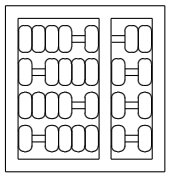

INSTITUTO DE COMPUTAÇÃO
UNIVERSIDADE ESTADUAL DE CAMPINAS

Gerência de Redes Definidas por Software em Nuvens Computacionais

Ulisses Malta Santos *Edmundo Roberto Mauro Madeira*

Relatório Técnico - IC-PFG-16-05 - Projeto Final de Graduação

December - 2016 - Dezembro

Gerência de Redes Definidas por Software em Nuvens Computacionais

Ulisses Malta Santos * Edmundo Roberto Mauro Madeira *

Resumo

Cada vez mais dispositivos estão conectados a Internet, aumentando a complexidade envolvida no gerenciamento das redes de computadores. A Internet das Coisas (*Internet of Things - IoT*), como é chamada, exige que as redes, ao contrário dos modelos e topologias tradicionais, sejam dinâmicas e flexíveis. As redes definidas por software (*Software Defined Networks - SDN*) surgiram nesse cenário e representam um novo modelo para a configuração, controle e operação das redes de computadores. Além de facilitar a virtualização de servidores e a orquestração de nuvens computacionais, as redes definidas por software ainda proporcionam grande flexibilidade e escalabilidade, pois permitem que sua administração seja feita de forma programática e que políticas de qualidade de serviço sejam implementadas. Este projeto tem por objetivo estudar a gerência de redes definidas por software dentro do ambiente de uma nuvem computacional OpenStack, focando no desenvolvimento de uma ferramenta que facilite a configuração das redes definidas por software. Também foram analisadas políticas de qualidade de serviço nessas redes virtuais, as quais permitiram uma diminuição do nível de congestionamento da rede, além de permitir que fluxos prioritários de dados fossem estabelecidos conforme a demanda.

Computação na Nuvem; Redes Definidas por Software; OpenStack; OpenFlow; Qualidade de Serviço

1 Introdução

A arquitetura das redes de computadores tradicionais apresenta uma série de limitações, tais como alto nível de complexidade (grande número de protocolos e recursos), baixa escalabilidade e difícil implementação de políticas de segurança e qualidade de serviço. Além disso, os equipamentos atuais disponíveis no mercado são baseados em uma arquitetura composta por uma camada de software fechada e que é executada em um hardware proprietário, resultando em soluções de alto custo e dificultando a implementação de novas funcionalidades. As redes definidas por software (*Software Defined Networks - SDN*) surgiram dentro desse cenário e representam um novo paradigma para o modo como as redes são criadas, configuradas e operadas, além de complementar a abstração de nuvem computacional. Essas redes são realizadas através da virtualização de elementos de rede, como

*Instituto de Computação, Universidade Estadual de Campinas (UNICAMP), Campinas, SP.

roteadores e switches e, seguindo o protocolo OpenFlow, propõem a desagregação entre o plano de dados (responsável pela comutação e repasse dos pacotes na rede) e o plano de controle (responsável pelos protocolos e pelas tomadas de decisão que resultam na criação e atualização das tabelas de encaminhamento na rede). Esse novo paradigma é baseado em tecnologias abertas e possibilita maior interoperabilidade, proporcionando uma arquitetura flexível e programável onde tanto o tráfego de dados quanto os elementos da rede podem ser administrados de forma eficiente.

Com cada vez mais dispositivos conectados a Internet, as redes de computadores passam a desempenhar um papel crítico no funcionamento e na usabilidade de tais dispositivos. As redes definidas por software demonstram grande potencial para atender essa nova demanda na medida que abstraem parte da complexidade envolvida na administração das redes, permitindo que o controle dos fluxos de dados seja realizado de forma programática a partir de um nó controlador. Essa nova abordagem permite que um modelo flexível de gerenciamento seja implantado, abrindo caminho para novas tecnologias e mecanismos de qualidade de serviço e políticas de segurança. A aplicação de políticas de segurança e de qualidade de serviço é importante para garantir que aplicações críticas tenham acesso aos recursos de que necessitam e também para permitir a implementação de diferentes modelos de negócios. A qualidade de serviço é uma tecnologia recente e que ainda está sendo estudada e desenvolvida, devendo assumir um papel de destaque num futuro próximo. Com essa tecnologia, seria possível estabelecer fluxos de dados prioritários, limitar a largura de banda consumida por um dispositivo ou aplicação específica e, com isso, facilitar a gerência das redes e ainda reduzir os custos associados.

2 Conceitos básicos

A seguir são apresentados alguns conceitos básicos e definições relevantes para o projeto.

2.1 Computação na nuvem - *Cloud Computing*

Computação na nuvem é um modelo que habilita o acesso conveniente e sob demanda via rede a uma porção de recursos computacionais configuráveis e compartilhados (redes, servidores, armazenamento, aplicações e serviços) e que podem ser rapidamente provisionados e liberados com um esforço mínimo de gerenciamento ou interação com o provedor de serviços. [7]

2.1.1 Modelos de serviço

Dentro do paradigma de computação na nuvem, podemos definir os seguintes modelos de serviço: software como um serviço (SaaS), plataforma como um serviço (PaaS) e infraestrutura como um serviço (IaaS). Pode-se visualizar essa divisão de forma esquemática na Figura 1.

- **Software como um serviço (SaaS):** Nesse modelo, o usuário é capaz de utilizar aplicações do provedor que estão rodando sobre a infraestrutura da nuvem. As

aplicações são acessíveis a partir de diversos dispositivos através de uma interface simples como a de um navegador ou ainda através de um programa específico. O usuário não gerencia ou controla a infraestrutura subjacente da nuvem, incluindo redes, servidores, sistemas operacionais, armazenamento ou até mesmo configurações individuais da aplicação, exceto no caso de algumas configurações limitadas e específicas para um único usuário.

- **Plataforma como um serviço (PaaS):** Esse modelo permite que o usuário implante aplicações criadas ou adquiridas por ele na infraestrutura da nuvem utilizando linguagens de programação, bibliotecas, serviços e ferramentas suportadas pelo provedor. O usuário não gerencia ou controla a infraestrutura subjacente da nuvem, incluindo redes, servidores, sistemas operacionais ou armazenamento, mas controla as aplicações que são implantadas e possivelmente as definições de configuração para o ambiente de hospedagem das aplicações.

- **Infraestrutura como um serviço (IaaS):** Nesse modelo, o usuário tem acesso à capacidade de processamento, armazenamento, redes e outros recursos computacionais onde ele é capaz de implantar e rodar um software arbitrário na nuvem, o que pode incluir sistemas operacionais e aplicações diversas. O usuário não gerencia ou controla a infraestrutura subjacente da nuvem, incluindo redes, servidores, sistemas operacionais, armazenamento e aplicações implantadas. No entanto é possível que ele tenha controle limitado sobre alguns componentes da rede.

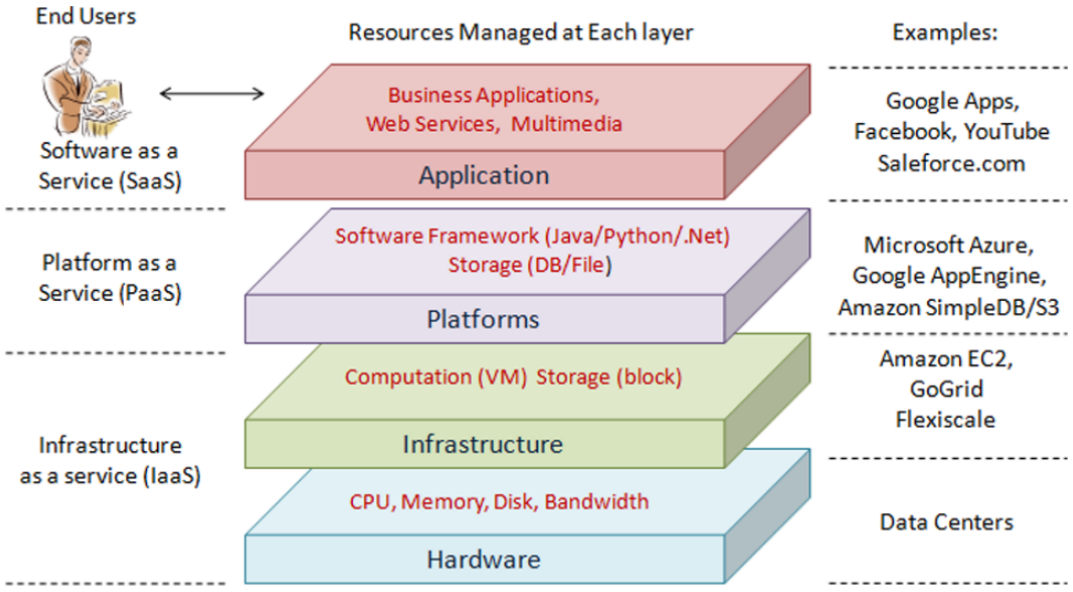

Figura 1: Arquitetura da nuvem computacional. [10]

2.1.2 Tipos de nuvem

- **Nuvem privada:** A infraestrutura da nuvem é provisionada para uso exclusivo de uma única organização composta de múltiplos consumidores (unidades de negócio). A nuvem pode ser de propriedade, gerenciada ou operada pela própria organização, por uma empresa terceirizada ou por uma combinação de ambas. Esse tipo de nuvem oferece alto grau de controle sobre o desempenho e a segurança.

- **Nuvem pública:** A infraestrutura da nuvem é provisionada para uso aberto do público em geral. Ela pode ser de propriedade, gerenciada ou operada por uma organização acadêmica, governamental, de negócios ou uma combinação dessas organizações. Esse tipo de nuvem não gera custos para o consumidor, mas dificulta o controle sobre os dados e as configurações de segurança.

- **Nuvem híbrida:** A infraestrutura da nuvem é composta por dois ou mais tipos distintos de nuvem (privada, pública, etc) que permanecem entidades únicas, mas são unidas por uma tecnologia padrão ou proprietária que habilita a portabilidade de dados e aplicações. Esse tipo de nuvem oferece maior flexibilidade do que os ambientes isolados.

