CSC579: Learning-Based Scheduler for Multipath QUIC

Shirley Shu, Tianfang Chang

April.11 2022

Abstract

Multipath transport protocols utilize multiple network paths (e.g., WiFi and LTE) to achieve improved performance and reliability. The scheduler of a multipath transport protocol determines how to distribute the data packets onto different paths. However, multipath schedulers have many challenges to solve. It's hard for a scheduler to keep efficient all the time when dealing with heterogeneous paths with dynamic path characteristics (i.e., packet loss, fluctuation of delay). Thus, it's important to design a learning-based scheduler to adapt to real-time changes in the network, which can significantly enhance the efficiency of transport. Multi-armed bandit (MAB) problem is a classic reinforcement learning problem that exemplifies the exploration-exploitation trade-off dilemma. Thus, we formulate the scheduler as a MAB problem and try to apply Upper Bound Confidence (UCB) algorithm to the optimization. To evaluate the performance of the MAB-based scheduler, we implement the scheduler in the MPQUIC module on the ns-3 platform. The resulting scheduler allows an increase in application goodput and a decrease in complete time as compared with two widely used scheduler algorithms, Round-Robin and Min-RTT.

1 Introduction

Over the last few decades, advances in computer networking have accelerated. Today's end devices have many network interfaces that operate in various modes. For quick and reliable data sharing, multipath transport protocols allow the simultaneous use of various radio access technologies, such as WiFi and cellular. Specifically, the sender distributes application data across multiple available radio interfaces. The receiver reassembles and reorders data from various paths, making it transparent to the application. By doing so, multipath transport protocols aim to improve both transmission capacity and reliability over single-path alternatives.

MPTCP [1] is a multipath extension on top of TCP, supporting transmission with multiple paths in the transport layer. Thanks to its outstanding features such as throughput aggregation and congestion shift, it has been considerably adopted for commercial use. An example is its usage since iOS11. Nevertheless, the next-generation networks pose a set of challenges to the protocols built upon the TCP/IP stack, e.g., connection breakage, and Head-of-Line (HoL) blocking issues. Motivated by the success of MPTCP, multipath extension over QUIC (MPQUIC) [2] is more promising to satisfy the demands of future applications. Even though MPQUIC is still under discussion by Internet Engineering Task Force (IETF), there are already some related works around the protocol implementation [3].

In multipath transport protocols, the scheduler determines how to distribute data across the available paths. The application's data packets are stored in the second buffer, and the scheduler assigns each packet to a separate interface based on the scheduling strategy. One of the most difficult aspects of developing a multipath scheduler is implementing a strategy that addresses the varied features of routes, such as the mix of WiFi and LTE networks. When the pathways are diverse, particularly in terms of latency and loss, sent packets arrive at the destination out of order, resulting in HoL blocking, which reduced the aggregated performance.

Existing research around the scheduling algorithms base on stable network scenarios

or focuses on the specific requirement by application. Blocking Estimation-based MPTCP Scheduler (BLEST) [4] and the Earliest Completion First (ECF) [5] schedulers tackle the issue with heterogeneous paths by introducing a wait action, which allows the scheduler to decide whether or not to send a packet until better conditions arise. These schedulers perform well when channel properties are fairly stable, but they were not designed to handle dynamically changing characteristics. Priority-Based Stream Scheduling (PStream) [6] finds that scheduling without the recognition of the stream features can aggravate inter-stream blocking when sharing paths. It is proposed to decrease the website loading time by scheduling high priority packets first, but not sufficient in other scenarios, like file downloading and video streaming.

However, as illustrated in Fig. 1, in a multipath scenario, the characteristics of the paths may be heterogeneous and they often vary over time, especially in wireless networks(e.g. WiFi or LTE). A multipath scheduler should take the dynamicity of each path into account while determining the scheduling policy [7]. For example, the delay characteristics in a public area of a WiFi network can vary significantly over time due to changing numbers of users creating varying levels of congestion, resulting in different dynamicity levels throughout the day. Such dynamic changes are particularly widespread in today's WiFi and LTE networks and are predicted to become even more prevalent in the next 6G technology. Considering the dynamicity, We propose a state-of-the-art multipath scheduler based on Multi-armed Bandit and found that it consistently outperformed the other typical schedulers.

The contributions of our work can be summarized as follows:

- We introduce a new learning-based multipath packet scheduling. The research problem is fundamental, and it can tackle some MPQUIC performance issues and adapt to the dynamic heterogeneous network environments.

