Performance Evaluation and Selection of Appropriate Congestion Control Algorithms for MPT Networks

Naseer Al-Imareen, Gábor Lencse
Department of Telecommunications, Széchenyi István University, Győr, Hungary

***Abstract*—Recent academic research highlights a growing interest in multipath technologies, which offer promising solutions to networking challenges in complex environments. This interest is reflected in the emergence of protocols such as Multipath TCP (MPTCP) and Multipath UDP-in-GRE (MPT-GRE). The development of network protocols, particularly various iterations of the Transmission Control Protocol (TCP), has been distinguished by congestion detection and control algorithms, such as HighSpeed, CUBIC, Reno, LP, BBR, and Illinois. This paper evaluates the performance and suitability of these algorithms for multipath MPT-GRE networks under varying conditions, including delay, jitter, and data loss at different transmission speeds (both symmetric and asymmetric). Using StarBED resources, we applied delay, jitter, or packet loss to one of two physical paths to simulate congestion. The results demonstrate that some algorithms, HighSpeed and BBR among them, significantly enhance Quality of Service (QoS) metrics and network throughput in multipath MPT-GRE networks. These findings provide valuable insights into their performance and practical applications.**

***Keywords*—Packet loss; congestion control; MPT-GRE; delay; throughput; jitter**

I. Introduction

The rapid advancement of network applications has increased demands on network infrastructure, posing challenges to its capacity and efficiency. Many emerging applications require high bandwidth and low latency to function effectively. These requirements strain current network capabilities, exacerbating bottlenecks and highlighting the need for robust, efficient data transmission methods [1] [2].

Modern communication technology supports a variety of devices equipped with multiple interfaces, enabling networks and applications to handle complex communication demands. However, the effectiveness of these communication sessions is constrained by the TCP/IP protocol architecture, which, by default, supports only single-session handling. Leveraging multiple network interfaces during communication sessions enhances flexibility and reliability, particularly in addressing network disruptions. By dynamically switching traffic between available paths, communication systems can ensure uninterrupted data transmission, even in the face of link failures, congestion, or performance degradation [3].

This approach significantly improves fault tolerance, load balancing, and network stability, making it an essential feature for modern networking environments that demand high reliability and performance.

To address the growing demand for efficient multipath solutions, numerous methods have been developed, including MPT-GRE [4] and MPTCP [5]. MPT-GRE enables the creation of a virtual tunnel across multiple physical paths, distinguishing it from alternatives like MPTCP and Huawei's Generic Routing Encapsulation (GRE) Tunnel Bonding Protocol. While multipath approaches offer significant advantages, throughput performance in these networks often suffers due to delays and congestion [6].

Various TCP congestion control algorithms, such as HighSpeed, LP, Reno, Vegas, and CUBIC, have been designed to mitigate these challenges. These algorithms detect, control, and preempt congestion, reducing packet loss and delays. Their effectiveness stems from their ability to monitor and manage packet transmission from source to destination. Some algorithms dynamically adjust the congestion window size based on round-trip time (RTT), while others, particularly those optimized for high-bandwidth networks, expand the window to enhance throughput.

Efficient congestion management in multipath networks improves stability, ensures proper packet reordering, and minimizes data loss and delays. Furthermore, the fair allocation of network resources among competing packets prevents bandwidth monopolization, safeguarding throughput and overall network performance [7] [8].

Multipath congestion control is an active area of research focused on maximizing resource utilization by leveraging the available bandwidth across multiple paths while maintaining fairness toward competitive single-path transfers—a constraint referred to as TCP-friendliness. Congestion control techniques are crucial in optimizing network resource use, enabling throughput aggregation, and reducing bandwidth waste.

However, the rise of multipath communication has introduced new challenges. Research has identified side effects, particularly the lack of TCP-friendliness in some implementations. For example, the uncoupled congestion control approach in Multipath TCP (MPTCP) treats each sub flow as an independent TCP connection. This can result in imbalanced resource allocation, where individual subflows dominate bandwidth, leading to unfairness and performance degradation in multipath network environments [8][9].

The MPT-GRE software is designed to enhance data transmission by distributing the load across multiple paths, often achieving throughput capacities close to the combined total of the physical paths [10]. The integration of congestion control algorithms in multipath networks holds significant promise for building robust and efficient network infrastructures. These

algorithms ensure fair resource allocation and improve overall network performance by enhancing fault tolerance, reducing congestion, and increasing throughput through effective traffic management across multiple paths.

Our contributions to this paper are as follows:

1) Evaluation of congestion control algorithms: We analyze multiple congestion control algorithms within multipath MPT-GRE network environments.

2) Performance assessment under diverse network conditions: We assess the performance of these algorithms under various network conditions, including delay, jitter, combined delay and jitter, and packet loss. The evaluation also considers both symmetric and asymmetric transmission speed environments.

3) Identification of throughput-optimizing algorithms: We identify algorithms that significantly enhance network throughput under the evaluated network conditions.

II. Related Work

Many studies have explored congestion control approaches in multipath networks, each focusing on optimizing network performance in various ways. These studies have examined how congestion control algorithms can effectively handle multiple paths, reduce packet loss, minimize latency, and ensure fair resource allocation among packet flows. Additionally, researchers have worked on utilizing algorithms that can adapt dynamically to changing network conditions, such as fluctuations in bandwidth, jitter, and delay, to maximize throughput and minimize congestion.

Szabolcs Szilágyi and Imre Bordán [11] examined the impact of various TCP congestion control algorithms on multipath communication technologies, specifically MPTCP and MPT-GRE. The researchers compared the performance of seven congestion control algorithms (CUBIC, Reno, Illinois, Scalable, Veno, High-Speed, and Vegas) in quad-path IPv4/IPv6 Fast Ethernet environments. Their findings show that CUBIC provided the best performance for MPTCP and MPT-GRE, while Vegas had the lowest performance. The study used these comparisons to emphasize CUBIC's effectiveness as the default algorithm in modern operating systems and aimed to extend the evaluation to more advanced network environments and recent TCP algorithms.

To address the energy consumption challenge in Multipath TCP (MPTCP), the authors [3] analyzed existing congestion control algorithms and identified the key factors influencing energy efficiency. They conducted real-world experiments using the MPTCP Linux kernel and found that energy consumption is closely related to throughput, path delay, and varying network scenarios. To improve energy efficiency, they proposed a congestion control model with a window-increasing factor to direct traffic toward low-delay paths and an energy-aware compensatory parameter for hierarchical Internet topologies. Their experiments confirmed that the enhanced model can increase energy efficiency without compromising transmission performance.

Yu Cao et al. [12] addressed the limitations of coarse-grained load balancing in multipath congestion control, which relies heavily on packet loss as a congestion indicator. They formulated the "Congestion Equality Principle," showing that fair and efficient traffic shifting occurs when flows equalize perceived congestion across all paths. To achieve this, they proposed the delay-based algorithm Weighted Vegas (wVegas), which uses queuing delays for fine-grained load balancing. Simulations showed that wVegas responds faster to congestion changes than loss-based algorithms, improving intra-protocol fairness and reducing packet loss. The study highlights wVegas as a complement to algorithms like TCP-Vegas and TCP-Reno.

