软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software [doi: 10.13328/j.cnki.jos.007045]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel: +86-10-62562563

网络拥塞控制方法综述*

蒋万春[1], 李昊阳[1], 陈晗瑜[1], 王 洁[1], 王建新[1], 阮 昌[2]

[1](中南大学 计算机学院, 湖南 长沙 410012)

[2](长沙理工大学 计算机与通信工程学院, 湖南 长沙 411014)

通信作者: 阮昌, E-mail: ruanchang@csust.edu.cn

摘 要: 网络拥塞控制方法是决定网络传输性能的关键因素. 近几年, 网络不断普及、网络带宽不断增长、用户对网络性能的需求不断提升, 为拥塞控制算法的设计带来挑战. 为适应不同的网络环境, 近期不少新颖的拥塞控制算法被研究者们提出来, 极大地提升网络的传输性能, 改善用户体验. 综述最新拥塞控制算法设计思想, 将其分为预约调度式、直接测量式、基于机器学习式以及迭代探测式 4 大类, 分别介绍相应的代表性拥塞控制算法, 并进一步对各种拥塞控制思想方法的优缺点进行对比和分析, 最后展望拥塞控制的未来发展方向, 以启发该领域的研究.

关键词: 拥塞控制; 网络环境; 吞吐量; 延时; 数据中心

中图法分类号: TP393

中文引用格式: 蒋万春, 李昊阳, 陈晗瑜, 王洁, 王建新, 阮昌. 网络拥塞控制方法综述. 软件学报. http://www.jos.org.cn/1000-9825/7045.htm

英文引用格式: Jiang WC, Li HY, Chen HY, Wang J, Wang JX, Ruan C. Survey on Network Congestion Control Algorithms. Ruan Jian Xue Bao/Journal of Software (in Chinese). http://www.jos.org.cn/1000-9825/7045.htm

Survey on Network Congestion Control Algorithms

JIANG Wan-Chun[1], LI Hao-Yang[1], CHEN Han-Yu[1], WANG Jie[1], WANG Jian-Xin[1], RUAN Chang[2]

[1](School of Computer Science and Engineering, Central South University, Changsha 410012, China)

[2](School of Computer and Communication Engineering, Changsha University of Science and Technology, Changsha 411014, China)

Abstract: Network congestion control algorithms are the key factor indetermining network transport performance. In recent years, the spreading network, the growing network bandwidth, and the increasing user requirements for network performance have brought challenges to the design of congestion control algorithms. To adapt to different network environments, many novel design ideas of congestion control algorithms have been proposed recently, which have greatly improved the performance of networks and user experience. This study reviews innovative congestion control algorithm design ideas and classifies them into four major categories: reservation scheduling, direct measurement, machine learning-based learning, and iterative detection. It introduces the corresponding representative congestion control algorithms, and further compares and analyzes the advantages and disadvantages of various congestion control ideas and methods. Finally, the study looks forward to future development direction on congestion control to inspire research in this field.

Key words: congestion control; network environment; throughput; delay; data center

目前, 网络拥塞控制的研究已有数十年, 它是决定网络传输性能的关键因素. 网络拥塞控制算法的运行环境近年来发生了诸多变化. 最显著的是快速增长的网络带宽. 例如, 在数据中心中, 网络带宽已经从 1–10 Gb/s 提升至 40–100 Gb/s. 目前 400 Gb/s 以太网标准化工作已经完成, 已经开始出现在少量商业场景中. 800 Gb/s 以太网标准正在制定中. 互联网的带宽同样快速增长. 截至 2021 年 12 月, 3 家基础电信企业的固定互联网宽带接入用户总数达 5.36 亿户, 比上年末净增 5 224 万户[1]. 短视频、在线教育等网络带宽需求很高的应用是驱动网络带宽快速增

* 基金项目: 国家重点研发计划 (2022YFB2901404); 国家自然科学基金 (61972421, 62132007); 华为创新研究计划旗舰项目; 湖南省优秀青年基金 (2022JJ20078); 湖南省科技创新项目 (2023RC3047)
收稿时间: 2023-02-06; 修改时间: 2023-04-11, 2023-10-09; 采用时间: 2023-10-25; jos 在线出版时间: 2024-02-05

长的核心力量. 另一方面, 如何设计拥塞控制算法来改善用户体验也日益得到重视. 衡量用户体验的核心指标——网络延时成为衡量网络拥塞控制算法性能的主要指标之一. 在数据中心网络中, 降低网络排队延时、减少流完成时间是重要的优化目标[2]. 在互联网中, 解决路由器缓存膨胀 (BufferBloat) 问题[3], 即因过量缓存造成的延迟变高以及吞吐量降低问题, 同样是当前拥塞控制算法设计的主要目标之一.

为了在更大范围内适应不同的网络环境、提升用户体验, 近几年涌现出了不少优化网络拥塞控制性能的研究成果. 在数据中心网络领域, 针对其高带宽、低延时的特性, pHost[4]、NDP[5]、Homa[6]等工作从接收端预约带宽的角度探讨了如何降低数据传输的完成时间, 实现延时敏感的网络传输; DCQCN (data center quantized congestion notification)[7]、ExpressPass[8]、HPCC (high precision congestion control)[2]、TIMELY[9], Swift[10]等工作研究了如何构造无损网络来避免数据丢失导致的远程直接内存存取 (remote direct memory access, RDMA) 协议性能下降、以及配合基于优先级的流控机制 (priority-based flow control, PFC)[9]应对网络拥塞、消除数据重传延时; 另外, 利用数据中心网络中丰富的冗余路径, 通过多路径传输提高数据传输效率也是当前的研究热点[11].

在互联网领域, 适应大范围的网络环境变化是拥塞控制算法设计的核心目标. 一方面, BBR (bottleneck bandwidth and round trip time)[12]、Copa[13]等工作改进了传统传输控制协议 (transport control protocol, TCP) 的设计理念, 创新性地提出了优化网络拥塞控制算法的操作点、利用排队延时信号计算目标速率等观点; 另一方面, Remy[14]、PCC (performance-oriented congestion control) 系列算法[15–17]、Indigo[18]、Orca[19]等工作倡导利用机器学习的方法, 训练出适应动态网络环境的拥塞控制算法.

此外, 针对流媒体传输、无线网络等特定网络环境, 也涌现出了 WebRTC[20]、Verus[21]、ABC (accel-brake control)[22]和 PBE-CC[23]等性能更好的专属拥塞控制算法.

鉴于网络拥塞控制领域的蓬勃发展, 本文综述近几年主要的网络拥塞控制研究工作. 表 1 统计了 2015–2023 年网络领域顶级会议 SIGCOMM (special interest group on data communication) 和 NSDI (networked systems design and implementation) 上发表的拥塞控制论文工作及其应用场景、特点与优势等. 从发展趋势来看, 网络拥塞控制方法的研究是持续的热点. 虽然目前已有一些相关综述工作, 但是它们要么只针对数据中心网络等单一网络场景[24–28]、要么因发表过早而未考虑最新的成果[29]. 本文综述最新的拥塞控制研究成果, 分析和探讨了它们的设计思想和方法. 从拥塞控制设计方法的角度将其分为预约调度式、直接测量式、基于机器学习式以及迭代探测式这 4 大类. 本文还进一步对比和分析了这些拥塞控制算法设计思想, 讨论了它们的优缺点及适用场景. 最后总结和展望了该领域的未来研究趋势. 通过该综述, 本文期望能深度展示和对比分析网络拥塞控制的前沿设计理念, 以启发该领域相关的研究人员取得更多的成果.

表 1　2015–2023 年 SIGCOMM 和 NSDI 发表的拥塞控制相关论文

名字	发表年份	发表会议	应用场景	特点	优势
TIMELY[9]	2015	SIGCOMM	数据中心网络	以延时为拥塞信号, 基于延时梯度进行速率调节	达到低延时和高吞吐量
FastPass[30]	2015	SIGCOMM	数据中心网络	基于全局控制器对每个包速率、路径进行细粒度控制	达到接近0的队列延时
PCC[15]	2015	NSDI	数据中心网络、无线网络、因特网	设计效用函数来衡量网络性能, 采用连续的4个监控区间为一组对发送速率进行随机试验测试, 以确定合适速率	易于部署, 利用效用函数实现良好的收敛性和公平性, 且通过调节其效用函数, 可满足不同的传输需求
DCQCN[7]	2015	SIGCOMM	数据中心网络	利用ECN感知拥塞, 接收端根据ECN标记定期生成拥塞通告包通知源端减速	达到低延时, 且算法适合硬件实现
NDP[5]	2017	SIGCOMM	数据中心网络	交换机会对拥塞的数据包进行裁剪, 接收端通过发送推拉包实现率分配	很好地区分乱序包和丢失包, 可与包散射技术一起使用, 达到极低延时
Express-Pass[8]	2017	SIGCOMM	数据中心网络	接收端发送信用包, 交换机根据拥塞情况对信用包速率进行限制	利用交换机已有功能, 实现极低延时

表 1　2015–2023 年 SIGCOMM 和 NSDI 发表的拥塞控制相关论文 (续)

名字	发表年份	发表会议	应用场景	特点	优势
Homa[6]	2018	NSDI	数据中心网络	通过动态优先级机制赋予包优先级, 接收端发送授权包进行速率分配	达到接近硬件水平延时, 在高负载下能保持良好性能
PCC Vivace[16]	2018	NSDI	数据中心网络、无线网络、因特网	效用函数更加考虑了延时, 加入延时梯度	易于部署, 胜过PCC
Copa[13]	2018	NSDI	无线网络、因特网	理论上计算出一个目标速率, 以此目标速率发送时能周期性排空队列	能与非基于延时的算法竞争
HPCC[2]	2019	SIGCOMM	数据中心网络	利用交换机的INT功能携带网络信息, 根据该信息调节速率	达到超低延时和高吞吐量
PCC Proteus[17]	2020	SIGCOMM	无线网络、因特网	针对不同场景, 设计多种模式: 主要模式、清道夫模式、混合模式	更有效地达到了清道夫目标
PBE-CC[23]	2020	SIGCOMM	无线网络	解码蜂窝网的控制信道信息, 基于此来精确调速	避免在无线链路拥塞时出现队列堆积, 公平性好
Orca[19]	2020	SIGCOMM	无线网络、因特网	基于Cubic和基于深度强化学习的双层逻辑控制	能够根据网络环境动态调节拥塞窗口
Aeolus[31]	2020	SIGCOMM	数据中心网络	设计保护机制来解决首RTT内丢包问题, 将数据分为未调度和调度数据两部分	解决首RTT内丢包, 使得短流更快完成
Swift[10]	2020	SIGCOMM	数据中心网络	基于延时设计拥塞控制机制, 缓解网络拥塞和主机拥塞	在高速数据中心网络下缓解主机拥塞
ABC[22]	2020	NSDI	无线网络	利用无线路由器监测网络带宽, 据此计算目标速率	能快速获取链路带宽
ACC[32]	2021	SIGCOMM	数据中心网络	基于深度强化学习对ECN的阈值进行动态调节	避免手工调节ECN阈值的繁琐
Zhuge[33]	2022	SIGCOMM	无线网络	利用接入点路由器处测量网络信息, 并使用带内和带外反馈机制携带信息	加快无线网络信息的感知
Nimbus[34]	2022	SIGCOMM	因特网	探测背景流量是否为弹性, 根据背景流量类型进入不同的数据调节模式	很好地解决端到端拥塞控制算法与背景流之间的共存问题
PowerTCP[35]	2022	NSDI	数据中心网络	提出用功率对拥塞程度进行衡量	在达到性能目标的同时, 能更快地收敛
Sage[36]	2023	SIGCOMM	因特网	学习存在的拥塞控制策略, 基于数据驱动的强化学习算法达到更好的拥塞控制	不需要在实际网络中运行获得大量训练数据
hostCC[37]	2023	SIGCOMM	数据中心网络	基于次RTT粒度拥塞信息分配主机资源, 以集成的拥塞控制算法来分配网络资源	能够获取主机拥塞的精确时间、位置和拥塞原因; 并与现在拥塞控制协议集成
Bolt[38]	2023	NSDI	数据中心网络	基于可编程交换机反馈次RTT的拥塞通知, 能够主动地快速增加速率	最小化反馈延时, 更快地充分利用带宽
Poseidon[39]	2023	NSDI	数据中心网络	利用INT携带拥塞信息, 只对瓶颈跳的拥塞响应	达到快速收敛和最大最小公平性

1　背景与研究现状

1.1　背　景

当进入网络的流量超过了网络的容纳能力之后, 网络便发生了拥塞, 导致网络排队延时增大, 缓存溢出丢包, 甚至因不断丢包重传而出现有效吞吐量急剧下降、网络崩溃. 由于网络资源的稀缺性, 拥塞在网络中总是存在. 但是在不同的网络环境中, 造成拥塞的主要原因不同.

在数据中心网络, 由于交换机缓存浅、流通信模式、网络拓扑结构的不对称性等原因, 拥塞不可避免地在网络中出现. 例如, 在研究较多的内播 (incast) 通信模式下[40], 多个发送端同时向一个接收端发送数据, 很容易造成浅缓存交换机丢包; 在非对称的网络中, 简单的等价多路径 (equal-cost multi-path, ECMP) 负载均衡机制, 可能使大

流路由到相同的路径上, 造成多条流碰撞[41], 从而引起拥塞.

在因特网中, 随着带宽增加, 流开始和结束得更快, 因而流的抖动增加了, 这是网络性能波动的一个原因. 另外, 带宽增加相当于延时带宽积 (bandwidth delay product, BDP) 也增加, 意味着理论上要求更大的路由器缓存, 进而加剧了缓存膨胀问题, 造成传输延迟变高以及吞吐量降低.

在无线网络中, 链路带宽经常随时间发生改变, 如卫星链路速率可随天气的变化发生改变. 在这种场景下, 如果流根据前一时刻的网络状况激进地增加速率, 很容易造成链路拥塞; 同样地, 如果流依据最近时刻的测量信息保守地减少速率, 则可能造成链路利用率不足.

拥塞控制即在网络发生拥塞时对流量进行调节, 以防止网络出现崩溃. 通常拥塞控制的目标是最大化链路吞吐量和最小化排队延时. 然而, 这两个目标存在互相冲突: 最大化链路吞吐量的拥塞控制方法更倾向于填充路由或交换设备缓存, 因而增加了排队延时. 除此以外, 拥塞控制的目标还包括保证公平性——即令多条竞争流最终占用相同的链路带宽, 最小化流完成时间 (flow completion time, FCT)——即令 1 条流从开始发送到停止发送数据所经历的时间最小等. 此外, 由于网络类型、流量类型的不同, 拥塞控制算法也会侧重不同的目标. 例如, 延迟敏感的流更偏重降低排队延时, 而尽力而为 (best effort, BE) 流要求高的链路吞吐量.

拥塞控制领域存在大量优秀的控制算法. 尽管这些算法各不相同, 但是拥塞控制基本模型是一致的, 可以由以下动态非线性方程描述: 假定 U 代表拥塞窗口或者发送速率, t 为计算时刻, 那么拥塞控制算法的动态变化过程可由微分方程表示为:

$$\frac{\mathrm{d}U(t)}{\mathrm{d}t} = f(U(t), q(t), p) \tag{1}$$

其中, q 代表瓶颈队列长度, p 为与拥塞控制算法相关的参数, f 为非线性函数. 这意味着在拥塞控制算法中, 下一时刻的拥塞窗口或发送速率与当前时刻的拥塞窗口或发送速率、瓶颈队列及控制参数呈非线性关系.

1.2 拥塞控制的发展概述

早期 TCP 规范只包含流控机制, 用于防止发送端过快发送数据而淹没接收端的缓存. 直到 1988 年, Jacobson 等人提出了加性增乘性减 (additive increase multiplicative decrease, AIMD) 法则用于减少拥塞发生时的网络负载, 防止拥塞崩溃. 最早的 TCP 拥塞控制版本是 TCP Taheo 协议[42]. 它以包丢失作为拥塞信号, 采用 AIMD 法则进行拥塞窗口的控制. 随后, TCP Reno[43]、TCP NewReno[44]、TCP SACK[45]被提出来, 改进了 TCP Taheo 在快速恢复和快速重传上的性能问题. TCP Vegas[46]作为 TCP 的另外变种版本, 采用延时而非包丢失作为拥塞信号. 所有这些算法均是针对早期的低带宽因特网而提出的.

