计算机研究与发展
Journal of Computer Research and Development
DOI:10.7544/issn1000-1239.2020.20190519
57(1):74-84,2020

数据中心网络传输协议综述

曾高雄　胡水海　张骏雪　陈　凯
(香港科技大学计算机科学与工程系　香港　999077)
(kaichen@cse.ust.hk)

Transport Protocols for Data Center Networks: A Survey

Zeng Gaoxiong, Hu Shuihai, Zhang Junxue, and Chen Kai
(*Department of Computer Science and Engineering, Hong Kong University of Science and Technology, Hong Kong* 999077)

Abstract Driven by the need of prevailing Web applications and services (e.g., search, online retailing, and cloud computing), data centers (DCs) have been built at an unforeseen rate and scale around the globe in the recent decades. In particular, data center networks (DCNs) have drawn great attention from both academia and industry. Under such a background, this paper surveys one of the key aspects of DCN—transport layer protocol. While transport protocol has a long history on the Internet, it is seldom systematically explored in the DCN context until 2010s. DCN presents different characteristics (e.g., single administrative domain, and homogeneous network structure) from the Internet. This brings about both opportunities and challenges for transport protocol design over it. Motivated by this, a bunch of transport protocols have been proposed. This paper classifies the early work (2010—2015) on DCN transport design into three categories—endhost-based congestion control, switch-assisted arbitration, and network priority scheduling. The paper discusses the pros and cons of the work from the three categories. At last, the paper analyzes the recent research trend on DCN transport design—receiver-driven proactive congestion control, and RDMA (remote direct memory access) transport design.

Key words data center network; transport protocol; congestion control; explicit congestion notification (ECN); remote direct memory access (RDMA)

摘　要 近10年来,在盛行的网络应用(如搜索、在线零售和云计算等)的需求驱动下,数据中心在全球范围内以前所未有的速度和规模发展建立起来.特别地,数据中心网络引起了学术界和工业界的广泛关注.在这样的背景下,调研了数据中心网络的一个核心方面——传输层协议.虽然传输协议在因特网上已经有很长的历史,它却直到2010年才在数据中心网络环境下被系统性地探索.数据中心网络有着和因特网不一样的特点(如单一控制域和同构网络架构),这给数据中心网络上的传输协议设计同时带来了机遇和挑战.在这驱使下,一系列的传输协议被设计提出.将早期(2010—2015年)数据中心网络传输设计方面的工作分成3类——基于端主机的拥塞控制、网络仲裁机制和交换机优先级调度,对这3类工作的优缺点作深入讨论.最后,分析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.

关键词 数据中心网络;传输层协议;拥塞控制;显式拥塞通告;远程直接内存访问

中图法分类号 TP393

收稿日期:2019-08-12;**修回日期**:2019-11-13

近10年来,网络应用(如搜索、在线零售和云计算等)飞速发展,对底层基础设施在计算、存储和网络等方面提出了严苛的要求.在此驱动下,微软、谷歌、亚马逊、脸书、阿里巴巴等大型互联网企业在全球范围内快速地建立起了高性能数据中心(data center, DC).这些数据中心通常采用商用器件,将服务器和交换机通过精心设计的网络互联,从而以更经济更便捷的方式达到高速计算和海量存储等需求.在支撑服务的各项技术中,数据中心网络(data center network, DCN)是一个潜在且重要的性能瓶颈[1-3],因而引起了包括学术界和工业界的广泛关注.

在这样的背景下,本文调研了数据中心网络的一个核心方面——传输层协议.本文首先概述了数据中心网络环境下传输层协议设计的机遇和挑战.在这驱使下,一系列的传输协议被设计提出.随后将早期(2010—2015年)数据中心网络传输设计方面的工作进行分类,并对各自的优缺点作了深入探讨.最后分析了2015年以来数据中心网络传输设计的研究趋势.

1 数据中心网络传输层协议概述

传输层协议旨在为应用提供高吞吐、低延迟的网络数据传输服务.虽然传输协议在因特网上已经有很长的历史了,它却直到2010年才在数据中心网络环境下被系统性地探索.数据中心网络有着和因特网不一样的特点(如单一控制域和同构网络架构).这给数据中心网络上的传输协议设计同时带来了机遇和挑战.本节将首先对因特网下的传输层协议作简单介绍;之后,分析数据中心网络的特点,并指出其对传输协议设计带来的机遇和挑战.

1.1 因特网下的传输层协议简介

传输层协议在因特网下已经发展了几十年.传输层协议包括:无连接的尽力传输协议,以用户数据报协议(user datagram protocol, UDP[4])为代表;面向连接的可靠传输协议,以传输控制协议(transport control protocol, TCP[5-6])为代表.在大多数网络场景(包括因特网和数据中心网络)中,应用层使用的传输协议以面向连接的可靠传输协议为主.TCP传输层协议提供的服务包括可靠性、流量控制、拥塞避免、多路复用等.其中,拥塞控制方面的研究最为广泛.

TCP最经典的拥塞控制算法是由Jacobson[7]在1988年的SIGCOMM会议上提出的TCP Tahoe.该算法包括慢启动(slow start)和拥塞避免(congestion avoidance)两部分.新建立的连接首先进入慢启动阶段,每个来回通信时间(round-trip time, RTT)拥塞窗口(congestion window, CWND)涨1倍,直到通过计时器超时(timeout)发现丢包或者窗口达到慢启动阈值(slow start threshold, ssthresh)进入拥塞避免阶段.拥塞避免阶段每个RTT拥塞窗口增加一个最大传输单元(maximum transmission unit, MTU),发现丢包则窗口降为初始值1并重新进入慢启动阶段.

TCP Tahoe算法为了避免网络丢包后因为等待超时重传而带来的时间开销,额外引入了快速重传(fast retransmission)机制,发送方收到多个相同序列号的确认包(acknowledgement, ACK)后,即认为下一个序列号对应的数据包已经丢失,并开始丢包重传.在此基础上,TCP Reno[8]增加了快速恢复(fast recovery)机制,在快速重传之初窗口降为当前值的一半,同时保持拥塞避免状态,避免了低效的初始化窗口和慢启动过程.更进一步地,TCP New Reno[9]维护了快速恢复的状态,记录快速恢复状态之初已发送的数据包,直到这些数据都已被确认才退出这一状态,只在快速恢复初期进行1次窗口砍半,避免了因为连续丢包导致的连续砍窗口的问题.最后,TCP SACK[10]加入了选择确认(selective ACK, SACK)与重传机制,避免了累计确认(cumulative ACK)与重传机制带来的低效的回退 N 步(go-back-N)问题.

