UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ
TECNÓLOGO EM REDES DE COMPUTADORES

**DÉBORACH RAYANNE MACÊDO VIEIRA**

# SOLUÇÃO PARA APROVISIONAMENTO DE QOS EM SERVIÇOS DE VOIP NA PLATAFORMA OPENFLOW

**QUIXADÁ**
**2016**

**DÉBORACH RAYANNE MACÊDO VIEIRA**

# SOLUÇÃO PARA APROVISIONAMENTO DE QOS EM SERVIÇOS DE VOIP NA PLATAFORMA OPENFLOW

Trabalho de Conclusão de Curso submetido à Coordenação do Curso Superior de Tecnologia em Redes de Computadores da Universidade Federal do Ceará como requisito parcial para obtenção do grau de Tecnólogo.

Área de concentração: Computação

Orientador: Prof. Dr. Arthur de Castro Callado

**QUIXADÁ**
**2016**

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Biblioteca Universitária
Gerada automaticamente pelo módulo Catalog, mediante os dados fornecidos pelo(a) autor(a)

V714s Vieira, Déborach Rayanne Macedo.
Solução para aprovisionamento de QoS em serviços de VoIP na plataforma OpenFlow / Déborach Rayanne Macedo Vieira. – 2016.
36 f. : il. color.

Trabalho de Conclusão de Curso (graduação) – Universidade Federal do Ceará, Campus de Quixadá, Curso de Redes de Computadores, Quixadá, 2016.
Orientação: Prof. Dr. Arthur de Castro Callado.

1. Redes definidas por software. 2. Openflow. 3. Qualidade de serviço. I. Título.

CDD 004.6

**DÉBORACH RAYANNE MACÊDO VIEIRA**

# SOLUÇÃO PARA APROVISIONAMENTO DE QOS EM SERVIÇOS DE VOIP NA PLATAFORMA OPENFLOW

Trabalho de Conclusão de Curso submetido à Coordenação do Curso Superior de Tecnologia em Redes de Computadores da Universidade Federal do Ceará como requisito parcial para obtenção do grau de Tecnólogo.

Área de concentração: Computação

Aprovado em: 21 / Dezembro / 2016.

BANCA EXAMINADORA

_____
Prof. Dr. Arthur de Castro Callado (Orientador)
Universidade Federal do Ceará - UFC

_____
Prof. Dra. Paulyne Matthews Jucá
Universidade Federal do Ceará - UFC

_____
Prof. Me. Marcos Dantas Ortiz
Universidade Federal do Ceará - UFC

À minha mãe, Solange, que é a mulher mais batalhadora que eu conheço; sem seu esforço e dedicação, nada disso seria possível.

## AGRADECIMENTOS

Agradeço a todos os professores que fizeram parte dessa fase acadêmica, em especial Michel Sales, Paulo Antonio, Atslands Rocha, Marcos Dantas, João Marcelo, Jeandro Bezerra, Ticiana Linhares, João Lavor, Fábio Dias, Marcio Maia e Erivelton Aragão.

Ao professor Arthur Callado, que além de um ótimo professor e orientador, se mostrou um exemplo de pessoa a ser seguido, sempre sendo compreensivo, ajudando sempre que possível e oferecendo os melhores conselhos.

Todos os colegas de classe, em especial as amigas Wilkelany, Liana, Geyciane e Patrícia, e os amigos Wendel, Lucas, Heli, Wellington, Matheus e Jorge.

Agradeço aos amigos da bolsa de Iniciação Científica do Projeto de Monitoramento de Tráfego Aéreo e à FUNCAP.

Aos amigos e aos profissionais da bolsa de Suporte na Secretaria Municipal da Educação de Quixadá.

Agradeço ao VoipMonitor, por disponibilizar uma versão paga da ferramenta, de forma gratuita, para que eu pudesse fazer a conclusão desse trabalho.

Aos meus amigos da escola, que me acompanham desde a infância, Geysa, Rouziane, Natália, Kaique e Sara.

Aos meus irmãos, Patrícia, Yres e Valdenor, meus avós, cunhados, tias, tios, primos e meu sobrinho Davi.

Rayanne, que além de tudo, é a minha melhor amiga e sempre me aconselhou das melhores formas para que eu me mantivesse focada nos meus objetivos.

E finalmente, ao meu pai, Irandir, que não tive a oportunidade de conhecer bem por causa de sua partida precoce, e a minha mãe, Solange, que é a mulher mais batalhadora que eu conheço. Sem seu esforço e dedicação, nada disso seria possível.

*Às vezes é difícil seguir seu coração, mas lágrimas não significam que você está perdendo, apenas seja verdadeiro com quem você é.*
(Jessica Cornish)

## RESUMO

As Redes Definidas por *Software* são um novo paradigma que permitem a introdução de inovações sem interromper o uso diário através do padrão *OpenFlow*. Prover QoS fim-a-fim ainda é um dos maiores problemas para o sucesso de determinados serviços, como multimídia, nos sistemas heterogêneos de telecomunicações usados. Esse trabalho propõe uma técnica para prover Qualidade de Serviço (QoS) fim-a-fim, através do controle e redistribuição eficiente dos recursos, em tráfego de Voz sobre IP (*VoIP*) usando o padrão *OpenFlow*. Este trabalho também realiza a avaliação dessa técnica.

Palavras chave: Redes de Definidas por *Software*, *OpenFlow*, Qualidade de Serviço, *VoIP*.

## ABSTRACT

The Software Defined Networks are a new paradigm that enable the introduction of innovations without disrupting everyday use through the OpenFlow standard. Providing end-to-end QoS is still a major problem for the success of certain services in heterogeneous systems used in telecommunications, such as multimedia. This work proposes a technique to provide end-to-end Quality of Service (QoS) through the control and efficient reallocation of resources in Voice Over IP (VoIP) traffic using the OpenFlow standard This work also presents a performance evaluation of the proposed technique.



Keywords: Software Defined Networks, OpenFlow, Quality Of Service, VoIP.

## LISTA DE ILUSTRAÇÕES

## LISTA DE TABELAS

## SUMÁRIO

# 1. INTRODUÇÃO

A tecnologia IP é muito flexível e provê um mecanismo uniforme de transporte fim-a-fim independente dos mecanismos implementados em camadas inferiores. A aceitação desse protocolo é tão grande que a Internet hoje tem uma importância fundamental em todas as empresas e organizações. É possível afirmar que a Internet é usada durante todo tempo, e por causa desse grande sucesso, as capacidades das redes ficaram limitadas, chegando a um alto nível de complexidade, um ponto conhecido como "ossificação" da rede (MCKEOWN et al., 2008), dificultando a adição de inovações, porque atrapalhariam o uso diário.

Uma das principais propostas para conciliar o desenvolvimento de inovações sem prejudicar o tráfego de produção é a virtualização de redes (RATNASAMY et al. 2005, apud FEAMSTER et al. 2007), sendo uma técnica promissora para o desenvolvimento de novas arquiteturas para a Internet (MATTOS; DUARTE, 2012). As Redes Definidas por Software (SDN) são uma das formas de virtualização de redes, e são uma tendência na área de redes de computadores.

O OpenFlow é um padrão aberto e é a primeira implementação com sucesso de SDN (MCKEOWN et al., 2008). Também é o principal recurso disponível para pesquisadores realizarem testes e implantarem novos protocolos em uma rede física ou virtualizada, levando posteriormente esses experimentos para o campo de produção.

