Designing and Testing of a Semi-Physical QUIC Video Streaming Simulation Platform

Jun Zhao[1], Siqie Zhang[2], Ben Liu[3], Hongjie Yu[4], Mingwu Yao[5], and Fuzheng Yang[6]
{zj410728@163.com[1], sqzhang1994@163.com[2], liuben@stu.xidian.edu.cn[3]}

State Key Laboratory on Integrated Services Networks, Xidian University, Xi'an, China

Abstract. QUIC (Quick UDP Internet Connections) is a streaming control protocol that proposed by Google in recent years. In order to study the ability of QUIC to support streaming video in a wireless mobile environment, we combine a DASH (Dynamic Adaptive Streaming HTTP) server and QUIC source code to design and implement a virtual machine-based testing and simulation system. The system can not only use real video traffic for streaming video playback, but also use the simulation system such as NS3 to introduce wireless network features. Based on the system, we perform a simulation analysis on the proposed QUIC stream video improvement strategy for large-delay ACK discarding (LDAD). The results demonstrate that the simulation platform can help to understand and investigate the interaction between streaming video and QUIC in-depth.

Keywords: QUIC; DASH; LDAD; NS3; Simulation Platform

1 Introduction

Video exchanging and browsing via social networking website and mobile apps is becoming a real killer application in recent years [1]. Aiming at improving the QoE (Quality of Experience) of users, the next-generation protocols such as HTTP/2 [2], QUIC [3] and HTTP/3 [4] have been proposed. A conventional way to support such applications is a combination of HTTP/1.1 and TCP (HTTP over Transportation Control Protocol) [5]. However, they often incur huge connection establishment delay with complex multi-connection management overhead, and suffer severe Head-of-Line Blocking (HOLB) [6] issues. HTTP/2 adopts a series of new features to solve the application-layer HOLB [2][5] but triggers blockage in the transport layer[7]. The latency of TCP Connection Establishment is another problem. Some enhancements such as TCP fast open transport-layer [8] have been proposed long before but are rarely used in practice. Moreover, TCP's close coupling with the kernel of popular Operating Systems (OSs) hampers its updating and optimization speed in a view of ordinary users.

In 2012, Google presented QUIC [3], a new type of multiplex and secure transmission protocol based on UDP. QUIC have serval new significant features including multiplexing and flow control that is equivalent to HTTP/2, encryption equivalent to TLS and connection semantics, reliability and congestion control equivalent to TCP. Fig. 1 presents the functionality of the QUIC in the OSI reference model. At the same time, the HTTP-over-QUIC experimental protocol has been renamed to HTTP/3 and is expected to be the third official HTTP release.

QUIC is purposely designed to overcome the shortcomings of TCP. QUIC's stream multiplexing greatly alleviates the transport layer HOLB. There is usually 0-RTT (Round Trip Time) in the connection establishment of QUIC. Furthermore, a QUIC connection is identified

EAI MOBIMEDIA 2020, August 27-28, Harbin, People's Republic of China

DOI 10.4108/eai.27-8-2020.2294432

by a 64-bit Connection ID instead of the quad of IP address and port number, which is significantly suitable for NAT rebinding and network switching. QUIC adopts a strict authentication and encryption mechanism. QUIC does flow control more efficient than TCP. QUIC's congestion control algorithm is pluggable and implemented in the user space of the OS. QUIC replaces TCP's Sequence Number with a strictly monotonically increasing Packet Number, avoiding the retransmission ambiguity of the TCP. Every QUIC packet contains an unencrypted public header and an encrypted payload. And the latter part contains one or more data or control frames, carrying application data and control messages respectively [9]. For the latest features, please refer to [10]. Finally, QUIC ACKs explicitly carry the delay inccured at the client receiver, explicitly inform the sender the latency.

