



ACC: 一种敏捷的数据中心网络拥塞控制技术

袁郭苑¹, 路远¹, 周仁杰², 董德尊^{1*}, 彭伟¹

1. 国防科技大学计算机学院, 长沙 410028

2. 西北核技术研究所, 西安 710024

* 通信作者. E-mail: dong@nudt.edu.cn

收稿日期: 2024-06-13; 修回日期: 2024-09-20; 接受日期: 2024-11-08; 网络出版日期: 2024-12-31

国家重点研发计划 (批准号: 2022YFB4501702) 资助项目

摘要 面对日益复杂且随时间迅速变化的数据中心网络流量, 基于带内网络遥测 (in-network telemetry, INT) 信息的拥塞控制技术面临诸多挑战. 一方面, 现有的较为成熟的算法主要依赖接收方的 ACK (acknowledge) 包反馈 INT 信息到发送方来获得链路负载信息, 由于 ACK 包的反馈时延是 RTT (round trip time) 级别的, 发送方在流传输的第 1 个 RTT 内无法获得 INT 信息. 另一方面, 现有的算法普遍缺乏在流传输完成时对网络空闲带宽的抢占, 因而网络不可避免地会出现带宽浪费. 本文深入探索基于 INT 信息的拥塞控制技术, 有效打破了 RTT 级别的流控时间屏障, 缓和了带宽浪费, 提出了一种超精准、高效的拥塞控制算法 — ACC (agile congestion control). 与现有较为成熟的基于 INT 信息的拥塞控制算法 HPCC (high precision congestion control) 相比, ACC 格外关注流传输的第 1 个 RTT, 在此期间利用交换机将路径拥塞信号传回发送方, 能够对网络拥塞做出更敏捷的反应. 此外, ACC 关注网络中长流对带宽的抢占能力, 能够帮助长流提前感知空闲带宽, 从而降低网络传输的尾延迟. 大量的实验验证了 ACC 方案的性能, 实验结果显示在保持与 HPCC 一致的高吞吐量的基础上, ACC 能够有效降低队列长度和流尾延迟. 具体来说, 在大规模 Clos 网络拓扑中, 当工作负载为 WebServer, 负载强度为 0.6 时, 与 HPCC 相比, ACC 能够将平均队列长度减少 9.6%, 将第 95 百分位流完成时间减少 29.7%.

关键词 数据中心网络拥塞控制, 带内网络遥测 (INT), 带宽抢占, 交换机主机协同

1 研究介绍

受云数据中心中不断增长的流量需求的驱动, 链路带宽已经达到 100~400 Gbps^[1], 并且该趋势仍在持续以满足存储^[2]、高性能计算、分布式深度学习、数据挖掘和大数据处理等新兴应用的需求^[3~6]. 拥塞控制 (congestion control, CC) 可以避免网络中堆积过长的队列, 能够有效减少流完成时间 (flow completion time, FCT), 因而对于提升大规模数据中心网络的传输性能至关重要. 网络带宽的持续增

引用格式: 袁郭苑, 路远, 周仁杰, 等. ACC: 一种敏捷的数据中心网络拥塞控制技术. 中国科学: 信息科学, 2025, 55: 46–63, doi: 10.1360/SSI-2024-0184

Yuan G Y, Lu Y, Zhou R J, et al. ACC: agile congestion control in datacenter networks. Sci Sin Inform, 2025, 55: 46–63, doi: 10.1360/SSI-2024-0184

长使得更多的流量可以在一个或几个 RTT (round trip time) 内传输完成,使得现在的数据中心网络呈现出复杂且随时间变化的流量特征,包括长流和突发短流^[7,8],它们各自对带宽和延迟敏感,并存的长流和短流要求数据中心网络同时提供高吞吐量、低延迟、稳定性良好的传输^[3,5,9].

为了在数据中心网络中实现上述传输目标,研究人员提出了一系列的拥塞控制机制.第1类是以显式拥塞通知 (explicit congestion notification, ECN)^[10] 位作为拥塞信号的基于 ECN 的拥塞控制算法,这类方法以 DCQCN (datacenter quantized congestion notification)^[4] 为代表,根据交换机的缓冲区队列长度选择性地标记数据包的 ECN 位来传输网络拥塞信号.第2类是基于 RTT 的拥塞控制算法,这类方法通过在发送方测量 RTT 变化或 RTT 梯度变化来评估网络拥塞,代表性的算法有 Swift^[11], Timely^[12] 等.然而 ECN 位和 RTT 都是不准确的拥塞信号,近年来,新一代的数据中心网络交换机拥有了网络遥测 (in-network telemetry, INT)^[13~15],交换机在数据包中记录精确的路径负载信息 (即 INT 信息),随后 INT 信息通过相应的 ACK (acknowledge) 包被传回发送方^[14~17],这样就可以指导发送方进行精准的速率调节.如 HPCC (high precision congestion control)^[3] 的性能所展示的一样,基于 INT 信息的拥塞控制算法为实现数据中心网络精准的速率调节带来全新的可能.

然而,现有较为成熟的基于 INT 信息的拥塞控制算法 (如 HPCC^[3]) 使用 ACK 包向发送方反馈 INT 信息,这个反馈过程需要至少一个 RTT 的时间,这带来了以下两个缺陷.

观察 1. 在流传输的第1个 RTT 内,没有任何 ACK 包返回发送方,因而发送方也无法感知网络状态的变化.我们考虑大量短流传输的场景,这些流量在一个或几个 RTT 内就可以传输完成,这时现有的基于 INT 信息的拥塞控制算法会因为控制回路太长而无法控制这些流量,在更糟糕的短流突发场景下,拥塞控制算法会失效,网络将遭遇严重的性能损失.

观察 2. 在流传输完成时,网络会出现不可避免的带宽浪费.对于即将传输完成的流,它占用的带宽也即将得到释放,然而只有在其带宽释放完成之后,链路上经过的其他数据包才会感知到有关空闲带宽的 INT 信息,并且 INT 信息随后还需要搭载在接收方产生的 ACK 包中才能够抵达发送方,可见网络带宽从流传输完成被释放到被其他流重新利用,这一过程会导致不可避免的带宽浪费^[18].

对于算法在流传输的第1个 RTT 内管理失效的问题,我们诉诸路径上的交换机寻求解决方案.交换机作为网络拥塞最重要的观察者,可以提供新鲜及时的 INT 信息,如果交换机在第一时间反馈 INT 信息,拥塞控制机制就可以使发送方在流传输的第1个 RTT 内获得链路负载信息.然而通过大量的研究和实验,我们发现利用交换机 INT 面临两个重要挑战.

挑战 1. 拥塞控制算法需要处理交换机 INT (由交换机直接发回发送方的 INT 信息) 和主机 INT (由主机直接发回发送方的 INT 信息) 之间的协同.假设一个场景,路径上的每个节点都在反馈 INT 信息,对于发送方来说,大量的 INT 信息源源不断地到来,其中一些比较新鲜及时,另一些比较陈旧甚至已经过期,此外这些 INT 信息的内容也不相同,一些包含完整路径的链路负载信息,另一些只包含某些局部路径的链路负载信息,发送方需要选择最新鲜最有效的信息来调节发送速率,这给发送方算法设计带来重大挑战.我们需要一个简洁高效的算法来完成 INT 信息的选择和使用.

挑战 2. 我们的设计需要将控制信息 (来自交换机和接收方的) 开销限制在可接受的范围内.由于交换机和接收方都会产生确认包来传递路径 INT 信息,当数据包数量增加时,如果不加控制,网络将会出现过量的确认包,这会浪费网络带宽并影响传输性能.

至于算法在流即将传输完成并释放空闲带宽时出现的带宽浪费问题,最关键的是尽可能早地通知网络中的其他流量,这里我们面临另一个挑战.

挑战 3. 我们的设计需要消除拥塞控制算法控制下带宽释放和带宽抢占之间的真空期.由于 HPCC 等方法是基于接收方产生的 ACK 包传递 INT 信息的,在流释放带宽后网络中其他活跃的发送方并不能立即感知到空闲带宽,因此算法设计需要尽可能早地向网络通知流完成事件,对流完成事件进行预反馈.

为了更好地区分 INT 信息的质量,我们提出 INT 信息的“新鲜度”属性.测量发送方收到 INT

表 1 4 种数据中心网络常见工作负载的组成.

Table 1 Composition of four common workloads in datacenters.

	DataMining	CacheFollower	WebServer	WebSearch
0–10 kB	78%	50%	63%	49%
10–100 kB	5%	3%	18%	3%
100 kB–1 MB	8%	18%	19%	18%
>1 MB	9%	29%	–	20%
Average flow size	7.41 MB	701 kB	64 kB	1.6 MB

信息的时间和 INT 信息的产生时间, 两时间差越小, INT 信息的新鲜度越高, 反之新鲜度越低. 以向发送方提供高新鲜度的 INT 信息为出发点, 本文提出了一种名为 ACC (agile congestion control) 的超精准的拥塞控制算法, 该算法兼具准确性、及时性和低控制信息开销的特性.

总的来说, 本论文的主要贡献如下:

- 提出了一种名为 ACC 的拥塞控制算法. ACC 能够在每条流传输的第 1 个 RTT 内检测和调控网络拥塞, 同时能够对网络空闲带宽进行预反馈和提前抢占, 该设计实现了数据中心网络中超精准的流量控制.
- 提出了两点关键设计: 第一, 本文提出了一套完备的交换机和主机协同的 INT 信息反馈和计算机制, 我们称之为交换机早期反馈机制 (3.3 小节). 该机制包含多视图技术 (3.3.1 小节) 和有限交换机 INT 技术 (3.3.2 小节), 能够在流传输的第 1 个 RTT 内控制数据中心网络流量, 对于短流和突发流量有明显的效果. 第二, 本文提出了 Cloak 机制 (3.4 小节), 能够对流完成事件进行预反馈, 帮助网络流量尽早抢占空闲带宽, 有助于提升长流的传输性能.
- 使用常见的数据中心网络工作负载进行了大量的实验, 包括设置消融实验验证交换机早期反馈机制和 Cloak 机制的有效性、对 ACC 的微基准性能评估以及大规模 Clos 拓扑上的传输性能验证.

