

文章编号:1001-9081(2007)02-0282-03

一种自适应改进无线 TCP 性能的算法

刘树立¹, 马 争¹, 蔡爱华²

(1. 电子科技大学 通信与信息工程学院, 四川 成都 610054;

2. 中国电子科技集团 中国电子科学研究院, 北京 100041)

(liushuli95@126.com)

摘 要:对采用 TCP 协议传输数据的实现过程及其在无线网络中可能遇到的问题进行了描述, 针对这些问题提出了一种算法对无线链路下一时刻的误码率进行了估计, 根据这个估计对 TCP 进行了修改以自适应选取合适的 TCP 报文段尺寸来传输数据。在网络仿真器(NS2)中仿真一个误码率时变的无线信道, 把修改后的 TCP 置于 NS 仿真环境中进行仿真, 实验结果表明了该改进方法有效。

关键词:往返时间; 吞吐量; 最大报文段尺寸; 拥塞窗口

中图分类号: TP393.04 **文献标识码:** A

Adaptive algorithm of improving TCP performance in wireless channel

LIU Shu-li¹, MA Zheng¹, CAI Ai-hua²

(1. School of Communication and Information Engineering, University of Electronic and Science Technology of China, Chengdu Sichuan 610054, China;

2. China Academy of Electronics and Information Technology, China Electronic Technology Group Corporation, Beijing 100041, China)

Abstract: The achieving process of TCP and the problems which TCP may encounter in wireless network were presented. Then an algorithm was presented to predict the next bit error rate of wireless link. Based on the prediction, the TCP was modified to adaptively select appropriate Maximum Segment Size (MSS) to transport data. By using NS2, a wireless channel with variable bit error rate was simulated. Last, a simulation about the modified TCP was done, and its result indicated the effectiveness of the modification.

Key words: RTT; throughput; Maximum Segment Size (MSS); Congestion Window (cwnd)

TCP 是 Internet 中一种可靠的传输层协议, 建立于网络层之上。TCP 旨在给互联网提供一种可靠的端到端的字节传输流, 被广泛地应用于电子邮件、文件传输等业务中。TCP 最初是为有线环境设计的, 其性能方面的优化主要是基于传输网络为有线网络这个假设。但随着通信技术的发展, 无线网络在未来因特网中将扮演极为重要的角色, 无线局域网、蓝牙、Ad hoc 网络等无线网络技术推动着无线/有线和移动因特网技术的发展。无线通信要跨越未知的空间以及未知的气象环境, 其通信是依靠可能具有人为干扰等突发差错的无线链路, 很容易造成无线报文的丢失, 从而造成通信的不可靠程度增加。研究表明传统的 TCP 在这种误码率高且变化的无线信道中传输时的性能还有待提升。本文对误码率时变的无线网络中的 TCP 提出一种算法来改进其性能, 并利用网络仿真器 (Net Simulator 2, NS2) 在变误码率的信道中进行仿真及对仿真结果进行分析。

1 TCP 实现机制与性能

1.1 TCP 简介^[1]

众所周知, TCP 通过三次握手建立连接后才开始数据的依序传输, 在传输中拥塞控制主要是通过调整发送端的发送速率, 而这又主要是通过三个变量实现的: 拥塞窗口

(Congestion Window, cwnd), 接收端窗口 (Receivers's Window), 慢速启动阈值 (Slow Start Threshold, SSTHRESH)。发送端一旦监测到数据包丢失 (其原因可能是重传计时器超时, 也可能是收到重复的 ACK 信令), 它就会开始调整发送速率。TCP Reno 是目前应用最广泛的 TCP 协议版本, 它包括慢启动、拥塞避免、快速恢复、快速重传 4 个阶段。TCP 在调整发送端的传送速度时, 以 slow-start threshold (sssthresh) 与 cwnd 的值来区分慢启动阶段 (slow-start phase) 和拥塞避免阶段 (congestion avoidance phase)。

