
SCITECH
RESEARCH ORGANISATION|
Volume 7, Issue 1
March 22, 2018
Journal of Information Sciences and Computing Technologies
www.scitecresearch.com/journals

# Performance Evaluation of HTTP/2 over TLS+TCP and HTTP/2 over QUIC in a Mobile Network

Yueming Zheng, Ying Wang, Mingda Rui, Andrei Palade, Shane Sheehan and Eamonn O Nuallain

School of Computer Science and Statistics
Trinity College Dublin
Dublin 2, Ireland

## Abstract

With the development of technology and the increasing requirement for Internet speed, the web page load time is becoming more and more important in the current society. However, with the increasing scale of data transfer volume, it is hard for the current bandwidth used on the Internet to catch the ideal standard. In OSI protocol stack, the transport layer and application layer provide the ability to determine the package transfer time. The web page load time is determined by the header of the package when the package is launched into the application layer. To improve the performance of the web page by reducing web page load time, HTTP/2 and QUIC has been designed in the industry. We have shown experimentally, that when compared with HTTP/2, QUIC results in lower response time and better network traffic efficiency.

**Keywords:** HTTP/2, QUIC, TCP, emulation, Mininet, Mininet-Wifi

## 1. Introduction

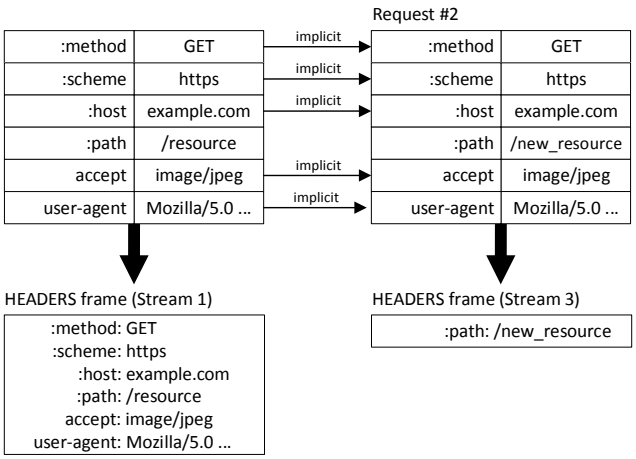
The web page response time is one of the most important criteria for the network performance. However, with the limited bandwidth and more and more higher speed requirements, the web page load time has caused a lot of concerns in the research and industry areas. How to save time to parse headers is important but challenging in the worldwide. To address this concern, HTTP/2 has been designed which covers SPDY in the industry; HTTP/2 uses multiplexing, server push, and header compression to reduce the web page load time. Nevertheless, traditional TCP limits performance of HTTP/2 because of the way connection establishes and the encryption mechanism. In order to answer these limitations and improve the performance, Google published QUIC, a UDP-based transport layer protocol.

This paper is to explore the difference of performance between HTTP/2 over TCP+TLS and HTTP/2 over QUIC using Mininet and Mininet-WiFi. Under various topologies, the difference of performance between these two protocol stacks will be evaluated.

The Hypertext Transfer Protocol (HTTP) is the most widely used data communication protocol for the World Wide Web in the past decades. However, the drawbacks of this solid protocol have been displayed more clearly, such as the limited concurrency ability, the long request queue with header blocking at the client side, etc. Especially, the high protocol overhead is the straw that broke HTTP to be replaced and HTTP/2 comes into being [2]. The data transmitted by HTTP/2 is based on the binary. Compared to HTTP/1.x, the big advantage of binary data

