UNIVERSIDADE FEDERAL DO PARANÁ

CARLOS EDUARDO MAFFINI SANTOS

UM NOVO MÉTODO DE GERÊNCIA ATIVA DE FILAS PARA FLUXOS DASH AO VIVO

CURITIBA

2022

CARLOS EDUARDO MAFFINI SANTOS

UM NOVO MÉTODO DE GERÊNCIA ATIVA DE FILAS PARA FLUXOS DASH AO VIVO

Tese apresentada ao Programa de Pós-Graduação em Engenharia Elétrica, Área de Concentração Telecomunicações, Departamento de Engenharia Elétrica, Setor de Tecnologia, Universidade Federal do Paraná, como parte das exigências para obtenção do título de Doutor em Engenharia Elétrica.

Orientador: Prof. Dr. Carlos Marcelo Pedroso

CURITIBA

2022

DADOS INTERNACIONAIS DE CATALOGAÇÃO NA PUBLICAÇÃO (CIP)
UNIVERSIDADE FEDERAL DO PARANÁ
SISTEMA DE BIBLIOTECAS – BIBLIOTECA DE CIÊNCIA E TECNOLOGIA

Santos, Carlos Eduardo Maffini
Um novo método de gerência ativa de filas para fluxos DASH ao vivo / Carlos Eduardo Maffini Santos. – Curitiba, 2022.
1 recurso on-line : PDF.

Tese (Doutorado) - Universidade Federal do Paraná, Setor de Tecnologia, Programa de Pós-Graduação em Engenharia Elétrica.

Orientador: Carlos Marcelo Pedroso

1. Tecnologia streaming (Telecomunicação). 2. Memória de longo prazo. 3. Gravações de vídeo - reprodução. I. Universidade Federal do Paraná. II. Programa de Pós-Graduação em Engenharia Elétrica. III. Pedroso, Carlos Marcelo. IV. Título.

Bibliotecário: Elias Barbosa da Silva CRB-9/1894

MINISTÉRIO DA EDUCAÇÃO
SETOR DE TECNOLOGIA
UNIVERSIDADE FEDERAL DO PARANÁ
PRÓ-REITORIA DE PESQUISA E PÓS-GRADUAÇÃO
PROGRAMA DE PÓS-GRADUAÇÃO ENGENHARIA ELÉTRICA - 40001016043P4

TERMO DE APROVAÇÃO

Os membros da Banca Examinadora designada pelo Colegiado do Programa de Pós-Graduação ENGENHARIA ELÉTRICA da Universidade Federal do Paraná foram convocados para realizar a arguição da tese de Doutorado de **CARLOS EDUARDO MAFFINI SANTOS** intitulada: **UM NOVO MÉTODO DE GERÊNCIA ATIVA DE FILAS PARA FLUXOS DASH AO VIVO**, sob orientação do Prof. Dr. CARLOS MARCELO PEDROSO, que após terem inquirido o aluno e realizada a avaliação do trabalho, são de parecer pela sua APROVAÇÃO no rito de defesa.

A outorga do título de doutor está sujeita à homologação pelo colegiado, ao atendimento de todas as indicações e correções solicitadas pela banca e ao pleno atendimento das demandas regimentais do Programa de Pós-Graduação.

CURITIBA, 07 de Março de 2022.

Assinatura Eletrônica
16/03/2022 21:51:58.0
CARLOS MARCELO PEDROSO
Presidente da Banca Examinadora

Assinatura Eletrônica
22/03/2022 14:02:30.0
EMILIO CARLOS GOMES WILLE
Avaliador Externo (UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ)

Assinatura Eletrônica
16/03/2022 18:09:58.0
EVELIO MARTÍN GARCÍA FERNÁNDEZ
Avaliador Interno (UNIVERSIDADE FEDERAL DO PARANÁ)

Assinatura Eletrônica
29/03/2022 17:12:38.0
CARLOS ALBERTO MALCHER BASTOS
Avaliador Externo (UNIVERSIDADE FEDERAL FLUMINENSE)

RESUMO

A transmissão de fluxos de vídeo tornou-se popular na Internet. Fluxos de vídeo de eventos ao vivo possuem exigências ainda mais restritas quando comparado a transmissão de vídeos sob demanda, sendo mais sensíveis ao atraso, *jitter* e descarte de pacotes. Minimizar o impacto desses parâmetros é fundamental para manter a qualidade dos vídeos em níveis satisfatórios. A transmissão de fluxos de vídeo de modo dinâmico e adaptativo (DASH, *Dynamic Adaptative Streaming over HTTP*) é uma das tecnologias mais populares para transmissão de vídeo ao vivo e também de vídeo sob demanda (VoD, *Video on Demand*), permitindo a transmissão de fluxos adaptativos de acordo com a largura de banda disponível. No DASH, o cliente é responsável por iniciar, gerenciar e manter a conexão com o servidor. A fim de manter a reprodução contínua do vídeo, as aplicações VoD comumente utilizam grandes *buffers* de recepção. Porém, em transmissões ao vivo, o uso grandes *buffers* não são permitidos devido ao aumento significativo do atraso. Assim, congestionamentos na rede tendem a degradar a qualidade do vídeo. Os gerenciadores ativos de fila (AQM, *Active Queue Management*) surgem como alternativa para controlar congestionamentos nas filas dos roteadores, fazendo com que as fontes de tráfego TCP reduzam suas taxas de transmissão. Como consequência, o cliente DASH tende a diminuir a qualidade do segmento do vídeo como forma de reagir ao congestionamento. Segmentos de menor qualidade possuem menor tamanho em bytes e, consequentemente necessitam de menos tempo para serem transmitidos na fila, diminuindo o congestionamento. Nesta tese, dois novos algoritmos de AQMs são propostos. Ambos utilizam as redes neurais LSTM (*Long Short Term Memory*) para realizar a previsão do atraso na fila, antecipando a ocorrência de futuros congestionamentos. Isto faz com que o cliente DASH diminua a qualidade do segmento, evitando congelamentos da imagem e pausas na reprodução. Considerando o cenário de trabalho e estudo remoto imposto pela COVID-19, os algoritmos propostos destacam-se por melhorar a qualidade de vídeo-conferências ao vivo. A avaliação de desempenho dos recentes algoritmos AQMs, para transmissão DASH de vídeo ao vivo, é realizada. Os resultados mostram que os métodos propostos superam os demais AQMs testados, principalmente quando o enlace da rede encontra-se muito congestionado. Os AQMs propostos melhoram a qualidade do vídeo em termos da relação sinal-ruído de pico, da similaridade estrutural e da quantidade e duração de interrupções na imagem.

Palavras-chave: Long Short Term Memory; Redes Neurais Artificiais; DASH; Active Queue Management; Fluxo de vídeo ao vivo; Atraso na fila.

ABSTRACT

Video streaming currently dominates global Internet traffic. Live video streaming imposes even more strict requirements than video-on-demand (VoD), being more sensitive to delay, jitter, and packet loss. Dynamic Adaptive Streaming over HTTP (DASH) is the most popular technology for live streaming and VoD, and has been massively deployed on the Internet. In DASH, the client probes the network path and adapts the video quality according to instantaneous bandwidth fluctuations. Therefore, DASH is an over-the-top application using unmanaged networks to distribute content with the best possible quality. In order to maintain a seamless playback, VoD applications commonly use large reception buffers. However, the use of large buffers in live streaming services is not allowed because of the induced delay. Hence, network congestion could decrease the user-perceived video quality. Active Queue Management (AQM) arises as an alternative to control the congestion in router's queue, pressing the TCP traffic sources to reduce their transmission rate in case of incipient congestion. As a consequence, DASH client decreases the quality of the streamed video segment. In this thesis, we evaluate the performance of recent AQM strategies for real-time adaptive video streaming and propose two new AQM algorithms using Long Short Term Memory (LSTM) neural networks to improve the user-perceived video quality. The LSTM forecasts the trend of queue delay to allow earlier packet discard in order to avoid the network congestion. This, in turn, presses the DASH clients to decrease the video quality to avoid freezing in live streaming. Considering the remote work and study scenarios imposed by COVID-19, the proposed methods improve the video quality in live video conferences. The results show that the proposed methods outperform the competing AQM algorithms, mainly in scenarios of congested networks. The proposed AQMs improve the video quality in terms of average peak signal-to-noise ratio, structural similarity and the duration of video freezing.

Keywords: Long Short Term Memory; Artificial Neural Networks; Live Streaming; Active Queue Management; Video Streaming; DASH; Queue Delay

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS

ACF	*Autocorrelation Function*
API	*Application Programming Interface*
AQM	*Active Queue Management*
ARED	*Adaptive Random Early Detection*
ARIMA	*Auto-Regressive Integrated Moving Average*
ARMA	*Auto-Regressive Moving Average*
AVC	*Advanced Video Coding*
BBB	*Big Buck Bunny*
B-Frame	*Bidirectionally-Coded Frame*
CBR	*Constant Bit rate*
CoDel	*Controlled Queue Delay*
DASH	*Dynamic Adaptive Streaming over HTTP*
DRAPA	Descarte Randômico para Vídeo Adaptativo Baseado na Predição do Atraso
DRAPTA	Descarte Randômico para Vídeo Adaptativo Baseado na Predição da Tendência do Atraso
DRR	*Deficit Round Robin*
DSCP	*Differentiated Services Code Point*
DSDC	*Dynamic Segment Duration Control*
DSL	*Digital Subscriber Line*
ECN	*Explicit Congestion Notification*
EWMA	*Exponential Weighted Moving Average*
FARIMA	*Fractionally Auto-Regressive Integrated Moving Average*
FFNN	*Feed Forward Neural Network*
FQ-Codel	*Flow Queue Controlled Queue Delay*
FR	*Full Reference*

GOP	*Group of Pictures*
HD	*High Definition*
HVS	*Human Visual System*
HTTP	*HyperText Transfer Protocol*
IETF	*Internet Engineering Task Force*
I-Frame	*Intra-Coded Frame*
IPTV	*Television over IP*
IP	*Internet Protocol*
ISP	*Internet Service Provider*
ITU	*International Telecommunication Union*
LRD	*Long Range Dependence*
LSTM	*Long Short Term Memory*
MSE	*Mean Square Error*
MOS	*Mean Opnion Score*
MPEG-4	*Moving Picture Expert Group 4*
MPD	*Media Presentation Description*
MTU	*Maximum Transfer Unit*
NS-2	*Network Simulator version 2*
NS-3	*Network Simulator version 3*
PA	*Pedestrian Area*
PACF	*Partial Autocorrelation Function*
P-Frame	*Predictive-Coded Frame*
PIE	*Proportional Integral Controlled Enhanced*
PSNR	*Peak Signal Noise Ratio*
QoE	*Quality of Experience*
QoS	*Quality of Service*
RB	*Riverbed*
RED	*Random Early Detection*
RFC	*Request for Comments*

RH	*Rush Hour*
RNA	*Redes Neurais Artificiais*
RNN	*Recurrent Neural Network*
RTT	*Round Trip Time*
SF	*Sunflower*
SI	*Spatial Information*
SLA	*Service Level Agreement*
SSIM	*Structural Similarity*
SVC	*Scalable Video Coding*
TCP	*Transmission Control Protocol*
TI	*Temporal Information*
TP	*Touchdown Pass*
TR	Tempo de Reação
VBR	*Variable Bit Rate*
VCEG	*Video Coding Experts Group*
VoD	*Video on Demand*

LISTA DE SÍMBOLOS

n	Número de pesos da rede neural
γ	Intervalo de geração dos quadros de vídeo
ρ	Utilização do canal
μ	Taxa de atendimento do servidor
δ	Tempo de amostragem do atraso da fila
σ	Função sigmoide de ativação do neurônio artificial
y_t	Valor de saída da rede LSTM
x_t	Vetor de entrada da rede LSTM
C_t	Célula de memória da rede LSTM
f_t	Porta de esquecimento da rede LSTM
i_t	Porta de entrada da rede LSTM
s_t	Valores candidatos da rede LSTM
o_t	Porta de saída da rede LSTM
X	Série temporal de dados
H	Parâmetro de *Husrt*
ρ_k	Probabilidade de descarte linear do algoritmo DRAPA
ρ_z	Probabilidade de descarte linear do algoritmo DRAPA
ρ_d	Probabilidade de descarte linear do algoritmo DRAPTA
Ω	Intersecção entre as retas de probabilidades do algoritmo DRAPA
Ω_d	Peso da probabilidade de descarte do algoritmo DRAPTA

SUMÁRIO

CAPÍTULO 1

INTRODUÇÃO

Os serviços de entrega de vídeo digital em alta qualidade (HD, *High Definition*) tornaram-se populares na Internet. Isto foi impulsionado por aplicações tais como serviços de multimídia, teleconferências, sistemas de vigilância, IPTV (*Internet Protocol Television*) e mais recentemente, pelas plataformas de *streaming* de vídeo, todos possíveis devido ao contínuo aumento da largura de banda ofertada pelas operadoras (ROBLES; SILLER; WOODS, 2007). O tráfego de vídeo é responsável por uma grande porção dos dados transmitidos nas redes, quando comparado com os outros tipos de mídia (SEELING; REISSLEIN, 2014). De acordo com o relatório da Sandvine (SANDVINE, 2021), empresas como Netflix e Youtube responderam por mais de 26% do tráfego global em 2020, durante os primeiros meses de *lockdown* da pandemia de COVID-19. Devido ao isolamento social imposto pela pandemia, o tráfego migrou de redes empresariais, públicas e universidades, pressionando as redes de acesso, onde as taxas de transmissão normalmente são menores (SANDVINE, 2021). De acordo com o relatório anual da Cisco (CISCO, 2020), o tráfego gerado por vídeo sobre IP corresponderá a 82% de todo o tráfego consumido da Internet em 2022.

A crescente demanda por serviços de entrega de vídeo em alta qualidade passa pelo desenvolvimento de algoritmos de codificação cada vez mais eficientes. Eles possibilitam a transmissão com melhor qualidade, menor largura de banda e consequentemente menor necessidade de espaço para armazenamento. O H.264/MPEG-4 part 10 é o algoritmo mais utilizado atualmente. Quando comparado aos algoritmos antecessores, o H.264 é capaz de prover imagens com melhor qualidade, reduzindo significativamente a quantidade de bits usados para representá-las (TANWIR; PERROS, 2013; KWON; TAMHANKAR; RAO, 2006).

O fluxo de vídeo codificado é caracterizado por possuir taxa variável (VBR, *Variable Bit Rate*), comumente exibindo um comportamento auto-similar e uma dependência de longa duração, mesmo quando agregado em vários níveis. Dependendo do nível de utilização das filas dos roteadores, esse comportamento pode ocasionar congestionamentos, levando ao aumento do atraso e possíveis perdas de pacotes. Congestionamentos ocorrem quando a taxa de chegada de pacotes em um enlace do roteador excede a respectiva taxa de saída (ADAMS, 2013). Alta latência e perda de pacotes são os principais resultados do congestionamento (ARPACI; COPELAND, 2000).

Atualmente, a entrega de fluxos de vídeo de modo dinâmico e adaptativo sobre o protocolo HTTP (DASH, *Dynamic Adaptive Streaming over Hypertext Transfer Protocol*) tornou-se o padrão para a transmissão de vídeo sob demanda (VoD, *Video on Demand*) e ao vivo (BOUTEN et al., 2014; KUA; ARMITAGE; BRANCH, 2017). Além disso, o DASH não exige a criação de regras adicionais em *firewalls*, muito menos uma infraestrutura de rede dedicada, tornando o seu uso vantajoso para implementação prática nos sistemas web existentes (LOHMAR et al., 2011). No DASH, os vídeos são divididos temporalmente em segmentos e codificados com diversas resoluções, cada uma exigindo diferentes taxas de transmissão. Fica sobre responsabilidade do dispositivo cliente iniciar a sessão com o servidor e requisitar os segmentos, retirando deste último, a sobrecarga do gerenciamento de cada conexão. A cada segmento recebido, o cliente realiza a medição da largura de banda disponível no enlace, através de parâmetros como a taxa de transferência e latência (LEDERER; MULLER; TIMMERER, 2012). Isso possibilita que o DASH se adapte às flutuações que ocorrem na largura de banda, permitindo que o cliente busque segmentos que melhor se adaptem à taxa de transmissão do canal, na tentativa de manter a exibição do vídeo constante com a melhor qualidade possível.

O aumento no consumo de conteúdos de vídeo criou problemas de congestionamento nas redes, principalmente nas redes de acesso, onde a capacidade é limitada

quando comparada com o núcleo da Internet (FIGUEROA; FAVALLI, 2016; ADAMS, 2013; KUA; ARMITAGE; BRANCH, 2016). Apesar do aumento nas velocidades de conexões dos usuários, a média global é de apenas alguns Mbps (FIGUEROA; FAVALLI, 2016). Em certos casos, as redes IP não são capazes de oferecer níveis de qualidade de serviço (QoS, *Quality of Service*) adequados aos requerimentos necessários para fluxos de vídeo em tempo real (LI et al., 2010). Como alternativa para reduzir o descarte de pacotes, as operadoras aumentaram significativamente o tamanho dos *buffers* dos roteadores, na tentativa de melhor acomodar o tráfego. Esse fenômeno ficou conhecido como *bufferbloat* (GETTYS; NICHOLS, 2011) e foi impulsionado pela queda nos preços das memórias (GRIGORESCU; KULATUNGA; FAIRHURST, 2013). Porém, grandes *buffers* tendem a aumentar o atraso na fila. Em especial, esse efeito é amplificado na presença de fluxos VBR, ocasionando congestionamentos e imperfeições na qualidade do fluxo de vídeo, principalmente em transmissões ao vivo. Por outro lado, roteadores com pouco espaço em *buffer* reduzem o atraso da fila ao custo do aumento do descarte de pacotes, o que implica na diminuição da utilização do enlace (ADAMS, 2013). Segundo S. Floyd (FLOYD; GUMMADI; SHENKER, 2001), não pode-se ter simultaneamente alta utilização do enlace e baixos níveis de atraso na fila. Para transmissões de eventos ao vivo, o atraso, o *jitter* e o descarte de pacotes são aspectos fundamentais para manter a qualidade do tráfego de vídeo em níveis elevados. Ainda, os segmentos possuem um limite de tempo para serem reproduzidos no cliente, bem como um tempo de permanência no servidor. Caso a rede enfrente um cenário de congestionamento com um aumento considerável do atraso, os segmentos podem tornar-se inválidos para reprodução ou expirarem no servidor. De maneira a atenuar esses efeitos, novos algoritmos de gerenciamento ativo de filas (AQM, *Active Queue Management*) surgem como importantes reguladores de congestionamentos e mecanismos para mitigar a queda na qualidade percebida do vídeo. Mecanismos de controle de congestionamentos são cruciais para aplicações de vídeo (YANG; LIU; LI, 2007).

Implementado nos roteadores, o AQM é definido como um algoritmo proativo de

descarte/marcação de pacotes, projetado para explorar a capacidade do controle de congestionamento dos protocolos da camada de transporte, fornecendo justiça entre os fluxos e evitando o colapso da rede (ADAMS, 2013). O RED (*Random Early Detection*) foi a primeira proposta formal de AQM (FLOYD; JACOBSON, 1993; ADAMS, 2013), originalmente desenhado para explorar a capacidade de adaptação do protocolo TCP (*Transmission Control Protocol*). O RED detecta congestionamentos pelo monitoramento do tamanho médio da fila, notificando as fontes geradoras de tráfego, seja pela marcação de pacotes, em conjunto com a extensão ECN (*Explicit Congestion Notification*) do protocolo IP, ou pelo descartes de pacotes (ADAMS, 2013). O objetivo é fazer com que as fontes reajam ao congestionamento, diminuindo suas taxas de transmissão e evitando o colapso da rede.

Vários métodos AQM são relatados pela literatura, porém, nenhum foi projetado especificamente para explorar a capacidade do mecanismo de adaptação do tráfego DASH (KUA; ARMITAGE; BRANCH, 2016). A maioria dos algoritmos AQM realiza descartes randômicos durante períodos de congestionamento, sem levar em consideração as características do tráfego de aplicação. Este comportamento, aliado ao fenômeno do *bufferbloat*, tende a aumentar o tempo médio que os pacotes permanecem nas filas dos roteadores. Em transmissões de eventos ao vivo, o aumento do atraso acaba prejudicando a qualidade do vídeo. Considerando que o tráfego DASH exibe características previsíveis, uma nova classe de AQM pode ser implementada para esses fluxos. Nela, descartes antecipados de pacotes podem ser feitos como prevenção a futuros congestionamentos, suavizando as transições entre a qualidade dos segmentos, prevenindo que eles expirem no servidor. Isso traria benefícios para a qualidade do vídeo.

1.1 Objetivo Geral

O objetivo geral deste trabalho é melhorar a qualidade de fluxos de vídeo DASH ao vivo. Para tal, um novo método AQM foi desenvolvido. A proposta baseia-se na implementação do método em roteadores das redes de acesso, onde concentram-se os enlaces com gargalos. O método proposto deve explorar o comportamento auto-similar do tráfego de vídeo e a capacidade de adaptação do algoritmo DASH, realizando a predição do atraso na fila do roteador através da aplicação de redes neurais LSTM (*Long Short Term Memory*). O uso das redes LSTM deve se a sua habilidade em modelar séries temporais com características de longa duração (LRD, *Long Range Dependence*) com maior exatidão do que as redes neurais artificiais (ANN, *Artificial Neural Network*) e modelos matemáticos de previsão tradicionais (AZZOUNI; PUJOLLE, 2017). Além disso, elas são computacionalmente eficientes, com complexidade no pior cenário dada por de $O(n)$, onde n é o número de pesos da redes (HOCHREITER; SCHMIDHUBER, 1997), o que permite sua implementação em roteadores.

1.2 Objetivos Específicos

Os objetivos específicos compreendem:

- Apresentar os principais algoritmos de gerência ativa de filas disponíveis para transmissão de vídeo sobre o sistema DASH.
- Desenvolver um novo método AQM tendo em vista o comportamento de transmissão DASH ao vivo.
- Implementar o método proposto no NS3, incluindo a integração com a LSTM.
- Projetar o cenário de simulação através do software NS3 que permita a transmissão de fluxos de vídeo ao vivo em conjunto com fontes de tráfego simuladas.

- Avaliar o desempenho dos métodos AQMs atuais na transmissão de fluxos de vídeo adaptativo ao vivo em enlaces com diferentes níveis de congestionamento. Atualmente não existem trabalhos disponíveis na literatura fazendo avaliação semelhante.
- Avaliar o desempenho do método proposto para transmissão de fluxos de vídeo adaptativo ao vivo e comparar os resultados com os AQMs atuais.

Todos os objetivos foram atingidos e seus resultados serão apresentados e descritos ao longo deste trabalho.

1.3 Trabalhos Publicados

Durante o desenvolvimento deste trabalho foram publicados dois artigos em revistas de pesquisa: o primeiro na revista *Entropy* (SANTOS; SILVA; PEDROSO, 2021), e o segundo na revista *Telecommunication Systems* (MORAIS et al., 2021). Outros dois artigos foram publicados no Simpósio Brasileiro de Telecomunicações e Processamento de Sinais (SBrT), sendo um deles no ano de 2019 (MORAIS; SANTOS; PEDROSO, 2019) e outro em 2021 (SANTOS; PEDROSO, 2021).

Além deste Capítulo introdutório, o restante deste trabalho está estruturado da seguinte forma: O Capítulo 2 descreve os principais conceitos utilizados na pesquisa e desenvolvimento do método proposto, como os métodos de codificação de vídeo, o padrão DASH, fundamentos de AQM e redes neurais LSTM. O Capítulo 3 apresenta a revisão bibliográfica dos principais algoritmos de gerência ativa de filas. No Capítulo 4, o método AQM proposto é apresentado. O Capítulo 5 descreve a avaliação de desempenho, bem como os resultados simulados. Por fim, o Capítulo 6 apresenta as conclusões.

CAPÍTULO 2

CONCEITOS FUNDAMENTAIS

Este capítulo apresenta os principais conceitos envolvidos na pesquisa deste trabalho, como os algoritmos de codificação de vídeo, a descrição da tecnologia DASH, o princípio de funcionamento dos AQMs e as premissas que envolvem as redes neurais e redes LSTM.

2.1 Codificação MPEG-4/AVC

Os algoritmos de codificação são utilizados para compressão de vídeos antes que eles sejam transmitidos na rede, de forma a diminuir a quantidade de largura de banda necessária, quando comparado com o formato original. Recentemente, esses codificadores tiveram significantes melhorias, emplacando em importantes vantagens para o transporte através das redes (SEELING; REISSLEIN, 2014).

O MPEG é uma família de padrões internacionais que fornecem ferramentas para o uso em aplicações multimídia (AUWERA; DAVID; REISSLEIN, 2008). Os algoritmos de codificação são utilizados na compressão dos vídeos antes que eles sejam transmitidos na rede, diminuindo a largura de banda necessária quando comparado com o formato original. O padrão MPEG-4/AVC (*Advanced Video Coding*) foi desenvolvido através de uma equipe formada por especialistas de grupos de pesquisa distintos, como o de codificação em vídeo (VCEG, *Video Coding Experts Group*) e de imagem em movimento (MPEG, *Moving Pictures Experts Group*). Este padrão é amplamente utilizado na codificação de vídeo e continuamente atualizado a medida que novas tecnologias surgem (PURI; CHEN; LUTHRA, 2004). O MPEG-4 teve significantes melhorias, com importantes vantagens para o transporte de vídeo através das redes de

computadores (SEELING; REISSLEIN, 2014). Além disso, o MPEG-4 fornece menores taxas de transmissão quando comparado com os seus antecessores.

Os algoritmos de codificação introduzem a notação de quadro (*frame*). Um quadro pode ser interpretado como uma foto de uma cena em uma sequência de vídeo. Cada quadro é divido em pequenos blocos de $M \times N$ pixels, dependendo do algoritmo utilizado. Com o avanço dos padrões de codificação de vídeo, foi possível tornar o particionamento de um quadro em blocos de tamanhos diferentes. Enquanto o MPEG-2 era limitado a blocos fixos de 16×16 pixels, o MPEG-4/AVC possibilita que blocos de 16×16 possam ser sub particionados em tamanhos menores de até 4×4. Blocos maiores funcionam melhor em regiões suavizadas da imagem, enquanto que blocos menores beneficiam curvas e regiões de texturas (POURAZAD et al., 2012).

Assim como no MPEG-1 e no MPEG-2, o MPEG-4 codifica os vídeos utilizando informações redundantes da sequência de quadros. Portanto, os blocos de um quadro podem ser intra-codificados, isto é, codificados levando em consideração apenas blocos pertencentes ao mesmo quadro, ou inter-codificados, sendo codificados com referencias à quadros vizinhos, sucessores ou antecessores do quadro atual, considerando a sequência temporal de amostragem. Neste último, blocos do quadro a ser codificado e que possuem semelhantes nos quadros vizinhos, são representados por um vetor de movimento apontado para o bloco de referência, ao invés de serem codificados por completo. Avanços na inter codificação, diminuíram significativamente a quantidade de informação a ser transmitida (SEELING; REISSLEIN, 2014).

O padrão ao MPEG-4 utiliza três tipos de quadros:

- Quadro I (*Intra-Coded Frame*): A imagem codificada contém toda a informação necessária para ser reconstruída no decodificador.
- Quadro P (*Predictive-Coded Frame*): A imagem é codificada utilizando informações do quadro I ou P anteriores.
- Quadro B (*Bidirectionally-Coded Frame*): A imagem é codificada utilizando infor-

mações tanto dos quadros I quanto P anterior e posterior.

