future internet

Article

Evaluating MPTCP Congestion Control Algorithms: Implications for Streaming in Open Internet

Łukasz Piotr Łuczak, Przemysław Ignaciuk * and Michał Morawski

Institute of Information Technology, Lodz University of Technology, Politechniki 8, 93-590 Łódź, Poland; lukasz.luczak@dokt.p.lodz.pl (Ł.P.Ł.); michal.morawski@p.lodz.pl (M.M.)
* Correspondence: przemyslaw.ignaciuk@p.lodz.pl

Abstract: In today's digital era, the demand for uninterrupted and efficient data streaming is paramount across various sectors, from entertainment to industrial automation. While the traditional single-path solutions often fell short in ensuring rapid and consistent data transfers, Multipath TCP (MPTCP) emerges as a promising alternative, enabling simultaneous data transfer across multiple network paths. The efficacy of MPTCP, however, hinges on the choice of appropriate congestion control (CC) algorithms. Addressing the present knowledge gap, this research provides a thorough evaluation of key MPTCP CC algorithms in the context of streaming applications in open Internet environments. Our findings reveal that BALIA stands out as the most suitable choice for MPTCP streaming, adeptly balancing waiting time, throughput, and Head-of-Line blocking reduction. Conversely, the wVegas algorithm, with its delay-centric approach, proves less adequate for multipath streaming. This study underscores the imperative to fine-tune MPTCP for streaming applications, at the same time offering insights for future development areas and innovations.

Keywords: MPTCP; congestion control; streaming applications; tactile internet

check for updates

Citation: Łuczak, Ł.P.; Ignaciuk, P.; Morawski, M. Evaluating MPTCP Congestion Control Algorithms: Implications for Streaming in Open Internet. *Future Internet* **2023**, *15*, 328. https://doi.org/10.3390/fi15100328

Academic Editors: Panagiotis Papageorgas, Dimitrios Piromalis and Dionisis Kandris

Received: 9 September 2023
Revised: 30 September 2023
Accepted: 2 October 2023
Published: 4 October 2023

1. Introduction

1.1. IP Systems in Modern Connectivity

The advent of Internet Protocol (IP)-based systems heralded a transformative era in the realm of digital communication. Conceived to serve the needs of traditional telecommunications, these systems have transcended their original scope and have become ubiquitous across diverse sectors, including healthcare, industrial automation, entertainment, Internet of Things (IoT), and tactile Internet [1]. This widespread adoption can be attributed to the unparalleled flexibility and scalability of IP networks. Unlike specialized solutions that may offer superior Quality of Service (QoS)—such as guaranteed minimum bandwidth, fault tolerance, or maximum latency—IP networks offer a more universal approach. Their architecture is designed for ease of expansion, requiring only a generic connection to the network and leveraging dynamic routing capabilities for data transport [2]. However, this universality comes at a price. IP networks have long grappled with challenges related to transmission quality, especially in the context of streaming services. While protocols like User Datagram Protocol (UDP) and Real-Time Transport Protocol (RTP) are available for time-sensitive transmissions, they often fall short in the face of stringent security requirements or application-specific constraints [3]. As a result, the Transmission Control Protocol (TCP) remains the preferred choice for data transfer in the open Internet, despite its limitations in providing a high-quality service [4].

The economic benefits of IP-based systems are also noteworthy. Their modular nature and ease of integration make them a cost-effective solution for both small-scale and large-scale deployments. This economic advantage has been a significant driver in the widespread adoption of IP-based systems, even in the sectors where specialized network solutions might offer better performance metrics [1]. As one delves deeper into the complexities and potentials of IP-based systems, particularly in the context of streaming applications and multipath transmission, it is beneficial to understand their foundational role in shaping the digital landscape. The subsequent sections of this paper aim to build upon this foundational remark, exploring the specifics of Multipath TCP (MPTCP) and related congestion control (CC) algorithms in the area of streaming solutions [5,6].

The proliferation of mobile devices has added a new layer of complexity to the already intricate world of IP-based systems. Smartphones, tablets, and other portable gadgets have become commonplace, serving as essential tools for communication, entertainment, and professional activities. However, their widespread use has also given rise to a multitude of questions that traditional IP-based systems are ill-equipped to answer [1,3]. One such pressing issue is the variability of link parameters. As mobile devices move from one location to another, they often need to switch networks seamlessly. While the logical IP address of the device may remain unmodified, the underlying link parameters can change dramatically. This variability poses significant challenges for maintaining a consistent QoS [2,7]. Adding to this complexity, modern mobile devices are frequently equipped with multiple network interfaces, such as cellular (LTE, 5G) and Wi-Fi. These interfaces have distinct characteristics, including bandwidth, latency, and reliability. The need to manage these diverse interfaces effectively further complicates the objectives of mobile connectivity [8]. External factors, such as interference from other users and devices limit the range of services that can be reliably offered [3]. To mitigate these issues, researchers and practitioners have proposed the simultaneous use of multiple transmission channels across different physical interfaces [9], such as MPTCP. However, the implementation of MPTCP and the associated CC algorithms in the context of mobile and *streaming applications* remains mostly unexplored, which motivates this work.

1.2. Multipath Data Transfer

The concept of multipath transmission is not new; however, its practical implementation has been fraught with challenges. The idea of utilizing multiple transmission channels has been proposed as an answer to the restrictions of traditional IP-based systems for devices equipped with multiple physical interfaces [9,10]. Early attempts to realize this idea often faltered due to the complexities involved in managing those interfaces and ensuring a consistent QoS [2,7]. However, the situation changed with the introduction of MPTCP, a promising extension of the traditional TCP protocol designed specifically for multi-interface traffic [5,11]. MPTCP offers a more flexible and robust approach to data transmission, allowing for the simultaneous use of multiple, diverse network paths. This multiplicity not only enhances the reliability of data transmission but also provides an opportunity for load balancing, thereby optimizing the use of network resources [12].

One of the most welcome aspects of MPTCP is its openness to innovation. The reference implementation of the protocol addresses only the general aspects of its behavior, leaving room for potential advancements in its implementation [13]. This flexibility is particularly relevant when it comes to the choice of CC algorithms, which play a pivotal role in determining the efficiency and fairness of data transmission. However, the design of MPTCP and its associated CC algorithms has primarily focused on boosting efficiency without compromising fairness [5,6]. This line of reasoning often left other aspects, such as meeting delay constraints, vital for streaming applications and tactile internet [7,14], unfulfilled.

This paper aims to fill the existing gap in the research on the effective use of MPTCP for streaming services. Specifically, we investigate the suitability of various CC algorithms for this purpose through experiments in the open Internet. One should note that streaming applications have unique requirements that set them apart from other types of data transmission. Even though they often employ adaptive data compression methods, their primary concern is not necessarily a high bandwidth, but rather low latency, low error rate, and negligible jitter [2,7]. In the context of multipath transmission, due to the need for stream synchronization at the receiver, meeting these unique requirements is not straightforward. The traditional MPTCP CC algorithms, designed primarily to enhance efficiency and fairness, perform below expectations when it comes to managing the delay and its fluctuation [6,15–17]. The current works in the MPTCP context address the streaming performance from the perspective of schedulers, thus emphasizing the allocation of resources rather than the behavior of CC algorithms [12,18]. Taking into account the profound impact the CC algorithms may exert on the QoS experienced by end-users [19,20], it is a significant oversight this paper aims to mitigate.

1.3. MPTCP Framework Explained

The MPTCP extends the capabilities of the standard TCP by enabling the use of multiple network paths to convey data between communicating parties, as illustrated in Figure 1. When a data transfer request is initiated, the standard TCP handshake process is employed to establish a connection. During this phase, both parties determine if they support a compatible version of the MPTCP protocol [5,11].