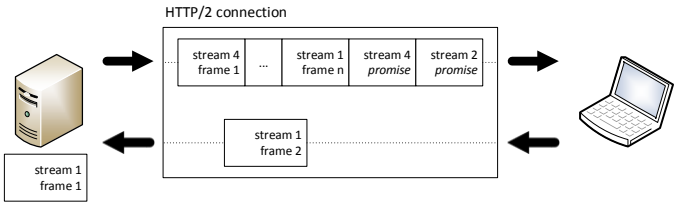
transmission is the smaller transmission volume instead of the HTML/Plain data, which means a lower payload [6]. Binary frames are also easier to parse and are less prone to error. The plain text frame is considered to deal with spaces, capitalization, blank lines and lines break when parsing, and binary frames does not have these problems. HTTP is a stateless protocol. In short, this means that each request must carry all the details, which the server needs, rather than having the server keep the metadata requested before. Because HTTP/2 does not change this paradigm, it also needs to do this (carry all the details). Therefore, the header of the HTTP request needs to contain identity data, such as cookies, and the amount of the data is growing over time. Each request contains the head of these large amounts of duplicate data, is undoubtedly a great burden. Compression of the request header will greatly reduce this burden. In the context of the mobile side, the performance is very obvious. HTTP/2 uses HPACK to compress the header and the process is shown in Figure 1[7].

**Figure 1. HTTP/2 Header Compression**




Another important feature about HTTP/2 is Server Push. The work of Server Push in HTTP/2 is to determine the client may also ask other resources when the server receives a client request for a resource. And then, these resources are sent together to the customer side, even if the client has not explicitly indicated that it needs these resources. The client can choose to put additional resources into the cache (so this feature is also called Cache Push), or it can send an RST-STREAM frame to reject any resources it does not want [9].

**Figure 2. HTTP/2 Server Push**




Also, HTTP/2 works on flow control. Whether the flow control rules can be applied or nor is determined by the type of the frame. The data frame is controlled by the flow; other types of frames are not controlled by the communication traffic. Therefore, some browsers want to get the size of the picture in advance. As a result, it can be used to design a better layout and reduce the reflow phenomenon. However, to meet the higher speed on the web

page, Quick UDP Internet Connections (QUIC) is published by Google. The transportation layer is mainly consisted by Transmission Control Protocol (TCP) and User Diagram Protocol (UDP). Compared with the handshake, acknowledgement, retransmission, and congestion control mechanisms in the TCP, UDP is a lightweight development choice; QUIC supports multiplexing connections between the client and the server over User Diagram Protocol (UDP), which means the lower transport latency, the less usage of the existed bandwidth, and the higher transmission efficiency in the network.

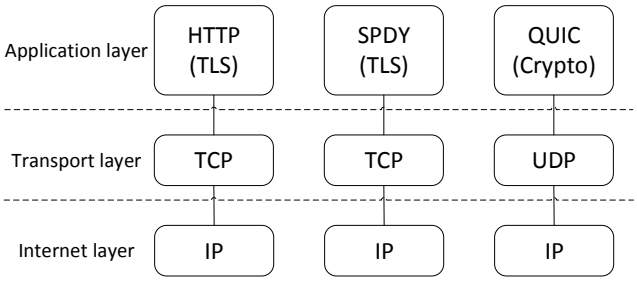
This paper will explore the performance difference between HTTP/2 over TCP+TLS and HTTP/2 over QUIC under Mininet and Mininet-WiFi. Technical details about QUIC and TCP+TLS will be introduced in the second part. And the experiment schemes will be displayed in the third section. The fourth part is about experiment results and relevant analysis.

## 2. Background

To provide a more responsive user interaction environment, the latency of the whole Internet needs to be reduced. However, the bandwidth is increasing over time with the unchanged round-trip time (RTT). The Internet needs a protocol with less delay and less retransmission time consumption to pass the entire Internet request, response, and interaction. From a technical perspective, the "middlebox" and the "firewall" will block or decrease the transmission speed in either TCP or UDP. Therefore, the protocol based on TCP or UDP can solve the problems which Internet encountered and to achieve the goals. Since Stream Control Transmission Protocol (SCTP) is DTLS-based (Datagram Transport Layer Security) requires too long time for delay in establishing a connection, approximately four RTTs, SCTP is not inappropriate for this case. Therefore, QUIC is launched into the industry world.

QUIC can be considered as a plan over UDP to resolve the bottleneck encountered by SPDY in TCP. The transmission content of QUIC is in two layers - the high-level is similar to SPDY, and the low-level is simulates the TCP-oriented connection characteristics and reliability on UDP and adds an encryption process, like TLS [1]. QUIC provides UDP-based multiplexing, ordered, and reliable streaming. It works in the same layer with HTTP but the core is to move the packet loss control work to the application layer. The protocol stack is shown in Figure 3.

