Meta-Reinforcement Learning based Schedulers for Multipath-QUIC to improve QoE in simulated Mobile Networks

submitted in partial fulfillment for the degree of master of science

Pijus Krisiukėnas
14456990

Master Information Studies
data science
Faculty of Science
University of Amsterdam

Submitted on 29.06.2023

	UvA Supervisor
Title, Name	Hongyun Liu
Affiliation	UvA Supervisor
Email	h.liu@uva.nl

ABSTRACT

Multipath-QUIC has been recently investigated for possible improvements in network bandwidth. The performance of Multipath-QUIC is network packet scheduler dependent, thus research has been done into developing reinforcement learning based schedulers, with recent work showing promise in the use of meta-learning to further improve the scheduler in mobile networks. In this paper, an LSTM based meta-learning model is used, with the goal of improving quality of experience in a video streaming application. To improve the validity of the evaluation, a framework is created that uses real world data to simulate typical network patterns the scheduler will experience. The results show that the use of meta-learning provides a 11% quality of experience improvement over a simple reinforcement learning model, specifically when applied in mobile scenarios. The proposed model does not outperform the minRTT baseline. Further research in improving the core model by the use of an online-offline approach is suggested.

KEYWORDS

multipath, quic, reinforcement learning, Quality of Experience, meta learning, video streaming

GITHUB REPOSITORY

https://github.com/pijuskri/MPQUIC-meta-learning

1 INTRODUCTION

Online streaming has become a core service provided using the internet. For users of streaming sites, quality of experience(QoE) is a more important factor than the pure throughput of a connection. Thus, stream scheduling is used to improve streaming performance and QoE. To optimise stream scheduling a new approach has been used: the transport protocol [35]. QUIC is a proposed high-performance transport protocol, which lies on the application layer and builds on top of UDP [16]. QUIC reduces latency in connection instantiation and incorporates security by design. Multipath-QUIC (MPQUIC) [28] provides support for devices with multiple interface links to boost performance gains. This is particularly useful for mobile phones, which may connect to the internet with a local WiFi and a cellular connection simultaneously. However, this protocol relies on a scheduler which makes decisions on which path to use for routing internet traffic. Given multiple paths of highly differing performance indicators, the use of only the fast path can provide higher throughput than using all available paths [17]. Thus, a good design of the scheduler is required to achieve good performance and achieve improvements for QoE, while a suboptimal scheduler can result in reducing the quality of the connection.

Therefore, multiple schedulers have been proposed over the history of MPQUIC. Algorithm and heuristic based models were developed initially, due to low computational overhead and simple implementation, like the default scheduler for MPQUIC "minRTT" [4]. Further research proposed the use of machine learning models to improve performance beyond algorithmic approaches, with all solutions focusing on reinforcement learning(RL) models [12]. This specific machine learning paradigm is chosen for its capabilities to perform "online" training and prediction, where no previous data or training is required to deploy the model. The model learns from the environment by taking actions which interact with the environment and then getting the reward for previous actions taken. After some time, the model can learn the correct actions to take given a state. However, due to this online-only learning approach, the model tends to perform poorly when the environment changes [29]. To remedy this, meta-learning can be used, which allows RL models to better generalize across different environments by training on previous experiences [9].

In the field of MPQUIC scheduling, meta-learning has been used before in a scheduler called "FALCON" [31]. This scheduler performed both online and offline learning, where a group of different models was created. While a model is being used online the most representative previously learned model is selected to perform meta-learning and help the initial training. The solution presented is a valid meta-learning tactic but it is a specific solution proposed only in the paper. There is no exploration of what tactic or meta-learning framework would work best. There are also no comparisons to other solutions using meta-learning, as this paper is the only one employing meta-learning for an MPQUIC scheduler.

Evaluation of the developed models is commonly performed using pure bandwidth as the metric, but given a video streaming task, quality of experience is the preferred alternative [27]. For MPQUIC, heuristic and QoE-oriented schedulers have been developed [35] and evaluated in terms of QOE, while RL-based approaches for video streaming have not performed QoE evaluation [17].

Further, testing in a mobile setting is important to accurately assess the performance of a scheduler, given the main application of MPQUIC is in mobile networks. In MPQUIC scheduler literature most evaluation is performed in static scenarios only, with the available mobile evaluations only being performed in real-world scenarios, complicating reproducibility.

To summarize, the research gap is present in both the underlying scheduler model and its evaluation. Firstly, there is a lack of research into meta-learning for use in schedulers. Second, there is a lack of QoE-oriented scheduler evaluation, especially in mobile environments.

Therefore, to further the performance of MPQUIC schedulers in video streaming and mobile networks, this paper defines the following research question: *Can applying meta-learning to a deep learning model be used to improve an MPQUIC scheduler's QoE performance in a mobile network?*. Sub questions for the research question are outlined below:

- How applicable is meta-learning for MPQUIC scheduling and what are the challenges?
- Does meta-learning provide a benefit to the scheduler?
- How does mobility impact performance?

2 RELATED WORK

2.1 State of the art

In the field of MPQUIC scheduling, a clear consensus on the state of the art(SOTA) has not been established. There is a lack of comparisons between the best available models, so determining SOTA has to be done using improvement over a baseline. There are also a lot of variations of different models and testing criteria, making comparisons harder. Due to this state of the field, this paper will limit the scope of comparisons with other machine learning based models, especially those that performed evaluation on mobile scenarios.

The best performing machine learning based models typically achieve 20% to 25% improvement in download time over a minRTT baseline in static scenarios [17] [31]. This improvement is similar between the different proposed models. In terms of mobile performance, the observed improvement is much smaller for simple models, with online Deep Q-learning achieving a 7% worse download time when compared to minRTT. The best available model performance improvements observed were 20% [29] and 30% [31] when looking at download time, falling in line with the improvements seen in static scenarios. The best performing model (FALCON with 30% improvement over baseline) made use of meta-learning for its design [31].

The only paper assessing the performance of their algorithm in terms of QoE [35], did not compare their proposed model to multipath minRTT, thus there are no papers available to compare QoE metrics with. This also means the state of the art in terms of QoE is also not known.

