Cubic+:用于跨数据中心网络的改进Cubic拥塞控制算法

龙铁, 肖甫, 樊卫北, 何昕, 王俊昌

引用本文

龙铁, 肖甫, 樊卫北, 何昕, 王俊昌. Cubic+:用于跨数据中心网络的改进Cubic拥塞控制算法[J]. 计算机科学, 2025, 52(8): 335-342.

LONG Tie, XIAO Fu, FAN Weibei, HE Xin, WANG Junchang. Cubic+:Enhanced Cubic Congestion Control for Cross-datacenter Networks [J]. Computer Science, 2025, 52(8): 335-342.

计算机科学 COMPUTER SCIENCE
http://www.jsjkx.com
DOI:10.11896/jsjkx.250100056

Cubic+:用于跨数据中心网络的改进 Cubic 拥塞控制算法

龙 铁 肖 甫 樊卫北 何 昕 王俊昌
南京邮电大学计算机学院 南京 210003
(tiefirst@163.com)

摘 要 跨数据中心网络是处于不同地区的数据中心网络(Data Center Networks,DCNs)通过广域网(Wide-Area Network,WAN)连接组成的网络,分布式应用通常基于该网络为用户提供高质量的服务。DCNs 和 WAN 的缓冲区大小、往返时延存在显著差异,这导致现有的 Cubic 拥塞控制算法在跨数据中心网络场景下出现降速不准确、丢包率过高以及与其他拥塞控制算法兼容性差等问题。针对以上挑战,提出了一种通过匹配不同发送速率模式的改进 Cubic 算法 Cubic+。具体地,Cubic+整合了网络中的时延、ECN(Explicit Congestion Notification)和丢包信号。当拥塞发生在浅缓冲交换机时,Cubic+会周期性地排空队列;当拥塞发生在深缓冲路由器时,Cubic+会快速减少堆积的数据包。基于大规模 NS3 仿真实验结果表明,在跨数据中心网络流量模型下,Cubic+与现有流行算法相比,平均流完成时间最多减少了 20.77%,第 99 百分位流完成时间最多减少了 15.88%,为跨数据中心网络提供了一种高吞吐的拥塞控制算法。

关键词:数据中心网络;拥塞控制;吞吐公平性;TCP 传输;广域网
中图分类号 TP393

Cubic+:Enhanced Cubic Congestion Control for Cross-datacenter Networks

LONG Tie,XIAO Fu,FAN Weibei,HE Xin and WANG Junchang
School of Computer Science,Nanjing University of Posts and Telecommunications,Nanjing 210003,China

Abstract Cross-datacenter networks connect data center networks(DCNs) across regions via wide-area networks(WANs),supporting distributed applications to deliver high-quality services. However,differences in buffer sizes and round-trip times between DCNs and WANs challenge the Cubic algorithm,leading to inaccurate rate reductions,high packet loss,and poor compatibility with other algorithms. To address these challenges,this paper proposes Cubic+,an improved version of Cubic that adapts to different congestion patterns. Specifically,Cubic+ integrates delay,ECN(Explicit Congestion Notification),and packet loss signals. Cubic+ adapts to shallow-buffered switch congestion by periodically emptying queues and quickly reduces packet backlogs for deep-buffered routers. Large-scale NS3 simulations show Cubic+ reduces average flow completion time by up to 20.77% and 99th percentile completion time by 15.88%,offering a high-throughput solution for cross-datacenter networks.

Keywords Datacenter networks,Congestion control,Throughput fairness,TCP transmission,Wide-area networks

1 引言

随着云计算的快速发展,跨数据中心网络在支持各种网络应用和服务等方面扮演着关键的角色。例如,谷歌支撑了成千上万个分布式计算应用程序在全球不同的数据中心网络之间进行频繁通信[1]。此外,用户流量的激增对带宽的诉求也在过去 5 年内增长了 100 倍[2]。因此,跨数据中心网络需要高吞吐、低时延的拥塞控制算法来满足日益增长的上层应用需求。

跨数据中心网络由两个处于异地的数据中心网络通过广域网连接组成。其特点是采用异构缓冲的网络架构,混合部署了具有不同缓冲区容量的路由器和交换机。具体而言,在数据中心网络与广域网的连接处部署具有深缓冲区的路由器,而在数据中心网络其他链路部署具有浅缓冲区的交换机[3]。

缓冲区大小是影响拥塞控制算法性能的关键因素。普遍的观点认为,浅缓冲区设计部署成本低,并有助于降低时延,但容易丢包进而降低吞吐;深缓冲区能减少丢包和提高吞吐,但其造价昂贵,还可能引起过高的排队时延[4]。过去 20 年来,已有大量研究探讨了交换机/路由器缓冲区容量与拥塞控制算法性能之间的关系[5-7]。然而,目前仍未能形成一个通用的结论来指导在不同场景下缓冲区的设置,从而难以实现高吞吐、低时延的性能目标。

此外,跨数据中心网络中不同商用路由器/交换机的缓冲

到稿日期:2025-01-09 返修日期:2025-05-08
基金项目:江苏省自然科学基金重点研发计划(SBK2024100096)
This work was supported by the Key R&D Program of Jiangsu Provincial Natural Science Foundation(SBK2024100096).
通信作者:肖甫(xiaof@njupt.edu.cn)

区容量差异巨大，每个端口缓冲容量可相差 1 000 倍[8]，从而导致当前 TCP 拥塞控制算法的性能在跨数据中心网络中未能达到预期。导致这些问题的根本原因是，不同拥塞控制算法为达到满带宽利用率，其速率调整机制对瓶颈交换机缓冲区大小的需求也不同。本文重点关注 Cubic 算法[9]在跨数据中心网络中的性能。Cubic 是当前 Linux 系统中默认的 TCP 拥塞控制算法，也是广域网中使用率最高的算法。文献[10]统计了主流网站中不同拥塞控制算法的使用频率，Cubic 占比最高，达到 36%，BBR 算法[11]次之，达到了 22%。以前的研究[12-13]已指出 Cubic 在传输数据时存在的问题：1）Cubic 无法识别不同的丢包场景，并做出对应的降速行为，可能导致吞吐率下降；2）当 Cubic 与 BBR 共存时，Cubic 的带宽可能会被 BBR 抢占。