Prover QoS fim-a-fim ainda é um dos maiores problemas para o sucesso de determinados serviços, tais como multimídia, nos sistemas heterogêneos de telecomunicações usados (SILVA et al., 2013), pois as arquiteturas normais não conseguem fazer Qos fim-a-fim. Esse trabalho propõe uma técnica para prover Qualidade de Serviço (QoS) fim-a-fim através do controle e redistribuição eficiente dos recursos para tráfego de Voz sobre IP (VoIP) usando o padrão OpenFlow. Ele poderá ajudar pesquisadores da área em seus projetos e no futuro ser útil aos usuários finais.

## 1.1. Objetivos

O objetivo geral desse trabalho é propor um novo mecanismo para prover qualidade de serviço em redes virtuais na plataforma OpenFlow para manter os requisitos de serviços de Voz sobre IP.

Os seguintes objetivos específicos foram levados em consideração para alcançar a meta do objetivo geral:

- Propor um mecanismo que possa controlar os recursos, priorizando o tráfego *VoIP*;
- Avaliar se o mecanismo garante a qualidade do serviço de *VoIP* através do controle dos recursos disponíveis, e se o uso dos recursos da rede é otimizado.

## 1.2. Metodologia

Esta seção tem por objetivo detalhar os passos utilizados nesse trabalho para criação da proposta e sua avaliação. Os passos foram:

- Pesquisa bibliográfica – nesta etapa, foram realizadas leituras, análises e interpretações de livros, periódicos, artigos e outros documentos para conhecer as diferentes contribuições científicas disponíveis sobre o tema;
- Criação de um cenário de exemplo para aprendizado do monitoramento dos recursos pelos serviços – foi criado um cenário para aprender sobre o funcionamento das ferramentas e monitorar o uso dos recursos pelos serviços;
- Escolha de técnica e métricas que foram utilizadas na análise dos dados – para fazer a análise dos dados, a escolha de métricas e das técnicas de análise foi feita com base nas informações do monitoramento dos recursos;
- Criação de cenário de avaliação de acordo com os requisitos de VoIP – depois de conhecer as ferramentas e de ter feito a escolha das técnicas, foi preciso conhecer os requisitos para que os serviços de VoIP funcionem bem e construir um cenário para avaliar se as técnicas e métricas escolhidas são válidas;
- Definição de valores mínimos e máximos para cada fila – foi preciso definir valores mínimos e máximos para as filas nos fluxos de acordo com a classe de serviço.
- Realização de testes para obter os resultados – foram colocados em execução alguns experimentos para obtenção de resultados sobre o comportamento do tráfego;
- Análise e interpretação dos resultados – após a realização dos testes, foi preciso fazer uma análise sobre os dados obtidos e interpretá-los de forma que se tenha informações para fazer uma conclusão.

## 1.3. Organização do Texto

Essa seção tem por objetivo detalhar a organização do texto. No capítulo **1,** de introdução, são mostrados o objetivo principal do trabalho e a metodologia. O capítulo **2** mostra os principais conceitos, e um levantamento bibliográfico sobre Redes Definidas por Software, incluindo as ferramentas de desenvolvimento utilizadas no trabalho, OpenFlow,

Qualidade de Serviço e VoIP. Já no capítulo **3** temos os resultados obtidos com a execução dos experimentos, em que é feita uma avaliação de desempenho da solução. Finalmente, na seção **4** são descritas as conclusões, tecidas as considerações resultantes da avaliação da solução e mostrados os trabalhos futuros.

## 2. REVISÃO BIBLIOGRÁFICA

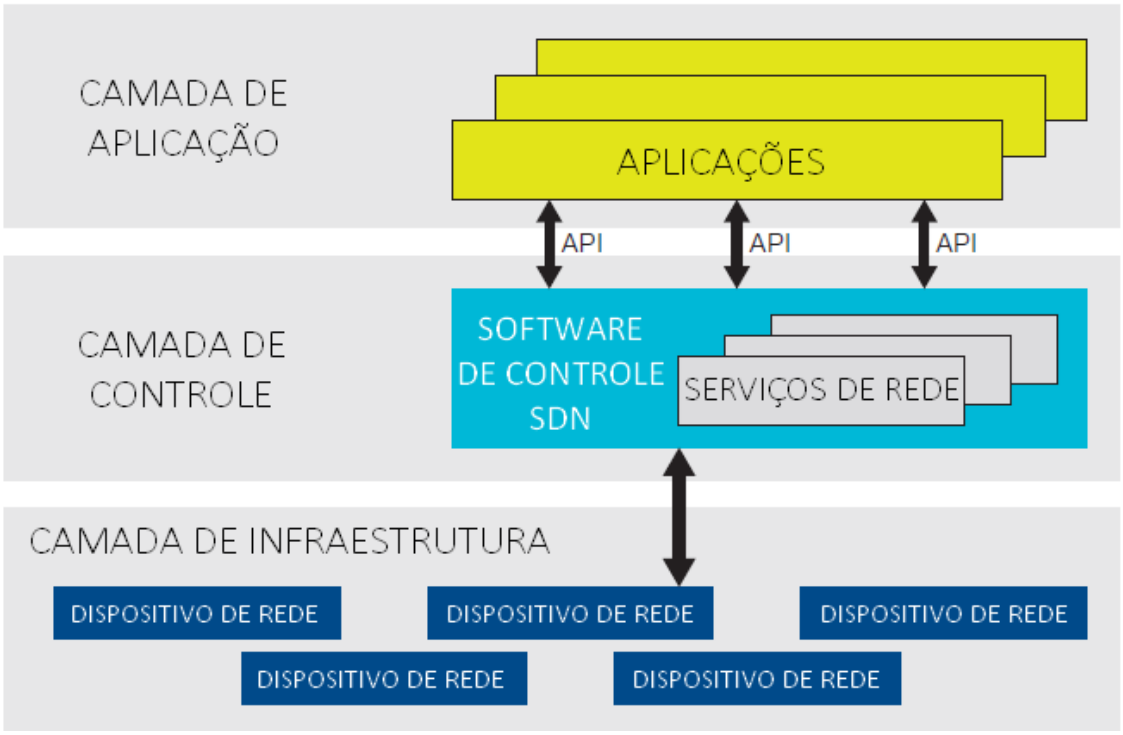
Nesta seção, são apresentados quatro conceitos-chave que ajudarão no entendimento do trabalho. Primeiro será falado um pouco mais sobre Redes Definidas por Software (SDN) e o OpenFlow, depois será explicado o conceito de Qualidade de Serviço (QoS) e por último será explicado o conceito de Voz sobre IP (VoIP).

### 2.1. Redes Definidas por Software (SDN)

Redes Definidas por Software representam uma evolução sobre inovações das redes. Os principais componentes da indústria de Redes, incluindo vendedores de equipamentos de redes, provedores de serviços de Internet, provedores de serviços de nuvem e usuários, estão trabalhando com SDN ou estudando os aspectos desta (PAUL; PAN; JAIN, 2011).

Na arquitetura SDN, o plano de controle e o plano de encaminhamento de dados são separados. O controle da rede é logicamente centralizado em controladores SDN baseados em software, que mantêm uma visão global da rede (SOFTWARE, 2012), a qual passa a ser dividida em camadas, como pode ser visto na Figura 1.

Figura 1 - Camadas de uma SDN.




Fonte: SOFTWARE, 2012.

