

# COS-Slow-Start: 一种新的 TCP 慢启动策略

茹新宇, 刘 渊

(江南大学信息工程学院, 无锡 214122)

**摘 要:** 拥塞控制已成为确保 Internet 稳定性、鲁棒性的关键因素。针对目前 TCP 拥塞控制机制的慢启动算法中存在的实际问题, 提出一种新的 TCP 慢启动策略 COS-Slow-Start, 从数学角度对新策略的稳定性和高效性进行理论与证明。NS2 仿真实验表明, 该策略能有效地减少分组丢失、平缓突发流量冲击, 并增加带宽的有效利用率。

**关键词:** TCP 拥塞控制; 慢启动; NS2 仿真

## COS-Slow-Start: A New Strategy of TCP Slow-start

RU Xin-yu, LIU Yuan

(School of Information Engineering, Southern Yangtze University, Wuxi 214122)

**【Abstract】** The congestion control is a most important protocol, which improves the Internet's stability and robustness. This paper investigates the current standard slow-start algorithm of TCP congestion control mechanism and its actual problem. A new variant slow-start scheme called "COS-Slow-Start" is presented. Mathematics theory analyses and proves that this new scheme can obviously improve the stability and efficiency of network. NS2 simulation results show that it can significantly reduce both packet losses and traffic burstness, and increase the bandwidth utilization ratio.

**【Key words】** TCP congestion control; slow-start; NS2 simulation

Internet 在过去几十年里经历了爆炸式增长。但有限的资源容量和处理能力使得拥塞日益严重。1986 年 10 月, 因拥塞崩溃的发生, LBL 到 UC Berkeley 的吞吐量从 32 Kb/s 跌至 40 b/s<sup>[1]</sup>。自此人们对拥塞控制展开了大量研究, 先后提出了多种算法。最初由 V. Jacobson 提出的 TCP Tahoe 采用了慢启动和拥塞避免机制。TCP Reno 在此基础上增加了快速重传和快速恢复。而 NewReno 则是对 Reno 中快速恢复的有效补充。SACK 使用选择性重复策略, 通过应答返回给发送端完整的信息来确定分组丢失, 这在一定程度上解决了多个分组丢失的问题。以上算法更多考虑了快速重传和恢复机制, 而慢启动过程并没有明显区别。由于慢启动在连接建立和超时重传阶段都用到, 因此策略好坏对算法的性能有很大影响。TCP Vegas 虽然在分组丢失前通过路由检测来限制窗口指数增长, 但仍无法避免多个分组丢失, 并降低了网络性能。

### 1 现有的慢启动策略

据统计, Internet 中 95% 的数据流使用 TCP 协议, 而由于缓存溢出网关会丢弃约 10% 的包<sup>[2]</sup>。

现有的拥塞控制采用 AIMD 算法, 它可以分为以下 4 个步骤:

- (1) 当  $cwnd$  小于  $ssthresh$  时, 采用慢启动机制来获取网络可用带宽。收到每个应答包后,  $cwnd = cwnd + 1$ ;
- (2) 当  $cwnd$  大于  $ssthresh$  时, 进入拥塞避免状态, 并重新探测可用带宽。收到每个应答包后,  $cwnd = cwnd + 1/cwnd$ ;
- (3) 当收到 3 个相同 ACK 时, 重发 ACK 指示的数据包, 并对  $cwnd$  和  $ssthresh$  重新赋值,  $ssthresh = cwnd/2$ ;
- (4) 当重传定时器 RTO 超时, 则再次进入慢启动阶段。