**Figure 3. Protocol Stack**




Compared with SPDY, QUIC resolves the obvious drawbacks to improve network traffic communication from the points of view of TCP/UDP features and network security.

- QUIC solves the blocking problem by the multiplexing feature. In TCP, if one of the package queue loss, the entire transmission stream will be blocked.
- QUIC is based on UDP, which means it does not oriented connections with more flexible communication windows [3]. TCP is not good at congestion control, resulting in reducing bandwidth and increasing the overhead of serialization of the waiting time.
- Also, TCP retransmission mechanism brings and extra handshakes bring higher overhead on waiting time. QUIC uses multiplexing to decrease the resource redundancy.

- Moreover, "QUIC Crypto" plays an important role in the protocol stack [5]. In traditional decryption process (TLS), it needs to stay at waiting for the state until all resources are in place.

Compared with the traditional network simulator with heavy coding jobs, such as NS-3 and OMNET, Mininet and Mininet-Wifi are chosen as the emulators because of their lightweight coding works and easier latency configurations.

Mininet is a network emulator that is connected by a number of virtual hosts, switches, and routers that use lightweight visualization technology to make the system comparable to a real network. It is easy for Mininet to create a network that supports SDN, in which hosts work like a real computer and the user can use SSH to log in and start the application. In Mininet, the program can be sent to the specific port in the Ethernet layer, and the packet will be received and processed by switches and routers. With Mininet, the user can add new features, test the network, and deploy to the real hardware environment easily.

There is a list of tools working for Software Defined Network (SDN) simulation but a few can run on Software Defined Wireless Network (SDWN). Mininet-WiFi provides a solution for SDWN simulation.

## 3.Methodology

Based on the previous study, the evaluation of the performance of the following protocol stacks (Figure 4) will be executed on experimental schemes. For the upper layer, HTTP/2 is enabled on Google Chrome and data packet can be captured by Wireshark. In the transportation layer, TCP is chosen as the first experiment with the use of TLS as the security protocol. On the other hand, QUIC has to be set up on both client side and server side. To simulate both on the general network and in the mobile environment, four experimental topologies are designed as shown in Figure 5. The first experimental scenario is evaluated under the single server and the single client environment, and the second one is to test the same purpose in the mobile network. The third and the fourth scenarios are for multiple clients and a single server for achieving load balancing. In these topologies, experimental values are Time to Completion (ToC) and Bytes/Goodput.

