基于链路容量的多路径拥塞控制算法

王竹[1,2]，袁青云[1,2]，郝凡凡[1,2]，房梁[1]，李凤华[1,2]
（1. 中国科学院信息工程研究所，北京 100093；2.中国科学院大学网络空间安全学院，北京 100049）

摘 要：多路径传输的链路差异性和 TCP 友好性约束等因素导致将现有的 TCP 拥塞控制机制直接用于多路径传输时，会带来带宽分配不公平的问题。针对此问题，提出了一种基于链路容量的多路径拥塞控制算法。所提算法基于反馈调节拥塞的思想，利用 M/M/1 缓存队列模型调控接收端缓存队列大小，对发送端吞吐量进行调节，实现多路径联合拥塞控制。实验结果证明，所提算法可提升多路径传输带宽利用率、多路径拥塞控制算法响应能力，保证多路径传输公平性。

关键词：多路径 TCP；拥塞控制；公平性；带宽利用率；反馈调节
中图分类号：TN 929
文献标识码：A
doi: 10.11959/j.issn.1000−436x.2020106

Multipath congestion control algorithm based on link capacity

WANG Zhu[1,2], YUAN Qingyun[1,2], HAO Fanfan[1,2], FANG Liang[1], LI Fenghua[1,2]
1. Institute of Information Engineering, Chinese Academy of Sciences, Beijing 100093, China
2. School of Cyber Security, University of Chinese Academy of Sciences, Beijing 100049, China

Abstract: Factors such as link differences and TCP friendliness constraints lead to the problem of unfair bandwidth allocation when the TCP congestion control mechanism was applied directly to multi-path transmission. To address the problem, it was proposed that a multipath congestion control algorithm was based on link capacity. The proposed algorithm which was based on the concept of feedback regulation achieved multipath joint congestion control by establishing the M/M/1 cache queue model to adjust the throughput rate of senders. Experimental results show the proposed algorithm can improve the multipath transmission bandwidth utilization and the multipath congestion control algorithm responsiveness, and ensure the fairness of multipath transmission.

Key words: multipath TCP, congestion control, fairness, bandwidth utilization efficiency, feedback regulation

1 引言

随着移动互联网的迅速发展及终端接入模块的微型化，越来越多的终端设备具备（如电子发票终端、多模移动通信终端等）多种网络接入模式（如 Wi-Fi、4G/5G、卫星通信等）。用户可根据不同网络场景下的不同需求，通过不同接入方式使用电子发票、移动通信、视频会议等网络信息服务。然而目前传输控制协议（TCP, transmission control protocol）模式仅可利用其中一种网络接口进行数据传输，这降低了网络资源利用率，无法满足海量电子发票数据、音视频等高效传输要求。因此，为了充

收稿日期：2020–02–03；**修回日期**：2020–04–09
通信作者：房梁，fangliang@iie.ac.cn
基金项目：国家重点研发计划基金资助项目（No.2018YFB0803903）；国家自然科学基金资助项目（No.U1836203, No.61672515）；山东省重点研发计划基金资助项目（No.2019JZZY020127）；中国科学院战略性先导科技专项基金资助项目（No.XDC02040400）
Foundation Items: The National Key Research and Development Program of China (No.2018YFB0803903), The National Natural Science Foundation of China (No.U1836203, No.61672515), The Key Research and Development Program of Shandong Province (No.2019JZZY020127), The Strategic Priority Research Program of the Chinese Academy of Sciences (No.XDC02040400)

分利用网络资源，多路径传输技术应运而生。国际互联网工程任务组（IETF, Internet Engineering Task Force）于 2009 年提出多路径传输协议（MPTCP, multipath TCP）[1]，该协议在继承 TCP 的基础上，利用多接口技术建立多条链路，提升网络带宽利用率，降低服务中断风险。近年来，MPTCP 的研究逐步从传统的Internet拓展到数据中心[2]、云平台[3]、卫星网络[4]等新的应用场景。在工业界，MPTCP 已得到初步应用，Linux 发行版已实现 MPTCP，三星 Galaxy 系列智能手机、iPhone Siri 等产品均支持 MPTCP。理论与工程的相互促进为 MPTCP 的完善注入了源源不断的动力。

与传统 TCP 相同，拥塞控制是 MPTCP 的关键技术之一，对 MPTCP 的性能有着重要影响。因此，需要对 MPTCP 的应用场景进行分析，对 MPTCP 的拥塞控制进行优化，从而实现高效、可靠的多路径传输。设计 MPTCP 拥塞控制算法会面临两方面挑战：在多路径传输环境下，每条子流均单独维护拥塞窗口，导致 MPTCP 多条流会过多地抢占带宽，严重时会导致 TCP 流无法正常工作；在链路质量差异较大的异构网络环境下，多路径传输稳定性变差，带宽利用率下降。

近年来，学者在多路径拥塞控制算法方面提出了多种解决方案，包括基于分组丢失的多路径拥塞控制算法[5-7]、基于时延的多路径拥塞控制算法[8-10]、基于瓶颈公平性的多路径拥塞控制算法[11-13]等。但上述算法存在无法分辨噪声、分组丢失严重、带宽竞争能力弱、共享瓶颈子流集合误判等缺陷。

为解决现有工作的不足，满足海量电子发票数据及相关用户数据传输的高效性，本文设计了适用于反馈调节拥塞的多路径拥塞控制框架，在 BBR（bottleneck bandwidth and round-trip propagation time）算法[14]反馈调节拥塞窗口和发送速率的思想基础上，通过建立 M/M/1 缓存队列模型调节接收端缓存队列对发送端吞吐量进行约束，实现了基于链路容量的多路径拥塞控制（MPLC, multipath congestion control based on link capacity）算法，提升了多路径传输的带宽利用率，保证了多路径传输公平性。本文主要贡献如下。

1) 设计了多路径拥塞控制框架，将拥塞控制状态和 TCP 状态分离，便于多级反馈调节拥塞的多路径拥塞控制算法的设计与实现。

2) 设计了 MPLC 算法，将 BBR 算法中根据链路容量反馈调节发送窗口和发送速率的思想引入多路径拥塞控制，提升了多路径传输带宽利用率。

3) 设计了 M/M/1 缓存队列模型，通过调节接收端缓存队列对发送端吞吐量进行约束，解决了多路径传输占用带宽过高的问题，保证了 MPLC 算法公平性。

4) 在 NS3（Network Simulator Version 3）仿真平台上实现了 MPLC 算法，实验结果证明，MPLC 算法实现了MPTCP流与TCP流竞争带宽资源的公平性，比 BALIA（balanced linked adaptation）算法[15]的带宽利用率提升了 9.5%，比 EWTCP（equally weighted TCP）算法[16]的响应时间减少了 1.25 s。

2 相关工作

随着多路径传输的快速发展，多路径拥塞控制面临的问题引起了研究者的广泛关注。本节主要对现有基于分组丢失、基于时延、基于瓶颈公平性的多路径拥塞控制算法，以及其他方式的多路径拥塞控制算法进行介绍。

基于分组丢失的多路径拥塞控制算法[5-7]根据链路是否发生分组丢失来判断网络拥塞状况。Wischik 等[5]提出 semi-coupled 拥塞控制算法，基本思想是通过多路径总拥塞窗口和侵略因子调整子流拥塞窗口增长模式，实现负载均衡，但不足是假设子流往返时延（RTT, round-trip time）相同。考虑到子流 RTT 差异对其吞吐量的影响，Raiciu 等[6]提出 LIA（linked increase algorithm），在计算联合拥塞窗口时通过引入权重因子，动态调整子流对带宽侵占能力，但没有对权重因子做出限制。当权重因子较大时，LIA 会侵占过多的带宽资源。Khalili 等[7]在 LIA 的基础上提出 OLIA（opportunistic linked increase algorithm），从最优化资源池和均衡拥塞响应能力角度出发，为优质路径提供更大的拥塞窗口变化加速度，从而使资源得到更充分的利用。该算法对网络状态变化反应较快，易于侵占过多带宽资源，TCP 友好性较差。

