Este trabalho tem como objetivo apresentar uma solução baseada em SDN para reduzir a utilização de recursos, como a utilização de CPU do servidor VoIP, utilização de largura de banda no segmento de rede em que se encontra o servidor VoIP, em ambiente LAN. Foram emulados cenários com a utilização de equipamentos físicos reais, sem o uso de SDN e com o uso de SDN quantificando a utilização de CPU, largura de banda e quantidade de dados trafegados em um servidor VoIP. Os resultados experimentais obtidos indicam redução no consumo dos recursos citados, com ganhos reais de implantação da solução.

Orientador: Adriano Fiorese

Coorientador: Charles Christian Miers

Joinville, 2017

ANO
2017

PAULO ROBERTO VIEIRA JUNIOR | UMA ABORDAGEM BASEADA EM REDES DEFINIDAS POR SOFTWARE PARA REDUÇÃO DO CONSUMO DE RECURSOS DE SERVIÇOS DE VOZ SOBRE IP

UNIVERSIDADE DO ESTADO DE SANTA CATARINA – UDESC
CENTRO DE CIÊNCIAS TECNOLÓGICAS – CCT
PROGRAMA DE PÓS GRADUAÇÃO EM COMPUTAÇÃO APLICADA

DISSERTAÇÃO DE MESTRADO

UMA ABORDAGEM BASEADA EM REDES DEFINIDAS POR SOFTWARE PARA REDUÇÃO DO CONSUMO DE RECURSOS DE SERVIÇOS DE VOZ SOBRE IP

PAULO ROBERTO VIEIRA JUNIOR

JOINVILLE, 2017

UNIVERSIDADE DO ESTADO DE SANTA CATARINA - UDESC
CENTRO DE CIÊNCIAS TECNOLÓGICAS - CCT
MESTRADO EM COMPUTAÇÃO APLICADA

PAULO ROBERTO VIEIRA JUNIOR

UMA ABORDAGEM BASEADA EM REDES DEFINIDAS POR SOFTWARE PARA REDUÇÃO DE CONSUMO DE RECURSOS DE SERVIÇOS VOZ SOBRE IP

JOINVILLE

2017

PAULO ROBERTO VIEIRA JUNIOR

UMA ABORDAGEM BASEADA EM REDES DEFINIDAS POR SOFTWARE PARA REDUÇÃO DE CONSUMO DE RECURSOS DE SERVIÇOS VOZ SOBRE IP

Dissertação submetida ao Programa de Pós-Graduação em Computação Aplicada do Centro de Ciências Tecnológicas da Universidade do Estado de Santa Catarina, para a obtenção do grau de Mestre em Computação Aplicada.

Orientador: Prof. Dr. Adriano Fiorese
Coorientador: Prof. Dr. Charles Christian Miers

JOINVILLE

2017

Ficha catalográfica elaborada pelo(a) autor(a), com
auxílio do programa de geração automática da
Biblioteca Setorial do CCT/UDESC

Vieira jr, Paulo Roberto
Uma Abordagem baseada em Redes Definidas por Software para redução de consumo de recursos de serviços Voz Sobre IP / Paulo Roberto Vieira jr. - Joinville , 2017.
104 p.

Orientador: Adriano Fiorese
Co-orientador: Charles Christian Miers
Dissertação (Mestrado) - Universidade do Estado de Santa Catarina, Centro de Ciências Tecnológicas, Programa de Pós-Graduação em Computação Aplicada, Joinville, 2017.

1. Sistemas Computacionais. 2. Redes Definidas por Software. 3. VoIP. I. Fiorese, Adriano. II. Miers, Charles Christian. , .III. Universidade do Estado de Santa Catarina, Centro de Ciências Tecnológicas, Programa de Pós-Graduação em Computação Aplicada. IV. Título.

Uma Abordagem Baseada em Redes Definidas por Software para Redução de Consumo de Recursos de Serviços Voz sobre IP

por

Paulo Roberto Vieira Junior

Esta dissertação foi julgada adequada para obtenção do título de

Mestre em Computação Aplicada

Área de concentração em "Ciência da Computação,
e aprovada em sua forma final pelo

CURSO Mestrado Acadêmico em Computação Aplicada
CENTRO DE CIÊNCIAS TECNOLÓGICAS DA
UNIVERSIDADE DO ESTADO DE SANTA CATARINA.

Banca Examinadora:

Prof. Dr. Charles Christian Miers
CCT/UDESC (Coorientador/Presidente)

Prof. Dr. Guilherme Piêgas Kollovski
CCT/UDESC

Prof. Dr. Marcos Dias de Assunção
INRIA - Lion - França

Joinville,SC, 25 de agosto de 2017.

Dedico este trabalho aos meus familiares, amigos, colegas e professores que me acompanharam, incentivaram, tiveram paciência e me deram forças nessa trajetória.

AGRADECIMENTOS

Ao Prof. Adriano Fiorese por sua dedicação, amizade e apoio no desenvolvimento desta dissertação.

Aos Professores Guilherme Piêgas Koslovski e Charles Christian Miers pelos comentários, sugestões e apoio durante o desenvolvimento deste trabalho.

Aos amigos Andrei Carniel, Luís Felipe Bilecki, Tathiana Duarte do Amarante, Ana Carolina Tomé Klock, Pedro Henrique Narloch, Anderson Henrique da Silva Marcondes, Emanoeli Madalosso, Adelaine Franciele Gelain, Marina Silva Fouto e todos que estiveram ao meu lado dando apoio e proporcionando momentos de alegria.

À UDESC pela oportunidade concedida, assim como pela disponibilidade do uso do laboratório para o desenvolvimento e testes do trabalho proposto.

À minha família por compreender os momentos que estive ausente e em especial à Elaini Simoni Angelotti, que sempre esteve ao meu lado apoiando e fazendo com que eu não desistisse deste trabalho, mesmo nos momentos mais difíceis sempre me ajudou a buscar uma solução e forças para continuar.

Finalmente, a todos que, direta ou indiretamente, prestaram alguma contribuição e apoio ao desenvolvimentos desta dissertação.

“O que agora é provado foi apenas uma vez imaginado”

Willian Blake

RESUMO

Já faz algum tempo que as telecomunicações vem se adaptando ao paradigma de comutação de pacotes. Esse paradigma, traz como benefícios a integração dos serviços de voz e dados, permitindo o surgimento e a disponibilização de Voz sobre o Protocolo Internet (*Voice over Internet Protocol* - VoIP). Assim, através da tecnologia VoIP, é possível que as pessoas se comuniquem com voz utilizando a Internet como meio de transmissão. Já na área de redes de computadores um dos temas que ganhou atenção foram as Redes Definidas por Software (SDN) que difundiram o desacoplamento entre o plano de controle e o plano de dados. Chamadas VoIP entre clientes que possuem o mesmo Codec consomem recursos de rede e do servidor VoIP. Fatores como configuração do servidor VoIP ou a localização dos clientes na rede, fazem com que os pacotes de voz trafeguem pelo servidor VoIP, processo que consome CPU e largura de banda entre outros recursos. Este trabalho tem como objetivo apresentar uma solução baseada em SDN para reduzir a utilização de recursos, como a utilização de CPU do servidor VoIP, utilização de largura de banda no segmento de rede em que se encontra o servidor VoIP, em ambiente LAN. Foram emulados cenários com a utilização de equipamentos físicos reais, sem o uso de SDN e com o uso de SDN quantificando a utilização de CPU, largura de banda e quantidade de dados trafegados em um servidor VoIP. Nesses cenários foram realizadas chamadas VoIP simultâneas entre clientes para que fosse possível realizar as medições. Os resultados experimentais obtidos indicam redução no consumo dos recursos citados, com ganhos reais de implantação da solução.

Palavras-chaves: redes de computadores, Redes Definidas por Software, Voz sobre IP, Asterisk.

ABSTRACT

For some time now, telecommunications has been adapting to the packet switching paradigm. This paradigm brings as benefits the integration of voice and data services, allowing the emergence and availability of Voice over Internet Protocol (VoIP). Thus, through VoIP technology, it is possible for people to communicate with voice using the Internet as a means of transmission. Software Defined Networking (SDN) has been disseminating the decoupling between control and data plane in network technologies. VoIP calls between clients that have the same Codec consume network and VoIP server resources. Factors such as configuring the VoIP server or the location of clients on the network, cause voice packets to be switched through the VoIP server. This process consumes CPU and bandwidth among other features. This work aims to present an SDN-based solution to reduce the use of resources, such as the CPU utilization of the VoIP server, the use of bandwidth in the network segment in which the VoIP server is located, in LAN environments. Therefore, this presents a management approach based on SDN to reduce the use of resources in a VoIP server. Scenarios were emulated with and without the use of SDN to assess CPU utilization and bandwidth used in a VoIP server. The experimental results obtained indicate reduction on the consumption of the mentioned resources.

Key-words: Computer Networks, Software Defined Networking, Voice over IP, Asterisk.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

ACL *Access Control List*

ACELP *Algebraic Code Excited Linear Prediction*

ADPCM *Adaptive Differential Pulse Code Modulation*

API *Application Programming Interface*

ARVS *Adaptive Routing Approach for Video Streaming*

CD Cliente Destino

CO Cliente Origem

Codec *Coder-Decoder*

CPU *Central Processing Unit*

GPL *General Public License*

GSM *Global System for Mobile*

HTTP *Hypertext Transfer Protocol*

iMOS *Intermediate Mean Opinion Score*

ITU-T *International Telecommunication Union-Telecommunication*

LAN *Local Area Network*

MAC *Media Access Control*

MAN *Metropolitan Area Network*

MOS *Mean Opinion Score*

MPC-MLQ *Multi Pulse-Maximum Likelihood Quantization*

NAT *Network Address Translation*

OpenWRT *Open Wireless Router*

OVS *Open Virtual Switch*

PABX *Private Automatic Branch Exchange*

PC *Personal Computer*

PCM *Pulse Code Modulation*

PPS *Packets per second*

PSTN *Public Switched Telephone Network*

QoE *Quality of Experience*

QoS *Quality of Service*

RAM *Random Access Memory*

REST *Representational State Transfer*

RFC *Request For Comments*

RR *Receiver Report*

RTCP *Realtime Transport Control Protocol*

RTP *Realtime Transport Protocol*

SDES *Source Description*

SDN *Software Defined Networking*

SDP *Session Description Protocol*

SIP *Session Initiation Protocol*

SR *Sender Report*

TCP *Transmission Control Protocol*

UDP *User Datagram Protocol*

VLAN *Virtual Local Area Network*

VoIP *Voice over Internet Protocol*

WAN *Wide Area Network*

WAV *Waveform Audio Format*

WLAN *Wireless Local Area Network*

SUMÁRIO

1 INTRODUÇÃO

Vive-se em uma sociedade na qual quase tudo encontra-se conectado a Internet, e uma informação pode ser acessível de praticamente qualquer lugar ou dispositivo. As redes de computadores fazem parte do dia a dia da população (*e.g.,* casas, empresas, universidades, escolas, hospitais, *etc.*), sendo que para que a informação trafegue nessas redes é necessário uma infraestrutura composta por equipamentos como *switches*, roteadores e pontos de acesso sem fio, entre outros.

Existem vários tipos de redes de computadores (*e.g., Local Area Network* (LAN), *Metropolitan Area Network* (MAN), *Wide Area Network* (WAN) e *Wireless Local Area Network* (WLAN), entre outras), e a técnica de comutação de pacotes de dados é a base de vários tipos de redes permitindo que pacotes endereçados a diferentes destinos compartilhem o mesmo meio de comunicação, também chamado de enlace. Os pacotes[1] são enfileirados em um *buffer* e transmitidos quando houver disponibilidade do enlace (COULOURIS et al., 2007a).

As redes de comutação de pacotes se difundiram rapidamente, e nessas redes um grande número de equipamentos foram e ainda são implantados. Cada equipamento deve ser configurado por um administrador usando interfaces de baixo nível ou frequentemente utilizando *software* embarcado do fabricante (*firmware*), tornando essas redes fechadas, proprietárias e verticalmente integradas, ossificando a infraestrutura (HANDLEY, 2006). Como resultado, os administradores de redes estão limitados aos comandos da interface, configurações e protocolos pré-definidos pelos fabricantes de equipamentos. Além disso, pesquisadores geralmente têm que criar seus próprios ambientes de testes ou tirar proveito de simulações para realizar experimentos. Em suma, nestas redes é difícil experimentar novas ideias, testar novos protocolos[2], inovar ou fazer melhorias (MCKEOWN et al., 2008; KREUTZ et al., 2014).

Em resposta às limitações da infraestrutura física das redes de comutação de pacotes, surgiu o paradigma *Software Defined Networking* (SDN), quebrando a integração vertical e separando a camada de dados da camada de controle, permitindo que novas ideias e protocolos sejam experimentados sem prejudicar o tráfego de produção (JAIN; PAUL, 2013) (KREUTZ et al., 2014). Em uma rede SDN, a camada de dados é formada pelos equipamentos de rede, os quais tornam-se dispositivos que

[1] A forma mais simples de pacote é uma sequência de dados binários (sequência de bits) de comprimento limitado, que encapsula a informação útil que trafega na rede, bem como informações de endereçamento suficientes para identificar a origem e o destino dessas informações (COULOURIS et al., 2007a).

[2] Conjunto de regras e formatos utilizados na comunicação entre processos com o objetivo de realizar uma tarefa (COULOURIS et al., 2007a)

apenas encaminham o tráfego baseado nas decisões tomadas por um controlador, elemento que assume o papel da camada de controle. Detalhes do paradigma SDN são apresentados no Capítulo 2.

Outra área com mudanças significativas nos últimos anos foi a área de telecomunicações, mais especificamente na forma como o serviço de voz vem sendo disponibilizado. Particularmente, a disponibilização de Voz sobre o Protocolo Internet, *Voice over Internet Protocol* (VoIP), vem revolucionando as possibilidades de comunicação aproveitando a expansão das redes de comutação de pacotes. A tecnologia VoIP permite que pessoas possam se comunicar utilizando áudio/voz utilizando a Internet como meio de transmissão ao invés das Redes Públicas de Telefonia Comutada - *Public Switched Telephone Network* (PSTN).

A PSTN é uma rede comutada por circuitos para comunicação por voz, fornecendo aos usuários conexões de circuito dedicadas fim a fim durante cada ligação, ou seja, uma ligação fecha um circuito quando um cliente chama outro e estabelece um caminho exclusivo entre eles. Inicialmente, a PSTN foi projetada com linhas fixas/analógicas e com capacidade para manipular uma largura de banda de 64Kbps, porém dedicada a cada circuito (WEISS; HWANG, 1998; VARSHNEY et al., 2002).

Já em uma rede VoIP, a comutação é feita por pacotes, o consumo de largura de banda é atribuído dinamicamente de acordo com o *Coder-Decoder* (Codec) utilizado e em uma chamada VoIP a voz deve ser transformada de sinal analógico em digital. Esse processo chama-se codificação, ou seja, a voz é digitalizada e transformada em pacotes de dados através de codificadores (Codecs). Uma vez codificada em pacotes, estes são encaminhados pela rede IP passando por roteadores e *switches* até o servidor VoIP, que por sua vez encaminha os pacotes até o destino, onde são decodificados e transformados em voz novamente.

Vários fatores influenciam uma chamada VoIP como: o protocolo de sinalização, escolha do Codec (ALI; VASSILARAS; NTAGKOUNAKIS, 2009), segurança, a habilidade de cruzar *firewalls* (KHLIFI; GREGOIRE; PHILLIPS, 2006) e servidores *Network Address Translation* (NAT) (CHEN; HUANG; CHAO, 2008) e a utilização de CPU em um servidor VoIP (COSTA et al., 2015). Um aumento na utilização de CPU do servidor VoIP também pode ocorrer em uma chamada entre clientes (*e.g.,* telefones IP, *smartphones*, celulares, *tablets*, PC com um *softphone* instalado) que possuem o mesmo Codec. Fatores como configuração do servidor VoIP ou a localização dos clientes na rede, fazem com que os pacotes de voz trafeguem pelo servidor e mesmo não havendo transcodificação esse processo consome CPU, pois o servidor precisa ler os dados de um cliente, enviar para o outro cliente e vice-versa.

Clientes localizados em domínio administrativo protegido por *firewall* e NAT

também representam um problema para o tráfego de voz sobre IP, pois o servidor VoIP não é capaz de resolver os endereços IPs privados dos clientes em uma chamada. Todo o tráfego de voz é feito baseado nos IPs públicos dos clientes e nesse caso, todo o tráfego passa pelo servidor VoIP consumindo CPU e largura de banda (KARAPANTAZIS; PAVLIDOU, 2009; LIN et al., 2010; PARK et al., 2008; CHEN; HUANG; CHAO, 2008; YERYOMIN; EVERS; SEITZ, 2008; KHLIFI; GREGOIRE; PHILLIPS, 2006; KHIRUL; ISLAM; HASAN, 2007). Em específico, para o caso do servidor VoIP Asterisk, caso exista *firewall* e/ou NAT entre os clientes, todo o tráfego de voz é feito passando por ele. A solução apresentada neste trabalho independe da existência destes elementos (*firewall* e NAT) na rede e realiza o tráfego direto de áudio entre os clientes sem passar pelo servidor VoIP.

A largura de banda e o consumo de dados também são preocupações quando se discute sobre a adoção da tecnologia VoIP, principalmente no provisionamento do serviço visando suportar uma quantidade substancial de usuários com qualidade de voz aceitável (COSTA et al., 2015). Empresas com matrizes e filiais podem adotar a tecnologia VoIP para reduzir custos com chamadas. Porém, muitos planos de utilização da Internet possuem franquias de consumo de dados que variam conforme a largura de banda contratada. A franquia determina a quantidade de dados em Gigabytes que podem ser trafegados em um período. Consumir dados além da franquia contratada usualmente gera cobrança pelo valor excedente e/ou redução na largura de banda até o fechamento do período de cobrança. A preocupação com a franquia é importante para o provisionamento do serviço VoIP, bem como a contratação de um plano que atenda às necessidades do cliente, seja pessoa física ou jurídica.

Nesse sentido, partindo do pressuposto da existência de uma arquitetura VoIP centralizada, com base na matriz, empresas com matrizes e filiais podem se beneficiar do presente trabalho reduzindo utilização de CPU no servidor VoIP, reduzindo a utilização da largura de banda no segmento no qual está localizado o servidor VoIP, além de reduzir a quantidade de dados trafegados entre as matrizes e filiais.

Nesse contexto, o presente trabalho tem como objetivo o desenvolvimento de uma solução baseada em SDN que permita reduzir o consumo de recursos em um servidor VoIP. A solução baseada em SDN permite que chamadas entre clientes que utilizem o mesmo Codec possam ser realizadas com o tráfego de voz fluindo diretamente entre os clientes. Dessa maneira, há redução no consumo de CPU e utilização de largura de banda.

1.1 OBJETIVOS DO TRABALHO

O presente trabalho tem como objetivo geral o desenvolvimento de uma solução baseada em SDN, denominada SDNVoIP, que permita reduzir o consumo de recursos em um servidor VoIP.

1.2 OBJETIVOS ESPECÍFICOS

Como objetivos específicos o presente trabalho visa reduzir o consumo dos seguintes recursos:

- Reduzir a utilização de CPU em um servidor VoIP;
- Reduzir a utilização da largura de banda em um servidor VoIP; e
- Reduzir o tráfego de dados no segmento de rede onde está localizado o servidor VoIP.

1.3 MÉTODO DE PESQUISA

Wazlawick (2014) conceitua Ciência como o esforço para descobrir e aumentar o conhecimento humano e o funcionamento da realidade. As Ciências Formais e Empíricas, a primeira pode ser vista como o estudo das ideias, independente de sua aplicação, o objeto de estudo dessas está em processos lógicos ou matemáticos, enquanto a segunda estudam fenômenos reais, sua fundamentação ocorre por observação.

Considerando pesquisas na área da Ciência da Computação, Wazlawick (2014) propõe cinco estilos para a classificação dos níveis de maturidade. No primeiro nível os trabalhos apenas apresentam algo novo, não apresentando embasamento científico para sua comprovação, não abordando a existência ou não de outros trabalhos. O segundo nível além de apresentar um produto, ou uma técnica, apresenta comparações qualitativas com outras pesquisas. Já o terceiro nível apresenta comparações quantitativas demonstrando os testes realizados e as diferenças de resultados obtidos. O quarto nível além de apresentar testes qualitativos, estes devem ser aceitos pela comunidade da área pesquisada. Por fim, o quinto nível apresenta a validação do estudo através de provas e teoremas matemáticos.

Gil (2002) propõe a classificação de pesquisas quanto aos objetivos em três grupos: exploratórias, descritivas e explicativas. As pesquisas exploratórias são bastante flexíveis, têm como objetivo familiarizar o pesquisador com o problema, tornando-o mais explícito ou construindo hipóteses, normalmente são realizadas com levanta-

mento bibliográfico, entrevistas com pessoas que possuam experiência com o problema abordado e análise de exemplos. As pesquisas descritivas apresentam como objetivo a descrição de algo, que pode ser características, fenômenos, relação entre variáveis, os pesquisadores geralmente são preocupados com atuações práticas. As pesquisas explicativas, consistem em identificar os fatores que determinam ou contribuem para a ocorrência de determinados fenômenos, tende a explicar as razões dos acontecimentos.

A classificação de pesquisas, as inseridas na área da Ciência da Computação, conforme Eden (2007) podem ser divididas em três paradigmas, sendo o racionalista o mais comum para pesquisas teóricas, busca o conhecimento a priori a partir de métodos formais e insere a Ciência da Computação na área da Matemática. O paradigma tecnocrático define a Ciência da Computação como uma área da Engenharia, buscando resolver problemas específicos com confiabilidade, porém sem garantia de generalização da solução proposta. Por último, o paradigma naturalista (científico) busca o conhecimento tanto a priori com a posteriori, as soluções propostas são validadas pela combinação de experimentação científica com deduções formais, e esta inserida na área das Ciências Naturais.

A pesquisa quanto ao procedimento de coleta pode ser classificada em: bibliográfica, de laboratório e de campo. A pesquisa de campo é a observação dos fatos tal como ocorrem, não permitindo isolar e controlar as variáveis, mas perceber e estudar as relações estabelecidas. A pesquisa experimental consiste em criar condições para interferir no aparecimento ou na modificação dos fatos, para poder explicar o que ocorre com fenômenos correlacionados. A pesquisa bibliográfica objetiva recuperar o conhecimento científico acumulado sobre um problema, colocando o pesquisador em contato direto com tudo o que foi escrito, dito ou filmado sobre determinado assunto (MARCONI; LAKATOS, 2003).

Dado esse contexto sobre metodologia da pesquisa, o presente trabalho é classificado como nível 2 (WAZLAWICK, 2014), pois apresenta uma solução com base científica e comparação qualitativa com outros trabalhos (Seção 3.3). Quanto ao método de pesquisa, é classificado como revisão da literatura, pois fora utilizando referências científicas da área na definição do problema introduzindo os conceitos e definições sobre SDN, sua arquitetura e principais elementos, a fim de possibilitar o entendimento do uso de SDN na solução proposta. A partir das informações obtidas, e juntamente com a fundamentação sobre redes SDN e VoIP, foi possível definir os requisitos de uma solução que permitisse a redução do consumo de recursos em um servidor VoIP. A solução foi então definida, sua arquitetura e funcionamento são apresentados. Com base em todas as informações levantadas, foram definidos e implementados cenários experimentais, a fim de validar a solução proposta.

De acordo com Gil (2002), o presente trabalho pode ser classificado como descritiva, pois descreve e detalha uma solução que tem como objetivo reduzir a utilização de recursos em um servidor VoIP com uma aplicação prática da solução. Já de acordo com Eden (2007) a pesquisa pode ser classificada como paradigma tecnocrático, pois resolve um problema específico em uma área específica. Segundo Marconi e Lakatos (2003) o presente trabalho é classificado como pesquisa de laboratório, pois consiste em uma análise experimental criando condições para interferir no aparecimento ou na modificação dos fatos.

A classificação da pesquisa quanto a forma de abordagem do problema pode ser qualitativa ou quantitativa. A pesquisa quantitativa considera que tudo pode ser quantificável, o que significa traduzir em números opiniões e informações para classificá-las e analisá-las. Isso requer o uso de recursos e técnicas estatísticas. A pesquisa qualitativa considera que existe uma subjetividade que não pode ser traduzida em números. Ela é descritiva e não requer o uso de métodos e técnicas estatísticas (TEIXEIRA, 2010). É importante ressaltar que a análise qualitativa é menos formal do que a análise quantitativa, pois nesta última seus passos podem ser definidos de maneira relativamente simples. No entanto, a análise qualitativa depende de muitos fatores, tais como a natureza dos dados coletados, a extensão da amostra, os instrumentos de pesquisa e os pressupostos teóricos que nortearam a investigação (GIL, 2002). Dessa maneira o presente trabalho quanto a abordagem do problema é classificado como quantitativo, apresentando resultados, análises e técnicas que permitem a reprodução dos valores obtidos.

Por fim, a análise dos resultados completou o método empregado neste trabalho, fornecendo uma avaliação sobre os experimentos realizados para testar a redução de utilização de recursos em um servidor VoIP, além de confrontar os requisitos da solução com os resultados obtidos. Esta análise permitiu também a definição de pontos de interesse para trabalhos futuros.

1.4 ORGANIZAÇÃO DO TEXTO

O presente trabalho está organizado como segue. O Capítulo 2 apresenta a revisão da literatura que servirá como base para entendimento dos conceitos e protocolos utilizados neste trabalho, bem como descreve em mais detalhes do problema atacado. O Capítulo 3 apresenta a proposta do presente trabalho, os requisitos funcionais e não funcionais e o plano de testes. O Capítulo 4 apresenta o desenvolvimento da solução proposta. O Capítulo 5 apresenta os experimentos realizados, bem como os resultados obtidos. O Capítulo 6 apresenta as considerações finais, trabalhos publicados e trabalhos futuros. Por fim, o Anexo A apresenta um relato de experiência com a utilização do módulo SDNVoIP em ambiente WAN.

2 FUNDAMENTAÇÃO E CONCEITOS

Este capítulo apresenta os conceitos e tecnologias relacionados com o presente trabalho. Inicialmente a Seção 2.1 apresenta com os conceitos de *Software Defined Networking* (SDN), sua arquitetura, seus elementos e aplicações. A Subseção 2.1.1 apresenta o histórico do protocolo OpenFlow, as diferenças entre as versões, a arquitetura de um *switch* OpenFlow e a definição das tabelas de fluxos. Essa apresentação é necessária para a compreensão do protocolo OpenFlow e sua utilização entre os elementos componentes da rede. Além desses conceitos, a Subseção 2.1.2 contextualiza sobre o funcionamento de um controlador, elemento chave em uma rede SDN. Ainda, na Seção 2.2, é introduzida a tecnologia VoIP, como é feito o tráfego de áudio, com destaque para os protocolos *Session Initiation Protocol* (SIP), *Realtime Transport Protocol* (RTP) e *Realtime Transport Control Protocol* (RTCP).

O presente trabalho utilizou em cenários experimentais o servidor VoIP Asterisk[1]. Portanto, é importante o entendimento de suas principais características, arquitetura e funcionalidades, assuntos que são apresentados na Subseção 2.2.3. Entretanto, a solução apresentada neste trabalho não é restrita somente ao uso do Asterisk, é possível utilizar outros servidores VoIP, desde que suportem os protocolos SIP, RTP e RTCP.

A Subseção 2.2.2 apresenta Codecs de voz com as características relevantes para a transmissão de voz em redes de comutação de dados. Por fim, a Seção 2.3 apresenta em mais detalhes o problema do redirecionamento de tráfego VoIP entre clientes, enfrentado por este trabalho, habilitando a melhor compreensão da solução proposta.

2.1 SOFTWARE DEFINED NETWORKING

De acordo com McKeown et al. (2008), a redução do impacto real de qualquer inovação na área de redes é devido a grande base instalada de equipamentos, protocolos e a relutância em realizar experimentos com tráfego de produção. A infraestrutura da Internet e das redes de computadores geralmente consiste de diferentes dispositivos de redes como roteadores, *switches* e *middle-boxes*, os quais são verticalmente integrados, com alta vazão e funções específicas. Para gerenciar e configurar estes dispositivos, é utilizado um conjunto específico e pré-definido de comandos de acordo com o sistema operacional utilizado (MASOUDI; GHAFFARI, 2016). Assim, as redes tradicionais são centradas em hardware e sofrem com problemas relacionados

[1] <http://www.asterisk.org/>

à pesquisa e inovações, confiabilidade, extensibilidade, flexibilidade e capacidade de gerenciamento. Uma vez que a Internet e as redes móveis se desenvolvem, e as novas tecnologias como nuvem, redes sociais e virtualização prosperam, a necessidade de redes com maior largura de banda, maior acessibilidade e gerenciamento dinâmico torna-se um problema crítico (MASOUDI; GHAFFARI, 2016; FARHADY; LEE; NAKAO, 2015; KREUTZ et al., 2014). Em suma, nessas redes é difícil experimentar novas ideias, testar novos protocolos, inovar ou fazer melhorias (MCKEOWN et al., 2008).

De acordo com Coulouris et al. (2007b) o *software* de rede é organizado em uma hierarquia de camadas ou níveis. Cada camada apresenta uma interface bem definida para as camadas que estão acima dela e implementa novos serviços sobre as funcionalidades do sistema de comunicação da camada inferior. Os equipamentos de rede geralmente são fechados e possuem suas próprias camadas de dados e controle, ou seja, quando os equipamentos de rede recebem um pacote, um programa é então executado e diferentes ações podem ser aplicadas no pacote, como por exemplo encaminhar ou excluir. Implantar novos protocolos e serviços, ou até mesmo atualizar existentes é um grande desafio porquê os equipamentos de rede precisam ser atualizados ou substituídos.

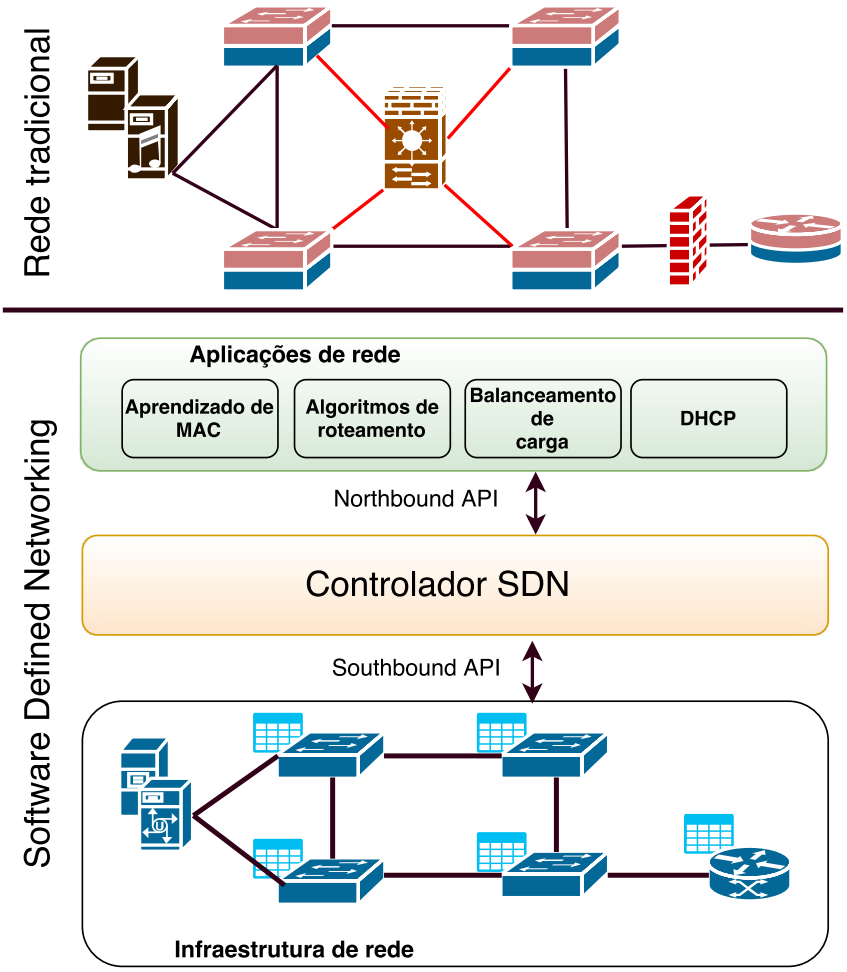
SDN é um paradigma emergente que surgiu em resposta às limitações das redes tradicionais, separando a camada de controle da camada de dados, prometendo melhorar a utilização dos recursos de rede, simplificar o gerenciamento de rede, reduzir custos de operação e promover inovação e evolução (AKYILDIZ et al., 2014). A Figura 1 apresenta na parte superior um exemplo de rede tradicional, onde há o acoplamento da camada de controle e da camada de dados embutidas nos equipamentos de comutação de pacotes da rede. Já na parte inferior da Figura 1 é apresentado a arquitetura SDN que é composta de três partes.

