Multipath Routing Compatible Congestion Control

by

Tianfang Chang
B.Eng., Beijing University of Posts and Telecommunications, China, 2020

A Thesis Submitted in Partial Fulfillment of the
Requirements for the Degree of

MASTER OF APPLIED SCIENCE

in the Department of Electrical and Computer Engineering

Multipath Routing Compatible Congestion Control

by

Tianfang Chang
B.Eng., Beijing University of Posts and Telecommunications, China, 2020

Supervisory Committee

---

Dr. Lin Cai, Supervisor
(Department of Electrical and Computer Engineering)

---

Dr. Amirali Baniasadi, Departmental Member
(Department of Electrical and Computer Engineering)

**Supervisory Committee**

---

Dr. Lin Cai, Supervisor
(Department of Electrical and Computer Engineering)

---

Dr. Amirali Baniasadi, Departmental Member
(Department of Electrical and Computer Engineering)

## ABSTRACT

The evolution of network applications poses significant challenges to network service provisioning. Multipath routing and packet spraying techniques have become crucial in networks. TCP performance declines sharply on multipath setups where significant packet reordering occurs, as unordered transmissions are misinterpreted as packet loss and congestion signals. We propose the Multipath Routing Compatible (MPRC) congestion control, which utilizes the delay-sensitive Fast Retransmission Timeout (FastRTO) to decouple reordering from loss signals and enhance loss detection. This modification improves congestion window adjustments in multipath environments and handles packet reordering effectively, ensuring stable TCP throughput across multipath settings. Our algorithm was implemented on the NS-3 simulator platform and compared with other congestion control algorithms across various network topologies, in both single-path and multipath routing scenarios. The results demonstrate that MPRC can handle both sporadic and persistent packet reordering, ensuring steady throughput in multipath routing environments while maintaining compatibility and fairness in bandwidth competition, which paves the way for efficient congestion control adopting multi-path routing networks.

# Contents

# List of Figures

# ACKNOWLEDGEMENTS

I would like to thank:

**Dr. Lin Cai,** for providing me with countless opportunities during my Master's program. I deeply appreciate the countless hours she invested in my development, instructing me in the nuances of research, effective time management, and professional interactions. Her profound expertise and wisdom have significantly shaped my abilities and approach. I admire her relentless dedication to pushing my boundaries and aiding me in reaching my academic aspirations. My heartfelt thanks go to her for her steadfast support and invaluable guidance throughout my Master's program.

**My family,** for unwavering support throughout my studies. Their generous financial and emotional backing allowed me to pursue my academic dreams without limitations. Their presence and encouragement were my constant source of strength and motivation, enabling me to focus fully on my goals. I am truly blessed to have such a supportive circle in my life.

**My research group members,** for their assistance and inspiration throughout my master's program.

# Chapter 1

# Introduction

The rapid advancement of network applications has introduced new demands on network infrastructure, significantly challenging its capacity and efficiency. Emerging applications such as extended reality (XR) [29] and large language models (LLMs) [40], which contain tens of thousands of parameters, require substantial bandwidth and low latency to function effectively. These applications push the limits of current network capabilities, exacerbating existing bottlenecks and highlighting the need for more robust and efficient data transmission methods. In the vision for 6G networks, it is essential to address the challenges of seamless integration across complex networks spanning air, space, land, and sea, while also ensuring that diverse network applications meet their quality-of-service (QoS) requirements on a best-effort underlying network [6]. Decades of research have nearly exhausted the potential of single-path routing. In contrast, multipath routing which offers greater flexibility but also higher complexity, will further enhance network capacity.

To address these challenges and meet the increasing demands of these high-bandwidth applications, multipath routing, and packet-spraying delivery techniques have emerged as promising solutions [1]. These techniques leverage multiple simultaneous paths to distribute network traffic, thereby enhancing load balancing, reducing congestion, and improving overall network reliability. By mitigating the risks associated with single-path failures and ensuring more consistent data delivery, they further optimize network performance.

However, the dominant transport layer protocol, Transmission Control Protocol (TCP) [9], faces substantial challenges in this new context. TCP heavily relies on duplicate acknowledgments (ACKs) for detecting packet losses and controlling congestion. In single-path networks, this mechanism works effectively by using duplicate

ACKs to quickly detect and retransmit lost packets. In multipath routing scenarios, packet reordering is common when packets take different paths to the destination. The congestion control mechanism interprets the reception of duplicate ACKs as an indication of packet loss and congestion, triggering retransmissions and reducing the congestion window (cwnd). This misinterpretation leads to unnecessary retransmissions and a decrease in network throughput, as the protocol mistakenly identifies reordering as packet loss. Consequently, the overall network performance suffers, posing a significant challenge to the effective implementation of multipath routing and packet-spraying techniques.

Jacobson [16] initially devised the fast retransmit mechanism as a heuristic for early packet loss detection, activated when the receiver identifies a lost segment through three duplicate acknowledgments before the retransmission timeout (RTO) timer expires. However, duplicate acknowledgments may arise from packet loss, packet reordering, or infrequent explicit packet duplication. Since this mechanism is tailored for a network model from four decades ago—specifically, a low-bandwidth, wired, unipath-routing network—it exhibits suboptimal performance on contemporary networks with multipath routing, leading to persistent issues with out-of-order packets.

Numerous approaches have been proposed across different network layers to address the challenges that TCP congestion control faces in the context of multipath routing. One such approach is the UDP-based QUIC [20] protocol, which does not enforce the ordered delivery of packets. Despite this, QUIC's congestion control mechanism still relies on duplicated ACKs as congestion signals, similar to TCP. This reliance on duplicate ACKs means that while QUIC can handle out-of-order delivery to some extent, it does not fully resolve the issues caused by packet reordering in multipath environments.

Research on congestion control algorithms (CCAs) [3, 4, 39] has also explored various methods to improve TCP performance in the face of packet reordering. For instance, studies have focused on distinguishing between false retransmission losses and actual packet losses to reduce unnecessary retransmissions. These algorithms attempt to make TCP more robust by improving its ability to correctly interpret duplicate ACKs, but they often do not directly address the persistent packet reordering that is inherent in multipath routing scenarios.

Data centers, on the other hand, have developed infrastructure-level solutions to mitigate the impact of packet reordering. The symmetric topology of fat-tree networks

and the use of customized packet buffers allow data centers to manage microsecond-level reordering without significant adverse effects. These architectural enhancements ensure that packets can be reordered and processed efficiently, minimizing the impact on overall network performance.

However, existing solutions cannot effectively address the negative impact of multipath routing on TCP multipath routing and packet spraying. To fill the gap, in this thesis, we propose Fast Retransmission Timeout (FastRTO) as a loss detection mechanism triggered by network congestion, departing from the conventional use of duplicate ACKs. In the event of FastTimeOut, the congestion window is halved, instead of dropping to the minimum. This design can tolerate packet reordering induced by multipath routing and so it can support multipath routing for load balancing to achieve higher network efficiency.

In this thesis, We delve into the root causes of packet reordering and conduct a detailed analysis of its impact on TCP transmission efficiency. We introduce the Multipath Routing Compatible (MPRC) congestion control, as a solution to mitigate the decrease in TCP transmission efficiency caused by packet reordering in multipath routing scenarios. We design FastRTO, RTT-sensitive timeout trigger to replace duplicate ACKs for fast packet loss detection. This provides a constructive approach to addressing the challenge of packet loss due to network congestion. This presents a more constructive approach for congestion control over multipath routing.

## 1.1 Glossary

**ACK** Acknowledgment, a signal used in computer networking to indicate that a packet of data has been received successfully.

**CWND** Congestion Window, a TCP congestion control mechanism that limits the amount of data a sender can transmit before receiving an acknowledgment, helping to prevent network congestion.

**RTT** Round-Trip Time, the duration it takes for a signal to travel from a sender to a receiver and back again.

**RTO** Retransmission Timeout, the duration that a sender waits for an acknowledgment of a transmitted packet before retransmitting it.

**RTTVAR** Round-Trip Time Variation: a measure of the variability or fluctuation in the round-trip time of packets.

**SRTT** Smoothed Round-Trip Time, an estimate of the average round-trip time, calculated by smoothing the measured RTT values to account for variability and provide a more stable estimate.

## 1.2 Thesis Overview

**Chapter 1** gives an introduction to this thesis, followed by an overview of the structure of the document itself.

**Chapter 2** gives a detailed background of multipath routing and TCP congestion control algorithms (CCAs).

**Chapter 3** analyzes reasons why current CCAs fail to adapt to multipath routing.

**Chapter 4** presents the design and implementation of Multipath Routing Compatible Congestion Control (MPRC).

**Chapter 5** evaluates MPRC in different network scenarios and compares it with classical CCAs.

**Chapter 6** gives a conclusion of the thesis and presents future work.

# Chapter 2

# Background

## 2.1 Multipath Routing

Multipath routing is a significant advancement in network technology, offering substantial improvements in reliability, load balancing, and throughput. By allowing data to traverse multiple paths simultaneously, it mitigates the risks associated with single-path failures, thereby enhancing network resilience. This redundancy ensures that even if one path encounters congestion or failure, others can compensate, maintaining continuous data flow. In addition, multipath routing facilitates more efficient utilization of available bandwidth, as data packets can be distributed across multiple links, reducing bottlenecks and improving overall network performance. This capability is particularly crucial in high-demand environments such as data centers, cloud computing, and real-time applications where uninterrupted service and optimal performance are paramount.

