Hindawi
Mobile Information Systems
Volume 2020, Article ID 6673134, 12 pages
https://doi.org/10.1155/2020/6673134

*Research Article*

# Failure-Aware and Delay-Predicted Multipath Virtual Queue Scheduling for Multimedia Transmission in Edge IoT

**Jiuren Qin**,[1] **Zhaoxue Wang**,[2] **Kai Gao**,[1] **and Lujie Zhong**[2]

[1]*State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing 100876, China*
[2]*Information Engineering College, Capital Normal University, Beijing 100048, China*

Correspondence should be addressed to Lujie Zhong; zhonglj@cnu.edu.cn

Received 20 October 2020; Revised 3 November 2020; Accepted 12 November 2020; Published 27 November 2020

Academic Editor: Xiaohong Jiang



The spread of Edge Internet of Things (IoT) radically changes our lifestyle. However, the multimedia services in edge IoT are still stuck by inefficiency. The dynamic typologies perplex the transmission of massive real-time data. To solve this problem, multipath transmission control protocol (MPTCP) which has a natural advantage in transmission robustness and bandwidth aggregation is becoming a good choice. In this paper, failure-aware and delay-predicted multipath virtual queue scheduling (FD-MVQS) is proposed to optimize the MPTCP performance in edge IoT. FD-MVQS constructs a two-plane cooperative scheduling system. In the control plane, the transmission failure estimation and chaos theory-based arrival delay prediction methods are introduced to provide the foundation for prescheduling. In the data plane, the multipath virtual queue scheduling is designed to allocate segments to different subflows. Simulation results showed that the proposed FD-MVQS performed better than standard and typical multipath transmission solutions in throughput, delay, and segment disorder.

## 1. Introduction

Recently, edge IoT experiences a rapid development benefiting from the advances in communication technologies (e.g., 5G and 801.11ax) [1, 2]. Thanks to this, the multimedia services such as self-driving, video surveillance, and augmented reality are also more and more applied in edge IoT. However, the transmission in edge IoT is still facing the problems of low throughput and unreliability caused by the complex and changeable networks [3]. The transmission requirements of multimedia services are often not met. The multipath transmission control protocol (MPTCP) standardized in RFC8684 [4] gives a new solution. Figure 1 illustrates a typical multipath transmission scenario in edge IoT. MPTCP can create multiple subflows between the edge server and the consumer through different network interfaces. The bandwidths of different networks are aggregated by transmitting data concurrently. The transmission robustness can also be improved compared with the single-path protocol for the reason that traffic can be transferred to other subflows when some subflows get into error. However, the node diversity in edge IoT makes the bandwidth and delay of subflows to be quite different which leads to massive disordered segments. These segments will deplete the limited receiver buffer and result in throughput degradation. Thus, scheduling the segments among subflows is a very important issue for multipath transmission.

The default scheduling algorithm of the MPTCP is LowRTT [5] which will allocate segments to the subflow with lowest round-trip time (RTT). LowRTT performs well in the networks with small subflow delay gap and low packet-loss rate but is not suitable for edge IoT. The limitations of LowRTT can be formulated as follows:

(1) Fail unaware: RTT allocates the segments without considering the transmission failure. The congestion and link error can both lead to the transmission failure. These segments must be retransmitted according to the MPTCP, which introduces new

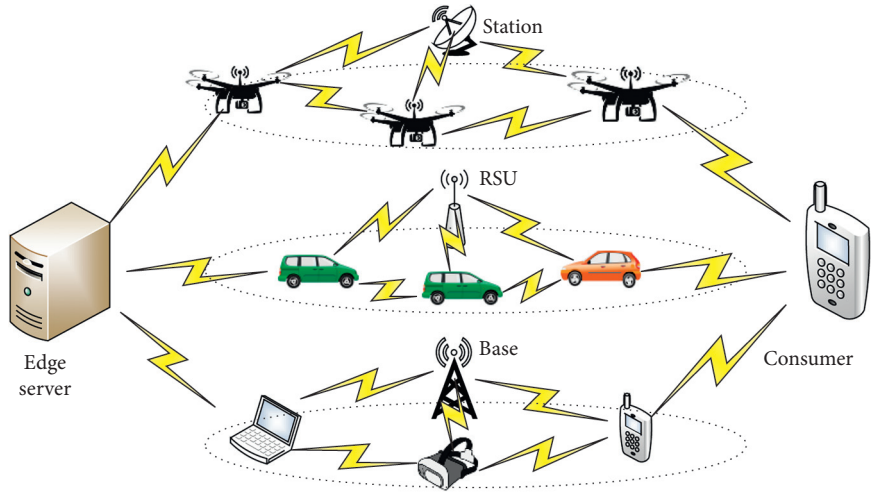


Figure 1: The typical multipath transmission scenario in edge IoT.

transmission delay. Besides, the congestion algorithms of the MPTCP will decrease the congestion window (cwnd) when transmission failure occurs, which reduces the number of segments transmitted in the next round.

(2) Delay estimation lag: LowRTT records the RTT of each subflow and firstly transmits segments on the subflow with the lowest RTT. However, RTT changes frequently in edge IoT. The RTT estimation lags and cannot reflect the future delay of subflows. Besides, the transmission in edge IoT is usually asymmetric which means the arrival and return delays of segments are quite different. It is not accurate to estimate the segment arrival order based on RTT.

(3) Sender window constraint: MPTCP is an extension of the TCP and inherits the sliding window mechanism. The sender window (swnd) limits the segment amount that one subflow can continuously send in one transmission round (TR, the time between a subflow transmitting all segments in the swnd and receiving the corresponding acknowledgments). In edge IoT, the duration of TR varies among subflows. When the duration gap is huge enough, the subflow with lowest RTT may finish several TRs, while others are still in the first one. Thus, a large number of out-of-order segments will be generated.

To improve the performance of LowRTT, many solutions have been proposed. Hurtig et al. [6] intruded two new scheduling algorithms: the block estimation (BLEST) scheduler which aims to reduce buffer blocking and the shortest transmission time first (STTF) scheduler which tries to minimise the transmission time of each segment. However, these two solutions do not consider the influence of transmission failure. DPSAF [7] considered window changes of the congestion control algorithm and utilized maximum likelihood estimation to estimate the segments on all subflows and determine the scheduling value. Nevertheless, DPSAF still lacks the prediction of the transmission delay. Also, our team previously proposed some solutions in [8–11] which optimize the multipath scheduling separately through path quality estimation, network coding, and cross-layer perception. However, all the above solutions are on the basis of standard protocol and do not break the sender window constraint.

To address the shortages of the above solutions, a fail-aware and delay-predicted multipath virtual queue scheduling (FD-MVQS) is designed in this paper. Specifically, the contributions of this paper can be summarized as follows:

(i) Build a two-plane multipath scheduling framework and break the three limitations of LowRTT through the cooperative scheduling of control and data planes.

(ii) Propose a transmission failure estimation (TFE) model in the control plane, which considers the relative parameters and evaluates the transmission factor.

(iii) Develop a chaos theory-based arrival delay prediction (CTADP) method in the control plane, which extracts the features of the arrival delays and calculates the future values.

(iv) Design a multipath virtual queue scheduling (MVQS) algorithm in the data plane, which breaks the swnd constraint through the virtual queue (VQ) scheme and allocates segments based on their transmission delays.

(v) Test FD-MVQS in different edge IoT scenarios. Simulation results show that compared with other scheduling schemes, FD-MVQS can effectively improve the performance of multipath transmission in terms of throughput, delay, and segment disorder.

The remainder of this paper is organized as follows. Section 2 describes related works. The system design of FD-MVQS is given in Section 3. Section 4 details the TFE, ADP, and MVQS modules of FD-MVQS. Simulation testing

results and discussion are included in Section 5. The conclusions and future work are provided in Section 6.

## 2. Related Works

This section firstly gives a brief introduction to the typical transmission in edge IoT problems and introduces some typical solutions. Then, the multipath scheduling solutions are focused. Finally, the shortages of these solutions are analyzed, and the design principle of FD-MVQS is shortly described.

