DOI: 10.15439/2023F9991
ISSN 2300-5963 ACSIS, Vol. 37

Experimental Assessment of MPTCP Congestion Control Algorithms for Streaming Services in Open Internet

Łukasz Piotr Łuczak
0000−0003−0892−7276
Institute of Information Technology
Lodz University of Technology
Łódź, Poland
lukasz.luczak@dokt.p.lodz.pl

Przemysław Ignaciuk, *IEEE Senior Member*
0000−0003−4420−9941
Institute of Information Technology
Lodz University of Technology
Łódź, Poland
przemyslaw.ignaciuk@p.lodz.pl

Michał Morawski
0000−0002−8902−1259
Institute of Information Technology
Lodz University of Technology
Łódź, Poland
michal.morawski@p.lodz.pl

***Abstract*—Efficient data transfer is required in various fields such as entertainment, business communications, image processing systems, and industrial applications. A fast speed, low latency, and stable transmission parameters are required to ensure high-quality streaming, which is difficult to achieve with a single data channel. Using multiple communication paths is a promising solution for elevating performance. The Multipath TCP (MPTCP) protocol allows for splitting the application stream among a few connections. A key element determining the overall transmission quality is the MPTCP congestion control algorithm. In this paper, the most common MPTCP congestion control algorithms are evaluated in the open Internet in the context of streaming applications. The results obtained indicate that for a streaming service that utilizes multiple paths the most effective pair of CC algorithms are BALIA at the MPTCP level and BBR at the path level. These algorithms provide the smallest path delay and Head-of-Line blocking degree under consistent throughput. Delay-based wVegas shows the weakest performance in terms of multipath streaming.**

***Index Terms*—MPTCP, congestion control, streaming applications, tactile Internet**

I. Introduction

IP-based systems are gradually replacing other network solutions in traditional telecommunications, medicine, and industrial automation, as well as in new areas like entertainment, Internet of Things (IoT), and tactile Internet [13]. It occurs despite the fact that other solutions can provide better Quality of Service (QoS) measures, e.g., guaranteed minimum bandwidth, fault tolerance, or maximum latency. An unquestioned advantage of IP networks is their universality and ease of expansion, which results in economic benefits. Unlike other network solutions, IP networks require a generic connection to the network, only, utilizing its dynamic routing capabilities as a transport basis. Despite continuous efforts to improve QoS [17], a disadvantage of IP networks is the lack of control over the transmission quality. For time-sensitive transmissions, UDP/RTP protocols may be used. However, due to security requirements or application restrictions, the preferred form of data transmission is the TCP protocol.

The widespread use of mobile appliances has created new possibilities and challenges. The link parameters vary with time, rapidly. In addition, the movement of devices makes it necessary to smoothly switch to another network. Although the logical IP address of the device may remain unchanged, the link parameters may be radically different. In the considered class, the terminals are often equipped with more than one network interface, e.g., a cellular (LTE, 5G) and a Wi-Fi one. Already, these two interfaces have completely different characteristics. In addition, changing the location entails a change of the access point. Various phenomena, outside the control of communicating agent, such as interference from other users and appliances, aggravate the system uncertainty and limit the available range of services. In order to address these problems, it has been proposed to simultaneously engage multiple transmission channels using different physical interfaces [17, 18, 19], thus mitigating the impact of uncertainty. However, early attempts to materialize this idea failed [20], until a new version of the TCP protocol tailored for multi-interface traffic – Multipath TCP (MPTCP) was proposed [21, 22]. Conveniently, the reference implementation of MPTCP [23] addresses the general aspects of the protocol's behavior, only, which allows for potential innovations in its implementation [24], in particular the choice of congestion control (CC) algorithm.