2.2 Meta-learning

Meta-learning for use in stream scheduling is not an explored field, with a single known representative study into its uses for MPQUIC [31]. This representative paper proposes a scheduler called "FALCON", which is an on- and off-line learning based reinforcement learning model. It is based on a meta learning technique of creating a batch of previously trained models, called meta-models. These are later bootstrapped for a specific network condition when a network change is detected, then performing actions and receiving feedback, which is used to define other meta-models and train the existing meta-models with collected experience. The paper tests its implementation with multiple schedulers, some being baseline heuristic based algorithms like minRTT [4], while others are deep-q learning based models [32]. They are tested in both static and mobile scenarios, which is important to check the relevance of using meta-learning. It shows performance improvements in static and mobile test scenarios. Therefore, this would be the primary model to compare with for usage of other meta-learning frameworks.

The paper did not perform an analysis in the field of meta-learning. The framework proposed does not attempt to work with existing meta-learning solutions, but instead creates a new approach to the problem. There is also a lack of analysis into the environment the scheduler operates in and how that restricts the possible designs of a meta-learning framework. By analysing other available approaches, a new method can be found that also overcomes the limits of traditional models when performing in mobile environments.

2.3 Mobile evaluation

Mobile networks are the primary use case of MPQUIC and as such have garnered research into evaluating various configurations of mobility and different use cases, in particular 5G. Most conducted research into mobility has been conducted using real-world experiments, but this approach requires appropriate hardware and does not allow for exact environment reproduction. To remedy this, simulated networks can be used. One of the most comprehensive papers in MPQUIC mobility testing proposing a mobile testing framework [29], using a fork of mini-net that supports wireless networks [7]. The framework uses multiple random walking patterns in a 2d space that can emulate real-life scenarios of signal strengths and cause network switching.

A different approach based on data has been used to evaluate the performance of adaptive bitrate algorithms for video streaming [10]. These involve the use of real-world measurements to set network limits of simulated connections each second. Any data that provides key performance indicator (KPI) measurements can be used, with a common dataset used also describing different mobility settings [23]. When compared to full simulation, a data-oriented approach allows for consistent results given the same data and simpler implementation. A data-oriented evaluation has not been performed in the MPQUIC field.

2.4 Quality of experience

Optimising QoE for video streaming has been focused on in other papers, but most research was conducted with the goal of developing a bit-rate algorithm for DASH [34]. MPQUIC extensions and schedulers have been proposed as an attempt to improve QoE specifically, with the only example being XLINK [35]. It used ack packets to communicate important QoE metrics from a client to the server, which was then used to adjust a scheduler's decisions. QoE is built up of multiple component parts [13], with papers choosing to evaluate different parts of a user's experience in an attempt to estimate their subjective viewing experience and empirically quantify it [34]. RL has been used specifically for video streaming [17], but using throughput as the model reward and evaluation metric. Thus no research has looked into the use of QoE as the reward or performed evaluation with it for RL models.

3 PROBLEM FORMULATION

The use of meta-learning in MPQUIC scheduling has not been previously thoroughly investigated, as established in section 2. Thus, to form a basis for the presented solution in the methodology, the problem will be formulated using existing research in the field of meta-learning. The background is presented first, with the analysis and proposed theoretical approach following after.

Meta-learning is an extensive field that provides a multitude of general and specific techniques to apply to many models and problems. It can be defined as "learning to learn", where a model performs higher-order training on multiple tasks, which can then be used to quickly adapt to a specific task. A task can be defined as a different environment that the model has to interact with over the course of training and testing. So the goal of a meta-learning model can be formulated as improving adaptivity and increasing performance over a regular approach. Adaptivity is an important quality for constantly changing environments, like mobile networks.

Meta-reinforcement learning is the application of meta-learning in the field of RL, which uses specific approaches that can cope with the lack of target labels. There are many types of models used, depending on the number of tasks and available training data. Due to how new the field is and the large number of variations, there is no consensus on what is considered state of the art in the field. The performance of a specific technique is also dependent on the RL environment used, with the one being investigated in this paper being rarely studied [3].

Specifically, the environment a scheduler participates in has constant states and actions, but does not have clear task boundaries. Also, due to the nature of constant and online deployment of a scheduler, the concept of an episode becomes blurred. Many meta-learning techniques, like the commonly used Model Agnostic Meta-Learning (MAML) [6], presume strict environment control and hard task, episode boundaries. Black-box meta-learning approaches, like *deep meta-reinforcement learning* [30], are the least environment specific meta-learning paradigms available.

One of the first proposed meta-RL based models outlined a framework titled as *deep meta-reinforcement learning*. It is built on a base of an actor-critic network and a long-short term memory(LSTM) layer, a variant of a recurrent neural network [30]. LSTM contains an attention mechanism that allows for the model to keep short-term (hidden state) and long-term (cell state) track of previously learned parameters. Short-term memory is used to keep track of previous states when considering the next best action, however when a task changes drastically these short-term dependencies can lose their use. By clearing the hidden state after a task change, but not cell state, local dependencies are reset while keeping long-term knowledge. This aproach is very versatile, but does still rely on knowing when a task change happens.

A way to deal with soft task boundaries is to detect these boundaries during runtime. One of these approaches was used for continual learning to allow for learning multiple tasks without requiring a hard definition of the types of tasks [8]. This approach allows for compromising an online environment with some styles of meta-learning that rely on task boundaries.

By combining task boundary detection and an LSTM layer on top of a regular RL model, meta-learning can be applied to an online MPQUIC scheduler. This approach is investigated in the rest of the paper.

4 METHODOLOGY

The scheduler and testing framework details are presented below. To form a basis for the testing framework and model, two key papers and code bases are used: SailFish [12] and MPQUIC-SBD [20]. Following the implementation, the evaluation and data used is described.

4.1 Scheduler & Reinforcement Learning model

The core investigatory part of the paper is the model used for scheduling. This constitutes a reinforcement learning based model that will perform scheduling and use those experiences to train its future scheduling decisions. A reinforcement learning model is chosen due to the ability to perform training without labels, by optimizing a certain reward instead. Deep Q-learning is specifically chosen due to its ability to estimate the state action values given a reward function, states, and possible actions. A variant of Deep-Q called "Advantage Actor-Critic"(A2C) is used due to stabilizing effects in learning and faster training when compared to a deep neural network [18].

