Métricas de Detecção de Gargalos Compartilhados em Transmissões por Múltiplos Caminho em MPQUIC

Thomas Paiva[1], **Simone Ferlin**[2], **Bruno Kimura**[1]

[1]Universidade Federal de São Paulo (UNIFESP), Brasil.

[2]Ericsson AB, R&D Networks, Sweden.

Resumo. *Este artigo descreve a implementação de um método SBD (Shared Bottleneck Detection) para detecção de compartilhamento de gargalos em transmissões por múltiplos caminhos junto ao protocolo MPQUIC (MultiPath Quick UDP Internet Connection). Transmissões MPQUIC são realizadas por múltiplos subfluxos UDP, os quais percorrem diferente caminhos fim-a-fim entre os nós de origem e destino. Um método SBD permite que os sistemas finais quantifiquem métricas estatísticas sobre qualidade dos caminhos em uso pelos subfluxos e, com isso, infiram se os gargalos desses caminhos estão ou não sendo compartilhados entre esses subfluxos. A investigação vem sendo realizada de forma experimental, através da implementação e validação de um mecanismo SBD baseado no método previsto na RFC 8382 junto ao protocolo MPQUIC em Linguagem Go.*

Abstract. *This paper describes the implementation of a Shared Bottleneck Detection method (SBD) for multipath transmissions in MPQUIC (MultiPath Quick UDP Internet Connection). MPQUIC transmissions are accomplished through multiple UDP subflows that take different end-to-end paths between source and destination nodes. An SBD method allows the end-systems to measure statistical metrics on the quality of the paths in use by the subflows and, thus, infer whether the bottlenecks of those paths are being shared between the subflows or not. The investigation has been carried out experimentally, through the implementation and validation of an SBD mechanism based on the method described in RFC 8382 along with MPQUIC in Go Language.*

1. Introdução

Para acomodar o crescimento do tráfego na Internet ao longo dos anos, inovação tecnológica tem ocorrido nos enlaces de dados das redes de acesso. Ao longo dos anos, pôde-se observar a evolução dos padrões da família IEEE 802.3 (Ethernet) e IEEE 802.11 (WiFi), a chegada de FTTH (*Fiber-to-the-Home*) às redes domésticas, das redes celulares ao atual 4G e as futuras redes 5G com implantação próxima. Essas tecnologias buscam melhorar qualidade de serviço, essencialmente, elevando as capacidades dos enlaces de dados nas redes de acesso.

Na pilha de protocolos TCP/IP, pode-se observar inovação nas camadas de transporte e, principalmente, na camada de aplicação. No transporte de dados na Internet, vem ocorrendo a evolução do protocolo TCP para o *MultiPath* TCP (MPTCP) [Ford et al. 2011], de modo a possibilitar transmissões por múltiplos caminhos na Internet, explorando a diversidade de redes de acesso disponíveis nos sistemas finais

multihomed. Ao agregar capacidades dos caminhos fim-a-fim disponíveis, o protocolo MPTCP possibilita maior taxa de transmissão e tolerância a falhas de conexões entre os sistemas finais envolvidos [Ford et al. 2013].

Na camada de aplicação, uma das principais inovações dos últimos anos tem sido através das conexões rápidas UDP com o protocolo QUIC (*Quick UDP Internet Connections*) [Iyengar and Thomson 2020]. Originalmente proposto para possibilitar menor latência para transações HTTPS [Langley et al. 2017], o protocolo QUIC provê serviços de transmissão confiável equivalentes aos disponíveis no TCP, como controle de congestionamento e controle de fluxo. Além disso, o próprio protocolo prevê transmissão segura, com rápido estabelecimento de conexão, juntamente com multiplexação de fluxos de dados dentro de uma mesma conexão. Diferente dos protocolo TCP e MPTCP, cuja implementação ocorre no núcleo dos sistemas operacionais, o protocolo QUIC é implementado no espaço de usuário, junto à aplicação, facilitando a sua implantação, atualização e, consequentemente, a sua rápida adoção. Recentemente, QUIC tem recebido bastante atenção da academia e da indústria, havendo diversas implementações em diferentes linguagens [Piraux et al. 2018]. Umas das implementações evolui o protocolo QUIC em linguagem Go (`quic-go`) [Seemann and Clemente 2020] para o *MultiPath* QUIC (MPQUIC) [Coninck and Bonaventure 2020, De Coninck and Bonaventure 2017], habilitando transmissões por múltiplos caminhos a partir de subfluxos QUIC/UDP. Assim como MPTCP, o MPQUIC possibilita melhor uso dos recursos de rede ao agregar capacidades dos múltiplos caminhos disponíveis e, com isso, aumentar a vazão fim-a-fim e reduzir ainda mais latência.

Embora tenham ocorrido melhorias nas tecnologias de transmissão e em protocolos de camadas superiores da pilha TCP/IP, a arquitetura da Internet não passou pela mesma inovação. Vislumbrada há mais de quatro décadas, a arquitetura da Internet esteve sujeita ao problema de enrijecimento de tecnologia [Peterson et al. 2003], cuja inflexibilidade impede que os sistemas finais obtenham informações sobre o tráfego corrente nos caminhos de dados. Os sistemas finais nas redes de acesso possuem visibilidade apenas entre si, a partir de um caminho lógico de dados fim-a-fim definido pelos pares de endereços IP e portas de origem e destino. Dessa forma, os sistemas finais não recuperam informações quanto ao estado dos enlaces que compõem os caminhos fim-a-fim, especialmente as condições atuais dos enlaces de gargalos. Para contornar esse problema, os protocolos de transporte, como TCP, MPTCP, QUIC e MPQUIC, os quais tomam decisões de controle de congestionamento e/ou escalonamento de pacotes, devem inferir as condições dos caminhos medições passivas do tráfego fim-a-fim sobre a própria carga gerada pelas aplicações. Embora soluções de rede possam ser aplicadas para recuperar informações dos enlaces, como ECN (*Explicit Congestion Notification*) [Ramakrishnan et al. 2001] e SDN (*Software-Defined Networks*) [Kreutz et al. 2013], essas soluções são mais custosas ao exigirem modificações na infraestrutura de rede e introdução de novos elementos.