Streaming applications, such as on-demand entertainment or video systems, often utilize adaptive data compression methods and do not require high bandwidth. Rather, they call for short latency, low error rate, and low jitter, while extensive buffering is to be avoided. However, ensuring these parameters in the case of multipath transmissions can be challenging, as the constraints on delay and its variation are difficult to impose [25]. It should also be noted that the principal objective in the design of MPTCP CC algorithms so far was to boost efficiency without compromising fairness [14], rather than cope with delay constraints [15], which are critical for streaming and tactile applications. Although some research on streaming transmission in the multipath framework has been carried out from the perspective of schedulers [16], the literature lacks works investigating the role of CC algorithms in this context. The objective of this paper is to examine whether the popular CC algorithms designed for MPTCP are suitable for streaming applications.

Frequently, the research on network protocols relies on simulations or tests conducted in a closed environment. However, the conclusions drawn from such findings may not be reliable. In this regard, this article tests the parameters of various CC algorithms for their application in a real-world setting using a public network. It follows from the conducted study that the MPTCP CC algorithm BALIA is found capable of achieving the lowest values of path delay and

This work has been performed in the framework of a project “Robustcontrol solutions for multi-channel networked flows” no. 2021/41/B/ST7/00108 financed by the National Science Centre, Poland.

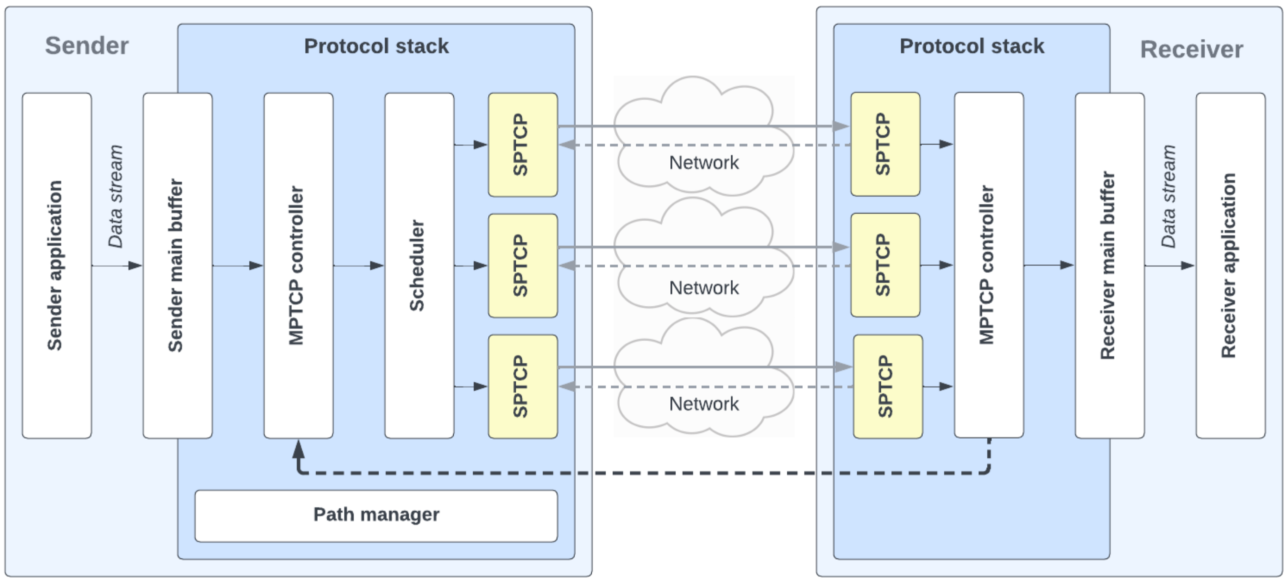


Fig. 1. Data flow in MPTCP architecture: solid lines – data, dashed lines – acknowledgements. Protocol stack and user application are sharing MPTCP buffer, whereas the SPTCP controllers have their own buffers.

head-of-line degree, as well as the highest throughput among four tested algorithms, whereas wVegas performed the worst in the context of streaming applications.