When sending data from R1 to R6, multipath routing offers two key advantages, as shown in Fig. 2.1. In this network scenario, there are two possible paths: `R1-R2-R4-R6` and `R1-R3-R5-R6`. Firstly, in ideal network conditions, single-path routing can only utilize the transmission bandwidth of one path. In contrast, multipath routing can benefit from the combined bandwidth of both paths, effectively increasing throughput. Secondly, in the event of a link failure, single-path routing would be severely impacted as it relies solely on the availability of that single path. However, multipath routing can dynamically route data through the available path, maintaining connectivity and potentially avoiding the bottleneck created by the failed link.

Many multipath routing algorithms have been proposed in the past decades.

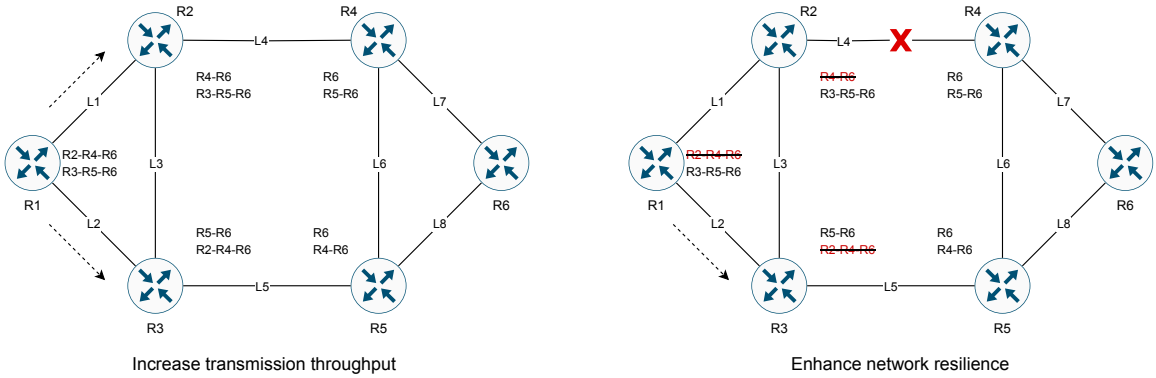


**Figure 2.1: Advantages of multipath routing**

Equal-Cost Multi-Path (ECMP) [15] routing is the classical routing strategy that enables using multiple paths with equal cost metrics for forwarding packets from a source to a destination. It is based on OSPF storing all paths that reach the destination at an equal cost in the routing table. This method aims to enhance network performance by improving resource utilization and providing increased reliability and fault tolerance.

More subsequent multipath algorithms are based on the k-shortest path routing algorithm. MDVA [35] is dedicated to finding all potential but non-oscillating paths, with the difference being the maintenance of distance vectors or link states. MARA [27] aims to maximize the minimum connectivity, and LFID [31] sets different priorities for different paths on the basis of MARA to further improve path selection. DGR [38] provides dynamic multipath routing based on the delay requirements of the service.

While multipath routing offers numerous advantages, it also presents significant challenges when used with traditional TCP protocols. One primary issue is the lack of coordination between multiple paths, which can lead to out-of-order packet delivery. TCP, designed for single-path communication, struggles to handle such scenarios efficiently, resulting in increased latency and reduced throughput.

In addition, existing TCP congestion control algorithms are not well-suited for the dynamic nature of multipath environments, which fail to fully utilize the available bandwidth across multiple paths, leading to suboptimal performance. This inefficiency is exacerbated by the varying path characteristics such as latency, bandwidth, and packet loss, which can cause significant throughput fluctuations and degrade the overall network performance. Furthermore, the reliance on a single congestion

window in TCP does not account for the independent congestion states of multiple paths, causing poor load balancing and increased congestion on certain routes. These limitations highlight the necessity for developing specialized congestion control algorithms tailored for multipath routing to leverage its full potential and ensure efficient, reliable data transmission.

## 2.2 TCP Congestion Control Algorithms (CCAs)

The Transmission Control Protocol (TCP) provides a communication service at an intermediate level between an application program and the Internet Protocol. It provides host-to-host connectivity at the transport layer of the Internet model. An application does not need to know the particular mechanisms for sending data via a link to another host, such as the required IP fragmentation to accommodate the maximum transmission unit of the transmission medium. At the transport layer, TCP handles all handshaking and transmission details and presents an abstraction of the network connection to the application typically through a network socket interface.

At the lower levels of the protocol stack, due to network congestion, traffic load balancing, or unpredictable network behavior, IP packets may be lost, duplicated, or delivered out of order. TCP detects these problems, requests re-transmission of lost data, rearranges out-of-order data, and even helps minimize network congestion to reduce the occurrence of other problems.

TCP uses several mechanisms to achieve high performance and avoid congestive collapse, a gridlock situation where network performance is severely degraded. These mechanisms control the rate of data entering the network, keeping the data flow below a rate that would trigger collapse. They also yield an approximately max-min fair allocation between flows. TCP uses a network congestion-avoidance algorithm that includes various aspects of an additive increase/multiplicative decrease (AIMD) [7] scheme, along with other schemes including slow start and cwnd, to achieve congestion avoidance.

NewReno [13] handles multiple packet losses more effectively by using partial acknowledgments to identify successfully received packets, allowing for the retransmission of multiple lost packets within the same recovery phase and enhancing transmission efficiency. CUBIC [14], designed for high-bandwidth and long-distance networks, employs a cubic function to control the growth of the cwnd, enabling aggressive bandwidth probing when underutilized and conservative growth as it approaches its

maximum value. This allows for faster recovery from packet loss and better scalability in high-speed networks. Unlike loss-based algorithms, BBR [8] periodically measures available bandwidth and RTT to adjust the cwnd and pacing rate. It continuously probes the network to determine the maximum available bandwidth and minimum RTT, adjusting the sending rate to match the bottleneck bandwidth while maintaining low queue occupancy.

Emerging applications that require high bandwidth and low latency as well as diverse Quality of Service (QoS) can be effectively handled by multipath routing. However, the performance of legacy TCP applications in multipath routing scenarios presents new challenges. From the perspective of network companies, they leverage the bandwidth advantages of the Multilink Point-to-Point Protocol (PPP) [32] while ensuring ordered packet delivery by incorporating sufficiently large reorder buffers in network layer devices for packet sorting. However, this approach requires customized hardware and additional overhead.

On the other hand, data centers significantly mitigate the issue of packet disorder by deploying RDMA [19], a data transfer protocol that operates independently of the TCP/IP stack. Meanwhile, deploying programmable switches with RDMA allows routing at a finer granularity and using separate queues to handle disorganized packets [33]. However, the success of this protocol stems from its highly symmetric network topology and characteristics of high bandwidth and low latency.

Considering the complexity and cost of hardware upgrading, many new routing algorithms were developed for TCP to resolve the packet reordering problem [12]. For example, Paxson [28] proposed adjusting DupThresh based on the degree of packet reordering or delaying the initiation of the fast retransmission process that significantly reduces the congestion window. Lee's team [21] employs a static DupThresh, with the receiver delaying the transmission of duplicate ACKs. Similar to the previous method, TCP-BA [3] dynamically increases DupThresh and transmits a new packet for every two duplicate ACKs received. Based on TCP-BA, TCP-RR [39] incorporates DupThresh resetting to prevent unbounded growth and enhances performance by sending the entire congestion window's data during fast recovery. However, the algorithms above still trigger fast retransmission on packet timeouts, significantly degrading TCP transmission performance.

Algorithms developed to overcome lossy wireless networks have inspired solutions to packet reordering in wired multipath networks. TCP-NC [34] uses the RTT increase of TCP-Vegas [5] as a congestion signal but transmits additional data to obfus-

cate packet loss, leading to congestion. TCP-Westwood [24] estimates bandwidth by sampling ACKs to assess network congestion, ignoring duplicate ACKs that trigger fast retransmission but fail to overcome the performance degradation associated with packet timeout. TCP-PR [4] is similar to TCP-Westwood but the series of power calculations for each received ACK is computationally expensive and overly sensitive to RTT variations. TACK [22] attempts to enhance TCP throughput by taming ACK frequency which may tolerate some packet reordering but falls short in addressing sustained reordering. These methods reduce the reliance on duplicate ACKs as a congestion signal and exhibit higher tolerance for out-of-order packets. However, they still fail to fully adapt to the persistent packet loss caused by multipath routing or introduce more complex computations that reduce deployability.