- We first formulate the multipath scheduling problem and propose a lightweight and deployable online learning solution to this problem. More specifically, a deterministic strategy is derived by using an RL algorithm applied in Multi-Armed Bandit(MAB)

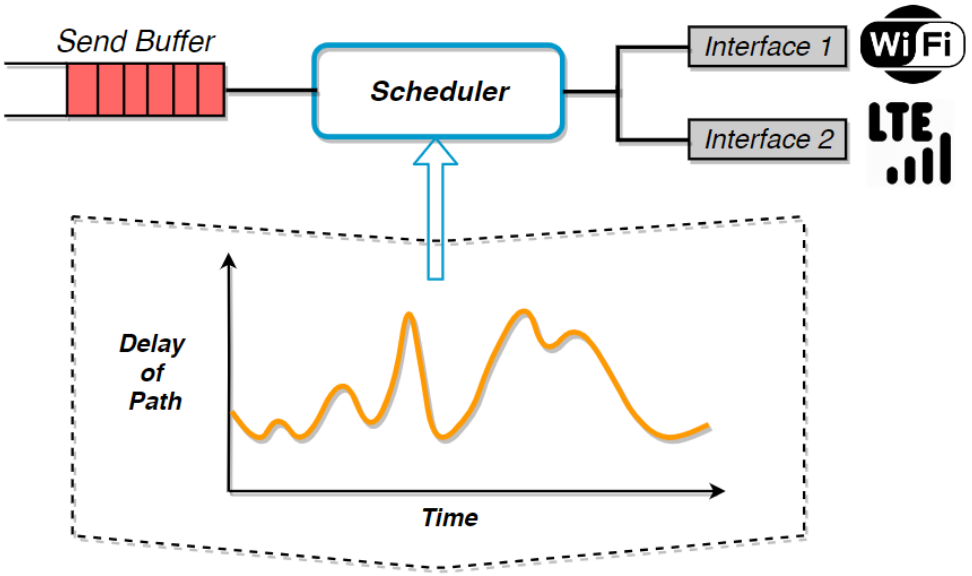


Figure 1: Illustration of MPQUIC scheduler.

scenarios.

- We implement the scheduler in the ns-3 and evaluate it over both emulated and real network conditions. We conduct extensive experiments to evaluate its performance. It shows the improvement of the goodput compared to MinRTT or Round-Robin, and it significantly outperforms the state-of-the-art schedulers on a variety of performance metrics.

The rest of this report is organized as follows. Section 2 summarizes the background and motivation, including the introduction of MPQUIC, existing schedulers, problem verification and the multi-armed bandit. Section 3 presents the system model and the problems to address. It elaborates on the detailed design of the MAB-based scheduler algorithm. To verify the performance gain of our proposal, experiments using NS-3 along with detailed analyses are given in Section 4, followed by concluding remarks and further research issues in Section 5.

2 Background and Motivation

2.1 MPQUIC

The multipath QUIC protocol intends to compensate for the missing features in QUIC by utilizing different paths that exist between a client and a server [2]. The layered structure of MPQUIC is illustrated in Fig. 2.

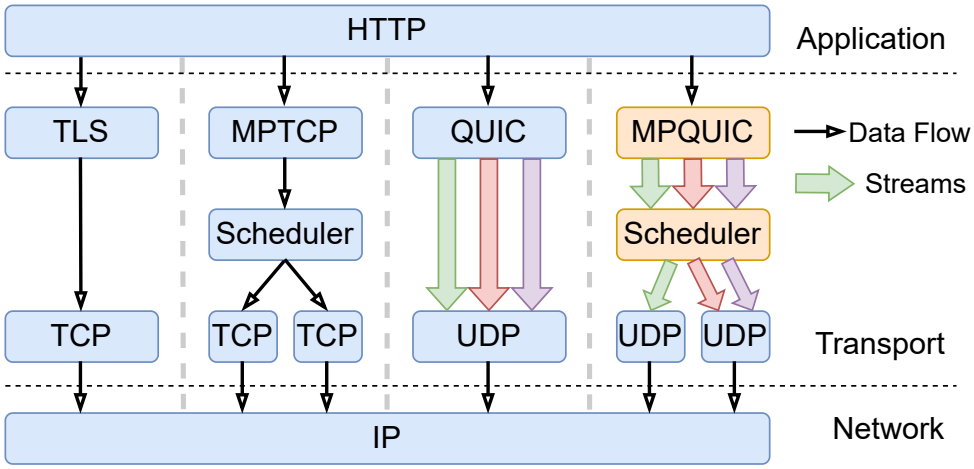


Figure 2: Structure of MPQUIC in comparison with others.

Based on several salient features and improvements in QUIC, the design specifications of MPQUIC [2, 3] are described in the following components.

Path Identification. MPQUIC introduced a path identification in its packet header to determine different paths in use. QUIC uses increasing packet numbers to identify and retransmit the lost packet. However, if all packets share one numbering space in MPQUIC and are sent over different paths, they might arrive out of order, resulting in a misinterpretation of packet loss. To deal with this issue, MPQUIC designed a per-path numbering space, which means packet numbers on different paths are isolated from each other and the sequential feature occurs only within that path.

Path Management. A path manager manages the path creation and removal in MPQUIC. In QUIC packets, the payload is comprised of multiple frames that can store stream data or control information. Benefiting from this frame structure, the extension of

QUIC can define some specific types of frames to store the multipath information. To manage multiple paths, new frames, e.g., `ADD_ADDRESS`, `REMOVE_ADDRESS`, and `PATH_ABANDON`, are introduced to help the path establishment and removal.