II. Multipath Transmission

The data flow in an MPTCP-capable network is illustrated in Fig. 1. When a request for data transfer is initiated, the standard TCP procedure is used to establish a connection. During the negotiation process, it is determined whether both parties support an appropriate version of the MPTCP protocol. When the multipath extension is available, an attempt is made to open as many transmission channels as possible. The path controller then decides on the number of channels and which paths to use. With multiple transmission channels, communication can be maintained even in the event of a channel failure, which would otherwise lead to a path break.

The stream of data generated by the application first passes through Master Controller, which shapes the data transfer characteristics, and then to Scheduler. Scheduler distributes the data among the active paths that have been established by Path Manager. The data in each path is then transmitted using the single path (SPTCP) controller module. The Scheduler is typically adjusted to achieve a desired strategy, such as reducing power consumption or delay [14], [15]. However, it does not directly influence the intensity of generated traffic. Instead, it responds to signals originating from the logic of the ordinary single-path TCP, which manages the data stream separately for each path. The Master Controller, SPTCP controllers, and Scheduler interact with one another in a complex manner to meet the communication objectives.

It should be noted that the TCP architecture was primarily designed to maximize throughput rather than minimize delay. However, one can shift this priority by an appropriate selection of SPTCP and MPTCP CC algorithms, which exert a significant impact on data transfer dynamics and the resulting quality of data streaming.

III. Congestion Control Algorithms

Currently, the TCP protocol serves as the foundation for the majority of Internet services. However, TCP versions operate differently in various conditions, corresponding to the method of adjusting the transfer speed. The default SPTCP CC algorithm in both Windows and Linux systems is Cubic. In addition to Cubic, the Linux kernel also includes 15 other versions of SPTCP (in alphabetical order): BBR, BIC, CDG, DCTCP, HSTCP, Hybla, Illinois, LP, NV, Reno, Scalable, Vegas, Westwood+, and YeAH. With the deployment of MPTCP, additional algorithms become available [16]: BaLIA, DMCTCP, LIA, OLIA, and wVegas, which are the primary emphasis of this work

None of these algorithms are specifically designed for streaming traffic. It presents a challenge for the transmission of data-intensive multimedia content, where factors such as low latency, low error rates, and low jitter are critical for the intended user experience [24]. There is a need to examine the performance of MPTCP CC algorithms in the context of multipath streaming traffic and to determine which algorithm is best suited for such type of content.

The following CC algorithms were selected for analysis:

On the SPTCP level: Reno and BBR. The legacy Reno algorithm [17] linearly increases the transfer speed and reduces it multiplicatively when packet loss is detected. Although it is no longer used in practical settings, it remains a common reference algorithm for TCP CC evaluation. Furthermore, both LIA and OLIA algorithms can be considered as multipath versions of Reno, since they also adjust the transmission speed in response to drops. In contrast, the BBR [18] algorithm operates by observing the speed at which the network is already delivering the traffic, along with the changes in the smoothed round-trip time (SRTT). Currently, BBR is promoted by Google and gains in popularity as an alternative to the traditional TCP CC algorithms.

TABLE 1.
IMPACT OF CONGESTION CONTROL ALGORITHMS ON STREAMING TRANSMISSION

		LIA		**OLIA**		**BALIA**		**wVegas**	
		Reno	**BBR**	**Reno**	**BBR**	**Reno**	**BBR**	**Reno**	**BBR**
Protocol delay	υ_{av}	139	115	122	118	109	112	266	120
[ms]	υ_{max}	726	565	556	526	491	533	860	556
HoL Degree	ζ_{av}	61	39	43	39	35	35	193	44
[ms]	ζ_{max}	83	425	423	390	353	370	726	426
SRTT [ms]	$\tau_{1,av}$	71	69	71	71	67	68	66	69
path 1	$\tau_{1,max}$	202	223	225	209	226	219	206	223
SRTT [ms]	$\tau_{2,av}$	64	63	65	65	62	64	60	63
path 2	$\tau_{2,max}$	221	229	242	217	229	219	218	218
Mean drop rate	d_1	9.6	10.2	10.3	8.2	7.9	7.8	9.6	12.3
[seg/s]	d_2	8.0	12.2	12.7	13.0	10.5	11.4	8.1	9.2
Throughput	*av*	4.09	4.57	4.47	4.72	5.49	5.47	3.69	3.75
[Mbps]	*max*	10.74	12.03	11.01	11.69	13.66	13.52	8.76	9.59