基于时延的多路径拥塞控制算法根据链路中数据分组传输的 RTT 变化来判断网络拥塞状态。Cao 等[8]将基于时延变化估计网络中缓存的数据分组数量来调节拥塞窗口的思想引入 MPTCP 中，实现了较好的拥塞均衡，但该算法与其他算法共存时带宽竞争能力较弱。Gonzalez 等[9]通过引入加性增

长乘性减少策略，设计了混合时延的多路径拥塞控制算法，增强了带宽竞争能力，但带宽利用率低于其他多路径拥塞控制算法。Li 等[10]在综合考虑时延和吞吐量对拥塞窗口影响的基础上，设计了最小化子流时延差异和基于遗传算法的速率分配方案，提升了吞吐量，但与其他拥塞控制算法相比，难以保证算法公平性。

基于瓶颈公平性的多路径拥塞控制算法通过瓶颈检测来区分共享瓶颈子流集合与非共享瓶颈子流集合，再对共享瓶颈子流集合与非共享瓶颈子流集合采取不同的拥塞控制策略。Hassayoun 等[11]提出基于瓶颈公平性的动态窗口耦合算法，该算法通过检测子流分组丢失相关程度，划分共享瓶颈带宽子流集合和非共享瓶颈带宽子流集合，使共享瓶颈子流集合保持 TCP 友好性，使非共享瓶颈子流集合可以独立地与 TCP 流竞争带宽资源，从而达到提升吞吐量、保持算法公平性的目的。但其检测子流的方法会对共享瓶颈带宽子流产生误判，从而导致算法性能下降。Ferlin 等[12]提出多路径共享瓶颈检测算法，通过计算单向时延的方差、偏度、关键频率这 3 个统计量来判断共享瓶颈带宽子流，但由于 3 个统计量的门限值通过经验值获取，因此导致应用场景受限。Zhang 等[13]提出基于时延趋势线性回归方法检测共享瓶颈子流集合的方案，可以有效地预测共享瓶颈带宽子流。但当网络环境差异较大时，该方案可能会带来误判。

除了上述多路径拥塞控制算法外，Xue 等[17]将网络编码引入多路径拥塞控制中，设计了“couple+”的拥塞控制方案，使编码子流可以及时发现拥塞，进行负载均衡，但实际应用中需要考虑编解码器的设计问题。Trinh 等[18]针对无线多媒体传感器网络提出节能型的多路径拥塞控制算法，基本思想是采用低能耗的路径传输数据，同时保证良好的服务质量级别，综合考虑节能和路径特征来调节拥塞窗口，以兼顾节能与服务质量，但需考虑应用场景，这给实际应用带来不便。Xu 等[19]将深度学习算法引入网络研究中，保证了 TCP 友好性，提升了高动态网络的稳健性，但学习规则并未全面考虑实际网络的复杂性。Sun 等[20]通过使用 SDN/NFV（software defined network / network function virtualization）上的段路由转发 MPTCP 子流，为终端用户提供优化的端到端 QoE（quality of experience），但流量分配需要大量的转发机制，这导致存储资源消耗过多，成熟的应用有待方案进一步优化。

在上述的拥塞控制算法中，基于分组丢失的拥塞控制算法无法区分拥塞分组丢失和噪声分组丢失，造成算法性能下降；基于时延的拥塞控制算法与其他拥塞控制算法并存时，往往带宽竞争能力较弱；基于瓶颈公平性的拥塞控制算法在检测共享瓶颈带宽子流时，由于网络环境的复杂往往会带来误判；其他方式的拥塞控制算法通常需要修改现有协议，或针对特殊网络环境进行设计。因此拥塞控制算法广泛运用还有待进一步发展。

3　预备知识

3.1　MPTCP 概述

MPTCP 是 TCP 的扩展[21]。支持 MPTCP 的多模终端设备可以利用多个网络接口建立多条网络链路，提升吞吐量和网络连接弹性。多路径传输场景如图 1 所示，支持 MPTCP 的终端设备（如智能手机、电脑等）利用 3G/4G、Wi-Fi、IP 等接入方式与应用服务器间建立多条网络链路，并选择其中一条或多条链路进行数据收发。

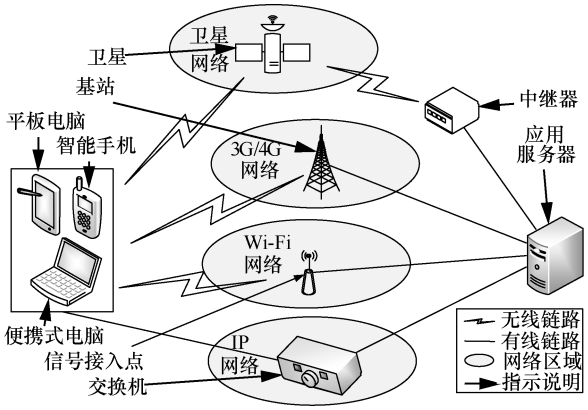

图 1　多路径传输场景

与 TCP 相比，MPTCP 在实现 TCP 确认应答、校验和、拥塞控制等功能的基础上，增加了路径管理、数据分组调度、子流接口等功能。在拥塞控制方面，TCP 中的经典拥塞控制算法会经历慢启动、拥塞避免、快速重传、快速恢复等过程，若直接用于 MPTCP 中会导致多路径流侵占带宽过多。因此，为实现多路径联合拥塞，多路径拥塞控制算法通常会对拥塞避免阶段的增窗策略进行调整，其目标[6]如下。

目标 1　MPTCP 连接的总吞吐量不应该小于其最好路径上的单 TCP 连接的吞吐量。

目标 2　与仅使用任一单路径 TCP 流相比，多

路径流不应该对其共享的任何资源占用过多。

目标 3 在满足目标 1 和目标 2 的前提下，MPTCP 需将最拥塞路径上传输的数据转移到其他路径。

3.2 BBR 算法分析

BBR 算法[14]通过测量链路最大可用带宽 $B_{\max}$ 和最小往返时延 $T_{\min}$，计算当前传输速率 PacingRate 和拥塞窗口 Cwnd，以调节数据发送量，从而形成反馈调节机制。当算法达到稳定状态时，系统将工作在拥塞控制最优点[22]，链路中传输的数据量 Inflight（发送端已经发送，但未确认的数据）为 $B_{\max}$ 与 $T_{\min}$ 的乘积，即带宽时延积（BDP, bandwidth-delay product）。相比于 TCP 中其他拥塞控制算法，BBR 算法减少了缓冲区排队数据分组，降低了传输时延，提升了带宽利用率。

BBR 算法的主要调控参数 PacingRate 和 Cwnd 的计算式分别为

$$\text{PacingRate}=\text{PacingGain}\cdot B_{\max}T_{\min} \tag{1}$$

$$\text{Cwnd}=\text{CwndGain}\cdot B_{\max}T_{\min} \tag{2}$$

其中，PacingGain 表示发送速率增益因子，用于调节算法不同运行状态下的发送速率；CwndGain 表示拥塞窗口增益因子，用于调节算法不同运行状态下的发送窗口。

BBR 算法的运行状态转移如图 2 所示，包括 4 种运行状态[14]：Startup、Drain、ProbeBW 和 ProbeRTT。通过 4 种运行状态可以准确测量出 $B_{\max}$ 和 $T_{\min}$，运行状态的工作过程具体如下。

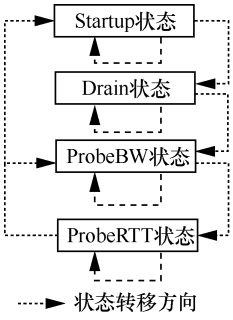

图 2 BBR 算法运行状态转移过程

1) Startup 状态。采用慢启动方式抢占带宽，在检测到带宽被占满 3 次后进入 Drain 状态。

2) Drain 状态。将 Startup 状态占用的接收端缓冲区排空，之后进入 ProbeBW 状态。

3) PrbeBW 状态。探测时长为 10 s，根据发送速率增益数组 $[m,n,k,k,k,k,k,k]$ 周期性改变 PacingGain（其中 $m>1$，默认值为 $\frac{5}{4}$；$n<1$，默认值为 $\frac{3}{4}$；k 默认值为 1。PacingGain = m 表示增加数据发送量，PaingGain = n 表示减小数据发送量，PaingGain = k 表示平稳发送数据），以测量 $B_{\max}$，之后进入 ProbeRTT 状态。

