计算机研究与发展
Journal of Computer Research and Development
DOI:10.7544/issn1000-1239.2016.20150589
53(11):2512-2529,2016

基于 MPTCP 的多路径传输优化技术综述

薛开平[1,2] 陈 珂[1,2] 倪 丹[1] 张 泓[1] 洪佩琳[1,2]
[1](中国科学技术大学电子工程与信息科学系 合肥 230027)
[2](中国科学院无线光电通信重点实验室(中国科学技术大学) 合肥 230027)
(kpxue@ustc.edu.cn)

Survey of MPTCP-Based Multipath Transmission Optimization

Xue Kaiping[1,2], Chen Ke[1,2], Ni Dan[1], Zhang Hong[1], and Hong Peilin[1,2]
[1](*Department of Electronic Engineering and Information Science*, *University of Science and Technology of China*, *Hefei* 230027)
[2](*Key Laboratory of Wireless-Optical Communications* (*University of Science and Technology of China*), *Chinese Academy of Sciences*, *Hefei* 230027)

Abstract The newly developed networks hope to bring some specific features, such as wide coverage, high bandwidth, and provide varieties of media services. Fortunately, multiple access techniques are at there to help. Nowadays, terminals are always equipped with multiple interfaces to adapt to the heterogeneous networks. Thus, there are more than one paths existing between two endpoints. MPTCP (multipath TCP) is proposed by IETF MPTCP working group, in addition to compatible with traditional TCP, which utilizes multiple paths to transmit data. It improves the efficiency of bandwidth usage and the resilience of the connection. What's more, it can adaptively move data from more congested path to less congested path. In recent years, many researches have been carried out to gradually make MPTCP from concept to practice. This paper introduces the technical details of MPTCP standard, and expounds the present situation of the related aspects, including simulation and actual deployment, out-of-order control, coupled congestion control, energy consumption maintenance, security, and other aspects such as path selection, mobility and QoS, the data center scenario, et al. Finally, this paper gives a summary and expectation.

Key words multiple path; multipath TCP (MPTCP); scheduling algorithms; congestion control; energy consumption management; security

摘 要 新型网络环境致力于为用户提供广覆盖、高带宽、包含各种多媒体业务的服务,而多种接入技术并存的网络环境为无处不在的高质量服务提供了可能.为了适应异构环境,用户终端通常配置多个接口(比如 WIFI 和 4G),通信终端之间可能存在多条路径.多径 TCP 协议(multipath TCP, MPTCP)是 IETF MPTCP 工作组提出的新型传输层多径协议,它在兼容 TCP 协议的基础上,同时利用多条路径的传输能力来进行数据传输,提高带宽利用率,增强连接的恢复能力,并且能够自适应地将数据从拥塞路径转移到非拥塞路径.近年来国内外机构先后作出了大量的研究,逐步使得 MPTCP 由概念走向实用.总结和阐述了 MPTCP 相关方面的研究现状,其中包括仿真实现与实际部署、乱序控制、联合拥塞控制、能耗管理、安全以及关于路径选择、移动性和 QoS、数据中心中的应用等其他方面的研究.最后给出总结和展望.

收稿日期:2015-06-17;**修回日期**:2015-12-29
基金项目:国家自然科学基金项目(61379129,61390513);国家"八六三"高技术研究发展计划基金项目(2014AA01A706)
This work was supported by the National Natural Science Foundation of China (61379129, 61390513) and the National High Technology Research and Development Program of China (863 Program) (2014AA01A706).

关键词 多路径;多径 TCP 协议;调度算法;拥塞控制;能耗管理;安全

中图法分类号 TP393

随着多种接入技术,如 DSL,WIFI,3G/4G 等的发展,用户终端通常具备多个接口(也被称为多宿主(multi-homing))来支持不同的接入技术. 比如,笔记本终端既有以太网有线接口又有 WLAN 接口,还可以支持 3G/4G;一般智能手机终端同时支持 3G/4G 和 WLAN. 但是,传统的 TCP/IP 协议只允许终端一次通过一个接口获取服务. 在多种接入方式带宽充足时只选择一种接入会造成一定的资源浪费,并且用户手动切换接入方式还会不可避免地造成移动过程中的服务中断. 为了解决这些问题,出现了对多路径传输技术的研究,并且近年来该方向的研究愈演愈热.

多路径传输技术旨在研究多宿终端之间如何同时使用多条路径进行数据传输,从而实现带宽聚合、负载均衡、动态切换以及自动将业务从最拥塞、最易中断的路径上转移到较好的路径上的功能.

多路径传输将应用层的业务数据流分发到多条路径并行传输,而多路径的传输控制功能可以在终端协议栈的任意层次进行实现,目前已经存在很多研究在不同层次考虑多路径传输的实现[1-4]. 为了保证对应用层的透明性,同时由于在 TCP/IP 协议栈中传输层是最低的端到端层次,可以获取所用路径带宽、延迟、丢包等信息,并且能够通过拥塞控制算法对网络中的拥塞事件作出快速反应,故而传输层多路径方案一直是多径传输技术研究中的重点.

Internet 工程任务组(Internet Engineering Task Force, IETF)于 2000 年提出一种传输层多径协议 SCTP(stream control transmission protocol),它是独立于 TCP 和 UDP 协议的传输层协议. 标准 SCTP 协议[5]中定义,一个 SCTP 连接可以使用多个 IP 地址,但是只有主路径用于数据传输,其他路径作为冗余备份,在主路径失效时才启用. 之后的研究又提出了 LS-SCTP(load sharing for SCTP)[6]和 CMT-SCTP(concurrent multipath transmission for SCTP)[7],对标准 SCTP 协议进行扩展以支持多径并行传输. 然而,SCTP 协议至今没有大规模部署,究其原因是其与现有的应用程序不兼容,通信双方的应用都必须调用 SCTP 接口才能使用 SCTP 协议传输.

鉴于当前 Internet 中大部分重要的应用程序都基于 TCP, Huitema[8] 早在 1995 年就提出基于 TCP 的多路径传输控制思想,IETF 在 2009 年专门成立了多径 TCP 协议(multipath TCP, MPTCP)工作组[9],旨在研究在现有 TCP 会话基础上增加多径传输的功能,包括相应的体系结构、拥塞控制、路由、API、安全等. 要求 MPTCP 与现有的网络架构和协议兼容,对应用层提供的仍然是传统 TCP 套接字,应用程序在不做任何更改的情况下也能使用,从而保证对应用层的透明性,并且在终端不支持 MPTCP 时可以回退到 TCP.

MPTCP 从根本上改变了数据的调度和传输方式,通过同时建立多条传输路径,将数据的传输方式由传统的单径变成了多径,有效地提升了网络的传输能力和稳定性,具有非常重要的意义.

1 MPTCP 协议技术细节

MPTCP 利用多条路径进行端到端传输,通信双方必须同时支持 MPTCP 协议,并且至少一端具有多个 IP 地址,实现多路径的分离. 当前 IETF MPTCP 工作组制定的标准(RFC)一览如表 1 所示,工作组主要讨论的内容包括 MPTCP 的协议细节[10-11],联合拥塞控制[12-14]、安全问题[15-16]、应用层接口[17]、MPTCP 代理[18]等.

Table 1 The List of IETF MPTCP RFCs

表 1 IETF MPTCP RFC 文档一览

Standard No.	Standardization Document Name	Date
RFC 6181[15]	Threat Analysis for TCP Extensions for Multipath Operation with Multiple Addresses	2011-03
RFC 6182[11]	Architectural Guidelines for Multipath TCP Development	2011-03
RFC 6356[12]	Coupled Congestion Control for Multipath Transport Protocols	2011-10
RFC 6824[10]	TCP Extensions for Multipath Operation with Multiple Addresses	2013-01
RFC 6897[17]	Multipath TCP (MPTCP) Application Interface Considerations	2013-03
RFC 7430[16]	Analysis of Residual Threats and Possible Fixes for Multipath TCP (MPTCP)	2015-07

MPTCP 的协议栈如图 1 所示,传输层的 TCP 划分为 2 部分:MPTCP 层和 TCP 层(也叫子流层). MPTCP 层对于应用层是透明的,应用层看见的

仍然是传统 TCP. MPTCP 层实现路径管理、数据包调度、拥塞控制等功能,并与子流层有接口,而子流层使用标准 TCP 协议.

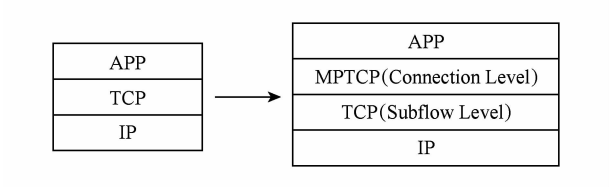

Fig. 1 MPTCP protocol stack.

图 1 MPTCP 协议栈

1.1 MPTCP 头标和基本操作

为保证 MPTCP 与 TCP 的兼容性,MPTCP 利用 TCP 头部区域携带 MPTCP 的选项,如图 2 所示. MPTCP 选项中的 Kind 域指明该选项是一个 MPTCP 选项,而 Subtype 域则指明该选项是 MPTCP 中的何种选项(比如 MP_CAPABLE,MP_JOIN,DSS,ADD_ADDR 等).

接下来我们给出 MPTCP 中的基本操作以及相应的简要描述,如表 2 所示,具体描述请参考 RFC 6824[10].

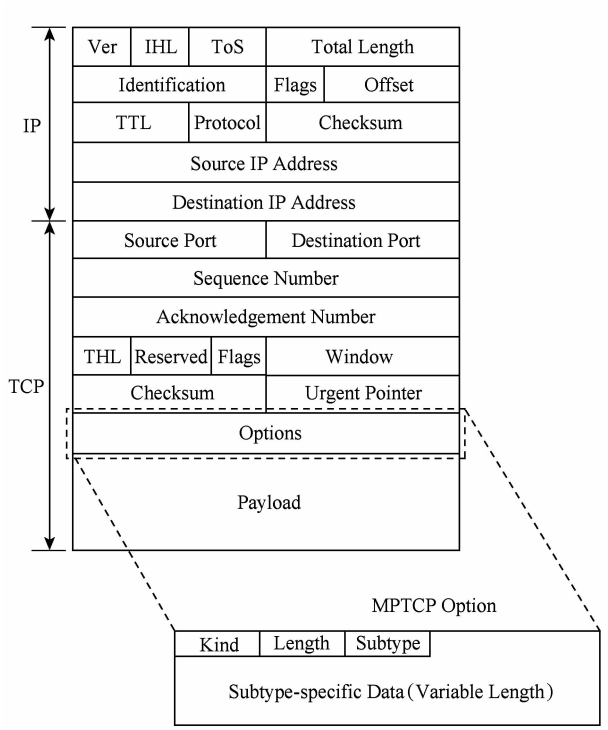

Fig. 2 The position of MPTCP header option in TCP.

图 2 MPTCP 头标选项在 TCP 中的位置

Table 2 Basic Operations of MPTCP

表 2 MPTCP 基本操作

Basic Operations	Description
Connection Initiation	Use MP_CAPABLE option to declare that its sender is capable of performing MPTCP and contains some corresponding information. Begin with a SYN, SYN/ACK, ACK exchange on a single path. Only if both ends support MPTCP, MPTCP can take place; otherwise it will gracefully fall back to regular single-path TCP.
Starting a New Subflow	Use MP_JOIN option to contain some corresponding information of staring a new subflow. It needs the three-way handshake of SYN, SYN/ACK and ACK.
Address Knowledge Exchange	Contain the implicit exchange and the explicit exchange. The former one uses ADD_ADDR to inform the coresponding node, and the coresponding node uses MP_JOIN option to setup new subflows. The latter one directly uses MP_JOIN option to setup new subflows.
Data Transmission	Use dual sequence numbers (data seuqence number and subflow sequence number) and dual acknowledgments (the connection level acknowledgment and the subflow level acknowledgment) both in connection level and subflow level. Expand congestion control, reliability, retransmission, flow control and so on of TCP.
Connection Close	Use MP_CAPABLE option to declare that its sender has no data to transmit. Each subflow uses standard TCP FIN to close.