O nível mais baixo representa a infraestrutura de rede ou camada de dados, ou seja, nessa camada encontram-se os dispositivos de rede que apenas encaminham o tráfego baseado nas decisões da camada de controle. Nessa camada os dispositivos de rede são simples encaminhadores de pacotes, não possuem o *software* que toma as decisões autonomamente (FARHADY; LEE; NAKAO, 2015).

A camada intermediária é a camada de controle ou controlador. O controlador é o elemento chave dessa camada. Atua como um *cérebro* e possui a inteligência do encaminhamento de pacotes e fornece uma visão centralizada da rede. É também responsável por gerenciar todos os dispositivos da rede, adicionando e removendo fluxos das tabelas de fluxos (MASOUDI; GHAFFARI, 2016) existentes nos equipamentos da camada inferior.

A camada superior (Figura 1) da arquitetura SDN é representada pelas aplica-

Figura 1 – Arquiteturas de rede: tradicional *vs.* SDN.



Fonte: Adaptado de (KREUTZ et al., 2014)

ções de rede. As aplicações são desenvolvidas utilizando uma *Northbound API* fornecida pelo desenvolvedor do controlador. Estas APIs fornecem um nível de abstração que permite às aplicações não depender de detalhes específicos da infraestrutura. Como exemplos de aplicações pode-se citar: algoritmos de roteamento, balanceamento de carga, monitoração, detecção de ataques, aplicação de políticas e *firewall*. A comunicação entre o controlador e os dispositivos encaminhadores é realizada por uma *Southbound* API. A *Southbound API* é a ponte de comunicação entre o controlador e os dispositivos encaminhadores e é um elemento importante para a separação das funcionalidades de controle e dados. A *Southbound API* fornece uma interface comum para as camadas superiores, além de permitir a utilização de diferentes protocolos como o OpenFlow (FOUNDATION, 2012), OVSDB (PFAFF; DAVIE, 2013), ForCES (WANG et al., 2010), SNMP (PRESUHN, 2002), BGP (LOUGHEED; REKHTER, 2006). Já a *Northbound API* oferece como opções de comunicação RESTful APIs,

bibliotecas para linguagens de programação como Java, PHP e Python, APIs especializadas como NVP NBAPI (KOPONEN et al., 2014), sendo que alguns controladores fornecem linguagem de programação específica como Frenetic (FOSTER et al., 2011), NetCore (MONSANTO et al., 2012) e Procera (VOELLMY; KIM; FEAMSTER, 2012). As APIs *Northbound* e *Southbound* são duas abstrações chaves na arquitetura SDN. O protocolo OpenFlow representa e implementa a *Southbound* API mais aceita pela indústria, porém uma *Northbound* API comum ainda é uma questão em aberto. Segundo Kreutz et al. (2014), a arquitetura SDN possui quatro pilares:

- Desacoplamento das camadas de controle e dados. O dispositivo de rede torna-se um simples encaminhador de pacotes;
- Decisões de encaminhamento são baseadas em fluxos. Um fluxo é um conjunto de valores de campos de cabeçalhos de pacotes atuando como critério de correspondência, bem como um conjunto de ações relacionada com os pacotes correspondentes;
- A lógica de controle é movida para uma entidade externa chamada controlador SDN e provê abstrações (*Northbound Interface*) para facilitar a programação dos dispositivos de encaminhamento; e
- A rede é programável através de *software* que localiza-se na camada acima do controlador interagindo com a camada abaixo do controlador.

Além dos protocolos citados (*i.e.,* OpenFlow, OVSDB, ForCES, SNMP, BGP), o canal de comunicação entre o controlador e os FDs pode ser feita através de uma rede dedicada (*out-of-band*) ou através da rede gerenciada pelo *switch* OpenFlow (*in-band*). Uma rede dedicada para a comunicação entre o controlador e os FDs permite segregar todo o tráfego de controle do tráfego de produção, porém o documento de especificação do protocolo OpenFlow não trata em detalhes esse assunto. A única condição é que exista uma conectividade TCP/IP (FOUNDATION, 2012).

2.1.1 OpenFlow

Para o presente trabalho, um pacote é considerado uma série de *bytes* compreendendo um cabeçalho, um *payload* e um *trailer*, nessa ordem, e é tratado como uma unidade para fins de processamento e encaminhamento. O tipo de pacote padrão é um quadro Ethernet, porém outros tipos de pacote também são suportados (FOUNDATION, 2012).

O OpenFlow é um protocolo aberto que padroniza a comunicação entre as camadas de controle e de dados. Descreve como um *software* pode programar a ta-

bela de fluxos em diferentes *switches*. Embora a ideia geral permaneça, o protocolo OpenFlow evoluiu com o tempo através de versões. Essa evolução trouxe melhorias, novas funcionalidades e novos suportes. A Tabela 1 elenca algumas das principais características em cada versão do protocolo OpenFlow.

Tabela 1 – Versões do protocolo OpenFlow.

	1.0	**1.1**	**1.2**	**1.3**	**1.4**	**1.5**
Tabelas de Fluxos	Única	Múltiplas	Múltiplas	Múltiplas	Múltiplas	Múltiplas
Tabela de Grupos	Não	Sim	Sim	Sim, com suporte mais flexível	Sim	Sim
Suporte ao IPv6	Não	Não	Sim	Sim, adicionado novo campo do cabeçalho	Sim	Sim
Suporte a múltiplos controladores	Não	Não	Sim	Sim, habilitadas conexões auxiliares	Sim	Sim

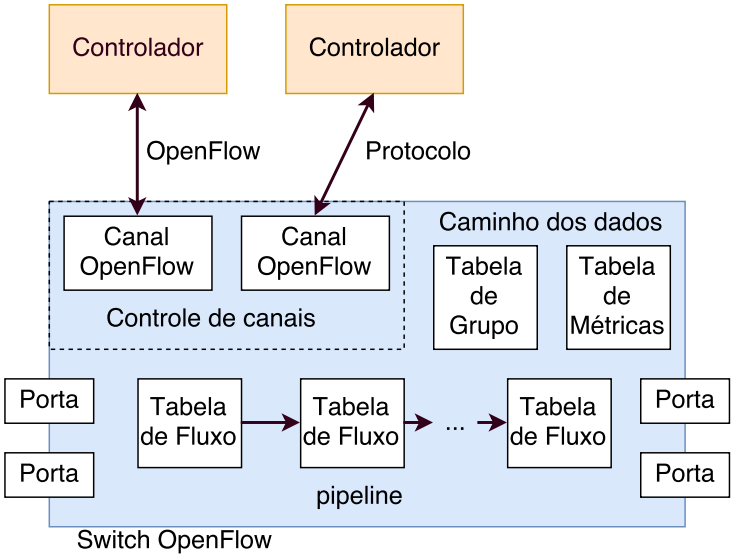
Fonte: Adaptado de Lara, Kolasani e Ramamurthy (2014)

De acordo com a Tabela 1, a primeira versão do protocolo possui suporte para apenas uma tabela de fluxo. Na versão 1.1 foi adicionado suporte para múltiplas tabelas de fluxos e o processamento por *pipeline*. A versão 1.2 possui uma estrutura de correspondência mais flexível, além de suportar mais protocolos como o IPv6 (FARHADY; LEE; NAKAO, 2015). A versão 1.3 possui suporte a mais protocolos como o MPLS, BoS bit e ao cabeçalho estendido do IPv6, além de melhorias nas métricas. A versão 1.4 melhorou a correspondência introduzida na versão 1.2, possibilitando maior flexibilidade na classificação de pacotes, permitindo combinar campos de cabeçalho de pacotes. Além disso, a versão 1.4 passa a suportar portas óticas. A versão 1.5 teve poucas inovações quando comparadas com as anteriores, possuindo apenas melhorias nas funcionalidades existentes (REN; XU, 2014; FARHADY; LEE; NAKAO, 2015).

Ao tempo da escrita deste trabalho, a especificação da última versão do protocolo foi liberada em 26/março/2015 pela Open Networking Foundation (FOUNDATION, 2015) (Versão 1.5.1), responsável por manter as especificações: do protocolo Openflow e de um *switch* OpenFlow. Embora a versão a 1.5.1 seja a mais atual, a versão do utilizada do OpenFlow neste trabalho foi a 1.3, pois foi utilizado também neste trabalho o controlador Floodlight versão 0.9, que por sua vez, suporta versões do OpenFlow somente até a versão 1.3. Desse modo, quando for mencionando OpenFlow tem-se como base a versão 1.3 salvo os casos nos quais a versão específica é citada.

A Figura 2 apresenta a arquitetura de um *switch* OpenFlow de acordo com a especificação do protocolo OpenFlow (FOUNDATION, 2012) e que deve possuir os seguintes componentes: uma ou mais tabelas de fluxos, uma tabela de grupo, um ou mais canais de comunicação para um controlador externo utilizando o protocolo OpenFlow, uma tabela de métricas e um ou mais controladores externos.

Figura 2 – Arquitetura de um *switch* OpenFlow.



Fonte: Adaptado de Foundation (2012)

Através do protocolo OpenFlow, o controlador pode adicionar, atualizar e remover registros da tabela de fluxo utilizando abordagem reativa ou pró-ativa (FOUNDATION, 2012). Na abordagem reativa, os *switches* que compõem o caminho físico de um fluxo são configurados individualmente sempre que um novo pacote é encaminhado ao controlador por não pertencer a um fluxo já existente no *switch*. Já na abordagem pró-ativa os fluxos são configurados nos *switches*, ou seja, os fluxos são instalados antes do tráfego de produção ter início (JUNIOR; FIORESE; KOSLOVSKI, 2016) (TOOTOONCHIAN et al., 2012).

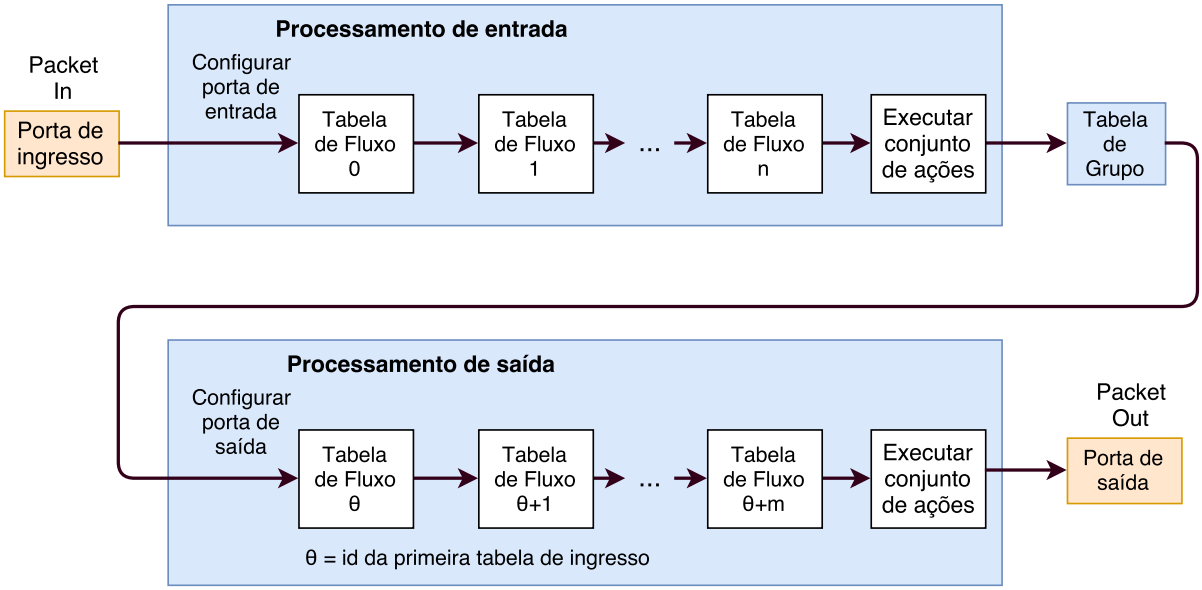
De acordo com a especificação do protocolo OpenFlow (FOUNDATION, 2012), as tabelas de fluxos são tabelas de encaminhamento, contém um conjunto de registros de fluxos, cada registro consiste em campos de correspondência (*match fields*), contadores (*counters*) e um conjunto de ações que devem ser aplicadas nos pacotes correspondentes. A correspondência (*matching*) inicia na primeira tabela do *switch* (Figura 2) e pode continuar nas tabelas seguintes. Caso uma correspondência seja encontrada, as instruções associadas com o fluxo são executadas. Caso nenhuma correspondência seja encontrada o pacote pode ser descartado, encaminhado para

o controlador ou continuar pela busca de uma correspondência na próxima tabela de fluxo.

Instruções associadas com cada entrada de fluxo ou contêm ações ou modificam o processamento no *pipeline*. As ações descrevem como o pacote deve ser encaminhado ou modificado. Já as instruções de processamento no *pipeline* permitem que os pacotes sejam encaminhados para processamento em tabelas subsequentes e permite que a informação em forma de metadados, seja comunicada entre as tabelas. O processamento no *pipeline* finaliza quando as instruções associadas não especificam uma próxima tabela e neste momento o pacote é modificado e encaminhado (FOUNDATION, 2012).

Em uma rede SDN baseada em OpenFlow, os *switches* podem ser de dois tipos: *OpenFlow-only* e *OpenFlow-hybrid*. Os *switches OpenFlow-only* suportam apenas operações OpenFlow. Os pacotes são processados somente pelo *OpenFlow pipeline* (Figura 3). Por outro lado, *switches OpenFlow-hybrid* suportam ambas as operações: *OpenFlow pipeline* e processamento normal Ethernet (camada 2), isolamento *Virtual Local Area Network* (VLAN), roteamento de camada 3, *Access Control List* (ACL) e *Quality of Service* (QoS), conforme a especificidade do equipamento.

Figura 3 – Processamento do Pipeline OpenFlow.



Fonte: Adaptado de Foundation (2012)

As ações podem transmitir os pacotes a uma porta, geralmente uma porta física, mas também pode ser uma porta lógica. Algumas portas são reservadas especificando ações de encaminhamento genéricas como o envio ao controlador, ou encaminhar usando o processamento de um *switch* normal, ou seja sem o processamento

do *OpenFlow Pipeline*.

A tabela de fluxo de acordo com a especificação do protocolo OpenFlow (FOUNDATION, 2012) possui as seguintes colunas:

- *Match fields* - Várias informações podem ser utilizadas nessa coluna para fazer a correspondência de pacotes. Estes consistem na porta de entrada, cabeçalhos dos pacotes, e opcionalmente outros campos da *pipeline*, como metadados especificados por uma tabela anterior;
- *Priority* - é um número que corresponde à precedência do *matching* na tabela;
- *Counter* - contador, é incrementado quando um pacote corresponde ao fluxo atual;
- *Instructions* - ações que devem ser executadas no pacote;
- *Timeouts* - tempo máximo de ociosidade para um fluxo expirar na tabela; e
- *Cookie* - valor escolhido pelo controlador, não utilizado no processamento de pacotes.

2.1.2 Controlador

O controlador pode ser caracterizado como a combinação das funções básicas de serviço de rede e suas APIs (ex. *southbound, northbound*). As funções básicas de serviço de rede são consideradas funcionalidades essenciais que todos os controladores devem fornecer como serviços básicos de sistemas operacionais, tais como execução de programas, operações de entrada e saída, controle, comunicação e segurança entre outros. Esses serviços são usados por outros serviços no nível do sistema operacional e por aplicações. Funções como topologia, estatísticas, notificações e gerenciamento de dispositivos, juntamente com encaminhamento e roteamento por caminhos mais curtos e mecanismos de segurança são funcionalidades de controle de rede que as aplicações utilizam em sua lógica (KREUTZ et al., 2014).

O controlador é responsável por adicionar, atualizar e remover registros das tabelas de fluxos e pode ser centralizado ou distribuído. O controlador provê também uma *Northbound API* que pode ser usada para o desenvolvimento de tarefas de gerenciamento e oferecer novas funcionalidades de rede. Existem diversos controladores disponíveis que fornecem APIs para diversas linguagens de programação, como por exemplo o Floodlight (FLOODLIGHT, 2017) que fornece API para a linguagem Java.

O controlador centralizado oferece vários benefícios como simplicidade e menor propensão a erros ao modificar políticas de rede. Podem reagir automaticamente

às mudanças do estado da rede e a centralização da lógica de controle com conhecimento global da rede simplifica o desenvolvimento de serviços e provê funções de rede mais sofisticadas (KREUTZ et al., 2014).

Entretanto, um controlador centralizado pode ser um ponto de falha na rede e pode possuir limitações de escalabilidade, além de introduzir um atraso no encaminhamento de pacote causado pela interação entre a camada de controle e a camada de dados (JAIN; PAUL, 2013). Já um controlador distribuído não possui essas limitações, porém pode possuir problemas comuns aos sistemas distribuídos como por exemplo problemas na sincronia do estado na rede. Uma abordagem híbrida foi utilizada no projeto Kandoo (Hassas Yeganeh et al., 2012). O projeto Kandoo utilizou controladores locais para aplicações locais e um controlador global para decisões que necessitam de informações do estado global da rede. Esta abordagem reduz a quantidade de requisições no controlador global enquanto provê respostas rápidas para aplicações locais.

Neste trabalho o controlador foi utilizado de forma centralizada, porém é importante destacar que o controlador utilizado pode funcionar de forma centralizada ou distribuída dependendo da maneira como é configurado e da necessidade.

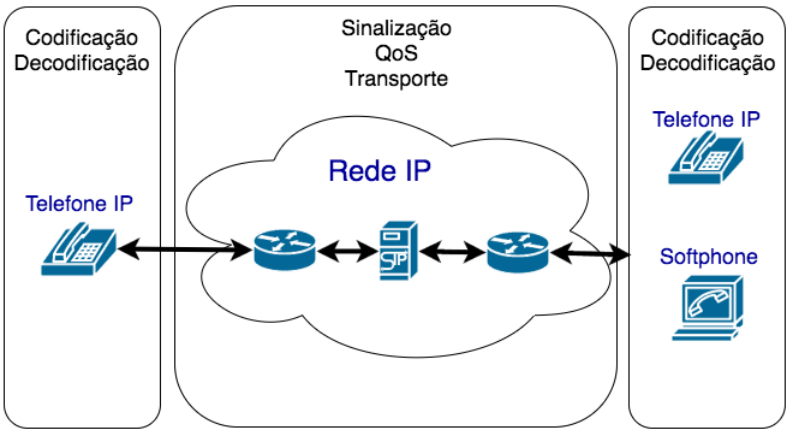
2.2 VOZ SOBRE IP

O Voz sobre o Protocolo Internet (*Voice over Internet Protocol* - VoIP) é uma tecnologia que permite as pessoas utilizarem a *Internet* ou uma rede de computadores baseada em IP como meio de comunicação por voz (KARAPANTAZIS; PAVLIDOU, 2009). O VoIP oferece funcionalidades *voicemail*, identificação de chamadas, conferência, chamada em espera, encaminhamento de chamadas e integração com a rede pública de telefonia (PSTN) permitindo chamadas locais, longa distância e internacionais entre clientes VoIP e clientes da PSTN. Além dessas funcionalidades, a principal razão para sua adoção é a redução de custos devido à uma única rede suportar ambos os serviços: voz e dados (WEISS; HWANG, 1998; VARSHNEY et al., 2002).

A tecnologia VoIP permite que empresas interliguem matrizes e filiais reduzindo custos com ligações entre ramais. Chamadas provenientes da rede PSTN podem ser facilmente direcionadas para um telefone VoIP independentemente da localização na rede e um número VoIP conectado na Internet pode receber ligações em qualquer localização geográfica. Outras funcionalidades podem ser integradas ao serviço VoIP como serviços de comunicação por video, mensageiros, compartilhamento de arquivos, *voicemail* e gerenciamento de contatos. O funcionamento de chamadas VoIP é similar ao de chamadas tradicionais e envolve codificação, decodificação e sinalização. A Figura 4 exemplifica uma arquitetura geral VoIP, na qual uma chamada

VoIP é iniciada em um telefone IP, a voz é digitalizada e transformada em pacotes através de codificadores, os pacotes são encaminhados na rede IP passando por roteadores e *switches* até o servidor VoIP, que por sua vez encaminha os pacotes até o destino onde os pacotes são decodificados e transformados em voz novamente.

Figura 4 – Arquitetura VoIP.



Fonte: Adaptado de (DIGIUM, 2017)

O tráfego de áudio é feito pelo protocolo *Realtime Transport Protocol* (RTP) que utiliza como transporte o *User Datagram Protocol* (UDP). O áudio é codificado por um Codec. Um Codec é um algoritmo que permite o transporte de um sinal analógico através de linhas digitais. Existem vários Codecs que variam em complexidade, largura de banda necessária para transmissão e qualidade de voz. Quanto maior a qualidade, maior largura de banda é necessária (KARAPANTAZIS; PAVLIDOU, 2009).O protocolo RTP e os Codecs de áudio são tratados em detalhes nas Subseções 2.2.1.2 e 2.2.2 respectivamente.

Outro protocolo utilizado na arquitetura VoIP é o *Session Initiation Protocol* (SIP). Seu propósito é fazer a sinalização das chamadas e controlar as sessões de comunicação. Detalhes do protocolo SIP são descritos na Subseção 2.2.1.1.

2.2.1 Protocolos para VoIP

Nesta Seção são apresentados os protocolos envolvidos com a tecnologia VoIP como o protocolo *Session Initiation Protocol* (SIP), cujo propósito é fazer a sinalização das chamadas e controlar as sessões de comunicação. O tráfego de áudio é realizado pelo protocolo *Realtime Transport Protocol* (RTP) que utiliza como transporte o protocolo *User Datagram Protocol* (UDP), auxiliado pelo protocolo de controle *Realtime Transport Control Protocol* (RTCP).

2.2.1.1 Session Initiation Protocol

O protocolo SIP foi publicado pela primeira vez em março de 1999 através da *Request For Comments* (RFC) 2543 e revisado em junho de 2002 pela RFC 3261. De acordo com a RFC 3261 (ROSENBERG et al., 2002) o SIP é um protocolo de controle (sinalização) da camada de aplicação que pode estabelecer, modificar e finalizar sessões com um ou mais participantes. Pode ser usado para estabelecer sessões de voz, sessões de video, *chat*, mensagens instantâneas e jogos entre outras.

Para exemplificar o funcionamento básico do SIP e facilitar a compreensão da proposta apresentada, onde a interpretação da informação SIP é importante para o estabelecimento das ações a serem executadas no *switch* OpenFlow, é descrito um exemplo resumido e adaptado ao encontrado na RFC 3261 (ROSENBERG et al., 2002), com um fluxo de troca de mensagens SIP entre dois clientes e um servidor VoIP. O fluxo de troca de mensagens SIP em uma chamada VoIP é exemplificado pela Figura 5. As portas SIP dos clientes não foram informadas, pois com exceção da porta SIP no servidor VoIP (porta 5060), as portas são escolhidas aleatoriamente ou pré-configuradas manualmente no cliente VoIP. Um exemplo de mapeamento de portas é descrito em detalhes na Seção 4.2.

A Figura 5, apresenta a troca de mensagens para o estabelecimento de uma sessão SIP (entre dois usuários hipotéticos Paulo e Adriano), necessária para o transporte das informações VoIP através dos protocolos RTP e RTCP, bem como sua finalização. Ainda de acordo com a Figura 5, a sequência de troca de mensagens ocorre da seguinte maneira:

1. Paulo e Adriano enviam cada um, uma mensagem SIP de registro (*REGISTER*) para o servidor VoIP. Essa mensagem contém as informações de usuário como nome de usuário, IP, porta SIP e IP do servidor de destino entre outras informações;

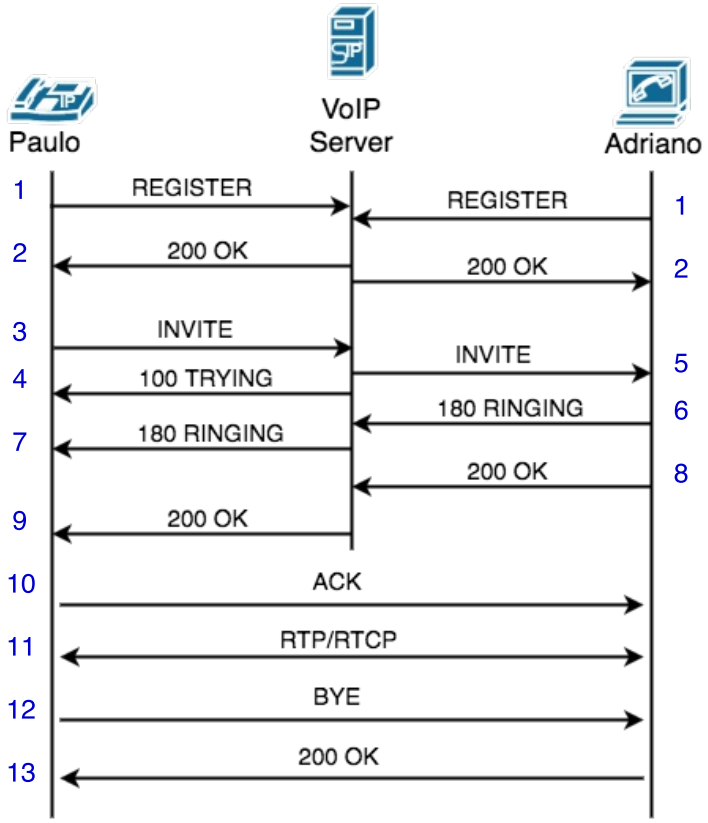
2. As informações de autenticação de ambos os usuários é aceita pelo servidor VoIP que envia para cada um uma mensagem SIP 200 OK;

3. Paulo inicia uma chamada enviando a mensagem SIP INVITE para o servidor VoIP. A mensagem contém o nome do usuário chamado, a porta que Paulo usará para receber e enviar áudio e o Codec que Paulo está habilitado a usar;

4. O servidor VoIP ao receber a mensagem INVITE de Paulo imediatamente responde com uma mensagem SIP 100 TRYING;

5. O servidor VoIP encaminha a mensagem INVITE para o usuário Adriano com as informações do usuário Paulo;

Figura 5 – Troca de mensagens SIP.



Fonte: Adaptado de (ROSENBERG et al., 2002)

6. O usuário Adriano recebe o INVITE e responde com uma mensagem SIP 180 RINGING;

7. O servidor VoIP devolve a mensagem SIP 180 RINGING para o usuário Paulo informando que o cliente VoIP de Adriano está sendo chamado;

8. Quando o usuário Adriano atende a chamada, o cliente VoIP envia uma mensagem SIP 200 OK em resposta à mensagem INVITE enviada anteriormente para o servidor VoIP. A mensagem contém as informações de Adriano como o IP, porta que Adriano usará para receber e enviar áudio e o Codec que Adriano está habilitado a usar;

9. O servidor VoIP informa o usuário Paulo que a chamada foi atendida enviando a mensagem SIP 200 OK com as informações de Adriano;

10. Paulo envia uma mensagem SIP ACK para Adriano assim que recebe a mensagem SIP 200 OK;

11. Quase ao mesmo tempo é estabelecida uma sessão RTP entre o IP de Paulo e o IP de Adriano e assim que a sessão é estabelecida pacotes RTP trafegam entre os dois usuários;

12. Após a conversa, qualquer um dos participante pode terminar a chamada enviando uma mensagem SIP BYE. Nesse exemplo Paulo termina a chamada; e

13. Por fim, Adriano confirma o término da chamada enviando uma mensagem SIP 200 OK.

O protocolo SIP utiliza o protocolo UDP como transporte. O SIP utiliza também o protocolo *Session Description Protocol* (SDP). O SDP é um protocolo com o propósito de representar os metadados necessários para o estabelecimento de teleconferências, chamadas VoIP, fluxo de video e outras sessões (CAMARILLO, 2006).

2.2.1.2 Realtime Transport Protocol e Realtime Transport Control Protocol

O RTP é um protocolo que provê entrega de dados fim a fim com características de tempo-real, como áudio e video interativo. Estes serviços incluem na carga do pacote de dados a identificação do tipo de mídia, número de sequência, data e hora e informações para monitoramento da entrega (SCHULZRINNE et al., 2003).

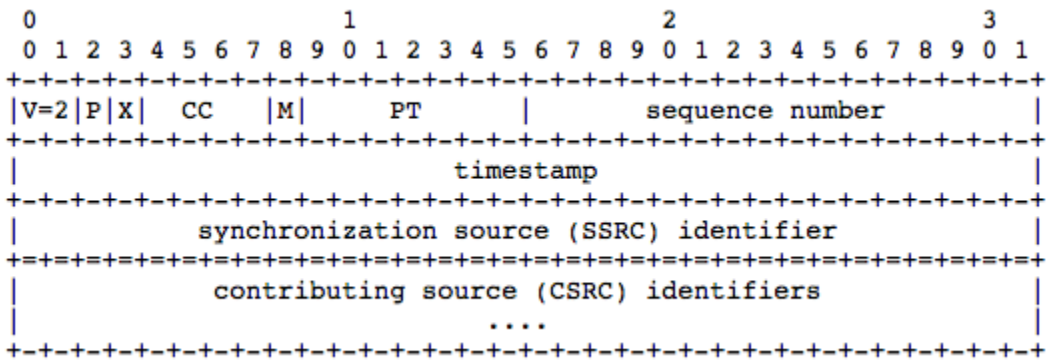
O protocolo RTP não garante qualidade de serviço e nem provê um mecanismo para assegurar a entrega temporal da informação, mas confia nas camadas inferiores para essas tarefas. Também não garante a entrega e nem a entrega em ordem. Assim um número de sequência é incluído no pacote RTP, permitindo ao receptor reconstruir a sequência de pacotes. Embora tenha sido pensado para satisfazer as necessidades de conferências de áudio, o protocolo não está limitado somente a este tipo de aplicação. Pode ser aplicado também para dados contínuos de armazenamento, simulações distribuídas interativas, controle e aplicações de monitoramento (SCHULZRINNE et al., 2003).

Como já descrito, uma chamada de áudio utiliza um par de portas lógicas. Por exemplo, um cliente determina que a porta 25000 será a porta RTP para enviar e receber áudio, uma segunda porta com numeração em sequência é utilizada para o controle dos pacotes RTCP, no caso, a porta 25001. Em uma chamada os participantes enviam os dados de áudio em pequenas partes, com aproximadamente 20ms de duração dependendo do Codec utilizado. Essas partes são precedidas pelo cabeçalho RTP e todos esses dados transformados na carga útil de um datagrama UDP.

O cabeçalho RTP apresentado na Figura 6 possui os seguintes campos e tamanhos:

- Versão (V): 2 bits;
- *Padding* (P): 1 bit;
- *Extension* (X): 1 bit;
- CSRC *count* (CC): 4 bits;
- *Marker* (M): 1 bit;
- *Payload type* (PT): 7 bits;
- *Sequence number*: 16 bits;
- *Timestamp*: 32 bits;
- SSRC: 32 bits; e
- CSRC *list*: 0 a 15 itens, 32 bits cada

Figura 6 – Cabeçalho RTP.



Fonte: (SCHULZRINNE et al., 2003)

Já o protocolo RTCP é baseado na transmissão periódica de pacotes de controle para todos os participantes da sessão e sua principal função é prover informações sobre a qualidade da distribuição dos dados.

Cada pacote RTCP começa com uma parte fixa semelhante a pacotes RTP, seguido por elementos estruturados que podem ter comprimento variável de acordo com o tipo de pacote, mas deve terminar com um limite de 32 bits, aém disso, pacotes RTCP transportam uma variedade de informações de controle:

- *Sender Report* (SR): estatísticas de transmissão e recepção de participantes remetentes ativos;
- *Receiver Report* (RR): estatísticas de recepção dos participantes que não são remetentes ativos;
- *Source Description* (SDES): descrição da fonte, incluindo CNAME (*canonical name*) que é um identificador de nível de transporte persistente para um ponto final RTP;
- BYE: indica o fim da participação;
- APP: Outros tipos de pacotes RTCP

2.2.2 Codecs de Voz

Codecs de voz são algoritmos que convertem um sinal analógico de voz em sinal digital habilitando-o a ser transmitido através de linhas de comunicação digitais. Os Codecs variam na qualidade do áudio, largura de banda necessária e complexidade (MEHTA; UDANI, 2001). Quanto maior a qualidade, maior largura de banda é necessária para a transmissão (KARAPANTAZIS; PAVLIDOU, 2009). O Codec utilizado em uma chamada é negociado entre as partes durante a troca de mensagens SIP. Clientes e servidores VoIP suportam diferentes Codecs. Os Codecs podem ser traduzidos de um para outro quando uma chamada entre os clientes utilizem Codecs diferentes, porém, este processo tem um custo computacional que eleva a utilização de CPU dependendo do *bit rate*[2]. A escolha de um Codec depende de alguns fatores como a qualidade desejada para as chamadas, largura de banda, resistência a perda de pacotes e em alguns casos o custo de licenciamento, pois alguns Codecs exigem pagamento de *royalties* para serem usados.

