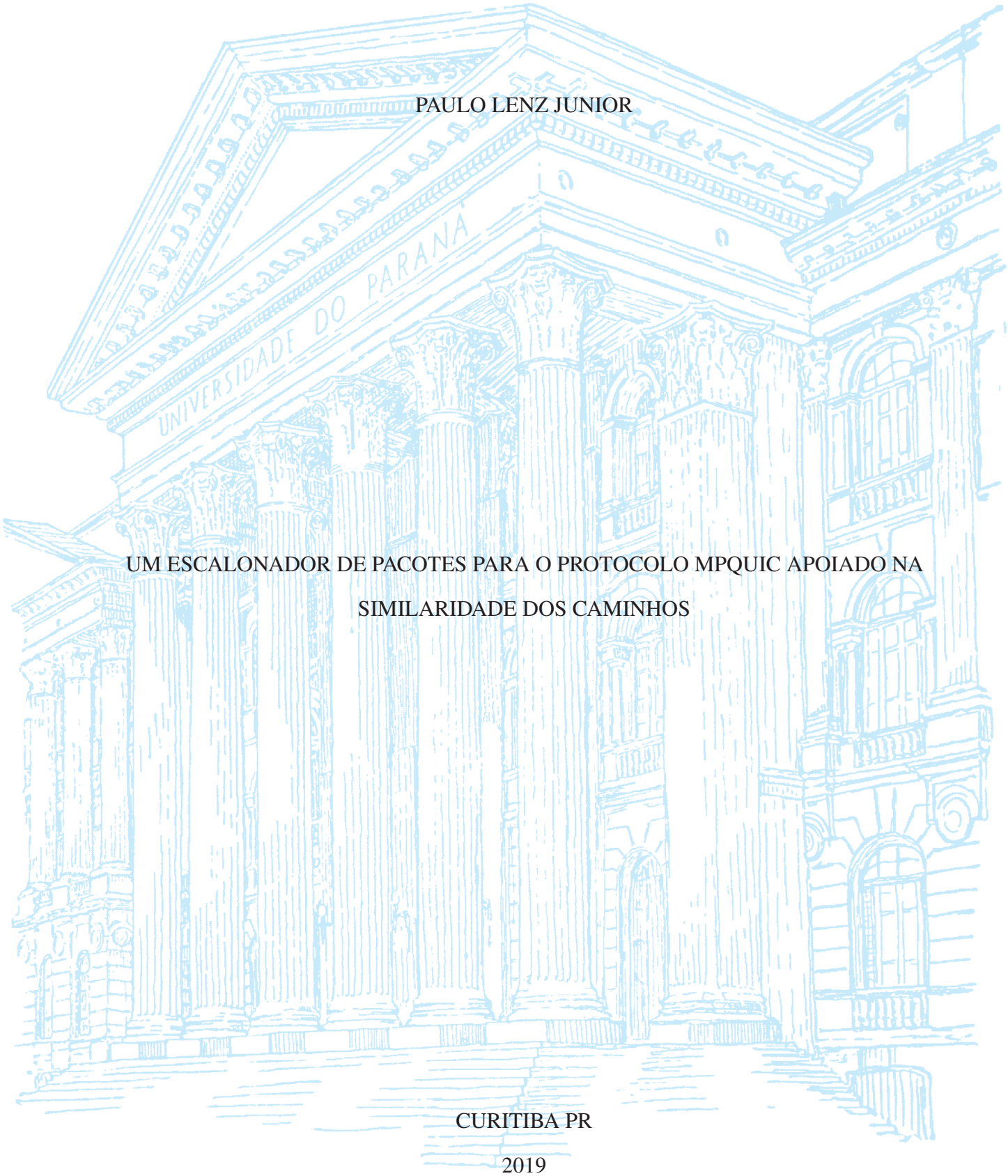
UNIVERSIDADE FEDERAL DO PARANÁ

PAULO LENZ JUNIOR



UM ESCALONADOR DE PACOTES PARA O PROTOCOLO MPQUIC APOIADO NA SIMILARIDADE DOS CAMINHOS

CURITIBA PR

2019

PAULO LENZ JUNIOR

UM ESCALONADOR DE PACOTES PARA O PROTOCOLO MPQUIC APOIADO NA SIMILARIDADE DOS CAMINHOS

Dissertação apresentada como requisito parcial à obtenção do grau de Mestre em Informática no Programa de Pós-Graduação em Informática, Setor de Ciências Exatas, da Universidade Federal do Paraná.

Área de concentração: *Ciência da Computação*.

Orientador: Michele Nogueira Lima.

CURITIBA PR

2019

Catalogação na Fonte: Sistema de Bibliotecas, UFPR
Biblioteca de Ciência e Tecnologia

L575e

Lenz Junior , Paulo
Um escalonador de pacotes para o protocolo MPQUIC apoiado na similaridade dos caminhos [recurso eletrônico] / Paulo Lenz Junior . – Curitiba, 2019.

Dissertação - Universidade Federal do Paraná, Setor de Ciências Exatas, Programa de Pós-Graduação em Informática, 2019.

Orientador: Michele Nogueira Lima

1. Sistemas de transmissão de dados. 2. Comutação de pacotes (Transmissão de dados). 3. Computação de alto desempenho. I. Universidade Federal do Paraná. II. Lima, Michele Nogueira. III. Título.

CDD: 005.71

Bibliotecário: Elias Barbosa da Silva CRB-9/1894

MINISTÉRIO DA EDUCAÇÃO
SETOR DE CIENCIAS EXATAS
UNIVERSIDADE FEDERAL DO PARANÁ
PRÓ-REITORIA DE PESQUISA E PÓS-GRADUAÇÃO
PROGRAMA DE PÓS-GRADUAÇÃO INFORMÁTICA -
40001016034P5

TERMO DE APROVAÇÃO

Os membros da Banca Examinadora designada pelo Colegiado do Programa de Pos-Graduação em INFORMÁTICA da Universidade Federal do Parana foram convocados para realizar a arguição da Dissertação de Mestrado de **PAULO LENZ JUNIOR** intitulada: **Um Escalonador de Pacotes para o Protocolo MPQUIC Apoiado na Similaridade dos Caminhos**, sob orientação do Prof. Dr. MICHELE NOGUEIRA LIMA, que após após terem inquirido o aluno e realizado a avaliação do trabalho, são de parecer pela sua APROVAÇÃO no rito de defesa.

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

CURITIBA, 16 de Setembro de 2019.

MICHELE NOGUEIRA LIMA
Presidente da Banca Examinadora (UNIVERSIDADE FEDERAL DO PARANÁ)

CHRISTIAN ESTEVE ROTHENBERG
Avaliador Externo (UNIVERSIDADE ESTADUAL DE CAMPINAS)

ALDRI LUIZ DOS SANTOS
Avaliador Interno (UNIVERSIDADE FEDERAL DO PARANÁ)

MINISTÉRIO DA EDUCAÇÃO - UNIVERSIDADE FEDERAL DO PARANÁ - Setor de Ciências Exatas - Coordenação de Pós-Graduação em Informática

"Muda que quando a gente muda
O mundo muda com a gente
A gente muda o mundo na mudança da mente
E quando a mente muda a gente anda pra frente
E quando a gente manda ninguém manda na gente."
Gabriel O Pensador

AGRADECIMENTOS

À minha mãe Cida, por toda educação que me deu, pelos ensinamentos sobre generosidade e firmeza. Ao meu pai Paulo, que me ensinou sobre o valor do trabalho, valor de um amigo. Minha irmã Fernanda. Pessoa mais gentil e amável não há. Ao meu falecido irmão Marcos, que me deu força mesmo na ausência.

Ao Agnaldo, Benevid, Bruno, Fernando, Lígia e Mateus por todos os ensinamentos de vida, amizade, respeito, paciência e ajuda extra com os meus devaneios. Adicionalmente aos amigos Arthur, Carlos, Danilo e Gustavo pelos momentos de deselegância.

Aos amigos da universidade, pela convivência, pelas discussões e trocas de ideias.

Aos gatos Peter e Olivia.

À Prof. Dra. Michele Nogueira Lima pela oportunidade, confiança e ensinamentos.

À mim mesmo.

RESUMO

Os serviços Web evoluíram significativamente desde o seu início no final dos anos 80. Ao longo dos anos, estes serviços aumentaram em número, tamanho e complexidade. Além disso, novos serviços como aqueles de *streaming*, os jogos online e a realidade virtual demandam cada vez mais recursos devido às exigências dos usuários e a concorrência entre as plataformas que disponibilizam estes serviços. Em geral, estes serviços são sensíveis às falhas e à latência da rede. Garantir a entrega de pacotes de maneira confiável é uma das principais funções da camada de transporte. Contudo, os protocolos de transporte dominantes na Internet, como o protocolo TCP e UDP, possuem limitações e são difíceis de serem atualizados, justamente pela sua alta utilização. Em face disso, a empresa Google desenvolveu o protocolo de transporte QUIC (*Quick UDP Internet Connection*) e um grupo de trabalho do IETF trata da sua padronização. O protocolo QUIC é fundamentado no protocolo UDP e sua proposta é substituir o conjunto de protocolos da Web formado pelos protocolos HTTP/TLS/TCP. O protocolo QUIC traz benefícios ao reduzir a latência da conexão por se abster do processo de estabelecimento de conexão (*three-way handshake*) na maioria de suas conexões, além de aumentar a segurança durante a comunicação ao criptografar todos os seus pacotes e parte das informações contidas no cabeçalho de seus segmentos. Em pesquisas realizadas pela empresa Google, o protocolo QUIC diminui 3% o tempo médio de carregamento na página de pesquisa da empresa, quando comparado ao protocolo TCP. Contudo, estes valores tendem a crescer com o uso simultâneo das múltiplas interfaces de rede disponíveis nos dispositivos dos usuários. Visando a este objetivo, outro grupo de trabalho do IETF impulsiona o desenvolvimento do protocolo QUIC sobre multicaminhos. O protocolo QUIC Multicaminhos (MPQUIC) apresenta vantagem no uso das múltiplas interfaces de rede disponíveis com o intuito de aumentar o desempenho e a disponibilidade dos serviços. Entretanto, o uso de vários caminhos simultaneamente resulta em desafios relacionados à diversidade e à heterogeneidade da rede. Um desses desafios é a entrega de pacotes fora de ordem (ou o problema de reordenamento), que ocorre devido a estas adversidades. As abordagens na literatura destinadas a mitigação do problema de reordenamento, direcionam suas propostas ao escalonador de pacotes, que é um dos principais componentes dos protocolos multicaminhos e emprega um importante papel no desempenho da transmissão, pois ele é o responsável pela seleção e alocação de dados nos diferentes caminhos disponíveis para a comunicação entre cliente e servidor. Sendo assim, este trabalho apresenta o escalonador STOUT (do inglês, ***S**imilarity against**T** packets **OUT**-of-order*) que visa reduzir os efeitos da heterogeneidade dos caminhos ao selecioná-los baseando-se na similaridade de suas características. Em testes realizados, o escalonador STOUT demonstrou melhor desempenho quando comparado ao escalonador padrão do protocolo MPQUIC, além de diminuir o tempo em que os pacotes aguardam para reordenação.

Palavras-chave: Transmissão Multicaminhos, Escalonador, MPQUIC, QUIC, Reordenamento de Pacotes, Desempenho

ABSTRACT

Web services have evolved significantly since their conception in the late 1980s. Over the years, these services have increased in number, size and complexity. In addition, new services, such as streaming services, online games, and virtual reality, are increasingly demanding resources due to user requests and competition among the platforms that provide these services. In general, these services are sensitive to network failures and latency. Ensuring reliable delivery of packages is one of the key functions of the transport layer. However, dominant Internet transport protocols, such as the TCP and UDP protocol, have limitations and are difficult to be updated due to their high utilization. Given these facts, Google has developed Quick UDP Internet Connection (QUIC) transport protocol and an IETF working group addresses its standardization. The QUIC protocol is based on the UDP protocol and its purpose is to replace the web protocol set formed by the HTTP/TLS/TCP protocols. The QUIC protocol has benefits in reducing connection latency by refraining from the three-way-handshake process on most of your connections, as well as increasing security during communication by encrypting all your connections, packets and part of the information contained in the header of their segments. In researches by Google, the QUIC protocol decreases the average load time on the company's search page by 3% compared to the TCP protocol. However, these values tend to grow with the simultaneous use of multiple network interfaces available on user devices. Towards this end, another IETF working group drives the development of the multipath QUIC protocol. The multipath QUIC (MPQUIC) protocol has the advantage of using the multiple network interfaces available to increase performance and service availability. However, using multiple paths simultaneously results in challenges related to network diversity and heterogeneity. One of these challenges is the delivery of out-of-order packages (or the reordering problem), which occurs because of these adversities. Approaches in the literature aimed at mitigating the reordering problem directing their proposals to the packet scheduler, which is a major component of multipath protocols and plays an important role in transmission performance as it is responsible for data selection and allocation through different paths available for client-server communication. Thus, this work presents the scheduler named **S**imilarity agains**T** packets **OUT**-of-order (STOUT) which aims to reduce the effects of path heterogeneity by selecting them based on the similarity of their characteristics. In tests performed, the STOUT scheduler has shown better performance compared to the standard MPQUIC protocol scheduler, as well as decreasing the time packets wait for reordering.

Keywords: Multipath Transmission, Scheduler, MPQUIC, QUIC, Packet Reordering, Performance

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ACRÔNIMOS

3G	*Third-generation cellular networks*
	Terceira geração de redes celulares
ABW	*Available Bandwidth*
	Largura de Banda Disponível
ACK	*Acknowledgement*
	Reconhecimento ou Confirmação
API	*Application Programming Interface*
	Interface de Programação de Aplicações
BALIA	*Balanced Linked Adaptation Congestion Control Algorithm*
	Algoritmo de Controle de Congestionamento
	com Adaptação do Balanceamento Acoplado
CMT	*Concurrent Multipath Transfer*
	Transferência Multicaminhos Concorrente
CSS	*Cascading Style Sheets*
	Folha de Estilo em Cascatas
CWND	*Congestion Window*
	Janela de Congestionamento
DSN	*Data Sequence Number*
	Número de Sequência de Dados
FEC	*Forward Error Correction*
	Correção antecipada de erros
HTML	*HyperText Markup Language*
	Linguagem de Marcação de Hipertexto vspace-0.1cm
HTTP	*Hypertext Transfer Protocol*
	Protocolo de Transferência de Hipertexto
HOL	*Head-of-line blocking*
	Bloqueio de Cabeça de linha ou Bloqueio da fila de recebimento
IETF	*Internet Engineering Task Force*
	Força Tarefa ou Grupo de Trabalho para Engenharia da Internet
iOS	Iphone Operating System
	Sistema Operacional do Iphone
IP	*Internet Protocol*
	Protocolo de Internet
LIA	*Linked Increase Algorithm*
	Algoritmo de Incremento Acoplado

LRF	*Lowest-RTT-First*
	Primeiro o menor RTT
LTE	*Long Term Evolution*
	Evolução a Longo Prazo
MAC	*Medium Access Control*
	Controle de Acesso ao meio
MAC	*Message Authentication Code*
	Código de autenticação de menssagem
MP	*Multipath*
	Multicaminhos
MPLS	*Multiprotocol Label Switching*
	Comutação de Rótulos Multiprotocolo
MPQUIC	*MultiPath QUIC*
	QUIC Multicaminhos
MPTCP	*MultiPath TCP*
	TCP Multicaminhos
NAT	*Network Address Translation*
	Tradução de Endereço de Rede
OLIA	*Opportunistic Linked Increase Algorithm*
	Algoritmo de Incremento Acoplado Oportunístico
OWD	*One-Way Delay*
	Atraso de uma via
QUIC	*Quick UDP Internet Connection*
	Conexão Rápida de Internet UDP
RFC	*Request for Comment*
	Requisição para comentário
RMT	*Resilient Multipath Transfer*
	Transferência Multicaminho Resiliente
RR	*Round-Robin*
RTO	*Retransmission Timeout*
	Tempo Limite de Retransmissão
RTT	*Round Trip Time*
	Tempo de ida e volta
SACK	*Selective Acknowledgments*
	Reconhecimento Seletivo
SCTP	*Stream Control Transmission Protocol*
	Protocolo de Transmissão de Controle de Fluxo
SPTCP	*Single Path TCP*
	TCP de Caminho Único

SSL	*Secure Sockets Layer*
	Camada de Soquetes Seguro
SSN	*Subflow Sequence Number*
	Número de Sequência de Sub-fluxo
TCP	*Transmission Control Protocol*
	Protocolo de Controle de Transmissão
TFRC	*TCP-Friendly Rate Control*
	Controle de Taxa amigável com TCP
TLS	*Transport Layer Security*
	Segurança da Camada de Transporte
UDP	*User Datagram Protocol*
	Protocolo de Datagrama do Usuário
VoIP	*Voice over Internet Protocol*
	Voz Sobre o Protocolo de Internet
Wi-Fi	*Wireless Fidelity*
	Fidelidade Wireless

LISTA DE SÍMBOLOS

α	Controle de incremento para a janela de congestionamento
$\mathcal{W}$	Tamanho total da janela de congestionamento de todos os sub-fluxos
w	Tamanho da janela de congestionamento de um sub-fluxo
$\sum$	Somatório
$\mathcal{R}_u$	Conjunto de caminhos disponíveis
$\mathcal{M}$	Conjunto de caminhos com os maiores tamanhos de janelas de congestionamento disponíveis
$\mathcal{B}$	Conjunto de melhores caminhos possíveis
p ou r	Um único caminho
Δ	Diferença entre duas variáveis
ABW	Largura de banda disponível
QLT	Qualidade de um caminho
d	Vetor
L_p	Número de pacotes perdidos
P_{env}	Número de pacotes enviados
P_{rec}	Número de pacotes recebidos
T	Tempo total da conexão
T_i	Tempo inicial da conexão
T_f	Tempo final da conexão
T_s	Tempo total de armazenamento em buffer
T_{si}	Tempo inicial do armazenamento em buffer
T_{sf}	Tempo final do armazenamento em buffer

SUMÁRIO

1 INTRODUÇÃO

Cada vez mais a Internet suporta serviços que demandam transmissões de dados com alta vazão e baixa latência. Os jogos *online*, a realidade virtual e os serviços de *streaming*, como os oferecidos pela Netflix, Amazon Prime e HBO GO, são exemplos de serviços sensíveis ao atraso e são populares entre os usuários. A exemplo desta demanda e também da expectativa dos usuários em relação à qualidade do serviço ofertado, a empresa HBO Brasil foi recorde em reclamações na plataforma *Reclame Aqui*, atingindo quase 700 reclamações apenas no Domingo, 28 de Abril de 2019, devido a problemas de exibição da série *Game of Thrones* (TecMundo, 2019). O serviço de *streaming* da empresa sofre com a grande quantidade de acessos simultâneos e apresenta instabilidade durante as transmissões dos episódios do seriado.

Estas intermitências durante a comunicação entre o cliente e o servidor geram frustrações para os usuários consumidores daquele conteúdo e uma má reputação para a plataforma que fornece o serviço. Estudos sobre o uso de dispositivos móveis realizados pela empresa Google mostram que os usuários têm pouca paciência para aguardar o carregamento de páginas Web (Google, 2018b). Eles apontam que o tempo médio para carregar completamente uma página em dispositivos móveis utilizando conexões sem fio é de 15 segundos. Cerca de 70% das páginas levam mais de 5 segundos até que o navegador comece a renderização e 7 segundos para carregar todos os elementos visuais. Contudo, eles apontam que mais da metade dos usuários abandonam a página se ela demorar mais de 3 segundos para ser carregada.

Assim como a Internet, os serviços que ela disponibiliza têm evoluído e, com eles, os requisitos para o seu funcionamento. As empresas responsáveis pela distribuição de conteúdo sensíveis à latência expressam grande interesse no desenvolvimento de soluções para lidar com estas adversidades (De Coninck, 2017). A Google, que atingiu a marca de aproximadamente 42 bilhões de acessos à sua página inicial apenas no mês de Abril de 2019 (Reserved, 2018), tem direcionado esforços no desenvolvimento de soluções que acompanhem o crescimento da Internet. O protocolo SPDY[1] é um exemplo desse empenho (Belshe e Peon, 2012). Seu projeto iniciou em 2009 e a primeira proposta de padronização foi apresentada à IETF[1] (do inglês, *Internet Engineering Task Force*) em 2012 (Wang et al., 2014).

O protocolo SPDY tinha como objetivo superar limitações de desempenho do protocolo HTTP (do inglês, *Hypertext Transfer Protocol*), que é o principal protocolo utilizado pelas páginas Web. Esta limitação é atribuída primeiro ao cliente, que solicita apenas um recurso por vez, como arquivos HTML (do inglês, *HyperText Markup Language*), JavaScript, CSS (do inglês, *Cascading Style Sheets*), entre outros. Mesmo que esse problema possa ser parcialmente mitigado com o processo de *pipeline*, o qual possibilita que várias solicitações HTTP sejam enviadas em uma única conexão. Segundo, o servidor também é limitado pelo protocolo HTTP, dado que mesmo prevendo a necessidade de enviar um recurso adicional ao cliente, não há mecanismo para antecipar a emissão deste conteúdo, acelerando o processo, diminuindo o número de requisições, de reconhecimentos e, consequentemente, reduzindo a sobrecarga da rede de transmissão. Por fim, ocorre o envio redundante de informações sobre o mesmo canal de transmissão. Em geral, os navegadores Web tentam resolver esses problemas abrindo várias conexões HTTP simultâneas, mesmo que essa abordagem seja desencorajada na RFC (do inglês, *Request for Comments*) do protocolo HTTP. A RFC sugere um limite de conexões simultâneas entre o cliente e servidor de no máximo duas conexões (Carlucci et al., 2015).

[1]SPDY não é um acrônimo, mas sim um trocadilho com a palavra em inglês *speedy*, que significa veloz.

[1]A Força-Tarefa de Engenharia da Internet (IETF) é o principal órgão regulamentador da Internet.

O protocolo SPDY trouxe inovações para a comunicação entre cliente e servidor. Dentre elas, a multiplexação de solicitações HTTP sobre um único canal de transmissão, gerando uma economia de canais utilizados, compressão de cabeçalhos, antecipação do servidor Web no envio de dados ao cliente sempre que possível e a habilidade de priorizar requisições paralelas (Google, 2018a). Apesar do ganho com a economia de canais utilizados, o protocolo SPDY apresenta uma enorme desvantagem em relação ao HTTP. Ao ocorrer uma perda de dados durante uma transmissão, o algoritmo de controle de congestionamento do protocolo TCP (do inglês, *Transmission Control Protocol*) reduzirá a janela de congestionamento para todos os fluxos que foram multiplexados, prejudicando toda a conexão.

Outra desvantagem do protocolo SPDY é a entrega de pacotes fora de ordem, resultante de perdas durante uma conexão TCP. Isto ocasiona o problema conhecido como bloqueio de fila de recebimento (do inglês, *Head-of-Line blocking* - Bloqueio HoL). Este fenômeno ocasiona o estacionamento de todo o fluxo entre o cliente e servidor aguardando pela retransmissão do pacote perdido. Além disso, a latência de inicialização da conexão SPDY depende do processo de negociação do protocolo TCP, conhecido como aperto de mãos de três vias (do inglês, *three-way-handshake*), que aguarda um tempo de ida e volta (do inglês, *Round-Trip Time - RTT*) para ser completado. Quando a conexão utiliza protocolos de criptografia como o SSL (do inglês, *Secure Sockets Layer*) ou TLS (do inglês, *Transport Layer Security*), o estabelecimento da comunicação acrescenta até três RTTs para ser concluído.

Apesar dos esforços para o desenvolvimento do protocolo SPDY, o projeto foi descontinuado devido aos problemas supracitados. Muitos desses problemas não estão atrelados unicamente ao protocolo SPDY, mas também ao protocolo TCP, que foi desenvolvido no início da década de 70 (Li et al., 2016) e está presente na maior parte do tráfego da Internet, além de possuir uma enorme gama de plataformas programadas sobre a sua estrutura, o que dificulta qualquer possibilidade de alteração e/ou atualização do mesmo. Entretanto, as experiências obtidas com o protocolo SPDY foram otimizadas para a versão 2 do protocolo HTTP, padronizado pela RFC 7540 em 2015 (Belshe et al., 2015).

Em razão da complexidade de modificar um protocolo como o TCP e sua efetiva implementação, a Google desenvolveu seu próprio protocolo da camada de transporte, o protocolo QUIC (do inglês, *Quick UDP Internet Connection*). Ele convive com o ambiente legado e não impõe nenhuma alteração no nível de sistema operacional ou núcleo para ser implementado. O protocolo QUIC utiliza o protocolo UDP (do inglês, *User Datagram Protocol - UDP*) para transportar as mensagens HTTP na Internet. O protocolo QUIC está em uso desde 2013 pela empresa. Como a Google possui domínio dos dois lados da comunicação, isto é, fornece tanto a ferramenta de acesso para o cliente (e.g., Chrome), quanto os serviços utilizados (e.g., Gmail e YouTube), isto facilita o processo de implementação e estudo do seu comportamento no mundo real. O protocolo QUIC não altera a estrutura do protocolo UDP. Isto significa que para os dispositivos intermediários da conexão entre cliente e servidor, como os comutadores de rede, *firewalls*, servidores de *proxy* e NAT, entre outros, também chamados de caixas do meio (do inglês, *middleboxes*), não precisam de programação adicional para que o tráfego ocorra de maneira habitual. Em síntese, o protocolo QUIC é a carga útil (*payload*) transportada pelo protocolo UDP.

O principal objetivo do protocolo QUIC é reduzir a latência da rede. Estimativas apontam que ele é utilizado em cerca de 7% do tráfego global da Internet (Langley et al., 2017). Segundo Berkleley Lab. (2018), as interações com os serviços da Google obtiveram 5% de melhoria em seu desempenho e 30% menos recarregamento (do inglês, *rebuffering*) para os aplicativos de *streaming*, como o YouTube. Houve também uma redução de latência de 8% para as pesquisas realizadas pelo motor de busca da Google para usuários de computadores pessoais e

3,6% para usuários de dispositivos móveis. Esses resultados se devem à utilização do protocolo UDP, que não é orientado à conexão. O protocolo QUIC infere uma conexão mais eficiente, abstendo-se do *three-way-handshake* na maioria de suas conexões e ao mesmo tempo provê segurança com a utilização do protocolo TLS.

Assim como o protocolo QUIC, a maioria dos protocolos de transporte presentes na Internet atualmente realizam a comunicação fim-a-fim sobre um único caminho. Se por um lado as aplicações e serviços trazem um novo conjunto de preocupações e desafios em relação ao desempenho, estabilidade e disponibilidade da rede (Singh et al., 2015), por outro lado, os dispositivos dos usuários e os servidores de conteúdo possuem recursos consideráveis e ociosos (e.g., múltiplas interfaces de rede). O uso simultâneo de múltiplas interfaces aumenta o desempenho das conexões e auxilia os dispositivos a atender os requisitos que novos serviços possam introduzir.

1.1 MOTIVAÇÃO

A Internet foi projetada para admitir mais de um caminho entre dois hospedeiros de forma a prover maior confiabilidade e resiliência para a comunicação (Li et al., 2016). Mesmo com estas características presentes na Internet, o suporte ao uso de múltiplos caminhos não tem sido amplamente explorado. Isto se deve a dois motivos principais: (*i*) as regras de roteamento no núcleo da Internet e (*ii*) a dependência entre o protocolo TCP e o IP (do inglês, *Internet Protocol*). De acordo com as regras implementadas pela maioria dos protocolos de roteamento, os dados de uma conexão TCP/IP[1] são encaminhados apenas pelo melhor caminho (He e Rexford, 2008), evitando que os pacotes pertencentes ao mesmo fluxo TCP cheguem fora de ordem ao receptor devido às variações de atraso entre os caminhos. O recebimento de pacotes fora de ordem é um problema para a maioria das implementações do protocolo TCP, pois ele considera a entrega desordenada como um indício de congestionamento na rede e como contramedida, reduz a taxa de transferência dos dados (Van Wambeke et al., 2008).

O protocolo TCP e o protocolo IP são diferentes, mas a separação entre a camada de rede e a camada de transporte não é completa. Para diferenciar os fluxos de dados individuais entre os pacotes recebidos, um hospedeiro receptor demultiplexa os pacotes com base na chamada *4-tupla*, composta pelos endereços IP e números de portas da origem e do destino, mais o protocolo de transporte (TCP ou UDP, na maioria dos casos). Isso implica que uma conexão TCP sempre estará vinculada aos endereços IP usados pelo cliente e pelo servidor no momento do estabelecimento da conexão.

Mesmo com a crescente importância dos dispositivos móveis, como *smartphones* e *tablets*, as conexões TCP não são transferíveis de um endereço IP para outro. Quando um dispositivo muda sua conexão da rede de celulares (e.g., 3G e LTE) para a rede Wi-Fi, ele obtém outro endereço IP. Com isso, todas as conexões TCP existentes são encerradas e novas conexões são iniciadas (Paasch e Bonaventure, 2014). Para o protocolo QUIC, isto não é um problema. Ele abstrai essa dependência entre as duas camadas para identificar uma conexão e utiliza parâmetros próprios desenvolvidos para este fim, independente do serviço da camada de aplicação ou dos endereços IPs de origem/destino.

A Internet é composta por uma variedade de hospedeiros e enlaces de rede (*links)*. Na maioria dos casos, há mais de um caminho possível entre estes dispositivos. Utilizar mais de um caminho simultaneamente traz benefícios para a conexão em geral, pois o tráfego pode ser rapidamente redirecionado para um caminho alternativo, mantendo assim a conectividade de rede quando ocorrem falhas. Embora esta funcionalidade esteja disponível no nível de

[1]A pilha de protocolos TCP/IP é um conjunto de protocolos de comunicação entre dispositivos em rede, subdivido conceitualmente em camadas de acordo com a função que o protocolo exerce na comunicação fim-a-fim.

hardware para uma grande parte dos dispositivos de rede, ela não é explorada no nível de *software*, ou seja, os principais protocolos que provêm comunicação entre processos de hospedeiros, como o TCP e o UDP, e até mesmo o QUIC, operam suas transmissões apenas por um único caminho por vez (Domżał et al., 2015).