2.2 OpenFlow

As redes se tornaram parte da infraestrutura crítica de vários negócios. No entanto, qualquer inovação nessa área enfrenta grande resistência e muitas vezes não é implementada devido ao enorme número de equipamentos e protocolos instalados. Isso criou uma barreira para que novas tecnologias fossem implantadas e resultou na sensação de que a infraestrutura da rede foi "engessada". Redes virtuais programáveis podem reduzir a barreira a entrada de novas ideias, aumentando a taxa de inovação na infraestrutura da rede. A ideia é simples: explorar o fato de que a maioria dos switches e roteadores ethernet modernos contém tabelas de fluxo. Embora as tabelas de fluxo de cada fabricante sejam diferentes, há um conjunto comum de funções que são executadas na maioria desses dispositivos. [6]

O OpenFlow, padronizado pela *Open Networking Foundation* (ONF), é um protocolo aberto para programar as tabelas de fluxo em diferentes switches e roteadores, explorando as funcionalidades comuns a todos os dispositivos. Um administrador da rede pode particionar o tráfego de dados em fluxos de produção e pesquisa, por exemplo. Desse modo, pesquisadores podem testar novos protocolos de roteamento, modelos de segurança, esquemas de endereçamento e até mesmo alternativas ao IP, enquanto que o tráfego de produção é isolado e processado da maneira convencional.

Esse é o protocolo mais utilizado atualmente na implementação de redes definidas por software, separando os planos de dados e de controle da rede. O protocolo OpenFlow descreve a interação entre um ou mais nós controladores com comutadores que seguem o mesmo protocolo. Um controlador OpenFlow instala entradas nas tabelas de fluxo dos comutadores, de modo que eles consigam realizar o encaminhamento do tráfego de dados na rede de acordo com essas entradas. Assim, os comutadores OpenFlow dependem da

configuração pelos controladores. O protocolo ainda coleta estatísticas sobre os fluxos de dados que trafegam na rede. [1]

O OpenFlow separa a programação de comutadores do hardware subjacente. Em uma rede convencional, cada equipamento comutador tem um software proprietário (dependendo do fabricante) para realizar suas funções de roteamento. Com o OpenFlow, as decisões de encaminhamento de cada pacote são centralizadas de tal forma que a rede pode ser programada independentemente dos equipamentos individuais que a compõem. Em um comutador convencional, o encaminhamento de pacotes (plano de dados) e o roteamento em alto nível (plano de controle) ocorrem no mesmo dispositivo. Por outro lado, um comutador OpenFlow separa essas funções de modo que o plano de dados reside no próprio dispositivo enquanto que um controlador a parte fica responsável pelas decisões de roteamento de alto nível. Isso permite um uso mais efetivo dos recursos da rede do que aquilo que seria possível com redes tradicionais.

O OpenFlow passou também a suportar políticas de qualidade de serviço a partir de sua versão 1.3. Na Figura 2, pode-se observar um esquema que ilustra o processo de encaminhamento de um pacote em um switch OpenFlow.

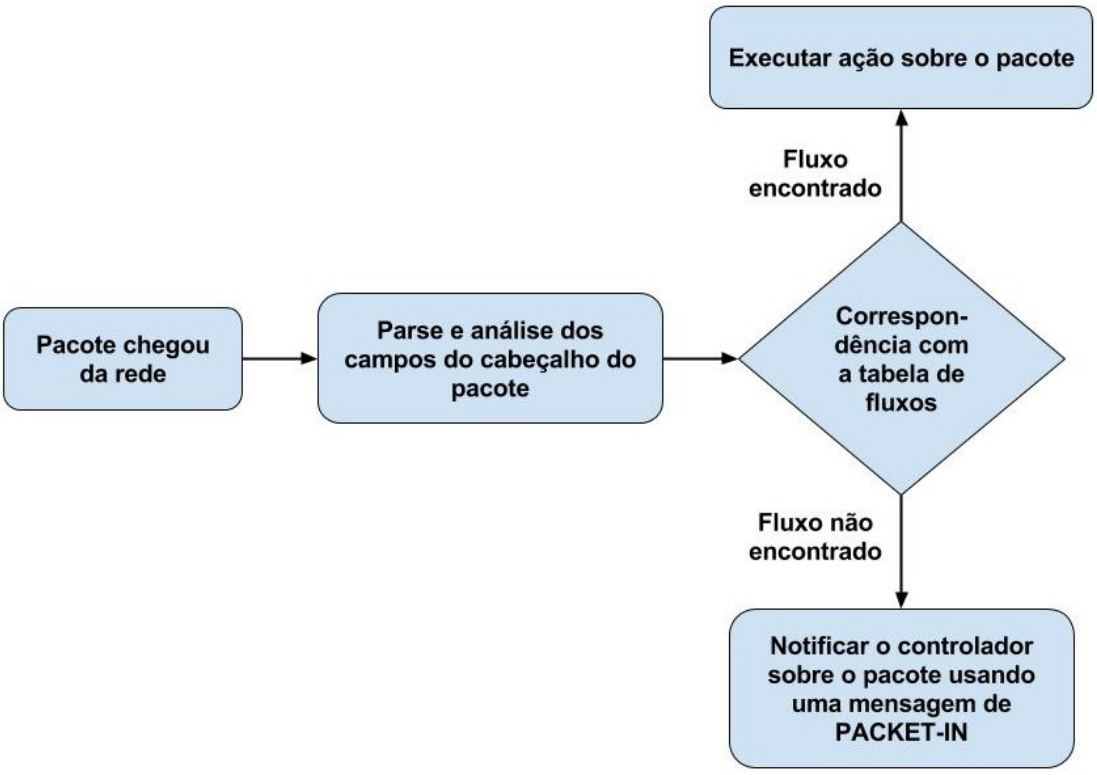

Figura 2: Esquema do encaminhamento de um pacote em um switch OpenFlow.

2.3 Redes definidas por software - *Software Defined Networks* (SDN)

A configuração e a instalação de uma rede requer pessoal altamente qualificado e experiente na configuração dos diversos elementos que a compõem, o que resulta em altos custos operacionais e ainda não garante uma interface para simulação e programação. Frente a esses problemas, um novo modelo é necessário. Esse novo tipo de rede é definido pela *Open Networking Foundation (ONF)* [2] da seguinte maneira: *"Na arquitetura das redes definidas por software, os planos de dados e de controle são desacoplados, inteligência e estado da rede são logicamente centralizados, e a infraestrutura subjacente da rede é abstraída das aplicações"* (ver Figura 3).

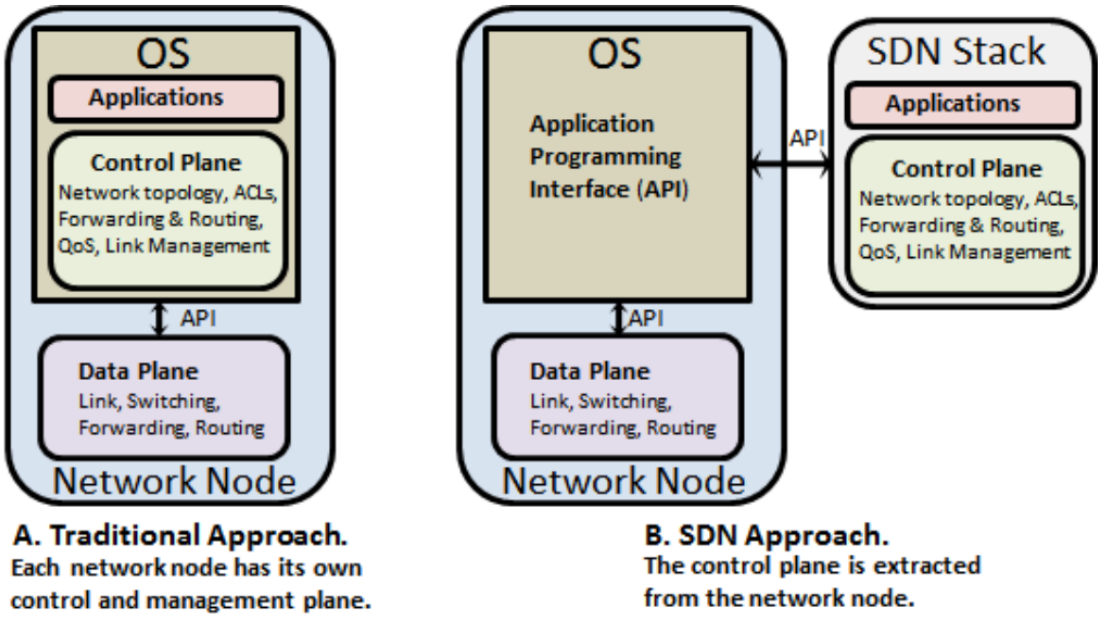

Figura 3: Visão tradicional de uma rede comparada à visão de uma rede definida por software. [9]

As redes definidas por software representam uma arquitetura emergente dinâmica e flexível (ver Figura 4). Essa arquitetura desacopla as funções de controle e encaminhamento da rede, permitindo que o plano de controle seja diretamente programável e que a infraestrutura subjacente da rede seja abstraída das aplicações. A base dessa tecnologia é a virtualização, a qual permite que um software seja executado separadamente do hardware subjacente. A virtualização tornou a computação na nuvem possível e agora permite que os *datacenters* provisionem recursos de forma dinâmica e precisa. Para acompanhar todo esse poder de processamento, as redes também devem se adaptar, tornando-se mais flexíveis e responsivas.

A virtualização, as nuvens e a mobilidade criaram ambientes mais complexos, exigindo que as redes se adaptem em termos de segurança, escalabilidade e modelo de gerenciamento.

As redes definidas por software introduzem uma camada de abstração que separa a inteligência da rede e sua configuração dos conectores físicos e do hardware. Dessa maneira, as redes definidas por software oferecem controle programático sobre os dispositivos físicos e virtuais da rede, podendo responder dinamicamente às mudanças que ocorrem na rede utilizando o procolo OpenFlow. Pode-se destacar as seguintes características das redes definidas por software:

- Plano de controle programável.
- Gerenciamento centralizado nos nós controladores.
- São baseadas em tecnologias de padrão aberto.
- São altamente configuráveis. Permitem uma configuração dinâmica de recursos.
- Permitem a interoperabilidade entre dispositivos de diferentes fabricantes.