The exponential growth of wireless networks has attracted more research on MPTCP, which can fully leverage the bandwidth advantages of simultaneous access to cellular networks, WLANs, satellite networks, etc. Notably, MPTCP can provide independent subflows for each access method to maintain packet queues, mitigating packet reordering issues more effectively than those associated with multipath routing at the network layer. However, reorder challenges faced by MPTCP at the application layer can be addressed by scheduling algorithms. Simultaneously, based on the relatively stable Round-Trip Time (RTT) of each access network, the ordering of packet arrivals can be optimized through scheduling algorithms [36].

Learning-based congestion control algorithms utilize various machine learning techniques, including supervised, unsupervised, and reinforcement learning (RL). Supervised and unsupervised methods have been widely used to estimate network conditions, but they often fail in real-world scenarios due to their offline training limitations [10, 11]. RL techniques, on the other hand, excel in handling dynamic and complex network states and offer better online learning capabilities [17, 26, 37]. Despite these advantages, learning-based CCAs are still evolving, most current algorithms adjust the cwnd to control the sending rate, which can lead to burstiness in high-speed networks when multiple acknowledgments arrive simultaneously [30]. Additionally, many learning-based CCAs focus on end-to-end rather than network-assisted control [23], and RL-based methods often struggle with time overhead in realistic networks. Therefore, developing robust learning-based CCAs for real-world applications remains a key challenge.

However, these methods have not addressed how to effectively leverage the benefits of multipath routing for legacy TCP applications without incurring performance

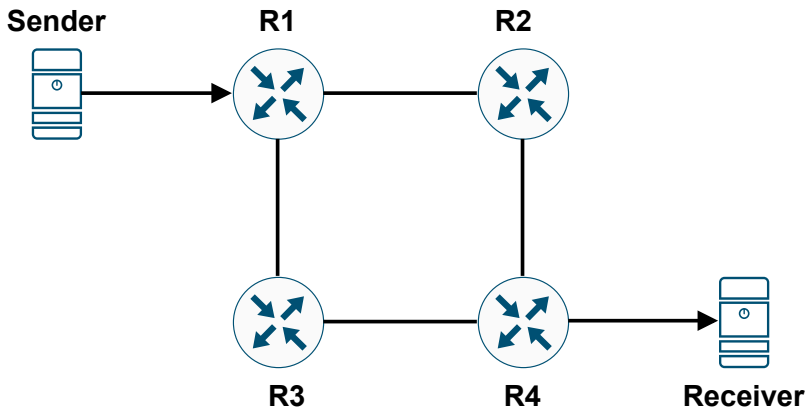


**Figure 2.2: 2 × 2 Grid Topology**

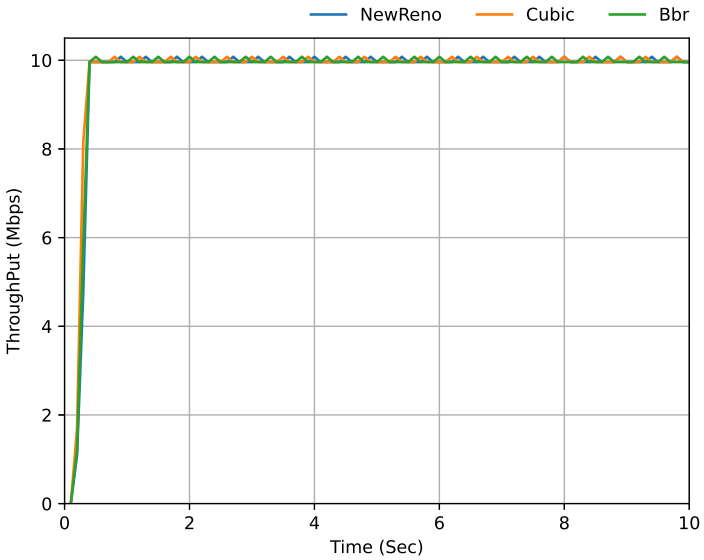


**Figure 2.3: TCP-variants over OSPF**

degradation due to packet reordering. This gap has led us to revisit the fundamentals of congestion control algorithms and to explore a strategy using FastRTO for loss detection, which can dynamically adapt to packet reordering in multipath routing environments.

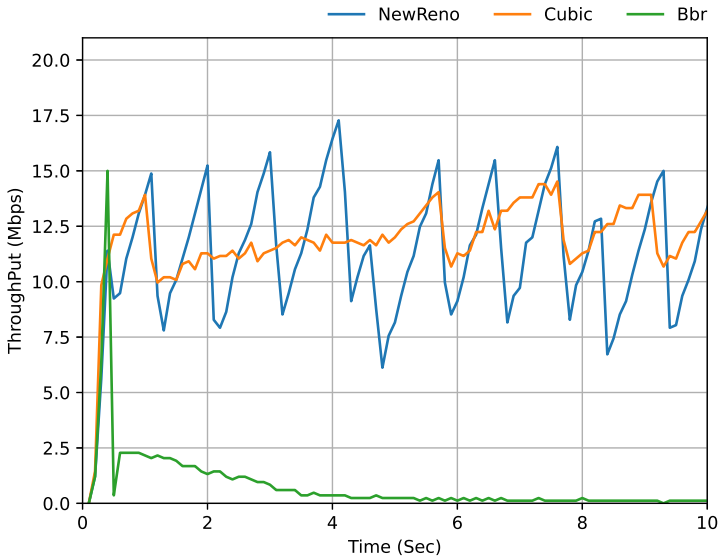


**Figure 2.4: TCP-variants over ECMP**

## 2.3 Problem verification

We simulate to validate TCP variants behave differently using single-path routing and multipath routing in the same network environment. Consider using a TCP application to deliver packets from **Sender** to **Receiver** in a $2 \times 2$ grid topology, as shown in Fig. 2.2. We set the same delay and bandwidth as 10 ms and 10 Mbps for each link, respectively.

Fig. 2.3 shows that different TCP variants have familiar performance in such simple topology and network environments, with throughput converging rapidly to the maximum link bandwidth and remaining stable. Conversely, when using multipath routing, these TCP variants exhibited varying degrees of performance degradation and were unable to fully utilize the maximum path bandwidth, which is twice the link bandwidth. As shown in Fig. 2.4, NewReno experienced greater throughput fluctuations compared to Cubic, but both had average values above 10 Mbps. BBR shows the most severe performance decay, rapidly declining after attempting to explore a maximum throughput of 15 Mbps, resulting in an inability to transmit properly.

We observed a throughput decline in existing congestion control algorithms due to multipath routing. In the following chapter, we will analyze the reasons in detail and propose a congestion control algorithm compatible with multipath routing.

# Chapter 3

# Problem Analysis

Fig. 3.1 summarizes the reasons for generating duplicate ACKs, namely packet reordering, packet loss, and the occasional duplicate ACK transmission resulting from parallel computing [2]. Multipath routing [25], route oscillations [28], and parallel computing are the primary causes of packet reordering, while network congestion, hardware errors, and router deadlock lead to packet loss. These factors occur with varying probabilities. Assuming occasional packet reordering, TCP sets the threshold for triggering fast retransmission to three duplicate ACKs. This setting has been effective for decades in filtering out unnecessary fast retransmissions. However, packet reordering caused by multipath routing can be persistent, continually triggering the congestion window halving mechanism in fast retransmission and significantly reducing TCP throughput. Thus, packet loss and out-of-order delivery, two distinct causes, trigger identical fast retransmissions due to the generation of duplicate ACKs, as shown in Fig. 3.2.

As mentioned in Chapter 2, certain congestion control algorithms aim to delay the fast retransmit process, hoping to receive a non-duplicate ACK before proceeding, or they alter the threshold for triggering fast retransmit with duplicate ACKs to prevent unnecessary retransmissions not caused by packet loss. However, in the context of multipath routing, the degree of packet reordering is related to the delay difference of paths selected making it hard to determine appropriate timeout value and fast retransmission threshold delays for fast retransmission threshold. The presence of dynamic routing protocols diversifies the path, rendering duplicate ACKs threshold adjustments ineffective.

