

多重慢启动 TCP 协议研究

王国栋^{1,2}, 任勇毛¹, 李 俊¹

(1 中国科学院 计算机网络信息中心, 北京 100190; 2 中国科学院 研究生院, 北京 100049)

摘 要: 多重慢启动 TCP 协议是为了适应高速网络的发展而提出的一类标准 TCP 协议的改进协议. 本文首先分析了多重慢启动 TCP 协议的原理, 然后详细阐述了近年来提出的一些典型多重慢启动 TCP 协议, 最后对多重慢启动 TCP 协议下一步的工作方向进行了展望.

关键词: 传输控制协议; 多重慢启动; 高速网络; 拥塞控制

中图分类号: TP393

文献标识码: A

文章编号: 1000-7180(2011)03-0178-04

Survey of Multiple Slow Start TCP

WANG Guo-dong^{1,2}, REN Yong-mao¹, LI Jun¹

(1 Computer Network Information Center of Chinese Academy of Sciences, Beijing 100190, China;

2 Graduate University of Chinese Academy of Sciences, Beijing 100049, China)

Abstract: Multiple Slow Start TCP is a set of TCP variants which can fit the High-Speed Network well. Firstly we analyze the theory of Multiple Slow Start TCP, and then we particularize several Multiple Slow Start TCP variants proposed recently. At last we give a perspective of the development of Multiple Slow Start TCP.

Key words: TCP; multiple slow start; high-speed network; congestion control

1 引言

随着网络技术的发展, 高速网络的应用越来越广泛, 高速网络的高带宽—延时积(Bandwidth Delay Product, BDP)特点使得标准 TCP 协议无法满足高速网络的要求^[1-2]. 为了适应高速网络和网络传输性能不断发展的需要, 研究人员提出了一系列基于标准 TCP 的改进协议, 多重慢启动 TCP 就是其中的一类改进协议. 多重慢启动 TCP 协议基于带宽测量技术, 细化了网络特征, 对不同的网络状态采用不同的慢启动方式, 有效减少了丢包重传问题, 不但能有效提高高速网络的性能, 而且也能获得较好的公平性.

2 多重慢启动协议原理

网络测量技术的发展, 尤其是网络可用带宽测量技术的发展, 如 PGM(Packet Gap Model)和

PRM(Packet Rate Model), 为基于可用带宽测量的 TCP 改进协议奠定了基础, 并导致了多重慢启动 TCP 的产生. 多重慢启动 TCP 的基本原理是, 首先通过网络测量技术估测目前网络的可用带宽, 在可用带宽较大的情况下, 采用“霸道”的拥塞窗口增长方式, 尽可能的挖掘网络可用带宽, 提高吞吐率; 在可用带宽较小的情况下, 采用“柔和”的拥塞窗口增长方式, 尽可能控制丢包重传的概率, 通过减少丢包重传来获得比较好的网络性能. 目前出现的典型多重慢启动 TCP 协议主要有 Capstart TCP、Hybrid Slow Start TCP、YeAH-TCP 和 Compound TCP. 表 1 列举了上述几种协议所对应的可用带宽检测方法和在不同的网络状况下所采取的拥塞窗口调整策略.

为了深入理解多重慢启动 TCP 的原理和特点, 下面对以上协议进行了详细的分析.

收稿日期: 2010-03-16; 修回日期: 2010-05-09

基金项目: 国家“八六三”计划项目(2007AA01Z214); 中国科学院研究生科技创新类项目

表 1 各种协议的可用带宽检测方法和拥塞窗口调整策略

协议名称	可用带宽检测方法	网络可用带宽低时采取的措施	网络可用带宽高时采取的措施
Capstart TCP	Packet pair dispersion	Limited Slow Start	Reno Slow Start
Hybrid Slow Start TCP	ACK train length Delay increase	Slow start	Congestion avoidance
YeAH-TCP	瓶颈路径缓存中包数量	Reno TCP	STCP
Compound TCP	Gamma auto-tuning	Delay-based CAA	Drop-based CAA

3 典型多重慢启动 TCP 协议分析

从协议特点、协议所采用的可用带宽测量方法和协议性能等方面分别对 Capstart TCP、Hybrid Slow Start TCP、YeAH-TCP 和 Compound TCP 四种多重慢启动协议进行介绍。

