Universidade Federal de Juiz de Fora
Instituto de Ciências Exatas
Bacharelado em Ciência da Computação

Provisão de QoS em SDN:
Um estudo de caso de reserva dinâmica de largura de banda para aplicações distribuídas de alto desempenho

Bruno José Cesário de Almeida Martins

JUIZ DE FORA
NOVEMBRO, 2018

Provisão de QoS em SDN:
Um estudo de caso de reserva dinâmica de largura de banda para aplicações distribuídas de alto desempenho

Bruno José Cesário de Almeida Martins

Universidade Federal de Juiz de Fora

Instituto de Ciências Exatas

Departamento de Ciência da Computação

Bacharelado em Ciência da Computação

Orientador: Alex Borges Vieira

Coorientador: Alexandre Tavares de Oliveira

JUIZ DE FORA

NOVEMBRO, 2018

PROVISÃO DE QOS EM SDN:
Um estudo de caso de reserva dinâmica de largura de banda para aplicações distribuídas de alto desempenho

Bruno José Cesário de Almeida Martins

MONOGRAFIA SUBMETIDA AO CORPO DOCENTE DO INSTITUTO DE CIÊNCIAS EXATAS DA UNIVERSIDADE FEDERAL DE JUIZ DE FORA, COMO PARTE INTEGRANTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE BACHAREL EM CIÊNCIA DA COMPUTAÇÃO.

Aprovada por:

Alex Borges Vieira
Doutor

Alexandre Tavares de Oliveira
Bacharel

Mario Antonio Ribeiro Dantas
Doutor

Luciano Jerez Chaves
Doutor

JUIZ DE FORA
29 DE NOVEMBRO, 2018

A Deus, por ser essencial em minha vida, autor de meu destino, meu guia, socorro presente na hora da angústia.

Aos meus pais, meu irmão, minha esposa e a toda minha família que, com muito carinho e apoio, não mediram esforços para que eu chegasse até esta etapa de minha vida.

Resumo

Diante da crescente demanda por aplicações distribuídas de alto desempenho e considerando o grande avanço das redes definidas por *software* (SDN), neste trabalho, investigamos como melhorar o tráfego de rede dessas aplicações. Usamos os conceitos de qualidade de serviço (QoS) para priorizar o tráfego dessas aplicações e minimizar o tráfego de outras aplicações não prioritárias. Em suma, conceituamos as principais partes do trabalho; apresentamos uma pesquisa sobre os trabalhos relacionados existentes; comparamos os tráfegos priorizados e não priorizados de aplicações distribuídas de alto desempenho; e, finalmente, mostramos o motivo da aplicação dinâmica da qualidade de serviço em uma rede definida por *software*. Nossa principal contribuição depende de uma arquitetura SDN e do desenvolvimento dos módulos necessários para fornecer a capacidade de programação de QoS nas SDNs. Nossos resultados mostram que uma QoS é necessária para priorização de tráfego. No entanto, depois de aplicar as regras para priorizar uma determinada classe de tráfego, as demais aplicações não priorizadas podem ter seu tráfego degradado, mesmo quando a rede não possuir tráfego priorizado. Desta forma, apresentamos também um cenário dinâmico de QoS, onde a rede monitora dinamicamente seus fluxos e aplica as regras de QoS somente quando necessário. Como consequência, de acordo com nossos resultados, a rede garante os recursos para priorizar o tráfego, e também permite que o restante flua para utilizar todos os recursos da rede, quando a rede não possuir tráfego prioritário.

Palavras-chave: Redes Definidas por *Software*, Qualidade de Serviço, Aplicações Distribuídas.

Abstract

In face of the growing demand for high-performance distributed applications and considering the great advance of software-defined networks, in this work, we investigate how to improve the network traffic of these applications. We use the concepts of quality of service in order to prioritize the traffic of these applications and to minimize the traffic of other non-prioritized applications. In short, we conceptualize the main parts of the work; we present a survey about the existing related work; we compare the prioritized and non-prioritized traffics of high-performance distributed applications; and finally, we show the reason for applying quality of service in a software-defined network dynamically. Our main contribution relies on an SDN architecture and the development of the necessary modules to provide QoS programmability in SDN. Our results show that an QoS is necessary for traffic prioritization. However, once one applies the rules to prioritize a given class of traffic, the remainder non-prioritized applications may have its traffic degraded, even when the network is offloaded. In this way, we also present a dynamic QoS scenario, were the network dynamically monitors its flows and apply the QoS rules only when it is necessary. As consequence, according to our results, network guarantees the resources to prioritize traffic, and also allows the remainder flows to utilize all network resources, when the network is offloaded.

Keywords: Software Defined Networking, Quality of Service, Distributed Applications.

Agradecimentos

A Deus pois sem ele eu não teria forças para essa longa jornada.

A minha mãe Rosânia, meu pai Ary e meu irmão Vitor por sempre estarem ao meu lado em todas as minhas conquistas, me apoiando e motivando.

Agradeço também a minha linda esposa, Marília, que de forma especial e carinhosa meu deu força e coragem, me apoiando nos momentos de dificuldades.

A todos os meus parentes, pelo encorajamento e apoio.

Ao professor Alex e meu grande amigo Alexandre pela orientação, amizade e principalmente, pela paciência, sem a qual este trabalho não se realizaria.

Ao meu grande amigo Cláudio que foi um grande parceiro durante todo o curso.

Aos professores do Departamento de Ciência da Computação pelos seus ensinamentos e aos funcionários do curso, que durante esses anos, contribuíram de algum modo para o nosso enriquecimento pessoal e profissional.

"O sucesso nada mais é que ir de fracasso em fracasso sem que se perca o entusiasmo".

Winston Churchill

Conteúdo

Lista de Figuras

Lista de Tabelas

Lista de Abreviações

AN	*Active Networks*
API	*Application Programming Interface*
BFD	*Bidirectional Forwarding Detection*
DCC	Departamento de Ciência da Computução
GSMP	*General Switch Management Protocol*
HFSC	*Hierarchical Fair-Service Curve*
IEEE	*Institute of Electrical and Electronic Engineers*
IETF	*Internet Engineering Task Force*
IPFIX	*IP Flow Information Export*
IPv6	*Internet Protocol version 6*
LACP	*Link Aggregation Control Protocol*
LLDP	*Link-Layer Discovery Protocol*
MAC	*Media Access Control*
NIC	*Network Interface Card*
QoS	*Quality of Service*
RSPAN	*Remote SPAN*
RSTP	*Rapid Spanning Tree Protocol*
SDN	*Software-Defined Networks*
SPAN	*Switch Port Analyzer*
SPBM	*Shortest Path Bridging Mac-in-Mac mode*
STP	*Spanning Tree Protocol*
TCP	*Transmission Control Protocol*
UDP	*User Datagram Protocol*
UFJF	Universidade Federal de Juiz de Fora
VLAN	*Virtual LAN*
VM	*Virtual Machine*
VoIP	*Voice over Internet Protocol*

1 Introdução

O aumento e a popularização da Internet ao redor do mundo fez crescer a busca por aplicações que ampliam a comunicação e a interatividade entre pessoas. Nesse cenário de constante evolução, podemos destacar o crescimento de aplicações como os serviços de *streaming*, jogos *online*, serviços de Voz sobre IP (*Voice over Internet Protocol* – VoIP) e os sistemas de saúde à distância. Podemos denotar tais aplicações como aplicações distribuídas de alto desempenho (OLIVEIRA et al., 2018).

Essas aplicações exigem, cada vez mais, uma grande demanda de requisitos de rede para que seu funcionamento seja considerado bom e satisfatório a seus usuários. Largura de banda, retardo e variação estatística do retardo (*jitter*) são exemplos de requisitos que essas diversas aplicações requerem da rede onde trafegam.

Esses requisitos de rede são qualificados como parâmetros de qualidade de serviço (*Quality of Service* — QoS). Muitas vezes, esses parâmetros de QoS são especificados manualmente e diretamente nos equipamentos de rede. Isso torna a gerência descentralizada e trabalhosa, exigindo dos administradores das redes conhecimentos de programação específicos de um determinado equipamento. Logo é percebido que uma simples troca de fabricante ou até mesmo de *firmware* pode ocasionar um grande incômodo, trazendo uma grande demanda de retrabalho e novas aprendizagens. Além disso, a escassez de recursos e a pouca flexibilidade das redes atuais nos fazem procurar alternativas que facilitem a especificação e a implementação de QoS na rede.

Uma dessas alternativas propostas são as Redes Definidas por *Software* (SDN – acrônimo de *Software-Defined Networks*). Tais redes vêm ganhando, cada vez mais, grande espaço no cenário de redes atual. Uma de suas principais vantagens é a visão global da rede que propõe aos seus administradores uma gerência centralizada e de fácil programação. A ideia de SDN é trazer novas abstrações na rede, simplificar a gerência da mesma e facilitar a sua evolução (KREUTZ et al., 2015).

Entretanto, diante do crescimento das redes definidas por *software*, percebe-se também que existe uma demanda em especificar requisitos de QoS, haja vista que existem

aplicações distribuídas de alto desempenho que trafegam sobre essas redes.

Portanto, indaga-se: como melhorar o tráfego de uma aplicação distribuída de alto desempenho sobre as redes definidas por *software*, utilizando, para isso, os conceitos de qualidade de serviço?

Assim, o objetivo geral da presente pesquisa é melhorar o tráfego de uma aplicação distribuída de alto desempenho em redes definidas por *software*, aplicando dinamicamente os conceitos de QoS, em especial a reserva de banda, prejudicando o mínimo possível o tráfego de outras aplicações.

Para tanto, foram delineados os seguintes objetivos específicos: conceituar as principais partes do trabalho, sendo elas a qualidade de serviço, as aplicações distribuídas de alto desempenho e as redes definidas por *software*; fazer um levantamento de trabalhos relacionadas a QoS em SDN; fazer uma comparação da qualidade dos tráfegos das aplicações que devam ser priorizadas, para tanto aplicando ou não os conceitos de QoS; e a razão de aplicar QoS em uma SDN de forma dinâmica.

Parte-se da hipótese de que se aplicarmos uma reserva de banda dinamicamente nas SDNs, podemos tanto melhorar o tráfego das aplicações distribuídas de alto desempenho, quanto prejudicar minimamente o tráfego de outras aplicações, o que no cenário de utilização real seria o ideal.

Assim, para viabilizar o teste da hipótese, realiza-se um pesquisa de finalidade básica estratégica, objetivo descritivo e exploratório, sob o método hipotético-dedutivo, com abordagens qualitativa e quantitativa, realizadas com procedimentos bibliográficos, documentais e experimentais.

No segundo capítulo são estabelecidos os principais conceitos deste trabalho, passando pelas aplicações distribuídas de alto desempenho, pela QoS e pelas SDNs. É realizado também um levantamento de trabalhos implementados com os conceitos de provisão de QoS em SDN.

No terceiro capítulo, realiza-se uma análise da proposta do trabalho, enunciando a abordagem e seu funcionamento, bem como a topologia utilizada.

No quarto capítulo são apresentadas as ferramentas utilizadas no trabalho para a realização das simulações.

No quinto capítulo são apresentados os resultados experimentais e as discussões sobre o trabalho, verificando as diferenças entre as abordagens: avaliação dos testes sem QoS; avaliação dos testes com QoS; e avaliação dos testes com QoS aplicada dinamicamente.

Ao final, conclui-se que os objetivos do trabalho são atendidos com a confirmação da hipótese, indicando que a abordagem da aplicação da QoS de forma dinâmica nas SDNs, além de priorizar as aplicações distribuídas de alto desempenho, preocupa-se em prejudicar minimamente o tráfego de outras aplicações.

2 Revisão Bibliográfica

O objetivo deste capítulo é fornecer explicações detalhadas dos assuntos abordados neste trabalho, trazendo uma informação clara e objetiva, além das visões de outros autores em relação aos problemas e desafios da aplicação de qualidade de serviço em SDN.

2.1 Aplicações Distribuídas de Alto Desempenho

A Internet atual vem proporcionando a integração de vários serviços, como telefonia, vídeo conferência, dentre outros. Antes, cada um desses serviços operava em redes separadas, pois ainda não havia tecnologia suficiente para que elas trabalhassem em conjunto. Hoje em dia, a realidade não é a mesma. Todos esses serviços e processos estão convergindo para a grande Rede de Internet. Para cada aplicação criada para gerenciar esses serviços, são demandadas determinadas características de rede para que se tenha uma experiência melhor de sua utilidade. Fluxos multimídias e fluxos VoIP são exemplos de aplicações que requerem uma determinada robustez da rede e também possuem como característica o tráfego de dados separados, como aborda (MIRCHEV, 2015).

Tendo em vista o amplo conceito desse tema, nossa abordagem consiste em considerar como aplicações distribuídas de alto desempenho sistemas como VoIP, *streaming* multimídia, jogos *online* e serviços semelhantes a esses. Dentro de cada uma dessas aplicações, podemos destacar algumas características de rede que influenciam seu comportamento. (MORENO; SOARES, 2002) destacam no início de sua obra alguns requisitos de QoS, tais como retardo máximo, variação estatística máxima do retardo e taxa mínima de transmissão, que são alguns requisitos que as próprias aplicações exigem da rede. Na Tabela 2.1 são mostrados os impactos dos requisitos e as exigências de QoS para que algumas aplicações possam executar de maneira desejável na rede.

De fato, ao olharmos a Tabela 2.1, verifica-se que as maiores demandas da rede atualmente são áudio/vídeo, mas também aplicações que requerem uma interatividade muito grande entre pessoas, que são as aplicações tipo VoIP e vídeo conferência. Es-

Tabela 2.1: Comparativo de QoS em algumas aplicações

Aplicação	Confiabilidade	Atraso	Jitter	Largura de banda
WWW	Alta	Média	Baixa	Média
e-mail	Alta	Baixa	Baixa	Baixa
FTP	Alta	Baixa	Baixa	Média
SSH	Alta	Média	Média	Baixa
áudio/vídeo	Baixa	Alta	Alta	Média
VoIP	Baixa	Alta	Alta	Baixa

sas aplicações requerem uma grande robustez da rede nas quais elas operam, tanto no atraso da entrega dos pacotes pela rede, como também na média de variação do atraso. Logo, a procura por soluções a problemas dessa natureza é objeto de estudo de vários pesquisadores.

2.2 Qualidade de Serviço

Como visto na seção 2.1, as aplicações requerem determinadas características na rede para seu bom funcionamento. Antes de tudo, é preciso entender como este conceito de qualidade de serviço surgiu e como está sendo utilizado atualmente.

2.2.1 O surgimento da qualidade de serviço

Ao longo da criação da Internet, o aumento de usuários e de aplicações, o congestionamento de *links* de internet e os grandes fluxos de tráfego de rede geraram e geram grandes problemas aos usuários, tanto quanto aplicações de alto nível que trafegam na rede. Isso fez com que surgissem novas formas de contornar os problemas trazidos pela própria evolução da rede. Assim surge o conceito de qualidade de serviço (QoS, do termo em inglês *Quality of Service*).

O conceito de qualidade de serviço pode ser interpretado e ter definições diferentes, como abordam (KAMIENSKI; SADOK, 2000). Uma abordagem feita por (COLCHER, 2005) define QoS como as caraterísticas que variam entre serviços, sendo, em sua grande maioria, uma função do tipo de aplicação e da mídia transmitida. Já (VOGEL et al., 1995) definem QoS como o efeito coletivo que é provocado por um conjunto de características quantitativas e qualitativas de um serviço fornecido por um determinado

sistema, que se faz necessário para alcançar uma boa e determinada funcionalidade das aplicações. Por outro lado, temos que todas as abordagens de qualidade de serviço nos remetem a uma única interpretação, que é a capacidade de diferenciar e classificar o tráfego de rede e os tipos de serviços fornecidos (KAMIENSKI; SADOK, 2000).

A qualidade de serviço veio como uma forma de tentar solucionar os problemas das aplicações e dar aos seus usuários uma melhor experiência de rede. Na abordagem de (KAMIENSKI; SADOK, 2000), os autores trazem cinco maneiras diferentes de contorno do problema e sua aplicação na Internet. Entretanto, a implementação de QoS nas atuais estruturas de redes não é uma tarefa fácil e simples, demandando, muitas vezes, uma sobrecarga grande de trabalho para aqueles que a operam e a fazem funcionar. Muitas das vezes, são soluções que requerem configurações robustas e um alto nível de conhecimento de programação de *hardware*, que em sua maioria são específicos para determinadas aplicações. Podemos ter, como exemplo, grandes empresas fabricantes de *hardwares* de rede, como a empresa Cisco (CISCO, 2018), que trazem soluções de implementação desses requisitos, porém exigem dos administradores de rede um determinado conhecimento sobre a operação daquele equipamento.

2.2.2 Aplicação de QoS como forma de melhoria da rede

A aplicação de qualidade de serviço nas redes pode refletir no comportamento das aplicações. A interatividade do usuário com a aplicação pode ser desejável ou não, dependendo do nível de usabilidade da mesma e a qualidade de como essa aplicação é apresentada a esse usuário. O impacto de se aplicar QoS como forma de amenizar problemas como, por exemplo, o retardo da rede, pode provocar mudanças drásticas na aplicação, as quais são facilmente percebidas, principalmente pelos usuários, como é o caso do *streaming* de vídeo (OLIVEIRA et al., 2018). (D'SOUZA et al., 2016) demonstram a diferença entre se aplicar QoS na transmissão de vídeo e a não aplicação desse conceito, onde podemos perceber uma grande diferença entre as duas visualizações.