**Figure 4. Experimental Protocol Stack**

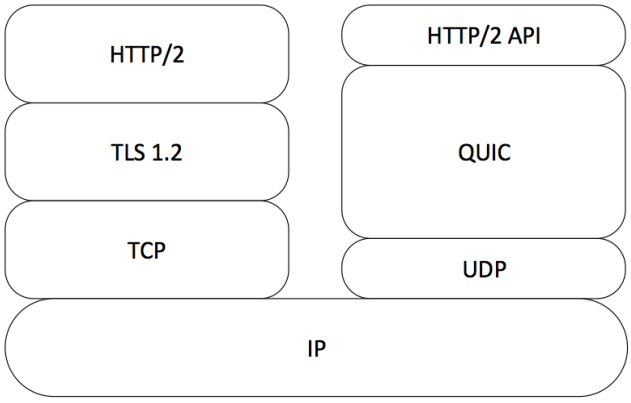
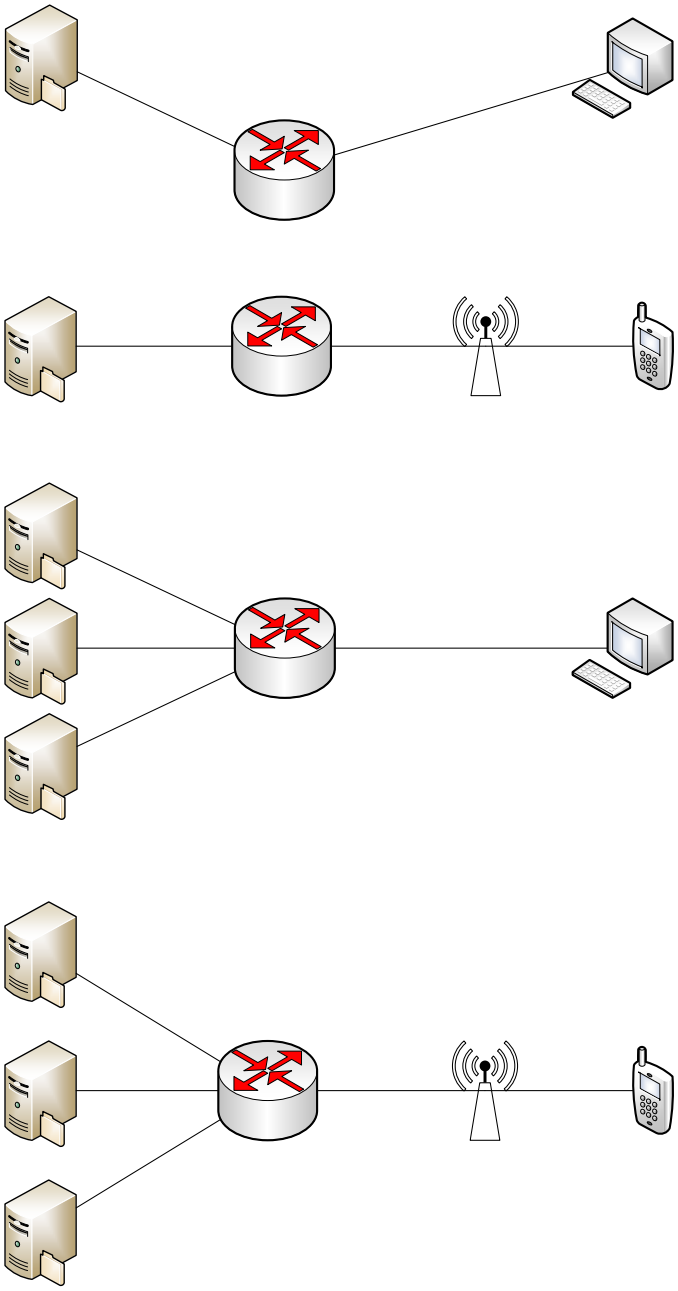


**Figure 5. Experimental Scenarios 1-4**



- **Time to Completion:** For evaluating the time cost of transmission, Toc will be captured. It reflects the transmission speed for these two protocols under the same network circumstance.

$$Toc=OverTime-StartTime$$

- **Goodput:** For transmission efficiency, Goodput will be captured. QUIC defines its own packet format and recovering mechanism that may cause a huge difference in Goodput. With the UDP multiplexing feature, QUIC has the significant data transfer efficiency.

$$Goodput=Server\ Pushed/(Server\ Pushed+Client\ Requested)$$

It is important to set the control variables in order to guarantee the experimental results more accurately, in this experimental investigation, the number of files and network latency will be controlled strictly. The latency will be set to 10 milliseconds and 100 milliseconds of each time; Different numbers of the file will be deployed in the server including 1 file, 10 files, and 100 files. For achieving more accurate experimental result, all files will be limited in the same size with no CSS or JavaScript, and the cache in the server and the cookies on the browser will be cleaned at every time.