A transmissão de dados multicaminhos, ao nível da camada de transporte, visa quebrar estes paradigmas, conectando dois dispositivos através da Internet por múltiplas interfaces de rede e, consequentemente, por múltiplos caminhos (Domżał et al., 2015). O uso de múltiplos caminhos provê maior capacidade de transmissão e maior resiliência em relação à uma conexão TCP, que utiliza um caminho único. No entanto, apesar dos esforços da comunidade acadêmica em desenvolver novos protocolos ou mesmo otimizar os protocolos existentes, esta tarefa ainda é um desafio. Por ser implementado no núcleo dos sistemas operacionais de *middleboxes*, efetuar mudanças significativas no protocolo TCP é quase impossível (Projects, 2018).

Em sua proposta, o protocolo QUIC apresentava funções como a correção antecipada de erros (do inglês, *Forward Error Correction* - FEC) e uma iniciativa de desenvolvê-lo para transporte de dados utilizando multicaminhos. Ambos os mecanismos foram descontinuados pela Google e não seguiram na especificação proposta para padronização (*IETF Draft*) apresentada por J. Iyengar e M. Thomson (2019). No entanto, um grupo de trabalho já mobiliza esforços para a padronização de sua versão para multicaminhos. O protocolo QUIC Multicaminhos (do inglês, *MultiPath QUIC - MPQUIC*) apresentou melhor desempenho em testes preliminares quando comparado com o protocolo TCP Multicaminhos (do inglês, *MultiPath TCP - MPTCP*), protocolo de transporte multicaminhos já consolidado e padronizado pela IETF (De Coninck, 2017).

A popularização de dispositivos móveis e a expansão das aplicações sensíveis à latência criam desafios sem precedentes para os provedores de serviço. Não apenas relacionados ao desempenho, mas também em relação ao armazenamento de dados e a sua segurança. O uso de protocolos multicaminhos é promissor e contribui para um melhor desempenho quando comparado com os protocolos de caminho único. O protocolo MPTCP já está em uso comercialmente e, a título de exemplo, a empresa Apple o utiliza em sua assistente digital *Siri*, além de ter acrescentado suporte nativo ao protocolo para o seu sistema operacional (*iOS7*). O protocolo MPQUIC é relativamente mais novo que o protocolo MPTCP, porém ambos os protocolos possuem desafios a serem superados. Contudo, como o protocolo MPTCP já possui grande atividade e pesquisa na comunidade científica, esta dissertação fomenta o desenvolvimento do protocolo MPQUIC.

1.2 PROBLEMA

Os dispositivos móveis, em geral, possuem mais de uma interface de rede e utilizam redes sem fio e/ou de celulares para comunicação com a Internet (Li et al., 2016; Kuhn et al., 2014; Ke et al., 2016; Frömmgen et al., 2018). As redes celulares diferem das redes de comunicação sem fio na medida em que fornecem cobertura de sinal mais ampla e conectividade mais confiável sob mobilidade (Chen et al., 2013). Os cenários com mobilidade que utilizam os meios de conexões não guiados, LTE e Wi-Fi por exemplo, vivenciam maiores problemas causados por perdas e retransmissão de pacotes devido às atenuações sofridas pelas ondas de rádio, utilizadas para transmitir os sinais. Além disso, as perdas de pacotes resultam em uma diminuição na taxa de transferência e são consideradas como congestionamentos de rede pelo protocolo TCP.

A comunicação multicaminhos apresenta desafios devido à heterogeneidade e à diversidade de caminhos (Alheid et al., 2016), especialmente para os cenários com mobilidade, problemas relacionados à utilização do canal de forma justa com outros fluxos (do inglês, *fairness*) (Becke et al., 2012), restrições de energia (Kaup et al., 2015), custo monetário de

transmissão (Secci et al., 2014), problemas de segurança (Pearce e Zeadally, 2015), entre outros. Os caminhos heterogêneos degradam o desempenho da transmissão multicaminhos por causa de perdas e pela chegada de pacotes fora de ordem (e.g., problema do reordenamento). O problema do reordenamento aumenta o atraso e reduz a vazão da transmissão multicaminhos, afetando, principalmente, aplicações sensíveis ao atraso, como jogos online, transmissões em tempo real (áudio e vídeo) e tráfego Web (Yedugundla et al., 2016). Prover uma transmissão multicaminhos para evitar a degradação do serviço de transporte, quando os caminhos são heterogêneos, tem sido um grande desafio abordado em vários trabalhos e que ainda está em aberto (Li et al., 2016).

Um escalonador de pacotes é necessário para a utilização de protocolos multicaminhos. Sua responsabilidade é decidir qual caminho será utilizado para encaminhar os pacotes de dados (Le e Bui, 2018). Além disso, ele ainda precisa entregar esses pacotes para a camada de aplicação do receptor na mesma ordem em que foram transmitidos pelo remetente. Devido à heterogeneidade dos caminhos, os dados podem chegar fora de ordem no receptor e precisar de reordenamento (Le e Bui, 2018). Essa situação ocasiona o bloqueio HoL, que estaciona todo o tráfego subsequente à espera do pacote faltante. Este problema é bastante comum para o tráfego Web sobre o TCP, mas é reduzido com o QUIC, devido ao seu sistema de multiplexação de dados. No processo de multiplexação do QUIC, o tráfego é dividido em *streams*. Um *stream* em específico pode sofrer com o problema de reordenamento, mas não afetará os outros *streams*, pois estes serão demultiplexados e entregues à aplicação de forma independente.

A transmissão multicaminhos acrescenta outros níveis de resiliência para o transporte fim-a-fim sobre a Internet, especialmente para a transmissão de dados feita de forma confiável proporcionada pelos protocolos orientados à conexão, sendo que o escalonador tem grande importância para a resiliência da transmissão multicaminhos. As etapas do processo de escalonamento envolvem: *(i)* a coleta de informações sobre os caminhos, *(ii)* a seleção do caminho para a transmissão, e (*iii*) a alocação dos dados pelos caminhos selecionados. Elas agregam funções que auxilia o protocolo a prover uma maior resiliência para a transmissão, além de incluírem métricas utilizadas para identificar o estado desses caminhos (Paasch et al., 2014), calcular a correlação entre eles (Hu et al., 2016) e estimar suas diferenças (Coudron et al., 2015), entre outros. Embora estas funções possam prover uma maior resiliência, elas têm sido pouco exploradas na literatura (Li et al., 2016).

O protocolo MPQUIC é uma evolução do protocolo QUIC. Sua principal diferença em relação ao protocolo QUIC é o uso de múltiplas interfaces simultaneamente (De Coninck e Bonaventure, 2019). Além disso, o protocolo MPQUIC agrega mecanismos de controle de fluxo e gerência de caminhos do protocolo MPTCP. Observando as características do protocolo MPTCP, dois componentes de grande importância foram incorporados na integra ao protocolo MPQUIC: (*i*) o algoritmo de controle de congestionamento e *(ii)* o escalonador de pacotes.

Apesar de ambos os protocolos utilizarem múltiplos caminhos paralelamente, eles diferem em sua arquitetura. O protocolo MPTCP é baseado no protocolo TCP e os protocolos QUIC e MPQUIC aplicam novas funções sobre o protocolo UDP, mantendo sua simplicidade. Segundo o trabalho De Coninck (2017), o protocolo MPQUIC lida melhor com os caminhos mais lentos, devido à estimativa mais precisa do atraso de um caminho comparado com o MPTCP e evita a limitação de carregamento (do inglês, *buffer*), que se deve à transmissão de quadros de controle em todos os caminhos utilizados. Ainda no trabalho de De Coninck (2017), os autores alegam que sem perdas aleatórias, as retransmissões são raras.

O protocolo MPQUIC possui informações mais precisas sobre o estado dos caminhos devido ao seu sistema de confirmação (do inglês, *Acknowledgement*) por pacote. No entanto, ele não está imune ao problema de reordenamento, resultante de fatores como perdas de pacotes, densidade da rede, heterogeneidade dos caminhos, entre outros. De forma geral, a

comunidade científica tem aplicado contramedidas para o problema otimizando as políticas de escalonamento. A atual política do protocolo MPQUIC é baseada exclusivamente no atraso observado de cada caminho (RTT) e no tamanho de sua janela de congestionamento, mesmo obtendo um conjunto maior de informações que poderiam ser utilizados para aumentar o desempenho da transmissão como a largura de banda e a taxa de perda de pacotes. Outro problema é a disparidade dessas características, uma vez que, quanto mais homogêneos os caminhos, maior a probabilidade de os dados chegarem em ordem ao seu destino. Contudo, os caminhos com características homogêneas e que não compartilham o mesmo enlace (disjuntos) entre os hospedeiros finais são raros (Li et al., 2016).

1.3 OBJETIVOS

O objetivo principal deste trabalho é reduzir a latência durante a transferência de dados na Web utilizando o protocolo MPQUIC. O desenvolvimento de protocolos multicaminhos têm atraído a atenção da comunidade científica. Além disso, grandes empresas como a Apple e a Google demonstram interesse no desenvolvimento de soluções para reduzir a latência e o tempo de resposta para seus serviços. Neste sentido, o protocolo MPQUIC proporciona benefícios ao utilizar múltiplos caminhos em conjunto de um protocolo simples e leve como o UDP. Contudo, apesar dos protocolos multicaminhos demonstrarem melhor desempenho para a conexão quando comparado aos protocolos que utilizam um único caminho, eles ainda possuem desafios. O reordenamento de pacotes representa um dos principais desafios deste tipo de comunicação e geralmente são causados devido à heterogeneidade dos caminhos utilizados. O escalonador de pacotes possui grande importância para os protocolos multicaminhos, pois ele é responsável pela seleção dos caminhos que serão utilizados durante a transmissão. Desta forma, este trabalho investiga a hipótese de escalonar os dados de acordo com a similaridade dos caminhos, como uma forma de reduzir o número de pacotes entregues fora de ordem.

1.4 CONTRIBUIÇÕES

A contribuição central deste trabalho corresponde ao desenvolvimento de um escalonador de pacotes para o protocolo MPQUIC. O escalonador STOUT (do inglês, ***S**imilarity agains**T** packets **OUT**-of-order*) baseia a tomada de decisão na tarefa de selecionar os caminhos para a transmissão de dados na similaridade entre as características dos caminhos disponíveis. Este trabalho também realizou um estudo sobre o protocolo MPQUIC e suas propriedades. Visto que o protocolo MPQUIC proporciona o uso de múltiplos caminhos simultaneamente e que dois de seus componentes principais (o controle de congestionamento e o escalonador de pacotes) são heranças do protocolo MPTCP o desenvolvimento desses componentes visando as características do protocolo MPQUIC torna-se chave para o seu progresso. Assim, este trabalho buscou na literatura os trabalhos que lidam com o problema de reordenamento de pacotes e sua relação com o escalonador para a comunicação multicaminhos. Além disso, este trabalho também implementa o escalonador ECF (do inglês, *Earliest Completion First*) no protocolo MPQUIC para fins comparativos.

1.5 ESTRUTURA DO TRABALHO

Este manuscrito está organizado em cinco capítulos. O Capítulo 2 apresenta os fundamentos da camada de transporte, de seus protocolos de caminho único e de suas variações que possibilitam a utilização de múltiplos caminhos simultaneamente. Em seguida, o Capítulo 3

apresenta uma revisão da literatura sobre as estratégias que vêm sendo empregadas junto aos métodos de escalonamento para lidar com o problema de reordenamento de pacotes. O Capítulo 4 descreve o escalonador de pacotes STOUT, proposto neste trabalho. O Capítulo 5 apresenta uma avaliação do escalonador STOUT e compara-o com outras propostas de políticas de escalonamento. O Capítulo 6 conclui este trabalho, apresentando as considerações finais e direções futuras.

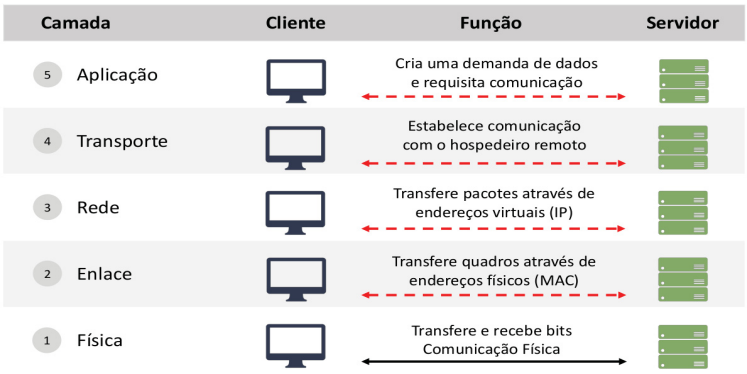


Figura 2.1: Pilha de protocolos TCP/IP

2 FUNDAMENTOS

Este capítulo relembra conceitos dos protocolos bem conhecidos da camada de transporte. Ele também explica os conceitos sobre os protocolos de transporte multicaminhos. Este conhecimento prévio é de extrema importância para dissertação sobre os novos protocolos QUIC e Multicaminhos QUIC. O protocolo QUIC Multicaminhos oferece ao protocolo QUIC a possibilidade de transmitir dados por múltiplos caminhos de forma simultânea. O protocolo QUIC, por sua vez, é um protocolo intercamadas executado pela camada de aplicação, encapsulado pelo protocolo UDP e que desempenha uma função similar ao principal conjunto de protocolos utilizados para transmissão de dados na Web. Este conjunto é formado pelos protocolos HTTP, TLS e TCP. Dessa forma, esses últimos protocolos também são revistos neste capítulo para fins de comparação. O capítulo está organizado da seguinte forma. A Seção 2.1 expõe os conceitos de protocolos tradicionais da camada de transporte, iniciando-se pelo protocolo TCP. Em seguida, os protocolos UDP e TLS são conceituados, pois são protocolos chaves no desenvolvimento do protocolo QUIC. A Seção 2.2 apresenta uma descrição de transmissões de dados por múltiplos caminhos e dos protocolos TCP Multicaminhos e QUIC Multicaminhos. Por fim, a Seção 2.3 resume e conclui o capítulo.

2.1 PROTOCOLOS DE TRANSPORTE

Os protocolos de transporte gerenciam a comunicação de processos fim-a-fim entre hospedeiros e residem na camada de transporte, a quarta camada da pilha TCP/IP. Seguindo o modelo cliente-servidor, sua função básica, na perspectiva do cliente, é receber dados da camada de aplicação, dividi-los em segmentos, passar estes segmentos para a camada de rede e garantir que todos eles cheguem corretamente na outra extremidade. Na perspectiva do servidor, quando a camada de rede lhe entrega os pacotes do cliente, eles são reagrupados em segmentos e passados para a camada de aplicação. A Figura 2.1 ilustra a pilha de protocolos TCP/IP, composta pelas camadas de aplicação (mais próxima do usuário), transporte, rede, enlace e física, e exemplifica a função que cada camada executa durante a comunicação entre cliente e servidor.

Dentre os protocolos da camada de transporte, o protocolo TCP e o protocolo UDP são os mais utilizados na comunicação cliente-servidor. O protocolo TCP é orientado à conexão, fornece entrega confiável, ordenada e com verificação de erros. As principais aplicações da Internet, como a Web, correio eletrônico, administração remota e transferência de arquivos

dependem do protocolo TCP. Apesar da sua confiabilidade, ele pode ser inadequado para algumas aplicações que não necessitam do controle exercido por ele. As aplicações que não exigem um serviço de fluxo de dados confiável podem usar o protocolo UDP, conveniente para as operações em que a verificação e a correção de erros não são necessárias ou são executadas na aplicação. Os jogos *online*, algumas plataformas de *streaming* de vídeo e a transmissão de voz sobre o protocolo de Internet (do inglês, *Voice over Internet Protocol - VoIP*) são exemplos de serviços que utilizam o protocolo UDP.

2.1.1 Protocolo TCP

O protocolo TCP é orientado à conexão e oferece entrega confiável dos dados. Isto significa que, para ocorrer a comunicação entre aplicações, primeiramente é necessário um consenso entre os dispositivos comunicantes sobre os parâmetros de transferência de dados. Esta negociação de parâmetros é chamada de aperto de mãos de três vias (do inglês, *three-way handshake)*. Quando um cliente solicita o estabelecimento de uma conexão TCP, ele deve informar qual a porta do serviço no campo porta de destino e para a porta de origem, usada para informar ao servidor em qual porta ele deve enviar a resposta. As portas lógicas bem conhecidas identificam o tipo de serviço solicitado, por exemplo, a porta 25 é utilizada pelo protocolo simples de envio de correio eletrônico (do inglês, *Simple Mail Transfer Protocol - SMTP*), a porta 80 é utilizada pelo protocolo HTTP para transferência de páginas Web. Existe um total de 65536 portas disponíveis, entretanto as portas bem conhecidas compreendem as portas de 0 a 1023 e são mantidas pela autoridade de atribuição de números de Internet (do inglês, *Internet Assigned Numbers Authority - IANA*[1]). Além disso, vários outros mecanismos são utilizados para a transferência confiável, tais como:

- **Soma de verificação:** detecta possíveis erros na transmissão de cada bit transmitido;
- **Temporizador:** controla a temporização/retransmissão de um pacote, possivelmente porque o pacote ou seu reconhecimento foi perdido durante a transmissão;
- **Número de sequência:** controla os pacotes que trafegam entre os hospedeiros. As lacunas na sequência recebida permitem que o destinatário identifique um pacote perdido;
- **Confirmação/reconhecimento:** recurso utilizado pelo destinatário da conexão para confirmar o recebimento correto de um pacote, ou conjunto de pacotes, ao remetente.
- **Confirmação/reconhecimento negativo:** recurso utilizado pelo destinatário para informar ao remetente que um pacote não foi recebido corretamente. O reconhecimento negativo normalmente informa o número de sequência do pacote que não foi entregue de forma correta;
- **Controle de fluxo:** mecanismo utilizado para evitar que o emissor não envie mais dados do que o receptor possa processar. O destinatário estipula o tamanho máximo da janela de recebimento de acordo com o tamanho e espaço livre em seu *buffer* de recepção, também chamado de tamanho máximo de segmento (do inglês, *Maximum Segment Size - MSS*); e
- **Controle de congestionamento:** minimiza o congestionamento no núcleo da rede de acordo com a observação do número de pacotes perdidos ou recebidos com atraso, fora do período estipulado pelo temporizador.

[1]IANA: Organização mundial que supervisiona a atribuição global dos números na Internet.

Os segmentos TCP possuem um cabeçalho com tamanho total de 32 bits por linha, desconsiderando a carga (*payload*) que ele transmite. Contudo, a quantidade de linhas pode variar de acordo com a adição de funções extras, que são especificadas no campo de opções do cabeçalho. Tipicamente, o cabeçalho de um segmento TCP possui as cinco primeiras linhas, totalizando 20 bytes. A Figura 2.2 apresenta um exemplo de segmento TCP.

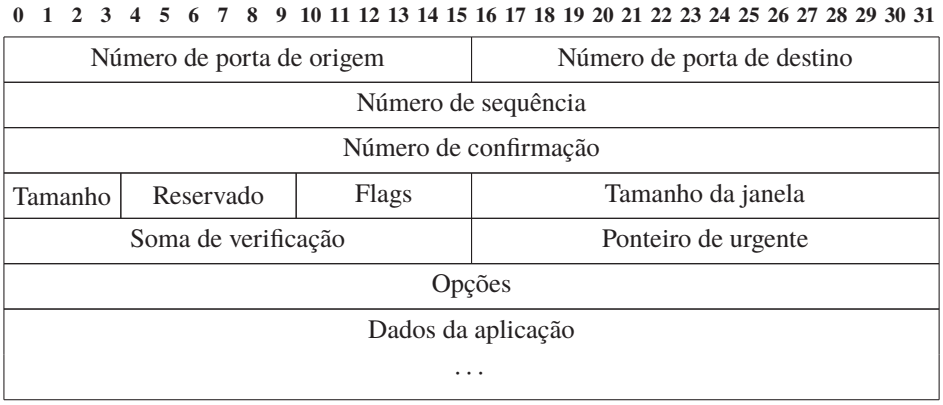


Figura 2.2: Cabeçalho do segmento TCP

2.1.1.1 Estabelecendo uma conexão TCP

Antes de uma aplicação cliente iniciar a troca de dados com uma aplicação servidora utilizando o protocolo TCP, o processo de *handshake* precisa ser efetuado. Durante este processo, o protocolo TCP inicia um conjunto de variáveis como o número de sequência e o tamanho do *buffer*, que o auxiliam na garantia de entrega e no controle de fluxo da transmissão. O *three-way handshake* executa três operações antes de permitir o envio de requisições de uma aplicação.

O protocolo TCP utiliza o campo bandeira (do inglês, *flag*) em seu cabeçalho para informar ao remetente que o dado transmitido precisa de tratamento suplementar. Para o *three-way handshake*, duas *flags* são utilizadas: a *flag* SYN e a *flag* ACK. A Figura 2.3 ilustra o processo de estabelecimento de conexão entre um cliente e um servidor. Inicialmente, o cliente envia uma mensagem ao servidor contendo um número de sequência inicial escolhido aleatoriamente por ele e o bit da *flag* SYN ativo. O servidor recebe a mensagem, aloca *buffers* e inicia suas variáveis pertinentes à conexão com o cliente. Em seguida, ele envia como resposta uma mensagem contendo as *flags* SYN e ACK ativas, para finalizar o processo de sincronização e confirmar o número de sequência inicial do cliente, e o número de sequência iniciado por ele próprio. Por fim, o cliente recebe o SYN/ACK do servidor. A partir desse momento, ele contabiliza o RTT, reserva seus próprios *buffers* e variáveis, e responde com uma mensagem ACK para confirmar o número de sequência inicial do servidor.

O *three-way handshake* é efetuado em todas as conexões TCP, mesmo que uma aplicação cliente perca a conexão temporariamente com a aplicação servidora. Isto porque uma conexão TCP está atrelada ao protocolo de Internet (do inglês, *Internet Protocol - IP*). O protocolo TCP utiliza o conjunto formado pelos endereços IPs de origem e destino, além das portas TCP de origem e destino, para identificar uma conexão, formando uma tupla de 4 elementos. O protocolo IP é um protocolo da camada de rede, que possui a tarefa de entregar os pacotes de um hospedeiro de origem para um hospedeiro de destino, exclusivamente com base nos endereços IPs inseridos no cabeçalho de seus pacotes. Quando um cliente solicita uma página Web, por exemplo, ele informa ao navegador Web a URL do servidor (e.g., `www.google.com/`). Porém, este nome

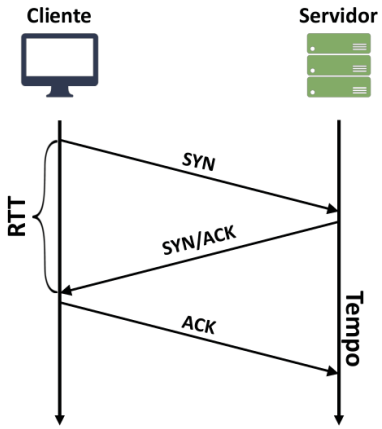


Figura 2.3: Handshake do protocolo TCP

é traduzido para o endereço IP (e.g., `172.217.28.142`) utilizado na Internet para localizar este hospedeiro. O sistema de nomes de domínio (do inglês, *Domain Name System - DNS*) é o responsável por essa tradução. Assim, sempre que uma conexão TCP é iniciada, ela é atrelada ao endereço IP dos hospedeiros (origem e destino) e às portas TCP (origem e destino).

2.1.1.2 Transferência de dados

O protocolo TCP precisa garantir que os dados da aplicação serão entregues ao receptor após o estabelecimento da conexão. Ele deve garantir, também, que essa entrega será feita de forma ordenada, ou seja, na mesma sequência em que os dados foram enviados pelo emissor. Além disso, o protocolo TCP deve garantir que os dados não estão corrompidos, que não há lacunas entre eles e que não exista duplicações. Na medida em que os dados da aplicação emissora são entregues à camada de transporte, o protocolo TCP aloca-os em um *buffer* de envio, processa-os e constrói seus segmentos de acordo com o tamanho estipulado pelo campo MSS. Assim que os dados chegam ao receptor, eles são alocados no *buffer* de recepção e ficam disponíveis para o processamento pela aplicação. O receptor confirma os segmentos indicando que recebeu um determinado número de bytes adjacentes. O sistema de reconhecimento do protocolo TCP pode fazer o reconhecimento por segmento recebido ou por um conjunto de segmentos (blocos). A confirmação por blocos também possibilita reconhecer segmentos fora de ordem. Este mecanismo é chamado de reconhecimento seletivo (do inglês, *Selective ACK* - SACK).

2.1.1.3 Controle de congestionamento

O protocolo TCP inicializa um temporizador visando aguardar a chegada da confirmação do seu recebimento pelo receptor sempre que um segmento é enviado. Caso a confirmação não chegue até o fim do temporizador (*timeout*), o protocolo reenvia o segmento. O valor do temporizador é sempre maior que o RTT (do inglês, *Round-Trip Time - RTT*) de um pacote, isto é, o tempo transcorrido entre o momento em que um segmento é enviado para a camada de rede e o recebimento de sua confirmação. Em geral, o protocolo TCP faz uma estimativa dinâmica do RTT para determinar o valor do temporizador. A partir desta estimativa, o controle de congestionamento calcula o valor médio do RTT e seu desvio, aplica um limiar de segurança para que o temporizador não assuma um valor muito baixo, gerando retransmissões desnecessárias, ou muito alto, aumentando o tempo de reação a uma possível falha na transmissão.

O controle de congestionamento do protocolo TCP identifica um congestionamento na rede de duas formas: após a ocorrência de um *timeout* ou após o recebimento de três ACKs con-

tendo o mesmo número de sequência do último segmento enviado (ACK duplicados). O controle de congestionamento do protocolo TCP previne problemas causados por congestionamentos na rede ou contribui para mitigá-los depois que eles ocorrem. Os algoritmos de controle de congestionamento do protocolo TCP usam a janela de congestionamento (do inglês, *congestion window* - `cwnd`) ao contrário do mecanismo de controle de fluxo, que o utiliza o MSS. Enquanto o MSS está presente no cabeçalho dos segmentos TCP, a *cwnd* é mantida apenas pelo remetente da conexão. Ela é monitorada para cada conexão TCP e representa a quantidade máxima de dados que podem ser enviadas para o receptor sem serem reconhecidos. Dessa forma, o desempenho da conexão é acrescido, pois o protocolo TCP não precisa esperar a chegada de um ACK para enviar um novo segmento. Em geral, a `cwnd` é iniciada com o valor de um MSS e incrementado com um MSS após o recebimento de cada ACK seguindo a fórmula: `cwnd` = `cwnd` + MSS. O remetente aumenta o tamanho da `cwnd` exponencialmente e esta fase é conhecida como **partida lenta**.

O controle de congestionamento limita a fase de partida lenta. O aumento exponencial da `cwnd` é suspenso a fim de evitar que muitos segmentos sejam transmitidos sem serem reconhecidos, o que pode gerar um congestionamento na rede. Para isso, a variável de estado `ssthresh` (do inglês, *slow start threshold*) é estabelecida em `cwnd`/2, ou seja, metade do tamanho da `cwnd` quando o congestionamento foi detectado. Se um *timeout* ocorrer nessa fase, a `cwnd` é reduzida para um MSS e o valor de `sstrhesh` atualizado. Dessa forma, a partida lenta é finalizada quando o valor da `cwnd` se igualar ao de `sstrhresh`.

Na fase seguinte, o protocolo TCP entra em modo de **prevenção de congestionamento**. Nesta fase o protocolo TCP adota um método mais conservador e aumenta em um único MSS a cada RTT. Caso ocorra um *timeout* nessa fase, o comportamento do algoritmo de controle de congestionamento é idêntico ao adotado na fase de partida lenta, reduzindo a `cwnd` para 1 MSS e atualizando o valor de `ssthresh`. Porém, se a causa do congestionamento for dada pelo recebimento de três ACKs duplicados, o algoritmo reduz a `cwnd` pela metade e atualiza o valor de `ssthresh`.

Para o protocolo TCP, a chegada de ACKs duplicados pode ocorrer devido à indicação de perdas, ao reordenamento dos pacotes no receptor e a replicação dos dados na rede. Assim, o protocolo TCP entra no estado de **recuperação rápida**. Quando o transmissor recebe três ACK duplicados, ele deve retransmitir o segmento pedido (retransmissão rápida) e acionar o algoritmo de recuperação rápida. Neste algoritmo, o valor de `cwnd` aumenta em um MSS para cada ACK duplicado recebido do segmento perdido. Se o ACK de confirmação do segmento retransmitido chegar corretamente no emissor, o protocolo TCP volta para o estado de prevenção de congestionamento. Se ele não chegar, o protocolo TCP aguarda no modo de recuperação até expirar o temporizador RTO (do inglês, *Retransmission Timeout*), que define o tempo máximo aguardado pelo ACK de uma retransmissão.

2.1.1.4 Finalizando uma conexão TCP

Durante uma conexão TCP, o cliente pode solicitar o encerramento da conexão assim como o servidor. Para isto, supondo que o cliente decida encerra-la, a aplicação envia um comando de encerramento ao protocolo TCP e ele envia um segmento com a *flag* FIN ativada ao servidor. O servidor recebe este segmento e envia um segmento ACK de confirmação ao cliente. Em seguida, o próprio servidor envia uma mensagem contendo a *flag* FIN ativada ao cliente. Por fim, o cliente reconhece a mensagem FIN do servidor, encerrando a conexão por definitivo.

