IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received December 26, 2017, accepted January 27, 2018, date of publication February 12, 2018, date of current version March 16, 2018.

Digital Object Identifier 10.1109/ACCESS.2018.2804222

Achieving Robust Mobile Web Content Delivery Performance Based on Multiple Coordinated QUIC Connections

PENG QIAN, NING WANG, (Senior Member, IEEE), AND RAHIM TAFAZOLLI, (Senior Member, IEEE)
5GIC, University of Surrey, Guildford GU2 7XH, U.K.

Corresponding author: Peng Qian (p.qian@surrey.ac.uk)

This work was supported in part by the EPSRC CONCERT Project under Grant EP/L018683/1 and in part by the EPSRC KCN Project under Grant EP/L026120/1.

ABSTRACT In order to minimize the downloading time of short-lived applications like Web browsing, Web application, and short video clips, the recently standardized HTTP/2 adopts stream multiplexing on one single TCP connection. However, aggregating all content objects within one single connection suffers from the head-of-line blocking issue. Quick UDP Internet connection (QUIC), by eliminating such an issue on the basis of UDP, is expected to further reduce the content downloading time. However, in mobile network environments, the single connection strategy still leads to a degraded and high-variant completion time due to the unexpected hindrance of congestion window growth caused by the common but uncertain fluctuations in round trip time and also random loss event at the air interface. To retain resilient congestion window against such network fluctuations, we propose an intelligent connection management scheme based on QUIC which not only employs adaptively multiple connections but also conducts a tailored state and congestion window synchronization between these parallel connections upon the detection of network fluctuation events. According to the performance evaluation results obtained from an LTE-A/Wi-Fi testing network, the proposed multiple QUIC scheme can effectively overcome the limitations of different congestion control algorithms (e.g., the loss-based new reno/CUBIC and the rate-based BBR), achieving substantial performance improvement in both median (up to 59.1%) and 95th completion time (up to 72.3%). The significance of this piece of work is to achieve highly robust short-lived content downloading performance against various uncertainties of network conditions as well as with different congestion control schemes.

INDEX TERMS Congestion control, mobile wireless networks, QUIC, transport protocols, Web content downloading.

I. INTRODUCTION

Nowadays, as the content volume downloaded through mobile wireless networks have already exceeded their wired counterparts, downloading acceleration of the diverse short-lived applications like webpage browsing, web application and short video clips in such environments has attracted significant research attentions in both academia and industry [1]. In recent years, in order to cater for user Quality of Experience (QoE) when consuming such content applications, various approaches have emerged targeting to tackle the inefficient interaction between Internet protocols and the variant network conditions unprecedentedly [1], [2]. Such initiatives have stimulated the standardization of next-generation protocols of mobile Internet like HyperText Transfer Protocol (HTTP) 2.0 [3] and Quick UDP Internet Connection (QUIC) [4].

In order to eliminate the application-layer Head-of-Line (HoL) blocking issue in HTTP/1.1, HTTP/2 introduces a series of new features such as efficient stream multiplexing, content pushing/hint and header compression, attaining enhanced bandwidth utilization on its single TCP connection [5]. However, given the dynamicity and uncertainty of network resource availability in mobile environments [6]–[8], this approach based on one single TCP connection still fails to best utilize the available bandwidth due to the transport layer HoL blocking [5] and the compulsory three-way handshake procedure. In order to address these issues of TCP, by leveraging UDP as underlying protocol, the emerging

protocol QUIC successfully eliminates HoL blocking at the transport layer and it further simplifies the handshake procedure to only 0 or 1 round-trip time (RTT) [4]. Compare to HTTP/2 over TCP, the content downloading time of QUIC is substantially improved in lossy network environments [9], and the reduced handshake complexity and flexible stream multiplexing also yield notable efficiency improvement in the delivery of today's short-lived applications [9].

Despite these improvements, the inherited Congestion Control Algorithms (CCA) on the single connection of QUIC still suffer from unexpected hindrance of congestion window growth caused by the unavoidable network fluctuation events (NFE) such as packet loss and RTT variations which are very common in wireless environments. It has been reckoned that this has become an essential technical barrier to further improving QUIC's performance [10]–[12]. For instance, by adopting the loss-based CCAs such as New Reno or CUBIC and the hybrid slow start algorithm [13] as default setting, the exponential growth of congestion window at the Slow Start (SS) phase can be prematurely terminated by either an inappropriate detection of RTT variation exceeding the pre-define threshold or the occurrence of a random packet loss [7], [12], despite that the design principle of these CCAs is to correctly identify the bandwidth bottleneck. Consequently, the connection staying at the Congestion Avoidance (CA) phase will only allow a conservative congestion increase ratio, thus significantly jeopardizing the downloading performance at the application layer. On the other hand, the most recently proposed rate-based Google BBR [2] which leverages per-round rate estimation to drive the congestion growth is expected to be more robust to such uncertainty in network condition fluctuations. However, currently its practical performance when integrated with QUIC is still unknown, and additionally existing findings of the TCP BBR performance can be even worse than CUBIC [14] under certain conditions.

Specific to the short-lived content applications, the usage of one single connection is more vulnerable to such inefficient interaction between protocol and network condition fluctuations. This is because, compare to long-live content which has the opportunity to achieve a full rate by the rate recovery algorithm in CUBIC or BBR during its long downloading period, the majority of short-lived content is expected to be completely downloaded within the SS phase, thus upon even single NFE, the sharp drop of congestion window or early termination of exponential growth phase on single connection which carries all the contents will fail to best utilize the available bandwidth before the content downloading is completed [12].

In this paper, we first study the impact of NFEs on QUIC's performance in order to quantify the significance of the technical issue described above. Towards this end we conducted a series of experiments on a real LTE-A testing network with various pluggable CCAs at the transport layer. The in-depth analysis based on the results revealed that different CCAs have distinct inefficiencies in the presence of NFEs that can take place in mobile wireless environments. Once such a problem has been numerically elaborated, we propose a sender-based connection management (mQUIC) scheme that is not only able to enable adaptively multiple connections to absorb the uncertainty in network condition fluctuations but also to execute a NFE-driven intelligence for orchestrating the multiple parallel connections. By aggregating the real-time feedback from multiple concurrent connections, each of which independently runs its own CCA, the mQUIC scheme adaptively synchronizes the transport layer state and congestion window of all connections to retain an optimized growth rate of congestion window, and the adopted synchronization strategy can be tailored to fit different plugged-in CCAs.

To the best of our knowledge, this represents the first work to provide comprehensive QUIC performance evaluation in real LTE-A testing network with all common CCAs, and on top of that novel intelligence is introduced to provide highly robust user downloading performance under different uncertainties. Different from conventional middlebox-base parallel TCP approaches like [15] and [16], mQUIC only requires servers-side and user-level modification, while protocol re-encapsulation, kernel modification or the assistant of third-party network entity are not necessary. Therefore, this design strictly retains the end-to-end security which is a compulsory feature of QUIC [17]. A prototype implementation is also specified in this paper, and extensive testing results have been carried out from local LTE-A/Wi-Fi testing network. Generally, the customized approaches effectively help each CCA to overcome its individual limitation under specific NFEs. Specifically, for loss-based CCAs like New Reno, the median completion time of a web content can be improved up to 59.1% and the 95th percentile completion time is improved by up to 72.3%. For rate-based CCA like BBR, mQUIC can successfully speed up its median and 95th downloading time up to 27.4% and 31.4 %, respectively. Furthermore, this technique is able to achieve highly robust content downloading performance under various network conditions, content size, as well as the initial congestion window size. This is in contrast to the plain QUIC-based approach where the actual performance can be very sensitive to specific network conditions and configurations.

II. BACKGROUND AND RELATED WORKS

In this section, first we give an overview of the QUIC protocol by shedding light on its latest practice enhancements in the literature accompanied by the corresponding performance analysis. Then we provide a review of different schemes that apply multiple connections for performance enhancements, pointing out their limitations which inspired us to design the proposed mQUIC scheme based on adaptively multiple connections.

A. OVERVIEW OF QUIC AND ITS PERFORMANCE ANALYSIS IN THE LITERATURE

The key features that enable QUIC to outperform traditional protocols include simplified handshake procedure, packet

pacing, enhanced loss recovery mechanism and the flexibility of the user-space CCAs. These features have been continuously improved according to Google's on-going experiments since it was proposed [4]. First, a QUIC client only needs 1 RTT to establish a secured connection to content server and thanks to the cached session security information, there is no time consumed when this client tries to re-connect to a previously visited server. Second, in order to reduce the retransmission latency of a lost packet, QUIC adopted Forward Error Correction (FEC) code at its early experimental stage. However, previous evaluation revealed that the bandwidth sacrificed to append the proactive XOR correction code is significant [11]. The reason behind this unexpected observation is that a recovered loss by the FEC module does not trigger the congestion control and even makes the algorithm more aggressive [11]. Therefore, in the latest version of QUIC, FEC has been disabled but another anti-loss approach ''packet pacing'' has been reserved. Content servers with packet pacing enabled can actively monitor the packet train to adaptively adjust the pacing interval, which avoids traffic burst and effectively reduces the number of lost packets [4]. Third, in addition to the elimination of the HoL blocking at transport layer, QUIC improves loss recovery by using unique packet numbers to avoid retransmission ambiguity and explicit signaling in ACKs for achieving accurate RTT measurements, thus guaranteeing the in-order packet delivery at the application layer. Fourth, QUIC also provides a flexible interface that allows the modification of the pluggable CCA at user-level space. Such an open feature can easily facilitate the deployment of various application-aware algorithms, supporting iterative changes at application update timescales. Currently the default CCAs supported are loss-based (New Reno and CUBIC), and the latest rate-based CCA Google BBR [2] is also ready to be enabled as an alternative option on the latest version of QUIC. In our work, all these three CCAs are analyzed and supported but to comply with today's practical CCA deployment [18] and the latest IETF standard [4], we select loss-based CCAs as the default configuration for QUIC. Finally, in terms of the Internet-Scale deployment of QUIC, some popular websites like YouTube gradually deploy QUIC for delivering their content services [17], and to the best of our knowledge the policy ''strict single connection per host'' is still applied on the QUIC enabled server [17].

As QUIC starts to attract increasing attentions in the community, its performance for short-lived application downloading has been investigated. There is a general observation that HTTP/2 with QUIC outperforms HTTP/2 with TCP for short-lived application in the presence of random loss [9], [10], [19], mainly thanks to the elimination of HoL blocking. Meanwhile, the reduced number of rounds required for connection establishment helps QUIC achieve noticeable acceleration, especially in the network environments with long RTT [19]–[21]. However, despite these QUIC's advantages, there are other content-related metrics for which QUIC may see its performance less attractive. For instance, as the number of embedded object in a webpage increases, the performance gain of QUIC over TCP becomes less significant due to a surging content queuing time at the server side [9]. Moreover, its downloading completion time under high packet loss condition is still relatively poor and sometimes even worse than HTTP 1.1 where multiple connections are in place [9]. Furthermore, works in [17] and [22] confirm that applying QUIC on video service can help improve the video quality, including initial playout latency and rebuffering frequency. However, these benefits cannot be commonly observed over different regions, and the performance disparity of different client-side video adaption algorithms like DASH BOLA [23], SQUAD [24] and BBA2 [25] also introduce practical concern of the upcoming large-scale deployment of QUIC. Given the ongoing trend of the network evolution, it is a critical question whether QUIC could boost various content performances in future mobile environments with high bandwidth capacity, low RTT and loss ratio, thus more in-depth performance studies are required especially in realistic mobile network environments.