3.1 Capstart TCP

Capstart TCP^[3]提出了网络场景的思想,将网络分为“capacity expansion”和“capacity reduction”两个场景,对这两个场景进行讨论如下,其中 C_{if} 表示网卡传输速率, C_i 表示路由器 i 的传输能力。

如图 1 所示在“capacity expansion”场景下($C_{if} < C_h < C_i$), C_{if} 传输能力最弱,因此整个网络的最大传输速率由 C_{if} 决定,拥塞窗口的最大值 $cwnd_{full} = RTT \times C_{if} / MSS$ 。

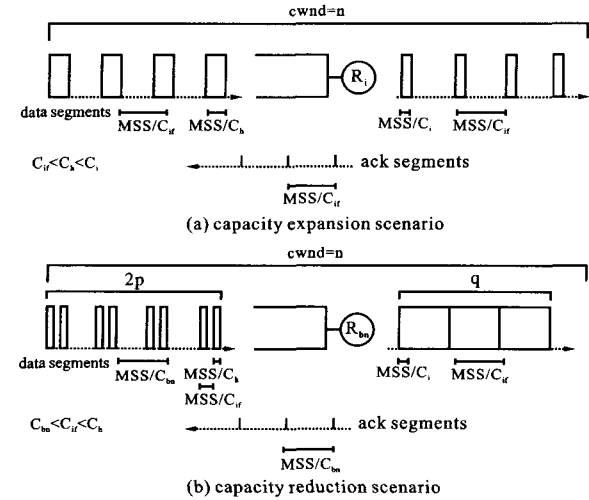


图 1 网络场景分析

在“capacity reduction”场景下, $C_{bn} < C_{if} < C_n$, C_{bn} 传输能力最低,为整个网络的瓶颈,因此假设网卡以速率 C_{if} 产生成对的包,则通过 C_{bn} 的成对包间隔为 MSS/C_{bn} 。每一个 RTT 期间,在路由器 bn 之前有 p 个成对的包,在其后有 q 个包,则 $cwnd = 2p + q$ 。当 $RTT \geq (2p + q)MSS / C_{bn}$ 的情况下,路由器 bn 缓存中的队列能被排空,因此, $cwnd_{drain} = RTT \times$

C_{bn} / MSS 。当 $cwnd$ 大到在 MSS / C_{bn} 时间内将路径填满的情况下,则 $cwnd_{max} = 2RTT \times C_{bn} / MSS$ 。

表 2 给出了 $cwnd$ 和缓存队列的关系,可以看出,对于“capacity reduction”场景,缓慢增加 $cwnd$ 的策略能减少瓶颈路径缓存队列的个数,从而降低拥塞丢包的发生率,使得网络具有更好的吞吐率,这是 Cap start TCP 设计的出发点。

表 2 拥塞窗口与瓶颈路径缓存的关系

cwnd	Backlog(per RTT)
$cwnd \leq \frac{RTT \times C_{bn}}{MSS}$	0
$\frac{RTT \times C_{bn}}{MSS} < cwnd \leq \frac{2RTT \times C_{bn}}{MSS}$	$cwnd - \frac{RTT \times C_{bn}}{MSS}$
$cwnd > \frac{2RTT \times C_{bn}}{MSS}$	$\frac{RTT \times C_{bn}}{MSS}$

为了识别网络目前所处的场景, Capstart TCP 利用 packet pair dispersion^[4]来测量网络当前的可用带宽。packet pair dispersion 的思想是发送成对的长度为 MSS 的包,然后计算这些成对包的 ACK 差($RTT_2 - RTT_1$),如果某个成对包的 RTT_1 和 RTT_2 均最小,则其差值 d 即可反映网络的可用带宽: $C = MSS / d$ 。如果测量的带宽不足网卡速率的 10%,则设定此场景为“capacity reduction”,使用更加温和的 LSS(limited Slow Start^[5])来启动,反之设定场景为“capacity expansion”,使用 SS(Slow Start)来启动。

参考文献[3],在 $RTT = 180ms$,瓶颈路径为 1G—100M 的网络中, CapStart 与 SS、LSS 相比完成传输的时间最短。且 CapStart 对网络带宽测量的结果与网络实际情况相符,受 UDP 背景流量影响较小。

