DOI:10.7544/issn1000-1239.2020.20190693
57(9):1864-1876,2020

互联网数据传输协议 QUIC 研究综述

李学兵[1,2,3]　陈　阳[1,2,3]　周孟莹[1,2]　王　新[1,2]
[1](复旦大学计算机科学技术学院　上海　201203)
[2](上海市智能信息处理重点实验室(复旦大学)　上海　201203)
[3](鹏城实验室　广东深圳　518066)
(xbli16@fudan.edu.cn)

Internet Data Transfer Protocol QUIC: A Survey

Li Xuebing[1,2,3], Chen Yang[1,2,3], Zhou Mengying[1,2], and Wang Xin[1,2]
[1](*School of Computer Science, Fudan University, Shanghai* 201203)
[2](*Shanghai Key Laboratory of Intelligent Information Processing, Fudan University, Shanghai* 201203)
[3](*Peng Cheng Laboratory, Shenzhen, Guangdong* 518066)

Abstract　QUIC is an Internet data transfer protocol proposed by Google as an alternative for TCP (transmission control protocol). Compared with TCP, QUIC introduces lots of new features to make it theoretically outperform TCP in many fields. For example, it supports multiplexing to solve the problem of head-of-line blocking, introduces 0-RTT handshake to reduce handshake latency, and supports connection migration to be mobility-friendly. However, QUIC's performance in the real world may not be as good as expected, because network environments and network devices are diverse and the protocol's security is challenged by potential attackers. Therefore, evaluating QUIC's impact on existing network services is quite important. This paper carries out a comprehensive survey of QUIC. We introduce the development history and the main characteristics of QUIC firstly. Secondly, taking the two most widely used application scenarios: Web browsing and video streaming as examples, we introduce and summarize domestic and international research analysis on the data transmission performance of QUIC under different network environments. Thirdly, we enumerate existing QUIC-enhancement work from the aspects of protocol design and system design. Fourthly, we summarize existing work on the security analysis on QUIC. We enumerate the security issues that are currently recognized by the academic community, as well as the researchers' efforts to address these issues. Lastly, we come up with several potential improvements on existing research outcomes and look forward to new research topics and challenges brought by QUIC.

Key words　Transmission Control Protocol (TCP); QUIC; HTTP; transport-layer performance; network security

摘　要　QUIC 是由 Google 提出的用于替代 TCP(Transmission Control Protocol)的互联网数据传输协议.它引入了许多新的特性,从而在理论上拥有比 TCP 更好的性能.例如,它通过多路传输解决了队头阻塞问题,通过 0-RTT 握手降低了传输层握手延时,以及通过连接迁移更好地对移动性提供支持.但是,现实生活中的网络环境和终端设备是多样性的,并且互联网中存在着各种各样的攻击,所以 QUIC

收稿日期:2019-09-25;**修回日期**:2020-02-14
通信作者:陈阳(chenyang@fudan.edu.cn)

在实际网络中的表现可能并不如预期.因此,探究 QUIC 对现有网络服务的影响是一项很重要的工作.首先介绍了 QUIC 的发展历史及其主要特性,并以目前使用最为广泛的 2 个应用场景——网页浏览和视频传输——为例,介绍并总结了国内外对 QUIC 在不同网络环境下的传输性能的研究分析.随后,从协议设计和系统设计 2 个方面列举了目前已有的对 QUIC 的优化工作,并对现有的对 QUIC 安全性分析的相关工作进行总结,还列举了目前学术界公认的 QUIC 所存在的安全性问题以及研究者为解决此类问题所作出的努力.最后,对现有研究成果可能的进一步提高之处进行了总结,并对 QUIC 带来的新的研究课题及其挑战进行了展望.

关键词 TCP 协议;QUIC 协议;超文本传输协议;传输层性能;网络安全

中图法分类号 TP391

QUIC 是一个由 Google 提出的、用于替代 TCP(Transmission Control Protocol)的数据传输协议[1].近几年来,在学术领域,对 QUIC 这一新型传输层协议的研究一直是一个热点.在 SIGCOMM,IMC,WWW,CoNEXT,CCS 等国际顶尖的网络、安全领域会议上[2-9]已经发表了一系列关于 QUIC 的传输性能、安全性等方面的研究成果.在工业界,QUIC 也已经得到了广泛的应用与部署.Google 在它的产品(Chrome 浏览器[10]、Google Cloud[11]等)中最早提供了对 QUIC 的支持.此外,一些内容分发网络(content delivery network, CDN)服务商,例如 Akamai[12]和 Cloudflare[13],也已经对 QUIC 提供了支持.截至 2018 年,互联网近 7.8%的流量使用 QUIC 进行网络通信[14].

与目前使用最为广泛的传输层协议 TCP 相比,QUIC 引入了许多新的特性.例如,它支持一条传输层连接上的多路传输和延时更低的握手.这些新特性使得 QUIC 在大部分场景下表现出比 TCP 更好的性能.例如更短的网页加载时间、更低的握手延时等.在安全性方面,QUIC 使用了比现有的 TLS1.2[15]更加安全的加密传输协议 TLS1.3[16],从而更好地保障通信安全以及用户隐私.然而,也有研究表明,QUIC 在特定场景下的表现不如预期,甚至不如 TCP.例如,QUIC 无法很好地适应高丢包率的网络[3,17].基于此,本文从安全性和传输性能 2 方面对现有的 QUIC 相关研究工作进行分类和总结,并对基于 QUIC 的优化方案进行介绍.

1 QUIC 概述

QUIC 最早由 Google 开发,正式发布于 2013 年,被称为 gQUIC[18].gQUIC 是一套相对独立的传输层协议,它同时保障了数据传输的可靠性和安全性,因此从功能性的角度出发,gQUIC 类似于 TCP+TLS.目前,gQUIC 已经得到了广泛的应用,包括 Chrome,IE,Safari,Firefox 在内的主流浏览器都支持 gQUIC.同时,Google 的网站服务器与云服务中也提供了 gQUIC 的支持.在 2013 年末,QUIC 的设计工作被互联网工程任务组(Internet Engineering Task Force, IETF)接管[19],并发布了正式的 QUIC 标准,我们称之为 IETF 版本的 QUIC.IETF 版本的 QUIC 与 gQUIC 最大的区别在于,它仅负责保障数据传输的稳定性,而将安全性的保障交由 TLS1.3.目前 IETF 版本 QUIC 的开源实现有 ngtcp2,aioquic 等.但是与 gQUIC 不同,IETF 版本的 QUIC 尚没有在互联网中大规模部署的例子.为了配合 QUIC 协议的标准化,Google 表示,将会在自己的服务器与浏览器中逐渐放弃 gQUIC 并使用 IETF 版本的 QUIC,gQUIC 的设计也正在逐步向 IETF 版本的 QUIC 靠拢.

此外,IETF 正致力于第 3 代 HTTP 协议,即 HTTP/3 标准的制定[20].HTTP/3 借鉴了其前身 HTTP/2[21]设计上的优点并从其不足之处中吸取了教训,最终基于 QUIC 实现了 HTTP 所规定的语义.HTTP/3 标准的确立将有利于 QUIC 在互联网上获得更为广泛的关注与应用.但是,鉴于 HTTP/3 的制定尚处于讨论阶段并且其性能主要决定于 QUIC 的表现,后文不会涉及到对 HTTP/3 的讨论.

1.1 QUIC 协议模型

QUIC 协议模型如图 1 所示.它向下使用了操作系统提供的 UDP(User Datagram Protocol)套接字,向上为应用层协议(例如 HTTP/2)提供了可靠且安全的传输通道.虽然 QUIC 在实现上基于传输层协议 UDP,但是它在协议设计上并没有依赖于 UDP 的特性,即没有使用 UDP 端口来标识一条传输层连接.QUIC 使用 UDP 的目的仅仅是为了保持

和现有网络的兼容性,因为目前互联网上的某些防火墙会屏蔽 TCP 和 UDP 之外的传输层协议.因此,尽管 QUIC 工作于传输层协议 UDP 之上,研究人员仍然普遍将它归类为传输层协议[1,5].