Os aplicativos comerciais encontram-se na camada de aplicação. Na camada de controle localiza-se o controlador SDN, esta age como um filtro entre a camada de aplicação e a de infraestrutura (NDE; KHONDOKER, 2016). Finalmente, na camada de infraestrutura situam-se os dispositivos de rede. Entre as camadas de controle e infraestrutura, há uma interface programável, que irá implementar as regras de encaminhamento. Nesse trabalho, essa interface será o OpenFlow. Essa arquitetura simplifica bastante o projeto e as operações da rede. SDN simplifica também os dispositivos, porque eles não precisam entender os padrões dos protocolos, apenas atender às instruções dos controladores.

### 2.1.1. Ferramentas de desenvolvimento SDN

A tecnologia SDN foi proposta para facilitar a inovação e evolução da internet e permitir o rápido desenvolvimento de novos serviços e protocolos (NUNES *et al*., 2014). Nas seções a seguir, são apresentadas breves introduções sobre as ferramentas que serão usadas no trabalho.

#### 2.1.1.1. Mininet

O *Mininet* foi uma ferramenta utilizada na fase de aprendizado sobre SDN, pois permite que todo o padrão *OpenFlow* seja emulado em apenas uma máquina, simplificando o processo de desenvolvimento (NUNES *et al*., 2014). O *Mininet* é leve e usa virtualização no nível de Sistema Operacional. Com ele os usuários podem implementar novas características para a rede ou até uma nova arquitetura, podendo testá-las em grandes topologias com aplicações que simulam tráfego, e depois desenvolver exatamente o mesmo código em ambiente de produção (LANTZ; HELLER; MCKEOWN, 2010). Além disso, o código pode ser salvo e compartilhado como uma máquina virtual, então outros desenvolvedores podem fazer download, rodar, examinar e modificar a topologia (LANTZ; HELLER; MCKEOWN, 2010).

#### 2.1.1.2. POX

O controlador SDN é o software responsável por tomar decisões e adicionar e remover as entradas na tabela de fluxos de acordo com o objetivo desejado (ROTHENBERG *et al*., 2011), neste trabalho, o controlador POX foi utilizado para tomada de decisões. O controlador POX é uma plataforma para o rápido desenvolvimento de controle da rede usando

a linguagem de programação *Python* (ABOUT, 2014). Ele interage com switches *OpenFlow*, tendo uma interface chamada *Pythonic* e funciona nos sistemas operacionais Linux, Mac OS e Windows (ABOUT, 2014).

O controlador POX possui um estoque de componentes, alguns destes proveem funcionalidades primordiais, enquanto alguns implementam novas aplicações (POX, 2016). Como exemplo de funcionalidade, temos o *forwarding.l2_learning*, que faz o *switch OpenFlow* agir como um *switch layer* 2 e é usado como base para implementação deste trabalho.

## 2.2. *OpenFlow*

O *OpenFlow* é o primeiro padrão de interface de comunicação definida entre as camadas de controle e encaminhamento de uma SDN (SOFTWARE, 2012). Esse padrão é aberto e permite modificar o comportamento dos dispositivos de encaminhamento, como switches e roteadores, tanto físicos como virtuais, através de políticas definidas pelo controlador.

McKeown (2008) define OpenFlow como uma plataforma de comutadores programáveis que permite ao administrador da rede particionar o encaminhamento dos pacotes com base em definições de fluxos. Um fluxo é definido como um conjunto de características da comunicação entre dois nós na rede (MCKEOWN et al., 2008). Os pesquisadores podem controlar os fluxos escolhendo as rotas (Origem/Destino) para os pacotes seguirem, assim, podem ser desenvolvidos novos protocolos de roteamento, modelos de segurança e até alternativas ao IP (MCKEOWN et al., 2008).

O caminho de dados em um *Switch OpenFlow* consiste em uma tabela de fluxos e uma ação ou política associada com cada entrada de fluxos (MCKEOWN et al., 2008). A lista de ações suportadas pelo OpenFlow é extensa, tais como encaminhamento de pacotes para uma determinada porta, encapsular e enviar um fluxo de pacotes para o controlador e descartar um fluxo de pacotes.

A ação de encapsular um fluxo de pacotes e enviar para o controlador tipicamente se refere a quando o primeiro pacote chega a um novo fluxo, e então, o controlador irá decidir se esse fluxo deve ser adicionado à tabela de fluxos (MCKEOWN et al., 2008), que pode ser vista na Figura 2.

Figura 2 - Principais componentes de uma tabela de fluxo.

| Regra de fluxo | Contador | Instruções |
|---|---|---|

Fonte: Adaptada de OPENFLOW, 2011.

Uma entrada na tabela de fluxos tem três campos: um cabeçalho que define o fluxo, um campo para estatísticas que mantém o número de pacotes e bytes para cada fluxo e o tempo passado desde que o último pacote correspondente à esse fluxo apareceu (MCKEOWN et al., 2008), este campo também ajuda a determinar se um fluxo está inativo, e por último, a ação que define como os pacotes devem ser processados.

Figura 3 - Campos dos pacotes que correspondem à entrada de fluxos.

| Porta de entrada | Ethernet | | | VLAN | | IP | | | | TCP/UDP | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | Origem | Destino | Tipo | ID | Prioridade | Origem | Destino | Protocolo | ToS | Porta de origem | Porta de destino |

Fonte: Adaptada de OPENFLOW, 2009.

Um fluxo é constituído pela combinação de campos do cabeçalho do pacote a ser processado pelo dispositivo (ROTHENBERG et al., 2011) e pode ser definido por vários parâmetros das camadas de rede, enlace ou transporte. Na Figura 3 são apresentados os campos dos pacotes que correspondem à entrada de fluxos. Entre esses campos, estão contidos: a porta de entrada, que é a representação numérica da interface pela qual o fluxo chegou no switch, as portas de origem e destino da camada de enlace, ID e prioridade VLAN, os IPs de origem e de destino, protocolo, tipo de serviço, e portas de origem e destino de transporte. Cada fluxo utiliza a combinação de campos necessária para atender ao objetivo, e neste trabalho foram utilizadas as portas de transporte do tráfego de VoIP (5060) e a instrução de encaminhamento para uma fila determinada.

## 2.3. Qualidade de Serviço (QoS)

Aplicações de internet estão constantemente crescendo e a demanda por recursos está cada vez maior. Alguns serviços requerem um alto nível de qualidade, como serviços de multimídia em tempo real. Havendo grande quantidade de perda de pacotes, ou um atraso muito grande, esses tipos de serviços tornam-se inutilizáveis.

A Qualidade de Serviço é geralmente uma característica de comunicação fim-a-fim, esse termo refere-se à capacidade de transmitir informações provendo qualidade para um

cliente (GOZDECKI et al., 2003). A qualidade de serviço depende das necessidades da aplicação, ou seja, do que ela requisita da rede a fim de que funcione bem, atendendo às necessidades do usuário. Estes requisitos são traduzidos em parâmetros indicadores do desempenho da rede como, por exemplo, o atraso máximo sofrido pelo tráfego da aplicação entre origem e destino.

Gozdecki et al. (2003) define como parâmetros de qualidade de serviço a taxa de transferência (vazão que pode ser alcançada), atraso (atraso sofrido pelos pacotes enquanto atravessam a rede), variação do atraso (variação do atraso sofrido por pacotes subsequentes) e a taxa de perda de pacotes (usualmente definida pela proporção do número de pacotes não entregues).

