Nadyan Suriel Pscheidt

Abordagem para Distribuição de Vídeo Baseada em Redes Definidas por Software

Joinville

2017

UNIVERSIDADE DO ESTADO DE SANTA CATARINA
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

Nadyan Suriel Pscheidt

ABORDAGEM PARA DISTRIBUIÇÃO DE VÍDEO BASEADA EM REDES DEFINIDAS POR *SOFTWARE*

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

Charles Christian Miers
Orientador

Joinville, Dezembro de 2017

ABORDAGEM PARA DISTRIBUIÇÃO DE VÍDEO BASEADA EM REDES DEFINIDAS POR *SOFTWARE*

Nadyan Suriel Pscheidt

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

Banca Examinadora

Charles Christian Miers - Doutor (orientador)

Guilherme Piegas Koslovski - Doutor

Débora Cabral Nazário - Doutora

Agradecimentos

Agradeço acima de tudo aos meus pais, por tornarem essa jornada possível e principalmente por me ensinarem a ser quem eu sou.

A Larissa, por ter me apoiado e acreditado em mim em tantos finais de semana dentro de casa, escrevendo parágrafos quase ilegíveis sobre transmissão de vídeo.

A Tarissa, Ricardo e Enzo por sempre dedicarem parte de seus tempos a me manter animado e alegre.

Aos meus amigos e colegas, por terem me acompanhado nessa jornada.

Finalmente, aos meus professores, pelos ensinamentos e palavras de conhecimento e sabedoria.

"A imaginação muitas vezes nos conduz a mundos que nunca fomos, mas sem ela não iremos a nenhum lugar."

Carl Sagan

Resumo

As redes de computadores precisam tratar uma quantidade cada vez maior de dados trafegados, principalmente quando os dados em questão são vídeos. A gerência de fluxos de rede, através de *Software Defined Network* (SDN), mostra-se como uma alternativa promissora para otimizar o consumo de recursos. Explorar características específicas de aplicações de vídeo, dependentes de fluxo de redes, através da gerência destes fluxos via SDN, permite ganho de escalabilidade sem consumir grandes recursos da infraestrutura do provedor. Esse trabalho apresenta a solução MMFA, uma alternativa escalável a aplicações de vídeo, sem alterações na arquitetura nem de clientes nem de servidores. A solução MMFA é agnóstica a aplicação do cliente e atua na gerência de fluxos reduzindo tráfego redundante entre servidores e usuários pertencentes a uma mesma rede de distribuição SDN.

Palavras-chaves: *Transmissão de vídeo*, *Vídeo Sob Demanda*, *Redes Definidas por Software.*

Abstract

Video transmission applications overload computer network infrastructures wich, with the accuracy and quality, need to handle large amounts of data. The management of network flows through Software Defined Networks (SDN), presents as a promising alternative to optimize bandwidth consumption. SDN allows to take advantage of characteristics specific of video applications, depending on the network flow via SDN, enabling scalability without infrastructure increment. Thus, we propose the MMFA solution, a scalable alternative to video applications such as NVoD, without changing the architecture of either clients or servers. MMFA is agnostic to application and acts on the management of flows by reducing redundant traffic between servers and users on the same video distribution network.
Keywords: *Video Streaming*, *Video On Demand*, *Software Defined Network*.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas

ALTO *Application Layer Traffic Optimization*

CDN *Content Delivery Network*

HTTP *Hypertext Transfer Protocol*

IGMP *Internet Group Management Protocol*

IVoD *Interactive Video On Demand*

IPTV *Internet Protocol Television*

LAN *Local Area Networks*

NVoD *Near Video On Demand*

MMFA *Multimedia Flow Aggregator*

P2P *Peer-to-Peer*

QoS *Quality of Service*

RTCP *Real-time Transport Control Protocol*

RTP *Real-time Transport Protocol*

RTSP *Real Time Streaming Protocol*

SCTP *Stream Control Transmission Protocol*

SDN *Software Defined Network*

TCP *Transmission Control Protocol*

TVoD *True Video On Demand*

UDP *User Datagram Protocol*

VoD *Video On Demand*

WAN *Wide Area Networks*

1 Introdução

A transmissão de vídeos através de redes internet é um serviço que vem sendo cada vez mais utilizado (JULURI; TAMARAPALLI; MEDHI, 2016; NAYFEH; SARHAN, 2013). As empresas fornecedoras de conteúdo em vídeo que disponibilizam esse tipo de serviço (*e.g.,* IPTV), precisam fornecer o serviço ao consumidor de forma satisfatória, sem problemas de atraso ou indisponibilidade do vídeo escolhido pelo usuário (JULURI; TAMARAPALLI; MEDHI, 2016). Além da agilidade na transmissão, esses serviços precisam suportar diversos usuários simultaneamente, quantidade que aumenta de forma considerável ano após ano (NAYFEH; SARHAN, 2016).

Serviços de transmissão de conteúdo em vídeo são comumente denominados de Vídeo Sob Demanda ou *Video On Demand* (VoD). VoD por sua vez, pode ser subdividido em duas categorias: *True Video On Demand* (TVoD) e *Near Video On Demand* (NVoD), que possuem características distintas. Enquanto o TVoD oferece uma maior interação com o usuário, os sistemas NVoD são bastante limitados no sentido de interações, porém amplamente utilizados (YU, 2016; NAYFEH; SARHAN, 2016).

O crescimento da quantidade de usuários solicitando os serviços de transmissão de vídeo afetou também a forma como as redes são gerenciadas e organizadas, tornando essa tarefa mais complexa do que costumava ser (HU; HAO; BAO, 2014). Este fato também fica evidente com o avanço da qualidade dos conteúdos multimídia, com resoluções cada vez maiores, o que aumenta a quantidade de dados trafegados na rede, tornando mais difícil manter um desempenho aceitável e causando congestionamentos nos enlaces de comunicação (NAYFEH; SARHAN, 2016). Algumas formas de melhorar o desempenho e a redução de tráfego nos enlaces foram criadas, como as *Content Delivery Network*s (CDNs), que distribuem geograficamente os servidores para diminuir a carga em cada *link* de comunicação (PATHAN; BUYYA, 2007). Um dos motivos para tamanho tráfego nas redes é a redundância dos pacotes trafegados, *i.e.,* pacotes iguais sendo transmitidos de forma duplicada ou redundante (GOYA, 2014). Os *backbones* podem usar de alguns subterfúgios como enlaces redundantes, fibra ótica e sistemas de *cache*, porém carecem de soluções uniformizadas por conta de cada rede ser um sistema autônomo, o que dificulta o uso de tecnologias para mensuração e implementação do *Quality of Service* (QoS).

O aumento do tráfego (assim como o tráfego multimídia) em redes locais, tem emergido as questões relacionando congestionamento nos enlaces para dentro das redes das organizações e empresas. Porém, por serem redes locais caracterizando um ambiente mais homogêneo e gerenciável, tais questões podem ser resolvidas de forma mais eficiente, principalmente pelas redes gerenciadas, que consistem no controle de atividades e monitoramento de tais redes, assim como seus recursos. Uma rede gerenciada pode ser um sistema, *Peer-to-Peer* (P2P), SDN, dentre outros.

Para auxiliar na gerência e desempenho da rede para diversas aplicações, incluindo transmissão de vídeos, surgiram as chamadas redes definidas por *software*, ou SDN, que diferente das Redes Convencionais, dissociam o plano de dados do plano de controle, comunicando-se através de um protocolo (LARA; KOLASANI; RAMAMURTHY, 2014). O *OpenFlow*, por exemplo, é um dos protocolos mais conhecidos na área e tornou-se padrão para as SDN. Com essa abordagem é possível modificar, da forma desejada, as tabelas de fluxo do *switch* que está sendo configurado (MASOUDI; GHAFFARI, 2016), explicitando como cada pacote deve ser roteado através da topologia da rede, auxiliando assim na entrega do conteúdo para os usuários simultâneos e na otimização do uso dos recursos da rede.

O trabalho aqui descrito propõe uma solução para o problema dos fluxos redundantes em NVoD, através da agregação de tais fluxos semelhantes. Essa agregação reduz o consumo do uso de recursos da rede, como a largura de banda no enlace de comunicação entre o servidor e o *switch* SDN, possibilitando uma maior escalabilidade do sistema, no quesito quantidade de usuários simultâneos.

1.1 Objetivo geral e objetivos específicos

O objetivo do trabalho, de forma geral, é especificar e implementar em uma rede LAN (ou WAN, caso tal suporte SDN de ponta a ponta), a transmissão e distribuição de vídeos em uma rede na qual existem clientes requisitantes e servidores fornecendo o conteúdo, de forma a utilizar conceitos e abordagens de Redes Definidas por *Software*. O propósito é diminuir o consumo de recursos de rede, como a largura de banda e a carga no enlace de comunicação, entre o servidor e o consumidor do conteúdo, possibilitando escalar a quantidade de usuários simultâneos recebendo o conteúdo.

Para alcançar o objetivo geral proposto, são necessários alguns objetivos específicos, como:

1. Estudar a forma de transmissão convencional e os protocolos envolvidos nessas transmissões, assim como os problemas da área;
2. Estudar o paradigma de Redes Definidas por *Software*, e como esse paradigma pode auxiliar na resolução de alguns dos problemas encontrados;
3. Definir os requisitos e elaborar uma abordagem para a transmissão de vídeo utilizando SDN para que o controlador reconheça os fluxos e agregue semelhantes evitando redundâncias na rede;
4. Desenvolvimento e implementação da abordagem proposta; e
5. Comparar experimentalmente a abordagem implementada com a transmissão em Rede Convencional.

O estudo aqui descrito apresenta todos os objetivos específicos, desde o estudo das formas de transmissão e protocolos, problemas existentes e elaboração da abordagem (1, 2 e 3), até o desenvolvimento, implementação e experimentação da solução implementada (4 e 5).

1.2 Organização do trabalho

Este trabalho é composto de cinco capítulos. Primeiramente, o Capítulo 2 contextualiza a transmissão de vídeos pela rede, assim como introduz os conceitos envolvendo Vídeo Sob Demanda, juntamente com o funcionamento, protocolos, abordagens para a transmissão e principais problemas da área. Nesse capítulo também é apresentada a problemática e o escopo do trabalho. Esses conceitos são necessários para o entendimento dos sistemas de transmissão de conteúdo em vídeo, a fim de encontrar a melhor forma de abordar tais sistemas com o intuito de resolver alguns problemas em aberto.

O Capítulo 3 apresenta os principais conceitos envolvendo as Redes Definidas por *Software*. Assim como, trabalhos relacionados que utilizam essa abordagem para a transmissão de vídeo que são analisados de acordo com os problemas presentes na área,

avaliando se tais problemas são resolvidos ou não para cada abordagem. O entendimento dos conceitos de SDN são necessários para aplicá-los nos sistemas de Vídeo Sob Demanda de forma a utilizá-los para a melhoria do sistema da forma desejada. A análise dos trabalhos relacionados é importante para definir uma abordagem ainda não utilizada.

Na sequência, o Capítulo 4 apresenta uma solução, denominada MMFA, em Redes Definidas por *Software* para a transmissão de vídeos propondo satisfazer os requisitos para a resolução dos problemas definidos na problemática do estudo. O Capítulo 5 descreve os detalhes e tecnologias utilizadas para a implementação da solução MMFA, assim como os experimentos realizados e seus resultados para a comparação do desempenho da solução MMFA com as Redes Convencionais.

Por fim, o Capítulo 6 descreve as considerações finais obtidas no desenvolvimento da solução MMFA, principais problemas encontrados e trabalhos futuros para a melhoria da solução proposta.

1.3 Método de pesquisa

O método empregado para a execução do trabalho é o de Pesquisa Referenciada. Isso permite o conhecimento da área, assim como seus problemas e desafios. Através desses problemas, é possível estabelecer os objetivos do estudo, que por sua vez é proposto para resolver tais problemas.

Com o intuito de alcançar os objetivos propostos, algumas etapas são necessárias. Desta forma, para a primeira parte do trabalho (ou estudo da literatura e elaboração da abordagem) são definidas as etapas:

- Compreender o funcionamento das transmissões de vídeo convencionais, os protocolos envolvidos e os problemas referentes à área. Nessa etapa será efetuada uma extensa revisão bibliográfica sobre o tema;
- Definir os requisitos para o método de transmissão utilizando SDN para que o controlador reconheça os protocolos estudados e instale corretamente regras nas tabelas de fluxo, assim como execute a agregação dos fluxos semelhantes; e
- Elaborar e especificar uma forma de transmissão de vídeos em uma rede SDN, a fim de construir uma abordagem formada por componentes da rede e *software* do

controlador. Essa etapa exige estudos sobre as Redes Definidas por *Software*.

Enquanto para a segunda parte do trabalho (ou implementação e experimentação da abordagem proposta) são definidas as etapas:

- Desenvolvimento e implementação do cenário proposto; e
- Análise experimental do método implementado, comparando-o com o método convencional de transmissão de vídeos.

Para a segunda parte do trabalho, o método empregado é a Pesquisa Aplicada. Esse método é utilizado com o objetivo de testar as capacidades da solução MMFA desenvolvida.

2 Vídeo Sob Demanda

Os conceitos envolvendo transmissão de vídeos, como definições e tecnologias envolvidas são introduzidas neste capítulo. Inicialmente, a Seção 2.1 contextualiza o surgimento dos serviços VoD através da evolução do entretenimento multimídia ao decorrer dos anos. Após um breve contexto sobre o surgimento desses serviços, a Seção 2.2 aborda as definições relacionadas ao VoD. Essas definições são necessárias para compreender o funcionamento desses sistemas e a forma nas quais são divididos de acordo com suas características. Com os conceitos necessários definidos, a Seção 2.3, detalha uma subcategoria citada na Seção 2.2, na qual é a base do estudo aqui descrito. Esse detalhamento descreve o funcionamento do sistema, protocolos utilizados para comunicação, infraestrutura da rede e caracterização do tráfego, conceitos importantes para a definição da problemática que é abordada.

A Seção 2.4 descreve os principais desafios e problemas de tráfego na rede envolvendo transmissão de vídeo. Essa seção é importante para compreender os pontos críticos desses sistemas. Com os problemas definidos, a Seção 2.5 utiliza os conceitos apresentados unindo-os aos problemas da área da transmissão de vídeo para definir a problemática e o escopo do estudo. Com isso, a Seção 2.6 descreve algumas alternativas para a diminuição dos problemas de tráfego na rede. Cada alternativa com uma infraestrutura e método de abordar a distribuição de forma diferente, com seus pontos positivos assim como pontos negativos. Por fim, a Seção 2.7 apresenta alguns trabalhos relacionados, nas quais abordam problemas de transferência de vídeos em Redes Definidas por *Software*.

2.1 Contextualização

Desde o início da era televisiva, o ser humano têm se aproximado cada vez mais das formas de entretenimento baseadas em vídeo e áudio. Iniciando com os aparelhos que projetavam apenas imagens em escala de cinza através de um tubo de raios catódicos, evoluindo para as imagens coloridas ainda utilizando os mesmos tubos (JUNIOR, 2016). Logo após, surgiram assinaturas mensais para canais fechados e a criação de aparelhos

que eram capazes de registrar o conteúdo que estava sendo transmitido na tela, como o *Video Home System* (VHS) (SHIRAISHI, 1985; GLINIS, 2015). Pouco depois, os VHS foram substituídos pelos DVD *players* com uma qualidade de imagem superior. Por fim, chegaram os *players* de Blu-Ray para continuarem o desenvolvimento do entretenimento a base de vídeo (ALECRIM, 2011).

Assim como os sistemas de televisores e reprodutores de vídeo evoluíam, paralelamente cresciam as capacidades e velocidade das conexões de Internet da população como um todo. Com esse crescimento, os serviços de *streaming* de vídeo tornaram-se cada vez mais populares e requisitados (JULURI; TAMARAPALLI; MEDHI, 2016). Outros serviços como o *Internet Protocol Television* (IPTV) foram criados com fim de transmitirem canais de TV pela Internet, como uma alternativa de canal de comunicação, além do convencional na qual era anteriormente a única opção das emissoras de TV (MYLER, 2007).

Pelo fato da grande audiência, esses serviços tornaram-se responsáveis por uma considerável parte do tráfego de dados nas redes, com cerca de 30% nos EUA em 2012 (ADHIKARI et al., 2012) e podendo chegar a até 80% em 2019 (CISCO, 2015). Com essas quantidades de dados trafegando nas redes é evidente a sobrecarga nos enlaces de comunicação, pela quantidade de usuários e pela necessidade desses serviços de utilizarem muitos recursos da rede (*e.g.,* Largura de Banda). Deste modo, a evolução do entretenimento toma novas proporções e meios de transmissão.

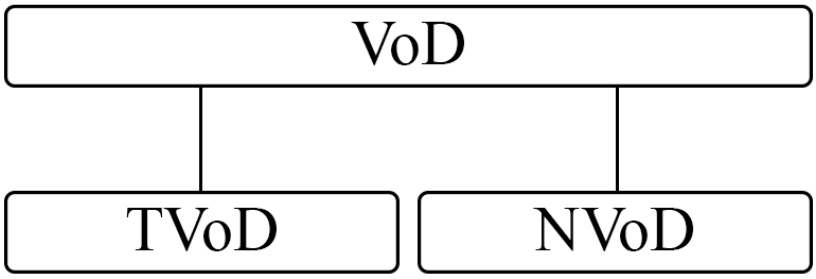
2.2 Definições

Os serviços de transmissão de vídeo são comumente chamados de *Video On Demand* (VoD), na qual o usuário escolhe um conteúdo a ser assistido e o reproduz imediatamente em seu *player* de vídeo, ou até mesmo realiza o *download* do arquivo e o reproduz posteriormente (ROUSE, 2007). Alguns serviços desse tipo são oferecidos por grandes empresas, como o *YouTube*, que disponibiliza o conteúdo de forma gratuita com bilhões de vídeos transmitidos diariamente (HOSSFELD et al., 2013).

É importante definir que existem algumas subcategorias dentro do conceito envolvendo VoD. Essas subcategorias recebem nomenclaturas diferentes dependendo do autor em questão, porém diversos autores utilizam *True Video On Demand* (TVoD) e

Near Video On Demand (NVoD) (YU, 2016; NAYFEH; SARHAN, 2016; ZHANG; HASSANEIN, 2010; MA; SHIN, 2002). Essa subdivisão é ilustrada na Figura 2.1.

Figura 2.1: Subcategorias de VoD



Fonte: Elaborado pelo Autor (2017)

Uma breve descrição das subcategorias listadas na Figura 2.1:

- **TVoD**: Essa classificação do VoD se destaca por possibilitar ao usuário uma maior interação com o vídeo, como por exemplo congelar a imagem, retroceder e avançar, por conta dessas características também pode ser chamada de *Interactive Video On Demand* (IVoD) (DHAGE; PATIL; MESHRAM, 2014). Usualmente, o método de transporte utilizado é o *Unicast*, justamente para possibilitar as opções de interação e também pelo fato do usuário conseguir escolher qual conteúdo assistir a qualquer hora (GALLO, 2009). Outra implicação por ser *Unicast* é a entrega dessincronizada para cada usuário, possibilitando a cada usuário assistir um conteúdo diferente ou o mesmo conteúdo em tempos diferentes. Além das possibilidades já citadas, o TVoD conta com um *buffer* que garante a disponibilidade do conteúdo para evitar travamentos e outros problemas na exibição.

- **NVoD**: Os sistemas NVoD diferem dos TVoD pelas limitações quanto à interação do usuário. Esse tipo de serviço se assemelha mais com uma transmissão de TV a cabo, na qual vários usuários recebem o mesmo conteúdo ao mesmo tempo em que ele está sendo transmitido. Normalmente o tipo de conexão utilizado é a *Multicast*, por existirem diversos usuários consumindo o mesmo conteúdo simultaneamente. Por ser *Multicast*, o conteúdo é transmitido de forma sincronizada para todos os usuários. Essa categoria se assemelha ao *Live Streaming*[1](WEN; LONGSHE; QIANG, 2006).

[1]O *Live Streaming* consiste em fluxos contínuos de áudio e vídeo em tempo real. O receptor recebe dados do transmissor de forma contínua e assiste quase imediatamente o que está sendo enviado pelo transmissor naquele momento (WEN; LONGSHE; QIANG, 2006).

Segundo Nayfeh e Sarhan (2013), é difícil alcançar o nível de requisitos de recursos de rede exigidos por um sistema TVoD, e um sistema NVoD automaticamente converge em um sistema TVoD quando esses recursos são disponibilizados. Por esse fato, o estudo aqui apresentado foca em sistemas NVoD, reduzindo a quantidade de recursos de rede exigidos, e que, consequentemente, facilitará essa convergência.

Para realizar a transmissão dos conteúdos nos sistemas VoD, existem diversos modelos que podem ou não ser baseadas em TCP/IP. Porém, o foco deste trabalho está em formas de entrega que são baseadas em tecnologias do modelo TCP/IP, como:

- ***Unicast:*** Em comunicações do tipo *Unicast*, existem apenas dois pontos comunicantes (HARVEY, 1999). Isto é: uma conexão exclusiva Um Para Um na qual um ponto irá enviar e outro ponto irá receber.

- ***Multicast:*** Em comunicações *multicast*, dados são enviados de um ponto para um considerável número de receptores se comparado ao *Unicast*, nos quais existe apenas um receptor (BIERSACK, 2005). Em outras palavras, é uma conexão Um Para Vários com o objetivo de poupar largura de banda evitando múltiplas cópias do mesmo conteúdo trafegando simultaneamente (HARVEY, 1999; GALLO, 2009).

- ***Broadcast:*** A comunicação *broadcast* é uma conexão Um Para Todos, na qual existe um remetente que envia informações e que todos os receptores conectados as recebem (SANTOS, 2016).

Como pode ser observado, existem diversas nomenclaturas para definir as subcategorias de VoD na literatura, que em alguns casos recebem as nomenclaturas voltadas ao meio comercial, não relevantes para o estudo. Devido ao foco deste trabalho estar relacionado a NVoD, torna-se necessário um estudo mais detalhado dos aspectos inerentes desta subcategoria do VoD.

2.3 *Near Video On Demand* (NVoD)

Conforme Yoshihisa e Nishio (2013), é importante para os clientes dos serviços VoD reproduzirem o conteúdo do início ao fim sem interrupções. Como em alguns serviços de redes via satélite estudados por Asorey-Cacheda et al. (2007) na qual a abordagem NVoD foi

utilizada. De acordo com Chen e Huang (2008), o método de transmissão NVoD também é útil para atender consideráveis quantidades de usuários simultâneos.

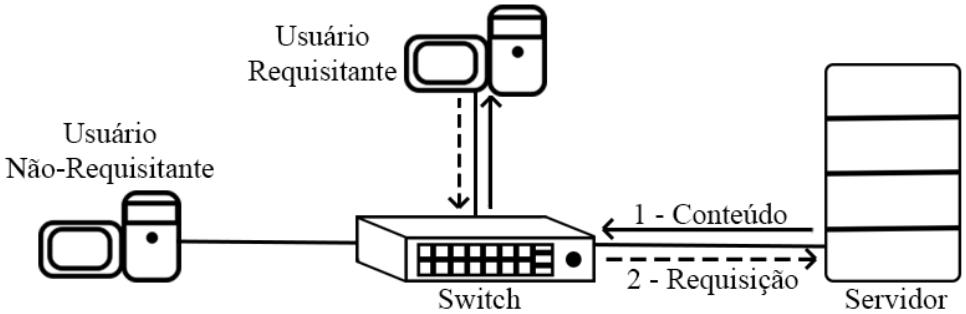
Esse atendimento à vários usuários simultaneamente com o mesmo fluxo de dados, caracterizando o serviço NVoD, foi proposto para resolver os problemas de limitação de acesso à conteúdos populares com um alto nível de requisições. Problemas nas quais eram pertinentes aos serviços TVoD que oferecem um canal de entrega para cada usuário (KWAK; YOO; HONG, 2010).

Para detalhar e caracterizar um sistema NVoD, é necessário o entendimento de como é feita a transferência de conteúdo até o usuário final, os protocolos que fazem parte da comunicação e a infraestrutura de rede. Juntamente ao conhecimento do funcionamento desses sistemas, faz-se necessário conhecer alguns dos desafios e problemas enfrentados, assim como algumas alternativas de formas de transmissão nas quais possuem como objetivo resolver tais problemas.

2.3.1 Funcionamento

A redução da quantidade de recursos de rede utilizados é dada pelo fato do vídeo ser dividido em múltiplos segmentos na qual são transmitidos em um conjunto de canais de transmissão usando *multicast* (YU, 2016). Essa divisão em segmentos é denominada *Harmonic Broadcasting* (JUHN; TSENG, 1997). O equipamento de transmissão envia o mesmo fluxo de dados entregando o conteúdo para todos os usuários requisitantes na área de transmissão (KWAK; YOO; HONG, 2010). No entanto, esses usuários precisam esperar até receberem o primeiro pacote da transmissão para começar a reproduzir o conteúdo (YOSHIHISA; NISHIO, 2013). A Figura 2.2 ilustra a movimentação de mensagens na infraestrutura.

