*AMSE JOURNALS-AMSE IIETA publication-2017-Series: Advances B; Vol. 60; N°1; pp161-173*
*Submitted Jan. 2017; Revised March 15, 2017, Accepted April 15, 2017*
*https://doi.org/10.18280/ama_b.600110*

# Web Optimization of QUIC under Wireless Network

*Huang Cheng, *Lv Yongbo

*School of Traffic and Transportation, Beijing Jiaotong University

P. R. China, No.3 Shangyuancun Haidian District Beijing 100044, (gaffer.c@foxmail.com)

## Abstract

With the development of wireless technology, many TCP-based HTTP applications are running slow, especially under the wireless environment. This paper first analyzes the causes of this situation, and then discusses the advantages of Google's experimental QUIC protocol in wireless transmission and finds that QUIC is easy to extend and deploy, providing a good framework for personalized optimization. This paper applies the QUIC framework and presents a QUIC optimization algorithm for the VPN or LAN scenarios, which mainly improves the congestion algorithm on the basis of QUIC. Through simulation tests, it is found in the wireless environment with high latency and high packet loss rate, the QUICMING-based HTTP applications perform significantly better than those based on TCP, SPDY and default QUIC.

## Key words

UDP, HTTP, wireless network, QUIC

## 1. Introduction

With the widespread use of wireless networks, a lot of associated problems have emerged, one of which is how to provide high-quality http services in a wireless network, especially in a weak wireless network environment. In a wired environment, these are usually provided by

the TCP on the transport layer. However, in a wireless environment, especially in a weak wireless environment, TCP is very slow. The TCP-based anti-congestion algorithm is not suited to a wireless environment filled with mutations [1]; instead, it is more suitable for the relatively stable access environment [2] [3]. The TCP retransmission is also designed to prevent router congestion, and due to the characteristics of wireless communication, the high air-interface latency and packet loss will seriously affect the overall transmission performance of TCP [4] [5]. In addition, TCP also has other issues like stream multiplexing. In fact, these problems have been identified and improvements have been made, like SCTP, SPDY and HTTP 2.0 (similar to SPDY), etc.[6] [7], but each method has its own shortcomings. The problem with SCTP is that it is a transport layer protocol that requires modification of the protocol stack of the operating systems. For SPDY and HTTP 2.0 [8] [9] [10], the inherent problems of TCP which they are based on cannot be resolved (TCP congestion algorithm, HOL effect problem and the difficulty in updating and deploying TCP, etc.).

To this end, following SPDY and HTTP 2.0, Google proposed a new protocol QUIC [11] [12]. The biggest difference between QUIC and the other two protocols is that it is based on UDP, which can avoid the inherent problems of other protocols based on TCP and does not have to modify the protocol stack of the operating system.

In this paper, we first analyze the characteristics of wireless environment and existing http transmission methods and describe in detail the advantages of QUIC and its framework. Then we improve the QUIC congestion algorithm, apply it in the QUIC framework and put forward QUICMING. In addition to all the advantages of QUIC, QUICMING also has made special optimization of VPN or LAN. If the network speed can be guaranteed, it will perform even better. Then we compare the overall performance of QUICMING and TCP, HTTP 2.0 and default QUIC in Web http services in the weak wireless network environment. Finally, we analyze the results.

## 2. Characteristics of Wireless Environment

There are various communication standards for wireless environment, among which common ones are 3G, 4G and wifi, etc. Although they are very different in the concrete implementation, they still have a lot in common due to the characteristics of the wireless channel, such as sudden big changes, possible great latency and packet loss rate at the air interface and centralized transmission of data packets [13]. The reasons for the above problems include path attenuation, wireless retransmission mechanism, wireless coding mechanism and wireless resource scheduling mechanism. In a wired environment, such unstable conditions would not usually occur in the first hop of the access network.