随着网络带宽的增加, TCP Taheo、TCP Reno 等协议不能适用于高速因特网, 这是由于这些协议均采用线性增长的方式, 每个 RTT 只增加 1 个数据包, 需要很长的时间才能饱和高带宽网络; 此外, 这些方法要维持一个大的平均拥塞窗口, 要求极低的丢包率 (10^{-7} 或更小). 因此, 出现了针对高速因特网的拥塞控制算法, 如 HSTCP (high speed TCP)[47]、STCP (scalable TCP)[48]、Fast TCP[49]、BIC-TCP[50]、Cubic[51]等. 它们在网络有剩余带宽时, 能以比传统 TCP 更快的速度获取这些网络带宽.

这些拥塞控制算法只涉及主机上的调节操作, 均被称为所谓的端到端的拥塞控制技术. 为了达到更好的拥塞控制效果, 研究者考虑利用网络中设备的能力, 提出基于主动队列管理 (active queue management, AQM)[52]方法进行拥塞控制. 例如, TCP RED[53]使路由器以一定概率丢弃新分组; 队列长度大于 $Th_{\max}$ 时, 新分组才会全部丢弃. 另一种机制显示拥塞通知 (explicit congestion notification, ECN)[54]是路由器根据队列情况对 TCP 包头进行标记, 以携带拥塞信息来通知主机调节速率. 此外, 许多研究如 P2I[55]、PID-AQM[56]、H_∞ PID[57]、DMPIC-AQM[58]等基于控制理论的方法设计了各种 AQM 策略.

除了上述拥塞控制算法之外, 还有一些可以缓解拥塞的技术, 其中包括软件定义网络 (software defined network, SDN)[59,60]、多路径传输[61]等. SDN 将网络的控制平面和数据平面分离, 利用中央控制器通过统一的接口对网络进行编程控制, 从而更精细地控制网络中的拥塞. 多条路径传输则可以利用除了拥塞路径意外的其他更好

路径进行快速的数据传输.

另外, 确定性网络、命名数据网络等新型网络体系中也涌现了一些特定场景的拥塞控制技术. 简要而言, 确定性网络在以太网的基础上为多种业务提供端到端确定性服务质量保障. 其中主流的时间敏感性网络 (time sensitive networks, TSN) 的目标是为流量提供了严格的延迟或抖动保障. IEEE TSN 工作组提出了一个门限机制来控制交换机端口的流量传输, 防止拥塞影响 TSN 中的流量. 同时, 一些工作研究通过路径选择来保证 TSN 中流量的延时[62,63]. 命名数据网络 (named data networking, NDN) 是一种新设计的互联网架构, 其核心思想是基于内容名称而不是 IP 进行数据包的路由. NDN 自身的特点使得它的拥塞控制与传统 TCP 有许多不一样的地方. 例如, 针对网络拥塞, ICP (interest control protocol)[64]、CCTCP (content centric TCP)[65]、ECP (explicit control protocol)[66]等工作将速率控制点放在接收端进行, 而 HoBHIS (hop-by-hop interest shaping)[67]、HR-ICP (hop-by-hop and receiver-driven interest control protocol)[68]则进行逐跳的速率控制.

总之, 拥塞控制算法在发展过程中出现了很多方案, 没有一种算法能够很好地适应所有的网络环境. 接下来, 本文将主要综述最新的拥塞控制思想.

1.3 相关综述工作的对比分析

近几年来随着新网络拥塞控制算法的不断出现, 一些相关的综述工作相继发表. 虽然它们都做了很好的工作, 但是仍然存在一些不足. 下面进行简单的介绍和对比分析.

文献 [24] 介绍了高速无损数据中心网络的拥塞控制算法. 它将拥塞控制算法分为反应式拥塞控制和主动拥塞控制两大类. 反应式拥塞控制即拥塞控制算法在网络产生拥塞之后做出反应, 主动式拥塞控制则通过接收端或者调度器主动为发送端流量分配可用带宽. 更进一步, 文献还从拥塞处理速度、公平性、部署复杂性等方面对拥塞控制算法进行了讨论. 但是文献 [24] 局限于数据中心网络内部, 且主要考虑无丢失需求带来的影响.

文献 [25] 调研了基于 RDMA 的高性能技术和数据中心网络的流量控制, 分别介绍了传统 RDMA、传统数据中心网络和 RDMA 支持的数据中心网络中的流量控制方案, 并从速率调节方式、算法部署机制和流量控制粒度等几方面对现有的典型拥塞控制算法进行了对比. 文献 [25] 同样局限于数据中心网络内部, 且主要注重探讨 RDMA 场景带来的影响.

文献 [26] 完全或者部分采用延时信号的拥塞控制算法进行了分析对比, 着重讨论了基于 TCP 模式的拥塞控制算法和现代 AQM, 并分析了 AQM 和延迟信号之间的相互作用. 文献 [26] 不再只关注数据中心网络, 但其局限于基于延时的拥塞控制算法.

文献 [27] 针对数据中心网络传输层协议展开了调研, 将 2010–2015 年的数据中心网络传输协议设计工作分成了 3 类——基于终端主机的拥塞控制、网络仲裁机制、交换机优先级调度, 并对这 3 类工作的优缺点作深入讨论. 此外, 文献 [27] 分析了近年来数据中心网络传输设计的研究趋势: 接收端驱动的主动拥塞控制和 RDMA 传输协议设计. 文献 [27] 专注于数据中心网络领域, 着重讨论如何实现降低延时和 FCT 等目标.

文献 [28] 分析研究了数据中心网络中进行流量控制的最新方案. 该文从拥塞控制和流量工程即流调度和负载均衡两个角度进行了综述. 对于拥塞控制方案, 文献将其分为了基于丢包、基于 ECN、基于往返延迟 (round trip time, RTT) 和基于其他信息的拥塞控制. 对于流调度方案, 文献归纳为截止时间优先、流大小优先和链路价格的调度. 对于负载均衡方案, 文献从简单机制和负载系统两个方向进行了方案的分析比较. 同样地, 该文只注重数据中心网络中的方案.

文献 [29] 主要介绍了数据中心网络中各种降低 FCT 的方案, 并总结了针对突发流量场景的解决方案; 同时, 该文讨论了数据中心网络中的流量模式, 分析了一些现有传输协议无法在实际网络环境中部署的原因. 文献 [29] 主要关注于 2015 年以前的工作, 而本文关注 2015 年以后的工作. 因此, 它和本文互为补充.

与这些工作不同, 本文主要关注多种网络环境中拥塞控制算法的设计思想和方法. 具体对比如表 2 所示.

表 2 相关工作研究范围对比

文献	数据中心网络	因特网	基于延时的拥塞控制	其他拥塞控制相关问题
[24]	√	N/A	√	√
[25]	√	N/A	√	√
[26]	√	√	√	N/A
[27]	√	N/A	N/A	√
[28]	√	N/A	√	√
[29]	√	N/A	N/A	√
本文	√	√	√	√

注: N/A表示“本栏不适用”

2 拥塞控制算法设计思想和方法

拥塞控制算法通过感知网络的拥塞情况, 直接或间接地调节发送端的数据发送速率, 从而防止网络拥塞导致网络性能坍塌, 维持高链路利用率. 本文综述近几年最新的拥塞控制算法设计思想和方法, 将其分为 4 类: 迭代探测式、直接测量式、预约调度式和基于机器学习式, 并分别进行探讨. 本文划分这 4 个大类的依据是根据拥塞控制方法采用什么核心机制来决定出每条流合适的发送速率. 如果合适的发送速率根据拥塞信号进行多次调节后决定, 则拥塞控制方法属于“迭代探测式”; 如果合适的发送速率根据可用带宽、队列等信息直接计算出来, 则属于“直接测量式”; 如果合适的发送速率通过对网络资源进行预约得到, 则属于“预约调度式”; 如果合适的发送速率由机器学习算法来得到的, 则属于“基于机器学习式”.

虽然流量控制机制也可限制网络中的数据量, 但与拥塞控制不同的是流量控制用于防止因发送端过快发送数据, 造成的接收端缓存被淹没和数据包丢失问题. 流量控制是为了预防拥塞, 而拥塞控制用于在拥塞发生后缓解拥塞. 流量控制和拥塞控制相互配合, 可有效提高网络传输性能.

2.1 迭代探测式

传统的拥塞控制算法, 如 TCP 及其各种改进版本, 其核心设计思想是通过迭代探测的方式, 探测可用的网络带宽. 在迭代探测过程中, TCP 采用拥塞窗口自同步的方式, 尽力确保发出去还未被接收的数据包或飞行包 (inflight packets) 大于链路的 BDP, 以最大化吞吐量. 这种自同步的方式, 每收到一个 ACK 包后拥塞窗口最多加一, 确保了发送端的发送速率不超过链路带宽的 2 倍. 为探测可用带宽, TCP 采用 AIMD 的方式迭代调节拥塞窗口, 实现公平和效率的折中. 近年来, 迭代探测式的拥塞控制设计方法有如下更进一步的发展:

(1) 迭代探测的拥塞标志点. TCP 以丢包为拥塞的标志, 观测到丢包时认为网络拥塞. 这种方法的优点是能够获得高吞吐量, 缺点是网络排队延迟较高. 新工作 BBR[12]将拥塞标志点设定为 BDP, 通过调节发送速率保持发出的数据包与 BDP 相等, 以达到既饱和链路, 又不增大排队延时的目标. 另一方面, 基于延时的拥塞控制算法, 如 Copa[13]、Timely[9]、Swift[10]等, 将拥塞标志点设定在基本延时 RTT_{min} 和 RTT_{max} (缓存填满时的往返延时) 之间.

(2) 迭代探测的幅度. 在探测是否达到拥塞标志点的过程中, 传统 AIMD 的方式由于其兼顾公平和效率的优点, 仍然被 Timely[9]、Swift[10]和 DCQCN[7]等算法采纳. 然而, 为了提升响应速度, Timely[9]和 Swift[10]优化了 AIMD 的系数, 使得发送速率的变化和探测的网络拥塞程度成正比; DCQCN[7]则用折半搜索替代了线性的加性增加 (additive increase, AI) 过程、用多个 ACK 包被 ECN 标记的比例替代了固定的倍乘 (multiplicative decrease, MD) 系数, 以便更快地探测并停留在拥塞标志点. 而 BBR[12]放弃 AIMD 的探测方式, 采用乘性增加乘性减少 (multiplicative increase multiplicative decrease, MIMD), 周期性地呈 1.25 倍或者 0.75 倍增大或者减小速率, 探测是否到达拥塞标志点; Copa[13]则采用加性增加加性减少 (additive increase additive decrease, AIAD) 的方式调节拥塞窗口, 使得瓶颈链路队列每 5 个 RTT 排空一次; 同时为 AIAD 方式配以动态变化的系数, 加快响应速度.

2.2 直接测量式

直接测量式方法指直接根据测量得到的可用网络带宽指导源端的发送速率调节. 经典的直接测量式拥塞控制

算法有 XCP (explicit control protocol)[69]、RCP (rate control protocol)[70]等, 它们的核心思想是在瓶颈链路交换机上测量出可用网络带宽, 反馈给源端. 这一方法的优点是无需迭代探测拥塞标志点, 响应速度快. 缺点则是需要修改交换机进行网络带宽的测量, 开销大. 近年来, 直接测量式方法有如下更进一步的发展:

虽然可用网络带宽能被直接测量, 但是在共享瓶颈链路的流数未知的情况下, 拥塞控制算法仍无法一步到位地将发送速率调节到公平共享带宽的值, 结果探测过程不可避免. 针对该问题, TFC (token flow control)[71]尝试在数据中心网络中估算共享瓶颈链路的流数, 配合测量的可用网络带宽指导调速. HPCC[2]、PowerTCP[35]则利用探测到的网络带宽、队列等信息计算出当前发送速率下网络的利用情况, 配合 MIMD 优先令聚合速率迅速收敛到理想值, 之后再依靠 MIMD+AI 逐渐令每条流的速率收敛到公平共享带宽.

PBE-CC[23]则在接收端通过解码控制信道获取小区蜂窝网基站收集到的网络链路总容量和可用容量, 以及活跃用户数量的信息, 直接计算出公平分配带宽的速率, 并将信息传送回发送端进行调速. 但是这种调速方法只适用于瓶颈发生在蜂窝网中的场景. 当瓶颈发生在有线网络中时, PBE-CC[23]采用类似 BBR[12]的迭代探测式调速方法.

2.3 预约调度式

最近提出的预约调度式拥塞控制算法在数据中心网络中能达到很好的传输性能. 预约调度式算法利用集中或分布式的方式对网络资源进行预约, 然后根据预约的结果以及流调度算法对竞争流分配相应的速率. 这种网络资源的分配方式可以避免传统的拥塞控制算法因盲目地发送数据造成的拥塞, 且通常能够达到接近零的瓶颈队列, 从而大大降低数据传输的延时. 同时, 流调度算法可以显著减低流的平均完成时间.

预约调度式的拥塞控制算法设计有两大关键: 一是网络资源的预约机制; 二是流调度的方法. 网络资源包括网络带宽、发送时隙等. 例如, 可以对流的发送时机进行预约, 只允许特定时间发送某条流; 也可以预约带宽或速率, 只允许流按授予的带宽或速率大小发送数据. 流调度的方法可以从集中式和分布式两种角度设计. 在集中式调度方法中, 需要一个全局的调度器计算所有流的调度结果. 通常, 网络中同时存在很多流, 因此调度器需要具有很高的运算性能以便短时间内计算出调度结果. 分布式的调度器不用具有高运算性能, 但是需要协调发送端分布式地发送数据. 这类设计方法最近的研究进展如下.

(1) 降低网络资源预约时的带宽浪费. 网络资源预约最初需要花费至少 1 个 RTT 的预约时间, 在这段时间内不发送数据, 导致带宽被浪费. 例如, FastPass[30]、pHost[4]、ExpressPass[8]等需要首先花费 1 个 RTT 发送信息进行网络资源的预约, 再进行数据传输. 而更新的方案如 NDP[5]、Homa[6]、Aeolus[31]同时进行资源预约和数据传输, 避免因资源预约而浪费带宽, 可进一步加快数据的传输.

(2) 分布式流调度方式. FastPass[30]等对流采用集中式调度. 这类调度器需要在短时间内对网络中所有流决定它的发送时间、发送速率. 当网络规模变大或者网络带宽很高时, 这种集中调度方式会越来越影响流的快速传输; 而新提出的方法如 NDP[5]、Homa[6]等利用接收端进行分布式的调度. 每个接收端只负责本地收到的流的调度, 即整个网络中的一部分流. 多个接收端通过协调, 如统一的优先级设定规则等, 完成对所有流的调度. 这种流调度方式, 具有更好的扩展性, 更利于部署在大规模网络中.

2.4 基于机器学习式

传统的拥塞控制算法往往存在对网络的假设, 或者需要手动调节参数使算法更加适应网络环境. 比如 TCP Cubic 假设丢包是拥塞导致, 在随机丢包环境下算法的吞吐率低, 浪费网络带宽; BBR 存在多个参数需要设置. 为使 BBR 适用于无线网络, 谷歌公司的工程师们花费了几年的时间调节模型的参数[19]. 基于机器学习式方案通过机器学习的方法训练调速模型, 指导发送速率调节. 这种方式将网络环境视为黑盒子, 不需要对网络进行假设和精确建模, 并且智能地学习网络环境对应的发送速率, 避免了手动调参, 具备良好的自适应性. Remy[14]首次将人工智能技术应用于拥塞控制, 它采用目标函数衡量调速规则的优劣程度, 通过离线训练直接生成从网络状态到调速规则的映射模型. 继 Remy 之后, 基于在线优化的 PCC 系列[15–17]、基于深度强化学习的 Orca[19]、Aurora[72]陆续被提出来. 在实际应用中, 如果应用场景包含于训练环境之内, 算法能取得较好的效果; 相反, 则有可能收敛到错误点甚至不收敛, 导致其性能坍塌. 近年来, 基于机器学习式拥塞控制设计方法有如下更进一步的发展.

(1) 机器学习方法和传统拥塞控制技术结合. 与传统的 TCP 算法和其他非机器学习式方案相比, 机器学习技术通常涉及大量数据集的训练和学习过程, 导致算法收敛慢、产生较高的中央处理器 (central processing unit, CPU) 运行开销, 造成算法的实用性差. 针对该问题, Orca[19]、Eagle[73]、Libra[74]等方案提出将经典拥塞控制和深度强化学习技术结合起来, 使经典拥塞控制的实用性与人工智能技术的自适应性互相补充.

(2) 防止收敛到错误点或不收敛. 当基于机器学习式方案采用离线训练模式时, 如果应用场景未经过预先训练, 这类方案有可能无法正确收敛到预期的目标点, 具体表现为收敛到一个错误的发送速率、导致算法的性能下降, 抑或是不断震荡、导致算法的性能不稳定. 针对这一问题, 有两种解决方案. 一种是采用在线学习的模式, 规避算法性能与训练场景的依赖问题, 但是这种方案往往面临收敛缓慢、运行开销大的问题; 另一种解决方案是主动为收敛过程添加干扰, 比如 Orca 的底层 Cubic 的持续探测为上层强化学习的收敛添加干扰信息, 避免了强化学习的调节稳定地停留在一个错误速率.

