Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Análise Comparativa do Algoritmo TCP BBR: Uma Implementação em Espaço de Usuário e Avaliação do Desempenho em Redes de Computadores

Brenno Pereira Cordeiro
Ítalo Eduardo D. Frota

Monografia apresentada como requisito parcial
para conclusão do Bacharelado em Ciência da Computação

Orientador
Prof. Dr. Marcelo Antonio Marotta

Brasília
2023

Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Análise Comparativa do Algoritmo TCP BBR: Uma Implementação em Espaço de Usuário e Avaliação do Desempenho em Redes de Computadores

Brenno Pereira Cordeiro
Ítalo Eduardo D. Frota

Monografia apresentada como requisito parcial
para conclusão do Bacharelado em Ciência da Computação

Prof. Dr. Marcelo Antonio Marotta (Orientador)
CIC/UnB

Prof. Dr. Donald Knuth
Stanford University

Dr. Leslie Lamport
Microsoft Research

Prof. Dr. Marcelo Grandi Mandelli
Coordenador do Bacharelado em Ciência da Computação

Brasília, 08 de fevereiro de 2023

Dedicatória

Dedicamos este trabalho as nossas famílias e amigos, que sempre acreditaram em nós e nos proporcionaram apoio incondicional ao longo de toda a graduação. Sem vocês, não teríamos chegado até aqui.

If I can just, keep on keeping on... I will arrive at my destination. Some day, some way.

Tinashe Kachingwe, "Nightride", 2016.

Agradecimentos

Agradecemos ao nosso orientador, o professor Dr. Marcelo Antonio Marotta (CIC/UnB), cuja presença, dedicação e paciência foram essenciais para o sucesso deste trabalho. Agradecemos ao corpo docente do Departamento de Ciência da Computação da Universidade de Brasília por todo o conhecimento compartilhado, cada professor nos proporcionou valiosas lições que contribuíram imensamente para a nossa formação através de desafios que estimularam nosso pensamento crítico.

Agradecemos ao projeto de pesquisa PROFISSA, fomentado pela FAPESP. O apoio concedido pelos integrantes foi fundamental para a realização dos experimentos aqui descritos. Agradecemos à Fundação Universidade de Brasília, por proporcionar um ambiente plural e propício à pesquisa e desenvolvimento de qualidade. A oportunidade de estudar em uma instituição reconhecida pela valorização a excelência acadêmica tem sido essencial para o nosso desenvolvimento pessoal e profissional.

Não podemos deixar de agradecer à Empresa Júnior de Computação da Universidade de Brasília - CJR, responsável por transformar a nossa vivência universitária e profissional através de projetos de alto impacto e conexões inestimáveis que levaremos por toda a nossa vida.

Por fim, agradecemos profundamente um ao outro, pela experiência de compartilhar esta jornada desafiadora e enriquecedora com companheirismo, bom humor e apoio mútuo.

O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES), por meio do Acesso ao Portal de Periódicos.

Resumo

Os algoritmos de controle de congestionamento cumprem uma função fundamental em redes TCP ao evitar a saturação de um caminho de rede, mantendo a taxa de transmissão regulada. Diversos algoritmos já estabelecidos estão presentes nas pilhas TCP/IP, como o CUBIC e o Reno. No entanto, esses algoritmos partem da suposição de que perda de segmentos significa congestionamento, o que nem sempre é verdade. Dado esse contexto, surge a necessidade de algoritmos mais robustos, como o BBR (Bandwidth Bottleneck and Round-trip propagation time), que utiliza de mecanismos de sondagem e estimativa para obter sinais mais precisos de congestionamento. Focando principalmente na versão 1 do BBR, o estudo propõe e executa uma implementação em espaço de usuário do algoritmo, discutindo seu desempenho em comparação com os outros dois algoritmos. O BBR é analisado em detalhes, explicando seus parâmetros, mecanismos e a eficiência em situações de congestionamento de rede. Um protótipo funcional é desenvolvido seguindo as especificações do BBR e testado em diversas condições de rede. O desempenho do BBR é avaliado através de medições de *goodput*, utilizando a ferramenta *iperf3* em condições simuladas com a ferramenta *mahimahi*. Os resultados indicam que o BBR geralmente apresenta desempenho superior, particularmente em termos de vazão média geral, embora seu desempenho varie com base nas características da rede. Futuramente, testes mais próximos a cargas de trabalho reais podem ser executados, e a implementação pode evoluir para abranger a versão 2 do algoritmo BBR.

Palavras-chave: Controle de Congestionamento, TCP BBR, espaço de usuário

Abstract

Congestion control algorithms fulfill a fundamental role within TCP networks by preventing network path saturation and maintaining regulated transmission rates. Various well-established algorithms, such as CUBIC and Reno, are integrated into TCP/IP stacks. Nevertheless, these algorithms operate under the assumption that segment loss signifies congestion, which does not always hold true. Consequently, a demand arises for more resilient algorithms, exemplified by BBR (Bandwidth Bottleneck and Round-trip propagation time). BBR employs probing and estimation mechanisms to acquire more accurate congestion indications. With a primary focus on BBR version 1, this study proposes and executes a user-space implementation of the algorithm, elucidating its performance relative to the aforementioned alternatives. A comprehensive analysis of BBR is conducted, encompassing its parameters, mechanisms, and effectiveness in network congestion scenarios. A functional prototype, adhering to BBR specifications, is developed and subjected to testing under diverse network conditions. The evaluation of BBR's performance centers around measurements of goodput, utilizing the iperf3 tool in simulated environments facilitated by the mahimahi tool. The findings consistently reveal BBR's tendency to outperform other algorithms, particularly in terms of overall average throughput, although its performance is subject to variability contingent upon network characteristics. Future research endeavors may encompass tests that more closely emulate real-world workloads, while also exploring the potential of accommodating BBR version 2 within the implementation.

Keywords: Congestion Control, TCP BBR, User space

Sumário

Lista de Figuras

Capítulo 1

Introdução

O protocolo TCP (Transmission Control Protocol) é um dos protocolos mais utilizados em redes de computadores, seu principal objetivo é a entrega confiável de dados através da conexão entre dispositivos. O TCP procura manter a integridade dos dados transportados utilizando detecção de perdas e erros para realizar a retransmissão de pacotes quando necessário, mantendo um fluxo contínuo de transporte de pacotes. Existem diversas variações deste protocolo, popularmente conhecidas como os *flavours* do TCP, implementadas para atender necessidades e requisitos específicos em diferentes tipos de rede.

A escolha de um *flavour* do TCP depende de diversos fatores dentro do ambiente de rede, requisitos da aplicação e preferências dos administradores de rede, e é influenciada por aspectos como a largura de banda disponível, latência de rede e as características de congestionamento. A crescente demanda por comunicação de dados em redes TCP impulsionou o desenvolvimento de um cenário mais flexível de opções para atender os mais diversos tipos de aplicações, assim, surgiram novos algoritmos de controle de congestionamento com diferentes características visando maior eficiência de uso. A utilização de um algoritmo de controle de congestionamento adequado é crucial para garantir um desempenho otimizado da rede, a fim de evitar perdas de pacotes, degradação da vazão total e atrasos em excesso.

1.1 TCP BBR

O controle de congestionamento em redes de computadores apresenta um desafio significativo, especialmente quando o protocolo TCP em sua forma padrão mostra-se inadequado para lidar com a complexidade das redes congestionadas. Nesse contexto, o TCP BBR (*Bottleneck Bandwidth and Round-trip propagation time*), desenvolvido pela Google, destaca-se como um algoritmo eficiente, oferecendo desempenho de alta velocidade

e estabilidade em redes sujeitas a mudanças frequentes na largura de banda disponível, como redes sem fio ou redes de nuvem.

O TCP BBR foi projetado visando o aproveitamento da largura de banda dinâmica para a rápida adaptação às mudanças ocorridas durante a transmissão de pacotes. Através da medição precisa de largura de banda e tempo de resposta (RTT), o protocolo é capaz de diminuir a latência e o *jitter*, garantindo a qualidade de comunicação da rede com entregas de pacotes de forma confiável. O algoritmo se baseia em sinais recentes e utiliza uma máquina de estados para regular o fluxo de dados de acordo com as condições da rede.

A perda de pacotes é esperada em redes de computadores, mas suas causas podem ser variadas. A depender do ambiente de rede analisado, podem não haver soluções avançadas o suficiente para lidar de forma eficiente com a complexidade do congestionamento, por exemplo, como levantado e proposto no trabalho de Gomez *et al.* [1], as implementações atuais de detecção rápida de perdas em redes programáveis definidas em P4, uma linguagem de programação voltada para redes definidas por *software*, não inferem a causa raiz das perdas de pacote, pois, somente reagem ao congestionamento. Novos trabalhos devem aprimorar o algoritmo de inferência de causa, e a utilização de esquemas que apliquem algoritmos de controle de congestionamento não-baseados em perdas, como o TCP BBR. Tal ponto demonstra a relevância e importância do algoritmo como uma opção para a implementação de novos ambientes de rede.

1.2 TCP NewReno

Por outro lado, o TCP NewReno, um dos algoritmos tradicionais de controle de congestionamento, segue uma abordagem baseada em perda de pacotes. Quando o receptor recebe um pacote fora de ordem, ele envia um reconhecimento para informar ao emissor sobre a chegada desse pacote. Se o emissor receber três reconhecimentos duplicados consecutivos para um mesmo pacote, isso indica que os pacotes subsequentes foram perdidos, pois o receptor já recebeu cópias suficientes do mesmo.

