数据中心网络中一种基于ECN的TCP慢启动拥塞控制策略

黄家玮[1]，徐文茜[1]，胡晋彬[1]，王建新[1]，叶 进[2]

(1. 中南大学信息科学与工程学院 长沙 410083；2. 广西大学计算机与电子信息学院 南宁 530004)

【摘要】在数据中心网络中，高带宽、低时延的链路和多对一的网络结构造成了TCP Incast吞吐率崩溃问题。现有的改进方法都关注于在TCP拥塞避免阶段改进其拥塞控制算法，而忽视了慢启动阶段中激进的指数增窗方式是造成Incast问题的关键原因。因此，该文提出了一种基于ECN的TCP慢启动拥塞控制策略(gentle slow sTart, GST)，利用已有的拥塞标志位动态反馈缓存拥塞状况，以调节慢启动阶段的增窗速率。实验结果表明，该方法帮助现有的数据中心TCP协议有效地避免了并发传输中的吞吐率崩溃现象，将并发度和吞吐率分别提升了3.4倍和85倍。

关 键 词 数据中心网络； 显式拥塞反馈； 慢启动； 传输控制协议

中图分类号 TN97 **文献标志码** A doi:10.3969/j.issn.1001-0548.2018.02.002

An ECN-based Slow-Start of TCP Congestion Control in Data Center Networks

HUANG Jia-wei[1], XU Wen-xi[1], HU Jin-bin[1], WANG Jian-xin[1], and YE Jin[2]

(1. School of Information Science and Engineering, Central South University Changsha 410083;
2. School of Computer, Electronics and Information, Guangxi University Nanning 530004)

Abstract TCP incast congestion happens in data center networks with high-bandwidth and low-latency, when multiple synchronized servers send data to a single receiver in parallel. The existing improved methods focus on the congestion control algorithm in the congestion avoidance phase, ignoring the aggressive exponential increasing window in the slow start phase that is a key reason for incast problem. Therefore, this paper proposes an ECN-based slow-start of transmission control protocol (TCP) congestion control strategy, which dynamically feedbacks congestion status by using the existing congestion flags to adjust the window-increasing speed during the slow start phase. Experimental results show that our approach helps the existing data center TCP effectively avoid the throughput collapse in concurrent transmissions. The number of concurrent flow and network throughput are increased by 3.4× and 85×, respectively.

Key words data center network; ECN; slow start; transmission control protocol

近年来，随着在线搜索、社交网络、电子商务等网络应用的飞速发展和普及，越来越多的在线应用系统被迁移到数据中心中，利用大规模的计算和存储资源为用户提供各种网络服务[1]。为了保证应用请求的快速响应，数据中心网络(data center network, DCN)采用了高并发的网络传输[2]。这样的同步并发流极易造成某些路径瞬时成为瓶颈，引起网络拥塞，导致TCP连接出现频繁丢包，最终出现TCP重传超时(retransmission time-out, RTO)。一旦TCP超时发生，新一轮的请求必须要等上一轮中经历超时的连接完成超时重传才会发出。此时，即使是一条TCP流超时带来的空等时间(一般在200 ms以上)都会造成传输链路空闲，导致TCP吞吐率严重崩溃[3]。而且数据中心网络具有超高带宽[4]、超低时延的特性，这与传统广域网有很大的差别，所以传统TCP在数据中心网络中的应用效果很差。

为了解决吞吐率崩溃问题，国内外学者做出了很多研究。其中基于延时的协议TIMELY[5]、DX[6]和ICTCP[7]等都通过RTT的测量来检测拥塞以调节拥塞窗口。基于显式拥塞反馈(explicit congestion notification, ECN)的协议DCTCP[8]、D^2TCP[9]和L^2TCP[10]等利用ECN更准确的反馈链路拥塞状态来调节发送速率。然而这些协议仅专注于对拥塞避免阶段的调窗算法改进，而忽略了由于慢启动的激进增窗导致在进入拥塞避免之前就出现了超时问题。

在分析数据中心网络中的流量发现，网络中传

收稿日期：2017－01－16；修回日期：2017－02－24
基金项目：国家自然科学基金(61572530, 61402541, 61462007, 61402542)；湖南省普通高等学校教学改革研究项目(2016jy41)
作者简介：黄家玮(1976－)，男，博士，教授，主要从事计算机网络算法和协议优化方面的研究.

输流大部分是延时敏感的小数据流(90%小于100 KB)[11]，而这些流很可能在慢启动阶段就完成传输。通过实验验证，DCTCP协议即使在并发度较小时，由于慢启动激进的指数增窗方式也会出现超时现象，这对小流的吞吐率影响极大。

本文提出了一种基于ECN的TCP慢启动拥塞控制策略GST来对TCP的慢启动阶段增窗算法进行调节改进，避免因为慢启动增长过快而导致进入拥塞避免之前出现超时现象。测试中分别在DCTCP、D^2TCP和L^2TCP协议上部署了GST，对比了使用和不使用GST反馈调节慢启动增窗的情况。测试结果表明，在使用了GST反馈调节慢启动增窗之后，能将并发度提升3.4倍，吞吐率提升85倍。

1 相关工作

随着互联网应用变得高度多样化和复杂化，为了解决传统传输控制协议的不足，缓解数据中心网络中的TCP Incast现象[12]，国内外许多学者在提高TCP对网络数据拥塞控制的调控能力方面做了大量研究，寻求一些稳定的方法让数量动态变化的网络用户能高效使用有限的网络带宽资源。

最普遍的方案就是基于RTT测量，利用RTT变化来反映当前链路拥塞情况，加速丢包恢复过程从而提高TCP吞吐量。TIMELY[5]作为基于延时的协议，无需交换机支持，利用RTT测量信息来调整发送速率，以获得低延迟和高吞吐量。与TIMELY类似，DX[6]使用端到端延迟来决定增减拥塞窗口，实现近零排队和高瓶颈链路利用率。但是数据中心网络的RTT很小，而且动态变化，精准测量成为改进的关键，直接影响协议性能。