最大报文段长度 (MSS) 表示 TCP 传往另一端的最大块数据的长度。当一个连接建立时, 每一方都有用于通告它期望接收的 MSS 选项 (MSS 选项只能出现在 SYN 报文段中)。当 TCP 发送一个 SYN 时, 它能将 MSS 值设置为外出接口上的最大传输单元 MTU 长度减去固定的 IP 首部和 TCP 首部长度。

1.2 TCP 在无线网络中的问题

在无线移动网络中, 鉴于无线网络的固有特性, 传统 TCP 重传定时器在无线网络下易出现过早超时, 致使 TCP 启用拥塞控制机制, 降低了 TCP 吞吐量。TCP 将无线信道比特差错和移动切换引起的数据包丢失误归于网络发生拥塞而采取拥塞控制措施, 不必要地降低了端到端的吞吐率, 导致自身性能的下降。因此首先需要解决的是如何区分出数据包的丢失是

收稿日期: 2006-08-04; 修订日期: 2006-11-09

作者简介: 刘树立 (1976-), 男, 湖南衡山人, 硕士研究生, 主要研究方向: 网络通信、Ad hoc 网络; 马争 (1957-), 男, 山东齐河人, 教授, 博士, 主要研究方向: 信号处理、信息安全、计算机应用、图像处理等; 蔡爱华 (1962-), 男, 江苏盐城人, 研究员级高工, 主要研究方向: 雷达及电子系统总体技术。

由于拥塞还是传输差错。

另外一个问题在于,传统的 TCP 的 MSS 出现在 SYN 报文段中,经双方协商建立以后,TCP 的发送端就会尽量地以 MSS 的大小发送数据包。但在误码率较高且时变的无线信道中可能会遇到一些问题。在数据传输中,数据块中只要有一个比特发生损坏就会使该数据块无法使用,而一旦发生 TCP 段损坏,就会导致 TCP 进入快速重传或者慢启动阶段,从而造成性能的降低。由概率论的知识,可以得到 TCP 段尺寸(segment_size)、段损坏率(segment_loss)和误码率(Pe)的关系:

$$\text{segment_loss} = 1 - (1 - \text{Pe})^{\text{segment_size}}$$

由上式可知,TCP 段尺寸越小,TCP 段损坏率越小。本文正是基于这一观点对无线网络中的 TCP 提出了一种改进方法。

2 调整 TCP 段尺寸原理与算法

在实际无线网络中,很多情况都是误码率处于逐渐的变化过程之中。对每一个处于暂稳态的误码率,文献[2-3]已经研究表明具有唯一的一个 TCP 段尺寸可以使网络传输达到最大性能。但现有的研究是利用当前时刻的段重传率来确定未来时刻使用的 TCP 报文段大小,这种方法可能缺少前瞻性,对于网络质量突然发生较大变化时可能因为滞后而使性能得不到提高,而且现有的研究没有形成具体的算法,并且其提出的机制有时可能会无法正常工作。为此,本文提出一种算法来预测未来时刻的误码率,从而自适应地改变 TCP 段尺寸。

在 TCP 建立连接后,在 TCP 发送端计算每个 10s 内发送的数据段总量 total_sent 和重传的数据段总量 total_retrans 两个变量,就可以得到 10s 内的段重传率,再根据段重传率计算此 10s 内的误码率 Pe,具体的实现算法描述如下:

```
/* 初始化,计算误码率. 本算法在 TCP 中实现是基于只优化到最大误码率 Pe=1E-4, 如果误码率更大,只需稍作改动即可. 计算误码率肯定存在误差,但影响不大. segment_size 包括了 TCP 和 IP 首部 */
double P[3], Pe[3]
double total_sent[3]
double total_retrans[3]
double segment_size[3]
double a[3], b[3], c[3]
for (k = 1; k < 3; k++)
{
    a[k] = total_sent[k]
    b[k] = total_retrans[k]
    c[k] = segment_size[k]
    //取得 10s 和 20s 时的数据
}
for (n = 1; n < 3; n++)
{
    if a[n] = 0, then
        Pe[n] = 1E-4 //网络最高 BER
    else
    {
        P[n] = 1 - [1 - b[n]/a[n]] * [1/8/c[n]]
        if p[n] >= 1E-4 then
            Pe[n] = 1E-4
        else Pe[n] = P[n]
    }
}
```