1.2 MPTCP 功能综述和设计原则

MPTCP 层在协议栈中位于应用层之下和 TCP 层之上,为应用层提供标准的 TCP 接口来隐藏多径,同时需要管理多条 TCP 子流. MPTCP 层具备路径管理、数据包调度、子流接口及拥塞控制的功能,总结这 4 点如下:

1) 路径管理. 路径管理功能负责检测和使用 2 个主机之间的多条路径. MPTCP 的路径管理机制要求把本机可用的地址发送给对端主机,而且可以建立新子流并加进已有的 MPTCP 连接中.

2) 数据包调度. 此功能要求把从应用程序接收到的字节流(stream)切割成分段,后续由子流进行

传输. MPTCP 设计了序列号映射机制,每条子流上发送的数据包都对应有一个连接级别的序列号,从而保证接收端可以正确重组从不同子流上接收到的数据包. 数据包调度功能依赖于路径管理功能所提供的可用路径信息.

3) 子流(采用单独路径 TCP)接口. 子流功能从数据包调度功能中取得分段,在特定的路径上确保到主机可检测的传送. MPTCP 考虑了网络兼容性,在下层使用 TCP,确保有序可靠的传送. TCP 在子流层给每个数据包添加子流序列号,用于实现子流级别的检测和重传丢包. 在接收端,每条 TCP 子流在按序重组本子流的数据包后递交给数据包调度模块,继而进行连接级别的数据包按序重组.

4) 拥塞控制. 用于在多个子流之间协调窗口,从而实现拥塞控制功能. MPTCP 设计了联合拥塞控制算法[12],保证在共享瓶颈处一个 MPTCP 连接与传统单径 TCP 相比不会占用过多的带宽,从而实现公平性.

总之,这些功能彼此配合、相互协作. 路径管理功能获取两通信主机间的可用路径;数据包调度功能获取应用数据流后需要进行一定的操作,比如分割数据段、添加连接级别的序列号,而后将数据包分发至各条 TCP 子流;子流再添加子流级别的序列号和 ACK 到每个数据包,接收端子流将数据包按序交付给数据包调度功能进行进一步的连接级别数据包的按序重组;拥塞控制功能是数据包调度功能的一部分,它决定了哪些数据包以何种速率在哪条子流上进行发送.

MPTCP 设计原则和优化目标可归纳为 5 点:

1) 提升吞吐量. 一个 MPTCP 连接的总吞吐量不应小于其最好路径上的单条 TCP 连接的吞吐量.

2) 公平性. 与只使用其中的一个路径时相比,MPTCP 连接不能占用过多的资源.

3) 均衡拥塞. 在满足前 2 项的前提下,多径流应该尽可能多地从最拥塞的路径上转移走一些业务.

4) 安全性. MPTCP 要求其所具有的安全性不差于单个 TCP 连接.

5) 恢复力. 在最坏情况下,MPTCP 的恢复力必须不低于常规的单路径 TCP.

2 仿真实现与实际部署

伦敦大学学院(University College London, UCL)的网络研究组开发了最早用于 MPTCP 方案验证和性能仿真的仿真器 htsim[19]. Google 公司负责开发和维护了目前最有影响力和知名度的 NS3 (network simulator)仿真环境下的 MPTCP 模块[20],该模块基本实现了在一个 MPTCP 连接下多 TCP 子流并行传输的功能. 虽然代码并没有严格按照 MPTCP 协议所规定的进行设计,但不影响将其作为仿真工具进行方案的性能评估. 另一个用于 NS3 的 MPTCP 模块由 Sussex 大学(University of Sussex)的 Kheirkhah 等人[21-22]提供.

UCL 网络研究组和以及来自比利时鲁汶天主教大学(Université catholique de Louvain, UCLouvain)的 IP 网络实验室分别提供了 MPTCP 的 Linux 内核开源代码[19,23],并在内核代码的基础上进行了 MPTCP 性能的测试[24-27],从拥塞窗口调整、数据中心支持、移动性支持、能耗考虑等多个方面对基本的 MPTCP 协议提出一系列机制改进. 另外一个重要的开源实现是以补丁的形式由 Apple 公司的研究组所提供的[28],并用于 IOS 和 MacOS 当中. 这 3 个研究组以及前面提及的来自 Google 公司的研究组也是 IETF MPTCP 工作组的主要参与者.

图 3 是内核代码框架示意图[23]. 从图 3 可以看到,应用层调用的是传统 TCP 套接字(socket),而在建立了 MPTCP 连接后调用多径控制模块(multipath control block),该模块也就是所说的 MPTCP 层. 初始的 TCP 套接字为主套接字,之后新建立的 TCP 套接字为从 TCP 套接字. 无论主 TCP 套接字上的子流关闭与否,主 TCP 套接字一直存在,用于向上层应用隐藏 MPTCP 的存在,从而实现透明化. 在具体实现[25,29]上,如图 4 所示,首先每个子流的处理都通过一个 *tcp_sock* 来加以实现,除此之外,该 *tcp_sock* 还包含一个指向 *mptcp_tcp_sock* 的指针,用以维护 TCP 之外与 MPTCP 有关的信息以及处理 MPTCP 层的相关功能. MPTCP 连接所产生的第 1 个子流的 *tcp_sock* 记为 *meta_sock*,供应用层调用的为标准的 TCP socket api. 作为 *meta_sock* 的 *tcp_sock* 除了包含指向 *mptcp_tcp_sock* 的指针之外,还将通过指针连接到一个新添加的 *mptcp_cb*,该结构体作为 MPTCP 的控制缓存,其中维护了到各个 TCP 子流的 *tcp_sock* 的指针以及地址列表信息等. 各个子流的 *mptcp_tcp_sock* 也需要将相关标识信息等存储在 *mptcp_cb*.

NS3 中 MPTCP 模块[20]中类的实现主要包含 4 个部分:

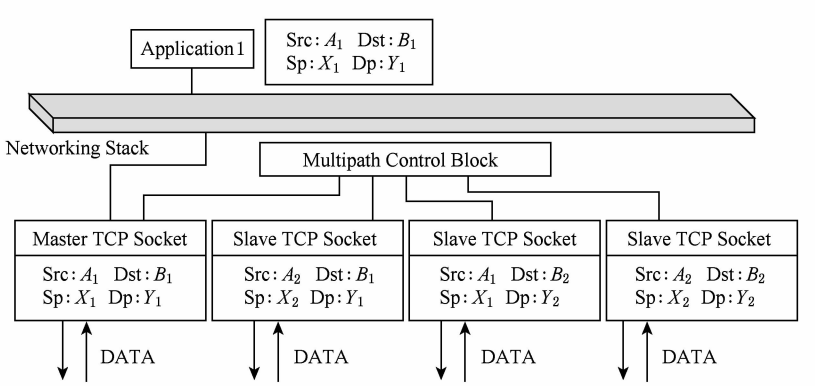

Fig. 3 Structure of MPTCP kernel code.

图 3 MPTCP 内核代码框架

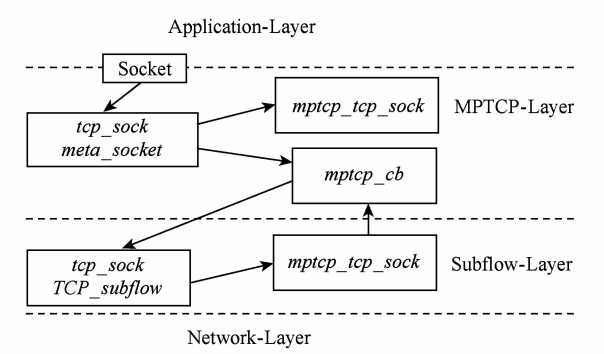

Fig. 4 Relationship and function description of structures in Linux implementation.

图 4 Linux 内核实现中结构体相互关系及功能

1) MpTcpSocketImph. MpTcpSocketImph 是类 TcpSocketImpl 的子类,它给应用层提供 MPTCP API(connet,bind 等)处理多路连接和包重排序.

2) MpTcpL4Protocol. MpTcpL4Protocol 是类 TcpL4Protocol 的子类,是多路传输层和网络层之间的接口.

3) MpTcpSubflow. MpTcpSubflow 提供 MPTCP 连接的子路径.

4) MpTcpHeader. MpTcpHeader 是 TcpHeader 的子类,提供了 MPTCP 的头选项等.

目前,除了 Linux 系统内核支持的开发之外,研究者和开发者还在 Android 系统进行了大量的尝试,可以用于 Google Nexus,Samsung Galaxy 系列等[30]. Apple IOS,MacOS 以及数据中心 Amazon EC2[31-32]也应用了 MPTCP. 除此之外,斯威本科技大学(Swinburne University of Technology, SWIN)先进互联网架构中心(Centre for Advanced Internet Architectures)开发了用于 FreeBSD10. x 的 MPTCP 代码[33].

目前的趋势是:随着 MPTCP 受到学术界和业界的重视,越来越多的研究者和开发者投入 MPTCP 相关功能的实现当中,日趋呈现百家争鸣之势.

3 乱序控制

MPTCP 与任何传统多径传输协议一样面临数据包乱序问题的重大挑战. 大多数多径并行传输协议的初衷是为了聚合各路径带宽,提升数据传输吞吐量. 然而多条路径将引入更为复杂的参数维度,亦对发送端的多径控制能力提出更高的要求.

以往的研究[26,34]表明 MPTCP 吞吐量受到通信双方间各条端到端路径的时延、丢包率、带宽等参数差异以及共享接收缓存大小的限制. 随着各路径差异越大,共享接收缓存越小,MPTCP 吞吐量性能急剧恶化,有时甚至比不上最好路径上使用传统 TCP 协议进行单径传输的吞吐量. 导致这一现象的主要原因是路径差异导致接收端数据包乱序,而在接收缓存过小无法容纳大量乱序包时性能将进一步恶化. 接收端乱序会造成 2 方面的影响:

1) MPTCP 多路径共享一个接收缓存,接收端乱序导致接收缓存阻塞.

2) 限制路径拥塞窗口滑动和增长,路径传输速率偏低.

这 2 方面因素引起 MPTCP 各路径拥塞控制窗口增长相互制约,严重影响吞吐量提升. 针对乱序导致的头阻塞问题,为了尽可能保证数据按序达到,现有方案主要包括发送端调度策略和 MPTCP 与网络编码相结合的方案.

3.1 MPTCP 发送端调度算法

MPTCP 的发送端调度算法可以解决路径不对称造成的接收端数据包乱序问题,尽力保证数据包按序到达接收端. 下面介绍 MPTCP 相关研究中提出的调度机制[25,27,34-40].

轮询算法(round-robin, RR)[25]是最简单的调度算法. MPTCP 连接的各个子流没有优先级,发送端轮询各个子流,发送缓存的数据按照轮询的顺序填满各子流的发送窗口进行发送. 此时,只是在多子流间简单地调配数据包,数据的按序性无法保证.

最小 RTT 优先轮询算法(lowest RTT first RR)[27]在轮询算法的基础上进行了一定的深化. MPTCP 连接的各个子流按照 RTT 排列优先级,RTT 越小优先级越高. 发送端按照优先级高低轮询各个子流,依次取发送缓存的数据包填满各个子流的发送窗口. 该算法让路径质量好的子流承载更多的数据包,有一定的负载均衡的效果. 无论是轮询算法还是最小 RTT 优先轮询算法都没有考虑数据包的按序性,没有从数据包序列号角度出发.