In this paper, a fully online model is used. The model will simultaneously train and perform predictions using the same weights, in contrast to an offline model which keeps track of 2 separate model instances where one trains and the other predicts. This is done to allow for easy modification to the base model, being simpler and allowing for better isolation of the effect meta-learning will have.

One of the important design decisions for a scheduler is the level that decisions are made. Algorithmic approaches usually perform scheduling per packet. For specific scenarios, like stream-based web object download, a stream scheduling approach has been used [12]. In a video streaming scenario, there are no concurrent streams of data, making a per-packet scheduler the most valid approach. However, due to the high bandwidth of modern networks and low packet size, scheduling needs to be done with above 500 packets per second frequency. Due to machine learning model inference time and other overheads, it is not always feasible to operate at those frequencies without causing packet delays. To remedy this problem specifically for video streaming, a paper proposed an RL scheduler that makes decisions at a set interval of 50ms [17]. These actions are then valid for the next 50ms after a decision is made. This does limit the granularity of the scheduling decisions given, but it is a necessary compromise given a complex model. Thus the defined model in this paper will be an asynchronous packet scheduler with static time interval updates.

As a q-learning model, there are 3 main components to define the model: reward, state, and actions.

Reward: As a proxy for QoE, download bandwidth at the client is used as the reward, expressed as Mb/s for the most recent segment. This ensures the model will prioritize the quick delivery of video segments at the client side, instead of purely focusing on server bandwidth.

State: The state of the model pertains to different network conditions for each path. Path specific metrics are defined as standard network measures, as used in previous research: packet loss ratio, mean round trip time, and estimated bandwidth in Kb/s. Bandwidth is estimated given the congestion window over the past 10 ACK packets. These *3* metrics are provided per each path, resulting in an input length of *6*.

Actions: A per-packet scheduler typically selects one path of given options. In this environment, there are 2 paths available. These can be selected directly, but due to the 50ms interval of action application, more granularity is needed. An approach that is possible is to define a set of actions based on probability to put a packet on a specific path [17]. Therefore, the action space for the used scheduler is **0** for using the baseline minRTT model, with actions **1-5** defining the probabilities P_1 and P_2 for paths 1 and 2 respectively, using the formula:

$$P_1 = 1 - P_2$$
$$P_2 = \frac{action - 1}{4}$$

As it can be seen from the model definition, it does not aim to directly optimise QoE. This decision is taken as the specific adaptive bitrate algorithm used for selecting the bitrate for the next segment also affects QoE metrics. Designing a good reward function that takes into account all QoE features is also difficult, and previously proposed compound QoE metrics do not necessarily equate to good reward functions.

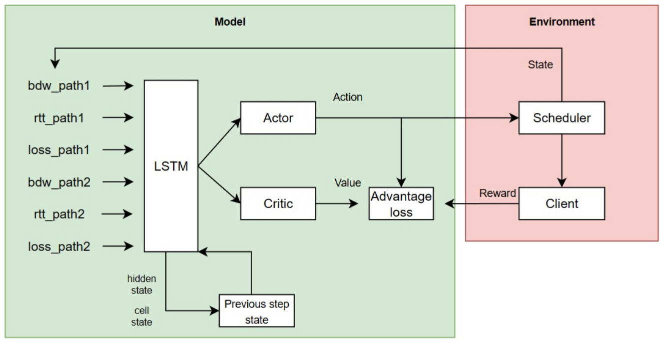
To perform meta-learning, an LSTM based approach is used to modify the existing model to accommodate temporal information. The visualisation of this model can be seen in Figure 1. It uses a similar structure as defined in [30], where state input is first fed into an LSTM, and then the output of the LSTM is propagated to the actor and critic. The hidden and cell states of the LSTM are stored to be used for the next forward pass. After a network change is detected, the hidden state is reset, while the cell state is maintained. The values of the cell state are persisted across episodes and across training, testing. Backpropagation on the hidden state and cell state is performed over the n-step batch. By storing memory about previous states and resetting them when needed, it is expected that the model would be able to use previous network conditions when making predictions, and also forget irrelevant short-term information when needed.

Like any reinforcement learning model, there needs to be a consideration for exploration vs exploitation. In particular, the strategy of ϵ-greedy is used, where with ϵ probability a random action is chosen from the action space instead of using the one found best by the model. In this video streaming scheduling environment, there is a risk that a randomly chosen action could cause a badly scheduled packet, thus resulting in reduced bandwidth. Due to this reason, the ϵ value differs on whether the model is in training or testing, having a larger value for training to properly explore all possible actions.

As the model needs to quickly adapt to changing network conditions, performing backpropagation per episode can be too limiting. Therefore, a technique called n-step bootstrapping is used, where n is the number of steps after which backpropagation is performed on. Memory of past n actions and rewards is used to calculate the loss. Having a larger n will cause loss calculation to consider the action-reward relation with a large time space, while a smaller value will allow for faster adaptation to the immediate situation.

The environment does not have hard task boundaries, thus to perform meta-learning and find these boundaries, sequentially discounting autoregressive online change point detection is used [33]. For each observed state by the model, an instance of the algorithm is created and then constantly updated with each model step. With each step, an anomaly score is outputted that indicates the likelihood of a significant change from the previously observed data. By setting a threshold *thresh* that the anomaly score has to reach, a boundary can be set at that point in time where the threshold was crossed. To remedy constant switching, there is also a cool-down period *cdp* in steps during which values above the threshold are ignored as an indication of a change. As each input value has a separate instance of the algorithm and a separate anomaly score, all scores are aggregated into a single value by cumulatively multiplying all anomaly scores. In contrast to taking the mean of the scores, this approach better captures a large change in one or more of the inputs, allowing for detection in one of the paths or in one of the network quality metrics.

Figure 1: LSTM+actor critic meta-learning model



4.2 Testing Framework

Training and testing the developed schedulers requires a framework for performing scheduling of packets that are sent to a client, while also retrieving QoE characteristics of the client's experience. The outline of the framework can be seen in Figure 2. The core of the whole system is the mininet [5] network simulation, which will route all packets sent between the client and server with defined network constraints. The client will request videos from the server, which will then be routed by the scheduler across multiple paths. QoE metrics are collected client side and then returned to the scheduler to perform training.

To ease providing QoE metrics from the client to models and to keep the implementation of MPQUIC model-agnostic, there is no modification of the MPQUIC communication protocol. Metrics are provided directly to the model using messaging sockets [1].

