Performance Analysis of QUIC-UDP Protocol Under High Load

Lin Qi[1,2], Zhihong Qiao[1,2], Aowei Zhang[1,2], Hui Qi[1,2(✉)], Weiwu Ren[1,2], Xiaoqiang Di[1,2,3], and Rui Wang[1,2]

[1] School of Computer Science and Technology, Changchun University of Science and Technology, Changchun, China
qihui@cust.edu.cn
[2] Jilin Key Laboratory of Network and Information Security, Changchun University of Science and Technology, Changchun, China
[3] Information Center, Changchun University of Science and Technology, Changchun 130022, China

Abstract. The QUIC protocol is currently recognized as a network protocol that is expected to replace TCP. It is characterized by encryption, multiplexing, and low latency. It was proposed by Google in 2013 and aims to improve the transmission performance of HTTPS traffic and achieve rapid deployment and continuous development of transmission mechanisms. QUIC has gone through many versions since its release, and previous researchers have compared the transmission performance of QUIC and TCP. In this paper, the pressure test tool vegeta is modified, and the modified vegeta is used to test the TCP and QUIC protocols in a high load environment and collect the results to evaluate the performance of different protocols. Experiments show that under high load network conditions, the TCP protocol has better performance than the QUIC protocol; IQUIC (IETF QUIC) performs better than GQUIC (Google QUIC) at lower attack frequencies; as the attack frequency increases, IQUIC Performance decreases faster than GQUIC performance.

Keywords: Load test · QUIC · TCP

1 Introduction

1.1 Development Status of QUIC

TCP has been the dominant transport protocol on the Internet for the last decade but several factors could reduce it in the coming years. With the use of TCP, it exposed some shortcomings. First, the Internet contains a variety of middle boxes that make some assumptions on specific TCP variants. Second, there are many in-kernel implementations of TCP and any change requires both engineering effort and negotiations within the IETF.

QUIC is a protocol that is expected to replace TCP. QUIC protocol is a UDP-based transport layer protocol proposed by Google in 2013, and its design goal

W. Li and D. Tang (Eds.): MOBILWARE 2020, LNICST 331, pp. 69–77, 2020.
https://doi.org/10.1007/978-3-030-62205-3_6

is to make the Internet faster and more secure [15–18]. QUIC incorporates the features of protocols including TCP, TLS, HTTP/2, etc., but is based on UDP transmission. One of the main goals of QUIC is to reduce the connection delay. When the client connects to the server for the first time, QUIC only needs 1 RTT (round trip time) delay to establish a reliable and secure connection. Compared to TCP + TLS, it requires 1–3 RTTs, which is faster. After that, the client can cache the encrypted authentication information locally, and can achieve a 0-RTT connection establishment delay when establishing a connection with the server again. QUIC also multiplexes the multiplexing function of HTTP/2 protocol at the same time, because QUIC is based on UDP, it avoids the head-of-line blocking problem of HTTP/2. Because QUIC is based on UDP and runs in the user domain instead of the system kernel, the QUIC protocol can be quickly updated and deployed, thus well solving the difficulties of TCP protocol deployment and update [1].

QUIC has gone through many versions since its inception [19]. With the continuous optimization of QUIC, some new versions of QUIC gradually no longer support Google Chrome. The latest version that supports Google is 0.10, while the version that does not support Googles QUIC is the latest version 0.14. This paper distinguishes GQUIC (supporting Googles version 0.10 QUIC) and IQUIC (IETF 0.14 version of Quic), and conducts experimental analysis separately.

1.2 Vegeta

Stress testing the server is a method of testing network performance. This test can find the performance bottleneck of the server by simulating a large number of requests, so as to make corresponding measures and response capabilities when encountering a large number of requests, and provide experimental basis and theoretical support for future work. This paper selects vegeta as a stress testing tool.

Vegeta is a multifunctional HTTP load test tool written in Go language, which provides command line tools and a development library. Vegeta can initiate requests of different frequencies to a certain website according to users' needs and collect experimental results, such as delay, throughput, etc., and then analyze server performance based on the collected results. Every request under high pressure is like an attack. The original Vegeta can only launch attacks against servers that support the TCP protocol. This paper transforms vegeta and follows the original Vegeta attack method. By changing the underlying protocol called when the request is initiated, the modified vegeta supports QUIC. The modified vegeta code can be downloaded by visiting [21]. In this paper, vegeta is used to test TCP, and the modified vegeta is used to test GQUIC and IQUIC and then observe the performance of these protocols under the same environment and different degrees of pressure.