4) ProbeRTT 状态。探测时长为 200 ms，窗口减小为 4 个 TCP 分段大小，以探测 $T_{\min}$。若此状态结束时，带宽仍被占满，则进入 Startup 状态，否则返回 ProbeBW 状态。

3.3 研究思路与目标描述

在网络传输系统中，TCP 友好性可表述为“在相同条件下，非 TCP 流相比于 TCP 流不应消耗过多的带宽资源[23]”。共享资源池如图 3 所示，S_1—D_1 的 MPTCP 双流和 S_2—D_2 的 TCP 单流共享 9 Mbit/s 瓶颈带宽，理想公平情况下，MPTCP 流和 TCP 流各占 4.5 Mbit/s 带宽资源。但实际中 MPTCP 有两条路径传输数据，而 TCP 流仅一条路径传输数据，因此 MPTCP 流占 6 Mbit/s 带宽资源，而 TCP 流占 3 Mbit/s 带宽资源，从而导致 MPTCP 流相对于 TCP 流抢占过多的带宽资源。这破坏了 TCP 友好性，会带来网络服务质量变差的问题，严重时会造成网络瘫痪。

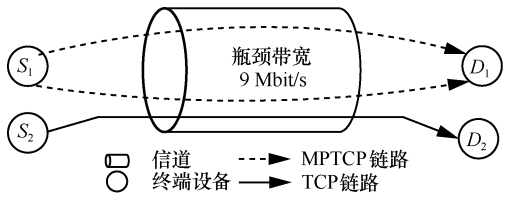

图 3 共享资源池

为充分利用带宽资源，保证多路径传输的友好性和公平性，本文研究思路如下。

1)为提升多路径传输效率，根据 BBR 算法传输时延低、吞吐量高的特性，将 BBR 算法根据链路容量反馈调节发送速率和发送窗口的设计思想引入多路径拥塞控制中，设计基于链路容量的多路径拥塞控制算法，保证多路径传输充分利用带宽资源。

2)为保证多路径传输的友好性和公平性，设计 M/M/1 缓存队列模型，对基于分组丢失的拥塞控制算法通过缓冲区溢出分组丢失判断拥塞和 BBR 算法不占用缓冲区而根据带宽延时积调节拥塞的做法进行折中，通过按一定比例占用接收端缓冲区对发送端吞吐量进行调节，并通过多级反馈模型，动态调节发送端的数据发送量，保证算法公平性。

基于上述研究思路，本文要达到的基本目标为

保证 MPTCP 流与 TCP 流竞争带宽的公平性，实现带宽资源的充分利用，提升动态环境下算法的响应能力。

4　多路径拥塞控制算法设计

4.1　多路径拥塞控制框架

多路径拥塞控制框架主要设计思想是将发送端 TCP 状态与多路径拥塞控制算法状态进行分离，便于反馈调节拥塞的多路径拥塞控制算法的设计。

多路径拥塞控制框架如图 4 所示，主要包括子流监测模块、网络监测模块、拥塞控制模块等。子流监测模块的主要功能是监测子流数据分组收发情况，计算相关通信参数，如往返时延、发送窗口等。网络监测模块的主要功能是维护每条链路的 TCP 状态，监测多路径传输子流的数据发送量。拥塞控制模块的主要功能是利用通信参数调节子流发送窗口和发送速率，利用上一轮子流的数据发送量预估接收缓冲区占有量，进而调节当前子流数据发送量。

在多路径拥塞控制框架中，发送端相当于多模客户端，接收端相当于服务端，发送端与接收端通过 MPTCP 建立多条链路，并利用拥塞控制算法维护多条链路数据收发管理。其中，拥塞控制算法包括两级反馈调节，分别是子流级反馈调节和连接级反馈调节。子流级反馈是指发送端通过接收端反馈的 ACK（acknowledge character）情况，更新子流通信参数，再通过预估链路容量调节子流发送窗口和发送速率。连接级反馈是指发送端通过检测所有子流的数据发送量，预估接收端缓冲区占有量，再根据拥塞控制算法原理调节子流的数据分配量，维持多路径传输的公平性。

4.2　MPLC 算法基本原理

MPLC 算法在 BBR 算法 4 种运行状态的基础上，通过调节每条链路向网络中传输的数据量，实现多路径联合拥塞控制。

MPLC 算法调节链路中传输的数据量基本原理如图 5 和图 6 所示，其中图 5 展示的是传输的数据量与往返时延关系，图 6 展示的是传输的数据量与发送速率关系。其中，Inflight 表示网络中传输的数据量，α 表示 Inflight 的调节因子，BufSize 表示接收端缓存大小，DeliveryRate 表示发送端的数据发送速率。

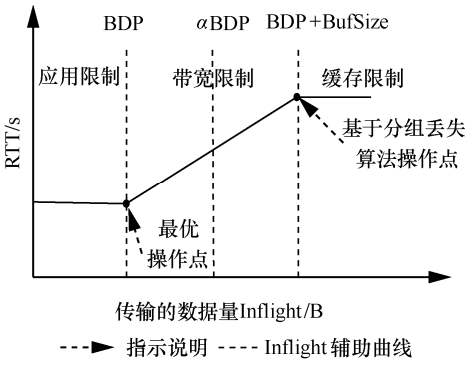

图 5　传输的数据量与往返时延关系

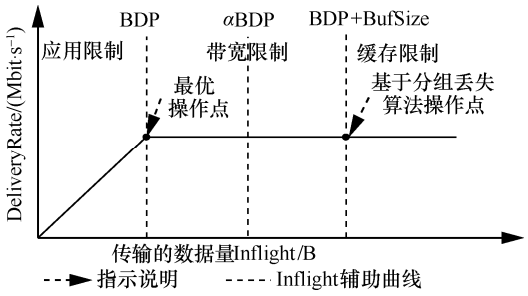

图 6　传输的数据量与发送速率关系

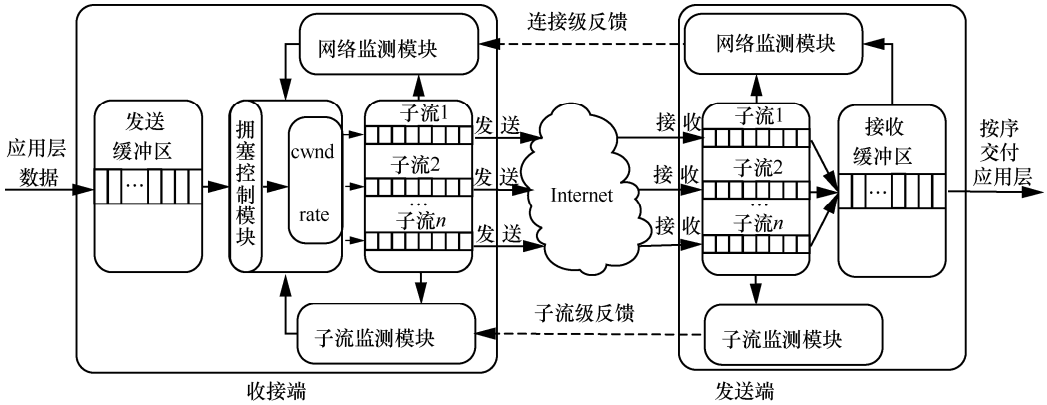

图 4　多路径拥塞控制框架

通过分析图 5 与图 6，可得出以下结论。

1) 当 Inflight ≤ BDP 时，随着 Inflight 增加，DeliveryRate 增大，RTT 不变，此时 RTT 与物理链路传输时延相等。

2) 当 BDP<Inflight ≤ BDP+BufSize 时，随着 Inflight 增加，无法及时处理的数据将以缓存的形式保存于接收端缓冲区，DeliveryRate 受限于网络瓶颈带宽而保持不变，RTT 将随着接收端缓存队列的增加而增大。

3)当 Inflight>BDP+BufSize 时，随着 Inflight 增加，链路中传输的数据量将超过接收缓冲区可承载的数据量上限，导致分组丢失。

MPLC 算法维持 $BDP < Inflight \leqslant BDP + BufSize$，其按照一定比例占用接收端缓冲区（$Inflight = \alpha BDP$）会带来两方面影响。