On the MPTCP level: LIA [19], OLIA [20], BALIA [21], and wVegas [22]. Their design premise is protocol fairness. LIA increases the transmission speed faster than the slowest path, whereas OLIA analyzes the underlying SPTCP control variables and responds to channel disparities and fluctuations. Therefore, OLIA is better adapted for heterogeneous environments. BALIA is a hybrid algorithm that combines the strengths of LIA and OLIA, which allows it to perform well in both homogeneous and heterogeneous environments. The main advantage of BALIA is its ability to dynamically adjust the aggressiveness of CC based on the network conditions. Finally, wVegas is a window-based algorithm that modifies the congestion window size based on the estimated round-trip time and packet loss rate. It has been designed to perform well in high-speed and long-distance connectivity. However, it is less effective in congested or lossy networks.

IV. QUALITY MEASURES

Transmission parameters are affected by numerous factors, such as congestion or buffering, whose impact cannot be predicted *a priori*. Therefore, to fairly evaluate the performance of different CC algorithms, the following quality metrics have been used.

A. Path Delay

The path delay refers to the time it takes a packet to traverse the path from the sender to the receiver. The length of this delay depends on the distance between the sender and receiver, the number of routers and switches along the path, and the degree of congestion on the path.

In the model developed in this paper, the total delay on path *i*, denoted by T_i, comprises the SRTT of this path τ_i and the waiting time for processing the data stream θ_i:

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

The waiting time θ_i is influenced by the scheduler algorithm. The value of τ_i can be reduced by limiting the buffer bloat via a prudent selection of a CC algorithm, as studied in this work.

As the transmission progresses, the delays on individual paths change, and another path may become the "slowest". The path delay is a metric for assessing the quality of service, particularly for streaming traffic. High path delay can result in prolonged buffering and poor user experience.

B. Protocol Delay

The protocol delay is defined as the time that a given piece of data, e.g., a packet, waits in the buffer for stream reassembly. It is measured from the instant when Master Controller receives the user data from the transmit buffer and ends when the corresponding data acknowledgment is received. The protocol delay is equivalent to the delay on the slowest path

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

The average protocol delay is calculated as

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

and maximum protocol delay as

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

The average protocol delay for all the experiment runs is determined as

$$\upsilon_{av} = \frac{1}{R}\sum_{r=1}^{R} \upsilon_{av}^r \tag{5}$$

and the average maximum protocol delay as

$$\upsilon_{\max} = \frac{1}{R}\sum_{r=1}^{R} \upsilon_{\max}^r \,. \tag{6}$$

K represents the number of samples collected in a single experiment run indexed by *r,* while *R* refers to the number of experiment runs. Streaming performance improves with lowering the values of υ_{av} and υ_{max}.

C. HoL Degree

In MPTCP, the Head-of-Line (HoL) blocking degree refers to the number of packets queuing up and waiting to be transmitted in a certain path before the head-of-line packet being delivered.

Real-time applications such as video streaming, online gaming, and video conferencing are highly sensitive to latency and packet loss, as these factors can significantly degrade the user experience. In particular, the HoL blocking [23] can have a substantial impact on the quality of service provided. From the application perspective, the actual visible value is T_{over}, i.e., the delay on the slowest path. Based on that value, the waiting time is defined as

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

Then, the average waiting time

$$\zeta_{av}^{r} = \frac{1}{K}\sum_{k=1}^{K}\left(\max\left(T_{over}(k) - \max_{i\in[1,m]}\tau_i(k), 0\right)\right) \tag{8}$$

and maximum waiting time for each experiment

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

The average waiting time across all the experiments

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

and the average maximum protocol delay across all the experiments

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

D. *Mean Drop Rate*

The mean drop rate is defined as the proportion of packets lost in the course of transmission. Packet drops can happen for a number of causes, such as route failure, network congestion, or packet reordering.