*2.1. Transmission in Edge IoT.* Benefitting from the advances in edge IoT, the requests of consumers can be processed nearby instead of being transmitted to the remote data centres. Edge IoT can effectively reduce the data processing delay and the transmission pressure of the central network. However, the number of IoT nodes is experiencing a dramatic increase which means massive real-time data. The efficient transmission of these data in heterogeneous and time-varying IoT networks is still challenging.

To improve the transmission efficiency of edge IoT, many attempts have been made. Yi and Cai [12] focused on the delay-constrained transmission in IoT-based healthcare networks and proposed a mechanism in the gateway to meet the priority awareness and the delay constraints of medical packet. Shan et al. [13] considered limited energy resources in wireless IoT and proposed a two-step approach which can minimise the energy consumption of the IoT device under the delay constraint. The authors of [14] designed two adaptive models to solve the optimization of the transmission path and the network topology in the network layer of the IoT system. Li et al. [15] combined the software-defined network (SDN) and edge computing (EC) to improve the performance of data exchange with different delay flows among different smart devices. Suzuki et al. [16] proposed an application-oriented optical transmission control under the SDN/NFV-based edge IoT to enhance computational efficiency. Sodhro et al. [17] aimed to obtain better QoS during multimedia transmission in V2V edge computing platforms and suggested an optimal algorithm based on the analysis of energy efficiency, battery charge consumption, and packet loss. Pace et al. [18] tried to use the artificial intelligence methods to solve the increasing density of transmission in edge IoT and introduced an exemplary case study where machine learning is successfully used to find the delicate balance between the spectrum and the energy efficiency. In [19], the transmission security in IoT was discussed and a chaotic secure communication scheme was detailed based on the synchronization of different-structure fractional-order chaotic systems with different orders.

However, the above solutions are all concentrated on the single-path transmission, and the multipath transmission which has advantages in bandwidth aggregation and robustness has been ignored.

*2.2. Multipath Scheduling Schemes.* In [4], the Internet Engineering Task Force (IETF) standardises a TCP Extensions for Multipath Operation with Multiple Addresses named MPTCP. MPTCP provides the ability to simultaneously use multiple paths between peers by extending the TCP head options. MPTCP provides the same type of service to applications as TCP (i.e., a reliable byte stream) but performs better in throughput and reliability which is more suitable for IoT environments. However, the transmission efficiency of the MPTCP is still limited by the scheduling algorithms.

To optimize the scheduling strategies of the MPTCP, different researchers have given their solutions. The authors of [20] designed a subflow allocation algorithm for the MPTCP in heterogeneous wireless networks to make a tradeoff between energy efficiency and video quality. Hwang et al. [21] proposed a fast coupled retransmission mechanism which can forward the segments from the congested flow onto the noncongested flow and quickly retransmit them and reduce the out-of-order segments. In [22], the authors leveraged the ECN (explicit congestion notification) mechanism to detect shared bottlenecks among subflows and designed a shared bottleneck-based scheduling scheme which can distribute segments according to the window size changes of each subflow and prevent throughput degradation due to out-of-order packets. Pokhrel and Choi [23] focused on the multipath transmission in high packet-loss rate and time-varying wireless networks and used the load balancing and forward error correction (FEC) to solve the out-of-order problem. The authors of [24] identified the limitations of MPTCP schedulers for thin streams and presented a MPTCP scheduling algorithm which can actively probe unused subflows and timely update the one-way delay information.

## 3. System Overview

This section gives a brief description of the proposed FD-MVQS solution. As shown in Figure 2, the sender communicates with the receiver through multiple subflows in the heterogeneous edge wireless networks. FD-MVQS contains three modules: multipath virtual queue scheduling (MVQS), transmission failure estimation (TFE), and chaos theory-based arrival delay prediction (CTADP), which are all marked by rounded rectangles. These modules work on data and control planes, respectively. To distinguish the segment status, we use some squares with different textures and make the solid arrows indicate the data flow, while the dotted arrows indicate the control flow.

In the data plane, the data flow of FD-MVQS is as follows: when getting data from the application layer, the sender will firstly segment them according to the negotiated maximum segment size (MSS). Then, these segments will be assigned a data sequence number (DSN). The numbered segment will be allocated to different subflow virtual queues (VQ) in the MVQS modules according to the time factor $T_i$ transmitted from the control plane. After that, each subflow will independently transmit the segments in its VQ as TCP does. Finally, these segments will arrive at the receiver through different subflows and be delivered to the application layer after the reassembling in the receiver buffer.

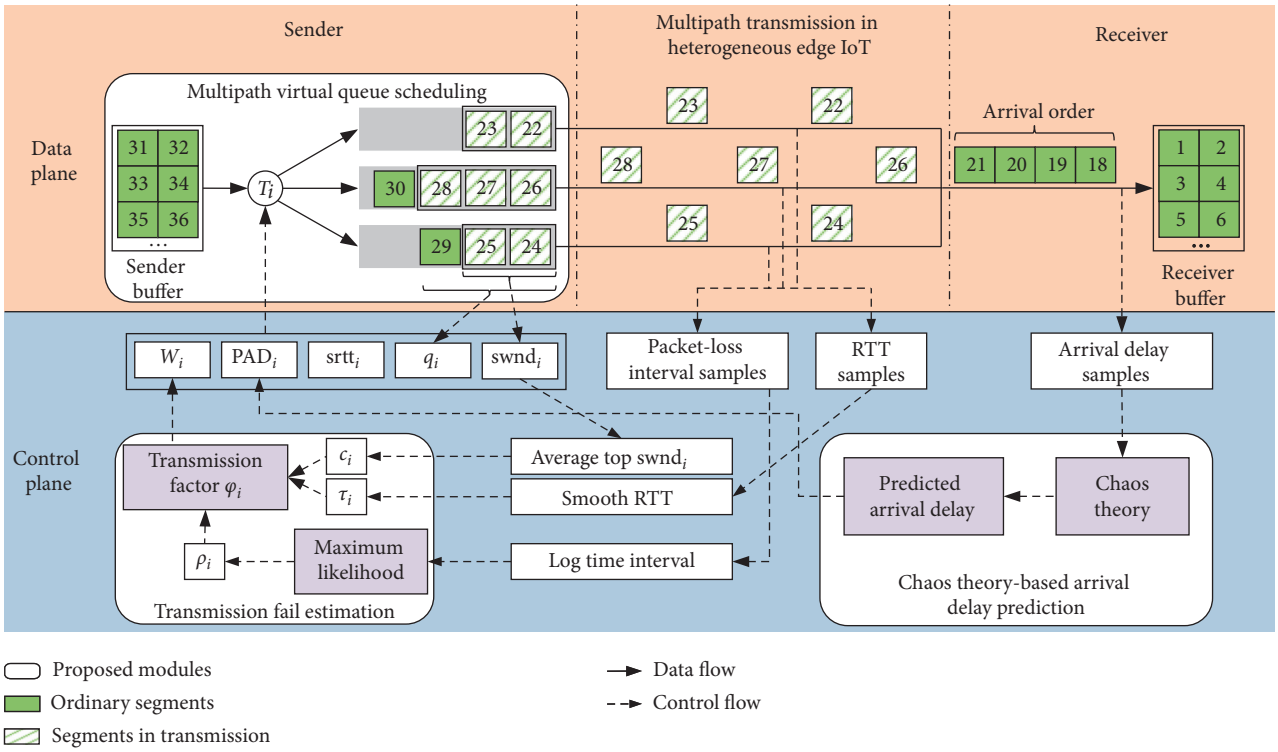


Figure 2: The FD-MVQS system architecture.

In the control plane, the control flow of FD-MVQS is as follows: receiving segments from one subflow every time, the receiver will evaluate the arrival delay (AD) and feed it back along with acknowledgment (ACK) segments. When receiving ACK, the sender will firstly update the round-trip time (RTT) samples. The sender will also update packet-loss interval samples when packet loss occurs. Based on these parameters, the TFE module will calculate the congestion factor $\varphi_i$, and the CTADP module will give the predicted arrival delay $(\text{PAD})_i$. Finally, $\varphi_i$ and $\text{PAD}_i$ will coinstruct the VQ scheduling by influencing the calculation of $T_i$.