A sequência de quadros sempre é iniciada por um quadro I, seguido por quadros B ou P. Essa estrutura é comumente referida como grupo de figuras (GOP, *Group of Picture*), como apresentado na Figura 2.1. O tamanho do GOP varia de acordo com a aplicação. A notação mais comum, utiliza o par (a, b), onde a indica o número total de quadros e b representa o número de quadros B entre os P e I no GOP.

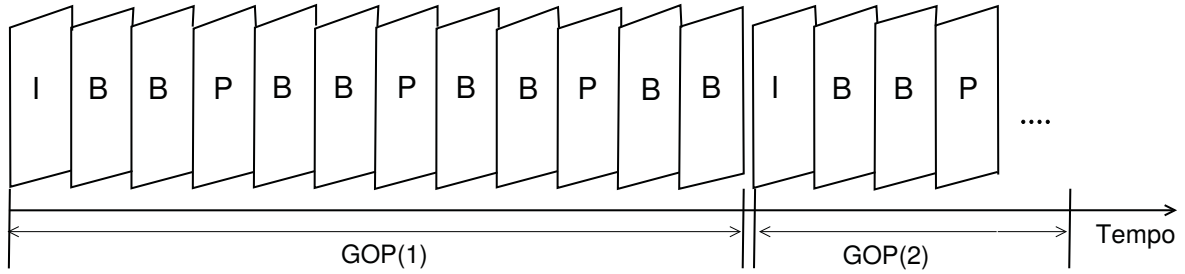

Figura 2.1: Sequência de quadros I, P e B, após serem codificados com MPEG-4.

Devido aos quadros I conterem toda a informação necessária para que a imagem possa ser reconstruída no decodificador, sem precisar das informações de outros quadros, normalmente eles possuem maior tamanho que os demais, como ilustra a Figura 2.2.

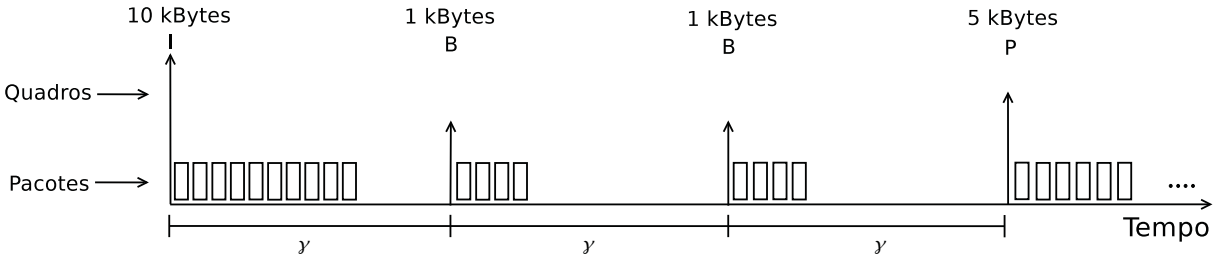

Figura 2.2: Codificação em pacotes para transportar os quadros do vídeo gerado, evidenciando o comportamento em rajada do tráfego.

Os quadros são gerados em intervalos fixos de tempo (γ) e encapsulados em pacotes. Como o tamanho da MTU (*Maximum Transfer Unit*) é normalmente menor que o tamanho do quadro, antes da transmissão pela rede, os dados são divididos e enviados em diversos pacotes. No caso dos quadros I, como eles são maiores que os quadros P e B, a quantidade de pacotes para transportá-los também é maior.

Na Figura 2.2 é possível perceber que a codificação é responsável pelo comportamento em rajada do tráfego de vídeo, uma vez que o tamanho do quadro é variável e dependente da informação. O tráfego de vídeo, comumente exibe características de dependência temporal de longa duração, com decaimento lento da sua função de autocorrelação (GARRETT; WILLINGER, 1994). Quando vários fluxos de vídeos são multiplexados, o tráfego resultante agregado também apresenta dependência temporal de longa duração (BIERNACKI, 2017).

Avanços na inter codificação, ou seja, codificação de quadros utilizando vetores de movimento, melhoraram significativamente nos últimos anos (SEELING; REISSLEIN, 2014). Novos algoritmos como a versão SVC (*Scalable Video Coding*) do H.264 e o H.265/HEVC (*High Efficiency Video Coding*), utilizam o conceito de inter codificação com hierarquia diádica (SEELING; REISSLEIN, 2014). Neste tipo de codificação, os quadros B podem ser codificados com origem em outros quadros B, passados ou futuros, de acordo com a camada a que pertencem na estrutura de codificação. Os quadros I e P formam a camada base da estrutura e os quadros B as demais camadas superiores. A quantidade de camadas que a estrutura possui depende da quantidade de quadros B entre sucessivos quadros I (ζ). Portanto, sendo τ o número de quadros B entre os quadros I, o número de camadas será dado por $\tau = log_2(\zeta + 1)$. Por exemplo, em um GOP com 15 quadros B entre os I tem-se uma estrutura de quatro camadas hierárquicas ($\tau = 4$). Neste tipo de codificação, um quadro B é codificado com referência aos quadros anteriores e sucessores, pertencentes às camadas abaixo. Apesar do H.264/SVC e H.265/HEVC serem mais novos que o H.264/AVC, eles não foram utilizados nesta tese. Como a transmissão de vídeos atualmente ocorre através da tecnologia DASH, espera-se um comportamento semelhante em termos de melhoria da qualidade pois o método proposto está baseado na capacidade de adaptação do DASH.

2.2 *Dynamic Adaptive Streaming over HTTP*

As plataformas modernas de distribuição de vídeo pela Internet adotaram o DASH como a principal técnica de transmissão (KUA; ARMITAGE; BRANCH, 2016). A fim de propor um padrão para a entrega de fluxos de vídeo sobre o HTTP, o MPEG desenvolveu uma solução chamada MPEG-DASH (BOUTEN et al., 2014).

No DASH, o mesmo vídeo é codificado em múltiplas versões e com diferentes taxas de qualidade, sendo disponibilizadas no servidor. Cada representação é então fragmentada em pequenos segmentos de alguns segundos, respeitando a sequência temporal do vídeo. Isso possibilita que o cliente requisite segmentos de diferentes qualidades ao longo da transmissão, de acordo com o nível de congestionamento da rede. Ao codificar e gerar os segmentos de vídeo, o servidor cria um arquivo de descrição, chamado MPD (*Media Presentation Description*). Nele, uma série de itens são descritos como o tipo da transmissão (ao vivo ou sob demanda), a duração total do vídeo e de um segmento, algoritmo de codificação utilizado, largura de banda necessária para transmitir cada segmento de acordo com a sua qualidade, entre outros.

Ao contrário das estratégias tradicionais de envios de fluxos de vídeo, o servidor DASH não controla a transmissão diretamente. O dispositivo cliente é responsável por iniciar e gerenciar a conexão com o servidor. Ao iniciar uma transmissão, o cliente DASH envia uma requisição HTTP ao servidor, solicitando o arquivo MPD. Assim que o arquivo é recebido e analisado, o cliente requisita o segmento de inicialização do vídeo (na Figura 2.3, o segmento identificado como init) e na sequência os segmentos. A qualidade do primeiro segmento é configurável no cliente, podendo ser o de maior ou menor qualidade. Como ilustrado na Figura 2.3 a cada segmento recebido, o cliente estima a largura de banda do enlace através do cálculo do tamanho do segmento pelo tempo que ele levou para ser entregue. A próxima requisição leva em consideração esse cálculo. A medida que a transmissão avança, o cliente DASH requisita pelos segmentos que melhor se adaptam às condições atuais da rede.

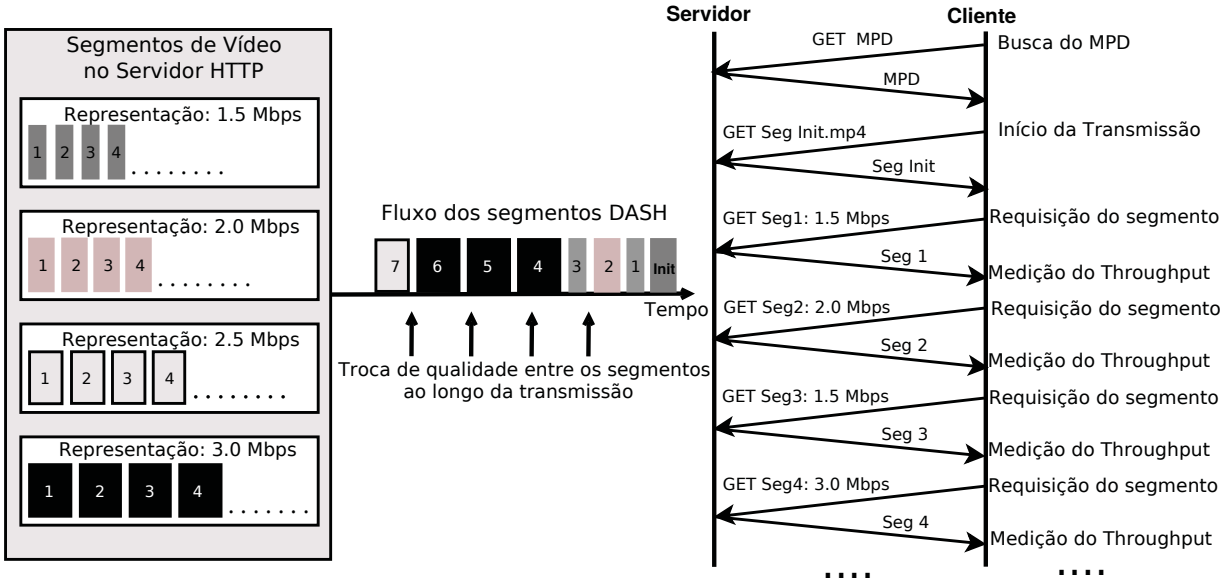

Figura 2.3: Comportamento adaptativo do DASH ao longo do tempo (KUA; ARMITAGE; BRANCH, 2017).

Na transmissão VoD, o servidor codifica e armazena todos os segmentos, deixando-os disponíveis para *download*. O cliente requisita os segmentos o mais rápido possível, a fim de preencher o seu *playout buffer*. Para tal, clientes DASH em transmissões VoD possuem de 20 a 30 segundos de capacidade no *playout buffer*. A reprodução do vídeo é iniciada após este preenchimento, a fim se assegurar uma reprodução contínua do vídeo, de modo a minimizar os efeitos do atraso e descarte de pacotes inerentes à rede.

Porém, em transmissões DASH de eventos ao vivo, armazenar essa quantidade de segmentos não é possível devido ao atraso induzido em função do preenchimento do *playout buffer*. Em transmissões ao vivo, o cliente DASH armazena de 2 a 3 segundos em seu *playout buffer*. Isso implica que o seu algoritmo de adaptação tenha que reagir de forma rápida e robusta às mudanças da rede (BOUTEN et al., 2014). Diferente das transmissões VoD, nos eventos ao vivo a geração dos segmentos de vídeo ocorre em etapas. O servidor gera alguns segmentos de vídeo, que permanecem ativos para *download* por alguns segundos. Passado esse tempo, os segmentos são apagados dando lugar à novos, seguindo a sequência temporal do vídeo. Isso garante que os clientes estejam reproduzindo o evento ao vivo com uma pequena diferença temporal

entre eles. Devido a congestionamentos na rede, atrasos demasiados na entrega do segmento atual impactam na perda da sequência do vídeo, fazendo com que o cliente sofra congelamentos na imagem. Os algoritmos AQM podem ser utilizados como importantes reguladores de congestionamentos, de forma a cooperar com o algoritmo de adaptação do DASH, mitigando os efeitos do atraso em eventos de transmissão ao vivo.

2.3 Gerenciamento Ativo de Filas

Algoritmos AQM são controladores proativos de congestionamentos que atuam nas filas dos roteadores (ADAMS, 2013). Eles são responsáveis por detectar de forma antecipada tais eventos e notificar as fontes geradoras de tráfego, através de uma marcação ECN ou de um descarte de pacote (ADAMS, 2013). Para tal, eles utilizam informações como o tamanho da fila, a taxa de chegada dos pacotes e os eventos de fila cheia e vazia. Em resposta, as fontes TCP tendem a diminuir suas taxas de transmissão como forma de prevenir que a fila atinja sua capacidade máxima (ZHU et al., 2002). A medida que o congestionamento aumenta, o AQM tende a intensificar o descarte/marcação dos pacotes (ADAMS, 2013).

Conforme ilustrado pela Figura 2.4, um AQM é composto por três componentes principais: (I) o indicador de congestionamento, (II) a função de controle de congestionamento e (III) o mecanismo de reação (*feedback*). O indicador de congestionamento (I) é encarregado de assinalar a existência de tal evento. Para esse fim, métricas como (a) o tamanho médio ou instantâneo da fila (*Queue-Based*) e/ou (b) a taxa de chegada dos pacotes (*Rate-Based*), são utilizadas. No caso da métrica (a), o propósito é manter o tamanho da fila (médio ou instantâneo) dentro de um certo limiar. Já para (b), o objetivo é manter a taxa de chegada dos pacotes dentro de um limite, o qual é determinado em função da capacidade do enlace de saída. A função de controle (II), calcula a probabilidade em que os pacotes serão marcados ou descartados, caso o

indicador de congestionamento notifique-a da existência de um evento. Por último, o mecanismo de reação (III) é responsável por descartar pacotes ou enviar notificações para as fontes geradoras de tráfego, sendo diretamente associado com a função de controle.

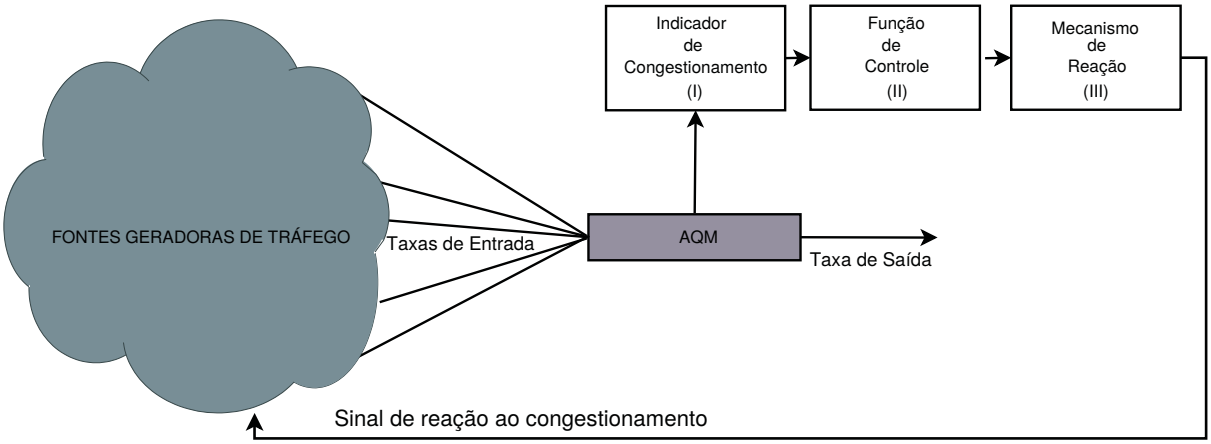

Figura 2.4: Estrutura de um gerenciador ativo de filas.

O RED, que será descrito mais adiante, foi o primeiro AQM proposto, com o objetivo de fazer com que as fontes de tráfego TCP diminuam suas taxas de transmissão a fim de mitigar o impacto do congestionamento. Apresentado em 1993 por Sally Floyd e Van Jacobson (FLOYD; JACOBSON, 1993), o algoritmo realiza descartes antecipados e randômicos de pacotes. Os AQM tornaram-se importantes reguladores de congestionamentos e melhoram a justiça no compartilhamento da largura de banda entre as aplicações.

Apesar da quantidade de trabalhos publicados sobre o tema, poucos exploram como o tráfego DASH interage com os AQM em transmissões ao vivo (KUA; ARMITAGE; BRANCH, 2016). O comportamento auto-similar do tráfego de vídeo e o problema do *bufferbloat* pode resultar no enfileiramento de pacotes por longos períodos, reduzindo a qualidade do vídeo em tempo real, uma vez que os segmentos expiram no servidor. Assim, realizar a predição do comportamento do tráfego na fila do roteador, de forma a realizar descartes antecipando congestionamentos e forçando os clientes DASH a reduzirem a qualidade dos segmentos, pode a melhorar a qualidade do vídeo em transmissões ao vivo.

2.4 Redes Neurais Artificiais

Referenciado como um computador complexo, não linear e paralelo, o cérebro humano possui a capacidade de realizar certos processamentos, como o reconhecimento de padrões (HAYKIN, 2001). Contando com uma grande estrutura de neurônios, o cérebro consegue desenvolver suas próprias regras através de experiências que são adquiridas ao longo do tempo. Cada neurônio realiza a comunicação com seu vizinho através das sinapses, as quais permitem transmitir impulsos elétricos entre neurônios vizinhos (HAYKIN, 2001). As ANNs foram projetadas de forma a imitar a funcionamento do cérebro humano.

Assim como no cérebro humano, a experiência também é essencial para a ANN. Nela, o conhecimento é adquirido através de um processo de aprendizagem e as forças das conexões entre os neurônios, conhecidas como pesos sinápticos, são utilizadas para armazenar o conhecimento adquirido.

A Figura 2.5 apresenta os elementos que compõe um neurônio artificial. Essencialmente, um conjunto de m entradas são aplicadas ao neurônio, de modo que cada entrada x_i $(0 < i \leq m)$ seja multiplicada pelo seu peso sináptico correspondente w_{ij} (peso da entrada x_i do neurônio j), sendo o resultado submetido a um somador. Di-

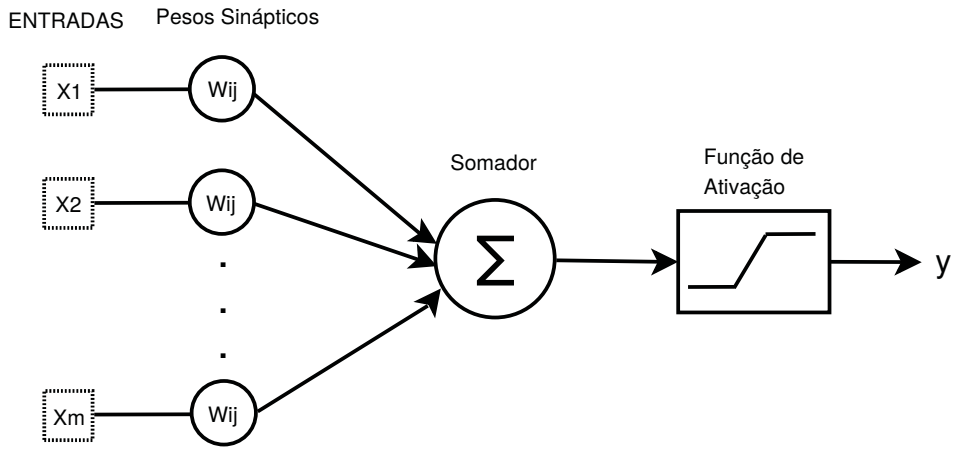

Figura 2.5: Estrutura de um neurônio artificial.

retamente conectado ao somador, a função de ativação (f) possui o papel de filtrar a amplitude do sinal a valores entre 0 e 1, ou, de -1 a 1. Portanto, a saída do neurônio é dada como: $y = f(\sum x_i.w_{ij})$, a qual será transmitida adiante, excitando ou inibindo o próximo neurônio.

Existe uma grande variedade de funções de ativação para a implementação das ANNs, sendo as mais comuns as funções: linear, degrau, tangente hiperbólica e sigmoide. A Figura 2.6 apresenta as curvas características de cada uma das funções citadas.

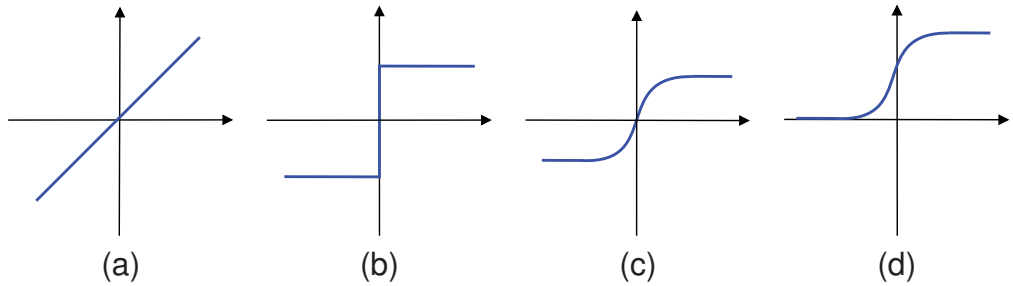

Figura 2.6: Funções de ativação (a) Linear; (b) Degrau; (c) Tangente Hiperbólica e (d) Sigmóide

A quantidade e a forma como os neurônios estão conectados definem a arquitetura da ANN. Basicamente elas podem ser classificadas em: (a) redes neurais alimentadas à frente (FFNN, *Feed-Forward Neural Network*) ou (b) redes neurais recorrentes (RNN, *Recurrent Neural Network*). Em ambos os casos, os neurônios estão organizados em camadas. As redes alimentadas à frente são caracterizadas pelo sinal de entrada ser propagado em um único sentido, da entrada para a saída, não existindo ligação entre os neurônios que estão na mesma camada. A Figura 2.7 (a) apresenta uma rede neural alimentada a frente, com uma camada de entrada, uma camada escondida e uma camada de saída. No caso das redes recorrentes, ocorre a existência de pelos menos um laço de realimentação, onde, a saída de um neurônio da camada escondida, serve de entrada para os neurônios pertencentes a camada de realimentação, como ilustra a Figura 2.7 (b). A saída dos neurônios da camada de realimentação serve de entrada para todos os neurônios da mesma camada escondida que iniciou este ciclo. A presença desse laço produz impacto profundo na capacidade de aprendi-

zado da rede, devido a existência de elementos de atrasos unitários (representado por $Z-1$ na figura). Pode-se considerar que uma rede recorrente possui memória e são muito utilizadas em resolver problemas de análises de séries temporais (MCCULLUM, 2020).

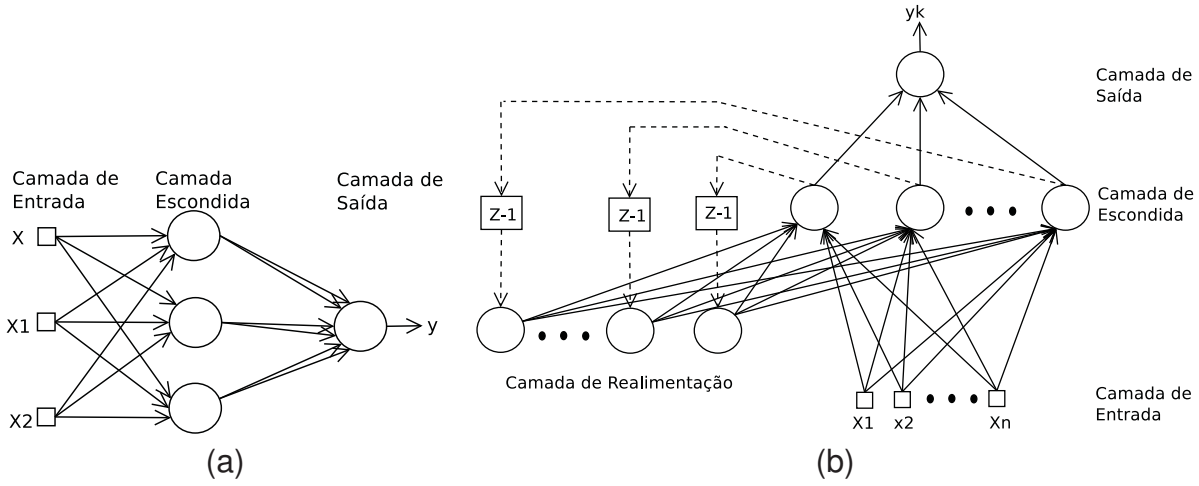

Figura 2.7: Topologia das redes neurais (a) Rede Alimentada a Frente e (b) Rede Recorrente.

Uma propriedade que é de importância primordial para uma rede neural é a sua habilidade de aprender e de melhorar o seu desempenho a partir de sua aprendizagem. O aprendizado de uma rede neural ocorre através de um processo de ajustes aplicados a seus pesos sinápticos, permitindo que a rede seja submetida à situações que não faziam parte do conjunto de treinamento original. Dentre os paradigmas de aprendizagem, destaca-se o de aprendizagem supervisionada, sendo o algoritmo de retro propagação do erro (*Backpropagation*) o mais comum. Nele, um conjunto de padrões são apresentados para a camada de entrada da rede, que irá servir de exemplo. A partir da entrada, as informações são processadas e fluem através da rede, camada por camada, até que a resposta seja obtida pela camada de saída. Esse procedimento é chamado de fase de propagação para frente (*forward propagation phase*). Em um segundo momento, a saída obtida pela rede é comparada com a saída fornecida pelos dados de entrada. Caso os valores das saídas não sejam iguais, o erro é calculado e propagado da camada de saída até a entrada, alterando os valores dos pesos sinápticos das conexões internas da rede. Esse procedimento é conhecido por

retro propagação do erro (*backward propagation*). O algoritmo de retro propagação é o mais famoso entre os algoritmos de aprendizado, podendo ser especialmente utilizado em casos de conjuntos amplos de treinamento com muitos exemplos similares. Os ajustes da rede são realizados passo a passo a cada nova iteração, de forma que a rede consiga abstrair as características do conjunto de exemplos. Após treinada, pode-se retirar o conjunto de exemplos e deixar a rede lidar com os demais dados inteiramente por si só.

Quando aplicado, o algoritmo de retro propagação ajusta os pesos da rede através do cálculo do gradiente do erro, buscando por mínimos locais ou globais (HAYKIN, 1998). Entretanto, se o sinal do erro propagado é muito pequeno, o algoritmo tende a realizar pequenos ajustes nos pesos da rede, interrompendo o aprendizado. Este evento é conhecido como desaparecimento ou explosão do gradiente (*vanishing or exploding gradient problem*) (HUA et al., 2019). Mesmo com o uso de algumas técnicas para tentar evitar que o gradiente fique próximo de zero, para sequências muito extensas, as RNN não conseguem reter a informação por muito tempo. A rede neural LSTM surge para resolver esse problema, conforme será detalhado mais adiante.

As ANNs são amplamente usadas na previsão de tráfego de vídeo devido sua habilidade em aprender padrões complexos e estimar funções lineares e não lineares (AZZOUNI; PUJOLLE, 2017), além da sua simplicidade, robustez, adaptabilidade e exatidão (FENG; SHU, 2005). A predição do tráfego de vídeo ao vivo usando RNAs supera os modelos matemáticos de previsão lineares como o ARIMA (*Auto-Regressive Integrated Moving Average*) e FARIMA (*Fractionally Auto-Regressive Integrated Moving Average*) (BARABAS et al., 2011).

2.4.1 Long Short Term Memory

Proposta por Hochreiter and Schmidhuber (HOCHREITER; SCHMIDHUBER, 1997), a LSTM é um tipo especial de rede neural recorrente. Sua arquitetura é composta

por unidades chamadas de blocos de memória, sendo mais complexo que o neurônio artificial de uma rede neural padrão. Além de resolver o problema de desaparecimento ou explosão do gradiente, as LSTMs são capazes de reter a informação por mais tempo e realizar a previsão de séries temporais de longa duração com maior precisão, quando comparada a uma RNN (HUA et al., 2019).

