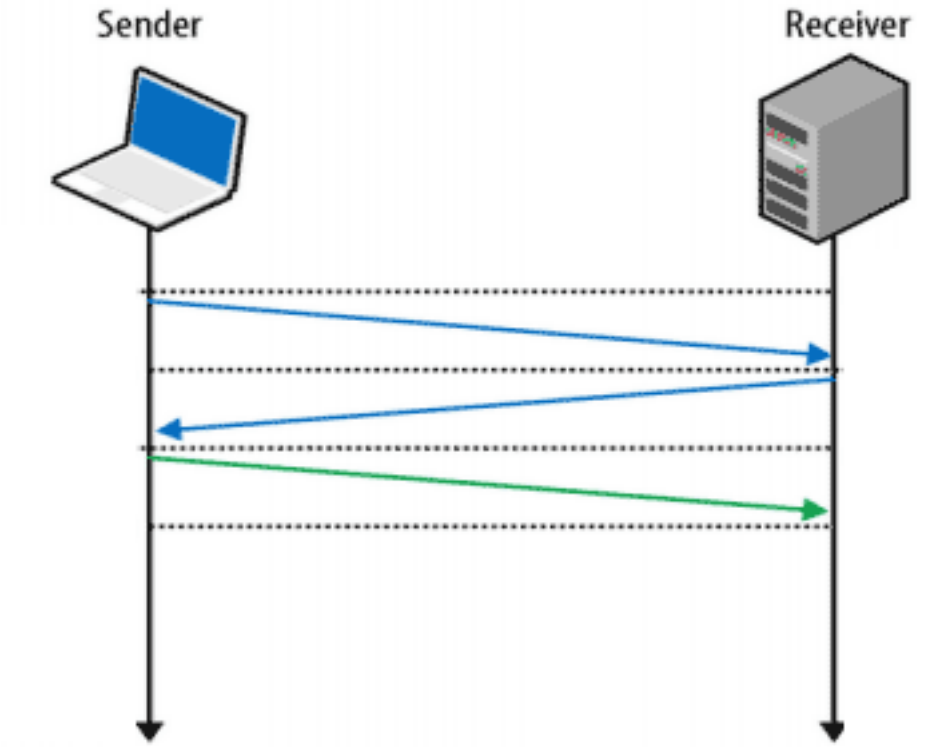
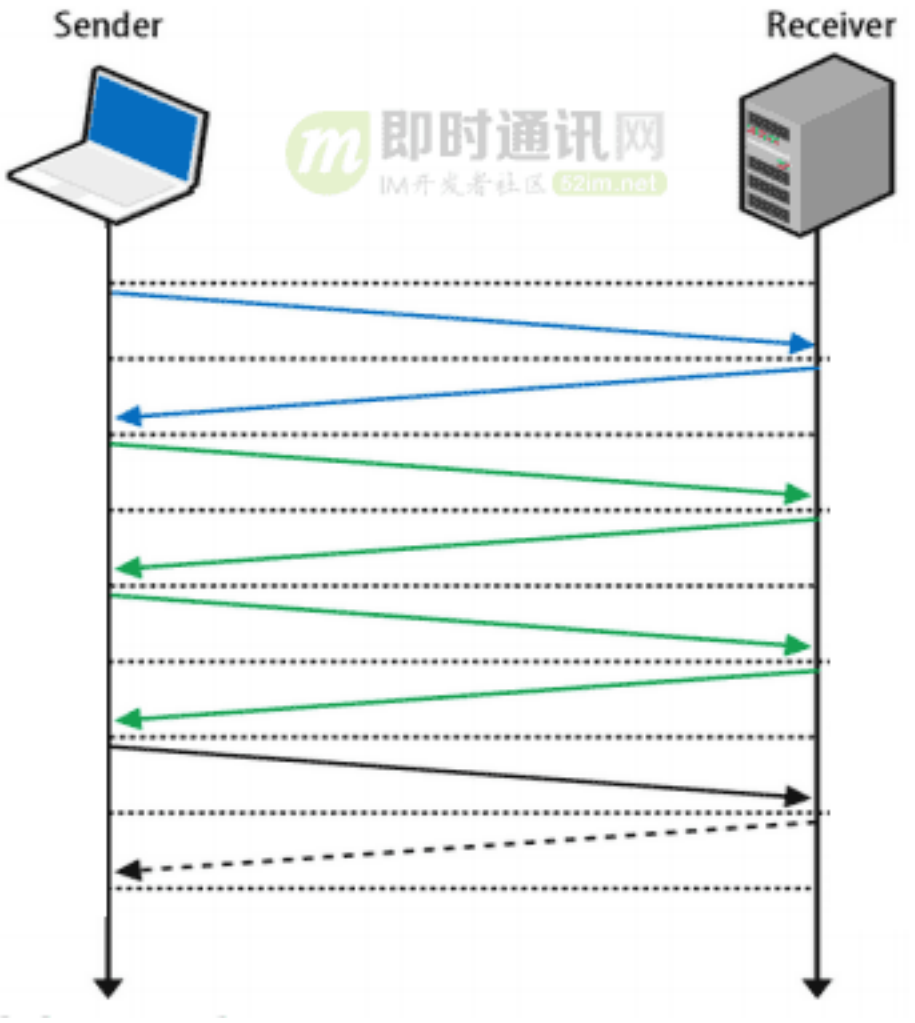
1、TCP协议到底怎么了？