/* 利用 n=1,2 时的 BER 预测 n=3 时的 BER. Pe[1], Pe[2] 为实测值, Pe[3] 为预测值. 引入调节因子 β 使 MSS 在网络质量变差时不致剧烈减小,而在网络质量趋好时能很快增加,在作者综合许多实验结果后,发现 β 取为 0.5 较为合适 */

```
if Pe[1] = 0
    Pe[3] = Pe[2];
if Pe[1] != 0
    Pe[3] =  $\beta$  * Pe[2] * Pe[2]/Pe[1];
//根据 Pe[3] 选取 MSS 大小
if (Pe[3] < 1E-6)
    MSS = 1000;
else if (1E-6 <= Pe[3] < 1E-5)
    MSS = 400;
else if (Pe[3] >= 1E-5)
    MSS = 100;
/* 用自适应选取的 MSS 进行这一个 10s 的数据传输,结束时分别取得这个 10s 的 total_sent[3] 和 total_retrans[3] */
//进行赋值
a[1] = a[2]
b[1] = b[2]
c[1] = c[2]
a[2] = total_sent[3]
b[2] = total_retrans[3]
c[2] = segment_size[3]
//赋值结束后进入下一个自适应阶段
```

在此,对于本算法还要说明的是:本文在对 MSS 选值时,只是参照文献[3]的测量值大致作了优化,不一定达到最优,例如在误码率 BER 极低时, MSS 应该取为路径 MTU 减去 TCP 和 IP 的首部大小时一般能获取最佳性能。

3 实验及数据

3.1 仿真拓扑结构和配置

为了分析算法在误码率变化的信道中的性能,本文用 NS2 开发出如图 1 所示的网络结构。在节点 2 和 3 之间为误码率时变的无线瓶颈链路,其余为有线网络,并且假设有线网络没有差错。在节点 1 和 5 之间为 TCP Reno 连接,在节点 0 和 4 之间为加入自适应算法的 TCP Reno (在本文中简称为 TCP RenoAd) 连接,两者通信业务均为 FTP,两个 TCP 连接分开进行,各自的仿真时间均为 480s。在无线信道中,240s 内有 48 种不同的误码率,在每一个误码率模型上停留 5s,它们分布在 $1.25E-7$ 和 $1E-4$ 之间,状态转换矩阵如下:

$$\begin{bmatrix} 0 & 1 & 0 & \cdots & \cdots & \cdots & 0 & 0 \\ 0 & 0 & 1 & 0 & \cdots & \cdots & 0 & 0 \\ 0 & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \vdots \\ \vdots & & & & & & & \vdots \\ \vdots & \cdots & \cdots & \cdots & \cdots & 0 & 1 & 0 \\ 0 & 0 & \cdots & \cdots & \cdots & \cdots & 0 & 1 \\ 1 & 0 & 0 & \cdots & \cdots & \cdots & 0 & 0 \end{bmatrix}$$

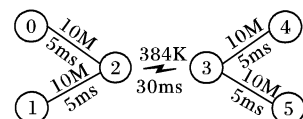


图1 仿真拓扑图及配置关系

3.2 仿真结果分析

1) 吞吐量的分析

整个仿真过程两个连接的平均吞吐量如表 1 所示,从表中可以看到:整个 480s 的仿真时间内,TCP Reno 在单位时间内的平均吞吐量明显不如改进后的 TCP RenoAd,这充分说明了本文提出的算法是有效的。

