TCP Veno 在 MANET 环境下的性能研究

彭海英,唐伶俐,唐 红
PENG Hai-ying,TANG Ling-li,TANG Hong

重庆邮电大学,重庆 400065
Chongqing University of Posts and Telecommunications,Chongqing 400065,China
E-mail:pphhyylbw@yahoo.com.cn

PENG Hai-ying,TANG Ling-li,TANG Hong.Performance study of TCP Veno in MANET environment.Computer Engineering and Applications,2010,46(9):85-87.

Abstract: The performance of TCP Veno is tested by using NS-2 in the MANET environment.The simulation results show that the performance of TCP Veno is superior to that of TCP Reno in the MANET environment with background traffic,random packet loss and congestion;furthermore,the larger background traffic and random packet loss are,and the shorter time congestion takes,the more obvious the superiority of TCP Veno is.

Key words: Mobile Ad hoc NETwork(MANET);Transmission Control Protocol(TCP);congestion;fast retransmit;timeout

摘 要:将 TCP Veno 运行在 MANET 环境下,采用 NS2 对其性能进行了仿真测试。仿真结果表明:在存在背景流、有随机丢包并且存在拥塞的 MANET 网络中,TCP Veno 的性能优于 TCP Reno,而且在背景流越大,达到拥塞的时间越短、随机丢包越大,TCP Veno 的优越性更会非常明显。

关键词:移动自组网络;传输控制协议;拥塞;快速重传;超时
DOI:10.3778/j.issn.1002-8331.2010.09.025 **文章编号**:1002-8331(2010)09-0085-03 **文献标识码**:A **中图分类号**:TN915.04

近十年来,互联网得到了快速发展和广泛应用。而这些应用很大程度上是在 TCP/IP 协议上发展起来的。TCP/IP 协议用以实现异种网络之间的互联与互通。1986 年,在一系列拥塞导致系统崩溃之后,TCP 协议被正式提出,以解决拥塞问题。TCP Tahoe 是早期的 TCP 版本,它包括了 3 个最基本的拥塞控制算法-"慢启动"、"拥塞避免"和"快速重传"。TCP Reno 在 TCP Tahoe 基础上增加了"快速恢复"算法。TCP Vegas[1]使用了"前摄拥塞检测",使相对于 TCP Reno 流量大幅增加,但在实际网络中相对于 TCP Reno 竞争性却不够。TCP NewReno 对 TCP Reno 中的"快速恢复"算法进行了修正,它考虑了一个发送窗口内多个数据包丢失的情况。TCP SACK[2]关注的也是一个窗口内多个数据包丢失的情况,它避免了之前版本的 TCP 重传一个窗口内所有数据包的情况,包括那些已经被接收端正确接收的数据包,而只是重传那些被丢弃的数据包。

以上版本的提出,极大地提高了 TCP 的性能,由于 TCP 协议最初是针对有线网进行制定的,其设计理论是基于"IP 包的丢失是网络拥塞引起的"这一假定。在无线环境中,除了网络拥塞会引起 IP 包丢失,还存在着普遍的随机包丢失,导致 TCP 协议性能下降。2001 年,傅承鹏博士提出了 TCP Veno[3]的方案,成功解决了 TCP 在无线环境中存在的随机丢包问题。经过大量的研究和测试,TCP Veno 被证明在不损失有线网络带宽的情况下,在无线网络中流量得到了较大的提高。但是在 MANET 环境中,对 TCP Veno 的研究还很少,特别是 MANET[4]环境下 TCP Veno 的性能还有待测试。

1 TCP Veno 的原理

1.1 TCP Veno 的原理

TCP Reno 将数据包的丢失归咎于网络拥塞,并针对网络拥塞做出了相应的改善。在有线网络中,数据的传输是比较可靠的,误比特率一般较低,一般认为数据丢失都是拥塞造成。但在无线环境中,无线链路不够可靠,常常伴随较高的误比特率,导致随机丢失经常出现,如果仍然使用 TCP Reno,就会把随机丢失当成拥塞丢失来处理,导致 TCP 性能的下降。因此,无线环境中,TCP Veno 应运而生。TCP Veno 在 TCP Reno 基础上进行改进。TCP Veno 的工作过程与 TCP Reno 一样包括 4 个阶段:超时重传,流量控制,慢启动,快速重传和和快速恢复[5-7]。

1.2 TCP Veno 在 TCP Reno 的修改

1.2.1 拥塞丢失和随机丢失的判定

TCP Veno[8-9]根据拥塞丢失和随机丢失的特征,做出了区分,在程序中进行区别处理,较大地改善了在无线环境中的传输率。TCP Veno 充分利用了 TCP Vegas 中的机制,根据拥塞窗口 *cwnd* 和往返时间对拥塞丢失和随机丢失进行区分。具体如下:

