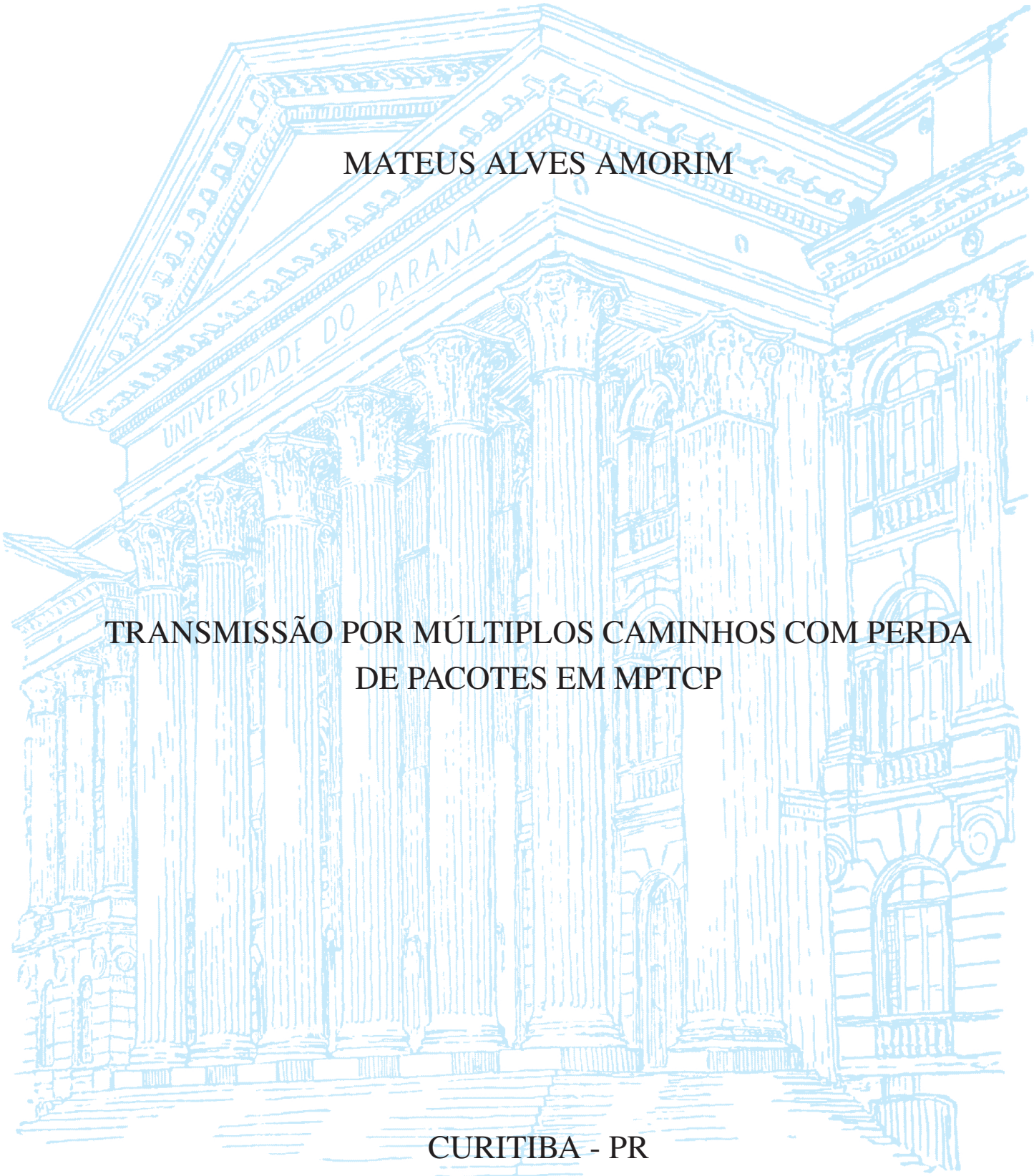
UNIVERSIDADE FEDERAL DO PARANÁ

MATEUS ALVES AMORIM



TRANSMISSÃO POR MÚLTIPLOS CAMINHOS COM PERDA DE PACOTES EM MPTCP

CURITIBA - PR
2018

MATEUS ALVES AMORIM

TRANSMISSÃO POR MÚLTIPLOS CAMINHOS COM PERDA DE PACOTES EM MPTCP

Dissertação apresentada como requisito parcial à obtenção do grau de Mestre em Engenharia Elétrica no Programa de Pós-Graduação em Engenharia Elétrica, setor de Tecnologia, da Universidade Federal do Paraná.

Área de concentração: *Telecomunicações*.

Orientador: Prof. Dr. Eduardo Parente Ribeiro.

CURITIBA - PR
2018

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

A524t Amorim, Mateus Alves
Transmissão por múltiplos caminhos com perda de pacotes em MPTCP / Mateus Alves Amorim. – Curitiba, 2018.

Dissertação - Universidade Federal do Paraná, Setor de Tecnologia, Programa de Pós-Graduação em Engenharia Elétrica, 2018.

Orientador: Eduardo Parente Ribeiro .

1. TCP/IP (Protocolo de rede de computador). 2. Rede de computador – Protocolos. 3. Sistemas de transmissão de dados. I. Universidade Federal do Paraná. II. Ribeiro, Eduardo Parente. III. Título.

CDD: 004.6

Bibliotecário: Elias Barbosa da Silva CRB-9/1894

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

TERMO DE APROVAÇÃO

Os membros da Banca Examinadora designada pelo Colegiado do Programa de Pós-Graduação em ENGENHARIA ELÉTRICA da Universidade Federal do Paraná foram convocados para realizar a arguição da Dissertação de Mestrado de **MATEUS ALVES AMORIM** intitulada: **Transmissão por múltiplos caminhos com perdas de pacotes em MPTCP**, após terem inquirido o aluno e realizado a avaliação do trabalho, são de parecer pela sua APROVAÇÃO no rito de defesa.

A outorga do título 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, 27 de Julho de 2018.

EDUARDO PARENTE RIBEIRO
Presidente da Banca Examinadora (UFPR)

EDGARD JAMHOUR
Avaliador Externo (PUC/PR)

EVELIO MARTÍN GARCÍA FERNÁNDEZ
Avaliador Interno (UFPR)

CARLOS MARCELO PEDROSO
Avaliador Interno (UFPR)

À minha família por me apoiar não só nesta importante caminhada, mas na vida

AGRADECIMENTOS

Em primeiro lugar agradeço a Deus por me permitir acordar a cada dia com saúde e força para transformar os meus sonhos em realidade através do meu trabalho e dedicação.

Em seguida, agradeço a minha amada família, composta por minha esposa Danieli Dyba, meu filho João Pedro e a minha filha recém chegada Gabriela, que me confortam e suportam, além de me cobrir com carinho e atenção que se refletem diretamente na pessoa que sou.

Agradeço imensamente ao Prof. Dr. Eduardo Parente Ribeiro por ter me dado a oportunidade de ser seu orientado. Obrigado pela orientação, apoio e atenção de sempre.

Por fim, agradeço a todos os professores, alunos e funcionários que compõem o Programa de Pós-graduação em Engenharia Elétrica da UFPR. Obrigado pelo convívio e atenção.

RESUMO

A heterogeneidade dos caminhos disponíveis nos dispositivos atuais, como redes WiFi, WiMAX e LTE, causa grande impacto no atraso e na taxa de transferência quando são empregados protocolos de múltiplos caminhos, como o *Multipath TCP* (MPTCP). Um dos principais problemas causadores dessa situação é o atraso decorrente do reordenamento dos pacotes no *buffer* de recebimento. Este trabalho propõe uma análise do impacto na taxa de transferência de uma conexão MPTCP de um caminho degradado por perda de pacotes, estuda um limiar da razão de perda de pacotes a partir do qual é melhor utilizar uma única conexão TCP pelo melhor caminho disponível e, por fim, propõe um método dinâmico para a filtragem do caminho mais degradado. Este trabalho descreve testes em cenários de rede heterogêneas comumente encontrados em dispositivos móveis, compostos por uma conexão WiFi e outra de rede celular, no simulador de redes NS-3.

Palavras-chave: MPTCP, escalonador, multi-abrigado, multi-caminhos, seleção de caminho.

ABSTRACT

The heterogeneity of available paths in the current devices, such as WiFi, WiMAX and LTE networks, has a great impact on delay and throughput when multipath protocols, such as Multipath TCP, are used. One of the main causes of this situation is the delay caused by reordering the packets in the receive buffer. This work proposes an analysis of the impact on the transfer rate of an MPTCP connection of a degraded path by the packet loss. A threshold of packet loss ratio from which it is best to use a single TCP connection with the best available path is studied, and, finaly, a dynamic method for filtering the most degraded path is proposed. This work describes tests in a heterogeneous network scenario commonly seen in mobile devices, consisting of a WiFi connection and a cellular network, in the NS-3 network simulator.

Keywords: MPTCP, scheduler, multihomed, multipath, path selection.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ACRÔNIMOS

ADSL	*Asymmetric Digital Subscriber Line*
WiFi	*Wireless Fidelity*
WiMax	*Worldwide Interoperability for Microwave Access*
TCP	*Transmission Control Protocol*
IP	*Internet Protocol*
SCTP	*Stream Control Transmission Protocol*
SIGTRAN	Derivado de *Signaling Transport*
IETF	*Internet Engineering Task Force*
VoIP	*Voice over Internet Protocol*
CMT	*Concurrent Multipath Transfer*
MPTCP	*Multipath TCP*
API	*Application Programming Interface*
NAT	*Network address translation*
RR	*Round Robin*
LowRTT	*Lowest-RTT-First*
NS-3	*Network simulator* versão 3
RFC	*Request For Comments*
RTT	*Round Trip Time*
SRTT	*Smoothed Round Trip Time*
EMA	*Exponential Moving Average*
EWMA	*Exponential Weighted Moving Average*
RTTVAR	Variação do RTT
RTO	*Retransmission Timeout*
RMT	*Resilient Multipath Transfer*
LTE	*Long Term Evolution*
MACD	*Moving Average Convergence/Divergence*
IANA	*Internet Assigned Numbers Authority*
LIA	*Linked Increases Algorithm*
DWC	*Dynamic Window Coupling*
OLIA	*Opportunistic Linked Increases Algorithm*
CWND	*Congestion Window*
RWND	*Receiver Window*
AIMD	*Additive Increase/Multiplicative Decrease*
HOL	*Head-of-line blocking*
LCM	*Least Common Multiple*
C++	Linguagem de programação compilada multiparadigma
IEEE	*Institute of Electrical and Electronics Engineers*
PPGEE	Programa de Pós-Graduação em Engenharia Elétrica
UFPR	Universidade Federal do Paraná

SUMÁRIO

Capítulo 1

Introdução

A crescente facilidade de acesso à internet por diferentes tecnologias, como por exemplo ADSL, fibra óptica, WiFi, WiMax e redes de telefonia móvel, e o advento de dispositivos com múltiplas possibilidades de conexão à internet, como por exemplo os *smartphones* e os *notebooks*, trouxeram uma gama de possibilidades de conectividade aos usuários finais. Essa característica é conhecida como *multihoming*. A internet desde sua concepção vislumbrava a possibilidade de múltiplos caminhos para o envio dos dados. Porém, na prática, a camada de rede, disponibiliza apenas o melhor caminho entre origem e destino. Assim protocolos da camada de transporte como o TCP e o UDP utilizam apenas este único caminho, independente da disponibilidade de múltiplos caminhos.

Nesse cenário, uma alternativa empregada pela camada transporte foi utilizar múltiplos endereços IP, num cenário de multihome, para que mais de um caminho pudesse ser utilizado. Esta estratégia foi possibilitada inicialmente pelo SCTP (*Stream Control Transmission Protocol*). Originalmente, o SCTP foi projetado como um protocolo de uso geral para transportar mensagens de sinalização telefônica. A definição do protocolo foi feita pelo grupo de trabalho SIGTRAN do IETF (*Internet Engineering Task Force*), que emitiu um documento de padronização do SCTP, RFC 2960, em outubro de 2000 [Stewart et al., 2000]. Porém, o SCTP é um protocolo que não está implementado na maioria dos sistemas, não sendo portanto um padrão em dispositivos finais e *middleboxes*. Por padrão o SCTP não utiliza os caminhos concorrentes, o protocolo troca de caminho apenas quando o caminho primário apresenta muitas perdas de pacotes consecutivas, logo, para aplicações como VoIP e *streaming* de vídeo, outras estratégias podem alcançar melhor desempenho. Existem diversos estudos como [Kelly et al., 2004], [Gavriloff, 2009], [Torres, 2014] e [Antunes et al., 2015] que propõem algoritmos para a escolha do caminho primário baseados no atraso dos caminhos buscando melhorar a eficiência do protocolo em termos da taxa de transferência e do atraso para a aplicação.

O SCTP-CMT (*Concurrent Multipath Transfer*) [Iyengar et al., 2006] é uma extensão do SCTP proposta para alcançar balanceamento de carga fim-a-fim, oferecendo suporte completo ao *multihoming* com a utilização simultânea de todos os caminhos disponíveis. Possibilita o compartilhamento de carga resultando no aumento da taxa de transferência. Originalmente nenhuma lógica para definição de pesos de cada caminho é utilizada e, por consequência, não há uma tratativa especial para sistemas com características heterogêneas entres os caminhos disponíveis.

Em oposto ao que vimos no caso do SCTP, o *Multipath TCP* (MPTCP) é uma extensão para o protocolo TCP proposta recentemente e atualmente implementada em muitos sistemas finais, como por exemplo o sistema operacional Linux [Paasch et al., 2013] e os sistemas MAC OS X e IOS da empresa *Apple* [Mehani et al., 2015]. O seu desenvolvimento teve como base

a compatibilidade com o TCP e com os protocolos das camadas adjacentes. Para tanto, teve 3 (três) objetivos de compatibilidade estabelecidos [Ford et al., 2011]: 1 - As aplicações devem utilizar a API de *socket* já existente para o TCP na utilização do MPTCP; 2 - O protocolo deve ser compatível com a rede onde está sendo implantado, ou seja, com a internet, podendo atravessar *middleboxes* de mercado, como firewalls, NATs e proxies; e 3 - O protocolo deve manter a equidade entre os usuários da rede, coexistindo amigavelmente com outros fluxos TCP padrão em situações de gargalo.

Outro ponto importante dos protocolos de múltiplos caminhos, não só do MPTCP, é que a taxa de transferência agregada deve ser melhor do que a conseguida por um único fluxo TCP quando da utilização do melhor caminho dentre os disponíveis. Caso contrário, a sua utilização não apresenta benefício, a não ser a resiliência.

Atualmente, esse último é o maior desafio para os protocolos de múltiplos caminhos, pois a multiplicidade de caminhos disponíveis aos dispositivos finais também apresenta grande heterogeneidade nas suas características [Li et al., 2016]. Redes heterogêneas, com diferentes valores de atraso, erro de pacotes e largura de banda, impactam significativamente nas taxas de transferência e no atraso conseguidos pela aplicação que utiliza serviço de transporte confiável, pois necessita de reordenamento dos pacotes recebidos antes que sejam entregues à aplicação.

Como mencionado anteriormente, o desafio de conseguir melhores taxas de transferência agregadas com menores atrasos nos sistemas multi-abrigados é um tema muito pesquisado atualmente. Nos protocolos de transporte para sistemas multi-abrigados com transferência multi-caminhos, os principais componentes de suas arquiteturas são o gerenciador de caminhos disponíveis, o controle de congestionamento e o escalonador. Destes, o escalonador é o componente que mais influencia na taxa de transferência agregada e nos atrasos vistos pela aplicação.

As implementações existentes hoje para os protocolos SCTP-CMT e MPTCP, que utilizam múltiplos caminhos concorrentes, possuem como padrão escalonadores baseados nos algoritmos de escalonamento *Round Robin* (RR) e *Lowest-Delay-First* (*LowRTT*). O primeiro alterna entre os caminhos disponíveis quando do envio de dados e o outro leva em consideração o menor atraso estimado quando da decisão de qual caminhos utilizar. Somente o algoritmo escalonador do *LowRTT* considera uma característica do caminho para a decisão, o que permite melhor desempenho quando comparado ao RR, mas ainda assim tem baixo desempenho em redes muito heterogêneas como demonstrado em [Paasch et al., 2014].

Por esse motivo, essa é uma área que tem despertado grande interesse em pesquisas tanto do meio acadêmico como da indústria. Ainda não foi encontrado um escalonador mais eficiente em diferentes cenários de redes heterogêneas. Por isso, um mecanismo escalonador ou um método dinâmico, baseado nas características dos caminhos, que se adapte aos diferentes cenários de rede permitindo alcançar maiores taxas de transferência agregada, é assunto recorrente nas pesquisas atuais sobre o tema.

1.1 Objetivo

Este trabalho aborda o desafio de alcançar melhores taxas de transferência agregada com protocolos de transporte confiáveis para transferência multi-caminhos em cenários de redes heterogêneas. Isso porque os algoritmos escalonadores implementados nos sistemas atuais não são capazes de alcançar desempenho satisfatório em ambientes com caminhos de características heterogêneas, prejudicando aplicações que demandem um maior desempenho da taxa de transferência agregada.

O objetivo geral deste trabalho é realizar um estudo sobre o protocolo de transporte para sistemas multi-abrigados *Multipath TCP* (MPTCP), focando no componente de escalonamento; analisar o impacto na taxa de transferência de uma conexão MPTCP de um caminho degradado por perda de pacotes; estudar um limiar da razão de perda de pacotes a partir do qual é melhor utilizar uma única conexão TCP pelo melhor caminho disponível. E por fim, propor um método dinâmico para a filtragem do caminho mais degradado baseado nas características dos caminhos.

1.2 Contribuição

Como contribuição este trabalho faz uma análise do limiar de razão de perda de pacotes a partir do qual é melhor utilizar somente o melhor caminho em um cenário de rede heterogênea e implementa um método dinâmico para filtragem dos caminhos disponíveis para o componente escalonador baseado na taxa de transferência do MPTCP e nas taxas de transferência estimadas dos subfluxos TCP que compõem a conexão MPTCP. O método de filtragem proposto, o *Round Robin* e o *LowRTT* foram implementados no simulador NS-3 para validação e comparação dos resultados de taxa de transferência agregada. Para tanto, essa pesquisa estabeleceu cenários de simulação baseados em características de ambientes de redes heterogêneas comumente encontrados em dispositivos móveis, compostos por uma conexão WiFi e outra de rede celular.

1.3 Estrutura do documento

Este trabalho está organizado da seguinte forma: o capítulo 2 apresenta uma visão geral sobre protocolos de transporte para sistemas multi-abrigados, a seguir, detalha a arquitetura do MPTCP com maior foco no componente escalonador e nos algoritmos escalonadores *Round Robin*, *LowRTT* e redundante para o MPTCP, e apresenta um levantamento dos principais problemas das transferências multi-caminhos.

O capítulo 3 apresenta um estudo do estado da arte sobre métodos para mitigação dos problemas relativos aos protocolos de sistemas multi-abrigados relacionados ao tema deste trabalho, com foco nas estratégias de escalonamento.

O capítulo 4 apresenta a proposta e o contexto para aplicação dessa pesquisa, bem como sua justificativa com base na análise do limiar da razão de perda de pacotes, apresenta o método de filtragem de caminhos propostos e as possíveis limitações decorrentes da arquitetura do protocolo.

O capítulo 5 discorre sobre a metodologia empregada e a implementação do método de filtragem de caminhos propostos, sobre a concepção dos cenários de simulação e seus atributos baseados em ambientes de redes heterogêneas e, por último, sobre as métricas utilizadas para avaliação e comparação dos resultados obtidos.

O capítulo 6 apresenta de maneira organizada todos os resultados obtidos das simulações nos cenários adotados, compara os resultados baseado nas métricas de desempenho e discute os resultados.

E, por fim, o capítulo 7 encerra apresentando as conclusões finais do trabalho.

Capítulo 2

Fundamentação Teórica

Este capítulo introduz a base dos conceitos necessários para o transcorrer do trabalho. O capítulo está organizado da seguinte forma: a Seção 2.1 apresenta termos e definições necessários para o prosseguimento do trabalho; a Seção 2.2 traz uma visão geral sobre protocolos de transporte para sistemas multi-abrigados, apresenta a arquitetura básica do MPTCP com maior detalhamento do componente escalonador e dos algoritmos escalonadores *Round Robin*, *LowRTT* e redundante para o MPTCP; a Seção 2.3 apresenta um breve levantamento dos principais problemas da transferência multi-caminhos; e, por fim, a Seção 2.4 apresenta um resumo do capítulo.

2.1 Termos e Definições

Antes de começarmos a falar de protocolos para sistemas multi-abrigados precisamos definir o que é um sistema multi-abrigado. Assim, vamos de início definir alguns conceitos e apresentar alguns termos necessários para o prosseguimento do trabalho. Destacamos que os termos e definições aqui utilizadas são aqueles definidos pelas [1]RFC 6128 [Ford et al., 2011], a RFC 6824 [Ford et al., 2013] e a RFC 6897 [Scharf e Ford, 2013] que especificam respectivamente a arquitetura, a própria extensão e a interface para a aplicação do protocolo MPTCP.