基于经典RED队列管理算法，利用交换机ECN反馈来判断拥塞状态，可以准确地判断拥塞。DCTCP第一次全面讨论尾部延迟，实现了数据中心网络的高突发、低延迟、高吞吐率。但是，DCTCP协议即使当并发度较小时，交换机缓存依然会出现部分流拥塞丢包甚至超时。L^2TCP针对减少短流的流完成时间，以分布式的方法在发送端实现LAS调度。TCP-FITDC[13]是一种基于DCTCP的改进传输协议，通过RTT测量来进一步精确地估计链路拥塞程度，从而实现更准确的拥塞控制。

在接收端，ICTCP[7]自适应地调整接收窗口以调节总吞吐量，从而缓解Incast拥塞。作为接收端驱动的拥塞控制方案，PAC[14]主动地控制接收器侧的ACK的发送速率，以防止Incast拥塞。ARS[15]是一种跨层设计，可缓解高度并发流下的TCP突发问题。ARS通过根据从传输层获取的端到端拥塞信号对应用请求进行批处理来动态调整并发TCP流的数量。

从交换机控制的角度，TCP-PLATO[16]引入了一个标记系统，以确保标记包优先在交换机上排队。因此，TCP发送方可以利用重复的ACK来触发重传，而不是等待超时。CP[17]为了保持TCP自时钟同步，简单地丢弃了包的数据字段而不是整个包，它需要快速和精确地检测丢失的数据包以通知发送端。GIP[18]开始传输具有最小拥塞窗口大小的服务器请求单元以避免分组丢失，并且还冗余地传输服务器请求单元的最后分组以进一步减轻TCP超时。为了缓解由微突发流量造成的丢包，当端口变得过载时，EDT[19]允许交换机的输出端口暂时占用所有共享交换机缓冲区。

以上方法的问题都在于着眼于拥塞避免阶段，并未考虑拥塞可能发生在更早的阶段。为了避免并发传输中慢启动阶段增窗过快而造成的吞吐率崩溃问题，本文旨在缓解慢启动阶段的增窗方式，利用ECN标记来动态调节慢启动阶段的拥塞控制算法。

2 问题分析

为了解数据中心网络中的场景特点，分析协议传输中产生拥塞崩溃的现象的原因，本文就数据中心网络中传输的流量大小和并发度进行研究分析，并从流发送窗口和数据包序列号出发探究造成大量丢包产生超时现象的根本原因。

2.1 流大小分布

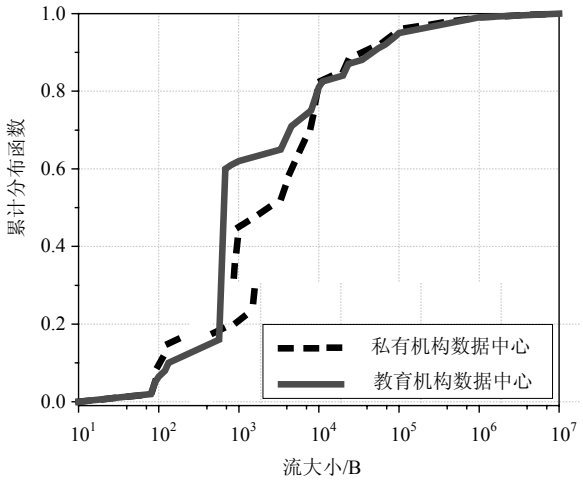


图1 流大小分布图

分析数据中心网络中流量特点，发现大部分应用产生的请求数据量很小[20]。如图1所示，对比私有数据中心(PRV)和教育数据中心网络(EDU)两种类

型的数据中心网络流量特点，结果显示90%的流量都小于100 KB。

而针对不同大小的数据流，计算在无阻塞情况下，按慢启动指数增长方式完成传输所需要的RTT数量。如图2所示，90%的数据流将在4个RTT内传输完成，即大部分数据流能在少数RTT内完成传输，若此时发生超时，将严重影响传输的吞吐率。

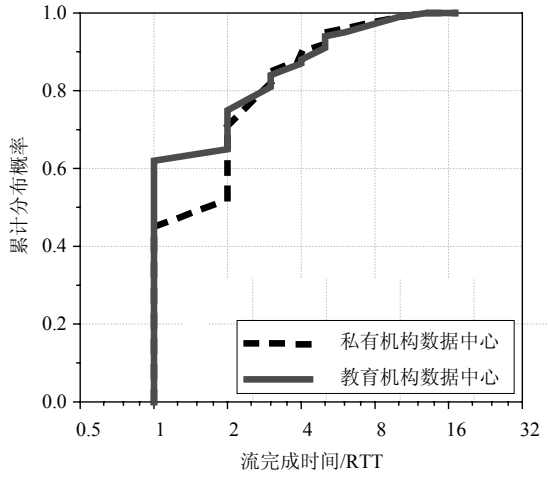


图2　流完成时间分布图

2.2　高并发传输

数据中心网络中的流量分布与其底层文件系统紧密相关。目前普遍使用的Hadoop文件系统(HDFS)[21]的分布式文件系统，使得数据中心网络流量经常表现为多对一的模式。在流到达的时间间隔上，文献[22]的研究结果表明，流平均到达率105 条/s，即有100条流/ms到达。在这样流量高并发的情况下，很容易使得瓶颈链路出现拥塞丢包引起超时，产生TCP Incast现象，引起吞吐率崩溃。