A *International Telecommunication Union-Telecommunication* (ITU-T) (UNION, 1988a) é o órgão que controla a aprovação de um Codec. A ITU avalia e atribui uma pontuação para o Codec após um processo de testes (UNION, 2005b). O *Mean Opinion Score* (MOS) é um método subjetivo de teste de qualidade de Codecs e atribui valores de 1 (ruim) até 5 (excelente).

Codecs de banda estreita como o G711, G723.1, G726, G729 entre outros, utilizam frequências de sinais entre 300 e 3400Hz com amostras de 8KHz. Codecs de banda larga operam sinais de áudio com frequências entre 50 e 7000Hz com amostras de 16KHz (UNION, 1988a). O Codec G711 é um *Pulse Code Modulation* (PCM)

[2] Taxa ou fluxo de bits ou fluxo de transferência de bits. Nas telecomunicações e na computação, o bit rate é o número de bits convertidos ou processados por unidade de tempo. O bit rate é medido em 'bits por segundo' (bps ou b/s), muitas vezes utilizado em conjunto com um prefixo do Sistema Internacional (SI), como Kbps, Mbps, Gbps, entre outros

que produz um *byte* a cada 125 µm, resultando em 64Kbps de *bit rate*. Este Codec é conhecido como *u-law* na América do Norte e Japão e como *a-law* na Europa e resto do mundo (UNION, 1988b).

O Codec G723.1 (UNION, 2006) pode operar em duas *bit rates*: 6.3Kbps usando o algoritmo *Multi Pulse-Maximum Likelihood Quantization* (MPC-MLQ) e também 5.3Kbps usando o algoritmo *Algebraic Code Excited Linear Prediction* (ACELP). O Codec G726 (UNION, 1990) pode operar em quatro *bit rates*, 16, 24, 32 e 40Kbps e utiliza o algoritmo *Adaptive Differential Pulse Code Modulation* (ADPCM). Este Codec é recomendado para a conversão de um pulso de 64Kbps *a-law* ou *u-law* de e para um canal de 16, 24, 32 ou 40Kbps. A relação entre as frequências de sinais e a codificação/decodificação do Codec G711 está totalmente especificada na Recomendação G.711 (UNION, 1988b).

O Codec G729 (UNION, 2012) opera em 8Kbps de *bit rate*. É o Codec que consome menos largura de banda e possui uma qualidade de áudio boa, porém o consumo de CPU é alto, além de ser um Codec proprietário sendo necessário pagamento para seu uso. Diferentes Codecs podem ser utilizados em uma chamada, porém isto resulta em transcodificação, ou seja os dados precisam ser decodificados e recodificados para que o áudio seja ouvido nos terminais da chamada. A transcodificação consome processamento e pode afetar a qualidade da voz (GOODE, 2002).

Tabela 2 – Características dos Codecs.

Codec & Bit Rate (Kbps)	**Codec Sample Interval (ms)**	**Codec Sample Size (Bytes)**	**Mean Opinion Score (MOS)**	**Voice Payload Size (Bytes)**	**Voice Payload Size (ms)**	**Packets Per Second (PPS)**	**Bandwidth Ethernet (Kbps)**
G.711 (64 Kbps)	10 ms	80 Bytes	4.1	160 Bytes	20 ms	50	87.2 Kbps
G.729 (8 Kbps)	10 ms	10 Bytes	3.92	20 Bytes	20 ms	50	31.2 Kbps
G.723.1 (6.3 Kbps)	30 ms	24 Bytes	3.9	24 Bytes	30 ms	33.3	21.9 Kbps
G.723.1 (5.3 Kbps)	30 ms	20 Bytes	3.8	20 Bytes	30 ms	33.3	20.8 Kbps
G.726 (32 Kbps)	5 ms	20 Bytes	3.85	80 Bytes	20 ms	50	55.2 Kbps
G.726 (24 Kbps)	5 ms	15 Bytes	3.5	60 Bytes	20 ms	50	47.2 Kbps

A Tabela 2, adaptada de (SYSTEMS, 2016), apresenta características dos principais Codecs sendo:

- *Bit Rate* (Kbps) é o número de *bits* por segundo necessários para serem transmitidos. É calculado da seguinte maneira: *bit rate = sample size / sample interval*;
- *Codec sample interval* é o intervalo de amostragem com o qual o Codec opera;
- *Codec Sample Size* é o número de *bytes* gerados a cada intervalo de amostragem;
- MOS pontuação do Codec de acordo com a ITU-T (UNION, 1988a);
- *Voice Payload Size*, número de *bytes* preenchido no pacote com voz; e
- *Packets per second* (PPS), número de pacote que precisam ser transmitidos para entregar o *bit rate* do Codec.

2.2.3 Asterisk

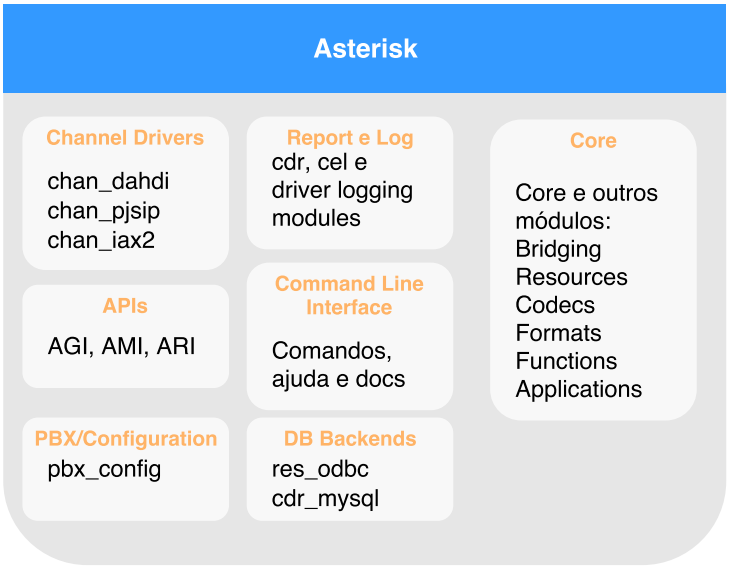
O Asterisk é um *software* servidor que executa a funcionalidade de *Private Automatic Branch Exchange* (PABX), que em tradução livre do inglês significa Troca Automática de Ramais Privados. Funciona como uma central telefônica. Permite efetuar ligações entre telefones de uma rede interna, ou realizar e receber ligações telefônicas de uma rede externa sem intervenção manual e utilizado como servidor VoIP. O Asterisk é *open source* e distribuído sob a licença *General Public License* (GPL) v2. Até a data de escrita deste trabalho o projeto é mantido pela empresa Digium e pela comunidade (DIGIUM, 2017).

De acordo com a documentação do Asterisk, seu projeto é modular provendo maior flexibilidade para os administradores, pois cada módulo pode ser configurado, habilitado e desabilitado separadamente. Além disso, é um programa com muitos componentes, com relacionamentos complexos. A Figura 7 apresenta a arquitetura geral do Asterisk com os principais componentes. Observa-se que o Asterisk possui um núcleo e os módulos: *Channel Drivers*, APIs, PBX, *Configuration*, *DB Backends*, *Reporting*, *Logging* e a Interface de linha de comando.

O núcleo, é o componente essencial que fornece uma grande quantidade de funcionalidades e infra-estrutura. Entre muitas funções, lê os arquivos de configuração, incluindo o *dialplan* e carrega todos os outros módulos que fornecem mais funcionalidades. O núcleo carrega e constrói o *dialplan*. O *dialplan* contém uma lista de instruções que o Asterisk deve seguir para saber como lidar com as ligações que entram e saem do sistema de telefonia.

Além da funcionalidade fornecida pelo núcleo do Asterisk, os módulos fornecem todas as outras funcionalidades. O código fonte de muitos módulos é distribuída

Figura 7 – Arquitetura do Asterisk.



Fonte: Adaptado de (DIGIUM, 2017)

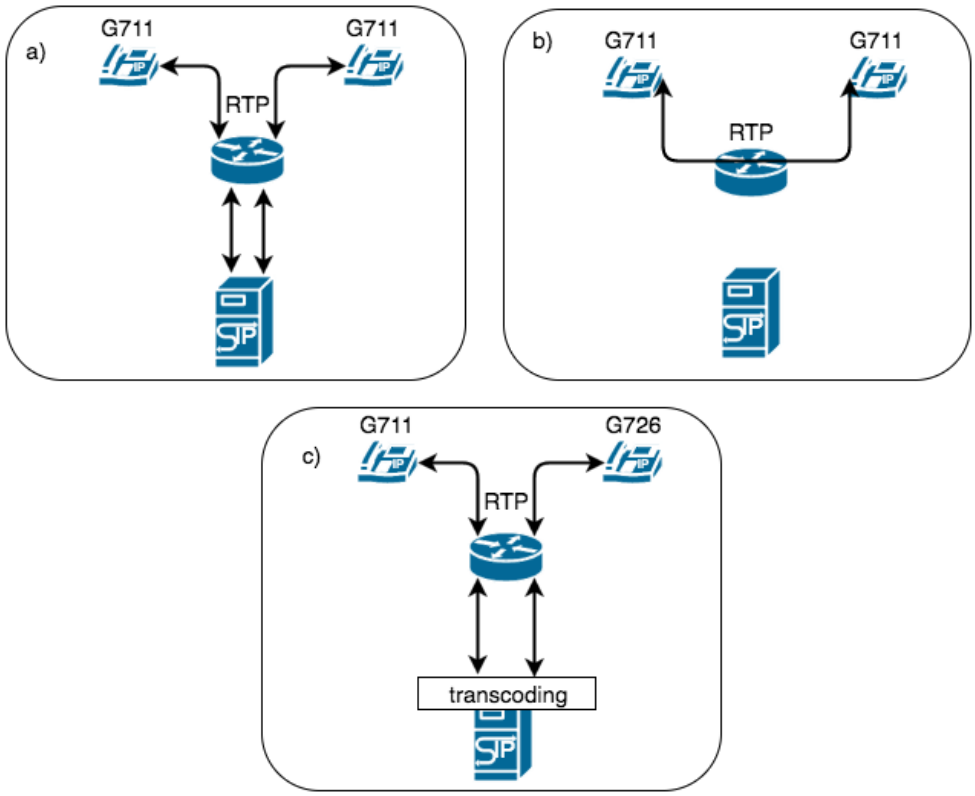
com o Asterisk. Entretanto, alguns módulos podem estar disponíveis a partir de membros da comunidade ou até mesmo por empresas que desenvolvem módulos comerciais. Os módulos podem ser afetados por suas configurações em tempo de carregamento e podem ser carregados ou descarregados em tempo de execução. A maioria dos módulos é configurável independentemente e tem seus próprios arquivos de configuração. Alguns módulos têm suporte para que a configuração seja lida estaticamente ou dinamicamente (em tempo real) a partir de *backends* de banco de dados.

O tráfego de áudio em uma chamada é feito utilizando-se o protocolo RTP, e quando os clientes da sessão VoIP utilizam o mesmo Codec pode ser feito das seguintes maneiras:

- Áudio é trafegado de um cliente para o outro através do servidor VoIP (Figura 8a). Nesse caso o servidor está no caminho do áudio e dois canais são criados no servidor, um para cada cliente. O servidor VoIP lê o áudio de um canal e escreve no outro. Nesse caso há um aumento no uso de memória alocada e no uso de CPU durante a leitura e escrita dos canais;
- Áudio é trafegado de um cliente diretamente para o outro (Figura 8b). Nesse caso o servidor VoIP não está no caminho do áudio e não é possível gravar a chamada, tocar música de fundo e transferir ou colocar a chamada em espera, entretanto, há uma redução na utilização de *Central Processing Unit* (CPU).

Quando os clientes não possuem o mesmo Codec:

Figura 8 – Tráfego RTP.



Fonte: Elaborado pelo autor

- Áudio é trafegado de um cliente para o outro com codificação através do servidor VoIP (Figura 8c). Nesse caso o servidor está no caminho do áudio, e dois canais são criados, um para cada cliente da mesma forma que no caso da Figura 8a. Contudo, o servidor VoIP Asterisk precisa efetuar a tradução dos Codecs, aumentando o uso de memória alocada bem como o uso de CPU, pois antes de fazer a escrita nos canais é necessário traduzir os dados transmitidos de acordo com os diferentes Codecs utilizados para que os clientes possam receber e ouvir o áudio.

No Asterisk, um canal é um caminho de comunicação entre um ponto e o próprio Asterisk. O caminho de comunicação abrange todas as informações passadas para e do ponto final. Isso inclui tanto a sinalização bem como mídia (o áudio ou vídeo sendo enviado/recebido para/a partir do ponto final).

2.3 DEFINIÇÃO DO PROBLEMA

Dada a descrição e fundamentação do estabelecimento e funcionamento de chamadas VoIP entre clientes, pode-se notar que, por padrão, mesmo quando os cli-

entes utilizam o mesmo Codec, o tráfego passa por um servidor VoIP. Mesmo nesse caso particular, no qual não há necessidade de transcodificação do áudio, há um consumo de CPU por parte do servidor VoIP, bem como utilização de largura de banda no segmento em que o mesmo se encontra, que poderiam ser economizados. Assim, há uma oportunidade de melhoria nesses quesitos, constituindo-se dessa forma em um problema a ser resolvido.

Apesar de, tradicionalmente, ser possível configurar o servidor VoIP para que habilite o tráfego VoIP diretamente entre clientes que utilizem o mesmo Codec, tal procedimento apresenta custo operacional (reconfiguração dos servidores VoIP, reconfiguração dos clientes, manutenção das bases de dados), não sendo escalável. A solução apresentada neste trabalho elimina esse custo operacional, pois não há necessidade de se fazer qualquer reconfiguração.

Além disso, um dos problemas mais comuns em sistemas VoIP está relacionado com a habilidade de cruzar *firewalls* e servidores *Network Address Translation* (NAT) (KARAPANTAZIS; PAVLIDOU, 2009; LIN et al., 2010; PARK et al., 2008; CHEN; HUANG; CHAO, 2008; YERYOMIN; EVERS; SEITZ, 2008; KHLIFI; GREGOIRE; PHILLIPS, 2006; KHIRUL; ISLAM; HASAN, 2007), entre outros. Esse problema afeta diretamente o tráfego de áudio. Segundo a documentação do Asterisk (DIGIUM, 2017), quando há NAT, *firewall* ou outro elemento na rede que esconda o IP real do cliente, todo o tráfego de áudio entre os clientes é feito passando pelo servidor VoIP. Isso faz com que ocorra utilização de largura de banda no segmento do servidor VoIP que poderia ser economizada caso o tráfego de áudio fosse realizado diretamente entre os clientes.

A localização dos clientes na rede também pode ser um problema, mesmo em uma rede LAN ou clientes localizados em domínios administrativos diferentes, podem estar protegidos por NAT, *firewall* e *proxy*, entre outros, pois o servidor VoIP não é capaz de resolver os endereços IPs privados dos clientes em uma chamada. Novamente, isso faz com que o tráfego de áudio entre os clientes seja feito através do servidor VoIP.

O redirecionamento do tráfego de áudio para que ocorra diretamente entre os clientes, pode fazer com que ocorra redução no consumo de largura de banda no servidor VoIP, além de reduzir a quantidade de dados trafegados no segmento de rede onde está localizado o servidor VoIP. Além disso, o redirecionamento de tráfego, entre clientes que utilizem o mesmo Codec, visa diminuir o consumo de CPU do servidor VoIP, pois mesmo não havendo necessidade de transcodificação, há necessidade do servidor VoIP ler os dados de um cliente e enviá-los para o outro.

A diminuição do consumo de CPU e de largura de banda do servidor VoIP,

possibilita o atendimento de mais chamadas, sem a necessidade de aumentar a infraestrutura de processamento (*hardware*), ou seja, sem incremento no custo de capital.

Em suma, os principais problemas relacionados com VoIP, e que são tratados por esse trabalho são listados da seguinte maneira:

- A configuração dos clientes para que utilizem o mesmo Codec;
- A configuração do servidor VoIP para que por padrão tente fazer o tráfego direto entre os cliente;
- Firewall e NAT fazem com que o áudio das chamadas sejam trafegados passando pelo servidor VoIP;
- Utilização de CPU no servidor VoIP; e
- Utilização de largura de banda no servidor VoIP.

A preocupação com esses problemas é importante principalmente para o provisionamento e dimensionamento do serviço VoIP. Nesse sentido, o Capítulo 3 apresenta a proposta deste trabalho para reduzir o consumo de recursos em um servidor VoIP, mais especificamente reduzindo a utilização de CPU no servidor VoIP e na utilização largura de banda no servidor VoIP sem a necessidade de reconfigurar/instrumentar os cliente e o servidor VoIP, utilizando como solução o redirecionamento automático de tráfego baseado em SDN.

2.4 CONSIDERAÇÕES PARCIAIS

Este capítulo apresentou a revisão de literatura deste trabalho. Foram abordados os conceitos do paradigma SDN com uma breve descrição da arquitetura SDN, componentes e características. O protocolo OpenFlow é um protocolo aberto que padroniza a comunicação entre as camadas de controle e de dados. São apresentadas as tabelas de fluxos do protocolo OpenFlow e a classificação atualmente aceita de *switches* compatíveis com OpenFlow. O controlador SDN que é o responsável por adicionar, atualizar e remover registros da tabelas de fluxos. A tecnologia VoIP também é apresentada, assim como o servidor PABX Asterisk e sua arquitetura. Foram abordados também os protocolos *Session Initiation Protocol* (SIP), *Realtime Transport Protocol* (RTP) e *Realtime Transport Control Protocol* (RTCP). A definição de Codecs de voz e suas características também foram abordadas. Por fim, é apresentada a definição do problema enfrentado neste trabalho, relacionado com a redução da utilização de recursos por parte do servidor VoIP.

3 PROPOSTA

O objetivo deste trabalho é o desenvolvimento de uma solução baseada em Redes Definidas por Software para reduzir o consumo de recursos em um servidor VoIP. Desta forma, este capítulo apresenta a formulação da solução proposta, na forma de um módulo de *software* chamado SDNVoIP. Para tanto, são introduzidos os requisitos funcionais e não funcionais necessários para compreender as necessidades da solução, bem como a descrição da arquitetura proposta, além da definição dos testes necessários para avaliar a solução implementada. Por fim, a Seção 3.3 apresenta os trabalhos relacionados de forma esquematizada, com o intuito de demonstrar e justificar a lacuna existente na utilização de SDN para o redirecionamento de tráfego em chamadas VoIP, como solução para a redução de recursos em servidor VoIP.

Para que o módulo SDNVoIP se torne operacional, ou seja, realize a transferência dos dados de voz diretamente entre os clientes, é considerado um ambiente de redes SDN com alguns pré-requisitos:

- Existência de ao menos um controlador SDN - OpenFlow capaz de acessar os elementos de rede, como *switches* OpenFlow, com a finalidade de obter informações e gerenciar as tabelas de fluxos;
- Possuir ao menos um servidor VoIP devidamente configurado;
- Possuir clientes VoIP devidamente configurados para a utilização dos mesmos Codecs; e
- Possuir uma lista de IPs dos servidores VoIP disponíveis.

A solução SDNVoIP é capaz de operar tanto em chamadas nas quais os clientes empreguem o mesmo Codec na origem/destino como com Codecs diferentes. Contudo, o redirecionamento do tráfego da chamada é possível de implementar quando a origem e destino empregarem o mesmo Codec. Além dos pré-requisitos citados, existem requisitos que devem ser atendidos. Os requisitos para atingir o objetivo e conseguir reduzir o consumo de recursos em um servidor VoIP são divididos em funcionais e não funcionais.

3.1 REQUISITOS FUNCIONAIS

Como requisitos funcionais do módulo SDNVoIP foram identificados:

- RF1. Analisar e identificar tráfego SIP/RTP e RTCP a fim de obter informações sobre os clientes;
- RF2. Identificar chamadas VoIP entre os clientes;
- RF3. Identificar chamadas que utilizem o mesmo Codec;
- RF4. Instalar fluxos nos *switches* OpenFlow para redirecionar o tráfego de áudio diretamente entre os clientes que utilizem o mesmo Codec; e
- RF5. Remover os fluxos após o término das chamadas;

3.2 REQUISITOS NÃO FUNCIONAIS

Como requisitos não funcionais, foram especificados:

- RNF1. Utilizar SDN na solução;
- RNF2. Não prejudicar a comunicação entre os clientes e o servidor VoIP.
- RNF3. Reduzir a utilização de CPU no servidor VoIP;
- RNF4. Reduzir a largura de banda no servidor VoIP; e
- RNF5. Permitir tráfego direto entre os clientes.

3.2.1 Descrição da Solução Proposta

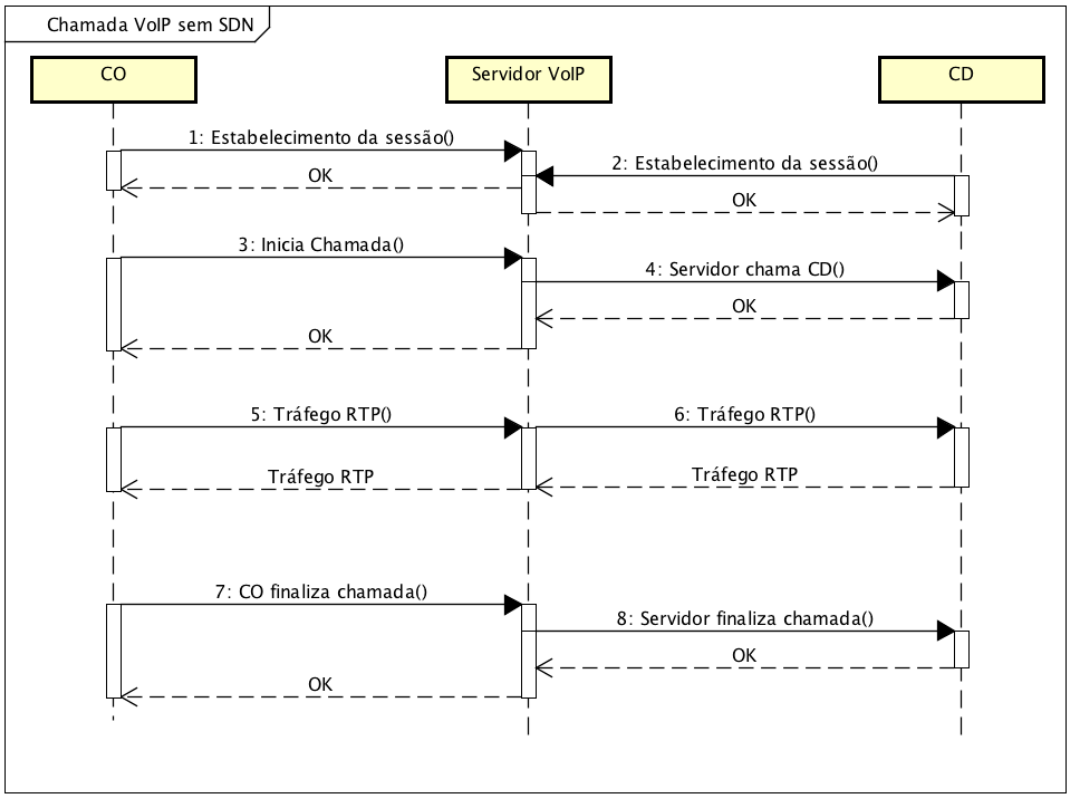
O funcionamento da solução proposta tem como objetivo reduzir o consumo de recursos em um servidor VoIP. Para alcançar o objetivo, o módulo proposto atua fazendo com que chamadas entre clientes que utilizem o mesmo Codec não trafeguem áudio pelo servidor VoIP. Vale ressaltar que não há necessidade de reconfigurar ou instrumentar o(s) servidor(es) VoIP(s), nem os clientes e também não há a necessidade de outros servidores na rede.

Para que o tráfego de voz trafegue diretamente entre os clientes sem passar pelo servidor VoIP, o tráfego tem que ser desviado no momento em que passa pelo *switch*, o qual faz um redirecionamento do tráfego. Para que isso ocorra, é necessária a instalação de regras adequadas na tabela de fluxos do *switch* OpenFlow.

É importante ressaltar o funcionamento de uma chamada VoIP. A Figura 9 apresenta o diagrama de sequência de uma chamada VoIP entre Cliente Origem (CO) e Cliente Destino (CD) sem a utilização de SDN, ambos utilizando o mesmo Codec. Assim, é possível observar que o tráfego de áudio se dá através do servidor VoIP. A sequência ocorre da seguinte maneira:

- 1 e 2 - CO e CD estabelecem a sessão SIP;
- 3 - CO inicia uma chamada;
- 4 - Servidor chama CD;
- CD envia uma resposta OK e servidor envia resposta OK para CO;
- 5 - CO envia áudio para o servidor;
- 6 - Servidor envia áudio de CO para CD;
- CD envia áudio para o servidor que, por sua vez, envia para CO;
- 7 - CO finaliza a chamada;
- 8 - Servidor finaliza a chamada com CD; e
- Por fim, a mensagem OK é enviada.

Figura 9 – Diagrama de sequência de uma chamada VoIP sem SDN.



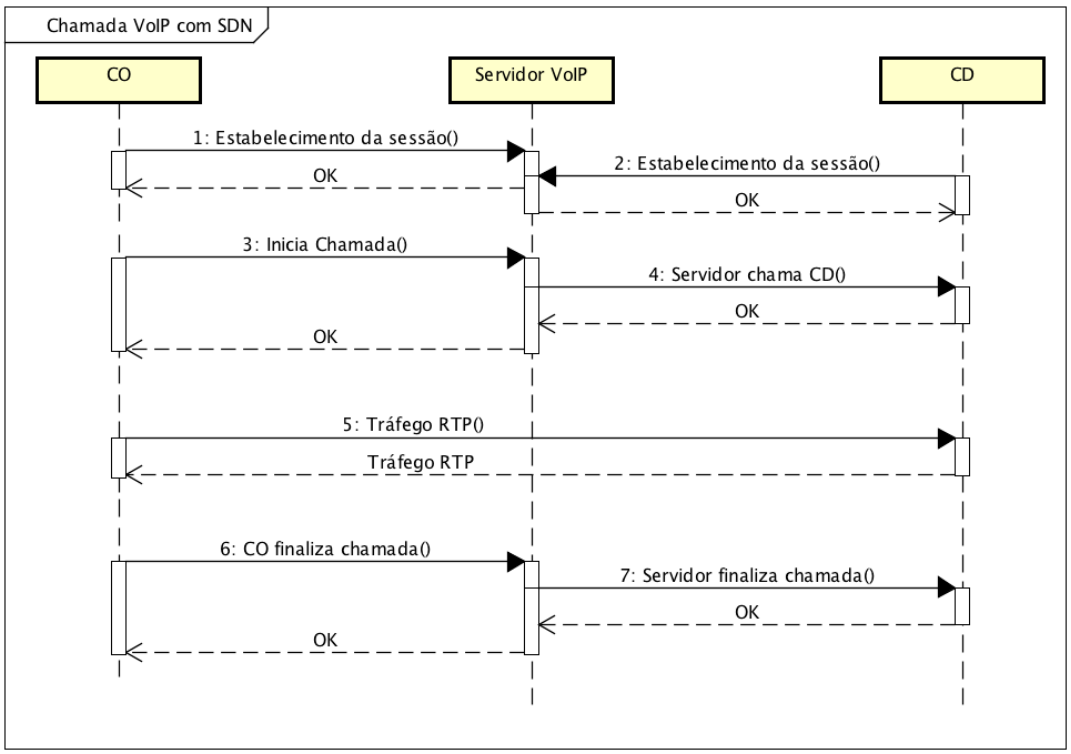
Fonte: Elaborado pelo autor

Já a Figura 10 apresenta o diagrama de sequência de uma chamada VoIP entre o CO e o CD com a utilização de SDN, ambos utilizando o mesmo Codec. Porém,

nesse caso, é possível observar que o tráfego de áudio é feito diretamente entre CO e CD, e ocorre da seguinte maneira:

- 1 e 2 - CO e CD estabelecem a sessão SIP;
- 3 - CO inicia uma chamada;
- 4 - Servidor chama CD;
- CD envia uma resposta OK e servidor envia resposta OK para CO;
- 5 - Tráfego direto de áudio entre CO/CD e vice-versa;
- 6 - CO finaliza a chamada;
- 7 - Servidor finaliza a chamada com CD; e
- Por fim, a mensagem OK é enviada.

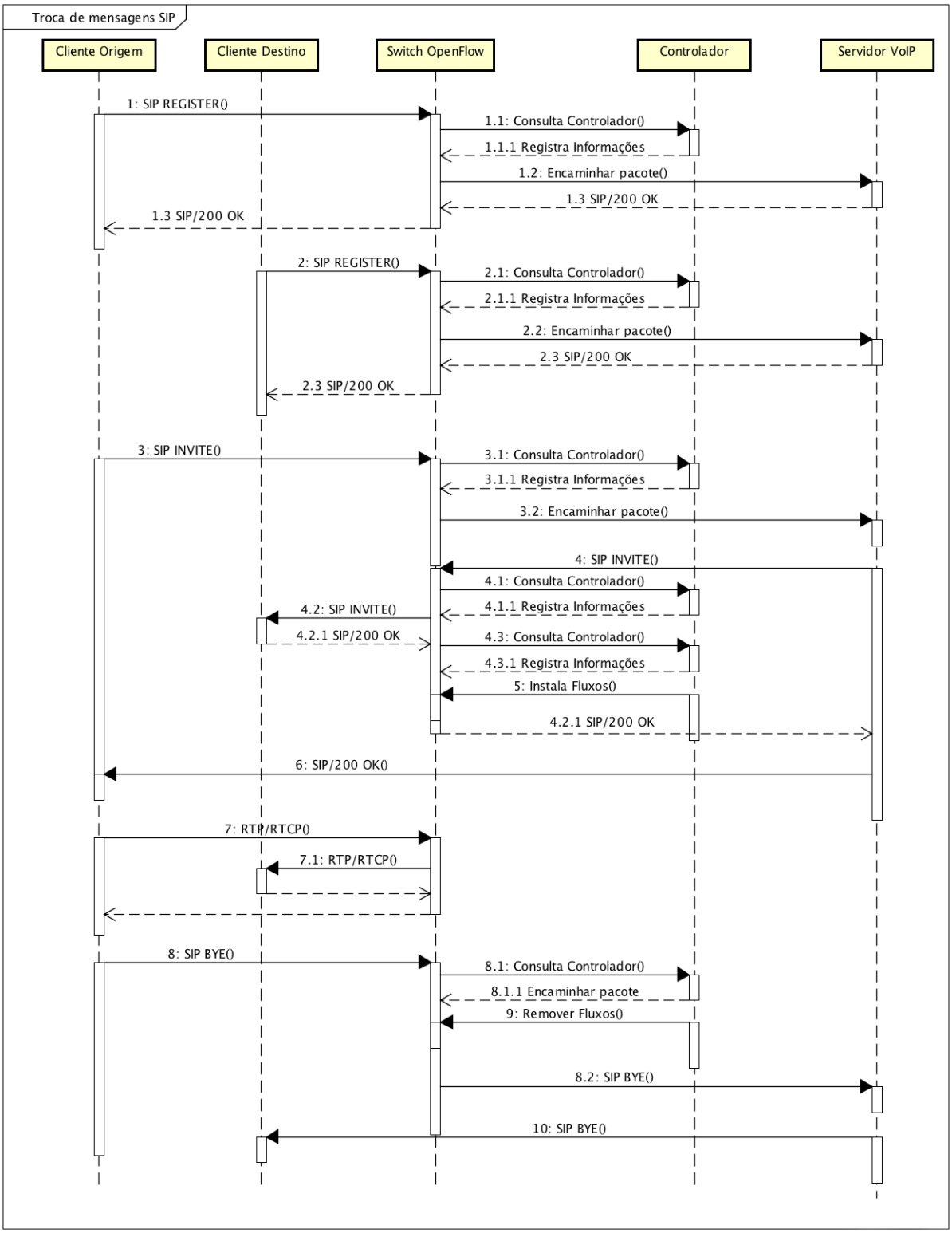
Figura 10 – Diagrama de sequência de uma chamada VoIP com SDN.



Fonte: Elaborado pelo autor

A Figura 11 apresenta o diagrama de sequência com a troca de mensagens SIP e eventos que ocorrem em uma chamada VoIP com a utilização de SDN e com o módulo SDNVoIP ativo.

Figura 11 – Sequência de troca de mensagens com o módulo SDNVoIP.