Caminho: uma ligação entre a origem e o destino que, dependendo do protocolo da camada de transporte, pode ser identificada como uma 4-tupla composta por endereço de origem/porta de origem e endereço de destino/porta de destino.

Fluxo: refere-se aos pacotes de uma dada aplicação partindo de uma origem e chegando a um destino.

Subfluxo: um fluxo padrão do protocolo TCP operando sobre um único caminho que se constitui como um elemento de uma conexão de múltiplos caminhos. Define assim o nível de subfluxo.

Conexão multi-caminhos: conjunto de um ou mais subfluxos. Estabelece uma relação direta para o fluxo de dados entre as aplicações de origem e de destino. Definindo assim o nível de dados ou nível de conexão.

[1]RFC (*Request for Comment*) são documentos técnicos desenvolvidos e mantidos pelo IETF (*Internet Engineering Task Force*), instituição que especifica os padrões que serão implementados e utilizados na internet.

ID da conexão ou *token*: um identificador local exclusivo dado a uma conexão multi-caminhos por um *host*.

Ciclo de escalonamento: processo de distribuição de pacotes para os caminhos disponíveis. Pode ser controlado pelo número de pacotes disponíveis no *buffer* de envio, pela disponibilidade de envio dos caminhos ou pela estratégia de escalonamento.

***Round Trip Time* (RTT):** tempo transcorrido desde o envio de um segmento por um subfluxo até o recebimento do reconhecimento do mesmo.

Muitos protocolos, como o TCP e o SCTP, calculam o valor do RTT como especificado na RFC 6298, sendo estimado a cada segmento enviado e com o seu valor mudando conforme as variações das características dos caminhos. Por ser muito variável e sujeito a grandes desvios, muitos protocolos preferem utilizar a forma suavizada do atraso, conhecida como *Smoothed Round Trip Time*, SRTT, que é uma média móvel exponencial - EMA (*Exponential Moving Averages*) do RTT. Ao se estabelecer uma conexão, tanto o RTT como SRTT não possuem nenhum valor atribuído. Quando da primeira estimativa do valor do RTT, o SRTT passa a valer $SRTT = RTT$. A partir desse ponto o valor do SRTT é atualizado de acordo com,

$$SRTT_i = (1 - \alpha) \times SRTT_{(i-1)} + \alpha \times RTT_i, \tag{2.1}$$

onde o valor de α é determinado na implementação, porém o valor comumente utilizado nas implementações para a internet é $\alpha = 0.125$.

Muitas implementações existentes utilizam, além do RTT e do SRTT, o valor do desvio do RTT, denotado como RTTvar. Assim como o SRTT, o RTTvar também utiliza o RTT no seu cálculo, logo no início da conexão não possui nenhum valor atribuído. Quando da primeira estimativa do valor do RTT, o valor do RTTvar passa a ser $RTTvar = RTT/2$. A partir desse ponto o valor do RTTvar é atualizado de acordo com,

$$RTTvar_i = (1 - \beta) \times RTTvar_{(i-1)} + \beta \times |SRTT - RTT|, \tag{2.2}$$

onde, assim como ocorreu para o valor de α, o valor de β depende da implementação, porém o valor comumente utilizado é $\beta = 0.25$.

Com os valores do SRTT e do RTTvar é possível calcular o valor do RTO (*retransmission timeout*). E diferente do ocorrido com o SRTT e o RTTvar, o RTO tem um valor inicial definido em 1 segundo de acordo com a RFC 6298 [Paxson et al., 2011], que diminuiu o tempo inicial do RTO anteriormente definido na RFC 1122 [Braden, 1989] como 3 segundos. A partir disso, o valor do RTO é atualizado de acordo com,

$$RTO = SRTT + 4 \times RTTvar. \tag{2.3}$$

Observando a equação (2.3) percebemos que o valor do RTO corresponde ao valor do último SRTT estimado adicionado de valor de guarda que, neste caso, é igual a quatro vezes o desvio do RTT.

Sistema multi-abrigado (*multihomed*): um sistema que possui mais do que um caminho disponível para a comunicação com a internet, com endereços IP distintos.

Transmissão multi-caminhos: transmissão em que dois ou mais caminhos estão disponíveis para envio de dados entre origem e destino. Não necessariamente os caminhos precisam ser disjuntos, ou seja, eles podem compartilhar enlaces entre a origem e o destino.

Assim como os caminhos não necessariamente estão associados a diferentes interfaces de rede, dado que uma interface pode estar associada a mais do que um endereço de rede. Porém, pelo menos um dos endereços deve diferir entre os múltiplos caminhos.

Ainda segundo [Ford et al., 2011], a utilização dos múltiplos caminhos na transmissão tem como seus principais objetivos aumentar a resiliência quanto a falhas na rede e aumentar a taxa de transferência.

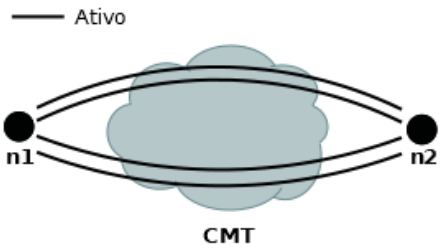


Figura 2.1: Transferência Multi-caminhos Concorrentes - CMT

***Concurrent Multipath Transfer* (CMT):** transferência multi-caminhos que agrega a capacidade de transmissão dos caminhos disponíveis para transmitir em paralelo os dados de um único fluxo de dados, com o objetivo de melhorar a taxa de transferência, como definido em [Iyengar et al., 2006] e representado na Figura 2.1.

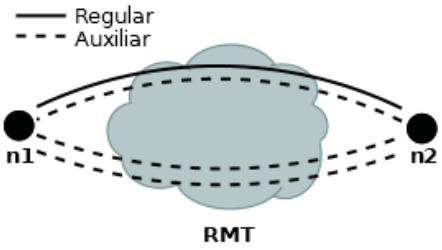


Figura 2.2: Transferência Multi-caminhos Resilientes - RMT

***Resilient Multipath Transfer* (RMT):** transferência multi-caminhos que utiliza os caminhos de forma redundante com o objetivo de aumentar a resiliência [Bonaventure et al., 2015]. Os caminhos regulares são utilizados para transmitir os dados, já os caminhos auxiliares (*backup*) são utilizados apenas na falha de todos os caminhos regulares. Nesse tipo de transferência multi-caminhos os dados também podem ser transmitidos em paralelo por múltiplos caminhos, mas sempre priorizando a resiliência. A Figura 2.2 apresenta um cenário de transferência multi-caminhos resilientes.

Redes homogêneas: cenário de redes onde as características como taxa de transferência, atraso e taxa de perda de pacotes nos caminhos são semelhantes e não tende a variar dinamicamente. Esse tipo de cenário de redes homogêneas é muito comum em redes de grandes *data centers*, onde os dispositivos de rede possuem grande desempenho e são ligados a redes de alta velocidade e com baixa latência. Transferência multi-caminhos em redes homogêneas permite melhorar as taxas de transferência e alcançar a alta disponibilidade.

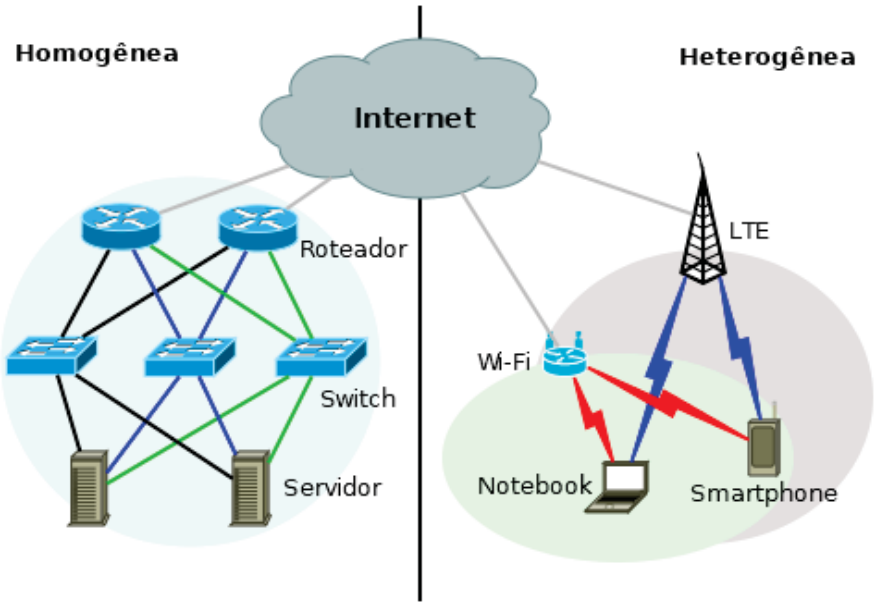


Figura 2.3: Representação de rede homogênea e heterogênea

Redes heterogêneas: cenário de redes onde as características como taxa de transferência, atraso e taxa de perda dos pacotes nos caminhos são muito diferentes, podendo variar dinamicamente. Esse tipo de cenário de redes heterogêneas é o encontrado em redes com dispositivos móveis como *notebooks*, *smartphones* e *tablets*, que comumente possuem conectividade através de redes sem fio *WiFi* e *LTE*, por exemplo, que possuem características bem distintas.

2.2 Protocolos para transferência multi-caminhos

Muito embora o protocolo TCP seja amplamente utilizado na internet, ele não foi projetado para ter suporte a transferências por multi-caminhos. Na verdade, o TCP estabelece uma conexão ponto-a-ponto com somente um par de endereços IP de origem e destino, não podendo ser associado a um novo endereço IP sem que a conexão seja reestabelecida. Para realizar essa tarefa é necessário um protocolo de transporte projetado especificamente para esse fim, ou seja, não basta ter um sistema multi-abrigado com múltiplas interfaces de acesso à internet disponíveis. Esse sistema multi-abrigado necessita de um protocolo de transporte para transferência por multi-caminhos.

Buscando esse objetivo, muitos foram os protocolos propostos para realizar a transferência por multi-caminhos, nas camadas de enlace, rede, transporte, aplicação ou *cross layers*, como podemos verificar no levantamento do estado da arte apresentado no artigo [Li et al., 2016]. Porém os únicos protocolos da camada de transporte que até o presente momento foram padronizados pelo IETF são o SCTP-CMT [Iyengar et al., 2006] e o MPTCP [Ford et al., 2013]. Os dois protocolos apresentam muitas diferenças nas suas arquiteturas, contudo enfrentam os mesmos problemas inerentes à transferência por multi-caminhos, como o reordenamento de pacotes, a justiça e outros.

2.2.1 *Stream Control Transmission Protocol* (SCTP)

A definição do protocolo SCTP padrão foi feita pelo grupo de trabalho SIGTRAN do IETF (Internet Engineering Task Force), que emitiu um documento de padronização do

SCTP, RFC 2960, em outubro de 2000 [Stewart et al., 2000]. Originalmente foi projetado como um protocolo de uso geral para transportar mensagens de sinalização telefônica. Em sua concepção o SCTP não é um protocolo de transferência por multi-caminhos concorrentes. Na verdade, o protocolo por padrão troca de caminho apenas quando o caminho primário apresenta muitas perdas de pacotes consecutivas (STEWART, 2007), e isso acaba sendo inaceitável para transferências de dados como *VoIP* ou *streaming* de vídeos, uma vez que a detecção da falha demora no mínimo 15 segundos para ocorrer.

Alguns trabalhos baseados no SCTP padrão buscaram melhorar o desempenho do protocolo com relação ao atraso fim-a-fim e a taxa de transferência, destacando o apresentado em [Kelly et al., 2004] que propõe o algoritmo *Delay-centric*, baseado no atraso, para as trocas do caminho primário. Ainda baseado no atraso, em [Gavriloff, 2009] é proposto a adição de um tempo de guarda para evitar a instabilidade causada pelas trocas excessivas de caminho primário, principalmente em situações de gargalo. Por último, os trabalhos [Torres, 2014] e [Antunes et al., 2015] apresentam modificações para o Delay-centric adicionando o fator preditivo através do método MACD (Moving Average Convergence / Divergence), analisando a tendência do atraso e fazendo a troca do caminho primário antes que o aumento do atraso ocorra.

O SCTP-CMT (*Concurrent Multipath Transfer*) [Iyengar et al., 2006] é uma extensão do SCTP proposta para alcançar balanceamento de carga fim-a-fim, oferecendo suporte completo ao multihoming com a utilização simultânea de todos os caminhos disponíveis, atingindo o compartilhamento de carga e aumentando o desempenho.

O SCTP surgiu muito antes do MPTCP, mas, ao contrário do SCTP, o MPTCP já desfruta de um grande alcance. Muito disso se deve a alguns problemas decorrentes da concepção inicial do SCTP:

- O SCTP não previa manter a compatibilidade com o TCP, protocolo que junto do IP compõe a base da internet atual. Por esse motivo, muitas *middleboxes* com serviços como *Firewall*, *Proxy* e *NAT* não processam os pacotes SCTP e os descartam.
- A API para as aplicações do SCTP é diferente da API padrão do TCP. Logo, aplicações teriam que usar a API específica para utilização do protocolo. Então mesmo que as *middleboxes* e os sistemas operacionais implementem o SCTP, as aplicações existentes hoje não usariam o protocolo por falta de compatibilidade. Isso gera um círculo vicioso em que os serviços de rede não usam o SCTP porque as aplicações não o utilizam e vice-versa.

Como consequência desses problemas, o SCTP não é amplamente aplicado, exceção feita para sua finalidade original, a sinalização em redes telefônicas. Por esses motivos, o SCTP não será o foco do nosso trabalho. Iremos concentrar nossos esforços no protocolo MPTCP que não apresenta os problemas listados nessa subseção. Entretanto, muitas das contribuições deste trabalho são aplicáveis também ao SCTP-CMT.

2.2.2 *Multipath TCP* (MPTCP)

O protocolo MPTCP [Ford et al., 2013] foi desenvolvido com o objetivo de ser uma extensão do protocolo TCP capaz de transferir dados através de múltiplos caminhos. O MPTCP é o esforço de padronização mais recente de um protocolo de transporte multi-caminhos confiável para a Internet. Resultado do grupo de trabalho MPTCP da IETF[2] que estuda o assunto desde

[2]https://tools.ietf.org/wg/mptcp/

2009 e tem como responsabilidade a padronização do protocolo quanto a aspectos da arquitetura, dos modos de operação, do controle de congestionamento e da segurança.

Como já mencionado no capítulo 1, um dos principais desafios do MPTCP como protocolo de transferência multi-caminhos é conseguir um melhor desempenho em termos de taxa de transferência e latência do que um único fluxo TCP padrão. Uma conexão MPTCP permite o envio de dados bidirecional fim-a-fim assim como o TCP, sem necessidade de qualquer alteração a nível de aplicação. Além de permitir que os nós utilizem os diferentes caminhos e endereços IP para transmitir pacotes pertencentes a uma mesma conexão.

Para as camadas inferiores, um subfluxo MPTCP é visto como um fluxo TCP padrão, mas para a camada de transporte esse subfluxo pertence a uma única conexão composta por outros subfluxos. Isso se dá através do campo *Options* do cabeçalho TCP, utilizado para identificar os subfluxos do MPTCP. Todas as operações do MPTCP são sinalizadas através do campo *Options*. De forma mais específica, a *Options* TCP 30 é reservada pela *Internet Assigned Numbers Authority* (IANA)[3] para uso exclusivo pelo MPTCP. No estabelecimento de um novo fluxo TCP, uma *Options* MP-CAPABLE é incluída no pacote de sincronização inicial (SYN). Se o nó que receber o SYN suportar e optar por negociar o protocolo MPTCP, ele também responde com uma *options* MP-CAPABLE no pacote SYN-ACK. As chaves trocadas dentro deste *handshake* são usadas no futuro para autenticar a adição ou a remoção de outras sessões TCP (subfluxos) nessa mesma conexão MPTCP estabelecida. A Figura 2.4 representa os processos de estabelecimento da conexão MPTCP e adição de um novo subfluxo.

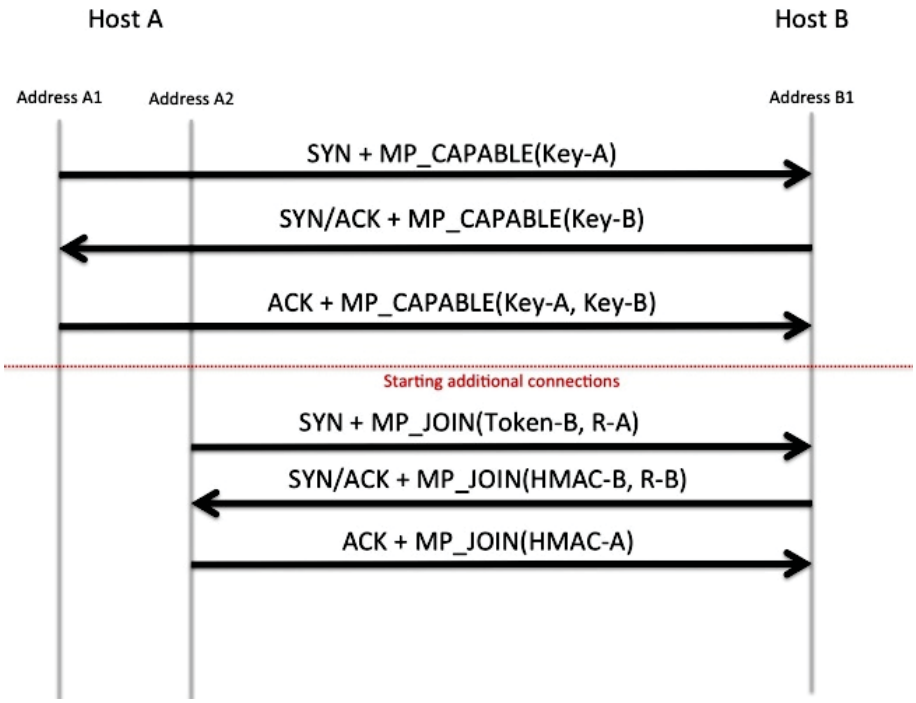


Figura 2.4: Estabelecimento da conexão e adição de novo subfluxo, extraído de [Cisco, 2013]

Quando da adição de um novo subfluxo, o Host-A pode iniciar subfluxos adicionais provenientes de uma interface ou endereço diferente para o Host-B. Como ocorreu com o subfluxo inicial, as *Options* TCP são usadas para indicar o desejo de mesclar esse subfluxo com o subfluxo inicial. As chaves que foram trocadas durante o estabelecimento do subfluxo TCP inicial (junto com um algoritmo de *hashing*) são usadas pelo Host-B para confirmar que a solicitação de adição

[3]https://www.iana.org/assignments/tcp-parameters/tcp-parameters.xhtml#mptcp-option-subtypes

é de fato enviada pelo Host-A. A 4-tupla (IP de origem, porta de origem, IP de destino, porta de destino) do subfluxo secundário é diferente da 4-tupla do subfluxo inicial, logo, este subfluxo pode ter um caminho diferente através da rede. Um processo similar é realizado para a remoção de subfluxos da conexão MPTCP. Tanto a adição quanto a remoção de subfluxos podem ser solicitadas pelo Host-A ou pelo Host-B. A descoberta e configuração dos subfluxos adicionais é realizada pelo componente gerenciamento de caminhos e para controlar o recebimento ordenado dos dados provenientes de diferentes subfluxos, o MPTCP utiliza, além do número de sequência em nível de subfluxo TCP, um número de sequência em nível de conexão.

A arquitetura do MPTCP que se encontra na RFC 6182 [Ford et al., 2011] apresenta os objetivos funcionais, que definem os serviços e recursos que uma implementação do MPTCP deve possuir, e os objetivos de compatibilidade, que determinam como o protocolo deve interagir com outras entidades.

Objetivos funcionais

Para permitir o uso de multi-caminhos, a implementação do MPTCP deve perseguir os seguintes objetivos funcionais.

- **Melhorar a taxa de transferência:** o MPTCP **deve** suportar o uso concorrente de multi-caminhos. Para cumprir o mínimo de desempenho que justifique uma implantação, uma conexão MPTCP não **deve** ter menos desempenho do que uma única conexão TCP padrão no melhor caminho disponível; e
- **Melhorar a resiliência:** o MPTCP **deve** suportar o uso de multi-caminhos intercambiáveis para fins de resiliência. Permitindo que segmentos possam ser enviados e reenviados por qualquer caminho disponível. Isso garante que, no pior dos casos, o protocolo não **deve** ser menos resiliente do que o TCP padrão por um único caminho.