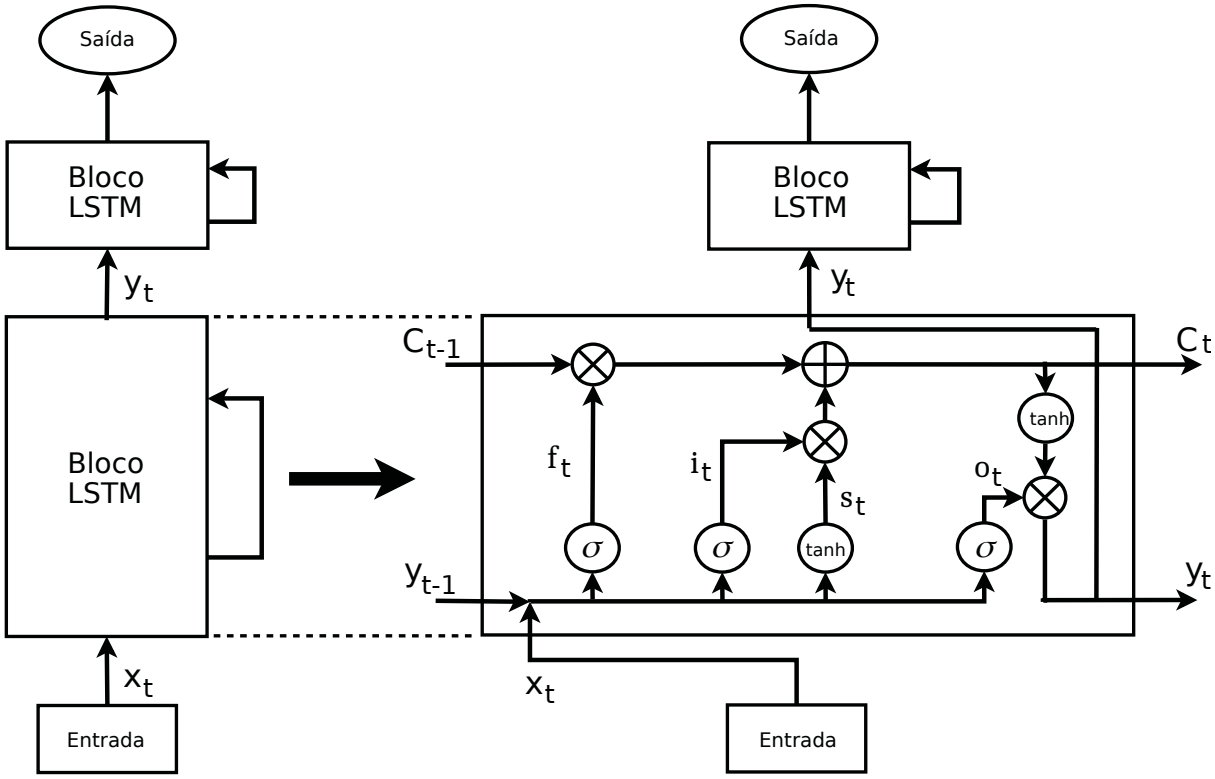

Figura 2.8: Bloco de memória da rede neural LSTM.

Como mostra a Figura 2.8, o bloco de memória da LSTM é composto por cinco unidade principais, que são:

- Célula de memória (C_t) - (*memory cell*).
- Porta de esquecimento (f_t) - (*forget gate*).
- Porta de entrada (i_t) - (*input gate*).
- Valores candidatos (s_t) - (*candidate values*).
- Porta de saída (o_t) - (*output gate*).

A célula de memória e a saída da rede (y_t) são realimentadas, ou seja, seus valores passados são utilizados como entrada para o mesmo bloco. Na Figura 2.8, C_{t-1} e y_{t-1} representam a realimentação. Além disso, C_t funciona como uma memória, sendo responsável por propagar as informações para a saída do bloco. A cada nova iteração, novos valores de entrada (x_t) e da saída anterior (y_{t-1}) são apresentados ao bloco de memória.

A porta de esquecimento (f_t) determina quais informações devem ser removidas ou lembradas em C_t. Para tal, ela é composta por uma função de ativação do tipo sigmoide (σ), onde os valores de saída serão entre 0 (totalmente esquecido) e 1 (totalmente lembrado), dado por:

$$f_t = \sigma(w_f[x_t, y_{t-1}]), \tag{2.1}$$

onde w_t é a matriz peso da porta de esquecimento. Como próximo passo, a LSTM determina quais informações devem ser adicionadas a C_t, através das operações de i_t e s_t. Em ambas as portas, os valores da entrada atual da rede e da saída anterior são utilizados, sendo as operações diferenciadas por suas respectivas funções de ativações: tangente hiperbólica para i_t e sigmoide para s_t. A porta i_t indicará quais valores serão atualizados em C_t, enquanto a porta s_t, cria um novo vetor com os valores candidatos a serem incluídos em C_t. Logo, i_t e s_t são definidas como:

$$i_t = \sigma(w_i[x_t, y_{t-1}]), \tag{2.2}$$

$$s_t = \tanh(w_s[x_t, y_{t-1}]), \tag{2.3}$$

sendo w_i e w_s as matrizes de pesos para i_t e s_t, respectivamente. Portanto, a cada nova iteração a célula de memória é atualizada:

$$C_t = (f_t \cdot C_{t-1}) + (i_t \cdot s_t), \tag{2.4}$$

Por fim, o valor de o_t é estimado, determinando quanta informação de C_t deve ser usada para computar a saída da rede (y_t). Para isto, os valores de x_t e y_{t-1} são

combinados com os valores obtidos em C_t. Assim, o_t e y_t são definidos como:

$$o_t = \sigma(w_o[x_t, y_{t-1}]), \tag{2.5}$$

$$y_t = \tanh(C_t) \cdot o_t, \tag{2.6}$$

sendo w_o a matriz de pesos da porta de saída. Devido a cooperação entre C_t e as suas portas, a LSTM é considerada uma ferramenta poderosa para realizar tarefas de previsões de séries temporais com dependência de longa duração. Em resumo, a célula de memória é responsável por lembrar dos estados temporais da rede neural e as portas, as quais são formadas por unidades de multiplicação, são responsáveis por controlar o padrão do fluxo da informação (HUA et al., 2019).

Este Capítulo tratou dos principais conceitos que serão utilizados ao decorrer deste trabalho. Dentre esses, os principais pontos que podem ser citados são (1) os algoritmos de codificação de vídeo, em particular, o MPEG-4 e a sua característica VBR; (2) a técnica de transmissão de vídeos ao vivo de modo adaptativo (DASH), a qual é amplamente utilizada pelas plataformas mais modernas de distribuição de vídeos; (3) algoritmos de gerência ativa de filas, realizando descartes de modo a evitar o problema do colapso da rede devido a superlotação de buffers e (4) o conceito das redes neurais artificiais LSTM, utilizadas neste trabalho como uma ferramenta de previsão do atraso, a fim de detectar cenários de congestionamentos na fila do roteador, cooperando com o algoritmo AQM na realização de descartes antecipados, de forma que os fluxos DASH em tempo real reduzam a qualidade do segmento, evitando o colapso da rede. O próximo Capítulo apresenta o estado da arte dos algoritmos de gerência ativa de filas.

CAPÍTULO 3

REVISÃO BIBLIOGRÁFICA

Este Capítulo trata da revisão bibliográfica dos métodos utilizados na melhoria da qualidade de vídeo para fluxos DASH. Fluxos de vídeo são codificados e transmitidos pela Internet, a qual não possui garantias de taxa de transmissão, atraso, *jitter* ou perda de pacotes. Caso a qualidade disponível na rede não seja suficiente para transmitir o conteúdo do vídeo, degradações na qualidade do vídeo podem ser percebidas pelos usuários. A fim de evitar essas consequências, sem ter que implementar mecanismos de qualidade de serviço complexos e custosos no núcleo da Internet, estão disponíveis diversas técnicas para melhorar a qualidade da transmissão de vídeos DASH (KUA; ARMITAGE; BRANCH, 2017).

3.1 Fatores que Influenciam a Qualidade DASH

De modo geral, a qualidade do vídeo DASH pode ser influenciada por fatores que impactam diretamente na percepção do usuário final, como por exemplo o tempo inicial de espera para o início da reprodução do vídeo, interrupções na imagem e as estratégias de adaptação do vídeo.

3.1.1 Atraso Inicial

Geralmente, o início da reprodução de um vídeo sob demanda é atrasado pelo tempo necessário para preencher o *playout buffer* do cliente (SEUFERT et al., 2015). Com isso evitam-se interrupções na reprodução, caso a largura de banda do enlace seja reduzida momentaneamente. Um menor tempo inicial de reprodução implica em um

menor *playout buffer*, o que pode causar um maior número de interrupções. Segundo alguns estudos, o tempo de atraso inicial é preferido por 90% dos usuários do que interrupções durante a reprodução do vídeo, sendo este último mais impactante na percepção de qualidade de vídeo para o ser humano (SEUFERT et al., 2015). Sendo assim, o atraso inicial na reprodução do vídeo possui menor impacto na qualidade percebida pelo usuário. Porém, em transmissões de vídeo ao vivo, preencher o *playout buffer* por mais de 3 segundos causa atrasos indesejados, fazendo com que o vídeo fique defasado em relação ao ponto ao vivo.

3.1.2 Interrupções

Interrupções são pausas temporárias na reprodução do vídeo ocorridas devido o *playout buffer* não conter segmentos suficiente para reproduzi-lo. De uma forma geral, uma das possíveis causas do esgotamento do *playout buffer*, deve-se aos congestionamentos que afetam o elance de acesso ao cliente. Segundo Yining Qi e Mingyuan Dai (QI; DAI, 2006), quanto maior a duração de uma interrupção, maior o impacto na qualidade assistida. Ainda segundo os autores, usuários de sistemas VoD preferem uma única e longa interrupção do que várias de curtas duração. Assim, o *playout buffer* é um dos mecanismos que auxilia a mitigar esses eventos. Quanto maior o *playout buffer*, maior o tempo que o vídeo fica paralisado aguardando o seu preenchimento. Entretanto, em eventos ao vivo, como o *playout buffer* possui de 2 a 3 segundos de vídeo, o algoritmo de adaptação do cliente DASH deve reagir de forma rápida e dinâmica aos congestionamentos na rede.

3.1.3 Estratégias de Adaptação de Vídeo

Impactos na qualidade do vídeo podem ser percebidas de acordo com as trocas e as amplitudes dos segmentos realizadas ao longo transmissão. O tamanho do segmento utilizado pelo servidor, a quantidade de mudanças entre as qualidades e as amplitudes

dessas mudanças (e.g. a troca de um segmento de melhor qualidade para um de pior qualidade), influenciam na percepção do usuário. Utilizar segmentos de curta duração (e.g. 1 segundo) resulta em uma maior quantidade de arquivos a serem armazenados no servidor, porém permite que o cliente DASH realize mudanças na qualidade em casos de congestionamentos na rede (SEUFERT et al., 2015). Segmentos maiores podem não ser adequados quando ocorrem mudanças abruptas no estado de congestionamento da rede, levando à paralisações na reprodução do vídeo. Em sistemas ao vivo, devido ao menor tamanho do *playout buffer* do cliente, segmentos menores são mais recomendados, uma vez que o DASH deve reagir de forma mais rápida às flutuações no congestionamento da rede.

De acordo com os estudos de Lewcio et al. (LEWCIO et al., 2011), trocas entre segmentos de diferentes taxas de qualidade acabam por degradar ou melhorar a percepção da imagem, dependendo do tipo da troca. Os autores argumentam que a frequência entre as adaptações de qualidade deve ser a menor possível e em caso dessa variação não ser evitada, a amplitude entre essas trocas deve ser a mais suave possível. Mudanças abruptas para segmentos de melhor qualidade podem aumentar a percepção de qualidade do usuário, uma vez que eles se sentem felizes pela sensação da melhora visual. Por fim, os autores também descrevem que segmentos de melhor qualidade reproduzidos ao final do vídeo tendem a aumentar a qualidade percebida pelo usuário.

3.2 Adaptações no Cliente

A fim de realizar requisições de segmentos, as decisões mais importantes que o algoritmo adaptativo (ABR, Adaptive Bitrate) do cliente DASH deve tomar, é em relação a quais segmentos buscar e como gerenciar a ocupação do seu *playout buffer*. Baseado na ocupação do *playout buffer* e na taxa de transmissão disponível na rede, o algoritmo ABR deve selecionar a representação apropriada a fim de maximizar a qua-

lidade do vídeo assistido (SEUFERT et al., 2015). De acordo com Kua et al. (KUA; ARMITAGE; BRANCH, 2017), o algoritmo ABR pode ser classificado em 3 categorias, de acordo com sinal de *feedback* que ele usa, sendo: a) baseado no *throughput* (*throughput-based*), b) baseado no *buffer* (*buffer-based*) ou c) híbrido (*hybrid/control theory-based*).

3.2.1 Adaptações Baseadas no Throughput

Para selecionar o próximo segmento de vídeo, esta classe de algoritmo ABR baseia-se na estimativa do *throughput* da rede feito pelos protocolos TCP e HTTP. O método proposto por Liu et al. (LIU; BOUAZIZI; GABBOUJ, 2011), realiza uma medida suavizada do *throughput* através desses protocolos. Caso a a taxa de transmissão disponível seja suficiente para a requisição de segmentos de maior qualidade, o método ABR desenvolvido por eles realiza uma subida conservadora, escalonando a qualidade dos segmentos uma a uma. No caso de um decremento na largura de banda do enlace, o método opta por uma troca agressiva para a representação de menor qualidade, a fim de garantir a continuidade de reprodução do vídeo. De acordo com os autores, os resultados obtidos mostram que o método é eficiente em detectar a taxa disponível na rede, bem como congestionamentos.

O trabalho desenvolvido por Jiang et al. (JIANG; SEKAR; ZHANG, 2014) foi projetado para prover justiça, eficiência e estabilidade quando múltiplos clientes DASH compartilham o mesmo enlace. Segundo os autores, neste cenário, o sinal de *feedback* que o algoritmo ABR recebe da rede não reflete o estado real da rede, fazendo com que uma seleção tendenciosa dos segmentos seja realizada pelo cliente. Deste modo, o algoritmo ABR proposto por eles realiza um processo de três passos antes de requisitar um segmento, que são: (1) um escalonador agenda a requisição do próximo segmento; (2) o algoritmo seleciona a taxa de representação adequada para cada segmento e (3) uma estimação da largura de banda da rede é feita. Para a taxa de transmissão disponível na rede, o algoritmo proposto computa a média harmônica do

throughput dos 20 últimos segmentos. Um selecionador da taxa de representação dos segmentos recebe o *throughput* estimado e calcula qual a representação do próximo segmento a ser transferido. O escalonador agenda, com um atraso aleatório, a requisição do próximo segmento. Caso a quantidade de segmentos no *playout buffer* seja menor que um valor alvo definido, o escalonador imediatamente requisita o próximo segmento. Os autores comparam o método proposto com vários outros métodos disponíveis. De acordo com eles, o algoritmo intitulado de FESTIVE melhora em 40% a justiça entre os fluxos, em 50% a estabilidade e em 10% a eficiência do enlace, sendo robusto quando vários usuários compartilham a mesma largura de banda. O algoritmo possui baixa carga computacional de implementação no cliente, não necessitando de modificações na rede ou no servidor.

3.2.2 Adaptações Baseadas em Buffer

Algoritmos ABR baseados no *buffer* do cliente DASH requisitam segmentos de vídeo levando em consideração a ocupação do *playout buffer*.

Huang et al. (HUANG et al., 2014) propõe um algoritmo ABR que usa a ocupação atual do *playout buffer* para decidir qual a qualidade do segmento a ser requisitado. Para isso, o *playout buffer* proposto pelos autores contém 3 limiares: $B < B_m < B_{max}$, além do valor da taxa de representação do segmento de menor qualidade (R_{min}) e de maior qualidade (R_{max}). Assim que a transmissão começa, o algoritmo ABR requisita os segmentos com R_{min}, a fim de preencher o *playout buffer* até o limiar B. Ao atingir B, o cliente DASH passa a requisitar segmentos de acordo com $f(B)$, que pode ser qualquer função matemática, selecionando segmentos entre R_{min} e R_{max}, até que o limiar B_m seja alcançado. De B_m até B_{max}, o método proposto seleciona os segmentos com a maior qualidade. Assim, o *playout buffer* deve ser mantido com segmentos acima do limiar B, de forma a ter segmentos suficientes para absorver as variações na largura de banda da rede. Segundo os autores, o algoritmo foi testado nos clientes DASH da empresa Netflix, apresentando uma redução de 10 a 20% na quantidade de

rebuffers quando comparado ao algoritmo padrão da empresa.

No algoritmo de adaptação e gerenciamento de *buffer* ABMA (*Adaptation and Buffer Management Algorithm*) proposto por Beben et al. (BEBEN et al., 2016), a seleção da qualidade do segmento a ser requisitado é baseada na previsão da probabilidade do vídeo ser interrompido, levando em consideração a ocupação do *buffer*. Além disso, o algoritmo continuamente estima o tempo que um segmento levou para ser transferido. De acordo com os autores, o ABMA é mais eficiente em ajustar a qualidade do vídeo de acordo com a variação da taxa disponível na rede, minimizando as interrupções na reprodução do vídeo e prevenindo frequentes trocas de qualidade dos segmentos, quando comparado com outras abordagens.

3.2.3 Adaptações Híbridas Baseadas na Teoria de Controle

Os algoritmos ABR híbridos baseados em teoria de controle, utilizam tanto a ocupação do *playout buffer*, como a largura de banda disponível no enlace, como indicadores para medir o nível de ocupação da rede. A fim de decidir qual qualidade de segmento requisitar, esses métodos fazem uso da teoria de sistema de controle.

Por exemplo, o trabalho publicado por Zhou et al. (ZHOU et al., 2013) tem como objetivo controlar a ocupação do *playout buffer* do cliente DASH, mantendo-o próximo de um valor limiar definido. O método proposto realiza a requisição de segmentos levando em consideração a diferença entre a ocupação atual do *playout buffer* e do limiar, em conjunto com o *throughput* estimado da rede.

3.3 Adaptações no Servidor

De modo a fornecer conteúdo de vídeo em diferentes taxas de bits e qualidades, alguns autores propõe mudanças no servidor DASH. Essas modificações podem ser em relação a forma como o vídeo é codificado, variando, por exemplo, a taxa de quadros

por segundo, a resolução do vídeo (de 1920x1080 até 128x72) ou a taxa de qualidade, comumente referenciado pela literatura como parâmetro de quantização (QP, *Quantization Parameter*).

Variações baseadas na mudança da taxa de quadros por segundo, podem diminuir ou aumentar a qualidade do vídeo percebida pelo usuário. Reduzindo a taxa de quadros por segundo que um vídeo é codificado, reduz-se a quantidade de bits para transmiti-lo. Neste caso, movimentações intensas no vídeo podem não ser mais percebidas pelo usuário, degradando substancialmente a qualidade (SEUFERT et al., 2015). Outro tipo de adaptação pode ser a variação na resolução do vídeo, onde decrementos diminuem a quantidade de pixels da imagem, reduzindo o tempo de transmissão dos segmentos e a quantidade de bits. Mesmo pequenas mudanças na resolução do vídeo podem reduzir ou aumentar significativamente a quantidade de bytes à serem transmitidos. Ajustes na taxa de qualidade dos segmentos de vídeo podem ser realizadas a fim de se obter o *throughput* desejado ao custo de afetar diretamente a qualidade percebida pelo usuário.

Akshabi et al.(AKHSHABI et al., 2013) propõe um método de adaptação no servidor baseado em *traffic shaping*, de modo a reduzir as oscilações e instabilidades do enlace quando vários usuário competem pela mesma largura de banda. Neste cenário, o padrão ON-OFF dos clientes DASH na busca de segmentos pode levar a uma sobrestimação da largura de banda e oscilações nas requisições da qualidade dos segmentos. Desta forma, eles desenvolveram um método de *shaping* que limita o *throughput* de cada segmento de vídeo de acordo com a sua taxa de codificação. Assim, o tempo de transferência do segmento tende a ser próximo do seu tempo de reprodução, reduzindo/eliminando os períodos OFF no enlace. Segundo os autores, o método estabiliza o *playout buffer* dos clientes e permite que eles requisitem os segmentos da maior qualidade possível sem causar oscilações na rede. Como o método é baseado em *traffic shaping*, uma sobrecarga extra é adicionada ao servidor de vídeo. Portanto, o servidor só ativa o método quando ele detecta oscilações no enlace.

Satoda et al. (SATODA et al., 2012) desenvolveram um algoritmo que verifica a transmissão do vídeo de forma a entregar segmentos “a tempo” (*just-in-time*) ao cliente, levando em consideração a ocupação do seu *playout buffer*. Chamado de *Zippy Pacing*, o método atrasa o envio dos segmento caso o *playout buffer* do cliente tenha atingido um certo limite. Ao início da transmissão, o servidor envia segmentos sem atrasos, até que o *buffer* de reprodução do cliente atinja uma quantidade suficiente de dados. Após, o método calcula um valor de atraso, chamando de *pacing delay*, de modo a manter a ocupação *playout buffer* próximo do valor alvo configurado, através do controle de envio dos segmentos. Para tal, a predição futura do *throughput* da rede é realizada através de um modelo estocástico. De acordo com os autores, o método aborda o problema do desperdício de largura de banda do enlace, causado pelo envio desnecessário de segmentos, quando os usuários abandonam o vídeo antes do fim.

3.4 Adaptações no Nível de Transporte

A pontualidade na entrega de um segmento de vídeo pode variar amplamente dependendo da largura de banda do enlace, do gerenciador de fila e da quantidade de outros fluxos concorrendo no mesmo enlace. Servidores DASH utilizam o protocolo TCP de modo a manter uma estimativas de quantos bytes estão em transmissão e sem reconhecimento de chegada pelo cliente. O protocolo TCP conta com uma janela de congestionamento (*cwnd*), que incrementa o número de pacotes a serem enviados a medida que o cliente vai reconhecendo as chegadas e avisando o servidor, que decai para seu valor inicial quando um pacote é descartado ou a conexão fica ociosa por muito tempo. No caso da *cwnd* ser muito pequena ou demorar a crescer, o cliente DASH pode experienciar um baixo *throughput* para o segmento requisitado. Caso a *cwnd* exceda a limite da capacidade do enlace, a quantidade de pacotes enviados pode aumentar significativamente a quantidade de bytes na fila do roteador, causando descartes e aumento no atraso.

Alguns trabalhos têm sido desenvolvidos para modelar e entender como o comportamento do tráfego gerado pelas aplicações afetam o TCP. Ghobadi et al. (GHOBADI et al., 2012) propuseram um mecanismo que usa o TCP para limitar o tráfego de vídeo, de modo a evitar rajadas de pacotes, o que causa congestionamentos nas filas e perca de pacotes. Chamado de *Trickle*, o mecanismo controla o fluxo do vídeo enviado ao cliente DASH, pela adição de um limite à janela *cwnd*, limite esse calculado em função do RTT (*Round Trip Time*). Os autores avaliaram o método colocando-o em atividade nos data centers do Youtube na Europa e na Índia. Eles analisaram o impacto do método com relação à perda de pacotes, largura de banda, RTT e eventos de re-preenchimento do *playout buffer* do usuário. De acordo com os autores, os resultados mostram que o *Trickle* reduz a taxa de perdas de pacotes em 43%.

McQuistin et al. (MCQUISTIN; PERKINS; FAYED, 2016a, 2016b) propuseram uma variação do protocolo TCP para suportar tráfego multimídia em tempo real. Intitulado de TCP *Hollywood*, a nova versão remove o bloqueio de HoL (*Head-of-Line*) tanto no servidor quando no cliente, entregando dados para a aplicação imediatamente, independente da ordem. Apenas dados que podem chegar a tempo de serem reproduzidos no cliente serão transmitidos. Neste caso, se o mecanismo estimador de RTT indicar que o pacote irá chegar demasiadamente atrasado no cliente, ficando inutilizado, ou que o pacote atual depende de pacotes passados não entregues corretamento, o TCP *Hollywood* envia novos pacotes ao invés de tentar retransmitir dados sem utilidade. Nessa versão, a pilha TCP/IP é modificada a fim de suportar entregas de dados fora de ordem, podendo ser habilitada ou desabilitada por *socket*.

3.5 Adaptações no Nível de Rede

A fim de entregar fluxos de vídeos contínuos e de boa qualidade a seus usuários, os provedores de Internet (ISP, *Internet Service Provider*) precisam garantir rigorosos parâmetros de QoS. Mesmo em redes com núcleos bem planejados, a última milha ainda

sofre com congestionamentos, devido ao constante aumento de demanda por maiores taxas de transmissão e o problema econômico de se implementar linhas de alta velocidade nestes segmentos de rede. Ambos os fatores acabam ocasionando gargalos nas redes de acesso, aumentando o atraso e os descartes de pacotes, impactando negativamente na qualidade de transmissões de eventos ao vivo.

As filas dos roteadores são os locais onde os congestionamentos são detectados. Tendo como base essa premissa, um algoritmo AQM obtém informações mais exatas e rápidas na detecção de congestionamentos do que as fontes que enviam tráfego (ADAMS, 2013). Como forma de avisar as fontes de tráfego sobre congestionamentos, o algoritmo AQM pode enviar uma notificação ECN ou descartar pacotes.

O RED foi o primeiro AQM proposto, com o objetivo de fazer com que as fontes de tráfego TCP diminuam suas taxas de transmissão a fim de mitigar o impacto do congestionamento. O RED monitora o tamanho da fila através de uma média móvel exponencial (EWMA, *Exponential Weighted Moving Average*) e realiza a indicação de congestionamento baseado em dois limiares, o min_{th} e o max_{th}. Se o valor médio do tamanho da fila (avg_q) estiver abaixo de min_{th}, não ocorrem descartes de pacotes (FLOYD; JACOBSON, 1993). Quando avg_q passa a ser maior que min_{th}, porém menor que max_{th}, os pacotes são descartados com probabilidade ρ_a, dada por: $\rho_a = \rho_b/(1 - count \cdot \rho_b)$, sendo $\rho_b = max_p((avg_q - mint_{th})/(max_{th} - min_{th}))$, onde max_p é a probabilidade máxima de descarte. A variável $count$ é incrementada a cada nova chegada de pacote, retornando a zero cada vez que um é descartado. Por fim, se avg_q exceder max_{th}, todos os pacotes que chegam são descartados. O Algoritmo 1 apresenta a operação do RED.

O RED possui dois problemas fundamentais. O primeiro é o fato do *throughput* do enlace depender de ajustes em seus parâmetros e do volume de tráfego que chega à fila (FLOYD; GUMMADI; SHENKER, 2001). O segundo ocorre quando o tamanho médio da fila ultrapassa o limiar max_{th}, reduzindo o *throughput* e aumentando consideravelmente o descarte de pacotes.