3 拥塞控制算法分类简介

本节将基于上述分类方式介绍最近提出的拥塞控制算法, 并分 4 类讨论, 包括迭代探测[7,9,10,12,13,22,33,34]、直接测量[2,23,35,38,39,71]、预约调度[4–6,8,30,31]和基于机器学习[16–19,32,36,75,76]. 迭代探测类算法被分为了端到端控制和交换机辅助 2 小类; 预约调度类算法按调度方式被分为了集中式调度、分布式端到端调度和分布式跳到跳调度 3 小类; 基于机器学习类算法被分为了在线学习、离线学习和离线在线混合学习. 具体如图 1 所示.

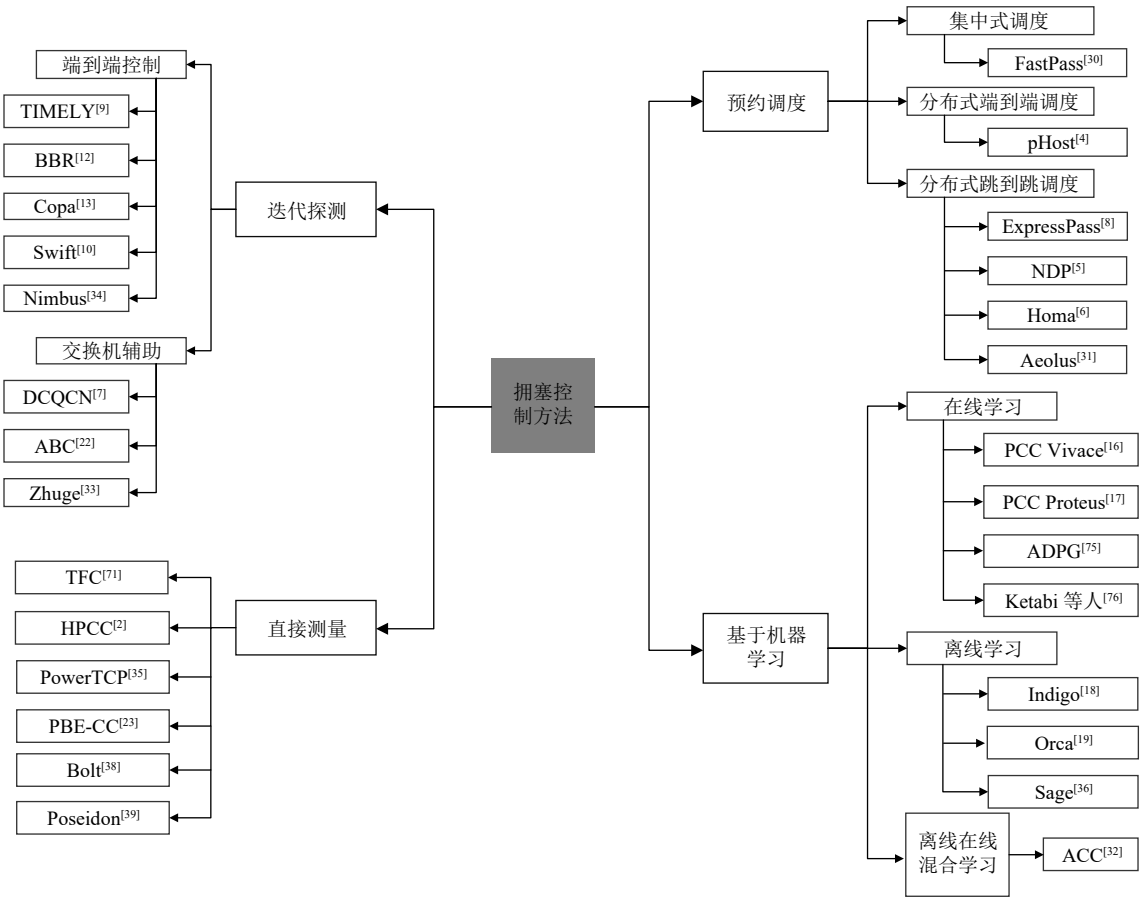


图 1 拥塞控制分类

3.1 迭代探测式

目前大多数算法都使用迭代探测的方式. 这类拥塞控制算法逐轮地多次调整发送速度, 当试探到网络拥塞时减速以避免较大的排队延时, 否则持续加速获取网络带宽. 我们根据迭代探测式的拥塞控制算法是否依赖交换机的支持进一步将其分成端到端控制类和交换机辅助类.

3.1.1 端到端控制

端到端控制类算法在执行过程无需中间节点提供链路状态信息, 而需通过接收端与发送端迭代交互、逐步地消除拥塞. 这类算法一般采用丢包或者延时作为反馈信号. TCP Reno、TCP Cubic 和 Compound TCP[77]是典型的基于丢包的拥塞控制算法. 这类算法不断填充瓶颈链路的缓冲区, 直到丢包后才能发现网络拥塞, 这会导致极大的排队延时. TIMELY[9]、Copa[13]和 Swift[10]等是以延时为拥塞信号的拥塞控制算法. 由于数据包传输延时的变化可以反映网络链路交换机缓冲区的队列信息, 所以它们能更加精确地指导发送端的速率调节.

端到端控制算法的框架如图 2 所示. 发送端负责通过丢包、RTT 或 ECN 等机制来感知网络拥塞, 再根据事先设定的拥塞控制算法迭代地调节速率, 发送数据包; 而接收端负责向发送端回复确认包. 关键之处是端到端控制算法如何感知拥塞以及采用什么方式迭代地调节速率.

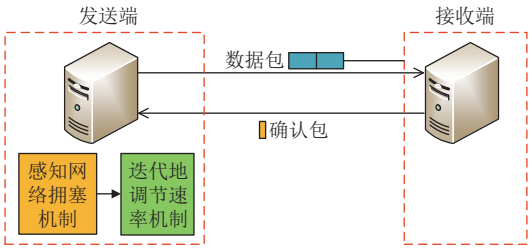


图 2 端到端控制的迭代探测算法框架

TIMELY: TIMELY[9]算法是最早用于数据中心中基于延时的拥塞控制算法, 其通过 RTT 的梯度来调整传输速率, 能在提供高带宽的同时保持较低的数据包延迟. TIMELY 的重要贡献在于证明了网络接口卡 (network interface card, NIC) 测量到的 RTT 足以精确估计交换机处的排队延时. 它以一个消息段为单位, 通过消息段的发送时间戳和接收时间戳来计算 RTT. 得到 RTT 后, TIMELY 根据每次计算的 RTT 与两个参数 RTT_{min} 和 RTT_{max} 比较来进行速率调整. 当 RTT 小于 RTT_{min} 时, 发送端加法增加发送速率. 当 RTT 大于 RTT_{max} 时, 发送端乘法减少发送速率. 当 RTT 介于 RTT_{min} 和 RTT_{max} 之间时, 则根据 RTT 的变化梯度进行速率调整.

TIMELY 的实验结果表明其相比 DCTCP[40]可减少 92.3% 的 99 分位数尾部延时. TIMELY 的不足之处在于其调速机制是启发式的, 收敛速度较慢, 在遭遇突发流的情况下容易产生较大的排队延时.

BBR: BBR[12]与传统拥塞控制算法不同, 它通过不断调节速率以使网络中的数据包数等于 BDP, 即将网络的拥塞标志点设为 BDP. 为此, BBR 需估计瓶颈带宽和网络基本延时 RTT_{min} . BBR 通过接收速率估计可用带宽, 它周期性地呈 1.25 倍增大发送速率以饱和瓶颈链路、探测瓶颈带宽; 呈 0.75 倍减少发送速率以减少网络中排队. 然后, BBR 在 1 个 RTT 内只发送 4 个数据包, 以排空网络中队列来获取 RTT_{min} , 得到以上两个估计的参数后就可以计算出链路 BDP. 对 BBR 的实验结果评价表明 BBR 在一些场景下比 Cubic 吞吐量更高.

BBR 的问题是它利用发送端的接收速率估计带宽及主动排空队列估计 RTT_{min} , 两者均存在缺陷[78], 导致 BBR 无法准确估计 BDP, 进而在某些场景下存在链路利用率下降或者拥塞的问题. 而且 BBR 的速率调节机制和固定的速率增加幅度对其他协议的竞争流量可能过于激进.

Copa: Copa[13]同样是一种基于延时的拥塞控制算法, 与 TIMELY 不同的是它应用于因特网. Copa 通过周期性清除瓶颈缓冲区队列来进行延迟估计, 并设计了保证 TCP 友好性的竞争者模式, 实现了低延时高吞吐量及良好的公平性. Copa 的关键设计是需要根据队列延时计算出一个目标速率. 该目标速率用于和当前速率的大小比较来判断拥塞. 如果当前速率小于目标速率, 则增速; 反之, 则减速. 通过保持目标速率, Copa 保证只存在 Copa 流竞争时,

缓冲区队列在 5 个 RTT 内被定期清空. 同时, 为了和基于丢包的 TCP 竞争, Copa 设计来了竞争者模式可更激进地增加速率. 当缓冲区队列在 5 个 RTT 内没有清空, 则 Copa 认为存在 TCP 流, 于是进入竞争者模式以获得良好的吞吐量. 之后一旦探测到队列清空, 就进入默认的非竞争者模式.

在真实网络中, Copa 相比 Cubic 和 BBR 达到了同等的吞吐量, 但排队延时降低了 50%–90%. Copa 的问题是在某些情况下, 它不会像预期那样定期清除缓冲区, 因此 Copa 可能会错误地估计存在其他类型流, 从而错误地进入竞争者模式, 导致较大的排队延迟和不公平.

Swift: Swift[10]是谷歌公司研究人员针对数据中心网络提出的基于延时的拥塞控制算法. 与 TIMELY 最关键的不同之处在于它创新性地提出主机拥塞, 并将网络系统的延时解耦. Swift 将延时分解成网络延时和主机延时. 为了同时应对网络延时和主机延时, Swift 的发送端维持两个拥塞窗口 *fcwnd* 和 *ecwnd*, 其分别以目标网络延时和目标主机延时为调整目标进行调速, 最终窗口为 *fcwnd* 和 *ecwnd* 中的最小值. 考虑到数据中心复杂和高可变的网络拓扑以及动态变化的用户流量, Swift 会动态调整目标网络延时, 而目标主机延时设置为固定值. 在 100 Gb/s 网络的实验结果表明, Swift 对于短消息可达到小于 50 μs 的尾部延时. 在实际的集群中, 相比 DCTCP, 可减少 90% 以上的包丢失率.

Swift 的良好性能依赖于用网卡的时间戳准确测量延迟, 但延时信号的准确测量需要解决时钟同步、ACK 压缩等问题, 这间接地要求软件技术的提升. 另外, 目标网络延时作为 Swift 进行拥塞控制的指导目标, 需要进行合理的设置, 以避免带宽的损耗或者延时/丢包率的增加, 目前的设置基于实验观察的结果, 并未经过严谨的理论推导, 在算法实际运行中存在未知的问题.

Nimbus: Nimbus[34]从一个新角度来进行拥塞控制, 它根据网络中的背景流量的类型决定速率调节方式. 依据背景流量对空闲带宽的响应特性, Nimbus 将背景流量分为弹性流量与非弹性流量. 如果一条数据流感知到空闲带宽时, 会增加其发送速率, 否则降低发送速率, 则称该数据流为弹性流, 反之则称该数据流为非弹性流. 当背景流量中包含任意一条弹性流, 则称当前的背景流量为弹性流量, 否则为非弹性流量. Nimbus 通过建立网络模型分析了背景流量发送速率与当前流量发送和接收速率的关系, 作为弹性流量探测的依据. 如果探测的背景流量为弹性流量时, Nimbus 会将拥塞控制模式切换至以获取高吞吐率为目标的 TCP 竞争模式 (TCP-competitive mode); 反之, 若 Nimbus 探测的背景流量为非弹性流量时, 它会将拥塞控制模式切换至以降低传输延时为目标的延时控制模式 (delay-control mode).

Nimbus 提出的弹性探测及模式切换机制可以很好地解决端到端拥塞控制算法与背景流之间的共存问题, 并且可以非常容易地推广应用到现行的各种端到端拥塞控制算法中. 模拟和真实实验表明 Nimbus 可达到与 Cubic 一样的吞吐量, 且当存在非弹性流量时, 延时可降低 50–70 ms. 然而, 由于当前大多数基于延时的拥塞控制算法流量均为弹性流量, 当这些流量作为背景流量与 Nimbus 共存时, Nimbus 会进入 TCP 竞争模式, 进而打压这些背景流量, 不仅会造成这些背景流量的吞吐率丢失, 也会造成较高的传输延时.

3.1.2 交换机辅助

与端到端控制相对的则是依赖交换机辅助的拥塞控制算法, 这类算法需要交换机提供网络中间节点的队列信息, 以此来衡量网络是否拥塞. 其框架如图 3 所示. 关键之处是发送端如何根据交换机的信息来感知拥塞并迭代地调节速率.

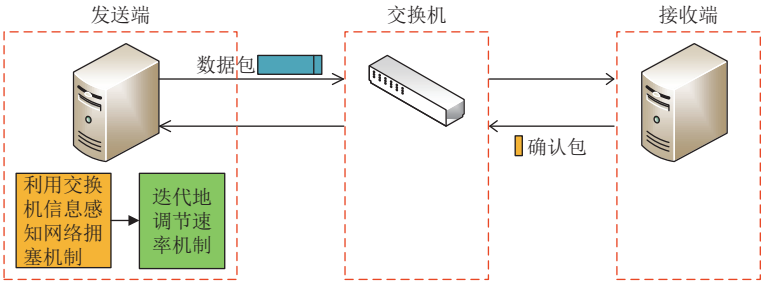


图 3 交换机辅助的迭代探测算法框架

DCQCN: DCQCN[7]结合之前经典的数据中心传输协议 DCTCP[40]以及量化拥塞通知机制 (quantized congestion notification, QCN)[79]而提出的方案, 它与 DCTCP 算法一样采用 ECN 作为拥塞信号, 且学习了 QCN 算法的速率调整机制. 如果队列长度超过设置的 ECN 阈值, DCQCN 交换机会将数据包头部的 ECN 位置为 1. 到达接收端的被标记了 ECN 的数据包指示网络发生了拥塞. 此时接收端便会定期生成拥塞通告包 (congestion notification packet, CNP) 通知源端减速, 直到交换机不再对数据包标记 ECN.

DCQCN 提高了 RDMA 流量的吞吐量, 改进了流量的公平性. 相比 DCTCP, 可进一步减少排队延时. 但由于 DCQCN 使用启发式的调速方式, 导致它在网络拥塞时, 需多次迭代减速, 拥塞消除速度慢、延时大. 而当网络不拥塞时, DCQCN 探测性的加速机制无法及时快速获取全部可用带宽, 导致损失部分吞吐量.

ABC: ABC[22]是一种针对无线网络的显式拥塞控制协议. 它利用无线路由器监测网络带宽, 并为每个数据包添加加速或中断标记, 发送端则根据收到 ACK 中的标记不断进行调速. 当发送端收到含加速标记的 ACK 时, 拥塞窗口加 1. 相反, 收到中断标记时, 减 1. ABC 路由器利用链路速率估计值和队列延时来计算目标速率, 然后通过标记每个数据包来引导发送端达到目标速率, 其计算目标速率 $tr(t)$ 的规则如下:

$$tr(t)=\eta\mu(t)-\max\left(\frac{\mu(t)}{\delta}(x(t)-d_t),0\right) \tag{2}$$

其中, $\mu(t)$ 为网络带宽, $x(t)$ 为队列长度, d_t 是预设的队列阈值, η 为小于 1 的常数, δ 为正常数. 接着, ABC 路由器会将目标速率 $tr(t)$ 变换为数据包的标记数量, 进而改变拥塞窗口大小.

ABC 在无线网络中相比 BBR 可降低 54.5% 的延时, 相比优化的 Cubic 仍提高 30%–40% 的吞吐量. 由于 ABC 需要修改交换机, 因此部署开销较大. 尽管 ABC 能在 ABC 路由器的帮助下快速获取链路带宽, 但当有新数据流进入时, ABC 存在收敛慢的问题[80].

Zhuge: 与 ABC 不同, Zhuge 直接将拥塞信号从无线接入点发回到源端. Zhuge[33]针对无线网络“最后一公里”, 即从无线接入点到端设备, 因拥塞信号反馈延迟引起的尾部延时问题在接入点路由器处测量队列长度和出队速率等信息, 用来估计数据包从该点出发直到被接收端接收所需要的时间, 使得发送端能够更早获知当前的网络状况以便及时响应拥塞, 从而有效降低尾延时. Zhuge 通过延时预测模块和反馈更新模块来预测数据包的网络延时和反馈信息给发送端.