Como a distribuição do tráfego entre os caminhos disponíveis e as respostas ao congestionamento são feitas de acordo com os princípios de agrupamento de recursos, um efeito secundário de atingir os objetivos funcionais é que o uso generalizado do MPTCP pela Internet tende a melhorar a utilização geral da rede, deslocando a carga dos caminhos mais congestionados e aproveitando a capacidade disponível de outros caminhos sempre que possível.

Além disso, o MPTCP **deve** possuir negociação automática para sua utilização. Um host que suporte MPTCP que deseje se conectar com outro host pelo MPTCP deve ser capaz de detectar de forma confiável se o outro host realmente suporta as extensões necessárias, usando-as, e, caso contrário, voltando automaticamente para o TCP por um caminho único.

Objetivos de Compatibilidade

Além dos objetivos funcionais já apresentados, uma implementação do MPTCP deve atender a uma série de objetivos de compatibilidade para ser suportado na Internet de hoje. Esses objetivos se enquadram nas seguintes categorias.

- **Compatibilidade de aplicação:** o MPTCP **deve** permitir a utilização da API de *socket* já existente para o TCP padrão quando da sua utilização;
- **Compatibilidade de rede:** o protocolo **deve** ser compatível com a rede onde está sendo implantado, ou seja, com a internet, podendo atravessar *middleboxes* de mercado, como firewalls, NATs e proxies; e

- **Compatibilidade com outros usuários da rede:** o protocolo **deve** manter a equidade entre os usuários da rede, coexistindo amigavelmente com outros fluxos TCP padrão em situações de gargalo.

Após apresentar os objetivos, a base da arquitetura do MPTCP é explicada como sendo uma divisão das funções realizadas pela camada de transporte, baseado nos estudos apresentados em [Ford e Iyengar, 2008]. São definidas duas subcamadas dentro da camada de transporte: orientada-à-aplicação (*Semantic*) e orientada-à-rede (*Flow+Endpoint*), como mostrado na Figura 2.5. A camada orientada-à-aplicação implementa funções focadas principalmente em suportar e proteger a comunicação fim-a-fim do aplicativo, enquanto a camada orientada-à-rede implementa funções como identificação de *hosts* (usando números de porta) e controle de congestionamento.

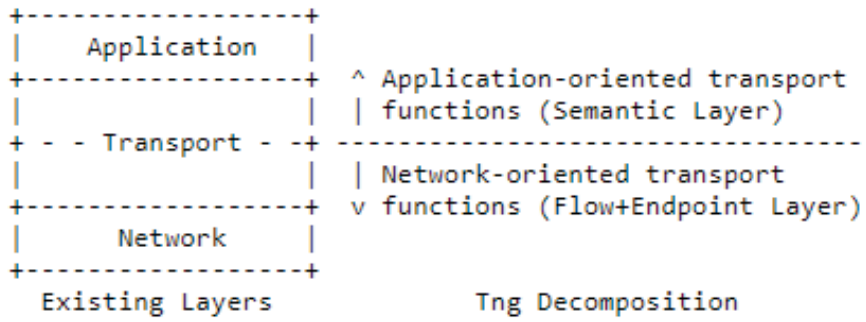


Figura 2.5: Decomposição das funções da camada de transporte, extraído de [Ford et al., 2011]

A Figura 2.6 mostra como as *middleboxes* interagem com as diferentes camadas neste modelo de camada de transporte decomposta: a camada orientada-à-aplicação opera fim-a-fim, enquanto a camada orientada-à-rede opera segmento-por-segmento e pode sofrer interferência das *middleboxes*.

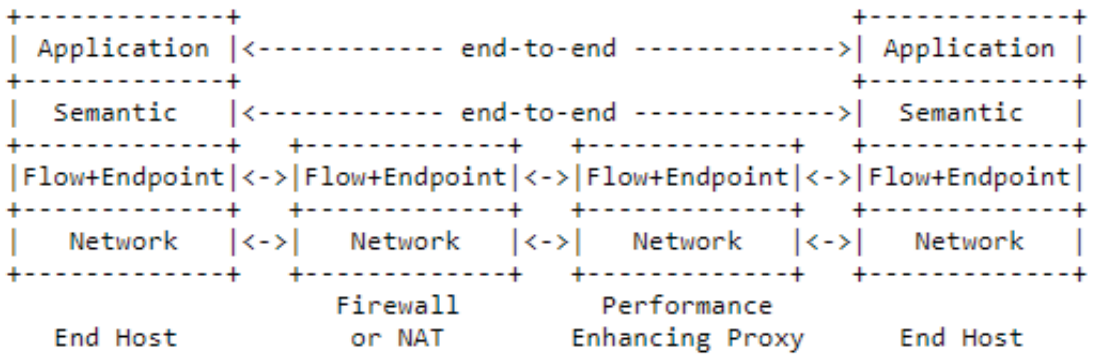


Figura 2.6: *Middleboxes* no novo modelo de internet, extraído de [Ford et al., 2011]

A arquitetura do MPTCP segue o modelo apresentado na Figura 2.7. O ***MPTCP***, seguindo esse modelo de decomposição das funções da camada de transporte em subcamadas, que provê a compatibilidade de aplicação, mantendo a semântica do TCP referente aos dados das aplicações e a confiabilidade, é uma instância da camada orientada-à-aplicação (*semantic*), e o componente ***subflow*** **(TCP)** que fornece compatibilidade de rede, aparentando-se e comportando-se como um fluxo TCP na rede, é uma instância da camada orientada-à-rede (*Flow + Endpoint*).

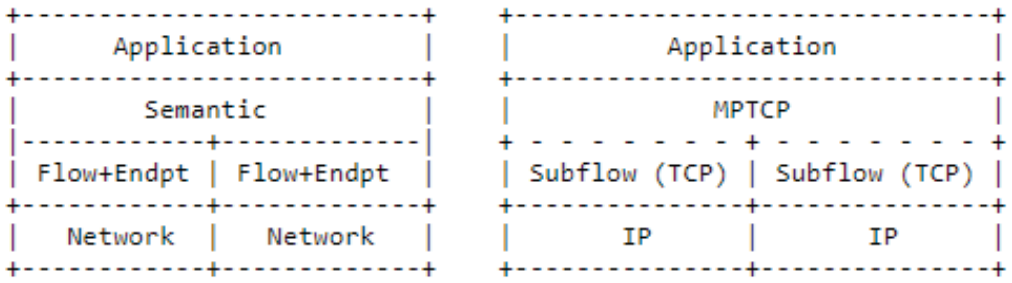


Figura 2.7: Relação entre o modelo (esquerda) e o MPTCP (direita), extraído de [Ford et al., 2011]

Como uma extensão de TCP, o MPTCP trata explicitamente *middleboxes* em seu projeto e, com isso, especifica um protocolo que opera em duas escalas: o componente MPTCP opera fim-a-fim, enquanto permite que o componente *subflow (TCP)* opere segmento-por-segmento.

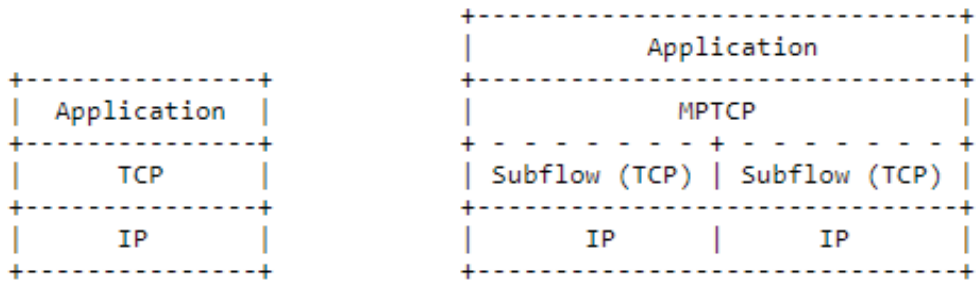


Figura 2.8: Comparação das pilhas do TCP padrão e do MPTCP, extraído de [Ford et al., 2011]

A arquitetura padronizada do MPTCP busca suportar de maneira eficiente os objetivos para o transporte multi-caminhos. O MPTCP faz uso de sessões TCP, o termo subfluxo (do inglês *subflow*), para prover o transporte pelos multi-caminhos, e com isso atingir a compatibilidade de rede desejada. Além disso, as informações específicas do MPTCP são portadas de maneira compatível nos subfluxos TCP. A Figura 2.8 apresenta a comparação das pilhas dos protocolos TCP padão e do MPTCP.

Abaixo da camada de aplicação, a extensão MPTCP, por sua vez, gerencia vários subfluxos TCP abaixo dele. Para fazer isso, o MPTCP deve implementar os seus principais componentes: o gerenciador de caminhos, o controle de congestionamento e o escalonador. Estes componentes, junto das interfaces de subfluxo TCP, são ilustrados na Figura 2.9.

A seguir apresentaremos as funções de cada um dos componentes básicos do MPTCP, com especial atenção ao componente escalonador.

Gerenciador de caminhos

Tem como função detectar e usar múltiplos caminhos entre dois *hosts*. O MPTCP usa a disponibilidade de vários endereços IP em um ou em ambos os *hosts* como um indicador para isso. Além disso, o gerenciador de caminhos do protocolo MPTCP é o mecanismo para sinalização de endereços alternativos para *hosts*, e o mecanismo para iniciar um novo subfluxo e adicioná-lo a uma conexão MPTCP existente. Os processos de estabelecimento do subfluxo TCP inicial, de adição e de remoção de subfluxos à conexão MPTCP já foram explicados anteriormente nesta seção e estão representados na Figura 2.4.

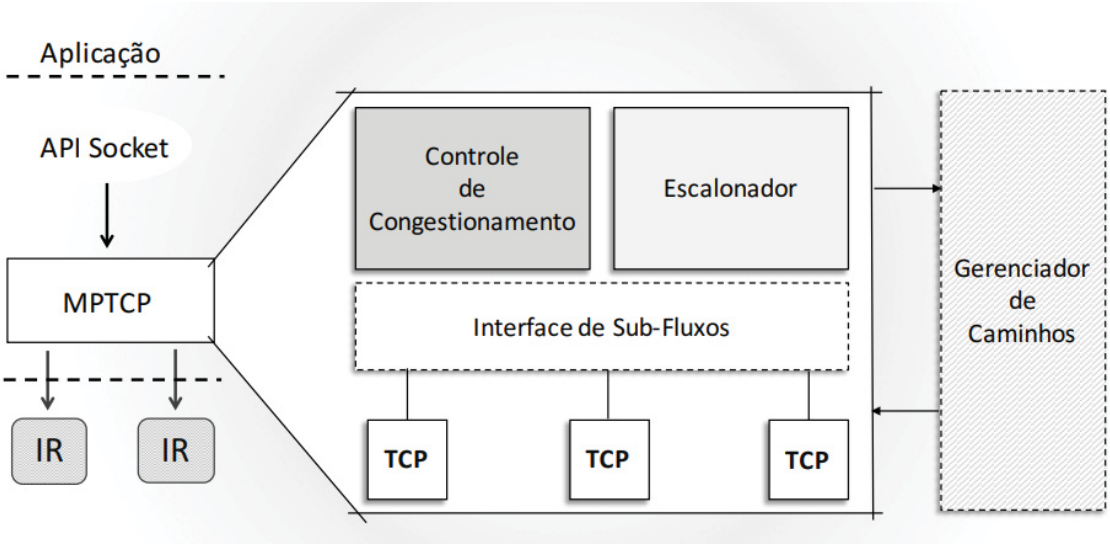


Figura 2.9: Componentes do protocolo MPTCP, adaptado de [Ford et al., 2011]

Controle de congestionamento

Este componente tem como função coordenar o controle de congestionamento nos subfluxos. Conforme especificado na arquitetura, o algoritmo de controle de congestionamento **deve** garantir que uma conexão MPTCP não ocupe injustamente mais largura de banda do que um único caminho com um fluxo de TCP ocuparia em um enlace congestionado compartilhado com outros fluxos, sejam eles TCP ou MPTCP.

O controle de congestionado do MPTCP tem como base do TCP [Singh et al., 2013], como a sua variante mais utilizada o NewReno. Porém, sua aplicação em conexões MPTCP com múltiplos subfluxos TCP não respeita o princípio de justiça na rede. Por esse motivo, algoritmos alternativos de controle de congestionamento surgiram para o protocolo MPTCP, tais como, o *Fully Coupled congestion control*, o *Linked Increases Algorithm (LIA)* ou também conhecido como *semi-coupled congestion control*, o *Dynamic Window Coupling (DWC)* e o *Opportunistic Linked Increases Algorithm (OLIA)*. Todos esses algoritmos não modificam as fases de inicialização lenta (Slow Start), retransmissão rápida (Fast Retransmit) e recuperação rápida (Fast Recovery), mas somente a fase de prevenção de congestionamento (*Congestion Avoidance*).

O controle de congestionamento utiliza janelas de congestionamento, $CWND$, individuais para cada subfluxo no *host* de origem e uma única janela de recebimento, $RWND$, compartilhada por todos os subfluxos. Os principais requisitos do controle de congestionamento são aumentar a vazão, ser justo com fluxos concorrentes e balancear o congestionamento. Logo, para atender esses requisitos a vazão dos subfluxos em uma transferência multi-caminhos **deve** ser pelo menos melhor que a de um único fluxo TCP através do melhor caminho disponível. Toda a conexão MPTCP **deve** se comportar como um único fluxo TCP na existência de um gargalo compartilhado. E por fim, o tráfego **deve** ser direcionado para os caminhos menos congestionados.

O algoritmo LIA (*Linked Increase Algorithm*) [Raiciu et al., 2011], que se baseia na abordagem de fases de aumento aditivo e diminuição multiplicativa (*AIMD*), foi adotado como padrão para o controle de congestionamento do MPTCP pelo IETF [Ford et al., 2013]. Este

algoritmo especifica apenas como deve ser o aumento da janela de congestionamento ao receber um ACK, mantendo a diminuição do mesmo jeito que ocorre no TCP. O crescimento das janelas de congestionamento dos subfluxos é acoplado de acordo com,

$$w_i = \begin{cases} \min\left(\frac{\alpha}{W}, \frac{1}{w_i}\right), & \text{para o aumento aditivo } (AI), \\ \frac{w_i}{2}, & \text{para a diminuição multiplicativa } (MD), \end{cases} \tag{2.4}$$

$$\text{onde } \alpha = W * \frac{\max\left(\frac{w_i}{RTT_i^2}\right)}{\left(\frac{\sum w_i}{RTT_i}\right)^2}. \tag{2.5}$$

O parâmetro α, calculado de acordo com (2.5), regula a agressividade dos subfluxos de maneira que o aumento da janela não seja superior ao de um único fluxo TCP com o mesmo tamanho de janela. A variável w_i corresponde ao tamanho da janela de transferência, a variável RTT_i ao tempo de ida e de volta pelo subfluxo i e a variável W ao tamanho total das janelas de congestionamento de todos os subfluxos disponíveis.

Escalonador

O escalonador é o componente do MPTCP responsável por distribuir os dados da aplicação divididos em segmentos através dos subfluxos, com o objetivo de que sejam transmitidos, recebidos e entregues de forma confiável e ordenada à aplicação de destino. Nos protocolos de transferência multi-caminhos, o escalonador é o principal responsável pelo desempenho na agregação dos caminhos, pois pode levar em conta variações das características dos caminhos, restrições da aplicação e requisitos dos usuários quando da sua decisão de escalonamento [Paasch et al., 2014].

Uma conexão MPTCP envolve o uso de muitos subfluxos TCP, um escalonador deve ser adicionado para decidir onde enviar cada um dos segmentos de dados. Duas possíveis posições para o escalonador foram avaliadas na distribuição Linux[4] *MultiPath TCP* [Barré et al., 2011]. Uma opção é escalonar os segmentos assim que chegarem do *buffer* de aplicação. Essa opção, que consiste em enviar dados para os subfluxos assim que estiverem disponíveis, foi implementada em versões mais antigas do Linux *MultiPath TCP* e agora está descontinuada. Uma outra opção é armazenar todos os dados de forma centralizada na camada de transporte do MPTCP, *semantic layer*, dentro de um buffer de envio compartilhado, vide Figura 2.10.

Nessa opção, que é a adotada atualmente, o escalonamento é feito no momento da transmissão, sempre que qualquer subfluxo esteja disponível para enviar mais dados, geralmente devido ao recebimento de um ACK que tenha aberto espaço na sua janela de congestionamento. Nesse cenário, os subfluxos solicitam segmentos do buffer de envio compartilhado sempre que estiverem com espaço disponível. Note que vários subfluxos podem ficar disponíveis para o envio simultaneamente, se um ACK alertar uma nova janela de recebimento, abrindo mais espaço na janela de envio compartilhada. Por esse motivo, quando um subfluxo solicita dados para envio, o escalonador é executado e outros subfluxos podem receber dados do escalonador no mesmo ciclo de escalonamento.

Conforme mostrado, na Figura 2.10, um segmento primeiro entra no *buffer* de envio compartilhado, então, ao chegar ao fim desse *buffer*, é solicitado por algum subfluxo. Enquanto não receber o ACK reconhecendo o segmento de dado, o MPTCP mantém uma cópia do dado

[4]https://multipath-tcp.org/

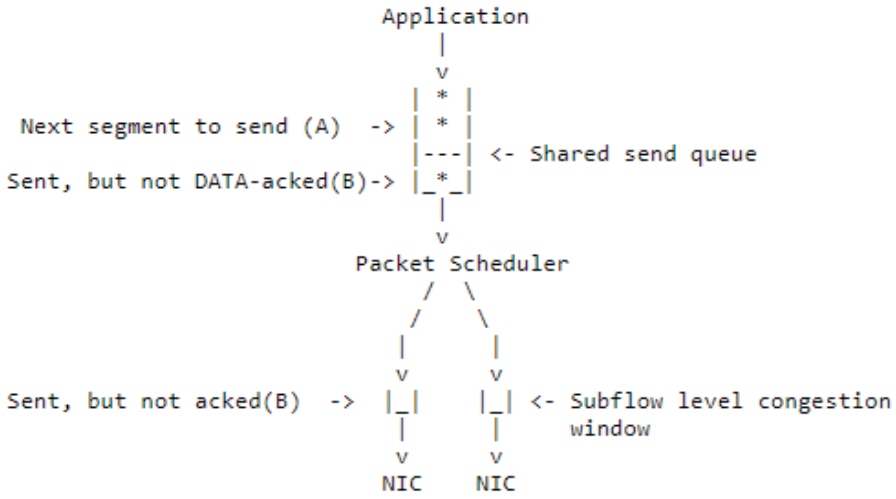


Figura 2.10: Posição do componente escalonador, extraído de [Barré et al., 2011]

no buffer de envio compartilhado, (*sent, but not DATA-acked (B)*) da figura. Mas para suportar falhas, a implementação do MPTCP precisa poder mover segmentos de um subfluxo para outro, de modo que a falha seja invisível a partir da aplicação. Isso é conseguido pelo uso do número de sequência em dois níveis: nível de subfluxo e nível de conexão. Basta ao escalonador retransmitir o dado em um novo segmento com o mesmo número de sequência a nível de conexão por outro subfluxo com janela disponível.

Essa abordagem apresenta muitas vantagens:

- Cada subfluxo pode facilmente preencher a sua janela de congestionamento, desde que haja dados para serem enviados no *buffer* de envio compartilhado e o escalonador não tenha uma política que restrinja o subfluxo;
- Se um subfluxo falhar, ele não receberá ACK dos segmentos já enviados, portanto, naturalmente, irá parar de solicitar segmentos do *buffer* de envio compartilhado. Isso remove a necessidade de um **estado de falha** explícito, para garantir que um subfluxo em falha não receba dados, ao contrário do que acontece, por exemplo, no SCTP-CMT, que precisa de uma marcação explícita de subfluxos em falha por definição de arquitetura, porque utiliza um único nível de número de sequência; e
- Da mesma forma, quando um subfluxo em falha se torna ativo novamente, os segmentos pendentes de sua janela de congestionamento são finalmente reconhecidos, permitindo-o solicitar segmentos novamente ao *buffer* de envio compartilhado. Observe que, nesse caso, os dados reconhecidos normalmente são descartados pelo receptor, porque os segmentos correspondentes foram retransmitidos em outro subfluxo durante o tempo de falha.

Resumindo, podemos dizer que o escalonador deve tomar três decisões básicas durante o processo de escalonamento:

- Selecionar os subfluxos: selecionar os subfluxos disponíveis, ou seja, aqueles que possuem espaço em sua janela de congestionamento, CWND;

- Distribuir entre os subfluxos: quando existem mais do que um subfluxo disponível o escalonador deve decidir, com base no algoritmo, a ordem de distribuição dos segmentos nos subfluxos; e
- Granularidade: definir a quantidade de dados (*bytes*) que será distribuída em segmentos a serem destinados a cada subfluxo.