Figura 2.2: Movimentação de mensagens

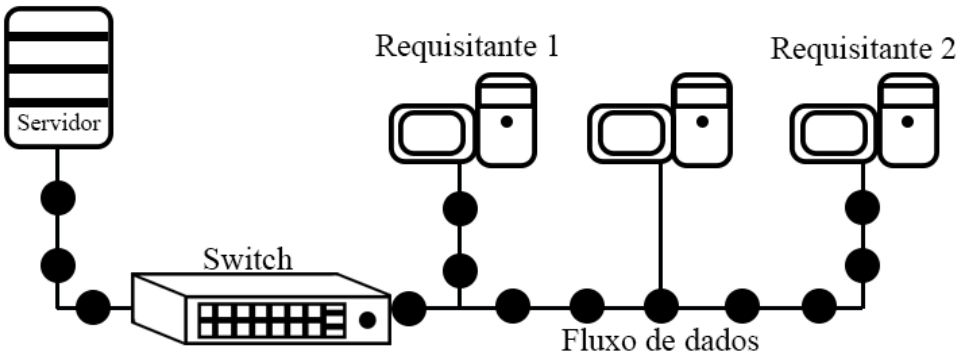


Fonte: Elaborado pelo Autor (2017)

Ilustrado na Figura 2.2, o usuário emite ao servidor uma requisição (linha tracejada) informando o conteúdo de interesse. Após receber a requisição, o servidor responde o usuário enviando o conteúdo em questão (linha contínua). Como o outro usuário não requisitou um conteúdo, não há troca de mensagens entre o mesmo e o servidor.

Outros métodos podem ser empregados para reduzir a carga na rede, como o método *batching* (ANDERSON, 1993). Esse método consiste em atrasar as requisições para diferentes conteúdos por um certo período de tempo para que mais requisições do mesmo conteúdo sejam efetuadas (NAYFEH; SARHAN, 2013; ALMEROTH; AMMAR, 1996; DAN; SITARAM; SHAHABUDDIN, 1996). Com isso, todas as requisições dos mesmos conteúdos, que chegaram durante esse período de atraso, são transmitidas utilizando o mesmo fluxo *multicast*, demonstrado na Figura 2.3.

Figura 2.3: Fluxo de dados *Multicast*



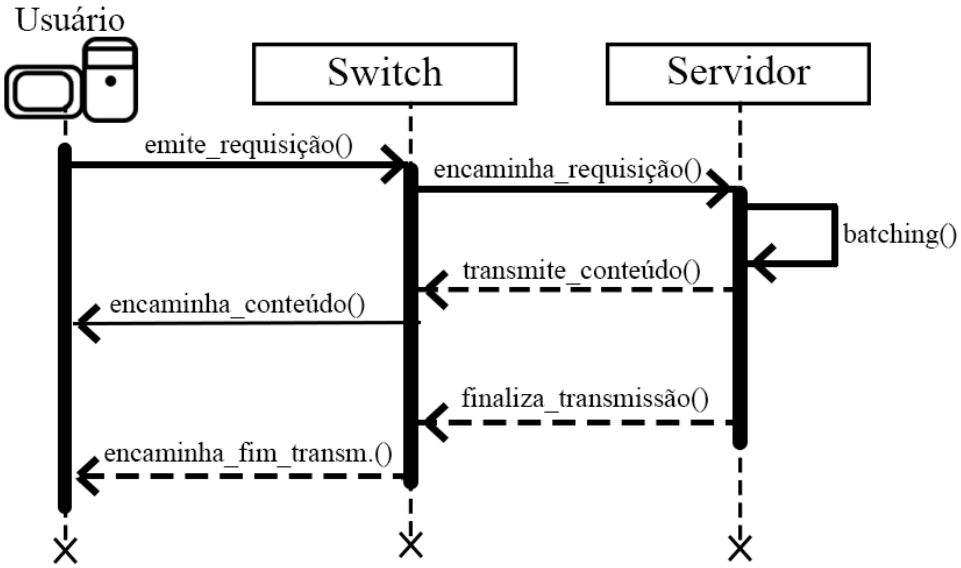
Fonte: Elaborado pelo Autor (2017)

Como ilustrado na Figura 2.3, o servidor provedor de conteúdo está conectado à três usuários, porém apenas dois fizeram a requisição de determinado conteúdo. Com isso, os dados serão entregues para os usuários requisitantes simultaneamente através de um fluxo *multicast*.

Com os devidos eventos relacionados à transmissão de conteúdo em vídeo definidos, é possível detalhar a sequência na qual os mesmos acontecem. O detalhamento é representado em um diagrama de sequência, ilustrado na Figura 2.4.

Segundo Gallo (2009), a transmissão *multicast* faz-se útil pois caso fossem criadas conexões exclusivas ponto a ponto para cada cliente, o número de conexões cresce linearmente com a quantidade de usuários ativos assim como a banda necessária à transmissão. Essas conexões exclusivas sobrecarregam o servidor de origem do conteúdo assim como a rede como um todo, por conta das múltiplas cópias do mesmo conteúdo sendo

Figura 2.4: Diagrama de Sequência da requisição de conteúdo



Fonte: Elaborado pelo Autor (2017)

transmitidas (YOSHIHISA; NISHIO, 2013; GALLO, 2009). Sendo assim, com a conexão *multicast* o sistema torna-se escalável possibilitando um número maior de usuários receberem conteúdo, por conta dos fluxos compartilhados com usuários que requisitaram o mesmo conteúdo, assim como o custo do sistema é minimizado (LEE, 2002).

Além das formas de transmissão, métodos para otimizar o envio e métodos para segmentar o conteúdo, também são necessários os protocolos nas quais padronizam a comunicação e a tornam viável entre diferentes pontos da malha da rede, sejam esses pontos servidores, usuários ou equipamentos que fazem o intermédio entre os outros dois, como *switches* e roteadores.

2.3.2 Protocolos

Assim como qualquer forma de comunicação entre máquinas em uma rede, as transferências de arquivos (incluindo VoD) precisam fazer o uso de protocolos para padronizar as suas comunicações para ambos, origem e destino, trabalharem com o conteúdo de forma adequada. Para isso, existem alguns protocolos que foram criados para transferências em sistemas tando TVoD quanto sistemas NVoD. Porém, também podem ser utilizados para outros tipos de transferências de dados. Além disso, é possível a criação de um novo protocolo focado em determinada aplicação, mas essa criação não se enquadra no foco deste trabalho. Nesse contexto, destacam-se alguns protocolos de licença aberta:

- *Real-time Transport Protocol* **(RTP)** / *Real-time Transport Control Protocol* **(RTCP)**: Normalmente utilizados sob *User Datagram Protocol* (UDP), os protocolos RTP e RTCP podem ser considerados complementos um do outro. Esses protocolos são utilizados para assegurar o controle e a sincronia de pacotes que são transmitidos em tempo real. A sincronia fica a cargo do RTP, que funciona através de *timestamps*, ou marcações de data e hora, que faz a sincronia do vídeo com o áudio (SCHULZRINNE et al., 1996). Enquanto o controle da seção é feito pelo RTCP, na qual transmite pacotes que são enviados periodicamente por ambas as pontas, cliente e servidor, contendo informações estatísticas, como a quantidade de pacotes enviados e quantidade de pacotes perdidos, dentre outros. Esses pacotes de informações podem ser de tipos específicos (SCHULZRINNE et al., 1996):

  - **Pacote *Sender Report*:** Relatório do Emissor, para estatísticas de transmissão e recepção sobre participantes que são emissores ativos;
  - **Pacote *Receiver Report*:** Relatório do Emissor, para estatísticas de transmissão e recepção sobre participantes que não são emissores ativos;
  - **Pacote *Source Description*:** Ítens de descrição da fonte, como identificadores; e
  - **Pacote *Bye*:** Sinaliza a finalização da transmissão.

  Existem ainda alguns outros tipos de pacotes enviados pelo protocolo RTCP para fins de controle da transmissão, porém menos relevantes e pouco pertinentes ao estudo.

- *Real Time Streaming Protocol* **(RTSP)**: Descrito em Schulzrinne, Rao e Lanphier (1998), o protocolo RTSP tem o papel de estabelecer e manter a sessão (JULURI; TAMARAPALLI; MEDHI, 2016). Seu funcionamento é baseado na transmissão de segmentos dos dados, ou o dado particionado, em vários pacotes com tamanhos dependendo da banda disponível no enlace de comunicação entre cliente e servidor. Porém, é baseado no protocolo RTP para seleção de canais de transmissão (*Transmission Control Protocol* (TCP), UDP) e também para o controle da sessão, assim como a garantia de entrega do conteúdo em tempo real (JULURI; TAMARAPALLI; MEDHI, 2016). Essa escolha de canal de transmissão é devido à possibilidade do RTSP transmitir fluxos de dados a serem armazenados, além do fluxo de dados em tempo real, com isso pode ser utilizado o TCP, que segundo Majumda et

al. (2002) não é indicado para transferências em tempo real por ser um protocolo assíncrono, diferente do UDP.

- *Hypertext Transfer Protocol* **(HTTP):** De acordo com Juluri, Tamarapalli e Medhi (2016), o protocolo HTTP se enquadra nas técnicas de *download* de fluxo progressivo, ou *Progressive Download Streaming Techniques*, na qual é feito o *download* do arquivo de vídeo enquanto o mesmo é reproduzido no receptor. Essa transferência é feita através do protocolo HTTP sobre TCP, por isso garante a entrega dos dados. Porém, como o conteúdo é transferido o mais rápido possível, pode ocorrer um desperdício de banda, caso o usuário desista antes de assistir completamente o vídeo (HOSSFELD et al., 2013). O serviço mais popular que faz o uso do protocolo HTTP para suas transferências é o *YouTube*, na qual suporta *Flash Player* e HTML5 para a reprodução enquanto armazena o conteúdo em uma pasta temporária (WAMSER et al., 2010; HOSSFELD et al., 2013).

- *Internet Group Management Protocol* **(IGMP):** Segundo Cain et al. (2002), o protocolo IGMP tem a função de controlar os membros de um grupo *multicast*, assim como controlar a entrada e saída de *hosts* desse grupo. De acordo com Yahav e Rabinovitz (2002), uma vez que um pacote chega ao dispositivo, tal dispositivo precisa executar a decisão de encaminhamento, caso seja a primeira vez que determinado pacote é analisado, é iniciada a construção do grupo. Caso algum *host* não requisite a entrada para o grupo, ou seja, receber os pacotes que chegarem ao dispositivo, os pacotes serão descartados pelo roteador.

- *Stream Control Transmission Protocol* **(SCTP):** Relativamente novo, o protocolo SCTP foi criado para ser adequado para transmissões, semelhante ao TCP/IP, devido ao julgamento de que outros protocolos são inadequados para a tarefa de transmissão (PFUTZENREUTER; FRIEDRICH, 2007; MAJUMDA et al., 2002). Sendo também Cliente-Servidor como o TCP, o SCTP possui alguns recursos adicionais, como (STEWART et al., 2000):

  - Confirmação e ordenação de mensagens indivisíveis, diferente dos octetos presentes no protocolo TCP;
  - Múltiplos fluxos de transmissão, com isso a perda de uma mensagem afeta apenas o fluxo da mesma;

– Capacidade de entregar mensagens em ordem ou seguindo prioridades; e

– Capacidade de dispensar a garantia de entrega, podendo assim ser utilizado para transmissões multimídia em tempo real.

Com essas características, o protocolo SCTP apresenta-se bastante flexível quanto às suas aplicações, porém por ser uma alternativa ao TCP, que por sua vez é bem estabelecido, não é frequentemente utilizado.

Com os protocolos definidos, a Tabela 2.1 compara-os de acordo com algumas características: Capacidade de transferir conteúdo, capacidade de controlar a sessão, aptidão para transferências em UDP e TCP, assim como posição no modelo OSI[2].

Tabela 2.1: Protocolos envolvidos em transferência de conteúdo

Protocolo	**Transferência**	**Controle**	**UDP**	**TCP**	**Camada**
RTP	Sim	Não	Sim	Não	Aplicação
RTCP	Não	Sim	Sim	Não	Aplicação
RTSP	Sim	Não	Sim	Sim	Aplicação
HTTP	Sim	Sim	Não	Sim	Aplicação
SCTP	Sim	Sim	-	-	Transporte
IGMP	Sim	Sim	-	-	Rede

Fonte: Elaborado pelo Autor (2017)

Como pode ser observado na Tabela 2.1, existem protocolos os quais são focados na transferência do conteúdo, como o RTP, e protocolos focados no controle da sessão, como o RTCP. Assim como protocolos qualificados para transferência e controle, como o HTTP. Também é possível inferir a existência de protocolos que podem transferir conteúdo em conexões TCP, e ou, UDP. Porém, como o foco são sistemas NVoD, os protocolos aptos para UDP são mais indicados, pela caracterização sequencial do tráfego (descrito na Subseção 2.3.4). A escolha dos parâmetros de comparação se deu pelas características principais dos protocolos para transferência de conteúdo, que é a capacidade em si de transferir o conteúdo, capacidade de controlar a seção, e operabilidade com o protocolo UDP e TCP, referente à caracterização do tráfego.

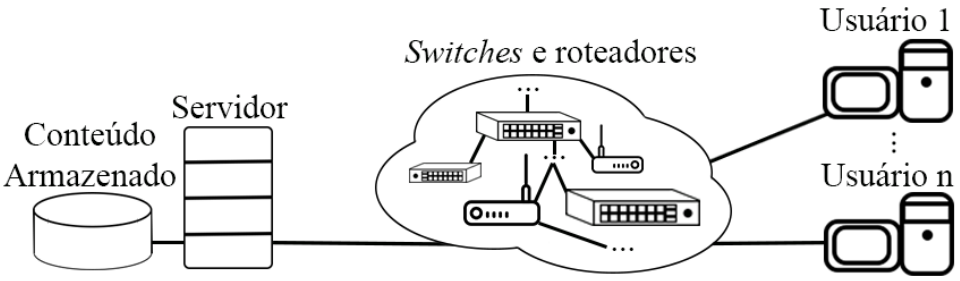
[2]O modelo OSI consiste em 7 camadas, sendo elas: Física (1), Enlace (2), Rede (3), Transporte (4), Sessão (5), Apresentação (6) e Aplicação (7) (ZIMMERMANN, 1980)

Em contrapartida aos protocolos abertos, algumas empresas privadas utilizam protocolos proprietários para a comunicação, como a *Netflix*[3], *Amazon Prime*[4], *NET Now*[5] dentre outras, porém esses protocolos de licença fechada não são o foco deste trabalho. Além dos protocolos de comunicação, os sistemas VoD necessitam de uma infraestrutura na qual essa comunicação ocorre.

2.3.3 Infraestrutura da rede

Como descreve Yoshihisa e Nishio (2013), uma infraestrutura necessária para suportar um sistema NVoD deve contar com um servidor que irá executar a transmissão do conteúdo, equipamentos para transporte, usuários que fizeram a requisição e o próprio conteúdo armazenado. Um exemplo dessa infraestrutura é ilustrada na Figura 2.5.

Figura 2.5: Esquema de um sistema NVoD



Fonte: Adaptado de Yoshihisa e Nishio (2013)

Esse transporte pode ser realizado via meios guiados ou meios sem fios. Caso seja a segunda opção, existe a Área de Transmissão, na qual representa o alcance do equipamento. Caso contrário, esse transporte é feito através de cabeamento que por sua vez será conectado ao usuário final através de equipamentos como *switches* e roteadores. Porém, no caminho percorrido entre a origem da transmissão e o usuário final existem diversos *switches* e roteadores que compõem a *Internet*. Esses equipamentos normalmente são heterogêneos, ou seja, são diferentes um do outro em quesitos de configuração, funcionamento, dentre outros. A configuração desses equipamentos é efetuada pelos seus proprietários, sendo assim inconcebível a configuração adequada e otimizada para cada um desses equipamentos, visando a aplicação em questão.

Além da dificuldade na configuração dos equipamentos dispostos entre origem e

[3] `https://www.netflix.com/`
[4] `https://www.primevideo.com/`
[5] `https://webportal.nowonline.com.br/`

destino, Gallo (2009) descreve as dificuldades na implantação global de conexões *multicast* em serviços *Internet*. Os provedores de serviços *Internet*, tipicamente, não disponibilizam roteamento *multicast* devido à falta de eficiência do modelo em funcionalidades para fins comerciais, como gerenciamento de grupo, na qual inclui o controle de acesso para possíveis transmissores e receptores dos conteúdos nessas conexões.

2.3.4 Caracterização do tráfego

Originalmente, a Internet foi projetada para dados estáticos como textos, ilustrações e *links* (VALE; COSTA; JR, 2001). Com isso, o *streaming* de vídeo nesse meio teve que ser adaptado, por exemplo, a continuidade da transmissão é de extrema importância. Um pacote de conteúdo que chega atrasado não é mais útil e pode afetar a reprodução (*e.g.,* um *frame* de uma cena anterior sendo reproduzido em uma cena subsequente). A mesma lógica pode ser aplicada aos pacotes de conteúdo que chegam adiantados ao tempo que deveriam ser reproduzidos (JAIN, 2011). Desta forma, em sistemas NVoD nas quais não possuem *Frame Buffer*[6] nativo, caso um pacote seja entregue em um tempo que não condiz com sua reprodução, é preferível seu descarte à sua reprodução, deixando o conteúdo em vídeo com uma lacuna em branco.

A característica de tráfego contínuo apresenta-se como um desafio, dentre vários outros existentes, na área de transmissão de conteúdos em vídeo. Esses desafios por sua vez, podem ser gerados através de algumas limitações presentes tanto na infraestrutura quanto nas implementações e nos protocolos desses sistemas de transmissão.

2.4 Problemas e desafios em NVoD

Embora já estabelecidos, os sistemas NVoD apresentam problemas e desafios. Alguns desses problemas são decorrentes da evolução contínua do conteúdo em vídeo, como qualidade e resolução superior de imagem, que afeta diretamente a largura de banda. Assim como o número crescente de usuários, que afeta diretamente no número de conexões necessárias para a distribuição do conteúdo. Esses problemas, ou desafios, podem ser classificados em algumas subáreas:

[6] *Frame Buffer* consiste em um sistema de armazenamento especializado em quadros de imagem. É usualmente utilizado para armazenar quadros que foram entregues antes de seu tempo de reprodução.

- **Topologia/Infraestrutura:** A infraestrutura utilizada para um sistema de transmissão de vídeos deve possuir os recursos de rede necessários para disponibilizar o serviço. Dentre esses recursos necessários, pode-se citar a gerência do conteúdo, gerência de usuários e a capacidade de transmissão da rede. Além disso, Molisch et al. (2014) descreve sobre os problemas enfrentados nas transmissões de vídeo em arquiteturas móveis, com o considerável crescimento da quantidade de dispositivos como *smartphones* e *tablets*. Esse crescimento faz com que mais tráfego seja gerado para transmissões de vídeo em arquiteturas sem fio.

  Outro desafio existente nas infraestruturas da rede são os equipamentos heterogêneos, citados na Subseção 2.3.3. Por exemplo, a existência de *switches* com suporte ao protocolo *OpenFlow*[7], enquanto outros não possuem tal suporte.

- **Usuários Simultâneos:** No trabalho de Bethanabhotla, Caire e Neely (2015), também são descritos desafios em transmissões com múltiplos usuários, tendo como exemplo a chegada de requisições a qualquer momento após a disponibilidade do conteúdo, assim como requisições para conteúdos diferentes. Essas requisições em tempos diferentes causam em certos momentos, necessidade de mesclagem de fluxos e utilização indevida da largura de banda do usuário, assim como a necessidade de duas ou mais requisições simultâneas. Alguns desses problemas também são citados por Eager, Vernon e Zahorjan (2000), Hua e Sheu (1997).

- **Largura de Banda:** Quanto aos problemas relacionados à largura de banda, Juluri, Tamarapalli e Medhi (2016) descrevem o crescente uso de banda para a transferência de vídeos, principalmente para conteúdos em alta definição. Com esse crescimento, faz-se necessária a otimização dos fluxos de conteúdo para um melhor aproveitamento dos recursos de rede disponíveis, mantendo uma boa qualidade de experiência para os usuários.

  Em seu trabalho, Georgopoulos et al. (2014) descreve alguns problemas para transmissão de vídeos em sistemas VoD em geral, nas quais podem ser estendidos para suas subcategorias. Exemplificando, a capacidade da rede precisa continuamente equiparar-se ao número crescente de usuários, assim como a crescente popularidade de conteúdos em alta definição, que representam uma maior quantidade de recursos de rede necessários para a transmissão. Em outras palavras: é necessário otimizar a

[7] O protocolo *OpenFlow* é utilizado em Redes Definidas por *Software*, como intermédio de comunicação entre os Planos de Dados e Controle.

transferência para múltiplos usuários simultaneamente, pois a quantidade de usuários requisitantes de conteúdo é crescente, assim como a largura de banda necessária para vídeos de resoluções cada vez maiores.

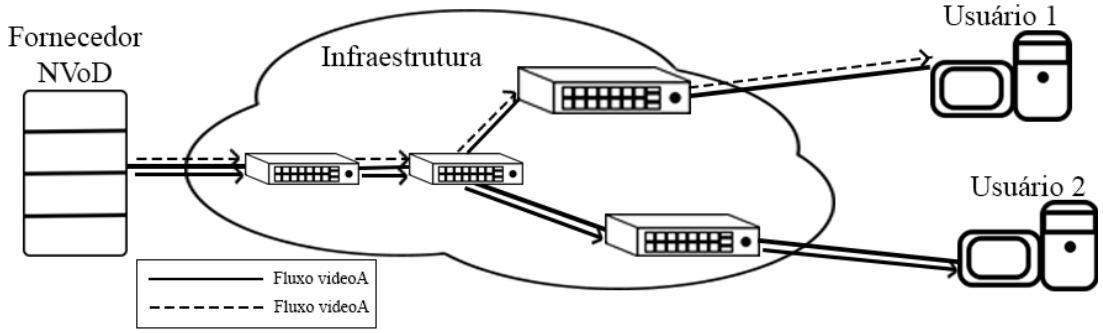
- **Segmentação:** Além dos problemas de infraestrutura, largura de banda e usuários simultâneos, Yu (2016) descreve sobre problemas relacionados o particionamento dos conteúdos em segmentos de vídeo, como citado na Seção 2.3.1 na qual o método *Harmonic Broadcasting* é o mais aceito. Porém, o *Harmonic Broadcasting* não é o único modelo, por isso, existem discussões sobre métodos de segmentação de conteúdo. O trabalho de Molisch et al. (2014) descreve um novo método para esse fim. O método descrito divide o arquivo de vídeo em sequências de segmentos de igual duração, na qual cada sequência contém uma quantidade diferente de bits codificados.

Existindo problemas em diversas áreas, naturalmente alguns possuem uma maior relação ao trabalho aqui apresentado, enquanto outros são pouco relacionados. Tendo em mente esses problemas, é possível definir a problemática, assim como o nível de relevância de cada problema para o estudo aqui descrito.

2.5 Definição do problema e escopo

Com os principais problemas em sistemas NVoD definidos, percebe-se que existe o problema do tráfego redundante, afetando a largura de banda dos enlaces. Isso é decorrente de situações como diversos usuários simultâneos. A Figura 2.6 ilustra o problema dos fluxos redundantes.

Figura 2.6: Fluxos redundantes



Fonte: Adaptado de Goya (2014)

Dessa maneira, um dos problemas que se tem nos sistemas NVoD, que é relacionado à problemática do estudo, é de como reduzir o tráfego redundante em um enlace de comunicação de uma rede, atendendo todos os usuários simultâneos. A Tabela 2.2 relaciona os problemas citados na Seção 2.4 com a problemática do estudo, descrevendo o nível de relação de cada problema com a problemática.

Tabela 2.2: Relação dos problemas com a problemática do estudo

Problema	**Nível de Relação**	**Descrição**
Topologia/Infraestrutura	Pouco Relacionado	A infraestrutura da rede é relacionada ao trabalho, contudo é dependente da infraestrutura física. Este trabalho parte do princípio que a infraestrutura física suporta o serviço, como um pré requisito.
Usuários Simultâneos	Muito Relacionado	Muitos usuários simultâneos implica em redundância de tráfego nos enlaces de comunicação da rede. Esse tráfego precisa ser tratado e otimizado para entregar o conteúdo para todos os usuários com os recursos de rede disponíveis.
Largura de Banda	Muito Relacionado	A existência de recursos de banda necessários para a transmissão de vídeos na rede contribui diretamente para a experiência do usuário, pois reduz a perda de quadros, na qual está diretamente ligada à característica do tráfego de sistemas NVoD, descrita na Subseção 2.3.4, que é contínuo e ininterrupto
Segmentação do conteúdo	Pouco Relacionado	A forma de segmentar o conteúdo não está diretamente ligado aos problemas de tráfego e redundância nos enlaces de comunicação da rede. Embora seja importante para a reprodução do conteúdo ao usuário, o problema da segmentação está pouco relacionado com a problemática do estudo.