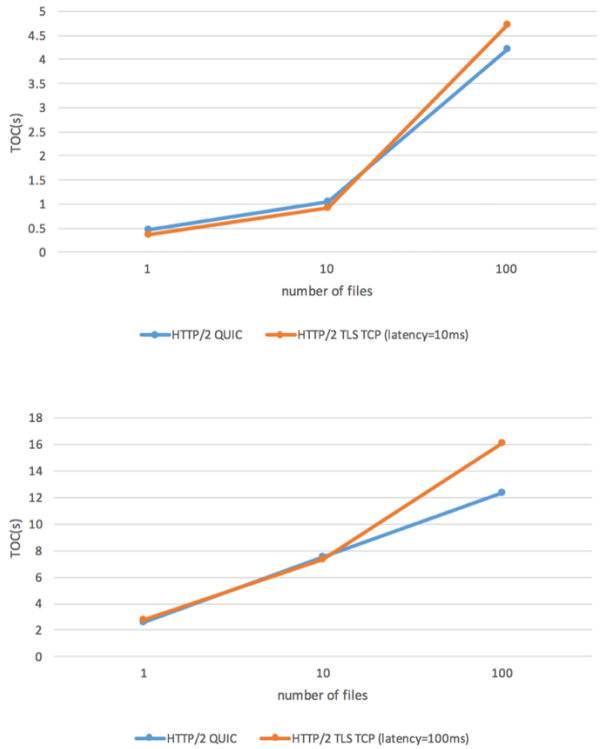
## 4. Results

Based on the Google's experiment in 2015, the packet pacing is the reason for QUIC to achieve the amount of percentage of the overall latency improvements, ranging from 50 percent to 80 percent. Also, the retransmission rate is reduced by 25 percent of QUIC compared to the traditional network protocol stack. Compared with the TCP + TLS, QUIC performs significantly better for slow connections with high latency and performs as good as TCP for fast connections with a low latency [4].

For the four different topologies designed for the experiment, the data can be analyzed in the following ways:

- For the first scenario, the experiment focuses on the basic or traditional connection method on the network. In the first scenario, TCP+TLS and QUIC are working under the single client and the single server environment. For the ToC difference with the latency changed for TCP+TLS and QUIC, the difference is clear. In the upper part of Figure 6, the trends are similar for these two protocol stacks. However, if the latency increases obviously, QUIC spends less time on the transportation with the growing number of files. This experiment result comes from the packet pacing feature of QUIC, where QUIC has a better performance with the higher latency configuration.

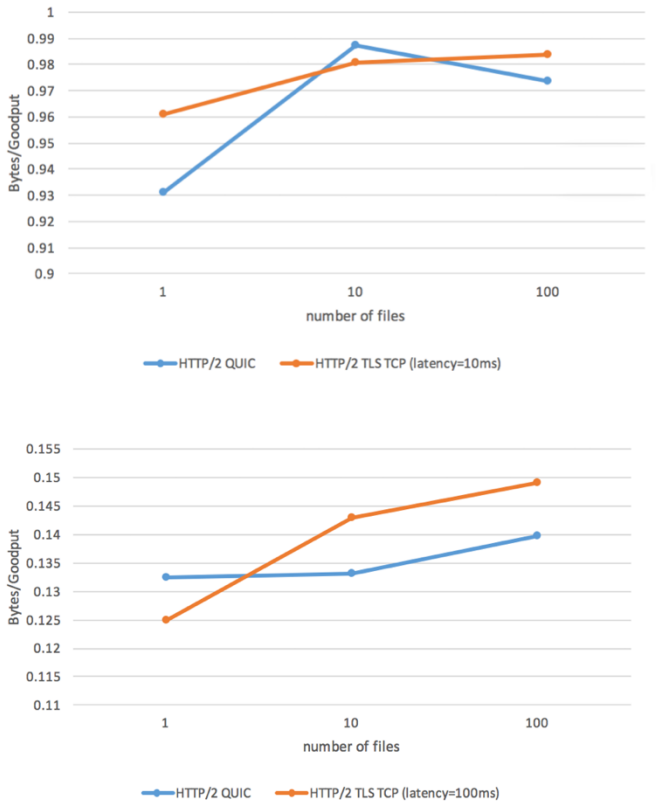
**Figure 6. Experimental results for Time to Completion comparison based on the different latencies for Scenario 1.**




- For the first scenario, the changing trends between 10ms latency and 100ms latency are totally difference. When the latency is 10ms, the traditional one is stable with the increasing volumes of files due to the