O IETF, diferente do que fez com o controle de congestionamento, não padronizou um escalonador para o MPTCP. No entanto, a implementação disponibilizada no Linux *MultiPath TCP* tem uma estrutura modular e permite optar por diferentes escalonadores. Nele existem três algoritmos de escalonamento implementados: o algoritmo ***Round-Robin (RR)***, que distribui os dados de modo circular entre os subfluxos disponíveis; o algoritmo ***LowRTT (Lowest-RTT-First)***, que distribui os dados priorizando os subfluxos com o menor RTT; e, por fim, o algoritmo **Redundante**, que replica os dados sobre os subfluxos disponíveis.

A estratégia de escalonamento utilizada pelo *Round Robin* distribui os segmentos para um subfluxo após o outro de maneira circular, como ilustrado pela Figura 2.11. A quantidade de segmentos enviados de forma consecutiva (granularidade) em cada ciclo de escalonamento pode ser configurada. Um outro modo de operar é optar por preencher a janela de congestionamento, CWND, de cada subfluxo. Neste primeiro caso, a distribuição de pacotes entre os subfluxos é igual. Ao optar pelo preenchimento da CWND, que é como está implementado no Linux MPTCP e nas outras implementações atuais, o algoritmo de escalonamento não é de fato um RR, porque o subfluxo só receberá mais segmentos quando liberar espaço em sua CWND. O tempo entre ciclos de escalonamento é conhecido como *Acknowledgement-Clock*, tempo de espera pelo ACK.

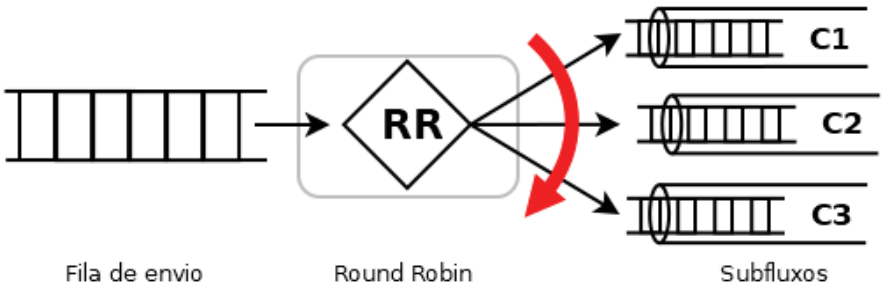


Figura 2.11: Escalonador *Round Robin*

A estratégia de escalonamento do *LowRTT* inicialmente ordena os subfluxos pelo menor RTT e distribui os segmentos preenchendo a CWND antes de passar para o próximo como representado na Figura 2.12. Como no Round Robin, assim que a CWND é preenchida, o escalonamento se torna *ACK-Clock*. Essa estratégia de escalonamento leva em consideração uma das características do caminho, o atraso, e se ajusta dinamicamente às variações dos caminhos. Como consequência, o *LowRTT* tem melhor desempenho em relação ao RR em termos de taxa de transferência e atraso fim-a-fim, embora em cenário homogêneos os resultados sejam semelhantes.

A estratégia de escalonamento redundante transmite o fluxo de dados nos subfluxos disponíveis de maneira redundante ou replicada. Essa estratégia é útil quando o principal requisito da transmissão é conseguir uma menor latência ao custo de desperdiçar a largura de banda disponível.

As estratégias de escalonamento RR e o LowRTT realizam a transferência concorrente, CMT, enquanto a estratégia de escalonamento redundante realiza uma transferência resiliente,

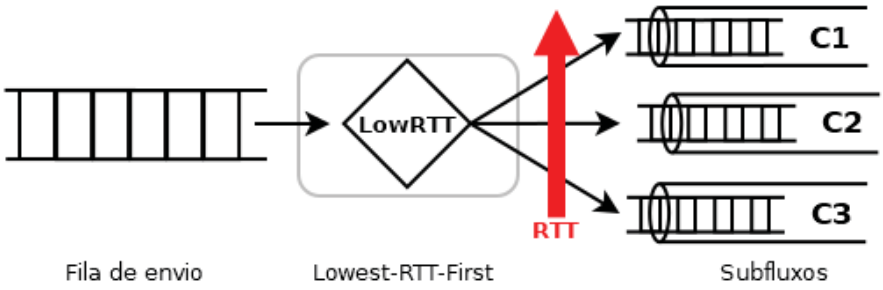


Figura 2.12: Escalonador *Lowest-RTT-First*

RMT. O problema com as estratégias RR e LowRTT é que elas não lidam bem com o problema do bloqueio do *buffer* de recebimento, *HOL - Head-of-line blocking*, que será melhor explicado na próxima seção. Isto causa intermitência no recebimento dos dados e atrasos na entrega ordenada dos dados à aplicação. Para compensar a heterogeneidade dos caminhos e evitar os problemas com o bloqueio HOL, o MPTCP usa o mecanismo de Retransmissão e Penalização [Paasch et al., 2014]. Este mecanismo reencaminha o segmento causador do bloqueio HOL em um subfluxo que tenha espaço na CWND. O objetivo é tornar mais rápida a recuperação das situações de bloqueio, compensando a diferença entre os RTT dos caminhos.

2.3 Principais problemas da transferência multi-caminhos

Os problemas na transferência multi-caminhos, em sua grande maioria, ocorrem em cenários de redes heterogêneas, causando perda de desempenho na taxa de transferência, aumento ou variação do atraso fim-a-fim e perda de pacotes. Dentre os problemas, os principais são o bloqueio do *buffer* de recebimento - HOL, causado pelo reordenamento de pacotes no *buffer* de recebimento, e o congestionamento *Bufferbloat*.

O bloqueio HOL ocorre devido ao esgotamento do *buffer* de recebimento em decorrência do recebimento de segmentos fora de ordem causado pela heterogeneidade dos caminhos, mais especificamente, por diferenças no RTT, como podemos ver na Figura 2.13. Enquanto os segmentos faltantes não forem recebidos, não é possível fazer a entrega ordenada dos dados, e os segmentos já recebidos ficam armazenados no *buffer*. Como o *buffer* possui um tamanho limitado, ela pode ser preenchida e não permitir que o emissor encaminhe mais segmentos até receber o segmento esperado e libere os dados para a aplicação.

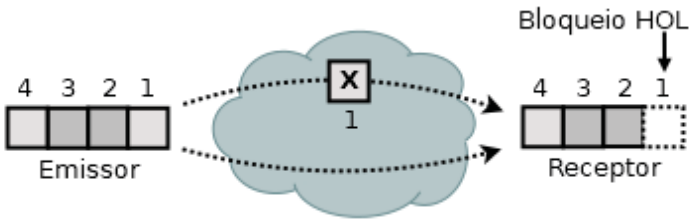


Figura 2.13: Bloqueio HOL

Esse é um dos motivos da RFC 6182 [Ford et al., 2011] **recomendar** que o tamanho do *buffer* de recebimento seja de no mínimo $2 \times \sum BW_i \times RTT_{max}$. Sendo BW_i a capacidade do caminho i e RTT_{max} o maior atraso entre os caminhos.

O fenômeno *Bufferbloat* [Felix et al., 2016] se refere à formação de longas filas de pacotes, congestionamentos, em equipamentos de rede com grande capacidade de memória, que apenas descartam os pacotes quando a *buffer* enche. Isto acarreta em longos atrasos para os protocolos de transporte que utilizam um controle de congestionamento baseado em perdas, degradando o desempenho da transmissão. Uma vez que o controle de congestionamento do MPTCP utiliza a perda de pacotes como sinal para identificar o congestionamento, o *bufferbloat* não percebido acaba interferindo no desempenho dos demais caminhos.

2.4 Resumo

Este capítulo apresentou conceitos importantes sobre a transferência multi-caminhos para sistemas multi-abrigados, introduziu termos e definições, apresentou o SCTP e o MPTCP, os principais protocolos para transferência multi-caminhos em sistemas multi-abrigados, bem como a arquitetura básica do MPTCP e seus principais componentes com foco no escalonador, alvo deste trabalho, e também os principais algoritmos escalonadores, o *Round Robin*, o *LowRTT* e o redundante, encontrados em implementações atuais do MPTCP. Por fim, fez uma breve apresentação dos principais problemas na transferência multi-caminhos em redes heterogêneas.

Capítulo 3

Trabalhos Relacionados

Este capítulo contém uma breve revisão de trabalhos relacionados mais relevantes envolvendo técnicas de escalonamento baseadas em características dos caminhos disponíveis em protocolos de sistemas multi-abrigados. Todos os trabalhos revisados levam em consideração características dos caminhos para o escalonamento e são divididos dentro das seções de acordo com a estratégia de escalonamento utilizada no algoritmo.

3.1 Estratégias de escalonamento focadas nas características dos caminhos

Como o MPTCP e suas aplicações são um tema recente em redes que tem despertado grande interesse na área de pesquisa acadêmica e no mercado, muitos artigos surgem com propostas de mecanismos baseados em estratégias de escalonamento para resolver problemas em protocolos de múltiplos caminhos. Esta seção apresenta um levantamento de trabalhos que utilizam características dos caminhos para tratar do problema de reordenamento dos pacotes no *buffer* de recebimento, da latência e que buscam a otimização na utilização dos múltiplos caminhos disponíveis.

No artigo [Lim et al., 2017] os autores propõem e implementam o algoritmo escalonador ECF (*Earliest Completion First*) que utiliza o RTT, o $CWND$ e a quantidade de dados a ser enviada, *send buffer*, para definir o melhor caminho para envio dos segmentos. Basicamente, o algoritmo do escalonador ECF avalia, caso o caminho de menor RTT esteja indisponível para envio (com a $CWNDi$ cheia), se o tempo para envio dos dados do *send buffer*, k dados, pelo caminho com maior RTT ($RTTmax$) é maior do que o tempo até o caminho de menor RTT ($RTTmin$) estar disponível para o envio, somado o tempo para o envio dos dados do *send buffer*, ou seja, $RTTmax > RTTmin = (k \div CWNDi) \times RTTmin$. Nesse caso é mais vantajoso esperar o caminho de menor RTT estar disponível para o envio do segmento de dados.

Em [Zeng et al., 2017] os autores apresentam o algoritmo escalonador MPTCP-MAPS (*Multi-attribute Aware Path Selection approach for MPTCP*) que baseado nos atributos RTT e taxa de perda de pacotes da camada de transporte busca atingir a eficiência na entrega dos pacotes. O MPTCP-MAPS projeta um mecanismo para comutação dinâmica entre os caminhos baseado nos atributos, assim aumentando a eficiência na utilização dos múltiplos caminhos. Esse resultado decorre da observação das características dos caminhos que podem evidenciar que um caminho está mais congestionado que outro.

Em [Ke et al., 2016] os autores apresentam o escalonador MA² (*Multiple Attribute-aware*) que para diminuir o problema do reordenamento, monitora informações dos caminhos,

como RTT e $CWND$ e em seguida ordena de forma ascendente pelo RTT. Se der empate, utiliza a $CWND$ como critério de desempate. De maneira adaptativa transmite os pacotes conforme a ordem dos caminhos. Então escalona os dados para o primeiro caminho da lista ordenada ou para o próximo, caso não tenha espaço na $CWND$.

O artigo [Li et al., 2015] propõe um algoritmo, AMTCP (*Adaptive Multi-path Transmission Control Protocol*), que busca otimizar os protocolos de múltiplos caminhos, para isso, adapta o número de subfluxos conforme a ocupação dos mesmos. Se a ocupação for grande, exceder um limiar, abre-se um novo subfluxo. Caso seja pequena, remove o subfluxo.

Em [Oh e Lee, 2015], os autores propõem o algoritmo CPS (*Constraint-based Proactive Scheduling*) que busca escalonar os pacotes considerando a observação do tamanho residual da fila de recebimento e do atraso em cada caminho com o objetivo de minimizar a necessidade de reordenamento. São propostos dois modos de operação no algoritmo: o radical e o conservador. O radical inicia escalonando para o subfluxo com menor RTT, como é feito pelo mecanismo *Lowest RTT First* (LowRTT), que ao lado do *Round Robin*, é padrão nas implementações do MPTCP, e só envia para o próximo subfluxo se $RWNDr$, proposto como estimativa da condição dos caminhos, for maior que o requerido. No conservativo, se o caminho mais rápido atende as condições, partida lenta ou $2 \times PcTp > RWNDr$, onde $PcTp$ é o número de pacotes pendentes de reconhecimento, só se utiliza o caminho mais rápido. E se $2 \times PcTp \leq RWNDr$, os pacotes são encaminhados para o próximo subfluxo apenas quando $RWNDr > RWNDr + 2 \times PcTp$.

Em [Garcia-Saavedra et al., 2017] os autores propõem o escalonador S-EDPF (*Stochastic Earliest Delivery Path First*) que é uma generalização do *Earliest Delivery Path First* (EDPF), apresentado no artigo [Chebrolu e Rao, 2006]. O modelo estima os valores do atraso dos caminhos com base nos valores prévios. Então gera-se uma matriz que é utilizada pelo escalonador EDPF. Esse escalonador encaminha os pacotes pelo caminho com o menor atraso de entrega estimado. O S-EDPF é uma variante estocástica do EDPF que tenta minimizar o atraso de reordenamento através do escalonamento apropriado dos pacotes pelos múltiplos caminhos.

Em [Chen et al., 2014] os autores propõem o *Round-Trip-Time based Concurrent Transmission Scheduling* (RTT-CTS), um algoritmo escalonador adaptativo para diminuir o problema do reordenamento no *buffer* de recebimento. Para isso, o algoritmo segue a seguinte lógica: se os RTT de dois caminhos são iguais, usa-se um simples *Round Robin* (RR), caso contrário, seleciona-se o caminho com menor RTT. Ao selecionar o caminho, incrementa o tempo estimado de transmissão. Esse processo se repete até o atraso dos caminhos ficarem compatíveis, voltando a utilizar um simples RR.

Na busca de solução para o mesmo problema, o artigo [Le e Bui, 2015] propõe o *Forward Delay-based Packet Scheduling* (FDPS). O algoritmo desse escalonador estima o atraso de cada caminho baseado no cálculo da diferença entre eles. Depois seleciona os caminhos com menor diferença de atraso para alocar os pacotes. O diferencial desse escalonador é a forma de estimar o caminho com menor atraso, pois mesmo através da diferença entre os atrasos dos caminhos não há a necessidade de sincronismo entre o lado que envia e o que recebe.

3.2 Estratégias de escalonamento com filtragem ou remoção dos caminhos baseado nas características

Um dos objetivos funcionais da arquitetura do MPTCP é a melhora do desempenho da conexão através da taxa de transferência agregada dos caminhos. Porém, em cenários de redes heterogêneas, muitas vezes esse objetivo não é atingido [Nikravesh et al., 2016]. O principal motivo disso é o problema do reordenamento no *buffer* de recebimento causado pela diferença de

atraso entre os caminhos e pela perda de pacotes. Isso se agrava em fluxos de curta duração, ou seja, de poucos KB, pois nesse caso a largura de banda é menos importante do que o atraso para o desempenho final da taxa de transferência. Os trabalhos apresentados nesta subseção propõem escalonadores com algoritmos de filtragem dos caminhos disponíveis baseado em suas características visando retirar os caminhos que causam degradação no desempenho da conexão do universo de caminhos disponíveis para escalonamento.

O artigo [Hwang e Yoo, 2015] propõe o algoritmo *Freeze Packet Scheduling* (FzPS) que, assim como o anterior, se baseia na diferença de atraso dos caminhos. Porém, se a diferença for significativa, só escalona pacotes para o caminho mais rápido, congelando o caminho mais lento. Essa opção decorre do fato do MPTCP ser geralmente benéfico para fluxos de longa duração, mas apresentar um desempenho pior que o TCP padrão explorando o melhor caminho para fluxos de tamanho pequeno, apenas centenas de KB. Neste caso, o artigo observa que seria melhor usar apenas o caminho mais rápido, uma vez que o atraso é muito mais importante do que a largura de banda da rede em tão pequeno fornecimento de dados.

Em [Hu et al., 2016], os autores propõem o algoritmo escalonador *Multi-level Correlation Scheduling* (MLCS) que apresenta um modelo para calcular a taxa de transferência e correlação dos caminhos. O MLCS classifica os caminhos podendo remover um subfluxo quando o desempenho do subfluxo está abaixo dos critérios, diminuindo os efeitos de subfluxos degradados e melhorando a taxa de transferência agregada. Além disso, o MLCS escalona os segmentos com base nas correlações entre os caminhos e suas capacidades de transmissão, visando a melhor utilização dos recursos da rede.

3.3 Estratégias de escalonamento baseadas na ordem de envio dos pacotes

Também encontramos artigos que atuam no problema com foco no escalonamento discriminado dos pacotes nos diferentes caminhos a fim de que cheguem em ordem, buscando assim diminuir o problema do reordenamento. Esses se dividem em duas estratégias: os que enviam os pacotes em ordem e os que enviam os pacotes fora de ordem. Começaremos falando dos artigos que propõem algoritmos para o envio dos pacotes em ordem.

3.3.1 Envio dos pacotes em ordem

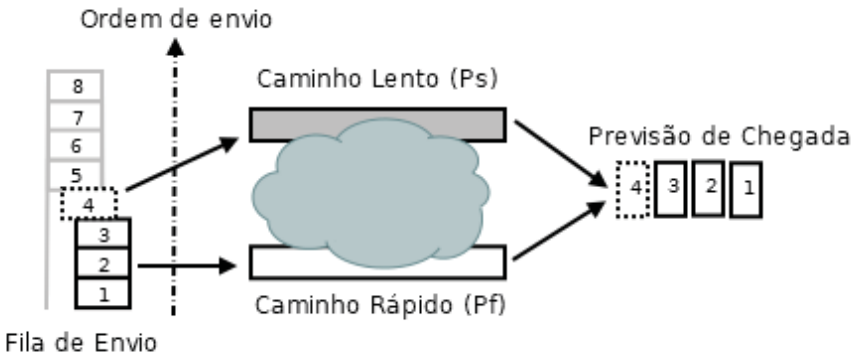


Figura 3.1: Processo de envio dos pacotes em ordem

Basicamente, os métodos de escalonamento que enviam os pacotes em ordem seguem o processo representado no exemplo da Figura 3.1. Considere dois caminhos com mesma capacidade, porém, o caminho lento (*Ps*) possui um atraso três vezes maior que pelo caminho rápido (*Pf*). Os pacotes são então escalonados em ordem para os caminhos: o 1º ao 3º pacote são designados para o caminho mais rápido *Pf* e o 4º pacote para o mais lento *Ps*. Espera-se que, com base na estimativa do atraso, os pacotes cheguem em ordem no destino.

O artigo [Mirani et al., 2010] propôs um escalonador para o SCTP-CMT que foi precursor em estimar o tempo de chegada dos pacotes no recebedor e transmiti-los de modo discriminado, *Forward Prediction Scheduling* (FPS). O principal problema desse tipo de método é a dificuldade em estimar com precisão o tempo de chegada dos pacotes no destino. Pois o caminho de ida e de volta pode ter características, como o atraso, diferentes e também sofrer variações no tempo. Se as características dos caminhos se mantiverem constantes, os métodos que utilizam esse tipo de estratégia podem obter bons resultados, caso contrário, podem até agravar o problema do reordenamento devido ao erro da estimativa como foi demonstrado no artigo [Ni et al., 2015].

O artigo [Kuhn et al., 2014] propõe o *Delay Aware Packet Scheduling* - DAPS que implementa estratégia similar, mas considera no tempo de entrega o atraso decorrente das filas dos caminhos. A fila de envio dos caminhos não é gerida pelo escalonador do nível da conexão, não cabendo-lhe a decisão sobre o momento do envio pelo caminho. Além disso, após escalonado para o caminho, essa decisão não pode ser alterada. Isso implica que o escalonador não tem total decisão sobre a estratégia de envio discriminado dos pacotes.

3.3.2 Envio dos pacotes fora de ordem

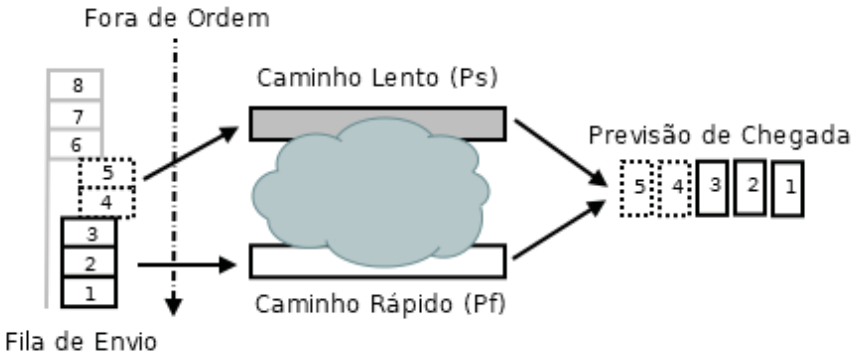


Figura 3.2: Processo de envio dos pacotes fora de ordem

Os métodos de escalonamento com envio dos pacotes fora de ordem seguem uma lógica similar ao exemplo dado para o envio em ordem. Porém o escalonador não respeita a ordem dos pacotes no *buffer* de envio para a tomada das decisões de escalonamento, mas sim a estimativa dos atrasos dos caminhos disponíveis e a previsão da ordem de chegada dos pacotes no destino. Assim o escalonador envia pacotes fora de ordem esperando que estes cheguem em ordem no destino. Esse processo pode ser observado na representação da Figura 3.2

Os artigos [Yang et al., 2014] - OTIAS e [Ni et al., 2015] - OCPS propõem mecanismo de escalonamento que transmitem os pacotes fora de ordem. Ambos mecanismos selecionam o pacote ainda não escalonado que pode ser enviado imediatamente por um determinado caminho sem obedecer a sequência da fila de envio. O mecanismo OTIAS decide o caminho para envio

com base em cada pacote. Ele considera também o *RTT* e o tamanho da fila dos subfluxos em um dado momento, o que se aproxima do que é feito pelo *Lowest RTT First* (LowRTT) do MPTCP padrão, embora considere mais informações e envie sem considerar a ordem dos pacotes.