Linux-MPTCP 调度算法[25]是 Linux-MPTCP 内核代码支持的一种预测调度算法. 以图 5 为例说明该算法的思想:一个 MPTCP 连接上的数据需要调度到 2 个 TCP 子流($subflow_i$ 和 $subflow_j$)上发送. 假设调度时刻 2 条子流的拥塞窗口均为 2,2 条子流的 RTT 值存在 $RTT_j=5RTT_i$ 的关系,也就是说,$subflow_i$ 上发送 5 轮数据所需的时间与 $subflow_j$ 上发送 1 轮数据包所需时间相同. 发送一轮是指拥塞窗口(cwnd)内的第 1 个数据包从发送开始直到接收到该数据包的 ACK,发送一轮的时间相当于一个 RTT. 如果使用 RR 算法,$subflow_i$ 取数据包 1,2,$subflow_j$ 则取数据包 3,4,那么,$subflow_i$ 上数据包 1,2 的 ACK 回来后,接着取数据包 5,6,数据包 5,6 将早于数据包 3,4 到达接收端,乱序的数据包需要缓存,且 $subflow_i$ 后续几轮的数据包仍将早于数据包 3,4 到达接收端. Linux-MPTCP 调度算法在调度时刻 $subflow_j$ 不取数据包 3,4 而是取数据包 11,12. $subflow_i$ 上发送 5 轮数据包,窗口大小为 2,一共可以发送 10 个包,$subflow_j$ 则将发送缓存中的前 10 个包都预留给 $subflow_i$ 后续发送. 这样一来,数据包 1~12 将能基本按序依次到达接收端. 不过该算法过于简略,没有考虑拥塞窗口在这 5 轮发送过程中会按照拥塞控制算法变化.

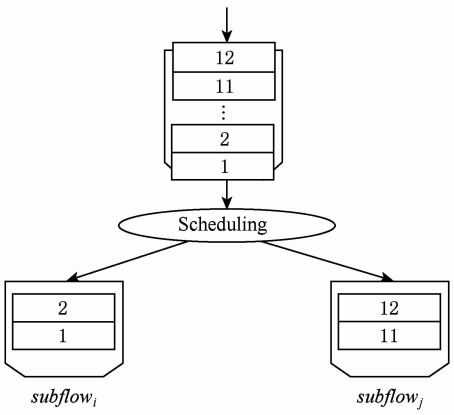

Fig. 5 Scenario example of the Linux-MPTCP scheduling algorithm.

图 5 Linux-MPTCP 调度算法适用的场景示例

前向预测调度(forward prediction scheduling, FPS)[34-35]是 Linux-MPTCP 调度算法的改进版本,想法来源于 Westwood SCTP 算法[41],更精细地考虑了序列号的预测调度算法. 假设一共有 I 条子流,对于 $path_i$ 而言,其往返时延为 RTT_i,数据包从离开发送端到到达接收端的时延为 $FT_i(\approx RTT_i/2)$,拥塞窗口大小为 s_i. 为了简化处理,数据包大小都设为 TCP 最大分段长度 MSS(maximum segment size),故而 $subflow_i$ 上每一个数据包的排队时延 ε_i 是确定的:

$$\varepsilon_i=\frac{PacketSize}{Throughput}=\frac{MSS}{Throughput}. \tag{1}$$

在时刻 t,$subflow_i$ 发送 s_i 个数据包,那么这些数据包到达接收端的时刻分别为

$$t+FT_i+\varepsilon_i,t+FT_i+2\varepsilon_i,\cdots,t+FT_i+s_i\varepsilon_i,$$

而发送端接收到这些数据包的 ACK 的时刻分别为

$$t+RTT_i+\varepsilon_i,t+RTT_i+2\varepsilon_i,\cdots,t+RTT_i+s_i\varepsilon_i.$$

发送端接收到数据包成功接收的确认消息后拥塞窗口 s_i 会按照拥塞控制算法增加,新一轮数据在时刻 $t+RTT_i+s_i\varepsilon_i$ 发送.

以图 6 为例说明 FPS 算法. MPTCP 连接的数据通过 $path_i$ 和 $path_j$ 这 2 条路径传输,2 条路径往返时延满足 $RTT_j>RTT_i$. 在时刻 t,路径 $path_j$ 上发送窗口产生滑动,从而准备发送新数据. 由于 $path_i$

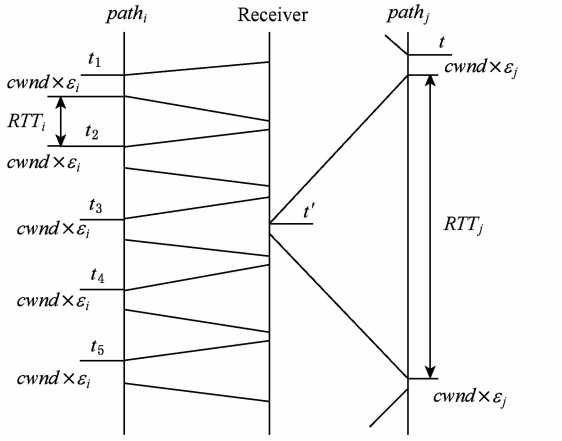

Fig. 6 Transmission sequence diagram of two subflows in FPS.

图 6 FPS 算法中 2 条子流数据包传输时序图

传输数据快于 $path_j$，$path_j$ 上发送的数据包需要经过调度. 发送端首先估计 $path_j$ 上数据包到达接收端的时刻为 t'，然后计算在时刻 t' 之前 $path_i$ 上可以发送的数据包总数. 假设 n_l 是从时刻 $t_l(t_l<t')$ 开始的一轮数据传输中在 t' 之前可以到达接收端的数据包数目，数据包到达接收端的时刻则为

$$t_l+FT_i+\varepsilon_i, t_l+FT_i+2\varepsilon_i, \cdots, t_l+FT_i+n_l\varepsilon_i, \tag{2}$$

其中，n_l 必须满足：

$$n_l \leqslant s_i, t_l+FT_i+n_l\varepsilon_i<t'. \tag{3}$$

如果 $t_l+FT_i+\varepsilon_i>t'$，那么该轮传输的所有数据包均在 t' 之后到达，此时 $n_l=0$，最后计算 t' 之前可以传输的数据包总数为

$$N=\sum_{t_l<t'} n_l, \tag{4}$$

其中，N 就是在 t' 之前在 $path_i$ 上成功发送的数据包总数. 那么，发送端调度发送缓存的前 N 个包给 $path_i$，$path_j$ 跳过发送缓存前 N 个数据包从第 $N+1$ 个开始取来填满自己的发送窗口. $path_i$ 上每接收到一个数据包成功接收的确认消息，RTT_i 平滑更新：

$$RTT_i \leftarrow \alpha RTT_i+(1-\alpha)rtt_i, \tag{5}$$

其中，rtt_i 是新测量得到的往返时延；α 是 0～1 之间的值，指明历史 RTT_i 和新测量 rtt_i 的权重关系. 单程时延 FT_i 按式(6)简单估计：

$$FT_i=RTT_i/2. \tag{6}$$

由于在无线链路环境中随机丢包事件不可避免，不考虑丢包的前向预测机制显然已经不再适用. 在 FPS[34-35] 基础上，F²P-DPS(fine-grained forward prediction based dynamic packet scheduling)[36] 充分考虑了子流的 TCP 特性以及路径的丢包率，使用 TCP 建模的方法，发送端根据子流上平滑得到的 RTT 和统计得到的丢包率估计未来一段时间内该子流可能发送的数据量 N. 所采用的方法是，根据子流上给出的初始条件，以 $subflow_i$ 为例，初始窗口大小 a 以及路径参数 (p_i, RTT_i)，对 $[t, t']$ 内 $subflow_i$ 上所有可能的数据包传输事件计算统计平均，最终该统计平均值即为 N.

但从根本上讲，FPS 和 F²P-DPS 都只是预测算法，可能与实际情况有差距. 为了防止多轮调度之后的误差累计，进一步在 F²P-DPS 的基础上，一种基于 TCP SACK 反馈的调度算法 OCPS(offset compensation based packet scheduling)[37] 被提出. 方案中在子流级别使用 TCP SACK 返回当前接收端乱序情况，发送端根据 TCP SACK 携带的信息来判断上一轮调度时预留给其他子流的数据包是过多还是过少，从而使用修正因子对下一轮的预留数据包总数进行修正. OCPS 在路径质量存在一定扰动的情况下有良好的适应能力.

3.2 结合网络编码的方案(MPTCP/NC)

借鉴单路径中引入网络编码带来的好处，大量研究者尝试在多路径 TCP 中也引入网络编码，一方面可以解决无线场景下由于无线随机丢包带来的问题[42-43]，另一方面可以简化多路径 TCP 各子流之间的调度问题[44-46].

以图 7 和图 8 为例进行说明，假定所有的丢包都是由于无线随机丢包造成的. 在图 7 中，子流 S_0 与子流 S_1 均非编码数据流，子流 0 传输 D_3, D_4, D_6，子流 1 传输 D_1, D_2, D_5，传输过程中假定子流 S_0 上存在无线随机丢包，D_3 由于无线传输错误而无法正常接收，则将会出现 2 个问题：1)子流 S_0 传输的 D_4 首先到达接收端，由于小序列号数据段未到达，则 D_4 只能暂时缓存在接收缓存中，不可以提交；2)当发生丢包后，接收端需要显式地发回反馈消息后发送端才会重新发送分组，这不仅通报了假的拥塞信息，同时较长的丢包通报延时再一次加重了连接级别的乱序. 如图 7 所示，当 D_4 到达时，接收端反馈 D_3 丢失了，D_4 不得不被缓存在接收缓存中等待小序列号数据段 D_1, D_2, D_3 的到达.

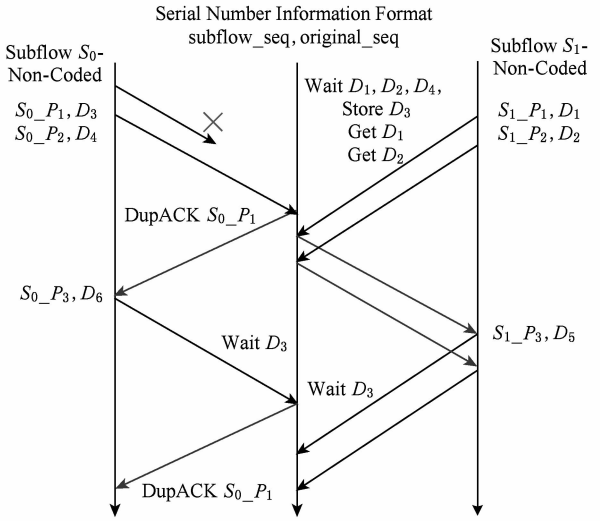

Fig. 7 An example of MPTCP transmission.

图 7 MPTCP 传输示例

与此不同的是，在图 8 中，由于采取了编码机制，按照 TCP 网络编码的原则，接收端并不以完全解码出原始数据段作为成功接收的标准，只要接收端在编码的分组中可以看到数据段，则认为已经接收

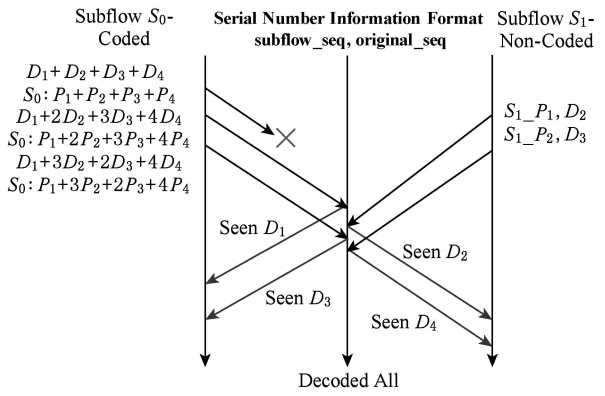

Fig. 8 An example of MPTCP subflow layer coded transmission.