Packet Scheduling. The path scheduler in MPQUIC is responsible for allocating packets onto different paths. There have been numerous scheduling policies proposed, each with pros and cons depending on the network scenarios and application requirements. The simplest scheduling algorithm is Round-Robin, which schedules packets on different paths sequentially, but may cause high latency when two flows have a large difference in bandwidth and round-trip time (RTT). So, MPQUIC uses Min-RTT by default, which is implemented in the Linux kernel for MPTCP. Provided that the congestion window still has space, the path with the lowest measured round-trip time (RTT) is preferred.

Multipath transport protocols utilize multiple network paths (e.g., WiFi and LTE) to achieve improved performance and reliability, compared with their single-path counterparts. The scheduler of a multipath transport protocol determines how to distribute the data packets onto different paths. However, multipath schedulers have many challenges to solve. It's hard for a scheduler to keep efficient all the time when dealing with heterogeneous paths with dynamic path characteristics (i.e., packet loss, fluctuation of delay). Thus, it's important to design a learning-based scheduler to adapt to real-time changes in the network, which can significantly enhance the efficiency of transport.

2.2 Scheduler

Research around the scheduling algorithms are getting popular in recent year. Here we introduce some typical works for packet scheduling algorithms in transport-layer protocols.

Default Algorithms. The basic scheduling algorithm in multipath transport-layer protocols are Round-Robin, which arranges packets on each paths one-by-one. This idea is easy to implement but causes large gaps when two paths have distinct data rates or delays. Min-RTT is introduced to deal with this problem, which always select the path that have

lowest round-trip-time (RTT). However, this algorithm will ignore the slow paths, which may not fully uses all essential paths. Blest [4] aims to deal with the problem of head-of-line blocking when multiple paths have heterogeneous data rates and delay. The main idea of this algorithm is to try to make the packets arriving in sequence with a prediction of the number of packets during RTT. The experiment shows it performs well with bulk send, but is not that good with websites or videos.

Improved Algorithms. Based on the idea of Blest, ECF [5] not only utilize the information of RTT but also uses other relevant information of paths, like the congestion window, to improve the scheduling algorithm. [8, 9] designed the stream-aware scheduling with the features of MPQUIC, refering the idea of ECF over MPTCP. With the popularity of MPQUIC, the scheduling algorithms need to consider not only arranging packets over different paths but also take the priority streams into account. Pstream [10, 6] finds that scheduling without the recognition of the stream features can aggravate inter-stream blocking when sharing paths. Thus, it proposes Priority-Based Stream Scheduling which has a global scheduler for allocating streams to paths and the stream managers for each path. It uses a job shop scheduling algorithm to distribute streams' data to heterogeneity paths according to their priorities and sizes so that it can decrease the overall transmission time. It also introduces a bandwidth sharing mechanism for streams on the same path.

Learning-based Algorithms. Since the network in real world is always dynamic and unsatable, some researchers consider to apply online learning algorithms with the scheduling strategy. [11] introduced an approach to apply deep reinforcement learning on MPQUIC scheduler. It uses Q-learning to train the data collected by running MPQUIC srtt scheduler, then apply the DNN model (trained by Q-learning) to the new scheduler. This paper shared a way of utilizing some existing libraries to connect with the MPQUIC scheduler, but the performance is not that significant with higher delay and background traffic. In addition, Peekaboo [12] uses online learning to improve the scheduler. The speed at which the network changes can surpass the learning speed achieved through online learning. Few data, like

mobility, may not help finish training and get a good schedule policy. [13] extends Peekaboo in 5G scenarios which has more homogeneous contexts, like average RTT, throughput, and loss rate. It shows that learning based scheduler (Peekaboo) performs a bit worse in a dynamic scenario as it needs frequent retraining. In the worst case, when the retraining is completed, the environment has already changed again. Thus, lifelong learning scheduler (or learning-based scheduler with a small training set) could be considered as our further research.

2.3 Problem Verification

To illustrate the unresolved performance issues of packet scheduling in MPQUIC, we conduct experiments on existing schedulers under different network scenarios. Particularly, we tested two widely used scheduler algorithms, Round-Robin and Min-RTT, with 10 rounds for each transmission.

The experiments were conducted with the topology shown in Fig.3 that two subflows each with TCP background traffic respectively. We first consider two subflows that have the same data rate and delay. The data rate for both subflows is 10 Mbps with a 10 ms delay. Fig. 4a illustrates the throughput comparison for Round-Robin (left-side) vs. Min-RTT (right-side). For Round-Robin, the throughput of both subflows is around 5 Mbps, which is close to the ideal data rate. For Min-RTT, the throughput of subflow 0 is close to the ideal data rate, but the throughput of subflow 1 is around 2.5 Mbps, which is not fully used.