Zhuge 将数据包的网络延时分成了排队延时和传输延时两部分进行测量和统计. 排队延时进一步被分为长期排队延时和短期排队延时. 长期排队延时是为了应对通常情况下队列的动态变化情况; 这是由于无线网络的共享特性导致了无线信道资源的竞争和频繁的带宽波动. 短期排队延时是为了反映队列瞬态变化的情况.

Zhuge 使用现有反馈机制携带反馈信息, 无需修改服务器和无线接入点的路由器. 一种反馈机制为带外反馈机制. 该机制采用诸如 ACK 的反馈信号, 发送端调速所需要的网络信息由发送端自行提取 ACK 信息进行计算; 另一种反馈机制是带内反馈机制, 由接收端收集数据包计算网络信息并定期发送给发送端.

Zhuge 可降低尾部延时和提升应用性能约 17%–95%. 现有的拥塞控制算法可以和 Zhuge 搭配, 加快发送端对网络环境和状态的感知, 提升对网络的适应性.

3.1.3 小 结

表 3 从拥塞探测机制、速率迭代机制对比了上述迭代探测式的方案, 表 4 则比较了端到端控制和交换机辅助两种方法的优缺点.

3.2 直接测量式

直接测量算法的关键是利用交换机或者路由器的功能来精确测量当前的网络状态并显式反馈信息, 其框架如图 4 所示, 以便发送端快速做出拥塞反应, 并能准确地根据测量信息进行速率分配、控制网络拥塞.

3.2.1 典型方案

本小节介绍了 TFC[71]、HPCC[2]、PowerTCP[35]以及 PBE-CC[23]这 4 种直接测量式的典型方案, 具体如下.

TFC: TFC[71]是一种可以直接测量链路带宽资源的算法, 通过将这些带宽资源平均分配给竞争流, 可以实现高链路利用率、快速收敛以及接近零的排队. 在 TFC, 链路带宽资源, 即带宽和时间的乘积, 以令牌数来分配. 为确定

令牌数, TFC 定义了时隙区间. 时隙区间是由 TFC 从经过交换机同一个端口的所有流中选择一条流, 将该流的 RTT 作为测量时隙. 链路带宽由交换机出口带宽即可确定. 接下来, TFC 按拥塞窗口大小发送数据, 它对每个拥塞窗口中的第一个数据包进行标记. 在每个时隙中, 交换机收到的带标记的包数目称为有效流数. 得到有效流数、时隙和链路带宽后, 便可以计算出每条流的拥塞窗口大小.在小规模实验床及大规模模拟中, TFC 达到了高吞吐量、快速收敛、接近 0 的队列及包丢失. 例如 DCTCP 查询流的完成时间是 TFC 流的 30 倍. 由于 TFC 中的交换机负责测量令牌数、有效流数以及拥塞窗口计算, 因此 TFC 需要修改交换机, 不利于立即部署.

表 3　迭代探测式方案对比

名字	拥塞探测机制	速率迭代机制
TIMELY	拥塞点设在 RTT_{min} 和 RTT_{max} 之间, 根据RTT梯度调整速率	AIMD
BBR	拥塞点设为BDP, 根据飞行包是否等于BDP调整速率	MIMD
Copa	拥塞点设在 RTT_{min} 和 RTT_{max} 之间, 利用队列延时计算目标速率	AIAD
Swift	拥塞点设在 RTT_{min} 和 RTT_{max} 之间, 利用队列延时计算目标速率	AIMD
Nimbus	竞争模式拥塞点设为缓存溢出, 延时控制模式拥塞点设在 RTT_{min} 和 RTT_{max} 之间	AIMD+AIAD
DCQCN	根据队列大小调节速率, 使其保持在目标速率	AIMD
ABC	利用目标时延与链路带宽计算出目标速率	AIAD
Zhuge	通过无线网络接入点的队列长度、出队率等信息, 估计数据包延迟	由搭配的拥塞控制算法决定

表 4　端到端控制和交换机辅助方法对比

方法	代表方案	优点	缺点
端到端控制	TIMELY、BBR、Copa、Swift、Nimbus	易于部署在各种网络中, 部署开销低	有时并不能正确感知网络中拥塞信息
交换机辅助	DCQCN、ABC、Zhuge	准确感知网络中拥塞信息	需要交换机支持, 部分方案需要修改交换机, 部署开销较大

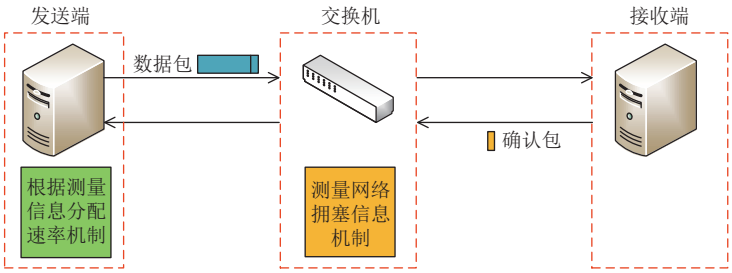


图 4　直接测量式的框架

HPCC: 与 TFC 不同, HPCC[2]采用商业交换机支持的 INT 技术 (inband network telemetry) 直接测量链路 BDP, 并通过调节拥塞窗口将飞行包数量控制在略低于 BDP 的点以实现高吞吐和低延迟, 其中飞行包为链路中传输的数据包数目与交换机队列中的数据包数目之和. 在 HPCC, 数据包经过交换机时会在包头部中添加 INT 信息, 其包括时间戳 ts 、队列长度 $qlen$ 、发送字节 $txBytes$ 和链路带宽 B . 接收端会将数据包头部的 INT 信息复制到 ACK 中, 并反馈给发送端. 发送端则根据 INT 信息调整流量.

每接收到 ACK, HPCC 的发送端就可以根据 INT 信息计算出链路利用率 U_j . 为了保持极低的排队延迟, HPCC 引入一个参数 η (默认等于 0.95) 用于限制链路利用率略低于 1, 这意味着通过牺牲少量带宽以达到零排队. 根据这些量, HPCC 计算发送端 i 的拥塞窗口更新公式为:

$$W_i = \frac{W_i}{\max_j (U_j)/\eta} + W_{AI} \tag{3}$$

其中, W_{AI} 为小的常数. HPCC 能够达到快速收敛以及高吞吐量, 实验结果发现 HPCC 相比 DCTCP 和 TIMELY 能

减少 95% 的流完成时间. 但是, 不是所有交换机都支持 INT 功能, 因此其应用场景范围受到了限制.

PowerTCP: PowerTCP[35]将网络属性与电路中的概念进行类比, 将现有的算法分为两类: 基于电压的拥塞控制和基于电流的拥塞控制. 基于电压的拥塞控制算法对 BDP 相关的绝对状态做出反应; 而基于电流的拥塞控制算法对网络状态变化, 如 RTT 梯度, 做出反应.

PowerTCP 认为拥塞控制算法应同时与“电压”和“电流”相关, 因此 PowerTCP 提出并定义网络功率, 表示网络中电流与电压的乘积, 对应于总传输速率 λ (电流) 和链路内飞行数据包 Inflight (电压) 的乘积. 对于每条数据流 i, PowerTCP 基于下列信息更新其拥塞窗口: 当前时刻 t 的窗口大小 $w_i(t)$ 、加性增长系数 β 、 $t-\theta(t)$ 时刻发送端的窗口大小 $w_i(t-\theta(t))$ 、从反馈信息测量的功率 $f(t)$. 定义 e 表示算法期望到达的平衡点, PowerTCP 中 e 为带宽延迟积乘以带宽, γ 表示权重常数, 拥塞窗口表达式为:

$$w_i(t) \leftarrow \gamma \cdot \left(w_i(t-\theta(t)) \cdot \frac{e}{f(t)} + \beta \right) + (1-\gamma) \cdot w_i(t-\theta(t)) \tag{4}$$

PowerTCP 利用 INT 技术来直接测量得到上式中的 $f(t)$ 和 e, 进而算出更新的窗口 $w_i(t)$. 进一步, 为扩大适用性、摆脱对 INT 的依赖, 作者还提出了 θ-PowerTCP, 以部署在带有传统交换机的数据中心中. θ-PowerTCP 只需要测量 RTT 就可以按与公式 (4) 相似的方式更新窗口.

相比于采用了相同 INT 技术的 HPCC, PowerTCP 在优化了调速目标的同时, 还考虑了系统变化率, 使得其调速能够更快收敛. 实验结果表明相比 DCTCP 和 TIMELY, 它减少了短流 80% 的尾部延时; 相比 HPCC, 则可减少 33%. 另外, θ-PowerTCP 降低了 PowerTCP 的部署门槛, 使其能部署在更广泛的场景.

PBE-CC: 与前述方案不同, PBE-CC[23]用于无线蜂窝网中的拥塞控制. 它的客户端 (接收端) 通过解码蜂窝网的控制信道获取蜂窝网基站收集的总网络容量、可用链路容量和活跃用户数量, 并反馈给 PBE-CC 发送端. 当瓶颈位于蜂窝网时, 发送端基于这些信息进行精准调速; 而当瓶颈位于有线网络时, PBE-CC 采用类似 BBR 的迭代探测机制进行调速. PBE-CC 调速主要分为启动阶段和稳定阶段.

● 启动阶段: 连接建立后, 发送端速率将在几个 RTT 内线性增长到目标速率. PBE-CC 通过感知蜂窝内基站上的总可用带宽 P_{cell} , 以及蜂窝内数据流数目 N , 计算出理想公平分配带宽:

$$P_{\text{exp}} = P_{\text{cell}}/N \tag{5}$$

结合无线链路的物理数据传输速率 R_w , 将公平分配带宽转换为公平分配目标速率 $C_f = R_w \times P_{\text{exp}}$.

● 稳定阶段: PBE-CC 根据瓶颈所在的网络进行拥塞避免. 当单向数据包传输延时小于阈值 D_{th} , 则 PBE-CC 判断瓶颈在无线网络中时, PBE-CC 会通过获取蜂窝内空闲带宽和数据流数目, 调节发送速率以使每条 PBE-CC 流公平地共享空闲带宽反之, PBE-CC 认为瓶颈处于有线网络中. 此时, 它会采用与 BBR 类似的机制进行速率调节. PBE-CC 会把发送速率设置为 $0.5Btlbw_{\max}$, 这里 $Btlbw_{\max}$ 是网络最大可用带宽. 此速率持续一个 RTT 以排空瓶颈处队列. 之后 PBE-CC 将进入最大可用带宽探测阶段. 当 PBE-CC 连续地以 C_f 的速率发送数据包, 且未造成有线网络中的数据包排队时, 也即此时单向传输延时小于 D_{th}, 则 PBE-CC 判断瓶颈转移至无线网络中, 并重新开始按照无线网络瓶颈下的规则调速.

PBE-CC 通过感知蜂窝网基站为数据流分配的带宽来设置与其相匹配的发送速率, 从而避免在无线链路拥塞时出现队列堆积. 此外, 当出现空闲带宽时, PBE-CC 能够令共享瓶颈的数据流获取较好的公平性. 实验结果表明 PBE-CC 比 BBR 提高了 6.3% 的平均吞吐量, 减少了 44.5% 的 95 分位数延时. 然而, 当有新数据流进入时, PBE-CC 没有保障公平分配带宽的机制, 此时数据流之间的公平性依赖于蜂窝网基站对带宽的分配.

3.2.2 小 结

表 5 从拥塞测量和调速方式两个方面对比了上述算法.

3.3 预约调度式

预约调度式的拥塞控制思想是通过调度器主动对网络带宽进行统一的预约和分配, 以使总发送速率尽可能匹配瓶颈链路带宽. 这样既可以充分利用带宽, 又能防止数据包过多堆积在交换机缓存. 调度方式有集中式调度和分布式调度 2 类. 进一步, 分布式调度又可分端到端和跳到跳的控制.

表 5 直接测量式方案比较

名字	拥塞测量方式	速率分配方式
TFC	测量每个规定时隙中的有效流数	直接计算每条流的拥塞窗口
HPCC	利用INT技术测量链路BDP	控制网络流量Inflight略低于BDP
PowerTCP	利用INT技术测量链路BDP以及网络传输速率	控制网络流量Inflight等于BDP且发送速率等于网络带宽
PBE-CC	通过解码蜂窝网的控制信道获取总网络容量、可用链路容量、活跃用户数量	启动阶段公平分配带宽, 稳定阶段公平共享空闲带宽

3.3.1 集中式调度

集中式方案主要依靠集中式调度器对网络资源预约和分配, 终端依据调度器的分配进行数据包发送, 其整体框架如图 5 所示, 该方法的关键是调度器如何对数据包进行全局调度.

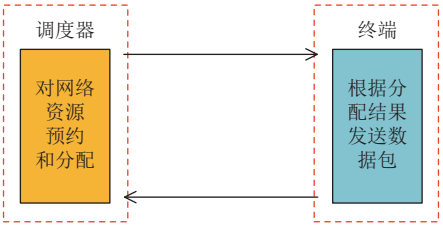


图 5 集中调度式框架

● FastPass: FastPass[30]采用集中式调度的方法来控制网络拥塞, 通过对每个包细粒度的控制实现了接近零的队列. 它的全局控制器是一个仲裁器, 相当于图 5 中的调度器. 终端主机使用可靠的 FastPass 控制协议 (FastPass control protocol, FCP) 与仲裁器通信. 仲裁器包括 FCP 服务器、时间片分配器和路径选择器. 在数据传输中, 终端主机首先向 FCP 服务器提供目标地址和流量大小信息. 然后, FCP 服务器将相关信息传递给时间片分配器为流分配时间片. 再由路径选择器决定流的传输路径, 最后 FCP 服务器将时间片和路径返回给终端主机. 具体地, FastPass 仲裁器的主要操作为:

● 时间片分配: 该功能模块的目标是为了在每一个时间片中选择一个源-目标对. 为充分利用链路, 时间片的大小是基于带宽和最大传输单元 (maximum transmission unit, MTU) 计算的. 仲裁器根据“一个时间片传输一个 MTU 大小的数据包”为源分配目的地资源. 仲裁程序可以通过更改源-目标对的分配顺序来实现不同的分配策略. 例如, 可以通过“最近分配最少优先”实现最大最小公平性; 通过“剩余 MTUs 最少优先”实现最小流完成时间 (flow completion time, FCT).

● 路径选择: 路径选择器为数据包分配传输路径, 使得在一个时间段内只能有一个数据包在该路径上传输. 这样 FastPass 就可以确保网络中没有包排队.

FastPass 不仅要对所有网络的需求有全面的了解, 还需要对每个数据包进行调度, 算法开销大, 不利于部署在规模大的网络中. 另外, 全局调度器会有单点故障的问题.

3.3.2 分布式端到端调度

分布式方案将拥塞控制的任务分配给网络中的各个节点协作完成. 在分布式端到端的拥塞控制中, 发送端直接发送请求到接收端, 由接收端预约和分配网络资源, 而不需要交换机的参与, 其框架如图 6 所示, 方案的关键之处是发送端和接收端需要协作调度发送数据包.

pHost: pHost[4]是基于终端主机的预约调度式算法, 不涉及中间交换机的修改, 达到了低延时传输. pHost 的主要执行过程如下. 在数据发送开始时, pHost 发送端首先发送一个 RTS (request-to-send) 包到对应的接收端, 该 RTS 包中包含一些流信息如流大小等以便接收端利用这些信息进行调度决策. 对于收到的所有流的 RTS 包, 接收端在每个包发送时隙, 根据调度策略 (比如最短剩余时间优先) 选择一条流的 RTS 包然后返回一个令牌包到该流的发送端. 每个令牌包可以引发发送端发出一个数据包. 发送端收到令牌后, 在每个数据包发送时隙发送一个包.

如果发送端收到多个接收端返回的令牌包, 则从多个接收端中选出一个向其发送数据包. 由于 pHost 的端主机可以采用不同的调度策略, 因此它能依据不同的性能目标而进行配置.

该方案的挑战是发送端调度和接收端调度如何进行完美的匹配. 如果接收端发送令牌包给发送端, 但是发送端没有选择该接收端发送数据包, 则会导致令牌包浪费, 造成网络利用率下降.

针对该问题, pHost 采用了 2 个机制: (1) 给令牌包设置一个超时时间. 如果时间到期时, 该令牌未被利用, 则记录未被利用的次数. 如果超过一定值, 则将令牌所属的流优先级降级. (2) 发送端初始时设置一定数量的自由令牌. 它能在没有收到令牌包时也能发送数据包, 防止了流等待, 这对于提高短流的性能很重要.

