一种基于链路带宽估计的 TCP 慢启动算法

李 云[1),2)] 陈前斌[2)] 隆克平[2)] 吴诗其[1)]

[1)](电子科技大学信息系统研究所 成都 610054)

[2)](重庆邮电学院光互联网及无线信息网络研究中心 重庆 400065)

摘 要 在慢启动阶段,TCP 以指数方式增加其拥塞窗口,这导致了慢启动阶段的多包丢失,并恶化了 TCP 的性能.该文对 TCP 连接的等效带宽进行了深入的理论分析.在此基础上,提出了一种改进的 TCP 慢启动算法——基于链路带宽估计的 TCP 慢启动算法,并通过仿真对其吞吐量、公平性和兼容性进行了评估.仿真的结果表明,该算法避免了慢启动阶段的多包丢失,并能有效改进 TCP 的性能,是简单、实用和有效的.

关键词 TCP; 慢启动; 等效带宽; 链路带宽估计

中图法分类号 TP393

A TCP Slow-Start Algorithm Based on Link Band-Width Estimation

LI Yun[1),2)] CHEN Qian-Bin[2)] LONG Ke-Ping[2)] WU Shi-Qi[2)]

[1)](*Institute of Information, University of Electronic Science and Technology of China, Chengdu*, 610054)

[2)](*Special Research Centre for Optical Internet & Wireless Information Networks, Chongqing University of Posts & Telecommunications, Chongqing* 400065)

Abstract Because TCP exponentially grows its congestion window during the initial slow start period, many packets are lost during this period. This makes TCP coarse-grained timeout and deteriorates TCP's performance. For the large bandwidth×delay product networks, this is more serious. Based on the deep analysis of the TCP connection, this paper gives the equivalent bandwidth concept and deduces its analytical expression. According to this theoretic analysis, this paper proposes an improved TCP slow-start algorithm based on link band-width estimation. At the same time, we evaluate the algorithm's performance, fairness and compatibility on the NS platform. The results of simulation prove that this algorithm can resolve the problem, of which many packets are lost during the initial slow start period. It increases the good throughput of TCP flow. Because this algorithm only modifies the sender of TCP a little, it is simple, practical and effective.

Keywords TCP; slow-start; equivalent bandwidth; link band-width estimation

1 引 言

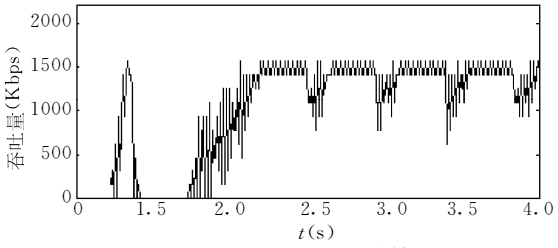
传输控制协议(TCP)是 Internet 上广泛使用的一种基于滑动窗口的传输层协议.它是面向连接的,能保证数据传输的有序和可靠性.TCP 的最大特点在于其具有自适应能力的拥塞控制策略,它能根据网络的拥塞情况,自动调整拥塞窗口的大小,从而使

收稿日期:2001-09-03;修改稿收到日期:2002-08-08.本课题得到国家"八六三"高技术研究发展计划项目(2001AA120303)、重庆市科委计划项目(D2002-35)、重庆邮电学院青年教师基金(A2002-20)和重庆市/信息产业部重点实验室开放课题基金(M2002-06)联合资助.

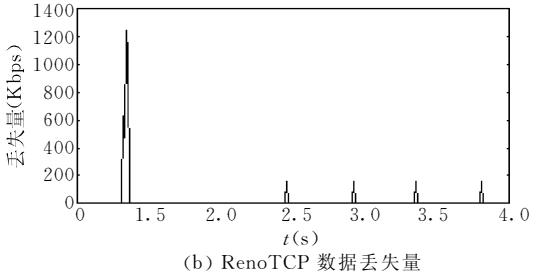
李 云,男,1974 年生,博士研究生,主要研究方向为 TCP/IP 改进、无线 TCP. E-mail:Liyun@cqupt.edu.cn. **陈前斌**,男,1967 年生,博士,副教授,主要研究领域为个人通信技术. **隆克平**,男,1968 年生,博士,教授,主要研究领域为光突发交换与性能分析、Internet QoS 及宽带网络生存性研究. **吴诗其**,男,教授,博士生导师,主要研究领域为无线通信.