2 研究动机

2.1 观察 1: 短流和突发流量引起的拥塞

随着数据中心网络带宽增加至 400 Gbps^[1], 更多的流能够在较短的时间内 (一个或几个 RTT 内) 传输完成, 短流占比加重逐渐成为一种趋势. 瞬时突发流量和大量短流会破坏网络传输的稳定性, 会给拥塞检测和拥塞控制带来较大的压力^[19], 如果这种瞬时拥塞没有得到妥善的处理, 就有可能引发网络 Incast, 造成比较严重的拥塞. 如果拥塞进一步加剧, 对于有损网络来说可能会造成丢包, 对于无损网络来说可能会触发 PFC (pause frame cap) 暂停帧, 无论是丢包引起的数据包重传还是 PFC 暂停帧造成的拥塞传播, 都会大大损害数据中心网络的传输性能.

表 1 展示了 4 种数据中心网络常见工作负载的流量分布情况, 可见数据中心网络中有大量的短流和一定量的超长流, 长流与突发的短流混合发送是常见的流量模式. 在流传输的第 1 个 RTT 内对流量进行控制能够有效应对瞬时突发流量和短流, 有益于提升数据中心网络整体的传输性能.

我们在图 1 所示的半哑铃拓扑中对 HPCC 的性能进行评估, 该拓扑包含 4 台交换机和 12 台主机. Switch3 和 Switch4 之间的链路带宽是 100 Gbps, 其他链路的带宽是 40 Gbps, 任意两相邻节点之间的延迟是 1 μ s. 为了观察 HPCC 对突发短流的管理, 在该拓扑中我们令每个发送方随机选择一个接收方发送数据, 使用短流占比较高的 WebServer 工作负载进行 8 组实验 (调整负载强度为 0.2~0.9), 测量网络瓶颈的队列堆积情况, 即 Switch1 和 Switch2 平均的出口端口队列长度.

实验结果如图 2 所示, 可以看出随着短流占比的上升, HPCC 控制队列的能力大大减弱. 当负载强度小于 0.4 时, Switch1 和 Switch2 平均的队列长度能够大致维持在 90 kB 以内的水平. 但当负载

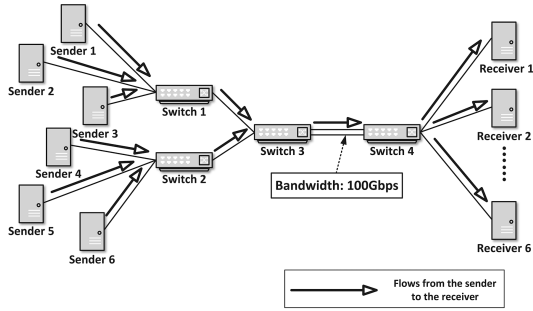


图 1 半哑铃拓扑: Switch3 和 Switch4 之间的链路带宽为 100 Gbps, 其他链路的带宽是 40 Gbps, 任意两相邻节点之间的延迟是 1 μ s.

Figure 1 Half-dumbbell topology: the link bandwidth between Switch3 and Switch4 is 100 Gbps, and the bandwidth of other links is 40 Gbps. The delay between two adjacent nodes is 1 μ s.

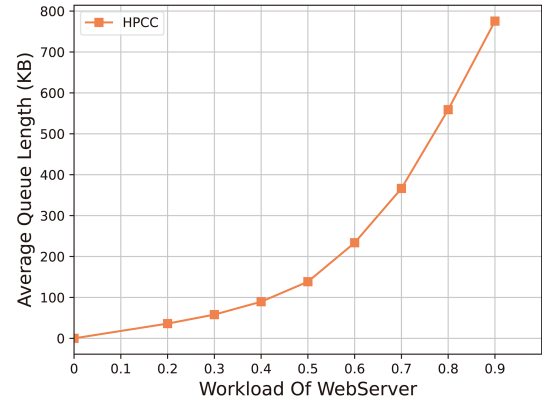


图 2 (网络版彩图) 不同负载强度下 Switch1 和 Switch2 平均的出口端口队列长度 (半哑铃拓扑, WebServer 工作负载).

Figure 2 (Color online) Average egress port queue length of Switch1 and Switch2 under different intensity loads (Half-dumbbell topology, WebServer workload).

强度超过 0.4 后, 队列出现明显堆积, 当负载强度为 0.5 时, 队列长度为 138 kB, 当负载强度为 0.7 时, 队列长度为 366 kB, 当负载强度到达 0.9 时, 队列长度高达 775 kB, 此时网络队列中数据包大量堆积, 出现拥塞, 网络传输性能大大受损. 值得注意的是与负载强度为 0.8 时相比, 当负载强度为 0.9 时, Switch1 和 Switch2 平均的出口端口队列长度增长了 38.7%. 交换机处的长队列表示网络陷入严重的拥塞, 由此可见仅依赖主机 INT 很难控制大量的短流.

2.2 观察 2: 长流引起的带宽浪费

如表 1 所示, 数据中心网络中存在一定数量的长流, 这些长流对吞吐量敏感, 会在较长一段时间内占用带宽资源. 当一条长流传输完成时, 其占用的带宽资源通常无法立刻被其他流量重新使用, 这与 INT 信息的反馈延迟有关. 具体来说, 随着某条长流最后一个数据包的传输, 长流占用的链路带宽逐渐被释放, 在 HPCC 的控制下, 这部分带宽资源会一直空闲直到某个正向传输的数据包收集了相关的链路负载信息, 然后接收方在收到该数据包后将相关信息搭载在对应的 ACK 包中传回发送方, 发送方才能够感知到空闲带宽并进行增速. 因此从某条长流完成传输并且释放带宽到其他流量感知到空闲带宽并且增速, 这两个阶段之间是存在一定时延的, 在这期间网络中有未被利用的空闲带宽, 存在带宽浪费的情况.

图 3 展示了长流引发带宽浪费的一个场景, 该图所示的 Four-to-one 拓扑包含 4 个发送方和 1 个接收方, 4 个发送方到接收方共用一条路径, 路径上有 5 台交换机, 任意两相邻节点之间的链路速率为 40 Gbps, 链路延迟为 1 μ s. 在此拓扑中, 4 个发送方同时持续向接收方发送长流并达到稳定状态, 此时 4 条流平分网络带宽, 即 4 个发送方各自的吞吐量均为 10 Gbps. 在到达某时刻后, 我们令网络中的 3 个发送方 (Sender1~Sender3) 同时停止发送, 并观察在 HPCC 控制下 Sender4 抢占空闲带宽的能力.

实验结果如图 4 所示, 图中紫色虚线表示 Sender1~Sender3 的吞吐量变化, 我们控制这 3 台主机都在约 548 μ s 时停止发送, 图中红色实线表示 Sender4 的吞吐量变化情况. 实验结果显示在 HPCC 的控制下, Sender4 约在 637 μ s 时完成空闲带宽的抢占, 其吞吐量达到 40 Gbps. 该网络的 RTT 是 12 μ s, 抢占时间约 7.4 个 RTT.

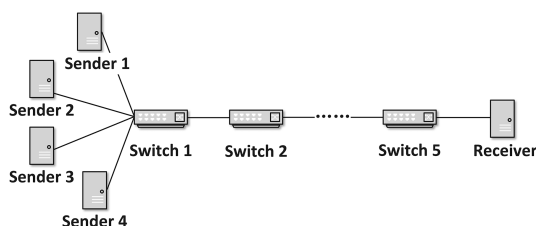


图3 Four-to-one 拓扑: 链路带宽是 40 Gbps, 任意两相邻节点之间的延迟是 1 μ s.

Figure 3 Four-to-one topology: the link bandwidth is 40 Gbps and the delay between two adjacent nodes is 1 μ s.

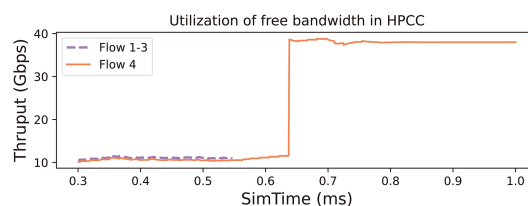


图4 (网络版彩图) HPCC 中空闲带宽的利用情况: 抢占时间约 7.4 个 RTT (Four-to-one 拓扑).

Figure 4 (Color online) Utilization of free bandwidth in HPCC: the preemption time is 7.4 RTTs (Four-to-one topology).

2.3 基于主机 INT 的算法所具有的内在缺陷分析及可行的解决方案

从本质上来说, 上述两个观察提到的问题都是由基于主机 INT 的算法的内在缺陷引起的, 即算法的拥塞控制回路过长, 其时延是 RTT 级别的.

起因 – 短流引发的拥塞. 仅使用主机 INT 的算法在设计上存在缺陷, 无法管理流传输的第 1 个 RTT. HPCC 使用主机 INT 传递链路负载信息, 对于一条流来说, 从发送方发送第 1 个数据包到收到第 1 个 ACK 包至少需要一个 RTT, 在这期间发送方没有任何链路负载信息, 无法感知拥塞.

起因 – 长流引发的带宽浪费. 仅使用主机 INT 的算法的拥塞控制回路时延是 RTT 级别的, 不可避免地会出现带宽浪费问题, 尤其是对于吞吐量敏感的长流. HPCC 使用 ACK 包反馈 INT 信息调节发送行为, 在一条流发送完成后, 其释放的空闲带宽需要等其他经过的数据包感知, 随后 INT 信息搭载在相应的 ACK 包中传回发送方, 发送方才会增速, 这一过程会出现带宽浪费.