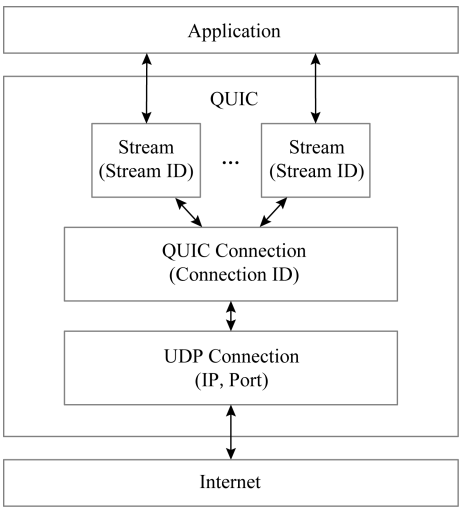

Fig. 1 QUIC protocol architecture

图 1 QUIC 协议模型

为了提供对移动性的支持,QUIC 放弃了 TCP/IP 网络中使用五元组(源 IP,源端口,目的 IP,目的端口,协议标识符)来唯一标识一条连接的方式,而是使用一个全局唯一的随机生成的 ID(即连接 ID)来标识一条连接.这样,当通信一方的物理网络发生变化时,例如从蜂窝网络切换到 WiFi 网络,在原先网络中建立的 QUIC 连接就可以无缝迁移到新的网络下,从而保证网络服务在用户切换网络的过程中不被打断.

1.2 QUIC 的协议特性

与传统的传输层协议如 TCP 相比,QUIC 引入了许多新的特性,其中最主要的是支持一条连接上的多路传输和更低的握手延时.

1.2.1 多路传输

QUIC 通过对多路传输的支持,解决了 TCP 中的队头阻塞问题[22].

如图 2 所示,在 TCP 连接中,它维护的是一条先进先出的通道,也就是说接收端必须严格按照发送端的顺序处理收到的数据,这样的设计导致了队头阻塞的问题.在图 2 的例子中,客户端先后向服务端发送了逻辑上没有相关性的数据包 2 和数据包 3.在接收端,即使数据包 3 先于数据包 2 到达,TCP 也不会先将数据包 3 中的数据交给上层应用处理,而是必须等待数据包 2 的接收完成才能进行后续处理.正是因为 TCP 通道的先进先出特性,导致对数据包 3 的处理被数据包 2 拖延,造成了不必要的延时.这是队头阻塞的一种情况,关于队头阻塞的更详细的研究见文献[23-24].

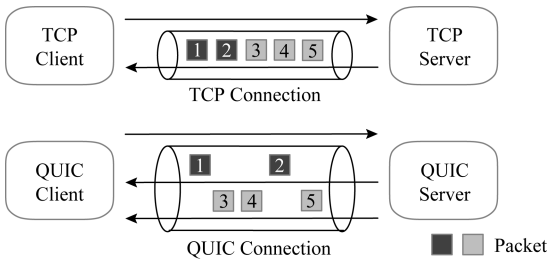

Fig. 2 Multiplexing vs singleplexing

图 2 多路传输与单路传输的对比

为了解决队头阻塞的问题,QUIC 增加了在一条传输层连接上的多路传输的支持.它在传输层连接下引入了流(stream)的概念.一条 QUIC 连接可以包含多个流,这些流各自保证了先进先出的有序性,但相互之间又不存在依赖关系.从这点来说,QUIC 中"流"的概念更接近于 TCP 中"连接"的概念,因为它们都是为上层应用提供了一条先进先出的通道.但是,QUIC 中不同流之间共享连接信息,所有的流共同参与拥塞控制、丢包检测以及数据加密.这样,和使用多条 TCP 连接相比,使用多个流的单条 QUIC 连接既达到了多路传输的目的,又节省了系统资源.

1.2.2 握手协议

QUIC 设计了自己的握手协议,并达到了比 TCP+TLS 更低的握手延时.

TLS 使用了对称加密和非对称加密相结合的方式为数据的安全性提供保障.其运行机制可以简单分为 2 步:1)客户端与服务端通过握手协议生成共享密钥;2)双方分别使用共享密钥进行数据的加密和解密.更详细的关于 TLS 介绍见文献[16-17].在 TLS1.2 中,握手协议通过 RSA 算法或者迪菲-赫尔曼算法完成.其中 RSA 要求通信双方首先交换各自的 RSA 公钥,再通过 RSA 加密传输一个新生成的共享密钥,因此完成握手需要 2 个往返时间(round trip time, RTT).而到了 TLS1.3,迪菲-赫尔曼成为唯一的密钥交换协议,通信双方可以直接通过对方的公钥和自己的私钥生成共享密钥,握手延时也从原来的 2 个 RTT 降为了 1 个 RTT.

图 3 列举了 QUIC 和 TCP 握手延时的区别.目前使用最为广泛的协议组是 TCP+TLS/1.2,在该

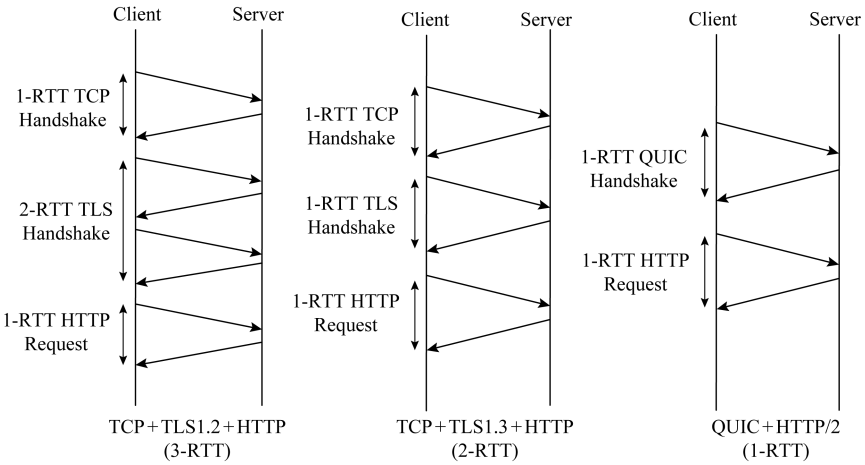

Fig. 3 Handshake latency of different protocols

图 3 不同协议握手延时的对比

情况下,客户端与服务端之间的握手至少需要消耗 3 个 RTT,包括 1 个 RTT 的 TCP 握手延时和 2 个 RTT 的 TLS 握手延时.在最新的 TCP+TLS1.3 协议组中,因为 TLS 的握手延时由原来的 2 个 RTT 减小到了 1 个 RTT,所以总的握手延时降为了 2 个 RTT.QUIC 在此基础上更进一步地进行了优化,它允许在建立传输层连接的同时进行 TLS 握手.也就是说,QUIC 的握手请求数据包中也包含了 TLS1.3 的握手请求数据,从而使整体的握手延时降为了 1 个 RTT.我们称 QUIC 的这种握手方式为 1-RTT 握手.

如果客户端在之前已经连接过服务端,客户端和服务端之间则会保留上次会话的相关加密信息,重用共享密钥.在这种情况下,客户端可以直接使用上次通信的共享密钥进行数据加密,而新的密钥交换过程会与数据传输同步进行.也就是说,数据传输不需要等待握手的完成,从而实现没有任何延时的握手.我们称 QUIC/TLS1.3 的这种握手方式为 0-RTT 握手.0-RTT 握手极大地降低了客户端的握手延时从而可以显著减少用户访问网络的延迟感,提升用户体验.

2 研究现状分析

图 4 对现有的 QUIC 相关研究工作进行了分类与总结.主要包括 3 个方面:QUIC 网络传输性能的测量、QUIC 性能的优化以及 QUIC 安全性的分析与改进.