The rest of the paper is structured as follows: Section 2 reviews related work; Section 3 describes the experiment; Section 4 conclusions, gives the evaluation results and future work.

2 Related Work

The safe and reliable transmission of data has always been a topic of great concern. Nowadays, the demand for networks is constantly increasing. It is very important to be able to transmit data without sacrificing security and reliability. TCP and QUIC are widely used protocols in data transmission and have received much attention in the industry. Since the QUIC protocol was proposed, many researchers have compared the performance of QUIC and TCP, and have related research in terms of throughput and fairness. Because the development of the QUIC protocol is so rapid, the research on the QUIC protocol in different scenarios has become extremely important. In order to study the performance of QUIC in different environments, [3–6,9] configured different network environments (bandwidth, delay, loss) to test the page load time which based on the QUIC protocol. Studies have shown that QUIC has low bandwidth and high latency And high loss network conditions show better performance. Yajun Yu et al. [2] evaluated the performance and fairness between QUIC and TCP. Studies have shown that when QUIC competes with TCP for bandwidth resources, it is affected by data loss rate and path buffer size. When the buffer is small, QUIC shows better performance and can obtain greater throughput; conversely, when the lossless network or larger buffer, QUIC is lower than the throughput obtained by TCP.

Due to the high latency and high loss characteristics of the space network, the research on the QUIC protocol under the space network is also worthy of attention. [11,12,20] According to the improvement of the ground network, the performance of the QUIC under the space network was evaluated. The results show that QUIC can reduce the overall page retrieval time of the spatial network. In the case of extended propagation and high packet loss rate, the performance of QUIC is better than TCP. This also shows the ability and future trend of QUIC protocol to solve space network problems. Sevket Arisu et al. [7] measured the transmission performance of QUIC in media streaming. The results show that when the network is more congested, QUIC can start media streaming better and provide better service and search experience for streaming media. There are other studies, for example [10] evaluated QUIC for the first time in a field scene, and [8] analyzed the performance differences of multiple versions of QUIC in different environments on the transport layer and application layer. In real time communications, such as voice and video conferencing, [14] measures packet loss, delay, and jitter, and proposes a performance measurement method that helps provide better QoE.

In addition to the transmission performance of the QUIC protocol, security and privacy are also guaranteed. QUIC combines the key negotiation function of TLS1.3 to encrypt all connections to prevent the middle box from tampering with the packet information. However, it hinders the future of QUIC protocol Development [13]. QUIC also has congestion control and loss recovery functions similar to TCP, and also provides rich signaling functions. In short, the advantages of QUIC protocol in low bandwidth, high latency and high loss

areattributed to the characteristics of QUIC, which can shorten the loading time of the page. Thanks to UDP-based stream multiplexing.

In summary, many studies have evaluated the performance differences of QUIC under different scenarios and different network conditions. Comparing the performance of QUIC and TCP, the QUIC protocol shows better performance in poor network environments. Due to the different network conditions in different scenarios (the bandwidth, delay and packet loss rate are different), the performance of the QUIC protocol will also fluctuate, and the throughput and bandwidth utilization will be greatly affected. Therefore, QUIC Whether the agreement can occupy an important position in the future needs to be continuously improved. There are also some studies to evaluate the fairness and resource utilization between QUIC and TCP. In this paper, we use the vegeta test tool to stress test the QUIC protocol, compare the network performance changes when different protocols are subjected to a large number of attacks, and evaluate the performance of different protocols. Experiments show that the performance of QUIC is lower than that of TCP.

3 Experiment

The experiment of this paper is to build a simulation environment through Ubuntu 18.04, in which the Ubuntu system is configured with 6G memory and a 4-core processor. We through the transformed vegeta, call the client of QUIC and TCP to initiate a request to the QUIC and TCP server. The page size requested in the experiment is 11000 byte. The relevant source code of this paper is on Github[1]. The experiment analyzes the performance of each protocol by continuously increasing the number of attacks per second, observing the average delay, throughput, and success rate under different protocols.