Balancing fairness, responsiveness, and window oscillation in Multipath TCP (MPTCP) congestion control is crucial for efficient multipath communication. MPTCP distributes traffic across multiple paths to enhance resource utilization and connection robustness. However, this distribution poses the challenge of adjusting transmission rates across these paths without disrupting other network traffic. To address this, researchers [13] proposed a novel fairness-based congestion control algorithm (FCCA) designed to enhance fairness among subflows while maintaining key performance metrics such as responsiveness and stability of the congestion window. FCCA dynamically adjusts the congestion window for each path based on real-time congestion feedback, optimizing bandwidth usage, improving network performance, and ensuring smooth traffic flow even under varying congestion conditions. The introduction of FCCA marks a significant step toward achieving more equitable and efficient traffic management in MPTCP, contributing to enhanced network performance and fairness in multipath communication scenarios.

III. Background

This section discusses the MPT-GRE multipath network and congestion control algorithms in detail.

A. MPT-GRE Network Technology

The MPT-GRE network is a multipath technology based on the GRE-in-UDP tunnel specification (IETF RFC 8086 [2]). MPT-GRE extends the traditional GRE-in-UDP architecture by supporting multiple physical paths and enhancing network performance through load balancing. Using the UDP source port for hashing distributes traffic more efficiently across numerous equal-cost multipath (ECMP) routes. This architecture shares some similarities with Multipath TCP (MPTCP) in its utilization of multiple paths. Still, it differs significantly in its underlying technology. MPT-GRE relies on UDP at the transport layer, building on GRE-in-UDP for a tunnel IP layer that supports TCP and UDP protocols. Huawei's GRE Tunnel Bonding Protocol has a similar objective but lacks UDP encapsulation, limiting its scalability to just two physical interfaces. In contrast, MPT-GRE's use of GRE-in-UDP offers greater flexibility, enabling a more scalable and robust multipath solution.

The MPT-GRE software architecture, shown in Fig. 1, introduces a logical tunnel layer that works independently of the physical network paths. This unique approach sets MPT apart from traditional TCP/IP protocols by directing application layer data to a tunnel path instead of directly to a physical path. The MPT-GRE software distributes incoming packets from the

logical tunnel interface across available physical paths in this environment. It allows for seamless multipath communication for applications, which only need to interact with the tunnel interface. This mechanism enables efficient traffic redistribution without changing the application's communication model. Additionally, the MPT-GRE software supports the transition to IPv6, as both IP tunnel and path versions can operate independently within the MPT-GRE library. Fig. 2 shows how data packets are transmitted and received in MPT-GRE.

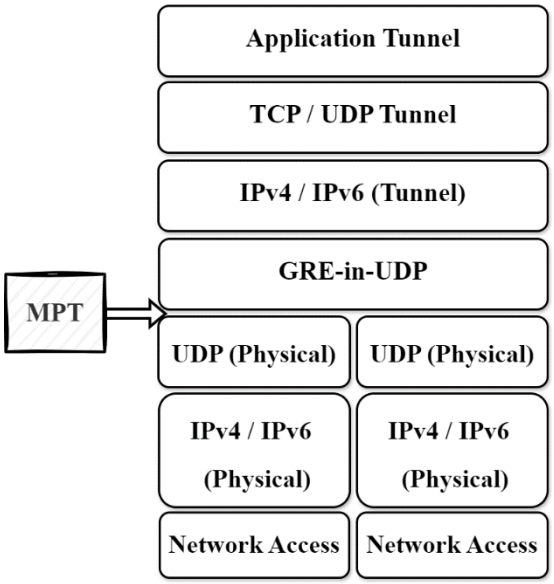

Fig. 1. Conceptual architecture of MPT-GRE [4].

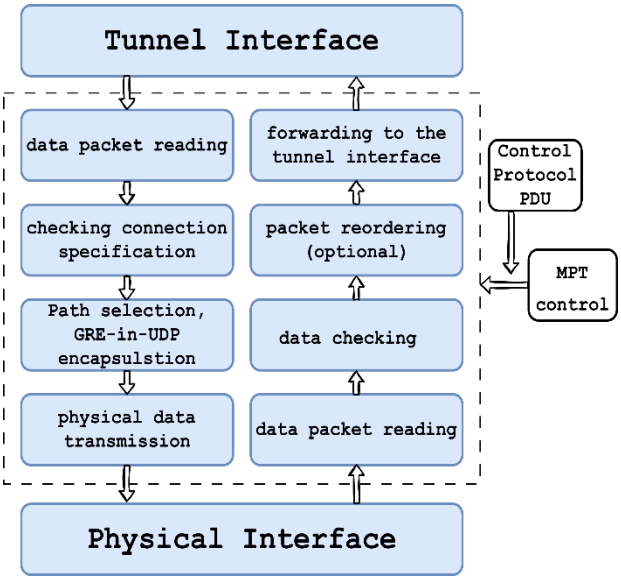

Fig. 2. Theoretical process of the MPT-GRE mechanism [4].

Upon receiving a packet from the tunnel interface, the MPT-GRE software identifies the connection specification and determines traffic distribution across multiple paths. The user data packet is then encapsulated into a GRE-in-UDP data unit. This unit may include optional GRE Sequence Numbers for reordering. The GRE header contains 16 bits of zeros and identifies the protocol type (e.g., 0x0800 for IPv4 or 0x86DD for IPv6). The GRE-in-UDP data unit is encapsulated within a UDP/IP data unit with the destination port set to 4754, which is the GRE-in-UDP port. The packet is then transmitted via the designated physical interface. Upon arrival, the MPT-GRE software verifies the packet by checking the destination port, validating the connection based on the tunnel IP header, and performing checks on the GRE sequence number or GRE Key value, if present. If the packet passes all checks without reordering, it is forwarded to the transport and application layers via the tunnel interface. If reordering is necessary, the packet is temporarily stored in a buffer for reordering before being transmitted [14].

B. Congestion Control Algorithms

Congestion control in computer networks is crucial for ensuring efficient data transfer and preventing issues such as increased latency, packet loss, and reduced throughput [15]. When network demand exceeds available bandwidth or specific data flows dominate resources, congestion control algorithms are essential for maintaining network stability. These algorithms handle transmission data by dynamically adjusting transmission rates and implementing mechanisms like slow start and fast retransmission, which help to alleviate congestion and reduce packet loss. They are designed to respond to dynamic and unpredictable network traffic, employing different strategies to manage congestion by monitoring packet flows and adjusting sending rates accordingly [11][16]. Widely used congestion control algorithms, such as LP, Veno, H-TCP, CUBIC, Reno, Hybla, Vegas, HSTCP, BBR, Westwood, BIC, and Scalable, adjust the rate of transmission to adapt to varying network structures and conditions. For example, CUBIC and Vegas monitor estimated round-trip times (RTT) to detect congestion and dynamically adjust transmission rates, while other algorithms like BBR estimate bandwidth to optimize throughput [17]. These algorithms are generally classified based on their method of detecting and responding to congestion as the following: loss-based, delay-based, hybrid (loss + delay), and bandwidth estimation-based algorithms [18][19][20][21].

