Amintas Coelho Miranda Dutra

A INFLUÊNCIA DO PACING DO QUIC EM SISTEMAS DASH

Trabalho de Graduação

Universidade Federal de Pernambuco
secgrad@cin.ufpe.br
www.cin.ufpe.br/~graduacao/

RECIFE
2017

Universidade Federal de Pernambuco
Centro de Informática
Graduação em Engenharia da Computação

Amintas Coelho Miranda Dutra

A INFLUÊNCIA DO PACING DO QUIC EM SISTEMAS DASH

Trabalho apresentado ao Programa de Graduação em Engenharia da Computação do Centro de Informática da Universidade Federal de Pernambuco como requisito parcial para obtenção do grau de Bacharel em Engenharia da Computação.

Orientador: *Djamel Sadok*

RECIFE
2017

Agradecimentos

Em primeiro lugar sou grato a Deus por ter me dado a oportunidade de iniciar e concluir esta jornada. Ele me deu forças para trilhar o caminho e também colocou pessoas ajudadoras para me impulsionar no percurso.

Sou imensamente grato a minha eterna namorada e também esposa, Rebeka, por ter me apoiado na caminhada, pela motivação e paciência sempre que precisei, por acreditar em mim e também por sonhar e planejar comigo os próximos caminhos a serem trilhados.

Agradeço a toda minha família por ter me apoiado e sustentado em todos os momentos. Meus pais, José Amintas e Ilinea Dias, que sempre estiveram à disposição para me ajudar e aos meus irmãos, Bruna Muniz, Paulo Coelho e Lucas Coelho, que me animaram e deram disposição sempre que precisei.

A todos os colegas e amigos do Grupo de Pesquisa, meu muito obrigado. Em especial, agradeço a Wesley Davison, Daniel Bezerra, Igor Oliveira e Thiago Rodrigues que me ajudaram com todas as dúvidas técnicas que tive. Agradeço também aos professores Djamel Sadok, Judith Kelner e Stenio Fernandes que me permitiram ter a excelente experiência de trabalhar no Grupo de pesquisa e me deram suporte no decorrer da minha graduação.

Agradeço também a Cristiane Silva, líder e amiga de trabalho, por todo o apoio que me deu nos meses finais da graduação. Seu apoio foi muito importante para que os últimos dias dessa jornada pudessem ser trilhados mais rapidamente.

Por fim, agradeço a todos que me apoiaram direta e indiretamente neste período da graduação. Não conseguirei lembrar de todos, mas registro aqui o meu “Muito obrigado”!

Resumo

Nas últimas décadas o protocolo HTTP teve um grande crescimento e tornou-se o protocolo mais utilizado na Web, sendo a maior parte do seu tráfego formado por vídeo sob demanda. Um dos padrões utilizados para transmissão de áudio e vídeo sob demanda é o Dynamic Adaptive Streaming over HTTP (DASH). Estudos anteriores mostram que o DASH não consegue consumir toda a largura de banda disponível quando utiliza o protocolo TCP, principalmente devido ao seu algoritmo de controle de congestionamento ao detectar perda de pacotes. A técnica de *Pacing* do TCP já foi utilizada anteriormente para tentar reduzir a perda de pacotes, porém sem sucesso. O objetivo deste trabalho é analisar a influência do *Pacing* do QUIC em Sistemas DASH, comparando com resultados do TCP. A análise será realizada através de extração de métricas que mensuram a qualidade da conexão, as quais serão extraídas durante a transmissão de um *streaming* de vídeo em DASH entre um servidor e um cliente. Os resultados finais apontam que o QUIC conseguiu fornecer melhor qualidade ao cliente que o TCP, porém a influência do *Pacing* do QUIC sobre o DASH se mostrou inconclusiva nos experimentos realizados, enquanto o desligamento desta técnica no TCP mostrou melhor desempenho em cenários com grande largura de banda disponível.

Palavras-chave: DASH, QUIC, TCP, PACING

Abstract

In the last decades the HTTP protocol had a great growth and became the most used protocol in the Web, being the majority of its traffic formed by video on demand. One of the standards used for audio and video on demand streaming is Dynamic Adaptive Streaming over HTTP (DASH). Previous studies have shown that DASH is not able to consume all the available bandwidth when using TCP protocol, mainly due to its congestion control algorithm when detecting packet loss. The TCP PACING technique has been used previously to try to reduce packet loss, but to no avail. The objective of this work is to analyze the influence of QUIC PACING in DASH Systems. The analysis will be performed by extracting metrics that measure the quality of the connection, which will be extracted during the transmission of DASH video streaming between a server and a client. The final results point out that QUIC was able to provide better quality to the client than TCP, but the influence of the QUIC Pacing on DASH was inconclusive in the experiments carried out, whereas in TCP the shutdown of this technique showed better performance in scenarios with large width Bandwidth available.

Keywords: DASH, QUIC, TCP, PACING

Lista de Figuras

Lista de Tabelas

Lista de Acrônimos

Sumário

1.Introdução

Criado inicialmente para transmissão de textos, o HTTP (*Hypertext Transfer Protocol*) é um protocolo da camada de aplicação que foi adotado em larga escala e que define um padrão para troca de informações entre clientes e servidores. Nas últimas décadas o protocolo teve um notório crescimento e precisou se adaptar para suportar troca de imagens e também vídeo. Este crescimento fez com que o HTTP se tornasse atualmente o protocolo mais utilizado na Web, sendo a maior parte do seu tráfego formado por vídeo sob demanda (SANDVINE, 2016).

Um dos motivos pelo qual o HTTP se tornou tão popular foi o surgimento da tecnologia DASH (*Dynamic Adaptive Streaming over* HTTP), que é um dos padrões utilizados para streaming de áudio e vídeo. O HTTP foi escolhido como protocolo pela sua ampla difusão, o que torna o DASH um padrão que pode ser utilizado sem ter a necessidade de desenvolver um protocolo específico para a troca de informações.

A arquitetura utilizada pelo DASH é cliente-servidor, embora a forma como foi idealizado seja bem diferente de padrões bem conhecidos, como o *download* progressivo. No DASH não é o servidor que decide a qualidade do conteúdo a ser enviado ao cliente, mas este último é quem decide o que deseja receber. A escolha é feita através de cálculos do cliente que determinam as condições de sua conexão e então permitem que o mesmo decida qual a qualidade do conteúdo que deseja receber. Esta característica do DASH permite com que haja uma melhor eficiência no servidor e uma melhor QoE (Quality Of Experience) para o cliente (SODAGAR, 2011) e (RIISER et al., 2012).

Para conseguir atender às necessidades das aplicações, como as que utilizam DASH por exemplo, o HTTP precisa de confiabilidade no transporte, isto é, que as requisições dos clientes cheguem sem problemas ao servidor e que os dados que este envia sejam recebidos com sucesso por aqueles. Por isso geralmente ele executa acima do TCP (*Transmission Control Protocol*) (Network Working Group, 1999). Este protocolo de transporte fornece a confiabilidade que o HTTP necessita, porém possui uma característica de enviar os dados do servidor para o cliente em rajadas, o que pode ser prejudicial principalmente em redes cujos roteadores possuem pequenos *buffers*, devido à sobrecarga nos momentos da rajada.

Sabe-se que esta característica de rajadas do TCP causa uma menor vazão e maior perda de pacotes (AGGARWAL; SAVAGE; ANDERSON, 2012). Visando neutralizar os impactos citados foi criada a técnica de *Pacing*, a qual acaba com o comportamento em rajadas, pois distribui os pacotes de forma que eles sejam enviados em quantidades mais uniformes no decorrer do tempo.

Devido ao desempenho da Web estar inferior às necessidades das aplicações atuais, a comunidade começou a propor novas sugestões de melhorar o desempenho da *Web*, dentre elas o aumento de conexões TCP paralelas para aumentar a vazão na troca de dados e reduzir o impacto em caso de perda de pacotes (GRIGORIK, 2013). Algumas dessas paliativas foram aperfeiçoadas e implementadas pela Google em um protocolo conhecido como SPDY (pronucia-se *speedy* e não trata-se de um acrônimo). Como melhorias principais deste protocolo podemos citar:

- Multiplexação de requisições e de respostas;
- Priorização de fluxos, permitindo que os conteúdos mais importantes possam ser enviados em fluxos separados, os quais podem ser priorizados para terem um tempo de resposta melhor;

- *Server Push*, que permite que o servidor envie conteúdo que o cliente provavelmente irá necessitar, mesmo que este ainda não tenha solicitado. Reduzindo a quantidade de requisições e consequentemente de RTT, o que melhora a velocidade percebida pelo cliente;

Estas melhorias trouxeram resultados tão positivos que a versão 2.0 do protocolo HTTP foi baseada em diversas funcionalidades do SPDY (GRIGORIK, 2013).

Apesar de todos os benefícios apresentados, o protocolo da Google executa sobre o TCP e consequentemente sofre de todos os problemas relacionados às limitações deste último, dentre as quais podemos citar:

- Latência: Ao utilizar o TCP um cliente precisa gastar no mínimo 1 RTT para estabelecer uma conexão com um servidor toda vez que se comunicar com ele. Ao utilizar uma conexão criptografada com TLS esta latência pode triplicar.
- HOL *blocking*: O TCP não permite recebimento de pacotes desordenados. Isso quer dizer que caso o pacote de número 2 chegue antes do 1, toda a conexão fica bloqueada enquanto o primeiro pacote não chegar.

Devido a estes problemas serem inerentes ao TCP a comunidade entendeu que o protocolo precisava ser otimizado ou substituído. A primeira opção torna-se muito lenta devido ao protocolo estar implementado não só em clientes e em servidores, mas também em aplicações de rede espalhadas por toda a internet (MEGYESI; KRÄMER; MOLNÁR, 2016). Por este motivo a Google propôs um novo protocolo chamado QUIC (*Quick UDP Internet Connections*) (Network Working Group, 2016), o qual já possui o HTTP 2.0 implementado, porém ao invés de TCP utiliza o protocolo UDP (*User Datagram Protocol*), o qual é conhecido por não possuir confiabilidade de transporte. Porém o QUIC foi idealizado de uma forma que ele garante a confiabilidade mesmo utilizando o UDP, o qual não possui as limitações de seu concorrente, permitindo que o protocolo da Google enderece os principais pontos negativos citados anteriormente.

Vimos que o HTTP impulsionou evoluções nos protocolos da internet, os quais são testados para identificar suas limitações e então melhorias são propostas para que novos avanços possam ser feitos. Com o intuito de continuar a análise de novos protocolos, o presente trabalho tem o objetivo de analisar como sistemas DASH se comportam ao utilizar o QUIC e qual a influência que sofrem quando este liga ou desliga sua técnica de *Pacing*.

2. *Background* Técnico

2.1. QUIC

QUIC (*Quick UDP Internet Connections*) é um protocolo de transporte desenvolvido pela Google. O mesmo foi criado com o objetivo de trazer um melhor desempenho à Web e é implementado sobre o protocolo UDP de forma a evitar os problemas inerentes ao TCP.

