一种基于动态阈值的 TCP 超时重传策略

闫二辉[1], 朱 敏[1], 毛明敏[2]
(1. 四川大学 计算机学院, 成都 610064; 2. 西南石油大学 经济管理学院, 成都 610500)

摘 要: 针对传统的 TCP 拥塞控制算法在发生超时后存在恢复时间长、收敛性差、网络抖动剧烈等问题,在超时重传策略的基础上提出一种基于动态阈值的超时重传算法。该算法不仅使得网络拥塞处理更为平滑,同时可显著提高数据传输效率。实验表明,新策略能明显降低网络振荡,提高网络吞吐量,有利于网络资源的利用。

关键词: 传输控制协议; 拥塞控制; 超时重传; 网络振荡; 动态阈值
中图分类号: TP393 **文献标志码**: A **文章编号**: 1001-3695(2010)08-3135-03
doi:10.3969/j.issn.1001-3695.2010.08.087

Timeout retransmission strategy based on dynamic threshold

YAN Er-hui[1], ZHU Min[1], MAO Ming-min[2]
(1. *College of Computer Science, Sichuan University, Chengdu* 610064, *China*; 2. *College of Economics & Management, Southwest Petroleum University, Chengdu* 610500, *China*)

Abstract: When the timeout retransmission occurs, the traditional TCP congestion control algorithm cause some problems, such as long recovery time, poor convergence, severe network vibration. The paper put forward a timeout retransmission strategy based on the dynamic threshold, it avoided network congestion smoothly, and improved the efficiency of the data transmission overall. Experiments show that this new strategy can reduce the vibration of network obviously, improve the output of network and make the network resources more useful.

Key words: TCP(transmission control protocol); congestion control; retransmission timeout; network vibration; dynamic threshold

0 引言

TCP 一直是研究的热点[1,2]。随着 Internet 的迅猛发展,人们对 TCP 拥塞控制的研究越来越重视[3],已有多篇文献先后提出一系列 TCP 拥塞控制版本,如 TCP Tahoe、TCP Reno、TCP New-Reno、TCP Sack、TCP Vegas,但以上算法更多地考虑了慢启动、快速重传和恢复机制[4],而超时重传机制并没有明显区别。最近的研究表明,83% ~95% 的 IP 流量是由 TCP 控制的,而 13% 的 TCP 包需要重传[5],其中由超时所触发的重传占到一半以上,因而其策略的好坏对拥塞控制机制至关重要。

本文提出了一种基于动态阈值的超时重传策略,当发生超时,根据当前的网络状态来动态调整拥塞窗口的大小,以最小的代价从拥塞中恢复过来。最后通过 NS2 仿真得出结论,表明此策略的有效性。

1 TCP 拥塞控制

TCP 拥塞控制主要是通过一些重要参数的改变来达到降低源端数据发送率来实现的[6]。常用的控制参数包括:

a)拥塞窗口(cwnd),表示源端在拥塞控制情况下一次最多能发送数据包的数量。

b)慢启动阈值(ssthresh),是慢启动阶段与拥塞避免阶段的分界点,初始值一般设为 64 KB。

c)回路响应时间(RTT),一个 TCP 数据包从源端发送到接收端,源端收到 ACK 确认时所经历的时间间隔。

d)超时重传计数器(RTO),是数据包从发送到失效的时间间隔,是判断数据包丢失与否、网络是否拥塞的重要参数。通常设为 2RTT 或 5RTT。

早期的拥塞控制协议是不含快速重传和快速恢复的 ADMI 算法,在启动一个 TCP 连接后,开始慢启动过程,每收到一个确认(ACK),拥塞窗口(cwnd)就加倍。这个阶段拥塞窗口以指数级(exponential)增长。当拥塞窗口增大到慢启动阈值(ssthresh)时,拥塞窗口的增长规律由指数增长改变为线性增长,进入拥塞避免阶段。cwnd 增加到一定程度,若发生超时,ssthresh 被重置为当前窗口大小的一半,cwnd 将被重置为 1,重新开始慢启动。即如果发生拥塞,cwnd 将会从一个较大的值急剧降为 1,重新开始慢启动,如果拥塞再度发生,同样的情况再度发生,如图 1 所示。