1) Loss-based algorithms: Loss-based congestion control algorithms detect network congestion by identifying packet loss. Typically, this occurs when network buffers overflow due to congestion. When packet loss is detected, the sender reduces the congestion window, which is the amount of data allowed in transit without acknowledgment. The sender then gradually increases the congestion window to probe the available bandwidth. These algorithms use packet loss as a congestion signal, detected through duplicate acknowledgments or timeouts. Upon congestion detection, the sender reduces the congestion window, often by half, and slowly increases it using strategies like additive increase multiplicative decrease (AIMD). Examples of loss-based algorithms include Reno, shown in Eq. (1) and (2), a traditional algorithm that reduces the congestion window after detecting packet loss, and CUBIC, shown in Eq. (3), which employs a cubic function to manage window growth but still relies on packet loss for congestion detection.

Reno equations:

$$cwnd = cwnd + \frac{1}{cwnd}\ (\ Additive\ Increase\ ) \quad (1)$$

$$cwnd = \frac{cwnd}{2}\ (\ Multiplicative\ Decrease\ ) \quad (2)$$

CUBIC equation:

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

where C is a scaling element that holds the window growth rate, and t is the time since the last congestion event (loss), whereas $K = \sqrt[3]{\frac{W_{max} \cdot \beta}{C}}$ is the time when the window moves W_{max} again. And W_{max} is the congestion window size at the last congestion event (maximum window size before packet loss). β represents a multiplicative decrease factor, usually set at 0.7.

HighSpeed, Binary Increase Congestion Control (BIC), Scalable, and Hamilton TCP (H-TCP) are advanced loss-based congestion control algorithms designed specifically for high-speed, high-latency networks. HighSpeed aggressively increases the standard congestion window, enhancing throughput in environments with a large bandwidth-delay product (BDP) shown in Eq. (4). BIC uses a binary search method for adjusting the window size, effectively balancing rapid bandwidth probing with a cautious approach to packet loss, as shown in Eq. (5). Scalable adopts a fixed-increment growth strategy based on the current window size, allowing it to maintain high throughput in high-capacity networks, as shown in Eq. (6). On the other hand, H-TCP dynamically adjusts its window growth by monitoring changes in network congestion and round-trip time (RTT), enabling it to respond effectively to fluctuations in network conditions and improve performance in high-BDP scenarios shown in Eq. (7).

HighSpeed equation:

$$cwnd = cwnd + \frac{a(cwnd)}{cwnd} \quad (4)$$

where a ($cwnd$) is an adaptive increase factor that scales with the size of $cwnd$. This factor becomes more significant as $cwnd$ grows.

BIC equation:

$$cwnd = \frac{(cwnd_{max} + cwnd_{min})}{2} \quad (5)$$

where $cwnd_{\max}$ and $cwnd_{\min}$ represent the maximum and minimum congestion window sizes that have been reached so far.

Scalable equation:

$$cwnd = cwnd + a.cwnd \quad (6)$$

where α is a constant scaling factor typically set to a small value, such as 0.01 to 0.1.

H-TCP Equation:

$$cwnd = cwnd + (2 \times (t - T)) \quad (7)$$

where t is the elapsed time since the last congestion event, and T is a constant.

2) Delay-based algorithms: Delay-based algorithms monitor round-trip time (RTT) changes or delays to detect network congestion, allowing them to adjust the sending rate before packet loss occurs. For example, Vegas continuously monitors RTT to adjust the congestion window size based on observed delays, as shown in Equation 8. Hybla compensates for longer RTTs by scaling window growth proportionally to RTT values, improving performance in high-latency networks, as shown in Eq. (9). On the other hand, LP (Low Priority) reduces its window size when delays increase, prioritizing higher-priority traffic and ensuring smooth operation in mixed-traffic environments, as shown in Eq. (10) and Eq. (11).

Vegas equations:

$$\Delta = \left(Expected\ Throughput - Actual\ Throughput\right) \quad (8)$$

$$cwnd = cwnd + \alpha, \quad if\ \Delta < \gamma$$
$$cwnd = cwnd - \beta, \quad if\ \Delta > \delta$$

where α is the increase factor. β is the decrease factor. Δ is the difference between expected and actual throughput. γ and δ are thresholds for adjusting the window.

Hybla equation:

$$\rho = \frac{RTT_{reference}}{RTT_{current}} \quad (9)$$

where $RTT_{reference}$ is a reference for round-trip time (RTT), usually for a low-latency network

LP equation:

$$\Delta = \frac{(RTT_{currnt} - RTT_{min})}{RTT_{min}} \quad (10)$$

If Δ > γ, LP reduces the congestion window as follows:

$$cwnd_{new} = \frac{cwnd_{current}}{2} \quad (11)$$

where the threshold for the increase in RTT is denoted by γ.

3) Hybrid (Loss + Delay) algorithms: These algorithms leverage packet loss and delay to adjust the congestion window. For example, Illinois uses delay and loss metrics to adjust the window size dynamically, enabling more adaptable congestion control. Veno combines Reno and Vegas's strategies, using delay and packet loss signals to optimize the balance between performance and congestion avoidance. Similarly, YEAH uses delay as an early indicator of congestion and relies on packet loss as a secondary, more conservative signal to adjust the transmission rate.

Illinois equation:

$$cwnd = cwnd + \frac{\alpha(delay)}{cwnd} \quad (12)$$

Veno equation:

$$cwnd_{new} = cwnd_{current} + \frac{1}{cwnd_{current}}\ (Add.Increase) \quad (13)$$

YEAH Equation:

$$\mathrm{cwnd}_{\mathrm{increase}} = \frac{\mathrm{MSS}}{\mathrm{RTT}_{\mathrm{min}}} \quad (14)$$

where, MSS stands for Maximum Segment Size, referring to the most significant amount (in bytes) of data that a TCP segment can carry in the payload part of a packet, excluding the TCP and IP headers.

4) Bandwidth estimation-based algorithms: These algorithms estimate the available bandwidth in the network and adjust the sending rate accordingly, optimizing performance based on network capacity. For example, TCP Westwood estimates bandwidth using packet loss and throughput measurements. It adjusts the congestion window dynamically, making it particularly effective in wireless networks where packet loss is common. Similarly, Bottleneck Bandwidth and Round-trip propagation time (BBR) models available bandwidth and round-trip delay to efficiently manage the congestion window and transmission rate.

Westwood equation:

$$\text{BWE} = \frac{\sum ACKed\_data}{\Delta t} \quad (15)$$

where *BWE* is the bandwidth estimate, *ACKed_data* is the amount of acknowledged data, and Δt is the time interval between receiving ACKs.

BBR equation:

$$\text{Throughput} = \frac{\text{Bottleneck Bandwidth}}{RTT_{min}} \quad (16)$$

where Bottleneck Bandwidth is the maximum rate at which data can be transmitted along the path.

IV. Test Environment

A. Hardware Environment Setting

As illustrated in Fig. 3, the test setup consisted of three Dell PowerEdge R650 servers, each equipped with the following:

- CPU: Intel Xeon Gold 6330N 2.2G, 28C/56T, 11.2GT/s, 42M cache, turbo, HT (165W) DDR4-2666 x 2.
- Memory: 32GB RDIMM, 3200MT/s, dual rank 16Gb base x 16 = 512GB.
- Storage: SSD 480GB SSD SATA Read Intensive 6Gbps 512, 2.5inch Hot Plug x 2 (RAID1).
- Network Interface: 25GigE×4.