TCP 的数据发送速率适应网络的拥塞状况. 最初，TCP 采用的是一种单纯的加性增加、乘性减小(Additive-Increase Multiple-Decrease，AIMD)的拥塞窗口控制策略[1]. 它的原理如下：在 TCP 完成三次握手建立 TCP 连接后，在 TCP 的发送端，拥塞控制策略进入慢启动(Slow-Start，SS). 在 SS 阶段，拥塞窗口从一个 MSS(Maximum Segment Size)开始，以指数形式增加拥塞窗的大小，直到检测到丢包. 当检测到丢包后，TCP 将拥塞窗口门限(ssthresh)设为当前拥塞窗口的一半，同时返回到慢启动过程. 当拥塞窗口大小达到 ssthresh 后，TCP 进入冲突避免阶段(CA). 在 CA 段，拥塞窗口以每 ACK MSS×MSS/W(W 为当前拥塞窗口大小)的速率增加，直到检测到丢包. 单纯的 AIMD 算法的最大缺点在于其频繁的粗粒度超时(coarse-grained timeout)，这种粗粒度超时将恶化 TCP 的性能，降低 TCP 的吞吐量. 为此，人们提出了一种快速重传快速恢复(Fast Retransmission Fast Recovery，FRFR)[2]算法，快速重传算法不再等到重传超时后才重传丢包，而是在收到对同一 TCP 包的连续多个(一般为三个)重复 ACK 时即重传丢包. 在一个典型的 TCP 连接中，快速重传算法能减少几乎 50% 的粗粒度超时，提高 20%左右的 TCP 的吞吐量. 快速恢复算法对 TCP 拥塞控制算法进行了进一步的改进，网络在 CA 段的丢包不会使 TCP 重新进入慢启动，发送方 TCP 只是将 ssthresh 设为当前拥塞窗口的一半，同时将拥塞窗口减半并保持在 CA 段. 快速恢复算法能进一步改进 TCP 的性能.

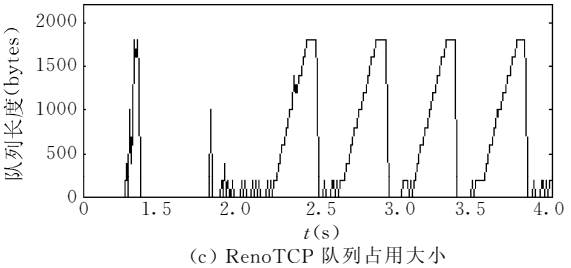
快速重传快速恢复算法主要侧重于改进 TCP 的 SS-CA 周期段的性能，TCP 建立连接后到第一个 CA 段之间，仍然采用老式的慢启动算法. TCP 建立连接后，由于 TCP 对当前网络可用带宽缺乏先验知识，TCP 采用指数形式增加拥塞窗口的大小，直到检测的丢包，并将网络可用带宽估计为当前数据发送速率的一半. 这种方式的最大缺点在于它导致连续的突发性的丢包(对带宽为 B(bps)的 TCP 连接链路，最大丢包数可达 $B/(8\times MSS)$). 这种连续的丢包不仅增加了包的丢失率，同时会引起发送端 TCP 粗粒度的超时，降低 TCP 的吞吐量. 随着大容量的链路引入 Internet，TCP 连接链路的带宽容量将增加，这一问题将显得更加突出. 我们在 UC Berkeley 大学的 NS-2① 仿真环境下研究了 TCP 慢启动阶段的突发丢包对 TCP 连接的吞吐量的影响. 该仿真通过在一条带宽为 1.5Mbps、延时为 10ms 的链路上直接运行 RenoTCP 来进行，其中队列采用 RED，队列大小为 10 个 packets. TCP 从 1.2s 开始承载流量，运行时间为 2.8s. 仿真结果如图 1 所示，从图 1 可知，由于慢启动阶段的突发丢包，TCP 在慢启动阶段结束后即进入了一段吞吐量为零的时间(1.35～1.8s). 尽管此时链路和队列空闲.



(a) RenoTCP 吞吐量



(b) RenoTCP 数据丢失量



(c) RenoTCP 队列占用大小

图 1 RenoTCP 的慢启动对 TCP 吞吐量、数据丢失率和队列长度的影响

基于选择确认(SACK[3])选项的 TCP 拥塞控制算法(如 FACK 拥塞控制算法[4,5])虽然能使 TCP 从多包丢失中恢复，但它们需要接收端 TCP 的配合，同时，它们增加了 TCP 的额外开销. 例外，在文

① The networks simulator ns-2. http://www.isi.edu/nsnam/ns/

献[6]中,作者建议通过对快速重传快速恢复算法的改进来使 TCP 从多包丢失中恢复,它同基于 SACK 选项的 TCP 拥塞控制算法相比,不需接收端 TCP 的配合,实现简单,但它同前者一样,并不能避免慢启动阶段的多包丢失.

为在无线网络中提供数据业务,近年来,无线 TCP 已成为研究热点.针对无线网络自身的特点,人们已提出了许多无线 TCP 改进方案[7],它们虽然能够在一定程度上改进 TCP 在无线环境中的性能,但其实现复杂,不能同原有 TCP 很好兼容,且同样地不能避免慢启动阶段的多包丢失.

本文主要针对 TCP 慢启动阶段存在的这一缺点,在对 TCP 连接的等效带宽进行深入分析的基础上,提出了一种称为基于链路带宽估计的 TCP 慢启动算法(BLBE-SS),该算法通过在慢启动阶段预测 TCP 连接的等效带宽,根据预测带宽来设置 TCP 的 ssthresh,从而避免 TCP 慢启动阶段的多包丢失,改进 TCP 的性能.

2　基于 TCP 连接链路带宽估计的慢启动算法

2.1　基本原理

设一 TCP 连接的源和目的端分别为 S,D,该 TCP 连接通过路由器 $R_1,R_2,R_3,\cdots,R_{n-1},R_n$,且 S 与 R_1,R_1 与 R_2,R_2 与 R_3,$\cdots$,R_{n-1} 与 R_n,R_n 与 D 之间的物理链路带宽分别为 $B_1,B_2,B_3,\cdots,B_n,B_{n+1}$.该 TCP 连接如图 2 所示.