2.3 *Software-Defined Networks*

A grande dificuldade de administração e a escassez de recursos faz com que as redes tradicionais sejam denominadas como "ossificadas". As redes atuais tendem a tratar problemas e trazer soluções através da programação e do desenvolvimento de novos protocolos. Muitas das vezes, o que acontece é que esses protocolos tendem a tratar problemas específicos, como aborda (MIRCHEV, 2015), tornando a complexidade da rede cada vez mais alta. Podemos perceber também a demora em se concretizar algo nas redes tradicionais, tendo como exemplo, a mudança do protocolo IPv4 para o protocolo IPv6, como é abordado por (KREUTZ et al., 2015). Esses autores ainda enfatizam e exemplificam como um protocolo de roteamento pode demorar algo em torno de cinco a dez anos para ser projetado e avaliado, para logo depois ser disposto para utilização da rede.

O alto grau de complexidade em sua administração e o alto custo em investimentos na troca de soluções estão tornando as redes atuais enfraquecidas. Isso gera uma busca por novas soluções de redes que melhorem a experiência, os estudos, que facilite a expansão e traga uma nova abordagem. Nesta linha de raciocínio surgem as redes definidas por *software*.

2.3.1 Definição de SDN

Com o intuito de auxiliar nas questões de administração das redes, na facilidade de programação e uma possível expansão, surge o conceito de redes definidas por *software*, ou SDN (do inglês *Software-Defined Networks*). "De maneira sucinta, Redes Definidas por Software (SDN) são um paradigma de redes de computadores que define a realização do encaminhamento de pacotes de uma maneira diferente da tradicional" (COSTA et al., 2016).

SDN consiste na inserção de um elemento novo na rede chamado de "controlador", onde o mesmo fica responsável por definir as ações de encaminhamento de pacotes que um determinado elemento de rede deva ter (MACEDO et al., 2015).O paradigma SDN nos traz uma visão global da rede e também a torna mais inteligente e flexível.

Orquestrados por controladores distribuídos ou não distribuídos, esse é um novo paradigma que surge na intenção de evoluir as redes e torná-las estudáveis, sendo possível

moldar todos os seus tráfegos. A proposta do SDN consiste em separar o plano de controle (entidade controladora e tomadora de decisão) do plano de dados (entidade controladora que executa as ações determinadas pelo controlador) (COSTA et al., 2016).

Uma visão simplificada desse paradigma é mostrada na Figura 2.1 (KREUTZ et al., 2015), onde pode ser visto uma estrutura de redes com seus componentes (*switches* por exemplo) interligados entre si. Nessa figura é possível ver a separação entre o plano de controle e o plano de dados de cada componente na rede.

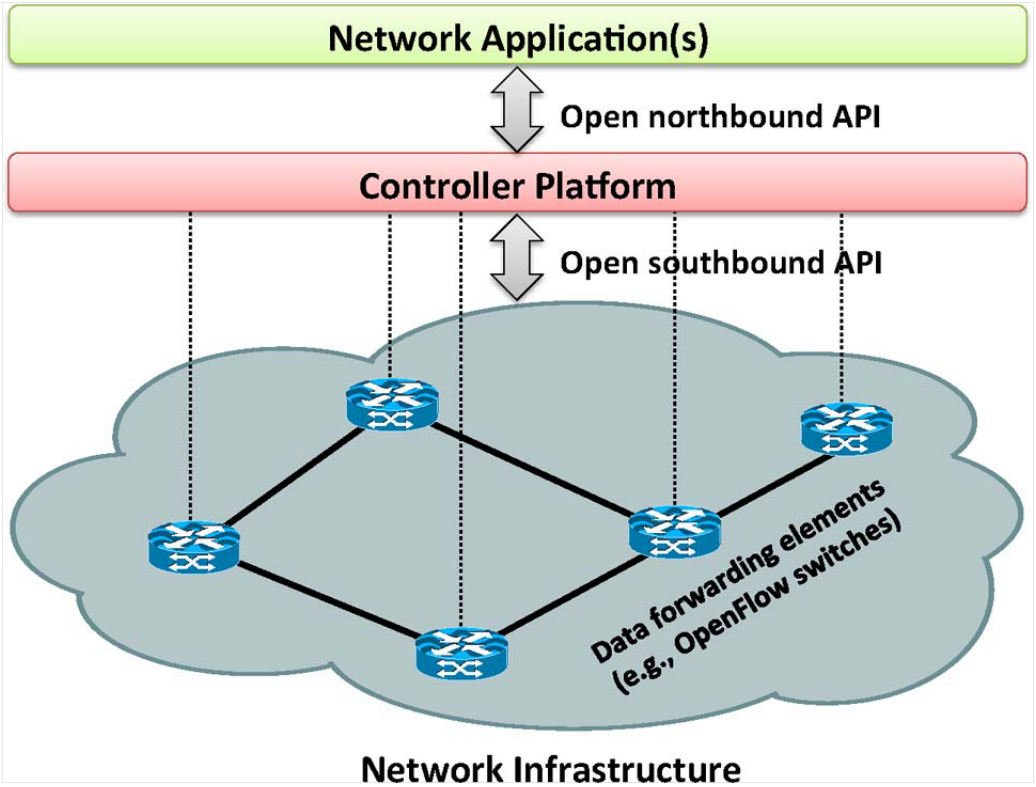


Figura 2.1: Visão simplificada de uma arquitetura SDN [(KREUTZ et al., 2015)]

De fato, ao analisarmos a Figura 2.1, vemos uma das principais diferenças entre as redes comuns e as SDNs, ou seja, nas redes comuns não é percebida essa separação lógica, pois tudo está agrupado diretamente nos dispositivos de rede. Já nas SDNs, vemos claramente que o plano de controle (ou podemos dizer a inteligência do equipamento de rede) é abstraído do mesmo e "levado" para o controlador da rede, deixando os dispositivos somente com a capacidade de entrega dos tráfegos, recebendo as ações de encaminhamento diretamente do controlador.

2.3.2 Breve histórico das SDNs

Para tentar solucionar o problema da ossificação das redes e a falta de espaço para inovação, surgiram várias tentativas de se tentar adicionar mais recursos de programação às redes. O intuito sempre foi tornar as redes mais flexíveis, permitindo a inovação, trazendo a facilidade da experimentação e melhorando o desenvolvimento, tanto de serviços quanto de protocolos (COSTA et al., 2016).

Redes Ativas (*Active Networks* – AN) foi uma das primeiras tentativas que tentou adicionar funcionalidades de programação na rede, podendo ser vista na publicação de (TENNENHOUSE et al., 1997). A ideia por trás dessas redes é que os comutadores de rede de uma Rede Ativa pudessem, de alguma forma, ter a capacidade de realizar cálculos personalizados sobre os pacotes, podendo, até mesmo, modificar o conteúdo contido neles (TENNENHOUSE et al., 1997), (KREUTZ et al., 2015).

Já em uma outra proposta, trazida pelo grupo de trabalho *OpenSignaling* (OPENSIG), houve uma tentativa de separar o *hardware* de comunicação e o *software* de controle. Nessa proposta, a ideia central era a manipulação do *hardware* através de interfaces programáveis, o que de fato já coincide com características do atual padrão SDN. Ao final do trabalho, foi gerado um protocolo chamado GSMP (*General Switch Management Protocol*), que era de uso geral, utilizado para controle de um *switch* (NUNES et al., 2014); (MACEDO et al., 2015); (GENERAL. . . , 2002); (COSTA et al., 2016).

Em seguida, diversas outras propostas surgiram ao longo do tempo, até se chegar no projeto Ethane. Esse projeto foi proposto por (CASADO et al., 2009) e desenvolvido pelo Departamento de Ciência da Computação da Universidade de Stanford. Nele é possível destacar a figura de um controlador centralizado, que atua gerenciando políticas de encaminhamento em um *switch* Ethane, o qual possuía uma tabela de fluxos que era alterada pelo controlador. Além disso, suas políticas de segurança eram de alto nível (NUNES et al., 2014); (FEAMSTER; REXFORD; ZEGURA, 2014); (CASADO et al., 2009). "De certa forma, pode se considerar o serviço de segurança em uma rede Ethane como sendo a primeira aplicação desenvolvida na história do SDN" (COSTA et al., 2016).

Sendo assim, a criação do projeto Ethane e a sua evolução serviram de base para

o desenvolvimento do paradigma SDN, o qual teve sua consolidação com a criação do protocolo OpenFlow – desenvolvido pela Universidade de Stanford (COSTA et al., 2016).

2.3.3 Componentes de uma arquitetura SDN simples

De uma maneira simplista, uma SDN pode ser caracterizada pela presença de uma entidade controladora que tem a função de determinar as ações executadas pelos comutadores de rede e os componentes de encaminhamento, os quais serão manipulados pelo controlador central através de um protocolo com uma segurança no canal de comunicação. A Figura 2.2 ilustra uma arquitetura SDN típica, onde podem ser vistos os seus principais componentes (COSTA et al., 2016).

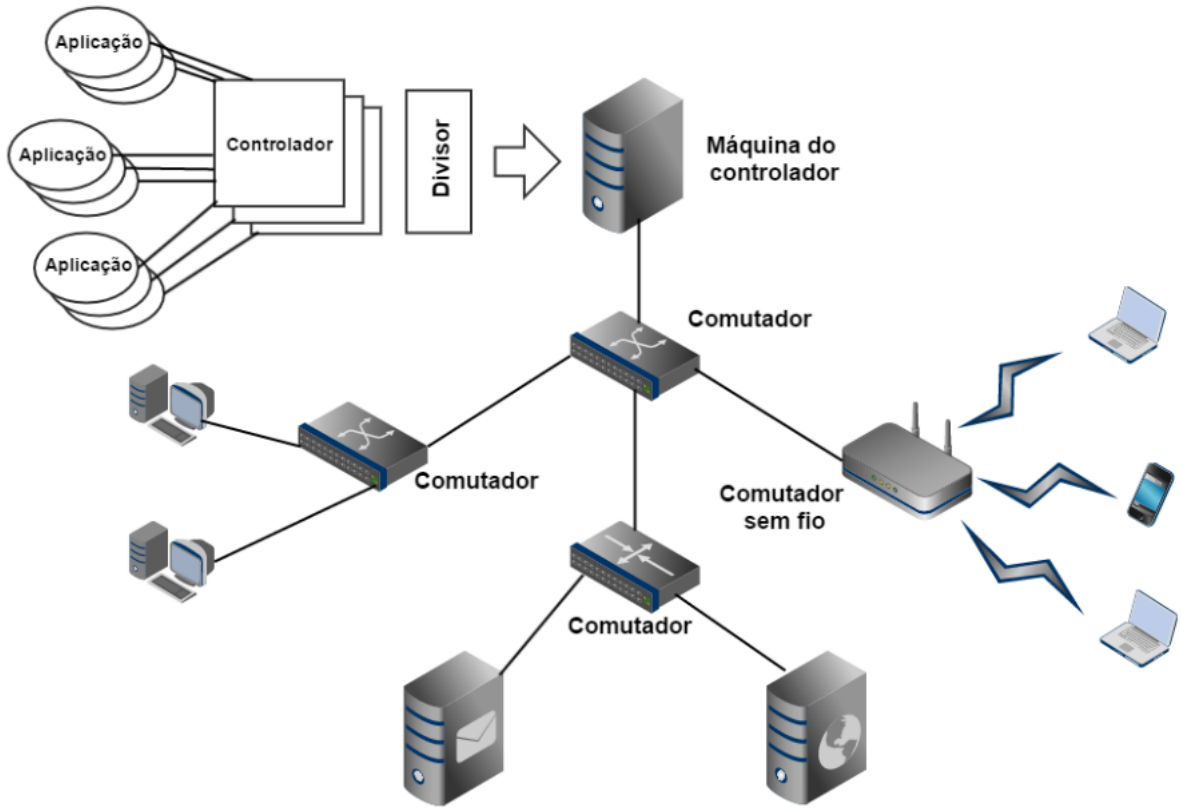


Figura 2.2: Arquitetura SDN típica (COSTA et al., 2016).

Um detalhamento simplificado – segundo (COSTA et al., 2016) – dessa arquitetura é apresentado a seguir:

- **Dispositivos de usuário final:** São equipamentos que estão ligados no final da rede, como os computadores pessoais. Tais elementos não são programáveis, mas sofrem interferência indireta das ações que acontecem na rede.

- **Equipamentos de comutação programáveis:** São comutadores de rede, iguais aos das redes tradicionais, porém sua inteligência é transferida para um controlador central. Esses equipamentos então atuam como um simples dispositivo de encaminhamento de pacotes, ou seja, atrelado a ele somente o plano de dados, já que o plano de controle é abstraído para o controlador central da rede.

- **Controlador:** Elemento responsável pela monitoração e modificação do comportamento dos elementos programáveis da rede. Utiliza um vasto conjunto de APIs para poder interagir com os comutadores programáveis e utiliza, nessa interação, canais seguros de comunicação fornecidos, por exemplo, pelo protocolo *OpenFlow*. Cabe ao controlador estabelecer a comunicação com todos os elementos programáveis da rede e obter uma visão unificada do *status* da rede.

- **API Southbound:** As interfaces *Southbound*, localizadas na plataforma de controle, são as responsáveis pela interação entre o controlador e os dispositivos de encaminhamento. Um exemplo desta API pode ser o protocolo *OpenFlow* (MCKEOWN et al., 2008), que é um dos protocolos mais utilizados nessa comunicação.

- **API Northbound:** Também localizadas na plataforma de controle, são as responsáveis pela interação entre o controlador e as aplicações de controle. Exemplo dessas APIs podem ser o Floodlight, o POX, o NOX e o RYU, que são *softwares* desenvolvidos em diversas linguagens que fornecem aos programadores e administradores de rede vários conjuntos de funções para a programação dos elementos programáveis da rede.

- **Aplicação:** São considerados os sistemas operacionais das SDNs, fornecendo, assim como nas redes tradicionais, conjuntos de funcionalidades. São exemplos dessas aplicações os *firewalls*, protocolos de roteamento, balanceadores de carga, dentre várias outras aplicações.

- **Virtualizador de rede (ou divisor):** É o componente da rede responsável pelo gerenciamento das redes virtuais criadas a partir das SDNs. É capaz também de prover uma divisão de recursos físicos na rede.

2.3.4 Adoção do paradigma SDN

Como SDN é um paradigma novo, ainda há provedores de serviço e usuários que tendem a não adotar essa nova forma de trabalho com redes. Um levantamento feito por (D'SOUZA et al., 2016) para tentar entender o que acontece com a não aderência das SDNs, considerou as opiniões de 50 engenheiros que estão no topo de grandes companhias de redes. A pesquisa levou em consideração os aspectos de QoS, escalabilidade, segurança, estado inicial, único ponto de falha, "controlador", falta de padrão, entre outros. Os resultados mostraram que os dois maiores aspectos que fazem com que SDN ainda não tenha sido emplacada são a qualidade de serviço e o ponto único de falha, que pode ser relacionado ao controlador central da rede.

Porém, a realidade vem mudando com o tempo e com o amadurecimento da ideia. Grande empresas, como a Google (GOOGLE, 2018), a Amazon (AMAZON, 2018) e a Cisco são exemplos de empresas que já estão adotando esse paradigma em suas redes e equipamentos. Isso gera uma grande importância em considerar propostas e trabalhos que envolvam as SDNs, pois essas empresas possuem alta influência no mercado e muitas vezes definem a direção do mesmo.

Enfim, o conceito de SDN já é de fato empregado e está sendo consolidado a cada dia. Porém, o conceito de QoS ainda é um dos temas que podem ser melhor discutidos na utilização dessas redes. É notório que nas redes tradicionais o conceito de QoS é bem mais maduro e consolidado, mas nas SDNs ainda é um conceito novo e muitos controladores ainda não são capazes de aplicarem as métricas e especificações por conta própria (D'SOUZA et al., 2016).

2.4 Provisão de QoS em SDN

O objetivo do nosso trabalho é trazer uma solução de aplicação de QoS em SDN para que aplicações de alto desempenho tenham seus requisitos de QoS atendidos de forma segura, rápida e de fácil implementação. Para isso, trazemos aqui nesta seção alguns trabalhos relacionados com a nossa abordagem.

Com o amadurecimento do paradigma SDN, os esforços em torno do provisiona-

mento de QoS nessas redes ganhou força, como é informado pelo trabalho de (KARAKUS; DURRESI, 2017), de tal forma que podemos relacionar algumas pesquisas com a proposta e o objetivo deste trabalho.

Uma proposta altamente dependente do contexto da aplicação, mas que se relaciona com o propósito deste trabalho, é a de (YU; WANG; HSU, 2015). No artigo, os autores propõem uma solução para aplicações de *streaming* de vídeo. A implementação sugerida por eles leva em consideração o roteamento adaptativo para melhorar a qualidade dos fluxos sobre as SDNs. Nessa abordagem, dois níveis de fluxos QoS são tratados separadamente, entre eles os pacotes da camada base e os da camada de aprimoramento dos fluxos de *bits* dos vídeos. Essa abordagem é nomeada pelos autores como ARVS.

Por outro lado, há também soluções que são amarradas a contextos/aplicações específicas. Temos como exemplo o trabalho de (DIORIO; TIMÓTEO, 2016), que apresentam e discutem o emprego de recursos para assegurar o encaminhamento de fluxos multimídias com suporte a QoS. Nessa rede, os recursos são ofertados por um *gateway* multimídia, que tem acesso à rede dos dois sistemas finais. Na abordagem, o *gateway* é capaz de identificar e classificar múltiplos fluxos multimídia. Isso pode trazer como benefício a diferenciação do tráfego quanto ao seu processamento e direcionamento.

(HUMERNBRUM; GLINKA; GORLATCH, 2014) também apresentam uma proposta para aplicação específica. Eles criam uma API *Northbound* SDN (entre controlador e aplicações da rede) para aplicações interativas *online* em tempo real que especificam seus requisitos de maneira dinâmica, de tal forma que eles possam também ser atendidos em tempo real.