Ju e Bi (1998) caracterizam cinco classes de serviço: Taxa de bits constante (CBR), Taxa de bits variável em tempo real (RT-VBR), Taxa de bits variável em tempo não real (NRT-VBR), Taxa de bits não especificada (UBR) e Taxa de bits disponível (ABR).

Em aplicações com CBR, a taxa de transmissão é constante com o tempo. Essas aplicações são muito sensíveis às variações do tempo de atraso. Exemplos de aplicações CBR são o tráfego telefônico e videoconferência. NRT-VBR permite ao usuário enviar informações usando taxas de transmissão que variam com o tempo de acordo com o requisito da mensagem, RT-VBR é similar ao NRT-VBR, mas projetado para serviços sensíveis às variações de atraso. Os serviços UBR são aqueles que não especificam a taxa de bits, enquanto os ABR controlam o fluxo com base na taxa de bits. Vale ressaltar que este trabalho atende a classe de serviço CBR, específica para aplicações sensíveis ao atraso e exigem uma taxa de transmissão constante.

## 2.4. Voz sobre IP (VoIP)

VoIP (*Voice over Internet Protocol*) é o suporte pela rede de dados que garante a comunicação de dois clientes, e tem se tornado cada vez mais importante devido ao seu baixo custo e acesso econômico da rede (ROHLOFF; COUSINS; SUMOROK, 2016). Quando a voz é transmitida sobre uma rede IP, é necessário o uso de um *codec* de áudio para transformar os sinais de voz em pacotes de voz antes de fazer o transporte (CHANTHONG; WUTTIDITTACHOTTI; DAENGSI, 2015).

O *codec* é responsável por codificar e decodificar os sinais de transmissão e nesse trabalho foram selecionados dois para a realização dos experimentos: GSM e G.711 a-law. O codec GSM é amplamente utilizado, visto que usa uma taxa de transmissão de dados de

apenas 13Kbps (RAJ; MIGDAL; SINGH, 2001), enquanto o G.711 é mais usado em redes LAN pois exige uma taxa de 64Kbps (CHANTHONG; WUTTIDITTACHOTTI; DAENGSI, 2015). O codec G.711 possui os subtipos G.711 μ-law, usado principalmente nos EUA, Canadá e Japão, e G.711 a-law, este último foi escolhido para o trabalho por ser mais usado na Europa e no restante do mundo (CHANTHONG; WUTTIDITTACHOTTI; DAENGSI, 2015).

Goode (2002) define que um grande número de fatores está envolvido em realizar uma ligação VoIP de alta qualidade, entre eles está o codec utilizado na chamada, perda de pacotes, atraso, variação do atraso e arquitetura da rede. O modelo-E utiliza as métricas citadas por Goode (2002) para garantir que clientes estarão satisfeitos com a chamada, porém ele disponibiliza apenas o Fator R (*Rating Factor*), que varia de acordo com a qualidade da conversa.

O VoIPmonitor é uma ferramenta que tem como propósito analisar a qualidade de ligações VoIP baseada no uso do modelo-E (VOIPMONITOR, 2016), transformando o Fator R para estabelecer uma estimativa da opinião de consumidores (MOS - *Mean Opinion Score*), que foi a métrica escolhida como a mais apropriada para o trabalho, visto que é específica para avaliar ligações telefônicas ou chamadas de vídeo e se dá em um valor que varia de 1 até 5 (HERSENT; GUIDE; PETIT, 2002).

Tabela 1 - Classificação do MOS de acordo com o valor.

| Valor do MOS | Classificação |
|---|---|
| Acima de 4,3 | Excelente |
| Entre 4,0 e 4,3 | Alta qualidade |
| Entre 3,6 e 4,0 | Qualidade média |
| Entre 3,1 e 3,6 | Baixa qualidade |
| Entre 2,6 e 3,1 | Pobre |
| Abaixo de 2,6 | Não recomendada |

Fonte: Elaborada pela própria autora.

Como pode ser visualizado na Tabela 1, Toral e Torres (2005) classificam que uma taxa do MOS acima de 4,3 corresponde a melhor qualidade, em que usuários estão muito satisfeitos, uma taxa entre 4,0 e 4,3 corresponde a alta qualidade, taxa entre 3,6 e 4,0 é classificada em qualidade média, entre 3,1 e 3,6 corresponde a baixa qualidade, taxa entre 2,6 e 3,1 é considerada pobre e abaixo de 2,6 não é recomendada.

# 3. TRABALHOS RELACIONADOS

Araújo (2013) propõe um sistema para aprovisionamento de QoS em redes OpenFlow. Assim, nos momentos de pico de tráfego na rede, alguns usuários podem ter uma qualidade de serviço diferenciada de acordo com sua classe de serviço. Nesse trabalho, é construído um módulo de aplicação para o controlador Floodlight, o qual permite que classes de serviço sejam gerenciadas através da criação e remoção de novas classes e o módulo QoS se comunica com o núcleo do controlador através da API Java, fornecendo uma API REST para gerenciamento. Uma das principais diferenças entre o trabalho proposto e o trabalho de Araújo (2013), será a utilização de controladores e linguagens de programação diferentes. Em Araújo (2013), é usado o controlador Floodlight e a linguagem de programação Java, já no trabalho proposto, será utilizado o controlador POX, que usa linguagem de programação Python, o conceito desse controlador será explicado na seção 4.1.1.2.

O trabalho de Egilmez et al. (2012) propõe o OpenQoS, um novo controlador que permite QoS para entrega de multimídia em redes OpenFlow. Com o OpenQoS, é proposto um esquema de priorização baseado em roteamento, em que o tráfego de multimídia segue por um fluxo com QoS e os outros fluxos permanecem em seus caminhos normais. Embora os autores sugerirem várias melhorias, apenas a vazão disponível é usada como parâmetro para demonstrar os resultados.

No artigo de Civanlar et al. (2010), é proposto um controlador SDN que configura rotas de fluxos levando em consideração os requisitos de QoS para tráfego de SVC (Scalable Video Coding). Esse trabalho se esforça para que o roteamento de tráfego com QoS não apresente nenhuma perda de pacotes, o que é alcançado por meio da introdução de mais perda para o tráfego de melhor esforço.

O artigo de Silva et al. (2013) propõe e avalia uma arquitetura, denominada RME (Resource Mediation Engine), que provê suporte à QoS fim-a-fim em redes virtuais baseadas no OpenFlow. É usada uma arquitetura para provisionamento de QoS, com um servidor e um cliente em domínios diferentes com RMEs gateway, um RME switch e um RME VLAN, que controla os gateways e o switch. Em Silva et al. (2013), os gateways e o switch são gerenciados pelo controlador RME VLAN, o qual é baseado no controlador NOX, que usa a linguagem de programação C++ e está disponível apenas para o sistema operacional Linux, enquanto o POX, que será usado nesse trabalho, está disponível para Windows, MacOS e Linux (ARAÚJO, 2013).

# 4. EXPERIMENTOS E RESULTADOS

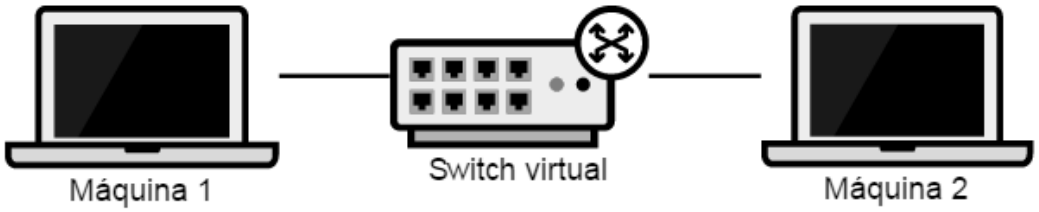
Nessa seção, será apresentada a forma como os experimentos foram realizados, além dos resultados colhidos e analisados de cada cenário de acordo com os fatores e níveis definidos.