为了解决跨数据中心网络传输性能问题，近年来研究者提出了多种新的拥塞控制算法，如 Annulus[14]，GTCP[15]。然而，上述算法的共同问题是速率控制机制过于复杂，且未考虑与其他拥塞控制算法共存时的兼容性。基于上述问题，本文方案旨在通过简单且实用的方式改进 Linux 内核默认的 Cubic 拥塞控制算法，同时兼容 Cubic 的原始版本，在与 BBR 共存的跨数据中心网络中实现高吞吐和低排队时延。

本文通过 NS3 仿真重新研究了 Cubic 在典型的缓冲异构广域网场景——跨数据中心网络的吞吐表现。仿真结果表明，Cubic 的吞吐受到跨数据中心网络架构和 BBR 算法的影响。造成上述问题的根本原因在于：1）Cubic 对待所有丢包事件的响应行为都是一致的，但发生在浅缓冲区和深缓冲区的丢包对网络吞吐的影响是不同的；2）Cubic 在遇到网络中的随机丢包事件时，会大幅降低吞吐；3）尽管 Cubic 具备 ECN 标记响应机制，但在跨数据中心网络中，难以设置合理的 ECN 阈值来应对混合流量带来的压力。

为了解决上述问题，本文提出了一种简单且实用的 Cubic 改进算法，该方法旨在改善传统 Cubic 算法在跨数据中心网络中过度降低发送速率以及被 BBR 算法不公平地抢占带宽的现象。Cubic＋的核心思想是整合传输链路中的丢包信号、ECN 信号和时延信号，根据网络状况动态调整拥塞窗口。与传统的 Cubic 算法不同，Cubic＋在遇到丢包事件或 ECN 标记时，并不是直接进行乘法减小窗口操作，而是首先判断是否需要减少窗口，再根据情况决定窗口的减小幅度。通过融合 3 种拥塞控制信号，Cubic＋能够解决传统 Cubic 在跨数据中心网络中遇到的 3 个局限：1）Cubic＋能够准确计算排队时延，并区分丢包发生在浅缓冲还是深缓冲交换机中，从而选择正确的降速模式，避免过于缓慢的降速或者不必要的降速；2）即便交换机没有开启 ECN 功能，Cubic＋也能在源端检测到丢包前，及时清空深缓冲区中堆积的数据包，减小深缓冲区数据包的排队时延；3）Cubic＋能够过滤广域网中的随机丢包事件，从而避免不必要的吞吐损失。

本文工作的关键贡献如下：

1）通过一系列完整的实验，分析了 Cubic 和 BBR 在跨数据中心网络共享瓶颈链路时，吞吐量急剧下降、传播时延显著增加的原因。

2）针对跨数据中心网络，本文在 Cubic 算法基础上提出了 Cubic＋算法。Cubic＋在检测到丢包信号后，通过计算 RTT（Round-Trip Time）的变化情况来选择不同的降速模式，从而达到高吞吐、低时延的性能指标。此外，Cubic＋能够过滤网络中的随机丢包情况，避免不必要的降速。

3）针对跨数据中心网络场景，本文在 NS3 网络仿真平台上进行了实验，使用真实的拓扑和流量模型验证了 Cubic＋的性能。实验结果表明，Cubic＋与现有流行算法相比，平均流完成时间最多减少了 20.77%，第 99 百分位流完成时间最多减少了 15.88%。

本文第 2 章讨论了跨数据中心网络下的异构缓冲设计对于当前主流拥塞控制算法的影响和本文设计的动机；第 3 章详细介绍了 Cubic＋算法的设计，并说明了设计足够简单且能够集成到 Linux 内核中；第 4 章展示了 Cubic＋的性能验证结果；最后总结全文并展望未来。

2 背景和动机

2.1 缓冲区容量的不确定性

缓冲区大小作为网络中的一个关键问题，已经被学术界和工业界讨论了数十年。尽管存在大量的研究成果[16-18]，但至今尚未找到一种完美的解决方案来指导用户在复杂的广域网环境中正确设置和部署网络设备的缓冲区大小。设备制造商和网络管理人员倾向于生产和配置尽可能大的缓冲区。普遍观点认为，大缓冲会增加排队时延，而小缓冲容易导致丢包，从而降低吞吐量。此外，缓冲区的合理大小还受到其他因素的影响，如拥塞控制算法、实际流量模型、部署成本等。

因此，在跨数据中心网络实际部署中，链路中的缓冲区往往表现出异构性，即同一链路上不同设备的缓冲容量差异显著。例如路由器/交换机每端口容量从 100 kB 到 10 MB 不等。近几年，随着短视频、云计算等服务的快速发展，以及处于不同地区的数据中心之间的频繁通信，广域网的流量和带宽都在以惊人的速度增长。这使得传统的网络设备和架构难以满足日益增长的网络通信需求。大型企业如微软、谷歌、阿里巴巴已经在广域网中部署了定制化的浅缓冲设备（交换机每端口缓冲不到 200 kB），以提供更高效的性能[19]。

尽管浅缓冲设计被公认能够减少排队时延，但广域网中仍有许多主机、交换机和路由器配备了大缓冲区。例如，一些边界路由器为保证连接不同地区的数据中心时能够提供高吞吐和可靠的传输服务，每端口的缓冲区大小从 100 MB 到 700 MB 不等[20]。具体而言，在跨数据中心网络场景中，广域网通过深缓冲路由器连接两个位于不同地理位置的数据中心来实现高吞吐传输。

传统观点认为，在实现链路最大带宽利用率时，拥塞控制算法对最小缓冲有一定要求。例如，Cubic 需要 3/7 BDP（时延带宽积）大小的缓冲区，BBR 则要求 1/4 BDP。然而，广域网中缓冲区大小的异构特性使得 BDP 值可能远小于或者远大于瓶颈交换机的缓冲容量。根据本文在云计算平台上的测量，广域网的 RTT 从几毫秒到几百毫秒不等，远高于数据中心网络的微秒级 RTT。这种差异使得广域网中的 BDP 可能高于浅缓冲区的。例如，1 Gbps 的链路，在 RTT 为 10ms 时，其 BDP 的值为 1.25 MB。而 BCM Trident＋交换机芯片每端