Without using vegeta and initiating only one request, the single delay of TCP, GQUIC and IQUIC and the number of data packets are shown in Table 1. The result is counted by Wireshark (a network packet analysis software). Although the same page is requested, because of using the different protocols, the delay and the total number of packets transmitted are not the same. As can be seen from the table, when requesting the experiment page, the number of data packets that TCP needs to transmit is smaller than GQUIC and IQUIC and IQUIC is slightly higher than GQUIC. In terms of latency, GQUIC and IQUIC are inferior to TCP, and IQUIC has obvious advantages over GQUIC.

Through a lot of experiments, this paper finally collected the experimental results of the stress test frequency range of 50–1000 times per second, and divided these results into three categories according to the performance of each protocol: low-frequency band, medium-frequency band and High-frequency band. The results of the experimental data collected in the low-frequency band are shown in Table 2. It can be seen that the TCP protocol is superior to the QUIC protocol in terms of average delay, throughput, and success rate at low-frequencies of 50–200 attacks per second; QUIC in terms of delay The protocol is 7–8 times

[1] https://github.com/qilin7070/quictest.git

Table 1. Single delay and number of packets for different protocols under no pressure.

Protocol	Single delay	Number of packet
TCP	0.0014	26
GQUIC	0.05	40
IQUIC	0.0076	49

Table 2. Average delay, effective throughput and success rate of different protocols under low-frequency attacks.

50 attacks/s	Protocol	Average delay (ms)	Throughput (Mbps)	Success rate (percent)
	TCP	1.05	4.20	100
	GQUIC	7.94	4.20	100
	IQUIC	7.46	4.20	100
100 attacks/s	Protocol	Average delay (ms)	Throughput (Mbps)	Success rate (percent)
	TCP	1.25	8.40	100
	GQUIC	8.40	8.38	99.77
	IQUIC	7.48	8.398	99.98
150 attacks/s	Protocol	Average delay (ms)	Throughput (Mbps)	Success rate (percent)
	TCP	1.26	12.60	100
	GQUIC	8.22	12.57	99.78
	IQUIC	7.59	12.58	99.82
200 attacks/s	Protocol	Average delay (ms)	Throughput (Mbps)	Success rate (percent)
	TCP	0.91	16.80	100
	GQUIC	11.35	16.72	99.55
	IQUIC	10.09	16.71	99.46

that of the TCP protocol, and IQUIC is slightly better than GQUIC in terms of latency. The results of the experimental data collected in the medium-frequency band are shown in Table 3. It can be seen that the TCP protocol is still better than the QUIC protocol in terms of average delay, throughput, and success rate at an intermediate frequency of 250–400 attacks per second, and remains very stable status, but the performance of the QUIC protocol gradually decreases. And it can be seen that IQUIC performance drops faster than GQUIC, so that it starts to be inferior to GQUIC in terms of average delay, throughput, and success rate. The results of the experimental data collected in the high-frequency band range are shown in Table 4. At this time, IQUIC has been unable to collect data normally and frequently reports errors. Therefore, the experimental results of IQUIC have not been added to the table. When the attack frequency is 600–1000 times per second, the performance of the GQUIC protocol drops sharply. This conclusion can be seen from the throughput and success rate of the GQUIC. Although the average GQUIC delay in this band is very low, it does not mean that GQUIC is performing well, because the average delay at this time is the result of a large number of error requests.

The average delay line chart of the experimental results (see Fig. 1). The horizontal axis is the attack frequency and the vertical axis is the average delay time. Because the success rate of the high-frequency band QUIC protocol drops

Table 3. Average delay, effective throughput and success rate of different protocols under medium-frequency attacks.

250 attacks/s	Protocol	Average delay (ms)	Throughput (Mbps)	Success rate (percent)
	TCP	0.87	21.00	100
	GQUIC	10.78	20.98	99.89
	IQUIC	13.95	20.97	99.84
300 attacks/s	Protocol	Average delay (ms)	Throughput (Mbps)	Success rate (percent)
	TCP	0.81	25.20	100
	GQUIC	11.95	25.17	99.87
	IQUIC	19.60	25.06	99.43
350 attacks/s	Protocol	Average delay (ms)	Throughput (Mbps)	Success rate (percent)
	TCP	0.87	29.40	100
	GQUIC	18.56	25.37	99.89
	IQUIC	26.48	25.34	99.37
400 attacks/s	Protocol	Average delay (ms)	Throughput (Mbps)	Success rate (percent)
	TCP	0.95	33.60	100
	GQUIC	28.59	33.49	99.66
	IQUIC	35.06	33.29	99.09