## 4.1. Configuração do Experimento

Nesta seção, é apresentada a forma como os experimentos foram realizados. Também são apresentados os fatores, níveis e a métrica utilizada.

A configuração dos cenários foi feita com duas máquinas físicas executando o *Ubuntu Desktop* 12.04, pois ao usar outras versões, houveram divergências de compatibilidade com os softwares usados. As máquinas foram conectadas por meio de interface de rede cabeada com interferência de um switch virtual, como pode ser visto na Figura 4.

Figura 4 - Topologia do experimento.




Fonte: Elaborada pela própria autora.

A máquina principal possuía 6GB de memória RAM e executava os seguintes softwares:

- Asterisk (Servidor *VoIP*);
- *VoipMonitor* (Utilizado para coletar os dados das amostras);
- *OpenVSwitch* (Switch virtual);
- POX (Controlador SDN);
- Zoiper (Cliente *VoIP*);
- Iperf (Servidor do gerador de tráfego de segundo plano).

A máquina secundária possuía 2GB de memória RAM e executava os seguintes softwares:

- Zoiper (Cliente *VoIP*);
- Iperf (Cliente do gerador de tráfego de segundo plano).

Foram definidos quatro fatores para a análise de desempenho, como pode ser visto na Tabela 2. Esses fatores são: Controlador SDN, *Codecs,* número de fontes de tráfego de segundo plano e Protocolo de transporte do tráfego de segundo plano. Os níveis do fator Controlador SDN foram dois: sua utilização ou não. Em relação aos *Codecs*, foram escolhidos dois níveis: G.711 a-law e GSM. Os números de fontes do tráfego de segundo plano definidos foram: 1, 10 e 100 fontes e os protocolos destas fontes de tráfego de segundo plano definidos foram TCP e UDP.

Tabela 2 - Fatores e níveis definidos para análise de desempenho.

| Fator | Níveis | | |
|---|---|---|---|
| Controlador SDN | Com controlador | Sem controlador | |
| Codec da ligação | G.711 a-law | GSM | |
| Número de fontes do tráfego de segundo plano | 1 | 10 | 100 |
| Protocolo de transporte do tráfego de segundo plano | TCP | UDP | |

Fonte: Elaborada pela própria autora.

Os experimentos foram realizados por meio de interface cabeada. A Máquina 1 possuía uma interface de rede *bridge*, que foi usada para criação de duas filas no *switch* virtual. As duas filas criadas no *switch* possuíam as mesmas capacidades de taxa de transferência de dados (100 Mbps).

Foi feita uma ligação para cada combinação diferente de níveis com cerca de 600 segundos, em paralelo ao tráfego de segundo plano, e as amostras foram coletadas a cada 8 segundos, descartando as primeiras e as últimas amostras de cada cenário. Todos os experimentos foram executados em dois cenários diferentes: com e sem controlador SDN. No nível do experimento sem controlador, todos os tipos de tráfego, tanto o *VoIP* quanto o de

segundo plano, usavam uma mesma porta do *switch*. A única diferença para os experimentos com o controlador, foi o uso do POX que influenciava para que todo o tráfego *VoIP* fosse para uma fila prioritária do *switch* virtual, percorrendo um caminho diferente do tráfego de segundo plano, é importante ressaltar que as linhas de código que fazem essa diferenciação são 84-89, e podem ser analisadas no ANEXO A – Código do *switch* com opções de fluxos diferentes de acordo com a classe do serviço, modificado do *switch l2_learning*, disponível na secção *forwarding* do POX. Um tutorial de como configurar o cenário com o controlador POX está disponível no APÊNDICE A – Configuração do ambiente com o controlador POX.

A métrica que foi escolhida como a mais apropriada para o trabalho foi o MOS (*Mean Opinion Score*), específica para avaliar ligações telefônicas ou chamadas de vídeo, que se dá em um valor que varia de 1 até 5 (HERSENT; GUIDE; PETIT, 2002). Toral e Torres (2005) classificam que uma taxa do MOS acima de 4,3 corresponde a melhor qualidade, em que usuários estão muito satisfeitos, uma taxa entre 4,0 e 4,3 corresponde a alta qualidade, taxa entre 3,6 e 4,0 é classificada em qualidade média, entre 3,1 e 3,6 corresponde a baixa qualidade, taxa entre 2,6 e 3,1 é considerada pobre e abaixo de 2,6 não é recomendada.

## 4.2. Resultados

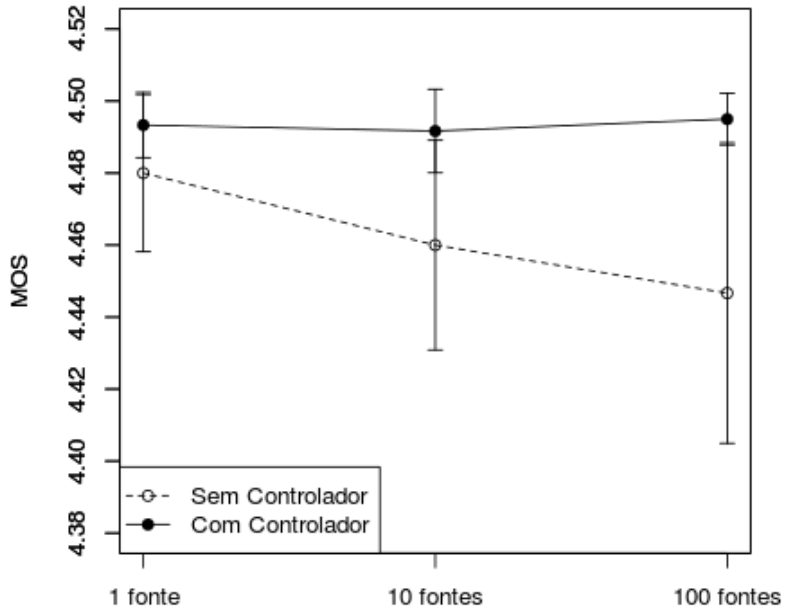
Nessa seção, serão apresentados os resultados das análises de desempenho em quatro partes. Na primeira parte, será apresentado o resultado para o *codec* G.711 a-law usando o protocolo de transporte TCP para o tráfego de segundo plano, na segunda será apresentado o resultado para o *codec* G.711 a-law utilizando o protocolo UDP. Na terceira parte, os resultados do *codec* GSM fazendo uso do protocolo TCP como transporte para o tráfego de plano de fundo são mostrados e por último serão apresentados os resultados com o uso do *codec* GSM e UDP como protocolo de transporte para o tráfego de segundo plano. Nesses experimentos, foram coletadas 30 amostras e os cálculos dos intervalos de confiança foram efetuados com nível de confiança de 95%.

### 4.2.1. Avaliação do G.711 a-law usando protocolo de transporte TCP para o tráfego de segundo plano

Para o cenário com o *codec* G.711 a-law usando TCP como protocolo de transporte do tráfego de plano de fundo, sem a influência do controlador POX, a média do MOS obtida decaiu um pouco conforme o nível de clientes enviando tráfego de segundo plano aumentou, como pode-se observar na Figura 5. Quando o controlador POX foi

utilizado, a média do MOS manteve-se estável a medida que o número de fontes de tráfego de segundo plano aumentou.