The original RTO algorithm shown in (3.1)–(3.3) was proposed in the 1980s, during a time when network environments predominantly featured low bandwidth

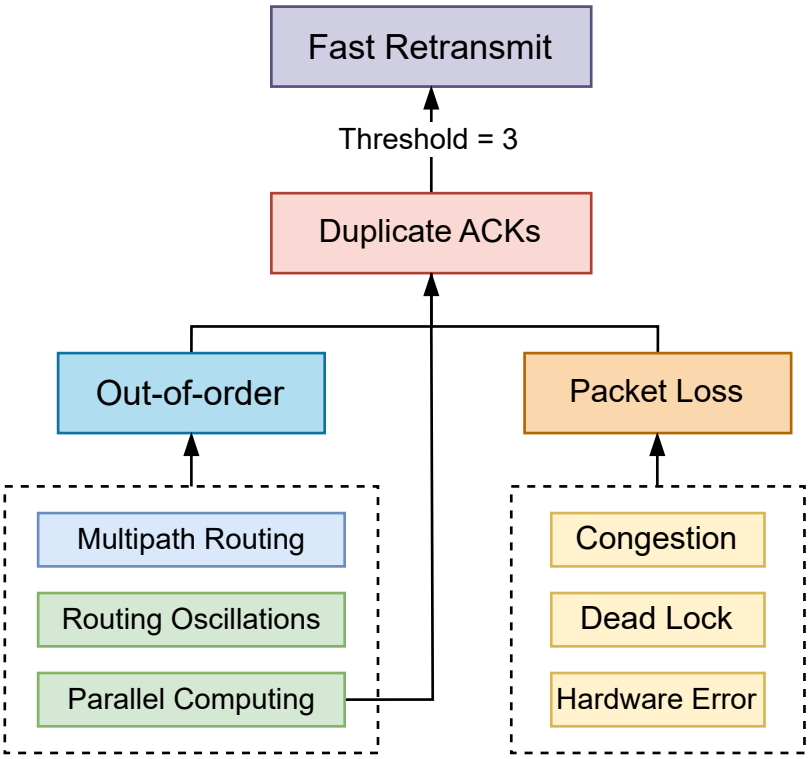


**Figure 3.1: Reason of Fast Retransmit**

and single-path routing. In the exponential weighted moving average formula used at the time, the larger smoothing factors, $\alpha$ and $\beta$ were intended to smooth out sampling noise caused by network fluctuations. However, given the significant improvements in network stability and reliability over the past forty years, these influence factors should be adjusted to accommodate modern algorithmic needs.

$$SRTT = \alpha \times SRTT + (1 - \alpha) \times R \tag{3.1}$$

$$RTTVAR = \beta \times RTTVAR + (1 - \beta) \times |SRTT - R| \tag{3.2}$$

$$RTO = SRTT + 4 \times RTTVAR \tag{3.3}$$

Similarly, the duplicate ACK threshold of three for triggering fast retransmissions, along with the recommended values of $\alpha$ and $\beta$ in the RFCs, and the coefficient of 4 used for $RTTVAR$ in RTO calculations, may not be desirable in different scenarios. As network conditions continue to evolve, there is a pressing need to reassess and recalibrate these parameters to ensure they remain effective and relevant in today's

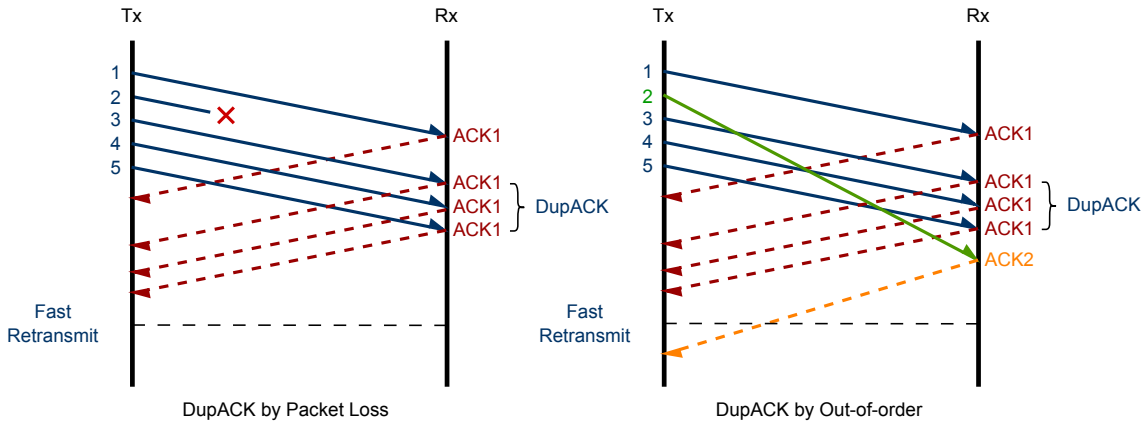


**Figure 3.2: Diverse Causes of Duplicate ACKs**

advanced and dynamic network environments.

The excessive smoothing of $SRTT$ and $RTTVAR$ in the traditional TCP algorithms helps to ignore minor network fluctuations but fails to promptly reflect the changes in RTT caused by different path selections in multipath routing scenarios. This inadequacy becomes particularly problematic when the RTT discrepancies across various paths are substantial, leading to inaccurate RTO estimations and consequently, unreasonable packet timeouts. The coefficient of 4 used in calculating $RTTVAR$, was originally based on the assumption that less than 1% of packets experience delays exceeding four standard deviations from the mean, which should change dynamically as the RTT changes.

In scenarios involving multipath routing, it would be more appropriate for the variability to adapt to changes in paths rather than applying a fixed coefficient that disregards the diversity of path changes and reduces the sensitivity of RTO estimations. Such adaptability would ensure that the RTO calculations remain responsive and accurate, reflecting the true conditions of the network and avoiding unnecessary retransmissions or delays. Thus, a new RTO algorithm is needed to maintain network efficiency and enhance the robustness and reliability of data transmission across complex networks.

# Chapter 4

# Algorithm Design

We propose MPRC, which employs the delay-sensitive FastRTO algorithm to detect packet losses while abandoning duplicate ACKs as the loss signal. This approach addresses the issue of throughput degradation caused by packet reordering in multipath routing environments. We retained the slow start and congestion avoidance mechanisms from NewReno, whose comparatively conservative bandwidth probing algorithm avoids prematurely filling up the queue, thereby preventing packet loss and ensuring the stability of throughput. When multipath routing causes persistent duplicate ACKs, the fast retransmission mechanism will not be triggered, thus avoiding continuous reduction in the size of the congestion window which could affect throughput. Packet loss is detected solely through timeouts, diverging from RFC recommendations by not setting a constant threshold like 500 ms.

When a packet timeout occurs, signaling congestion-induced packet loss, the cwnd is reduced by half instead of being reset to 1. Simultaneously, the slow start threshold is set to half of the new cwnd value. At the same time, the cwnd update mechanism shifts to congestion avoidance. Packet transmission is resumed from the packet that last received an ACK. To prevent consecutive reductions in the cwnd, if a new packet timeout occurs and its sequence number is less than that of the highest sequence number of the packet sent before the last slow start event, we will not trigger slow start again while fast retransmitting the lost packet.

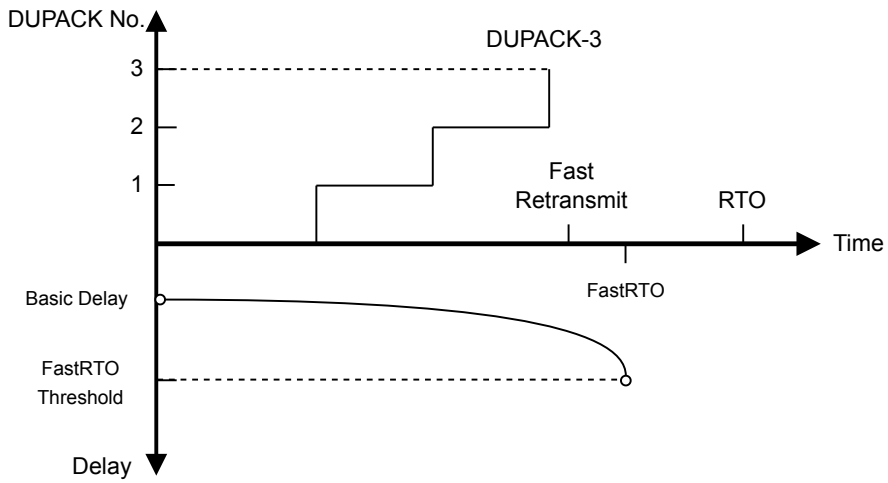


**Figure 4.1: FastRTO vs DUPACKs for Loss Detection**

$$RTT_{\min} = \min(\gamma \times RTT_{\min}, RTT) \tag{4.1}$$

$$RTO_{est} = \max(\gamma \times RTT_{\min}, SRTT + K \times RTTVAR) \tag{4.2}$$

$$FastRTO = \min(RTO_{est}, RTO_{\max}) \tag{4.3}$$

First, we propose reducing the smoothing factors $\alpha$ and $\beta$, which influence the $SRTT$ and $RTTVAR$. This modification aims to enhance the responsiveness of the RTO estimations to changes in RTT measurements, ensuring that variations due to different path conditions are more accurately reflected.

Subsequently, we design a method for estimating the RTO, as shown in (4.1)–(4.3). This process begins by initializing the minor RTT ($RTT_{\min}$) and updating it across the various paths in each round. We introduce the parameter $\gamma$ (where $\gamma > 1$) to prevent $RTT_{\min}$ from converging to a minimal value, which could lead to retransmissions due to slight queuing delays. We then calculate $RTO_{est}$ by comparing $RTT_{\min}$ with a revised RTO calculation, where $K$ varies with RTT diversity. This approach offers flexibility in accommodating the RTT diversity resulting from multipath routing, providing a robust basis for timeout settings.