B. APPLYING MULTIPLE CONNECTIONS AT THE TRANSPORT LAYER

In order to overcome the limitation of TCP protocol, various multiple connection based approaches have been proposed in the literature. We provide a brief literature review according to the following three categories:

1) STEADY-STATE BASED BANDWIDTH AGGREGATION TO IMPROVE THE THROUGHPUT FOR BULK DATA TRANSFER

Altman *et al.* [26] introduced a steady-state model of New Reno to validate that leveraging multiple connections can overcome the throughput drop in a lossy network and this has been further extended to a wireless network where different loss patterns were considered [27], confirming that the multi-connection approach is still beneficial in mobile network for large data transfer. Additionally, in [28], a parallel algorithm is proposed to maximize the bandwidth utilization for file transfer in cloud data center environments. However, these works only focused on bulk data transfer (e.g. at the order of hundreds of mega-bytes [28]) based on the steady-state model. The performance degradation is largely caused by packet loss, and this is in contrast to the short-lived content transfer scenario which normally completes during the SS phase in a wireless network with not only random loss but also RTT variations.

2) EXPLORING MULTI-PATH OR MULTI-SOURCE FEATURES TO OVERCOME NETWORK BOTTLENECKS

In today's mobile Internet, the widely-adopted content distribution technologies (e.g. CDNs) and the diverse radio access technologies provide fertile ground for utilizing multiple servers or paths to avoid single performance bottleneck. For instance, Kim *et al.* [29] leverage multiple uncorrelated paths to the distributed content servers to dynamically allocate HTTP byte-range request, according to the real-time application transferred volume. Similarly, the content requests are

separated to multiple wireless interfaces served by different ISPs to overcome the throughput degradation in [30]. However, performance enhancements relying on additional network facilities raise the concern of practical deployment cost. In [31], a systematic evaluation of multi-path TCP with HTTP/2 proved that under a high latency variation scenario, multi-path TCP can have worse perform due to its suboptimal TCP connection management.

3) INTEGRATING WITH MIDDLEBOX OR LEVERAGING PROTOCOL RE-ENCAPSULATION TO OVERCOME HoL BLOCKING

Qian *et al.* [15] proposed a multi-pipe proxy with transport layer re-encapsulation to allocate a global sequence number to reschedule the blocked packet when HoL blocking occurs caused by packet loss. Such a solution outperforms standard HTTP/2 by 24% in terms of average completion time. Xu *et al.* [32] investigated the performance of web browsing with the help of a TCP proxy in various cellular networks. The commonly observed performance improvement confirms that the network uncertainty such as random loss from proxy to server can be effectively alleviated. Similarly, beside integrating application layer approaches including intelligent cache and prefetching on the middlebox, it is also revealed in [33] that up to 36% of the webpage downloading time can be reduced by applying multiple connections from the proxy to the server in real network environments. However, proxy enabled approaches can introduce various practicality concerns such as the breaking of the end-to-end security, kernel level modification on the client, server and proxy sides [15], as well as the compatibility with firewall across mobile core networks [34]. More importantly, the benefit of utilizing multiple connections in UDP (a HoL blocking free protocol) in a fluctuating mobile environment is still unknown.

III. QUANTIFYING THE BOTTLENECKS OF QUIC WITH DIFFERENT CCAs IN MOBILE ENVIRONMENTS

In order to realistically quantify QUIC's performance bottleneck caused by the negative impact of NFEs in mobile networks, in this section we present an in-depth analysis based on the experiment results carried out in a locally controlled LTE-A testing network. In the experiments, the pluggable CCAs commonly used at today's content servers are separately tested to give a comprehensive understanding of their distinctive limitations. Based on the key observations analyzed in this section, we will formally introduce our proposed mQUIC scheme in section IV.

A. EXPERIMENT SETUP

Figure 1 illustrates the setup of the local LTE-A testing network. To cater for the trend of the evolution of mobile networks offering high bandwidth, low latency and low loss, our test-bed configuration adopts a band 41 LTE-A TDD network with all locally controlled network elements including SGW, PGW and eNodeB, which has been used for performance evaluation of our previous work [35]. The detailed

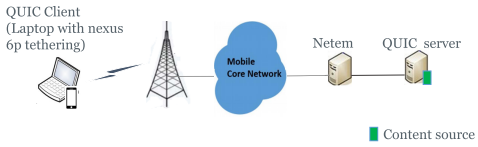


FIGURE 1. Local LTE-A testing network.

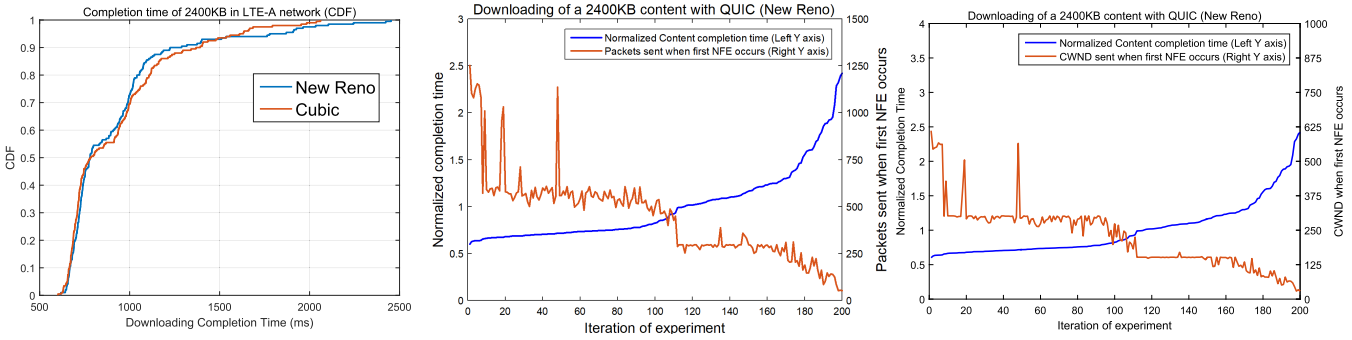
TABLE 1. Measured network condition in local LTE-A testing network (R: radio B: backhaul).

LTE-A				
RTT		Bandwidth		Loss
Median	Std	Median	Std	Median
R: 21.2 ms B:25 ms	3.46 ms	65.9 Mbps	5.1 Mbps	R: 0.09% B: 0.05%

specification of specific component can be found in [36]. In terms of the client side setup, we use a laptop (Ubuntu 16.04 with kernel 4.10) tethering a nexus 6P phone attached to the locally deployed LTE-A testing network. The laptop is kept as stationary during the experiment and its RSRP is around −76dBm, which can be regarded as "Good" in an LTE/LTE-A scenario. The measured network performances at radio interface are listed in Table 1. The QUIC content server is placed behind the core network and the size of synthetic web content varies as 500KB, 2400KB and 6000KB to emulate a small/medium/large web content [37]. A middlebox running *netem* [38] is deployed before the QUIC content server to add a 25ms latency and 0.05% loss rate [31], [39]. To ensure all latest updates are correctly enabled in our experiment we fork the QUIC code from Google Chromium 61 and leave all settings as default for both the client and server sides. In order to maximally guarantee the reliability of the test result, the downloading of each content with different CCAs are repeated by 200 times on a pre-warm connection [40] without any network condition parameters cached and all necessary configurations like enabling *fq* on Ubuntu system for BBR are set as instructed [40].

B. PERFORMANCE OF QUIC WITH NEW RENO/CUBIC

First, in order to investigate the negative effect of NFE on QUIC's loss-based CCAs, we provide a detailed analysis of the 2400KB content downloading with its CDF of completion time and a coupled transport layer trace. From Fig. 2a, it can be observed that CUBIC and New Reno share a similar performance curve. Specifically, the best case of the downloading time for both New Reno and CUBIC is around 0.56s, indicating an exponential growth of the congestion window during each round without any interruption of NFE (verified by inspecting the experiment dataset). Holding this best case as a baseline, its median completion time (around 0.82s) has substantial potential to be further improved. Furthermore, the long-tailed curve indicates a severe performance fluctuation with a high Coefficient-of-Variation (CoV) of 0.36. This performance variation can be attributed to the occurrence of NFE during the content downloading that triggers the connection to prematurely exit its SS phase, consequently being



(a) (b) (c)

FIGURE 2. Content downloading time of a 2400KB content in LTE-A network. (a) CDF of content downloading time (2400KB). (b) Coupled trace: Sorted content downloading time and number of packet sent when first NFE occurs. (c) Coupled trace: Sorted content downloading time and the corresponding congestion window when first NFE occurs.

TABLE 2. Content downloading performance of QUIC with New Reno/CUBIC in LTE-A testing network.

	Network type	LTE-A	
CCA	Content	Median	CoV
NEW RENO	6000KB	1.66 s	0.4
	2400KB	0.81 s	0.36
	500KB	0.28 s	0.22
CUBIC	6000KB	1.58 s	0.39
	2400KB	0.82 s	0.34
	500KB	0.27 s	0.23

TABLE 3. Content downloading performance of QUIC with BBR in LTE-A test network.

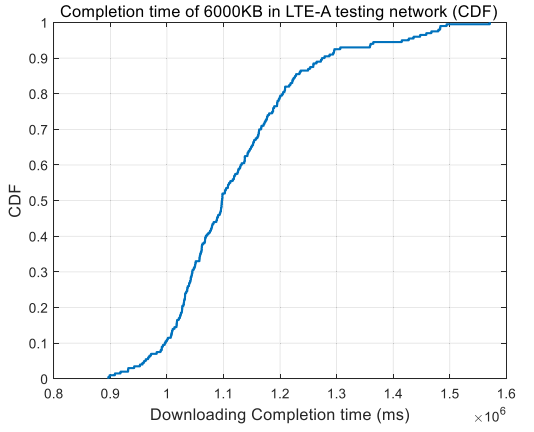
	Network type	LTE-A	
CCA	Content	Median	CoV
BBR	6000KB	1.13 s	0.11
	2400KB	0.62 s	0.1
	500KB	0.28 s	0.05

far away from the best case that the SS phase is expected to finally achieve. To further elaborate the fundamental cause of the severe performance variation, in Fig. 2b, we present the coupled trace of the CDF of the downloading time (normalized by the median) and its corresponding number of sent packets when the first NFE happens. Here the NFE is referred to as any single trigger of hybrid slow start's latency variation detection and New Reno/CUBIC's loss event. The high negative cross-correlation (−0.85) between these two curves substantiates that the downloading time which is much worse than median completion time (e.g. 1.5 times of the median completion time) is caused by an earlier occurrence of the NFE when only a few parts of content objects have been delivered. Therefore, this NFE forces the single QUIC connection to prematurely exit the SS phase and more content will be transmitted at a lower rate, resulting in a comparatively longer transmission time. In addition, in Fig. 2c, the congestion window when the first NFE happens and the corresponding completion time are compared, the high negative correlation (−0.86) between the two provides further evidence that the peak congestion window and the performance is determined by the stage when NFE happens.