## 3. Existing Common Methods

Currently, the methods mainly used to provide HTTP services include: TCP, SPDY, HTTP 2.0 and self-implementation protocol, all of which are based on TCP. As TCP only provides streaming services, it has the simplest function and easy to implement, but it has the lowest efficiency in carrying the http protocol. SPDY and HTTP 2.0 are also based on TCP, but with a lot of optimizations, especially in the http field, but they cannot avoid the inherent problems of TCP.

### 3.1 Performance Analysis of TCP alone in the weak wireless environment

Due to the basic feature of TCP, a lot of problems cannot be solved. Next we will elaborate on the specific reasons behind the problems.

1. TCP is difficult to update. There are also many optimized wireless TCP congestion algorithms (WTCP etc.). Even if TCP has been well optimized (in fact, there are a lot of TCP optimizations, many of which are very efficient), the optimized TCP protocol stack is difficult to deploy on various machines, because TCP is on the transport layer, which is implemented by the operating system. Most operating systems are updated every few years (XP has been used for 16 years and is still in use), seriously dragging down the development of optimizations. The upgraded system may have better TCP capabilities, but at the same time may bring a lot of other problems, like the incompatibility with various hardware and

software. This has hindered the system from being upgraded in time - usually we will not be so willing to upgrade the system kernel and thus we will not upgrade the TCP version. What is more, communication technologies are also updated every few years, so the TCP optimization deployment is always slower than the development of communication technologies. so TCP optimization deployment speed is always slower than the development of communication technology. As the operating system is underlying software, usually the complex TCP optimizations will not be incorporated into the operating system. Many creative TCP optimizations exist only in theory and cannot be applied in daily life. SCTP has the same problem.

2. As TCP is implemented by the default of the operating system, it can hardly meet various personalized needs. There are a great number of methods of and papers on TCP optimization, but many of them are special optimizations for particular scenarios (e.g. wireless environment) or applications. For an operating system, it is almost impossible to put them within its own protocol stack, because in order to be convenient, stable and light, the operating system will usually only provide the most basic implementation. At present, in the most mainstream operating systems WINDOWS and Linux, TCP does not work very well in the weak wireless network.

3. TCP uses IP address couple as the connection identifier. When the wireless network connection fails, even if there are other backup links, there will still be a disconnection (for example: the wifi is disconnected, but there is still a 4G connection). When the wifi is disconnected, the context of TCP serving the transfer becomes invalid due to the identifier connection failure and connection must be reestablished, which, However, is completely unnecessary as 4G can still be used. The reestablishment is also a waste of resources.

## 3.2 Performance analysis of SPDY and HTTP 2.0 in the weak wireless environment

Both SPDY and HTTP 2.0 are TCP-based methods. Though they offer additional functions like transport stream multiplexing and header compression [14][15] and part of their

performance has been improved, they still have the TCP HOL effect – having the same problems as TCP, as listed in the above 1-3.

### 3.3 Self-implementation protocol

It can implement a private protocol by itself so that it can implement all the optimizations proposed in theory. But in that case, the workload will be huge, which can only be done by a large organization, such as an open source organization like Chromium.

## 4. Introduction to QUIC

Recently, Google has proposed an experimental protocol QUIC. Based on UDP, it moves the stream reliability assurance to the application layer [16]. Therefore, it can avoid the inherent problems of TCP. Meanwhile, its overall framework can be easily reused and specially optimized in various applications.

1. QUIC is easy to deploy. It is implemented in the user mode; in other words, it is an application, often known as APP on a wireless terminal. In this way, the part which was difficult to upgrade and deploy in the past can be quickly deployed in the form of application.

2. QUIC can meet personalized needs in an APP. A typical example is the optimized application in the weak wireless network that we are going to discuss later. The optimization in the application is only made for this specific app and does not affect the default behavior of other apps.

3. QUIC can get rid of many inherent limitations of the TCP protocol settings. Unlike TCP, which requires that connection must be based on IP address, QUIC can still maintain connection after the switching between 4G and wifi, which is very important in the weak wireless environment. At the same time, QUIC can avoid the HOL effect that SPDY (HTTP2.0) will have in the transport stream multiplexing.