现时的互联网应用中，Web平台（准确地说是基于HTTP及其延伸协议的客户端/服务器应用）的数据传输都基于TCP 协议。

但TCP 协议在创建连接之前需要进行三次握手（如下图1，更详细原理请见《理论经典：TCP协议的3次握手与4次挥手过程详解》），如果需要提高数据交互的安全性，既增加传输层安全协议（TLS），还会增加更多的更多握手次数（如下图 2）。





▲图 1 - TCP的三次握手原理图 ▲

图 2 - TLS的初始化握手原理图

正如上面两张图里演示的原理，TCP 协议连接建立的成本相对较高。

所以，一般的稳定网络传输都是通过TCP，但是在网络基建本身就已经越来越完善的情况下，TCP设计本身的问题便暴露了出来，特别是在弱网环境下，让我们不得不考虑一些新的可能性。

2、QUIC协议登场

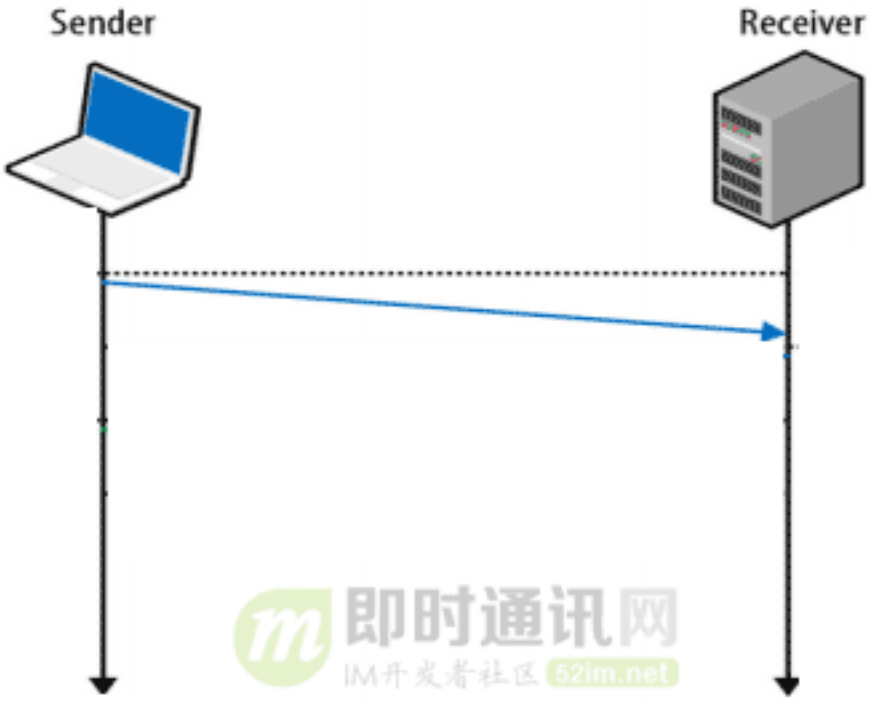
和 TCP 相反，UDP 协议是无连接协议。客户端发出 UDP 数据包后，只能“假设”这个数据包已经被服务端接收。这样的好处是在网络传输层无需对数据包进行确认，但存在的问题就是为了确保数据传输的可靠性，应用层协议需要自己完成包传输情况的确认。

