VSiM: Improving QoE Fairness for Video Streaming in Mobile Environments

Yali Yuan*¶, Weijun Wang‡¶, Yuhan Wang*, Sripriya S. Adhatarao*, Bangbang Ren*, Kai Zheng§, Xiaoming Fu*
*University of Göttingen, Germany. Emails: {yali.yuan, adhatarao, fu}@cs.uni-goettingen.de, wangyuhan601@gmail.com
‡University of Göttingen & Nanjing University. Email: weijunwang@smail.nju.edu.cn
*University of Göttingen and National University of Defense Technology. Email: renbangbang11@nudt.edu.cn
§Huawei Technologies Co., China. Email: kai.zheng@huawei.com

***Abstract*—The rapid growth of mobile video traffic and user demand poses a more stringent requirement for efficient bandwidth allocation in mobile networks where multiple users may share a bottleneck link. This provides content providers an opportunity to optimize multiple users' experiences jointly, but users often suffer short connection durations and frequent handoffs because of their high mobility. This paper proposes an end-to-end scheme, VSiM, to support mobile video streaming applications in heterogeneous wireless networks. The key idea is allocating bottleneck bandwidth among multiple users based on their mobility profiles and Quality of Experience (QoE)-related knowledge to achieve max-min QoE fairness. Besides, the QoE of buffer-sensitive clients is further improved by the novel server push strategy based on HTTP/3 protocol without affecting the existing bandwidth allocation approach or sacrificing other clients' view quality. We evaluated VSiM experimentally in both simulations and a lab testbed on top of the HTTP/3 protocol. We find that the clients' QoE fairness of VSiM achieves more than 40% improvement compared with state-of-the-art solutions, *i.e.*, the viewing quality of clients in VSiM can be improved from 720p to 1080p in resolution. Meanwhile, VSiM provides about 20% improvement on average of the averaged QoE.**

I. Introduction

The rapid growth of mobile video traffic and user demand leads to a higher probability for multiple video-streaming clients sharing a bottleneck link. The experience of multiple users can be significantly affected by the network conditions and users' high mobility, such as high fluctuation in the available bandwidth and high moving speed of clients, when multiple clients simultaneously compete for the shared bottleneck link in mobile video streaming applications [1].

This problem becomes more severe in 5G networks, where clients are high mobility. The base station (BS) is typically smaller, and the directional antenna is often employed to prevent severe propagation loss and ensure a good transmission rate [2]. Because of the directional antenna, video content can only be transmitted when the antennas of both the base station and mobile user are directed towards each other. In this case, handoffs [1] occur more frequently due to the small cell region (*e.g.*, picocells with range under 100 meters [3]) and clients' high mobility characteristics. Frequent handoffs may cause rebuffering, which will diminish the QoE significantly. Besides, some clients may obtain lower QoE, resulting in QoE unfairness. However, QoE and QoE fairness are two key metrics to evaluate the performance of video traffic for clients. QoE is an essential aspect in keeping a single customer's satisfaction in isolation, while QoE fairness is especially of importance for video content providers where operators want to keep all users sufficiently satisfied (*e.g.*, high QoE) in a fair manner. Therefore, optimization models are needed to achieve the maximum (or at least ensure an acceptable) QoE while ensuring fairness among users for mobile video streaming applications in terms of resource (*e.g.*, bandwidth) allocation in shared bandwidth links.

VSiM. Based on these observations, we design VSiM, an easy-deployment and high-compatibility end-to-end solution to the QoE fairness problem in mobile video streaming applications with a shared bottleneck bandwidth. In particular, clients of VSiM inherit the excellent performance of Dynamic Adaptive Streaming over HTTP (DASH) protocol; VSiM deployed on the server achieves the QoE fairness by allocating bandwidth based on the advantages of the HTTP/3 protocol. To achieve this goal, we face several challenges:

Challenge 1: How to profile the mobility impact and use the profile to maximize clients' QoE fairness in a mobile network? Most of the existing works [2], [4]–[10] depend on off-the-shelf mechanisms to ensure the network performance, like QoE, by dividing bandwidth evenly among multiple clients' connections, which neglects the knowledge from clients. Clients with the same bandwidth may experience different viewing experiences in mobile video streaming applications because of clients' mobility profiles (*e.g.*, speed, direction, and acceleration). For instance, fast-moving clients may suffer more frequent handoffs [11], [12], which causes the rebuffering and reduces clients' QoE.

Mobility-profiled QoE-driven bandwidth allocation. To meet the first challenge, we leverage clients' mobility profiles and QoE-related information to design an end-to-end scheme. At the client end, each client first collects its state information, including mobility profiles (*e.g.*, speed, location, and direction) from mobile devices by GPS and Inertial Measurement Unit (IMU) [13] as well as QoE-related information (*e.g.*, rebuffering and bitrate) from DASH video player. The collected state information is then grouped, encrypted, and sent along with

Yali Yuan¶ and Weijun Wang¶ have equal contributions in this work.
Corresponding author: Xiaoming Fu.
Sripriya S. Adhatarao did this work at University of Göttingen, and now the author is working at Huawei Munich Research Center, Germany.

[1]A handoff occurs when a mobile device moves between two BSes or cells.

the `HTTP Request` for downloading the chunk at a specific bitrate to the server. Utilizing these values and BSes' information, the proposed bandwidth allocation technique (§ III-B) chooses the optimal allocated bandwidth for each client to maximize clients' QoE fairness dynamically in a mobile environment for real-time video streaming applications.

Challenge 2: How to satisfy the buffer requirement of buffer-sensitive clients due to their mobility?
Because of the movement, after a while, some clients may be more sensitive to the playback buffer size [14]. Besides, clients may not receive the complete chunk with the requested bitrate due to the short stay time in one BS.

High-compatibility server push strategy. To tackle this challenge, we propose a novel server push algorithm over HTTP/3 protocol named Slow Degrade Fast Recovery (SDFR) (§ III-C). Unlike traditional server push methods [15], [16], SDFR adds the buffer for needed clients by sacrificing their own requested bitrate level in time dynamically without affecting the existing bandwidth allocation strategy and other clients' view quality. It is designed with a transparent mechanism compatible with all existing ABR algorithms. Specifically, based on clients' current staytime, handover time, and remaining buffer size, the server identifies clients who suffer high-frequent rebuffering and activates the server push function for them.

Contribution and road map:

• We propose the first end-to-end QoE fairness scheme (§ III) for the mobile video traffic with multiple mobile clients over a shared bottleneck link, named VSiM, which consists of two key techniques: 1) dynamic and fair bandwidth allocation by incorporating clients' mobile profile and QoE-related information (§ III-B); 2) fast buffer filling for clients with lower playback time according to the requirement of the buffer-sensitive clients (§ III-C).

• We implement VSiM in both simulation and prototype. The experiment results show that VSiM improves more than 40% on QoE fairness (equal to resolution improvement of clients' viewing quality from 720p to 1080p) and ∼20% on average of the averaged QoE compared to state-of-the-art solutions.

II. Background and Motivation

We start with the background of the Quality of Experience and QoE fairness. We then use empirical measurements to elucidate the limitations of prior solutions and our motivations.

