MITSUBISHI ELECTRIC RESEARCH LABORATORIES
https://www.merl.com

Learning Based Scheduling and Adaptive Congestion Control for Multipath QUIC

Das, Souryendu; Guo, Jianlin; Parsons, Kieran; Nagai, Yukimasa; Sumi, Takenori; Sakaguchi, Naotaka; Orlik, Philip V.; Kalafatis, Stavros

TR2025-083 June 10, 2025

Abstract

As the number of devices with multiple communication interfaces increases, the connection redundancy is being considered for efficient bandwidth utilization and improved reliability. Accordingly, multipath transport technologies are attracting attention. This paper investigates the Multipath Quick UDP Internet Connection (MPQUIC) transport protocol with the goal of enhancing data throughput and packet transmission efficiency. We introduce a novel Blocking Probability (BLP) path scheduler, which leverages a learned blocking threshold for probability-based blocking decision-making, and an innovative congestion controller, which not only dynamically adjusts congestion window size but also optimizes packet size. Our simulations demonstrate that BLP significantly improves network through- put, packet delay time, and overall transmission efficiency in both homogeneous and heterogeneous network environments. By outperforming benchmark schedulers, BLP showcases the potential of advanced scheduling strategies to adapt to network dynamics, ensuring reliable and efficient data delivery.

IEEE International Conference on Communications Workshops (ICC) Workshop 2025

Mitsubishi Electric Research Laboratories, Inc.
201 Broadway, Cambridge, Massachusetts 02139

Learning Based Scheduling and Adaptive Congestion Control for Multipath QUIC

Souryendu Das*‡, Jianlin Guo*, Kieran Parsons*, Yukimasa Nagai†, Takenori Sumi†, Naotaka Sakaguchi†, Philip Orlik*, Stavros Kalafatis‡
*Mitsubishi Electric Research Laboratories (MERL), Cambridge, MA 02139, USA
†Information Technology R&D Center, Mitsubishi Electric Corporation, Kamakura, Kanagawa 2478501, Japan
‡Texas A&M University, 400 Bizell Street, College Station, TX 77843, USA

***Abstract*—As the number of devices with multiple communication interfaces increases, the connection redundancy is being considered for efficient bandwidth utilization and improved reliability. Accordingly, multipath transport technologies are attracting attention. This paper investigates the Multipath Quick UDP Internet Connection (MPQUIC) transport protocol with the goal of enhancing data throughput and packet transmission efficiency. We introduce a novel Blocking Probability (BLP) path scheduler, which leverages a learned blocking threshold for probability-based blocking decision-making, and an innovative congestion controller, which not only dynamically adjusts congestion window size but also optimizes packet size. Our simulations demonstrate that BLP significantly improves network throughput, packet delay time, and overall transmission efficiency in both homogeneous and heterogeneous network environments. By outperforming benchmark schedulers, BLP showcases the potential of advanced scheduling strategies to adapt to network dynamics, ensuring reliable and efficient data delivery.**

***Index Terms*—Multipath QUIC, blocking-based path scheduling, blocking threshold learning, adaptive congestion control, optimal packet size, and data transmission efficiency.**

I. Introduction

The rise of multi-homed devices like smartphones, tablets, and smart meters has increased the demand for efficient multipath transport protocols. Traditional protocols like TCP and UDP struggle with utilizing multiple network interfaces, leading to poor throughput and reliability. Multipath TCP (MPTCP) improves these aspects by allowing multiple paths for a single connection, enhancing throughput and network efficiency, as demonstrated by its use in iOS 11 for Siri.

However, MPTCP's kernel-level implementation requires operating system updates and limits flexibility. In addition, its three-way handshake requires a longer path setup time, and its more extended frame header reduces application data throughput. To address these issues, Google introduced Quick UDP Internet Connection (QUIC), a transport protocol for HTTP/3 traffic that operates at the application layer. This design choice allows for quicker deployment and updates, bypassing the need for kernel-level changes.

Building on the foundational MPTCP and QUIC protocols, Multipath QUIC (MPQUIC) extends QUIC to support multiple paths, leveraging features such as stream multiplexing and reduced handshake latency. MPQUIC offers several advantages over MPTCP:

- **Enhanced Flexibility:** MPQUIC can utilize application-layer data like bit rates and buffer states to optimize performance based on specific needs, a level of customization not easily achievable with MPTCP.
- **Improved Efficiency with 0-RTT:** MPQUIC supports zero round-trip time (0-RTT) for faster path setup compared to MPTCP's three-way handshake, reducing additional latency.
- **Efficient Frame Structure:** MPQUIC uses a more efficient frame structure to allow the frame to carry more application data.
- **Enhanced Security:** MPQUIC includes mandatory encryption of traffic, making it a more secure alternative to MPTCP
- **Extensibility and Future-Proofing:** MPQUIC's architecture is designed for easy integration of future enhancements, allowing it to adapt seamlessly to evolving network requirements and technologies.