CapStart TCP 存在的主要不足在于当服务器网卡带宽与瓶颈路径带宽相近时,背景 UDP 流量会使得网络真实场景从 expansion 转化为 reduction,而网络测量结果并未受到 UDP 的影响,认为

目前网络场景为 Capacity Expansion, 使用较“霸道”的启动方式导致网络带宽利用率有所下降. 而在其他情况下 (Capacity Expansion 场景和 Capacity Reduction 场景) CapStart 都能获得较好的带宽利用率. 另外, 在带宽更大的如 10G 或 100G 网络中, SS 增加 cwnd 的策略显得过于保守, 可以考虑使用更加“霸道”的 cwnd 增长方式, 以获得更快的启动速度.

3.2 Hybrid Slow Start TCP

Hybrid Slow Start TCP^[6] 考虑到瓶颈路径缓存的增大也会成为网络性能降低的因素. 因为 TCP 慢启动并不慢, 在慢启动之初, 瓶颈路径由于传输能力有限将缓存多余的包, TCP 源端无法感知此状态而以极大的速率传输, 这会导致缓存溢出从而发生丢包重传, 极大的降低高 BDP 网络的性能. Hybrid Slow Start TCP 通过准确把握从慢启动到拥塞避免的“转换点”, 很好的解决了这类问题.

Hybrid Slow Start TCP 利用了网络测量技术中的 packet-pair 测量技术^[3]和 packet-train^[5]测量技术, 通过“ACK train length”和“Delay increase”来决定何时 TCP 从慢启动到拥塞避免阶段的转换.

文献[6]中指出, 当发送端 S 发送 N 个 ($N > 2$) 背靠背, 大小为 L 的测量包时, 称这些测量包为“packet train”. “packet train length”可以表示为 $\Delta(N) = \sum_{k=1}^{N-1} \delta_k$. 其中, N 表示一个 train 中的包数量, δ_k 表示发送端包 K 与包 $K+1$ 之间的时间间隔. “ACK train length”可表示为 $\Lambda(N) = \sum_{k=1}^{N-1} \lambda_k$. 其中, λ_k 是相应的 ACK k 和 $k+1$ 的时间间隔. 由于 $\Lambda(N) \geq \Delta(N)$, 可以通过比较 $\Lambda(N)$ 和 $\Delta(N)$ 的差值来粗略判断网络的可用带宽.

Delay increase 表示发送端在 TCP 的慢启动阶段测量每个发送包的延迟, 当前后两个包的延迟超过一定阈值的时候, 表示瓶颈路径的缓存开始缓存多余的包了.

当“ACK train length”与“packet train length”大于一定阈值时, Hybrid Slow Start TCP 从慢启动转为拥塞避免状态, “Delay increase”大于一定阈值时, 表示网络带宽已经饱和, 可以停止增加 cwnd, 并将此 cwnd 设置为当前的 ssthresh.

通过四种不同的实验环境, 文献[6]从各个角度评价了 Hybrid Slow Start TCP 的性能: ① 在带宽利用率上, Hybrid Slow Start TCP 保持了较高和较稳定的带宽利用率, 没有发生突发传输所导致的丢

包重传现象. ② 在引入背景流量的情况下, 通过不同的 ACK 策略 (Quick ACK、Quick ACK + Delayed ACK、Delayed ACK) 来验证 Hybrid Slow Start TCP 对网络可用带宽的测量. 实验表明, Hybrid Slow Start TCP 能处理这三种不同的 ACK 并能成功检测到瓶颈路径拥塞的产生. ③ 在第 10 秒和第 40 秒分别加入两个 CUBIC 流, 并在改变瓶颈路径带宽、RTT 和瓶颈路径缓存大小的情况下, Hybrid Slow Start TCP 依然保持了较高的带宽利用率, 这是 SS 和 LSS 无法做到的. ④ 在接收端分别使用 Linux、FreeBSD 和 Windows 系统进行测试, Hybrid Slow Start TCP 均获得了较好的性能.

Hybrid Slow Start TCP 通过粗略估计网络状态, 提前预测慢启动阶段与拥塞避免阶段的转换点, 而不是等到发生丢包之后才采取补救措施, 提高了网络的性能. 但是 Hybrid Slow Start TCP 未考虑背景流量对协议的影响, 基于“ACK train length”与“Delay increase”预测容易受到背景流量的影响, 比如突发 UDP 流会导致其对网络状态判断错误从而影响性能.