This experimental design was carefully structured to evaluate the performance of the MPT-GRE multipath network, mainly focusing on the impact of congestion control algorithms on throughput and network efficiency. The experiments were conducted using Ubuntu 22.04.4 LTS (Jammy Jellyfish) / Linux operating system servers. These experiments were designed to evaluate the performance of the MPT-GRE tunnel throughput library and to monitor and prove the effectiveness of various congestion control algorithms.

B. MPT-GRE Configuration Setting

The version of MPT used in this experiment, *mpt-gre-lib64-2019.tar.gz*, is available publicly from GitHub [22]. Two primary configuration files within the MPT installation directory were modified to enable the effective operation of the MPT-GRE multipath system across the network infrastructure.

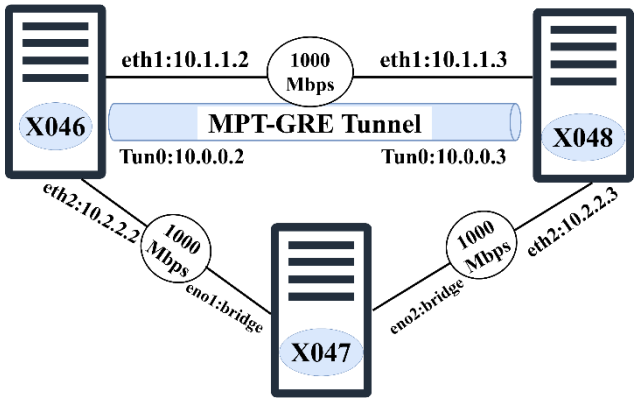

Fig. 3. Experimental topology of the MPT-GRE multipath network.

The first file, conf/interface.conf, contains essential parameters for network interfaces and tunnels. Adjustments were made to specify the Local Command UDP Port Number for managing communication between MPT-GRE and network interfaces; define the Interface Number, maximum transfer unit (MTU), and permissions for remote requests; and establish the tunnel interface name, IPv4 address, and subnet prefix. This file also manages tunnel traffic protocols, with the same configuration mirrored on the second server to maintain consistency in the multipath environment.

The second file, conf/connections/IPv4.conf, manages logical connections and defines transmission paths. Each MPT-GRE tunnel has its own connection file, specifying IP addresses for both tunnel endpoints and using IPv4 encapsulation within an IPv4 GRE tunnel. Previous publications [10][23] have provided detailed information on MPT-GRE configuration settings, and the updated configuration files for this study have also been made available on GitHub [24].

V. Experiments, Results, and Performance Analysis

The primary objective of this study is to confirm the influence of congestion control algorithms on MPT-GRE multipath networks and determine the optimal throughput aggregation value for the tunnel. A detailed evaluation of tunnel throughput is critical for enhancing our understanding of MPT-GRE operations.

Various scenarios were designed for the experiments to ensure precision and effectiveness. A Python script, available on GitHub [25], was employed to automate and facilitate the experimental process over a period of 30 seconds. This script not only enabled frequent execution but also automated the saving of tunnel throughput results, thereby enhancing consistency and repeatability. Additionally, a set of congestion control algorithms was integrated into the Python code and executed using iperf3 [26].

To evaluate the impact of various Quality of Service (QoS) metrics, we used a middle server to adjust traffic parameters with the Linux *tc* command, experimenting with delay, jitter, packet loss, and transmission speed limits. Each set of

congestion control algorithms was tested under various scenarios using the same QoS metrics.

In the first scenario, the transmission speeds were configured with eth1 set at 1000 Mbps, while eth2 varied incrementally from 100 Mbps to 1000 Mbps in steps of 100 Mbps. In the second scenario, similar network conditions were applied; the transmission speeds were reduced tenfold, with eth1 set to 100 Mbps and eth2 ranging from 10 Mbps to 100 Mbps in increments of 10 Mbps.

Due to the large number of results without loss of generality, the remaining transmission cases follow the same approach. The following cases are discussed in detail:

- The symmetrical case: *eth1 = 1000 Mbps and eth2 = 1000 Mbps* under the effect of all QoS metrics.
- The case of asymmetric paths: *eth1 = 100 Mbps* and *eth2 = 10, 20, 30... 100 Mbps* under the effect of delay and packet loss.

A. First Scenario: Analyzing QoS of Symmetrical Paths

1) Assessment impact of delay metric: We applied delays across both server network interfaces, ensuring bidirectional effects on transmitted packets. The delay parameter *x* was set at intervals of 10 ms, 20 ms, 30 ms, 40 ms, and 50 ms, using the *tc* command:

tc qdisc add dev eth1 root netem delay xms

For example, when symmetric paths (eth1=1000 Mbps and eth2=1000 Mbps) are used, the impact of the delay metric on loss-based algorithms, delay-based algorithms, hybrid algorithms, and bandwidth estimation-based algorithms is as is shown in Fig. 4. Analyzing throughput values across different congestion control algorithms under various delay conditions reveals key performance characteristics for each category.

The HighSpeed algorithm demonstrates the throughput achieving 1160 Mbps at 10 ms in the loss-based algorithms category. It sustains a relatively high performance of 202 Mbps even under moderate delays, making it particularly effective in environments with low to medium delays. The Scalable and CUBIC algorithms also perform well in this category, especially at lower delays; the Scalable algorithm reaches 531 Mbps at 20 ms. While CUBIC maintains competitive throughput as delay increases, it is less effective than the HighSpeed algorithm. The BIC algorithm offers balanced performance, achieving high throughput at low delays (1160 Mbps at 10 ms) and proving suitable in mixed delay environments. Therefore, HighSpeed and Scalable are the optimal loss-based algorithms for their peak performance and resilience under medium delays.

In the delay-based algorithms, at lower delay values (10 ms), Hybla achieves a higher throughput of 1180 Mbps compared to LP's 1080 Mbps, demonstrating better performance under minimal delay conditions. However, as the delay increases from 20 ms to 50 ms, LP outperforms Hybla, maintaining higher throughput at greater delays. For example, LP achieves 239 Mbps compared to Hybla's 178 Mbps at 40 ms. In contrast, Vegas experiences a steep drop in throughput as delay increases, making it less suitable for high-delay conditions.

Hence, while Hybla excels in low-delay situations, LP is the better option for higher-delay scenarios.

Within the hybrid (loss + delay) algorithms, Illinois stands out for its high throughput, reaching up to 1150 Mbps at 10 ms and demonstrating adaptability even at higher delays, achieving 215 Mbps at 40 ms. This makes it highly suitable for environments characterized by both loss and delay. Veno and YEAH exhibit medium performance but show sharper reductions in throughput as delay increases.

Therefore, Illinois is the most effective hybrid algorithm, excelling in mixed loss and delay conditions while consistently maintaining high throughput. In contrast, Veno and YEAH perform well in low-delay scenarios but underperform in higher-delay environments.