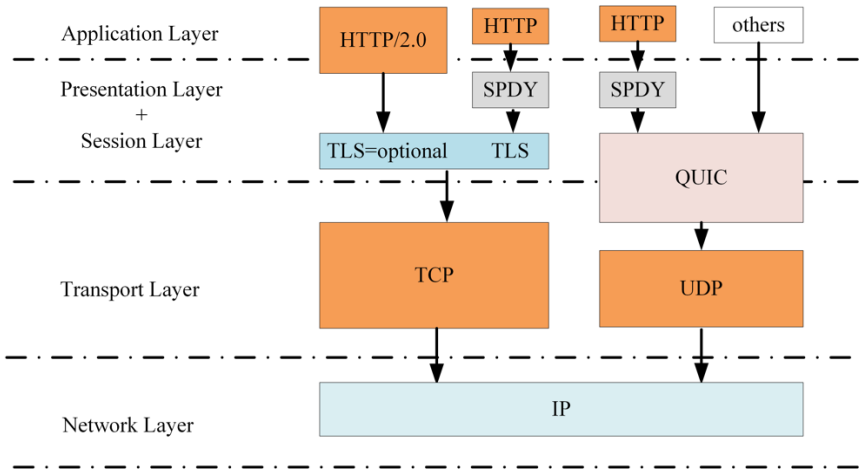

Fig. 1. The architecture of QUIC.

QUIC has arosen widespread research interests. Megyesi et al. compare the QUIC with HTTP and SPDY[11], and proposed a method to know when it is appropriate to use QUIC. Somak gives a detailed analysis of QUIC access to Web pages [12]. Kakhki Arash Molavi et al. test QUIC in a large number of environments, showing that, though QUIC is generally better than TCP, its performance is significantly reduced under mobile and cellular networks [13]. Jan Rüth et al. show that QUIC usage is increasing as a protocol deployed in user space [14]. Szabo Geza et al. report the QoE of QUIC for media application experience [15]. Clark proposes a non-multiplexed relay transmission protocol QUUX based on QUIC [16].

To study the performance of QUIC more effectively, we build an emulation/simulation platform combing DASH, QUIC and NS3 (Network Simulator). DASH technology is a hybrid media distribution method, which uses HTTP protocol to download and distribute content, likes the HTTP progressive download method. NS3 provides a wireless network simulation scenario. QUIC is used to transmit video service streams, and DASH is used to request video service streams and play received video segments. The DASH, QUIC and NS3 are independently developed and sophistically connected. To the best of our knowledge, there is no published literature depicts such a design so far. In the system, DASH plays the role of the application layer entity and QUIC takes the responsibility of the function of the transport layer. NS3 works as a network. We use the subclass rewriting and dynamic library technology to connect DASH

and QUIC, and use the datagram Socket technology and multithreading multitasking parallel mechanisms to attach QUIC and NS3.

Using the platform, we verify our improvement towards QUIC. Different to known methods, we implemented the improvement at AP (Access Point)/Base Station (BS) of the QUIC based on the feature of the wireless network. The performance is verified on the designed system, especially detailing the performance of the algorithm in the scenario of LTE and WLAN.

The rest of this paper is organized as follows. We detail the simulation platform of QUIC and do some test work on it in Section 2. In section 3 we introduce the mechanism of LDAD. Performance evaluation is presented in Section 4. Section 5 concludes this paper.

2 Simulation platform

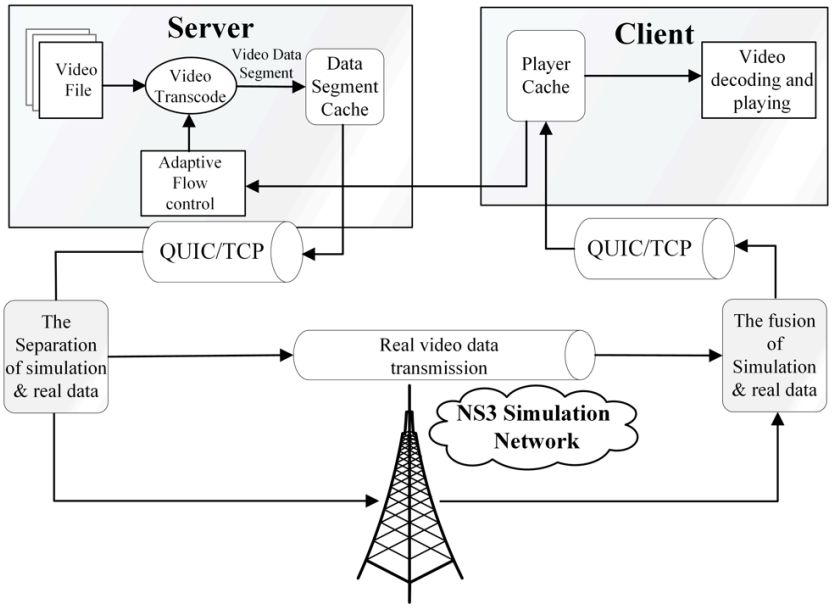

Fig. 2. The simulation system.