Após a detecção de perda, independentemente do motivo, o protocolo reduz o tamanho da janela de congestionamento para controlar o envio de novos pacotes. Sendo assim, ocorre a diminuição da taxa de envio de dados, possibilitando a recuperação da rede através do processo de "retransmissão rápida", onde apenas os pacotes perdidos são retransmitidos, sem esperar pelo tempo de expiração do temporizador TCP. Essa abordagem permite que os pacotes perdidos sejam reintroduzidos na transmissão de forma mais rápida, contribuindo para a recuperação do congestionamento e a garantia da confiabilidade da comunicação.

Embora o TCP NewReno seja amplamente utilizado e tenha mostrado resultados satisfatórios em muitos cenários, em redes com características específicas, como alta latência e largura de banda variável, ele pode não se adaptar de forma eficiente, resultando em subutilização da capacidade da rede.

1.3 TCP CUBIC

Outro algoritmo de controle de congestionamento amplamente utilizado é o *TCP CUBIC*, inicialmente introduzido como uma alternativa ao *TCP NewReno* e posteriormente adotado como o algoritmo padrão no *kernel* do Linux a partir da versão 2.6.19 [2]. O CUBIC é projeto para otimizar o desempenho em redes com alta largura de banda e latência, utilizando uma função cúbica para modelar a taxa de congestionamento, o que permite um aumento mais expressivo da taxa de envio de dados. O algoritmo opera em duas fases: a de crescimento lento e a de crescimento cúbico. Ao decorrer da fase de crescimento lento, a taxa de envio aumenta de forma linear à medida que a transmissão dos pacotes ocorre e estes são recebidos com sucesso. Ao atingir a taxa de congestionamento, a conexão entra na fase de crescimento cúbico, onde a taxa de envio é aumentada de forma cúbica em relação ao tempo.

Embora o *TCP CUBIC* ofereça bom desempenho em ambientes com largura de banda elevada, ele pode não ser adequado para todos os cenários. Em redes com largura de banda baixa e latência reduzida, o algoritmo pode resultar em congestionamentos excessivos que diminuirão o desempenho. Nestes casos, outros *flavors* do TCP podem ser mais adequados.

1.4 Contribuições

A motivação para comparar o TCP BBR com NewReno e CUBIC surgiu da relevância desses algoritmos no contexto de controle de congestionamento em redes de computadores e, especialmente, no contexto da implementação padrão do *kernel* do Linux. Ao mencionar que o CUBIC é a implementação padrão no desse sistema operacional, fica explícito que esse algoritmo é amplamente utilizado em muitos ambientes. No entanto, o fato de ser a implementação padrão não implica necessariamente que seja a melhor opção em todos os cenários ou que não existam alternativas com desempenho superior em certos contextos.

Dessa forma, a motivação para comparar o TCP BBR com NewReno e CUBIC está em investigar e avaliar o desempenho relativo desses algoritmos em diferentes cenários e condições de rede. O TCP BBR tem recebido crescente atenção e reconhecimento devido à sua abordagem inovadora, que visa maximizar o uso da largura de banda disponível e

minimizar os atrasos em redes de alta velocidade. Por outro lado, o CUBIC e o NewReno são algoritmos mais estabelecidos e amplamente utilizados, cada um com suas próprias características e desempenho.

O controle de congestionamento em redes TCP é um desafio significativo na área de redes de computadores, especialmente em ambientes com alta latência e largura de banda variável. Nesse contexto, o algoritmo NewReno enfrenta dificuldades para se adaptar efetivamente às condições da rede. Analogamente, embora o TCP CUBIC seja amplamente utilizado e tenha mostrado resultados positivos em vários casos, sua implementação no *kernel* do sistema operacional pode apresentar desafios em termos de complexidade e segurança.

Por outro lado, o algoritmo BBR despertou um grande interesse tanto na comunidade acadêmica quanto na indústria, devido à sua abordagem inovadora e adaptativa para melhorar o desempenho das redes TCP. Diante disso, adotamos uma abordagem de implementação em espaço de usuário, buscando contornar esses desafios e oferecer uma alternativa mais acessível e adaptável. Todavia, as implementações e aplicações do algoritmo TCP BBR costumam ser realizadas diretamente no *kernel* do sistema operacional. O que pode gerar desafios em termos de complexidade de desenvolvimento, além de implicações de segurança. Contornando essas dificuldades, proporemos a implementação em espaço de usuário, comparando-a às implementações em *kernel* dos demais algoritmos.

Ao realizar a comparação entre o TCP CUBIC, o TCP NewReno e nossa implementação do BBR em espaço de usuário, pretendemos obter percepções valiosas sobre o desempenho relativo desses algoritmos em diferentes cenários de rede. Esses resultados podem fornecer uma compreensão mais abrangente sobre as vantagens e limitações de cada abordagem, contribuindo para a melhoria do controle de congestionamento em redes TCP e abrindo possibilidades para aplicações futuras em diferentes sistemas e contextos.

No próximo capítulo serão apresentados fundamentos teóricos importantes para a compreensão dos demais componentes deste trabalho, evidenciando o funcionamento dos protocolos e os resultados esperados de suas implementações; no Capítulo 3 são tratados os trabalhos relacionados e suas respectivas contribuições; em seguida, no Capítulo 4 é explicitada a metodologia dos experimentos realizados; o Capítulo 5 conta com os resultados e a discussão a respeito dos experimentos; o Capítulo 6 reúne as lições aprendidas durante o desenvolvimento deste trabalho; finalizando, o Capítulo 7 concretiza as conclusões e apresenta sugestões para os trabalhos futuros.

Capítulo 2

Fundamentação Teórica

Este capítulo tem como objetivo apresentar uma análise abrangente dos conceitos e algoritmos fundamentais relacionados ao controle de congestionamento em redes TCP. O gerenciamento eficiente do tráfego de dados em redes de computadores é um desafio crítico que requer uma compreensão aprofundada das abordagens teóricas e práticas desenvolvidas para lidar com a complexidade da congestão. Nesse contexto, este capítulo proporcionará uma exploração detalhada do TCP BBR, um algoritmo inovador de controle de congestionamento, juntamente com os algoritmos tradicionais amplamente utilizados, TCP CUBIC e TCP NewReno. Além disso, será abordada a implementação em espaço de usuário do TCP BBR, baseada na biblioteca smoltcp [3], oferecendo uma perspectiva técnica adicional e uma base sólida para a comparação e avaliação experimental desses algoritmos.

2.1 Proposta

Dado a problemática de controle de congestionamento em redes TCP, a proposta consiste em desenvolver uma implementação em espaço de usuário do algoritmo de controle BBR. Para avaliar se a implementação gera valor, será comparada com os algoritmos de controle fornecidos pelo TCP CUBIC e TCP NewReno, observando a vazão total no envio de dados.

A implementação em espaço de usuário implica que o programa gerado será executado como uma aplicação comum ao usuário, sem envolver programação do kernel. Isso facilita o desenvolvimento, evitando as complexidades e implicações de segurança de alterar o kernel do sistema operacional.

2.2 TCP BBR

A versão desenvolvida para o BBR é a versão 1, definida no rascunho [4]. A decisão de implementação da versão 1 envolve o aumento de complexidade apresentado nas versão 2 de algoritmo. Isso nos permite uma abordagem incremental, dando liberdade para adicionar as mudanças necessárias para chegar a versão 2.

Como observado, o TCP BBR é um algoritmo de controle de congestionamento baseado em um modelo explícito do caminho de rede sobre o qual um fluxo de transporte de dados trafega. O modelo inclui dois parâmetros de estimativa: *Max Bandwidth* (BBR.BtlBw na documentação) e *Min RTT* (BBR.RTprop na documentação).

1. *Max Bandwidth*: é uma estimativa do gargalo de largura de banda disponível para transmissão de dados. É possível que as transmissões de uma conexão sejam limitadas por um *link* mais lento que causa um gargalo, a taxa de entrega deste gargalo determina a taxa máxima de entregas de dados da respectiva conexão. O algoritmo busca um equilíbrio de taxa, onde a taxa de entrega dos pacotes do fluxo no gargalo corresponda à *Max Bandwidth*. A estimativa dessa variável é feita de forma contínua, recolhendo amostras recentes da taxa de entrega observadas pela conexão durante um determinado número de viagens de ida e volta.

   As amostras de taxa de entrega são filtradas com a utilização de um filtro de valor máximo, uma vez que as amostras de taxa de entrega podem ser menores que a largura de banda disponível. O comprimento da janela do filtro de valor máximo do *Max Bandwidth* deve ser longo o suficiente para cobrir um ciclo completo de ganho do *ProbeBW* (utilizado na sonda de largura de banda) e garantir amostras de taxa de entrega que resultem de dados transmitidos com uma taxa de transmissão maior que 1.0. É necessário que o comprimento seja longo o suficiente para lidar com as flutuações de curto prazo na taxa de entrega, mas que seja curto o suficiente para responder com prontidão às reduções reais na largura de banda disponível para o fluxo analisado.

2. *Min RTT*: é uma estimativa do atraso de propagação bidirecional do caminho por onde um fluxo de dados trafega. O atraso determina o tempo mínimo pelo qual a conexão deve manter as transmissões na taxa de *max bandwidth*, logo, define a quantidade mínima de dados em trânsito necessária para que a conexão esteja em completa utilização (*pipe* cheio). O atraso de propagação das viagens de ida e volta pode variar ao longo do tempo, então o BBR faz estimações contínuas do *Min RTT* usando amostras recém coletadas do atraso bidirecional.

Aqui, é utilizado um filtro mínimo para filtrar e descartar as amostras de atraso de ida e volta que estejam acima do atraso de propagação do caminho, o que pode acontecer por variações aleatórias introduzidas pelos processos de transmissão física, estratégias de ACK filas ao longo do caminho da rede e etc. O BBR estima o *Min RTT* utilizando a amostra mínima de atraso de ida.