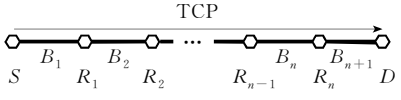


图 2　TCP 连接简图

对 S 到 D、大小为 L 的 TCP 包 P,设 S(为了方便,下面我们称 S 为路由器 R_0)、路由器 $R_i(i=1,2,\cdots,n)$ 的队列处理延时为 $D_{\mathrm{q}i}$,第 $i(i=1,2,\cdots,n+1)$ 条链路(该链路对应的带宽为 B_i)的传播延时为 $D_{\mathrm{pp}i}$,各层协议处理(如 IP 选路等)延时为 $D_{\mathrm{pr}i}$,则对通过该 TCP,大小分别为 L_1,L_2 的包 P_1,P_2,其对应的延时为

$$Delay^{(1)}=\sum_{i=1}^{n+1}\frac{L_1}{B_i}+\sum_{i=0}^{n}D_{\mathrm{q}i}^{(1)}+\sum_{i=1}^{n+1}D_{\mathrm{pp}i}^{(1)}+\sum_{i=0}^{n}D_{\mathrm{pr}i}^{(1)} \tag{1}$$

$$Delay^{(2)}=\sum_{i=1}^{n+1}\frac{L_2}{B_i}+\sum_{i=0}^{n}D_{\mathrm{q}i}^{(2)}+\sum_{i=1}^{n+1}D_{\mathrm{pp}i}^{(2)}+\sum_{i=0}^{n}D_{\mathrm{pr}i}^{(2)} \tag{2}$$

通常,对链路 $i(i=1,2,\cdots,n+1)$,其传播延时是因电波的传播引起的,跟包的大小无关;同时,各层协议对包的处理是以包(或帧)为单位,在不对包进行分片的情况下,其处理时间基本上与包的大小无关.因此,我们有

$$D_{\mathrm{pp}i}^{(1)}=D_{\mathrm{pp}i}^{(2)},\ i=1,2,\cdots,n+1 \tag{3}$$

$$D_{\mathrm{pr}i}^{(1)}=D_{\mathrm{pr}i}^{(2)},\ i=0,1,\cdots,n \tag{4}$$

由式(2)减去式(1),并代入式(3),(4)得

$$Delay^{(2)}-Delay^{(1)}=(L_2-L_1)\cdot\sum_{i=1}^{n+1}\frac{1}{B_i}+\sum_{i=0}^{n}D_{\mathrm{q}i}^{(2)}-\sum_{i=0}^{n}D_{\mathrm{q}i}^{(1)} \tag{5}$$

对 $D_{\mathrm{q}i}^{(1)}$ 和 $D_{\mathrm{q}i}^{(2)}$ $(i=0,1,\cdots,n)$,其值跟包 P_1,P_2 的大小、P_1,P_2 到达路由器 i 的时间以及包 P_1,P_2 到达路由器 i 时,路由器 i 采用的队列机制和路由器 i 正处理的流(类)的个数有关.下面,我们分情况对其进行讨论.

情况 1.　当路由器 i 不具备带宽分配功能(如只采用 FIFO,RED 或 WRED① 队列机制)或路由器 i 正在处理的流(类)的个数为 0 时,路由器 i 将直接对包 P_1,P_2 进行存储转发,而不会根据当前路由器中流(类)的多少和各流(类)的优先级,按包 P_1,P_2 的大小对其进行调度.对于这种情形,$D_{\mathrm{q}i}^{(2)}-D_{\mathrm{q}i}^{(1)}$ 只跟包 P_1,P_2 到达路由器 i 的时间有关.当 P_2 到达路由器 i 时,如果 P_1 正位于路由器 i 的队列中,且还没开始发送,则 $D_{\mathrm{q}i}^{(2)}-D_{\mathrm{q}i}^{(1)}$ 取得最大值 L_1/B_{i+1};当 P_2 到达路由器 i 时,如果 P_1 已经发送完毕,则 $D_{\mathrm{q}i}^{(2)}-D_{\mathrm{q}i}^{(1)}$ 取得最小值,0.因此,在这种情况下,我们有

$$D_{\mathrm{q}i}^{(2)}-D_{\mathrm{q}i}^{(1)}=\alpha_i\cdot L_1/B_{i+1},0\le\alpha_i\le 1 \tag{6}$$

情况 2.　当路由器 j 具备带宽分配功能(如采用 WFQ,W^{2}FQ[8] 之类的队列机制)且路由器 j 正在处理的流(类)的个数不为 0 时,路由器 j 将根据当前队列中流(类)的个数和各流(类)的优先级,按包 P_1,P_2 的大小对包 P_1,P_2 进行调度.在这种情形下,$D_{\mathrm{q}j}^{(2)}-D_{\mathrm{q}j}^{(1)}$ 不仅跟包 P_1,P_2 到达路由器 j 的时间有关,而且跟路由器当前正在处理的流(类)的个数和流(类)的优先级有关.对大小为 L 的包 P,设路由器 j 根据其队列调度算法对包 P 的延时为 $D_{\mathrm{q}j}$,对于这种队列延时 $D_{\mathrm{q}j}$,我们可将其等效为包 P 通过一段带宽为 $B_{\mathrm{q}j}$ 的链路的延时,这里 $B_{\mathrm{q}j}=P/D_{\mathrm{q}j}$.单纯从包 P 的延时意义上看,路由器 j 的队列机制可等效为 FIFO 队列机制加上一条带宽为 $B_{\mathrm{q}j}$ 的链