Em uma conexão de múltiplos caminhos, os subfluxos pertencentes à conexão podem compartilhar gargalos em comum e, com isso, disputarem recursos de redes entre si, levando a um problema conhecido como *contenção inter-caminhos* [Kimura et al. 2019]. Sem saber se os subfluxos compartilham ou não gargalos entre si, a dosagem da carga de trabalho sobre os subfluxos pode ser equivocadamente definida pelos mecanismos de controle de congestionamento e escalonamento de pacotes. Umas das for-

mas de se mitigar esse problema é prover junto aos protocolos de transporte o suporte à detecção de compartilhamento de gargalos, comumente denominado mecanismo SBD (*Shared Bottleneck Detection*). Conforme o padrão de Internet definido pela RFC 8382 [Hayes et al. 2018], um mecanismo SBD prevê o uso de métodos de medições passiva, principalmente de latência, sobre cada subfluxo existente uma conexão de múltiplos caminhos, para a extração de métricas estatísticas. Algoritmos de agrupamento podem então ser aplicados para determinar os grupos de subfluxos que compartilham gargalos a partir das similaridades das métricas correntes entre os subfluxos. Ao passo que os benefícios dos mecanismos SBDs em transmissões do protocolo MPTCP já são conhecidos [Ferlin et al. 2016a, Ferlin et al. 2016b, Wei et al. 2018], pouco se sabe sobre sua aplicação em transmissões do protocolo MPQUIC. No momento, até onde vai o conhecimento dos autores, as implementações atualmente disponíveis do protocolo MPQUIC não possuem suporte para detecção de gargalos compartilhamentos. Neste contexto, este artigo apresenta a implementação de um mecanismo SBD de acordo com o método previsto no padrão de Internet RFC 8382 [Hayes et al. 2018] junto ao protocolo MPQUIC.

Este artigo está organizado da seguinte forma. A seção seguinte apresenta os principais trabalhos relacionados. A Seção 3 descreve o método SBD conforme RFC 8382. A Seção 4 descreve a implementação do método junto ao protocolo MPQUIC. A Seção 5 discutes os resultados experimentais preliminares. Finalmente, a Seção 6 traz as conclusões e os trabalhos futuros.

2. Trabalhos Relacionados

O trabalho de [Rubenstein et al. 2002] apresenta as primeiras investigações sobre a detecção de gargalos compartilhados. Os autores propõem a medição de correlação de perda e atraso entre pares de pacotes, relacionando a correlação cruzada com a correlação entre pares dentro do fluxo. Caso a correlação cruzada seja maior que a correlação dentro do fluxo, conclui-se que os fluxos compartilham gargalos. Os autores identificaram que o método com correlação por atraso converge mais rapidamente para resposta correta, em comparação à correlação baseado em perdas de pacotes. Além disso, pode não haver amostras suficientes para que a correlação de perdas seja capaz de concluir se os subfluxos estão compartilhando gargalo.

[Zhu 2013] propôs um método de medição ativa para identificar fluxos que compartilham gargalo. Sondas são injetadas no tráfegos para medir o intervalo de tempo dos pacotes nas extremidades receptoras. Uma transformação de Fourier com o sinal de atraso de chegada entre pacotes injetados no tráfego é usada para detectar se os fluxos TCP compartilham gargalo.

[Hayes et al. 2014] propuseram o método SBD (*Shared Bottleneck Detection*), para detecção de gargalos compartilhados, fazendo uso de análise estatística de fluxos fim-a-fim. A análise baseia-se na coleta de tempos de ida OWD (*One-Way Delay*) e, com base em conjuntos de OWD amostrados, determinam-se métricas estatísticas de resumo das transmissões nos domínios de tempo e frequência. As métricas agrupam sub-fluxos a partir das diferenças estatísticas de pares de fluxo para, então, determinar se estão compartilhando gargalo. O método proposto se mostrou-se robusto em experimentos tanto em simulações quanto em ambientes reais. O método possui uma ampla variedade de aplicação, desde aplicações WebRTC [Alvestrand 2017] até possível implementação na

pilha de protocolos no núcleo de sistemas operacionais. Em [Ferlin et al. 2016b], os autores aplicam o método SBD junto ao controle de congestionamento do protocolo MPTCP, e para o cenário de gargalo não compartilhado observando ganhos de até 40% na taxa de transferência com dois sub-fluxos e os ganhos aumentam significativamente à medida aumenta o número de sub-fluxos, atingindo mais de 100% durante cinco sub-fluxos. Além disso, para o cenário de gargalo compartilhado, os autores mostraram que o MPTCP-SBD permanece justo com fluxos TCP.

[Wei et al. 2018] propuseram SBDV (*Shared Bottleneck Detection Based on Congestion Interval Variance Measurement*), método que prevê a medição da variância de OWDs para detectar gargalos compartilhados. O objetivo é melhorar a acurácia de medições de fluxos que compartilham gargalos em comum, mas, por percorrerem caminhos distintos, apresentam latências diferentes – essas situações são denominadas *path lag* na literatura. Para tanto, SBDV opera da seguinte forma. Primeiro, define-se um delta $\Delta = t_2 - t_1 = (t_1 - t_0) - (t_2 - t_0)$, sendo t_1 e t_2 os tempos de chegada no receptor dos fluxos 1 e 2, respectivamente, correspondendo aos atrasos dos enlaces P1 e P2. Com isso, define-se a variância em relação ao delta, $Vari(\Delta)$, sendo um valor que reflete a intensidade da flutuação do atraso na fila nos enlaces não congestionados, ou seja, P1 e P2. Gargalos compartilhados são detectados entre P1 e P2 quando $Vari(\Delta)$ ultrapassa um limiar, que é a média quadrática dos tempos de duração do evento de congestionamento dos dois subfluxos que estão compartilhando gargalo. O algoritmo demonstrou robustez e eficácia em cenários de *path lag*. Em um outro trabalho [Wei et al. 2020], os autores abordam o problema de forma diferente, utilizando o suporte do mecanismo ECN (*Explicit Congestion Notification*) para detectar gargalos compartilhados e estimar o grau de congestionamento sobre cada subfluxo. Essa detecção é aplicada nos mecanismos de controle de congestionamento e escalonamento de pacotes, denominados SB-CC e SB-FBS, respectivamente, para melhorar a vazão de transmissões MPTCP. SB-CC equilibra as cargas nos subfluxos e suaviza a flutuação da janela de congestionamento. Para evitar a degradação da taxa de transmissão por devido aos pacotes fora de ordem, SB-FBS distribui os pacotes de acordo com as alterações nas janelas de congestionamento, podendo escalonar com maior precisão os pacotes em cenários de gargalo compartilhado.