在传输性能方面,研究人员最关心的是 QUIC 和 TCP+TLS 相比会有多大的性能提升,包括在不同网络环境和不同传输内容下的表现.在 QUIC 的性能优化方面,研究者一方面为 QUIC 协议带来了诸如多路径等新特性,另一方面在系统实现中使用内核旁路等新技术达到了比现有系统更高的 I/O 吞吐率.在安全性方面,研究人员使用多种方法对 QUIC

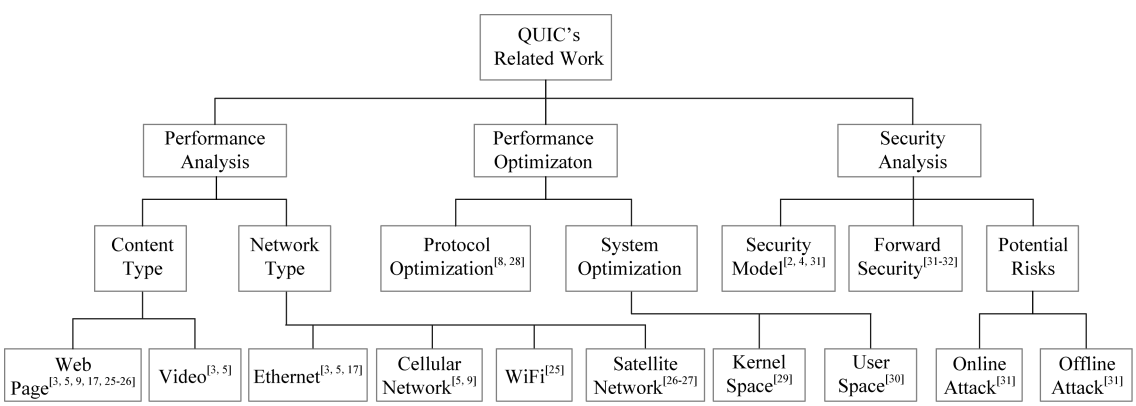

Fig. 4 Category of QUIC-related studies

图 4 QUIC 相关工作的分类

的安全性进行了建模分析,同时也指出了 QUIC 存在的一些安全隐患,并给出了相应的改进建议.

2.1 QUIC 的传输性能

自 Google 第 1 次发布 QUIC 以来,研究人员便对 QUIC 的网络传输性能产生了浓厚的兴趣.本节列举了近年学术界在 QUIC 性能分析上的工作.考虑到网络协议性能的分析离不开具体的应用场景,我们根据不同的应用场景对相关工作分为了 2 类,分别是网页浏览和视频传输.表 1 对 QUIC 性能分析的相关工作进行了总结.

Table 1 Related Work on QUIC Performance Analysis

表 1 QUIC 性能分析的相关工作

Reference	Publication Place	Publication Time	Transmission Content Type	Experiment Environment	Network Type/ Control Variables	Conclusion
Das[15]	MIT thesis	2014	Web Browsing	Simulated Network	Bandwidth, RTT	QUIC is faster in low bandwidth; TCP is faster in high bandwidth.
Langley, et al[5]	SIGCOMM	2017	Web Browsing, Online Video	Internet	Ethernet, Cellular Network	With the help of 0-RTT handshake, QUIC is faster than TCP. The performance gain is more obvious for desktop devices than mobile devices.
Kakhki, et al[3]	IMC	2017	Web Browsing, Online Video	Simulated Network	Bandwidth, RTT, Packet Loss Rate	QUIC's throughput degrades severely in high bandwidth network because of its wrong strategy in packet loss detection.
Kharat, et al[25]	ICCSP	2018	Web Browsing	Simulated Network	WiFi	QUIC occupies more bandwidth than TCP when QUIC and TCP are used simultaneously.
Yang, et al[26]	IWCMC	2018	Web Browsing	Simulated Network	Satellite Network	QUIC performs better than TCP in satellite network which is of high RTT.
Rajiullah, et al[9]	WWW	2019	Web Browsing	Internet	Cellular Network	A QUIC-compatible website may be delayed by referencing QUIC-incompatible websites.

2.1.1 网页浏览

网页浏览是互联网中使用最为广泛的应用.浏览器在进行网页访问时会先下载网站资源,包括 HTML,CSS,JavaScript 等类型的文件,再渲染在界面上,最终呈现给用户.一个好的网络协议应该保证浏览器花费最少的时间完成网站资源的下载,也就是最小化网页加载时间(page load time, PLT)[33].

Das[17]在他的硕士论文中最早进行了对 QUIC 网络传输性能的系统性分析.他在可控网络下模拟了不同的网络参数,包括带宽和端到端延时.他选择了 500 个网站,测量了使用 TCP 和 QUIC 在不同的网络环境下访问这些网站的网页加载时间.他根据实验结果总结并分析了 QUIC 的主要优势为:1)当带宽很低时,拥塞控制窗口会很小,导致丢包的发生更容易阻塞住后面的数据包,这时没有队头阻塞问题的 QUIC 更不容易受到影响.2)当端到端延时较高时,因为 QUIC 在握手过程比 TCP 消耗更少的 RTT,所以能够有效降低初始数据包的延时.然而,作者在实验过程中也发现了 QUIC 存在的问题.一方面,因为 QUIC 协议实现仍然处于迭代式开发中,所以代码没有很好地进行优化,在与 TCP 的对比中处于劣势.另一方面,QUIC 强制要求进行数据加密,而数据加密在 TCP 是可选项(即 TLS 协议),由加密解密带来的额外计算开销会降低 QUIC 的数据包处理速度.即便同时启用加密,因为 QUIC 的安全性更强,仍然导致其加密解密过程比 TCP 慢 12%.当网络带宽变大,尤其是大于 25 Mb/s 时,QUIC 的网络传输速度甚至会比 TCP/TLS 慢 80%.这是因为高吞吐率使得 CPU 因无法及时完成加密解密操作,最终成为网络传输的瓶颈.

Google 在 2017 年 SIGCOMM 的论文中总结了他们在大规模部署实验中使用 QUIC 的经验[5].从 2014 年开始,Google 在 Chrome 浏览器以及手机上的 YouTube 应用和 Google 搜索应用上对部分用户启用了 QUIC,并与 TCP 进行了对比.到 2017 年,几乎所有的 Chrome 和 YouTube 应用都已启用 QUIC.他们发现,在搜索服务中,有 88%来自桌面电脑的连接使用了 0-RTT 握手,在握手延时方面与原来的 TCP/TLS 相比至少减少了 2 个 RTT.另外,QUIC 使用了更多的信号用于网络拥塞及丢包的检测,这也使得 QUIC 对高丢包率有更好的容忍性.这些因素导致搜索服务的整体延时降低了 3%~8%.而 QUIC 在移动设备上的优势则没有桌面端的优势那么明显.这是因为移动端的 0-RTT 握手概率更低,只有 68%.0-RTT 握手概率降低的原因是,一方面,移动设备会频繁更换网络,使得服务端强制要求验证客户端新的 IP 地址并退回到 1-RTT 握手.另一方面,在切换网络后,客户端可能会选择与之前

不一样的数据中心获得服务,从而无法利用之前的连接信息,只能使用 1-RTT 握手与新的服务器建立新的连接.与 Das 一样,Google 也发现,当传输速率过快时(高于 100 Mb/s),CPU 会成为系统的瓶颈从而限制网络吞吐率.

Kakhki 等人[3]在可控环境下完成了更为详细的对比实验.他们使用软件模拟了不同的网络环境,包括带宽、延时以及丢包率,并在这些网络下对比了 QUIC 与 TCP 的网页加载延时.这些网页根据包含资源的数量和大小的不同被分为了很多组.实验表明,QUIC 几乎在所有场景下都获得了比 TCP 更短的网页加载时间.作者发现,0-RTT 握手是造成该场景下 QUIC 性能优越的主要原因.在所有情况下,只有当网页中存在大量的小资源时 QUIC 才表现出比 TCP 更差的性能.其原因是,QUIC 发送端会因为观测到传输过程中最小 RTT 的增大而错误地认为网络中发生了拥塞,从而过早地结束了拥塞窗口的快启动阶段,导致发送速率无法快速增长.这一点在传输数据流大时没有显著的影响,因为比较长的传输时间让 QUIC 有充足的时间增长传输速率.然而,当传输数据量较小时,往往在 QUIC 的传输速率增长到物理带宽限制之前,传输就已经完成了.此外,作者发现,QUIC 在数据包乱序严重的场景下,比如 3G 网络,性能会严重下降,甚至低于 TCP.其原因在于 QUIC 在丢包检测算法的参数选择上过于严苛,导致算法错误地将乱序到达的数据包判断为丢包,造成不必要的重传,最终浪费带宽资源.