图 8 MPTCP 子流级别编码传输示例

到数据段,通过这种方式,使得 2 条子流不需要严格遵循序列号顺序.①发送端会发出冗余的分组,在不需要进行显示反馈的情况下,丢失的分组就可以被恢复,进一步缩短接收缓存被占用的时间;②无论是编码子流还是非编码子流,只要到达的编码分组满足 2 个条件,则发回成功接收的确认消息,避免造成拥塞误判:

1) 编码分组线性独立于其他收到的编码分组;

2) 最新收到的编码分组中包含有接收端期望的序列号信息,或者可以借助新接收的编码分组从解码缓存中推导出接收端期望的序列号信息.

从编码内容上,多路径 TCP 网络编码的方式主要可以分为子流级别编码[42-43]以及连接级别编码[44-46].子流级别编码对多路径 TCP 协议的改进较小,兼容性更好.子流级别编码和连接级别编码的主要不同之处在于:执行数据在不同子流的分配调度与执行编码的先后顺序.若先执行数据段调度,再执行编码,则为子流级别编码;否则为连接级别编码.

从传输形式上看,可以分为编码子流与非编码子流[43].在图 8 中,子流 0 是编码子流,而子流 1 是非编码子流.

多路径 TCP 网络编码的确认机制尚未统一,可以分为子流级别确认机制和连接级别确认机制.子流级别编码采用子流级别的确认机制[42-43],连接级别编码可以采用子流级别确认[44]或者连接级别的确认机制.子流级别确认机制与多路径 TCP 协议后向兼容性好,所有的丢包均在子流级别进行分析,以子流级别序列号信息作为反馈依据,反馈信息通报的丢包和延时变化均是该路径上的情况,便于做出拥塞判决.连接级别反馈仅适用于连接级别编码,以连接级别序列号信息作为反馈依据,不需要在连接级别和子流级别做映射,但是其不足之处在于,通过序列号信息可以获得丢包信息,由于不再与子流信息相关,无法准确判定丢包发生在哪一条路径上.在图 8 中,若反馈的序列号信息为 SSN:PX,则是子流的反馈;若为 DSN:DX,则为连接级别的反馈.

3.3 其他可行的方案

除了以上提到的这 2 类方案可以用于解决包乱序的问题之外,MPTCP 内核部署研究[26,34]中还提到了 4 种简单的改进机制:

机制 1. 在最快路径上重传阻塞数据包.当发生接收缓存阻塞时,将未到达接收端的序列号最小的数据包在最快路径上重传.最快路径是 RTT 最小的路径.

机制 2. 惩罚阻塞接收窗口滑动的子流.发送端判断阻塞接收缓存的子流,也即未到达接收端的序列号最小的数据包所在子流,并对该子流的拥塞窗口减半惩罚操作.不过为了防止总是惩罚某一条子流,在一个往返时延内同一条子流只能惩罚 1 次.

机制 3 和机制 4. MPTCP 的发送和接收缓存支持自动调整,在需要时才增大缓存,这样可以防止直接设置大缓存造成的空间浪费.当接收缓存中所存储的乱序数据包的数目超过 1 个 BDP(BDP 的计算值为 $Bandwidth \times Delay$)大小时,对拥塞窗口设置上限.MPTCP 中具体的部署方式是:当 sRTT (smoothed RTT)超过了 2 倍的基准 RTT 时就为拥塞窗口设置上限,基准 RTT 为所有子流中的最小 RTT 值.机制 3 与机制 4 在 sRTT 的测量方式上有所不同.机制 3 使用数据包 timestamp 选项所记录的发送与接收时间来测量 sRTT 值,而机制 4 使用子流接收一个接收窗口大小的数据所需要的实际时间来估算 sRTT 值.

3.4 乱序控制小结

MPTCP 接收端数据包乱序现象将直接导致接收端缓存阻塞,并进一步阻止发送端的拥塞窗口向前滑动.现有处理方案主要集中在 3 个方面:1)依据 MPTCP 各子流特性为其预分配数据包,使得不同子流所交付的数据包保持基本有序;2)使用网络编码的方式掩盖上层数据包丢失状况,发送冗余的编码数据包以保证接收端数据包有序解码;3)调整快速重传策略以迅速向接收端交付造成缓存阻塞的数据包,使得接收端数据有序.目前尚无切实可行的方案可以完全应对乱序问题,以上方案的设计也只是处于初步阶段,这与 MPTCP 归根结底依然属于端到端协议存在一定关系,网络状态是实变的,源端

只能根据反馈信息估计出当前的网络状况,因而具有一定的滞后性,这也就给方案的有效执行带来了难度.所设计方案往往很难从根本上解决问题,但力求可以更加准确地做到这一点,将乱序问题的影响降到最低.

4 联合拥塞控制算法

MPTCP的每条TCP子流可以使用标准的TCP拥塞控制算法,而联合拥塞控制算法是MPTCP研究的一大热点.沿袭单路径TCP拥塞控制的思想,多路径TCP拥塞控制机制可以分为2类:1)以丢包作为拥塞发生的标志;2)以延时变化评估当前是否发生拥塞.

4.1 基于丢包判据的拥塞控制算法

在MPTCP提出之前,采用的方法是一个连接的多个子流共享同一个拥塞窗口,比如E-TCP[47],CM[48],MPAT[49].在这些方案中,窗口调整都采用了基本的AIMD(additive increase/multiplicative decrease)方式[50]:如果在一个RTT中没有发生丢包,拥塞窗口就增加1;如果共享此拥塞窗口的任一子流发生了丢包,拥塞窗口就减半.这类方法在减小窗口的算法上过于激进,会对没有发生丢包的子流带来过多的性能限制,因此在公平性上有很大缺陷.之后多路径TCP的拥塞控制机制设计为每个子流拥有自己的拥塞窗口,在AIMD基础上,不同子流之间采用联合拥塞控制的方法.比如EWTCP(equally-weighted TCP)[51],COUPLED[52-53],SEMI-COUPLED[54],LIA[55],RTT-Compensator[56],OLIA[57]等.表3给出了这6种方案的窗口调整方式.

EWTCP方案中,通过"加权"限制多路径TCP在瓶颈链路上的吞吐量,多路径TCP子流以高耦合方式增大拥塞窗口,以独立的方式减少拥塞窗口.参数a表示多路径TCP流与单路径TCP流吞吐量的关系,当$a=1$,可以满足多路径TCP流与单路径TCP流吞吐量相当.EWTCP的不足之处在于,未考虑不同子流RTT的不同.

COUPLED算法可以保证数据流迁移至最不拥塞的路径上,其不足之处在于:1)COUPLED仅考虑RTT相同的假设条件;2)COUPLED算法在流量迁移过程中会切断最拥塞的路径,当该条路径恢复非拥塞后,也无法再继续使用.

SEMI-COUPLED对COUPLED的改进在于,SEMI-COUPLED偏向于使用拥塞程度轻的路径,但不会完全封闭拥塞最为严重的路径,而是在拥塞的路径中留一部分少量的数据以监测拥塞的路径,当发送端的该路径拥塞状态减轻至低于其他路径的程度时,将负载回迁至该路径上,减轻其他更为拥塞的路径压力.SEMI-COUPLED依旧基于RTT相同的假设.

Table 3 Window Size Change of MPTCP Congestion Control Schemes

表3 MPTCP各种拥塞控制机制窗口调整方式

Congestion Control Scheme	Window Size Increasing	Windows Size Decreasing
EWTCP[51]	$\Delta w_r=a/w_r$, $a=1/\sqrt{n}$	$w_r/2$
COUPLED[52-53]	$\Delta w_r=1/w_{\text{total}}$	$w_{\text{total}}/2$
SEMI-COUPLED[54]	$\Delta w_r=a/w_{\text{total}}$	$w_r/2$
LIA[55]	$\Delta w_r=a/w_{\text{total}}$, $a=\hat{w}_{\text{total}}\dfrac{\max\limits_r \hat{w}_r/RTT_r^2}{(\sum\limits_r \hat{w}_r/RTT_r)^2}$	$w_r/2$
RTT-Compensator[56]	$\Delta w_r=\min(a/w_{\text{total}},1/w_r)$, $a=\hat{w}_{\text{total}}\dfrac{\max\limits_r \hat{w}_r/RTT_r^2}{(\sum\limits_r \hat{w}_r/RTT_r)^2}$	$w_r/2$
LIA[57]	$\Delta w_r=\dfrac{w_r/RTT_r^2}{(\sum\limits_r \hat{w}_r/RTT_r)^2}+\dfrac{a}{w_r}$, $a=\begin{cases}\dfrac{1/n}{\lvert B\backslash M\rvert}, & r\in B\backslash M,\\ -\dfrac{1/n}{\lvert M\rvert}, & r\in M,\ B\backslash M\neq\varnothing,\\ 0, & \text{otherwise.}\end{cases}$	$w_r/2$

LIA从瓶颈公平性的角度出发,其最大的改进之处在于该机制考虑到各路径RTT的不同设计出a的参考值,用来控制窗口变化的加速度.a的值与各条子流的RTT有密切的关联.

RTT-Compensator在LIA算法之上进行改进,提出每收到一个ACK,则窗口的增大幅度上限设定为$1/w_r$,其目的是为了保证若一条或者多条多路径TCP子流经过一个瓶颈链路,在相同的拥塞丢包率下以同一瓶颈中TCP流窗口增大值作为上限.RTT-Compensator兼顾了不同路径RTT的区别,是最为常用的多路径TCP拥塞控制机制,其也被应用于多路径TCP网络编码传输机制中[43].

OLIA延用LIA中控制窗口变化加速度的思想,从公平性以及最优资源池角度对LIA进行了改进.划分路径优劣集合,为不具有最大拥塞窗口的较好路径提供更大的窗口变化加速度,并减小较差路径的窗口变化加速度,从而使得资源利用最大化,

并使得拥塞链路中的大部分发送数据被转移.目前 OLIA 已经在 Linux-MPTCP 内核代码中实现.

4.2 基于延时判据的拥塞控制算法

在单路径 TCP 拥塞控制中,TCP Vegas[58]是典型的基于延时的拥塞控制算法,其主要思想是基于延时的变化估计网络中缓存的数据包数量,将其固定在一定的数量范围,调整拥塞窗口的大小.由于延时对拥塞表现得更为敏感,因此可以在数据流未拥塞至丢包的状态时便触发拥塞控制,调整速率,减少分组的丢失.在此基础之上,将基于延时的思想引入至 MPTCP 中,便提出了基于延时的多路径 TCP 拥塞控制机制——加权 Vegas(wVegas)算法[59].wVegas 算法遵循拥塞均衡准则,即网络资源效用最大化出现在网络中每一条路径上的拥塞程度相同的时候,实现拥塞程度相同的方法是流量迁移.wVegas 机制中,欠拥塞的路径可以获得更大的权重,该路径可以以更为激进的方式竞争网络带宽,这使得这条路径倾向于更为拥塞;反之,过拥塞的路径获得的权重较小,通过反复变化,最终实现各条路径上负载均衡.