此时的超时重传算法可描述为:

```
Timeout;
ssthresh = cwnd/2;
cwnd = 1;
```

目前互联网上采用的 TCP Reno 拥塞控制算法在上述算法的基础上,增加了快速重传和快速恢复机制。即,如果源端连续收到三个重复 ACK,就认为该数据包已经丢失,于是重传丢失的数据包而无须等待超时定时器溢出。实质上,不含快速重传和快速恢复的 ADMI 算法仅用超时重传计数器(RTO)作为判定是否发生拥塞的条件,而 TCP Reno 不但使用 RTO 作为判

收稿日期: 2009-12-25; **修回日期**: 2010-01-29

作者简介:闫二辉(1985-),男,河北石家庄人,硕士研究生, 主要研究方向为网络协议与拥塞分析(yanerhui @ yahoo.cn);朱敏(1971-),女,教授,硕导,主要研究方向为网络与信息系统;毛明敏(1987-),女,四川泸州人,硕士研究生,主要研究方向为最优化控制.

定拥塞发生的条件,又增加了“连续收到对同一数据包的三个确认”作为拥塞判定条件,以此来减少拥塞所带来的网络振荡,提高网络利用率。

2　超时重传的相关问题

虽然 TCP Reno 拥塞算法可以根据三个重复的确认(ACK)来判断分组丢失的出现,但是正常情况下收到三个重复的 ACK 需要三个 RTT 的时间,加之 ACK 在传输的过程中可能由于信噪而发生丢失现象,这个时间就可能会更长。超时重传计数器(RTO)通常设为 2RTT 或 5RTT,就存在还没有收到三个 ACK 就已经超时的情况。或是,在 HTTP 短连接业务环境下(目前的网络流量中,短连接业务所占的份额较大),ACK 信息不足,或者根本就没有 ACK 可用于触发高级丢包恢复机制时,就只能依靠超时重传机制[7]。所以 TCP Reno 中,如果拥塞是通过 RTO 检测出来的话,仍然采用 AMID 算法,发送窗口的变化同样会发生剧烈的变化。

另外,当网络发生拥塞时,往往不仅是一个源端,而是网络上一定范围内的若干个源端,这些源端的发送窗口都将从一个较高值急剧下降到 1,重新开始慢启动。显然,这些源端从 1 恢复到一个理想的发送窗口值将会耗费相当长的一段时间,在这段时间内,网络带宽被极大地浪费了,而网络的造价是比较高的,特别是在高带宽、高延迟的网络中,影响尤其突出。

本文通过引入一个新的动态阈值,不仅使网络能有效地从拥塞中恢复过来,而且极大地避免了网络的振荡,使网络利用率有很大程度的提高。

3　一种新的超时处理策略

通过上述分析可见当前超时机制中存在上述缺陷的原因,是因为超时后 cwnd 的值急剧降为慢启动初始值,由一个数据包开始增长,这需要很长时间才能达到稳定传输状态,链路的带宽利用率在很大程度上被浪费。如果能够在保证拥塞得到控制的前提下,适当减小 cwnd 的值,使其在不发生剧烈变化的情况下从拥塞中恢复过来,就可以使上述情况得到缓解。所以,加速拥塞控制算法的收敛是解决问题的关键。理想的情况就是使 cwnd 的值保持在 knee 点和 cliff 点之间,如图 2 所示。

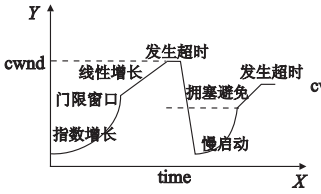

图1　ADMI拥塞控制机制

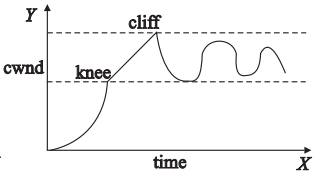

图2　理想的拥塞控制机制

针对现有超时重传机制存在的问题,本文提出了一种新的超时重传策略,其主要思路是在发生超时重传后,利用阈值 w 使拥塞窗口的大小根据发生拥塞前的网络状况,动态调整拥塞窗口,达到以最小的代价从拥塞中恢复过来的目的。如果当前拥塞窗口的减少量不足以使网络从拥塞中恢复过来时,发送端将再一次侦测到网络拥塞,在当前基础上再次执行上一过程,如此重复直到网络从拥塞中恢复过来。

