**多路径 TCP**

学号 杨小帆

北京理工大学计算机学院XXXXXX班，北京 100081

(qbhou@nlpr.ia.ac.cn)

#### Multipath TCP

Yang Xiaofan

(Class XXXXX,School of Computer Science, Beijing Institute of Technology, Beijing 100081)

**Abstract** Multipath TCP (MPTCP) is an extension of the traditional TCP protocol that enables a single transport connection to transmit data across multiple paths simultaneously, thereby improving throughput and connection reliability. This paper provides a comprehensive overview of the protocol structure and major extensions of MPTCP, including RFC 6824 and RFC 8684. It further examines the protocol’s scalability and security mechanisms, such as key negotiation and defenses against man-in-the-middle attacks. The paper also reviews research progress on various packet scheduling algorithms used in MPTCP, including MinRTT, redundant transmission, BLEST, and ECF. In addition, a comparative analysis is conducted between MPTCP and QUIC in terms of multiplexing, multipath support, and security architecture. While QUIC uses TLS encryption and supports native stream multiplexing, MPTCP relies on TCP options and HMAC-based verification. Finally, the paper introduces the current state of MPTCP implementation in iOS and Linux systems, highlighting typical application scenarios such as seamless connectivity in Apple’s Siri assistant and server-side bandwidth aggregation through multiple network interfaces.

**Key words** Multipath TCP; TCP extension; multipath communication; multipath transport; network protocol security

摘要 多路径 TCP (MPTCP) 是对传统 TCP 的扩展，使一个传输连接可以跨越多条路径并行通信，从而提高吞吐量和连接可靠性。本文综述了 MPTCP 的协议结构及主要扩展（RFC 6824 和 RFC 8684 等），以及其可扩展性和安全机制（包括密钥协商和防止中间人攻击的方法）。同时评述了常见的 MPTCP 调度算法（如最小 RTT、冗余发送、BLEST、ECF等）的研究进展。还比较了 MPTCP 与 QUIC 在多路复用、多路径支持和安全方面的设计差异和性能表现，指出 QUIC 原生使用 TLS 加密并支持多流，而 MPTCP 则依赖 TCP 选项和基于密钥的 HMAC 机制。最后介绍了 MPTCP 在 iOS 和 Linux 系统中的实现现状及典型应用场景，例如 Apple Siri 语音助手的无缝接入和服务器端多链路聚合。摘要五号楷体，300字左右

关键词多路径TCP；TCP 扩展；多链路通信；多路径传输；网络协议安全

随着移动互联网的发展和多网络接口设备的普及，传统 TCP 协议所依赖的单一路径通信模式逐渐暴露出瓶颈。在智能手机、平板电脑、笔记本电脑等移动终端上，用户通常同时连接多个网络（如 Wi-Fi 与蜂窝网络），但原生 TCP 协议无法同时利用多条路径，只能从中选择一条作为通信通道。这种限制不仅无法实现带宽叠加，也降低了通信的鲁棒性，一旦主路径发生拥塞或断链，通信会中断并重新建立连接，造成应用层体验下降。

为解决上述问题，IETF 多路径 TCP 工作组提出了Multipath TCP（MPTCP）协议，其核心思想是在保留 TCP 报文格式和语义的基础上，通过扩展选项支持多个子流并行传输，从而实现在传输层的路径冗余、容错能力增强与带宽聚合。MPTCP 保持对应用层透明，这意味着不需要修改任何现有的网络应用即可获得性能提升。这种设计使得 MPTCP 能够平滑部署于现有互联网环境中，同时兼容中间设备如 NAT、防火墙等。

MPTCP 最初的标准规范由 RFC 6824（2013）定义，并在多年实践和反馈后，于 2020 年被更新为 RFC 8684，该文档不仅对握手过程、安全机制和可扩展性进行了完善，也显著增强了 MPTCP 在部署中的稳健性和互操作性。与此同时，学术界和工业界围绕 MPTCP 的调度算法、路径管理、拥塞控制和安全性等方面展开了广泛研究，提出了一系列用于优化多路径性能的策略，如 BLEST、ECF、Redundant 等调度器，以应对路径异构性和动态变化等挑战。