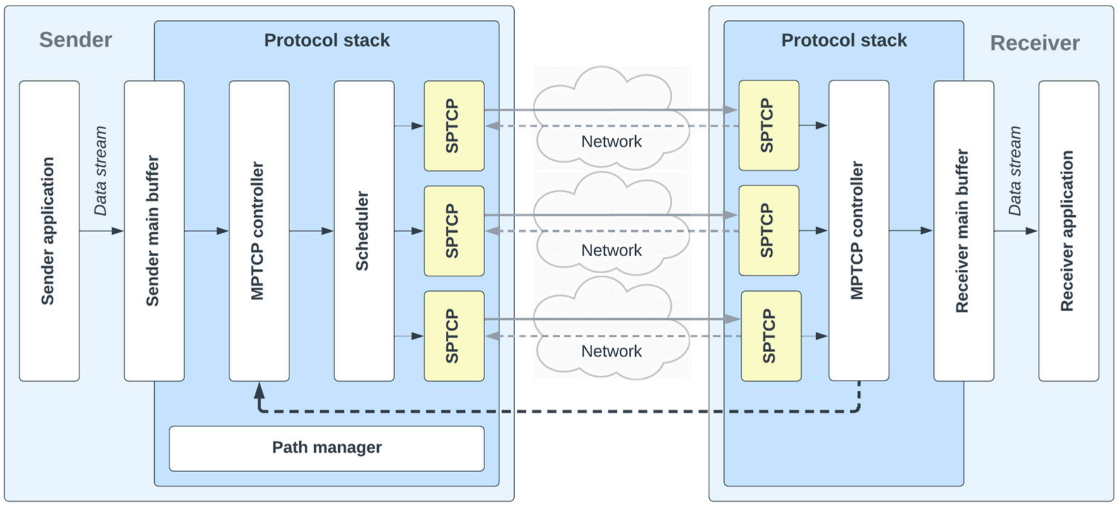


Figure 1. Data transfer in MPTCP architecture. Solid lines represent the transmission of data and the dashed ones—the acknowledgments. The MPTCP buffer is shared between the protocol stack and the user application, whereas individual buffers are allocated for each SPTCP controller.

If the multipath extension is available, the protocol attempts to open multiple transmission channels. A component known as the Path Manager then decides on the number of channels and which network paths to utilize. This multipath approach enhances resilience, as communication can continue even if some channels fail [8,10].

The data generated by the application first pass through the Master Controller, which shapes the overall data transfer characteristics [21]. From there, it moves to the Scheduler, which distributes the data among the active paths established by the Path Manager. Each path then employs a Single Path TCP (SPTCP) controller for data transfer [12,22]. The Scheduler can be adjusted to achieve specific objectives, such as minimizing power consumption or delay [23–25]. However, it does not directly influence the traffic intensity; it responds to signals from the standard SPTCP, which governs each path's data stream separately.

The Master Controller, SPTCP controllers, and Scheduler work in a union, interacting in a complex manner to meet the communication objectives. Even though TCP was initially designed to maximize throughput, the choice of SPTCP and MPTCP CC algorithms can shift this focus toward minimizing delay, thereby affecting the quality of a streaming service [6,7].

1.4. Congestion Control Algorithms in TCP Networks

CC algorithms are responsible for managing the flow of data packets in a network, ensuring that the network does not become overwhelmed with traffic, which can lead to packet loss, increased latency, and reduced throughput [26]. The choice of an adequate CC algorithm is even more important in the context of MPTCP given the complexities arising from the concurrent management of multiple network paths.

The reference design of MPTCP does not follow the single-responsibility programming pattern. The SPTCP CC algorithms used at the path level are tailored to the application needs or link specifics; thus, besides the default one (currently Cubic [27], both for Windows and Linux), there are several other variants [17,28]. The application may engage a given version on request. The idea explored in this work is allowing the applied algorithms to be responsible for one task only. In the experiments, we modified the kernel and allowed for the selection of an arbitrary CC algorithm for each link autonomously and independently of the MPTCP controller.

1.4.1. Single-Path TCP CC Algorithms

One can find comprehensive surveys of TCP CC algorithms in, e.g., [27,29]. Currently, several dozen TCP variants exist. They can be grouped as loss-sensitive, where the reduction in the transmission speed is induced by a segment drop, and time-sensitive, when the transmission slows down in response to a growing Round Trip Time (RTT). Generally, time-sensitive ones promise a smoother transmission but lose to the more aggressive loss-sensitive competitors in the case of aggravated congestion. The loss-sensitive ones show a tendency to exacerbate delay variation (jitter).

For the experimental evaluation conducted in the latter part of this work, the most representative SPTCP variants have been chosen:

- **Cubic** is the default CC algorithm in contemporary operating systems. It is designed for high-speed and long-distance networks. Cubic's window growth function is a third-degree polynomial, allowing it to perform a smoother transmission than Reno. As a result, a better utilization of available bandwidth, especially in the networks with a high bandwidth-delay product, is obtained [30,31].
- **Reno** is one of the earliest CC algorithms. While it is currently unused, it can be recognized as a basis of many other CC algorithms, notably the MPTCP ones. Reno increases the transmission speed linearly and reduces it sharply when a loss is detected. Such aggressive changes cause numerous unwanted phenomena, like considerable jitter, session synchronization, and increased vulnerability to buffer bloat [32]. Those phenomena do not harm best-effort transmission significantly, but they highly perturb real-time streaming [17].
- **BBR** (Bottleneck bandwidth and round-trip propagation time) is a recent development from Google. Unlike the traditional CC algorithms that react to the packet loss or delay as signals of network congestion, BBR measures the actual bottleneck bandwidth and round-trip propagation time. This allows BBR to achieve higher throughput and reduced latency, making it particularly effective in today's Internet, where packet loss can occur for reasons other than congestion [28].

1.4.2. Multipath TCP CC Algorithms

With the advent of MPTCP, several algorithms have been introduced to address the challenges of managing multiple network paths. Possibly, the most representative ones are:

- **LIA** (Linked Increases Algorithm)—one of the first MPTCP CC algorithms—ensures that the sum of the congestion windows of all the paths does not exceed that of a single-path one. LIA aims to be friendly to the regular TCP traffic, making sure that the MPTCP flows do not monopolize the network resources. This balances the congestion windows of different paths to provide efficient utilization of all the available paths [5,33].
- **OLIA** (Opportunistic Linked-Increases Algorithm)—an improvement of LIA—was designed to achieve better throughput and fairness across multiple paths. OLIA dynamically adjusts the congestion windows of different paths based on the current network conditions toward optimal utilization of each path. This results in better overall performance and improved user experience [33,34].

- **BALIA** (Balanced Linked Adaptation) focuses on balancing the congestion windows of different paths. By doing so, BALIA ensures that no single path is overly congested, leading to better overall network performance. BALIA is particularly effective in scenarios where the available paths have diverse and variable characteristics [35].
- **wVegas**—a delay-based MPTCP algorithm—, monitors the delay on each path and adjusts the congestion window accordingly, unlike the traditional loss-based ones. By focusing on delay as the primary metric, wVegas can achieve better performance in networks where delay is a critical factor. wVegas seems particularly well-suited for real-time applications where low latency is crucial. However, as the conducted tests reveal, it is not always the case in a multipath traffic scenario.

Each MPTCP CC algorithm reflects a unique methodology for managing data flow across multiple network paths. Their distinct design principles, combined with their strengths and potential challenges, make them apt for diverse scenarios and applications. As the landscape of digital services evolves and the demand for seamless, multipath connectivity intensifies, the selection of an appropriate MPTCP algorithm becomes a critical factor. Their strengths and weaknesses have been summarized in Table 1.

Table 1. MPTCP CC Algorithms Comparison.

Algorithm	Characteristics	Strengths	Weaknesses
LIA	Balances the congestion windows at different paths	Fairness across paths, efficient utilization	Might not be as aggressive as other algorithms
OLIA	Dynamic adaptation to network conditions	Better throughput and fairness	More complex implementation
BALIA	Focuses on balancing congestion windows	No path is overly congested, overall network performance improves	Might not be as fast as other algorithms
wVegas	Delay-based, monitors delay on each path	Low latency, suitable for real-time applications	Might be less aggressive in high-speed networks

While successfully addressing many aspects of Internet connectivity [14,22,36], the standard MPTCP algorithms discussed above were not designed with streaming traffic in mind. Their popularity, however, leads to the question we will try to answer in this work: Which one handles such delay-sensitive content most efficiently for a good user experience?

1.5. Related Works

An overview of recent research on MPTCP and its CC algorithms is given in Table 2. The table lists key findings, investigated algorithms, and applied metrics.

The current state-of-the-art in the MPTCP CC studies shows a global interest in exploring the multifaceted dimension of the problem. At the heart is the work by Jowkarishasaltaneh and But, who have meticulously examined the performance limitations of MPTCP coupled CC algorithms, especially regarding disjoint paths [37]. Their insights set the stage for Abbas's comparative analysis [38]. Drawing a contrast between MPTCP and its single-path counterpart, Abbas highlighted the superior performance of MPTCP in heterogeneous networks, with a particular recognition of the LIA and BLIA algorithms. The recognized the limitations of standard MPTCP CC algorithms motivated Ignaciuk and Morawski to search for new formalized solutions. Using the principles of discrete-time dynamic systems modeling, they proposed new flow control algorithms for MPTCP based on the concept of discrete sliding modes in [39]. In this way, they offered a fresh perspective on optimizing MPTCP's performance in networking scenarios where robustness becomes a primary issue, e.g., in industrial communication systems. As the world gravitates toward the next generation communication systems, Mahmud, Lubna, and Cho turn their attention to 5G and 4G networks in [40]. The conducted study unveils the potential of the BLEST scheduler when paired with the BALIA CC algorithm, hinting at the optimal throughput and minimized delay in the 4G/5G environments governed by MPTCP.