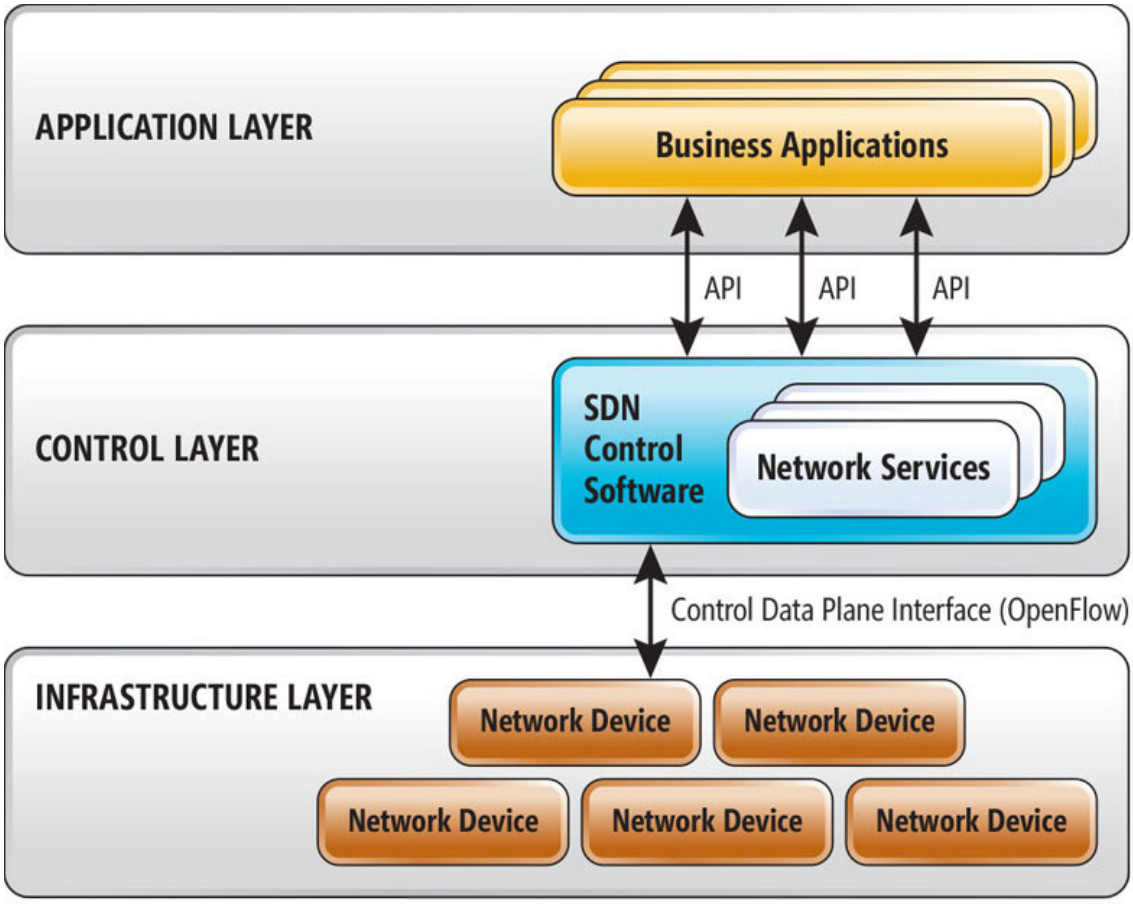

Figura 4: Arquitetura das redes definidas por software. (Fonte: *Open Networking Foundation*)

2.4 OpenStack

O OpenStack é um software livre (*open source*) composto por vários módulos voltado para a construção e o gerenciamento de plataformas de nuvens computacionais sejam elas públicas ou privadas, provendo infraestrutura como um serviço (*Infrastructure as a Service - IaaS*) para os seus usuários. O OpenStack permite que os usuários implantem máquinas virtuais e outras instâncias para tratar as diferentes tarefas envolvidas no gerenciamento de um ambiente de nuvem em tempo real, além de facilitar a alocação de recursos do *cluster* conforme as necessidades das aplicações. A seguir é apresentada uma visão geral dos módulos do OpenStack que foram utilizados no projeto. A Figura 5 ilustra o modelo de serviço provido pelo OpenStack.

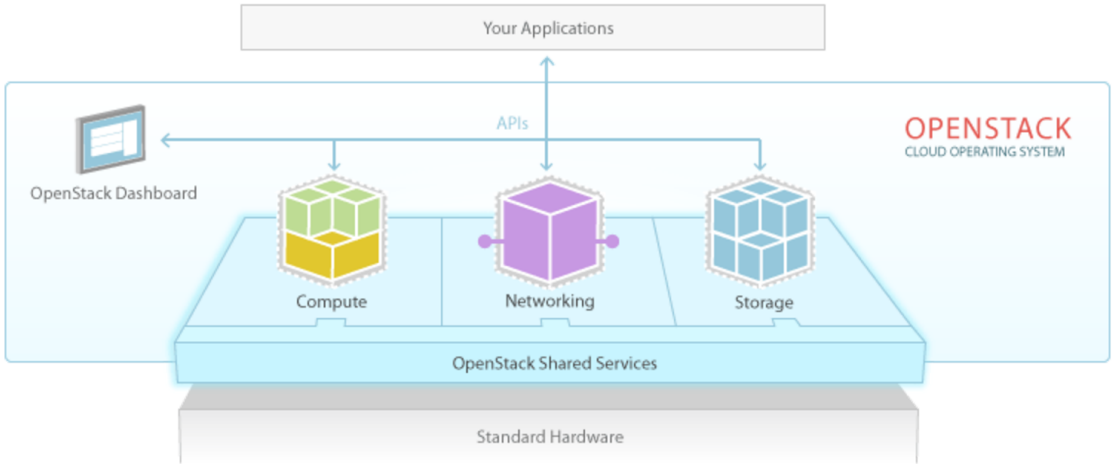

Figura 5: Modelo de serviço oferecido pelo OpenStack. (Fonte: `https://www.openstack.org/software/`)

2.4.1 Keystone

O Keystone é o módulo responsável pelos serviços de autenticação e identidade do OpenStack, registrando todos os projetos e usuários autorizados e agregando funções de autenticação de usuários, geração de *tokens* de autorização, criação e gerenciamento de políticas que afetam todos os usuários, além do gerenciamento do catálogo de *endpoints* de todos os serviços disponíveis (mantém um catálogo com todos os serviços disponíveis e seus respectivos endereços acessíveis pela rede tal como uma URL, permitindo que um usuário acesse um serviço). Todos os serviços do OpenStack fornecem *endpoints* para o Keystone através dos quais usuários podem requisitar recursos e realizar operações.

O objeto principal do sistema de gerenciamento de identidade é o usuário, que por sua vez é atribuído a um ou mais projetos (*tenants*), utilizados para isolar recursos dentro do OpenStack. Depois que o Keystone confirmou a identidade de um usuário, é gerado um *token* que confirma a sua identidade e pode ser utilizado em requisições futuras. Cada *token* é válido apenas por um período de tempo finito, podendo ser revogado caso seja necessário

remover o acesso de um usuário em particular. Políticas de segurança são estabelecidas com base em regras. Após a autenticação de um usuário, o próximo passo é determinar seu nível de autorização. O Keystone encapsula um conjunto de direitos e privilégios dentro da noção de papel ou função (*role*). Os *tokens* emitidos pelo serviço de autenticação incluem uma lista com os papeis que um usuário autenticado pode assumir. [4]

2.4.2 Nova

O Nova representa o núcleo de qualquer carga de trabalho ou computação no OpenStack. Há dois módulos dentro do OpenStack relacionados a computação: o Glance e o Nova. Enquanto o Glance gerencia todas as imagens estáticas de disco que contém código executável e um ambiente operacional, o Nova fica responsável pelas instâncias que estão sendo executadas, agindo como núcleo do serviço de infraestrutura. O Nova é o módulo mais complexo do OpenStack, principalmente porque é altamente distribuído e engloba uma grande quantidade de processos. Esse módulo interage com vários outros serviços do OpenStack: utiliza o Keystone para realizar autenticação, o Horizon como interface administrativa e o Glance para obter imagens de disco.

O Nova entra em ação para que uma instância ou máquina virtual seja lançada, tipicamente a partir de uma imagem fornecida pelo Glance. Embora não incorpore nenhum tipo de software de virtualização, o Nova pode ser integrado com várias plataformas que permitem a virtualização (*hypervisors*) através de *drivers* que realizam a interface com as tecnologias de virtualização.

De uma forma prática, lançar uma instância envolve identificar e especificar um modelo de hardware virtual (chamados de *flavors* no OpenStack). Esses modelos descrevem o número de CPUs, quantidade de memória RAM e configuração de armazenamento (disco rígido) que serão atribuídos a uma máquina virtual. A instalação padrão do OpenStack fornece 5 *flavors* que são configuráveis pelos administradores. O Nova então escalona a instância requisitada atribuindo sua execução para um nó de computação específico. Cada nó deve reportar seu status e capacidades regularmente para o escalonador do Nova, o qual utiliza esses dados para otimizar a alocação de máquinas virtuais, atribuindo a instância em questão para o *host* que satisfizer os requisitos de recursos com o menor custo. Ao se lançar uma instância, também podem ser especificados uma rede virtual e um *host* (*availability zone*) específicos de acordo com as necessidades de cada usuário. Caso seja necessário, o Nova também pode prover outros serviços como, por exemplo, o serviço de redes (utilizado somente em versões mais antigas do OpenStack). [3]

2.4.3 Neutron

O Neutron é o módulo responsável pela conectividade dentro do OpenStack. Não seria possível desenvolver um sistema escalável e flexível sem um serviço específico de rede. O Neutron é uma camada de abstração que pode acomodar uma grande variedade de plug-ins que tratam da integração com outros serviços de rede. Ele fornece uma API (*Application Programming Interface*) para projetos na nuvem com a qual usuários podem configurar políticas flexíveis e construir topologias sofisticadas de redes. Antigamente, os componentes

de rede do OpenStack ficavam no módulo Nova. A API de rede do OpenStack é baseada em um modelo simples de rede virtual, sub-rede e abstrações de portas para descrever recursos da rede.