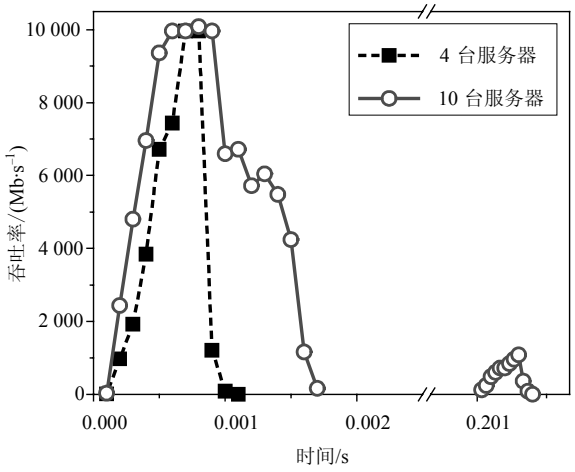
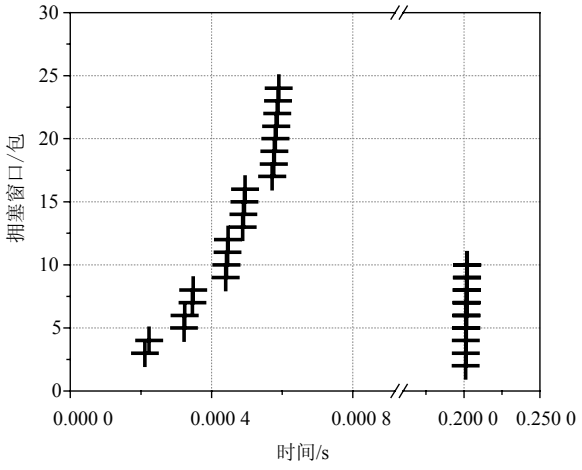


图3　不同并发度吞吐量情况对比

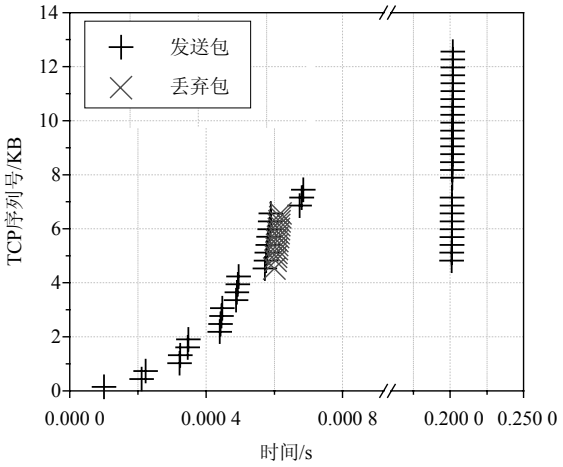
通过实验测试DCTCP协议并发情况下出现的吞吐率崩溃现象。验证实验利用NS2模拟器模拟网络场景，链路带宽为10 Gbps，缓存大小为100 packets，DCTCP协议默认阈值K_{dc}大小为65 packets，RTT为100 μs。从图3可以看出，当小并发(如4台服务器)的时候，数据在0.001 1 s内传输完成。而当加大并发(如10台服务器)的时候，则出现了超时现象，导致链路空闲等待200 ms，数据到0.202 2 s时才完成传输，出现吞吐量崩溃。

2.3　超时产生原因分析

为了探究出现超时现象究竟发生在传输的哪一个阶段，验证慢启动的增长激进导致的超时现象。本文对发送端的窗口大小、发包情况和丢包情况进行实验分析。



a. 拥塞窗口



b. TCP序列号

图4　基础性能结果

图4a显示了慢启动阶段拥塞窗口的增长呈指数级。经过几轮RTT，出现超时现象，一轮RTO之后再进行重传。图4b显示了TCP序列号。大量丢包发生在指数增长阶段。可见，由于慢启动指数增长方

式过于激进才导致大量丢包，从而引发超时。

2.4 小 结

综上对数据中心网络特点的研究和慢启动传输过程中传输控制的分析，得出：1) 数据中心网络中90%的数据流都小于100 KB。2) 分布式的存储模式也使得数据中心网络流量经常表现为多对一的模式，数据流并发度大。3) 慢启动激进的指数增长方式是引发大量丢包产生超时的关键原因。而现有的改进协议都专注于拥塞避免阶段，忽视了激进的慢启动增长阶段。因此本文提出了一种基于ECN的TCP慢启动拥塞控制策略，动态反馈链路拥塞状况来缓解慢启动的增窗速率。

3 协议设计

本文设计目的是新的慢启动拥塞窗口增长算法，缓解激进的指数增长方式，以避免TCP Incast问题。设计要点包括：1) 获取慢启动阶段链路拥塞状况。考虑使用低开销、更精准的拥塞感知方法，将对交换机队列设置新阈值K，利用ECN显式拥塞反馈策略来感知网络中的拥塞状况[23]。2)设计平滑过渡阶段的增窗算法。将从网络容量角度出发，利用缓存队列长度信息作为调窗因子，使得过度阶段的增窗算法既能有效避免增长过快造成大量丢包，又能最大限度利用网络容量。3) 普遍适应性。与原TCP协议兼容，保证链路共享公平性。在现有DCTCP协议基础上，利用已有标志位来反馈当前链路拥塞情况。

需要说明的是，交换机连接并发主机的数量与网络性能息息相关。相关理论研究与真实数据中心网络应用场景都表明[24]，当并发服务器数量较小时，增加服务器数量能提升链路带宽利用率。但当并发服务器的数量增加到一定程度时，即使所有的发送方都降低拥塞窗口(甚至降为极小值)，交换机缓存依然会拥塞丢包，造成网络性能的急剧下降。因此，寻求一种在链路带宽和缓存容忍范围内，最大限度提升并发数量，同时保证较高的链路带宽利用率的方法。

3.1 总体设计

协议设计为在慢启动阶段增加新交换机队列长度阈值K，反馈链路状况，超过K值之后进入慢启动平滑过渡阶段。如图5所示，以DCTCP协议为例，整个传输过程分成3个部分：传统慢启动阶段、平滑过渡阶段和拥塞避免阶段。新阈值K和DCTCP默认阈值K_{dc}决定这3个阶段的分界。

