Gerência Ativa de Filas usando Redes Neurais LSTM para Fluxos DASH ao Vivo

Carlos Eduardo Maffini Santos, Carlos Marcelo Pedroso

***Resumo*— A transmissão de vídeo ao vivo pela Internet vem se tornando popular, sendo o *Dynamic Adaptive Streaming over HTTP* (DASH) uma das tecnologias mais empregadas para esse tipo de transmissão. Fluxos de vídeo ao vivo são sensíveis à variações no atraso, jitter e perda de pacotes. Assim, os gerenciadores ativos de fila (AQM, *Active Queue Management*) surgem como alternativa para controlar o congestionamento na fila dos roteadores, minimizando os impactos desses parâmetros. Como reação ao congestionamento, a camada de aplicação dos clientes DASH tende a requisitar segmentos de vídeo de qualidades inferiores. Neste artigo, é apresentado um novo método AQM para melhorar a qualidade percebida do usuário em transmissões DASH de vídeo ao vivo. O método proposto faz uso das redes neurais LSTM (*Long Short Term Memory*) de modo a prever futuros congestionamentos na fila do roteador, realizando descartes antecipados e pressionando o cliente DASH a reduzir a qualidade do segmento. Além disso, o artigo apresenta uma análise de desempenho dos principais AQMs citados na literatura em transmissões ao vivo. Os resultados mostram que o método proposto supera os AQMs concorrentes, principalmente em enlaces muito congestionados.**

***Palavras-Chave*— DASH, AQM, LSTM, Multimídia, Fluxos ao vivo**

***Abstract*— Live video streaming over the Internet has become popular, with DASH being the most used technology. Live streaming is sensitive to variations in delay, jitter and packet loss. The Active Queue Management (AQM) arises as an alternative to control the congestion in the router's queue, reducing the quality impairments in live stream. As a reaction to congestion, the DASH client requests low-quality video segments. In this article, we propose an AQM method to improve user-perceived video quality in DASH live streaming. The proposed method uses LSTM (Long Short Term Memory) neural network in order forecast congestion in the router's queue, early discarding packets and pressing the DASH client to reduce the segment quality. Besides, this article presents a performance evaluation of the most cited AQMs in the literature for live streaming. The results show that the proposed AQM overcome the competing AQMs, mainly when the link is highly congested.**

***Keywords*— DASH, AQM, LSTM, Multimedia, Live Streaming**

I. Introdução

Os serviços de entrega de vídeo digital em alta qualidade se tornam cada vez mais populares na Internet, impulsionados por aplicações como teleconferências, IPTV (IP *Television*) e plataformas de *Streaming* de vídeo. Empresas como Netflix e Youtube responderam por mais de 28% do tráfego total da rede mundial em 2020, nos primeiros meses da pandemia de COVID-19 [1]. Além disso, o tráfego gerado pelos usuários da Internet cresceu cerca de 40% neste mesmo ano [1]. Este crescimento foi possível através da evolução da infraestrutura das redes das ISPs (*Internet Service Providers*) e de novos e eficientes algoritmos de codificação de vídeo.

Atualmente, a entrega de fluxo de vídeo de modo dinâmico e adaptativo através do protocolo HTTP (*HyperText Transfer Protocol*), vem se tornando o padrão para a transmissões pela Internet [2]. No DASH (*Dynamic Adaptive Streaming over HTTP*), o vídeo é fragmentado em pequenos segmentos e codificado em diferentes níveis qualidades.

Fluxos de vídeo codificados possuem taxa de transmissão variável (VBR, *Variable Bit Rate*). Dependendo do nível de utilização das filas dos roteadores, esse comportamento pode ocasionar congestionamentos, levando a possíveis perdas de pacotes e aumento do atraso, impactando negativamente na qualidade da imagem percebida pelo usuário.