A. Background

Quality of Experience (QoE). For a video V of length L, let c_k represents the k-th chunk at a bitrate r where $r \in \{r_1, r_2, ...r_m\}$, and R_k denote the time spent for rebuffering. Then according to the video streaming literature [17], [18], the QoE observed by a client for the k-th chunk is calculated as follows:

$$QoE\left(c_k\right) = q\left(c_k\right) - \beta R_k - \gamma ||q\left(c_k\right) - q\left(c_{k-1}\right)||, \qquad (1)$$

where $q(c_k)$ refers to the improvement in quality with the requested bitrate for the chunk c_k, $R_k = \frac{c_k}{r} - b$ refers to the rebuffering time calculating by the difference between downloading time $\frac{c_k}{r}$ and remaining playing time b. $q(c_k) - q(c_{k-1})$ represents the smoothness between the chunk c_k and c_{k-1}. The parameter β penalises the gain in QoE with $q(c_k)$ for rebuffering while γ penalises the QoE gain with the loss of smoothness between c_k and c_{k-1}. Therefore, as per Eq. (1), in order to maintain a good QoE, a video player must ensure higher bitrates, low rebuffering, and higher smoothness during video streaming.

QoE Fairness. Let B denote the bottleneck bandwidth and N be the client's number. At a time period T, each client i watches a video consisting of M chunks. The total QoE of i-th client for viewing this video is denoted as QoE_i. Then, the max-min QoE fairness [2], a standard QoE fairness metric, is to *maximize* $min_{i\in[N]}\frac{QoE_i}{M}$, where $[N]$ is the set of positive integers $\leq N$. Max-min QoE fairness reflects the QoE improvement of the worst performing clients, which helps service providers to offer a fairer service for clients and encourages their engagements [18].

B. Motivation

For mobile video streaming applications, current models, like [2], [5], [6], [8], [10], with the connection-level fairness, (*i.e.*, occupying equal shared bandwidth of competing flows) may not ensure the QoE fairness for all clients, especially for those content providers with a larger number of users.

In fact, to encourage more users to participate, video service providers are more inclined to improve the viewing quality of users with lower bitrates, rather than improving the viewing quality of users with higher bitrates [18]. One of the largest video content providers, Netflix, already considered this problem and adopted a series of techniques [19], [20] (*e.g.*, three parallel TCP connections) to allocate a larger bandwidth for Netflix videos instead of considering connection-level fairness, reducing the rebuffering probability for low-buffer clients at video startup. Nevertheless, the fair clients' view quality among competitive network traffic is not incorporated, especially in a mobile network environment. Specifically, from the perspective of the service provider, there are two major drawbacks.

Clients' QoE can be affected by their mobility. Clearly, users may have different bandwidth allocation requirements in different scenarios [18], [21]. In mobile wireless networks, the mobility of users significantly affects network performance, including QoE and QoE fairness [22]–[27]. For instance, compared to low-speed clients, high-speed clients may require more allocated bandwidth within the same period to accomplish the same viewing quality due to the frequent handoffs or possible connection loss between BSes. Fig. 1 shows the view quality of clients over various speeds at time T with uniform linear motion. For simulation settings details, please refer to Section V. It is obvious to see that a client with high speed is more likely subject to low QoE.

[2]VSiM is flexible and different QoE fairness metrics can be used to evaluate VSiM. This paper uses the max-min QoE fairness as an example to demonstrate the performance of VSiM.

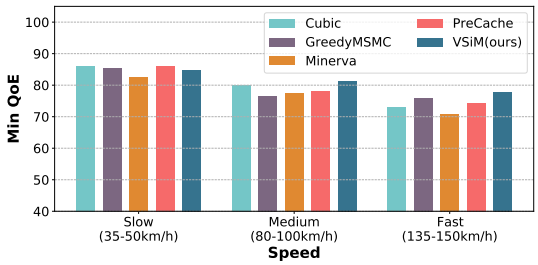


Figure 1: *High-speed clients may experience lower QoE.*

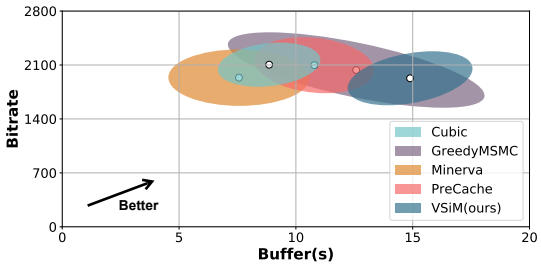


Figure 2: *The trade-off space between bitrate and buffer with the bottleneck bandwidth in a highly mobile environment.*

Mobile clients have different buffer-sensitive levels. Existing approaches, like [2], [5], [6], [8], [10], did not consider the state of the buffer-sensitive mobile video clients, like the playback buffer size; hence they are blind to the guidance information in the application level, such as increasing the playback buffer size for buffer-sensitive clients. For example, a mobile video client with a short staytime in a BS will experience a handoff time or a connection loss zone to go to the next BSes or networks. This may result in a higher chance for this client to suffer the rebuffering, which reduces its QoE. For such clients, increasing the playback buffer size is more critical to improving their QoE than requesting high-quality chunks. Besides, the whole QoE fairness can also be improved because of the improvement of minimum QoE. Therefore, *dynamic* and *adaptive* buffer update strategy is a good choice to help buffer-sensitive clients fast replenish their buffer size.

Traditional server push approaches did not consider the mobile characteristics in their design. Some server push strategies, like [15], [28] transmit the same quality chunk as the previous chunk, resulting in high downloading time and unsuitable for mobile video clients. Besides, [28] may drop all pushed chunks and wastes bandwidth resources. A novel server push strategy in VSiM is proposed to update the buffer size adaptively without affecting the existing bandwidth allocation strategy and other clients' viewing experience. Fig. 2 illustrates the trade-off space between the bitrate and buffer when facing bottleneck bandwidth in mobile environments. The larger the ellipse is, the greater the variance, resulting in the worse performance of the model. We can see that VSiM can increase the buffer size of clients significantly while slightly affecting the average bitrate.

III. VSiM System

This section introduce the design goals and overview of our system VSiM, then discusses its two key techniques, namely bandwidth allocation and server push.

A. Overview of VSiM

VSiM is an end-to-end solution for improving QoE and QoE fairness in a highly mobile environment. The main design goals that we wish to achieve in VSiM are: 1) efficiently incorporate various factors in mobile environments that can potentially impact the QoE fairness of clients during video streaming, 2) easy to deploy and configure in the real world, 3) maximize the QoE fairness while ensuring the total QoE, 4) adapt to the uncertainties in heterogeneous mobile wireless networks.

System architecture. At the client end, we exploit the ABR controller (see ❶ in Fig. 3) to collect the bitrate of the requested chunk and the buffer state of the Dash player. We further collect the information about each client's mobility profile from the sensors (see ❷ in Fig. 3) in their mobile devices. Since many smart devices collect such information using GPS and IMU [13], we assume that our system can also access such information on the clients' devices. The collected state information (see ❸ in Fig. 3) is then grouped, encrypted, and sent along with `HTTP Request` (see ❹ in Fig. 3) for downloading the chunk c_k at bitrate b_i to server.