现有拥塞控制机制的有限状态自动机如图 1 所示, 其中 MSS 为最大分组长度。

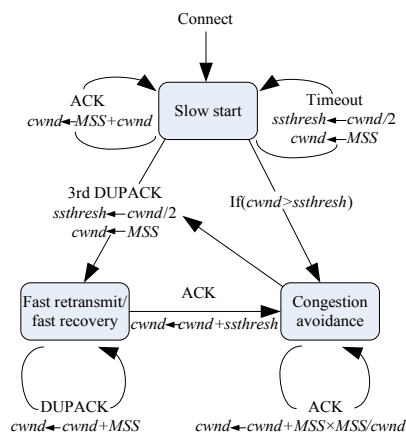


图 1 现有拥塞控制机制的有限状态自动机 FSM

为了保证 Internet 的稳定性, TCP 采用了较保守的 AIMD 算法。这种基于窗口的端到端慢启动策略对大批量文件传输等尽量做好型服务具有较好的适应性, 但在现代网络高带宽低延迟的环境下, 它已被证明是低效的。有个典型例子<sup>[3]</sup>: 一条 7.2 Gb/s 的链路, 设 RTT 为 100 ms, TCP 包大小 1 500 B, 发送窗口峰值可达 80 000。当一次分组丢失后, 发送窗口减半, 那么发送端需要 40 000 个 RTT 时间来恢复到它丢包前的发送速率, 约需要 70 min。这意味着链路将在相当长的一段时间得不到充分利用。因此速率增长过慢减少过快是 AIMD 的主要问题。

**基金项目:** 国家部委预研基金资助项目

**作者简介:** 茹新宇(1977 -), 男, 硕士研究生, 主研方向: 网络拥塞控制, 网络安全; 刘 渊, 副教授

**收稿日期:** 2007-03-15 **E-mail:** ruxinyu21@163.com

目前,随着 Web 业务的迅猛增长,主要表现为短生存期连接的 Web 流已占据了 Internet 总量的 70%以上。其特点是突发性强,连接时间短,平均窗口小。由于它大部分时间都处于慢启动阶段,因此对连续丢包更敏感,甚至连续 1,2 个丢包就会大大抑制发送速率,而高突发性更增加了其竞争带宽的劣势。针对以上问题,文献[4]推荐将拥塞窗口初值增加为 4 个分组,而研究人员也尝试性地提出了一些新的改进算法,典型的有 Scalable-TCP, HighSpeed-TCP 和 FAST-TCP 等。

## 2 COS-Slow-Start 策略

为了有效地提高短生存期连接的传送效率并改善长生存期连接的启动和丢包重发过程,本文提出了一种基于余弦函数的新慢启动策略: COS-Slow-Start。定义拥塞窗口  $cwnd$  从 1 个分组开始,以余弦函数的增长方式来逐步执行慢启动过程,如图 2 所示。

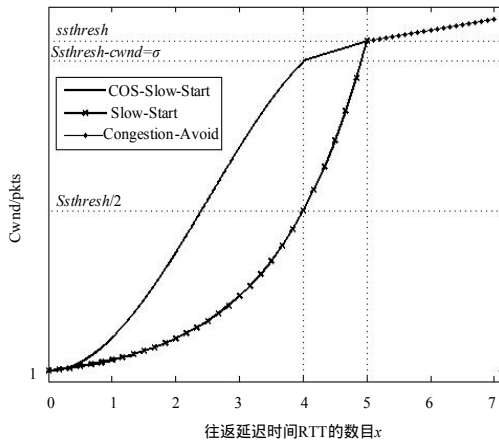


图 2 COS-Slow-Start 与标准 Slow-Start 的比较

为了便于分析,以  $ssthresh/2$  为界,把慢启动过程分成前后 2 个半程,并假设慢启动阶段所有分组在往返延迟时间 RTT 内都正确返回应答报文 ACK,所有 RTT 均相等。由图 2 可看出,由于原 TCP 慢启动机制采用 AIMD 算法,拥塞窗口  $cwnd$  从 1 开始以 2 的指数幂增长方式探测网络可用带宽。 $cwnd$  到达  $ssthresh$  将耗费多个 RTT,使得发送窗口远小于路径带宽延迟乘积,这样短生存期连接的可用带宽利用率较低。同时,由于拥塞窗口按指数增长,后半程( $cwnd > ssthresh/2$ )递增太快,这在一定程度上增加了丢包风险。而瞬间发送过多的分组,往往导致瓶颈链路阻塞严重,引发多个分组丢弃。接着超时重传的拥塞退避机制可能导致网络的全局同步,引起整个网络的流量负载和排队延迟抖动,使得 TCP 性能大幅下降。而采用了新的慢启动算法后,前半程( $cwnd < ssthresh/2$ )增大发送分组量,以较快地获得网络可用带宽,提高了网络的利用率。后半程逐步减小窗口增长速度,平滑从慢启动阶段过渡到拥塞避免阶段,从而达到既减少了慢启动前期所经历的时间,又降低了后期突发流量对网络的冲击,有效地改善了网络稳定性,提高了带宽利用率。