Como alternativa para conter descartes indesejados de pacotes e melhor acomodar o tráfego, principalmente na última milha, onde tem-se gargalos na rede, as operadoras buscam aumentar a capacidade dos *buffers* dos roteadores, impulsionadas pela queda do preço da memória. Esta tendência pode causar um problema conhecido como *Bufferbloat*, que mantém os dados enfileirados por longos períodos de tempo e aumenta em demasia a latência, causando impactos negativos na qualidade do vídeo [3].

De maneira a atenuar os efeitos da perda de pacotes e atrasos indesejados nas filas dos roteadores, os algoritmos de gerenciamento ativo de filas (AQM - *Active Queue Management*) surgem como importantes reguladores de congestionamentos e mecanismos para mitigar a queda na qualidade de experiência do vídeo [4]. Tais mecanismos são cruciais para aplicações de fluxos ao vivo [5]. Implementado nos roteadores, os AQMs surgiram com o propósito de prevenir o congestionamento da rede, explorando a capacidade de adaptação dos protocolos de camada de transporte, fazendo com que as fontes diminuam suas taxas de transmissão [6]. Muitos dos AQMs relatados pela literatura não foram projetados para tráfego de vídeo DASH em tempo real, realizando descartes randômicos sem considerar o impacto na reprodução do vídeo no cliente. Apesar do considerável esforço de pesquisa na área, poucas delas exploram como o tráfego DASH ao vivo interage com o método AQM [7].

Este trabalho apresenta a proposta de um novo algoritmo de gerenciamento ativo de filas para transmissões de vídeos DASH ao vivo. O método usa redes neurais LSTM para realizar as previsões do atraso da fila no roteador. O AQM desempenha descartes aleatórios antecipando futuros congestionamentos, forçando as fontes de tráfego a reduzirem suas taxas de transmissão. Isso induz o cliente DASH a diminuir antecipadamente a qualidade do segmento. O método foi

Carlos Marcelo Pedroso Departamento de Engenharia Elétrica, Universidade Federal do Paraná, Curitiba-PR, Brasil, E-mails: carlos.maffini@ifpr.edu.br, pedroso@eletrica.ufpr.br.

avaliado em uma topologia híbrida, utilizando vídeos reais trafegando por um cenário simulado implementado em NS-3 (*Network Simulator 3*). A melhora da qualidade de vídeo foi estimada através da métrica PSNR (*Peak Signal Noise Ratio*) e pela quantidade e duração das interrupções percebidas no vídeo. Os resultados mostram que o método proposto melhora a qualidade do vídeo quando comparado aos demais AQMs. Ainda, este trabalho apresenta uma comparação de desempenho dos AQMs citados na literatura em transmissões de fluxos ao vivo DASH.

Além desta seção introdutória, o restante deste trabalho está estruturado da seguinte forma: a Seção II apresenta uma visão geral da tecnologia DASH. Na Seção III são apresentados os principais métodos AQMs citados na literatura. A seção IV exibe o conceito da rede neural LSTM. O método proposto é descrito na Seção V. Na Seção VI a avaliação de desempenho é apresentada. Os resultados obtidos são discutidos na Seção VII. Finalizando, a Seção VIII apresenta as conclusões.

II. Transmissão de fluxo de vídeo de modo dinâmico e adaptativo através do protocolo HTTP

Na transmissão DASH, os vídeos são temporalmente divididos em segmentos e codificados em diferentes taxas de resoluções, ficando sobre responsabilidade do dispositivo cliente requisitar os segmentos ao servidor. Através da constante avaliação da largura de banda disponível, o cliente requisita o segmento que melhor se adapta, na tentativa de manter a execução do vídeo constante.

A fim de garantir um fluxo contínuo na reprodução em sistemas VoD (*Video on Demand*), o cliente DASH deve possuir espaço para armazenar de 20 a 30 segundos de vídeo [2]. Porém, em transmissões de vídeo em tempo real, armazenar essa quantidade de segmentos impõe um significante aumento no atraso, não sendo adequado para esse tipo de aplicação. [8].