Fonte: Elaborado pelo Autor (2017)

Existem também outros problemas como heterogenia da infraestrutura física da rede, que não é algo que se consiga mudar por ser diretamente dependente das operadoras. Com isso, diversas empresas necessitam migrar suas infraestruturas de rede para obter melhor desempenho. Essas situações são também aliadas à questões do próprio IPv4[8] (como a implementação global do *multicast*, citado na Subseção 2.3.3), que com a renovação de *hardware* para IPv6 pode se esperar que exista algum suporte e recursos

[8]O IPv4 tem sido o protocolo da camada de Rede desde o início da Internet. Porém, com o crescimento das redes mundiais o protocolo tem se deparado com diversos problemas, como o limite dos endereços IP, escalabilidade de roteamento e propriedades fim a fim (WU et al., 2013)

para o auxílio nesses problemas, como o controle das transmissões de início ao fim das transmissões *multicast*.

Definida a problemática do estudo, é necessário definir também o escopo na qual essa problemática será abordada. Por ser um estudo de escala reduzida, a abordagem aos problemas definidos será em uma *Local Area Networks* (LAN), ou rede local, justamente pelo controle necessário dos recursos de *hardware* do início ao fim.

Já foram desenvolvidas arquiteturas (ou infraestruturas) para a resolução dos desafios nas transmissões de conteúdo, sendo esses desafios principalmente a quantidade de tráfego gerado em diversos escopos como LAN e *Wide Area Networks* (WAN). Algumas dessas alternativas de arquiteturas são citadas na Seção 2.6, como as *Content Delivery Network* (CDN), redes *Peer-to-Peer* (P2P) e *Software Defined Network* (SDN).

2.6 Alternativas para diminuição de tráfego

Como relatado, algumas alternativas para a diminuição de tráfego em transferência de dados surgiram ao longo dos anos. Cada um dos modelos possui características que são melhor aplicadas em determinadas situações.

- **CDN:** As CDNs disponibilizam um melhor desempenho para as redes através da maximização da largura de banda e melhorando a acessibilidade. A melhora é devida a replicação de conteúdo em servidores em localidades próximas ao usuário (PATHAN; BUYYA, 2007).

- **P2P:** Chamados de *peers*, os usuários das redes P2P disponibilizam conteúdo com o intuito de compartilhar com outros usuários. Assim que um *peer* estiver interessado no conteúdo, é estabelecida a conexão com o *peer* portador do conteúdo e feito o *download* do mesmo. Com isso, aumenta a quantidade de *peers* disponibilizando o conteúdo (FOROUZAN; MOSHARRAF, 2013).

- **SDN:** De acordo com Rawat e Reddy (2016), o paradigma das redes definidas por *software* é tido como uma tecnologia capaz de gerenciar redes complexas de forma eficiente. O princípio desse paradigma é desassociar o Plano de Dados do Plano de Controle, na qual implementa o Plano de Controle em um *software* externo.

2.6.1 *Content Delivery Networks*

As CDNs são redes de distribuição de conteúdo, nas quais utilizam-se múltiplos servidores distribuídos geograficamente para fornecer um serviço para seus usuários. Alguns autores, como Shen et al. (2011), consideram as CDNs uma expansão do modelo Cliente-Servidor clássico.

A organização dos servidores nas CDNs é feita através de estruturas em árvore. Essa estrutura consiste basicamente do Servidor de Origem dos provedores de conteúdo, os nós de serviço centrais e os nós de serviço nas folhas da estrutura. Cada um desses nós folhas entrega o conteúdo para o usuário final (SHEN et al., 2011). As requisições dos usuários são dinamicamente encaminhadas para o servidor que se encontra geograficamente mais próximo ao requisitante (SU et al., 2009). Algumas técnicas para esse encaminhamento dinâmico podem ser aplicadas, como o *DNS Redirection* (ou Redirecionamento DNS)[9] (SHAIKH; TEWARI; AGRAWAL, 2001) e o *URL Rewriting* (ou Reescrever URL)[10] (KANGASHARJU; ROSS; ROBERTS, 2001).

As CDNs são eficientes para a distribuição de conteúdo, assim como para a diminuição da carga nos enlaces de comunicação da rede, porém exigem um alto valor de investimento para a construção e manutenção da infraestrutura necessária (SHEN et al., 2011). Neste contexto, a viabilidade da implementação de uma CDN é consideravelmente limitada, sendo viável para os principais provedores de serviços VoD, por exemplo *Netflix* e *YouTube*.

2.6.2 *Peer-to-Peer*

Arquiteturas de sistemas como a do programa SETI@Home [11] que busca inteligência extraterrestre através de ondas de rádio, que são processadas por computadores voluntários espalhados pelo mundo, são fundadas em arquiteturas P2P (FOSTER; IAMNITCHI, 2003). Essas arquiteturas são geralmente caracterizadas pelo compartilhamento de processamento, armazenamento e conteúdos sem precisar do intermédio de um servidor centralizado. Isso permite uma maior escalabilidade e capacidade de auto-organização

[9] *DNS Redirection consiste em traduzir uma requisição Web de um usuário para um servidor de domínio do provedor de conteúdo na qual está próximo ao usuário (SU et al., 2009)*

[10] *URL Rewriting* consiste em substituir URLs para conseguir uma melhor eficiência no roteamento, por exemplo um servidor de distribuição de conteúdo mais próximo ao usuário (GNAGY et al., 2006)

[11] `https://setiathome.berkeley.edu/`

(ANDROUTSELLIS; SPINELLIS, 2004).

Por possuírem nós, que são os próprios usuários e não um servidor centralizado, as arquiteturas P2P fazem proveito da velocidade de *upload* desses nós para uma potencial redução da carga de tráfego na rede (SHEN et al., 2011). Esses nós/usuários são chamados de *peers* e contribuem com os próprios recursos para suportar o tráfego na rede, isso possibilita crescimento e escalabilidade da infraestrutura de forma mais viável financeiramente, quando comparado a *data centers* (TAN; MASSOULIÉ, 2013), e por consequencia, até mesmo CDN.

2.6.3 *Software Defined Network*

Recentemente, SDN (ou Redes Definidas por *Software*) emergiu como uma arquitetura que possibilita ao administrador gerenciar os fluxos da rede através de abstrações de alto nível. Desta forma, esse paradigma auxilia na resolução de algumas limitações presentes nas infraestruturas de rede atuais (KREUTZ et al., 2015). Esse gerenciamento facilitado é possível devido à separação dos Planos de Controle e de Dados, com isso, o controle da rede tornou-se diretamente programável (CHIANG; LI, 2016). Esse módulo de controle que foi extraído do *hardware* e implementado externamente em *software* é denominado Controlador, enquanto o Plano de Dados fica a cargo do *hardware* de rede, como o *switch*.

A ideia por trás da criação das redes definidas por *software* foi facilitar a evolução das redes (NUNES et al., 2014). Essa evolução pode ser vista como uma preparação para o aumento anual considerável de tráfego nas quais as redes mundiais precisam lidar, como previso por Cisco (2015).

2.6.4 Comparação das alternativas

Cada uma das alternativas citadas para a resolução dos problemas em sistemas NVoD, ataca um ou mais problemas dos citados na Seção 2.4. A Tabela 2.3 lista as alternativas, abordando quais dos problemas (dos que são muito relacionados com o estudo) cada uma resolve, assim como o escopo das aplicações e a complexidade apresentada.

Como o escopo do estudo é um cenário LAN, para transmissão de vídeo em pequena escala, as CDNs são pouco viáveis, por serem grandes infraestruturas com servidores geograficamente dispersos. Já as redes P2P podem possuir qualquer distância geográfica

Tabela 2.3: Problemas abordados por alternativa

Alternativa	**Usuários simultâneos**	**Largura de banda**	**Escopo**	**Complexidade**
CDN	X	X	WAN	Média
P2P	X	X	WAN/LAN	Alta
SDN	X	X	WAN/LAN	Baixa

Fonte: Elaborado pelo Autor (2017)

entre os *peers*, por não serem servidores (ou *data centers*) construídos de maneira fixa. As CDNs podem ser consideradas de média complexidade pelo fato de existirem serviços que ofereçam suas estruturas prontas para utilização, sem a necessidade de construí-las. Um serviço desse gênero é o *Amazon Web Services*[12], que disponibiliza arquiteturas, dentre elas as *CDNs*.

Dentre as alternativas comparadas, constata-se que o paradigma de Redes Definidas por *Software* está mais intensamente relacionado a otimização de fluxos para comunicação. Podendo assim, auxiliar na distribuição de vídeos para diversos usuários os quais possuem fluxos de conteúdos similares, gerenciáveis através do controlador SDN. A complexidade da configuração do controlador SDN é relativamente baixa, quando comparando a configuração do mesmo serviço com as configurações necessárias para a criação de uma infraestrutura P2P com diversos *peers* conectados.

2.7 Abordagem do problema com SDN

Alguns trabalhos podem ser encontrados na literatura envolvendo transmissão de vídeo e SDN (GEORGOPOULOS et al., 2015; CHIANG; LI, 2016; TANG; HUA; WANG, 2014). Essas pesquisas possuem como objetivo melhorar a qualidade das transmissões, assim como a experiência do usuário. Na busca dessa melhora, uma das alternativas dos pesquisadores é abordar esse problema criando sistemas de *cache*, como *OpenCache* e *Extended SDN-Based Caching* (ESC). Esses sistemas armazenam informações nas quais podem ou não ser utilizadas para agilizar processos de transferência de conteúdo, como por exemplo a informações de qual a melhor rota entre servidor e usuário.

[12] `https://aws.amazon.com/`

2.7.1 OpenCache

Em Georgopoulos et al. (2015) é descrito um serviço de *cache* para sistemas VoD, chamado *OpenCache*. De acordo com os autores, esse serviço é transparente, flexível e altamente configurável possibilitando qualquer estratégia de *cache* na infraestrutura da rede. Para atingir seus objetivos, *OpenCache* utiliza SDN para prover um Plano de Controle no qual é possível instrumentar o *cache* e as funcionalidades de distribuição, com o objetivo de posicionar o conteúdo o mais próximo possível do usuário, semelhante a uma infraestrutura CDN. Essa estratégia permite a redução da carga nos enlaces através do sistema de *cache*, porém não leva em consideração a quantidade de usuários requisitando um mesmo conteúdo, a fim de evitar redundâncias.

2.7.2 *Extended SDN-Based Caching* (ESC)

No trabalho de Chiang e Li (2016), é proposta uma arquitetura de *cache* para sistemas VoD denominado *Extended SDN-Based Caching* (ESC). Segundo os autores, a arquitetura ESC decompõe a função na qual inspeciona o tráfego com o objetivo de tomar decisões relacionadas ao *cache*. A arquitetura é composta por três tipos de entidades de redes: *Switch*, Controlador de *Cache* e Nó de *Cache*. Nessa arquitetura, é possível de armazenar diferentes partes do conteúdo em diferentes Nós, que de acordo com os autores, aumenta tanto a capacidade quanto flexibilidade do sistema. Essa Possibilidade é dada pelo fato das decisões serem tomadas por um controlador central. Porém, assim como a estratégia OpenCache, não leva em consideração as redundâncias de conteúdo trafegando nos enlaces de comunicação da rede, apenas a largura de banda para a tranmissão.

2.7.3 Framework of SVC Video Multicast Application

Diferente dos trabalhos de Georgopoulos et al. (2015), Chiang e Li (2016), o trabalho de Tang, Hua e Wang (2014) descreve um método para implementar a transmissão de vídeos *Scalable Video Coding* (SVC)[13] em redes SDN. O objetivo do trabalho é adaptar a transferência de vídeos SVC de conexões *unicast* para conexões *multicast* através da criação de um *framework* denominado *Framework of SVC Video Multicast Application*.

[13]As aplicações SVC comprimem o vídeo em múltiplas camadas de fluxos, na qual a camada da base tem a funcionalidade de reconstruir as demais com a qualidade dependendo da quantidade de camadas de aprimoramento existentes.

Essa adaptação para conexões *Multicast* auxilia na gerência dos usuários simultâneos. Essa estratégia trata a questão dos usuários simultâneos através de transmissões *multicast*, reduzindo a largura de banda necessária, porém com um maior enfoque na compressão de vídeo, o que difere no escopo da abordagem, que é a transmissão pura.

2.7.4 Comparação das abordagens

Dadas as abordagens definidas através de trabalhos relacionados descritas na Seção 2.7, nas quais utilizaram tecnologias SDN, pode-se definir o problema na qual a abordagem se propõe a resolver. A Tabela 2.4 ilustra essa relação entre a abordagem e os problemas atacados, análise se possui o escopo da problemática e a complexidade da implementação.

Tabela 2.4: Problemas abordados por alternativa

Alternativa	**Usuários simultâneos**	**Largura de banda**	**Escopo**	**Complexidade**
OpenCache		X	Transmissão	Baixa
ESC		X	Transmissão	Baixa
SVC Video Multicast	X	X	Transmissão / Compressão	Baixa

Fonte: Elaborado pelo Autor (2017)

Como pode ser verificado, os trabalhos relacionados não abordam de forma integral os parâmetros descritos (Usuários simultâneos, Largura de banda, Escopo e Complexidade de implementação). A abordagem SVC Video Multicast aborda os problemas de usuários simultâneos e largura de banda, mas o escopo também está relacionado com a compressão do conteúdo, sendo um conteúdo específico e comprimido em formato SVC. Porém, a complexidade abordada é a mesmo para todos: baixa por ser a alternativa SDN.

Com as abordagens descritas na Tabela 2.4, torna-se interessante o desenvolvimento de uma abordagem na qual leva em consideração a entrega do conteúdo para todos os usuários simultâneos no sistema assim como o tráfego nos enlaces de comunicação, evitando redundâncias e com o escopo de transferência de dados em redes LAN. Portanto, o estudo aqui descrito foca em uma abordagem para distribuição de vídeo baseada em Redes Definidas por *Software*.

2.8 Considerações parciais

Neste capítulo são descritos os principais conceitos de transmissão de vídeos em sistemas VoD e suas subcategorias TVoD e NVoD, estabelecendo-se da parte conceitual, bem como a infraestrutura, funcionamento e protocolos desses sistemas. O foco do estudo são os sistemas NVoD, por ser um modelo mais dependente da otimização de fluxos por conta da característica do tráfego contínuo e ininterrupto. A otimização de fluxos pode ser bastante explorada pelo paradigma SDN, pela sua proposta na qual consiste em uma arquitetura orientada à fluxos de dados.

Quanto aos protocolos de transmissão em sistemas NVoD, apresentados na Subseção 2.3.2, o conjunto RTP e RTCP tem fortes indícios de ser capaz de suprir as exigências das transmissões de vídeo. Isso se dá por conta das capacidades tanto de transmissão quanto de controle da sessão, assim como o fluxo contínuo através do protocolo UDP da camada de transporte.

3 Redes Definidas por *Software*

Este capítulo tem como objetivo introduzir, definir e detalhar os conceitos envolvendo as Redes Definidas por *Software*, ou SDN. Inicialmente, é necessário o entendimento das chamadas Redes Convencionais, que em outras palavras, são as redes nas quais o paradigma SDN foi oriundo.

As Redes Convencionais foram por muito tempo suficientes para os mais diversos tipos de aplicações, incluindo a transmissão de vídeos. Porém, com o aumento de demanda e qualidade tanto de vídeo quanto de áudio nos conteúdos multimídia, tais redes tornaram-se insuficientes para a transmissão, apresentando problemas para a distribuição desses conteúdos. Como um meio de solucionar tais problemas, o paradigma SDN foi introduzido e tornou-se bastante visível dentro do contexto das redes de computadores, por possibilitar uma maior flexibilidade e controle dos dados trafegados na rede.

3.1 Redes de Computadores Convencionais

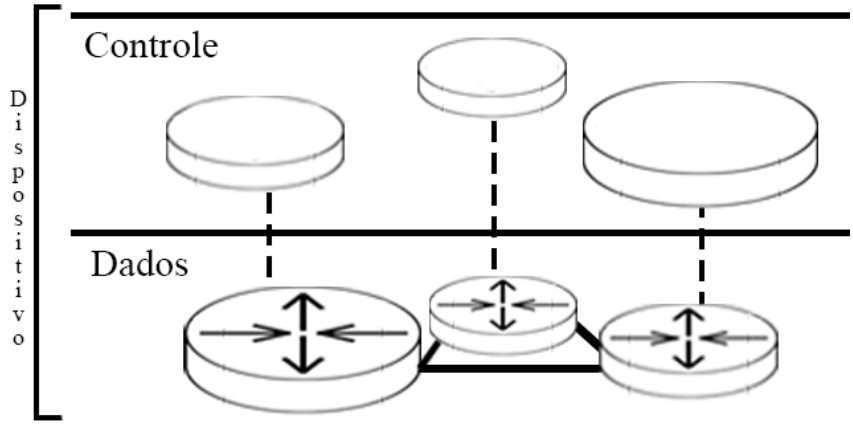
A informação nos meios digitais é representada e transmitida por meio de pacotes de dados. Essa transmissão é possibilitada por equipamentos de rede, como o *switches* e roteadores, nos quais podem ser definidos por dois planos distintos, como descrito por Kreutz et al. (2015):

- **Plano de Dados:** Corresponde ao equipamento da rede (*hardware*), na qual é responsável em encaminhar dados. Esse encaminhamento segue as regras armazenadas em uma Tabela de Fluxos, que por sua vez é construída pelo Plano de Controle; e
- **Plano de Controle:** Representa os protocolos (*software*) responsáveis pela lógica de roteamento e população das Tabelas de Fluxos utilizadas pelo Plano de Dados. Essa população segue a lógica implementada no protocolo utilizado.

Enquanto o Plano de Dados está presente na camada inferior, representando o *hardware*, os Planos de Controle e de Gerenciamento são componentes de *software* da camada superior. Essas camadas são interconectadas, na qual cada componente do Plano

de Controle gerencia e controla um componente do Plano de Dados. A Figura 3.1 ilustra a disposição dos planos nas redes em suas camadas.

Figura 3.1: Disposição dos Planos da Rede em camadas



Fonte: Adaptado de Kreutz et al. (2015)

Como ilustrado na Figura 3.1, cada componente do Plano de Dados é gerenciado por uma entidade do Plano de Controle. Porém, mesmo que o Plano de Controle seja um elemento de *software*, esse elemento está presente unido ao *hardware* do Plano de Dados. Em outras palavras, os Planos de Controle e Dados são implementados no equipamento da rede, como *switches* e roteadores. Com isso, a configuração dos equipamentos da rede (que é através do Plano de Controle) deve ser executada de forma individual para cada equipamento, por não existir um componente de controle centralizado. Essa configuração individual de cada equipamento dificulta o gerenciamento e configuração da rede, problemas que a abordagem SDN auxilia na resolução.

3.2 Definição de SDN

De acordo com Masoudi e Ghaffari (2016), o paradigma SDN foi projetado para simplificar e melhorar o gerenciamento de redes através da separação dos Planos de Dados e de Controle. Em uma definição mais detalhada, Kreutz et al. (2015) refere-se a SDN por uma arquitetura de rede na qual o encaminhamento de dados no Plano de Dados é gerenciado remotamente por um controlador externo. Com essa separação de planos, os equipamentos de rede como os *switches* tornaram-se apenas equipamentos encarregados de encaminhar o tráfego, no Plano de Dados. Enquanto a lógica de controle é implementada em um ou mais controladores externos sendo mais comum o uso de um controlador centralizado.

A separação dos planos atinge diretamente a configurabilidade do equipamento, pois um controlador centralizado no Plano de Controle é capaz de controlar e gerenciar diversos equipamentos no Plano de Dados, gerando uma maior flexibilidade na rede (LARA; KOLASANI; RAMAMURTHY, 2014). Essa capacidade difere das Redes Convencionais, nas quais cada dispositivo precisa ser configurado de forma individual, pelo fato dos planos serem unidos dentro do equipamento. Kreutz et al. (2015) define os quatro pilares para as Redes Definidas por *Software*:

- O plano de Controle e de Dados são separados. As funcionalidades de controle são executadas externamente aos equipamentos de rede, nas quais são apenas elementos para encaminhamento de pacotes (dados);
- As decisões de encaminhamento são baseadas em fluxos, diferentemente das decisões baseadas em destinos, como nas Redes Convencionais; Um fluxo consiste em um conjunto de pacotes preenchidos por valores, nas quais são avaliados através de critérios de compatibilidade ao serem comparados às regras presentes na Tabela de Fluxos; Caso exista uma regra para determinado fluxo, uma ação predeterminada é aplicada.
- A lógica de controle, ou Plano de Controle, é implementada em uma entidade externa ao equipamento de rede, denominada Controlador. Os controladores são detalhados na Subseção 3.3.2; e
- A rede é programada através de um *software* na qual oferece os recursos necessários para a programação dos dispositivos de forma centralizada. Esse *software*, denominado NOS (*Network Operating System*), interage com os dispositivos do Plano de Dados.

Com as principais características envolvendo as Redes Definidas por *Software* devidamente apresentadas, faz-se necessário detalhar a arquitetura desse paradigma, como seus componentes e protocolos. Para que com isso, seja possível identificar possíveis meios de administrar os fluxos de pacotes à fim de otimizar a transferência de conteúdo em vídeo.

Embora existam outros modelos para implementar as arquiteturas SDN, como a Cisco ACI (*Application Centric Infrastructure*)[1], esses modelos são proprietários, na

[1] `http://www.cisco.com/c/en_ca/solutions/software-defined-networking/implementation.html`

qual torna sua implementação inviável para este trabalho. Porém, existe o modelo OpenFlow[2], na qual é de licença aberta e também considerado o modelo padrão para SDN, que é o foco deste trabalho. Segundo Hu, Hao e Bao (2014), o principal protocolo para arquiteturas SDN é o chamado OpenFlow, que consiste em um protocolo orientado à fluxos, diferentemente dos protocolos de Redes Convencionais que são orientados a destino.

3.3 OpenFlow

O protocolo OpenFlow foi proposto para padronizar as comunicações entre o *hardware* (ou *switch*) e o controlador implementado em *software*, permitindo aplicações programarem as Tabelas de Fluxo de diferentes *switches* (MASOUDI; GHAFFARI, 2016; KREUTZ et al., 2015; HU; HAO; BAO, 2014). Com isso, as redes SDN nas quais utilizam o protocolo OpenFlow são denominadas Redes OpenFlow.

Segunda Lara, Kolasani e Ramamurthy (2014), originalmente o propósito do OpenFlow era de auxiliar pesquisadores a executarem testes em redes de produção[3]. Porém, as empresas adotaram o OpenFlow como uma estratégia para reduzir custos da rede e complexidade dos equipamentos, assim como aumentar as funcionalidades e características das redes com SDN. Embora as redes OpenFlow apresentem inovações e novas possibilidades, elas também apresentam desafios, como:

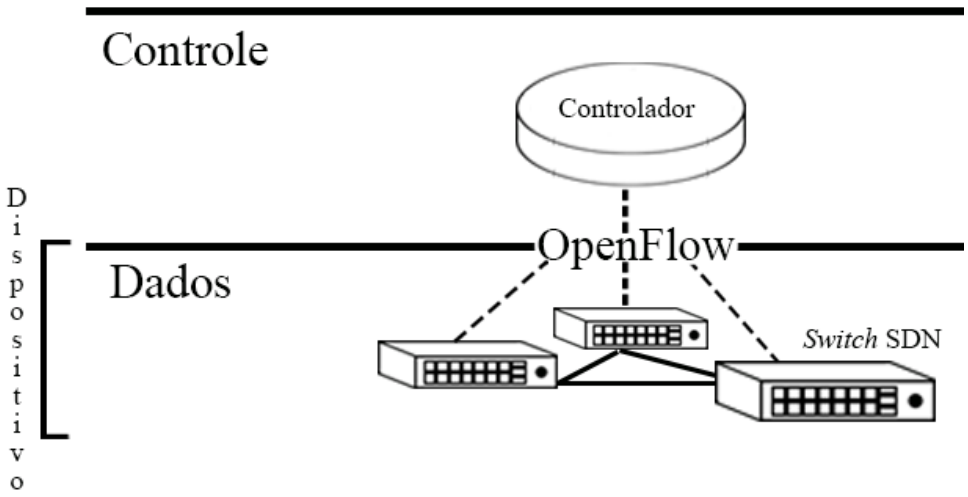
- Disponibilidade da rede, por em certos casos depender de um controlador centralizado. Vale a pena ressaltar que é possível existir mais de um controlador em uma rede OpenFlow;
- Segurança, por toda a informação sobre a rede estar contida em um ponto central da rede;
- Incompatibilidades que podem existir entre equipamentos da rede, como entre *switches* e controladores; e
- Por fim, a dificuldade da implementação de SDN em redes WAN, que apresentam heterogeneidade de dispositivos quanto ao suporte ao OpenFlow.

[2] `https://www.opennetworking.org/sdn-resources/openflow`
[3] As redes de produção representam as redes principais das empresas.

Criados para suportar o protocolo OpenFlow, os equipamentos de rede dessas arquiteturas são denominados *switches* OpenFlow (NGUYEN et al., 2016). Embora sejam otimizados para SDN, esses equipamentos podem comportar-se como equipamentos de uma Rede Convencional caso o OpenFlow seja desligado.