O protocolo da Google provê segurança através de criptografia própria e já utiliza os padrões do HTTP/2. Para ser comparável ao QUIC, o TCP precisa utilizar TLS e também HTTP/2, porém o primeiro protocolo possui diversas vantagens nessa comparação que vão além do fato da criptografia ser utilizada por padrão. As principais vantagens são referentes a:

- **Estabelecimento da conexão (*handshake*)**: TCP+TLS utiliza entre 1 e 3 *Round Trip Time* (RTT), enquanto o QUIC frequentemente usa 0.
  Na primeira vez em que uma conexão QUIC é estabelecida há 1 RTT obrigatório para que o cliente receba os certificados e as credenciais do servidor. Nas próximas vezes o *handshake* já não é mais necessário e o cliente pode usar os dados obtidos anteriormente para mandar requisições diretamente ao servidor. Este comparativo está demonstrado na Figura 1 e resumido na Tabela 1.

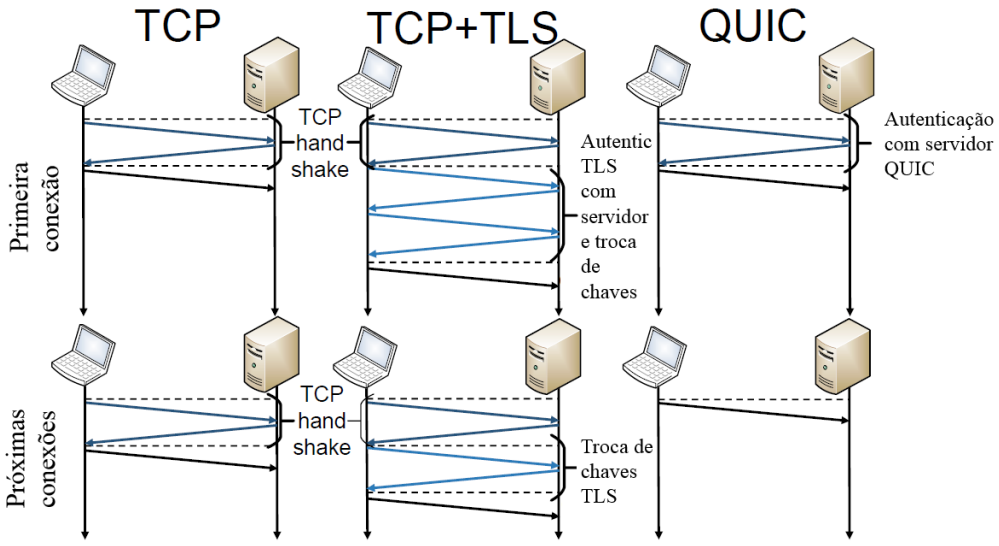

Figura 1 – Comparativo do *handshake*

Tabela 1 – Comparativo do *handshake*

	TCP	**TCP+TLS**	**QUIC**
Primeira conexão	1 RTT	3 RTT	1 RTT
Próximas conexões	1 RTT	2 RTT	0 RTT

- **Algoritmo de Controle de Congestionamento**: Atualmente o algoritmo implementado por padrão no QUIC é uma modificação do TCP CUBIC, provendo uma maior riqueza de informações que a implementação clássica.

Um exemplo dessa riqueza é a escolha do QUIC em determinar uma nova numeração para todos os pacotes, seja ele um envio original ou uma retransmissão. Isso permite que o servidor QUIC consiga diferenciar facilmente entre um ACK de um pacote normal e de uma retransmissão. Outro exemplo é que no QUIC o ACK de um pacote contém a informação de atraso de processamento, isto é, a diferença entre o momento do envio do ACK e do recebimento do pacote, o que traz uma maior precisão no cálculo do RTT.

- **Multiplexação**: Um dos maiores problemas inerentes ao TCP é conhecido como *head-of-line-blocking* (HOL *blocking*) e significa que quando um pacote é perdido, toda a conexão fica parada até o mesmo ser retransmitido e recebido, pois este protocolo não suporta entrega fora de ordem.
  Como o QUIC foi desenvolvido desde o começo para operar de forma multiplexada, ele possui diversos fluxos de dados dentro de uma conexão. Sendo assim, quando há a perda de um pacote apenas o seu fluxo é comprometido, enquanto os outros continuam trocando informações normalmente. Isso significa que o QUIC suporta entrega fora de ordem de pacotes. A multiplexação do QUIC e sua comparação com o TCP é demonstrada na Figura 2.

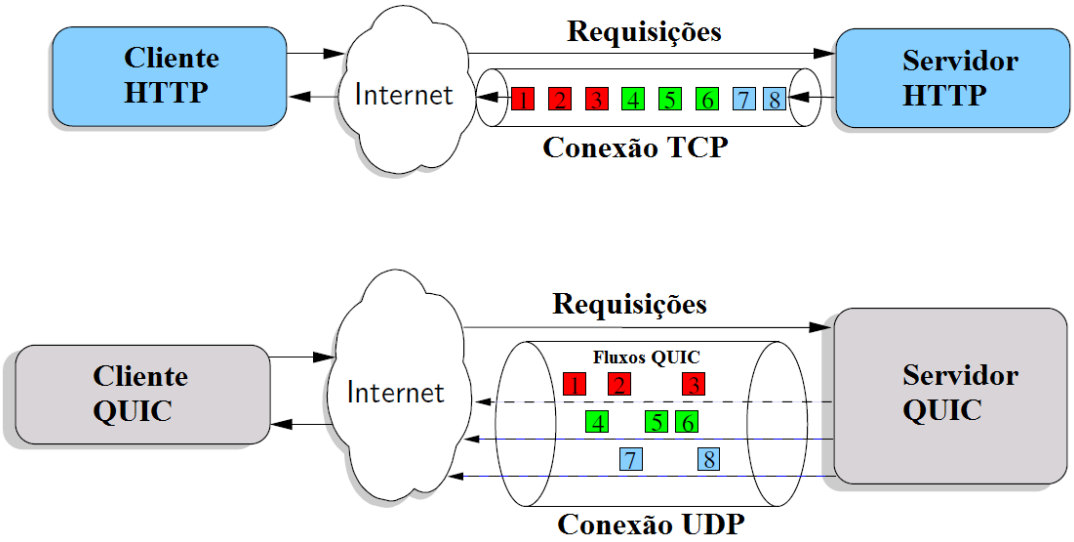

Figura 2 – Multiplexação do TCP e QUIC

Além de reduzir o impacto do HOL, a multiplexação também permite ao QUIC utilizar priorização entre fluxos e também permite a compressão de cabeçalhos HTTP em uma mesma conexão.

- **Migração de Conexão**: Uma conexão TCP é identificada por uma 4-upla (IP de origem, porta de origem, IP destino, porta de destino). Como o tráfego Web de dispositivos móveis tem aumentado, a continuidade da transmissão durante a migração da conexão entre seus adaptadores de rede (Wireless e dados móveis, por exemplo) tem sido um desafio.
  Ao contrário da 4-upla do TCP, uma conexão QUIC é identificada por um número de 64 bits (CID) gerado aleatoriamente pelo cliente. No caso do TCP, quando existe alguma alteração na 4-upla a conexão precisa ser

reestabelecida. O QUIC resolve o problema da migração de conexão, pois durante a transição o CID não se altera e então a transmissão continua sem a necessidade de reestabelecimento.

2.2. *PACING*

Para entender a utilidade do *Pacing*, é útil entender o que é um tráfego em rajadas. Este tipo de tráfego é caracterizado por grandes quantidades de dados sendo enviados em um curto período de tempo, o qual é seguido por um período de ociosidade ou de envios com quantidades menores. A Figura 3 demonstra um tráfego em rajadas, onde os picos são as rajadas.

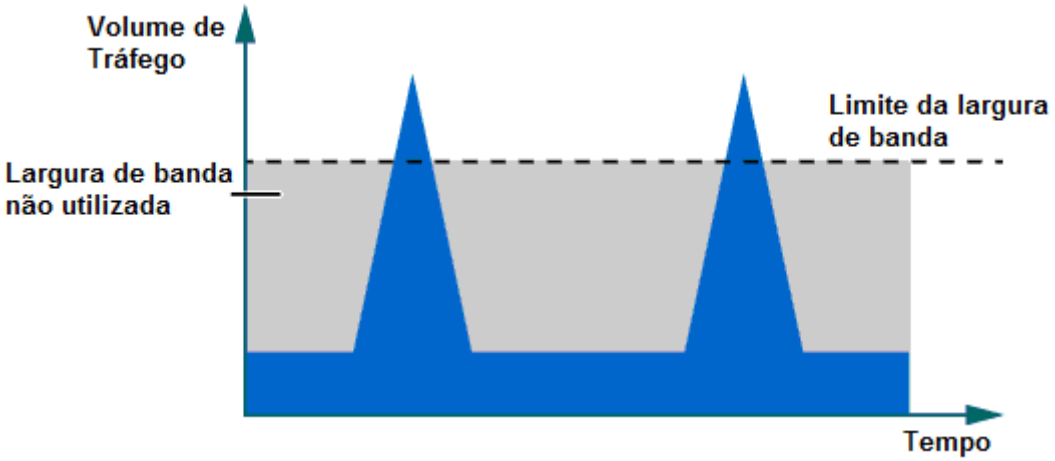

Figura 3 – Demonstrativo do tráfego em rajadas

Pacing é uma técnica que foi implementada inicialmente no TCP e existe também no QUIC. Seu objetivo principal é reduzir o tráfego em rajadas através de um espalhamento temporal dos dados, isto é, considerando a visão gráfica da Figura 3, os dados que antes encontravam-se concentrados nos momentos de pico serão distribuídos no tempo de forma a uniformizar a utilização da largura de banda. Redes cujos roteadores possuem *buffers* pequenos se utilizam desta técnica para reduzir a perda de pacotes, pois durante as rajadas os pacotes enfileiram-se no *buffer*, o qual pode ficar cheio e desprezar pacotes.

2.3. DASH

O DASH é um padrão de transmissão de vídeo que foi criado para substituir o anterior *download* progressivo. A principal diferença entre os dois é que um servidor que utiliza o padrão predecessor entrega a mesma qualidade de áudio e vídeo para todos os clientes independente da qualidade da rede de cada um, enquanto o sucessor codifica um mesmo conteúdo em várias qualidades diferentes e permite que cada cliente escolha a que melhor atende às condições de sua rede. A Figura 4 demonstra a diferença entre o DASH e o *download* progressivo.