Nas transmissões em tempo real, os segmentos são gradualmente criados no servidor de acordo com a ocorrência do evento ao vivo. O servidor continuamente atualiza os segmentos, deletando os mais antigos e criando novos, a fim de manter o cliente sincronizado. Congestionamentos na rede podem ocasionar perdas de segmentos devido ao seu tempo de expiração. Isso pode gerar períodos de congelamento na imagem, afetando negativamente a qualidade percebida pelo usuário.

III. Algoritmos de Gerenciamento Ativo de Filas

Algoritmos de gerenciamento ativo de fila são controladores proativos de congestionamentos que atuam em um roteador [4]. Eles são responsáveis por realizar descartes e/ou enfileiramento de pacotes, de forma a melhor acomodar o tráfego. Para tal, eles utilizam informações disponíveis nos roteadores como o tamanho da fila e a taxa de chegada dos pacotes.

O RED foi o primeiro AQM desenvolvido e um dos mais estudados pela comunidade científica [4]. Proposto em 1993 por Sally Floyd e Van Jacobson em [9], o algoritmo realiza descartes antecipados e randômicos. No RED, a indicação de congestionamento ocorre baseada no tamanho da fila, através de uma média móvel exponencial (EWMA - *Exponential Weighted Moving Average*). O RED necessita de constantes ajustes em seus parâmetros, de modo a melhor se adaptar as condições do tráfego.

Desenhado como uma alternativa para o RED, o *Adaptive RED* (ARED) altera a probabilidade máxima de descarte (max_p) de acordo com as condições atuais da rede, otimizando o mecanismo de descarte de pacotes. O ARED foi originalmente proposto por Feng et al. [10] e posteriormente modificado por Floyd et al. [11].

O CoDel (*Controlled Queue Delay*) [12] foi desenhado de modo a solucionar o problema do *Bufferbloat*. Sua operação é baseada no controle do atraso da fila, através da criação de estampas de tempos fixadas aos pacotes que chegam. O CoDel monitora o atraso experimentado pelos pacotes, de forma que se o atraso for maior que um determinado limiar, o algoritmo entra em estado de descarte, excluindo um pacote e agendando o tempo para o próximo descarte.

O PIE [13] (Proportional Integral Controller Enhanced) é um método que combina os benefícios do descarte randômico do RED com a detecção de congestionamento em função do atraso médio da fila proposto pelo CoDel. No PIE, a probabilidade de descarte é atualizada periodicamente, baseando-se na quantidade de pacotes desenfileirados e no limiar de atraso. Além disso, ele utiliza um valor máximo permitido para que rajadas de pacotes sejam alocadas no *buffer* sem que ocorram descartes.

IV. Rede Neural LSTM

Proposta por Hochreiter and Schmidhuber em [14], a LSTM é um tipo especial de RNN (*Recurrent Neural Network*), cuja arquitetura é composta por unidades chamadas de bloco de memória e é mais complexa que o neurônio artificial. A LSTM é capaz de reter a informação por períodos de tempo maior que as redes neurais recorrentes. Com isso, ela é dita provedora de memória, sendo capacitada a prever séries temporais com dependência de longa duração (LRD - *Long Range Dependence*) com maior precisão do que as RNNs [15].

O bloco de memória da LSTM contém quatro unidades principais: a de esquecimento (f_t), a de entrada (i_t), a dos valores candidatos (S_t) e a unidade de saída (O_t). Essas quatro unidades são responsáveis por manter e ajustar os valores do estado da célula (C_t) e da saída do bloco (y_t). O estado da célula é recorrentemente conectado como forma de reter ou não as informações no bloco de memória. A cada nova iteração os valores de uma nova entrada (x_t) e da saída anterior (y_{t-1}) são usados para determinar a saída das quatro unidades principais. A saída do estado de esquecimento (f_t) é combinado com o valor passado do estado da célula (C_{t-1}) para estabelecer quais informações devem ser esquecidas ou transmitidas para o estado da célula. Da mesma forma, as saídas das unidades i_t e S_t são combinadas e somadas aos valores previamente filtrados por f_t. Assim, a cada iteração o estado da célula (C_t) é dado por:

$$C_t = f_t \otimes C_{t-1} \oplus i_i \otimes S_t \tag{1}$$

A operação da unidade de saída (O_t), determina quais informações de (C_t) devem ser usadas para computar a saída

do bloco de memória (y_t):

$$y_t = tanh(C_t) \otimes O_t \tag{2}$$

V. Método Proposto

Este trabalho propõe um método AQM que realiza descarte aleatórios randômicos baseado na previsão do atraso da fila para fluxos DASH em tempo real.