① Technical Specification from Cisco. Distributed weighted random early detection. URL: http://www.cisco.com/univer- cd/cc/td/doc/produc t/software/ios111/cc111/wred.pdf

路.这样,同情况 1,当 P_2 到达路由器 j 时,如果 P_1 正位于路由器 j 的队列中,且还没开始发送,则 $D_{qj}^{(2)}-D_{qj}^{(1)}$ 取得最大值 $\frac{L_2-L_1}{B_{qj}}+\frac{L_1}{B_{qj}}$;当 P_2 到达路由器 j 时,如果 P_1 已经发送完毕,则 $D_{qj}^{(2)}-D_{qj}^{(1)}$ 取得最小值 $(L_2-L_1)/B_{qj}$.因此,在这种情况下,我们有

$$D_{qj}^{(2)}-D_{qj}^{(1)}=\frac{L_2-L_1}{B_{qj}}+\alpha_j\cdot\frac{L_1}{B_{qj}},0\leq\alpha_j\leq 1 \tag{7}$$

设属于情况 1 的所有路由器构成集合 A,属于情况 2 的所有路由器构成集合 B,且 $|A|=N_1$,$|B|=N_2$,则 $A\cap B=\varnothing$,$N_1+N_2=n+1$.进一步有

$$\sum_{i=0}^{n}D_{qi}^{(r)}=\sum_{i=1,R_i\in A}^{N_1}D_{qi}^{(r)}+\sum_{j=1,R_j\in B}^{N2}D_{qi}^{(r)},r=1,2 \tag{8}$$

根据式(6),(7),有

$$\sum_{i=1,R_i\in A}^{N_1}D_{qi}^{(2)}-\sum_{i=1,R_i\in A}^{N_1}D_{qi}^{(1)}=\sum_{i=1,R_i\in A}^{N_1}\alpha_i\cdot\frac{L_1}{B_{i+1}} \tag{9}$$

$$\sum_{j=1,R_j\in B}^{N_2}D_{qj}^{(2)}-\sum_{j=1,R_j\in B}^{N_2}D_{qj}^{(1)}$$

$$=\sum_{j=1,R_j\in B}^{N_2}\frac{L_2-L_1}{B_{qj}}+\sum_{j=1,R_j\in B}^{N_2}\alpha_j\cdot\frac{L_1}{B_{qj}} \tag{10}$$

将式(8),(9),(10)代入式(5)并整理得

$$Delay^{(2)}-Delay^{(1)}$$

$$=(L_2-L_1)\cdot\sum_{i=1}^{n+1}\frac{1}{B_i}+\sum_{i=1,R_i\in A}^{N_1}\alpha_i\cdot\frac{L_1}{B_{i+1}}+$$

$$\sum_{j=1,R_j\in B}^{N_2}\frac{L_2-L_1}{B_{qj}}+\sum_{j=1,R_j\in B}^{N_2}\alpha_j\cdot\frac{L_1}{B_{qj}} \tag{11}$$

对式(11),当 $L_2\gg L_1$ 时,有 $L_2-L_1\approx L_2$,从而将式(11)近似为

$$Delay^{(2)}-Delay^{(1)}$$

$$=(L_2-L_1)\cdot\sum_{i=1}^{n+1}\frac{1}{B_i}+\sum_{j=1,R_j\in B}^{N_2}\frac{L_2-L_1}{B_{qj}}$$

$$=(L_2-L_1)\cdot\left[\sum_{i=1,R_i\in A}^{N_1}\frac{1}{B_i}+\sum_{j=1,R_j\in B}^{N_2}\left(\frac{1}{B_j}+\frac{1}{B_{qj}}\right)\right] \tag{12}$$

在式(12)中,设 $\frac{1}{B_{TCP}^{j}}=\frac{1}{B_j}+\frac{1}{B_{qj}}$,即 $B_{TCP}^{j}=\frac{1}{\frac{1}{B_j}+\frac{1}{B_{qj}}}$,则由式(12)得

$$Delay^{(2)}-Delay^{(1)}=$$

$$(L_2-L_1)\cdot\left(\sum_{i=1,R_i\in A}^{N_1}\frac{1}{B_i}+\sum_{j=1,R_j\in B}^{N_2}B_{TCP}^{j}\right) \tag{13}$$

设 $L_2-L_1=L$,$Delay^{(2)}-Delay^{(1)}=Delay$,则对于式(13),我们可以这样理解:大小为 L 的包,通过该 TCP 连接所需的延时为 $Delay$.而 $\frac{1}{B_i}$ 为路由器 $i(R_i\in A)$ 在其下游链路上分配给该 TCP 的实际带宽;B_{TCP}^{j} 为路由器 $j(R_j\in B)$ 在其下游链路上分配给该 TCP 的实际带宽.我们定义

$$B_{eq}=\frac{1}{\left(\sum_{i=1,R_i\in A}^{N_1}\frac{1}{B_i}+\sum_{j=1,R_j\in B}^{N_2}B_{TCP}^{j}\right)}$$

为图 2 所示 TCP 连接通过一系列路由器和链路后最终得到的等效带宽,则图 2 可等效为图 3.