Figura 5 - G.711 a-law usando protocolo TCP para o tráfego de segundo plano.



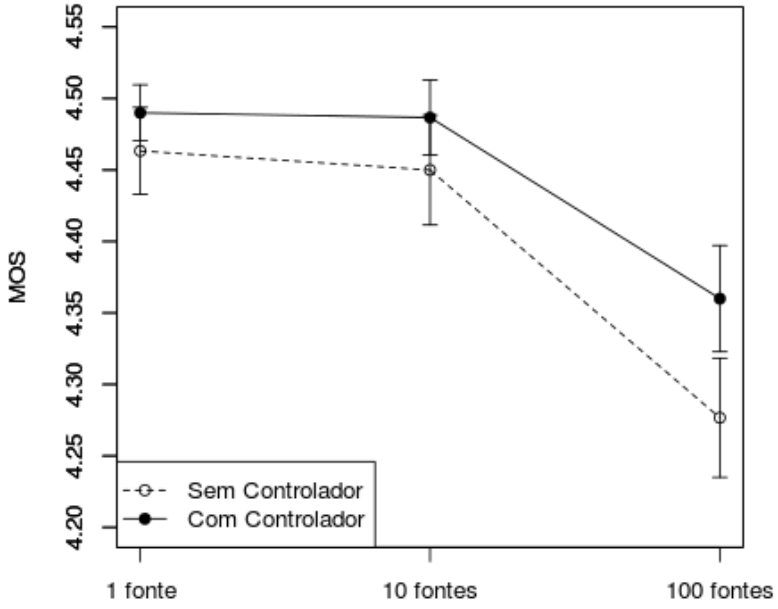

Fonte: Elaborada pela própria autora.

Em todos os cenários usando o TCP, independente do uso do controlador ou do número de fontes enviando tráfego de segundo plano, a soma dos dados enviados manteve-se sempre entre 6,55 e 6,58GB, enquanto a vazão permaneceu entre 93,7 e 94,0 Mbps.

### 4.2.2. Avaliação do G.711 a-law usando protocolo de transporte UDP para o tráfego de segundo plano

Em relação ao *codec* G.711 a-law fazendo uso do protocolo UDP para transportar o tráfego de plano de fundo, foi possível perceber que à medida que o número de clientes enviando tráfego simultaneamente à ligação aumentou, a média do MOS decaiu para os cenários com ou sem uso do controlador POX, porém, como pode ser verificado na Figura 6, não houve intersecção do intervalo de confiança entre os cenários com 100 fontes de tráfego concorrente.

Figura 6 - G.711 a-law usando protocolo UDP para o tráfego de segundo plano.



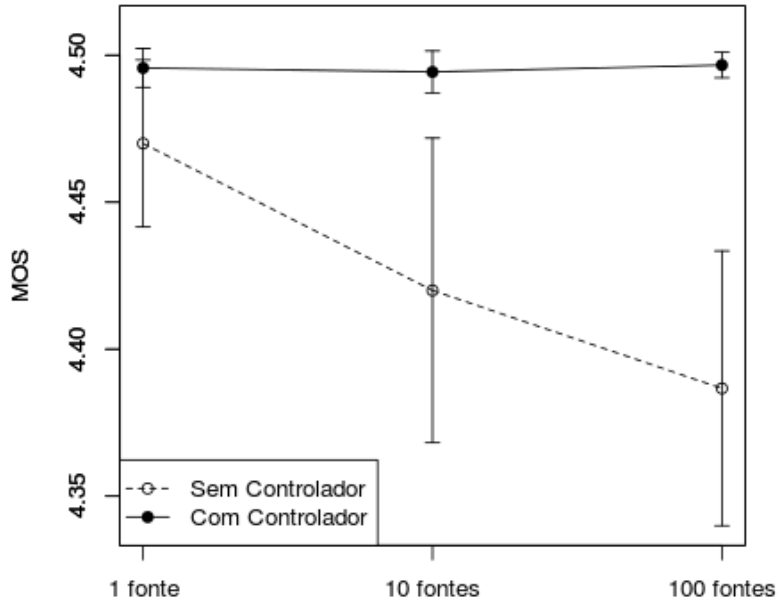

Fonte: Elaborada pela própria autora.

Nos dois cenários usando 1 fonte de tráfego concorrente, a soma de todos os dados enviados foi de 75,0 MB, com vazão em 1,05 Mbps. Em relação aos dois cenários com 10 clientes do Iperf, a soma dos dados foi de 750 MB, com vazão em 10,5 Mbps. Quanto aos cenários com 100 clientes, a soma dos dados enviados por todas as fontes foi de 6,68 GB para o contexto sem controlador, com vazão em 95,5 Mbps, e 6,89 GB e vazão em 95,7 Mbps, com uso do POX.

### 4.2.3. Avaliação do GSM usando protocolo de transporte TCP para o tráfego de segundo plano

Para o codec GSM usando o protocolo TCP como transporte para o segundo plano e com o uso do controlador POX, foi possível constatar que o MOS se manteve estável conforme as fontes do tráfego de plano de fundo foram aumentando, como pode-se verificar na Figura 7, contudo, sem o uso do controlador, foi possível perceber que o MOS decaiu à medida que as fontes aumentaram.

Figura 7 - GSM usando protocolo TCP para o tráfego de segundo plano.



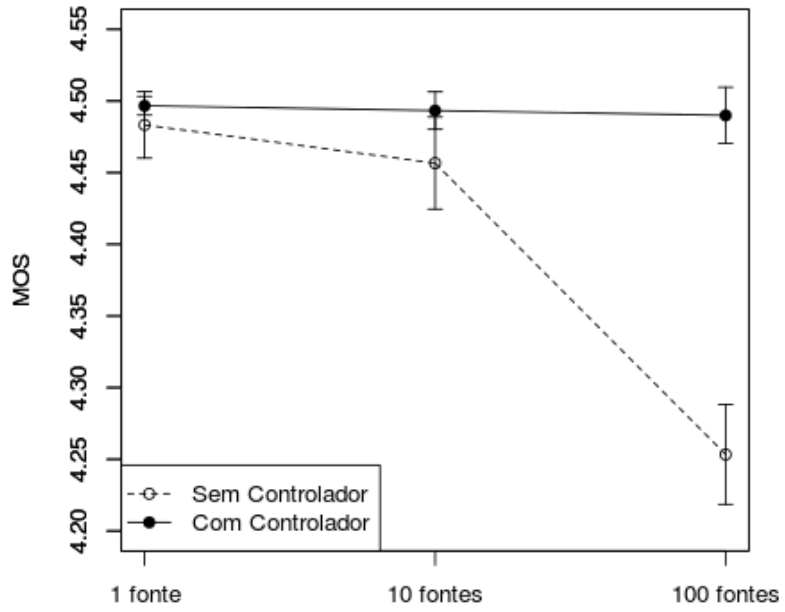

Fonte: Elaborada pela própria autora.

Em todos os contextos, usando ou não o controlador POX e independentemente da quantia de fontes de tráfego concorrente, a vazão se manteve entre 93,7 e 94,1 Mbps, enquanto a soma de todos os dados preservou-se entre 6,55 e 6,58 GB.