Programada em python através da biblioteca Keras *deep learning* [16], a rede neural foi desenhada com três entradas e um de saída, de maneira que para cada entrada, um valor de atraso deslocado no tempo é atribuído (x_{t-2}, x_{t-1}, x_t). A saída da rede (y_t) indica a previsão para o próximo atraso na fila do roteador. A cada δ segundos, o atraso da fila ($x_t = \frac{q_l}{C}$) é atualizado e passado a rede LSTM, onde q_l é o tamanho atual da fila em bytes e C a taxa de transmissão do enlace.

O método AQM utiliza três limiares em função do atraso para realizar descartes de pacotes, que são: low_{th}, mid_{th} e up_{th}. Os três limiares podem ser ligados por duas retas (p_1 e p_2), com inclinações diferentes. O eixo x representa os valores do atraso e o eixo y a probabilidade de descarte, com valor máximo 1. As retas são intersectadas em mid_{th}, no eixo x, e w no eixo y, onde este representa o valor máximo de descarte possível para a reta p_1 e o valor mínimo para p_2. Assim, a cada chegada de um novo pacote, se o atraso previsto pela rede LSTM (y_t) estiver entre os limiares low_{th} e mid_{th}, o algoritmo realiza descarte aleatório de pacotes com probabilidade p_1, dado por:

$$p_1 = \frac{(y_t - low_{th})}{(mid_{th} - low_{th})} * w \tag{3}$$

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:

$$p_2 = \frac{[y_t - (w * y_t) - mid_{th} + (w * up_{th})]}{(up_{th} - mid_{th})} \tag{4}$$

Finalmente, caso y_t exceda o limiar up_{th}, todos os pacotes que chegam a fila serão descartados. A lógica do AQM proposto é apresentada pelo Algoritmo 1.

Considerando que o método proposto é submetido a tráfegos de vídeo DASH em tempo real e a habilidade de aprendizado da rede LSTM, a reparametrização dos limiares é desnecessária para casos em que as condições do tráfego da rede sejam alteradas. Ainda, o método considera que o tráfego de vídeo é alocado em filas separadas dos demais tipos de tráfego, através de algoritmos de escalonamento de pacotes.

Algorithm 1 Algoritmo do método proposto

A cada δ milésimos:
  Executa a LSTM e atualiza y_t;
Para cada pacote que chega na fila:
if $low_{th} < y_t \leq mid_{th}$ **then**
  calcula a probabilidade p_1;
  $n \leftarrow rand()$;
  if $p_1 \leq n$ **then**
    descarta o pacote;
  end
end
else if $mid_{th} < y_t \leq up_{th}$ **then**
  calcula a probabilidade p_2;
  $n \leftarrow rand()$;
  if $p_2 \leq n$ **then**
    descarta o pacote;
  end
end
else
  descarta o pacote;
end

VI. Avaliação de Desempenho

O método proposto foi avaliado através da integração de um cliente/servidor "real" DASH e o simulador NS-3. O cliente e servidor foram implementados através do *Framework Multimedia GPAC*. A topologia da rede foi desenhada de acordo como recomendado pela norma G.1050 da ITU (*International Communication Union*) [17], a qual descreve um modelo para avaliar transmissões multimídia sobre redes IP. A Figura 1 mostra o cenário usado nas simulações. O enlace DSL (*Digital Subscriber Line*) é empregado de modo de simular o gargalo que ocorre nas redes de acesso. Tanto o método AQM como os demais comparados, foram instalados no roteador borda, diretamente conectado ao roteador do cliente.