Uma rede é um segmento isolado, análogo a uma LAN virtual (VLAN) nas redes físicas. Mais especificamente, a rede definida por software do OpenStack é um domínio reservado para o projeto a que está associada ou ainda pode ser compartilhada caso seja necessário. A rede também é o objeto primário da API do Neutron. Em outras palavras, portas e sub-redes são sempre atribuídas a uma rede específica. Uma sub-rede é um bloco de endereços IP (versão 4 ou 6) e suas configurações associadas, ou seja, é uma faixa de endereços a partir da qual o OpenStack pode obter endereços IP para atribuir às máquinas virtuais que forem lançadas. Junto com a sub-rede, também podem ser especificados um *gateway*, uma lista de servidores DNS e uma série de rotas que irão compor a rede. Máquinas virtuais nessa sub-rede automaticamente herdam essa configuração. A porta é um ponto de conexão virtual. Cada instância pode se conectar a uma rede virtual a partir de uma porta. No momento da criação, a porta recebe um endereço IP fixo de uma das sub-redes designadas. Um usuário pode especificar um endereço IP da faixa ou deixar que o Neutron aloque um dos endereços disponíveis. Após uma porta ser desalocada, qualquer endereço IP associado a ela volta a estar disponível. [5]

Versões mais recentes do Neutron passaram a incorporar mecanismos que permitem a implementação de políticas de qualidade de serviço nas redes definidas por software. Tais políticas são de grande utilidade para o gerenciamento adequado e flexível das redes virtuais. A princípio, somente políticas relacionadas a limitação da largura de banda foram implementadas pela API de qualidade de serviço do Neutron.

2.4.4 Ceilometer

O Ceilometer é o módulo responsável por coletar dados sobre a utilização dos recursos físicos e virtuais das nuvens OpenStack. Os dados coletados são armazenados e podem ser recuperados para análise. Além disso, esse módulo também permite que eventos sejam acionados sempre que certos critérios de utilização do *cluster* são atingidos. Dentre os dados monitorados, pode-se citar a quantidade de memória RAM utilizada por cada instância, utilização da CPU, entre outros. O Ceilometer pode trabalhar em conjunto com outros serviços do OpenStack para produzir um ambiente dinâmico e escalável.

2.4.5 Horizon

O Horizon implementa uma interface web baseada no framework Django para acesso aos diferentes serviços do OpenStack. A partir dessa interface é possível administrar diferentes funções da nuvem OpenStack, como lançar máquinas virtuais, criar redes definidas por software, etc. Esse módulo suporta uma série de plug-ins para agregar novas funcionalidades à sua interface administrativa.

2.5 Qualidade de Serviço - *Quality of Service* (QoS)

Conforme o número de usuários e dispositivos conectados a Internet continua a crescer, os requisitos de desempenho das redes de computadores devem crescer no mesmo ritmo para manter a usabilidade e atender a demanda. Muitos dos serviços online mais recentes necessitam de redes com grande desempenho e altas larguras de banda, sendo que o desempenho da rede é um motivo de preocupação tanto para o usuário final quanto para o provedor de serviços. O objetivo primário da qualidade de serviço (*Quality of Service - QoS*) é prover um modelo para a priorização dos diferentes fluxos de dados nas redes, incluindo controle da largura de banda dedicada a cada usuário e/ou aplicação, controle de jitter (variação da latência) e diminuição das perdas de pacotes.

A qualidade de serviço envolve controlar e gerenciar os recursos disponíveis na rede atribuindo prioridades para tipos específicos de dados (vídeo, áudio, arquivos, etc). A qualidade de serviço pode ser dirigida a uma interface da rede, a um servidor ou roteador, ou a aplicações específicas. Esse modelo é especialmente importante para aplicações em tempo real. Redes com políticas de qualidade irão priorizar certas aplicações, serviços e/ou usuários de modo que fluxos de dados prioritários possam consumir maior largura de banda.

Os diversos dispositivos conectados a rede competem por largura de banda. A tecnologia de qualidade de serviço pode prevenir uma distribuição desigual dos recursos disponíveis. A qualidade de serviço atribui uma prioridade para cada dispositivo e serviço operando na rede e controla a largura de banda que cada um pode consumir baseado em sua função ou tipo de tráfego. Uma transferência de arquivo, por exemplo, é um processo tolerante a falhas. O cliente e o servidor trocam dados para garantir que todos os bits foram entregues corretamente. Se algum bit foi perdido, ele será reenviado até que todo o pacote seja recebido. Isso não acontece com uma transmissão em tempo real, como uma chamada de vídeo ou voz ou ainda uma partida de um jogo online. O cliente não pode pedir ao servidor para reenviar os bits perdidos, pois qualquer interrupção na transmissão gera um atraso indesejado. Com a qualidade de serviço, transferências de arquivo podem demorar mais enquanto um vídeo é assistido ou um jogo online é executado, mas uma boa usabilidade será garantida. Há um *tradeoff* na rede, pois diminui-se o atraso de uma fonte específica de tráfego e em contrapartida aumenta-se o atraso de outra fonte. Qualidade de serviço envolve monitorar o tráfego de dados e comparar os níveis de atividade com valores pré-definidos (políticas). Essas políticas de tráfego de dados tipicamente resultam em perdas de pacote, já que mensagens são descartadas quando uma fonte excede os limites da política.

Diferentes abordagens podem ser seguidas para a implementação da qualidade de serviço. Pode-se simplesmente identificar o tráfego que se deseja gerenciar e então atribuir-lhe uma prioridade: alta, média ou baixa. Pode-se também escolher aplicações específicas, ou ainda identificar portas específicas que um serviço ou aplicação usa para se conectar a Internet. Outra possibilidade seria atribuir uma prioridade para um dispositivo específico usando seu endereço IP ou MAC. Configurar a qualidade de serviço pode ser bem complicado, requerendo conhecimentos específicos sobre protocolos de rede e sobre a operação de um roteador. Uma configuração incorreta pode levar a rede a ter um desempenho inferior ao que teria sem qualquer política de qualidade de serviço.

Administradores de rede podem garantir o tráfego de dados para aplicações críticas,

de modo que transações possam ser processadas em uma quantidade de tempo aceitável. A qualidade de serviço também pode ser utilizada para controlar o tráfego de pacotes UDP (*User Datagram Protocol*), visto que esse protocolo não implementa mecanismos para detectar e prevenir o congestionamento da rede. A qualidade de serviço proporciona os seguintes benefícios: [8]

- Permite que administradores controlem os recursos da rede e os administrem a partir de uma perspectiva voltada ao modelo de negócios ao invés de uma perspectiva técnica.
- Assegura que aplicações sensíveis ao tempo e aplicações críticas tenham acesso aos recursos que necessitam, enquanto permite que outras aplicações também tenham acesso a rede.
- Melhora a experiência do usuário.
- Reduz custos ao utilizar os recursos existentes de maneira mais eficiente. Com isso, atrasa ou reduz a necessidade de expansão ou atualização da rede.

3 Modelo

O OpenStack provê infraestrutura como um serviço, possibilitando a orquestração de nuvens computacionais dinâmicas e flexíveis. Nas Figuras 6 e 7, pode-se observar o modelo de configuração e controle proposto pelas redes definidas por software. Os planos de dados e controle são desacoplados e políticas de qualidade de serviço podem ser implantadas.

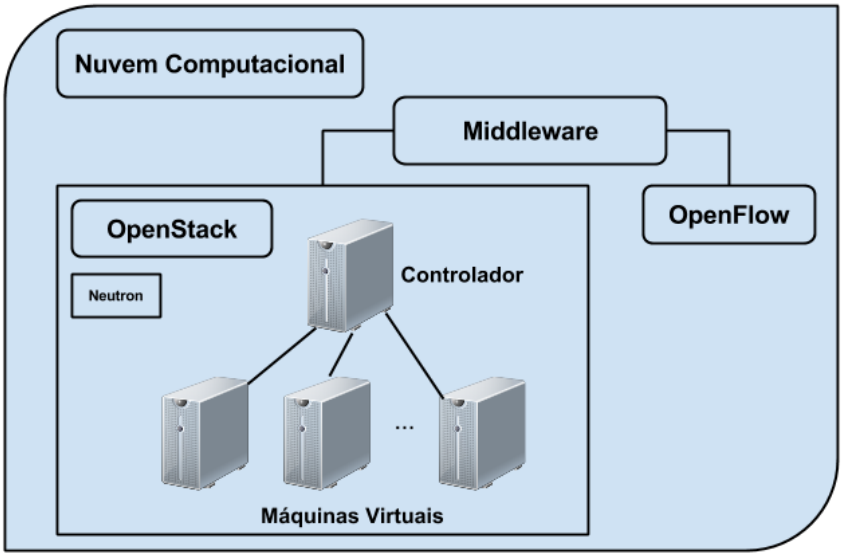

Figura 6: Topologia da nuvem OpenStack.

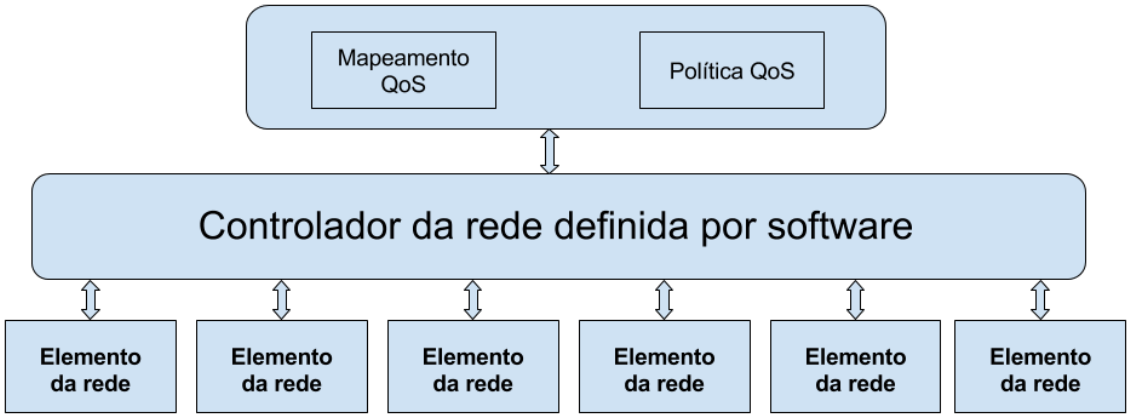

Figura 7: Qualidade de serviço na rede definida por software.