MPTCP 的工程价值已被实际部署所验证。Apple 是该协议最早的大规模应用者之一，从 iOS 7 起在 Siri 语音助手中引入 MPTCP，用于保障语音请求在 Wi-Fi 和蜂窝网络切换间的无缝传输。此后，Apple 将 MPTCP 扩展至系统级 API，允许第三方应用透明使用多路径连接，提升下载稳定性和实时交互能力。此外，Linux 内核自 5.6 起也将 MPTCP 正式合入主线，逐步成为数据中心和边缘设备多链路接入的解决方案。

然而，MPTCP 也面临一些新的挑战与竞品协议的对比压力。近年来兴起的基于 UDP 的 QUIC 协议通过用户态实现了可靠传输、原生 TLS 加密与多路复用，并逐步被 HTTP/3 等现代协议栈采用。虽然 QUIC 默认并不支持多路径传输，但其灵活的设计为未来的 MP-QUIC 留下了发展空间。因此，比较 MPTCP 与 QUIC 在协议设计、安全性、多路径能力与应用部署方面的异同，也是当前研究的热点。

综上所述，MPTCP 是传输层协议发展的重要方向，其标准演进、核心机制、调度算法、系统实现以及与 QUIC 的对比是非常值得研究的。本文将围绕上述方面展开综述。

## MPTCP 协议结构与主要扩展

Multipath TCP（MPTCP）是在不破坏传统 TCP 语义的前提下，为支持多路径并行传输而对 TCP 进行的扩展。其设计目标是向上兼容 TCP 应用和现有网络栈，同时支持在不同 IP 地址、不同接口之间建立多个子流（subflow），提升吞吐和容错能力。这一扩展最早由 IETF MPTCP 工作组提出，正式的协议规范最初由 RFC 6824 定义，后在 2020 年被 RFC 8684 所取代，成为当前广泛接受的标准实现版本。

### 1.1 协议设计总览

MPTCP 在保留“可靠、按序字节流”这一传统 TCP 特性的同时，通过在 TCP 报文头部引入新的选项字段，增加了多路径连接管理与数据调度机制。例如，它定义了 MP\_CAPABLE、MP\_JOIN、ADD\_ADDR、REMOVE\_ADDR、DATA\_FIN 等一系列选项，用于连接初始化、路径添加/移除、连接关闭等操作（RFC 8684，第 3~4 节）[1]。其中MP\_CAPABLE 选项用于建立最初的主连接（initial subflow），并协商 MPTCP 功能与密钥信息；MP\_JOIN 用于在初始子流之后创建额外子流，连接同一对端MPTCP实例；ADD\_ADDR 与 REMOVE\_ADDR 选项使连接双方能够通告新地址或通知对方地址不可达，实现动态路径管理；DATA\_SEQUENCE\_SIGNALING（如 DSN 映射）用于在接收端重建字节流顺序。

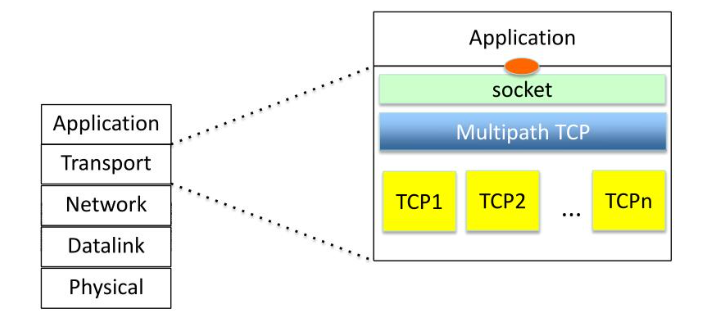
每条子流在 TCP 层独立拥塞控制和丢包恢复，但共享统一的 MPTCP 连接上下文。为了维护整体数据顺序，MPTCP 引入了Data Sequence Number（DSN）系统，将应用数据映射到全局 DSN 上，并通过每个子流的序号与其映射关系进行重组，从而解决多路径乱序的问题（RFC 8684，第 3.3 节）[1]。