At the server end, for each arriving `Request` from clients, the server decrypts the state information, and the trajectory predictor (see ❺ in Fig. 3) calculates clients' trajectories using mobility profile information and topology information of base stations (see ❻ in Fig. 3). Once the trajectory is known, the server identifies the BSes that the client connects with and the associated parameters such as the handover latency, staytime, and possible connection-less zones that will impact the QoE of the mobile client. We assume that the server is aware of the information of its needed BSes. Utilizing these values and the information from clients' DASH players (*e.g.*, buffer level and bitrate level), the utility computation module (see ❼ in Fig. 3) applies the utility function to calculate the optimal weight w_i for each client and transfer them to the bandwidth allocation module (see ❽ in Fig. 3) (§ III-B).

Since the server has a global view of all clients, it efficiently calculates and allocates the available bandwidth among the participating clients by considering their mobility profiles and QoE-related information. We assign the bandwidth by the weight w_i for each client using the Cubic congestion control approach [18], [29] such that the link capacity is utilized entirely and there is an improvement in the QoE fairness of all participated clients.

Meanwhile, based on these values and information, the server also identifies clients who are more likely to experience increased rebuffering due to a short staytime in a BS, handover latency, or connection-less zones. In order to improve the QoE of such clients, our system tries to fill their player

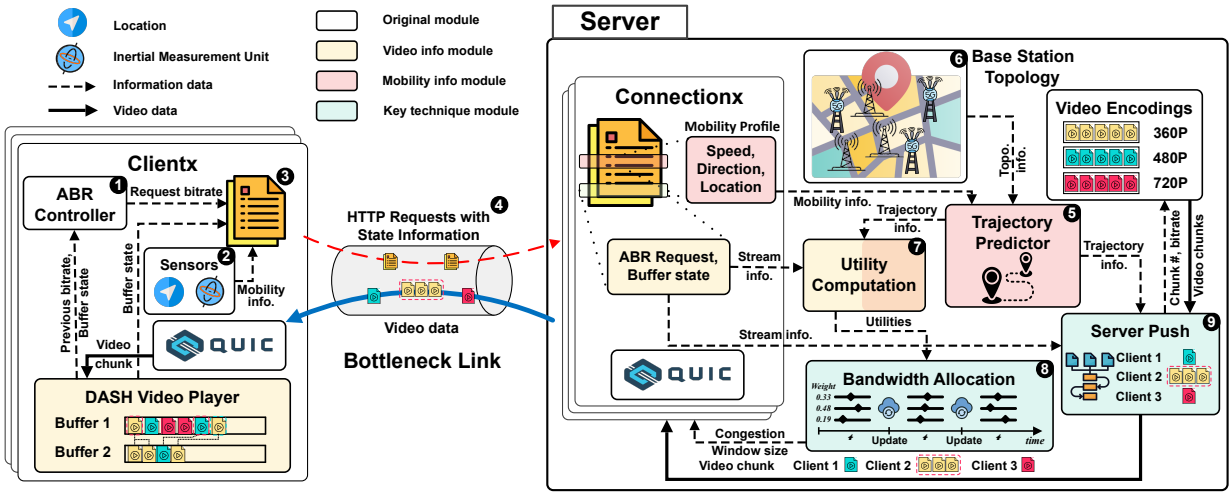


Figure 3: *VSiM improves the QoE and QoE fairness in a mobile environment by key techniques including Bandwidth Allocation strategy considering clients' mobile profile and QoE-related information, Server Push algorithm to avoid rebuffering.*

buffer to increase playtime and thereby reduce the effect of rebuffering. Server push module (see ❾ in Fig. 3) identifies these potential clients, prioritizes such clients, and pushes extra chunks to them. The original chunks and the pushing chunks will be stored in two buffers (§ III-C). Once the optimal bandwidth is allocated for each client and the server push gives an optimal push strategy, the prepared chunks are transmitted over the allocated bandwidth using the QUIC transport protocol. However, the push bandwidth is allocated only when necessary, and if the QoE fairness of all clients that share the same bottleneck bandwidth can be guaranteed.

Discussion. In VSiM, we place the mobility predictor on the server side because the server with high storage and computation ability can easily get BSes' information to predict clients' trajectories and ensure global QoE fairness for all clients. However, privacy leakage might happen when clients and BSes share their mobility and topology information respectively to the server. Privacy algorithms like differential privacy can protect users' data while ensuring models' performance, but it is beyond the scope of this paper. We will consider this in our future work.

B. Bandwidth Allocation

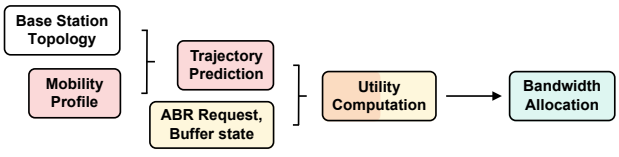


Figure 4: *Bandwidth Allocation strategy considering clients' mobile profile and QoE-related information.*

The bandwidth allocation is implemented by the utility computation based on various information, which is described in Fig. 4.

Utility Computation. We build the mathematical model for QoE fairness optimization of VSiM by allocating bandwidth. Let $U(r)$ be the utility function for bandwidth allocation. Based on the QoE definition in Eq. (1), $U(r)$ is built as a function of each client's download rate r, which is optimized by leveraging the clients' mobility profile information and QoE-related information (*e.g.*, bitrate level and buffer level). $U(r)$ is defined as Eq. (2).

$$U(r) = q(c_k) - \beta R'_k - \lambda |q(c_k) - q(c_{k-1})|, \tag{2}$$

where $q(c_k)$ represents the requested bitrate for chunk c_k, $\beta R'_t$ represents the rebuffering penalty, and $\lambda|q(c_k) - q(c_{k-1})|$ represents the penalty for variance in smoothness. The bandwidth allocation is dynamically changing in real-time.

Intuitively, rebuffering time aggrades along with handover time and chunk download time, degrades along with playtime remaining and staytime. Formally, rebuffering time is,

$$R'_k = \frac{c_k}{r} - b - t_s + t_h, \tag{2a}$$

where $\frac{c_k}{r}$ denotes the time to download the remaining chunk of c_k at a download rate of r, b denotes the playtime remaining in the clients' video player buffer. The parameter $t_s = \frac{d_s}{v}$ denotes the remaining time within the connection zone of BSes, where v is the client's moving speed and d_s is the remaining distance within the connection zone of BSes. t_h denotes the handover latency between the current and the next BS. Therefore, we mapped users' mobility characteristics to the rebuffer parameter R'_k. Both the staytime t_s and handover time t_h are decided by users' mobility profile, like the speed, direction, and acceleration. The higher t_s is, the higher QoE while t_h is inversely proportional to QoE.

The handover time t_h is defined as:

$$t_h = \begin{cases} \frac{d_h}{v} + \tau, & \text{no overlap between BSes,} \\ \tau, & \text{overlap between BSes,} \end{cases} \tag{2b}$$

where d_h is the distance occurring connection loss between BSes. τ is the time when clients switch between BSes. For example, a client travels from its current BS to next BS, if there exists the connection loss between these two BSes, then $t_h = \frac{d_h}{v} + \tau$. Otherwise, $t_h = \tau$. Please note that t_h and t_s are two predicted parameters reflecting characteristics of users' high mobility. Since the trajectory of each client is varying over time by changing the speed, direction, and acceleration, the calculation results of both t_h and t_s are also varying over time. Intuitively, when t_s is small and t_h is high, VSiM allocates more bandwidth to clients to improve their performance.