1) 当发送端向网络中发送的数据量大于 BDP 时，将导致接收端缓冲区排队数据分组增多，数据分组传输时延增大。上述方法虽然约束了多路径传输效率，但有利于维护 MPTCP 流与 TCP 流竞争带宽的公平。

2) 当发送端向网络中发送的数据量小于 BDP+ BufSize 时，不容易造成接收端因缓冲区溢出而发生分组丢失，避免了发送端发送数据过多所导致的分组丢失问题，提升了网络传输效率。

MPLC 算法的 4 个阶段如图 7 所示，具体如下。

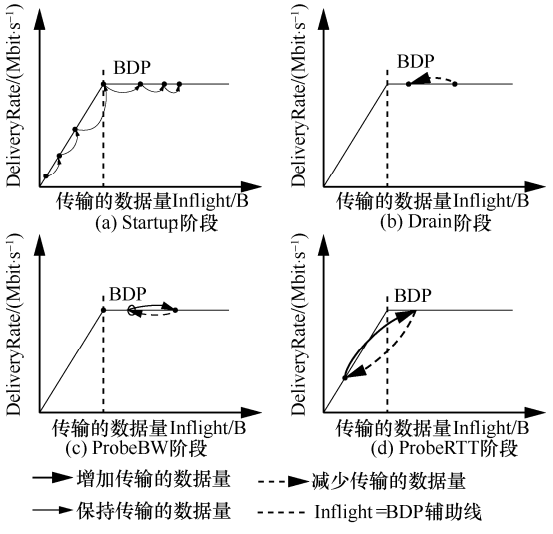

图 7 MPLC 算法 4 个阶段

1) Startup 阶段。采用慢启动方式抢占带宽，在检测到带宽被占满 3 次后进入 Drain 阶段。

2) Drain 阶段。将 Startup 阶段占用接收缓冲区的数据按比例排空，使 Inflight 回退到大于 BDP 的某个位置。

3) ProbeBW 阶段。设置探测时长为固定时间段（默认为 10 s，可根据网络环境调整），并设置发送速率增益数组为 $[1+\beta, 1-\beta, 1, 1, 1, 1, 1, 1]$（$\beta < 1$，表示 PacingGain 调节因子），并根据增益数组周期性调节 PacingRate，以探测最大可用带宽。

4) ProbeRTT 阶段。设定探测时长为固定时间（默认为 200 ms，可根据网络环境调整），将发送窗口减小为 4，一方面是探测最小 RTT 值，另一方面是排空 ProbeBW 阶段在接收缓冲区累积的数据，避免缓冲区溢出而丢失分组。

4.3 基于 M/M/1 缓存队列的传输数据量求解

为了清晰地描述 Inflight 所占用接收缓冲区的规律，本节引用排队论中的基本规律对接收端缓存队列进行建模。

在通信网络中，网络中的数据分组会按照一定规律进入网络中间节点（如路由器、交换机等）。当节点无法及时处理时，数据分组便会进入节点缓冲区进行排队，待数据被处理完后便离开节点缓冲区。其基本过程是数据达到、排队等待、数据处理、离开，满足排队系统的基本属性。

对于网络中的数据流，数据分组到达缓冲区的稳态概率与泊松过程的概率趋势一致[24]，且网络流量可用基于马尔可夫调制泊松分布的参数模型进行精确估算[25]。对于 MPTCP 传输链路，每条子流都满足 M/M/1 的排队系统，其中第一个 M 表示数据分组到达服从泊松分布，第二个 M 表示服务器服务时间服从负指数分布，1 表示只有一个服务器提供服务。下面对基于 M/M/1 缓存队列模型中使用的符号及其含义进行说明，如表 1 所示。

表 1 基于 M/M/1 缓存队列模型中的符号及其含义

符号	含义
λ	数据分组到达节点缓冲区服从参数为 λ 的泊松分布
μ	数据分组离开节点缓冲区服从参数为 μ 的负指数分布
N	缓冲区大小，即可容纳的数据分组个数
P_n	系统稳定时，缓冲区中有 n 个数据分组的概率
t_0	传输时延与处理时延之和
λ_{limit}	系统瓶颈带宽
W	发送端一次向网络中发送的数据量
$B_{\max}$	链路中最大可用带宽

令 n（$n < N$）为 t（$t > 0$）时刻缓存队列中数

据分组个数，在数据分组的到达率为 λ、服务率为 μ 的条件下，接收端缓存队列模型从 n 个数据分组切换到 $n+1$ 个数据分组时，系统状态转移如图 8 所示。

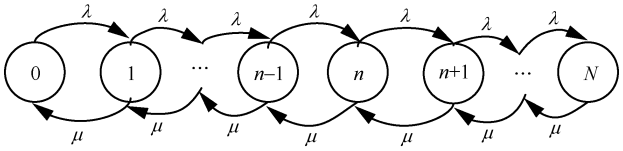

图 8　接收端缓存队列数据分组状态转移

设 P_n 为系统达到稳定时接收端缓存队列中有 n 个数据分组的概率，则根据图 8 可得系统状态平衡方程为

$$\begin{cases}\lambda P_0 = \mu P_1 \\ \lambda P_{n-1} + \mu P_{n+1} = \lambda P_n + \mu P_n\end{cases} \tag{3}$$

根据系统状态平衡方程，在接收端缓存无限大的条件下，数据分组的排队时延 t 可表示为

$$t = \frac{1}{\mu - \lambda} \tag{4}$$

数据分组传输的往返时延等于传输时延、排队时延及处理时延三者之和，其中传输时延和处理时延之和用 t_0 表示，结合式(4)， RTT 可表示为

$$\mathrm{RTT} = f(\lambda) = \frac{1}{\mu - \lambda} + t_0 \tag{5}$$

根据 MPLC 算法，数据分组到达率 λ 即是 MPLC 算法中发送端的 DeliveryRate。结合式(5)可以得到发送速率与往返时延的关系，如图 9 所示。图 9 中曲线 $f(\lambda)$ 与横坐标轴所围成的面积即是发送端一次向网络中发送的数据量 W，其计算方法为

$$W = \int_0^{\lambda} f(\lambda)\mathrm{d}\lambda, 0 < \lambda < \lambda_{\mathrm{limit}} \tag{6}$$

根据 MPLC 算法，链路一次向网络中发送的数据量即是网络中传输的数据量 Inflight。结合式(6)可得，Inflight 计算式为

$$\mathrm{Inflight} = \int_0^{\lambda} f(\lambda)\mathrm{d}\lambda = \alpha\mathrm{BDP} \tag{7}$$

根据 MPLC 算法可知，$\alpha > 1$。另外，根据 MPTCP 体系结构[1]推荐的多路径接收缓冲区大小为 $2(\mathrm{sum}(\mathrm{BW}_i))\mathrm{RTT}_{\max}$，其中 BW_i 表示多路径流第 i 条路径的带宽，$\mathrm{RTT}_{\max}$ 表示多路径流中往返时延最大值。对 $2(\mathrm{sum}(\mathrm{BW}_i))\mathrm{RTT}_{\max}$ 进行不等式放缩可得，$2\mathrm{sum}(\mathrm{BW}_i)\mathrm{RTT}_{\max} \geqslant 2\mathrm{BW}_i\mathrm{RTT}_i = 2\mathrm{BDP}$，其中 RTT_i 表示第 i 条路径的往返时延。当链路中传输的数据量占满瓶颈带宽和接收缓冲区时，将达到 BDP 的 3 倍，因此可得 α 上限最小取值为 3。

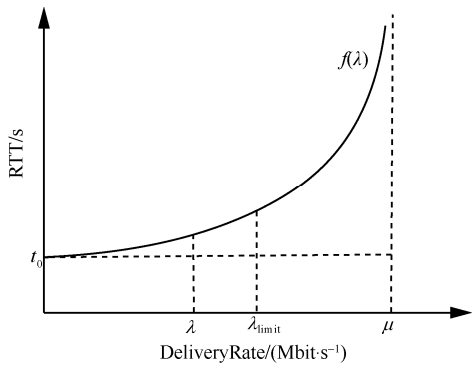

图 9　DeliveryRate 与 RTT 的关系

4.4　MPLC 算法描述

MPLC 算法为调节每条链路向网络中发送的数据量，以实现多路径联合拥塞控制，需对 MPLC 算法的 Drain 阶段排空下限进行设置，对 ProbeBW 阶段发送速率增益数组切换条件进行调整。