Fonte: Elaborado pelo autor

O diagrama de sequência envolve os elementos de infraestrutura de rede necessários (*switch* e controlador OpenFlow), bem como os atores envolvidos na chamada (cliente origem/destino e servidor VoIP), além de apresentar uma visão onde se pode reconhecer a interação dos elementos de rede com os atores, através da abordagem OpenFlow para SDN, com enfoque na atuação do controlador SDN, no qual o módulo SDNVoIP encontra-se. Assim, tem-se que:

1. A primeira mensagem SIP enviada pelos clientes é uma mensagem SIP REGISTER (1 e 2), ou seja, é a autenticação dos clientes no servidor VoIP;
2. O *switch* OpenFlow recebe o pacote SIP/REGISTER e consulta o controlador (1.1 e 2.1);
3. O Controlador deve analisar os pacotes e obter os endereços IPs e MACs dos clientes e armazenar estas informações em memória (1.1.1 e 2.1.1);
4. O controlador responde ao *switch* para encaminhar o pacote, o qual é encaminhado para o servidor VoIP (1.2 e 2.2);
5. O servidor VoIP envia uma mensagem SIP/200 OK para cada cliente (1.3 e 2.3);
6. Uma chamada é iniciada por CO com a mensagem SIP INVITE (3);
7. O *switch* OpenFlow recebe o pacote SIP/INVITE e consulta o controlador (3.1);
8. O controlador deve analisar o pacote e obter o Codec e a porta RTP que CO irá utilizar. Em seguida, deve-se armazenar estas informações em memória (3.1.1);
9. O *switch* então encaminha o pacote para o servidor VoIP (3.2);
10. O servidor VoIP envia uma mensagem SIP/INVITE para o CD (4);
11. O pacote SIP/INVITE é recebido no *switch*, que por sua vez, consulta o controlador novamente (4.1);
12. O Controlador analisa o pacote e deve obter a porta RTP que o servidor reservou para CD (4.1.1);
13. O *switch* encaminha a mensagem SIP/INVITE para o CD (4.2);
14. O CD envia uma mensagem SIP/200 OK (4.2.1);
15. O *switch* recebe a mensagem SIP/200 OK de CD e consulta o controlador (4.3);
16. O controlador deve analisar o pacote e obter o Codec e a porta RTP utilizado por CD e armazenar as informações em memória (4.3.1);

17. O *switch* encaminha a mensagem SIP/200 OK de CD para o servidor VoIP (4.2.1);
18. Quase ao mesmo tempo, o controlador possui em memória todas as informações necessárias para a instalação dos fluxos e envia um comando para o *switch* com os fluxos que devem ser instalados na tabela de fluxo (5);
19. O servidor envia uma mensagem SIP/200 OK para CO (6);
20. O tráfego de áudio é então iniciado, porém, o *switch* deve possuir os fluxos instalados possibilitando a correspondência; dos pacotes de áudio. Nesse caso, o *switch* redireciona os pacotes entre CO e CD diretamente (7 e 7.1);
21. A chamada é terminada com uma mensagem SIP/BYE enviada pelo CO (8);
22. A mensagem SIP/BYE é recebida no *switch* que consulta o controlador (8.1);
23. O controlador deve analisar a mensagem SIP/BYE e responder para o *switch* encaminhar a mensagem (8.1.1);
24. Quase ao mesmo tempo, o controlador envia um comando para o *switch* remover os fluxos referentes à chamada que estava ocorrendo entre CO e CD (9); e
25. A chamada é então finalizada (10).

Observa-se que a redução do consumo de recursos no servidor VoIP é obtida com o item 7 e 7.1 do diagrama 11. Assim, quando o CO envia pacotes RTP para o servidor VoIP, o *switch* recebe os pacotes e faz a correspondência dos pacotes e os redireciona para CD e vice-versa. Dessa maneira não há trafego de áudio no servidor VoIP reduzindo-se assim, a utilização de CPU e largura de banda.

Com a descrição da solução proposta e com os requisitos definidos, é necessário o desenvolvimento de um plano de testes. O testes serão executados com o objetivo de avaliar a solução proposta.

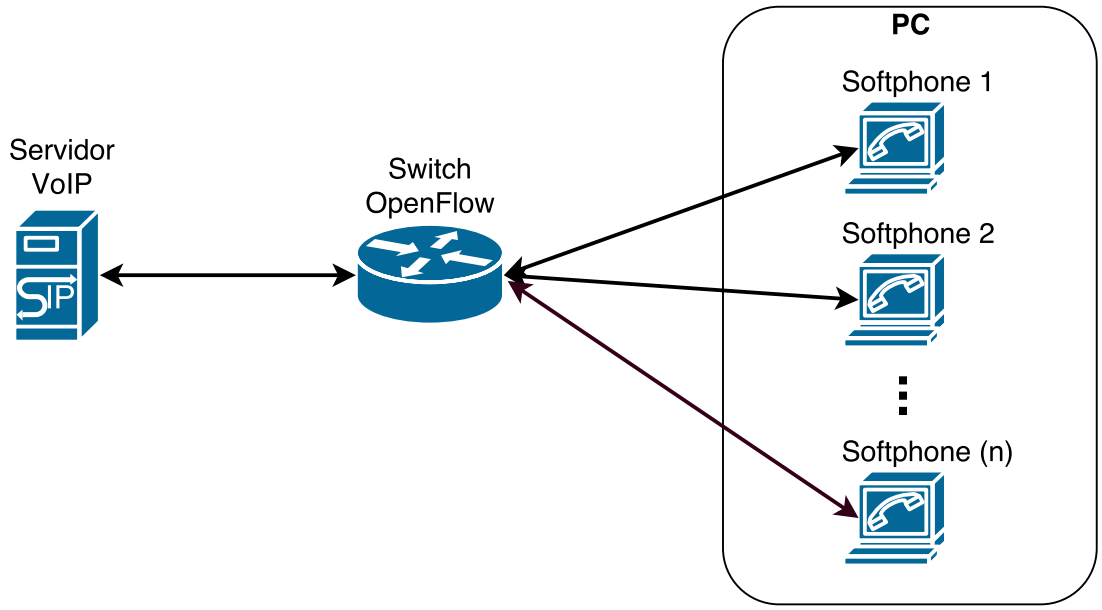
3.2.2 Plano de Testes

No plano de testes são definidos os experimentos a serem executados para avaliar a solução proposta. O plano de testes está dividido em dois cenários, cada um com cargas diferentes (quantidade de chamadas simultâneas). Um cenário sem a utilização do paradigma SDN e outro com a utilização de SDN com a solução proposta ativa, ambos em ambiente LAN.

O cenário sem SDN apresentado na Figura 12 possui um servidor VoIP, um *switch* e um PC no qual devem ser inicializadas várias instâncias de um *softphone*

representando os clientes das chamadas. Já o cenário com SDN, apresentado na Figura 13, possui um servidor VoIP, um *switch* OpenFlow, um controlador SDN e um PC no qual devem ser inicializadas várias instâncias de um *softphone* representando os clientes das chamadas.

Figura 12 – Topologia de testes sem SDN.

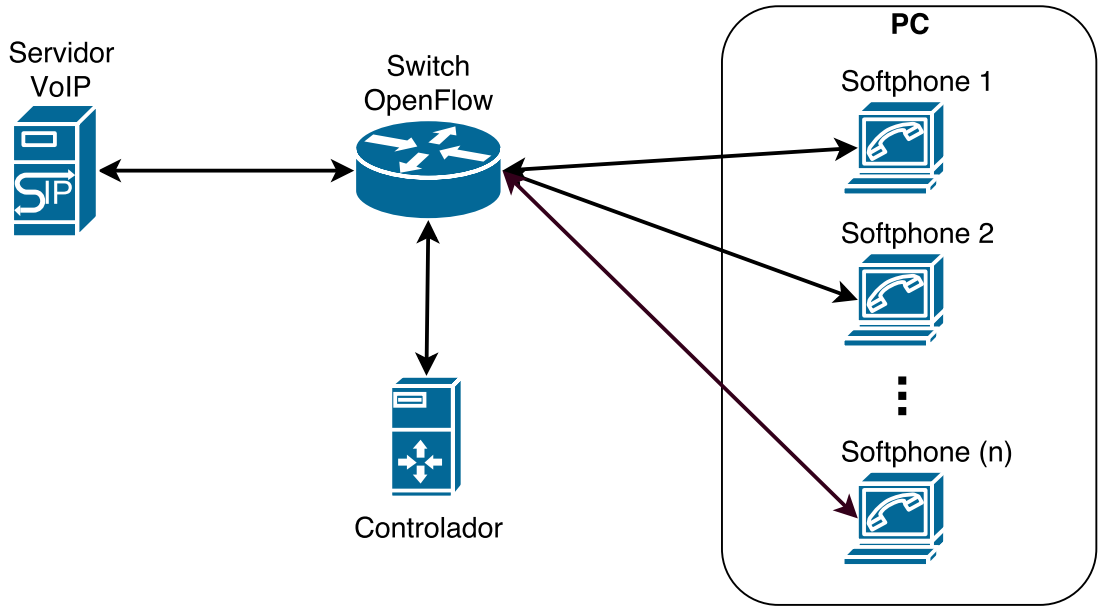


Fonte: Elaborado pelo autor

A execução de chamadas simultâneas com cargas de 50, 100, 150, 200, 250 e 300 chamadas simultâneas devem ser realizadas em ambos os cenários, com e sem a utilização de SDN com o módulo SDNVoIP. As cargas foram definidas inicialmente em 50 chamadas até o limite de 300 chamadas devido à limitação do equipamento utilizado nos experimentos e para minimamente estabelecer relevância estatística. No cenário sem SDN é esperada uma maior utilização de CPU e largura de banda no servidor VoIP, consequentemente maior quantidade de dados trafegados. Ambos os cenários devem ser experimentados com clientes utilizando o mesmo Codec.

Já no cenário com o uso de SDN é esperado uma redução na utilização de CPU e largura de banda do servidor VoIP, além da consequente redução na quantidade de dados trafegados no segmento de rede do servidor. Essa redução é devida ao redirecionamento do tráfego de áudio entre os clientes, que resulta da atuação do módulo SDNVoIP na instalação de fluxos no *switch* OpenFlow, fazendo com que o áudio das chamadas entre clientes que utilizam o mesmo Codec, não trafegue pelo servidor VoIP fluindo diretamente entre os clientes.

Figura 13 – Topologia de testes com SDN.



Fonte: Elaborado pelo autor

3.3 TRABALHOS RELACIONADOS

Nesta seção são identificados e discutidos os principais trabalhos que visam alguma melhoria/otimização em tráfego VoIP, particularmente relacionados com o uso de SDN com o objetivo de melhorar a qualidade de chamadas VoIP, evitar congestionamentos na rede ou fazer a troca de Codecs de forma adaptativa quando ocorrem congestionamento e de forma indireta reduzem a utilização de CPU e ou largura de banda no servidor VoIP. Estão relacionados também, trabalhos que não utilizam SDN, entretanto tem como objetivo melhorar o desempenho e dimensionamento de servidores VoIP.

Kwon et al. (2014) propuseram uma arquitetura de serviço em redes SDN chamada de *adaptive Mobile Voice over Internet Protocol* (mVoIP) com o objetivo de melhorar a qualidade de serviços VoIP para usuários móveis. Um dos pontos chave é melhorar a *Quality of Experience* (QoE), melhorando o caminho dos dados com fluxos VoIP e seleção otimizada de Codec considerando congestionamentos na rede. A arquitetura proposta possui agentes chamados mVoIP *QoE measurement agents*, são implantados em redes sem fio e tem como objetivo coletar informações de QoS. Outro elemento da arquitetura é o *mVoIP QoS Manager*, uma aplicação SDN que coleta dados dos mVoIP *QoE measurement agents* e de *switches OpenFlow* em redes SDN. O mVoIP *QoS Manager* analisa os dados coletados e decide qual o melhor caminho

para os fluxos mVoIP e qual o Codec mais adequado para as condições de estado da rede. Contudo, é apresentado apenas uma proposta de arquitetura, e como trabalhos futuros serão realizados testes experimentais. O principal diferencial do trabalho apresentado nesta dissertação, é a preocupação com a redução de utilização de recursos em um servidor VoIP. Outro detalhe é que a proposta de Kwon et al. (2014) tem como alvo redes móveis, a solução apresentada nessa dissertação, o SDNVoIP, atua em redes LAN podendo ser adaptado para outras redes como as redes móveis.

Maribondo e Fernandes (2016) propuseram um novo método de Adaptação de Codec em redes SDN (*Codec Adaptation over SDN* - CAoS). O trabalho é baseado na arquitetura SDN e, segundo os autores, implementa uma abordagem eficiente para evitar a degradação da voz em redes corporativas, reduzindo a qualidade da voz em chamadas utilizando um Codec de menor largura de banda quando ocorre congestionamentos. O CAoS é composto de três módulos. Um módulo para monitoramento do tráfego, um para as decisões e um módulo para as mensagens SIP. De acordo com os resultados obtidos, o CAoS permite que uma quantidade maior de chamadas possam ser efetuadas em períodos de muita utilização sem degradar a qualidade da voz para valores MOS (Subseção 2.2.2) muito piores (MARIBONDO; FERNANDES, 2016). Diferente da proposta deste trabalho, o CAoS não tem como preocupação a utilização de recursos no servidor VoIP.

De acordo com Hasrouty et al. (2016) atingir alta qualidade em chamadas em conferência pela Internet é uma tarefa difícil, dispositivos móveis heterogêneos e redes dinâmicas devem ser adequadamente gerenciados por sistemas VoIP multipartes para garantir uma boa qualidade de experiência. Para Hasrouty et al. (2016), um sistema VoIP multipartes é uma sessão ou uma chamada entre três ou mais clientes chamados participantes ou partes. Os autores propõem um sistema VoIP multipartes baseado na tecnologia SDN que utiliza distribuição *multicast* e adaptação dinâmica de fluxos para otimizar a qualidade de uma chamada em conferência para cada participante. Com o uso de SDN, as redes podem ser observadas globalmente a partir de uma unidade central. Uma vez que a topologia de rede e as condições dos canais são conhecidas, o MVoIP atua monitorando caminhos com menor latência criando fluxos nesses caminhos e entregando taxas de *bit rates* dinâmicas. Observa-se que a preocupação está na qualidade de experiência e no estado da rede, além de ter como alvo chamadas em conferência. A solução apresentada nesta dissertação tem como alvo servidores VoIP, além de, como já citado, pode também reduzir o tráfego evitando congestionamentos no segmento do servidor VoIP.

Thorpe et al. (2016) propuseram uma solução baseada no uso de SDN para fornecer um serviço avançado de monitoramento intermediário de chamadas VoIP com a capacidade de detectar e localizar problemas de qualidade para o tráfego

VoIP. A solução é chamada de *Intermediate Mean Opinion Score* (iMOS) e foi desenvolvida como um módulo do controlador Floodlight (FLOODLIGHT, 2017), além do módulo para Floodlight a solução utiliza uma versão modificada do *Open Virtual Switch* (OVS) (VSWITCH, 2017) para coletar a métrica proposta também chamada de métrica iMOS. O módulo iMOS coleta métricas de QoS que podem ser utilizadas para calcular o MOS de nós intermediários (versão modificada do OVS) no caminho de um fluxo VoIP, localizando a origem da degradação do QoS e permitindo ações corretivas. Os nós intermediários coletam informações de pacotes RTCP para calcular a métrica iMOS e enviá-las para o módulo no Floodlight. O módulo no Floodlight possui uma interface gráfica que mostra todos os valores coletados pelos nós intermediários além de permitir ligar e desligar a coleta das informações. Entretanto, a principal diferença é a necessidade de uma versão modificada do OVS, além da solução proposta ser uma ferramenta que apenas monitora chamadas e apresenta as informações em uma interface gráfica. O módulo SDNVoIP pode ser utilizado sem a necessidade de instrumentar o OVS e nem os clientes, além de não ser necessário realizar configurações especificas no servidor VoIP.

Ng, Hoh e Singh (2005) propuseram e avaliaram um algoritmo de comutação de Codec adaptativo para aplicações VoIP em redes sem fio. A troca de Codecs durante a chamada ativa é alcançado renegociando a sessão de áudio usando uma mensagem SIP/SDP REINVITE. Para avaliar o algoritmo proposto, um servidor *proxy* SIP foi implantado em uma rede real e os clientes SIP foram conectados a uma rede sem fio. Para simular o congestionamento da rede, foi utilizado um emulador de tráfego de rede. De acordo com os pesquisadores a proposta obteve sucesso com um aumento da pontuação MOS ao alternar entre os Codecs PCMU e GSM. Diferente do SDNVoIP, o trabalho apresentado por Ng, Hoh e Singh (2005) precisa de um servidor *proxy* que renegocia o Codec utilizado em caso de congestionamentos, além disso a solução não utiliza SDN e novamente não se preocupa com a utilização de recursos do servidor VoIP.

Alguns trabalhos propuseram uma forma de controle adaptativo de *bit rate* para pacotes de áudio (ROYCHOUDHURI; AL-SHAER, 2004), (QIAO et al., 2004), porém é necessário a utilização de uma ferramenta para analisar a qualidade do áudio, além de não utilizarem SDN.

Costa et al. (2015) investigam a adequação do servidor PABX Asterisk para fornecer capacidades de comunicação VoIP com MOS aceitável para um grande número de chamadas. A métrica de probabilidade de bloqueio é usado para medir a capacidade do servidor PBX enquanto o MOS é utilizado para avaliar a qualidade das chamadas de voz. Os resultados do trabalho mostram que o servidor PBX Asterisk pode efetivamente lidar com 165 chamadas de voz simultâneas com uma probabi-

lidade de bloqueio de menos de 5%, proporcionando chamadas de voz com MOS acima da média 4. Ainda em Costa et al. (2015), o MOS é medido por uma ferramenta chamada VoIPMonitor. O VoIPMonitor observa o tráfego VoIP e faz o cálculo do MOS seguindo a recomendação da ITU-T (UNION, 2005a). O SDNVoIP reduz a utilização de recursos no servidor VoIP, redirecionando chamadas que utilizem o mesmo Codec, porém não se preocupa com a qualidade das chamadas. A redução na utilização de recursos no servidor permite um redimensionamento do servidor VoIP e uma maior quantidade de chamadas podem ser suportadas. Apesar de não estar diretamente relacionado, o estudo feito por Costa pode ser realizado com o SDNVoIP e comparado os resultados com e sem a utilização de SDN.

Egilmez et al. (2012) propuseram o OpenQoS, que é um controlador desenhado para entrega de conteúdo multimídia com QoS fim-a-fim. O OpenQoS coleta informações atualizadas do estado da rede para recalcular as rotas e ajustar o QoS dinamicamente. O OpenQoS leva em consideração os valores de atraso, largura de banda e a taxa de perda de pacote para cada *link*. A solução foi testada em um cenário de *streaming* video entre um cliente e um servidor. Segundo os autores, o OpenQoS pode garantir a entrega contínua de vídeo com pouco ou nenhum distúrbio experimentado pelos usuários finais, mesmo se um protocolo de transporte não confiável, como o UDP, for utilizado. De forma similar, Yu, Wang e Hsu (2015) propuseram uma abordagem de roteamento adaptável para transmissão de video com QoS em redes SDN chamado de *Adaptive Routing Approach for Video Streaming* (ARVS). Nessa abordagem, pacotes de camada base e pacotes de *streaming* de vídeo são tratados separadamente, com dois níveis de fluxo de QoS (nível 1 e nível 2 de QoS), o cálculo das rotas é feito com o algoritmo de Dijkstra. Resultados simulados mostraram que, em comparação com o OpenQoS, o ARVS pode reduzir em até 77,3% a taxa de perda de pacotes para a camada base e para pacotes de *streaming* de vídeo há uma melhora de pelo menos 51,4% de cobertura sob várias cargas de rede no caminho mais curto e viável. Apesar de ambas as soluções utilizarem SDN, novamente, ambas não se preocupam com o consumo de recursos no servidor e foram testadas apenas com *streaming* de vídeo entre um servidor e um cliente.

A Tabela 3 apresenta uma comparação entre os trabalhos relacionados de acordo com os requisitos funcionais apresentados na Seção 3.1, sendo que a marcação “sim” significa que o trabalho atende completamente o requisito, caso contrário a coluna apresenta o valor “não”. A terceira coluna da Tabela 3 apresenta “sim” caso o trabalho relacionado analise e identifique todos os protocolos do requisito, caso contrário a coluna apresenta o nome do protocolo suportado.

Observa-se na Tabela 3 que dados os requisitos funcionais apresentados na Seção 3.1, os trabalhos relacionados não atendem completamente aos requisitos,

Tabela 3 – Comparação entre os trabalhos relacionados tendo como base os requisitos.

Trabalho/Requisitos	**RF1**	**RF2**	**RF3**	**RF4**	**RF5**
Kwon et al. (2014)	RTP	não	não	não	não
Maribondo e Fernandes (2016)	SIP	sim	não	não	não
Hasrouty et al. (2016)	RTP	não	não	não	não
Thorpe et al. (2016)	sim	não	não	não	não
Egilmez et al. (2012)	RTP	não	não	não	não
Yu, Wang e Hsu (2015)	RTP	não	não	não	não
Ng, Hoh e Singh (2005)	RTP	não	não	não	não
Roychoudhuri e Al-Shaer (2004)	RTP	não	não	não	não
Costa et al. (2015)	RTP	não	não	não	não

dessa forma, justifica-se a originalidade deste trabalho.

3.4 CONSIDERAÇÕES PARCIAIS

Este capítulo apresentou a proposta deste trabalho, cujo objetivo é o desenvolvimento de uma solução baseada em SDN para reduzir o consumo de recursos em um servidor VoIP. A solução proposta deste trabalho se dá na forma de um módulo de *software* chamado SDNVoIP. Foram definidos os pré-requisitos em um ambiente de redes SDN, assim como foram definidos e apresentados os requisitos funcionais e não funcionais para atingir o objetivo e conseguir reduzir o consumo de recursos em um servidor VoIP. Foi apresentado também a descrição da solução proposta, bem como a descrição da sequência de eventos que ocorrem com a troca de mensagens entre os clientes e o servidor VoIP. Um plano de testes foi apresentado com a definição dos experimentos a serem executados para avaliar a solução proposta e, por fim, os trabalhos relacionados foram elencados e comparados, fornecendo evidências da adequação do trabalho proposto.

4 MÓDULO SDNVOIP

Este capítulo apresenta em detalhes o desenvolvimento do módulo SDNVoIP. O desenvolvimento desse módulo tem como intuito atingir os objetivos elencados na definição inicial do trabalho, particularmente reduzindo o consumo de CPU por parte do servidor VoIP, largura de banda utilizada e consequentemente a quantidade de dados trafegados no segmento de rede onde o está localizado o servidor VoIP.

Como o SDNVoIP foi desenvolvido como um módulo do controlador Floodlight, é necessário compreender a arquitetura do controlador Floodlight, seu funcionamento e seus principais módulos. É apresentado também como o SDNVoIP analisa e trata os pacotes SIP, SIP/SDP, RTP e RTCP, extraindo informações de portas e Codecs utilizados pelos clientes. Caso os clientes efetuem chamadas VoIP e utilizem o mesmo Codec, o SDNVoIP identificará todas as informações necessárias para a instalação de fluxos nos *switches* no caminho do áudio, fazendo com que o tráfego de áudio ocorra diretamente entre os clientes. Por fim, são apresentadas algumas limitações do presente trabalho.

4.1 FLOODLIGHT

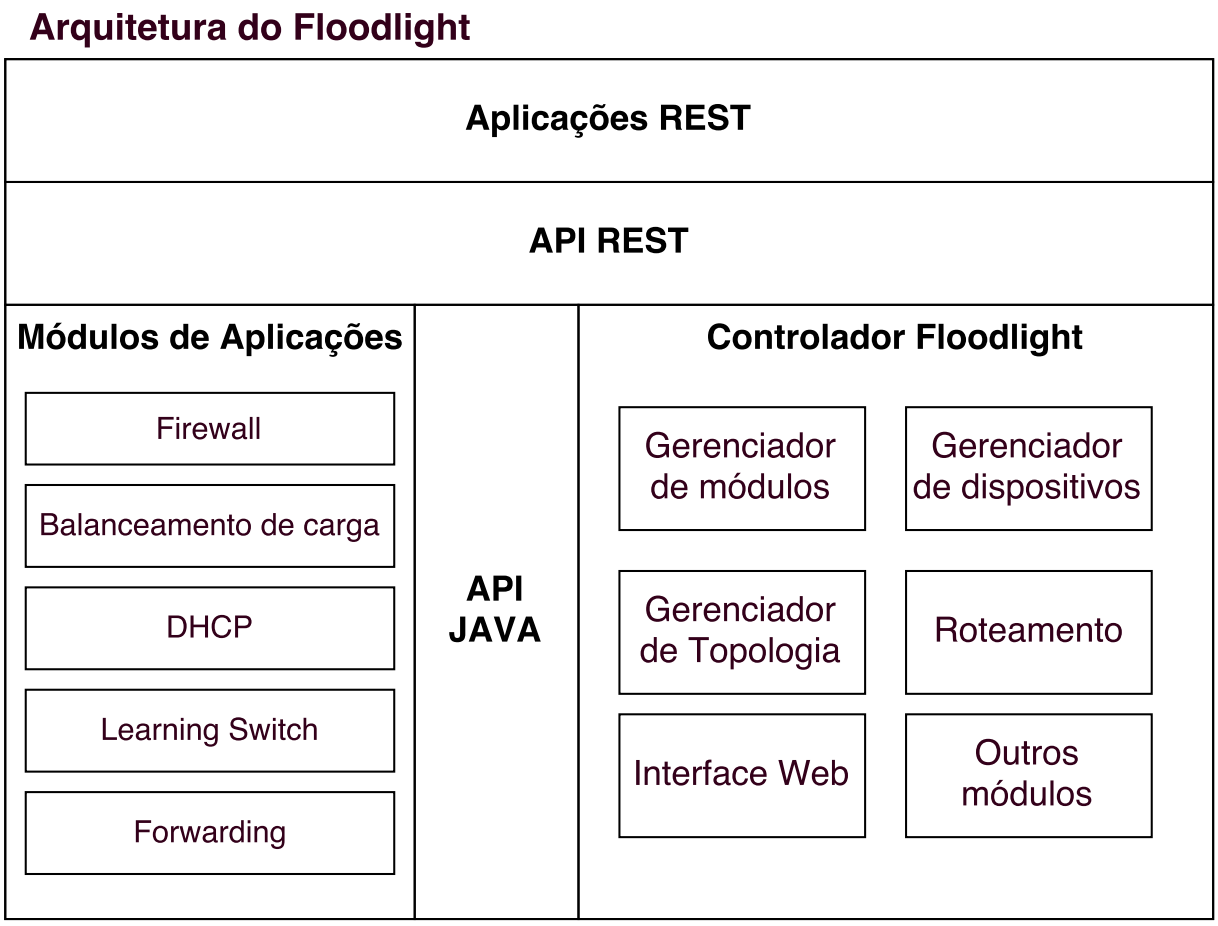
Dentre os controladores existentes, foi selecionado para a realização deste trabalho o controlador *Floodlight* (Requisito Não Funcional 1, Seção 3.2) por apresentar um estágio maduro de desenvolvimento e ser adotado em diversos cenários experimentais (THORPE et al., 2016), (HUMERNBRUM; HAGEDORN; GORLATCH, 2016), (CHIRIVELLA-PEREZ et al., 2015), (PANG; JIANG; LI, 2016), (ARTUSO et al., 2016), (HASROUTY et al., 2016) e (MORZHOV; ALEKSEEV; NIKITINSKIY, 2016) entre outros. O *Floodlight* é um controlador centralizado escrito em linguagem *Java*, que suporta *switches* físicos e virtuais, baseados no protocolo *OpenFlow*. Dessa maneira, como controlador foi utilizado o Floodlight em sua versão 0.9 e adaptado com a existência do módulo SDNVoIP. Além disso, durante os experimentos realizados para validação da proposta, o módulo padrão de encaminhamento e configuração de fluxos (*forwarding*) do *Floodlight* foi desabilitado a fim de obter-se maior controle dos fluxos, além de permitir que o módulo desenvolvido possa analisar todos os pacotes SIP.

O Floodlight (FLOODLIGHT, 2017) não é apenas um controlador OpenFlow. O Floodlight é um controlador OpenFlow e uma coleção de aplicações construídas no topo do Floodlight Controller. De acordo com seu desenvolvedor (FLOODLIGHT, 2017), o Floodlight oferece um sistema de carregamento de módulos que facilita o desenvolvimento de aplicações. Possui fácil configuração com dependências mínimas

e suporta uma ampla gama de *switches* virtuais e físicos (OpenFlow). Pode lidar com redes OpenFlow e não OpenFlow, além de realizar um conjunto de funcionalidades comuns para controlar e investigar uma rede OpenFlow.

Sendo assim, o Floodlight possui uma arquitetura modular para implementar os recursos de um controlador SDN (OpenFlow), além de possuir algumas aplicações para gerenciamento, operação e funcionamento da rede, que estão disponíveis a partir de sua instalação. A Figura 14 apresenta a arquitetura do Floodlight. O controlador Floodlight é composto por vários módulos como o Gerenciador de Módulos, Gerenciador de Topologia, Roteamento, Gerenciamento de Dispositivos, Descoberta de Links, entre outros. O controlador possui uma *Application Programming Interface* (API) para linguagem Java, além de implementar o conceito de Módulos de aplicações, ou seja, cada aplicação de rede desenvolvida é um módulo na arquitetura do Floodlight.

Figura 14 – Arquitetura genérica do controlador Floodlight.



Fonte: Adaptado de Floodlight (ARCHITECTURE, 2017)

O Floodlight fornece algumas aplicações desenvolvidas utilizando uma API para a linguagem Java, entre elas, o Módulo de Encaminhamento (*Forwarding*), *Firewall*, DHCP, *Learning Switch* entre outros. O controlador disponibiliza também uma API *Representational State Transfer* (REST) que permite o desenvolvimento de aplicações em qualquer linguagem de programação que suporte REST. As APIs REST do controlador são disponibilizadas para quaisquer aplicativos REST terceiros, que podem recuperar informações e invocar serviços enviando comandos *Hypertext Transfer Protocol* (HTTP) REST para o controlador.

Quando o controlador Floodlight é executado, um conjunto de aplicativos do módulo de aplicações também são executados, disponibilizando funções que são de uso comum para a maioria das demais aplicações, tais como:

- Descobrir e expôr os estados e eventos de rede (topologia, dispositivos, fluxos);
- Permite a comunicação do controlador com *switches* de rede (protocolo OpenFlow);
- Gerenciar módulos do Floodlight e recursos compartilhados, como armazenamento, por exemplo; e
- Fornecer uma interface web para gerenciamento do controlador.

O Floodlight permite que módulos possam ser carregados, habilitados e desabilitados, além de permitir que a ordem em que são executados possa ser configurada. Essa configuração é feita através de um arquivo de configuração. A Listagem 4.1 apresenta a lista completa de módulos do Floodlight versão 0.9 e a ordem em que são carregados. É importante observar porém, que a execução de cada módulo depende do tipo de serviço que cada um fornece.

O Floodlight possui vários módulos como pode ser observado na Listagem 4.1. Entretanto, alguns merecem ser detalhados, como por exemplo o módulo *Topology Manager* que gerencia a topologia de rede baseado na descoberta de *links* e na conexão com *switches*, ou seja, sua execução somente será realizada quando ocorrem eventos dessa natureza. Outro módulo importante é o módulo de encaminhamento *Forwarding*, que realiza o encaminhamento de pacotes entre dois dispositivos e vem habilitado por padrão.

Uma vez que o Floodlight foi projetado para funcionar em redes que contenham tanto *switches* OpenFlow como não-OpenFlow, o encaminhamento deve levar em consideração essas características. Assim, o módulo de encaminhamento possui um algoritmo que encontrará todas as ilhas OpenFlow que possuem pontos de conexão tanto para os dispositivos de origem como de destino. Fluxos serão então ins-

talado ao longo do caminho mais curto nos *switches* OpenFlow. Se uma mensagem *PACKET_IN* for recebida em uma ilha e não houver nenhum ponto de conexão para o dispositivo nessa ilha, o pacote será encaminhado para todas as portas de saída do *switch*. Em outras palavras, esse módulo é importante no sentido de que sua utilização habilita o funcionamento do *switch* OpenFlow em modo híbrido, ou seja, permite o funcionamento de uma rede legada não OpenFlow como se não houvessem *switches* e controladores OpenFlow.