1.2 数据中心网络的特点、机遇与挑战

相对于因特网这类复杂多变且不完全可控的异构网络,数据中心网络是一个单一自治域(administrative domain)下的同构网络环境.单一自治域意味着端主机(endhost)及其使用的协议、网络设备及配置,甚至应用对网络服务的需求(如延迟敏感还是带宽敏感等)等都相对可控可预测.同构性体现在固定有规律的拓扑,如Fat-tree[1],VL2[11]等;相同的网络设备及配置,如显式拥塞通告(explicit congestion notification, ECN[12]),相对一致的缓存大小和链路带宽;有限的路由条数以及可预测的RTT等.

数据中心网络的这些特点给传输协议设计带来了许多机遇,列举如下:

1) 数据中心网络下可以协调端主机协议和网络设备设置,存在更多性能优化空间.传统因特网

因为网络不可控,只能使用丢包信息作为拥塞反馈信号,不可避免地带来了高延迟和高丢包率等问题.即使网络设备使用了一些复杂的网络优化机制,端主机上的协议也很可能没能充分配合使用.数据中心网络则可以协调两者,如网络侧采用主动队列管理(active queue management, AQM[13])在拥塞早期标记 ECN 信号,接收端收到信号通过 ACK 反馈给发送端,发送端得以及时调整发送窗口或速率实现高效拥塞控制等.

2) 数据中心网络下的传输协议不受公平性的约束.因特网不是单一自治域,存在多种不同协议之间的竞争,因而公平性是一个重要指标,给网络性能方面的优化带来了约束.数据中心网络作为单一自治域,一方面可以协调内部不同应用使用不同的网络优先级队列,例如,区别对待延迟敏感和带宽敏感的应用,可以优化两者各自不同的目标;另一方面可以统一全局采用相同协议和网络配置,避免不同协议间的竞争.

3) 同构的数据中心网络更有利于协议和网络设备的参数设置.例如,相对稳定的 RTT 可以帮助设置最小重传计时器(retransmission timeout, RTO);相对稳定的网络链路带宽和 RTT 意味着可预测的网络带宽时延乘积(bandwidth delay product, BDP),可以帮助设置初始拥塞窗口大小和 ECN 标记的阈值等.

与此同时,数据中心网络面对着更加复杂严苛的应用要求,给传输协议设计带来了巨大的挑战,列举如下:

1) 应用要求严苛.一个典型的应用设计模式(网页搜索、广告筛选等的基础)是分割/聚合(partition/aggregate[2]).上层应用需求被分割成多个小的任务给多个底层工作机器(worker),底层机器完成任务后将结果返回给应用汇聚层得到最终结果.为了满足服务级协议(service-level agreement, SLA),每一轮迭代(包括计算和网络传输)都需要在 10 ms 量级的期限内完成.99.9%的尾部时延仍然会对用户带来巨大的影响(要么返回不准确的结果,要么造成长等待时间).

2) 应用需求多样.不同的应用通常会有不同的网络传输需求.有些应用要求短时间内的低时延,比如面向用户请求的短流量响应.有些应用要求长时间的稳定吞吐量,比如数据中心内部定期的海量数据更新.有些应用有限定完成期限,在期限内完成的数据才有效,快于期限完成的数据并不一定会带来更大的应用价值.

3) 应用通信模式高同步与高突发.再次以分割/聚合应用模式为例,当底层大量(如几百台)机器被同时分配任务后,可能会在很相近的时间内同步完成返回结果,造成高同步的并发流量通信模式(Incast[14]).这种高同步的并发流可以轻易地占满汇聚层机器的下行带宽,造成相应交换机网口缓存溢出而丢包,进而导致网络通信的高尾部时延.与此同时,数据中心流大小通常表现出长尾分布.比如,在一个支撑网页搜索的数据中心内,50%的流是小于 100 KB 的小流,而 80%的数据量属于前 10%的大于 10 MB 的大流.这意味着网络中有很多几个 RTT 内即可完成的突发性的小流,因而对拥塞控制的突发容忍提出了更高的要求.

2 早期研究

基于第 1 节对数据中心网络的观察,工业界和学术界开始了对数据中心网络环境下传输协议设计的研究.本节将早期(2010—2015 年)数据中心网络传输设计方面的工作分成 3 类——基于端主机的拥塞控制、网络仲裁机制和交换机优先级调度,并分别对这 3 类工作的优缺点作深入讨论.表 1 概括了本节介绍的 3 类工作及其相应的优缺点.

Table 1 Comparison Between Early (2010—2015) Work on DCN Transport Protocol

表 1 数据中心网络传输层协议早期(2010—2015 年)工作的比较

Transport Protocol	Strength	Weakness	Typical Work
Endhost-based Congestion Control	Easy to Deploy	No Strict Priority	DCTCP[2], D^{2}TCP[15], L^{2}DCT[16]
Arbitration	1) Support Strict Priority 2) Fast Convergence	1) High Scheduling Delay	D^3[17], PDQ[18], FastPass[19]
Priority Scheduling	1) Work Conserving 2) Low Scheduling Delay	1) Limited Priority Queues 2) Local Suboptimal Decision	pFabric[20], PIAS[21], Karuna[22]

2.1 基于端主机的拥塞控制

数据中心网络最经典的传输层拥塞控制算法是由斯坦福大学和微软的研究人员在2010年的SIGCOMM会议上提出的数据中心传输协议(datacenter transmission control protocol, DCTCP)[2].DCTCP是一种基于端主机的拥塞控制算法,主要包括2部分:交换机端的ECN标记和主机端的速率控制.DCTCP在这2部分与基于ECN的TCP协议都有明显不同.一方面,DCTCP使用了基于交换机瞬时队列长度的单一阈值标记算法.基于瞬时队列的标记可以对突发流带来的网络拥塞作出快速响应,使得算法有更好的突发容忍性.此外,单一阈值标记配置使用更简单,这也是基于DCN内较稳定的基础RTT的条件.另一方面,DCTCP将单个位的ECN序列转化为多个位的拥塞程度并依此调整窗口,而不像TCP那样简单粗暴地窗口砍半.这样既避免了轻微拥塞下过度响应导致的带宽浪费,又防止了严重拥塞下响应不足导致的高延迟和高丢包率.

DCTCP具体算法为:

1) 交换机基于瞬时队列长度以单一阈值对经过的数据包标记ECN.

2) 接收端将收到的数据包中的ECN信息通过ACK中的ECN-Echo反馈回发送方.

3) 发送方统计收到ACK中ECN-Echo比例的指数加权滑动平均值α.当没有ECN-Echo时,每个ACK窗口增加$1/W_{\mathrm{CWND}}$;当遇到ECN-Echo时,每个RTT窗口乘性砍$\alpha/2$.