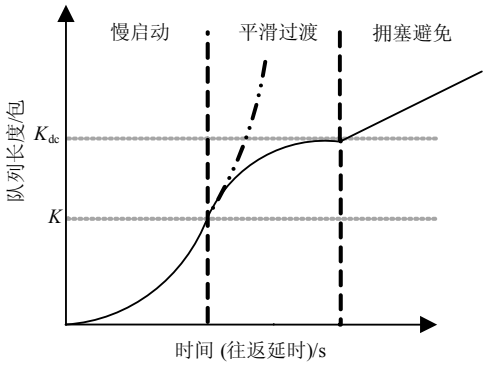


图5 设计框架图

当队列长度超过阈值K之前，协议将保持原慢启动指数增长方式，目的是当跟其他TCP协议共存时保证链路带宽的公平共享。队列长度超过K之后，将进入平滑过渡阶段。若这阶段仍然按原慢启动指数增长方式，窗口将大幅增长，造成大量丢包。因此本文设计平滑过渡阶段拥塞窗口控制算法，使得链路能有较好的利用率，并避免大量丢包成功过渡到拥塞避免阶段。而当队列长度超过K_{dc}之后，将按原协议拥塞避免算法进行窗口调节。

3.2 协议详细设计

平滑过渡阶段算法设计是保证从慢启动阶段平滑过渡到拥塞避免阶段的重要部分。本文通过慢启动阶段的显式拥塞反馈信息，设计新的慢启动增窗算法，使得既能保持窗口持续增长又能缓解慢启动激进的增长幅度，避免因为慢启动增长速度过快造成的吞吐量崩溃现象。

3.2.1 慢启动拥塞标记

现有的显式拥塞反馈机制ECN在IP包头设置两位(bit)的ECN域，一个是ECT(ECN-capable transport)位，另一个是CE(congestion experienced)位。同时，在TCP包头中需要设置两个标志位：ECN-Echo和CWR(congestion window reduced)[25]。

设计旨在慢启动阶中采用交换机上的ECN显式反馈位来获取拥塞信息。为了避免额外的开销，联合使用ECN机制的两个二进制位来编码拥塞信息。两个二进制位可以表达4种显示情况：00,01,10,11。本文利用其中3种情况：00表示ECN不可用；10表示慢启动阶段阈值标记(队列长度超过K)；01表示DCTCP阈值标记位(队列长度超过K_{dc})。这样，在不增加包头开销的情况下，利用已有的ECN标志位编码来表示不同程度的拥塞状态。

3.2.2 拥塞控制算法

发送端根据反馈回来的标记ACK来调节下一轮

发送窗口大小，因此K的取值直接影响下一轮发送窗口的大小。首先，在每个发送端基于标记包的比例来估计当前交换机缓存队列长度q。

发送端的当前发送窗口大小为w，已收到的标记包个数为β。则队列中未标记包个数K与当前队列长度q的比值，跟接收端未收到标记的包个数$w-\beta$与总发送窗口w大小的比值有近似等比关系：

$$\frac{K}{q}=\frac{w-\beta}{w} \tag{1}$$

则可估算出当前队列长度：$q=wK/(w-\beta)$。

本文算法设计的目的是使得既能缓解慢启动激进的指数增窗方式，又能高效的利用网络带宽。因此设计缓冲阶段的算法根据显式反馈信息，使得调窗函数在指数增长和线性增长之间调节变化。调窗函数如下：$\mathrm{cwnd}_{i+1}=\mathrm{cwnd}_i+\mathrm{cwnd}_i^{\partial}$，其中$\partial=(q-K)/(K_{\mathrm{dc}}-K)$。当前队列长度$q$接近$K$，说明链路拥塞情况不严重，则发送端发送窗口越接近指数增长模式；而当q越接近K_{dc}，说明链路状况开始出现拥塞，则使得发送窗口的增长接近线性增长模式。

3.3　K取值分析

本文对慢启动阶段增加一位标志位来显示反馈当前链路拥塞状况。基于ECN的显式拥塞反馈，即在交换机上设置门限值阈值取值为K($K<K_{\mathrm{dc}}$)。当交换缓存大小超过K，则对通过数据包进行标记。发送端收到带标记的包之后，按缓冲阶段的增窗方式发送数据包。此时标记位的阈值K选取变得关键。

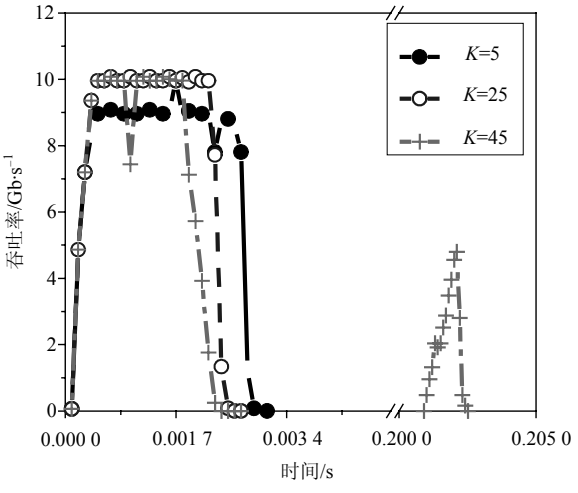


图6　不同的K值对吞吐率的影响

如图6，当K取值很小(如K=5)时，虽未出现超时现象，但网络利用率较低，带宽利用率约为88%。当K取值过大(如K=45)时，则无法及时退出慢启动进入拥塞避免阶段，导致大量丢包出现超时，导致吞吐率崩溃。而K=25时，既取得了100%的带宽利用率，避免了大量丢包引发的超时。