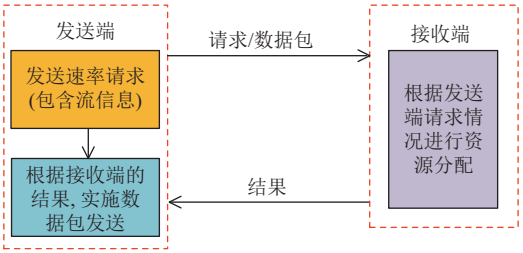


图 6　分布式端到端调度的框架

实验结果表明 pHost 在降低流的平均完成时间上达到了 FastPass 的 26.3%. 但是, 由于 pHost 的发送端和接收端同时参与调度, 整个算法比较复杂, 且发送端和接收端难以协调地对流进行最优调度.

3.3.3　分布式跳到跳调度

跳到跳的拥塞控制机制需要交换机对网络中间链路辅以检测和管理, 发送端、接收端共同完成资源的分配和调度, 其框架如图 7 所示. 方法的关键是如何利用交换机提供的信息来进行或辅助数据包的调度发送.

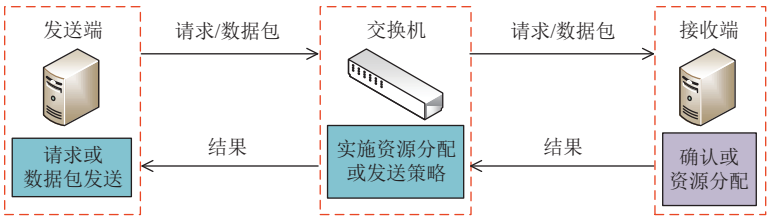


图 7　分布式跳到跳调度的框架

ExpressPass: ExpressPass[8]是基于跳到跳的拥塞控制. 其受基于信用的流量控制 (credit-based flow control, CBFC) 所启发, 并将其扩展到端到端. 在网络链路内部, 交换机为信用包 (credit packet) 维持一个队列, 并对经过的信用包进行速率限制, 如果信用包速率超过设定阈值, 则丢弃超过设定阈值的信用包, 从而达到了主动限制发送端的发包速率的目的. 接收端总是以线速发送信用包, 这种激进的方式可以很快获取带宽, 但是它会在多瓶颈链路场景时存在链路利用率低以及不公平分配带宽的问题.

为了缓解多瓶颈链路下的不公平和链路利用率不足, 以及适应多变和复杂的数据中心环境, ExpressPass 在交换机对信用包进行速率限制, 并设计信用反馈控制机制, 以探测是否发生信用包损失, 据此来动态地调节数据包的发送速率.

在 10 Gb/s 网络中, ExpressPass 可比 DCTCP 收敛快 80 倍, 并减少了流完成时间. 尽管 ExpresssPass 提出一些机制可以减轻链路利用率低和不公平带宽共享的问题, 但是完全解决这些问题仍存在技术上的难度.

NDP: 相比 ExpressPass, NDP[5]重新架构了整个传输协议栈. 新架构中, NDP 可以很好地区分乱序包和丢失包, 避免不必要的重传带来的性能损失为此, NDP 交换机需要支持特别的功能. 其在缓存超过一个值, 如 12 KB 时, 对进来的数据包或者队列中最后一个包进行“包负载丢弃”操作, 使得数据包只剩包头, 以避免完全的包丢失. 其次, 交换

机让剩余包头进入高优先级队列, 普通的数据包进入低优先级队列, 以便接收端尽快收到这种只剩包头的数据包.

NDP 接收端会收到 2 类数据包: 正常的数据包和只剩包头的数据包. 若收到一个正常的数据包, NDP 接收端会生成一个推拉包 (pull packet), 保存在推拉队列中, 并根据轮转策略返回推拉包给相应的发送端. NDP 发送端每收到一个推拉包发送一个数据包, 因此 NDP 是一种通过接收端驱动来预约带宽的协议. 如果收到了只剩包头的数据包, NDP 接收端立即返回一个 NACK 包, 其中包含了被切包的数据包序列号, 以使发送端据此快速重传丢失了负载的数据包.

NDP 相比 DCTCP 可使短流完成时间降低 66.7%–75%, 并优于 DCQCN. 但由于需要交换机支持定制的操作, NDP 需要对交换机进行一定的修改, 所以其部署成本较其他拥塞控制算法要高.

Homa: Homa[6]是针对消息传输来设计的. 消息不同于流的地方在于发送端在发送消息时需要指定消息大小. 利用消息大小, Homa 可实现剩余字节最少的消息优先传输的策略, 以降低短消息延时. Homa 同样采用接收端进行速率预约, 它通过发送授权包 (grant packet) 控制速率的方式来达到极低的消息传输延迟.

Homa 将待传输的消息被划分为 2 个部分: 非调度部分和调度部分. 消息初始的一部分数据均为非调度部分, 直接以线速发送, 且优先级低. 接收端每收到一个数据包后都以接收端链路速率返回一个授权包, 该授权包引出的数据包为调度部分数据, 优先级高. 消息的具体优先级根据消息大小确定. 但由于交换机通常只支持有限的优先级队列数目, Homa 采用了动态优先级的策略. 对于非调度部分的数据包, Homa 预先分配固定的优先级. 对于调度部分的数据包, Homa 接收端根据收到的发送端信息, 如剩余消息大小, 动态地给每个数据包赋予一个优先级. 剩余字节最少的消息的数据包被赋予最高的优先级.

Homa 可在 10 Gb/s 网络上以小于 15 μs 的延时发送短消息. 由于 Homa 假定网络拥塞不发生在核心层, 并通过接收端以链路速率来进行流控, 因此丢包很少发生. 在少数情况下, 如果发生了丢包, Homa 利用超时重复进行丢包恢复.

Aeolus: Aeolus[31]构建于 pHost、Homa 和 NDP 这些拥塞控制机制之上, 以增强这些机制的传输能力.

Aeolus 注意到 pHost、Homa 和 NDP 均存在首 RTT 容易丢包的问题, 这是由于这些方案均在首 RTT 以线速发送数据包, 如果发送端数目多时, 很容易造成交换机缓存被迅速填满而导致丢包发生, 进而影响流的传输性能. 但是, 如果不以线速发送, 则影响短流的完成时间, 造成本来只需要 1 个 RTT 完成的流发送多个 RTT 才能完成. 针对这一问题, Aeolus 提出一种保护机制来改进之前这些方案的性能.

具体地, Aeolus 将数据分为未调度和调度数据两部分. Aeolus 被设计为保护调度数据不丢失而让未调度数据尽可能发送. 刚开始时, 未调度数据以最大速率发送, 以使得短流可以在首个 RTT 内完成. 但是, 如果 Aeolus 交换机发现缓存队列超过阈值时, 则优先丢弃未调度数据. 为保证未调度数据包尽快得到成功重传, Aeolus 对丢弃的未调度数据包重传时, 把它作为调度数据包来发送.

Aeolus 在模拟实验中可使 ExpressPass 减少 56% 的平均流完成时间. Aeolus 的不足之处是对于能在一个 RTT 内完成的短流, 当它被划为未调度数据, 则这种短流的 FCT 很可能受到其他流的影响而受到极大损害.

3.3.4 小 结

表 6 从预约方式和调度算法两个方面对这些方案进行了总结.

表 6 预约调度式方案比较

名字	预约方式	调度方法
FastPass	全局仲裁器分配	最近分配最少调度
pHost	接收端在每个传输时隙发出一个令牌包, 令牌包到期时会被发送端丢弃	发送端和接收端模拟不同调度算法
ExpressPass	接收端发送信用包预约, 交换机处会根据瓶颈速率决定通过的信用包数	交换机公平调度信用包
NDP	接收端根据接收端速率发送推拉包, 一个推拉包引出一个未丢失的数据包	接收端进行先进先出调度
Homa	接收端可以同时向多个发送端发送授权包, 每个授权包可使发送端投递一个数据包	交换机利用优先级队列进行动态优先级调度
Aeolus	由其所增强的算法决定	与其所增强的算法一样

3.4 基于机器学习式

由于网络系统十分复杂, 要建立精确的网络模型很困难. 此外, 为了适应不同的网络环境, 需要不断修正模型参数. 针对这一问题, 基于机器学习式的拥塞控制方法将网络环境视为一个黑盒子, 构建学习模型刻画当前网络环境, 指导发送速率调节, 从而达到避免对网络假设和手动调整模型、具备良好自适应性的目标.

3.4.1 在线学习

在线学习方案不需要在特定环境下预先训练模型, 而是利用实时的信息不断更新模型参数和神经网络的权重, 理论上可适应所有网络环境, 其框架如图 8 所示. 此类方法关键在于设计学习算法满足在线环境下的约束, 如时间、CPU 开销等. 这是由于在线模型训练可能造成 CPU 开销大和收敛慢等问题. 具体介绍如下.

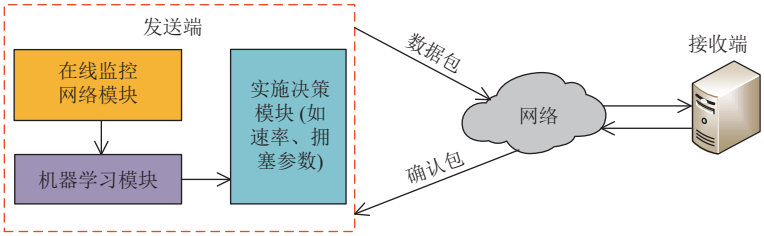


图 8 在线学习的框架

PCC Vivace/Proteus: 近两年著名的在线学习方案 PCC Vivace[16]和 PCC Proteus[17]都基于其母版 PCC 框架[15]进行优化. PCC 通过合理的设计效用函数来衡量网络性能, 以使系统达到良好的收敛性和公平性[15–17]. PCC 采取无悔式的在线学习模式, 确保决策速率不会低于当前速率产生的效用值, 从而使网络性能向提升的方向演化. 无悔式在线学习包含 2 个阶段.

(1) 决策阶段: 将时间线划分为连续的监控区间 (monitor interval, MI). 假设当前的发送速率为 r, 以连续的 4 个 MIs 为一组对 r 进行随机试验测试. 将连续的两个 MIs 分为一对, 每一对交替使用 $r+\varepsilon$ 和 $r-\varepsilon$ 进行发包. 在 4 个 MIs 结束之后使用速率 r 进行发包, 等待前面 4 个 MIs 发出的数据包的 SACK 信息, 以计算得到 4 个效用值 $u_1(r+\varepsilon)$、$u_1(r-\varepsilon)$、$u_2(r+\varepsilon)$、$u_2(\mathrm{r}-\varepsilon)$. 假如 $u_1(r+\varepsilon)\geqslant(\leqslant)u_1(r-\varepsilon)$ 且 $u_2(r+\varepsilon)\geqslant(\leqslant)u_2(r-\varepsilon)$, 则使用 $r+\varepsilon$ ($r-\varepsilon$) 作为当前的决策速率, 并且将在线学习的方向 dir 置为+1(−1); 反之则加大探索步长, 继续进行随机实验测试直到找到正确的决策速率.

(2) 速率调整阶段: 假如当前的调整方向为 dir(±1), 则每个 MI 都等量增加或降低速率, 并通过收集 SACK 信息计算对应速率的效用值, 直到效用值出现下降, 则退回上一个 MI 的发送速率, 再次进入决策阶段.

PCC Vivace 和 PCC Proteus 继承了 PCC 的控制框架和无悔式的在线学习方式, 来保证系统的整体性能. 与 PCC 不同的是, PCC Vivace 对延时的要求更高, 在效用函数中将 PCC 原来的平均延时项替换为延时梯度. PCC Proteus 则进一步将效用函数改进, 引入延时偏差, 并优化了 PCC Vivace 存在的协议间友好共存的问题.

总的来说, PCC 系列的拥塞控制方案建立了一种新型的在线学习框架. 相比 Cubic, 在吞吐量、缓存占用等 PCC 可达到 10 倍好的性能. PCC Vivace 相对于 BBR 和 Cubic 分别降低了 50.7% 和 95.5% 的延时. 然而, PCC 的在线学习机制面临着收敛慢或不收敛的问题[19], 并且 PCC 难以内核实现. 因此, PCC 往往部署在基于用户数据包协议 (user datagram protocol, UDP) 的用户空间, 存在较大的 CPU 开销; 此外, 在后续测试中, PCC 的性能表现并不总是很理想[15,73,74].

ADPG: Chen 等人[75]考虑将强化学习应用到数据中心网络中速率的控制. 他们指出将强化学习应用到网络中有几个独特的问题需要考虑: (1) 拥塞控制任务涉及多条流的交互, 如果每条流看作由一个智能体控制, 则该任务涉及多智能体; (2) 由于每个智能体只能获得其控制的流的状态, 而对其他流的状态未知, 所以该任务是状态部分可观测的. 针对网络中拥塞控制任务属于多智能体且状态部分可观测的任务, 文章提出了解析的确定策略梯度 (analytic seterministic policy gradient, ADPG), 在确定策略梯度算法基础上, 给出回报梯度的一个近似的解析式. 该

算法采用同策略 (即要优化的策略与用于生成结果的策略是同一策略), 对每条流测量其发送速率和 RTT, 根据这些信息来更新策略, 得到新的速率. 进一步, 为增加算法的泛化能力, 其输出是前一个速率的倍数 a, 满足 $a \in [0.8, 1.2]$.

作者们在 3 种模拟场景下进行测试: 多对一、多对多及多条短流抢占长流带宽场景. ADPG 在交换机利用率、公平性和队列延时上达到甚至超过 HPCC、DCQCN 和 Swift. 在真实的多对一场景中, ADPG 利用率好于 HPCC, 与 DCQCN 性能接近.

文献 [76]: Ketabi 等人[76]同样利用深度强化学习, 但他们的目标是自动生成拥塞控制的参数来优化拥塞控制算法在数据中心网络中的性能. 该工作的主要挑战在于数据中心网络对延时要求严格, 在微秒级别. 此外, 网络状态会动态变化. 作者们设置参数的调节区间在数秒至数小时, 以便机器学习算法可以在该时间内统计流的状态, 并计算出优化后的参数. 作者们采用近端策略优化 (proximal policy optimization, PPO) 算法, 该算法根据部署在主机上的智能体收集的状态量, 如 RTT、吞吐量等, 来计算优化后的拥塞控制参数.

作者们将该算法应用到 BBR 上, 在测试期间, 能在 60% 时间内正确估计出 RTT, 并减少了 BBR 2.7 倍的收敛时间和 40% 的峰值 RTT.

3.4.2 离线学习

离线学习的方案需要预先在特定网络环境中训练好模型, 训练好的模型可以是基于特定网络状态和发送速率的映射模型 [14,18], 亦或是权重固定的神经网络模型 [19,72,74]. 由于在应用中不再需要中央处理器和图形处理器辅助训练和更新模型, 离线学习的方案具有比在线方案较低的中央处理器开销和较快的收敛时间. 但是, 离线学习模型的性能和训练场景强相关, 当测试场景未经训练时, 面临性能崩塌的风险, 其框架如图 9 所示, 方法关键在于设计学习模型, 并能在离线场景中满足各种约束条件下训练得到良好的性能.

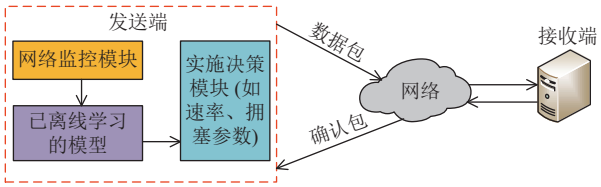


图 9 离线学习的框架

Indigo: Indigo[18]通过模仿学习建立网络状态到拥塞控制动作的映射模型, 它将状态与行为的映射存储在长短期记忆 (long short-term memory, LSTM) 递归神经网络中. Indigo 是数据驱动的算法, 其核心负责两件事: (1) 观察网络状态 (2) 调整拥塞窗口, 以控制网络中的飞行包数量.

Indigo 用模拟链路搭建训练场景, 并用模仿学习算法 DAgger 进行学习. 对于每个训练链路, DAgger 的训练过程如下: 首先, 神经网络在训练链路的模拟器上做出一系列拥塞控制决策, 记录每个决策对应的状态向量. 接着, 它利用已有数据从记录中标记出与每组状态向量对应的正确决策. 最后, 它使用这些正确的状态-动作映射作为训练数据来更新神经网络. 重复此过程, 直到进一步的训练不会改变神经网络.