Table 2. Overview of recent research publications on MPTCP congestion control and related topics.

Title	Authors	Year	Key Findings	Algorithms Studied
An Analysis of MPTCP Congestion Control [37]	F. Jowkarishasaltaneh, J. But	2022	Highlighted performance limitations of MPTCP coupled congestion control algorithms in disjoint paths.	MPTCP Coupled Congestion Control
Technical Comparison between MPTCP and TCP in Heterogeneous Networks [38]	A. S. Abbas	2022	Demonstrated the superiority of MPTCP over single-path TCP in heterogeneous networks, with LIA and BLIA emphasized.	LIA, BALIA
Discrete-Time Sliding-Mode Controllers for MPTCP Networks [39]	P. Ignaciuk, M. Morawski	2021	Provides a discrete-time model of data exchange in MPTCP networks and introduces new flow control algorithms based on the concept of discrete sliding modes.	Discrete sliding-mode control
Performance Evaluation of MPTCP on Simultaneous Use of 5G and 4G Networks [40]	I. Mahmud, T. Lubna, Y.-Z. Cho	2022	Revealed that the BLEST scheduler with the BALIA CC algorithm can produce the highest throughput and lowest delay in 5G and 4G networks.	BALIA
An Intelligent TCP Congestion Control Method Based on Deep Q Network [41]	Y. Wang, L. Wang, X. Dong	2021	Introduced TCP-DQN, an intelligent TCP congestion control method based on DQN, showing a significant throughput boost.	TCP-DQN
D-OLIA: A Hybrid MPTCP Congestion Control Algorithm with Network Delay Estimation [42]	T. Lubna, I. Mahmud, G.-H. Kim, Y.-Z. Cho	2021	Proposed D-OLIA, a hybrid MPTCP-CCA that enhances OLIA's performance by integrating network delay awareness.	D-OLIA
Learning to Harness Bandwidth With Multipath Congestion Control and Scheduling [43]	S. R. Pokhrel, A. Elwalid	2021	Introduced a Deep Q-Learning-based framework for joint congestion control and packet scheduling for MPTCP.	DQL-MPTCP
Adaptive Decrease Window for BALIA (ADW-BALIA): Congestion Control Algorithm for Throughput Improvement in Nonshared Bottlenecks [44]	G.-H. Kim, Y.-J. Song, I. Mahmud, Y.-Z. Cho	2021	Proposed an adaptive decrease window for BALIA to improve throughput in scenarios with nonshared bottlenecks.	ADW-BALIA
A Price to Pay for Increased Throughput in MPTCP Transmission of Video Streams [45]	M. Morawski, P. Ignaciuk	2020	Investigated the impact of TCP CC algorithms on video stream characteristics.	Various TCP CC algorithms
Influence of Congestion Control Algorithms on Head-of-Line Blocking in MPTCP-based Communication [46]	M. Morawski, P. Ignaciuk	2019	Examined the influence of TCP CC algorithms on HoL blocking in MPTCP networks.	Various TCP CC algorithms
Synchronizing Scheduler for MPTCP Transmission of Streaming Content [47]	M. Morawski, P. Ignaciuk	2022	Proposed a new scheduler for MPTCP targeting the aspects related to protocol delay.	N/A

The landscape of MPTCP research is not restricted to traditional algorithms and network comparisons. A more avant-garde approach has been presented by Wang, Wang, and Dong in [41]. By integrating Deep Q Networks into the TCP CC, they designed TCP-DQN, an intelligent method that showcased noticeable improvements in throughput. A similar line follows the design of D-OLIA proposed by Lubna, Mahmud, Kim, and Cho in [42]. As a hybrid MPTCP CC algorithm, D-OLIA elevates the OLIA's performance by weaving in the network delay awareness, thus offering a service-tailored approach to CC design. Further steps in this direction were made by Pokhrel and Elwalid while introducing a Deep Q-Learning-based framework for joint CC and packet scheduling in MPTCP, named DQL-MPTCP [43]. Their work underscores the potential of advanced machine learning techniques in optimizing MPTCP's performance. Yet, another scenario-specific solution was explored by Kim, Song, Mahmud, and Cho in [44]. By incorporating an adaptive decrease window mechanism into MPTCP CC, termed ADW-BALIA, a throughput improvement in nonshared bottleneck networks was obtained. In turn, the work of Morawski and Ignaciuk [45], highlighted the intricacies of video streaming in MPTCP environments with no definite answer within the current multipath protocol solutions. A major obstacle seems to be the Head-of-Line (HoL) blocking phenomenon [44], examined in more depth in [46]. Also, in a recent publication, they proposed a new Scheduler for MPTCP, specifically targeting the protocol delay and stream reassembly challenges in multipath traffic scenarios [47].

Clearly, the domain of MPTCP CC studies is vast and multilayered. Researchers have examined both the performance of the already established solutions in new application areas, as well as sought new targeted ones tailored to the particular needs of a given domain. The objective of the study reported in this work is to explore the potential of key popular MPTCP CC algorithms, namely LIA, OLIA, BALIA, and wVegas, in the context of streaming applications. In order to gauge their performance in the communication scenarios relevant for end-users, the algorithms are rigorously tested in the open Internet, rather than a closed laboratory environment or simulations, similarly to many previous works. Both qualitative and quantitative analysis is performed, employing delay- and throughput-related metrics.

2. Materials and Methods

2.1. Quality Measures

In this analysis, we aim to identify the most suitable MPTCP CC algorithms for multipath streaming traffic. For quantitative comparison, the following metrics have been chosen: HoL Blocking Degree, protocol delay, Path Delay, mean drop rate, and Throughput.

2.1.1. Path Delay

Path Delay, often referred to as end-to-end delay, measures the time taken by a data packet to traverse from the source (sender) to the destination (receiver) over a path. It is a critical metric for gauging the efficiency and responsiveness of a network, especially for real-time applications where a low delay is paramount.

Several factors contribute to Path Delay, including:

- **Propagation Delay**: The time taken by a packet to cover the distance between the sender and receiver. This is primarily influenced by the physical distance and the medium, e.g., fiber, copper, or wireless, the packet traverses.
- **Transmission Delay**: The time taken to place the bits onto the link. This depends on the packet size and link bandwidth.
- **Processing Delay**: The time the routers take to process header and control information.
- **Queuing Delay**: The time the packet spends in a buffer before it is relayed on the outgoing link.

Given these components, the total delay T_i on a specific path i can be mathematically expressed by

$$T_i = \tau_i + \theta_i \tag{1}$$

- T_i—total delay on path i.

- τ_i—Smoothed Round-Trip Time (SRTT) for path i. It represents the low-pass filtered time taken by a packet traveling from the sender to the receiver and acknowledgment backward.
- θ_i—waiting time for processing the data stream on path i. This delay component is influenced by the scheduler algorithm, which determines the order in which the packets are directed to the interfaces.

2.1.2. Protocol Delay

Protocol delay, denoted by T_{over}, reflects the time a data packet waits in the buffer for stream reassembly. This delay starts from the moment the Master Controller receives the user data from the transmit buffer and ends when the acknowledgment for that data is received. Essentially, the protocol delay is equivalent to the delay on the slowest path and can be mathematically expressed as

$$T_{over}(k) = \max_i T_i(k) \tag{2}$$

where k denotes the sample index.

To provide a finer evaluation of protocol delay, we introduce additional metrics that capture its average and maximum values over multiple experimental runs. They are useful for assessing the consistency and reliability of CC algorithm operation.

The average protocol delay v^r_{av} is calculated as

$$v^r_{av} = \frac{1}{K}\sum_{k=1}^{K} T_{over}(k) \tag{3}$$

where K is the number of samples collected in a single experiment run r.

The maximum protocol delay $v^r_{\max}$ for the same run is given by

$$v^r_{\max} = \max_{k\in[1,K]} T_{over}(k) \tag{4}$$

To obtain metrics for multiple runs, we introduce the average protocol delay v_{av} and the average maximum protocol delay $v_{\max}$:

$$v_{av} = \frac{1}{R}\sum_{r=1}^{R} v^r_{av} \tag{5}$$

$$v_{\max} = \frac{1}{R}\sum_{r=1}^{R} v^r_{\max} \tag{6}$$

where R is the total number of experiment runs.