As the MPQUIC implementation is written in Go and the model is in Python, an intermediate layer is needed for communication between the processes. To allow for flexibility and for practical reasons given the experiments are run in a virtual machine (VM), this communication is handled using networking based sockets [1]. An implementation proposed in a reinforcement learning based scheduler paper [12] is used, where a process called the middleware routes traffic between the VM's scheduler and the machine learning model run on the host machine. The state of the network available at the QUIC level is sent as a request to the model, while the model responds with the action the scheduler should take.

The call to the middleware and forwarding of the scheduler request is implemented in an asynchronous manner. While waiting for the response from the model, the MPQUIC scheduler will perform the last returned action. This way the latency of the model and the networking overhead do not inhibit the regular operation of packet scheduling.

Finally, the specific interaction with the MPQUIC scheduler is important. The scheduler is able to return a path for a packet to be routed through or a response indicating no paths are available. Paths can become temporarily unavailable, meaning the probabilistic path-picking action from the proposed model breaks down. A possible approach is to route all packets to the available path, but that can cause performance degradation, as argued in [17]. If the path that is unavailable also happens to be substantially faster than any other option, the slow path can become overloaded with too much traffic. This is the default behaviour of minRTT, which can be argued to be one of its downsides. To make the proposed model have more control and patch gaps in minRTT's strategy, the scheduler is designed to not route all packets to the only available path. It instead follows the probabilities given by the active model action, and when a packet is picked to be scheduled on a non-available path, it is indicated to the MPQUIC packet sender that no paths are available instead. This approach allows the model to choose to move all traffic on a single path or wait for the fast path to become available instead.

4.3 Network setup

The primary network constraints, also defined as key performance indicators (KPIs), are comprised of bitrate, latency and packet loss. Bitrate is the main throughput limitation of a path, but is affected by other factors, like latency which causes delays for the user to start receiving requested data and also further reduces throughput by introducing communication overhead when new video pieces are requested. Packet loss is the proportion of dropped or lost packets while performing transmission, then causing expensive re-transmission of the same packets.

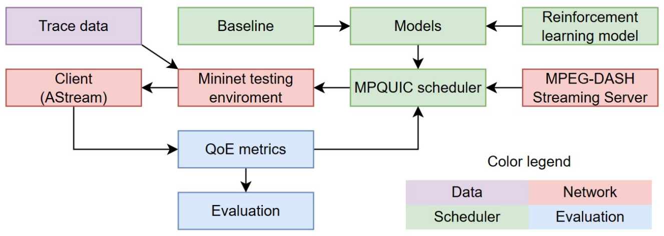
To allow for multipath usage with mobile networks, two separate connections need to be set up for both the client and server. For mobile networks, that would involve two separate 5g or WLAN connections. It is possible to simulate these connections fully by creating a 2d space and placing the client and connection points on this space, like done in [29]. This however represents extra layers of complexity, like needing to simulate the mobility patterns of the user and determining how signal strength/distance relates to KPIs and affects bandwidth [34]. Therefore, a data-driven approach is used. This involves using traces recorded on real world networks with changes in network conditions over time, like lower bandwidths and switches between cell towers [10]. Each time step (1 second in this case), the new conditions are applied to the network in the mininet simulation, which then requires the scheduler to adapt to these changes.

In mininet, the network is configured with two separate Linux switches, that are connected the client to the server. For the incoming and outgoing packets, limitations on bandwidth and artificial packet loss and latency will be applied. Due to the multipath testing scenario, a different set of KPIs are applied to the switches, needing a *pair* of traces or data. To simulate a static streaming scenario, a pair of a static 5g trace and WLAN KPIs are used. For a mobile scenario, 2 different mobile 5G traces are used. These will be referred to as *static test pairs* and *mobile test pairs*. The dataset used is described in detail in subsection 4.5.

4.4 Evaluation

The goal of this paper is to evaluate the investigated models specifically with the use of quality of experience. However, QoE evaluation is a complex field with many available abstractions. Studies have been done to compare different QoE estimation algorithms [2]. Due to good performance, simple implementation and data requirements, the Claye QoE model is used [22]. It considers *bit-rate*, *re-buffering ratio* and *video startup delay* to produce a single numerical representation of QoE in a range of $[0, 5]$. Bit-rate indicates the quality of the video received. Re-buffering ratio is the ratio between the time spent waiting for the video buffer to receive new data and the total time of the video watch time. Video startup delay is the latency between requesting a video and the actual start time of the video stream on the user end. Bitrate is positively correlated with QoE while re-buffering and startup delay are negatively correlated. These metrics are often used due to their objectivity and ease of collection given a simulated client and network, and they have also been shown to correlate individually with user-reported QoE [19] [15]. These metrics have been used in previous literature [34] [10]. Thus, the 3 component parts of the Claye model are also collected and presented. As a general metric for comparison, client-side *bandwidth* is also collected, representing the speed at which the most recent segment was downloaded with.

Figure 2: Full Testing Framework outline



To measure these metrics a virtual DASH video player (AStream) [21] is used which collects various playback information, including the mentioned metrics, which can be sent to the server to train the scheduler.

Multiple models will be evaluated, focusing on the baseline model MinRTT [4], which provides easy comparisons to other papers due to its common usage as a baseline model and it being the default MPQUIC scheduler. Furthermore, the defined base reinforcement learning model is also tested, to provide a base performance without meta-learning. Finally, the studied meta-learning strategy is evaluated. Comparison with baseline models in a simulated mobile network provides a clear-cut establishment of the true effectiveness of reinforcement-learning based models.

To train the models, a main train set is used. Then, online training is performed on the test set. Both sets will be of equal length. Evaluation metrics will be taken from the results on the test set, as this shows the ability of the model to adapt to an unseen environment with limited time, modelling the use case of a video streaming app.

Mobility is also a big factor when considering the performance of the models tested, thus the performance of the models will also be presented separately for static and mobile test pairs. To properly evaluate the impact of mobility of a specific model alone, performance on both mobility types is presented for comparison. These classes of mobility are not balanced in their KPIs, thus to make the side-by-side comparison valid, final results need to be adjusted based on available base bandwidth. The results are normalized by the ratio difference between the mean bandwidth of static and mobile test pairs. The presented QoE metrics are not linearly correlated with bandwidth, thus only bandwidth is adjusted for and presented for the comparison. This way, the impact of mobility on performance is isolated from the underlying differences in the data.