De modo a inserir tráfego de fundo concorrente com o video DASH ao vivo, servidores e clientes do tipo TCP On-Off foram criados. A duração dos estados On e Off foram modeladas por variáveis randômicas seguindo as distribuições de probabilidade de Pareto e Exponencial, respectivamente. Este tipo de modelo possibilita a simulação de tráfego de vídeo com características auto-similares [18]. Ambos os estados foram configurados usando a duração e o intervalo de tempo médio entre os quadros dos vídeos. Durante o período On um fluxo de taxa constante é produzido.

Várias simulações foram realizadas variando (i) o vídeo transmitido, (ii) o método AQM e (iii) o número de fontes de tráfego de fundo. Para este último a quantidade de fontes foi variada de 0 até 14, cada uma gerando 500 kbps de tráfego. Além do método proposto e do Droptail, os seguintes AQMs foram testados: ARED, RED, CoDel e PIE. Em específico, o RED foi configurado com os limiares max_{th} e min_{th} com 250 e 500 pacotes, respectivamente, possibilitando que o AQM suporte 1 segundo de tráfego, para uma MTU (*Maximum Transfer Unit*) de 1500 bytes. O *buffer* do Droptail foi configurado com capacidade infinita, simulando um *bufferbloat*.

Seis vídeos full-HD (*High Definition*) disponíveis publicamente em [19] foram utilizados nos testes: Big Buck Bunny (BBB), Sunflower (SF), Rush Hour (RH), Pedestrian Area (PA), Riverbed (RB) e Touchdown Pass (TP). Eles foram escolhidos de forma a compor diferentes tipos de gênero, movimentação e detalhe, gerando diversidade nas simulações.

Os vídeos foram codificados *offline* com a ferramenta *ffmpeg* [20] e os segmentos gerados ao vivo pelo servidor DASH, contendo 1 segundo de duração cada, de acordo como recomendado por S. Lederer et al. em [21]. Além disso, cada