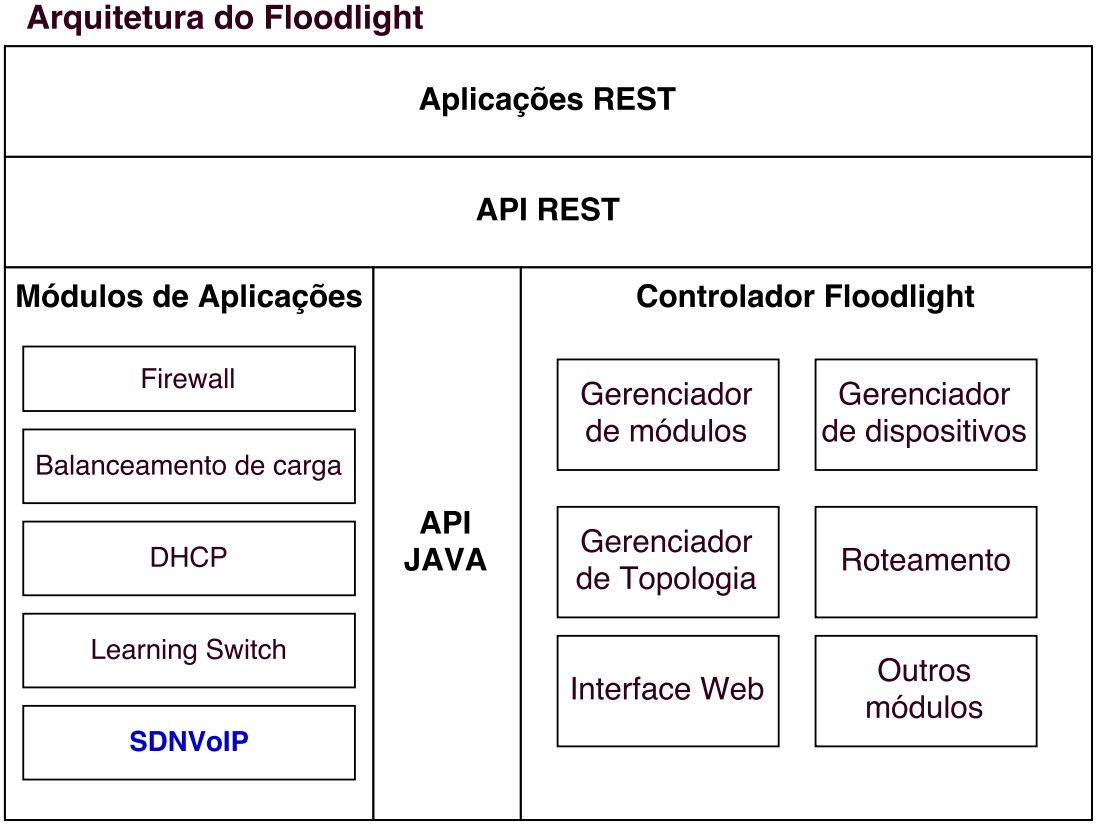
Listagem 4.1 – Arquivo de configuração dos módulos do Floodlight.

```
1  net.floodlightcontroller.core.internal.FloodlightProvider
2  net.floodlightcontroller.storage.memory.MemoryStorageSource
3  net.floodlightcontroller.flowcache.FlowReconcileManager
4  net.floodlightcontroller.hub.Hub
5  net.floodlightcontroller.jython.JythonDebugInterface
6  net.floodlightcontroller.debugcounter.DebugCounterServiceImpl
7  net.floodlightcontroller.debugevent.DebugEventService
8  net.floodlightcontroller.threadpool.ThreadPool
9  net.floodlightcontroller.perfmon.PktInProcessingTime
10 net.floodlightcontroller.restserver.RestApiServer
11 net.floodlightcontroller.ui.web.StaticWebRoutable
12 net.floodlightcontroller.virtualnetwork.VirtualNetworkFilter
13 net.floodlightcontroller.devicemanager.internal.DefaultEntityClassifier
14 net.floodlightcontroller.core.internal.OFSwitchManager
15 net.floodlightcontroller.threadpool.ThreadPool
16 net.floodlightcontroller.core.internal.ShutdownServiceImpl
17 org.sdnplatform.sync.internal.SyncManager
18 org.sdnplatform.sync.internal.SyncTorture
19 net.floodlightcontroller.staticflowentry.StaticFlowEntryPusher
20 net.floodlightcontroller.testmodule.TestModule
21 net.floodlightcontroller.topology.TopologyManager
22 net.floodlightcontroller.loadbalancer.LoadBalancer
23 net.floodlightcontroller.linkdiscovery.internal.LinkDiscoveryManager
24 net.floodlightcontroller.devicemanager.internal.DeviceManagerImpl
25 net.floodlightcontroller.firewall.Firewall
26 net.floodlightcontroller.accesscontrollist.ACL
27 net.floodlightcontroller.dhcpserver.DHCPServer
28 net.floodlightcontroller.learningswitch.LearningSwitch
29 net.floodlightcontroller.forwarding.Forwarding
```

4.2 DESENVOLVIMENTO DO MÓDULO SDNVOIP

O módulo SDNVoIP foi desenvolvido como um módulo do controlador Floodlight, ou seja, dentro da arquitetura do Floodlight SDNVoIP é um módulo, já em uma rede SDN 1, o módulo é uma aplicação da camada superior da arquitetura. A Figura 15 apresenta a arquitetura do Floodlight com o módulo SDNVoIP.

Figura 15 – Arquitetura do controlador Floodlight incluindo o módulo SDNVoIP.



Fonte: Adaptado de Floodlight (ARCHITECTURE, 2017)

É importante notar que o funcionamento do módulo SDNVoIP para o Floodlight tem como objetivo reduzir o uso de CPU em um servidor VoIP fazendo com que chamadas que utilizem o mesmo Codec não trafeguem áudio pelo servidor (Requisito Não Funcional 3, Seção 3.2), além disso não há necessidade de reconfigurar ou instrumentar o(s) servidor(es) VoIP(s), nem os clientes e sem a necessidade de outros servidores na rede (Requisito Não Funcional 3, Seção 3.2).

Para fazer a instalação das regras no *switch* o módulo precisa das seguintes informações:

- endereço IP do servidor VoIP;

- endereço IP do Cliente Origem (CO);
- endereço IP do Cliente Destino (CD);
- endereço MAC do CO;
- endereço MAC do CD;
- Codec utilizado pelo CO;
- Codec utilizado pelo CD;
- Porta RTP do CO;
- Porta RTP do CD;
- Porta RTP do servidor VoIP (canal do CO); e
- Porta RTP do servidor VoIP (canal do CD).

O endereço IP do servidor VoIP deve estar em um arquivo de configuração conforme descrito no Manual do Módulo SDNVoIP (Anexo B) (JUNIOR et al., 2017). O restante das informações são obtidas pelo módulo SDNVoIP a partir da análise dos pacotes SIP trocados durante o estabelecimento da chamada (troca de mensagens SIP descrita no Subseção 2.2.1.1) (Requisito Funcional 1, Seção 3.1), e ilustrada pela Figura 5. Os endereços MAC são necessários para que, caso os clientes possuam o mesmo endereço IP (por encontrarem-se sob regime NAT), suas chamadas possam ser diferenciadas e realizadas.

Assim, o SDNVoIP obtém o restante das informações da seguinte maneira:

1. A primeira mensagem SIP enviada pelos clientes, para o estabelecimento de uma chamada VoIP, é uma mensagem SIP REGISTER, ou seja, é a autenticação dos clientes no servidor VoIP. Com essa mensagem o módulo SDNVoIP obtém os endereços IPs e MACs dos clientes e armazena estas informações em memória;
2. Uma chamada VoIP é iniciada com a mensagem SIP INVITE. O CO envia um SIP INVITE para o servidor VoIP, que é interceptada pelo módulo SDNVoIP que analisa essa mensagem e armazena o Codec e a porta RTP que CO irá utilizar;
3. O servidor VoIP responde para o CO com uma mensagem SIP 200 OK, que também é interceptada pelo módulo SDNVoIP. Com essa mensagem o módulo obtém a porta RTP que o servidor reservou para o CO;

4. O servidor envia uma mensagem SIP INVITE para CD, que é interceptada pelo módulo SDNVoIP. Com essa mensagem o módulo obtém a porta RTP que o servidor reservou para o CD; e

5. CD envia uma mensagem SIP 200 OK para o servidor VoIP. Com essa mensagem o módulo obtém o Codec e a porta RTP utilizada por CD.

O Algoritmo 1 apresenta um resumo da lógica para o funcionamento do módulo SDNVoIP. O algoritmo inicia sempre que um pacote seja recebido pelo controlador Floodlight por não ter encontrado correspondência na tabela de fluxos do *switch* OpenFlow. O módulo SDNVoIP é então executado recebendo como entrada a mensagem *PACKET_IN* (linha 1). Uma série de verificações são feitas em seguida para descobrir se o pacote é IPv4, em seguida se é UDP e se é uma mensagem SIP. Como existem vários tipos de mensagem SIP, o módulo trata apenas as mensagens SIP para o estabelecimento e término de sessão, autenticação e estabelecimento e término de chamadas. À medida que as mensagens SIP são trafegadas conforme Figura 5, o módulo faz as verificações necessárias e vai armazenando as informações dos CO, CD e portas do servidor VoIP (linhas 10, 12, 16 e 18).

Algoritmo 1 Processamento de um pacote no módulo SDNVoIP.

Input: packetIn

```
 1: pkt ← packetIn;
 2: if (isIPv4(pkt))
 3:     if (isUDP(pkt))
 4:         if (isSIPType(pkt))
 5:             sip ← pkt;
 6:             if (isREQUEST(sip))
 7:                 if (isINVITE(sip))
 8:                     if (isSDP(sip))
 9:                         if (isFromCOToVoipServer(sip))
10:                             storeInfoCOToServer(sip);
11:                         else if (isFromVoipServerToCD(sip))
12:                             storeInfoServerToCD(sip);
13:                 else if (isRESPONSE(sip))
14:                     if (isOK200MsgToINVITE(sip))
15:                         if (isFromVoipServerToCO(sip))
16:                             storeInfoServerToCO(sip);
17:                         else if (isFromCDToVoipServer(sip))
18:                             storeInfoCDToServer(sip);
19:                             if (isClientsSameCodec())
20:                                 installFlows()
```

Após obter as informações já descritas, e caso os clientes estejam utilizando o mesmo Codec (Requisito Funcional 3, Seção 3.1), o módulo SDNVoIP faz a insta-

lação de quatro fluxos no *switch* (linha 20) (Requisito Funcional 4, Seção 3.1). Cada fluxo, corresponde a instalação de regras de *matching* e ações, nas tabelas de fluxo do *switch* SDN, que são executadas sobre cada pacote correspondente. A porta RTCP é calculada pelo módulo, pois conforme descrito na Subseção 2.2.1.2, a porta RTCP possui numeração subsequente à porta RTP. A instalação desses fluxos é feita de forma reativa, pois não há como saber previamente quais portas RTP serão utilizadas nas chamadas. Os fluxos possuem sete ações que devem ser aplicadas no pacote correspondente ao fluxo instalado, para que o redirecionamento de áudio ocorra diretamente entre os clientes (Requisito Não Funcional 5, Seção 3.2). Estas ações alteram os pacotes fazendo com que os clientes recebam os pacotes como se tivessem sido enviados pelo servidor VoIP:

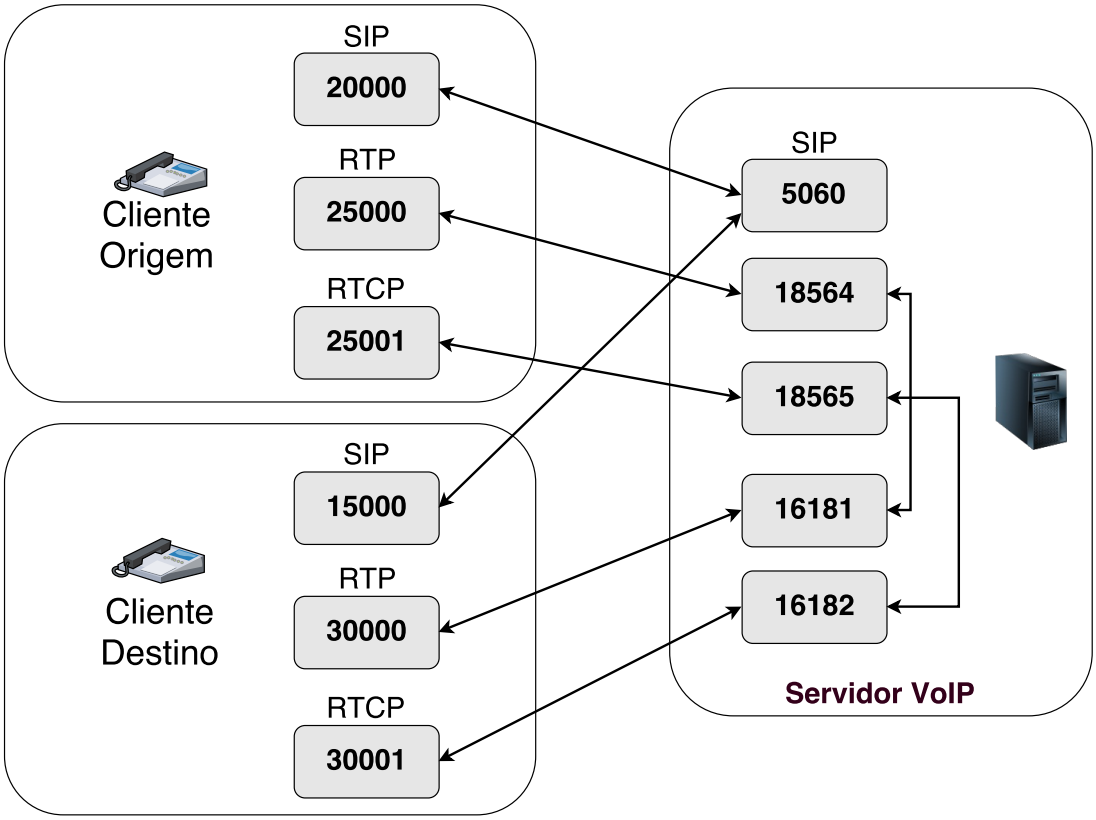
- Alterar o endereço *Media Access Control* (MAC) de origem para o endereço MAC do servidor VoIP;
- Alterar o endereço MAC de destino para o endereço MAC do cliente destino;
- Alterar o endereço IP de origem para o endereço IP do servidor VoIP;
- Alterar o endereço IP de destino para o endereço IP do cliente destino;
- Alterar a porta RTP/RTCP de origem para a porta RTP/RTCP do servidor VoIP;
- Alterar a porta RTP/RTCP de destino para a porta RTP/RTCP do cliente destino; e
- Saída do pacote em uma porta física do *switch*.

Os fluxos são removidos do *switch* quando a chamada termina e uma mensagem SIP/BYE é enviada (Requisito Funcional 5, Seção 3.1). O módulo detecta esta mensagem e remove os fluxos referentes aos clientes envolvidos. Outro detalhe importante é que os fluxos são instalados no *switch* com um tempo de expiração (*timeout*) para o caso em que uma chamada termine abruptamente e a mensagem SIP/BYE não seja enviada. Com isso o módulo SDNVoIP não consegue detectar o término da chamada para remover os fluxos, sendo estes removidos do *switch* após o tempo de expiração.

Um exemplo de mapeamento de portas em uma chamada normal (sem a utilização de SDN) é mostrado na Figura 16. Os clientes origem e destino comunicam-se na porta SIP 20000 e 15000, respectivamente, com a porta SIP 5060 do servidor VoIP. Nesse caso, todo o áudio trafegado entre os clientes passa pelo servidor VoIP. O servidor recebe na porta RTP 18564 o áudio enviado pelo cliente origem na porta 25000 e encaminha o áudio na porta 16181 para o cliente destino na porta RTP 30000. Como

as portas RTP e RTCP são definidas em pares, os pacotes RTCP do cliente origem na porta 25001 são recebidos pelo servidor na porta 18565 e encaminhados do servidor na porta 16182 para o Cliente 2 na porta RTCP 30001. Essas portas são apenas exemplos e mudam de acordo com a configuração dos clientes e do servidor VoIP. A única porta que não mudará é a porta SIP do servidor VoIP, que por padrão é a porta 5060.

Figura 16 – Mapeamento de portas em uma chamada normal.



Fonte: Elaborado pelo autor

O módulo SDNVoIP trabalha nos dois modelos de comunicação com o *switch*, *out-of-band* e/ou *in-band*, ou seja, está preparado para trabalhar independentemente do modo de comunicação entre o *switch* OpenFlow e o controlador Floodlight. O SDNVoIP, como já citado, foi desenvolvido como um módulo do Floodlight, portanto fica evidente a dependência do controlador Floodlight. Além disso, foi necessário alterar o código do controlador Floodlight para que os pacotes dos protocolos SIP, RTP e RTCP fossem manipulados pelo módulo (Requisito Funcional 1, Seção 3.1). Caso uma versão mais atual do Floodlight seja utilizada é necessário fazer a portabilidade do código deste projeto, além de realizar as alterações necessárias para que não ocorra incompatibilidade com a versão atualizada.

Outro detalhe importante é que o módulo SDNVoIP foi desenvolvido com o módulo padrão de encaminhamento do Floodlight desabilitado. Caso seja necessário utilizar o módulo de encaminhamento padrão do Floodlight, são necessárias alterações no código do módulo SDNVoIP e no código do Floodlight. A ordem de execução dos módulos deve ser alterada mantendo a execução do módulo SDNVoIP anterior à execução do módulo de encaminhamento. Outros módulos do Floodlight podem ser executados antes ou depois do módulo SDNVoIP. O único módulo que irá influenciar no redirecionamento de pacotes é o módulo de encaminhamento padrão do Floodlight. Caso o módulo padrão do Floodlight execute antes do SDNVoIP, todo o tráfego de áudio será feito passando pelo servidor VoIP.

4.3 CONSIDERAÇÕES PARCIAIS

Este capítulo apresentou o desenvolvimento de um módulo denominado SDNVoIP. O desenvolvimento desse módulo tem como intuito reduzir a utilização de CPU em um servidor VoIP, a largura de banda utilizada e a quantidade de dados trafegados em uma seção da rede. O SDNVoIP foi desenvolvido como um módulo do controlador Floodlight e trata pacotes SIP, SIP/SDP, RTP e RTCP, analizando-os e extraindo informações de portas e Codecs utilizados pelos clientes. Caso os clientes efetuem chamadas VoIP e utilizem o mesmo Codec, o SDNVoIP identificará as portas envolvidas, bem como os Codecs, e fará a instalação de fluxos nos *switches* no caminho do áudio, fazendo com que o tráfego de áudio seja feito diretamente entre os eles. O capítulo é finalizado com a apresentação das limitações do presente trabalho.

5 ANÁLISE EXPERIMENTAL

Para avaliar a aplicabilidade do SDNVoIP, foram desenvolvidos dois cenários experimentais conforme descrito no plano de testes (Subseção 3.2.2). Na Seção 5.1 são apresentados os softwares e ferramentas utilizados.

Os experimentos foram realizados sem a utilização de SDN e com a utilização de SDN com o módulo SDNVoIP ativo, além disso, os experimentos quantificaram a utilização de CPU do servidor VoIP, largura de banda no servidor VoIP e utilização de CPU no equipamento de rede. Os resultados obtidos são apresentados, bem como o ganho real com a utilização da solução proposta. Por fim, é apresentado uma análise da satisfação do presente trabalho com relação aos requisitos funcionais e não funcionais.

5.1 SOFTWARES E FERRAMENTAS UTILIZADAS

Foi utilizado o SDNVoIP como um módulo do controlador Floodlight, versão 0.9, como já mencionado na Seção 4.1. Como servidor VoIP foi utilizado o Asterisk versão 1.13, executado em um computador com processador Intel QuadCore de 3.2GHz com 8GB de memória RAM.

O *softphone* Linphone versão 3.6.1 foi utilizado para gerar as chamadas de voz. O Linphone permite a realização de chamadas de voz e video, gerenciamento de contatos, histórico de chamadas e pode ser executado em linha de comando entre outras funcionalidades (LINPHONE, 2001). A configuração do Linphone é feita através de um arquivo texto que contém informações como nome de usuário, senha do usuário, Codecs disponíveis e IP do servidor VoIP, entre outras. A inicialização do Linphone por linha de comando permite que o arquivo de configuração seja informado como parâmetro além de permitir a automatização de tarefas, como realizar e receber chamadas, por exemplo.

Foram desenvolvidos quatro *scripts* para auxiliar a execução dos testes e carga do servidor VoIP. O primeiro *script* tem como objetivo inicializar várias instâncias do Linphone. Cada instância possui um arquivo de configuração diferente, ou seja, em cada instância há um usuário (cliente) registrado no servidor VoIP. O segundo *script* inicializa as chamadas de voz entre os usuários em cada instância do Linphone e faz reprodução de um arquivo de áudio fazendo com que haja tráfego de voz em ambos os sentidos da chamada. Dessa maneira a carga gerada no servidor VoIP é feita com chamadas entre dois clientes, trafegando áudio e simulando uma conversa em que os dois participantes falam e ouvem. O terceiro *script* serve para finalizar as chamadas

de voz entre os clientes e o quarto *script* para finalizar as instâncias do *softphone*.

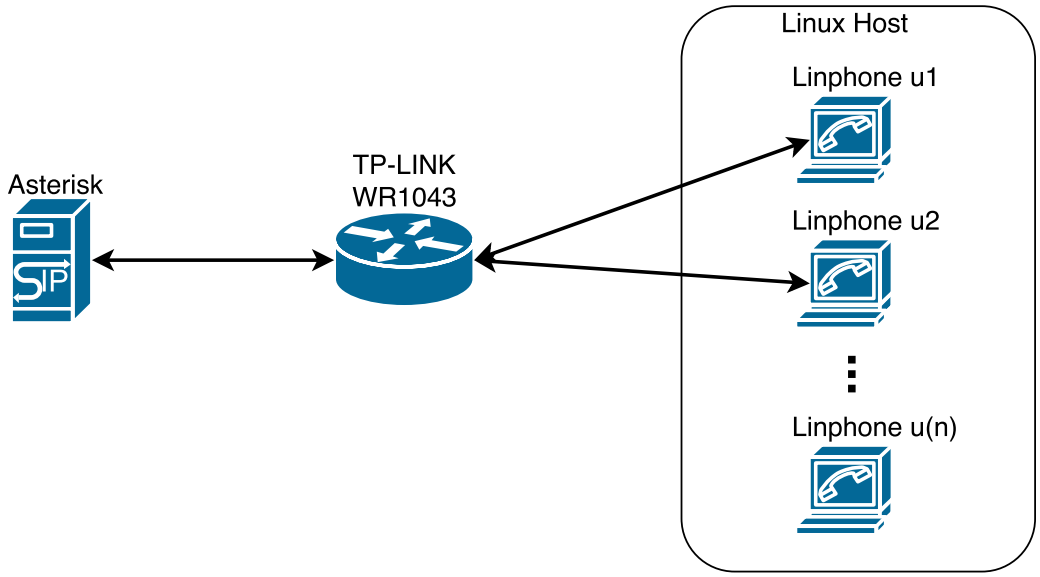
O *switch* utilizado foi o TP-Link modelo WR1043, que de acordo com o fabricante (TECHNOLOGIES, 2017) possui processador de 720MHz, 8MB de memória *flash*, 64MB de RAM, wifi b/g/n e 5 portas Gigabit entre outras características. O *firmware* original foi substituído pelo *Open Wireless Router* (OpenWRT) versão 15.01. O OVS 2.3 foi instalado junto ao OpenWRT para desempenhar o papel de *switch* virtual. A versão utilizada do OpenFlow foi a 1.3, pois o controlador Floodlight em sua versão 0.9 suporta versões do OpenFlow até a versão 1.3.

5.2 CENÁRIOS EXPERIMENTAIS

Dois cenários foram utilizados para realização dos experimentos de avaliação do módulo SDNVoIP. Um cenário sem a utilização do paradigma SDN e outro com SDN, ambos em ambiente LAN.

O cenário sem SDN apresentado na Figura 17 possui o servidor VoIP, o *switch* TP-Link WR1043 (com *firmware* OpenWRT 15.01 sem utilização de SDN/SDNVoIP) e um computador com sistema operacional Ubuntu versão 16.04 LTS no qual foram inicializadas as instâncias do *softphone* Linphone.

Figura 17 – Topologia sem SDN.



Fonte: Elaborado pelo autor

O *script* de inicialização do *softphone* recebe como parâmetro a quantidade de instâncias que devem ser executadas no *Linux Host*. Cada instância do *softphone* possui um cliente com nome de usuário **u** seguido por um número sequencial. Assim

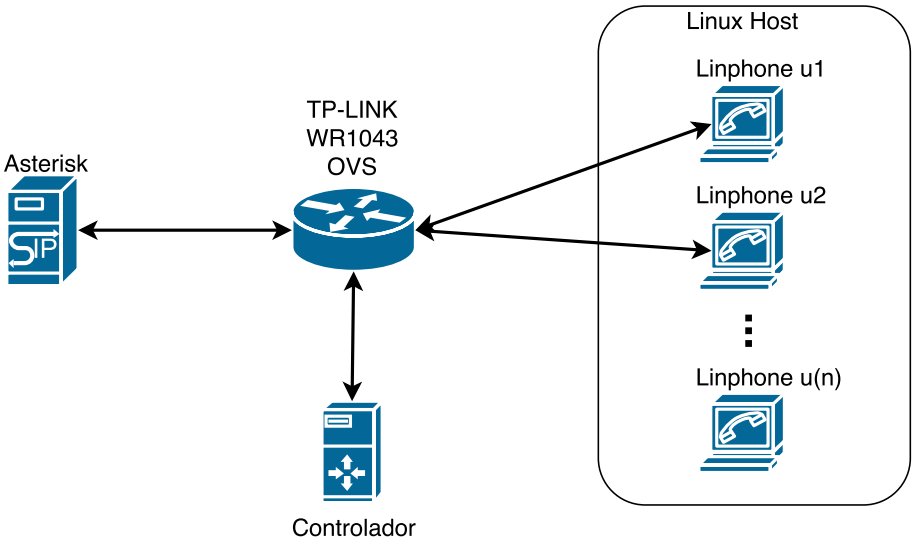
os clientes possuem como nome de usuário u1, u2 e assim por diante até o limite informado como parâmetro.

As chamadas são geradas por um *script* e a maneira como são geradas segue uma lógica determinística: Por exemplo, para o caso de 50 chamadas entre um grupo de 100 clientes tem-se que, cliente u51 faz uma chamada para o cliente u1, cliente u52 faz uma chamada para o cliente u2 e assim por diante até o cliente final, u100 chamar o cliente u50.

O *softphone* foi configurado para atender as chamadas recebidas automaticamente e logo após o atendimento realizar a reprodução de um arquivo de áudio em formato *Waveform Audio Format* (WAV), taxa de amostragem de 44,100Hz, 16bits e aproximadamente 20 minutos de duração. Todas as chamadas utilizaram o mesmo Codec (G711, alaw/ulaw) e todo o áudio é trafegado pelo servidor VoIP, no cenário sem a utilização de SDN. O Codec G711 é utilizado em uma grande quantidade de aplicações, além de ter sido utilizado em outros cenários experimentais (ALI; VASSILARAS; NTAGKOUNAKIS, 2009), (HASROUTY et al., 2016), (GOODE, 2002), (MONTORO; CASILARI, 2009).

O cenário com SDN (Figura 18) é similar ao cenário apresentado na Figura 17. As únicas diferenças são a adição do controlador Floodlight com o módulo SDNVoIP ativo e o *switch* TP-Link WR1043 com OVS habilitada e comunicando-se *out-of-band* com o controlador Floodlight usando o protocolo OpenFlow.

Figura 18 – Topologia com SDN.



Fonte: Elaborado pelo autor

5.3 RESULTADOS

As métricas utilizadas para avaliação dos experimentos foram: a taxa de utilização de CPU do servidor VoIP, largura de banda utilizada, quantidade de dados trafegados e taxa de utilização de CPU do *switch*.

Para coletar a taxa de utilização de CPU na máquina do servidor VoIP foi utilizado o comando *mpstat* do sistema operacional Linux, executado após o estabelecimento das chamadas em cada amostra. O intervalo de coleta foi de 1 segundo durante o período de cinco minutos para cada amostra. Os dados foram tratados da seguinte maneira: para cada amostra foi calculada a média aritmética, em seguida foi calculada a variância e o desvio padrão. Um total de dez amostras foram coletadas para cada experimento realizado.

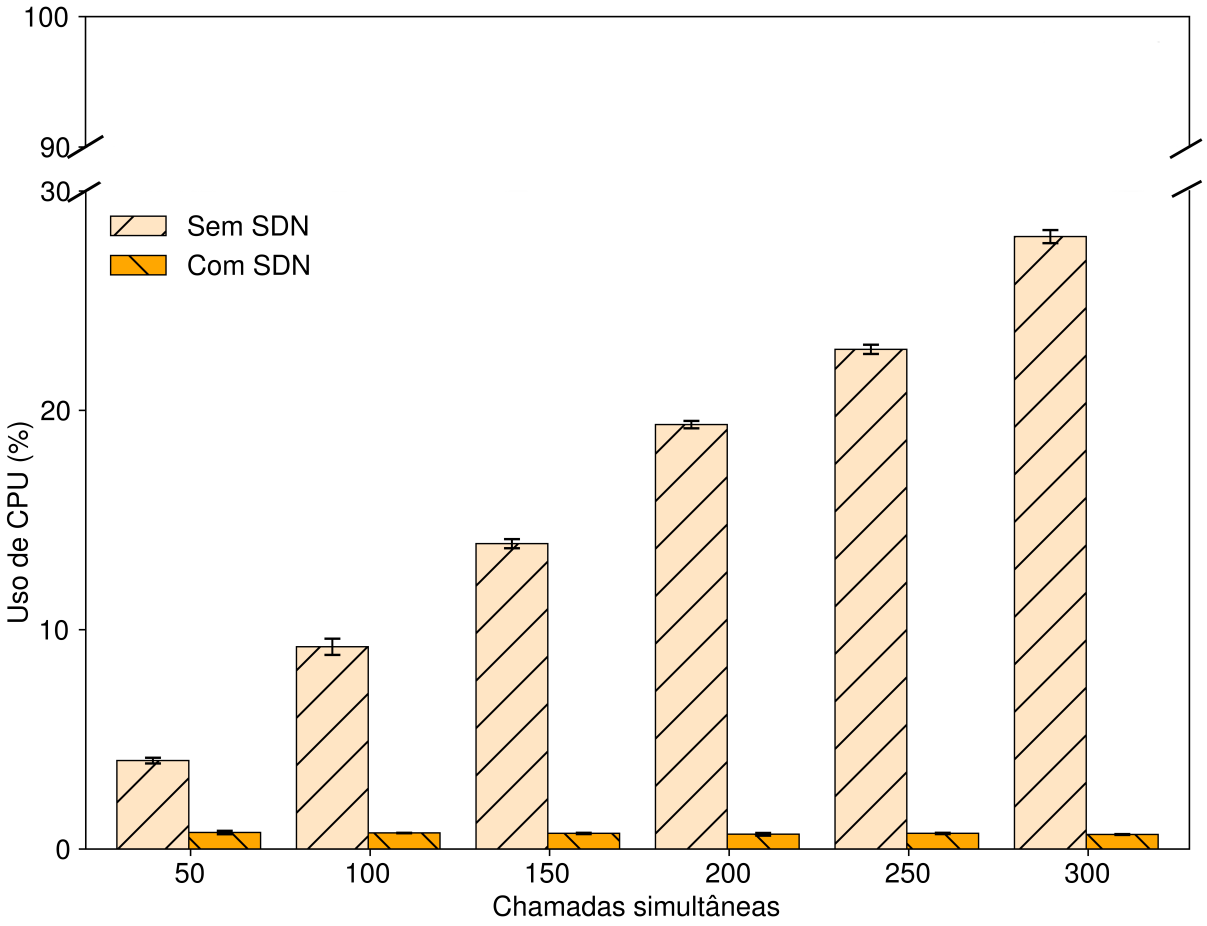
A Tabela 4 apresenta os dados de utilização de CPU em um servidor VoIP com as respectivas cargas nos cenários sem a utilização de SDN e com a utilização de SDN com o módulo SDNVoIP. Observa-se que 300 chamadas simultâneas utilizando o mesmo Codec e todo o tráfego passando pelo servidor VoIP, gera uma taxa de utilização de CPU do servidor Asterisk de aproximadamente 27%. Já com o uso de SDN, todo o tráfego gerado pelas chamadas é encaminhado diretamente entre os clientes, fazendo com que a CPU do servidor VoIP fique aproximadamente 99% do tempo ociosa.

Tabela 4 – Comparação do uso de CPU.

Sem SDN						
Chamadas simultâneas	**50**	**100**	**150**	**200**	**250**	**300**
Uso de CPU%	4,03	9,22	13,92	19,35	22,78	27,92
Variância	0,53	5,59	1,33	1,04	1,37	2,74
Desv. Padrão	0,13	0,37	0,21	0,17	0,21	0,30
Com SDN						
Uso de CPU%	0,75	0,73	0,71	0,67	0,71	0,66
Variância	0,01	0,00	0,00	0,00	0,00	0,00
Desv. Padrão	0,08	0,01	0,03	0,06	0,03	0,02

A Figura 19 apresenta graficamente a taxa de utilização de CPU. O eixo x apresenta a quantidade de chamadas simultâneas, enquanto que o eixo y apresenta a utilização de CPU em %. As barras verticais apresentam: em laranja claro (padrão /) a taxa de utilização de CPU sem o uso de SDN, enquanto que as barras na cor laranja (padrão \) apresentam a utilização de CPU com o uso de SDN. A solução com o uso de SDN reduz a utilização de CPU, dessa maneira a CPU permanece mais tempo ociosa e disponível para outras tarefas como por exemplo, a transcodificação entre clientes que efetuam chamadas com Codecs diferentes.