3.3 YeAH-TCP

YeAH-TCP^[7] (Yet Another High-speed TCP) 考虑到拥塞的发生是导致高 BDP 网络性能下降的主要因素^[8], 为了减少网络拥塞的产生, YeAH-TCP 将网络划分为“Fast”和“slow”两个状态. 在“Fast”状态下, 拥塞窗口使用更加迅速的方式增加 (STCP), 以充分利用网络带宽资源, 在“slow”状态下, 拥塞窗口使用较温和的方式增加 (Reno TCP), 以避免网络拥塞的产生.

YeAH-TCP 对网络状态的感知是通过瓶颈路径所缓存的包数量来获得的, 具体方法如下.

RTT_{base} 表示发送端测量的最小的 RTT, RTT_{min} 表示在 cwnd 数据窗口中估计的 RTT 的最小值, 随着数据的发送进行更新. 当前队列延迟为 $RTT_{queue} = RTT_{min} - RTT_{base}$, 通过 RTT_{queue} 可以知道数据流中包排队的数量为: $Q = RTT_{queue} \times G = RTT_{queue} \times \left(\frac{cwnd}{RTT_{min}} \right)$. 其中, G 为吞吐率, 网络中的拥塞状况可以通过 $L = RTT_{queue} / RTT_{base}$ 来表示.

如果 $Q < Q_{max}$ 并且 $L < 1/\phi$, 则当前的网络状态为“Fast”, 反着则为“Slow”状态. 其中 Q_{max} 表示缓存中允许单流的最大包, $1/\phi$ 表示缓存拥塞允许的最大值.

通过改变 RTT、丢包率和瓶颈路径带宽, 与当

前流行的 TCP 改进协议进行比较试验可知, YeAH-TCP 在不同 RTT 下均可以有效降低瓶颈路径的缓存, 降低了由于缓存溢出而导致的丢包次数, 从而减少了网络拥塞的产生; 在链路丢包率较大的情景下, 也具有较好的性能, 并且网络的公平性也得到了较好的保障. 协议同样未考虑突发 UDP 流对自身性能的影响, 在突发 UDP 流出现时, 瓶颈路径的缓存会被迅速填满, YeAH-TCP 可能避免不了产生丢包, 相应的会降低网络性能.

3.4 Compound TCP

CTCP^[9] (Compound TCP) 将基于丢包的 CAA (Congestion Avoidance Algorithm) 和基于时延的 CAA 相结合, 在保证了较高的带宽利用率的情况下, 又获得了较好的公平性. 基于时延的 CAA 是通过引入 dwnd(delay window) 来实现的, dwnd 用来控制 CTCP 中基于时延的 CAA 组件. 在引入 dwnd 之后, 协议中的发送窗口变为 $\text{win} = \min(\text{cwnd} + \text{dwnd}, \text{awnd})$, 其中 awnd 是来自接收端的广播窗口. 相应的在拥塞避免阶段, 每收到一个 ACK, 窗口大小调整为 $\text{cwnd} = \text{cwnd} + 1/\text{win}$, 而在慢启动阶段, CTCP 保留了 Reno TCP 的慢启动策略.

基于时延的拥塞避免与 TCP Vegas 类似, 用 baseRTT 表示网络传输的基本延时, 通过最小的 RTT 获得. 网络中缓存的包数量可以表示为 $\text{Diff} = (\text{Expected} - \text{Actual}) \times \text{baseRTT}$, 其中 $\text{Expected} = \text{win}/\text{baseRTT}$, $\text{Actual} = \text{win}/\text{RTT}$. 当 $\text{Diff} < \gamma$ 时, 网络为低利用率状态, 使用基于丢包的 CAA, 反之网络为繁忙状态, 使用基于延时的 CAA.