O *BDP* (*Bandwidth-Delay Product*) é um conceito que combina o *Max Bandwidth* e *Min RTT* em um único valor estimativo da capacidade máxima de fluxo de dados suportada pela rede de maneira eficiente em um determinado período de tempo. O cálculo é obtido através do produto da largura de banda do gargalo com o atraso de propagação do caminho de rede. O ajuste da taxa de envio é crucial para a otimização do desempenho da conexão, e para tal, o BBR busca manter a taxa de entrega dos pacotes próxima ao valor do *BDP*.

Em resumo, o BBR utiliza o *Max Bandwidth* no ajuste da taxa de envio e o *Min RTT* para controlar a quantidade de dados em trânsito na rede. Ambas variáveis são atualizadas de forma contínua com base nas informações coletadas sobre a taxa de entrega e o atraso de propagação de ida e volta, o que permite ao algoritmo garantir a otimização do desempenho da conexão em redes com gargalos. O *BDP* é fundamental nesse contexto, pois considera tanto a largura de banda disponível quanto o atraso de propagação para determinar a taxa de envio ideal e garantir uma utilização eficiente da rede.

2.2.1 Máquina de Estados

O BBR é um algoritmo *stateful* baseado em sinais recentes para a regulagem do fluxo de dados. Para controlar o comportamento do algoritmo de acordo com as condições da rede, é implementada uma máquina de estados que define os diferentes modos de operação, que contam com ações e regras específicas e bem definidas baseadas em um modelo de caminho de rede e seus parâmetros de controle.

O funcionamento da máquina de estados visa garantir os seguintes objetivos:

- Alcançar alta taxa de transferência através da utilização eficiente da largura de banda disponível;
- Alcançar baixa latência, mantendo filas pequenas e limitadas;
- Realizar envios multiplicativamente mais rápidos para sondar a rede em busca de descobrir rapidamente a largura de banda recém-disponível;
- Se necessário, reduzir a taxa de envio para drenar a fila de gargalo e sondar a rede para estimar o atraso bidirecional de propagação da rede;

- Compartilhar a largura de banda com os demais fluxos da forma mais justa possível;
- Gerar amostras aos estimadores de *max bandwith* e *min RTT* para atualizar o modelo.

A máquina de estados do TCP BBR conta com dois principais mecanismos de controle. O primeiro, essencial para a capacidade de aprendizado do BBR, é o "*pacing_gain*", que controla a velocidade de envio de pacotes em relação ao *max bandwidth*. A diminuição do intervalo entre os pacotes e o aumento da quantidade de pacotes em trânsito ocorrem com um *pacing_gain* maior que 1. Quando o valor é menor, aumenta-se o intervalo entre os pacotes e consequentemente pode ser diminuída a quantidade de pacotes em trânsito. O segundo mecanismo é o *congestion window* ("cwnd"), utilizado para reduzir rapidamente a quantidade de pacotes em trânsito para um valor específico, quando necessário. Além disso, são implementados os seguintes observados na Figura 2.1.

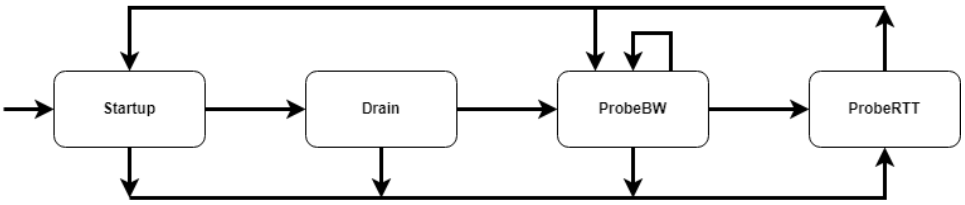

Figura 2.1: Máquina de estados do TCP BBR simplificada

1. *Startup*: O estado inicial. Aqui, o algoritmo inicializa as variáveis e configurações para iniciar o processo de descoberta do gargalo da rede, aumentando rapidamente a taxa de envio de dados para encontrar a capacidade máxima da rede;
2. *Drain*: Neste estado, o BBR drena qualquer fila criada durante o estado *Startup*, mantendo uma taxa de envio menor que 1. O objetivo é estabilizar a largura de banda utilizada, e há a transição para o estado *ProbeBW* quando a quantidade de pacotes em trânsito corresponde à estimativa do BDP;
3. *ProbeBW*: Se trata do estado principal, onde o BBR realiza ciclos de ganho para sondar a largura de banda recém-disponível. A taxa de envio de dados é aumentada para explorar o limite superior da largura de banda, medindo a taxa de entrega dos pacotes. É aplicada uma sequência de valores de *pacing_gain* para manter uma pequena fila e alta utilização da largura de banda medida;
4. *ProbeRTT*: É o estado responsável pela medição do RTT, é ativado quando ocorre uma mudança significativa nesta variável. O algoritmo realiza sondagens para estimar o RTT atual, fundamental para o cálculo do tamanho da janela de congestionamento.

5. *Idle*: Estado de ociosidade, ocorre quando não há tráfego.

A abordagem *stateful* acima permite respostas ágeis e eficientes às mudanças no RTT e na largura de banda da rede, garantindo a otimização do desempenho da comunicação TCP em redes sujeitas à alterações frequentes no fluxo de envio de dados.

2.3 TCP CUBIC

Atualmente, o TCP CUBIC é o algoritmo de controle de congestionamento padrão do *kernel* do Linux. Seu desenvolvimento se deu com o intuito de melhorar o desempenho do protocolo TCP/IP em redes de alta velocidade e grandes latências.

O nome "CUBIC"vem da forma cúbica da função de crescimento aplicada na janela de congestionamento durante a fase de prevenção de congestionamento do TCP. A função cúbica utilizada aumenta gradualmente o tamanho da janela de congestionamento, o que permite um crescimento controlado da taxa de envio. A abordagem utilizada é baseada no modelo de janela deslizante, o que permite que o emissor envie um número limitado de pacotes antes de aguardar os ACKs de confirmação do receptor.

Inicialmente, o tamanho da janela de congestionamento aumenta de forma exponencial, aproveitando ao máximo a capacidade disponível da rede, entretanto, ao atingir o limite de *threshold*, o algoritmo entra na fase de prevenção de congestionamento. Nesta fase, é aplicada a função cúbica no ajuste do tamanho da janela, o cálculo é realizado com base no tempo decorrido desde o início da fase de prevenção e no valor máximo atingido pela janela de congestionamento nesta fase. A fórmula da função cúbica do TCP CUBIC é a seguinte:

$$W(t) = C \cdot (t - K)^3 + W_{\max}$$

Onde:

- $W(t)$ representa o tamanho atual da janela de congestionamento no tempo t.
- C é o coeficiente de crescimento que determina a taxa de aumento da janela de congestionamento.
- K é o tempo em que a janela de congestionamento atingiu o valor máximo ($W_{\max}$) durante a fase de prevenção de congestionamento
- $W_{\max}$ é o valor máximo alcançado pela janela de congestionamento durante a fase de prevenção de congestionamento

O TCP CUBIC ainda utiliza o algoritmo de detecção de perda de pacotes baseado em ACKs duplicados e temporizadores expirados para identificar e reagir a congestionamentos na rede. Perante uma perda de pacote, o algoritmo reduz significantemente o tamanho da janela de congestionamento, dando início a um período de *slow start* para a recuperação gradual da taxa de envio.

Conforme levantado por Ha *et al.* (2008) [2], nos contextos em que o TCP padrão é capaz de utilizar a largura de banda com eficiência satisfatória, o CUBIC não demonstra mudanças significativas. O TCP padrão apresenta boa performance nos seguintes dois tipos de redes:

1. redes com um BDP pequeno;
2. redes com um RTT pequeno, mas não necessariamente um BDP pequeno.

Para manter equidade ao TCP padrão, o CUBIC se comporta da mesma maneira enquanto a função cúbica de crescimento da janela é mais lenta do que a do TCP padrão. A natureza em tempo real do protocolo mantém a taxa de crescimento da janela independente do RTT. Os testes executados confirmam que o CUBIC alcança uma equidade intra-protocolo satisfatória.

2.4 TCP NewReno

Embora o TCP NewReno não seja uma abordagem tão recente quanto o TCP BBR, ainda é amplamente utilizado e possui características que mantém sua relevância. Introduzido em 1990 como uma modificação do algoritmo original do TCP, com o intuito de melhorar o desempenho de rede em cenários de congestionamento mantendo a equidade entre as conexões e a eficiência na utilização da rede. Em adição, há o gerenciamento do equilíbrio da taxa de entrega dos pacotes de diferentes conexões, evitando o excesso de congestionamentos e perdas desnecessárias de dados.

Quando ocorrem perdas de pacotes em uma conexão, o TCP NewReno não retransmite todos os pacotes desde o último confirmado, mas sim, apenas os pacotes nos quais foram identificadas perdas. Este mecanismo é conhecido como "retransmissão seletiva", e reduz a redundância na transmissão, minimizando o impacto do congestionamento na rede.

A estratégia de redução de janela de congestionamento pode ser considerada menos agressiva em comparação com outros algoritmos. Ao invés de reduzir abruptamente a janela de congestionamento, o NewReno realiza uma redução linear, evitando quedas bruscas na taxa de transmissão e colaborando com a recuperação da rede após o congestionamento. Adicionalmente, o algoritmo utiliza mecanismos de controle baseados em

timers e no retorno de ACKs. O ajuste dinâmico do tamanho da janela de congestionamento tem como base a taxa de confirmações recebidas e no tempo de ida e volta da comunicação. Assim, o TCP NewReno se propõe a se adaptar eficientemente às condições da rede, mitigando a possibilidade de subutilização da capacidade disponível.