4. Besides, QUIC has the overall framework of HTTP applications, including encryption, authentication, HTTP header compression and transport stream multiplexing, etc., which is much more convenient than implementing a UDP-based HTTP protocol by yourself.

5. QUIC add the size of an app. But the size is very small (less than 1Mb).

Below is a comparison of the features of TCP, SPDY (HTTP2.0) and QUIC:

Comparison between the Features of TCP, SPDY (HTTP2.0) and QUIC

| | TCP | SPDY (HTTP2.0) | QUIC | Self-implementation protocol |
|---|---|---|---|---|
| Header compression | General effect | Good effect | Good effect | Generally no effect |
| Supporting transport stream multiplexing | No | Yes | Yes | Generally yes |
| HOL effect after transport stream multiplexing | NULL | Yes | No | NULL |
| Encryption | Yes | Yes | Yes | Generally no |
| RRT established through encryption | 7 times | Better then TSL | Minimum | NULL |
| Difficulty in deploying optimizations | Difficult | Easy | Easy | Easy |
| Optimization at the application level | Difficult | Partially difficult | Easy | Easy |
| Difficulty in implementation | Easiest | Easy | Easy | Difficult |
| Support multiple operating system | Yes | Yes | Yes | Generally no |
| Support Ip address changing | No | No | Yes | Generally no |

From this table, we can see that QUIC is very suitable for carrying http services and easy to deploy and optimize.

## 5. Congestion Algorithm in the Wireless Network with a MinGS

As we mentioned earlier in this paper, there are numerous TCP optimizations. Some are wireless optimizations and some are TCP fast open for TLS. These optimizations can improve TCP performance in most scenarios, but are rarely used because they are for specific cases and not suitable for being integrated into the system. The optimization for the weak wireless network that we propose here is just a typical example [17].

In a wireless environment, sometimes there will be packet loss and high latency [18]. The default anti-congestion algorithm in the mainstream operating system has no good way to tell whether it is the lack of router capacity in the intermediate process or the wireless air interface

that causes the problem. In order to avoid congestion in the router, it will slow down in sending the packets. In addition, due to the high latency and the small initialized congestion window, this algorithm has really poor performance for HTTP with various small resource needs. There are also many optimized wireless TCP congestion algorithms, and the general idea is to use different ways to distinguish the reasons for latency and packet loss – wireless network or router [19] [20]. These methods are very different and can optimize the TCP performance in a wide range. Nevertheless, the TCP optimizations are still difficult to deploy, so the general wireless optimization can hardly be applied in reality. What is more, these algorithms are developed for implementation of TCP in the protocol stack, so they are intended to solve general problems rather than personalized optimizations for particular applications, and as a result, they are not the optimal choices for individual cases. Therefore, neither the general-purpose wireless optimization algorithms nor the personalized optimization algorithms for particular application scenarios for TCP can be widely applied easily. QUIC, however, is UDP-based and can be optimized for each application. It can be customized for each particular application and at the same time easy to deploy. This feature provides a suitable platform for personalized wireless optimization algorithms and easy deployment. Thus, we use this platform to achieve optimization at the application level and put forward QUICMING. Apart from all the advantages of QUIC, QUICMING can also make special optimizations for VPN and other environments.

In many cases, we know the environment in which the software is running, so we can set the minimum guaranteed speed (minGS) according to the characteristics of the environment and optimize the congestion algorithm accordingly. For example, we are in a wireless private network, a VPN with a guaranteed speed or an LAN. This is very common in many software and private network application scenarios. In this case, we can use a specific optimization.

These scenarios have one thing in common – no router congestion occurs at a certain network speed; in other words, all packet losses and latencies are caused by the wireless network. Therefore, We can use this feature to optimize. Keep the trffic constant. We can

make optimizations in two aspects – initialized sending speed and congestion algorithm. Here we introduce several concepts:

MinGS: depending on the wireless environment, which can be determined by configuration or other ways, like hardware information retrieval.