Já no âmbito de redes de *data center*, (AFAQ; REHMAN; SONG, 2015) utilizam técnicas para a limitação de taxa e com isso garantem QoS aos chamados fluxo "camundongos". Fluxos como esses são curtos e sensíveis à latência, como por exemplo, VoIP. Tais fluxos concorrem com os chamados fluxos "elefantes", que se caracterizam por serem mais longos e gerados por rotinas de *backup*, por exemplo. Nessa abordagem, os autores utilizam um *framework* que se baseia em detectar uma amostragem de fluxos demorados e em seguida submetê-los a um módulo QoS, que é administrado por um controlador SDN. Por sua vez, a função do controlador é rotear os pacotes gerados para caminhos da rede

onde serão aplicadas taxas de vazão restritas.

Como vimos anteriormente, poucas propostas tratam QoS sem dependência de uma aplicação alvo ou específica. Porém, (TOMOVIC; PRASAD; RADUSINOVIC, 2014) apresentam um projeto de um ambiente de controle SDN/OpenFlow que garante largura de banda para fluxos prioritários através de limitação de taxas. Os autores propõem que o controlador SDN execute cálculos de rotas e a reserva de recursos. Nesse trabalho, a criação das filas nas interfaces dos dispositivos é realizada de maneira estática no início da execução do controlador SDN.

3 A Proposta

O objetivo deste capítulo é realizar a apresentação da proposta do trabalho, abordando a topologia utilizada e as estratégias adotadas.

A proposta da experimentação foi analisar um ambiente onde fosse possível uma avaliação prática e de fácil entendimento. Uma estrutura que tivesse poucos elementos, porém que demonstrasse bons resultados na experimentação.

A topologia de avaliação considerada neste trabalho segue um modelo hierárquico, como demonstrado na Figura 3.1. Esse padrão é bem comum nas redes existentes, pois além da facilidade de administração, proporciona uma melhor visão da rede. Tal padrão de estrutura pode ser visto, por exemplo, em uma rede de *campus* de uma universidade, onde pode existir a figura de um *switch core* (*switch* com maior poder de processamento) na função de concentrador de redes e vários outros *switches* ligados a ele, os quais podem estar distribuídos em diversos institutos, por exemplo.

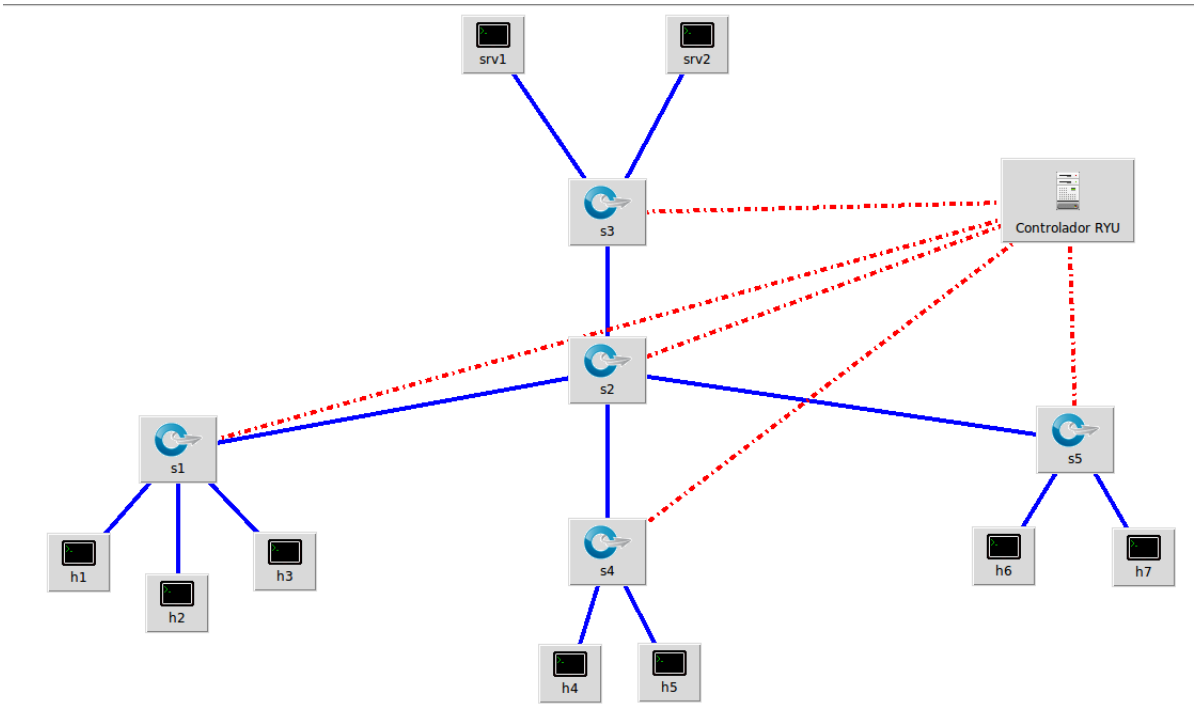


Figura 3.1: Topologia hierárquica de avaliação do trabalho

Procuramos seguir esse tipo de topologia, pois além de ser uma topologia encon-

trada em diversos cenários de rede, como já dito anteriormente, também é uma melhor maneira de demonstração dos resultados, haja vista sua melhor compreensão.

O objetivo da proposta é garantir uma largura de banda adequada entre o servidor que disponibiliza uma aplicação e algum cliente na rede que solicite serviços dessa aplicação, para tanto, aplicando essa garantia de forma dinâmica na rede.

Para exemplificar melhor o objetivo da proposta, pode-se imaginar um cenário hipotético onde uma determinada faculdade possui vídeos educativos para fornecê-los aos alunos da instituição em uma rede local. Nesse cenário, pode haver um servidor que forneça os vídeos – o qual provavelmente está centrado em algum lugar específico da universidade – e os alunos que estão espalhados por diversos institutos da universidade, os quais gostariam de assistir os vídeos disponíveis. Portanto, a ideia principal é garantir uma largura de banda apropriada entre o consumidor (no cenário hipotético, o aluno) e o servidor do serviço (no cenário hipotético, o servidor de vídeos).

Pensando assim, propomos uma distribuição de vídeo através da rede, sendo disponibilizada por um servidor que contenha a aplicação de *streaming* de vídeo. Esta simulação visa representar uma aplicação distribuída de alto desempenho, em particular, a distribuição de vídeo, sendo ofertada em uma rede local. Logo, para os testes e simulações, nossos cenários de experimentações levaram em consideração esse tipo de aplicação, onde há um serviço de vídeo ofertado por um servidor de aplicações e um cliente solicitante a esse serviço. Porém, em algumas avaliações, é acrescentado um tráfego de fundo, o qual chamamos de "tráfego *background*", cuja principal função é adicionar um tráfego extra na rede, com o propósito de gerar uma competição no meio de propagação.

3.1 Funcionamento da proposta

Com o objetivo de garantir um largura de banda adequada entre o cliente solicitante e o servidor da aplicação, foi tomado como base o funcionamento da arquitetura "Cliente/Servidor". Nesse tipo de abordagem, geralmente existe um servidor ofertando um determinado serviço e um ou vários clientes que desejam acessá-lo. Para que algum cliente tenha acesso a esse serviço, o servidor da aplicação fica aguardando requisições em uma determinada porta de serviço e assim que o pedido chega, logo que possível, ele é atendido

e tratado.

Além de trabalharmos com esta arquitetura, o nosso trabalho também foi baseado em SDN. Com isso, a estrutura de rede é acrescida de um elemento controlador central SDN. Seu papel fundamental é dar suporte tanto aos comutadores de rede quanto aos administradores do ambiente. Além de estar centralizando a programação da rede, como é feito nas maiorias dos controladores SDN, o conceito nos fornece uma visão global da rede, o que na maioria dos casos ajuda a identificar possíveis problemas, como por exemplo, o "gargalo" na rede.

Como o modelo SDN abstrai o plano de controle dos comutadores de rede, os mesmos precisam, de alguma forma, conhecerem as ações a serem tomadas em relação aos fluxos de rede que passam por eles. Toda vez que um comutador recebe um fluxo de rede, o mesmo verifica em suas regras se existe um padrão que combine ou se assemelhe àquele fluxo. Se a resposta for positiva, ou seja, há uma regra relacionada àquele fluxo, o mesmo segue com a ação determinada. Caso a resposta seja negativa, o mesmo encaminha o pacote ao controlador central da rede, que verifica a ação a ser tomada para aquele fluxo e devolve como resposta a regra a ser utilizada em relação ao fluxo.

Tome como exemplo a Figura 3.2. Nela é possível ver a abstração de um *switch OpenFlow* que faz a ligação entre dois *hosts*. Note que o *switch* é monitorado e controlado por um controlador SDN chamado c0. O diagrama da Figura 3.3 demonstra um pedido de conexão entre os *hosts* h1 e h3.

Cada controlador SDN, disponível para uso, possui um conjunto de diversas APIs prontas que já abstraem grande parte da programação, ou seja, são disponibilizados módulos já prontos para uso, os quais podem ou não serem modificados, dependendo de sua aplicação.

No trabalho foi utilizada uma API que se assemelha a um *switch* comum com suporte ao protocolo *OpenFlow 1.3*. Sua necessidade foi atender a demanda dos comutadores com regras mais comuns que são utilizadas no cotidiano da rede. Um exemplo de utilização dessa aplicação é a descoberta de caminhos quando um teste de *ping* é executado entre máquinas da rede.

Assim, temos que a rede vai sendo descoberta ao longo do tempo, tanto pelo

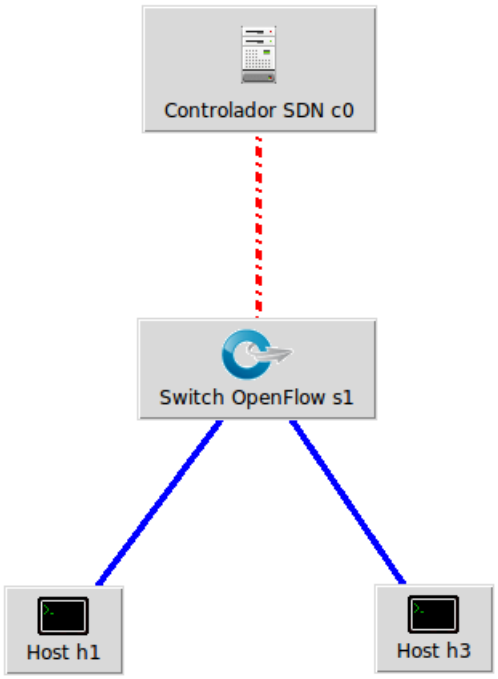


Figura 3.2: Modelo simples de uma rede SDN.

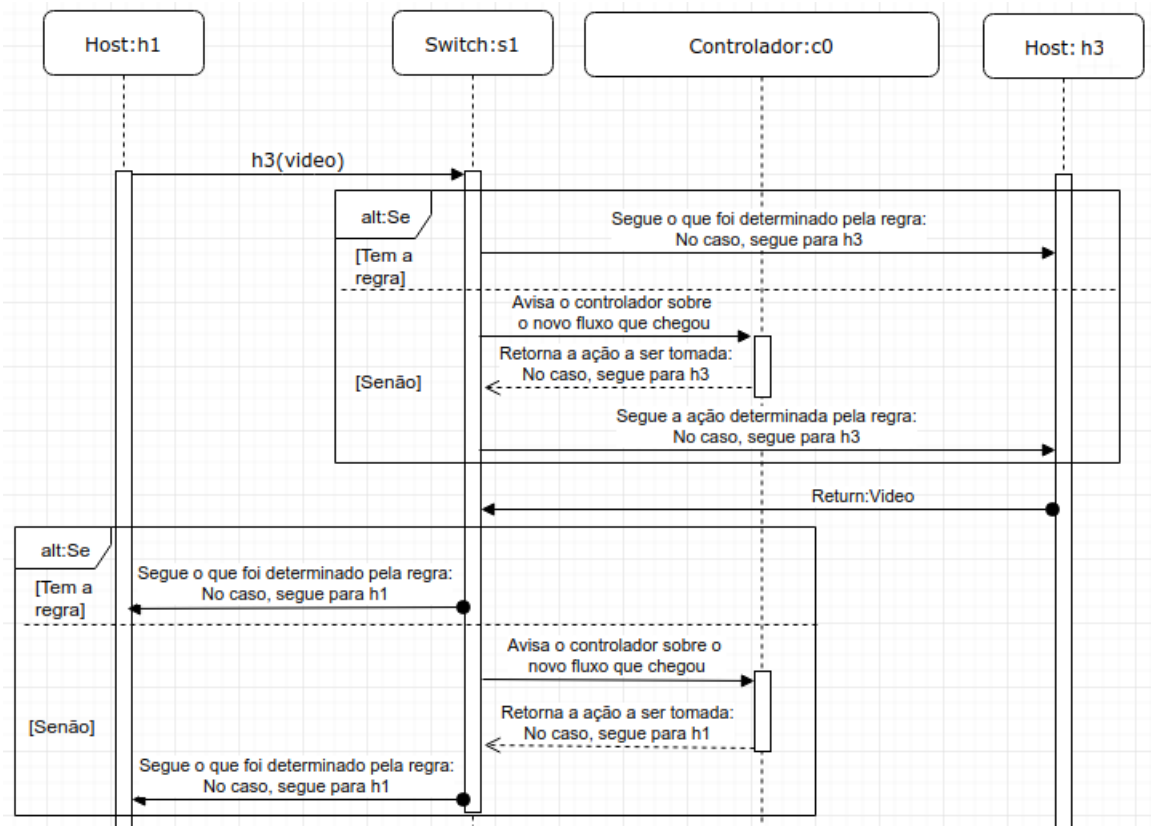


Figura 3.3: Diagrama de sequência de um pedido de aplicação entre h1 e h3

controlador central da rede como pelos seus componentes programáveis. Situação essa muito comum nas redes, onde temos um aprendizado ao longo do tempo e no momento

da inserção de novos elementos na estrutura da rede.

Além disso, temos um outro módulo chamado "MeuAPP", o qual foi criado para atender as demandas das funções de provisão de QoS na rede. Esse módulo também identifica os caminhos entre os possíveis clientes da rede e os servidores de aplicações. Para isso é montando um grafo direcional, onde seus vértices são os *hosts* e os comutadores de rede, e suas arestas são as ligações entre eles: (*host*,comutador) e (comutador,comutador). Logo, com o uso desse mecanismo, o mesmo é capaz de verificar as portas que receberão as filas de QoS para que seja garantida a largura de banda especificada.

Uma pergunta que ainda não foi respondida é a seguinte: como o controlador saberá qual a reserva de largura de banda que cada aplicação terá? Essa é uma demanda que cabe ao administrador da rede saber e fazer, pois isso é um processo manual e que deverá ser feito logo após o início da operação da rede. O administrador de redes deverá indicar a quantidade de reserva de banda que cada aplicação deva ter. Para isso, o mesmo deverá se basear, ou mesmo supor, a quantidade específica para cada uma, variando entre a mínima e a máxima.

Após essas breves explicações sobre o conjunto de uma SDN e utilização de alguns módulos, vamos determinar o funcionamento da nossa abordagem de reserva de banda entre cliente e o servidor. No trabalho, a requisição do serviço parte do cliente solicitante, passa por toda a rede e chega ao servidor que o disponibiliza. Nesse processo, se o caminho de rede entre as duas entidades em questão ainda não foi aprendido pelo módulo do controlador, o mesmo o faz neste momento.

Assim que o pedido é recebido pelo servidor, o mesmo guarda as informações solicitadas pelo cliente e avisa ao controlador central da rede. Ao receber o pedido de aviso, vindo do servidor de aplicação, o controlador verifica as configurações desejadas e, se possível, faz a reserva de largura de banda no caminho da rede entre o servidor da aplicação e o cliente solicitante. A representação desse processo é ilustrada na Figura 3.4.

Toda a comunicação inicial entre o servidor e o cliente são feitas via *socket* TCP. Ao final do processo, estando tudo certo com a provisão da QoS (reserva de largura de banda), a aplicação é iniciada no servidor, ofertando o serviço para o cliente. Nesse ponto, a aplicação é iniciada nos dois elementos, tanto no servidor quanto no cliente. Através do

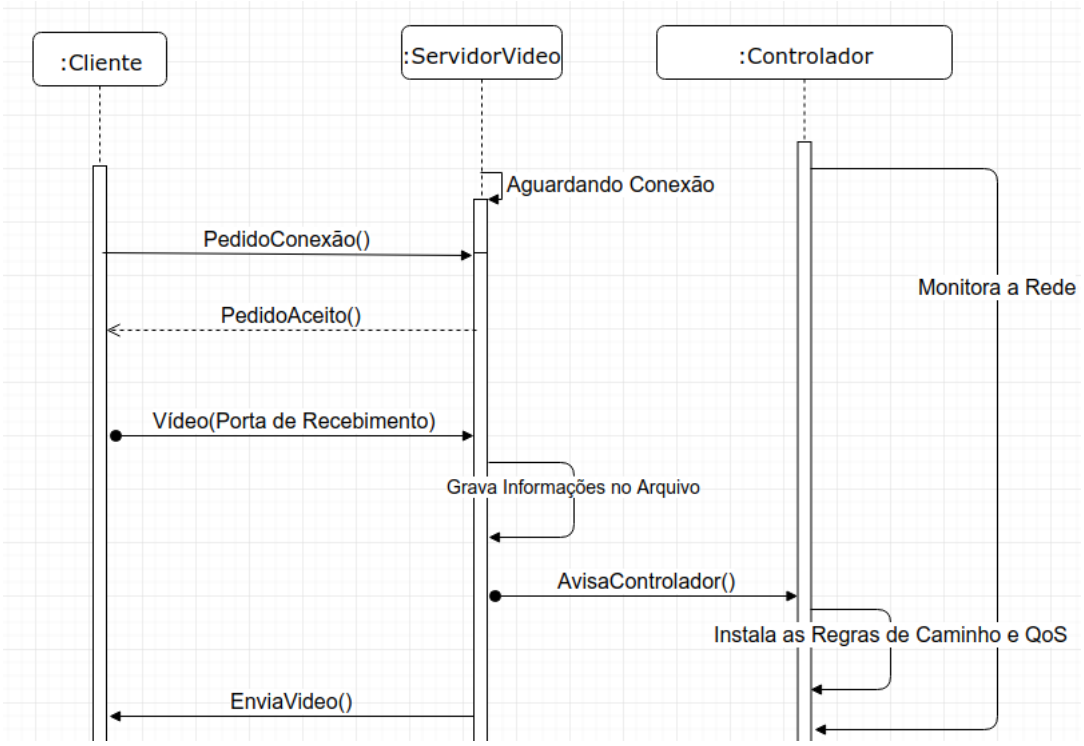


Figura 3.4: Diagrama de sequência do pedido do vídeo

protocolo UDP, um *streaming* de vídeo gerado pelo servidor é enviado ao cliente através da rede.