2.1.2 Protocolo UDP

O protocolo UDP é um protocolo simples da camada de transporte. Ele possui um cabeçalho de 8 bytes composto apenas pelos campos que identificam as portas de origem e de destino, o tamanho do cabeçalho, a soma de verificação, em adição à carga que ele transporta. Uma representação do segmento, também chamado de datagrama, pode ser observado na Figura 2.4. Apesar de possuir o campo soma de verificação, o protocolo UDP apenas realiza a detecção de erros, mas não possui mecanismo para corrigi-los. Ele também não possui um processo de negociação inicial entre os hospedeiros, controle de congestionamento ou de fluxo. Contudo, estas funções de controle e a correções de erros podem ser realizadas em outras camadas, geralmente pela própria aplicação que escolhe o protocolo UDP para transmissão.

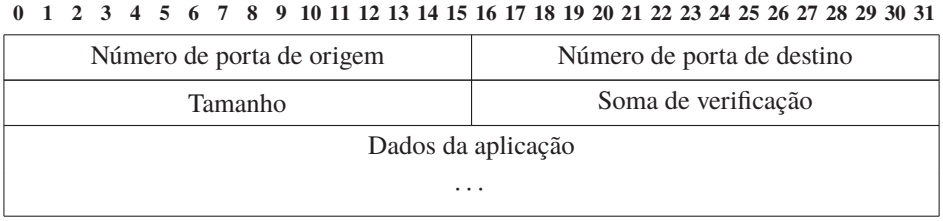


Figura 2.4: Cabeçalho do segmento UDP

2.1.3 Protocolo TLS

O protocolo seguro da camada de transporte (do inglês, *Transport Layer Security - TLS*), apesar do nome referenciar a quarta camada da pilha TCP/IP, é um protocolo da camada de aplicação e seu principal objetivo é fornecer um canal seguro para a comunicação entre os serviços na Internet. O protocolo TLS tem aplicações em vários serviços, como a navegação Web, correio eletrônico, mensagens instantâneas e VoIP. Contudo, seu único requisito consiste na presença de um protocolo orientado à conexão na camada de transporte. O protocolo QUIC, que será detalhado nas próximas seções, implementa uma versão do protocolo TLS para garantir a segurança durante a troca de dados entre cliente e servidor. Assim, de acordo com sua RFC (Rescorla, 2018), o protocolo TLS está fundamentado em duas funções principais:

- **Handshake:** Esta função é utilizada para autenticar os dispositivos que irão se comunicar, negociar o algoritmo de criptografia e parâmetros relacionados à troca de mensagens, além de executar uma troca de chaves secretas, utilizadas pelo protocolo de registro; e
- **Registro:** Utiliza as informações coletadas durante o momento de *handshake* para proteger a troca de mensagens entre os dispositivos. Ele também divide o tráfego em uma série de registros, cada um deles está protegido de forma independente, usando as chaves criptográficas trocadas durante a função de *handshake*.

O *handshake* do protocolo TLS é independente do *handshake* efetuado pelo protocolo TCP. A Figura 2.5 ilustra o *handshake* do protocolo TLS de acordo com a versão 1.3 (Rescorla, 2018). A comunicação é iniciada pelo cliente com uma mensagem `ClientHello` (*passo 1*). Esta mensagem contém uma chave pública e assimétrica, gerada através de um algoritmo de troca de chaves, além de informações sobre os algoritmos de criptografia e sobre os métodos de compressão de cabeçalho de mensagens suportados pelo cliente. Ao receber a mensagem, o servidor gera uma chave pública e assimétrica para aquela sessão, seleciona um dos algoritmos

de criptografias informados pelo cliente e envia uma mensagem `ServerHello` para o cliente. Esta mensagem contém a indicação do algoritmo selecionado, a chave pública do servidor e um certificado digital, para confirmar sua identidade para o cliente (*passo 2*). Quando o cliente recebe esta mensagem e extrai seu conteúdo, ele verifica o certificado digital informado pelo servidor, seleciona o algoritmo de criptografia, gera uma chave secreta utilizando a chave pública enviada pelo servidor e a envia, solicitando o encerramento do processo de *handshake* com a mensagem `ClientKeyExchange` (*passo 3*). Por fim, o servidor reconhece esta mensagem e só então, a aplicação cliente inicia o envio de suas requisições.

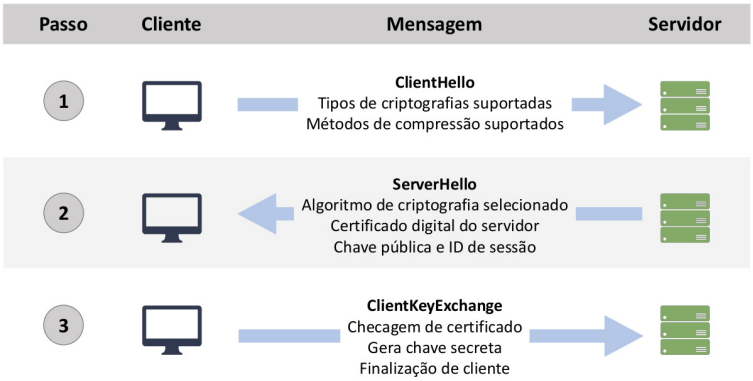


Figura 2.5: Handshake do protocolo TLS

O protocolo TLS foi projetado para prover três serviços essenciais para as aplicações que o utilizam: criptografia, autenticação e integridade. Durante o *handshake*, o protocolo TLS faz uso da criptografia de chave pública (também conhecida como criptografia de chave assimétrica), que permite aos hospedeiros negociar uma chave secreta compartilhada sem precisar estabelecer um conhecimento prévio um do outro. A criptografia de chave pública geralmente usa algoritmos como o RSA ou Diffie Hellman. Além disso, para garantir a autenticidade, o protocolo TLS também permite que os dois hospedeiros autentiquem suas identidades utilizando certificados digitais. Quando usado por um navegador Web, esse mecanismo de autenticação permite que o cliente verifique se o servidor é quem ele alega ser, evitando a falsificação das entidades. Finalmente, com a garantia de criptografia e a autenticação, o protocolo TLS também fornece seu próprio mecanismo de troca de mensagens e assina cada mensagem com um código de autenticação de mensagem (do inglês, *Message Authentication Code* - MAC). Sempre que um registro TLS é enviado, um valor MAC é gerado e anexado a essa mensagem, e o receptor pode então calcular e verificar o valor MAC enviado para garantir a integridade e autenticidade da mensagem.

A RFC do protocolo TLS define que a autenticação deve sempre ser executada pelo servidor (Rescorla, 2018). A autenticação do lado do cliente é opcional. Com o sistema de autenticação baseado em certificados digitais, a identidade dos hospedeiros é mantida em segredo. Em geral, utilizam-se algoritmos de troca de chaves assimétricas nesta fase. Após o processo de autenticação, chaves simétricas são utilizadas para criptografar as mensagens transmitidas entre o par (cliente/servidor). Com isto, o protocolo TLS consegue garantir a confidencialidade das mensagens, isto é, garantir que apenas o cliente e/ou o servidor serão capazes de ler e interpretar estas mensagens. Mesmo que elas sejam interceptadas por um atacante durante a transmissão, sem as chaves utilizadas para criptografar/decriptografar as mensagens não poderão ser lidas.

2.1.4 Protocolo QUIC

A empresa Google tem direcionado esforços no desenvolvimento de soluções que aumente o desempenho de serviços Web. Em 2009, ela desenvolveu o protocolo SPDY, o qual

foi inicialmente implementado nos seus navegadores Chrome e Chromium. Posteriormente, uma extensão foi desenvolvida para o navegador Firefox. O protocolo SPDY era uma alternativa para o protocolo HTTP e o seu propósito constituía em reduzir a latência no carregamento de páginas Web (Google, 2018a). Inicialmente, com o protocolo HTTP em sua versão 1.0, o cliente poderia transferir um único objeto do servidor por conexão (Berners-Lee et al., 1996). A versão 1.1 do protocolo HTTP introduziu o conceito de conexões persistentes, que têm a capacidade de reutilizar a mesma conexão TCP para solicitar vários objetos do servidor de forma sequencial. O protocolo HTTP/1.1 também implementa a técnica de *pipeline*, que possibilita ao cliente realizar múltiplas requisições HTTP para a mesma transferência.

Neste cenário, novos objetos não são solicitados até que a transferência do objeto anterior tenha terminado (Fielding et al., 1999). Entretanto, caso haja perda de um segmento durante a transmissão de um objeto, esta perda introduz o problema de bloqueio de fila de encaminhamento (do inglês, *Head-of-Line Blocking* - bloqueio HoL) (Erman et al., 2015), no qual as solicitações subsequentes ficam suspensas, esperando a retransmissão do segmento perdido. O protocolo SPDY aplica a redução dos cabeçalhos HTTP, multiplexação das requisições e a possibilidade de priorizar a requisição de um objeto. Em cenários com menor complexidade, ele apresentava melhor desempenho comparado com o HTTP/1.1 (Padhye e Nielsen, 2012). Entretanto, com o uso de certificados SSL (do inglês, *Secure Sockets Layer*) para aumentar a segurança das páginas Web, o protocolo demonstrou baixo desempenho. Além disso, por utilizar apenas uma conexão TCP e com a multiplexação das requisições sobre esta conexão, o protocolo SPDY agrava o problema causado pelo bloqueio HoL, que estaciona todo o tráfego subsequente quando uma perda ocorre, esperando pelo reenvio do pacote faltante.

A empresa Google descontinuou o projeto do SPDY em 2015, mas todo o conhecimento adquirido com o seu desenvolvimento foi aplicado na versão 2 do protocolo HTTP (HTTP/2) (Belshe et al., 2015). Contudo, observando que muitos dos problemas enfrentados estavam relacionados ao protocolo de transporte (TCP) (Projects, 2018), em 2013, a empresa começou a desenvolver uma nova proposta. O protocolo QUIC (do inglês, *Quick UDP Internet Connection*) é um protocolo *middlelayer*, executado pela camada de aplicação fundado no protocolo UDP. Seu objetivo é fornecer um conjunto flexível de recursos, permitindo a ele ser um protocolo de transporte generalizado para múltiplas aplicações, bem como a redução do atraso ao longo das conexões entre cliente e servidor e, sobretudo, oferecer maior segurança para a troca de dados (J. Iyengar e M. Thomson, 2019).

Baseado em técnicas e experiências obtidas com outros protocolos, como o TCP, SPDY, TLS e HTTP/2, o protocolo QUIC é executado na aplicação cliente (e.g. Chrome) e utiliza o protocolo UDP em sua essência para transferir dados a um servidor Web com suporte ao protocolo QUIC. A grande vantagem dessa característica diz respeito a sua facilidade de implementação, sem necessidade de alterar sistemas operacionais dos hospedeiros ou *middleboxes*. Apesar dos protocolos da pilha TCP/IP seguirem as especificações de suas respectivas RFCs, propor e aplicar uma atualização em algum desses protocolos pode levar anos para serem consolidadas e implantados nos dispositivos. A título de exemplo, o protocolo IP em sua versão 6, levou cerca de 20 anos no processo de desenvolvimento até sua implementação (Dhamdhere et al., 2012).

Apesar de utilizar um protocolo não orientado à conexão, o protocolo QUIC contorna as adversidades conhecidas do protocolo UDP oferecendo vantagens em relação ao protocolo TCP. Dentre elas, podemos citar a redução de atraso no processo de estabelecimento da conexão, multiplexação com *streams* de dados independentes, resultando na redução do problema causado pelo bloqueio HoL, autenticação entre o cliente e o servidor antes de iniciar a transmissão, encriptação de cabeçalhos e dos dados transmitidos. O protocolo QUIC também oferece maiores informações sobre a transmissão para o controle de congestionamento, o que contribui para

uma recuperação mais eficiente diante de perdas de pacotes (de acordo com a documentação do protocolo QUIC, a unidade de referência aos dados do protocolo QUIC é tratado como pacotes e não segmentos, por este motivo, este trabalho descreverá suas unidades como pacote).

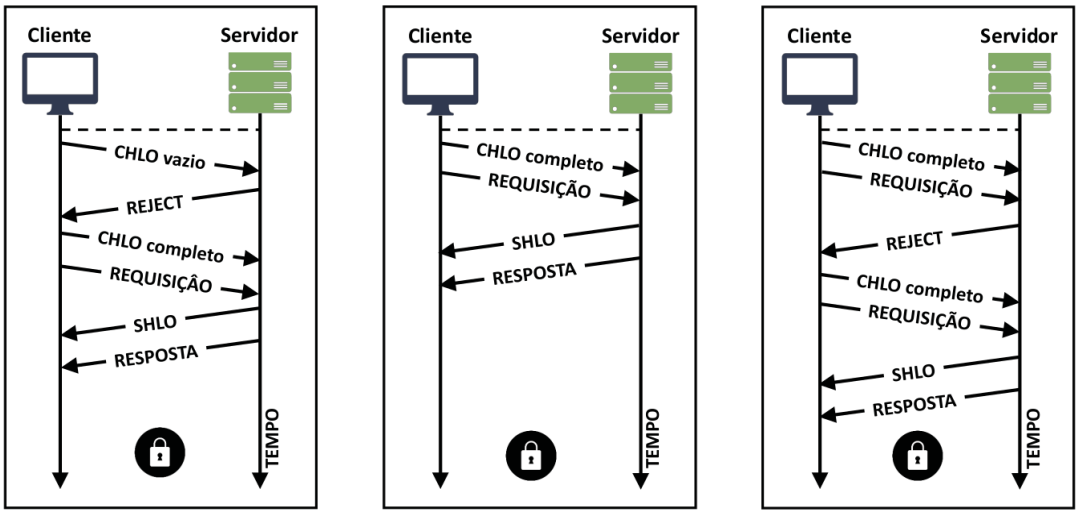
O protocolo QUIC é uma alternativa para o conjunto formado pelos protocolos *HTTP/2+TLS+TCP*. Em síntese, ele recebe as informações da camada de aplicação, passadas pelo protocolo HTTP/2, criptografa essas informações com uma variação do protocolo TLS adaptada para o protocolo QUIC, insere um cabeçalho também criptografado, encapsula este pacote em um datagrama UDP e o passa o protocolo IP, na camada de rede. O protocolo QUIC possui duas variações cabeçalho, podendo ser do tipo *longo* ou *curto*, de acordo com a negociação inicial entre os hospedeiros (*handshake*). Os cabeçalhos longos são utilizados na primeira conexão entre o cliente e o servidor e criam uma autenticação entre eles. Os cabeçalhos curtos são utilizados quando os hospedeiros já trocaram dados previamente. Assim, o processo de *handshake* é dispensado e o cliente pode iniciar a conexão enviando dados imediatamente.

2.1.4.1 Estabelecendo uma conexão QUIC

Em sua grande maioria, a expectativa é que as conexões QUIC utilizem o processo de 0-RTT *handshake*. Dessa forma, as requisições serão enviadas de imediato ao cliente, sem a necessidade de esperar por um acordo prévio. O protocolo QUIC provê um *stream* dedicado para a execução do *handshake*. Neste processo, ele utiliza o TLS em sua versão 1.3 (Rescorla, 2018), o qual apresenta melhor desempenho com relação à latência, quando comparado com as versões anteriores. O protocolo TLS fornece a dois hospedeiros uma maneira de estabelecer uma comunicação segura em um meio não confiável, como a Internet, garantindo que as mensagens trocadas não possam ser observadas, modificadas ou falsificadas. Os recursos do protocolo TLS podem ser separados em duas funções básicas: troca de chaves autenticada e registro de proteção, como descrito na Subseção 2.1.3. O protocolo QUIC usa o protocolo TLS durante o *handshake*, mas executa sua própria forma de registro de proteção para seus pacotes (Langley e Chang, 2016).

Quando a conexão cliente-servidor ocorre pela primeira vez, o protocolo QUIC usa o processo de 1-RTT para estabelecê-la. A Figura 2.6(a) demonstra o processo em que o cliente envia um pacote *CHLO (ClientHello)* sem carga útil para o servidor. Durante este processo, o protocolo QUIC executa as funções do protocolo TLS para a troca de chaves secretas. Além disso, o servidor gera um *token* para validação de endereço do cliente. Nos casos em que a conexão tenha ocorrido previamente, há duas situações possíveis: (*i*) se o *token* trocado na primeira conexão ainda for válido, o cliente envia um pacote *CHLO* e as requisições criptografadas. Em seguida, o servidor responde com um pacote *SHLO (ServerHello)* para finalizar o processo de 0-RTT *handshake* e as respostas das requisições também criptografadas, conforme ilustra a Figura 2.6(b); e (*ii*) se o *token* perdeu a sua validade, o cliente irá iniciar a conexão enviando dados criptografados para o servidor, conforme ilustra a Figura 2.6(c). Entretanto, o servidor irá rejeitar a solicitação do cliente enviando um *REJ Packet* com um novo *token* para o cliente que o utilizará para transmitir suas requisições (Thomson e S. Turner, 2019).

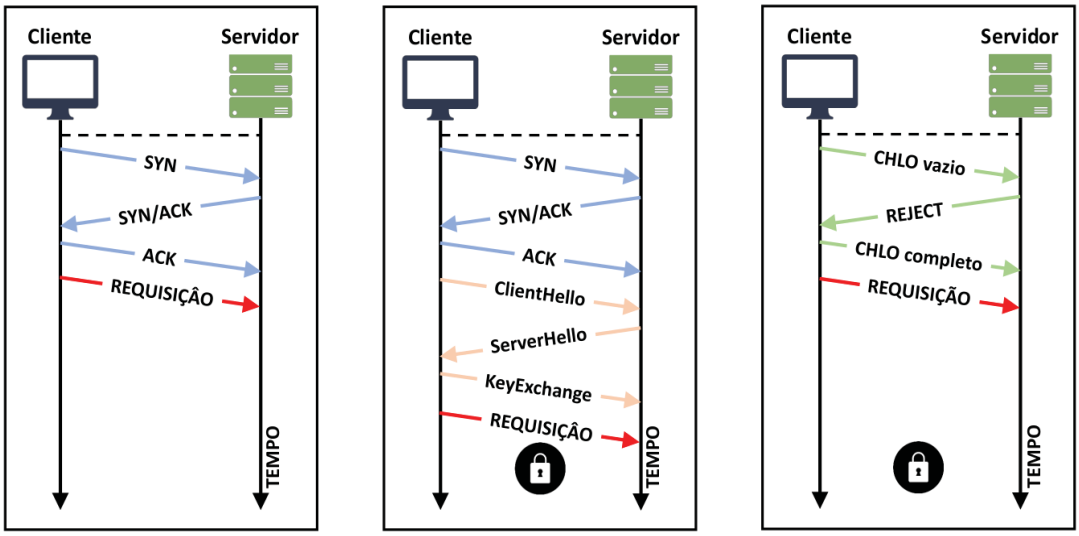
O *three-way-handshake* está presente em todas as conexões tradicionais do protocolo TCP e leva um tempo considerável para ser concluído, mesmo nos casos em que a conexão foi estabelecida previamente entre os hospedeiros, como demonstra a Figura 2.7(a). A título de exemplo comparativo entre os protocolos TCP, TCP+TLS e QUIC, consideremos um RTT de 100ms, o protocolo TCP levaria cerca de 100ms para fazer o *handshake* e enviar as requisições do cliente. Em casos em que a conexão utiliza o protocolo TLS para autenticar e criptografar os dados e se essa for a primeira conexão entre o cliente e o servidor, conforme a Figura 2.7(b), esse processo leva mais 2 RTTs para ser finalizado, devido à negociação, troca de chaves e certificados, totalizando 300ms. O protocolo QUIC reduz todo esse processo para no máximo 1-RTT nos



(a) 1-RTT (b) 0-RTT (c) 0-RTT REJECT

Figura 2.6: Tipos de handshake do protocolo QUIC

casos em que o cliente nunca tenha estabelecido conexão com o servidor e 0-RTT para os casos em que a conexão tenha sido efetuada anteriormente, como demonstra a Figura 2.7(c).



(a) TCP *handshake* (b) TCP+TLS *handshake* (c) QUIC *handshake*

Figura 2.7: Comparativo de handshake

O protocolo QUIC também aplica uma validação de endereço da origem antes de iniciar uma transmissão. Esta validação é uma contramedida para evitar que o protocolo QUIC seja utilizado em ataques de amplificação. Em tal ataque, um agente malicioso envia vários pacotes para um (ou mais) servidor com o endereço de origem falsificado, o servidor responde a estas solicitações, porém as respostas são enviadas para o endereço falsificado da vítima. O protocolo TLS fornece as ferramentas que suportam o recurso, mas a validação básica é executada pelo protocolo QUIC. Após completar o TLS *handshake*, o cliente terá aprendido e autenticado uma identidade para o servidor. Para o servidor, esse passo é opcional. A troca de chaves TLS é resistente a violações de invasores e proporciona confidencialidade aos hospedeiros (Langley e Chang, 2016; J. Iyengar e M. Thomson, 2019).

2.1.4.2 Pacotes QUIC

Todos os pacotes QUIC possuem um cabeçalho longo ou curto, conforme indicado pelo primeiro bit configurado em seu cabeçalho. Os cabeçalhos longos, identificados com o valor 1, são usados no início da conexão, antes da negociação da versão do protocolo e do estabelecimento de chaves secretas, utilizadas para criptografar as mensagens. Os cabeçalhos curtos são identificados com o primeiro bit do cabeçalho configurado com valor 0 e são específicos da versão em uso, que são utilizados após a negociação de versão e após as chaves secretas serem trocadas. Como o protocolo continua em desenvolvimento e caminha para sua padronização, alguns parâmetros desses pacotes podem ser atualizados conforme este processo progride. Entretanto, os parâmetros como a forma do cabeçalho, tipo de pacote, ID da conexão, número do pacote e versão tendem a permanecer imutáveis (Thomson, 2019). Além disso, os cabeçalhos dos pacotes QUIC possuem campos em texto puro, chamado de cabeçalho público, como também campos criptografados, chamados de cabeçalho privado.

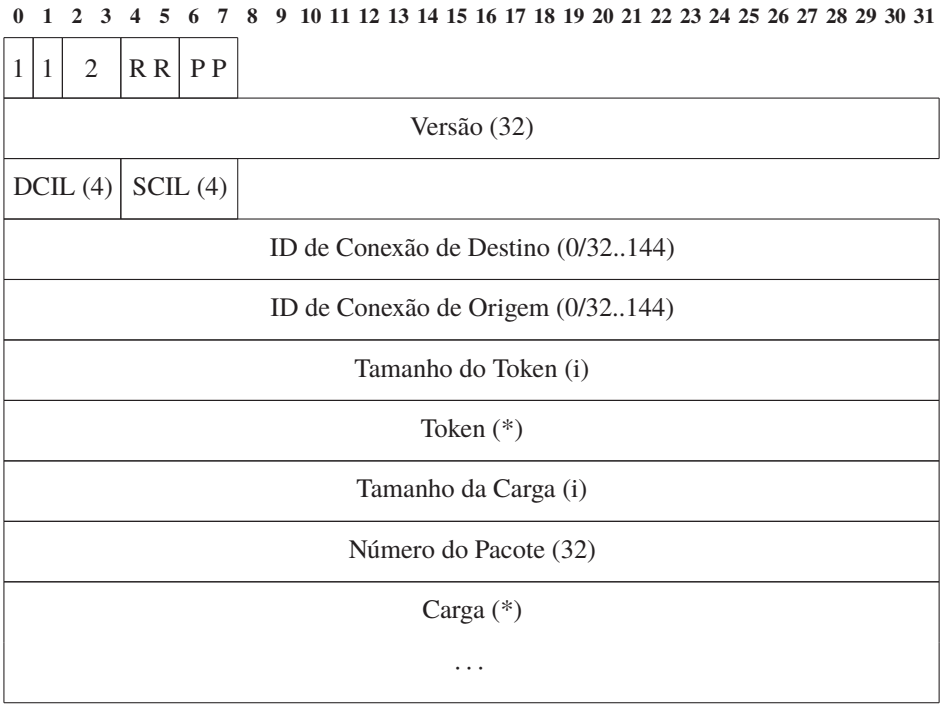


Figura 2.8: Cabeçalho longo de pacotes QUIC

A Figura 2.8 apresenta o cabeçalho longo usado em pacotes enviados antes da conclusão da negociação de versão e estabelecimento de chaves no processo de 1-RTT. Quando ambas as condições são atendidas, um remetente alterna para o envio de pacotes usando o cabeçalho curto. A forma longa permite que pacotes especiais, como o pacote de negociação de versões, sejam representados neste formato uniforme de pacote de tamanho fixo. De acordo com a versão mais recente do *Draft* do protocolo QUIC (J. Iyengar e M. Thomson, 2019), o cabeçalho longo é formado pelos seguintes campos:

- **Tipo de cabeçalho:** O bit mais significativo do primeiro octeto (octeto 0) define o tipo de cabeçalho utilizado, 1 para cabeçalhos longos;
- **Bit fixo:** este campo deve ser configurado com o valor 1. Pacotes configurados com o valor 0 representam versões antigas do protocolo e devem ser descartados;

- **Tipo de pacote (T):** contém dois bits para identificar o tipo de pacote. Os pacotes compreendem 4 tipos: (1) pacote inicial, utilizado ao enviar a primeira mensagem entre cliente e servidor para troca de chaves; (2) pacote 0-RTT, utilizado para enviar as requisições do cliente durante o estabelecimento de conexão; (3) pacote de *handshake*, ele é usado para transportar mensagens criptografados de handshake e mensagens de reconhecimentos do servidor e do cliente; (4) pacote de reestabelecimento, utilizado pelo servidor que deseja executar uma nova tentativa de conexão quando o *token* não possui validade;

- **Bit de tipo específico (X):** é utilizado para identificar tipos específicos de pacotes, geralmente utilizados no desenvolvimento do protocolo, como por exemplo, a identificação de um pacote modificado para fins de experimentação em meio ao tráfego normal;

- **Versão:** este campo indica qual versão do protocolo QUIC está em uso, podendo ser a versão em padronização ou a versão *gQUIC* utilizada pela Google. O campo é formado por 32 bits;

- **DCIL e SCIL:** O campo comprimento do ID de conexão de destino (do inglês, *Destination Connection ID Lenght - DCIL*) ocupa os primeiros 4 bits do terceiro octeto e o campo comprimento do ID de conexão de origem (do inglês, *Source Connection ID Lenght - SCIL*) ocupa os 4 bits restantes, ambos informam o tamanho do ID de origem e destino;

- **ID de conexão de destino:** o campo ID de conexão de destino é utilizado para identificar o destinatário das requisições;

- **ID de conexão de origem:** informa o ID de conexão usado na origem para que o destinatário possa encaminhar as repostas;

- **Tamanho do token:** informa o tamanho do *Token* em bytes;

- **Token:** Valor do *Token* utilizado para validar os endereços;

- **Tamanho da carga:** Informa o comprimento do campo carga em octetos, codificado como um inteiro de comprimento variável;

- **Número de pacote:** Informa o número de sequência do pacote. Os pacotes QUIC são acrescidos monotonicamente. O número do pacote é um inteiro no intervalo de 0 a 2^{62-1} e deve iniciar do número 0; e

- **Carga protegida:** Os pacotes com cabeçalho curto sempre incluem uma carga útil protegida através de criptografia derivada do protocolo TLS.

O cabeçalho curto é utilizado após o processo de 1-RTT *handshake*. Ele retira os campos que não são mais necessários para a comunicação. Isto traz ganhos em relação ao atraso de processamento, pois diminui a quantidade de informações que devem ser lidas e processadas. A Figura 2.9 demonstra o formato do cabeçalho curto e os campos diferentes do cabeçalho longo serão explanados logo a seguir.

- **Tipo de cabeçalho:** Assim como o cabeçalho longo, o bit mais significativo do primeiro octeto define o tipo de cabeçalho, sendo que o valor 0 indica cabeçalhos curtos;

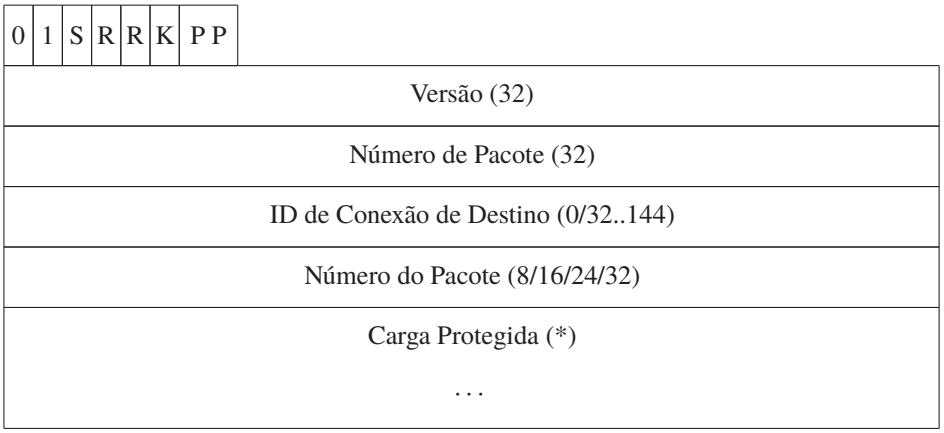


Figura 2.9: Cabeçalho curto de pacotes QUIC

- **Bit fixo:** este campo deve ser configurado com o valor 1. Pacotes configurados com o valor 0 representam versões antigas do protocolo e devem ser descartados;
- **Rotação de latência (S):** o bit de rotação de latência permite o monitoramento passivo da latência por agentes intermediários da conexão. A configuração deste campo é opcional;
- **Bit reservado (R):** Os próximos dois bits do cabeçalho curto são reservados;
- **Chave (K):** O campo número de pacotes é protegido usando uma chave derivada e separada da chave de criptografia dos pacotes. Após o *handshake*, é possível que o protocolo atualize as chaves secretas trocadas durante o processo inicial. O bit K indica a fase de chave (do inglês, *key phase*) no cabeçalho curto e é usado para indicar atualizações nessa chave secreta. Caso um hospedeiro identifique um bit K alterado, ele pode atualizar a chave e decriptografar o pacote que contém o bit alterado; e
- **Tamanho de número de pacote (P):** o número do pacote é um número inteiro, acrescido monotonicamente do valor 0 a 2^{62-1}. Sendo assim, os dois últimos bits do primeiro byte informam o tamanho do campo *número de pacote*.