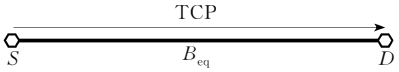


图 3 图 2 的等效

即 S,D 通过一带宽为 B_{eq} 的链路直接相连.当 S 通过该链路上的 TCP 向 D 发送数据包时,为有效利用带宽,同时避免慢启动阶段的多包丢失,我们应该将慢启动阶段的 TCP 拥塞窗口门限 ssthresh 设为 $\frac{B_{eq}\cdot RTT}{MSS}$,其中,$RTT$ 为大小为 MSS 的包通过图 2 所示 TCP 连接的往返时间[1].

2.2 基于链路带宽估计的 TCP 慢启动算法

虽然我们在 2.1 节中给出了利用延时测量 TCP 连接等效带宽的原理,但在 TCP 慢启动算法中,却不适宜直接使用这一原理,原因在于:延时的获取需要发送端和接收端协调配合,即发送端需要在发送的 TCP 包中加入发送时间戳,接收端在收到该包后,需根据其发送时间戳,得到该包的延时.然后,接收端再将该延时告诉发送端.这就需要 TCP 接收端的配合,同时增加了额外的开销.因此,我们采用根据给定 TCP 包的 RTT 来估计带宽,方法如下:

对长度分别为 L_1,L_2 的 TCP 包 P_1,P_2,其对应的 ACK 分别为 ACK_1,ACK_2.设 P_1,P_2 对应的发送端到接收端的延时分别为 $Delay_{P_1}$,$Delay_{P_2}$;ACK_1,ACK_2 对应的接收端到发送端的延时分别为 $Delay_{ACK_1}$,$Delay_{ACK_2}$,则

$$RTT_1=Delay_{P_1}+Delay_{ACK_1},$$

$$RTT_2=Delay_{P_2}+Delay_{ACK_2},$$

$$RTT_2-RTT_1$$

$$=Delay_{P_2}-Delay_{P_1}+Delay_{ACK_2}-Delay_{ACK_1}.$$

仍以图 2 的 TCP 连接为例,由于不带数据的 ACK 包的大小相等,且同 TCP 包相比,ACK 包小得多,

因此,在相同的假设条件下,采用同 2.1 节相同的推理方式,我们有 $Delay_{ACK_2} \approx Delay_{ACK_1}$,这样,

$$RTT_2 - RTT_1 = Delay_{P_2} - Delay_{P_1} =$$

$$(L_2 - L_1) \cdot \left(\sum_{i=1, R_i \in A}^{N_1} \frac{1}{B_i} + \sum_{j=1, R_j \in B}^{N_2} \frac{1}{B_{\mathrm{TCP}}^j} \right).$$

其等效带宽 B_{eq} 为

$$B_{\mathrm{eq}} = \frac{(L_2 - L_1)}{(RTT_2 - RTT_1)}.$$

上面,我们给出了通过发送连续的长度分别为 L_1,L_2 的包 P_1,P_2 来估计 TCP 连接的链路带宽的原理和方法.在实际的网络环境中,可能出现 P_1,P_2 中的一个或者两者都被丢弃的现象,在这种情况下,发送方应该重新进行链路带宽的测试尝试,直到测试成功或 TCP 慢启动阶段结束.

考虑到以上情况,根据链路带宽估计,改进的 TCP 慢启动算法(BLBE-SS)如下所述:

1. TCP 连接建立后,发送端连续发送两个大小分别为 L_1,L_2($L_1 < L_2$)的包 P_1,P_2.

2. 如果发送端收到接收端对 P_1,P_2 的确认,则记录其 RTT 分别为 RTT_1,RTT_2,并根据 RTT_1,RTT_2 和 L_1,L_2 估计 TCP 连接的等效带宽:

$$B_{\mathrm{eq}} = (L_2 - L_1)/(RTT_2 - RTT_1).$$

据估计的链路带宽 B 设置

$ssthresh = \max(1, B_{\mathrm{eq}} \times RTT/MSS)$,转步 3;否则,返回到步 1,直到 TCP 慢启动阶段结束.

3. TCP 按指数方式增加拥塞窗口大小,当拥塞窗口大小达到 *ssthresh* 后,以同 CA 段相同的方式线性增加拥塞窗口,直到检测到丢包.之后采用快速重传快速恢复算法管理拥塞窗口.

在 2.1 节中,我们在假设 $L_2 \gg L_1$ 的条件下由式(11)得到式(12).在实际算法的实现中,L_2 和 L_1 的值是确定的,这将引入估计误差,显然,该误差值跟 N_2,L_2 和 L_1 的大小有关.L_2 越大,N_2 和 L_1 越小,误差越小.N_2 由网络负荷和路由器采用的队列机制决定,我们不能通过算法对其进行控制.对于 L_2 和 L_1,我们能对其进行合理选择.通常,在选择 L_2 和 L_1 时,应该使 L_2 是 L_1 的 5 倍以上,同时不要使 L_2 因过大而导致其被 IP 分段和重组.