此时，QUIC 协议就登场了。

QUIC 是 Quick UDP Internet Connections 的缩写，谷歌发明的新传输协议。

与 TCP 相比，QUIC 可以减少延迟。

QUIC 协议可以在 1 到 2 个数据包（取决于连接的服务器是新的还是已知的）内，完成连接的创建（包括 TLS）（如下图3所示）。

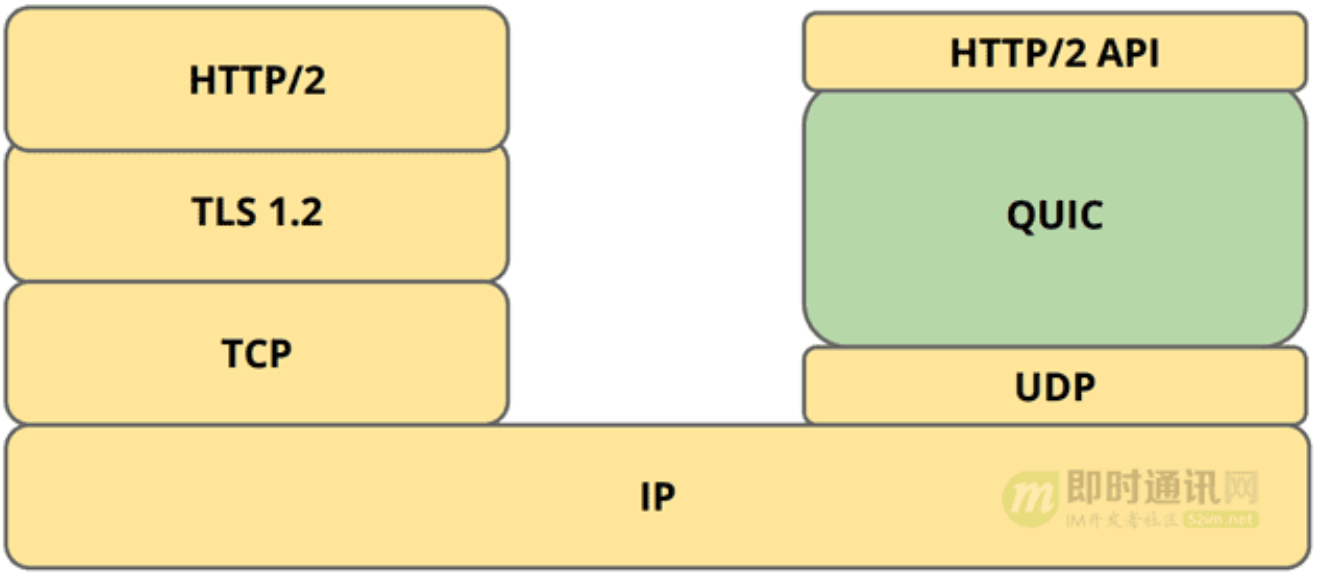


▲图 3 - QUIC 协议握手原理图

从表面上看：QUIC 非常类似于在 UDP 上实现的 TCP + TLS + HTTP/2。由于 TCP 是在操作系统内核和中间件固件中实现的，因此对 TCP 进行重大更改几乎是不可能的（TCP 协议栈通常由操作系统实现，如 Linux、Windows 内核或者其他移动设备操作系统。修改 TCP 协议是一项浩大的工程，因为每种设备、系统的实现都需要更新）。但是，由于 QUIC 建立在 UDP 之上，因此没有这种限制。QUIC 可以实现可靠传输，而且相比于 TCP，它的流控功能在用户空间而不在内核空间，那么使用者就不受限于 CUBIC 或是 BBR，而是可以自由选择，甚至根据应用场景自由调整优化。

QUIC 与现有 TCP + TLS + HTTP/2 方案相比，有以下几点主要特征：

- 1）利用缓存，显著减少连接建立时间；
- 2）改善拥塞控制，拥塞控制从内核空间到用户空间；
- 3）没有 head of line 阻塞的多路复用；
- 4）前向纠错，减少重传；
- 5）连接平滑迁移，网络状态的变更不会影响连接断线。