Indigo 利用模拟链路搭建训练场景, 其训练数据获取代价较低, 但训练耗时较长. 在应用场景和训练场景相符时, 能取得较好的传输性能. 然而在面临带宽有波动的场景时, Indigo 存在局部最优的问题, 无法取得高链路利用率 [19].

Orca: Indigo 这类方案只根据网络环境训练训练拥塞控制算法模型, 完全未参考和利用基于网络知识设计的启发式拥塞控制算法. 而 Orca[19]则将传统启发式设计的 Cubic 和深度强化学习 (deep reinforcement learning, DRL) 模型结合起来. Orca 的拥塞控制主要由 Cubic 进行, 以利用 Cubic 的快速响应优点; DRL 模型则被用于定期调整 Cubic 决策后的拥塞窗口, 以保证 Orca 具有好的自适应性. 同时, 两者结合使得 Orca 训练时间更少、收敛性更快且 CPU 开销更低.

Orca 构建了基于 Cubic 的细粒度演化和基于 DRL 的粗粒度调整的双层控制. 其中, DRL 模块在每个监控时

段 (monitoring time period, MTP) 末强制输出拥塞窗口, 最大化奖励函数. 该窗口将作为下一个 MTP 内底层 Cubic 的调窗基准.

Orca 在跨洲通信中可达到比 Cubic 好 5–20 倍更高的吞吐量, 比 BBR 更低的延时. 由于 Orca 的 DRL 模型采用离线学习的模式, Orca 的性能受制于上述的离线场景.

3.4.3 离线在线混合学习

不同于前文提到的基于机器学习的拥塞控制方案, 离线在线混合学习, 首先预训练得到一个离线模型, 再通过在线监控的结果更新该模型, 以不断优化控制结果, 其框架如图 10 所示, 方法关键在于设计合理的模型, 能根据数据不断更新优化.

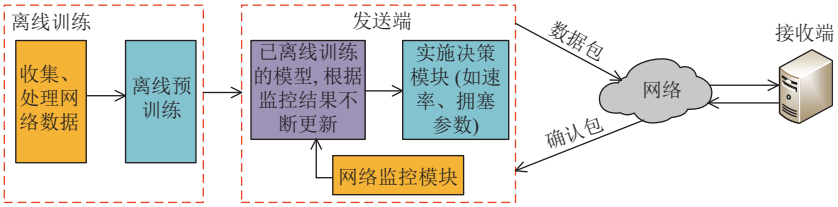


图 10 离线在线混合学习的框架

ACC: ACC[32]采用了离线预训练和在线深度强化学习的混合技术路线. ACC 不再以设计整个拥塞控制算法为目标, 而是着眼于设计 RDMA 网络中的 ECN 阈值自动配置策略, 实现“零配置”的网络内自动优化.

已有研究文献[2,7,40]主要采用固定的 ECN 配置, 无法满足数据中心网络中多种流量模式的需求. 而如果利用人工调整的方法根据流量模式来配置 ECN, 则繁琐耗时. 针对该问题, ACC 提出了一种基于机器学习的 ECN 配置调优方法. ACC 首先通过离线预训练得到一个称为 DRL 代理的神经网络模型, 再根据收集的网络状态信息来对 ECN 配置进行调节, 同时根据奖励值不断更新初始的神经网络模型. ACC 主要设计有:

● 分布式设计: 理想状态下, ECN 配置自动调优应基于网络全局信息, 即将每个交换机的状态上传到中心控制器中的 DRL 代理. 然而, 在实际应用中, 这种中心化的设计面临网络状态和动作空间复杂、状态收集和配置更新的迟滞、数据收集带来的带宽开销等一系列问题. 因此, ACC 采用了分布式设计的方案, 即在每个交换机上部署独立 DRL 代理, 各交换机只基于本地状态的 ECN 配置进行调优.

● 深度强化学习: ACC 采用了深度 Q 学习 (deep Q learning, DQN) 算法设计 DRL 代理. ACC 使用交换机本地的排队长度 ($qlen$)、出口传输速率 ($txRate$)、被 ECN 标记的数据包的输出速率 ($txRate^{(m)}$) 以及当前 ECN 配置 (ECN) 等信息反映网络拥塞. 通过连续 k 个时隙观测上述信息并进行归一化, 形成最终的输入状态. ACC 中 t 时隙的动作 A_t 被直接定义为 ECN 的配置, 即 $A_t=\{K_{\max},K_{\min},P_{\max}\}_t$, $K_{\max}$ 为最高门限, $K_{\min}$ 是最低门限, $P_{\max}$ 为标记概率. ACC 通过离散化, 缩小了动作空间. 相比于其他基于机器学习的拥塞控制方案, ACC 可以作为插件优化现有的各种利用 ECN 作为拥塞信号的拥塞控制算法, 无需对终端做任何修改. 然而其分布地为每个交换机部署独立 DRL 代理的部署方式, 对交换机的性能和资源提出了相对较高的需求.

3.4.4 小 结

我们从学习方式和速率决策方式两个方面对基于机器学习式的方案进行了比较, 如后文表 7 所示.

4 对比分析

本节分析和讨论以上 4 类拥塞控制算法的性能特点. 拥塞控制算法的性能指标主要包括高链路利用率或吞吐量、低丢包率、低延时、收敛速度、公平性和友好性. 这些性能指标之间往往互相冲突. 例如, 网络利用率提高时, 网络排队延时以及丢包率往往增加. 也就是说高吞吐量与低延时、低丢包率之间存在冲突. 另外, 公平性好的方案往往收敛的时间更长一些. 同时, 这些性能大都由拥塞控制算法的瞬态和稳态行为共同决定, 如高吞吐量不仅受拥塞控制算法稳定状态下所能达到的网络利用率影响, 同时也取决于拥塞算法的带宽收敛速度. 同样, 低丢包率、

低延时也由稳定状态下的延时和拥塞消除能力共同决定.

表 7 基于机器学习式方案比较

名字	学习方式	速率决策方式
PCC Vivace/Proteus	无悔式在线学习, 通过速率探测方式决策出效用值最大的发送速率	选择效用值最大的发送速率
ADPG	使用解析的确定策略梯度算法	由学习到的策略决定速率
Ketabi等人[76]	使用PPO强化学习算法	由使用的拥塞控制算法根据优化的参数决定速率
Indigo	使用DAgger算法进行学习, 重复训练并记录状态-动作映射直至不再改变神经网络	根据神经网络中的状态-动作映射做出速率决策
Orca	基于mahimahi搭建模拟链路, 采用TD3算法进行神经网络的训练	根据监测到的环境统计数据, 采取使回报最大化的速率发送数据
ACC	离线预训练模型, 在线DQN学习ECN阈值	根据本地交换机状态, 选择令奖励最大化的ECN配置, 根据ECN标记调节速率

在当前的网络环境中, 流量复杂且变化频繁, 数据流难以长期工作在稳态. 对于拥塞控制算法来说, 瞬态性能和稳态性能同等重要. 因此, 本节首先分别从瞬态性能和稳态性能两方面定性对比拥塞控制方法, 并讨论这些方法适用的网络环境. 瞬态性能指拥塞消除能力、带宽收敛速度、公平性收敛等. 其中, 拥塞消除指网络拥塞使得瓶颈链路队列堆积时, 拥塞控制算法消除堆积的缓存队列的速度; 带宽收敛主要指出现空闲带宽时, 例如网络中有数据流退出后, 拥塞控制算法获取空闲带宽速度, 反映了算法避免带宽浪费的能力; 公平性收敛指当瓶颈链路中只存在同一种拥塞控制协议的流时, 这些流的吞吐量收敛到公平的速度. 稳态性能主要指拥塞控制算法最终达到的稳定状态后的网络利用率 (或吞吐量) 和延时 (如流平均延时等). 其次, 本节对不同网络下的方案进行了定量分析, 以使读者对它们的定量性能有一定了解.

4.1 不同方案的定性对比分析

我们从稳态性能、瞬态性能和适用环境归纳 4 种不同方案定性对比分析结果, 如表 8 所示. 下面进行详细介绍.

表 8 拥塞控制方案定性对比分析

方案	稳态性能		瞬态性能		适用环境
	优点	缺点	优点	缺点	
迭代探测式 (TIMELY, BBR, Copa, Swift, Nimbus, DCQCN, ABC, Zhuge)	吞吐量较高	容易造成队列堆积, 导致流的平均延时较大	通常能够很好地收敛到公平性	拥塞消除能力较差; 带宽收敛时间依赖于调速方式, 一般较长	因特网、数据中心网络、无线网络
直接测量式 (TFC, HPCC, PowerTCP, PBE-CC)	超低延时	额外的网络开销和为超低延时的牺牲令网络利用率无法达到最优	极好的拥塞消除性能和带宽收敛性能	在多瓶颈或动态场景下, 瞬态性能下降; 公平性收敛慢或错误收敛	数据中心网络、无线网络
预约调度式 (FastPass, pHost, ExpressPass, NDP, Homa, Aeolus)	通常流的平均延时很短; 在简单的场景下吞吐量高	在复杂的场景如多瓶颈网络、动态性网络会存在带宽浪费	拥塞消除能力很好	公平性取决于流的调度策略; 收敛时间短, 但带宽常未能得到充分利用	数据中心网络
基于机器学习式 (PCC, PCC Vivace/ Proteus, ADPG, Ketabi等人, Indigo, Orca, ACC)	依据效用函数或奖励函数, 吞吐量通常较高、流的平均延时短	错误收敛的问题严重; 稳态性能和学习结果有关	在多种不同的网络中可以达到好的拥塞消除效果	公平性收敛速度与模型收敛速度相关; 瞬态性能与学习结果有关	数据中心网络、无线网络、因特网

4.1.1 迭代探测式

迭代探测式算法的稳态性能由拥塞探测机制决定, 通过设置拥塞标志点可以权衡高吞吐量与低延时这两个相冲突的目标. 而瞬态性能则主要由速率迭代机制决定, 使用改进的 AIMD、MIMD 或 AIAD 等权衡公平性和收敛性.

● 瞬态性能: 在拥塞消除方面, 迭代探测式拥塞控制算法主要采用试探的方式降低发送速率以消除拥塞. 主要采用 MD、改进的 AD 和减速到接收速率的方式进行拥塞消除. 其中 MD 消除拥塞的速度最快, 但在缓冲区占用较小时, 容易过度减速, 造成网络利用率降低. 相反, 改进的 AD 虽然通过加速因子不断扩大减速步伐以加快消除拥塞, 但是加速因子的更新间隔通常是 RTT 级别的. 当拥塞情况严重时, 改进的 AD 通常需要多个 RTT 来更新加速因子, 进而较缓慢地消除拥塞. 最后一种减速到接收速率的方式理论上能够快速准确地消除拥塞, 但是经常会面临测量接收速率过大, 无法正确消除拥塞的问题.

在带宽收敛方面, 基于迭代探测的拥塞控制算法通过试探性地增加发送速率来探索可用带宽. 由于正确的目标速率未知, 故此类算法的带宽收敛时间都较长, 需要多个 RTT 进行探测.

在公平性收敛方面, 部分迭代探测式算法采用 AIMD 的调速方式令系统收敛到公平, 能够实现较快的收敛. 而另一部分则依赖于为每条数据流设置独立且相同目标, 使数据流在共同的调速规则下, 都能自然地收敛至公平, 其收敛性相对于前一种方案更好.

总的来说, 迭代探测式的算法具有良好的公平性, 但其拥塞消除能力较差, 带宽收敛也较慢.

● 稳态性能: 迭代探测式的拥塞控制算法均将拥塞标志点设置在略大于 BDP 处, 以保证高网络利用率和低延时. 但感知系统是否达到拥塞标志点的方案不尽相同. 一些算法采用端到端的信号感知拥塞标志, 如接收速率、延时等. 这些算法要么容易错误感知拥塞标志, 导致系统稳定在错误的点; 要么其拥塞标志点随数据流数目变化, 导致拥塞标志点可能变得不适宜, 最终造成系统稳态时队列堆积、延时增大. 另外一些算法则利用交换机进行显式拥塞通告, 虽然能够更准确地感知拥塞标志点, 但由于需要交换机支持, 其适应性和可扩展性都受到了一定的限制.

● 适用环境: 迭代探测式拥塞控制算法能够有效地降低传输延时, 并保证较高的吞吐量. 虽然整体性能不算突出, 但算法思想结构简单, 硬件需求较低, 易于部署, 适用于大部分网络场景, 如因特网、数据中心网络和无线网络等.

4.1.2 直接测量式

直接测量式通过直接测量可用带宽, 配合探测到的瓶颈链路流数或网络带宽等网络信息指导调速以收敛到公平.

● 瞬态性能: 在消除拥塞和带宽收敛方面, 通过直接测量的方式, 能快速、精确地感知网络状况, 并通过计算直接把源端的发送速率或发送窗口调节至合适的位置, 快速消除拥塞或获取带宽, 通常在一个 RTT 内就可以充分利用空闲出来的网络带宽. 然而, 不同测量方式下, 测量不准的问题不可避免. 比如, 利用 INT 测量交换机信息时, 在多网络瓶颈的状况下瓶颈网络带宽和链路容量难以直接测量, 需要多轮迭代完成; 同样地, 当测量数据流数目时, 选取的测量时隙过大或过小时, 会导致计数的有效流数不准确. 这是由于对网络拥塞程度的判断不足, 需要通过几轮迭代才能够消除累积的缓存队列.

在公平性收敛方面, 收敛性能由速率分配方式决定. 基于直接测量的拥塞控制算法主要有两种速率分配方式. (1) 直接测量网络总资源, 根据测量到的数据流数目或计算的有效流数将网络资源平均分配给竞争流. 这种方式能够较快地收敛到公平, 但容易因为测量不准的问题造成带宽利用率不足. (2) 不直接公平分配资源, 而是让所有发送端根据测量到的网络信息进行调速以实现共同的调速目标. 这种方式收敛速度依赖于调速机制. 通常采用的 MIMD+AI 方式, 收敛到公平需要进行多轮迭代, 因此收敛较慢.

● 稳态性能: 在稳态性能方面, 通过精确感知网络状况和调节窗口, 直接测量式拥塞控制在稳定状态下可以保证系统维持在 BDP 附近, 保证了超低延时. 然而, 直接测量式的算法往往需要更多字节数的数据包头来记录交换机处测量到的数据. 这给网络带来了额外的负担, 并牺牲了网络利用率.

● 适用环境: 直接测量式拥塞控制适用于具有可控制性的网络, 如数据中心网络, 可以通过控制交换机改变其功能来进行带宽、流数等的测量; 或者无线蜂窝网络, 可以利用控制信道来进行网络信息的测量.

4.1.3 预约调度式

预约调度式通过特定的网络资源预约机制以及流调度方法来达到接近零的交换机队列, 并极大地减少流的平均完成时间.

● 瞬态性能: 在消除拥塞方面, 该类拥塞控制算法根据预约的信令等指定发送速率, 使得网络中不会出现拥塞,

因而无需消除拥塞, 且能达到极低的传输延时.

在带宽收敛方面, 预约调度式的拥塞控制通过 1–2 个 RTT 的时间来进行带宽预约, 将资源直接分配给流调度算法预发送出的流, 因此收敛速度快. 但是面对网络环境变化时, 这种拥塞控制机制容易因不合理的资源分配达不到带宽收敛.

在公平性收敛方面, 由于预约调度式的拥塞控制采用流调度策略来决定流的发送顺序. 例如, 当接收端采用最短流优先策略时, 短流会首先被发送完毕. 但是, 这类方法常牺牲流之间的公平性来达到更低的平均流完成时间.

● 稳态性能: 此类拥塞控制方法在稳态时可以使每条流的速率保持在一个稳定值附近, 较好地利用带宽. 特别在简单的网络场景, 如全对分网络, 该方法吞吐量高, 且能达到低延时. 但是, 当面对复杂的场景如多瓶颈网络、带宽动态变化的网络, 预约调度式方法会存在带宽浪费的情况.

● 适用环境: 预约调度式的拥塞控制算法常需要交换机功能支持, 因此适合部署在局部网络中, 如数据中心网络. 在这类网络中, 所有交换机以及发送端、接收端都由同一公司拥有, 均可以被修改. 在因特网中, 已有的交换机分属于不同实体, 几乎不可能被更改, 因而难以部署预约调度式拥塞控制算法.

4.1.4 基于机器学习式

基于机器学习式采用机器学习方法构建模型指导发送速率调节, 增强拥塞控制算法对网络环境的适应性.

● 瞬态性能: 在拥塞消除方面, 基于机器学习的算法同样通过迭代试探的方式感知拥塞和调节速率, 其拥塞消除性能取决于先前学习的模型精度以及近期网络环境变化模式是否能被模型捕捉.