在 wVegas 算法中,式(7)表示因拥塞路径中缓存的分组数量;式(8)中,q_r 表示该条路径的拥塞状况,这里采用基于延时的拥塞控制方式,使用延时的增加作为拥塞程度的度量;$a_{s,r}$ 代替 $diff$ 表示多路径 TCP 流 s 在路径 r 上因拥塞缓存的分组量,且吞吐量可以表示为 $x_{s,r}=cwnd_{s,r}/rtt_{s,r}$,最终可以用式(9)表示路径上的吞吐量.经过推导,在 wVegas 算法中,速率的变化可以用式(10)表示,a_s 表示多路径 TCP 流在所有子流路径上缓存的分组数量,y_s 表示多路径 TCP 流总的吞吐量,多路径 TCP 流中发送速率的增加与该路径当前数据流的吞吐量占总吞吐量的比值、多路径 TCP 缓存的分组总量以及当前路径的拥塞状态 q_r 有关,其中,权重表示见式(11).

$$diff=\left(\frac{cwnd}{baseRTT}-\frac{cwnd}{rtt}\right)\times baseRTT, \quad (7)$$

$$q_r=rtt-baseRTT, \quad (8)$$

$$x_{s,r}=\frac{\alpha_{s,r}}{q_r}, \quad (9)$$

$$x_{s,r}(t+1)=\frac{x_{s,r}(t)}{y_s}\times\frac{\alpha_s}{q_r}, \quad (10)$$

$$k_{s,r}(t)=\frac{x_{s,r}(t)}{y_s}, \quad (11)$$

其中,$r\in R_s$.

4.3 动态窗口耦合机制

动态窗口耦合(dynamic windows coupling,DWC)[60]机制强调了瓶颈的检测.DWC 的核心思想在于,将经过同一个瓶颈的所有子流汇聚成为一条流,同时确保这一逻辑汇聚流与同一瓶颈上的 TCP 流之间的公平性,DWC 同时兼顾丢包率与延时去检测瓶颈链路.DWC 机制保证在某一共享的瓶颈链路上子流集合(subflow set)可以与 TCP 流保证公平性,在不同瓶颈上的子流集合可以独立地竞争带宽,最大化其传输效率,DWC 同时需要对瓶颈的迁移做出及时的响应,经过同一个瓶颈的所有子流被认为是一个子流集合.

DWC 机制主要过程包括:瓶颈检测、拥塞窗口调整.初始状态下,所有子流处于慢启动状态,每一条子流都是一个独立的子流集合.瓶颈检测过程实际上是子流集合调整的过程.瓶颈检测的触发点是某一条子流出现第 1 个拥塞丢包,继而其他子流出现延时增加、丢包的现象,则将这些子流分配至一个子流集合内,即该集合内的所有子流目前经过的路径都包含了某一或者某些瓶颈链路.其具体过程如下:假定子流 f 检测到一个拥塞丢包,则将子流 f 从其原子流集合中取出放入新的活跃集(active set)中,检测所有其他的子流,统计其在该时间点前后的延时,延时明显增加的子流进入活跃集中,在任意时刻,或者没有活跃集或者有且仅有一个活跃集,活跃集中的子流通过相同的瓶颈.DWC 以丢包为触发不断更新活跃集,并对活跃集中的子流进行拥塞控制.DWC 的不足之处在于,先后出现丢包或者延时增大的子流不一定经过同一个瓶颈,而是有可能分别经过不同的瓶颈.

4.4 联合拥塞控制算法小结

联合拥塞控制算法致力于达到最佳公平性,不过公平性的定义很多,包括子流公平性、瓶颈公平性、网络公平性等.子流公平性是指瓶颈处每条子流都是独立的,分享瓶颈的一部分;瓶颈公平性是指 MPTCP 的子流在瓶颈处与其他 TCP 子流公平分享瓶颈带宽;网络公平性是指 MPTCP 多子流的总吞吐量不应该好于最好路径单 TCP 吞吐量.侧重点不同时设计的联合拥塞控制算法就不同[60].

从更广泛的角度来说,联合拥塞控制算法还需要考虑在友好性(friendliness)、快速性(responsiveness)和窗口震荡(window oscillation)三个矛盾因素间取得折中[61].友好性是指对单 TCP 的公平性,快速性是指对于网络的拥塞做出快速的反应,窗口震荡是指子流上窗口大小的反复震动.比如快速性和窗口震荡之间的矛盾,如果 MPTCP 能对路径拥塞做出

快速反应,那么必然意味着某条子流在检测到拥塞后将快速减小窗口而在发现路径质量良好时将及时增大窗口,从而可能造成窗口大小的反复震动.

5 能耗管理

MPTCP需要开启多个接口进行数据传输,对于终端而言会增加能量消耗,而有时候开启多个接口带来能量消耗甚至造成得不偿失的后果.随着对能耗要求的重视,结合底层技术的研究进展,这方面的研究将会进一步成为热点.目前主要工作包括2个部分:1)基于MPTCP传输过程中的能耗进行量化测量;2)将能耗作为因素之一,用于性能优化.下面分别从这2方面对已有的工作进行介绍.

5.1 MPTCP能耗测量

基于对MPTCP能耗问题的关注,一些MPTCP研究小组采取了多种测量方法对MPTCP传输过程中的能耗进行了量化测量,以最小化能耗作为优化目标之一来改进MPTCP设计,从而使得其在保证QoS的同时减少能耗将具有指导性的意义.

文献[62]提出了通过邻居节点共享WIFI,Bluetooth接入从而在现存的3G/4G连接之外具有多余的连接,并通过MPTCP组合使用这些不同的接入方式形成多路径,通过在Galaxy Nexus phone上的实际测试,采用MPTCP能够在提升性能的同时有效节省能耗.在文献[10]中也提到了合理使用MPTCP可以有效地降低数据的单位传输所使用的能耗.

3G/4G和WIFI拥有不同的能耗模型.文献[63]指出,当MPTCP同时使用WIFI和3G这2种接口时,WIFI需要耗费更多的电能以维护连接,但当终端非常靠近接入点(access point, AP)时,WIFI连接可以提供高速率、低能耗的服务;另一方面,3G/4G连接相较于WIFI更易于维护,但会在文件传输完毕后仍保持一段时间的高能耗状态,即尾电消耗(tail-energy),这对于小文件传输是非常不利的.基于MPTCP的Linux内核实现,文献[64]在实际3G/WIFI环境下使用智能手机对MPTCP进行能耗测量,分别模仿了大文件下载和浏览网页的业务场景,测量结果表明,WIFI可以在文件传输完成后更快地转入节能状态,而3G网络有尾电消耗过程,由于MPTCP同时使用上述2种接口传输,故而传送每位数据所消耗的电能也位于两者之间.由于3G连接的高能耗特性,采用全连接的方式进行数据传输并不是节约电能的首选模式.在上述结论的基础上,文献[65]进一步细分了MPTCP能耗测量,分别探究了全连接模式(full-MPTCP mode)和备用模式(backup mode)下的MPTCP能耗情况.在备用模式中,MPTCP使用SYN,FIN信令开启及关闭备用路径,当使用LTE接口传输信令时,由于尾电消耗的作用,LTE连接将会保持一定时间额外的高能耗状态.这对于小文件传输,将基本无法节约能耗.

文献[66]提出MPTCP能耗应主要分为CPU能耗和连接能耗2个部分进行考虑.目前针对MPTCP能耗方面的测量和方案设计主要还是考虑连接能耗.文献[66]通过实验得出结论,目前的MPTCP还远没有得到其应该具备的性能,除了连接能耗之外,多核CPU的计算能耗也应该考虑能耗和传输能力的折中式方案设计.

此外,通过在MPTCP中引入编码[44-45,67]、高效的调度机制[25]等也可以有效地降低能耗.但从测量的角度,目前还没有发现太多这方面的工作.

5.2 MPTCP能耗优化方案

能耗大小是路径选择和数据调度的标准之一,然而能耗较低的路径可能拥塞程度较高,如何平衡两者之间的选择成为MPTCP相关研究中的一个难点.文献[68]证明了该优化问题是一个NP难问题,虽然并未提出有效的近似算法,但从理论论证了MPTCP实现吞吐量与能耗双优的可能性.

基于内核实现的减小能耗方案[69],在3G/WIFI场景下进行讨论.由于开启3G接口的能耗远大于WIFI接口,故而方案中延迟3G的开启时间.对于小文件的传输,可以在3G开启前就利用WIFI单接口传输完毕;而对于大文件的传输,在一定时延后同时启用3G和WIFI接口,从而提升吞吐量提升,实现鲁棒性以及补偿能耗的损失.

文献[70]基于移动设备使用WIFI传输数据能耗低于LTE数据传输能耗的原理,提出了MPTCP能耗优化方案eMTCP.建立子流接口状态监测器,实时监测WIFI/LTE的信道状态,当LTE需要传输数据且WIFI信道正空闲时,将数据转移到WIFI接口上进行传输,最大限度地利用低能耗接口,减少能耗.在此基础上,文献[71]进一步考虑了突发流量的情况.

文献[72-73]同时对能耗和路径选择、拥塞控制等的结合进行了考虑.其中,ecMTCP[72]在联合拥塞控制算法LIA的基础上加入能耗因子,从而设计出

一种新型的拥塞控制算法.文献[73]使用了最优化理论,通过 2 个步骤进行路径选择和拥塞控制.

然而实际上,不同接入技术在不同网络场景下都有可能具有不同的能耗表现.文献[74]认为,将 WIFI 能耗优于 3G 能耗作为固有知识是不科学的行为,故而提出在 MPTCP 中使用实时探测的方式来决定能耗较优路径.文献[75]则通过综合不同无线接口的能耗模型以及不同应用的业务模型来建立相应的预测模型,从而决定所应该选择的低能耗路径.由于其中的调度策略的高计算代价,该方案选择使用云服务器完成调度策略的计算,用户设备仅需将相关测量参数上传至云服务器即可获得选择结果.

5.3 能耗管理小结

MPTCP 协同使用多个网络接口提升了网络传输的性能,但相较于单接口协议也带来更多的能耗.如何平衡能耗与性能的关系成为 MPTCP 能耗优化方案所考虑的一个关键问题.根据不同接入网的传输效率以及能耗模型的不同,能耗优化方案大多依据上层应用的不同特性动态选择 MPTCP 所用接口,如优先使用 WIFI 网络.但由于不同接入网(如 WIFI 和 3G)覆盖范围有所不同,MPTCP 能耗优化方案也应将由地域限制所导致的频繁接口切换所带来的能耗纳入到考虑范围内,以更好地适应实际网络环境.已有的研究表明,MPTCP 路径的选择和数据分配需要考虑不同接入网络的能耗.能耗的研究往往是结合其他方面的机制进行综合考虑的.同时在某些特定条件下,在 TCP 能满足传输要求时,MPTCP 并不一定具有优势,这完全可以综合评价具有优势的接口采用 TCP 作为传输层协议.

6 安　　全

随着 MPTCP 的普及使用,其安全性也受到了越来越多的重视.首先,由于一个 MPTCP 连接是由多个 TCP 连接组成,因此 TCP 所面临的威胁同样会给 MPTCP 带来风险;其次,MPTCP 中使用的新方法会引入新的威胁.除了第 3～5 节所提及的性能目标之外,MPTCP 要求其所具有的安全性不差于单个 TCP 连接.以下将从 MPTCP 中的安全机制、面临的威胁以及对网络安全的影响 3 个方面进行阐述.

6.1 MPTCP 中的安全机制

从安全的角度,MPTCP 在连接初始化过程中要求能够提供简单的机制为后续子流的建立提供验证信息,同时也是确保所建立的多条 TCP 子流同属于一个 MPTCP 连接.在 RFC 6824[10] 当中,提出基于密钥交换来提供相应的功能.在连接初始化过程中,两端通过 3 次握手的第 2 和第 3 个消息(SYN/ACK 和 ACK)中的 MPTCP_CAPABLE 选项以明文的形式各自包含发送方的 64 b 密钥,在后续创建新子流时则可以基于双方所发送的这 2 个密钥生成相应的截断的 32 b 令牌(token),令牌用于标识一组隶属于同一个 MPTCP 的子流.另外,还产生一个 64 b 截断的密钥 Hash 用作初始数据序列号.密钥和随机的现时值(nonce)还用于消息认证码(message authentication code, MAC)的计算,用于认证和防重放.