Nos artigos [Ferlin et al., 2016] e [Liu et al., 2014] os autores propõem algoritmos que utilizam escalonamento por bloco de pacotes, BLEST (*Blocking estimation-based*) e NTS (*Novel Trunk Scheduling*) respectivamente, onde nesse caso a sequência de pacotes para o envio é respeitada somente dentro dos blocos. Um bloco é um determinado número de pacotes em sequência transmitidos em um caminho. O escalonador determina previamente o tamanho dos blocos com base nas informações dos caminhos. As decisões de escalonamento são tomadas por bloco e não por pacotes. Enquanto um caminho envia um determinado bloco, outro bloco pode ser encaminhado em um caminho mais rápido, não aguardando um novo ciclo.

No artigo [Sarwar et al., 2013] os autores propõem o algoritmo DAS (*Delay Aware Scheduling*) que escolhe o caminho e o momento para envio dos pacotes baseado no atraso dos caminhos. O envio não obedece a ordem dos pacotes e espera-se que cheguem em ordem no destino, com o objetivo de reduzir o espaço ocupado no *buffer* do destino. Para atingir esse objetivo, o algoritmo calcula o lcm, sigla de *Least Common Multiple* (mínimo múltiplo comum), dos atrasos dos caminhos, e através dele calcula o número ideal de pacotes a serem enviados em cada um dos caminhos. Depois é definido o vetor O que contém a ordem ideal de recebimento dos pacotes no destino, assim determinando a ordem de utilização dos caminhos para envio dos pacotes de tal forma que estes cheguem em ordem no *buffer* de destino. O principal problema dessa estratégia é que qualquer variação no atraso dos caminhos pode levar ao inverso do objetivo, ou seja, aumentar a ocupação do buffer de recebimento devido a um grande número de pacotes fora de ordem. Essa variação é comum em redes heterogêneas de multi-caminhos sem fio.

3.4 Considerações finais

Observa-se, mesmo focando a revisão de trabalhos relacionados em pesquisas que tratam de algoritmos escalonadores baseados nas características dos caminhos, muitos dos trabalhos apresentados neste capítulo foram publicados nos dois últimos anos, demonstrando o interesse recente no tema. Como já comentado anteriormente no Capítulo 1, isso decorre do fato do escalonador ser um componente presente nos protocolos de sistemas multi-abrigados que interfere diretamente em muitos de seus problemas e desafios, como por exemplo o bloqueio HOL (*Head-of-Line*), o reordenamento dos pacotes no *buffer* de recebimento, a latência e a taxa de transferência agregada.

A principal diferença deste trabalho para os apresentados nas seções anteriores é a análise do limiar da razão de perda de pacotes a partir da qual a taxa de transferência de um fluxo TCP padrão pelo melhor caminho disponível é superior ao da conexão MPTCP e, como será apresentado no capítulo seguinte, o método de filtragem do caminho mais degrado que pode ser utilizado em conjunto com diferentes algoritmos de escalonamento.

Em sequência, a Tabela 3.1 contém um resumo dos trabalhos relacionados apresentados nesse capítulo organizados pela estratégia utilizada no algoritmo de escalonamento.

Tabela 3.1: Resumo organizado das estratégias de escalonamento

Escalonador	**Estratégia**	**Descrição**
ECF, MPTCP-MAPS, MA², AMTCP, CPS, S-EDPF, RTT-CTS e FDPS	Características dos caminhos	Algoritmos que fazem o escalonamento baseado somente nas características e na correlação entre os caminhos
FzPS e MLCS	Filtragem ou remoção dos caminhos	Algoritmos que filtram os caminhos disponíveis baseado em suas características visando retirar os que degradam o desempenho da conexão
FPS e DAPS	Ordem de envio: em ordem	Algoritmos que tentam estimar a chegada dos pacotes no recebedor e alocar os dados seguindo uma ordem descriminada respeitando a ordem dos números de sequência dos pacotes
OTIAS, OCPS, BLEST, NTS e DAS	Ordem de envio: fora de ordem	Algoritmos que tentam estimar a chegada dos pacotes no recebedor e alocar os dados nos pacotes para envio independente da ordem do número de sequência dos pacotes

Capítulo 4

Proposta

Os protocolos de transferência multi-caminhos para sistemas multi-abrigados têm como objetivos principais o aumento da taxa de transferência agregada e da resiliência na transmissão. Dos protocolos de transferência multi-caminhos padronizados[1] pela IETF, o MPTCP foi o que conseguiu maior abertura e aceitação pelo fato da sua arquitetura ser focada na compatibilidade com a rede já existente e na funcionalidade [Ford et al., 2011]. Mesmo assim, existem cenários de rede onde os requisitos de funcionalidade do MPTCP não são totalmente atendidos. Isso se deve em grande parte pela ação do algoritmo do componente escalonador na arquitetura do protocolo, responsável por decidir por qual subfluxo um dado segmento será enviado.

No caso em que os caminhos possuem características similares, redes homogêneas, como em redes de centro de dados, os escalonadores providos pelo MPTCP, Round Robin e o LowRTT, podem melhorar o desempenho e alcançar alta disponibilidade [Paasch e Bonaventure, 2014]. Porém, se os caminhos são heterogêneos, redes heterogêneas, os escalonadores podem degradar o desempenho do MPTCP. Dependendo do nível de diversidade dos caminhos em termos de atraso e largura de banda, é melhor utilizar o TCP padrão [Hu et al., 2016]. Neste sentido, a comunidade acadêmica tem despendido grande esforço para identificar o que tem degradado o desempenho destes algoritmos escalonadores [Li et al., 2016] e propor algoritmos que possam mitigar o problema.

Os escalonadores implementados inicialmente no Linux MPTCP[2] e os estudados e propostos pela comunidade acadêmica podem ser divididos como algoritmos escalonadores simples e complexos. Os algoritmos escalonadores simples tomam decisões de forma inerente, ou seja, independente de variações no estado dos caminhos e normalmente tem um desempenho ruim em cenários heterogêneos, como exemplo temos o Round Robin. Porém, em cenários homogêneos, conseguem atingir desempenhos superiores aos de um fluxo TCP padrão. Isso acontece porque, em redes homogêneas, os atrasos são muito próximos e não ocorrem os problemas causados pelo grande número de reordenamento de pacotes no *buffer* de recebimento. Logo, em cenários de rede homogênea, não há a necessidade de utilização de escalonadores complexos. Porém, em redes heterogêneas, os escalonadores simples degradam significativamente a taxa de transferência devido ao impacto do reordenamento de pacotes, ocasionando o Bloqueio HOL, como mencionado na seção 2.3.

Os algoritmos escalonadores complexos se propõem a utilizar dinamicamente as informações dos caminhos, como por exemplo o atraso, a largura de banda, o tamanho da janela de congestionamento e o seu espaço disponível e a taxa de erro de pacotes, como

[1] Os protocolos de transferência multi-caminhos padronizados pela IETF são o SCTP-CMT e o MPTCP

[2] https://multipath-tcp.org/pmwiki.php/Users/ConfigureMPTCP - Escalonadores Lowest RTT First - LowRTT, Round Robin e o Redundante

indicadores para a decisão do melhor caminho disponível para o envio dos segmentos. O *Lowest-RTT-First* - LowRTT e os protocolos do resumo apresentado na Tabela 3.1 são exemplos de escalonadores complexos. O uso de um escalonador complexo, que utilize informações do estado dos caminhos, também não garante que a transferência multi-caminhos terá um bom desempenho [Singh et al., 2012]. O escalonador LowRTT é um exemplo disto. Ele utiliza informações de atraso dos caminhos (RTT) para distribuir os dados e mesmo assim, em alguns casos, alcança um desempenho pior que o alcançado por uma transferência TCP [Nikravesh et al., 2016].

Diante disso, nas duas próximas seções serão apresentados: uma análise da influência de uma caminho degradado pela razão de perda de pacotes (P) no desempenho da conexão MPTCP composta por dois caminhos, o melhor caminho e o caminho degradado, buscando definir o limiar de P, a partir do qual é melhor utilizar somente o melhor caminho; e um método dinâmico para filtragem dos caminhos disponíveis para o componente escalonador baseado na taxa de transferência do MPTCP e nas taxas de transferência estimadas dos subfluxos TCP que compõem a conexão MPTCP. O método de filtragem proposto independe do algoritmo escalonador adotado, funcionando como entrada para o algoritmo escalonador.

4.1 Análise do impacto de um caminho degradado no desempenho da transmissão MPTCP

Os protocolos de transporte são responsáveis pela comunicação fim-a-fim de processos de hospedeiros diferentes, permitindo a comunicação inter-processos. Os principais e mais conhecidos protocolos da camada de transporte são o UDP e o TCP, definidos pelas RFC 768 [Postel, 1980] e RFC 793 [Postel, 1981], respectivamente.

O UDP é um protocolo simples não orientado à conexão, e por isso, não garante a entrega, a ordem ou evita a duplicidade dos dados transmitidos. Já o TCP é um protocolo orientado à conexão que provê comunicação entre processos de hospedeiros diferentes com garantia de entrega em ordem e sem duplicidade dos dados transmitidos. Além disso, o TCP possui controle de congestionamento, o que permite que conexões TCP compartilhem com justiça um enlace congestionado. O algoritmo de controle de congestionamento mais tradicional e encontrado na maioria das implementações é o *New Reno*, especificado na RFC 6582 [Henderson et al., 2012]. Este é um algoritmo de prevenção de congestionamento de Aumento Aditivo/Diminuição Multiplicativa (AIMD).

Considerando o comportamento da taxa de transferência da conexão TCP por consequência do controle de congestionamento, de acordo com a RFC 3819 [Karn et al., 2004], podemos calcular a taxa de transferência média da conexão TCP (T) em função da razão de perda de pacotes (P), do tempo de viagem de ida e volta (RTT), do *time-out* do pacote (RTO) e do tamanho máximo de segmento (MSS). Uma simplificação adicional desta fórmula é geralmente feita assumindo que RTO é aproximadamente $5 \times RTT$, de acordo com a equação definida pela RFC 3819,

$$T = \frac{MSS}{RTT\sqrt{1.33P} + 5RTT.P(1+32P^2)min(1,3\sqrt{0.75P})}. \tag{4.1}$$

Para demonstrar o impacto da razão de perda de pacotes (P) no desempenho da taxa de transferência média de uma conexão TCP, o gráfico da Figura 4.1 apresenta o resultado da equação quando consideramos um cenário compostos por dois *host* conectados por um enlace com largura de banda de 20 Mbps e 10 ms de atraso, quando variamos P de 0 até 0.1.

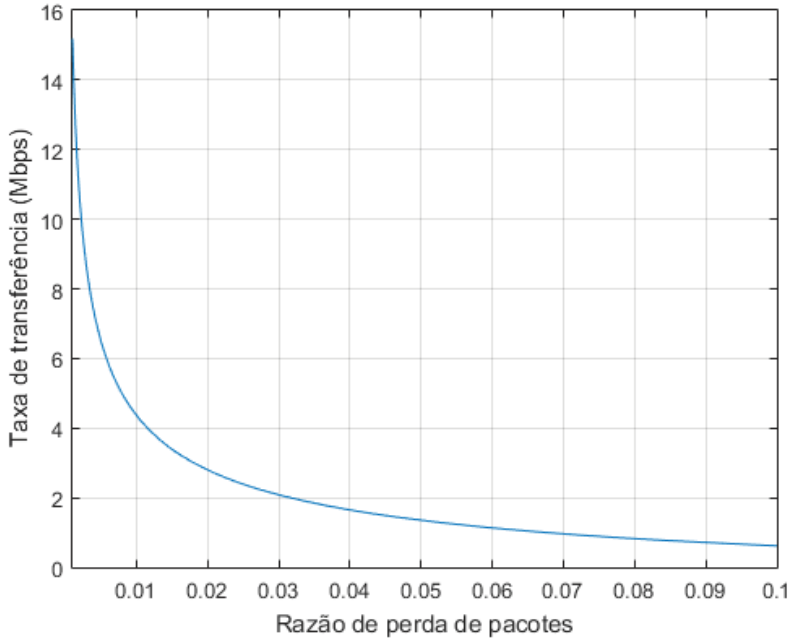


Figura 4.1: Taxa de transferência média da conexão TCP em função da razão de perda de pacotes (P), segundo a equação 4.1

Pelo gráfico, quando o valor de *P* é de apenas 1%, o valor da taxa de transferência média é de aproximadamente 4.4 Mbps, o que representa apenas 22% da largura de banda disponível no enlace. Quando *P* é de 10%, o enlace está muito degradado, tornando a comunicação impraticável, demonstrando o grande impacto da razão de perda de pacotes na taxa de transferência efetiva da conexão TCP padrão.

Observe que a largura de banda dos enlaces que compõem o caminho da conexão TCP não aparece explicitamente em (4.1). Tentar enviar mais rápido que a velocidade do enlace mais lento do caminho da conexão TCP faz com que a fila cresça no transmissor, causando congestionamento. Isso aumenta o RTT, o que, por sua vez, reduz a taxa de transferência.

Como já apresentado anteriormente, o MPTCP foi desenvolvido com o objetivo de ser uma extensão do protocolo TCP capaz de transferir dados através de múltiplos caminhos. A arquitetura do MPTCP, que se encontra na RFC 6182 [Ford et al., 2011], apresenta os objetivos funcionais, que definem os serviços e recursos que uma implementação do MPTCP deve possuir, e os objetivos de compatibilidade, que determinam como o protocolo deve interagir com outras entidades.

O primeiro objetivo funcional do MPTCP apresentado em sua arquitetura é a melhora da taxa de transferência. Em outras palavras, a taxa de transferência agregada deve ser melhor do que a conseguida por um único fluxo TCP quando da utilização do melhor caminho dentre os disponíveis. Esse objetivo nem sempre é atingido quando em ambientes de rede heterogêneas, principalmente em decorrência dos problemas já apresentados na Seção 2.3, o bloqueio HOL e o fenômeno *Bufferbloat*.

Desta forma, neste trabalho, propomos uma análise do limiar da razão de perda de pacotes de um caminho degradado, a partir da qual a taxa de transferência agregada do MPTCP seja pior do que a conseguida por um único fluxo TCP padrão pelo melhor caminho, pois, a partir deste limiar, o objetivo funcional de melhorar a taxa de transferência não será atingido pelo protocolo MPTCP.

4.2 Método de filtragem do caminho mais degradado

Em ambientes de rede heterogêneas, o grau de diversidade dos caminhos é um fator crítico para o desempenho da taxa de transferência agregada. Logo, o módulo de escalonamento deve levar em consideração as características dos caminhos que resultam em diferentes desempenhos atingidos pelos subfluxos. Por exemplo, um subfluxo lento, devido a alta perda de pacotes, resulta no baixo desempenho de um subfluxo rápido, causado pelo bloqueio do *buffer* de recebimento decorrente da chegada de pacotes fora de ordem [Hu et al., 2016].

Para mitigar esse efeito, este trabalho propõe um método de filtragem do caminho mais degradado, pois subfluxos TCP em caminhos com elevada taxa de erro têm seu desempenho degradado significativamente [Kelly, 2003] e, como já analisado na seção anterior, dependendo do impacto do caminho degradado na taxa de transferência do MPTCP, a utilização de um único fluxo TCP pelo melhor caminho pode conseguir melhor desempenho.

Dito isso, um ponto importante é definir o que será considerado um caminho degradado para o método de filtragem. Do que foi discutido, fica evidente que, o ponto a partir do qual a taxa de transferência do MPTCP é inferior à de um fluxo TCP pelo melhor caminho, pode ser considerado o momento em que o pior caminho da conexão MPTCP é classificado como degradado. Para estimar a taxa de transferência de um fluxo TCP padrão pelos caminhos disponíveis na conexão MPTCP utilizaremos a equação (4.1).

O método de filtragem do caminho mais degradado proposto independe do algoritmo escalonador escolhido para utilização pelo módulo de escalonamento, podendo funcionar por exemplo com o RR, LowRTT e outros algoritmos. A Figura 4.2 tem uma representação da posição do filtro antes do algoritmo de escalonamento. Assim, o filtro realiza a seleção dos caminhos e só então repassa o conjunto de caminhos disponíveis ao algoritmo de escalonamento.

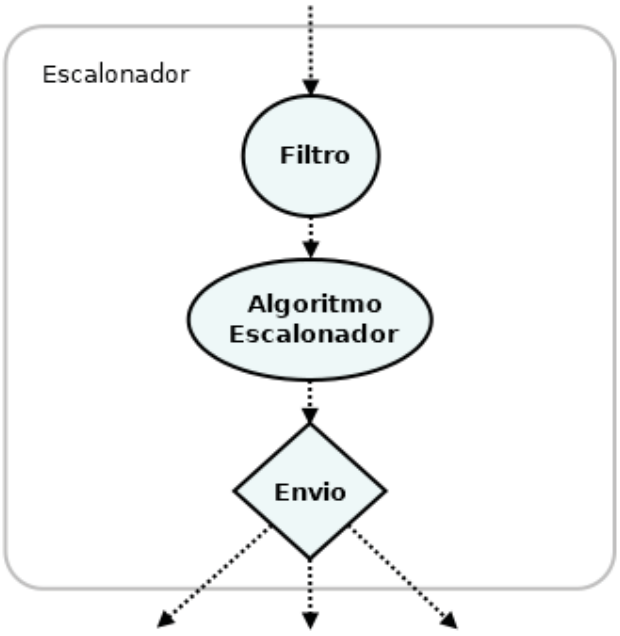


Figura 4.2: Representação da posição do filtro no componente escalonador

Para a realização da filtragem do caminho degradado, o método proposto realiza duas etapas em cada ciclo de escalonamento. A primeira é a etapa de classificação dos caminhos

quanto à degradação e a segunda é a etapa de filtragem dos caminhos disponíveis para o algoritmo de escalonamento. O algoritmo 1 apresenta o fluxo proposto pelo método.

Algoritmo 1

Definições:
p_i: caminho de índice i
T_i: estimativa da taxa de transferência de um fluxo TCP pelo caminho p_i
P_i: razão de perda de pacotes pelo caminho p_i
T_{mptcp}: EWMA da taxa de transferência agregada da transmissão MPTCP
T_{max}: registra o maior valor de T_i
id_{min}: registra o índice i do caminho de menor T_i
$p_{degradado}$: índice do caminho classificado como degradado
p_{list}: lista de caminhos disponíveis
$CWND_i$: janela de congestionamento do fluxo por p_i

$T_i \leftarrow \frac{MSS}{RTT_i\sqrt{1.33P_i}+5RTT_i.P_i(1+32P_i^2)min(1,3\sqrt{0.75P_i})}$
$T_{max} \leftarrow max(T_i)$
$id_{min} \leftarrow argmin(Ti)$
if $T_{mptcp} < T_{max}$ **then**
  if $CWND_{id_{min}} == min(CWND_i)$ **then**
    $p_{degradado} \leftarrow id_{min}$ // classifica o caminho de índice id_{min} como degradado
  end if
end if
for all p_i **do**
  if $p_i \neq p_{degradado}$ **then**
    $p_{list} \leftarrow p_i$ // coloca p_i na lista de caminhos disponíveis
  end if
end for

A. Classificação dos caminhos quanto à degradação

Suponha um cenário de rede composto por dois *hosts* conectados por dois caminhos identificados por p_1 e p_2. Para a classificação dos caminhos é necessário monitorar três características dinâmicas dos dois caminhos: o RTT, a razão de perda de pacotes (P) e a janela de congestionamento ($CWND$). De posse dos valores das duas primeiras características, calcula-se a estimativa da taxa de transferência (T_i) de um único fluxo TCP em cada caminho com a equação 4.1.