在这之后,一系列的基于ECN的拥塞控制算法被设计出来.D^{2}TCP[15]以降低期限错失率(deadline miss rate)为目标,在DCTCP的基础上引入了流的完成期限D,并以此计算期限紧急程度$e=T/D$,其中T为预期完成时间;当数据发送方收到ECN-Echo时,每个RTT窗口乘性砍$\alpha^e/2$,使得期限更紧急的流可以更快地完成.类似地,L^{2}DCT[16]则以降低流的完成时间(flow completion time, FCT)为目标,在DCTCP的基础上引入了与发送数据量反相关的流的权值函数f;当数据发送方收到ECN-Echo时,每个RTT窗口乘性砍$\alpha^f/2$,使得更小的流可以更快地完成,从而逼近在流大小信息不可知下理论最优的最少获得服务优先(least attained service, LAS)[23]调度算法.此外,有一些工作(如ECN[24])认为只需优化调整交换机ECN机制,采用顺时出队列的方式标记,即可以在不改动端主机TCP协议的情况下达到和DCTCP相近的性能.

基于端主机的拥塞控制机制只需要改动端主机上的软件协议栈,交换机上配置简单的ECN标记,因而具有配置简单和易于实现部署的优点.不足的是,因为没有很好地结合更多的交换机支持,基于端主机的拥塞控制机制在性能上通常与理论最优存在较大差距.例如,DCTCP没有区分大小流,使得大小流同时存在于单一队列中,小流的完成时间会受到大流的排队延迟影响而变得很长.与此同时,基于端主机的拥塞控制机制需要较长的时间周期去收敛到稳定状态.

2.2 网络仲裁机制

为了克服基于端主机的拥塞控制收敛时间长等缺陷,网络仲裁机制得到了广泛关注.网络仲裁机制借助于网络中交换设备等可以观察到经过流量的需求,从而基于一定的算法计算发送窗口或速率或调整方向,通过数据包头和接收端ACK反馈回发送端,从而实现拥塞控制和流量调度.

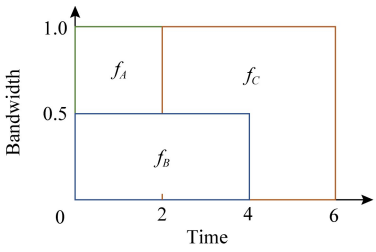
最早的代表性工作是由微软的研究人员在2011年的SIGCOMM会议上提出的期限驱动传输控制协议(deadline-driven delivery control protocol, D^3)[17].D^{3}以降低期限错失率为目标,假设上层应用可以将数据流大小和规定期限信息传给传输层.基于这些信息,D^{3}设计了网络仲裁机制,让交换机根据全局流信息分配带宽,优先分配数据流在规定期限内完成所需的最小带宽,即$r=s/D$,其中s为流的大小,D为流的完成期限;最后将剩余的带宽平均分配给所有的流.

不足的是,D^{3}需要交换机来记录流状态并分配带宽,这样复杂的功能远远超出了目前商用交换机的支持功能范围.此外,D^{3}采用了贪婪式的先到先服务的带宽分配方案,先到达的流一旦被分配带宽,将占用带宽直至最终完成,而不允许中途释放带宽给新流,即不允许抢占.这导致了非最优的流调度,如图1(b)所示,由于一个接近完成期限的流(f_A)可能会为了等待一个先到达的完成期限较远的流(f_B)而被阻滞,进而错过了规定的完成期限.

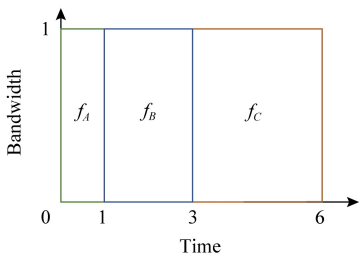
针对D^{3}的缺点,伊利诺伊大学香槟分校的研究人员在2012年的SIGCOMM会议上提出了抢占式分布快速控制协议(preemptive distributed quick control protocol, PDQ)[18].与D^{3}类似,PDQ假设上层应用可以将数据流大小和规定期限等信息暴露给传输层.利用这些信息,PDQ同样也设计了显式速率控制的实现机制,逼近不同的调度算法.例如:1)最短任务优先(shortest job first, SJF)[25]调度

Flow	Flow Size	Deadline
f_A	1	1
f_B	2	4
f_C	3	6

(a) Flow information



(b) D^3 scheduling



(c) Optimal scheduling

Fig. 1 Illustration to show the suboptimal performance of D^3 without preemption

图 1 实例说明 D^3 在没有抢占机制下的非最优调度性能

算法.数据量越小的流将会被优先分配带宽.2)最近期限优先(earliest deadline first,EDF)[26].越接近完成期限的流将会被优先分配带宽.与 D^3 不同的是,PDQ 采用了抢占式的带宽分配方案,可以根据流的紧急程度将已分配的带宽重新分配,从而解决了 D^3 非最优调度的问题(图 1(c)中最优调度与 PDQ 相同).

PDQ 离理想的 SJF 和 EDF 仍有较大的差距.这是因为在 PDQ 系统里,暂停或者启动一个流都需要至少一个 RTT 来与路径上所有的交换机协调.这一个 RTT 的额外开销对于数据中心内大量小流来说是不可接受的负担.此外.精确地计算出一个流的发送速率并不是一个简单的事情,比如一个流的发送瓶颈可能不是在网络里,而是在端主机的磁盘上.

与此同时,网络仲裁机制中还包括一类利用服务器做集中式拥塞控制调度的设计.代表性工作是由麻省理工学院的研究人员在 2014 年的 SIGCOMM 会议上提出的 FastPass[19].这类工作在发送数据前,发送端会向集中控制器发送请求,集中控制器结合请求信息和网络状况作出相应的调度指令,包括发送数据的时间、速率和路径等.这种集中式调度优点明显,因为可以获取网络全局信息,理论上可以达到最优的调度,但是也存在明显缺点,系统实现上需要额外一个 RTT 开销去获取调度决策,因而对数据中心下小流的完成时间有较大的负面影响.

2.3 交换机优先级调度

为了克服网络仲裁机制不可实现等缺陷,斯坦福大学的研究人员在 2013 年的 SIGCOMM 会议上提出了基于交换机优先级调度的 pFabric[20].pFabric 既不要求交换机配置大量内存以维持所有流的状态信息,也不要求交换机实现复杂的调度算法以计算反馈信息.pFabric 的核心思想是将流调度从速率控制中分离开来.pFabric 的设计包含 2 部分:1)端上的速率控制协议.新流以线速发出,去除任何快速丢包重传机制,直到连续多次超时才进入探索模式,定期发送最小数据包并在接收到确认时重新进入慢启动.2)交换机上的优先级调度和优先级丢包.交换机上优先调度最高优先级的数据包;当缓冲已满时,优先丢弃低优先级的数据包.