在建立新子流时,此前在连接初始化过程中使用 MP_CAPABLE 握手过程中交换的密钥为验证终端主机提供了信息.新子流的建立也和初始化标准 TCP 连接是类似的,只是在 SYN,SYN/ACK,ACK 的数据包中携带了 MP_JOIN 选项.假设 Node *A* 在它的一个地址和 Node *B* 的地址间建立一个新的子流,如图 9 所示.从密钥中生成的令牌决定此新子流应该加入哪一个 MPTCP 连接,MAC 为验证信息.前 2 个消息中还各自包含一个现时值,用于 MAC 值的计算当中,防止重放攻击.

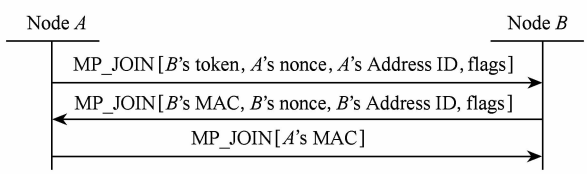

Fig. 9 Signaling flow of establishing a new subflow in MPTCP.

图 9 MPTCP 创建新子流的信令流程

6.2 MPTCP 面临的威胁

截止目前,IETF MPTCP 工作组当中有 2 个 RFC 文档[15-16]来讨论 MPTCP 可能带来的安全威胁.在 MPTCP 从草案变成标准建议[10]之前,RFC 6181[15]给出了使用单连接多地址进行数据传输的安全威胁分析,期望借用 SHIM6[76], SCTP[77], MIPv6[78]等的经验来指导 MPTCP 设计.这些可能的威胁在 RFC 6824[10]中的解决方案是采用 6.1 节中所介绍的基于交换密钥的安全机制,虽然该机制并不能从根本上解决安全威胁,但具有较高的执行效率.最近,RFC 7430[16]在 RFC 6181[15]的基础上,对 MPTCP 的安全威胁做了进一步的补充.

MPTCP 对每个端点引入多个地址的支持,与单路径 TCP 比较,MPTCP 可以在已有的连接上添加新的地址,这将会导致重定向攻击的发生. RFC 6181[15] 给出了 2 种可能的攻击方式:泛洪攻击(flooding attack)和劫持攻击(hijack attack),这两者都可以由 off-Path 的攻击者来发起攻击,而在引入一些安全机制的情况下,依然可以通过攻击者初始 on-path 获取基于明文传送的安全信息,然后依然发起 off-path 的攻击(这种组合的攻击方式称为 time-shifted 攻击或者 partial-time on-path 攻击). 这里的泛洪攻击指的是攻击者通过与服务器建立初始的 MPTCP 连接,然后欺骗服务器与攻击目标主机建立多个子流,触发其发送大量的数据流给攻击目标主机,从而耗费其资源. 而在劫持攻击中,首先由攻击者劫持合法通信两端的 MPTCP 连接,然后添加其地址用于和其中一端建立新的子流,或者作为中间人分别与合法通信的两端建立子流,而实际的通信双方还以为是与合法对端建立的子流,通过这样的方式,攻击者就可以非法获得通信数据. RFC 6181[15] 总结出 3 类可行的解决方案可供使用:1)基于 cookie 的方法;2)使用明文进行秘密交互;3)强加密锚交换. 前两者的缺陷是需要使用明文进行初始交换,无法避免 on-path 攻击者对明文交换信息的获取,但新增加的开销较小. 其中,RFC 6824[10] 引入的是第 2 种方案. 第 3 种方案的安全性较高,目前所提出的方案包括基于公钥机制[15,79] 和基于 Hash chain[80] 等. RFC 7430[16] 还进一步提出了可以采用 SSL, CGA, TCPcrypt, DNSSec 等强安全机制,但可能会引入额外的交互,计算开销和系统维护开销的增加也不容忽视.

除了以上在 RFC 6181[15] 中所提及的攻击之外,RFC 7430[16] 还对这些攻击和建议解决方案做了进一步的细化(包括初始握手过程的窃听、SYN/JOIN 消息的中断和篡改攻击、ADD_ADDR 攻击等). 此外,还针对信令交互过程的不合规操作以及可能产生的安全攻击做了阐述,同时也给出了可行的方案. 这些攻击包括基于 MP_JOIN 的 DoS 攻击、SYN Flooding 攻击放大. 其中,基于 MP_JOIN 的 DoS 攻击类似于 TCP 中的 half-open 攻击,但危害性更大,可以实现分布式的攻击. 如图 9 所示的交互过程,当需要增加一个新的 TCP 子流时,通过包含 MP_JOIN 选项的 3 次握手来完成. 在第 1 个 SYN+MP_JOIN 的消息中,包含 32 b 的 token 和一个用于 MAC 计算的现时值,由于需要对第 3 个消息发送过来的 MAC 进行验证,而不再携带 token 和现时值,那么主机 B 就必须在接收到第 1 个消息后维护相应的信息. 作为攻击者就可以假冒主机 A 发送大量包含不同五元组的 SYN+MP_JOIN 消息,从而可以耗费主机 B 的资源. 针对 TCP 的 SYN Flooding 攻击指的是攻击者短时间内发送大量的 SYN 请求,而服务器端需要维护相应的状态,大量的请求则会耗费有限的资源;而在 MPTCP 中,除了需要维护与 TCP 相关的信息之外,还需要维护初始连接的 SYN+MP_CAPABLE 中的密钥信息以及后续子流建立中 SYN+ MP_JOIN 有关的 token 和现时信息,这无疑会使得 SYN Flooding 的攻击能力得到放大. 针对这 2 个安全威胁,RFC 7430 给出的建议方案是使得 MPTCP 的握手过程从有状态变成无状态,由消息重复携带必须的信息以及限制 half-open 的子流的数量.

6.3 MPTCP 对网络安全的影响

MPTCP 提供了同时使用在 2 个节点之间的多个路径的能力,这就使得数据流不再依附于某个单独的 TCP 来进行传输,而是可以分布在构成 MPTCP 连接的多个子流上,实现数据流的分裂且提供连接的弹性. 由于子流可以随时添加和删除,MPTCP 的连接与实际的地址之间是可以实现解耦的,但由于 MPTCP 连接的不变性,基于 token 机制,新添加的地址和路径依然可以关联与特定的 MPTCP 连接,这势必会给现有的网络安全带来影响.

文献[81]总结了 MPTCP 在现有网络中的安全机制所带来的破坏性影响,例如,利用 MTCP 便于实施跨路径的分片攻击、安全检测和过滤系统的绕过(bypass),以及作为移动目标防御(moving target defense)[82] 的实现手段等. 对于数据监测也带来了极大的难度,例如对于防火墙而言,往往属于特定的网络,这就很难中断包含经过不同网络的多个 TCP 子流的 MPTCP 连接.

MPTCP 对于多个 TCP 子流的数据传输往往采用联合拥塞控制,这就给实现跨路径的推理攻击(inference attack)引入了一种有效途径. 文献[83]提出了利用 MPTCP 可以有效推断出非监测路径上的吞吐量. 此外,该文还初步探讨了对 RTT 和发送端的拥塞窗口大小的推断方法. 该文的作者 Shafiq 和 Liu 近期获得了美国国家科学基金会(National Science Foundation, NSF)的资助(2015-2018)[84],用于研究利用 MPTCP 实现旁路攻击的安全隐患和应对措施,势必会取得新的进展,对于 MPTCP 协议下一步的制定也会产生一定的影响.

6.4 安全小结

MPTCP 基于传统 TCP,可能会受到与 TCP 相同的安全威胁,此外,MPTCP 可动态添加或删除子流连接,但 MPTCP 提供基于令牌验证的子流管理机制用于验证子流有效性,其安全保护级别较低,易于导致新的安全威胁.然儿,MPTCP 相关的工作主要还是处于如何发挥其传输性能,已有的针对 MPTCP 安全性的考虑并不充分,研究还比较初步,仅包括 SYN Flooding 等典型攻击的预防.然而随着 MPTCP 受到越来越多的关注,其安全性的考量也逐渐被各研究机构作为一个重要的研究方向.

7 其他方面的研究

7.1 路径选择

MPTCP 的通信双方之间可能有多条可供选择的路径,选择最优的路径子集来传输数据有时候会优于使用所有路径,特别是对于路径不对称,即时延、丢包、带宽、能耗等差异较大.

博弈路径选择算法[85]利用博弈的思想在各种开销之间取平衡,比如接口开启连接开销和路径时延开销.另外,还有一些基于实现的方案执行路径选择策略.MAPS[86]提出在未用于发送数据的路径上发送少量探测包探知吞吐量,一旦该路径吞吐量增大到一定阈值则替换已选用集合吞吐量最小的路径.

文献[87]使用跨层信息作为移动设备路径选择的基本依据,数据接收端使用 MAC 层测量所得的路径信号强度来估计路径质量,从而决定是否启用或中止一条路径.该方法比基于时延或丢包率的方法反应更加灵敏准确.

此外,第 5 节中与能耗相关的工作很多也涉及到路径管理方面的研究,并且该方向日趋成为热点.

7.2 移动性和 QoS

MPTCP 可协同使用多个路径,增加或者删除若干路径而不影响上层应用的连通性,并且能够将数据从拥塞路径转移到其他路径上进行传输.上述特征使得 MPTCP 天然集成了移动特性.

除了前面的能耗研究之外,文献[64]还首次使用具有 Linux-MPTCP 内核版本的移动设备在实际网络环境中进行移动性测试,同时使用移动终端的 WIFI 接口和 3G 接口进行数据传输.当 WIFI 信号被终止后,MPTCP 仍可以使用 3G 接口继续进行数据传输,上层业务并没有发生中断,从而完成了不同接口间数据传输的无缝切换,验证了 MPTCP 的移动切换特性.

文献[63]在此基础上提出更完善的 MPTCP 移动切换架构,即对于非 MPTCP 终端,可以使用 MPTCP 代理作为锚点,与 MPTCP 终端建立连接,该场景同样支持移动性.

MPTCP 与 QoS 结合方案[74,88-89]主要针对视频业务而提出.根据视频业务对时延敏感但是对丢包不敏感的特性,在 MPTCP 中增加部分可靠性策略,即接收端设置时延阈值(400 ms),超过该阈值则接收端直接舍弃未接收到的数据包,这样在容忍的丢包率范围内保证小时延.另外,方案中还将视频业务的 I,P,B 帧排优先级,对于优先级高的 I 帧在吞吐量较高的路径上发送,而 P 帧在次优的路径上发送.

7.3 数据中心中的 MPTCP

随着数据中心内部数据传输量的迅速增大,其传统的分层集中式拓扑将造成核心交换机的数据传输压力剧增.近年来的研究提出了 VL2[90]以及 FatTree[91]等新型拓扑结构来解决这一问题,这些拓扑在数据中心内部任意 2 台主机间都有多个核心交换机以提供全带宽连接.

文献[92-93]提出在上述密集并行网络拓扑结构中使用 MPTCP 替代 TCP 作为数据中心内部的传输协议.不仅利用 MPTCP 多径传输的特性提升了连接的吞吐量,且由于对上层应用屏蔽部分路径的失效,也提升了连接的健壮性.另一方面,路径选择和拥塞控制功能允许 MPTCP 将流量转移到可用空间较大的路径上,平衡了数据中心核心交换机上的流量,相较于 TCP 具有更好的公平性.

然而 MPTCP 在数据中心也存在 TCP incast 的问题,即同一路径上的多个子流均遭遇拥塞而同步减小窗口,使得有效数据量急剧减少的情况.这种情况在一个终端向多个服务器同时请求数据时较为常见.EW-MPTCP (equally-weighted MPTCP)[94]在每个 MPTCP 连接执行自身拥塞控制机制的基础上,对同一终端的不同 MPTCP 连接建立联合拥塞控制机制,为各个连接“加权”,使得这些不同 MPTCP 以高耦合的方式增大拥塞窗口,其连接吞吐量之和相当于单条 TCP 连接,以能够在瓶颈路径上与单条 TCP 流公平竞争链路带宽,而不至于引起路径拥塞,进而避免 TCP incast 问题.