A arquitetura dessas redes OpenFlow SDN possuem um significante diferencial quando comparada às Redes Convencionais, das quais é oriunda. Esse diferencial se dá pela separação dos Planos de Dados e de Controle, como citado na Seção 3.2. Com essa separação, é necessário um meio de comunicação entre ambos os planos, o qual é realizado através do protocolo OpenFlow (LARA; KOLASANI; RAMAMURTHY, 2014). A Figura 3.2 ilustra a disposição dos planos em uma arquitetura SDN, assim como os componentes de cada plano.

Figura 3.2: Disposição dos planos em SDN



Fonte: Adaptado de Kreutz et al. (2015)

Como ilustrado na Figura 3.2, o intermédio entre os Planos de Dados e de Controle é feito através do protocolo OpenFlow, o que permite o controle de diversos equipamentos no Plano de Dados através de um controlador centralizado no Plano de Controle (HU; HAO; BAO, 2014). Embora a ideia geral se mantenha, o protocolo OpenFlow evoluiu ao decorrer dos anos através de versões. Essa evolução trouxe novos suportes e possibilidades. A Tabela 3.1 apresenta as principais mudanças em cada versão.

Como pode ser observado na Tabela 3.1, a primeira versão do protocolo não possuía suporte a mais de uma Tabela de Fluxos, ideia introduzida na segunda versão (1.1). O mesmo aconteceu com as Tabelas de Grupos, que foram introduzidas na versão 1.1. As Tabelas de Grupos consistem em agrupar operações comuns à vários fluxos,

Tabela 3.1: 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)

assim como esses vários fluxos são controlados para executarem ações coletivas dentro de um grupo (LARA; KOLASANI; RAMAMURTHY, 2014). Outra característica que foi atribuída ao OpenFlow foi o suporte ao IPv6, a partir da terceira versão, ou 1.2. Todas as características atribuídas ao protocolo ao decorrer das versões foram melhoradas na versão 1.3, como descrito na Tabela 3.1. Além dessas, existem versões posteriores a 1.3, como a 1.4 e 1.5, porém com inovações pouco expressivas quando comparadas as anteriores, como a adição de *bundles* na versão 1.4, que consiste em um agrupamento de mensagens atuando como uma única operação, na qual auxilia na sincronização dos *switches* da rede (REN; XU, 2014). A escolha da versão 1.3 para o estudo se deu pelo fato de ser uma versão completa, e que possui amplo suporte tanto dos equipamentos de redes quanto dos controladores SDN.

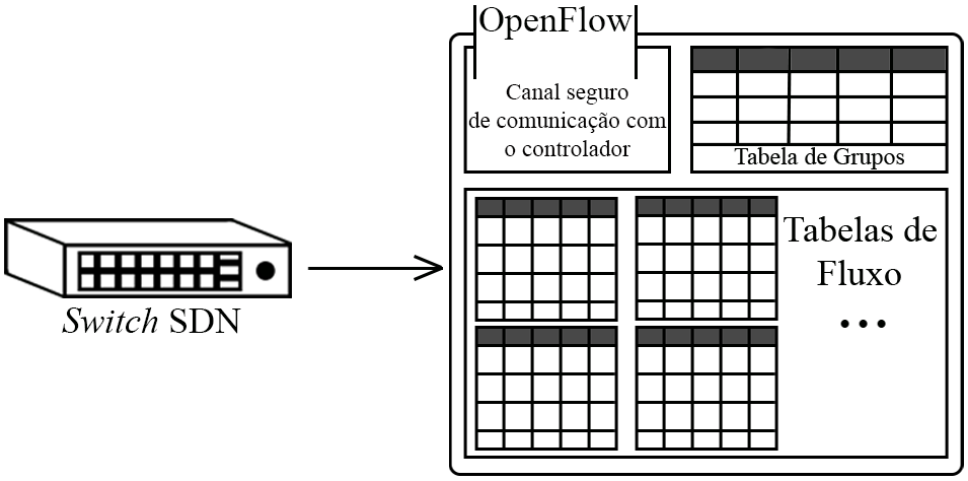
Como definido, o protocolo OpenFlow faz a comunicação entre os Planos de Dados e Plano de Controle nas redes OpenFlow. As Subseções 3.3.1 e 3.3.2 detalham esses planos no contexto de arquiteturas SDN.

3.3.1 Plano de Dados

Como o Plano de Controle foi extraído dos equipamentos (*e.g.*, *switch* e roteador) nas redes SDN, tais equipamentos tornaram-se simples elementos de encaminhar dados sem capacidade de tomar decisões de forma autônoma (MASOUDI; GHAFFARI, 2016). Essa extração culminou nos equipamentos responsáveis apenas pelo Plano de Dados, na qual

irá encaminhar os fluxos de acordo com as Tabelas de Fluxos. A Figura 3.3 ilustra a arquitetura interna de um dispositivo SDN com suporte ao OpenFlow 1.3.

Figura 3.3: Arquitetura interna de um dispositivo SDN



Fonte: Adaptado de Masoudi e Ghaffari (2016)

As Tabelas de Fluxo, por sua vez, consistem em três principais componentes nas quais caracterizam um fluxo (KREUTZ et al., 2015):

- **Regra de Compatibilidade:** Essa regra é composta de campos nas quais caracterizam um fluxo, como: a porta de origem; endereço MAC da origem; endereço MAC do destino; endereço IP da origem; endereço IP do destino, dentre outros;
- **Ação:** Consiste na ação a ser executada caso algum fluxo se encaixe na Regra de Compatibilidade. Essa ação pode ser do tipo Encaminhar para porta x; Encapsular e encaminhar para o Controlador; e Ignorar pacotes; e
- **Contadores:** Os contadores armazenam estatísticas sobre os fluxos que já passaram pelo equipamento.

De acordo com Nunes et al. (2014), caso um fluxo entrante não seja compatível com nenhuma regra presente na Tabela de Fluxos, esse fluxo é encaminhado para o Plano de Controle, na qual irá decidir qual ação será tomada. Após tomada a decisão uma nova regra é adicionada na Tabela de Fluxos, para caso um fluxo do mesmo tipo chegar ao dispositivo de encaminhamento, não seja necessário consultar novamente o Controlador. Caso contrário, se uma regra já estiver presente na Tabela de Fluxos, é apenas aplicada a ação descrita na regra para o fluxo.

3.3.2 Plano de Controle

O Plano de Controle nas Redes Definidas por *Software* pode ser gerenciado por um controlador único ou por vários controladores (HU; HAO; BAO, 2014). Porém, múltiplos controladores podem interferir uns nos outros, com isso, tipicamente, um controlador é escolhido como principal, enquanto os outros são utilizados como *backups* (MASOUDI; GHAFFARI, 2016). Esse Plano é considerado como uma parte essencial da rede, por ser responsável por todo o controle lógico de roteamento de fluxos, assim como o preenchimento das Tabelas de Fluxos do Plano de Dados (KREUTZ et al., 2015). Por ser centralizado, o controlador possui uma visão geral de toda a rede.

Segundo Masoudi e Ghaffari (2016), o projeto do controlador é a parte mais importante de uma arquitetura SDN, afetando diretamente o desempenho geral da rede. Com isso, o projeto do controlador deve levar em consideração alguns pontos críticos, como:

- **Consistência:** Os controladores e dispositivos no Plano de Dados precisam possuir as mesmas regras nas Tabelas de Fluxos para um roteamento estável.
- **Escalabilidade:** Devido ao aumento da popularidade das SDN, é necessário que um controlador suporte um crescimento de dispositivos nele conectados.
- **Disponibilidade:** Um controlador precisa ser capaz de recuperar-se de falhas como quedas de *links* e *switches*.

O preenchimento das Tabelas de Fluxos é executado através da aplicação das políticas de encaminhamento da rede em regras. O controlador conecta-se com os dispositivos no Plano de Dados através de um Canal Seguro (ilustrado na Figura 3.3), na qual é capaz de inserir, apagar e atualizar as regras. Assim como o controlador preenche a Tabela de Fluxos seguindo as políticas da rede, ele deve ser apto à tratar um fluxo entrante na qual ainda não exista uma regra na tabela. Após tratado, o fluxo é encaminhado e uma nova regra é inserida, ou atualizada (KIM; FEAMSTER, 2013).

Quanto aos controladores compatíveis com OpenFlow, diversos projetos e implementações foram desenvolvidas. A Tabela 3.2 apresenta e descreve três dos principais controladores SDN juntamente com suas características.

Tabela 3.2: Comparativo entre Controladores SDN

Controlador	Linguagem	Versão OpenFlow	Capacidades adicionais	Descrição
FloodLight	Java	1.0 e 1.3 (suporte experimental para 1.1, 1.2 e 1.4)	Inserção de fluxos de forma manual, criação do fluxo em todos os *switches* do caminho de roteamento do fluxo, descoberta de topologia e *learning switch*	Floodlight suporta diversos *switches* OpenFlow físicos e virtuais, assim como dispositivos tanto físicos quanto virtuais na mesma rede (MASOUDI; GHAFFARI, 2016).
NOX	C++	1.0	Descoberta de topologia e *learning switch*	Considerado como o primeiro controlador SDN, o NOX teve sua primeira versão lançada em 2009. Foi posteriormente subdividido em linhas de desenvolvimento, tornando-se o NOX *Classic*, enquanto são desenvolvidos o NOX e o POX, que suporta a linguagem Python (RAO, 2014).
OpenDayLight	Java	1.0 e 1.3	*Learning switch*, descoberta de topologia, inserção manual de fluxos, criação do fluxo em todos os *switches* do caminho de roteamento do fluxo	Promovido pela Linux Foundation, o projeto OpenDayLight tem como objetivo impulsionar o desenvolvimento do SDN e do *Network Function Virtualization*, que consiste em virtualizar classes inteira de funções de dispositivos de redes para a criação de serviços de comunicação (LINUXFOUNDATION, 2017).

Fonte: Elaborado pelo Autor (2017)

Existem outros controladores, como o Beacon, Maestro, NodeFlow, assim como uma versão escrita em Python para o controlador NOX, denominado POX. Porém, tais controladores não são tão populares quando comparados aos controladores descritos na Tabela 3.2.

A parte principal de uma arquitetura OpenFlow é a Camada de Controle, pois através dela é possível controlar os fluxos de pacotes trafegando na rede. Com isso, é possível administrar de forma flexível a entrega de conteúdo em vídeo para usuários simultâneos, otimizando o consumo de largura de banda nos enlaces de comunicação. Em outras palavras, a arquitetura OpenFlow é capaz de auxiliar na resolução dos principais problemas levantados para os sistemas de transmissão de vídeo em sistemas NVoD.

3.4 OpenFlow e NVoD

A área de transmissão de conteúdo multimídia tem sido bastante explorada por diversos trabalhos que utilizam a estratégia OpenFlow (CETINKAYA; SAYIT, 2016; DIORIO; TIMÓTEO, 2015; YU; WANG; HSU, 2015; EGILMEZ; CIVANLAR; TEKALP, 2013). Isso se dá pela possibilidade fornecida por essas arquiteturas de controlar de forma centralizada o tráfego de pacotes nas rede através das regras de fluxos com a visibilidade geral da rede pelo controlador, isso permite flexibilidade na gerência de tráfego, otimizando a utilização dos recursos da rede. Essa abordagem é oportuna para sistemas NVoD pelo fato desses conteúdos transmitidos requererem uma quantia significativa de recursos da

rede, principalmente se possuem alta definição.

O principal objetivo desses trabalhos é melhorar a QoS dos serviços NVoD através da otimização dos fluxos trafegados, reduzindo o consumo de largura de banda, e consequentemente a carga nos enlaces de comunicação da rede. A otimização dos fluxos pode ser abordada de diferentes formas:

- Em Diorio e Timóteo (2015) a transmissão multimídia é abordada utilizando o conhecimento centralizado do controlador SDN sobre a rede, processando os fluxos desses conteúdos diferentemente dos fluxos dos demais conteúdos. É proposta uma plataforma de processamento que fornece mecanismos para simplificar o encaminhamento dos fluxos de vídeo, levando em consideração as capacidades de banda da rede em questão.

- Em Cetinkaya e Sayit (2016) é proposta uma arquitetura na qual o controlador SDN utiliza o protocolo *Application Layer Traffic Optimization* (ALTO)[4] para otimizar o desempenho de sistemas de transmissão de vídeos em CDNs. Essa integração entre SDN e ALTO pode promover vantagens para as aplicações ao trocar pacotes com diferentes provedores de conteúdo.

- Em Yu, Wang e Hsu (2015), as propriedades das redes SDN, como controle centralizado e flexibilidade, são utilizadas para auxiliar o serviço provedor de conteúdo no desempenho de suas transmissões através da diferenciação de tráfego. É proposta uma abordagem de roteamento adaptável com suporte a QoS para melhorar a qualidade da transmissão de vídeos pelas redes SDN.

- Por fim, em Egilmez, Civanlar e Tekalp (2013) é proposta uma arquitetura baseada em OpenFlow com um esquema de roteamento com priorização de fluxos A abordagem é feita através de um *framework* executado sobre o controlador, na qual cumpre dinamicamente os requisitos de QoS entre o serviço provedor e o cliente.

A Tabela 3.3 compara os trabalhos relacionados nos quesitos envolvendo os problemas abordados na problemática do estudo, assim como os protocolos utilizados em cada abordagem. As métricas comparadas correspondem à otimização do uso de largura

[4]O protocolo ALTO executa entre o servidor e o cliente fornecendo informações de rede para as aplicações. Essas informações podem ser largura de banda disponível, custo do caminho e políticas de roteamento (ALIMI; YANG; PENNO, 2014).

de banda, escalabilidade do sistema quanto à quantidade de usuários simultâneos e a capacidade de agregação de fluxos para transmissões *multicast*, na qual possibilita tal escalabilidade de usuários.

Tabela 3.3: Comparativo entre os trabalhos relacionados

Referência	**Protocolo**	**Largura de Banda**	**Usuários Simultâneos**	**Agregação de Fluxos para *multicast***
Diorio e Timóteo (2015)	UDP, DSCP/IP, ECN/IP.	O trabalho aborda a otimização do uso dos recursos da rede, através da diferenciação dos fluxos.	Não é levado em consideração a quantidade de usuários simultâneos, apenas um único usuário.	Como é abordado o problema apenas para um usuário, não é feita a agregação de fluxos para transmissão *multicast*.
Cetinkaya e Sayit (2016)	ALTO, HTTP.	Não é abordado o problema da redução do uso de recursos, e sim a escolha de um caminho de roteamento com a maior banda disponível.	Não é levado em consideração a quantidade de usuários simultâneos, pois considera a estrutura como uma CDN.	Como considera uma CDN, a agregação de fluxos para transmissões *multicast* não é executada.
Yu, Wang e Hsu (2015)	Não especificado.	Não é otimizado o uso da largura de banda, e sim e escolha de um caminho para o roteamento com a maior banda disponível, como em Cetinkaya e Sayit (2016).	A solução é modelada para um único usuário, na qual melhora a qualidade do serviço disponibilizado para o mesmo, e não leva em consideração usuários simultâneos.	Como não leva em consideração usuários simultâneos, não é feita a agregação de fluxos, apenas o roteamento desse fluxo pelo caminho de menor custo.
Egilmez, Civanlar e Tekalp (2013)	HTTP, RTP.	A utilização da largura de banda é levada em consideração para realizar o roteamento dos fluxos pelo caminho de melhor banda, e não com o intuito de reduzir a quantidade de banda necessária para a transmissão.	Assim como os demais trabalhos, não é abordado o problema de usuários simultâneos, apenas o roteamento de melhor caminho para um único usuário.	Como é uma abordagem de roteamento para um único usuário, não é feita a agregação de fluxos para transmissões *multicast* para a otimização do uso dos recursos da rede.

Fonte: Elaborado pelo Autor (2017)

Como descrito na Tabela 3.3, embora auxiliem na resolução de alguns problemas de sistemas de transmissão de vídeo NVoD das mais variadas formas, nenhum dos autores identificados aborda todos os pontos descritos na Tabela 3.3. A grande maioria dos trabalhos utilizam técnicas de roteamento para a entrega do conteúdo, priorizando apenas um usuário, sem levar em consideração que possam existir (e usualmente existem) vários usuários simultâneos requisitando o mesmo conteúdo descrito pela métrica de Agregação de Fluxos.

3.5 Considerações parciais

Neste capítulo são abordadas as Redes Definidas por *Software* e suas principais diferenças quando comparadas às Redes Convencionais. Essas características diferenciadas pertencentes as SDN possibilitam a pesquisa e desenvolvimento para as mais diversas aplicações. Com isso, os sistemas NVoD para a transmissão de vídeos são bastante pesquisados e experimentados com abordagens referentes a SDN, principalmente as arquiteturas OpenFlow,

por seu paradigma orientado a fluxos, na qual auxilia na entrega do conteúdo para usuários simultâneos.

Através de trabalhos relacionados na área de transmissão de vídeo, mais especificamente utilizando arquiteturas OpenFlow, foi constatado que os problemas existentes em sistemas NVoD são dificilmente abordados por uma mesma solução. Portanto, o desenvolvimento de um trabalho para um sistema NVoD que suporte diversos usuários simultaneamente, juntamente com a otimização do uso dos recursos da rede, como a largura de banda, torna-se necessário.

4 Solução proposta

O objetivo deste trabalho é desenvolver uma abordagem baseada em Redes Definidas por *Software* para sistemas de transmissão de vídeos, mais especificamente sistemas NVoD. Desta forma, o presente capítulo descreve a solução proposta, assim como os requisitos funcionais, que são necessários para compreender as necessidades do sistema, a fim de resolver os problemas levantados na problemática do estudo. Juntamente com os requisitos, é definido o plano de testes, necessário para avaliar a abordagem experimentalmente.

4.1 Especificação de requisitos

A fim de resolver os problemas descritos na problemática do estudo, apresentada na Seção 2.5, e utilizados como parâmetros de análise para os trabalhos relacionados, apresentados na Seção 3.4, este capítulo apresenta uma abordagem para a transferência de conteúdo em vídeo com Redes Definidas por *Software*. Desta forma, para a elaboração de uma abordagem, existem requisitos que devem ser levados em consideração. Contudo, existem também alguns pré-requisitos para se tornar operacional:

- Ter acesso aos elementos de rede, como os *switches* OpenFlow, com o objetivo de obter informações e manipular suas tabelas de fluxo; e
- Ser capaz de manipular o fluxo de dados sendo transmitidos na rede através do controlador SDN.

Os requisitos são abstrações dos problemas apresentados nos sistemas NVoD, os quais não foram abordados integralmente nos trabalhos relacionados identificados nesta pesquisa. Os requisitos para conseguir transmitir fluxos de vídeo em redes gerenciáveis[1] de modo a otimizar a largura de banda, suportar diversos usuários simultâneos e buscar a agregação de fluxos *multicast*, são divididos em funcionais e não funcionais:

Requisitos funcionais:

[1] A rede gerenciável em questão trata-se de uma SDN, com acesso ao(s) controlador(es) da rede, responsáveis pelo gerenciamento.

- RF1 - Oferecer serviço de distribuição de conteúdo em vídeo empregando redes SDN de forma a otimizar o uso dos recursos da rede;

- RF2 - Identificar as condições da rede e usar essa informação para definir as ações a serem tomadas com uma nova requisição, ou seja, se é necessário iniciar uma nova conexão com o servidor provedor de conteúdo ou não; e

- RF3 - Minimizar o tráfego na rede, eliminando os fluxos redundantes. Porém, tal redução de tráfego não reduz a qualidade da transmissão para o usuário.

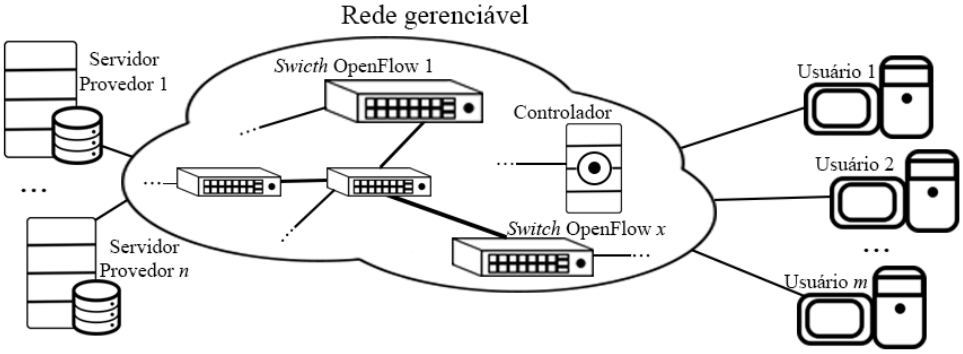
Requisitos não funcionais:

- RNF1 - Permitir que um número considerável de usuários receba o conteúdo de forma satisfatória, ou seja, sem atrasos ou perda de quadros.

4.2 Descrição da solução MMFA

Uma abordagem denominada *Multimedia Flow Aggregator* (MMFA), para a resolução dos problemas descritos na problemática do estudo (Seção 2.5) é descrita nesta Seção. Com os requisitos tanto funcionais como não funcionais especificados, a Figura 4.1 ilustra a abordagem de forma genérica, que por sua vez engloba todas as possibilidades de casos de uso e ambientes suportados pela proposta. Esses casos de uso são referentes à quantidade de Servidores, quantidade de *switches* OpenFlow e a quantidade de usuários solicitando o serviço, assim como um controlador da rede gerenciável.

Figura 4.1: Ilustração da abordagem



Fonte: Elaborado pelo autor (2017)

Conforme ilustrado na Figura 4.1, a transmissão de conteúdo em vídeo do tipo NVoD caracteriza-se pela transmissão de um mesmo conteúdo de forma sincronizada a

múltiplos usuários simultâneos. O conteúdo disponibilizado pelo fornecedor é transferido aos usuários, muitas vezes, em infraestruturas de rede compartilhadas. Neste contexto, surgiu a solução MMFA, uma abordagem que aproveita-se das características peculiares do NVoD e do compartilhamento da infraestrutura de rede para reduzir o número de pacotes trafegados na rede. A solução MMFA atua junto aos fluxos de rede, otimizando o uso da infraestrutura através de um mecanismo de identificação e agregação de fluxos redundantes em um mesmo enlace.

Uma solução de transmissão de conteúdo em vídeo NVoD consiste na existência de f fornecedores de conteúdo ($f \geq 1$), s *switches* ($s \geq 1$), u usuários ($u \geq 1$) e e enlaces ($e = u*s+1$). Uma transmissão (t) caracteriza-se pela transferência de conteúdo em vídeo, no formato de pacotes (p), de um fornecedor (f) a um usuário (u) através de n enlaces (e). Isso só ocorre se $t, p, f, u, e, s, n > 0$. A abordagem MMFA acrescenta a modelagem a restrição que os *switches* (s) compartilhados entre os usuários (u) sejam gerenciáveis e suportem SDN (sg). Então, o novo e é igual a $u * (s + sg) + 1$, no qual $(s + sg) > 0$. Comparando a abordagem MMFA com a modelagem existente, pode-se concluir que não há mudanças nos fornecedores nem nos usuários, tornando a implantação mais simples.

A solução MMFA é aplicada nos *switches* SDN (sg), portanto só atua se $sg > 0$. Um determinado conteúdo é escolhido por m usuários u de um mesmo fornecedor f e é formado por h pacotes p, onde $m, h > 0$. Estes $h * p$ pacotes trafegam através do conjunto de e enlaces. MMFA monitora os fluxos nos sg, avalia o conteúdo de p e, caso o e de destino seja o mesmo, cataloga como fluxo redundante, enviando somente p. Sem MMFA, o número de pacotes repassados no enlace e seria $m * p$. Portanto, MMFA reduz o envio de $(m - 1) * p$ pacotes no enlace e.

Vale ressaltar que o escopo é uma rede LAN, o que implica em uma restrição da solução MMFA. Porém, caso uma rede WAN satisfaça os pré-requisitos, assim como o suporte ao protocolo OpenFlow em seus equipamentos, a solução pode ser aplicada.