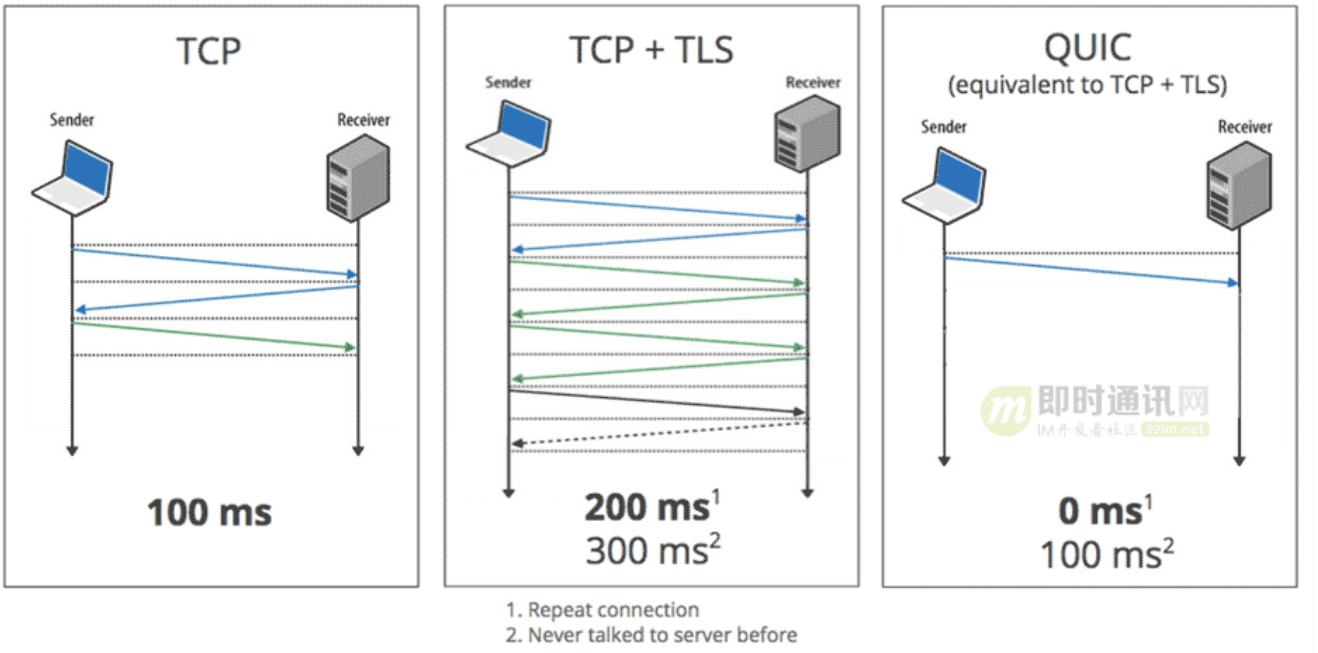
从图上可以看出，QUIC 底层通过 UDP 协议替代了 TCP，上层只需要一层用于和远程服务器交互的 HTTP/2 API。这是因为 QUIC 协议已经包含了多路复用和连接管理，HTTP API 只需要完成 HTTP 协议的解析即可。

有关QUIC的详解请见：《技术扫盲：新一代基于UDP的低延时网络传输层协议——QUIC详解》。

3、QUIC协议的目标

QUIC 协议的主要目的，是为了整合 TCP 协议的可靠性和 UDP 协议的速度和效率。

一张图看懂QUIC协议的优势：



对于 Google 来说优化 TCP 协议是一个长期目标，QUIC 旨在创建几乎等同于 TCP 的独立连接，但有着低延迟，并对类似 SPDY 的多路复用流协议有更好的支持。 如果 QUIC 协议的特性被证明是有效的，这些特性以后可能会被迁移入后续版本的 TCP 和 TLS 协议（它们都有很长的开发周期）。

值得注意的是，虽然理论上来说，如果 QUIC 的特性被证明是有效的，这些特性以后可能会被迁移到后续版本的 TCP 协议中，但鉴于TCP协议长达几十年在互联网通信里的垄断地位，以及这么多年积累下来的沉重历史报复，想要根本性地优化或改进TCP协议，难度相当大（或许，有些事情，只能是想想而已，IPV6还喊了这么多年呢，不是一样没普及。。。）。

4、QUIC协议这么好，可以大规模切换为QUIC吗?