4.5 Data

There are 2 primary types of data required for the setup of the testing environment: traces to model network conditions and videos to be played with AStream.

First, a network trace dataset is needed. One of the most comprehensive and recent datasets includes LTE(4G) and 5G network datasets, gathered in Ireland in 2018 and 2020 by the same group of researchers [24][23]. The usage of 4G could be beneficial for a possible comparison with 5G and the effect on a scheduler, however, there are some differences between these datasets. As defined in subsection 4.2, there is a need for 3 main KPIs for a full definition of a connection. The 4g dataset lacks latency information, while the 5G dataset contains it. Therefore, it is chosen to only use the 5G dataset in this paper.

The trace dataset provides KPI measurements for download and video streaming loads over static and mobility scenarios, with 83 traces overall and a resolution of 1 second. To remove the effects of the specific streaming app used in the video streaming traces, only the file download sub-dataset is used, representing the practical maximum of the bandwidth available.

This data is further processed by filling in missing value gaps in bandwidth, packet loss and latency. It is noticed that in some dataset entries bandwidth is reported as very low (below 100kb) when surrounding values are much higher, even when it is indicated that downloading was performed during that second. The dataset also only shows reports of packet loss and latency every 10 seconds, therefore missing values are given the values of the nearest known observation. Then, over bandwidth, packet loss and latency, a rolling average is performed with a window size of 15 seconds. After filling in missing values, all rows with missing values are removed. To remove the possibility of total connection loss, the bandwidth is set to a minimum value of 50Kb/s. Then, only rows where the download was active are kept.

Finally, each trace is filtered to ensure it is at least as long as the video being streamed, which in this case is 586 seconds. To allow for video buffering that causes video playback to finish later, traces that are at least 800 seconds are kept. This results in 9 total traces, with 4 of them being static, and the rest being mobile.

In regards to WLAN KPI measurements, those are often recorded locally due to large variance, as a sample by the authors. For ease of comparison, these measurements are taken as provided by the FALCON[31] paper.

As detailed in subsection 4.2, pairs of mobile traces and static traces with WLAN are generated. For static traces, each available static 5G trace is combined with the WLAN KPI, giving 5 pairs. For mobile pairs, each possible pairing of 2 different 5G traces is taken, resulting in 15 pairs. All pairs are concatenated into a single list and then split in half. This results in test and train sets with 10 pairs each. The train data contains 7 static and 3 mobile pairs, while the test data contains 8 and 2 mobile and static pairs respectively. This represents some bias toward a more mobile approach, which is accurate to the intended use case of a video streaming application. The differences in class distribution between test and train sets may provide more challenges to reinforcement learning models, which is important when considering a real-world scenario.

Finally, the second data source provides the data being sent over the network, which involves seven videos or varying origin and length [25]. The videos are provided in compliance with the DASH [26] technology, allowing adaptive bit-rate video streaming. In this paper, the animation video "Big Buck Bunny" is used. The video is split into equal length segments of 2 seconds and streamed to the client from a DASH web server [14]. To allow full saturation of multiple 5g connections, the video has a max bitrate of 20 Mbps. Other bitrates provided follow an approximate halving of the bitrate down to a minimum of 0.25 Mbps.

4.6 Experimental setup

The A2C model uses a double layer fully connected network with a network shape of *(6, 256, 6)* for the actor and *(6, 256, 1)* for the critic. The LSTM model adds an extra LSTM layer before the A2C model with a hidden size of 32, thus resulting in a final network shape of *(6, 32, 256, 6)* for the actor, and *(6, 32, 256, 1)* for the critic. ReLU activation function is used for outputs between layers, while softmax is used for the output of the actor. Following values seen in previous research [31], the Adam optimiser is used with a learning rate of 10^{-3}. Training and testing ϵ-greedy values are set to 0.15 and 0.01 respectively. Change detection threshold *thresh* and cooldown *cdp* are set to *50* and *60* respectively, following manual testing of the best values.

The testbed setup is a Windows 10 host with a Ryzen Zen 3 architecture based 8 core CPU and 128GB ram. The test VM is running an Ubuntu 14.04 distribution with 8 logical cores and 8 GB of RAM allocated.

5 RESULTS

The results for each tested model on the test set are presented in this section. The tested models were: LSTM based meta-learning model (lstm), baseline minRTT and non meta-learning actor critic(a2c). The reinforcement learning models were pre-trained on a separate train set of 10 trace pairs. Each model was run on the test set containing 10 trace pairs, resulting in 8000 seconds of unique network KPIs. Both combined and separate mobility results are presented. Separate results are calculated per type of trace mobility.

5.1 Mobility impact on performance

To understand the difference mobility makes on a model's performance, separated results by mobility are presented. Bandwidth is adjusted to account for differences between mobile and static traces. Trace pairs between static and mobile scenarios involved had mean bandwidths of 38.6 and 48.4 Mbps respectively. Thus bandwidth was raised for mobile traces with a multiplier of *1.25*. The results are presented in figure 3.

Firstly, for each model static traces provided better performance than mobile ones, on average 27% (9.67 vs 12.34 Mbps) for all models. The standard deviation of the bandwidths was also larger for static traces at difference of 25% (2.67 vs 3.36 Mbps). The performance on static traces are the same for all the models, but there are major

Mobility	Model	QoE	Bandwidth (Mbps)	Bitrate (Mbps)	Buffering ratio (%)	Initial buffering (s)
mobile	a2c	1.45	8.83	8.93	0.06	0.61
	lstm	1.63	9.63	9.66	0.04	0.64
	minrtt	**1.85**	**10.55**	**10.27**	**0.00**	**0.58**
static	a2c	**1.01**	**9.89**	**9.96**	**0.33**	**0.39**
	lstm	1.00	9.88	9.87	1.19	0.41
	minrtt	0.83	9.80	9.64	1.45	**0.39**

Table 1: Comparison of tested schedulers with mobile and static traces, showing mean claye QoE, bandwidth and bitrate in megabits per second, buffering ratio in percent of playtime, and initial buffering in seconds.