Bandwidth allocation. Given the above definition for the utility computation, the bandwidth weight for client i is calculated as $w_i = \frac{r_i}{\widetilde{U}(r_i)}$ where $\widetilde{U}(r_i) = \frac{U(r_i)}{U(B)}$ and the allocated bandwidth will be $r_i = \frac{w_i}{\sum_{i=1}^{n} w_i} B$, where i, n, and B represents the i-th client, clients number, and server's total bandwidth, respectively.

It is reasonable to say that the available bandwidth for client c_i can be B at most, thus, we could get

$$\arg\max_{r_i} U_i(r_i) = B. \tag{3}$$

Therefore, we could normalize the utility function as follows:

$$\widetilde{U_i}(r_i) = \begin{cases} 0, & r_i = 0, \\ \dfrac{U_i(r_i)}{U_i(B)}, & 0 < r_i < B, \\ 1, & r_i = B. \end{cases} \tag{4}$$

With the above utility function definition, We could model the problem of fair bandwidth allocation with maximizing the minimum QoE among all clients as

$$\max \quad \min_i \widetilde{U_i}(r_i) \quad \text{s.t.} \quad \sum_i r_i = B. \tag{5}$$

Theorem 1. *There exists an optimal allocation $\{r_i^*\}$ that reaches the goal in which $\{\widetilde{U_i}(r_i)\}$ are equal for all participating clients. At each time window, we could get a series of weights using $w_i = \frac{r_i}{\widetilde{U_i}(r_i)}$, then we could allocate the egress bandwidth as*

$$r_i = \frac{w_i}{\sum_{i=1}^{N} w_i} B.$$

The above allocation ensures that r_i will converge to r_i^* after t iterations.

Due to space constraint, we omit the convergence proof of bandwidth allocation. Please note that the time complexity of VSiM is very small, *i.e.*, $O(cn)$, where n is the number of clients sharing the bottleneck link and c is the iteration times required to converge to the optimal bandwidth allocation that maximizes the QoE fairness. The space complexity is also very small since clients' state information is refreshed on the server for each bandwidth allocation.

C. Server Push

The server push module is employed to decrease the frequency of rebuffering for buffer-sensitive mobile video clients. It significantly increases these clients' playback buffer by pushing multiple lower-bitrate chunks when they drop into an emergency. Meanwhile, server push should be compatible with the arbitrary ABR algorithms and should not offset bandwidth allocation benefit. Therefore, we carefully designed a novel server push algorithm called Slow Degrade Fast Recovery (SDFR). The core thinking includes: 1) Server *encapsulates multiple lower-bitrate chunks* back to the client according to client's state; 2) The bitrate of pushing chunk degrades level by level to *control the QoE smoothness*.

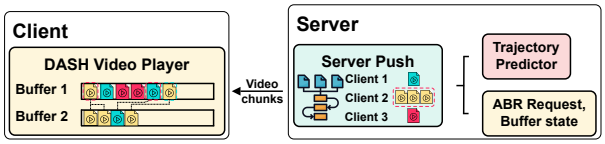


Figure 5: *Server push module utilizes clients' mobility and video stream info to determine if and how to push extra chunks.*

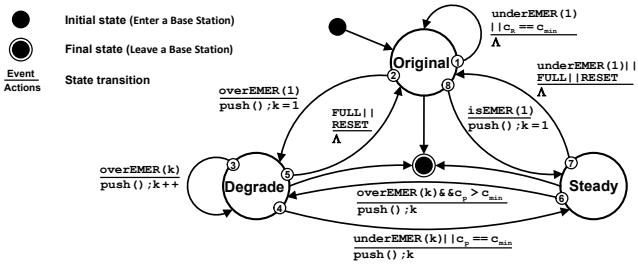


Figure 6: *Illustration on the server push algorithm Slow Degrade Fast Recovery (SDFR) as a state machine.*

Workflow. Fig. 5 illustrates the overview of server push. At the clients' end, Dash player maintains two buffers logically, in which Buffer 1 stores the original request video chunks and Buffer 2 stores the pushing ones;

it also maintains the relation of each chunk in Buffer 2 (*e.g.*, the encapsulation relation as illustrated by dotted line in Fig. 5) and delivers fake bitrate information to ABR algorithms. At the sever end, after receiving `HTTP Request` message, the server first estimates the client's state with the buffer state and the trajectory information predicted by trajectory predictor. Then, the server push module determines whether to push chunks or not with the client's state. If so, it will encapsulate multiple lower-bitrate chunks according to the SDFR algorithm; if not, it will respond to the requested chunk.

Fig. 6 shows the server push algorithm with a state machine model. SDFR sorts the bitrate level with an descending order resulting in a list $[c_{max}, c_2, \cdots, c_{min}]$. The bandwidth demand $B(\cdot)$ of these bitrate levels follows a total order: $B(c_i) < B(c_j)$ if $c_i < c_j$. c_p and c_R denote the pushing chunk bitrate and the client's bitrate request, respectively. Variable `k` quantifies client's emergency level. `isEMER(k)` denotes the event of the k-level emergency, whose condition is $t_h - b == k * t_s$ (The physical meaning is that the client needs to download *extra* k chunks per request in the following staytime such that its playback buffer have enough video to play during the handover time). Similarly, `overEMER(k)` and `underEMER(k)` denote the event of emergency exceeds k-level or lacks k-level, whose

conditions are $t_h - b > k * t_s$ and $t_h - b < k * t_s$, respectively. SDFR also receives *messages* from other modules: bitrate request of ABR algorithm c_R in `HTTP Request` from client; `FULL` when client's buffer is full; `RESET` when trajectory prediction module detects t_h changes. Sever maintains a server push state machine for each client and the state transition happens when the server receives the corresponding client's `HTTP Request`. Below, we describe each state and transition.

- **Initial.** Client enters a base station, then server push starts execution and changes to `Original`.
- **Original: responding chunk with `Request` bitrate.** Server push holds sending chunk with c_R bitrate. It stops at `isEMER(1)` (to `Steady`) or `overEMER(1)` (to `Degrade`). If t_h==0, it changes to `Final`.
- **Degrade: estimating the emergency level.** Degrading c_R directly to c_p may significantly decrease client's QoE if $c_R >> c_p$. To allow a smooth decrease, SDFR probes the emergency level by level (*i.e.*, adding one additional chunk controlled by `k++`). A larger k allows more chunks delivery. During emergency, server push runs the `push()` procedure to encapsulate k chunks with bitrate c_p into `HTTP Response` message, where c_p satisfies $\max\ c, s.t., k \cdot B(c) \leq B(c_R)$. After emergency estimated (`underEMER(k)||`c_p==c_{min}), server push changes to `Steady`. If server push receives `FULL` message or `RESET` message, it changes to `Original`. If t_h==0, it changes to `Final`.
- **Steady: multiple chunks pushing.** Client achieves high buffer fill rate by server push encapsulating and sending multiple c_p bitrate chunks with $B(c_R)$ bandwidth. It changes to `Degrade` when `overEMER(k)&&`c_p>c_{min}. If `underEMER(1)` or server push receives `FULL` message or `RESET` message, it changes to `Original`. If t_h==0, it changes to `Final`.
- **Final.** Client leaves a base station, server push ends.