V. Experimental Setup

The test setup depicted in Fig. 2 was employed to assess the effect of CC algorithm interoperability on the quality of streaming content delivery within the MPTCP framework. The created test setup represents a typical data transmission scenario in which a client device connects to a high-end server device to retrieve the content. The server, accessed through a public IP address, is located in a remote data center. A specialized program is utilized to generate the streaming content. Both the client and server devices run under the Linux operating system version 4.19, which had been patched to support MPTCP version 0.95. The client device has two communication interfaces – one connected to an LTE router through an Ethernet cable and the other linked through Wi-Fi 802.11bgn to the same LTE router. Two different LTE networks from different operators were used, with good signal quality ensured. The packets transmitted through one interface arrive at their destination after 10 hops, while packets transmitted through the other arrive after 12 hops. A single scenario lasts 10 seconds and each is repeated 30 times.

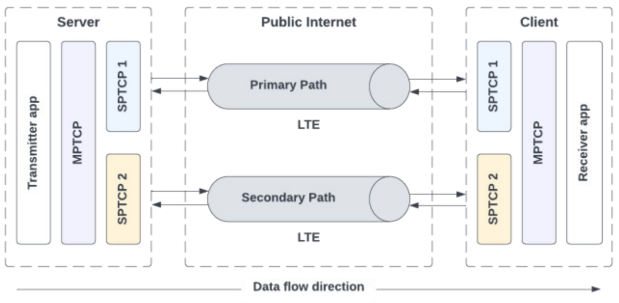


Fig. 2. Experimental setup

VI. Tests and Results

Two CC algorithms: Reno and BBR, were examined for SPTCP, and four algorithms: LIA, OLIA, BALIA, and wVegas, for MPTCP. The tests were performed for each combination, resulting in eight different scenarios. Table 1 summarizes the obtained measurements, whereas Fig. 3 depicts graphically a chosen test run.

The gathered data show that using BALIA at the MPTCP level leads to the best overall performance. This was particularly visible in the case of path delays, where the algorithm achieved the lowest average and maximum delay values. The graphs reveal that the most significant differences occur at the beginning of transmission, where the path delay is nearly three times longer, and the protocol delay and HoL degree are almost five times higher than those observed after stabilization which occurred approximately three seconds after that period. Although peaks resulting from the fluctuation of network parameters were observed, they are negligible after averaging all test runs.

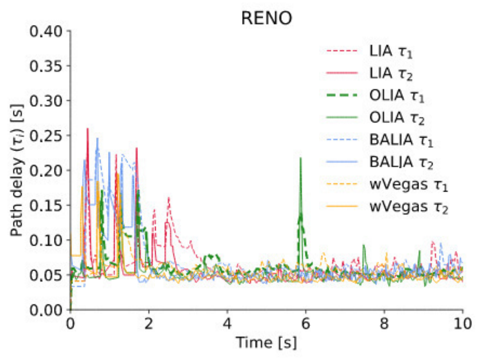
Similar observations apply to the protocol delays, where BALIA also exhibits the lowest average and maximum delays. Moreover, BALIA happened to achieve the lowest HoL degree, implying the smallest proclivity to the multipath queue build-up.

It is worth noting that wVegas underperformed in all the scenarios. Although the path delay was consistent between the different scenarios, the protocol delay was slightly worse for wVegas with BBR, and over two times worse in the scenario with RENO. The HoL degree was over five times larger for wVegas with RENO, and almost one and a half times larger for wVegas with BBR. The throughput was worse when Vegas was used, scoring 33% lower than the other scenarios. Consequently, the wVegas protocol is not recommended for use in MPTCP streaming transmissions.