总的来说, 尽可能缩短拥塞控制回路以增强拥塞信号的新鲜度是提升拥塞控制算法性能的关键. 交换机能够捕捉到最即时的链路状态信息变化, 并具备向网络其他设备主动报告的能力, 因此我们诉诸交换机与主机的协同方案来解决前述的拥塞和带宽浪费问题.

解决方案 – 短流引发的拥塞. 借助交换机, 我们可以快速地检测和控制网络拥塞. 一个自然的想法是使用交换机直接向发送方反馈拥塞, 以实现流生命周期中细粒度的拥塞控制. 这需要考虑两个问题, 第一, 要兼顾交换机 INT 的新鲜性和主机 INT 的完整性, 第二, 要将控制信息 (交换机 INT 和主机 INT) 的开销控制到合理的水平.

解决方案 – 长流引发的带宽浪费. 解决长流引发的带宽浪费问题的关键是将链路空闲的 INT 信息尽可能快地传回发送方. 一个可行的方案是利用交换机的 INT 标记功能隐藏长流尾部 (最后一个或几个 RTT 内) 数据包的发送行为, 实现对流完成事件的“预反馈”, 帮助网络中其他流量尽快抢占空闲带宽.

3 交换机与主机协同的设计方案: ACC

3.1 拥塞信号说明

3.1.1 交换机 INT 和主机 INT

依据 INT 信息所搭载的确认包的不同, 我们将 INT 信息分为交换机 INT 和主机 INT, 其中交换机 INT 是搭载在路径上交换机产生的 SAK (switch ACK) 包中传回发送方的, 主机 INT 是搭载在接收方产生的 ACK 包中传回发送方的^[20~23].

对于路径上的每台交换机, INT 信息记录了其出口端口的链路负载数据, 具体包含交换机标识 switch_id, 出口端口的链路带宽 bwth, 时间戳 TS (time stamp), 出口端口累积发送的字节数 txBytes

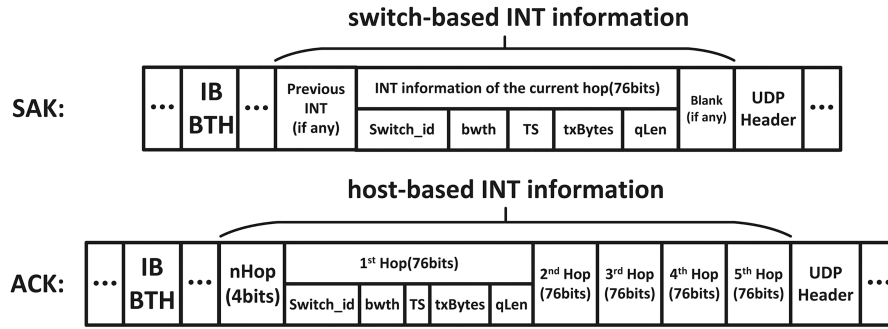


图5 SAK包和ACK包的包格式。

Figure 5 Formats of SAK packets and ACK packets.

和出口端口的队列长度 $qLen$ 。图5展示了SAK包和ACK包的包格式, INT填充区位于UDP头和IB BTH (base transport header) 之间, SAK包包含从发送方到当前交换机的局部路径上的INT信息, ACK包包含整条路径完整的INT信息。统计数据显示在数据中心网络中, 中间交换机节点的数量通常不会超过5, 因此我们在SAK包和ACK包中均预留5跳INT信息的长度以存储路径负载数据。

3.1.2 衡量拥塞信号及时性的属性 — 新鲜度

拥塞信号直接表征网络状态并指导拥塞控制算法调节发送行为, 因此拥塞信号的质量直接决定了拥塞控制的有效性, 这里我们主要讨论两个关键指标: 精准性和及时性。

在INT信息作为拥塞信号的拥塞控制算法中, 网络向端主机传递精确的链路负载信息 (如图5所示), 这些信息可以辅助端上的拥塞控制算法进行精准的速率调节, 因此作为拥塞信号的INT信息天然地具有良好的精准性。但是从及时性来看, 目前INT信息的表现无法让人完全满意。如2.1和2.2小节所述, 由于拥塞控制回路过长, 当面对短流和突发流量时网络会遭遇拥塞, 而当面对长流时网络会出现带宽浪费。因此在设计拥塞控制算法时, 提升拥塞信号的及时性显得格外重要。

为了更准确地刻画拥塞信号的及时性, 本文为拥塞信号设置“新鲜度”的属性。我们测量拥塞信号的产生时刻 t_G 和被发送方接收时刻 t_R , 计算两个时刻的时间差 t_{gap} , 即 $t_{gap} = t_R - t_G$ 。拥塞信号的新鲜度与时间差 t_{gap} 成负相关关系, 时间差越大, 表示发送方收到的拥塞信号越陈旧, 新鲜度越低, 反之, 时间差越小, 拥塞信号的新鲜度越高, 越能够反映网络的即时特征, 因此我们用时间差 t_{gap} 来表征新鲜度。对于某单个数据包传输过程中的测量, 我们将一条路径上的多个INT信息 (即多段链路的INT信息) 视为一组, 并使用一组中最大的 t_{gap} 作为整组INT信息的新鲜度, 在评估拥塞信号及时性的使用无拥塞发生时的理论新鲜度。传递高新鲜度的拥塞信号是当前拥塞控制算法设计的重要目标。

表2展示了4种常用的拥塞控制算法的拥塞信号的新鲜度 (即及时性) 和精准性, 这里用 n 表示数据包传输路径的链路数, 用 p 表示DCQCN中传输路径上进行ECN标记的交换机所在位置之前的链路数。TIMELY和Swift采用RTT作为拥塞信号, 精准性中等, t_{gap} 为RTT, DCQCN采用交换机对数据包的ECN标记作为拥塞信号, 精准性较低, t_{gap} 为 $\frac{2n-p}{2n}$ RTT, HPCC采用INT信息作为拥塞信号, 精准性极高, t_{gap} 为 $\frac{2n-1}{2n}$ RTT。

3.2 ACC概览

ACC是使用了交换机INT和主机INT的发送方驱动的拥塞控制算法, 图6展示了算法的框架, 其包含两个主要设计模块, 分别是交换机早期反馈机制和Cloak机制。其中交换机早期反馈机制利用路径上的交换机向发送方传递INT信息, 能够在流传输的第1个RTT内检测网络状态并调控发送行为, 可以应对突发短流导致的拥塞。Cloak机制利用路径上的交换机隐藏长流尾部的INT信息, 实现对流传输完成事件的预反馈, 可以应对长流传输带来的带宽浪费。在后面的章节, 我们会详细介绍这

表 2 常用拥塞控制算法的拥塞信号的新鲜度分析 (n 表示传输路径的链路数, p 表示传输路径上进行 ECN 标记的交换机所在位置之前的链路数).

Table 2 Freshness of congestion signals of common congestion control algorithms (n represents the number of links on the transmission path, and p represents the number of links before the switch that performs ECN marking on the transmission path).

CC algorithm	t_G	t_R	t_{gap} (timeliness)	Accuracy
DCQCN	t	$t + \frac{2n-p}{2n} \text{RTT}$	$\frac{2n-p}{2n} \text{RTT}$	★★
HPCC	t	$t + \frac{2n-1}{2n} \text{RTT}$	$\frac{2n-1}{2n} \text{RTT}$	★★★★★
TIMELY	t	$t + \text{RTT}$	RTT	★★★
Swift	t	$t + \text{RTT}$	RTT	★★★

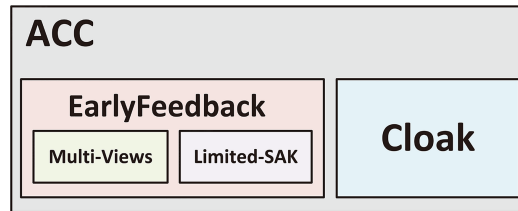


图 6 (网络版彩图) ACC 框架图: ACC 包含交换机早期反馈机制和 Cloak 机制.

Figure 6 (Color online) ACC framework: ACC includes switch early feedback mechanism and Cloak mechanism.

两个机制.

从流传输的角度看, 在 ACC 的控制下一条流的传输可以被划分为 3 个阶段: 第 1 个 RTT、中间 RTT 和尾部 RTT. 图 7 展示了 ACC 在流传输的第 1 个 RTT 内的数据传输情况, 交换机会向发送方反馈高新鲜度的 INT 信息, 帮助发送方实现第 1 个 RTT 内的流量调控. 图 8 展示了 ACC 在流传输的中间 RTT 和尾部 RTT 的数据传输情况. 在流中间 RTT 内, ACK 包开始连续不断地到达发送方, 发送方基于 ACK 包中搭载的 INT 信息调节发送速率, 该阶段交换机不再产生 SAK 包. 在流尾部 RTT 内, 算法采用 Cloak 机制在交换机处隐藏流的发送行为, 实现流完成事件的预反馈, 以便当前流即将释放的带宽资源能够迅速被网络中的其他流量重新利用.

图 9 展示了 ACC 和 HPCC 各自获取的 INT 信息量随时间的变化. 我们假设传输路径包含 n 段链路, 此处 x 轴表示时间变化, 单位用数据包在网络中传输的跳数 hop 表示, y 轴表示 INT 信息量, 即路径信息的完整程度, 单位为链路数 hop. y 轴的最大值为 n , 表示完整的路径信息, 其包括 n hop 的 INT 信息. 可以看出, 与 HPCC 相比, ACC 能够在流传输的第 1 个 RTT 内获得 INT 信息, 并且数据量随时间线性增长.