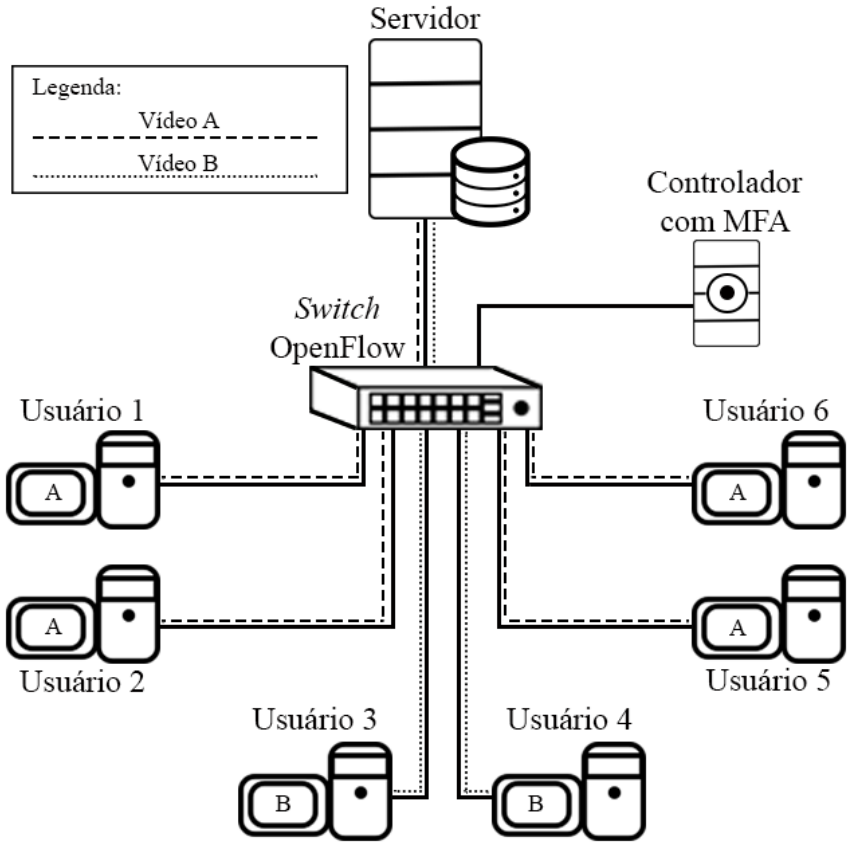
Com a MMFA, espera-se uma maior escalabilidade do sistema no quesito quantidade de usuários simultâneos, justamente pela otimização dos fluxos evitando redundância de conteúdo no enlace de comunicação entre o servidor e o *switch* mais próximo ao usuário. Essa melhora de escalabilidade é comparada com a transmissão em uma Rede Convencional, na qual não executa a agregação de fluxos semelhantes. Com isso, na MMFA mais usuários conseguirão assistir conteúdos diferentes de forma simultânea.

Assim como a escalabilidade, espera-se uma melhora na qualidade da transmissão, nos quesitos de latência no tempo de resposta do servido para a requisição. Quando comparado com uma transmissão em uma Rede Convencional com a mesma quantidade de usuários simultâneos, o uso do enlace entre o provedor de vídeo e a rede SDN de acesso deve permanecer com apenas o tráfego de vídeo independente da quantidade de usuários. Tal otimização é obtida pela otimização do uso da largura de banda e eliminação dos fluxos redundantes.

4.3 Arquitetura

Com os requisitos do sistema definidos e a solução MMFA apresentada, a Figura 4.2 ilustra um exemplo de sistema que satisfaz os requisitos impostos para um sistema NVoD. O sistema apresenta múltiplos usuários simultâneos e conteúdos em vídeo diferenciados sendo transmitidos para cada usuário.

Figura 4.2: Cenário exemplificando o sistema



Fonte: Elaborado pelo autor (2017)

Conforme ilustrado na Figura 4.2, a solução MMFA consiste em um servidor provedor do conteúdo em vídeo, um *switch* OpenFlow, um controlador SDN e vários

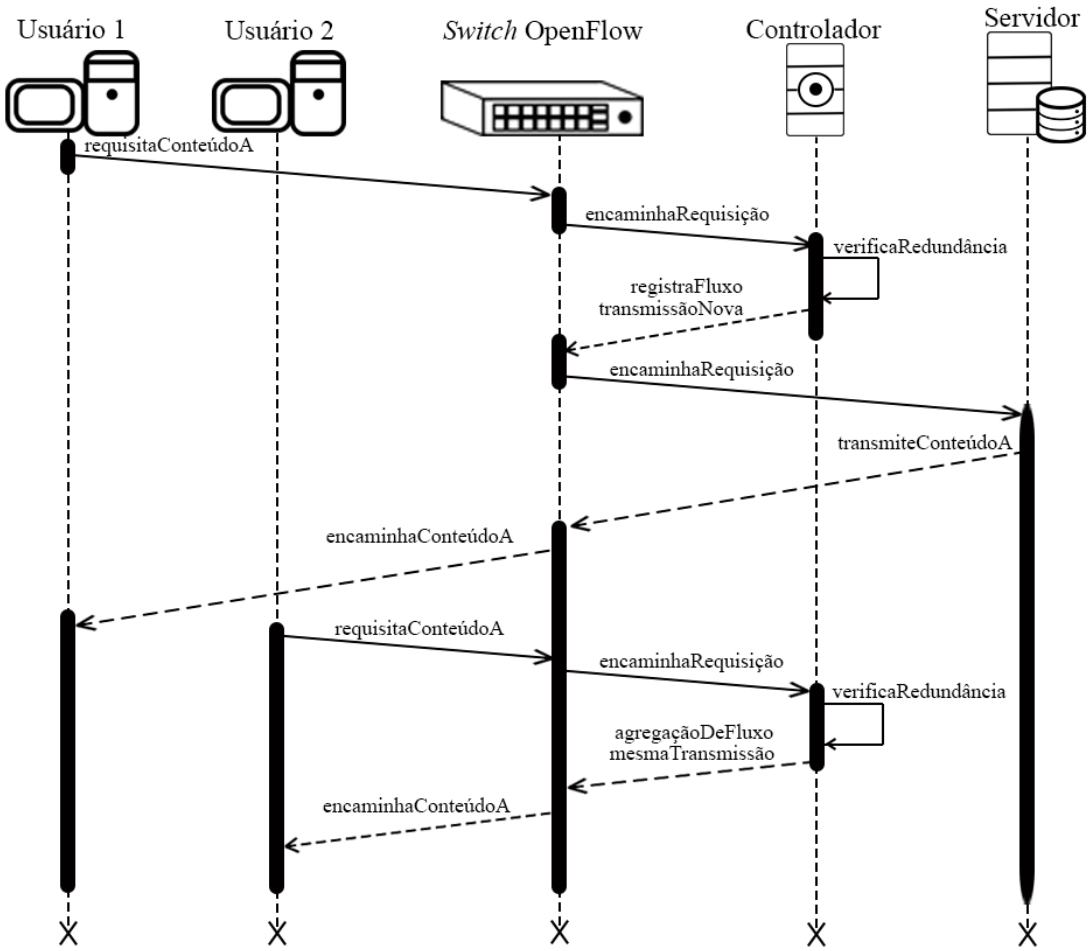
usuários requisitando conteúdos em vídeo para o servidor. O controlador verifica as requisições de conteúdo e agrupa as requisições semelhantes, com a finalidade de distribuir tal conteúdo a todos os usuários que requisitaram o mesmo conteúdo. Porém, isso deve ser feito para todas as requisições que são de conteúdos diferentes, como ilustra o exemplo na Figura 4.2, na qual o servidor fornece dois tipos de conteúdo, o A e o B, e quatro usuários requisitam o conteúdo A enquanto dois usuários requisitam o conteúdo B. No total são seis requisições, porém apenas duas conexões são feitas com o servidor a partir do *switch*, em vez de seis conexões, o que otimiza o uso da largura de banda do servidor e não congestiona tal enlace de comunicação, e principalmente, evita os fluxos redundantes.

De um modo geral, a sequência de eventos no sistema é ilustrada através de um diagrama de sequência descrito na Figura 4.3. O exemplo utilizado aborda uma topologia com dois usuários, que requisitam o mesmo conteúdo. O Usuário 1 requisita um conteúdo A, essa requisição é enviada até o *switch*, que por sua vez requisita ao controlador a verificação de redundância. A verificação de redundância analisa se já existe um fluxo do mesmo conteúdo sendo transmitido para outro usuário. Caso não exista um fluxo, é registrado um novo fluxo na tabela do *switch* e iniciada uma nova conexão com o servidor provedor de conteúdo, que inicia a transmissão e envia o conteúdo até o *switch*, que por fim encaminha o conteúdo para o Usuário 1. Com essa primeira conexão estabelecida, o Usuário 2 requisita o mesmo conteúdo, as verificações são efetuadas novamente, e constado que tal conteúdo já está sendo transmitido para outro usuário. Com isso, o *switch* apenas encaminha o mesmo fluxo de conteúdo para o Usuário 2, totalizando apenas uma conexão com o servidor, e dois usuários atendidos.

Embora o sistema tenha o objetivo de otimizar o uso de banda no enlace de comunicação entre o servidor e o *switch*, existem situações limítrofes. Como o sistema estabelece uma conexão nova com o servidor para transmitir um conteúdo que ainda não está sendo transmitido, existe um limite de conexões suportadas, ou seja, existe uma quantidade máxima de vídeos diferentes que podem ser transmitidos simultaneamente. Com isso, é possível definir situações limítrofes de melhor e pior cenário:

- Melhor cenário: Banda suficiente para fornecer conteúdo para todos os usuários simultaneamente, com diferentes conteúdos sendo transmitidos, e todos os usuários recebendo o conteúdo em sua forma integral e com baixa latência; e
- Pior cenário: Quantidade alta de conteúdos diferentes sendo transmitidos simulta-

Figura 4.3: Sequência de eventos para a transmissão no sistema MMFA



Fonte: Elaborado pelo autor (2017)

neamente, que extrapola o limite de banda disponível no enlace de comunicação. Essa situação implica em usuários insatisfeitos, pois podem receber o conteúdo com cortes pela perda de pacotes, ou até mesmo com uma maior latência.

Com as características do ambiente e seus componentes definidos, é necessário desenvolver um plano de testes. Posteriormente, tais testes são executados a fim de avaliar a solução MMFA. Essa avaliação terá como base as transmissões em Redes Convencionais.

4.4 Plano de testes

No plano de testes, são definidos os experimentos a serem executados na abordagem proposta, com finalidade de avaliação em relação a uma transmissão em uma Rede Convencional. O plano de testes aqui descrito está dividido em dois critérios, cada um com um objetivo diferente.

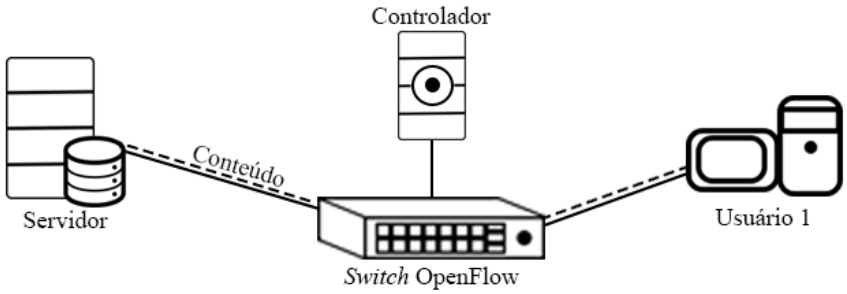
- O primeiro critério consiste na avaliação do tempo de resposta entre a requisição e o recebimento do conteúdo; e

- O segundo critério consiste na avaliação do consumo da banda no enlace entre o servidor e o *switch* OpenFlow.

Os experimentos são executados em três cenários distintos, com características próprias, como quantidade de usuários e quantidade de vídeos trafegados. São eles:

- **Cenário 1.** Inicialmente, a topologia experimental envolve um servidor de conteúdo, um *switch* OpenFlow controlado por um controlador e um único usuário requisitando um conteúdo em vídeo. Esse experimento inicial tem como objetivo verificar o funcionamento da abordagem em um cenário simples e estabelecer um *baseline* da vazão de dados, ilustrado na Figura 4.4.

Figura 4.4: Primeiro cenário de experimentação

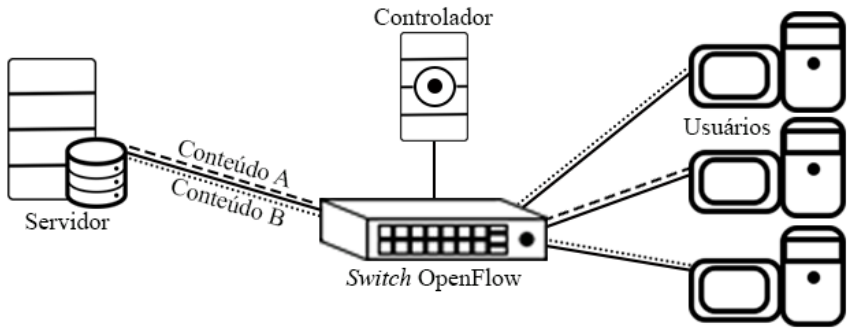


Fonte: Elaborado pelo autor (2017)

Nesse primeiro cenário, espera-se um desempenho muito semelhante ao obtido na transmissão em Rede Convencional com a mesma topologia, tanto no primeiro experimento quanto no segundo. Esse resultado semelhante é por conta de não haver fluxo redundante trafegando na rede, justamente pela existência de um único usuário.

- **Cenário 2.** A segunda topologia contém um servidor, um *switch* OpenFlow controlado por um controlador e três usuários requisitando dois conteúdos diferentes. Esse experimento tem como objetivo avaliar a solução MMFA em sua capacidade de agregar fluxos, pois dois usuários estarão recebendo um conteúdo igual, enquanto um outro usuário estará recebendo um conteúdo diferente. O cenário é ilustrado pela Figura 4.5.

Figura 4.5: Segundo cenário de experimentação



Fonte: Elaborado pelo autor (2017)

O esperado para esse experimento é um desempenho superior ao compará-lo à Rede Convencional, pois existe a agregação de fluxo, evitando redundância. Esse desempenho superior é dado pela otimização do uso dos recursos no enlace entre o servidor e o *switch* OpenFlow, reduzindo o consumo. Também é esperada a melhora no tempo de resposta, pelo fato dos usuários que tem seus fluxos agregados não precisarem iniciar uma conexão com o servidor.

- **Cenário 3.** O terceiro cenário é semelhante ao Cenário 2, porém os três usuários requisitam o mesmo conteúdo. Esse experimento tem como objetivo analisar a escalabilidade da solução MMFA, comportando uma quantidade maior de usuários em relação à Rede Convencional.

  Espera-se um desempenho superior quando comparado à Rede Convencional, assim como no Cenário 2, pelo fato de existir a agregação de fluxos semelhantes. A utilização dos recursos do enlace nesse cenário (com três usuários) deve ser similar à utilização do Cenário 1 (com apenas um usuário). Enquanto o tempo de resposta deve ser otimizado em relação à Rede Convencional, exceto ao primeiro usuário requisitante, por conta da não necessidade da conexão com o servidor.

Como comparativo, são utilizados cenários de Redes Convencionais semelhantes às topologias da abordagem proposta em SDN, descritas em cada um dos cenários. Porém, não há a inclusão do controlador em cada uma das topologias, ou seja, tornando a rede não gerenciável.

Todos os vídeos utilizados para os experimentos possuem resolução HD 720p com duração de 2 minutos, e tamanho aproximado de 85MB. A resolução escolhida se dá pelo fato de ser amplamente utilizada, e também pelo fato de que a resolução do

conteúdo não influencia no comportamento da solução, sendo desnecessária a repetição dos experimentos com resoluções diferenciadas. A duração de 2 minutos é suficiente para as aferições de consumo de banda com múltiplos usuários e tempo de resposta.

4.5 Considerações parciais

A definição de requisitos, tanto funcionais como não funcionais é relevante para a criação de uma solução para garantir seu desempenho e bom funcionamento. Assim como, a especificação e elaboração da arquitetura da solução, na qual descreve suas capacidades e limitações. Neste capítulo, tanto requisitos como especificação de arquitetura da solução MMFA são apresentados, assim como a sequência de eventos da abordagem, fundamental para a compreensão do seu funcionamento. Também é definido o plano de testes, necessário para comparar a solução com as Redes Convencionais. O plano de testes é dividido em cenários, os quais permitem analisar o funcionamento da solução MMFA em diversas situações distintas.

5 Implementação, experimentos e resultados

Este capítulo descreve os detalhes da implementação da solução MMFA, juntamente com as tecnologias utilizadas. Assim como a implementação, são apresentadas as execuções dos dois experimentos em cada um dos três cenários (Seção 4.4), e os resultados obtidos em cada experimento. São utilizadas como comparativo as transmissões em Redes Convencionais, que não executam o tratamento para fluxos redundantes.

5.1 Implementação

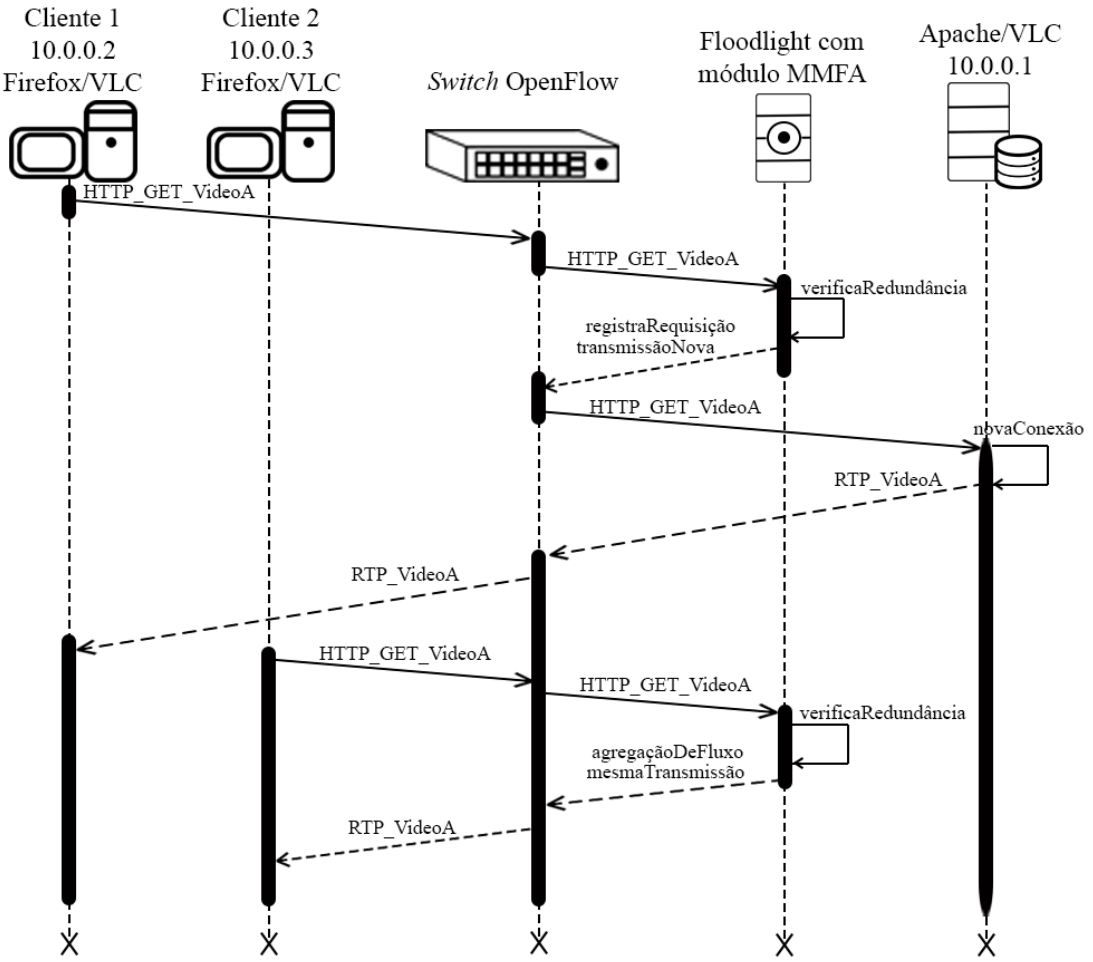
A solução MMFA é implementada em linguagem Java como um módulo no controlador Floodlight[1], para o gerenciamento da rede e das requisições de conteúdo em vídeo. Esse módulo tem como objetivo receber as requisições, registrá-las em uma lista, analisar a redundância e criar as regras na tabela de fluxos do *switch*, caso tal redundância exista. Caso exista a redundância e a regra seja criada, a requisição é barrada no próprio controlador, não permitindo que a mesma prossiga em direção ao servidor, pois o cliente receberá o conteúdo em vídeo através da agregação dos fluxos redundantes. Caso contrário, o módulo é capaz de encaminhar a requisição para uma nova conexão com o servidor, criando um novo fluxo de vídeo.

A criação da regra da tabela de fluxos utiliza a lista de requisições, na qual dispõe de informações como o identificador do vídeo, endereço IP do cliente requisitante e porta de saída do *switch* na qual o cliente está conectado. Assim que o controlador recebe uma requisição redundante, um laço de repetição percorre a lista procurando o mesmo identificador de vídeo recebido, adicionando as portas de saída dos clientes (com o mesmo identificador de vídeo) na regra, que posteriormente é escrita no *switch* OpenFlow. Com a regra escrita, o servidor continua encaminhando o vídeo normalmente para o primeiro cliente que o requisitou, porém esse fluxo de pacotes é encaminhado pelo *switch* para todos os clientes que foram identificados com requisições de mesmo identificador de vídeo. A Figura 5.1 utiliza como base a sequência de eventos da Figura 4.3, porém descreve especificamente a troca de mensagens entre os nós e as tecnologias utilizadas pela solução

[1] `http://www.projectfloodlight.org/floodlight/`

MMFA, como o servidor Apache[2], VLC *Media Player*[3], controlador OpenFlow Floodlight[4] e os clientes criados com a ferramenta de simulação de redes Mininet[5], assim como o *switch* OpenFlow, também criado através da mesma ferramenta.

Figura 5.1: Diagrama de sequência da solução MMFA



Fonte: Elaborado pelo autor (2017)

A rede criada para o experimento é definida com endereço IP 10.0.0.0/24, e como pode ser observado na Figura 5.1, os endereços dos nós dessa rede são definidos como Servidor: 10.0.0.1, Cliente 1: 10.0.0.2 e Cliente 2: 10.0.0.3. O endereço do Cliente 3 é definido como 10.0.0.4. Como fornecedor de vídeo, é utilizado um servidor Apache executando na máquina do nó servidor para o recebimento das requisições e disponibilidade da interface Web para o cliente escolher o conteúdo, e a ferramenta VLC *Media Player* para o envio do conteúdo em vídeo. Ao receber uma requisição, é executado um *script* em Shell no lado do servidor, na qual captura o IP do cliente e inicializa a transmissão

[2]Versão Apache utilizada: 2.4.18 - `https://httpd.apache.org/`
[3]Versão VLC utilizada: 2.2.6 - `https://www.videolan.org/vlc/index.pt-BR.html`
[4]Versão Floodlight utilizada: 1.2 - `http://www.projectfloodlight.org/download/`
[5]Versão Mininet utilizada: 2.2.1 - `http://mininet.org/`

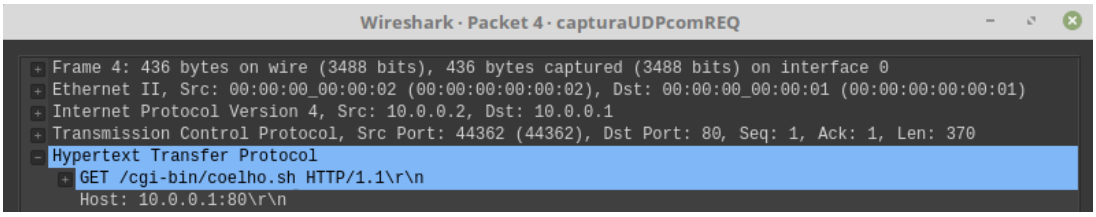
com o protocolo RTP através do VLC *Media Player*. A Figura 5.2, apresenta a interface disponibilizada aos clientes para a escolha do conteúdo desejado. Conforme ilustrado na Figura 5.2, o usuário deve inserir em seu navegador o endereço do servidor seguido da porta, no formato <endereço>:<porta>. No exemplo, o servidor está configurado para atender na porta TCP/80, com isso o endereço completo fica 10.0.0.1:80. Em seguida, o cliente escolhe o conteúdo através dos botões disponíveis na interface, apenas clicando sobre um.

Figura 5.2: Interface Web para o usuário

Fonte: Elaborado pelo autor (2017)

A escolha do servidor Apache se dá pelo fato da necessidade de uma requisição partindo do usuário. Essa requisição é feita através de um pacote HTTP com método GET na Camada de Aplicação, informando qual conteúdo o usuário está requisitando, ilustrada na Figura 5.1. Tal pacote é utilizado pelo módulo do controlador para analisar a requisição, assim como armazená-la e verificar os dados do usuário. A Figura 5.3 ilustra o pacote HTTP com método GET capturado pelo Wireshark.

