electronics

MDPI

Article

Access Latency Reduction in the QUIC Protocol Based on Communication History

Jinhwa Jung [1] and Donghyeok An [2,*]

[1] System Development Department, DSME Information & Consulting, Geoje-si, Gyeongsangnam-do 53302, Korea; jinhwajung93@gmail.com

[2] Department of Computer Engineering, Changwon National University, Changwon-si, Gyeongsangnam-do 51140, Korea

* Correspondence: donghyeokan@changwon.ac.kr

Received: 18 September 2019; Accepted: 18 October 2019; Published: 22 October 2019

Abstract: Internet traffic is experiencing rapid growth, with the majority of traffic generated from video steaming, web data services and Internet of Things. As these services include the transmission of small data, such as web pages, video chunks and sensing data, data latency affects the quality of experience rather than the throughput. Therefore, this study aims to decrease latency to improve the quality of the user experience. To this end, we measure the web service delay and throughput in mobile networks. The results indicate a low quality experience for mobile users, even though mobile networks support a large throughput. We therefore propose a light-weight latency reduction scheme for the Quick UDP Internet Connections (QUIC) protocol. The proposed scheme calculates the average congestion window, which is utilized as the initial congestion window when a new connection is established. The proposed scheme is evaluated through experiments on a testbed. The results show that our scheme reduces latency significantly. The results of this study can help improve user experiences of video streaming and web data services.

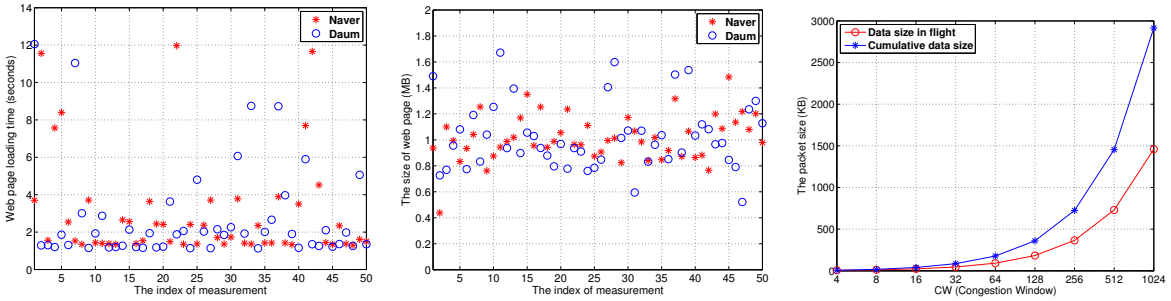
Keywords: access latency; QUIC; communication history

1. Introduction

With the wide deployment of smart devices and the advancement of computing and communication technology, Internet traffic is experiencing explosive growth. By 2022, Internet traffic is expected to be three times that of 2017 [1]. In 2017, 75% of traffic was generated from video services such as Internet video, Internet Protocol Video on Demand (IP VoD) and video-streamed gaming. Web service and data transmission such as sensing data accounts for 17% of all Internet traffic; therefore, video streaming, web services and data transmission are the most preferred services by Internet users. Traditionally, these services use the Transmission Control Protocol (TCP) as a transport layer, which provides several features such as reliability, in-order delivery, congestion control and flow control. Various types of TCPs have been proposed for throughput improvement, some of which have been developed to increase utilization of the network bandwidth, whereas others have dealt with throughput degradation due to frequent bit errors in wireless networks [2–9].

Despite the throughput increase achieved by previous studies, the latency reduction of TCP is not large. Therefore, video streaming and web and data service users experience low quality. When using video streaming, web services and data transmission, users determine the quality of experience based on the response time. If web page access, sensing data transmission or video buffering are completed quickly, the quality of experience increases. To observe the access latency from the desktop to a web page in a portal, we measured the web page loading time of Naver and Daum, which are famous portals in South Korea. According to the results in Figure 1a, the web page loading times of

Naver and Daum are approximately 3 s and 2.7 s, respectively. However, the average web page size is small (approximately 1 MB in Figure 1b) and the network bandwidth is large (more than 100 Mbps); therefore, the time required to download a web page is slower than expected. This is due to the fact that TCP utilizes three-way handshaking and a slow start in TCP [10,11]. Indeed, as shown in Figure 1c, for a congestion window value of 512, which has a cumulative data size of approximately 1.5 MB, eight Round Trip Time (RTT) are required to download 1 MB of data.



(**a**) Measured web page loading times

(**b**) Measured web page size

(**c**) Transmission data size during the slow start phase

Figure 1. Web page loading times and characteristics of Naver and Daum. (**a**) Measured web page loading times, (**b**) measured web page size and (**c**) transmission data size during the slow start phase.

Therefore, to improve the quality of experience for Internet users, latency reduction is necessary, which has been proposed by several previous studies [10–13]. To rapidly fetch data from a website, the initial congestion window size increases or the data is piggybacked during the three-way handshake and Domain Name System (DNS) lookup. However, these approaches require modification of the kernel, which leads to backward compatibility issues. For fast video streaming, User Datagram Protocols (UDPs) are employed but they also require modification [14,15]. Google developed the Quick UDP Internet Connections (QUIC) protocol to support low access latency [16–18]. As the QUIC protocol uses a UDP for backward compatibility, it can be used without any modification. The QUIC protocol utilizes 1-RTT and 0-RTT connection establishment to decrease the overhead of three-way handshaking. However, latency reduction in the QUIC protocol is limited because it also sets a small initial congestion window and employs slow start. Several studies have been conducted focusing on performance improvement, implementation validation and security [19–28]. Among these studies, some authors have proposed latency reduction schemes for QUIC. In References [21,22], either the number of connections between the client and the server is adjusted or the congestion window is controlled based on object priority. However, as these schemes transmit QUIC packets aggressively without considering the network conditions, congestion in the networks can be aggravated. A partial reliable transmission is used to decrease latency; however, this causes data loss [23,24].

The aim of this study is to reduce access latency in the QUIC protocol. First, because mobile traffic accounted for 52% of IP traffic in 2017 [1], we observe the quality of the web service experience for users of smart devices with commercial mobile networks. Specifically, we measure the throughput and web page loading time in both Long-Term Evolution (LTE) and 3G networks. While the throughput of LTE and 3G networks is approximately 120 Mbps and 5 Mbps, respectively, the gap in web page loading time between LTE and 3G networks is 1–3 s. This implies that users experience low quality in terms of latency, even though the difference in throughput between LTE and 3G networks is very large.

Second, we propose a light-weight latency reduction scheme based on communication history. The proposed scheme sets the initial transmission rate according to the network conditions. If the network environment is good, the initial transmission is high. The proposed scheme determines the larger value of the average congestion window and the default congestion window as the initial

congestion window. To obtain the average congestion window, our scheme calculates the average congestion window under two conditions: when the connection is closed or when the 0-RTT handshake occurs. To avoid a latency increase due to computational overheads, our algorithm has low complexity and the proposed scheme boasts a small computational effort.

Finally, we evaluate the proposed scheme using two experimental topologies with different numbers of desktops. Our scheme achieves significant latency reduction in good and average network environments. Additionally, the standard deviation of latency indicates that our scheme adapts well to network conditions.

The remainder of this paper is organized as follows. Related works are presented in the Section 2. Section 3 describes the measurement results of the throughput and web page loading time in LTE and 3G networks. In the Section 4, we propose a light-weight latency reduction scheme and evaluate our scheme with experiments using a testbed in Section 5. Finally, we present the conclusions in Section 6.