2) Assessment impact of jitter metric: We measured the throughput of the MPT-GRE tunnel under jitter values of 2 ms, 4 ms, 6 ms, 8 ms, and 10 ms while keeping the delay set to zero. This was done both with and without congestion control algorithms. To enable a more comprehensive assessment, we also combined delay with jitter using the following command:

tc qdisc add dev eth1 root netem delay dms jms

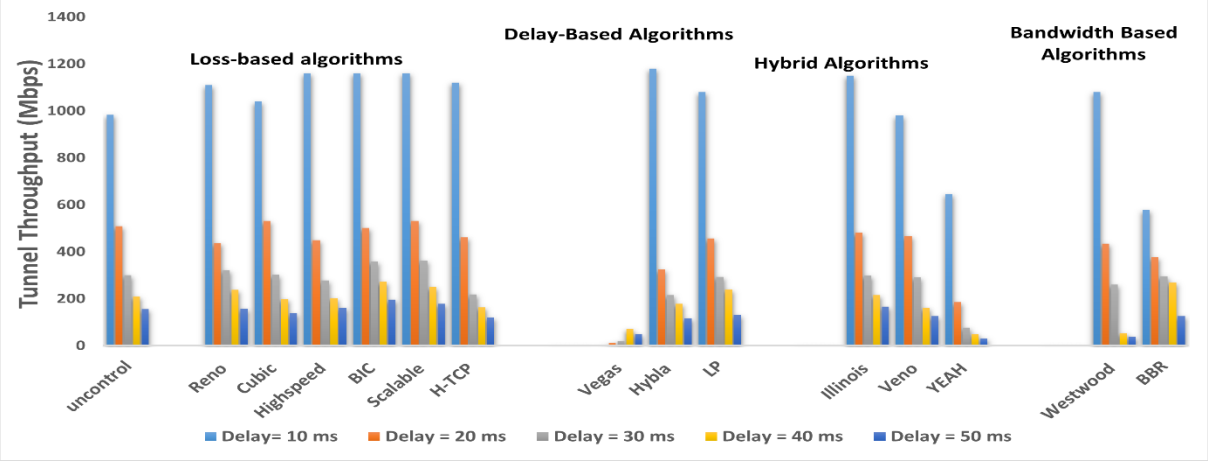

Fig. 4. Throughput performance of the MPT-GRE tunnel under varying delay metrics (eth1 = 1000 Mbps, eth2 = 1000 Mbps)/.

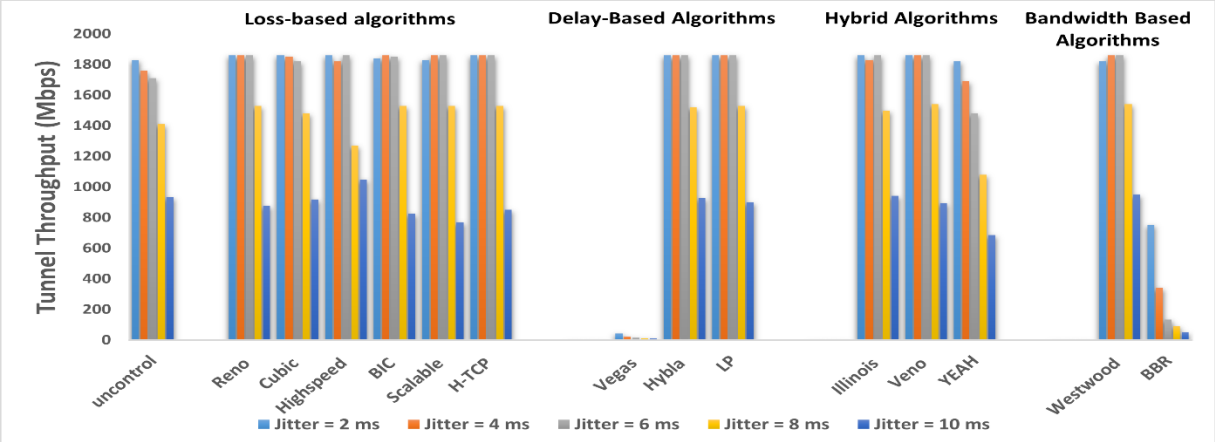

Fig. 5. Throughput performance of the MPT-GRE tunnel under varying jitter metrics (eth1 = 1000 Mbps, eth2 = 1000 Mbps).

The analysis presented in Fig. 5 illustrates the performance trends of various congestion control algorithms based on throughput at different jitter levels. Reno exhibits a consistent throughput of 1860 Mbps in the loss-based algorithms under low to moderate jitter conditions (ranging from 2 to 6 ms). However, its performance significantly declines to 1530 Mbps at 8 ms and drops further to 876 Mbps at 10 ms, indicating that it is sensitive to higher jitter levels. Other algorithms like CUBIC and H-TCP maintain relatively good throughput, showing reductions as jitter increases. Scalable and BIC perform well up to 8 ms, but they experience a significant decline in performance at 10 ms. In contrast, HighSpeed displays stable performance under low jitter levels and maintains a better throughput of 1270 Mbps at 8 ms and 1050 Mbps at 10 ms than other algorithms under higher jitter conditions.

HighSpeed's balanced performance makes it more suitable for environments with increased jitters, while Reno performs better in certain low-jitter situations.

In the delay-based algorithms, throughput remains high for Hybla and LP, maintaining a consistent 1860 Mbps under low to moderate jitter conditions (2–6 ms). However, as jitters increase to 8 ms and 10 ms, their throughput decreases, with Hybla reaching 1520 Mbps and 929 Mbps and LP achieving 1530 Mbps and 900 Mbps, respectively. In contrast, Vegas performs poorly across all jitter levels. Hybla and LP effectively manage moderate jitter, whereas Vegas is unsuitable for jitter-sensitive environments.

Of the hybrid (loss + delay) algorithms, Illinois and Veno show robust throughput despite jitter. Illinois performs slightly better at higher jitter levels, achieving 943 Mbps at 10 ms, while Veno reaches 895 Mbps. However, YEAH is less adaptable in high-jitter scenarios.

Finally, in the bandwidth estimation-based algorithms, there is a considerable difference in the performance of Westwood and BBR under jitter conditions. For example, at 10 ms of jitter, Westwood significantly outperforms BBR, achieving 949 Mbps compared to BBR's 51 Mbps.

Overall, some congestion control algorithms performed better than the Uncontrolled case (the performance baseline without applying any congestion control algorithm). For instance, the HighSpeed congestion control algorithm outperformed the Uncontrolled case in all scenarios.

To better understand the effects and identify the optimal congestion control algorithm, delay and jitter were applied simultaneously, with the jitter value set to 20% of the delay. Fig. 6 analyzes the performance of various congestion control algorithms compared to the Uncontrolled case. Several algorithms performed better than the Uncontrolled throughput, particularly under different delay and jitter conditions.

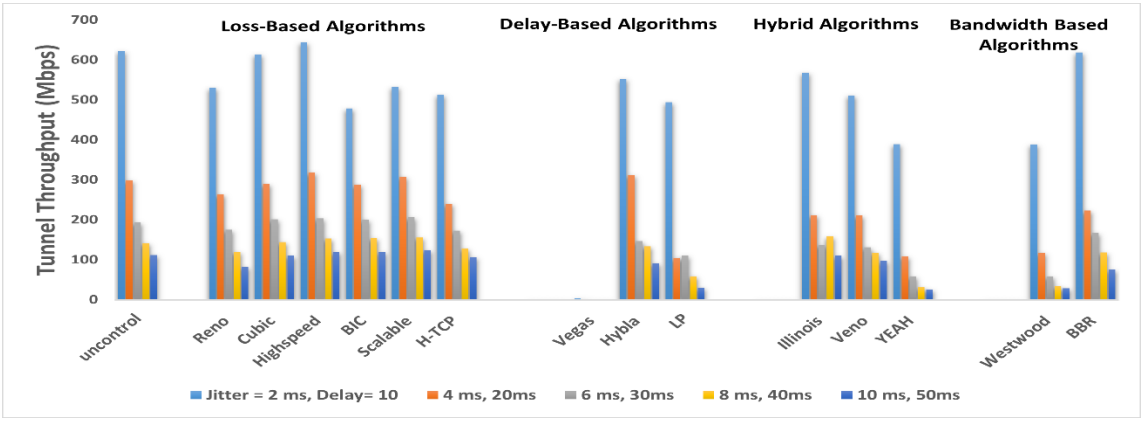

Fig. 6. Throughput performance of the MPT-GRE tunnel under combined delay and jitter metrics (eth1 = 1000 Mbps, eth2 = 1000 Mbps).