与网络仲裁机制相比,pFabric 设计更简单,同时保证了接近理论最优的性能.但是,pFabric 仍然无法在已有的商用交换机上实现.一方面,商用交换机芯片支持的优先级队列数量非常有限(大约 8 个);另一方面,商用交换机芯片并未广泛支持优先级丢包机制,一旦一个数据包进入了交换机的缓冲区,它就不会被新到来的包“挤出去”.后续工作 EPN 尝试使用 2 个优先级队列逼近 pFabric 的性能,然而需要对交换机做更多复杂的改动,因而同样无法在商用交换机上实现.

pFabric 等工作都假设我们可以从应用获得流的信息,比如流的大小.事实上,虽然数据中心是一个单独管理域,应用的流信息仍然很难获得.主要原因有 2 个:1)很多应用是一边计算一边传输的,所以当应用开始传一个流时,它自己也不知道这个流总共要传多少字节.2)对于许多可以获得流大小的应用,管理员仍然需要去修改其网络通信模块获得流大小,并将这个信息通过合适的接口传递给协议栈.考虑到数据中心有大量的应用,逐一修改这些应用获得流信息对于管理员来说无疑是巨大的负担.

基于上述观察,无信息感知的流调度工作应运而生.香港科技大学的研究人员在 2015 年的 NSDI 会议上提出了实际的无信息感知的流调度机制(practical information-agnostic flow scheduling,PIAS)[21].PIAS 采用多级反馈队列作为其调度策略.在多级反馈队列里面,所有流一开始都属于最高优先级,当一个流已经发送的字节数超过一个阈值时,这个流就被降入下一个优先级.根据多级反馈队

列策略,小流会在较高优先级内完成,而大流会在较低优先级内完成.多级反馈队列可以在长尾分布下有效地近似理想的最短剩余时间优先,降低流的整体平均完成时间以及小流的完成时间.PIAS 利用端主机来记录每个流已经发送的字节数并标记数据包的优先级,而在交换机里只需要配置普通的优先级队列.因此,PIAS 可以在普通的商用交换机和标准的 TCP/IP 协议栈上实现,容易在生产环境里部署.

在 PIAS 系统里,一个关键的部分就是多级反馈队列的阈值.阈值决定了如何将不同大小的流隔离在不同优先级里.如果阈值设置得不好,大小流可能某个时刻全部混杂在一个优先级里面,严重拖长小流的完成时间.简单的理论分析发现,理想的阈值跟每条链路的流大小分布和平均利用率都有关系.在真实的数据中心环境里,流的大小分布和平均利用率是在时间和空间 2 个维度上同时变化的.在这种条件下,阈值和流量的错配几乎是无法避免的.对此,PIAS 搜集数据中心全局的流量信息来统一计算阈值,并利用交换机的显式拥塞通知机制来减轻错配的影响.在 2018 年的 SIGCOMM 会议上,香港科技大学的研究人员进一步提出了 AuTO[27],其中一个机制就是通过深度强化学习来自动优化多级反馈队列的阈值.

PIAS 把所有流都简单视为不能预先知道大小且没有截止完成时间限制的流.事实上,在数据中心里面,多种类型的流是共存的.第 1 类流是有规定期限的流.对于这种流,流的大小往往也可以预先知道或者估计出来.第 2 类流是没有规定期限的限制,也无法预先知道大小的流.对于这种流,PIAS 的优化效果是最好的.第 3 类流是没有规定期限的限制,但是可以预先知道大小的流.对于这种流,使用少量优先级的 pFabric 近似方案优化效果是最好的.为了解决上述“混合流”(mix-flows)的调度问题,香港科技大学的研究人员在 2016 年的 SIGCOMM 会议上提出了 Karuna[22].严格来说,Karuna 并不是一个纯粹的无信息感知的方案,它只是不知道第 2 类流的信息.Karuna 把有规定期限的流(第 1 类)固定在最高优先级,以降低期限错失率.但是为了降低第 1 类流对于另外 2 类流的影响,Karuna 使用了一种叫作 MCP[28]的传输层协议,让第 1 类流以较低的吞吐量“恰好”在规定期限内完成.对于没有规定期限的限制,也无法预先知道大小的流(第 2 类),Karuna 采用了多级反馈队列来调度,让一个流在传输过程中优先级不断降低.对于没有规定期限的限制,但是可以预先知道大小的流(第 3 类),Karuna 根据流的大小,赋予它们一个固定的优先级,一个流越大,得到的优先级就越低.通过对 3 类流的区分处理,Karuna 可以同时降低流的期限错失率和平均完成时间.

2.4 其 他

2.1~2.3 节介绍的工作要么侧重基于端主机的拥塞控制,要么侧重基于交换机的仲裁机制,要么侧重网络的优先级调度,密歇根州立大学的研究人员在 2014 年的 SIGCOMM 会议上提出了 PASE[29],认为这 3 个机制应该结合起来.PASE 同时使用了仲裁、优先级队列和基于 ECN 的速率控制协议.与 D^3 和 PDQ 那样利用网络仲裁每个流的速率不同,PASE 把仲裁机制实现在端主机或者交换机的 CPU 上,计算出每个流在网络里的优先级和粗粒度的参考速率,再利用基于端主机的速率控制对速率做细粒度控制.因此,PASE 不需要对交换机芯片进行修改,而且不用担心流的发送速率计算不够准确的问题.PASE 依靠商用交换机上有限的优先级队列来调度流,并且利用基于 ECN 的速率控制协议来充分利用链路带宽.因此,跟 D^3, PDQ, pFabric 相比,PASE 不仅更容易部署,而且在许多场景下甚至可以取得更好的性能.

3 趋势与展望

随着数据中心网络的升级换代(如带宽从 1 Gbps 增加到 10 Gbps 再到 40 Gbps/100 Gbps,基础往返延迟从几百微秒降低到几微秒),传统的 TCP 传输层协议及其变型逐渐难以达到预期的网络传输性能.一方面,TCP 协议采用了反应式的拥塞控制算法,无法胜任流量突发性更强(大多数流在几个 RTT 内就完成了)的高速网络传输;另一方面,TCP/IP 协议栈基于系统内核软件,造成了高 CPU 消耗和高网络协议栈延迟.针对这 2 个问题,2015 年以来的科研工作分别尝试采用了接收端驱动的主动拥塞控制和 RDMA 网络传输的方式加以解决.本节将分别讨论分析这 2 类工作.表 2 和表 3 分别对这 2 类工作做了比较分析.