Kharat 等人[25]在 WiFi 网络下对 QUIC 和 TCP 的公平性进行了分析.他们发现,在只有一条连接时,QUIC 能比 TCP 更有效地利用带宽.但是当同时使用 2 条 TCP 连接和 2 条 QUIC 连接时,TCP 能够抢占更多的带宽.原因是某些浏览器(例如 Opera 和 Firefox)会为 TCP 连接保留部分带宽用于小文件的传输,造成了对 QUIC 的不公平.

Yang 等人[26]分析了 QUIC 在卫星网络下的表现.相较于传统网络,用于空间通信的卫星网络拥有高延时、高丢包率的特征.他们基于仿真的卫星网络,比较了 QUIC 和 TCP 在地球同步轨道卫星和低地轨道卫星 2 种情况下的性能差异.比较结果显示,在所有的情况下,QUIC 都能表现出更好的性能.通过对传输过程更细化的分析,他们总结出 QUIC 的优势主要在于通过 0-RTT 握手减少了握手延时以及通过更大的拥塞窗口减少了接收端的丢包.张晗[27]更进一步地在基于 QUIC 的卫星网络中使用 BBR 算法[34]替代 QUIC 中默认的 NewReno 算法[35]用于拥塞控制,进一步提升了 QUIC 在高延时和高丢包率网络下的传输性能.

与以上在仿真网络下进行的测量工作不同,Rajiullah 等人[9]在真实的蜂窝网络下对支持 QUIC 的网站进行了测量.他们发现,使用 QUIC 访问这些网站往往比直接使用 TCP 会耗费更长的网页加载时间.其原因在于,支持 QUIC 的网站通常需要引用其他网站的资源,而这些资源所在的服务器通常不支持 QUIC,因此浏览器需要花费一定的延时从 QUIC 退回到 TCP,从而导致总的加载延时被延长.这一发现为后续从事互联网上 QUIC 测量工作的研究者提供了新的思路,即互联网对 QUIC 的支持情况也会对我们所观察到的 QUIC 的性能产生不小的影响.

对于以上测量工作,我们注意到,所有这些对 QUIC 的测量工作都是基于 gQUIC 完成的.主要原因是 Google 目前在 Chrome 上实现的是 gQUIC 而不是 IETF 版本的 QUIC.Chrome 作为一个主流浏览器,gQUIC 为它带来的性能改变更受研究人员关心.同时,Google 在全球部署的大量服务器也为 gQUIC 的测量提供了可靠的实验平台.但是这也导致了科研人员过分专注于 gQUIC 在互联网上的表现,而忽视了对 IETF 版本 QUIC 性能的分析.虽然这 2 个版本的 QUIC 有着相同的设计思想,但是在很多细节上仍有不同,这些差异可能会导致一些 gQUIC 上的测量结果在 IETF 版本 QUIC 上不再适用.此外,QUIC 的安全性更接近于 TLS1.3,但是在上述的对比实验中,大部分研究人员选择将 QUIC 与 TCP+TLS1.2 进行对比,这样的比较并不合理,因为 TLS1.3 的安全复杂程度远高于 TLS1.2,性能必然会有明显地下降.随着 TLS1.3 标准的正式完成,相信以后会有更多测量是基于 TLS1.3 进行的.

2.1.2 视频传输

视频传输是互联网中另一个重要的应用.Cisco 的报告表明,在 2014 年,视频流量占整个移动网络流量的 55%.并且这一数据有望在 2019 年达到 72%[36].在视频领域,研究人员普遍使用缓存等待率来衡量视频的服务质量,也就是用户等待视频缓存的时间占总播放时间的比例.

Google 在 YouTube 应用上的测量数据表明,与搜索服务的场景下类似,QUIC 能够在桌面端和移动端分别使得 85%和 65%的连接实现 0-RTT 握手[5].但与网页访问不同的是,YouTube 会在用户打开视频前提前与服务端建立连接,从而导致 QUIC

的 0-RTT 握手的优势不再明显.尽管如此,实验表明,QUIC 仍然能在很大程度上优化视频播放的质量,它在移动端和桌面端分别降低了 13.5%和 18.0%的缓存等待率.这是因为,相较于更低的握手延时,及时的丢包重传对视频传输来说更重要.一个视频或者音频帧的丢失往往会阻塞住后面的内容,导致播放器暂停并进行缓冲.YouTube 在视频传输时同时使用 2 条 TCP 连接,一条连接无法得知另一条连接的数据包是否丢失,从而导致每条连接对数据包的敏感性降低.而 QUIC 只使用了一条连接,可以最大限度地使用已有信息作为判断依据,增加对数据包的敏感性.此外,QUIC 支持更多的传输层信号,可以帮助发送方判断数据包是否已经丢失,从而能更精确地检测网络中的丢包行为,避免因为错误检测而造成的不必要的数据包重传.

然而 Kakhki 等人[3]却得出了与 Google 不完全相同的结论.他们在丢包率为 1%,带宽为 10 Mb/s,50 Mb/s,100 Mb/s 的网络下对比了使用 TCP 和 QUIC 播放 YouTube 视频的效果.实验发现,在低帧率时,QUIC 和 TCP 的性能基本一致.只有在高帧率的场景下,QUIC 才能提高视频传输的性能,减小缓存等待时间.

2.2 QUIC 的性能优化

目前对 QUIC 的优化工作主要有 2 方面,分别是对协议本身的优化以及对其实现方法的优化.在协议方面,De Coninck 等人[8]以及 Viernickel 等人[28]分别将多路径的概念引入 QUIC,使得一条 QUIC 连接可以同时使用多条底层网络链路,增加带宽.在系统实现方面,Wang 等人[29]在系统内核态实现了 QUIC 协议,而 Duan 等人[30]则使用了用户态的 UDP 实现,他们均在一定程度上提高了 QUIC 的处理数据包的速度.

2.2.1 多路径 QUIC

多路径是指允许一条传输层连接同时使用多条物理网络链路的技术.其优势在于,可以同时使用多个物理网络从而增加有效网络带宽.例如,手机用户可以同时使用蜂窝网络和 WiFi,电脑用户可以同时使用以太网和 WiFi.多路径在 TCP 上已有大量研究[37-38],其中 MPTCP[39]已经作为网络协议标准被 IETF 收录,并且被 Linux 内核所支持.

De Coninck 等人[8]最早提出了多路径 QUIC 的设计:MPQUIC.他们在流之下定义了路径(path),用于描述数据传输所使用的物理网络路径.在每个 Stream 帧中会包含一个 Path ID,根据不同的 Path ID,QUIC 可以在发送和接收时将数据包对应到不同的物理网络.在握手阶段,客户端与服务端会先在一条网络链路上建立 QUIC 连接.之后每有一个网络链路需要使用,便添加一个新的 Path.由此,客户端可以以多个 IP 地址为源地址,向服务端发送数据包.在发送端,拥塞控制是对每个物理网络连接分别进行的.也就是说,每个 Path 都会维护自己的拥塞窗口、ACK 计数、RTT 估计等信息,从而实现对每条物理网络分别进行流量控制的目的.在接收端,来自不同 Path 的数据包会被合并从而得到完整的 Stream 帧,并进行下一步处理.在多路径的拥塞控制算法上,MPQUIC 使用了在 MPTCP 上表现良好的 OLIA 算法[40].De Coninck 等人[8]的实验表明,在下载大文件的场景下,大部分时候,MPQUIC 能够比 MPTCP 更快地完成传输.这是因为 MPTCP 缺乏一个有效的对物理连接延时的判断机制,从而会使得某些数据包被阻塞在较慢的连接上,影响了整体的速度.而 MPQUIC 则不存在这些问题,因为它对每个物理连接都进行了精确的控制.