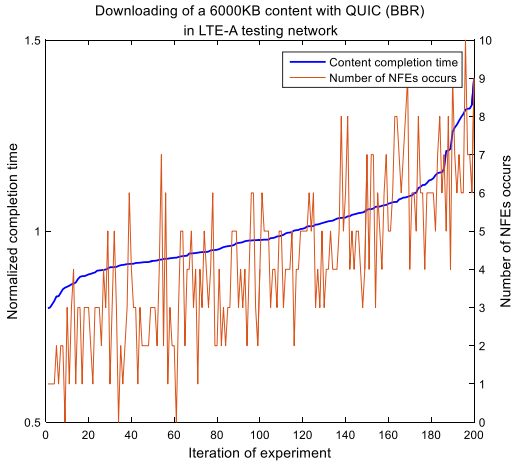
Table 2 provides the extended evaluation results by varying the content size. It is obvious that smaller content (500KB) has comparatively more robust performance (CoV is 0.22) while medium and large contents (2400KB and 6000KB) experience similar high variant downloading time (with CoV of 0.36 and 0.4, respectively). This is because that larger content will experience a longer transmitting time in the network, during which more NFEs are likely to happen. Meanwhile, if these NFEs happen at the early phase of the downloading period, more remaining content will be delivered at the limited rate. In terms of different loss-based CCAs like CUBIC and New Reno, their performance difference is minor since they both adopt hybrid slow start algorithm at the SS phase. Their disparity in CA phase is also minor due to the relatively shorter downloading time.

C. PERFORMANCE OF QUIC WITH BBR

Table 3 summarizes the content downloading performance when BBR is used as the CCA in QUIC. Thanks to the per-round rate-based estimation, the completion time is improved and more robust compared to loss-based CCAs. However, for large content 6000KB in Fig.3a, its completion time still has a relatively higher CoV (0.11). Similarly, the medium content also achieves an improved median completion time with a CoV at 0.09. In contrast, the small content performs more stable (CoV is 0.05), although compare to loss-based CCA, its median completion time stays the same. With these general performances in mind, we turn our focus to the performance robustness, especially for the large content case which still suffers from a comparatively high variant completion time. Fig. 3b depicts the relationship between the CDF of content completion time and the corresponding number of NFEs during each single downloading session. The high cross correlation (0.90) between these two metrics reveals that, unlike loss-based CCA, the key metric caused by NFE that determines the performance of BBR is the number of NFEs experienced during the content downloading. More importantly,



(a)



(b)

FIGURE 3. Content downloading time of a 6000KB content in LTE-A network (BBR enabled). (a) CDF of 6000KB content downloading time. (b) Correlation between NFE number and completion time.

here the majority of NFEs we observe from the experiment dataset is the random packet loss, while the RTT variation's effect is less impactful since BBR does not adopt the RTT variation based hybrid slow start algorithm. Furthermore, according to our verification, currently the exit condition on SS in the latest BBR version in both TCP kernel and QUIC is that if the current rate estimation is less than 1.25 times of previous rate estimation and this event occurs more than 3 times, the SS phase will be terminated. Therefore, in our test environment, the BBR connection rarely prematurely exits its startup phase, although the RTT variation can still lead to a rate degradation. Regarding the random packet loss event, according to [41], the corresponding behavior of BBR is defined as "*Upon exiting loss recovery (RTO recovery or Fast Recovery), either by repairing all losses or undoing recovery, BBR restores the best-known cwnd value we had upon entering loss recovery.*" This means that upon a single random packet loss or several lost packets which are automatically identified as a random loss event by QUIC, the exponential growth will be temporally stalled by the loss recovery phase. This loss recovery is expected to take at least one RTT and once the recovery phase ends, the exponential growth phase will be resumed. Apparently, as displayed in Fig. 3b, if more random losses occur, the connection will stay at the loss recovery phase for a longer time, leading to a degraded overall downloading time. To briefly summarize the above evaluation results and analysis, we list our key findings based on different CCAs as follows:

1) All CCAs are negatively affected by NFE, while loss-based CCAs are more sensitive to both of RTT variation and random loss since they could both lead to a prematurely exit of the SS phase. In contrast, the rate-based CCA such as BBR is mainly affected by the packet loss which can result in a temporal rate stall, while the negative effect caused by RTT variation is comparatively less significant due to its aggressive start up algorithm.
2) The performance issue associated to short-lived content downloading in mobile network is not only the degraded median completion time caused by NFE but also the severe performance variation which corresponds to the time point and the number that NFE occurs.
3) Large-sized content is more vulnerable to such NFEs compared to small content since the overall downloading time is longer, thus any early rate or congestion window stall will lead to more severe performance degradation.

From the above findings, there is no doubt that dealing with the unavoidable NFE like packet loss and RTT variation becomes a critical issue for provisioning a fast and robust content downloading performance of QUIC. More specifically, the distinctive and inevitable limitations of various CCAs also inspire us to design an adaptive approach to tackle each CCA that is more preferred over necessarily proposing a brand new CCA.

IV. STATE AND CONGESTION WINDOW SYNCHRONIZATION WITH mQUIC ENGINE

In this section, we formally introduce the proposed mQUIC scheme. The key novelty of the mQUIC scheme is that by applying multiple connections, it synchronizes the state and congestion window on each parallel connection in a NFE-driven manner, sustaining a sufficient aggregated congestion window in the multi-connection scenario. The detailed illustration of NFE-driven synchronization for different CCA types is presented, including both standardized (New Reno) and recently proposed (BBR) techniques.

A. OVERVIEW OF mQUIC SCHEME

The proposed mQUIC scheme adopts multiple parallel QUIC connections. The advantage of applying multiple connections which share the same bottleneck path is that, given the improved network performance of LTE/LTE-A network [7], [8], [42], the loss event and RTT fluctuation may only simultaneously affect a single connection or parts of the connections. Therefore, the unaffected connections can still stay in

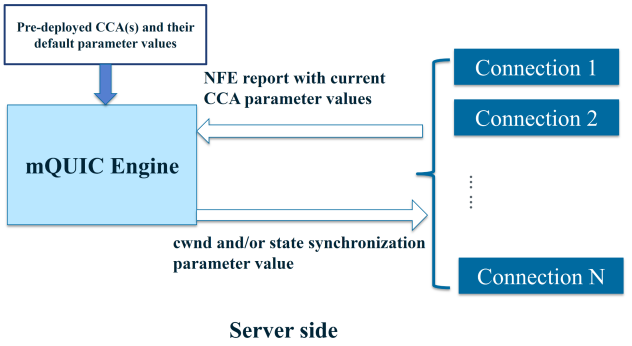


FIGURE 4. Overview of mQUIC scheme.

TABLE 4. Notation list.

Notation	Description
γ_i	The aggregated congestion window growth rate of the multi-connection system after i^{th} NFE occurs
γ_0	The default growth rate of congestion window at SS phase
β	The default back-off ratio of congestion window after the detection of NFE
δ	The default growth ratio of congestion window defined at CA phase
IW	The default Initial Congestion window
cwnd_n	The real-time congestion window on n^{th} connection
$\text{CWND}_{i,j}$	The aggregated congestion window of the multi-connection system when j^{th} RTT has passed since i^{th} NFE occurs
$\text{CWND}'_{i,0}$	The adjusted aggregated congestion window in reaction to i^{th} NFE

the SS phase, sustaining the exponential growth of congestion window. However, barely applying multiple connections still has two unavoidable disadvantages. First, compared to other connections staying in the SS phase, the affected connections which stay in the CA phase will only hold a minor congestion window growth rate. Consequently, these affected connections will lead to state and congestion window imbalance across all the connections and thus become the bottleneck of the eventual downloading time. Second, aggregating more initial congestion windows (e.g. from IW to $N * IW$) without the concern of network condition may fail to guarantee the friendliness to other users and it may increase the packet loss rate in the network that even leads to ultimately worse performances [43].

In order to avoid these above-mentioned disadvantages, the design principle of our mQUIC scheme is based on separating the network resource of the single connection to multiple coordinated connections, each of which independently runs the plug-and-play CCA and the mQUIC engine intelligently orchestrates these underlying connections by executing customized congestion window and state synchronization strategy in a NFE-driven manner.

Figure 4 illustrates the design principle of the mQUIC scheme. The mQUIC engine is typically embedded at the QUIC server side where one of different CCAs (e.g. loss-based or rate-based) is applied at transport layer. According to the running CCA type, a specific connection management algorithm is loaded with default parameters that can be further adapted upon actual incoming web content request according to the specific network condition. During the content transmission period, any real-time information on the actually occurred NFE will be specifically reported from each underlying connection back to the mQUIC engine. By executing the loaded connection management algorithm, the mQUIC engine will compute the optimized parameters and in real-time enforce them in the underlying connections to synchronize the CCA state and congestion window.

The mQUIC engine can be (re-) configured in an offline manner, for instance upon server bootstrap or the change of CCA deployment or upgrading of a CCA. According to the running CCA type at server side, it will load the corresponding connection management algorithm as well as the default CCA parameters (e.g. IW, γ_0, β, δ, see in Table 4). For instance, if the running CCA at server side is New Reno, then the mQUIC engine will load the pre-deployed connection management algorithm for loss-based CCAs. After that, once the N connections are established between server and client, to maintain the friendliness to other users by avoiding an aggregation of excessive initial congestion window, the mQUIC engine will calculate IW/N and re-set it to each underlying connection. Here we propose that the default connection number between a client and a server with mQUIC scheme enabled is 3, and we will discuss the impact of different connection numbers from both the numerical and experimental perspectives in the next subsection and section V.