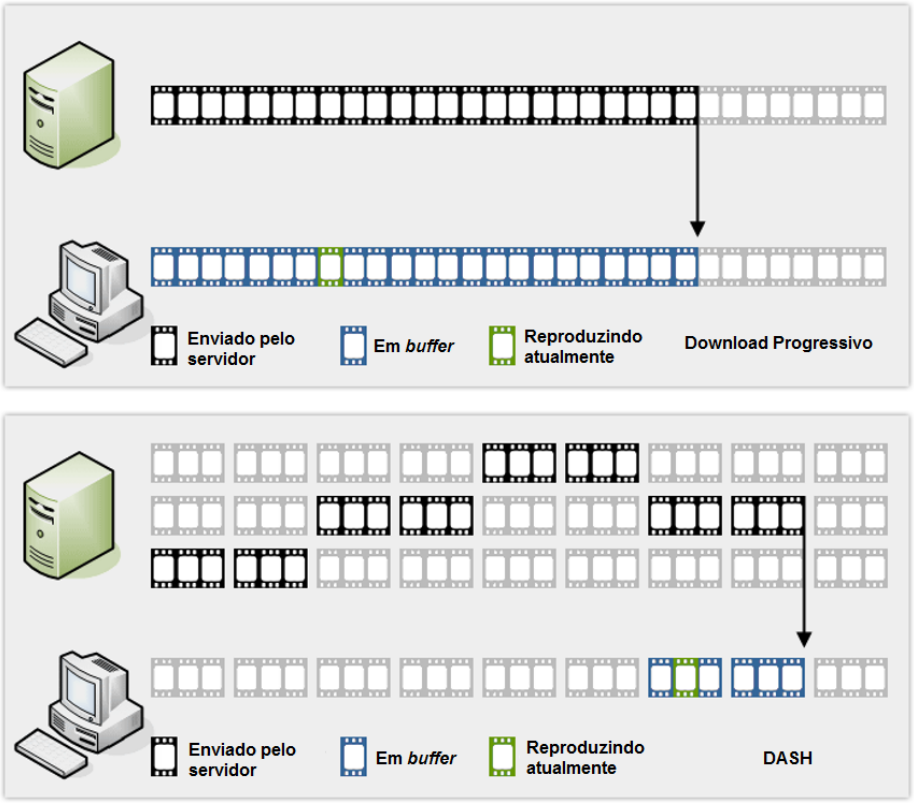

Figura 4 – Comparativo entre DASH e download progressivo

No DASH existe um arquivo XML que é essencial para seu funcionamento que se chama MPD (*Media Presentation Description*). Este arquivo contém todas as informações que o cliente precisa saber para usufruir do DASH. Algumas informações do MPD são:

- Qualidades disponíveis: Como foi dito anteriormente, o servidor codifica o vídeo em diversas qualidades. É no arquivo MPD que ele informa todas as opções disponíveis.
- URL para requisição: Cada qualidade é requisitada de um local diferente. No arquivo MPD é informada a URL para onde deve ser feita a requisição de cada qualidade disponível.
- Largura de banda para decidir a qualidade: O cliente realiza cálculos de largura de banda aproximada de sua conexão, os quais serão utilizados para escolher a qualidade desejada no arquivo MPD. Isto ocorre, pois, o servidor disponibiliza neste arquivo XML um valor estático de largura de banda que é necessária para consumir determinada qualidade. Então o cliente realiza o cálculo e a partir do valor obtido consulta o MPD e determina a qualidade desejada.

3. Estado da Arte

3.1. DASH e TCP

Um vídeo em DASH é codificado em várias qualidades diferentes no servidor, cada uma delas fragmentada em *chunks* (pequenos pedaços de vídeo, cada um com alguns segundos). O mecanismo de adaptação da qualidade, no lado do cliente, baseando-se nas condições locais da rede, escolhe qual a qualidade do próximo *chunk* que será solicitada ao servidor (SODAGAR, 2011). Um cliente DASH solicita um fragmento por vez via HTTP, realiza os cálculos para verificar a condição da rede e decide se será necessário alterar a qualidade para se adaptar. Como os *chunks* são pequenos, eles são solicitados um após o outro, criando no DASH um padrão de tráfego conhecido como "*start/stop*", o qual é bem diferente dos padrões de transmissão de vídeo com uma única solicitação (ESTEBAN, 2012).

Uma das características intrínsecas do TCP é ser otimizado para conexões de longa duração e com grande troca de dados, mantendo a rede cheia de pacotes. Devido ao padrão "*start/stop*" do DASH, não é possível conseguir uma máxima utilização da largura de banda quando se utiliza o TCP, o que ocorre principalmente devido à perda de pacotes durante a conexão (ESTEBAN, 2012). É importante ter em mente que os problemas acima enfrentados pelo DASH são uma consequência de se utilizar o TCP, o qual possui limitações intrínsecas ao seu funcionamento, dentre as quais podemos citar latência e HOL *blocking*, já detalhadas em seções anteriores e que fazem com que o TCP seja um impedimento à melhora do desempenho da *Web* como um todo.

3.2. QUIC e *PACING*

Com o objetivo de superar algumas das limitações do TCP, a Google propôs em 2012 o seu protocolo QUIC, o qual fornece confiabilidade sobre o UDP (Network Working Group, 2016), provê segurança na conexão ao utilizar criptografia por padrão e resolve o HOL *blocking* e o problema da latência do TCP.

Em estudos anteriores o desempenho do QUIC já foi comparado com o TCP no que se refere ao carregamento de páginas Web. Em (MEGYESI; KRÄMER; MOLNÁR, 2016) e (CARLUCCI; CICCO; MASCOLO, 2015) é possível verificar que:

- O QUIC se sai melhor do que o TCP em páginas com elevada quantidade de objetos pequenos. Este resultado positivo é obtido, pois, para páginas com objetos pequenos são feitas muitas conexões com o servidor, então graças ao seu RTT nulo o QUIC possui vantagem contra o TCP. O resultado é positivo especialmente quando há uma baixa largura de banda disponível e um alto RTT.
- O benefício do RTT nulo do QUIC vai desaparecendo à medida que os dados transmitidos são maiores, pois o maior peso na performance deixa de ser o RTT e se torna o tempo de transferência.
- O QUIC se sai pior que o TCP em cenários onde há perda de pacotes, principalmente se os dados transmitidos forem grandes. Isso ocorre, pois, o TCP utiliza várias conexões com controle de congestionamento separado para cada, enquanto o QUIC, apesar de utilizar multiplexação de fluxos, só utiliza uma única conexão. Portanto, no caso de uma perda o TCP só

sofrerá impacto na conexão que perdeu o pacote, enquanto o QUIC sofre em toda a sua transmissão já que só utiliza uma.

- Em cenários com largura de banda muito alta o QUIC se mostra pior que o TCP, o que é justificado principalmente pelo *Pacing* estar habilitado.

3.3. TCP e *PACING*

Uma característica do TCP é o envio de dados em rajadas, a qual era considerada como principal suspeita pela perda de pacotes e consequentemente pela subutilização da largura de banda. Os autores de (ESTEBAN, 2012), em uma tentativa de contornar o principal suspeito e então melhorar a utilização, implementaram o *Pacing* no TCP, o que resultou num cenário pior ou igual ao anterior. Embora a implementação da técnica tenha reduzido a perda de pacotes no momento inicial de rajada, ela a transferiu para o momento final da conexão, onde uma perda é ainda mais prejudicial por existirem poucos dados na rede.

O resultado obtido em (ESTEBAN, 2012) é justificado em (AGGARWAL; SAVAGE; ANDERSON, 2012), onde é explicado que em cenários de concorrência os fluxos com *Pacing* ativo atingem menor vazão do que os que não implementam a técnica. A justificativa é que devido ao *Pacing* fazer com que os pacotes sejam espalhados no tempo, a probabilidade de disputar espaço nos *buffers* dos roteadores com uma rajada concorrente é superior à de fluxos que não implementam a técnica, já que estes enviam rajadas que utilizam curtos períodos de tempo.

Todos os estudos citados que analisaram o QUIC utilizaram-no com o *Pacing* habilitado, e a partir das análises de (ESTEBAN, 2012) e (AGGARWAL; SAVAGE; ANDERSON, 2012) temos evidências que esta técnica causa uma redução de vazão. Em (MEGYESI; KRÄMER; MOLNÁR, 2016) vemos que o principal motivo do QUIC possuir um desempenho pior que o do TCP em cenários de largura de banda elevada é o fato do *Pacing* estar ativo.

Ao realizar uma busca pelo estado da arte sobre QUIC *Pacing* e DASH, encontramos apenas o estudo de (SZABÓ; RÁCZ; BEZZERA, 2016), o qual propõe um sistema que altera o algoritmo de controle de congestionamento para aumentar a agressividade do QUIC no carregamento inicial do vídeo. Porém não foi encontrado nenhum estudo que analise o QUIC sem o *Pacing*, nem tampouco um que compare o DASH ao utilizar este protocolo com a técnica desligada e ligada.

O presente trabalho se propõe a realizar uma análise comparativa que permita entender o comportamento do DASH ao utilizar o QUIC com *Pacing* ligado e desligado.

4.Metodologia

4.1. Visão geral:

Neste capítulo será detalhado o plano de experimentos, as métricas de interesse, os fatores e seus níveis de variação e as ferramentas utilizadas para emular a rede, fornecer e consumir conteúdo. O *testbed* é composto por dois computadores configurados para funcionar de acordo com a arquitetura cliente-servidor. O sistema operacional das máquinas é o Ubuntu 16.04 e ambas possuem processador com arquitetura de 64bits. Outro componente do *testbed* é o emulador de rede TC (*Traffic Control*) que é executado no cliente, a versão do *kernel* Linux utilizada foi a 4.4.0. Por fim temos do lado do cliente o Shaka Player (versão 1.5.2), o qual é executado dentro navegador Chrome (64 bits versão 51.0.2704.106).

A figura 5 exemplifica a configuração do *testbed*. O Shaka Player, player de vídeo que suporta conteúdo DASH, será executado no cliente, o qual fará requisições de *chunks* de um vídeo ao servidor remoto até encher seu *buffer* de 30 segundos de vídeo. Caso o *buffer* fique cheio, ele aguarda cerca de 4 segundos antes de solicitar novos *chunks* ao servidor. A troca de informações entre os dois computadores é realizada utilizando o protocolo QUIC, o qual utiliza o UDP como protocolo de transportes. O servidor utilizado foi o GoQUIC, o qual consegue fornecer conteúdo através de TCP ou QUIC. A modelagem do tráfego na rede é realizada na máquina cliente, utilizando a ferramenta TC, a qual foi utilizada para configurar os diversos cenários de rede necessários ao experimento.

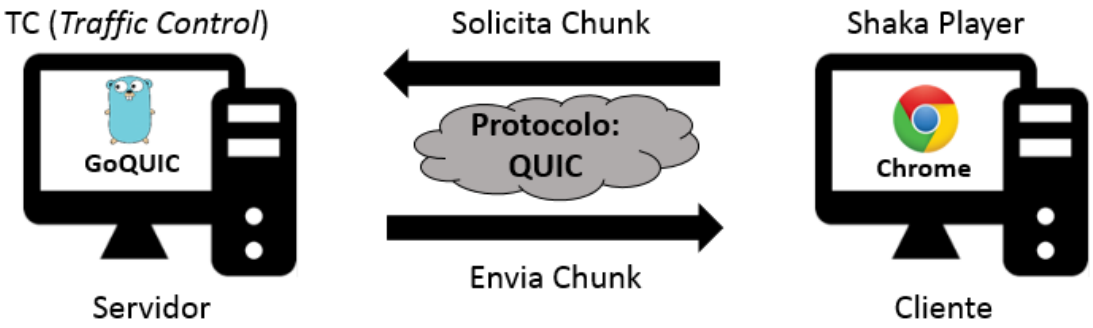

Figura 5 - Testbed

4.2. Métricas de interesse:

As métricas de interesse estão relacionadas ao estado da conexão e foram escolhidas pois permitem analisar como o *Pacing* do QUIC afeta a transmissão de vídeo em DASH.