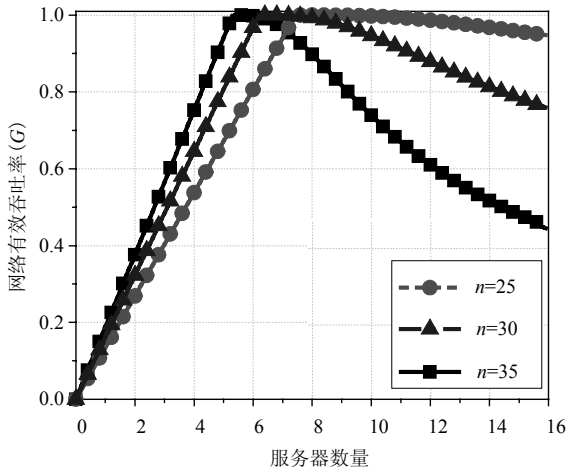


图7　G值变化图

因此，为了选取最佳的K值，将根据网络利用率以及交换机缓存利用率来进行建模分析。众所周知，TCP的超时(一般为200 ms)是导致吞吐量崩溃的主要原因。发送端窗口整窗丢失[20]计算单条流出现超时的概率为：

$$p=\left(1-\frac{B}{nw-C\times\mathrm{RTT}}\right)^{w} \tag{2}$$

式中，B为交换机缓存大小；n为并发服务器台数；w为当前流发送窗口大小；RTT为网络往返时延；C为链路带宽。而当有n台服务器并发时，网络中发生超时产生吞吐率崩溃的概率为：

$$P=1-(1-p)^{n}=1-\left\{1-\left[1-\left(\frac{B}{nw-C\times\mathrm{RTT}}\right)\right]^{w}\right\}^{n} \tag{3}$$

注入网络包数量小于网络容量时，链路利用率为$nw/(B+C\times\mathrm{RTT})$；而发送包数量大于网络容量时，若未超时，利用率近似为1；若出现超时，此时近似利用比率为RTT/RTO。综合考虑网络中超时概率和利用率来计算阈值K的取值，网络有效利用率G为：

$$G=\begin{cases}\dfrac{nw}{B+C\times\mathrm{RTT}} & w\leqslant\dfrac{B+C\times\mathrm{RTT}}{n}\\[2ex](1-P)\times1+P\times\dfrac{\mathrm{RTT}}{\mathrm{RTO}} & w>\dfrac{B+C\times\mathrm{RTT}}{n}\end{cases} \tag{4}$$

本文将在不同的并发度场景下计算不同的G值变化曲线，选取G值最高时的发送端发送窗口大小w来计算最优的阈值K。对G进行求导发现：

① 当$w\leqslant\dfrac{B+C\times\mathrm{RTT}}{n}$时，$\dfrac{\mathrm{d}G}{\mathrm{d}w}=\dfrac{n}{B+C\times\mathrm{RTT}}$

在此阶段，导数恒大于零，函数为增函数。

② 当 $w>\dfrac{B+C\times\mathrm{RTT}}{n}$ 时，

设 $S=\dfrac{B}{C\times\mathrm{RTT}-nw}+1$，$0<S<1$；

设 $T=\dfrac{BnwS^{w-1}}{(C\times\mathrm{RTT}-nw)^2}$，$r=\left(\dfrac{\mathrm{RTT}}{\mathrm{RTO}}-1\right)$；

$$\frac{\mathrm{d}G}{\mathrm{d}w}=[n(1-S^w)^{n-1}](S^w\log S+T) \tag{5}$$

由于 $\mathrm{RTT}<<\mathrm{RTO}$，则 $\mathrm{RTT}/\mathrm{RTO}<<1$；则有 $\dfrac{BnwS^{w-1}}{(C\times\mathrm{RTT}-nw)^2}>>S^w\log S$，因此 $\dfrac{\mathrm{d}G}{\mathrm{d}w}<0$，函数为减函数。

通过代入参数取值，得到随着w取值变化的G值变化如图7所示。图7验证了以上的分析，曲线是随w变化的凸函数，其最高点时拥塞窗口的取值有 $w'=(B+C\times\mathrm{RTT})/2n$。而为了避免超时，至少要在最小超时窗口的前一轮进入慢启动缓冲阶段。因此计算出慢启动调窗队列长度阈值K：

$$K\leqslant n\frac{w'}{2}-C\times\mathrm{RTT} \tag{6}$$

4 性能评估

本文将GST算法应用到不同的数据中心TCP协议，利用NS2模拟器对协议性能进行评估。首先，基于DCTCP协议对GST基本性能进行测试；然后，将GST方法应用到D^{2}TCP、L^{2}TCP等数据中心TCP协议；最后，在两种主流的应用场景Web Search和MapReduce中测试GST的性能。

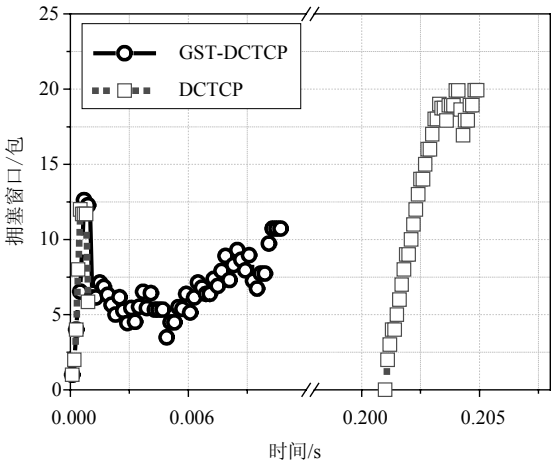
采用典型的数据中心网络多对一的拓扑，多个服务器通过一个瓶颈链路同时发送数据给一个聚合器。默认包大小为1.5 KB，链路带宽为10 Gbps，RTT为100 μs，服务器请求单元(server request unit, SRU)大小为64～512 KB。在其他对比协议中，平均权重因子g为1/16，DCTCP默认交换机标记门限值为65 pkts。