Algoritmo 1: ALGORITMO DO RED

w_q: Peso ajustado de maneira empírica
q: Tamanho instantâneo da fila
início
  para cada *Chegada de Pacote* **faça**
    Calcula o tamanho médio da fila:
      $avg_q = (1 - w_q) \cdot avg_q + w_q \cdot q$
    se $min_{th} < avg_q < max_{th}$ **então**
      $\rho_b = max_p((avg_q - min_{th})/(max_{th} - min_{th}))$
      $\rho_a = (\rho_b)/(1 - count \cdot \rho_b)$
      Marca Pacote com probabilidade ρ_a
    fim
    senão se $avg_q \geqslant max_{th}$ **então**
      Marca o pacote que chegou
    fim
  fim
fim

Como alternativa, o algoritmo ARED (*Adaptive Random Early Detection*) foi proposto por Feng *et al.* (FENG et al., 1999). Nele, a probabilidade máxima de descarte max_p é adaptada de acordo com o tamanho instantâneo da fila, de modo a manter avg_q longe do limiar max_{th}, melhorando o *throughput*, reduzindo o descarte de pacotes e aumentando a robustez. Posteriormente modificado por Floyd *et al.* (FLOYD; GUMMADI; SHENKER, 2001), o ARED realiza adaptações mais frequentes de max_p, mantendo avg_q dentro de um valor médio entre min_{th} e max_{th}. Como resultado, o ARED é capaz de manter o tamanho da fila em um estado estável. Segundo Floyd et al., o algoritmo não é uma solução ótima, porém apresenta um bom funcionamento para uma vasta quantidade de cenários (FLOYD; GUMMADI; SHENKER, 2001). O Algoritmo 2 mostra a lógica de funcionamento do ARED. A adaptação de max_p como forma de manter o tamanho médio da fila dentro de um certo limiar é uma das questões do RED abordadas pelo ARED. Em enlaces congestionados, ambos os algoritmos induzem um alto nível de atraso na fila, aumentam a quantidade de descarte de pacotes e não são eficientes em manter uma boa taxa de *throughput*. A fim de resolver esses problemas, Patel e Karmeshu (PATEL; KARMESHU, 2019), sugerem um método AQM fundamentado no RED e no ARED, porém com diferente função de probabilidade de descarte. No método proposto por eles, se avg_q estiver entre min_{th} e max_{th}, os paco-

Algoritmo 2: Adaptive RED (FENG et al., 1999)

α : Fator de Decremento; $min(0.01, max_p/4)$
β : Fator de Incremento; 0.9
início
  para cada *Atualização de* avg_q **faça**
    se $min_{th} < avg_q < max_{th}$ **então**
      status = *Between*;
    fim
    se $avg_q < min_{th}$ $\&\&$ $status! = Below$ **então**
      status = *Below*;
      $max_p \leftarrow \frac{max_p}{\alpha}$;
    fim
    se $avg_q > max_{th}$ $\&\&$ $status! = Above$ **então**
      status = *Above*;
      $max_p \leftarrow max_p * \beta$;
    fim
  fim
fim

tes são descartados com probabilidade dada por $p_2 = 1 - \{p_1[-log(p_1)]/(count + 1)\}$, com $p_1 = p_b$. Os resultados apresentados pelos autores mostram que o método previne que o tamanho da fila exceda max_{th}. O método mantém avg_q em níveis menores quando comparado ao RED e ARED, devido a probabilidade de descarte ser melhor estimada. Com isso, o atraso fim-a-fim é diminuído, bem como o congestionamento na fila, trazendo melhorias para o *throughput*. Porém, o método necessita que os valores dos limiares min_{th} e max_{th} sejam cuidadosamente ajustados para que o AQM atinja as melhorias mencionadas.

O *bufferbloat* é um fenômeno provocado pelo acréscimo demasiado de memória nos *buffers* dos roteadores ao longo da Internet, sem a devida consideração ou testes (GETTYS; NICHOLS, 2011). Ele é responsável por degradar parâmetros como atraso e *jitter*, repercutindo negativamente em diversas aplicações. O CoDel (*Controlled Queue Delay*) (NICHOLS; JACOBSON, 2012; NICHOLS et al., 2018) é um AQM projetado para reduzir os efeitos do *bufferbloat*. O método identifica o congestionamento baseado no monitoramento do tempo que cada pacote permanece na fila. O CoDel usa duas variáveis principais: $target$ e $interval$, ambas ajustadas em função do RTT (*Round Trip Time*), cujo valor típico para redes terrestres cabeadas é de 100 ms

(DISCHINGER et al., 2007). De acordo com a RFC 8289 (NICHOLS et al., 2018), os valores ideais de ajuste para $target$, variam de 5 a 10% do RTT. A RFC adota $target$ e $interval$ como sendo 5 e 100 ms, respectivamente. A cada pacote que chega a fila, o AQM adiciona uma estampa de tempo, como forma de monitorar o atraso da fila. Quando esse valor excede $target$, o algoritmo entra em estado de descarte, aguardando pelo tempo de $interval$, antes de excluir um único pacote. O algoritmo altera o valor de $interval$ de modo proporcional ao inverso da raiz quadrada do número de pacotes descartados. A medida que o número de descartes aumenta, $interval$ diminui. Assim, a sequência padrão da variável $interval$, quando o CoDel está em modo de descarte, é dada por: $100, 100/\sqrt{2}, 100/\sqrt{3}, \ldots$. O CoDel sai do modo de descarte assim que o atraso da fila voltar a ser menor que $target$, fazendo com que $interval$ retorne ao seu valor padrão. Segundo os autores, o CoDel torna-se atrativo para o gerenciamento de filas por ser de fácil e eficiente implementação, além da ausência da necessidade de ajustes de parâmetros (NICHOLS; JACOBSON, 2012). Os resultados mostram que o CoDel atinge melhor desempenho para a taxa de utilização do enlace e do atraso quando comparado ao RED. Para algumas situações, o RED até alcança menores valores de atraso, porém ao custo de uma menor utilização do enlace.

Uma extensão do CoDel foi publicada em (HOEILAND-JOERGENSEN et al., 2018), com o título de FQ-CoDel (*Flow Queue-CoDel*). O algoritmo classifica os pacotes que chegam em uma das 1024 filas disponíveis, selecionando-os de acordo com os endereços IP e portas de origem e destino. Cada fila é separadamente atendida pelo CoDel. Um escalonador DRR (*Deficit Round Robin*) modificado, retira os pacotes das filas, dando prioridade aquelas que contém fluxos recém chegados. Conforme os autores, o algoritmo é uma ferramenta poderosa e de baixa complexidade computacional, para a solução dos problemas causados pelos *bufferbloat*.

O PIE (*Proportional Integral Controller Enhanced*) (PAN et al., 2017) é um método AQM que combina os benefícios das funcionalidades de descarte randômico com o mecanismo de detecção de congestionamento em função do atraso médio da fila.

O algoritmo do PIE baseia-se em três módulos: (a) a estimação da taxa de saída dos pacotes da fila, (b) a atualização periódica da probabilidade de descarte e (c) o descarte aleatório de pacotes no enfileiramento. A taxa de saída (a) é estimada sempre que o AQM identifica congestionamentos, ou seja, quando o tamanho da fila (q_len) exceder um certo limiar (dq_th). Nesse estado, a cada pacote transmitido, seus bytes são contabilizados (dq_count). Caso dq_count seja maior que dq_th, a taxa média de saída (avg_drate) é computada, levando em consideração o tempo total para transmitir os dq_count bytes. A probabilidade de descarte (b) é periodicamente atualizada e determinada em função do valor atual do atraso (cur_del), anterior e de referência (ref_del), sendo cur_del obtido em função de q_len e avg_drate. O PIE é caracterizado como um controlador Proporcional Integral, sendo o termo Proporcional o erro estimado entre o valor atual do atraso e o valor do *set point* e o termo Integral, o erro entre os valores do atraso atual e anterior. O PIE utiliza as tendências de aumento ou diminuição do atraso para ajudar na detecção do congestionamento e manter a fila dentro de um certo limiar de atraso, aumentando ou diminuindo a probabilidade de descarte. Conforme ela é atualizada, novos pacotes são descartados aleatoriamente, com a probabilidade de descarte dada por:

$$\rho = \rho + \alpha \cdot (cur\_del - ref\_del) + \beta \cdot (cur\_del - old\_del), \tag{3.1}$$

sendo α e β fatores de escalonamento que determinam o equilíbrio entre o *set point* e a variação do atraso. Ambos os parâmetros são ajustados automaticamente, tornando o AQM estável e de resposta rápida à mudanças repentinas no estado de congestionamento da rede. Rajadas de pacotes são toleradas pelo PIE. Segundo os autores, como a probabilidade de descarte é atualizada periodicamente, rajadas de curta duração podem ocorrer dentro desse período sem sofrerem com descartes extras, pois não implicaria em um aumento dessa probabilidade. Os resultados mostram que o PIE mantém o atraso da fila em torno do valor de referência, bem como a utilização do enlace fica próximo da sua capacidade máxima. Em comparação com o CoDel, os

autores mencionam que ambos os AQMs desempenham ótimos resultados, com uma leve vantagem para o PIE, onde 70% dos pacotes passam por atrasos menores do que o limiar de referência (5 ms), contra apenas 30% do CoDel.

Métodos AQMs tais como o PIE e o CoDel são progressivamente desenvolvidos tanto na rede de acesso ao cliente das ISPs quando no roteador residencial do cliente (*home gateway*), de modo a prevenir os efeitos do *bufferbloat*. A fim de explorar como CoDel e PIE interagem com o tráfego DASH VoD, Kua *et al.* (KUA; ARMITAGE; BRANCH, 2016) avaliaram o impacto causado por esses algoritmos, simulando o acesso dos clientes à conteúdos remotos, a partir de redes residenciais, com diversos fluxos competindo pela largura de banda. Os resultados apresentados são em função da variação da qualidade dos segmentos de vídeo em relação ao aumento do RTT, como forma de reproduzir a perturbação causada pelo congestionamento na rede entre servidor e cliente. De acordo com os autores, ambos os AQMs alcançam resultados similares, fazendo com que o cliente DASH requisite segmentos de menor qualidade a medida que o RTT aumenta. Os autores não comparam o CoDel e o PIE com o RED, nem apresentam a quantidade de descartes de pacotes observada. A qualidade dos vídeos recebidos também não foi avaliada.

Bouten *et al.* (BOUTEN et al., 2014), propõe um algoritmo de priorização do fluxo de vídeo DASH VoD, explorando as camadas do codec H.264/SVC (*Scalable Video Coding*), que é uma extensão do H.264/AVC. Como descrito no Capítulo 2, na codificação SVC, os quadros I e P formam a camada base e servem de referência para os quadros B pertencentes as camadas superiores. Os quadros pertencentes à última camada não servem de referência de codificação para outros quadros. Em casos de congestionamentos, eles podem ser priorizados para descartes. Assim, de acordo com o trabalho de Bouten *et al.*, quando o cliente está próximo de passar pela falta de segmentos em seu *playout buffer*, as próximas requisições realizadas solicitam que a camada base da codificação hierárquica seja priorizada na fila do roteador, de forma que a reprodução contínua do vídeo seja mantida no cliente. Através de um proxy,

disposto antes do servidor DASH, as mensagens de requisição HTTP são interceptadas e interpretadas. Os pacotes contendo o fluxo de vídeo são marcados pelo proxy no campo DSCP (*Differentiated Services Code Point*) do protocolo IP, e alocados em filas distintas no roteador. Uma combinação de classificador IP, divisor de largura de banda, AQM RED e escalonador prioritário, é aplicado no roteador, de modo que o tráfego alocado na fila de mais alta prioridade não consuma toda a largura de banda do enlace, garantindo justiça no atendimento entre as filas. Caso segmentos estejam com seus prazos de reprodução expirados, pelo fato de terem sofrido atrasos demasiados na fila, a conexão *socket* TCP é cancelada entre o servidor e o cliente, evitando consumo ineficaz da largura de banda do enlace. Para fins de comparação, os autores criaram um cenário onde o fluxo de vídeo é codificado com H.264/AVC e alocado em uma fila única sem prioridade. Os resultados mostram que, para clientes configurados com *playout buffer* de 2 segundos o método proposto permite a reprodução contínua do vídeo, enquanto que, para o fluxo codificado com H.264/AVC e alocado em uma única fila, ocorrências de congelamento na imagem são percebidas. Mesmo em clientes configurados com 20 segundos de *playout buffer*, o cenário sem prioridade não é capaz de proteger o tráfego AVC de imperfeições na imagem. Os autores mencionam que o método permite reduzir significativamente o tamanho do *playout buffer*, diminuindo o atraso na reprodução e mantendo a estabilidade. Porém, existe a necessidade da instalação de um proxy na topologia da rede e a análise dos cabeçalhos HTTP dificulta a implementação prática da proposta. Além disso, o servidor proxy representa um possível gargalo no sistema.

Similar ao trabalho anterior, o método proposto por Kua e Armitage (KUA; ARMITAGE, 2017) utiliza um proxy próximo a rede do cliente, como forma de enviar requisições de segmentos DASH VoD em paralelo ao servidor. Chamado de *Chuncklet*, o método faz com que o proxy intercepte as mensagens de requisição HTTP do cliente e analise qual segmento está sendo solicitado ao servidor. O proxy fragmenta a mensagem de requisição em N partes, enviando múltiplas requisições ao servidor. No roteador central da rede, o AQM FQ-CoDel, separa cada uma das múltiplas conexões

em diferentes filas. Conforme as N repostas chegam do servidor, o proxy concatena-as, recriando um único segmento para envio ao cliente. Os autores citam que as ações do proxy são transparentes ao cliente e servidor DASH, não necessitando de configuração extras em ambos. Os resultados mostram que o cliente DASH é capaz de alcançar maiores porções da largura de banda do enlace quando concorrendo com fontes de tráfego do tipo TCP *Bulk*. Os autores avaliam o método substituindo o CoDel pelo PIE e Droptail. Quando o valor de N é menor que 6, o PIE obtém melhores resultados, porém com maior variação entre as qualidades dos segmentos do que o CoDel. A medida que o valor de N aumenta, o CoDel consegue requisitar segmentos de maior qualidade do que os outros AQM, sendo mais estável. Assim como no trabalho anterior, esse método depende da implementação de um proxy no enlace, podendo levar a problemas de escalabilidade - como o servidor de proxy proposto está mais próximo da rede de acesso, o custo de implementação é mais baixo devido ao menor tráfego. No entanto, devem ser implementados diversos servidores de proxy para atender a ampla quantidade de clientes, resultando em dificuldade de implantação.

O trabalho proposto por Abbas *et al.* (ABBAS; MANZOOR; HUSSAIN, 2018), apresenta um método AQM para melhorar a justiça entre fluxos TCP e UDP. O método identifica e penaliza fluxos UPD, devido a sua característica não amigável, os quais continuam enviando pacotes mesmo em situações de congestionamentos. Intitulado de CHOKeH, o método apresenta funcionamento similar ao RED, usando o tamanho médio da fila (avg_q) para medir o nível de congestionamento. Para cada pacote que chega, se avg_q estiver entre min_{th} e max_{th}, o método divide o tamanho atual da fila (q) em duas regiões de tamanhos iguais: r_q (*rear queue*) e f_q (*front queue*). Iniciando pela região r_q (região mais propensa a concentrar fluxos não colaborativos), o CHOKeH sorteia aleatoriamente d pacotes para descartes, onde d é em função de q e da capacidade total do *buffer* (B). Assim, se os d pacotes sorteados possuem o mesmo *FlowId* que os pacotes que acabaram de chegar à fila, todos eles são descartados. Caso o *FlowId* não seja o mesmo, o CHOKeH seleciona para descarte $d/2$ pacotes, porém desta vez, da região f_q. Os resultados mostram que o CHOKeH atinge me-

lhores níveis de *throughput* e um comportamento mais estável de avg_q que os AQMs comparados. Ainda, o método não penaliza os fluxos colaborativos do tipo TCP.

Este Capítulo tratou de alguns dos principais métodos para melhoria da qualidade de vídeo para fluxos DASH referenciados pela literatura. Com o aumento gradativo do consumo de tráfego de vídeo na Internet, a necessidade de reduzir parâmetros como atraso e descarte de pacotes faz-se necessária, dada a sensibilidade desse tipo de tráfego. Como solução ao descarte de pacotes, as ISPs aumentaram demasiadamente o tamanho do *buffer* dos roteadores, degradando parâmetros como atraso e *jitter*, impactando negativamente em diversas aplicações, principalmente em aplicações de vídeo ao vivo. Modificações no cliente e no servidor DASH, bem como no protocolo TCP da camada de transporte, são propostas como forma de manter a reprodução contínua do vídeo ao cliente. Além deles, métodos AQM são propostos como forma de prevenir o problema do *bufferbloat*. Dentre eles pode-se citar o CoDel e o PIE. O surgimento do RED ocorreu algumas décadas atrás e o seu uso foi fortemente recomendado de modo a melhorar o desempenho da Internet (CLARK et al., 1998). Porém, a necessidade de ajustes de parâmetros de acordo com o volume do tráfego e o problema da redução do *throughput* quando o tamanho médio da fila ultrapassa o limiar max_{th}, proporcionaram o surgimento de novos AQMs tais como o ARED, o PIE, o CoDel, entre outros. De modo a melhorar *throughput* do RED, no ARED, a probabilidade máxima de descarte é adaptada de acordo com o tamanho instantâneo da fila. A proposta do CoDel consiste em atribuir uma estampa de tempo para cada pacote que chega a fila, de forma a monitorar o atraso. A medida que o atraso dos pacotes excede um certo limiar, o AQM descarta um pacote e agenda a exclusão do próximo. No PIE, a combinação entre descartes aleatórios com o monitoramento do tamanho médio da fila e um controlador Proporcional Integral, fazem com que o AQM mantenha o atraso da fila em torno de um valor de referência. Alguns autores exploram como o PIE e o CoDel se comportam com fluxo DASH VoD, porém eles não mostram qual

o impacto desses AQMs na qualidade do vídeo. Outros trabalhos propõe métodos de priorização de fluxos DASH VoD ou explorando as camadas da codificação SVC ou através da fragmentação das mensagens DASH de forma a manter a reprodução contínua do vídeo. Porém, esses algoritmos dependem de um proxy instalado no enlace. Apesar da maioria dos métodos atacar problemas como do *bufferbloat* na tentativa de minimizar o impacto do atraso em diversas tipos de aplicações, poucos deles foram pensados para colaborar com fluxos de vídeo DASH ao vivo. Como o tráfego de vídeo DASH apresenta características previsíveis e sendo a fila do roteador o local aonde os congestionamentos ocorrem, uma nova classe de AQM pode ser proposta. Esta tese explora a utilização da previsão do atraso para fluxos DASH ao vivo de forma a realizar descartes, pressionando o cliente DASH a reduzir a qualidade dos segmentos requisitados, mantendo a reprodução contínua do vídeo. A inserção de novos equipamentos na rede ou a alteração nas aplicações de cliente e servidor DASH, não seria necessária, reduzindo o custo de implementação.

CAPÍTULO 4

GERENCIAMENTO ATIVO DE FILAS PARA DASH AO VIVO ATRAVÉS DA PREVISÃO DO ATRASO

Este trabalho propõe dois métodos AQM que realizam o descarte antecipado aleatório de pacotes de acordo com o atraso previsto da fila (y_t). A variável y_t possui unidade em segundos e expressa o valor de previsão do atraso feito pela rede neural LSTM a cada δ segundos. Ambos os métodos usam a rede neural LSTM treinada, para realizar a predição *online* do atraso. O primeiro algoritmo proposto é o Descarte Randômico para Vídeo Adaptativo Baseado na Predição do Atraso (DRAPA) e o segundo é o Descarte Randômico para Vídeo Adaptativo Baseado na Predição da Tendência do Atraso (DRAPTA), que serão descritos a seguir.

4.1 Descarte Randômico para Vídeo Adaptativo Baseado na Predição do Atraso

No DRAPA, para cada pacote que chega a fila, se o atraso previsto pela rede LSTM (y_t) estiver entre low_{th} e mid_{th} o algoritmo realiza descarte aleatório de pacotes com probabilidade p_1, dado por:

$$p_1 = \frac{(y_t - low_{th})}{(mid_{th} - low_{th})} \cdot \Omega, \tag{4.1}$$

Caso o valor de y_t seja maior que mid_{th} porém menor que up_{th}, o AQM descarta pacotes aleatoriamente de acordo com p_2:

$$p_2 = \frac{[y_t - (\Omega \cdot y_t) - mid_{th} + (\Omega \cdot up_{th})]}{(up_{th} - mid_{th})}, \tag{4.2}$$

sendo Ω a intersecção entre as retas p_1 e p_2. Finalmente, caso y_t exceda o limiar up_{th}, todos os pacotes que chegam a fila serão descartados. Quando y_t for menor que low_{th}, os pacotes que chegam a fila são enfileirados. A lógica do AQM proposto é apresentada pelo Algoritmo 3.

Algoritmo 3: Algoritmo DRAPA.

```
A cada δ segundos:
    Executa a LSTM e atualiza y_t;
para cada pacote que chega na fila faça
    se low_th < y_t ≤ mid_th então
        calcula a probabilidade p_1 de acordo com eq. 4.1;
        n ← rand();
        se p_1 ≤ n então
            descarta o pacote;
        fim
    fim
    senão se mid_th < y_t ≤ up_th então
        calcula a probabilidade p_2 de acordo com eq. 4.2;
        n ← rand();
        se p_2 ≤ n então
            descarta o pacote;
        fim
    fim
    senão se y_t > up_th então
        descarta o pacote;
    fim
    senão
        Enfileira o pacote;
    fim
fim
```

As probabilidades de descarte p_1 e p_2 foram calculadas de acordo com a Figura 4.1. Nela, os três limiares são interligados por duas retas, gerando as equações de p_1 e p_2. Na Figura 4.1, o eixo das abscissas representa os valores do atraso previsto (y_t) e o eixo das ordenadas a probabilidade de descarte, com valor máximo em 1. As retas são intersectadas em mid_{th}, no eixo das ordenadas, e Ω no eixo das abscissas. Para valores de y_t maiores que up_{th}, todos os pacotes que chegam a fila são descartados. A reta p_1 possui uma maior inclinação quando comparado à reta p_2. De fato, pretende-se que a maior parte dos descartes seja realizado com probabilidade linear p_1, de modo a fazer com que o tráfego se adapte antes que o atraso da fila atinja os limiares

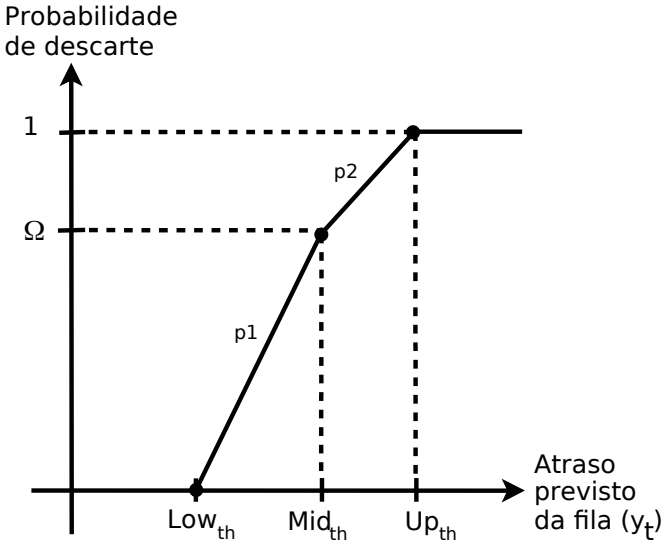

Figura 4.1: Retas das probabilidades de descarte p_1 e p_2.

superiores. Caso y_t seja maior que mid_{th}, a inclinação da reta p_2 reduz a probabilidade de descarte. Porém, mesmo com menor inclinação, os valores obtidos em p_2 estão próximos de 1, resultando em quantidades significativas de descarte de pacotes.

4.2 Descarte Randômico para Vídeo Adaptativo Baseado na Predição da Tendência do Atraso

No DRAPTA, o descarte de pacotes é realizado de acordo com o tempo de reação (TR) que resta ao AQM, antes que o atraso da fila atinja níveis elevados. No caso, TR simboliza a quantidade de tempo que o algoritmo possui antes que os valores do atraso atual (x_t) da fila ou o atraso previsto (y_t) cheguem próximos aos limiares da fila. Assim, pode-se dizer que esse método leva em consideração a tendência do atraso, reagindo com descartes de pacotes de acordo com a orientação atual e futura do atraso. A lógica desse AQM é apresentada pelo Algoritmo 4.

Para cada pacote que chega à fila, caso x_t esteja entre low_{th} e mid_{th}, o algoritmo calcula o TR disponível, antes que o atraso da fila atinja mid_{th}:

$$TR_{mid} = \frac{mid_{th} - x_t}{y_t - x_t}, \tag{4.3}$$

Algoritmo 4: Algoritmo DRAPTA.

```
A cada δ segundos:
    Executa a LSTM e atualiza y_t;
para cada pacote que chega na fila faça
    se (low_th < x_t ≤ mid_th) então
        Calcula Tempo de Reação: TR_mid;
        Atribui valor a Ω_d de acordo com TR_mid;
        Calcula p_d;
        n ← rand();
        se p_d ≤ n então
            descarta o pacote;
        fim
    fim
    senão se mid_th < x_t ≤ up_th então
        Calcula Tempo de Reação: TR_up;
        Atribui valor a Ω_d de acordo com TR_up;
        Calcula p_d;
        n ← rand();
        se p_d ≤ n então
            descarta o pacote;
        fim
    fim
    senão se x_t > up_th então
        descarta o pacote;
    fim
    senão
        enfileira o pacote;
    fim
fim
```

Através do valor encontrado em TR_{mid}, o algoritmo seleciona um peso (Ω_d), que será atribuído a função da probabilidade de descarte, dada por:

$$p_d = \left(1 - \frac{(up_{th} - x_t)}{(up_{th})}\right), \tag{4.4}$$

O descarte de pacotes é obtido através de $p_d \cdot \Omega_d$, sendo este último um valor adimensional que impulsiona ou diminui a probabilidade de um pacote ser descartado. A medida que TR_{mid} diminui, o valor atribuído a Ω_d aumenta, elevando a probabilidade de descarte. De fato, o decremento no valor de TR indica que x_t ou y_t estão se aproximando de mid_{th}, indicando que o atraso da fila irá aumentar. Aumentando o valor de Ω_d, o valor de p_d também é incrementado, intensificando o descarte dos próximos

pacotes que chegarem à fila. Da mesma forma, caso o valor de x_t esteja entre mid_{th} e up_{th}, o AQM calcula o tempo de reação disponível, porém, em relação ao limiar superior (up_{th}):

$$TR_{up} = \frac{up_{th} - x_t}{y_t - x_t}, \tag{4.5}$$

Novamente, a relação entre o limiar e os valores de x_t e y_t, selecionam o peso que será atribuído à probabilidade de descarte.

Dependendo do valor estimado para TR_{up}, caso o atraso tenha uma tendência de subida forte e um tempo de reação baixo, os valores atribuídos a Ω_d podem fazer com que a probabilidade p_d aumente de forma substancial. Isso faz como que as fontes TCP diminuam suas taxas de transmissão antes que o atraso da fila atinja níveis elevados. Por fim, se o valor de x_t for maior que up_{th}, os pacotes que chegam a fila são descartados. Quando os valores do atraso atual da fila estão abaixo de low_{th}, todos os pacotes que chegam são enfileirados. Tanto os valores de Ω_d para TR_{up} e TR_{mid} foram selecionados empiricamente através de testes. De acordo com a equação 4.4, p_d é dependente do limiar superior da fila (up_{th}) e do peso atribuído. O uso do limiar mid_{th} para o cálculo da probabilidade de descarte acaba inferindo em valores próximos de 1 (descarte de todos os pacotes que chegam a fila) antes mesmo que a fila chegue próximo de sua capacidade máxima.