Figura 19 – Comparação do uso de CPU do servidor VoIP.



Fonte: Elaborado pelo autor

A segunda métrica utilizada nos experimentos foi a utilização de largura de banda no servidor VoIP. O comando utilizado foi o *bmon*, que é uma ferramenta do sistema operacional Linux e serve para realizar o monitoramento de utilização da largura de banda em tempo real. A ferramenta foi executada no servidor VoIP. As leituras foram executadas com intervalos de 1 segundo durante o período da carga do servidor VoIP, estipulada em cinco minutos.

Somente a vazão da interface física pertencente ao servidor VoIP que recebeu todo o tráfego das chamadas foi monitorada. A Tabela 5 apresenta as métricas de desempenho coletadas sendo rx a vazão de recepção, $rxpps$ a taxa de recepção de pacotes, em pacotes por segundo, tx a vazão de transmissão e $txpps$ a taxa de transmissão de pacotes, em pacotes por segundo.

Tabela 5 – Vazão em um servidor VoIP sem e com a utilização de SDN.

Sem SDN						
	50	**100**	**150**	**200**	**250**	**300**
rx	8,65Mbps	17,20Mbps	25,76Mbps	34,40Mbps	43,88Mbps	50,76Mbps
rxpps	5,10Kbps	10,13Kbps	15,19Kbps	20,23Kbps	25,84Kbps	28,44Kbps
tx	8,65Mbps	17,36Mbps	26Mbps	34,72Mbps	44,30Mbps	51,25Mbps
txpps	5,04Kbps	10,08Kbps	15,12Kbps	20,16Kbps	25,73Kbps	28,32Kbps
Com SDN						
rx	0,065Kbps	0,065Kbps	0,065Kbps	0,065Kbps	0,065Kbps	0,073Kbps
rxpps	0,005Kbps	0,01Kbps	0,01Kbps	0,005Kbps	0,008Kbps	0,01Kbps
tx	2,62Kbps	5,73Kbps	15,15Kbps	24,08Kbps	28,18Kbps	45,38Kbps
txpps	1,43Kbps	4,19Kbps	11,28Kbps	17,91Kbps	21,08Kbps	34,42Kbps

De acordo com a Tabela 5, observa-se que para 50 chamadas simultâneas a vazão típica de recepção e transmissão é de 1,03Mbps. As taxas típicas de recepção e transmissão de pacotes foram de 5,09Kbps e 5,04Kbps respectivamente. A estimativa de dados recebidos e transmitidos durante a medição foram de 320,75MB e 322,54MB respectivamente. Já com o uso do módulo SDNVoIP, houve uma redução vazão do servidor para 0,065Kbps na recepção e 0,32Kbps na transmissão. O total de dados recebidos e transmitidos durante a medição com o módulo SDNVoIP, foi de 1,63KB e 59,46KB para recepção e transmissão respectivamente.

Já para 300 chamadas simultâneas observa-se que a vazão típica de recepção e transmissão é de 50,76Mbps e 51,25Mbps respectivamente. As taxas típicas de recepção e transmissão de pacotes foram de 28,44Kbps e 28,32Kbps respectivamente. O total de dados recebidos e transmitidos durante a medição foram de 1887,23MB e 1903,46MB respectivamente. Já com o uso da solução baseada em SDN há uma redução para 0,073Kbps e 45,38Kbps da vazão de recepção e transmissão no servidor VoIP. Outro benefício é a redução do tráfego entre o *switch* e o servidor que nesse experimento com o uso de SDN e o módulo SNDVoIP, resultou em um total de dados trafegados de 1,93KB para recepção e 1,36MB para transmissão com 300 chamadas simultâneas. Estes resultados indicam que o módulo SDNVoIP pode ser utilizado para evitar congestionamentos em parte da rede, principalmente no segmento de rede onde o servidor VoIP está localizado.

Outra preocupação é a carga no *switch*. Diminui-se a utilização de CPU no servidor VoIP, porém qual é o aumento de carga de CPU que a solução impõe no equipamento de rede? Para se obter qual a carga da solução SDN no *switch*, os cenários descritos na Seção 5.3 foram repetidos e dados de utilização de CPU do *switch* foram coletados. Foram executados os cenários com chamadas simultâneas sem a utilização de SDN e com a utilização de SDN, ou seja, cenários onde o *switch* não utiliza tabelas de fluxo e portanto o módulo SDNVoIP não é utilizado. Para tanto o *switch* executa OpenWRT com OVS. Esse experimento tem por objetivo o estabele-

cimento do consumo de CPU no *switch* em utilização tradicional de rede. Por outro lado, também foram executados os experimentos de medição do consumo de CPU do *switch* com o suporte a OpenFlow, ou seja, executando OpenWRT com o módulo OVS e OpenFlow. Nesse caso, o SDNVoIP está ativo e instala fluxos nas tabelas de fluxo, que são processados pelo *switch* durante a ocorrência das chamadas de voz. A diferença entre as taxas de utilização de CPU do *switch* no cenário OVS sem SDN e OVS com SDN foi considerado como sendo a carga (ou *overhead*) da solução apresentada neste trabalho.

De acordo com a Tabela 6, para a menor carga, 50 chamadas simultâneas, sem a utilização de SDN, a média de uso de CPU do *switch* foi de 19,72%, já com a utilização de SDN e com o módulo SDNVoIP ativo, a média de uso de CPU subiu para 19,93%. Para a maior carga, 300 chamadas simultâneas, sem SDN a média de utilização de CPU do *switch* foi de 83,82%, já com SDN/SDNVoIP a média de uso de CPU subiu para 93,05%.

Tabela 6 – Carga de CPU do *switch*.

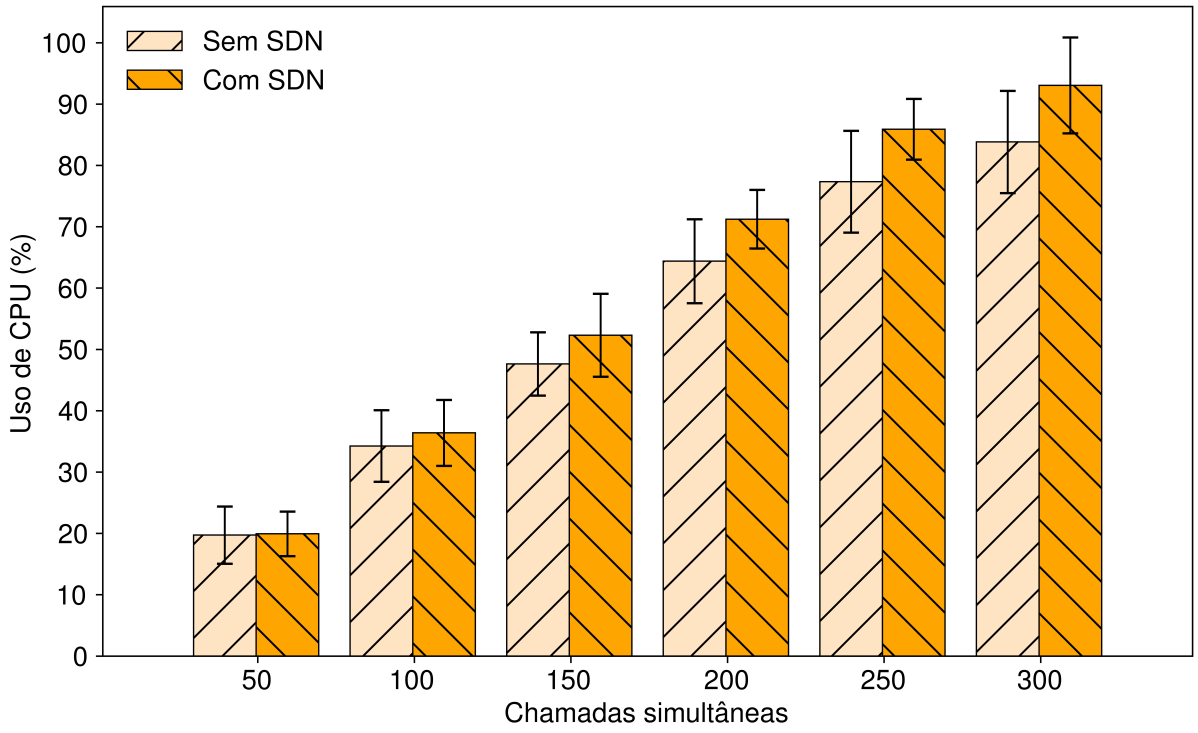
Sem SDN						
Chamadas	50	100	150	200	250	300
Uso de CPU%	19,72	34,25	47,63	64,38	77,35	83,82
Variância	21,73	33,93	26,58	46,90	68,90	69,56
Desv. Padrão	4,66	5,83	5,16	6,85	8,30	8,34
Com SDN						
Uso de CPU%	19,93	36,39	52,31	71,23	85,90	93,05
Variância	13,21	28,84	45,87	22,86	24,47	61,14
Desv. Padrão	3,63	5,37	6,77	4,78	4,95	7,82

A Figura 20 apresenta graficamente os valores de uso de CPU do *switch* com as respectivas cargas em cada cenário, sem a utilização de SDN e com a utilização de SDN/SDNVoIP.

A Tabela 7 apresenta a diferença entre o uso de CPU no *switch* nos cenários sem a utilização de SDN e com a utilização de SDN com o módulo SDNVoIP ativo. Para a menor carga, 50 chamadas simultâneas, o módulo SDNVoIP aumentou em 0,02% a taxa de utilização de CPU do *switch*. Para a maior carga, 300 chamadas simultâneas, houve um aumento de 9,23% de utilização de CPU do *switch* em relação ao cenário sem o uso de SDN. Resultado já esperado, pois, com a utilização de SDN/SDNVoIP é esperado que ocorra maior utilização de CPU no *switch*, carga atribuída ao fato de que para cada pacote enquadrado em um fluxo o *switch* deve aplicar 7 ações conforme descrito na Subseção 4.2.

De acordo com a Tabela 7, observa-se um crescimento aproximadamente linear na taxa de utilização de CPU do *switch*. Pode-se concluir que a solução com o uso de SDN reduz a utilização de recursos no servidor VoIP, embora acabe por

Figura 20 – Comparação do uso de CPU do *switch*



Fonte: Elaborado pelo autor

Tabela 7 – Carga do módulo SDNVoIP

Chamadas	Aumento de consumo de CPU%
50	0,21%
100	2,15%
150	4,68%
200	6,85%
250	8,55%
300	9,23%

transferir uma pequena parcela da utilização de CPU do servidor VoIP para os equipamentos de rede. Mesmo assim, a transferência compensa pois é muito menor do que a carga retirada. Entretanto, o TP-LINK WR1043 é um *switch* destinado para uso doméstico e para se obter uma conclusão mais precisa, é necessário realizar mais estudos utilizando um equipamento profissional.

Essa compensação pode ser observada através do comparativo entre a carga retirada do servidor VoIP e colocada no *switch*. Ou seja, se para 50 ligações há 4,03% de uso de CPU no servidor VoIP sem utilização de SDN e 0,75% com a utilização de SDN/SDNVoIP, há uma diferença de 3,28% que poderiam ser transferidos para o

switch. Contudo, para essa carga, o consumo de CPU do *switch* teve um aumento de 0,02% com SDN/SDNVoIP. Assim, sendo a diferença entre o que poderia ser transferido de 3,28% menos 0,02% de aumento no consumo de CPU do *switch*, o que resulta em 3,07%, é o ganho da solução. Para a maior carga, 300 chamadas simultâneas, há 27,92% de uso de CPU no servidor sem SDN e 0,66% com a utilização de SDN/SDN-VoIP, havendo portanto, uma diferença de 27,26% que poderiam ser transferidos para o *switch*. Entretanto, o consumo de CPU do *switch* teve um aumento de 9,23% com SDN/SDNVoIP. Assim, a diferença entre o que poderia ser transferido que é 27,26% e o que de fato foi transferido que é 9,23%, resulta em 18,07%, que é o ganho da solução. Ou seja, o ganho real da solução é o quanto de fato foi economizado em termos de utilização total CPU (servidor VoIP e equipamento de rede) com a utilização do SDNVoIP.

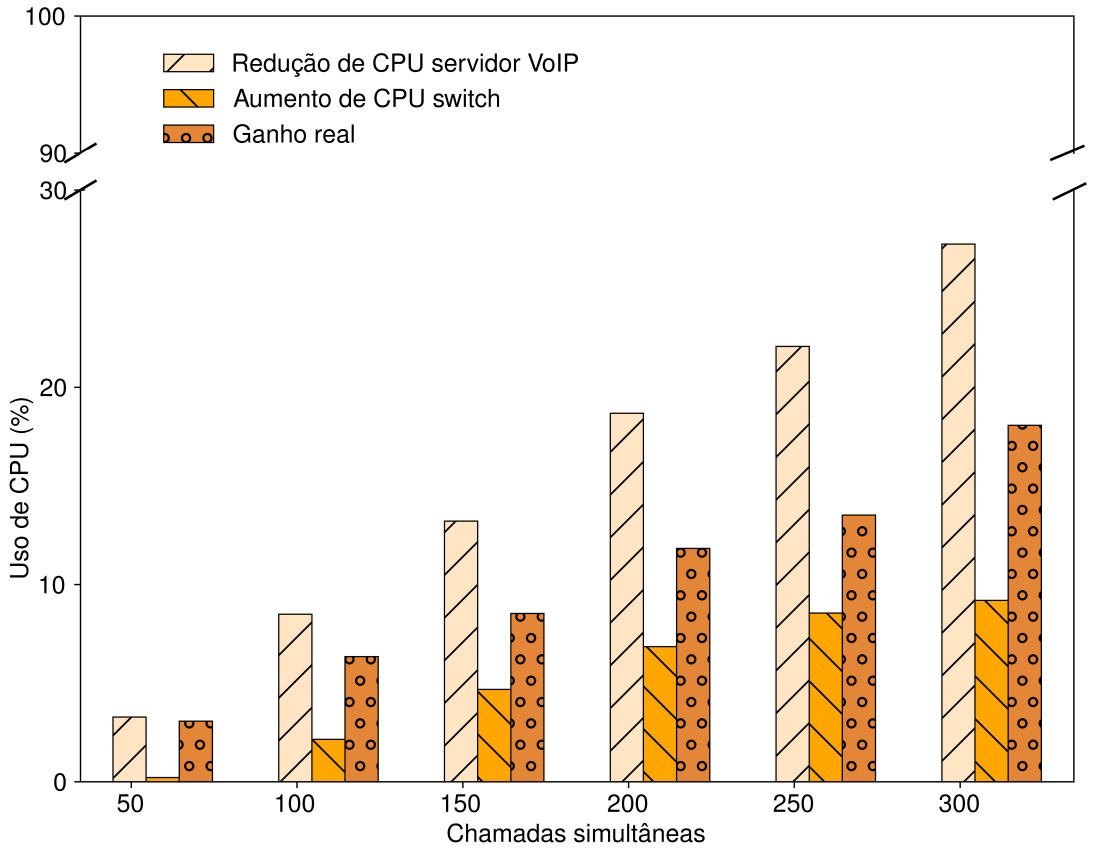
A Tabela 8 apresenta os valores do comparativo entre a carga retirada do servidor VoIP e colocada no *switch*, na qual a primeira linha representa a quantidade de chamadas simultâneas, a segunda linha apresenta o uso de CPU do servidor VoIP sem a utilização de SDN/SDNVoIP, a terceira linha apresenta o uso de CPU do servidor VoIP com a utilização de SDN/SDNVoIP. A quarta linha apresenta o ganho da solução no servidor VoIP, ou seja a diferença entre os cenário sem e com a utilização de SDN/SDNVoIP, em outras palavras, o consumo de CPU que poderia ser transferido para o *switch*. A quinta linha apresenta o consumo de CPU do *switch* já contabilizada a diferença entre os cenários sem e com a utilização de SDN/SDNVoIP. Por fim, a última linha apresenta o ganho final da solução, ou seja, a diferença entre o consumo de CPU que poderia ser transferido do servidor VoIP par o *switch* e o que de fato foi transferido.

Tabela 8 – Ganho da solução em termos de consumo de CPU.

	Chamadas simultâneas	**50**	**100**	**150**	**200**	**250**	**300**
Servidor VoIP	**Sem SDN - CPU%**	4,03	9,22	13,92	19,35	22,78	27,92
	Com SDN - CPU%	0,75	0,73	0,71	0,67	0,71	0,66
	Redução de CPU	3,28	8,49	13,21	18,68	22,07	27,26
Switch	**Aumento de CPU%**	0,21	2,15	4,68	6,85	8,55	9,19
	Ganho da solução	**3,07**	**6,34**	**8,53**	**11,83**	**13,52**	**18,07**

A Figura 21 apresenta o ganho real da solução em consumo de CPU. No eixo x, a primeira do conjunto de 3 colunas apresenta o valor do ganho na redução de utilização de CPU do servidor VoIP, ou seja, a diferença entre o cenário sem utilização de SDN e com SDN/SDNVoIP. A segunda coluna apresenta o valor de consumo extra de CPU no *switch*, ou seja, a diferença entre o cenário com e sem a utilização de SDN/SDNVoIP. A terceira coluna é o valor do ganho real da solução, ou seja, o valor da diferença entre o ganho no servidor VoIP e o consumo extra no *switch*. O eixo x apresenta quantidade de chamadas simultâneas e o eixo y a utilização de CPU em %.

Figura 21 – Ganho real da solução SDNVoIP.



Fonte: Elaborado pelo autor

Além dos cenários de avaliação elencados, em uma rede LAN tradicional, um cenário como o de empresas com matriz e filiais, que configura-se como um ambiente LAN em uma infraestrutura WAN, pode obter benefícios da solução apresentada no presente trabalho. Um relato de experiência, com o intuito de estudar a viabilidade de uso da solução proposta em um cenário semelhante ao de uma empresa com matriz e filial, simulando um ambiente físico WAN, com a utilização de *Network Address Translation* (NAT) e *firewall* é apresentado no Apêndice A.

5.3.1 Requisitos *versus* Resultados

Uma vez que os cenários experimentais foram executados, torna-se necessário avaliar se os requisitos foram satisfeitos. As Tabelas 9 e 10 listam os requisitos funcionais e não funcionais (Subseções 3.1 e 3.2) e informam se o mesmo é ou não satisfeito pelo módulo SDNVoIP.

Tabela 9 – Satisfação dos requisitos funcionais.

Requisito	**Satisfeito**	**Observação**
Analisar e identificar tráfego SIP/RTP e RTCP a fim de obter informações sobre os clientes	Sim	Foi implementado no módulo um *parser* para identificar os pacotes SIP, RTP e RTCP.
Identificar chamadas VoIP entre os clientes	Sim	O módulo é capaz de analisar a troca de mensagens SIP e identificar as informações que estabelecem uma chamada VoIP.
Identificar chamadas que utilizem o mesmo Codec	Sim	O módulo identifica o Codec utilizado pelos clientes durante a troca de mensagens SIP no estabelecimento da chamada.
Instalar fluxos nos *switches* OpenFlow para redirecionar o tráfego de áudio diretamente entre os clientes que utilizem o mesmo Codec	Sim	Após identificar que os clientes utilizam o mesmo Codec, o módulo instala 4 fluxos no *switch* fazendo com que o tráfego de áudio ocorra diretamente entre os clientes.
Remover os fluxos após o término das chamadas	Sim	O módulo remove os fluxos quando uma mensagem SIP/BYE é enviada por um dos clientes. Os fluxos são instalados no *switch* com um tempo de expiração para o caso em que uma chamada termine abruptamente, ou seja, sem o envio da mensagem SIP/BYE.

Tabela 10 – Satisfação dos requisitos não funcionais.

Requisito	**Satisfeito**	**Observação**
Utilizar SDN na solução	Sim	O SDNVoIP foi desenvolvido como um módulo do controlador SDN Floodlight.
Não prejudicar a comunicação entre os clientes e o servidor VoIP	Sim	O módulo consegue analisar os pacotes SIP, RTP e RTCP sem interromper ou prejudicar a comunicação entre os clientes e o servidor VoIP.
Reduzir a utilização de CPU no servidor VoIP	Sim	Com a utilização do módulo SDNVoIP houve uma redução de aproximadamente 27% de utilização de CPU no maior cenário experimentado (Tabela 4).
Reduzir a largura de banda no servidor VoIP	Sim	Com a utilização do módulo SDNVoIP, todo o tráfego de áudio é feito diretamente entre os clientes (Tabela 5).
Permitir tráfego direto entre os clientes	Sim	O SDNVoIP obtém informações sobre os clientes e instala fluxos no *switch* OpenFlow fazendo com que o tráfego ocorra direto entre os clientes.

De acordo com os resultados obtidos da execução dos cenários experimentais, é possível afirmar que os requisitos foram satisfeitos, e que portanto a análise de requisitos *versus* resultados é satisfatória.

5.4 CONSIDERAÇÕES PARCIAIS

Este capítulo apresentou a análise experimental realizada para avaliar a redução de utilização de CPU e largura de banda em um servidor VoIP, com a utilização da solução proposta baseada em SDN. O capítulo detalha os *softwares* e ferramentas utilizadas para realização dos testes experimentais, os cenários experimentados, a métrica utilizada, bem como os resultados obtidos. Os cenários e experimentos utilizaram o controlador Floodlight versão 0.9, servidor VoIP Asterisk versão 1.13, *softphone* Linphone, *switch* TP-Link modelo WR1043 versão 3, OpenWRT versão 15.01 e OVS versão 2.3. Os experimentos foram realizados em dois cenários: sem a utilização de SDN e utilizando SDN com o módulo SDNVoIP proposto e implementado, ativo. Foram coletados os dados, taxa de utilização de CPU do servidor VoIP, a largura de banda no servidor VoIP, bem como a taxa de utilização de CPU do *switch*. Os dados foram apresentados e comparados e com a utilização do módulo SDNVoIP foi obtida uma redução de aproximadamente 28% de utilização de CPU no maior cenário experimentado, com 300 chamadas simultâneas. O consumo mensurado de largura de banda foi de 50,76Mbps para 300 chamadas simultâneas sem SDN. No ambiente SDN, com a utilização da proposta SDNVoIP, a largura de banda no servidor VoIP foi reduzida para aproximadamente 0,073Kbps, com as mesmas 300 chamadas simultâneas. Ainda, dados de utilização de CPU do *switch* também foram coletados. Para o maior cenário, 300 chamadas simultâneas, houve um aumento de aproximadamente 9,23% na utilização de CPU do *switch* com a utilização de SDN, quando comparado com o mesmo cenário sem a utilização do SDNVoIP.

Com relação ao ganho real da solução, ou seja o valor do ganho na redução de utilização de CPU do servidor VoIP, foi considerado a diferença entre o cenário sem utilização de SDN e com a utilização de SDN/SDNVoIP. Para 50 chamadas simultâneas foi obtido um ganho de 3,07%, já para a maior carga, 300 chamadas simultâneas, há um ganho 18,07%. Ou seja, o ganho real da solução é o quanto de fato foi economizado em termos de utilização total CPU (servidor VoIP e equipamento de rede) com a utilização do SDNVoIP. Por fim, o Capítulo apresenta uma análise dos requisitos funcionais e não funcionais da solução *versus* resultados obtidos, em uma tabela, comprovando que os requisitos foram todos satisfeitos.

6 CONSIDERAÇÕES FINAIS

Na área de telecomunicações houve mudanças significativas nos últimos anos. A tecnologia VoIP permitiu que pessoas se comuniquem utilizando a Internet como meio de transmissão. Para os usuários dessa tecnologia o principal ganho é a redução de custos, além da agregação de outros serviços como gravação de ligação, encaminhamento de chamadas, secretária virtual, entre outros recursos.

O surgimento do paradigma SDN tem permitido a implementação de novas funcionalidades, experimentação de novas ideias e novos protocolos na área de redes de computadores. Essas características inovadoras permitem também novas possibilidades para monitoramento de serviços e equipamentos de rede, disponibilizando um aspecto mais dinâmico para aplicações de rede. É possível criar aplicações que reagem conforme o estado da rede alterando rotas para evitar congestionamentos ou para otimizar a utilização dos recursos disponíveis.

Nesse sentido, o presente trabalho agregou benefícios do uso de SDN na área de telecomunicações, mais especificamente na utilização de SDN para reduzir a utilização de recursos em um servidor VoIP. Os resultados obtidos mostram que é possível reduzir a utilização de CPU em um servidor VoIP através da criação de fluxos que redirecionam o tráfego de voz diretamente entre as partes envolvidas e que utilizam o mesmo Codec. Nesse caso, há também a redução na utilização da largura de banda no servidor VoIP e consequentemente redução no tráfego de dados em uma seção da rede.

Durante o desenvolvimento deste trabalho, houve algumas dificuldades. As principais dificuldades foram relacionadas ao equipamentos necessários para realização dos experimentos. No primeiro ambiente experimentado foi utilizado um PC no qual foram executadas máquinas virtuais para cada papel da topologia (servidor VoIP, OVS, mininet e clientes) e o controlador sendo executado na máquina *host*, porém os valores de utilização de CPU do servidor VoIP foram influenciados pela execução das outras máquinas virtuais. Dessa maneira, um segundo ambiente foi utilizado com os seguintes equipamentos, um PC para o papel de servidor VoIP, um PC para o papel de clientes VoIP, um PC para o controlador, um PC para os clientes VoIP e um *switch* TP-LINK WR1043 versão 1. Neste segundo ambiente a limitação foi o *switch* que atingia sua capacidade máxima após a execução de 140 chamadas VoIP simultâneas. O *switch* foi então substituído por outro de maior capacidade sendo assim possível realizar os experimentos.

Os resultados obtidos comprovam o ganho da solução. Embora a taxa de uti-

lização da CPU do *switch* tenha aumentado, esse aumento é menor se comparado ao ganho obtido na redução de utilização de CPU do servidor VoIP. O ganho real da solução, ou seja o valor do ganho na redução de utilização de CPU do servidor VoIP, para 50 chamadas simultâneas foi obtido um ganho de 3,07%, já para a maior carga, 300 chamadas simultâneas, há um ganho 18,07%. É importante ressaltar que solução apresentada neste trabalho tem implicação prática relevante, pois gerou um artefato de *software* disponível para a comunidade em um repositório no Github (JUNIOR et al., 2017). Outro ponto importante é o caráter único deste trabalho como observado no Capítulo 3.3.

A redução da utilização de CPU permite que haja maior disponibilidade de processador para chamadas que precisem de tradução de Codec, assim como a redução na utilização de largura de banda pode permitir que mais chamadas possam ser adicionadas no servidor. Outro detalhe importante é a redução na quantidade de dados trafegados no segmento de rede onde o servidor VoIP está localizado, principalmente em um cenário de empresas com matrizes e filiais que possuem plano de utilização da Internet com franquias de consumo de dados. A preocupação com a franquia é importante para o provisionamento do serviço VoIP, além disso a redução de utilização de CPU do servidor VoIP e da utilização de largura de banda, permite que a infraestrutura de servidores VoIP possa ser redimensionada sem a necessidade de adquirir novos servidores ou com maior capacidade computacional para suportar mais clientes e chamadas.

6.1 TRABALHOS FUTUROS

Como trabalhos futuros, o módulo SDNVoIP pode ser modificado para redirecionar o tráfego de clientes que possuem Codecs diferentes e precisam ser transcodificadas, para um servidor VoIP com menor utilização de CPU. Outra possibilidade é redirecionar o tráfego para um servidor VoIP com melhor localização em um ambiente de nuvem. Por fim, outra possibilidade é redirecionar o tráfego de chamadas para o servidor VoIP com um caminho de rede com menos tráfego, evitando congestionamentos. Ainda como trabalhos futuros, alguns trabalhos relacionados podem ser combinados ou integrados com o módulo SDNVoIP.

A proposta de Maribondo e Fernandes (2016), o CAoS, poderia ser combinado com a proposta deste trabalho gerando uma solução que reduz a utilização de recursos no servidor VoIP enquanto preocupa-se com a qualidade das chamadas quando ocorrem congestionamentos na rede. Vale ressaltar que o SDNVoIP também pode ser utilizado para reduzir congestionamentos no segmento de rede do servidor VoIP, pois realiza o redirecionamento de tráfego direto entre os clientes.

A proposta de Thorpe et al. (2016), o iMOS, poderia ser facilmente integrado com o SDNVoIP, pois também foi desenvolvido como um módulo do controlador Floodlight, dessa maneira a solução se tornaria mais abrangente, reduzindo a utilização de recursos no servidor VoIP e servindo como ferramenta para monitorar as chamadas VoIP. Outro detalhe é que o SDNVoIP pode ser adaptado para detectar congestionamentos e enviar uma mensagem SIP/SDP REINVITE e renegociar o Codec utilizado nas chamadas para um Codec com menor *bit rate* sem a necessidade de um servidor *proxy* como a proposta de Ng, Hoh e Singh (2005) que não utiliza SDN.

O trabalho de Egilmez et al. (2012) apresenta uma abordagem QoS para entrega de conteúdo multimídia fim a fim coletando informações sobre o estado da rede para recalcular rotas e ajustar o QoS dinamicamente. O módulo SDNVoIP pode ser alterado para utilizar a abordagem proposta por Egilmez et al. (2012) e dessa maneira entregar uma solução que reduz a utilização de recursos no servidor VoIP e preocupa-se com a qualidade do serviço.

6.2 CONTRIBUIÇÕES E PUBLICAÇÕES

A principal contribuição deste trabalho é a redução do consumo de recursos em um servidor VoIP. Especificamente recursos como a utilização de CPU, largura de banda e a quantidade de dados trafegados no segmento de rede onde está localizado o servidor VoIP. Além disso o trabalho teve uma contribuição prática que foi o desenvolvimento de um módulo que permite o redirecionamento de chamadas que utilizem o mesmo Codec possibilitando assim a redução da utilização dos recursos citados.

Além disso, durante a realização deste trabalho, contribuições foram desenvolvidas e publicadas nos seguintes eventos:

- VIEIRA JR, Paulo Roberto; FIORESE, Adriano; KOSLOVSKI, Guilherme P.; Comparação de políticas para configuração de fluxos em redes definidas por software. Computer on the Beach 2016, Anais do Computer on the Beach/Florianópolis/SC, 2016.

- VIEIRA JR, Paulo Roberto; FIORESE, Adriano; KOSLOVSKI, Guilherme P.; MARCONDES, Anderson H. M., Uma Abordagem para Redução de Consumo de Recursos de Serviços de Voz sobre IP baseada em Redes Definidas por Software. XVI Escola Regional de Alto Desempenho do Rio Grande do Sul - ERAD/RS 2017, n. Anais do ERAD/RS, 2017.

- VIEIRA JR, Paulo Roberto; FIORESE, Adriano; KOSLOVSKI, Guilherme P.; MARCONDES, Anderson H. M., Gerenciamento de recursos de serviços de Voz so-

bre IP baseado em Redes Definidas por Software. XXII Workshop de Gerência e Operação de Redes e Serviços - WGRS 2017, Anais do WGRS/Belém-PA, 2017.

- VIEIRA JR, Paulo Roberto; FIORESE, Adriano; KOSLOVSKI, Guilherme P.; MARCONDES, Anderson H. M., SDNVoIP: gerenciamento de recursos de serviços de Voz sobre IP baseado em Redes Definidas por Software. SBRC 2017 - Salão de Ferramentas, Anais do SBRC/Belém-PA, 2017.

Ainda, no decorrer da execução deste trabalho houve participação nas seguintes publicações:

- A. H. S. Marcondes, G. Diel, F. R. de Souza, P. R. Vieira Jr, A. Fiorese and G. P. Koslovski, "Executing distributed applications on SDN-based Data Center: A study with NAS Parallel Benchmark,"2016 7th International Conference on the Network of the Future (NOF), Buzios, 2016, pp. 1-3.
- A. H. S. Marcondes, G. Diel, F. R. de Souza, P. R. Vieira Jr, A. Fiorese and G. P. Koslovski, "Análise de Aplicações Distribuídas em SDN: um Estudo com o Benchmark NAS", VI Brazilian Symposium on Computing Systems Engineering (SBESC), João Pessoa- Paraíba, 2016.

REFERÊNCIAS

AKYILDIZ, I. F. et al. A roadmap for traffic engineering in sdn-openflow networks. **Computer Networks**, Elsevier, v. 71, n. Supplement C, p. 1–30, 2014. ISSN 1389-1286. Disponível em: <http://www.sciencedirect.com/science/article/pii/S1389128614002254>.