Viernickel 等人[28]提出了另一套多路径 QUIC 的设计方案.与 MPQUIC 不同的是,他们通过不同的 UDP 套接字直接区分不同的物理网络路径,从而避免了引入 Path ID 的需要.但是,添加一个新的物理网络路径需要连接的一方主动发送 announcement 数据包通知对方新的物理网络连接即将被加到当前 QUIC 连接中,这样导致添加新物理网络连接需要耗费 1 个 RTT 的延时.

为了对比多路径 QUIC 与 MPTCP 在实际互联网下性能的差异,De Coninck 等人设计了一个 iOS 应用:MultipathTester[41].该应用支持在不同网络环境下对多路径 QUIC 和 MPTCP 进行基准测试.实验结果表明,多路径 QUIC 与 MPTCP 在数据传输性能方面没有显著的差异[42].唯一的区别在于,多路径 QUIC 会在客户端主动检测每条物理链路的可用性并向服务端告知其物理网络的变化.例如,当手机的 WiFi 连接断开时,客户端会通过移动网络告知服务端放弃该网络连接,从而避免服务端继续使用其 WiFi 连接而造成不必要的丢包与重传.而 iOS 内核中实现的 MPTCP 并不具备这样的功能.由此可见,尽管多路径 QUIC 在协议设计上已经取得了一定的进展,但是正如 MPTCP 所面临的挑战一样[34-44],如何设计出性能更好的拥塞控制算法并将其应用在 QUIC 上才是让多路径 QUIC 普及的关键因素.

2.2.2 用户态与内核态

在操作系统中,出于安全性的考虑,地址空间被分为用户空间和内核空间.一般来说,用户程序,例如应用程序,运行在用户空间;而系统程序,例如操作系统,运行在内核空间.用户程序可以使用系统函数调用的方式进入内核空间.传统意义上,传输层协议(例如 TCP 和 UDP)均在 Linux 内核中实现,并且运行在内核空间.但是近年来的研究发现基于内核旁路[45]的用户空间 TCP 可以达到比内核空间 TCP 更高的性能[46].因为应用程序工作于用户空间,所以用户空间 TCP 可以避免应用程序进行系统调用时的空间切换开销,同时也能够避免数据在内核空间和用户空间之间的复制,最终达到加速应用程序运行速度的目的.另外,实现于用户空间的网络协议不受系统内核更新周期的限制,可以实现更快速的更新.

目前使用的最为广泛的 QUIC 实现是在 Chrome 中的 gQUIC,它工作于用户空间并且仍处于迭代开发中.这样的设计虽然可以允许协议的快速修改但是却不能提供最好的性能,因为它的下层网络使用了系统内核的 UDP 套接字,这样既不能利用内核态程序的高优先级又不能像用户空间 TCP 一样避免空间切换的开销.

Wang 等人[29]在内核空间实现了 gQUIC,用于在公平的环境下进行 QUIC 和 TCP 的性能对比.Duan 等人[30]在用户空间实现了包括 UDP 在内的 gQUIC 协议栈.他们测试了客户端和服务端在短时间内进行快速握手的性能,发现使用内核旁路的 QUIC 能够比 Chrome 多处理 100 倍以上的握手.

此外,也有研究者致力于加速 QUIC 数据包的处理.因为 QUIC 性能最大的瓶颈在于加密与解密算法的开销,YouTube 服务器在使用 QUIC 时的 CPU 开销是 TCP+TLS 的 2 倍[5].所以对该过程的优化也是未来的一个研究方向.一种可能的方式是 PixelVault[47]一样,将加密解密相关的操作移到 GPU 上进行执行,从而降低 CPU 负载并加速数据包的处理.

2.3 QUIC 的安全性

gQUIC 的安全性与其传输层协议相绑定,这导致早期对 QUIC 安全性的研究主要针对 QUIC 整体进行.随着 IETF 版本 QUIC 的推出,安全性相关的设计被从 QUIC 中剥离出来,并形成了 TLS1.3.于是后续关于 QUIC 安全性的研究更倾向于对 TLS1.3 的安全性分析.在本节,我们首先列举研究者对 QUIC 和 TLS1.3 安全性的建模分析,然后对已发现的安全漏洞进行介绍与总结.表 2 对 QUIC 安全性的相关工作进行了总结.

Table 2 Related Work on QUIC Security

表 2 QUIC 安全性的相关工作

Reference	Publication Place	Publication Time	Content
Fischlin, et al[2]	CCS	2014	Propose a multi-stage key exchange model to describe QUIC's handshake protocol and propose QUICi, which is a variant of QUIC but with higher security level.
Lychev, et al[31]	S&P	2015	Propose the concept of quick communication protocols and QACCE model to prove the security of QUIC's handshake protocol. Enumerate QUIC's vulnerable attacks.
Jager, et al[4]	CCS	2015	Prove that QUIC (TLS1.3) is not vulnerable to Bleichenbacher attach.
Günther, et al[48]	PLDI	2016	Propose a 0-RTT handshake protocol that supports forward security.

2.3.1 安全模型

Fischlin 等人[2]提出了多阶段公钥交换(multi-stage key exchange)模型,用于描述 QUIC 的握手机制.利用这个模型,他们证明了 QUIC 握手机制的安全性,然而该模型无法很好地涵盖握手之后的数据传输过程.也就是说,即便通信双方使用了安全的包含身份验证机制的加密协议用于数据加密,QUIC 协议作为整体的安全性也无法得到保证.为了解决这个问题,他们提出了一种可以符合其安全模型的 QUIC 设计:QUICi.QUICi 采用了更为复杂的密钥生成机制,从而使得用于不同目的的密钥之间相关性降低,增加了 QUIC 的安全性.

与 Fischlin 等人不同,Lychev 等人[31]对未经修改的 QUIC 的安全性进行了更进一步的分析.针对 QUIC 和 TLS1.3 的 0-RTT 握手机制,他们提出了快速通信协议(quick communication protocols)的概念,用于描述在最终会话密钥(final session key)生成之前先使用初始会话密钥(initial session key)

的做法,QUIC 和 TLS1.3 均属于快速通信协议.同时,他们提出了 QACCE(quick authenticated and confidential channel establishment)模型,并使用该模型证明了 QUIC 连接建立过程和数据加密传输过程的安全性.

同时,Lychev 等人[31]指出,基于最终会话密钥的 1-RTT 握手和基于初始会话密钥的 0-RTT 握手拥有不同的安全级别.前者可以确保通信双方在攻击者存在的情况下能够生成一致的共享密钥进行数据加密,并且提供前向安全性的保障.而后者可能因为攻击者的存在而使通信双方产生不一致的共享密钥,导致数据的加密解密过程失败.并且,0-RTT 握手也不能保证前向安全性.关于 0-RTT 的安全问题我们会在 2.3.2 节进行更为详细的介绍.

在此基础上,Jager 等人[4]从攻击者的角度分别对 QUIC 和 TLS1.3 的安全性给出了分析.在 TLS1.2 中,Bleichenbacher 攻击及其变形能够快速猜测出服务端的密钥,从而破坏其安全性[49-50].在 QUIC 和 TLS1.3 中,因为移除了 TLS1.2 中导致安全性漏洞的 PKCS#1 v1.5 标准的支持,从而理论上不再受该类攻击威胁.Jager 等人的模拟攻击结果表明,TLS1.3 和 QUIC 极大地增加了攻击者在进行攻击过程中所消耗的时间,从而使得该种攻击不再具有可行性.

2.3.2 前向安全性与 0-RTT 握手的安全问题

前向安全性是通信协议的一种安全属性,它指的是长期使用的主密钥泄露不会导致过去的会话密钥泄露[32].其意义在于,前向安全性能够保护过去进行的通讯不受密码或密钥在未来暴露的威胁.TLS1.2 与 QUIC/TLS1.3 的 1-RTT 握手均很好地支持了前向安全性.但是 QUIC/TLS1.3 的 0-RTT 握手却无法为通信双方提供前向安全性[31].这是因为,0-RTT 通信过程中使用的是初始会话密钥,而这个密钥是根据服务端的静态配置生成的,未来如果该配置泄露,亦会导致 0-RTT 的密钥也随之泄露.