2.1.4.3 Multiplexação por stream

O protocolo QUIC é capaz de multiplexar vários fluxos de dados em uma única conexão. Isto significa que sobre uma conexão QUIC, o protocolo é capaz de estabelecer vários subfluxos, chamados de *streams*. As requisições e as respostas são paralelas e assíncronas, isto é, a aplicação cliente é capaz de solicitar vários objetos ao mesmo tempo. Da mesma forma, a aplicação servidora pode enviar diversos objetos sobre a mesma conexão e de forma independente. Isso beneficia a conexão, pois caso uma perda de pacote ocorra em um *stream* específico, ela afeta apenas esse *stream* de dados e não interfere no fluxo dos outros *streams* que foram multiplexados. Dessa forma, o protocolo QUIC fornece uma transmissão multiplexada para o protocolo HTTP, sobre o protocolo UDP, sem o problema do bloqueio HoL (J. Iyengar e M. Thomson, 2019).

A multiplexação empregada pelo protocolo QUIC usa quadros de *stream* para transportar os dados da aplicação. O quadro *stream* contém um campo de ID do fluxo utilizado na identificação do fluxo ao qual aquele quadro pertence. Um único pacote QUIC pode conter vários quadros diferentes, incluindo quadros *stream* de diferentes fluxos. Para maximizar a eficiência, um

pacote QUIC deve ter o maior número de quadros possível. Nota-se que todos os pacotes transportados em um fluxo *stream* é bloqueado quando uma perda ocorre. Porém, o protocolo QUIC não lida com perdas ao nível de pacotes. Isto significa que os pacotes perdidos não são retransmitidos. Em vez disso, os quadros contidos dentro do pacote perdido são retransmitidos ao serem reempacotados em novos pacotes. Isto implica que os quadros são independentes dos pacotes que os transportam (J. Iyengar e M. Thomson, 2019).

2.1.4.4 Controle de fluxo e congestionamento

O protocolo QUIC implementa o controle de fluxo no nível de *stream* e de conexão. O receptor QUIC anuncia a quantidade máxima de dados que deseja receber em cada *stream*. Conforme os dados são enviados, entregues e reconhecidos em um *stream* específico, o receptor atualiza o tamanho da janela de recebimento, aumentando-a para esse fluxo e permitindo que o remetente envie mais dados nesse *stream*. Além do controle de fluxo no nível de *stream*, o protocolo QUIC implementa o controle de fluxo no nível de conexão. Dessa forma, ele limita o *buffer* agregado que um receptor QUIC está disposto a alocar para todos os fluxos em uma conexão. O controle de fluxo no nível de conexão funciona da mesma forma que o controle de fluxo no nível de *stream*, porém os bytes entregues e os limites são agregados de forma a obter um controle para o fluxo como um todo.

Assim como o protocolo TCP, o protocolo QUIC utiliza o algoritmo de controle de congestionamento *CUBIC* (Ha et al., 2008), o mesmo utilizado pelas distribuições Linux e Windows (Chavan, 2016). Contudo, os pacotes QUIC e seus reconhecimentos (ACK) contêm mais informações em relação aos segmentos ACKs do protocolo TCP. Isto beneficia o controle de congestionamento e a recuperação de perdas. Por exemplo, cada pacote QUIC transporta um novo número de pacote, incluindo aqueles que transportam dados retransmitidos. Isso elimina a necessidade de um mecanismo auxiliar para distinguir as confirmações de retransmissões das transmissões originais, evitando o problema de ambiguidade de retransmissão do TCP (Zhou et al., 2019). Os reconhecimentos QUIC também codificam explicitamente o atraso entre o recebimento de um pacote e seu reconhecimento sendo enviado e, juntamente com os números de pacote com aumento monotônico, isso permite o cálculo preciso do RTT.

2.1.4.5 Migração de conexão

De forma geral, os protocolos TCP e UDP são dependentes da tupla formada pelos endereços IPs e portas de comunicação. O protocolo QUIC abstrai esse conjunto de elementos e aplica seu próprio mecanismo para identificar uma conexão. Para as *middleboxes*, o fluxo é identificado de acordo com o padrão do protocolo UDP, contendo o IP de destino do servidor e a porta UDP, que por padrão é a 443, a mesma utilizada pelo HTTPS sobre o protocolo TCP. Entretanto, quando um pacote QUIC atinge o seu destino, é desencapsulado e entregue para a camada superior, o protocolo QUIC identificará a conexão que este pacote pertence de acordo com o campo ID de conexão no cabeçalho do pacote.

O protocolo QUIC utiliza pacotes ACKs (chamados de *ACK frames* em sua especificação), para confirmar que os dados foram recebidos e processados corretamente. Se o ponto final de recepção demorar muito para enviar a confirmação, um evento RTO ocorre. Um único pacote ACK pode reconhecer cumulativamente um ou uma série de pacotes QUIC. Cada pacote ACK contém um campo *largest acknowledgment*, que representa o maior número de pacotes recebidos até o momento, independentemente dos intervalos. Os pacotes QUIC ACK também possuem uma seção de blocos ACK, contendo até 256 blocos ACK, que representam os vários intervalos de pacotes que são reconhecidos (J. Iyengar e M. Thomson, 2019).

Dessa forma, o protocolo QUIC não retransmite pacotes perdidos. Em caso de perda, os quadros transportados pelo pacote perdido são empacotados novamente em outro pacote com um novo número de sequência. Assim, os números de sequência não se repetem em uma conexão. Isso significa que o número de sequência transportado por um pacote não é indicativo da ordem dos dados transportados. Para determinar a ordem dos dados em um fluxo, os pacotes em *stream* contêm os campos deslocamento e comprimento (J. Iyengar e M. Thomson, 2019).

2.1.4.6 Finalizando uma conexão QUIC

Uma conexão QUIC pode ser terminada de três maneiras diferentes:

1. **Tempo limite:** Durante o *handshake* QUIC, cliente e servidor negociam um valor de tempo limite ocioso. Se nenhum dado for transmitido ou recebido por mais tempo do que o valor estipulado para o contador de tempo limite ocioso, a conexão será encerrada;

2. **Encerramento imediato:** O encerramento imediato da conexão pode ser usado para a finalização da conexão. Este evento pode ser iniciado por qualquer uma das partes, enviando um pacote de encerramento. Depois de enviar um pacote de encerramento, o remetente entra no estado de encerramento. Durante esse estado, o remetente do pacote de encerramento responde a todos os pacotes de entrada com outro pacote de encerramento; e

3. **Reinicialização de estado:** A reinicialização de estado pode ser usada para finalizar uma conexão imediatamente. Para executar este evento, uma das partes transmite um pacote contendo um *token* de redefinição de estado, derivado do *token* utilizado no *handshake* do protocolo QUIC.

2.2 TRANSMISSÃO MULTICAMINHOS

Os computadores com múltiplas interfaces de rede não eram uma prioridade em seus projetos iniciais (Bonaventure e Seo, 2016). Somente comutadores foram inicialmente equipados com várias interfaces de rede desde os seus primórdios. No entanto, a Internet evoluiu significativamente desde então. Assim, a maioria dos servidores é equipada hoje com mais de uma interface. A abundância de recursos de rede do domínio do servidor estimulou a adoção da transmissão de múltiplos caminhos nas redes dos centros de dados (*data centers*). A proliferação de dispositivos móveis equipados com interfaces celulares (por exemplo, 3G e LTE) e Wi-Fi, traz consigo um número crescente de dispositivos *multihomed* para a Internet. Existe uma incompatibilidade entre o transporte de caminho único e a multiplicidade de caminhos de rede disponíveis. Os dispositivos multi-interfaces exigem capacidade de trabalhar com multicaminhos para melhorar o desempenho da comunicação de fim-a-fim e sua resiliência (Li et al., 2016).

As múltiplas interfaces de rede não são suficientes para utilizar os recursos oferecidos pelos multicaminhos (Yap et al., 2012). É necessário ter um protocolo de transporte projetado especificamente para esta função. Os protocolos TCP e UDP não oferecem suporte a este tipo de transmissão. Suas conexões são baseadas na estrutura composta pelos endereços IP e por portas de comunicação, criando uma dependência entre a camada de transporte e a camada de rede. Logo, os fluxos desses protocolos não podem ser associados a um novo endereço IP, sendo ou não de uma nova interface de rede, sem que haja a necessidade de se reestabelecer a conexão (Comer, 2008). Muitos protocolos foram propostos com objetivo de utilizar transmissão por multicaminhos (Li et al., 2016). A exemplo, o protocolo SCTP (do inglês, *Stream Control Transmission Protocol*) (Iyengar et al., 2006) e o protocolo MPTCP são protocolos padronizados pelo IETF.

O protocolo SCTP é um protocolo de transporte alternativo, capaz de suportar vários endereços IPs por conexão. As primeiras versões do protocolo SCTP usavam vários endereços em cenários de falha (do inglês, *failover*), mas extensões recentes permitiram que ele suportasse o uso simultâneo de vários caminhos. Infelizmente, exceto em aplicações específicas, como sinalização em redes de telefonia, o protocolo SCTP não é amplamente implementado. Os principais problemas enfrentados pelo protocolo são relacionados aos *firewalls* e aos servidores NAT, que não conseguem processar os segmentos SCTP e os descartam. Outra razão é que o protocolo SCTP expõe um conjunto de rotinas e padrões (do inglês, *Application Programming Interface - API*) diferente da API utilizada pelo protocolo TCP. Os fabricantes de dispositivos de rede não implementam o protocolo SCTP em seus *firewalls* porque nenhuma aplicação o utiliza, em contrapartida, desenvolvedores de aplicação não usam o protocolo SCTP porque os *firewalls* descartam os segmentos SCTP. Houve tentativas de quebrar esse círculo vicioso ao encapsular o protocolo SCTP sobre o protocolo UDP, mas o uso generalizado do protocolo SCTP ainda é indescritível (Paasch e Bonaventure, 2014).

2.2.1 Requisitos e desafios para a comunicação multicaminhos

Independente do protocolo ou da camada, as propostas que visam o uso de multicaminhos compartilham requisitos para manter o fluxo da rede e reduzir os desafios na transmissão (Li et al., 2016). Em relação aos requisitos da transmissão multicaminhos, ou os objetivos que se almeja alcançar com o seu uso, destacam-se o aumento na disponibilidade dos serviços de rede, a agregação de largura de banda, o princípio de justiça, o agrupamento de recursos (do inglês, *Resource Pooling - RP*) e a eficiência de Pareto ou ótimo de Pareto é um estado de máxima eficiência, onde qualquer realocação de um determinado recurso para melhorar o estado de um indivíduo irá necessariamente piorar as condições de outro indivíduo. O uso de multicaminhos também contribui com a proteção dos dados durante as transmissões. A Tabela 2.1 apresenta uma breve descrição de cada atributo supracitado.

Tabela 2.1: Principais requisitos para a transmissão multicaminhos

Requisitos	Descrição
Aumento da disponibilidade	O uso de caminhos adicionais contribui para manter a conexão ativa em caso de falhas ou em casos de menor desempenho de um caminho específico.
Aumento da taxa de transmissão	A transmissão de dados sobre multicaminhos utilizada de forma concorrente agrega a largura de banda dos canais disponíveis. Desta forma, espera-se multiplicar a taxa de transmissão pelo número de caminhos disponíveis.
Princípio de justiça	O princípio de justiça do protocolo TCP exige que um protocolo de transmissão de múltiplos caminhos não receba maior largura de banda em um gargalo do que um único fluxo TCP concorrente. Isto é importante porque o protocolo TCP é o protocolo de transporte dominante na Internet. Se novos protocolos adquirem uma capacidade injusta, eles tendem a causar problemas como o congestionamento de rede.
Agrupamento de recursos (RP)	Em vez de manipular os recursos de cada caminho de forma independente, o princípio de RP defende o uso aprimorado de vários recursos simultaneamente, agrupando-os e permitindo que caminhos distintos funcionem como se fossem um único recurso. O RP é um conceito que mudou as noções de justiça, tornando as comunicações multicaminhos concebível na prática.
Eficiência de Pareto	A eficiência de Pareto é um estado de alocação de recursos em que é impossível realocá-los, tal que a situação de qualquer participante seja melhorada sem piorar a situação individual de outro participante. No caso de transmissão de multicaminhos, isso significa que a sua utilização não pode reduzir a taxa de transferência de outros usuários de caminhos únicos.
Proteção de dados	Durante uma transmissão multicaminhos, em geral, os dados são distribuídos por caminhos independentes. Com a divisão das mensagens por caminhos distintos, a captura de todo o conteúdo por uma entidade maliciosa torna-se uma atividade complexa e custosa.

Tabela 2.2: Principais desafios para a transmissão multicaminhos

Desafios	Descrição
Reordenamento de Pacotes	Escalonar pacotes por caminhos heterogêneos sem causar o problema de reordenamento e penalidades no desempenho da transmissão é um grande desafio para as transmissões multicaminhos. Uma solução robusta de transmissão multicaminhos deve ser capaz de lidar com qualquer tipo de diversidade e heterogeneidade de caminhos.
Compatibilidade com o ambiente legado	Desenvolver uma solução multicaminhos genérica sem modificar protocolos padronizados ou alterar os equipamentos de rede é um grande desafio.
Princípio de justiça	Cumprir os requisitos do princípio de justiça tem sido um obstáculo para a transmissão multicaminhos. Tradicionalmente, aplica-se controle de fluxo por interface de rede, o que gera injustiça com outros fluxos diante de um gargalo compartilhado.
Heterogeneidade e diversidade	A diversidade dos caminhos está relacionada à capacidade de transmitir dados através de vários caminhos distintos para alcançar um destino. Com isto, espera-se obter um alto desempenho e maiores taxas de transferência. Mas se os caminhos trafegarem por um gargalo compartilhado, os múltiplos fluxos devem obter o mesmo comportamento de um único fluxo TCP devido ao princípio de justiça. Atualmente, a detecção confiável de gargalos é realmente difícil na prática. Nenhum esquema de seleção de caminho individual atende todos os ambientes de rede.
Pareto-Optimal	O protocolo MPTCP é a primeira proposta de transmissão multicaminhos que atende a eficiência de Pareto como um de seus requisitos. Contudo, há poucos trabalhos na literatura que identificam o quão bem ou o quanto ele atende esse requisito. Na realidade, há casos em que o protocolo MPTCP apresenta um desempenho pior do que o protocolo TCP devido à heterogeneidade dos caminhos.
Segurança	A transmissão multicaminhos anula mecanismos consolidados de segurança dos provedores. Por exemplo, embora a divisão de tráfego dificulte a interceptação dos dados, as *middleboxes* podem perder parte dos dados que trafegam por mais de um provedor e assim, dificultar a análise de fluxo e coleta de estatísticas. Isso resulta em soluções de segurança ineficientes, incluindo a detecção de invasão, prevenção de vazamento e injeção de dados.

A transmissão multicaminhos também possui desafios. Há também problemas de compatibilidade das novas abordagens com o ambiente legado. Os caminhos disjuntos contribuem para a comunicação por proporcionar a diversidade de caminhos, uma vez que caminhos cruzados durante uma transmissão podem compartilhar um gargalo na rede, diminuindo a taxa de transmissão e agravando o problema de congestionamento (Hu et al., 2016). Em contrapartida, a heterogeneidade das características dos caminhos, como o atraso observado (RTT), a largura de banda e a taxa de perda, contribuem para a chegada de pacotes fora de ordem no receptor. A Tabela 2.2 apresenta os principais desafios para as propostas de transmissão multicaminhos de acordo com a pesquisa de Li et al. (2016).

2.2.2 Escalonamento de pacotes

Com o uso de múltiplos caminhos, espera-se obter uma maior resiliência e um maior desempenho durante a transmissão de dados. Neste contexto, o escalonador de pacotes é um componente fundamental para a comunicação multicaminhos (Ramaboli et al., 2012), isso pois ele é o principal responsável na escolha dos caminhos e na alocação dos dados dos caminhos escolhidos. Uma decisão incorreta durante o processo de escalonamento impacta diretamente o desempenho do protocolo, podendo ocasionar uma diminuição considerável da capacidade em termos de vazão, dos vários caminhos (Kimura et al., 2017).

Os protocolos de transporte multicaminhos tentam prover uma maior sobrevivência (diversidade) e tolerância a falhas (redundância) para a conexão de diferentes formas. Por exemplo, na transmissão multicaminhos de forma concorrente (do inglês, *Concurrent Multipath Transfer - CMT*), os pacotes de um mesmo fluxo de dados são encaminhados através de múltiplos caminhos de forma paralela, como pode ser observado na Figura 2.10(a). Assim, a capacidade dos múltiplos canais é agregada ao transferir a sequência de pacotes $M1$, $M2$ e $M3$ por diferentes

caminhos, a fim de obter maior desempenho e vazão. Contudo, quando o objetivo é aumentar a resiliência e garantia de entrega, a transmissão multicaminhos resiliente (do inglês, *Resilient Multipath Transfer* - RMT) tenta evitar a interrupção ou a degradação da transmissão utilizando os caminhos de forma redundante. A Figura 2.10(b) ilustra uma conexão RMT. Observe que além da redundância física, através de múltiplas interfaces, e da redundância lógica com o uso de múltiplos caminhos virtuais, a redundância de informação com replicações de pacotes, como o pacote M que é replicado em M' e M'' nos caminhos disponíveis, tem sido empregada. Neste caso, um escalonador redundante simplesmente replica os dados sobre os caminhos disponíveis.

(a) Transmissão CMT (b) Transmissão RMT

Figura 2.10: Tipos de conexão multicaminhos

De forma geral, ao ser acionado pelo protocolo de transporte multicaminhos, o escalonador de pacotes precisa fazer três decisões. A primeira decisão diz respeito à coleta de informações dos caminhos disponíveis. Geralmente, o escalonador observa as características dos caminhos nesta fase, como o atraso e a largura de banda. A segunda decisão está atrelada a seleção do caminho. Dado um conjunto de caminhos, o escalonador deve decidir qual caminho será selecionado para transferir os dados da aplicação de acordo com as informações analisadas. E a terceira decisão corresponde a granularidade de dados (Yang et al., 2014). De acordo com Barré et al. (2011), a granularidade dos dados corresponde à quantidade de *bytes* que será enviado por um caminho antes de selecionar outro caminho.

2.2.3 TCP multicaminhos

O protocolo MPTCP foi projetado para lidar com as adversidades e experiências aprendidas com o protocolo SCTP (Paasch e Bonaventure, 2014). Com um de seus principais objetivos, o protocolo MPTCP almeja obter um desempenho melhor do que o oferecido por um único fluxo TCP, em termos de vazão e latência. Além de possibilitar um aumento para a resiliência da comunicação, agregação de largura de banda e prover redundância. Ele baseia-se no protocolo TCP e possibilita a transmissão de dados através de múltiplos caminhos e diferentes endereços para transmitir pacotes pertencentes a uma mesma conexão. Para a camada de rede, cada subfluxo é visto como um subfluxo TCP padrão, como pode ser observado na Figura 2.11, mas que transportam no cabeçalho do segmento TCP um tipo de campo *Opções* específico do protocolo.

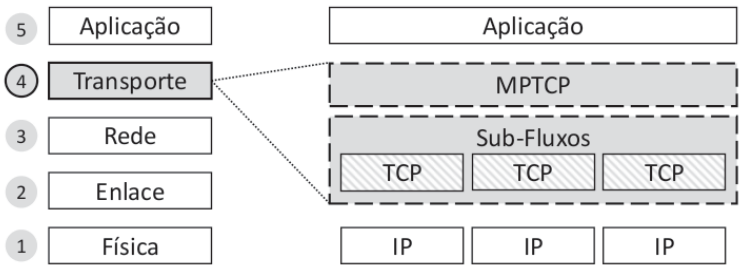


Figura 2.11: Protocolo MPTCP no modelo TCP/IP

2.2.3.1 *Arquitetura do MPTCP*

O uso de vários caminhos disjuntos visa aumentar a resiliência da conexão, pois transmitir dados por caminhos diferentes beneficia os hospedeiros que presenciam falhas em um de seus enlaces. Além disso, as transmissões multicaminhos tendem a aumentar a eficiência do uso de recursos e, consequentemente, aumentar a taxa de transmissão. O protocolo MPTCP possui um conjunto de atributos que o auxiliam durante a transmissão de dados. A arquitetura do protocolo MPTCP, descrita na RFC 6182 (Iyengar et al., 2011), inclui quatro componentes:

- **Gerenciador de caminhos:** Responsável pela descoberta dos caminhos entre os dois hospedeiros.
- **Escalonador de pacotes:** Após receber um fluxo de dados da aplicação, o escalonador realiza operações como segmentar os dados, adicionar um número de sequência no nível de conexão e distribuir os segmentos para os subfluxos conforme a política adotada.
- **Controle de congestionamento:** Realiza o controle do fluxo de dados e a prevenção de congestionamentos nos subfluxos.
- **Interface de subfluxos:** Ao receber um segmento, a interface de subfluxo adiciona o seu próprio número de sequência e encaminha-o através da rede.

2.2.3.2 *Estabelecendo uma conexão MPTCP*

Uma conexão MPTCP inicia de modo similar a uma conexão TCP, utilizando o *three-way handshake*. A Figura 2.12 ilustra o estabelecimento de uma conexão MPTCP entre cliente e servidor. Neste exemplo, o cliente possui duas interfaces de rede disponíveis para a comunicação, que correspondem às interfaces $C1$ e $C2$. O servidor possui igualmente duas interfaces, representadas por $S1$ e $S2$. A partir das interfaces $C1$ e $S1$, o cliente inicia o *three-way handshake* iniciando a conexão entre eles, formando o caminho $P1$. Após a configuração inicial, os hospedeiros trocam informações sobre os endereços adicionais que possuem e um novo *three-way handshake* é realizado para adicionar um subfluxo sobre as interfaces $C2$ e $S2$, formando o caminho $P2$. O protocolo MPTCP ainda permite que outros subfluxos sejam criados, por exemplo, combinando os endereços $C1 + S2$ e $C2 + S1$.

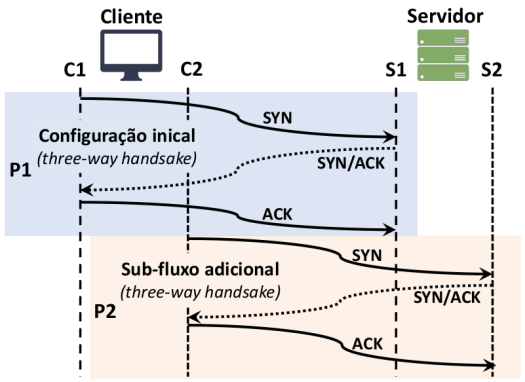


Figura 2.12: Exemplo de uma conexão MPTCP

2.2.3.3 *Transferência de dados MPTCP*

Para o protocolo MPTCP, todos os subfluxos compartilham o mesmo *buffer* de recebimento e anunciam a mesma janela de recebimento. Dessa forma, ele aplica dois níveis de reconhecimento: (*i*) por subfluxo (do inglês, *Subflow Sequence Number - SSN*) e (*ii*) por conexão (do inglês, *Data Sequence Number - DSN*) como pode ser observado na Figura 2.13. Os reconhecimentos SSN são utilizados para confirmar segmentos no nível dos subfluxos TCP de maneira independente. Os reconhecimentos DSN são utilizados para garantir a entrega ordenada no nível dos dados transmitidos pelos subfluxo. Esses reconhecimentos acompanham o avanço da transmissão e são responsáveis pelo deslocamento da janela de recebimento (Ford et al., 2013).

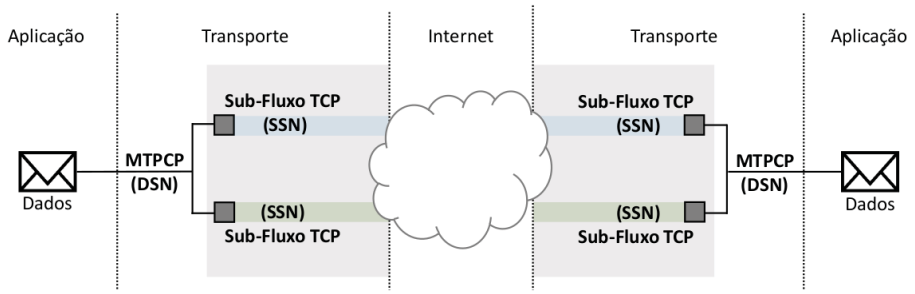


Figura 2.13: MPTCP

2.2.3.4 *Gerência de caminhos MPTCP*

Após o estabelecimento de uma conexão MPTCP, os hospedeiros podem iniciar novos subfluxos. A maneira mais simples de um hospedeiro comunicar que possui um endereço adicional é iniciar um subfluxo diretamente sobre esse caminho. Um hospedeiro também pode aprender sobre os endereços adicionais de seus pares (se houver algum) por meio da sinalização de endereço. Um hospedeiro pode informar endereços adicionais transmitindo segmentos com a opção `ADD ADDR`, no campo de opções do cabeçalho MPTCP, sobre um subfluxo já estabelecido. Os endereços anunciados anteriormente podem ser revogados usando a opção `REMOVE ADDR`.

Um subfluxo é aberto após a transmissão de um segmento SYN que inclui a opção `MP JOIN`. A opção `MP JOIN` contém um *token* proveniente do *handshake* sobre o subfluxo inicial. O *token* é utilizado para evitar ataques de repetição, explanado e categorizado por Syverson (1994). Uma função *hash* é utilizada sobre o *token*, que em seguida é transmitido ao longo de um *nonce*, um número arbitrário que é usado apenas uma vez.

O mecanismo de gerenciamento de caminhos do protocolo MPTCP ainda provê duas formas de associação de subfluxos. A primeira, associa cada subfluxo a uma interface/endereço IP. Na segunda, o gerenciador de caminhos associa múltiplos subfluxos com uma mesma interface/endereço IP, alterando apenas as portas de comunicação. Neste último caso, o protocolo MPTCP visa alcançar uma maior diversidade de caminhos, uma vez que cada subfluxo é tratado pela rede como um único fluxo TCP independente, e deste modo, podem ser encaminhados por caminhos distintos, aumentando a resiliência da transmissão.

2.2.3.5 *Controle de congestionamento*

Ao utilizar vários caminhos, deve-se ter cautela para que múltiplos fluxos obtenham equidade com os outros fluxos nos enlaces de rede e, ao mesmo tempo, alcançar o máximo de desempenho com os recursos disponíveis. Como vários caminhos podem compartilhar um

único enlace (gargalo de rede), o uso do controle de congestionamento padrão do protocolo TCP pode afetar o princípio de justiça (Aydin et al., 2012). Além do controle de fluxo, o controle de congestionamento tem como função balancear o tráfego entre os subfluxos, evitando aqueles mais congestionados e garantindo que o MPTCP seja amigável com os demais fluxos (TCP-*Friendly*) (Barré et al., 2011). Dessa forma, o protocolo MPTCP usa um controle de congestionamento acoplado para todos os seus subfluxos.

O algoritmo de controle de congestionamento executado pelo protocolo MPTCP tem como base os algoritmos do protocolo TCP, como a variante NewReno. Em geral, os algoritmos propostos na literatura modificam apenas a fase de prevenção de congestionamento, ficando as fases de inicialização lenta, retransmissão rápida e recuperação rápida inalteradas (Singh et al., 2013). O controle de congestionamento é executado no nível de subfluxo, utilizando uma janela de congestionamento individual e uma janela de recebimento compartilhada para facilitar a entrega ordenada dos dados. Seus requisitos principais incluem aumentar a vazão, não prejudicar fluxos concorrentes e balancear o congestionamento. A vazão dos subfluxos em uma transferência multicaminhos deve ser, pelo menos, melhor que um único fluxo através do melhor caminho. Os subfluxos precisam se comportar como um único fluxo TCP na existência de um gargalo comum. Por fim, o tráfego deve ser direcionado para os caminhos menos congestionados

O algoritmo LIA (*Linked Increase Algorithm*) é definido como algoritmo padrão segundo o IETF (Ford et al., 2013). Ele especifica apenas como deve ocorrer o incremento da janela de congestionamento ao receber um ACK, mantendo o decremento padrão como no protocolo TCP. O crescimento das janelas dos subfluxos são acoplados (Equação. 2.1). O parâmetro α, Equação. 2.2, controla a agressividade dos subfluxos de modo que o incremento da janela não seja superior ao de um fluxo de caminho único com o mesmo tamanho de janela. Nas Equações. 2.1 e 2.2, w_i e RTT_i referem-se ao tamanho da janela de congestionamento e o tempo de ida e volta (RTT) de um subfluxo i, respectivamente. O parâmetro W representa o tamanho total das janelas de congestionamento de todos os subfluxos.

$$w_i = \begin{cases} min(\alpha/W, 1/w_i) & incremento \\ w_i/2 & decremento \end{cases} \tag{2.1}$$

$$\alpha = W * \frac{max\left(w_i/RTT_i^2\right)}{\left(\sum_i w_i/RTT_i\right)^2} \tag{2.2}$$

2.2.3.6 Escalonamento de pacotes

A implementação do protocolo MPTCP no núcleo do Linux (Paasch e Barre, 2019) oferece uma infraestrutura modular para o escalonador de pacotes, oferecendo aos usuários a opção de escolher um entre três políticas de escalonamento disponíveis. A política de escalonamento Round-Robin (RR) é a política mais simples encontrado na implementação do protocolo MPTCP. Ao utilizar a política RR, os subfluxos irão alternar a transmissão de dados de maneira cíclica. O escalonador RR não leva em consideração as características do caminho, o que pode comprometer o desempenho, a menos que os caminhos sejam simétricos. Se o remetente for capaz de preencher as janelas de congestionamento de todos os subfluxos, o escalonador se tornar *ACK-clocked* (Paasch et al., 2014), isto é, novos pacotes serão escalonados em qualquer subfluxo que tenha espaço disponível à medida que sua janela de congestionamento se esvaziar.