Então calcula-se a taxa de transferência agregada da transmissão, T_{mptcp}, dividindo-se a quantidade de bytes enviados durante um intervalo de tempo Δ_t, por esse mesmo intervalo de tempo Δ_t, de acordo com

$$T_{mptcp} = \frac{\text{bytes enviados}}{\Delta_t}.$$

Buscando suavizar o impacto de grandes variações no cálculo da estimativa de T_{mptcp}, utilizamos uma média móvel exponencial ponderada (EWMA) com o valor de α igual a 0,125.

Em seguida, compara-se a T_{mptcp} calculada com a máxima taxa de transferência TCP estimada,

$T_{mptcp} < T_{max}$.

Se a taxa de transferência do MPTCP for menor que a máxima taxa de transferência TCP estimada para um caminho, verifica-se se o pior caminho quanto à taxa de transferência é o que possui a menor janela de congestionamento. Se sim, o caminho é classificado como degradado ($p_{degradado}$):

$id_{min} \leftarrow argmin(Ti)$ –> registra o índice (i) do caminho de menor T; e

Se $CWND_{id_{min}} == min(CWND_i)$ então $p_{degradado} \leftarrow id_{min}$

Essa verificação com relação à janela de congestionamento tem o objetivo de garantir que o subfluxo só será classificado como degradado caso este tenha a menor $CWND$, pois a pior $CWND$ tende a refletir a pior taxa de transferência dentre os subfluxos da conexão, principalmente em cenários de rede onde o atraso dos caminhos não difere muito significativamente. Essa estratégia visa mitigar a probabilidade do caminho classificado como degradado, ou seja, aquele que tem maior probabilidade de causar o baixo desempenho da transmissão MPTCP, ser o de maior taxa de transferência naquele momento. Com isso, aumenta-se a chance de cumprimento do objetivo funcional nº 1 da arquitetura (uma transmissão MPTCP não deve ter menos desempenho do que uma única conexão TCP padrão pelo melhor caminho disponível).

Caso a T_{mptcp} não seja menor que a máxima T_i estimada para os fluxos TCP e/ou o pior caminho não seja o de menor $CWND$, então nenhum dos caminhos disponíveis é classificado como degradado.

B. Filtragem dos caminhos disponíveis para o algoritmo de escalonamento

Nesta etapa, o filtro apenas atualiza a lista de caminhos disponíveis para o algoritmo de escalonamento, retirando da lista o caminho classificado como degradado na etapa inicial.

Para todo caminho i:

Se $pi \neq p_{degradado}$ –> coloca p_i na lista de caminhos disponíveis

Esse método permite melhorar o desempenho da transferência multi-caminhos em ambientes heterogêneos, como redes sem fio sujeitas a perda de pacotes, quando dinamicamente um subfluxo degradado é excluído dos subfluxos disponíveis para envio de dados. Dessa forma, o desempenho dos outros subfluxos deixa de sofrer o impacto de um subfluxo demasiadamente degrado e a taxa de transferência agregada da conexão MPTCP tende a melhorar.

É importante observar que esse processo ocorre a cada ciclo de escalonamento, ou seja, um subfluxo excluído por ter sido classificado como degradado pode retornar dinamicamente à lista de subfluxos disponíveis para envio no próximo ciclo de escalonamento, caso este não seja novamente classificado como degradado.

4.3 Limitações

O método de filtragem proposto nesse trabalho têm como objetivo mitigar o impacto na taxa de transferência agregada do MPTCP de um subfluxo por um caminho degradado, em um cenário de rede heterogênea como o encontrado em redes sem fio. Por isso, o foco na elaboração do método foi delimitado somente para esse escopo reduzido. Casos genéricos não são tratados por essa proposta.

Este trabalho realiza uma análise do valor do *limiar* de perda de pacotes a partir do qual um caminho degradado impacta na transmissão MPTCP ao ponto de ser melhor utilizar um fluxo TCP pelo melhor caminho. Porém, essa análise será realizada em um cenário comumente encontrado em dispositivos móveis como *smartphones*, ou seja, uma conexão WiFi e outra de rede celular. Cenários distintos não serão abordados.

4.4 Comparação com trabalhos relacionados

A coletânea de trabalhos apresentada no capítulo 3 nos fornece exemplos de diferentes estratégias de escalonamento baseadas nas características dos caminhos, desde algoritmos que utilizam atributos dos caminhos na escolha do caminho para envio, até aqueles que possuem estratégias diferenciadas para a ordem de envio dos segmentos. Todos estes trabalhos influenciaram e embasaram a proposta do método de filtragem apresentado.

Este trabalho se diferencia por propor um método dinâmico de filtragem de caminho degradado que pode ser utilizado em conjunto com diferentes algoritmos de escalonamento, mas também por propor uma análise do limiar da razão de perda de pacotes a partir da qual a taxa de transferência de um fluxo TCP padrão pelo melhor caminho disponível é superior ao da transmissão MPTCP.

4.5 Resumo do capítulo

Este capítulo apresentou a proposta deste trabalho dentro de seu contexto de aplicação, uma análise do impacto de um caminho degradado no desempenho da transmissão MPTCP, um método de filtragem do caminho mais degradado do conjunto de subfluxos disponíveis para escalonamento com o objetivo de mitigação do impacto no desempenho da transmissão MPTCP, discutiu as suas limitações e fez uma breve comparação deste trabalho com a coletânea do capítulo de trabalhos relacionados.

Capítulo 5

Metodologia e implementação

Este capítulo descreve a pesquisa exploratória realizada sobre a análise do impacto do caminho degradado e do método de filtragem do caminho degradado proposto. Buscando confirmar ou refutar as conclusões das análises e discussões preliminares realizadas e apresentadas no capítulo anterior, este trabalho seguiu a linha experimental. Em um simulador de eventos discretos, um cenário de rede heterogênea foi construído e simulado, aplicando mudanças sugeridas de forma controlada e medindo a sua influência no desempenho da transmissão MPTCP. O texto a seguir discorre sobre a metodologia empregada, sobre as implementações necessárias à análise e ao método proposto, sobre o cenário e as adaptações necessárias para a simulação, sobre os resultados obtidos e demais recursos utilizados para a avaliação.

5.1 Ambiente de simulação

Os problemas observados em transferências multi-caminhos com o protocolo de transporte MPTCP em ambientes de rede heterogêneas têm sido amplamente estudados pela comunidade acadêmica. Por serem problemas complexos que envolvem muitas variáveis e elementos, como por exemplo diferentes nós, enlaces, pilhas de protocolos, taxas de erro, agentes e outros, além de possuírem comportamento dinâmico, realizar um estudo analítico se torna uma tarefa muito complexa. Por esse motivo, a grande maioria dos trabalhos sobre esse tema optam por seguir uma linha de pesquisa experimental em ambiente simulado de rede, realizando mudanças controladas no cenário de rede e medindo o resultado obtido. Seguindo esta mesma linha, as propostas que são objeto desse estudo foram avaliadas através de simulações, o que permite obter dados mais concretos, com diferentes tipos de cenários de rede e contextos.

A análise do impacto do caminho degradado e o método de filtragem propostos foram reproduzidos e avaliados através do simulador de eventos discretos NS-3 [NS-3 Consortium, 2011]. Esta ferramenta foi escolhida pela sua ampla utilização em estudos voltados para sistemas de Internet. O seu código é aberto, baseado nas linguagens C++ e *Python*. Além do mais, o código está disponível para pesquisadores, permitindo a implementação das mudanças sugeridas e possibilitando a inclusão de funções para coleta de dados e a aplicação das métricas selecionadas para a avaliação. Com isto, os experimentos desenvolvidos nesta pesquisa podem ser reproduzidos e validados pela comunidade acadêmica e demais interessados.

O simulador NS-3 não possui um módulo para o MPTCP implementado e validado em sua versão oficial. Para a realização das simulações no NS-3 utilizamos o módulo do MPTCP proposto em [Kheirkhah et al., 2014][1], que está em conformidade com a RFC 6824

[1]Disponível em https://github.com/mkheirkhah/mptcp

[Ford et al., 2013] e segue de perto o *design* do kernel Linux MPTCP [Paasch et al., 2013]. O módulo do MPTCP utilizado foi desenvolvido baseado na versão 3.19 do NS-3. Todos os protocolos e elementos necessários para a simulação e avaliação estão presentes nessa versão, portanto, não foi necessário realizar a atualização para versão mais recente.

Essa implementação do módulo MPTCP para o NS-3 seguiu o previsto pela RFC 6824. Como as RFCs que tratam do MPTCP não padronizaram um escalonador para o protocolo, o desenvolvedor do módulo decidiu por utilizar o algoritmo *Round Robin* (RR) para o escalonamento. O problema é que a implementação do algoritmo não foi modular, ou seja, não era possível desacoplar o algoritmo RR sem ter que alterar outras funções do protocolo. Por esse motivo, o componente de escalonamento, o algoritmo RR e a função de envio dos segmentos tiveram de ser reescritas para desacoplar a função de seleção do algoritmo escalonador utilizado pelo MPTCP.

Como foi apresentado na subseção 2.2.2, todos os algoritmos escalonadores implementados no trabalho utilizaram a opção adotada nas implementações atuais do *Linux Multipath TCP*. Nessa opção, os dados encaminhados pela aplicação são inicialmente adicionados a um *buffer* compartilhado de envio para depois serem distribuídos segmentados aos subfluxos pelo escalonador de acordo com a disponibilidade de espaço na janela de congestionamento (CWND) do subfluxo. O caminho selecionado para receber os segmentos terá sua janela esgotada antes de dar início a outro ciclo de escalonamento, a menos que a quantidade de dados a ser enviada seja menor que o espaço disponível na CWND.

Para a avaliação do desempenho do método de filtragem proposto escolhemos comparar o desempenho alcançado pela transmissão MPTCP quando esta utiliza os algoritmos de escalonamento *Round Robin* e o *Lowest-RTT-First* (LowRTT), com o desempenho alcançado quando estes mesmos algoritmos trabalham em conjunto com o método de filtragem. Para isso, este trabalho utilizou como referência a implementação do algoritmo **Fastest RTT** presente na classe `MpTcpSchedulerFastestRTT` do trabalho [Coudron e Secci, 2017]. A implementação do RR citada anteriormente também foi baseada na classe `MpTcpSchedulerRoundRobin` desse mesmo trabalho.

Para adicionar erro aos enlaces optou-se por utilizar a classe `RateErrorModel` do NS-3 que permite aplicar diferentes razões de erro de pacote (P) aos caminhos. Nessa classe, os pacotes são marcados como errados segundo uma distribuição aleatória. Para as nossas simulações utilizamos a distribuição uniforme e configuramos a classe para utilizar a unidade de erro como sendo o pacote.

5.2 Cenários de simulação

A topologia base dos cenários de simulação utilizados para os experimentos é a apresentada na Figura 5.1, e é composta por dois *hosts*, conectados por dois caminhos com características diferentes. Os dois caminhos foram modelados utilizando enlaces ponto-a-ponto (*Point-To-Point Model*) e foram configurados parâmetros para representar as características desejadas para os caminhos. Optamos por essa forma de modelagem porque o importante para a simulação era o resultado do ponto de vista dos protocolos da camada de transporte (fim-a-fim), não sofrendo interferência de protocolos das camadas inferiores. No terminal transmissor foi instalado uma aplicação geradora de fluxo de dados que utiliza o protocolo MPTCP, `MpTcpBulkSend`, e no terminal receptor uma aplicação consumidora de fluxo de dados que também utiliza o MPTCP, `MpTcpPacketSink`. O gerador de tráfego envia dados o mais rápido possível até atingir a capacidade máxima ou até que o aplicativo seja parado. Uma vez que o *buffer* de envio da camada inferior é preenchido, ele aguarda até que tenha espaço disponível para enviar mais dados, essencialmente mantendo um fluxo constante de dados.

O controle de congestionamento do TCP padrão adotado foi o *New Reno* e para o MPTCP utilizamos o *Linked Increase Algorithm* (LIA), que provê um desempenho similar ao conseguido pelo *New Reno* para um fluxo TCP. O tipo de fila nos dispositivos de rede adotado foi o *Droptail* com limite da fila de 50 pacotes. O tamanho do pacote foi configurado em 1400 *Bytes*. Para outros parâmetros de simulação não detalhados utilizou-se os valores sugeridos para o MPTCP ou os valores padrões do simulador NS-3.

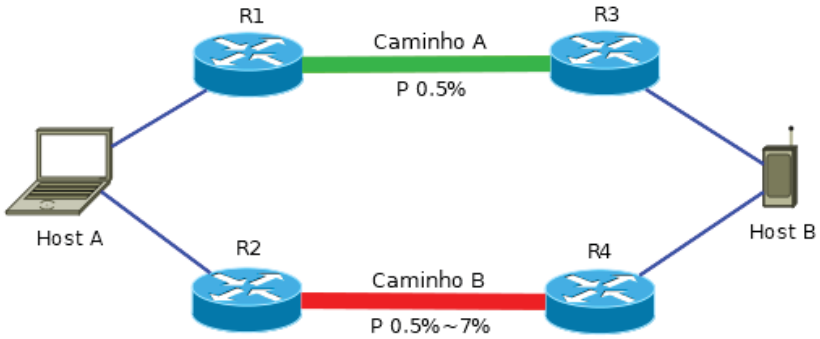


Figura 5.1: Topologia de rede utilizada nas simulações

5.2.1 Cenário para a análise do impacto do caminho degradado

Na análise do impacto do caminho degradado no desempenho da transmissão MPTCP, o caminho A, identificado como o melhor caminho, possui largura de banda de 10Mbps, atraso de 10ms e razão de perda de pacotes de 0.005 (0.5%). Valores próximos aos comumente encontrados em um enlace com tecnologia WiFi/IEEE 802.11b. Já o caminho B, identificado como o caminho degradado, possui largura de banda de 2Mbps, atraso de 10ms e razão de perda de pacotes variando de 0.005 (0.5%) até 0.07 (7%). Esses valores são próximos aos encontrados em redes móveis 3G. O algoritmo escalonador do MPTCP utilizado foi o padrão do módulo, *Round-Robin*. A Tabela 5.1 resume as configurações dos caminhos usadas nas simulações para a análise.

Tabela 5.1: Parâmetros das simulações para a análise do impacto de um caminho degradado na transmissão MPTCP

Parâmetro	**Caminho A**	**Caminho B**
Tecnologia	WiFi/IEEE 802.11b	UMTS (3G)
Largura de banda	10Mbps	2Mbps
Atraso	10ms	10ms
Limite da fila	50	50
Tipo de fila	Droptail	Droptail

Os valores de P para o caminho B simulados foram de 0.5%, 1% e, assim por diante, incrementando em 1% até alcançar 7%. Para cada valor de P foram realizadas 10 simulações de 60s de duração, com diferentes sementes geradoras de números aleatórios a cada repetição. Como indicador de desempenho utilizamos o *Goodput* médio, tanto para o TCP padrão pelo melhor

caminho como para o MPTCP. O limiar será o valor de *P* do caminho degradado (caminho B) imediatamente menor ao do *P* cujo o *Goodput* médio da conexão MPTCP obteve um desempenho inferior ao conseguido pelo TCP padrão pelo melhor caminho (caminho A).

Para análise dos resultados alcançados pelo MPTCP agregado e por cada subfluxo em comparação ao desempenho de uma única conexão TCP pelo melhor caminho disponível, simulamos um fluxo TCP pelo caminho A, com razão de perda *P* igual a 0.5% e com os mesmos parâmetros apresentados na Tabela 5.1 para o caminho A. Foram realizadas 10 simulações de 60s de duração, com diferentes sementes geradoras de números aleatórios a cada repetição. O valor da média foi utilizado como linha de referência para a realização das análises.

5.2.2 Cenários para análise do desempenho do método de filtragem

Os cenários utilizados nas simulações para a análise do desempenho da transmissão MPTCP, quando o método de filtragem do caminho degradado está em uso, também é o apresentado na Figura 5.1.

No primeiro cenário, o caminho A possui largura de banda de 10Mbps, atraso de 5ms e razão de perda de pacotes de 0.005 (0.5%). Já o caminho B possui largura de banda de 2Mbps, atraso de 5ms e razão de perda de pacotes variando de 0.005 (0.5%) até 0.07 (7%). A Tabela 5.2 resume os parâmetros utilizados para os caminhos.

Tabela 5.2: Parâmetros do primeiro cenário para análise do método de filtragem

Parâmetro	**Caminho A**	**Caminho B**
Largura de banda	10Mbps	2Mbps
Atraso	5ms	5ms
Limite da fila	50	50
Tipo de fila	Droptail	Droptail

No segundo cenário, o caminho A possui largura de banda de 11Mbps, atraso de 10ms e razão de perda de pacotes de 0.005 (0.5%). Já o caminho B possui largura de banda de 10Mbps, atraso de 10ms e razão de perda de pacotes variando de 0.005 (0.5%) até 0.07 (7%). A Tabela 5.3 resume os parâmetros utilizados para os caminhos. Neste cenário, buscamos verificar se, com os dois caminhos com largura de banda disponível com valores próximos, o limiar de razão de perda de pacotes será mantido ou até mesmo se o impacto do caminho degradado no *Goodput* será similar ao analisado anteriormente.

Tabela 5.3: Parâmetros do segundo cenário para análise do método de filtragem

Parâmetro	**Caminho A**	**Caminho B**
Largura de banda	11Mbps	10Mbps
Atraso	10ms	10ms
Limite da fila	50	50
Tipo de fila	Droptail	Droptail

Para a avaliação do desempenho do método de filtragem proposto, será comparado o *Goodput* alcançado pela transmissão MPTCP, quando esta utiliza os algoritmos de escalonamento

RR e o LowRTT, com o desempenho alcançado quando estes mesmos algoritmos trabalham em conjunto com o método de filtragem. Será comparado o *Goodput* alcançado pelo RR em comparação com o RR+F (RR com filtro), assim como o *Goodput* alcançado pelo LowRTT em comparação com o LowRTT+F (LowRTT com filtro).

Os valores de P para o caminho B simulados, em ambos os cenários de simulação, foram de 0.5%, 1% e, assim por diante, incrementando em 1% até alcançar 7%. Para cada valor de P foram realizadas 100 simulações de 60s de duração, com diferentes sementes geradoras de números aleatórios a cada repetição, com intervalo de confiança de 95%.

5.3 Métricas

A principal métrica do ponto de vista da aplicação, utilizada como indicador de desempenho neste trabalho, é a taxa de transferência útil agregada (***Goodput***). O *Goodput* foi calculado de acordo com,

$$Goodput = \frac{\text{Bytes Recebidos}}{\text{Tempo de Simulação}}, \tag{5.1}$$

onde a quantidade de *Bytes* recebidos pela a aplicação foi dividida pelo tempo de simulação.

Para a avaliação do desempenho dos algoritmos pelo *Goodput*, foram coletados dados dos subfluxos utilizando duas estratégias. A primeira foi a utilização da classe `flowmonitor` presente na versão 3.19 do NS-3. O objetivo do módulo `FlowMonitor` é fornecer um sistema flexível para medir o desempenho dos protocolos de rede. O módulo usa monitores, instalados em nós da rede, para rastrear os pacotes trocados pelos nós e medir uma série de parâmetros. Os pacotes são divididos de acordo com o fluxo a que pertencem, onde cada fluxo é definido de acordo com a 4-tupla que o origina. Os dados coletados de cada fluxo são os da Tabela 5.4.

Uma vez que os pacotes são rastreados no nível de IP, qualquer retransmissão causada por protocolos de transporte, como o MPTCP, será vista pelo monitor como um novo pacote. Por esse motivo, para o cálculo do *Goodput*, taxa de transferência útil entre as aplicações, foi implementada uma função batizada de `Progress()` que retorna o valor do *Goodput*, simplesmente guardando a quantidade de dados que foi recebido pela aplicação destino, *sink*, durante cada segundo de simulação. Além disso, ao final de cada simulação é calculado a *Average Goodput* pelo total de dados recebidos na aplicação dividido pelo tempo de simulação.

Em cada cenário, para cada um dos algoritmos foram realizadas 100 simulações de 60 segundos de duração, aplicando diferentes sementes geradoras de números aleatórios a cada repetição, com intervalo de confiança de 95%. A parametrização dos caminhos e a geração de tráfego foram conferidas através do utilitário [2]*NetAnim*, capaz de apresentar visualmente os dados coletados. Desse modo, buscou-se garantir fidedignidade ao trabalho, assegurando o rigor científico necessário para a avaliação do experimento.

Para a análise dos resultados de *Goodput* dos algoritmos RR e LowRTT, com e sem o método de filtragem, este trabalho utilizou o *software* MATLAB para gerar gráficos do tipo *plot*, *errobar* e *boxplot*, com os dados obtidos nas 100 (cem) simulações de cada algoritmo.