IV. Implementation

We implement VSiM in both simulation and prototype. VSiM sits between the low-level functions (QUIC protocol) and the high-level applications (Dash video player on the client end and Video encoding on the server end). On the client end, VSiM modifies Dash player (Version 3.1.0) [30] by maintaining two buffers logically. On the sever end, all modules of VSiM in Fig. 3 are implemented in Go language (Version 1.13.8) based on QUIC-GO (Version 0.17.1) [31]. Two penalty parameters for rebuffering and smoothness in Eq. (2) are $\beta = 20$ and $\gamma = 0.1$. These values are set according to the simulation experience. Besides, the movement of clients incorporates three groups: slow movement ($v \in [35, 50] km/h$), medium movement ($v \in [80, 100] km/h$), fast movement ($v \in [135, 150] km/h$). The accelerations of car/motorcycle and train are within $[-8, 2.5] m/s^2$ and $[-7, 0.5] m/s^2$, respectively.

A. System Settings, Metrics, Dataset, and Benchmarks

System settings. We use a server equipped with Intel Core i7-5930K CPU at 3.5GHz, 32GB (DDR4 3000MHz) of RAM, Killer E3000 2.5Gbps Ethernet network port. All clients use Google Chrome (Version 83) with QUIC (HTTP/3) support enabled. We use ten devices including iPhone XR, Xiaomi Mi 8, Surface Go 2, IPad Air 4, IPad Mini 4, ThinkPad X1 Carbon, and MacBook Pro, as the mobile clients for the prototype test (§ V-C).

Evaluation Metrics. To better see the performance of VSiM, we regularize the scope of QoE within [0,100] [18] by Equation $a \times ln(x) - b$, where $a = 16.61$ and $b = 42.94$ for our employed datasets. For instance, about 5.8 points improvement with QoE normalization can accomplish a video quality jump from 720p to 1080p. It is defined by a large number of experimental statistics over the employed dataset. VSiM is evaluated by the following metrics: 1) QoE: It is employed to describe clients' viewing experience for the mobile streaming video, calculated in Eq. (1) (§ II-A). 2) Max-min QoE fairness: It reflects the QoE improvement of clients with the minimum QoE (§ II-A). 3) Minimum QoE: It represents the QoE of the client with the minimum value among all clients. 4) Cumulative Distribution Function (CDF): It reflects the QoE fairness improvement.

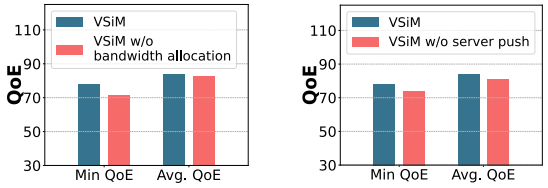
Datasets. VSiM works well on videos with varied bitrates from different sources. In this paper, we evaluate VSiM by a standard test dataset [32], which reflects the real-world distribution. It includes 20 videos. The value of c_k in our dataset ranges from 45kbps to 3936kbps. It includes low, middle, and high levels of bitrates.

Benchmarks. We have four benchmarks for comparison to prove VSiM's performance: (1) Cubic [33] with the average bandwidth; (2) Minerva [18], where QoE fairness is targeted for video streaming with a bottleneck link but without mobility consideration; (3) GreedyMSMC [8] achieving the QoE improvement by leveraging the load balance in base stations in a mobile environment; (4) PreCache [2] improves the QoE performance by pre-storing video in next base station's cache in mobile wireless networks. Cubic, GreedyMSMC, and PreCache are originally suitable for mobile scenes. For Minerva, we transplant its utility function and perceptual quality concept in our mobile experimental scenes. All algorithms are implemented in the same mobile environment for comparison.

B. Mobility Pattern

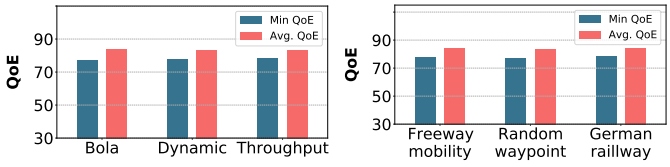
VSiM adapts to various mobility patterns, mapped to stay-time and handover time, to fulfill the QoE fairness for high mobile clients. This paper uses three mobility models as examples to evaluate our mechanism. 1) freeway mobility model [26] and railway mobility model [34]: mobile users are restricted to their lanes on the freeway/railway, and its velocity is temporally dependent on its previous velocity; 2) random waypoint model [35][3]: at every instant, a user randomly selects a destination and moves towards it with a velocity selected uniformly randomly from $[0, v_{max}]$, where v_{max} is the preset maximum velocity for each user.

[3]It is commonly used in simulations.



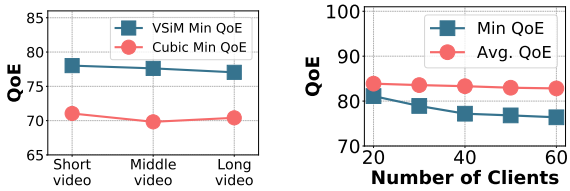
(a) Bandwidth allocation **(b) Server push**

Figure 7: *Each key technique brings contribution to VSiM QoE improvement on both QoE fairness and average QoE, where* VSiM w/o *means VSiM without the technique.*



(a) Various ABR algorithms. **(b) Various mobility patterns.**

Figure 8: *VSiM maintains stable and high QoE under various mobility patterns of clients and various ABR algrithms.*



(a) Various video lengths (Min. QoE). **(b) Various clients scale.**

Figure 9: *VSiM achieves high QoE under various video lengths; It also can ensure stable and high QoE fairness and average QoE under large-scale number of clients.*

V. Evaluation

In this section, we first introduce the simulation and prototype scenario. Then, we verify the contribution of each key technique and robustness of VSiM by simulations. Following that, the comparison of VSiM against state-of-the-art solutions with various metrics is illustrated over a prototype wireless network. We repeated the experiment for each bandwidth for 20 runs for all results.

Key takeaways. Key takeaways of our evaluations are:

- SDFR server push approach that fulfills the minimum QoE in VSiM is about 2.4 points (equal to clients' viewing experience jump from the bitrate level 2944kbps to 3340kbps in resolution 1080p) (Fig. 7).
- VSiM is robust for heterogeneous wireless networks including various video lengths and clients scale (Fig. 9), various ABR algorithms and mobility patterns (Fig. 8).
- VSiM improved more than 40% QoE fairness (equal to resolution improvement of clients' viewing quality from 720p to 1080p) compared to state-of-the-art while ensuring about 20% improvement for the average total QoE (Fig. 11 and 13).