Este trabalho parte dos estudos de [Hayes et al. 2014] e [Ferlin et al. 2016b], os quais levaram a definição do padrão RFC 8382 [Hayes et al. 2018], para a implementação e validação de um mecanismo SBD junto ao protocolo MPQUIC. Embora existam diferentes abordagens para detectar compartilhamento gargalos, o padrão RFC 8382 representa o estado da arte. Neste contexto, este trabalho e difere dos demais ao habilitar suporte SBD baseada na RFC 8382 junto ao protocolo MPQUIC.

3. Métricas SBD conforme RFC 8382

O padrão RFC 8382 [Hayes et al. 2018] prevê um método SBD com três importantes etapas: (1) medições de latência unidirecional (OWD) em cada subfluxo pertencente a uma conexão de múltiplos caminhos; (2) extração de métricas estatísticas a partir de amostras de OWD medido. As subseções seguintes descrevem maiores detalhes destas etapas.

3.1. Medições de OWD

O método SBD utiliza métricas estatísticas baseadas em medições relativas de OWD para inferir se os subfluxos pertencentes a uma conexão de múltiplos caminhos estão compar-

tilhando um gargalo entre si em um dado momento da transmissão. Uma das formas de se obter medições relativas de OWD é a partir da diferença entre as marcações de tempo de pacotes recebidos no nó de destino, sendo

$$OWD = t_i - t_{i-1}, \tag{1}$$

onde t_i é a marcação de tempo do pacote do i-ésimo pacote.

O método prevê múltiplas janelas T de medições em um período de $M \leq N$ janelas. Portanto, a coleta de medições e extração de estatística ocorrem sob o deslocamento de $M \times T$ janelas no tempo. A parametrização sugerida na RFC está descrita na Tabela 1.

Tabela 1. Parâmetros sugeridos pela RFC 8382 [Hayes et al. 2018].

Parâmetro	Valor	Descrição
T	350 ms	O intervalo de tempo base no qual as medições são feitas.
N	50	Número máximo de intervalos de tempo base, T, necessários para determinar métricas estatísticas.
M	30	Número mínimo de intervalos de tempo base, T, necessários para determinar métricas estatísticas, sendo $M \leq N$.

3.2. Extração de Métricas Estatísticas

As métricas previstas no método são o atraso médio, a assimetria, a variância, a oscilação e, por fim, a perda de pacotes. Essas métricas são detalhadas a seguir. Dentro de um intervalo de tempo base T (350 ms) várias medições de OWD podem ser amostradas. Dessa forma, o valor esperado é o valor médio de OWD no intervalo T, denominado $E_T(OWD)$. A partir de $E_T(OWD)$, determina-se a média global sobre o espaço M de valores. Dessa forma **o atraso médio** é dado por

$$MeanDelay = \frac{SUM_M(E_T(OWD))}{M}, \tag{2}$$

A **estimativa de assimetria** ($SkewEst$) é uma métrica que varia de positivo para negativo, quando se aproxima de 100% da carga suportada pelo caminho. Isso torna a métrica interessante, pois a ocorrência de congestionamento é justamente quando a carga excede a capacidade do enlace. Para tanto, a variável $SkewBase_T$ determina de forma discreta essa assimetria, conforme o Algoritmo 1, sendo positiva quando a quantidade de observações OWD em T é menor que o atraso médio conhecido, ou negativa, caso contrário. Portanto, ao final de T, a variável $SkewBase_T$ fornecerá um indicador discreto do quantitativo da assimetria, a qual pôde ter variado positiva ou negativamente.

Algorithm 1 $SkewBase_T$: assimetria em T

1: **if** $OWD < MeanDelay$ **then**
2: $\quad SkewBase_T += 1$
3: **end if**
4: **if** $OWD > MeanDelay$ **then**
5: $\quad SkewBase_T -= 1$
6: **end if**

Uma vez conhecida a assimetria do intervalo T, a estimativa de assimetria no intervalo $M \times T$ é dada pela média:

$$SkewEst = \frac{SUM_{M \times T}(skewBaseT)}{M}, \tag{3}$$

A **estimativa de variância** ($VarEst$) dos atrasos observados pelo subfluxo durante um período é determinada por uma função monotonicamente decrescente para os subfluxos que experimentam congestionamento. A estimativa é dada por:

$$VarEst = \frac{SUM_{M \times T}(varBaseT)}{NUM_{M \times T}(OWD)}, \tag{4}$$

onde o numerador indica a soma no período $M \times T$ das variância observadas em cada intervalo T, e o denominador traz o total de medições de OWD realizadas no intervalo $M \times T$. Para cada nova medição de OWD no intervalo T, o valor residual entre o OWD observado e o OWD médio conhecido é cumulativamente armazenado em $VarBaseT$. Dessa forma, a variância observada em um intervalo T é dada por:

$$VarBaseT = SUM_T(|OWD - E_T(OWD)|), \tag{5}$$

onde $| x |$ é o valor absoluto de x, OWD é a nova medição de OWD, $E_T(OWD)$ é a média calculada no intervalo T anterior.

A **estimativa de oscilação** ($FreqEst$) determina a oscilação de baixa frequência no atraso observado pelo subfluxo durante um período. Uma medição consistente da frequência do subfluxo é capaz de distinguir subfluxos que estão ou não sob congestionamento [Hayes et al. 2014]. A estimativa de oscilação de baixa frequência é dada por:

$$FreqEst = \frac{numberOfCrossings}{M}, \tag{6}$$

onde $numberOfCrossings$ armazena a contagem de oscilações do OWD médio durante a transmissão, e M é número máximo de intervalos de tempo base T. Para determinar as oscilações, considera-se então a contagem de vezes em que o OWD médio conhecido foi maior ou menor que a amplitude média observada. Essa amplitude média é dada por:

$$MeanDelay \pm pv \times VarEst, \tag{7}$$

onde $MeanDelay$ é o atraso médio conforme Eq.(2), pv é um limiar de sensibilidade que considera eventuais ruídos amostrais, e $VarEst$ é a estimativa de variância conforme Eq.(4). As oscilações da média $E_T(OWD)$ então ocorrem a partir da alternância entre dois estados:

(1) Quando $E_T(OWD)$ for alto a ponto de cruzar o limite superior da amplitude média;
(2) Quando $E_T(OWD)$ for baixo a ponto de cruzar o limite inferior da amplitude média.

Para cada nova média $E_T(OWD)$ conhecida, o número de oscilações em $numberOfCrossings$ é determinado pela contagem de alternâncias entre dois estados, conforme descreve o Algoritmo 2.