To prevent the growth of $RTO_{est}$ from exceeding the time range required for packet delivery, we further employ $RTO_{\max}$ as the upper limit, which can be set to 500 ms, as recommended in RFCs. Additionally, $RTO_{\max}$ can be dynamically adjusted based on the service's latency requirements. By comparing $RTO_{\max}$ and $RTO_{est}$, we derive

FastRTO. This calculated value represents a more accurate and dynamic approach to handling packet loss detection and improving the efficiency of data transmission across increasingly complex network architectures.

Fig. 4.1 illustrates that FastRTO and duplicate ACKs are described as distinct mechanisms for loss detection. In traditional congestion control algorithms using single-path routing, three duplicate ACKs as a loss signal detect path congestion earlier than RTO and thus enter the fast retransmission phase to reduce the cwnd and prevent further congestion. In contrast, FastRTO utilizes a dynamically calculated timer for packet loss detection, which is faster than traditional RTO and avoids the performance degradation caused by duplicate ACKs repeatedly triggering fast retransmissions in multipath routing scenarios.

# Chapter 5

# Evaluation, Analysis and Comparisons

To evaluate the performance of MPRC, we implemented the new protocol in ns-3, a discrete-event network simulator for Internet systems. The evaluation was performed on a consumer-grade computer equipped with Ubuntu 22.04, an Intel i7-12700 CPU, 32GB of RAM, and SSD storage. To investigate its performance in a realistic network, we selected two well-known network topologies Abilene and AT&T, as shown in Fig. 5.1 and Fig. 5.2, which are obtained from the Internet Topology Zoo [18] dataset.

The Abilene network topology is a pioneering infrastructure tailored to the needs of advanced research and education. This high-speed backbone network interconnected numerous academic and research institutions across the United States, facilitating collaborative research and innovative endeavors. It is composed of eleven nodes and fourteen edges in a simple topology, where typically only two paths can be found for multipath routing. On the other hand, the AT&T network topology exemplifies cutting-edge telecommunications infrastructure. Compared to Abilene topology, the highly meshed nature of AT&T topology with twenty-five nodes and twenty-six edges facilitates the implementation of multipath routing.

In the Abilene network, data transmission targets the Seattle to Atlanta route; in the AT&T network, it extends from Orlando to San Fernando. The source and destination selection is chosen to maximize the topological features, specifically aiming to increase path delay diversity and enhance path selectability, since ECMP can only find paths with the same cost. Path delays are calculated using latitude and longitude data from the Topology Zoo, combined with the propagation speed in fiber optics,

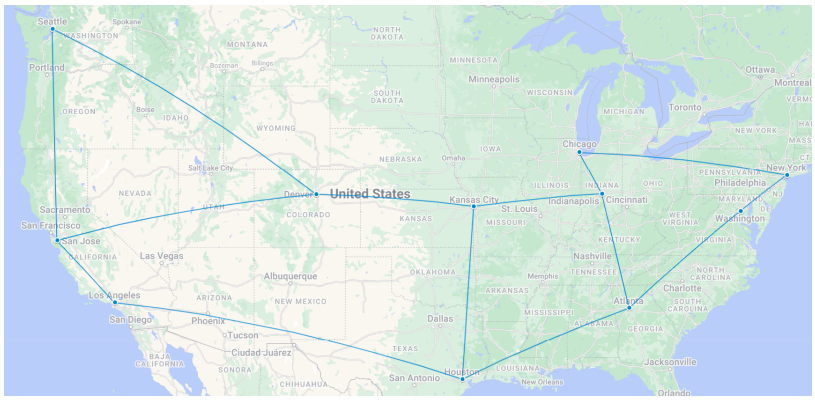


**Figure 5.1: Abilene Topology**

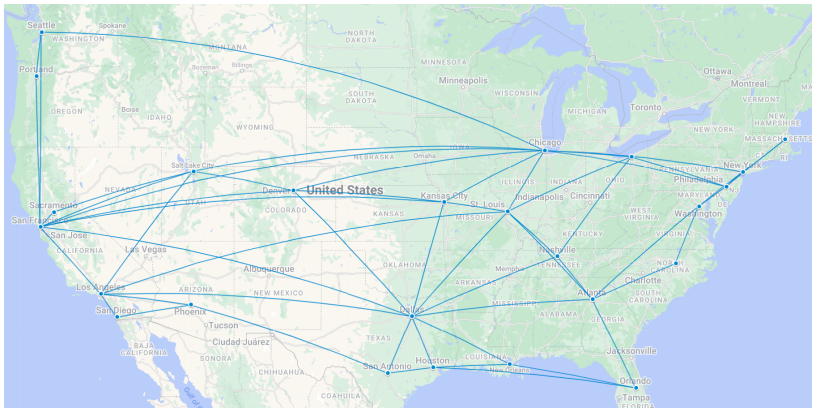


**Figure 5.2: AT&T Topology**

and an additional 1 ms is added to simulate transmission and processing delays. Due to the constraints of the outdated dataset, we assume uniform bandwidth across all backbone network nodes.

Inspired by Cubic [14], we tested the following multiple scenarios for MPRC. These are compatibility with single-path routing, support for multipath routing, tolerance for path error rates, and competitive fairness for multiple applications.

We compare MPRC with NewReno, Cubic, and BBR, using single-path routing protocol OSPF, and equal-cost multipath routing protocol, ECMP. NewReno enhances performance by accelerating retransmissions after packet losses. Cubic improves scalability by modifying the congestion window to a cubic function. BBR optimizes network performance by prioritizing bandwidth and latency through a model-based approach.

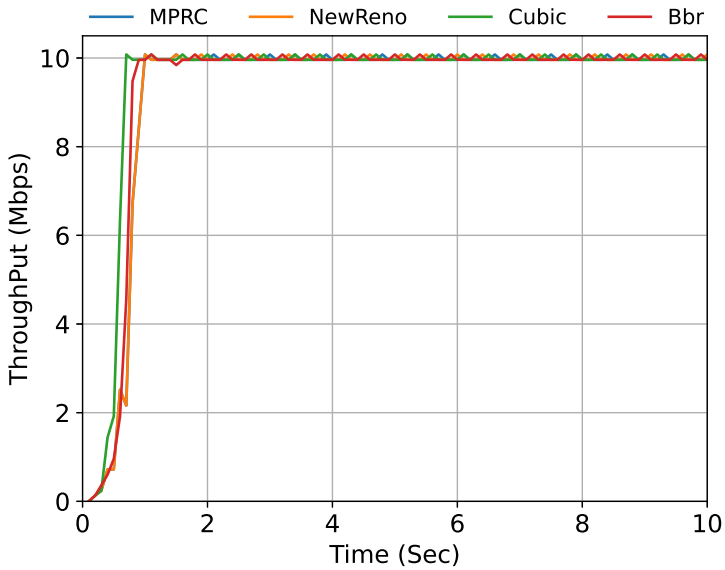


**Figure 5.3: MPRC vs TCP-variants over OSPF - Abilene**

## 5.1 Compatible on OSPF

While designed for multipath routing, MPRC also strives to maintain competitive performance when used in single-path routing scenarios compared to other CCAs. We utilize the TCP-bulk application to generate and send data, simulating the download of large files in a real network environment to evaluate the performance of different congestion control algorithms.

Figures 5.3 and 5.4 illustrate the throughput comparison of various CCAs using the OSPF routing protocol across two different network topologies. These figures provide a comprehensive analysis of how different CCAs perform under varying network conditions.

In both network topologies, the throughput of MPRC is comparable to that of other CCAs, demonstrating its robustness and adaptability. All the algorithms successfully detect the maximum path bandwidth and reach 10 Mbps within one second, highlighting their efficiency in utilizing available resources. Cubic shows a slight advantage in bandwidth convergence speed due to its aggressive bandwidth probing algorithm. This feature allows Cubic to quickly adapt to changes in network conditions and achieve higher throughput more rapidly than other algorithms.

The results underscore the effectiveness of MPRC in maintaining high throughput and efficient bandwidth utilization across different network topologies. This consis-

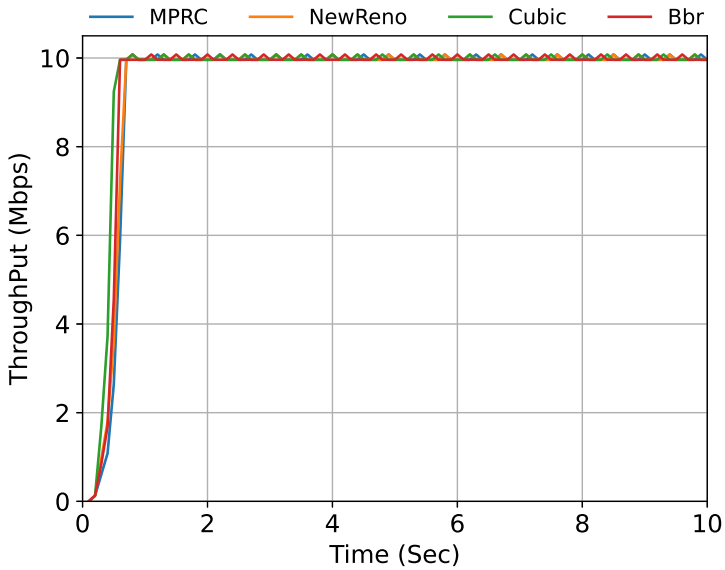


**Figure 5.4: MPRC vs TCP-variants over OSPF - AT&T**

tency is crucial for applications requiring reliable and fast data transmission, ensuring that MPRC can be a viable option alongside established CCAs.