To encounter the transmission close to the real world, our second experiment scenario simulates the transmission similar to mobile devices, which contains both Wi-Fi and LTE network and is able to do packets transmission with both interfaces at the same time. From our daily experience, we know that mostly Wi-Fi has a higher data rate than LTE. The data rate for subflow 0 is 10 Mbps with a 10 ms delay, which is used to simulate the LTE connection. The data rate for subflow 1 is 50 Mbps with a 10 ms delay, which is used to simulate the Wi-Fi connection. Fig. 4b illustrates the throughput comparison for Round-

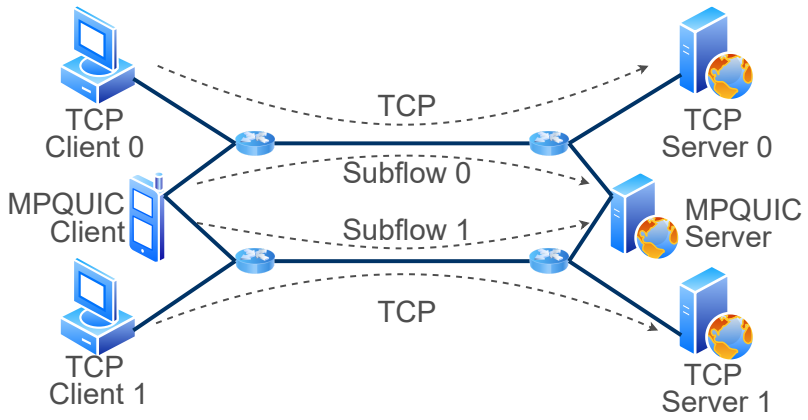


Figure 3: Topology of two subflows each with TCP background traffic respectively

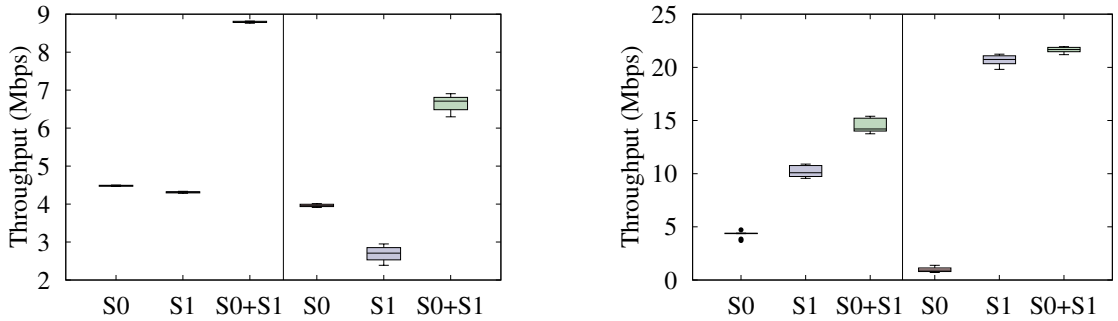


Figure 4: Throughput comparison for Round-Robin vs. Min-RTT with (a) the same data rate of subflows (S0: 10 Mbps, S1: 10 Mbps) and (b) the different data rates of subflows (S0: 10 Mbps, S1: 50 Mbps).

Robin (left-side) vs. Min-RTT (right-side). For Round-Robin, the throughput of subflow 0 is around 5 Mbps, which is close to the ideal data rate, but subflow 1 is not fully used. For Min-RTT, the throughput of subflow 1 is around 20 Mbps, which is close to the ideal data rate, but subflow 0 is not fully used.

From the experiment, Round-Robin has a better performance with a similar data rate while Min-RTT outperforms with a distinct data rate. Thus, we verified the problem that the default schedulers have limitations that cannot always show good performance with different network scenarios.

2.4 Bandit Algorithm Analysis

The most important feature distinguishing reinforcement learning from other types of learning is that it uses training information that evaluates the actions taken rather than instructs by giving correct actions. This is what creates the need for active exploration, for an explicit search for good behaviour.

The multi-armed bandit problem is a classic problem that well demonstrates the exploration vs exploitation dilemma. Imagine you are in a casino facing multiple slot machines and each is configured with an unknown probability of how likely you can get a reward at one play.

Exploration will give up some known reward information and try some new options - that is, in a certain state, the algorithm may have learned what action to choose to make the reward larger, but it cannot be done every time. The same choice, maybe another choice that has not been tried will have a greater reward, that is, Exploration hopes to explore more potential information. Exploitation refers to maximizing rewards based on known information.

The difference can also be simply understood that the Exploration algorithm searches for the global optimal solution and is not based on the existing experience; the Exploitation algorithm searches for the local optimal solution and maximizes the use of the existing experience information.

- We have k-armed bandit with winning probabilities, $p_1, ..., p_k$.
- At each time step t, we take an action(pull a arm) on the machine and receive a reward r_t.
- A is a set of actions, each referring to the interaction with one arm. The value of action a is the expected reward, $Q(a) = \mathbb{E}[r|a] = p$. If action a_t at the time step is on the k_i arm, then $Q(a_t) = p_i$.

- R is a reward function. In the case of a k-armed bandit, we observe a reward r in a stochastic fashion. At the time step t, $r_t = R(a_t)$ may return reward 1 with a probability $Q(a_t)$ or 0.

Let $Q_t(a)$ denotes the estimated value of action $a(k_i$ arm) at time step t.