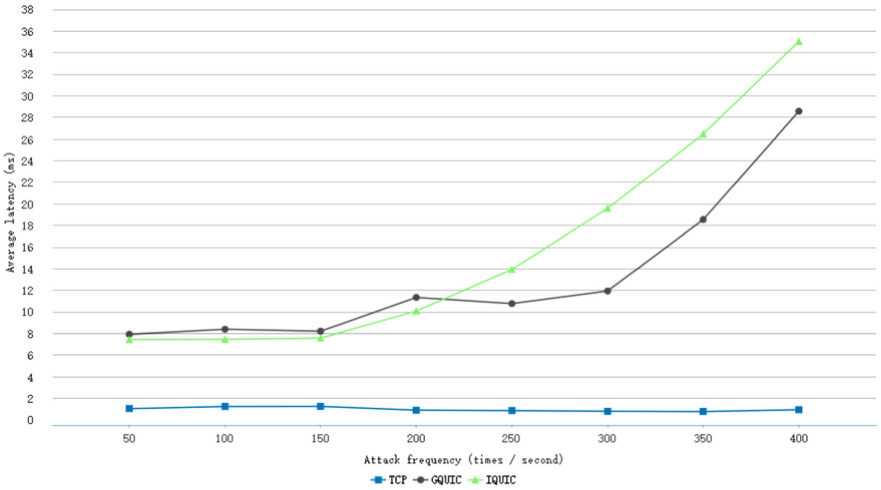

Fig. 1. The average latency of TCP, GQUIC, and IQUIC under different frequency attacks.

sharply, the average delay obtained at this time is the average delay in the case of failure, which has no practical significance, so the line chart only includes the experimental results of low-frequency band and intermediate frequency. It can be seen from the figure that when the attack frequency is less than 150 times per second, the average delay of IQUIC and GQUIC both remain relatively stable, and the average delay of IQUIC is slightly better than GQUIC. When the attack frequency is greater than 150 times per second, the average delay of IQUIC and GQUIC started to increase. The average delay of IQUIC increased significantly

Table 4. Average delay, effective throughput and success rate of different protocols under high-frequency attacks.

600 attacks/s	Protocol	Average delay (ms)	Throughput (Mbps)	Success rate (percent)
	TCP	0.90	50.40	100
	GQUIC	9901	18.07	35.86
800 attacks/s	Protocol	Average delay (ms)	Throughput (Mbps)	Success rate (percent)
	TCP	0.88	67.20	100
	GQUIC	12018	9.07	13.49
1000 attacks/s	Protocol	Average delay (ms)	Throughput (Mbps)	Success rate (percent)
	TCP	0.87	84.00	100
	GQUIC	11895	4.84	5.76

faster than GQUIC and soon exceeded GQUIC, the average delay of GQUIC was more stable. The average delay of the TCP protocol remains very low and stable throughout.

4 Conclusion

Through the pressure test experiments of different protocols by the vegeta test tool, we can see that the TCP protocol is stable and far superior to the QUIC protocol. As an optimized QUIC protocol, IQUIC has better performance under no pressure and low pressure. As the pressure increases, the performance of GQUIC is more stable than that of IQUIC. In theory, because the QUIC protocol is better than TCP in terms of handshake, the QUIC protocol should be superior to TCP in terms of latency. This experimental result is different from the original assumption. In future work, the author will do more indepth research on TCP and QUIC protocols, and explore the reasons leading to this experimental result.

Acknowledgement. The manuscript was supported in part by the National Key Research and Development Program of China under Grant No. 2018YFB1800303 and the Science and Technology Planning Project of Jilin Province under Grant No. 20200401105GX.