当没有拥塞产生时, $\text{win}(t+1) = \text{win}(t) + \alpha \cdot \text{win}(t)^k$, 当发生丢包时, $\text{win}(t+1) = \text{win}(t) \cdot (1 - \beta)$. cwnd 调整遵循传统的 TCP 策略, dwnd 的调整使用如下策略:

$$\text{dwnd}(t+1) =$$

$$\begin{cases} \text{dwnd}(t) + (\alpha \cdot \text{win}(t)^k - 1)^+, & \text{if } \text{diff} < \gamma \\ (\text{dwnd}(t) - \zeta \cdot \text{diff})^+, & \text{if } \text{diff} \geq \gamma \\ (\text{win}(t) \cdot (1 - \beta) - \text{cwnd}/2)^+, & \text{if loss is detected} \end{cases}$$

其中, $(\cdot)^+$ 表示 $\max(\cdot, 0)$, ζ 表示当检测到拥塞发生时时延组件以多快的速度减小窗口. CTCP 中各参数设定的依据是使 CTCP 与 HSTCP 具有相同的快速启动效率^[9], 据此, 各参数如下 $k=0.8$, $\alpha=1/8$, $\beta=1/2$, γ 的取值根据 gamma auto-tuning 方法动态获得.

CTCP 动态调整 γ 的方法称为 gamma auto-tuning, 如图 2 所示. 假设瓶颈路径的缓存大小为 B ,

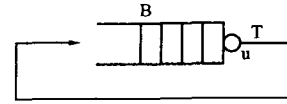


图 2 gamma auto-tuning 原理图

其中缓存包数量为 m , 以下两种情况下 CTCP 基于时延的组件无效: 瓶颈路径缓存包数量小于阈值 γ , 即 $\gamma > B/m$ 或者预期窗口大于阈值窗口 W_{low} , 即 $\omega_{\text{low}} < \frac{B + uT}{m}$, 其中 uT 为链路中的 BDP. 假设 B 是整个链路 BDP 的一部分, 即 $B = kuT$, 则合并以上两种情况得 $\gamma > W_{\text{low}} \cdot \frac{k}{k+1}$, 因此为了避免这两种

情况发生, γ 的取值为 $\gamma = \max(\gamma_{\text{min}}, W_{\text{low}} \cdot \frac{k}{k+1})$,

其中 $k = \frac{R_{\text{max}} - R_{\text{min}}}{R_{\text{min}}}$, R_{max} 和 R_{min} 分别表示目前观察到的最大和最小 RTT.

文献[9]通过仿真得出在网络吞吐率方面 CTCP 较 Reno TCP 有 28% 到 52% 的提升. 在公平性方面, 将多个 CTCP 流逐个加入到 TCP 流中, γ 始终保持着一个相对较小的值, 对原 TCP 流的影响较小, 保持了较好的 TCP 公平性. 主要缺点是其保留的 Reno TCP 慢启动策略会成为网络性能的障碍^[7], 另外协议也没考虑背景流量对性能的影响.

4 结束语

多重慢启动 TCP 是可用带宽测量技术和高速传输协议相结合的产物, 通过细化拥塞窗口调整策略, 多重慢启动获得了比较好的性能.

可用带宽测量技术是多重慢启动 TCP 的基础, 只有准确获得目前网络可用带宽才能决定使用哪种启动方式, 在保证传输效率的同时又避免拥塞导致的丢包重传, 达到较好的网络性能; 因此在代价允许的情况下, 使用性能优异的带宽测量技术是多重慢启动 TCP 研究的方向. 当网络可用带宽较大时, 考虑采用更加“霸道”的 cwnd 增长方式会获得更好的传输效率, 这在多重慢启动 TCP 协议的改进上也是值得研究的. 同时, 文中提到的多重慢启动仅停留在二重上, 根据网络特性采用更加精细的 cwnd 调整策略, 比如三重或四重的慢启动方式, 对网络带宽的挖掘会更加有效; 不过高吞吐率和公平性是一对矛盾体, 在充分挖掘网络可用带宽的同时会降低 TCP 公平性. 如何在满足一定吞吐率的同时保证网络的公平性也是多重慢启动需要解决的问题.

(下转第 185 页)

后,用写存储器指令写 BCR,启动存储器 BIST.再用读存储器指令读 BQR 查询结果.最后,根据 BIST 的完成情况,软件进行分支处理.如果存储器 BIST 成功完成,则马上转入正常程序运行;否则,无论对于处理器自身还是处理器所在的应用系统,都可以考虑使用中断方式进行后处理.