超时重传发生在拥塞避免阶段,此时 ssthresh 的值总是小于 cwnd 的值[8],即 cwnd/ssthresh > 1 故设 p = cwnd/ssthresh。如果 p 比较大说明在发生拥塞前 cwnd 比 ssthresh 大很多,说明此时网络有较长时间没有发生拥塞,网络状况比较好,可以尝试对发送窗口进行微调看能否从拥塞中恢复出来,如果能则不仅可以减少网络的振荡还可以充分利用网络带宽;如果 p 比较小,则说明,当前网络可能刚发生拥塞不久,应对发送窗口进行较大的调整才可能使网络从拥塞中恢复过来。权值取 $w = \cos(1/p)$即可实现上述功能。

证明如下:

$$\text{设 } x = 1/p \text{ 则有 } 0 < x < 1 \text{ 此时}$$

$$w = \cos x, \quad 0 < x < 1 \tag{1}$$

对 w 求导得

$$w' = -\sin x \quad 0 < x < 1 \tag{2}$$

$$w'' = -\cos x, \quad 0 < x < 1 \tag{3}$$

$w' < 0, w'' < 0$,即 w 为减函数且在区间(0,1)上比较光滑,恰好可以实现平滑从拥塞中恢复过来的目的[9]。

改进超时重传算法可以描述如下:

```
Timeout:
ssthresh = w * cwnd;
cwnd = max (2,min( w * cwnd/2,awin));
```

即发生超时后将阈值 ssthresh 设置为 cwnd × w,发送窗口 cwnd 设置为 w × cwnd/2,以此实现根据不同的网络状况动态减小拥塞窗口的目的。

如果发送端收到超时信息,认为网络发生拥塞,则将 cwnd、ssthresh 在当前基础上根据阈值 w 进行相应调整。如果所减小的发送量足以从拥塞中恢复过来,发送端继续慢启动的过程;另一种情况是,由于 cwnd 减少的量不够大,不能使网络从拥塞中恢复过来,发送端必将再一次侦测到链路发生拥塞,又将 cwnd、ssthresh 在当前基础上进行减少。如此重复,直到将 cwnd 的值减到一个特定的值,网络必将从拥塞中恢复过来。

另外,由于改进算法在拥塞发生后 cwnd 和 ssthresh 减小量没有现行 TCP 拥塞控制算法降低量明显,在拥塞比较严重的情况下,可能会导致从拥塞中恢复的时间较长。但考虑到实际中,所有的发送端一旦侦测到拥塞发生都会立即实施拥塞控制[10],发生严重拥塞的可能性较小。由于改进算法的发送窗口和慢启动阈值改变量不是很大,较现行 TCP 拥塞控制算法而言,吞吐量变化不是很剧烈,更加有利于网络带宽的利用,即使在拥塞恢复阶段付出一些时间代价也是值得的。