理想和现实总是有一定的差距：虽然经过多年的推广的应用，但QUIC协议目前仍未达到大量普及的阶段，在 IETF上的QUIC 依然还是草稿，并且还存在Google QUIC与IETF QUIC两类不稳定的协定。

而且，QUIC还面临以下挑战：

- 1）小地方，路由封杀UDP 443端口（ 这正是QUIC部署的端口）；
- 2）UDP包过多，由于QS限定，会被服务商误认为是攻击，UDP包被丢弃；
- 3）无论是路由器还是防火墙目前对QUIC都还没有做好准备。

5、QUIC协议实践

Chrome 浏览器从 2014 年开始已经实验性的支持了 QUIC 协议。可以通过在 Chrome 浏览器中输入 chrome://net-internals/#quic 查看是否已经支持 QUIC 协议。如果还未支持，可以在 chrome://flags/#enable-quic 中进行开启。

开始 Chrome 浏览器对 QUIC 协议的支持之后，可以在 chrome://net-internals/#quic 中查看到当前浏览器的 QUIC 一些连接。当然目前只有 Google 服务才支持 QUIC 协议（如 YouTube、 Google.com） 。

Host	Secure	Version	Peer address	Connection UID	Active stream count
36.docs.google.com:443	true	QUIC_VERSION_30	[2a00:1450:4013:c00::bd]:443	2708254184554045987	1
apis.google.com:443	true	QUIC_VERSION_30	[2a00:1450:400e:802::200e]:443	5189742635553804178	0
clients4.google.com:443	true	QUIC_VERSION_30	[2a00:1450:400e:802::200e]:443	5174608782190849431	0
i.ytimg.com:443	true	QUIC_VERSION_30	[2a00:1450:4013:c01::8a]:443	10559272118787914470	0
plus.google.com:443	true	QUIC_VERSION_30	[2a00:1450:400e:801::200e]:443	2461447815203244151	0
r18---sn-5hne6ned.googlevideo.com:443	true	QUIC_VERSION_30	[2a00:1450:401c:f::17]:443	14426173135210551355	0
s.ytimg.com:443	true	QUIC_VERSION_30	[2a00:1450:4013:c01::65]:443	814538457547024801	0
ssl.google-analytics.com:443	true	QUIC_VERSION_30	[2a00:1450:4007:80b::2008]:443	16111488254187388150	0
ssl.gstatic.com:443	true	QUIC_VERSION_30	[2a00:1450:400e:801::2003]:443	13147793992039561928	0
www.google.be:443	true	QUIC_VERSION_30	[2a00:1450:400c:c04::5e]:443	4019955848903944504	0
www.youtube.com:443	true	QUIC_VERSION_30	[2a00:1450:400e:801::200e]:443	1993955220975030604	0
yt3.ggpht.com:443	true	QUIC_VERSION_30	[2a00:1450:400e:801::2001]:443	12318925982785982092	0

Google 在 2015 年的一篇博文中分享了一些关于 QUIC 协议实现的结果，这些优势在诸如 YouTube 的视频服务上更为突出：用户报告通过 QUIC 协议在观看视频的时候可以减少 30% 的重新缓冲时间。

6、我想试试QUIC协议，可以怎么做?

目前支持 QUIC 协议的 web 服务只有 0.9 版本以后的 Caddy 。其他常用 web 服务如 nginx、apache 等都未开始支持。

整个 QUIC 协议比较复杂，想自己完全实现一套对笔者来说还比较困难。

所以先看看开源实现有哪些。

1）Chromium：
这个是官方支持的。优点自然很多，Google 官方维护基本没有坑，随时可以跟随 chrome 更新到最新版本。不过编译 Chromium 比较麻烦，它有单独的一套编译工具。暂

时不建议考虑这个方案。

2）proto-quic：
从 chromium 剥离的一个 QUIC 协议部分，但是其 github 主页已宣布不再支持，仅作实验使用。不建议考虑这个方案。

3）goquic：
goquic 封装了 libquic 的 go 语言封装，而 libquic 也是从 chromium 剥离的，好几年不维护了，仅支持到 quic-36， goquic 提供一个反向代理，测试发现由于 QUIC 版本太低，最新 chrome 浏览器已无法支持。不建议考虑这个方案。