2.4.1 Máquina de Estados

A máquina de estados do TCP NewReno descreve o comportamento do algoritmo de controle de congestionamento. Seu objetivo é regular o fluxo de dados em uma rede, a fim de evitar congestionamentos e garantir o desempenho eficiente da comunicação. Através de uma visão geral do funcionamento da máquina de estados, os principais mecanismos de controle do NewReno podem ser observados na Figura 2.2.

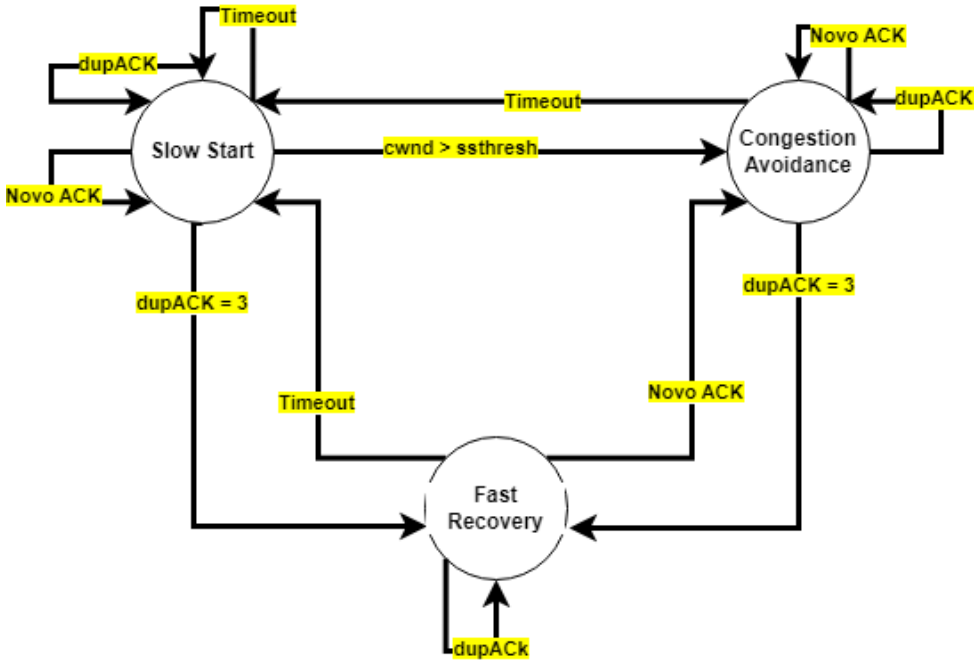

Figura 2.2: Máquina de estados do TCP NewReno simplificada

1. *Slow Start*: Se trata do estado inicial do TCP NewReno e inicia uma conexão de forma conservadora a evitar congestionamentos iniciais. Nele, a janela de congestionamento é inicializada com um valor pequeno é aumentada de maneira exponencial à medida que os pacotes são transmitidos e confirmados. Assim, a quantidade de dados enviados pelo TCP aumenta rapidamente até que ocorram perdas ou sinais de congestionamento.

2. *Congestion Avoidance*: Quando são identificadas perdas de pacotes ou sinais de congestionamento, o NewReno entra neste estado. Aqui, a janela de congestiona-

mento aumenta linearmente, adicionando uma quantidade menor à janela a cada confirmação de pacote.

3. *Fast Recovery*: Caso ocorra uma perda de pacote, o NewReno entra neste estado onde a janela de congestionamento é reduzida pela metade em relação ao valor atual, e realiza a retransmissão seletiva. Durante a recuperação rápida, a janela de congestionamento é aumentada em um pacote para cada confirmação recebida.

Estes mecanismos de controle trabalham em conjunto para ajustar a taxa de envio de pacotes, controlando a quantidade de pacotes em trânsito a fim de evitar congestionamentos na rede. Além destes estados principais, o TCP NewReno também pode contar com outros estados e transições para lidar com eventos como a recepção de novos ACKs, detecção de ACKs duplicados ($dupACK$), eventos de *timeout*, gerenciamento da janela de congestionamento ($cwnd$), limiar de início lento ($sstresh$), entre outros.

Considerando o embasamento teórico apresentado neste capítulo, a seguir serão apresentadas as literaturas relacionadas ao tema, responsáveis por nortear a abordagem metodológica dos experimentos realizados. A revisão bibliográfica abrangerá estudos-chave que se debruçam sobre aspectos relevantes do assunto em questão, fornecendo uma base sólida para a condução dos experimentos e análises subsequentes.

Capítulo 3

Trabalhos Relacionados

Neste capítulo, apresenta-se a literatura relacionada ao tema deste trabalho, que diz respeito à implementação do algoritmo de controle de congestionamento TCP BBR em espaço de usuário e sua comparação com as implementações dos protocolos TCP CUBIC e NewReno do *kernel* do Linux. Serão abordados estudos e pesquisas que exploram os aspectos de eficiência, desempenho e características de diferentes abordagens de implementação para controle de congestionamento.

Durante a fase inicial de pesquisa, enfrentamos desafios significativos no processo de identificação de literatura científica específica que abordasse diretamente a implementação e comparação de protocolos TCP em espaço de usuário e em *kernel*. Embora os estudos dedicados a esta área sejam amplos, a produção recente de trabalhos se concentra em contextos experimentais de domínios e aplicações específicas. No entanto, apesar das limitações na disponibilidade de fontes diretamente relacionadas com o contexto do presente experimento, foram encontrados estudos que exploram aspectos relevantes do TCP BBR em diferentes situações. A análise dessas produções permite uma compreensão mais aprofundada e crítica do tema, dando uma base concreta às análises e conclusões aqui apresentadas.

O artigo *BBR: Congestion-Based Congestion Control* (2016) [5] é o ponto de partida para a compreensão do funcionamento do TCP BBR em sua primeira versão. São discutidos os desafios enfrentados pela internet atual em termos de eficiência na transferência de dados e atrasos, são destacadas as limitações do mecanismo tradicional de controle de congestionamento baseado em perdas utilizado por outros *flavors* do protocolo TCP, que interpreta a perda de pacotes como um indicador de congestionamento, entretanto, é notável que a relação entre estes dois pontos se tornou menos confiável. Assim, problemas como o "*bufferbloat*", em que tamanhos grandes de *buffer* causam taxas altas de atrasos, além da interpretação incorreta de perdas como congestionamento, o que resulta em uma baixa taxa de transferência.

Os testes realizados incluíram comparações de desempenho entre o TCP BBRv1 e o CUBIC em diferentes cenários e configurações de rede, como testes em servidores de vídeo do YouTube, conexões móveis e redes do Google. Em resumo, os testes demonstraram que o BBR superou o CUBIC em termos de *throughput*, redução do RTT e melhorias na qualidade do serviço em diferentes cenários, além de demonstrar eficiência no gerenciamento de filas e congestionamentos, mostrando desempenho geralmente superior ao do CUBIC.

Edwards *et al.* (1995) [6] já demonstraram que protocolos desenvolvidos e executados em espaço de usuário oferecem níveis de desempenho aceitáveis, contanto que sejam capazes de explorar um conjunto apropriado de interfaces de baixo nível. Embora a implementação do TCP em espaço de usuário feita pelos autores opere a 80% da implementação em *kernel*, ficou claro que protocolos complexos de rede poderiam ser movidos para este contexto mesmo com as dificuldades em torno da divisão aceitável entre funções do espaço de usuário e do espaço do *kernel*. Para fornecer uma implementação completa e robusta, adotaram uma arquitetura multi-processual e, por consequência, o protocolo realizava constantes trocas de contexto, o que reduzia o desempenho e adicionava gasto de tempo e recursos com o controle de concorrência. Por fim, os autores esperavam investigar técnicas para fornecer uma implementação completa do TCP em um único processo sem causar comprometimentos à robustez.

Lueke (2019) [7] explora a transferência da pilha TCP/IP para o espaço de usuário, visando maior segurança em relação à memória. Em adição, apresenta a biblioteca *usnet_sockets* baseada na biblioteca *smoltcp* [3], a mesma base da implementação do presente trabalho. São levantados aspectos a respeito do funcionamento do *smoltcp* e suas aplicações em sistemas operacionais como o *RedoxOS* [8], abrindo espaço para as discussões sobre serviços de rede seguros para memória por meio de *switches* de rede em espaço de usuário.

A compreensão da literatura relacionada ao tema é fundamental para embasar a abordagem metodológica adotada neste trabalho. Embora a disponibilidade de fontes específicas sobre a implementação e comparação de protocolos TCP em espaço de usuário e em *kernel* seja relativamente escassa, os estudos encontrados permitiram explorar aspectos relevantes do TCP BBR e protocolos similares. Esta análise crítica e aprofundada fornece uma base sólida para as avaliações e conclusões apresentadas. Além disso, a revisão da literatura ressaltou a importância de compreender as limitações e desafios inerentes à implementação de protocolos complexos de rede em espaço de usuário, incluindo o compromisso entre funções do espaço de usuário e do *kernel*, o desempenho e a segurança perante aos recursos utilizados.

No capítulo seguinte é descrita a abordagem metodológica aplicada neste trabalho.

Apresentaremos os pontos de partida e as características das ferramentas utilizadas nos experimentos, adicionalmente, caracterizamos as análises realizadas em conjunto com os resultados esperados com base no embasamento teórico levantado.

Capítulo 4

Metodologia

A fim de realizar um análise aprofundada da aplicação do algoritmo BBR e da nossa implementação, foi coordenada uma série de testes para validar a nossa problemática. A principal tese que orienta esta pesquisa é a hipótese de que o BBR apresenta um desempenho superior, especialmente em termos de produção de um *goodput* mais elevado ao transmitir dados reais. Essa proposição deriva do próprio projeto do algoritmo BBR, que tem como objetivo otimizar a utilização da largura de banda, sem reagir diretamente à perda de pacotes, diferentemente dos algoritmos CUBIC e NewReno, que dependem da perda como o principal indicador de congestionamento.