图1 多路径TCP及其架构

Fig.1 Multipath TCP and the architecturet

**1.2 RFC 6824与RFC 8684 的差异与演进**

RFC 6824 是最早定义 MPTCP 的标准文档，于 2013 年发布[2]，它为 MPTCP 奠定了架构基础，并引入了上述选项机制。然而在实际部署和实验中，研究者与工程团队发现了诸多细节问题，例如 HMAC 验证强度不够、选项字段歧义、部分功能定义不够清晰等。

因此，IETF 于 2020 年发布了RFC 8684对其进行标准化修订，主要改进包括：在子流加入与地址通告（ADD\_ADDR）等过程中引入强制 HMAC 签名，防止第三方伪造控制信息（以加强安全性RFC 8684 §3.4.1、§3.6.1）[1]；对 MP\_CAPABLE 和 MP\_JOIN 等选项字段定义了更严格的格式解析规范，消除歧义（RFC 8684 §3.1.1）；引入 connection-level token与receiver token用于识别连接身份和子流关联，便于连接恢复与迁移；删除了一些实验性的功能，例如RFC 6824中定义但未广泛采用的DSS Checksum被删除，以简化整个协议的实现（RFC 8684 §5.3.3）[1]。

总之，RFC 8684对原始架构进行了标准化和增强，提升了协议鲁邦性、互操作性与部署可行性。

**1.3 与网络中间设备的兼容性与回退机制**

为了在真实网络中可用，MPTCP 特别设计了中间盒（middlebox）兼容性机制。由于许多网络设备（如 NAT、防火墙）对未知 TCP 选项或非标准行为持阻拦态度，MPTCP 在设计上采用以下策略：每个子流均为合法 TCP 连接，能够独立完成握手过程；所有 MPTCP 控制信息通过标准 TCP 选项字段封装；一旦发现对方或中间设备不支持 MPTCP，端系统可以自动回退为普通 TCP 通信（fallback），保证应用不受影响（RFC 8684 §3.1.1）[1]；

Apple 在 官方支持文档 中指出：“如果远程服务器或其网络不支持 MPTCP，系统将自动建立标准 TCP 连接”[3]。

一项研究[4]也指出，MPTCP 的 fallback 机制极大提升了协议部署的成功率，并确保其能在现有互联网架构中渐进式部署。

**2** MPTCP 扩展性与安全机制

MPTCP 设计考虑了高度可扩展性。在子流管理方面，它定义了地址添加(ADD\_ADDR)和移除(REMOVE\_ADDR)等控制消息，允许在会话期间动态添加或移除可用网络接口地址。此外，还提供子流优先级 (MP\_PRIO) 等机制供应用控制链路偏好。[5]

安全性方面，MPTCP 采用基于对称密钥的方案来防止中间人攻击。按照 RFC 6181 的建议，双方在第一条子流建立时交换一个随机密钥（通常以明文方式放在握手报文里），后续新增子流时则使用该密钥进行 HMAC 校验。RFC 8684 进一步指出：“MPTCP 连接的安全性依赖于在第一个子流开始时共享的密钥，并且该密钥不会再次在网络上传输”。通过这种方式，即使潜在攻击者截获了初始握手，也无法在后续交换中伪造子流。为了应对 NAT 等中间盒造成的地址篡改攻击，MPTCP 要求每次添加新地址前进行可达性检查（如握手确认），并使用 HMAC 对 ADD\_ADDR 消息中的地址进行校验。除了默认方案，标准还允许使用预共享密钥来加强保护，并建议引入序号防重放等机制。对于更高安全需求的场景，研究者还提出了将 MPTCP 与 TLS/SSL 或 tcpcrypt 等加密方案结合的思路，以利用现有安全协议对传输数据进行加密认证。