A Figura 6 ilustra a topologia desenvolvida na nuvem OpenStack utilizada no projeto. Essa nuvem possui um nó controlador e vários outros nós utilizados exclusivamente para computação. As funções relacionadas as redes definidas por software são orquestradas pelo Neutron, enquanto o núcleo da computação é desenvolvido pelo Nova. Várias máquinas virtuais podem ser instanciadas e conectadas a rede virtual, sendo que todo o processo de configuração e administração dos elementos virtuais da nuvem pode ser realizado a partir da ferramenta desenvolvida no projeto. A partir desse modelo de configuração, pode-se observar a separação entre os planos de dados e de controle proposta pelo protocolo OpenFlow. Já a Figura 7 ilustra a aplicação de políticas de qualidade de serviço aos elementos da rede definida por software, possibilitando que os administradores da rede tenham controle sobre os diferentes fluxos de dados. A partir desse modelo de controle programático, fica evidente os benefícios proporcionados pelo OpenFlow à gerência das redes de computadores.

4 Resultados

O projeto desenvolvido consistiu de três etapas principais: instalação e configuração adequada do ambiente de uma nuvem OpenStack, desenvolvimento de uma ferramenta Java para a configuração e gerência de redes definidas por software dentro da nuvem OpenStack e teste e análise de políticas de qualidade de serviço aplicadas às redes definidas por software a partir da API oferecida pelo Neutron.

4.1 Instalação da nuvem OpenStack

O OpenStack é um projeto complexo composto por diferentes módulos que juntos operam para habilitar toda a infraestrutura necessária para o funcionamento de uma nuvem computacional. Apesar de fornecer diversos serviços para facilitar a orquestração e operação da nuvem, o OpenStack ainda exige um processo trabalhoso e complicado de configuração para ser instalado. Cada módulo de interesse precisa ser configurado adequadamente por

um administrador durante o processo de instalação para que a nuvem opere de maneira aceitável, o que torna tal processo demorado e acaba exigindo certo grau de experiência dos administradores da nuvem. Isso acontece em parte pelo grande número de serviços e opções disponíveis para configuração.

Para facilitar a instalação desse ambiente, é disponibilizada uma versão voltada para desenvolvimento do OpenStack que visa facilitar o processo de instalação e conta com uma série de scripts e opções pré-definidas ao contrário da instalação tradicional. Essa versão de desenvolvimento, chamada de devstack, está disponível online no Github em `https://github.com/openstack-dev/devstack`, sendo que cada branch desse repositório corresponde a uma release diferente do OpenStack. Esse ambiente de desenvolvimento foi utilizado em parte do projeto e possibilitou que o OpenStack fosse instalado por completo em uma única máquina virtual, simulando o mesmo ambiente encontrado em um *cluster* real. A branch utilizada na instalação foi a branch estável com a release mais recente do OpenStack (`stable/mitaka`), de modo a aproveitar as funcionalidades mais recentes incorporadas a esse software. A qualidade de serviço, por exemplo, foi incorporada apenas nas versões mais recentes do OpenStack e ainda está sendo desenvolvida e estendida. Por isso, a API de qualidade de serviço disponível atualmente ainda é bem limitada e funciona somente quando o OpenStack é instalado com plug-ins bem específicos.

Para instalar o OpenStack a parir do devstack, é preciso apenas clonar o repositório e executar o script `stack.sh`, no entanto quando instalado dessa maneira o OpenStack é instalado apenas com os serviços essenciais habilitados e sem que serviços como a qualidade de serviço estejam disponíveis. Por isso, é interessante criar um arquivo de configuração local para customizar a instalação e habilitar todos os módulos necessários. Esse arquivo de configuração deve ser chamado `local.conf` e estar no mesmo diretório que o script `stack.sh`. O próprio devstack fornece um arquivo de configuração de exemplo dentro do diretório `samples` do repositório. Esse arquivo de exemplo é bem limitado e não habilita todos os serviços necessários para esse projeto, mas é um bom ponto de partida para entender como a instalação pode ser customizada. A seguir é apresentado o arquivo `local.conf` utilizado nesse projeto:

```
1  [[local|localrc]]
2
3  ADMIN_PASSWORD=admin
4  MYSQL_PASSWORD=$ADMIN_PASSWORD
5  RABBIT_PASSWORD=$ADMIN_PASSWORD
6  SERVICE_PASSWORD=$ADMIN_PASSWORD
7  SERVICE_TOKEN=$ADMIN_PASSWORD
8
9  RECLONE=yes
10
11 HORIZON_BRANCH=stable/mitaka
12 KEYSTONE_BRANCH=stable/mitaka
13 NOVA_BRANCH=stable/mitaka
14 NEUTRON_BRANCH=stable/mitaka
15 GLANCE_BRANCH=stable/mitaka
```

```
16 CINDER_BRANCH=stable/mitaka
17 HEAT_BRANCH=stable/mitaka
18
19 DEST=/opt/stack
20 LOGDIR=$DEST/logs
21 SCREEN_LOGDIR=$LOGDIR
22 LOGFILE=$LOGDIR/devstack.log
23 LOGDAYS=2
24 LOG_COLOR=True
25
26 ENABLED_SERVICES=key,rabbit,mysql,horizon
27 ENABLED_SERVICES+=,n-api,n-crt,n-cpu,n-cond,n-sch,n-novnc,n-cauth
28 ENABLED_SERVICES+=,g-api,g-reg
29 ENABLED_SERVICES+=,heat,h-api,h-api-cfn,h-api-cw,h-eng
30
31 # Disable nova networking
32 disable_service n-net
33 # Neutron
34 enable_service neutron q-svc q-agt
35 enable_service q-dhcp
36 enable_service q-l3
37 enable_service q-meta
38 enable_service q-fwaas
39 enable_service q-vpn
40 # Enable QoS
41 enable_service q-qos
42 enable_service q-flavors
43
44 # Necessary plug-in to enable the QoS API
45 Q_PLUGIN=ml2
46
47 enable_service ceilometer-acompute ceilometer-acentral ceilometer-collector
       ceilometer-api
48
49 enable_plugin neutron https://git.openstack.org/openstack/neutron stable/mitaka
```

Arquivo de configuração utilizado na instalação do OpenStack.

Esse arquivo de configuração habilita os serviços mais importantes do OpenStack, incluindo o Neutron, módulo mais importante para esse projeto. Para utilizar o Neutron, é preciso desabilitar a função de conectividade disponível no Nova (n-net). É preciso ainda deixar explícito que o serviço de qualidade de serviço deve ser habilitado e, para que ele funcione corretamente, é necessário utilizar o plug-in ml2 em conjunto com o Open vSwitch. Na release atual do OpenStack, a qualidade de serviço só funciona com esses plug-ins específicos. Feito o arquivo de configuração só é preciso executar o script `stack.sh` para realizar a instalação. O processo de instalação é demorado, pois uma séries de dependências e configurações devem ser resolvidas. Apesar de facilitar a instalação devido a automatização de grande parte do processo de configuração do OpenStack, o devstack pode apresentar

diversos problemas e erros durante a instalação dependendo do sistema onde a mesma é realizada. Durante a instalação do ambiente do projeto foram encontrados diversos erros que foram resolvidos para que a instalação fosse realizada.

Ao término da instalação, o devstack irá exibir um relatório com o tempo gasto durante todo o processo e uma mensagem semelhante àquela exibida na Figura 8.

Figura 8: Término da instalação do devstack.

Por padrão são criadas duas redes definidas por software, uma pública e uma privada que podem ser utilizadas para conectar diferentes máquinas virtuais. Obviamente, novas redes virtuais por ser criadas conforme a necessidade. O OpenStack pode ser administrado a partir da linha de comando do terminal ou a partir da interface web fornecida pelo módulo Horizon. Os usuários parão são admin e demo e a senha é aquela que foi definida no arquivo de configuração. Para utilizar o OpenStack via linha de comando é preciso realizar o processo de autenticação na nuvem, no entanto é cansativo e pouco prático ter que especificar o nome de usuário e senha a cada comando executado. Para facilitar esse processo, as informações pertinentes a autenticação podem ser especificadas indiretamente através de variáveis de ambiente pré-definidas, ou seja, se essas variáveis forem definidas, o OpenStack irá utilizá-las para a autenticação ao invés de exigir que o usuário as especifique a cada novo comando executado. Para definir essas variáveis, o devstack disponibiliza o script `openrc`. Para se autenticar com o usuário admin basta executar o comando `source openrc admin`, por exemplo. O OpenStack possui uma quantidade muito grande de serviços e comandos, sendo que cada comando ainda suporta um grande número de opções, o que acaba dificultado um pouco o processo de configuração das redes definidas por software. Por isso, é interessante o desenvolvimento de uma ferramenta que permita a automatização desse processo de configuração. O OpenStack é distribuído com uma imagem disco padrão (cirros) que serve apenas para uma checagem inicial do funcionamento da nuvem. Para testes e execução de aplicações é preciso adicionar outras imagens, as quais serão utilizadas para o lançamento de máquinas virtuais. No caso desse projeto, foi criada uma imagem do Ubuntu 14.04 que posteriormente foi utilizada durante os testes executados.

Além do ambiente fornecido pelo devstack, também foi possível realizar alguns testes em um *cluster* real instalado no Laboratório de redes de Computadores (LRC) do Instituto de Computação (IC).

4.2 Ferramenta desenvolvida

Foi desenvolvida uma ferramenta na linguagem Java com intuito de facilitar e agilizar o processo de criação e configuração de redes definidas por software dentro do ambiente de uma nuvem computacional OpenStack e, assim, possibilitar a gerência desse tipo de rede virtual.

Para o desenvolvimento dessa ferramenta, foi utilizada a biblioteca OpenStack4j (`http://www.openstack4j.com/`) que fornece o suporte necessário para acessar os serviços e recursos da nuvem OpenStack, provendo abstrações para acesso aos principais módulos da instalação. Essa biblioteca foi adotada para o projeto pois é uma ferramenta livre de código aberto e que dá suporte para as principais funcionalidades do OpenStack. Além disso, essa biblioteca também é desenvolvida especificamente para a nuvem OpenStack, utilizando abstrações e nomenclaturas que facilitam a sua compreensão e aplicação ao projeto. Outra biblioteca que poderia ser utilizada para o desenvolvimento da ferramenta é a biblioteca jclouds, que também é livre e muito utilizada para o desenvolvimento de aplicações na nuvem. No entanto, essa biblioteca não foi projetada especificamente para o OpenStack, mas sim para ser uma ferramenta de propósito geral e poder ser utilizada em conjunto com vários tipos de infraestrutura. Apesar de também ser bastante utilizada com o OpenStack, a biblioteca jclouds ainda não dá suporte a alguns módulos do OpenStack e é por isso que a biblioteca OpenStack4j foi escolhida para o projeto.