Through the cooperation of data and control planes, FD-MVQS achieves a closed-loop control chain which is illustrated in Figure 3. The control loop of FD-MVQS consists of six parts: analysis, action, policy, event, environment, and sense. In the analysis period, the sender will calculate the predicted arrival delay and sense the transmission failure. Then, the results will be given to the action procedure where segments will be scheduled following the virtual queue policy. After that, the event that segments is multipath transmitted through the environment of heterogeneous edge IoT networks. Then, the transmission failure will be reestimated according to the changed subflow status. This control loop improves the adaption of FD-MVQS to dynamic networks.

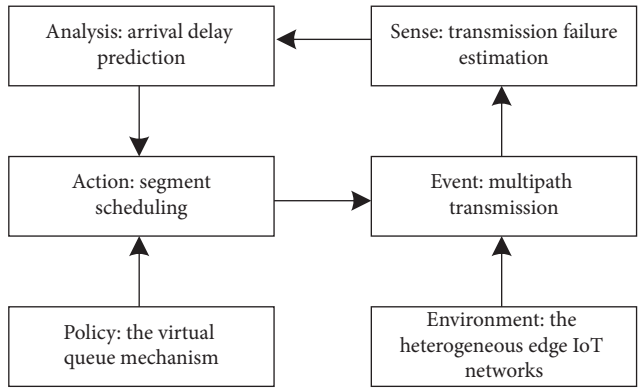


Figure 3: The control loop of FD-MVQS.

## 4. FD-MVQS Design

This section details the proposed FD-MVQS method and, respectively, introduces the TFE, CTADP, and MVQS modules.

*4.1. Transmission Failure Estimation.* In this section, we introduce the transmission failure estimation model. Let $R = \{1, 2, \ldots, r\}$ be the subflow set of a multipath transmission connection. For each $i \in R$, the proposed TFE model is formulated as

$$\varphi_i = \frac{1}{1 + \mu^{\nu - (1-\rho_i)\cdot\sigma_i\cdot c_i}}, \tag{1}$$

where $\varphi_i$ is the transmission factor, $\tau_i$ is the parameter to reflect the features of RTT, and $c_i$ is the parameter to describe the swnd state. $\sigma_i$ and $c_i$ evaluate the transmission state from two different perspectives. $\rho_i$ is the packet-loss rate which reflects the state of networks. The parameters $\mu$

and $\nu$ are constants which determine the offset and sensitivity of the TFE model, respectively.

The RTT parameter $\sigma_i$ in equation (1) is defined as

$$\sigma_i = \frac{\text{srtt}_i}{\text{srtt}_i'}, \tag{2}$$

where $\text{srtt}_i$ and $\text{srtt}_i'$ are the latest and last smooth round-trip time (SRTT), respectively. For each ACK on subflow $i$, srtt can be updated through

$$\text{srtt}_i = (1-\gamma)\cdot \text{srtt}_i' + \gamma \cdot \text{rtt}_i, \tag{3}$$

where $\text{rtt}_i$ is the newly observed RTT of subflow $i$. With the help of the TCP timestamp option standardized in RFC7323 [25], the sender can get a new RTT sample of subflow $i$ when receiving new ACK from it. $\gamma$ is the smooth factor which determines the weight of new samples.

The swnd parameter $c_i$ in equation (1) is defined as

$$c_i = \frac{\text{swnd}_i}{\overline{\text{swnd}_i^{\text{top}}}}, \tag{4}$$

where $\text{swnd}_i$ is the sender window of subflow $i$. According to RFC5681 [26], we have $\text{swnd}_i = \min[\text{rwnd}_i, \text{cwnd}_i]$. $\text{rwnd}_i$ and $\text{cwnd}_i$ in the formulation denote the receiver window and congestion window of subflow $i$, respectively. Due to the congestion control algorithms and receiver buffer changes, $\text{swnd}_i$ will experience many fluctuations during the lifetime of subflow $i$. $\overline{\text{swnd}_i^{\text{top}}}$ is the average of top values that $\text{swnd}_i$ has reached before dropping which can reflect the maximum capacity of the current subflow and avoid the error caused by random events. Additionally, to facilitate the calculation, we use MSS (maximum segment size) as the measure unit of cwnd, rwnd, and swnd and adopt the following equation to calculate the average value:

$$\overline{A_j} = \frac{\overline{A_{j-1}}\cdot (j-1) + A_j}{j}, \tag{5}$$

where $j$ is the number of samples and $A_j$ is the result of the $j$th sampling. $\overline{A_j}$ and $\overline{A_{j-1}}$ are the average values after $j$th and $j-1$ sampling, respectively. This recursive formula can save the storage resources and reflect the changes in real time.

The packet-loss rate $\rho_i$ in equation (1) can be expressed as

$$\rho_i = \rho_{\text{ic}} + \rho_{\text{ie}}, \tag{6}$$

where $\rho_{\text{ic}}$ is the packet-loss rate caused by congestion. When queuing segments in the bottleneck route exceed the threshold, random early detection (RED) [27] will drop the incoming segments randomly. $\rho_{\text{ie}}$ is the packet-loss rate caused by the link error or terminal mobility which can be seen as a random variable that follows normal distribution. From equation (6), we can know that the packet-loss rate $\rho_i$ changes with time. In order to calculate $\rho_i$, the maximum likelihood estimation method was adopted. In the transmission, the interval between every two packet-losses of subflow $i$ can be sampled as $D_i = \{d_i^1, d_i^2, \ldots, d_i^s\}$. $s$ is the sample size. The conditional probability of the $k$th sample can be formulated as

$$p\left(d_i^k|\rho_i\right) = \left(1-\rho_i\right)^{N_i}\cdot \rho_i, \tag{7}$$

where $N_i$ is the number of segments successfully transmitted through subflow $i$ during $d_k$. $N_i$ can be estimated by

$$N_i = \frac{d_i^k}{\text{srtt}_i}\cdot \text{swnd}_i. \tag{8}$$

Based on the packet-loss interval set $D$, the likelihood function of $\rho_i$ can be formulated as

$$L(\rho_i) = p(D|\rho_i) = p\left(d_i^1, d_i^2, \ldots, d_i^s|\rho_i\right) = \prod_{k=1}^{s} p\left(d_i^k|\rho_i\right). \tag{9}$$

Thus, the maximum likelihood estimator of $\rho_i$ can be expressed as

$$\widehat{\rho}_i = \arg\max_{\rho_i}\left(\prod_{k=1}^{s} p\left(d_i^k|\rho_i\right)\right). \tag{10}$$

To facilitate the analysis, the logarithmic likelihood function is defined:

$$H\left(\widehat{\rho}_i\right) = \ln\left(L\left(\rho_i\right)\right) = \ln\left(\prod_{k=1}^{s} p\left(d_i^k|\rho_i\right)\right) = \sum_{k=1}^{s} \ln\left(p\left(d_i^k|\rho_i\right)\right). \tag{11}$$

According to equation (11), $\widehat{\rho}_i$ can be calculated through

$$\widehat{\rho}_i = \arg\max_{\rho_i}\left(\sum_{k=1}^{s} \ln\left(p\left(d_i^k|\rho_i\right)\right)\right). \tag{12}$$

Summarily, for each subflow $i$, the packet-loss rate can be estimated as

$$\rho_i = \begin{cases} 0, & s = 0, \\ \widehat{\rho}_i, & s \geq 1. \end{cases} \tag{13}$$

Algorithm 1 gives the specific process of transmission factor estimation. During the transmission, the TFE model updates the transmission factor $\varphi$ of each subflow. When different signals are detected, the parameters will be updated which leads to the changes in $\varphi$. The multipath virtual queue scheduling algorithm can inquire the value of $\varphi$ when necessary.