MPLC 算法的 Drain 阶段排空下限设置如算法 1 所示。首先获取当前的运行状态、网络中传输的数据量、预估带宽时延积，然后对当前的运行状态进行判断，若处于 Drain 状态，则直到 Inflight 值小于 αBDP 时，才进入 ProbeBW 阶段。

算法 1　Drain 阶段的排空下限设置方法

输入　传输数据量调节因子 α

输出　排空完成情况 drainComplete

```
1) drainComplete ← false
2) state ← 获取算法当前运行状态
3) inflight ← 获取当前传输的数据量
4) BDP ← 获取当前估算的带宽时延积
5) if state == drainState then
6)   if inflight ≤ αBDP then
7)     state ← probeBWState
8)     drainComplete ← true
9)   end if
10) end if
11) return drainComplete
```

MPLC 算法的 ProbeBW 阶段发送速率增益按增益数组周期循环如算法 2 所示。首先获取当前算法运行状态，若为 ProbeBW 阶段，根据发送速率增益值，对增益切换条件进行检测，在满足增益切换条件的情况下，进入下一阶段。其中增益切换条件需满足如下条件之一。

条件 1 PacingGain>1，满足探测时长，且向网络中发送的数据量 Inflight 值超过 $\alpha(1+\beta)$BDP 。

条件 2 PacingGain<1，满足探测时长，或满足向网络中传输的数据量 Inflight 值小于 αBDP 。

条件 3 PacingGain=1，且满足探测时长。

算法 2 ProbeBW 阶段发送速率增益按增益数组循环方法

输入 传输数据量调节因子 α，发送速率增益调节因子 β

输出 进入下一阶段标志 nextState

1)state ← 获取算法当前运行状态
2) nextState ← false
3)if state == probeBWState then
4) minRTT ← 获取最小传输时延
5) timeLen ← 获取探测时长
6) pacingGain ← 获取发送速率增益
7) inflight ← 获取当前传输的数据量
8) BDP ← 获取当前估算的带宽时延积
9) nextState ← timeLen > minRTT ? 1:0
10) if pacingGain > 1 then
11) nextState ← nextState and Inflight ≥ $\alpha(1+\beta)$BDP
12) end if
13) if pacingGain < 1 then
14) nextState ← nextState or Inflight < αBDP
15) end if
16)end if
17)return nextState

4.5 理论分析

4.5.1 公平性分析

对于多路径拥塞控制，每条链路都独立运行相同的拥塞控制算法。在这种情况下，MPTCP 多路径流相比于 TCP 单路径流会抢占过多的带宽资源，因此保持 MPTCP 流对 TCP 流竞争带宽的公平性变得十分重要。此公平性[6]可概述为 MPTCP 流的总吞吐量不应该小于其最好路径上单 TCP 流的吞吐量；且与仅使用任一单路径 TCP 流相比，多路径流不应该对其共享的任何资源占用过多。

根据公平性原则，MPTCP 流获得的吞吐量应等于 TCP 流最优路径获得的吞吐量，因此单位时间内二者传输数据量相等，即

$$\max_{i\in R}\int_0^{\lambda_i} f(\lambda)\mathrm{d}\lambda=\sum_{i\in R}\int_0^{\lambda_i} f(\lambda)\mathrm{d}\lambda \tag{8}$$

其中，R 表示多路径子流集合，λ_i 表示第 i 条子流的数据发送速率。

根据 MPLC 算法可以得到，单位时间内 TCP 单路径传输的数据量极大值如式(9)所示，MPTCP 多路径传输的数据量极大值如式(10)所示。

$$\mathrm{Inflight}_{\mathrm{TCP}}=\mathrm{BDP}_{\mathrm{TCP}}+\mathrm{BufSize} \tag{9}$$

$$\mathrm{Inflight}_{\mathrm{MPTCP}}=\sum_{i\in R}\mathrm{BDP}_i+\mathrm{BufSize} \tag{10}$$

其中，$\mathrm{BDP}_{\mathrm{TCP}}$ 表示 TCP 单路径的带宽时延积，BDP_i 表示多路径第 i 条子流的带宽时延积。对于共享瓶颈的 MPTCP 流与 TCP 流，二者在瓶颈带宽处所占带宽资源大小相等，即 $\mathrm{BDP}_{\mathrm{TCP}}=\sum_{i\in R}\mathrm{BDP}_i$，并且二者接收端缓冲区大小相等。因此，通过对式(9)和式(10)进行推导可以得出式(8)，即 MPLC 算法满足 MPTCP 多路径传输公平性原则。

另外，根据 MPLC 算法在 ProbeBW 阶段遍历速率增益数组 $[1+\beta,1-\beta,1,1,1,1,1,1]$ 调节发送速率的特点，MPTCP 多路径子流发送的数据量受限于预估的瓶颈带宽。对于多路径中第 i 条子流在网络中传输的数据量 $\mathrm{Inflight}^i_{\mathrm{MPTCP}}=\int_0^{\lambda_i} f(\lambda)\mathrm{d}\lambda$ 应满足式(11)所示的不等式。

$$\alpha\,\mathrm{BDP}_i\leqslant\int_0^{\lambda_i} f(\lambda)\mathrm{d}\lambda\leqslant\alpha(1+\beta)\mathrm{BDP}_i \tag{11}$$

根据式(11)可知，当 $\mathrm{Inflight}^i_{\mathrm{MPTCP}}$ 超过 $\alpha(1+\beta)\mathrm{BDP}_i$ 时，子流通过降低发送速率，减少数据发送量；当 $\mathrm{Inflight}^i_{\mathrm{MPTCP}}<\alpha\mathrm{BDP}_i$ 时，子流通过提升发送速率，增加数据发送量。通过上述动态调节发送速率的方式，维持 MPTCP 流对 TCP 流的公平性。

4.5.2 抗分组丢失性能分析

根据 MPLC 算法 4 个运行状态，ProbeBW 阶段占整个算法运行周期的绝大部分时间（约占整个周期的 98%）。从吞吐量的角度分析，要使算法的吞吐量不急剧下降，其 ProbeBW 阶段带宽抢占量不应该小于分组丢失量。

在 ProbeBW 阶段，发送速率按照增益数组 $[1+\beta,1-\beta,1,1,1,1,1,1]$ 周期性进行调节，增益为 $1+\beta$ 表示增大数据发送量，增益为 $1-\beta$ 表示减小

数据发送量，增益为 1 表示平稳发送数据。结合式(7)，对于利用 MPTCP 建立多条链路传输数据的网络系统，可以估算出在增益为$1+\beta$时的数据增量为

$$\Delta_1=\alpha(1+\beta)\sum_{i=1}^{n}\mathrm{BDP}_i-\alpha\sum_{i=1}^{n}\mathrm{BDP}_i \tag{12}$$

假设第 i 条路径上的分组丢失率为 P_i，在增益为$1+\beta$时，分组丢失数据量表示为

$$\Delta_2=\alpha(1+\beta)\sum_{i=1}^{n}(P_i\mathrm{BDP}_i) \tag{13}$$

由于 $\Delta_1>\Delta_2$，结合式(12)和式(13)，因此分组丢失率应满足式(14)所示的不等式。

$$\sum_{i=1}^{n}(P_i\mathrm{BDP}_i)<\left(1-\frac{1}{1+\beta}\right)\sum_{i=1}^{n}\mathrm{BDP}_i \tag{14}$$

另外，根据 MPLC 算法连接级反馈特点，多路径传输的数据量 $\mathrm{Inflight}=\sum_{i\in R}\int_0^{\lambda_i}f(\lambda)\mathrm{d}\lambda$ 应该满足式(15)所示的不等式。

$$\sum_{i\in R}\mathrm{BDP}_i\leqslant\mathrm{Inflight}\leqslant\sum_{i\in R}\mathrm{BDP}_i+\mathrm{BufSize} \tag{15}$$