A política de escalonamento LRF (do inglês, *Lowest-RTT-First*) é a política padrão do protocolo MPTCP. Inicialmente, o escalonador ordena os caminhos a partir do menor RTT e escalona os pacotes preenchendo a janela de congestionamento do caminho selecionado antes de

pular para o próximo. Assim como o RR, logo que a janela de congestionamento é preenchida, o escalonamento se torna *ACK-clocked* (Paasch et al., 2014).

Por fim, o escalonador redundante transmite um fluxo de dados nos caminhos disponíveis de modo redundante ou replicado provendo uma comunicação de alta confiabilidade. Esta política é útil quando o requisito da transmissão é alcançar uma menor latência ao custo de desperdiçar a largura de banda disponível. Embora isto aumente a resistência falhas de rede, ele multiplica o uso dos recursos da rede (Paasch et al., 2014).

2.2.4 QUIC multicaminhos

Com os protocolos TCP e UDP, todos os pacotes que pertencem a um determinado fluxo contêm a mesma tupla que atua como um identificador para esse fluxo. Isso impede um dispositivo de usar vários caminhos simultaneamente. O protocolo QUIC usa o protocolo IP como um fator de identificação para a conexão, mas não está atrelado a ele. Um fluxo QUIC é identificado pelo ID de conexão. Isso permite que os fluxos sobrevivam a eventos de mobilidade nos quais o endereço IP de um dos hospedeiros seja alterado. Esse recurso de conexão é fundamental para que o protocolo QUIC migre um fluxo de um caminho para outro. No entanto, esse recurso é implícito e o projeto atual do protocolo QUIC pressupõe apenas fluxos de caminho único. Para lidar com interferências contínuas do acesso ao meio, é importante usar dois (ou mais) caminhos simultaneamente durante a troca de dados (De Coninck e Bonaventure, 2019).

Em sua proposta inicial, os desenvolvedores do protocolo QUIC almejavam a implementação de multicaminhos e de um mecanismo de correção antecipada de erros (do inglês, *Forward Error Correction - FEC*). Entretanto, para o grupo de trabalho do protocolo QUIC, essas funções ficaram fora do escopo. Além disso, o grupo se concentra nos problemas de gestão de rede que o protocolo pode introduzir, com o objetivo de proporcionar uma maior aplicabilidade e capacidade de gerenciamento em paralelo ao trabalho real do protocolo (Cui et al., 2017).

Uma iniciativa externa do grupo de trabalho QUIC, apresentou uma proposta de multicaminhos (De Coninck e Bonaventure, 2019). Assim como o protocolo QUIC, o protocolo Multicaminhos QUIC (MPQUIC) caminha em seu desenvolvimento para se tornar um padrão. A proposta foi apresentada no final do primeiro trimestre do ano de 2018, e deve ser atualizada conforme a evolução do projeto. A Figura 2.14 apresenta uma visão geral do protocolo MPQUIC e sua disposição em relação à pilha de protocolos TCP/IP. Em testes preliminares (De Coninck, 2017), ambos os protocolos QUIC e MPQUIC, demonstram melhor desempenho quando comparado com os protocolos TCP e MPTCP, sucessivamente.

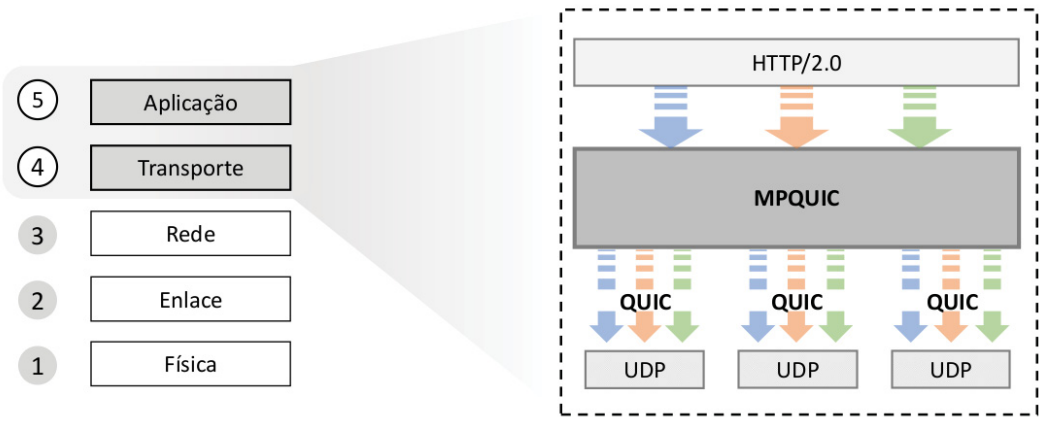


Figura 2.14: Visão geral do protocolo MPQUIC

2.2.4.1 *Arquitetura MPQUIC*

Além dos benefícios que o protocolo QUIC adiciona à comunicação cliente/servidor, o protocolo MPQUIC também proporciona o uso de múltiplas interfaces. Uma conexão MPQUIC inicia como uma conexão QUIC regular. Contudo, durante o *handshake*, ocorre uma verificação e negociação de multicaminhos entre os hospedeiros. Para estabelecer e gerenciar a comunicação mutlicaminhos, o protocolo MPQUIC conta com o auxílio de 5 componentes: (*i*) o gerenciador de caminhos, (*ii*) o controle de fluxo, (*iii*) o gerenciador de endereços, (*iv*) o controle de congestionamento e, (*v*) o escalonador de pacotes. A Figura 2.15 ilustra a arquitetura do protocolo e seus componentes, que serão explicados a seguir.

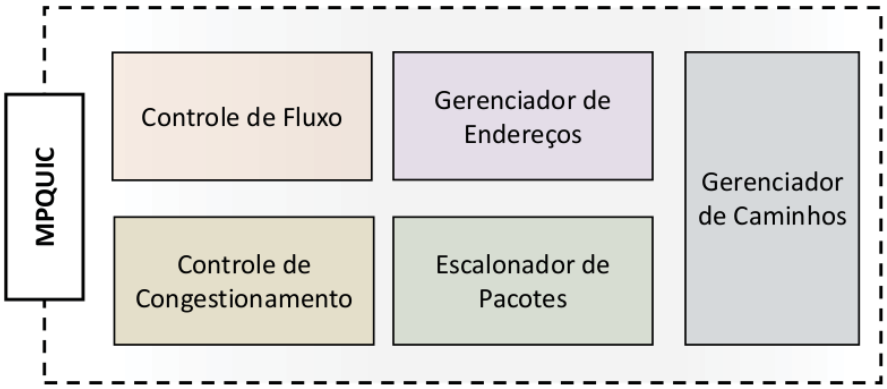


Figura 2.15: Arquitetura do protocolo MPQUIC

2.2.4.2 *Gerenciador de caminhos*

Uma vez estabelecida, uma conexão MPQUIC é composta por um ou mais caminhos. Cada caminho é associado a diferentes elementos e identificado pelo seu `PATH_ID` conforme demonstra a Figura 2.16. Para o MPQUIC, existem dois níveis de IDs de conexão. O primeiro é o ID de conexão principal `MCID` (`Master Connection ID`). Ele identifica exclusivamente a conexão. O segundo é o `PCID` (`Path Connection ID`) do caminho, que em conjunto com o número de pacote (`Packet Number - PN`), formam a tupla de identificação do tráfego da origem ao destino. Os pacotes pertencentes a um determinado caminho compartilham o mesmo **PCID** descrito no campo ID de Conexão (`Connection ID`) do cabeçalho público. O `PCID` atua como um identificador de caminho implícito para os pacotes. O `MCID` de uma conexão será o `PCID` do caminho inicial. Além do `PCID`, algumas informações adicionais são mantidas para cada caminho. Por exemplo, o `PATH_ID` identifica o caminho no nível de pacotes, e proporciona exclusividade para o *nonce* (número aleatório emitido pelo TLS para garantir que as comunicações antigas não possam ser reutilizadas em ataques de repetição). Além disso, cada caminho preserva sua janela de congestionamento e cada hospedeiro coleta informações e medições de cada caminho utilizado, como o RTT e informações de perdas de pacotes.

O gerenciador de caminhos controla a adição e remoção de novos caminhos durante uma conexão MPQUIC. O parâmetro de transporte `MAX_PATH_ID` trocado durante o *handshake* determina o limite superior do número de caminhos que podem ser criados na conexão, em adição ao caminho inicial. Em seguida, os hospedeiros devem concordar sobre quais caminhos poderão ser utilizados e quais IDs de conexão `Connection ID` serão empregados. Ao contrário do MPTCP (Ford et al., 2013), o servidor é quem controla dinamicamente quantos caminhos estão atualmente em uso. Isto porque geralmente, o servidor possui uma grande quantidade de

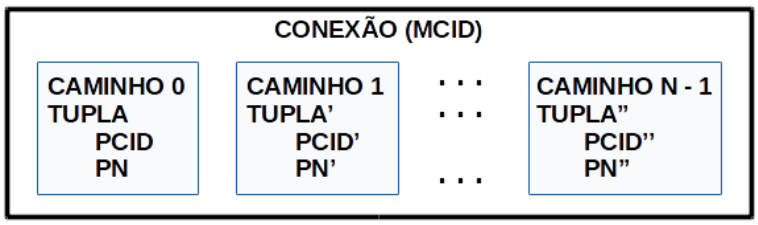


Figura 2.16: Tupla de identificação de uma conexão MPQUIC

requisições, ele deve limitar e otimizar seus recursos de forma a atender a todos os clientes de forma justa. Isso é realizado usando um novo pacote de controle chamado `MAX_PATHS`, que indica quantos caminhos adicionais podem ser usados simultaneamente.

O servidor propõe novos caminhos com uma versão estendida do pacote `NEW_CONNECTION_ID`. Esse pacote propõe um `PCID` para o caminho depois que o cliente recebe e reconhece o pacote, ele começa a usar o novo caminho inserindo o `PCID` no campo `Connection ID` do cabeçalho dos pacotes. O servidor começa a usar um caminho assim que o `NEW_CONNECTION_ID` correspondente for reconhecido. Ambos os hospedeiros armazenam as informações de `NEW_CONNECTION_ID` para lidar com um hospedeiro que tenta utilizar um novo caminho. Como os pacotes são criptografados, as informações continuam protegidas com o estabelecimento de novos caminhos.

Embora um caminho possa ser usado pela primeira vez por qualquer hospedeiro, dificilmente será iniciado por um servidor. Isto porque os clientes normalmente estão atrás de um servidor NAT, responsável por traduzir endereços locais para os endereços da Internet e vice-versa, a fim de reduzir o consumo de endereços IPs utilizáveis. O cliente nunca receberia este pacote, levando a problemas de conectividade nesse caminho. Para evitar tais problemas, o endereço remoto é validado durante o *handshake* antes de ser utilizado.

2.2.4.3 Controle de fluxo

O pacote QUIC atua como um contêiner para um ou mais quadros. O MPQUIC usa os mesmos pacotes *stream* que o QUIC para transportar dados. Um *byte offset* está associado à carga útil dos dados (*payload*). Uma das principais decisões de projeto do MPQUIC é que os quadros são independentes dos pacotes que os transportam. Isso implica que um quadro transmitido por um caminho poderia ser retransmitido posteriormente em outro caminho sem qualquer alteração. O caminho no qual os dados são enviados é independente a nível do próprio pacote. Isso significa que um quadro é enviado independentemente do caminho do pacote que o transporta. Além disso, como o deslocamento de dados é uma informação no nível de quadros, não há necessidade de definir números de sequência adicionais para lidar com a reordenação, ao contrário do protocolo MPTCP que usa um número de sequência de dados no nível do próprio protocolo. Outras considerações de controle de fluxo, como a janela de recebimento anunciada pelo quadro `MAX_STREAM_DATA`, permanecem inalteradas quando há vários caminhos.

No entanto, o MPQUIC está sujeito a reordenação de pacotes ao usar caminhos com diferentes latências. A presença do ID do caminho no cabeçalho público garante que os pacotes enviados em um determinado caminho contenham números de pacote com aumento monotônico. Para garantir maior flexibilidade e, potencialmente, reduzir o número de quadros ACK ao agregar largura de banda de caminhos que exibem características de rede diferentes, cada caminho mantém seu próprio espaço de número de pacote crescente e monotônico. Isso potencialmente permite enviar até $2^{32} \times 2^{62}$ pacotes em uma conexão QUIC, pois cada caminho contém seu próprio espaço para número de pacotes. O quadro ACK também é modificado para permitir confirmações de

pacotes por caminho. Isso permanece compatível com a independência entre pacotes e quadros, fornecendo mais flexibilidade aos hospedeiros para decidir em qual caminho eles desejam enviar suas confirmações. Tal decisão de escalonamento não é possível no MPTCP (Ford et al., 2013), que deve reconhecer dados no caminho em que foi recebido.

2.2.4.4 Gerência de endereços

Quando um dispositivo móvel *multihomed* se conecta a um servidor de pilha dupla em seu endereço IPv4, ele está ciente de seus endereços locais (por exemplo, o WiFi e os celulares) e o endereço remoto IPv4 usado para estabelecer a conexão QUIC. Se o cliente quiser criar novos caminhos sobre o IPv6, ele precisará aprender os outros endereços do hospedeiro remoto. Isso é possível graças ao gerenciador de endereços do protocolo MPQUIC. Ele utiliza os quadros `ADD_ADDRESS` para anunciar seus endereços atuais. Cada endereço anunciado é identificado por um ID de endereço. Os endereços anexados a um hospedeiro podem variar durante a vida útil de uma conexão MPQUIC. Um novo quadro `ADD_ADDRESS` é transmitido quando um hospedeiro tem um novo endereço. O quadro `ADD_ADDRESS` é protegido como outros quadros de controle QUIC, o que implica que ele não pode ser falsificado por invasores. O endereço comunicado é primeiramente validado pelo hospedeiro receptor antes de começar a usá-lo. Isso garante que o endereço realmente pertence ao hospedeiro e permite o envio e recebimento de pacotes.

Se o cliente estiver por trás de um serviço NAT, ele poderá anunciar um endereço particular em um quadro `ADD_ADDRESS`. Em tais situações, o servidor não poderá validar o endereço comunicado. O cliente ainda pode usar seu endereço NAT para iniciar um novo caminho. Para permitir que o servidor faça o link entre os endereços privado e público, o MPQUIC fornece o quadro `PATHS` que lista os IDs de caminho ativos atuais do hospedeiro transmissor. Um caminho é chamado de ativo quando foi criado sobre uma tupla validada e ainda está em uso. O quadro indica o ID do endereço local que o caminho usa. Com essa informação, o servidor valida o endereço público e associar o anunciado aos endereços percebidos.

Durante o tempo de vida de uma conexão MPQUIC, um hospedeiro pode perder alguns de seus endereços. Um simples exemplo é um *smartphone* saindo da acessibilidade de uma rede WiFi ou desligando uma de suas interfaces de rede. Tais remoções de endereço são anunciadas usando quadros `REMOVE_ADDRESS`. O quadro `REMOVE_ADDRESS` contém o ID do endereço perdido comunicado anteriormente por meio de `ADD_ADDRESS`.

2.2.4.5 Controle de Congestionamento

Apesar do algoritmo LIA ser o algoritmo de controle de congestionamento padrão do protocolo MPTCP, a implementação do protocolo no Linux traz duas variações de algoritmos Paasch e Barre (2019), como o algoritmo BALIA (*Balanced Linked Adaptation Congestion Control Algorithm)* (Ferlin et al., 2016) e o algoritmo OLIA (*Opportunistic Linked-Increases Algorithm*) (Khalili et al., 2013). O protocolo MPQUIC define este último (OLIA) como seu algoritmo padrão. O algoritmo OLIA modifica o comportamento da fase de prevenção de congestionamento, enquanto se comporta como o controle de congestionamento TCP padrão. Durante a fase de prevenção de congestionamento, o tamanho da janela de congestionamento do caminho r é aumentado, de acordo com a Equação 2.3, para cada ACK recebido pelo caminho p, onde w é o tamanho da janela, RTT é o tempo de ida e volta e $\mathcal{R}_u$ é o conjunto disponível caminhos (Khalili et al., 2013). α_r é calculado usando a Equação 2.4, onde $\mathcal{M}$ é o conjunto de caminhos com os maiores tamanhos de janela de congestionamento e $\mathcal{B}$ é o conjunto de melhores caminhos possíveis presumidos, em qualquer ponto no tempo. $\mathcal{B}\setminus\mathcal{M}$ representa caminhos que

ainda não preencheram por completo sua janela de congestionamento (Khalili et al., 2013). Através do termo α, o OLIA aumenta os tamanhos das janelas de bons caminhos que não são totalmente utilizados, redirecionando o tráfego de caminhos totalmente utilizados. Após de detectar uma perda, o tamanho da janela de congestionamento é reduzido pela metade.

$$\frac{w_r/RTT_r^2}{\left(\sum_{p\in\mathcal{R}_u} w_p/RTT_p\right)^2} + \alpha_r/w_r \tag{2.3}$$

$$\alpha_r = \begin{cases} \frac{1/|\mathcal{R}_u|}{|\mathcal{B}\setminus\mathcal{M}|}, & \text{se r} \in \mathcal{B}\setminus\mathcal{M} \neq \emptyset \\ -\frac{1/|\mathcal{R}_u|}{|\mathcal{M}|}, & \text{se r} \in \mathcal{M} \text{ e } \mathcal{B}\setminus\mathcal{M} \neq \emptyset \\ 0, & \text{caso contrário.} \end{cases} \tag{2.4}$$

2.2.4.6 Escalonador de pacotes

Na atual implementação do protocolo MPQUIC, o escalonador de pacotes padrão é o mesmo utilizado pelo MPTCP, o qual é baseado nas observações de RTT de cada caminho. Entretanto, a versão implementada para o protocolo MPQUIC possui algumas diferenças importantes em relação a versão do protocolo MPTCP. Conforme supracitado, o protocolo MPQUIC pode retransmitir seus pacotes por caminhos diferentes dos quais foram enviados. Em contrapartida, o protocolo MPTCP deve retransmitir seus segmentos no mesmo caminho que foram enviados inicialmente, a fim de evitar bloqueios de *middleboxes* que identificam números de sequência diferente do esperado. Como não há como saber o RTT de um caminho que ainda não transmitiu dados, o escalonador do protocolo MPTCP duplicará todo o tráfego de outro caminho até que o RTT do caminho se torne conhecido.

2.3 RESUMO

Neste capítulo foram revisados os conceitos da camada de transporte bem como seus principais protocolos e finalidade. Apresentou-se o protocolo QUIC, que faz parte desta camada e outros conceitos e métodos para a comunicação fim-a-fim entre processos, além de comparar sua arquitetura, quando possível, com o protocolo de transporte mais utilizado na Internet, o protocolo TCP. Este capítulo também descreveu o conceito dos protocolos multicaminhos com exemplos de suas aplicações e arquiteturas, como a versão multicaminhos para o TCP (MPTCP) e também a recente versão multicaminhos do protocolo QUIC, foco de estudo deste trabalho, o MPQUIC. Descreveu-se a estrutura do protocolo MPQUIC, suas características e definições conforme a sua atual proposta (*Draft*) apresentada ao IETF.

3 REVISÃO BIBLIOGRÁFICA

Este capítulo apresenta uma revisão da literatura acerca do problema de reordenamento de pacotes e das abordagens de políticas de escalonamento para a comunicação multicaminhos na camada de transporte. Dentre os principais desafios para a comunicação em multicaminhos, a entrega de pacotes fora de ordem causa o problema de reordenamento de pacotes e é um tópico tratado entre a maioria das propostas de transmissão multicaminhos concorrentes (do inglês, *Concurrent Multipath Transfer - CMT*). Na literatura, as abordagens que visam a entrega ordenada, geralmente direcionam suas propostas ao escalonador de pacotes, pois ele é responsável pela alocação dos dados nos caminhos disponíveis. Desta forma, a Seção 3.1 apresenta uma visão geral do problema de reordenamento de pacotes para as propostas CMT. A Seção 3.2 descreve o processo de escalonamento de pacotes de acordo com os critérios de seleção dos caminhos. A Seção 3.3 apresenta as atuais abordagens de escalonamento para protocolo MPQUIC. A Seção 3.4 apresenta uma discussão e as perspectivas acerca das propostas estudadas. Por fim, a Seção 3.5 resume e conclui o capítulo.

3.1 REORDENAMENTO DE PACOTES

De forma geral, o problema do reordenamento de pacotes pode ocorrer devido funcionamento inerente ao roteamento dos pacotes no núcleo da rede (He e Rexford, 2008), como o roteamento sobre vários caminhos, a oscilação de rotas, o paralelismo em roteadores de alto desempenho, as retransmissões da camada de enlace e o encaminhamento de roteadores (Leung et al., 2007). Além disso, o mecanismo de reordenamento de pacotes do protocolo TCP não interage bem com o mecanismo de retransmissão rápida do controle de congestionamento (Farrington, 2009). Entretanto, para as transmissões multicaminhos, o problema de reordenamento é muito mais expressivo, pois adiciona desafios ocasionados pela diversidade e heterogeneidade dos caminhos utilizados (Nam et al., 2016). Isso acontece porque em uma transmissão multicaminhos concorrente (do inglês, *Concurrent Multipath Transmission - CMT*), os pacotes de um mesmo fluxo de dados são encaminhados através de múltiplos caminhos que possuem diferentes características e por isso, chegam fora de ordem no receptor. O reordenamento de pacotes afeta principalmente as aplicações de tempo real, devido ao aumento do atraso em decorrência da espera pelos pacotes na ordem correta.

Ao executar o protocolo TCP sobre várias interfaces de rede, como faz o protocolo MPTCP, a entrega de pacotes fora de ordem se torna constante, causando a diminuição da janela de congestionamento e da taxa de vazão (Habak et al., 2015). A fim de lidar com os problemas que o reordenamento causa na comunicação sobre o protocolo TCP, Habak et al. (2014) apresentaram OSCAR, um sistema colaborativo para agregação de banda. Ao invés de entregar pacotes fora de ordem ao protocolo TCP, ele armazena os pacotes fora de ordem em *buffers* na camada de rede até que os pacotes faltantes cheguem. No trabalho de Lan e Li (2012), os autores apresentaram o protocolo ETOM (do inglês, *Enhancements for TCP On a Multi-homed mobile router*), que adota uma arquitetura cliente/servidor, servidor de *proxy* e roteadores móveis equipados com múltiplas interfaces. Similar ao OSCAR, a proposta ETOM lida com o reordenamento utilizando *buffers* na camada de rede. A Figura 3.1 demonstra o conceito utilizado por essas abordagens. Embora o reordenamento na camada de rede iniba a entrega de pacotes fora de ordem para a camada de transporte e, portanto, evita o encolhimento da janela de congestionamento, isso resulta na

detecção de perda de pacotes somente pelo esgotamento do tempo limite (*timeout*) no remetente, contribuindo para agravar o problema de congestionamento na rede.

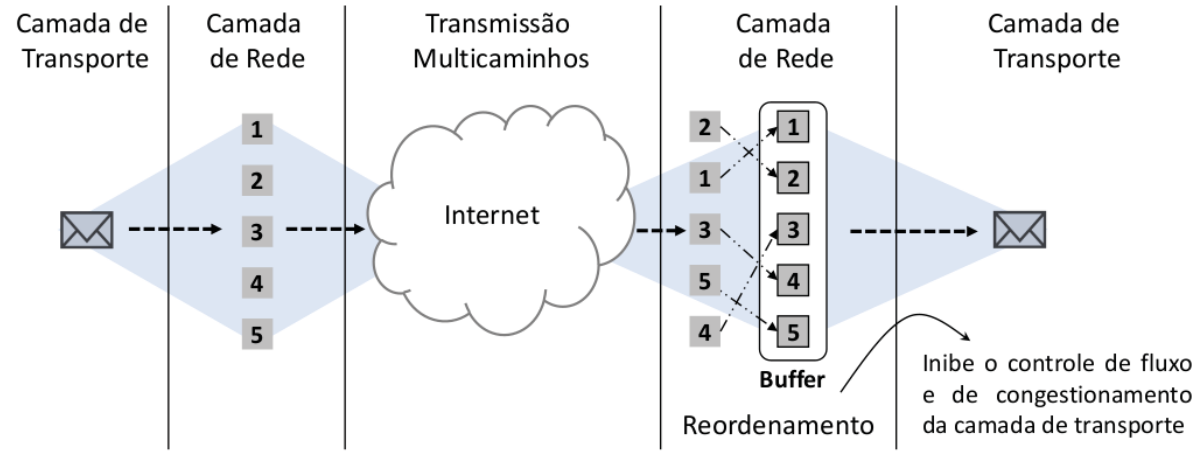


Figura 3.1: Reordenamento na camada de rede

Outras propostas lidaram com o problema de reordenamento ao implementar mecanismos de comunicação entre processos (*middleware*) transparentes sobre um protocolo de camada de transporte confiável, como o TCP. Contudo, como exposto anteriormente, trabalhar sobre várias interfaces simultaneamente provoca entregas fora de ordem. Assim, este tipo de solução CMT deve prover mecanismos extras para a entrega ordenada. O trabalho de Saeed et al. (2010) apresentou DNIS (do inglês, *Dynamic multiple Network Interfaces Scheduling*), um *middleware* de rede voltado para a agregação de largura de banda. O objetivo da proposta DNIS é escalonar os dados baseando-se nos serviços da camada de aplicação. Habak et al. (2012) desenvolveram o *framework* G-DBAS (do inglês, *Green and Deployable Bandwidth Aggregation System*), voltado para a agregação de banda e para a eficiência energética de dispositivos móveis. Em ambas as propostas (DNIS e G-DBAS), todos os pacotes pertencentes à mesma conexão TCP são transmitidos sobre a mesma interface. Essa abordagem permite que as soluções propostas convivam com o ambiente legado, não impondo alterações nos dispositivos.

Sakakibara et al. (2006), em contrapartida, propõem o mecanismo SBAM (do inglês, *a Socket-level Bandwidth Aggregation Mechanism*) para agregação de largura de banda de interfaces de rede sem fio no nível de *socket*. Os autores modificam o núcleo dos sistemas operacionais dos hospedeiros finais, alteram o cabeçalho dos segmentos TCP e dos datagramas UDP para controle de fluxo e tratamento do problema de reordenamento. No trabalho de Miyazaki e Oguchi (2011), os autores implementaram um *middlelayer* entre a camada de aplicação e a camada de transporte para a agregação de banda. O reordenamento de pacotes é realizado com o uso de *buffers* extras implementados na *middlelayer* apresentado. Apesar de apresentarem um bom desempenho nas avaliações dos autores, estas propostas demandam atualizações no núcleo dos hospedeiros, tornando-se uma barreira para a sua implantação em larga escala. Além disso, o uso de *buffers* extras ou o alargamento excessivo de *buffers* existentes acarreta problemas como o *bufferbloat* que ocorre devido à existência de grandes filas que absorvem enormes quantidades de tráfego em um canal congestionado, gerando alta latência com baixa perda de pacotes (Jiang et al., 2012).

3.2 ESCALONAMENTO DE PACOTES

O primeiro artigo sobre o protocolo TCP foi publicado em 1974 (Cerf e Kahn, 1974). No ano seguinte, o Dr. Maxemchuk apresentou em sua tese de doutorado uma proposta

para roteamento disperso, com o propósito de transmitir simultaneamente dados através de múltiplos caminhos (Maxemchuk, 1975). De acordo com Li et al. (2016), a primeira ideia de desenvolvimento do uso de multicaminhos para o protocolo TCP foi sugerida por Huitema (1995). Com isso, várias formas de transmissão multicaminhos foram propostas, pois seu uso visa principalmente obter um melhor desempenho comparado à comunicação baseada em caminhos únicos (Singh et al., 2015; Ramaboli et al., 2012). Contudo, ao utilizar múltiplos caminhos, um escalonador de pacotes é necessário, pois ele é responsável por decidir quais caminhos serão selecionados e utilizados na transferência de dados (Gurtov e Polishchuk, 2009). De acordo com Hu et al. (2016), o IETF não define como o escalonador deve ser implementado, tampouco caracteriza o processo de escalonamento (Singh et al., 2012). Porém, é possível classificar as políticas de escalonamento de acordo com o critério de seleção dos caminhos. A Figura 3.2 apresenta os principais critérios de seleção observados nos trabalhos estudados. Dentre eles, o atraso dos caminhos está presente em todas as propostas estudas, além da qualidade dos caminhos, que geralmente é associada à taxa de perda, à capacidade dos caminhos em termos de vazão, à correlação entre os caminhos, o tamanho de *buffer* de envio e à similaridade dos caminhos.

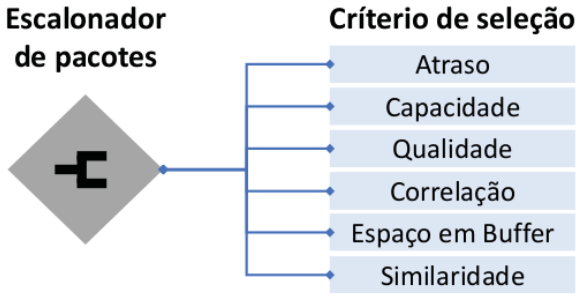


Figura 3.2: Critérios de seleção de caminhos utilizados pelo escalonador

3.2.0 Atraso

As abordagens que utilizam o atraso (latência) dos caminhos como principal critério de seleção de caminho representam a maioria de propostas encontradas na literatura. O RTT é comumente utilizado como único critério por estes trabalhos. O protocolo TCP usa o RTT para calcular o RTO ou para detectar um congestionamento na rede. Entretanto, devido à assimetria de características como a latência na Internet, há uma dificuldade em obter estatísticas de atraso unidirecional, ou seja, o tempo de ida separado do tempo de volta. Em geral, as estimativas de atraso consideram as medições do RTT assumindo que o tempo de envio e o tempo de confirmação são iguais. Porém, isso nem sempre é aplicável devido às flutuações do atraso no tempo de ida serem diferentes das flutuações do tempo de volta. Há também atualizações constantes das regras de roteamento, fazendo com que os pacotes enviados por um enlace sejam confirmados por outro (Li et al., 2016). Deste modo, muitos trabalhos propõem-se a calcular o atraso de uma via (do inglês, *One-Way Delay* - OWD). A estimativa OWD pode melhorar o desempenho e a vazão da transmissão, pois os pacotes são enviados pelos caminhos que os entregam mais rapidamente e, consequentemente, a confirmação desses pacotes (ACK) também poderá ser enviada pelo caminho com menor atraso.