*4.2. Chaos Theory-Based Arrival Delay Prediction.* In the MPTCP, the standard RTT calculated by the sender includes three main parts: the time between the segments leaving the sender and arriving at the receiver (AD), the time that the receiver processes these segments (processing delay, PD), and the time that the ACKs of these segments return from the receiver to the sender (returning delay, RD). Traditional scheduling algorithm uses the RTT to evaluate the transmission speed of different subflows and select the fastest one to transmit segments. However, this means usually

(1) **Initialize**: subflow set $R$, packet-loss interval set $D = \varnothing$, the packet-loss time $t = 0$, the last packet-loss time $t' = 0$.
(2) **while** transmission not end **do**
(3) **for** each $i \in R$ **do**
(4) **if** $\text{swnd}_i$ dropping **then**
(5) Get new $\text{swnd}_i^{\text{top}} = \text{swnd}_i$;
(6) Update $\overline{\text{swnd}_i^{\text{top}}}$ according to equation (5)
(7) **if** packet-loss occurs in $\text{subflow}_i$ **then**
(8) **if** $D == \varnothing$ **then**
(9) Get the packet-loss time $t$
(10) Add $t$ to $D$;
(11) else
(12) $t' \leftarrow t$;
(13) Get the packet-loss time $t$;
(14) Calculate $d = t - t'$;
(15) Add $d$ to $D$
(16) **if** receive new ACK on subflow $i$ **then**
(17) Get new $\text{rtt}_i$, $\text{swnd}_i$;
(18) $\text{srrt}_i' \leftarrow \text{srtt}_i$;
(19) Updated $\text{srtt}_i$ according to equation (3)
(20) Calculate $\sigma_i$ according to equation (2);
(21) Calculate $c_i$ according to equation (4);
(22) Calculate $\widehat{\rho}_i$ according to equation (12);
(23) Update $\varphi_i$ according to equation (1);

Algorithm 1: Transmission factor estimation.

engenders a great error because of the existence of PD and RD. Especially in the dynamic edge IoT networks, the AD and PD are with huge difference due to the changes of the network topology. Therefore, AD is taken as an important parameter to estimate the arrival order of segments. Taking the advantage of the TCP timestamp option, when receiving one segment, the receiver can calculate the AD of subpath $i$ by

$$\text{AD}_i = r_i - s_i, \tag{14}$$

where $r_i$ is the time of receiving this segment and $s_i$ is the segment sending time carried by the TCP timestamp option. It is important to note that $\text{AD}_i$ is the relative time due to the system time asynchrony of the sender and receiver. Since we only compare the ADs of different subflows with each other, the relative time is enough.

In the transmission, AD usually changes due to the congestion and randomness of networks. It is not accurate to estimate the segment arrival order based on the existing AD samples. Thus, the chaos theory is used to predict the AD for each subflow $i$. In the transmission, the AD time series can be obtained: $\text{AD}_i = \{\text{AD}_i^1, \text{AD}_i^2, \ldots, \text{AD}_i^n\}$, $n$ is the sample size. To simplify the calculation, the log arrival delay is employed: $\text{LAD}_i = \{l_i^1, l_i^3, \ldots, l_i^n\}$, where $l_i^n = \ln(\text{AD}_i^n - \min(\text{AD}_i) + 1)$. With the chaos theory, the phase-space reconstruction of time series is as follows:

$$\left[\begin{array}{cccc} \mathbf{Y}_1 & \mathbf{Y}_2 & \ldots & \mathbf{Y}_X \end{array}\right] = \left[\begin{array}{cccc} l_i^1 & l_i^2 & \ldots & l_i^X \\ l_i^{1+\tau} & l_i^{2+\tau} & \ldots & l_i^{X+\tau} \\ l_i^{1+2\tau} & l_i^{2+2\tau} & \ldots & l_i^{X+2\tau} \\ \vdots & \vdots & \vdots & \vdots \\ l_i^{1+(m-1)\tau} & l_i^{2+(m-1)\tau} & \ldots & l_i^{X+(m-1)\tau} \end{array}\right], \tag{15}$$

where $m$ is the embedded dimension, $\tau$ is the coordinate delay time, $\mathbf{Y}$ is the phase point, and $X$ is the number of phase points, which can be calculated by

$$X = n - (m-1)\tau. \tag{16}$$

According to Tackens embedding theorems [28], the $m$-dimensional phase space can be found in the sense of topology invariance if $m \geq 2m' + 1$. $m'$ is the relevance dimension of time series, which can be obtained through the G-P algorithm [29].

Based on the phase-space reconstruction, the weighted first-order local prediction method was used to estimate the next phase point. Let $\mathbf{Y}_i^m$ be the central point and $\mathbf{Y}_i^{ma}$ ($a = 1, 2, \ldots, h$) be the set of adjacent points. $h$ is the number of adjacent points. The distance between $\mathbf{Y}_i^m$ and $\mathbf{Y}_i^{ma}$ is expressed as $\text{dis}_i^a$. $\text{dis}_i^{\min}$ is the minimum value of $\text{dis}_i^a$. The weight of $\mathbf{Y}_i^{ma}$ can be formulated as

$$P_i^a = \frac{\exp\left(-b\left(\text{dis}_i^a - \text{dis}_i^{\min}\right)\right)}{\sum_{a=1}^{h} \exp\left(-b\left(\text{dis}_i^a - \text{dis}_i^{\min}\right)\right)}, \quad (17)$$

where $b$ is the parameter. When $b = 1$, the weighted first-order fitting can be expressed as

$$\mathbf{Y}_i^{ma+1} = \alpha\mathbf{e} + \beta\mathbf{Y}_i^{ma}, \quad a = 1, 2, \ldots, q, \quad (18)$$

where $\mathbf{e} = (1, 1, \ldots, 1)^T$, $\alpha$ and $\beta$ are fit coefficients, and $\mathbf{Y}_i^{ma+1}$ is the predicted value of $\mathbf{Y}_i^{ma}$. According to the least square method, the value of $\alpha$ and $\beta$ can be obtained from

$$\sum_{a=1}^{h} P_i^a \left(\mathbf{Y}_i^{ma+1} - \alpha e + \beta \cdot \mathbf{Y}_i^{ma}\right)^2 = \min. \quad (19)$$

Based on the above analysis, the predicted arrival delay $\text{PAD}_i$ can be obtained by

$$\text{PAD}_i = \text{AD}_i^{l+1} = \mathbf{Y}_i^{X+1}(m). \quad (20)$$

*4.3. Multipath Virtual Queue Scheduling.* In this section, the multipath virtual queue scheduling (MVQS) is detailed which can effectively ensure that segments reach the receiver in order based on the failure estimation and delay prediction proposed in Sections 4.1 and 4.2.

Figure 4 illustrates the proposed virtual queue (VQ) scheme. In a multipath connection, subflows transmit segments concurrently. For each subflow in one multipath connection, a VQ is maintained. The VQ length of subflow $i$ is $q_i$. The segments in VQ can be sent sequentially in one or more rounds. In each transmission round, the number of segments will be sent is $\text{swnd}_i$. If no packet loss occurs, the segments transmitted in one round will reach the receiver together. $\text{swnd}_i$ usually changes with the transmission round due to the congestion control algorithm. For example, $\text{swnd}_i$ will be larger if all segments in the last round are transmitted successfully or be smaller if packet loss occurs. Thus, the transmission failure possibility must be considered when constructing the VQ. The TFE model proposed in Section 4.1 can help to determine the length of VQ. Besides, the segments in VQ will only be sent when the segments of front rounds are all ACKed. The PAD estimated in Section 4.2 can be used as an important parameter to determine the segment order in VQ.