A. Simulation and Prototype Scenario

We select the railway and highway from the train station in city A to the train station in city B (around $110km$). Along the road, we assume that this area is covered with 237 BSes consisting of 37 4G BSes and 200 5G BSes. The reason is that the 5G BSes are deployed every 500 meters and 4G BSes every 3km, depending on the communication range of BSes and area requirements (*e.g.*, high density of BSes in an urban area while low density in a rural area). The transmission ranges of 4G and 5G BS are $2km$ and $300m$, respectively [36].

B. Baseline Comparison

We verify the contribution of each key technique in VSiM and its robustness by simulations, where 20 to 60 clients are employed in the topology A and B.

Contribution of each key technique. We measure the contribution of each key technique by individual verifying VSiM with/without each technique with 60 clients and 150Mbps bandwidth in topology A, which is given in Fig. 7.

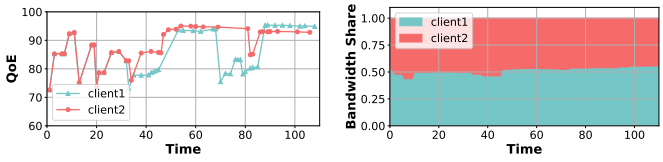
Specifically, in Fig. 7(a), we observe that the minimum QoE in VSiM is about 33% (about 4.8 points with the QoE normalization) higher than that in VSiM without the bandwidth allocation technique while accomplishing a desirable average QoE. This is equivalent to the minimum viewing quality of clients in VSiM without the bandwidth allocation strategy being 240p, while the minimum viewing quality of clients with that is 360p. Thanks to the bandwidth allocation technique (§ III-B), which leverages users' mobility profiles, requested bitrate, and playback buffer size to allocate the bandwidth fairly among clients. In Fig. 7(b), we notice that the minimum QoE and average QoE per client in VSiM is about 15% (about 2.4 points, equal to a viewing experience jump from the bitrate level 2944kbps to 3340kbps in resolution 1080p) and 13% higher than that in VSiM without the server push technique. Thanks to our proposed SDFR server push strategy (§ III-C) given the current buffer level, staytime, and handover time. This mechanism greatly improves the QoE fairness of clients.

Robustness of our system. VSiM is robust over various uncertainties, like different video lengths, ABR algorithms, and mobility patterns and the number of clients.

- *Impact of various video lengths.* In Fig. 9(a), we observe that VSiM accomplishes a stable minimum QoE over various lengths of videos, which are much better compared with those of Cubic, especially for the minimum QoE.
- *Impact of large-scale clients numbers.* In Fig. 9(b), we find that the performance (*e.g.*, minimal QoE or average QoE per client) of VSiM is almost constant under various clients numbers, showing the stability of VSiM against the variant number of clients. The slightly reducing trend of the minimum QoE and average QoE with the increasing number of clients in Fig. 9(b) is caused by the probability that the greater the number of users, the greater the likelihood that some clients will obtain lower QoE.
- *Impact of various ABR algorithms.* VSiM is transparent

Algorithm	100ms	200ms	300ms
VSiM (Min QoE)	**76**	**73**	**68**
Cubic (Min QoE)	69	67	63
VSiM (Avg. QoE)	**82**	**80**	**78**
Cubic (Avg. QoE)	80	78	73

Table I: *VSiM maintains high Min (Minimum) and Avg. (Average) QoE than Cubic under various latency conditions.*



(a) QoE of 2 clients. **(b) 2 clients bandwidth share.**

Figure 10: *QoE and bandwidth allocation of VSiM videos by a real wireless link over 10Mbps bandwidth in mobile networks.*

to ABR algorithms. In Fig. 8(a), we see that VSiM achieves good performance over different ABR algorithms regarding the minimum QoE and average QoE.

• *Impact of various mobility patterns.* In VSiM, we convert the users' mobility profiles to the staytime and handover time, which adapts VSiM to different mobility patterns, which is given in Fig. 8(b).

C. Prototype Test

We build a prototype test in a lab testbed to check VSiM's performance in real-world scenarios in the topology described in § V-A over 10 clients with bandwidths [10Mbps, 15Mbps, 25Mbps, 35Mbps]. We run the experiment under a multi-user scenario who travel between two German railway stations with $110km$ distance and run VSiM over an actual wireless network link in mobile networks.

Sensitivity to network settings. The impact of network uncertainties, like bandwidth variance and latency variance, are tested in this section.

• *Impact of bandwidth variance.* We report the bandwidth and QoE variations over time by a real wireless link in mobile networks at bandwidth 10Mbps with two mobile clients in Fig. 10 to show how the system assigns the bandwidth and the impact of bandwidth allocation on the QoE changes.

In Fig. 10(a) and (b), we observe that both the average QoE and allocated bandwidth of two mobile clients C_1 and C_2 are close at an initial period t (*e.g.*, $t \in [0, 60s]$). This is because both C_1 and C_2 at this period are moving inside the BS with a long staytime (*e.g.*, greater than $60s$). In this case, we give the same staytime value (*e.g.*, 60s) for these two clients, which is given to avoid one client occupying the whole bandwidth and further improve the QoE fairness. Then, after a while, C_2 is still inside the BS, but the staytime of C_1 is short and may go to the next BSes or experience some connection loss area because of the fast movement. VSiM captures this and utilizes the optimization strategy to improve the allocated bandwidth and QoE of C_1. In Fig. 10(b), it is clear to see that C_1's bandwidth increases. Because of the fixed total bandwidth, C_2's bandwidth is reduced.

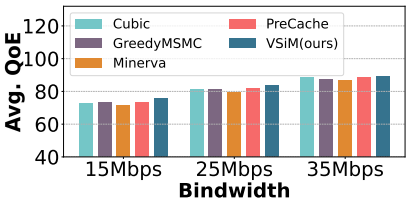


Figure 11: *VSiM fulfills a higher average QoE compared with various algorithms over different bandwidths.*

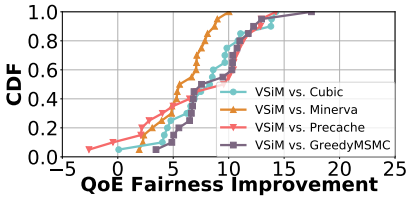


Figure 12: *QoE fairness improvement achieved by clients under various algorithms with 25Mbps bandwidth.*

• *Impact of latency variance.* Table I illustrates the impact of latency variance on VSiM and Cubic. For Table I, we observe that VSiM achieves better performance in terms of both the minimum QoE and average QoE than those of Cubic.

Compare with state-of-the-art. This section compares VSiM with state-of-the-art regarding the average QoE, QoE fairness, and CDF over various bandwidths.

• *Average QoE Comparison.* In Fig. 11, we observe that: 1) increasing the bandwidth will improve the mobile client's total QoE. This is because a higher bandwidth value leads to the DASH requesting a higher bitrate, which further improves each client's QoE; 2) VSiM outperforms state-of-the-art solutions with both ten mobile clients over different bandwidths regarding the average QoE.