## 5.2 Performance over Multipath Routing

Compared to the previous evaluation experiment, the topology structure and transmission application remain unchanged. We switched the routing protocol from OSPF to ECMP, a load-balancing technique utilized within OSPF to distribute traffic across multiple equal-cost paths. Since paths with identical latency costs are only feasible in highly symmetric network topologies such as data centers, we utilize bandwidth as the equal cost to explore the potential of additional paths for general networks.

Fig. 5.5 and Fig. 5.6 illustrate a performance comparison of TCP variants using multipath routing across network topologies of varying complexities. Regardless of the topology complexity, while there are fluctuations during each bandwidth probing interval, MPRC consistently outperforms other CCAs. MPRC effectively utilizes the bandwidth available through multipath routing, achieving a stable throughput of 20 Mbps with minimal fluctuation.

From Fig. 5.5, we find that on the Abilene topology with multipath routing, NewReno experiences significant, regular throughput fluctuations but its average throughput reaches the maximum bandwidth of single-path routing, which is 10 Mbps.

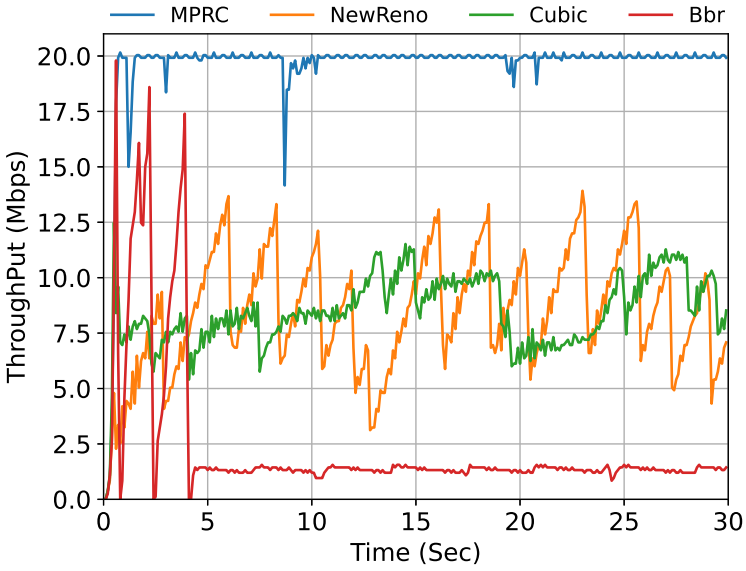


**Figure 5.5: MPRC vs TCP-variants over ECMP - Abilene**

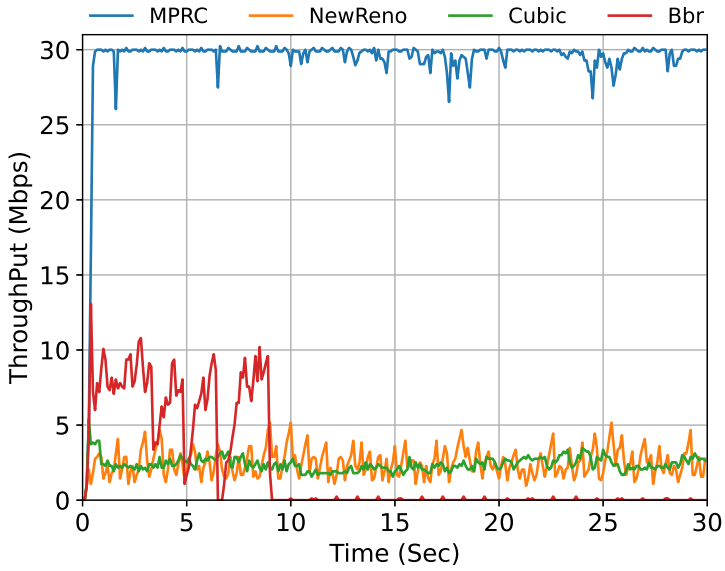


**Figure 5.6: MPRC vs TCP-variants over ECMP - AT&T**

Cubic's average throughput, although lower than 10 Mbps, exhibits less variability compared to NewReno. Meanwhile, BBR, struggling to accurately estimate the Bandwidth-Delay Product in multipath environments, initially reaches the maximum bandwidth but quickly drops below 2.5 Mbps, demonstrating the poorest performance among the compared CCAs.

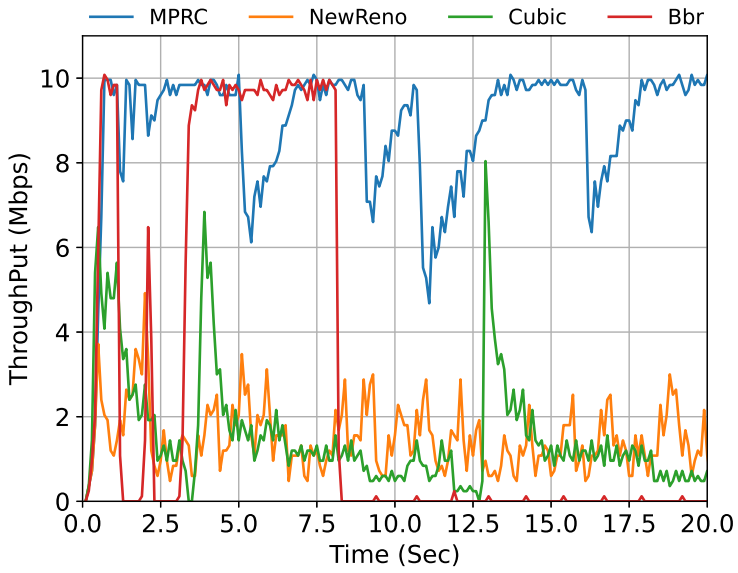


**Figure 5.7: Path Error over OSPF**

In Fig. 5.6, due to increased topological complexity, only MPRC successfully handles TCP transmission over multipath routing and fully unleashes its potential, achieving a total throughput of 30 Mbps. NewReno, Cubic, and BBR struggle with persistent packet reordering, which causes the senders to continuously receive duplicate ACKs and trigger fast retransmissions, resulting in their throughput falling below 5 Mbps, which is also lower than the inter-node bandwidth of 10 Mbps.

## 5.3 Error Tolerance

In the previous simulation, CCAs other than MPRC struggled to function properly on the AT&T topology with multipath routing. In this set of experiments, we exclusively use the Abilene topology and introduce an error rate of 0.01% along the critical paths to test the tolerance of different TCP variants to path errors in both single-path and multipath routing.

Figures 5.7 and 5.8 demonstrate that MPRC exhibits some tolerance to path errors, in contrast to the intolerance shown by other algorithms. In Figure 5.7, the performance of various congestion control algorithms is evaluated using the OSPF routing protocol. Notably, BBR experiences a rapid decrease in throughput after an initial period of high performance. This sudden drop indicates the vulnerability of BBR to path errors, as it fails to maintain stable throughput under these conditions.

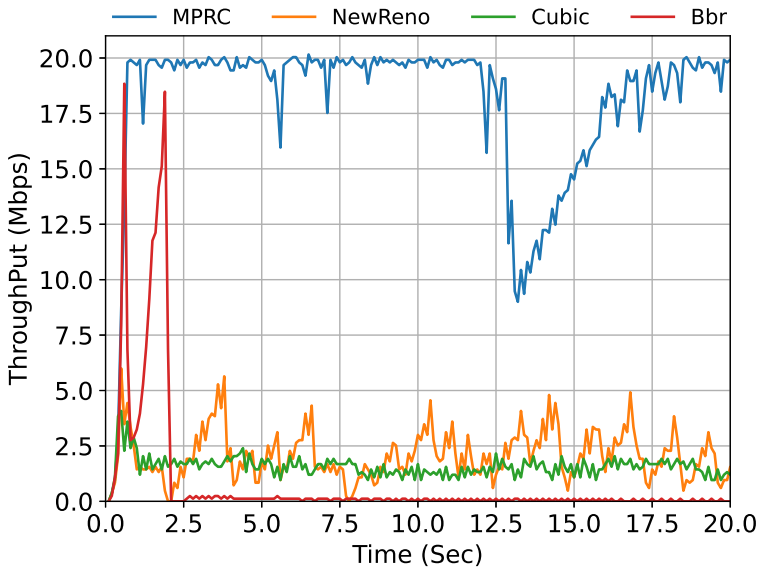


**Figure 5.8: Path Error over ECMP**

Cubic, on the other hand, shows peaks in throughput during bandwidth exploration phases. However, similar to NewReno, the average throughput of Cubic remains well below the maximum path bandwidth, indicating that it cannot consistently utilize the available resources.