To sum up, the time factor $T_i$ is denoted to predict the transmission delay of different subflows (the delay before new segments reach the receiver). $T_i$ is related to VQ length $q_i$. When $q_i < \text{swnd}_i$, the new segments can be transmitted immediately. Thus, $T_i = \text{PAD}_i$. When $q_i \geq \text{swnd}_i$, $T_i$ can be calculated by

$$T_i = \text{PAD}_i + \left(1 + \frac{q_i - \text{swnd}_i}{W_i}\right) \cdot \text{srtt}_i, \quad (21)$$

where $W_i$ is the window factor that assesses the segments sent in one round. We make

$$W_i = (1 - \varphi_i) \cdot \text{swnd}_i + \eta \cdot \varphi_i \cdot \text{swnd}_i. \quad (22)$$

Parameter $\eta$ is the decrease factor of multipath congestion control algorithms when congestion is found and is 0.5 for the standard MPTCP.

In summary, $T_i$ of $\text{VSQ}_i$ is represented as

$$T_i = \begin{cases} \text{PAD}_i & \text{if } q_i < \text{swnd}_i, \\ \text{PAD}_i + \left(1 + \dfrac{q_i - \text{swnd}_i}{W_i}\right) \cdot \text{srtt}_i & \text{if } q_i \geq \text{swnd}_i. \end{cases} \quad (23)$$

During the multipath transmission, the MVQS maintains a VQ for each subflow. If there are unallocated segments, MVQS will cyclically calculate the transmission delay $T$ of each VQ and allocate segments to the VQ with minimum $T$. When detecting new ACK or packet loss, the parameters need to be updated which will lead to changes in $T$. More details of MSAS are shown in Algorithm 2.

After the allocation, subflows will transmit the segments in their VQ as TCP does. Thanks to the transmission failure estimation and delay prediction, the idle bandwidth of different subflows will be better aggregated, and the out-of-order problem will be effectively suppressed.

## 5. Performance Evaluation

This section evaluates the throughput, RTT, and segment disorder performances of FD-MVQS by comparing it with LowRTT [6] and DPSAF [7] in different scenarios.

*5.1. Simulation Setup.* Testing has been carried out using Network Simulator version 3 which installed the MPTCP implementation published by Kashif [30]. The experiment topology considers a heterogeneous edge IoT network environment which is illustrated in Figure 5. A mobile consumer requests multimedia data from an edge server via three subpaths. To imitate the performance differences among network interfaces in the real environment, different parameters are set for these three paths, as shown in Table 1.

Besides, the variable bit rate (VBR) with Pareto distribution was also been leveraged to imitate background flow in real transmission. The VBR generates the cross-traffic which flows through the three subpaths to arrive at the sink. 90% of the cross-packets are carried by the TCP, while the rest 10% are carried by UDP. The packet sizes are chosen as follows: 49% are 44 bytes, 1.2% are 576 bytes, 2.1% are 628 bytes, 1.7% are 1300 bytes, and 46% are 1500 bytes according to the results of the VBR. The proportion of cross-traffic for each subpath varies from 0 to 50%.

Based on the results of pretesting, the other parameter settings of FD-MVQS are as follows: in the transmission failure estimation model, $\mu = 10$, $\nu = 0.8$, and $\lambda = 0.125$.

*5.2. Simulation Results.* The simulation results of fixed and dynamic parameters are shown separately.

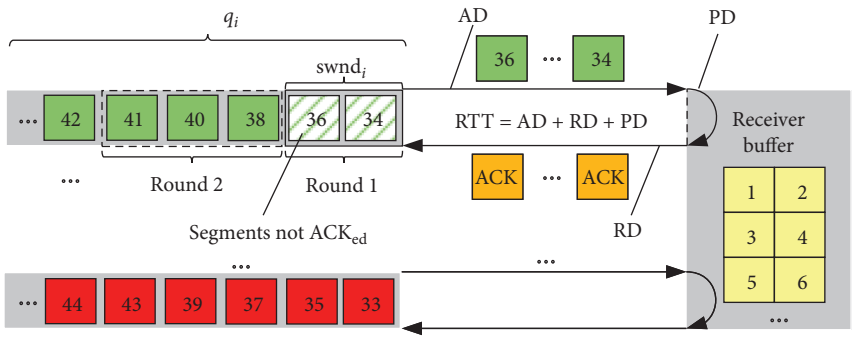


Figure 4: The schematic of virtual queue.

(1) **Initialize:** subflow set $R$, transmission delay $T$, virtual wueue VQ.
(2) **while** there are unallocated segments **do**
(3) **if** receive new ACK from subslow $i$ **then**
(4) Get new swnd$_i$, rtt$_i$
(5) Update srtt$_i$ according to equation (3)
(6) Update $\varphi_i$ with the feedback of Algorithm 1
(7) Update PAD$_i$ according to equation (20)
(8) Update $W_i$ according to equation (22)
(9) **if** packet loss occurs in subflow $i$ **then**
(10) Get new swnd$_i$;
(11) Update $\varphi_i$ with the feedback of Algorithm 1
(12) Update $W_i$ according to equation (22)
(13) **for** each $i \in R$ **do**
(14) Calculate $T_i$ according to equation (23)
(15) Find the subflow $i$ with minimum $T$
(16) **if** $q_i <$ swnd$_i$ **then**
(17) **if** undistributed segments <swnd$_i - q_i$ **then**
(18) Allocate all undistributed segment to VQ$_i$
(19) else
(20) Allocate swnd$_i - q_i$ segments to VQ$_i$
(21) Update $q_i$
(22) else
(23) **if** undistributed segments <swnd$_i$ **then**
(24) Allocate all undistributed segment to VQ$_i$
(25) else
(26) Allocate swnd$_i$ segments to VQ$_i$
(27) Update $q_i$

Algorithm 2: Multipath virtual queue scheduling.

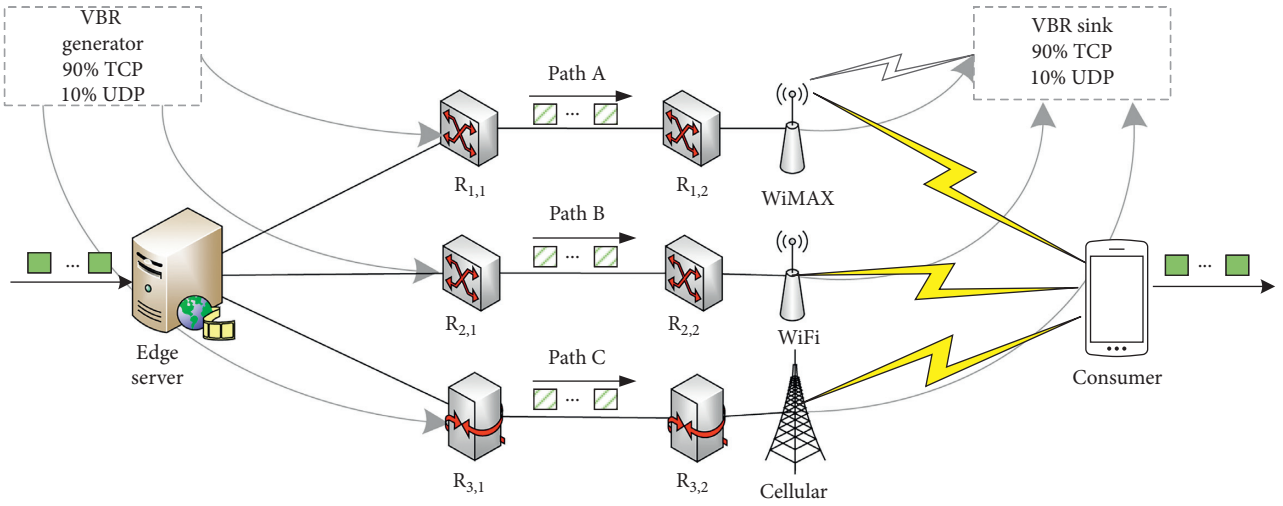


Figure 5: The FD-MVQS system architecture.