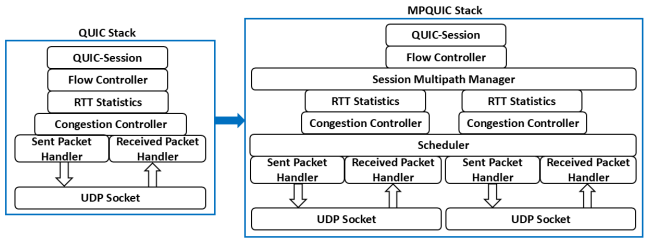


Fig. 1: Protocol Stacks of QUIC and MPQUIC Extension

Fig. 1 illustrates the protocol extension of QUIC to

This work was done while Souryendu Das was working at Mitsubishi Electric Research Laboratories (MERL) as an intern.

MPQUIC, incorporating a Session Multipath Manager, multiple congestion controllers, and a scheduler. These additions enable efficient handling of various paths within a single session using multiple UDP sockets, similar to MPTCP's use of TCP sockets, but with application-layer flexibility.

Our work introduces a learning-based scheduler that adapts scheduling decisions to real-time network conditions and historical data, enhancing MPQUIC's efficiency. We propose a novel blocking probability (BLP) scheduler with a reward and penalty model for congestion window and packet size adjustment. NS3-based simulations demonstrate improved performance metrics compared to baseline MPQUIC schedulers.

This paper is structured as follows: Section II covers related works, Section III details the BLP scheduling algorithm with the learned blocking threshold, Section IV describes the reward and penalty model for congestion window ($cwnd$) size and packet size adjustment, Section V presents the comprehensive performance evaluation results, and Section VI concludes the paper.

II. Related Works

Multipath transport protocols like MPTCP enhance throughput and reliability by using multiple network paths. MPTCP aggregates bandwidth and shifts congestion but faces challenges such as Head-of-Line (HoL) blocking, i.e., missing packets block the processing of the arrived packets, and suboptimal performance in heterogeneous environments [1]. QUIC, introduced by Google, operates at the application layer, reducing latency and improving performance [2]. Scheduling is crucial for multipath transport. The simplest scheduler is Round Robin, but it may cause high latency when paths are highly heterogeneous. Therefore, MPQUIC uses the Minimal RTT (MinRTT) scheduler by default, which sends data on the path with the lowest RTT. However, MinRTT faces a HoL blocking issue. To improve MinRTT, advanced MPQUIC schedulers, such as BLEST [3], ECF [4], and Peekboo [5], are proposed to minimize HoL blocking and optimize throughput. BLEST estimates blocking potential and incorporates a waiting mechanism, while ECF maximizes fast path utilization. Peekaboo, a learning-based scheduler, adapts to heterogeneous path conditions using an online adaptive learning mechanism. Recently, other MPQUIC schedulers have been proposed. Khan *et al.* [6] developed a delay-based scheduler for MPQUIC. Zeng *et al.* [7] optimized MPQUIC transmission over heterogeneous paths with the Estimated Transfer Completion scheduler. Surveys [8] highlight MPQUIC's potential to meet stringent requirements in 5G environments. There are also other relevant schedulers designed for MPTCP, e.g., delay-aware scheduler DAPS [9], decoupled scheduler DEMS [10], and loss-aware scheduler [11]. Congestion control is also crucial for multipath transport. MPQUIC can use the OLIA congestion control algorithm. Experiments in work [1] shows that MPQUIC outperforms MPTCP in all aspects except fairness.

Our work advances blocking-based scheduling by proposing the BLP scheduler for MPQUIC to enhance data transmission efficiency and network throughput.

III. Learning-Based BLP Scheduler

Multipath characteristics can be highly heterogeneous, especially in a wireless environment where many factors, such as path bandwidth, network topology, and channel access mechanism, can significantly impact path characteristics. Traditional Round Robin and MinRTT face HoL blocking. Recent BLEST, ECF, and Peekaboo address this issue by considering more path characteristics, but they are inconsistent. To improve these methods, we introduce a novel BLP scheduler, which switches between MinRTT mode and blocking mode to adapt to dynamic network conditions while using a learning-based blocking threshold and a dynamically adjusted congestion window size. The proposed BLP scheduling method overcomes the limitations of the schedulers above by mitigating non-congestive jitter and ensuring efficient bandwidth distribution among competing flows, thereby enhancing data transmission efficiency and network throughput.

BLEST [3] was proposed for MPTCP. It first estimates the amount of data X to be sent on the fast path and then checks whether X fits into the MPTCP send window. If the X does not fit the send window, the slow path is blocked. The X estimation, however, can be inaccurate. To correct the X estimation, a correction factor λ is introduced to scale X. The λ is initially set to 1 and then updated based on HoL blocking using a fixed variance $\Delta\lambda$ as

- λ increases by $\Delta\lambda$ if HoL blocking occurs.
- λ decreases by $\Delta\lambda$ if HoL blocking does not occur.