### 4.2.4. Avaliação do GSM usando protocolo de transporte UDP para o tráfego de segundo plano

Fazendo uso do protocolo de transporte UDP para o tráfego de plano de fundo, juntamente com o *codec* GSM, foi possível verificar preservação da média do MOS ao se usar o controlador POX, o que pode ser analisado na Figura 8. Em contrapartida, a média do MOS, nas situações sem o uso do controlador, decaiu conforme as fontes de tráfego de plano de fundo aumentaram.

Figura 8 - GSM usando protocolo UDP para o tráfego de segundo plano.




Fonte: Elaborada pela própria autora.

É importante ressaltar que a vazão do tráfego de segundo plano para os cenários com 1 cliente do Iperf foi de 1,05 Mbps, e os dados enviados totalizaram 75,0 MB para cada caso. Na situação em que foram usados 10 clientes do Iperf, a vazão e a soma dos dados enviados também se mantiveram as mesmas para os cenários com uso ou não do controlador POX, ficando no valor de 10,5 Mbps e 750 MB, respectivamente. Em relação às situações com 100 fontes de tráfego de plano de fundo, a vazão ficou em 95,4 Mbps para o caso sem controlador e 95,7 Mbps para o cenário com o uso do POX, enquanto os dados enviados ficaram em 6,57 GB para a situação sem uso do controlador e 6,89 GB quando se fez o uso do POX.

## 5. CONSIDERAÇÕES FINAIS

Por meio desse trabalho, espera-se melhorar a qualidade percebida pelo usuário em relação ao tráfego de *VoIP* através do tratamento desse tipo de tráfego com QoS alterada no nível de fluxo, e consequentemente, espera-se amenizar o problema relacionado à obtenção de Qualidade de Serviço em Redes Definidas por Software, o que poderá ajudar os pesquisadores da área.

Com a análise de desempenho, foi possível verificar com nível de confiança de 95% que em todos os casos em que haviam apenas 1 ou 10 fontes de tráfego concorrente, no uso dos protocolos TCP e UDP e independente do *codec* utilizado, a média do MOS da ligação se manteve acima de 4,3, considerada excelente, seguindo as faixas citadas por Toral e Torres (2005), da mesma forma nos cenários em que se fez uso do protocolo TCP, com 100 fontes de tráfego de plano de fundo.

Contudo, quando foram usadas 100 fontes para gerar o tráfego de segundo plano, concomitante ao uso do protocolo UDP, tanto no uso do *codec* G.711 a-law quanto do GSM, não houve intersecção dos intervalos de confiança entre os casos em que o controlador foi utilizado e deixou de ser utilizado, concluindo que o uso do controlador POX se fez eficiente, mantendo a média da qualidade em um nível considerado excelente segundo a escala de Toral e Torres (2005), enquanto a média do MOS sem o uso do POX se manteve em nível de alta qualidade, entre 4,0 e 4,3, segundo as faixas de Toral e Torres (2005).

De acordo com os resultados obtidos, pode-se ressaltar que o uso do controlador SDN é indicado para ambientes com um elevado tráfego de plano de fundo, e caso a média do MOS continue seguindo o padrão apresentado, é possível esperar que em um ambiente com mais fontes de tráfego de segundo plano o uso do controlador destaque uma maior diferença na qualidade da chamada. Como possível trabalho futuro, pode-se apontar a realização de experimentos fazendo uso do *codec* iLBC, cuja adoção está crescendo na Internet.

## REFERÊNCIAS

ABBASI, T. et al. **A comparative study of the SIP and IAX VoIP protocols**. In: Canadian Conference on Electrical and Computer Engineering, 2005.

**ABOUT POX**. 2014. Disponível em: <http://www.noxrepo.org/pox/about-pox/>.

ARAÚJO, M. R. A. G. **Uma Abordagem para Aprovisionamento de QoS em Redes Definidas por Software baseadas em OpenFlow**. Trabalho apresentado ao Programa de Graduação em Engenharia da Computação da Universidade Federal de Pernambuco, 2013.

CHANTHONG, R.; WUTTIDITTACHOTTI, P.; DAENGSI, T. **A study of G.711 and ILBC over WLAN 802.11n with EDCA.** In: International Conference on Information Networking (ICOIN), 2015.

CIVANLAR, S. et al. **A QoS-Enabled OpenFlow Environment for Scalable Video Streaming**. In: IEEE Globecom 2010 Workshop on Network of the Future (FutureNet-III), 2010, Miami, p. 351–356.

EGILMEZ, H. E. et al. **OpenQoS: An OpenFlow controller design for multimedia delivery with end-to-end Quality of Service over Software-Defined Networks**. In: Signal & Information Processing Association Annual Summit and Conference (APSIPA ASC), 2012, Los Angeles.

FEAMSTER, N.; GAO, L.; REXFORD, J. **How to lease the Internet in your spare time**. In: ACM SIGCOMM Computer Communication Review, 37, 2007, p.61–64.

GOZDECKI, J. et al. **Quality of service terminology in IP networks**. In: IEEE Communications Magazine, vol.41, 2003, s.l., p.153–159.

GOODE, B. **Voice over Internet protocol (VoIP).** In: Proceedings of the IEEE, 90, 2002.

HERSENT, O.; GUIDE, D.; PETIT, J. **Telefonia IP**. São Paulo: Prentice Hall, 2002.

JU, S.; BI, G. **A algorithm of dynamically assigning upstream bandwidth of hybrid fiber-coax networks.** In: IEEE Transactions on Broadcasting, 44, 1998, p. 330-335.

LANTZ, B.; HELLER, B.; MCKEOWN, N. **A network in a laptop: rapid prototyping for software-defined networks**. In ACM SIGCOMM Workshop on Hot Topics in Networks, 9th, 2010.

MATTOS, D. M. F.; DUARTE, O. C. M. B. **QFlow: Um Sistema com Garantia de Isolamento e Oferta de Qualidade de Serviço para Redes Virtualizadas**. In: Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC), 30, 2012, Ouro Preto, p. 536–549.

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

NDE, G. N.; KHONDOKER, R. **SDN testing and debugging tools: A survey**. In: International Conference on Informatics, Electronics and Vision (ICIEV), 5th, 2016.

NUNES, B. A. A. et al. **A Survey of Software-Defined Networking: Past, Present, and Future of Programmable Networks**. In: IEEE Communications Surveys & Tutorials, 2014.

**OPENFLOW Switch Specification**. In: Open Networking Foundation, 2009. Disponível em: <https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-spec-v1.0.0.pdf> Acesso em: jun, 2014.

**OPENFLOW Switch Specification.** In: Open Networking Foundation, 2011. Disponível em: < http://archive.openflow.org/documents/openflow-spec-v1.1.0.pdf > Acesso em: dez, 2016.

PAUL, S.; PAN, J.; JAIN, R. **Architectures for the future networks and the next generation internet: A survey**. In: Computer. Communication, 2011, p.34:2–42.

**POX** Wiki. 2016. Disponível em <https://openflow.stanford.edu/display/ONL/POX+Wiki>.

RAJ, B.; MIGDAL, J.; SINGH, R. **Distributed speech recognition with codec parameters.** In: IEEE Workshop on Automatic Speech Recognition and Understanding, 2001.

RATNASAMY, S.; SHENKER, S.; MCCANNE, S. **Towards an evolvable Internet architecture**. In: ACM SIGCOMM Computer Communication Review, 35, 2005, p.313–324.