São elas:

- **Porcentagem de largura de banda utilizada**: O TC será utilizado para definir um limite máximo conhecido para a largura de banda. O interesse nesta métrica é verificar como o *Pacing* impactou na capacidade de uso total da largura de banda, isto é, se causou uma limitação da capacidade ou se permitiu utilizar toda a banda disponível.
- ***Bytes in Flight***: Contém a quantidade de bytes que estão "voando" na rede, isto é, a quantidade de bytes que o servidor enviou, mas que ainda não recebeu o *ack* de confirmação. Métrica utilizada para analisar se a conexão

consegue colocar um maior volume de dados na rede com o *Pacing* ligado ou desligado. Além disso, é útil como apoio para entendimento das decisões do algoritmo de controle de congestionamento em casos de perda de pacotes.

- ***Round Trip Time***: Tempo entre a saída de um pacote e a chegada de seu *ack*. Será utilizado para explicar a causa de determinados valores obtidos na métrica de largura de banda.
- ***Jitter***: É a variação estatística do atraso na entrega de pacotes, isto é, é a diferença média entre os RTTs consecutivos. Esta métrica influencia diretamente no QoE. Quanto maior o Jitter, pior a qualidade experimentada pelo cliente. Será utilizado para verificar como o *Pacing* interfere na variação de atraso.

4.3. Fatores e níveis:

Para que as métricas sejam válidas em uma análise comparativa é necessário que as mesmas sejam extraídas de várias configurações diferentes da rede. Estas diferentes configurações foram criadas pela ferramenta TC, que foi utilizada para emular a rede ao modificar os fatores abaixo em diferentes níveis:

Tabela 2 – Fatores e níveis

Fatores	Níveis
Largura de banda máxima (Mbps)	2, 10, 50
Perda de pacotes (%)	0%, 1%, 2%
Atraso na rede (RTT)	20 ms, 200 ms
Pacing	Ligado / Desligado

4.4. Ferramenta TC e suas filas

TC é uma ferramenta que está presente no *kernel* dos sistemas LINUX e é responsável pelo controle de tráfego de todos os dados que chegam pelas interfaces de rede. Com ela é possível realizar a modelagem dos dados, programar a ordenação do tráfego, desprezar pacotes, entre outras funcionalidades.

Estas funcionalidades são alcançadas pela utilização de "disciplina de filas", mais conhecidas como QDISCs (*Queuing Disciplines*). Existem outros objetos importantes no TC, como classes e filtros, porém as QDISCs foram majoritárias neste trabalho.

Embora existam várias outras QDISCs disponíveis para uso no TC, as utilizadas foram:

- **FQ (*Fair Queue*)**: Dentre outras funcionalidades, permite ligar e determinar uma vazão máxima para o *pacing* do TCP. Foi utilizada para extrair as métricas do TCP com *pacing* ativo.
- **NetEM (*Network Emulator*)**: Possui várias funcionalidades de emulação de rede, porém esta QDISC foi utilizada apenas para definir o atraso na rede e também o percentual de perda de pacotes.
- **TBF (*Token Bucket Filter*)**: Esta QDISC foi utilizada para determinar a largura máxima de banda que pode ser utilizada pelo experimento.

4.5. GoQUIC e certificados:

Para fornecer conteúdo QUIC é obrigatório o uso de criptografia. Por este motivo foi necessário utilizar um certificado para o servidor GoQUIC. Tendo em vista que o intuito deste trabalho é analisar o comportamento da transmissão de dados entre dois

computadores conhecidos em uma rede também conhecida, não foi necessário comprar nenhum certificado de uma autoridade certificadora. Para atender a criptografia exigida pelo protocolo QUIC, o tutorial disponível em (NGUYEN, 2015) foi seguido e então todos os certificados necessários para que o servidor funcionasse devidamente com QUIC foram criados localmente.

O servidor GoQUIC utiliza uma biblioteca chamada *libquic*, a qual contém todas as funções do protocolo QUIC e que foi extraída do código do navegador Google Chromium. Foi o código desta biblioteca que foi alterado para coletar as métricas deste trabalho, as quais são coletadas no servidor no momento em que um pacote é dado como perdido ou quando recebe um *ack* do cliente. Para alterar o nível do fator "*Pacing*", ora para "Ligado", ora para "Desligado", foi necessário ajustar uma variável no código desta mesma biblioteca.

4.6. Plano de experimentos:

O plano de experimentos foi criado com o objetivo de permitir a avaliação da influência do Pacing do QUIC nos diferentes cenários de rede mostrados na Tabela 2. O vídeo utilizado nos experimentos é o "Big Buck Bunny" (Blender, 2008), o qual está disponível para download gratuito. A versão do vídeo utilizada para os experimentos não possui áudio e foi disponibilizada pelo servidor nas qualidades (320x240), (480x360), (854x480), (1280x720) e (1920x1080). Enquanto o vídeo era transmitido as métricas eram extraídas no servidor. Após coletar as métricas de todos os cenários, o impacto do *Pacing* do QUIC na transmissão do conteúdo DASH foi avaliado.

5.Resultados e discussão

Neste capítulo iremos mostrar os resultados obtidos ao se executar o plano de experimentos e discutiremos o que significa cada um.

O plano de experimentos foi executado para o QUIC e para o TCP, tanto com *Pacing* ligado como desligado, formando assim 36 cenários diferentes para cada protocolo. Para formar os diferentes cenários utilizamos os fatores e níveis da tabela 2. Os cenários estão expostos na tabela 3. Cenários de números ímpares possuem *Pacing* ligado, enquanto os pares estão com a técnica desligada.

Tabela 3 - Cenários criados a partir dos fatores e níveis

Cenários	Largura de Banda Máxima (Mbps)	Perda de pacotes (%)	Atraso na rede (ms)	Pacing
Cenário 1	2	0	20	Ligado
Cenário 2	2	0	20	Desligado
Cenário 3	2	0	200	Ligado
Cenário 4	2	0	200	Desligado
Cenário 5	2	1	20	Ligado
Cenário 6	2	1	20	Desligado
Cenário 7	2	1	200	Ligado
Cenário 8	2	1	200	Desligado
Cenário 9	2	2	20	Ligado
Cenário 10	2	2	20	Desligado
Cenário 11	2	2	200	Ligado
Cenário 12	2	2	200	Desligado
Cenário 13	10	0	20	Ligado
Cenário 14	10	0	20	Desligado
Cenário 15	10	0	200	Ligado
Cenário 16	10	0	200	Desligado
Cenário 17	10	1	20	Ligado
Cenário 18	10	1	20	Desligado
Cenário 19	10	1	200	Ligado
Cenário 20	10	1	200	Desligado
Cenário 21	10	2	20	Ligado
Cenário 22	10	2	20	Desligado
Cenário 23	10	2	200	Ligado
Cenário 24	10	2	200	Desligado
Cenário 25	50	0	20	Ligado
Cenário 26	50	0	20	Desligado
Cenário 27	50	0	200	Ligado
Cenário 28	50	0	200	Desligado
Cenário 29	50	1	20	Ligado
Cenário 30	50	1	20	Desligado
Cenário 31	50	1	200	Ligado
Cenário 32	50	1	200	Desligado

Cenário 33	50	2	20	Ligado
Cenário 34	50	2	20	Desligado
Cenário 35	50	2	200	Ligado
Cenário 36	50	2	200	Desligado

5.1. QUIC

O desenvolvimento do código QUIC foi realizado utilizando de forma padrão o *Pacing* ligado, sendo otimizado a trabalhar desta maneira. Dentro do código fonte do protocolo existe um *flag* que permite desligar o *Pacing*, porém os desenvolvedores deixam claro que é apenas para fins de teste.

As imagens e comentários a seguir mostram os resultados obtidos para o DASH ao utilizar o QUIC, o qual foi executado com *Pacing* ligado e também desligado para testarmos o comportamento do protocolo sem esta funcionalidade.

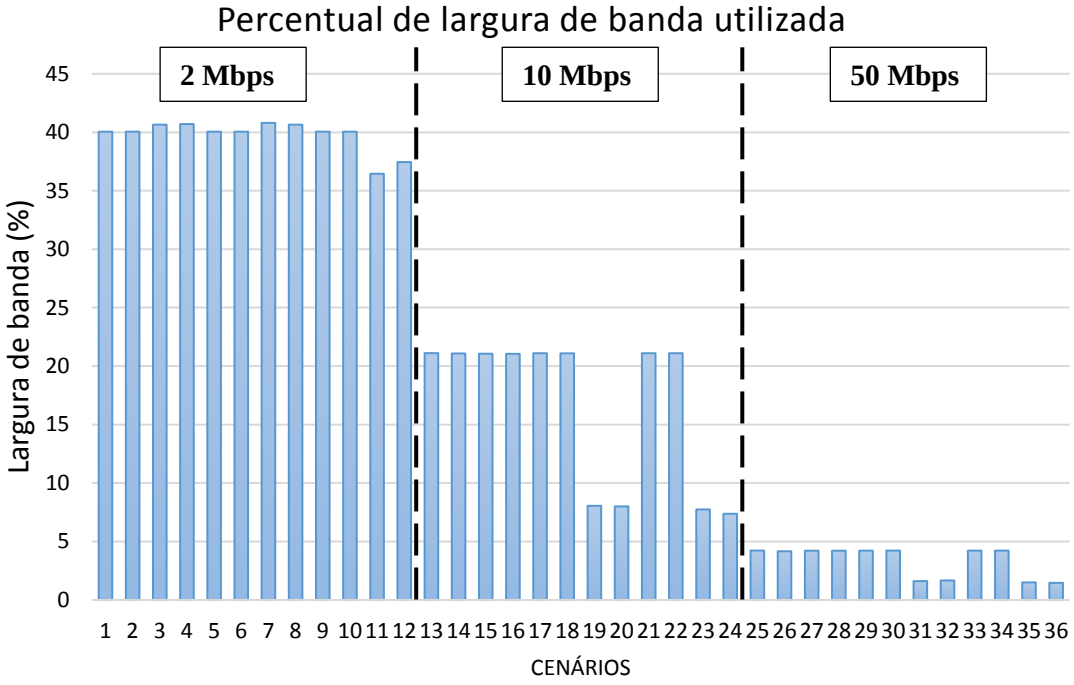

Figura 6 - Porcentagem de Largura de Banda utilizada pelo QUIC

A partir da análise da figura 6 é possível perceber que o *Pacing* do QUIC não causou grande influência na porcentagem de largura de banda utilizada durante os experimentos, pois ao comparar os pares (ligado e desligado) é possível perceber que quando há uma divergência, esta fica aproximadamente em 1%.

A partir da análise da figura 6 é possível identificar que, nas condições utilizadas neste trabalho, à medida que a largura de banda vai ficando maior, seu uso pelo QUIC vai sendo reduzido. Isso fica claro ao verificar os últimos 12 cenários, referentes ao cenário de 50Mbps, onde a porcentagem de uso não chegou a 5%. Apesar desta redução percentual que citamos, não significa que o QUIC utiliza menos banda à medida que ela aumenta. Isto fica mais claro ao verificarmos o uso real da largura de banda, o qual está demonstrado na figura 7.