Figura 5.3: GET transmitindo as informações da requisição



Fonte: Elaborado pelo autor (2017)

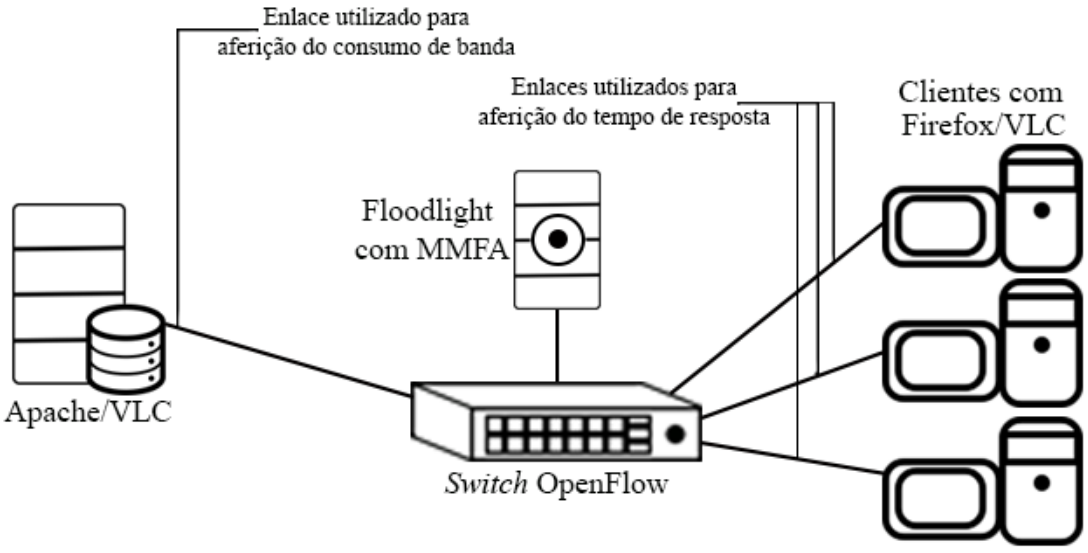
Como ilustrado na Figura 5.3, é possível identificar diversos campos de informações no pacote HTTP com método GET. Na Camada de Aplicação é possível identificar pelo protocolo HTTP o conteúdo requisitado, no exemplo "coelho.sh"na qual define o *script* de execução para o vídeo "coelho.mp4". Na Camada de Rede, identifica-se através

do protocolo IPv4 o endereço do usuário requisitante (Src) e do servidor (Dst). Essas informações são utilizadas pelo módulo no controlador para a identificação das requisições e montagem das regras na tabela de fluxos do *switch*, assim como são utilizadas pelo servidor para o envio do conteúdo para o usuário, identificando o endereço do mesmo pelo pacote HTTP com método GET e utilizando no *script* de execução do VLC *Media Player* para a montagem do comando de transmissão pelo protocolo RTP. O motivo para a escolha do protocolo RTP para as transmissões de vídeo é descrito no Capítulo 2, Subseção 2.3.2.

Para a criação dos três cenários, é utilizada a ferramenta de simulação de redes Mininet. Essa ferramenta proporciona a criação de uma rede virtual, com dispositivos virtuais, assim como *links* entre os mesmos, emulando uma rede real. Essas topologias são criadas através de *scripts* em linguagem Python, especificando os *hosts* e *switches*, assim como os endereços de cada nó e endereço do controlador OpenFlow.

A coleta de dados da rede é executada através da ferramenta Wireshark[6]. A ferramenta captura pacotes trafegados na rede e os organiza por protocolos, com o objetivo de análise e monitoramento da rede, utilizando o filtro para capturar o tráfego desejado. A Figura 5.4 apresenta os enlaces utilizados para a aferição de cada experimento.

Figura 5.4: Enlaces utilizados em cada critério do experimento



Fonte: Elaborado pelo autor (2017)

Conforme a Figura 5.4, o monitoramento e coleta de dados ocorre na porta do *switch* em que cada usuário está conectado, para principalmente aferir os tempos

[6]Versão Wireshark utilizada: 2.4.2 - `https://www.wireshark.org/`

de resposta entre o envio da requisição e o recebimento do vídeo, utilizado no Primeiro Experimento apresentado na Seção 4.4. Para a aferição do consumo de banda, é utilizado o módulo *Statistics* do controlador Floodlight, na qual utiliza REST API para retornar a vazão em cada porta do *switch*, utilizado no Segundo Experimento também descrito na Seção 4.4. A execução do módulo *Statistics* é ilustrada na Figura 5.5.

Figura 5.5: Exemplo de retorno do módulo *Statistics* informando a vazão no enlace


```
[{"dpid":"00:00:00:00:00:00:00:01","port":"4","updated":"Thu Oct 19 21:24:13
10000000","bits-per-second-rx":"1688224","bits-per-second-tx":"458"}]\n\n
```


Fonte: Elaborado pelo autor (2017)

Exemplificado na Figura 5.5, o módulo retorna em bits por segundo a quantidade de tráfego na porta especificada. Esse tráfego é tanto os dados recebidos (rx) quanto os dados enviados (tx) expressos em bits por segundo. Como cada comando REST retorna uma aferição, um *shell script* é necessário para criar um laço de execução de tal comando, retornando a quantidade de tráfego no enlace a cada segundo durante toda a transmissão do conteúdo em vídeo.

5.2 Experimentos

Conforme definido, o Plano de Testes (Seção 4.4) possui cenários com o uso da solução MMFA e sem o seu uso (Redes Convencionais), usando como critério para comparação:

- O consumo de banda consiste na quantidade de recursos utilizados para atender as requisições dos clientes, no enlace de comunicação entre o servidor e o *switch*. Para a obtenção do consumo, faz-se necessária a utilização do módulo de estatísticas do controlador Floodlight, denominado *Statistics*. O módulo *Statistics* monitora a vazão de dados na porta do *switch* na qual está conectado o servidor, retornando a vazão em bits por segundo, o que indica o consumo de banda no enlace entre o *switch* e o servidor. Embora o módulo *Statistics* retorne os valores em bits por segundo, para maior clareza nas ilustrações o consumo é representada em kilobits por segundo.

- O tempo de resposta consiste no intervalo de tempo decorrido entre o envio da requisição (pacote HTTP com método GET) do cliente para o servidor, e o recebimento do primeiro pacote de vídeo no lado do cliente (pacote RTP). Para a aferição desses

valores, faz-se necessária a ferramenta Wireshark capturando o tráfego no enlace entre o cliente requisitante e o *switch*, pelo fato dos clientes que possuem o fluxo agregado não terem suas requisições enviadas até o servidor. Com isso, a aferição pode ser feita apenas no enlace entre cliente requisitante e *switch*.

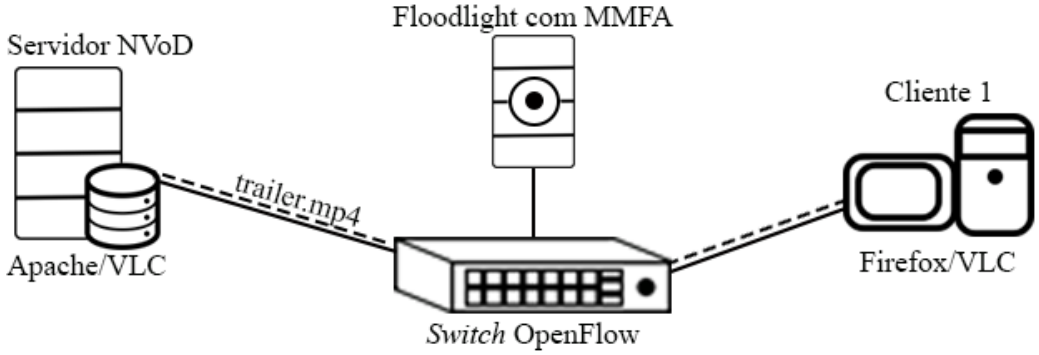
O objetivo é aferir os valores para a solução MMFA e compará-los com os valores obtidos nas transmissões em Redes Convencionais. Cada aspecto dos experimentos foi executado dez vezes, com isso, todos os valores apresentados são a média das execuções, acompanhados do desvio padrão. A ordem de requisição manteve-se a mesma para todos os critérios e cenários:

- Cliente 1 requisita trailer.mp4 aos 10 segundos de experimento;
- Cliente 2 requisita trailer.mp4 aos 50 segundos de experimento; e
- Cliente 3 requisita coelho.mp4 aos 90 segundos de experimento.

5.2.1 Cenário 1

Como descrito da Seção 4.4, o primeiro cenário consiste em apenas um cliente recebendo um conteúdo em vídeo. O principal objetivo de experimentação nesse cenário é estabelecer um *baseline* do consumo de banda. Além disso, é possível verificar se a solução proposta está, de alguma forma, interferindo na transferência em si, o que não deveria acontecer. A Figura 5.6 ilustra a topologia de experimentação utilizada no Cenário 1.

Figura 5.6: Topologia utilizada para o Cenário 1

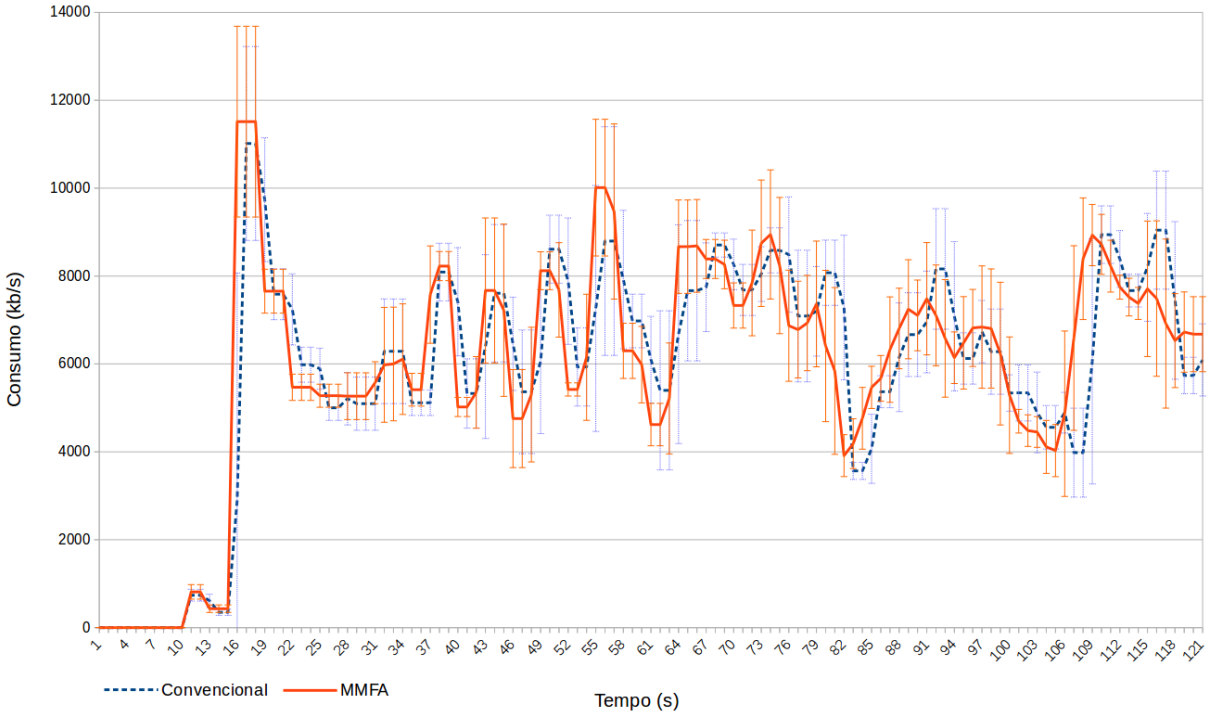


Fonte: Elaborado pelo autor (2017)

Após dez execuções do experimento com o intuito de medir o consumo de banda no enlace entre servidor e *switch*, obteve-se o gráfico de consumo para o Cenário 1.

A Figura 5.7 ilustra os resultados obtidos, informando o consumo em kilobits por segundo (kb/s) no eixo y e o tempo de experimento em segundos (s) no eixo x.

Figura 5.7: Consumo de banda para o Cenário 1



Fonte: Elaborado pelo autor (2017)

Para o critério de tempo de resposta, as aferições foram realizadas no enlace de comunicação entre o cliente e o *switch*. Os valores obtidos para esse critério são expressos em segundos (s) na Tabela 5.1, enquanto o desvio padrão é representado pelas colunas σ.

Tabela 5.1: Tempo de resposta para o Cenário 1

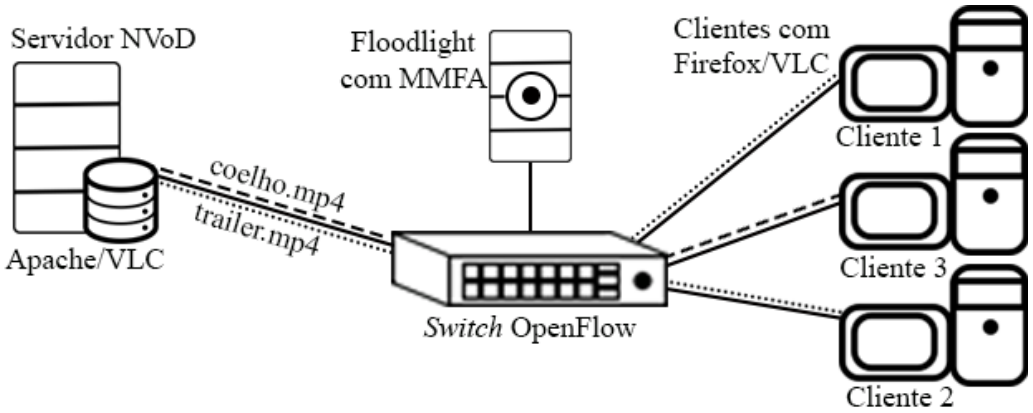
	Cliente 1	σ
Cenário 1 - Convencional	1.184720517	0.01112
Cenário 1 - MMFA	1.212694582	0.03877

Fonte: Elaborado pelo autor (2017)

5.2.2 Cenário 2

O segundo cenário de experimentação possui elementos a mais quando comparado ao primeiro cenário. Esses elementos extras são dois clientes e um conteúdo em vídeo diferenciado disponibilizado. A Figura 5.8 ilustra a topologia para esse cenário.

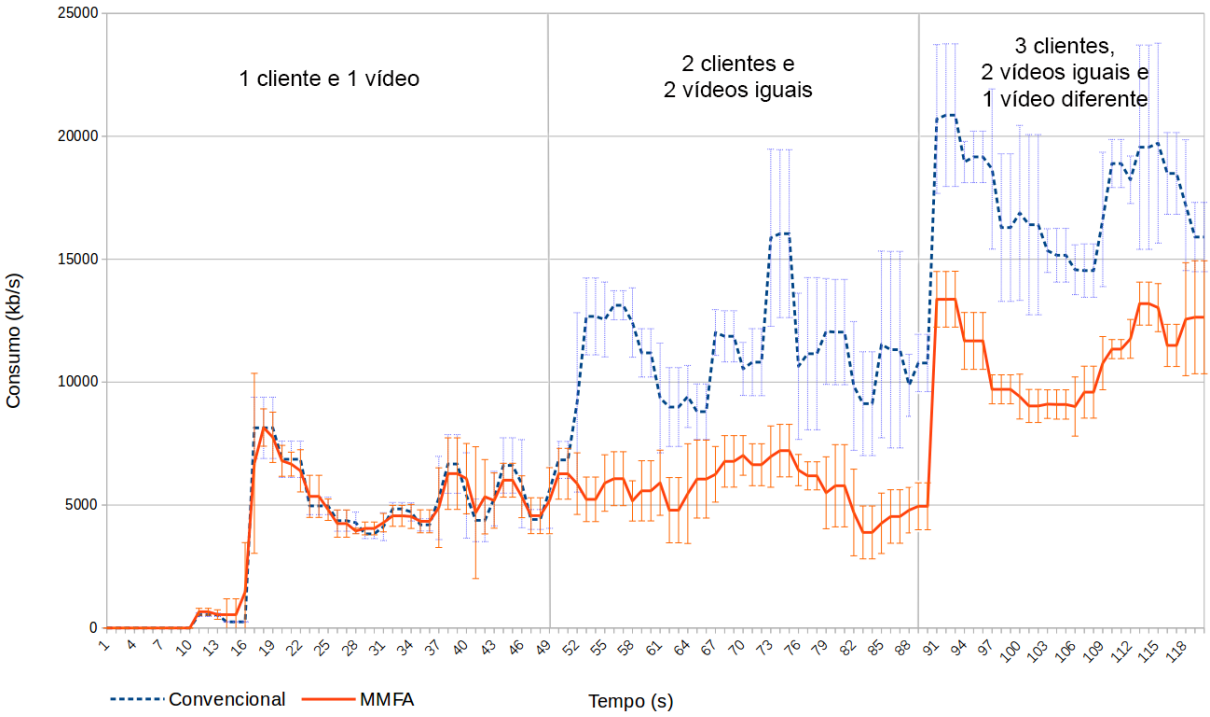
Figura 5.8: Topologia utilizada para o Cenário 2



Fonte: Elaborado pelo autor (2017)

Após dez execuções do experimento com o intuito de medir o consumo de banda no enlace entre servidor e *switch*, obteve-se o gráfico de consumo para o Cenário 2. A Figura 5.9 ilustra os resultados obtidos. Assim como no Cenário 1, é informado o consumo em kilobits por segundo (kb/s) no eixo y e o tempo de experimento em segundos (s) no eixo x.

Figura 5.9: Consumo de banda para o Cenário 2



Fonte: Elaborado pelo autor (2017)

No critério Tempo de Resposta, as aferições foram realizadas no enlace de comunicação entre o cliente e o *switch*, como no Cenário 1, porém agora com os três clientes. Os valores obtidos para esse critério são descritos em segundos (s) na Tabela 5.2.

Tabela 5.2: Tempo de resposta para o cenário 2

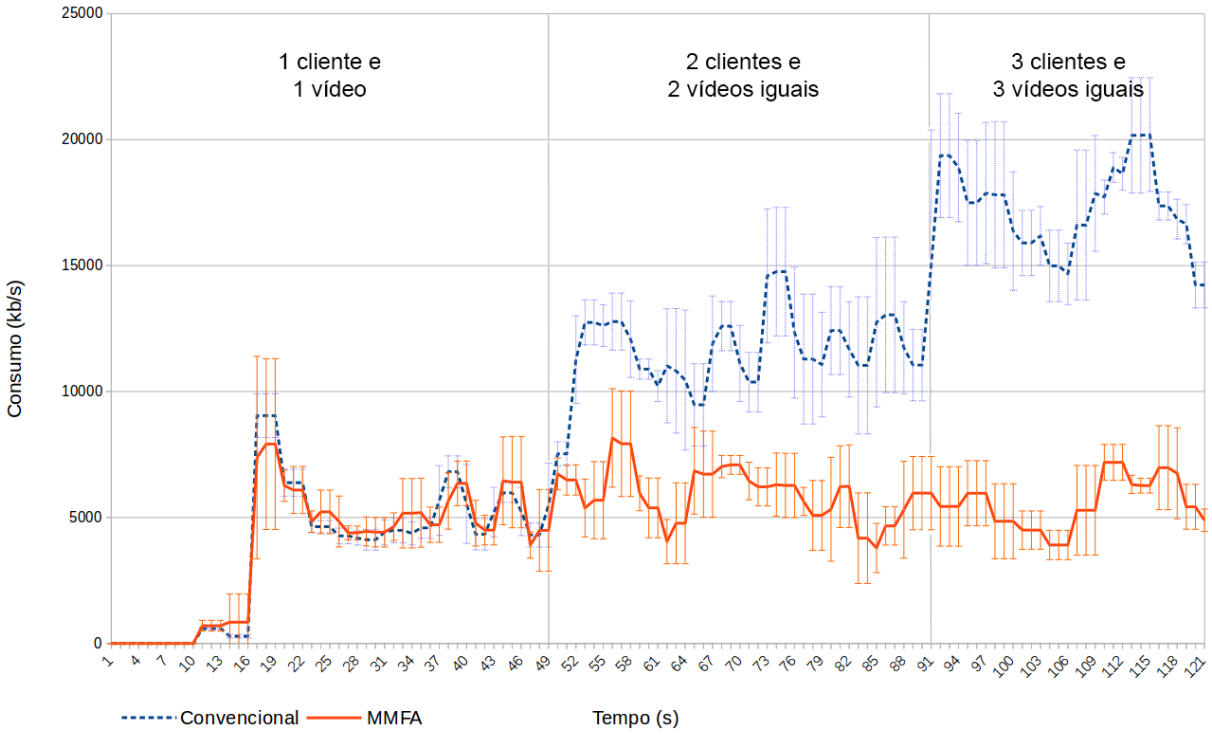
	Cliente 1	σ	**Cliente 2**	σ	**Cliente 3**	σ
Cenário 2 - Convencional	1.194428354	0.02197	1.188755432	0.01402	1.184200267	0.01740
Cenário 2 - MMFA	1.218026603	0.01692	0.067012619	0.04217	1.204483151	0.02411

Fonte: Elaborado pelo autor (2017)

5.2.3 Cenário 3

O Cenário 3 tem como objetivo analisar a escalabilidade do sistema em relação ao comportamento com diversos usuários simultâneos requisitando um mesmo conteúdo. Com isso, apenas um conteúdo é disponibilizado. Contudo, três clientes fazem a requisição desse mesmo conteúdo em momentos distintos. A Figura 5.10 ilustra os resultados obtidos para esse cenário de experimentação, e assim como os demais cenários, é informando o consumo em kilobits por segundo (kb/s) no eixo y e o tempo de experimento em segundos (s) no eixo x.

Figura 5.10: Consumo de banda para o Cenário 3



Fonte: Elaborado pelo autor (2017)

Para o critério de tempo de resposta, as aferições foram realizadas no enlace de comunicação entre o cliente e o *switch*, como nos demais cenários, porém agora com os três clientes requisitando o mesmo conteúdo, não exigindo uma nova conexão para todos

na solução MMFA. Os valores obtidos para esse critério são descritos em segundos (s) na Tabela 5.3.

Tabela 5.3: Tempo de resposta para o conário 3

	Cliente 1	σ	**Cliente 2**	σ	**Cliente 3**	σ
Cenário 3 - Convencional	1.194428354	0.02197	1.188755432	0.01402	1.184200267	0.01740
Cenário 3 - MMFA	1.246489618	0.03765	0.057044535	0.02182	0.105130000	0.05078

Fonte: Elaborado pelo autor (2017)

Os valores de tempo de resposta para as Redes Convencionais são similares aos valores do Cenário 2 para esse tipo de transmissão. Isso se dá pelo fato de todas as requisições, tanto de conteúdo redundante quanto de conteúdo diferenciado, serem encaminhadas para uma nova conexão com o servidor.

5.3 Análise dos resultados

A análise dos resultados, diferente da execução dos experimentos que foi dividida em cenários, é dividida de acordo com os critérios de avaliação: consumo de banda e tempo de resposta. Essa divisão é feita para melhor comparar os três cenários e as duas abordagens, solução MMFA e Redes Convencionais, de acordo com cada um dos critérios. Isso possibilita uma visão mais abrangente das abordagens em relação a cada critério de avaliação.

5.3.1 Consumo de banda

Conforme os dados expressos na Figura 5.7, o consumo de banda quando não há agregação de fluxos se mantém similar para a solução MMFA e a Rede Convencional. O experimento de consumo de banda nesse cenário indica que a solução MMFA não interfere na transmissão em si, apenas interfere na gerência das requisições e regras da tabela de fluxos do *switch*, analisando a necessidade ou não de criar novas conexões com o servidor. Os *picos* e *vales* presentes no gráfico são causados pelo VLC *Media Player*, por conta do controle de fluxo feito na camada de aplicação do VLC. Esse controle é feito na camada de aplicação, pelo fato do protocolo UDP na camada de transporte não possuir controle de fluxo.

No Cenário 2, diferente do experimento de consumo de banda no Cenário 1, é possível observar uma considerável diferença no consumo de banda entre as duas abordagens, MMFA e Rede Convencional. Como pode ser observado na Figura 5.9, para o segundo cenário de experimentos a solução MMFA obteve um considerável ganho em relação a Rede Convencional. Para atender aos dois primeiros clientes, o consumo de banda se mantém constante pois o segundo cliente não necessita de uma nova conexão com o servidor para receber o conteúdo, enquanto na Rede Convencional o consumo de banda dobra pela criação da nova conexão com o servidor, criando um novo fluxo de pacotes. Para atender ao terceiro cliente, a solução MMFA dobra o consumo, pois é um conteúdo diferente que precisa iniciar uma nova conexão com o servidor, enquanto a Rede Convencional triplica o consumo, em relação ao primeiro cliente, pois novamente cria uma conexão com o servidor como o segundo cliente.

Explícito na Figura 5.10 do Cenário 3, a solução MMFA é capaz de manter constante o consumo de banda para atender as requisições de três clientes consumindo o mesmo vídeo. Enquanto a transmissão em Rede Convencional necessita do triplo de banda para atender aos mesmos três clientes com o mesmo vídeo, por conta do tráfego redundante das três conexões estabelecidas com o servidor.

O experimento do consumo de banda indica que além da solução MMFA ser escalável em relação a quantidade de clientes, também é mais escalável quanto a quantidade de conteúdos em vídeo diferenciados que podem ser distribuídos. As transmissões em Rede Convencional tem sua escalabilidade dependente apenas da quantidade de clientes simultâneos, pois todos os clientes iniciam uma nova conexão com o servidor, sobrecarregando o enlace entre o servidor e o *switch*. Já a solução MMFA tem sua escalabilidade dependendo na quantidade de conteúdos diferenciados trafegando simultaneamente (e não diretamente na quantidade de usuários) que são os conteúdos que necessitam de uma nova conexão com o servidor, enquanto todos os clientes que fazem requisições de conteúdos redundantes não consomem recursos do enlace entre servidor e *switch*, por receberem o mesmo fluxo de conteúdo que já estava sendo trafegado. A Tabela 5.4 lista as médias de consumo (em kb/s) para cada usuário em todos os cenários de teste.