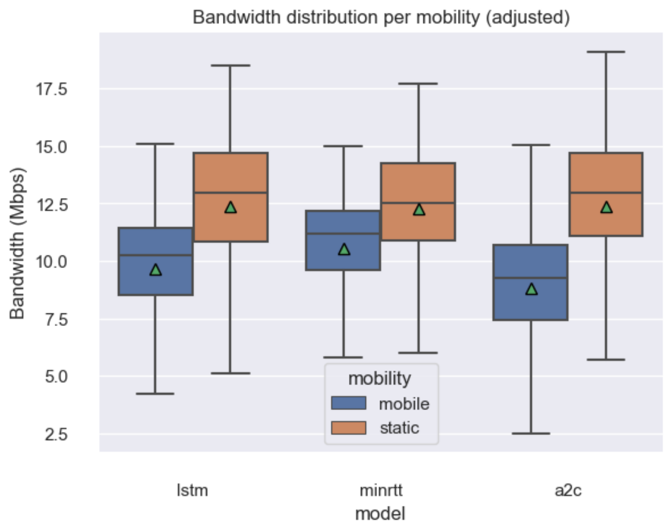


Figure 3: Comparison of video download bandwidth distribution between static and mobile environments for all schedulers. Adjusted for mean bandwidth difference between static and mobile traces.

differences when looking and mobile traces. The baseline minRTT scheduler was the most similar between mobilities (static-mobile ratio of 1.16), while a2c had the largest difference (ratio of 1.4).

5.2 Meta learning

To evaluate the impact of meta-learning and the performance of reinforcement learning, the presented results are no longer adjusted for mobility types. The overall test data QoE distribution for each model is shown in figure 4. It is evident from the results, that the baseline minRTT outperforms both RL models, with 21% better QoE compared to A2C and 10% for LSTM. QoE and bandwidth both showed similar performance differences when compared to minRTT. This data demonstrates that the meta-learning LSTM model provides an improvement over the base RL model, with a difference of 11% better QoE.

A breakdown of all metrics and separate statistics per mobility can be seen in table 1. It can be seen that both RL models performed the same on static traces across most metrics, except Initial buffering. They also both outperformed minRTT in a static scenario, with an improvement of 21% in QoE. This improvement can be mostly explained by the difference in buffering ratio, with 4.4 times less buffering in A2C when compared to minRTT. The achieved bandwidth was similar, thus the RL models provided more stability instead of pure bandwidth.

In terms of mobile performance, the difference between the baseline and proposed RL models followed the same pattern as in the general comparison. The RL models both had more buffering and lower bandwidth when compared to minRTT. The meta-learning based LSTM model again outperformed A2C, with a 12% higher QoE.

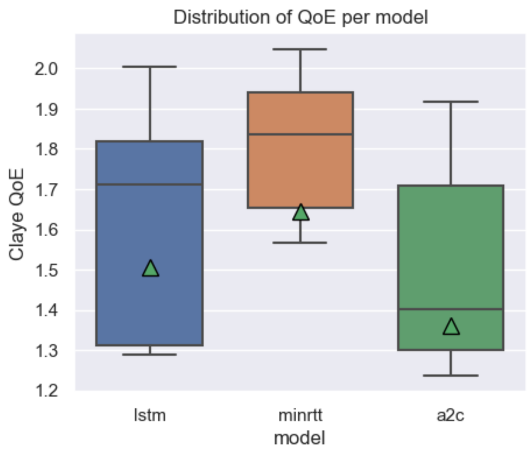


Figure 4: Distribution of Claye quality of experience per tested model, run on test pairs of all mobilities.

6 DISCUSSION

6.1 Comparison to state of the art

For both RL models, performance was worse than the baseline in mobile scenarios. This is expected of a purely online model, as shown in [31], but with lower performance than observed in the mentioned paper (A2C is worse by 20%, compared to 7% in the reference paper). This difference can be possibly explained by the difference in environment setups between papers, caused by a different application of the scheduler. The performance of RL models is influenced by many factors including reward-action delay, interactivity with the environment and sample complexity [11].

Online algorithms in general do not seem to be suitable for mobile environments, having a hard time adapting to quick changes. This effect is likely caused by the model overfitting on most recent observations when predicting the next action, which has been observed in previous research [30]. Meta-learning reduces this overfitting but does not fully overcome the underlying limitations of the online approach, performing worse than an offline model.

A lot of meta-learning research focuses on highly controlled and simulated environments, allowing the replay of predefined environments with known boundaries for the model to learn on [30]. The performance of online RL models is sufficient given these simpler conditions, but may not apply to the vastly different MPQUIC environment. The most promising solution to this problem in the MPQUIC field is the use of combined online-offline models, like that used in FALCON [31]. It is likely the use of both the online-offline model and meta-learning are what provides FALCON with large performance improvements.

The performance seen in static scenarios shows to be partly in line with other RL models, indicating no gain in bandwidth, when others did report an increase of 20% when compared to minRTT. However, the improvement in QoE did show to be in the range of 20-25% as seen in other papers. This difference in metrics performance does indicate that making a scheduler's reward and evaluation rely only on bandwidth could lead to inaccurate evaluation and sub-optimal training. The result reinforces the observation that online models are best suited for mostly static environments, due to good adaption to most recently seen states.

6.2 Limitations

The main limitation of this paper is that the main SOTA models were not tested in the created environment. This can cause differences in observed performance, specifically given the mobility and video streaming setup. MinRTT is used as a proxy for performance comparisons, but it does not necessarily follow the same evaluative patterns as RL models.

With the RL models proposed in the paper, a specific set of hyperparameters was used and no automatic and extensive tuning was done. Given different values, the performance indications could be different, primarily when compared to other papers that did invest in hyperparameter tuning.

Model-wise, the strategy used for setting up the actions is not easily extensible in a scenario with more than 2 paths, with each path growing the action space exponentially. A modification of the model with a continuous action space could alleviate this problem.

While the dataset used was extensive, the validity of the model can be further improved by using more data. The use of 4G and WLAN traces could also provide another dimension to compare the results.

Finally, the construction of the mobility testing setup is not entirely realistic. Using 2 different traces without consideration for the space and actual movement of a client is an approximation of a real-world scenario. It is presumed, that the large amount of data used will still provide environments with realistic outcomes.