3.1 接收端驱动的主动拥塞控制

数据中心网络下接收端驱动的主动拥塞控制(proactive congestion control, PCC)的雏形是由加州大学伯克利分校的研究人员在 2015 年的 CoNEXT 会议上提出的 pHost[30].pHost 的设计初衷是以一种可实现的方式达到与 pFabric(性能上被

认为是当时最优的设计)相近的性能.一种可实现的做法是使用集中式控制器,带来的问题是调度时间开销很大,通常导致小流的完成时间远远差于pFabric.另辟奇径地,pHost将流调度的任务放到了数据接收端:数据发送方发送数据前请求发送(request to send, RTS),可以包含相关调度信息(如流大小等);数据接收方根据收到的所有请求按照一定的调度算法分配发送许可(token或credit或grant或pull);数据发送方按照接收到的发送许可发送相应数量的数据.这样既保证了商用设备可实现,又达到了可扩展的流调度,实验验证可以达到pFabric相近的性能.

Table 2 Comparison Between Existing Work on Proactive Congestion Control

表2 主动拥塞控制的现有工作的比较

Protocol	Strength	Weakness
pHost[30]	Readily deployable	1) Don't work under network core congestion 2) Switch-local suboptimal multipath 3) Waste bandwidth with unused credits
ExpressPass[31]	1) Work under network core congestion 2) Zero loss and low queueing delay	1) Hard to deploy (e.g., symmetric routing) 2) Idle in the first RTT (extra delay) 3) Waste bandwidth with unused credits
NDP[32]	1) Endhost-based multipath 2) Optimize for incast problem	1) Don't work under network core congestion 2) Can't be deployable with commodity switch 3) Waste bandwidth with unused credits
Homa[33]	1) No bandwidth waste 2) Support priority scheduling with preemption	Don't work under network core congestion

Table 3 Comparison Between Existing Work on RDMA Congestion Control

表3 RDMA拥塞控制的现有工作的比较

Protocol	Lossless	Retransmission	Congestion Signal	Per-packet ACK and Reaction
DCQCN[34]	Yes	Go-back-*N*	ECN	No
TIMELY[35]	Yes	Go-back-*N*	RTT	No
IRN[36]	No	SACK	Packet drop	Yes
HPCC[37]	Yes	Not mentioned	INT (In-network telemetry)	Yes

接收端驱动的主动拥塞控制因为其接近零排队时延和快速收敛等优点被学术界和工业界广泛关注.然而,现有的主动拥塞控制设计仍然存在一些问题.例如,pHost只能解决网络拥塞发生在网络边缘,即发送端网络入口链路及接收端网络出口链路处的情形.它假设数据中心网络采用全双向(full bisection)带宽和交换机逐包分撒(packet spray[38])链路负载衡以达到网络内部无拥塞.对于网络入口处的拥塞,在发送端上可以解决;对于网络出口处的拥塞,则由接收端调度完美解决.然而,实际数据中心网络中这一假设并不一定成立,例如,在发生链路故障导致网络不对称时,网络内部仍然会发生拥塞,pHost在这时候将无法发挥作用.

为了弥补pHost的不足,韩国先进科学技术研究院和谷歌的研究人员在2017年的SIGCOMM会议上提出了ExpressPass[31],是一种能够解决网络任意节点拥塞的主动拥塞控制算法.ExpressPass的数据发送方发送数据前请求发送;数据接收端发送credit在网络中通过独立的队列及限速反向模拟数据发送过程,在credit轻微排队时即丢包;数据发送方按照接收到的发送许可发送相应数量的数据.ExpressPass利用了credit的反向模拟获取网络状态,限制了队列深度,避免了真实数据包的丢失,同时credit的控制回路可以做得更加激进,从而实现快速收敛.

与此同时,欧洲的研究人员也在2017年的SIGCOMM会议上提出了另一种主动拥塞控制算法NDP[32].NDP在pHost的基础上使用基于端主机的逐包多路负载均衡机制替代网络交换机本地的逐包分撒机制,可以避免因为链路故障等原因导致的网络内部的拥塞(但仍然无法解决网络拓扑不对称或核心带宽不足等导致的网络内部拥塞问题);同时加入砍去有效载荷(cutting payload, CP)[39]的机制,可以避免在incast等极度拥塞状况下的长尾部

时延.NDP 虽然性能上比 pHost 优越,但是却因为引入 CP 而失去了商用设备可实现的优点.

斯坦福大学和麻省理工学院的研究人员在 2018 年 SIGCOMM 会议上提出了 Homa[33],对 pHost 作了进一步的改进.Homa 的创新之处有 2 点:1)它发现了当 pHost 发送端与多个接收端通信并同时收到发送许可时,部分发送许可及相应的带宽会浪费的问题.对此,Homa 引入了过度承诺(overcommitment)的机制,发送超过网络容量的发送许可来保证网络带宽不被浪费.2)它发现了 pHost 在新流到达时不能及时抢占,需要等待一个 RTT 才能调整调度策略的问题.对此,Homa 使用了网络多优先级队列的机制,从而达到第 1 个 RTT 快速抢占,这同时也解决了过度承诺超发导致的拥塞丢包问题.

现有的基于接收端的主动拥塞控制存在的另一问题是往返时延不同的流共存时发送许可触发的数据报文不协调.原来以计算好的时间间隔发送的发送许可,触发的数据报文可能因为时延不同而同时到达同一条链路,产生意外的拥塞.另外,如何将这类设计应用到 RDMA 和广域网的场景下也是一个有挑战的课题,上述因时延不同导致的问题在这些场景下会更加突出.

3.2 RDMA 网络拥塞控制

为了解决 3.1 节的高速低延迟数据中心网络发展的问题,特别是 TCP/IP 内核软件协议栈带来的性能瓶颈,远程直接内存访问(remote direct memory access, RDMA)[40]开始被引入到 DCN 中.RDMA 可以将数据直接从一台计算机的内存传输到另一台计算机,无需双方操作系统的介入,因而具有极高的性能.