3.3 交换机早期反馈机制

3.3.1 多视图技术: 在流传输的第 1 个 RTT 内控制流

为了在流传输的第 1 个 RTT 内控制流量, 发送方需要高新鲜度的 INT 信息, 交换机是网络中首先感知到链路状态变化的设备, 因此由交换机反馈的 INT 信息有较高的新鲜度. 图 7 展示了在流传输的第 1 个 RTT 内 ACC 的控制行为, 此时网络中既有 SAK 包又有 ACK 包.

在 ACC 的控制下, 对于每条流, 发送方都维护一个视图集合 VSet, VSet 由多个视图组成, 每个视图 View _{M} 对应于从发送方到节点 M (交换机或接收方) 的局部路径信息. 具体来说 View _{M} 包含局部路径的 RTT RTT _{M} 、标准化的飞行中字节数 U_M 、发送窗口 W_M 、参考窗口 W_{CM} 、发送速率 rate _{M} 、上一组 INT 信息 Last _{M} , 以及发送窗口当前轮次所经历的加性增加次数 incStage _{M} 等.

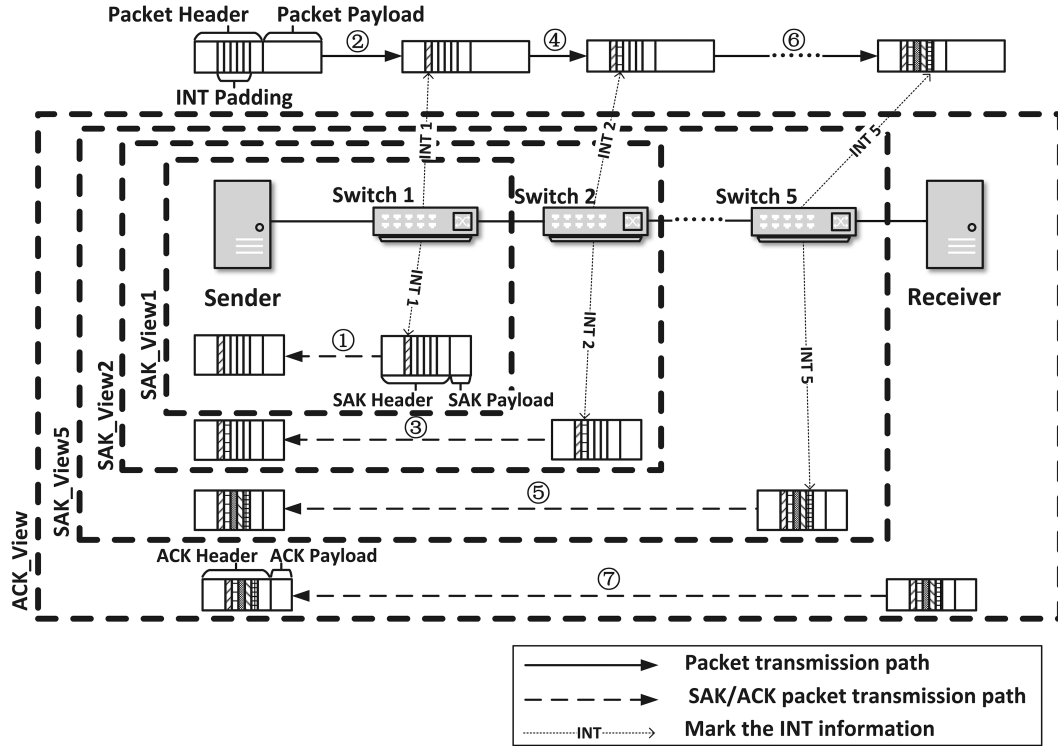


图 7 ACC 数据传输概览 (在流传输的第 1 个 RTT 内): 在流传输的第 1 个 RTT 内, 与 HPCC 相比, 多视图技术允许 ACC 获取额外的 INT 信息。

Figure 7 ACC overview (in the first RTT of a flow): in the first RTT of the flow, compared with HPCC, the multi-views allow ACC to obtain additional INT information.

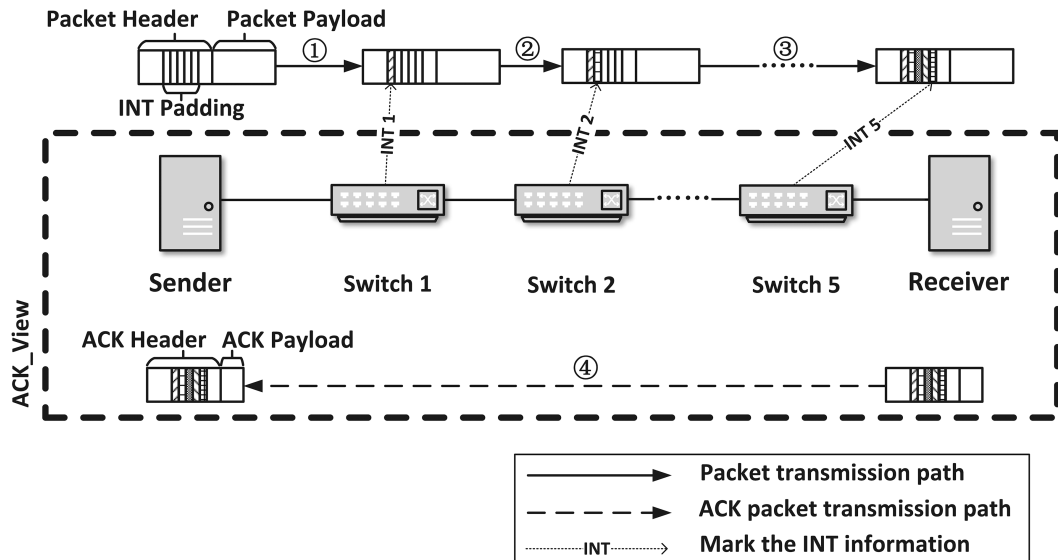


图 8 ACC 数据传输概览 (在流传输的其他 RTT 内): 第 1 个 RTT 之后, ACK 包开始源源不断地抵达发送方, 此时 ACC 只保留最大的视图 $View_{n+1}$, 网络不再传输 SAK 包, 减少了不必要的带宽开销。

Figure 8 ACC overview (in other RTTs of a flow): after the first RTT, ACK packets begin to return to the sender continuously, and ACC only maintains the largest view $View_{n+1}$. At this time, the network no longer performs the transmission of SAK packets, reducing unnecessary bandwidth overhead.

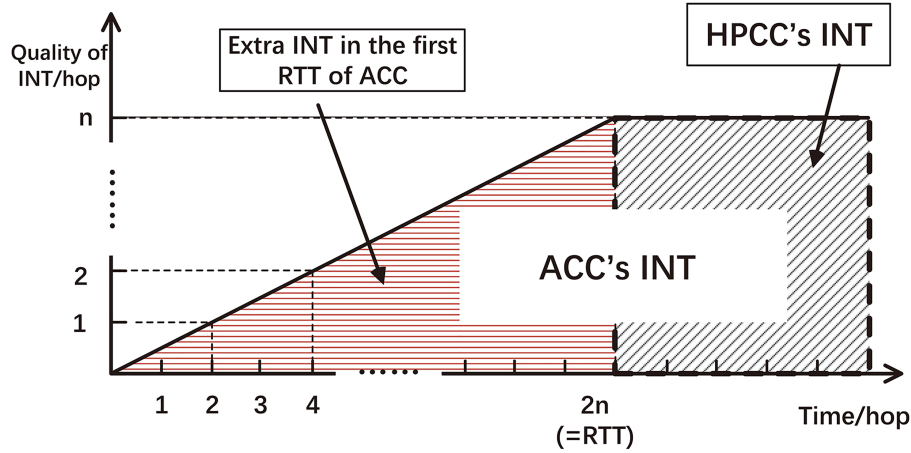


图9 (网络版彩图) 在ACC控制下, 流传输第1个RTT内INT信息的量随时间呈线性增长.

Figure 9 (Color online) In the first RTT of the flow, the amount of INT information increases linearly with time.

单视图的发送速率根据飞行中字节数计算^[3]. 如图7所示, 对于路径上除发送方之外的任何节点 M , 从发送方到 M 的局部路径构成视图 View_M , 在 View_M 中, 当前局部路径上最拥塞的链路的标准化的飞行中字节数被记为 $U_{\max M}$. 对于一条流, 通过将 $U_{\max M}$ 与预期的带宽利用率 η 进行比较, 可以更新发送窗口 W_M , 进一步可以计算发送速率 rate_M . 按照该计算方法, 在流传输的第1个RTT内, 发送方能够获得新鲜的网络负载信息并调节网络中的流量.

发送方收到确认包(SAK包或ACK包)后首先根据到达的INT信息更新VSet, 然后从VSet中选择发送速率最小的视图 $\text{View}_{\min}$. $\text{View}_{\min}$ 的发送速率 $\text{rate}_{\min}$ 表示在最拥堵的条件下网络允许的发送速率, 因此将 $\text{rate}_{\min}$ 设置为流的发送速率.

3.3.2 有限交换机INT技术: 从交换机INT到主机INT的过渡

发送方收到第1个ACK包后, 来自接收方的ACK包开始源源不断地到达发送方, 网络中的主机INT达到相对稳定的状态. 相关实验表明在流量稳定时, 使用主机INT的算法可以使网络中的队列长度接近于零^[3], 换句话说, 在流传输的第1个RTT之后, 主机INT能够满足数据中心网络拥塞控制的需求. 尽管交换机INT可以提供高新鲜度的链路负载信息, 但是大量的SAK包会带来额外的带宽开销, 因此在流传输的第1个RTT之后, ACC逐渐转向主机INT以获取精确的链路负载信息.