Minimum guaranteed interval (minGT): depending on the minGS and the network MTU.

$$\text{minGT} = \frac{\text{MTU}}{\text{minGS}} \quad (1)$$

Wireless congestion mode: under this mode, wireless network is congested and all data are sent at the minGT. Data are sent at a consistent interval rather than continuously to avoid the packet loss of the router caused by unstable sending interval. If an ACK indication of no packet loss is received for consecutive N times, this mode can be switched to the high speed mode. N is a customizable constant. N usually is 10.

High speed mode: under this mode, there is no congestion in the wireless network, and the linux's default CUBIC (WTCP is also good) congestion algorithm is used to control the traffic. In this mode, the network speed will be monitored at real time. If the speed is less than minGS, it will be switched to the wireless congestion mode.

## 5.1 Optimization of the initialized sending speed

As the minGS has been set, we can send data directly at the minGS; in other words, the initial mode is the wireless congestion mode, which can significantly improve HTTP performance in a high-latency network.

## 5.2 Optimization of the congestion algorithm

As there is a minGS, if the speed in the high-speed mode is less than the minGS, it means that at this time both the packet loss and the latency are caused by the wireless network. In this case, it does not have to reduce the speed, but to switch to the wireless congestion mode. The congestion window (cwnd) is not tuned by the CUBIC algorithm in the wireless congestion mode, but instead it is the packet decision actually sent in the wireless congestion mode. The congestion window is not subject to the adjustment by the CUBIC algorithm under the

wireless congestion mode; on the contrary, it depends on the number of packets actually sent under this mode. Under the wireless congestion mode, data are sent at the minGT, but the total amount cannot exceed the receive window on the other side; once the amount reaches the receive window, it will stop sending. In the wireless congestion mode, once there is an ACK indication of packet loss, the packet will be retransmitted again, but still at the minGT. This is because in the wireless congestion mode, all packet losses result from the wireless network and have nothing to do with the router. If the correct ACK indication is received for N times, it will switch to the high-speed mode. When it switches to this high-speed mode, the initial cwnd will be the cwnd in the wireless congestion mode (the number of packets sent but not confirmed) and CUBIC is going to enter the probing state, that is, ready to accelerate the speed.

The whole optimization process is summarized below:

Table 1. QUICMING Optimization Process

| | QUICMING optimization process |
|---|---|
| 1 | The client reads the information and calculates the minGT and minGS |
| 2 | It enters the wireless congestion mode |
| 3 | **WHILE TRUE THEN** |
| 4 | **IF** currently it is in the wireless congestion mode |
| 5 | Packets are sent at the minGT and the cwnd depends on the number of packets actually sent. |
| 6 | Upon receiving the ACK indication of packet loss, it resends the packet without having to reduce the speed. |
| 7 | After receiving N correct ACK indications, it switches to the high-speed mode. |
| 8 | **ELSE** |
| 9 | The initial cwnd is the number of packets actually sent in the wireless congestion mode. |
| 10 | CUBIC is about to enter the probing state |
| 11 | It monitors the real-time network speed. If the speed is less than the minGS, it will switch to the wireless congestion mode. |
| 12 | **END IF** |
| 13 | **END WHILE** |

This optimization process is simple and effective, and can be easily implemented under the QUIC framework. It is consistent with the simplicity and effectiveness requirements for the congestion algorithm.

## 6. Test Environment

In order to verify the effectiveness of QUICMINg, we perform a simulation test. The test environment is summarized as follows:

Table 2. Test environment table

| | Test environment |
|---|---|
| Test web resources | 100 requests, sent to 3 hosts, with a total download size of 1311KB |
| Network simulation method | Network emulator |
| Latency | 0-3000ms |
| Packet loss | 0-20% |
| Encryption | Encrypted (CA certificate-based encryption) |
| MinGS | 1mb |

We compare the general TCP, SPDY, general QUIC and QUICMING. We choose them because they are very representative. TCP is supported by all browsers; SPDY is supported by most advanced browsers, and as it is similar to HTTP2, we do not include HTTP2 in the test; general QUIC is included in the test to verify its own performance; QUICMING is used to verify our new algorithm.