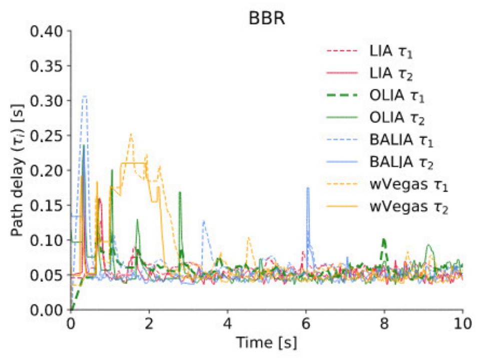
Finally, the throughput data show that the BALIA algorithm was more efficient in utilizing the available resources, resulting in a maximum value that was about 15% (2 Mbps) higher and an average that was about 20% (1 Mbps) higher than the other scenarios.

VII. Conclusions

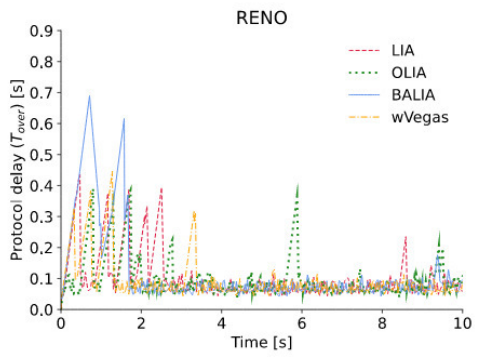
The paper's focus was to investigate how the main MPTCP CC algorithms handle streaming transmission over heterogeneous public networks. The use of multiple communication paths can be an answer for achieving high data speed, low latency, and stable transmission parameters, which are essential for quality streaming. Indeed, it was also