During the SS phase of each particular web content downloading session (see in Fig. 5a), when the server starts to send the content, the mQUIC engine will monitor the underlying connections and executes a synchronization of not only the congestion window but also the CCA state upon any NFE reported from any underlying connection. For instance, according to the default behavior of hybrid slow start and New Reno algorithms, if any connection detects that the RTT variation exceeds a pre-defined threshold or identifies a random loss event, the SS phase will be immediately terminated and the congestion window will be reduced. Then, the affected connection will report the occurrence of the detected NFE to mQUIC engine with its current congestion window embedded. After that, the mQUIC engine will execute the loaded state and congestion window algorithm immediately. For instance, the customized state and congestion window synchronization algorithm for New Reno relies on calculating the per-round update of γ_i which denotes the aggregated congestion window growth rate of the multi-connection system after i^{th} NFE (see in Table 4). To calculate γ_i per each NFE, the real-time congestion window from the underlying connections will be used as inputs. If the calculated γ_i is larger than default congestion growth rate δ defined in CA phase of the running CCA, mQUIC engine continues to calculate a new $\text{CWND}'_{i,0}$ (see in Table 4) which represents the immediately (0 RTT has elapsed) adjusted aggregated congestion window of the whole multi-connection system once the i^{th} NFE occurred. In the next step, to synchronize

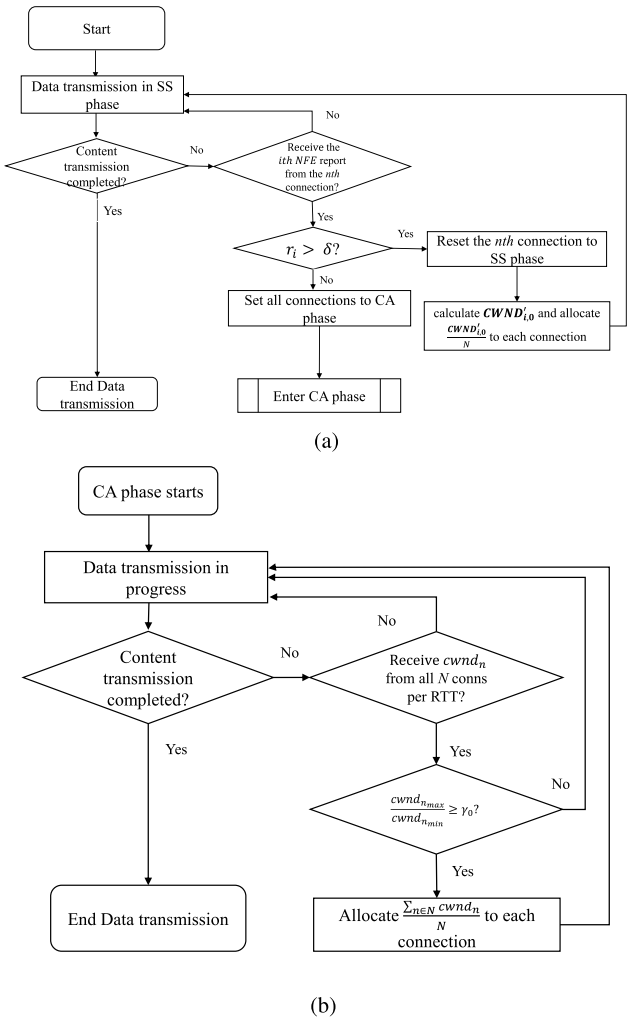


FIGURE 5. Working procedure of mQUIC scheme (Loss-based CCA plugged). (a) SS phase. (b) CA phase.

the congestion window and state, the mQUIC engine equally allocates $\frac{\text{CWND}'_{i,0}}{\text{N}}$ to each connection and re-set the state of the affected connection to SS. As NFE continuously occurs, once the γ_i is less than the default congestion growth rate δ, the mQUIC engine will force all underlying connections to exit SS phase and stop the state synchronization in order to comply with the design principle of the plugged CCA.

After the SS phase, the next working phase of mQUIC engine is the CA phase (see in Fig. 5b). In this phase, since all connections will permanently stay at the CA state, the only potential risk is that the variant RTT or frequent packet loss events will further reduce the congestion window on a random connection, leading to a congestion window imbalance issue. Therefore, the mQUIC engine only executes a congestion window synchronization between the underlying connections. For instance, it will receive the real-time report of congestion window per round from each underlying connection. Once the mQUIC engine detects that within the same round, among the N connections, if the ratio of maximum congestion window to the minimum congestion window exceeding a threshold γ_0, the mQUIC engine will add up the real-time congestion window cwnd_n (see in Table 4) on each connection and equally set $\sum_{n\in \text{N}} \frac{\text{cwnd}_n}{\text{N}}$ to each connection.

B. OPTIMIZING LOSS-BASED CCAs

Then we elaborate the principle of state and congestion window synchronization customized to loss-based CCA. To avoid the state and congestion window imbalance after the reaction of default CCA on underlying connection, the state/cwnd synchronization executed by the mQUIC engine follows two policies: **1) after the occurrence of the i^{th} NFE, all connections should stay in SS phase if theoretically the system's aggregated window growth ratio γ_i is still larger than that defined in CA phase δ, 2) after 1 RTT of the i^{th} NFE, the aggregated congestion window of the whole system with state and cwnd synchronization should be theoretically equal to the congestion window of the whole system without synchronization.** Fig. 6 shows a simple illustration of how the principle of state and congestion window synchronization works. Assuming a total number of N connections sharing the congestion window CWND before the first NFE occurs, each connection sharing a common bottleneck path that holds an equal congestion window $\frac{\text{CWND}}{\text{N}}$.

Once a NFE occurs on a random connection, by default the affected connection will immediately execute a congestion window reduction with a back-off ratio β, then the congestion window of the whole system will becomes $\text{CWND}_{1,0}$ (see in Table 4) as

$$\text{CWND}_{1,0} = \frac{1}{\text{N}} * \text{CWND} * \beta + \frac{\text{N}-1}{\text{N}} * \text{CWND}. \quad (1)$$

After 1 RTT, different from TCP which will stay in HoL blocking state if packet loss occurs, QUIC can seamlessly send the rest of the packets with a growth rate at δ, thus the corresponding aggregated congestion window after 1 RTT will become $\text{CWND}_{1,1}$ as

$$\text{CWND}_{1,1} = \frac{1}{\text{N}} * \text{CWND} * \beta * \delta + \frac{\text{N}-1}{\text{N}} * \text{CWND} * \gamma_0. \quad (2)$$

Therefore, the growth rate of the whole system after the first NFE can be obtained as

$$\gamma_1 = \frac{\text{CWND}_{1,1}}{\text{CWND}_{1,0}}. \quad (3)$$

Recall that the customized state synchronization strategy which targets to retain the equal $\text{CWND}_{1,1}$, if all connections are expected to stay in SS phase after first NFE with default growth rate γ_0, the congestion window must be adjusted to $\text{CWND}'_{1,0}$ which satisfies

$$\text{CWND}'_{1,0} * \gamma_0 = \text{CWND}_{1,0} * \gamma_1. \quad (4)$$

This equation aims to guarantee that the synchronization of mQUIC engine does not change the aggregated congestion window when 1 RTT has passed since the first NFE occurs.

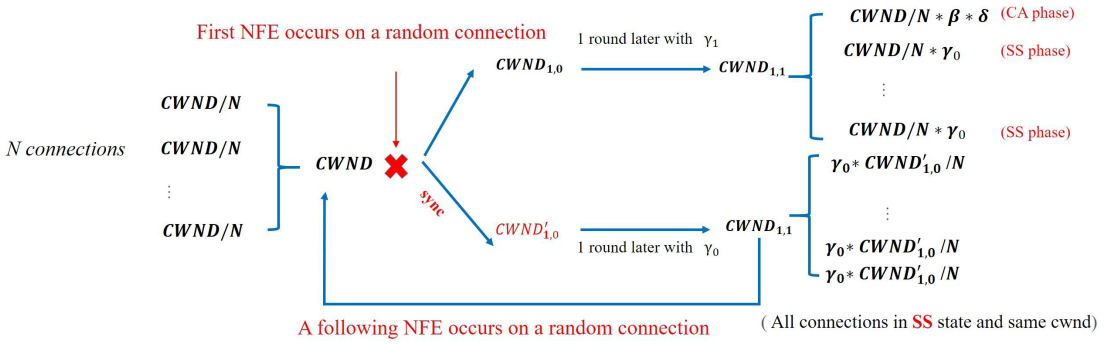


FIGURE 6. Principle of state and congestion window synchronization.

Furthermore, once the multi-connection system is synchronized, any following NFEs can be tackled following the same operation. This is because before the occurrence of each following NFE, the system is always synchronized with the help of mQUIC engine. Accordingly, in such case, the adjusted congestion window after each NFE should satisfy

$$\mathrm{CWND}'_{i,0} * \gamma_0 = \mathrm{CWND}_{i,0} * \gamma_i. \tag{5}$$

Holding

$$\gamma_1 = \frac{\mathrm{CWND}_{1,1}}{\mathrm{CWND}_{1,0}} = \frac{\beta * \delta + (\mathrm{N}-1) * \gamma_0}{\beta + (\mathrm{N}-1)}, \tag{6}$$

consequently, γ_i can be obtained similarly by

$$\gamma_i = \frac{\mathrm{CWND}_{i,1}}{\mathrm{CWND}_{i,0}} = \frac{\beta * \delta + (\mathrm{N}-1) * \gamma_{i-1}}{\beta + (\mathrm{N}-1)}. \tag{7}$$

Thus the corresponding $\mathrm{CWND}'_{i,0}$ is given by

$$\mathrm{CWND}'_{i,0} = \mathrm{CWND}_{i,0} * \frac{\gamma_i}{\gamma_0}, \tag{8}$$

where $\mathrm{CWND}_{i,0}$ can be directly obtained by adding up the congestion window from each underlying connections.

Now we study γ_i, which is the key metric driving the synchronization strategy. The mQUIC engine will stop the state synchronization once $\gamma_i < \delta$. The rationale behind that is if the continued NFEs degrades the aggregated growth rate to a value that is less than the default growth rate in the CA phase, the mQUIC engine should force the system to exit SS phase and enter CA phase.

Figure 7 depicts the numerical trend of γ_i/γ_0 as NFE continuously occurs. It is worth mentioning that, on a single connection, by default all loss-based CCAs like CUBIC and New Reno can correctly identify multiple consecutive packet losses as a single random loss event. When calculating the γ_i, the mQUIC engine treats both the RTT variation and packet loss event detected by the underlying connection as same signal of network fluctuation. This is because prematurely existing the exponential growth rate is the main cause of performance degradation of short-lived application, which is a common consequence of both the detected RTT variation and packet loss event. Therefore, the mQUIC engine uniformly

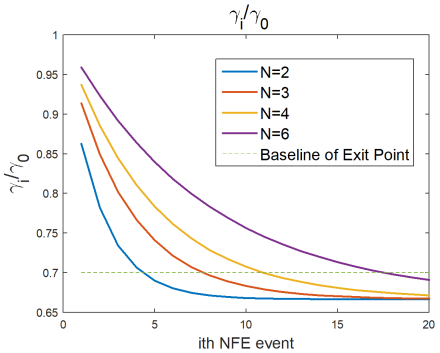


FIGURE 7. γ_i/γ_0 when continuous NFEs occurs and the connection number varies.

adopts $\beta = 0.7$, $\gamma_0 = 1.5$ and $\delta = 1.05$ for these two kinds of NFEs in New Reno and CUBIC. When mQUIC engine should force all connections to exit the SS phase, the ratio of γ_i and γ_0 equals to $\delta/\gamma_0 = 0.7$ (see the green dash line in Fig. 7). The value of γ_i/γ_0 rapidly falls as the first several NFEs occur and then stays steady at around the ratio of 0.65. Furthermore, in terms of different connection numbers, it is apparent that the smaller the connection number is, the earlier the system will exit the SS phase. For instance, when the fifth NFE happens, a 2-connection system will exit SS phase since $\gamma_5/\gamma_0 = 0.69 < 0.7$ ($\gamma_5 = 1.04$ accordingly). In contrast, a 6-connection system can resist more NFEs ($\gamma_{18}/\gamma_0 = 0.69 < 0.7$) but it comes at the expense of more connections and sockets resources.