3　性能仿真分析

为研究改进的 TCP 慢启动算法的性能,我们采用 UC Berkeley 大学的仿真平台(NS-2)对其进行仿真.仿真时采用的网络拓扑如图 4 所示.该拓扑由 6 台主机和 4 台路由器组成,主机 H1a,H1b,H1c 和路由器 Ra,主机 H2a,H2b,H2c 和路由器 Rb 均通过 100Mbps 的以太网相连;路由器 Ra,R1 之间,路由器 Rb,R2 之间也通过带宽为 10Mbps、时延为 10ms 的链路相连;路由器 R1,R2 之间通过带宽为 1.5Mbps、时延为 50ms 的链路相连.Ra,Rb 采用的队列机制均为 RED,R1,R2 采用的队列机制为 FQ.

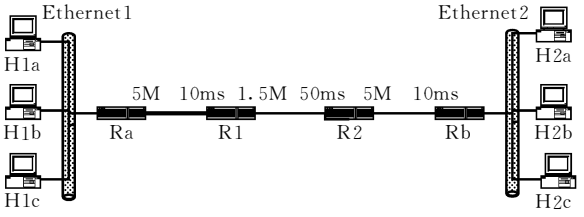


图 4　仿真拓扑

仿真时的 TCP 相关参数和包 P_1,P_2 的大小如表 1 所示.

表 1　仿真参数

参数名	参数值
MSS	1600bits
窗口上界	300
拥塞窗口初值	1
ssthresh 初值	0
*P*1 大小	1600bits
*P*2 大小	8000bits

下面,我们将分别在单连接和多连接的情况下,通过对采用了 BLBE-SS 的 RenoTCP(以下简称 BLBE RenoTCP)和未采用 BLBE-SS 的 RenoTCP 的比较来论证 BLBE RenoTCP 对慢启动阶段性能的有效改进.

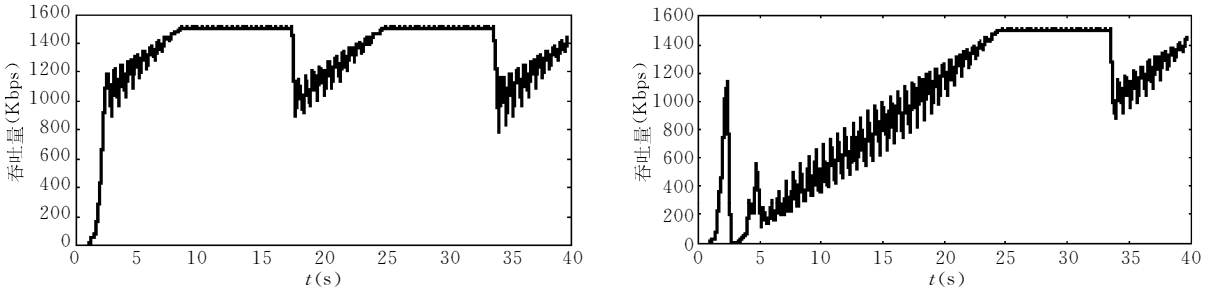
3.1　单连接时 RenoTCP 和 BLBE RenoTCP 的比较

为在单连接下比较 RenoTCP 和 BLBE RenoTCP 的性能,我们先在主机 H1a 和 H2a 之间建立一条 BLBE RenoTCP 连接,并在其上承载 FTP 流量,FTP 流量从 1s 开始,40s 后结束.根据 BLBE-SS 算法测到的 TCP 连接的链路带宽如表 2 所示.然后,我们将连接类型改为 RenoTCP 连接,其他条件不变,BLBE RenoTCP 和 RenoTCP 连接的吞吐量如图 5 所示.

表 2　估计的 B_{eq} 和实际的 B_{eq}

B_{eq} 的估计值	B_{eq} 的实际值	*RTT*	*ssthresh*
1060(Kbps)	1128(Kbps)	149ms	99

在图 4 中,H1a 和 H2a 之间的 TCP 连接通过了 5 条物理链路,其链路带宽分别为 100Mbps,10 Mbps,1.5 Mbps,10 Mbps 和 100 Mbps.由于在这



(a) BLBE RenoTCP 吞吐量 (b) RenoTCP 吞吐量

图 5 RenoTCP 和 BLBE RenoTCP 的性能比较

些链路上只有该 TCP 流,因此,它将占用各物理链路上整个的物理带宽.这样,根据 2.1 节的理论分析,其等效的 TCP 连接带宽应为

$$\frac{1}{1/100+1/10+1/1.5+1/10+1/100}=1.12782\text{Mbps}.$$

然而,在 2.1 节的理论推导中,为了在实际中能对 B_{eq} 进行估计,我们对式(11)进行了相应的近似,因此,使表 2 的估计等效带宽比实际的等效带宽小约 6%左右,这会使 *ssthresh* 偏小(该例中,理想的 *ssthresh* 是 105),其结果是使 TCP 的慢启动阶段的结束时间稍稍提前.这在一定程度上影响了 TCP 在慢启动阶段的吞吐量.然而,同 RenoTCP 相比,BLBE RenoTCP 大大缩短了 TCP 达到稳定状态的时间.并在很大程度上提高了 TCP 慢启动阶段的吞吐量.这一点可以从图 5 中明显看出.