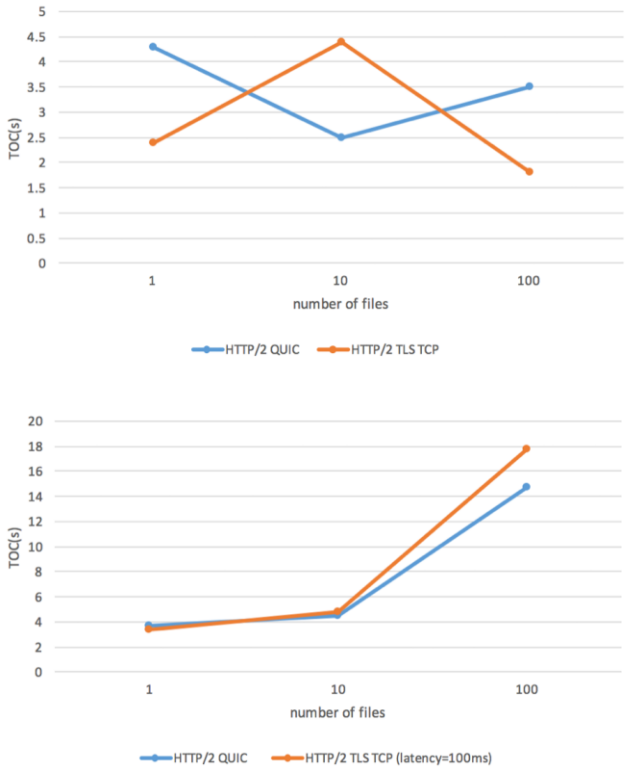
transmission mechanism in TCP. TCP deliveries all packets in their sending ability boundary while QUIC aims to provide the smaller packet loss rate. Therefore, when the latency is higher than the previous one, there is more packet loss in the transmission process of TCP; and QUIC always provides a higher packet transmission accuracy due to packet pacing and multiplexing features. The goodput results are shown in Figure 7.

**Figure 7. Experimental results for Goodput comparison based on the different latencies for Scenario 1.**



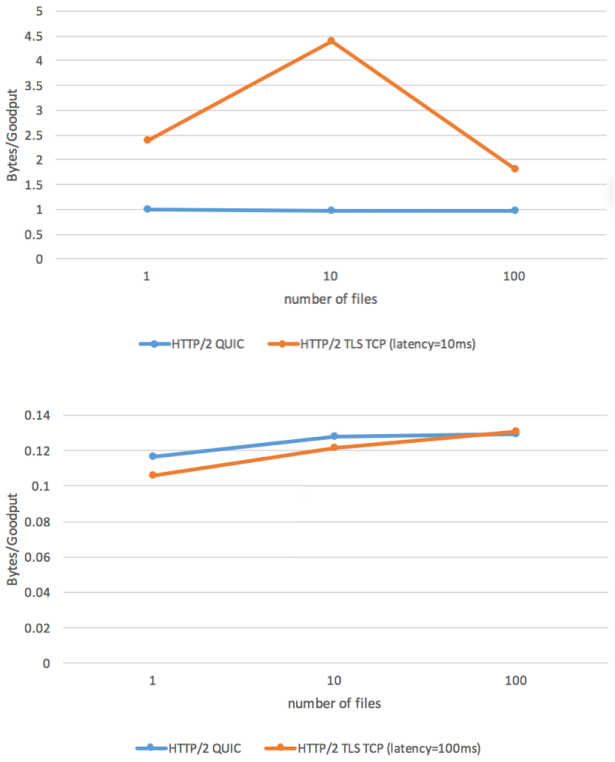

- For the second scenario, Mininet-Wifi is used for the evaluation under the mobile environment. With the moving nodes in this scenario, the data seems more different compared to the fixed nodes in the first scenarios. As shown in Figure 8, when the node is moving outside of the environment, it has a big impact on the transmission rate of TCP. In most cases, TCP spends more time on packets retransmission. On the other side, QUIC keeps its power and has a stable performance when the latency is set as 10ms. When the latency is set to 100ms, these two protocol stacks have the similar trends because all of them need time to react with the network due to the transmission latency. Also, QUIC has a better performance due to its features.
- For the goodput results in the mobile network, these two protocol stacks have the totally different performance on the goodput. It will be shown in Figure 9. In the diagram upper in Figure 9, QUIC always has the stable goodput due to the multiplexing to improve the usage efficiency of the channels. And TCP still needs to find the available area and set handshake again. When the experiment sends a bigger number of the latency, TCP and QUIC have the similar tendency by the different number of files. QUIC performs better when the client accesses more files on the server, due to its packet pacing and multiplexing features, it is able to decrease the handshake time and maxmize the advantages of the channels all the time.