Derived from BLEST for MPQUIC, BLP's correction factor λ improves over BLEST by incorporating a blocking probability factor *pf* in its calculations with *pf* being defined as

$$pf = \frac{B_s}{B_s + B_f} \times \frac{rtt_s}{rtt_f} \times \frac{1}{nc}, \tag{1}$$

where B_s and B_f are bytes in flight on slow path and fast path, respectively, rtt_s and rtt_f are RTTs of slow path and fast path, respectively, and nc is the number of scheduling cycles for next data segment. BLP dynamically calculates its blocking probability factor, focusing on bytes in flight, the RTT ratio, the number of flows, and the number of scheduling cycles. This probability factor represents the likelihood of slow path blocking based on whether the MPQUIC send window can accommodate the bytes on the fast path during the slow path RTT period. Blocking occurs with a calculated probability rather than an absolute decision, as in BLEST [3].

Blocking Threshold Learning: The BLP scheduler employs a learning-based approach to optimize blocking probability. The blocking threshold (BT) learning process includes:

- **Data Collection:** Gathering data under various network conditions, focusing on features like lost packets, total bytes dropped, retransmissions, average throughput, mean delay, mean jitter, inter-packet arrival time, packet delivery ratio, and correction factor variance.
- **Preprocessing Using SMOTE:** Using Synthetic Minority Over-sampling Technique (SMOTE) to ensure data quality and balance, preventing model bias.
- **Model Training:** Using a stacked classifier approach where random forest and gradient boosting are base learners and logistic regression is the meta-learner, combining outputs for robust predictions.
- **Hyperparameter Tuning:** Employing RandomizedSearchCV to fine-tune hyperparameters such as the number of estimators, maximum depth, and learning rate, optimizing model performance through cross-validation.

BLP Scheduling Algorithm: Our proposed BLP scheduling is detailed in Algorithm 1, where $lpid$ is the ID of last used path, snd[x] indicates next subflow will be sent on path "x", $fpid$ is fast path ID, $spid$ is slow path ID, tx is transmission buffer available on a MPQUIC connection, mss_s is the maximum segment size of slow path, mss_f is the maximum segment size of fast path, and $cwnd_f$ is congestion window of fast path. The BLP algorithm switches between MinRTT mode (for low path heterogeneity) and blocking mode (for high path heterogeneity) to make learning-based blocking decisions on the slow path. This dynamic mechanism adjusts blocking probability based on real-time network conditions, improving data transmission efficiency and throughput across varied scenarios.

Algorithm 1 Blocking Probability Scheduling Algorithm

if subflows_size ≤ 1 **then**
  $lpid = 0$
  snd[$lpid$] = TRUE
else
  if next_subflow $\rightarrow$ lastrtt $== 0$ **then**
    $lpid = (lpid + 1)/$subflows_size
    snd[$lpid$] = TRUE
  else
    $fpid = 1$
    $spid = 0$
    if subflow[$spid$] $\rightarrow$ lastrtt $>$ subflow[$fpid$] $\rightarrow$ lastrtt **then**
      $rtt_s =$ current_subflow $\rightarrow$ lastrtt
      $rtt_f =$ next_subflow $\rightarrow$ lastrtt
      $spid = 0$
      $fpid = 1$
    else
      $rtt_s =$ next_subflow $\rightarrow$ lastrtt
      $rtt_f =$ current_subflow $\rightarrow$ lastrtt
      $spid = 1$
      $fpid = 0$
    if windowavailable($fpid$) > 0 **then**
      $lpid = fpid$
    else
      $bytes\_left = tx - (B_s + mss_s)$
      $nc = \lceil \frac{\text{bytes\_left}}{\text{ss}} \rceil$
      $rtt\_f = \frac{rtt_s}{rtt_f}$
      if rtt_f $<$ BT **then**
        $lpid = spid$ //MinRTT mode
      else
        $sm = pf \times mss_f \times rtt\_f \times (cwnd_f + \frac{rtt\_f}{2})$
        $\lambda = pf * (\lambda + \Delta\lambda)$
        if $sm * \lambda >$ bytes_left **then**
          $lpid = fpid$ //Blocking mode
        else
          $lpid = spid$ //MinRTT mode
    snd[$lpid$] = TRUE
Output: snd[$lpid$]

IV. Reward and Penalty Model for BLP Congestion Window Adaptation

As an extension of the QUIC, MPQUIC is a connection-based protocol where the receiver acknowledges packets. The congestion window ($cwnd$) defines the maximum amount of data that can be sent on a path without being acknowledged. Therefore, the $cwnd$ optimization is critical. We propose a reward and penalty-based congestion control method to adapt $cwnd$ size based on network conditions and path properties.