为了解决这个问题,Günther 等人[48]通过在服务端的修改使得 QUIC 的 0-RTT 握手支持前向安全性.他们使用了一种特殊的密钥设计:每当服务端用当前密钥解密一个密文后,就将当前密钥进行修改,新的密钥可以像原来的密钥一样进行密文解析,唯一不同的是不能对已经解析过的密文再次进行解密,从而保证了前向安全性.

同时,该设计也会使得 QUIC 不再遭受重放攻击的威胁.但是它带来了性能上的挑战,因为服务端的密钥的复杂度会随着时间的推移逐渐增加.为此,他们又提供了新的协议机制使得服务端可以每过一段时间将当前密钥重置一次,从而降低密钥的复杂度.

2.3.3 易遭受的攻击

虽然许多研究工作证明了 QUIC 可以保证其传输的数据无法被恶意攻击者所获取,但是攻击者仍然有能力对通信双方的正常通信进行干扰,甚至影响客户端或者服务端的正常运行.Lychev 等人[31]列举了 QUIC 易遭受的几种安全攻击.根据攻击者是否处于客户端和服务端之间的网络路径上,这些攻击被分为了离线攻击和在线攻击.

1) Server Config 重复攻击,离线.类似于 TCP 的 reset 攻击[51],攻击者可以通过嗅探获得服务端的基本配置信息,然后利用这些配置信息并伪装成服务端的 IP 地址向客户端发送 reset 数据包,终止 QUIC 连接.

2) Crypto Stream Offset 攻击,离线.因为 QUIC 的握手数据包的大小也会被统计入 Stream 的偏移量(offset),攻击者可以在握手数据包后面附加一些字符串,导致接受方获得错误的 Stream 偏移量数值,从而无法正确解析之后的数据包.

3) 连接 ID 篡改攻击,在线.通过篡改握手过程中 Client 使用的连接 ID,攻击者可以使得通信双方在很长的一段时间内(10 s)都认为连接已经成功,却无法正常解析接收到的数据,并导致连接中断.

4) Source-Address Token 篡改攻击,在线.与连接 ID 篡改攻击相似,攻击者可以通过篡改 Source-Address Token 的方式导致通信双方无法正常解析对方收到的数据包.但是双方在很长的一段时间内(10 s)都认为连接已经成功,直到主动中断连接.

可以看出,QUIC 在特定情况下无法提供有效的手段阻止在线和离线的攻击者中断一条 QUIC 连接.对于在线的攻击者,它能够让通信双方的连接处于闲置状态一段时间,并且该行为无法立刻被通信双方检测到.而对于离线的攻击者,它只需要获取极少的连接相关信息便可以让一条 QUIC 连接被迫中断.

此外,McMillan 等人[6]使用 Ivy[52]对 QUIC 协议进行了详细的定义,并通过自动生成的随机攻击者对 QUIC 各版本的安全漏洞进行探测.经过实验,他们一共发现了 QUIC 运行过程中可能产生的 27 个错误.出于安全性的考虑,McMillan 等人只公布了 QUIC 历史版本中所发现的并且已经被解决的问题,并没有对现存问题进行详细介绍.

3 未来研究方向

尽管研究人员已经对 QUIC 在网络传输性能测量、系统性能优化以及安全性分析等方面进行了大量工作.但是,目前已有的科研工作仍有进一步提高的地方:

1) 缺乏对 IETF 版本 QUIC 的分析.gQUIC 只是 QUIC 的早期设计,属于过渡期的协议.IETF 作为一个负责开发和推广互联网标准的组织,它制定的 QUIC 协议才是未来 QUIC 的标准版本.尽管有许多开源组织致力于跟进 IETF QUIC 的每一个草稿并提供实现.但是目前 IETF 版本 QUIC 的许多设计细节并没有确定,而且也没有一个被广泛使用的代码实现.反之,gQUIC 借助于 Chrome 的影响力已经被数十亿人使用.因此,大部分的测量工作都是基于 gQUIC 进行的,学术界仍然缺少对 IETF 版本 QUIC 的性能的有效认识.

2) 软件实现对分析结果的影响过大.尽管目前对 QUIC 的性能分析工作很多,但是大多局限于特定的实现,导致学术界对 QUIC 性能没有一个统一的认识.在分析方法上,研究人员也普遍根据测量结果猜测造成差异的原因,没有进行严格的控制变量法的对比.如果可以通过禁用 QUIC 某些特性的方法进行对比,就能更加深入地了解 QUIC 在不同网络环境下表现好坏的根本原因.

3) CPU 成为 QUIC 的性能瓶颈,导致无法充分利用带宽.与现有的安全性协议相比,QUIC 借用 TLS1.3 达到了更高的安全性.但是随之而来的是加密解密复杂度的提高以及 CPU 负载的增加.这一点在手机等 CPU 性能相对薄弱的设备上尤为明显.如何让 QUIC 在现有的 CPU 性能下达到更高的带宽是一个亟待解决的问题.

基于现有对 QUIC 的研究工作,我们分别从网络传输性能、系统性能以及安全性 3 个方面预测了在未来可能有意义的 QUIC 的研究方向.

1) 现有的网络测量结果表明,QUIC 在大部分情况下表现出了比 TCP+TLS 更低的延时.同时,测量结果也显示 QUIC 为了降低延时而进行的设计达到了预期效果.我们发现很多 QUIC 表现不如 TCP 的场景是由其不合适甚至错误的算法选择造成的,而并非其协议本身的缺陷.造成这一问题的原因正是 QUIC 在算法选择上的灵活性.因此,如果研究人员能够归纳并总结 QUIC 在拥塞控制、丢包检测等方面的可行算法,并对不同算法的适用环境进行分析,定能帮助 QUIC 更好地适应复杂的互联网环境.

2) 虽然研究人员已经为提高 QUIC 的系统性能和降低数据包处理的延时提出了很多设计,但是目前并没有一个真正有效且通用的策略.无论是内核旁路还是 GPU 都对硬件及其驱动有额外的要求,并且难以在移动设备上实现.一种可行的提高 QUIC 系统性能的思路是将 QUIC 的实现进行并行化.因为 QUIC 本身就是一个多路传输协议,如果能够有效利用现代处理器的多个核心,必然能为 QUIC 的处理速度带来进一步提升.但如何对 QUIC 协议进行解耦以使得它能够将原本单线程的工作分配到不同的 CPU 核心上还有待研究人员解决.

3) QUIC 的 0-RTT 握手无法保证前向安全性,而 Günther 等人[48]所提出的前向安全的 0-RTT 握手却对系统性能有更大的消耗.另一方面,TLS1.3 带来的加密解密负荷使得 QUIC 对 CPU 的运算需求已经大大提高.所以,如何在安全性与计算开销之间进行权衡是一个值得研究与讨论的话题.另外,现有的研究发现 QUIC 连接容易被在线或者离线的攻击者所打断.研究人员应当为 QUIC 连接增加更多的保障机制从而帮助它更好地鉴别攻击者的恶意行为,增加连接的安全性.

除了以上的研究方向外,我们注意到,已经有科研工作者将 QUIC 应用于 HTTP 以外的其他应用层协议,并且为系统性能带来了不错的提升[53].未来,如果能够将 QUIC 应用于更多的应用层协议,例如 WebRTC,FTP,DNS 等,很有可能会让它们在网络传输性能、系统性能以及安全性方面获得提升.