Algorithm 2 Contagem de alternância $numberOfCrossings$.

1: **if** $E_T(OWD) \geq (MeanDelay + pv \times VarEst)$ **and** $lastState \neq 1$ **then**
2: $\quad numberOfCrossings += 1$
3: $\quad lastState = 1$
4: **else if** $E_T(OWD) \leq (MeanDelay - pv \times VarEst)$ **and** $lastState \neq 2$ **then**
5: $\quad numberOfCrossings += 1$
6: $\quad lastState = 2$
7: **end if**

A **taxa de perda de pacotes** observada na janela $M \times T$ é dada por:

$$PktLoss = \frac{SUM_{M \times T}(lostPackets)}{SUM_{M \times T}(totalPackets)}. \tag{8}$$

onde $lostPackets$ e $totalPackets$ são a quantidades de pacotes perdidos e o total enviado em cada intervalo T, respectivamente.

4. Implementação do método SBD no MPQUIC

Como ponto de partida utilizamos a implementação do MPQUIC `mpquic-go` de [De Coninck and Bonaventure 2017], a qual foi a primeira implementação do protocolo proposta, estendendo a implementação de QUIC em linguagem Go (`quic-go`) [Seemann and Clemente 2020]. Dessa forma, o método SBD está sendo implementado em Go, estendendo a implementação de `mpquic-go`. As subseções seguintes descrevem pontos importantes da implementação realizada até o momento, em termos de componentes da arquitetura, principais procedimentos e estrutura dos pacotes.

4.1. Considerações de Projeto

Como as métricas estatísticas baseiam-se nas amostras de OWD coletadas pelo receptor, enquanto o emissor contabiliza as perdas de pacotes, não sendo possível implementar o método SBD de forma autônoma, complemente no emissor ou completamente no receptor. Dessa forma, tem-se ao menos duas abordagens de implementação SBD:

(1) **Emissor complexo.** Além dos mecanismos de controle MPQUIC existentes no emissor, como controle de congestionamento e de fluxo, bem como escalonamento de pacotes, a implementação do método SBD também seria acoplada ao emissor. Nesse caso, requer que o receptor retorne ao emissor (via pacotes ACK, por exemplo) os valores de OWD por ele medidos, de modo que o emissor execute todo as operações de extração de métricas e agrupamentos dos subfluxos.
(2) **Receptor complexo.** Nesse caso, a implementação do método SBD seria acoplada ao receptor. Além das medições de OWD, o receptor também seria responsável por extrair as métricas e realizar o agrupamento dos subfluxos, retornando ao emissor (via pacotes ACK) somente o resultado do método, i.e., a informação de agrupamento dos subfluxos.

Independente da abordagem, as decisões que o emissor (controle de congestionamento e escalonador de pacotes) poderá tomar com base no suporte SBD sofrerá ao menos um atraso de $\frac{1RTT}{2}$. Ambos pares precisam de informações um do outro para prosseguirem tanto no método SBD ou quanto na tomada de decisão. Entretanto, a abordagem (2) apresenta-se mais vantajosa. Nesse caso, um mecanismo que implementa o método pode

ser mais ágil, pois as entradas para o mesmo estarão disponíveis depois que o receptor realizar as medições de OWD. Além disso, o receptor pode executar quase todas as operações do mecanismo de forma independente, exceto sobre a perda de pacotes. Outra vantagem é que evita-se maior transmissão de dados de controle (amostras de OWD e métricas), via pacotes ACK que podem ser perdidos e, neste caso, não retransmitidos.

Uma sessão MPQUIC entre o emissor e receptor e seus principais componentes são ilustrados na Figura 1. A figura ilustra uma sessão de quatro subfluxos. O processamento de pacotes por múltiplos caminhos requer que o MPQUIC tenha determinados componentes específicos por subfluxo, como os *handlers* de envio e recebimento, controle de congestionamento. No escopo da sessão, o Gerenciador de Caminhos é responsável por lidar com o estabelecimento e monitoramento dos subfluxos. O Controle de Congestionamento dos subfluxos opera de forma acoplada, considerando a agregação das capacidades de todos os subfluxos para então ajustar dinamicamente as janelas individuais. Quando um pacote está pronto para ser transmitido, o escalonador é responsável por determinar qual subfluxo irá enviá-lo. O mecanismo SBD é um novo módulo acoplado ao gerenciador de caminhos no receptor, o qual mede OWDs e extrai métricas estatísticas de cada subfluxo, já descrito na Seção 3, para agrupar os subfluxos sob gargalos compartilhados.

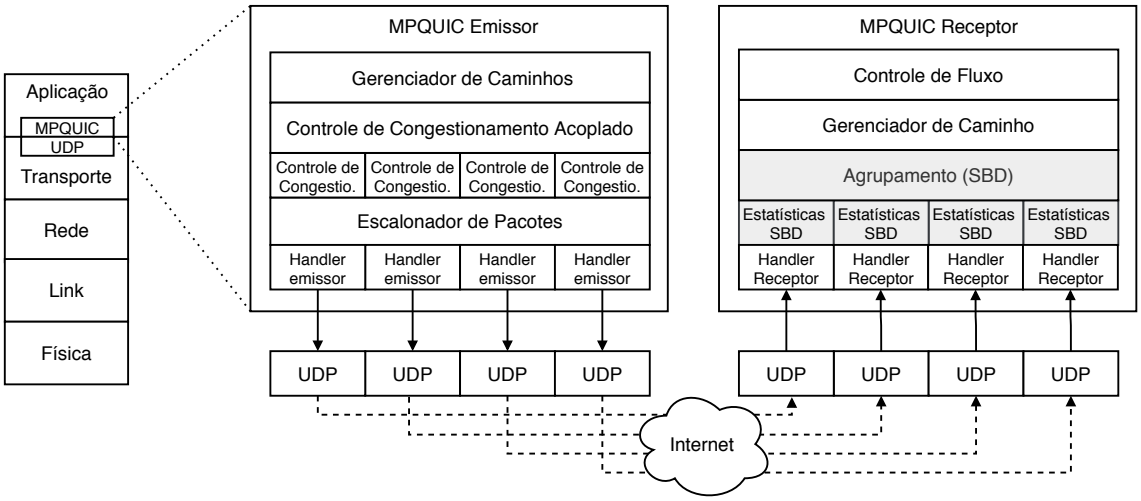


Figura 1. Componentes de uma implementação do MPQUIC e sua modificação (cinza) com o mecanismo SBD, considerando a abordagem de implementação de um receptor complexo. Imagem adaptada de [Viernickel et al. 2018].