根据式(15)可知，MPLC 算法限制 Inflight 上限小于 $\sum_{i\in R}\mathrm{BDP}_i+\mathrm{BufSize}$，避免了大量数据填充接收端缓冲区，缓解了接收端缓冲区溢出导致的分组丢失现象，因此 MPLC 算法相对于基于分组丢失的拥塞控制算法具有更好的抗分组丢失性能。

4.5.3　系统效率分析

效率是指单位时间完成的工作量。在 MPTCP 多路径传输条件下，系统效率可表示为单位时间传输的数据量。定义多路径传输系统效率为 E，则系统效率计算式为

$$E=\frac{W}{\mathrm{RTT}} \tag{16}$$

对于多路径传输系统，设子流往返时延向量可表示为$(\mathrm{RTT}_1,\mathrm{RTT}_2,\cdots,\mathrm{RTT}_n)$，子流的数据发送速率向量可表示为$(\lambda_1,\lambda_2,\cdots,\lambda_n)$，子流在往返时延内可以传输的数据量向量可表示为$(W_1,W_2,\cdots,W_n)$，子流传输时延和数据处理时延之和的向量可表示为$(t_1,t_2,\cdots,t_n)$。根据系统效率定义，MPTCP 多路径传输系统总效率等于各个子流效率之和，可表示为

$$E=\frac{W_1}{\mathrm{RTT}_1}+\frac{W_2}{\mathrm{RTT}_2}+\cdots+\frac{W_n}{\mathrm{RTT}_n} \tag{17}$$

结合式(5)和式(6)，将子流往返时延表达式和子流一次向网络中发送的数据量表达式代入式(17)，可得

$$E=\sum_{i\in R}\frac{\int_0^{\lambda_i}f(\lambda)\mathrm{d}\lambda}{f(\lambda_i)}=\sum_{i\in R}\frac{\int_0^{\lambda_i}\left(\dfrac{1}{\mu-\lambda}+t_i\right)\mathrm{d}\lambda}{\dfrac{1}{\mu-\lambda_i}+t_i} \tag{18}$$

根据式(18)，对系统效率函数 E 求一阶导数 E' 和二阶导数 E''，可得 $E''<0$，故 E' 为单调递减函数。其中，E 的一阶倒数 E' 表示为

$$E'=\sum_{i\in R}\frac{f^2(\lambda_i)-f'(\lambda_i)\int_0^{\lambda_i}f(\lambda)\mathrm{d}\lambda}{f^2(\lambda_i)} \tag{19}$$

由于 $E'\dfrac{\mu}{2}>0$、$E'(\mu)<0$，故系统函数 E 在 $\lambda_i\in\left(\dfrac{\mu}{2},\mu\right)$ 先增后减，即系统效率函数 E 在 $\lambda_i\in\left(\dfrac{\mu}{2},\mu\right)$ 存在最优解。

5　实验及分析

为验证 MPLC 算法公平性、分组丢失率对吞吐量的影响、动态环境下算法响应能力，本文在 NS3 网络仿真平台上设计并实现了 MPLC 算法[26]。另外，实验测试中的数据调度算法采用 Linux 中默认的轮询调度算法。

5.1　拥塞算法测试场景

为测试 MPLC 算法公平性，采用图 10 所示的共享瓶颈带宽、图 11 所示的非共享瓶颈带宽的实验拓扑[15,27]。在图 10 与图 11 中，客户端、服务端连接路由器的线路表示接入带宽，路由器与路由器连接的线路表示瓶颈带宽。另外，根据图 11 所示的拓扑结构进行了分组丢失率对吞吐量的影响、动态环境下算法响应能力的实验测试。

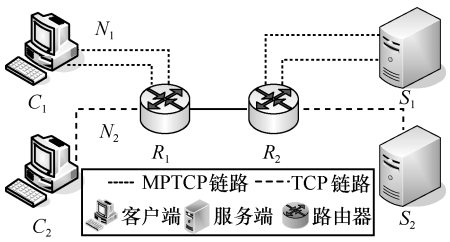

图 10　共享瓶颈带宽的实验结构

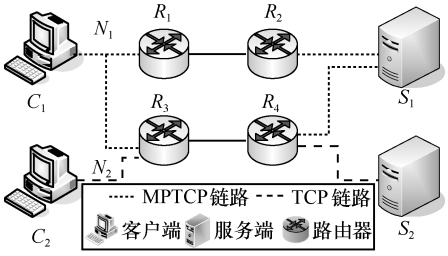

图 11 非共享瓶颈带宽的实验拓扑结构

实验仿真参数如表 2 所示，为了满足 4G/5G 应用高带宽低时延的发展趋势，其中瓶颈带宽为 100 Mbit/s，往返时延为 5 ms；另外，分组丢失率 P、MPTCP 流的对数 N_1、TCP 流的个数 N_2、传输数据量调节因子 α、发送速率增益调节因子 β 为可变参数，其他值为不变参数。

表 2 实验仿真参数

参数	值
段大小/ B	1 500
初始窗口/ B	7×1 500
瓶颈带宽/ (Mbit·s^{-1})	100
接入带宽/ (Mbit·s^{-1})	200
往返时延（RTT）/ ms	5
分组丢失率（P）	0
仿真启动时间/ s	0.01
仿真结束时间/ s	60
MPTCP 流的对数（N_1）	1
TCP 流的个数（N_2）	1
Inflight 调节因子（α）	2
PacingGain 调节因子（β）	0.25

5.2 公平性评估

为了评估不同协议占用网络带宽资源的公平性，利用吞吐量等价比[28]来衡量两种协议之间的公平性。对于协议 a 和协议 b，在时间尺度 δ 下吞吐量等价比函数定义为

$$e_\delta(t)=\min\left(\frac{R_{a,\delta}(t)}{R_{b,\delta}(t)},\frac{R_{b,\delta}(t)}{R_{a,\delta}(t)}\right),R_{a,\delta}>0,R_{b,\delta}>0 \quad (20)$$

其中，$R_{a,\delta}(t)$ 与 $R_{b,\delta}(t)$ 分别表示协议 a 数据流和协议 b 数据流在时间尺寸 δ 内的吞吐量。时间序列 $\{e_\delta(t_0+i\delta)\}_{i=0}^{n}$ 中各元素的均值 e_δ^i 是协议 a 数据流与协议 b 数据流的吞吐量等价比，且 $e_\delta^i\in(0,1]$。e_δ^i 越接近 1，表明两种协议流在时间尺度 δ 下越公平。

在共享瓶颈带宽条件下，仿真测试结果如图 12 所示。在非共享瓶颈带宽条件下，仿真结果如图 13 所示。图 12 和图 13 分别展示了 MPTCP 流吞吐量与 TCP 流吞吐量随时间的变化趋势。

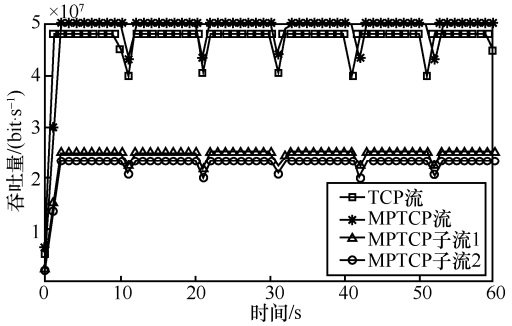

图 12 共享瓶颈带宽条件下仿真结果

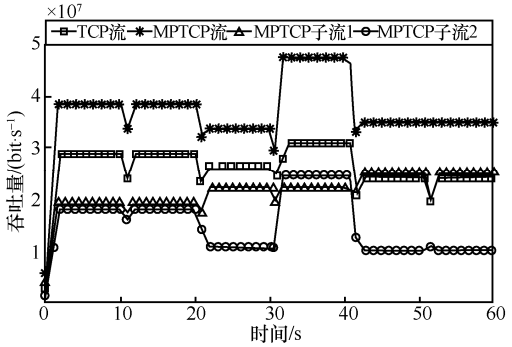

图 13 非共享瓶颈带宽条件下仿真结果

在时间尺度 δ=10 s 条件下，利用式(20)分别计算 10 s、20 s、30 s、40 s、50 s、60 s 时刻的 MPTCP 流和 TCP 流的吞吐量在共享瓶颈带宽和非共享瓶颈带宽条件下的等价比，其结果如图 14 所示。由图 14 可知，在共享瓶颈带宽条件下，MPTCP 流与 TCP 流的吞吐量等价比接近 1，表明 MPTCP 流与 TCP 流竞争带宽资源的公平性，实现了 MPTCP 对 TCP 的友好性。在非共享瓶颈带宽条件下，MPTCP 流与 TCP 流的吞吐量等价比接近 0.8，表明 MPTCP 流比 TCP 流在竞争带宽资源中具有一定优势，表明 MPTCP 具有较好的 TCP 友好性。