ROHLOFF, K.; COUSINS, D.; SUMOROK, D. **Scalable, Practical VoIP Teleconferencing with End-to-End Homomorphic Encryption.** In: IEEE Transactions on Information Forensics and Security, 2016, p.1-1.

ROTHENBERG, C. E. et al. **OpenFlow e redes definidas por software: um novo paradigma de controle e inovação em redes de pacotes**. In: Cadernos CPqD Tecnologia, Vol 7. No 1, 2011.

SILVA, P. D. et al**. Uma Arquitetura para o Aprovisionamento de QoS Interdomínios em Redes Virtuais baseadas no OpenFlow**. In: Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC), 31, 2013, Brasília, p. 893–906.

**SOFTWARE defined networking: the new norm for networks**. In: Open Networking Foundation, 2012, Palo Alto, CA. Disponível em: <https://www.opennetworking.org/images/stories/downloads/sdn-resources/white-papers/wp-sdn-newnorm.pdf> Acesso em: mar, 2014.

TORAL, C. H.; TORRES, R. D. **Traffic analysis for IP telephony**. In: International Conference on Electrical and Electronics Engineering, 2nd, p.136 - 139, Set. 2005.

VENILA, G.; SHALINI, N. S.; MANIKADAN, M. S. K. **Performance analysis of VoIP spoofing attacks using classification algorithms.** In: Applications and Innovations in Mobile Computing (AIMoC), 2014.

VISOOTTIVISETH, V.; BUREENOK, N. **Performance Comparison of ISATAP Implementations on FreeBSD, RedHat, and Windows 2003**. In: International Conference on Advanced Information Networking and Applications - Workshops.

**VOIPMONITOR**. 2016. Disponível em: < http://www.voipmonitor.org/>.

# APÊNDICES

## APÊNDICE A – Configuração do ambiente com o controlador POX

Criar interface de rede *bridge* e configurar porta entre as interfaces br0 e eth0.

***ovs-vsctl add-br br0***
***ovs-vsctl add-port br0 eth0***
***ifconfig br0 <IP da máquina>/<máscara>***
***ifconfig eth0 0.0.0.0***

Comando para configurar uma porta de QoS na interface *bridge* e criação das duas filas no *OpenVSwitch* com as configurações de taxas máximas e mínimas das filas em bps.

***ovs-vsctl -- set Port br0 qos=@newqos -- ***
***--id=@newqos create QoS type=linux-htb other-config:max-rate=100000000 queues=0=@q0,1=@q1 -- ***
***--id=@q0 create Queue other-config:min-rate=100000000 other-config:max-rate=100000000 -- ***
***--id=@q1 create Queue other-config:min-rate=100000000 other-config:max-rate=100000000***

Conectar OVS com controlador.

***ovs-vsctl set-controller br0 tcp:<IP do controlador>:<porta>***

Iniciar o controlador.

***~/pox$ ./pox.py log.level --DEBUG forwarding.switchqos***

Iniciar o servidor de tráfego de segundo plano.

***iperf –s*** ou ***iperf –s –u*** (Caso escolha usar o protocolo UDP).

Iniciar o cliente de tráfego de segundo plano

***iperf -c <IP do servidor> -P<número de clientes paralelos>*** ou ***iperf -c <IP do servidor> -P<número de clientes paralelos> -u*** (Caso escolha usar o protocolo UDP)***.***

Após isso, deve-se configurar os clientes VoIP com o *codec* que se pretende usar e fazer uma ligação normalmente.

## ANEXOS

### ANEXO A – Código do *switch* com opções de fluxos diferentes de acordo com a classe do serviço, modificado do *switch l2_learning*, disponível na secção *forwarding* do POX

```
1 from pox.core import core
2 import pox.openflow.libopenflow_01 as of
3 from pox.lib.util import dpid_to_str
4 from pox.lib.util import str_to_bool
5 import time
6
7 log = core.getLogger()
8
9 _flood_delay = 0
10
11 class LearningSwitch (object):
12
13 def __init__ (self, connection, transparent):
14   self.connection = connection
15   self.transparent = transparent
16
17   self.macToPort = {}
18
19   connection.addListeners(self)
20
21   self.hold_down_expired = _flood_delay == 0
22
23  def _handle_PacketIn (self, event):
24    packet = event.parsed
25
26   def flood (message = None):
27     msg = of.ofp_packet_out()
28
29     if time.time() - self.connection.connect_time >= _flood_delay:
30
31       if self.hold_down_expired is False:
32         self.hold_down_expired = True
33         log.info("%s: Flood hold-down expired -- flooding",
34             dpid_to_str(event.dpid))
35
36       if message is not None: log.debug(message)
37       msg.actions.append(of.ofp_action_output(port = of.OFPP_FLOOD))
38       else:
39       pass
40       msg.data = event.ofp
41       msg.in_port = event.port
42       self.connection.send(msg)
```

```
43
44    def drop (duration = None):
45      if duration is not None:
46        if not isinstance(duration, tuple):
47          duration = (duration,duration)
48        msg = of.ofp_flow_mod()
49        msg.match = of.ofp_match.from_packet(packet)
50        msg.idle_timeout = duration[0]
51        msg.hard_timeout = duration[1]
52        msg.buffer_id = event.ofp.buffer_id
53        self.connection.send(msg)
54      elif event.ofp.buffer_id is not None:
55        msg = of.ofp_packet_out()
56        msg.buffer_id = event.ofp.buffer_id
57        msg.in_port = event.port
58        self.connection.send(msg)
59
60    self.macToPort[packet.src] = event.port
61
62    if not self.transparent:
63      if packet.type == packet.LLDP_TYPE or packet.dst.isBridgeFiltered():
64        drop()
65        return
66
67    if packet.dst.is_multicast:
68      flood()
69    else:
70      if packet.dst not in self.macToPort:
71       flood("Port for %s unknown -- flooding" % (packet.dst,)) # 4a
72      else:
73        port = self.macToPort[packet.dst]
74        if port == event.port:
75          log.warning("Same port for packet from %s -> %s on %s.%s.  Drop."
76              % (packet.src, packet.dst, dpid_to_str(event.dpid), port))
77          drop(10)
78          return
79        log.debug("installing flow for %s.%i -> %s.%i" %
80                (packet.src, event.port, packet.dst, port))
81        msg = of.ofp_flow_mod()
82        msg.match = of.ofp_match.from_packet(packet, event.port)
83
84 #Seção do código que foi modificada de acordo com a porta do serviço.
85
86        if msg.match.tp_src == 5060 or msg.match.tp_dst == 5060:
87          msg.actions.append(of.ofp_action_enqueue(port = port,queue_id=1))
88        else:
89          msg.actions.append(of.ofp_action_enqueue(port = port, queue_id=0))
90
91        msg.data = event.ofp
92        self.connection.send(msg)
```

```
93
94 class l2_learning (object):
95   def __init__ (self, transparent):
96     core.openflow.addListeners(self)
97     self.transparent = transparent
98
99   def _handle_ConnectionUp (self, event):
100     log.debug("Connection %s" % (event.connection,))
101     LearningSwitch(event.connection, self.transparent)
102
103 def launch (transparent=False, hold_down=_flood_delay):
104   try:
105     global _flood_delay
106     _flood_delay = int(str(hold_down), 10)
107     assert _flood_delay >= 0
108   except:
109     raise RuntimeError("Expected hold-down to be a number")
110
111  core.registerNew(l2_learning, str_to_bool(transparent))
```