2. Related Work

In TCP, several studies have addressed the impact of changing the initial congestion window size. In Reference [12], Aggregated TCP (ATCP) enabled an intermediate router to divide one connection into two connections. The initial congestion window sizes of both connections were set to the average congestion window size of other connections for decrement of the download time of web page. However, ATCP requires the modification of the intermediate router and the management of multiple connections. In Reference [13], to decrease the latency without the intervention of the intermediate router, a sender used a specific value for the initial congestion window size. However, because the initial congestion window size was fixed, if the network environment such as its bandwidth and RTT was changed, the initial congestion window size became unsuitable and performance gain decreased. In cellular networks, performance varies with different initial congestion window sizes; that is, the performance is proportional to the size [29]. If a delay spike (an unexpected long delay) occurs, as the initial congestion window size increases, the performance decreases owing to the increased queueing delay.

Many studies have addressed latency reduction to provide fast services to users. The TCP Fast Open protocol has been proposed to decrease the latency of short transmissions such as web page access [10]. When a connection is initiated, a client piggybacks data in an SYN packet then a server transmits data to the client. Because modification of this three-way handshake causes security vulnerability, the TCP Fast Open protocol utilizes security cookies to deal with security issues such as a SYN flood attack. In Reference [11], the Accelerated Secure Association Protocol (ASAP) was proposed to reduce access delay by modifying the DNS lookup process and connection establishment process. In ASAP, the DNS query message including data is transmitted from the client to the DNS server, which forwards the DNS query message to the server. Then, after receiving the DNS query with the data request message, the server sends data to the client without a three-way handshake. However, as TCP Fast Open and ASAP protocols require kernel modification, they exhibit low backward compatibility. For efficient video frame transmission, the Complete User Datagram Protocol (CUDP) utilizes a Cyclic Redundancy Check (CRC) to check the UDP header error. Because multimedia applications are resilient to any frame error, the UDP payload is delivered to the application. In Reference [15], Lightweight Secure Streaming Protocol (LSSP) was proposed for a light weight secure streaming service. This inserts authentication data into the UDP header. However, both these protocols require modification of the kernel.

The QUIC protocol proposed by Google is a new transport layer protocol for providing fast access and easy deployment for users that is implemented in a Chrome browser [16–18]. As shown in Figure 2, to support backward compatibility, the QUIC protocol, which is built on UDP, provides encryption, congestion control, flow control and reliability to substitute Transport Layer Security (TLS) and TCP. Therefore, it is widely deployed without kernel modification.

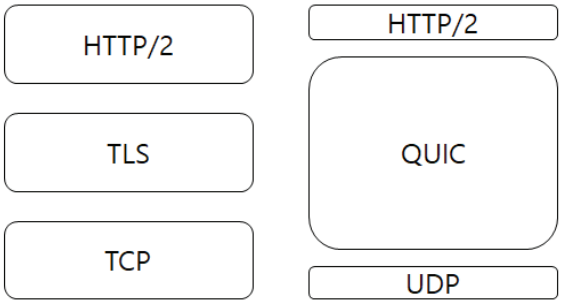


Figure 2. Architecture of the Quick UDP Internet Connections (QUIC) protocol.

The QUIC protocol provides different functionalities for performance enhancement, three of which are discussed here. First, QUIC supports 1-RTT and 0-RTT handshakes to reduce the connection establishment delay as depicted in Figure 3. If the client sets up the first connection to the server, the 1-RTT handshake is used to exchange connection information including the connection identifier and encryption/decryption key. After a successful 1-RTT handshake, the client sends and receives data via the 0-RTT handshake; therefore, the connection establishment delay decreases. If the 0-RTT handshake is not used for a period of time, the connection information is deleted and the connection is closed. Then, the client initializes a new connection using the 1-RTT handshake. Second, the QUIC protocol provides stream multiplexing to deal with head-of-line blocking due to in-order delivery. If a packet is dropped, a destination node cannot receive subsequent packets until the dropped packet is successfully retransmitted. To reduce performance degradation, QUIC divides a connection into multiple streams and utilizes flow control for each stream. Third, QUIC exploits the connection ID instead of a four-tuple including the source IP address, source port number, destination IP address and destination port number. As the connection ID does not depend on the IP address, QUIC provides seamless connectivity to mobile users.

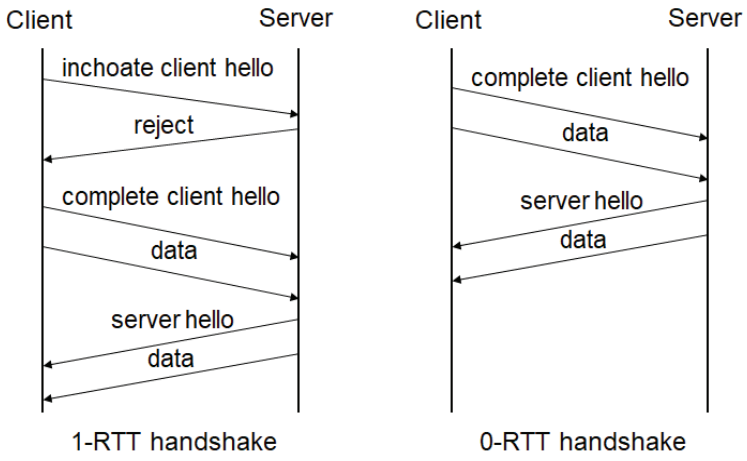


Figure 3. Connection establishment in the QUIC protocol.

Since the standardization of the QUIC protocol by the Internet Engineering Task Force (IETF), multiple QUIC implementations have been developed. To enhance the interoperability of QUIC implementations, some studies have proposed test cases, log formats and visual tools for debugging [25–27]. Several works have attempted to improve the performance of the QUIC protocol. In Reference [19], Multipath QUIC (MPQUIC) was used to exploit several paths in parallel to increase the throughput. The path manager, which selected the transmission path based on the lowest RTT and the path identifier were used. The packet number was maintained for each path for reliability. To deal with throughput degradation owing to asymmetric multipaths, Stream-Aware Earliest Completion

First (SA-ECF) scheduler was proposed in Reference [20]. The SA-ECF scheduler determined the path to minimize the transmission completion time on each stream. However, MPQUIC focuses on increasing the throughput rather than reducing the latency.

Some works have also addressed latency decrement. Adaptive Segment Retrieval (ASR) based on QUIC has been proposed to reduce initial buffering and latency for video streaming services [21]. ASR adapts the number of emulated connections to control the window size. If ASR increases the number of emulated connections, the file is downloaded quickly; otherwise, the download speed decreases. In Reference [22], QUIC was employed for the performance enhancement of multimedia services by managing the congestion window based on object priorities that are acquired from the web browser. If the priority of the multimedia content is high, QUIC increases the throughput by increasing the aggressiveness of congestion control. However, because these studies do not consider network conditions such as the throughput, loss ratio and the number of concurrent connections, aggressive downloads can lead to excessive competition and performance degradation. In Reference [23,24], QUIC was extended to support partial reliable transmission for the performance enhancement of multimedia applications. In partial reliable transmission, essential data were transmitted reliably and, less important data were transmitted unreliably to avoid retransmission. However, while these works permit packet loss for optimization of multimedia services, our approach maintains reliability.