Despite the reduced stability, MPRC manages to achieve the maximum bandwidth, showcasing its relative resilience to path errors compared to other algorithms. This ability to maintain higher throughput, even if less stable, highlights the potential advantages of MPRC in environments where path reliability is variable.

In Figure 5.8, the introduction of ECMP routing, which allows some packets to bypass error-prone paths, results in improved stability for MPRC. Under these conditions, the throughput of MPRC is more stable compared to its performance with OSPF, and it consistently reaches the maximum usable bandwidth. The performance of other algorithms, however, deteriorates further under the combined conditions of ECMP routing and increased path error rates. Specifically, the peak throughput of NewReno reaches only half of the path bandwidth, and both NewReno and Cubic see their average throughput drop to below 2.5 Mbps. BBR, as in previous experiments, performs the worst in scenarios involving multipath routing and the introduction of path errors, failing to maintain acceptable throughput levels.

These results underscore the significant challenges that path errors pose to traditional congestion control algorithms and highlight the enhanced capability of MPRC

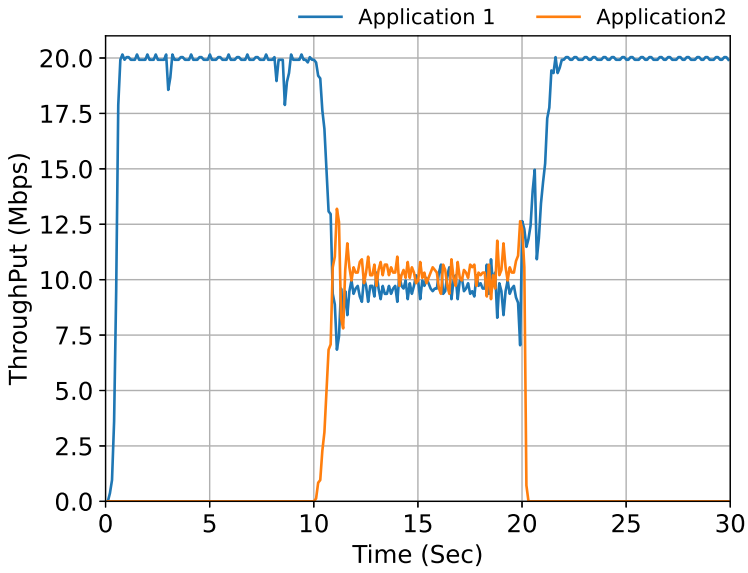


**Figure 5.9: Two Applications Fairness**

to cope with such issues. The ability to achieve and maintain higher throughput under error-prone conditions suggests that MPRC is better suited for multipath routing environments where path reliability cannot be guaranteed. This makes MPRC a strong candidate for deployment in networks that require robust and reliable data transmission despite variable path conditions.

## 5.4 Fairness

We investigated the fairness of multipath routing using MPRC by configuring several TCP bulk applications. Each application dynamically adapts its bandwidth allocation. In the first scenario, depicted in Fig. 5.9, we observe the behavior of two applications. Application 1 operates continuously, while Application 2 starts at around 10 seconds and runs until 20 seconds. Initially, Application 1 fully utilizes the available bandwidth, achieving a throughput of around 20 Mbps. When Application 2 begins, the two applications compete for bandwidth, eventually stabilizing at a fair distribution where each application utilizes about half of the available bandwidth (around 10 Mbps each). This indicates that MPRC effectively manages the bandwidth allocation between two competing applications, ensuring fairness.

In a more complex scenario, shown in Fig. 5.10, we analyze the competition among three applications. Application 1 starts first, followed by Application 2 at 10 seconds,

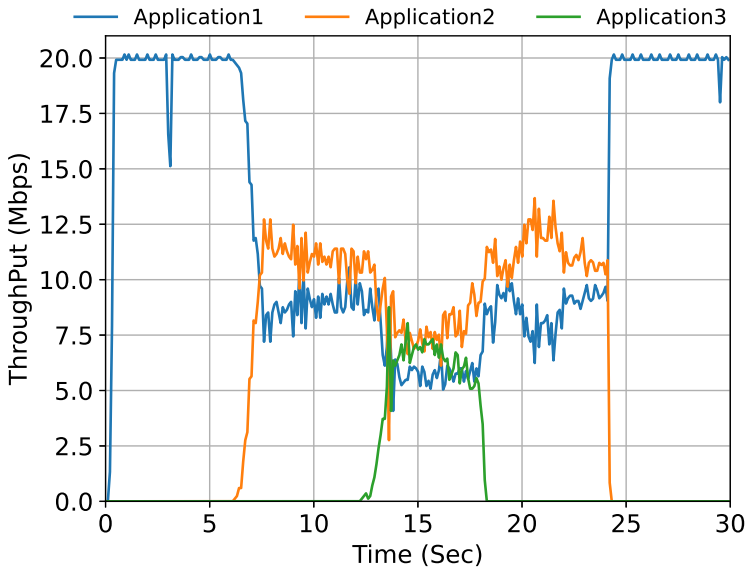


**Figure 5.10: Three Applications Fairness**

and then Application 3 at 15 seconds. Initially, Application 1 again utilizes the full bandwidth. As Application 2 joins, the two applications compete, leading to a reduction in throughput for Application 1, with both stabilizing around 10 Mbps. When Application 3 starts, all three applications compete for the available bandwidth. Although the throughput stability decreases compared to the two-application scenario with some fluctuations, the average bandwidth distribution remains relatively fair. Each application receives a reasonable share of the bandwidth, demonstrating that MPRC maintains fairness even with increased competition.

# Chapter 6

# Conclusions

Facing the challenge of persistent packet reordering caused by multipath routing, which degrades the performance of traditional TCP congestion control algorithms, we introduced the Multipath Routing Congestion control (MPRC) algorithm. The MPRC algorithm prohibits duplicate ACKs from triggering fast retransmissions and employs FastRTO, a more delay-sensitive method, to determine packet loss. Our extensive testing has confirmed its compatibility with the traditional OSPF routing protocol and demonstrated superior transmission performance over other congestion control algorithms in multipath routing scenarios. In addition, MPRC shows good tolerance to a certain level of path error rates and maintains bandwidth fairness relative to itself. These results indicate that MPRC can ensure high-speed data delivery in TCP networks utilizing multipath routing.

As new networking protocols and standards continue to emerge, ensuring compatibility and integration with these advancements will be necessary. We suggest testing MPRC with more complex background traffic and assessing its robustness by introducing not only link errors but also link and node failures. Future work should also investigate the potential for MPRC to be integrated with multipath routing protocols, which could provide more flexible path selection, as well as newly deployed transport layer protocols like QUIC. This will help maintain the relevance and applicability of MPRC as network technologies continue to advance.

Although our simulation results demonstrate the efficacy of MPRC in enabling legacy TCP applications to function efficiently in multipath routing scenarios, it is imperative to validate these findings in real-world network environments. Building large-scale network testing platforms, particularly for dynamic environments such as IoT, satellite networks, and edge computing scenarios, presents a significant challenge.

Addressing this will require the development of advanced testing infrastructures that can accurately simulate real-world conditions.

To enhance the scalability of the algorithm, we propose incorporating learning-based algorithms that can dynamically adapt to varying network conditions and topologies in real-time. These algorithms can dynamically adapt to varying network conditions and topologies in real time, precisely predicting network changes and adjusting the size of the congestion window accordingly. In addition, we should also add comparisons with other formula-based congestion algorithms, which tend to have lower overhead and hardware requirements than learning-based algorithms.

In conclusion, the development and implementation of the MPRC algorithm mark a significant step forward in addressing the limitations of traditional TCP congestion control in multipath routing scenarios. By leveraging the benefits of multipath routing while mitigating its drawbacks, MPRC paves the way for more efficient and reliable data transmission in modern and future TCP networks. Continued research and real-world testing will be crucial in refining the algorithm and extending its applicability across a broader range of network environments.

# Bibliography

[1] Ultra ethernet consortium white paper, 2023. `https://ultraethernet.org/`.

[2] Jon CR Bennett, Craig Partridge, and Nicholas Shectman. Packet reordering is not pathological network behavior. *IEEE/ACM Transactions on networking*, 7(6):789–798, 1999.

[3] Ethan Blanton and Mark Allman. On making tcp more robust to packet reordering. *ACM SIGCOMM Computer Communication Review*, 32(1):20–30, 2002.

[4] Stephan Bohacek, Joao P Hespanha, Junsoo Lee, Chansook Lim, and Katia Obraczka. Tcp-pr: Tcp for persistent packet reordering. In *23rd International Conference on Distributed Computing Systems, 2003. Proceedings.*, pages 222–231. IEEE, 2003.