A fim de aumentar o desempenho de transmissões multicaminhos e lidar com o desafio gerado pelos caminhos heterogêneos, Sarwar et al. (2013) apresentaram o escalonador DAPS (do inglês, *Delay-Aware Packet Scheduling*) como uma alternativa ao escalonador *Round-Robin*. O escalonador DAPS tenta obter uma entrega ordenada considerando o atraso de cada caminho

utilizado. No trabalho de Kuhn et al. (2014), os autores desenvolveram melhorias no escalonador DAPS. Para isso, o escalonador decide por qual caminho deve-se enviar cada pacote com base no atraso apenas de envio e na janela de congestionamento individualmente. Desta forma, espera-se prever quando um fluxo enviado por um caminho chegará no destino e, assim, enviar outro fluxo esperando que ambos cheguem na sequência correta no receptor.

A política de escalonamento padrão LRF (do inglês, *Lowest-RTT-First*) nos protocolos MPTCP e MPQUIC é baseada em observações do RTT para classificar os caminhos, selecioná-los e, em seguida, alocar os pacotes por esses caminhos disponíveis (Ford et al., 2013; J. Iyengar e M. Thomson, 2019). Como resultado do uso de várias interfaces para agregação de banda, cada pacote passa por caminhos que variam de comportamento em relação ao RTT. Segundo os trabalhos de (Paasch et al., 2014) e (Albaladejo et al., 2016), as estimativas do valor do RTT para cada caminho geralmente são incertas e mudam ao longo do tempo devido às filas, retransmissões, perdas de pacotes e políticas do algoritmo de congestionamento, entre outras. Estas flutuações no tempo de entrega dos pacotes são um dos principais fatores que causam o problema de reordenamento dos pacotes (Garcia-Saavedra et al., 2017). Além disso, a reordenação afeta o próprio cálculo de estimativa do RTT.

O trabalho de Le e Bui (2018), os autores lidaram com o problema de reordenamento de pacotes do protocolo MPTCP usando um algoritmo de escalonamento baseado no atraso de encaminhamento (do inglês, *Forward-Delay-Based Packet Scheduling* - FDPS). A proposta do trabalho usa a distribuição de pacotes através de múltiplos caminhos de acordo com o atraso estimado e a diferença de vazão de cada caminho individualmente. Além disso, os autores utilizaram um método de sincronização de relógios, a fim de obter o atraso apenas de ida dos pacotes. Baseado no *timestamp* de pacotes enviados por dois caminhos, os autores calculam as diferenças entre as estatísticas estimadas para obter o atraso apenas de envio por cada caminho. Porém, garantir a sincronização de relógios ainda é um desafio para essas abordagens.

3.2.0 Capacidade

A capacidade de um caminho está diretamente ligada à largura de banda dos enlaces de rede. De acordo com Prasad et al. (2003), existem três métricas relacionadas à capacidade de largura de banda de um caminho: (*i*) a largura de banda máxima possível de um enlace, (*ii*) a largura de banda disponível (do inglês, *Available Bandwidth* - ABW) que representa a largura de banda não utilizada e, (*iii*) a vazão atingida durante a transferência de um volume de dados (do inglês, *Bulk Transfer Capacity* - BTC). Em geral, os trabalhos que consideram a capacidade dos caminhos usam a métrica ABW (Li et al., 2016), pois a sua estimativa previne ultrapassar a capacidade dos caminhos e gerar congestionamentos na rede (Yang et al., 2013).

Em Tsai et al. (2010), os autores defendem que os serviços que requerem maior qualidade da rede, como vídeos e *streaming*, disponíveis na Internet atualmente, não possuem suporte e qualidade de serviço (QoS) que os atendam. Esses serviços dependem de uma alta largura de banda e baixos valores para o atraso da conexão. Entretanto, as condições das variáveis de cada caminho como a largura de banda disponível, o compartilhamento de gargalos na rede, o atraso e o congestionamento devem ser detectados e levados em consideração. Apenas a variação do RTT não é suficiente para detectar e medir a capacidade de um caminho. Neste cenário, Tsai et al. (2010) propuseram um esquema de controle de transmissão multicaminhos (do inglês, *Multipath Transmission Control Scheme* - MTCS) que estima a capacidade dos caminhos de acordo com uma modificação do algoritmo de controle de taxa amigável do TCP (do inglês,*TCP-Friendly Rate Control - TFRC*) para considerar o tempo de retransmissão do pacote no cálculo do RTT. Além disso, o esquema MTCS proposto leva em consideração a taxa de perda, a fim de atingir uma taxa de transmissão ideal de acordo com as características analisadas.

Park et al. (2016) também modificaram o algoritmo TFRC para calcular a ABW em cada caminho. Sua proposta é voltada para a transmissão de pacotes de vídeo por diferentes redes sem fio. Além de considerar o tempo de retransmissão no cálculo da capacidade dos caminhos, os autores propuseram um algoritmo para prever a sequência de chegada dos pacotes. O algoritmo ajusta o valor de incremento do número de sequência dos pacotes, de acordo com a ABW do caminho em que ele será enviado a fim de diminuir o número de pacotes que chegam fora de ordem no receptor.

3.2.0 Qualidade

Um dos primeiros trabalhos a definir uma métrica para a qualidade do caminho foi Mathis et al. (1997). Seus autores avaliaram o desempenho do algoritmo de controle de congestionamento TCP Reno, focando especificamente no mecanismo de prevenção de congestionamento. Com isso, eles desenvolveram um modelo simples de derivação aproximada, a fim de melhorar o desempenho da transmissão. Ao estimar a taxa de transferência do canal (B), que é obtida através da razão da quantidade dados transmitidos em um dado período de tempo (Equação 3.1), os autores consideraram o tamanho da janela de congestionamento W, o tamanho máximo de segmento (do inglês, *Maximum Segment Size* - MSS), o RTT. Além disso, eles determinam a perda de pacotes aleatórios com uma probabilidade constante p, assumindo que o enlace transmite aproximadamente $1/p$ pacotes consecutivos, conforme a Equação 3.2, para também estimar a qualidade do caminho.

$$B = \frac{Dados}{Tempo} \tag{3.1}$$

$$BW < \left(\frac{MSS}{RTT}\right) * \left(\frac{1}{\sqrt{p}}\right) \tag{3.2}$$

No trabalho de Liu et al. (2017), os autores apresentaram um esquema de transmissão multicaminhos simultânea baseada em retransmissão seletiva (do inglês, *CMT-Selective Rretransmission* - CMT-SR). A proposta do trabalho é direcionada para as transmissões simultâneas de vídeo de alta qualidade sobre múltiplos caminhos. Quando uma perda ocorre, o pacote perdido é retransmitido pelo caminho com menor atraso observado, a fim de diminuir o tempo de espera no receptor. Além disso, para obter uma estimativa mais precisa das informações dos caminhos utilizados, os autores propõem que cada caminho tenha uma fila de recebimento individual. Com isto, a ABW de cada caminho pode ser estimada a partir da diferença entre a quantidade de *bytes* que chega ao receptor e a quantidade de dados enviada sobre um caminho específico em uma dada janela de tempo.

Em Baidya e Prakash (2014), os autores propuseram uma técnica para identificar a qualidade dos caminhos definida pela métrica $\ell_P(t)/RTT(t)^2$, onde $\ell_P(t) = max\{\ell_{1P}(t), \ell_{2P}(t)\}$. Sendo $\ell_{1P}(t)$ o número de *bits* transmitidos com sucesso sobre o caminho P entre duas perdas, e ℓ_{2P} o número de *bits* transmitidos com sucesso pelo caminho P após a última perda. Se a razão entre a qualidade do caminho P_i em relação ao caminho mais rápido, dado pela métrica R (Equação 3.3), for maior que um limiar pré-definido, o caminho então é definido como um caminho de baixa qualidade.

$$R = \frac{\ell_{P_{max}}/RTT^2_{min}}{\ell_{P_i}/RTT^2} \tag{3.3}$$

3.2.0 Correlação

As abordagens que procuram estimar a correlação entre os caminhos têm por objetivo calcular a probabilidade de interação de dois ou mais caminhos durante a transmissão, evitando o compartilhamento de gargalos e aumentando a diversidade dos caminhos. Para esse fim, utiliza-se a correlação e, geralmente, empregam-se os coeficientes de Pearson e de Spearman, a fim de determinar o grau de correlação entre os caminhos. No trabalho de Felix et al. (2018), os autores apresentaram o escalonador redundante RED, que utiliza o coeficiente de correlação de Spearman para identificar o grau de correlação entre fatores como a taxa de perda, largura de banda e atraso. Contudo, eles utilizaram um algoritmo de controle de congestionamento desacoplado (CUBIC) em sua avaliação, o que fere o princípio de justiça com os fluxos concorrentes. No trabalho de Hu et al. (2016), os autores propuseram a política de escalonamento MLCS (do inglês, *Multi-level Correlation Scheduling*), que apresenta um modelo para calcular a taxa de transferência e a correlação dos caminhos. O MLCS classifica os caminhos de acordo com o RTT, a taxa de perda e grau de correlação entre os caminhos. Além disso, ele usa uma penalização para os caminhos quando este é considerado com um caminho de baixa qualidade, podendo remover um sub-fluxo quando o seu desempenho estiver abaixo dos critérios pré-estipulados.

3.2.0 Espaço em Buffer

O trabalho de Lim et al. (2017) apresentou o escalonador ECF (do inglês, *Earliest Completion First*), o qual usa o comprimento do *buffer* de envio para estimar o tempo total de fluxo (do inglês, *Flow Complete Time* - FCT) para cada caminho. Se o uso do caminho mais lento aumentar o FCT, ele aguardará pelo caminho mais rápido. O escalonador ECF toma decisões de escalonamento baseado no tamanho da janela de congestionamento e no RTT para evitar períodos de transmissão ociosos e, assim, alcançar uma maior taxa de transferência agregada. Da mesma forma que o escalonador LRF, o ECF prioriza o caminho mais rápido em termos de RTT. Além disso, uma vez que o caminho mais rápido é bloqueado após seu preenchimento pelo controle de congestionamento, o ECF avalia se é realmente benéfico enviar pacotes pelo caminho mais lento. Em seguida, ele pode decidir não enviar por esse caminho, a fim de aguardar a recuperação do caminho mais rápido. O ECF baseia essa decisão na quantidade de dados que aguardam o envio, no RTT do caminho e nas estimativas de capacidade. Contudo, ele não leva em conta o atraso de uma via. Portanto, não é capaz de obter uma chegada precisa e ordenada.

3.2.0 Similaridade

Em Kou et al. (2013), os autores propuseram uma estratégia de seleção de caminhos baseada na análise relacional *Grey* (GRA). A teoria cinzenta foi apresentada pela primeira vez por Deng et al. (1982). Na terminologia dessa teoria, uma situação sem informação é definida como preta e uma situação com informação completa como branca. Como as duas opções são idealizadas, os problemas do mundo real estão em algum lugar no meio, em uma situação definida como cinza. Assim, uma situação cinza pode ser modelada usando um modelo cinza (do inglês, *Grey Model*) (dos Prazeres et al., 2010; Husák et al., 2018). Dessa forma, a proposta de Kou et al. (2013) seleciona os caminhos de acordo com a similaridade das características calculadas através da técnica GRA. Apesar de utilizar essa técnica GRA para medir a similaridade entre dois caminhos, os autores não consideraram o impacto que ela causa à transmissão devido à necessidade de efetuar cálculos complexos, o que demanda de maior tempo de processamento e maiores recursos de *hardware*.

Özcan et al. (2017) apresentaram um esquema baseado na similaridade do RTT dos caminhos utilizáveis. Eles também implementaram uma técnica para retransmissão de segmento através de uma interface diferente da utilizada para transmitir o segmento original. A solução proposta estabelece várias conexões TCP sobre cada interface, mas usa apenas uma por vez. O esquema detecta quando um segmento de uma interface específica precisa ser retransmitido, finaliza a conexão onde houve a perda do pacote original e começa a usar uma das conexões de *backup*. Contudo, os autores não consideraram que as *middleboxes* bloqueiam fluxos desconhecidos e descartam seus pacotes.

3.3 ESCALONAMENTO NO PROTOCOLO MPQUIC

O protocolo MPQUIC está sendo padronizado pelo IETF e foi apresentado por De Coninck (2017). Em paralelo ao trabalho de De Conick e Bonaventure, Viernickel et al. (2018) também desenvolveram uma versão multicaminhos para o protocolo QUIC. A versão desenvolvida por De Conick e Bonaventure foi baseada e fundamentada nas características do protocolo MPTCP, já consolidado como um protocolo multicaminhos. Com isso, dois importantes componentes para a comunicação multicaminhos, como o controle de congestionamento e o escalonador de pacotes, foram adaptados para o protocolo MPQUIC, conforme explanado na Seção 2.2.4. Em contrapartida, Viernickel et al. apresentou sua versão do protocolo MPQUIC com um escalonador que permite a priorização de *streams* (do inglês, *Stream-Aware Scheduler* - SA-Scheduler). Isso possibilita um escalonamento de baixa granularidade. Além disso, a avaliação de Viernickel et al. (2018) considera vários cenários em conjunto com uma aplicação da proposta em ambiente real.

Rabitsch et al. (2018) desenvolveram um escalonador de pacotes que prioriza os *streams* do protocolo MPQUIC, similar ao trabalho apresentado por Viernickel et al. (2018). O escalonador *Stream-Aware ECF* (*SA-ECF*) leva em conta o RTT dos caminhos e o espaço do *buffer* de envio, assim como faz o escalonador ECF, além de contabilizar informações de prioridade de *streams* para a proposta. Contudo, os resultados obtidos não mostraram uma diferença significativa entre o desempenho dos escalonadores em relação ao tempo total de carregamento das páginas Web utilizadas como cargas de trabalho nas simulações.

Outro trabalho que trata do escalonamento para o protocolo MPQUIC foi recentemente apresentado por Mogensen et al. (2019). Os autores apresentaram a primeira proposta de escalonador redundante para o protocolo MPQUIC. O escalonador redundante seletivo (do inglês, *Selective Redundant Scheduler* - SR-Scheduler) oferece maior confiabilidade e disponibilidade para os *streams* prioritários. Além disso, os autores categorizam o tráfego em *background* e dessa forma, a proposta consegue replicar os *streams* prioritários e oferecer agregação de banda para o tráfego secundário. Os caminhos são classificados e selecionados de acordo com o OWD, obtido através do *timestamp* dos pacotes.

3.4 DISCUSSÃO E PERSPECTIVAS

O problema de reordenamento de pacotes é um problema clássico na literatura que afeta tanto os protocolos de caminhos únicos quanto os protocolos multicaminhos. Contudo, os protocolos multicaminhos são mais suscetíveis ao problema, pois o uso de múltiplas interfaces geralmente resulta em caminhos com características discrepantes. Para os protocolos multicaminhos, o escalonador de pacotes é um elemento chave em relação ao seu desempenho. Além disso, o escalonador contribui na redução dos impactos causados pelo problema de reordenamento, visto que ele é o responsável pela distribuição dos dados sobre os múltiplos caminhos disponíveis. Uma decisão errada do escalonador degrada toda a comunicação entre os hospedeiros.

As abordagens estudas utilizam as características como o atraso (latência), a capacidade do canal (ABW), a qualidade do caminho, o grau de correlação, o espaço em *buffer* de envio e a similaridade dos caminhos, como critério de seleção dos caminhos para o escalonamento de dados. A Tabela 3.1 apresenta os trabalhos de acordo com o protocolo utilizado e indica os critérios utilizados pelo escalonador de pacotes. Durante o desenvolvimento desta pesquisa, existiam poucos trabalhos sobre o protocolo MPQUIC na literatura. Esse número é ainda menor se considerado apenas os trabalhos que focam no desenvolvimento de soluções para o problema de reordenamento de pacotes. Entretanto, ressalta-se que o desenvolvimento acerca do protocolo cresceu no último ano, indicando um nicho de pesquisa relevante.

Tabela 3.1: Trabalhos estudados

Protocolo	**Trabalhos**	**Proposta**	**Critério de seleção de caminho**					
			Atraso	**Capacidade**	**Qualidade**	**Correlação**	**Buffer**	**Similaridade**
TCP	Habak et al. (2014)	OSCAR	✓	–	–	–	✓	–
	Habak et al. (2012)	G-DBAS	✓	–	–	–	–	–
	Lan e Li (2012)	ETOM	✓	–	–	–	✓	–
	Miyazaki e Oguchi (2011)	N/A	✓	–	–	–	–	–
	Saeed et al. (2010)	DNIS	✓	–	–	–	–	–
	Sakakibara et al. (2006)	SBAM	✓	–	–	–	–	–
CMT-SCTP	Kuhn et al. (2014)	DAPS	✓	–	–	–	–	–
	Sarwar et al. (2013)	DAPS	✓	–	–	–	–	–
MPTCP	Le e Bui (2018)	FDPS	✓	✓	–	–	–	–
	Felix et al. (2018)	RED	✓	✓	✓	✓	–	–
	Liu et al. (2017)	CMT-SR	✓	✓	✓	–	–	–
	Lim et al. (2017)	ECF	✓	–	–	–	✓	–
	Özcan et al. (2017)	N/A	✓	–	–	–	–	✓
	Park et al. (2016)	N/A	✓	✓	–	–	–	–
	Hu et al. (2016)	MLCS	✓	✓	✓	✓	–	–
	Baidya e Prakash (2014)	N/A	✓	✓	✓	–	–	–
	Kou et al. (2013)	N/A	✓	–	–	–	–	✓
	Tsai et al. (2010)	MTCS	✓	✓	–	–	–	–
MPQUIC	Mogensen et al. (2019)	SR-Scheduler	✓	–	–	–	–	–
	Viernickel et al. (2018)	SA-Scheduler	✓	–	–	–	–	–
	Rabitsch et al. (2018)	SA-ECF	✓	–	–	–	✓	–
	De Coninck (2017)	LRF	✓	–	–	–	–	–

3.5 RESUMO

Este capítulo apresentou uma revisão dos trabalhos da literatura que visam lidar com o problema de reordenamento de pacotes. Esse é um problema causado pelo recebimento de pacotes fora de ordem e o uso de multicaminhos tende a agravá-lo devido ao uso de caminhos com diferentes características. No contexto de protocolos de transporte multicaminhos, a literatura atribui as técnicas e os métodos para lidar com o problema ao escalonador de pacotes, o qual desempenha um importante papel no desempenho desses protocolos. A revisão realizada acerca desses trabalhos identificou que a maioria deles leva em consideração somente o atraso dos caminhos (latência) e utiliza-o como critério de seleção. Contudo, apenas o atraso não

provê informações suficientes dos estados desses caminhos e outras características devem ser incorporadas como critério de seleção.

4 STOUT: UM ESCALONADOR DE PACOTES BASEADO NA SIMILARIDADE DE CAMINHOS

Este capítulo apresenta o escalonador **STOUT** (do inglês, ***S**imilarity agains**T** packets **OUT**-of-order*) para o protocolo Multicaminhos QUIC. Com base na similaridade de características dos caminhos, o escalonador STOUT seleciona-os para a alocação de dados. Dessa forma, espera-se diminuir os efeitos causados pela heterogeneidade dos caminhos. Este capítulo está organizado conforme segue. A Seção 4.1 apresenta uma visão geral do escalonador STOUT e seus objetivos. A Seção 4.2 detalha a estrutura do escalonador STOUT e as etapas que compreendem o seu processo de escalonamento. A Seção 4.3 descreve o funcionamento do escalonador. A Seção 4.4 resume e conclui o capítulo.

4.1 VISÃO GERAL

O escalonador STOUT tem como objetivo reduzir a latência de transmissões que utilizam o protocolo MPQUIC. Ele se baseia em informações coletadas durante a transmissão de dados para estimar o grau de similaridade entre os caminhos que estão sendo utilizados durante a comunicação. Observa-se que a heterogeneidade dos caminhos é uma das principais causas da entrega de pacotes fora de ordem e, consequentemente, do problema de reordenamento. Neste sentido, transmitir os dados por caminhos com características semelhantes contribui para que a entrega ocorra de forma ordenada e melhore o desempenho do protocolo MPQUIC.

Este trabalho considera o modelo de rede seguindo a arquitetura cliente-servidor conforme conceituado por Kurose (2006). Além disso, assume-se que os hospedeiros possuem duas ou mais interfaces de rede e que a comunicação entre eles ocorre através da Internet. Conforme ilustra a Figura 4.1, a comunicação fim-a-fim na Internet pode transcorrer sobre meios de transmissão guiados e/ou não guiados. Os meios de transmissão guiados, como o padrão Ethernet, são meios de transmissão que geralmente atingem altas taxas de vazão e são menos suscetíveis às interferências de agentes externos à transmissão e atenuações de sinais. Os meios de transmissão não guiados, como o padrão IEEE 802.11 (Wi-Fi), as redes celulares (LTE) e os satélites de comunicação (ComSat), compreendem os meios de transmissão com uma menor taxa de vazão em relação aos meios guiados, maior probabilidade de erros, retransmissões e perdas de pacotes. Isto acontece por se tratarem de ondas eletromagnéticas que se propagam em um meio comum e estarem mais suscetíveis à atenuação do sinal durante a transmissão.

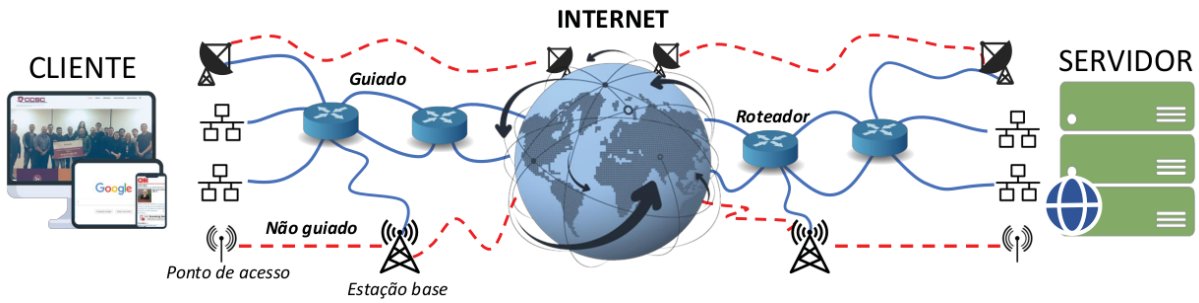


Figura 4.1: Modelo de rede

4.2 COMPOSIÇÃO DO ESCALONADOR STOUT

Esta seção descreve o funcionamento do escalonador STOUT, que consiste em três etapas. A primeira etapa compreende a **coleta de informações** acerca do número de caminhos disponíveis e informações relacionadas às características desses caminhos. Em seguida, a segunda etapa avalia as informações coletadas e, com base na **similaridade dos caminhos**, classifica e seleciona-os para a transmissão de dados. Por fim, a terceira etapa é responsável pela **alocação dos pacotes** na fila de envio dos caminhos selecionados. A Figura 4.2 ilustra a composição do escalonador STOUT, bem como suas três etapas e sua relação como o protocolo MPQUIC.

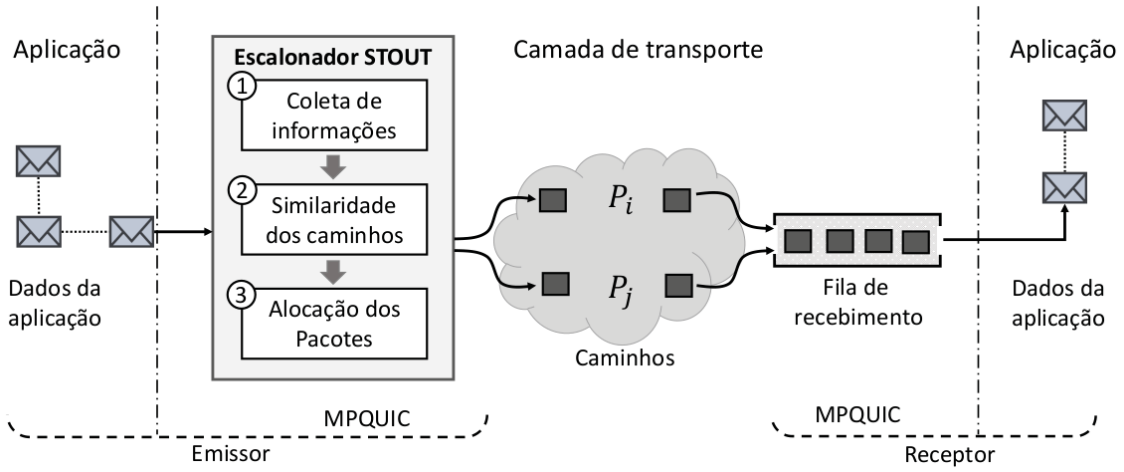


Figura 4.2: Etapas do escalonador STOUT

4.2.1 Coleta de informações

A primeira etapa é responsável pela coleta de informações e estatísticas sobre o estado dos caminhos que estão sendo utilizados. O escalonador STOUT obtém a quantidade de caminhos disponíveis e utilizáveis do componente gerenciador de caminhos do protocolo MPQUIC. Em seguida, ele interage com o controle de congestionamento e coleta 4 informações sobre cada caminho: (*i*) o RTT, (*ii*) o RTT suavizado (do inglês, *Smoothed RTT* - RTT_s), (*iii*) o tamanho máximo do segmento (do inglês, *Maximum Segment Size* - MSS), e (*iv*) informações sobre a taxa de perda de pacotes. O RTT_s representa uma média ponderada das observações do RTT medido (RTT_m) e do próprio RTT suavizado (RTT_s). Devido às flutuações do atraso durante a transmissão de dados, o controle de congestionamento estima o RTT_s para que o cálculo do valor de RTO (do inglês, *Retransmission Timeout*) seja mais consistente. De acordo com Forouzan e Fegan (2009), o RTT_s é obtido da seguinte forma:

$$\begin{aligned} \text{RTT inicial} &\Rightarrow \text{Nenhum valor;} \\ \text{após a primeira medição} &\Rightarrow RTT_s = RTT_m; \\ \text{após qualquer outra medição} &\Rightarrow RTT_s = (1-\alpha)RTT_s + \alpha \times RTT_m; \\ \text{onde} &\Rightarrow \alpha = \frac{1}{8}. \end{aligned}$$

O MSS especifica a maior quantidade de dados, em bytes, que um hospedeiro pode receber em um único segmento. O escalonador STOUT usa o MSS e o RTT_s para calcular a capacidade máxima de cada canal. A taxa de perda é usada para mensurar a qualidade de cada caminho.

4.2.2 Similaridade dos caminhos

Nesta etapa, o escalonador STOUT utiliza os fatores obtidos na etapa de coleta de informações para estimar a similaridade dos caminhos. Inicialmente, os valores de RTT_s são normalizados e são armazenados em um vetor. Em seguida, o escalonador STOUT, baseando-se na Equação de Mathis et al. (1997), aplica a Equação 4.1 para calcular a qualidade de cada caminho, onde a qualidade QLT é dada através da capacidade do canal ($\frac{MSS}{RTT}$) e a razão entre uma constante (1) e as observações da quantidade de pacotes que foram transmitidos desde a última perda (S_{count}). Dessa forma, quanto menor o valor de QLT, maior a qualidade do canal. Os valores obtidos com a Equação 4.1 também são normalizados e armazenados em um vetor.

$$QLT = \left(\frac{MSS}{RTT}\right) \times \left(\frac{1}{S_{count}}\right) \tag{4.1}$$

O protocolo MPQUIC realiza o reconhecimento por pacotes e, dessa forma, tanto o RTT e RTT_s, quanto a QLT são atualizados a cada confirmação recebida. Assim, o escalonador STOUT dispõe de informações precisas sobre o estado dos caminhos. Isto possibilita uma rápida identificação de flutuações desses fatores e, consequentemente, auxilia a tomada de decisão no momento de selecionar os caminhos para a transmissão.

O Algoritmo 1 demonstra o processo de seleção dos caminhos. A Tabela 4.1 resume a lista de varáveis utilizadas. Inicialmente, o Algoritmo recebe como entrada o conjunto de caminhos $\mathcal{P}$ do componente gerenciador de caminhos, as informações sobre o RTT e QLT de cada caminho do conjunto $\mathcal{P}$. Para cada par de caminhos do conjunto $\mathcal{P}$ ($l.1$), o escalonador STOUT normaliza os valores do RTT ($l.2$) e armazena-os em um vetor. O mesmo acontece com a QLT ($l.3$). Após a normalização, os valores que compõem esses vetores são comparados e ordenados de forma crescente ($l.4$). O menor valor encontrado entre os vetores Δ_{RTT} e Δ_{QLT} representa o par de caminhos que apresentam o maior grau de similaridade, seja em relação ao RTT ou QLT. Por fim, como saída o algoritmo devolve o índice dos pares de caminhos ordenados, de acordo com o RTT e com a QLT dos pares analisados ($l.6$).