口的缓冲区为 192 kB,1.25 MB 的 BDP 大约是 192 kB 的缓冲区容量的 6 倍,显然无法满足拥塞控制算法对于最小缓冲的要求,导致无法实现链路的满带宽利用率。相关学者以及谷歌都曾发表关于在浅缓冲交换机中发生大量丢包的经验[21]。然而,在深缓冲交换机中的情况则相反,1.25 MB 的 BDP 小于上百 MB 的深缓冲区容量。瓶颈交换机缓冲大小的不确定性,严重影响了不同拥塞控制算法在共享瓶颈链路时的性能,造成吞吐下降和排队时延增加。

2.2　Cubic 存在不必要的吞吐损失

Cubic 拥塞控制算法是 Linux,macOS 和 Android 系统的默认拥塞控制算法。它遵循 AIMD(线性增,乘法减)窗口的原则。Cubic 通过式(1)线性增加窗口:

$$cwnd(t)=C*(t-k)^3+W_{\max} \tag{1}$$

其中,t 是上次减小窗口所花费的时间,$W_{\max}$ 是窗口减小之前的窗口大小。K 的取值由式(2)所决定:

$$K=\sqrt[3]{W_{\max}*(1-\beta)/C} \tag{2}$$

其中,C 和 β 是 Cubic 的参数。Cubic 不会在缓冲溢出前减少窗口大小,只有在发生丢包时才会根据 β 系数(默认值为0.2)乘法缩减窗口。因此,Cubic 具有填满缓冲的特性。传统观点认为,Cubic 至少需要占用 3/7 BDP 大小的缓冲才能达到满链路带宽。

单一的丢包降速机制:在跨数据中心网络中,如果深缓冲路由器出现丢包,此时大量数据包会堆积在缓冲区队列中,即使数据中心内部的流量能够迅速反应并缓解拥塞,但由于缓冲区过大,队列排空仍需要一定时间,这会显著增加数据包的排队时延。当数据中心内部的浅缓冲区发生丢包时,广域网的拥塞反馈环路较长,这也给数据中心内部的拥塞控制带来较大压力。在这两种不同缓冲区容量的队列中发生丢包时,所造成的性能影响是极为显著的。而 Cubic 对于所有丢包事件都采用相同的反馈机制,这使得它难以精准调节发送速率,从而无法有效维持高吞吐。

吞吐不公平性:学术界和工业界已有理论分析和实验结果表明,Cubic 与 BBRv1 和 BBRv2 之间的吞吐公平性与瓶颈链路的缓冲区大小以及时延带宽积(BDP)的关系密切相关[22]。具体而言,当瓶颈链路为浅缓冲区时,BBR 的吞吐量明显高于 Cubic;而当瓶颈链路为深缓冲区时,BBR 与 Cubic 在竞争瓶颈带宽时的表现较差。本文通过 NS3 仿真实验进一步验证了这一结论。实验采用的网络拓扑如图 1 所示。

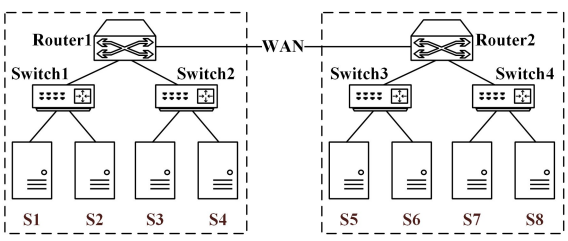

图 1　跨数据中心网络拓扑

Fig. 1　Cross-datacenter network topology

在实验中,发送端 S1 和 S3 同时发送一条 Cubic 和 BBR 流到同一个接收端,拥塞发生在 Router1 处。所有流量共享 100 Mbps 的瓶颈链路,RTT 设置为 30 ms,一个 BDP 值为 375 kB。Cubic 和 BBR 流同时从 0 秒开始并持续 5 秒。实验设置了不同大小的缓冲区,包括 0.1 BDP,0.3 BDP,0.5 BDP,0.7 BDP,1 BDP,3 BDP,5 BDP,7 BDP,10 BDP 和 20 BDP。Cubic 与 BBRv1 和 BBRv2 共享瓶颈链路的实验结果分析如下。

1)如图 2 所示,当拥塞交换机为浅缓冲区时,Cubic 会被 BBRv1 抢占带宽,导致吞吐显著小于 BBRv1。BBRv1 会引起 Cubic 持续大量丢包,导致 Cubic 吞吐量急剧下降。当拥塞交换机是深缓冲区时,BBRv1 吞吐远低于 Cubic,这是因为 BBRv1 会限制其飞行字节数大约在 2 * BDP,而 Cubic 会不断填充深缓冲区,直至丢包。

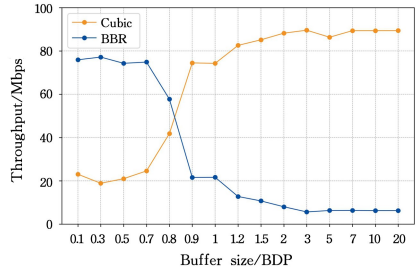

图 2　Cubic 与 BBR 共存时吞吐变化

Fig. 2　Throughput in Cubic and BBR coexistence

2)如图 3 所示,当拥塞交换机为浅缓冲区时,BBRv2 改善了与 Cubic 共存时的吞吐公平性,其中的关键改动是 BBRv2 通过设置 inflight_hi 和 inflight_lo 两个参数来限制网络中的飞行数,并增加对丢包信号的响应机制。这能够减少缓冲溢出导致的丢包,使得 Cubic 顺利地增加窗口大小。但是 BBRv2 与 BBR 相似,在深缓冲区仍然无法与 Cubic 公平地共享带宽。

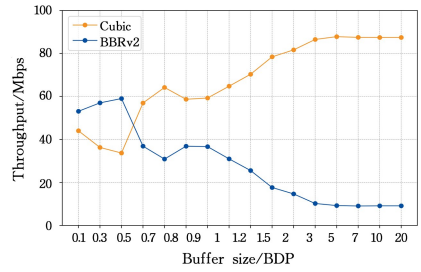

图 3　Cubic 与 BBRv2 共存时吞吐变化

Fig. 3　Throughput in Cubic and BBRv2 coexistence

3)对于随机丢包场景测试,本文设定随机丢包率从 0.0001%到 10%。当丢包率超过 0.001%时,Cubic 吞吐量明显下降,而 BBR 和 BBRv2 则能很好地容忍随机丢包事件。

2.3　Cubic+:跨数据中心网络中的 Cubic 版本

在 TCP 演进历程中,Cubic 曾完全取代 New Reno,成为 Linux 内核的默认拥塞控制算法。类似地,谷歌主推的 BBR 算法是否也会取代 Cubic?