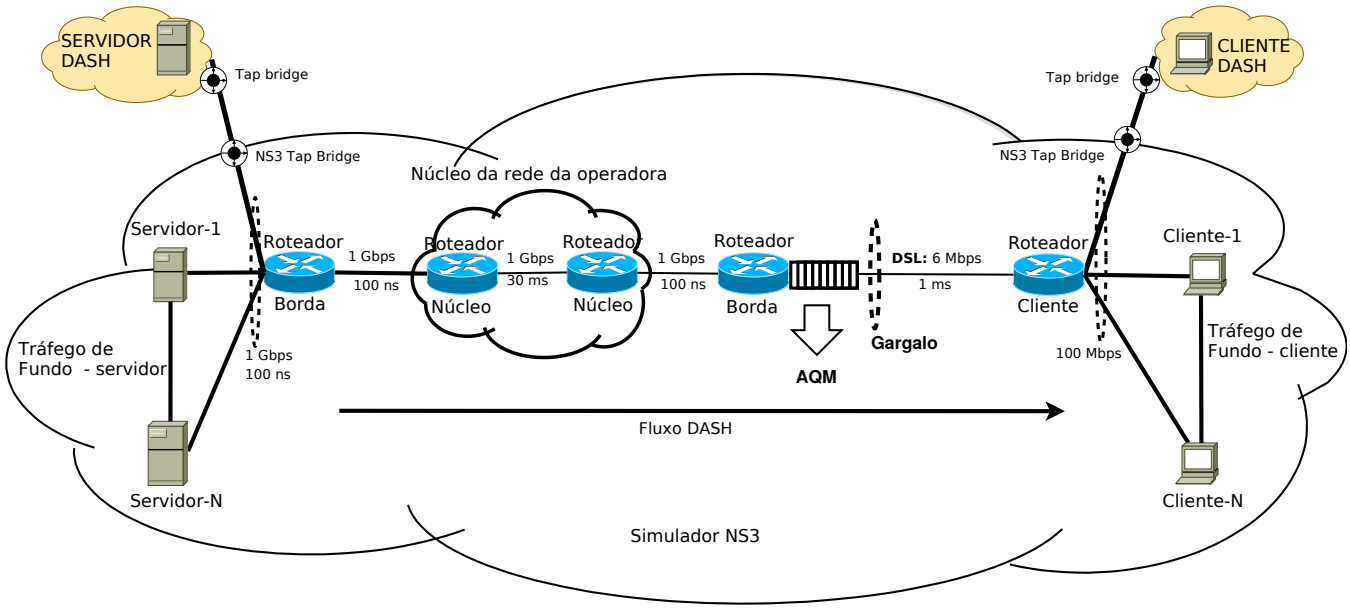

Fig. 1. Topologia para avaliação de desempenho.

vídeo foi codificado em cinco representações com diferentes qualidades: 349, 600, 927, 2114 e 4464 kbps, compatível com outros estudos [22].

A. Parametrização do Método Proposto

O descarte de pacotes desempenhado pelo método proposto é baseado no atraso previsto pela rede neural. Portanto, como forma de atribuir valores para os limiares low_{th}, mid_{th} e up_{th}, o impacto do atraso sobre o PSNR médio, foi avaliado. Para tal, os vídeos BBB, SF e TP foram transmitidos através do NS-3 em um cenário ponto-a-ponto com 6 Mbps de taxa de transmissão entre os enlaces e sem fontes de tráfego de fundo concorrendo com os vídeos. A cada transmissão, o atraso do enlace foi acrescido de 10 milésimos de segundo, até atingir 300ms. O PSNR foi mapeado em MOS (*Mean Opinion Score* - métrica subjetiva, ranqueada de 1 (péssimo) a 5 (excelente), frequentemente usada para medir a Qualidade de Experiência dos usuários), de modo a facilitar a visualização do impacto do atraso na qualidade dos vídeos.

Com o atraso abaixo de 150ms, o PSNR médio é alto e o MOS excelente. Assim, o valor atribuído a low_{th} foi de 150ms; abaixo desse valor não existe a necessidade de descartes de pacotes. Entre 150 e 210ms, o MOS é reduzido de excelente (5) para ruim (2) e entre 210ms e 300ms, para péssimo (1). Logo, low_{th}, mid_{th} e up_{th} foram ajustados para 150, 210 e 300ms, respectivamente. A intersecção entre as retas p_1 e p_2 foi configurada em $w = 0.8$, de modo que a maior parte dos descartes sejam concentrados em p_1. Para a variável δ o valor atribuído foi de 100ms.

VII. Resultados

A Figura 2 apresenta os resultados do PSNR médio em função da quantidade de fontes de tráfego ativas. Nela é possível perceber que o método proposto supera os demais AQMs, com a diferença aumentando quando mais fontes de tráfego concorrem. O ARED alcança o segundo melhor desempenho para a grande maioria dos vídeos. O pior comportamento fica por conta do Droptail, devido ao massivo aumento do atraso imposto pelo *bufferbloat*. No RED, pontualmente quando o enlace está bastante congestionado (14 fontes de tráfego ativas), uma boa performance é alcançada, para a maioria dos vídeos. Isso deve-se ao alto nível de congestionamento e ao tamanho pequeno da fila (750kbytes), levando a uma maior quantidade de descartes e suavizando as transições abruptas entre os segmentos. Porém, com 12 ou 13 fontes ativas, o RED obtém o segundo pior desempenho. Nesse caso, o cliente DASH requisita segmentos das mais elevadas qualidades, como reação ao comportamento em rajada do tráfego agregado. Como, durante a transmissão, congestionamentos atingem a fila do roteador, o segmento acaba expirando. Isso faz com que o cliente DASH seja duplamente penalizado, por tornar o atual segmento inválido e por perder os próximos segmentos gerados no servidor.