ALI, A. A.; VASSILARAS, S.; NTAGKOUNAKIS, K. A comparative study of bandwidth requirements of voip codecs over wimax access networks. In: INTERNATIONAL CONFERENCE ON NEXT GENERATION MOBILE APPLICATIONS, SERVICES AND TECHNOLOGIES, 3., 2009, Cardiff, Wales, UK. **Proceedings ...** Cardiff, Wales, UK: IEEE, 2009. p. 197–203. ISSN 2161-2889.

ARCHITECTURE, F. **Floodlight Is an Open SDN Controller**. 2017. <https://floodlight.atlassian.net/wiki/display/floodlightcontroller/Architecture>. Acessado em: 2017-06-17.

ARTUSO, M. et al. Towards flexbile sdn-based management for cloud-based mobile networks. In: IEEE/IFIP NETWORK OPERATIONS AND MANAGEMENT SYMPOSIUM, 2016, Turkey, Istanbul. **Proceedings ...** Turkey, Istanbul: IEEE, 2016. p. 474–480.

CAMARILLO, G. **Session Description Protocol (SDP) Format for Binary Floor Control Protocol (BFCP) Streams**. Jorvas, Finland, 2006. 1-12 p. Disponível em: <https://tools.ietf.org/html/rfc4583>.

CHEN, W.-E.; HUANG, Y.-L.; CHAO, H.-C. Nat traversing solutions for sip applications. **EURASIP Journal on Wireless Communications and Networking**, Springer, v. 2008, n. 1, p. 1–9, 2008.

CHIRIVELLA-PEREZ, E. et al. Towards a sdn-based architecture for analyzing network traffic in cloud computing infrastructures. In: INTERNATIONAL CONFERENCE ON SOFTWARE, TELECOMMUNICATIONS AND COMPUTER NETWORKS (SOFTCOM), 2015, Split, Croatia. **Proceedings ...** Split, Croatia: IEEE, 2015. p. 295–299.

COSTA, L. R. et al. Asterisk pbx capacity evaluation. In: IEEE INTERNATIONAL PARALLEL AND DISTRIBUTED PROCESSING SYMPOSIUM WORKSHOP, 2015, Hyderabad, India. **Proceedings ...** Hyderabad, India: IEEE, 2015. p. 519–524.

COULOURIS, G. et al. **Sistemas Distribuídos**: Conceitos e projetos. 4. ed. Porto Alegre, Brasil: Bookman, 2007. 74–76 p.

COULOURIS, G. et al. **Sistemas Distribuídos**: Conceitos e projetos. 4. ed. Porto Alegre, Brasil: Bookman, 2007. 81,82 p.

DIGIUM, I. **Asterisk**. 2017. <https://wiki.asterisk.org>. Acessado em: 2017-03-12.

EDEN, A. H. Three paradigms of computer science. **Minds and machines**, Springer, v. 17, n. 2, p. 135–167, 2007.

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: ASIA PACIFIC SIGNAL AND INFORMATION PROCESSING ASSOCIATION ANNUAL SUMMIT AND CONFERENCE, 2012, Hollywood, CA, USA. **Proceedings ...** Hollywood, CA, USA: IEEE, 2012. p. 1–8.

FARHADY, H.; LEE, H.; NAKAO, A. Software-defined networking: A survey. **Computer Networks**, v. 81, p. 79 – 95, 2015. ISSN 1389-1286. Disponível em: <http://www.sciencedirect.com/science/article/pii/S1389128615000614>.

FLOODLIGHT. **Floodlight Is an Open SDN Controller**. 2017. <http://www.projectfloodlight.org/floodlight>. Acessado em: 2017-06-15.

FOSTER, N. et al. Frenetic: A network programming language. **SIGPLAN Not.**, ACM, New York, NY, USA, v. 46, n. 9, p. 279–291, set. 2011. ISSN 0362-1340. Disponível em: <http://doi.acm.org/10.1145/2034574.2034812>.

FOUNDATION, O. N. **OpenFlow Switch Specification**. 2012. <https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-spec-v1.3.0.pdf>. Acessado em: 2017-07-16.

FOUNDATION, O. N. **Floodlight Is an Open SDN Controller**. 2015. <https://www.opennetworking.org/>. Acessado em: 2016-10-16.

GIL, A. C. **Como elaborar projetos de pesquisa**. 5. ed. São Paulo, Brasil: Atlas, 2002. 16–17 p.

GOODE, B. Voice over internet protocol (voip). **Proceedings of the IEEE**, v. 90, n. 9, p. 1495–1517, Sep 2002. ISSN 0018-9219.

HANDLEY, M. Why the internet only just works. **BT Technology Journal**, Springer, v. 24, n. 3, p. 119–129, 2006.

HASROUTY, C. A. et al. Sdn-driven multicast streams with adaptive bitrates for voip conferences. In: IEEE INTERNATIONAL CONFERENCE ON COMMUNICATIONS (ICC), 2016, Kuala Lumpur, Malaysia. **Proceedings ...** Kuala Lumpur, Malaysia: IEEE, 2016. p. 1–7.

Hassas Yeganeh, S. et al. Kandoo: a framework for efficient and scalable offloading of control applications. **Proceeding HotSDN '12 Proceedings of the first workshop on Hot topics in software defined networks**, p. 19–24, 2012. ISSN 01464833. Disponível em: <http://dl.acm.org/citation.cfm?id=2342446{\%}5Cnhttp://doi.acm.org/10.1145/2342441.2342>.

HUMERNBRUM, T.; HAGEDORN, B.; GORLATCH, S. Towards efficient multicast communication in software-defined networks. In: INTERNATIONAL CONFERENCE ON DISTRIBUTED COMPUTING SYSTEMS WORKSHOPS (ICDCSW), 2016, Nara, Japan. **Proceedings ...** Nara, Japan: IEEE, 2016. p. 106–113.

JAIN, R.; PAUL, S. Network virtualization and software defined networking for cloud computing: A survey. **IEEE Communications Magazine**, v. 51, n. 11, p. 24–31, 2013. ISSN 01636804.

JUNIOR, P. R. V.; FIORESE, A.; KOSLOVSKI, G. P. Comparação de políticas para configuração de fluxos em redes definidas por software. Computer on the Beach, p. 357–366, 2016. ISSN 23580852. Disponível em: <http://www.computeronthebeach.com.br/arquivos-2016/Anais%20completos%20-%20Computer%20on%20the%20Beach%202016.pdf>.

JUNIOR, P. R. V. et al. Manual do módulo sdnvoip. In: SIMPóSIO BRASILEIRO DE REDES DE COMPUTADORES E SISTEMAS DISTRIBUíDOS, 2017, Belém, PA, Brasil. **Proceedings ...** Belém, PA, Brasil: SBC, 2017. p. 1151–1158. ISSN 2177-496X.

KARAPANTAZIS, S.; PAVLIDOU, F. N. VoIP: A comprehensive survey on a promising technology. **Computer Networks**, Elsevier B.V., v. 53, n. 12, p. 2050–2090, 2009. ISSN 13891286. Disponível em: <http://dx.doi.org/10.1016/j.comnet.2009.03.010>.

KHIRUL, I.; ISLAM, M. K.; HASAN, K. M. A. An efficient approach for nat traversal problem on security of voice over internet protocol. In: 2007 10TH INTERNATIONAL CONFERENCE ON COMPUTER AND INFORMATION TECHNOLOGY, 2007, Dhaka, Bangladesh. **Proceedings ...** Dhaka, Bangladesh: IEEE, 2007. p. 1–5.

KHLIFI, H.; GREGOIRE, J.; PHILLIPS, J. Voip and nat/firewalls: issues, traversal techniques, and a real-world solution. **IEEE Communications Magazine**, IEEE INSTITUTE OF ELECTRICAL AND ELECTRONICS, v. 44, n. 7, p. 93, 2006.

KOPONEN, T. et al. Network virtualization in multi-tenant datacenters. In: **Proceedings of the 11th USENIX Conference on Networked Systems Design and Implementation**. Berkeley, CA, USA: USENIX Association, 2014. (NSDI'14), p. 203–216. ISBN 978-1-931971-09-6.

KREUTZ, D. et al. Software-defined networking: A comprehensive survey. **Proceedings of the IEEE**, v. 103, n. 1, p. 14–76, 2014. Disponível em: <http://www.scopus.com/inward/record.url?eid=2-s2.0-84919935425{&}partnerID=40{&}md5=bfb4c52eb3ed4e0412cdc42828>.

KWON, D. et al. Qoe-based adaptive mvoip service architecture in sdn networks. **The Thirteenth International Conference on Networks**, ICN 2014, p. 73, 2014.

LARA, A.; KOLASANI, A.; RAMAMURTHY, B. Network innovation using openflow: A survey. **IEEE Communications Surveys Tutorials**, v. 16, n. 1, p. 493–512, First 2014. ISSN 1553-877X.

LIN, Y.-D. et al. How nat-compatible are voip applications? **IEEE Communications Magazine**, IEEE, v. 48, n. 12, p. 58–65, 2010.

LINPHONE. **Linphone Open source SIP Phone**. 2001. <https://www.linphone.org/>. Acessado em: 2016-10-15.

LOUGHEED, K.; REKHTER, Y. **A Border Gateway Protocol 4 (BGP-4)**. Ann Arbor, MI 48108, USA, 2006. 1-104 p. Disponível em: <https://www.rfc-editor.org/rfc/rfc4271.txt>.

MARCONI, M. d. A.; LAKATOS, E. M. **Fundamentos de metodologia científica**. 7. ed. São Paulo, Brasil: Atlas, 2003.

MARIBONDO, P. D. S.; FERNANDES, N. C. Avoiding voice traffic degradation in ip enterprise networks using caos. In: **Proceedings of the 2016 Workshop on Fostering Latin-American Research in Data Communication Networks**. New York, NY, USA: ACM, 2016. (LANCOMM '16), p. 34–36. ISBN 978-1-4503-4426-5.

MASOUDI, R.; GHAFFARI, A. Software defined networks: A survey. **Journal of Network and Computer Applications**, v. 67, p. 1 – 25, 2016. ISSN 1084-8045. Disponível em: <http://www.sciencedirect.com/science/article/pii/S1084804516300297>.

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

MEHTA, P.; UDANI, S. Voice over ip. **IEEE Potentials**, [New York, NY]: Institute of Electrical and Electronics Engineers,[1982-, v. 20, n. 4, p. 36–40, Oct 2001. ISSN 0278-6648.

MONSANTO, C. et al. A compiler and run-time system for network programming languages. **SIGPLAN Not.**, ACM, New York, NY, USA, v. 47, n. 1, p. 217–230, jan. 2012. ISSN 0362-1340. Disponível em: <http://doi.acm.org/10.1145/2103621.2103685>.

MONTORO, P.; CASILARI, E. A comparative study of voip standards with asterisk. In: 2009 FOURTH INTERNATIONAL CONFERENCE ON DIGITAL TELECOMMUNICATIONS, 2009, Colmar, France. **Proceedings ...** Colmar, France: IEEE, 2009. p. 1–6.

MORZHOV, S.; ALEKSEEV, I.; NIKITINSKIY, M. Firewall application for floodlight sdn controller. In: 2016 INTERNATIONAL SIBERIAN CONFERENCE ON CONTROL AND COMMUNICATIONS (SIBCON), 2016, Moscow, Russia. **Proceedings ...** Moscow, Russia: IEEE, 2016. p. 1–5.

NG, S. L.; HOH, S.; SINGH, D. Effectiveness of adaptive codec switching voip application over heterogeneous networks. In: 2005 2ND ASIA PACIFIC CONFERENCE ON MOBILE TECHNOLOGY, APPLICATIONS AND SYSTEMS, 2005, Guangzhou, China. **Proceedings ...** Guangzhou, China: IEEE, 2005. p. 7 pp.–7.

PANG, C.; JIANG, Y.; LI, Q. Fade: Detecting forwarding anomaly in software-defined networks. In: 2016 IEEE INTERNATIONAL CONFERENCE ON COMMUNICATIONS (ICC), 2016, Kuala Lumpur, Malaysia. **Proceedings ...** Kuala Lumpur, Malaysia: IEEE, 2016. p. 1–6.

PARK, C. et al. Nat issues in the remote management of home network devices. **IEEE network**, IEEE, v. 22, n. 5, p. 48–55, 2008.

PFAFF, B.; DAVIE, B. **The Open vSwitch Database Management Protocol**. Palo Alto, CA 94304, USA, 2013. 1-35 p. Disponível em: <https://www.rfc-editor.org/rfc/rfc7047.txt>.

PRESUHN, R. **Version 2 of the protocol operations for the simple network management protocol (SNMP)**. San Jose, CA 95131, USA, 2002. 1-31 p. Disponível em: <https://www.rfc-editor.org/rfc/rfc3416.txt>.

QIAO, Z. et al. A new method for voip quality of service control use combined adaptive sender rate and priority marking. In: 2004 IEEE INTERNATIONAL CONFERENCE ON COMMUNICATIONS (IEEE CAT. NO.04CH37577), 2004, Paris, France. **Proceedings ...** Paris, France: IEEE, 2004. v. 3, p. 1473–1477 Vol.3.

REN, T.; XU, Y. Analysis of the new features of openflow 1.4. In: 2ND INTERNATIONAL CONFERENCE ON INFORMATION, ELECTRONICS AND COMPUTER (ICIEAC-2014), 2014, Wuhan, China. **Proceedings ...** Wuhan, China: Atlantis Press, 2014. p. 73–77.

ROSENBERG, J. et al. **SIP: Session Initiation Protocol**. East Hanover, New Jersey, USA, 2002. 1-269 p. Disponível em: <https://www.rfc-editor.org/rfc/rfc3261.txt>.

ROYCHOUDHURI, L.; AL-SHAER, E. S. Adaptive rate control for real-time packet audio based on loss prediction. In: GLOBAL TELECOMMUNICATIONS CONFERENCE, 2004. GLOBECOM '04. IEEE, 2004, Dallas, TX, USA. **Proceedings ...** Dallas, TX, USA: IEEE, 2004. v. 2, p. 634–638 Vol.2.

SCHULZRINNE, H. et al. **Version 2 of the protocol operations for the simple network management protocol (SNMP)**. New York, NY, USA, 2003. 1-104 p. Disponível em: <https://www.rfc-editor.org/rfc/rfc3550.txt>.

SYSTEMS, I. C. **Voice Over IP - Per Call Bandwidth Consumption**. 2016. <http://www.cisco.com/c/en/us/support/docs/voice/voice-quality/7934-bwidth-consume.html>. Acessado em: 2016-10-16.

TECHNOLOGIES, C. T.-L. **TP-LINK WR1043 V3**. 2017. <http://www.tp-link.com.br/products/details/cat-9_TL-WR1043ND.html#specifications>. Acessado em: 2017-03-15.

TEIXEIRA, E. **As três metodologias: acadêmica, da ciência e da pesquisa**. 11. ed. Rio de Janeiro, Brasil: Vozes, 2010. 208 p.

THORPE, C. et al. imos: Enabling voip qos monitoring at intermediate nodes in an openflow sdn. In: IEEE INTERNATIONAL CONFERENCE ON CLOUD ENGINEERING WORKSHOP (IC2EW), 2016, Berlin, Germany. **Proceedings ...** Berlin, Germany: IEEE, 2016. p. 76–81.

TOOTOONCHIAN, A. et al. On controller performance in software-defined networks. **Proceeding Hot-ICE'12 Proceedings of the 2nd USENIX conference on Hot Topics in Management of Internet, Cloud, and Enterprise Networks and Services**, p. 10–10, 2012. Disponível em: <https://www.usenix.org/system/files/conference/hot-ice12/hotice12-final33{_}0.>

UNION, I. T. **ITU-T Recommendation for Codecs.** 1988. <https://www.itu.int/rec/T-REC-G/en>. Acessado em: 2016-10-18.

UNION, I. T. **ITU-T Recommendation G.711, Pulse Code Modulation (PCM) of Voice Frequencies.** 1988. <https://www.itu.int/rec/dologin_pub.asp?lang=e&id=T-REC-G.711-198811-I!!PDF-E&type=items>. Acessado em: 2016-10-18.

UNION, I. T. **ITU-T Recommendation G.726 : 40, 32, 24, 16 kbit/s Adaptive Differential Pulse Code Modulation (ADPCM).** 1990. <https://www.itu.int/rec/dologin_pub.asp?lang=e&id=T-REC-G.726-199012-I!!PDF-E&type=items>. Acessado em: 2016-10-18.

UNION, I. T. **Methods for subjective determination of transmission quality**. 2005. <http://www.itu.int/rec/T-REC-P.800-199608-I/en>. Acessado em: 2016-11-15.

UNION, I. T. **TESTING REQUIREMENTS AND METHODOLOGY**. 2005. <https://www.itu.int/en/ITU-T/C-I/Pages/HFT-mobile-tests/Methodologies.aspx>. Acessado em: 2016-10-18.

UNION, I. T. **ITU-T Recommendation G.723.1, Dual rate speech coder for multimedia communications transmitting at 5.3 and 6.3 kbit/s.** 2006. <https://www.itu.int/rec/T-REC-G.723.1-200605-I/en>. Acessado em: 2016-10-18.

UNION, I. T. **ITU-T Recommendation G.729: Coding of speech at 8 kbit/s using conjugate-structure algebraic-code-excited linear prediction (CS-ACELP).** 2012. <https://www.itu.int/rec/T-REC-G.729-201206-I/en>. Acessado em: 2016-10-18.

VARSHNEY, U. et al. Voice over ip. **Commun. ACM**, ACM, New York, NY, USA, v. 45, n. 1, p. 89–96, jan. 2002. ISSN 0001-0782. Disponível em: <http://doi.acm.org/10.1145/502269.502271>.

VOELLMY, A.; KIM, H.; FEAMSTER, N. Procera: A language for high-level reactive network control. In: **Proceedings of the First Workshop on Hot Topics in Software Defined Networks**. New York, NY, USA: ACM, 2012. (HotSDN '12), p. 43–48. ISBN 978-1-4503-1477-0.

VSWITCH, O. **Open vSwitch is a production quality, multilayer virtual switch**. 2017. <http://openvswitch.org/>. Acessado em: 2017-08-30.

WANG, W. et al. **Forwarding and Control Element Separation (ForCES) Protocol Specification**. Xuezheng Str., Xiasha University Town, Hangzhou, China, 2010. 1-124 p. Disponível em: <https://www.rfc-editor.org/rfc/rfc5810.txt>.

WAZLAWICK, R. **Metodologia de pesquisa para ciência da computação**. 2. ed. Rio de Janeiro, Brasil: Elsevier Brasil, 2014. 168 p.

WEISS, M. B.; HWANG, J. Internet telephony or circuit switched telephony: which is cheaper? CiteSeerX, 1998. Disponível em: <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.127.293>.

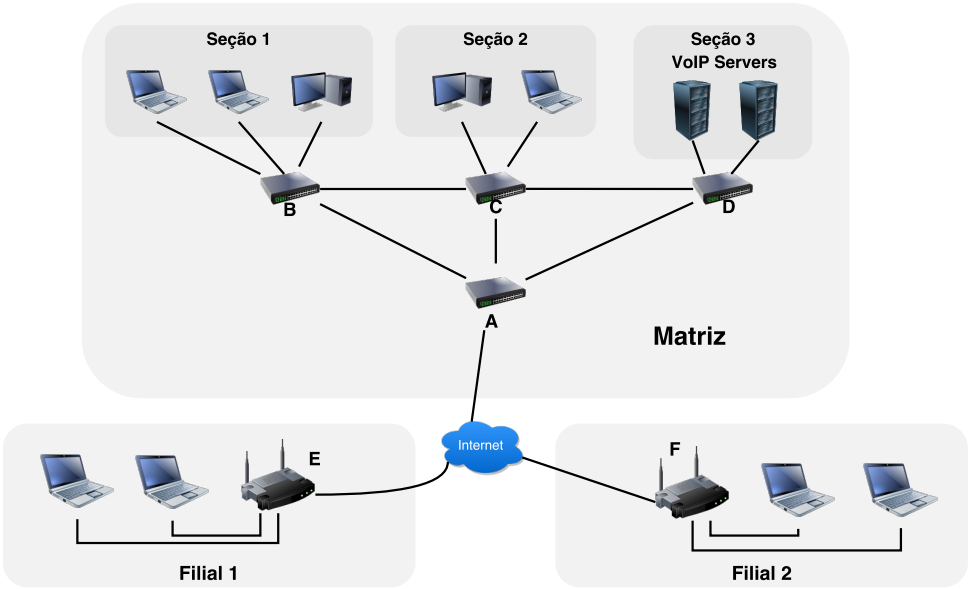
YERYOMIN, Y.; EVERS, F.; SEITZ, J. Solving the firewall and nat traversal issues for sip-based voip. In: 2008 INTERNATIONAL CONFERENCE ON TELECOMMUNICATIONS, 2008, St. Petersburg, Russia. **Proceedings ...** St. Petersburg, Russia: IEEE, 2008. p. 1–6.

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), 2009, Cambodia, Cambodia. **Proceedings ...** Cambodia, Cambodia: IEEE, 2015. p. 318–323. ISSN 1550-445X.

APÊNDICE A – RELATO DE EXPERIÊNCIA

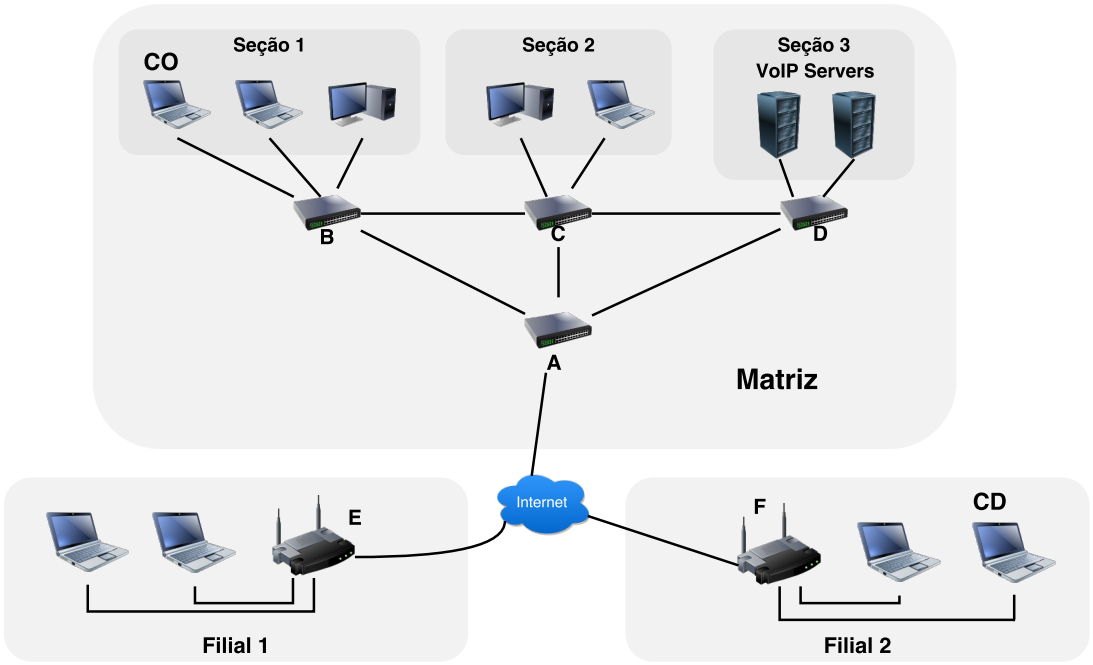
Além dos cenários de avaliação elencados, em uma rede LAN tradicional, um cenário como o de empresas com matriz e filiais, que configura-se como um ambiente LAN em uma infraestrutura WAN, pode obter benefícios da solução apresentada no presente trabalho. A Figura 22 apresenta a topologia de rede de uma empresa que possui uma matriz e duas filiais. Essa topologia exemplo possui dois servidores VoIP e alguns clientes distribuídos em duas seções de rede. As duas filiais estão interligadas com a matriz por meio de um enlace Internet com *firewall* e NAT. As filiais possuem apenas clientes na topologia de rede.

Figura 22 – Cenário matriz - filial sem SDN.



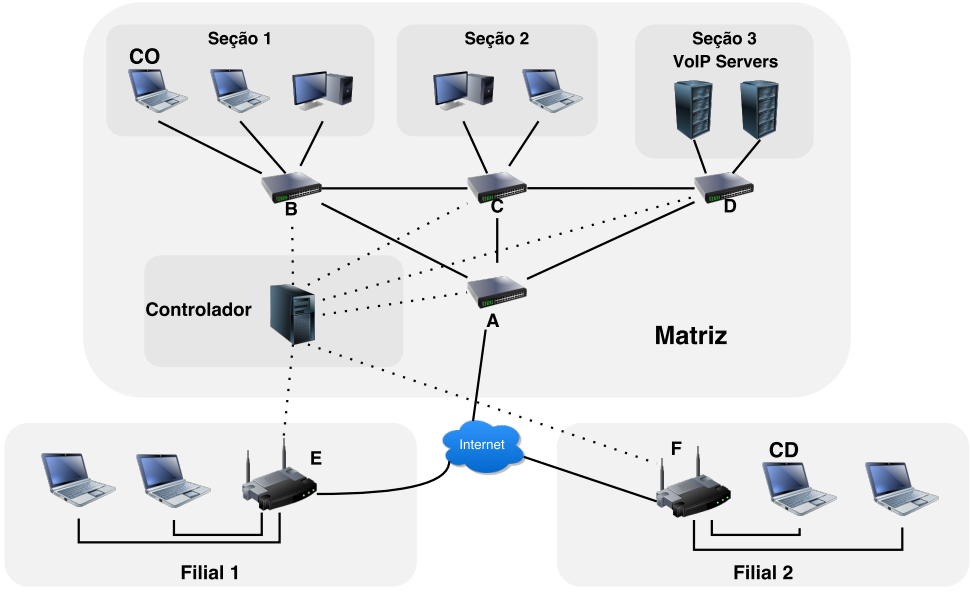
Uma chamada entre um Cliente Origem (CO) localizado na Seção 1 na matriz e um Cliente Destino (CD) localizado na filial 2, apresentado na Figura 23, possui uma rota que passa pelos *switches* B, C, D, A e F. Todo o tráfego de voz de CO para CD e vice-versa irá trafegar pelo servidor VoIP localizado na matriz, aumentando a utilização de CPU, consumindo largura de banda e consumindo uma certa quantidade de megabytes ou até mesmo gigabytes da franquia à qual a empresa tem por direito de acordo com o contrato firmado com a provedora de conectividade Internet.

Figura 23 – Chamada entre matriz e filial.



Com a utilização do módulo SDNVoIP, a chamada entre CO e CD passará a ter uma rota que passa pelos *switches* B, A e F (Figura 24). Todo o tráfego de voz é feito diretamente entre CO e CD. Dessa maneira há uma redução na utilização de CPU, largura de banda no servidor VoIP e uma redução na quantidade de dados trafegados entre os *switches* C e D.

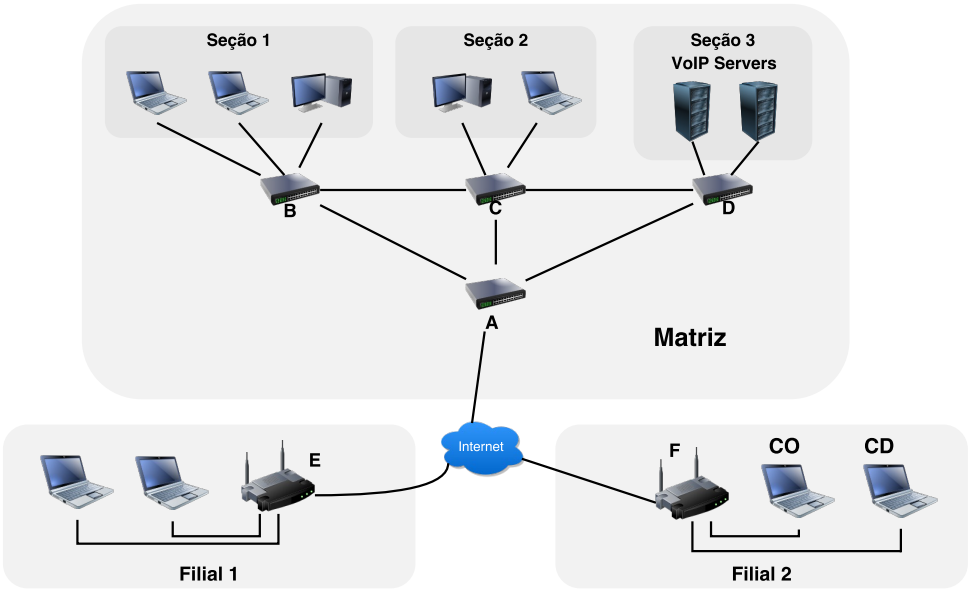
Figura 24 – Chamada entre matriz e filial com SDN.



Outra situação é uma chamada entre CO e CD, ambos localizados na Filial 2 (Figura 25). A chamada possui uma rota passando pelos *switches* F, A e D. Todo o tráfego de voz de CO para CD e vice-versa irá trafegar pelo servidor VoIP locali-

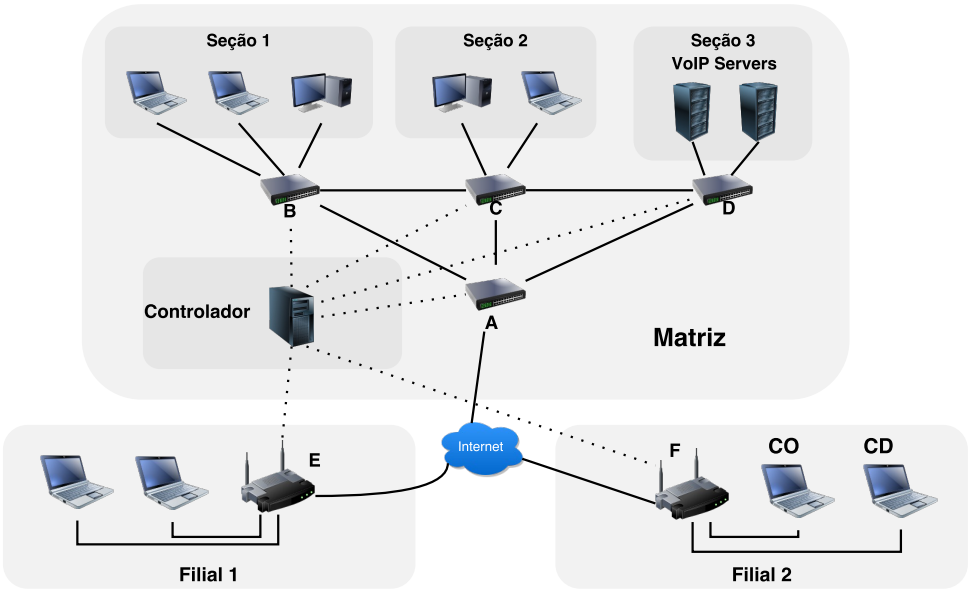
zado na matriz, e assim como no exemplo anterior, aumentando a utilização de CPU, consumindo largura de banda e consumindo dados.

Figura 25 – Chamada entre clientes na Filial 2 sem SDN.



Já com a utilização do módulo SDNVoIP (Figura 26), pode-se além de reduzir a utilização de CPU e largura de banda no servidor VoIP, também reduzir o consumo de dados no link Internet. O módulo detecta a chamada feita entre CO e CD e instala os fluxos necessários no *switch* F fazendo com que o tráfego de voz ocorra diretamente entre os clientes na Filial 2.

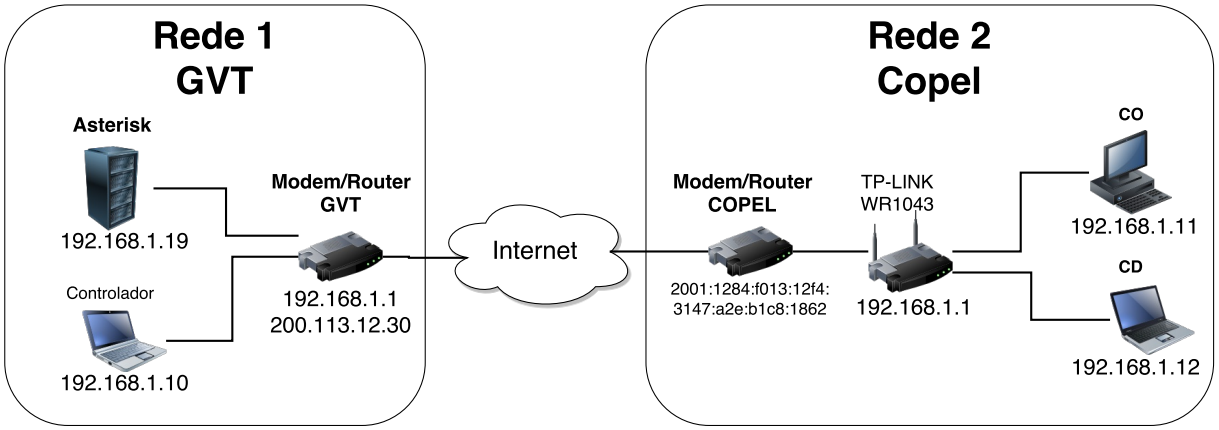
Figura 26 – Chamada entre clientes na Filial 2 com SDN.