4　算法仿真与验证

为了验证改进算法在不同网络环境下的有效性及稳定性,本文利用网络模拟器 NS 2.27 作为仿真工具,分别对图 3、4 所示的拓扑结构进行了场景模拟。

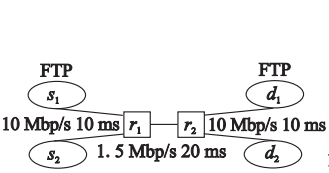

图3　网络拓扑结构1

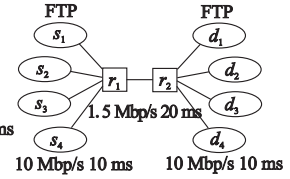

图4　网络拓扑结构2

图 3 中:s_1、s_2 为源端;r_1、r_2 为主干道上的路由器;d_1、d_2 为接收端。在此拓扑中建立两条链路,一条为 s_1 将数据通过主干道路由器转发至 d_1,另外一条为 s_2 将数据通过主干道路由器转发送至 d_2。图 4 在图 3 的基础上多加了两个源端 s_3、s_4 和接收端 d_3、d_4。图中每条链路都采用基于 TCP Reno 算法的 FTP 连接,传输速率为 10 Mbps,延迟为 10 ms,分组大小为 1 000 Byte。主干道路由器 r_1、r_2 之间构成瓶颈链路,传输速率

为 1.5 Mbps,延迟为 20 ms。

4.1　改进算法对发生网络拥塞后的影响

超时重传发生在网络出现拥塞的情况下,为此验证数据传输中发生拥塞后改进算法对吞吐量的影响。在 NS 2.27 中运行图 3 所设计的网络连接,然后对链路上的吞吐量进行检测,可以得到图 5 中的吞吐量变化图。其中:original 代表标准超时重传算法的吞吐量,modified 代表改进超时重传算法的吞吐量。两者在初始慢启动阶段是重合的,当网络发生拥塞时两者图像发生了不同的变化。改进算法的吞吐量比标准算法的吞吐量更稳定,没有像标准算法那样发生剧烈抖动,而且平均吞吐量也较标准算法更大。由此可见,改进算法不仅避免了网络出现剧烈振荡而且也提高了网络利用率。另外,通过对主干道数据的排队情况进行检测可以得到图 6 所示的平均队列变化图,其中 original 代表标准算法平均队列大小,modified 代表改进算法的平均队列大小。可以发现,改进算法的平均队列长度大于标准算法,由此可见改进算法提高了发送速率,并间接提高了主干道数据队列长度。

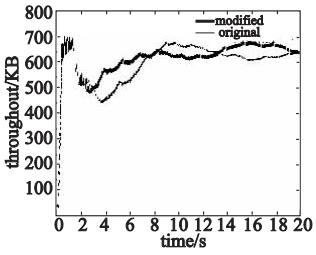

图5　低负载下吞吐量对比

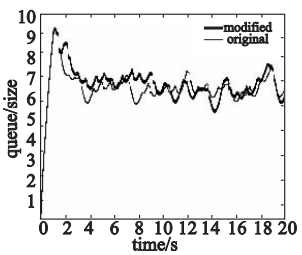

图6　主干路由器平均队列对比

同样,也可以根据表 1 中的实验数据比较标准算法和改进算法对数据传输的影响。在初始的慢启动过程内,标准算法和改进算法所传输的数据量大致相同,从第 2 s 开始发生了网络拥塞,此时改进算法比标准算法更平滑地从拥塞中恢复过来,而且提高了网络利用率,与标准算法相比,发送数据量提高了 4.3%,在整个数据传输阶段链路利用率提高了 1.7%。

表 1　主干路由器队列对比

比较指标	标准算法	改进算法	提高效率/%
0 2 s	24	22	——
2 20 s	1 610	1 680	4.3
链路利用率	0.724	0.737	1.7

考虑到网络拥塞多发生在网络负载比较重的情况下,为此检测高负载环境下改进算法的性能。在图 4 的条件下进行测试可以得到图 7 所示的吞吐量变化图,original 代表标准超时重传算法的吞吐量,modified 代表改进超时重传算法的吞吐量。相比较图 3 而言,图 4 有四条 FTP 链路同时竞争瓶颈处的带宽,更容易发生超时重传,因此改进算法的效果更加明显。图 7 中改进算法的吞吐量始终明显优于标准算法且网络抖动较小,相比较图 5 更加明显地提高了网络利用率。