2.1.3. HoL Blocking Degree

HoL blocking manifests itself when the first packet in a queue obstructs the subsequent packets from being conveyed to the application, leading to increased latency and degraded QoS [4]. In MPTCP, HoL blocking can be particularly detrimental for real-time applications like video streaming and online gaming. The waiting time is defined as

$$T_{over}(k) - \max_{i\in[1,m]} \tau_i(k) \tag{7}$$

The average waiting time across all the paths is then calculated as

$$\zeta^r_{av} = \frac{1}{K}\sum_{k=1}^{K}\left(T_{over}(k) - \max_{i\in[1,m]} \tau_i(k)\right) \tag{8}$$

and the maximum waiting time for each experiment as:

$$\zeta^r_{\max} = \max_{k\in[1,K]}\left(T_{over}(k) - \max_{i\in[1,m]}\tau_i(k)\right) \tag{9}$$

The average waiting time across all the experiments is

$$\zeta_{av} = \frac{1}{R}\sum_{r=1}^{R}\zeta^r_{av} \tag{10}$$

and the average maximum waiting time across all the experiments

$$\zeta_{\max} = \frac{1}{R}\sum_{r=1}^{R}\max(\zeta^r_{\max}, 0) \tag{11}$$

2.1.4. Mean Drop Rate

The mean drop rate, denoted as δ, allows for the gauging of the reliability and efficiency of a network. For applications like file transfers, occasional packet drops do not impact the user experience much since there is time enough time for retransmissions. However, for real-time applications, like video streaming or VoIP, the packet loss may lead to significant quality degradation. The lost video frames or audio snippets can lead to jittery playback or call drops.

The mean drop rate is calculated as

$$\delta = \frac{D}{S} \times 100\% \tag{12}$$

where:

- D—total number of dropped packets;
- S—total number of sent packets.

2.2. Experimental Setup

The applied experimental setup depicted in Figure 2 reflects a typical client–server data transmission scenario, in which the client device fetches content from a server situated in a remote data center.

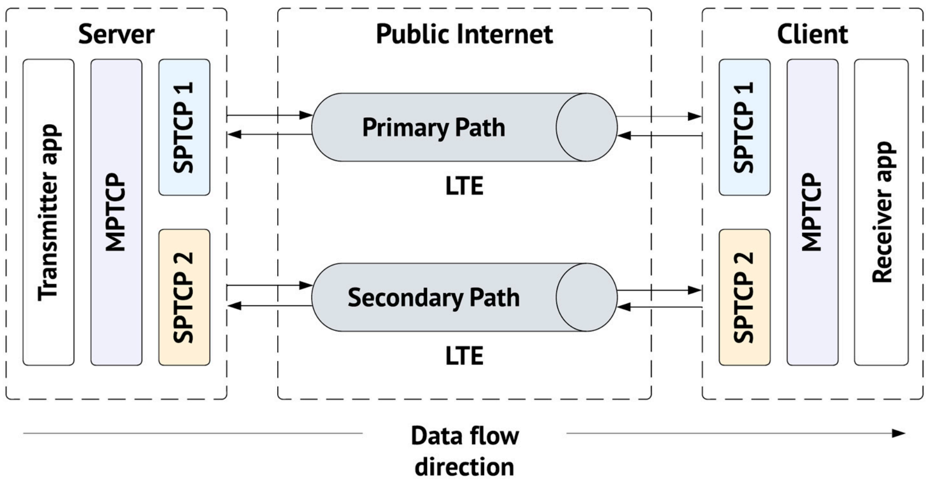


Figure 2. Experimental setup.

The server is accessible via a public IP address and the streaming content is transferred via public networks. The server device is equipped with high processing capabilities and sufficient storage to handle the streaming without extra delay.

The client device features dual communication interfaces:

- Ethernet: Connected to the first LTE router via an Ethernet cable;
- Wi-Fi: Linked to the second LTE router through Wi-Fi 802.11 bgn.

Two distinct LTE networks from different operators were used for the network path diversity. The signal quality was continuously monitored to maintain appropriate test conditions. The packets sent through the Ethernet interface traverse 10 network hops, while those sent via the Wi-Fi interface go through 12 hops.

Both the client and server devices have Linux operating system, version 4.19, installed. It was patched to support MPTCP version 0.95. A specialized software program was used to generate the streaming data.

The test scenario lasted 10 s and was repeated 30 times to collect a representative data sample. Statistical methods were used to calculate the extreme and average values for each metric. The following metrics were applied to gauge the algorithm performance:

- HoL Blocking Degree;
- protocol delay;
- Path Delay;
- mean drop rate;
- throughput.

as explained in Section 2.1.

3. Results and Discussion

At the MPTCP level, four CC algorithms, LIA, OLIA, BALIA, and wVegas, were evaluated, whereas at the path level, Reno, Cubic, and BBR were considered, which gives 12 protocol pairs. The performance measures, calculated from the obtained data, are gathered in Table 3. In addition, for visual representation complementing the numerical data, the results of one representative run are depicted in Figures 3 and 4.

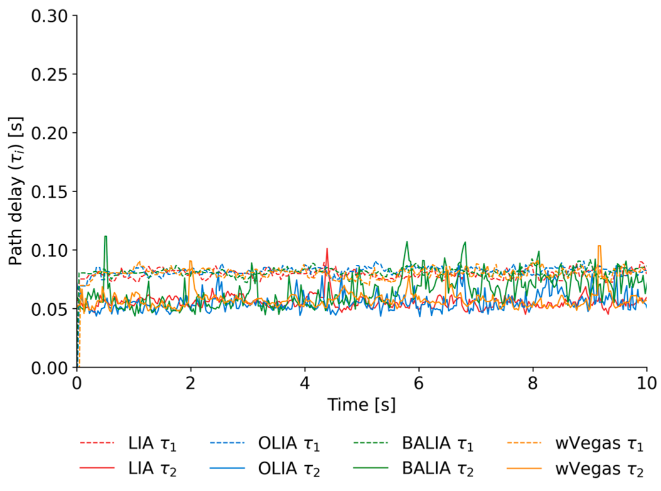
Table 3. Performance metrics.

		LIA			**OLIA**			**BALIA**			**wVegas**		
		Reno	**Cubic**	**BBR**	**Reno**	**Cubic**	**BBR**	**Reno**	**Cubic**	**BBR**	**Reno**	**Cubic**	**BBR**
Protocol delay (ms)	v_{av}	96	97	231	98	96	95	95	99	95	271	99	99
	v_{max}	225	224	569	256	235	238	231	295	243	816	259	254
HoL Degree (ms)	ζ_{av}	17	17	151	19	16	16	15	18	15	191	19	19
	ζ_{max}	143	143	488	175	150	155	148	211	157	733	178	174
SRTT (ms) path 1	$\tau_{1,av}$	79	79	79	78	78	77	79	80	79	77	79	78
	$\tau_{1,max}$	94	97	95	98	98	96	98	102	97	96	95	94
SRTT (ms) path 2	$\tau_{2,av}$	57	56	56	56	58	57	59	59	58	56	57	57
	$\tau_{2,max}$	88	86	89	92	97	103	104	102	105	79	83	75
Mean drop rate (seg/s)	d_1	0.5	1.4	0.1	0.2	1.5	2.7	1.3	3.0	0.5	3.4	0.5	2.1
	d_2	1.0	0.8	1.2	1.5	1.0	0.8	0.7	1.3	1.5	1.6	1.6	1.4
Throughput [Mbps]	*av*	7.51	7.84	7.67	9.65	10.58	11.08	11.40	11.02	11.23	5.60	5.86	5.77
	max	16.24	17.01	16.98	19.53	21.28	22.71	23.62	23.09	23.54	13.53	12.97	12.91