4.1 基础性能结果

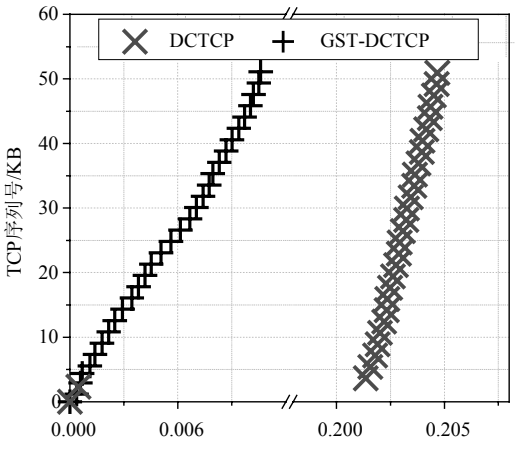
将GST-DCTCP与DCTCP进行基础性能对比。模拟场景为25台服务器并发，缓存大小为100 pkts。其中图8a、图8b选取对象为出现超时现象的单条流数据，而图8c、图8d为链路整体性能对比。

1) 拥塞窗口：从图8a看出，GST-DCTCP经过缓冲慢启动阶段，缓解了慢启动最激进的增长趋势，有效地避免了超时现象；原DCTCP由于慢启动阶段增窗过于激进导致大量丢包，极易出现图中所示的超时现象。

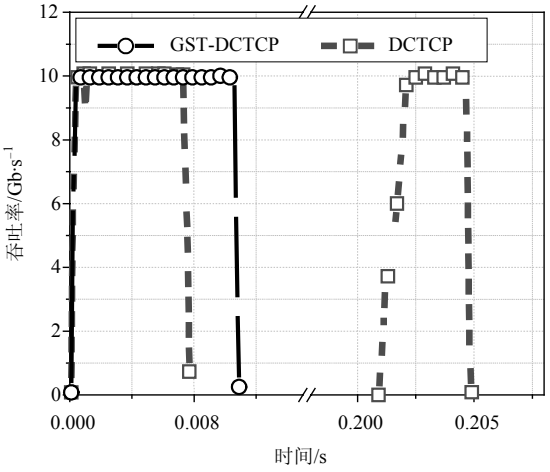
2) TCP序列号：为了从数据包的角度反映GST对超时现象的避免。随机选择一条流观察发送数据包的TCP序号，如图8b所示，原DCTCP因为慢启动的激进增长导致了在0.000 8 s进入超时等待，最后传输完成时间在0.200 47 s；对比GST-DCTCP没有发生超时现象，传输在0.010 7 s全部完成传输。



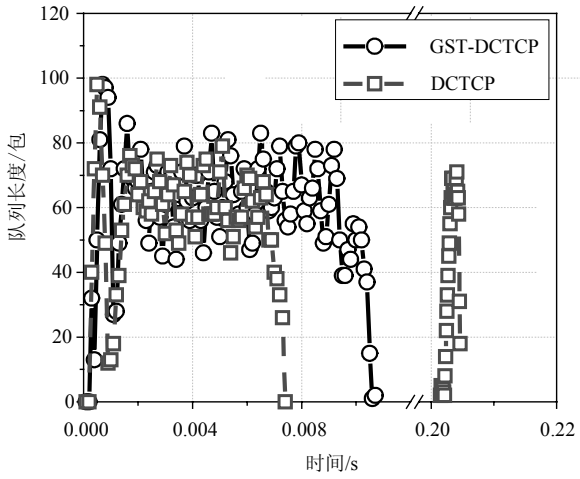
a. 窗口对比情况



b. 序列号对比情况



c. 吞吐量情况对比



d. 队列情况对比

图8　基础性能对比

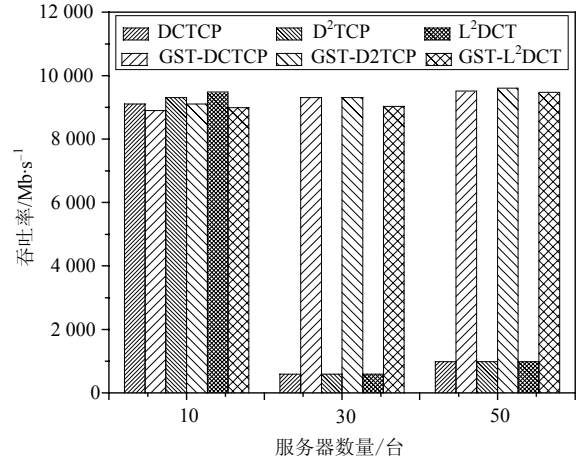
3) 瞬时吞吐量：图8c给出网络瞬时吞吐率变化情况，GST-DCTCP由于很好地缓解慢启动增长及时避免了大量丢包，未出现超时现象，整个传输过程几乎使得瓶颈链路满带宽被利用；而DCTCP则由于超时的影响，链路空等200 ms，极大的影响了传输效率。

4) 交换机缓存队列：图8d从队列的角度看出对数据流的控制，GST-DCTCP能在关键时刻及时缓解慢启动的增长，也能最大限度的利用链路容量(包括交换机缓存容量)，即能在不超时的情况下最大化传输速率。

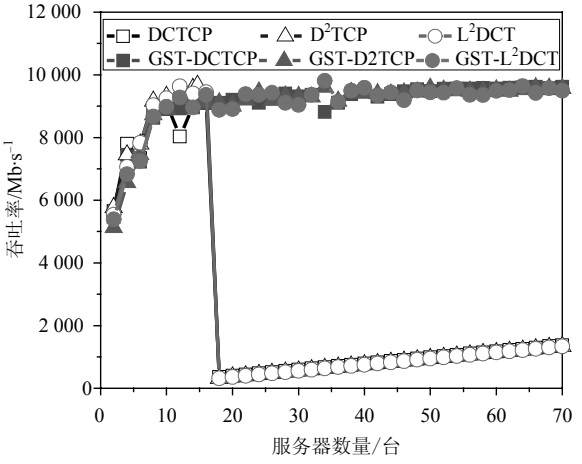
4.2　不同协议性能对比

GST具有协议普适性，能很好地部署到各种数据中心TCP协议。从吞吐量的角度对比DCTCP、D^{2}TCP和L^{2}TCP部署GST前后的性能提升情况。测试场景为并发服务器数量从1台到72台，缓存大小为150 pkts。发送流大小为512 KB。