Specifically, the average QoE improvement at bandwidth 25Mbps of VSiM is about 13% (about 2.0 points) and 30% (about 4.3 points) higher than that of Cubic [33] and Minerva [18] while around 11% (about 1.8 points) and 16% (about 2.4 points) on average improvement is achieved by VSiM compared with PreCache [2] and GreedyMSMC [8]. VSiM with more than 2.0 points improvement can at least jump one level of bitrate compared with state-of-the-art in terms of 1080p video. This means that in the same bottleneck bandwidth and mobile networks, the average bitrate value of all mobile clients that can be used is 3340kbps in Cubic while all mobile clients in VSiM can at least watch videos with 3613kbps for the average bitrate value regarding 1080p. This is because VSiM considers the mobility pattern and HTTP/3 characteristics (such as server push) to optimize the bandwidth allocation for different mobile users.

• *CDF Comparison.* Fig. 12 illustrates the CDF of QoE fairness improvement over Cubic, Minerva, GreedyMSMC, and PreCache with ten mobile clients collected by 20 runs in a real wireless network at 25Mbps bandwidth. As we discussed in Section IV-A, VSiM can accomplish a video quality jump from 720p to 1080p if the improvement with QoE normalization is greater than 5.8 points. We notice that there are $\sim$ 55% (about

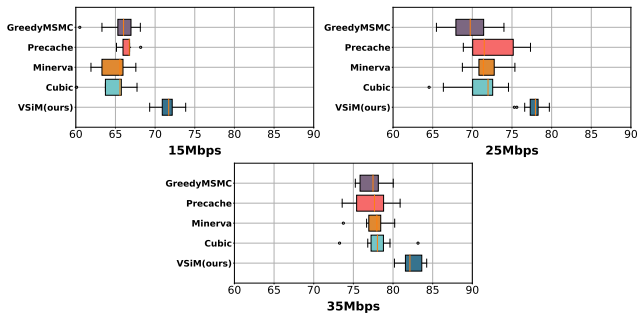


Figure 13: *VSiM fulfils a significant improvement of QoE fairness comparison with various algorithms.*

6.7 points), $\sim$ 40% (about 6.5 points), $\sim$ 30% (about 6.3 points), and $\sim$ 25% (about 6.4 points) probability for VSiM to achieve the value of QoE fairness improvement being larger than 5.8 points compared to Minerva, PreCache, Cubic, and GreedyMSMC. VSiM fulfills a video quality jump from 720p to 1080p with these probabilities. For example, suppose the minimum video quality of Minerva over all clients is 720p. In that case, the minimum video quality of VSiM over all clients has a probability of $\sim$ 55% to fulfill 1080p in the same bottleneck environment. This significant improvement depends on server push and bandwidth allocation in VSiM for a mobile environment.

• *QoE fairness comparison.* Fig. 13 records the clients with minimum QoE for each run. The longer the box, the greater the variance of the experimental results of different runs is, which means that the results are worse. As expected, the lowest bandwidth has the lowest QoE and vice versa. Besides, we observe that VSiM achieves good QoE fairness over varied bandwidths. For example, in Fig. 13, the QoE fairness for the median value of VSiM improves an average of about 40% (about 5.9 points with QoE normalization) for all the bandwidth than that of Minerva, which means that VSiM can accomplish a jump from 720p to 1080p. Especially for 15M bandwidth, the median QoE fairness of VSiM improves about 51% (about 6.9 points with QoE normalization) than Cubic. Compared to Minerva, GreedyMSMC, and PreCache, VSiM fulfills about 49% (about 6.6 points), 43% (about 5.9 points), and 36% (about 5.0 points) QoE fairness improvement on the average with aspect to the median value overall bandwidth. Additionally, we observe that the variance (*e.g.*, box size) of VSiM is small over different bandwidths.

VI. Related Work

We broadly classify the related video streaming optimization literature into the following two categories.

Single user: Huang et al. [37] proposed a method to improve the QoE. Mao et al. [38] presented a system Pensieve, which trains a neural network model to select future video chunks based on the current environment state. Dong et al. [39] proposed an online-learning congestion control algorithm called PCC Vivace to improve the video streaming performance. [4] attempted to optimize QoE by selecting the optimal initial video segment using deep reinforcement learning according to the network conditions (*e.g.*, signal strength). To improve QoE, [5] proposed to integrate the video super-resolution algorithm into the adaptive video streaming strategy by using the deep reinforcement learning approach.

Multiple users: A simple fairness definition would be to provide connection-level fairness, which ensures an equal allocation of network resources among competing flows [29], [40]. In this regard, Jiang et al. [41] proposed an algorithm to improve the fairness, while methods in [42], [43] try to ensure the QoE fairness for competing for flows by exploiting TCP-based bandwidth sharing. [44] proposed a method based on game theory to avoid selfish behavior, which achieves stable viewer QoE during video streaming. Vikram et al. [18] built a system named Minerva, which optimizes max-min QoE fairness by taking into consideration users' priorities in wired networks. QoE optimization is achieved by leveraging the load balance in base stations in [8], where Jain's fairness is used to achieve the bitrate-level fairness for the video streaming traffic. [2] predicted the next base station for mobile clients by using their mobility information and pre-store video in the next base station's cache to achieve the video quality consistency. [9] considered the video content, playing buffer, and channel status to optimize the QoE and achieve buffer-level fairness for HTTP adaptive streaming applications. Inspired by the congestion control of transmission control protocol, [10] considered the buffer filling rate, network capacity, congestion avoidance, and detection to optimize the QoE and QoE fairness.

VII. Conclusion

In this paper, we propose VSiM, the first end-to-end QoE fairness scheme for mobile video traffic with multiple mobile clients. VSiM leverages clients' mobility profiles, QoE-related information, and SDFR server push strategy to allocate bandwidth that maximizes the QoE fairness in real-time. VSiM is easy to deploy in the real world without touching the underlying network infrastructure. We implement VSiM in both simulation and prototype tests on top of HTTP/3. In the simulation, we verify the contribution of each key technique and robustness of VSiM, like different video lengths, various mobility patterns, as well as various clients number and ABR algorithms. In the prototype, we find that VSiM outperforms state-of-the-art approaches, with about 40% QoE fairness improvement (equal to clients' viewing experience in resolution from 720p to 1080p). Meanwhile, VSiM ensures about 20% improvements on average of the averaged QoE (equal to the bitrate level improvement of clients' viewing experience from 2087kbps to 2409kbps in 1080p resolution over the public dataset). In future work, we plan to test and deploy VSiM in real-world service provider networks.

References

[1] H. Riiser *et al.*, "Video streaming using a location-based bandwidth-lookup service for bitrate planning," *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, vol. 8, no. 3, pp. 1–19, 2012.

[2] J. Qiao *et al.*, "Proactive caching for mobile video streaming in millimeter wave 5g networks," *IEEE Transactions on Wireless Communications*, vol. 15, no. 10, pp. 7187–7198, 2016.

[3] J. G. Andrews, "Seven ways that hetnets are a cellular paradigm shift," *IEEE communications magazine*, vol. 51, no. 3, pp. 136–144, 2013.

[4] H. Wang *et al.*, "Rldish: Edge-assisted qoe optimization of http live streaming with reinforcement learning," in *IEEE INFOCOM 2020-IEEE Conference on Computer Communications*, 2020.

[5] Y. Zhang *et al.*, "Improving quality of experience by adaptive video streaming with super-resolution," in *IEEE INFOCOM 2020-IEEE Conference on Computer Communications*, 2020.