文献[23]从理论建模角度给出结论,BBR 不会完全取代 Cubic,理由是因特网中 BBR 和 Cubic 流量的分布会达到纳什均衡。即便谷歌后续改进 BBR 算法,推出了 BBRv2,但其未被集成至 Linux 内核,用户在默认内核中无法使用 BBRv2。此外,作为谷歌推出的商业拥塞控制算法,BBR 能否在其他

应用场景大规模使用仍然是未知数。例如，Netflix 在尝试使用 BBRv2 仅 3 个月后便转向了其他拥塞控制算法。

可以明确的是，Cubic 和 BBR 作为当前最主流的两种拥塞控制算法，将会不可避免的共存于广域网中。2019 年的测量研究指出，因特网中 30%的网站采用的是 Cubic 算法，是使用率最高的拥塞控制算法，BBR 位居第二。此外，超过 40%的因特网下载流量采用了 BBR 算法。然而，当这两种算法共享瓶颈链路时，BBR 算法容易过度抢占带宽，导致 Cubic 算法吞吐量低。更为关键的是，针对 Cubic 与 BBR 算法共存时出现的吞吐不公平问题，谷歌已经对 BBR 算法进行了修复，但目前尚未有相关研究通过改进 Cubic 算法来解决这一问题。

因此，迫切需要通过简单地升级原有 Cubic 算法来解决跨数据中心网络场景下，Cubic 造成的数据包堆积、排队时延增加和吞吐下降的问题。受到针对主机端的拥塞控制算法[24]的启示，本文提出可以利用时延拥塞信号辅助 Cubic 在跨数据中心网络场景进行拥塞控制。与使用 ECN 作为拥塞信号的算法不同，基于时延的拥塞控制算法不需要额外修改交换机。其核心思想是，根据 RTT 的变化来限制广域网中端到端的飞行字节数，从而实现高吞吐传输。然而，由于时延信号在长距离传输过程中可能发生失真，粗粒度的 RTT 信号容易导致数据发送方错误判断网络链路的状态，进而使数据包填满浅缓冲区，导致数据包丢失和吞吐急剧下降。

3 设计

Cubic+的设计目标是通过升级原有 Cubic 算法，使其能够在跨数据中心网络中与其他拥塞控制算法共存，在保证算法公平性的前提下，实现高吞吐和低排队时延的传输性能。在设计 Cubic+拥塞控制算法过程中，面临以下两个主要挑战。

1)如何减少对现有协议栈的侵入性。与 BBR 等复杂算法不同，Cubic+ 需要在设计上尽量保持简洁高效，以便最小化对 Cubic 原有机制的修改。这也有利于后续将 Cubic+ 集成到 Linux 内核协议栈中。因此，Cubic+ 仅使用当前网络系统中常见的机制(如时延信息、丢包信号、ECN 标志)，并未引入新的拥塞窗口调整机制，而是在原有 Cubic 的“加法增窗口，乘法减窗口”基础上，增加了多种条件匹配机制。

2)如何在不同场景下的丢包事件中选择合适的发送速率。由于 Cubic 是基于丢包的拥塞控制算法，其会尝试填满缓冲区直至触发丢包。当瓶颈链路为深缓冲区时，可能导致排队时延急剧增加。为解决这一问题，Cubic+周期性地计算链路中的时延信号，当队列未过度堆积时，采用适度的乘法减窗口操作。当瓶颈链路为浅缓冲区时，其他拥塞控制算法(如 BBR)可能导致 Cubic 出现持续丢包，迫使其频繁进行乘法减窗口操作。图 2 表明，与 BBR 在浅缓冲区共存时，Cubic 频繁丢包导致带宽利用率几乎降为 0。为解决这一问题，Cubic+引入了周期性乘法减小窗口的机制，即使在浅缓冲区下频繁丢包，也能维持足够的带宽竞争能力。

此外，广域网链路中的非拥塞因素也会导致数据包丢失。如链路损坏丢包或随机丢包检测等。基于丢包的算法通常会对所有丢包事件采取降窗口操作。Cubic+ 通过整合丢包信号和链路时延变化情况，利用降噪函数过滤随机丢包信号，以更准确地判断丢包是否由拥塞引起。

3.1 Cubic+架构

Cubic+是一个基于时延和丢包的拥塞控制算法。其核心思想是，根据不同拥塞场景中丢包信号的实际含义，采取差异化的响应措施，而非对所有丢包事件统一处理。Cubic 在保留传统 Cubic 响应丢包和 ECN 信号机制的同时，增加了对 RTT 变化的监测，以推测瓶颈链路缓冲区的占用情况，从而实现更加精准的窗口调整。Cubic+的架构如图 4 所示。

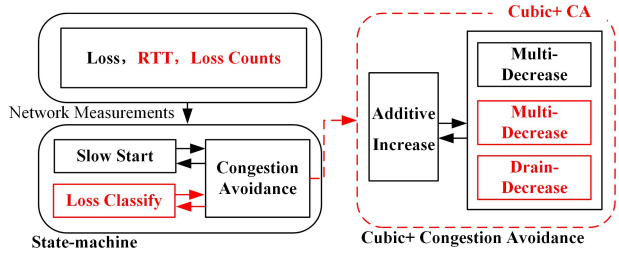

图 4 Cubic+算法架构

Fig. 4 Architecture of Cubic+ algorithm

在传统 Cubic 算法的基础上，Cubic+ 新增了两种发送模式，从而在不同的场景下对 Cubic 降速机制做更细粒度的控制。具体而言，Cubic+ 会根据链路中 RTT 和丢包数量的变化趋势提取链路状态信息，并通过“Loss Classify”状态分析链路状况，以决定采用何种窗口调整策略。在默认情况下，Cubic+ 保持传统 Cubic 的拥塞控制机制，仅当时延和丢包信号的变化超过一定阈值时，才会主动调整降速模型。