1) 无背景负载时的性能：如图9a所示，3种协议在小并发情况下由于慢启动的快速增窗使得吞吐率相比部署了GST的协议能有更高的吞吐率；而随着并发服务器数量的增加，原慢启动增窗太过激进，出现超时现象。而对比部署了GST之后，避免了激进的慢启动导致的超时现象，吞吐率基本能保持满带宽传输，大大提升了网络利用率。由图9b可知，DCTCP、D^{2}TCP和L^{2}TCP在较小的并发度都出现吞吐率崩溃现象，之后随着并发数的增加吞吐率都保持在比较低的水平，而缓解慢启动增长之后，有效地避免了超时。并发度由16台提升到70台，提高比例约为3.3倍。



a. 吞吐率情况



b. 支撑流数情况

图9　无背景流不同协议对比

2) TCP背景长流负载测试：文献[7]中对真实数据中心流量负载进行了测量和分析并发现同一机架内并发长流数目的第75分位数为2。因此在本次此时场景中，测试了同时存在2条TCP背景长流时各种协议的性能，每条TCP长流均无限发送数据。

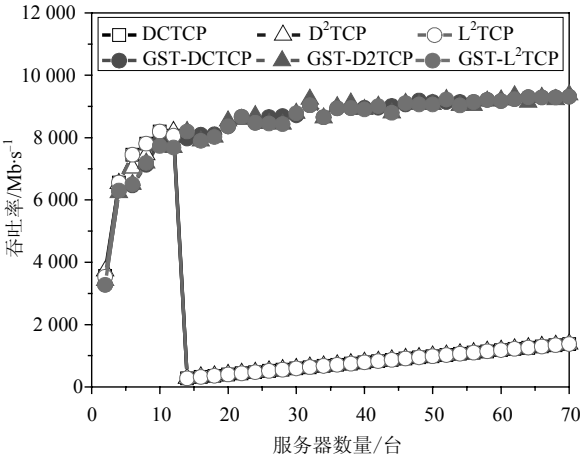


图10　有背景流不同协议对比

图10中，在有2条TCP长背景流情况下，同样在未出现超时前(总流数n<16)，未部署GST的协议吞吐率略高；而当并发数超过16台后，未部署GST的协议都出现了吞吐量崩溃的现象。但是部署了GST的协议则能保持接近满带宽的吞吐率，大幅提升链路利用率。GST支持并发数提升比例约为3.4倍。

4.3 不同场景性能对比

数据中心网络的流量分布表现出两种不同的模型[24]：work seeks bandwidth pattern和scatter gather pattern。而对应网络中应用广泛的两种场景分别为Web search和Map Reduce。针对这两种应用模型，本文在NS2上模拟了两种场景，缓存大小为100 pkts，Web search应用场景固定传输数据总量为2 MB，各服务器发送数据量相等。Map Reduce应用场景每个发送端发送流大小为100 KB。

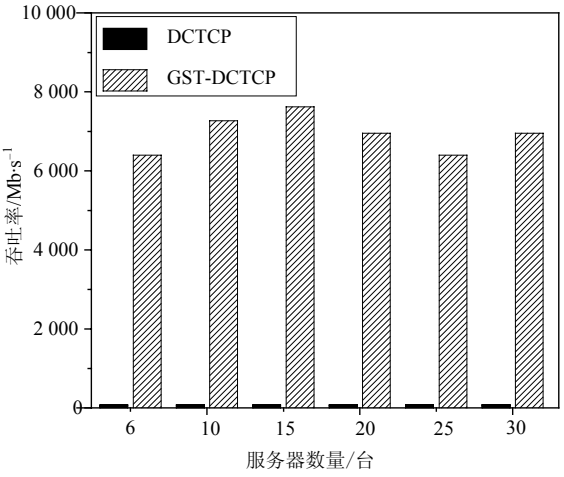


图11 Web Search流量模型

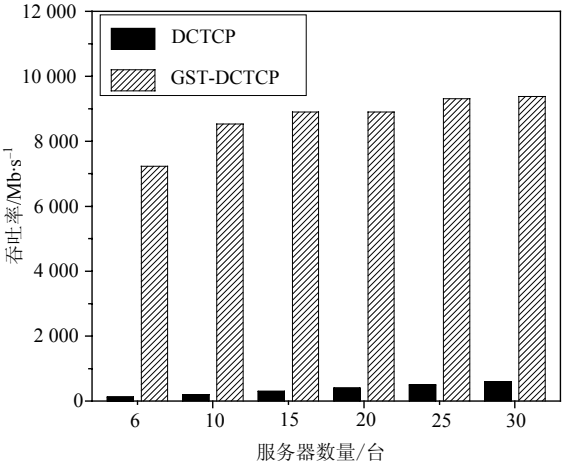


图12 MapReduce流量模型

从图11中可以看出，对于Web search应用模型，固定传输总数据量随着并发服务器数量的增长，由于DCTCP在小并发的情况下就出现了Incast现象，吞吐率一直比较低，而GST-DCTCP避免了慢启动激进增长造成的超时现象，吞吐率保持比较高，而且随着并发度的增加波动也并不是很大。平均吞吐率由79 Mbps提升为6 800 Mbps，提升比例高达85倍。图12表明在MapReduce应用场景下，固定服务器发送数据流大小，增加服务器并发数量，同样GST-DCTCP性能相比DCTCP有了大幅的提升。

5 结 束 语

针对数据中心网络特点的研究，本文总结并发现数据中心网络中大部分都是小数据流，并具有并发度大的特点；而慢启动激进的指数增长方式是引发大量丢包产生超时的关键原因。针对现有研究都关注改进于拥塞避免阶段算法，而忽视了对慢启动阶段激进增长的问题，本文提出了一种基于ECN的TCP慢启动拥塞控制策略，动态反馈网络拥塞状况来缓解慢启动的增窗速率。实验表明，本方法有效避免了数据中心网络并发传输中由于慢启动阶段发送窗口增长过快导致的吞吐率崩溃现象，使得并发度提升了3.4倍，吞吐率倍数提升了85倍，大幅优化网络应用的整体性能。

参 考 文 献

[1] MEISNER D, SADLER C M, BARROSO L A, et al. Power management of online data-intensive services[C]// Proceedings of ISCA. New York, USA: ACM, 2011: 319-330.