8 总结和展望

MPTCP 是目前网络领域的重要研究方向之一.本文介绍了 MPTCP 的技术细节以及不同方面

的研究进展,这些研究促使 MPTCP 由当初的概念逐步走向实用.但本文的部分研究进展还处于模拟实验阶段,需要进一步的实验,并完善到内核代码当中.此外在一些研究方向上,相应机制还需要进一步加以研究.要使得 MPTCP 完全发挥其潜力,在端到端通信方面比拟于 TCP 在当前的普及程度还有待于进一步的研究.

参 考 文 献

[1] Song Wei, Zhuang Weihua. Performance analysis of probabilistic multipath transmission of video streaming traffic over multi-radio wireless devices [J]. IEEE Trans on Wireless Communications, 2012, 11(4): 1554-1564

[2] Kurant M. Exploiting the path propagation time differences in multipath transmission with FEC [J]. IEEE Journal on Selected Areas in Communications, 2011, 29(5): 1021-1031

[3] Wang Peng, Lan Julong, Chen Shuqiao. Multipath traffic splitting algorithm based on adaptive granularity [J]. Journal on Communications, 2015, 36(1): 211-217 (in Chinese)
(王鹏, 兰巨龙, 陈庶樵. 粒度自适应的多径流量分割算法[J]. 通信学报, 2015, 36(1): 211-217)

[4] Wu Chunming, Wang Baojin, Chen Junhua, et al. A traffic splitting algorithm based on nonius in multi-path [J]. Acta Electronic Sinica, 2010, 38(11): 2550-2554 (in Chinese)
(吴春明, 王保进, 陈均华, 等. 一种基于游标的多径流量分割算法[J]. 电子学报, 2010, 38(11): 2550-2554)

[5] Stewart R. RFC2960 Stream control transmission protocol [S/OL]. Fremont, CA: IETF, 2000 [2015-06-17]. https://tools.ietf.org/html/rfc2960

[6] Amer P, Becke M, Dreibholz T, et al. draft-tuexen-tsvwg-sctp-multipath-11 load sharing for the stream control transmission protocol (SCTP) [S/OL]. Fremont, CA: IETF, 2013 [2015-12-03]. https://tools.ietf.org/html/draft-tuexen-tsvwg-sctp-multipath-11

[7] Iyengar J R, Amer P, Stewart R. Concurrent multipath transfer using SCTP multihoming over independent end-to-end paths [J]. IEEE/ACM Trans on Networking, 2006, 14(5): 951-964

[8] Huitema C. draft-huitema-multi-homed-01 multi-homed TCP [S/OL]. Fremont, CA: IETF, 1995 [2015-06-17]. https://tools.ietf.org/html/draft-huitema-multi-homed-01

[9] Eardley P, Nishida Y. IETF MPTCP WG [OL]. [2015-06-17]. http://datatracker.ietf.org/wg/mptcp

[10] Ford A. RFC 6824 TCP extensions for multipath operation with multiple addresses [S/OL]. Fremont, CA: IETF, 2013 [2015-06-17]. https://tools.ietf.org/html/rfc6824

[11] Ford A, Raiciu C, Handley M, et al. RFC 6182 Architectural guidelines for multipath TCP development [S/OL]. Fremont, CA: IETF, 2011 [2015-06-17]. https://tools.ietf.org/html/rfc6182

[12] Raiciu C, Handly M, Wischik D. RFC 6356, Coupled congestion control for multipath transport protocols [S/OL]. Fremont, CA: IETF, 2011 [2015-06-17]. https://tools.ietf.org/html/rfc6356

[13] Walid A, Peng Qiuyu, Hwang J, et al. draft-walid-mptcp-congestion-control-02 balanced linked adaptation congestion control algorithm for MPTCP [S/OL]. Fremont, CA: IETF, 2015 [2015-12-03]. https://tools.ietf.org/html/draft-walid-mptcp-congestion-control-02

[14] Xu Mingwei, Cao Yu, Dong Enhuan. draft-xu-mptcp-congestion-control-02 delay-based congestion control for MPTCP [S/OL]. Fremont, CA: IETF, 2015 [2015-12-03]. https://tools.ietf.org/html/draft-xu-mptcp-congestion-control-02

[15] Bagnulo M. RFC 6181 threat analysis for TCP extensions for multipath operation with multiple addresses [S/OL]. Fremont, CA: IETF, 2011 [2015-06-17]. https://tools.ietf.org/html/rfc6181

[16] Bagnulo M, Paasch C, Gont F, et al. RFC 7430 Analysis of residual threats and possible fixes for multipath TCP (MPTCP) [S/OL]. Fremont, CA: IETF, 2015 [2015-12-03]. https://tools.ietf.org /html/rfc7430

[17] Scharf M, Ford A. RFC 6897 multipath TCP (MPTCP) application interface considerations [S/OL]. Fremont, CA: IETF, 2013 [2015-06-17]. https://tools.ietf.org/html/rfc6897

[18] Wei X, Lopez E. draft-wei-mptcp-proxy-mechanism-02 MPTCP proxy mechanisms [S/OL]. Fremont, CA: IETF, 2015 [2015-06-17]. https://tools.ietf.org/html/draft-wei-mptcp-proxy-mechanism-02

[19] Handley M, Raiciu C. Multipath TCP implementations [OL]. 2015 [2015-06-17]. http://nrg.cs.ucl.ac.uk/mptcp/implementation.html

[20] Google Inc. MPTCP-NS3 project [OL]. 2015 [2015-06-17]. http://code.google.com/p/mptcp-ns3

[21] Kheirkhah M. Morteza Kheirkhah's homepage on github.com [OL]. 2015 [2015-06-17]. https://github.com/mkheirkhah

[22] Kheirkhah M, Wakeman I, Parisis G. MultiPath-TCP in NS3 [OL]. 2014 [2015-06-17]. http://arxiv.org/pdf/1510.07721.pdf

[23] Barré S, Paasch C, Duchêne F, et al. Linux kernel multipath TCP project [OL]. 2015 [2015-06-17]. http://multipath-tcp.org/pmwiki.php/Main/HomePage

[24] Barré S, Paasch C, Bonaventure O. MultiPath TCP: From theory to practice [C] //Proc of IFIP Networking. Berlin: Springer, 2011: 444-457

[25] Barré S. Implementation and assessment of modern host-based multipath solutions [D]. Louvain-la-Neuve: Université catholique de Louvain, 2011

[26] Raiciu C, Paasch C, Barré S, et al. How hard can it be? Designing and implementing a deployable multipath TCP [C] //Proc of the 9th USENIX Symp of Networked Systems Design and Implementation (NSDI'12). Berkeley, CA: USENIX Association, 2012: No. 29

[27] Paasch C, Ferlin S, Alay O, et al. Experimental evaluation of multipath TCP schedulers [C] //Proc of the 2014 ACM SIGCOMM Workshop on Capacity Sharing Workshop (CSWS 2014). New York: ACM, 2014: 27-32

[28] Apple Inc. MPTCP patch for Apple IOS and MacOS [OL]. 2015 [2015-06-17]. http://www.opensource.apple.com/source/xnu/xnu-2782.1.97/bsd/netinet

[29] Paasch C. Improving multipath TCP [D]. Louvain-la-Neuve: Université catholique de Louvain, 2014

[30] Detal G, Baerts M, Caninck Q D. Bundles for Android [OL]. 2015 [2015-06-17]. http://multipath-tcp.org/pmwiki.php/Users/Android

[31] Raiciu C, Pluntke C, Barré S, et al. Data center networking with multipath TCP [C] //Proc of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks (HotNets IX workshop). New York: ACM, 2010: No.10

[32] Raiciu C, Barré S, Pluntke C, et al. Improving datacenter performance and robustness with multipath TCP [J]. ACM SIGCOMM Computer Communication Review, 2011, 41(4): 266-277

[33] Armitage G, Williams N, Stewart L, et al. Multipath TCP work in CAIA [OL]. 2015 [2015-06-17]. http://caia.swin.edu.au/urp/newtcp/mptcp

[34] Chen Y C, Lim Y, Gibbens R J, et al. A measurement-based study of multipth TCP performance over wireless networks [C] //Proc of the 2013 Internet Measurement Conf (IMC'13). New York: ACM, 2013: 455-468

[35] Mirani F, Boukhatem N, Tran M A. A data-scheduling mechanism for multi-homed mobile terminals with disparate link latencies [C] //Proc of the 72nd IEEE Vehicular Technology Conf (VTC 2010-Fall). Piscataway, NJ: IEEE, 2010: 1-5

[36] Ni Dan, Xue Kaiping, Hong Peilin, et al. Fine-grained forward prediction based dynamic packet scheduling mechanism for multipath TCP in lossy networks [C] //Proc of the 23rd Int Conf on Computer Communications and Networks (ICCCN 2014). Piscataway, NJ: IEEE, 2014: 1-7

[37] Ni Dan, Xue Kaiping, Hong Peilin, et al. OCPS: Offset compensation based packet scheduling mechanism for multipath TCP [C] //Proc of Int Conf on Communications (ICC 2015). Piscataway, NJ: IEEE, 2015: 6187-6192

[38] Xu Changqiao, Liu Tianjiao, Guan Jianfeng, et al. CMT-QA: Quality-aware adaptive concurrent multipath data transfer in heterogeneous wireless networks [J]. IEEE Trans on Mobile Computing, 2013, 12(11): 2193-2205

[39] Kim H, Oh B, Lee J. Improvement of MPTCP performance in heterogeneous network using packet scheduling mechanism [C] //Proc of the 18th Asia-Pacific Conf on Communications (APCC 2012). Piscataway, NJ: IEEE, 2012: 842-847

[40] Yang Fan, Amer P, Ekiz N. A scheduler for multipath TCP [C] //Proc of the 22nd Int Conf on Computer Communications and Networks (ICCCN 2013). Piscataway, NJ: IEEE, 2013: 1-7

[41] Casetti C, Gaiotto W. Westwood SCTP: Load balancing over multipaths using bandwidth-aware source scheduling [C] //Proc of the 60th IEEE Vehicular Technology Conf (VTC 2004-Fall). Piscataway, NJ: IEEE, 2004: 3025-3029

[42] Cloud J, Calmon F, Zeng Weifei, et al. Multi-path TCP with network coding for mobile devices in heterogeneous networks [C] //Proc of the 78th IEEE Vehicular Technology Conf (VTC 2013-Fall). Piscataway, NJ: IEEE, 2013: 1-5

[43] Li Ming, Lukyanenko A, Cui Yong. Network coding based multipath TCP [C] //Proc of the 2012 IEEE Conf on Computer Communications Workshops (INFOCOM WKSHPS). Piscataway, NJ: IEEE, 2012: 25-30

[44] Cui Yong, Wang Lian, Wang Xin, et al. FMTCP: A fountain code-based multipath transmission control protocol [J]. IEEE/ACM Trans on Networking, 2015, 23(2): 465-478

[45] Sharma V, Troy K, Kar K, et al. MPLOT: A transport protocol exploiting multipath diversity using erasure codes [C] //Proc of the 27th Conf on Computer Communications (INFOCOM 2008). Piscataway, NJ: IEEE, 2008: 592-600

[46] Li Ming, Ludreyanenko A, Tarkoma S, et al. Tolerating path heterogeneity in multipath TCP with bounded receive buffers [J]. Computer Networks, 2014, 64(2014): 1-14

[47] Eggert L, Heidemann J, Touch J. Effects of ensemble-TCP [J]. ACM SIGCOMM Computer Communication Review, 2000, 30(1): 15-29