As shown in Fig. 2, the semi-physical QUIC platform consists of three major modules: DASH, QUIC and NS3. The DASH system is a suite of video streaming tools. The server is used to generate the video stream and carry out the video code. The DASH client receives, decodes, and plays the streaming video. The client code include also the functions of QoE evaluation to track and evaluate the playback effect. The video stream data output by the DASH server are delivered by QUIC. After being multiplexed and sorted by the QUIC server, the video stream data are assembled into UDP frames, and then enter into the NS3 simulation network. The simulated network will incur delay, jitter, and loss to the UDP frames, to simulating the affections of a real network on the QUIC transportation. Afterwards the data enter into the QUIC client, where they are error corrected, resorted, confirmated, and/or required to be retransmitted. The resulting stream is passed to the DASH client where the video is decoded and played.

2.1 System building

The building blocks of the platform are DASH, QUIC, and NS3, arranged in the order from top to bottom. The DASH and QUIC are implemented in the same OS of one virtual machine, and the NS3 is running at another virtual machine. One of the key tasks of platform construction is to connect the above three blocks so that the three platforms operate in synchronization and cooperation.

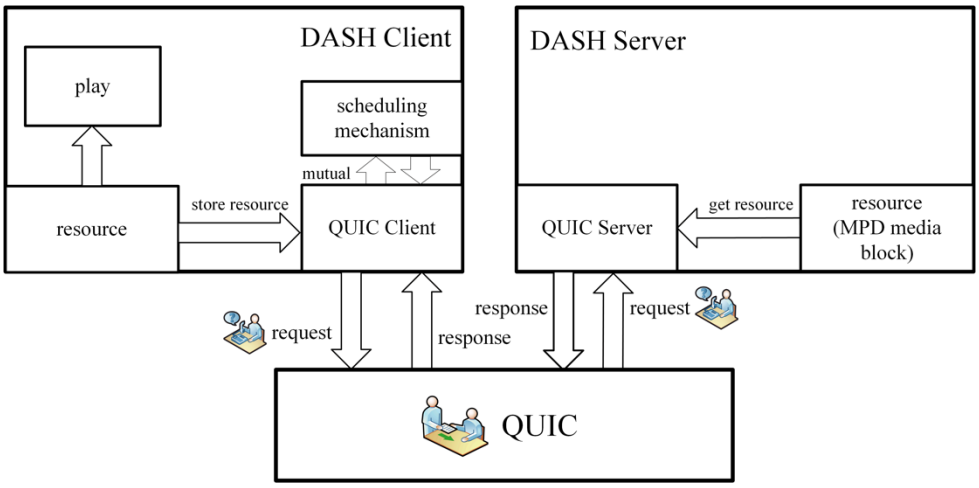

Fig. 3. DASH-QUIC connection scheme.

The Connection between DASH platform and QUIC platform: In this scheme, subclass rewriting technology and dynamic library technology are used. The DASH-QUIC connection scheme is based on adaptive streaming media and changes the selected media block from TCP transmission to QUIC transmission. The DASH-QUIC as shown in Fig. 3. A QUIC client is created in the DASH client, and the DASH client interacts with the DASH server through the QUIC server-client interactions that providing a transportation link.

The Connection between QUIC platform and NS3 platform: This scheme is implemented based on the datagram socket communication technology. The specific connection scheme implementation diagram is shown in Fig. 4.

As NS3 has difficulties to deliver the real stream data packets due to its capacity, the data packets are treated in a modificated way. When the QUIC client sends a packet, it first transmits the packet size information to the NS3 client through the socket process communication mechanism (In theory, QUIC needs to transmit packet size and packet number informations to NS3, but since the packet number in QUIC increases successively, and the packet information transmitted by QUIC to NS3 will not be lost, so we can set a self-increment variable as the mapping of the QUIC package number. Therefore, QUIC only needs to transmit packet size information to NS3.); then the actual data packet is transmitted to the receiving buffer of the QUIC server via local transmission. The NS3 client first puts the packet size information into the cache, and the simulate gateway node will generate a dummy data packet with the same size and header and sends it through the simulated network.