Nas seções seguintes, nós delineamos nossa abordagem metodológica, ilustrando como desenhamos e conduzimos os experimentos para avaliar e comparar esses algoritmos. Para obter uma resposta que valida ou desafia nossa tese, buscamos experimentos que envolvem o envie de quantidades substanciais de dados, representativos de cenários de trabalho reais. No processo, utilizamos uma combinação de técnicas de simulação em ambiente controlado e análise de registros de rede reais, permitindo uma análise mais abrangente e abordando diferentes perspectivas do desempenho dos algoritmos de controle de congestionamento.

4.1 Protótipo

Aqui delineamos o processo de projeto e execução do nosso protótipo funcional, bem como o funcionamento de alguns dos mecanismo, que vão fundamentar os resultados experimentais. Para garantir o funcionamento e a confiança do protótipo, seguimos as especificações definidas em duas RFCs (*Request for Comments*). As RFCs são documentos formais que descrevem padrões, protocolos e diretrizes para diversas áreas de tecnologia.

A primeira RFC utilizada serve como referência fundamental para amostragem de dados e estimação de taxa de entrega, que são informações necessárias para o funcionamento do BBR [9]. A segunda RFC utilizada define o projeto e lógica do algoritmo BBR em si,

e descreve as funções essenciais para implementar o algoritmo [4]. Essas RFCs representam o resultado de esforço coletivo, e nos ajudaram a navegar pelas complexidades de se implementar um algoritmo de controle de congestionamento.

Para desenvolver o protótipo, utilizamos a pilha TCP/IP do smoltcp [3], que oferece suporte a recursos essenciais, como janela de congestionamento e retransmissão rápida, necessários para a implementação do BBR e outros algoritmos de controle de congestionamento. Para permitir que o algoritmo BBR fosse integrado à biblioteca, algumas pequenas modificações foram realizas no código original. Notavelmente, essas modificações incluíram a adição de chamadas a ganchos quando determinados eventos ocorrem na pilha. Esses eventos são o recebimento de um ACK, o envio de um segmento, o esgotamento de um temporizador e a entrada e saída da retransmissão rápida.

Como as métricas obtidas pelo BBR dependem de uma estimativa de taxa de entrega, também foram implementas as funções definidas na RFC *Delivery Rate Estimation*. Ao enviar ou retransmitir um pacote, salvamos alguns metadados desse pacote, que codificam o registro de tempo atual, o último registro de tempo salvo e a quantidade total de pacotes entregues até o momento. A cada ACK recebido, calculamos o intervalo desde o último ACK, e o intervalo entre envios, usando os registros codificados nos metadados. Definimos o intervalo de amostragem como o valor máximo entre os dois intervalos calculados. Então definimos o volume de entrega como a quantidade de pacotes entregues nesse momento menos a quantidade total de pacotes no confirmados até o último ACK. Com esses resultados, calculamos a estimativa de taxa de entrega conexão como o volume de entrega dividido pelo intervalo de amostragem.

A implementação do algoritmo BBR consiste em funções que respondem a determinados ganchos que a pilha chama durante eventos discutidos previamente. Quando uma confirmação de entrega é recebida, atualizamos o nosso modelo interno, que consiste em checar se as estimativas de limite superior de largura de banda e o limite inferior de tempo de resposta estão desatualizados, e atualizar o estado atual do BBR. O limite superior da largura de banda é atualizado utilizando uma janela que contém as duas amostras mais recentes da taxa de entrega. Desta janela, extraímos o valor mais alto como uma estimativa. Em contraste, o limite inferior do tempo de resposta é atualizado sempre que obtemos uma estimativa menor ou após um intervalo de 10 segundos desde a última atualização.

Usando o seu modelo interno, consistindo nos limites atualizados descritos anteriormente, é realizado o cálculo do tamanho da janela de congestionamento. Usando o BDP estimado, definimos o objetivo do tamanho da janela pela equação abaixo, onde o ganho da janela é definido pelo estado atual do BBR. A proposta é que o BDP estimado modela a quantidade máxima de dados que pode ser transmitida através de um determinado

caminho de rede, permitindo que o fluxo envie na taxa de *MaxBw* pela duração de um *MinRTT*. O cálculo do objetivo é definido abaixo.

$$EstimatedBdp = MaxBW \cdot MinRtt$$

$$TargetCwnd = EstimatedBdp \cdot CwndGain + 3$$

Durante períodos de envio normal, nós gradualmente aumentamos a janela de congestionamento para corresponder ao objetivo. Durante períodos de recuperação de perdas, salvamos a janela de congestionamento e a reduzimos pela quantidade de dados perdidos. Quando a recuperação de perdas é concluída, restauramos a janela de congestionamento para o último valor salvo. Quando a pilha TCP necessita enviar dados, a quantidade máxima de dados enviados será limitada, entre outros fatores, pela tamanho janela de congestionamento atual menos quantidade de dados em trânsito.

Quando um temporizador TCP expira, ou seja, quando não recebe um reconhecimento esperado dentro da janela de tempo esperada, avisamos o algoritmo de congestionamento BBR para entrar no modo de recuperação de perda e reduzir a janela de congestionamento para 1 segmento. Quando recebemos 3 confirmações duplicadas, entramos no estado de retransmissão rápida e sinalizamos o controle de congestionamento que estamos entrando no modo de recuperação de perda, mas agora reduzimos a janela para a quantidade de dados em trânsito mais 1 pacote. Quando determinamos que a recuperação de perda terminou, restauramos a janela de congestionamento para o valor que estava antes de entrar na recuperação.

4.2 Ferramentas

Para tornar possível a coleta de uma métrica útil para comparação entre os algoritmos, utilizamos a ferramenta *iperf3* [10]. *iperf3* é uma ferramenta de código aberto que permite o envio ou recebimento de dados sobre os protocolos TCP ou UDP e exibe a largura de banda atingida para cada conexão realizada. A ferramenta foi utilizada nos experimentos para envio dos dados utilizando NewReno e CUBIC, e para recebimento utilizando os três algoritmos sob avaliação. Escolhemos o *iperf3* por ser amplamente utilizado para testar o desempenho de redes e por medir e exibir a vazão de transmissões, que é a métrica sobre análise nesse estudo.

Com o objetivo de criar experimentos que reproduzissem fluxos observados em situações reais, empregamos a ferramenta *mahimahi* [11]. Essa ferramenta nos permitiu introduzir atrasos artificiais na entrega dos pacotes, descartar pacotes de acordo com uma distribuição aleatória uniforme e limitar a taxa de transferência dos pacotes. Realizamos

essas ações com o objetivo de simular condições que se aproximam do congestionamento observado em um *link* de rede. Ao utilizar o *mahimahi*, pudemos explorar diferentes cenários e obter resultados mais representativos e relevantes para a avaliação das implementações de algoritmos de controle de congestionamento. Escolhemos o *mahimahi* pois a ferramenta permite diferentes degradações de rede ao mesmo tempo que permitem emular os cenários desejados.

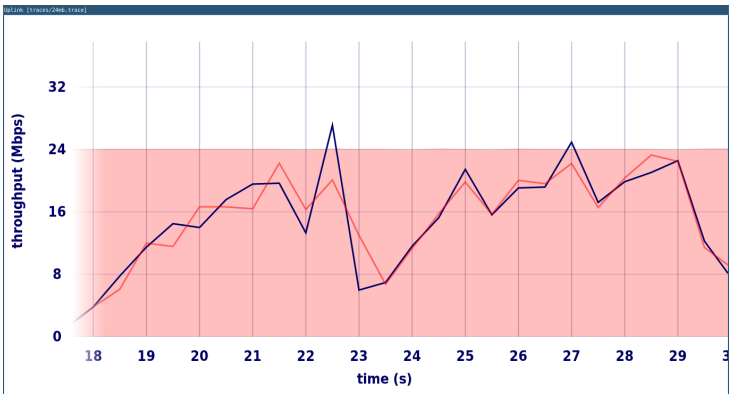

Figura 4.1: Captura de tela com a ferramenta *mahimahi* exibindo um gráfico em tempo real da vazão de envio de dados.

4.3 Experimentos

Para investigar e comparar de forma empírica o desempenho do BBR, CUBIC e NewReno, realizamos uma pesquisa quantitativa, pois esse método melhor avalia nosso objetivo. A natureza da nossa questão de pesquisa, que envolve a medição da vazão sob condições variadas, necessita da objetividade de métodos quantitativos. Dessa forma, somos capazes de gerar dados numéricos concretos que podem ser analisados de forma estatística para encontrar diferenças entre cada algoritmo.

Para realizar a coleta de dados usados para análise, obtemos medições de *goodput* utilizando a ferramenta *iperf3*. Definimos *goodput* como a taxa efetiva de transferência de dados úteis em um fluxo TCP, excluindo o *overhead* de protocolo e retransmissões. Ou seja, ao transferir um arquivo, o *goodput* experienciado é igual ao tamanho do arquivo dividido pelo intervalo de tempo entre o início e o fim da transmissão.

Escolhemos esse método como o ideal para nossa avaliação, pois o *goodput* fornece uma medida direta do desempenho de transferência de dados relevantes e por ser de fácil medição. Outras métricas, como a quantidade de retransmissões, foram consideradas, mas descartadas por dificuldades de medição. Essa métrica nos ajuda a avaliar se o algoritmo de controle de congestionamento está cumprindo seu objetivo de reduzir congestionamento com eficiência. O congestionamento na rede reflete diretamente na quantidade

de dados perdidos, e por consequência na quantidade de retransmissões. Dessa forma, diferentemente da taxa de transferência bruta, o *goodput* considera apenas os dados úteis transmitidos, refletindo com mais precisão a capacidade da rede em lidar com a carga de trabalho real. Adicionalmente, escolhemos as seguintes analises pois acreditamos que os resultados podem oferecer percepções valiosas a respeito do desempenho dos algoritmos em diferentes estados de rede.