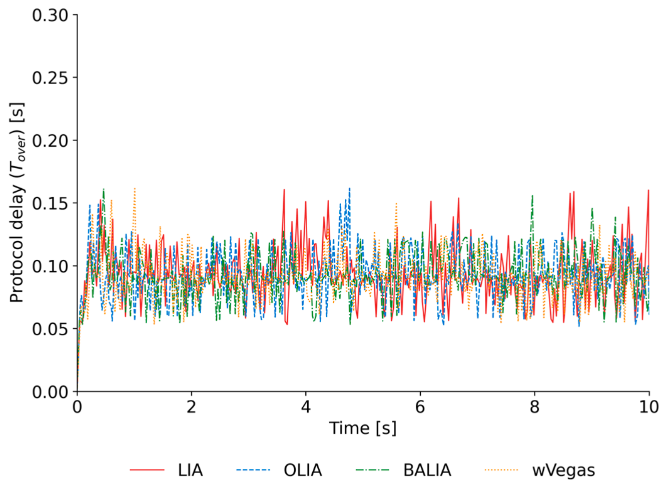
3.1. Result Analysis

3.1.1. Mean Drop Rate

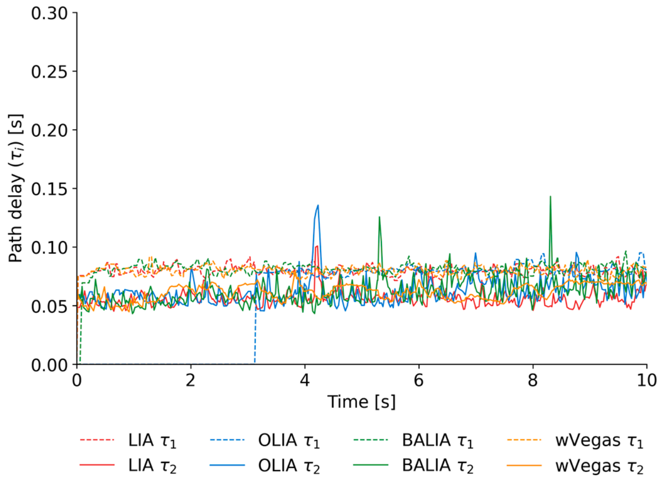
The lowest mean drop rate was observed for LIA as the MPTCP CC algorithm combined with BBR at the path level for both d_1 and d_2. Thus, this combination is highly efficient in terms of losses and uninterrupted transmission. In turn, wVegas with Reno and BALIA with Cubic at the path level resulted in the highest drop rate, indicating a loss of reliability and proclivity to cause stream pauses or even breaks, e.g., when keyframes are missing.



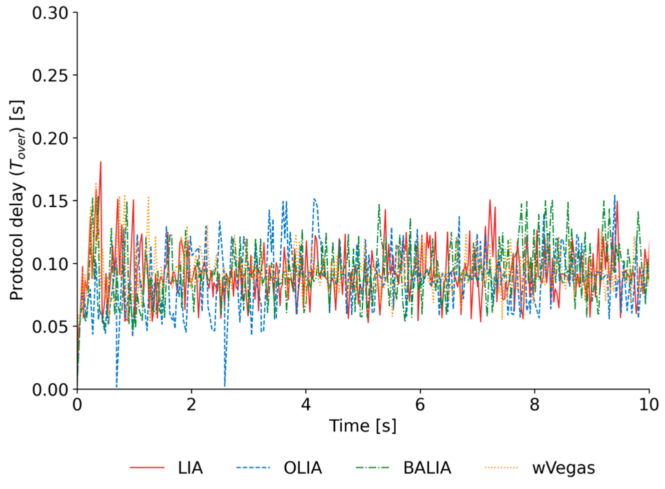
(**a**) Path Delay with Reno as SPTCP



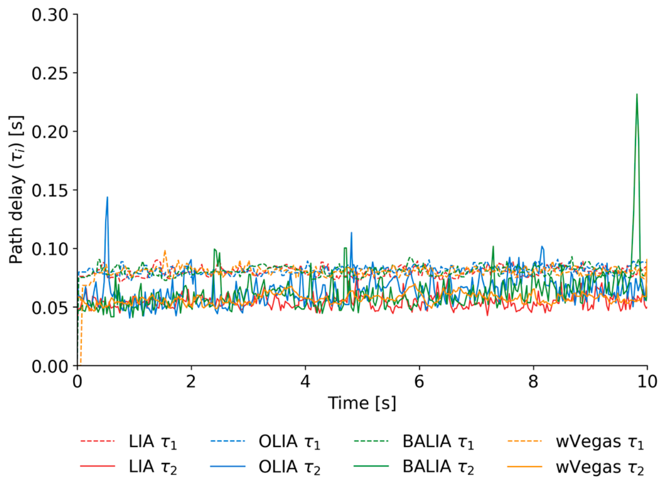
(**b**) Protocol delay with Reno as SPTCP



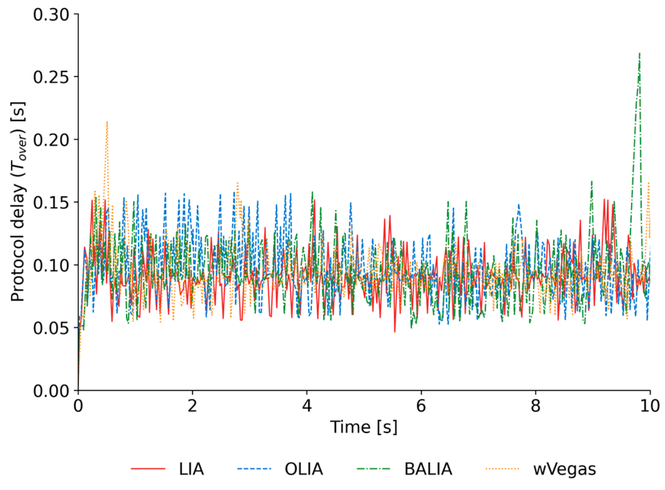
(**c**) Path Delay with Cubic as SPTCP



(**d**) Protocol delay with Cubic as SPTCP

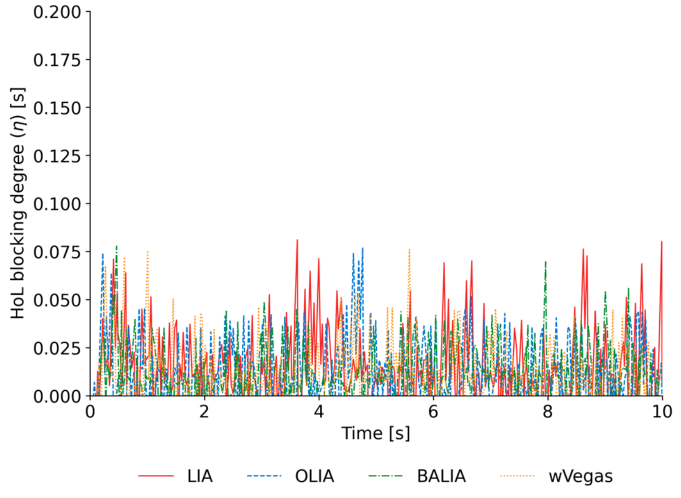


(**e**) Path Delay with BBR as SPTCP

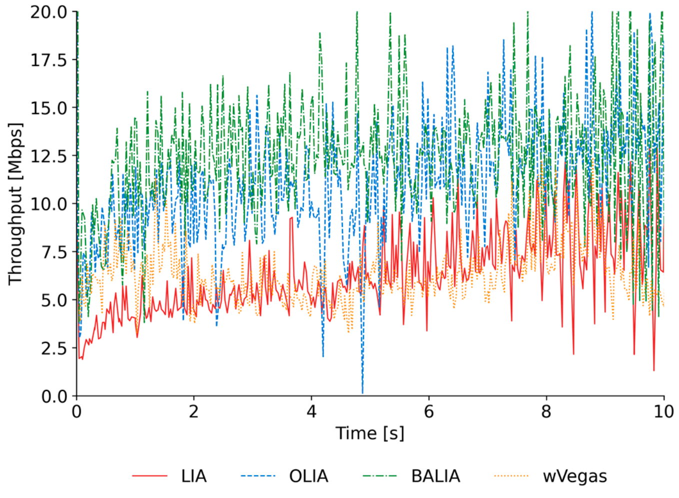


(**f**) Protocol delay with BBR as SPTCP

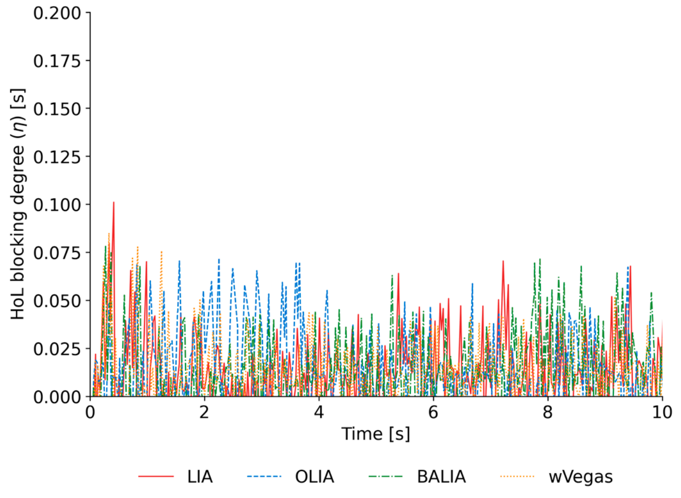
Figure 3. Transmission characteristics of LIA, OLIA, BALIA, and mVegas at the MPTCP level, juxtaposed with Reno (top row: (**a**,**b**)), Cubic (middle row: (**c**,**d**)), and BBR (bottom row: (**e**,**f**)) at the SPTCP level. The left column (**a**,**c**,**e**) showcases Path Delay, and the right column (**b**,**d**,**f**) shows protocol delay.