References

1. Cui, Y., Li, T., Liu, C., et al.: Innovating transport with QUIC: design approaches and research challenges. IEEE Internet Comput. **21**(2), 72–76 (2017)
2. Yu Y, Xu M, Yang Y.: When QUIC meets TCP: an experimental study. In: 36th International Performance Computing and Communications Conference (IPCCC), pp. 1–8. IEEE (2018)
3. Biswal P, Gnawali O.: Does QUIC make the web faster?. In: 2016 IEEE Global Communications Conference (GLOBECOM), pp. 1–6. IEEE (2017)
4. Carlucci G, Cicco L D, Mascolo S.: HTTP over UDP: an experimental investigation of QUIC. In: Proceedings of the 30th Annual ACM Symposium on Applied Computing, pp. 609–614. ACM (2015)

5. Wang, P., Bianco, C., Riihijärvi, J., Petrova, M.: Implementation and performance evaluation of the QUIC protocol in Linux kernel. In: Proceedings of the 21st ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems, pp. 227–234. ACM (2018)
6. Nepomuceno, K., de Oliveira, I.N., Aschoff, R.R., et al.: QUIC and TCP: a performance evaluation. In: 2018 IEEE Symposium on Computers and Communications (ISCC), pp. 00045–00051. IEEE (2018)
7. Arisu, S., Begen, A.: Quickly starting media streams using QUIC. In: Proceedings of the 23rd Packet Video Workshop, pp. 1–6. ACM (2018)
8. Kakhki, A., Jero, S., Choffnes, D., Nita-Rotaru, C., Mislove, A: Taking a long look at QUIC: an approach for rigorous evaluation of rapidly evolving transport protocols. In: Proceedings of the 2017 Internet Measurement Conference, pp. 290–303. ACM (2017)
9. Papastergiou, G., Fairhurst, G., Ros, D., et al.: De-ossifying the Internet transport layer: a survey and future perspectives. IEEE Commun. Surv. Tutorials **19**(1), 619–639 (2017)
10. Rüth, J., Poese, I., Dietzel, C., Hohlfeld, O.: A first look at QUIC in the wild. In: Beverly, R., Smaragdakis, G., Feldmann, A. (eds.) PAM 2018. LNCS, vol. 10771, pp. 255–268. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-76481-8_19
11. Zhang, H., Wang, T., Tu, Y., Zhao, K., Li, W.: How quick Is QUIC in satellite networks. In: Liang, Q., Mu, J., Jia, M., Wang, W., Feng, X., Zhang, B. (eds.) CSPS 2017. LNEE, vol. 463, pp. 387–394. Springer, Singapore (2019). https://doi.org/10.1007/978-981-10-6571-2_47
12. Yang, S., Li, H., Wu, Q.: Performance analysis of QUIC protocol in integrated satellites and terrestrial networks. In: 2018 14th International Wireless Communications and Mobile Computing Conference (IWCMC), pp. 1425–1430. IEEE (2018)
13. Bulgarella, F., Cociglio, M., Fioccola, G., Marchetto, G., Sisto, R.: Performance measurements of QUIC communications. In: Proceedings of the Applied Networking Research Workshop, pp. 8–14. ACM (2019)
14. Jager, T., Schwenk, J., Somorovsky, J.: On the security of TLS 1.3 and QUIC against weaknesses in PKCS 1 v1. 5 encryption. In: Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, pp. 1185–1196. ACM (2015)
15. Cook, S., Mathieu, B., Truong, P., et al.: QUIC: better for what and for whom? In: 2017 IEEE International Conference on Communications (ICC), pp. 1–6. IEEE (2017)
16. Langley, A., Iyengar, J., Bailey, J., et al.: The QUIC transport protocol: design and internet-scale deployment. In: Proceedings of the Conference of the ACM Special Interest Group on Data Communication, pp. 183–196. ACM (2017)
17. Palmer, M., Krüger, T., Chandrasekaran, B., Feldmann, A.: The QUIC fix for optimal video streaming. In: Proceedings of the Workshop on the Evolution, Performance, and Interoperability of QUIC, pp. 43–49. ACM (2018)
18. Seufert, M., Schatz, R., Wehner, N., Casas, P., Quicker or not?-An empirical analysis of QUIC vs TCP for video streaming QOE provisioning. In: 2019 22nd Conference on Innovation in Clouds, Internet and Networks and Workshops (ICIN), pp. 7–12. IEEE (2019)

19. Casas, P.: Is QUIC becoming the new TCP? On the potential impact of a new protocol on networked multimedia QoE. In: 2019 Eleventh International Conference on Quality of Multimedia Experience (QoMEX), pp. 1–6. IEEE (2019)
20. Lopes, R.H., Franqueira, V.N., Rand, D.: Integration and Evaluation of QUIC and TCP-BBR in longhaul science data transfers. In: EPJ Web of Conferences, vol. 214, p. 08026. EDP Sciences (2019)