4）quic-go：
quic-go 是完全用 go 写的 QUIC 协议栈，开发很活跃，已在 Caddy 中使用，MIT 许可，目前看是比较好的方案。

那么，对于中小团队或个人开发者来说，比较推荐的方案是最后一个，即采用 caddy 来部署实现 QUIC。caddy 这个项目本意并不是专门用来实现 QUIC 的，它是用来实现一个免签的 HTTPS web 服务器的（caddy 会自动续签证书）。而QUIC 只是它的一个附属功能（不过现实是——好像用它来实现 QUIC 的人更多）。

从Github的技术趋势来说，有关QUIC的开源资源越来越多，有兴趣可以自己逐一研究研究：https://github.com/search?q=quic

7、本文小结

QUIC 协议开创性的使用了 UDP 协议作为底层传输协议，通过各种方式减少了网络延迟。

虽然目前 QUIC 协议已经运行在一些较大的网站上，但离大范围普及还有较长的一段距离，期待 QUIC 协议规范能够成为终稿，并在除了谷歌浏览器之外的其他浏览器和应用服务器中也能够实现。

8、参考资料

- 《技术扫盲：新一代基于UDP的低延时网络传输层协议——QUIC详解》
- 《让互联网更快：新一代QUIC协议在腾讯的技术实践分享》
- 《七牛云技术分享：使用QUIC协议实现实时视频直播0卡顿！》

Google的“ Next generation multiplexed transport over UDP”文档：

Next generation multiplexed transport over UDP.pdf
(563.01 KB，下载次数: 2，售价: 1 金币)

9、系列文章

本文是系列文章中的第10篇，本系列文章的大纲如下：

- 《网络编程懒人入门(一)：快速理解网络通信协议（上篇）》
- 《网络编程懒人入门(二)：快速理解网络通信协议（下篇）》
- 《网络编程懒人入门(三)：快速理解TCP协议一篇就够》
- 《网络编程懒人入门(四)：快速理解TCP和UDP的差异》
- 《网络编程懒人入门(五)：快速理解为什么说UDP有时比TCP更有优势》
- 《网络编程懒人入门(六)：史上最通俗的集线器、交换机、路由器功能原理入门》
- 《网络编程懒人入门(七)：深入浅出，全面理解HTTP协议》
- 《网络编程懒人入门(八)：手把手教你写基于TCP的Socket长连接》
- 《网络编程懒人入门(九)：通俗讲解，有了IP地址，为何还要用MAC地址？》
- 《网络编程懒人入门(十)：一泡尿的时间，快速读懂QUIC协议》（本文）

附录：更多网络编程相关资料推荐

《TCP/IP详解 - 第11章·UDP：用户数据报协议》
《TCP/IP详解 - 第17章·TCP：传输控制协议》
《TCP/IP详解 - 第18章·TCP连接的建立与终止》
《TCP/IP详解 - 第21章·TCP的超时与重传》
《技术往事：改变世界的TCP/IP协议（珍贵多图、手机慎点）》
《通俗易懂-深入理解TCP协议（上）：理论基础》
《通俗易懂-深入理解TCP协议（下）：RTT、滑动窗口、拥塞处理》
《理论经典：TCP协议的3次握手与4次挥手过程详解》
《理论联系实际：Wireshark抓包分析TCP 3次握手、4次挥手过程》
《计算机网络通讯协议关系图（中文珍藏版）》
《UDP中一个包的大小最大能多大？》
《P2P技术详解(一)：NAT详解——详细原理、P2P简介》
《P2P技术详解(二)：P2P中的NAT穿越(打洞)方案详解》
《P2P技术详解(三)：P2P技术之STUN、TURN、ICE详解》
《通俗易懂：快速理解P2P技术中的NAT穿透原理》
《高性能网络编程(一)：单台服务器并发TCP连接数到底可以有多少》
《高性能网络编程(二)：上一个10年，著名的C10K并发连接问题》
《高性能网络编程(三)：下一个10年，是时候考虑C10M并发问题了》
《高性能网络编程(四)：从C10K到C10M高性能网络应用的理论探索》
《高性能网络编程(五)：一文读懂高性能网络编程中的I/O