下面是每隔 10s 进行一次平均吞吐量统计的情况,两个不同的 TCP 连接的平均吞吐量变化情况分别如图 2 所示,在

吞吐量变化图中,每一个时间点 t 的值代表着在 $[t-10, t]$ 这段时间内的平均吞吐量。

表1 单位时间内吞吐量

TCP 类型	吞吐量
TCP Reno	168 kbps
TCP RenoAd	213 kbps

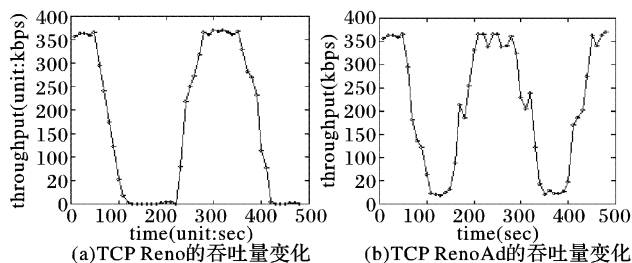


图2 TCP Reno 和 TCP RenoAd 的吞吐量变化

从图中可以看出:TCP RenoAd 较好地预测到了 $[80, 160]$ 和 $[320, 400]$ 两个高误码率的阶段,并适时地调整了 MSS 的大小,使得这两个阶段的吞吐量有了较大的改善;而在 TCP Reno 中由于没有自适应算法,不能追踪到误码率的变化以及由于 RTT 和 cwnd 导致的滞后的影响,使得 $[100, 200]$ 和 $[400, 480]$ 这两个区间出现了许多吞吐量为 0 的时刻,其中 $[400, 480]$ 这个阶段的误码率设置得很小却由于 $[300, 400]$ 这个时间段高误码率所产生的滞后效应导致了吞吐量为 0。对于 $[200, 280]$ 这个极低误码率阶段的吞吐量:TCP Reno 由于 $[80, 160]$ 时间段的高误码率所产生的滞后影响,吞吐量处在上升阶段,浪费了这时期网络能进行高效传输的能力;而 TCP RenoAd 由于预测到 $[80, 160]$ 时间段的高误码率从而自适应地减小 MSS,使得数据段丢失率大为降低,消除了滞后影响,于是在 $[200, 280]$ 这个极低误码率阶段吞吐量达到了最大。

2) 往返时间 RTT 的分析

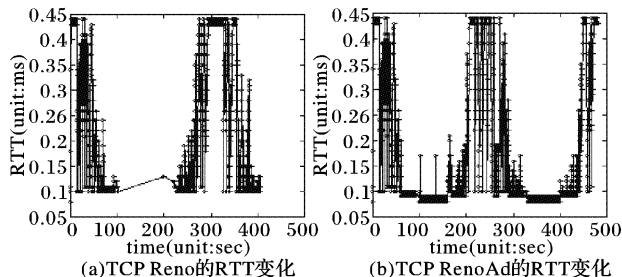


图3 TCP Reno 和 TCP RenoAd 的 RTT 变化

下面,从 TCP 拥塞机制的细节方面对仿真进行分析,首先对两个连接的 RTT 进行了跟踪,如图 3 所示。TCP 超时与重传中最重要的部分就是对一个给定连接的往返时间(RTT)的测量。由于路由器和网络流量均会变化,因此可以认为这个时间可能经常会发生变化,TCP 应该跟踪这些变化并相应地改变其超时时间(RTO)。从图中可以看出:TCP Reno 由于高误码率和滞后的影响,在 $[100, 200]$ 时间段 TCP 分组过大不能传输成功,测量不到 RTT 的数值,而且这个时间段附近 RTT 的变化起伏很大,同样由于滞后的影响,在 $[400, 480]$ 时间段没有跟踪到 RTT 的值,这说明了这个时期没有数据的传输进行;但同时可以看出在 TCP RenoAd 中, $[80, 160]$ 和 $[320, 400]$ 这两个时间段里减小了 MSS 的值,使得 TCP 分组的传输成功概率大为提高,同时也跟踪到了 RTT 的值,说明了这两个阶段进行了有效的数据传输。更能说明问题的是在