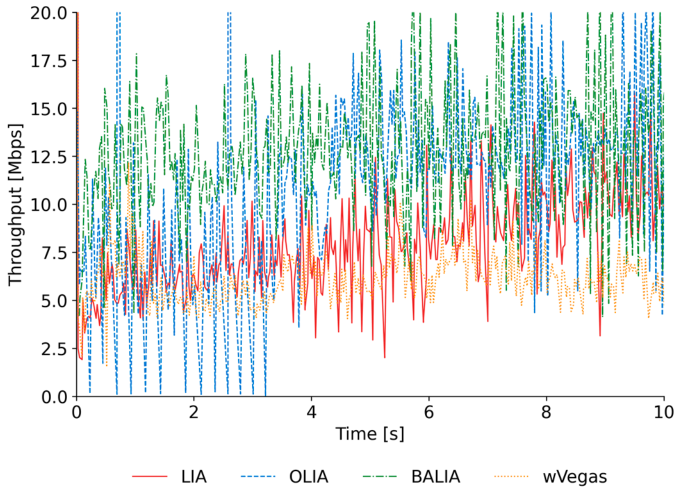
(**a**) HoL Blocking Degree with Reno as SPTCP



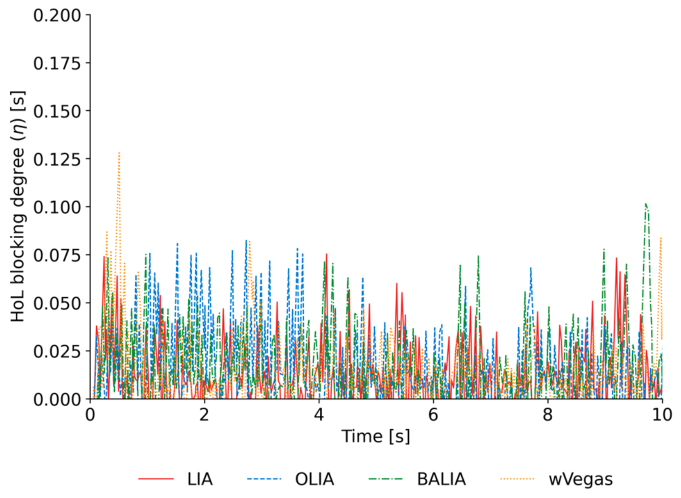
(**b**) Throughput with Reno as SPTCP



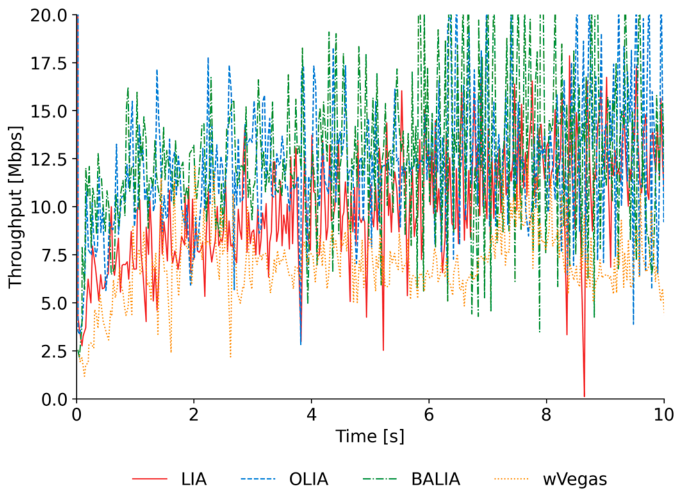
(**c**) HoL Blocking Degree with Cubic as SPTCP



(**d**) Throughput with Cubic as SPTCP



(**e**) HoL Blocking Degree with BBR as SPTCP



(**f**) Throughput with BBR as SPTCP

Figure 4. Transmission characteristics of LIA, OLIA, BALIA, and mVegas at the MPTCP level, juxtaposed with Reno (top row: (**a**,**b**)), Cubic (middle row: (**c**,**d**)), and BBR (bottom row: (**e**,**f**)) at the SPTCP level. The left column (**a**,**c**,**e**) showcases HoL Blocking Degree, and the right column (**b**,**d**,**f**) shows throughput.

3.1.2. Path Delay (SRRT)

With respect to SRRT (on both paths), wVegas paired with Reno at the path level and OLIA with BBR performed the best, respectively. These combinations seem to offer the lowest latency and jitter, which are crucial for real-time applications. Conversely, the maximum SRRT was observed for BALIA paired with Cubic and BALIA paired with BBR. In addition, the associated occasional latency spikes can negatively impact the stream consistency, leading to a diminished user experience.

Detailed insights into the SRRT performance can be obtained from Figure 3, where plots (a), (c), and (e) present the charts from one experiment run. Upon a closer examination, small fluctuations in latency are evident. For the first LTE connection, the transfer appears more stable, with insignificant jitter (about $\pm$10 ms). In contrast, for the second connection, the latency fluctuations are more pronounced, especially for BALIA and OLIA, with variations reaching up to $\pm$30 ms. However, these variations oscillate between two bounds, suggesting that they can be compensated for with appropriate buffer usage. It is also noteworthy that, barring certain network phenomena resulting in peak-like artifacts, all three SPTCP CC algorithms—Reno, Cubic, and BBR—perform similarly well concerning Path Delay.

3.1.3. Protocol Delay

LIA with BBR exhibited the highest average protocol delay, a result that was unexpected given its low mean drop rate. This anomaly might be attributed to BBR's embedded reluctance to prompt throughput change, which can lead to temporarily longer queuing delays. In turn, wVegas paired with Reno at the path level registered the highest maximum protocol delay, rendering it less suitable for the applications that prioritize low latency, such as those in the tactile Internet.

Figure 3 gives a more granular view of the protocol delay, specifically plots (b), (d), and (f). Across all three SPTCP CC algorithms—Reno, Cubic, and BBR—fluctuations in the protocol delay are evident. The maximum values hover around 135 ms, with occasional peaks reaching 150 ms, whereas the minimum values stabilize near 60 ms. The mean is approximately 95 ms, aligning closely with the average derived from all the experiments, as detailed in Table 3. In the BBR plot, a lone peak of around 275 ms points out abnormal temporary traffic conditions.

3.1.4. HoL Blocking Degree

On the one hand, wVegas paired with Reno at the path level exhibited the highest average and maximum HoL Blocking Degree, making it suboptimal for streaming applications. Conversely, BALIA with Reno and BALIA with BBR demonstrated the lowest average and maximum HoL, meaning superior performance in terms of packet reordering. Low HoL values are especially advantageous for streaming applications. With reduced jitter and diminished buffer requirements, they not only enhance the overall user experience, but also relieve the resource pressure for low-end and mobile terminals.

A detailed examination of the HoL Blocking Degree can be found in Figure 4, specifically plots (a), (c), and (e). All three SPTCP CC algorithms displayed pronounced fluctuations, with peaks reaching up to 125 ms for the combination of BBR and wVegas, approximately 100 ms for Cubic and LIA, and 80 ms for Reno and LIA. When juxtaposed with the data from Table 3, these values are consistent, but for combinations like BBR with LIA and Reno with wVegas, values from other runs were observed to be four to six times higher than those depicted in the presented graphs. The magnitude of these fluctuations becomes evident when considering the averages computed from all the experiments. Except for BBR paired with LIA and Reno with wVegas, these averages oscillate around 20 ms. However, the graphs indicate that the fluctuation band spans from 0 to 40 ms and occasionally exceeds this range.

3.1.5. Throughput

BALIA paired with Reno at the path level exhibited the highest average throughput, followed closely by OLIA with BBR. This observation suggests that such algorithm combinations are more efficient in utilizing the available bandwidth than other examined configurations. As a result, they may offer a superior quality streaming service, such as delivering a higher-resolution video. Conversely, wVegas with Reno registered the lowest average throughput, indicating less efficient bandwidth usage, which could result in a lower-resolution stream.