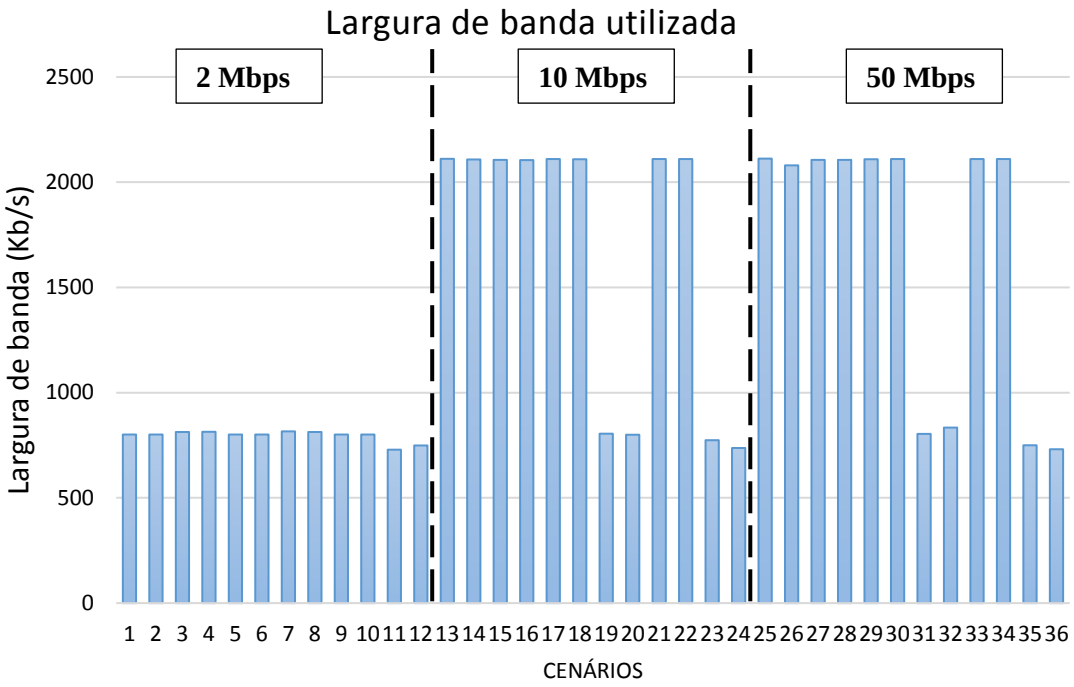

Figura 7 - Largura de Banda utilizada pelo QUIC em Kb/s

Na figura 7 é possível ver com mais detalhes que apesar da redução percentual citada anteriormente, ainda assim o QUIC utiliza uma banda maior quando ela está disponível.

Ainda na figura 7 é possível perceber que os cenários de 10 e 50 Mbps chegaram a uma largura máxima de banda de aproximadamente 2100Kb/s, até mesmo nos cenários sem perda de pacotes. Esta peculiaridade ocorre, pois, o cliente consumidor do conteúdo possuía um *buffer* máximo de 30 segundos, aguardando cerca de 4 segundos antes de voltar a consumir e encher mais uma vez seu *buffer*. Estes segundos nos quais a conexão fica ociosa influenciam na média. Como nos cenários sem perda, 10 e 50 Mbps são suficientes para consumir a melhor qualidade, ambos consomem a mesma qualidade do vídeo e então os dois ficam limitados pelo tempo de vídeo e chegam na mesma média de largura de banda.

Com o objetivo de obter uma visão mais realista da banda atingida por cada cenário, foi extraída a largura de banda obtida durante a transmissão de um único *chunk* de vídeo por volta de 70% do filme, onde já pode ter ocorrido perdas e a rede já está adaptada às condições do cenário. Esta visão está detalhada na figura 8.

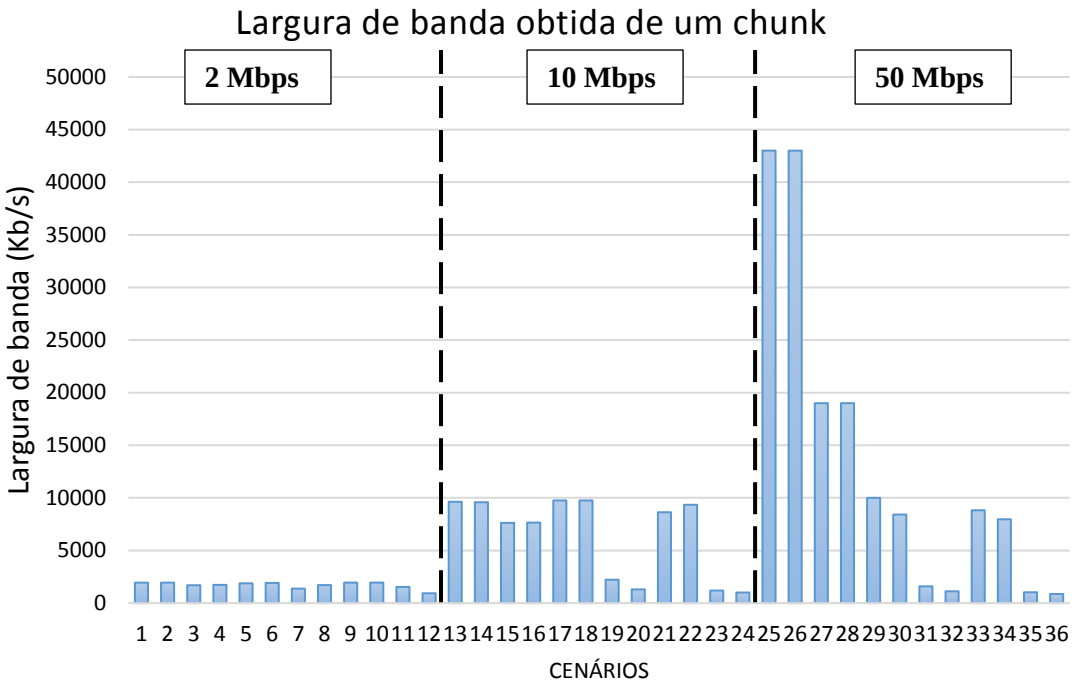

Figura 8 – Largura de banda obtida de um *chunk* no QUIC em Kb/s

A partir da figura 8 é possível observar a banda real utilizada pelo QUIC na transmissão de um *chunk* de forma ininterrupta e sem momentos onde a transmissão fica ociosa. Através da análise deste gráfico conseguimos perceber que à medida que as perdas vão aumentando, a largura de banda resultante vai se aproximando dos resultados obtidos nos cenários de 2Mbps mesmo quando há 50Mbps disponíveis. Esta característica é um indicativo que ele é um protocolo justo, pois se retrai consideravelmente quando percebe sinais de sobrecarga na rede.

Com relação à comparação dos cenários com *Pacing* ligado e desligado, é possível verificar que os únicos cenários onde a técnica desligada teve um efeito positivo foram nas comparações entre os 7/8 e 21/22. Nos demais cenários o desligamento trouxe resultados iguais ou inferiores aos de quando a técnica está ligada.

Nas figuras seguintes iremos verificar outras variáveis de rede para entendermos melhor como o QUIC se comportou nos diferentes cenários:

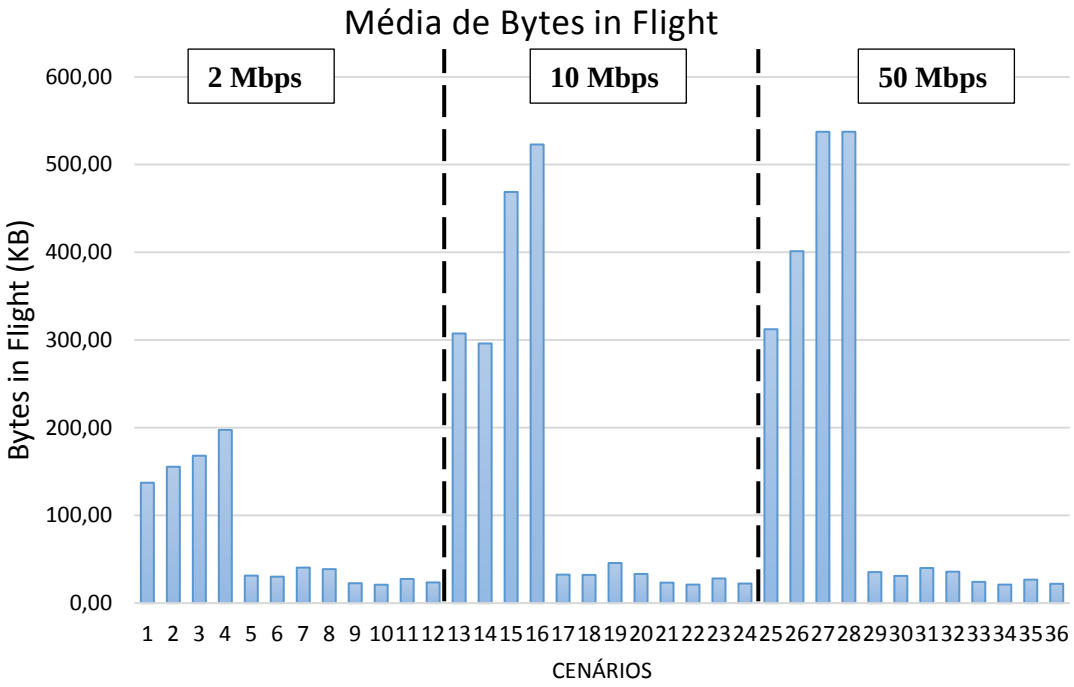

Figura 9 - Média de *Bytes in Flight* do QUIC

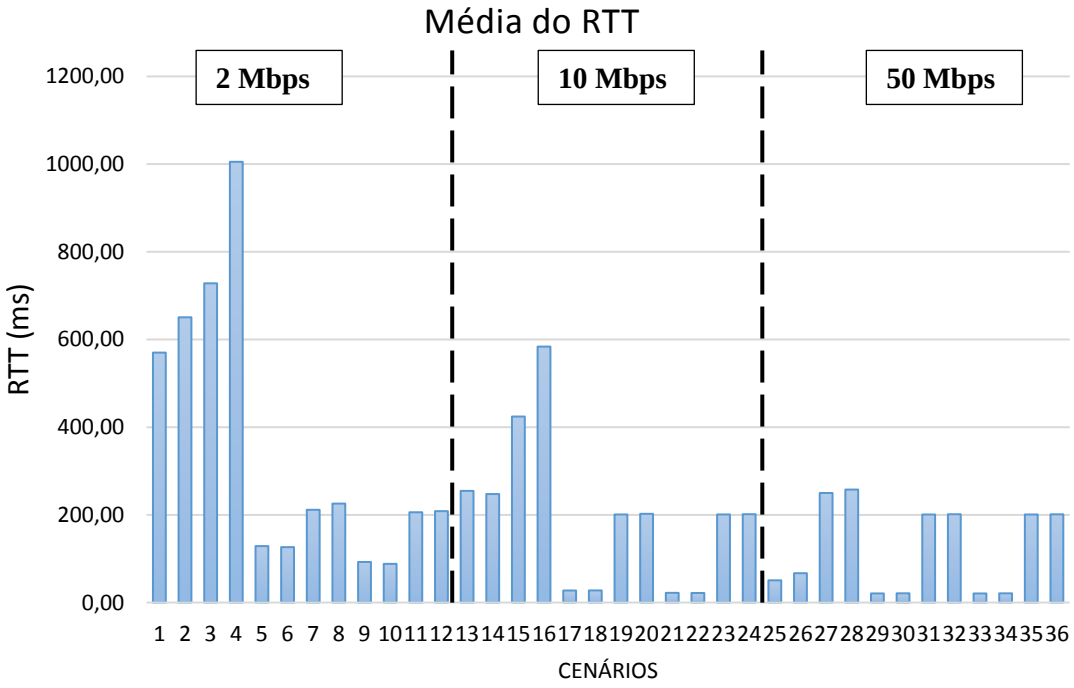

Figura 10 - Média do RTT do QUIC em ms

Na figura 9 pode-se perceber o impacto que a perda de pacotes causou nos *Bytes in Flight* do QUIC, pois é visível a redução significativa dos cenários sem perda para os com perda induzida.