When the dummy packet is received by the NS3 receiving node, the NS3 server's HandleRead function sends the correctly received packet number and corresponding

transmission delay to the QUIC server via the socket. The lost packet in NS3 will incur a real packet loss in the QUIC. The QUIC server puts the received NS3 message into the NS3 message receiving buffer. Each time the QUIC server receives a local transmission packet from the QUIC client, if the packet is not lost, after a certain delay according to the delay information, the packet is forwarded upward. When the packet processing is completed, the NS3 informatio n buffer will be traversed again until NS3 message cache is empty. The QUIC server enters a state waiting to receive the actual data packet transmitted locally by the QUIC client.

The packet sending process of the QUIC server is basically the same as that of the QUIC client, except that the corresponding processing interfaces are different.

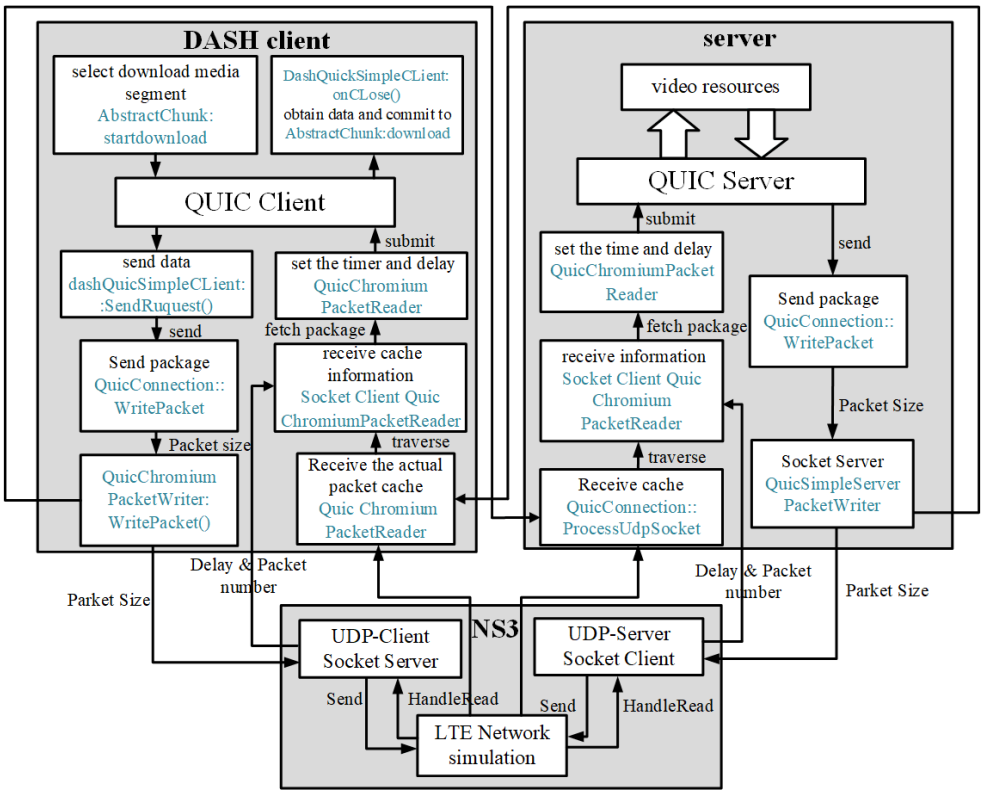

Fig. 4. Detail design of the DASH-QUIC-NS3 connection scheme.

If NS3 retransmission fails, the corresponding NS3 client or NS3 server cannot successfully receive packets from the simulated network. At this time, if NS3 only sends information about successfully received packets to QUIC, it is obvious that the actual packet receiving buffer of QUIC may be filled with the failed transmission in the simulated network which will make it impossible to receive the actual packets transmitted from the QUIC opposite end. Therefore the information of the underlying retransmission failure packet needs to be submitted upwards so that QUIC can delete the actual data packet corresponding to the network transmission failure. The current solution is: during the underlying retransmission process of the NS3 server and client, if the number of retransmissions exceeds the maximum retransmission threshold set by the source code, socket communication is triggered to send corresponding packet information

to the corresponding QUIC, and QUIC clears the corresponding receiving buffer according to the information.

2.2 System Verification

We do some verification tests after the system is built. In the first verification, the connectivity of the simulation platform is tested without packet loss. In this case, we have tested that QUIC, NS3, and DASH are connected as we expected (after a lot of debugging and rewriting). The smoothly displayed video showing our treatment of the simulated packet streams is correct.

The second verification is showing the effects of losses introduced by the NS3 network on the video playback qualities. As shown in Table 1, the playback of DASH was tested with the packet loss rate of 0.1, 0.2, 0.3, 0.4, and 0.5 respectively, with different QoE performances.