模型》

《高性能网络编程(六)：一文读懂高性能网络编程中的线程模型》

《不为人知的网络编程(一)：浅析TCP协议中的疑难杂症(上篇)》

《不为人知的网络编程(二)：浅析TCP协议中的疑难杂症(下篇)》

《不为人知的网络编程(三)：关闭TCP连接时为什么会TIME_WAIT、CLOSE_WAIT》

《不为人知的网络编程(四)：深入研究分析TCP的异常关闭》

《不为人知的网络编程(五)：UDP的连接性和负载均衡》

《不为人知的网络编程(六)：深入地理解UDP协议并用好它》

《不为人知的网络编程(七)：如何让不可靠的UDP变的可靠？》

《不为人知的网络编程(八)：从数据传输层深度解密HTTP》

《不为人知的网络编程(九)：理论联系实际，全方位深入理解DNS》

《技术扫盲：新一代基于UDP的低延时网络传输层协议——QUIC详解》

《让互联网更快：新一代QUIC协议在腾讯的技术实践分享》

《现代移动端网络短连接的优化手段总结：请求速度、弱网适应、安全保障》

《聊聊iOS中网络编程长连接的那些事》

《移动端IM开发者必读(一)：通俗易懂，理解移动网络的“弱”和“慢”》

《移动端IM开发者必读(二)：史上最全移动弱网络优化方法总结》
《IPv6技术详解：基本概念、应用现状、技术实践（上篇）》
《IPv6技术详解：基本概念、应用现状、技术实践（下篇）》
《从HTTP/0.9到HTTP/2：一文读懂HTTP协议的历史演变和设计思路》
《脑残式网络编程入门(一)：跟着动画来学TCP三次握手和四次挥手》
《脑残式网络编程入门(二)：我们在读写Socket时，究竟在读写什么？》
《脑残式网络编程入门(三)：HTTP协议必知必会的一些知识》
《脑残式网络编程入门(四)：快速理解HTTP/2的服务器推送(Server Push)》
《脑残式网络编程入门(五)：每天都在用的Ping命令，它到底是什么？》
《脑残式网络编程入门(六)：什么是公网IP和内网IP？NAT转换又是什么鬼？》
《以网游服务端的网络接入层设计为例，理解实时通信的技术挑战》
《迈向高阶：优秀Android程序员必知必会的网络基础》
《全面了解移动端DNS域名劫持等杂症：技术原理、问题根源、解决方案等》
《美图App的移动端DNS优化实践：HTTPS请求耗时减小近半》
《Android程序员必知必会的网络通信传输层协议——UDP和TCP》

《IM开发者的零基础通信技术入门(一)：通信交换技术的百年发展史(上)》
《IM开发者的零基础通信技术入门(二)：通信交换技术的百年发展史(下)》
《IM开发者的零基础通信技术入门(三)：国人通信方式的百年变迁》
《IM开发者的零基础通信技术入门(四)：手机的演进，史上最全移动终端发展史》
《IM开发者的零基础通信技术入门(五)：1G到5G，30年移动通信技术演进史》
《IM开发者的零基础通信技术入门(六)：移动终端的接头人——“基站”技术》
《IM开发者的零基础通信技术入门(七)：移动终端的千里马——“电磁波”》
《IM开发者的零基础通信技术入门(八)：零基础，史上最强“天线”原理扫盲》
《IM开发者的零基础通信技术入门(九)：无线通信网络的中枢——“核心网”》
《IM开发者的零基础通信技术入门(十)：零基础，史上最强5G技术扫盲》
《IM开发者的零基础通信技术入门(十一)：为什么WiFi信号差？一文即懂！》
《IM开发者的零基础通信技术入门(十二)：上网卡顿？网络掉线？一文即懂！》
《IM开发者的零基础通信技术入门(十三)：为什么手机信号差？一文即懂！》
《IM开发者的零基础通信技术入门(十四)：高铁上无线上网有多难？一文即懂！》
《IM开发者的零基础通信技术入门(十五)：理解定位技

术，一篇就够》

《百度APP移动端网络深度优化实践分享(一)：DNS优化篇》

《百度APP移动端网络深度优化实践分享(二)：网络连接优化篇》

《百度APP移动端网络深度优化实践分享(三)：移动端弱网优化篇》

《技术大牛陈硕的分享：由浅入深，网络编程学习经验干货总结》

《可能会搞砸你的面试：你知道一个TCP连接上能发起多少个HTTP请求吗？》

《知乎技术分享：知乎千万级并发的高性能长连接网关技术实践》

>> 更多同类文章 ……