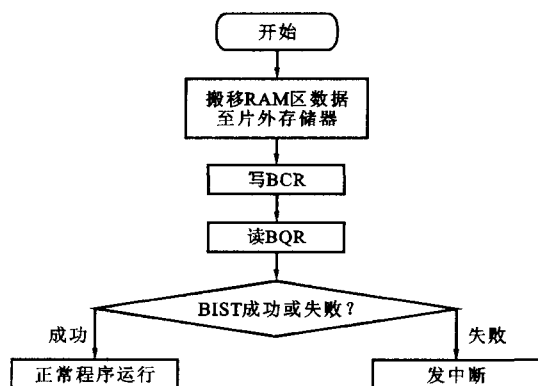


图2 程序控制存储器 BIST 的软件流程

4 结束语

文中程序控制存储器 BIST 的方法,具有以下优点:①整个处理所用到的指令具有通用性,易于扩展;②关键处理(如保持处理的原子性)对用户透明,易于使用;③用户可以根据需要,在应用系统运行的任何时候,通过调用特定的程序段,启动存储器 BIST 并查询结果,可用于冗余系统中的实时在线

检测;④在程序控制的存储器 BIST 工作过程中,处理器还可以访问片外存储器,进行与片上存储器不相关的一些处理工作,从而进一步提高了芯片特殊情况下的可用性.这种方法已经在某款 DSP 芯片上成功应用,有效地提高了芯片的可测试性和应用系统的容错性.

参考文献:

- [1] Michael L. Bushnell, Vishwani D. Agrawal. Essentials of electronic testing for digital, memory & mixed-signal VLSI circuits[M]. 北京:电子工业出版社,2005.
- [2] Agrawal V D, Kime C R, Saluja K K. A tutorial on built-in self-test, part1:principles[J]. Design & Test of Computers, IEEE, 1993,10(1):73-82.
- [3] Nicolaidis M. Transparent BIST for RAMs[C]// Proceed of the international test Conf. USA: IEEE, 1992: 598-607.
- [4] IEEE std 1364-2001. IEEE standard verilog hardware description language[S]. NY: Design Automation Standards Committee of the IEEE Computer Society, 2001.

作者简介:

刘蓬侠 男,(1971-),博士,副研究员.研究方向为高性能微处理器和高性能数字信号处理器的研究与设计,重点研究可测试性和可调试性.

鲁建壮 男,(1977-),博士,副研究员.研究方向为处理器体系结构、微体系结构及存储系统.

(上接第 181 页)

参考文献:

- [1] 王强,普杰信,刘伟. TCP 拥塞控制慢启动策略的研究[J]. 微电子学与计算机,2007,24(12):210-212.
- [2] 钟伯成,张杰. IP 网络中 TCP 拥塞控制协议的稳定性分析[J]. 微电子学与计算机,2005,22(12):177-178.
- [3] Dirceu Cavendish, Kazumi Kumazoe, Masato Tsuru. An adaptive TCP slow start for high speed networks [C]//Proceedings of First International Conference on Evolving Internet. Cannes: PFLDnet, 2009(8):15-20.
- [4] Kapoor R, Chan J J, Chen Y C. CapProbe: A simple and accurate capacity estimation technique[C]// Proceedings of Sigcomm 04, Portland, Oregon, 2004(9):67-78.
- [5] Floyd S. Limited Slow Start for TCP with Large Congestion Windows. RFC 3742 [S]. The Internet Society, 2004.
- [6] Sangtae Ha, Injong Rhee. Hybrid slow start for high-bandwidth and long-distance networks [C]// North

Carolina State: PFLDnet, 2007.

- [7] Andrea Baiocchi, Angelo P. Castellani, Francesco Vacirca. YeAH-TCP: Yet Another Highspeed TCP[C]. Italy: PFLDnet, 2007.
- [8] Makoto Nakamura, Mary Inaba, Kei Hiraki. Fast ethernet is sometimes faster than gigabit ethernet on LFN [C]. Tokyo: PDCS, 2003.
- [9] Kun Tan, Qian Zhang, Murari Sridharan. Compound TCP: a scalable and TCP-friendly congestion control for high-speed networks[C]. Beijing: PFLDnet, 2006.

作者简介:

王国栋 男,(1981-),博士研究生.研究方向为高速网络,传输协议.

任勇毛 男,(1981-),博士,助理研究员.研究方向为高速网络,传输协议.

李俊 男,(1968-),博士,研究员,博士生导师.研究方向为下一代互联网、高速网络.