Table 1. Packet loss rate and video playback test.

Packet loss rate	DASH playback
0.1	Can be played, no stutter, no errors
0.2	Can be played, some freezes, no errors
0.3	Can be played, obvious freeze, no errors
0.4	Can be played, obvious freeze, no errors
0.5	Can be played, obvious freeze, poor playback effect

3 Using the platform to test LDAD

3.1 Why LDAD

Chrome-based experiments show that packet loss can be reduced by the Pacing mechanism. Pacing mechanism reduces packet flow fluctuations, thereby reducing congestion-based losses (i.e. packet drops in routers due to overflow). The current QUIC is based on bandwidth estimation to dynamically adjust the pacing rate. The pacing rate in QUIC is calculated as:

$$r = \begin{cases} 2 \times \dfrac{C_w}{SRTT} & \text{slow start phase} \\ 1.25 \times \dfrac{C_w}{SRTT} & \text{not slow start phase} \end{cases} . \quad (1)$$

Here C_w refers to the current congestion window, and SRTT is the Smooth Round Trip Time.

We want to improve the performance of QUIC in a wireless network where the user terminals are poorly covered by the BS (Base Station) or AP (Access Point). The essence of increasing the QUIC transmission rate is to increase the Pacing rate of QUIC. According to the source code of the QUIC protocol, the QUIC packet rate calculated is based on the current updated congestion window and the current updated SRTT. Each time the QUIC receives an

ACK, the SRTT will be updated by the current RTT. When the congestion or sudden interference occurs in the network, it will cause the RTT to become larger, thereby reducing the rate of Pacing.

In LDAD, parts of large delay ACKs that may reduce the transmission rate of QUIC are blocked on a certain rule to prevent it from continuing to submit . Conventionally, the sending rate will drop adaptively when the network suffers from the congestion or a sudden interference. The drop in rate can prevent the network from further congestion, which is positive in the congestion case. But for the sudden interference, which is almost instantaneous, so the drop is not necessary. And the sudden interference is frequent in the wireless network, especially when its wirelss coverage is poor. On the one hand, our algorithm can improve congestion sensitivity, that is to say, the congestion can be detected more easily. On the other hand, by discarding ACKs with large dealy, the unnecessary drop of the rate caused by burst interference can be avoided.

3.2 Overview of LDAD

The algorithm can be deployed at BS/AP, where it just needs a ACK blocking model as detailed in Fig. 5. Moreover the algorithm can be easily transplanted to the QUIC server.

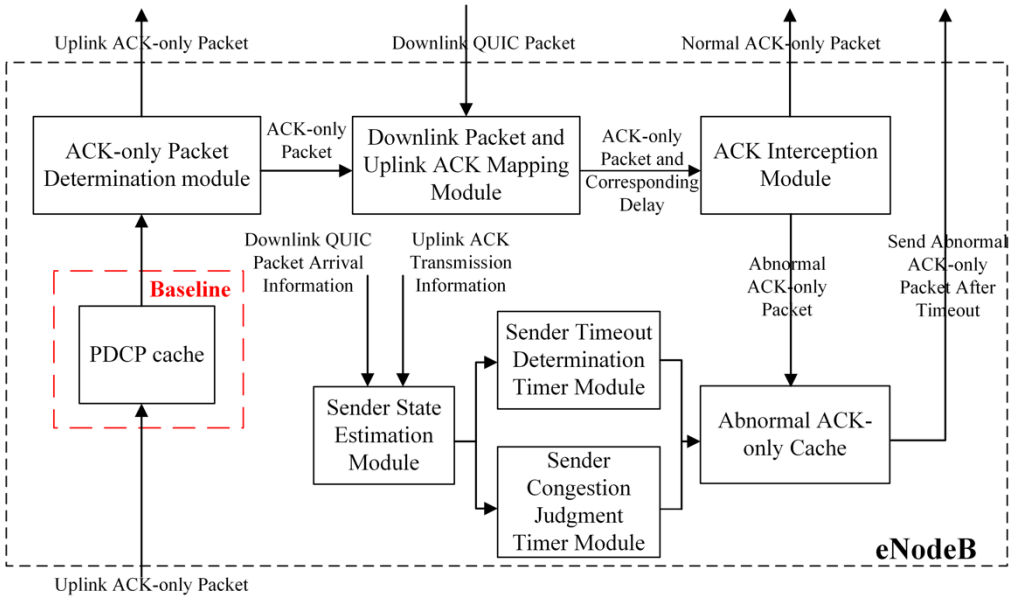

Fig. 5. Algorithm flow.