5.3 分组丢失率对吞吐量影响评估

将 MPLC 算法与 BALIA 算法在不同分组丢失率下的吞吐量进行对比。通常分组丢失率小于 1%认为网络环境是“好的”，分组丢失率在 1%~2.5%认为是可以接受的[29]。因此设计分组丢失率取值为 0、0.01%、0.1%、1%、5%。实验拓扑如图 11 所示，参数如表 2 所示，其中 $N_1=1$、$N_2=0$，两条路径分组丢失率 P 设置如表 3 所示，其他参数保持不变。

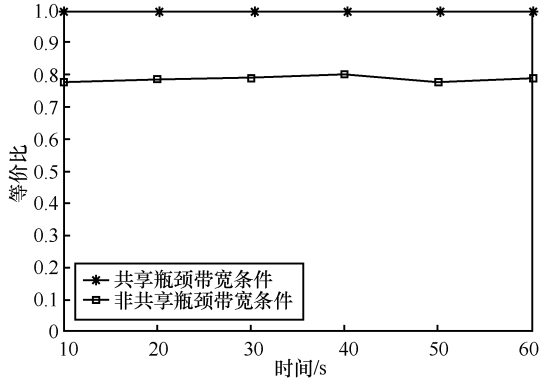

图 14　MPTCP 流与 TCP 流的吞吐量等价比

表 3　两条路径上 MPTCP 的分组丢失率

路径	分组丢失率				
路径 1	0	0.01%	0.1%	1%	5%
路径 2	0	0.01%	0.1%	1%	5%

按照表 3 的分组丢失率参数值，固定其中一条路径分组丢失率，变化另外一条路径分组丢失率，分别测量 MPLC 算法和 BALIA 算法分组丢失率对吞吐量的影响，结果如图 15 所示，图 15(a)表示在路径 1 没有分组丢失（即 P_1=0）的情况下，路径 2 分组丢失率 P_2 按照表 3 变化时，MPLC 算法和 BALIA 算法吞吐量的变化情况。图 15(b)~图 15(e)分别表示路径 1 在其他分组丢失条件下，2 种算法的对比情况。此处仅以图 15(a)为例进行分析。在路径 2 的分组丢失率 $P_2=0$ 的条件下，MPLC 算法吞吐量为 179.4 Mbit/s，带宽占用率为 89.7%。BALIA 算法吞吐量为 160.3 Mbit/s，带宽占用率为 80.2%。相比 BALIA 算法，MPLC 算法带宽利用率提升了 9.5%，并且 MPLC 算法和 BALIA 算法的吞吐量都随着分组丢失率的增加而降低，且在相同分组丢失率情况下，MPLC 算法的吞吐量是 BALIA 算法吞吐量的1.12～11.3倍，故 MPLC 算法分组丢失容忍能力优于 BALIA 算法。

5.4　动态环境下算法响应能力评估

实验拓扑如图 11 所示，参数如表 2 所示，其中 $N_1=1$、$N_2=5$，其他参数保持不变。为了测试动态环境下算法响应能力，在程序运行 25 s 后，撤出 TCP 流，观察 MPTCP 流抢占带宽的变化情况，结果如图 16 所示。由图 16 可知，在 0~25 s 时，MPTCP 流与 TCP 流竞争带宽资源并达到稳定；在 25 s 时，TCP 流撤离，MPTCP 流感知到可用带宽增加，在 27 s 时占满带宽并达到稳定。故 MPLC 算法对带宽变化反应时间大约为 2 s。

对比 MPLC 算法与其他多路径拥塞控制算法对网络带宽变化的响应能力[15]，其对比结果如表 4 所示。在多路径拥塞控制算法 EWTCP[16]、OLIA[7]、BALIA[15]中，EWTCP 算法对带宽变化响应速度最快，为 3.25 s；而 MPLC 算法对带宽变化响应速度

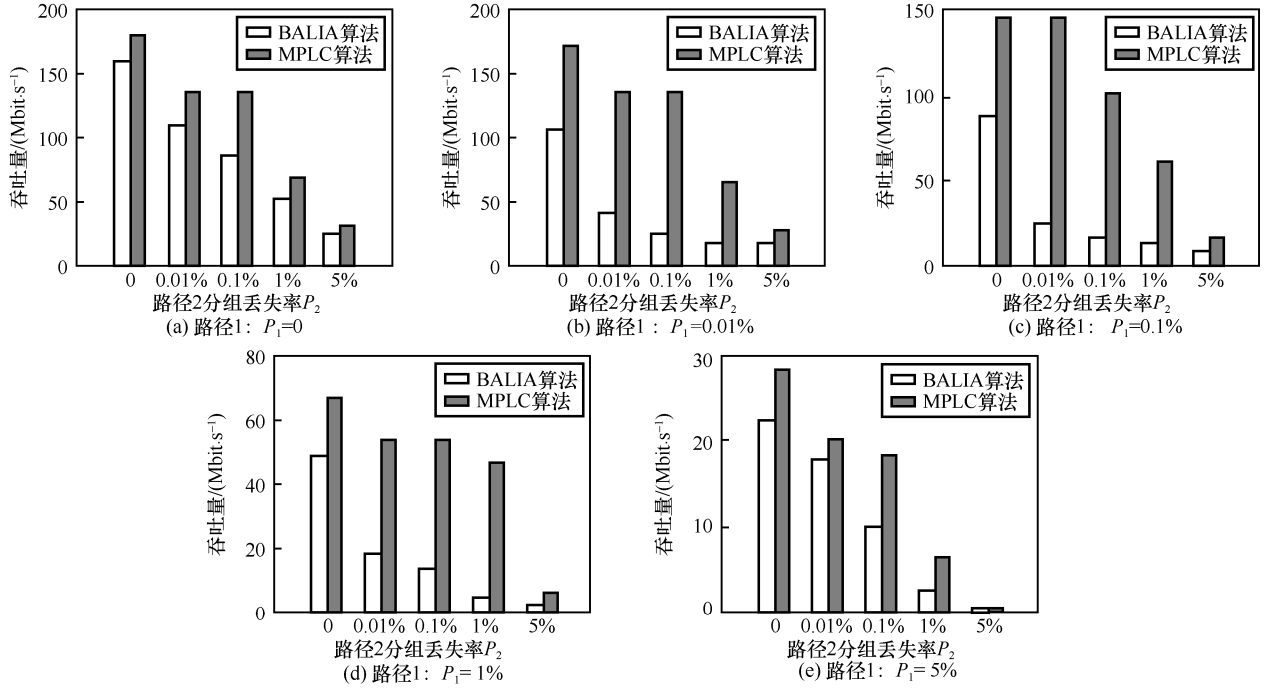

图 15　分组丢失率对 BALIA 算法和 MPLC 算法吞吐量的影响

为 2.0 s，所以 MPLC 算法较 EWTCP 算法响应时间减小了 1.25 s。

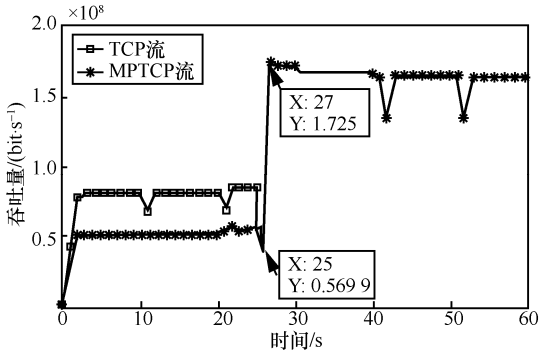

图 16 MPLC 算法对带宽变化的响应能力

表 4 几种多路径拥塞控制算法对带宽变化的响应速度

算法	响应速度/ s
EWTCP	3.25
OLIA	58.50
BALIA	14.73
MPLC	2.00

6 结束语

传统的多路径拥塞控制算法主要依据拥塞窗口、往返时延判断网络拥塞状态，新兴的多路径拥塞控制算法尝试使用人工智能或 SDN 方法辅助多路径拥塞控制，而本文将 TCP 单路径传输中根据链路容量调节发送速率的思想引入多路径传输，提出基于链路容量的多路径拥塞控制算法，为现有的多路径拥塞控制提供了新的实现方案，有效提升了多接入模式环境下电子发票等数据的传输效率。

