Poster: OpenQUIC: Software-defined Transmission like Building Blocks

Lizhuang Tan, Wei Su
NGIID, School of Electronics and Information Engineering, Beijing Jiaotong University
lzhtan,wsu@bjtu.edu.cn

Xiaochuan Gao*
China Unicom
gaoxc50@chinaunicom.cn

Wei Zhang
Shandong Computer Science Center (National Supercomputer Center in Jinan), Qilu University of Technology (Shandong Academy of Sciences)
wzhang@qlu.edu.cn

ABSTRACT

The experience of TCP tells us that protocol expansion through patching will cause protocol stack to become complex and fragile. In this poster, we introduced the idea of improving QUIC innovation, scalability and compatibility by standardizing QUIC components, which is called OpenQUIC. OpenQUIC encapsulates multiple functions related to a certain function into one module, and divides all modules into core modules and pluggable modules. The total size of the core modules is only 62KB. Through the combination of different modules, OpenQUIC can software-define transmission like building blocks, with stronger scalability and adaptability.

CCS CONCEPTS

• **Networks** → **Transport protocols**.

KEYWORDS

QUIC, Transfer protocol, Modular development, Componentization

ACM Reference Format:
Lizhuang Tan, Wei Su, Xiaochuan Gao, and Wei Zhang. 2020. Poster: OpenQUIC: Software-defined Transmission like Building Blocks. In *The 16th International Conference on emerging Networking EXperiments and Technologies (CoNEXT '20), December 1–4, 2020, Barcelona, Spain.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3386367.3431903

1 INTRODUCTION

Since Google first released gQUIC in 2013, academia and industry have continued to pay great attention to the research, measurement, standardization and deployment of QUIC[1]. As shown in Table 1, according to our incomplete statistics, there are more than 20 QUIC projects. Although these projects are working hard to follow up the stable draft of IETF QUIC, many design details of QUIC have not yet been determined, so there is no extensive code implementation. From a horizontal perspective, these solutions must fully support various flow control and congestion control(CC) algorithms. From a vertical perspective, the IETF QUIC iteration requires these solutions to consider support for subsequent versions. Therefore, it is necessary for us to sort out the relationship between the various functions of QUIC system, strip out some common interfaces that have nothing to do with version iteration, and use standardized components to accelerate QUIC innovation. Therefore, we are implementing an open source QUIC project called OpenQUIC[3].

Table 1: Incomplete statistics on QUIC projects.

Project	Language	Version	Open Source	MPQUIC	Logging	Cross-platform
ATS	C++	draft-29	●	○	○	●
Chromium	C, C++	draft-29	●	○	●	●
MSQUIC	C	draft-29	●	○	○	●
Quic-Go	Go	draft-29	●	●	○	●
quiche	Rust	draft-29	●	○	○	●
P-QUIC	C	draft-29	●	○	○	○
Picoquic	C	draft-29	●	○	●	●
XQUIC	C	draft-29	○	●	●	○

Note: ● represents support, and ○ represents not support. As of now, the latest version of IETF QUIC is draft-31.

2 OPENQUIC

As is shown in Figure 1, OpenQUIC includes core modules and pluggable modules. They are composed of some functions with specific functions. Through free combination like building blocks, OpenQUIC can be customized for function, size, logic and platform compatibility. Through general interface, OpenQUIC supports the horizontal expansion of the QUIC protocol, including congestion control algorithms, verification algorithms, encryption algorithms, and so on. Through the iteration of core modules, OpenQUIC supports the vertical evolution of QUIC, including the update of different versions of QUIC.

Modularity and pluggability are the main features of OpenQUIC. Therefore, OpenQUIC inherently supports horizontal expansion and vertical iteration. If device performance is good, OpenQUIC can select core modules and more pluggable modules to compile.

*Lizhuang Tan and Xiaochuan Gao contributed equally to this work.

CoNEXT '20, December 1–4, 2020, Barcelona, Spain
© 2020 Association for Computing Machinery.
ACM ISBN 978-1-4503-7948-9/20/12…$15.00
https://doi.org/10.1145/3386367.3431903