Unlike previous research, the purpose of our study is latency reduction by minimizing computational and network overheads. The proposed scheme determines the initial transmission rate based on previous network conditions. To estimate network conditions, we calculate the average congestion window when a connection is terminated or a 0-RTT connection establishment is used. As the proposed approach employs a low complexity algorithm, it has low computational overheads.

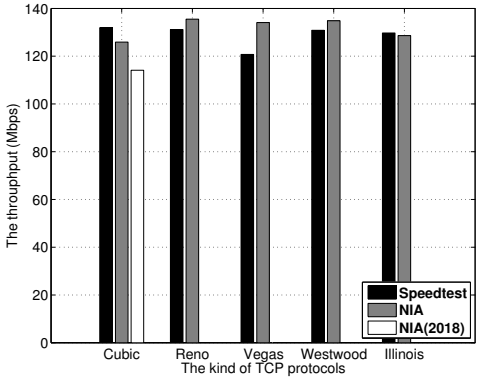
3. Measurement of Mobile Network Performance

In this section, we describe the performance measurement results for LTE and 3G networks, which are operated by the KT corporation, a well-known telecommunications corporation in South Korea. We used a Nexus 5X with Android 6.0.1. We used the Speedtest app and NIA app with different TCP protocols to precisely measure the throughput in LTE networks and 3G networks [30,31]. We varied the type of TCP in 2016 and conducted the measurement with TCP Cubic in 2018. To observe the quality of the web service experience for smart device users, we measured web page loading times for Naver and Daum, which are famous portals in South Korea. The measurements were conducted from 3 June 2016 to 20 September 2016 at Keimyung University, Daegu, South Korea and from 7 August 2018 to 28 August 2018 at Changwon National University, Changwon, South Korea.

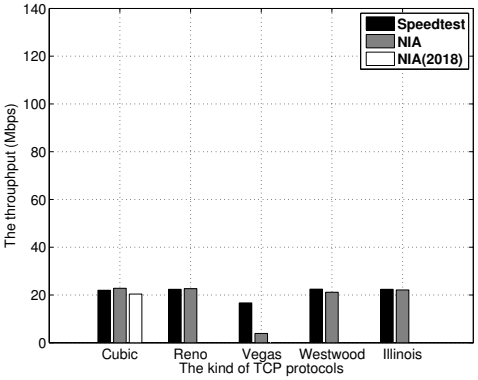
Figure 4 presents the measured throughput for LTE and 3G networks. In LTE networks, the average download and upload throughput values were approximately 129 and 20 Mbps, respectively. In 3G networks, the respective throughput values were 5.4 and 2.4 Mbps. For both networks, the download throughput was similar for all types of TCP protocol. In LTE networks, the upload throughput of TCP Vegas was lower than that of the others because TCP Vegas maintains the congestion window to avoid unnecessary transmission failure. The throughput was lower in 2018 than in 2016 because the measurement location had changed; however, the download throughput was sufficient to support video streaming and web services.

We then measured the web page loading times for both networks and the results are presented in Figure 5. In LTE networks, the web page loading time for both portals decreased from 2016 to 2018. However, the average access time for Naver and Daum portals was still approximately 1.4 s and 2.4 s, respectively. In 3G networks, the average web page loading time for Naver and Daum was approximately 3.9 s and 3.6 s, respectively. 3G networks require a longer web page loading time than LTE networks due to the low transmission rate. However, the gap between LTE and 3G networks was less than the difference in throughput between the networks; therefore, the large loading time is due to the slow start phase and connection initiation overheads, even though the network bandwidth is large

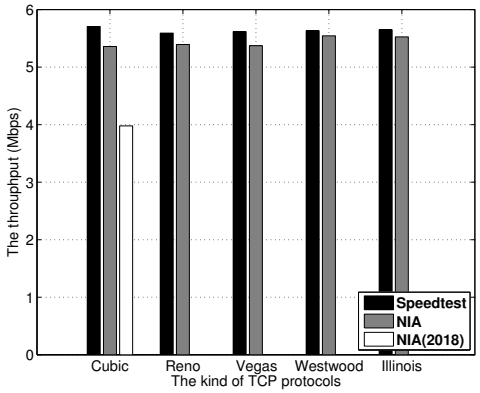
and the RTT for sites in South Korea is tens of milliseconds. This results in a low quality experience due to large latency.



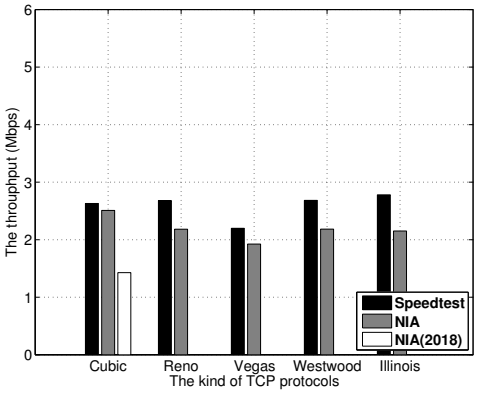
(**a**) Download throughput in LTE networks



(**b**) Upload throughput in LTE networks



(**c**) Download throughput in 3G networks



(**d**) Upload throughput in 3G networks

Figure 4. Measured throughput in Long-Term Evolution (LTE) and 3G networks for different Transmission Control Protocols (TCPs). (**a**) Download and (**b**) upload throughput in LTE networks and (**c**) download and (**d**) upload throughput in 3G networks.

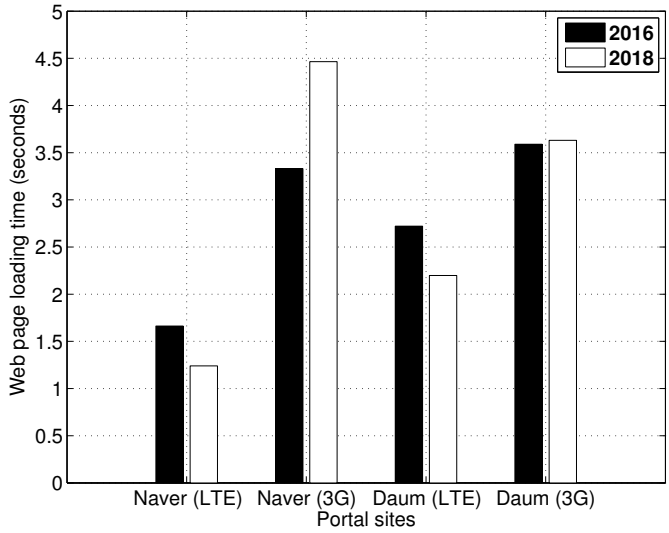
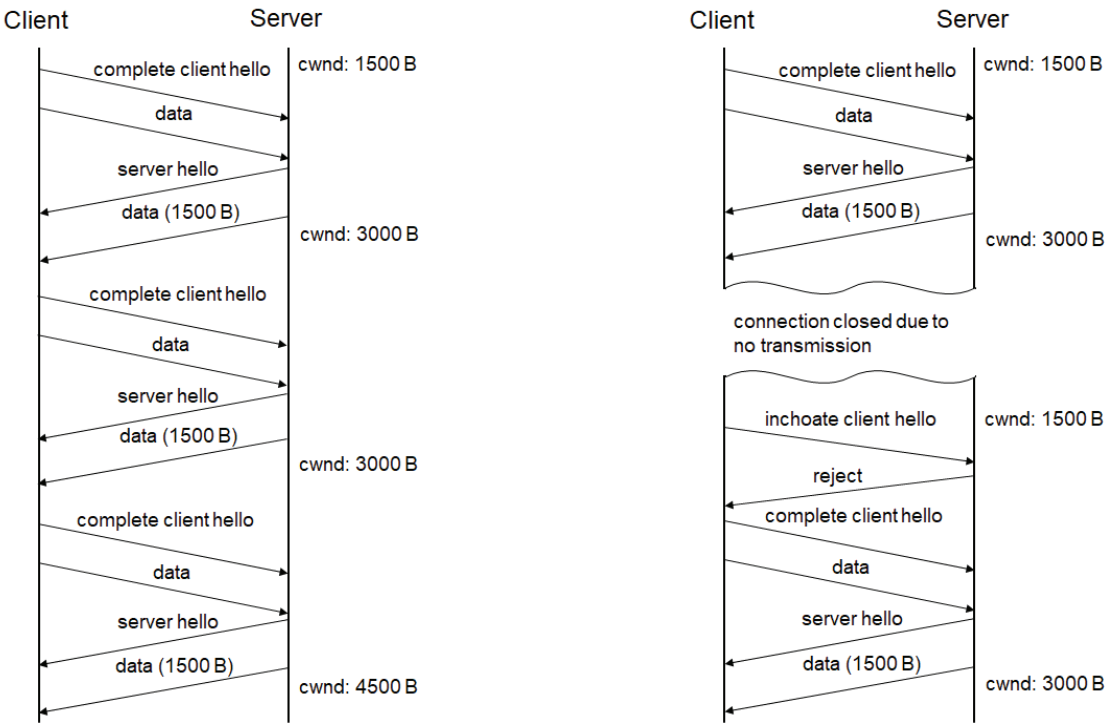