[5] Lawrence S Brakmo, Sean W O'malley, and Larry L Peterson. Tcp vegas: New techniques for congestion detection and avoidance. In *Proceedings of the conference on Communications architectures, protocols and applications*, pages 24–35, 1994.

[6] Lin Cai, Jianping Pan, Wenjun Yang, Xiangyu Ren, and Xuemin Shen. Self-evolving and transformative protocol architecture for 6g. *IEEE Wireless Communications*, 30(4):178–186, 2022.

[7] Lin Cai, Xuemin Shen, and Jon W Mark. *Multimedia services in wireless internet: modeling and analysis*. John Wiley & Sons, 2009.

[8] Neal Cardwell, Yuchung Cheng, C Stephen Gunn, Soheil Hassas Yeganeh, and Van Jacobson. Bbr: Congestion-based congestion control: Measuring bottleneck bandwidth and round-trip propagation time. *Queue*, 14(5):20–53, 2016.

[9] V. Cerf and R. Kahn. A protocol for packet network intercommunication. *IEEE Transactions on Communications*, 22(5):637–648, 1974.

[10] Ibtissam El Khayat, Pierre Geurts, and Guy Leduc. Improving tcp in wireless networks with an adaptive machine-learnt classifier of packet loss causes. In *NETWORKING 2005. Networking Technologies, Services, and Protocols; Performance of Computer and Communication Networks; Mobile and Wireless Communications Systems: 4th International IFIP-TC6 Networking Conference, Waterloo, Canada, May 2-6, 2005. Proceedings 4*, pages 549–560. Springer, 2005.

[11] Ibtissam El Khayat, Pierre Geurts, and Guy Leduc. Enhancement of tcp over wired/wireless networks with packet loss classifiers inferred by supervised learning. *Wireless Networks*, 16:273–290, 2010.

[12] Nathan Farrington. Multipath tcp under massive packet reordering. *UC San Diego Technical Report*, 2009.

[13] Sally Floyd, Tom Henderson, and Andrei Gurtov. The newreno modification to tcp's fast recovery algorithm. Technical report, 2004.

[14] Sangtae Ha, Injong Rhee, and Lisong Xu. Cubic: a new tcp-friendly high-speed tcp variant. *ACM SIGOPS operating systems review*, 42(5):64–74, 2008.

[15] Christian Hopps. Analysis of an equal-cost multi-path algorithm. Technical report, 2000.

[16] Van Jacobson. Congestion avoidance and control. *ACM SIGCOMM computer communication review*, 18(4):314–329, 1988.

[17] Nathan Jay, Noga Rotman, Brighten Godfrey, Michael Schapira, and Aviv Tamar. A deep reinforcement learning perspective on internet congestion control. In *International Conference on Machine Learning*, pages 3050–3059. PMLR, 2019.

[18] S. Knight, H.X. Nguyen, N. Falkner, R. Bowden, and M. Roughan. The internet topology zoo. *Selected Areas in Communications, IEEE Journal on*, 29(9):1765–1775, October 2011.

[19] Xinhao Kong, Jingrong Chen, Wei Bai, Yechen Xu, Mahmoud Elhaddad, Shachar Raindel, Jitendra Padhye, Alvin R Lebeck, and Danyang Zhuo. Understanding {RDMA} microarchitecture resources for performance isolation. In *20th USENIX Symposium on Networked Systems Design and Implementation (NSDI 23)*, pages 31–48, 2023.

[20] Adam Langley, Alistair Riddoch, Alyssa Wilk, Antonio Vicente, Charles Krasic, Dan Zhang, Fan Yang, Fedor Kouranov, Ian Swett, Janardhan Iyengar, et al. The quic transport protocol: Design and internet-scale deployment. In *Proceedings of the conference of the ACM special interest group on data communication*, pages 183–196, 2017.

[21] Youngseok Lee, Ilkyu Park, and Yanghee Choi. Improving tcp performance in multipath packet forwarding networks. *Journal of Communications and Networks*, 4(2):148–157, 2002.

[22] Tong Li, Kai Zheng, Ke Xu, Rahul Arvind Jadhav, Tao Xiong, Keith Winstein, and Kun Tan. Tack: Improving wireless transport performance by taming acknowledgments. In *Proceedings of the Annual conference of the ACM Special Interest Group on Data Communication on the applications, technologies, architectures, and protocols for computer communication*, pages 15–30, 2020.

[23] Wei Li, Fan Zhou, Kaushik Roy Chowdhury, and Waleed Meleis. Qtcp: Adaptive congestion control with reinforcement learning. *IEEE Transactions on Network Science and Engineering*, 6(3):445–458, 2018.

[24] Saverio Mascolo, Claudio Casetti, Mario Gerla, Medy Y Sanadidi, and Ren Wang. Tcp westwood: Bandwidth estimation for enhanced transport over wireless links. In *Proceedings of the 7th annual international conference on Mobile computing and networking*, pages 287–297, 2001.

[25] Jeffrey C Mogul. Observing tcp dynamics in real networks. *ACM SIGCOMM Computer Communication Review*, 22(4):305–317, 1992.

[26] Xiaohui Nie, Youjian Zhao, Zhihan Li, Guo Chen, Kaixin Sui, Jiyang Zhang, Zijie Ye, and Dan Pei. Dynamic tcp initial windows and congestion control schemes through reinforcement learning. *IEEE Journal on Selected Areas in Communications*, 37(6):1231–1247, 2019.

[27] Yasuhiro Ohara, Shinji Imahori, and Rodney Van Meter. Mara: Maximum alternative routing algorithm. In *IEEE INFOCOM 2009*, pages 298–306. IEEE, 2009.

[28] Vern Paxson. End-to-end internet packet dynamics. In *Proceedings of the ACM SIGCOMM'97 conference on Applications, technologies, architectures, and protocols for computer communication*, pages 139–152, 1997.

[29] Jack Ratcliffe, Francesco Soave, Nick Bryan-Kinns, Laurissa Tokarchuk, and Ildar Farkhatdinov. Extended reality (xr) remote research: A survey of drawbacks and opportunities. In *Proceedings of the 2021 CHI conference on human factors in computing systems*, pages 1–13, 2021.

[30] Seungwan Ryu, Christopher Rump, and Chunming Qiao. Advances in internet congestion control. *IEEE Communications Surveys & Tutorials*, 5(1):28–39, 2003.

[31] Klaus Schneider, Beichuan Zhang, and Lotfi Benmohamed. Hop-by-hop multipath routing: Choosing the right nexthop set. In *IEEE INFOCOM 2020-IEEE Conference on Computer Communications*, pages 2273–2282. IEEE, 2020.

[32] William Simpson. The point-to-point protocol (ppp). Technical report, 1993.

[33] Cha Hwan Song, Xin Zhe Khooi, Raj Joshi, Inho Choi, Jialin Li, and Mun Choon Chan. Network load balancing with in-network reordering support for rdma. In *Proceedings of the ACM SIGCOMM 2023 Conference*, pages 816–831, 2023.

[34] Jay Kumar Sundararajan, Devavrat Shah, Muriel Médard, Michael Mitzenmacher, and Joao Barros. Network coding meets tcp. In *IEEE INFOCOM 2009*, pages 280–288. IEEE, 2009.

[35] Srinivas Vutukury and JJ Garcia-Luna-Aceves. Mdva: A distance-vector multipath routing protocol. In *Proceedings IEEE INFOCOM 2001. Conference on Computer Communications. Twentieth Annual Joint Conference of the IEEE Computer and Communications Society (Cat. No. 01CH37213)*, volume 1, pages 557–564. IEEE, 2001.

[36] Hongjia Wu, Özgü Alay, Anna Brunstrom, Simone Ferlin, and Giuseppe Caso. Peekaboo: Learning-based multipath scheduling for dynamic heterogeneous en-

vironments. *IEEE Journal on Selected Areas in Communications*, 38(10):2295–2310, 2020.

[37] Zhiyuan Xu, Jian Tang, Jingsong Meng, Weiyi Zhang, Yanzhi Wang, Chi Harold Liu, and Dejun Yang. Experience-driven networking: A deep reinforcement learning based approach. In *IEEE INFOCOM 2018-IEEE conference on computer communications*, pages 1871–1879. IEEE, 2018.

[38] Pu Yang, Lin Cai, and Tianfang Chang. Dgr: Delay-guaranteed routing protocol. In *2023 IEEE 20th International Conference on Mobile Ad Hoc and Smart Systems (MASS)*, pages 19–27. IEEE, 2023.

[39] Ming Zhang, Brad Karp, Sally Floyd, and Larry Peterson. Rr-tcp: a reordering-robust tcp with dsack. In *11th IEEE International Conference on Network Protocols, 2003. Proceedings.*, pages 95–106. IEEE, 2003.

[40] Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, et al. A survey of large language models. *arXiv preprint arXiv:2303.18223*, 2023.