虽然 RDMA 技术已经广泛应用于高性能计算(high performance computing, HPC)领域,在 DCN 中大规模部署仍然存在一些挑战.一个核心的问题是如何将网络传输机制集成到 RDMA 硬件设备(包括网卡和交换机)中,目前存在 3 类解决方案:1)IB(InfiniBand)[41].IB 使用专用网卡和交换机等硬件实现无损网络(即网络不丢包),与传统以太网络不兼容.目前广泛应用于 HPC 领域,难以部署于广泛基于 IP 和以太网技术的 DCN.2)iWARP[42].网卡硬件实现了完整的 TCP 传输协议,支持传统以太网络.受限于复杂的网卡实现,基本性能(延迟、吞吐量)被普遍认为弱于 RoCE.3)RoCE[43].网卡硬件实现了简单的传输协议,支持传统以太网络,普遍被认为在有损网络下性能不佳,通常配合基于优先级的流量控制(priority flow control, PFC)使用.然而,使用 PFC 存在线头阻塞和死锁等问题.

数据中心网络 RDMA 拥塞控制最早的代表性工作是由微软、迈络思和加州大学圣塔芭芭拉分校的研究人员在 2015 年的 SIGCOMM 会议上提出的 DCQCN[34].这也是目前工业界采用最广的 RDMA 拥塞控制算法,已经被集成到迈络思 ConnectX 系列网卡上.DCQCN 为了支持以太网采用了 RoCEv2 技术,配合使用了 PFC 流量控制技术以保证无损网络.为了避免频繁触发 PFC 带来线头阻塞和死锁等问题,DCQCN 采用了其核心拥塞控制算法,包括 3 部分:1)拥塞点(congestion point, CP),即交换机,采用基于 RED[44] 和瞬时队列长度的 ECN 标记算法.2)通告点(notification point, NP),即数据接收端,根据收到数据包含的 ECN 信息返回拥塞通告包(congestion notification packet, CNP).受当时硬件限制,CNP 生成间隔(默认为 $N=50\,\mu s$)只能达到微秒级别,因而很难做到逐包反馈.3)反应点(reaction point, RP),即数据发送端,与 DCTCP 类似,维护了 ECN 比例的指数加权滑动平均值 α.每收到 CNP 则乘性砍 $\alpha/2$;每隔 $K=55\,\mu s$ 内若没有收到 CNP 则增加发送速率.DCQCN 拥塞控制算法与 DCTCP 非常相似,但受硬件条件约束,有 2 点明显的不同:一是直接控制发送速率而非拥塞窗口;二是基于计时器调整速率而非逐包调整.这些硬件限制导致的差异给 DCQCN 的设计与参数配置带来了极大的挑战,该工作在理论上给出了指导原则与证明,但是在实际部署中仍然存在许多问题.

与此同时,谷歌、加州大学伯克利分校和微软的研究人员也在 2015 年的 SIGCOMM 会议上提出了一套拥塞控制方案 TIMELY[35],并围绕 RDMA 环境作了详细分析与评估.TIMELY 是第 1 个在数据中心内部采用 RTT 作为拥塞反馈信号的传输控制协议.TIMELY 发现当前网卡技术支持精确的 RTT 测量,并且 RTT 可以有效反映网络拥塞队列深度.基于这一观察,TIMELY 通过 RTT 与 T_{low} 和 T_{high} 两个参数比较,将核心算法划分为 3 部分:1)当 RTT 小于 T_{low} 时,增加发送速率;2)当 RTT 大于 T_{high} 时,减小发送速率;3)当 RTT 介于两者间时,根据 RTT 的梯度变化动态调整速率.当 RTT 梯度为正时,说明延迟及拥塞仍在增加,应该减小发送速率;当 RTT 梯度为负时,说明延迟及拥塞在减小,应该增加发送速率.TIMELY 主体部分采用基于 RTT 梯度的速率调整算法在性能上广受争议,普遍

被人认为在动态网络下没有唯一的收敛点,即有可能在一个很高延迟的范围内收敛[45].

DCQCN 依赖于 PFC 保证网络无损,因而存在 PFC 带来的各种问题,包括线头阻塞和死锁等.对此,许多工作[46-47]尝试去逐个解决这些问题.2018 年在 SIGCOMM 会议上由加州大学伯克利分校等的研究人员提出的 IRN[36]则反其道而行之,认为网络无损或 PFC 对于以太网上支持 RDMA 不是根本上不可或缺的.IRN 结合了 iWarp 和 RoCE 各自的优势,一方面类似于 iWarp,将网络可靠性建立在端传输协议而非无损网络上;另一方面类似于 RoCE,在网卡上实现尽可能简化的协议机制.IRN 设计上有 2 个主要部分:1)高效的丢包恢复,使用选择重传取代低效的回退 N 步(go-back-N)机制,同时采用双超时重传定时器减小尾部丢包等问题的影响.2)简单的流速控制,将发送数据量限制在一个 BDP 内,在不浪费带宽的前提下尽可能地避免拥塞.IRN 在仿真器上做了大量实验证明其在没有 PFC 的支持下能够达到甚至超过使用 PFC 的 DCQCN 和 TIMELY 的性能.然而,IRN 目前并未集成到商用网卡中.数据中心 RDMA 网络应该向无损网络还是有损网络方向发展仍然是一个热门话题.

更进一步地,哈佛大学和阿里巴巴的研究人员在 2019 年的 SIGCOMM 会议上提出了全新的 RDMA 拥塞控制方案 HPCC[37]. HPCC 发现 RDMA 拥塞控制方案存在许多参数,这些参数的调整涉及到网络吞吐量、延迟和稳定性(是否频繁触发 PFC 等)等性能指标相互之间的利益权衡.文献[37]认为这种非此即彼的利益权衡是由以前的硬件约束(如基于计时器的速率调整等)导致的,现有硬件技术的发展可以打破这种尴尬局面,具体表现在 2 个方面:一方面,现有网卡有更强大的计算能力和资源,可以支持更精确快速的逐包确认和响应;另一方面,网络交换机有了更开放可自定义的数据平面(data plane),比如,可以支持网络内部自动测量(in-network telemetry, INT),交换机在每个经过的数据包包头加入当前时间、队列长度、历史发送数据量等信息,帮助端主机作出更精确的拥塞控制响应.

HPCC 拥塞控制主要由 2 部分组成:1)交换机采用了 INT 反馈机制,在每个经过的数据包的包头加入当前时间、队列长度、历史发送数据量等信息;2)端主机拥塞控制同时限制发送速率 R 和窗口 CWND,速率调整算法包含加性和乘性调整 2 部分.加性部分每个 RTT 增加固定的发送速率 $aiRate$,这样类似于 TCP 可以保证小流不被饿死,即保证了流竞争的公平性.乘性部分首先基于 INT 反馈信息计算出网络逐跳的链路利用率 U_i,再根据逐跳最大的瓶颈利用率 $\max_i(U_i)$ 与设定的目标利用率 U_{target} 的比值乘性地调整发送速率 R.为了避免对单一拥塞事件的多次重复响应,HPCC 还结合了逐包和逐 RTT 的速率调整机制.逐 RTT 的速率调整会在每个 RTT 开始时设定一个参考速率 R^c,逐包的速率调整则根据 INT 反馈围绕当前 RTT 下的 R^c 上下波动调整速率.这样在保证快速响应的同时,可以避免触发不必要的降速和带宽浪费.