O RTT médio pode ser observado na figura 10 e ao realizar uma análise é possível verificar que, com exceção do cenário 14, todos os cenários sem perda de pacotes

possuem um RTT maior para o caso onde o *Pacing* está desligado. Este resultado está condizente com estudos anteriores, conforme citado em (AGGARWAL; SAVAGE; ANDERSON, 2012), pois fluxos em rajadas causam um maior atraso dos pacotes nas filas e também podem causar uma menor largura de banda.

Para os casos onde há perda de pacotes o RTT se mantém muito próximo com ou sem *Pacing*, porém ao comparar estes cenários na figura 10 com os mesmos cenários na figura 7 é possível identificar que apesar do empate no RTT, os que possuem *Pacing* ativo atingiram uma largura de banda ligeiramente melhor.

Para garantir que não houve impacto na qualidade de experiência para o usuário, decidimos extrair o *Jitter*, pois um valor acima de 150ms nesta métrica começa a degradar a experiência de serviços multimídia. O *Jitter* médio para cada cenário encontra-se na figura 11 e mostrou-se dentro de um valor aceitável para serviços multimídia, pois ficou abaixo dos 14ms para todos os cenários.

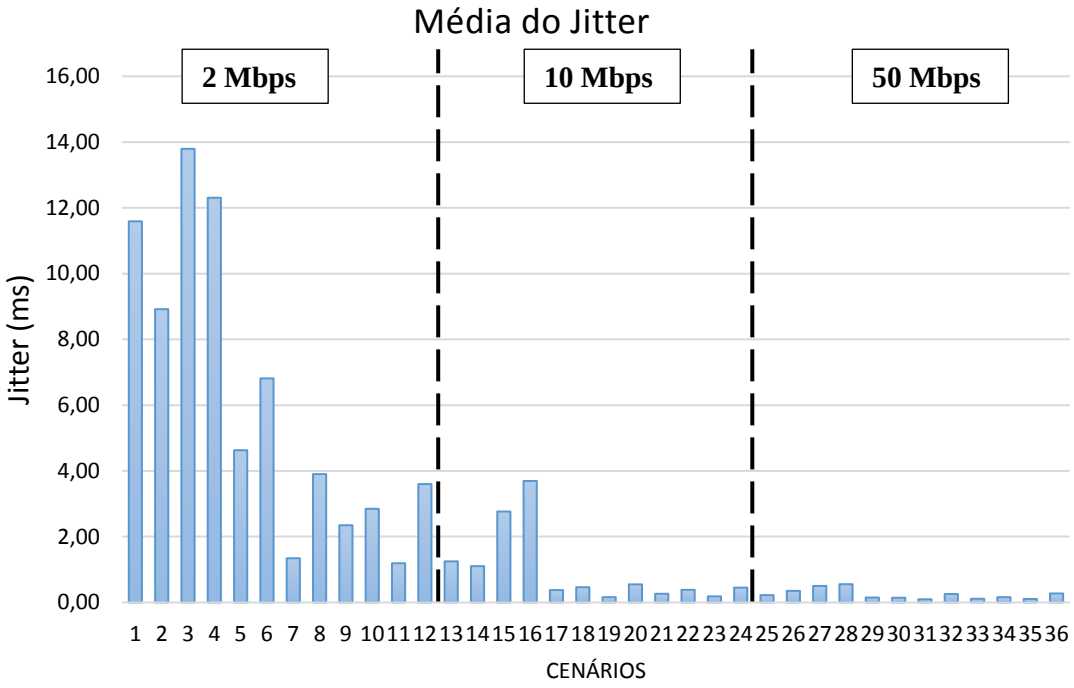

Figura 11 - Média do *Jitter* do QUIC em ms

5.2. TCP

Executamos os testes no TCP assim como no QUIC, utilizando os mesmos cenários. Os resultados a seguir mostram como o DASH se comportou ao utilizar o TCP tanto com *Pacing* ligado como desligado.

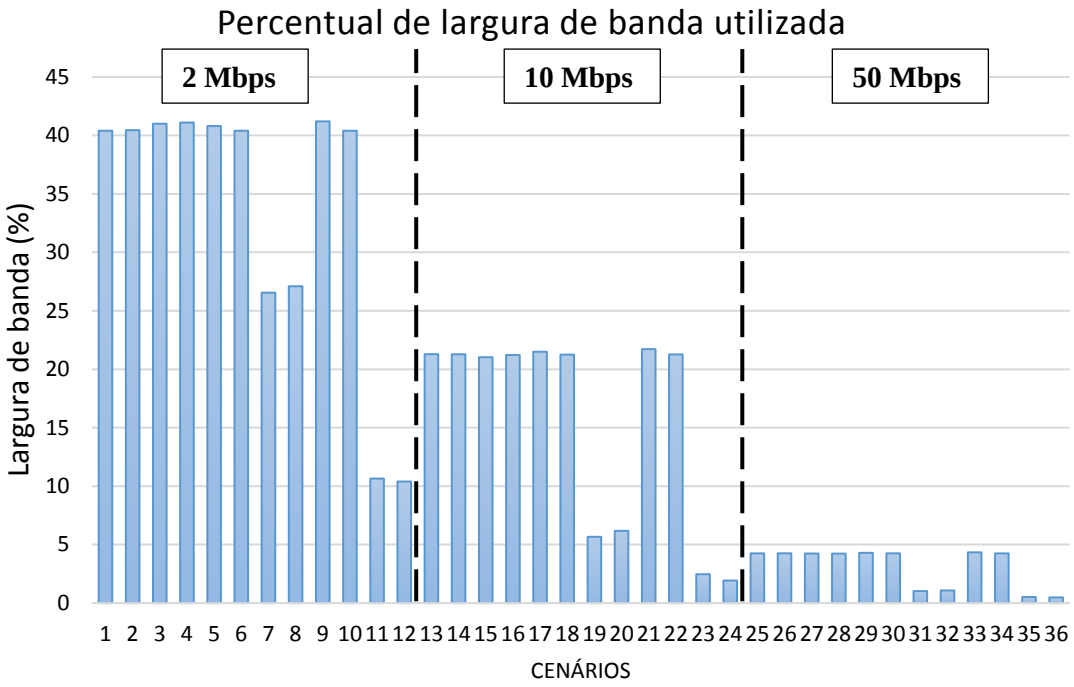

Figura 12 - Porcentagem de Largura de Banda utilizada pelo TCP

A partir da análise da figura 12 é possível perceber que o *Pacing* do TCP, assim como aconteceu com o QUIC, não causou grande influência na porcentagem de largura de banda utilizada durante os experimentos, pois ao comparar os pares (técnica ligada e desligada) é possível perceber as divergências são mínimas, não chegando a mais que 1%.

Ainda analisando a figura 12 é possível verificar que nos cenários de perda, quando o atraso da rede está configurado para 200ms a largura de banda sofre uma queda considerável, o que não é observado para atrasos de 20ms. Isso mostra que em redes com perda de pacotes e com pequeno atraso o TCP é resiliente, porém sofre grave impacto em casos onde o atraso é considerável.

Assim como no QUIC, é possível visualizar que o TCP também possui uma redução percentual à medida que a largura de banda vai aumentando, porém isso não significa que este protocolo utilize menos banda no cenário de 50Mbps que no de 2Mbps. Isto é melhor visualizado na figura 13, onde é possível observar a largura de banda real utilizada.

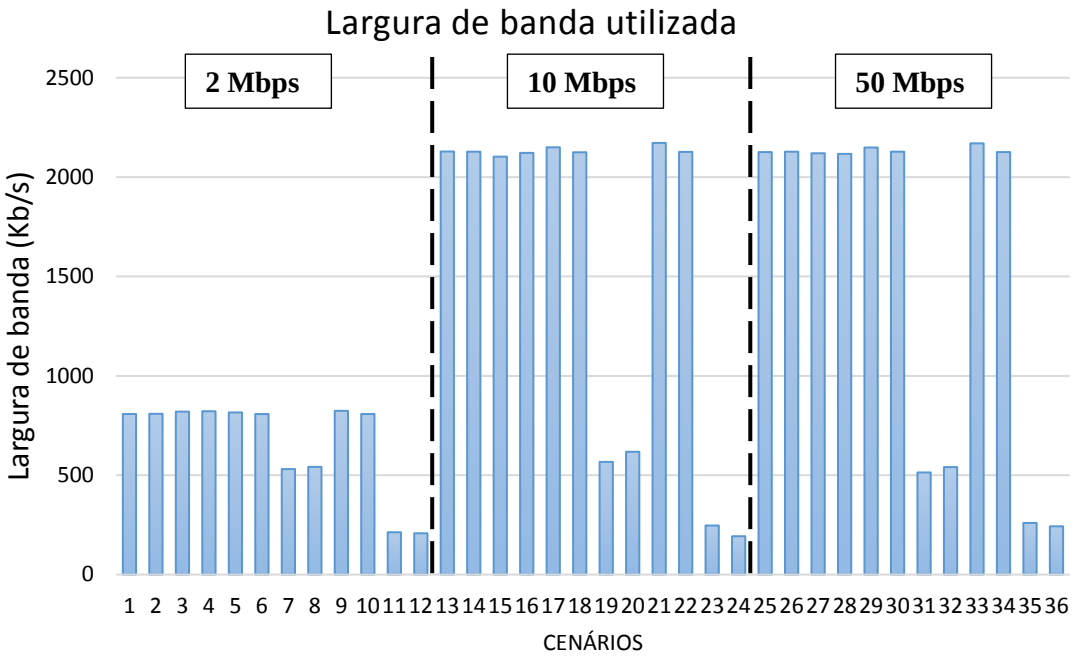

Figura 13 - Largura de Banda utilizada pelo TCP em Kb/s

Na figura 13 é possível ver com mais detalhes que apesar da redução percentual observada na figura 12, o TCP utiliza uma banda maior quando ela está disponível.

Também na figura 13 é possível identificar a mesma peculiaridade que ocorre no QUIC, isto é, larguras de banda de 10 e 50 Mbps disponíveis e o TCP só conseguiu obter uma largura média máxima de aproximadamente 2100Kb/s. A causa deste efeito é exatamente a mesma do QUIC, pois o cliente utilizado foi o mesmo.