Figure 5. Web page loading times for LTE and 3G networks in 2016 and 2018.

4. Proposed Scheme

This section explains the details of the proposed scheme. Our scheme consists of two parts: observation of the network conditions and adaptation of the initial transmission rate to the network conditions. To observe network conditions, we measure the size of the congestion window and then calculate the average value. As the measurement and calculation require computational effort, more frequent calculation leads to a greater computational cost, which results in an unnecessary delay. However, if network observations are rare, the proposed scheme can incorrectly interpret the network conditions.

To decrease the computation overheads and estimate the network condition, the proposed scheme collects the value of the congestion window when two events occur. The first event is the 0-RTT handshake. QUIC maintains the congestion window while the connection is active. For example, we assumed that the initial congestion window size was 1500 B and that the connection had already been established. As shown in Figure 6a, the server sends 1500 B of data to the client after 0-RTT initiation, then the congestion window becomes 3000 B. After two transmissions from the server to the client, the congestion window size is 4500 B. When the proposed scheme receives complete client hello messages by the 0-RTT handshake, it calculates the average congestion window size to estimate the network conditions. The second event is connection termination. When a connection is closed due to no communication during a period of time, the proposed scheme calculates the average value. For example, as shown in Figure 6b, if the connection is closed, the connection information is deleted. After the 1-RTT handshake and successful transmission, the congestion window increases from 1500 B to 3000 B. Therefore, our scheme exploits 3000 B of the congestion window for the average calculation before the value of the congestion window is initialized.



(**a**) With connection open (**b**) With connection closed

Figure 6. Congestion window variation in the QUIC protocol (**a**) with connection open and (**b**) with connection closed.

Algorithm 1 presents the pseudo-code for the light-weight latency reduction scheme. The proposed scheme calculates the average size of the congestion window when the 0-RTT handshake

occurs or when the connection is closed. To calculate the average, the arithmetic mean or moving average can be used. Then, if the 1-RTT handshake occurs, our scheme determines the initial congestion window size as either the average congestion window size or the default congestion window size. If the average size is larger than the default size, the proposed scheme sets the initial congestion window size to the average congestion window size. If not, the initial congestion window size is set to the default size.

Algorithm 1 Algorithm for the Proposed Scheme.

$cwnd_{average} \leftarrow$ average size of congestion window
$cwnd_{default} \leftarrow$ default value for congestion window
$cwnd_{initial} \leftarrow$ initial value for congestion window
$cwnd_{initial} \leftarrow cwnd_{default}$
loop
  if receive complete client hello message **or** close a connection **then**
    measure the current congestion window size
    calculate average size of congestion window using the measured size
    $cwnd_{average} \leftarrow$ new average congestion window size
  end if

  if receive inchoate client hello message **then**
    $cwnd_{initial} \leftarrow Max(cwnd_{default}, cwnd_{average})$
  end if
end loop

For example, Figure 7 describes the congestion window variation by event over time, where the x-axis and y-axis present time and the congestion window size, respectively. The slope indicates the increment rate of the congestion window. At t_0, the 0-RTT handshake occurs and the node calculates the average congestion window size with simultaneous data transmission. The node increases the congestion window because the data is successfully transmitted without loss. At t_1, the node completes transmission and maintains connection information such as congestion window and connection ID until the connection is closed. At t_2, when the node receives the complete client hello message by the 0-RTT handshake, it calculates the average size of the congestion window. During transmission from hlt_2 to t_3, data is lost and the congestion window size decreases at t_3. All data including lost data are transmitted at t_4. At t_5, the node calculates the average congestion window size before the connection is terminated because no communication occurs for a period of time. At t_6, the 1-RTT handshake occurs. As the average congestion window is larger than the default size, the node sets the initial congestion window size to the average congestion window then sends the data. As the node transmits more data, the latency decreases. At t_7, the node finishes the transmission.

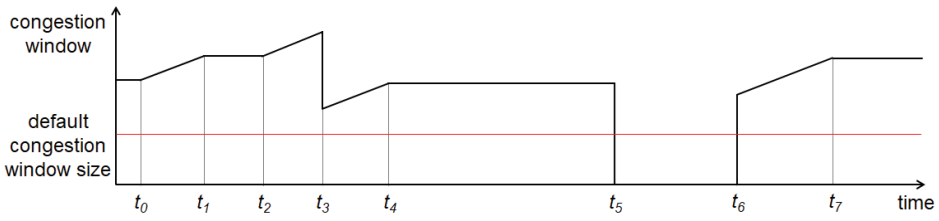


Figure 7. Change in the congestion window by event in the proposed scheme.

5. Performance Evaluation of the Proposed Scheme

In this section, we evaluate the performance of the proposed scheme using proto-quic in two different topologies and compare our results to the default QUIC configuration [32]. First, we measured

the transmission latency, adaptivity to network environment and data loss ratio in simple topology without a background flow. Then, we extended the experiment to evaluate the performance when our scheme competes with background flow. We conducted the experiment and measured the performance for a dumbbell topology with TCP traffic. According to Reference [18] and Figure 4, the network environments for both simple and dumbbell topologies are described in Table 1. Here, the "average" network environment indicates a global average RTT and global average loss rate. The "good" network environment indicates the lowest RTT and lowest retransmission. The "bad" network environment has the highest RTT and highest transmission failure. The bandwidth was set to 100 Mbps for similarity with the LTE networks. In both experiments, the client sends two different sizes of receiver data: 1 MB data for a web service and a 10 MB data chunk for a video streaming service.

Table 1. Network environments for simple and dumbbell topologies.