3.2 周期性排空浅缓冲区队列机制

Cubic+在收到丢包信号后，可根据链路 RTT 的变化幅度轻松判断丢包是否发生在浅缓冲交换机中。这是因为，当丢包发生在浅缓冲交换机时，排队时延的增长通常非常有限。尤其是在广域网的高传播时延链路中，RTT 的变化通常不会过于剧烈。因此，本文用 $\Delta RTT < T_{\_low}$ 来表示丢包发生在浅缓冲交换机中。算法 1 中的 ΔRTT 代表链路中由于排队而增加的时延，ΔRTT 定义中的 $RTT_{\_min}$ 代表在最近 0.5 个窗口内观察到的最小 RTT。选择 $RTT_{\_min}$ 而非平均或最大 RTT，是因为它能更准确地检测长时间队列堆积的情况，同时容忍短暂突发引起的临时排队[3]。此外，$RTT_{\_min}$ 能实现排队时延与网络中其他时延变化的解耦，最大化排除噪声对排队时延计算的影响。α 为 EWMA 权重参数。$T_{\_low}$ 是时延阈值，用来表示没有数据包排队转发的理想环境。*Drops* 表示当前 RTT 内丢包的数量。Periodic-Decrease 模式是 Cubic+引入的一种周期性降速机制，即在每发生 3 次丢包事件后，触发 1 次 Cubic 的乘法减窗口操作，计算式为：

$$CWND = CWND * beta \tag{3}$$

其中，*beta* 是 Cubic 乘法减参数。Periodic-Decrease 模式的核心在于，在保持 Cubic 响应丢包行为不变的前提下，降低其响应频率，从而缓解浅缓冲交换机丢包引起的吞吐量过度下降问题，并确保 Cubic 在与 BBR 共存时能争夺到更多带宽资源。

Cubic+算法具体如算法 1 所示。

算法 1 Improve Cubic throughput

Data: NewComingACK, RTTmin, RTTbase

Result: New Congestion WindowSize

```
1. ΔRTT_new=new_rtt−RTT_min;
2. ΔRTT=(1−α)·ΔRTT+α·ΔRTT_new;
3. LOSS_signal:loss indicated by Cubic;
   ECN_signal:ecn indicated by Cubic;
4. if LOSS_signal then
5.     Execute operation;
6.     if Δ RTT>T_high then
7.         CWND=CWND * newbeta;
8.     else if ΔRTT<T_low and Loss < L then
9.         CWND * = max(β,1−ΔRTT RTT_base);
10.    else
11.        Cubic
12. else
13.    if !ECN_signal and ΔRTT>Thigh then
14.        CWND * =β·max(δ,1−ΔRTT/RTT_base);
15.    else
16.        Cubic
```

3.3　快速排空深缓冲区队列机制

对于标准 TCP 算法 Cubic,在深缓冲链路中,通过基于固定参数的乘法减窗口操作可以较容易地实现高吞吐量[7]。然而,Cubic 倾向于尝试填满缓冲区,数据包可能在队列中过度堆积,导致排队时延显著升高。为解决这一问题,Cubic+在传统 Cubic 的拥塞避免状态中新增了一种子状态:Drain-Decrease。该子状态通过执行一次减窗口操作来快速排空缓冲队列,从而降低排队时延。具体而言,Cubic+从连接开始便周期性地监测网络时延变化。当时延超过设定阈值后,Cubic+进入 Drain-Decrease 状态。在此状态下,每当新的 ACK 数据包到达时,Cubic+会实时更新当前链路的 RTT 值并计算 ΔRTT,以此判断瓶颈链路的队列堆积情况。若在周期 T 内 $\Delta RTT > T_{\_high}$,则认为队列过度堆积,并触发一次乘法减窗口操作,窗口减小公式与 Cubic 处理丢包信号时相同。Drain-Decrease 机制有效限制了端到端飞行字节数,避免瓶颈交换机缓冲区溢出,从而显著降低排队时延。

然而,周期性地排空队列机制可能存在过降速问题。例如,其他交叉流量引起的缓冲堆积可能被误判为 Cubic+自身导致的队列堆积,进而触发不必要的降速操作。为此,本文为 Drain-Decrease 状态增加了一项约束:在连续两次排空队列操作后,若 ΔRTT 仍然高于阈值,则 Cubic+认为队列堆积是由其他流量造成的,下一次排空操作时将不减小拥塞窗口。此优化确保了 Cubic+在复杂网络环境中不会被其他流量过度抢占带宽。

3.4　非拥塞丢包信号过滤机制

为了过滤广域网中的非拥塞丢包信号,Cubic+整合丢包信号和 RTT 的变化率来判断是否发生非拥塞丢包。在广域网中发生丢包的原因有很多,比如链路损坏丢包、随机丢包检测、拥塞丢包,等等。物理层或随机丢包导致的丢包事件有一个显著特征:它们不会引起 RTT 升高。因此,Cubic+利用这一特性过滤非拥塞丢包信号,提升吞吐。在具体实现中,Cubic+每隔一个 RTT 时间计算最新丢包率和加权丢包率,以区分丢包来源。最新丢包率的计算方式为:丢包总数/RTT。其中,丢包总数为当前 RTT 内累计的丢包数量。在进入下一个 RTT 周期时,Cubic+计算上一周期的最新丢包率,并通过指数加权移动平均(EWMA)计算加权丢包率,如式(4)所示:

$$Loss=(1-\alpha)*Loss+\alpha*Loss\_new \tag{4}$$

其中,α 为平滑因子,用于控制历史丢包率和当前丢包率的权重分配。随机丢包事件发生时,经过 EWMA 计算,加权丢包率会接近于 0。本文设置随机丢包的判断阈值为 L,当收到丢包信号后,若 Cubic+计算出当前 $\Delta RTT < T_{\_high}$,同时满足加权丢包率小于 L 的条件,Cubic+判断丢包并非由网络拥塞引起,因此保持当前拥塞窗口大小不变。