If $Q_t(a)$ is an accurate estimate of $q_*(a)$, we can select actions using $Q_t(a)$. Greedy action estimate of $q_*(a)$: $A_t \doteq argmax Q_t(a)$

Sample average estimate of $q_*(a)$:

$$Q_t(a) \doteq \frac{sum\ of\ rewards\ by\ choosing\ a\ prior\ to\ t}{number\ of\ times\ of\ choosing\ a\ prior\ to\ t}$$

If a is chosen more often, the estimate becomes more accurate. Exploratory action selection to improve value estimation.

Greedy action selection exploits current knowledge of action value. Exploitation maximizes the expected reward for the current step.

Non-greedy action selection explores other actions and improves their value estimates. Exploration may improve total reward in the long run.

Constant memory requirement and constant per-time-step computation:

$$NewEstimate = OldEstimate + StepSize[NewObervation - OldEstimate]$$

Sample average estimate:

$$Q_t(a) = Q_{t-1}(a) + \frac{1}{t}[R_t(a) - Q_{t-1}(a)]$$

$R_t(a)$ is the reward after selecting action a at time step t.

To solve the non-stationary problem, the updated samples will be significant, so we can

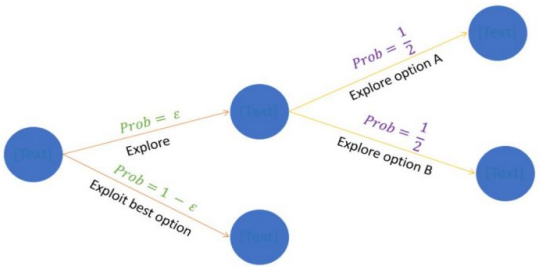


Figure 5: $\epsilon-$greedy

use a constant discount factor alpha, we can rewrite the update equation like this:

$$Q_t(a) = Q_{t-1}(a) + \frac{1}{\alpha}[R_t(a) - Q_{t-1}(a)]$$

Note that we have replaced t with a constant alpha, which ensures that the most recent samples have a higher weight, and these most recent samples more determine the delta.

Using the epsilon probability, we will choose a random action (exploration) and choose the action with the largest $Q_t(a)$ with probability $1 - \epsilon$.

With probability $1 - \epsilon$, we choose the action with the maximum value ($argmax\ Q_t(a)$)

With probability ϵ, we randomly choose an action A a set of all actions.

For example, if we have a problem with two actions A and B, then the epsilon greedy algorithm works as follows:

The UCB algorithm keeps a track of the mean reward for each arm up to the present trial and also calculates the upper confidence bound for each arm. The upper bound indicates the uncertainty in our evaluation of the potential of the arm.

The algorithm is highly unsure of an arm's potential if it has a very high upper confidence bound and hence chooses the arm because of a great exploration opportunity.

$$A_t \doteq \arg\max_a [Q_t(a) + c\sqrt{\frac{\log t}{N_t(a)}}]$$

Exploitation:

- $Q_t(a)$ represents the exploitation part of the equation. UCB is based on the principle of "optimism in the fact of uncertainty", which basically means if you don't know which action is best then choose the one that currently looks to be the best. Taking this half of the equation by itself will do exactly that: the action that currently has the highest estimated reward will be the chosen action.

Exploration:

- The second half of the equation adds exploration, with the degree of exploration being controlled by the hyper-parameter c. Effectively this part of the equation provides a measure of the uncertainty for the action's reward estimate.

- If an action has not been tried very often, or not at all, then $N_t(a)$ will be small. Consequently, the uncertainty term will be large, making this action more likely to be selected. Every time an action is taken we become more confident about its estimate. In this case $N_t(a)$ increments, and so the uncertainty term decreases, making it less likely that this action will be selected as a result of exploration (although it may still be selected as the action with the highest value, due to the exploitation term).

- When an action is not being selected, the uncertainty term will grow slowly, due to the log function in the numerator. Whereas, every time that the action is selected, the uncertainty will shrink rapidly due to the increase in $N_t(a)$ being linear. So the exploration term will be larger for actions that have been selected infrequently, due to the uncertainty in the estimates of their rewards.

- As time progresses the exploration term gradually decreases until eventually actions are selected based only on the exploitation term.

3 Problem Description and Solution Framework

Based on the discussions in previous sections, we present here in detail the design and implementation of the MAB-based scheduler. We first describe the learning aspects, i.e., the online learning of a deterministic scheduling decision. Then, we also depict how these aspects are combined and deployed in the MPQUIC.

3.1 Problem Formulation

The traditional scheduler (e.g. Min-RTT) is similar to the greedy algorithm. It may fall into local optimality when it finds an efficient path in the past. However, such an efficient path may not always adapt to the dynamic network at a later time. Thus, we consider using MAB (Multi-armed bandits) as the core of learning. MAB is a simple but very powerful framework for algorithms that make decisions over time under uncertainty. It's a good way to face the trade-off between "exploitation" and "exploration" when we tried to find an efficient schedule algorithm. Specifically, we formulate the problem with Upper Confidence Bound (UCB). Let $(X_t)_{t-1}^n$ be a sequence of independent 1-subgaussian random variables with mean μ and $\hat{\mu} = \frac{1}{n}\sum_{t=1}^{n} X_t$, for all $\delta \in (0, 1)$,

$$P\left(\mu > \hat{\mu} + \sqrt{\frac{2\log(1/\delta)}{T_i(t-1)}}\right) \leq \delta \tag{1}$$

A typical MAB model consists of agent, arm, state, action, reward, and regret, whose definitions in our framework are explained in the following.