Network Environment	Bandwidth (Mbps)	RTT (ms)	Loss Ratio (%)
Good	100 Mbps	38 ms	1
Average	100 Mbps	50 ms	2
Bad	100 Mbps	188 ms	8

5.1. Simple Topology Experiment

We first evaluated the performance of the proposed scheme with simple topology in Figure 8. The simple topology consists of three desktops with i5-7500 (3.4 GHz quad core CPU), 8 GB RAM and Ubuntu 16.04 LTS. The router utilized NetEm to configure a bottleneck link [33] based on Table 1.

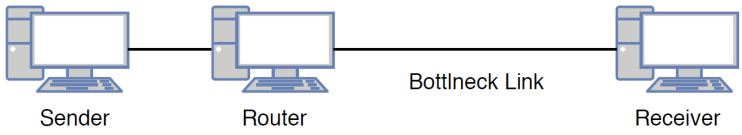
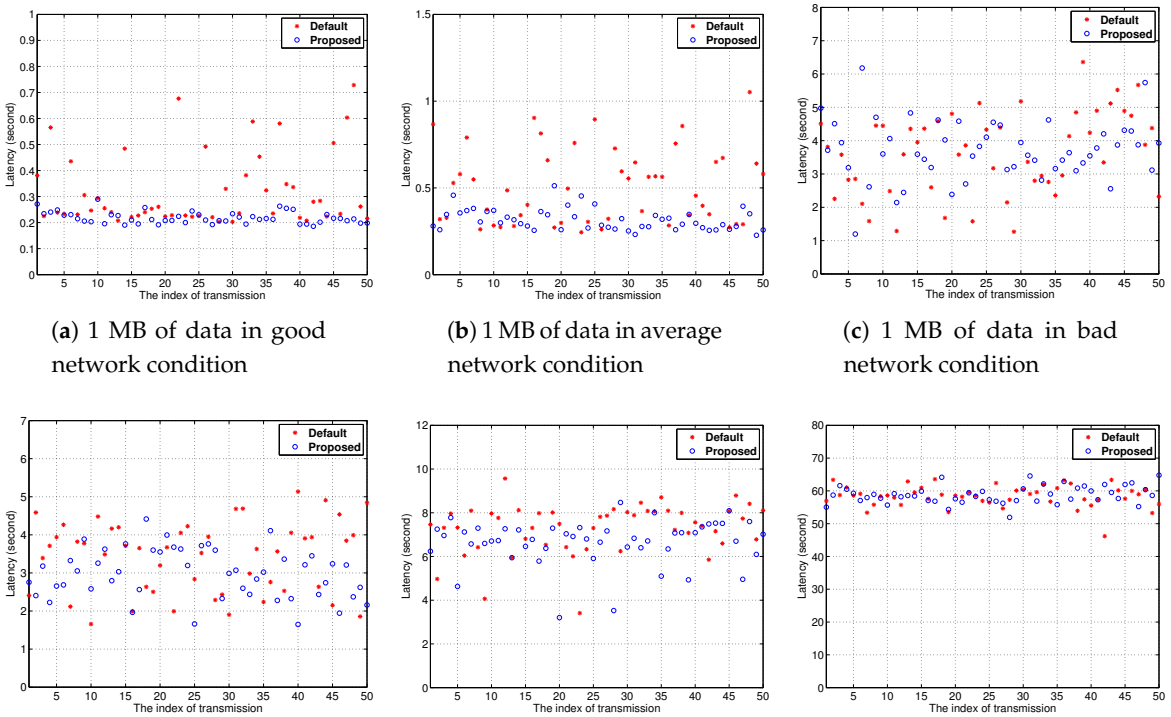


Figure 8. Schematic of the simple topology used in the experiment.

5.1.1. Latency

We conducted 50 transmission latency measurements when transmitting 1 MB and 10 MB of data from the sender to the receiver. As shown in Figure 9, the transmission latency increases as the network condition deteriorates. For "good" and "average" network environments, the proposed scheme shows better performance than the default configuration because the sender rapidly determines an appropriate transmission rate by setting the initial congestion window to the average congestion window of a connection.

Compared to the default configuration with 1 MB data transmission, the proposed scheme achieved a latency reduction of approximately 31% in the "good" network environment. In the "average" network environment, the latency decreased from approximately 0.52 to 0.31 s, indicating a decrease of approximately 38%. For 10 MB data transmission, the latency decrease was 13% and 9% for the "good" and "average" network environments, respectively. In the good network condition, the average latency of the proposed scheme and the default configuration was 2.98 s and 3.43 s, respectively. Because web service users leave if a web page takes longer than 3 s to load, the proposed scheme can be accommodated by users; however, the default configuration may cause user dissatisfaction [34,35]. As the data size increased, the performance enhancement decreased. Because the default QUIC configuration adapted the transmission rate to the network environment, the performance gap decreased over time. In the "bad" network environment, as the measured average congestion window was lower than the default congestion window, the proposed scheme used the default value. Therefore, both schemes showed a similar transmission delay.



(**a**) 1 MB of data in good network condition

(**b**) 1 MB of data in average network condition

(**c**) 1 MB of data in bad network condition

(**d**) 10 MB of data in good network condition

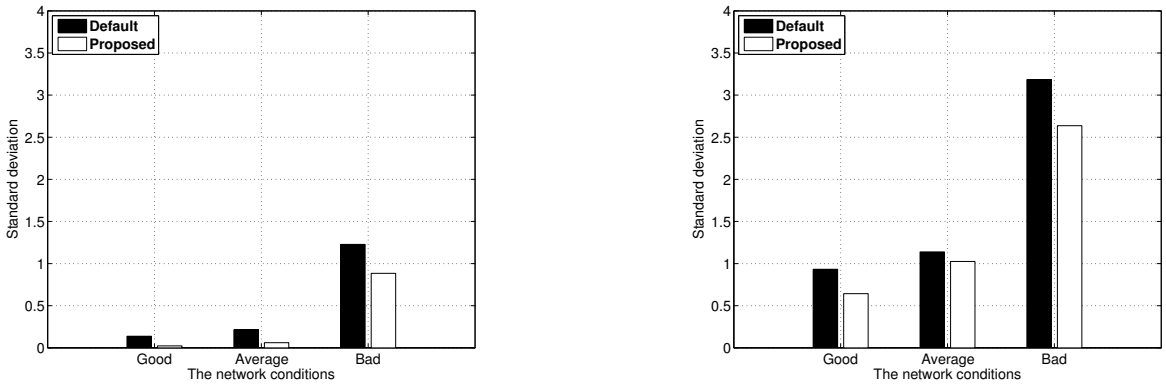
(**e**) 10 MB of data in average network condition

(**f**) 10 MB of data in bad network condition

Figure 9. Transmission latency with simple topology (**a**–**c**) 1 MB of data and (**d**–**f**) 10 MB of data in (**a**,**d**) good, (**b**,**e**) average and (**c**,**f**) bad network conditions.

5.1.2. Adaptivity to Network Environments

To reveal the adaptivity to network environments, we determined the standard deviation of the latency in Figure 10. In “good" and “average" network environments, our scheme exhibited a lower standard deviation than the default configuration. The large standard deviation implies low adaptivity to the network environment; therefore, the average calculation in our scheme estimates network conditions as the appropriate average congestion window. However, since the default configuration took a longer time to determine the proper value, it experienced larger congestion window variation over a short time. As the data size decreased, the gap between the proposed scheme and the default configuration increased; thus, the proposed scheme adjusted the transmission rate to the network environment and avoided unnecessary retransmission during 1 MB data transmission.