[2]https://www.nsnam.org/wiki/NetAnim

Tabela 5.4: Dados dos subfluxos coletados pelo `FlowMonitor`

Dado Coletado	**Descrição**
timeFirstTxPacket	Quando o primeiro pacote do fluxo foi transmitido
timeLastTxPacket	Quando o último pacote do fluxo foi transmitido
timeFirstRxPacket	Quando o primeiro pacote do fluxo foi recebido
timeLastRxPacket	Quando o último pacote do fluxo foi recebido
delaySum	Soma dos atrasos de todos os pacotes recebidos no fluxo
jitterSum	Soma das variações do atraso de todos os pacotes recebidos no fluxo
txBytes, txPackets	Número total de Bytes/Pacotes transmitidos no fluxo
rxBytes, rxPackets	Número total de Bytes/Pacotes recebidos no fluxo
lostPackets	Número total de pacotes que se pressupõem perdidos
timesForwarded	O número de vezes que um pacote foi reenviado
delayHistogram, jitterHistogram, packetSizeHistogram	Versões de histograma para os tamanhos do atraso, jitter e pacote, respectivamente
packetsDropped, bytesDropped	O número de pacotes e bytes perdidos

5.4 Resumo

Este capítulo apresentou uma descrição detalhada do ambiente de simulação e do cenário utilizado para estudar o impacto de um caminho degradado na transmissão MPTCP e também o desempenho do método de filtragem de caminho degradado proposto. A metodologia e as ferramentas utilizadas foram descritas com as suas devidas justificativas. Por fim, a métrica empregada foi retratada e detalhada, bem como o processo de avaliação do desempenho. Dessa maneira, espera-se que o experimento possa ser reproduzido pela comunidade acadêmica e demais interessados.

Capítulo 6

Resultados

Este capítulo apresenta e analisa os resultados obtidos nas simulações dos cenários dessa pesquisa. No capítulo anterior foram explicados os cenários de simulação e os seus respectivos objetivos. Para facilitar o entendimento no decorrer do trabalho, este capítulo está dividido em seções de acordo com os cenários apresentados. Por fim, a seção 6.3 apresenta um resumo dos resultados.

6.1 Impacto do caminho degradado na transmissão MPTCP

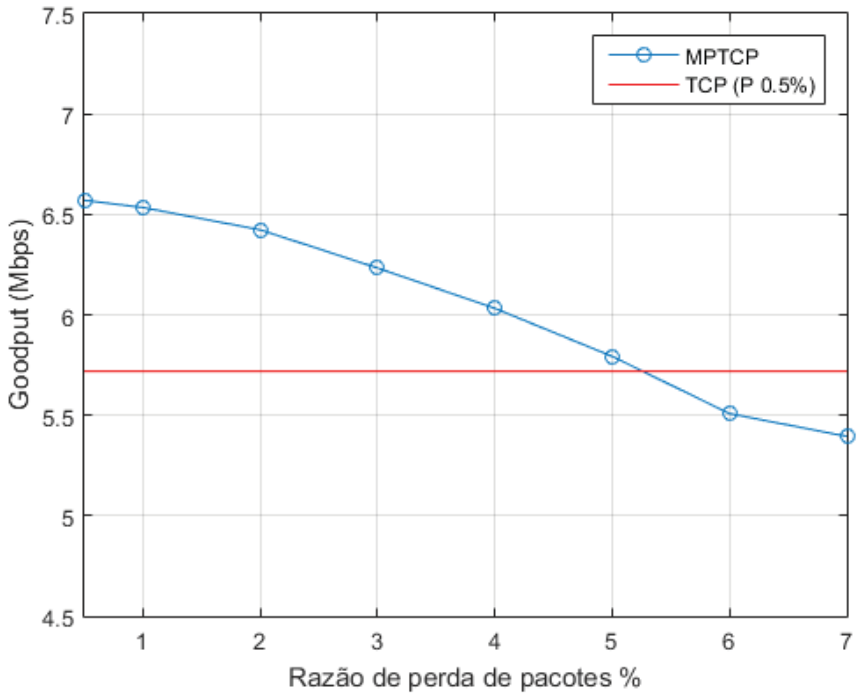


Figura 6.1: Limiar - *Goodput* do MPTCP vs TCP de referência

A Figura 6.1 apresenta o *Goodput* alcançado pela conexão MPTCP para os diferentes valores de P. Foi traçado uma linha de referência que corresponde ao *Goodput* do fluxo TCP padrão pelo melhor caminho (caminho A) e com P igual a 0.5%. O objetivo da linha de referência é verificar o momento de interseção entre as curvas, ou seja, onde o *Goodput* do

MPTCP cruza o limiar a partir do qual é melhor utilizar somente o melhor caminho. Pelo gráfico percebemos que entre 5% e 6% de *P* ocorre a interseção, logo, o limiar para o cenário proposto pode ser estabelecido como 5%. Qualquer valor de *P* acima de 5% para o caminho degradado irá gerar perda de desempenho na conexão MPTCP a ponto de não compensar a sua utilização, contrariando o objetivo funcional nº 1 da arquitetura do MPTCP [Ford et al., 2011].

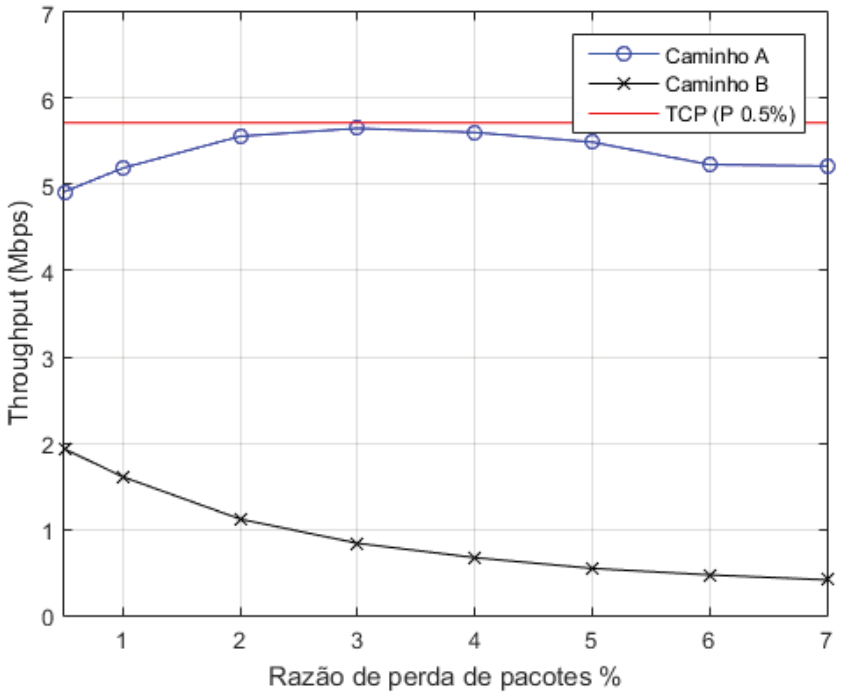


Figura 6.2: *Throughput* do melhor caminho (A) na conexão MPTCP e na conexão TCP

No gráfico da Figura 6.2 são apresentados o *throughput* em função de *P* dos subfluxos da conexão MPTCP pelos caminhos A e B e a linha de referência correspondente ao *throughput* do fluxo TCP padrão utilizando o caminho A. Podemos observar que, para todos os valores de *P* simulados, o desempenho do subfluxo da conexão MPTCP pelo caminho A é pior do que o do fluxo TCP padrão pelo mesmo caminho. Isso ocorre devido aos problemas da conexão MPTCP em cenários de rede heterogênea, já discutidos na seção 2.3, decorrentes do reordenamento dos pacotes no *buffer* de recebimento causados pelo caminho degradado.

Além disso, o desempenho do subfluxo do caminho A aumenta no intervalo de 0 a 3% de *P*. No início do intervalo citado, o desempenho do subfluxo do caminho A não sofre com o efeito do reordenamento no *buffer* de recebimento decorrente das perdas de pacotes que ocorrem no caminho B. O escalonador do MPTCP seleciona um subfluxo para enviar os segmentos de acordo com o critério adotado em seu algoritmo, então envia os segmentos para a sua janela de congestionamento (CWND). Quando a CWND enche, o escalonador escolhe outro subfluxo entre os com CWND disponível para o envio. Com o aumento de *P* pelo caminho B, o controle de congestionamento reduz o tamanho da CWND do subfluxo pelo caminho B e, como consequência, o escalonador acaba por enviar mais segmentos pelo subfluxo do caminho A, pelo fato deste possuir uma maior CWND, refletindo na tendência de crescimento. Porém, quando o valor de *P* passa de 3%, o subfluxo do caminho A começa a sofrer com a perda de pacotes pelo caminho B, e o seu desempenho tende a diminuir. Isso ocorre pela ação do controle de fluxo e do controle de congestionamento.

6.2 Análise do desempenho do método de filtragem

A Figura 6.3 apresenta o *Goodput* médio alcançado pela transmissão MPTCP quando da utilização do algoritmo RR sem o método de filtragem e com o método de filtragem, no primeiro cenário descrito na seção 5.2.2, de acordo com os parâmetros da tabela 5.2, para os diferentes valores de razão de perda de pacotes. Podemos observar que, para valores de P menores que 4%, o resultado do RR é pouco superior ao alcançado pelo RR+F, sendo insignificante para P de 0.5%, aproximadamente 1,5% maior para P de 1% e 2% e novamente insignificante para P de 3% e 4%. Porém, a partir deste ponto, o *Goodput* médio alcançado com o RR+F foi superior ao conseguido pelo RR, ficando aproximadamente 2.5% maior para P de 5%, 3.5% maior para P de 6% e em torno de 4.5% maior para P de 7%. A Tabela 6.1 apresenta um resumo do ganho percentual de *Goodput* alcançado pelo RR+F em relação ao RR no primeiro cenário.

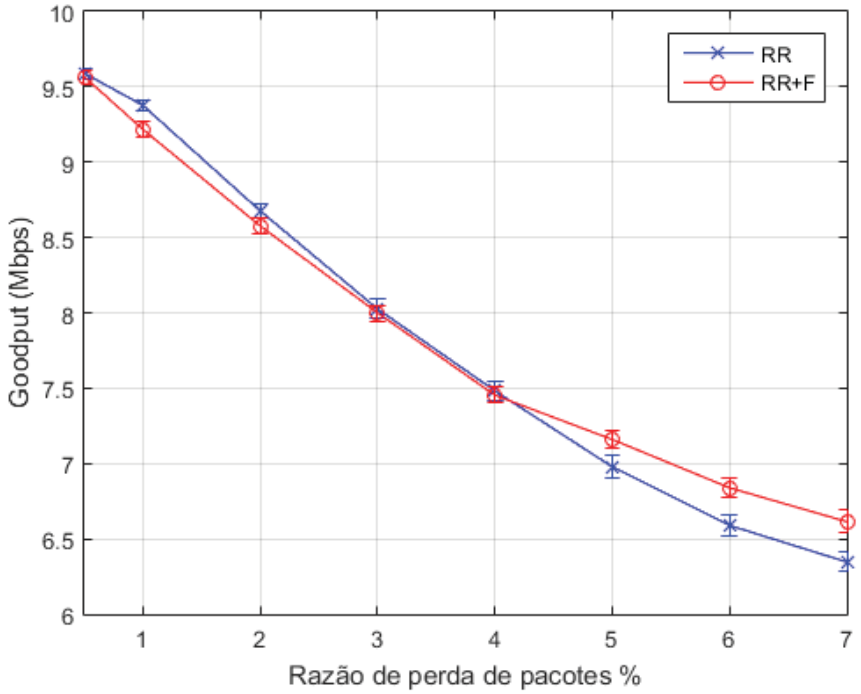


Figura 6.3: *Goodput* médio alcançado pelo RR e pelo RR+F no 1º cenário

Esse resultado é coerente com o observado na análise do impacto do caminho degradado no desempenho da transmissão MPTCP, onde encontramos que, para o cenário estudado, o valor de 5% de razão de perda de pacotes é o limiar a partir do qual é melhor a utilização de um fluxo TCP padrão pelo melhor caminho. A partir de P igual a 5%, o RR com o método de filtragem passa a obter um melhor desempenho na transmissão em comparação com o RR sem o método de filtragem.

Na Figura 6.4 temos o *boxplot* dos resultados de *Goodput* obtidos nas 100 simulações realizadas para os diferentes valores de razão de perda de pacotes, para o RR e para o RR+F. Podemos observar a variação dos resultados alcançados e perceber que houveram poucos resultados com valores discrepantes, distantes da mediana em mais do que 1,5 vezes a amplitude interquartil, nos resultados das simulações.

Na Figura 6.5 observamos o *Goodput* médio alcançado pela transmissão MPTCP quando da utilização do algoritmo LowRTT sem o método de filtragem e com o método de filtragem, no primeiro cenário descrito na seção 5.2.2, de acordo com os parâmetros da Tabela 5.2, para

Tabela 6.1: Resumo comparativo do ganho percentual do *Goodput* alcançado pelo RR+F em relação ao RR no 1º cenário

Resumo comparativo (RR e RR+F) - 1º cenário								
P	0.5%	1%	2%	3%	4%	5%	6%	7%
Goodput (Ganho Percentual)	0%	-1.5%	-1.5%	0%	0%	2.5%	3.5%	4.5%

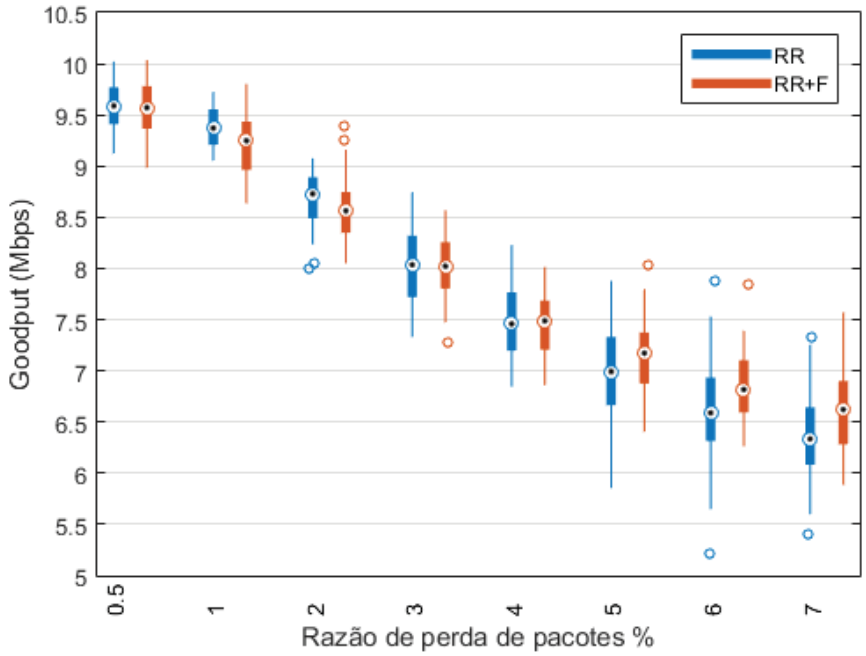


Figura 6.4: *Boxplot* do *Goodput* alcançado pelo RR e pelo RR+F no 1º cenário

os diferentes valores de razão de perda de pacotes. Podemos observar que para valores de *P* menores que 2%, o resultado do LowRTT é pouco superior ao alcançado pelo LowRTT com o filtro, sendo 0.5% maior para *P* de 0.5%, aproximadamente 1,5% maior para *P* de 1% e 2%. Já para *P* igual a 3% o LowRTT+F foi superior em 1.5%. Para *P* igual a 4% o resultado foi idêntico, porém, a partir deste ponto, o *Goodput* médio alcançado com o LowRTT+F foi sempre superior ao conseguido pelo LowRTT, ficando aproximadamente 3% maior para *P* de 5%, em torno de 4.5% maior para *P* de 6% e 5% maior para *P* de 7%. A Tabela 6.2 apresenta um resumo do ganho percentual de *Goodput* alcançado pelo LowRTT+F em relação ao LowRTT no primeiro cenário.

Tabela 6.2: Resumo comparativo do ganho percentual do *Goodput* alcançado pelo LowRTT+F em relação ao LowRTT no 1º cenário

Resumo comparativo (LowRTT e LowRTT+F) - 1º cenário								
P	0.5%	1%	2%	3%	4%	5%	6%	7%
Goodput (Ganho Percentual)	-0.5%	-1.5%	-1.5%	1.5%	0%	3%	4.5%	5%

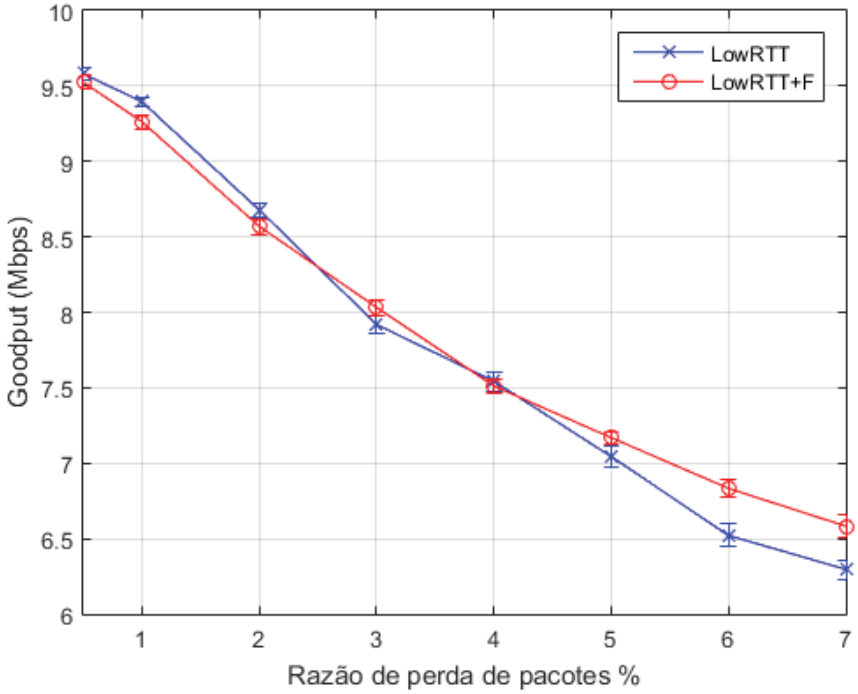


Figura 6.5: *Goodput* médio alcançado pelo LowRTT e pelo LowRTT+F no 1º cenário

De forma semelhante, na Figura 6.6, temos o *boxplot* dos valores de *Goodput* alcançados nas simulações realizadas para os diferentes valores de razão de perda de pacotes, com o algoritmo LowRTT e o LowRTT com o filtro. Assim como ocorreu com o RR e o RR+F, percebemos poucos resultados discrepantes nas simulações.

A Figura 6.7 apresenta o *Goodput* médio alcançado pela transmissão MPTCP quando da utilização do algoritmo RR sem o método de filtragem e com o método de filtragem, no segundo cenário descrito na seção 5.2.2, de acordo com os parâmetros da Tabela 5.3, para os diferentes valores de razão de perda de pacotes. Podemos observar que, para valores de P menores que 3%, o resultado do RR é pouco superior ao alcançado pelo RR+F, sendo aproximadamente iguais para P de 0.5%, 1% maior para P de 1% e 1.5% maior para P de 2%. Em P igual a 3% novamente o desempenho foi semelhante. Porém, a partir de 4% de P, o *Goodput* médio alcançado com o RR+F foi superior ao conseguido pelo RR, ficando 2% maior para P de 4%, aproximadamente 3% maior para P de 5%, 4% maior para P de 6% e em torno de 5.5% maior para P de 7%. A Tabela 6.3 apresenta um resumo do ganho percentual de *Goodput* alcançado pelo RR+F em relação ao RR no segundo cenário.

Tabela 6.3: Resumo comparativo do ganho percentual do *Goodput* alcançado pelo RR+F em relação ao RR no 2º cenário

Resumo comparativo (RR e RR+F) - 2º cenário								
P	0.5%	1%	2%	3%	4%	5%	6%	7%
Goodput (Ganho Percentual)	0%	-1%	-1.5%	0%	2%	3%	4%	5.5%

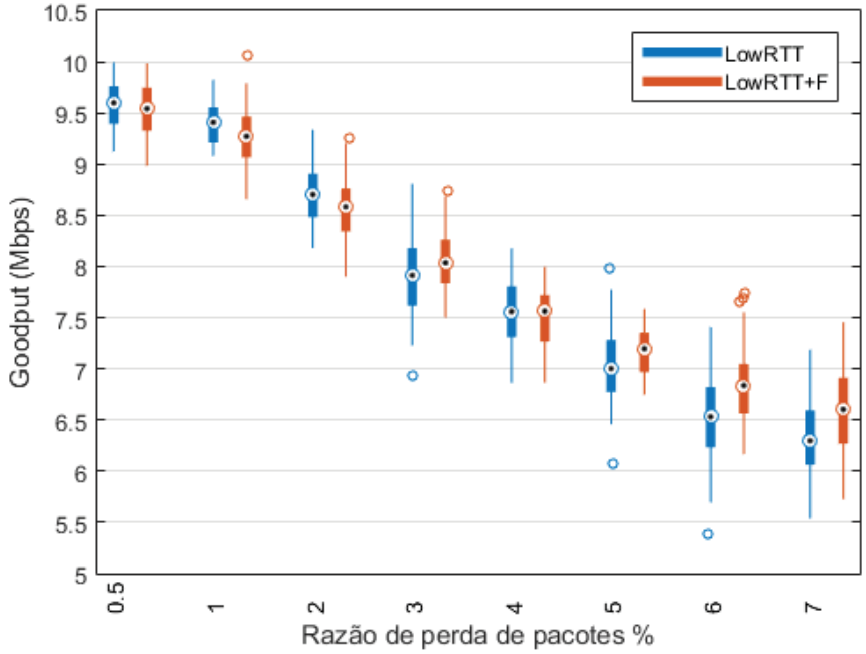


Figura 6.6: *Boxplot* do *Goodput* alcançado pelo LowRTT e pelo LowRTT+F no 1º cenário

O resultado mostra um limiar de razão de perda de pacotes menor do que o encontrado no primeiro cenário, mais precisamente em P igual a 3%. Nesse cenário, para P maior que 3%, o desempenho alcançado pelo RR com filtro é sempre superior que o alcançado pelo RR sem filtro.