As shown in Fig. 5, the packets will be judged by several modules: ACK-only Packet Determination Module, Downlink Packet and Uplink ACK Mapping Module, ACK Interception Module, Sender Congestion Judgment Timer Module, Sender Timeout Determination Timer Module and State Estimation Module. The purpose of this algorithm is to generate a mapping of downlink data packets and uplink ACKs, and then calculate the current instantaneous RTT and intercept the ACK which with abnormally large delay RTT, so as to prevent the QUIC

transmission rate from deteriorating when there is a sudden interference in the current wireless network. By the way, there is a different processing flow for uplink and downlink packets.

For the downlink packet received by the current BS, it will be judged whether is a QUIC packet according to the port number. Then it enters the Downlink Packet and Uplink ACK Mapping Module if it is a QUIC packet, where it is joined with the current packet to the mapping table according to the judgment mechanism of the module. Then it enters the State Estimation Module to update the parameters, and finally goes into the timer module to perform necessary timer updates after updating the parameters.

For an uplink packet, the process is approximately the same. The BS determines the type of the received uplink packet. If it is a QUIC packet, the ACK-only Packet Determination Module is activated, and the ACK-only packet estimation algorithm in the module determines whether the current packet is an ACK-only packet. If it is an QUIC ACK-only packet, then it enters the ACK Interception Module, where the instantaneous RTT of the current ACK is calculated according to the mapping table. The mapping result will determine forwarding the ACK-only packet directly or not. The subsequent process is the same as the downlink process. That is, the packet enters into the State Estimation Module and timer module orderly. The function of Abnormal ACK-only Cache in the module is to store the ACK-only packet currently intercepted. The ACK-only packet reckoned as with large delay is purposely cached instead of being directly discarded, in case of that the QUIC server may not receive any ACK for a long time, which will be more harmful than receiving a delayed one. If that happens, the congestion window will be completely occupied and the data exchange process between the QUIC server and the QUIC will be totally stucked, resulting in a very poor user experience.

4 Experimental results

In order to verify the effectiveness of the LDAD, tests are performed using the designed QUIC semi-physical simulation platform with the LTE and WLAN scenario respectively. To be more specific, we implement the algorithm at eNB in the LTE scenario and at the AP in the WLAN scenario respectively, and then test the QUIC average throughput with various BLER/PER, with and without using the LDAD. Here the QUIC server throughput is defined as the total amount of data transmitted by the QUIC server per second. It is worth mentioning that the average throughput is obtained after abandoning the outliers due to the systematic error. The simulation structure refers to the LTE design document of the NS3, as shown in Fig. 6.

1) The performance in LTE: We first test the idea in the scenario of LTE, as is shown in Fig. 6. The LTE scenario includes: a remote host, an eNB, and an UE, whose nodes are all stationary. The remote host communicates with the UE through the eNB. The remote host in the simulation scenario is connected to the QUIC server of the QUIC platform, and the UE is connected to the QUIC client of the QUIC platform. We use the virtual box to operate two virtual machines running QUIC and NS3 respectively. The QUIC client requests 256 kB data from the QUIC server. The uplink and downlink transmission delays of the LTE fixed network are all set to 50 ms. Testing is performed in the case of 5%, 10%, 15%, 20%, 25%, and 30% BLER of the LTE wireless network. The change average throughput rate of the QUIC server with the various packet lost in physical layer of the LTE wireless network is shown in Fig. 7.

The blue line (solid or dotted) in the Fig. 7 are the changes of the average throughput of the QUIC server with the BLER when the algorithm is used or not respectively. From the figure, it can be seen that the throughput decreases with the increase of the BLER in both cases, the reason

is that as the probability of packet loss at the physical layer increases, the number of retransmissions of the underlying data packet increases, and the end-to-end instant delay jitter increases and the time delay fluctuates more frequently. As can be seen from the figure, compared with the original case where the algorithm is not used, almost all the algorithms used in the current test scenario can bring gains, especially when the BLER is 0.05, 0.1, and 0.15, and the gain is about 30%. Obviously, the gains brought by using the scheme in the LTE scenario are significant.