A Tabela I mostra a quantidade (N) e a duração (D) das interrupções ocorridas no vídeo BBB. É possível observar que o método proposto supera os demais AQMs em ambas as métricas. O AQM proposto interrompe o vídeo uma única vez e por apenas 1 segundo, quando o nível de congestionamento está muito elevado. Com 10 fontes de tráfego ativas, o Droptail e o RED interrompem o vídeo. Acima de 12 fontes, o Droptail falha em tocar o vídeo por completo.

No geral o AQM proposto apresenta um desempenho superior aos demais AQMs tanto em relação ao PSNR, quanto no número e duração das interrupções dos vídeos.

VIII. Conclusões

O contínuo aumento do tráfego de vídeo DASH pela Internet requer mecanismos que melhorem a qualidade percebida pelo usuário. Neste artigo, um novo algoritmo AQM é proposto para

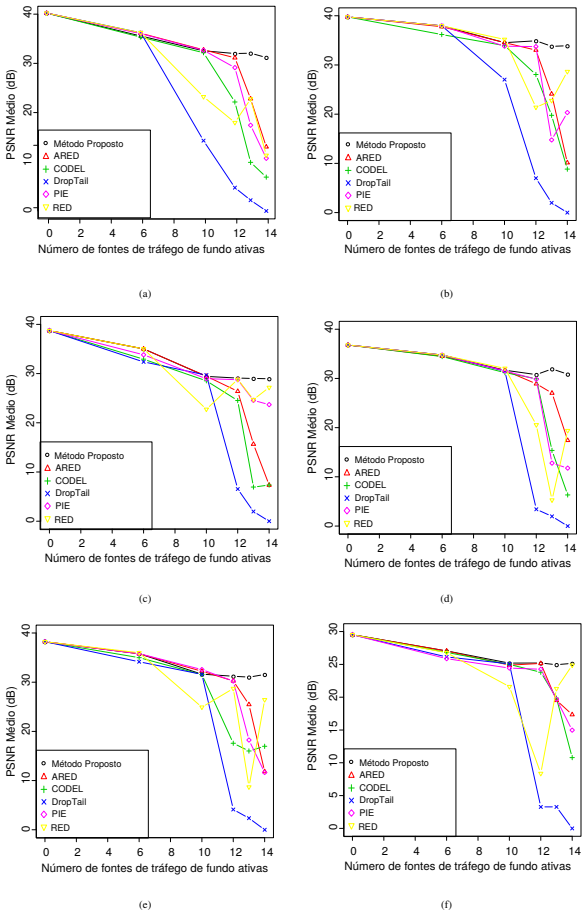

Fig. 2. PSNR médio para os vídeos (a) BBB, (b) RH, (c) SF, (d) TP, (e) PA e (f) RB.

TABELA I

NÚMERO E DURAÇÃO DAS INTERRUPÇÕES PARA O VÍDEO BBB

Background Traffic Sources->	10		12		13		14	
	N	D (s)	N	D (s)	N	D (s)	N	D (s)
Proposed Method	0	0	0	0	0	0	1	1
ARED	0	0	1	1	2	9	2	19
CoDel	0	0	2	9	1	28	1	32
Droptail	4	15	1	35	1	38	1	40
RED	2	12	1	20	1	13	1	26
PIE	0	0	1	3	1	19	2	19

transmissões de fluxos DASH ao vivo. Redes neurais LSTM são usadas de forma a realizar a predição do atraso a fila, possibilitando que o cliente DASH diminua a qualidade do segmento requisitado, antecipando o congestionamento. Através de uma simulação híbrida misturando tráfego simulado e a transmissão real de fluxos DASH ao vivo, foi possível avaliar o desempenho do método proposto, bem como de outras estratégias AQMs.

Os resultados mostram que o AQM proposto supera o ARED, CoDel, Droptail, PIE e RED em relação ao PSNR médio e ao número e duração das interrupções no vídeo, principalmente quando o enlace está altamente congestionado.