7 CONCLUSION

This paper analysed the effect that applying meta-learning has in improving the performance of an MPQUIC scheduler, looking at the quality of experience in a video streaming context, while simulating a changing mobile environment.

First, an inquiry is performed into the field of meta-learning and its applicability for MPQUIC scheduling. It is noted that meta-learning has been used in the field for MPQUIC, showing improvements over a regular machine-learning approach. However, the use of generalized meta-learning frameworks and approaches is difficult due to the constantly online nature of the scheduler's environment. An LSTM based meta-learning approach is identified as a suitable option. Then, for the core of the paper, the exact benefit meta-learning provides to the scheduler is investigated. Overall, applying meta-learning on a regular reinforcement learning model improves QoE performance by 11%. The main advantage observed occurs in mobile scenarios, while static scenarios show no improvement. It does not however outperform a minRTT baseline. Finally, the aspect of mobility is assessed. It is shown that mobility causes a degradation in performance for all schedulers, particularly reinforcement learning based ones.

By considering these aspects in MPQUIC scheduling, this paper offers the first analysis of the use of meta-learning in this field, simplifying further research in this direction. A video streaming, quality of experience-oriented framework is developed, showing the first use of data-driven simulation of real-world mobility patterns in the scheduler field, allowing testing of other schedulers in this framework. Finally, a new meta-learning model is proposed with a comprehensive comparison with the baseline and a regular reinforcement learning model.

Some limitations are presented in the paper, but none significantly affect the conclusions reached. The main limitation of this paper is the use of a baseline as a proxy to compare to other models, instead of testing them using the same environment used in this paper.

To answer the main question, meta-learning can be used to improve scheduler performance, but it alone does not provide enough improvement to replace the existing baseline. Thus, further research is needed to improve other aspects of the model and improve the performance past the baseline. This could be done by performing scheduling per packet instead of being time-based, or altering the state and reward. Other meta-learning approaches can also be attempted to be applied in this environment. Finally, the use of different base model, like a combination online-offline setup seen in state-of-the-art research, is a promising direction to improve the model and further asses the usefulness of meta-learning.

REFERENCES

[1] ZeroMQ authors. 2022. ZeroMQ: An open-source universal messaging library. https://zeromq.org/

[2] Nabajeet Barman and Maria G. Martini. 2019. QoE Modeling for HTTP Adaptive Video Streaming–A Survey and Open Challenges. *IEEE Access* 7 (2019), 30831–30859. https://doi.org/10.1109/ACCESS.2019.2901778

[3] Jacob Beck, Risto Vuorio, Evan Zheran Liu, Zheng Xiong, Luisa Zintgraf, Chelsea Finn, and Shimon Whiteson. 2023. A Survey of Meta-Reinforcement Learning. arXiv:2301.08028 (Jan 2023). https://doi.org/10.48550/arXiv.2301.08028 arXiv:2301.08028 [cs].

[4] Sebastien Barre Christoph Paasch et al. 2017. Multipath TCP in the Linux Kernel. available from http://www.multipath-tcp.org.

[5] Mininet Project Contributors. 2022. Mininet: An Instant Virtual Network on Your Laptop (or Other PC) - Mininet. http://mininet.org/

[6] Chelsea Finn, Pieter Abbeel, and Sergey Levine. 2017. Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks. arXiv:1703.03400 [cs.LG]

[7] R. R. Fontes, S. Afzal, S. H. B. Brito, M. A. S. Santos, and C. E. Rothenberg. 2015. Mininet-WiFi: Emulating software-defined wireless networks. In *Network and Service Management (CNSM), 2015 11th International Conference on.* 384–389. https://doi.org/10\.1109/CNSM.2015.7367387

[8] James Harrison, Apoorva Sharma, Chelsea Finn, and Marco Pavone. 2020. Continuous Meta-Learning without Tasks. arXiv:1912.08866 [cs.LG]

[9] Timothy Hospedales, Antreas Antoniou, Paul Micaelli, and Amos Storkey. 2020. Meta-Learning in Neural Networks: A Survey. https://doi.org/10.48550/ARXIV.2004.05439

[10] Cyriac James, Emir Halepovic, Mea Wang, Rittwik Jana, and N. K. Shankaranarayanan. 2016. Is Multipath TCP (MPTCP) Beneficial for Video Streaming over DASH?. In *2016 IEEE 24th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS).* 331–336. https://doi.org/10.1109/MASCOTS.2016.75

[11] Sham Kakade. 2003. On the Sample Complexity of Reinforcement Learning. https://homes.cs.washington.edu/~sham/papers/thesis/sham_thesis.pdf

[12] Marios Evangelos Kanakis. 2020. *A Learning-based Approach for Stream Scheduling in Multipath-QUIC.* https://linwang.info/thesis/thesis20-kanakis.pdf

[13] Fernando Kuipers, Robert Kooij, Danny De Vleeschauwer, and Kjell Brunnström. 2010. Techniques for Measuring Quality of Experience. 216–227. https://doi.org/10.1007/978-3-642-13315-2_18

[14] Light Code Labs. 2023. Caddy 2 - The Ultimate Server with Automatic HTTPS. https://caddyserver.com/

[15] Khalil ur Rehman Laghari, Omneya Issa, Filippo Speranza, and Tiago H. Falk. 2012. Quality-of-Experience perception for video streaming services: Preliminary subjective and objective results. In *Proceedings of The 2012 Asia Pacific Signal and Information Processing Association Annual Summit and Conference.* 1–9.

[16] Adam Langley, Al Riddoch, Alyssa Wilk, Antonio Vicente, Charles 'Buck' Krasic, Cherie Shi, Dan Zhang, Fan Yang, Feodor Kouranov, Ian Swett, Janardhan Iyengar, Jeff Bailey, Jeremy Christopher Dorfman, Jim Roskind, Joanna Kulik, Patrik Göran Westin, Raman Tenneti, Robbie Shade, Ryan Hamilton, Victor Vasiliev, and Wan-Teh Chang. 2017. The QUIC Transport Protocol: Design and Internet-Scale Deployment.