Tabela 5.4: Consumo de banda geral

	Cliente 1	σ	**Cliente 2**	σ	**Cliente 3**	σ
Cenário 1 - Convencional	5922	2009	-	-	-	-
Cenário 1 - MMFA	5942	2044	-	-	-	-
Cenário 2 - Convencional	3900	2189	11841	2368	18096	1933
Cenário 2 - MMFA	4182	1928	4830	911	11471	1608
Cenário 3 - Convencional	4894	2275	12120	1283	17790	1739
Cenário 3 - MMFA	4520	1946	5027	1056	5348	1004

Fonte: Elaborado pelo autor (2017)

5.3.2 Tempo de resposta

Conforme listado nas Tabelas 5.1, 5.2 e 5.3, os tempos de resposta para o primeiro cliente requisitante é praticamente sempre o mesmo, tanto na Rede Convencional quanto na solução MMFA, embora a solução MMFA tenha o tempo de instalação da regra de fluxo, o que aumenta ligeiramente o tempo. Contudo, na solução MMFA, o segundo cliente ao requisitar um conteúdo em exibição possui o tempo de resposta relativamente inferior na solução MMFA. Isso se dá pelo fato da requisição do primeiro cliente precisar ir até o servidor e completar uma nova conexão através do protocolo TCP, enquanto um cliente que requisitou um conteúdo redundante tem sua mensagem de requisição barrada no *switch*. Já nas Redes Convencionais, todos os clientes tem suas requisições encaminhadas para o servidor, necessitando aguardar o estabelecimento da conexão.

O Cenário 3 ilustra uma situação na qual é utilizada a solução MMFA. O tempo de resposta do Cliente 1 é similar aos demais tempos dos outros cenários, por ser de uma requisição nova, na qual ainda não possuía tal conteúdo sendo transmitido. Já os tempos para os demais clientes, que requisitaram um conteúdo redundante, segue uma linha crescente: Cliente 2 = 0.0570s e Cliente 3 = 0.1051s. Esse aumento do tempo de resposta é por conta da montagem da regra, que necessita de um laço *for* percorrendo a lista de requisições e agregando as requisições semelhantes. Embora o tempo tenha um crescimento diretamente proporcional a quantidade de usuários, a solução MMFA revelou um desempenho melhor ao tempo de resposta das transmissões em Rede Convencional.

A Tabela 5.5 lista os resultados para tempo de resposta em todos os cenários de teste.

Tabela 5.5: Tempo de resposta geral

	Cliente 1	σ	**Cliente 2**	σ	**Cliente 3**	σ
Cenário 1 - Convencional	1.184720517	0.01112	-	-	-	-
Cenário 1 - MMFA	1.212694582	0.03877	-	-	-	-
Cenário 2 - Convencional	1.194428354	0.02197	1.188755432	0.01402	1.184200267	0.01740
Cenário 2 - MMFA	1.218026603	0.01692	0.067012619	0.04217	1.204483151	0.02411
Cenário 3 - Convencional	1.194428354	0.02197	1.188755432	0.01402	1.184200267	0.01740
Cenário 3 - MMFA	1.246489618	0.03765	0.057044535	0.02182	0.105130000	0.05078

Fonte: Elaborado pelo autor (2017)

5.4 MMFA vs. Requisitos

Conforme descrito na Seção 4.1, a solução deve atender alguns requisitos tanto funcionais como não funcionais. Esses requisitos asseguram o funcionamento e desempenho desejado da solução MMFA: A Tabela 5.6 descreve os requisitos e se não foram atendidos, parcialmente atendidos ou atendidos pela solução MMFA.

Tabela 5.6: Requisitos atendidos pela solução MMFA

Requisito	**Solução MMFA**
RF1: Oferecer serviço de distribuição de conteúdo em vídeo empregando redes SDN de forma a otimizar o uso dos recursos da rede	Atendido, otimizando o consumo de banda no enlace entre o servidor e o *switch*, assim como o tempo de resposta.
RF2: Identificar as condições da rede e usar essa informação para definir as ações a serem tomadas com uma nova requisição	Atendido, pois a solução MMFA é capaz de armazenar os conteúdos em transmissão e agregar os fluxos semelhantes, ou encaminhar a requisição para uma nova conexão com o servidor caso necessário.
RF3: Minimizar o tráfego na rede, eliminando os fluxos redundantes. Porém, sem reduz a qualidade da transmissão para o usuário	Atendido, pois a solução MMFA não interfere na transmissão, e sim encaminha a mesma para diversos clientes.
RNF1: Permitir que um número considerável de usuários receba o conteúdo de forma satisfatória, ou seja, sem atrasos ou perda de quadros	Atendido, pois a solução MMFA não interfere na transmissão e até mesmo reduz o tempo de resposta.

Fonte: Elaborado pelo autor (2017)

Como pode ser observador na Tabela 5.6, os requisitos funcionais e não funcionais propostos foram todos atendidos pela solução MMFA. Embora os requisitos foram atendidos, existem limitações na solução MMFA. Uma dessas limitações é com o

tratamento caso todos os clientes que recebem determinado conteúdo decidam parar de reproduzi-lo antes que o mesmo seja completamente transmitido, pois não existe um pacote que informe o servidor caso um usuário cancele ou interrompa a transmissão. Dessa forma, mesmo sem ter clientes recebendo o conteúdo (tais clientes que interromperam a reprodução), o servidor NVoD continuará a transmissão até o término do conteúdo. Outra limitação é o fato da solução MMFA necessitar de *hardware* de rede (como o *switch*) com suporte ao protocolo OpenFlow.

5.5 Considerações parciais

A utilização dos critérios (Consumo de banda e Tempo de Resposta) é relevante para comparar a solução MMFA com as transmissões em Redes Convencionais, principalmente para a avaliação da escalabilidade do sistema quanto aos usuários simultâneos, assim como a capacidade da solução MMFA em otimizar o consumo de recursos através da agregação de fluxos semelhantes.

Analisando os resultados de uma forma geral, é possível observar que a solução MMFA teve um desempenho consideravelmente melhor em ambos os critérios utilizados para avaliação, ao ser comparada à Rede Convencional. Isso se dá pela otimização do uso dos recursos da rede, solucionando assim, os problemas envolvendo o tráfego redundante da rede e possibilitando que mais clientes recebam o conteúdo simultaneamente, assim como a possibilidade de disponibilizar mais conteúdos para os clientes escolherem.

6 Considerações finais

Com o constante crescimento da demanda de conteúdos em vídeo possibilitada pela evolução das conexões de Internet, diversos problemas e desafios surgiram. O trabalho aqui descrito analisou alguns desses problemas com o objetivo de buscar melhores soluções através de uma abordagem de Redes Definidas por *Software*. Verifica-se que em muitos casos, embora tenham os problemas identificados, nem sempre o solucionam completamente, por conta dos problemas serem bastante diversos.

O NVoD, muito utilizado na transmissão de conteúdo em vídeo, possui características exploradas pela abordagem MMFA através de SDN. A identificação de conteúdo via controlador SDN e agregação de fluxos permite a otimização da infraestrutura de rede. A abordagem MMFA baseia-se no princípio de agregação de fluxos redundantes com o intuito de reduzir o tráfego em enlaces compartilhados por usuários simultâneos, isso, sem afetar a qualidade do serviço.

Os resultados obtidos com o mecanismo MMFA comprovam que a abordagem é mais eficiente quando comparada às Redes Convencionais. Essa eficiência é tanto no tempo de resposta (desde a requisição até o recebimento do vídeo) quanto no consumo dos recursos de rede no enlace compartilhado. Em termos de tempos, o acesso a um conteúdo de vídeo passou de aproximadamente 1.21s das Redes Convencionais para 0.06s na solução MMFA para os clientes subsequentes ao primeiro solicitante. Já em termos de consumo de recursos da rede, mais especificamente a largura de banda no enlace entre servidor NVoD e *switch*, nas Redes Convencionais o consumo ultrapassou 20.000kb/s enquanto na solução MMFA o consumo não chegou a alcançar os 14.000kb/s quando transmitindo para três usuários simultâneos. Assim como a solução MMFA manteve o consumo constante (de cerca de 8.000kb/s) no experimento no terceiro cenário (com três clientes recebendo o mesmo conteúdo), enquanto a Rede Convencional ultrapassou os 20.000kb/s para fornecer o conteúdo para os mesmos três clientes.

Outra contribuição relevante da solução MMFA é o fato de que não exige *software* específico ou alterado nas pontas (servidor e cliente), apenas no controlador do *switch* OpenFlow. Isso permite que os *softwares* sejam alterados, sem afetar o desempenho

da solução.

Ao decorrer do desenvolvimento do trabalho, algumas dificuldades foram encontradas. É possível relatar como a principal dificuldade o entendimento do funcionamento dos sistemas de Vídeo Sob Demanda, assim como a divisão entre TVoD e NVoD através de suas características. Essa divisão não é comumente descrita na literatura de forma clara, o que exigiu uma considerável quantidade de tempo e reflexão para distinguir ambas de forma sucinta e compreensiva. Embora tais dificuldades surgiram ao decorrer do estudo, felizmente foi possível superá-las e cumprir as etapas do cronograma de forma satisfatória.

Para o reconhecimento do vídeo que está sendo requisitado, é necessário um campo de cabeçalho de um pacote (RTP/UDP) que contenha a informação identificando o conteúdo. Porém, com apenas essas ferramentas não é possível identificar o conteúdo. Nesse contexto, é necessário que o usuário encaminhe uma requisição para o servidor com o conteúdo desejado através de um pacote HTTP com método GET. Com isso, a principal dificuldade foi definir um modo de que seja possível o cliente emitir uma requisição para o servidor, para que seja iniciada a transmissão para o cliente através do protocolo RTP, que utiliza o protocolo UDP na camada de transporte. Foram implementadas algumas formas na tentativa de resolução desse problema, como a utilização do servidor de Mídia *Universal Media Server*[1] com o cliente *Kodi*[2], pois desta forma o cliente emite uma requisição para o servidor, porém a transmissão é efetuada através do protocolo TCP na camada de transporte. Essa forma não obteve sucesso pela complexidade do protocolo TCP e seu vasto controle de fluxos, na qual garante o recebimento dos pacotes pelo cliente. Desta forma, a quantidade de tratamentos necessários no módulo a ser implementado no controlador Floodlight é consideravelmente alta. Com isso, optou-se pela criação de um servidor NVoD com o Servidor Apache e VLC *Player*, possibilitando a recepção da requisição e envio através do protocolo RTP na camada de aplicação e UDP na camada de transporte.

Na finalização do trabalho, todas as cinco etapas propostas para tal foram concluídas conforme planejado. Os principais fatores que contribuíram para o sucesso na execução das etapas foi o extenso levantamento bibliográfico sobre os sistemas de transmissão de vídeo, assim como o breve conhecimento prévio envolvendo as Redes Definidas

[1] `http://www.universalmediaserver.com/`
[2] `https://kodi.tv/`

por *Software*, juntamente com o levantamento bibliográfico sobre SDN. Assim como o extenso estudo sobre o controlador OpenFlow Floodlight, para a implementação do módulo gerenciador de fluxos, juntamente com os protocolos envolvidos nas transmissões.

Como trabalhos futuros, é possível realizar mais experimentos na solução MMFA para análise mais profunda de escalabilidade do sistema. Esse experimento pode ser realizado adicionando mais clientes simultâneos, a fim de saturar o enlace de comunicação. Assim como os experimentos para análise mais profunda, fica como trabalho futuro executar a solução MMFA em um *switch* SDN real, em uma rede com clientes reais. Essa execução posteriormente pode ser comparada com os experimentos feitos no simulador de redes Mininet.

É interessante também, o desenvolvimento de um cliente unificado para a escolha e recebimento do conteúdo, pois na implementação da solução MMFA apresentada utiliza-se o navegador Web para a escolha do conteúdo e o *Player* VLC para a recepção do vídeo. Analogamente, o desenvolvimento de um servidor unificado pode ser listado como trabalho futuro, pois na implementação da solução MMFA aqui apresentada utiliza o Servidor Apache para disponibilizar a página Web com os conteúdos e receber as requisições, e o *Player* VLC para transmitir o vídeo para o cliente requisitante. Anseia-se que a abordagem aqui descrita possa auxiliar em trabalhos futuros como uma base de estudo. Esse auxílio pode gerar a solução de outros problemas nas áreas tanto de transmissão de vídeo quanto em outras áreas de transmissão de dados, ou até mesmo a melhoria da abordagem aqui proposta.

6.1 Publicações

O trabalho desenvolvido possibilitou a submissão de dois artigos (conferências):

- (PSCHEIDT; FIORESE, 2017) - "Avaliando recursos de hardware e software em redes definidas por software: Um estudo de caso". Submetido e aceito para o ERAD/RS (Escola Regional de Alto Desempenho), sendo apresentado na cidade de Ijuí/RS no mês de Abril/2017.
- (PSCHEIDT et al., 2018) - "MMFA: Uma abordagem para distribuição de vídeo baseada em Redes Definidas por *Software*". submetido para o *Computer on The Beach*/2018, notificação de aceite em 11/12/2017.

Referências

ADHIKARI, V. K. et al. Unreeling netflix: Understanding and improving multi-cdn movie delivery. In: IEEE. *INFOCOM, 2012 Proceedings IEEE*. [S.l.], 2012. p. 1620–1628.

ALECRIM, E. *O que é Blu-Ray?* 2011. `https://www.infowester.com/blu-ray.php`. Accessed: 2017-04-26.

ALIMI, R.; YANG, Y.; PENNO, R. Application-layer traffic optimization (alto) protocol - rfc 7285. 2014.

ALMEROTH, K. C.; AMMAR, M. H. The use of multicast delivery to provide a scalable and interactive video-on-demand service. *IEEE Journal on Selected Areas in Communications*, IEEE, v. 14, n. 6, p. 1110–1122, 1996.

ANDERSON, D. P. Metascheduling for continuous media. *ACM Transactions on Computer Systems (TOCS)*, ACM, v. 11, n. 3, p. 226–252, 1993.

ANDROUTSELLIS, S.; SPINELLIS, D. A survey of p2p content distribution technologies. *ACM. Comput. Surv*, 2004.

ASOREY-CACHEDA, R. et al. A survey and perspective on nvod systems for satellite networks. In: IEEE. *Satellite and Space Communications, 2007. IWSSC'07. International Workshop on*. [S.l.], 2007. p. 230–233.

BETHANABHOTLA, D.; CAIRE, G.; NEELY, M. J. Adaptive video streaming for wireless networks with multiple users and helpers. *IEEE Transactions on Communications*, IEEE, v. 63, n. 1, p. 268–285, 2015.

BIERSACK, E. W. Where is multicast today? *ACM SIGCOMM Computer Communication Review*, ACM, v. 35, n. 5, p. 83–84, 2005.

CAIN, B. et al. *Internet group management protocol, version 3*. [S.l.], 2002.

CETINKAYA, C.; SAYIT, M. Video-on-demand system architecture with alto-sdn integration. In: IEEE. *Black Sea Conference on Communications and Networking (BlackSeaCom), 2016 IEEE International*. [S.l.], 2016. p. 1–5.

CHEN, Y.-W.; HUANG, K.-T. Multiple videos broadcasting scheme for near video-on-demand services. In: IEEE. *Signal Image Technology and Internet Based Systems, 2008. SITIS'08. IEEE International Conference on*. [S.l.], 2008. p. 52–58.

CHIANG, W.-K.; LI, T.-Y. An extended sdn-based in-network caching service for video on demand. In: IEEE. *Computer Symposium (ICS), 2016 International*. [S.l.], 2016. p. 159–164.

CISCO. Forecast and methodology, 2014-2019 white paper. *Cisco Technical Report*, 2015.

DAN, A.; SITARAM, D.; SHAHABUDDIN, P. Dynamic batching policies for an on-demand video server. *Multimedia systems*, Springer, v. 4, n. 3, p. 112–121, 1996.

DHAGE, S. N.; PATIL, S. K.; MESHRAM, B. Survey on: Interactive video-on-demand (vod) systems. In: IEEE. *Circuits, Systems, Communication and Information Technology Applications (CSCITA), 2014 International Conference on*. [S.l.], 2014. p. 435–440.

DIORIO, R. F.; TIMÓTEO, V. S. A platform for multimedia traffic forwarding in software defined networks. In: ACM. *Proceedings of the 21st Brazilian Symposium on Multimedia and the Web*. [S.l.], 2015. p. 177–180.

EAGER, D. L.; VERNON, M. K.; ZAHORJAN, J. Bandwidth skimming: A technique for cost-effective video on demand. In: INTERNATIONAL SOCIETY FOR OPTICS AND PHOTONICS. *Electronic Imaging*. [S.l.], 2000. p. 206–215.

EGILMEZ, H. E.; CIVANLAR, S.; TEKALP, A. M. An optimization framework for qos-enabled adaptive video streaming over openflow networks. *IEEE Transactions on Multimedia*, IEEE, v. 15, n. 3, p. 710–715, 2013.

FOROUZAN, B. A.; MOSHARRAF, F. *Redes de computadores: uma abordagem top-down*. [S.l.]: AMGH Editora, 2013.

FOSTER, I.; IAMNITCHI, A. On death, taxes, and the convergence of peer-to-peer and grid computing. *Peer-to-Peer Systems II*, Springer, p. 118–128, 2003.

GALLO, D. S. Arquitetura de iptv com suporte à apresentação deslocada no tempo baseada em distribuição peer-to-peer. *Dissertação (Mestrado) - Universidade de São Paulo*, 2009.

GEORGOPOULOS, P. et al. Using software defined networking to enhance the delivery of video-on-demand. *Computer Communications*, Elsevier, v. 69, p. 79–87, 2015.

GEORGOPOULOS, P. et al. Cache as a service: Leveraging sdn to efficiently and transparently support video-on-demand on the last mile. In: IEEE. *Computer Communication and Networks (ICCCN), 2014 23rd International Conference on*. [S.l.], 2014. p. 1–9.

GLINIS, S. *VCR's: The end of TV as ephemera*. Tese (Doutorado) — The University of Wisconsin-Milwaukee, 2015.

GNAGY, M. R. et al. *System and method for generalized URL-rewriting*. [S.l.]: Google Patents, 2006. US Patent 7,058,633.

GOYA, W. A. *Uma solução para o desenvolvimento de aplicações distribuídas visando o gerenciamento automático de recursos no cenário de computação em nuvem.* Tese (Doutorado) — Universidade de São Paulo, 2014.

HARVEY, L. In search of a rate control policy for xtp: unicast & multicast. 1999.

HOSSFELD, T. et al. Internet video delivery in youtube: From traffic measurements to quality of experience. Springer, p. 264–301, 2013.

HU, F.; HAO, Q.; BAO, K. A survey on software-defined network and openflow: From concept to implementation. *IEEE Communications Surveys & Tutorials*, IEEE, v. 16, n. 4, p. 2181–2206, 2014.

HUA, K. A.; SHEU, S. Skyscraper broadcasting: A new broadcasting scheme for metropolitan video-on-demand systems. In: ACM. *ACM SIGCOMM Computer Communication Review*. [S.l.], 1997. v. 27, n. 4, p. 89–100.

JAIN, R. *Video Streaming over Mobile Networks: Issues, Challenges, and Opportunities*. 2011. Keynote at National Workshop on Wireless Communications and Mobile Networks (WCMN-2011).

JUHN, L.-S.; TSENG, L.-M. Harmonic broadcasting for video-on-demand service. *IEEE transactions on broadcasting*, IEEE, v. 43, n. 3, p. 268–271, 1997.

JULURI, P.; TAMARAPALLI, V.; MEDHI, D. Measurement of quality of experience of video-on-demand services: A survey. *IEEE Communications Surveys & Tutorials*, IEEE, v. 18, n. 1, p. 401–418, 2016.

JUNIOR, J. S. da S. *Evolução da TV*. 2016. `http://mundoeducacao.bol.uol.com.br/curiosidades/evolucao-tv.htm`. Accessed: 2017-04-26.

KANGASHARJU, J.; ROSS, K. W.; ROBERTS, J. W. Performance evaluation of redirection schemes in content distribution networks. *Computer Communications*, Elsevier, v. 24, n. 2, p. 207–214, 2001.

KIM, H.; FEAMSTER, N. Improving network management with software defined networking. *IEEE Communications Magazine*, IEEE, v. 51, n. 2, p. 114–119, 2013.

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

KWAK, H. G.; YOO, J. J.; HONG, J. W. A hybrid approach to broadcasting method in near video on demand service for bandwidth-restricted clients. In: IEEE. *Advanced Communication Technology (ICACT), 2010 The 12th International Conference on*. [S.l.], 2010. v. 1, p. 591–595.

LARA, A.; KOLASANI, A.; RAMAMURTHY, B. Network innovation using openflow: A survey. *IEEE communications surveys & tutorials*, IEEE, v. 16, n. 1, p. 493–512, 2014.

LEE, J. Y. On a unified architecture for video-on-demand services. *IEEE Transactions on Multimedia*, IEEE, v. 4, n. 1, p. 38–47, 2002.

LINUXFOUNDATION. *OpenDayLight: Platform Overview*. 2017. `https://www.opendaylight.org/platform-overview`. Accessed: 2017-05-27.

MA, H.; SHIN, K. G. Multicast video-on-demand services. *ACM SIGCOMM Computer Communication Review*, ACM, v. 32, n. 1, p. 31–43, 2002.

MAJUMDA, A. et al. Multicast and unicast real-time video streaming over wireless lans. *IEEE Transactions on Circuits and Systems for Video Technology*, IEEE, v. 12, n. 6, p. 524–534, 2002.

MASOUDI, R.; GHAFFARI, A. Software defined networks: A survey. *Journal of Network and Computer Applications*, Elsevier, v. 67, p. 1–25, 2016.

MOLISCH, A. F. et al. Caching eliminates the wireless bottleneck in video aware wireless networks. *Advances in Electrical Engineering*, Hindawi Publishing Corporation, v. 2014, 2014.

MYLER, H. R. Bandwidth issues in advanced video quality and delivery. In: IEEE. *Information and Communications Technology, 2007. ICICT 2007. ITI 5th International Conference on*. [S.l.], 2007. p. 205–208.

NAYFEH, K. K.; SARHAN, N. J. Design and analysis of scalable and interactive near video-on-demand systems. In: IEEE. *Multimedia and Expo (ICME), 2013 IEEE International Conference on*. [S.l.], 2013. p. 1–6.

NAYFEH, K. K.; SARHAN, N. J. A scalable solution for interactive near video-on-demand systems. *IEEE Transactions on Circuits and Systems for Video Technology*, IEEE, v. 26, n. 10, p. 1907–1916, 2016.

NGUYEN, X.-N. et al. Rules placement problem in openflow networks: a survey. *IEEE Communications Surveys & Tutorials*, IEEE, v. 18, n. 2, p. 1273–1286, 2016.

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

PATHAN, A.-M. K.; BUYYA, R. A taxonomy and survey of content delivery networks. *Grid Computing and Distributed Systems Laboratory, University of Melbourne, Technical Report*, p. 4, 2007.

PFUTZENREUTER, E.; FRIEDRICH, L. F. Avaliação de desempenho do protocolo sctp no linux. *IEEE Latin America Transactions*, IEEE, p. 171–176, 2007.

PSCHEIDT, N. S.; FIORESE, A. Avaliando recursos de hardware e software em redes definidas por software: Um estudo de caso. 2017.

PSCHEIDT, N. S. et al. Mmfa: Uma abordagem para dristribuição de vídeo baseada em redes definidas por *Software*. 2018.

RAO, S. *SDN Series Part Three: NOX, the original OpenFlow Controller*. 2014. `https://thenewstack.io/sdn-series-part-iii-nox-the-original-openflow-controller/`. Accessed: 2017-05-27.

RAWAT, D. B.; REDDY, S. R. Software defined networking architecture, security and energy efficiency: A survey. *IEEE Communications Surveys & Tutorials*, IEEE, 2016.

REN, T.; XU, Y. Analysis of the new features of openflow 1.4. In: ATLANTIS PRESS. *2nd International Conference on Information, Electronics and Computer*. [S.l.], 2014. p. 73–77.

ROUSE, M. *Video On Demand (VoD)*. 2007. `https://www.searchtelecom.techtarget.com/definition/video-on-demand/`. Accessed: 2017-02-27.

SANTOS, A. H. O. *Redes de Comunicação de Dados: Unicast, Multicast e Broadcast*. 2016. `https://www.uniaogeek.com.br/redes-de-comunicacao-de-dados-unicast-multicast-e-broadcast/`. Accessed: 2017-05-07.

SCHULZRINNE, H. et al. Rtp: A transport protocol for real-time applications (rfc 1889). *Network Working Group*, 1996.

SCHULZRINNE, H.; RAO, A.; LANPHIER, R. Rfc 2326: real time streaming protocol. *Disponível em* `http://www.ietf.org/rfc/rfc2326.txt`, 1998.