Table 1: Parameters in simulation.

| Parameters | Path A | Path B | Path C |
|---|---|---|---|
| Access technology | WiMAX | WiFi | Cellular |
| Access bandwidth | 40 Mbps | 40 Mbps | 80 Mbps |
| Default access delay | 20 ms | 20 ms | 20 ms |
| Default server delay | 10 ms | 10 ms | 10 ms |
| Edge delay | 40 ms | 60 ms | 100 ms |
| Loss rate | 0.1-0.2 | 0.1 | 0.05 |
| Link bandwidth | 100 Mbps | 100 Mbps | 100 Mbps |

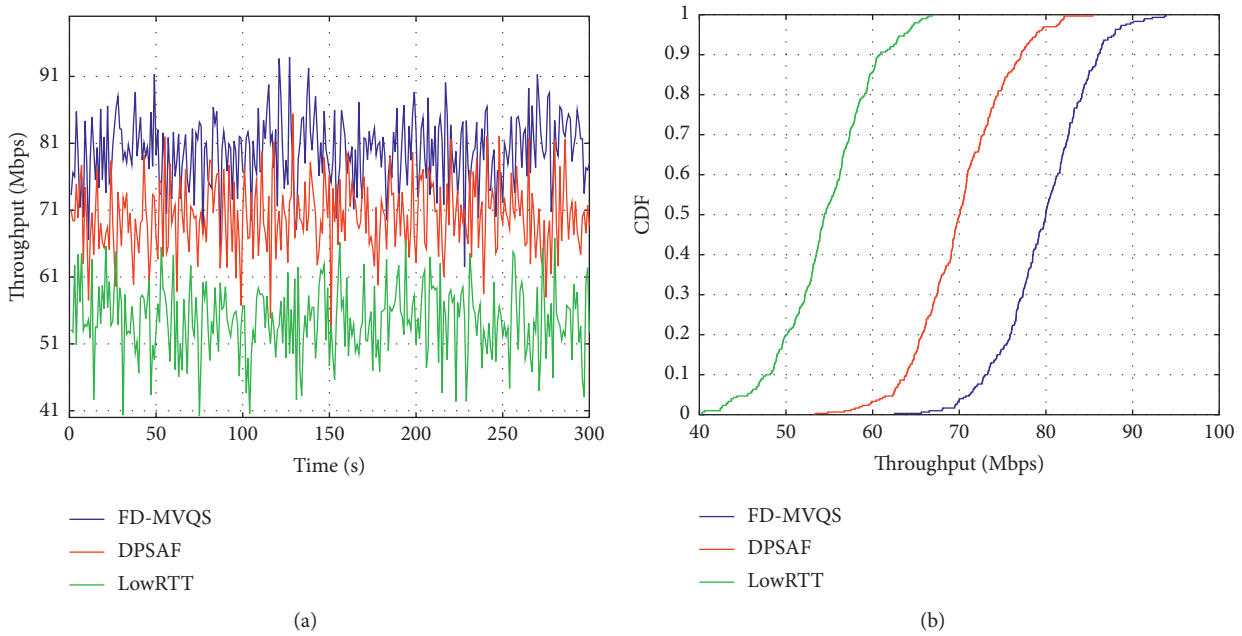


Figure 6: Comparison of throughput. (a) The throughput fluctuation. (b) The CDF of throughput.

*5.2.1. Performance in Fixed Parameters.* The network parameters were firstly fixed by setting the receiver buffer as 128 kB and WiMAX loss rate as 0.1.

Figure 6 shows the throughput comparison of FD-MVQS, DPSAF, and LowRTT. Figure 6(a) displays the throughput tested in the receiver. From the figure, we can know that the throughput fluctuates over time due to congestion control algorithms and packet loss. However, the throughput of FD-MVQS is higher than DPSAF most of the time and higher than LowRTT all the time. To quantify the difference, the CDFs of throughput are illustrated in Figure 6(b). We can see that about 50% sampled throughput are over 80 Mbps, while it is only about 5% for DPSAF. All throughput samples of LowRTT are below 68 Mbps. This is owing to that the FD-MVQS estimates the transmission failure. Based on the estimation results, the server can transfer traffic to subflows with better performance more quickly and reduce the failure possibility. Besides, the arrival delay prediction can effectively help the scheduler to inhibit the disorder problem. The continuous ACKs from the receiver make the swnd grow faster.

Figure 7 shows the RTT comparison of FD-MVQS, DPSAF, and LowRTT. Figure 7(a) displays the RTT samples calculated by the sender when receiving ACKs. From the figure, we can know that the fluctuation of RTT is drastic during the transmission which is owing to the change in the queuing delay of different routing nodes and retransmission. The CDFs of RTT are illustrated in Figure 7(b) to quantify the difference. We can see that about 80% RTT samples of FD-MVQS are below 190 ms, while the DPSAF is about 46%, and LowRTT is about 25%. This is the result of virtual queue scheduling which can break the swnd constraint and allocate segments to the subflow with the shortest transmission time. These segments can sequentially arrive at the consumer and be acknowledged more quickly which reduces the sampled RTTs.

Figure 8 shows the comparison of disordered segments which are calculated by the receiver. From the figure, we can know that the disordered segments of FD-MVQS are significantly less than DPSAF and LowRTT. Benefitting of the virtual queue scheduling, the number of disordered segments of FD-MVQS is mainly under 10 during the transmission. The DPSAF samples are mainly under 24, and LowRTT even has samples higher than 100. FD-MVQS can decrease the disordered segments by about 60 percent compared with DPSAF and about 90 percent compared with LowRTT. The smaller the number of disordered segments is, the faster the arrived segments will be delivered to the

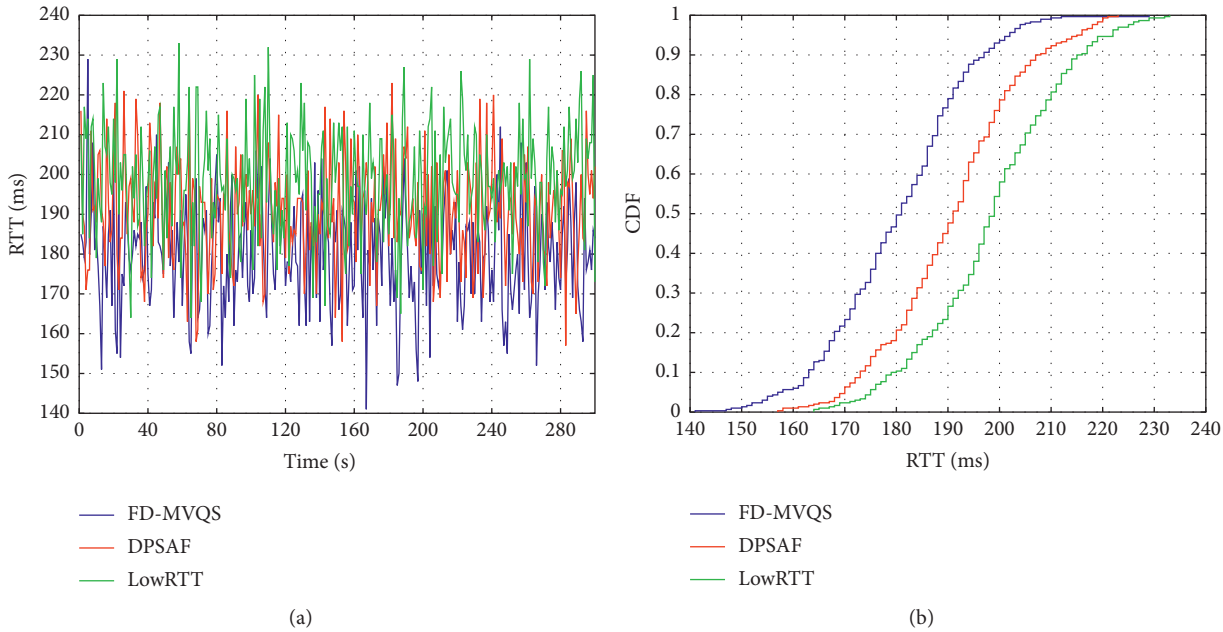


Figure 7: Comparison of RTT. (a) The RTT fluctuation. (b) The CDF of RTT.