A detailed view of the throughput performance is provided in Figure 4, specifically plots (b), (d), and (f). All three SPTCP CC algorithms showed aggravated fluctuations, with a tendency to improve over time. BALIA outperforms the other algorithms in all three combinations, providing the highest throughput. LIA exhibits a stable rise in throughput over time, wVegas oscillates around its mean value, resulting in an almost flat average throughput, while OLIA emerges as the most unstable algorithm. OLIA's throughput shows the most significant variations, and in combination with Cubic at the path level, it often approaches zero.

These results emphasize the importance of effective buffer management. While the observed throughput variations are evident, they do not pose an inherent threat to stream consistency as long as there are no significant drops in the transmission speed. With proper buffer management, even with these fluctuations taking place, the amount of data transferred will be sufficient to achieve a decent streaming quality.

3.2. Implications and Future Research

The presented study evaluated the common MPTCP and SPTCP algorithms from the perspective of real-time and streaming services.

LIA with BBR acting at the path level seems best suited for applications where packet loss is a critical concern, while OLIA with BBR and BALIA with Reno are intended for bandwidth-intensive ones. The lower HoL Blocking Degree attained by BALIA paired either with Reno or BBR is particularly attractive for conveying multimedia content. They allow for smaller buffers (important for resource-constrained mobile appliances) and reduced jitter.

Nevertheless, these standard CC algorithms face fundamental structural limitations in the considered application area originating from different design premises—boosting throughput while keeping fairness—which are not critical for real-time systems. A valuable direction for future research is to develop new MPTCP CC algorithms that would target the major challenges of streaming in the multipath paradigm, i.e., to maintain coherency and high quality despite the inevitable jitter and HoL blocking in the open Internet.

4. Conclusions

The paper's objective was to assess the effectiveness of popular MPTCP CC algorithms, specifically LIA, OLIA, BALIA, and wVegas, in conveying delay-sensitive content for streaming-oriented applications. The algorithms were tested in the open Internet connectivity with performance quantified through objective measures of protocol delay, Path Delay, mean drop rate, throughput, and HoL blocking. LIA paired with BBR at the path level and BALIA with Reno ensured a low mean drop rate and high throughput. BALIA when paired with either BBR or Reno, also achieved the lowest HoL Blocking Degree, which translates to a reduced waiting time for the stream reassembly at the receiver, hence, an enhanced user experience regarding multimedia traffic.

From the perspective of Path Delay, the combination of wVegas and Reno, as well as OLIA and BBR, emerged as the most efficient ones, offering the lowest latency and jitter, which is paramount for real-time applications. In terms of the protocol delay, LIA with BBR displayed the highest average, while the combination of wVegas and Reno was much less efficient. BALIA paired with Reno proved to be the most attractive with respect to bandwidth utilization, making it a primary choice for high-resolution streaming.

Overall, LIA with BBR appears to be the best choice for the applications where packet loss is most detrimental. In contrast, OLIA paired with BBR and BALIA combined with Reno are adequate for bandwidth-intensive ones. The lower HoL degree achieved by BALIA, when paired with either Reno or BBR, is especially beneficial for streaming involving resource-constrained devices, like mobile appliances.

However, one should recognize that the common MPTCP and SPTCP algorithms evaluated in this study have inherent limitations when considered in the context of real-time and streaming services. Their primary design objectives, i.e., to maximize throughput while maintaining fairness, may not always align with the requirements of delay-sensitive applications. This realization underscores the need to develop new MPTCP CC algorithms. These new algorithms should address the unique challenges of streaming systems, particularly HoL blocking and delay variability.

Author Contributions: Conceptualization, P.I. and M.M.; methodology, P.I., M.M. and ŁP.Ł.; software, M.M. and ŁP.Ł.; validation, ŁP.Ł. and P.I.; formal analysis, ŁP.Ł. and P.I.; investigation, ŁP.Ł. and P.I.; resources, ŁP.Ł.; data curation, ŁP.Ł.; writing—original draft preparation, ŁP.Ł; writing—review and editing, P.I. and M.M.; visualization, Ł.P.Ł.; supervision, P.I.; project administration, P.I.; funding acquisition, P.I. All authors have read and agreed to the published version of the manuscript.

Funding: This work has been performed in the framework of the project "Robust control solutions for multi-channel networked flows" no. 2021/41/B/ST7/00108 financed by the National Science Centre, Poland.

Data Availability Statement: No application.

Acknowledgments: This work has been completed while the first author was the Doctoral Candidate in the Interdisciplinary Doctoral School at the Lodz University of Technology, Poland.

Conflicts of Interest: The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript; or in the decision to publish the results.

References

1. Cisco Annual Internet Report (2018–2023) White Paper. Available online: https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html (accessed on 1 September 2023).
2. Barreiros, M.; Lundqvist, P. *QOS-Enabled Networks: Tools and Foundations*, 1st ed.; Wiley: Hoboken, NJ, USA, 2016; ISBN 978-1-119-10910-5.
3. Qadir, J.; Ali, A.; Yau, K.-L.A.; Sathiaseelan, A.; Crowcroft, J. Exploiting the Power of Multiplicity: A Holistic Survey of Network-Layer Multipath. *IEEE Commun. Surv. Tutor.* **2015**, *17*, 2176–2213. [CrossRef]
4. Morawski, M.; Ignaciuk, P. Choosing a Proper Control Strategy for Multipath Transmission in Industry 4.0 Applications. *IEEE Trans. Ind. Inform.* **2022**, *18*, 3609–3619. [CrossRef]
5. Raiciu, C.; Handley, M.; Wischik, D. *Coupled Congestion Control for Multipath Transport Protocols*; Internet Engineering Task Force (IETF): Fremont, CA, USA, 2011; ISSN 2070-1721.
6. Xu, C.; Zhao, J.; Muntean, G.-M. Congestion Control Design for Multipath Transport Protocols: A Survey. *IEEE Commun. Surv. Tutor.* **2016**, *18*, 2948–2969. [CrossRef]
7. Yedugundla, K.; Ferlin, S.; Dreibholz, T.; Alay, Ö.; Kuhn, N.; Hurtig, P.; Brunstrom, A. Is Multi-Path Transport Suitable for Latency Sensitive Traffic? *Comput. Netw.* **2016**, *105*, 1–21. [CrossRef]
8. Morawski, M.; Ignaciuk, P. A Green Multipath TCP Framework for Industrial Internet of Things Applications. *Comput. Netw.* **2021**, *187*, 107831. [CrossRef]
9. Peng, Q.; Walid, A.; Hwang, J.; Low, S.H. Multipath TCP: Analysis, Design, and Implementation. *IEEE/ACM Trans. Netw.* **2016**, *24*, 596–609. [CrossRef]
10. Li, M.; Lukyanenko, A.; Ou, Z.; Yla-Jaaski, A.; Tarkoma, S.; Coudron, M.; Secci, S. Multipath Transmission for the Internet: A Survey. *IEEE Commun. Surv. Tutor.* **2016**, *18*, 2887–2925. [CrossRef]
11. Ford, A.; Raiciu, C.; Handley, M.; Bonaventure, O. *TCP Extensions for Multipath Operation with Multiple Addresses*; Internet Engineering Task Force (IETF): Fremont, CA, USA, 2013.
12. Paasch, C.; Ferlin, S.; Alay, O.; Bonaventure, O. Experimental Evaluation of Multipath TCP Schedulers. In Proceedings of the 2014 ACM SIGCOMM Workshop on Capacity Sharing Workshop, Chicago, IL, USA, 18 August 2014; ACM: Chicago, IL, USA, 2014; pp. 27–32.
13. MultiPath TCP—Linux Kernel Implementation. Available online: https://www.multipath-tcp.org/ (accessed on 1 September 2023).