为了完成从交换机INT到主机INT的过渡并且尽量不影响发送方INT信息的新鲜度, 首先要弄清楚同一数据包触发的多个SAK包中INT信息的生成时间顺序. 从生成时间的角度看, SAK包、数据包和ACK包中搭载的同一链路的INT信息略有不同, 图10展示了ACC管理下网络传输一个带有带宽延迟积(bandwidth-delay product, BDP)标记的数据包的过程. 在数据包pkt到达Switch1之后, Switch1立即生成一个SAK包SAK1, 并在SAK1中标记当前的INT信息INT-t1(当前时刻t1的INT信息), 然后将SAK1发送回发送方, 之后pkt在交换机中排队. 当pkt离开队列时, Switch1在pkt的特定字段中标记t2时刻的INT信息INT-t2, 然后pkt继续在路径上传输. 当到达Switch2时, Switch2生成一个SAK包SAK2, 并将包含在pkt中的INT-t2和当前时刻t3的INT信息INT-t3搭载在SAK2中, 然后SAK2被传回发送方. 当pkt在Switch2的队列中出队时, Switch2在数据包的特定字段中标记当前时刻t4的INT信息INT-t4. 该过程一直持续到pkt到达接收方为止.

比较SAK1和SAK2中INT信息的生成时间, 对于Switch1的INT信息, SAK2中INT-t2的生成时间晚于SAK1中的INT-t1, 换句话说, 对于同一数据包触发的多个SAK包, 远离发送方的交换机会反馈生成时间较新的INT信息. 对于一条流, 假设其第1个BDP中最后一个数据包是pktlast, 在路径上最后一个交换机SwitchLast反馈SAK包SAKlast之后, 在SwitchLast之前的交换机返回

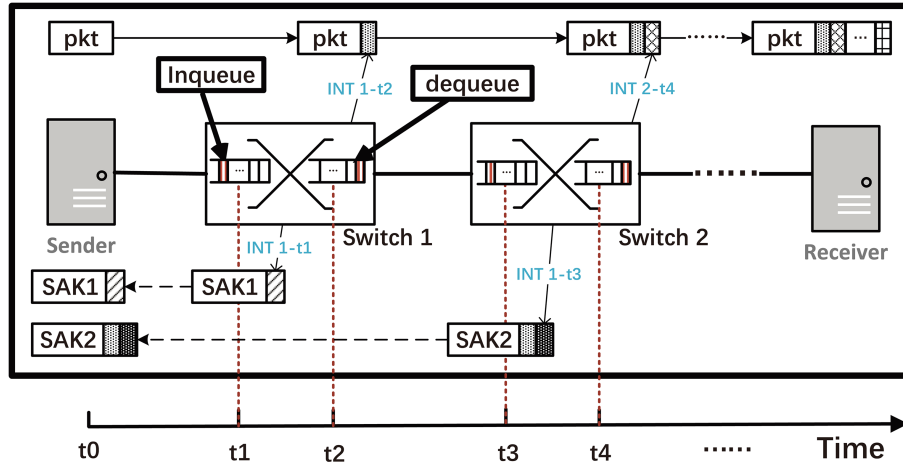


图 10 (网络版彩图) 同一个数据包触发的多个 SAK 包的时间序列.

Figure 10 (Color online) Time sequence of SAK packets triggered by the same data packet.

的 INT 信息就都是过期的了, 即在 VSet 中保存的 SwitchLast 之前的交换机信息也已过时, 当发送方收到 SAKlast, 过时的信息就可以被删掉了. 此外当发送方收到与 pktlast 相对应的 ACK 包 ACKlast 时, 表明流第 1 个 BDP 中的所有数据包已经发送完毕, 此时发送方不再需要维护交换机的信息, 并且可以清除 VSet, 从这时起, ACC 依靠主机 INT 来控制网络流量.

3.4 Cloak 机制

RTT 级别的链路负载信息反馈时延会导致流在传输完成后不可避免地出现带宽浪费. 当一条流即将传输完成时, 其占用的带宽资源也会被释放, 当流的最后一个数据包被发送, 发送方将不会再向网络中注入流量, 流占用的带宽资源也会逐段链路地被释放. 但是这些被释放的资源不会被立即重新利用, 这些带宽资源将保持空闲直到有其他流的数据包经过这段链路, 收集对应的 INT 信息, 并经由接收方的 ACK 包将 INT 信息传回相应的发送方, 才能够引发发送方增速, 重新抢占空闲带宽, 这一过程的时延是 RTT 级别的. 因此总的来说, 在 HPCC 的控制下, INT 信息的反馈时延导致流发送结束后出现 RTT 级别的带宽浪费现象, 这一点对于长流来说尤为显著.

针对上述带宽浪费的情况, ACC 采用 Cloak 机制帮助网络中其他的流量尽可能地提前抢占带宽. 具体来说, ACC 会对流尾部 RTT 内的数据包进行 Lst 标记, 交换机在收到带有 Lst 标记的数据包时, 会选择性忽视这些数据包对链路 INT 信息的影响. 这样一来, 对于网络中的其他流量, ACC 隐藏了流尾部 RTT 的发送行为, 实现了流传输完成事件的预反馈, 能够帮助其他流量提前抢占网络带宽资源.

3.5 ACC 中拥塞信号的新鲜度及带宽开销分析

关于 ACC 的新鲜度, 可以分两个阶段进行考虑. 在流传输的第 1 个 RTT 内, ACC 依赖交换机反馈 INT 信息, 高新鲜度的拥塞信号会被传回发送方用于即时的拥塞控制, 这一阶段拥塞信号的 t_{gap} 甚至可以达到 $\frac{1}{2n}\text{RTT}$, 即 ACC 为发送方提供了理论上最高新鲜度的 INT 信息. 在流传输的中间 RTT 和尾部 RTT, 对发送方来说, 接收方反馈的完整路径的 INT 信息已经开始源源不断地到达发送方, 发送方基于这些拥塞信号可以实现高精度、高收敛性的控制^[3], 因此在第 2 和第 3 阶段可以适当降低拥塞信号的新鲜度, 节省控制信息开销.

ACC 会额外引入交换机 SAK 包的开销, 具体来说, 交换机会为每条流第 1 个 RTT 内所发送的数据包产生 SAK 包, 当网络中长流连续发送或网络状态稳定时, SAK 包的数量会降低. 实验统计结果

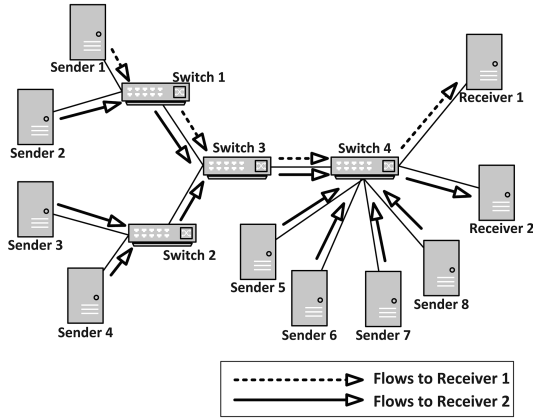


图 11 Burst 拓扑: Sender1~Sender4 发送长流, 网络达到稳定, Sender5~Sender8 在一个特定的时刻向 Receiver2 发送突发短流, Switch4 很快陷入拥塞, 甚至出现 PFC 暂停帧。

Figure 11 Burst topology: Sender1~Sender4 send long flows, and the network remains stable. Sender5~Sender8 send flows to Receiver2 at a certain moment. Then Switch4 quickly becomes congested, and even PFC pause frames appear.

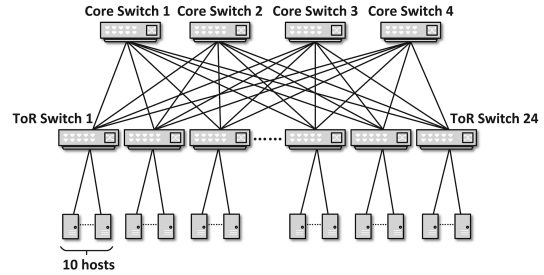


图 12 大规模 Clos 拓扑: 4 台核心交换机, 每台核心交换机连接 24 台 ToR 交换机, 每台 ToR 交换机连接 10 台主机。

Figure 12 Large-scale Clos topology: there are 4 core switches, each core switch is connected to 24 ToR switches, and each ToR switch is connected to 10 hosts.

表明, 当 BDP 为 30 kB (线速率为 10 Gbps, RTT 为 24 μ s) 时, 在 CacheFollower^[24] 和 WebServer^[25] 工作负载下, 有 BDP 标记的数据包占比分别为 2.15% 和 16.58%。可见即使在具有大量短流且平均流长度短的 WebServer 中, 也只有大约 16.6% 的数据包会触发 SAK 包, 在其他工作负载中该比例通常很低, 交换机确认包带来的带宽开销在大规模数据中心网络中是完全可以接受的。

4 实验评估

本工作采用常见数据中心网络工作负载进行大量的实验以验证 ACC 的传输性能, 选取的对照算法包括 HPCC^[3] 和 APCC^[26], 实验在 OMNeT++ 模拟平台^[27] 上开展, 使用 INET 框架^[8]。

4.1 实验设置

拓扑. 实验用到 4 种拓扑。第 1 个是图 1 所示的半哑铃拓扑, 其中 Switch3 和 Switch4 之间的链路带宽为 100 Gbps, 其他链路带宽为 40 Gbps, 任意两个相邻节点之间的延迟 (处理延迟 + 传播延迟) 为 1 μ s, Sender1~Sender6 将数据随机发送到 Receiver1~Receiver6。第 2 个是图 3 所示的 Four-to-one 拓扑, 该拓扑任意两个相邻节点之间的带宽和延迟分别是 40 Gbps 和 1 μ s。第 3 个是图 11 所示的 Burst 拓扑, 该拓扑任意两个相邻节点之间的带宽和延迟分别是 40 Gbps 和 1 μ s。该拓扑有 4 台交换机和 10 台主机, 其中 Sender1 的目的主机是 Receiver1, Sender2~Sender8 的目的主机是 Receiver2。第 4 个是图 12 所示的包含 240 台主机的大规模 Clos 拓扑, 其中 4 台核心交换机连接 24 台 ToR 交换机, 每台 ToR 交换机连接 10 台主机, ToR 交换机与核心交换机之间的链路带宽为 100 Gbps, ToR 交换机与主机之间的链路带宽为 10 Gbps, 任意两个相邻节点间的延迟为 3 μ s。

