 169–180, ago. 2005. ISSN 0146-4833. Disponível em: <http://doi.acm.org/10.1145/1090191.1080112>.

ZHANG, H. et al. An analytics approach to traffic analysis in network virtualization. In: *10th International Conference on Network and Service Management (CNSM) and Workshop*. IEEE: Rio de Janeiro, Brazil, 2014. p. 316–319. ISSN 2165-9605.

ZHANG, J.; MOORE, A. Traffic trace artifacts due to monitoring via port mirroring. In: *2007 Workshop on End-to-End Monitoring Techniques and Services*. Munich, Germany: IEEE, 2007. p. 1–8.