Agent: an agent is an entity in the system that performs a learning task. In the MPQUIC packet scheduling problem, the agent is responsible for determining the traffic distribution over multiple subflows on the sender side of an MPQUIC connection.

Arm: in the MPQUIC packet scheduling problem, we consider each subflow is an arm i, where $(1 \leq i \leq k)$. We have k-subflows in total.

State: a state of the system is the information of a snap shot of the environment that can

be observed by an agent. At the beginning of each time step, the agent observes the system state. Assume in the t-th time step, the system state is represented by $s_t = (s_{t,1}, s_{t,2}, ..., s_{t,k})$, where $s_{t,i}$ is the observed state of the i-th subflow which can be represented by a tuple $s_{t,i} = (x_{t,i}, d_{t,i}, u_{t,i})$, where

- $x_{t,i}$ is the subflow throughput measurement in time step t;
- $d_{t,i}$ is the subflow mean RTT in time step t;
- $u_{t,i}$ is the number of lost packets in time step t;

Action: an action indicates how an agent response on the observed state. In MPQUIC packet scheduling, an action is a scheduling decision, which determines how to distribute the current traffic over multiple paths. An action can be represented by a vector $a_t = (p_1, p_2, ..., p_k)$, where $p_i = 1$ means the packets will be delivered over the i-th subflow. Otherwise, $p_i = 0$. The action set is represented by

$$A_t = \arg\max_i \hat{\mu}_i(t-1) + \sqrt{\frac{2\log f(t)}{T_i(t-1)}} \tag{2}$$

where $\hat{\mu}_i$ is the expected value of rewards with i-th subflow, and T_i is number of samples choosing subflow i in t-th round. $f(t) = 1 + t\log^2(t)$ is the error probability. $\sqrt{\frac{2\log f(t)}{T_i(t-1)}}$ works for the exploration. During the previous rounds, if the use of subflow i smaller, $\sqrt{\frac{2\log f(t)}{T_i(t-1)}}$ larger. So, it is more possible to choose i if $\hat{\mu}_i$ is same.

Reward: In the t-th time step, the agent observes the state s_t, and takes an action (pull an arm) a_t on the machine and observe a reward X_t. After applying the action, the state of the environment transitions to $s, t+1$ and the agent receives a reward by taking the action. A reward function is used to evaluate the long-term performance of MPQUIC, which could be represented as

$$X_{t,i} = \frac{MSS}{d_{t,i}} * \frac{1}{\sqrt{u_{t,i}}} \tag{3}$$

where MSS is maximum segment size and $u_{t,i}$ is number of lost packets in t-th round.

Regret: a regret indicates the performance of the MAB algorithm. In MPQUIC packet scheduling, our objective is to maximize the total rewards. So, we measure the performance with Regret

$$R_n = n\mu^* - \mathbb{E}[\sum_{t=1}^{n} X_t] \tag{4}$$

where $\mu^* = \max_i \mu_i$, μ_i is the mean reward of P_i.

Given the above preliminaries, the MPQUIC packet scheduling problem can be represented by a MAB problem: learning an optimal policy to maximize the expected cumulative reward in the episode.

3.2 Current Algorithm

The overall MAB-based scheduler algorithm is shown in Algorithm 1. We first need the number of subflows as the input of the algorithm. During the transmission process, the scheduler will choose each subflow once to get the initial reward for each subflow. Then, for each round of distributing packets by the scheduler, it observes the new reward for each subflow calculated by the state. The expected value of total rewards is updated after observing the new states. With the value of rewards on each subflow, the scheduler chooses the action that maximizes the rewards. It then processing the transmission based on the action.

Algorithm 1 MPQUIC Scheduler with UCB

Require: k

Choose each action once;
for $t \in 1, 2, ..., n$ **do**
    for $i \in 1, 2, ..., k$ **do**
        Observe reward $X_i(t-1) = \frac{MSS}{d_{t,i}} * \frac{1}{\sqrt{u_{t,i}}}$;
        Update $\hat{\mu}_i(t-1)$ = expected value of X_i in previous $t-1$ rounds;
    end for
    Choose action $A_t = \arg\max_i \hat{\mu}_i(t-1) + \sqrt{\frac{2 \log f(t)}{T_i(t-1)}}$;
    Process action in scheduler;
end for

4 Performance Evaluation

4.1 Experiment Setup

Recall that our design will be based on the up-to-date multipath transport-layer protocol, MPQUIC. Since the experiments on transport-layer protocols are difficult to realize with the hardware in the real world, a network simulator is a valuable choice with its various modules, high precision, and low cost. Exploring the popular simulators, we selected ns-3, a discrete-event network simulator for Internet systems, targeted primarily for research and educational use. Benefiting from the open-source contribution, we can use the ns-3 QUIC module [14] with some multipath extension to implement our design. The multipath extension of the QUIC module is a previous research work, which is able to have the basic data transmission over two paths.