Recall that MPQUIC applies the delayed acknowledgment mechanism that aggregates multiple ACKs, which can potentially leave packets temporarily unacknowledged. The unacknowledged packets include packets still in-flight due to path congestion, especially over multi-hop wireless paths, and packets lost due to communication layer transmission failure, e.g., the maximum number of the transmission attempts has been reached. In MPQUIC, lost packets are retransmitted with new sequence numbers, and path heterogeneity can result in earlier transmitted packets remaining unacknowledged. Thus, not all transmitted packets are promptly acknowledged, and the unacknowledged packets indicate path congestion.

Degree of Congestion: The degree of congestion α is the ratio of unacknowledged packets to the total packets scheduled in the congestion window, defined as

$$\alpha = \frac{N_s - N_r}{N_s}, \tag{2}$$

where N_s is the $cwnd$ size in packets and N_r is number of acknowledged packets. The α values range from 0 (all packets acknowledged) to 1 (no ACK received, complete congestion).

Congestion Window Adjustment Model: The $cwnd$ adjustment is based on successful ACK (reward) or packet loss/retransmission (penalty). If the ACK indicates successful packet reception, the $cwnd$ is adjusted as

$$N_s = N_s \left(1 + \frac{\alpha l}{B}\right), \tag{3}$$

and otherwise, the $cwnd$ is adjusted as

$$N_s = N_s \left(1 - \frac{\alpha l}{B}\right), \tag{4}$$

where l is fraction of packets lost and B is the maximum segment size (mss). Denote as N_f and N_l the number of packets in flight and the number of packets lost, respectively. We have $N_s - N_r = N_f + N_l$, which implies $\alpha = \frac{N_f}{N_s} + l$. Let n be number of flows in flight, ss be stream size, and s be packet size. Considering that $N_f = \frac{n \times ss}{s}$, we get $l = \alpha - \frac{n \times ss}{s \times N_s}$.

Success Probability: Based on works [12] and [13], we model the probability of incurring loss in the next scheduling cycle to have an exponential filter function with a tuning factor that would determine the steepness of the loss rate impact on the loss probability as

$$\text{P(l)} = \frac{1}{1 + e^{-(\alpha(l - BT)) \times \frac{N_l + N_s}{N_s}}}. \tag{5}$$

Thus, the probability of success for all remaining cycles is given by

$$\text{P(S, nc)} = \left(1 - \left(\frac{1}{1 + e^{-(\alpha(l - BT)) \times \frac{N_l + N_s}{N_s}}}\right)\right)^{nc}. \tag{6}$$

As a result, the success probability over n flows is

$$\text{S(n)} = \left(1 - \left(\frac{1}{1 + e^{-(\alpha(l - BT)) \times \frac{N_l + N_s}{N_s}}}\right)\right)^{n \times nc}. \tag{7}$$

Note that it is efficient to use the maximum packet size in reliable networks like wired networks. However, packet size optimization is necessary in unreliable networks like wireless networks. Therefore, our objective is to maximize success probability with respect to packet size, which gives optimal packet size and a fraction of packets lost

$$\begin{aligned} s &= \left(\alpha - \frac{\alpha^2 - \alpha BT - 1}{\alpha}\right) \frac{N_l + N_s}{N_s}, \\ l &= \alpha \left(1 - \frac{n \times ss}{1 + (BT \times (N_l + N_s))}\right). \end{aligned} \tag{8}$$

Therefore, we get $cwnd$ gain in case of successful ACK as

$$N_s = N_s \left(1 + \frac{\alpha^2 \left(1 - \frac{n \times ss}{1 + (BT \times (N_l + N_s))}\right)}{B}\right), \tag{9}$$

and $cwnd$ reduction in case of packet drop/retransmission as

$$N_s = N_s \left(1 - \frac{\alpha^2 \left(1 - \frac{n \times ss}{1 + (BT \times (N_l + N_s))}\right)}{B}\right). \tag{10}$$

Fig. 2 shows the interaction among the BLP scheduler, congestion controller, and blocking threshold learner. The BLP scheduler optimizes performance by dynamically adjusting the $cwnd$ based on network conditions (wired or wireless communications, etc.), path properties (bandwidth, RTT, etc.), and a learned blocking threshold. It uses a reward mechanism to increase $cwnd$ on successful ACKs and a penalty mechanism to decrease it on packet drops or retransmissions, preventing congestion. A feedback loop adapts $cwnd$ and scheduling decisions to current network conditions.