Outra ferramenta importante para o desenvolvimento do projeto foi o Maven. O Maven é uma ferramenta utilizada no gerenciamento de projetos feitos principalmente em Java baseada no conceito de um modelo de projeto objeto (*Project Object Model - POM*). O Maven pode gerenciar a compilação e a documentação de um projeto, tomando conta das dependências necessárias, isto é, baixa as bibliotecas Java e plug-ins dinamicamente de um ou mais repositórios de acordo com as necessidades de cada projeto. Isso é importante para o projeto desenvolvido, pois facilita o gerenciamento de bibliotecas uma vez que serão utilizadas várias bibliotecas para trabalhar com o OpenStack que não são padrão da linguagem Java e, então, caso essa ferramenta não fosse utilizada, seria necessário instalar essas bibliotecas em todas as máquinas utilizadas no processo de desenvolvimento. A utilização dessa ferramenta torna a compilação muito mais simples e prática. Os projetos Maven são configurados a partir de um arquivo `pom.xml`. No caso desse projeto, o Maven foi configurado para resolver as dependências relacionadas a biblioteca OpenStack4j. Novas dependências podem ser facilmente adicionadas ao arquivo de configuração.

O código desenvolvido foi dividido em algumas classes (conforme ilustrado no diagrama UML da Figura 9) para separar as funções implementadas de acordo com a sua funcionalidade e módulo do OpenStack com o qual interagem. A classe `Identity` é a classe principal do projeto sendo responsável pela autenticação na nuvem OpenStack (módulo Keystone), além de identificar os comandos especificados pelo usuário e chamar os métodos apropriados. A classe `Compute` está relacionada as funções de computação do OpenStack (módulo Nova), sendo responsável por criar e deletar máquinas virtuais e por gerenciar *flavors* (modelos de hardware utilizados no lançamento de uma instância) e *keypairs* (conjunto composto por um nome e uma chave pública e utilizado no processo de acesso a uma máquina virtual). A classe `BootMultiThread` é utilizada para o lançamento multithread de máquinas virtuais. A classe `Networking` fica responsável por implementar todas as funções relacionadas as redes virtuais do projeto (módulo Neutron). Essa classe cria e deleta redes virtuais, sub-redes, roteadores virtuais, entre outros. Essa classe também gerencia políticas de qualidade de serviço associadas às redes definidas por software. A classe `Telemetry` é responsável por coletar estatísticas a respeito da utilização dos recursos do *cluster* (módulo Ceilometer). Por fim, a última classe implementada foi a classe `Util` que implementa algumas funções

de uso geral não relacionadas ao OpenStack e que podem ser utilizadas pelas outras classes.

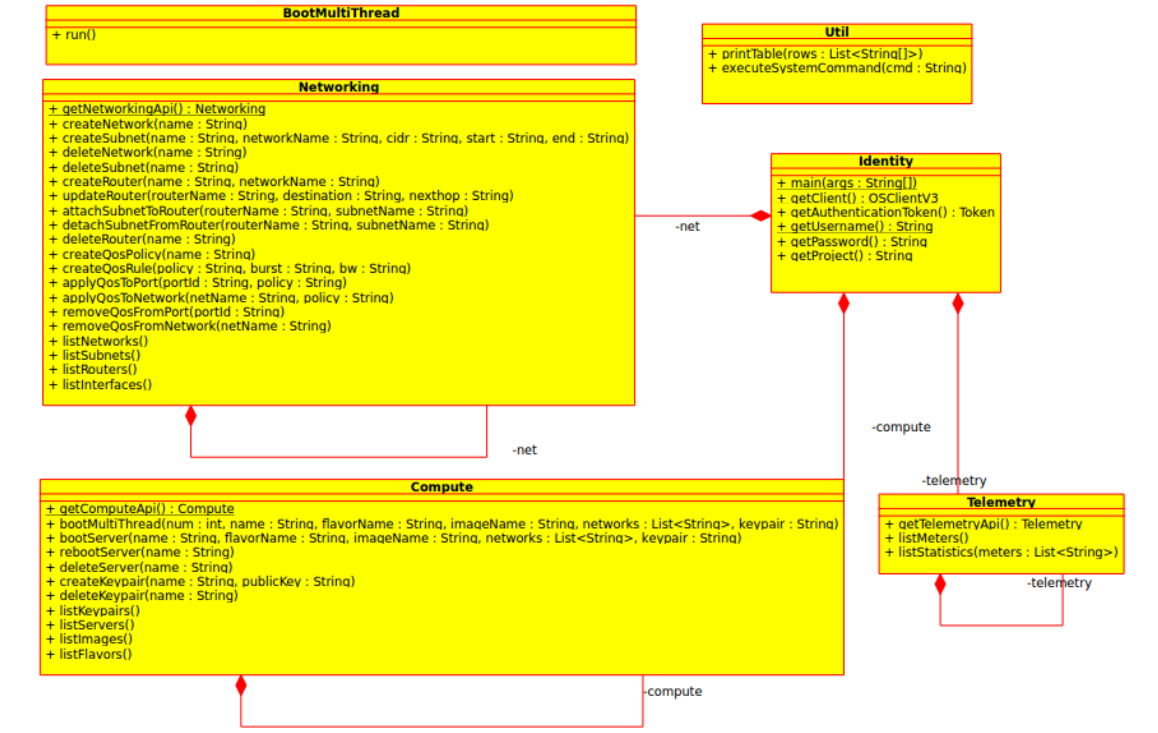

Figura 9: Diagrama UML da ferramenta Java desenvolvida.

Para criar uma rede definida por software funcional e que possa ser utilizada por diferentes aplicações é preciso inicialmente criar uma rede no OpenStack. No momento de criação da rede deve-se especificar somente o nome da nova rede, sendo que o usuário deve estar autenticado para realizar essa operação.

Criada a rede no OpenStack, é preciso criar uma sub-rede e associa-la a rede criada anteriormente para reservar um bloco de endereços IP que será utilizado pelas máquinas virtuais que se conectarem a essa rede. Para criar a sub-rede, deve-se especificar os seguintes argumentos: o nome da sub-rede (este nome não precisa ser único, mas é recomendável que não se use nomes repetidos), o nome da rede para a qual essa sub-rede será atribuída, o espaço de endereçamento IP em formato do tipo "10.10.10.0/24"(deve-se tomar cuidado para não utilizar uma faixa de endereços IP que já foi reservada por outra rede) e os endereços IP de início e fim do bloco. Além disso, a nova sub-rede é criada no mesmo projeto especificado pelo usuário no momento da autenticação, com um servidor DHCP habilitado (sem esse servidor seria preciso atribuir manualmente um endereço IP para cada máquina virtual) e com o servidor DNS do Google ("8.8.8.8") adicionado a lista de servidores DNS. No momento da criação, a sub-rede também é inserida em um grupo de segurança e, no caso da ferramenta desenvolvida, esse grupo é o grupo de segurança padrão (*default*) e é utilizado para estabelecer regras de segurança dentro da sub-rede. Para permitir que o ping e o acesso remoto via ssh funcionem dentro da sub-rede, é preciso adicionar regras de

segurança que permitam esse tipo de tráfego de dados na sub-rede. No caso do ping, deve-se permitir o tráfego de mensagens ICMP na rede e no caso do ssh nas máquinas virtuais é preciso habilitar o tráfego TCP na porta 22. Sem essa configuração de regras de segurança, essas funcionalidades podem não funcionar. Se nenhum grupo de segurança é especificado no momento da criação da sub-rede, ela é automaticamente inserida no grupo de segurança padrão.

Em seguida é preciso criar um roteador virtual para rotear o tráfego de dados dentro da sub-rede e permitir o acesso externo às máquinas virtuais. Esse roteador deve ser criado especificando-se o nome do roteador e o nome da rede externa a qual ele será conectado (*gateway* da rede virtual). O roteador pode se conectar a várias sub-redes. Após criado, o roteador deve ser conectado a sub-rede criada utilizando-se os nomes do roteador e da sub-rede, respectivamente. Feito isso, o último passo é adicionar a rede virtual criada a tabela de roteamento do roteador da rede externa para permitir que as máquinas virtuais sejam acessadas a partir do ambiente externo e também para permitir que essas máquinas acessem a Internet. Com isso, temos uma rede virtual funcional e que já pode ser utilizada.

O próximo passo é lançar diferentes máquinas virtuais para permitir que aplicações sejam executadas. Para instanciar uma máquina virtual é preciso passar como parâmetros o nome do novo servidor, o nome do *flavor* desejado, o nome da imagem e uma lista com os identificadores (IDs) das redes a que essa máquina virtual deve ser conectada. Para que essa máquina virtual seja acessível por meio de ssh também é preciso especificar um *keypair* no momento de sua criação (a chave de acesso pode ser gerada com o comando `ssh-keygen` do Linux por exemplo e então ser adicionada a instalação OpenStack). No estado atual, a ferramenta também permite que todos elementos virtuais criados sejam deletadas e ainda oferece opções para listar diferentes informações a respeito da nuvem OpenStack.

A ferramenta desenvolvida ainda possibilita a coleta de estatísticas a partir do Ceilometer e permite a gerência de políticas de qualidade de serviço. A API de qualidade de serviço do Neutron ainda não foi completamente desenvolvida e pode funcionar parcialmente dependendo do tipo de *hypervisor* e plug-ins utilizados na instalação do OpenStack. No momento, a qualidade de serviço está baseada em um conjunto de políticas e regras. As políticas de qualidade de serviço atuam como contêineres para um conjunto de regras, podendo ser aplicadas a uma porta específica de uma rede virtual ou até mesmo a uma rede virtual inteira (nesse caso, todas as portas dessa rede virtual irão herdar e aplicar essa política de qualidade de serviço). Já as regras de qualidade de serviço são utilizadas para especificar parâmetros de qualidade de serviço a serem seguidos. Nessa versão do OpenStack, pode-se apenas criar regras que limitam a máxima largura de banda (`qos-bandwidth-limit-rule`) a ser utilizada por uma porta da rede virtual. Essa largura de banda é especificada em kilobits por segundo e este valor pode ser atualizado a qualquer momento, proporcionando grande flexibilidade a rede virtual. Essa regra depende também de um valor de *burst* que ajuda a regular a largura de banda, impedindo que a política de qualidade de serviço seja aplicada prematuramente às portas especificadas. Esse valor é especificado em kilobits e representa a quantidade de dados que pode ser enviada antes que o limite de largura de banda seja de fato aplicado a uma porta, sendo importante para garantir o funcionamento correto da limitação da largura de banda. Se o valor do *burst* for muito baixo, o uso da largura de banda pode ser prejudicado (para tráfego TCP, recomenda-se que esse valor seja

igual a 80% do valor limite da largura de banda).