由此可见,改进算法在网络重负载情况下明显优于标准算法,即使在一般状况下也不差于标准算法。

4.2　改进算法的公平性和侵略性

源端拥塞控制算法要考虑其公平性和侵略性[11]。公平性是指竞争流间的发送速率差别不大,验证方法是看改进算法的各个源端长期发送速率是否相当。侵略性是指当可用带宽增加时,算法能快速增加源端的发送速率,以提高网络利用率。为此将第一条 FTP 链路在 0 s 打开,第二条 FTP 链路在 5.0 s 时打开,这样使得不同时间打开的链路相互竞争以测试改进算法的公平性和侵略性。通过对两条 FTP 链路的吞吐量进行检测,如图 8 所示,其中 FTP-1 表示的是在 0 s 打开的第一条 FTP 链路,FTP-2 表示在 5.0 s 打开的第二条 FTP 链路。可以发现,在打开第二条 FTP 后第一条 FTP 的吞吐量逐渐下滑,同时第二条 FTP 的吞吐量逐渐上升,直到两者差距不大。这不但说明改进算法具有公平性,而且还说明了改进算法具有侵略性,即抢占网络资源的能力。

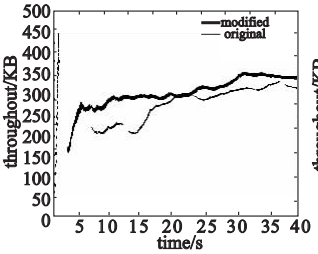

图7　高负载下吞吐量对比

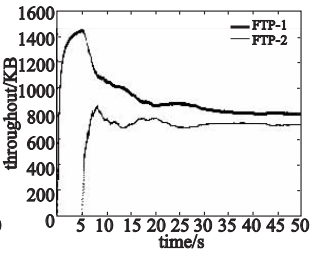

图8　公平性和侵略性

5　结束语

超时重传是 TCP 拥塞控制中非常重要的手段。在网络发生拥塞时,当不能触发拥塞控制算法的高级丢包恢复机制时,只能依靠超时重传从网络拥塞中恢复过来,所以超时重传策略的好坏对 TCP 拥塞控制算法的性能有直接的影响。好的超时重传策略不但能够从网络拥塞中恢复出来,而且还能最大程度地减少网络抖动,维持网络吞吐量在较高水平。

本文提出了一种改进的超时重传策略,通过引入动态阈值的方式改变发生超时时拥塞窗口和慢启动阈值的大小,使拥塞窗口和慢启动阈值根据当前网络状况动态进行调整,以实现维持网络吞吐量在较高水平且不发生剧烈抖动的情况下,从拥塞中恢复出来的目的;并通过 NS2 仿真实验证明了改进算法的有效性。

参考文献:

[1] 林闯,任丰原,单志广. 计算机网络的服务质量(QoS) [M]. 北京:清华大学出版社, 2004.

[2] STALLINGS W. High-speed networks and internets: performance and quality of service[M]. Beijing: Publishing House of Electronics Industry, 2003.

[3] FLOYD S. RFC 3649, HighSpeed TCP for large congestion windows [S]. 2003.

[4] 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.

[5] BALAKRISHNAN H, PADMANABHAN V, SESHAN S, *et al*. TCP behavior of a busy Internet server: analysis and improvements[C]// Proc of the 17th IEEE Annual Joint Conference of the IEEE Computer and Communications Societies. San Francisco, CA: [s. n], 1998: 252-262.

[6] ALLMAN M, PAXSON V, SETVENS W. RFC 2581, TCP congestion control[S]. 1999.

[7] 陈翔,刘卫东,任丰原. 基于最小均方滤波的 RTO 预测算法[M]. 清华大学学报:自然科学版, 2007, 47(4):603-605.

[8] ALLMAN M, FLOYD S, PARTRIDGE C. RFC 2414, Increasing TCP's initial window[S]. 1998.

[9] 茹新宇,刘渊. COS-Slow-Start:一种新的 TCP 慢启动策略[J]. 计算机工程, 2008, 34(5):116-118.

[10] LIN Chuang, LUO Wang-ming. A survey of congestion control in the Internet[J]. Chinese Journal of Computers, 2001, 31(1):1-18.

[11] 杨晓萍,史帅,陈虹. 一种改进的 TCP 拥塞控制算法[J]. 吉林大学学报:工学版, 2006, 36(3):433-437.