A figura 14 mostra a largura de banda medida durante a transmissão de um único *chunk* de vídeo por volta de 70% do filme, com o objetivo de mostrar a largura de banda real obtida quando os segundos sem tráfego não são considerados.

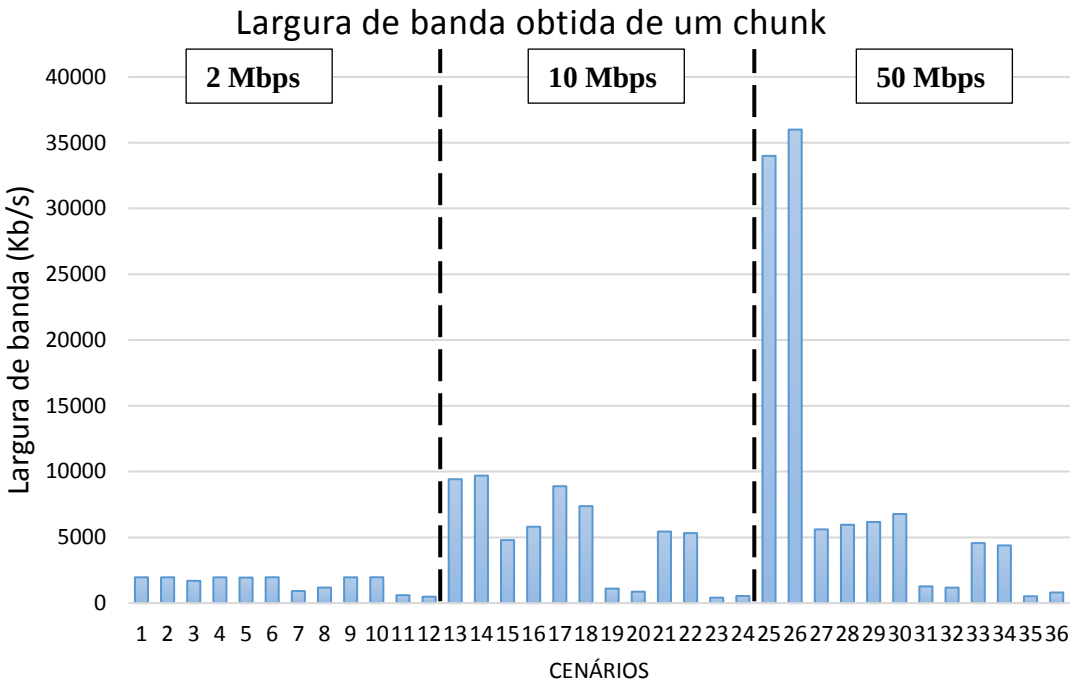

Figura 14 - Largura de banda obtida de um *chunk* no TCP em Kb/s

É importante ressaltar que para melhorar a visualização dos dados na figura 14, o eixo vertical só exibe valores até o limite de 40000 Kb/s, diferente dos 50000 Kb/s exibidos para o QUIC na figura 8.

Conforme mostrado em (MEGYESI; KRÄMER; MOLNÁR, 2016), o QUIC se mostra um protocolo de melhor desempenho quando comparado ao TCP em cenários com alto RTT. Esta mesma afirmação pode ser notada em uma comparação dos cenários 27 e 28 das figuras 8 e 14, onde o TCP teve uma redução de largura de banda considerável, ficando com cerca de 31% do valor obtido pelo QUIC.

Nas figuras seguintes iremos verificar outras variáveis de rede para entendermos melhor como o TCP se comportou nos diferentes cenários:

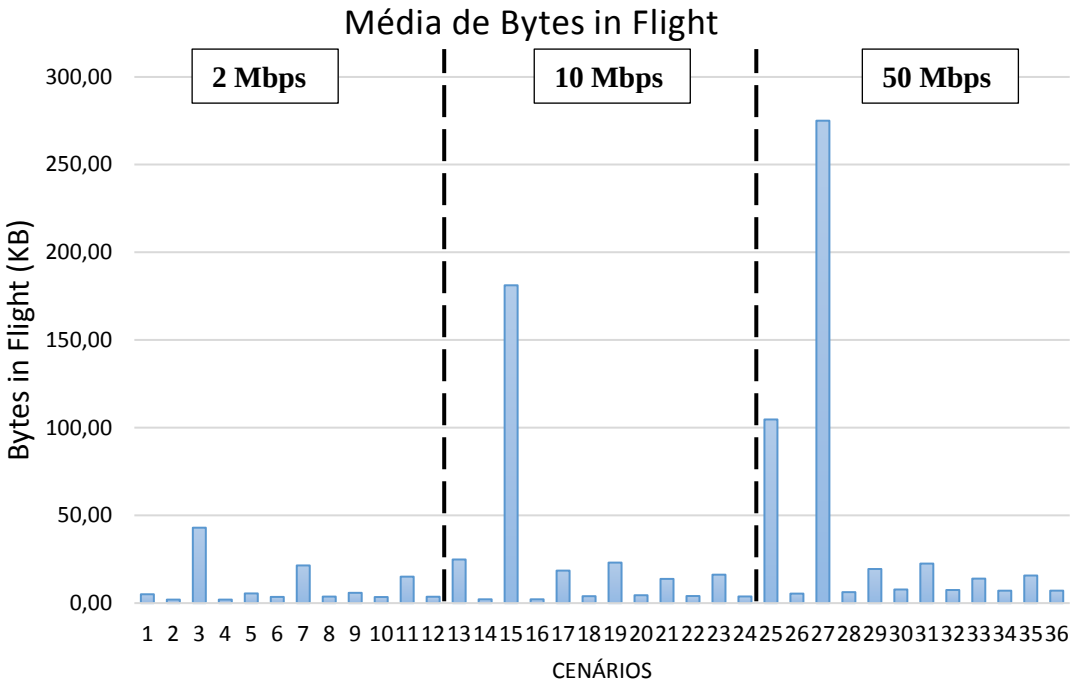

Figura 15 - Média de *Bytes in Flight* do TCP

É importante ressaltar que para melhorar a visualização dos dados na figura 15, o eixo vertical só exibe valores até o limite de 300 KB, diferente dos 600 KB exibidos para o QUIC na figura 9. O mesmo se aplica para a figura 16, que exibe um limite de 250 ms, diferente dos 1200 ms exibidos para o QUIC na figura 10.

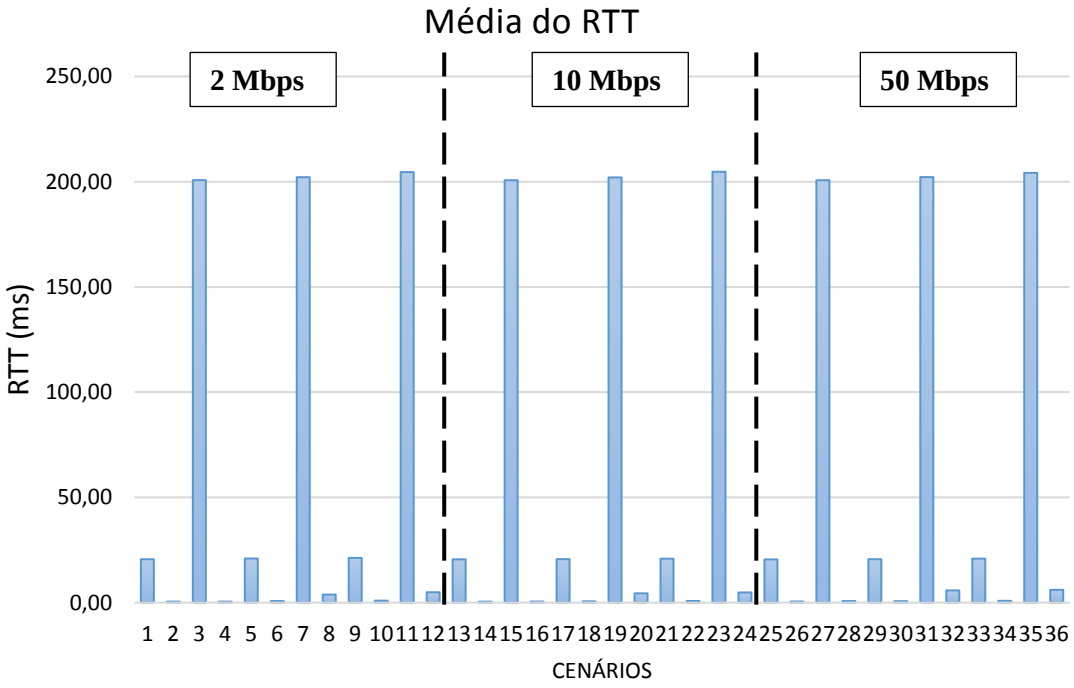

Figura 16 - Média do RTT do TCP em ms

Ao analisar a figura 15 é possível observar que as maiores taxas de *Bytes in Flight* são atingidas quando os atrasos são de 200ms. Isso é justificável, pois com o *ack* demorando a chegar, mais pacotes são enviados à rede, aumentando assim o valor dos *Bytes in Flight*.

Devido aos valores do RTT da figura 16 se manterem tão constantes, é possível realizar uma comparação entre as figuras 14, 15 e 16 e perceber a relação entre *Bytes in Flight*, RTT e largura de banda. Devido ao RTT ser o mesmo para os cenários 3, 15 e 27 e os *Bytes in Flight* serem respectivamente maiores, a largura de banda segue a mesma lógica, isto é, é maior onde os *Bytes in Flight* tiverem um valor maior.

Assim como foi feito no QUIC, também extraímos o *Jitter* no TCP para garantirmos que a qualidade não foi deteriorada. O resultado pode ser visto na figura 17, o qual também ficou dentro dos valores aceitáveis para serviços multimídia, pois ficou abaixo de 10ms para todos os cenários.

É importante ressaltar que para melhorar a visualização dos dados na figura 17, o eixo vertical só exibe valores até o limite de 10 ms, diferente dos 16 ms exibidos para o QUIC na figura 11.