3.2 Topologia utilizada

Como visto na seção 3.1, o trabalho seguiu uma estrutura de rede no padrão hierárquico, a qual é composta por cinco *switches*, nove *hosts* e, devido a utilização de uma SDN, a figura do controlador central da rede. A representação dessa topologia é apresentada na Figura 3.1.

Nessa topologia é possível perceber que o *switch* s2 possui a função de concentrador de redes e os demais *switches* possuem a função de interligar os *hosts*. Nela é possível identificar que o servidor de vídeo srv1 e um segundo servidor srv2 (Web, por exemplo) encontram-se conectados ao *switch* s3. Por sua vez, os *hosts* seguem a seguinte distribuição: h1, h2 e h3 foram fixados no *switch* s1; os *hosts* h4 e h5 no *switch* s4; e os *hosts* h6 e h7 fixados no *switch* s5. Na topologia é possível perceber também que os *switches* s1, s3, s4 e s5 foram ligados diretamente ao *switch* S2.

Na estrutura apresentada, é notável que não existe redundância de *links* entre as

redes disponíveis. Essa redundância pode ser muito comum e é, muitas vezes, vista em cenários de utilização real. Geralmente a redundância é fornecida como uma alternativa de caminhos distintos entre redes ou simplesmente como um "*backup*" de links.

Porém, tal fato foi desconsiderado no trabalho, pois em nossos experimentos isso somente causaria o aumento da complexidade, tanto no entendimento quanto nos testes. Além disso, não auxiliaria nos resultados dos testes, tendo em vista que nossa proposta é assegurar uma largura de banda adequada entre cliente e servidor, bastando para isso que se tenha um *link* de comunicação entre eles.

Com o intuito de reduzir a complexidade e melhorar a apresentação dos resultados, foi destacada uma pequena parte da arquitetura para a provisão da qualidade de serviço. A topologia deste trabalho segue conforme demonstrado na Figura 3.1, porém, para as nossas análises e a aplicação da qualidade de serviço, foram destacados os *switches* s1, s2 e s3, os quais representam um segmento da infraestrutura de rede. Chamaremos essa pequena fração de "topologia fracionada" e, a partir deste ponto, nosso enfoque será dado nessa fração. A representação dessa topologia fracionada pode ser vista na Figura 3.5.

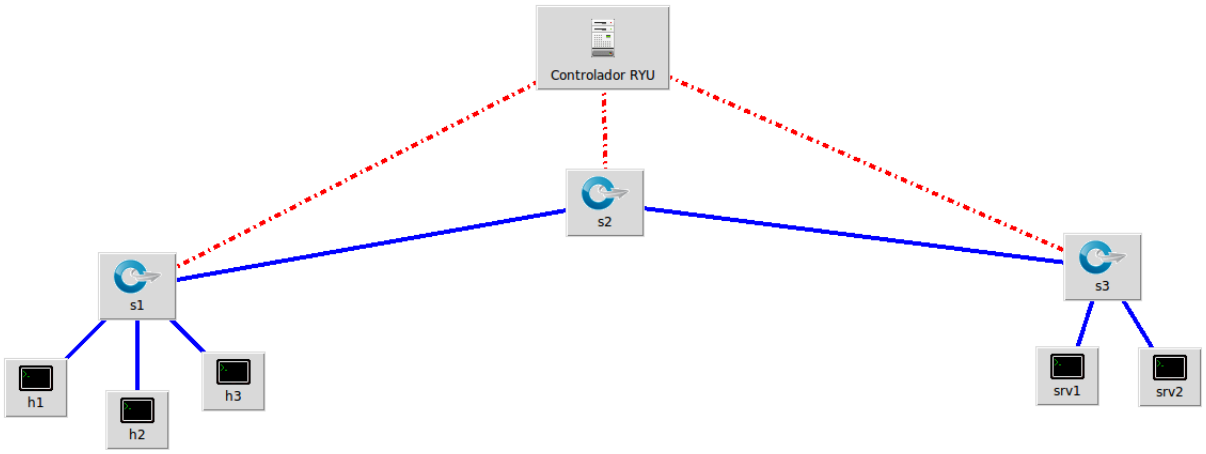


Figura 3.5: Topologia Aplicação da QoS

A figura do *switch* concentrador s2 permanece, juntamente com o *switch* s3 – o qual possui o servidor de vídeo e o servidor web – e também o *switch* s1 – que possui os *hosts* h1, h2 e h3.

Cada *host* apresentado na Figura 3.5 teve uma função específica no trabalho. O

host h1 fez o papel de cliente solicitante do vídeo. Por outro lado, o *host* srv1 fez o papel de servidor e forneceu o vídeo solicitado por h1. Os *hosts* h2 e h3 tiveram a função de gerar a carga em *background* na rede, juntamente com o *host* srv2.

3.3 Abordagens propostas para comparação de resultados

No trabalho foram analisados três cenários de avaliação, implementados por meio de três diferentes simulações. Todas as simulações buscaram verificar o comportamento do tráfego de rede de uma aplicação de vídeo (representando aqui uma aplicação distribuída de alto desempenho). Para tanto, o tráfego dessa aplicação foi avaliado sob três condições: (a) um ambiente sem tráfego *background* de rede; (b) um ambiente com apenas um tráfego *background* de rede; e (c) um ambiente com dois tráfegos *background* de rede. O tráfego *background* é representado neste trabalho como uma carga de rede que simboliza tráfegos de rede gerados por diversas outras aplicações. Essa carga de rede tem, por objetivo, gerar tráfego que disputará o meio de transmissão com a aplicação que deverá ser priorizada.

Assim, em cada cenário de avaliação foram executadas três simulações, sendo elas divididas em: (i) análise da qualidade da aplicação em um ambiente onde nenhum tráfego é priorizado, ou seja, não há aplicação de QoS; (ii) análise da qualidade da aplicação em um ambiente onde a aplicação da QoS é dada ao longo de todo o tempo do experimento; e (iii) análise da qualidade da aplicação em um ambiente onde a aplicação da qualidade de serviço é dada de forma dinâmica, ou seja, a aplicação requisita a largura de banda desejada e o controlador faz essa provisão somente antes do início do envio do vídeo.

No primeiro cenário, isto é, no ambiente de teste onde não há o tratamento do vídeo com a qualidade de serviço, ao final do processo de solicitação o vídeo já era enviado através de *streaming*. Seu tráfego disputava largura de banda com os outros tráfegos da rede, como dito anteriormente.

No segundo cenário, onde a aplicação da qualidade de serviço foi dada no início da simulação de cada experimento, o tráfego de resposta de vídeo recebia seu devido tratamento e era enviado para o solicitante, porém a reserva de banda da aplicação foi

garantida mesmo ela não estando em uso. Para exemplificar melhor, suponha que a aplicação de vídeo necessite de uma reserva de banda de sete (7) Megabits por segundo (Mbps). Essa reserva de banda ficaria alocada no comutador de rede mesmo a aplicação não a utilizando.

Por outro lado, no terceiro cenário, onde a provisão da qualidade de serviço foi dada de forma dinâmica, a reserva de banda para a aplicação foi dada no momento da sua requisição, ou seja, após o pedido de vídeo, a reserva de banda é solicitada e atendida. A aplicação servidora executa o *streaming* e, ao término do envio, a reserva da largura de banda é desfeita, finalizando assim o processo.

Os detalhes sobre a implementação e os testes de avaliação são fornecidos no capítulo 5, onde também são abordados os resultados obtidos nos experimentos e as discussões sobre os mesmos. No capítulo em questão, será possível notar como foi fornecida a reserva de largura de banda à aplicação – tanto no experimento dois (ii), quanto no experimento três (iii), que foram os experimentos onde a largura de banda para a aplicação foi exigida.

3.4 Considerações Finais

Inicialmente, nossa proposta pode parecer um pouco complexa. Porém, no Capítulo 5, diante dos gráficos apresentados e os resultados obtidos, poderemos ver que existem melhores maneiras de se aplicar QoS na rede, de uma forma mais inteligente e amigável.

Com isso, a contribuição da proposta fica mais clara, tendo em vista as dificuldades encontradas em especificar os requisitos de QoS onde, muitas vezes, o cenário pode não ser favorável a isso.

4 Ferramentas Utilizadas

O propósito deste capítulo é ambientar e familiarizar sobre as ferramentas utilizadas no processo dos testes e seus conceitos associados, detalhando o seu funcionamento e trazendo sua importância para o trabalho.

4.1 Mininet

A falta de recursos físicos para implementação de grandes infraestruturas de rede muitas vezes dificulta o trabalho de pesquisadores que necessitam de grandes ambientes para experimentações de seus projetos.

Com o intuito de auxiliar essa falta de recursos, surgem diversas soluções, e dentre elas, algumas *Open Source*. Essas soluções visam auxiliar a emulação de meios físicos de rede em um determinado computador. No nosso caso, todo o desenvolvimento do trabalho foi feito com o emulador de redes Mininet (TEAM, 2018), que é uma solução *Open Source*.

O aplicativo Mininet é um criador de redes virtuais realista, executando o *kernel* real, o *switch* e o código do aplicativo em uma única máquina (VM, nuvem ou nativa).

Segundo informações que constam em seu site oficial, o Mininet cria uma maneira fácil de interagir com a rede, sendo útil para o desenvolvimento, ensino e pesquisa. Ele também é um *software Open Source* licenciado pela *BSD Open Source*.

Os autores (LANTZ; HELLER; MCKEOWN, 2010) descrevem em sua obra alguns importantes pontos a serem destacados no processo de criação do Mininet. Dentre os pontos frisados, podemos salientar:

- **Flexibilidade:** Usando uma linguagem familiar e um sistema operacional, novas topologias e novas funcionalidades podem ser definidas por *software*.
- **Interatividade:** O gerenciamento da rede e sua execução devem ocorrer em tempo real, simulando o que acontece na vida real com meios físicos.
- **Escalabilidade:** A dimensão do ambiente de prototipagem deve poder suportar

centenas de nós, sejam eles *hosts* ou comutadores de rede, em apenas um computador de simulação.

- **Realismo:** O comportamento das redes emuladas deve apresentar um alto grau de confiança da realidade. Por exemplo, as aplicações e pilhas de protocolos devem ser suportadas e utilizáveis sem modificações.

Em uma outra abordagem, é possível destacar também a importância do Mininet ao permitir uma maior pesquisa na redes definidas por *software* e no protocolo *OpenFlow* (protocolo utilizado na comunicação entre os comutadores da rede e o controlador central da rede) (KAUR; SINGH; GHUMMAN, 2014). Na Figura 4.1 é possível ver uma estrutura de rede real na qual os elementos da rede são exibidos pela interface gráfica que o próprio *software* Mininet dispõe. Nela, são mostrados dois comutadores de rede (dois *switches* comuns) e quatro *hosts*.

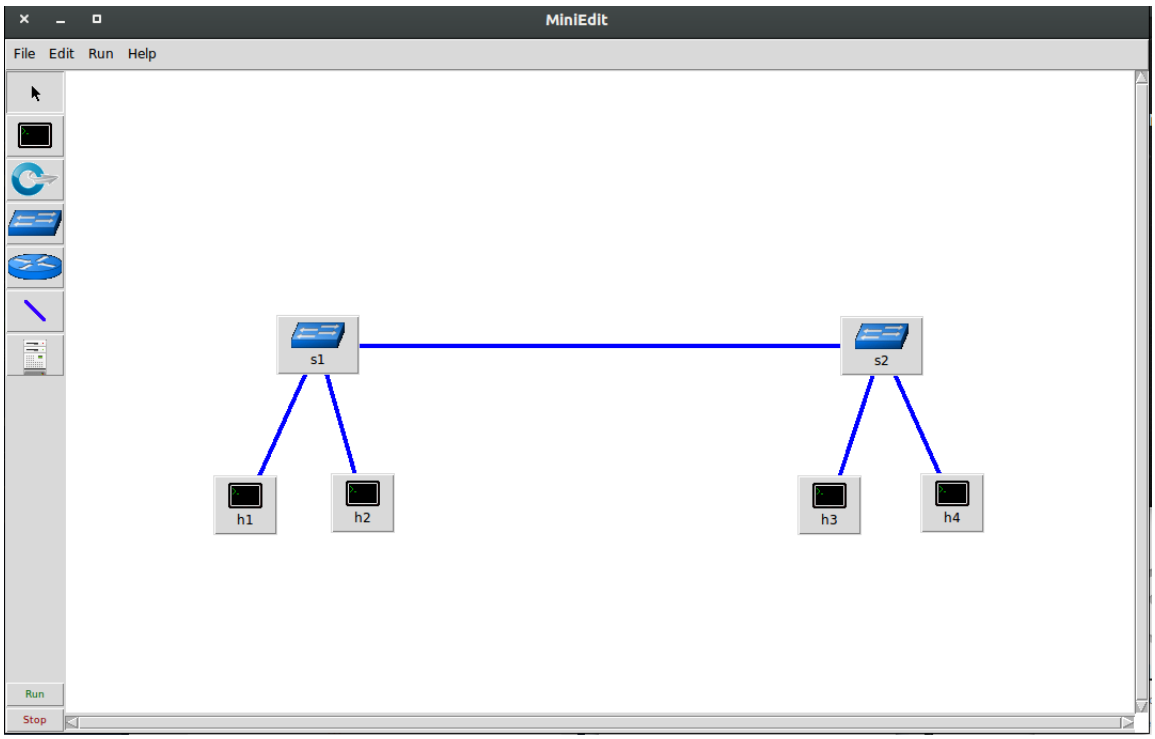


Figura 4.1: Protótipo de uma rede no Mininet

A interface gráfica do Mininet é um auxílio visual para montagem da topologia, mas ela não é a única forma de construí-la. É possível criar estruturas de redes bastante complexas através de *scripts*, gerados automaticamente ou manualmente, sendo uma prática muito comum na criação dessas topologias. Além disso, o Mininet possui

uma documentação de suporte que encontra-se disponível em seu site oficial.

O Mininet disponibiliza *switches* emulados. Essa emulação é possível graças ao *software Open vSwitch*, que tem como um de seus propósitos criar uma abstração de rede, fazendo uma emulação de recursos físicos. Juntamente com o suporte do *Open vSwitch*, o Mininet apresenta uma outra grande vantagem, que é a emulação de *switches* que suportem o protocolo *OpenFlow*. Tanto o *software Open vSwitch* quanto o protocolo *OpenFlow* são abordados nas seções seguintes desta monografia.

Neste trabalho, a utilização do Mininet é justificada pela sua capacidade de emulação de um cenário real de utilização de redes em uma única máquina física. Tal fato é viável pois, apesar de simulado, o ambiente fornecido pelo *software* nos aproxima do ambiente real, além de extrair a complexidade de ter que possuir vários equipamentos reais para as simulações exigidas no trabalho.

4.2 *Open vSwitch*

Segundo consta na definição do seu próprio site – que tem a colaboração da *Linux Foundation* – o *Open vSwitch* (VSWITCH, 2018) é considerado um *switch* virtual multicamadas licenciado sob a licença Apache 2.0 de código aberto. Foi desenvolvido para permitir automação na rede e possuir uma extensão programática.

Possui suporte a protocolos de gerenciamento padrão, como o *NetFlow*, que é considerado um coletor de características e informações sobre o tráfego de redes IP. O *Open vSwitch* também foi projetado para dar suporte à distribuição em vários servidores físicos, que é parecido ao *vswitch*, que tem sua distribuição pela *vNetwork* da VMware ou ao Nexus 1000V da Cisco. Logo abaixo são apresentados os recursos suportados pelo *Open vSwitch*:

- Visibilidade na comunicação entre Máquinas Virtuais (VMs) via os protocolos NetFlow, sFlow(R), IPFIX, SPAN, RSPAN e Gre-tunneled.
- LACP (IEEE 802.1AX-2008).
- Modelo 802.1Q VLAN padrão com entroncamento.

- *Multicast snooping*.
- IETF *Auto-Attach* SPBM e suporte a requerimentos rudimentares LLDP.
- Monitoramento de link BFD e 802.1ag.
- STP (IEEE 802.1D-1998) e RSTP (IEEE 802.1D-2004).
- Controle de QoS refinado.
- Suporte para qdisc do HFSC.
- Monitoramento de tráfego de interface por *Virtual Machine* (VM).
- Ligação de NIC com balanceamento de carga de origem MAC, *backup* ativo e *hash* L4.
- Suporte ao protocolo *OpenFlow* (incluindo muitas extensões para virtualização).
- Suporte IPv6.
- Vários protocolos de encapsulamento (GRE, VXLAN, STT e Geneve, com suporte a IPsec).
- Protocolo de configuração remota com ligações C e Python.
- Opções de mecanismo de encaminhamento de *kernel* e espaço do usuário.
- *Pipeline* de encaminhamento de várias tabelas com mecanismo de armazenamento em *cache* de fluxo.
- Encaminhamento da abstração da camada para facilitar a migração para novas plataformas de *software* e *hardware*.

Este *software* teve por objetivo no trabalho dar suporte ao aplicativo Mininet na emulação do ambiente virtual da rede e toda sua estrutura.