**Figure 8. Experimental results for Time to Completion comparison based on the different latencies for Scenario 3.**




Observations: the raw data does not show a significant gap between TCP+TLS and QUIC because there is no 0 round trip time (RTT). The 0 RTT will have a great effect on QUIC performance [8]. In TCP, there are three times handshaking to set the connection between the client and the server, and four times handshaking after the transmission. If it wants to transmit packets to the same address, it has to set up connection again; the handshaking will bring additional overhead and QUIC discards this part to mitigate the overhead. In QUIC, it will use the connection ID to bridge the client and the server. In other words, it is unnecessary for QUIC to spend time and space on the connections for the same hosts. However, in this experiment, the both cache and the cookie in the Nginx server and the web browser will be cleaned. As a result, "no 0 RTT" feature of QUIC cannot be used in this experiment and the gap in the above diagrams will be larger.

**Figure 9. Experimental results for Goodput comparison based on the different latencies for Scenario 3.**




## 5. Conclusion

The experiment is to explore the difference of performance between TCP+TLS and QUIC based on the HTTP/2 protocol. To make sure the experiment environment is scientific and precise, the experiment chooses the variable controlling principle.

From the data received, the following conclusion could be summarized.

- The more latency, the better performance of QUIC.

  Because of the packet pacing feature of QUIC, QUIC saved transmission ability to guarantee the high packet transmission rate and reduce the loss rate.

- The number of smaller objects, the better performance of QUIC.

  Because of the multiplexing feature, the transmission channels will be used during the whole process. In TCP, there is one channel provided for one packet sending and it will cause low rate of channels usage during the transmission process. However, in QUIC, all of the channels will be seen as the entire entity to finish the jobs to reduce the space caused by the unused channels.

For the further work, the multiple clients will be investigated for the same purpose. The result of the multi-client-single-server may be similar with the single environment. Assuming QUIC provides the better performance in the multiple client's side is positive.

## References

[1] G. Carlucci, L. De Cicco, and S. Mascolo. Http over udp: an experimental investigation of quic. pages 609–614, 2015.

[2] H. De Saxce, I. Oprescu, and Y. Chen. Is http/2 really faster than http/1.1? In *Computer Communications Workshops*, pages 293–299, 2015.

[3] M. Fischlin and F. Nther. Multi-stage key exchange and the case of google's quic protocol. In *ACM Sigsac Conference on Computer and Communications Security*, pages 1193–1204, 2014.

[4] F. Gratzer. Quic-quick udp internet connections. Future In- ternet (FI) and Innovative Internet Technologies and Mobile Communications (IITM), 39, 2016.

[5] R. Lychev, S. Jero, A. Boldyreva, and C. Nita-Rotaru. How secure and quick is quic? provable security and performance analyses. pages 214–231, 2015.

[6] H. F. Nielsen, J. Gettys, A. Bairdsmith, E. Prud'Hommeaux, H. W. Lie, and C. Lilley. Network performance effects of http/1.1, css1, and png. *Acm Sigcomm Computer Communi- cation Review*, 27(4):155–166, 1997.

[7] R. Peon and H. Ruellan. Hpack: Header compression for http/2. 2013.

[8] G. Szabo´, S. Ra´cz, D. Bezzera, I. Nogueira, and D. Sadok. Media qoe enhancement with quic. In *Computer Communi- cations Workshops (INFOCOM WKSHPS), 2016 IEEE Con- ference on*, pages 219–220. IEEE, 2016.

[9] B. Thomas, R. Jurdak, and I. Atkinson. Spdying up the web. *Communications of the Acm*, 55(12):64–73, 2012.