In order to simulate a wireless environment, we use a network emulator. The network diagram is shown below:

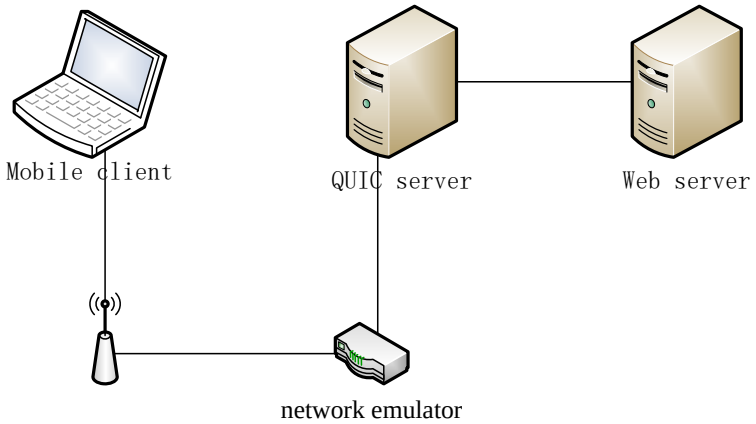


Fig.1. Network Testing Diagram

The mobile client uses QUICMING. Through the network emulator, the QUIC server sends data to the web server. The downlink process is the other way around. The QUIC server is powerful. It has powerful performance. Concurrency of QUIC server is better than most of TCP server (Apache Server).

## 7. Simulation Test Results

In the test, we choose different latencies and packet loss rates, and compare the time different protocols spend in loading the services (unit: s). After the simulation test, the results are as follows:

Table 3. Table for Simulation Test Results

| | Wireless simulation environment (Latency ms/ packet loss rate %) | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 0/0 | 0/5 | 0/10 | 200/0 | 200/5 | 200/10 | 500/0 | 500/5 | 500/10 | 1000/0 | 1000/5 | 1000/10 |
| General TCP | 12 | 12 | 20 | 19 | 25 | 32 | 29 | 49 | 63 | 51 | 73 | 135 |
| SPDY | 10 | 10 | 21 | 17 | 22 | 29 | 24 | 41 | 58 | 43 | 59 | 112 |
| General QUIC | 9 | 9 | 11 | 11 | 11 | 12 | 14 | 18 | 18 | 19 | 29 | 30 |
| QUICMING | 7 | 7 | 9 | 9 | 9 | 10 | 11 | 13 | 14 | 14 | 25 | 25 |

The packet loss rate is the packet loss rate of uplink and downlink. From this table, we can clearly see that when QUICMING is used, the web page loading speed is significantly greater than those of the others. From the results we can see that when the latency and packet loss rate are low, there is not much difference between the protocols, but when the latency and packet loss rate are increased, the differences are enlarged, which fully prove the outstanding performance of QUICMING in the weak wireless network environment. QUICMING has all the advantages of QUIC. The special optimizations of QUICMING work, when packet loss is increased in wireless network.

## 8. Conclusions

QUIC is not subject to any restriction of TCP in implementation, and that is why QUIC excels TCP and TCP-based SPDY in the overall performance. As the congestion algorithm of QUIC can be implemented without the TCP system, its optimization can also be easily implemented, and even in the application level, the optimization can still be quickly deployed.

This feature makes it very suitable for the application-level optimization, therefore, we propose QUICMING. At last, we prove the actual effects of QUICMING through test. One of the core advantages of QUIC is that it provides us with an upper frame for optimization in the application level. This gives us more inspiration than the conclusions of QUIC itself because we can make optimizations and modifications for various special scenarios without affecting other applications and can quickly deploy them in our actual practice.