在 5G 领域,QUIC 也有着不错的应用前景.3GPP 在 5G 标准中规定了其核心网将使用虚拟化技术实现其各组件的功能.不同组件之间将使用基于 HTTP/2 的 RESTful 接口进行通信[54].但是鉴于 QUIC 在传输性能方面的提升,3GPP 正在考虑使用 QUIC 替代 HTTP/2 用于核心网各组件之间的数据通信[55].除此之外,因为 5G 相较于上一代无线技术(4G)提供了更低的无线通信延时[56],所以它使得通过蜂窝网络为用户提供极低延时的网络服务成为了可能.而 QUIC 在握手延时方面的优势,尤其是 0-RTT 握手,将有助于网络服务更进一步地降低网络延时,提升用户体验.在我们的前期工作中,我们提出了一个基于 QUIC 的域名解析系统:Artemis[57].Artemis 借助于 QUIC 在移动性上的优势,实现了

比传统 DNS 更低的域名解析延时.我们相信,未来将会有更多的基于 5G 以及 QUIC 实现的低延时服务出现在云计算、边缘计算以及物联网等领域.

4 结 论

QUIC 作为一个新的传输层协议,它在设计上针对 TCP 的不足进行了很多优化.它提供的多路传输、快速握手等新特性使得它和 TCP 相比在理论上可以获得更低的数据传输延时.现有测量工作表明,QUIC 在大部分情况下的确能比 TCP 达到更低的传输延时,但是仍然有部分情况下 QUIC 的表现不如 TCP.这些 QUIC 性能表现较差的场景往往是拥塞算法的选择、服务器部署等外部因素造成的,而非 QUIC 本身的设计缺陷.因此,QUIC 的软件实现仍然有很大的进步空间.在安全性方面,基于 TLS1.3 的 QUIC 一方面牺牲了一定的计算资源为用户提供了比 TLS1.2 更高的安全性,另一方面以牺牲前向安全性为代价换来了更低的握手延时.但总体来说,TLS1.3 还是很好地保障了数据传输的安全性.此外,许多在 TCP 上未被解决的问题在 QUIC 上也依然存在,仍然有很多后续工作等待着科研工作者们继续探索.

在 QUIC 的应用上,随着 5G 等新兴应用场景的出现以及 HTTP/3 标准的制定,相信将来会有越来越多的基于 QUIC 的网络服务的出现.如何充分利用 QUIC 的特性提升网络服务的服务质量有待工业界和学术界的共同探索.

参 考 文 献

[1] Iyengar J, Thomson M. QUIC: A UDP-Based Multiplexed and Secure Transport [S/OL]. 2019 [2019-09-21]. https://tools.ietf.org/html/draft-ietf-quic-transport-23

[2] Fischlin M, Günther F. Multi-stage key exchange and the case of Google's QUIC protocol [C] //Proc of the ACM Conf on Computer and Communications Security. New York: ACM, 2014: 1193-1204

[3] Kakhki A M, Jero S, Choffnes D, et al. Taking a long look at QUIC [C] //Proc of the ACM Internet Measurement Conf. New York: ACM, 2017: 290-303

[4] Jager T, Schwenk J, Somorovsky J. On the security of TLS 1.3 and QUIC against weaknesses in PKCS#1 v1.5 encryption [C] //Proc of the ACM Conf on Computer and Communications Security. New York: ACM, 2015: 1185-1196

[5] Langley A, Riddoch A, Wilk A, et al. The QUIC transport protocol: Design and Internet-scale deployment [C] //Proc of the ACM Special Interest Group on Data Communication. New York: ACM, 2017: 183-196

[6] McMillan K L, Zuck L D. Formal specification and the testing of QUIC [C] //Proc of the ACM Special Interest Group on Data Communication. New York: ACM, 2019: 227-240

[7] De Coninck Q, Michel F, Piraus M, et al. Pluginizing QUIC [C] //Proc of the ACM Special Interest Group on Data Communication. New York: ACM, 2019: 59-74

[8] De Coninck Q, Bonaventure O. Multipath QUIC: Design and evaluation [C] //Proc of the Conf on Emerging Network Experiment and Technology. New York: ACM, 2017: 160-166

[9] Rajiullah M, Lutu A, Khatouni A S, et al. Web experience in mobile networks: Lessons from two million page visits [C] //Proc of the World Wide Web Conf. New York: ACM, 2019: 1532-1543

[10] Google. A QUIC update on Google's experimental transport [EB/OL]. (2015-04-17) [2019-09-21]. https://blog.chromium.org/2015/04/a-quic-update-on-googles-experimental.html

[11] Swett L, Behr M. Introducing QUIC support for HTTPS load balancing [EB/OL]. (2018-06-13) [2019-09-21]. https://cloud.google.com/blog/products/gcp/introducing-quic-support-https-load-balancing

[12] Yakan M, Jayaprakash A. Introducing QUIC for Web content [EB/OL]. (2018-10-10) [2019-09-21]. https://developer.akamai.com/blog/2018/10/10/introducing-quic-web-content

[13] Jones N. Get a head start with QUIC [EB/OL]. (2018-09-25) [2019-09-21]. https://blog.cloudflare.com/head-start-with-quic/

[14] Jan R. How much of the Internet is using QUIC [EB/OL]. (2018-03-15) [2019-09-21]. https://blog.apnic.net/2018/05/15/how-much-of-the-internet-is-using-quic/

[15] Dierks T, Rescorla E. The Transport Layer Security (TLS) Protocol Version 1.2 [S/OL]. RFC 5246, 2008 [2019-09-21]. https://tools.ietf.org/html/rfc5246

[16] Rescorla E. The Transport Layer Security (TLS) Protocol Version 1.3 [S/OL]. RFC 8446, 2018 [2019-09-21]. https://tools.ietf.org/html/rfc8446

[17] Das S R. Evaluation of QUIC on Web page performance [D]. Cambridge, MA: Massachusetts Institute of Technology, 2014

[18] Roskind J. Experimenting with QUIC [EB/OL]. (2013-06-27) [2019-09-21]. https://blog.chromium.org/2013/06/experimenting-with-quic.html

[19] Roskind J. Quick UDP Internet connections [EB/OL]. (2013-11-07) [2019-09-21]. https://www.ietf.org/proceedings/88/slides/slides-88-tsvarea-10.pdf

[20] Bishop M. Hypertext Transfer Protocol Version 3 (HTTP/3) [S/OL]. 2019 [2019-09-21]. https://tools.ietf.org/html/draft-ietf-quic-http-23

[21] Belshe M, Peon R, Thomson M. Hypertext Transfer Protocol Version 2 (HTTP/2)[S/OL]. RFC 7540, 2015 [2019-09-21]. https://tools.ietf.org/html/rfc7540

[22] Scharf M, Kiesel S. Head-of-line blocking in TCP and SCTP: Analysis and measurements [C] //Proc of the IEEE Globe Communications Conf. Piscataway, NJ: IEEE, 2006: 1-5

[23] Qian Feng, Gopalakrishnan V, Halepovic E, et al. TM3: Flexible transport-layer multi-pipe multiplexing middlebox without head-of-line blocking [C] //Proc of the Conf on Emerging Network Experiment and Technology. New York: ACM, 2015: 1-13

[24] Mi Xianghang, Qian Feng, Wang Xiaofeng. SMig: Stream migration extension for HTTP/2 [C] //Proc of the Conf on Emerging Network Experiment and Technology. New York: ACM, 2016: 121-128

[25] Kharat P K, Rege A, Goel A, et al. QUIC protocol performance in wireless networks [C] //Proc of the Int Conf on Communication and Signal Processing. Piscataway, NJ: IEEE, 2018: 472-476

[26] Yang Siyu, Li Hewu, Wu Qian. Performance analysis of QUIC protocol in integrated satellites and terrestrial networks [C] //Proc of the Int Conf on Wireless Communications and Mobile Computing. Piscataway, NJ: IEEE, 2018: 1425-1430

[27] Zhang Han. Research on the performance of hypertext transfer protocol in satellite networks [D]. Nanjing: Nanjing University, 2018 (in Chinese)
(张晗. 卫星网络中超文本传输协议的性能研究[D]. 南京: 南京大学, 2018)

[28] Viernickel T, Froemmgen A, Rizk A, et al. Multipath QUIC: A deployable multipath transport protocol [C] //Proc of the Int Conf on Communications. Piscataway, NJ: IEEE, 2018: 1-7

[29] Wang Peng, Bianco C, Riihijärvi J, et al. Implementation and performance evaluation of the QUIC protocol in Linux kernel [C] //Proc of the ACM Int Conf on Modeling, Analysis and Simulation of Wireless and Mobile Systems. New York: ACM, 2018: 227-234