由图5可以看出,虽然BLBE RenoTCP和RenoTCP在稳定后其吞吐量完全相同,但它们达到稳定的过程却存在明显的差别.其主要原因在于 RenoTCP 在慢启动阶段根本不知道其可用带宽是多少,因此,它以指数方式增加其拥塞窗口,并在检查到丢包的情况下将 *ssthresh* 设为当前拥塞窗口的一半.这导致慢启动阶段的多包丢失,使其在随后一段时间内吞吐量为 0.同时,由于其在检查到多包丢失后设置的 *ssthresh* 过小,导致其过早进入 CA 段,这样,RenoTCP 将经过一段很长时间才能达到稳定状态.而 BLBE RenoTCP 却不同,在慢启动阶段,它通过对 TCP 等效带宽的估计来设置 *ssthresh*,使其既不会出现多包丢失也不会过早进入 CA 段.由图 5 可知,RenoTCP 在启动后经过约 25s 才达到稳定,而 BLBE RenoTCP 只需约 8s.同时,在 1～25s 这段时间里,RenoTCP 的平均吞吐量约为 768Kbps,而 BLBE RenoTCP 的平均吞吐量却高达 1316Kbps,提高了大约 72%.

3.2 多连接时 RenoTCP 和 BLBE RenoTCP 的比较

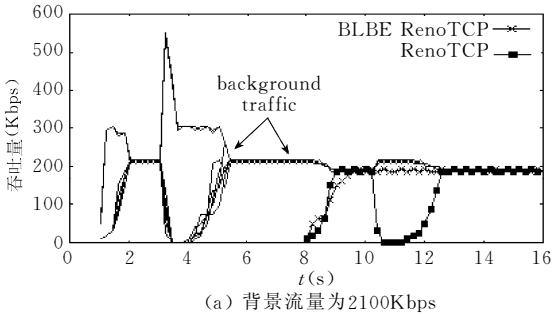
为进一步分析 BLBE RenoTCP 的兼容性和公平性,我们需要在多连接下运行 BLBE RenoTCP,并将其同 Reno TCP 比较.

我们仍采用图 4 所示的仿真拓扑.在 H1a,H2a 之间,我们建立一条 UDP 连接、两条 RenoTCP 连接;在 H1b,H2b 之间,我们建立一条 UDP 连接、两条 RenoTCP 连接;在 H1c,H2c 之间,我们建立一条 RenoTCP 连接.以上连接均承载 CBR 流量.在这样的条件下,我们将在 H1c,H2c 之间分别运行 BLBE RenoTCP 和 RenoTCP,并在其上承载 FTP 流量.FTP 流量从 8s 开始,CBR 流量从 1s 开始.通过控制 CBR 流量的大小,我们来仿真 BLBE RenoTCP 在不同网络负荷下的性能.表 3 为当网络背景流量分别为 280Kbps(相当于网络轻负荷)、800Kbps(相当于网络中等负荷)和 2100Kbps(相当于网络拥塞)时,BLBE RenoTCP 在慢启动阶段估计的 B_{eq},图 6 表示在不同网络负荷下分别运行 BLBE RenoTCP 和 RenoTCP 时各连接的吞吐量.

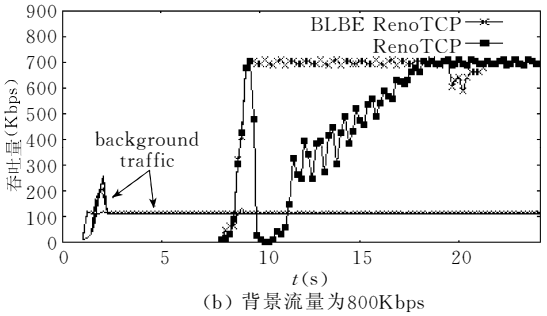
表 3 不同背景流量下估计的 B_{eq} 和实际的 B_{eq}

背景流量(Kbps)	B_{eq} 的估计值(Kbps)	B_{eq} 的实际值(Kbps)	*RTT*(ms)	*ssthresh*
280	1060	1128	149	99
800	693	644	152	66
2100	148	180	151	14

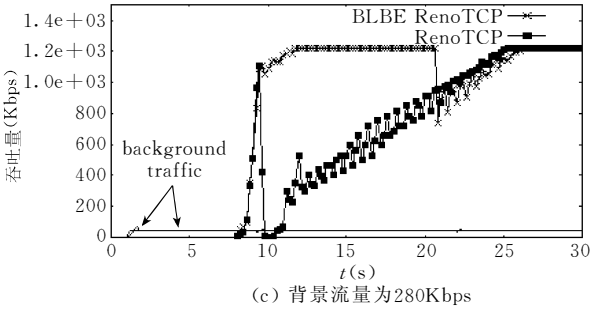
当背景流量为 280Kbps 时,由于网络负荷轻,当属于 BLBE RenoTCP 的包到达图 4 所示的各路由器时,各路由器的队列为空,因此,FQ 队列机制将按其下游链路的实际物理带宽对属于 BLBE RenoTCP 的包进行调度,这样,我们得到同表 2 一样的结果.当背景流量为 2100Kbps 时,由于网络负荷超过了网络的瓶颈带宽(1.5Mbps),网络处于拥塞状态,当属于 BLBE RenoTCP 的包到达图 4 所示的各路由器时,各路由器的队列非空,根据各路由器采用的队列机制和 2.1 节的理论推导可知,BLBE RenoTCP 连接的等效带宽应为