本文针对多路径拥塞控制带宽资源利用不充分、分配不公平的问题，首先设计了反馈调节拥塞的多路径拥塞控制框架；然后在聚合多路径链路容量基础上，设计 M/M/1 缓存队列模型调控发送端 MPTCP 子流吞吐量；最后根据接收端数据处理情况，利用多级反馈动态调节发送端的数据发送量。通过 NS3 仿真实验结果表明，MPLC 算法保证了公平性，提升了带宽利用率和动态环境下算法响应能力。考虑到未来工作，可以根据 MPLC 算法的反馈调节机制，设计合理的数据调度算法，进一步提升多路径传输性能。

参考文献：

[1] FORD A, RAICIU C, HANDLEY M, et al. Architectural guidelines for multipath TCP development: RFC6182[S]. IETF, (2011-03) [2019-02-03].

[2] YE J, FENG L T, XIE Z Q, et al. Fine-grained congestion control for multipath TCP in data center networks[J]. IEEE Access, 2019:1.

[3] LUCAS C, AHMED A, PRATEEK S, et al. Performance evaluation of multipath TCP for data center and cloud workloads[M]. New York: Association for Computing, 2019: 13-24.

[4] MAI T, YAO H, JING Y, et al. Self-learning congestion control of MPTCP in satellites communications[C]// 2019 15th International Wireless Communications and Mobile Computing Conference. Piscataway: IEEE Press, 2019: 775-780.

[5] WISCHIK D, RAICIU C, GREEHAKGH A, et al. Design, implementation and evaluation of congestion control for multipath TCP[C]// Proceedings of the 8th USENIX Conference on Networked Systems Design and Implementation. Berkeley: USENIX Association, 2011: 8-22.

[6] RAICIU C, HANDLEY M, WISCHIK D. Coupled congestion control for multipath transport protocols: RFC6356[S]. IETF, (2011-10) [2020-02-04].

[7] KHALILI R, GAST N G, 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.

[8] CAO Y, XU M, FU X. Delay-based congestion control for multipath TCP[C]// 2012 20th IEEE International Conference on Network Protocols (ICNP). Piscataway: IEEE Press, 2013:1-10.

[9] GONZALEZ R, PRADILLA J, ESTEVE M, et al. Hybrid delay-based congestion control for multipath TCP[C]// 2016 18th Mediterranean Electrotechnical Conference. Piscataway: IEEE Press, 2016:1-6.

[10] LI H, WANG Y, SUN R. Delay-based congestion control for multipath TCP in heterogeneous wireless networks[C]// 2019 IEEE Wireless Communications and Networking Conference Workshop. Piscataway: IEEE Press, 2019: 1-6.

[11] HASSAYOUN S, IYENGAR J, ROS D. Dynamic window coupling for multipath congestion control[C]// Proceedings of the 19th Annual IEEE International Conference on Network Protocols. Piscataway: IEEE Press, 2011: 341-352.

[12] FERLIN S, ALAY O, HAYES D, et al. Revisiting congestion control for multipath TCP with shared bottleneck detection[C]//IEEE INFO COM−The IEEE International Conference on Computer Communications. Piscataway: IEEE Press, 2016: 1-9.

[13] ZHANG S, LEI W, ZHANG W, et al. Shared bottleneck detection based on trend line regression for multipath transmission[EB/OL]. (2018-12-14) [2020-02-04].

[14] YEGANEH S H, HASSAS S, CHENG Y Z, et al. BBR: congestion-based congestion control[J]. Communications of the ACM, 2017, 60(2): 58-66.

[15] WALID A, PENG Q Y, HWANG J, et al. Balanced linked adaptation congestion control algorithm for MPTCP: Internet draft draftwalid-MPTCP-congestion-control-04[S]. IETF, (2016-01-26) [2020-02-03].

[16] HONDA M, NISHIDA Y, EGGERT L, et al. Multipath congestion

control for shared bottleneck[C]// Proceedings of the 7th International Workshop on Protocols for Future, Large-Scale and Diverse network Transports. New York: ACM Press, 2009:19-24.

[17] XUE K P, HAN J P, ZHANG H, et al. Migrating unfairness among subflows in MPTCP with network coding for wired-wireless networks[J]. IEEE Transactions on Vehicular Technology, 2017, 66(1): 798-809.

[18] TRINH B, MURPHY L, MUNTEAN G. An energy-efficient congestion control scheme for MPTCP in wireless multimedia sensor networks[C]// 2019 IEEE 30th Annual International Symposium on Personal, Indoor and Mobile Radio Communications . Piscataway: IEEE Press, 2019: 1-7.

[19] XU Z Y, TANG J, YIN C X, et al. Experience-driven congestion control: when multi-path TCP meets deep reinforcement learning[J]. IEEE Journal on Selected Areas in Communications, 2019, 37(6): 1325-1336.

[20] BARAKABITZE A A, MKWAWA I H, SUN L F, et al. QualitySDN: improving video quality using MPTCP and segment routing in SDN/NFV[C]//4th IEEE Conference on Network Softwarization and Workshops. Piscataway: IEEE Press, 2018:182-186.

[21] HANDLEY M, BONAVENTURE O, RAICIU C, et al. TCP extensions for multipath operation with multiple addresses：RFC6824[S]. IETF, (2013-11)[2020-02-04].

[22] KLEINROCK L.Power and deterministic rules of thumb for probabilistic problems in computer communications[C]// Proceedings of the International Conference On Communications. Piscataway: IEEE Press, 1979: 1.

[23] FLOYD S, FALL K. Promoting the use of end-to-end congestion control in the Internet[J]. IEEE/ACM Transactions on Networking 1999(7): 458-472.

[24] CAO J, RAMANAN K. A poisson limit for buffer overflow probabilities[C]// Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. Piscataway: IEEE Press, 2002:994-1003.

[25] SALVAODR P, VALADAS R, PACHECO A. multiscale fitting procedure using Markov modulated poisson processes[J]. Telecommunication Systems, 2003, 23(1-2): 123-148.

[26] GitHup. Multipath congestion control algorithm based on link capacity [EB]. (2019-12-28) [2020-02-04].

[27] WEI W J, WANG Y S, XUE K P, et al. Shared bottleneck detection based on congestion interval variance measurement[J]. IEEE Communications Letters, 2018, 22(12):2467-2470.

[28] FLOYD S. Equation-based congestion control for unicast applications[C]//Conference on Applications. New York: ACM Press, 2000(30): 43-56.

[29] NGUYEN V D, RO S. Performance evaluation of MPTCP over shared bottleneck link[J]. The Journal of Korean Institute of Communications and Information Sciences, 2015, 40(1): 70-77.

[作者简介]

王竹（1972– ），女，山西太原人，博士，中国科学院信息工程研究所研究员、在站博士后，主要研究方向为信息安全、人工智能。

袁青云（1994– ），男，湖北孝感人，中国科学院信息工程研究所硕士生，主要研究方向为计算机网络、信号处理。

郝凡凡（1995– ），女，河北石家庄人，中国科学院信息工程研究所硕士生，主要研究方向为安全协议与理论、计算机网络。

房梁（1989– ），男，山西太原人，博士，中国科学院信息工程研究所助理研究员，主要研究方向为信息安全、访问控制。

李凤华（1966– ），男，湖北浠水人，博士，中国科学院信息工程研究所研究员、博士生导师，主要研究方向为网络与系统安全、信息保护、隐私计算。