**3** MPTCP 调度算法研究现状

MPTCP 的性能在很大程度上受数据包调度算法

影响。最简单也是默认的调度策略是MinRTT，即优先使用往返时延最小的可用子流。Linux 内核集成的 MPTCP 实现（内核 5.6+）默认采用该算法（也支持 Round-Robin、冗余发送等简单方式。然而，当可用链路特性（RTT、带宽）差异较大时，MinRTT 会导致慢链路阻塞整个连接。针对异构网络，不少研究提出了改进调度策略：BLEST（Blocking ESTimation）算法在发送前估计阻塞风险，以避免将包发送到高延迟链路；ECF（Earliest Completion First）算法则综合考虑 RTT 和拥塞窗口等信息，选择预计完成最早的路径[6]。实验结果表明，ECF 在多链路聚合场景下能更充分地利用所有路径带宽，改善视频流和 Web 浏览的性能。除上述算法外，还有 Round-Robin、固定优先级、冗余副本（Redundant）等策略，以及 MuSher、STTF（Smallest-Transmission-Time-First）等学术方案在不同文献中被提出。Linux 内核社区和研究者正在不断丰富调度模块，以适应不同应用对延迟敏感性和带宽利用率的需求。

**4** MPTCP 与 QUIC 设计差异及性能比较

QUIC 是新一代基于 UDP 的传输协议（HTTP/3 的基础），原生集成了 TLS 加密并支持多流复用。与之相比，MPTCP 继续使用 TCP 语义，需要单独使用上层 TLS/SSL 来加密数据。

根据De Coninck等人[8]的分析，QUIC从一开始就包含了用于认证服务器和协商加密密钥的密码机制，因此 QUIC 对数据和几乎所有头部字段都进行加密[8]；而 MPTCP 本身只加密了控制消息，数据负载默认未加密。另一方面，MPTCP 天生支持多路径：一连接可同时跨越多条 TCP 子流；而当前 QUIC 规范仅支持单一路径（一个 UDP 源-目的对），多路径需依赖扩展。

正如一项研究指出的：“QUIC 当前无法支持（如 Siri 场景的）多路径切换，因为它使用单个 UDP 流” [8]。目前已有提案尝试为 QUIC 增加多路径能力（Multipath QUIC），但尚处于开发阶段。由于 QUIC 的多流和加密特性，MP-QUIC 在设计上可以更灵活地分配帧和避免中间盒干扰，因此在性能上具有优势。实验证明，在无丢包环境下，MP-QUIC 通常能跑赢 MPTCP；即使在有随机丢包的条件下，MP-QUIC 也往往表现更好。此外，QUIC 的多流特性能够天然避免头阻塞（HoL）问题[8]，而 MPTCP 在多链路情况下可能出现慢链路阻塞快链路的情况。

综上，MPTCP 和 QUIC 在分层设计、流复用方式、安全策略等方面存在显著差异：MPTCP易部署且兼容 TCP 应用，但安全性依赖于子流级别的 HMAC；QUIC 则以加密和灵活复用为特点，但需依赖更复杂的协议栈。

**5** MPTCP在操作系统中的实现与典型应用场景

**5.1 iOS**

Apple 在 iOS7 及以上版本中集成了 MPTCP 支持，用于增强网络连接的可靠性。官方文档指出，当用户向 Siri 发问时，设备会尝试在 Wi-Fi 上建立 MPTCP 主连接，并在蜂窝数据上建立备用连接；如果 Wi-Fi 失效，系统可无缝切换到蜂窝链路[3]。这一特性使得 Siri 在用户移动时不易因Wi-Fi掉线而中断[3][9]。从 iOS11 开始，任何应用均可显式请求使用 MPTCP，使用场景包括数据下载、语音识别等对连接连续性要求高的业务[3][9]。macOS 同样在 10.10 及以上版本提供了类似支持。

**5.2 Linux**

Linux 社区提供了开放源码的 MPTCP 内核实现，项目主页显示自 Linux 5.6 起内核主线已开始合并 MPTCP 代码。2020 年正式将 MPTCP 合入内核，此后不断更新（最新稳定版 v0.96 发布于 2023 年）。

该实现目前主要在服务器端或需要链路聚合的系统中使用。根据 Cloudflare报道，“目前主要有两种操作系统实现了MPTCP：Linux和iOS/macOS”。在服务器端部署的通常是 Linux MPTCP，而在客户端（如手机、笔记本）则部署 iOS 版本[6]。