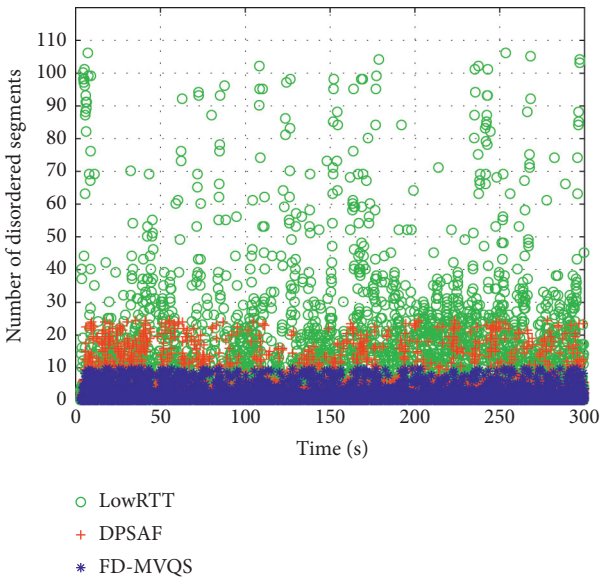


Figure 8: Comparison of disordered segments.

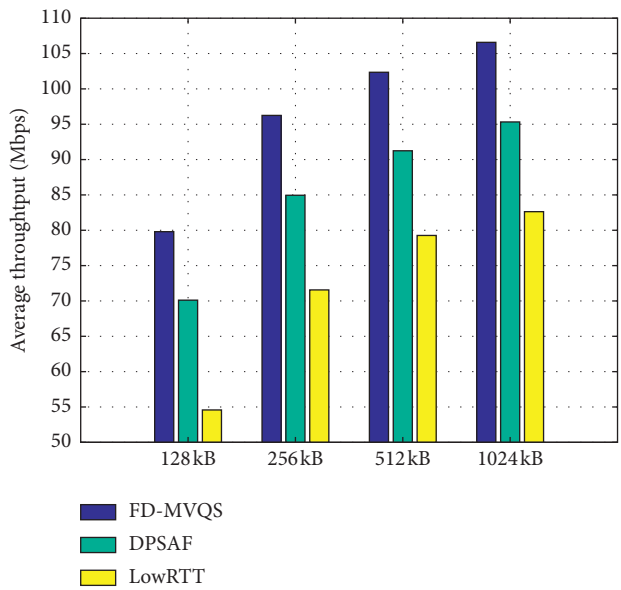


Figure 9: Average throughputs in different receiver buffers.

application layer which can effectively improve the consumer experience.

*5.2.2. Performance in Dynamic Parameters.* To test the performance of FD-MVQS in different scenarios, we change the receiver buffer and loss rate of the WiMAX path and compare the average throughput, RTT, and retransmission rate.

Figure 9 shows the average throughput comparison of FD-MVQS, DPSAF, and LowRTT in different receiver buffers. From the figure, we can see that the average throughput of all three solutions increases with the receiver buffer. There are two main reasons: (1) a small receiver buffer will limit the growth of the sender window; (2) when the receiver buffer is depleted by the out-of-order segments, these segments will be dropped and must be retransmitted by the server. Besides, we can also know that the average throughput of FD-MVQS is higher than the other two solutions, especially in smaller buffer size. When the buffer size is 128 kB, the

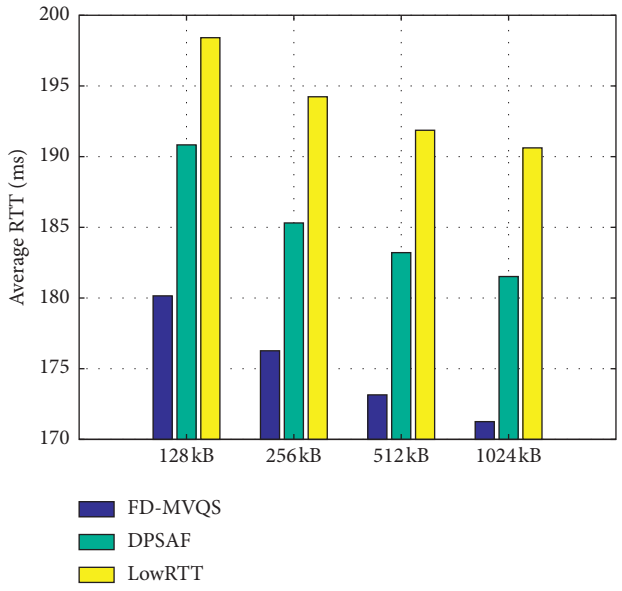


Figure 10: Average RTTs in different receiver buffers.

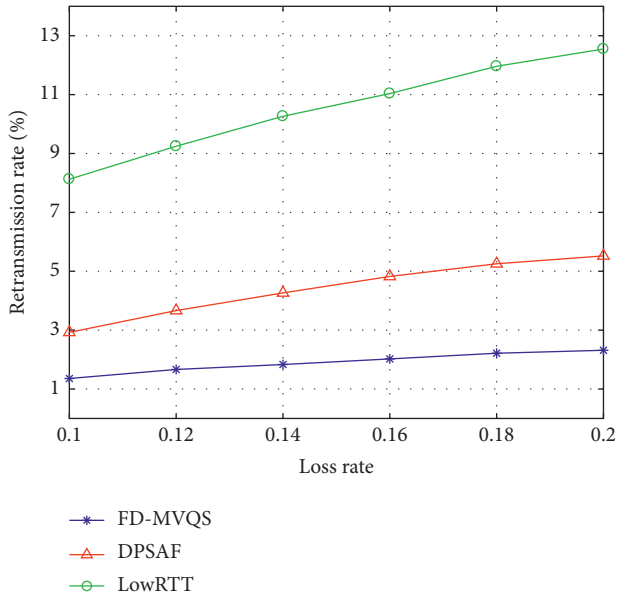


Figure 11: Retransmission rates for different loss rates.

throughput of FD-MVQS, DPSAF, and LowRTT is separately about 80 Mbps, 70 Mbps, and 55 Mbps. FD-MVQS improves throughput by about 14 percent compared with DPSAF and about 45 percent compared with LowRTT.

Figure 10 shows the average RTT comparison of FD-MVQS, DPSAF, and LowRTT in different receiver buffers. From the figure, we can see that the average RTT of all three solutions decreases with the receiver buffer. This is because that the higher buffer can reduce the number of segments dropped by the receiver and avoid the transmission time introduced by retransmission. Besides, we can also see that the average RTT of FD-MVQS is lower than DPSAF and LowRTT in different buffer sizes. Compared with DPSAF, FD-MVQS can separately decrease the average RTT by about 5.59%, 4.88%, 5.49%, and 5.65%. Compared with LowRTT, FD-MVQS can separately decrease the average RTT by about 9.21%, 9.25%, 9.75%, and 10.16%.

To simulate the changeable networks in edge IoT, we tried to improve the loss rate of the WiMAX path from 0.1 to 0.2. The results are shown in Figure 11. The retransmission rates are calculated by $num_{re}/num_{all}$, where $num_{re}$ is the number of segments being retransmitted and $num_{all}$ is the total number of segments being transmitted. From the figure, we can know that FD-MVQS has a lower retransmission rate in different loss rates. Compared with DPSAF, FD-MVQS can separately decrease the retransmission rate by about 53.61%, 54.58%, 57.05%, 58.07%, 57.84%, and 58.07%. Compared with LowRTT, FD-MVQS can separately decrease the retransmission rate by about 83.31%, 82.01%, 82.16%, 81.68%, 81.48%, and 81.54%. This is because that FD-MVQS can transfer segments to a better path more quickly through transmission failure estimation and avoid disordered segments by arrival delay prediction. Both of these can decrease the retransmission rate.

## 6. Conclusion and Future Work

This paper proposes a failure-aware and delay-predicted multipath virtual queue scheduling (FD-MVQS) for multimedia transmission in edge IoT. In the control plane, the proposed transmission failure estimation model and chaos theory-based arrival delay prediction method can give an evaluation to the multiple subflows. Based on this, the multipath virtual queue scheduling algorithm can effectively allocate the segment with the transmission time in the data plane. The simulation-based testing results show FD-MVQS performs better than two alternative solutions in terms of throughput, RTT, and disordered segments in different scenarios. Since this paper concentrates only on the multipath scheduling, future work will study the effect of diverse congestion control strategies in conjunction with FD-MVQS.