4.2. Novos Campos nos Quadros

Tendo em vista que a abordagem escolhida de implementação de um receptor MPQUIC mais complexo, foi necessário modificar a estrutura tanto de pacotes `ACK` quanto de pacotes `stream`. A ilustração na Figura 2 mostra as modificações. Nos pacotes de confirmação, agora há um novo campo chamado `Groups`, de 1 byte de comprimento, o qual informa ao emissor a saída do mecanismo SBD, contendo a identificação do grupo o qual o subfluxo pertence naquele momento. Um espaço de 1 byte permite a identificação de até 256 grupos, portanto, é bastante suficiente para a finalidade de agrupamento de subfluxos no MPQUIC. O quadro *Ack Frame* originalmente contém o campo `PathID` que indica qual caminho o quadro pertence. Portanto, a informação da classificação de cada caminho é armazenada em um *buffer* e inserida no *Ack Frame* correspondente. O

mecanismo é redundante, enviando a mesma informação de agrupamento atual em todos quadros ACK do receptor para o emissor, e não somente quando uma nova configuração de agrupamento é determinada. Esse mecanismo pode mitigar a possibilidade de perda de informação, caso quadros `ACK` sejam perdidos. Como o mecanismo SBD no receptor depende da informação de pacotes perdidos, a qual é determinada pelo emissor, os pacotes *stream* utilizados pelo emissor na transmissão de dados da aplicação tem seu cabeçalho modificado para, agora, conter o novo campo de $LossCount$. Esse atributo de também tem 8 bits de comprimento para informar o receptor sobre a quantidade de pacotes perdidos até o momento, de modo que o mesmo possa calcular a taxa de $Pktloss$.

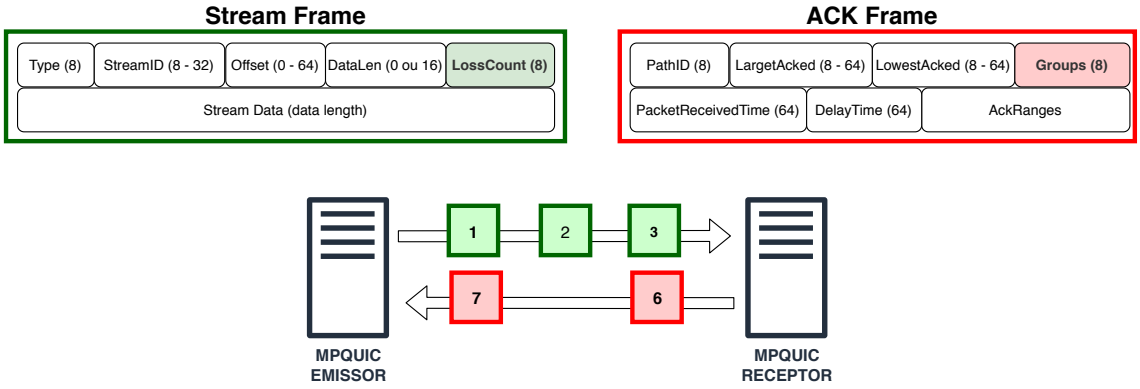


Figura 2. Emissor e receptor e seus quadros de fluxo de dados (*stream*) e reconhecimento (*ACK*) transmitidos em um subfluxo. Novos campos incluídos pelo mecanismo SBD são `LossCount` e `Groups` nos quadros de dados e de reconhecimento, respectivamente. Comprimento dos campos em bits.

4.3. Principais Procedimentos

O mecanismo SBD é executado conforme o fluxograma na Figura 3, a partir das principais procedimentos. Com a medição mais recente de OWD, o receptor então:

(1) Atualiza-se a a média $E_T(OWD)$ com o procedimento `ComputeMean()`.
(2) Soma-se à $VarBaseT$ o valor residual entre o OWD observado e a média conforme Equação (5).
(3) Computa de forma discreta a assimetria em $SkewBaseT$, conforme Algoritmo 1.

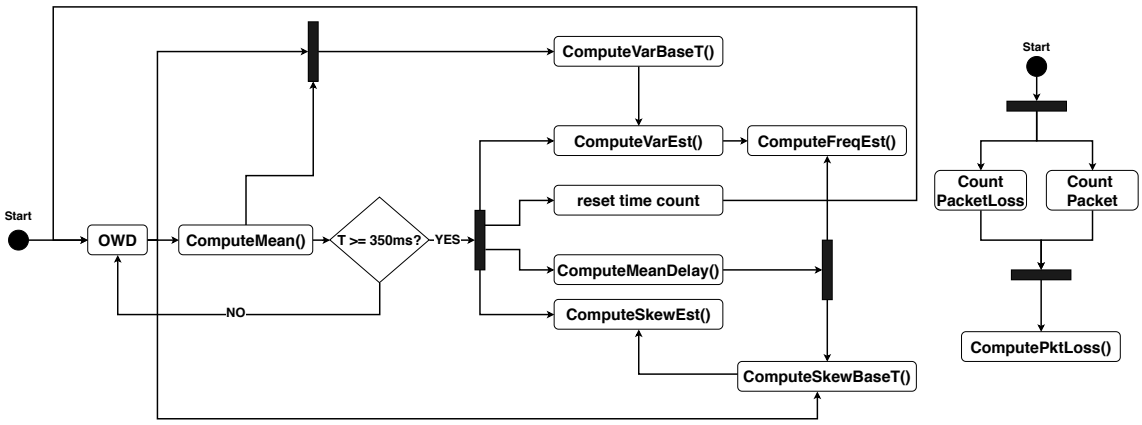


Figura 3. Fluxograma da operação do mecanismo SBD junto ao MPQUIC.