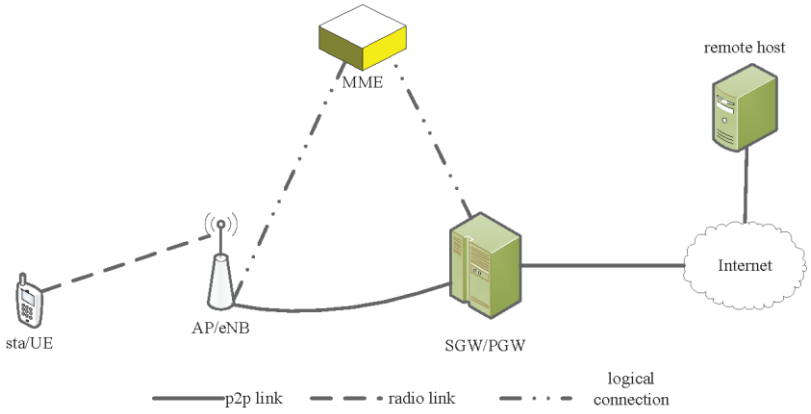

Fig. 6. The simulation scenario without background streams.

2) The performance in WLAN: As shown in Fig. 6, the WLAN consists of a remote host, a sta (station) and an AP, all stationary. Remote host interacts with sta through the AP. The remote host in the simulation scenario is connected to the QUIC server in the platform, and the sta connects to the QUIC client of the QUIC platform. To simulate the actual situation, the remote host is connected to the AP through a point-to-point channel in the current scenario. The uplink and downlink transmission delays of the point-to-point channel are set to 60 ms and the transmission rate is 5 Mbps. The WLAN network uses the 802.11b protocol, and the WLAN transmission bandwidth is a constant 1 Mbps. The QUIC client requests 256 kB data from the QUIC server. The packet loss rate at the bottom layer (physical layer) of the WLAN network is 0%, 5%, 10%, 15%, 20%, 25%, and 30%, respectively. Under the condition that the tested 802.11b protocol has a constant 1Mbps transmission rate, the average throughput of the QUIC server changes with the various physical layer packet loss rate as follows:

As shown in Fig. 7, the red line (solid or dotted) indicates the variation of the QUIC server's throughput with the physical layer packet loss rate when the algorithm is not used or not. Similar to the test in the LTE scenario, the QUIC server throughput under two cases both decreases with the increase of the packet loss rate at the physical layer. The transmission performance of the performance is basically the same when the packet loss rate at the physical layer is small. However, as the packet loss rate increases, the performance gain after the use of the LDAD becomes obvious, especially in the case of a packet loss rate of 0.15, the usage scheme brings about a 10% performance gain.

In summary, the proposed algorithm can bring about certain gains in the LTE scenario or WLAN scenario and the gain of the algorithm in WLAN is about 10%, and the gain is 20%~35% in LTE.

3) The performance with estimation error: There is a high probability of an uplink ACK-only packet loss in the actual poor wireless environment. At this time, the estimation error would occur in the algorithm. Therefore, the impact of the estimation error on the performance of the algorithm is tested. In the tests, the QUIC server throughput performance with some estimation error in the scenario of LTE is shown.

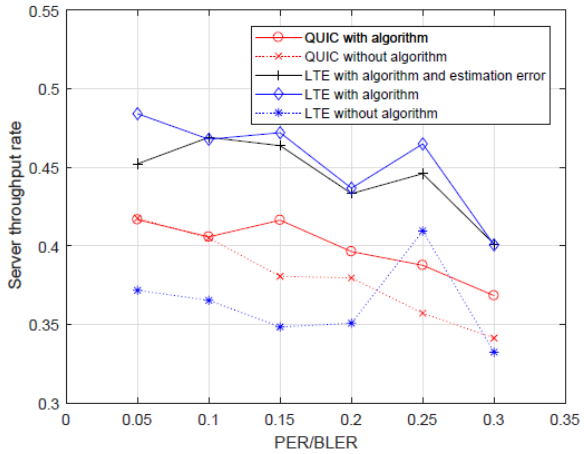

Fig. 7. The algorithm performance in WLAN and LTE.

The simulation settings are basically the same as before. In NS3, we manually set an estimation error of 20% when the node receives the upstream ACK and calculates the current RTT. Here, the estimation error is defined as 20% delay disturbance. For example, if the current calculated instantaneous RTT is 100 ms, there is a 20% probability that the instantaneous RTT submitted to the QUIC is 120 ms.