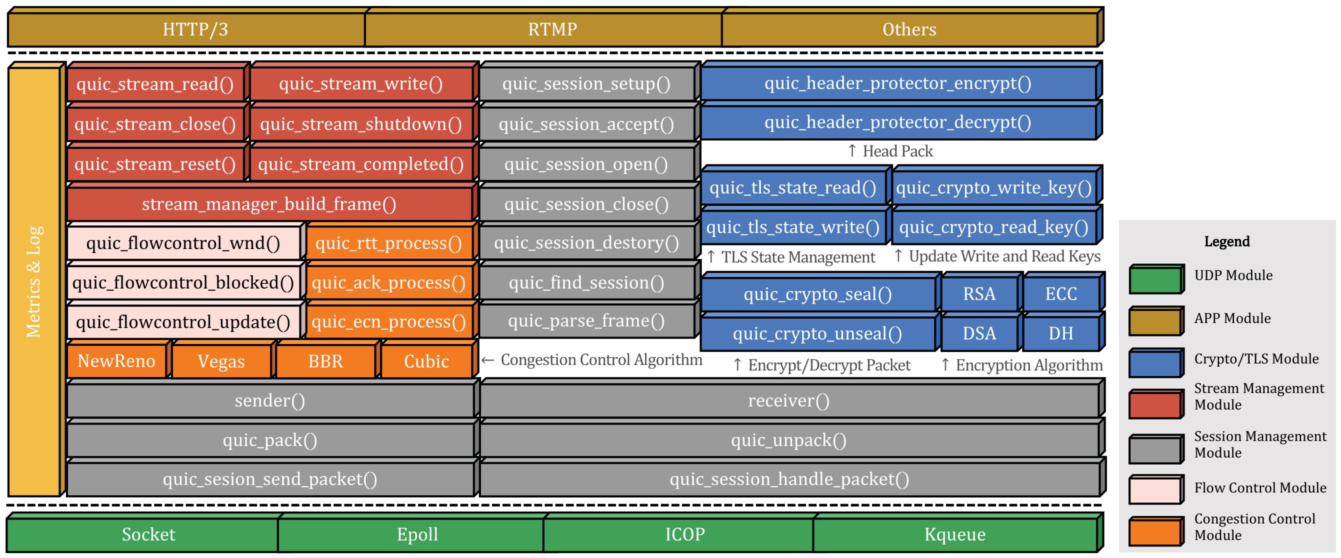

Figure 1: Key modules in OpenQUIC.

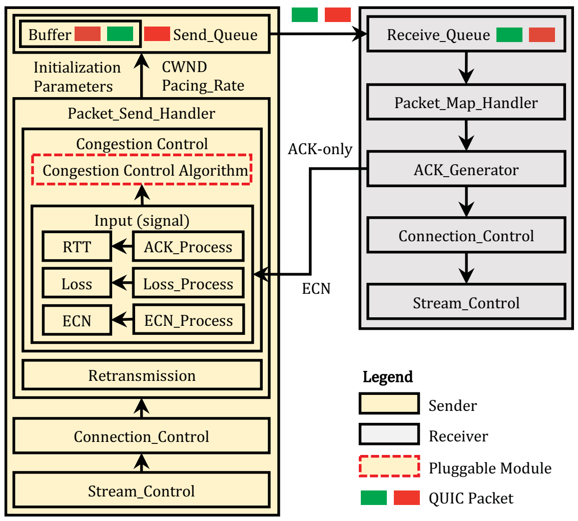

Figure 2: Key functions and interfaces of CC in OpenQUIC.

Congestion control (CC) is the core module of QUIC. As shown in Figure 2, we separate the measurement module from the CC parameter adjustment module. The measurement module is responsible for detecting and processing congestion signals such as RTT, Loss, and ECN. They may be called by different CC algorithms. Those CC algorithms, as pluggable modules, are put into the congestion parameter adjustment module, and their task is to calculate the appropriate CWND and Pacing-Rate.

Table 2 summarizes the CC algorithms supported by OpenQUIC, including the Cubic, New Reno, and BBR. By calling the measurement and adjustment interfaces, developers can quickly deploy self-developed CC algorithm. In different network quality and business scenarios, APP or OpenQUIC optimization engine can adaptively execute different CC algorithms and adjust CC parameters at the granularity of connection.

Table 2: CC algorithms in OpenQUIC.

Signal	Delay		Loss		ECN		Hybrid		BDP
Strategy	Vegas	Westwood	NewReno	Cubic	DCTCP	D2TCP	Libra	Illinois	BBR
State[1]	●	●	●	●	●	●	●	○	●

[1] ● represents completed, and ○ represents nearing completion.

Encryption algorithms such as RSA, ECC, and DSA can also be used by Crypto as pluggable modules, which have different complexity and size.

3 CONCLUSION

In this poster, we introduced the progress of OpenQUIC, a solution to accelerate QUIC innovation through standardized QUIC components. Currently, we have implemented OpenQUIC in C language[2]. In the future, OpenQUIC will support more programming languages and custom protocols, and can interoperate and be consistent with other open source QUIC implementations based on draft-31 or newer. For more information, please refer to [3].

ACKNOWLEDGMENTS

This work was supported by National Key R&D Program of China (No. 2018YFB1800305) and NSFC (No. 61802233).

REFERENCES

[1] Quentin De Coninck et al. 2019. Pluginizing QUIC. In *SIGCOMM '19*. ACM, Beijing, China, 59–74. https://doi.org/10.1145/3341302.3342078
[2] Xiaochuan Gao. 2020. OpenQUIC. https://github.com/OpenQUIC/OpenQUIC
[3] Lizhuang Tan and Xiaochuan Gao. 2020. OpenQUIC. http://www.openquic.net/