4.3 Redes Neurais Recorrentes LSTM

A rede neural LSTM foi utilizada como ferramenta para previsão do atraso da fila em transmissão de vídeos DASH em tempo real, de modo a prever futuros congestionamentos. A justificativa de seu uso deve-se por ela exibir baixa complexidade computacional e ser usada com sucesso em vários trabalhos de previsão de séries temporais e tráfego de vídeo (AZZOUNI; PUJOLLE, 2017). A previsão de tráfego de vídeo ao vivo utilizando LSTM, supera os modelos matemáticos lineares, como o ARIMA e FARIMA (BARABAS et al., 2011; FENG; SHU, 2005). Além disso, esses modelos apresen-

tam complexidade computacional de $O(n^2)$, sendo n o tamanho da série de dados (KRUNZ; MAKOWSKI, 1998), superior ao da LSTM. De acordo Hua et. al. (HUA et al., 2019), as redes neurais de aprendizado profundo têm sido cada vez mais utilizadas na predição de séries temporais com características de longa duração. Em especial destacam-se as redes LSTM por possuírem a célula de memória, capaz de reter a informação por períodos de tempo mais longos.

A LSTM foi escrita em linguagem python fazendo uso da API (*Application Programming Interface*) Keras para redes de aprendizado profundo (GULLI; PAL, 2017). De modo a integrar a LSTM com os métodos proposto no NS-3, utilizou-se uma memória compartilhada. A cada δ segundos, os métodos propostos amostram o valor instantâneo do atraso da fila ($x_t = q_l/C$), sendo q_l o tamanho atual da fila em bytes e C a taxa de transmissão do enlace em bits por segundo. Esse valor é adicionado à memória compartilhada, ficando a disposição para que a rede LSTM atualize a sua entrada x_t. Assim que a previsão é realizada, a LSTM insere o valor do atraso previsto para a fila (y_t) na memória compartilhada, deixando o valor disponível para que os métodos propostos utilizem-no na decisão de descarte de pacotes. As entradas da LSTM são deslocadas no tempo, utilizando o conceito de janela de aprendizado (*Learning Window*) (AZZOUNI; PUJOLLE, 2017). Portanto, a cada δ segundos os valores das entradas da rede (x_t, x_{t-1} e x_{t-2}) são deslocados no tempo, como mostra a Figura 4.2. A saída da LSTM (y_t) indica a previsão para o próximo atraso na fila do roteador,

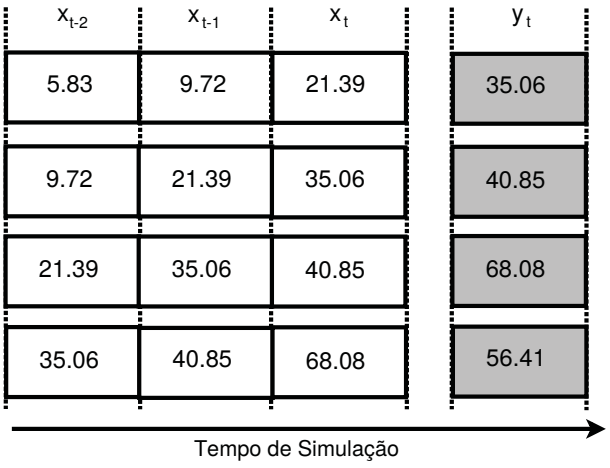

Figura 4.2: Padrão de entradas e saída para treinamento e validação da rede LSTM.

em segundos. A LSTM realiza a previsão do tráfego através do conhecimento adquirido com o conjunto de treinamento, generalizando a natureza do fluxo DASH para as previsões futuras.

A topologia escolhida para a rede LSTM consiste em uma camada de entrada com 3 unidades, uma camada escondida com 4 blocos de memória e uma única unidade de saída, responsável por prever o atraso da fila (BROWNLEE, 2017). Outras arquiteturas foram analisadas, utilizando um maior número de entradas e mais blocos de memória na camada escondida. Porém, essas topologias obtiveram resultados semelhantes ou inferiores à topologia escolhida, sendo seu uso injustificado por aumentar a complexidade computacional de execução da rede. Cabe lembrar que a LSTM, no pior caso, possui complexidade de $O(n)$, sendo n o número de pesos da rede. Aumentando a quantidade de entradas e dos blocos de memória da camada escondida, aumentam-se a quantidade de pesos da rede e sua complexidade computacional. A Figura 4.3 apresenta a arquitetura da rede utilizada para ambos os métodos AQMs desenvolvidos. Esse tipo de arquitetura é comumente chamado de Vanilla LSTM.

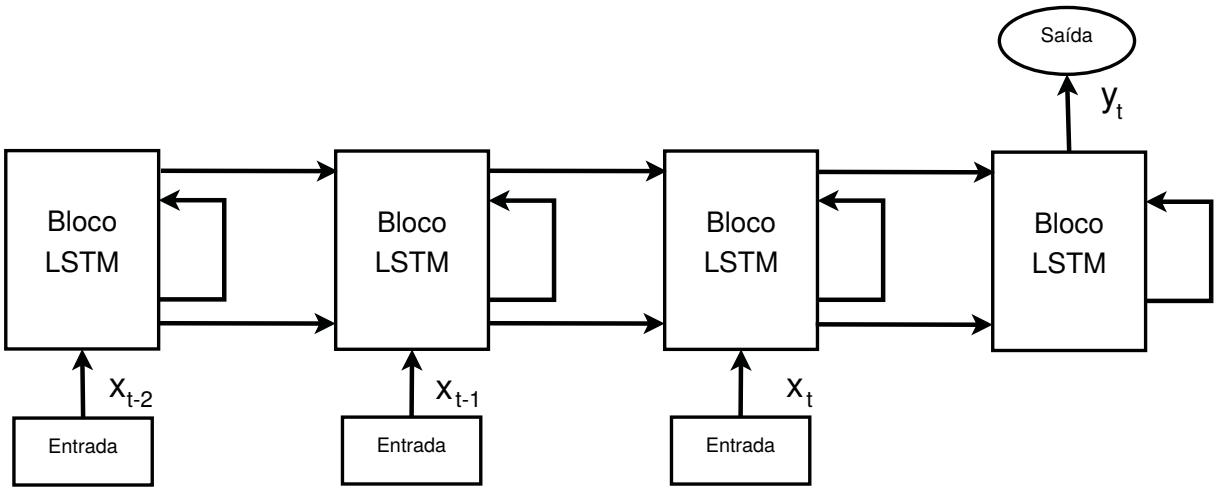

Figura 4.3: Modelo da rede LSTM utilizado para a previsão do atraso da fila.

A fim de treinar e validar o modelo da rede usado, um conjunto de dados contendo uma coluna com 60.000 amostras do atraso da fila, foi utilizado. Esses dados foram coletados através da execução do cenário simulado, utilizando 12 fontes de tráfego de fundo e o algoritmo de fila Droptail, com capacidade para 3 Gbytes, sendo o tamanho

médio dos pacotes de 1500 bytes. A capacidade excessiva da fila do Droptail evita que pacotes sejam descartados durante a simulação, permitido que o atraso da fila seja capturado sem alterações. O conjunto de dados foi divido em duas partes, 60% do total para treinamento da rede e os 40% restante para validação. A rede foi treinada de modo *offline* com 20 épocas, onde cada época significa uma iteração completa sobre os valores do conjunto de treinamento. A fim de otimizar o treinamento da rede, o algoritmo *Adam Optimization* foi o escolhido dentre os disponíveis. De acordo a publicação do Keras (GULLI; PAL, 2017), esse algoritmo é um método estocástico de gradiente descendente baseado na estimação adaptativa dos momentos de primeira e segunda ordem. Segundo Kingma et al. (KINGMA; BA, 2014), o método é computacionalmente eficiente, utilizando pouca memória e adequado para problemas que possuem grande base de dados.

4.4 Parametrização dos Métodos

De modo a ajustar os valores de low_{th}, mid_{th} e up_{th}, foi avaliado o impacto que o atraso causa na similaridade estrutural (SSIM, *Structural Similarity*). Os vídeos Touchdown Pass, Big Buck Bunny e Sunflower foram transmitidos em um cenário ponto a ponto simulado no NS-3. A escolha desses vídeos deve a diversidade que eles representam. No próximo Capítulo, as características estatísticas desses vídeos são apresentadas. A taxa de transmissão do enlace principal foi variada em 6 Mbps, 10 Mbps, 100 Mbps, 1 Gbps e 10 Gbps. O valor do atraso no enlace foi alterado de 50 a 310 milissegundos, sendo incrementado de 10 em 10 ms. Para cada taxa de transmissão e valor de atraso, o SSIM de cada vídeo foi registrado. Posteriormente, foi determinada a média geral do SSIM, denotado por $SSIM_{avg}$, de todos os vídeos, taxas de transmissão e para cada um dos valores de atraso. Por fim, o $SSIM_{avg}$ foi interpretado em função de uma aproximação de qualidade de vídeo, de acordo com a Tabela 4.1 e conforme sugerido por Zinner et al. (ZINNER et al., 2010).

Tabela 4.1: Mapeamento do SSIM em qualidade do vídeo (ZINNER et al., 2010).

Qualidade percebida	SSIM
Excelente	$SSIM > 0,99$
Bom	$0,95 \leq SSIM < 0,99$
Médio	$0,88 \leq SSIM < 0,95$
Ruim	$0,5 \leq SSIM < 0,88$
Péssimo	$SSIM < 0,5$

Conforme a Tabela 4.2, quando o atraso atinge o valor de 150 ms, o $SSIM_{avg}$ é alto $(0,983)$ e a qualidade percebida pelo usuário é boa, muito próximo de excelente. Assim, o valor do limiar low_{th} foi ajustado em 150 ms, ou seja, com atrasos menores que esse valor, não existe a necessidade de descarte de pacotes. A medida que o

Tabela 4.2: Impacto do atraso no SSIM médio e na qualidade percebida pelo usuário.

Delay	$SSIM_{avg}$	Qualidade Percebida	Delay	$SSIM_{avg}$	Qualidade Percebida
50	0,983	Bom	190	0,692	Ruim
60	0,923	Bom	200	0,779	Ruim
70	0,983	Bom	210	0,694	Ruim
80	0,981	Bom	220	0,634	Ruim
90	0,979	Bom	230	0,618	Ruim
100	0,979	Bom	240	0,587	Ruim
110	0,976	Bom	250	0,563	Péssimo
120	0,978	Bom	260	0,532	Péssimo
130	0,978	Bom	270	0,545	Péssimo
140	0,971	Bom	280	0,529	Péssimo
150	0,958	Bom	290	0,524	Péssimo
160	0,905	Médio	300	0,517	Péssimo
170	0,868	Ruim	310	0,429	Péssimo
180	0,856	Ruim	—	—	—

atraso ultrapassa os 150 milissegundos e continua crescendo, a qualidade percebida é reduzida de bom para ruim, este último em 240 ms, definindo o limite de mid_{th}. Atrasos acima de 240 milissegundos, a qualidade percebida cai para péssimo e se mantém assim quando este ultrapassa os 300 ms. Portanto, o limiar up_{th} foi definido em 300 ms. Para valores de atraso acima de 300 ms, todos os pacotes que chegam à fila, em ambos os métodos propostos, são descartados. Acima de 300 ms, a qualidade percebida pelo usuário é péssima.

A variável δ foi ajustada em 100 ms, proporcionando à rede neural LSTM tempo suficiente para fazer a previsão do atraso. Devido ao uso de um cenário híbrido, misturando tráfego simulado com real, valores de δ menores que 100 ms interferem nos fluxos ao vivo, reduzindo o desempenho do AQM. Assim, a cada 100 ms, o valor do atraso atual da fila foi amostrado pelo NS-3.

Para o algoritmo DRAPA, o valor de Ω foi ajustado em $0,8$. Para o DRAPTA, as variáveis TR_{mid}, TR_{up} e Ω_d assumem valores de acordo com a Tabela 4.3. Os valores atribuídos à Ω_d foram definidos de forma empírica. O princípio envolvido na obtenção dos valores de TR leva em consideração a maior tendência de subida do atraso obtido pela rede LSTM (y_t^{max}), para o conjunto de dados de treinamento, amostrados de δ segundos. Neste caso, y_t^{max} foi de 90 ms, isto é, o valor da previsão futura que pode ter o maior impacto na fila dependendo do nível que se encontra o atraso atual. Os valores de TR_{mid} e Ω_d foram obtidos partindo da premissa que x_t atingiu o limiar low_{th} e que y_t irá realizar previsões em passos de 10 ms até atingir y_t^{max}. Como exemplo, caso $x_t = 150ms$ e $y_t = 10ms$, tem-se $TR_{mid} = 9$. Caso a fila esteja novamente em 150 ms de atraso e o valor de previsão da rede neural seja de 20 ms, tem-se que $TR_{mid} = 4,5$. Essa lógica foi seguida para os demais valores, até que $y_t = y_t^{max}$, obtendo-se $TR_{mid} = 1$. Quanto maior o tempo de reação (sabe-se que y_t ou x_t estão longe de atingir o limiar mid_{th}), menor o valor de Ω_d. Caso TR_{mid} seja menor que zero, existe um previsão de queda no atraso da fila. Obtidos os valores de TR_{mid}, os valores de Ω_d foram distribuídos de acordo a Tabela 4.3, partindo de $0,1$ - maior valor do tempo de reação ou uma tendência de queda do atraso - até 1.

Caso $x_t = 150ms$ e $y_t = 90ms$, tem-se uma probabilidade de descarte $p_d = 0,5$, com $\Omega_d = 1$. No caso de $x_t = mid_{th}$, a probabilidade de descarte irá aumentar para $p_d = 0,8$, independente do valor y_t. De fato, se o valor atual do atraso da fila atingiu o limiar mid_{th}, isto indica que o atraso está em níveis indesejados. A mesma lógica foi utilizada para encontrar os valores de TR_{up}, com a diferença que os valores atribuídos à Ω_d foram concentrados em 3 regiões, determinadas de modo empírico, e sendo $1,05$

Tabela 4.3: Valores de TR e $weight$

$low_{th} \leq x_t < mid_{th}$	Ω_d	$mid_{th} \leq x_t < up_{th}$	Ω_d
$TR_{mid} \geq 9$ ou $TR_{mid} < 0$	0,1	$TR_{up} \geq 6$ ou $TR_{up} < 0$	0,8
$4,5 \leq TR_{mid} < 9$	0,2	$1,5 \leq TR_{up} < 6$	1
$3 \leq TR_{mid} < 4,5$	0,3	$0 \leq TR_{up} < 1,5$	1,05
$2,25 \leq TR_{mid} < 3$	0,4	–	–
$1,8 \leq TR_{mid} < 2,25$	0,5	–	–
$1,5 \leq TR_{mid} < 1,8$	0,6	–	–
$1,3 \leq TR_{mid} < 1,5$	0,7	–	–
$1,12 \leq TR_{mid} < 1,3$	0,8	–	–
$1 < TR_{mid} < 1,12$	0,9	–	–
$0 \leq TR_{mid} \leq 1$	1	–	–

o maior valor possível. Valores de Ω_d maiores que $1,05$ fazem com que praticamente todos os pacotes que cheguem à fila sejam descartados. Como o atraso está em níveis muito elevados, acima de mid_{th}, pretende-se que o método acelere o descarte de pacotes, de maneira a notificar as fontes TCP sobre o congestionamento.

Este Capítulo apresentou os métodos AQMs propostos e suas parametrizações. Ambos AQMs utilizam o atraso previsto pela rede LSTM para descartar pacotes. No DRAPA, o valor de y_t é diretamente comparado aos limiares low_{th}, mid_{th} e up_{th}, sendo a probabilidade de descarte incrementada linearmente de acordo com as retas p_k e p_z. No DRAPTA, o descarte de pacotes é baseado na tendência do atraso e no tempo de reação restante ao AQM. A probabilidade p_d é aumentada ou diminuída a medida que o tempo de reação varia.

CAPÍTULO 5

AVALIAÇÃO DE DESEMPENHO

Este capítulo apresenta a avaliação de desempenho dos métodos DRAPA e DRAPTA e a comparação com os métodos concorrentes.

5.1 Origem dos Dados

A fim de avaliar o desempenho dos AQMs propostos e dos concorrentes, seis sequências de vídeo Full HD (1920x1080) foram utilizadas nos testes: Big Buck Bunny (BBB), Sunflower (SF), Rush Hour (RH), Pedestrian Area (PA), Riverbed (RB) e Touchdown Pass (TP). Todos são publicamente disponíveis em (SEELING; REISSLEIN, 2012) e (FOUNDATION, 1999) e são frequentemente utilizados por outros autores no estudo de sistemas de video *streaming*, como em (KUA; ARMITAGE, 2017), (KUA; ARMITAGE; BRANCH, 2016), e (GREENGRASS; EVANS; BEGEN, 2009). A Tabela 5.1 resume as características dos vídeos utilizados, como a duração, o número de quadros, gênero, informação temporal (TI, *temporal information*), informação espacial (SI, *spatial information*) e o formato do vídeo. A métrica TI representa a quantidade de mudanças temporais na sequência do vídeo, apresentando maiores valores para cenas de maior movimentação (TELECOMMUNICATION, 2008). Já a métrica SI indica o detalhamento da imagem, com maiores valores para cenas espacialmente mais complexas (TELECOMMUNICATION, 2008). Os vídeos foram selecionados de modo a englobar uma ampla faixa de diversidade entre gênero, complexidade temporal e espacial, aumentando a variedade na avaliação de desempenho.

Produzido por computação gráfica, o vídeo BBB apresenta um padrão de cenas dinâmicas, gerado a partir da troca do foco da câmera entre os personagens da ani-

Tabela 5.1: Características dos vídeos utilizados na avaliação de desempenho

Video	Duração (s)	Nº Quadros	Gênero	TI	SI	Formato	Taxa de Quadros (fps)
BBB	40	1440	Animação	67,02	73,86	YUV420	24
SF	20	500	Natureza	26,19	39,38	YUV420	25
RH	20	500	Cenário	18,86	26,72	YUV420	25
PA	15	375	Cenário	21,08	37,08	YUV420	25
RB	10	250	Natureza	29,66	39,45	YUV420	25
TP	19	570	Esportes	27,13	57,94	YUV422	25

mação e da movimentação de alguns objetos ao redor do cenário. O vídeo é o maior dentre os usados nos testes; com 40 segundos de duração, 1440 quadros e apresentando os maiores valores para TI e SI. No vídeo SF, uma câmera fixa em uma flor captura o movimento de uma abelha coletando pólen. Apesar de exibir cenas estáticas, SF apresenta o quarto maior valor para SI e TI entre os vídeos selecionados. Além disso, a baixa movimentação do vídeo se contrapõe aos demais. RH registra imagens a partir de uma câmera fixa em umas das ruas da cidade de *Munich*. A câmera captura a movimentação dos carros, condutores e de pedestre que circulam pelo local. Do mesmo modo, o vídeo PA captura pessoas passando em frente a uma câmera estática. Apesar da intensidade de movimentação, ambos os vídeos apresentam os menores valores de TI e SI. No vídeo RB, uma câmera fixa captura os movimentos da água bem próximos, atribuindo ao vídeo o terceiro maior valor de SI e o segundo maior de TI. Por fim, o vídeo *Touchdown Pass* apresenta 19 segundos de um jogo de futebol americano, onde uma câmera posicionada no estádio, aproxima o seu foco dos jogadores, proporcionando um padrão dinâmico de mobilidade e variação entre as cenas, com segundo maior valor para SI.

A Figura 5.1 apresenta a imagem de uma cena para cada um dos vídeos utilizados nos testes. Todos os vídeos foram codificados *offline* com o algoritmo H.264/AVC, através da ferramenta *ffmpeg* (FFmpeg Developers, 2019), a qual possibilita configurações como a taxa de quadros por segundo, resolução em pixels do vídeo, tamanho do GOP, entre outras. O ajuste do tamanho do GOP foi realizado de acordo com a taxa de quadros por segundo dos vídeos, de forma que cada segmento DASH possua GOPs inteiros, conforme a recomendação de Irond et al. (IRONDI; WANG; GRECOS,

2015) e Lederer et al. (LEDERER; MULLER; TIMMERER, 2012). Assim, a maioria dos vídeos foram codificados com o GOP ajustado com a sequência $IBBPB$, contendo um total de 5 quadros por GOP, sendo dois quadros B entre os quadros I e P, e um quadro B ao final (SODAGAR, 2011). Para esses vídeos, o GOP pode ser descrito com o par $(5, 2)$. Como o vídeo BBB possui uma taxa de 24 quadros por segundo, o grupo de figuras é referenciado com o par $(6, 2)$, apresentando a sequência $IBBPBB$. Os segmentos foram codificados e disponibilizados no servidor DASH nas seguintes representações de qualidade: 0,349, 0,600, 0,927, 2,114 e 4,464 Mbps, todos com resolução Full HD, seguindo o padrão utilizado pelo o Youtube (KRISHNAPPA; BHAT; ZINK, 2013).

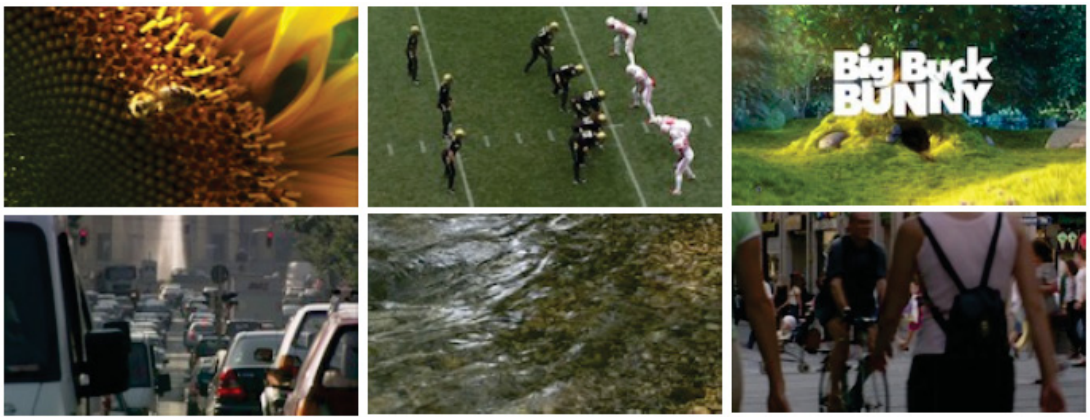

Figura 5.1: Cenas dos vídeos SF, TP, BBB, PA, RB, e RH, respectivamente da esquerda para a direita e de cima para baixo (SEELING; REISSLEIN, 2012).

Para cada qualidade, os vídeos foram fragmentados em segmentos de 1 segundo, através da ferramenta *MP4Box*, disponibilizada pelo GPAC (FEUVRE; CONCOLATO; MOISSINAC, 2007). Segundo o trabalho realizado por *Feuvre et al.* (FEUVRE et al., 2014), segmentos com curta duração, tipicamente de 1 a 2 segundos, são utilizados na transmissão de fluxos ao vivo, uma vez que segmentos maiores aumentam o atraso em cenários de tempo real. Os segmentos foram gerados para transmissões, possuindo tempo de vida limitada no servidor, ou seja, após alguns segundos da sua geração eles expiram, forçando o cliente requisitar sempre o próximo segmento disponível. Para as simulações, a configuração de cliente e servidor DASH reproduzem a transmissão de um evento ao vivo.

5.2 Cenário de Simulação

A simulação foi realizada utilizando o simulador NS-3 (RILEY; HENDERSON, 2010). Sendo baseado em eventos discretos e, assim como a versão anterior NS-2 (*Network Simulator 2*), foi escrito em linguagem de programação C++. O NS-3 é um simulador de código aberto, disponível para pesquisa e uso educacional, além de permitir desenvolvimento de modelos realísticos e conexões com redes reais (RILEY; HENDERSON, 2010). O NS-3 permite a confecção e simulação de diversas topologias de redes, incluindo classes de pilhas de protocolos, como a TCP/IP, e de diversos algoritmos AQMs, como o RED, ARED, Droptail, CoDel e PIE. Porém, o simulador não contempla uma classe em C++ que emule servidor e cliente de tráfego de vídeo adaptativo, como DASH. Por isso, o cenário confeccionado neste trabalho conta com uma topologia híbrida, onde duas máquinas virtuais desempenham a função de servidor e cliente DASH, enviando o tráfego real para o simulador através de uma interface virtual, criada com o aplicativo *TapBridge* do linux e atrelado ao simulador pela classe *TapBridgeHelper*. Segundo o manual do NS-3 (RILEY; HENDERSON, 2010), essa classe é projetada para integrar hosts reais com o simulador, com o objetivo de fazê-las parecerem como nós reais. Os métodos AQMs desenvolvidos nesse trabalho foram implementados através da criação de uma nova classe no NS-3.

A topologia da rede foi implementada como recomendado pela ITU (*International Telecommunication Union*) G.1050 (TELECOMMUNICATION, 2016). Essa recomendação descreve um modelo para avaliar o desempenho de transmissões multimídia através de redes IP. A Figura 5.2 apresenta o cenário usado nas simulações. O enlace DSL (*Digital Subscriber Line*) simula o gargalo na rede de acesso, conectando a rede do cliente ao roteador de borda da ISP, por meio de um enlace de 6 Mbps de taxa de transmissão. Os demais links foram ajustados em 1 Gbps e 100 Mbps como proposto pela recomendação da ITU. Além disso, servidores e clientes TCP foram programados no cenário. Isso possibilitou a geração de tráfego de fundo de modo a avaliar os algoritmos AQMs em vários níveis de congestionamento. Seguindo a recomendação,

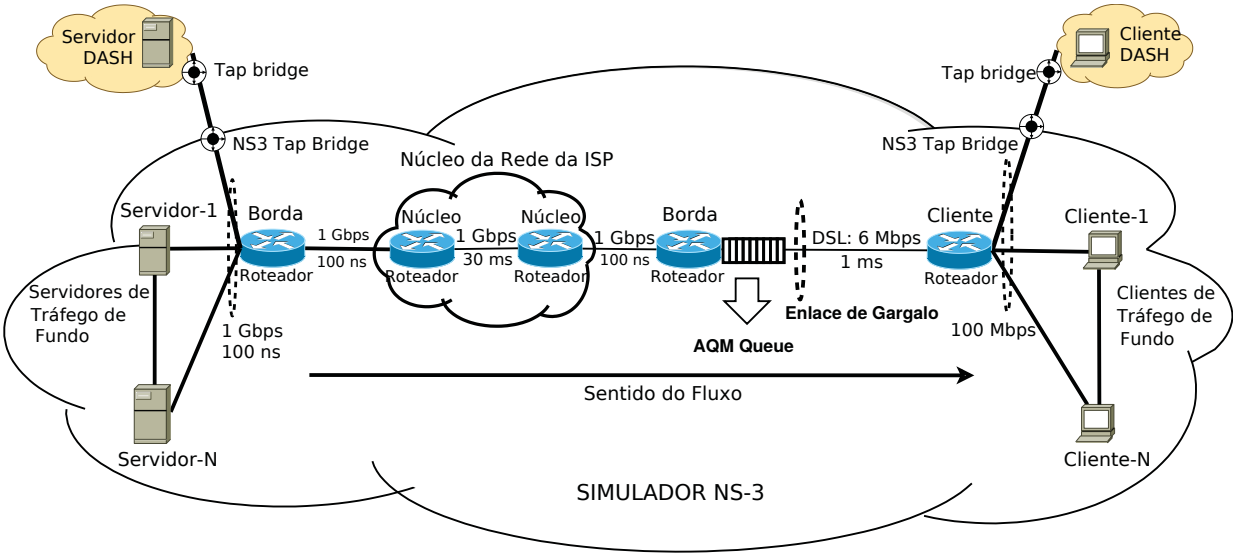

Figura 5.2: Cenário usado nas simulações.

uma rede de acesso conecta os servidores ao núcleo da rede por intermédio de um roteador de borda, com enlaces de 1 Gbps de velocidade. Clientes TCP alocados na rede do cliente recebem o tráfego gerado pelos servidores. Ambos emulam usuários realizando requisições e recebendo tráfego dentro da rede do cliente. A quantidade de clientes e servidores TCP foram adicionados granularmente durante as simulações, a fim de impor níveis variados de congestionamento no cenário. O servidor DASH foi anexado ao mesmo roteador de borda dos servidores TCP, bem como o cliente DASH, incorporado na rede do usuário. Com essa configuração, os fluxos trafegam sempre do núcleo em sentido a rede do cliente. Os AQMs testados e os propostos foram instalados no roteador de borda, como ilustrado na Figura 5.2.