Tabela 4.1: Lista de variáveis

Variável	**Significado**
S_{count}	Contador de pacotes enviados e recebidos
$\mathcal{P}$	Conjunto de caminhos
p_n	Caminho $p \in \mathcal{P}$
RTT_n	RTT do caminho p_n
QLT_n	Qualidade do caminho p_n
Δ	Diferença entre variáveis
d	Vetor
d_k	k-ésimo índice do vetor d
d_m	Último índice do vetor d

4.2.3 Alocação dos pacotes

Nesta etapa, o escalonador STOUT aloca os segmentos para os caminhos que foram selecionados de acordo com o grau de similaridade encontrado. Esta etapa também defini a

Algoritmo 1 STOUT

Entrada:
Conjunto de caminhos: $\mathcal{P} = \{p_1, p_2, ..., p_n\}$
RTT dos caminhos do conjunto $\mathcal{P}$
QLT dos caminhos do conjunto $\mathcal{P}$
Saída:
Pares de caminhos similares e ordenados de acordo com RTT e QLT

1: **para todo** $p_i, p_j \in \mathcal{P}$ **tal que** $i < j$ **faça**
2: $\quad \Delta_{RTT}[\{i,j\}] \leftarrow \left|\frac{RTT(p_i)-RTT(p_j)}{\max(RTT(p_i),RTT(p_j))}\right|$
3: $\quad \Delta_{QLT}[\{i,j\}] \leftarrow \left|\frac{QLT(p_i)-Q(p_j)}{\max(QLT(p_i),QLT(p_j))}\right|$
4: $\quad \Delta_P[\{i,j\}] \leftarrow \min(\Delta_{RTT}[\{i,j\}], \Delta_{QLT}[\{i,j\}])$
5: **fim para**
6: **devolve** $(d_1, d_2, ..., d_m)$ **tal que** $\Delta[d_k] \leq \Delta[d_{k+1}]$ **para todo** $1 \leq k \leq m-1$

granularidade de alocação, ou seja, quantidade de dados (bytes) que será alocado em cada pacote e que será destinada a cada caminho selecionado. Observa-se o espaço disponível na janela de congestionamento de cada caminho e elas são preenchidas de forma cíclica, isto é, o escalonador STOUT insere dados na janela do caminho que possui menor valor do RTT ou de QLT deste par até que a janela seja preenchida e após isso, passa para o outro caminho.

4.3 FUNCIONAMENTO DO ESCALONADOR STOUT

A título de exemplo, considere dois hospedeiros, um cliente e um servidor Web. Ambos possuem aplicações e suporte ao protocolo MPQUIC, além de múltiplas interfaces de rede disponíveis. Considere também que estes hospedeiros se comunicam através da Internet. Após o estabelecimento da conexão multicaminhos, o cliente envia uma requisição de *download* de um arquivo qualquer ao servidor utilizando o protocolo MPQUIC. Neste momento o escalonador do emissor é acionado pelo protocolo MPQUIC e deve alocar os dados pelos caminhos disponíveis, a fim de entregá-los na mesma ordem em que foram enviados.

Suponha que entre o cliente e o servidor exista um conjunto formado por três caminhos distintos $\mathcal{P} = \{p_1, p_2, p3\}$ e que estes caminhos não compartilham um gargalo. Na etapa de coleta de informações, o escalonador STOUT obtém dados pertinentes a cada caminho individualmente e da totalidade de caminhos estabelecidos entre o cliente e o servidor. Neste cenário, p_1 possui o valor do RTT de 100ms, p_2 de 110ms e p_3 de 70ms. O escalonador STOUT analisa essas informações em pares, neste caso, formados pelos pares (p_1, p_2), (p_1, p_3) e (p_2, p_3). Em seguida, os valores do RTT são normalizados. Como resultado, a normalização dos valores do RTT de (p_1, p_2) apresenta o valor de $0,09$, a do par (p_1, p_3) apresenta o valor de $0,3$ e a do par (p_2, p_3) apresenta o valor de $0,36$. Percebe-se que o menor valor encontrado corresponde ao par (p_1, p_2), o qual possui maior similaridade neste fator. O mesmo princípio é executado para as estimativas QLT dos pares caminhos. Em seguida, o escalonador STOUT faz uma ordenação crescente dos valores normalizados, tanto para o RTT, quanto para o QLT. De acordo com a menor variação encontrada, seja dos valores de RTT, seja dos valores de QLT, o escalonador STOUT classifica os pares de caminhos de forma ordenada e crescente.

Após identificar a característica e o par de caminhos com maior grau de similaridade, o escalonador STOUT aloca uma determinada quantidade de dados pelos caminhos selecionados, neste caso, o caminho p_1 e o caminho p_2, conforme o espaço disponível na janela de congestio-

namento de cada caminho. Assim que os segmentos forem confirmados, o emissor atualiza as informações referentes ao RTT e à QLT de cada caminho para um novo processo de seleção.

4.4 RESUMO

Este capítulo apresentou o escalonador STOUT, um escalonador de pacotes para o protocolo MPQUIC fundamentado na similaridade das características dos múltiplos caminhos disponíveis. O escalonador STOUT tem como objetivo diminuir a latência das transmissões MPQUIC reduzindo o problema de reordenamento de pacotes. Este problema é causado pela entrega de pacotes fora de ordem, perdas de pacotes e retransmissões durante a transmissão de dados. O problema de reordenamento é um desafio para os protocolos multicaminhos, pois eles dividem o tráfego sobre múltiplos caminhos com características heterogêneas. Assim, baseando-se em informação relacionadas ao atraso e à qualidade dos caminhos durante a transmissão, o escalonador STOUT identifica o maior grau de similaridade entre esses fatores e aloca os dados pelos caminhos selecionados de acordo com a janela de congestionamento.

5 AVALIAÇÃO

Este capítulo apresenta uma avaliação de desempenho do escalonador de pacotes proposto STOUT. Ele é comparado com o escalonador LRF (do inglês, *Lowest-RTT-First*), padrão do protocolo MPQUIC, e com o escalonador ECF (do inglês, *Earliest Completion First*), o qual foi implementado adicionalmente ao protocolo MPQUIC para fins comparativos. A avaliação conta com um conjunto de cenários que visam emular as redes heterogêneas, presentes na comunicação cliente/servidor através da Internet. O capítulo está organizado da seguinte forma. A Seção 5.1 descreve a metodologia de avaliação, os cenários de avaliação, os fatores e as métricas utilizadas. A Seção 5.2 apresenta e discute os resultados obtidos. A Seção 5.3 conclui o capítulo

5.1 METODOLOGIA DE AVALIAÇÃO

A fim de avaliar o desempenho do escalonador STOUT, foram construídos 50 cenários distintos. Cada cenário é composto por um cliente e um servidor Web, um hospedeiro N (doravante denominado nó N) e três comutadores de acesso. A topologia de rede empregada representa uma comunicação entre o cliente e o servidor Web, em que o cliente inicialmente estabelece uma conexão MPQUIC através do caminho p_1 e, posteriormente, adiciona o caminho p_2 à comunicação. Além disso, o nó N conecta-se ao comutador *r1* e gera tráfego com destino ao servidor, com o intuito de criar concorrência pelo canal, conforme ilustra a Figura 5.1.

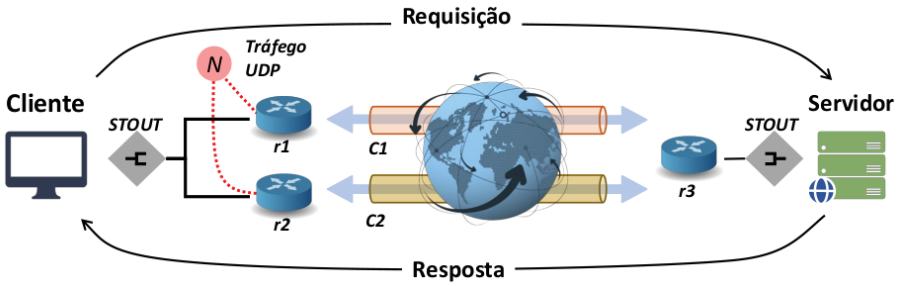


Figura 5.1: Cenário geral de avaliação

5.1.1 Escalonadores avaliados

O **escalonador LRF** é o escalonador padrão para o protocolo MPQUIC. Ele se baseia no RTT dos caminhos como critério de seleção e faz a alocação dos dados de acordo com o espaço disponível na janela de congestionamento de cada caminho. O **escalonador ECF** foi implementado conforme o trabalho de Lim et al. (2017). Ele avalia o comprimento do *buffer* de envio para estimar o tempo total de fluxo (do inglês, *Flow Complete Time* - FCT) para cada caminho. Após preencher a janela de congestionamento de um caminho, o ECF verifica se é realmente benéfico enviar os dados por um caminho mais lento e, pode decidir não os enviar, a fim de aguardar pelo caminho mais rápido. O **escalonador STOUT**, proposto neste trabalho, baseia-se na similaridade das características. Ele seleciona os caminhos para alocação dos dados analisando tanto o RTT, quanto estimativas da qualidade do caminho.

5.1.2 Especificidade dos cenários

Para cada um dos 50 cenários, o caminho p_1 possui características diferentes das apresentadas pelo caminho p_2. Ressalta-se que os cenários de avaliação diferem entre si, pois aplicam diferentes variações de características para o caminho p_1 e p_2. Contudo, um mesmo cenário é utilizado na avaliação comparativa para os diferentes escalonadores analisados. A Tabela 5.1 apresenta os fatores utilizados durante a avaliação efetuada. A construção dos cenários, a topologia utilizada, os fatores e os parâmetros dos caminhos estão fundamentados nos trabalhos de Paasch et al. (2013) e De Coninck (2017). Estes trabalhos avaliam apenas dois tipos de fluxo, com uma pequena carga de *300 KBytes* e uma maior de *20 MBytes*, sem adicionar fluxos extras e gerar maior competitividade pelo canal. A fim de melhor avaliar o comportamento dos escalonadores, foram utilizadas cargas de trabalho com valores de: *300, 500* e *800 KBytes*, *1* a *10 MBytes* e *20 MBytes*.

Tabela 5.1: Fatores e variações

Fator	Min	Max
Capacidade [Mbps]	0,1	100
Atraso (RTT) [ms]	50	400
Atraso de fila [ms]	0	2000
Perdas [%]	0	2,5

5.1.3 Ambiente experimental

Para a realização dos experimentos, utilizou-se um Notebook Dell Inspiron 15 com processador de 1,6 GHz e quatro núcleos, e 8 GB de memória RAM. O Notebook está configurado com o sistema operacional *Ubuntu-Server-16.04 LTS*. A construção do ambiente de rede utilizou o emulador de redes *Mininet* na versão 2.2.1 (Dao et al., 2017). Em conjunto, o emulador de rede *netem* (Hemminger et al., 2005) foi usado para emular as propriedades dos enlaces da rede, como o atraso de propagação e de enfileiramento, e a perda de pacotes. A implementação do protocolo MPQUIC segue a versão apresentada por De Coninck (2017), utilizando a linguagem de programação *Go Lang*, disponibilizada por Clemente (2016) e acessível no *Github*. Para a geração de tráfego adicional do nó *N*, foi utilizado a ferramenta *iPerf* (Hsu e Kremer, 1998).

5.1.4 Métricas

As métricas aferidas para avaliação de desempenho dos escalonadores consideram a perda e a retransmissão de pacotes, bem como o tempo total de conexão e a soma do tempo em que os pacotes que necessitam de reordenamento aguardam em *buffer* no receptor.

Perda de pacotes: a perda de pacotes L_p é computada pela diferença entre o número de pacotes enviados (P_{env}) pelo emissor e o número de pacotes recebidos (P_{rec}) pelo receptor. Como consequência, os serviços utilizados apresentam demora excessiva para o carregamento, interrupções na conexão de rede e/ou perda total de conectividade entre o cliente e o servidor.

$$L_p = P_{env} - P_{rec} \tag{5.1}$$

Retransmissão de pacotes: as retransmissões são o reenvio de uma informação para o receptor. Elas ocorrem sempre que um pacote é perdido durante a transmissão ou após o recebimento de reconhecimentos (*ACKs*) duplicados. O protocolo MPQUIC possui um contador específico para os pacotes retransmitidos. Assim, essa métrica contabiliza o valor total desse contador ao fim da transmissão.

Tempo de conexão: contabiliza o tempo total (T) da conexão, desde o envio do primeiro pacote no processo de *handshake* do protocolo MPQUIC (T_i), até o fim da conexão (T_f).

$$T = T_f - P_i \tag{5.2}$$

Tempo em buffer: quando os pacotes chegam fora de ordem, o receptor armazena os pacotes deste segmento que chegaram em ordem no *buffer* de reordenamento e transmite uma confirmação do último pacote recebido em ordem. Essa métrica calcula o tempo que os pacotes ficam armazenados T_s, aguardando a retransmissão do pacote faltante para realizar o reordenamento e serem entregues para a aplicação.

$$T_s = T_{sf} - P_{si} \tag{5.3}$$

5.2 RESULTADOS

Esta seção descreve os resultados obtidos através das emulações de rede. Para a coleta das informações foram utilizados *scripts* desenvolvidos em *Shell Script*, bem como para a organização e o tratamento dos dados. Em seguida, estes dados foram analisados no software Cran-R (R Core Team, 2013). Para uma melhor compreensão, inicialmente discute-se sobre os resultados alcançados de acordo com a métrica utilizada e em seguida, apresentam-se os gráficos referentes a tal métrica. A análise conta com gráficos do tipo *boxplot*, por apresentarem a distribuição empírica dos dados na amostragem e uma boa comparação visual entre os grupos de amostras. Também é apresentado um diagrama de pontos com as médias dos resultados avaliados e um gráfico de barras que representa a soma do tempo de ocupação do *buffer* de recepção durante a execução dos testes.

5.2.0 Perdas de pacotes

A Figura 5.2 apresenta a taxa de perdas de pacotes em relação as cargas de trabalho utilizadas na emulação para o escalonador LRF. Observa-se que há uma maior variação dos dados em cargas menores e uma proeminência de pontos discrepantes, principalmente para as cargas de *0,3*, *0,5*, *0,8* e *1* MB. Da mesma forma, a Figura 5.3 ilustra os resultados obtidos para o escalonador ECF. Ele apresentou um desempenho marginalmente superior quando comparado ao escalonador LRF em relação à todas as cargas. Contudo, para as cargas menores, o ECF também demonstrou uma maior distribuição dos dados e pontos discrepantes. Os resultados para o escalonador STOUT são similares aos resultados do escalonador ECF e LRF e, pode ser analisado na Figura 5.4. Nota-se que o STOUT demonstra o mesmo comportamento em relação as cargas de trabalho menores. A Figura 5.5 apresenta as médias da taxa de perda, e evidencia a similaridade dos resultados entre os escalonadores. Estes resultados eram esperados, pois na definição dos fatores dos cenários, inseriu-se uma variação para as perdas aleatórias de 0 a 2,5%. Mesmo com a adição de outro fluxo pelo nó N, o que gera maior competição pelo canal e concorrência pelos recursos de rede, os três escalonadores apresentaram desempenho similar.

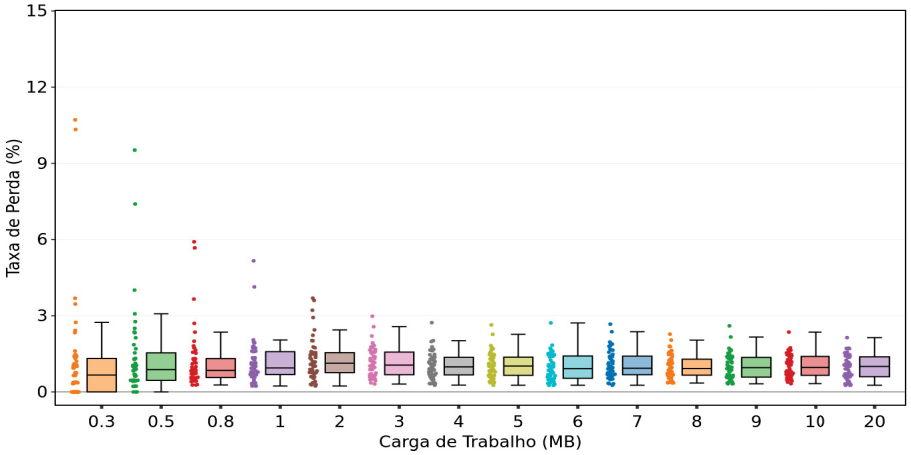


Figura 5.2: Taxa de perda de pacotes do escalonador LRF

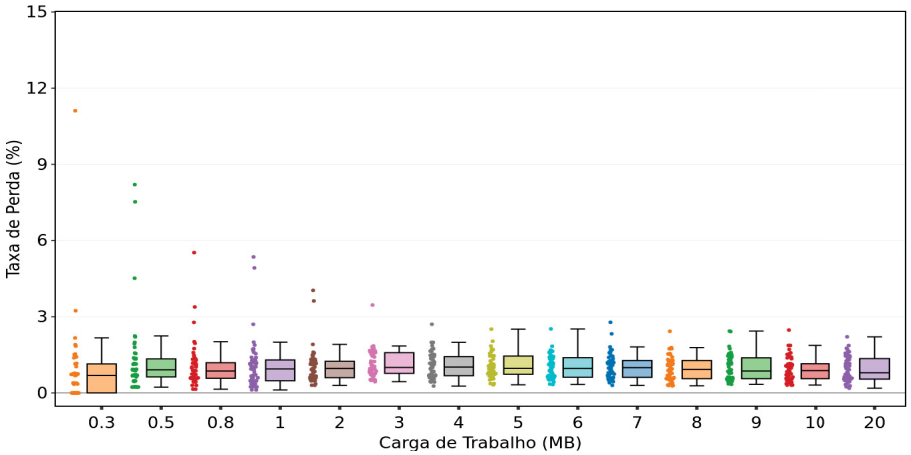


Figura 5.3: Taxa de perda de pacotes do escalonador ECF

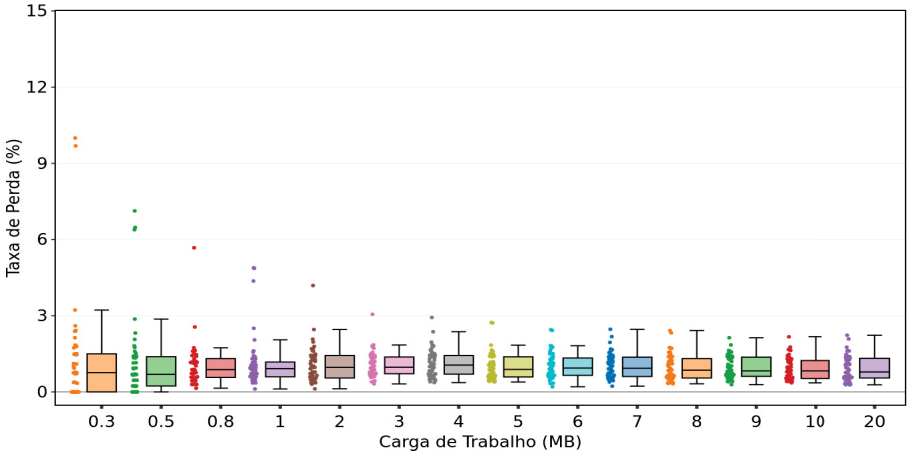


Figura 5.4: Taxa de perda de pacotes do escalonador STOUT

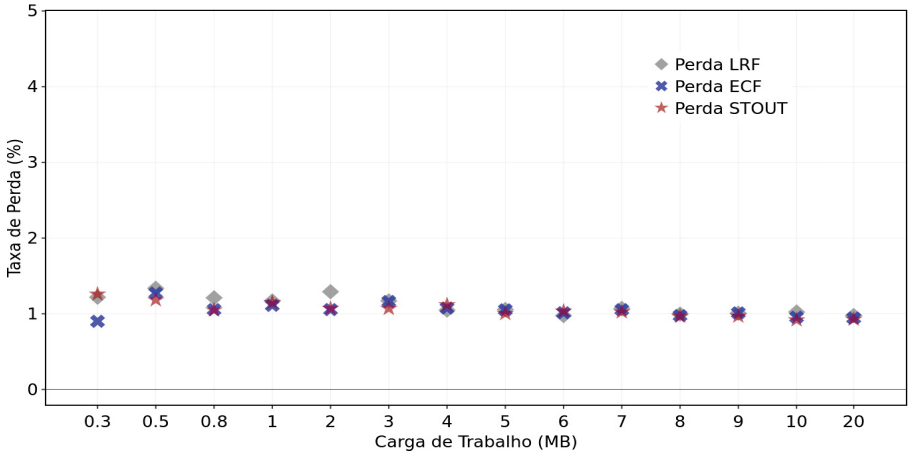


Figura 5.5: Médias das taxas de perdas de pacotes

5.2.0 Retransmissões de pacotes

As retransmissões de pacotes são um reflexo da taxa de perdas e da entrega de pacotes fora de ordem. De acordo com as Figuras 5.6, 5.7 e 5.8, o escalonador LRF apresentou as maiores taxas de retransmissões entre os três escalonadores analisados, principalmente para as cargas de trabalho menores. Este resultado evidencia o problema de escalonar dados utilizando somente o atraso dos caminhos como fator de seleção. Em contrapartida, os escalonadores que utilizam múltiplos fatores para o processo de escalonamento obtiveram melhores resultados. A Figura 5.9 apresenta as médias de taxa de retransmissão obtidas para os escalonadores analisados. Em termos de valores absolutos, o escalonador STOUT e o ECF apresentaram valores semelhantes contudo, o STOUT demonstrou as menores taxas de retransmissão.

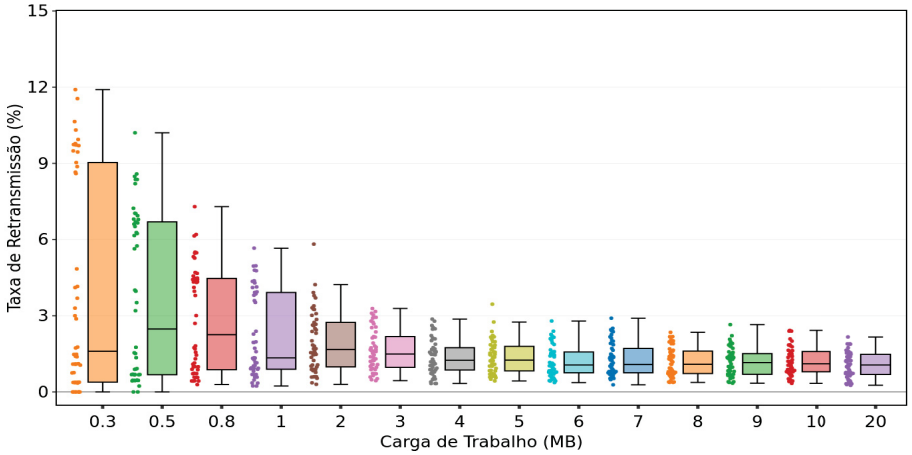


Figura 5.6: Taxa de retransmissão de pacotes do escalonador LRF

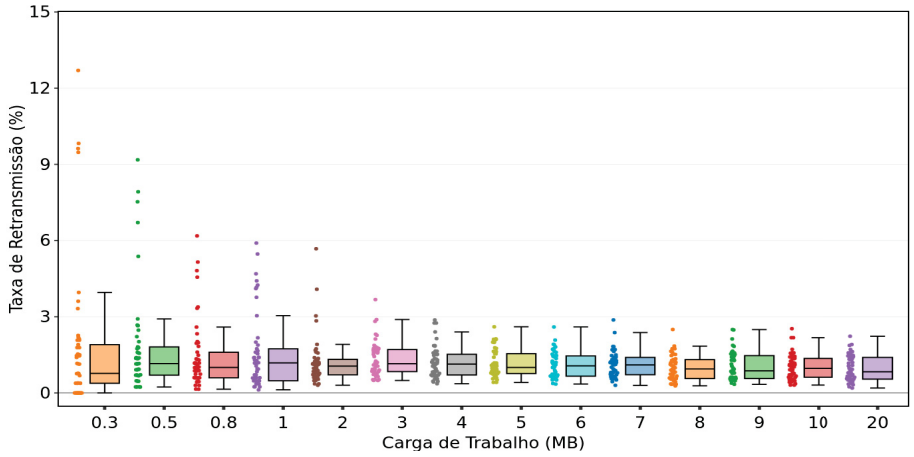


Figura 5.7: Taxa de retransmissão de pacotes do escalonador ECF

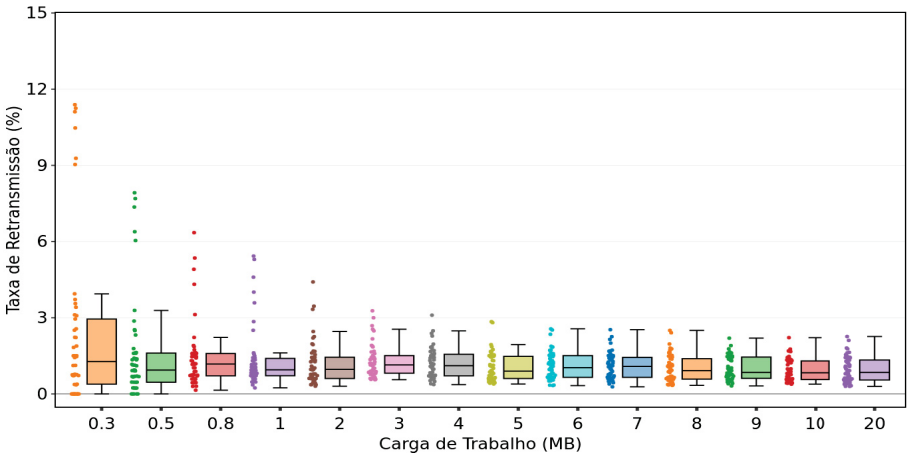


Figura 5.8: Taxa de retransmissão de pacotes do escalonador STOUT

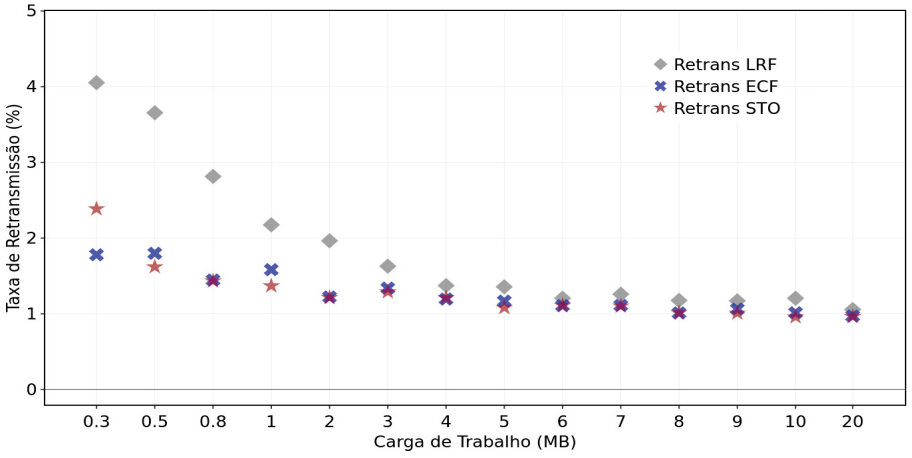


Figura 5.9: Médias das taxas de retransmissões

5.2.0 Tempo total de conexão

As Figuras 5.10, 5.11 e 5.12 apresentam o tempo total de conexão desde o envio do primeiro pacote de requisição do cliente até o recebimento do último pacote enviado pelo servidor, para os escalonadores LRF, ECF e STOUT, respectivamente. Nesta avaliação, o escalonador LRF demonstrou o pior desempenho, principalmente em relação às cargas de trabalho maiores. Esta discrepância fica evidente quando observamos as médias dessa métrica, conforme ilustra a Figura 5.13. O escalonador STOUT apresentou maior desempenho em todas as cargas de trabalho quando comparado ao escalonador LRF, necessitando entre 27% e 40% menos tempo para completar a transferência de dados. Em relação ao ECF, o escalonador STOUT obteve resultados muito similares. Nas cargas menores, o ECF demonstrou ser até 2,4% mais eficiente que o escalonador STOUT. Entretanto, em cargas de trabalho mais altas, o STOUT foi até 7,8% mais eficiente.