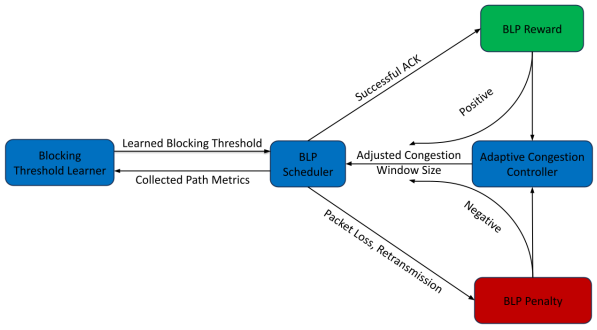
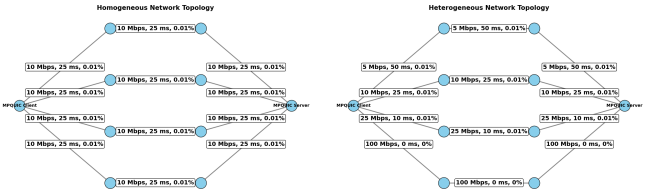


Fig. 2: BLP Scheduler with Adaptive Congestion Window and Learned Blocking Threshold

V. Performance Evaluation

We use an NS3-based MPQUIC simulator [14] to evaluate the performance of the BLP scheduler against benchmarks like Round Robin, MinRTT, BLEST, ECF, and Peekaboo in both homogeneous and heterogeneous network topologies. Simulations involved a 25 Mb file transfer under various network conditions. In the homogeneous case (0.01% loss rate), we tested six bandwidth-delay combinations (5Mbps-50ms, 10 Mbps-25ms, 25Mbps-10ms, 50Mbps-5ms, 125Mbps-2ms, 250 Mbps-1ms), with all links sharing the same setting, as shown in Fig. 3(a). For the heterogeneous case, we permuted these settings with varying flow durations and packet sizes, illustrated in Fig. 3(b).

An error model is randomly applied to low-speed links, considering errors only in the forward path. Benchmark schedulers use OLIA congestion control, while BLP applies adaptive congestion control. Both BLEST and BLP adjust blocking predictions with a correction factor λ, evaluated over five $\Delta\lambda$ variances. Each path setting runs 10 seeds, yielding 300 results for BLEST and BLP and 60 for other schedulers.



(a) Homogeneous Paths (b) Heterogeneous Paths

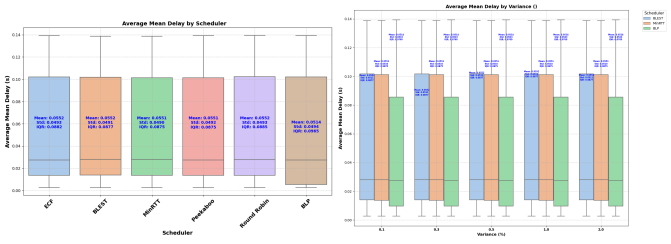
Fig. 3: Simulation Path Setting Illustration

Results are averaged for performance metrics, where in Fig. 4(a), Fig. 5(a), Fig. 7(a), Fig. 8(a), and Fig. 9(a), $\Delta\lambda = 0$.

A. Homogeneous Path Evaluation

Simulations were conducted using six predefined path settings one by one. For BLEST and BLP, the correction factor λ was applied. While the variance $\Delta\lambda$ improves BLP's performance, it does not affect BLEST's performance. This indicates the contributions of our probability factor pf, learned blocking threshold (BT), and adaptive congestion control. Additionally, the variation of $\Delta\lambda$ does not impact BLP's performance in homogeneous paths since no path offers an advantage. Although MinRTT does not use correction factor λ, we included MinRTT in Fig. 4(b), Fig. 5(b), Fig. 7(b), Fig. 8(b), and Fig. 9(b) as it is the default MPQUIC scheduler, and BLP incorporates a MinRTT mode based on the BT.

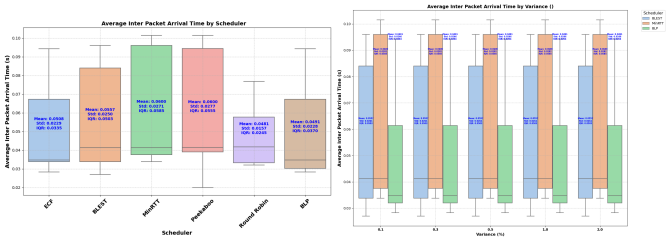
1) Average Data Packet Delay: Fig. 4(a) illustrates the average data packet delay. BLP achieves the lowest mean value of 0.0514s, which is 6.7% better than 0.0551s of the next best schedulers, MinRTT and Peekaboo. With the λ variance, BLP further reduces the IQR from 0.0965s to the smallest value of 0.076s and Q3 from 0.11s to 0.086s as shown in Fig. 4(b). The results indicate that BLP is more stable and effective in minimizing delay in homogeneous network conditions, which is suitable for latency-sensitive applications.