4.3 Protocolo *OpenFlow*

A comercialização da Internet e as grandes "caixas pretas"que se tornaram os equipamentos da rede fizeram surgir a necessidade da separação do plano de controle e do plano de dados dos equipamentos de rede, dando surgimento ao protocolo *OpenFlow* (ROTHENBERG et al., 2010).

Tal protocolo foi proposto pela Universidade de Stanford visando uma melhoria ao surgimento de novas tecnologias, como as redes definidas por *software*. Além disso, podemos considerar este protocolo como uma tecnologia nova e de rápida evolução, que muitas vezes pode ajudar na habilitação da implementação do paradigma de Redes Definidas por *Software* (FERNANDES; ROTHENBERG, 2014).

Logo depois de sua criação, o protocolo iniciou sua evolução. À medida que novas necessidades foram surgindo, novas versões foram sendo implementadas. A Figura 4.2 resume o desenvolvimento do protocolo ao longo de suas versões.

Evolução do Protocolo *OpenFlow*	
OpenFlow 1.0	Protocolo mais conhecido
OpenFlow 1.1	❖ Múltiplas tabelas e grupos de tabelas ❖ Algumas combinações tipos fluxo e ações
OpenFlow 1.2	❖ Suporte a IPv6 ❖ Possibilidade de conectar um comutador a múltiplos controladores ❖ Recuperação rápida de falhas ❖ Balanceamento de carga.
OpenFlow 1.3	❖ Possibilidade de controlar a taxa de pacotes por fluxo ❖ Métricas ❖ Tabela de características
OpenFlow 1.4	❖ Sincronização da tabela de fluxo ❖ Monitoramento de fluxo
OpenFlow 1.5	❖ Combinações e ações mais refinadas ❖ Tabelas de saída ❖ Melhorias da tabela de grupo / métrica

Figura 4.2: Tabela de evolução do *OpenFlow* (REIZNAULTT, 2018).

O protocolo *OpenFlow* determina, de forma padrão, as ações de encaminhamento

de pacotes que os componentes de rede (comutadores e roteadores) devem seguir. Tais ações são especificadas por um agente externo, ou seja, o controlador da rede, o qual deverá especificar claramente ao dispositivo a medida a ser tomada com determinado fluxo de pacotes, ou até mesmo a ação a se tomar em relação a um simples pacote que passa pela rede.

Em uma rede que trabalha com o protocolo *OpenFlow*, ou seja, uma rede programável que utiliza o protocolo, é necessário que os componentes de rede possuam tabelas que suportem receber regras e ações, vindas através de um canal de comunicação seguro entre o controlador da rede e o comutador em questão.

Cada pacote que passa pela rede possui alguns cabeçalhos específicos, como o endereço de origem e o endereço de destino. Com base nessas informações contidas nos cabeçalhos dos pacotes, o controlador da rede, através do protocolo *OpenFlow*, determina a regra a ser enviada para os componentes da rede.

Na instalação de uma regra em um comutador da rede, é possível determinar os campos a serem visualizados nos cabeçalhos do pacote e a ação a ser tomada em relação a ele ou ao fluxo que possui as mesmas características. A Figura 4.3 mostra como é uma entrada de um fluxo em uma tabela de fluxos. Já a Figura 4.4 mostra um conjunto de regras de fluxos instaladas em um *switch* emulado pela ferramenta *Open vSwitch*.

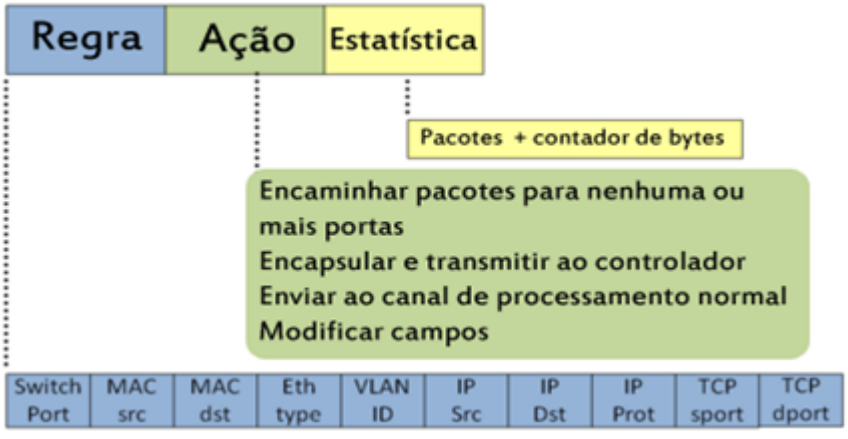


Figura 4.3: Entrada de fluxo em uma tabela de fluxo

Neste trabalho, a instalação das regras nas tabelas de fluxos dos comutadores de rede foram baseadas no protocolo *OpenFlow*, cujo controlador SDN RYU possui suporte.

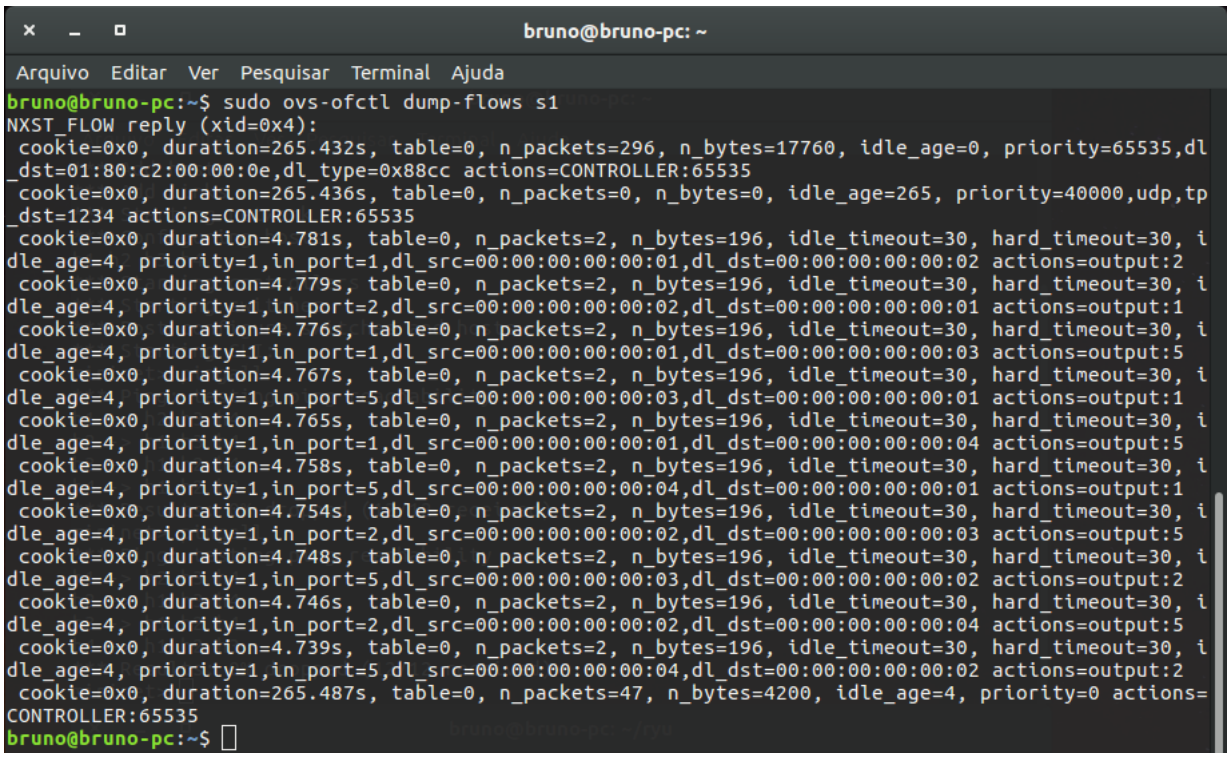


Figura 4.4: Regras de fluxos instaladas em um *switch*

4.4 Controlador SDN RYU

Para o controle de uma rede definida por *software*, com componentes que possuem o protocolo OpenFlow, é necessário um controlador SDN. O objetivo do controlador é fazer toda a orquestração da rede que contenha tais elementos.

Existem no mercado diversas soluções para atender esse quesito. Podemos encontrar controladores que suportam diferentes linguagens de programação. Como escolhemos a linguagem de programação Python para a implementação deste trabalho, tivemos como opções de controladores desenvolvidos para essa linguagem os controladores POX (muito utilizado para o aprendizado em SDN) e o controlador RYU (COMMUNITY, 2017), que é um controlador de nível comercial e que suporta diversas versões do protocolo *OpenFlow*.

Na descrição em seu site oficial, o RYU é definido como uma estrutura de rede definida por *software* baseado em componentes, que fornece uma API bem definida. Isso facilita aos desenvolvedores a criação de novos aplicativos de gerenciamento e controle da rede. O controlador possui suporte a vários protocolos, como *OpenFlow*, *Netconf*, *OF-config*, dentre outros. Além de suportar o protocolo *OpenFlow* na sua versão 1.0, o controlador RYU suporta as versões 1.2, 1.3, 1.4, 1.5, e também a versão Nicira. É

importante ressaltar que todo o código do RYU está disponível gratuitamente sob a licença Apache 2.0.

Um controlador SDN possui várias funções que podem ser acessadas através de suas APIs, as quais fornecem um vasto conjunto de funções úteis que auxiliam na programação de um componente. Na utilização desse controlador, são destacadas duas funções que são de grande importância no trabalho, que são as funções de envio de regras através de um evento chamado "*Packet-IN*" e o envio de regras através de um evento de conexão chamado "*Handle Connect*".

O evento "*Packet-IN*"é um evento gerado pelo comutador da rede que segue os padrões do protocolo *OpenFlow*. Tal evento é disparado quando uma regra é definida para que o evento "*Packet-IN*"ocorra, ou quando não há nenhuma regra instalada no comutador que combine com a descrição do pacote que chegou até ele. Todo ou parte do pacote é encapsulado e enviado ao RYU, para que o mesmo tenha o seu tratamento. Logo após essa verificação, são montadas uma ou mais regras pelo controlador RYU, seguindo os padrões do protocolo *Openflow*. Ao final, a regra ou as regras são enviadas para o comutador da rede que gerou o evento, para que o mesmo saiba a ação a ser tomada com o pacote ou com o fluxo de pacotes que seguem o mesmo padrão.

Trabalhando de outra forma, o evento "*Handle Connect*"é executado toda vez que um comutador entra na rede. O RYU recebe uma notificação de evento de conexão através do protocolo *OpenFlow* e reage enviando as ações especificadas e predeterminadas para esse evento. Dessa forma, as ações são instaladas de forma proativa no comutador de rede, isto é, as ações são enviadas ao comutador logo que o controlador RYU o identifica na rede.

De certa forma, podemos diferenciar os eventos "*Packet-IN*"e "*Handle Connect*", dizendo que um age de maneira reativa e outro de forma proativa. Para melhor exemplificar, os eventos *Packet-IN* são gerados de acordo com a necessidade, e os eventos do tipo *Handle Connect* são gerados no início de uma comunicação entre um comutador de rede e o controlador RYU. De fato, o entendimento desses dois eventos traz uma melhor clareza na apresentação dos resultados deste trabalho.

A escolha da utilização do controlador RYU para a implementação do trabalho

teve um peso maior, pois a escolha de uma linguagem de programação de fácil utilização e com um alto grau de aprendizado – neste caso, a linguagem de programação Python – viabiliza uma proposta que, ao final do processo, possui potencial comercial para uso.

4.5 Geração de carga de trabalho

Como já visto anteriormente, a proposta do trabalho é oferecer uma reserva de largura de banda entre o cliente e o servidor da aplicação. Para simulação desse cenário, duas ferramentas foram de grande importância na geração da carga de trabalho. São elas o VLC Media Player e o iPerf, as quais são detalhadas nas subseções a seguir.

4.5.1 VLC Media Player

Neste trabalho foi necessário fazer uma escolha de uma aplicação distribuída de alto desempenho para simulação e testes da proposta. Para isso, a escolha foi um servidor de *streaming* que simulasse o envio de um vídeo a partir de um servidor de vídeos para um computador solicitante do serviço, que no caso é o próprio vídeo.

Nesse cenário, foi utilizado um *software* de visualização e envio de vídeos através da rede, o VLC Media Player, disponibilizado pela VideoLAN Organization (ORGANIZATION, 2018). A VideoLAN Organization é um projeto e também uma organização sem fins lucrativos.

O VLC media player é um reprodutor multimídia livre que possui suporte a diversas plataformas. Possui um arcabouço que reproduz a maioria dos arquivos multimídia, além de possuir suporte a vários protocolos de fluxos de rede.

4.5.2 iPerf

O iPerf (IPERF.FR, 2018) é um *software* livre que foi desenvolvido pelo *National Laboratory for Applied Network Research/Distributed Applications Support Team* (NLANDR/-DAST). Seu intuito foi criar uma alternativa para medição do rendimento da banda de redes, utilizando, para isso, os protocolos TCP e UDP.

Com esta ferramenta, é possível realizar a verificação de várias métricas de rede,

como a capacidade máxima fim-a-fim a nível de transporte, o *jitter* e também a perda de pacotes. Além disso, o iPerf tem seu funcionamento baseado no modelo cliente/servidor, onde o servidor deve atender às solicitações de testes vindos através das requisições dos clientes (DINIZ; JUNIOR, 2014).

Assim como em diversos outros experimentos que tratam cenários de medições em redes, este trabalho também utiliza esta ferramenta como uma forma de auxílio. A ideia é que todo tráfego fora da aplicação distribuída de alto desempenho seja gerado pela ferramenta iPerf.

Em quase todos os testes que foram realizados houve um tráfego de fundo na rede, o que tornou os testes mais fiéis à realidade. O iPerf foi o responsável por simular o tráfego gerado pelo servidor Web na rede. Além disso, seu tráfego foi analisado como uma carga da rede, simulando a utilização de vários usuários acessando o servidor em questão.

Os detalhes de utilização da ferramenta nos testes realizados neste trabalho são dados no capítulo 5 desta monografia.

5 Ambiente e Resultados Experimentais

Neste capítulo são apresentados os resultados obtidos com os experimentos propostos. Os experimentos foram descritos de uma forma sucinta no Capítulo 3 desta monografia. O objetivo é deixar claro como foram desenvolvidos os testes, a maneira como foi utilizada a aplicação da qualidade de serviço e apresentar os dados obtidos.

Para o envio do vídeo fornecido pelo servidor srv1 e o seu recebimento por h1 (conforme abordado na Seção 3.2 desta monografia), foi utilizada a ferramenta de *player* de vídeo VLC Media Player. Essa ferramenta tem a capacidade de fazer *streaming* de vídeo, utilizando para isso diversos tipos de protocolos, como TCP e UDP. Neste trabalho, o protocolo utilizado foi o UDP, pois é um protocolo capaz de fazer o envio de pacotes na rede sem se preocupar com a confirmação do recebimento por parte do receptor. Isso, de fato, teve grande importância nos testes, pois não houve *overhead* causado pelas confirmações de recebimento, característico do protocolo TCP. Ressalta-se também que a codificação do vídeo na hora do envio é feita pelo codificador de vídeo H.264.

Por outro lado, para geração de carga *background* na rede, foi utilizada a ferramenta iPerf. Essa ferramenta permite escolher entre dois protocolos para seus testes de rede, sendo eles o protocolo TCP e o protocolo UDP. Devido ao *streaming* de vídeo ser transportado pelo protocolo UDP, as gerações de cargas de trabalho com a ferramenta iPerf também foram executadas sobre esse protocolo. A escolha pelo UDP segue o mesmo motivo apresentado para o envio do vídeo, ou seja, retirar o *overhead* ocasionado pelas confirmações de entrega. Outro fator também importante para a escolha do UDP com carga de trabalho é que, ao usar o iPerf com esse protocolo, é possível fazer a escolha dos tamanhos dos pacotes a serem enviados na rede. Em todas as simulações que tiveram tráfego *background* na rede, as saídas de geração de cargas através do iPerf foram definidas em 20 Mbps, ou seja, cada tráfego gerado pela ferramenta tinha um tamanho constante de 20 Mbps, ao longo de toda a execução das simulações.

Outra informação muito importante que merece destaque neste momento é a definição das filas de QoS por parte do administrador de rede. Como já mencionado

anteriormente, cabe ao administrador do ambiente criar as filas de QoS que deverão ser utilizadas na rede ao longo dos trabalhos.

Como nos cenários 2 e 3 há provisão de QoS, foi necessário utilizar um módulo para criação das filas necessárias aos testes. Para isso, foi criado um módulo chamado "Admin". Sua função é fornecer uma interface ao administrador de rede para que o mesmo possa criar as filas de QoS que deverão ser utilizadas pelo controlador RYU no momento necessário. Esse módulo "Admin"deverá ser executado sempre no início do processo, logo após a criação da topologia, levando em consideração a utilização do *software* Mininet. Tal requisito é relacionado a uma característica do *Open vSwitch*, que requer que a aplicação das filas de QoS nas portas específicas já estejam definidas previamente em seu banco de dados interno ao *software*.

Por padrão, quando utilizamos o *software Open vSwitch* na emulação de redes juntamente com o Mininet, é necessária a criação das filas de prioridades que cada aplicação terá. Além disso, também se torna necessária a criação de uma fila padrão, a qual será responsável por receber todo o tráfego que não seja o priorizado pelas filas, ou seja, o tráfego que não possuir priorização é enfileirado imediatamente nessa fila padrão. Neste trabalho, damos o nome de "fila de melhor esforço" a essa fila padrão.

Na criação das filas de QoS, deve ser observada a largura de banda exigida pela aplicação. Logicamente, o administrador de rede poderá fornecê-la tanto para mais quanto para menos. Porém, deverá ser levado em consideração a qualidade da entrega do serviço e, juntamente com isso, o bom senso por parte do administrador.