4　实验评估

本文基于 NS3 仿真平台[25]和实际网络场景对 Cubic+算法的性能进行了评估。4.1 节详细介绍了实验设置;4.2 节通过 NS3 仿真实验,分析了两条流共存场景下 Cubic+ 与 BBR 和 BBRv2 共存时的性能指标;接着,评估了 Cubic+ 在网络随机丢包条件下的容忍能力;最后,通过与主流算法的流完成时间进行对比,进一步验证了 Cubic+ 的性能优势。在实际网络场景下的实验说明,NS3 仿真实验的结论是可靠的。

4.1　实验设置

为了模拟不同网络环境下影响吞吐量变化的真实因素,本文在多种流量模型、拥塞控制算法及缓冲大小条件下进行了实验,以评估 Cubic+ 在不同网络条件下的性能。

1)比较算法

为了评估 Cubic+ 与其他算法的兼容性,本文对比了 Cubic+与 BBR 和 BBRv2 在瓶颈链路共存场景下的性能表现,其中 BBR 和 BBRv2 的参数均采用谷歌提供的默认设置。此外,还在真实流量场景下比较了 Cubic+,BBR,BBRv2,Copa 和 Cubic 的性能。本文选用吞吐量和流完成时间作为主要性能指标,以直观反映各算法的传输效率。

2)实验参数

本文选择 NS3 作为仿真平台,采用如图 1 所示的仿真网络拓扑,总共设置 6 个发送方、2 个接收方。不同数据中心服务器之间的传播时延为 15.3 ms(RTT 为 30.6 ms),交换机端口速率设置为 100 Mbps。RTO 设为 50 ms,并开启选择性重传机制。

4.2　两条流共存时的性能评估

本文首先评估一条 Cubic+ 流与一条 BBR 流共存时的性能表现。相比多条流共存的场景,单流评估能够避免同种算法间的交互影响(如多条 BBR 流之间的影响),从而更真实地反映 Cubic+ 与 BBR 的交互过程。实验设置为一条 Cubic+流与一条 BBR 流分别从两个不同的发送端向同一个接收端发送数据,总持续时间约为 40 s,初始窗口数设为 10。瓶颈链路位于交换机 0,缓冲大小从 0.1 BDP 逐步增加至 20 BDP。

本文对重复 5 次实验的共存平均吞吐值进行对比分析。图 5 表明,在浅缓冲(0.1 个 BDP)条件下,Cubic+ 相比 Cubic 提升了 48.8% 的吞吐量;在深缓冲(20 BDP)条件下,Cubic+ 吞吐提升 16.5%,使 BBR 的带宽分配更加公平。图 6 显示,Cubic+ 显著改善了 BBRv2 在深缓冲场景下被 Cubic 抢占吞吐的不公平现象。当瓶颈链路缓冲为 20 BDP 时,Cubic+ 相比 Cubic 的吞吐量降低了 21.2%。这些结果验证了

Cubic+ 提升吞吐的有效性。

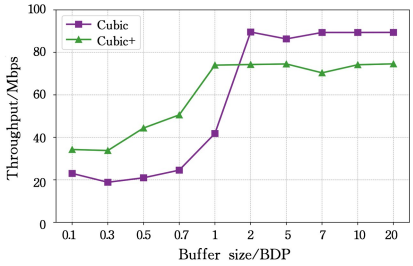

图 5 与 BBR 共存时平均吞吐

Fig. 5 Avg throughput in BBR coexistence

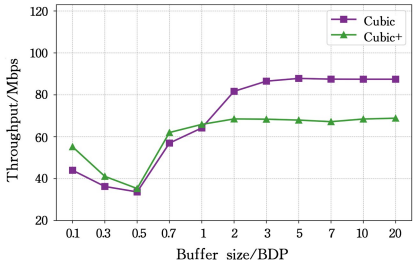

图 6 与 BBRv2 共存时平均吞吐

Fig. 6 Average throughput in BBRv2 coexistence

4.3 缓冲区队列长度对比

从图 7 中可以看到，Cubic+ 在与 BBR 共存瓶颈链路时，深缓冲的队列堆积程度明显小于 Cubic。当缓冲为 20 BDP 时，Cubic+ 的队列长度减少了 23%。同样，在与 BBRv2 共存瓶颈链路的场景下，如图 8 所示，Cubic+ 的队列长度比 Cubic 降低了 33%。仿真结果表明，Cubic+ 能有效缓解深缓冲中的队列堆积问题。

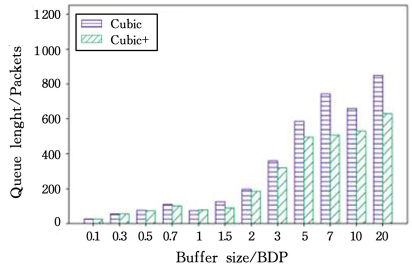

图 7 与 BBR 共存时平均队列长度

Fig. 7 Average queue length in BBR coexistence

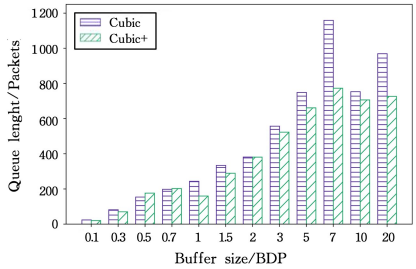

图 8 与 BBRv2 共存时平均队列长度

Fig. 8 Average queue length in BBRv2 coexistence

4.4 重传率

本文进一步评估 Cubic+算法在浅缓冲中是否会引起大量数据包丢失和重传(影响算法的健壮性)。图 9 表明，Cubic+ 与 BBR 共存时的数据包重传率与 Cubic 基本相当。类似地，图 10 显示 Cubic+ 与 BBRv2 共存时也未引入额外的重传开销。这是因为 Cubic+在获取带宽后能够使 BBR 的吞吐回归公平点，从而避免在瓶颈缓冲区中出现更激进的丢包现象。