(a) Average Delay by Scheduler (b) Average Delay by λ Variance

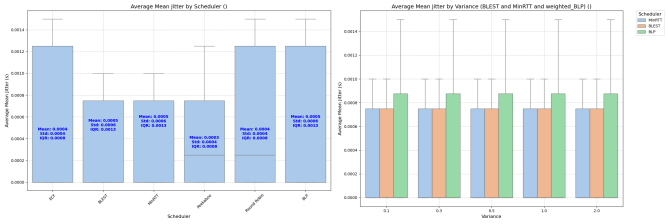
Fig. 4: Average Data Packet Delay by Scheduler and λ Variance

2) Average Inter-Packet Arrival Time: Fig. 5(a) shows the average inter-packet arrival time. BLP achieves the second lowest mean value of 0.0491s compared to 0.0481s by Round Robin. However, the lower median of BLP means that BLP has smaller inter-packet arrival times than Round Robin, which indicates that BLP is competitive with Round Robin overall. The λ variance further improves BLP performance by reducing the IQR from 0.037s to 0.0294s and Q3 from 0.068s to 0.061s, as shown in Fig. 5(b).



(a) Arrival Time by Scheduler (b) Arrival Time by λ Variance

Fig. 5: Average Inter-Packet Arrival Time by Scheduler and λ Variance



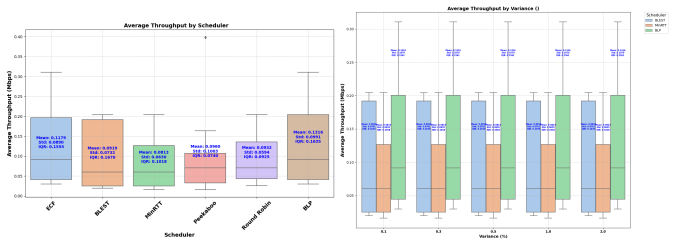
(a) Average Mean Jitter by Scheduler (b) Average Mean Jitter by Variance

Fig. 6: Jitter Variation

3) Average Mean Jitter: Figure 6a shows the average mean jitter. MinRTT and BLEST perform best, achieving a mean jitter of 0.0005 seconds. This metric is crucial for applications requiring consistent packet delivery timing, such as video streaming. Despite BLP's strong performance, it does not lead in this specific metric. BLP exhibits a higher average mean jitter than MinRTT and BLEST from Fig. 6b. While BLP excels in throughput, it incurs higher jitter. This variability in packet transmission timing may affect applications sensitive to jitter, suggesting further optimization of BLP is needed.

4) Average Throughput: Data throughput is a key performance metric for multipath technologies. Fig. 7(a) illustrates the average data throughput. BLP outperforms all benchmark schedulers, achieving the highest mean value of 0.1316 Mbps, which is 11.6% better than 0.1179 Mbps of the following best scheduler, ECF, and the highest Q3 value of 0.21 Mbps, indicating its effectiveness in maximizing data transmission

rates. The λ variance further reduces the IQR from 0.1635 Mbps to 0.1561 Mbps, as shown in Fig. 7(b).



(a) Average Throughput by Scheduler(b) Average Throughput by λ Variance

Fig. 7: Average Throughput by Scheduler and λ Variance

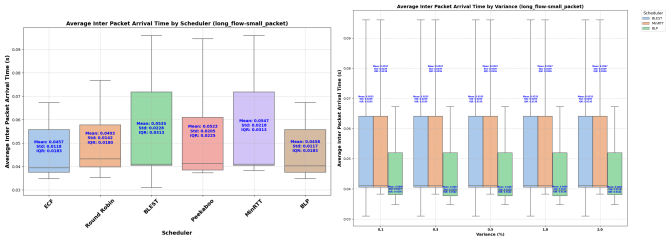
5) Total Delay Time: Fig. 4(a) to Fig. 7(b) show performance metrics averaged over six path settings. This simulation runs a two-path topology with each path having 5 Mbps bandwidth and 50ms delay, creating constrained conditions for all schedulers. This setup evaluates total delay time, emphasizing efficient scheduling. The total delay times for Round Robin, MinRTT, BLEST, ECF, Peekaboo, and BLP were 951.24 ms, 1001.75 ms, 970.31 ms, 1161.68 ms, 1001.75 ms, and 656.88 ms, respectively. BLP scheduler achieves the lowest total delay, outperforming the second best scheduler, BLEST, by 32.3%.

B. Heterogeneous Path Evaluation

The simulations were conducted for the link settings shown in Fig. 3b. We also vary the traffic based on flow length and packet size. Long flow is 1000000 Packets, and short flow is 512000 Packets, whereas large packet size is 10,000 Bytes and small packet size is 100 Bytes. Due to space limitations, we present results for the "long flow and small packet" scenario. In all cases, we observe that λ improves BLP's performance through its subtle variation in $\Delta\lambda$.