(**a**) 1 MB data transmission (**b**) 10 MB data transmission

Figure 10. Latency standard deviation values for different network conditions with simple topology. (**a**) 1 MB and (**b**) 10 MB data transmission.

5.2. Dumbbell Topology Experiment

Next, we evaluated the performance of the proposed scheme with dumbbell topology, which included five desktops with i5-7500 (3.4 GHz quad core CPU), 8 GB RAM and Ubuntu 16.04 LTS as presented in Figure 11. The dumbbell topology consists of two senders, two receivers and one router. To incorporate background flow, the sender transmits TCP packets to the receiver using iperf3 [36]. The other pair utilizes the proposed scheme for data transmission. The router acts as a bottleneck link using NetEm and the network configuration is based on the network environment in Table 1.

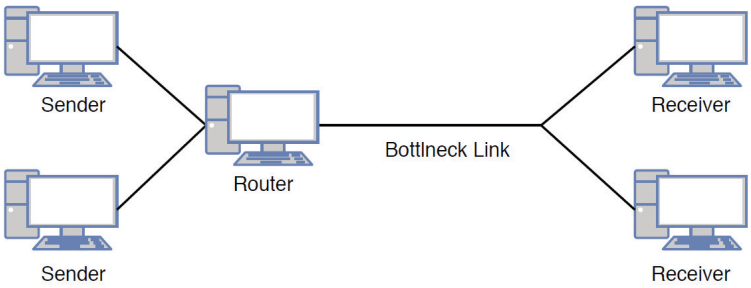
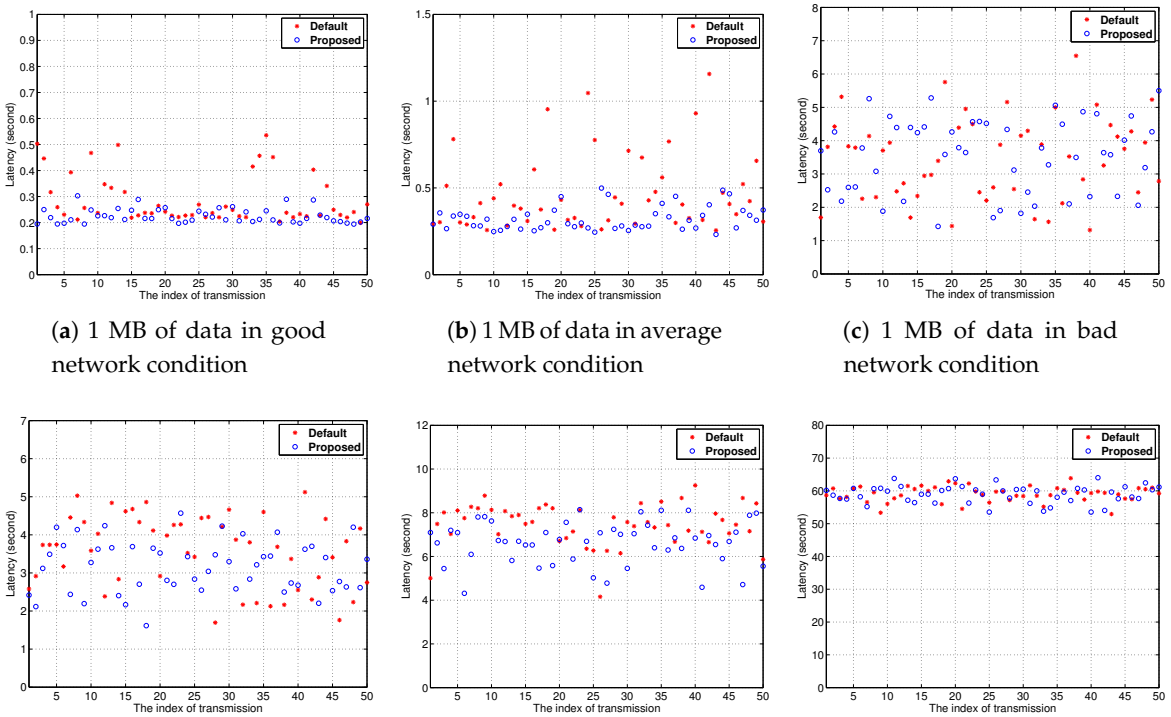


Figure 11. Schematic of the dumbbell topology used in the experiment.

5.2.1. Latency

We measured the transmission latency for the proposed scheme and the default configuration 50 times. The results are presented in Figure 12. With 1 MB data transmission, the average latency of the proposed scheme was approximately 225 ms and 323 ms in “good” and “average” network environments, respectively. The average latency of the default configuration was 288 ms and 467 ms for the two network conditions. Therefore, our approach achieves latency reduction of 22% and 31% compared to the default configuration. This latency decrease was no larger than that in simple topology because the appropriate transmission rate was reduced due to background TCP flow. With 10 MB data transmission, the proposed scheme resulted in average transmission latency of approximately 3.16 s and 6.63 s for “good” and “average” network conditions, respectively, compared to approximately 3.59 s and 7.47 s in the default configuration. As the data size increased, the benefit of the latency reduction decreased; however, the proposed scheme still reduced the average latency by approximately 12% and 11%. In the “bad" network environment, as the proposed scheme utilized the default congestion window, the latency was approximately the same as that of the default configuration, regardless of data size.



(**a**) 1 MB of data in good network condition

(**b**) 1 MB of data in average network condition

(**c**) 1 MB of data in bad network condition

(**d**) 10 MB of data in good network condition

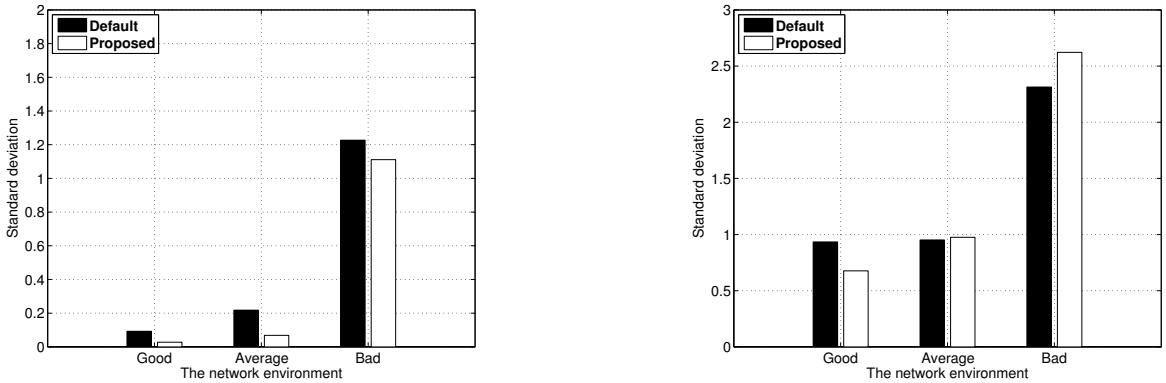
(**e**) 10 MB of data in average network condition

(**f**) 10 MB of data in bad network condition

Figure 12. Transmission latency with dumbbell topology (**a**–**c**) 1 MB of data and (**d**–**f**) 10 MB of data in (**a**,**d**) good, (**b**,**e**) average and (**c**,**f**) bad network conditions.