流量模式. 实验用到 3 种常见的数据中心网络工作负载: CacheFollower^[24], WebServer^[25] 以及 WebSearch^[25], 这些工作负载的组成如表 1 所示。

参数设置. HPCC^[3] 和 APCC^[26] 根据论文设置参数: $\text{maxStage} = 5$, $\eta = 0.95$, $W_{AI} = \text{linkspeed} \times \text{RTT} \times (1 - \eta)/n$, 这里 n 是网络中发送方的数量。为了实验的公平性, ACC 中每个视图的参数设置与

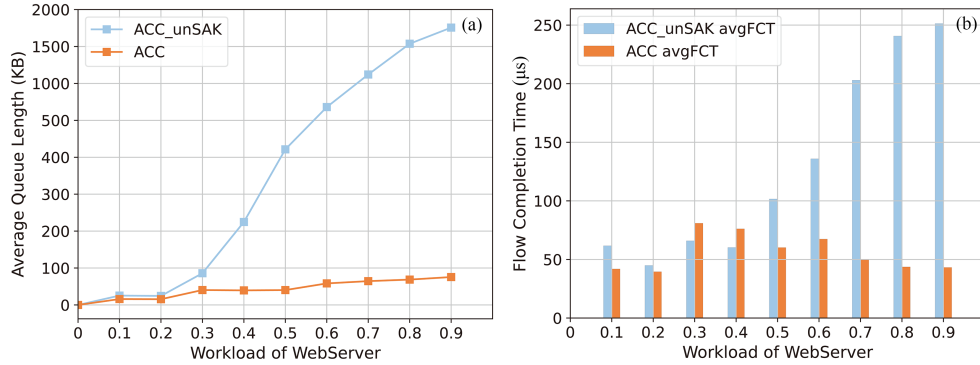


图 13 (网络版彩图) 交换机早期反馈机制的消融实验验证. (a) 不同负载强度下 Switch1 和 Switch2 的平均出口端口队列长度情况; (b) 流完成时间情况 (半哑铃拓扑, WebServer 工作负载).

Figure 13 (Color online) Ablation experiment of the switch early feedback mechanism. (a) The average egress port queue length of Switch1 and Switch2; (b) flow completion time under different load intensities (half-dumbbell topology, WebServer workload).

前述相同.

评估方案. 为了评估 ACC 的性能, 我们进行了 3 个方面的实验: (1) 设置消融实验验证交换机早期反馈机制和 Cloak 机制的有效性; (2) 评估 ACC 的微基准性能; (3) 评估 ACC 在大规模 Clos 拓扑里的传输性能.

4.2 ACC 核心机制的消融实验验证

为了验证 ACC 两个核心机制的有效性, 我们分别对交换机早期反馈机制和 Cloak 机制进行消融实验. 这里需要说明的一点是, 由于仿真时间的差异, 本文对比实验结果与 4.3 小节中 ACC 的部分具体性能数据会略有不同.

4.2.1 交换机早期反馈机制的消融实验验证

我们设置两个拥塞控制算法来验证交换机早期反馈机制的有效性, 其中一个是 ACC, 另一个是未开启交换机早期反馈机制的 ACC_unSAK. 实验在图 1 所示的半哑铃拓扑上开展, 我们使用 WebServer 工作负载, 调整负载强度为 0.1~0.9, 设置 9 组实验场景, 测试在 ACC 和 ACC_unSAK 控制下 Switch1 和 Switch2 的平均出口端口队列长度, 同时测试在 ACC 和 ACC_unSAK 控制下的流完成时间, 实验结果如图 13 所示.

图 13(a) 展示了 Switch1 和 Switch2 的平均出口端口队列长度, 实验结果显示当负载强度从 0.4 提升至 0.7, ACC_unSAK 控制下的瓶颈链路队列长度提升 399%, 而 ACC 控制下的瓶颈链路队列长度仅提升 63.6%, 当负载强度高达 0.9 时, ACC_unSAK 的瓶颈链路队列长度更是高达 1755.578 kB. 由此可见交换机早期反馈机制可以帮助 ACC 大大降低瓶颈链路的队列累积. 图 13(b) 展示了流完成时间的情况, 由实验结果可以看出交换机早期反馈机制能够帮助 ACC 降低网络流完成时间, 这一点在网络负载较高时表现得更加显著.

4.2.2 Cloak 机制的消融实验验证

我们设置两个拥塞控制算法来验证 Cloak 机制的有效性, 其中一个是 ACC, 另一个是未开启 Cloak 机制的 ACC_unCloak. 实验在图 3 所示的 Four-to-one 拓扑上开展, 我们令 4 台发送方发送 4 条长流, Sender1~Sender4 所对应的流分别为 Flow1~Flow4, 其中 Flow1~Flow4 流长度递增. 当网络达到稳定状态后 4 条流平分网络带宽, 各自 10 Gbps, 随后我们观察 Flow1~Flow3 依次完成发送后 Flow4 对带宽的抢占情况, 实验结果如图 14 所示.

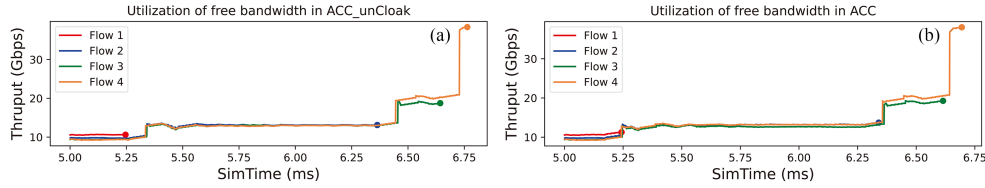


图 14 (网络版彩图) Cloak 机制的消融实验验证: 与 ACC_unCloak 相比, ACC 能够使 Flow4 提前 5.8 个 RTT 完成传输 (Four-to-one 拓扑). (a) ACC_unCloak 的带宽抢占情况; (b) ACC 的带宽抢占情况.

Figure 14 (Color online) Ablation experiment of the Cloak mechanism: compared with ACC_unCloak, ACC enables Flow4 to complete its transmission 5.8 RTTs in advance (Four-to-one topology). (a) The bandwidth preemption in ACC_unCloak; (b) the bandwidth preemption in ACC.

图 14(a) 展示的是 ACC_unCloak 控制下流传输过程中各条流吞吐量随时间的变化情况, 图中圆点表示流传输完成事件, 图 14(b) 展示的是 ACC 控制下的对应情况. 由实验结果可见, 在每次有流完成发送时, Cloak 机制能够帮助 ACC 迅速抢占空闲带宽, 使网络中的流在较短的时间内重新收敛, 更进一步地, 我们观察 Flow4 的传输完成时间, 可知与 ACC_unCloak 相比, ACC 能够使 Flow4 提前 5.8 个 RTT 完成传输.

4.3 微基准实验评估

4.3.1 避免突发流量拥塞: ACC 对短流的控制

我们在图 1 所示的半哑铃拓扑上比较 ACC 和 HPCC 对短流的控制能力, 工作负载选择短流占比比较高的 WebServer.

实验记录不同负载强度下 Switch1 和 Switch2 的平均出口端口队列长度, 结果如图 15 所示. 与 HPCC 相比, 在 ACC 的控制下尽管随着负载强度的增加队列长度也在增长, 但由于交换机提供了及时的反馈, 整体的队列长度明显降低. 具体来看, 当负载强度小于 0.5 时, ACC 能够将队列长度减少至少 30%, 当负载强度较高时, ACC 能够将队列长度减少 49.3%~68.8%.

4.3.2 避免带宽浪费: ACC 对长流的控制

我们在图 3 所示的 Four-to-one 拓扑上测试 ACC 对长流的控制能力. 实验场景如 2.2 小节中所述, 4 个发送方同时向接收方发送长流, 在稳定状态下各发送方平分链路带宽, 每台 10 Gbps, 在某一时刻我们令 3 个发送方 (Sender1~Sender3) 同时停止发送, 观察在 ACC 控制下第 4 个发送方 (Sender4) 对空闲带宽的抢占能力.

图 16 展示了该实验的结果, 图中紫色虚线表示 Sender1~Sender3 的吞吐量变化情况, 蓝色实线表示 ACC 控制下 Sender4 的吞吐量变化情况. 我们控制 Sender1~Sender3 都在约 548 μ s 时终止发送, 实验结果显示在 ACC 的控制下, Sender4 在约 597 μ s 时完成空闲流量的抢占, 吞吐量达到 40 Gbps. 该网络的 RTT 是 12 μ s, 抢占时间约为 4 个 RTT, 相比于 HPCC 所需的 7.4 个 RTT, 缩短了 45.9%.

4.3.3 避免 PFC 暂停帧的传播

数据中心网络通常在链路层部署无损的流量控制机制以避免数据包丢失, 例如 PFC 机制. 当拥塞严重时, PFC 会驱使拥塞节点向上游端口发送 PFC 暂停帧^[4], 中止流的注入以确保交换机缓冲区不会溢出, 从而避免数据包丢失. 但是 PFC 具有副作用, 当拥塞严重时 PFC 暂停帧会导致拥塞从拥塞节点向源节点蔓延, 形成以拥塞节点为根的拥塞树^[28], 损害正常流的传输性能.