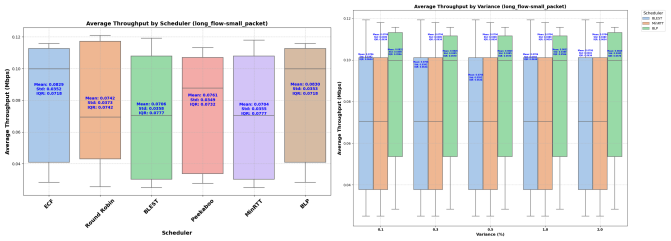
1) Average Inter-Packet Arrival Time: Fig. 8(a) shows that BLP and ECF are the better schedulers with mean values of 0.0458s and 0.0457s, respectively. The variance $\Delta\lambda = 0.003$ further improves BLP mean value to the best value of 0.0457s, reduces the IQR from 0.0183s to the smallest value of 0.0142s and Q3 from 0.056s to 0.052s as shown in Fig. 8(b), suggesting its efficiency in managing packet transmission windows in heterogeneous conditions. The smallest IQR indicates that BLP is more stable. In addition, Fig. 8(b) also illustrates that the performance metrics of BLP vary with respect to the variance $\Delta\lambda$, indicating that BLP reveals the impact of heterogeneous paths.

2) Average Throughput: Fig. 9(a) shows that BLP is the best scheduler with the highest mean average throughput of



(a) Arrival Time by Scheduler (b) Arrival Time by λ Variance

Fig. 8: Average Inter-Packet Arrival Time by Scheduler and λ Variance



(a) Average Throughput by Scheduler(b) Average Throughput by λ Variance

Fig. 9: Average Throughput by Scheduler and λ Variance

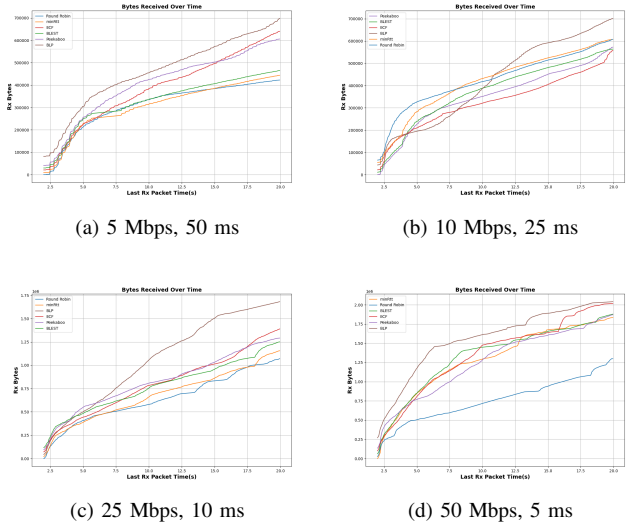
0.0830 Mbps, making it suitable for high-throughput applications. Although the Q1 and Q3 of Round Robin are higher than those of BLP, its lower median value indicates lower throughput values, resulting in a lower mean value of 0.0742 Mbps. The variance $\Delta\lambda = 0.001$ further improves BLP mean average throughput to 0.0832 Mbps, reduces the IQR from 0.0718 Mbps to the smallest value of 0.0579 Mbps and increases the Q1 from 0.041 Mbps to 0.054 Mbps as shown in Fig. 9(b), outperforming all benchmarks. These results show that BLP not only achieves the highest throughput but also accomplishes the smallest IQR, indicating that BLP is more stable. In addition, Fig. 9(b) also shows that the performance metrics of BLP vary with respect to the variance $\Delta\lambda$.

3) Results for Other Packet and Flow Combinations:

- **Long Flow, Large Packet:** BLP achieves high throughput and low inter-packet arrival times by efficiently managing paths and adapting to network conditions.
- **Short Flow, Large Packet:** BLP sustains high throughput and handles variability in short flows through effective, learning-based adjustments.
- **Short Flow, Small Packet:** BLP ensures robust throughput despite frequent decision points required by small packets in short flows.

4) Analysis of Packet Delivery Over Time: Fig. 10(a) to Fig. 10(d) depict the amount of data delivery with respect to time by various schedulers over a fixed simulation time of 20s.

BLP scheduler consistently achieves the highest throughput and the highest data packet delivery rate, as indicated by its steep slope and final Rx Bytes, respectively. It delivers 3.03% more data at 5 Mbps, 9.67% more data at 10 Mbps, 14.28% more data at 25 Mbps, and 2.63% more data at 50 Mbps compared to the second-best scheduler. The dynamic CWND adjustment facilitates aggressive yet controlled data transmission, maximizing the throughput. These results also show that BLP is a more reliable scheduler.