(a) 背景流量为2100Kbps



(b) 背景流量为800Kbps



(c) 背景流量为280Kbps

图 6　多连接、不同网络负荷下 RenoTCP 和 BLBE RenoTCP 的吞吐量

$$\frac{1}{1/100+1/10+1/(1.5/8)+1/10+1/100}=180.072\text{Kbps}.$$

当背景流量为 800Kbps 时,由于网络负荷适中,各路由器中的队列占用将不稳定.当属于 BLBE RenoTCP 的包到达图 4 所示的各路由器时,路由器可能对不同的包按不同的带宽进行调度.这样,会使估计的 B_{eq} 同实际的 B_{eq} 有较大的偏差,如表 3 所示.然而,如图 6 所示,即使在这样的情况下,BLBE RenoTCP 也不会出现慢启动阶段的多包丢失.原因在于只要 *ssthresh* 的值控制在一定的范围内,路由器中的队列便能"吸收"网络中过多的包.BLBE RenoTCP 的慢启动算法能将 *ssthresh* 的值控制在一定范围内.

从图 6 可以看出,当仿真采用多连接时,在三种网络负荷条件下,RenoTCP 都存在慢启动阶段的多包丢弃.相反,BLBE RenoTCP 却表现出很好的性能,避免了慢启动阶段的多包丢弃.同时,BLBE RenoTCP 能更快地到达稳定状态,从而提高慢启动阶段的吞吐量.其达到稳定状态的时间和在这段时间内的平均吞吐量如表 4 所示.同时,从图 6 可以看出,BLBE RenoTCP 跟 UDP 和其他类型的 TCP 连接能很好地兼容.由于 BLBE RenoTCP 只改变了 TCP 慢启动阶段的算法,而在 CA 段,它仍采用快速重传快速恢复算法控制拥塞窗口,因此,在稳定后,它同 RenoTCP 将具有同样的性质,也不会改变原有 TCP 的公平性.

表 4　多连接、不同网络负荷下 BLBE RenoTCP(表中简称 B-TCP)和 RenoTCP(表中简称 R-TCP)在慢启动阶段的性能比较

背景流量(Kbps)	B-TCP 稳定时间(s)	R-TCP 稳定时间(s)	B-TCP 平均吞吐量(Kbps)	R-TCP 平均吞吐量(Kbps)
280	2.2	17.8	1086	648
800	2.0	9.2	642	341
2100	1.6	5.6	166	99

4　结束语

本文介绍了目前在 Internet 上广泛使用的 TCP 的拥塞控制原理,分析了已有的 TCP 慢启动算法的性能,突出其突发丢包的缺点,尤其是在大容量的 TCP 连接链路上.然后,我们通过对 TCP 连接等效带宽的深入分析,提出了一种基于 TCP 连接链路带宽估计的慢启动算法,并对其进行了仿真,我们的结论是:该算法能极大地改进 TCP 慢启动阶段的性能.同时,由于该算法只需对发送方的 TCP 进行简单的修改,因此,它不仅实现简单,而且能跟已有的 TCP 实现兼容,具有明显的有效性和实用性.

参　考　文　献

1　van Jacobson, Karels M J. Congestion avoidance and control. In: Proceedings of the SIGCOMM'88 Symposium, 1988, 18: 314～329

2 Allman *et al*. TCP congestion control. Internet Eningeering Task Force,RFC 2581, 1999

3 Mahdavi J, Mathis M, Podolsky M. An extension to the selective acknowledgement (SACK) option for TCP. Internet Eningeering Task Force, RFC 2883, 2000

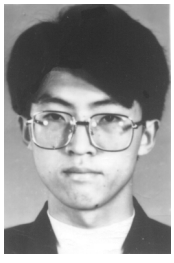
4 Mathis M, Mahadavi J. Forward acknowledgment: Refining TCP congestion control. In: Proceedings of the ACM SIGCOMM'96, Stanford, 1996. 281～291

5 Floyd S, Henderson T. The newreno modification to TCP's fast recovery algorithm. Internet Eningeering Task Force,RFC 2582, 1999

6 Hoe J C. Improving the start-up behavior of a congest- ion control scheme for TCP. In: Proceedings of the ACM SIGCOMM'96, Stanford,1996. 270～280

7 Ewerlid A. Reliable communication over wireless links. In: Proceedings of the Nordic Radio Symposium, Saltsjobaden, Sweden, 2001. 40～45

8 Zhang H. WF^{2}Q: Worst-case fair weighted fair queue. In:Proceedings of IEEE INFOCOM'96, San Francisco, California, 1996. 120～128



LI Yun, born in 1974, Ph. D. candidate. His research interests include TCP/IP improvement and wireless TCP.

CHEN Qian-Bin, born in 1967, Ph. D., professor. His current interests are in personal communications.

LONG Ke-Ping, born in 1968, Ph. D., associate professor. His current interests include TCP/IP improvement, IP QoS and optical burst switch.

WU Shi-Qi, professor and Ph. D. supervisor. His current interests are in wireless communications.