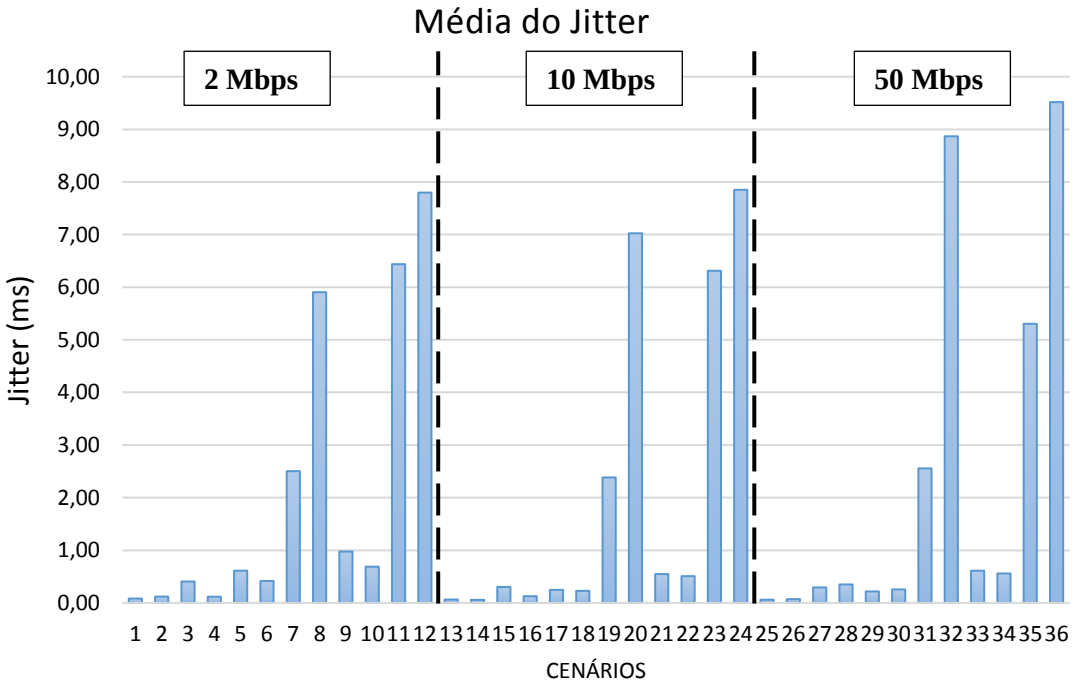

Figura 17 - Média do *Jitter* do TCP em ms

5.3. QUIC vs TCP

A figura 18 mostra uma comparação entre as figuras 8 e 14. Nela é possível observar o QUIC se saiu melhor que o TCP no quesito largura de banda utilizada em quase todos os cenários, especialmente quando a rede possui uma largura de banda disponível igual ou maior que 10Mbps. A diferença se torna maior quando o RTT é alto e não há perdas. Nos cenários com largura de banda disponível de 2Mbps é possível ver que a figura reforça a afirmação feita por (MEGYESI; KRÄMER; MOLNÁR, 2016), isto é, que o QUIC tem um melhor desempenho que o TCP em cenários de alto RTT e pequena largura de banda.

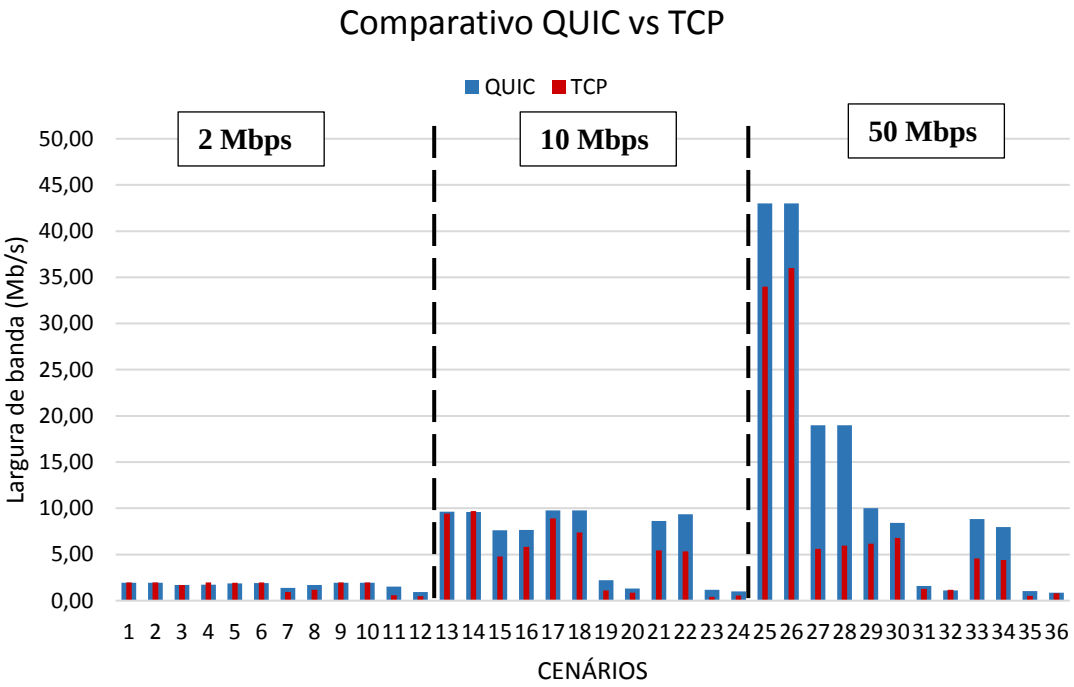

Figura 18 - Comparativo entre QUIC e TCP

O RTT e o J*itter* médio do TCP ficaram consideravelmente menor que os do QUIC. Apesar das grandes vantagens nessas duas últimas variáveis, o QUIC se saiu melhor no quesito QoE, pois conseguiu atingir uma largura de banda maior e consequentemente conseguiu oferecer ao usuário uma melhor qualidade do vídeos do que o TCP. Além disso, apesar do TCP ter alcançado menores valores de *Jitter* que o QUIC, isso não chega a influenciar a qualidade de experiência do usuário, pois o player possui um *buffer* que neutraliza os impactos do *Jitter* mais alto do QUIC.

Portanto pode-se dizer que, considerando especificamente as condições utilizadas neste trabalho para os experimentos realizados, o QUIC obteve vantagem sobre o TCP por ter conseguido entregar uma melhor QoE ao usuário.

Porém ao comparar a técnica de *Pacing* dentro de cada protocolo separadamente, de forma a analisar seu impacto, a mesma se mostrou inconclusiva no que se refere ao QUIC, pois em alguns cenários ela se mostra melhor, mas em outros semelhantes ela não consegue obter o mesmo resultado. Já no TCP, o desempenho observado ao desligar a técnica se mostrou igual ou superior nos cenários onde não há perdas. Devido ao *Pacing* limitar a quantidade de dados que pode ser enviada por vez, à medida que a largura de banda vai aumentando o desligamento se torna mais eficaz, pois pode usufruir de toda a largura disponível.

6.Conclusão

O presente trabalho analisou a influência do *Pacing* do QUIC em transmissões de conteúdo utilizando DASH e também foi realizada uma comparação entre este protocolo e o TCP.

A comparação entre os protocolos foi idealizada após verificar que em (ESTEBAN, 2012) o DASH não consegue utilizar toda a largura de banda disponível ao utilizar o TCP. Após realizar a análise comparativa, observou-se que embora o TCP tenha obtido menores valores de RTT e *Jitter*, isto não foi suficiente para ter uma melhora na qualidade, pois o *buffer* do *player* de vídeo no cliente elimina os impactos do *Jitter*. Considerando especificamente as condições utilizadas neste trabalho para os experimentos realizados, quem conseguiu fornecer a melhor QoE para o cliente foi o QUIC, pois foi capaz de atingir valores maiores ou iguais de largura de banda, o que permitiu que entregasse a mesma qualidade ou superior ao que o TCP conseguiu.

No que se refere à influência do *Pacing* destes dois protocolos na transmissão DASH, observou-se que:

- A influência do *Pacing* do QUIC se mostrou inconclusiva nas condições específicas deste trabalho, pois apesar da técnica desligada conseguir obter uma melhor largura de banda em alguns cenários, a vantagem não se manteve em cenários semelhantes. Consideramos o resultado inconclusivo devido aos desenvolvedores do QUIC deixarem claro que o *Pacing* só deve ser desabilitado para fins de teste, isto é, não é para ser usado normalmente. Sendo assim não é possível concluir se os resultados com *Pacing* desligado se mostraram iguais ou piores devido ao *design* protocolo QUIC de fato possuir um desempenho inferior com o *Pacing* desligado, ou se o desempenho observado é apenas um reflexo de falta de esforço em otimizações no código quando a técnica está desligada.
- Já com relação ao *Pacing* do TCP foi possível perceber que na maioria dos cenários o desligamento da técnica se mostrou mais eficaz, principalmente à medida que a largura de banda vai aumentando.

Como trabalhos futuros pretende-se:

- Realizar uma análise semelhante, porém introduzindo fluxos TCP e QUIC para concorrerem pela largura de banda disponível, com combinações de *Pacing* ligado e desligado para verificar quem é mais justo e quem sofre mais impacto com a concorrência.
- Comparar mais uma vez o TCP e o QUIC, desta vez utilizando redes de dados móveis como 3G e também Wi-Fi, com o objetivo de verificar a característica do QUIC ao trocar de IP sem precisar refazer a conexão e se esta traz grandes vantagens sobre o TCP.

Referências

SANDVINE Intelligent Broadband Networks. 2016 Global Internet Phenomena: LATIN AMERICA & NORTH AMERICA. Disponível em https://www.sandvine.com/downloads/general/global-internet-phenomena/2016/global-internet-phenomena-report-latin-america-and-north-america.pdf. Acesso em: 31 de jan. 2017

SODAGAR, I. The MPEGDASH Standard for Multimedia Streaming Over the Internet. MultiMedia, IEEE, [S.l.], v.18, n.4, p.62–67, April 2011.
Network Working Group. RFC 2616: Hypertext Transfer Protocol -- HTTP/1.1. Disponível em https://tools.ietf.org/html/rfc2616. Acesso em: 31 de jan. 2017.

MEGYESI, P.; KRÄMER, Z.; MOLNÁR, S. How quick is QUIC? In: Communications (ICC), 2016 IEEE International Conference on, 2016. Kuala Lumpur. Anais… Kuala Lumpur, 2016.

Network Working Group. QUIC: A UDP-Based Secure and Reliable Transport for HTTP/2. Disponível em https://tools.ietf.org/html/draft-hamilton-early-deployment-quic-00. Acesso em: 31 de jan. 2017.

ESTEBAN, J. et al. Interactions Between HTTP Adaptive Streaming and TCP. In: International Workshop on Network and Operating System Support for Digital Audio and Video, 22, 2012, Toronto. Proceedings… Toronto: ACM, 2012. p. 21-26.

AGGARWAL, A.; SAVAGE, S.; ANDERSON, T. Understanding the Performance of TCP Pacing. In: Annual Joint Conference of the IEEE Computer and Communications Societies, 19, 2000, Tel Aviv. Proceedings… Tel Aviv: IEEE INFOCOM, 2012. p. 1157-1165.

CARLUCCI, G.; CICCO, L. De; MASCOLO, S. HTTP over UDP: an Experimental Investigation of QUIC. In: Annual ACM Symposium on Applied Computing, 30, 2015, Salamanca. Proceedings… Salamanca: ACM SAC, 2015. p. 609-614.

Nguyen, J. OpenSSL Certificate Authority. Disponível em https://jamielinux.com/docs/openssl-certificate-authority/. Acesso em: 31 de jan. 2017.
Blender. Big Buck Bunny. Disponível em: https://peach.blender.org/about/. Acesso em: 31 de jan. 2017.

RIISER, H. et al. A Comparison of Quality Scheduling in Commercial Adaptive HTTP Streaming Solutions on a 3G Network. In: WORKSHOP ON MOBILE VIDEO, 4., New York. Proceedings… New York: ACM MoVid, 2012. p.25–30.

GRIGORIK, I. Making the Web Faster with HTTP 2.0. *Communications of the ACM*, New York, v. 56, n. 12, p. 42-49, dez. 2013.

SZABÓ, G.; RÁCZ, S.; BEZZERA, D. Media QoE Enhancement With QUIC. In: IEEE Conference on Computer Communications Workshops, 2016, San Francisco. Anais… San Francisco: IEEE INFOCOM WORKSHOPS, 2016.