[48] Balakrishnan H, Rahul H, Seshan S. An integrated congestion management architecture for Internet hosts [J]. ACM SIGCOMM Computer Communication Review, 1999, 29(4): 175-187

[49] Singh M, Pradhan P, Francis P. MPAT: Aggregate TCP congestion management as a building block for Internet QoS [C] //Proc of the 12th IEEE Int Conf on Network Protocols (ICNP 2004). Piscataway, NJ: IEEE, 2004: 129-138

[50] Handley M, Padhye J, Floyd S. RFC2861 TCP congestion window validation [S/OL]. Fremont, CA: IETF, 2000 [2015-06-17]. https://tools.ietf.org/html/rfc2861

[51] Honda M, Nishida Y, Eggert L, et al. Multipath congestion control for shared bottleneck [C] //Proc of the 7th Int Workshop on Protocols for Future, Large-Scale and Diverse Network Transports (PFLDNeT Workshop). New York: ACM, 2009: 19-24

[52] Han H, Shakkottai S, Hollot C, et al. Overlay TCP for multi-path routing and congestion control [C/OL] //Proc of IMA Workshop on Measurements and Modeling of the Internet. Montvale, NJ: IMA, 2004 [2015-06-17]. http://www.ifp.illinois.edu/~sshakkot/multipath_v3.pdf

[53] Kelly F, Voice T. Stability of end-to-end algorithms for joint routing and rate control [J]. ACM SIGCOMM Computer Communication Review, 2005, 35(2): 5-12

[54] Wischik D, Raiciu C, Greenhalgh A, et al. Design, implementation and evaluation of congestion control for multipath TCP [C] //Proc of the 8th USENIX Conf on Networked Systems Design and Implementation. Berkeley, CA: USENIX Association, 2011: No. 8

[55] Raiciu C, Handley M, Wischik D. RFC 6356 coupled congestion control for multipath transport protocols [S/OL]. Fremont, CA: IETF, 2000 [2015-06-17]. https://tools.ietf.org/html/rfc6356

[56] Raiciu C, Wischik D, Handley M. Practical congestion control for multipath transport protocols [R/OL]. London, United Kingdom: University College London. 2009 [2015-06-17]. http://nrg.cs.ucl.ac.uk/mptcp/mptcp-techreport.pdf

[57] Khalili R, Gast N, Popovic M, et al. MPTCP is not Pareto-optimal: Performance issues and a possible solution [C] //Proc of the 8th Int Conf on Emerging Networking Experiments and Technologies (CoNEXT 2012). New York: ACM, 2012: 1-12

[58] Brakmo L, O'Malley S W, Peterson L L. TCP vegas: New techniques for congestion detection and avoidance [J]. ACM SIGCOMM Computer Communication Review, 1994, 24(4): 24-35

[59] Cao Yu, Xu Mingwei, Fu Xiaoming. Delay-based congestion control for multipath TCP [C] //Proc of the 20th IEEE Int Conf on Network Protocols (ICNP 2012). Piscataway, NJ: IEEE, 2012: 1-10

[60] Hassayoun S, Iyengar J, Ros D. Dynamic window coupling for multipath congestion control [C] //Proc of the 19th IEEE Int Conf on Network Protocols (ICNP 2011). Piscataway, NJ: IEEE, 2011: 341-352

[61] Peng Qiuyu, Walid A, Low S H. Multipath TCP: Analysis and design [OL]. 2013 [2015-06-17]. http://arxiv.org/pdf/1308.3119.pdf

[62] Nicutar C, Niculescu D, Raiciu C. Using cooperation for low power low latency cellular connectivity [C] //Proc of the 10th ACM Int Conf on Emerging Networking Experiments and Technologies (CoNEXT'14). New York: ACM, 2014: 337-348

[63] Raiciu C, Niculescu D, Bagnulo M, et al. Opportunistic mobility with multipath TCP [C] //Proc of the 6th Int Workshop on MobiArch. New York: ACM, 2011: 7-12

[64] Paasch C, Detal G, Duchene F, et al. Exploring mobile/WiFi handover with multipath TCP [C] //Proc of the 2012 ACM SIGCOMM Workshop on Cellular Networks: Operations, Challenges, and Future Design. New York: ACM, 2012: 31-36

[65] Deng Shuo, Netravali R, Sivaraman A, et al. WiFi, LTE, or both?: Measuring multi-homed wireless Internet performance [C] //Proc of the 2014 Conf on Internet Measurement. New York: ACM, 2014: 181-194

[66] Nika A, Zhu Yibo, Ding Ning, et al. Energy and performance of smartphone radio bundling in outdoor environments [C] //Proc of the 24th Int Conf on World Wide Web. Republic and Canton of Geneva, Switzerland: IW3C2 (International World Wide Web Conferences Steering Committee), 2015: 809-819

[67] Zhang Hong, Xue Kaiping, Hong Peilin, et al. Congestion exposure enabled TCP with network coding for hybrid wired-wireless network [C] //Proc of the 23rd Int Conf on Computer Communication and Networks (ICCCN 2014). Piscataway, NJ: IEEE, 2014: 1-8

[68] Minear L, Zhang E. Impact of energy consumption on multipath TCP enabled mobiles [OL]. 2014 [2015-06-17]. http://arxiv.org/pdf/1412.7912v1.pdf

[69] Lim Y S, Chen Y C, Nahum E M. How green is multipath TCP for mobile devices? [C] //Proc of the 4th Workshop on All Things Cellular: Operations, Applications, & Challenges. New York: ACM, 2014: 3-8

[70] Chen Shengyang, Yuan Zhenhui, Muntean G M. An energy-aware multipath-TCP-based content delivery scheme in heterogeneous wireless networks [C] //Proc of the 2013 IEEE Wireless Communications and Networking Conf (WCNC 2013). Piscataway, NJ: IEEE, 2013: 1291-1296

[71] Chen Shengyang, Yuan Zhenhui, Muntean G M. A traffic burstiness-based offload scheme for energy efficiency deliveries in heterogeneous wireless networks [C] //Proc of the 2013 IEEE Global Communications Conf Workshops (IEEE Globecom Workshops). Piscataway, NJ: IEEE, 2013: 538-543

[72] Le T A, Hong C S. ecMTCP: An energy-aware congestion control algorithm for multipath TCP [J]. IEEE Communications Letters, 2011, 16(2): 275-277

[73] Peng Qiuyu, Chen Minghua, Anwar W, et al. Energy efficient multipath TCP for mobile devices [C] //Proc of the 15th ACM Int Symp on Mobile Ad Hoc Networking and Computing (Mobihoc 2014). New York: ACM, 2014: 257-266

[74] Singh V, Ahsan S, Ott J. MPRTP: Multipath considerations for real-time media [C] //Proc of the 4th ACM Multimedia Systems Conf (MMSys'13). New York: ACM, 2013: 190-210

[75] Pluntke C, Eggert L, Kiukkonen N. Saving mobile device energy with multipath TCP [C] //Proc of the 6th Int Workshop on MobiArch. New York: ACM, 2011: 1-6

[76] Nordmark E, Bagnulo M. RFC5533 Shim6: Level 3 multihoming shim potocol for IPv6 [S/OL]. Fremont, CA: IETF, 2009 [2015-06-17]. https://tools.ietf.org/html/rfc5533

[77] Stewart R. RFC4960 stream control transmission protocol [S/OL]. Fremont, CA: IETF, 2007 [2015-06-17]. https://tools.ietf.org/html/rfc4960

[78] Gundavelli S, Leung K, Devarapalli V, et al. Proxy mobile IPv6 [S/OL]. Fremont, CA: IETF, 2008 [2015-06-17]. https://tools.ietf.org/html/rfc4960

[79] Bagnulo M. draft-bagnulo-mptcp-secure-00 secure MPTCP [S/OL]. Fremont, CA: IETF, 2014 [2015-06-17]. https://tools.ietf.org/html/draft-bagnulo-mptcp-secure-00

[80] Diez J, Bagnulo M, Valera F, et al. Security for multipath TCP: A constructive approach [J]. International Journal of Internet Protocol Technology, 2011, 6(3): 146-155

[81] Pearce C, Zeadally S. Ancillary impacts of multipath transmission control protocol (MPTCP) on current and future network security [J]. IEEE Internet Computing, 2015, 19(5): 58-65

[82] Jajodia S, Ghosh A K, Swarup V, et al. Moving Target Defense: Creating Asymmetric Uncertainty for Cyber Threats [M]. Berlin: Springer, 2011

[83] Shafiq M Z, Le F, Srivatsa M, et al. Cross-path inference attacks on multipath TCP [C] //Proc of the 12th ACM Workshop on Hot Topics in Networks (Hotnets 2013). New York: ACM, 2013: No. 15

[84] Shafiq M Z. Multipath TCP side channel vulnerabilities and defenses [OL]. 2015 [2015-12-03]. http://www.nsf.gov/awardsearch/showAward?AWD_ID=1524329

[85] Nguyen S C, Nguyen T M T, Pujolle G, et al. Strategic evaluation of performance-cost trade-offs in a multipath TCP multihoming context [C] //Proc of the 2012 IEEE Int Conf on Communications (ICC 2012). Piscataway, NJ: IEEE, 2012: 1443-1447

[86] Chen Yu, Wu Xin, Yang Xiaowei. MAPS: Adaptive path selection for multipath transport protocols in the Internet, TR2011-09 [R]. Durham, North Carolina: Department of Computer Science, Duke University, 2011

[87] Lim Y, Chen Y, Nahum E, et al. Cross-layer path management in multi-path transport protocol for mobile devices [C] //Proc of the 2014 IEEE Conf on Computer Communications (INFOCOM 2014). Piscataway, NJ: IEEE, 2014: 1815-1823

[88] Diop C, Dugué G, Chassot C, et al. QoS-aware and autonomic-oriented multi-path TCP extensions for mobile and multimedia applications [J]. International Journal of Pervasive Computing and Communications, 2012, 8(4): 306-328

[89] Dugue G, Chassot C, Exposito E. QoS-aware multipath-TCP extensions for mobile and multimedia applications [C] //Proc of the 9th Int Conf on Advances in Mobile Computing and Multimedia(MoMM'11). New York: ACM, 2011: 139-146

[90] Greenberg A, Hamilton J R, Jain N, et al. VL2: A scalable and flexible data center network [J]. ACM SIGCOMM Computer Communication Review, 2009, 39(4): 51-62

[91] Al-Fares M, Loukissas A, Vahdat A. A scalable, commodity data center network architecture [J]. ACM SIGCOMM Computer Communication Review, 2010, 38(4): 63-74

[92] Raiciu C, Pluntke C, Barre S, et al. Data center networking with multipath TCP [C] //Proc of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks. New York: ACM, 2010: No. 10

[93] Raiciu C, Barre S, Pluntke C, et al. Improving datacenter performance and robustness with multipath TCP [J]. ACM SIGCOMM Computer Communication Review, 2011, 41(4): 266-277

[94] Li Ming, Lukyanenko A, Tarkoma S, et al. MPTCP incast in data center networks [J]. China Communications, 2014, 11(4): 25-37

Xue Kaiping, born in 1980. PhD, associate professor. Senior member of China Computer Federation and IEEE. His main interests include next generation network architecture, network security.

Chen Ke, born in 1992. Master candidate. Her main research interests include multipath TCP performance improvement and mobility management (chenke13@mail.ustc.edu.cn).

Ni Dan, born in 1991. Received her master degree from University of Science and Technology of China in 2015. Her main research interests include network performance optimization (nidan@mail.ustc.edu.cn).

Zhang Hong, born in 1990. Received her master degree from University of Science and Technology of China in 2015. Her main research interests include network performance optimization (zhh006@mail.ustc.edu.cn).

Hong Peilin, born in 1961. Professor. PhD supervisor. Her main research interests include next-generation network architecture, network virtualization, and information security (plhong@ustc.edu.cn).