[30] Duan Yufeng, Gallo M, Traverso S, et al. Towards a scalable modular QUIC server [C] //Proc of the Workshop on Kernel-Bypass Networks. New York: ACM, 2017: 19-24

[31] Lychev R, Jero S, Boldyreva A, et al. How secure and quick is QUIC?Provable security and performance analyses [C] //Proc of the IEEE Symp on Security and Privacy. Piscataway, NJ: IEEE, 2015: 214-231

[32] Bellare M, Yee B. Forward-security in private-key cryptography [C] //Proc of the Cryptographers' Track at the RSA Conf. Berlin: Springer, 2003: 1-18

[33] Basques K, Garbee J. Network issues guide [EB/OL]. (2019-02-12) [2019-09-21]. https://developers.google.com/web/tools/chrome-devtools/network-performance/issues

[34] Hock M, Bless R, Zitterbart M. Experimental evaluation of BBR congestion control [C] //Proc of the Int Conf on Network Protocols. Piscataway, NJ: IEEE, 2017: 1-10

[35] Henderson T, Floyd S, Gurtov A. The NewReno Modification to TCP's Fast Recovery Algorithm [S/OL]. RFC 6582, 2012 [2019-09-21]. https://tools.ietf.org/html/rfc6582

[36] Cisco. Cisco visual networking index: Global mobile data traffic forecast update, 2017-2022 white paper [EB/OL]. (2019-02-08) [2019-09-21]. https:// www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/white-paper-c11-738429.html

[37] Jiang Zhuo, Wu Qian, Li Hewu, et al. Survey on Internet end-to-end multipath transfer research with cross-layer optimization [J]. Journal of Software, 2019, 30 (2): 302-322 (in Chinese)
(江卓, 吴茜, 李贺武, 等. 互联网端到端多路径传输跨层优化研究综述[J]. 软件学报, 2019, 30(2): 302-322)

[38] Xue Kaiping, Chen Ke, Ni Dan, et al. Survey of MPTCP-based multipath transmission optimization [J]. Journal of Computer Research and Development, 2016, 53 (11): 2512-2529 (in Chinese)
(薛开平, 陈珂, 倪丹, 等. 基于 MPTCP 的多路径传输优化技术综述[J]. 计算机研究与发展, 2016, 53(11): 2512-2529)

[39] Raiciu C, Paasch C, Barre S, et al. How hard can it be? Designing and implementing a deployable multipath TCP [C] //Proc of the Symp on Networked Systems Design and Implementation. Berkeley, CA: USENIX Association, 2012: 399-412

[40] Khalili R, Gast N, Popovic M, et al. MPTCP is not pareto-optimal: Performance issues and a possible solution [J]. IEEE/ACM Transactions on Networking, 2013, 21 (5): 1651-1665

[41] Coninck D Q, Bonaventure O. Observing network handovers with multipath TCP [C] //Proc of the ACM SIGCOMM Conf on Posters and Demos, New York: ACM, 2018: 54-56

[42] Coninck D Q. The stream traffic and the (unexpected) impact of happy eyeballs [EB/OL]. (2018-05-23) [2019-09-21]. https://multipath-quic.org/multipathtester/2018/05/23/stream-traffic-eyeballs.html

[43] Raicui C, Handley M, Wischik D. Coupled Congestion Control for Multipath Transport Protocols [S/OL]. RFC 6356, 2011 [2019-09-21]. https://tools.ietf.org/html/rfc6356

[44] Damon W, Raicui C, Greenhalgh A, et al. Design, implementation and evaluation of congestion control for multipath TCP [C] //Proc of the USENIX Conf on Networked Systems Design and Implementation. Berkeley, CA: USENIX Association, 2011: 99-112

[45] Rizzo L. Netmap: A novel framework for fast packet I/O [C] //Proc of the USENIX Conf on Annual Technical Conf. Berkeley, CA: USENIX Association, 2012: 101-112

[46] Jeong E, Wood S, Jamshed M, et al. mTCP: A highly scalable user-level TCP stack for multicore systems [C] // Proc of the USENIX Conf on Networked Systems Design and Implementation. Berkeley, CA: USENIX Association, 2014: 489-502

[47] Vasiliadis G, Athanasopoulos E, Polychronakis M, et al. PixelVault: Using GPUs for securing cryptographic operations [C] //Proc of the ACM Conf on Computer and Communications Security. New York: ACM, 2014: 1131-1142

[48] Günther F, Hale B, Jager T, et al. 0-RTT key exchange with full forward secrecy [C] //Proc of the Annual Int Conf on the Theory and Applications of Cryptographic Techniques. Berlin: Springer, 2017: 519-548

[49] Aviram N, Schinzel S, Somorovsky J, et al. DROWN: Breaking TLS using SSLv2 [C] //Proc of the USENIX Security Symp. Berkeley, CA: USENIX Association, 2016: 689-706

[50] Wei Junlin, Duan Haixin, Wan Tao. A survey of security deficiencies in design and implementation of HTTPS/TLS [J]. Journal of Cyber Security, 2018, 3(2): 1-15 (in Chinese)
(韦俊琳, 段海新, 万涛. HTTPS/TLS 协议设计和实现中的安全缺陷综述[J]. 信息安全学报, 2018, 3(2): 1-15)

[51] Lü Yanli, Li Xiaojian, Xia Chunhe, et al. Research on the security of initial sequence number generation arithmetic [J]. Journal of Computer Research and Development, 2005, 42(11): 1940-1945 (in Chinese)
(吕艳丽, 李肖坚, 夏春和, 等. 初始序列号生成算法的安全性研究[J]. 计算机研究与发展, 2005, 42(11): 1940-1945)

[52] Padon O, McMillan K, Panda A, et al. Ivy: Safety verification by interactive generalization [C] //Proc of the ACM Conf on Programming Language Design and Implementation. New York: ACM, 2016: 83-92

[53] Kumar P, Dezfouli B. Implementation and analysis of QUIC for MQTT [J]. Computer Networks, 2019, 150: 28-45

[54] Mayer G. 3GPP 5G CoreNetwork status [EB/OL]. 2017 [2019-09-21]. https:// www. 3gpp.org/ftp/Information/presentations/Presentations_2017/webinar-ct-status-11-2017.pdf

[55] Kymalainen K. Study on IETF QUIC transport for 5GC service based interfaces [EB/OL]. (2019-05-04) [2019-09-21]. https://portal. 3gpp. org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3494

[56] Pedersen I K, Frederiksen F, Berardinelli G, et al. The coverage-latency-capacity dilemma for TDD wide area operation and related 5G solutions [C] //Proc of the Vehicular Technology Conf. Piscataway, NJ: IEEE, 2016: 1-5

[57] Li Xuebing, Liu Bingyang, Chen Yang, et al. Artemis: A practical low-latency naming and routing system [C] //Proc of the Int Conf on Parallel Processing. New York: ACM, 2019: 1-10

Li Xuebing, born in 1993. Received his MSc and BSc degrees in computer science from Fudan University. His main research interests include network measurements, network protocols, and Internet architectures.

Chen Yang, born in 1981. Associate professor in the School of Computer Science at Fudan University. Received his BSc and PhD degrees from the Department of Electronic Engineering, Tsinghua University in 2004 and 2009, respectively. Senior member of IEEE and member of CCF. His main research interests include computer networks, social computing and big data analytics.

Zhou Mengying, born in 1997. Received her BSc degree in information security from Lanzhou University in 2019. Currently PhD candidate at Fudan University. Her main research interests include network measurement, social networks, and data mining.

Wang Xin, born in 1973. Received his BSc degree in information theory and MSc degree in communication and electronic systems from Xidian University, China in 1994 and 1997, respectively. He received his PhD degree in computer science from Shizuoka University, Japan, in 2002. Currently professor in Fudan University, Shanghai, China. Distinguished member of CCF and member of IEEE. His main research interests include quality of network service, next-generation network architecture, mobile Internet and network coding.