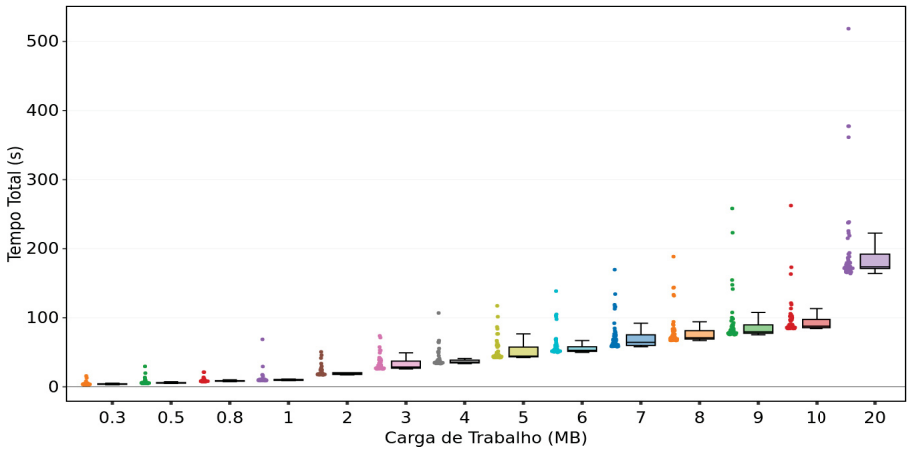


Figura 5.10: Tempo total de conexão do escalonador LRF

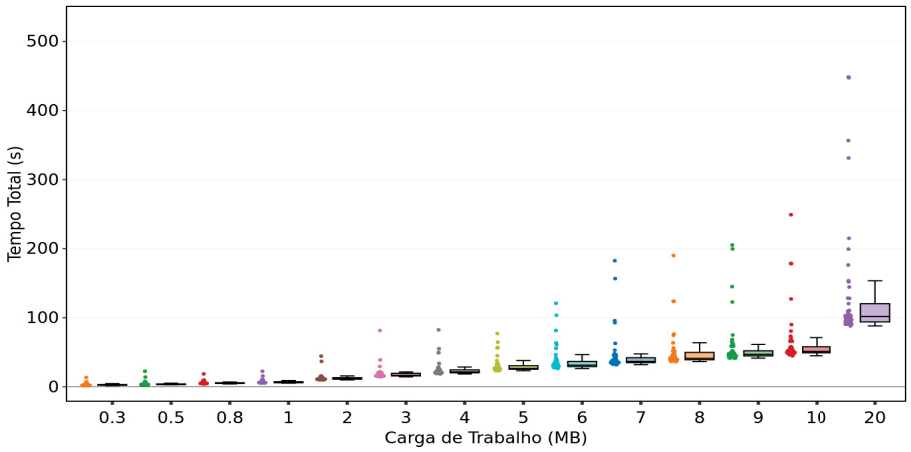


Figura 5.11: Tempo total de conexão do escalonador ECF

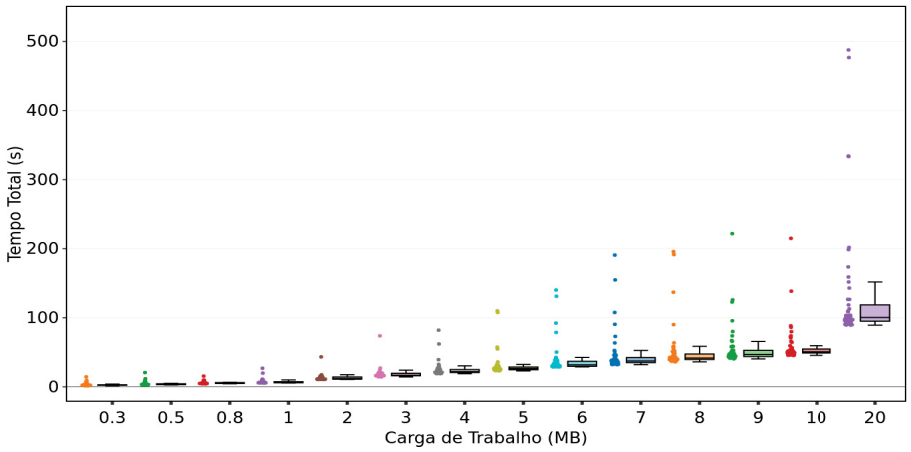


Figura 5.12: Tempo total de conexão do escalonador STOUT

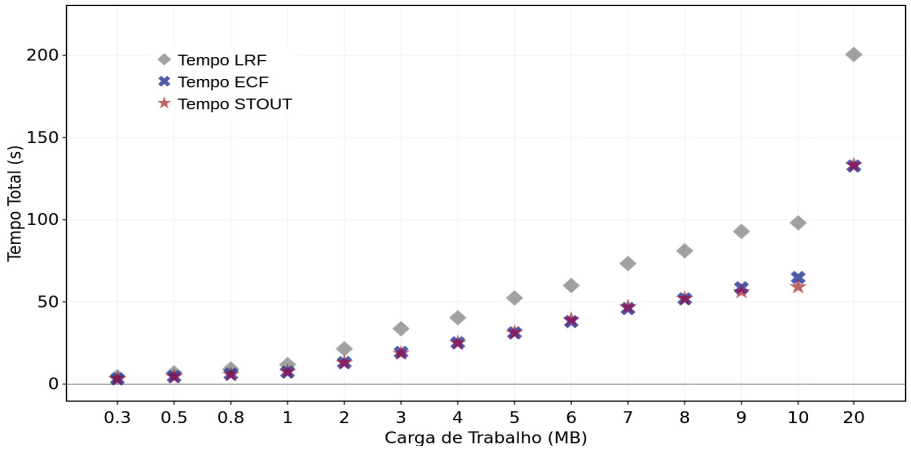


Figura 5.13: Médias dos tempos totais de conexões

5.2.0 Tempo em buffer

A Figura 5.14 apresenta a soma do tempo total em que os pacotes ficaram armazenados em *buffer* durante as emulações. Ambos os escalonadores ECF e STOUT obtiveram bons resultados nas avaliações supracitadas. Contudo, o ECF foi o escalonador que mais onerou recursos de *buffer* para o reordenamento de pacotes. O LRF e STOUT denotam os escalonadores que menos utilizaram este recurso. Este resultado reforça a premissa de que escalonar pacotes baseando-se na similaridade dos caminhos contribui para a entrega de pacotes de forma ordenada e reduz os efeitos causados pela heterogeneidade, aumentando assim o desempenho da transmissão.

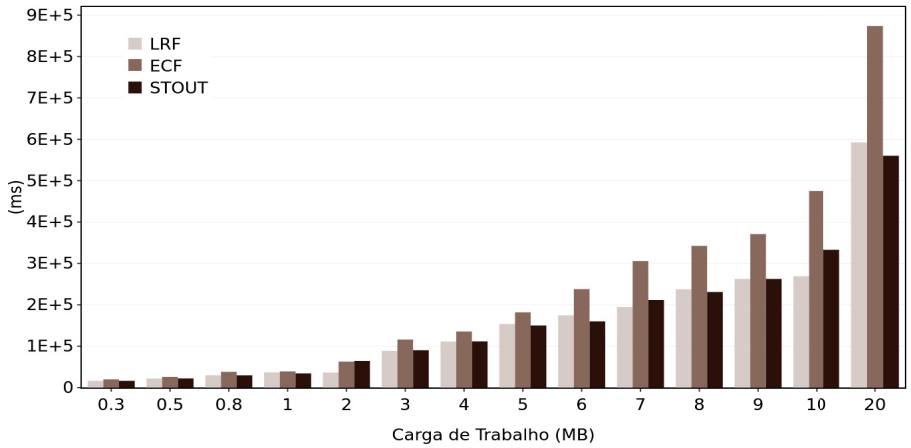


Figura 5.14: Tempo total de ocupação em buffer

5.3 RESUMO

Este capítulo apresentou uma avaliação de desempenho do escalonador de pacotes STOUT, bem como sua comparação com o escalonador LRF e ECF. Para isso, foram construídos 50 cenários distintos com o intuito de emular diferentes redes heterogêneas presentes na comunicação entre cliente e servidor através da Internet. Os resultados obtidos demonstraram que entre os escalonadores avaliados, o LRF atingiu a maior taxa de retransmissões e, por consequência, obteve o maior tempo de transmissão observado. A taxa de perda de pacotes foi similar para os três escalonadores analisados. Os resultados obtidos para o escalonador ECF e STOUT foram parecidos para três das 4 métricas utilizadas. Em contrapartida, o STOUT foi o escalonador que menos ocupou o *buffer* para reordenamento, evidenciando a eficácia de sua proposta.

6 CONCLUSÃO E DIREÇÕES FUTURAS

Os serviços de rede sensíveis à latência vêm crescendo progressivamente na Internet. A popularização de dispositivos com diferentes tecnologias de comunicação (*multihomed*), contribui para o desenvolvimento de soluções que provêm transmissão de dados sobre múltiplos caminhos. Os protocolos de transporte convencionais não oferecem suporte para este tipo de comunicação. O protocolo Multicaminhos QUIC é uma alternativa que oferece comunicação sobre múltiplos caminhos de rede. Apesar de fundamentado em um protocolo simples e não orientado como o UDP, o protocolo MPQUIC apresenta bom desempenho quando comparado com outras propostas de multicaminhos, já consolidadas na literatura. Contudo, a comunicação multicaminhos ainda possui muitos desafios a serem superados. Dentre os mais significativos, a heterogeneidade da rede é um dos principais fatores que causam a chegada de pacotes fora de ordem.

A fim de lidar com esses problemas, este trabalho propôs o escalonador de pacotes STOUT (do inglês, ***S**imilarity agains**T** packets **OUT**-of-order*) para o protocolo MPQUIC. Com o objetivo de reduzir a latência, o STOUT provê um aumento na entrega de pacotes de forma ordenada e, assim, aumenta o desempenho do protocolo MPQUIC. O escalonador STOUT foi comparado com o escalonador LRF, padrão do protocolo MPQUIC, e com o escalonador ECF. Os resultados apontaram que, em geral, os escalonadores apresentam um comportamento similar nos diferentes cenários avaliados. Contudo, com o STOUT os pacotes tendem a chegar em ordem no receptor, diminuindo o tempo em que os pacotes ficam armazenados em *buffer* para reordenação.

6.1 DIREÇÕES FUTURAS

Os resultados obtidos na avaliação do escalonador STOUT demonstraram que a similaridade contribui para a entrega de pacotes de forma ordenada. Apesar de diminuir os efeitos da heterogeneidade dos caminhos, a similaridade de características contribui para reduzir a diversidade dos caminhos, fazendo com que o escalonador priorize caminhos que compartilhem um gargalo e, consequentemente, possuam características similares. Dessa forma, as direções futuras visam o estudo e experimentação de métodos que estimam a correlação dos caminhos, como os coeficientes de Pearson e de Spearman, para aumentar a probabilidade de uso dos caminhos disjuntos durante a transmissão de dados. Há também a necessidade de se avaliar o desempenho dos escalonadores de pacotes com cargas de trabalhos específicas, como o *streaming* de video, além de considerar a qualidade de experiência do usuário (QoE), baseando-se nas métricas já consolidadas na literatura.

REFERÊNCIAS

Albaladejo, M. B., Leith, D. J. e Manzoni, P. (2016). Measurement-based modelling of lte performance in Dublin city. Em *27th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC)*, páginas 1–6. IEEE.

Alheid, A., Doufexi, A. e Kaleshi, D. (2016). A study on MPTCP for tolerating packet reordering and path heterogeneity in wireless networks. Em *Wireless Days (WD)*, páginas 1–7. IEEE.

Aydin, I., Iyengar, J., Conrad, P., Shen, C.-C. e Amer, P. (2012). Evaluating TCP-friendliness in light of Concurrent Multipath Transfer. *Computer Networks*, 56(7):1876–1892.

Baidya, S. H. e Prakash, R. (2014). Improving the performance of multipath TCP over heterogeneous paths using slow path adaptation. Em *International Conference on Communications (ICC)*, páginas 3222–3227. IEEE.

Barré, S., Paasch, C. e Bonaventure, O. (2011). MultiPath TCP-Guidelines for implementers. Relatório técnico, IETF.

Becke, M., Dreibholz, T., Adhari, H. e Rathgeb, E. P. (2012). On the fairness of transport protocols in a multi-path environment. Em *International Conference on Communications (ICC)*, páginas 2666–2672. IEEE.

Belshe, M. e Peon, R. (2012). SPDY Protocol. Internet-Draft draft-mbelshe-httpbis-spdy-00, IETF Secretariat. http://www.ietf.org/internet-drafts/draft-mbelshe-httpbis-spdy-00.txt.

Belshe, M., Peon, R. e Thomson, M. (2015). Hypertext transfer protocol version 2 (http/2). RFC 7540, RFC Editor. http://www.rfc-editor.org/rfc/rfc7540.txt.

Berkleley Lab., E. (2018). QUIC traffic estimate. https://fasterdata.es.net/data-transfer-tools/quic-quick-udp-internet-connections/. Acessado em 22/05/2019.

Berners-Lee, T., Fielding, R. e Frystyk, H. (1996). Hypertext Transfer Protocol. RFC 1945, IETF.

Bonaventure, O. e Seo, S. (2016). Multipath TCP deployments. *IETF Journal*, 12(2):24–27.

Carlucci, G., De Cicco, L. e Mascolo, S. (2015). Http over udp: an experimental investigation of quic. Em *Proceedings of the 30th Annual ACM Symposium on Applied Computing*, páginas 609–614. ACM.

Cerf, V. e Kahn, R. (1974). A protocol for packet network intercommunication. *IEEE Transactions on communications*, 22(5):637–648.

Chavan, S. (2016). Should paced TCP Reno replace CUBIC in Linux? Em *8th International Conference on Communication Systems and Networks (COMSNETS)*, páginas 1–8. IEEE.

Chen, Y.-C., Lim, Y.-s., Gibbens, R. J., Nahum, E. M., Khalili, R. e Towsley, D. (2013). A measurement-based study of multipath TCP performance over wireless networks. Em *Proceedings of the Internet measurement conference*, páginas 455–468. ACM.

Clemente, L. (2016). A quic implementation in pure go. https://github.com/lucas-clemente/quic-go. Acessado em 21/05/2019.

Comer, D. E. (2008). *Computer Networks and Internets*. Prentice Hall Press, Upper Saddle River, NJ, USA, 5th edition.

Coudron, M., Secci, S. e Pujolle, G. (2015). Differentiated pacing on multiple paths to improve one-way delay estimations. Em *International Symposium on Integrated Network Management (IM)*, páginas 672–678. IEEE.

Cui, Y., Li, T., Liu, C., Wang, X. e Kühlewind, M. (2017). Innovating transport with quic: Design approaches and research challenges. *IEEE Internet Computing*, 21(2):72–76.

Dao, N. N., Tran, Q. D., Park, M. e Cho, S. (2017). SDNbox: A portable open-source testbed for SDN study. Em *International Conference on Information and Communication Technology Convergence (ICTC)*, páginas 829–833.

De Coninck, Quentin and Bonaventure, O. (2017). Multipath quic: Design and evaluation. Em *13th International Conference on emerging Networking EXperiments and Technologies (CoNEXT 2017). http://multipath-quic. org*.

De Coninck, Q. e Bonaventure, O. (2019). Multipath Extension for QUIC. Relatório técnico, IETF.

Deng, J.-L. et al. (1982). Control problems of grey systems. *Sys. & Contr. Lett.*, 1(5):288–294.

Dhamdhere, A., Luckie, M., Huffaker, B., Elmokashfi, A., Aben, E. et al. (2012). Measuring the deployment of IPv6: topology, routing and performance. Em *Proceedings of the Internet Measurement Conference*, páginas 537–550. ACM.

Domżał, J., Duliński, Z., Kantor, M., Rząsa, J., Stankiewicz, R., Wajda, K. e Wójcik, R. (2015). A survey on methods to provide multipath transmission in wired packet networks. *Computer Networks*, 77:18–41.

dos Prazeres, T. F., Junior, I. C. L. e de Almada Garcia, P. A. (2010). Análise relacional grey e método de análise hierárquica: Um estudo comparativo aplicado ao caso de movimentação e armazenagem de material siderúrgico. *VII SEGeT–Simpósio de Excelência em Gestão e Tecnologia. Resende/RJ*.

Erman, J., Gopalakrishnan, V., Jana, R. e Ramakrishnan, K. K. (2015). Towards a SPDY'ier mobile web? *IEEE/ACM Transactions on Networking*, 23(6):2010–2023.

Farrington, N. (2009). Multipath TCP under Massive Packet Reordering. *UC San Diego Technical Report*.

Felix, B., Steuck, I., Santos, A., Secci, S. e Nogueira, M. (2018). Redundant packet scheduling by uncorrelated paths in heterogeneous wireless networks. Em *IEEE Symposium on Computers and Communications (ISCC)*, páginas 00498–00503. IEEE.

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

Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P. e Berners-Lee, T. (1999). Rfc 2616: Hypertext transfer protocol–http/1.1.

Ford, A., Raiciu, C., Handley, M. e Bonaventure, O. (2013). TCP Extensions for Multipath Operation with Multiple Addresses. RFC 6824, RFC Editor. http://www.rfc-editor.org/rfc/rfc6824.txt.

Forouzan, B. A. e Fegan, S. C. (2009). *Protocolo TCP/IP-3*. AMGH Editora.

Frömmgen, A., Heuschkel, J. e Koldehofe, B. (2018). Multipath tcp scheduling for thin streams: Active probing and one-way delay-awareness. Em *International Conference on Communications (ICC)*, páginas 1–7. IEEE.

Garcia-Saavedra, A., Karzand, M. e Leith, D. J. (2017). Low delay random linear coding and scheduling over multiple interfaces. *IEEE Transactions on Mobile Computing*, 16(11):3100–3114.

Google (2018a). SPDY: an experimental protocol for a faster web. http://www.chromium.org/spdy/spdy-whitepaper. Acessado em 21/05/2018.

Google, T. (2018b). Estatisticas de acesso web. https://bit.ly/2tregbW. Acessado em 21/02/2019.

Gurtov, A. e Polishchuk, T. (2009). Secure multipath transport for legacy Internet applications. Em *Sixth International Conference on Broadband Communications, Networks, and Systems*, páginas 1–8. IEEE.

Ha, S., Rhee, I. e Xu, L. (2008). CUBIC: a new TCP-friendly high-speed TCP variant. *ACM SIGOPS operating systems review*, 42(5):64–74.

Habak, K., Harras, K. A. e Youssef, M. (2014). OSCAR: A collaborative bandwidth aggregation system. *arXiv preprint arXiv:1401.1258*.

Habak, K., Harras, K. A. e Youssef, M. (2015). Bandwidth aggregation techniques in heterogeneous multi-homed devices: A survey. *Computer Networks*, 92:168–188.

Habak, K., Youssef, M. e Harras, K. A. (2012). G-DBAS: A green and deployable bandwidth aggregation system. Em *Wireless Communications and Networking Conference (WCNC)*, páginas 3290–3295. IEEE.

He, J. e Rexford, J. (2008). Toward Internet-Wide Multipath Routing. *IEEE Network*, 22(2).

Hemminger, S. et al. (2005). Network emulation with NetEm. Em *Linux conf au*, páginas 18–23.

Hsu, C.-H. e Kremer, U. (1998). Iperf: A framework for automatic construction of performance prediction models. Em *Workshop on Profile and Feedback-Directed Compilation (PFDC), Paris, France*. Citeseer.

Hu, B., Xing, L., Wang, Z. e Liu, N. (2016). MLCS: A multi-level correlation scheduling algorithm for multipath transport. Em *International Conference on Information Networking (ICOIN)*, páginas 166–171. IEEE.

Huitema, C. (1995). Multi-homed TCP. Relatório técnico, IETF *Internet-Draft (Expired)*.

Husák, M., Komárková, J., Bou-Harb, E. e Čeleda, P. (2018). Survey of attack projection, prediction, and forecasting in cyber security. *IEEE Communications Surveys & Tutorials*, 21(1):640–660.

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

Iyengar, J. R., Amer, P. D. e Stewart, R. (2006). Concurrent Multipath Transfer using SCTP Multihoming over Independent End-to-End Paths. *IEEE/ACM Transactions on Networking*, 14(5):951–964.

J. Iyengar, E. e M. Thomson, E. (2019). QUIC: A UDP-Based Multiplexed and Secure Transport. Relatório técnico, IETF.

Jiang, H., Liu, Z., Wang, Y., Lee, K. e Rhee, I. (2012). Understanding bufferbloat in cellular networks. Em *Proceedings of the ACM SIGCOMM workshop on Cellular networks: operations, challenges, and future design*, páginas 1–6. ACM.

Kaup, F., Wichtlhuber, M., Rado, S. e Hausheer, D. (2015). Can Multipath TCP save energy? A measuring and modeling study of MPTCP energy consumption. Em *40th Conference on Local Computer Networks (LCN)*, páginas 442–445. IEEE.

Ke, F., Huang, M., Liu, Z., Liu, Q. e Cao, Y. (2016). Multi-attribute aware multipath data scheduling strategy for efficient MPTCP-based data delivery. Em *22nd Asia-Pacific Conference on Communications (APCC)*, páginas 248–253. IEEE.

Khalili, R., Gast, N., Popovic, M. et al. (2013). Opportunistic linked-increases congestion control algorithm for MPTCP.

Kimura, B. Y., Lima, D. C. e Loureiro, A. A. (2017). Alternative scheduling decisions for multipath TCP. *IEEE Communications Letters*, 21(11):2412–2415.

Kou, L., Chen, S. R. e Wang, R. (2013). A MPTCP path selection strategy based on improved grey relational analysis. Em *Applied Mechanics and Materials*, volume 401, páginas 1766–1771. Trans Tech Publ.

Kuhn, N., Lochin, E., Mifdaoui, A., Sarwar, G., Mehani, O. e Boreli, R. (2014). DAPS: Intelligent delay-aware packet scheduling for multipath transport. Em *International Conference on Communications (ICC)*, páginas 1222–1227. IEEE.

Kurose, James F e Ross, K. W. (2006). *Redes de Computadores e a Internet*. Pearson Education Brasil.

Lan, K.-c. e Li, C.-Y. (2012). Improving TCP performance over an on-board multi-homed network. Em *Wireless Communications and Networking Conference (WCNC)*, páginas 2961–2966. IEEE.

Langley, A. e Chang, W.-T. (2016). QUIC Crypto. https://bit.ly/2K1mfVJ. Acessado em 06/06/2019.

Langley, A., Riddoch, A., Wilk, A., Vicente, A., Krasic, C., Zhang, D., Yang, F., Kouranov, F., Swett, I., Iyengar, J. et al. (2017). The QUIC transport protocol: Design and Internet-scale deployment. Em *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*, páginas 183–196. ACM.

Le, T.-A. e Bui, L. X. (2018). Forward delay-based packet scheduling algorithm for multipath TCP. *Mobile Networks and Applications*, 23(1):4–12.

Leung, K.-C., Li, V. O. e Yang, D. (2007). An overview of packet reordering in transmission control protocol (TCP): problems, solutions, and challenges. *IEEE transactions on parallel and distributed systems*, 18(4):522–535.

Li, M., Lukyanenko, A., Ou, Z., Yla-Jaaski, A., Tarkoma, S., Coudron, M. e Secci, S. (2016). Multipath Transmission for the Internet: A Survey. *IEEE Communications Surveys & Tutorials*.

Lim, Y.-s., Nahum, E. M., Towsley, D. e Gibbens, R. J. (2017). ECF: An MPTCP path scheduler to manage heterogeneous paths. Em *Proceedings of the 13th International Conference on emerging Networking EXperiments and Technologies*, páginas 147–159. ACM.

Liu, S., Lei, W., Zhang, W. e Guan, Y. (2017). CMT-SR: A selective retransmission based concurrent multipath transmission mechanism for conversational video. *Computer Networks*, 112:360–371.

Mathis, M., Semke, J., Mahdavi, J. e Ott, T. (1997). The macroscopic behavior of the TCP congestion avoidance algorithm. *ACM SIGCOMM Computer Communication Review*, 27(3):67–82.

Maxemchuk, N. (1975). Dispersity routing in store-and-forward networks.

Miyazaki, E. e Oguchi, M. (2011). Evaluation of middleware for bandwidth aggregation using multiple interface in wireless communication. *International Journal on Advances in Networks and Services Volume 4*.

Mogensen, R. S., Markmoller, C., Madsen, T. K., Kolding, T., Pocovi, G. e Lauridsen, M. (2019). Selective redundant mp-quic for 5g mission critical wireless applications. Em *2019 IEEE 89th Vehicular Technology Conference (VTC2019-Spring)*, páginas 1–5. IEEE.

Nam, H., Calin, D. e Schulzrinne, H. (2016). Towards dynamic mptcp path control using sdn. Em *2016 IEEE NetSoft Conference and Workshops (NetSoft)*, páginas 286–294. IEEE.

Özcan, Y., Guillemin, F., Houzé, P. e Rosenberg, C. (2017). Fast and smooth data delivery using mptcp by avoiding redundant retransmissions. Em *2017 IEEE International Conference on Communications (ICC)*, páginas 1–7. IEEE.

Paasch, C. e Barre, S. (2009-2019). Multipath TCP - linux kernel implementation. https://multipath-tcp.org/. Acessado em 06/02/2019.

Paasch, C. e Bonaventure, O. (2014). Multipath TCP. *Queue*, 12(2):40–51.

Paasch, C., Ferlin, S., Alay, O. e Bonaventure, O. (2014). Experimental evaluation of multipath tcp schedulers. Em *Proceedings of the 2014 ACM SIGCOMM workshop on Capacity sharing workshop*, páginas 27–32. ACM.

Paasch, C., Khalili, R. e Bonaventure, O. (2013). On the benefits of applying experimental design to improve multipath tcp. Em *Proceedings of the ninth ACM conference on Emerging networking experiments and technologies*, páginas 393–398. ACM.

Padhye, J. e Nielsen, H. F. (2012). A comparison of spdy and http performance. https://bit.ly/2Xu5dCH. Acessado em 21/05/2019.

Park, S., Kim, D., Yang, C. M. e Song, C.-J. (2016). A packet scheduling scheme for seamless transmission of life media contents. Em *2016 Eighth International Conference on Ubiquitous and Future Networks (ICUFN)*, páginas 478–480. IEEE.

Pearce, C. e Zeadally, S. (2015). Ancillary impacts of multipath tcp on current and future network security. *IEEE Internet Computing*, 19(5):58–65.

Prasad, R., Dovrolis, C., Murray, M. e Claffy, K. (2003). Bandwidth estimation: metrics, measurement techniques, and tools. *IEEE network*, 17(6):27–35.

Projects, C. (2018). QUIC, a multiplexed stream transport over udp. https://www.chromium.org/quic. Acessado em 24/05/2019.

R Core Team (2013). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria.

Rabitsch, A., Hurtig, P. e Brunstrom, A. (2018). A stream-aware multipath quic scheduler for heterogeneous paths: Paper# xxx, xxx pages. Em *Proceedings of the Workshop on the Evolution, Performance, and Interoperability of QUIC*, páginas 29–35. ACM.

Ramaboli, A. L., Falowo, O. E. e Chan, A. H. (2012). Bandwidth aggregation in heterogeneous wireless networks: A survey of current approaches and issues. *Journal of Network and Computer Applications*, 35(6):1674–1690.

Rescorla, E. (2018). The Transport Layer Security (TLS) Protocol Version 1.3. RFC 8446.

Reserved, S. L. . A. R. (2018). Estatisticas de acesso web. https://www.similarweb.com/website/google.com. Acessado em 21/05/2019.

Saeed, A., Habak, K., Fouad, M. e Youssef, M. (2010). DNIS: a middleware for dynamic multiple network interfaces scheduling. *ACM SIGMOBILE Mobile Computing and Communications Review*, 14(2):16–18.

Sakakibara, H., Saito, M. e Tokuda, H. (2006). Design and implementation of a socket-level bandwidth aggregation mechanism for wireless networks. Em *Proceedings of the 2nd annual international workshop on Wireless internet*, página 11. ACM.

Sarwar, G., Boreli, R., Lochin, E., Mifdaoui, A. e Smith, G. (2013). Mitigating receiver's buffer blocking by delay aware packet scheduling in multipath data transfer. Em *Advanced Information Networking and Applications Workshops (WAINA), 2013 27th International Conference on*, páginas 1119–1124. IEEE.

Secci, S., Pujolle, G., Nguyen, T. M. T. e Nguyen, S. C. (2014). Performance–cost trade-off strategic evaluation of multipath tcp communications. *IEEE Transactions on Network and Service Management*, 11(2):250–263.

Singh, A., Goerg, C., Timm-Giel, A., Scharf, M. e Banniza, T.-R. (2012). Performance comparison of scheduling algorithms for multipath transfer. Em *Global Communications Conference (GLOBECOM), 2012 IEEE*, páginas 2653–2658. IEEE.

Singh, A., Xiang, M., Konsgen, A., Goerg, C. e Zaki, Y. (2013). Enhancing fairness and congestion control in multipath tcp. Em *6th Joint IFIP Wireless and Mobile Networking Conference (WMNC)*, páginas 1–8. IEEE.

Singh, S. K., Das, T. e Jukan, A. (2015). A survey on internet multipath routing and provisioning. *IEEE Communications Surveys & Tutorials*, 17(4):2157–2175.

Syverson, P. (1994). A taxonomy of replay attacks. Relatório técnico, NAVAL RESEARCH LAB WASHINGTON DC.

TecMundo (2019). Hbo go bate recorde no reclame aqui durante exibição de got. https://bit.ly/2W7QBvn. Acessado em 15/05/2019.

Thomson, M. (2019). Version-Independent Properties of QUIC. Relatório técnico, IETF.

Thomson, M. e S. Turner, E. (2019). Using TLS to Secure QUIC. Relatório técnico, IETF.

Tsai, M.-F., Chilamkurti, N., Park, J. H. e Shieh, C.-K. (2010). Multi-path transmission control scheme combining bandwidth aggregation and packet scheduling for real-time streaming in multi-path environment. *IET communications*, 4(8):937–945.

Van Wambeke, N., Exposito, E., Jourjon, G. e Lochin, E. (2008). *End-to-End Quality of Service Over Heterogeneous Networks*, capítulo: Enhanced Transport Protocols, páginas 111–129. Springer Berlin Heidelberg, Berlin, Heidelberg.

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

Wang, X. S., Balasubramanian, A., Krishnamurthy, A. e Wetherall, D. (2014). How speedy is spdy? Em *NSDI*, volume 1, página 1.

Yang, F., Amer, P. e Ekiz, N. (2013). A scheduler for multipath tcp. Em *2013 22nd International Conference on Computer Communication and Networks (ICCCN)*, páginas 1–7. IEEE.

Yang, F., Wang, Q. e Amer, P. D. (2014). Out-of-order transmission for in-order arrival scheduling for multipath TCP. Em *2014 28th International Conference on Advanced Information Networking and Applications Workshops*, páginas 749–752. IEEE.

Yap, K.-K., Huang, T.-Y., Kobayashi, M., Yiakoumis, Y., McKeown, N., Katti, S. e Parulkar, G. (2012). Making use of all the networks around us: a case study in android. *ACM SIGCOMM Computer Communication Review*, 42(4):455–460.

Yedugundla, K., Ferlin, S., Dreibholz, T., Alay, Ö., Kuhn, N., Hurtig, P. e Brunstrom, A. (2016). Is multi-path transport suitable for latency sensitive traffic? *Computer Networks*, 105:1–21.

Zhou, J., Li, Z., Wu, Q., Steenkiste, P., Uhlig, S., Li, J. e Xie, G. (2019). Tcp stalls at the server side: Measurement and mitigation. *IEEE/ACM Transactions on Networking (TON)*, 27(1):272–287.