Como os métodos AQMs desenvolvidos fazem uso da rede neural LTSM, a qual foi escrita em linguagem de programação python, semáforos e uma memória compartilhada foram utilizados de modo a permitir que ambos algoritmos (NS-3 e python) fossem executados de forma concorrente. Assim, ao entrar em execução, o NS-3 adquiri o valor atual do atraso da fila, adiciona esse valor a uma memória compartilhada, desbloqueia a LSTM. A rede LSTM, por sua vez, extrai o valor do atraso da memória compartilhada e realiza a predição do atraso futuro, disponibilizando esse valor ao NS-3, também por meio da memória compartilhada. Com o valor previsto, a nova

classe de AQM realiza descartes de pacotes caso o limite do atraso futuro indique congestionamentos. Amostras do valor atual do atraso são feitas a cada δ segundos.

5.3 Tráfego de Fundo

Quando medido em diversas escalas de tempo, o tráfego de vídeo ao vivo exibe uma característica em rajadas, mostrando uma dependência temporal de longa duração (FITZEK; REISSLEIN, 2001). O conceito de LRD em series temporais está relacionado com o decaimento hiperbólico da função de autocorrelação à medida que o deslocamento aumenta (BERAN et al., 1995). Segundo Leland et al. (LELAND et al., 1993), um processo estocástico estacionário $X = (X_t : t = 0, 1, 2, ..)$, com média constante $\mu = E[X_t]$, variância finita $\sigma^2 = E[(X_t - \mu)^2]$, possui função de autocorrelação definida como:

$$r(k) = \frac{E[(X_t - \mu)(X_{t+k} - \mu)]}{E[(X_t - \mu)^2]}, \tag{5.1}$$

para $k = 0, 1, 2, ...$ De acordo com Beran et al. (BERAN et al., 1995), se a função de autocorrelação decai para zero com uma taxa menor que uma função exponencial, de modo que $\sum_{k=1} r(k) = \infty$, então o processo X_t exibe memória de longa duração.

De modo a simular o comportamento de um tráfego de vídeo com característica auto-similar, as fontes de tráfego de fundo foram implementadas utilizando a classe *OnOffApplication* do NS-3. Uma série de dados é dita auto-similar quando suas características são preservadas independentemente da escala em que é observado (PARK; WILLINGER, 2000), onde uma pequena parte da série, quando expandida, se parece com o todo. Portanto, sendo $X^{(m)} = X_k^{(m)} : k = 1, 2, 3, ..$ uma nova série temporal obtida a partir da série de dados original X_t, de tamanho m e particionada em blocos não sobrepostos; X_t é chamado de um processo auto-similar se $X^{(m)}$ tiver a mesma estrutura de correlação de X_t, ou seja, $r^{(m)}(k) = r(k)$ para todo $m = 1, 2, ..(k = 1, 2, 3, ..)$ (LELAND et al., 1993). De acordo com o manual do NS-3 (RILEY; HENDERSON, 2010), a classe *OnOffApplication* é um gerador de tráfego seguindo o padrão On/Off.

O transmissor alterna-se entre os estados *On* e *Off*, com o tempo de permanência nestes estados modelados por variáveis randômicas. Durante o período Off, não existe geração de tráfego. No estado On, um tráfego com taxa constante (CBR, *Constant Bit Rate*) é produzido, com o tamanho e intervalo entre os pacotes configuráveis, permitindo o ajuste da taxa de transmissão desejada. Nas simulações, os períodos On e Off das fontes de tráfego de fundo foram modeladas com as distribuições de probabilidade Pareto e Exponencial, respectivamente. Tais características, permitem a simulação de tráfego de vídeo com características auto-similares (AMMAR; BEGIN; GUERIN-LASSOUS, 2011; WILLINGER et al., 1997), com o parâmetro de *Hurst* dado por:

$$H = \frac{3 - \alpha}{2}, \tag{5.2}$$

onde α é o parâmetro de forma da distribuição de Pareto (WILLINGER et al., 1997). O parâmetro de *Hurst* expressa a intensidade da auto similaridade de uma série temporal (PARK; WILLINGER, 2000). Para séries temporais com características LRD, *H* é tipicamente ajustado entre 0.5 e 1.0. A medida que *H* tende para 1, o grau de dependência de longa duração aumenta (PARK; WILLINGER, 2000). Uma análise do tráfego de vídeo foi realizada no estudo de Fitzek e Reisslein (FITZEK; REISSLEIN, 2001). Nele, para diversos níveis de agregação dos quadros dos vídeos, o parâmetro de *Hurst* manteve-se maior ou igual a $0,72$.

Para cada fonte de tráfego de fundo, o período *Off* foi implementado com o inverso da taxa média de quadros por segundo dos vídeos, sendo:

$$T_{Off} = \frac{1}{\bar{fps}}. \tag{5.3}$$

Já o período *On* foi modelado utilizando o valor médio da distribuição de Pareto, dado por:

$$T_{On} = \frac{\alpha \cdot x_m}{\alpha - 1}, \tag{5.4}$$

A fim de obter o valor da variável *scale* (x_m), o estado *On* foi configurado com o tempo

médio de duração de um quadro dos vídeos em estudo, codificados com taxa de 1 Mbps. O parâmetro α foi determinado de modo a produzir tráfego com características auto-similares. Para tal, o valor do parâmetro *H* escolhido foi de $0,85$. Cada fonte foi implementada de modo a fornecer 500 Kbps de tráfego, sendo:

$$Taxa_{fonte} = CBR_{rate} \cdot \frac{T_{On}}{T_{On} + T_{Off}}, \tag{5.5}$$

sendo CBR_{rate} a taxa constante de envio de bits durante o período *On*, determinada em função de T_{On}, T_{Off} e da taxa total desejada ($Taxa_{fonte}$).

5.4 Métricas de Análise de Desempenho

A avaliação de desempenho dos métodos propostos foi realizada estimando a qualidade dos vídeos DASH recebidos através das métricas PSNR(*Peak Signal Noise Ratio*), SSIM e da quantidade e duração das interrupções percebidas nos vídeos (JULURI; TAMARAPALLI; MEDHI, 2016). PSNR e SSIM são métricas objetivas com referência completa (FR, *Full Reference*), ou seja, são baseadas na comparação quadro a quadro entre o vídeo original (ou de referência) e o vídeo recebido. Ambas são adequadas para a avaliação da qualidade da imagem em sistemas tradicionais de televisão (JULURI; TAMARAPALLI; MEDHI, 2016).

O uso do termo qualidade de experiência (QoE, *Quaity of Experience*) foi evitado nesse trabalho, uma vez que a QoE é definida como a aceptabilidade geral de uma aplicação ou serviço, como percebida subjetivamente pelo usuário final (TELECOMMUNICATION, 2004; STANKIEWICZ; CHOLDA; JAJSZCZYK, 2011). A quantificação da QoE é tipicamente baseada na métrica MOS (*Mean Opnion Score*), a qual é avaliada de forma subjetiva por usuários, atribuindo a opinião deles ao desempenho de serviços como voz e vídeo (STANKIEWICZ; CHOLDA; JAJSZCZYK, 2011). Medir a QoE demanda um cenário controlado com uma diversidade de usuários dispostos a assistir/ouvir os conteúdos transmitidos. Adicionalmente, a avaliação pode ser afetada

pelo ambiente ao qual os usuários se encontram, por fatores psicológicos e sociais, além da expectativa dos usuários e a experiência com serviços similares (STANKIEWICZ; CHOLDA; JAJSZCZYK, 2011). Com isso, medir a QoE de forma subjetiva demanda tempo, é custoso e tem que ser realizada por usuários humanos e validada estatisticamente, não permitindo a realização automática da qualidade por meio de softwares (ZINNER et al., 2010). As comparações de desempenho entre os métodos AQMs não necessitam o conhecimento exato do MOS como percebido pelo usuário, e sim uma medida que permita inferir se houve melhoria de qualidade em comparação com outros métodos. Desta forma, as métricas PSNR e SSIM foram usadas na avaliação de desempenho do métodos AQMs, as quais que serão descritas a seguir.

5.4.1 *Peak Signal Noise Ratio*

O PSNR é uma métrica que afere a similaridade entre duas imagens, calculando o erro médio quadrático (MSE, *Mean Square Error*) de cada pixel entre a imagem original e recebida. O MSE é calculado quadro a quadro, dado por:

$$\text{MSE} = \frac{1}{rc} \sum_{i=1}^{r} \sum_{j=1}^{c} \left[X_o(i,j) - X_r(i,j) \right]^2 , \tag{5.6}$$

onde r e c representam o número de linhas e colunas da imagem, respectivamente, e $X_o(i,j)$ e $X_r(i,j)$, representam o pixel (i,j) dos quadros original e recebido. O PSNR pode ser obtido como:

$$\text{PSNR} = 20 \log_{10} \left(\frac{\text{MAX}_I}{\sqrt{\text{MSE}}} \right) , \tag{5.7}$$

onde MAX_I representa o valor máximo de intensidade do pixel. Para os vídeos utilizados nesse trabalho, $\text{MAX}_I = 255$.

O PSNR é medido em decibéis (dB). Assim, imagens com maior similaridade resultam em um maior PSNR (SANTOS; RIBEIRO; PEDROSO, 2014). Para valores maiores que 37 dB, a qualidade percebida do vídeo é excelente e para valores menores que 15 dB, a qualidade é muito ruim (ZINNER et al., 2010). A principal desvanta-

gem do PSNR deve-se a métrica não considerar como a percepção humana trabalha, não capturando, em alguns casos, pertubações visíveis ao olho humano (SERRAL-GRACIÀ et al., 2010).

5.4.2 *Structural Similarity*

De modo a suprir a insuficiência da métrica PSNR, o SSIM combina os efeitos da luminância, do contraste e da similaridade estrutural da imagem para comparar as correlações existentes entre o vídeo original e o recebido (TELECOMMUNICATION, 2016). Desenvolvida por Wang and Bovik (WANG et al., 2004), o SSIM é uma métrica objetiva de referência completa que avalia a qualidade da imagem levando em consideração características do sistema visual humano (HVS, *Human Visual System*). No SSIM, leva-se em consideração que os *pixels* da imagem possuem forte dependência entre si e que essa dependência possui informações importantes sobre a estrutura dos objetos na imagem (ALENCAR, 2012). Segundo Wang et al. (WANG; LU; BOVIK, 2004), em um conjunto de imagens contendo distorções propositais, apesar delas renderem valores similares de MSE, as distorções são drasticamente percebidas pelo sistema visual humano, como borrões e distorções no contraste. Desta forma, o algoritmo do SSIM utiliza as informações estruturais dos objetos que compõe a cena, levando à uma avaliação da qualidade da imagem que separa a luminância, o contraste e as distorções estruturais. Sendo a e b dois sinais discretos representando fragmentos extraídos da mesma localização espacial de duas imagens sendo comparadas, e μ_a, μ_b, σ_a^2, σ_b^2 e σ_{ab} a média de a, a média de b, a variância de a, a variância de b e a covariância de a e b, respectivamente. A média e o desvio padrão ($\sqrt{\sigma^2}$) são considerados como os sinais de luminância e o contraste da imagem e, a covariância entre a e b, a quantidade de informação que é modificada do sinal original e o sinal

sendo comparado:

$$l_{a,b} = \frac{2\mu_a\mu_b}{\mu_a^2 + \mu_b^2}, \tag{5.8}$$

$$c_{a,b} = \frac{2\sigma_a\sigma_b}{\sigma_a^2 + \sigma_b^2}, \tag{5.9}$$

$$s_{a,b} = \frac{\sigma_{ab}}{\sigma_a\sigma_b}, \tag{5.10}$$

Nota-se que os termos são independentes, onde mudanças no contraste ($l_{a,b}$) e na luminância ($c_{a,b}$) da imagem não geram impactos na modificação das estruturas ($s_{a,b}$). De acordo Wang et al. (WANG; LU; BOVIK, 2004), a métrica SSIM é determinada através da combinação entre esses três componentes e é dada por:

$$SSIM(a,b) = \frac{(2\mu_a\mu_b + C_1)(2\sigma_a\sigma_b + C_2)}{(\mu_a^2\mu_b^2 + C_1)(\sigma_a^2 + \sigma_b^2 + C_2)}, \tag{5.11}$$

sendo C_1 e C_2 duas constantes adicionadas de modo que, se $(\mu_a^2 + \mu_b^2)$ ou $(\sigma_a^2 + \sigma_b^2)$ forem próximos de zero, instabilidades na aquisição do SSIM não sejam observadas (WANG; LU; BOVIK, 2004). Sendo, $C_1 = (K_1L)^2$ e $C_2 = (K_2L)^2$, onde $L = 255$ (para imagens de 8 bits/pixel em escalas de tons cinzas) e, K_1 e K_2 valores constantes muito pequenos. Nos experimentos realizados por Wang et al. (WANG; LU; BOVIK, 2004), $K_1 = 0,01$ e $K_2 = 0,03$. O valor estimado do SSIM é dado entre 0 e 1, sendo que o valor 1 indica que a e b são muito similares enquanto 0, que a e b são diferentes (WANG et al., 2004).

5.4.3 Quantidade e Duração das Interrupções nos Vídeos

Além das métricas PSNR e SSIM, também foram avaliadas a quantidade e a duração das interrupções nos vídeos recebidos. Uma interrupção ocorre quando a reprodução do vídeo é temporariamente parada devido à falta de segmentos no *playout buffer* de reprodução do cliente, como consequência de congestionamentos na rede (JULURI; TAMARAPALLI; MEDHI, 2016). As interrupções não são percebidas pelo usuá-

rio quando a taxa de transmissão dos segmentos for maior que a taxa em que o vídeo é reproduzido. Caso a taxa de transmissão diminua em comparação da taxa de reprodução do vídeo, o cliente DASH automaticamente reduz a qualidade do segmento, a fim de manter uma reprodução contínua. Porém, mesmo segmentos da mais baixa qualidade podem demorar para serem transferidos, caso o enlace esteja congestionado, interrompendo a reprodução no cliente. De acordo com Juluri, Tamarapalli e Medhi (JULURI; TAMARAPALLI; MEDHI, 2016), eventos de interrupção na reprodução do vídeo resultam em uma péssima qualidade do vídeo assistido. Além da quantidade de interrupções, o tempo de duração da interrupção também é um fator importante a ser medido.

5.5 Dinâmica dos Experimentos

Várias simulações foram realizadas com o cenário implementado. Para cada simulação foram variados (i) o método AQM utilizado no roteador de borda conectado ao enlace DSL; (ii) o número de fontes de tráfego de fundo e (iii) o vídeo transmitido. A quantidade de fluxos de tráfego de fundo foi variada de 0, 6, 10, 12, 13, 14, 15, 16, 17 e 18. Teoricamente, com 12 fontes de tráfego de fundo ativas, a capacidade máxima do menor enlace - 6 Mbps do enlace DSL - é atingida. A justificativa do uso de mais fontes além da capacidade teórica do enlace, deve-se à adaptação da taxa transmissão das fontes TCP, caso congestionamentos sejam percebidos na rede. Assim, utilizando um maior número de fontes, foi possível investigar o desempenho dos AQMs para níveis elevados de congestionamentos.

Para cada simulação, antes que os vídeos fossem transmitidos, o cenário de simulação era acionado, iniciando a transmissão dos fluxos de fundo entre servidor e cliente *OnOff*. De forma a respeitar o tempo de incremento e estabilização das janelas de congestionamento do protocolo TCP, o tempo de 20 segundos foi aguardado antes de dar inicio à transmissão dos vídeos em tempo real.

Os AQMs concorrentes foram configurados de acordo com a Tabela 5.2. Os va-

Tabela 5.2: Parâmetros dos AQMs

AQM	Parâmetros
RED	$min_{th} = 250\,pkt$, $max_{th} = 500\,pkt$, $tam.max. = 2M\,pkt$
ARED	$Target_{del} = 5\,ms$, $bw = 6\,Mbps$, $tam.max. = 2M\,pkt$
CoDel	$Target_{del} = 5\,ms$, $Interval = 100\,ms$, $tam.max. = 2M\,pkt$
PIE	$dq\_th = 7\,pkt$, $ref\_del = 20\,ms$, $tam.max. = 2M\,pkt$
Droptail	$tam.max. = 2M\,pkt$

lores de min_{th} e max_{th} para o RED foram ajustados de forma a permitir o valor 1 segundo de atraso na fila, considerando o tamanho médio dos pacotes de 1500 bytes. Para o ARED, min_{th} e max_{th} são ajustados automaticamente de acordo com a velocidade do enlace ($bw = 6\,Mbps$) e o valor alvo atraso ($Target_{del} = 5\,ms$). Tanto o CoDel como o PIE utilizaram suas configurações padrões, ou seja, $Target_{del} = 5\,ms$ e $Interval = 100\,ms$, para o primeiro e, limiar da fila (dq_th) de 7 pacotes e atraso de referência (ref_del) de 20 ms para o segundo. Para todos os AQMs, o tamanho máximo configurado para a fila foi de 2 M pacotes. Em exceto para o Droptail, o demais AQMs não permitem que a fila atinja a capacidade máxima de ocupação, devido à dinâmica de cada método. No caso do Droptail, essa quantidade de pacotes foi ajustada como forma a permitir a ocorrência do fenômeno do *bufferbloat*.

5.6 Resultados

Esta seção apresenta os resultados obtidos pelos métodos DRAPA e DRAPTA em comparação com os AQMs concorrentes. Os resultados indicam que a aplicação DRAPA e DRAPTA apresenta um menor nível na degradação da qualidade da imagem recebida, se comparado com o RED, ARED, CoDel PIE e Droptail, principalmente em casos onde o congestionamento do enlace atinge níveis elevados.

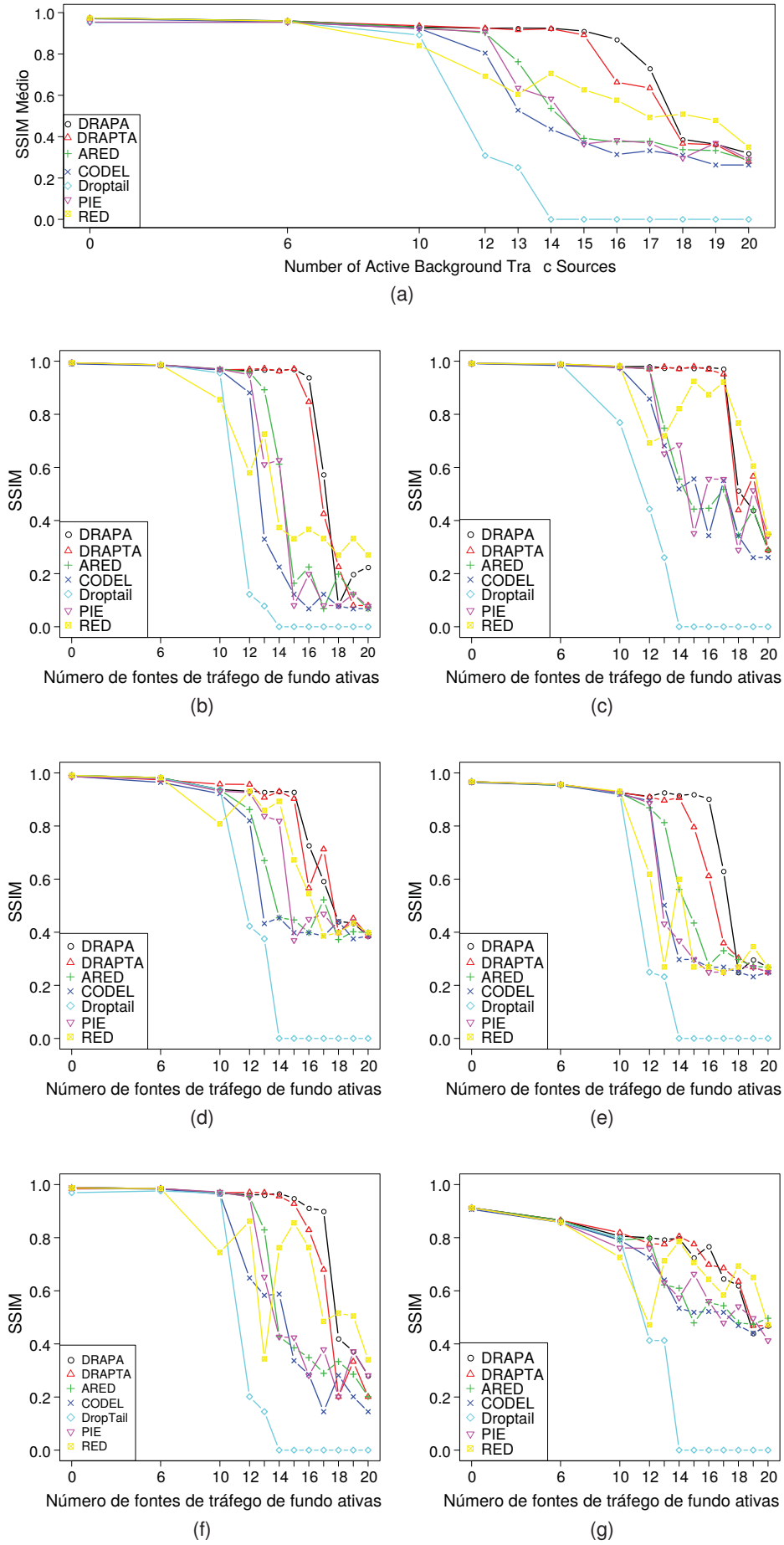

Figura 5.3: Resultado do SSIM para (a) média de todos os vídeo, (b) BBB, (c) RH, (d) SF, (e) TP, (f) PA e (g) RB.

Na Figura 5.3 (a), o valor médio do SSIM para cada AQM é avaliado, de modo a permitir um melhor entendimento na melhora da qualidade do vídeo atingida pelos métodos DRAPA e DRAPTA. Neste caso, foi somado o valor do SSIM de cada vídeo em relação à quantidade de fontes de tráfego ativas, dividindo esse valor pela quantidade de vídeos usados. Nota-se que o SSIM dos métodos propostos estão praticamente sobrepostos nas figuras. Como é possível perceber, ambos os métodos propostos superam os demais AQMs, principalmente quando o enlace de acesso ao cliente, está muito congestionado. Quando 13, 14 e 15 fontes de tráfego de fundo estão ativas, o DRAPA e o DRAPTA ainda apresentam uma boa qualidade enquanto os demais AQMs já atingiram nível péssimo de qualidade. Com 16 fontes, o DRAPA ainda mantém a qualidade dos vídeos em níveis elevados, tendo o DRAPTA uma leve queda. Acima de 17 fontes, ambos os AQMs começam a reduzir o valor do SSIM médio e para 18, 19 e 20, eles se aproximam dos níveis atingidos pelos AQMs concorrentes. O Droptail apresenta o pior desempenho dentre os AQMs. Quando até 13 fontes de tráfego estão ativas, o ARED obtém o terceiro melhor desempenho médio. Porém, com 14 fontes, o congestionamento intenso no enlace produz uma melhor resposta do RED, fazendo com que ele atinja o terceiro melhor desempenho. Acima de 17 fontes, a qualidade do vídeo reproduzido é péssima para todos os métodos. Como pode-se observar na Tabela 4.2, valores de SSIM abaixo de 0,5 representam que a qualidade do vídeo é péssima.

As Figuras 5.3 (b), (c), (d), (e), (f) e (g) apresentam os resultados do SSIM em função do número de fontes de tráfego de fundo ativas, para os vídeos BBB, RH, SF, TP, TA e RB, respectivamente. Tanto o DRAPA quanto o DRAPTA, obtém resultados semelhantes e superiores aos demais AQMs. A previsão do atraso realizada pela rede LSTM permite que os métodos reduzam a qualidade dos segmentos, antecipando o congestionamento. A partir de 17 fontes de tráfego de fundo ativas, tanto o DRAPA quanto o DRAPTA começam apresentar queda no seu desempenho. Isto deve-se ao congestionamento intenso ao qual o enlace está submetido. Mesmo assim, os métodos propostos possuem rendimento melhores que os AQMs concorrentes. Com 18

fontes nenhum dos métodos consegue entregar o vídeo com qualidade que permita que seja assistido. Segmentos menores diminuem a quantidade total de bytes a serem transmitidos aos clientes, o que paradoxalmente, reduz a qualidade do video, porém evita congelamentos da imagem, mantendo o SSIM em padrões elevados. Para a maioria dos casos, o ARED obtém o terceiro melhor desempenho. O Droptail apresenta o pior desempenho entre os AQMs testados. Devido ao tamanho excessivo da fila, com o efeito do *bufferbloat*, os pacotes acabam sofrendo atrasos exagerados, fazendo com que seu tempo limite para reprodução seja expirado e que o cliente DASH perca a sincronia com o servidor. Já no caso do RED, sua política de descarte e o padrão em rajada do tráfego fazem com que o seu desempenho seja reduzido em alguns casos. Como pode-se observar, o RED apresenta pontos em que seu desempenho fica abaixo dos demais AQMs. De fato, o cliente DASH acaba requisitando segmentos de maior qualidade, como consequência do cálculo sobrestimado da largura de banda do canal. Isso deve-se à períodos de rajadas com menor intensidade do tráfego de fundo. Em contrapartida, quando o enlace está com altos níveis de congestionamento (a partir de 14 fontes ativas), nota-se que seu desempenho melhora, atingindo níveis de SSIM elevados. Neste caso, o cliente DASH não consegue requisitar segmentos de alta qualidade em função do aumento substancial na ocupação do enlace. Isto reduz a quantidade de mudanças abruptas nas qualidades do segmentos, permitindo que o cliente requisite apenas os segmentos de menor qualidade. Porém, para a maioria dos casos, o aumento substancial na ocupação do enlace como forma do cliente DASH reduzir o número de transições abruptas, não é suficiente para que o RED alcance o desempenho dos métodos propostos. Ao longo da transmissão o cliente DASH acaba perdendo segmentos de vídeo.