在带宽收敛方面, 基于机器学习的拥塞控制算法的带宽收敛性能同样取决于先前学习的模型精度以及近期网络环境变化模式是否能被模型捕捉. 一些算法通过自信放大器加速获取带宽. 但是, 由于基于机器学习算法本身的收敛问题[19], 其带宽收敛性能难以得到保证.

在公平性收敛方面, 完全基于机器学习的方案依赖于数据流的效用函数达到纳什均衡, 其收敛速度直接取决于模型收敛的速度. 而结合机器学习和经典拥塞控制的算法则具有良好公平性, 这主要得益于其底层经典控制方案能保证公平的调速.

● 稳态性能: 基于机器学习的算法的稳态性能取决于其效用函数或奖励函数的设置, 一般奖励利用率和惩罚延迟. 这些算法通过基于机器学习的黑盒方法进行调速, 从而最大化效用函数或奖励函数. 一般在效用函数或奖励函数最大时, 也会同时满足高利用率和低延时. 但是, 纯粹基于机器学习的算法大都面临难以收敛到稳态或者错误收敛 (即收敛到局部最优) 的问题[19].

● 适用环境: 基于机器学习的拥塞控制算法不需要依赖于对网络模型的假设, 因此具备良好的自适应性. 其中, 基于离线学习的拥塞控制算法的自适应性受制于离线训练的网络环境; 而基于在线学习的拥塞控制算法采集最新网络信息实时更新模型, 其有潜力自适应所有的网络环境.

4.2 不同网络下的定量对比分析

在数据中心网络中, 直接测量式和预约调度式的方案由于获取了准确网络资源信息, 因而它们的性能表现往往明显优于迭代探测式算法. 例如, 直接测量式的 PowerTCP 的实验结果表明相比 DCTCP 和 TIMELY, 它减少了短流 80% 的尾部延时; 相比 HPCC, 则可减少 33%; 预约调度式的 NDP 则在大规模的多对一场景下低于迭代探测式的 DCTCP 66.7%–75% 的的流量完成时间, 并优于 DCQCN. 在迭代探测式方案中, 基于延时方案对拥塞信息的感知更加细粒度, 所以其性能往往优于基于丢包和基于 ECN 的方案. 如 Swift 在实际的集群中, 相比 DCTCP, 可减少 90% 以上的包丢失率. 机器学习式算法的性能则很依赖于训练场景、测试场景, 如 ADPG 只在模拟场景下进行了测试.

在无线网络中, 有交换机辅助的迭代探测式和直接测量式方案性能更优. 比如 ABC 相比 BBR 可降低 54.5% 的延时, PBE-CC 比 BBR 提高了 6.3% 的平均吞吐量, 减少了 44.5% 的 95 分位数延时. 这是由于它们利用交换机得到了更准确的网络信息. 基于机器学习式算法 Orca 则在蜂窝网场景下降低了 BBR 50% 以上的平均延时, 但是吞吐率比 BBR 下降了 10% 以上.

在因特网下, 主要采用迭代探测式和基于机器学习式的方案. 其中, 迭代探测式的 Copa 算法基于延时这种细

粒度拥塞信号实现了同类算法中较优越的性能. Copa 相比 Cubic 和 BBR 达到了同等的吞吐量, 但排队延时降低了 50%–90%. 而基于机器学习式的拥塞控制算法性能则不太稳定, 受测试环境影响较大, 例如 PCC Vivace 在真实链路测试中, 实现了高于迭代探测式算法 BBR 和 CUBIC 11.6% 和 3.7 倍的吞吐率, 而在 Indigo 作者进行的同类测试中则分别实现了低于 BBR 和 Cubic 6.5% 和 6.0% 的吞吐率.

5 总结和展望

5.1 总　结

本文主要综述了近 5 年来发表于网络领域中顶级会议上的拥塞控制算法, 关注数据中心网络、无线网络和因特网 3 种类型的网络. 根据拥塞控制算法的特点, 本文将它们分为迭代探测式、直接测量式、预约调度式、基于机器学习式的算法 4 类, 并对每一类的典型方案进行了介绍; 更进一步, 我们对比和分析了这些方案在拥塞消除、带宽收敛和公平性收敛等瞬态性能、稳态性能以及相应的部署开销; 最后我们在第 5.2 节展望拥塞控制算法中发展趋势及存在的研究机会. 本文有助于全面了解本领域的近期进展, 促进相关研究.

5.2 展　望

从本文对典型方案的综述可以发现, 为达到更好的传输性能, 4 类方案的设计上在不断发展. 迭代探测式在各种网络中对拥塞的检测上不再以丢包这种二元信号作为拥塞标志, 而是考虑更复杂的信号, 如 BDP、RTT 等. 在迭代探测幅度上, 考虑了除 AIMD 外的其他方式, 比如 MIMD、AIAD. 直接测量式仍以交换机或路由器进行网络信息测量, 但当前更多地利用交换机或路由器已有的功能完成该工作, 特别是利用了最新的 INT 技术, 以利于部署, 这也是该类方法的发展趋势之一. 预约调度式的主要发展趋势是降低预约开销, 比如在首 RTT 内即发送数据包, 并通过设计各种机制更好地利用网络带宽, 以增强性能. 机器学习式的方案一方面是采用更新的机器学习算法, 比如 PPO, 进行拥塞控制速率或参数优化; 另一方面则是设计新机制来加快机器学习算法的收敛速度, 比如利用已有的拥塞控制算法提供的信息.

由于 4 类方案在设计上明显的特点, 它们在不同网络中的发展前景并不一样. 从表 8 中可以看到, 在数据中心网络, 4 类方案均有最新的成果出现; 在无线网络, 则主要是迭代探测式和直接测量式; 在因特网, 迭代探测式和基于机器学习式的方案会有进一步发展.

尽管最新的研究提出了许多性能优越的拥塞控制方法, 但是在各种场景下, 拥塞控制仍然面临不少问题. 例如: 高带宽场景下的带宽收敛速度问题、主机内部拥塞如何控制的问题等. 此外, 随着新技术的出现——包括数百 Gb/s 的高速带宽、智能网卡、可编程交换机、全虚拟技术等——存在新的机会来设计拥塞控制机制. 下面, 我们探讨了这样的挑战和机会.

● 高带宽场景下达到低延时和高吞吐量: 低延时和高吞吐量这两个拥塞控制的主要性能指标往往是矛盾的. 这种矛盾关系在带宽变得很高时, 如数百 Gb/s 的场景下, 变得更加明显. 当链路的利用率很高时, 每秒通过交换机的数据包数很多, 如果网络发生拥塞, 则交换机处很容易产生很大的排队队列, 导致排队延时急剧增大. 如果快速降低速度, 会使交换机队列迅速排空, 这导致链路利用率很低的问题. 因此, 在高带宽场景下, 如何使延时和吞吐量两者之间达到较好的折中需要进一步研究.

● 主机端延时控制: 在数据中心场景下, 主机端延时对总延时的影响有时占很大的比例. 例如, 在虚拟环境和计算密集性任务中, 主机处理数据的延时可能大于网络传输延时. 因此, 主机端的处理延时应该在拥塞控制算法设计中予以考虑. 已有的工作 Swift 虽然进行了初步的探讨, 但是它只是简单地将主机端延时控制目标设置为固定值, 并没有考虑不同环境和任务对主机端延时的影响. 因此, 如何设置合理的主机端延时目标, 并实现恰当的控制方案是未来的一个研究方向.

● 减少在线学习拥塞控制的收敛时间: 基于机器学习的拥塞控制方法需要大量的数据来进行训练, 以得到最优的模型指导速率调节. 在基于离线学习的拥塞控制方法中, 可以预先收集数据再进行训练; 但是基于在线学习的拥塞控制算法需要在 1 个或者几个 RTT 后才能得到优化后的速率调节动作. 这要求基于在线学习的拥塞控制方

法能够根据少数的样本快速地训练得到动作结果. 现有的深度学习或强化学习算法均不能满足这样的要求. 因此, 如何减少在线学习拥塞控制的收敛时间是一个待研究的问题.

● 协调拥塞控制与负载均衡机制: 拥塞控制机制与负载均衡机制均可以提升网络的利用率. 但是, 目前负载均衡机制主要采用网络层的方法, 比如 ECMP 是根据一条流的五元组进行哈希后的结果来选路, 没有考虑选路后对传输层协议的影响; 另一方面, 尽管部分传输层的拥塞控制协议, 如 MPTCP[61], 将选路与拥塞控制算法结合起来进行设计, 但是并不能很好地达到均衡利用网络链路的目标. 因此, 如何协调拥塞控制与负载均衡机制, 提升网络传输性能需要更深入的研究.

● 利用和适用新硬件设备: 未来的网络中将部署更多新的硬件设备, 比如 P4 可编程交换机、光交换机、更多核的 CPU 等. 相比传统的交换机, 可编程交换机允许更加灵活的数据包处理; 光交换机则交换速率更高, 但存在电路重配置延迟问题; 多核 CPU 可使得一些数据包处理的操作并行, 从而加快数据传输. 利用和适用这些新硬件设备, 将是新拥塞控制算法需要考虑的.

致谢 衷心感谢评审专家和编辑们对本文提出的宝贵意见和建议!

References:

[1] China Internet Network Information Center. The 49th statistical report on the development of Internet in China. Beijing: China Internet Network Information Center, 2022 (in Chinese). https://www.cnnic.com.cn/IDR/ReportDownloads/202204/P020220424336135612575.pdf

[2] Li YL, Miao R, Liu HH, Zhang Y, Feng F, Tang LB, Cao Z, Zhang M, Kelly F, Alizadeh M, Yu ML. HPCC: High precision congestion control. In: Proc. of the 2019 ACM Special Interest Group on Data Communication. Beijing: ACM, 2019. 44–58. [doi: 10.1145/3341302.3342085]

[3] Cerf V, Jacobson V, Weaver N, Gettys J. BufferBloat: What's wrong with the Internet? A discussion with Vint Cerf, Van Jacobson, Nick Weaver, and Jim Gettys. Queue, 2011, 9(12): 10–20. [doi: 10.1145/2076796.2076798]

[4] Gao PX, Narayan A, Kumar G, Agarwal R, Ratnasamy S. pHost: Distributed near-optimal datacenter transport over commodity network fabric. In: Proc. of the 11th ACM Conf. on Emerging Networking Experiments and Technologies. Heidelberg: ACM, 2015. 1. [doi: 10.1145/2716281.2836086]

[5] Handley M, Raiciu C, Agache A, Voinescu A, Moore AW, Antichi G, Wójcik M. Re-architecting datacenter networks and stacks for low latency and high performance. In: Proc. of the 2017 Conf. of the ACM Special Interest Group on Data Communication. Los Angeles: ACM, 2017. 29–42. [doi: 10.1145/3098822.3098825]

[6] Montazeri B, Li YL, Alizadeh M, Ousterhout J. Homa: A receiver-driven low-latency transport protocol using network priorities. In: Proc. of the 2018 Conf. of the ACM Special Interest Group on Data Communication. Budapest: ACM, 2018. 221–235. [doi: 10.1145/3230543.3230564]

[7] Zhu YB, Eran H, Firestone D, Guo CX, Lipshteyn M, Liron Y, Padhye J, Raindel S, Yahia MH, Zhang M. Congestion control for large-scale RDMA deployments. ACM SIGCOMM Computer Communication Review, 2015, 45(4): 523–536. [doi: 10.1145/2829988.2787484]

[8] Cho I, Jang K, Han DS. Credit-scheduled delay-bounded congestion control for datacenters. In: Proc. of the 2017 Conf. of the ACM Special Interest Group on Data Communication. Los Angeles: ACM, 2017. 239–252. [doi: 10.1145/3098822.3098840]

[9] Mittal R, Lam VT, Dukkipati N, Blem E, Wassel H, Ghobadi M, Vahdat A, Wang YG, Wetherall D, Zats D. TIMELY: RTT-based congestion control for the datacenter. ACM SIGCOMM Computer Communication Review, 2015, 45(4): 537–550. [doi: 10.1145/2829988.2787510]

[10] Kumar G, Dukkipati N, Jang K, Wassel HMG, Wu X, Montazeri B, Wang YG, Springborn K, Alfeld C, Ryan M, Wetherall D, Vahdat A. Swift: Delay is simple and effective for congestion control in the datacenter. In: Proc. of the 2020 Annual Conf. ACM Special Interest Group on Data Communication on the Applications, Technologies, Architectures, and Protocols for Computer Communication. ACM, 2020. 514–528. [doi: 10.1145/3387514.3406591]

[11] Chen G, Lu YW, Li BJ, Tan K, Xiong YQ, Cheng P, Zhang JS, Moscibroda T. MP-RDMA: Enabling RDMA with multi-path transport in datacenters. IEEE/ACM Trans. on Networking, 2019, 27(6): 2308–2323. [doi: 10.1109/TNET.2019.2948917]

[12] Cardwell N, Cheng YC, Gunn CS, Yeganeh SH, Jacobson V. BBR: Congestion-based congestion control. Communications of the ACM,

2017, 60(2): 58–66. [doi: 10.1145/3009824]

[13] Arun V, Balakrishnan H. Copa: Practical delay-based congestion control for the Internet. In: Proc. of the 15th USENIX Symp. on Networked Systems Design and Implementation. Renton: USENIX Association, 2018. 329–342.

[14] Winstein K, Balakrishnan H. TCP ex machina: Computer-generated congestion control. ACM SIGCOMM Computer Communication Review, 2013, 43(4): 123–134. [doi: 10.1145/2534169.2486020]

[15] Dong M, Li QX, Zarchy D, Godfrey PB, Schapira M. PCC: Re-architecting congestion control for consistent high performance. In: Proc. of the 12th USENIX Symp. on Networked Systems Design and Implementation. Oakland: USENIX Association, 2015. 395–408.

[16] Dong M, Meng T, Zarchy D, Arslan E, Gilad Y, Godfrey PB, Schapira M. PCC Vivace: Online-learning congestion control. In: Proc. of the 15th USENIX Symp. on Networked Systems Design and Implementation. Renton: USENIX Association, 2018. 343–356.

[17] Meng T, Schiff NR, Godfrey PB, Schapira M. PCC Proteus: Scavenger transport and beyond. In: Proc. of the 2020 ACM Special Interest Group on Data Communication on the Applications, Technologies, Architectures, and Protocols for Computer Communication. ACM, 2020. 615–631. [doi: 10.1145/3387514.3405891]

[18] Yan FY, Ma J, Hill GD, Raghavan D, Wahby RS, Levis PA, Winstein K. Pantheon: The training ground for Internet congestion-control research. In: Proc. of the 2018 USENIX Annual Technical Conf. Boston: USENIX Association, 2018. 731–743.

[19] Abbasloo S, Yen CY, Chao HJ. Classic meets modern: A pragmatic learning-based congestion control for the Internet. In: Proc. of the 2020 ACM Special Interest Group on Data Communication on the Applications, Technologies, Architectures, and Protocols for Computer Communication. ACM, 2020. 632–647. [doi: 10.1145/3387514.3405892]

[20] Carlucci G, De Cicco L, Holmer S, Mascolo S. Analysis and design of the Google congestion control for Web real-time communication (WebRTC). In: Proc. of the 7th Int'l Conf. on Multimedia Systems. Klagenfurt: ACM, 2016. 13. [doi: 10.1145/2910017.2910605]

[21] Zaki Y, Pötsch T, Chen J, Subramanian L, Görg C. Adaptive congestion control for unpredictable cellular networks. ACM SIGCOMM Computer Communication Review, 2015, 45(4): 509–522. [doi: 10.1145/2829988.2787498]

[22] Goyal P, Agarwal A, Netravali R, Alizadeh M, Balakrishnan H. ABC: A simple explicit congestion controller for wireless networks. In: Proc. of the 17th USENIX Symp. on Networked Systems Design and Implementation. Santa Clara: USENIX Association, 2020. 353–372.