考虑短流突发的场景, 在图 11 所示的拓扑中有 4 台交换机和 10 台主机, 链路带宽是 40 Gbps, 相邻节点之间延迟为 1 μ s. 在初始状态时, Sender1 向 Receiver1, Sender2~Sender4 向 Receiver2 发送连续的长流, 网络达到稳定, 在某个特定的时刻 t , Sender5~Sender8 开始同时向 Receiver2 发送突发的短

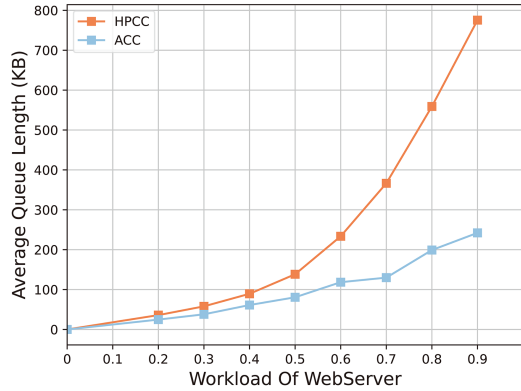


图 15 (网络版彩图) 不同负载强度下 Switch1 和 Switch2 的平均出口端口队列长度 (半哑铃拓扑, WebServer 工作负载).

Figure 15 (Color online) Average queue length of Switch1 and Switch2 under different intensity loads (half-dumbbell topology, WebServer workload).

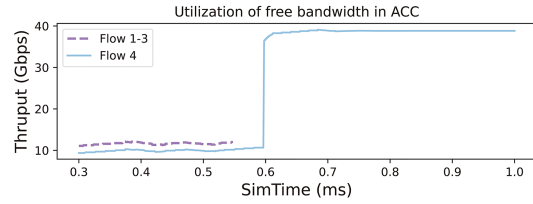


图 16 (网络版彩图) ACC 中空闲带宽的使用情况: ACC 的抢占时间为 4 个 RTT (four-to-one 拓扑).

Figure 16 (Color online) Utilization of free bandwidth in ACC: the preemption time is 4 RTTs (four-to-one topology).

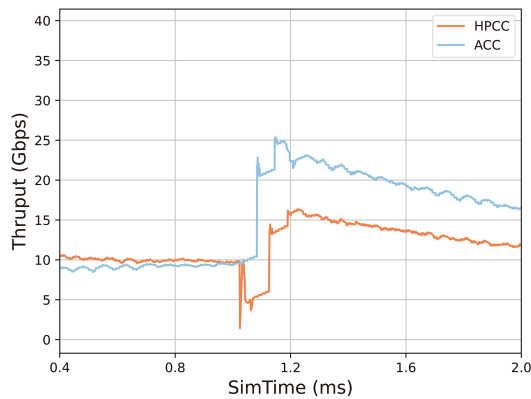


图 17 (网络版彩图) 突发流开始时 Sender1 的吞吐量 (Burst 拓扑).

Figure 17 (Color online) Throughput of Sender1 as burst flows start (Burst topology).

流. 由于 HPCC 需要 ACK 包来传输路径 INT 信息, Sender5~Sender8 至少需要一个 RTT 来获取负载信息, 因此这后续的 4 个发送方在至少一个 RTT 的时间里以线速率发送数据. 这样一来, Switch4 连接 Receiver2 的出口端口会在短时间内累积超长的队列, 这种情况下网络中的突发流量在交换机处产生了过长的队列造成网络拥塞, 损害了整体的网络传输性能, 并且甚至可能触发 PFC 暂停帧. PFC 暂停帧会不加区分地暂停拥塞交换机上游的流量, 在图 11 所示的 Burst 拓扑中, 如果在 Switch4 处 PFC 暂停帧被触发, 向着发送方的方向, 更多的 PFC 暂停帧可能被进一步触发, 形成拥塞树. 尽管 Sender1 的目的主机是 Receiver1, 并且连接 Switch4 和 Receiver1 的端口是空闲的, Sender1 仍会因为 PFC 暂停帧的传播而被暂停, 换句话说, 即使 Sender1 到 Receiver1 的路径上没有拥塞, Sender1 也会因为暂停帧而减速, 使得 Sender1 成为受害者流.

在图 11 所示的 Burst 拓扑中, 令 Sender1~Sender4 发送长流, Sender5~Sender8 在 1 ms 的时候发送 40 kB 的短流 (以线速率发送两个 RTT, 即 8 μ s), Sender1 的吞吐量如图 17 所示, 我们发现与 HPCC 相比, ACC 可以在一定程度上避免 PFC 暂停帧的触发.

具体来说, 对于 HPCC, 1 ms 时刻的突发流量导致队列累积过长并且 PFC 暂停帧被触发, Sender1 受暂停帧影响在 1015 μ s 时减速, 直到突发流量结束链路相对空闲后, Sender1 在 1146 μ s 时才开始升

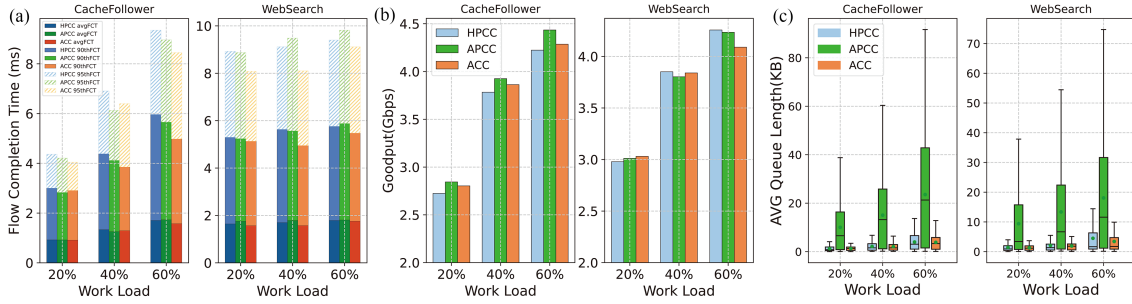


图 18 (网络版彩图) Clos 拓扑上常见工作负载的 (a) 流完成时间 (包括平均流完成时间、第 90 百分位流完成时间和第 95 百分位流完成时间), (b) 吞吐量, 以及 (c) 交换机平均队列长度分布 (大规模 Clos 拓扑, CacheFollower 工作负载和 WebSearch 工作负载).

Figure 18 (Color online) (a) FCTs (including average FCTs, 90th-percentile FCTs, and 95th-percentile FCTs), (b) goodputs and (c) the switch average queue length distribution under different workload intensities on Clos topology (large-scale Clos topology, CacheFollower workload & WebSearch workload).

速, 随后速率慢慢收敛. 对于 ACC, 由于交换机的提前反馈, 发送方能够在流传输的第 1 个 RTT 内检测和调控拥塞, 因而 Sender1 没有成为受害者, 相反, 当 Sender2~Sender4 由于拥塞而降速后, Sender1 到 Receiver1 的路径上出现空闲带宽, 实验结果显示 Sender1 此时抢占了部分空闲带宽, 在 1080 μ s 时升速, 等突发流量结束后速率慢慢收敛.

4.4 使用常见的数据中心网络工作负载在大规模 Clos 拓扑上的实验评估

我们在图 12 所示的大规模 Clos 拓扑上评估了 ACC 的传输性能, 该拓扑包含 4 台核心交换机和 24 台 ToR 交换机, 每台 ToR 交换机连接 10 台主机, 这是数据中心网络中一种常用的拓扑. 在 Clos 拓扑中, 每台主机随机地向其他任意一台主机发送数据, 实验选取 HPCC^[3] 和 APCC^[26] 为对照算法, 用到两种常见的工作负载: CacheFollower^[24] 和 WebSearch^[25].

Clos 拓扑上常见数据中心网络工作负载的流完成时间. 我们在 3 种不同负载强度下使用两种工作负载分别测量了平均、第 90 百分位、第 95 百分位的流完成时间, 结果如图 18(a) 所示. 实验结果显示与 HPCC 和 APCC 相比, ACC 的平均、第 90 百分位和第 95 百分位流完成时间均有所降低. 以负载强度为 0.6 的 CacheFollower 工作负载为例, 与 HPCC 相比, ACC 的平均流完成时间降低约 7.6%, 第 90 百分位流完成时间降低约 16.6%, 第 95 百分位流完成时间降低约 9.6%, 与 APCC 相比, ACC 的平均流完成时间降低约 9.2%, 第 90 百分位流完成时间降低约 12%, 第 95 百分位流完成时间降低约 5.7%. 另外我们发现随着网络传输时间的增长 ACC 对流传输尾延迟的降低效果会更加显著, 我们在负载强度为 0.6 的 WebServer 工作负载上运行较长时间的网路传输后观测到, 与 HPCC 相比, ACC 能将平均队列长度减少 9.6%, 将第 95 百分位流完成时间减少约 29.7%.

平均流完成时间的降低表示 ACC 能够降低网络流传输所需时间的整体水平, 第 90 百分位和第 95 百分位流完成时间的降低表示 ACC 明显优化了网络流传输的尾延迟, 因此我们认为在交换机早期反馈机制和 Cloak 机制的优化下, ACC 对网络流量有了更强的控制能力, 数据中心网络的传输性能有了较大的提升.

Clos 拓扑上常见数据中心网络工作负载的吞吐量. 我们在 3 种不同的负载强度下使用两种工作负载, 测量了吞吐量, 结果如图 18(b) 所示. APCC 依靠交换机在流传输的全过程中反馈 SAK, 因此能够一直向发送方提供高新鲜度的 INT 信息, 其在网络吞吐量方面表现优异, 但这是以网络中队列堆积为代价的, 这一点我们会在 Clos 拓扑上队列长度部分的实验中详细分析. 相比之下, ACC 兼顾了带宽抢占和控制信息开销, 可以实现优于 HPCC 的吞吐量表现.