Um experimento utilizando duas redes reais com um *link* de Internet cada uma, foi realizado. A Figura 27 apresenta a topologia utilizada, que possui na Rede 1 GVT: um servidor Asterisk, um PC executando o contrador Floodlight com o módulo

SDNVoIP e um *modem/router* com um *link* de Internet fornecido pela empresa GVT. A Rede 2 é composta: um PC e *notebook*, ambos executando o *softphone Linphone*, um roteador TP-LINK WR1043 V3 executando OVS comunicando-se com o controlador na Rede 1 e um *modem/router* com *link* de Internet fornecido pela empresa Copel Telecom.

Figura 27 – Chamada entre clientes na Rede 2 com SDN.



O *modem* da GVT possui um endereço IPv4 privado, um endereço IPv4 público, NAT ativo e foi configurado para fazer encaminhamento das portas 5060 (SIP), 20000 a 25000 (RTP/RTCP) para o endereço do servidor Asterisk (192.168.1.19) e o encaminhamento da porta 6653 do controlador (192.168.1.10).

Para os experimentos, o *modem* da Copel possui um IPv4 privado (192.168.1.1) e um IPv6 público IPv6 (2001:1284:f013:12f4:3147:a2e:b1c8:1862) e NAT ativo. Um roteador TP-LINK WR1043 V3 foi conectado ao *modem* da Copel e os clientes CO e CD conectados ao TP-LINK. O roteador TP-LINK possui um IPv4 interno (192.168.1.1), NAT ativo, executando OVS comunicando-se com o controlador no IP público da Rede 1. Os clientes CO e CD foram configurados para se autenticar no servidor Asterisk na Rede 1.

Uma chamada entre os dois clientes foi executada e o roteador TP-LINK comunicou-se com o controlador que extraiu as informações necessárias dos pacotes SIP/SDP/RTP/RTCP e instalou fluxos no *switch* TP-LINK fazendo com que o áudio entre CO e CD fosse encaminhado diretamente entre os clientes.

Em suma, em ambiente WAN mesmo não sendo totalmente SDN, ou seja, com *switches* SDN nas bordas das redes, apenas interconectando-as, pode-se obter redução na utilização de CPU do servidor VoIP, redução na utilização da largura de banda no segmento de rede física com o qual o servidor VoIP está conectado, ou seja, em uma seção da rede ou entre redes e por fim, uma redução na quantidade de dados trafegados entre as localizações físicas da Rede 1 e Rede 2.

APÊNDICE B – MANUAL DO MÓDULO SDNVOIP

Este apêndice apresenta o manual de instalação e configuração da ferramenta SDNVoIP. Como será possível perceber, para tal é necessária a utilização das seguintes ferramentas adicionais, Java, IDE Eclipse e Git. O manual apresenta os principais arquivos, como alterá-los e a principal classe Java do módulo SDNVoIP. Além disso é apresentado um cenário para demonstração de uso da solução proposta. O Código fonte e um vídeo de demonstração estão disponíveis em forma de *link* no manual.

O formato do manual segue o padrão da SBC para publicação de artigos em congressos, revistas e eventos nacionais. Além disso, o manual foi submetido, aceito, apresentado e publicado nos anais do salão de ferramentas do SBRC 2017 (JUNIOR et al., 2017).

SDNVoIP: gerenciamento de recursos de serviços de Voz sobre IP baseado em Redes Definidas por Software

Paulo Roberto Vieira Jr [1], Anderson H. S. Marcondes[1], Guilherme P. Koslovski[1], Adriano Fiorese [1]

[1]Programa de Pós-Graduação em Computação Aplicada (PPGCA)
Universidade do Estado de Santa Catarina (UDESC) — Joinville, SC — Brasil

paulorvj@gmail.com, {adriano.fiorese, guilherme.koslovski}@udesc.br
anderson.marcondes@sfs.ifc.edu.br

1. Introdução

Este trabalho apresenta o módulo SDNVoIP, uma aplicação para controle de tráfego VoIP em SDN. A aplicação gerencia o tráfego VoIP, reduzindo a utilização de CPU em servidores e a largura de banda utilizada no segmento do servidor VoIP em redes SDN. Para reduzir o uso de CPU nos servidores, SDNVoIP reconfigura as chamadas que utilizem o mesmo *Codec* para realizarem uma comunicação fim-a-fim, sem passar pelo servidor. Ou seja, o cenário da Figura 1(a) é transformado no cenário (b), sem a necessidade reconfigurar os servidores VoIP.

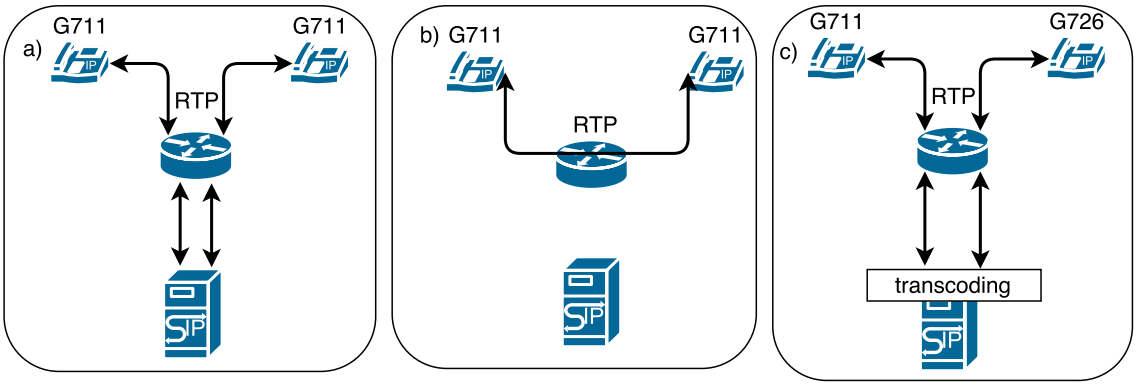


Figura 1. Exemplos de tráfego VoIP com *Codec* único (a e b) e com *Codecs* distintos (c).

A Figura 1 apresenta três maneiras pelas quais pode ocorrer o tráfego de dados em chamadas VoIP: (a) Os dados de uma chamada com o mesmo *Codec* de áudio (G711, por exemplo) são trafegados de um cliente para o outro através do servidor VoIP. Nesse caso o servidor está no caminho do áudio e dois canais são criados, um para cada cliente. O servidor VoIP lê o áudio de um canal e escreve no outro, aumentando o uso de memória e de CPU durante a leitura e escrita dos canais. Essa configuração é comumente utilizada para monitorar chamadas, coletando informações relacionadas com a qualidade do atendimento prestado por centrais de vendas. (b) O tráfego de sinalização de clientes com o mesmo *Codec* de áudio e os dados de voz são trafegados diretamente entre os clientes. Nesse caso, o servidor VoIP não está no caminho do áudio e não é possível gravar a chamada, acrescentar música de fundo, transferir ou colocar a chamada em espera. Entretanto, essa abordagem reduz o uso de CPU, memória alocada e largura de banda utilizada

pelo servidor. (c) Clientes com *Codecs* diferentes e os dados são trafegados sempre pelo servidor VoIP. Semelhante ao cenário (a), o servidor está no caminho do áudio e dois canais são criados, um para cada cliente. Entretanto, o servidor VoIP efetua a tradução dos pacotes de voz de acordo com os *Codecs* utilizados para que os clientes possam receber e ouvir o áudio [Goode 2002], aumentando o consumo dos recursos do servidor.

2. Requisitos de software

Para instalar e utilizar o módulo SDNVoIP é necessário os seguintes *softwares*:

- Java 1.7 ou posterior
- Eclipse 4.5 ou posterior
- Git

O código fonte pode ser encontrado em:

`https://github.com/paulorvj/sdnvoip`

O projeto foi desenvolvido como um módulo do controlador Floodlight [Project Floodlight 2016] e o código disponível no *link* acima possui um projeto completo e configurado para ser importado na IDE Eclipse, não sendo necessário fazer a instalação isolada do controlador Floodlight. Informações sobre como instalar e construir o controlador Floodlight podem ser encontradas em:

`https://floodlight.atlassian.net/wiki/spaces/floodlightcontroller/pages/1343544/Installation+Guide`

Informações sobre como desenvolver um módulo para o Floodlight podem ser encontradas em:

`https://floodlight.atlassian.net/wiki/spaces/floodlightcontroller/pages/1343513/How+to+Write+a+Module`

2.1. Importando o projeto

Primeiro é necessário fazer um clone do repositório Git, para isso utilize o comando (Figura 2):

```
# git clone https://github.com/paulorvj/sdnvoip.git
```

```
wy63xzy-2:sdnvoip paulorvj$ git clone https://github.com/paulorvj/sdnvoip.git
Cloning into 'sdnvoip'...
remote: Counting objects: 2917, done.
remote: Compressing objects: 100% (896/896), done.
remote: Total 2917 (delta 1971), reused 2917 (delta 1971), pack-reused 0
Receiving objects: 100% (2917/2917), 53.06 MiB | 2.38 MiB/s, done.
Resolving deltas: 100% (1971/1971), done.
wy63xzy-2:sdnvoip paulorvj$
```

Figura 2. Git clone.

Após fazer o clone, abra o Eclipse e vá em “File - Import - Existing projects into workspace” (Figura 3):

Selecione o diretório onde foi realizado o clone do repositório, marque o nome do projeto e clique em “Finish” (Figura 4):

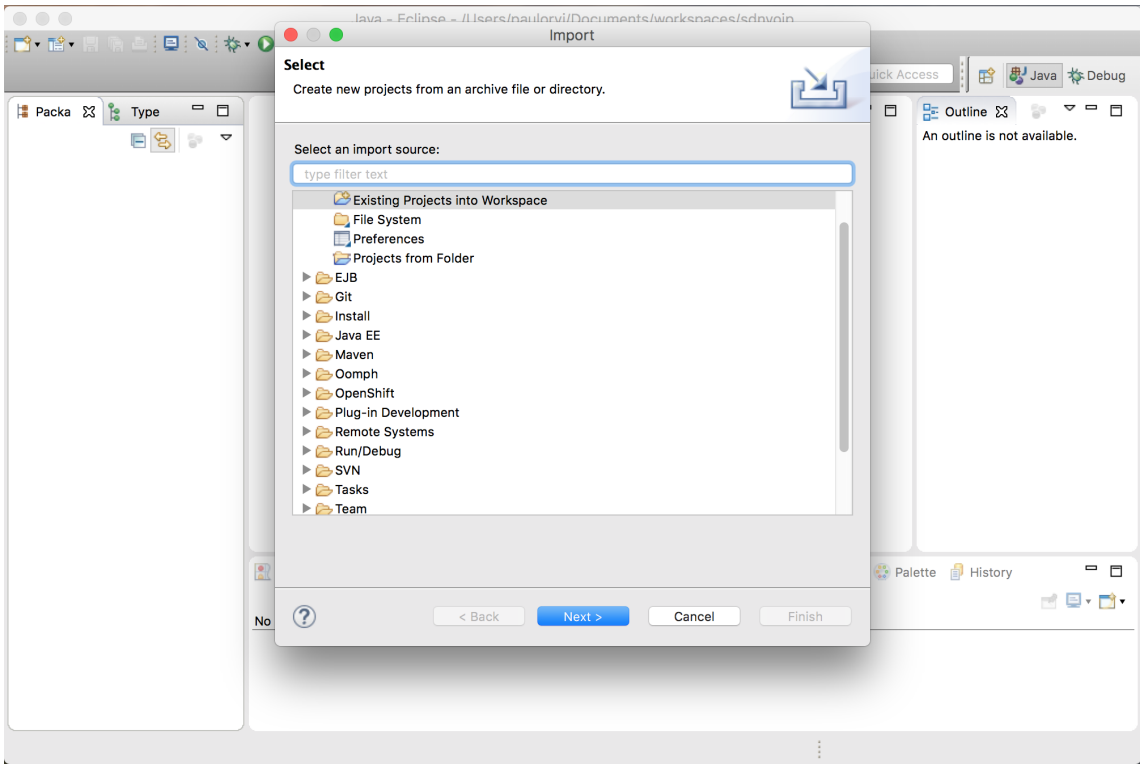


Figura 3. Importando o projeto no eclipse.

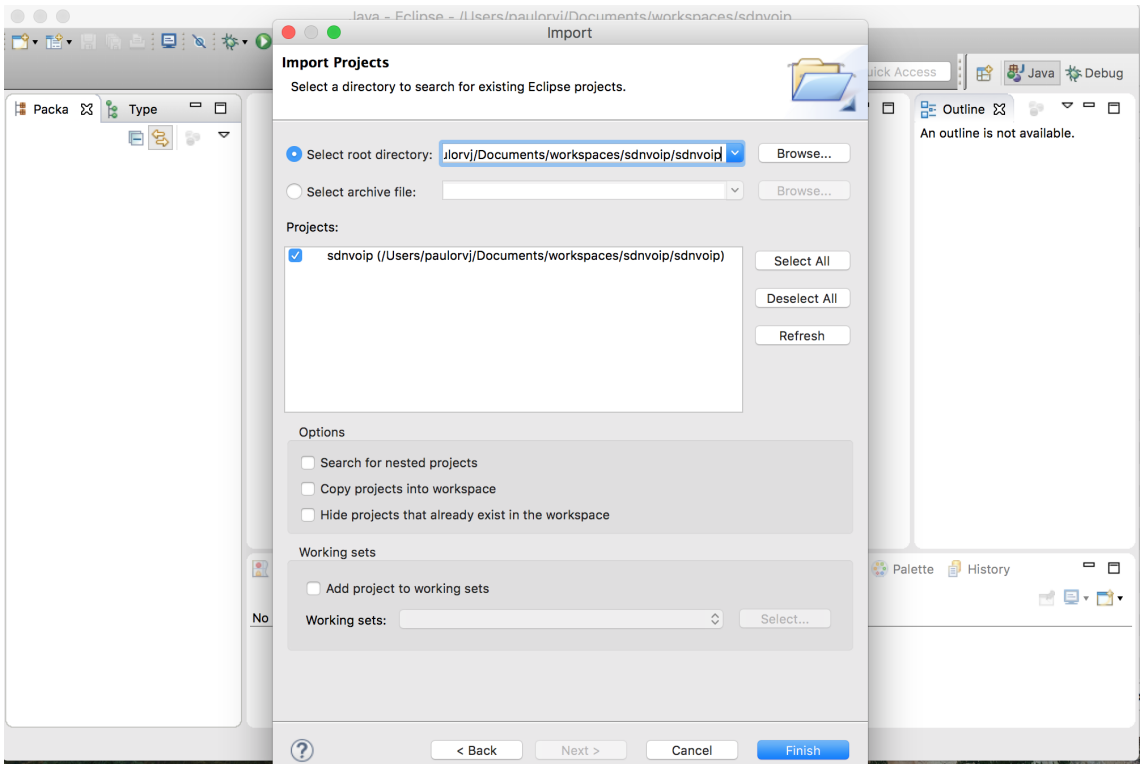


Figura 4. Importando o projeto no eclipse.

Após a importação o projeto estará pronto para execução no Eclipse. Para executar

o controlador clique em “Run - Run configurations”, clique em “Java Application” e depois no botão “New launch configuration” e em “Main class” digite (Figura 5):

net.floodlightcontroller.core.Main

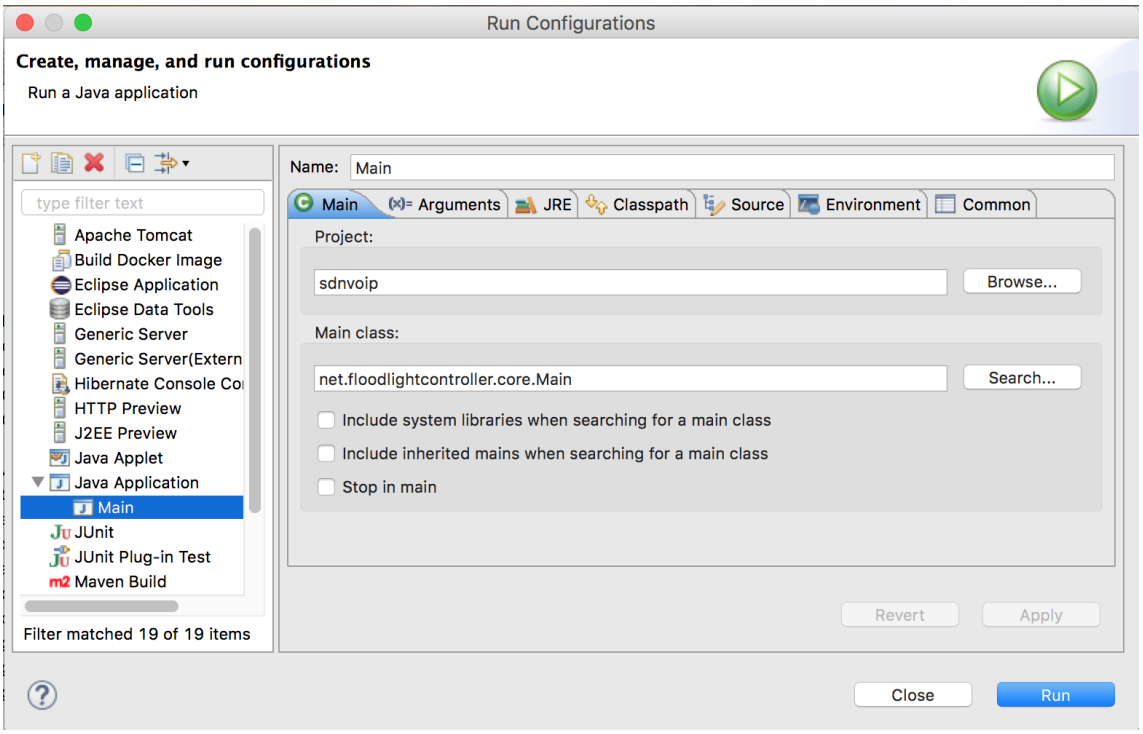


Figura 5. Run configuration.

Clique em “Run” e caso não aconteça nenhum erro o controlador será executado (Figura 6):

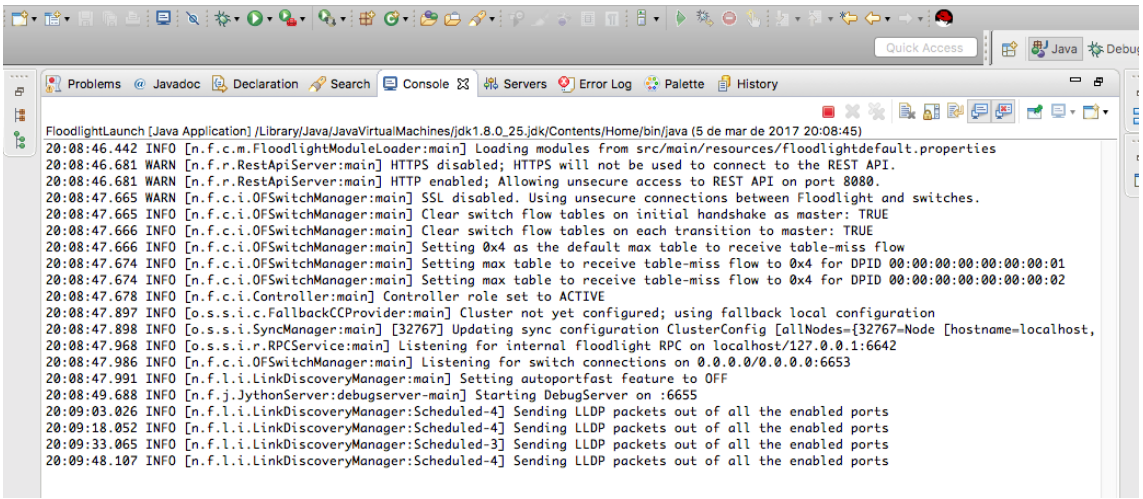


Figura 6. Execução do controlador

A partir desse momento o controlador está pronto para detectar chamadas VoIP que utilizem o mesmo Codec, não é necessário nenhuma configuração adicional.

2.2. Arquivos importantes

O código fonte do módulo encontra-se no seguinte caminho:

/src/main/java/net/floodlightcontroller/voip/Voip.java

Para que o módulo funcione corretamente é necessário que os IPs dos servidores VoIP sejam colocados em um arquivo texto, o arquivo está localizado no seguinte caminho e deve conter um IP por linha:

/sdnvoip/src/main/resources/voipservers.txt

O módulo SDNVoIP mantém informações dos clientes que estão executando chamadas, permitindo a instalação dos fluxos nos *switches* OpenFlow no caminho entre eles.

Quatro fluxos por chamada são instalados nos *switches* SDN. Dois fluxos para o caminho no sentido origem-destino e destino-origem dos pacotes RTP; e dois fluxos para o caminho no sentido origem-destino e destino-origem dos pacotes RTCP. Para cada fluxo VoIP, sete ações [Open Networking Foundation 2012] são aplicadas nos pacotes RTP e RTCP que trafegam no *switch* correspondente:

(i) Alterar o endereço MAC de origem para o endereço MAC do servidor VoIP;
(ii) Alterar o endereço MAC de destino para o endereço MAC do cliente destino;
(iii) Alterar o endereço IP de origem para o endereço IP do servidor VoIP;
(iv) Alterar o endereço IP de destino para o endereço IP do cliente destino;
(v) Alterar a porta RTP/RTCP de origem para a porta RTP/RTCP do servidor VoIP;
(vi) Alterar a porta RTP/RTCP de destino para a porta RTP/RTCP do cliente destino;
(vii) Saída do pacote em uma porta física do *switch*.

As entradas nas tabelas de fluxos devem ser instaladas nos *switches* que compõem o caminho da chamada com um tempo de expiração. Quando o término de uma chamada é sinalizado pelos clientes, a entrada é removida. Em caso de interrupção abrupta da chamada, as entradas relacionadas são automaticamente removidas dos *switches* após o tempo de expiração.

2.3. Topologia

Para demonstrar a solução em um cenário com SDN a seguinte topologia foi utilizada (Figura 7): um servidor VoIP executando Asterisk 13.01, um *switch* TP-Link WR1043 com OpenWRT e OVS habilitados, um controlador, um notebook executando *softphone* Linphone representando o cliente origem (CO) e por fim, um PC executando *softphone* Linphone representando o cliente destino (CD).

A chamada feita entre CO e CD é realizada utilizando o mesmo *Codec* (G711) e todo o áudio por padrão é trafegado passando pelo servidor VoIP.

2.4. Chamadas VoIP sem SDN

A primeira demonstração é com uma chamada entre CO e CD sem a utilização de SDN, dessa maneira todo o audio da chamada passa pelo servidor VoIP.

O servidor VoIP deve estar em execução e os dois clientes são iniciados, podemos observar que os dois clientes (u1 e u2) iniciam uma sessão no servidor VoIP (Figura 8):

Em seguida o cliente u1 incia uma chamada para o cliente u2, observa-se na Figura 9 o inicio da chamada e o tráfego RTP passando pelo servidor VoIP:

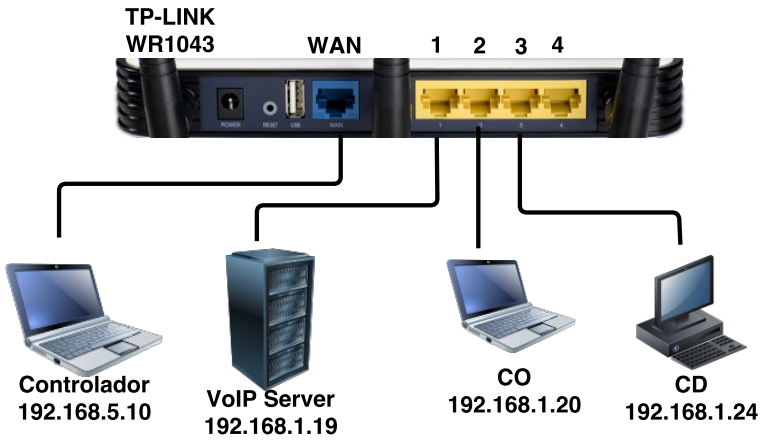


Figura 7. Cenário experimental com SDN.

```
root@paulorvj-EP43-DS3L: ~
root@paulorvj-EP43-DS3L:~# asterisk -rvvvvvvvvg
Asterisk 13.1.0~dfsg-1.1ubuntu4, Copyright (C) 1999 - 2014, Digium, Inc. and others.
Created by Mark Spencer <markster@digium.com>
Asterisk comes with ABSOLUTELY NO WARRANTY; type 'core show warranty' for details.
This is free software, with components licensed under the GNU General Public
License version 2 and other licenses; you are welcome to redistribute it under
certain conditions. Type 'core show license' for details.
=========================================================================
Connected to Asterisk 13.1.0~dfsg-1.1ubuntu4 currently running on paulorvj-EP43-DS3L (pid = 9866)
paulorvj-EP43-DS3L*CLI>
paulorvj-EP43-DS3L*CLI>
paulorvj-EP43-DS3L*CLI>
    -- Registered SIP 'u1' at 192.168.1.20:20000
    -- Registered SIP 'u2' at 192.168.1.24:20001
```

Figura 8. Registro dos clientes no servidor VoIP

```
  == Using SIP RTP CoS mark 5
    -- Executing [u2@teste:1] Answer("SIP/u1-00000002", "") in new stack
       > 0x7fee24008730 -- Probation passed - setting RTP source address to 192.168.1.20:25000
    -- Executing [u2@teste:2] Dial("SIP/u1-00000002", "SIP/u2") in new stack
  == Using SIP RTP CoS mark 5
    -- Called SIP/u2
    -- SIP/u2-00000003 is ringing
    -- SIP/u2-00000003 answered SIP/u1-00000002
    -- Channel SIP/u1-00000002 joined 'simple_bridge' basic-bridge <35d483d0-33f6-4410-8c63-dbfd8e424018>
    -- Channel SIP/u2-00000003 joined 'simple_bridge' basic-bridge <35d483d0-33f6-4410-8c63-dbfd8e424018>
       > Bridge 35d483d0-33f6-4410-8c63-dbfd8e424018: switching from simple_bridge technology to native_r
tp
       > 0x7fee48005ef0 -- Probation passed - setting RTP source address to 192.168.1.24:25002
paulorvj-EP43-DS3L*CLI> rtp set debug on
RTP Debugging Enabled
Sent RTP P2P packet to 192.168.1.24:25002 (type 08, len 000160)
Sent RTP P2P packet to 192.168.1.20:25000 (type 08, len 000160)
Sent RTP P2P packet to 192.168.1.24:25002 (type 08, len 000160)
Sent RTP P2P packet to 192.168.1.24:25002 (type 08, len 000160)
Sent RTP P2P packet to 192.168.1.20:25000 (type 08, len 000160)
```

Figura 9. Chamada VoIP e tráfego RTP

2.5. Chamadas VoIP com SDN

Para a demonstração com SDN foi executado o controlador Floodlight e feita a reconfiguração do *switch* TP-Link WR1043 com OpenWRT e OVS habilitados e comunicando-se com o controlador. Assim como no cenário sem SDN, todo o áudio entre os clientes é trafegado pelo servidor VoIP. Entretanto, após o controlador obter todas as informações necessárias, quatro fluxos para cada chamada são instalados no *switch*, fazendo com que o áudio das chamadas não trafegue pelo servidor VoIP, fluindo diretamente entre os clientes.

Para comprovar a redução na utilização de recursos, dois cenários foram utilizados para realização dos experimentos. Um cenário sem a utilização do paradigma SDN e outro com SDN, ambos em ambiente LAN. Foi coletado a taxa de utilização de CPU utilizado o comando mpstat. O intervalo de coleta foi de 1 segundo durante o período de cinco minutos. Os dados foram tratados da seguinte maneira: para cada amostra foi calculada a média aritmética, em seguida foi calculada a variância e o desvio padrão. O servidor utilizado possui processador Intel Quad Core 3.4GHz, com 8GB de memória RAM.

A Tabela 1 apresenta os dados obtidos de utilização de CPU em um servidor VoIP com as respectivas cargas. Observa-se que 300 chamadas simultâneas utilizando o mesmo Codec e todo o tráfego passando pelo servidor VoIP, gera uma taxa de utilização de CPU de aproximadamente 27%. Já com o uso de SDN, todo o tráfego gerado pelas chamadas é encaminhado diretamente entre os clientes, fazendo com que a CPU do servidor VoIP fique aproximadamente 99% do tempo ociosa.

Tabela 1. Comparação do uso de CPU

Sem SDN						
Chamadas simultâneas	**50**	**100**	**150**	**200**	**250**	**300**
Uso de CPU%	4,03	9,22	13,92	19,35	22,78	27,92
Variância	0,53	5,59	1,33	1,04	1,37	2,74
Desv. Padrão	0,13	0,37	0,21	0,17	0,21	0,30
Com SDN						
Uso de CPU%	0,75	0,73	0,71	0,67	0,71	0,66
Variância	0,01	0,00	0,00	0,00	0,00	0,00
Desv. Padrão	0,08	0,01	0,03	0,06	0,03	0,02

Um video demonstrando o funcionamento do módulo pode ser encontrado em:

`https://www.youtube.com/watch?v=f9tir_EPIIU`

3. Considerações Finais

O presente trabalho agregou benefícios do uso de SDN na área de telecomunicações, mais especificamente na utilização de SDN para reduzir a utilização de recursos em um servidor VoIP. Os resultados obtidos mostraram que é possível reduzir a utilização de CPU em um servidor VoIP através da criação de fluxos que redirecionam o tráfego de voz diretamente entre as partes envolvidas que utilizam o mesmo Codec. Nesse caso, há também a redução na utilização da largura de banda no servidor VoIP e consequentemente redução no tráfego de dados em uma seção da rede.

A redução da utilização de CPU permite que haja maior disponibilidade de processador para chamadas que precisem de tradução de Codec, assim como a redução na largura de banda pode permitir que mais chamadas possam ser adicionadas no servidor, além de reduzir o tráfego em uma seção da rede(ex: *switch*/servidor, servidor/*switch*). Por fim, é possível redimensionar a quantidade de chamadas suportada pelos servidores VoIP existentes em uma rede, sem a necessidade de aquisição de outro servidor para suportar mais chamadas.

Referências

Chen, W.-E., Huang, Y.-L., and Chao, H.-C. (2008). Nat traversing solutions for sip applications. *EURASIP Journal on Wireless Communications and Networking*, 2008(1):1–9.

Goode, B. (2002). Voice over internet protocol (voip). *Proceedings of the IEEE*, 90(9):1495–1517.

Handley, M. (2006). Why the internet only just works. *BT Technology Journal*, 24(3):119–129.

Jain, R. and Paul, S. (2013). Network virtualization and software defined networking for cloud computing: A survey. *IEEE Communications Magazine*, 51(11):24–31.

Karapantazis, S. and Pavlidou, F. N. (2009). VoIP: A comprehensive survey on a promising technology. *Computer Networks*, 53(12):2050–2090.

Khirul, I., Islam, M. K., and Hasan, K. A. (2007). An efficient approach for nat traversal problem on security of voice over internet protocol. In *10th International Conference on Computer and information technology, 2007*, pages 1–5. IEEE.

Khlifi, H., Gregoire, J., and Phillips, J. (2006). Voip and nat/firewalls: issues, traversal techniques, and a real-world solution. *IEEE Communications Magazine*, 44(7):93.

Kreutz, D., Ramos, F. M. V., Verissimo, P. E., Rothenberg, C. E., Azodolmolky, S., and Uhlig, S. (2014). Software-defined networking: A comprehensive survey. *Proceedings of the IEEE*, 103(1):14–76.

Lin, Y.-D., Tseng, C.-C., Ho, C.-Y., and Wu, Y.-H. (2010). How nat-compatible are voip applications? *IEEE Communications Magazine*, 48(12):58–65.

McKeown, N., Anderson, T., Balakrishnan, H., Parulkar, G., Peterson, L., Rexford, J., Shenker, S., and Turner, J. (2008). Openflow: enabling innovation in campus networks. *ACM SIGCOMM Computer Communication Review*, 38(2):69–74.

Open Networking Foundation (2012). OpenFlow Switch Specification. `http://www.cs.yale.edu/homes/yu-minlan/teach/csci599-fall12/papers/openflow-spec-v1.3.0.pdf`.

Park, C., Jeong, K., Kim, S., and Lee, Y. (2008). Nat issues in the remote management of home network devices. *IEEE network*, 22(5):48–55.

Project Floodlight (2016). Project floodlight: Open source software for building software-defined networks. `http://www.projectfloodlight.org/floodlight`.

Yeryomin, Y., Evers, F., and Seitz, J. (2008). Solving the firewall and nat traversal issues for sip-based voip. In *Telecommunications (ICT), International Conference on*, pages 1–6. IEEE.