[6] S. Altamimi *et al.*, "Qoe-fair dash video streaming using server-side reinforcement learning," *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, vol. 16, no. 2s, pp. 1–21, 2020.

[7] I. Triki *et al.*, "Context-aware mobility resource allocation for qoe-driven streaming services," in *2016 IEEE Wireless Communications and Networking Conference*, pp. 1–6, 2016.

[8] A. Mehrabi *et al.*, "Joint optimization of qoe and fairness through network assisted adaptive mobile video streaming," in *2017 IEEE 13th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*, pp. 1–8, 2017.

[9] S. Cicalo *et al.*, "Improving qoe and fairness in http adaptive streaming over lte network," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 26, no. 12, pp. 2284–2298, 2015.

[10] M. Seufert *et al.*, "A fair share for all: Novel adaptation logic for qoe fairness of http adaptive video streaming," in *2018 14th International Conference on Network and Service Management (CNSM)*, pp. 19–27, 2018.

[11] L. Li *et al.*, "A measurement study on multi-path tcp with multiple cellular carriers on high speed rails," in *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication*, pp. 161–175, 2018.

[12] S.-H. Lin *et al.*, "Coverage analysis and optimization for high-speed railway communication systems with narrow-strip-shaped cells," *IEEE Transactions on Vehicular Technology*, 2020.

[13] W. Y. B. Lim *et al.*, "Federated learning in mobile edge networks: A comprehensive survey," *IEEE Communications Surveys & Tutorials*, 2020.

[14] J. Yuen *et al.*, "A fair and adaptive scheduling protocol for video stream transmission in mobile environment," in *Proceedings. IEEE International Conference on Multimedia and Expo*, vol. 1, pp. 409–412, 2002.

[15] S. Rosen *et al.*, "Push or request: An investigation of http/2 server push for improving mobile performance," in *Proceedings of the 26th International Conference on World Wide Web*, pp. 459–468, 2017.

[16] H. T. Le *et al.*, "Http/2 push-based low-delay live streaming over mobile networks with stream termination," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 28, no. 9, pp. 2423–2427, 2018.

[17] X. Yin *et al.*, "A control-theoretic approach for dynamic adaptive video streaming over http," in *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication*, pp. 325–338, 2015.

[18] V. Nathan *et al.*, "End-to-end transport for video qoe fairness," in *Proceedings of the ACM Special Interest Group on Data Communication*, pp. 408–423, 2019.

[19] J. Summers and p. y. others, booktitle=2016 IEEE International Symposium on Workload Characterization (IISWC), "Characterizing the workload of a netflix streaming video server,"

[20] H. Nam *et al.*, "A mobile video traffic analysis: Badly designed video clients can waste network bandwidth," in *2013 IEEE Globecom Workshops (GC Wkshps)*, pp. 506–511, 2013.

[21] K. Nagaraj *et al.*, "Numfabric: Fast and flexible bandwidth allocation in datacenters," in *Proceedings of the 2016 ACM SIGCOMM Conference*, pp. 188–201, 2016.

[22] X. Ge *et al.*, "User mobility evaluation for 5g small cell networks based on individual mobility model," *IEEE Journal on Selected Areas in Communications*, vol. 34, no. 3, pp. 528–541, 2016.

[23] X. Lin *et al.*, "Towards understanding the fundamentals of mobility in cellular networks," *IEEE Transactions on Wireless Communications*, vol. 12, no. 4, pp. 1686–1698, 2013.

[24] B. Gavish *et al.*, "The impact of mobility on cellular network configuration," *Wireless Networks*, vol. 7, no. 2, pp. 173–185, 2001.

[25] W. Alasmary and W. Zhuang, "Mobility impact in ieee 802.11 p infrastructureless vehicular networks," *Ad Hoc Networks*, vol. 10, no. 2, pp. 222–230, 2012.

[26] F. Bai *et al.*, "Important: A framework to systematically analyze the impact of mobility on performance of routing protocols for adhoc networks," in *IEEE INFOCOM*, vol. 2, pp. 825–835, 2003.

[27] J. Manner *et al.*, "Evaluation of mobility and quality of service interaction," *Computer Networks*, vol. 38, no. 2, pp. 137–163, 2002.

[28] J. Van Der Hooft *et al.*, "An http/2 push-based approach for low-latency live streaming with super-short segments," *Journal of Network and Systems Management*, vol. 26, no. 1, pp. 51–78, 2018.

[29] S. Ha, I. Rhee, and L. Xu, "Cubic: a new tcp-friendly high-speed tcp variant," *ACM SIGOPS operating systems review*, vol. 42, no. 5, pp. 64–74, 2008.

[30] "dash.js." https://github.com/Dash-Industry-Forum/dash.js.

[31] "A QUIC implementation in pure Go ." https://github.com/lucas-clemente/quic-go.

[32] S. Lederer *et al.*, "Dynamic adaptive streaming over http dataset," in *Proceedings of the 3rd multimedia systems conference*, pp. 89–94, 2012.

[33] S. Ha *et al.*, "Cubic: a new tcp-friendly high-speed tcp variant," *ACM SIGOPS operating systems review*, vol. 42, no. 5, pp. 64–74, 2008.

[34] "Railway mobility model." https://en.wikipedia.org/wiki/Metronom_Eisenbahngesellschaft. Online.

[35] L. Breslau *et al.*, "Advances in network simulation," *Computer*, vol. 33, pp. 59–67, May 2000.

[36] Y. Hao *et al.*, "Wireless fractal ultra-dense cellular networks," *Sensors*, vol. 17, no. 4, p. 841, 2017.

[37] T. Huang *et al.*, "A buffer-based approach to rate adaptation: evidence from a large video streaming service," in *In Proc. of ACM SIGCOMM*, pp. 187–198, 2014.

[38] H. Mao *et al.*, "Neural adaptive video streaming with pensieve," in *In Proc. of ACM SIGCOMM*, pp. 197–210, 2017.

[39] M. Dong, T. Meng, D. Zarchy, E. Arslan, Y. Gilad, B. Godfrey, and M. Schapira, "PCC vivace: Online-learning congestion control," in *In Proc. of USENIX NSDI*, 2018.

[40] M. Allman, V. Paxson, W. Stevens, *et al.*, "Tcp congestion control," 2009.

[41] J. Jiang, V. Sekar, and H. Zhang, "Improving fairness, efficiency, and stability in http-based adaptive video streaming with FESTIVE," in *In Proc. of CoNext*, 2012.

[42] Z. Li, X. Zhu, J. Gahm, R. Pan, H. Hu, A. C. Begen, and D. Oran, "Probe and adapt: Rate adaptation for HTTP video streaming at scale," *IEEE J. Sel. Areas Commun.*, vol. 32, no. 4, pp. 719–733, 2014.

[43] X. Yin, M. Bartulovic, V. Sekar, and B. Sinopoli, "On the efficiency and fairness of multiplayer http-based adaptive video streaming," in *In Proc. of IEEE ACC*, 2017.

[44] A. Bentaleb, A. C. Begen, S. Harous, and R. Zimmermann, "Want to play dash?: a game theoretic approach for adaptive streaming over HTTP," in *In Proc. of ACM MMSys*, 2018.