用数学表达式描述如下:

$$\begin{cases} cwnd = \lfloor -(ssthresh/2)\cos(\pi x / lbssthresh) + ssthresh/2 + 1 \rfloor \\ x \in [0, lbssthresh/2] \\ cwnd = \lfloor -(ssthresh/2)\cos(\pi x / lbssthresh) + ssthresh/2 + 1 \rfloor \\ x \in (lbssthresh/2, lbssthresh] \end{cases}$$

其中,拥塞窗口  $cwnd$ ;门限阈值  $ssthresh$ ;  $x$  为往返延迟时间 RTT 的数目。

因为  $cwnd$  是基于分组发送的,只是按字节计数而已,而每个点的余弦函数值并非整数,所以引入取整符号来修饰  $cwnd$  值。为了避免慢启动初期  $cwnd$  开得太大会引起不必要的分组丢失重传,前半程  $cwnd$  向下取整。同时,为了平滑稳定地进入到拥塞避免阶段,不致于引起过多的流量颠簸,后半程  $cwnd$  向上取整。而由于  $\cos(\pi/2) = 0$ ,即  $cwnd$  上升曲线最高点斜率为 0,此时拥塞窗口将不再增长。为了更好地与原标准 TCP 拥塞避免阶段衔接,特规定:当 (1)  $ssthresh - cwnd > \sigma$  时,采用上述所提出的改进后的慢启动算法;(2)  $ssthresh - cwnd \leq \sigma$  时(见图 2),采用原拥塞避免算法,  $cwnd = cwnd + 1/cwnd$ 。

上式中  $\sigma$  为调节因子。理论上  $\sigma$  取值越小,  $cwnd$  上升曲线由慢启动过渡到拥塞避免阶段越平稳。但  $\sigma$  过小会使得  $cwnd$  上升曲线在慢启动阶段末端过于平缓(因  $cwnd$  曲线斜率为 0),反而影响到向拥塞避免阶段的平稳过渡。经实验取  $\sigma = 2$ 。

## 3 性能分析与效率证明

### 3.1 性能分析

为简便起见,定义图 3 中  $a, b$  分别代表新旧 2 种策略,曲线在顶点处重合,且整个慢启动阶段门限阈值  $ssthresh$  保持不变。并令  $w = cwnd$ ,  $A = ssthresh/2$ ,  $B = \pi/lbssthresh$ 。

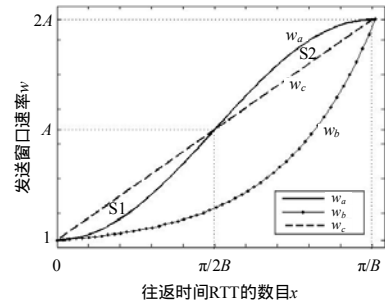


图 3 新旧策略慢启动窗口速率对比

如图 3 所示,新旧 2 种慢启动函数可分别写成

$$w_a = -A \cdot \cos(Bx) + A + 1, \quad w_b = 2^x$$

则它们的曲线斜率分别为

$$w'_a = AB \cdot \sin(Bx), \quad w'_b = \ln 2 \cdot 2^x$$

其中,  $x \in [0, \pi/B]$ 。

由图 4 可看出,  $a$  与  $b$  相比,前半程,由于  $w'_a$  上升较快,能够快速地探测网络可用带宽,因此新策略效率明显高于旧策略。后半程,由于  $w'_a$  稳定地下降,从而限制了突发流量的涌入,因此新策略比旧策略稳定性更好。