C. OPTIMIZING RATE-BASED CCAs

The recently proposed rate-based BBR congestion control algorithm becomes an alternative option for QUIC's practical deployment. According to our findings reported in Section III, by retaining the exponential growth under random loss and RTT variation with the help of per-round rate estimation, BBR's bottleneck for short-lived content is the rate stall when the connection temporarily stays at the loss-recovery state upon a random loss event. Theoretically, this rate stall can also be mitigated by applying multiple connections. Assuming one random loss occurs on one of the

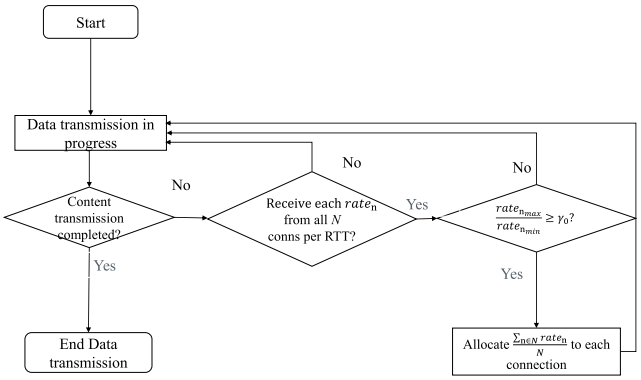


FIGURE 8. Working procedure of mQUIC scheme (Rate-based CCA plugged).

N connections, only the affected connection will temporarily enter loss-recovery phase while other connections can proceed to ramp up the available bandwidth at the SS phase. Compare to the default setting of loss-based CCA which to the maximum degree allows a congestion window increment at 1.5 times, BBR applies a more aggressive exponential rate growth at $2/\ln 2$ and in practice [44] the congestion window approximately doubles per round. Given this improved mechanism, the temporarily rate loss can also be partially absorbed by other independent and parallel connections. For instance, without applying multiple connections, if one random connection experiences this temporal rate stall at i^{th} round and its corresponding rate when entering loss-recovery is $rate_i$, after one round, the rate $rate_{i+1}$ will stay at the same, i.e. $rate_{i+1} = rate_i$. In contrast, theoretically without this NFE, the rate $rate_{i+1}$ is expected to be $2 * rate_i$, thus the rate loss caused by the stall period is $2 * rate_i - rate_i = rate_i$. By applying N connections, the aggregated rate when exiting the loss-recovery will be $rate_{i+1} = \frac{N-1}{N} * 2 * rate_i + \frac{1}{N} * rate_i = \frac{2N-1}{N} * rate_i$, thus being able to mitigate the rate loss from $rate_i$ to $(2 - \frac{2N-1}{N}) * rate_i = 1/N * rate_i$. However, following the same reason mentioned above, the rate imbalance issue still exists since the connection affected by random loss and RTT variation will have a comparatively low congestion window than others, although its SS state can be always retained. With that in mind, to strike a balance between performance robustness and practice complexity, the mQUIC engine only adopts a rate synchronization for BBR enabled connection. This is because after exiting the loss-recovery phase, the affected connection will recover the exponential growth thus the state synchronization is no longer required for BBR which does not prematurely exit SS phase.

Figure 8 depicts the flow chart of the algorithm applied to BBR. After equally set $\frac{\mathrm{IW}}{\mathrm{N}}$ to each connection, during the data transmission, each underlying connection reports its estimated rate at previous round to mQUIC engine. Upon receiving all these per-round rate estimations, the mQUIC engine calculates the ratio of maximum rate over the minimum rate $\frac{rate_{n_{max}}}{rate_{n_{min}}}$. Once this ratio exceeds the pre-define threshold γ_0 (2.0 by default), the mQUIC engine will calculate the aggregated rate $\sum_{n\in \mathrm{N}} rate_n$ and then equally allocate $\frac{\sum_{n\in \mathrm{N}} rate_n}{\mathrm{N}}$ to each underlying connection.

TABLE 5. Network settings to emulate different content locations.

Content location	RTT	Loss rate
Domestic, low loss	R: 21.2ms B: 25ms	R:0.09% B: 0.05%
International, low loss	R: 21.2ms B: 150ms	R:0.09% B: 0.05%
Domestic, high loss	R: 21.2ms B: 25ms	R:0.09% B: 1%
International, high loss	R: 21.2ms B: 150ms	R:0.09% B: 1%

V. PERFORMANCE EVALUATIONS ON mQUIC

In order to realistically evaluate the proposed mQUIC scheme with various CCAs, we implemented and tested the mQUIC engine on the basis of the test infrastructure mentioned in Section III. We keep all QUIC and CCA related parameters as default setting. We use netem to introduce different synthetic RTTs (25ms or 150ms) and different loss rates (0.05% or 1%) at backhaul to emulate the scenarios where content deployed domestically or internationally with a low or high loss rate [31], [39] (see in Table 5). Similar to the methodology in Section III, performance comparison between different approaches are repeated by 200 times in a back-to-back manner. Additionally, for simplicity, for loss-based CCAs we only present New Reno. This is because New Reno is currently adopted in QUIC IETF [45] and according to [46] and our experiments in Section III, the performance difference of short-live content downloading between CUBIC and New Reno is minor.

A. PERFORMANCE EVALUATIONS UNDER VARIOUS NETWORK CONDITIONS

1) CONTENT DEPLOYED AT DOMESTIC (LOW LOSS)

Figure 9 compares the CDFs of completion time of New Reno enabled QUIC with plain 1 connection (1c), plain 3 connections (3c) and 3c with mQUIC for small, medium and large contents, respectively. All the observed values are normalized by the median completion time of 1 connection. In terms of a medium content size (2400KB), mQUIC achieves the best completion time among all the three options, accelerating the median completion time by 37.9% (over 1c) and 23.8% (over 3c), respectively. Meanwhile, substantial gain in 95th percentile of the completion time (67.9% over 1c and 60.9% over 3c) can also be observed, indicating that the vulnerability of short application downloading to fluctuating network conditions is significantly alleviated by the proposed intelligence (with the CoV also dramatically reduces from 0.36 to 0.06). More importantly, these comparisons between 3c without any intelligence and 3c with mQUIC enabled solidly confirm that, the mQUIC scheme with necessary synchronization can effectively solve the issue of imbalance state and congestion window between multiple connections than barely employ multiple connections without any intelligence. Additionally, considering the impact of content sizes, when the content size is small (e.g. 500KB), the main performance

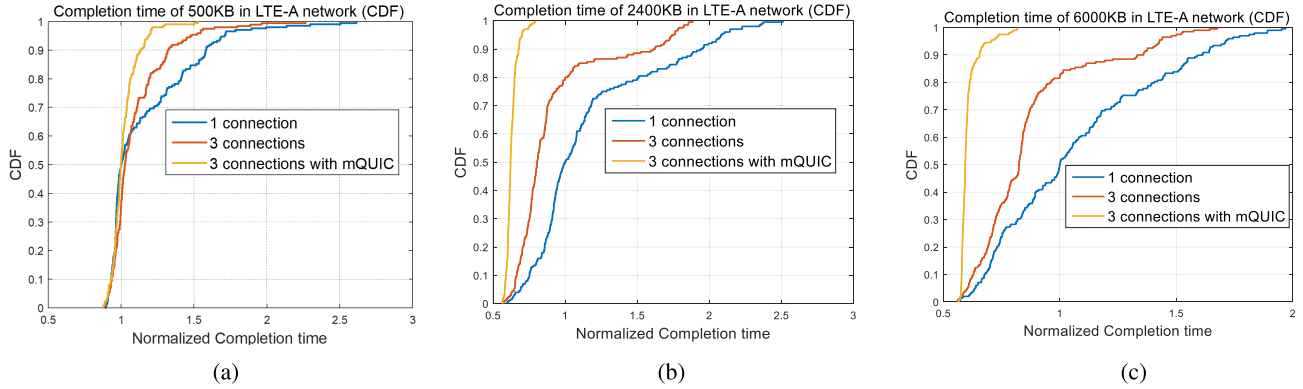


FIGURE 9. Comparison of content completion time between 1 connection, 3 connections and 3 connections with mQUIC (new Reno enabled). (a) Small content: 500KB. (b) Medium content: 2400KB. (c) Large content: 6000KB.

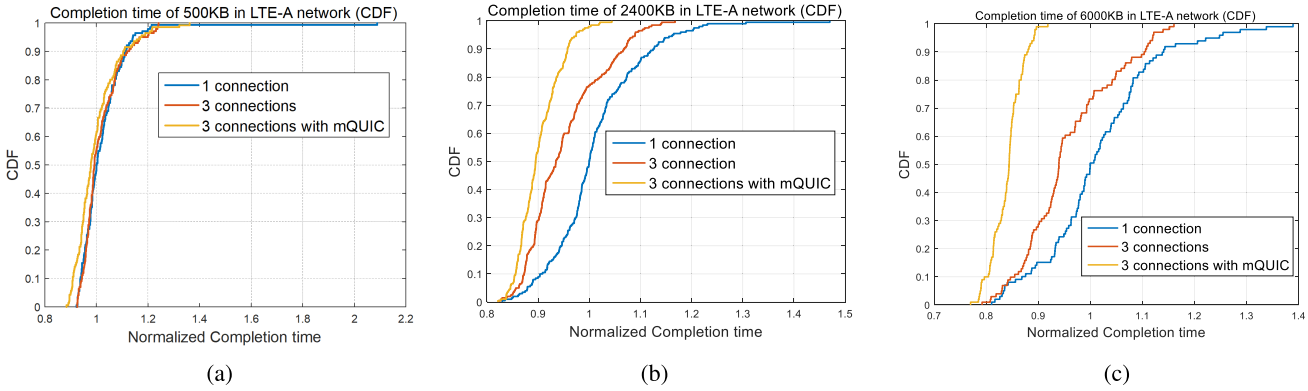


FIGURE 10. Comparison of content completion time between 1 connection, 3 connections and 3 connections with mQUIC (BBR enabled). (a) Small content: 500KB. (b) Medium content: 2400KB. (c) Large content: 6000KB.

gain is the improved 95th completion time (30.5% over 1c and 19.9% over 3c). In contrast, a larger content (e.g. 6000KB) experiences a more significant improvement in both median completion time (40.7% over 1c and 27.9% over 3c) and 95th percentile completion time (65.3% over 1c and 52.4% over 3c). The reason is that a larger content size will experience a longer transmission time, thus will have higher chance to experience more NFEs, resulting in a limited rate for the majority of the content being transmitted. In contrast, smaller content is not sensitive to the reduced rate as it can be downloaded within only several RTTs.