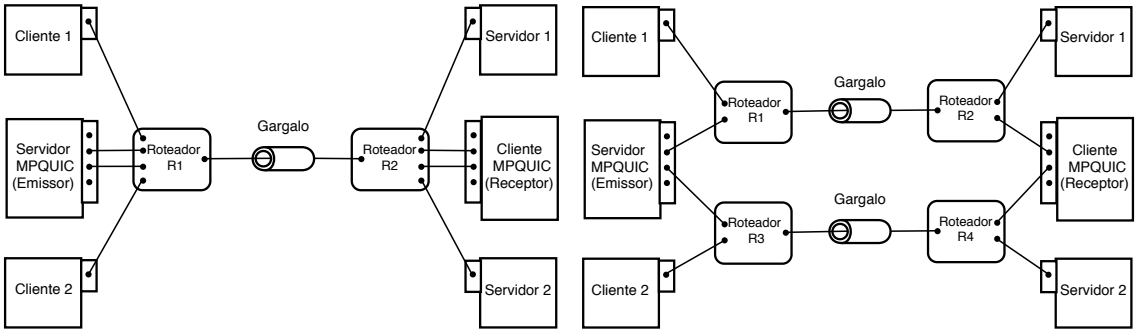
Como prevê a RFC 8382, as métricas são obtidas em janelas de tempo $M \times T$, tendo $M = 30$ e $T = 350$ ms. Dessa forma, dentro da janela $T < 350$ ms, apenas as métricas básicas de média $E_T(OWD)$ e $VarBaseT$ são calculadas. A cada término do intervalo T, reinicializa-se o contador e atualizam-se as métricas $MeanDelay$, $SkewEst$, $VarEst$ e $FreqEst$, conforme Equações (2), (3), (4) e (6), implementadas através dos procedimentos `ComputeMeanDelay()`, `ComputeSkewEst()`, `ComputeVarEst()` e `ComputeFreqEst()`, respectivamente.

5. Resultados Preliminares

Esta seção discute os resultados experimentais obtidos até o momento.

5.1. Ambiente experimental

Para avaliar o funcionamento do mecanismo implementado foram realizados experimentos no emulador `mininet`[1]. Dois cenários base foram implementados no emulador, um com compartilhamento de gargalo e o outro sem compartilhamento de gargalo, conforme ilustra as Figuras 4a e 4b, respectivamente.



(a) Cenário com um gargalo compartilhado **(b) Cenário com gargalos não-compartilhados**

Figura 4. Ambiente experimental emulado no mininet.

Os cenários foram configurados com gargalos de capacidade de 20 Mbps ou 10 Mbps e RTT de 20 ms. No cenário com gargalo compartilhado utilizou 20 Mbps e para o cenário com gargalos não compartilhado utilizou 20 Mbps para um gargalo e 10 Mbps para o outro, como em [Ferlin et al. 2016a]. O tamanho das filas nos roteadores ligados aos gargalos foi definido em 1 BDP, cerca de 35 pacotes. Nesses cenários, os nós MPQUIC são servidores emissores e clientes receptores, os quais estão conectados aos roteadores por duas interfaces ethernet. O servidor utilizado foi `Caddyserver`[2], que é uma implementação de servidor HTTP em linguagem Go, possibilitando integração junto ao MPQUIC. Do outro, no cliente foi utilizado *player* emulador `AStream`[3] de transmissão multimídia via DASH (*Dynamic Adaptive Streaming over HTTP*), através de extensão[4] para operar sobre `mpquic-go`. No mesmo sentido do fluxo de *downstream* (da esquerda para a direita na Fig. 4), o tráfego de *background* foi gerado através da ferramenta `DTIG` nos nós clientes 1 e 2 aos servidores 1 e 2, onde cada cliente transmitia por cerca de 10 segundos rajadas de fluxos UDP e TCP, alternando para 2 segundos de ociosidade, sem emissão de tráfego de *background*.

[1] `http://mininet.org`
[2] `https://caddyserver.com`
[3] `https://github.com/pari685/AStream`
[4] `https://github.com/deradev/mpquicScheduler`

5.2. Métricas observadas em cenários de gargalos compartilhados e não-compartilhados

As métricas obtidas durante as transmissões MPQUIC são apresentadas nas Figuras 5 e 6 para os cenário de gargalos compartilhados e não-compartilhados, respectivamente. Por questões de visualização, os gráficos apresentam as métricas extraídas no intervalo de transmissão de 10 a 60 segundos. Para detectar o compartilhamento de gargalos, a RFC 8382 prevê primeiro determinar se os subfluxos passam por congestionamento, para então considerar as métricas em um algoritmo de agrupamento. Isso inicialmente é verificado pela métrica de $SkewEst$, ao atingir limiares de 0.3, ou quando a perda $PktLoss$ atingir 0.1. Os limiares, contudo, podem ser ajustados conforme os cenários, os valores da RFC 8382 baseiam-se em experimentos de certa complexidade em relação à combinação de gargalos e tráfego de rede obtidos e ajustados durante o desenvolvimento do algoritmo.