[2] JUDD G. Attaining the promise and avoiding the pitfalls of TCP in the datacenter[C]//Proceedings of NSDI. Berkeley, USA: USENIX Press, 2015: 145-157.

[3] REN Yong-mao, ZHAO Yu, LIU Pei, et al. A survey on TCP incast in data center networks[J]. International Journal of Communication Systems, 2014, 27(8): 1160-1172.

[4] LIU Fang-ming, GUO Jian, HUANG Xiao-meng. EBA: Efficient bandwidth guarantee under traffic variability in datacenters[C]//IEEE/ACM Transactions on Networking. Piscataway: IEEE, 2017, 25(1): 506-519.

[5] MITTAL R, LAM V T, DUKKIPATI N, et al. TIMELY: RTT-based congestion control for the datacenter[C]// Proceedings of ACM SIGCOMM. New York, USA: ACM, 2015: 537-550.

[6] LEE C, PARK C, JANG K, et al. Accurate Latency-based congestion feedback for datacenters[C]//Proceedings of USENIX ATC. Berkeley, USA: USENIX Press, 2015: 403-415.

[7] WU Hai-tao, FENG Zhen-qian, GUO Chuan-xiong, et al. ICTCP: Incast congestion control for TCP in data center networks[C]//Proceedings of CoNEXT. New York, USA: ACM, 2010.

[8] ALIZADEH M, GREENBERG A, MALTZ D A, et al. Data center TCP (DCTCP)[C]//Proceedings of ACM SIGCOMM. New York, USA: ACM, 2010: 63-74.
[9] VAMANAN B, HASAN J, VIJAYKUMAR T N. Deadline-aware datadcenter TCP (D2TCP)[C]//Proceedings of ACM SIGCOMM. New York, USA: ACM, 2012: 115-126.
[10] MUNIR A, QAZI I A, UZMI Z A, et al. Minimizing flow completion times in data centers[C]//Proceedings of INFOCOM. Piscataway, USA: IEEE, 2013: 2157-2165.
[11] CHEN Yan-pei, GRIFFITH R, LIU Jun-da, et al. Understanding TCP incast throughput collapse in datacenter networks[C]//Proceedings of WREN Workshop. New York, USA: ACM, 2009: 73-82.
[12] VASUDEVAN V, PHANISHAYEE A, SHAH H, et al. Safe and effective fine-grained TCP retransmissions for datacenter communication[C]//Proceedings of ACM SIGCOMM. New York, USA: ACM, 2009: 303-314.
[13] ZHANG Jun, WEN Jiang-tao, WANG Jing-yuan, et al. TCP-FITDC: an adaptive approach to TCP incast avoidance for data center applications[C]//Proceedings of ICNC. Piscataway, USA: IEEE, 2013: 1048-1052.
[14] BAI Wei, CHEN Kai, WU Hai-tao, et al. PAC: Taming TCP incast congestion using proactive ACK control[C]//Proceedings of ICNP. Piscataway, USA: IEEE, 2014: 385-396.
[15] HUANG Jia-wei, HE Tian, HUANG Yi, et al. ARS: Cross-layer adaptive request scheduling to mitigate TCP incast in data center networks[C]//Proceedings of INFOCOM. Piscataway, USA: IEEE, 2016: 1-9.
[16] SHUKLA S, CHAN S, TAM A S W, et al. TCP PLATO: Packet labelling to alleviate time-out[J]. IEEE Journal on Selected Areas in Communications, 2014, 32(1): 65-76.
[17] CHENG Peng, REN Feng-yuan, SHU Ran, et al. Catch the whole lot in an action: Rapid precise packet loss notification in data centers[C]//Proceedings of USENIX NSDI. Berkeley, USA: USENIX Press, 2014: 17-28.
[18] ZHANG Jiao, REN Feng-yuan, TANG Li, et al. Taming TCP incast throughput collapse in data center networks [C]//Proceedings of ICNP. Piscataway, USA: IEEE, 2013: 1-10.
[19] SHAN Dan-feng, JIANG Wan-chun, REN Feng-yuan, et al. Absorbing micro-burst traffic by enhancing dynamic threshold policy of data center switches[C]//Proceedings of INFOCOM. Piscataway, USA: IEEE, 2015: 118-126.
[20] BENSON T, AKELLA A, MALTZ D A. Network traffic characteristics of data centers in the wild[C]//Proceeding of IMC. New York, USA: ACM, 2010: 267-280.
[21] BORTHAKUR D. The hadoop distributed file system: Architecture and design[EB/OL]. [2016-10-16]. https://svn.eu.apache.org/repos/asf/hadoop/common/tags/release-0.16.3/docs/hdfs_design.pdf.
[22] KANDULA S, SENGUPTA S, GREENBERG A, et al. The nature of data center traffic: Measurements and analysis [C]//Proceedings of ACM SIGCOMM. New York, USA: ACM, 2009.
[23] BAI W, CHEN L, CHEN K, et al. Enabling ECN in multi-service multi-queue data centers[C]//Proceedings of NSDI. Berkeley, USA: USENIX Press, 2016: 537-549.
[24] CHEN Wen, REN Feng-yuan, XIE Jing, et al. Comprehensive understanding of TCP incast problem [C]//Proceedings of INFOCOM. Piscataway, USA: IEEE, 2015: 1688-1696.
[25] XIA Yong, SUBRAMANIAN L, STOICA I, et al. One more bit is enough[C]//Proceedings of ACM SIGCOMM. New York, USA: ACM, 2005: 37-48.

编 辑 税 红