Figure 10 compares the CDFs of completion time of BBR enabled QUIC with 1c, 3c and 3c with mQUIC for small, medium and large contents, respectively. Similar to the previous case, the improvement on small content (500KB) is minor due to its shorter transmission time. In contrast, in terms of medium (2400KB) and large contents (6000KB), the median completion time still experiences a valuable enhancement, presenting at 10.5% and 16.1%, respectively. Moreover, more notable improvements can be observed on the 95th completion time of the medium (16.5%) and large (29.4%) content, which validates that with the help of mQUIC scheme, the robustness of content downloading can be guaranteed (CoV decreases from 0.09 to 0.05 for 2400KB and from 0.10 to 0.04 for 6000KB). Furthermore, compare with plain 3 connections, the main advantage of mQUIC is the improved 95th completion time (e.g. 10.7% for 2400KB and 20.5% for 6000KB) while the benefit for median completion time becomes smaller (e.g. 4.5% for 2400KB and 10.4% for 6000KB).

2) CONTENT DEPLOYED AT VARIOUS LOCATIONS

Tables 6 and 7 list the improvement of median and 95th completion time when comparing 3c with mQUIC with 1c and 3c, considering all the four scenarios, i.e. domestic/international with low/high loss. The enabled CCA is loss-based New Reno. It can be seen from the two tables that 3c with mQUIC in general outperforms 1c in median and 95th completion time across all the scenarios, especially for medium and large contents. However, we observe in Table 6 that in the international with low loss scenario (i.e. long RTT with low loss), the improvement of median completion time is much less than other scenarios. For instance, a 2400KB content only experiences a marginal improvement of median completion time (7.5%), although the 95th completion time improvement is still dramatic (65.1%). This is because, according to the principle of CCA, a sender will take a much longer time to ramp up the bandwidth when RTT is long, where the negative

TABLE 6. Improvement of median completion time when mQUIC enabled on varies network settings (Loss-based CCA).

	500KB		2400KB		6000KB	
Content Location	1c	3c	1c	3c	1c	3c
Domestic low loss	≈0%	≈0%	37.9%	23.8%	40.7%	27.9%
International low loss	≈0%	≈0%	7.5%	4.1%	25.0%	15.9%
Domestic high loss	21.3%	11.4%	51.1%	24.1%	59.8%	26.7%
International high loss	18.4%	9.5%	48.1%	29.8%	55.4%	47.3%

TABLE 7. Improvement of 95th completion time when mQUIC enabled on varies network settings (Loss-based CCA).

	500KB		2400KB		6000KB	
Content location	1c	3c	1c	3c	1c	3c
Domestic Low loss	30.5%	19.9%	67.9%	60.9%	65.3%	52.4%
International Low loss	28.4%	24.2%	65.1%	36.5%	56.5%	24.4%
Domestic High loss	35.5%	25.9%	65.8%	23.2%	58.8%	23.8%
International High loss	32.5%	24.4%	72.3%	46.2%	69.3%	31.8%

TABLE 8. Improvement of median completion time when mQUIC enabled on varies network settings (Rate-based CCA).

	500KB		2400KB		6000KB	
Content location	1c	3c	1c	3c	1c	3c
Domestic Low loss	≈0%	≈0%	10.5%	4.5%	16.1%	10.4%
International Low loss	≈0%	≈0%	≈0%	≈0%	20.1%	≈0%
Domestic High loss	≈0%	≈0%	18.2%	7.5%	27.4%	12.4%
International High loss	≈0%	≈0%	15.2%	7.8%	25.2%	15.1%

TABLE 9. Improvement of 95th completion time when mQUIC enabled on varies network settings (Rate-based CCA).

	500KB		2400KB		6000KB	
Content location	1c	3c	1c	3c	1c	3c
Domestic Low loss	≈0%	≈0%	16.5%	10.7%	29.4%	20.5%
International Low loss	≈0%	≈0%	≈0%	≈0%	20.2%	21.2%
Domestic High loss	≈0%	≈0%	21.2%	17.3%	31.4%	32.2%
International High loss	≈0%	≈0%	21.1%	18.4%	30.3%	29.3%

effect of each NFE is partially absorbed by this comparatively longer and more stable RTT at backhaul. In contrast, the congestion window in a higher loss rate scenario (e.g. 1% at backhaul) is severely constrained by the frequent packet loss event, thus RTT in this case is not the main bottleneck for the eventual downloading time. Meanwhile, comparing with plain 3c, mQUIC also widely helps to improve both the median and 95th completion time, no matter where the content is deployed. Furthermore, regarding the impact of content size, more noticeable improvement can be achieved for large content in both median and 95th completion time across all four scenarios due to its comparatively longer transfer time.

Tables 8 and 9 list the improvement of median and 95th completion time of 3c with mQUIC over 1c and 3c by considering all the four scenarios on top of BBR. These results show that for BBR, the main contribution of mQUIC is that it helps medium and large content to retain a robust performance through various network conditions (e.g. 16.5% to 31.4% reduction in 95th completion time). Meanwhile, in terms of the median completion time, a large content (6000KB) can experience a more notable improvement which ranges from 16.1% to 27.4%. Obviously here packet loss plays a critical role impacting on BBR's performance since more packets are expected to be randomly lost (e.g. 1% loss rate at backhual), the duration that one connection stays at loss recovery state will becomes longer, limiting the aggressively exponential speeding up of the congestion window. However, unlike loss-based CCA which experiences a more dramatically improvement when backhaul loss rate increases from 0.05% to 1%, the improvement on BBR only increases from 16.1% to 27.4%. This is due to the per-round rate-based estimation, the congestion window of BBR does not perform a back-off when packet loss occurs, maximally absorbing the performance degradation when packet loss surges. Additionally, a glance at the performance improvement over plain 3c without any mQUIC intelligence also reveals that the rate synchronization algorithm can still help to improve the 95th completion time at a valuable level, although that improvement on median completion time becomes less significant due to connection separations.

TABLE 10. Measured network performance for different radio access conditions.

Access type/condition	RTT	Loss rate	Bandwidth
LTE-A RSRP -97dbm	R: 21.2ms B: 25ms	R: 0.11% B:0.05%	35.4Mbps
LTE-A RSRP -76dbm	R: 21.2ms B: 25ms	R: 0.09% B:0.05%	65.9Mbps
LTE-A RSRP -65dbm	R: 21.2ms B: 25ms	R: 0.04% B:0.05%	94.1Mbps
WiFi 802.11n	R: 10.5ms B: 25ms	R: 0.13% B:0.05%	190.2Mbps

3) IMPACT OF DIFFERENT RADIO CONDITIONS

After examining the impact of backhaul network conditions, we switch the position of the mobile device within the cell coverage to set the RSRP as poor (−97 dBm), good (−76 dBm) and very good (−65 dBm). The content location is fixed at domestic low loss scenario. Additionally, we configure Wi-Fi 802.11n as another access type which offers higher bandwidth but more unstable RTTs (e.g. the standard deviation of RTT is 6.1ms) at radio interface. Table 10 lists the measured network condition and the median and 95th completion time of a medium content are depicted in Figs. 11 and 12, respectively. It is obvious that, for loss-based CCA New Reno, if the available bandwidth increases (e.g. good RSRP or Wi-Fi), the improvement of the median and 95th completion time becomes more significant and vice versa. For instance, the gain of median completion time increases from 25.1% to 37.9% when RSRP rises from −97 dBm to −76 dBm and stays at 44.1% in Wi-Fi scenario. Similarly, the improvement of median completion time for rate-based CCA BBR increases from 1.2% (RSRP −97 dBm) to 10.5% (RSRP −76 dBm) and then stays at 20.1% in Wi-Fi scenario. This trend can be attributed to that, in a relatively poor RSRP scenario where the available bandwidth is smaller, the network pipe can be rapidly saturated by the

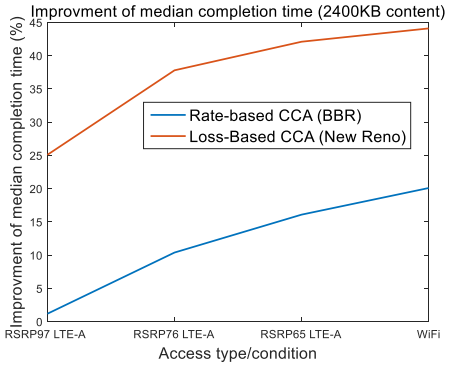


FIGURE 11. Improvement of median completion time when mQUIC enabled on varies network settings.

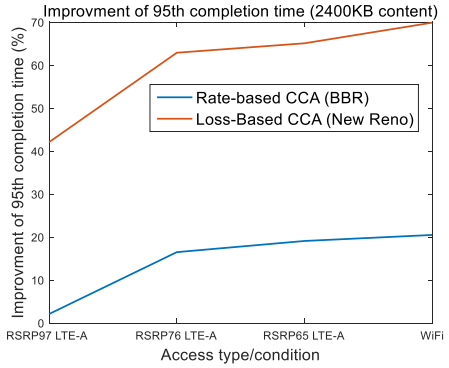


FIGURE 12. Improvement of 95th completion time when mQUIC enabled on varies network settings.

SS algorithm, thus the negative effect of NFE on short-lived content becomes smaller.

B. PERFORMANCE EVALUATIONS UNDER VARIOUS PROTOCOL/CONTENT SETTINGS

In addition to the above experiments under various network configurations, we continue to conduct extensive evaluations by varying several protocol settings like synchronization strategy, connection number and initial congestion window. For the simplicity of presentation, we only adopt the default loss-based CCA in a domestic, low backhaul loss scenario, which accounts for the majority of today's web content access scenarios.

1) COMPARISON WITH CWND SYNCHRONIZATION ONLY (LOSS-BASED CCA)

Recall that the customized strategy for loss-based CCA, state synchronization is a distinctive optimization algorithm. It helps all the connections to stay at SS phase or consistently enter the CA phase, which retains a reasonable congestion window growth rate after the occurrence of each NFE. To validate its individual benefit, we compare the performance between mQUIC with both state and congestion window synchronization and only congestion window synchronization with various content sizes that deployed in the domestic, low loss scenario. Table 11 shows the performance gain originated

TABLE 11. Completion time comparison between 3 connections with mQUIC and 3 connections with congestion window synchronization only.