(a) 5 Mbps, 50 ms (b) 10 Mbps, 25 ms

(c) 25 Mbps, 10 ms (d) 50 Mbps, 5 ms

Fig. 10: Bytes Received Over Time for Various Network Conditions

5) Total Delay Time: The simulation uses Path 1 with 5 Mbps bandwidth and 50ms delay and Path 2 with 50 Mbps bandwidth and no delay, ensuring all schedulers can deliver 25 Mb data within the simulation time of 20s. Total delay times for Round Robin, MinRTT, BLEST, ECF, Peekaboo, and BLP are 1208.92 ms, 408.92 ms, 384.97 ms, 454.92 ms, 408.92 ms, and 382.77 ms, respectively. BLP achieves the shortest total delay, demonstrating its efficiency and effectiveness in utilizing the faster path.

C. TCP and MPQUIC Cross-Traffic Impact Evaluation

In this simulation, we ran TCP and MPQUIC in parallel on one path while varying conditions on another path. This setup assesses the mutual impact of TCP and MPQUIC traffic, revealing that TCP suffered 68,837 flow interruptions across six schedulers, compared to only 553 for MPQUIC. This indicates that MPQUIC is significantly more resilient to TCP cross-traffic, making it better suited for real-world environments.

VI. Conclusion

This paper studies Multipath QUIC (MPQUIC), an efficient multipath transport protocol suite developed by the IETF. We propose a Blocking Probability (BLP) scheduler and a dynamic congestion controller. The BLP scheduler uses probability-based decisions and dynamically adjusts the congestion window, learning a blocking threshold from network conditions and historical metrics. Simulations validate significant performance gains, with BLP consistently surpassing benchmark schedulers in throughput, delivery rate, delay, and inter-packet arrival time. It excels in high-throughput, low-latency applications by leveraging a learning-based approach and real-time feedback for stability. BLP enhances MPQUIC's suite, ensuring efficient data transmission across diverse network scenarios.

References

[1] Morawski, M. and Ignaciuk, P., "MPTCP or MPQUIC - Which One is Better for General-Purpose Networking," in *25th International Conference on System Theory, Control and Computing (ICSTCC)*, 2021.

[2] Langley, A, Riddoch, A., et al., "The QUIC Transport Protocol: Design and Internet-Scale Deployment," in *Proceedings of the ACM Special Interest Group on Data Communication (SIGCOMM)*, 2017.

[3] S. Ferlin, T. Dreibholz, and O. Bonaventure, "BLEST: Blocking Estimation-based MPTCP Scheduler for Heterogeneous Networks," in *Proceedings of the IFIP Networking Conference*, 2017.

[4] Q. De Coninck and O. Bonaventure, "ECF: An MPTCP Path Scheduler to Manage Heterogeneous Paths," in *IEEE Transactions on Network and Service Management*, vol. 14, no. 2, 2017.

[5] L. Peng, L. Wang, S. Zhao, and Y. Liu, "Peekaboo: Learning-Based Multipath Scheduling for Dynamic Heterogeneous Environments," in *IEEE Transactions on Mobile Computing*, vol. 19, no. 4, 2020.

[6] Khan, F., Fernández, F., Diez, L., and Aguero, R., "Exploiting QUIC multi-streaming over NTN: Delay-based scheduling policies," in *IEEE Global Communications Conference (GLOBECOM)*, 2023.

[7] Zeng, H., Cui, L., et al., "Optimizing multipath QUIC transmission over heterogeneous paths," in *Computer Networks*, 2022.

[8] Wu, H., Ferlin, S., Caso, G., Alay, Ö., and Brunstrom, A., "A survey on multipath transport protocols towards 5G access traffic steering, switching and splitting", in *IEEE Access*, 2021.

[9] Kuhn, N., Lochin, E., Mifdaoui, A., et al., "DAPS: Intelligent delay-aware packet scheduling for multipath transport", in *IEEE International Conference on Communications (ICC)*, 2017.

[10] Guo, Y.E., Nikravesh, A., Mao, Z.M., Qian, F., and Sen, S., "Accelerating Multipath Transport Through Balanced Subflow Completion", in *23rd Annual International Conference on Mobile Computing and Networking (MobiCom)*, 2017.

[11] Guo, J., Parsons, K., Nagai, Y., Sumi, T., Sakaguchi, N., Tsuchida, H., Wang, P., and Orlik, P., "Multipath TCP Over Multi-Hop Heterogeneous Wireless IoT Networks", in *IEEE International Conference on Communications (ICC)*, 2024.

[12] Lorincz, J., Klarin, Z., and Ožegović, J., "A comprehensive overview of TCP congestion control in 5G networks: Research challenges and future perspectives", *Sensors*, 21(13), 2021.

[13] Mertens, J., "Developing a Hybrid Stochastic and Deterministic Alpha, Beta, Gamma Filter with SNR for Sensorless Control Using Propagation of Uncertainties with a Two Phase Stepper Motor as an Example", in *National Technical Reports Library*, 2020.

[14] Shu, S., Yang, W., Pan, J., and Cai, L., "A multipath extension to the QUIC module for ns-3", in *2023 Workshop on ns-3*, 2023.