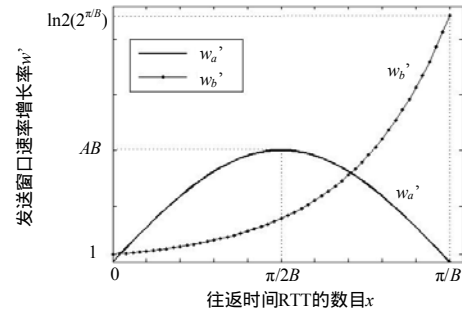


图 4 慢启动窗口速率增长率对比

### 3.2 效率证明

(1)在证明网络效率之前先介绍一个有用的定理:连续函数  $H(x)$  具有连续的一阶、二阶导数,并且  $H(x_0) = H(x_N) = 0$ 。

如果对于任意  $x \in [x_0, x_N]$  , 它的二阶导数恒小于 0 , 那么必有  $H(x) = 0$  。

证明 反证法。

(1)假设存在一个  $\delta \in [x_0, x_N]$  , 使得  $H(\delta) < 0$  。由于它是连续函数, 根据中值定理, 则必定存在  $\delta_1 \in [x_0, \delta]$  ,  $\delta_2 \in [\delta, x_N]$  , 使得

$$H'_{x^2}(\delta_1) = \frac{H(\delta) - H(x_0)}{\delta - x_0} < 0$$

然而

$$H'_{x^2}(\delta_2) = \frac{H(x_N) - H(\delta)}{x_N - \delta} > 0$$

因为  $H(x)$  具有一阶连续性, 再次由中值定理可知, 必定存在  $\delta_3 \in [\delta_1, \delta_2]$  , 使得

$$H''_{x^2}(\delta_3) = \frac{H'_{x^2}(\delta_2) - H'_{x^2}(\delta_1)}{\delta_2 - \delta_1} > 0$$

这与二阶导数恒小于零相矛盾, 因此假设不成立。所以, 对于任意  $x \in [x_0, x_N]$  , 当二阶导数恒小于零时, 必有  $H(x) = 0$  。

(2)定义慢启动的网络效率为: 在整个慢启动阶段内, 拥塞窗口  $w$  的总发送量与门限阈值( $ssthresh$ )所允许的发送量之比。用  $\eta$  表示, 即

$$\eta = \frac{\sum_{i=0}^{N-1} w(x_i)}{N \cdot ssthresh}$$

其中, 分母  $N \cdot ssthresh$  在整个慢启动阶段不变, 所以只需要比较分子即可。

由于余弦函数曲线在半周期内凹凸性不一致, 具有拐点, 不便于分析, 因此引入等效函数:

$$w_c(x) = 2AB/\pi \cdot x + 1, x \in [0, \pi/B]$$

1)证明:  $\eta_c = \eta_a$  (即  $c$  与  $a$  等效)

因为  $\int_0^{\pi/B} (2AB/\pi \cdot x + 1) \cdot dx = \int_0^{\pi/B} (-A \cos(Bx) + A + 1) \cdot dx$  成立, 即  $\int_0^{\pi/B} w_c \cdot dx = \int_0^{\pi/B} w_a \cdot dx$  。所以在  $[0, \pi/B]$  时间内, 2 种策略  $w_c$  与  $w_a$  具有相同的发送量。同时, 由图 3 可知:

$$\left. \begin{array}{l} x < \pi/2B \quad w_c > w_a \\ x = \pi/2B \quad w_c = w_a \\ x > \pi/2B \quad w_c < w_a \end{array} \right\} \Rightarrow S_1 = S_2$$

所以曲线  $w_a$  以  $\pi/2B$  为 midpoint, 在直线  $w_c$  上下呈对称分布。所以

$$\sum_{i=0}^{N-1} w_c(x_i) = \sum_{i=0}^{N-1} w_a(x_i), \text{ 则 } \eta_c = \eta_a \text{ 得证。}$$

2)证明:  $\eta_a > \eta_b$  (即  $a$  比  $b$  效率高)

首先构造函数  $H(x_i) = w_c(x_i) - w_b(x_i)$  , 则

$$H''_{x^2}(x_i) = w''_{cx^2}(x_i) - w''_{bx^2}(x_i)$$

因为  $w_c(x) = 2AB/\pi \cdot x + 1$  , 所以  $w''_{cx^2}(x_i) = 0$  。而  $w_b(x) = 2^x$  , 所以  $w''_{bx^2}(x_i) = \ln 2 \cdot 2^x > 0$  ,  $H''_{x^2}(x_i) < 0$  。

同时, 由图 3 可以看出慢启动窗口曲线  $c$  与  $b$  的递增收敛时间相同。即

$$w_c(x_0) = w_b(x_0) = 0, w_c(x_N) = w_b(x_N) = 2A$$

所以  $w_c(x_0) - w_b(x_0) = w_c(x_N) - w_b(x_N) = 0$  , 则  $H(x_0) = H(x_N) = 0$  。