**5.3 MPTCP的典型应用场景**

除了移动设备无感切换场景外，MPTCP 还可用于服务端聚合多链路带宽。例如在数据中心，传统做法是对链路做物理 bonding，但 MPTCP 可在传输层利用多条路径并行传输，实现按字节级流量汇聚[6]。实际案例包括三星 Galaxy S6 结合 Wi-Fi 和 LTE 达到 1Gbps 下载速率。在科研和运营中，还涌现出诸如 OpenMPTCPRouter 等项目，利用 MPTCP 在边缘设备实现多链路聚合。

总的来说，MPTCP 在各大操作系统（iOS、macOS、Linux）均已得到支持，典型应用涉及移动互联网、服务器链路聚合、物联网网关等领域，展示了多路径传输对提升连接健壮性和吞吐的潜力。

## 6 结束语

Multipath TCP 作为一种创新的传输层扩展，通过同时利用多条路径，为多网卡环境带来了更高的吞吐和更好的可靠性。本文概述了 MPTCP 的协议结构与演进（尤其是 RFC 8684 规范）、扩展机制及其安全设计，并梳理了主流的调度算法研究。进一步比较了 MPTCP 与现代传输协议 QUIC 在设计理念和性能方面的差异，指出各自的优势与适用场景。最后介绍了 MPTCP 在 iOS 和 Linux 中的实现现状以及典型应用案例，如 Apple Siri 的无缝切换、服务器带宽聚合等。未来，随着多路径传输需求的增长，MPTCP 及其与其他协议的集成（如与 TLS/SSL 安全层结合、QUIC 多路径扩展）将继续成为研究和工程关注的重点。

**参 考 文 献**

[1] Ford, A., Raiciu, C., Handley, M., Bonaventure, O. (2020). *RFC 8684: TCP Extensions for Multipath Operation with Multiple Addresses*. IETF. <https://datatracker.ietf.org/doc/html/rfc8684>

[2] Ford, A., Raiciu, C., Handley, M., Bonaventure, O., & Barre, S. (2013). *RFC 6824: TCP Extensions for Multipath Operation with Multiple Addresses (Obsoleted)*. IETF. https://datatracker.ietf.org/doc/html/rfc6824

[3] Apple Inc. (2023). *Use Multipath TCP to create backup connections for iOS*. <https://support.apple.com/en-us/101905>

[4] De Coninck, Q., Bonaventure, O. (2017). *Multipath QUIC: Design and Evaluation*. In Proc. CoNEXT 2017. <https://multipath-quic.org/conext17-deconinck.pdf>

[5] Bagnulo, M., Paasch, C., & Bonaventure, O. (2011). *RFC 6181: Threat Analysis for TCP Extensions for Multipath Operation with Multiple Addresses*. IETF. https://datatracker.ietf.org/doc/html/rfc6181

[6] Cloudflare. (2021). *Multipath TCP in Practice: A Look at Scheduler Performance*.

[7] Ferlin, S., Alay, Ö., Dreibholz, T., & Paasch, C. (2016). *BLEST: Blocking Estimation-based Scheduler for MPTCP*. In IFIP Networking 2016.

[8] De Coninck, Q., & Bonaventure, O. (2017). *Multipath QUIC: Design and Evaluation*. In Proceedings of the 13th International Conference on emerging Networking EXperiments and Technologies (CoNEXT '17). ACM. <https://multipath-quic.org/conext17-deconinck.pdf>

[9] Barre, S., Paasch, C., & Bonaventure, O. (2011). *Multipath TCP: From Theory to Practice*. In Proc. IFIP Networking. Université catholique de Louvain.

[10] PACS-1： public-access computer systems forum [EB/OL]．Houston,Tex：University of Houston Libraries, 1989 [1995-05-17]．http://info.lib.uh.edu/pacsl.html

[11] Dublin Core Metadata Initiative. Dublin Core Metadata Element Set, Version 1.1: Reference Description [EB/OL]. (2003-06-02)[2005-03-21]. <http://dublincore.org/documents/2003/08/26/usageguide>