5.2.2. Adaptivity to Network Environments

The background flow affects the throughput and latency performance due to bandwidth sharing. To monitor the impact of background flow on the adaptivity to network environments, we measured the standard deviation of the latency. The results are presented in Figure 13. The standard deviations of the proposed scheme were less than or equal to those of the default configuration in "good" and "average" network environments. This implies that our scheme can estimate the network conditions despite the effect of background flow. In the "bad" network environment, both schemes used the default configuration; therefore, the results were similar.



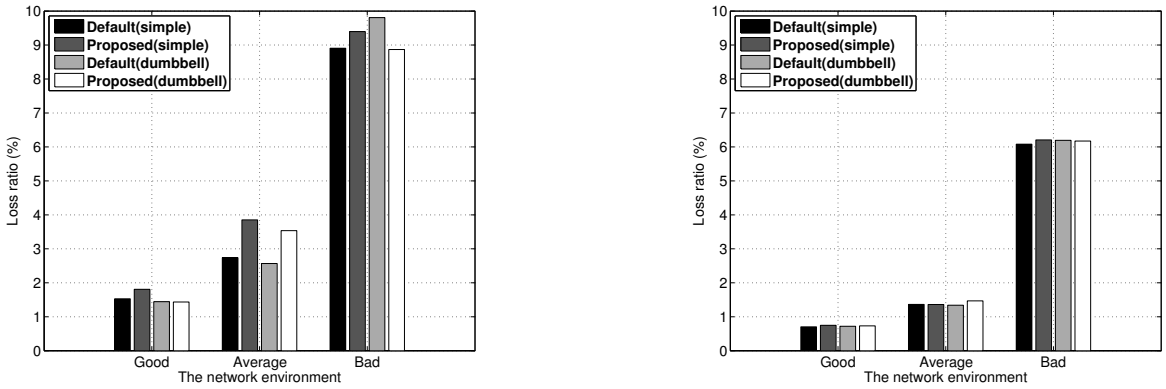
(**a**) 1 MB data transmission

(**b**) 10 MB data transmission

Figure 13. Latency standard deviation values for different network conditions with dumbbell topology. (**a**) 1 MB and (**b**) 10 MB data transmission.

5.3. Packet Loss Ratio

We measured the loss ratio for both simple topology and dumbbell topology and results are presented in Figure 14. Packet loss during 1 MB data transmission was more frequent than that during 10 MB data transmission in “good" and “average" network environments. This can be explained as follows. During 10 MB data transmission, the transmission rate was adjusted to the network condition. Therefore, the proposed scheme and the default configuration experienced a similar packet loss ratio over time. During 1 MB data transmission, however, many packets were aggressively transmitted before determining the appropriate transmission rate. This is because of the slow start phase in the default configuration and the large initial congestion window in the proposed scheme. Therefore, for a short transmission time, both schemes experienced greater packet losses than the loss ratio specified in Table 1. During 1 MB data transmission in both experiment topologies, the proposed scheme showed a higher loss ratio than the default configuration in “good” and “average” network environments. As the initial congestion window was set to the average value, the proposed scheme sent data more aggressively than the default configuration, resulting in a higher loss ratio.

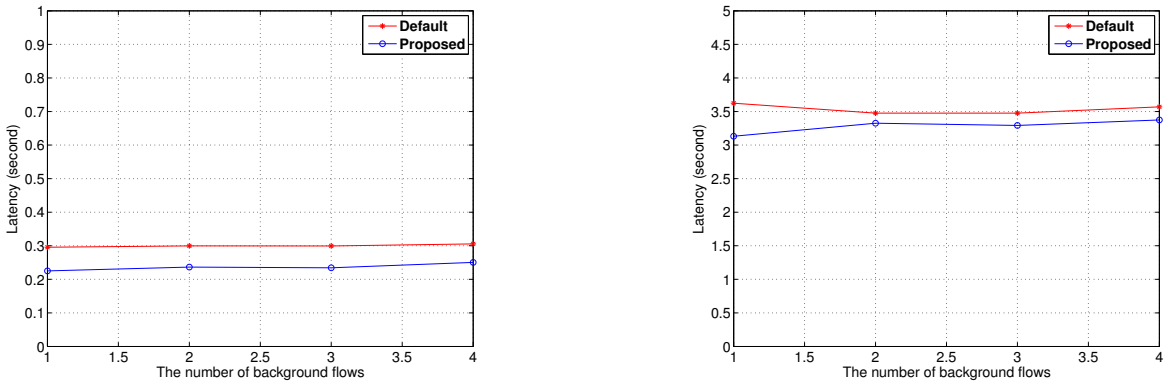


(**a**) 1 MB data transmission (**b**) 10 MB data transmission

Figure 14. Packet loss ratios for different network conditions. (**a**) 1 MB and (**b**) 10 MB data transmission.

5.4. Latency with Multiple Background Flows

In an actual environment, the QUIC protocol competes with multiple TCP flows for transmission. Therefore, we conducted 50 transmission latency measurements under good network conditions with varying numbers of background flows. Figure 15 shows the average latency with different flows. With 1 MB data transmission, the average latency of the proposed scheme and the default configuration remains constant regardless of the number of background flows. The proposed scheme exhibits a latency reduction of 18% to 24% because it sends data at appropriate initial transmission rate. As the data size increases, the latency difference between the proposed scheme and the default configuration decreases because the transmission rate increases with time in the default configuration. However, the latency of the proposed scheme is still less than that of the default configuration.



(**a**) 1 MB data transmission (**b**) 10 MB data transmission

Figure 15. Transmission latency for different background flows with dumbbell topology. (**a**) 1 MB and (**b**) 10 MB data transmission.

6. Conclusions

In this study, we measured the performance of LTE and 3G networks to determine the quality of experience for smart device users. Based on these results, we then proposed a latency reduction scheme in QUIC to improve the quality of the user experience. To reduce the delay due to computational overheads, a light-weight scheme with low complexity was proposed. The performance of this scheme was then verified through experiments on a testbed. Compared to the default QUIC protocol configuration, the proposed scheme achieved significant latency reduction.

Author Contributions: Conceptualization, J.J. and D.A.; Methodology, D.A.; Software, J.J.; Validation, D.A.; Investigation, J.J. and D.A.; Data curation, J.J.; Writing–original draft, D.A.; Writing—review and editing, D.A.; Visualization, J.J. and D.A.; Supervision, D.A.; Funding acquisition, D.A.