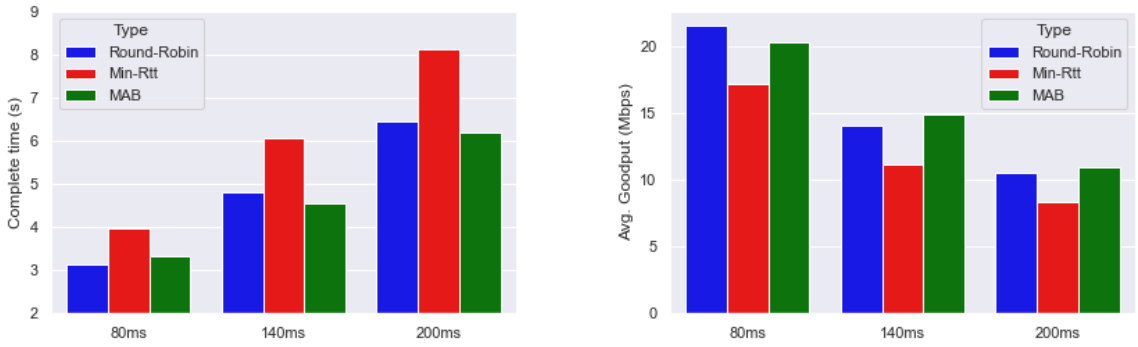
We implement the MAB-based scheduler in MPQUIC in ns-3 according to our solution framework. To verify the performance of our MAB scheduler, we experiment with it in both heterogeneous and homogeneous network environments to compare its performance with other multipath schedulers.

4.2 Performance Analysis

We test for transmitting a file with 8 MB, which is using the bulk sending application. In the first scenario, we consider two-path have the same data rate and delay which is 25 Mbps with 80ms delay, 140ms delay and 200 ms delay. Figure 6a show the complete time. From the diagram, the blue one is Round-Robin, the red one is Min-RTT, and the green one is our MAB scheduler. The round-robin performs better than min-RTT with the same data rate. We can see MAB performance is good compared to both min-RTT and round-robin. The goodput shown in figure 6b also present that mab has a better performance in this scenario.

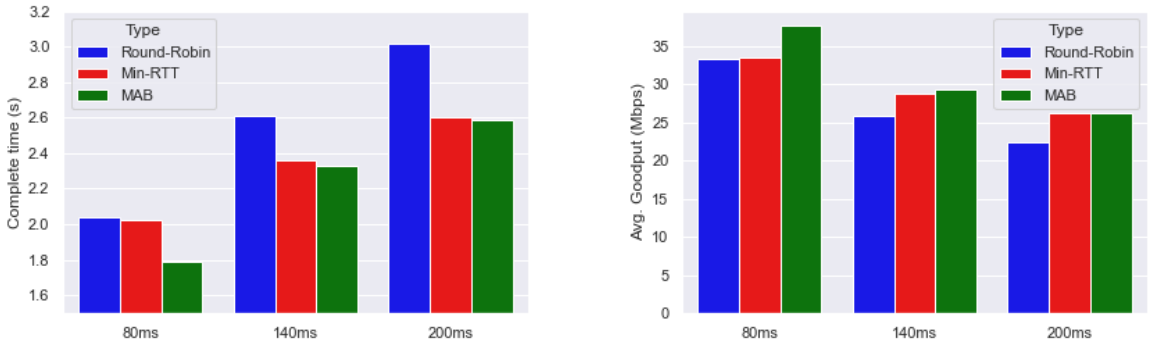
For the second scenario, we consider the transmission with different data rates and delays. For subflow 0, we fix the data rate and delay with 50 Mbps and 20 ms. and for subflow 1,

we test with 10 Mbps 80ms delay, 10 Mbps 140ms, and 10Mbps, 200ms delay. From figure 7, we can see min-RTT perform better than round-robin, and our mab performs better in both complete time and goodput.



(a) Complete Time (b) Average Goodput

Figure 6: s0 and s1: 25Mbps/80ms to 25Mbps/200ms



(a) Complete Time (b) Average Goodput

Figure 7: s0: 50Mbps/20ms, s1: 10Mbps/80ms to 10Mbps/200ms

From the experiment in these two scenarios, we can see that round-robin or min-RTT can have good performance in one of them, which means Round-Robin does better with the same data rate and min-RTT do better with different data rate. They cannot maintain better performance in different scenarios, while our mab can do better in both scenarios, which presents the adaptation of our scheduler.

5 Conclusion

The multipath scheduler is a fundamental mechanism that has a significant impact on the performance of MPQUIC. To cope with the challenges of network heterogeneities, we formulate the scheduling problem with Multi-armed Bandit. we propose an adaptive multipath scheduler that leverages an online learning mechanism. It consists of three key designs: (1) A multi-armed bandit to generate the scheduling actions; (2) A comprehensive reward function to consider the network throughput for performance optimization; and (3) an online learning algorithm to enable adaptive packet scheduling. It is computationally lightweight and easily deployable. We implement the MAB-based scheduler in MPQUIC in ns-3 and compare its performance with other default schedulers over both similar and distinct network scenarios, which show the adaptive of the MAB-based scheduler outperforms the widely used schedulers.