In comparison, Reno shows a more rapid and consistent decline in throughput, starting at 530 Mbps and dropping to 82.6 Mbps under higher jitter and delay, indicating a less adaptive response to congestion than the Uncontrolled case. The Uncontrolled case performs better than CUBIC and BIC under low delay and jitter conditions, specifically at 2 ms/10 ms and 4 ms/20 ms. However, as network conditions worsen, these algorithms outperform the Uncontrolled case.

Notably, the BIC algorithm excels under higher delay and jitter conditions, achieving throughput values of 200 Mbps, 154 Mbps, and 120 Mbps at 6 ms/30 ms, 8 ms/40 ms, and 10 ms/50 ms, respectively. Similarly, the Scalable algorithm outperforms the Uncontrolled case as delay and jitter increase, with throughput values of 308 Mbps, 206 Mbps, 156 Mbps, and 124 Mbps under jitter from 4 ms to 10 ms and delay from 20 ms to 50 ms. In contrast, the Uncontrolled case achieves 193 Mbps, 142 Mbps, and 112 Mbps under the same conditions, clearly showing lower throughput than Scalable and BIC as delay and jitter increase.

The HighSpeed algorithm significantly outperforms the Uncontrolled case and other algorithms, achieving 644 Mbps at jitter = 2 ms, delay = 10 ms, and maintaining a higher throughput of 120 Mbps even under the highest jitter and delay conditions. This reflects its superior congestion control capabilities.

Some of these algorithms, such as H-TCP, Hybla, and Illinois, achieved throughput levels close to those in the Uncontrolled case. In contrast, algorithms like Vegas, Westwood, YEAH, Veno, and LP consistently performed worse than the Uncontrolled values. Additionally, their performance declined as delay and jitter increased.

3) Assessment impact of packet loss metric: We tested packet loss conditions with loss rates set at 1%, 2%, 3%, 4%, and 5%. These rates were applied to network interfaces as follows:

tc qdisc add dev eth1 root netem loss x

Fig. 7 illustrates the impact of packet loss on the performance of various congestion control algorithms. Analyzing these results, it is evident that packet loss leads to a decline in throughput for all algorithms as the percentage of loss increases. This decrease is expected, as packet loss commonly results in retransmissions and delays, negatively affecting network throughput.

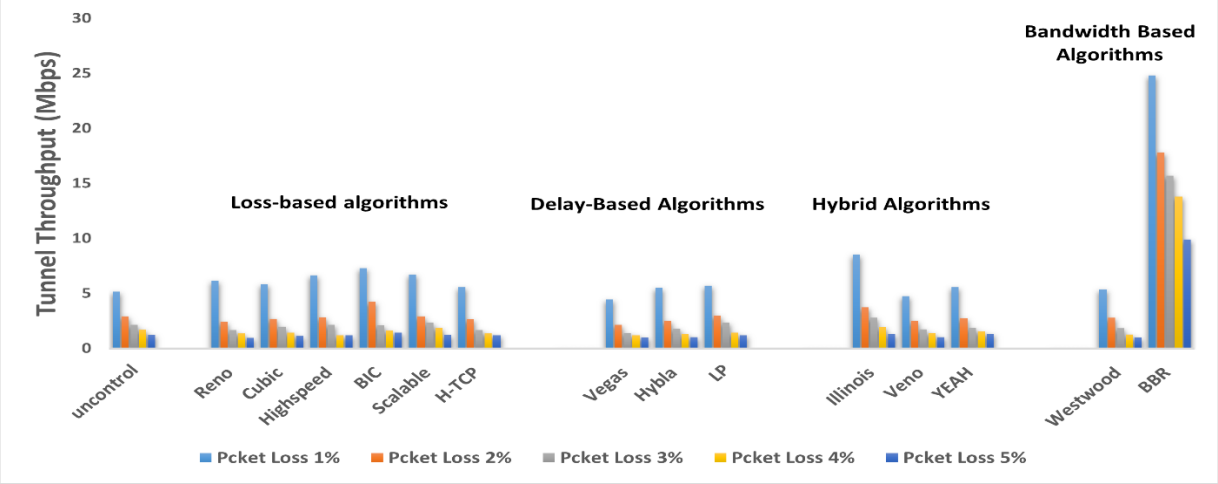

Fig. 7. Throughput performance of the MPT-GRE tunnel under varying packet loss metrics (eth1 = 1000 Mbps, eth2 = 1000 Mbps).

Among the congestion control algorithms examined, BBR is the most resilient to packet loss, consistently achieving significantly higher throughput across all levels of packet loss. Even at a packet loss rate of 5%, BBR maintains a throughput of 9.9 Mbps, far surpassing the performance of other algorithms. BBR's effectiveness minimizes the adverse effects of packet loss on throughput.

In contrast, the Uncontrolled case (i.e., without any congestion control algorithm applied) experiences substantial reductions in throughput as packet loss increases, with values dropping below 1.5 Mbps at 5% packet loss. These results highlight that congestion control algorithms significantly influence MPT-GRE tunnel throughput under varying packet loss conditions. While some algorithms perform better than others in the presence of packet loss, BBR stands out as the most promising option for maintaining higher throughput in challenging conditions.

B. *Second scenario: Analyzing QoS of Appropriate Algorithms*

In the second scenario, we analyzed:

- The HighSpeed congestion control algorithm compared to the Uncontrolled case on symmetric and asymmetric paths, where eth1 is set to 100 Mbps, and eth2 varies from 10 to 100 Mbps in increments of 10 Mbps under the delay metric.
- The BBR algorithm under the packet loss metric.

The results showed a significant increase in tunnel throughput under these specified delay and packet loss conditions. An analysis of the results, as demonstrated in Fig. 8, indicates that the HighSpeed algorithm achieved improved tunnel throughput. The MPT-GRE tunnel throughput performance between the HighSpeed algorithm and the

Uncontrolled case was assessed across symmetric and asymmetric transmission speeds and delay conditions. This analysis highlights how the HighSpeed algorithm enhances tunnel throughput in the MPT-GRE multipath network compared to the Uncontrolled case, particularly under increased network delays.

In the HighSpeed algorithm case, the MPT-GRE tunnel throughput shows a relatively modest decrease as delay increases. For example, at asymmetric transmission speeds (eth1 = 100 Mbps and eth2 = 10 Mbps), the throughput drops only slightly, from 103 Mbps to 102 Mbps, as the delay increases from 10 ms to 50 ms. At symmetric transmission speeds (eth1 = 100 Mbps and eth2 = 100 Mbps), the throughput decreases from 186 Mbps to 184 Mbps. This throughput stability under higher delays suggests that the HighSpeed congestion control algorithm efficiently manages congestion and minimizes throughput loss even as network delay increases.