Domestic low loss		
	Median	**95th**
500KB	≈0%	≈0%
2400KB	25.5%	33.2%
6000KB	25.1%	29.1%

by state and congestion window synchronization together over that when only congestion window synchronization is enabled. For medium and larger content, around 25% improvement can be observed on median completion time while the 95th percentile completion time is accelerated by around 30%. This performance gain comes from the restored congestion window growth rate γ_0 between two consecutive NFEs, when the mQUIC engine keeps the whole system at the SS phase. In contrast, synchronizing the congestion window can only eliminate the rate imbalance issue but the growth rate of whole system will recover much slower than the state synchronization.

2) IMPACT OF DIFFERENT CONNECTION NUMBER

Figure 13 shows the CDF of completion time of mQUIC by varying the number of connections from 2 to 6 for different content sizes. It is obvious that across all content sizes, there is no noticeable performance difference between 3 and more connections, while applying only 2 connections still suffers from a comparatively poor 95th completion time. This is because according to our analysis in Section IV, 2 connections will exit SS phase after the occurrence of 5 NFEs while the increased connection number can tolerate more than 8 NFEs which achieves a more robust performance in the testing LTE-A network. In terms of the impact of content size, for a small content size this performance difference caused by connection number is minor, and for medium and large contents, applying 3 connections is sufficient to retain fast and robust downloading time. Another practical concern is the complexity of deploying multiple connections, especially on mobile devices which only offer limited energy and process capabilities. For instance, previous study [47] shows that most mobile device fixed the maximum configurable number of concurrent connections at 4 per server. However, thanks to the effort of the emerging security protocol like TLS 1.3 [48] and QUIC, the simplified encryption and handshake procedure successfully alleviate these additional consumptions. Besides, it is proved in [49] that well engineered parallel connections which bring throughput improvement can yield up to 20% energy saving, eliminating the concern of the overuse of on-device resources. Consequently, the potential scenarios for further investigating a trade-off between performance gain and cost can be that when there is no benefit caused by the applied multiple connections (e.g. downloading small content in a limited bandwidth network). We leave this context-related topic as our future work.

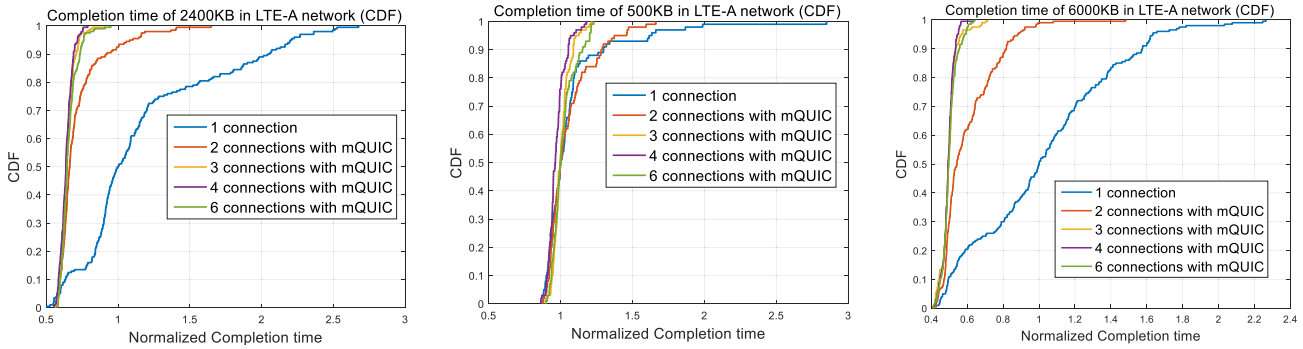


FIGURE 13. CDF of completion time while applying different connection number. (a) Medium content: 2400KB. (b) Small content: 500KB. (c) Large content: 6000KB.

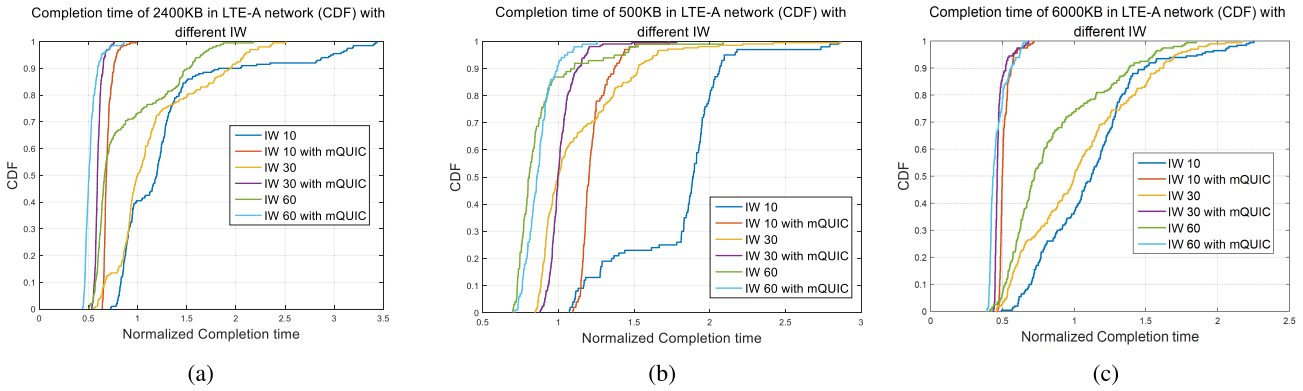


FIGURE 14. CDF of completion time while tuning IW value. (a) Medium content: 2400KB. (b) Small content: 500KB. (c) Large content: 6000KB.

3) IMPACT OF INITIAL CONGESTION WINDOW

Tuning IW is a popular approach to overcome the bandwidth under-utilization issue of short-lived content and it has been widely investigated in recent years [50]. Fig. 14 shows the CDF of completion time when applying our approach with different IW settings as 10 (the standardized value adopted by TCP), 30 (the default value current adopted by QUIC) and 60 (an increased value that expected to achieve more bandwidth utilization). In terms of medium and large contents (see in Figs. 14a, 14c), with the help of mQUIC, fast and robust performances of the content downloading time can be observed for all IW settings. For instance, mQUIC achieves the best performance when IW is 60, but only minor gap can be observed among the three IW settings. Whereas content downloading without mQUIC in general suffers from poor and high variant downloading time regardless of the IW sizes, although a larger IW 60 has better median completion time than IW 30 and IW 10. This is because, according to prior model analysis in [50] and [51], tuning IW favors a smaller content or a larger network Bandwidth-Delay Production (BDP), while for a larger content in a short RTT scenario, its expected benefit becomes less significant. According to this analysis, in Fig. 14b, for a small content, it can be seen that when IW is large ($IW = 60$), its median completion time can be significantly reduced whereas the benefit of enabling mQUIC can be only observed on 95th completion time. In contrast, this benefit of mQUIC becomes more significant if the IW is small ($IW = 10$), indicating that a combination of tuning IW and enabling mQUIC is more effective for smaller content while enabling mQUIC only is sufficient for median and large content to guarantee an accelerated downloading time.

C. MANIPULATING HTTP/2 MESSAGES WITH mQUIC

To realistically support HTTP/2 messages, there is a practical concern that whether to request single object on each connection or to separate single HTTP object request to multiple HTTP byte-range requests on multiple connections. The answer is that it depends on the application type in practice. For webpage browsing, only the critical objects which have dependency on its following objects [52] need to be requested simultaneously on multiple connections in a byte-range manner. This is because this kind of objects have higher priority, thus the aggregated rate on all connections can guarantee its prior delivery. Regarding the overhead caused by separating one HTTP request to multiple byte-range requests, since the number of critical objects only accounts for a minor part of the total objects in a webpage [52], only minor overhead will be added by enabling mQUIC on web content. In terms of single file application like short video or software downloading,

it is obvious that the additional overhead caused by separating a single request to a large object (e.g. several MBs) is trivial.

VI. CONCLUSION

In this paper, we proposed an mQUIC scheme that employs multiple connections instead of the default single UDP connection adopted by QUIC furthering order to comprehensively enhance the performance of web content with QUIC in mobile networks. By performing intelligent state and congestion window synchronization, this mQUIC scheme is capable of mitigating the unexpected hindrance of congestion window growth caused by the CCA on the single connection upon the occurrence of a NFE (like RTT variations and random loss events). The evaluation results based on a real implementation of the mQUIC scheme in a local LTE-A/Wi-Fi testing network reveal that the customized approaches effectively help each CCA to overcome its individual limitations under network fluctuation, thus attaining a substantially improved content downloading time. In detail, for loss-based CCAs like New Reno, the median completion time of a piece of web content can be improved up to 59.1% and the 95th percentile completion time is improved by up to 72.3%. Regarding the latest rate-based CCA like BBR, mQUIC successfully speeds up the median and 95th downloading time up to 27.4% and 31.4%, respectively. Furthermore, fast content delivery can be also guaranteed when the network condition or initial congestion window varies.

ACKNOWLEDGEMENT

The authors would also like to acknowledge the support of the University of Surrey's 5G Innovation Centre (5GIC) (http://www.surrey.ac.uk/5gic) members for this work.

REFERENCES

[1] T. Han, N. Ansari, M. Wu, and H. Yu, "On accelerating content delivery in mobile networks," *IEEE Commun. Surveys Tuts.*, vol. 15, no. 3, pp. 1314–1333, 3rd Quart., 2013.

[2] N. Cardwell, Y. Cheng, C. S. Gunn, S. H. Yeganeh, and V. Jacobson, "BBR: congestion-based congestion control," *Commun. ACM*, vol. 60, no. 2, pp. 58–66, 2017.

[3] M. Belshe, R. Peon, and M. Thomson, *Hypertext Transfer Protocol Version 2 (HTTP/2)*, document RFC 7540, 2015.

[4] QUIC Project. *QUIC, A Multiplexed Stream Transport Over UDP*. Accessed: Dec. 2017. [Online]. Available: https://www.chromium.org/quic

[5] X. S. Wang, A. Balasubramanian, A. Krishnamurthy, and D. Wetherall, "How speedy is SPDY?" in *Proc. USENIX Conf. Networked Syst. Design Implement.*, 2014, p. 1.

[6] J. Huang, F. Qian, A. Gerber, Z. M. Mao, S. Sen, and O. Spatscheck, "A close examination of performance and power characteristics of 4G LTE networks," in *Proc. ACM MobiSys*, 2012, pp. 225–238.

[7] J. Huang *et al.*, "An in-depth study of LTE: effect of network protocol and application behavior on performance," in *Proc. ACM SIGCOMM*, 2013, pp. 363–374.

[8] M. B. Albaladejo, D. J. Leith, and P. Manzoni, "Measurement-based modelling of LTE performance in Dublin city," in *Proc. IEEE PIMRC*, Sep. 2016, pp. 1–6.