A seguir é apresentado o menu de ajuda da ferramenta desenvolvida, com as opções disponíveis.

```
usage: java -jar network.sdn-1.0.jar <args>
 -applyQosPolicyToNet <arg>   Apply a qos policy to the specified
                              network. -applyQosPolicyToNet <network
                              name:policy name>
 -applyQosPolicyToPort <arg>  Apply a qos policy to the specified port.
                              -applyQosPolicyToPort <port id:policy
                              name>
 -attachSubnetToRouter <arg>  Attach subnet to router.
                              -attachSubnetToRouter <router name:subnet
                              name>
 -bootMth <arg>               Boot servers using multithread. -bootMth
                              <number of servers:server prefix:flavor
                              name:image name:network IDs>
 -bootServer <arg>            Boot new server. -bootServer <server
                              name:flavor name:image name:network IDs>
 -createKeypair <arg>         Create new keypair. -createKeypair
                              <keypair name:public key>
 -createNetwork <arg>         Create new network. -createNetwork
                              <network name>
 -createQosPolicy <arg>       Create a new qos policy. -createQosPolicy
                              <policy name>
 -createQosRule <arg>         Create a new qos rule. -createQosRule
                              <policy name:max burst in Kb:max
                              bandwidth in Kbps>
 -createRouter <arg>          Create new router. -createRouter <router
                              name:external network name>
 -createSubnet <arg>          Create new subnet. -createSubnet <subnet
                              name:network name:address space:start
                              address:end address>
 -default                     Use default user information.
 -deleteKeypair <arg>         Delete existing keypair. -deleteKeypair
                              <keypair name>
 -deleteNetwork <arg>         Delete network. -deleteNetwork <network
                              name>
 -deleteRouter <arg>          Delete router. -deleteRouter <router
                              name>
 -deleteServer <arg>          Delete server. -deleteServer <server
                              name>
 -deleteSubnet <arg>          Delete subnet. -deleteSubnet <subnet
                              name>
 -detachSubnetFromRouter <arg> Detach subnet from router.
                              -detachSubnetFromRouter <router
                              name:subnet name>
 -dkey                        Use default authentication endpoint.
```

```
-g                          Get authentication information from
                            environment variables. (OS_USERNAME,
                            OS_PASSWORD, OS_AUTH_URL,
                            OS_PROJECT_NAME, OS_PROJECT_DOMAIN_NAME,
                            OS_USER_DOMAIN_NAME)
-h                          Help.
-key <arg>                  Set authentication endpoint. -key
                            <authentication endpoint>
-keypair <arg>              Specify keypair to boot servers. -keypair
                            <keypair name>
-listFlavors                List all flavors.
-listImages                 List all images.
-listInterfaces             List all interfaces.
-listKeypairs               List all keypairs.
-listMeters                 List telemetry meters.
-listNetworks               List all networks.
-listRouters                List all routers.
-listServers                List all servers.
-listServices               List available OpenStackServices.
-listSubnets                List all subnets.
-p <arg>                    Set password. -p <password>
-pd <arg>                   Set project domain. -pd <project domain
                            name>
-rebootServer <arg>         Soft reboot the specified server.
                            -rebootServer <server name>
-removeQosFromNet <arg>     Remove the qos policy from the specified
                            network. -removeQosFromNet <network name>
-removeQosFromPort <arg>    Remove the qos policy from the specified
                            port. -removeQosFromPort <port id>
-t <arg>                    Set project. -t <project name>
-u <arg>                    Set username (password and project must
                            be specified as well). -u <username>
-ud <arg>                   Set user domain. -ud <user domain name>
-updateRouter <arg>         Update router. Add new route.
                            -updateRouter <router
                            name:destination:next hop>
```

Menu de ajuda da ferramenta desenvolvida.

4.3 Testes com qualidade de serviço

Nesta seção serão descritos e analisados os testes realizados com a API de qualidade de serviço do Neutron. Foram realizados testes contemplando diferentes cenários de congestionamento da rede e aplicações utilizando diferentes protocolos de transporte (TCP/UDP). Durante os testes, os fluxos de dados foram gerados utilizado o programa `iperf3` (software livre disponível em `https://github.com/esnet/iperf`). Esse programa permite simular diferentes tipos de tráfego de dados em uma rede, tanto TCP quanto UDP, além de oferecer diversas opções para monitorar o tráfego de dados gerado por essa ferramenta.

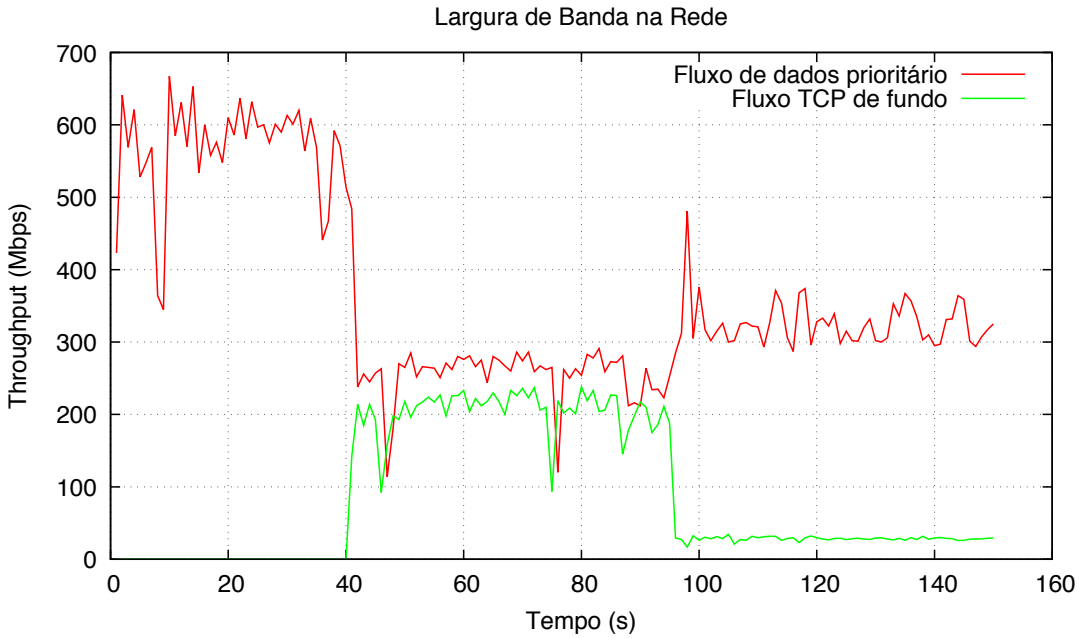

Figura 10: Fluxos de dados na rede definida por software. O tráfego de fundo foi gerado por uma única fonte TCP.

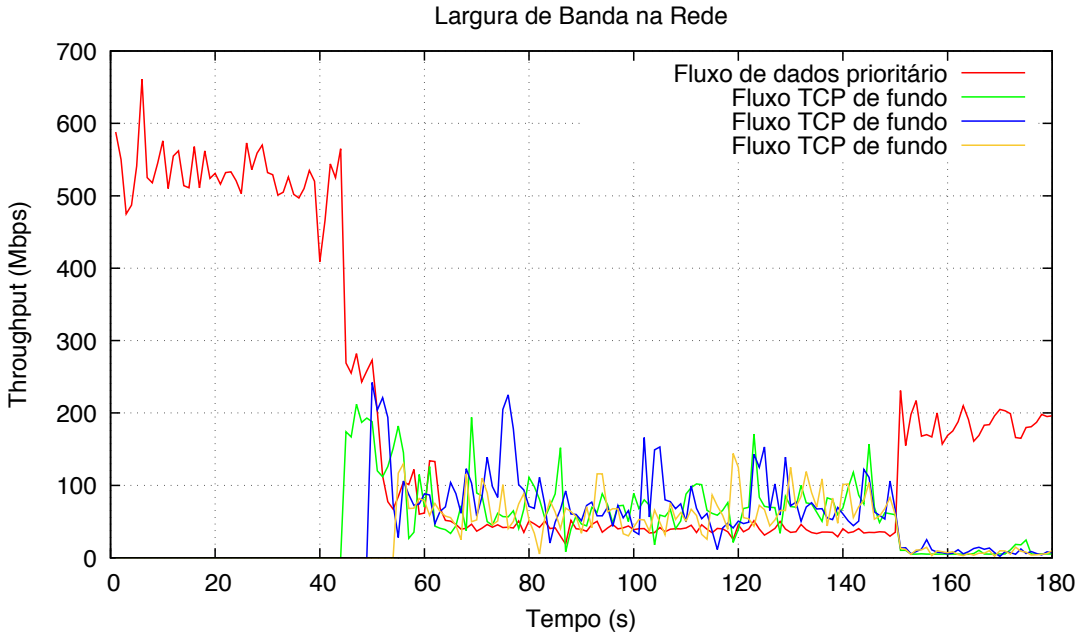

Figura 11: Fluxos de dados na rede definida por software. O tráfego de fundo foi gerado por múltiplas fontes TCP.