[17] Seunghwa Lee and Joon Yoo. 2022. Reinforcement Learning Based Multipath QUIC Scheduler for Multimedia Streaming. *Sensors* 22, 17 (Aug 2022), 6333. https://doi.org/10.3390/s22176333

[18] Volodymyr Mnih, Adrià Puigdomènech Badia, Mehdi Mirza, Alex Graves, Timothy P. Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. 2016. Asynchronous Methods for Deep Reinforcement Learning. arXiv:1602.01783 [cs.LG]

[19] Ricky Mok, Edmond Chan, and Rocky Chang. 2011. Measuring the Quality of Experience of HTTP Video Streaming. *Proc. of IFIP/IEEE International Symposium on Integrated Network Management (IM)*, 485–492. https://doi.org/10.1109/INM.2011.5990550

[20] Thomas W. P. Paiva, Simone Ferlin, Anna Brunstrom, Ozgu Alay, and Bruno Y. L. Kimura. 2023. A First Look at Adaptive Video Streaming over Multipath QUIC with Shared Bottleneck Detection. *To appear in Proceedings of The 14th ACM Multimedia Systems Conference (MMSys'23).*

[21] pari685. 2018. pari685/AStream: A DASH segment size aware rate adaptation model for DASH. https://github.com/pari685/AStream

[22] Stefano Petrangeli, Jeroen Famaey, Maxim Claeys, Steven Latré, and Filip De Turck. 2015. QoE-Driven Rate Adaptation Heuristic for Fair Adaptive Video Streaming. *ACM Trans. Multimedia Comput. Commun. Appl.* 12, 2, Article 28 (oct 2015), 24 pages. https://doi.org/10.1145/2818361

[23] Darijo Raca, Dylan Leahy, Cormac J. Sreenan, and Jason J. Quinlan. 2020. Beyond Throughput, the next Generation: A 5G Dataset with Channel and Context Metrics. In *Proceedings of the 11th ACM Multimedia Systems Conference* (Istanbul, Turkey) *(MMSys '20).* Association for Computing Machinery, New York, NY, USA, 303–308. https://doi.org/10.1145/3339825.3394938

[24] Darijo Raca, Jason J. Quinlan, Ahmed H. Zahran, and Cormac J. Sreenan. 2018. Beyond Throughput: A 4G LTE Dataset with Channel and Context Metrics. In *Proceedings of the 9th ACM Multimedia Systems Conference* (Amsterdam, Netherlands) *(MMSys '18).* Association for Computing Machinery, New York, NY, USA, 460–465. https://doi.org/10.1145/3204949.3208123

[25] Christopher Müller Stefan Lederer and Christian Timmerer. 2012. "Dynamic Adaptive Streaming over HTTP Dataset". In *In Proceedings of the ACM Multimedia Systems Conference 2012* (Chapel Hill, North Carolina).

[26] Thomas Stockhammer. 2011. Dynamic Adaptive Streaming over HTTP –: Standards and Design Principles. In *Proceedings of the Second Annual ACM Conference on Multimedia Systems* (San Jose, CA, USA) *(MMSys '11).* Association for Computing Machinery, New York, NY, USA, 133–144. https://doi.org/10.1145/1943552.1943572

[27] Raza Ul Mustafa, Md Tariqul Islam, Christian Rothenberg, Simone Ferlin, Darijo Raca, and Jason J. Quinlan. 2020. DASH QoE Performance Evaluation Framework with 5G Datasets. In *2020 16th International Conference on Network and Service Management (CNSM).* 1–6. https://doi.org/10.23919/CNSM50824.2020.9269111

[28] Tobias Viernickel, Alexander Froemmgen, Amr Rizk, Boris Koldehofe, and Ralf Steinmetz. 2018. Multipath QUIC: A Deployable Multipath Transport Protocol. In *2018 IEEE International Conference on Communications (ICC).* 1–7. https://doi.org/10.1109/ICC.2018.8422951

[29] Minh Hai Vu, Giang T. T. Nguyen, Hai Dang Tran, Thanh Trung Nguyen, Phan Thuan Do, Phi Le Nguyen, and Kien Nguyen. 2022. An Empirical Study of MPQUIC Schedulers in Mobile Wireless Networks *(SoICT '22).* Association for Computing Machinery, New York, NY, USA, 193–200. https://doi.org/10.1145/3568562.3568654

[30] Jane X Wang, Zeb Kurth-Nelson, Dhruva Tirumala, Hubert Soyer, Joel Z Leibo, Remi Munos, Charles Blundell, Dharshan Kumaran, and Matt Botvinick. 2017. Learning to reinforcement learn. arXiv:1611.05763 [cs.LG]

[31] Hongjia Wu, Ozgu Alay, Anna Brunstrom, Giuseppe Caso, and Simone Ferlin. 2022. FALCON: Fast and Accurate Multipath Scheduling using Offline and Online Learning. https://doi.org/10.48550/ARXIV.2201.08969

[32] Hongjia Wu, Ozgu Alay, Anna Brunstrom, Simone Ferlin-Reiter, and Giuseppe Caso. 2020. Peekaboo: Learning-Based Multipath Scheduling for Dynamic Heterogeneous Environments. *IEEE Journal on Selected Areas in Communications* PP (03 2020). https://doi.org/10.1109/JSAC.2020.3000365

[33] Kenji Yamanishi and Jun-ichi Takeuchi. 2002. A Unifying Framework for Detecting Outliers and Change Points from Non-Stationary Time Series Data. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (Edmonton, Alberta, Canada) *(KDD '02).* Association for Computing Machinery, New York, NY, USA, 676–681. https://doi.org/10.1145/775047.775148

[34] Wang Yang, Jing Cao, and Fan Wu. 2021. Adaptive Video Streaming with Scalable Video Coding using Multipath QUIC. In *2021 IEEE International Performance, Computing, and Communications Conference (IPCCC).* 1–7. https://doi.org/10.1109/IPCCC51483.2021.9679445

[35] Zhilong Zheng, Yunfei Ma, Yanmei Liu, Furong Yang, Zhenyu Li, Yuanbo Zhang, Jiuhai Zhang, Wei Shi, Wentao Chen, Ding Li, Qing An, Hai Hong, Hongqiang Harry Liu, and Ming Zhang. 2021. XLINK: QoE-driven multi-path QUIC transport in large-scale video services. *Proceedings of the 2021 ACM SIGCOMM 2021 Conference* (Aug 2021). https://doi.org/10.1145/3452296.3472893