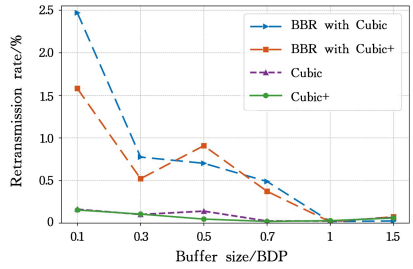

图 9 与 BBR 共存时重传率

Fig. 9 Retransmission rate in BBR coexistence

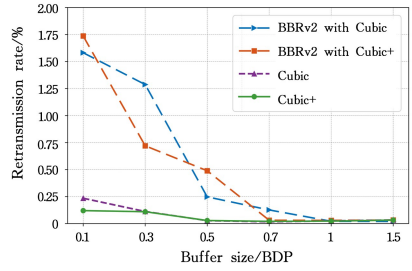

图 10 与 BBRv2 共存时重传率

Fig. 10 Retransmission rate in BBRv2 coexistence

4.5 随机丢包容忍性

Cubic 对于随机丢包非常敏感，劣于 BBRv1 和 BBRv2。这是因为 Cubic 无法区分丢包原因(如拥塞或随机丢包)，对所有丢包事件都采用相同的窗口缩减策略。本次实验拓扑和参数的设置与 4.1 节中的保持一致。为了排除拥塞丢包对实验结果的干扰，将瓶颈链路缓冲设置为 32 个 BDP。随机丢包率从 0.0001% 逐步增加至 10%。图 11 显示，当随机丢包率分别为 0.01%，0.1% 和 1% 时，Cubic+ 的吞吐相较于 Cubic 分别提升了 0.26 倍、6 倍和 32 倍。

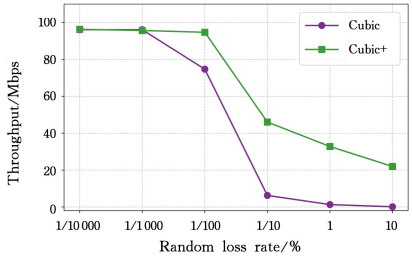

图 11 不同随机丢包率时的平均吞吐

Fig. 11 Average throughput at different packet loss rate

4.6 在跨数据中心网络流量下的性能表现

为验证 Cubic+ 在跨数据中心网络流量场景的性能表现，本文基于 NS3 仿真，选用文献[26]采集的真实网络流量数据集作为实验流量。该数据集的显著特点是大流占比高，即单流大小(Flow Size)大于 10MB 的流占比超过了 80%。拓扑与 4.1 节的实验设置保持一致，共有 4 个发送方，随机选择一个服务器作为接收方，流到达时间符合泊松分布。本文选择当前常见的广域网控制算法作为比较对象，包括 Copa，

Cubic,BBRv1,BBRv2。实验分别设置两种缓冲大小:浅缓冲(0.1BDP)和深缓冲(10BDP)。本文将平均 FCT(流完成时间)和 99th FCT(第 99 百分位流完成时间)作为主要的性能比较指标。

从图 12 和图 13 可以看出,Cubic+相较于其他算法表现出明显的优势。在浅缓冲场景下,Cubic+相比于 BBR 和 Cubic,平均 FCT 分别降低了 10.73%和 14.09%,99th FCT 分别降低了 5.75%和 10.05%。在深缓冲场景下,尽管 Cubic 表现最差,Cubic+也能将平均 FCT 和 99th FCT 分别降低 9.41%和 0.98%。这主要得益于 Cubic+ 在浅缓冲场景下通过周期性降速保持较高的带宽,而在深缓冲场景下则能够根据 RTT 的增长情况动态排空队列,从而有效避免高排队时延的产生。

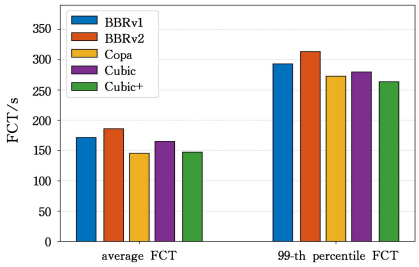

图 12　浅缓冲场景流完成时间对比

Fig. 12　Comparison of FCT in shallow buffer

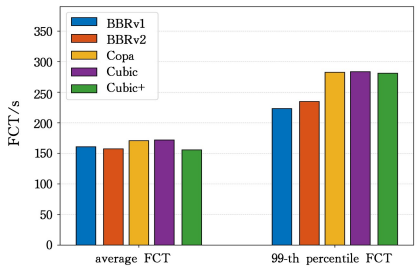

图 13　深缓冲场景流完成时间对比

Fig. 13　Comparison of FCT in deep buffer

4.7　NS3 实验结论的可靠性

NS3 作为数据中心网络领域内许多研究者使用的仿真软件,可用于验证网络协议和设备的性能。NS3 网络协议栈与 Linux 内核 TCP 协议栈的一致性设计,确保了仿真结果的高精度。但在硬件、操作系统模拟等方面与实际网络场景仍然存在差异。本文分别在实际网络场景下和 NS3 仿真软件中进行对照实验,来确保 NS3 仿真实验结论的可靠性。实验采用图 1 简化后的拓扑,设置两个发送方和一个接收方,链路速率设置为 1 Gbps。RTT 设置为 10.2 ms,服务器通过 Linux 工具 TC 在网卡上模拟广域网链路的传播时延。S1 以 1 Gbps 向 S6 发送长稳流,共持续 1 s。在第 0.1 s,选择 S5 或 S7 向 S6 同时全速发送 50 条流,持续 0.05 s,从而造成浅或深缓冲区的拥塞。

由图 14 可知,当浅缓冲交换机(0.2 BDP)拥塞时,实际网络场景和 NS3 仿真软件的平均吞吐差异小于 3%;当深缓冲交换机(2 BDP)拥塞时,二者之间的平均吞吐差异小于 2%。这表明不同实验平台下的平均吞吐表现基本一致,不会影响现有的结论,进一步证明了 Cubic+ 在 NS3 仿真环境中的实验结论是可靠的。

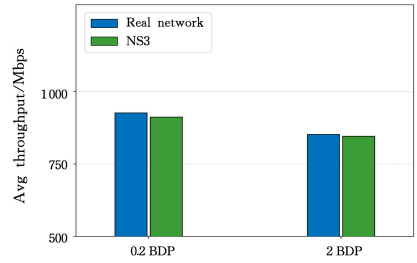

图 14　不同实验平台的平均吞吐对比

Fig. 14　Average throughput comparison across platforms

结束语　随着云计算和跨数据中心网络的发展,对网络传输性能的要求日益提高。然而,当前主流的拥塞控制算法在面对跨数据中心网络场景时,仍然存在显著的性能局限。本文针对 Linux 内核的默认算法 Cubic,分析了其在深浅缓冲异构场景中的性能瓶颈,并提出了一种简单高效的改进算法 Cubic+。通过融合丢包信号、ECN 信号和时延信号,Cubic+能够更精准地调整拥塞窗口,解决传统 Cubic 算法在浅缓冲场景中频繁降速和深缓冲场景中排队时延过高的问题。此外,Cubic+在应对随机丢包时表现出更强的鲁棒性,避免了不必要的吞吐损失。