而  $H(x)$  自身是连续函数, 且具有连续的一阶、二阶导数。

所以由上述定理可知  $H(x_i) > 0$  , 即  $w_c(x_i) > w_b(x_i)$  , 则

$$\sum_{i=0}^{N-1} w_c(x_i) > \sum_{i=0}^{N-1} w_b(x_i)$$

所以  $\eta_c > \eta_b$  , 即  $\eta_a > \eta_b$  得证。

综上所述, 当不同的慢启动策略具有相同的窗口递增收敛时间时, 网络效率依赖于拥塞窗口对时间的二阶导数。该值越小, 网络效率反而越高。所以, 在整个慢启动阶段, 新

策略  $a$  的网络效率要高于原有旧策略  $b$ 。由图 3 曲线  $w_a$  与  $w_b$  分别对  $x$  的积分面积就可明显得到  $\eta_a > \eta_b$  的结论。

#### 4 仿真与验证

为了验证新策略的有效性, 笔者采用了 U.C.Berkely 大学 LNBL 研究小组开发的网络模拟器 NS2.28 作为仿真工具, 在 Linux 9.03 操作系统下, 对图 5 所示的拓扑结构进行场景模拟。其中  $S_1$  至  $S_n$  为发送端,  $R_1$  至  $R_n$  为接收端, 它们分别到路由器  $R_1, R_2$  的链路带宽均设为 10 Mb/s, 延迟为 2 ms。瓶颈链路出现在  $R_1$  与  $R_2$  之间, 带宽设为 1.5 Mb/s, 传送延迟为 50 ms,  $ssthresh$  取接近于网络实际带宽的近似值。为简化分析, 设路由器缓存为 20 个分组大小, 发送端以每个分组 1Kb 大小, 连续发送 1Mb FTP 单向数据流。笔者采用增加了 COS-Slow-Start 后的 TCP-Reno 算法与原算法进行比较, 路由队列管理采用 FIFO 和 Drop-Tail 策略, 进行了多组实验。随着发送端联入链路的连接数逐渐上升, 新算法的分组丢失数较少, 并增长缓慢(见图 6), 优于原算法。

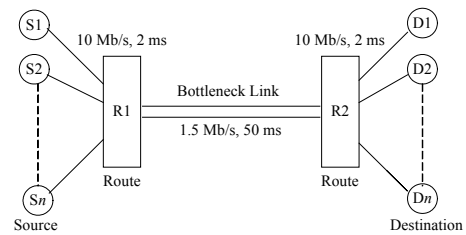


图 5 网络仿真拓扑结构

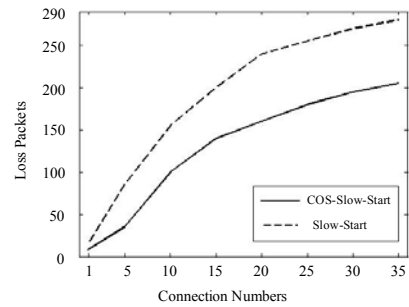


图 6 不同连接数下分组丢失数的比较

由于在新算法中, 慢启动阶段的后半程( $cwnd > ssthresh/2$ )曲线上升较为缓慢, 使突发流量平缓地进入网络, 从而避免了不必要的超时重传, 使得有效传送时间(发送时间与传输时间之和)和分组丢失数都优于原算法。为了比较改进前后的 2 种拥塞控制机制对队列瞬间长度的影响, 采用了 RED 队列管理算法进行对比实验, 发现新算法的路由瞬间队列长度小于原算法(见图 7), 可见新的慢启动策略还有利于改善路由器的队列管理性能。

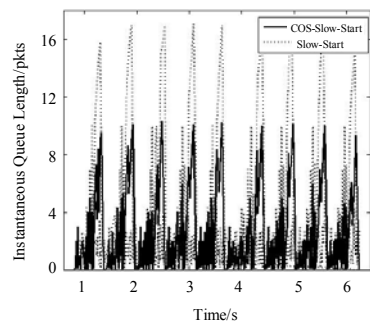


图 7 慢启动瞬间队列长度比较

(下转第 130 页)