## Data Availability

The tests were carried out on Network Simulator version 3 (NS 3).

## Conflicts of Interest

The authors declare that they have no conflicts of interest.

## Acknowledgments

This work was partially supported by the National Natural Science Foundation of China (NSFC) under Grant nos. 61871048 and 61872253.

## References

[1] W. Rafique, L. Qi, I. Yaqoob, M. Imran, R. U. Rasool, and W. Dou, "Complementing IOT services through software defined networking and edge computing: a comprehensive

survey," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 3, pp. 1761–1804, 2020.

[2] Y. Liu, M. Peng, G. Shou, Y. Chen, and S. Chen, "Toward edge intelligence: multiaccess edge computing for 5G and internet of things," *IEEE Internet of Things Journal*, vol. 7, no. 8, pp. 6722–6747, 2020.

[3] W. Na, S. Jang, Y. Lee, L. Park, N.-N. Dao, and S. Cho, "Frequency resource allocation and interference management in mobile edge computing for an internet of things system," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4910–4920, 2019.

[4] A. Ford, C. Raiciu, M. J. Handley, O. Bonaventure, and C. Paasch, "TCP extensions for multipath operation with multiple addresses, RFC 8684," 2020, https://rfc-editor.org/rfc/rfc8684.txt.

[5] C. Paasch, S. Ferlin, O. Alay, and O. Bonaventure, "Experimental evaluation of multipath TCP schedulers," in *Proceedings of the 2014 ACM SIGCOMM Workshop on Capacity Sharing Workshop, CSWS'14*, pp. 27–32, Association for Computing Machinery, New York, NY, USA, August 2014.

[6] P. Hurtig, K.-J. Grinnemo, A. Brunstrom, S. Ferlin, O. Alay, and N. Kuhn, "Low-latency scheduling in MPTCP," *IEEE/ACM Transactions on Networking*, vol. 27, no. 1, pp. 302–315, 2019.

[7] K. Xue, J. Han, D. Ni et al., "DPSAF: forward prediction based dynamic packet scheduling and adjusting with feedback for multipath TCP in lossy heterogeneous networks," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 2, pp. 1521–1534, 2018.

[8] C. Xu, T. Liu, J. Guan, H. Zhang, and G.-M. Muntean, "CMT-QA: quality-aware adaptive concurrent multipath data transfer in heterogeneous wireless networks," *IEEE Transactions on Mobile Computing*, vol. 12, no. 11, pp. 2193–2205, 2013.

[9] C. Xu, Z. Li, L. Zhong, H. Zhang, and G.-M. Muntean, "CMT-NC: improving the concurrent multipath transfer performance using network coding in wireless networks," *IEEE Transactions on Vehicular Technology*, vol. 65, no. 3, pp. 1735–1751, 2016.

[10] C. Xu, P. Wang, C. Xiong, X. Wei, and G.-M. Muntean, "Pipeline network coding-based multipath data transfer in heterogeneous wireless networks," *IEEE Transactions on Broadcasting*, vol. 63, no. 2, pp. 376–390, 2017.

[11] C. Xu, Z. Li, J. Li, H. Zhang, and G. Muntean, "Cross-layer fairness-driven concurrent multipath video delivery over heterogeneous wireless networks," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 25, no. 7, pp. 1175–1189, 2015.

[12] C. Yi and J. Cai, "A truthful mechanism for scheduling delay-constrained wireless transmissions in IOT-based healthcare networks," *IEEE Transactions on Wireless Communications*, vol. 18, no. 2, pp. 912–925, 2019.

[13] F. Shan, J. Luo, J. Jin, and W. Wu, "Offloading delay constrained transparent computing tasks with energy-efficient transmission power scheduling in wireless iot environment," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4411–4422, 2019.

[14] G.-J. Jong, Z.-H. Wang, K.-S. Hsieh, and G.-J. Horng, "A novel adaptive optimization of intragrated network topology and transmission path for iot system," *IEEE Sensors Journal*, vol. 19, no. 15, pp. 6452–6459, 2019.

[15] X. Li, D. Li, J. Wan, C. Liu, and M. Imran, "Adaptive transmission optimization in SDN-based industrial internet of things with edge computing," *IEEE Internet of Things Journal*, vol. 5, no. 3, pp. 1351–1360, 2018.

[16] T. Suzuki, S. Kim, Y. Koyasako, J. Kani, and J. Terada, "Application-oriented optical transmission control for computationally efficient edge computing," in *Proceedings of the 2020 IEEE 17th Annual Consumer Communications Networking Conference (CCNC)*, pp. 1-2, Las Vegas, NV, USA, January 2020.

[17] A. H. Sodhro, M. S. Obaidat, Q. H. Abbasi et al., "Quality of service optimization in an IOT-driven intelligent transportation system," *IEEE Wireless Communications*, vol. 26, no. 6, pp. 10–17, 2019.

[18] P. Pace, G. Fortino, Y. Zhang, and A. Liotta, "Intelligence at the edge of complex networks: the case of cognitive transmission power control," *IEEE Wireless Communications*, vol. 26, no. 3, pp. 97–103, 2019.

[19] H. Hui, C. Zhou, S. Xu, and F. Lin, "A novel secure data transmission scheme in industrial internet of things," *China Communications*, vol. 17, no. 1, pp. 73–88, 2020.

[20] J. Wu, R. Tan, and M. Wang, "Energy-efficient multipath TCP for quality-guaranteed video over heterogeneous wireless networks," *IEEE Transactions on Multimedia*, vol. 21, no. 6, pp. 1593–1608, 2019.

[21] J. Hwang, A. Walid, and J. Yoo, "Fast coupled retransmission for multipath TCP in data center networks," *IEEE Systems Journal*, vol. 12, no. 1, pp. 1056–1059, 2018.

[22] W. Wei, K. Xue, J. Han, D. S. L. Wei, and P. Hong, "Shared bottleneck-based congestion control and packet scheduling for multipath TCP," *IEEE/ACM Transactions on Networking*, vol. 28, no. 2, pp. 653–666, 2020.

[23] S. R. Pokhrel and J. Choi, "Low-delay scheduling for internet of vehicles: load-balanced multipath communication with FEC," *IEEE Transactions on Communications*, vol. 67, no. 12, pp. 8489–8501, 2019.

[24] A. Froemmgen, J. Heuschkel, and B. Koldehofe, "Multipath TCP scheduling for thin streams: active probing and one-way delay-awareness," in *Proceedings of the 2018 IEEE International Conference on Communications (ICC)*, pp. 1–7, Kansas City, MO, USA, May 2018.

[25] D. Borman, R. T. Braden, V. Jacobson, and R. Scheffenegger, "TCP extensions for high performance," 2014, https://rfc-editor.org/rfc/rfc7323.txt.

[26] E. Blanton, D. V. Paxson, and M. Allman, "TCP congestion control, RFC 5681," 2009, https://rfc-editor.org/rfc/rfc5681.txt.

[27] S. Floyd and V. Jacobson, "Random early detection gateways for congestion avoidance," *IEEE/ACM Transactions on Networking*, vol. 1, no. 4, pp. 397–413, 1993.

[28] F. Takens, "Detecting strange attractors in turbulence," in *Dynamical Systems and Turbulence, Warwick 1980*, D. Rand and L.-S. Young, Eds., pp. 366–381, Springer Berlin Heidelberg, Berlin, Germany, 1981.

[29] P. Grassberger and I. Procaccia, "Measuring the strangeness of strange attractors," *Physica D: Nonlinear Phenomena*, vol. 9, no. 1-2, pp. 189–208, 1983.

[30] N. Kashif, "Mptcp implemention in Ns3," 2019, https://github.com/Kashif-Nadeem/ns-3-dev-git.