[9] P. Biswal and O. Gnawali, "Does QUIC make the Web faster?" in *Proc. IEEE GLOBECOM*, Dec. 2016, pp. 1–6.

[10] P. Megyesi, Z. Krämer, and S. Molnár, "How quick is QUIC?" in *Proc. IEEE ICC*, May 2016, pp. 1–6.

[11] G. Carlucci, L. De Cicco, and S. Mascolo, "HTTP over UDP: An experimental investigation of QUIC," in *Proc. ACM SIGAPP*, 2015, pp. 609–614.

[12] E. Atxutegi, F. Liberal, K.-J. Grinnemo, A. Brunstrom, Å. Arvidsson, and R. Robert, "TCP behaviour in LTE: Impact of flow start-up and mobility," in *Proc. IEEE IFIP WMNC*, Jul. 2016, pp. 73–80.

[13] S. Ha and I. Rhee, "Hybrid slow start for high-bandwidth and long-distance networks," in *Proc. PFLDnet*, 2008 pp. 1–6.

[14] *BBR vs CUBIC Test on Mobile Wireless Path*. Accessed: Dec. 2017. [Online]. Available: https://groups.google.com/forum/topic/bbr-dev/yR-g-zHEF6w

[15] F. Qian, V. Gopalakrishnan, E. Halepovic, S. Sen, and O. Spatscheck, "TM3: Flexible transport-layer multi-pipe multiplexing middlebox without head-of-line blocking," in *Proc. ACM CoNEXT*, 2015, p. 3.

[16] P. Qian, N. Wang, G. Foster, and R. Tafazolli, "Enabling context-aware HTTP with mobile edge hint," in *Proc. IEEE CCNC*, Jan. 2017, pp. 420–426.

[17] A. Langley *et al.*, "The QUIC transport protocol: Design and internet-scale deployment," in *Proc. ACM SIGCOMM*, 2017, pp. 183–196.

[18] P. Yang, J. Shao, W. Luo, L. Xu, J. Deogun, and Y. Lu, "TCP congestion avoidance algorithm identification," *IEEE/ACM Trans. Netw.*, vol. 22, no. 4, pp. 1311–1324, Aug. 2014.

[19] S. Cook, B. Mathieu, P. Truong, and I. Hamchaoui, "QUIC: Better for what and for whom?" in *Proc. ICC*, May 2017, pp. 1–6.

[20] T. Zinner, S. Geissler, F. Helmschrott, S. Spinsante, and A. Braeken, "An AAL-oriented measurement-based evaluation of different HTTP-based data transport protocols," in *Proc. IFIP Netw.*, 2017, pp. 1162–1167.

[21] T. Zinner, S. Geissler, F. Helmschrott, and V. Burger, "Comparison of the initial delay for video playout start for different HTTP-based transport protocols," in *Proc. IFIP Netw.*, 2017, pp. 1027–1030.

[22] D. Bhat, A. Rizk, and M. Zink, "Not so QUIC: A performance study of DASH over QUIC," in *Proc. ACM NOSSDAV*, 2017, pp. 13–18.

[23] K. Spiteri, R. Urgaonkar, and R. K. Sitaraman, "BOLA: Near-optimal bitrate adaptation for online videos," in *Proc. IEEE INFOCOM*, Apr. 2016, pp. 1–9.

[24] C. Wang, A. Rizk, and M. Zink, "SQUAD: A spectrum-based quality adaptation for dynamic adaptive streaming over HTTP," in *Proc. ACM MMSys*, 2016, p. 1.

[25] T.-Y. Huang, R. Johari, N. McKeown, M. Trunnell, and M. Watson, "A buffer-based approach to rate adaptation: Evidence from a large video streaming service," in *Proc. ACM SIGCOMM*, 2014, pp. 187–198.

[26] E. Altman, D. Barman, B. Tuffin, and M. Vojnovic, "Parallel TCP sockets: Simple model, throughput and validation," in *Proc. IEEE INFOCOM*, Apr. 2006, pp. 1–12.

[27] Q. Fu, "Improving throughput in high bandwidth-delay product networks with random packet losses," in *Proc. IEEE ICC*, Jun. 2009, pp. 1–6.

[28] E. Yildirim, D. Yin, and T. Kosar, "Prediction of optimal parallelism level in wide area data transfers," *IEEE Trans. Parallel Distrib. Syst.*, vol. 22, no. 12, pp. 2033–2045, Dec. 2011.

[29] J. Kim, Y.-C. Chen, R. Khalili, D. Towsley, and A. Feldmann, "Multi-source multipath HTTP (mHTTP): A proposal," in *Proc. ACM SIGMETRICS*, 2014, pp. 583–584.

[30] D. Kaspar, K. Evensen, P. Engelstad, and A. F. Hansen, "Using HTTP pipelining to improve progressive download over multiple heterogeneous interfaces," in *Proc. IEEE ICC*, May 2010, pp. 1–5.

[31] B. Han, F. Qian, S. Hao, and L. Ji, "An anatomy of mobile web performance over multipath TCP," in *Proc. ACM CoNEXT*, 2015, p. 5.

[32] X. Xu, Y. Jiang, T. Flach, E. Katz-Bassett, D. Choffnes, and R. Govindan, "Investigating transparent web proxies in cellular networks," in *Passive and Active Measurement*. New York, NY, USA: Springer, 2015, pp. 262–276.

[33] A. Sehati and M. Ghaderi, "WebPro: A proxy-based approach for low latency web browsing on mobile devices," in *Proc. IEEE IWQoS*, Jun. 2015, pp. 319–328.

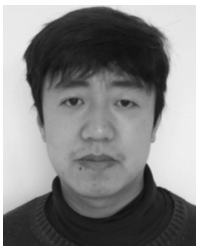
[34] N. Becker, A. Rizk, and M. Fidler, "A measurement study on the application-level performance of LTE," in *Proc. IFIP Netw.*, Jun. 2014, pp. 1–9.

[35] C. Ge, N. Wang, G. Foster, and M. Wilson, "Toward QoE-assured 4K video-on-demand delivery through mobile edge virtualization with adaptive prefetching," *IEEE Trans. Multimedia*, vol. 19, no. 10, pp. 2222–2237, Oct. 2017.

[36] *Huawei Lampsite*. Accessed: Dec. 2017. [Online]. Available: http://carrier.huawei.com/en/products/wireless-network/Small-Cell/LampSite

[37] *HTTP Archive*. Accessed: Dec. 2017. [Online]. Available: http://httparchive.org/

[38] *NetEm*. Accessed: Dec. 2017. [Online]. Available: http://manpages.ubuntu.com/manpages/wily/man8/tc-netem.8.html
[39] *Sprint SLA Performance*. Accessed: Dec. 2017. [Online]. Available: https://www.sprint.net/sla_performance.php
[40] N. Cardwell *et al.*, "BBR congestion control," in *Proc. IETF Meeting*, 2016, pp. 1–37. [Online]. Available: https://www.ietf.org/proceedings/97/slides/slides-97-iccrg-bbr-congestion-control-02.pdf
[41] *BBR Congestion Control*. Accessed: Dec. 2017. [Online]. Available: https://tools.ietf.org/html/draft-cardwell-iccrg-bbr-congestion-control-00
[42] D. Baltrunas, A. Elmokashfi, A. Kvalbein, and O. Alay, "Investigating packet loss in mobile broadband networks under mobility," in *Proc. IFIP Netw.*, 2016, pp. 225–233.
[43] R. Barik and D. M. Divakaran, "Evolution of TCP's initial window size," in *Proc. IEEE LCN*, Oct. 2013, pp. 500–508.
[44] N. Cardwell, Y. Cheng, C. S. Gunn, S. H. Yeganeh, and V. Jacobson, "BBR congestion control: An update," in *Proc. IETF Meeting*, 2017, pp. 1–37. [Online]. Available: https://www.ietf.org/proceedings/98/slides/slides-98-iccrg-an-update-on-bbr-congestion-control-00.pdf
[45] *QUIC Loss Detection and Congestion Control*. Accessed: Dec. 2017. [Online]. Available; https://tools.ietf.org/html/draft-ietf-quic-recovery-08
[46] *QUIC Experiments*. Accessed: Dec. 2017. [Online]. Available: https://datatracker.ietf.org/meeting/93/materials/slides-93-hopsrg-6/
[47] J. Huang, Q. Xu, B. Tiwana, Z. M. Mao, M. Zhang, and P. Bahl, "Anatomizing application performance differences on smartphones," in *Proc. ACM MobiSys*, 2010, pp. 165–178.
[48] *Example Handshake Traces for TLS 1.3*. Accessed: Dec. 2017. [Online]. Available: https://tools.ietf.org/html/draft-ietf-tls-tls13-vectors-02
[49] J. K. Nurminen, "Parallel connections and their effect on the battery consumption of a mobile phone," in *Proc. IEEE CCNC*, Jan. 2010, pp. 1–5.
[50] M. Scharf, "Comparison of end-to-end and network-supported fast startup congestion control schemes," *Comput. Netw.*, vol. 55, no. 8, pp. 1921–1940, 2011.
[51] N. Cardwell, S. Savage, and T. Anderson, "Modeling TCP latency," in *Proc. IEEE INFOCOM*, Mar. 2000, pp. 1742–1751.
[52] J. Vesuna, C. Scott, M. Buettner, M. Piatek, A. Krishnamurthy, and S. Shenker, "Caching doesn't improve mobile Web performance (much)," in *Proc. USENIX ATC*, 2016, pp. 159–165.



PENG QIAN received the B.Eng. degree from the Nanjing University of Posts And Telecommunications, Nanjing, China, in 2008, and the M.Sc. degree (Hons.) from the University of Surrey in 2013, where he is currently pursuing the Ph.D. degree with the Institute for Communication Systems. He was involved in the field of telecommunication industry from 2008 to 2012. His research interests include Web content acceleration, mobile edge computing, and next-generation Internet protocol.

NING WANG received the B.Eng. degree (Hons.) from the Changchun University of Science and Technology, Changchun, China, in 1996, the M.Eng. degree from Nanyang University, Singapore, in 2000, and the Ph.D. degree from the University of Surrey, U.K., in 2004, respectively. He is currently a Reader (an Associate Professor) with the 5G Innovation Centre, Institute for Communication Systems, University of Surrey. His research interests mainly include information centric networking, content and data caching management, network optimisation techniques, and quality of service.

RAHIM TAFAZOLLI has over 25 years of experience in digital communications research and teaching. He is currently a Professor of mobile and satellite communications, Director of the Institute for Communication Systems, and the Founder and the Director of the 5G Innovation Centre with the University of Surrey, U.K. He is regularly invited to deliver keynote talks and distinguished lectures to international conferences and workshops. He is regularly invited by many governments for advise on 5G technologies. He has authored or co-authored over 500 research publications. He holds over 30 granted patents, all in the field of digital communications. He is a fellow of the Wireless World Research Forum in recognition of his personal contributions to the wireless world and the Head of one of Europe's leading research groups.