O PSNR também foi utilizado para medir o desempenho dos AQMs em relação a qualidade da imagem entregue. O PSNR é a métrica mais simples e uma das mais utilizadas para medir a qualidade da imagem de forma objetiva (WANG et al., 2004). Para melhor visualização e entendimento desses resultados, a Figura 5.4 (a) mostra a média do PSNR normalizado de cada AQM. Novamente, o DRAPA e o DRAPTA

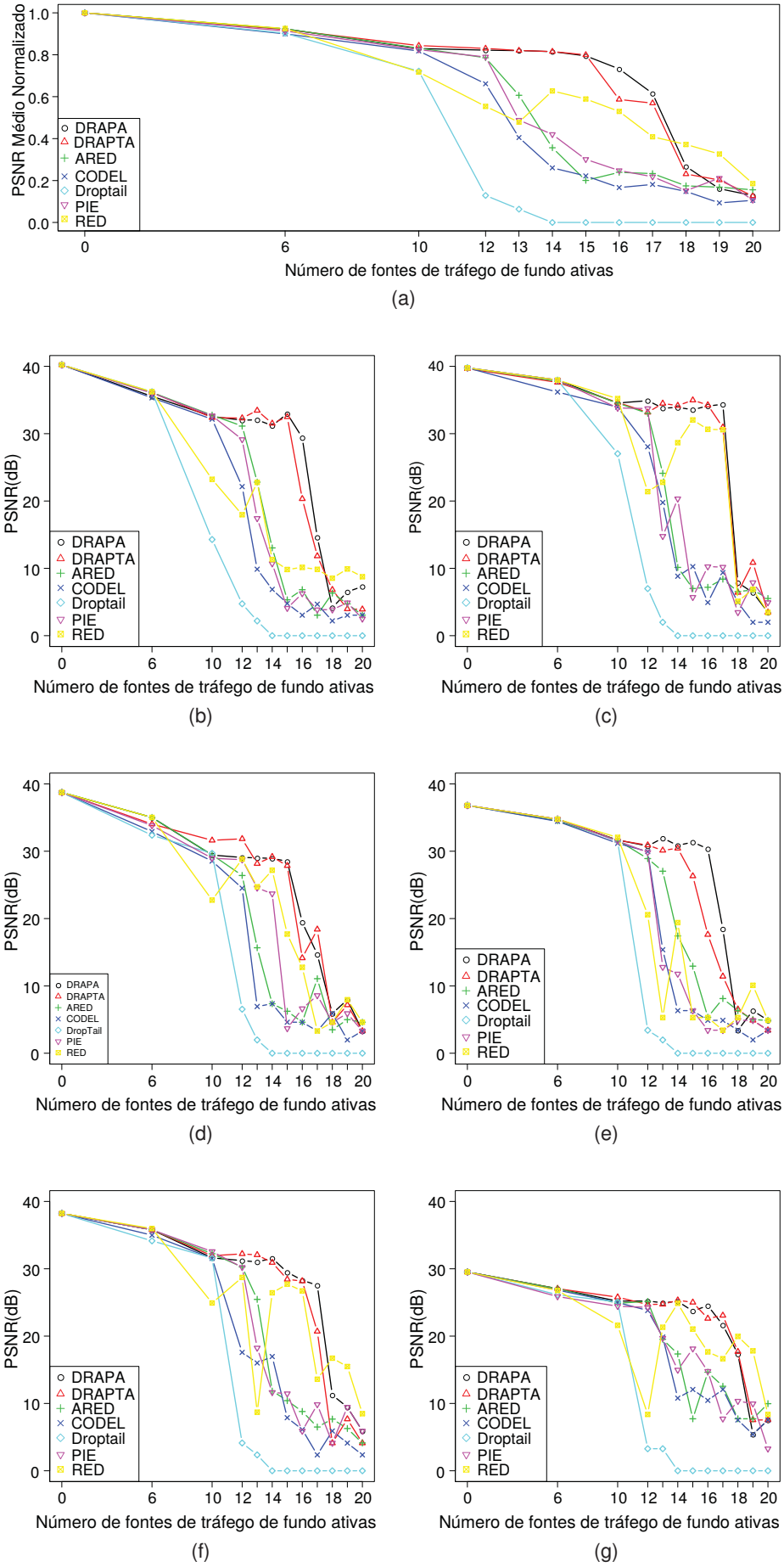

Figura 5.4: Resultados do PSNR para (a) média normalizada de todos os vídeos, vídeo (b) BBB, (c) RH, (d) SF, (e) TP, (f) PA e (g) RB.

atingem resultados próximos, superando todos os demais AQMs testados. Quando 12, 13, 14 e 15 fontes de tráfego estão ativas, os métodos propostos obtém ganhos sobre os AQMs melhores classificados. Assim como para o SSIM médio, para o PSNR médio e normalizado, a partir de 16 fontes ativas, o DRAPA e o DRAPTA começam a reduzir os seus desempenhos devido a intensidade do tráfego de fundo. Como mostram as Figuras 5.4 (b), (c), (d), (e), (f), e (g), o desempenho do DRAPA e do DRAPTA superam os demais AQMs. Ambos possuem comportamento similar para a gama de vídeos testados. Dentre os AQMs concorrentes, o ARED destaca-se sobre os demais. O RED obtém melhor desempenho quando o enlace está muito congestionado, em específico nas Figuras 5.4 (d), (f) e (g), ele chega a níveis de PSNR muito próximos aos dos métodos proposto. Assim como para o SSIM, o Droptail apresenta o pior desempenho, devido ao fenômeno do *Bufferbloat*. O teste com o PSNR confirma a análise realizada com o SSIM.

A Tabela 5.3 apresenta a quantidade (N) e a duração (D) das interrupções para cada um dos vídeos. Como os vídeos foram codificados com segmentos de 1 segundo, a duração das interrupções determinam a quantidade de tempo que o vídeo permaneceu paralisado. Os vídeos BBB, PA, RB, RH, SF e TP possuem 40, 15, 10, 20, 20 e 19 segmentos, respectivamente. Como pode-se perceber, os métodos propostos superam os demais AQMs na quantidade e duração das interrupções. Para o vídeo BBB, o DRAPA interrompe o vídeo por apenas 1 segundo, quando o enlace está com 14 fontes de tráfego ativas. Em comparação, o ARED e o PIE interrompem o vídeo duas vezes por 19 segundos, enquanto o CoDel, Droptail e RED, interrompem por mais de 20 segundos. Já o DRAPTA não interrompe o vídeo, para essa mesma quantidade de fontes ativas. Com 16 fontes ou mais, todos os AQMs interrompem o vídeo BBB. Um comportamento similar pode ser observado para o vídeo SF.

O Droptail interrompe a reprodução do vídeo BBB por 4 vezes com 10 fontes ativas. Para todos os vídeos transmitidos, o Droptail apresenta o pior desempenho, falhando por completo em reproduzir o vídeo quando 12 fontes de tráfego de fundo estão ativas.

Tabela 5.3: Quantidade e Duração das interrupções para os vídeos.

	Nº Fontes	10		12		13		14		15		16		17		18	
		N	D (s)	N	D (s)	N	D (s)	N	D (s)	N	D (s)	N	D (s)	N	D (s)	N	D (s)
BBB	DRAPA	0	0	0	0	0	0	1	1	0	0	1	5	1	23	1	36
	DRAPTA	0	0	0	0	0	0	0	0	0	0	2	14	1	31	1	32
	ARED	0	0	1	1	2	9	2	19	1	34	1	32	1	37	1	33
	CoDel	0	0	2	9	1	28	1	32	1	35	1	37	1	35	1	38
	Droptail	**4**	**15**	1	35	1	38	1	40	1	40	1	40	1	40	1	40
	RED	**2**	**12**	**1**	**20**	**1**	**13**	1	26	1	28	1	27	1	36	1	30
	PIE	0	0	1	3	1	19	2	19	1	36	1	33	1	28	1	36
	Nº Fontes	10		12		13		14		15		16		17		18	
		N	D (s)	N	D (s)	N	D (s)	N	D (s)	N	D (s)	N	D (s)	N	D (s)	N	D (s)
PA	DRAPA	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	9
	DRAPTA	0	0	0	0	0	0	0	0	0	0	1	6	1	5	1	13
	ARED	0	0	0	0	1	2	1	9	1	10	1	11	1	12	1	11
	CoDel	0	0	2	6	1	7	**1**	**7**	1	11	1	12	1	14	1	12
	Droptail	0	0	1	13	1	14	1	15	1	15	1	15	1	15	1	15
	RED	**1**	**4**	**1**	**2**	**1**	**11**	**1**	**7**	1	2	1	3	1	10	1	2
	PIE	0	0	0	0	2	6	1	9	1	9	1	12	1	8	1	13
	Nº Fontes	10		12		13		14		15		16		17		18	
		N	D (s)	N	D (s)	N	D (s)	N	D (s)	N	D (s)	N	D (s)	N	D (s)	N	D (s)
RB	DRAPA	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	3
	DRAPTA	0	0	0	0	0	0	0	0	0	0	1	1	0	0	1	3
	ARED	0	0	0	0	1	2	**1**	**3**	1	7	1	4	1	5	1	7
	CoDel	0	0	0	0	1	2	**1**	**6**	1	5	1	6	1	5	1	7
	Droptail	0	0	1	9	1	9	1	10	1	10	1	10	1	10	1	10
	RED	**1**	**2**	**1**	**7**	**1**	**2**	0	0	1	3	1	3	1	7	1	2
	PIE	0	0	0	0	1	2	1	4	1	4	1	4	1	3	1	4
	Nº Fontes	10		12		13		14		15		16		17		18	
		N	D (s)	N	D (s)	N	D (s)	N	D (s)	N	D (s)	N	D (s)	N	D (s)	N	D (s)
RH	DRAPA	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	15
	DRAPTA	0	0	0	0	0	0	0	0	0	0	1	11	0	0	1	16
	ARED	0	0	0	0	1	5	1	14	1	16	1	16	1	15	1	17
	CoDel	0	0	1	3	1	8	**1**	**15**	1	16	1	17	1	14	1	17
	Droptail	**1**	**5**	**1**	**15**	1	19	1	20	1	20	1	20	1	20	1	20
	RED	0	0	**1**	**8**	**1**	**7**	**1**	**4**	1	2	1	3	1	2	1	15
	PIE	0	0	0	0	2	10	1	8	1	7	1	14	1	14	1	18
	Nº Fontes	10		12		13		14		15		16		17		18	
		N	D (s)	N	D (s)	N	D (s)	N	D (s)	N	D (s)	N	D (s)	N	D (s)	N	D (s)
SF	DRAPA	0	0	0	0	0	0	0	0	0	0	1	7	1	10	1	16
	DRAPTA	0	0	0	0	0	0	0	0	0	0	1	11	2	6	1	17
	ARED	0	0	0	0	2	7	1	15	1	16	1	17	1	12	1	18
	CoDel	0	0	1	2	1	16	1	15	1	17	1	17	1	18	1	16
	Droptail	0	0	1	16	1	20	1	20	1	20	1	20	1	20	1	20
	RED	**2**	**6**	0	0	1	3	1	2	1	8	1	12	1	18	1	17
	PIE	0	0	0	0	1	2	1	3	1	18	1	16	1	14	1	17
	Nº Fontess	10		12		13		14		15		16		17		18	
		N	D (s)	N	D (s)	N	D (s)	N	D (s)	N	D (s)	N	D (s)	N	D (s)	N	D (s)
TP	DRAPA	0	0	0	0	0	0	0	0	0	0	0	0	2	7	1	17
	DRAPTA	0	0	0	0	0	0	0	0	1	2	1	7	1	12	1	15
	ARED	0	0	0	0	1	2	1	9	1	12	1	17	1	14	1	15
	CoDel	0	0	0	0	1	10	1	15	1	15	1	16	1	16	1	17
	Droptail	0	0	1	17	1	18	1	19	1	19	1	19	1	19	1	19
	RED	0	0	**1**	**7**	**1**	**16**	1	7	1	16	1	16	1	17	1	16
	PIE	0	0	0	0	1	11	1	12	1	15	1	17	1	17	1	16

Além do Droptail, o RED é o único AQM que paralisa o vídeo com 10 fontes de tráfego. Esse comportamento pode ser observado para os vídeo BBB, PA, RB e SF. Neste

caso, a instabilidade na ocupação da fila em razão do tráfego em rajadas, faz com que o cliente DASH troque abruptamente de segmentos de menor qualidade para os de maior qualidade, como resposta à baixa ocupação momentânea da fila. Na Tabela 5.3, os valores de N e D marcados em negrito informam que o cliente DASH realizou uma transição abrupta para os segmentos de maior qualidade, provocando atrasos na entrega do segmento atual e acarretando na perda de outros. Para a grande maioria dos vídeos, quando 18 fontes de tráfego estão ativas, o RED é o AQM que menos paralisa o vídeo. Como mencionado anteriormente, o alto nível de ocupação da fila e o seu tamanho, contribuem para que o cliente não consiga requisitar segmentos de alta qualidade, evitando que o vídeo seja paralisado. Porém, com esta quantidade de fontes de tráfego de fundo o vídeo é pouco reproduzido, passando a maior parte do tempo em interrupção, tanto para o DRAPA e DRAPTA, quanto para os AQMs concorrentes.

A Figura 5.5 exibe as transições de qualidade dos segmentos para o vídeo BBB quando 10 fontes de tráfego de fundo estão ativas. Como pode-se observar, tanto o RED quanto o Droptail permitem que o cliente DASH requisite os segmentos de maior qualidade, mesmo com a rede congestionada. O cliente DASH acaba perdendo os próximos segmentos, pelo motivo de terem expirados no servidor. Como consequência, o vídeo é paralisado durante sua reprodução, impactando negativamente na qualidade do vídeo assistido. Os métodos propostos foram capazes de prever o atraso da fila. Com isso o cliente DASH foi induzido a reduzir o qualidade dos segmentos antecipadamente, resultando em transições mais suaves e melhorando a qualidade do vídeo assistido.

A Figura 5.6 apresenta as transições de qualidade dos segmentos para o vídeo BBB, porém com 6 fontes de tráfego ativas. Com um menor número fontes injetando tráfego na rede, percebe-se um comportamento mais estável na transições de qualidade. Isso deve-se ao tráfego agregado apresentar rajadas de menor intensidade, mantendo uma ocupação média do enlace. Outro fator que colabora para esse com-

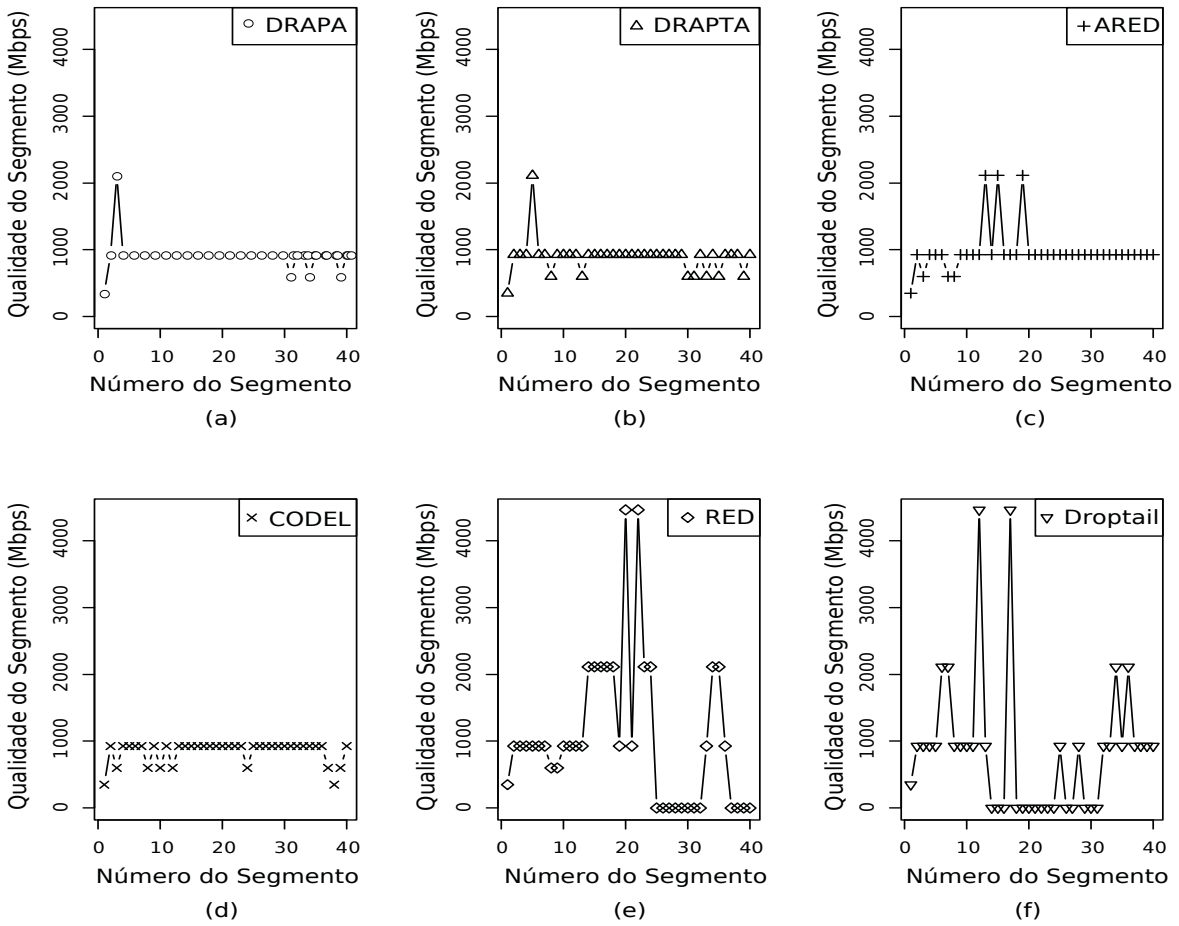

Figura 5.5: Transições entre as qualidades dos segmentos para (a) DRAPA, (b) DRAPTA, (c) ARED, (d) CoDel, (e) RED e (f) Droptail, com 10 fontes de tráfego de fundo ativas para o vídeo BBB.

portamento é baixa quantidade de descartes de pacotes, possibilitando que as fontes mantenham o envio constante de pacotes.

A Figura 5.7 apresenta o atraso da fila para os AQMs DRAPA, DRAPTA, ARED, CoDel, RED e Droptial quando 10 fontes de tráfego de fundo estão ativas, para o vídeo BBB. De acordo com a Figura, os métodos propostos mantém o atraso da fila em torno de um valor médio de 60 ms, enquanto o ARED e o CoDel em 11 e 14 ms. Porém, tanto o ARED quanto o CoDel sustentam esses baixos valores através de uma maior quantidade de descartes de pacotes. Desta forma, quando ocorre o aumento do número de fontes de tráfego, esse comportamento se repete, aumentando o atraso e prejudicando a transmissão dos segmentos. Em específico, a Figura 5.7 (e)

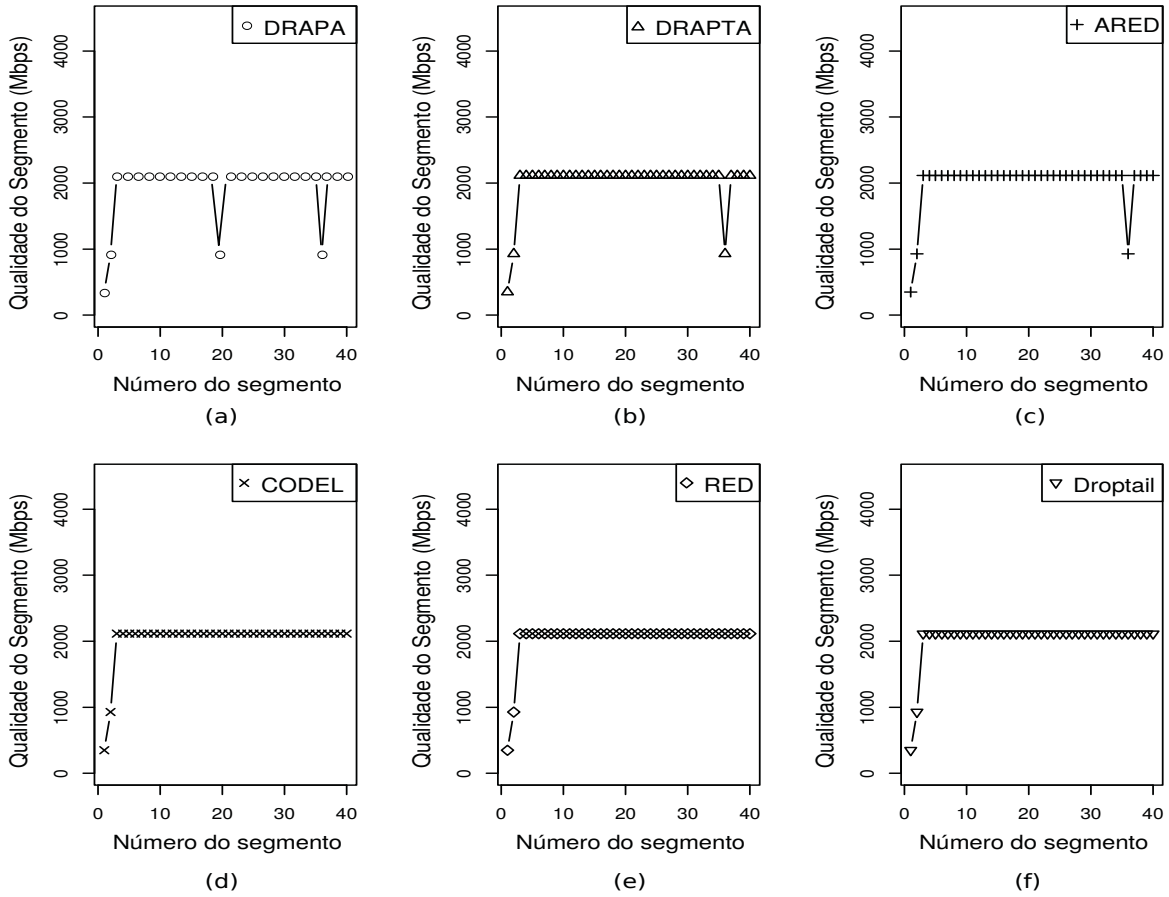

Figura 5.6: Transições entre as qualidades dos segmentos para (a) DRAPA, (b) DRAPTA, (c) ARED, (d) CoDel, (e) RED e (f) Droptail, com 6 fontes de tráfego de fundo ativas para o vídeo BBB.

mostra os valores de atraso para o RED. Nela, é possível perceber que o atraso possui uma tendência de subida a partir de 20 segundos de simulação. Essa tendência é impulsionada pela transmissão do vídeo no cenário. Como o atraso começa com níveis baixos (algumas dezenas de milissegundos), possibilita que o cliente DASH requisite segmentos de maior qualidade. Porém, durante a transmissão dos demais segmentos, o atraso na fila aumenta gradativamente, fazendo com que o segmento demore a chegar no cliente, gerando interrupções no vídeo. O mesmo comportamento é observado na Figura 5.7 (f). Porém nela, o atraso da fila chega a atingir 1 segundo, devido a configuração do Droptail.

A fim de apresentar o desempenho que a rede LSTM possui no DRAPA e no

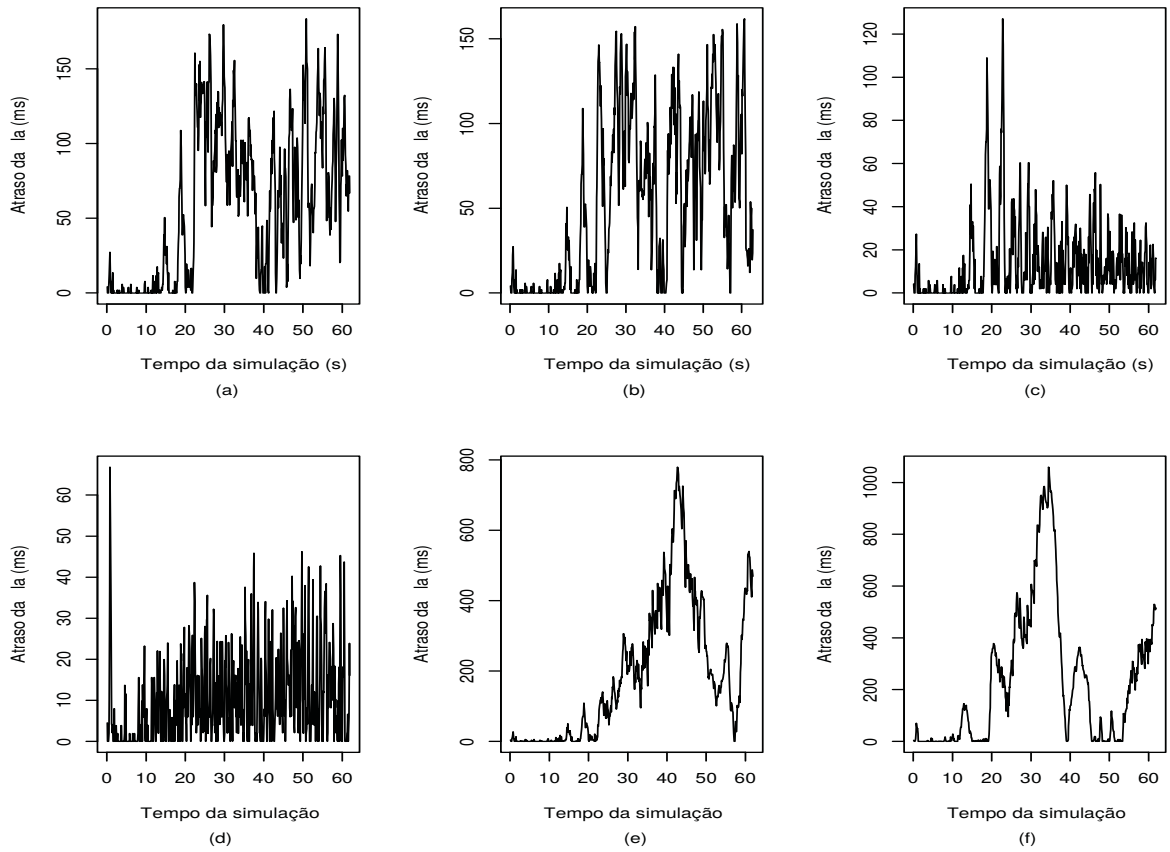

Figura 5.7: Atraso na fila para (a) DRAPA, (b) DRAPTA, (c) ARED, (d) CoDel, (e) RED e (f) Droptail, com 10 fontes de tráfego de fundo ativas para o vídeo BBB.

DRAPTA, a Figura 5.8 mostra a relação de ambos com e sem a LSTM, para as métricas PSNR e SSIM. Para o caso dos AQMs sem a LSTM, a predição do atraso (y_t) foi substituído pelo valor atual do atraso da fila (x_t). Pela Figura, nota-se que a previsão do atraso desempenhado pela LSTM é mandatório para que os AQMs atinjam bons valores de PSNR e SSIM. Percebe-se que, a medida que o número de fontes aumenta, o desempenho do DRAPA e DRAPTA sem o auxílio da rede neural, é reduzido substancialmente, tanto para o PSNR quanto para o SSIM.