Acredita-se que o algoritmo BBR possa apresentar um desempenho geral de *goodput* superior ao lidar com perdas, uma vez que, em contraste com os algoritmos tradicionais que foram comparados, o design do BBR não interpreta todas as perdas como um sinal de congestionamento. Isso permite que o BBR mantenha um fluxo constante de transmissão de dados com maior eficiência. Como resultado, espera-se que o BBR seja capaz de alcançar um *goodput* total mais alto em comparação com os algoritmos tradicionais durante situações de perda na rede.

Cada experimento envolveu a transferência de 50 megabytes de dados, que é uma quantidade grande o suficiente para permitir observar os efeitos dos diferentes algoritmos, mas não grande demais para tornar testes muito demorados e reduzir a viabilidade dos testes. O caminho de dados foi limitado com taxa de perda e limitações de largura de banda que variam conforme o experimento, e utilizando um atraso sintético de 50 milissegundos, que representa um tempo de propagação representativo de redes reais na Internet. Para cada variação dos parâmetros de rede, foram realizadas 30 transferências para aumentar o intervalo de confiança nos resultados, para cada algoritmo sob avaliação.

4.3.1 Análise comparativa de variação de chance de queda

Uma análise comparativa de variação de chance de queda tem o objetivo de comparar e analisar as probabilidades de ocorrência de falhas ou quedas em uma rede de computadores. Assim, é possível identificar e compreender as causas subjacentes das quedas e os impactos de diferentes configurações de rede no desempenho. Esta chance refere-se à probabilidade de que a rede deixe de funcionar corretamente, o que pode ocorrer devido a vários fatores, como problemas de conectividade, erros de configuração, sobrecarga de tráfego e até mesmo ataques cibernéticos.

Por se tratar de uma análise comparativa, além do BBR, serão observadas as variações nos protocolos CUBIC e NewReno. Os resultados da observação devem auxiliar a identificação de possíveis problemas na implementação e a constatação do desempenho desta implementação do TCP BBR em espaço de usuário, fornecendo uma melhor compreensão a respeito da aplicação do TCP BBR em diferentes contextos de rede.

É esperado que o algoritmo BBR apresente uma taxa de perda menor em comparação com os algoritmos NewReno e CUBIC. Uma vez que o BBR, através da redução da taxa

de envio para prevenir perdas adicionais, é mais ágil em tempo de reação aos eventos de perda, enquanto as estratégias de controle de congestionamento dos demais são menos sensíveis às perdas de pacote.

Além disso, espera-se que a média de vazão do TCP BBR seja maior em comparação com os demais algoritmos. O BBR é projetado para explorar eficientemente a largura de banda disponível, através do ajuste dinâmico da taxa de envio baseado na estimativa de gargalo da rede e no atraso do RTT. Assim, a vazão da conexão neste cenário deve ser maximizada, especialmente em comparação com os algoritmos NewReno e CUBIC, que são menos adaptativos em relação às condições da rede.

4.3.2 Análise comparativa de variação do limite de taxa de envio

A análise comparativa de variação do limite de taxa de envio aponta como esta variação afeta o desempenho geral da rede, considerando métricas como vazão, atraso e taxa de perdas de pacotes. Para realizar esse experimento, utilizamos limites de envio desde 6 megabits por segundo até 24 megabits por segundo, já que esses valores são facilmente obtidos na ferramenta de limitação e são valores representativos de redes reais. O experimento foi realizado em diversas iterações, visando a exclusão de possíveis variações estatísticas.

Como o foco é isolar a taxa de perda adicional e variar apenas o limite de banda, é esperado que a taxa de perda adicional seja relativamente baixa para os três algoritmos analisados. Não espera-se que a variação do limite de taxa de enfio afete diretamente a ocorrência de perdas de pacotes, embora a taxa de perda possa depender de outros fatores relacionados aos recursos e condições da rede. Em contrapartida, espera-se que a média de vazão varie de acordo com o limite da taxa de envio para as três algoritmos.

Ainda que a taxa de crescimento da vazão possa variar entre os protocolos, a média de vazão deve aumentar de acordo com o aumento do limite da taxa de envio. De forma análoga, a porcentagem de uso do caminho também deve acompanhar esta variação. Por fim, espera-se que a implementação do TCP BBR em espaço de usuário apresente desempenho superior quando comparada aos protocolos NewReno e CUBIC, especialmente em cenários com maior limite de banda.

4.3.3 Análise comparativa com registros de redes reais

A utilização de registros reais na análise visa a obtenção de resultados aproximados à cenários reais de carga de trabalho. Os registros reais podem ser obtidos através de ferramentas de monitoramento de rede e contêm informações a respeito do tráfego de rede, como endereços de origem e destino, protocolos, pacotes transmitidos e etc. A

análise permite identificar quais algoritmos resultam em melhor desempenho, estabilidade e utilização eficiente dos recursos da rede, além do comportamento em cenários de redes reais por meio da comparação de parâmetros como taxa de transferência, atraso e perda de pacotes.

Os registros utilizados foram coletados de diferentes redes móveis comerciais, a partir do registro de horário de recebimento de pacotes durante um intervalo de tempo [12]. Esses registros de rede são disponibilizados junto com a ferramenta *mahimahi*, que tem suporte a interpretar esses registros e limitar a entrega de pacotes simulando a rede origem. Eles fornecem uma visão mais abrangente e representativa das dinâmicas da rede em situações reais. Ao aplicar os algoritmos BBR, NewReno e CUBIC a esses registros, pudemos observar como cada algoritmo se comporta diante de variações nas características da rede, como taxa de transferência, atraso e perda de pacotes. Essa análise com registros reais complementa nossos resultados anteriores, fornecendo revelações valiosas sobre o desempenho dos algoritmos em ambientes mais próximos das condições do mundo real.

Capítulo 5

Resultados

5.1 Análise comparativa de variação de chance de queda

O objetivo deste experimento foi avaliar o desempenho da implementação do algoritmo BBR quando enfrentando perda de pacotes na rede, em comparação com outros algoritmos de controle de congestionamento já existentes. A Figura 5.1 apresenta os resultados do envio de um arquivo de 50 megabytes utilizando os algoritmos BBR, NewReno e CUBIC, com uma limitação de banda constante de 12Mbps e um atraso sintético de 50 milissegundos. A taxa de perda variou de nenhuma perda até 1%. Os fluxos observados apresentaram uma média de vazão de 6,39 Mbps nos experimentos para o BBR, maior que as médias observadas para o NewReno e o CUBIC, de 4,66 Mbps e 4,61 Mbps respectivamente.

A implementação do BBR é mais eficiente em termos de vazão quando comparada aos algoritmos estabelecidos, em níveis de perda acima de 0,001. À medida que a taxa de perda aumenta, a diferença de desempenho tende a aumentar, apresentando uma relação de 2,44 no maior nível de perda do experimento em comparação aos demais algoritmos. Isso pode ser explicado pela tendência dos outros algoritmos de interpretar perda como um sinal de congestionamento e pela falta de mecanismos de recuperação da janela de congestionamento antes da perda. Portanto, em redes com níveis consideráveis de perda, BBR tende a exibir uma utilização melhor do caminho.

Estes resultados reforçam a importância do algoritmo BBR como uma solução robusta para o controle de congestionamento em redes com condições adversas. A capacidade do BBR de adaptar-se dinamicamente às variações na taxa de perda e otimizar a taxa de envio de pacotes torna-o uma opção promissora para melhorar a eficiência da comunicação em ambientes de rede desafiadores.

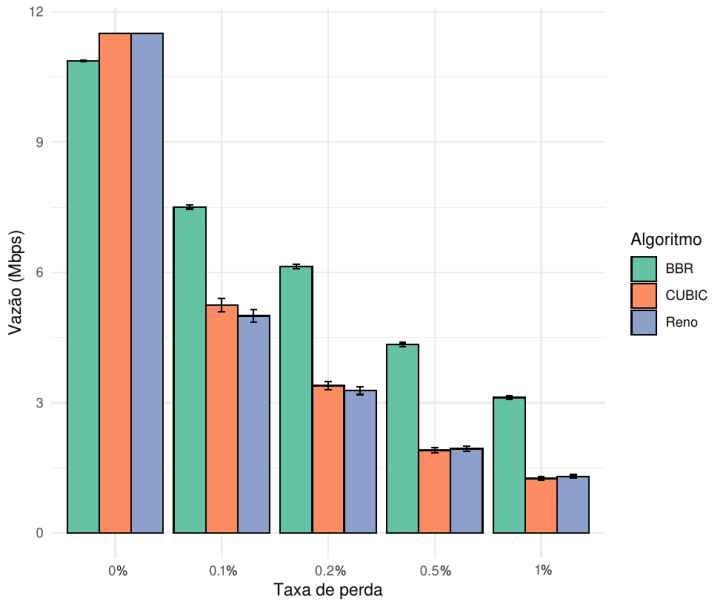

Figura 5.1: Vazão observada de cada algoritmo para diferentes níveis de taxa de perda

5.2 Análise comparativa de variação do limite de taxa de envio

Neste experimento, buscamos isolar a taxa de perda adicional e variar apenas o limite de banda, a fim de simular um caminho com restrição. Na Figura 5.2, podemos observar como a vazão observada de cada algoritmo corresponde ao limite atribuído. Os limites variaram de 6 Mbps a 24 Mbps, com uma chance de perda de pacotes de 0,1%. Utilizamos os algoritmos BBR, NewReno e CUBIC para o envio de arquivos de 50 megabytes, durante 30 rodadas e obtemos a vazão média dessas 30 rodadas. O algoritmo BBR seguiu as expectativas descritas na seção de metodologia, e apresentou uma vazão média de 7,35 Mbps, comparado com as médias observadas para o NewReno e o CUBIC de 5,26 Mbps e 4,95 Mbps respectivamente. Além disso, apresentou um coeficiente de variação médio entre os testes de 0,47, enquanto o NewReno e o CUBIC apresentaram medidades de 0,12 e 0,13 respectivamente, indicando que o BBR tem uma variabilidade menor entre execuções sob as mesmas condições.