RDMA 下的传输层设计仍然有很大的探索空间.如何设计 RDMA 传输层使得网络可以扩展到广域网中?如何设计 RDMA 网络下的负载均衡机制?能否在 RDMA 网络卡中使用基于接收端的主动拥塞控制取代传统的被动拥塞控制?如何处理好传统 TCP/IP 流量和 RDMA 流量共存的场景?这些问题的探索和解决将会是未来的研究趋势.

4 总 结

近 10 年来,在盛行的网络应用(如搜索、在线零售和云计算等)的需求驱动下,数据中心在全球范围内以前所未有的速度和规模发展建立起来.特别地,数据中心网络引起了包括学术界和工业界的广泛关注.在此背景下,本文围绕数据中心网络中的传输层协议设计作了详细的总结和未来的展望.首先,本文回顾了传输层协议在因特网上的发展,探讨了数据中心网络和因特网不一样的特点,以及这些特点给传输层协议设计带来的机遇和挑战.随后,本文探讨了早期(2010—2015 年)数据中心网络传输设计方面的工作,主要包括 3 类——基于端主机的拥塞控制、网络仲裁机制和交换机优先级调度.这 3 类工作在可实现性和性能等方面各有优缺点,在各自方向上都有了相当成熟和完善的研究.最后,本文分析了 2015 年以来数据中心网络传输设计的研究方向,主要包括 2 类——接收端驱动的主动拥塞控制和 RDMA 传输协议设计.这 2 类工作在理论和系统实现上至今仍存在着许多尚未解决的问题,将会成为数据中心网络传输协议研究的未来趋势.

参 考 文 献

[1] Al-Fares M, Loukissas A, Vahdat A, et al. A scalable, commodity data center network architecture [C] //Proc of the ACM Special Interest Group on Data Communication. New York: ACM, 2008: 63-74

[2] Alizadeh M, Greenberg A G, Maltz D A, et al. Data center TCP (DCTCP)[C] //Proc of the ACM Special Interest Group on Data Communication. New York: ACM, 2010: 63-74

[3] Deng Gang, Gong Zhenghu, Wang Hong. Characteristics research on modern data center network [J]. Journal of Computer Research and Development, 2014, 51(2): 395-407 (in Chinese)
(邓罡, 龚正虎, 王宏. 现代数据中心网络特征研究[J]. 计算机研究与发展, 2014, 51(2): 395-407)

[4] Postel J. User Datagram Protocol [S/OL]. RFC 768, 1980 [2019-10-25]. https://tools.ietf.org/html/rfc768

[5] Postel J. Transmission Control Protocol [S/OL]. RFC 793, 1981[2019-10-25]. https://tools.ietf.org/html/rfc793

[6] Allman M, Paxson V, Blanton E. TCP Congestion Control [S/OL]. RFC 5681, 2009 [2019-10-25]. https://tools.ietf.org/html/rfc5681

[7] Jacobson V. Congestion avoidance and control [C] //Proc of the ACM Special Interest Group on Data Communication. New York: ACM, 1988: 314-329

[8] Jacobson V. Modified TCP Congestion Control and Avoidance Algorithms [OL]. 1990 [2019-10-25]. ftp://ftp.ee.lbl.gov/email/vanj.90apr30.txt

[9] Henderson T, Floyd S, Gurtov A, et al. The NewReno Modification to TCP's Fast Recovery Algorithm [S/OL]. RFC 6582, 2012 [2019-10-25]. https://tools.ietf.org/html/rfc6582

[10] Mathis M, Mahdavi J, Floyd S, et al. TCP Selective Acknowledgment Options [S/OL]. RFC 2018, 1996 [2019-10-25]. https://tools.ietf.org/html/rfc2018

[11] Greenberg A G, Hamilton J R, Jain N, et al. VL2 A scalable and flexible data center network [C] //Proc of the ACM Special Interest Group on Data Communication. New York: ACM, 2009: 51-62

[12] Ramakrishnan K, Floyd S, Black D. The Addition of Explicit Congestion Notification (ECN) to IP [S/OL]. RFC 3168, 2001[2019-10-25]. https://tools.ietf.org/html/rfc3168

[13] Wikipedia. Active queue management [EB/OL]. [2019-06-19]. https://en.wikipedia.org/wiki/Active_queue_management

[14] Chen Yanpei, Griffith R, Liu Junda, et al. Understanding TCP incast throughput collapse in datacenter networks [C] // Proc of Workshop on Research on Enterprise Networking. New York: ACM, 2009: 73-82

[15] Vamanan B, Hasan J, Vijaykumar T N, et al. Deadline-aware datacenter TCP (D2TCP) [C] //Proc of the ACM Special Interest Group on Data Communication. New York: ACM, 2012: 115-126

[16] Munir A, Qazi I A, Uzmi Z A, et al. Minimizing flow completion times in data centers [C] // Proc of the Int Conf on Computer Communications. Piscataway, NJ: IEEE, 2013: 2157-2165

[17] Wilson C, Ballani H, Karagiannis T, et al. Better never than late meeting deadlines in datacenter networks [C] //Proc of the ACM Special Interest Group on Data Communication. New York: ACM, 2011: 50-61

[18] Hong Chiyao, Caesar M, Godfrey P B, et al. Finishing flows quickly with preemptive scheduling [C] //Proc of the ACM Special Interest Group on Data Communication. New York: ACM, 2012: 127-138

[19] Perry J, Ousterhout A, Balakrishnan H, et al. FastPass a centralized "zero-queue" datacenter network [C] //Proc of the ACM Special Interest Group on Data Communication. New York: ACM, 2015: 307-318

[20] Alizadeh M, Yang Shuang, Sharif M, et al. pFabric minimal near-optimal datacenter transport [C] //Proc of the ACM Special Interest Group on Data Communication. New York: ACM, 2013: 435-446

[21] Bai Wei, Chen Li, Chen Kai, et al. Information-agnostic flow scheduling for commodity data centers [C] //Proc of the USENIX Symp on Networked Systems Design and Implementation. Berkeley, CA: USENIX Association, 2015: 455-468