Percebe-se que com 12 fontes de tráfego ativas no cenário, o PSNR do DRAPA e DRAPTA sem a previsão do atraso, decai para valores abaixo de 10 dB, enquanto que, para ambos AQMs usando a predição da LSTM, o PSNR permanece em níveis elevados. O mesmo efeito pode ser observado para o SSIM. A previsão do atraso feita pela LSTM permite ao DRAPA e DRAPTA anteciparem o congestionamento, descar-

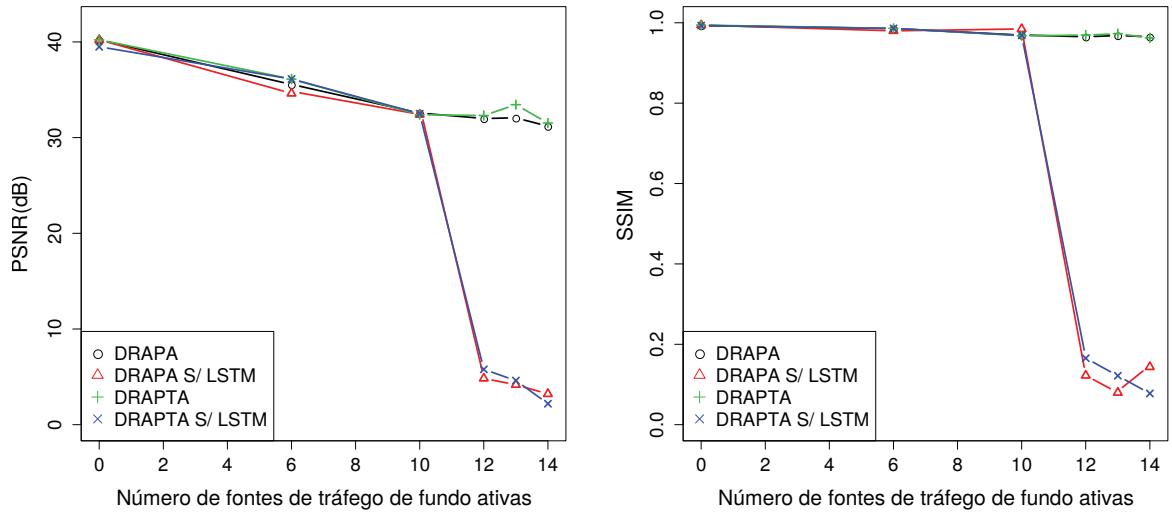

Figura 5.8: Resultados do PSNR e SSIM para o vídeo BBB em função do número de fontes de tráfego ativas para os métodos DRAPA e o DRAPTA, com e sem o uso da LSTM.

tando pacotes e forçando o cliente DASH a requisitar segmentos de menor qualidade.

Logo, a combinação da predição do atraso e diferentes funções de probabilidade de descartes, fizeram com que os métodos propostos atingissem melhor desempenho que os AQMs concorrentes. Ambos AQMs propostos preservam a qualidade da imagem, mesmo em situações com níveis elevados de congestionamento. Além disso, seu uso resultou em um menor número de interrupções no vídeo recebido, circunstância que justifica seus usos.

Este Capítulo tratou da avaliação de desempenho dos métodos propostos. Os AQMs propostos utilizam a rede neural LSTM para realizar a predição do atraso na fila do roteador. A LSTM foi treinada e validada com dados coletados através da transmissão de vídeo na fila do roteador que representa o gargalo da rede. Esses dados foram adquiridos executando o cenário com 12 fontes de tráfego ativas. Nessa configuração, o enlace DSL apresenta congestionamento constante sem descarte de

pacotes. Assim, foi possível treinar a LSTM com um padrão de tráfego que representa um certo nível de congestionamento do enlace. Com o padrão de tráfego em rajadas, espera-se que a LSTM aprenda a antecipar as variações do tráfego de vídeo. Esta capacidade foi usada para realizar descartes antecipados, com o objetivo de fazer as fontes DASH reduzirem a taxa de transmissão para evitar situações de congestionamento. Para analisar o impacto do atraso nos vídeos, as métricas PSNR e SSIM foram usadas, além da quantidade e duração das interrupções. Além dos testes com os métodos propostos, os AQMs RED, ARED, CoDel, PIE e Droptail foram testados. Tanto o DRAPA quando o DRAPTA apresentam resultados melhores que os métodos concorrentes. Na média, eles superam os demais AQMs tanto para o SSIM quanto para o PSNR. Adicionalmente, os métodos propostos permitem que o cliente DASH mantena a reprodução do vídeo constante, mesmo quando o congestionamento do enlace está em níveis elevados. Ambos evitam transições abruptas de segmentos.

CAPÍTULO 6

CONCLUSÕES E TRABALHOS FUTUROS

6.1 Conclusões

O contínuo aumento do tráfego de vídeo DASH pela Internet requer mecanismos que melhorem a qualidade percebida pelo usuário. Em sistemas VoD, a fim de evitar interrupções na imagem, a atual geração de aplicações DASH demandam que o cliente seja apto a armazenar de 20 a 30 segundos de segmentos de vídeo antes de iniciar a reprodução. Porém, em transmissões DASH de eventos ao vivo, armazenar essa quantidade não é viável. Em sistemas DASH ao vivo, o cliente armazena de 2 a 3 segundos de vídeo. Isso requer que o algoritmo do aplicativo de adaptação reaja rapidamente às mudanças na rede. Porém, a característica auto-similar do tráfego pode induzir instabilidade, fazendo o cliente a requisitar segmentos de maior qualidade, paradoxalmente, impactando de forma negativa no vídeo assistido. Caso o enlace seja atingido por uma rajada de pacotes, o atraso da fila pode torná-los inválidos para a reprodução e levar segmentos DASH a expirarem no servidor, gerando impactos severos na qualidade da imagem.

Nesta tese, dois novos algoritmos de gerenciamento ativo de filas (AQM) são propostos, com o objetivo de melhorar a qualidade do vídeo assistido em sistema de transmissão DASH ao vivo. Intitulados de DRAPA (Descarte Randômico para vídeo Adaptativo baseado na Predição do Atraso) e DRAPTA (Descarte Randômico para vídeo Adaptativo baseado na Predição da Tendência do Atraso), os algoritmos realizam descartes de pacotes de acordo com limiares em função do atraso. Os algoritmos utilizam as redes neurais LSTM para realizar a previsão de um passo a frente do atraso da fila - isso é possível devido ao tráfego de vídeo ser auto-correlacionado, podendo

ser previsto. De acordo com o valor futuro do atraso, os AQMs desempenham descartes de pacotes forçando que as fontes TCP reduzam suas taxas de transmissão e consequentemente, que o cliente DASH diminua a qualidade do segmento a ser requisitado, antecipando situações de congestionamento. Esse comportamento possibilita transições mais suaves entre as qualidades dos segmentos DASH, principalmente em cenários onde o enlace está muito congestionado. Diminuindo a qualidade do segmento e notificando as demais fontes sobre futuros congestionamentos, viabiliza a continua reprodução do vídeo no cliente e eleva a qualidade percebida pelo usuário, quando comparado com outros AQMs.

A avaliação de desempenho foi realizada através de uma simulação híbrida misturando tráfego simulado e a transmissão real de fluxos DASH ao vivo. Com ela foi possível avaliar o desempenho dos métodos propostos, bem como de outras estratégias AQMs citadas pela literatura, o que é outra contribuição deste trabalho. Não haviam experimentos mostrando o desempenho dos principais AQMs disponíveis na literatura em relação ao tráfego de vídeo DASH ao vivo. Com a pandemia de COVID-19, fluxos de vídeo ao vivo foram intensificados, o que aumenta a relevância dos resultados atingidos. Além disso, as novas aplicações de realidade virtual demandam a transmissão de vídeo ao vivo em taxas bastantes elevadas e também podem se beneficiar do uso do DRAPA e do DRAPTA.

Os resultados mostram que os métodos propostos superam o ARED, CoDel, Droptail, PIE e RED em relação ao SSIM, PSNR e ao número e duração das interrupções no vídeo, principalmente quando o enlace está altamente congestionado. Tanto o DRAPA quanto a DRAPTA superam os AQMs concorrentes em relação ao valor médio do SSIM e ao valor médio normalizado do PSNR. Essa comparação utiliza a média geral de todos os vídeos em função do número de fontes de tráfego ativas para o PSNR e SSIM, dando uma visão geral dos benefícios que os métodos propostos trazem quando comparados aos AQMs concorrentes. Aplicado na rede de acesso do cliente, local onde os congestionamentos normalmente ocorrem, o DRAPA e o DRAPTA me-

lhoram a qualidade do vídeo e diminuem o número de interrupções percebidas pelo cliente. O uso dos métodos propostos permite que o cliente DASH se adapte às condições da rede antes da ocorrência do congestionamento, resultando em um maior nível do SSIM e PSNR médios. De acordo com os resultados, o Droptail obteve o pior desempenho entre os AQMs, diminuindo a qualidade do vídeo muito antes do enlace tornar-se congestionado. Baseado nos resultados, o uso do Droptail deveria ser evitado.

6.2 Trabalhos Futuros

Apesar dos bons resultados obtidos, diversas alternativas poderiam ser investigadas para obter resultados ainda melhores.

Pode ser investigado o uso das redes neurais LSTM com um grande número de camadas escondidas e um número maior de entradas, levando em conta um histórico passado maior de tráfego. O uso de redes neurais com muitas entradas/camadas escondidas tem sido uma tendência na comunidade de pesquisa nesta área, com resultados excelentes sendo reportados. Além disso, existe a tendência em se implementar redes neurais em *hardware*, o que torna esta alternativa bastante atraente. A expansão do tamanho da rede neural também permite que o treinamento seja realizado utilizando-se um grande conjunto de situações (maior/menor compressão de vídeo, maior/menor quantização, maior/menor resolução, diferentes tipos de vídeo e etc.), o que tornaria o método proposto capaz de reagir adequadamente a diversos casos de aplicação.

Os métodos propostos utilizaram funções de probabilidade do tipo uniforme para realizar o descarte antecipado. Outras funções de distribuição podem ser testadas para obter melhores resultados.

Os resultados foram obtidos através de simulações computacionais, com o auxílio do NS-3. O tráfego de fundo foi gerado utilizando-se o modelo ON-OFF Pareto,

principalmente em função de limitações de poder computacional disponível. Um novo teste poderia ser realizado utilizando clientes reais DASH, com avaliação da qualidade sendo feita também nestes clientes. Um próximo passo seria a implementação dos métodos em roteadores e a realização de testes em ambientes reais, com a transmissão de diversos fluxos simultâneos.

REFERÊNCIAS

ABBAS, G.; MANZOOR, S.; HUSSAIN, M. A stateless fairness-driven active queue management scheme for efficient and fair bandwidth allocation in congested Internet routers. **Telecommunication Systems**, v. 67, n. 1, p. 3–20, January 2018.

ADAMS, R. Active queue management: A survey. **IEEE Communications Surveys Tutorials**, v. 15, n. 3, p. 1425–1476, October 2013.

AKHSHABI, S. et al. Server-based traffic shaping for stabilizing oscillating adaptive streaming players. In: ACM. **Proceeding of the 23rd ACM Workshop on Network and Operating Systems Support for Digital Audio and Video**. : Association for Computing Machinery, 2013.

ALENCAR, J. V. de M. Cardoso e Augusto C. S. Mariano e Carlos D. M. Regis e M. S. Comparação das métricas objetivas de qualidade de vídeos baseadas na similaridade estrutural e na sensibilidade ao erro. **Revista de Tecnologia da Informação e Comunicação**, v. 1, n. 2, p. 33–40, Abril 2012.

AMMAR, D.; BEGIN, T.; GUERIN-LASSOUS, I. A new tool for generating realistic Internet traffic in NS-3. In: **Proceedings of the 4th International ICST Conference on Simulation Tools and Techniques**. 2011. p. 81–83.

ARPACI, M.; COPELAND, J. A. An adaptive queue management method for congestion avoidance in TCP/IP networks. In: **Globecom '00 - IEEE. Global Telecommunications Conference.** 2000. p. 309–315.

AUWERA, G. Van der; DAVID, P.; REISSLEIN, M. Traffic and quality characterization of single-layer video streams encoded with the H.264/MPEG-4 advanced video coding standard and scalable video coding extension. **IEEE Transactions on Broadcasting**, v. 54, n. 3, p. 698–718, September 2008.

AZZOUNI, A.; PUJOLLE, G. **A Long Short-Term Memory Recurrent Neural Network Framework for Network Traffic Matrix Prediction**. 2017. Available on line https://arxiv.org/abs/1705.05690.

BARABAS, M. et al. Evaluation of network traffic prediction based on neural networks with multi-task learning and multiresolution decomposition. In: **IEEE 7th International Conference on Intelligent Computer Communication and Processing**. 2011. p. 95–102.

BEBEN, A. et al. Abma+: Lightweight and efficient algorithm for http adaptive streaming. In: ACM. **Proceedings of the 7th International Conference on Multimedia Systems**. New York, NY, USA: Association for Computing Machinery, 2016.

BERAN, J. et al. Long-range dependence in variable-bit-rate video traffic. **IEEE Transactions on Communications**, v. 43, n. 4, p. 1566–1579, April 1995.

BIERNACKI, A. Analysis and modelling of traffic produced by adaptive HTTP based video. **Multimedia Tools Application**, v. 76, n. 10, p. 12347–12368, May 2017.

BOUTEN, N. et al. Deadline-based approach for improving delivery of SVC-based HTTP adaptive streaming content. In: IEEE. **2014 IEEE Network Operations and Management Symposium (NOMS)**. 2014. p. 1–7.

BROWNLEE, J. **Long short-term memory networks with python: develop sequence prediction models with deep learning**. : Machine Learning Mastery, 2017.

CISCO. **Cisco Visual Networking Index: Forecast and Methodology 2015 to 2020**. 2020. Disponível em: <http://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/complete-white-paper-c11-481360.html>.

CLARK, D. D. D. et al. **IETF Recommendations on Queue Management and Congestion Avoidance in the Internet**. April 1998. RFC 2309. (Request for Comments, 2309).

DISCHINGER, M. et al. Characterizing residential broadband networks. In: **Proceedings of the 7th ACM SIGCOMM Conference on Internet Measurement**. 2007. p. 43–56.

FENG, H.; SHU, Y. Study on network traffic prediction techniques. In: **International Conference on Wireless Communications, Networking and Mobile Computing.** 2005. p. 1041–1044.

FENG, W. et al. A self-configuring RED gateway. In: **IEEE INFOCOM '99 Conference on Computer Communications.** 1999. p. 1320–1328.

FEUVRE, J. L.; CONCOLATO, C.; MOISSINAC, J.-C. **GPAC: Open Source Multimedia Framework**. 2007. Available online https://gpac.wp.imt.fr.

FEUVRE, J. L. et al. Ultra high definition HEVC DASH data set. In: **Proceedings of the 5th ACM Multimedia Systems Conference**. 2014. p. 7–12.

FFmpeg Developers. **FFmpeg Multimedia Framework**. 2019. Available online https://www.ffmpeg.org.

FIGUEROA, A. A. M.; FAVALLI, L. Buffer management for scalable video streaming. In: **Proceedings of the 9th EAI International Conference on Mobile Multimedia Communications**. 2016. p. 18–23.

FITZEK, F. H. P.; REISSLEIN, M. MPEG-4 and H.263 video traces for network performance evaluation. **IEEE Network**, v. 15, n. 6, p. 40–54, 2001.

FLOYD, S.; GUMMADI, R.; SHENKER, S. **Adaptive RED: An algorithm for increasing the robustness of RED's active queue management**. 2001. Tech Report. AT&T Center for Internet Research at ICSI.

FLOYD, S.; JACOBSON, V. Random early detection gateways for congestion avoidance. **IEEE/ACM Transactions on Networking**, v. 1, n. 4, p. 397–413, August 1993.

FOUNDATION, X. (Ed.). **Xiph.Org Foundation**. May 1999. On Line: http://media.xiph.org/video/derf/.

GARRETT, M. W.; WILLINGER, W. Analysis, modeling and generation of self-similar VBR video traffic. **SIGCOMM Comput. Commun. Rev.**, Association for Computing Machinery, v. 24, n. 4, p. 269–280, October 1994.

GETTYS, J.; NICHOLS, K. Bufferbloat: Dark buffers in the Internet. **ACMqueue**, v. 9, n. 11, p. 40–54, November 2011.

GHOBADI, M. et al. Trickle: Rate limiting youtube video streaming. In: ACM. **Proceedings of the 2012 USENIX Conference on Annual Technical Conference**. : USENIX Association, 2012.

GREENGRASS, J.; EVANS, J.; BEGEN, A. C. Not all packets are equal, part I: Streaming video coding and SLA requirements. **IEEE Internet Computing**, IEEE Educational Activities Department, v. 13, p. 70–75, January 2009.

GRIGORESCU, E.; KULATUNGA, C.; FAIRHURST, G. Evaluation of the impact of packet drops due to aqm over capacity limited paths. In: **2013 21st IEEE International Conference on Network Protocols (ICNP)**. 2013. p. 1–6.

GULLI, A.; PAL, S. **Deep learning with Keras**. : Packt Publishing Ltd, 2017.

HAYKIN, S. **Neural Networks: A Comprehensive Foundation**. 2nd. ed. : Prentice Hall PTR, 1998.

____. **Redes Neurais - Princípios e Prática**. : Bookman, 2001.

HOCHREITER, S.; SCHMIDHUBER, J. Long short-term memory. **Neural Computation**, v. 9, n. 8, p. 1735–1780, November 1997.

HOEILAND-JOERGENSEN, T. et al. **The flow queue CoDel packet scheduler and active queue management algorithm**. 2018. Request for Comments.

HUA, Y. et al. Deep learning with long short-term memory for time series prediction. **IEEE Communications Magazine**, v. 57, n. 6, p. 114–119, March 2019.

HUANG, T.-Y. et al. A buffer-based approach to rate adaptation: Evidence from a large video streaming service. **SIGCOMM Comput. Commun. Rev.**, Association for Computing Machinery, v. 44, n. 4, p. 187–198, August 2014.

IRONDI, I.; WANG, Q.; GRECOS, C. Subjective evaluation of H.265/HEVC based dynamic adaptive video streaming over HTTP (HEVC-DASH). In: **Proceedings Volume 9400, Real-Time Image and Video Processing 2015**. 2015. p. 1–11.

JIANG, J.; SEKAR, V.; ZHANG, H. Improving fairness, efficiency, and stability in http-based adaptive video streaming with festive. **IEEE/ACM Transactions on Networking**, IEEE, v. 22, p. 326–340, January 2014.

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, February 2016.

KINGMA, D. P.; BA, J. Adam: A method for stochastic optimization. **arXiv preprint arXiv:1412.6980**, 2014.

KRISHNAPPA, D. K.; BHAT, D.; ZINK, M. Dashing youtube: An analysis of using dash in youtube video service. In: **38th Annual IEEE Conference on Local Computer Networks**. 2013. p. 407–415.

KRUNZ, M.; MAKOWSKI, A. A source model for VBR video traffic based on m/g/∞ input processes. In: **Proceedings. IEEE INFOCOM '98, the Conference on Computer Communications. Seventeenth Annual Joint Conference of the IEEE Computer and Communications Societies. Gateway to the 21st Century**. 1998. p. 1441–1448.

KUA, J.; ARMITAGE, G. Optimising DASH over aqm-enabled gateways using intra-chunk parallel retrieval (chunklets). In: IEEE. **2017 26th International Conference on Computer Communication and Networks (ICCCN)**. 2017. p. 1–9.

KUA, J.; ARMITAGE, G.; BRANCH, P. The impact of active queue management on dash-based content delivery. In: **2016 IEEE 41st Conference on Local Computer Networks (LCN)**. 2016. p. 121–128.

____. A survey of rate adaptation techniques for dynamic adaptive streaming over HTTP. **IEEE Communications Surveys Tutorials**, IEEE, v. 19, p. 1842–1866, March 2017.

KWON, S.-k.; TAMHANKAR, A.; RAO, K. Overview of H.264/MPEG-4 part 10. **J. Visual Communication and Image Representation**, v. 17, n. 31, p. 186–216, August 2006.

LEDERER, S.; MULLER, C.; TIMMERER, C. Dynamic adaptive streaming over HTTP dataset. In: **Proceedings of the 3rd Multimedia Systems Conference**. 2012. p. 89–94.

LELAND, W. E. et al. On the self-similar nature of ethernet traffic. **SIGCOMM Computer Communication Review**, v. 23, n. 4, p. 183–193, October 1993.

LEWCIO, B. et al. Video quality in next generation mobile networks - perception of time-varying transmission. In: IEEE. **2011 IEEE International Workshop Technical Committee on Communications Quality and Reliability (CQR)**. 2011.

LI, Y. et al. An autonomic active queue management mechanism to improve multimedia flow delivery quality. In: **2010 International Conference on Communications and Mobile Computing**. 2010. p. 493–497.

LIU, C.; BOUAZIZI, I.; GABBOUJ, M. Rate adaptation for adaptive http streaming. In: ACM. **Proceedings of the Second Annual ACM Conference on Multimedia Systems**. : Association for Computing Machinery, 2011. p. 169–174.

LOHMAR, T. et al. Dynamic adaptive HTTP streaming of live content. In: **2011 IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks**. 2011. p. 1–8.

MCCULLUM, N. **The ultimate guide to recurrent neural networks in python**. 2020. Available online https://www.freecodecamp.org/news/the-ultimate-guide-to-recurrent-neural-networks-in-python/.

MCQUISTIN, S.; PERKINS, C.; FAYED, M. Tcp goes to hollywood. In: ACM. **Proceedings of the 26th International Workshop on Network and Operating Systems Support for Digital Audio and Video**. : NOSSDAV, 2016a.

____. Tcp hollywood: An unordered, time-lined, tcp for networked multimedia applications. In: IEEE. **2016 IFIP Networking Conference (IFIP Networking) and Workshops**. : IFIP Networking, 2016b.

MORAIS, W. G.; SANTOS, C. E. M.; PEDROSO, C. M. Gerência ativa de filas para melhoria da qualidade na transmissão de vídeo adaptativo. In: SBRT. **XXXVII Simpósio Brasileiro de telecomunicações e processamento de sinais**. 2019. p. 1–5.

MORAIS, W. G. de et al. Application of active queue management for real-time adaptive video streaming. **Telecommunication Systems**, November 2021.

NICHOLS, K.; JACOBSON, V. Controlling queue delay. **ACMqueue**, v. 10, n. 5, p. 20–34, May 2012.

NICHOLS, K. et al. **Controlled delay active queue management**. January 2018. RFC 7567. (Request for Comments, 8289).

PAN, R. et al. **Proportional integral controller enhanced (PIE): A lightweight control scheme to address the bufferbloat problem**. 2017. Request for Comment 8033.

PARK, K.; WILLINGER, W. **Self-Similar network traffic and performance evaluation**. : Wiley & Son, 2000.

PATEL, S.; KARMESHU. A new modified dropping function for congested aqm networks. **Wireless Personal Communications**, v. 104, n. 1, p. 37–55, January 2019.

POURAZAD, M. T. et al. HEVC: The new gold standard for video compression: How does HEVC compare with H.264/AVC. **IEEE Consumer Electronics Magazine**, IEEE, v. 1, n. 12817651, p. 11, July 2012.

PURI, A.; CHEN, X.; LUTHRA, A. Video coding using the H.264/MPEG-4 AVC compression standard. **Signal Processing: Image Communication**, v. 19, n. 9, p. 793–849, July 2004.

QI, Y.; DAI, M. The effect of frame freezing and frame skipping on video quality. In: IEEE. **2006 International Conference on Intelligent Information Hiding and Multimedia**. 2006.

RILEY, G. F.; HENDERSON, T. R. The NS-3 network simulator. In: **Modeling and Tools for Network Simulation**. : Springer, 2010. p. 15–34.

ROBLES, V.; SILLER, M.; WOODS, J. Active discarding packet mechanisms for video transmission. In: **2007 IEEE International Conference on System of Systems Engineering**. 2007. p. 1–5.

SANDVINE. **The global Internet phenomena report COVID-19 spotlight**. 2021. Disponível em: <http://www.sandvine.com/hubfs/Sandvine-Redesign-2019/Downloads/2020/Phenomena/COVID-Internet-Phenomena-Report-20200507.pdf>.

SANTOS, C. E. M.; PEDROSO, C. M. Gerência ativa de filas usando redes neurais lstm para fluxos DASH ao vivo. In: SBRT. **XXXIX Simpósio Brasileiro de telecomunicações e processamento de sinais**. 2021. p. 1–5.

SANTOS, C. E. M.; RIBEIRO, E. P.; PEDROSO, C. M. The application of neural networks to improve the quality of experience of video transmission over IP networks. **Engineering Applications of Artificial Intelligence**, v. 27, p. 137–147, January 2014.

SANTOS, C. E. M.; SILVA, C. A. G. da; PEDROSO, C. M. Improving perceived quality of live adaptative video streaming. **Entropy**, v. 23, n. 8, July 2021.

SATODA, K. et al. Adaptive video pacing method based on the prediction of stochastic tcp throughput. In: IEEE. **2012 IEEE Global Communications Conference (GLOBECOM)**. : GLOBECOM, 2012.

SEELING, P.; REISSLEIN, M. Video transport evaluation with H.264 video traces. **IEEE Communications Surveys Tutorials**, v. 14, n. 4, p. 1142–1165, September 2012.

____. Video traffic characteristics of modern encoding standards: H.264/AVC with SVC and MVC extensions and H.265/HEVC. **The Scientific World Journal**, Hindawi, n. 189481, p. 16, July 2014.

SERRAL-GRACIÀ, R. et al. An overview of quality of experience measurement challenges for video applications in IP networks. In: **Wired/Wireless Internet Communications**. 2010. p. 252–263.

SEUFERT, M. et al. A survey on quality of experience of http adaptive streaming. **IEEE Communications Surveys Tutorials**, v. 17, n. 1, p. 469–492, September 2015.

SODAGAR, I. The MPEG-DASH standard for multimedia streaming over the Internet. **IEEE MultiMedia**, v. 18, n. 4, p. 62–67, April 2011.

STANKIEWICZ, R.; CHOLDA, P.; JAJSZCZYK, A. QoX: What is it really? **IEEE Communications Magazine**, v. 49, n. 4, p. 148–158, April 2011.

TANWIR, S.; PERROS, H. A survey of VBR video traffic models. **IEEE Communications Surveys Tutorials**, v. 15, n. 25, p. 1778–1802, January 2013.

TELECOMMUNICATION, I. T. U. **Vocabulary for performance, quality of service and quality of experience**. 2004. ITU-T P.10/G.100.

____. **Subjective video quality assessment methods for multimedia applications**. 2008. ITU-T P.910.

____. **Network model for evaluating multimedia transmission performance over Internet Protocol**. 2016. ITU-T G.1050.

WANG, Z. et al. Image quality assessment: from error visibility to structural similarity. **IEEE Transactions on Image Processing**, IEEE, v. 13, n. 4, p. 600–612, April 2004.

WANG, Z.; LU, L.; BOVIK, A. C. Video quality assessment based on structural distortion measurement. **Signal Processing: Image Communication**, v. 19, n. 2, p. 121–132, February 2004.

WILLINGER, W. et al. Self-similarity through high-variability: statistical analysis of ethernet lan traffic at the source level. **IEEE/ACM Transactions on networking**, v. 5, n. 1, p. 71–86, February 1997.

YANG, X.; LIU, J.; LI, N. Congestion control based on priority drop for H.264/SVC. In: **2007 International Conference on Multimedia and Ubiquitous Engineering (MUE'07)**. 2007. p. 1–5.

ZHOU, C. et al. Buffer-based smooth rate adaptation for dynamic http streaming. In: IEEE. **2013 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference**. : Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA), 2013.

ZHU, C. et al. A comparison of active queue management algorithms using the OPNET modeler. **IEEE Communications Magazine**, v. 40, n. 6, p. 158–167, 2002.

ZINNER, T. et al. Towards QoE management for scalable video streaming. In: IEICE. **Conference: 21th ITC Specialist Seminar on Multimedia Applications - Traffic, Performance and QoE**. 2010.