O BBR demonstrou uma capacidade inferior de utilização do caminho somente quando restrito a 6 Mbps. À medida que o limite de banda atribuído aumenta, o BBR começa a exibir uma vazão superior aos outros algoritmos. Isso se deve à sua capacidade de explorar eficientemente a banda do caminho, graças ao seu mecanismo de estimativa do limite superior de banda. Os algoritmos NewReno e CUBIC não possuem mecanismos semelhantes, e falham utilizar de forma eficiente a largura de banda disponível no caminho.

No limite mais baixo dos nossos experimentos, BBR e NewReno exibiram utilização

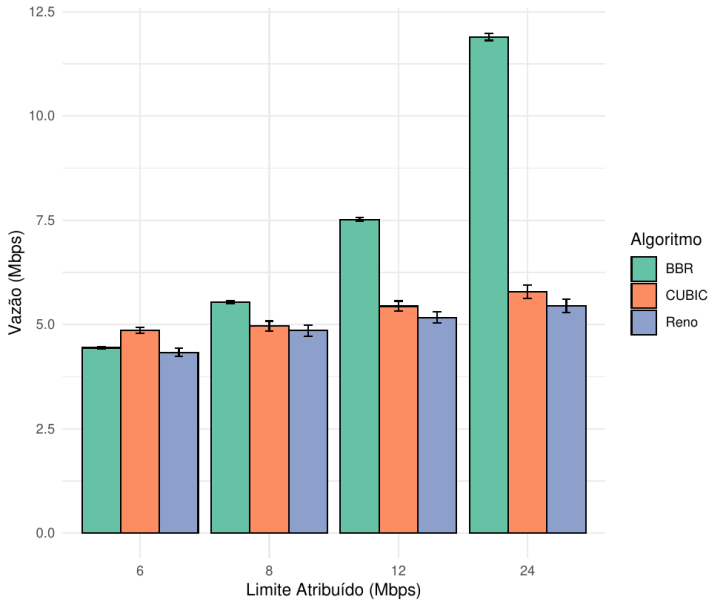

Figura 5.2: Vazão observada dos algoritmos para diferentes níveis de limite de largura de banda.

similar de 73%, com desempenho marginalmente inferior ao CUBIC, que exibiu utilização de 81%. Por outro lado, no maior limite de banda do nosso experimento, BBR superou ambos NewReno e CUBIC, mantendo uma utilização do caminho de aproximadamente 50%, enquanto os outros ficaram restritos a 24%. O desempenho inferior do BBR nos limites mais baixos pode ser explicado por seus períodos de redução de janela de congestionamento para atualização de métricas, que para certos fluxos torna o algoritmo muito conservador. No entanto, nos demais níveis de limite de banda, BBR foi capaz de manter uma utilização mais alta do caminho de rede.

Estas observações observações destacam a importância de considerar as demandas de limite e latência da rede ao escolher o algoritmo de controle de congestionamento mais adequado. O desempenho diferenciado do BBR em diferentes limites de taxa de envio ressalta sua capacidade de se adaptar às condições da rede e otimizar o uso dos recursos disponíveis.

5.3 Análise comparativa com registros de redes reais

Este experimento tem o objetivo de se aproximar a cenários reais de carga de trabalho utilizando registros de rede gravados com dados reais. Os resultados desse experimento exibem o comportamento dos 3 algoritmos, onde BBR exibiu, em média uma vazão de 5,3 Mbps, seguido do CUBIC com 4,66 Mbps e do NewReno com 3,71 Mbps. Essas descobertas sugerem que o algoritmo BBR deve oferecer um desempenho superior aos

algoritmos CUBIC e NewReno quando considerando a vazão média nessas condições de rede.

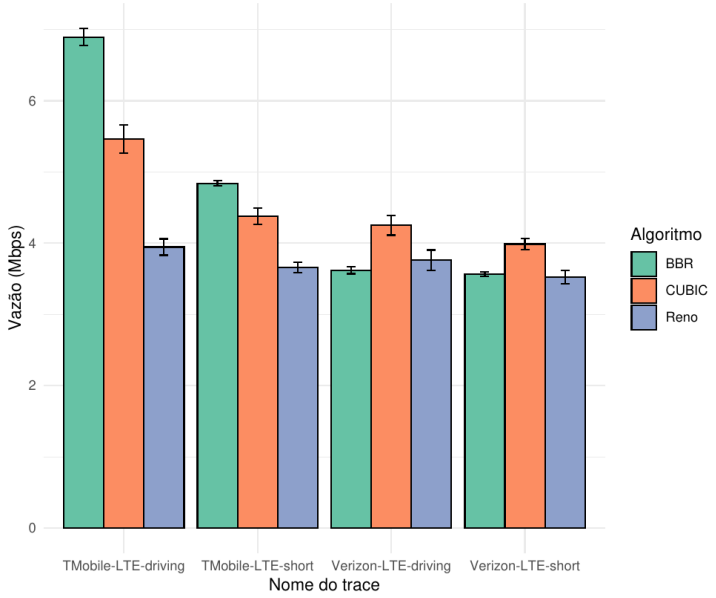

Figura 5.3: Vazão observada dos algoritmos para diferentes registros de rede

Ao observar os dados de acordo com cada registro de rede, uma visão mais detalhada surge. Como observado na Figura 5.3, o desempenho superior do BBR é limitado aos registros *TMobile*, onde exibe uma vazão média nos testes 30% maior comparado aos algoritmos tradicionais. Nos registros *Verizon*, enquanto o BBR demonstra desempenho competitivo em comparação ao NewReno, sem diferença estatística significativa entre os dois, o algoritmo CUBIC se mostra na frente. Isso destaca que o desempenho desses algoritmos é dependente do contexto, enfatizando a importância de conhecer as condições da rede e realizar testes para a decisão do algoritmo de controle de congestão. Em resumo, a vantagem relativa do BBR é em grande parte limitada aos experimentos da rede *TMobile*.

Capítulo 6

Lições Aprendidas

Originalmente, o presente trabalho contava com um escopo diferente do relatado nesta monografia. Pretendíamos realizar uma abordagem sobre Redes Definidas por Software (SDN, na sigla em inglês), um paradigma de redes de computadores cujo objetivo é flexibilizar a gestão de redes através da separação entre controle e infraestrutura de transporte. A programabilidade encontrada em SDNs oferece visibilidade centralizada, facilitando o gerenciamento e a evolução da rede, uma vez que permite alterações dinâmicas no fluxo a partir de um ponto único de controle.

A implementação de uma SDN pode levantar desafios relacionados à complexidade, segurança, interoperabilidade e performance, a depender da aplicação analisada. No processo de mitigação destes pontos, a utilização de diretrizes apresentadas por protocolos já consolidados em redes convencionais é bem-vinda enquanto ponto de partida para o cumprimento dos requisitos da rede.

Para superar as barreiras de acesso ao hardware, havíamos optado tomar como base a utilização do P4 (*Programming Protocol-Independent Packet Processors*), uma linguagem de programação *open-source* voltada para processamento de pacotes em redes definidas por software, que conta com alta flexibilidade para aplicações devido ao suporte em diversas plataformas. Dentre as plataformas suportadas, destaca-se o P4Pi (*P4 Programming on Raspberry Pi*), uma implementação da linguagem P4 para os dispositivos Raspberry Pi. Sua aplicabilidade se mostra atraente para fins de pesquisa devido à portabilidade e baixo custo para a construção de redes de computadores reais, entretanto, ressalta-se a preocupação e o cuidado com a utilização otimizada dos recursos de hardware disponíveis, pois estes são limitados.

Como levantado e proposto no trabalho de Gomez *et al.* (2022) [1], as implementações atuais de detecção rápida de perdas em P4 não inferem a causa raiz das perdas de pacote, pois, somente reagem à congestão. Novos trabalhos devem aprimorar o algoritmo de inferência de causa, e a utilização de esquemas que apliquem algoritmos de controle de

congestão não-baseados em perdas, como o TCP BBR.

Feldman *et al.* (2020) [13] propuseram um sistema chamado *Network-assisted Congestion Feedback* (NCF), que controla a taxa de transmissão ao enviar notificações explícitas de congestão ao remetente utilizando NACKs. O NCF classifica o tráfego em duas categorias principais: fluxo-elefante e fluxo-rato, e aplica o controle de congestão somente em pacotes do fluxo-elefante, pois correspondem aos dados congestionados. Todavia, ainda não foram apresentados resultados de implementação para avaliar a efetividade da solução.

No escopo inicialmente proposto, buscamos implementar o sistema NCF combinado com as técnicas de controle de congestionamento do TCP BBR em um ambiente P4Pi. Por fim, o resultado pretendido seria o de uma rede definida por software que contasse com uma combinação ideal de eficiência de largura de banda e confiabilidade de entrega de dados.

O hardware foi o principal empecilho na implementação do escopo descrito acima. A utilização do P4Pi se mostrou inconsistente em diversos testes, apresentando lentidão na compilação e execução de determinados exemplos. Adicionalmente, o Raspberry Pi nos surpreendia com interrupções abruptas durante as execuções, deixando de funcionar corretamente em grande parte de nossas tentativas. Destacando os resultados que obtivemos ao utilizar o aparelho em ambientes com baixas temperaturas, a principal hipótese que levantamos é a de que o dispositivo estava superaquecendo, e por conta disso, desligando prematuramente.