Como nosso trabalho visa verificar a classificação do tráfego, comparando as entradas e as saídas das aplicações, foram tomados como base para a criação da fila de QoS específica para o vídeo os dados especificados na Tabela 5.1 (CONFIGURAÇÕES. . . , 2018).

Nos testes de envio do vídeo através da rede, foi utilizado um vídeo com resolução de 1080p. O vídeo em questão foi o trailer do filme "*First Man*"[1], o qual possui um tempo de duração de 140 segundos. De posse do vídeo e tomando como base a Tabela 5.1, nós resolvemos criar uma fila de QoS que garantiria uma largura mínima de banda de

[1]http://www.hd-trailers.net/movie/first-man/. Acessado em: 09 nov. 2018

Tabela 5.1: Intervalo de taxa de bits das resoluções de vídeo (CONFIGURAÇÕES..., 2018).

Resolução	**Intervalo da taxa de bits do vídeo (Kbps)**
240p	[300 a 700]
360p	[400 a 1000]
480p	[500 a 2000]
720p	[1500 a 4000]
1080p	[3000 a 6000]

12000 Kbps e com máximo de 24000 Kbps. Tais valores foram considerados devido aos altos picos de envio de bits que são gerados por esse vídeo, e também para que isso fosse melhor demonstrado nos gráficos que são apresentados nas avaliações dos cenários. Portanto, para a taxa mínima da fila de QoS, definimos o dobro do ideal de envio de um vídeo com resolução de 1080p. Por sua vez, estabelecemos para a taxa máxima o dobro da taxa mínima estipulada. Dessa forma, conseguimos uma melhor demonstração nos gráficos. Para a fila de melhor esforço foi reservada uma largura de banda de 5000 Kbps, tanto para o mínimo quanto para o máximo, sendo também uma escolha de melhor demonstração para os resultados do trabalho.

As filas de QoS criadas foram utilizadas somente nos cenários que levariam em consideração a aplicação das mesmas, que neste caso são os cenários 2 e 3. No cenário 1, como não há priorização dos dados, não foi necessária a criação das filas. Porém, em todos os testes de avaliação de cada cenário, foi utilizado o mesmo vídeo citado anteriormente.

Para que haja um melhor entendimento da provisão da qualidade de serviço na topologia fracionada, a Figura 5.1 mostra os locais exatos da aplicação das políticas de QoS, ou seja, as portas dos *switches* as quais recebem a configuração das filas de QoS, levando em consideração que o *host* cliente do vídeo é o h1 e o *host* servidor é o srv1.

Nas avaliações dos cenários foram colhidas estatísticas das entradas e saídas de algumas portas dos *switches*. Para a realização dessas medições foram usadas as próprias métricas fornecidas pelo *Open vSwitch*. O *Open vSwitch* possui a capacidade de coletar e armazenar dados de números de pacotes e quantidade de *bytes* que passam por uma determinada porta. Logicamente, tal porta e tal *switch* devem ser emulados pelo próprio *software*. Portanto, de posse dessa informação, as estatísticas foram colhidas nas saídas do *switch* s1 e nas entradas do *switch* s3, conforme é mostrado na Figura 5.2.

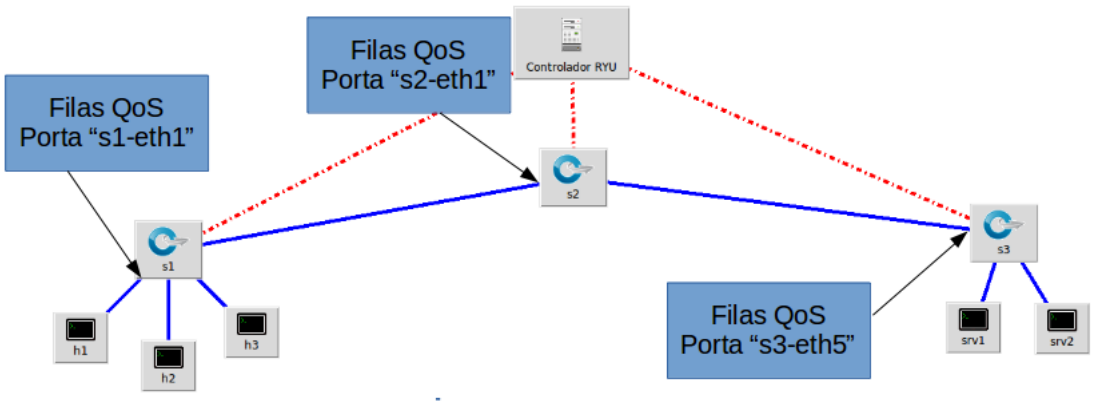


Figura 5.1: Portas do *switch* onde são aplicadas as filas de QoS.

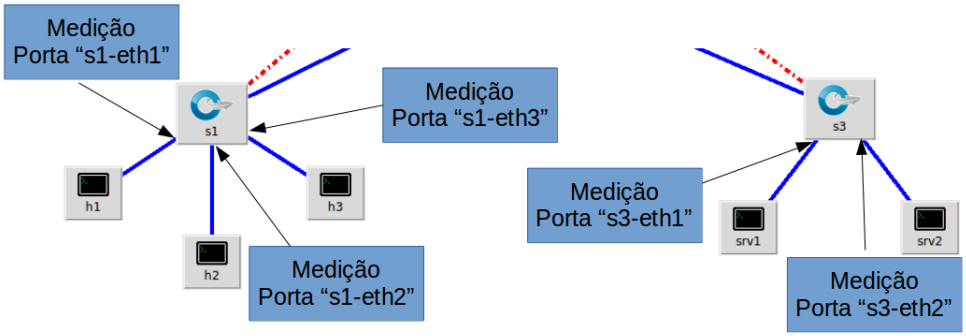


Figura 5.2: Locais das medições para as simulações

Para a demonstração dos resultados das avaliações dos cenários, foram feitos os seguintes mapeamentos: Porta “s3-eth1” = Entrada Vídeo; Porta “s3-eth2” = Entrada iPerf; Porta “s1-eth1” = Saída Vídeo; Porta “s1-eth2” = Saída iPerf 1; Porta “s1-eth3” = Saída iPerf 2.

Todas as simulações tiveram um tempo total de execução de 180 segundos. Isso leva em consideração o tempo total do vídeo, que é de 140 segundos, mais o tempo de simulação em que o vídeo não está sendo executado. Foram realizadas 30 rodadas de simulação para cada um dos testes realizados em cada cenário.

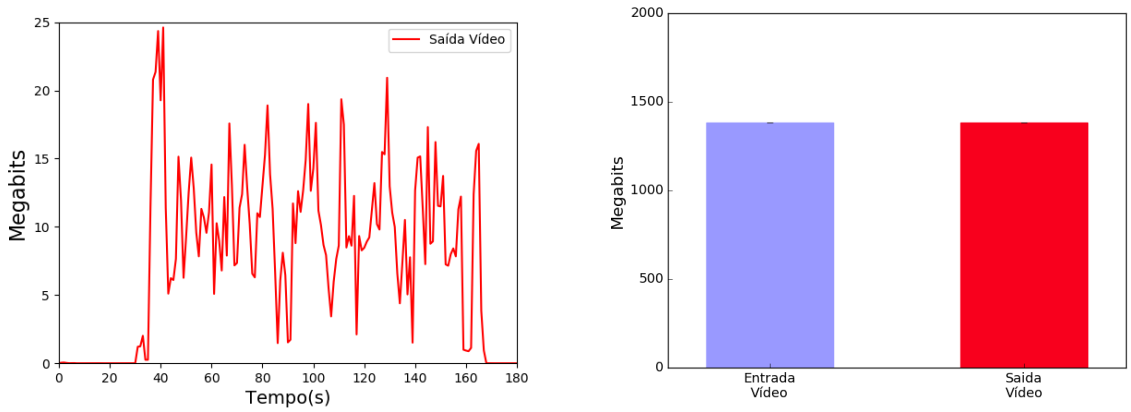
Por fim, todos os testes e simulações realizados neste trabalho foram executados em um *Desktop* 64-bits equipado com um processador Intel Core i7-2600K CPU 3.40 GHz, 8 cores, 8.0 GB de Memória RAM e sistema operacional Ubuntu 16.04.

5.1 Avaliação dos Testes sem QoS

Neste cenário nossa preocupação foi avaliar o comportamento dos fluxos gerados pelas aplicações quando não é provida a QoS.

Um fato a ser considerado neste cenário foi a limitação dos *links* de comunicação entre os *switches*. Um particularidade do uso do Mininet juntamente com o *Open vSwitch* são as altas taxas de transmissão de dados alcançadas na emulação de redes em um computador. Como limitamos o uso do iPerf a 20 Mbps, o mesmo não conseguiria atingir a ocupação da maioria ou do total dos *links*. A não limitação do *link* e o uso do iPerf a essa taxa de transferência não prejudicaria a qualidade do tráfego do vídeo, pois apesar do tráfego *background*, haveria ainda muita largura de banda disponível para que o vídeo pudesse ser transferido sem interferência. Logo, tivemos que aplicar uma redução nos *links* de comunicação entre os *switches* para uma taxa de 25 Mbps. Esse fato ocorre somente aqui neste cenário.

A Figura 5.3(a) demonstra a saída do vídeo gerado ao longo do tempo pela aplicação VLC sem nenhum tipo de tráfego na rede, ou seja, somente o envio do vídeo por parte do servidor e o recebimento por parte do cliente.



(a) Gráfico de saída da simulação 1

(b) Médias da simulação 1

Figura 5.3: Avaliação do cenário 1 sem tráfego *background*

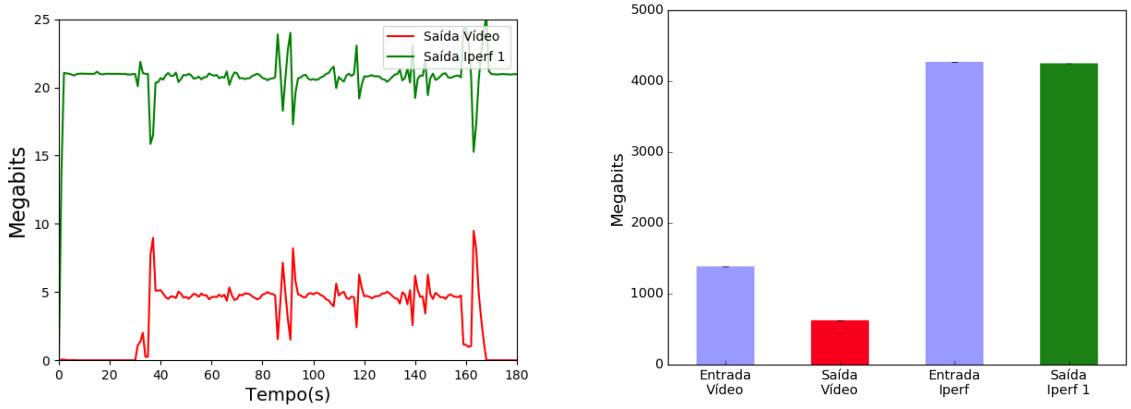
Na Figura 5.3(b) é possível perceber que a média de bits de dados gerada na entrada do vídeo é praticamente idêntica à média produzida na saída do vídeo. A Tabela 5.2 demonstra em mais detalhes as estatísticas dos resultados obtidos nesta primeira simulação, para um intervalo de confiança de 95%.

Tabela 5.2: Estatísticas do cenário 1 sem tráfego *background*

Porta	**Média (Mb)**	**Desvio Padrão (Mb)**	**Variância (Mb)**	**Intervalo de Confiança (LI, LS) (Mb)**
Entrada Vídeo	1382.99	0.00	0.00	[1382.99, 1382.99]
Saída Vídeo	1383.19	0.01	0.00	[1383.19, 1383.20]

É válido ressaltar que quando não há outro tráfego de fundo para concorrer com o tráfego da aplicação de vídeo, não se faz necessário o uso da QoS, como de fato é mostrado na Tabela 5.2. Tal aspecto será reforçado na demonstração dos testes nos outros dois cenários.

Por outro lado, quando há um tráfego de fundo – no nosso trabalho, demonstrado com a ajuda da ferramenta iPerf – podemos ver o decaimento do tráfego de vídeo ao longo da execução da simulação (Figura 5.4(a)), e também uma diferença entre o tráfego gerado na entrada do vídeo em relação aos dados recebidos na saída do vídeo, onde houve uma perda de 54,94% dos bits de dados (Figura 5.4(b)). Por outro lado, a carga de trabalho utilizada no teste não apresentou praticamente nenhuma perda. A Tabela 5.3 demonstra os dados médios obtidos nesta simulação com um intervalo de confiança de 95%.



(a) Gráfico de saída da simulação 2

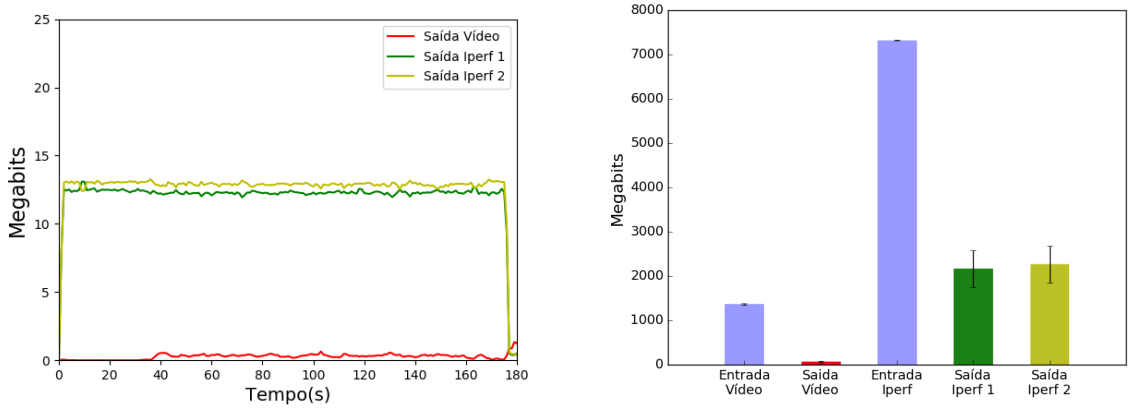
(b) Médias da simulação 2

Figura 5.4: Avaliação do cenário 1 com uma carga de tráfego *background*

Na terceira e última simulação deste cenário, nós adicionamos mais uma carga de iPerf, também gerada nas mesmas taxas de envio de dados da segunda simulação. As Figuras 5.5(b) e 5.5(b) demonstram as entradas e saídas médias dos resultados obtidos nesta simulação.

Tabela 5.3: Estatísticas do cenário 1 com uma carga de tráfego *background*

Porta	**Média (Mb)**	**Desvio Padrão (Mb)**	**Variância (Mb)**	**Intervalo de Confiança (LI, LS) (Mb)**
Entrada Vídeo	1382.99	0.0	0.0	[1382.99, 1382.99]
Saída Vídeo	623.22	0.64	0.41	[622.99, 623.46]
Entrada iPerf	4272.50	2.95	8.7	[4271.40, 4273.60]
Saída iPerf 1	4248.35	2.87	8.23	[4247.28, 4249.42]



(a) Gráfico de saída da simulação 3

(b) Médias da simulação 3

Figura 5.5: Avaliação do cenário 1 com duas cargas de tráfego *background*

Na Tabela 5.4, podemos ter uma visão numérica dos resultados médios obtidos na simulação (intervalo de confiança de 95%). Podemos notar que houve uma perda de 95,30% na saída do vídeo. Houve também uma perda 39,50% de bits de dados quando relacionamos a entrada de dados do iPerf com as suas duas saídas. Isso demonstra que tanto a aplicação de vídeo quanto o tráfego *background* foram prejudicados nesta simulação.

Portanto, levando em consideração essas três simulações em um cenário emulado pelo Mininet sobre as especificações apresentadas no início da seção 5.1, podemos perceber que há uma grande degradação de qualidade, tanto na aplicação do vídeo quanto no tráfego *background*, causados pela disputa do meio. Com isso, percebemos que, se a minimização da perda é desejada, devemos de alguma maneira aplicar os conceitos de QoS para que se tenha uma priorização ou classificação do tráfego, de forma a dar a essas aplicações uma garantia de parâmetros de qualidade. Com isso, demonstramos a necessidade de se especificar requisitos de QoS em uma rede com recursos limitados (largura de banda), de

Tabela 5.4: Estatísticas do cenário 1 com duas cargas de tráfego *background*

Porta	**Média (Mb)**	**Desvio Padrão (Mb)**	**Variância (Mb)**	**Intervalo de Confiança (LI, LS) (Mb)**
Entrada Vídeo	1367.96	50.02	2502.11	[1349.29, 1386.64]
Saída Vídeo	64.31	53.47	2859.47	[44.35, 84.28]
Entrada iPerf	7323.46	37.01	1369.64	[7309.65, 7337.28]
Saída iPerf 1	2163.36	1121.14	1256958.61	[1744.77, 2581.96]
Saída iPerf 2	2267.64	1121.20	1257090.63	[1849.03, 2686.26]

forma a priorizar um tráfego específico.