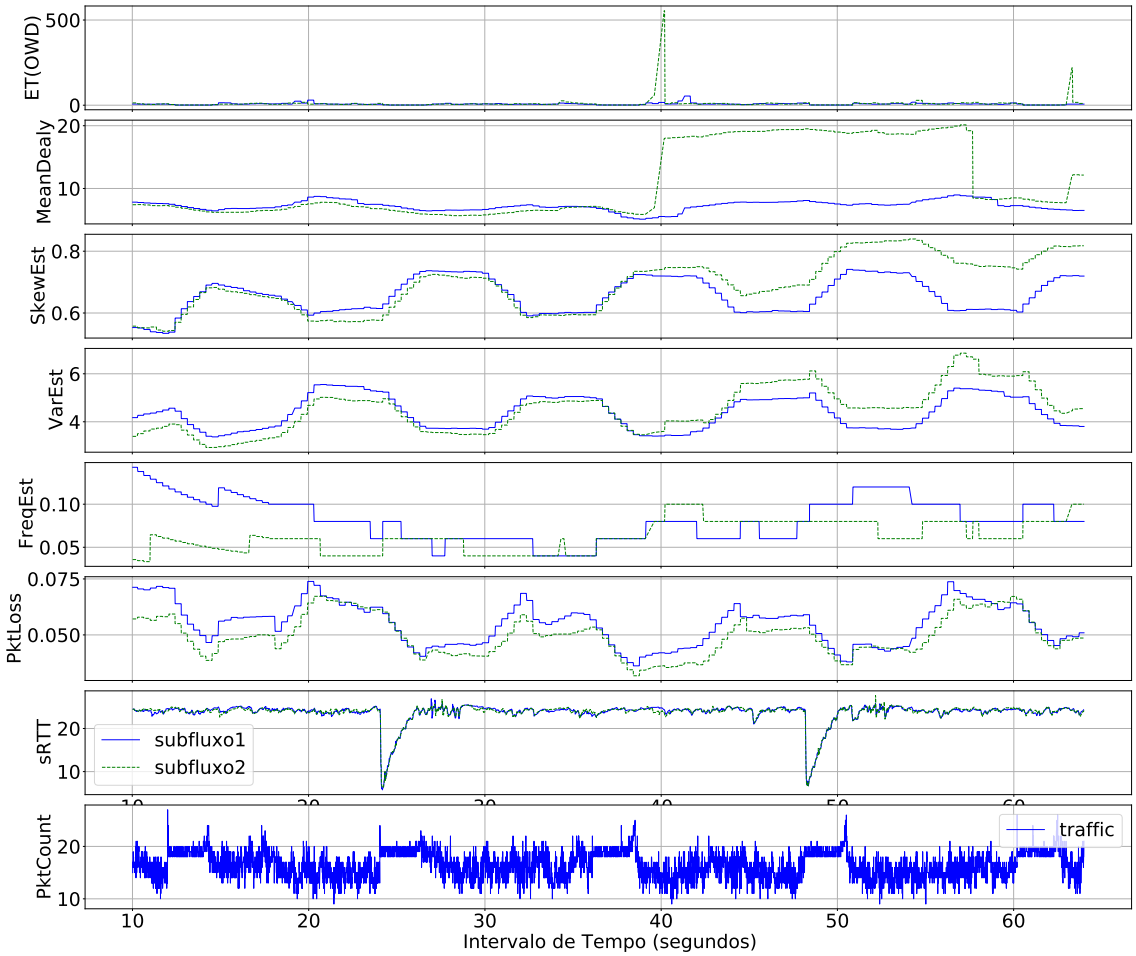


Figura 5. Amostras de métricas SBD em cada subfluxo em uma conexão MPQUIC no cenário de gargalo compartilhado (Fig. 4a).

As métricas observadas possuem efeito serrilhado devido ao intervalo base T, de 350 ms, em que são atualizadas. A métrica de $MeanDelay$ possui curvas similares quando os subfluxos compartilham um mesmo gargalo, conforme mostra o gráfico da Figura 5. Já no cenário onde os subfluxos percorrem gargalos não-compartilhados, as curvas do $MeanDelay$ não são equivalentes, conforme Figura 6. Nesse cenário, em

geral as que métricas observadas pelos subfluxos não possuem similaridade ou correlação que seja possível identificar visualmente pelo gráfico, como observado nas métricas de $SkewEsT$, $VarEst$, $FreqEst$, $PktLoss$ e $sRTT$.

No cenário compartilhado, as curvas bastante similares entre os subfluxos. Observa-se a reação das métricas ($VarEst$) e assimetria ($SkewEst$) mediante à variação da carga no enlace de gargalo (número de pacotes no gráfico *pktCount*) sob as rajadas de tráfego de *background*. Essa carga se refere o total de pacotes trafegando no enlace de gargalo em janelas de 10 ms. A oscilação de baixa frequência observada no cenário de gargalos compartilhados sofreu maior variação que no cenário não-compartilhado. Isso ocorreu, pois, a oscilação é uma métrica pouco sensível a amostragem de OWD, retratando o longo-prazo ao considerar a média de OWD em T, a variância e de $MeanDelay$, as duas últimas sumarizadas em janelas $M \times T$. Para torná-lo mais sensível ao ruído gerado das oscilações de latência ao longo da transmissão, o limiar pv requer foi ajustado para $pv = 0.8$. As taxas de perdas no cenário compartilhado tendem a acompanhar a carga de trabalho (*pktCount*) observada no gargalo. Os tempos de ida-e-volta observados são levemente superiores ao RTT de 20 ms configurado no cenário compartilhado. Ambos subfluxos apresentam latência muito similares, já que compartilham o mesmo gargalo.

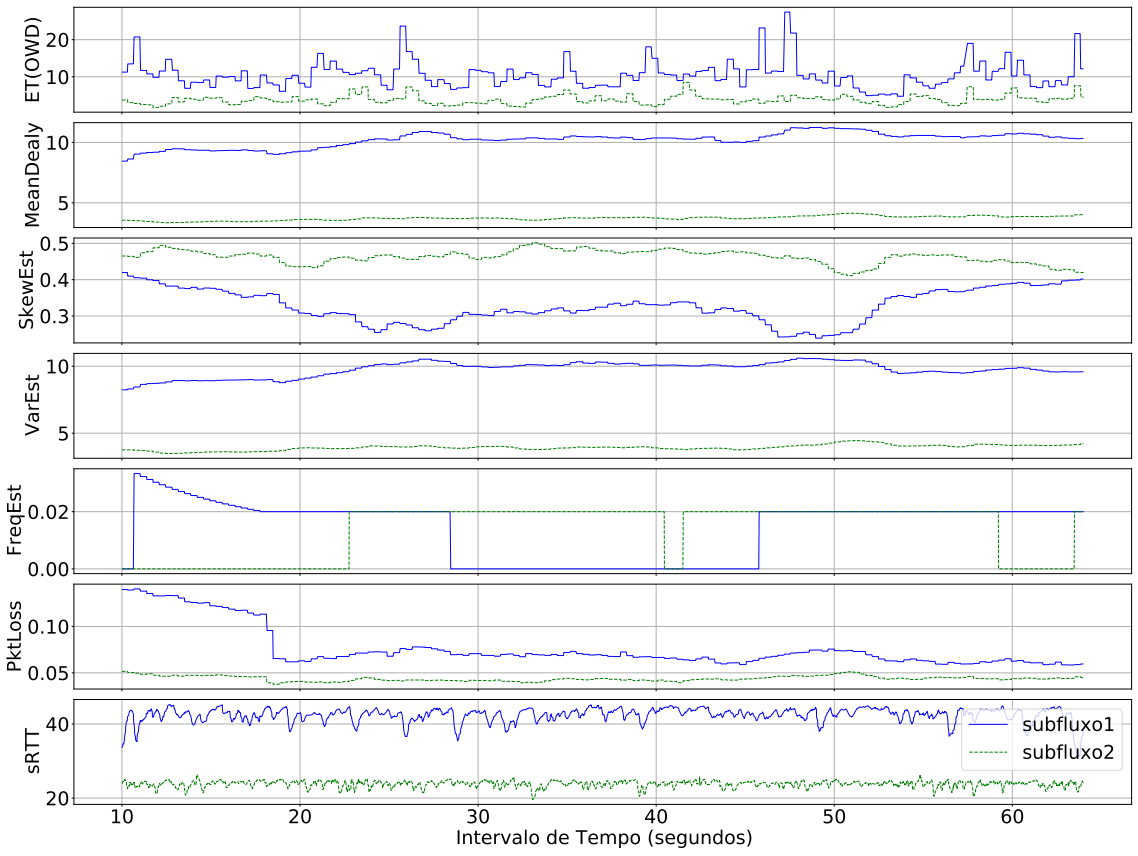


Figura 6. Amostras de métricas SBD em cada subfluxo em uma conexão MPQUIC no cenário de gargalo não-compartilhado (Fig. 4b).