REFERÊNCIAS

[1] The global Internet phenomena report covid-19 spotlight. [Online]. Available: https://www.sandvine.com/hubfs/Sandvine_Redesign_2019/Downloads/2020/Phenomena/COVIDInternetPhenomenaReport20200507.pdf

[2] N. Bouten, M. Claeys, S. Latré, J. Famaey, W. V. Leekwijck, and F. D. Turck, "Deadline-based approach for improving delivery of SVC-based HTTP adaptive streaming content," in *2014 IEEE Network Operations and Management Symposium (NOMS)*. IEEE, May 2014, pp. 1–7.

[3] J. Gettys and K. Nichols, "Bufferbloat: Dark buffers in the Internet," *ACMqueue*, vol. 9, no. 11, pp. 40–54, November 2011.

[4] R. Adams, "Active queue management: A survey," *IEEE Communications Surveys Tutorials*, vol. 15, no. 3, pp. 1425–1476, October 2013.

[5] X. Yang, J. Liu, and N. Li, "Congestion control based on priority drop for H.264/SVC," in *2007 International Conference on Multimedia and Ubiquitous Engineering (MUE'07)*. IEEE, April 2007, pp. 1–5.

[6] Y. Li, X. Gong, W. Wang, X. Que, and J. Ma, "An autonomic active queue management mechanism to improve multimedia flow delivery quality," in *2010 International Conference on Communications and Mobile Computing*. IEEE, April 2010, pp. 493–497.

[7] J. Kua, G. Armitage, and P. Branch, "The impact of active queue management on DASH-based content delivery," in *2016 IEEE 41st Conference on Local Computer Networks (LCN)*. IEEE, November 2016, pp. 121–128.

[8] T. Lohmar, T. Einarsson, P. Fröjdh, F. Gabin, and M. Kampmann, "Dynamic adaptive HTTP streaming of live content," in *2011 IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks*. IEEE, June 2011, pp. 1–8.

[9] S. Floyd and V. Jacobson, "Random early detection gateways for congestion avoidance," *IEEE/ACM Transactions on Networking*, vol. 1, no. 4, pp. 397–413, August 1993.

[10] W. Feng, D. D. Kandlur, D. Saha, and K. G. Shin, "A self-configuring RED gateway," in *IEEE INFOCOM '99 Conference on Computer Communications*. IEEE, March 1999, pp. 1320–1328.

[11] S. Floyd, R. Gummadi, and S. Shenker, "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.

[12] K. Nichols and V. Jacobson, "Controlling queue delay," *ACMqueue*, vol. 10, no. 5, pp. 20–34, May 2012.

[13] R. Pan, P. Natarajan, F. Baker, and G. White, "Proportional integral controller enhanced (PIE): A lightweight control scheme to address the bufferbloat problem," 2017, request for Comment 8033.

[14] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, November 1997.

[15] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural Networks*, vol. 61, no. 6, pp. 85–117, May 2015.

[16] A. Gulli and S. Pal, *Deep learning with Keras*. Packt Publishing Ltd, 2017.

[17] I. T. U. Telecommunication, "Network model for evaluating multimedia transmission performance over Internet protocol," 2016, iTU-T G.1050.

[18] D. Ammar, T. Begin, and I. Guerin-Lassous, "A new tool for generating realistic Internet traffic in NS-3," in *Proceedings of the 4th International ICST Conference on Simulation Tools and Techniques*. ACM, March 2011, pp. 81–83.

[19] P. Seeling and M. Reisslein, "Video transport evaluation with H.264 video traces," *IEEE Communications Surveys Tutorials*, vol. 14, no. 4, pp. 1142–1165, September 2012.

[20] FFmpeg Developers, "FFmpeg multimedia framework," 2019, available on line https://www.ffmpeg.org.

[21] S. Lederer, C. Muller, and C. Timmerer, "Dynamic adaptive streaming over HTTP dataset," in *Proceedings of the 3rd Multimedia Systems Conference*. ACM, February 2012, pp. 89–94.

[22] D. K. Krishnappa, D. Bhat, and M. Zink, "Dashing youtube: An analysis of using dash in youtube video service," in *38th Annual IEEE Conference on Local Computer Networks*. IEEE, October 2013, pp. 407–415.