Posteriormente, entendemos que possivelmente a plataforma P4Pi não seja a mais adequada para o escopo planejado, neste caso, faz-se necessária a utilização de dispositivos com maior poder de processamento. Entretanto, apesar dos desafios encontrados na implementação do escopo originalmente proposto, a utilização de máquinas virtuais do P4Pi pode ser considerada como uma opção viável para fins de pesquisa, uma vez que estas possuem maior poder de processamento para atender à diferentes demandas com maior eficiência.

Por fim, é evidente que a jornada de pesquisa deste trabalho foi desafiadora e levou a alterações significativas no escopo. No entanto, os obstáculos encontrados durante a implementação, como a lentidão na compilação, execução e interrupções abruptas, levantaram dúvidas sobre a adequação do P4Pi para atender às demandas específicas deste estudo. Assim, fica claro que a escolha da plataforma de pesquisa requer uma cuidadosa avaliação dos recursos necessários e da capacidade dos dispositivos disponíveis.

Este capítulo de Lições Aprendidas surgiu como uma valiosa oportunidade para refletir sobre os desafios encontrados e aprender com as limitações encontradas, reforçando a importância da adaptabilidade e flexibilidade na condução de experimentos.

Capítulo 7

Conclusão

Neste trabalho, foi apresentado o algoritmo BBR de controle de congestionamento e um projeto de implementação da sua funcionalidade em uma biblioteca de comunicação da pilha TCP/IP em espaço de usuário. O algoritmo implementado segue o rascunho apresentado em Cardwell *et al.* (2017) [4]. Essa implementação foi testada em uma série de experimentos para medir a vazão do envio de dados, em comparação com outros algoritmos de controle de congestionamento já estabelecidos, especificamente NewReno e CUBIC.

A implementação discutida cumpriu o objetivo de controle de congestionamento, exibindo desempenho comparável ao do NewReno e CUBIC através dos testes. Ademais, o BBR demonstrou um desempenho superior em termos de vazão média geral em cada experimento avaliado. Entretanto, quando isolamos algumas variáveis nos experimentos, o BBR nem sempre se mostrou o melhor algoritmo em termos de vazão, principalmente quando analisamos o terceiro experimento, que se aproxima mais de cenários de uso reais. Isso reforça a ideia de que a escolha do algoritmo deve ser sempre relativa aos parâmetros da rede em questão. Além disso, sua complexidade de implementação elevada, que envolve implementação de duas RFCs, pode ser um agravante na sua adoção, principalmente em dispositivos embarcados e com recursos limitados.

Para o futuro, recomendamos a realização de mais testes para compreender as limitações do BBR e identificar em quais cenários ele se destaca em comparação com outros algoritmos. Seria interessante conduzir testes que envolvam múltiplos fluxos concorrentes competindo por banda no caminho de rede e também testes com topologias complexas que possuam filas reais.

Para o futuro, recomendamos a realização de um conjunto abrangente de testes adicionais, visando compreender profundamente as nuances e limitações do algoritmo BBR e identificar com precisão em quais cenários ele se destaca em comparação com outros algoritmos de controle de congestionamento. Nesse sentido, sugerimos a condução de expe-

rimentos que envolvam múltiplos fluxos concorrentes, competindo por banda em diversos caminhos de rede. Essa abordagem permitirá uma análise mais realista do desempenho do BBR em situações de tráfego intenso e carga elevada, onde o algoritmo pode enfrentar desafios e oportunidades de otimização.

Além disso, seria igualmente interessante realizar testes em topologias complexas, aquelas que apresentam configurações de rede mais intricadas e que possuam filas reais. Isso permitirá uma avaliação mais precisa do comportamento do BBR em ambientes reais, onde os atrasos de fila, as variações de rota e outras condições da rede podem desempenhar um papel significativo no desempenho do algoritmo. Além do escopo técnico, também incentivamos a realização de testes com diferentes cenários de aplicação, explorando casos de uso variados, desde aplicações de transmissão de mídia até aplicações de tempo real, como jogos online e chamadas de voz/vídeo. Ao entender como o BBR se comporta em diferentes contextos, será possível estabelecer suas vantagens e limitações de forma mais holística.

A implementação apresentada também pode ser estendida no futuro, adicionando as alterações feitas na versão 2 do BBR, descritas em [14]. Adicionalmente, modificações podem ser feitas na biblioteca base para dar suporte ao TCP Pacing, que pode trazer melhorias em termos de eficiência no controle de congestionamento. O TCP Pacing refere-se à técnica de espaçamento uniforme dos pacotes enviados em uma conexão, com o objetivo de evitar a criação de disparos excessivos de pacotes. Entretanto, este caso deve ser analisado com cautela, uma vez que a aplicação da técnica pode aumentar a latência e diminuir a performance geral.

Por fim, este trabalho contribuiu para a compreensão do algoritmo BBR e sua implementação, fornecendo percepções valiosas sobre seu desempenho em diferentes cenários. Esperamos que as descobertas aqui apresentadas inspirem pesquisas adicionais e contribuam para o aprimoramento contínuo dos algoritmos de controle de congestionamento em redes de computadores.

Referências

[1] Gomez, Jose, Elie F. Kfoury, Jorge Crichigno e Gautam Srivastava: *A survey on TCP enhancements using P4-programmable devices*. Computer Networks: The International Journal of Computer and Telecommunications Networking, 212(C), julho 2022, ISSN 1389-1286. https://doi.org/10.1016/j.comnet.2022.109030, acesso em 2023-06-28. 2, 27

[2] Ha, Sangtae, Injong Rhee e Lisong Xu: *CUBIC: a new TCP-friendly high-speed TCP variant*. ACM SIGOPS Operating Systems Review, 42(5):64–74, julho 2008, ISSN 0163-5980. https://doi.org/10.1145/1400097.1400105, acesso em 2023-06-26. 3, 10

[3] M-Labs: *smoltcp - Rust*. https://docs.rs/smoltcp/latest/smoltcp/, acesso em 2023-06-26. 5, 14, 17

[4] Cardwell, Neal, Yuchung Cheng, Soheil Hassas Yeganeh e Van Jacobson: *BBR Congestion Control*, julho 2017. https://datatracker.ietf.org/doc/html/draft-cardwell-iccrg-bbr-congestion-control-00. 6, 17, 29

[5] Cardwell, Neal, Yuchung Cheng, C. Stephen Gunn, Soheil Hassas Yeganeh e Van Jacobson: *BBR: Congestion-Based Congestion Control: Measuring bottleneck bandwidth and round-trip propagation time*. Queue, 14(5):20–53, outubro 2016, ISSN 1542-7730. https://dl.acm.org/doi/10.1145/3012426.3022184, acesso em 2023-06-25. 13

[6] Edwards, Aled e Steve Muir: *Experiences implementing a high performance TCP in user-space*. ACM SIGCOMM Computer Communication Review, 25(4):196–205, outubro 1995, ISSN 0146-4833. https://dl.acm.org/doi/10.1145/217391.318122, acesso em 2023-06-25. 14

[7] Lueke, Kai Tobias: *Memory-safe network services through a userspace networking switch*. 2019. https://koasas.kaist.ac.kr/handle/10203/267040, acesso em 2023-06-26. 14

[8] Developers, Redox: *Redox - Your Next(Gen) OS*. https://www.redox-os.org/, acesso em 2023-06-26. 14

[9] Cheng, Yuchung, Neal Cardwell, Soheil Hassas Yeganeh e Van Jacobson: *Delivery Rate Estimation*. Internet Draft draft-cheng-iccrg-delivery-rate-estimation-00, Internet Engineering Task Force, julho 2017. https://datatracker.ietf.org/doc/draft-cheng-iccrg-delivery-rate-estimation/00/, acesso em 2023-06-15. 16

[10] *iPerf - The TCP, UDP and SCTP network bandwidth measurement tool*. `https://iperf.fr/`, acesso em 2023-06-28. 18

[11] Netravali, Ravi, Anirudh Sivaraman, Somak Das, Ameesh Goyal, Keith Winstein, James Mickens e Hari Balakrishnan: *Mahimahi: accurate record-and-replay for HTTP*. Em *Proceedings of the 2015 USENIX Conference on Usenix Annual Technical Conference*, USENIX ATC '15, páginas 417–429, USA, julho 2015. USENIX Association, ISBN 9781931971225. 18

[12] Winstein, Keith, Anirudh Sivaraman e Hari Balakrishnan: *Stochastic Forecasts Achieve High Throughput and Low Delay over Cellular Networks*. Em *10th USENIX Symposium on Networked Systems Design and Implementation (NSDI 13)*, páginas 459–471, Lombard, IL, abril 2013. USENIX Association, ISBN 978-1-931971-00-3. `https://www.usenix.org/conference/nsdi13/technical-sessions/presentation/winstein`. 22

[13] Feldmann, Anja, Balakrishnan Chandrasekaran, Seifeddine Fathalli e Emilia N. Weyulu: *P4-enabled Network-assisted Congestion Feedback: A Case for NACKs*. Em *Proceedings of the 2019 Workshop on Buffer Sizing*, BS '19, páginas 1–7, New York, NY, USA, janeiro 2020. Association for Computing Machinery, ISBN 9781450377454. `https://doi.org/10.1145/3375235.3375238`, acesso em 2023-06-28. 28

[14] Cardwell, Neal, Yuchung Cheng, Soheil Hassas Yeganeh, Ian Swett e Van Jacobson: *BBR Congestion Control*. Internet Draft draft-cardwell-iccrg-bbr-congestion-control-02, Internet Engineering Task Force, março 2022. `https://datatracker.ietf.org/doc/draft-cardwell-iccrg-bbr-congestion-control/02/`, acesso em 2023-06-27. 30