[23] Xie YX, Yi F, Jamieson K. PBE-CC: Congestion control via endpoint-centric, physical-layer bandwidth measurements. In: Proc. of the 2020 ACM Special Interest Group on Data Communication on the Applications, Technologies, Architectures, and Protocols for Computer Communication. ACM, 2020. 451–464. [doi: 10.1145/3387514.3405880]

[24] Huang S, Dong DZ, Bai W. Congestion control in high-speed lossless data center networks: A survey. Future Generation Computer Systems, 2018, 89: 360–374. [doi: 10.1016/j.future.2018.06.036]

[25] Guo ZH, Liu S, Zhang ZL. Traffic control for RDMA-enabled data center networks: A survey. IEEE Systems Journal, 2020, 14(1): 677–688. [doi: 10.1109/JSYST.2019.2936519]

[26] Al-Saadi R, Armitage G, But J, Branch P. A survey of delay-based and hybrid TCP congestion control algorithms. IEEE Communications Surveys & Tutorials, 2019, 21(4): 3609–3638. [doi: 10.1109/COMST.2019.2904994]

[27] Zeng GX, Hu SH, Zhang JX, Chen K. Transport protocols for data center networks: A survey. Journal of Computer Research and Development, 2020, 57(1): 74–84 (in Chinese with English abstract). [doi: 10.7544/issn1000-1239.2020.20190519]

[28] Du XL, Xu K, Li T, Zheng K, Fu ST, Shen M. Traffic control for data center network: State of the art and future research. Chinese Journal of Computers, 2021, 44(7): 1287–1309 (in Chinese with English abstract). [doi: 10.11897/SP.J.1016.2021.01287]

[29] Rojas-Cessa R, Kaymak Y, Dong ZQ. Schemes for fast transmission of flows in data center networks. IEEE Communications Surveys & Tutorials, 2015, 17(3): 1391–1422. [doi: 10.1109/COMST.2015.2427199]

[30] Perry J, Ousterhout A, Balakrishnan H, Shah D, Fugal H. FastPass: A centralized "zero-queue" datacenter network. Proc. of the 2014 ACM Conf. on SIGCOMM, 2014, 44(4): 307–318. [doi: 10.1145/2619239.2626309]

[31] Hu SH, Bai W, Zeng GX, Wang ZL, Qiao BC, Chen K, Tan K, Wang Y. Aeolus: A building block for proactive transport in datacenters. In: Proc. of the 2020 Annual Conf. of the ACM Special Interest Group on Data Communication on the Applications, Technologies, Architectures, and Protocols for Computer Communication. ACM, 2020. 422–434. [doi: 10.1145/3387514.3405878]

[32] Yan SY, Wang XL, Zheng XL, Xia YB, Liu DR, Deng WS. ACC: Automatic ECN tuning for high-speed datacenter networks. In: Proc. of the 2021 ACM SIGCOMM Conf. ACM, 2021. 384–397. [doi: 10.1145/3452296.3472927]

[33] Meng ZL, Guo YN, Sun C, Wang B, Sherry J, Liu HH, Xu MW. Achieving consistent low latency for wireless real-time communications with the shortest control loop. In: Proc. of the 2022 ACM SIGCOMM Conf. Amsterdam: ACM, 2022. 193–206. [doi: 10.1145/3544216.3544225]

[34] Goyal P, Narayan A, Cangialosi F, Narayana S, Alizadeh M, Balakrishnan H. Elasticity detection: A building block for Internet congestion control. In: Proc. of the 2022 AACM SIGCOMM Conf. Amsterdam: ACM, 2022. 158–176. [doi: 10.1145/3544216.3544221]

[35] Addanki V, Michel O, Schmid S. PowerTCP: Pushing the performance limits of datacenter networks. In: Proc. of the 19th USENIX Symp. on Networked Systems Design and Implementation. Renton: USENIX Association, 2022. 51–70.

[36] Yen CY, Abbasloo S, Chao HJ. Computers can learn from the heuristic designs and master Internet congestion control. In: Proc. of the 2023 ACM SIGCOMM Conf. New York: ACM, 2023. 255–274. [doi: 10.1145/3603269.3604838]

[37] Agarwal S, Krishnamurthy A, Agarwal R. Host congestion control. In: Proc. of the 2023 ACM SIGCOMM Conf. New York: ACM, 2023. 275–287. [doi: 10.1145/3603269.3604878]

[38] Arslan S, Li YL, Kumar G, Dukkipati N. Bolt: Sub-RTT congestion control for ultra-low latency. In: Proc. of the 20th USENIX Symp. on Networked Systems Design and Implementation. Boston: USENIX Association, 2023. 219–236.

[39] Wang WT, Moshref M, Li YL, Kumar G, Ng TSE, Cardwell N, Dukkipati N. Poseidon: Efficient, robust, and practical datacenter CC via deployable INT. In: Proc. of the 20th USENIX Symp. on Networked Systems Design and Implementation. Boston: USENIX Association, 2023. 255–274.

[40] Alizadeh M, Greenberg A, Maltz DA, Padhye J, Patel P, Prabhakar B, Sengupta S, Sridharan M. Data center TCP (DCTCP). ACM SIGCOMM Computer Communication Review, 2010, 40(4): 63–74. [doi: 10.1145/1851275.1851192]

[41] Zhang H, Zhang JX, Bai W, Chen K, Chowdhury M. Resilient datacenter load balancing in the wild. In: Proc. of the 2017 ACM Special Interest Group on Data Communication. Los Angeles: ACM, 2017. 253–266. [doi: 10.1145/3098822.3098841]

[42] Jacobson V. Congestion avoidance and control. ACM SIGCOMM Computer Communication Review, 1988, 18(4): 314–329. [doi: 10.1145/52325.52356]

[43] Allman M, Paxson V, Stevens W. TCP congestion control. RFC2581, 1999. [doi: 10.17487/RFC2581]

[44] Floyd S, Henderson T. The NewReno modification to TCP's fast recovery algorithm. RFC2582, 1999. [doi: 10.17487/RFC2582]

[45] Mathis M, Mahdavi J, Floyd S, Romanow A. TCP selective acknowledgement options. RFC2018, 1996. [doi: 10.17487/RFC2018]

[46] Brakmo LS, Peterson LL. TCP Vegas: End to end congestion avoidance on a global Internet. IEEE Journal on Selected Areas in Communications, 1995, 13(8): 1465–1480. [doi: 10.1109/49.464716]

[47] Floyd S. HighSpeed TCP for large congestion windows. RFC3649, 2003. [doi: 10.17487/RFC3649]

[48] Kelly T. Scalable TCP: Improving performance in highspeed wide area networks. ACM SIGCOMM Computer Communication Review, 2003, 33(2): 83–91. [doi: 10.1145/956981.956989]

[49] Jin C, Wei D, Low SH, Bunn J, Choe HD, Doyle JC, Newman H, Ravot S, Singh S, Paganini F, Buhrmaster G, Cottrell L, Martin O, Feng WC. FAST TCP: From theory to experiments. IEEE Network, 2005, 19(1): 4–11. [doi: 10.1109/MNET.2005.1383434]

[50] Xu LS, Harfoush K, Rhee I. Binary increase congestion control (BIC) for fast long-distance networks. In: Proc. of 2004 Int'l Conf. on Computer Communications. Hong Kong: IEEE, 2004. 2514–2524. [doi: 10.1109/INFCOM.2004.1354672]

[51] Ha S, Rhee I, Xu LS. CUBIC: A new TCP-friendly high-speed TCP variant. ACM SIGOPS Operating Systems Review, 2008, 42(5): 64–74. [doi: 10.1145/1400097.1400105]

[52] Braden B, Clark D, Crowcroft J, Davie B, Deering S, Estrin D, Floyd S, Jacobson V, Minshall G, Partridge C, Peterson L, Ramakrishnan K, Shenker S, Wroclawski J, Zhang L. Recommendations on queue management and congestion avoidance in the Internet. RFC2309, 1998. [doi: 10.17487/RFC2309]

[53] Floyd S, Jacobson V. Random early detection gateways for congestion avoidance. IEEE/ACM Trans. on Networking, 1993, 1(4): 397–413. [doi: 10.1109/90.251892]

[54] Ramakrishnan K, Floyd S, Black D. The addition of explicit congestion notification (ECN) to IP. RFC3168, 2001. [doi: 10.17487/RFC3168]

[55] Ren FY, Wang FB, Ren Y, Shan XM. PID controller for active queue management. Journal of Electronics & Information Technology, 2003, 25(1): 94–99 (in Chinese with English abstract).

[56] Xu X, Tang K, Ma YF. Congestion control scheme on wireless loss-prone links. Journal of China Institute of Communications, 2004, 25(12): 8–13 (in Chinese with English abstract). [doi: 10.3321/j.issn:1000-436X.2004.12.002]

[57] Yang JW, Gu DY, Zhang WD. An analytical design method of PID controller based on AQM/ARQ. Ruan Jian Xue Bao/Journal of Software, 2006, 17(9): 1989–1995 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/17/1989.htm [doi: 10.1360/jos171989]

[58] Wu QL, Tao J, Yao J. An active queue management algorithm based on predictable PI controller in self-similar network. Acta Electronica Sinica, 2006, 34(5): 938–943 (in Chinese with English abstract). [doi: 10.3321/j.issn:0372-2112.2006.05.035]

[59] Hwang J, Yoo J, Lee SH, Jin HW . Scalable congestion control protocol based on SDN in data center networks. In: Proc. of the 2015 IEEE Global Communications Conf. (GLOBECOM). San Diego: IEEE, 2015. 1–6. [doi: 10.1109/GLOCOM.2015.7417067]

[60] Jouet S, Perkins C, Pezaros D. OTCP: SDN-managed congestion control for data center networks. In: Proc. of the 2016 IEEE/IFIP

Network Operations and Management Symp. Istanbul: IEEE, 2016. 171–179. [doi: 10.1109/NOMS.2016.7502810]

[61] Raiciu C, Barre S, Pluntke C, Greenhalgh A, Wischik D, Handley M. Improving datacenter performance and robustness with multipath TCP. ACM SIGCOMM Computer Communication Review, 2011, 41(4): 266–277. [doi: 10.1145/2043164.2018467]

[62] Nakayama Y. Rate-based path selection for shortest path bridging in access networks. In: Proc. of the 2014 IEEE Int'l Conf. on Communications. Sydney: IEEE, 2014. 1266–1271. [doi: 10.1109/ICC.2014.6883495]

[63] Allan D, Farkas J, Mansfield S. Intelligent load balancing for shortest path bridging. IEEE Communications Magazine, 2012, 50(7): 163–167. [doi: 10.1109/MCOM.2012.6231293]

[64] Carofiglio G, Gallo M, Muscariello L. ICP: Design and evaluation of an Interest control protocol for content-centric networking. In: Proc. of the 2012 IEEE INFOCOM Workshops. Orlando: IEEE, 2012. 304–309. [doi: 10.1109/INFCOMW.2012.6193510]

[65] Saino L, Cocora C, Pavlou G. CCTCP: A scalable receiver-driven congestion control protocol for content centric networking. In: Proc.of the 2013 IEEE Int'l Conf. on Communications (ICC). Budapest: IEEE, 2013. 3775–3780. [doi: 10.1109/ICC.2013.6655143]

[66] Ren YM, Li J, Shi SS, Li LL, Chang XQ. An interest control protocol for named data networking based on explicit feedback. In: Proc. of the 2015 ACM/IEEE Symp. on Architectures for Networking and Communications Systems. Oakland: IEEE, 2015. 199–200. [doi: 10.1109/ANCS.2015.7110139]

[67] Rozhnova N, Fdida S. An effective hop-by-hop Interest shaping mechanism for CCN communications. In: Proc. of the 2012 IEEE INFOCOM Workshops. Orlando: IEEE, 2012. 322–327. [doi: 10.1109/INFCOMW.2012.6193514]

[68] Carofiglio G, Gallo M, Muscariello L. Joint hop-by-hop and receiver-driven interest control protocol for content-centric networks. ACM SIGCOMM Computer Communication Review, 2012, 42(4): 491–496. [doi: 10.1145/2377677.2377772]

[69] Katabi D, Handley M, Rohrs C. Congestion control for high bandwidth-delay product networks. ACM SIGCOMM Computer Communication Review, 2002, 32(4): 89–102. [doi: 10.1145/964725.633035]

[70] Dukkipati N. Rate Control Protocol (RCP): Congestion control to make flows complete quickly [Ph.D. Thesis]. Stanford: Stanford University, 2008.

[71] Zhang J, Ren FY, Shu R, Cheng P. TFC: Token flow control in data center networks. In: Proc. of the 11th European Conf. on Computer Systems. London: ACM, 2016. 23. [doi: 10.1145/2901318.2901336]

[72] Jay N, Rotman N, Godfrey B, Schapira M, Tamar A. A deep reinforcement learning perspective on Internet congestion control. In: Proc. of the 36th Int'l Conf. on Machine Learning. Long Beach: PMLR, 2019. 3050–3059.

[73] Emara S, Li BC, Chen YJ. Eagle: Refining congestion control by learning from the experts. In: Proc. of the 29th IEEE Conf. on Computer Communications. Toronto: IEEE, 2020. 676–685. [doi: 10.1109/INFOCOM41043.2020.9155250]

[74] Du ZX, Zheng JQ, Yu HB, Kong LT, Chen GH. A unified congestion control framework for diverse application preferences and network conditions. In: Proc. of the 17th Int'l Conf. on Emerging Networking Experiments and Technologies. ACM, 2021. 282–296. [doi: 10.1145/3485983.3494840]

[75] Tessler C, Shpigelman Y, Dalal G, Mandelbaum A, Kazakov DH, Fuhrer B, Chechik G, Mannor S. Reinforcement learning for datacenter congestion control. ACM SIGMETRICS Performance Evaluation Review, 2021, 49(2): 43–46. [doi: 10.1145/3512798.3512815]

[76] Ketabi S, Chen HK, Dong HW, Ganjali Y. A deep reinforcement learning framework for optimizing congestion control in data centers. In: Proc. of the 36th IEEE/IFIP Network Operations and Management Symp. Miami: IEEE, 2023. 1–7. [doi: 10.1109/NOMS56928.2023.10154411]

[77] Tan K, Song J, Zhang Q, Sridharan M. A compound TCP approach for high-speed and long distance networks. In: Proc. of the 25th IEEE Int'l Conf. on Computer Communications. Barcelona: IEEE, 2006. 1–12. [doi: 10.1109/INFOCOM.2006.188]

[78] Hock M, Bless R, Zitterbart M. Experimental evaluation of BBR congestion control. In: Proc. of 25th Int'l Conf. on Network Protocols. Toronto: IEEE, 2017. 1–10. [doi: 10.1109/ICNP.2017.8117540]

[79] Pan R, Prabhakar B, Laxmikantha A. QCN: Quantized congestion notification an overview. 2007. https://www.ieee802.org/1/files/public/docs2007/au_prabhakar_qcn_overview_geneva.pdf

[80] Teymoori P, Welzl M. LGCC: Food chain multi-hop congestion control. Technical Report 949. Oslo: University of Oslo, 2020. 1–16.

附中文参考文献:

[1] 中国互联网络信息中心. 第 49 次中国互联网络发展状况统计报告. 北京: 中国互联网络信息中心, 2022. https://www.cnnic.com.cn/IDR/ReportDownloads/202204/P020220424336135612575.pdf

[27] 曾高雄, 胡水海, 张骏雪, 陈凯. 数据中心网络传输协议综述. 计算机研究与发展, 2020, 57(1): 74–84. [doi: 10.7544/issn1000-1239.2020.20190519]

[28] 杜鑫乐, 徐恪, 李彤, 郑凯, 付松涛, 沈蒙. 数据中心网络的流量控制: 研究现状与趋势. 计算机学报, 2021, 44(7): 1287–1309. [doi: 10.11897/SP.J.1016.2021.01287]
[55] 任丰原, 王福豹, 任勇, 山秀明. 主动队列管理中的 PID 控制器. 电子与信息学报, 2003, 25(1): 94–99.
[56] 续欣, 汤凯, 马刈非. 无线误码信道上的拥塞控制策略. 通信学报, 2004, 25(12): 8–13. [doi: 10.3321/j.issn:1000-436X.2004.12.002]
[57] 杨吉文, 顾诞英, 张卫东. 主动队列管理中 PID 控制器的解析设计方法. 软件学报, 2006, 17(9): 1989–1995. http://www.jos.org.cn/1000-9825/17/1989.htm [doi: 10.1360/jos171989]
[58] 吴清亮, 陶军, 姚婕. 一种基于预测 PI 控制器的自相似网络主动队列管理算法. 电子学报, 2006, 34(5): 938–943. [doi: 10.3321/j.issn:0372-2112.2006.05.035]

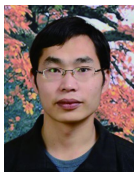
蒋万春(1986－), 男, 博士, 副教授, 博士生导师, CCF 专业会员, 主要研究领域为网络传输协议分析和设计, 流媒体传输优化, 基于机器学习的计算机网络优化.

王洁(1997－), 女, 硕士生, 主要研究领域为网络传输协议分析与设计.

李昊阳(1997－), 男, 博士生, 主要研究领域为网络传输协议分析与设计.

王建新(1969－), 男, 博士, 教授, 博士生导师 , CCF 高级会员, 主要研究领域为计算机算法与优化, 网络优化, 生物信息学.

陈晗瑜(1996－), 女, 硕士生, 主要研究领域为网络传输协议分析与设计.



阮昌(1986－), 男, 博士, 讲师, CCF 专业会员, 主要研究领域为数据中心网络, 基于机器学习的网络优化.