Due to the time limitation of a course project, we will strengthen our experiment in future work. We first consider comparing with more state-of-art schedulers like BLEST, ECF, or Peekaboo. Since our current scenario is stable, it may not present the performance of the adaptive scheduler well. We will consider testing in more dynamic scenarios close to the real world. On the other hand, we only consider the throughput as our rewards, while other characteristics are related to the transmission performance. We will strengthen the definition of rewards like congestion windows and error control. Another improvement of our work could focus on the bandit algorithm. We currently just use the UCB, which predicted that the rewards are getting immediately after the auction. However, in the network transmission, the information we got is always delayed. So, the delayed rewards in MAB might match the scheduler algorithm better.

References

[1] C. Raiciu, S. Barre, C. Pluntke, A. Greenhalgh, D. Wischik, and M. Handley, "Improving datacenter performance and robustness with multipath tcp," in *Proceedings of the ACM SIGCOMM 2011 Conference*, ser. SIGCOMM '11, 2011, p. 266–277.

[2] Q. De Coninck and O. Bonaventure, "Multipath quic: Design and evaluation," in *Proceedings of the 13th International Conference on Emerging Networking EXperiments and Technologies*, ser. CoNEXT '17, 2017, p. 160–166.

[3] T. Viernickel, A. Froemmgen, A. Rizk, B. Koldehofe, and R. Steinmetz, "Multipath quic: A deployable multipath transport protocol," in *Proceedings of the IEEE International Conference on Communications (ICC)*, 2018, pp. 1–7.

[4] S. Ferlin, Ö. Alay, O. Mehani, and R. Boreli, "Blest: Blocking estimation-based mptcp scheduler for heterogeneous networks," in *2016 IFIP Networking Conference (IFIP Networking) and Workshops*. IEEE, 2016, pp. 431–439.

[5] Y.-s. Lim, E. M. Nahum, D. Towsley, and R. J. Gibbens, "Ecf: An mptcp path scheduler to manage heterogeneous paths," in *Proceedings of the 13th International Conference on Emerging Networking EXperiments and Technologies*, ser. CoNEXT '17. New York, NY, USA: Association for Computing Machinery, 2017, p. 147–159. [Online]. Available: https://doi.org/10.1145/3143361.3143376

[6] X. Shi, L. Wang, F. Zhang, B. Zhou, and Z. Liu, "Pstream: Priority-based stream scheduling for heterogeneous paths in multipath-quic," in *2020 29th International Conference on Computer Communications and Networks (ICCCN)*. IEEE, 2020, pp. 1–8.

[7] S. Deng, R. Netravali, A. Sivaraman, and H. Balakrishnan, "Wifi, lte, or both? measuring multi-homed wireless internet performance," in *Proceedings of the 2014 Conference on Internet Measurement Conference*, ser. IMC '14. New York, NY,

USA: Association for Computing Machinery, 2014, p. 181–194. [Online]. Available: https://doi.org/10.1145/2663716.2663727

[8] B. Jonglez, M. Heusse, and B. Gaujal, "Srpt-ecf: challenging round-robin for stream-aware multipath scheduling," in *2020 IFIP Networking Conference (Networking)*, 2020, pp. 719–724.

[9] A. Rabitsch, P. Hurtig, and A. Brunstrom, "A stream-aware multipath quic scheduler for heterogeneous paths," in *Proceedings of the Workshop on the Evolution, Performance, and Interoperability of QUIC*, ser. EPIQ'18. New York, NY, USA: Association for Computing Machinery, 2018, p. 29–35. [Online]. Available: https://doi.org/10.1145/3284850.3284855

[10] X. Shi, F. Zhang, and Z. Liu, "Prioritybucket: A multipath-quic scheduler on accelerating first rendering time in page loading," in *Proceedings of the Eleventh ACM International Conference on Future Energy Systems*, ser. e-Energy '20. New York, NY, USA: Association for Computing Machinery, 2020, p. 572–577. [Online]. Available: https://doi.org/10.1145/3396851.3402923

[11] M. M. Roselló, "Multi-path scheduling with deep reinforcement learning," in *2019 European Conference on Networks and Communications (EuCNC)*, 2019, pp. 400–405.

[12] H. Wu, Ö. Alay, A. Brunstrom, S. Ferlin, and G. Caso, "Peekaboo: Learning-based multipath scheduling for dynamic heterogeneous environments," *IEEE Journal on Selected Areas in Communications*, vol. 38, no. 10, pp. 2295–2310, 2020.

[13] H. Wu, G. Caso, S. Ferlin, Alay, and A. Brunstrom, "Multipath scheduling for 5g networks: Evaluation and outlook," *IEEE Communications Magazine*, vol. 59, no. 4, pp. 44–50, 2021.

[14] A. De Biasio, F. Chiariotti, M. Polese, A. Zanella, and M. Zorzi, "A quic implementation for ns-3," in *Proceedings of the 2019 Workshop on Ns-3*, ser. WNS3 2019, 2019, p. 1–8.