From the Fig. 7, the black solid line indicates the results of using algorithm with 20% estimation error in LTE. It can be seen that using the algorithm can bring a large gain no matter there is an estimation error or not. The RTT of the wireless segment can obtain real-time updated data at the eNB, and therefore can accurately estimate that there is a certain estimation bias in the fixed RTT currently set. Based on the performance analysis above, it can be seen that deviations in fixed RTT will only affect the threshold setting of the two timer modules. This will cause the duration of the algorithm to be several milliseconds longer than that of the ideal case. Compared with the second-level test time, it is apparent that the wired-link RTT exists. The deviation has little effect on the performance of the algorithm. The simulation test results are in good agreement with the performance analysis. Therefore, it can be concluded that under the current settings of the simulation scenario parameters, estimation error of the wired-link RTT has little effect on the performance of the algorithm.

5 Conclusion

We design and implement a QUIC simulation system that combines DASH, QUIC, and NS3 platforms and test its effectivity. DASH is used to send service requests and play successfully received video segments. QUIC plays the role of transmitting DASH-based video

service streams; and NS3 is used to provide emulated networks including WLAN and LTE. After the simulation platform passing some initial verifications, we use the platform to implement the proposed LDAD algorithm on the BS of the LTE and the AP of the WLAN, and test its performance. According to the result, the proposed algorithm can bring significant gains in poor wireless environment. Among them, the performance gain of the algorithm in the WLAN scenario is about 10%, and 20% to 35% in the LTE scenario. The results show our semi-physical QUIC simulation plaform is powerful in investigating such problems. The platform can be improved to have more features to modify and observe the interactions between QUIC and user applications in the future.

Acknowledgment. The research on the protocols and QoS of networks is supported by the National Science Foundation of China (No.61671353), and Huawei Technology Co., Ltd..

References

[1] Tao H, Ansari N, et al.: On accelerating content delivery in mobile networks. IEEE Communications Surveys & Tutorials. pp. 1314-1333. (2013)

[2] Stenberg, Daniel.: HTTP2 explained. Acm Sigcomm Computer Communication Review. pp. 120-128 (2014)

[3] Wilk A, Iyengar J, Swett I, et al.: QUIC: A UDP-based secure and reliable transport for HTTP/2. IETF draft-tsvwg-quic-protocol-02. (2016)

[4] Bishop, M., Ed.: Hypertext Transfer Protocol Version 3(HTTP/3).IETF draft-ietf-quic-http-19 (2019)

[5] Alepuz I, Cabrejas J, Monserrat J F, et al.: Use of mobile network analytics for application performance design. IEEE Network Traffic Measurement and Analysis Conference. pp. 1-6 (2017)

[6] Scharf M, Kiesel S.: Head-of-line Blocking in TCP and SCTP: Analysis and Measurements. IEEE Global Telecommunications Conference. pp. 1-5 (2006)

[7] de Saxcé H, Oprescu I, Chen Y.: Is HTTP/2 really faster than HTTP/1.1?. IEEE Conference on Computer Communications Workshops. pp. 293-299 (2015)

[8] Chu J, Jain A, Cheng Y, et al.: Tcp fast open (2014)

[9] Thomson, M, Iyengar, J : QUIC: A UDP-Based Multiplexed and Secure Transport. IETF draft-ietf-quic-transport-10. (2018)

[10] Iyengar J, Thomson M.: Quic: A UDP-Based Multiplexed and Secure Transport. IETF draft-ietf-quic-transport-25 (work in progress) (2020)

[11] Megyesi P, Krämer Z, Molnár S.: How quick is QUIC?. IEEE International Conference on Communications. pp. 1-6 (2016)

[12] Das S R.: Evaluation of QUIC on web page performance. Massachusetts Institute of Technology (2014)

[13] Kakhki A M, Jero S, Choffnes D, et al.: Taking a long look at QUIC: an approach for rigorous evaluation of rapidly evolving transport protocols. Commun. ACM. pp. 86-94 (2019)

[14] Rüth J, Poese I, Dietzel C, et al.: A First Look at QUIC in the Wild. International Conference on Passive and Active Network Measurement. pp. 255-268 (2018)

[15] SzabóG, Rácz S, Bezzera D, et al.: Media qoe enhancement with quic. IEEE Computer Communications Workshops. pp. 219-220 (2016)

[16] Clark A.: QUUX: a QUIC un-multiplexing of the Tor relay transport (2016)