SHAIKH, A.; TEWARI, R.; AGRAWAL, M. On the effectiveness of dns-based server selection. In: IEEE. *INFOCOM 2001. Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*. [S.l.], 2001. v. 3, p. 1801–1810.

SHEN, Z. et al. Peer-to-peer media streaming: Insights and new developments. *Proceedings of the IEEE*, IEEE, v. 99, n. 12, p. 2089–2109, 2011.

SHIRAISHI, Y. History of home videotape recorder development. *SMPTE Journal*, SMPTE, v. 94, n. 12, p. 1257–1263, 1985.

STEWART, R. et al. Rfc 2960. *Stream control transmission protocol*, 2000.

SU, A.-J. et al. Drafting behind akamai: Inferring network conditions based on cdn redirections. *IEEE/ACM Transactions on Networking (TON)*, IEEE Press, v. 17, n. 6, p. 1752–1765, 2009.

TAN, B.; MASSOULIÉ, L. Optimal content placement for peer-to-peer video-on-demand systems. *IEEE/ACM Transactions on Networking (TON)*, IEEE Press, v. 21, n. 2, p. 566–579, 2013.

TANG, S.; HUA, B.; WANG, D. Realizing video streaming multicast over sdn networks. In: IEEE. *Communications and Networking in China (CHINACOM), 2014 9th International Conference on*. [S.l.], 2014. p. 90–95.

VALE, M.; COSTA, D.; JR, N. A. Internet: Histórico, evolução e gestão. *CBPF NT005/01*, 2001.

WAMSER, F. et al. *YoMo: A YouTube Application Comfort Monitoring Tool*. [S.l.]: March, 2010.

WEN, G.; LONGSHE, H.; QIANG, F. Recent advances in peer-to-peer media streaming systems. *China Communications*, v. 10, p. 52–57, 2006.

WU, P. et al. Transition from ipv4 to ipv6: A state-of-the-art survey. *IEEE Communications Surveys & Tutorials*, IEEE, v. 15, n. 3, p. 1407–1424, 2013.

YAHAV, U.; RABINOVITZ, A. *Multicast Enables Efficient Streaming of Media over IP*. 2002. `http://www.eetimes.com/document.asp?doc_id=1206337`. Accessed: 2017-07-05.

YOSHIHISA, T.; NISHIO, S. A division-based broadcasting method considering channel bandwidths for nvod services. *IEEE Transactions on Broadcasting*, IEEE, v. 59, n. 1, p. 62–71, 2013.

YU, H.-F. Efficient seamless channel transition for near video-on-demand with small receiving bandwidth. *Journal of Information Science & Engineering*, v. 32, n. 5, 2016.

YU, T.-F.; WANG, K.; HSU, Y.-H. Adaptive routing for video streaming with qos support over sdn networks. In: IEEE. *Information Networking (ICOIN), 2015 International Conference on*. [S.l.], 2015. p. 318–323.

ZHANG, X.; HASSANEIN, H. Video on-demand streaming on the internet—a survey. In: IEEE. *Communications (QBSC), 2010 25th Biennial Symposium on*. [S.l.], 2010. p. 88–91.

ZIMMERMANN, H. Osi reference model–the iso model of architecture for open systems interconnection. *IEEE Transactions on communications*, IEEE, v. 28, n. 4, p. 425–432, 1980.

A Apêndice: Códigos

Os códigos desenvolvidos para a solução MMFA estão disponibilizados no Github através do link: `https://github.com/Nadyan/SDN_VideoTransmission`. O repositório contém:

- O módulo do controlador Floodlight (aggregatorUDP.java) e as classes HTTP (HTTP.java e HTTPMethod.java) em:
  `http://bit.ly/2zUjznW`;
- Os *scripts Shell* para transmissão e pagina Web estão localizados em:
  `http://bit.ly/2zFsPcz`; e
- O *Script* para mensurar o consumo de banda está disponível em:
  `http://bit.ly/2iaxk7t`.
- Os *scripts* em Python para a criação das topologias no Mininet:
  `http://bit.ly/2hpGjkg`

- Também está disponibilizado no Github o código do controlador Floodlight:
  `https://github.com/floodlight/floodlight`

A.1 Módulo MMFA

O código do módulo aqui apresentado deve ser adicionado no projeto do controlador Floodlight, para ser executado com os demais módulos.

```
package net.floodlightcontroller.aggregator;
import java.util.ArrayList;
import java.util.Collection;
import java.util.HashSet;
import java.util.List;
import java.util.Map;
import java.util.Set;
import org.projectfloodlight.openflow.protocol.OFFactories;
import org.projectfloodlight.openflow.protocol.OFFactory;
```

```
import org.projectfloodlight.openflow.protocol.OFFlowAdd;
import org.projectfloodlight.openflow.protocol.OFFlowModFlags;
import org.projectfloodlight.openflow.protocol.OFMessage;
import org.projectfloodlight.openflow.protocol.OFType;
import org.projectfloodlight.openflow.protocol.OFVersion;
import org.projectfloodlight.openflow.protocol.action.OFAction;
import org.projectfloodlight.openflow.protocol.action.OFActionOutput;
import org.projectfloodlight.openflow.protocol.action.OFActionSetField;
import org.projectfloodlight.openflow.protocol.action.OFActions;
import org.projectfloodlight.openflow.protocol.match.Match;
import org.projectfloodlight.openflow.protocol.match.MatchField;
import org.projectfloodlight.openflow.protocol.oxm.OFOxms;
import org.projectfloodlight.openflow.types.EthType;
import org.projectfloodlight.openflow.types.IPv4Address;
import org.projectfloodlight.openflow.types.IpProtocol;
import org.projectfloodlight.openflow.types.MacAddress;
import org.projectfloodlight.openflow.types.OFBufferId;
import org.projectfloodlight.openflow.types.OFPort;
import org.projectfloodlight.openflow.types.TransportPort;
import org.projectfloodlight.openflow.types.U64;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import net.floodlightcontroller.core.FloodlightContext;
import net.floodlightcontroller.core.IFloodlightProviderService;
import net.floodlightcontroller.core.IOFMessageListener;
import net.floodlightcontroller.core.IOFSwitch;
import net.floodlightcontroller.core.module.FloodlightModuleContext;
import net.floodlightcontroller.core.module.FloodlightModuleException;
import net.floodlightcontroller.core.module.IFloodlightModule;
import net.floodlightcontroller.core.module.IFloodlightService;
import net.floodlightcontroller.packet.Data;
import net.floodlightcontroller.packet.Ethernet;
import net.floodlightcontroller.packet.HTTP;
import net.floodlightcontroller.packet.HTTPMethod;
import net.floodlightcontroller.packet.IPv4;
import net.floodlightcontroller.packet.TCP;
import net.floodlightcontroller.util.FlowModUtils;

public class AggregatorUDP implements IFloodlightModule, IOFMessageListener {
    /*
     * UDESC - Universidade do Estado de Santa Catarina
     * Bacharelado em Ciencia da Computacao
     * Nadyan Suriel Pscheidt
     *
     * Modulo responsavel pela identificacao de requests,
     * identificacao de fluxos redundantes,
     * agregacao de fluxos redundantes
     * e criacao de regras na tabela de fluxos.
     */
    private IFloodlightProviderService floodlightProvider;
    private static Logger logger;
```

```
public static final TransportPort UDP_PORT = TransportPort.of(5004);
public static final TransportPort TCP_PORT = TransportPort.of(5001);
/*  Listas de requisicoes */
protected List<Request> requests;
protected List<String> videosInTraffic;
/* Enderecos */
private IPv4Address ipUser1 = IPv4Address.of("10.0.0.2");
private IPv4Address ipUser2 = IPv4Address.of("10.0.0.3");
private IPv4Address ipUser3 = IPv4Address.of("10.0.0.4");
/* Portas */
private int u1Port = 1;
private int u2Port = 2;
private int u3Port = 3;
private int userPort = -1;

@Override
public String getName() {
    return AggregatorUDP.class.getSimpleName();
}
@Override
public boolean isCallbackOrderingPrereq(OFType type, String name) {
    return false;
}
@Override
public boolean isCallbackOrderingPostreq(OFType type, String name) {
    /* Deve ser executado antes do modulo Forwarding */

    if (type.equals(OFType.PACKET_IN) && name.equals("forwarding")) {
        return true;
    } else {
        return false;
    }
}
@Override
public Collection<Class<? extends IFloodlightService>> getModuleServices() {
    return null;
}
@Override
public Map<Class<? extends IFloodlightService>, IFloodlightService> getServiceImpls() {
    return null;
}
@Override
public Collection<Class<? extends IFloodlightService>> getModuleDependencies() {
    return null;
}
@Override
public void init(FloodlightModuleContext context) throws FloodlightModuleException {
floodlightProvider = context.getServiceImpl(IFloodlightProviderService.class);
logger = LoggerFactory.getLogger(AggregatorTCP.class);

/* Criacao das listas */
```

```
requests = new ArrayList<Request>(); // Lista com as informacoes dos clientes
videosInTraffic = new ArrayList<String>(); // Lista com os videos trafegando
}
@Override
public void startUp(FloodlightModuleContext context) throws FloodlightModuleException {
    floodlightProvider.addOFMessageListener(OFType.PACKET_IN, this);
}
@Override
public Command receive(IOFSwitch sw, OFMessage msg, FloodlightContext cntx) {

    Ethernet eth = IFloodlightProviderService.bcStore.get(cntx,
    IFloodlightProviderService.CONTEXT_PI_PAYLOAD);
    MacAddress srcMac = eth.getSourceMACAddress();              // MAC cliente
    if (eth.getEtherType() == EthType.IPv4) {
        /* Pacote IPv4 */
        IPv4 ipv4 = (IPv4) eth.getPayload();
        IPv4Address srcIp = ipv4.getSourceAddress();            // IP cliente
        IPv4Address dstIp = ipv4.getDestinationAddress();       // IP server

        /* Define qual porta do switch pertence ao usuario */
        if (srcIp.equals(ipUser1) == true) {
            userPort = u1Port;
        } else if (srcIp.equals(ipUser2) == true) {
            userPort = u2Port;
        } else if (srcIp.equals(ipUser3) == true) {
            userPort = u3Port;
        }
        if (ipv4.getProtocol() == IpProtocol.TCP) {
            /* Pacote TCP com a requisicao*/
            TCP tcp = (TCP) ipv4.getPayload();


            /* Processa a requisicao */
            /* Pega header HTTP do payload do TCP */
            Data dt = (Data) tcp.getPayload();
            byte[] txt = dt.getData();
            String headerHttp = new String(txt);
            /* Divisao do header em 3 partes: Metodo, URI e resto */
            String arr[] = headerHttp.split(" ", 3);
            String method = arr[0];

            HTTP http = new HTTP();

            /* Identificacao do metodo da requisicao HTTP */
            if (method.compareTo("HEAD") == 0) {
                http.setHTTPMethod(HTTPMethod.HEAD);
            } else if (method.compareTo("GET") == 0) {
                http.setHTTPMethod(HTTPMethod.GET);
            } else {
                http.setHTTPMethod(HTTPMethod.NONE);
            }
```

```
if(http.getHTTPMethod() == HTTPMethod.GET) {
    /* Pacote GET com a requisicao */
    String videoId = arr[1].substring(arr[1].lastIndexOf("/") + 1);
    if (videoId.equals("favicon.ico") || videoId.isEmpty()) {
        /* Se for um GET de icone ou outro que nao seja video
         * que nao interessa */
        return Command.CONTINUE;
    }
    if (videosInTraffic.contains(videoId) == false) {
        /* Se ainda nao havia uma transferencia do video requisitado */
        Request novoRequest = new Request(videoId, srcMac, srcIp, userPort);
        videosInTraffic.add(videoId);
        requests.add(novoRequest);
        logger.info("Requisicao nova para " + videoId + " de " + srcIp);
        return Command.CONTINUE;
    } else {
        /*  Se o video requisitado ja estava sendo transmitido,
         *  agrega os fluxos,
         *  criando uma regra que recebe o fluxo do primeiro
         *  usuario e encaminha para
         *  ele e os demais requisitantes do detrminado conteudo.
         */
        IPv4Address originalIp = srcIp;
        boolean getOriginal = false;
        Request novoRequest = new Request(videoId, srcMac, srcIp, userPort);
        requests.add(novoRequest);
        /* Inicio da montagem da regra.
         * A regra possui as informacoes (MAC, IP e porta)
         * dos clientes requisitantes.
         * Com isso, o fluxo UDP destinado ao primeiro usuario e
         * direcionado para todas as portas especificadas como OFActionOutput
         * (primeiro usuario e demais requisitantes).
         */
        OFFactory my13Factory = OFFactories.getFactory(OFVersion.OF_13);
        OFActions actions = my13Factory.actions();
        OFOxms oxms = my13Factory.oxms();
        /* Lista que armazena as caracteristicas da regra */
        List<OFAction> actionsTo = new ArrayList<OFAction>();
        /* Lista nula para fazer drop de pacotes */
        List<OFAction> actionsNull = new ArrayList<OFAction>();
        /* Criacao da regra com um laco For que percorre a lista de
         * requests e verifica todos os de mesmo videoId */
        for (int i = 0; i < requests.size(); i++) {
            if (requests.get(i).getVideoId().equals(videoId)) {
                /* Se for do mesmo videoId */
                if (getOriginal == false) {
                    /* Armazena o IP do primeiro requisitante para
                     * criacao do match,
                     * que no caso e o usuario na qual o servidor
                     * esta enviando o conteudo */
                    originalIp = requests.get(i).getIpAddress();
```

```
                                getOriginal = true;
                            }
                            OFActionSetField setDstIp = actions.buildSetField()
                                                        .setField(oxms.buildIpv4Dst()
                                                        .setValue(requests.get(i)
                                                                  .getIpAddress())
                                                        .build()).build();
                            OFActionSetField setDstMac = actions.buildSetField()
                                                         .setField(oxms.buildEthDst()
                                                         .setValue(requests.get(i)
                                                                   .getMacAddress())
                                                         .build()).build();
                            OFActionOutput outputUser = actions.output(OFPort.of(requests
                                                    .get(i).getPort()),Integer.MAX_VALUE);
                            actionsTo.add(setDstIp);
                            actionsTo.add(setDstMac);
                            actionsTo.add(outputUser);
                        }
                    }
                    OFFlowAdd flowTo = fluxoUDP(createMatch(sw,UDP_PORT,cntx,
                                                            originalIp,dstIp),
                                                            my13Factory,
                                                            actionsTo,
                                                            FlowModUtils.PRIORITY_HIGH);
                    if (sw.write(flowTo)) {
                        /* Se obter sucesso na escrita da regra */
                        logger.info("Fluxos agregados para o request " + videoId);
                        /* Cria regra de drop de pacotes para o usuario
                         * parar de reenviar gets por falta de resposta */
                        OFFlowAdd flowNull = fluxoNullUDP(createMatchNull(sw, TCP_PORT,
                                        cntx, srcIp, dstIp), my13Factory, actionsNull);
                        sw.write(flowNull);
                        /*  Barra o pacote para ele nao seguir para o servidor,
                         *  pois ja sera transmitido para o usuario novo pela
                         *  regra flowTo que foi instalada.
                         */
                        return Command.STOP;
                    } else {
                        /* Se a escrita da regra falhar */
                        logger.info("ERRO: Falha na agregacao de fluxos para o request "
                                +videoId+ ",o conteudo sera transmitido sem agregacao!");
                        /* Como nao conseguiu criar a regra, transmite normalmente
                         * sem agragacao.
                         * O comando Continue ira enviar o request para o servidor */
                        return Command.CONTINUE;
                    }
                }
            }
        }
    }
    return Command.CONTINUE;
```

```
}
private Match createMatch(IOFSwitch sw, TransportPort port, FloodlightContext cntx,
                                                    IPv4Address dst, IPv4Address src) {
    /* Criacao do Match */
    Match.Builder mb = sw.getOFFactory().buildMatch();
    mb.setExact(MatchField.ETH_TYPE, EthType.IPv4)
                .setExact(MatchField.IPV4_SRC, src)
                .setExact(MatchField.IPV4_DST, dst)
                .setExact(MatchField.IP_PROTO, IpProtocol.UDP)
                .setExact(MatchField.UDP_DST, port);
    return mb.build();
}
private Match createMatchNull(IOFSwitch sw, TransportPort port, FloodlightContext cntx,
                                            IPv4Address srcIp, IPv4Address dstIp) {
    /* Criacao do match para dar drop nos pacotes de requisicao vindos
     * do segundo usuario apos o fluxo agregado ser criado
     */
    Match.Builder mb = sw.getOFFactory().buildMatch();
    mb.setExact(MatchField.ETH_TYPE, EthType.IPv4)
      .setExact(MatchField.IPV4_SRC, srcIp)              // IP Cliente
      .setExact(MatchField.IPV4_DST, dstIp)              // IP Server
      .setExact(MatchField.IP_PROTO, IpProtocol.TCP);
    return mb.build();
}
private OFFlowAdd fluxoUDP(Match match, OFFactory myFactory, List<OFAction> actions,
                                                                    int prioridade) {
    /* Criacao da regra */
    Set<OFFlowModFlags> flags = new HashSet<>();
    flags.add(OFFlowModFlags.SEND_FLOW_REM);
    OFFlowAdd flowToUDP = myFactory
                            .buildFlowAdd()
                            .setFlags(flags)
                            .setActions(actions)
                            .setBufferId(OFBufferId.NO_BUFFER)
                            .setIdleTimeout(60)
                            .setHardTimeout(0)
                            .setMatch(match)
                            .setCookie(U64.of(1L << 59))
                            .setPriority(prioridade)
                            .build();
    return flowToUDP;
}
private OFFlowAdd fluxoNullUDP(Match match, OFFactory myFactory, List<OFAction> actions){
    /* Montagem do fluxo para dar drop nos pacotes */
    Set<OFFlowModFlags> flags = new HashSet<>();
    flags.add(OFFlowModFlags.SEND_FLOW_REM);
    OFFlowAdd flowNull = myFactory
                        .buildFlowAdd()
                        .setActions(actions)
                        .setBufferId(OFBufferId.NO_BUFFER)
                        .setIdleTimeout(120)
```

```
                                .setHardTimeout(0)
                                .setMatch(match)
                                .setCookie(U64.of(1L << 59))
                                .setPriority(FlowModUtils.PRIORITY_HIGH)
                                .build();
            return flowNull;
        }
}
```

A.2 Classes HTTP

As classes HTTP devem ser adicionadas ao código do controlador Floodlight para serem utilizadas pelo módulo MMFA.

A.2.1 HTTP

```
package net.floodlightcontroller.packet;
import java.nio.ByteBuffer;
import java.util.Arrays;

public class HTTP extends BasePacket {
        private HTTPMethod method;
        private String methodRead;

        public HTTPMethod getHTTPMethod() {
                return method;
        }
        public void setHTTPMethod(HTTPMethod method) {
                this.method = method;
        }
}
```

A.2.2 HTTPMethod

```
package net.floodlightcontroller.packet;

public enum HTTPMethod {
        GET(1, "GET"),
        HEAD(2, "HEAD"),
        POST(3, "POST"),
        PUT(4, "PUT"),
        DELETE(5, "DELETE"),
        CONNECT(6, "CONNECT"),
        OPTIONS(7, "OPTIONS"),
        TRACE(8, "TRACE"),
        NONE(9, "NOT HTTP METHOD");

        protected int type;
        protected String label;

        private HTTPMethod(int type, String label) {
                this.type = type;
                this.label = label;
        }
        public static HTTPMethod getType(String value) {
                switch(value) {
                        case "GET":
                                return GET;
                        case "HEAD":
                                return HEAD;
                        case "POST":
                                return POST;
                        case "PUT":
                                return PUT;
                        case "DELETE":
                                return DELETE;
                        case "CONNECT":
                                return CONNECT;
                        case "OPTIONS":
                                return OPTIONS;
                        case "TRACE":
                                return TRACE;
                        default:
                                return NONE;
                }
        }
        public String getLabel() {
                return this.label;
        }
        public int getIntType() {
                return this.type;
        }
}
```

A.3 Criação de Topologias

A.3.1 Topologia 1

```
#!/usr/bin/python

import sys
from mininet.net import Mininet
from mininet.node import OVSSwitch,Controller,RemoteController
from mininet.topolib import TreeTopo
from mininet.log import setLogLevel, info
from mininet.cli import CLI
setLogLevel( 'info' )
def emptyNet(nsw = 1):

    net = Mininet(topo = None, build = False)
    info( '*** Init Experimento 1\n\n' )
    info( '*** Adding controller\n' )
    controller1=net.addController('c0',controller=RemoteController,ip='127.0.0.1',port=6653)
    info( '*** Adding hosts\n' )
    user1 = net.addHost('user1', ip = '10.0.0.2', mac = '00:00:00:00:00:02')
    server1 = net.addHost('server1', ip = '10.0.0.1', mac = '00:00:00:00:00:01')
    info( '*** Adding switch\n' )
    switch1 = net.addSwitch( 'switch1' )
    switch1.start([controller1])
    info( '\n*** Creating links\n' )
    net.addLink( user1, switch1 )
    net.addLink( server1, switch1 )
    info( '*** Starting network\n')
    net.start()
    info( '*** Opening xterm\n')
    net.startTerms()
    info( '*** Running CLI\n' )
    CLI( net )
    info( '*** Stopping network' )
    net.stop()

if __name__ == '__main__':
    setLogLevel( 'info' )
    emptyNet()
```

A.3.2 Topologia 2

```
#!/usr/bin/python

import sys
from mininet.net import Mininet
from mininet.node import OVSSwitch,Controller,RemoteController
from mininet.topolib import TreeTopo
from mininet.log import setLogLevel, info
from mininet.cli import CLI
setLogLevel( 'info' )

def emptyNet(nsw = 1):
    net = Mininet(topo = None, build = False)
    info( '*** Init Experimento 2\n\n' )
    info( '*** Adding controller\n' )
    c0 = net.addController('c0', controller = RemoteController, ip = '127.0.0.1', port = 6653)
    info( '*** Adding hosts\n' )
    server1 = net.addHost('server1', ip = '10.0.0.1', mac = '00:00:00:00:00:01')
    user1 = net.addHost('user1', ip = '10.0.0.2', mac = '00:00:00:00:00:02')
    user2 = net.addHost('user2', ip = '10.0.0.3', mac = '00:00:00:00:00:03')
    user3 = net.addHost('user3', ip = '10.0.0.4', mac = '00:00:00:00:00:04')
    info( '*** Adding switch\n' )
    switch1 = net.addSwitch( 'switch1' )
    switch1.start([c0])
    info( '\n*** Creating links\n' )
    net.addLink(user1, switch1)
    net.addLink(user2, switch1)
    net.addLink(user3, switch1)
    net.addLink(server1, switch1)
    info( '*** Starting network\n')
    net.start()
    info( '*** Opening xterm\n')
    net.startTerms()
    info( '*** Running CLI\n' )
    CLI( net )
    info( '*** Stopping network' )
    net.stop()

if __name__ == '__main__':
    setLogLevel( 'info' )
    emptyNet()
```

A.4 Interface Web

Interface disponibilizada pelo Servidor Apache para o cliente escolher o conteúdo.

```
<!DOCTYPE html>

<html>
    <body>
        <h1>Escolha o video para iniciar a transmissao!</h1>
        <form method="GET" action="/cgi-bin/trailer.sh">
                <button type="submit">Trailer.mp4</button>
        </form>
        </br>
        <form method="GET" action="/cgi-bin/coelho.sh">
                <button type="submit">Coelho.mp4</button>
        </form>
    </body>
</html>
```

A.5 *Scripts* para transmissão

Scripts utilizados pelo servidor para iniciarem a transmissão do conteúdo escolhido para o cliente requisitante.

A.5.1 trailer.sh

```
#!/bin/bash
# Script para a transmissao do video trailer.mp4

# Cria a transmissao atraves do VLC utilizando RTP
COMANDO="vlc -vvv videos/trailer.mp4 --sout '#rtp{mux=ts,dst='$REMOTE_ADDR',sdp=sap}'"

# Executa
eval $COMANDO
```

A.5.2 coelho.sh

```
#!/bin/bash
# Script para a transmissao do video coelho.mp4

# Cria a transmissao atraves do VLC utilizando RTP
COMANDO="vlc -vvv videos/coelho.mp4 --sout '#rtp{mux=ts,dst='$REMOTE_ADDR',sdp=sap}'"

# Executa
eval $COMANDO
```

A.6 *Script* para mensuração do consumo de banda

```
#!/bin/bash
# Script shell para medir o throughput no enlace
# Switch -> Server

# Habilitar as estatisticas:
curl http://127.0.0.1:8080/wm/statistics/config/enable/json -X POST

# Topo1 -> porta server = 2
# Topo2 -> porta server = 4
PORTA=4

for i in {0..119}
do
    # Executa 120 medicoes, uma a cada segundo
    curl http://127.0.0.1:8080/wm/statistics/bandwidth/
                    00:00:00:00:00:00:00:01/$PORTA/json -X GET
    sleep 1 # espera 1 segundo
    echo " \n\n $i !!"
done
```