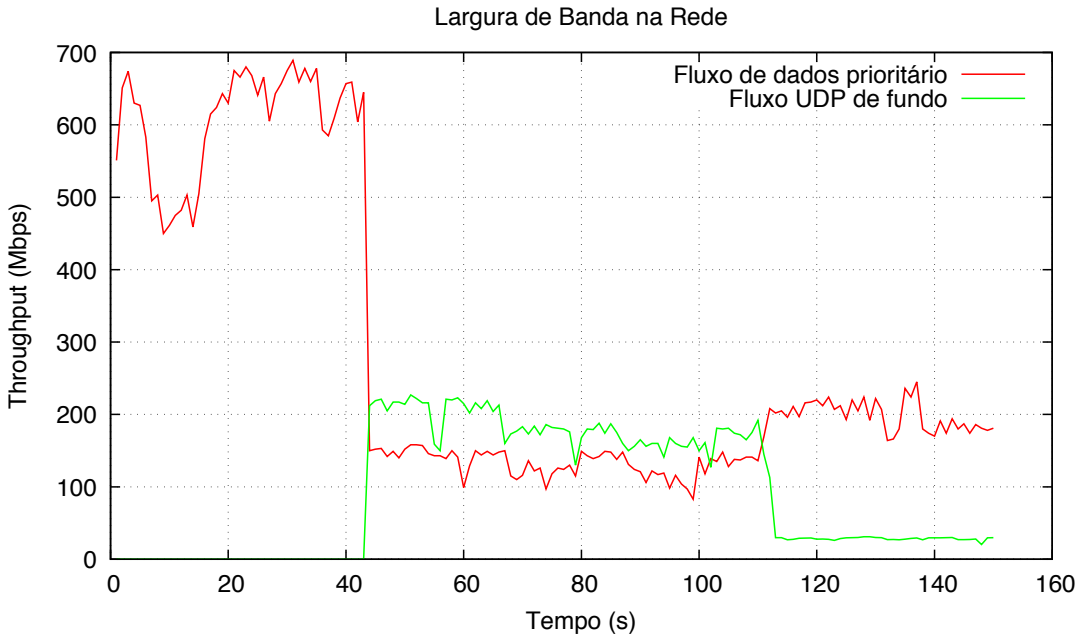

Figura 12: Fluxos de dados na rede definida por software. O tráfego de fundo foi gerado por uma única fonte UDP.

O primeiro teste realizado, representado pelo gráfico da Figura 10, mostra o que acontece com o *throughput* de um fluxo de dados prioritário na rede definida por software do OpenStack quando uma outra fonte, neste caso com protocolo TCP, começa a transmitir simultaneamente na mesma rede virtual. Inicialmente, como mostra a Figura 10, apenas o fluxo de dados TCP prioritário foi estabelecido na rede definida por software e é transmitido a taxas que chegam a ultrapassar os 600 Mbps. Aproximadamente aos 40 segundos da execução do teste, uma fonte secundária passa a competir pela largura de banda disponível e acaba fazendo com que o *throughput* do fluxo de dados prioritário caia bastante, ficando abaixo dos 300 Mbps. Isso pode prejudicar o fluxo de dados prioritário caso a aplicação em questão exigisse altas taxas de transmissão para manter a qualidade do serviço e operar corretamente, tornando este um cenário interessante para a aplicação de políticas de qualidade de serviço para garantir que esse fluxo de dados tivesse de fato maior prioridade para uso da largura de banda disponível. O TCP implementa mecanismos para controle de congestionamento e é por isso que o fluxo de dados secundário não consome a maior parte da largura de banda, proporcionando uma divisão mais igualitária dos recursos disponíveis, evitando uma perda excessiva de pacotes e, consequentemente, um alto número de retransmissões. Por volta do centésimo segundo do teste, foi aplicada uma política de qualidade de serviço ao fluxo de dados secundário, limitando sua taxa de transmissão a 30 Mbps. Com isso, o fluxo de dados secundário deixou de consumir boa parte da largura de banda e o fluxo de dados prioritário conseguiu elevar sua taxa de transmissão, ficando acima dos 300 Mbps no restante do teste. Com políticas desse tipo, um administrador da rede pode facilmente priorizar diferentes fluxos de dados conforme a demanda e garantir

que aplicações críticas tenham acesso aos recursos de que necessitam. Num cenário voltado ao modelo de negócios, clientes com um plano de dados superior poderiam consumir uma largura de banda maior. As políticas de qualidade de serviço também seriam interessantes para evitar que a rede ficasse ociosa, possibilitando uma alocação dinâmica de recursos. Assim que o fluxo de dados prioritário cessasse, a limitação da largura de banda do fluxo secundário poderia ser removida, por exemplo. Isso permitiria que toda largura de banda da rede definida por software fosse utilizada pelo fluxo secundário, melhorando qualidade dessa transmissão.

O gráfico da Figura 11, ilustra um cenário similar ao da Figura 10 com a diferença que nesse teste foram utilizadas quatro fontes distintas para gerar tráfego TCP de fundo. Inicialmente, a rede só era utilizada para a transmissão do fluxo de dados prioritário, o qual passou gradativamente a competir por largura de banda a partir do segundo 40 quando diferentes fontes também iniciaram sua transmissão. Como esperado, o fluxo prioritário teve sua taxa de transmissão reduzida drasticamente ficando abaixo dos 100 Mbps. Como todo tráfego de dados era TCP, pode-se observar uma divisão mais igualitária da largura de banda. Na parte final do teste, novamente uma política de qualidade de serviço para limite da largura de banda em 30 Mbps foi aplicada em todo tráfego de fundo, permitindo que o *throughput* do fluxo de dados prioritário aumentasse e oscilasse em torno dos 200 Mbps.

O gráfico da Figura 12 ilustra um cenário onde o tráfego de fundo é gerado por uma fonte com protocolo UDP. Inicialmente o fluxo de dados prioritário é capaz de atingir velocidades superiores a 600 Mbps, mas tem seu desempenho afetado quando passa a competir por largura de banda com um fluxo de dados UDP de fundo por volta do segundo 40 do teste. O protocolo UDP não implementa qualquer mecanismo para controle de congestionamento e é por isso que consome a maior parte largura de banda da rede definida por software, ignorando a existência do fluxo de dados prioritário. O UDP também não realiza retransmissões e acaba aumentado o número de pacotes perdidos na rede. O fluxo de dados prioritário por sua vez é um fluxo TCP e, detectando o congestionamento causado pelo tráfego UDP, reduz sua taxa de transmissão na tentativa de evitar que a rede fique sobrecarregada. Por isso, como pode ser observado na Figura 12, o *throughput* do fluxo prioritário não ultrapassa os 200 Mbps e fica sempre abaixo da velocidade atingida pelo fluxo secundário. No final do teste, uma política de qualidade de serviço que limita a largura de banda em 30 Mbps é aplica ao fluxo UDP de fundo e, assim, permite que o fluxo prioritário transmita com velocidades em torno dos 200 Mbps.

5 Conclusão

No decorrer do projeto, verificou-se o funcionamento e modelos de gerência para as redes definidas por software. Esse novo paradigma pode deixar as redes de computadores mais flexíveis, possibilitando a interoperabilidade entre equipamentos de diferentes fabricantes e aumentando a taxa de inovação na área ao proporcionar um plano de controle programável e diminuir a resistência a mudanças gerada pelo grande número de equipamentos e protocolos instalados. O OpenStack é uma ferramenta versátil para prover infraestrutura como um serviço e orquestrar o funcionamento de uma nuvem computacional, no entanto apresenta

um processo de instalação trabalhoso e pouco intuitivo. Há várias opções e módulos que devem ser configurados, sendo que processo de instalação esbarra em inúmeros erros antes de ser concluído com sucesso. Foi desenvolvida uma ferramenta em Java que permite a configuração e o gerenciamento de redes definidas por software em nuvens computacionais OpenStack. Também foi possível estudar a aplicação de políticas de qualidade de serviço associadas às redes definidas por software. Os testes realizados mostraram que as políticas de qualidade de serviço podem ser importantes ferramentas para o estabelecimento de fluxos de dados prioritários conforme a demanda e para a implantação de diferentes modelos de negócios. As políticas de qualidade de serviço também podem proporcionar um ambiente dinâmico onde a largura de banda disponível é melhor aproveitada. Conforme verificado, múltiplos fluxos de dados podem competir por largura de banda e congestionar a rede definida por software, por isso é importante a aplicação de políticas de qualidade de serviço para garantir que as aplicações tenham acesso a largura de banda de que necessitam. As políticas de qualidade de serviço ainda estão em desenvolvimento e devem assumir um papel de destaque na gerência de redes virtuais. Essa nova abordagem possibilita que administradores gerenciem os recursos disponíveis de forma mais eficiente, reduzindo custos e garantindo um ambiente mais flexível.

6 Agradecimentos

Gostaria de agradecer ao Carlos Senna pela ajuda e disponibilidade para esclarecer dúvidas e discutir soluções para o projeto. Esse projeto contou com o apoio do CNPq/ PIBIC.

Referências

[1] Wolfgang Braun. Software-Defined Networking Using OpenFlow: Protocols, Applications and Architectural Design Choices. *Future Internet*, Volume 6:302–336, May 2014.

[2] Open Networking Foundation. Software-Defined Networking: The New Norm for Networks, April 2012.

[3] IBM. Discover OpenStack: The Compute components Glance and Nova. `http://www.ibm.com/developerworks/cloud/library/cl-openstack-nova-glance/`. [Online: Acessado 01-Dezembro-2016].

[4] IBM. Discover OpenStack: The Identity component Keystone. `http://www.ibm.com/developerworks/cloud/library/cl-openstack-keystone/`. [Online: Acessado 01-Dezembro-2016].

[5] IBM. Discover OpenStack: The Networking component Neutron. `http://www.ibm.com/developerworks/cloud/library/cl-openstack-neutron/`. [Online: Acessado 01-Dezembro-2016].

[6] Nick McKeown, Tom Anderson, Hari Balakrishnan, Guru Parulkar, Larry Peterson, Jennifer Rexford, Scott Shenker, and Jonathan Turner. OpenFlow: Enabling Innovation in Campus Networks. *ACM (Association for Computing Machinery) SIGCOMM (Special Interest Group on Data Communications) Computer Communication Review*, March 2008.

[7] Peter Mell and Timothy Grance. The NIST Definition of Cloud Computing. *National Institute of Standards and Technology - U.S. Department of Commerce*, Special Publication 800-145, September 2011.

[8] Microsoft. What is qos? `https://technet.microsoft.com/pt-br/library/cc757120(v=ws.10).aspx`. [Online: Acessado 01-Dezembro-2016].

[9] Sakir Sezer, Sandra Scott-Hayward, Pushpinder Kaur Chouhan, Barbara Fraser, David Lake, Jim Finnegan, Niel Viljoen, Marc Miller, and Navneet Rao. Are we ready for SDN? Implementation challenges for software-defined networks. *Communications Magazine, IEEE*, Volume 51(Issue 7):36–43, July 2013.

[10] Qi Zhang, Lu Cheng, and Raouf Boutaba. Cloud computing: state-of-the-art and research challenges. *Journal of Internet Services and Applications*, Volume 1(Issue 1):7–18, April 2010.