[22] Chen Li, Chen Kai, Bai Wei, et al. Scheduling mix-flows in commodity datacenters with Karuna [C] //Proc of the ACM Special Interest Group on Data Communication. New York: ACM, 2016: 174-187

[23] Rai I A, Urvoykeller G, Vernon M K, et al. Performance analysis of LAS-based scheduling disciplines in a packet switched network [J]. Measurement and Modeling of Computer Systems, 2004, 32(1): 106-117

[24] Wu Haitao, Ju Jiabo, Lu Guohan, et al. Tuning ECN for data center networks [C] // Proc of the Conf on Emerging Network Experiment and Technology. New York: ACM, 2012: 25-36

[25] Wikipedia. Shortest job next [EB/OL]. [2019-06-19]. https://en.wikipedia.org/wiki/Shortest_job_next

[26] Wikipedia. Earliest deadline first scheduling [EB/OL].[2019-06-19]. https://en.wikipedia.org/wiki/Earliest_deadline_first_scheduling

[27] Chen Li, Lingys J, Chen Kai, et al. AuTO scaling deep reinforcement learning for datacenter-scale automatic traffic optimization [C] //Proc of the ACM Special Interest Group on Data Communication. New York: ACM, 2018: 191-205

[28] Chen Li, Hu Shuihai, Chen Kai, et al. Towards minimal-delay deadline-driven data center TCP [C] //Proc of the ACM Workshop on Hot Topics in Networks. New York: ACM, 2013: 21-27

[29] Munir A, Baig G, Irteza S M, et al. Friends, not foes synthesizing existing transport strategies for data center networks [C] //Proc of the ACM Special Interest Group on Data Communication. New York: ACM, 2015: 491-502

[30] Gao P X, Narayan A, Kumar G, et al. pHost distributed near-optimal datacenter transport over commodity network fabric [C] //Proc of the Conf on Emerging Network Experiment and Technology. New York: ACM, 2015: 1-12

[31] Cho I, Jang K, Han D, et al. Credit-scheduled delay-bounded congestion control for datacenters [C] //Proc of the ACM Special Interest Group on Data Communication. New York: ACM, 2017: 239-252

[32] Handley M, Raiciu C, Agache A, et al. Re-architecting datacenter networks and stacks for low latency and high performance [C] //Proc of the ACM Special Interest Group on Data Communication. New York: ACM, 2017: 29-42

[33] Montazeri B, Li Yilong, Alizadeh M, et al. Homa a receiver-driven low-latency transport protocol using network priorities [C] //Proc of the ACM Special Interest Group on Data Communication. New York: ACM, 2018: 221-235

[34] Zhu Yibo, Eran H, Firestone D, et al. Congestion control for large-scale RDMA deployments [C] //Proc of the ACM Special Interest Group on Data Communication. New York: ACM, 2015: 523-536

[35] Mittal R, Dukkipati N, Blem E R, et al. TIMELY: RTT-based congestion control for the datacenter [C] //Proc of the ACM Special Interest Group on Data Communication. New York: ACM, 2015: 537-550

[36] Mittal R, Shpiner A, Panda A, et al. Revisiting network support for RDMA [C] //Proc of the ACM Special Interest Group on Data Communication. New York: ACM, 2018: 313-326

[37] Li Yuliang, Alizadeh M, Yu Minlan, et al. HPCC high precision congestion control [C] //Proc of the ACM Special Interest Group on Data Communication. New York: ACM, 2019: 44-58

[38] Dixit A, Prakash P, Hu Y C, et al. On the impact of packet spraying in data center networks [C] //Proc of the Int Conf on Computer Communications. Piscataway, NJ: IEEE, 2013: 2130-2138

[39] Cheng Peng, Ren Fengyuan, Shu Ran, et al. Catch the whole lot in an action rapid precise packet loss notification in data centers [C] //Proc of the USENIX Symp on Networked Systems Design and Implementation. Berkeley, CA: USENIX Associaton, 2014: 17-28

[40] Wikipedia. Remote direct memory access [EB/OL]. [2019-06-19]. https://en.wikipedia.org/wiki/Remote_direct_memory_access

[41] Wikipedia. Inifiniband [EB/OL]. [2019-06-19]. https://en.wikipedia.org/wiki/InfiniBand

[42] Wikipedia. iWARP [EB/OL]. [2019-06-19]. https://en.wikipedia.org/wiki/IWARP

[43] Wikipedia. RDMA over converged Ethernet [EB/OL]. [2019-06-19]. https://en.wikipedia.org/wiki/RDMA_over_Converged_Ethernet

[44] Floyd S, Jacobson V. Random early detection gateways for congestion avoidance [J]. IEEE ACM Transactions on Networking, 1993, 1(4): 397-413

[45] Zhu Yibo, Ghobadi M, Misra V, et al. ECN or delay lessons learnt from analysis of DCQCN and TIMELY [C] //Proc of the Conf on Emerging Network Experiment and Technology. New York: ACM, 2016: 313-327

[46] Guo Chuanxiong, Wu Haitao, Deng Zhong, et al. RDMA over commodity Ethernet at scale [C] //Proc of the ACM Special Interest Group on Data Communication. New York: ACM, 2016: 202-215

[47] Hu Shuihai, Zhu Yibo, Cheng Peng, et al. Tagger practical PFC deadlock prevention in data center networks [C] //Proc of the Conf on Emerging Network Experiment and Technology. New York: ACM, 2017: 451-463

Zeng Gaoxiong, born in 1992. Received his BEn degree in electronic engineering and information science from University of Science and Technology of China in 2015. PhD candidate at Hong Kong University of Science and Technology. His main research mainly focuses on data center networks, with particular interest in transport protocols.

Hu Shuihai, born in 1990. Received his PhD degree in computer science from Hong Kong University of Science and Technology in 2019, and received his BSc degree in computer science from University of Science and Technology of China in 2013. He currently works at Clustar as VP of technology and research. His main research interests include networked systems design and implementation, data center networks, cloud computing and distributed AI system.

Zhang Junxue, born in 1990. PhD candidate in the Department of Computer Science Engineering, Hong Kong University of Science and Technology. Received his MSc and BSc degrees in computer science from Southeast University. His main research interests include data center networks, cloud computing and distributed AI system.

Chen Kai, born in 1980. Associate professor at Hong Kong University of Science and Technology (HKUST). Director of Networking System Lab (SING) and HKUST-WeChat Joint Lab for Artificial Intelligence Technology (WHAT). Received his BSc and MSc degrees from University of Science and Technology of China in 2004 and 2007, respectively, and received his PhD degree from Northwestern University in 2012. His main research interests include data center networking, machine learning systems, and AI infrastructure.