14. Cao, Y.; Xu, M.; Fu, X. Delay-Based Congestion Control for Multipath TCP. In Proceedings of the 2012 20th IEEE International Conference on Network Protocols (ICNP), Austin, TX, USA, 30 October–2 November 2012; IEEE: Austin, TX, USA, 2012; pp. 1–10.
15. Yu, C.; Quan, W.; Cheng, N.; Chen, S.; Zhang, H. Coupled or Uncoupled? Multi-Path TCP Congestion Control for High-Speed Railway Networks. In Proceedings of the 2019 IEEE/CIC International Conference on Communications in China (ICCC), Changchun, China, 11–13 August 2019; pp. 612–617.
16. Wei, W.; Xue, K.; Han, J.; Wei, D.S.L.; Hong, P. Shared Bottleneck-Based Congestion Control and Packet Scheduling for Multipath TCP. *IEEE/ACM Trans. Netw.* **2020**, *28*, 653–666. [CrossRef]
17. Floyd, S.; Henderson, T.; Gurtov, A. *The NewReno Modification to TCP's Fast Recovery Algorithm*; Internet Engineering Task Force (IETF): Fremont, CA, USA, 2004.
18. Bonaventure, O.; Piraux, M.; Coninck, Q.; Baerts, M.; Paasch, C.; Amend, M. Multipath Schedulers. Internet Engineering Task Force, Internet-Draft Draft-Bonaventure-Iccrg-Schedulers-00. 2020. Available online: https://datatracker.ietf.org/doc/html/draft-bonaventure-iccrg-schedulers-02 (accessed on 1 September 2023).
19. Kimura, B.Y.L.; Lima, D.C.S.F.; Loureiro, A.A.F. Alternative Scheduling Decisions for Multipath TCP. *IEEE Commun. Lett.* **2017**, *21*, 2412–2415. [CrossRef]
20. Hussein, A.; Elhajj, I.H.; Chehab, A.; Kayssi, A. SDN for MPTCP: An Enhanced Architecture for Large Data Transfers in Datacenters. In Proceedings of the 2017 IEEE International Conference on Communications (ICC), Paris, France, 21–25 May 2017; IEEE: Paris, France, 2017; pp. 1–7.
21. Williams, N.; Armitage, G.; But, J. Implementing a Multipath Transmission Control Protocol (MPTCP) Stack for FreeBSD with Pluggable Congestion and Scheduling Control. Ph.D. Thesis, Swinburne University of Technology, Melbourne, Australia, 2016.
22. Kimura, B.Y.L.; Lima, D.C.S.F.; Loureiro, A.A.F. Packet Scheduling in Multipath TCP: Fundamentals, Lessons, and Opportunities. *IEEE Syst. J.* **2021**, *15*, 1445–1457. [CrossRef]
23. Morawski, M.; Ignaciuk, P. Energy-Efficient Scheduler for MPTCP Data Transfer with Independent and Coupled Channels. *Comput. Commun.* **2018**, *132*, 56–64. [CrossRef]
24. Paasch, C. Improving Multipath TCP. Ph.D. Thesis, Université Catholique de Louvain (UCL), London, UK, 2014.
25. Hurtig, P.; Grinnemo, K.-J.; Brunstrom, A.; Ferlin, S.; Alay, O.; Kuhn, N. Low-Latency Scheduling in MPTCP. *IEEE/ACM Trans. Netw.* **2019**, *27*, 302–315. [CrossRef]
26. Al-Saadi, R.; Armitage, G.; But, J.; Branch, P. A Survey of Delay-Based and Hybrid TCP Congestion Control Algorithms. *IEEE Commun. Surv. Tutor.* **2019**, *21*, 3609–3638. [CrossRef]
27. Afanasyev, A.; Tilley, N.; Reiher, P.; Kleinrock, L. Host-to-Host Congestion Control for TCP. *IEEE Commun. Surv. Tutor.* **2010**, *12*, 304–342. [CrossRef]
28. Cardwell, N.; Cheng, Y.; Gunn, C.S.; Yeganeh, S.H.; Jacobson, V. BBR: Congestion-Based Congestion Control: Measuring Bottleneck Bandwidth and Round-Trip Propagation Time. *Queue* **2016**, *14*, 20–53. [CrossRef]
29. Polese, M.; Chiariotti, F.; Bonetto, E.; Rigotto, F.; Zanella, A.; Zorzi, M. A Survey on Recent Advances in Transport Layer Protocols. *IEEE Commun. Surv. Tutor.* **2019**, *21*, 3584–3608. [CrossRef]
30. Yong Lee, J.; Chul Kim, B.; Kwon, Y.; Han, K. Coupled CUBIC Congestion Control for MPTCP in Broadband Networks. *Comput. Syst. Sci. Eng.* **2023**, *45*, 99–115. [CrossRef]
31. Wang, J.; Wen, J.; Han, Y.; Zhang, J.; Li, C.; Xiong, Z. CUBIC-FIT: A High Performance and TCP CUBIC Friendly Congestion Control Algorithm. *IEEE Commun. Lett.* **2013**, *17*, 1664–1667. [CrossRef]
32. Gettys, J. Bufferbloat: Dark Buffers in the Internet. *IEEE Internet Comput.* **2011**, *15*, 96. [CrossRef]
33. Kato, T.; Diwakar, A.; Yamamoto, R.; Ohzahata, S.; Suzuki, N. Experimental Analysis of MPTCP Congestion Control Algorithms; Lia, Olia and Balia. In Proceedings of the 8th International Conference on Theory and Practice in Modern Computing 2019, Porto, Portugal, 16–18 July 2019; IADIS Press: Lisbon, Portugal, 2019; pp. 135–142.
34. Gast, N.; Khalili, R.; Boudec, J.-Y.L.; Popovic, M. Opportunistic Linked-Increases Congestion Control Algorithm for MPTCP. 2014. Available online: https://www.semanticscholar.org/paper/Opportunistic-Linked-Increases-Congestion-Control-Gast-Khalili/0d138cfb2412b931de2a461f21c7a6e19dbf99ef (accessed on 1 September 2023).
35. Walid, A.; Peng, Q.; Hwang, J.; Low, S.H. *Balanced Linked Adaptation Congestion Control Algorithm for MPTCP*; Internet Engineering Task Force: Fremont, CA, USA, 2016.
36. Khalili, R.; Gast, N.; Popovic, M.; Le Boudec, J.-Y. MPTCP Is Not Pareto-Optimal: Performance Issues and a Possible Solution. *IEEE/ACM Trans. Netw.* **2013**, *21*, 1651–1665. [CrossRef]
37. Jowkarishasaltaneh, F.; But, J. An Analysis of MPTCP Congestion Control. *Telecom* **2022**, *3*, 581–609. [CrossRef]
38. Abbas, A.S. Technical Comparison between MPTCP and TCP in Heterogeneous Networks. *Int. J. Interact. Mob. Technol.* **2022**, *16*, 163–175. [CrossRef]
39. Ignaciuk, P.; Morawski, M. Discrete-Time Sliding-Mode Controllers for MPTCP Networks. *IEEE Trans. Syst. Man Cybern. Syst.* **2021**, *51*, 6029–6039. [CrossRef]
40. Mahmud, I.; Lubna, T.; Cho, Y.-Z. Performance Evaluation of MPTCP on Simultaneous Use of 5G and 4G Networks. *Sensors* **2022**, *22*, 7509. [CrossRef] [PubMed]
41. Wang, Y.; Wang, L.; Dong, X. An Intelligent TCP Congestion Control Method Based on Deep Q Network. *Future Internet* **2021**, *13*, 261. [CrossRef]

42. Lubna, T.; Mahmud, I.; Kim, G.-H.; Cho, Y.-Z. D-OLIA: A Hybrid MPTCP Congestion Control Algorithm with Network Delay Estimation. *Sensors* **2021**, *21*, 5764. [CrossRef]
43. Pokhrel, S.R.; Walid, A. Learning to Harness Bandwidth with Multipath Congestion Control and Scheduling. *IEEE Trans. Mob. Comput.* **2023**, *22*, 996–1009. [CrossRef]
44. Kim, G.-H.; Song, Y.-J.; Mahmud, I.; Cho, Y.-Z. Adaptive Decrease Window for BALIA (ADW-BALIA): Congestion Control Algorithm for Throughput Improvement in Nonshared Bottlenecks. *Electronics* **2021**, *10*, 294. [CrossRef]
45. Morawski, M.; Ignaciuk, P. A Price to Pay for Increased Throughput in MPTCP Transmission of Video Streams. In Proceedings of the 2020 24th International Conference on System Theory, Control and Computing (ICSTCC), Sinaia, Romania, 8–10 October 2020; pp. 673–678.
46. Morawski, M.; Ignaciuk, P. Influence of Congestion Control Algorithms on Head-of-Line Blocking in MPTCP-Based Communication. In Proceedings of the 2019 27th Telecommunications Forum (TELFOR), Belgrade, Serbia, 26–27 November 2019; pp. 1–4.
47. Morawski, M.; Ignaciuk, P. Synchronizing Scheduler for MPTCP Transmission of Streaming Content. In Proceedings of the 2022 IEEE International Conference on Systems, Man, and Cybernetics (SMC), Prague, Czech Republic, 9–12 October 2022; pp. 909–914.