In contrast, the Uncontrolled case exhibits a more significant degradation in MPT-GRE tunnel throughput with increasing delays. At asymmetric transmission speeds (eth1 = 100 Mbps and eth2 = 10 Mbps), throughput decreases from 102 Mbps at a 10 ms delay to 93.3 Mbps at a 50 ms delay, indicating greater sensitivity to delay. At symmetric transmission speeds (eth1 = 100 Mbps and eth2 = 100 Mbps), throughput drops from 183 Mbps at a 10 ms delay to 137 Mbps at a 50 ms delay. This reduced responsiveness to delay leads to more pronounced degradation in MPT-GRE tunnel throughput.

On the other hand, Fig. 9 illustrates how varying levels of packet loss affect the throughput aggregation of the MPT-GRE tunnel. Under packet loss conditions, the BBR algorithm maintains higher throughput across all transmission speeds compared to the Uncontrolled case. An exception occurs when eth1 = 100 Mbps and eth2 = 10 Mbps at 1% packet loss, where throughput for the Uncontrolled case and BBR is 24.4 Mbps and 23.9 Mbps, respectively—a slight difference.

For example, at transmission speeds of eth1 = 100 Mbps and eth2 = 20 Mbps, BBR achieves a throughput of 33.5 Mbps, while the Uncontrolled case drops to 14.8 Mbps, representing nearly a 56% reduction in MPT-GRE tunnel throughput. This trend persists at all packet loss rates, with BBR consistently mitigating the negative effects of packet loss more effectively than the Uncontrolled case.

As packet loss increases to 5%, both scenarios experience a decline in MPT-GRE tunnel throughput. However, the degradation is significantly more pronounced in the Uncontrolled case, particularly at higher transmission speeds. For instance, when eth1 and eth2 are set to 100 Mbps, throughput in the Uncontrolled case plummets to 1.05 Mbps at 5% packet loss, while BBR maintains a throughput of 7.83 Mbps.

This comparison highlights the resilience of the BBR algorithm, which consistently maintains significantly higher throughput levels than the Uncontrolled case under packet loss conditions.

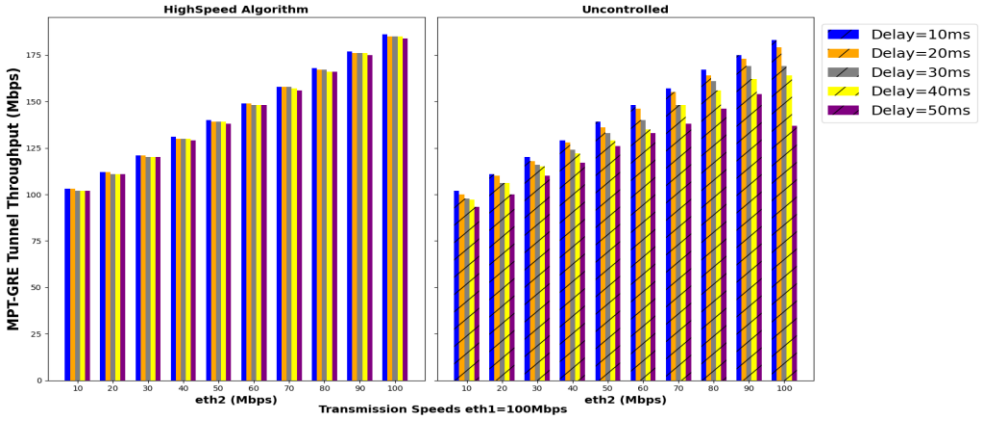

Fig. 8. Impact of the HighSpeed algorithm and the Uncontrolled case on MPT-GRE tunnel throughput under varying delay conditions.

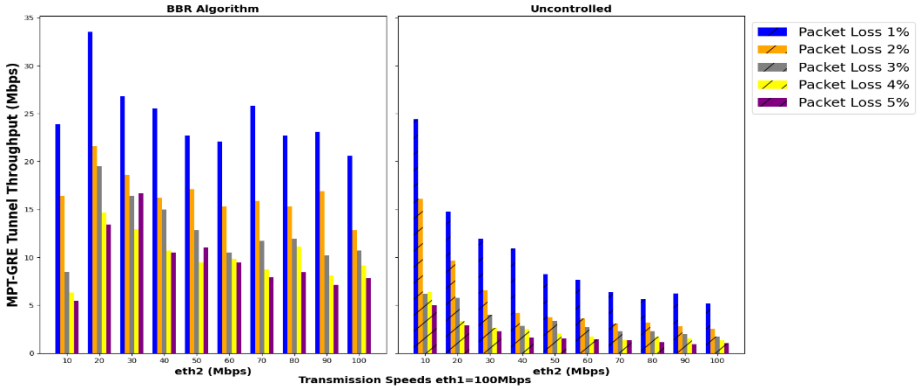

Fig. 9. Impact of the BBR algorithm and the Uncontrolled case on MPT-GRE tunnel throughput under varying packet loss conditions.

C. Determine the Appropriate Algorithm

Upon analyzing the results from the two scenarios, the HighSpeed congestion control algorithm outperforms others when network conditions are affected by delay, jitter, or a combination of both. This algorithm consistently maintains high throughput despite fluctuations in delay and jitter, making it particularly suitable for MPT-GRE multipath environments where these factors are common.

In contrast, the BBR algorithm excels under packet loss conditions, effectively adapting to varying levels of packet loss. Therefore, HighSpeed is the most appropriate choice in scenarios where delay and jitter are the primary challenges, while BBR is the preferred solution for networks where packet loss is the dominant issue.

When considering delay, jitter, and packet loss across various network conditions, the HighSpeed and BBR algorithms consistently outperform other options, making them optimal choices for maximizing throughput in diverse scenarios.

VI. Conclusion

In this study, we evaluated the impact of various congestion control algorithms on tunnel throughput in a multipath MPT-GRE network under different network conditions. The analysis focused on loss-based, delay-based, hybrid, and bandwidth estimation-based algorithms, including HighSpeed, CUBIC, H-TCP, LP, BBR, and Illinois. These algorithms were tested under varying delay, jitter, and packet loss conditions at symmetric and asymmetric transmission speeds.

The results indicated that some algorithms significantly improved MPT-GRE network performance, particularly when specific congestion control mechanisms were applied. HighSpeed significantly improved tunnel throughput as delay and jitter increased, while BBR enhanced throughput under packet loss conditions. BBR consistently outperformed the Uncontrolled case, exhibiting stable performance across various QoS conditions.

Our findings emphasize the importance of selecting appropriate congestion control algorithms for MPT-GRE networks. For example. HighSpeed performs well under delay and jitter, while BBR excels in packet loss scenarios, maximizing throughput and leveraging MPT-GRE's aggregation capabilities. This enables service providers to deliver more reliable and efficient communication solutions.

One of our future works is to create and design a specific congestion control algorithm for the MPT library.