Funding: This work was supported by the National Research Foundation of Korea(NRF) grant funded by the Korea government(MSIT) (No. 2018R1C1B6008187).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Cisco. Cisco Visual Networking Index: Forecast and Trends, 2017–2022. Available online: https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/white-paper-c11-741490.pdf (accessed on 10 September 2019).
2. Jacobson, V. *Modified TCP Congestion Avoidance Algorithm*; Technical Report; LBNL Network Research Group: Berkeley, CA, USA, April 1990.
3. Henderson, T.; Floyd, S.; Gurtov, A.; Nishida, Y. The NewReno Modification to TCP's Fast Recovery Algorithm. RFC-6582. 2012. Available online: https://tools.ietf.org/html/rfc6582 (accessed on 10 September 2019).
4. Brakmo, L.S.; O'Malley, S.W.; Peterson, L.L. TCP Vegas: New Techniques for Congestion Detection and Avoidance. *ACM SIGCOMM Comput. Commun. Rev.* **1994**, *24*, 24–35. [CrossRef]
5. Xu, L.; Harfoush, K.; Rhee, I. Binary increase congestion control (BIC) for fast long-distance networks. In Proceedings of the IEEE INFOCOM, Hong Kong, China, 9 March 2004 .
6. Ha, S.; Rhee, I.; Xu, L. CUBIC: A New TCP-Friendly High-Speed TCP Variant. *ACM SIGOPS Oper. Syst. Rev.* **2008**, *42*, 64–74. [CrossRef]
7. Casetti, C.; Gerla, M.; Mascolo, S.; Sanadidi, M.; Wang, R. TCP Westwood: end-to-end congestion control for wired/wireless networks. *Wirel. Netw.* **2002**, *8*, 467–479. [CrossRef]
8. Liu, S.; Basar, T.; Srikant, R. TCP-Illinois: A loss- and delay-based congestion control algorithm for high-speed networks. *Perform. Eval.* **2008**, *65*, 417–440. [CrossRef]
9. Zaki, Y.; Pötsch, T.; Chen, J.; Subramanian, L.; Görg, C. Adaptive Congestion Control for Unpredictable Cellular Networks. *ACM SIGCOMM Comput. Commun. Rev.* **2015**, *45*, 509–522. [CrossRef]

10. Radhakrishnan, S.; Cheng, Y.; Chu, J.; Jain, A.; Raghavan, B. TCP Fast Open. In Proceedings of the ACM CoNEXT, Tokyo, Japan, 6–9 December 2011.
11. Zhou, W.; Li, Q.; Caesar, M.; Godfrey, P. ASAP: A Low-Latency Transport Layer. In Proceedings of the ACM CoNEXT, Tokyo, Japan, 6–9 December 2011.
12. Pradhan, P.; Chiueh, T.; Neogi, A. Aggregate TCP Congestion Control Using Multiple Network Probing. In Proceedings of the IEEE International Conference on Distributed Computing Systems, Genoa, Italy, 21–25 June 2010.
13. Dukkipati, N.; Refice, T.; Cheng, Y.; Chu, J.; Herbert, T.; Agarwal, A.; Jain, A.; Sutin, N. An Argument for Increasing TCP's Initial Congestion Window. *ACM SIGCOMM Comput. Commun. Rev.* **2010**, *40*, 26–33. [CrossRef]
14. Zheng, H.; Boyce, J. An Improved UDP Protocol for Video Transmission Over Internet-to-Wireless Networks. *IEEE Trans. Multimedia* **2001**, *3*, 356–365. [CrossRef]
15. Venckauskas, A.; Morkevicius, N.; Bagdonas, K.; Damasevicius, R.; Maskeliunas, R. A Lightweight Protocol for Secure Video Streaming. *Sensors* **2018**, *18*, 1554. [CrossRef] [PubMed]
16. QUIC. QUIC, a Multiplexed Stream Transport over UDP. Available online: https://www.chromium.org/quic (accessed on 10 September 2019).
17. Roskind, J. QUIC: Design Document and Specification Rationale. Available online: https://github.com/addogra/QUIC-Documents/blob/master/QUIC_DesignDocumentandSpecificationRationale.pdf (accessed on 10 September 2019).
18. Langley, A.; Riddoch, A.; Wilk, A.; Vicente, A.; Krasic, C.; Zhang, D.; Yang, F.; Kouranov, F.; Swett, I.; Iyengar, J.; et al. The QUIC Transport Protocol: Design and Internet-Scale Deployment. In Proceedings of the ACM SIGCOMM, Los Angeles, CA, USA, 21–25 August 2017.
19. De Coninck, Q.; Bonaventure, O. Multipath QUIC: Design and Evaluation. In Proceedings of the ACM CoNEXT, Seoul/Incheon, Korea, 12–15 December 2017.
20. Rabitsch, A.; Hurtig, P.; Brunstro, A. A Stream-Aware Multipath QUIC Scheduler for Heterogeneous Paths. In Proceedings of the ACM CoNEXT Workshop on EPIQ, Crete, Greece, 14 December 2018.
21. Choi, J.; Jung, C.; Yeom, I.; Kim, Y. Streaming Service Enhancement on QUIC Protocol. In Proceedings of the ICOMP, Budapest, Hungary, 24–27 September 2017.
22. Szabo, G.; Racz, S.; Bezzera, D.; Nogueira, I.; Sadok, D. Media QoE Enhancement with OUIC. In Proceedings of the IEEE INFOCOM Workshop, Atlanta, GA, USA, 10–15 April 2016.
23. Palmer, M.; Kruger, T.; Chandrasekaran, B.; Feldmann, A. The QUIC Fix for Optimal Video Streaming. In Proceedings of the 14th International Conference on Emerging Networking EXperiments and Technologies, Heraklion/Crete, Greece, 4–7 December 2018.
24. Perkins, C.; Ott, J. Real-time Audio-Visual Media Transport over QUIC. In Proceedings of the 14th International Conference on Emerging Networking EXperiments and Technologies, Heraklion/Crete, Greece, 4–7 December 2018.
25. Piraux, M.; Coninck, Q.; Bonaventure, O. Observing the Evolution of QUIC Implementations. In Proceedings of the 14th International Conference on Emerging Networking EXperiments and Technologies, Heraklion/Crete, Greece, 4–7 December 2018.
26. Rath, F.; Schemmel, D.; Wehrle, K. Interoperability-Guided Testing of QUIC Implementations using Symbolic Execution. In Proceedings of the 14th International Conference on Emerging Networking EXperiments and Technologies, Heraklion/Crete, Greece, 4–7 December 2018.
27. Marx, R.; Reynders, J.; Quax, P. Towards QUIC debuggability. In Proceedings of the 14th International Conference on Emerging Networking EXperiments and Technologies, Heraklion/Crete, Greece, 4–7 December 2018.
28. Hall-Andersen, M.; Wong, D.; Sullivan, N.; Chator, A. nQUIC: Noise-Based QUIC Packet Protection. In Proceedings of the 14th International Conference on Emerging Networking EXperiments and Technologies, Heraklion/Crete, Greece, 4–7 December 2018.
29. Xin, F.; Jamalipour, A. TCP performance in wireless networks with delay spike and different initial congestion window sizes. *Comput. Commun.* **2006**, *29*, 926–933. [CrossRef]
30. Speedtest app. Available online: https://www.speedtest.net (accessed on 10 September 2019).
31. NIA app. Available online: http://speed.nia.or.kr/index.asp (accessed on 10 September 2019).
32. proto-quic. Available online: https://github.com/google/proto-quic (accessed on 10 September 2019).

33. netem. Available online: https://wiki.linuxfoundation.org/networking/netem (accessed on 10 September 2019).
34. Akamai. STATE OF ONLINE RETAIL PERFORMANC: 2017 HOLIDAY RETROSPECTIVE. Available online: https://www.akamai.com/us/en/multimedia/documents/report/akamai-state-of-online-retail-performance-2017-holiday.pdf (accessed on 11 October 2019).
35. Butkiewicz, M.; Wang, D.; Wu, Z.; Madhyastha, H.V.; Sekar, V. Klotski: Reprioritizing Web Content to Improve User Experience on Mobile Devices. In Proceedings of the USENIX Symposium on NSDI, Oakland, CA, USA, 4–6 May 2015.
36. iperf3. Available online: https://iperf.fr/iperf-download.php (accessed on 10 September 2019).

© 2019 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (http://creativecommons.org/licenses/by/4.0/).