6. Conclusão e trabalhos futuros

Este artigo introduz a implementação e validação de um mecanismo SBD baseado na RFC 8382 junto ao MPQUIC, possibilitando, neste momento, a amostragem de OWD e a extração de métricas estatísticas sobre os subfluxos QUIC/UDP pertencentes a uma conexão MPQUIC. Os impactos do suporte de tal mecanismo, enquanto já são conhecidos sobre o protocolo MPTCP, são desconhecidos em transmissões MPQUIC. O próximo passo do trabalho é implementar e avaliar algoritmos de agrupamento dos subfluxos que tomam como entrada as métricas estatísticas extraídas e também explorar recursos do QUIC como um protocolo que difere do TCP, para melhorar a precisão do mecanismo SBD e a transmissão de informações entre o emissor e o receptor. Com isso, a qualidade de transmissões diversas, especialmente as que não são comumente realizadas com MPTCP, e.g. multimídia, poderá ser investigada experimentalmente, considerando tomadas de decisão baseadas em SBD junto ao controle de congestionamento e escalonamento de pacotes no emissor MPQUIC.

Referências

Alvestrand, H. T. (2017). Overview: Real Time Protocols for Browser-based Applications. Internet-Draft draft-ietf-rtcweb-overview-19, Internet Engineering Task Force. Work in Progress.

Coninck, Q. D. and Bonaventure, O. (2020). Multipath Extensions for QUIC (MP-QUIC). Internet-Draft draft-deconinck-quic-multipath-04, Internet Engineering Task Force. Work in Progress.

De Coninck, Q. and Bonaventure, O. (2017). Multipath quic: Design and evaluation. In *Proceedings of the 13th International Conference on Emerging Networking EXperiments and Technologies*, CoNEXT '17, page 160–166, New York, NY, USA. Association for Computing Machinery.

Ferlin, S., Alay, Ö., Dreibholz, T., Hayes, D. A., and Welzl, M. (2016a). Revisiting congestion control for multipath tcp with shared bottleneck detection. In *IEEE INFOCOM 2016 - The 35th Annual IEEE International Conference on Computer Communications*, pages 1–9.

Ferlin, S., Alay, Ö., Dreibholz, T., Hayes, D. A., and Welzl, M. (2016b). Revisiting congestion control for multipath TCP with shared bottleneck detection. In *The 35th Annual IEEE International Conference on Computer Communications (INFOCOM 2016)*, pages 1–9.

Ford, A., Raiciu, C., Handley, M., Barre, S., and Iyengar, J. (2011). Architectural Guidelines for Multipath TCP Development. Internet Standart RFC 6182, IETF.

Ford, A., Raiciu, C., Handley, M. J., and Bonaventure, O. (2013). TCP Extensions for Multipath Operation with Multiple Addresses. RFC 6824.

Hayes, D., Ferlin, S., Welzl, M., and Hiorth, K. (2018). Shared Bottleneck Detection for Coupled Congestion Control for RTP Media. RFC 8382.

Hayes, D. A., Ferlin, S., and Welzl, M. (2014). Practical passive shared bottleneck detection using shape summary statistics. In *39th Annual IEEE Conference on Local Computer Networks*, pages 150–158.

Iyengar, J. and Thomson, M. (2020). QUIC: A UDP-Based Multiplexed and Secure Transport. Internet-Draft draft-ietf-quic-transport-29, Internet Engineering Task Force. Work in Progress.

Kimura, B. Y. L., Lima, D. C. S. F., Villas, L. A., and Loureiro, A. A. F. (2019). Interpath contention in multipath tcp disjoint paths. *IEEE/ACM Transactions on Networking*, 27(4):1387–1400.

Kreutz, D., Ramos, F. M., and Verissimo, P. (2013). Towards secure and dependable software-defined networks. In *Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking*, pages 55–60.

Langley, A., Riddoch, A., Wilk, A., Vicente, A., Krasic, C., Zhang, D., Yang, F., Kouranov, F., Swett, I., Iyengar, J., Bailey, J., Dorfman, J., Roskind, J., Kulik, J., Westin, P., Tenneti, R., Shade, R., Hamilton, R., Vasiliev, V., Chang, W.-T., and Shi, Z. (2017). The quic transport protocol: Design and internet-scale deployment. In *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*, SIGCOMM '17, page 183–196, New York, NY, USA. Association for Computing Machinery.

Peterson, L., Anderson, T., Culler, D., and Roscoe, T. (2003). A blueprint for introducing disruptive technology into the internet. *ACM SIGCOMM Computer Communication Review*, 33(1):59–64.

Piraux, M., De Coninck, Q., and Bonaventure, O. (2018). Observing the evolution of QUIC implementations. In *Proceedings of the Workshop on the Evolution, Performance, and Interoperability of QUIC*, pages 8–14.

Ramakrishnan, K., Floyd, S., and Black, D. (2001). The addition of explicit congestion notification (ecn) to ip. Internet Standart RFC 3168, IETF.

Rubenstein, D., Kurose, J., and Towsley, D. (2002). Detecting shared congestion of flows via end-to-end measurement. *IEEE/ACM Transactions on Networking*, 10(3):381–395.

Seemann, M. and Clemente, L. (2020). A quic implementation in pure go. `https://github.com/lucas-clemente/quic-go`.

Viernickel, T., Froemmgen, A., Rizk, A., Koldehofe, B., and Steinmetz, R. (2018). Multipath quic: A deployable multipath transport protocol. In *2018 IEEE International Conference on Communications (ICC)*, pages 1–7.

Wei, W., Wang, Y., Xue, K., Wei, D. S. L., Han, J., and Hong, P. (2018). Shared bottleneck detection based on congestion interval variance measurement. *IEEE Communications Letters*, 22(12):2467–2470.

Wei, W., Xue, K., Han, J., Wei, D. S. L., and Hong, P. (2020). Shared bottleneck-based congestion control and packet scheduling for multipath tcp. *IEEE/ACM Transactions on Networking*, pages 1–14.

Zhu, W. (2013). Tcp path sharing detection. In *2013 IEEE 11th Malaysia International Conference on Communications (MICC)*, pages 244–249.