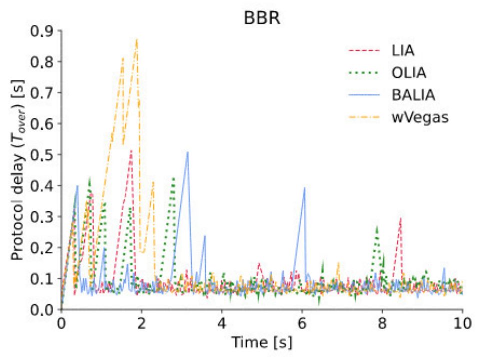
(a) Reno path delay



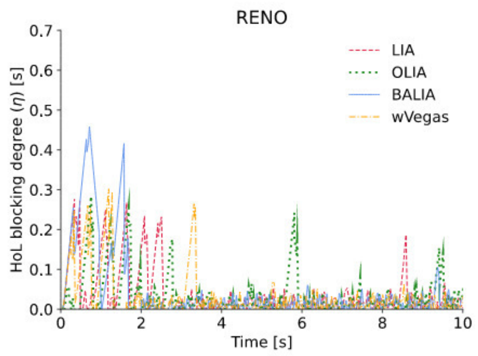
(b) BBR path delay



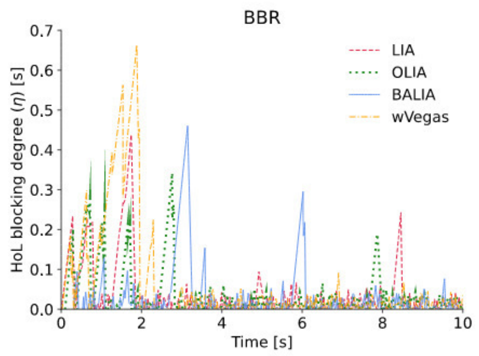
(c) Reno protocol delay



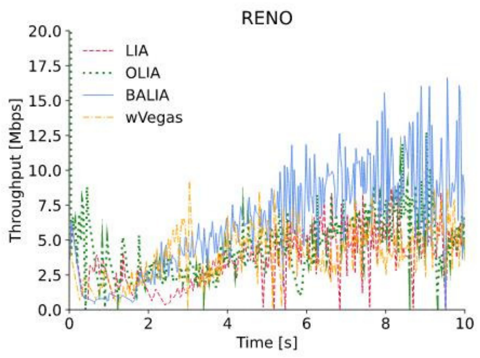
(d) BBR protocol delay



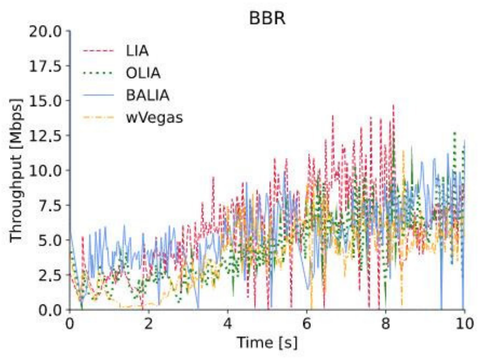
(e) Reno HoL blocking degree



(f) BBR HoL blocking degree



(g) Reno throughput



(h) BBR throughput

Fig. 3 Measured transmission properties – left column Reno, right column BBR acting on the paths

found that the choice of CC protocol is a decisive factor affecting the transmission quality.

The results show that data transfer over multiple paths does increase latency, but this is acceptable and should not negatively affect streaming applications. The BALIA algorithm was found to be the most effective CC algorithm on the MPTCP level and BBR at the path level. Using these two algorithms together provides the lowest latency, lowest error, and highest throughput, making them the best choice out of the off-the-shelf algorithms for efficient streaming over multiple communication channels. In turn, mVegas achieved the lowest transmission parameters and its use in multimedia transmission is ill-advised.

Acknowledgement

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.

References

[1] Cisco, "Cisco Annual Internet Report (2018-2023) White Paper, " 2020. [Online]. Available: https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html

[2] C. Raiciu, M. J. Handley, and D. Wischik, "Coupled Congestion Control for Multipath Transport Protocols, " Request for Comments, no. 6356, RFC Editor, Oct. 2011. [Online]. Available: https://www.rfc-editor.org/info/rfc6356, doi: 10.17487/RFC6356.

[3] M. Morawski and P. Ignaciuk, "Choosing a Proper Control Strategy for Multipath Transmission in Industry 4.0 Applications," *IEEE Transactions on Industrial Informatics*, vol. 18, no. 6, pp. 3609-3619, 2022, doi: 10.1109/TII.2021.3105499.

[4] C. Paasch, S. Ferlin, O. Alay, and O. Bonaventure, "Experimental Evaluation of Multipath TCP Schedulers," *Proceedings of the 2014 ACM SIGCOMM Workshop on Capacity Sharing Workshop*, CSWS '14, Chicago, Illinois, USA, 2014, pp. 27-32, doi: 10.1145/2630088.2631977.

[5] M. Barreiros and P. Lundqvist, "QoS-Enabled Networks: Tools and Foundations," Wiley & Sons, 2016.

[6] M. Morawski and P. Ignaciuk, "Network nodes play a game – a routing alternative in multihop ad-hoc environments," *Comp. Netw.*, vol. 122, pp. 96-104, 2017, doi: 10.1016/j.comnet.2017.04.031.

[7] J. Qadir, A. Ali, K. A. Yau, A. Sathiaseelan, and J. Crowcroft, "Exploiting the power of multiplicity: a holistic survey of network-layer multipath," *IEEE Commun. Surv. Tut.*, vol. 17, no. 4, pp. 2176-2213, "4Q", 2015, doi: 10.1109/COMST.2015.2453941.

[8] M. Li et al., "Multipath Transmission for the Internet: A Survey, " *IEEE Commun. Surv. Tut.*, vol. 18, no. 4, pp. 2887-2925, Q4, 2016 , doi: 10.1109/COMST.2016.2586112.

[9] S. Barré, C. Paasch, and O. Bonaventure, "MultiPath TCP: From theory to practice, " *Tech. Report*, Université Catholique de Louvain, 2011, doi: 10.1007/978-3-642-20757-0_35.