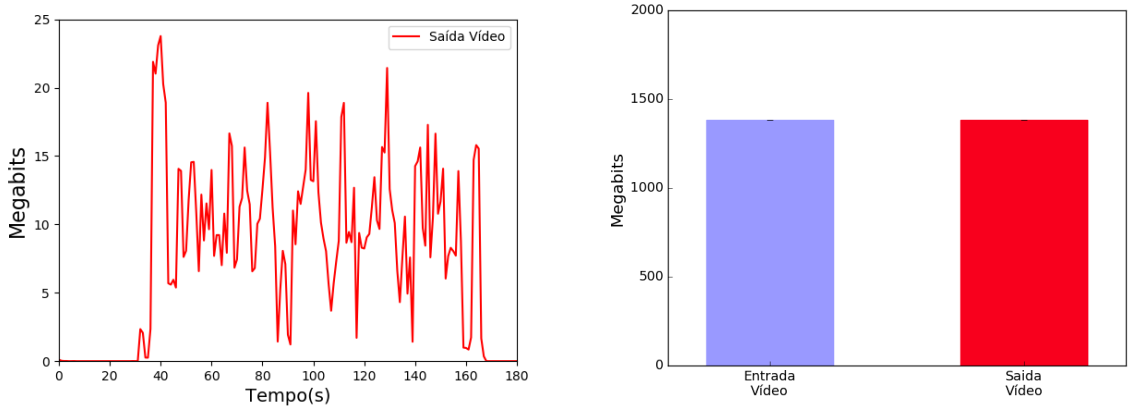
5.2 Avaliação dos Testes com QoS

No segundo cenário temos uma provisão de QoS, porém sua aplicação ocorre no início das atividades, isto é, no momento em que a estrutura de rede é montada e seu funcionamento começa, a aplicação de QoS já é imediatamente aplicada. Neste cenário, desde o início, todo e qualquer tráfego da rede já recebe tratamento de QoS, mesmo sem a sua necessidade imediata (lembrando que todo tráfego não priorizado é enfileirado na fila de melhor esforço).

Levando em consideração a aplicação imediata da QoS nas simulações executadas, podemos ver o quanto o tráfego das aplicações não priorizadas apresentam de perda.

Na primeira simulação, consideramos a aplicação da QoS, porém sem nenhum tráfego gerado em *background*. Os resultados desta análise são apresentados nas Figuras 5.6(a) e 5.6(b). Note novamente que, na média, como esperado, não houve perda entre a entrada do vídeo e a sua saída. A Tabela 5.5 demonstra numericamente os resultados obtidos com um intervalo de confiança de 95%.

Na segunda simulação deste cenário, podemos perceber que o tráfego gerado pelo iPerf não ultrapassa os 5 Mbps, respeitando o tamanho da fila de melhor esforço. Por outro lado, como priorizamos o tráfego da aplicação de vídeo, todo o seu tráfego segue de maneira normal como se não houvesse um tráfego extra na rede. A Figura 5.7(a) demonstra os



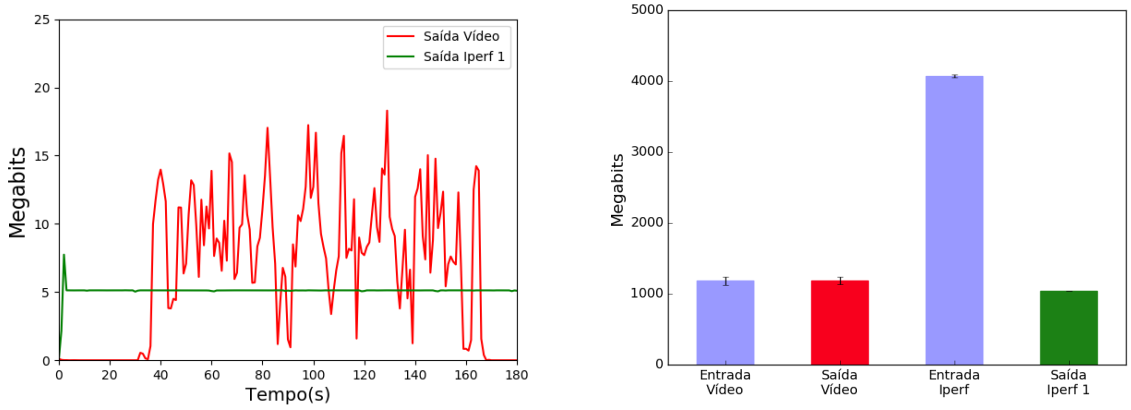
(a) Gráfico de saída da simulação 1 (b) Médias da simulação 1

Figura 5.6: Avaliação do cenário 2 sem carga de tráfego *background*

Tabela 5.5: Estatísticas do cenário 2 sem carga de tráfego *background*

Porta	**Média (Mb)**	**Desvio Padrão (Mb)**	**Variância (Mb)**	**Intervalo de Confiança (LI, LS) (Mb)**
Entrada Vídeo	1383.04	0.00	0.00	[1383.04, 1383.04]
Saída Vídeo	1383.36	0.00	0.00	[1383.36, 1383.36]

resultados das médias obtidas ao longo do tempo. Por sua vez, a Figura 5.7(b) apresenta as médias totais geradas nas entradas do vídeo e do iPerf, e também suas respectivas saídas.



(a) Gráfico de saída da simulação 2 (b) Médias da simulação 2

Figura 5.7: Avaliação do cenário 2 com uma carga de tráfego *background*

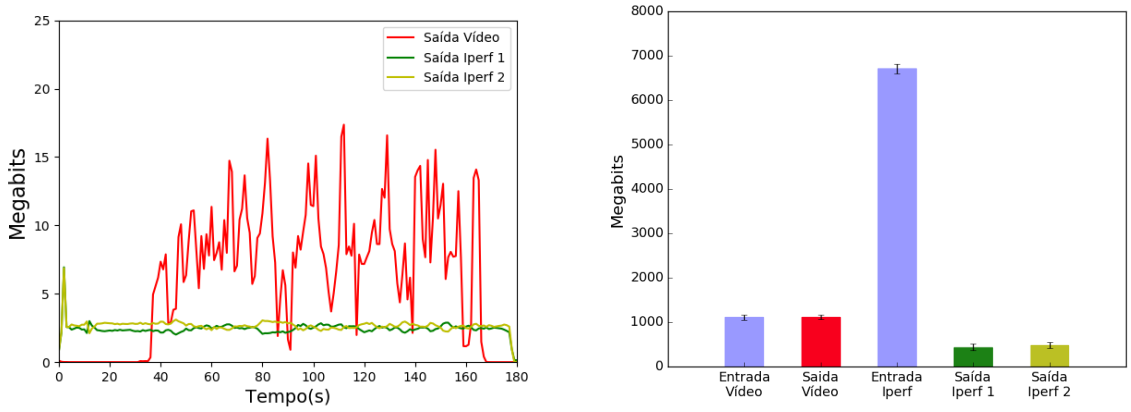
Como podemos perceber, ao aplicarmos a QoS garantindo uma largura de banda para o tráfego de vídeo, na média, o mesmo segue o padrão normal de utilização da rede, como se não houvesse o tráfego em *background*. Por outro lado, o tráfego da saída do iPerf

apresenta uma perda de 74,37%, ou seja, garantimos a largura de banda para a aplicação de vídeo, porém prejudicamos muito a saída do tráfego iPerf. Os detalhes das medições desta simulação podem ser vistos na Tabela 5.6. O intervalo de confiança é de 95%.

Tabela 5.6: Estatísticas do cenário 2 com uma carga de tráfego *background*

Porta	**Média (Mb)**	**Desvio Padrão (Mb)**	**Variância (Mb)**	**Intervalo de Confiança (LI, LS) (Mb)**
Entrada Vídeo	1183.47	142.88	20414.41	[1130.13, 1236.82]
Saída Vídeo	1183.76	142.88	20415.74	[1130.41, 1237.11]
Entrada iPerf	4073.04	65.84	4334.4	[4048.46, 4097.62]
Saída iPerf 1	1044.02	0.83	0.69	[1043.71, 1044.33]

Para a última simulação deste cenário, aumentamos a carga de *background* com mais um envio de iPerf. As Figuras 5.8(a) e 5.8(b) demonstram as saídas obtidas nesta simulação. Podemos ver que, mais uma vez, o tráfego da aplicação de vídeo é priorizado e as saídas geradas pelo iPerf estão limitadas, durante todo o tempo, a 5 Mbps. Notamos também que os tráfegos gerados pelo iPerf disputam a quantidade de largura de banda especificada para o tráfego de melhor esforço, porém o limite estabelecido para eles não é ultrapassado. A Tabela 5.7 mostra maiores detalhes sobre as médias obtidas nas simulações, com um intervalo de confiança de 95%.



(a) Gráfico de saída da simulação 3

(b) Médias da simulação 3

Figura 5.8: Avaliação do cenário 2 com duas cargas de tráfego *background*

Pela Tabela 5.7, podemos perceber que o tráfego gerado pela aplicação de vídeo, na média, teve sua saída garantida, ou seja, o que foi gerado no servidor, foi recebido no

Tabela 5.7: Estatísticas do cenário 2 com duas cargas de tráfego *background*

Porta	**Média (Mb)**	**Desvio Padrão (Mb)**	**Variância (Mb)**	**Intervalo de Confiança (LI, LS) (Mb)**
Entrada Vídeo	1110.05	138.99	19317.15	[1058.16, 1161.95]
Saída Vídeo	1110.35	138.99	19319.11	[1058.45, 1162.24]
Entrada iPerf	6709.46	272.29	74142.41	[6607.79, 6811.12]
Saída iPerf 1	439.25	188.45	35511.99	[368.89, 509.61]
Saída iPerf 2	478.75	181.52	32948.46	[410.98, 546.52]

cliente. Porém, no tráfego *background*, houve uma perda de 86,32% dos bits de dados, isto é, do que foi gerado na entrada do iPerf, somente 13,68% chegou à saída.

Portanto, através dos testes realizados, podemos perceber que o cenário proposto e emulado ofereceu à aplicação de vídeo uma garantia de largura de banda de fato. Porém, o tráfego *background* esteve durante todo o tempo prejudicado, mesmo em momentos onde não havia o tráfego priorizado. Com isso, percebemos que a provisão de QoS garantiu à aplicação de vídeo a saída esperada. Porém, durante toda a execução, o tráfego de outras aplicações foram prejudicados, o que, às vezes, pode não ser bom para um cenário de utilização real de rede.

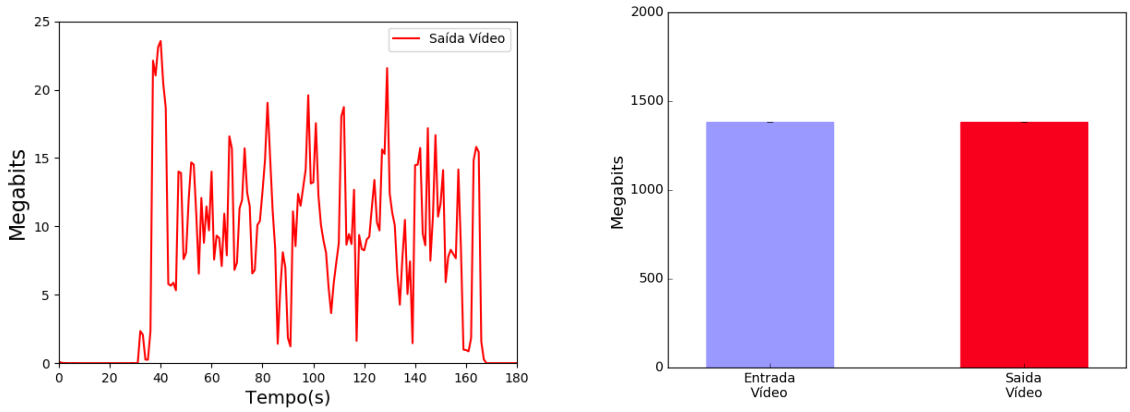
5.3 Avaliação dos Testes com QoS Dinâmica

Neste terceiro e último cenário de avaliação, que representa a nossa proposta de fato, temos uma aplicação dinâmica de QoS, isto é, a provisão da QoS é dada no momento exato da sua necessidade, ficando a cargo da aplicação distribuída o pedido de alocação de reserva de banda. O tráfego gerado em momentos em que não há nenhuma priorização segue seu uso normal, ou seja, é utilizada toda a largura de banda disponível pelas aplicações que ali trafegam. A provisão da QoS neste cenário segue o procedimento demonstrado na Figura 3.4.

No início das execuções, a rede segue o seu comportamento normal. Quando um pedido de vídeo é solicitado ao servidor, o mesmo se encarrega de armazenar as

informações em um arquivo texto e avisar ao controlador RYU sobre o novo pedido de priorização. O controlador RYU, de posse dessa informação, verifica em seu grafo de rede o caminho entre o cliente e o servidor e aplica, se possível, as filas de QoS nas portas específicas de envio, conforme demonstrado na Figura 5.1. Por fim, ao ser aplicada a QoS, o vídeo segue seu fluxo pela rede, saindo do servidor e chegando até o cliente solicitante.

Em uma primeira simulação, demonstramos o envio e o recebimento do vídeo sem nenhum tráfego de fundo, seguindo a mesma metodologia dos cenários 1 e 2. As médias das simulações podem ser vistas nas Figuras 5.9(a) e 5.9(b). Os valores numéricos da simulação podem ser vistos na Tabela 5.8.



(a) Gráfico de saída da simulação 1

(b) Médias da simulação 1

Figura 5.9: Avaliação do cenário 3 sem carga de tráfego *background*

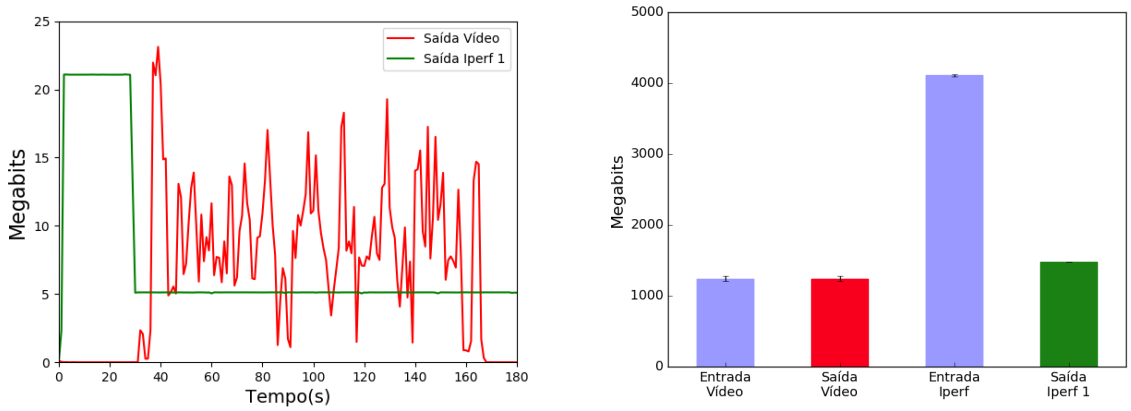
Tabela 5.8: Estatísticas do cenário 3 sem carga de tráfego *background*

Porta	**Média (Mb)**	**Desvio Padrão (Mb)**	**Variância (Mb)**	**Intervalo de Confiança (LI, LS) (Mb)**
Entrada Vídeo	1383.04	0.00	0.00	[1383.04, 1383.04]
Saída Vídeo	1383.36	0.00	0.00	[1383.36, 1383.36]

Aqui, nesta execução, mais uma vez podemos perceber que, se não há um tráfego concorrente com a aplicação a ser priorizada, não se faz necessário o uso da QoS. Porém, quando temos um tráfego *background*, a história pode ser outra, como se vê nas simulações a seguir.

Para a segunda simulação deste cenário, acrescentamos um tráfego de *background*. Nesta simulação, é possível identificar que o tráfego gerado pelo iPerf inicialmente se

mantém com sua saída no máximo, como pode ser visto na Figura 5.10(a). Entretanto, logo que o vídeo é iniciado, ou seja, a priorização da aplicação é requerida, podemos ver que o tráfego *background* sofre uma redução, sendo limitado a 5 Mbps devido a fila de melhor esforço. Com isso, o tráfego da aplicação do vídeo é priorizado e se mantém até o final do experimento com sua largura de banda adequada. É possível ver a comparação das entradas e saídas do vídeo e do iPerf na Figura 5.10(b).



(a) Gráfico da simulação 2 (b) Médias da simulação 2

Figura 5.10: Avaliação do cenário 3 com uma carga de tráfego *background*

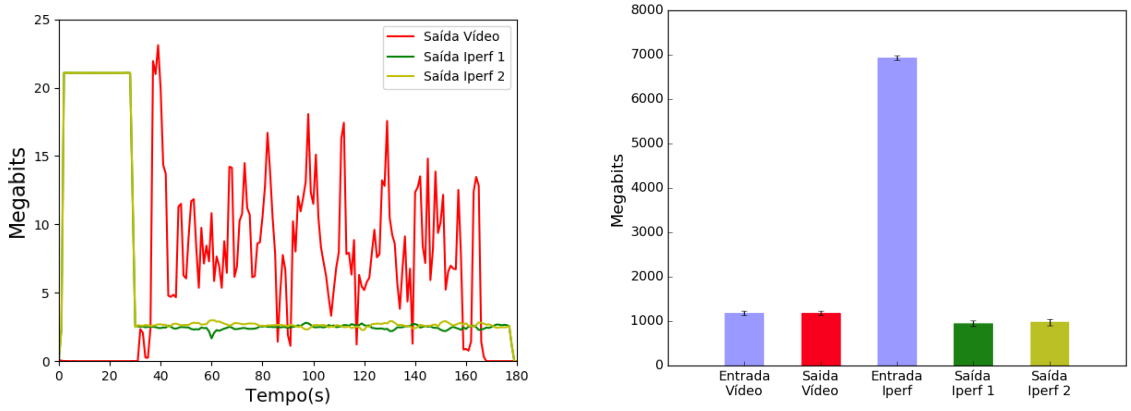
Na Figura 5.10(b) é possível perceber que o tráfego gerado na entrada do vídeo, em média, foi o mesmo da saída (como de fato já era esperado), pois estamos reservando uma largura de banda para esse tráfego específico. Porém, ao analisarmos o tráfego gerado na entrada do iPerf, podemos perceber que houve, em média, uma redução de 63.99% no tráfego de saída. Os resultados numéricos desta simulação podem ser vistos na Tabela 5.9 com um intervalo de confiança de 95%.