## References


1. M.C. Chan, R. Ramjee, TCP/IP performance over 3G wireless links with rate and delay variation, 2005, Wireless Networks, Vol. 11, No. 1, pp.81-97.
2. Dalal, Purvang, Link Layer Correction Techniques and Impact on TCP's Performance in IEEE 802.11 Wireless Networks, 2014, Communications and Network.
3. M. Ivan, V. Ramos, Choosing a TCP version over static ad hoc wireless networks: wired TCP or wireless TCP?, 2013, In Next Generation Mobile Apps, Services and Technologies (NGMAST), Seventh International Conference on, pp 170-174.
4. A.V. Reddy, D. Kavitha, N. Kasiviswanath, TCP over On-Board IP Networks with High Bit Error Rate and Frequent Link Outages, 2016, In Advanced Computing (IACC), 2016 IEEE 6th International Conference on, pp 506-511.
5. A. Cardaci, L. Caviglione, F. Erina, Using SPDY to improve Web 2.0 over satellite links, 2016, International Journal of Satellite Communications and Networking.
6. H.J. Kim, G.S. Yi, S.W. Lee, A Research on the Performance Analysis of SPDY Protocol in Mobile Networks, 2014, Lecture Notes in Electrical Engineering, Vol. 19, No. 1, pp.199-206.
7. N. Arianpoo, V.C.M. Leung, A Smart Fairness Mechanism for Concurrent Multipath Transfer in SCTP over Wireless Multi-hop Networks, 2016, Ad Hoc Networks.
8. C. Luca, A. Gotta, A. Abdel Salam, Michele Luglio, Performance Evaluation of HTTP and SPDY Over a DVB-RCS Satellite Link with Different BoD Schemes, 2014, In Personal

Satellite Services. Next-Generation Satellite Networking and Communication Systems: 6th International Conference, pp 34-44.

9. S. Korakit, K. Piromsopa, When should we use HTTP2 multiplexed stream?, 2016, Computer Science and Software Engineering (JCSSE), 2016 13th International Joint Conference on
10. R. Aissaoui, O. Erdene-Ochir, A. Sabah, QUTor: QUIC-based Transport Architecture for Anonymous Communication Overlay Networks, 2016, In Qatar Foundation Annual Research Conference Proceedings
11. google, https://www.chromium.org/quic, access date 2017
12. google, https://www.chromium.org/spdy, access date 2017
13. D. Kumar, S. Aishwarya, A. Srinivasan, Adaptive Video Streaming Over HTTP Using Stochastic Bitrate Prediction in 4G Wireless Networks, 2016, ITU Kaleidoscope: ICTs for a Sustainable World (ITU WT), pp 1-8.
14. X.S. Wang, A. Balasubramanian, A. Krishnamurthy, How speedy is SPDY?, 2014, Usenix Conference on Networked Systems Design and Implementation, USENIX Association.
15. M. Varvello, K. Schomp, D. Naylor, Is the Web HTTP/2 Yet?, 2016, Passive and Active Measurement, Springer International Publishing.
16. H. Bakri, C. Allison, A. Miller, HTTP/2 and QUIC for virtual worlds and the 3D web?, 2015, Procedia Computer Science, Vol. 56, No. 1, pp. 242-251.
17. D.F. Wei, The Design Method of Embedded Web Based on Model-View-Controller Pattern, 2016, Review of Computer Engineering Studies, Vol. 3, No. 2, pp. 43-46
18. X.H. Ren, Q. Liu, Y.M. Zhang, The proportion of energy consumption structure prediction based on markov chain, 2015, Mathematical Modelling of Engineering Problems, Vol. 2, No. 1, pp. 1-4.
19. P. Sinha, T. Nandagopal, N. Venkitaraman, WTCP: a reliable transport protocol for wireless wide-area networks, 2002, Wireless Networks, Vol. 8, No. 2, pp. 301-316.
20. C. Casetti, M. Gerla, S. Mascolo, TCP Westwood: End-to-End Congestion Control for Wired/Wireless Networks 2002, Wireless Networks, Vol. 8, No. 5, pp. 467-479.