在未来的研究中,可以考虑采用更加细粒度的 INT 信号来优化拥塞控制。同时,在跨数据中心网络场景下,拥塞控制算法应充分考虑数据中心内部流量与跨数据中心流量之间的相互影响,以及 TCP 流量与 RDMA 流量之间的相互作用,这些因素都会影响拥塞控制算法的性能。

参 考 文 献

[1] JAIN S,KUMAR A,MANDAL S,et al. B4:Experience with a globally-deployed software defined WAN[J]. ACM SIGCOMM Computer Communication Review,2013,43(4):3-14.

[2] HONG C Y,MANDAL S,AL-FARES M,et al. B4 and after:managing hierarchy,partitioning,and asymmetry for availability and scale in google's software-defined WAN[C]// Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication. 2018:74-87.

[3] ZENG G,BAI W,CHEN G,et al. Congestion control for cross-datacenter networks[J]. IEEE/ACM Transactions on Networking,2022,30(5):2074-2089.

[4] SPANG B,ARSLAN S,MCKEOWN N. Upda-ting the theory of buffer sizing[J]. ACM SIGMETR-ICS Performance Evaluation Review,2022,49(3):55-56.

[5] APPENZELLER G,KESLASSY I,MCKEOWN N. Sizing router buffers[J]. ACM SIGCOMM Computer Communication Review,2004,34(4):281-292.

[6] MCKEOWN N,APPENZELLER G,KESLASSY I. Sizing router buffers(redux)[J]. ACM SIGCOMM Computer Communication Review,2019,49(5):69-74.

[7] GETTYS J,NICHOLS K. Bufferbloat:dark buffers in the internet[J]. Communications of the ACM,2012,55(1):57-65.

[8] ARISTA NETWORKS. Arista Networks Products [EB/OL]. (2024-11-21) [2025-02-17]. https://www.arista.com/en/products.

[9] RHEE I, XU L, HA S, et al. CUBIC for fast long-distance networks: RFC9438[R]. 2018.

[10] MISHRA A, SUN X, JAIN A, et al. The great Internet TCP congestion control census[C]// Abstracts of the 2020 SIGMETRICS/Performance Joint International Conference on Measurement and Modeling of Computer Systems. 2020: 59-60.

[11] CARDWELL N, CHENG Y, GUNN C S, et al. BBR: congestion-based congestion control[J]. Communications of the ACM, 2017, 60(2): 58-66.

[12] WARE R, MUKERJEE M K, SESHAN S, et al. Modeling BBR's interactions with loss-based congestion control[C]// Proceedings of the Internet Measurement Conference. 2019: 137-143.

[13] YANG F, WU Q, LI Z, et al. BBRv2+: Towards balancing aggressiveness and fairness with delay-based bandwidth probing [J]. Computer Networks, 2022, 206: 108789.

[14] SAEED A, GUPTA V, GOYAL P, et al. Annulus: A dual congestion control loop for datacenter and WAN traffic aggregates [C]// 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: 735-749.

[15] ZOU S, HUANG J, LIU J, et al. Gtcp: Hybrid congestion control for cross-datacenter networks[C]// 2021 IEEE 41st International Conference on Distributed Computing Systems (ICDCS). IEEE, 2021: 932-942.

[16] SPANG B, WALSH B, HUANG T Y, et al. Buffer sizing and video QoE measurements at Netflix[C]// Proceedings of the 2019 Workshop on Buffer Sizing. 2019: 1-7.

[17] ADDANKI V, APOSTOLAKI M, GHOBADI M, et al. ABM: Active buffer management in datacenters[C]// Proceedings of the ACM SIGCOMM 2022 Conference. 2022: 36-52.

[18] ADDANKI V, BAI W, SCHMID S, et al. Reverie: Low pass filter-based switch buffer sharing for datacenters with RDMA and TCP traffic[C]// 21st USENIX Symposium on Networked Systems Design and Implementation (NSDI 24). 2024: 651-668.

[19] ZENG G, QIU J, YUAN Y, et al. FlashPass: Proactive congestion control for shallow-buffered WAN[C]// 2021 IEEE 29th International Conference on Network Protocols (ICNP). IEEE, 2021: 1-12.

[20] WARNER J. Packet Buffers [EB/OL]. [2025-02-17]. https://people.ucsc.edu/~warner/buffer.html.

[21] SINGH A, ONG J, AGARWAL A, et al. Jupiter rising: A decade of clos topologies and centralized control in Google's datacenter network[J]. ACM SIGCOMM Computer Communication Review, 2015, 45(4): 183-197.

[22] YUAN H, HAN Y, ZHONG Y, et al. FEBBR: A Fairness-Enhanced Approach for BBR Congestion Control[C]// 2023 IEEE International Symposium on Broadb and Multimedia Systems and Broadcasting (BMSB). IEEE, 2023: 1-6.

[23] MISHRA A, TIU W H, LEONG B. Are we heading towards a BBR-dominant Internet? [C]// Proceedings of the 22nd ACM Internet Measurement Conference. 2022: 538-550.

[24] AGARWAL S, KRISHNAMURTHY A, AGARWAL R. Host congestion control[C]// Proceedings of the ACM SIGCOMM 2023 Conference. 2023: 275-287.

[25] NS 3. ns-3 Homepage [EB/OL]. [2025-02-17]. https://www.nsnam.org/.

[26] ALIZADEH M, GREENBERG A, MALTZ D A, et al. Data center TCP (DCTCP)[C]// Proceedings of the ACM SIGCOMM 2010 Conference. 2010: 63-74.

LONG Tie, born in 1999, postgraduate. His main research interests include computer networks and congestion control.

XIAO Fu, born in 1980, Ph. D, professor, Ph.D supervisor. His main research interests include computer networks and Internet of Things.

(责任编辑:柯颖)