Tabela 5.9: Estatísticas do cenário 3 com uma carga de tráfego *background*

Porta	**Média (Mb)**	**Desvio Padrão (Mb)**	**Variância (Mb)**	**Intervalo de Confiança (LI, LS) (Mb)**
Entrada Vídeo	1244.07	86.86	7544.89	[1211.64, 1276.50]
Saída Vídeo	1244.37	86.86	7545.24	[1211.94, 1276.8]
Entrada iPerf	4111.26	40.0	1599.97	[4096.32, 4126.19]
Saída iPerf 1	1480.43	0.86	0.74	[1480.11, 1480.75]

Na terceira e última simulação deste cenário, foram acrescentados dois tráfegos

de iPerf para serem executados paralelamente à aplicação de vídeo. Novamente, os dois tráfegos de iPerf têm seu envio iniciado a uma taxa 20 Mbps cada um. Do mesmo modo que na simulação 2 deste cenário, assim que há um pedido de priorização, os dois tráfegos são reduzidos, limitando-se a 5 Mbps. A partir daí, disputam a largura de banda determinada para eles. Mais uma vez, o tráfego da aplicação de vídeo é priorizado, o que resulta, em média, em uma saída completa dos dados, como mostram as Figuras 5.11(a) e 5.11(b).



(a) Gráfico de saída da simulação 3

(b) Médias da simulação 3

Figura 5.11: Avaliação do cenário 3 com duas cargas de tráfego *background*

Sobretudo, ao analisarmos a Figura 5.11(b), observando os tráfegos do iPerf, podemos perceber que seu tráfego total foi muito prejudicado pela limitação da fila de melhor esforço. Houve, em média, uma redução de 72.23% na saída do tráfego, conforme mostra a Tabela 5.10 (intervalo de confiança de 95%).

No entanto, a proposta geral deste cenário, tanto na simulação 2 quanto nesta, é tentar disponibilizar o máximo possível de largura de banda na rede, para que as aplicações que não tenham o seu tráfego priorizado possam utilizar os recursos da infraestrutura de comunicação em sua máxima capacidade, até que se tenha algum pedido de priorização de tráfego. Portanto, consideramos que essa é a estratégia mais adequada e justa de provisão de QoS no aspecto de garantia de largura de banda.

Portanto, neste cenário, já podemos perceber que houve uma pequena melhora em relação ao cenário 2 no que se refere ao tráfego de aplicações não priorizadas. Já em relação ao cenário 1, observamos que a aplicação que deveria ser priorizada apresentou um

Tabela 5.10: Estatísticas do cenário 3 com duas cargas de tráfego *background*

Porta	**Média (Mb)**	**Desvio Padrão (Mb)**	**Variância (Mb)**	**Intervalo de Confiança (LI, LS) (Mb)**
Entrada Vídeo	1177.45	130.45	17017.65	[1128.74, 1226.15]
Saída Vídeo	1177.74	130.46	17018.98	[1129.04, 1226.45]
Entrada iPerf	6936.54	136.69	18684.01	[6885.51, 6987.58]
Saída iPerf 1	950.55	196.33	38545.14	[877.25, 1023.85]
Saída iPerf 2	975.46	196.33	38545.14	[902.23, 1048.7]

tráfego muito abaixo do esperado. De fato, isso demonstra a necessidade de se especificar a QoS na rede quando há outros tráfegos não priorizados disputando o meio de comunicação.

5.4 Comparação das Três Avaliações

O intuito desta seção é fazer uma comparação entre cada uma das três simulações realizadas nos três cenários.

As primeiras simulações de cada cenário nos mostram que, se tivermos somente o tráfego da aplicação que deva ser priorizado, a provisão de QoS torna-se opcional, pois na média, em todos os testes, a entrada da aplicação priorizada é praticamente idêntica à saída. Isso nos mostra que quando não há tráfego competindo pelos recursos da rede, a provisão de QoS – em particular a nossa abordagem – não é necessária.

Entretanto, quando há um tráfego que possa gerar uma disputa do meio, a aplicação da QoS se torna necessária, visto que, na segunda e terceira simulações do primeiro cenário, tivemos uma perda na saída da aplicação do vídeo, em média, de 54,94% e 95,30%, respectivamente.

Porém, ao aplicarmos a QoS, constatamos que o tráfego não priorizado é bastante prejudicado, dependendo das especificações que o administrador de rede define para o tráfego de melhor esforço. Na segunda e na terceira simulações do segundo cenário, podemos perceber que o tráfego *background* têm suas saídas, em média, prejudicadas em 74,37% e 86,32%, respectivamente.

Por outro lado, ao aplicarmos a QoS de forma dinâmica, ainda teremos as perdas, mas tentamos minimizá-las ao máximo. Logo, ao compararmos a segunda simulação do cenário 2 com a segunda simulação do cenário 3, temos um ganho, em média, de 10,38% do tráfego de saída gerado pelo iPerf. Da mesma forma, comparando a terceira simulação do cenário 2 com a terceira simulação do cenário 3, temos um ganho, em média, de 14,69% do tráfego de saída gerado pelo iPerf.

Portanto, podemos dizer que houve um pequeno ganho de eficiência de nossa abordagem comparando-a com a abordagem da provisão de QoS durante todo o processo de execução. Logicamente, podemos dizer isso levando em consideração as formas nas quais os testes foram realizados, ou seja, considerando a topologia utilizada nas avaliações, o ambiente todo simulado por *software*, o tempo de execução das simulações, as filas de QoS, a quantidade de dados gerados em *background* e a máquina onde os ambientes foram simulados. De fato, não podemos afirmar com precisão se esta proposta continuará com esse ganho de eficiência em outros cenários, considerando as variáveis envolvidas nos ambiente de rede.

5.5 Considerações Finais

Antes de toda a execução do trabalho, pensamos e elaboramos algumas hipóteses que poderiam ocorrer ao longo do desenvolvimento e na obtenção dos resultados.

Assim sendo, partimos para a execução e elaboração e algumas destas hipóteses foram sendo comprovadas e apresentadas ao longo desta monografia. Porém, houve uma que não conseguimos identificar o motivo do comportamento, que foi o comportamento dos gráficos apresentados na Figura 5.4(a) e 5.5(a).

Intuitivamente, os tráfegos das aplicações deveriam disputar o meio físico entre eles. Entretanto, vimos que isso não ocorreu e podemos perceber, ao longo de outros experimentos, que isto é um comportamento do *software* de emulação de redes.

Portanto, constatamos que quase todas as hipóteses se confirmaram desde o início deste projeto. Somente o comportamento diferenciado do *software* de emulação de redes, no primeiro cenário, que nos surpreendeu.

6 Conclusão e Trabalhos Futuros

A crescente demanda por aplicações distribuídas de alto desempenho vem gerando cada vez mais a busca por melhores maneiras de se prover a qualidade de serviço na rede.

Uma forma mais simples de especificação de parâmetros de QoS na rede é o uso das redes definidas por *software*. Porém, essa especificação pode ocorrer de uma forma mais inteligente e justa com outras aplicações que não requerem rígidos parâmetros garantidos pela rede.

Diante disso, este trabalho visou abordar uma maneira mais justa de aplicação de QoS sobre as SDNs, de forma a minimizar o impacto causado a tráfegos não priorizados pela rede. Buscamos um método de aplicação dinâmica de QoS em SDN, visando melhorar o tráfego das aplicações distribuídas de alto desempenho, ao mesmo tempo minimizando a degradação de tráfegos não priorizados.

De uma forma geral, nossa abordagem teve um pequeno ganho em relação a uma abordagem prática de utilização, onde é visado a especificação da QoS durante todo o funcionamento do experimento. Mas, para isso, levamos em consideração o ambiente restrito e simulado, não podendo afirmar nada com relação a um ambiente real de utilização, onde as variáveis consideradas neste trabalho podem variar bastante.

Logo, a hipótese de se aplicar uma reserva de banda dinamicamente nas SDNs para tentar melhorar o tráfego das aplicações distribuídas de alto desempenho, prejudicando minimamente o tráfego de outras aplicações, foi confirmada. Logicamente, essa confirmação leva em consideração o ambiente simulado e controlado, com uma quantidade de testes razoáveis e específicos.

Junto a isso, respondemos a pergunta levantada no início deste trabalho: como melhorar o tráfego de uma aplicação distribuída de alto desempenho sobre as redes definidas por *software*, utilizando, para tanto, os conceitos de QoS? De uma maneira sucinta e levando em consideração a emulação de um ambiente de redes, podemos perceber que, fornecendo uma garantia de largura de banda de forma dinâmica, podemos sim melhorar o tráfego dessas aplicações, e ainda impactar minimamente o tráfego de aplicações

não priorizadas. Para tal, utilizamos um *software* de visualização de vídeo (VLC Media Player) simulando uma aplicação distribuída de alto desempenho e um *software* capaz de gerar grandes tráfegos na rede (iPerf), trazendo os ambientes para mais próximo da realidade, porém sendo emulados por um *software* de redes (Mininet).

Com isso, conseguimos mostrar que, nos ambientes propostos no trabalho, nossa implementação se mostrou eficaz. De certa forma, este trabalho contribui para que novas propostas de provisão de QoS sobre as SDNs possam surgir e, cada vez mais, possam colaborar com pesquisas nessas áreas de interesse.

Por fim, a falta de um cenário real de implementação, as limitações do *software* de emulação de redes Mininet, a falta de computadores com um maior poder de processamento e os poucos cenários utilizados, não nos permitem afirmar de fato que nossa proposta de aplicação de QoS sobre as redes definidas por *software*, de forma dinâmica, terá um ganho real sobre outras propostas de provisão de QoS, como em ambientes reais, ambientes de maior complexidade e ambientes onde o tráfego de aplicações distribuídas de alto desempenho supere o tráfego de aplicações de uso comum, onde praticamente todas as aplicações deverão ter seu tráfego priorizado.

6.1 Trabalhos Futuros

Devido às limitações enfrentadas por nós neste trabalho, como a falta de um cenário real de utilização, a escassez de recursos, as limitações do *software* Mininet e a baixa complexidade da arquitetura, não podemos afirmar de fato que nossa proposta teve grandes ganhos em relação a outras propostas.

Portanto, diante dessas circunstâncias, visamos novas implementações de nossa proposta em um cenário não emulado e que tenha uma maior complexidade, para que, de fato, possa se afirmar algo em relação a mesma.

Uma outra proposta que poderia também ser feita seria uma comparação entre a proposta aqui apresentada e outras formas de utilização de QoS sobre as SDNs, e assim dizer que realmente há um ganho na sua utilização, ou há perdas em relação a mesma.

Bibliografia

AFAQ, M.; REHMAN, S. U.; SONG, W.-C. Visualization of elephant flows and QoS provisioning in SDN-based networks. In: IEEE. *Network Operations and Management Symposium (APNOMS), 2015 17th Asia-Pacific*. [S.l.], 2015. p. 444–447.

AMAZON. *Amazon, 2018. Página inicial*. 2018. ⟨https://www.amazon.com/⟩. Acessado em: 07 dez. 2018.

CASADO, M. et al. Rethinking enterprise network control. *IEEE/ACM Transactions on Networking*, IEEE, v. 17, n. 4, p. 1270–1283, 2009.

CISCO. *Cisco Brasil, 2018. Página inicial*. 2018. ⟨https://www.cisco.com/c/pt_br/index.html⟩. Acessado em: 07 dez. 2018.

COLCHER, S. *VoIP: Voz sobre IP*. [S.l.]: Elsevier, 2005.

COMMUNITY, R. S. F. *COMPONENT-BASED SOFTWARE DEFINED NETWORKING FRAMEWORK - Build SDN Agilely, 2017. Página inicial*. 2017. ⟨https://osrg.github.io/ryu/⟩. Acessado em: 07 dez. 2018.

CONFIGURAÇÕES, taxas de bits e resoluções do codificador ao vivo. 2018. ⟨https://support.google.com/youtube/answer/2853702?hl=pt-BR⟩. Acessado em: 09 nov. 2018.

COSTA, L. C. et al. Balanceamento de carga utilizando planos de dados OpenFlow comerciais. Universidade Federal de Juiz de Fora, 2016.

DINIZ, P. H.; JUNIOR, N. A. Ferramenta iPerf: geração e medição de tráfego TCP e UDP. *NOTAS TÉCNICAS*, v. 4, n. 2, 2014.

DIORIO, R. F.; TIMÓTEO, V. S. Per-flow routing with QoS support to enhance multimedia delivery in OpenFlow SDN. In: *Proceedings of the 22Nd Brazilian Symposium on Multimedia and the Web*. New York, NY, USA: ACM, 2016. (Webmedia '16), p. 167–174. ISBN 978-1-4503-4512-5. Disponível em: ⟨http://doi.acm.org/10.1145/2976796.2976844⟩.

D'SOUZA, D. et al. Improving QoS in a software-defined network. *University of Colorado, Boulder, Capstone*, 2016.

FEAMSTER, N.; REXFORD, J.; ZEGURA, E. The road to SDN: an intellectual history of programmable networks. *ACM SIGCOMM Computer Communication Review*, ACM, v. 44, n. 2, p. 87–98, 2014.

FERNANDES, E. L.; ROTHENBERG, C. E. OpenFlow 1.3 software switch. *Salão de Ferramentas do XXXII Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos SBRC*, p. 1021–1028, 2014.

GENERAL Switch Management Protocol (GSMP) v3 specification. In: . [s.n.], 2002. Disponível em: ⟨https://www.ietf.org/rfc/rfc3292.txt⟩.

GOOGLE. *Google Brasil, 2018. Página inicial*. 2018. ⟨https://www.google.com.br/⟩. Acessado em: 07 dez. 2018.

HUMERNBRUM, T.; GLINKA, F.; GORLATCH, S. A northbound API for QoS management in real-time interactive applications on software-defined networks. v. 9, p. 607–615, 08 2014.

IPERF.FR. *iPerf - The ultimate speed test tool for TCP, UDP and SCTP, 2018. Página inicial*. 2018. ⟨https://iperf.fr/⟩. Acessado em: 07 dez. 2018.

KAMIENSKI, C. A.; SADOK, D. Qualidade de serviço na Internet. *minicurso, 18o SBRC, Belo Horizonte/MG*, 2000.

KARAKUS, M.; DURRESI, A. Quality of Service (QoS) in Software Defined Networking (SDN): A survey. *Journal of Network and Computer Applications*, Elsevier, v. 80, p. 200–218, 2017.

KAUR, K.; SINGH, J.; GHUMMAN, N. S. Mininet as software defined networking testing platform. In: *International Conference on Communication, Computing & Systems (ICCCS)*. [S.l.: s.n.], 2014. p. 139–42.

KREUTZ, D. et al. Software-defined networking: A comprehensive survey. *Proceedings of the IEEE*, Ieee, v. 103, n. 1, p. 14–76, 2015.

LANTZ, B.; HELLER, B.; MCKEOWN, N. A network in a laptop: Rapid prototyping for software-defined networks. 2010.

MACEDO, D. F. et al. Programmable networks—from software-defined radio to software-defined networking. *IEEE communications surveys & tutorials*, IEEE, v. 17, n. 2, p. 1102–1125, 2015.

MCKEOWN, N. et al. OpenFlow: enabling innovation in campus networks. *ACM SIGCOMM Computer Communication Review*, ACM, v. 38, n. 2, p. 69–74, 2008.

MIRCHEV, A. Survey of concepts for QoS improvements via SDN. *Future Internet (FI) and Innovative Internet Technologies and Mobile Communications (IITM)*, v. 33, p. 1, 2015.

MORENO, M. F.; SOARES, L. F. G. *Um framework para provisão de QoS em sistemas operacionais*. Dissertação (Mestrado) — Pontifícia Universidade Católica do Rio de Janeiro, 2002.

NUNES, B. A. A. et al. A survey of software-defined networking: Past, present, and future of programmable networks. *IEEE Communications Surveys & Tutorials*, IEEE, v. 16, n. 3, p. 1617–1634, 2014.

OLIVEIRA, A. et al. Arquitetura de provisão de qualidade de serviço para aplicações distribuídas de alto desempenho em redes definidas por software. In: *SBRC 2018 - XXIII WGRS*. [s.n.], 2018. Disponível em: ⟨http://ojs.sbc.org.br/index.php/wgrs/article/view/2364⟩.

ORGANIZATION, V. *VideoLAN, um projeto e uma organização sem fins lucrativos, 2018. Página inicial*. 2018. ⟨https://www.videolan.org/index.pt-BR.html⟩. Acessado em: 07 dez. 2018.

REIZNAULTT, W. L. *SDN/OpenFlow*. 2018. ⟨http://www.ic.unicamp.br/~edmundo/MC822/mc822/MO655/SDN%20e%20OpenFlow-final.pdf⟩. Acessado em: 22 out. 2018.

ROTHENBERG, C. E. et al. OpenFlow e redes definidas por software: um novo paradigma de controle e inovação em redes de pacotes. *Cad. CPqD Tecnologia, Campinas*, v. 7, n. 1, p. 65–76, 2010.

TEAM, M. *Mininet, 2018. Página inicial*. 2018. ⟨http://mininet.org/⟩. Acessado em: 07 dez. 2018.

TENNENHOUSE, D. L. et al. A survey of active network research. *IEEE communications Magazine*, IEEE, v. 35, n. 1, p. 80–86, 1997.

TOMOVIC, S.; PRASAD, N.; RADUSINOVIC, I. SDN control framework for QoS provisioning. In: IEEE. *Telecommunications Forum Telfor (TELFOR), 2014 22nd*. [S.l.], 2014. p. 111–114.

VOGEL, A. et al. Distributed multimedia and QoS: A survey. *IEEE multimedia*, IEEE, v. 2, n. 2, p. 10–19, 1995.

VSWITCH, O. O. *Open vSwitch, 2018. Página inicial*. 2018. ⟨https://www.openvswitch.org/⟩. Acessado em: 07 dez. 2018.

YU, T.-F.; WANG, K.; HSU, Y.-H. Adaptive routing for video streaming with QoS support over SDN networks. In: *2015 International Conference on Information Networking (ICOIN)*. [S.l.: s.n.], 2015. p. 318–323. ISSN 1550-445X.