[10] A. Ford, C. Raiciu, M. Handley and O. Bonaventure, "TCP extensions for multipath operation with multiple addresses," RFC 6824, 2013, doi: 10.17487/RFC6824.

[11] S. Barré and C. Paasch, "MultiPath TCP – Linux kernel implementation," http://www.multipath-tcp.org.

[12] M. Morawski and P.Ignaciuk, "A green multipath TCP framework for Industrial Internet of Things applications," *Comp. Netw.*, vol. 187, 107831, 2021, doi: 10.1016/j.comnet.2021.107831.

[13] K. Yedugundla, S. Ferlin, T. Dreibholz, Ö. Alay, N. Kuhn, P. Hurtig, and A. Brunstrom, "Is multipath transport suitable for latency sensitive traffic?, " *Comp. Netw.*, vol. 105, pp. 1-21, 2016, doi: 10.1016/j.comnet.2016.05.008.

[14] C. Paasch, S., Ferlin, O., Alay and O., Bonaventure, "Experimental evaluation of multipath TCP schedulers," *Proc. on ACM SIGCOMM CSWS*, pp. 27–32, 2014, Chicago, USA, doi: 10.1145/2630088.2631977.

[15] M. Morawski and. P. Ignaciuk, "Energy-efficient scheduler for MPTCP data transfer with independent and coupled channels," *Comp. Commun.*, vol. 132, pp. 56-64, 2018, doi: 10.1016/j.comcom.2018.09.008.

[16] C. Xu, J. Zhao, G. Muntean, "Congestion control design for multipath transport protocols: A survey," *IEEE Commun. Surv. Tut.*, vol. 18, no. 4, pp. 2948–2969, 2016, , doi: 10.1109/COMST.2016.2558818.

[17] T. Henderson, S. Floyd, A. Gurtov, and Y. Nishida, "The NewReno modification to TCP's fast recovery algorithm," RFC 6582, Apr. 2012, doi: 10.17487/RFC3782.

[18] N. Cardwell, Y. Cheng, C. S. Gunn, S. H. Yeganeh and V. Jacobson, "BBR: Congestion-Based Congestion Control: Measuring Bottleneck Bandwidth and Round-Trip Propagation Time," Queue, vol. 14, no. 5, pp. 20-53, Sep.-Oct. 2016, doi: 10.1145/3012426.3022184.

[19] C. Raiciu, M. J. Handley and D. Wischik, "Coupled Congestion Control for Multipath Transport Protocols," RFC 6356, RFC Editor, Oct. 2011, pp. 1-12, doi: 10.17487/RFC6356.

[20] N. Gast, R. Khalili, J.-Y. Le Boudec and M. Popovic, "Opportunistic Linked-Increases Congestion Control Algorithm for MPTCP," *Proceedings of the 13th International IFIP Networking Conference (Networking)*, Trondheim, Norway, 2014, pp. 1-9.

[21] Q. Peng, A. Walid, J. Hwang, and S. H. Low, "Multipath TCP: Analysis, Design, and Implementation, " IEEE/ACM Transactions on Networking, vol. 24, no. 1, pp. 596-609, 2016, doi: 10.1109/TNET.2014.2379698.

[22] Y. Cao, M. Xu, and X. Fu, "Delay-based congestion control for multipath TCP, ", 2012 20th *IEEE International Conference on Network Protocols (ICNP)*, pp. 1-10, 2012, doi: 10.1109/ICNP.2012.6459943.

[23] R. Khalili, N. Gast, M. Popovic, and J.-Y. Le Boudec, "MPTCP is not Pareto-optimal: performance issues and a possible solution," *IEEE/ACM Trans. Netw.*, vol. 21, no 5, pp. 1651-1665, 2013, doi: 10.1109/TNET.2013.2274462.

[24] S. Grzyb and P. Orłowski, "Congestion feedback control for computer networks with bandwidth estimation, ", *2015 20th International Conference on Methods and Models in Automation and Robotics (MMAR)*, pp. 1151-1156, 2015, doi: 10.1109/MMAR.2015.7284041.