Acknowledgment

The measurements were conducted remotely using the facilities provided by the National Institute of Information and Communications Technology (NICT) StarBED, located at 2–12 Asahidai, Nomi-City, Ishikawa 923-1211, Japan.

The authors thank Bertalan Kovács for reading and commenting on the manuscript.

The authors thank Brant von Goble from Széchenyi István University for proofreading the English-language version of the manuscript.

References

[1] B. Nawaz, K. Mahmood, J. Khan, M. Ul, A. Munir, and M. Kashif, "Congestion Control Techniques in WSNs: A Review," International Journal of Advanced Computer Science and Applications, vol. 10, no. 4, 2019, doi: 10.14569/IJACSA.2019.0100423.

[2] L. Yong, E. Crabbe, X. Xu, and T. Herbert, "GRE-in-UDP encapsulation," 2017.

[3] J. Zhao, J. Liu, H. Wang, C. Xu, and H. Zhang, "Multipath Congestion Control: Measurement, Analysis, and Optimization From the Energy Perspective," IEEE Transactions on Network Science and Engineering, vol. 10, no. 6, pp. 1–12, 2023, doi: 10.1109/TNSE.2023.3257034.

[4] B. Almási, G. Lencse, and S. Szilágyi, "Investigating the multipath extension of the GRE in UDP technology," Computer Communications, vol. 103, pp. 29–38, May 2017, doi: 10.1016/j.comcom.2017.02.002.

[5] A. Ford, C. Raiciu, M. Handley, O. Bonaventure, and C. Paasch, "TCP Extensions for Multipath Operation with Multiple Addresses," Mar. 2020. doi: 10.17487/RFC8684.

[6] Y. Thomas, G. Xylomenos, and G. C. Polyzos, "Multipath congestion control with network assistance," Computer Communications, vol. 153, pp. 264–278, 2020, doi: https://doi.org/10.1016/j.comcom.2020.01.071.

[7] Ł. Łuczak, P. Ignaciuk, and M. Morawski, "Experimental Assessment of MPTCP Congestion Control Algorithms for Streaming Services in Open Internet," in FedCSIS (Communication Papers), Oct. 2023, pp. 359–364, doi: 10.15439/2023F9991.

[8] H. Moradiya and K. Popat, "Evaluating TCP Performance with RED for Efficient Congestion Control," in International Conference on Advancements in Smart Computing and Information Security, Springer, 2024, pp. 403–414.

[9] J. Zhang, Z. Yao, Y. Tu, and Y. Chen, "A Survey of TCP Congestion Control Algorithm," in 2020 IEEE 5th International Conference on Signal and Image Processing (ICSIP), 2020, pp. 828–832, doi: 10.1109/ICSIP49896.2020.9339423.

[10] N. Al-Imareen and G. Lencse, "Effect of Path QoS on Throughput Aggregation Capability of the MPT Network Layer Multipath Communication Library," Infocommunications journal, vol. 15, no. 2, pp. 14–20, 2023, doi: 10.36244/ICJ.2023.2.3.

[11] S. Szilágyi and I. Bordán, "The effects of different congestion control algorithms over multipath fast ethernet IPv4/IPv6 environments," CEUR Workshop Proceedings, vol. 2650, pp. 341–349, 2020, [Online]. Available: https://api.semanticscholar.org/CorpusID:221662730.

[12] Y. Cao, M. Xu, and X. Fu, "Delay-based congestion control for multipath TCP," Proceedings - International Conference on Network Protocols, ICNP, pp. 1–10, 2012, doi: 10.1109/ICNP.2012.6459978.

[13] R. Melki, M. M. Mansour, and A. Chehab, "A fairness-based congestion control algorithm for multipath TCP," IEEE Wireless Communications and Networking Conference, WCNC, vol. 2018-April, pp. 1–6, 2018, doi: 10.1109/WCNC.2018.8377078.

[14] G. Lencse, S. Szilagyi, F. Fejes, and M. Georgescu, "MPT Network Layer Multipath Library," 2021. [Online]. Available: https://datatracker.ietf.org/doc/html/draft-lencse-tsvwg-mpt-10.

[15] M. B. M. Kamel, I. Ahmed Najm, and A. Khalaf Hamoud, "Congestion Control Prediction Model for 5G Environment Based on Supervised and Unsupervised Machine Learning Approach," IEEE Access, vol. 12, pp. 91127–91139, 2024, doi: 10.1109/ACCESS.2024.3416863.

[16] R. Al-Saadi, G. Armitage, J. But, and P. Branch, "A Survey of Delay-Based and Hybrid TCP Congestion Control Algorithms," IEEE Communications Surveys & Tutorials, vol. 21, no. 4, pp. 3609–3638, 2019, doi: 10.1109/COMST.2019.2904994.

[17] H. Jamal and K. Sultan, "Performance analysis of TCP congestion control algorithms," International Journal of Computers and Communications, vol. 2, no. 1, pp. 30–38, 2008, [Online]. Available: http://w.naun.org/multimedia/UPress/cc/cc-27.pdf.

[18] S. Patel, Y. Shukla, N. Kumar, T. Sharma, and K. Singh, "A Comparative Performance Analysis of TCP Congestion Control Algorithms: Newreno, Westwood, Veno, BIC, and Cubic," in 2020 6th International Conference on Signal Processing and Communication (ICSC), Mar. 2020, pp. 23–28, doi: 10.1109/ICSC48311.2020.9182733.

[19] A. Roy, J. L. Pachuau, and A. K. Saha, "An overview of queuing delay and various delay based algorithms in networks," Computing, vol. 103, no. 10, pp. 2361–2399, Oct. 2021, doi: 10.1007/s00607-021-00973-3.

[20] M. A. Yousuf, M. M. Islam, M. S. Hosen, and M. L. Ali, "Round-Trip Time and Available Bandwidth Estimation Based Congestion Window Reduction Algorithm of MPTCP in Lossy Satellite Networks," Journal of Physics: Conference Series, vol. 1624, no. 4, p. 042024, Oct. 2020, doi: 10.1088/1742-6596/1624/4/042024.

[21] R. Gonzalez, J. Pradilla, M. Esteve, and C. E. Palau, "Hybrid delay-based congestion control for multipath TCP," in 2016 18th Mediterranean Electrotechnical Conference (MELECON), Apr. 2016, pp. 1–6, doi: 10.1109/MELCON.2016.7495389.

[22] F. Fejes, "MPT – multi-path tunnel," precompiled version can be downloaded from:, 2019. http://github.com/spyff/mpt.

[23] N. Al-Imareen and G. Lencse, "On the Impact of Packet Reordering in MPT-GRE Multipath Networks," in 2023 46th International Conference on Telecommunications and Signal Processing (TSP), Jul. 2023, pp. 82–86, doi: 10.1109/TSP59544.2023.10197737.

[24] N. Al-Imareen and G. Lencse, "MPT connections files," 2023. [Online]. Available: https://github.com/NaseerAJabbar/MPT_Connections_files.

[25] N. Al-Imareen and G. Lencse, "Implement congestion control algorithms using iperf3 for MPT-GRE multipath network," 2024. https://github.com/NaseerAJabbar/Congestion-Control-Algorithms.

[26] K. P. Jon Dugan, Seth Elliott, Bruce A. Mah, Jeff Poskanzer, "Iperf3 documentation," 2018. https://iperf.fr/iperf-doc.php.