Na Figura 6.8 observamos o *Goodput* médio alcançado pela transmissão MPTCP quando da utilização do algoritmo LowRTT sem o método de filtragem e com o método de filtragem, no segundo cenário descrito na seção 5.2.2, de acordo com os parâmetros da Tabela 5.3, para os diferentes valores de razão de perda de pacotes. Podemos observar que para valores de P menores que 4%, o resultado do LowRTT é pouco superior ao alcançado pelo LowRTT com o filtro, sendo 0.5% maior para P de 0.5%, 1% para P de 1% e 2% e 0.5% maior para P de 3%. Porém, a partir de 4% de P, o *Goodput* médio alcançado com o LowRTT+F foi superior ao conseguido pelo LowRTT, ficando 1.5% maior para P de 4%, 2% maior para P de 5%, 4% maior para P de 6% e em torno de 5% maior para P de 7%. A Tabela 6.4 apresenta um resumo do ganho percentual de *Goodput* alcançado pelo LowRTT+F em relação ao LowRTT no segundo cenário.

Tabela 6.4: Resumo comparativo do ganho percentual do *Goodput* alcançado pelo LowRTT+F em relação ao LowRTT no 2º cenário

Resumo comparativo (LowRTT e LowRTT+F) - 2º cenário								
P	0.5%	1%	2%	3%	4%	5%	6%	7%
Goodput (Ganho Percentual)	-0.5%	-1%	-1%	-0.5%	1.5%	2%	4%	5%

Novamente, o resultado apresentado neste cenário, difere do alcançado no primeiro cenário no que se refere ao limiar de razão de perda de pacotes. Isso evidencia que o valor

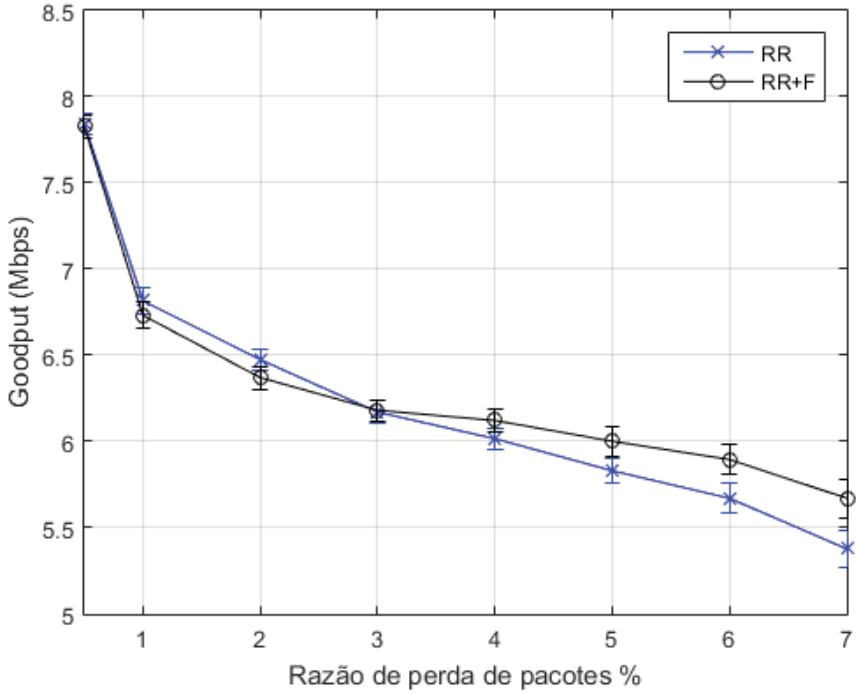


Figura 6.7: *Goodput* médio alcançado pelo RR e pelo RR+F no 2º cenário

de limiar varia de acordo com as características dos caminhos disponíveis para a transmissão MPTCP.

O esperado era que os resultados obtidos com o método de filtragem fossem semelhantes aos alcançados sem o método para os valores de razão de perda de pacotes inferiores ao limiar, porém a ocorrência de falsos positivos decorrentes do erro da estimativa levam aos resultados observados nos gráficos das simulações.

6.3 Resumo

Este capítulo apresentou os resultados dos experimentos realizados através de simulações sobre um ambiente de rede heterogênea com o protocolo MPTCP, com o objetivo de avaliar o impacto de um caminho degradado no desempenho da transmissão e de avaliar o desempenho do método de filtragem de um caminho degradado proposto. Os resultados mostram que, quando o caminho degradado passa de um limiar de razão de perda de pacotes específico para aquele cenário, é melhor quanto ao desempenho utilizar somente um único fluxo TCP pelo melhor caminho disponível. Além disso, os resultados obtidos nas simulações com os algoritmos de escalonamento RR e LowRTT, mostraram que, em situações de alta degradação decorrente da perda de pacotes, o uso do método de filtragem do caminho degradado proposto apresenta um ganho de desempenho na transmissão MPTCP.

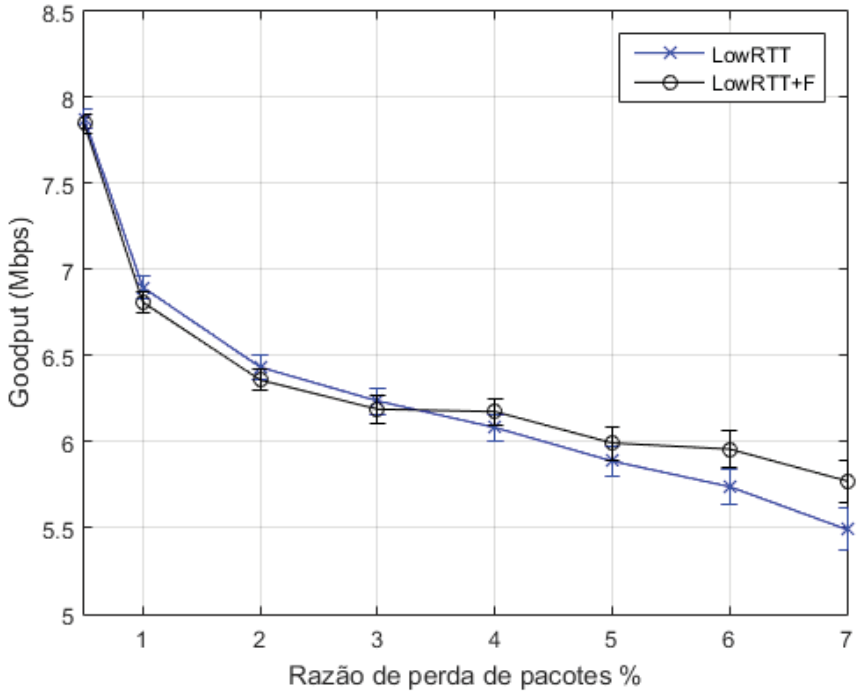


Figura 6.8: *Goodput* médio alcançado pelo LowRTT e pelo LowRTT+F no 2º cenário

Capítulo 7

Conclusão

Este trabalho foi desenvolvido com o objetivo de avaliar soluções para o problema do baixo desempenho de protocolos multi-caminhos em ambiente de rede heterogênea, como redes sem fio, degradado pela perda de pacotes. Inicialmente, foi realizada uma breve revisão sobre protocolos de transporte para sistemas multi-abrigados e detalhada a arquitetura do MPTCP com maior foco no componente escalonador padrão e nos algoritmos escalonadores *Round Robin* e *LowRTT* para o MPTCP, visando com isso situar o leitor no enredo do estudo. Em seguida, um levantamento do estado da arte sobre métodos para mitigação dos problemas relativos aos protocolos de sistemas multi-abrigados, com foco nas estratégias de escalonamento, foi apresentado e discutido, elencando-se os algoritmos e os dividindo de acordo com as suas estratégias de escalonamento.

Além disso, foi analisado o impacto de um caminho degradado no desempenho da transmissão MPTCP e proposto um método de filtragem do caminho mais degradado do conjunto de subfluxos disponíveis para escalonamento com o objetivo de mitigação do impacto no desempenho da transmissão MPTCP. Foram elencadas limitações da proposta e feita uma comparação deste trabalho com a coletânea do capítulo de trabalhos relacionados.

O método de filtragem do caminho degradado foi detalhado e avaliado, assim como as proposições que levaram a esse estudo. Posteriormente, desenvolveu-se o planejamento dos experimentos para avaliação, definindo-se a metodologia e a forma de implementação. Por fim, os resultados alcançados foram apresentados e discutidos. Os efeitos e consequências da solução foi analisada pela comparação dos desempenhos atingidos pelos algoritmos de escalonamento RR e LowRTT, com e sem o método de filtragem, no cenário de rede heterogênea adotado.

Por fim, verificou-se que o problema do baixo desempenho observado no MPTCP, quando um dos caminhos está degradado pela perda de pacotes, possui relação no impacto que esse caminho tem no desempenho dos outros caminhos utilizados para a transmissão, decorrente principalmente do reordenamento dos pacotes no *buffer* de recebimento, que obriga os outros subfluxos a esperarem para enviar mais segmentos ao destino. O uso do método de filtragem dinâmica do caminho degradado se mostrou capaz de mitigar esse impacto do caminho degradado na transmissão MPTCP, utilizando o pior caminho somente quando o desempenho da transmissão MPTCP é superior ao estimado para uma transmissão TCP pelo melhor caminho disponível.

Como trabalho futuro, pode-se sugerir a adição de tráfego de fundo, podendo ser tráfego TCP concorrente, para validação e comparação do desempenho dos algoritmos RR e LowRTT, com e sem o método de filtragem proposto, em um enlace compartilhado com outros fluxos de dados. Outra sugestão é o ajuste da estimativa da taxa de transferência do MPTCP e do TCP, com o objetivo de mitigar os erros de estimativa, fazendo o resultado alcançado pelo algoritmo

escalonador com o método de filtragem ser igual ao do algoritmo sem o método para valores de razão de perda de pacotes inferiores ao limiar.

Referências

[Antunes et al., 2015] Antunes, P. M., de Souza Miguel, G., Pedroso, C. M. e Ribeiro, E. P. (2015). On the stability of delay-centric multihomed transmission. Em *XXXIII Simpósio Brasileiro de Telecomunicações (SBrT'2015)*, página 4.

[Barré et al., 2011] Barré, S., Paasch, C., Bonaventure, O. et al. (2011). Multipath TCP-guidelines for implementers. *Working Draft, IETF Secretariat, Internet-draft draftbarre-mptcp-impl-00*.

[Bonaventure et al., 2015] Bonaventure, O., De Coninck, Q., Baerts, M., Duchene, F. e Hesmans, B. (2015). Improving multipath TCP backup subflows. *IETF, Internet Draft draft-bonaventure-mptcpbackup-00*.

[Braden, 1989] Braden, R. (1989). Requirements for internet hosts-communication layers. RFC 1122, IETF.

[Chebrolu e Rao, 2006] Chebrolu, K. e Rao, R. R. (2006). Bandwidth aggregation for real-time applications in heterogeneous wireless networks. *IEEE Transactions on Mobile Computing*, 5(4):388–403.

[Chen et al., 2014] Chen, R., Wang, W.-N., Zhu, J.-L. e Wang, B. (2014). A round-trip-time based concurrent transmission scheduling for MPTCP. Em *Wireless Communications and Signal Processing (WCSP), 2014 Sixth International Conference on*, páginas 1–5. IEEE.

[Cisco, 2013] Cisco (2013). MPTCP and product support overview. https://goo.gl/c9cuwV. Acessado em 23/11/2017.

[Coudron e Secci, 2017] Coudron, M. e Secci, S. (2017). An implementation of multipath TCP in ns3. *Computer Networks*, 116:1–11.

[Felix et al., 2016] Felix, B., Santos, A. e Nogueira, M. (2016). A new queue discipline for reducing bufferbloat effects in hetnet concurrent multipath transfer. *arXiv preprint arXiv:1609.09314*.

[Ferlin et al., 2016] Ferlin, S., Alay, Ö., Mehani, O. e Boreli, R. (2016). BLEST: Blocking estimation-based MPTCP scheduler for heterogeneous networks. Em *IFIP Networking Conference (IFIP Networking) and Workshops, 2016*, páginas 431–439. IEEE.

[Ford et al., 2011] Ford, A., Raiciu, C., Handley, M., Barre, S. e Iyengar, J. (2011). Architectural guidelines for multipath TCP development. RFC 6182, IETF.

[Ford et al., 2013] Ford, A., Raiciu, C., Handley, M. e Bonaventure, O. (2013). TCP extensions for multipath operation with multiple addresses. RFC 6824, IETF.

[Ford e Iyengar, 2008] Ford, B. e Iyengar, J. R. (2008). Breaking up the transport logjam. Em *HotNets*, páginas 85–90.

[Garcia-Saavedra et al., 2017] 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.

[Gavriloff, 2009] Gavriloff, I. (2009). Análise de aspectos envolvidos no mecanismo de seleção de caminho baseado em atraso para sistemas multiabrigados utilizando SCTP. Dissertação de Mestrado, Pós-Graduação em Engenharia Elétrica - Universidade Federal do Paraná, Curitiba - PR.

[Henderson et al., 2012] Henderson, T., Floyd, S., Gurtov, A. e Nishida, Y. (2012). The NewReno Modification to TCP's Fast Recovery Algorithm. Proposed Standard. RFC 6582, RFC Editor.

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

[Hwang e Yoo, 2015] Hwang, J. e Yoo, J. (2015). Packet scheduling for multipath TCP. Em *Ubiquitous and Future Networks (ICUFN), 2015 Seventh International Conference on*, páginas 177–179. IEEE.

[Iyengar et al., 2006] 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.

[Karn et al., 2004] Karn, P., Bormann, P., Fairhurst, P., Grossman, P., Ludwig, P., Mahdavi, J., Montenegro, G., Touch, J. e Wood, L. (2004). Advice for Internet Subnetwork Designers. Best Current Practice. RFC 3819, RFC Editor.

[Ke et al., 2016] 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 *Communications (APCC), 2016 22nd Asia-Pacific Conference on*, páginas 248–253. IEEE.

[Kelly et al., 2004] Kelly, A., Muntean, G., Perry, P. e Murphy, J. (2004). Delay-centric handover in SCTP over wlan. *Transactions on Automatic Control and Computer Science*, 49(63):1–6.

[Kelly, 2003] Kelly, T. (2003). Scalable TCP: Improving performance in highspeed wide area networks. *ACM SIGCOMM computer communication Review*, 33(2):83–91.

[Kheirkhah et al., 2014] Kheirkhah, M., Wakeman, I. e Parisis, G. (2014). Multipath-TCP in ns-3. Em *2014 Workshop on ns-3 (WNS3)*.

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

[Le e Bui, 2015] Le, T.-A. e Bui, L. X. (2015). Forward delay-based packet scheduling algorithm for multipath TCP. *Mobile Networks and Applications*, páginas 1–9.

[Li et al., 2015] Li, L., Hu, N., Liu, K., Fu, B., Chen, M. e Zhang, L. (2015). AMTCP: an adaptive multi-path transmission control protocol. Em *Proceedings of the 12th ACM International Conference on Computing Frontiers*, página 29. ACM.

[Li et al., 2016] Li, M., Lukyanenko, A., Ou, Z., Ylä-Jääski, A., Tarkoma, S., Coudron, M. e Secci, S. (2016). Multipath transmission for the internet: A survey. *IEEE Communications Surveys and Tutorials*, 18(4):2887–2925.

[Lim et al., 2017] 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 2017 ACM SIGMETRICS/International Conference on Measurement and Modeling of Computer Systems*, páginas 33–34. ACM.

[Liu et al., 2014] Liu, K., Huang, Q., Liu, Y., Fan, Y. e Xie, G. (2014). A novel trunk scheduling for concurrent multipath transfer in heterogeneous wireless networks. Em *Wireless Communications, Networking and Mobile Computing (WiCOM 2014), 10th International Conference on*, páginas 398–403. IET.

[NS-3 Consortium, 2011] NS-3 Consortium (2011). NS-3 a discrete-event network simulator for internet system. https://www.nsnam.org. Acessado em 23/11/2017.

[Mehani et al., 2015] Mehani, O., Holz, R., Ferlin, S. e Boreli, R. (2015). An early look at multipath tcp deployment in the wild. Em *Proceedings of the 6th International Workshop on Hot Topics in Planet-Scale Measurement*, páginas 7–12. ACM.

[Mirani et al., 2010] Mirani, F. H., Boukhatem, N. e Tran, M. A. (2010). A data-scheduling mechanism for multi-homed mobile terminals with disparate link latencies. Em *Vehicular Technology Conference Fall (VTC 2010-Fall), 2010 IEEE 72nd*, páginas 1–5. IEEE.

[Ni et al., 2015] Ni, D., Xue, K., Hong, P., Zhang, H. e Lu, H. (2015). OCPS: Offset compensation based packet scheduling mechanism for multipath TCP. Em *Communications (ICC), 2015 IEEE International Conference on*, páginas 6187–6192. IEEE.

[Nikravesh et al., 2016] Nikravesh, A., Guo, Y., Qian, F., Mao, Z. M. e Sen, S. (2016). An in-depth understanding of multipath TCP on mobile devices: Measurement and system design. Em *Proceedings of the 22nd Annual International Conference on Mobile Computing and Networking*, páginas 189–201. ACM.

[Oh e Lee, 2015] Oh, B.-H. e Lee, J. (2015). Constraint-based proactive scheduling for MPTCP in wireless networks. *Computer Networks*, 91:548–563.

[Paasch et al., 2013] Paasch, C., Barré, S. et al. (2013). Multipath TCP in the linux kernel. https://multipath-tcp.org/. Acessado em 23/11/2017.

[Paasch e Bonaventure, 2014] Paasch, C. e Bonaventure, O. (2014). Multipath TCP. *Communications of the ACM*, 57(4):51–57.

[Paasch et al., 2014] 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.

[Paxson et al., 2011] Paxson, V., Allman, M., Chu, J. e Sargent, M. (2011). Computing TCP's retransmission timer. RFC 6298, IETF.

[Postel, 1980] Postel, J. (1980). User Datagram Protocol. Internet Standard. RFC 768, RFC Editor.

[Postel, 1981] Postel, J. (1981). Transmission Control Protocol. Internet Standard. RFC 793, RFC Editor.

[Raiciu et al., 2011] Raiciu, C., Handley, M. e Wischik, D. (2011). Coupled congestion control for multipath transport protocols. RFC 6356, IETF.

[Sarwar et al., 2013] 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.

[Scharf e Ford, 2013] Scharf, M. e Ford, A. (2013). Multipath TCP (MPTCP) application interface considerations. RFC 6897, IETF.

[Singh et al., 2012] 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 et al., 2013] Singh, A., Xiang, M., Konsgen, A., Goerg, C. e Zaki, Y. (2013). Enhancing fairness and congestion control in multipath TCP. Em *Wireless and Mobile Networking Conference (WMNC), 2013 6th Joint IFIP*, páginas 1–8. IEEE.

[Stewart et al., 2000] Stewart, R., Xie, Q., Morneault, K., Sharp, C., Schwarzbauer, H., Taylor, T., Rytina, I., Kalia, M., Zhang, L. e Paxson, V. (2000). Stream control transmission protocol. Proposed standard. RFC 2960, RFC Editor.

[Torres, 2014] Torres, A. J. F. (2014). Método para melhoria da qualidade na transmissão de vídeos sobre o protocolo SCTP. Dissertação de Mestrado, Pós-Graduação em Engenharia Elétrica - Universidade Federal do Paraná, Curitiba - PR.

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

[Zeng et al., 2017] Zeng, J., Ke, F., Zuo, Y., Liu, Q., Huang, M. e Cao, Y. (2017). Multi-attribute aware path selection approach for efficient MPTCP-based data delivery. *J. Internet Serv. Inf. Secur.*, 7(1):28–39.