Clos 拓扑上常见数据中心网络工作负载的队列长度. 我们在 3 种不同的负载强度下使用两种工

作负载,测量了最小、第 25 百分位、第 50 百分位、第 75 百分位、最大和平均的交换机平均队列长度情况,结果如图 18(c) 所示. 在箱线图中,上下边缘代表了该组数据的最大值和最小值,箱子的上下底分别是数据的上四分位数 (第 75 百分位数) 和下四分位数 (第 25 百分位数),这意味着箱体包含了 50% 的数据,箱子的中间有一条线,代表数据的中位数,箱子中的绿点表示数据的平均值. 总体上,箱子的高度在一定程度上反映了数据的波动程度.

实验结果显示与 HPCC 相比, ACC 总体上保持着较低的队列水平,这是由于交换机早期反馈机制能够对网络流量进行更细粒度的控制,从而降低了队列堆积. 但是在一些长流多发的情况下,我们观察到 ACC 的队列堆积略高于 HPCC,这是因为 ACC 采用 Cloak 机制屏蔽了部分流传输信息,较为激进地实现了对网络带宽的抢占,因而会造成一定的队列堆积. 相比之下, APCC 则呈现出较高的队列堆积水平,这是因为该算法在流传输的全过程中依靠交换机发送 SAK 包传输拥塞信号,因而网络中会存在数量较多的控制包,造成较高的队列堆积.

5 总结

本论文提出了一种名为 ACC 的敏捷的数据中心网络拥塞控制算法. 一方面, ACC 使用交换机早期反馈机制在流传输的第 1 个 RTT 内传递链路负载信息. 通过结合交换机 INT 和主机 INT, ACC 实现了对流量的超精准控制,打破了传统的基于 INT 信息的解决方案控制回路为 RTT 级别的限制,允许拥塞控制用一种更细粒度的方式管理网络流量. 另一方面, ACC 依赖 Cloak 机制实现了对流完成事件的预反馈,能够向网络预告即将空闲的带宽. Cloak 机制帮助网络中的流尽快抢占带宽,确保了数据中心网络的高吞吐量和低延迟.

我们相信本研究能够为大规模高速数据中心网络中基于 INT 信息的拥塞控制算法的研究提供引导,在未来的工作中,我们会就定长的 INT 信息、INT 信息高效压缩以及 Cloak 机制的屏蔽尺寸等方面展开研究,探究更有效的数据中心网络拥塞控制技术.

参考文献

- 1 Cisco. 400G data center and cloud networking. 2022. <https://www.cisco.com/c/en/us/solutions/data-center/high-capacity-400g-data-center-networking/index.html>
- 2 Pan S, Stavrinou T, Zhang Y Q, et al. Facebook's tectonic filesystem: efficiency from exascale. In: Proceedings of the 19th USENIX Conference on File and Storage Technologies, 2021. 217–231
- 3 Li Y L, Miao R, Liu H Q, et al. HPCC: high precision congestion control. In: Proceedings of the ACM Special Interest Group on Data Communication, 2019. 44–58
- 4 Zhu Y B, Eran H, Firestone D, et al. Congestion control for large-scale rdma deployments. In: Proceedings of the ACM Conference on Special Interest Group on Data Communication, 2015. 523–536
- 5 Alizadeh M, Kabbani A, Edsall T, et al. Less is more: trading a little bandwidth for ultra-low latency in the data center. In: Proceedings of the 9th USENIX Conference on Networked Systems Design and Implementation, 2012. 253–266
- 6 Alizadeh M, Yang S, Sharif M, et al. pFabric: minimal near-optimal datacenter transport. SIGCOMM Comput Commun Rev, 2013, 43: 435–446
- 7 Benson T, Akella A, Maltz D A. Network traffic characteristics of data centers in the wild. In: Proceedings of the 10th ACM SIGCOMM Conference on Internet Measurement, 2010. 267–280
- 8 INET Framework. Inet. 2022. <https://inet.omnetpp.org/>
- 9 Yuan G Y, Zhou R J, Dong D Z, et al. Breaking one-rtt barrier: ultra-precise and efficient congestion control in datacenter networks. In: Proceedings of International Conference on Computer Communications and Networks (ICCCN), 2021. 1–9
- 10 Floyd S. TCP and explicit congestion notification. SIGCOMM Comput Commun Rev, 1994, 24: 8–23
- 11 Kumar G, Dukkipati N, Jang K, et al. Swift: delay is simple and effective for congestion control in the datacenter. In: Proceedings of the Annual Conference of the ACM Special Interest Group on Data Communication on the Applications,

- Technologies, Architectures, and Protocols for Computer Communication, 2020. 514–528
- 12 Mittal R, Lam V T, Dukkipati N, et al. TIMELY: RTT-based congestion control for the datacenter. In: Proceedings of the ACM Conference on Special Interest Group on Data Communication, 2015. 537–550
 - 13 Liang J Z, Bi J, Zhou Y, et al. In-band network function telemetry. In: Proceedings of the ACM SIGCOMM 2018 Conference on Posters and Demos, 2018. 42–44
 - 14 Nature Research. In-band network telemetry in broadcom tomahawk 3. <https://www.broadcom.com/company/news/product-releases/2372840>
 - 15 Nature Research. In-band network telemetry in broadcom trident 3. <https://www.broadcom.com/blog/>
 - 16 Basat R B, Ramanathan S, Li Y L, et al. PINT: probabilistic in-band network telemetry. In: Proceedings of the Annual Conference of the ACM Special Interest Group on Data Communication on the Applications, Technologies, Architectures, and Protocols for Computer Communication, 2020. 662–680
 - 17 Cui M W, Li X, Wang Y, et al. SPT: sketch-based polling in-band network telemetry. In: Proceedings of IEEE/IFIP Network Operations and Management Symposium, 2022. 1–7
 - 18 Arslan S, Li Y L, Kumar G, et al. Bolt: sub-RTT congestion control for ultra-low latency. In: Proceedings of NSDI, 2023. 219–236
 - 19 Abdous S, Sharafzadeh E, Ghorbani S. Burst-tolerant datacenter networks with vertigo. In: Proceedings of the 17th International Conference on Emerging Networking EXperiments and Technologies, 2021. 1–15
 - 20 Wang M W, Huang S J, Cui Y, et al. Learning buffer management policies for shared memory switches. In: Proceedings of IEEE Conference on Computer Communications, 2022. 730–739
 - 21 Miao R, Zhu L J, Ma S, et al. From luna to solar: the evolutions of the compute-to-storage networks in Alibaba cloud. In: Proceedings of the ACM SIGCOMM 2022 Conference, 2022. 753–766
 - 22 Chen B R, Liu Z T, Song J H, et al. FlowTele: remotely shaping traffic on internet-scale networks. In: Proceedings of the 18th International Conference on Emerging Networking EXperiments and Technologies, 2022. 349–368
 - 23 HUAWEI. Huawei 6866 datacenter switch. 2021. https://e.huawei.com/mediares/Marketing-Material_MCD/EBG/PUBLIC/zh/2020/12/5e68b1a5-7c03-4724-a640-926462874f30.pdf
 - 24 Roy A, Zeng H Y, Bagga J, et al. Inside the social network's (datacenter) network. In: Proceedings of the ACM Conference on Special Interest Group on Data Communication, 2015. 123–137
 - 25 Alizadeh M, Greenberg A, Maltz D A, et al. Data center TCP (DCTCP). SIGCOMM Comput Commun Rev, 2010, 40: 63–74
 - 26 Zhou R J, Yuan G Y, Dong D Z, et al. APCC: agile and precise congestion control in datacenters. In: Proceedings of IEEE International Conference on Parallel & Distributed Processing with Applications, Big Data & Cloud Computing, Sustainable Computing & Communications, Social Computing & Networking (ISPA/BDCloud/SocialCom/SustainCom), 2020. 649–656
 - 27 Nature Research. Omnet++. 2020. <https://omnetpp.org/>
 - 28 Huang S, Dong D, Bai W. Congestion control in high-speed lossless data center networks: a survey. Future Generation Comput Syst, 2018, 89: 360–374

ACC: agile congestion control in datacenter networks

Guoyuan YUAN¹, Yuan LU¹, Renjie ZHOU², Dezun DONG^{1*} & Wei PENG¹

1. School of Computer Science, National University of Defense Technology, Changsha 410028, China;

2. Northwest Institute of Nuclear Technology, Xi'an 710024, China

* Corresponding author. E-mail: dong@nudt.edu.cn

Abstract Faced with the increasingly complex and rapidly changing datacenter network traffic, congestion control technology based on INT (in-network telemetry) information faces many challenges. On the one hand, the existing state-of-the-art algorithms mainly rely on the receiver's ACK (acknowledge) packet to feedback INT information to the sender to obtain link load information. Since the feedback delay of the ACK packet is at the RTT (round trip time) level, the sender cannot obtain INT information within the first RTT of the flow transmission. On the other hand, the existing algorithms generally lack the ability to seize the idle network bandwidth when the flow transmission is completed, so the network would inevitably waste bandwidth. In this paper, we explore INT-based techniques, break the one-RTT barrier efficiently and eliminate bandwidth waste. We come up with an ultra-precise and efficient congestion control algorithm, called ACC (agile congestion control). Compared with the existing state-of-the-art INT-based congestion control algorithm HPCC (high precision congestion control), ACC pays special attention to the first RTT of the flow transmission. During this period, the switch is used to transmit the path congestion signal back to the sender, which can respond more quickly to network congestion. In addition, ACC pays attention to the bandwidth preemption ability of long flows in the network, which can help long flows perceive idle bandwidth in advance, thereby reducing the tail delay of network transmission. Furthermore, we conduct extensive experiments to evaluate the performance of our design. The result shows that while keeping the same high throughput as HPCC, ACC can maintain a lower queue length and effectively reduce the tail delay of the flow. Specifically, in large-scale Clos topology, when the workload is WebServer and the workload intensity is 0.6, compared with HPCC, ACC can reduce the average queue length by 9.6%, shorten the 95th-percentile flow completion time by 29.7%.

Keywords datacenter congestion control, in-network telemetry (INT), bandwidth preemption, coordination of switch and host