$[400, 480]$ 这个时间段内,TCP RenoAd 跟踪到了大量的 RTT 的数据,并且与 $[0, 40]$ 阶段的 RTT 图形极为相近,这也说明了 TCP RenoAd 能消除高误码率带来的滞后影响,很快地适应网络质量变好的情况,当然这主要归功于它比 TCP Reno 多了一个自适应算法。

3) 拥塞窗口 cwnd 的分析

拥塞窗口是发送端进行流量控制的工具,其变化很好地说明了 TCP 在信道中进行流量控制的过程。图 4 给出了两个 TCP 连接的 cwnd 的变化情况。

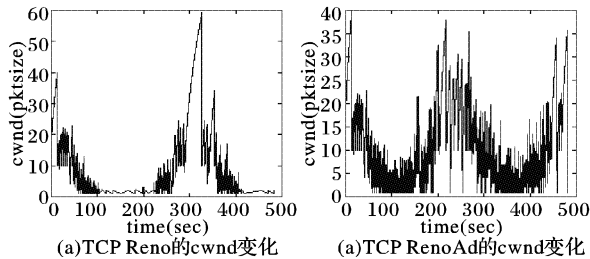


图4 TCP Reno 和 TCP RenoAd 的 cwnd 变化

从图中可以看出:就整体而言,TCP RenoAd 的 cwnd 明显比 TCP Reno 要大;在 $[80, 160]$ 这个阶段,TCP Reno 由于 TCP 分组不能成功传输,误以为发生拥塞,频繁进入拥塞避免和慢启动阶段,最后导致 $[100, 200]$ 时间段内 cwnd 总维持在最小值;但可以看到 TCP RenoAd 在同样的时间里,cwnd 是有增长的,这说明即使在高误码率的时间段里,TCP RenoAd 也成功地进行了分组的传输;同时还可以看到,TCP RenoAd 能较快地适应网络质量的变化,充分地利用网络的传输能力,但 TCP Reno 却由于滞后效应,高误码率阶段造成的重传、指数退避、慢启动和窗口变小将会浪费网络质量改善时的传输能力,这种情形突出地表现在 $[400, 500]$ 时间段内。

4 结语

本文首先对 TCP Reno 作了简要的介绍,指出了传统 TCP 在无线环境中可能遇到的问题,提出了一种自适应改变 TCP 最大报文段尺寸 MSS 的算法,将这种算法融合进 TCP Reno 形成了 TCP RenoAd 协议,并对这两种协议进行了仿真和分析,结果表明该算法能提高变误码率的无线网络通信的性能。关于无线链路中 TCP 的性能,已有学者提出了如间接 TCP(indirect TCP)^[4] 和增加一种探查代理^[5] 等机制。但同时可以看到,在无线网络中,很多时候网络质量的变化是有迹可寻的、是可以预测到的,那么如何改进 TCP 协议使其能自适应地改变自己的某些机制以适应网络质量的变化将会成为无线 TCP 的研究方向。

参考文献:

- [1] ALLMAN M, PAXSON V, STEVENS WR. IETF RFC2581, TCP Congestion Control[S]. IETF RFC2581, April 1999.
- [2] 刘俊,隆克平,徐昌彪,等. 两种改善无线 TCP 性能的新机制[J]. 电子学报, 2004, 32(12): 2059 - 2062.
- [3] 徐昌彪,隆克平,杨士中. 无线网络中基于误码丢包的 TCP 速率调节策略[J]. 计算机学报, 2002, 25(4): 438 - 444.
- [4] BAKNE A, BADRINATH BR. I-TCP: Indirect TCP for Mobile Hosts [A]. Proc of the 15th Int' 1 Conf on Distributed Computer Systems [C]. Vancouver, CA: IEEE Computer Society, 1995. 136 - 143.
- [5] BALAKRISHNAN H, SESHAN S, KATZ RH. Improving Reliable Transport and Hand off Performance in Cellular Wireless Networks [J]. ACM WirelessNetworks. 1995, 1(4): 469 - 481.