基金项目:国家自然科学基金(the National Natural Science Foundation of China under Grant No.#60572089)。
作者简介:彭海英(1973-),女,副教授,研究方向:计算机通信网;唐伶俐(1971-),女,讲师,研究方向:移动通信;唐红(1957-),女,博士,教授,研究方向:计算机网络与通信,嵌入式系统及应用。
收稿日期:2008-09-19 **修回日期**:2008-12-03

采用了期望速率和实际速率的概念：

期望速率：$EXPECTED=CWND/BASERTT$

实际速率：$ACTUAL=CWND/RTT$

其中 RTT 为平滑的 TCP 数据包往返时间，而 $BASERTT$ 为观测到的最小的 RTT。

两个速率的差为：$DIFF=EXPECTED-ACTUAL$。设置参数 $N=ACTUAL\times(RTT-BASERTT)=DIFF\times BASERTT$。

假设 $BASERTT=1/2\times RTT$，则 $N=1/2\times CWND$；假设 $BASERTT=1/3\times RTT$，则 $N=2/3\times CWND$。可见，如果 N 越大，表示 $BASERTT$ 与 RTT 差距越大，即网络越拥塞。因此，可以用 N 值的大小来代表拥塞的程度。

在检测到包丢失时，如果 $N<\beta$，N 值较小，TCP Veno 认为本次丢失是拥塞随机丢失；如果此时 $N\geqslant\beta$，则认为是拥塞丢失。其中，β 的典型值为 3。

$BASERTT$ 在连接中是不断更新的，每次都比较并记录下最小的 RTT 值。在超时或重复 ACK 发生时，$BASERTT$ 也需要重置，并在其后的慢启动或快速恢复中进行更新。

1.2.2 线性增加(AI)策略

在 TCP Reno 中，在 $CWND>SSTHRESH$ 时，进入拥塞避免阶段。在每个往返时间里，拥塞窗口增加 1。即：每次收到一个 ACK，拥塞窗口变为 $CWND+1/CWND$。而在 Veno 中，采用以下的策略：

if $N<\beta$ then //此时 N 较小，带宽没有完全得到利用，即不拥塞

{每次收到一个新的 ACK，使 $CWND=CWND+1/CWND$；}

else if $N\geqslant\beta$ then //N 较大，表示有拥塞迹象，带宽得到充分利用

{每两次收到一个新的 ACK，使 $CWND=CWND+1/CWND$；}

在有拥塞迹象的情况下，每两个 ACK 才使 $CWND$ 增加 $1/CWND$，可使 TCP 振荡时间（即从慢启动到拥塞的时间）增加，增加了数据传输量，提高了 TCP 的效率。

1.2.3 倍数减少(MD)策略

在 TCP Reno 中，倍数减少策略用于两个地方，一个是慢启动，另一个是快速重传。在慢启动时，将慢启动门限倍数减少，即 $SSTHRESH=CWND/2$。TCP Veno 此时与 TCP Reno 相同。在快速重传时，即使没有等到超时，收到 3 个重复 ACK，也认为相应的包丢失。这时也采用了倍数减少：

1.快速重传时，重传丢失的包

设 $SSTHRESH=CWND/2$ //倍数减少

$CWND=SSTHRESH+3$； //快速恢复

2.在之后如果每次还收到重复 ACK，每次将拥塞窗口增加 1；

3.若收到新的 ACK，令 $CWND=SSTHRESH$。

在 Veno 中，修改了 Reno 快速重传的第 1 步：

if $N<\beta$ then $SSTHRESH=CWND\times4/5$ //可能发生了随机丢失

else if $N\geqslant\beta$ then $SSTHRESH=CWND/2$ //拥塞丢失，类似于 TCP Reno

通过修改 AI/MD 策略，TCP Veno 在拥塞避免阶段区分了不拥塞和拥塞的情况，在快速重传时区分了随机丢失和拥塞丢失的情况。前者减少了 TCP 的振荡，增加了数据量；后者保证了在随机丢失时，在一个高位开始拥塞避免，同样也提高了数据量。因此 TCP 通过拥塞状态的判断，很好地解决了随机丢失的问题。由此，在随机丢失大量存在的 MANET 环境中，TCP 的传输速率理应得到较明显的改善。

2 MANET 仿真环境下对 TCP Veno 协议的测试

2.1 测试方案设计

TCP Veno 在 NS2 下的仿真与 TCP Reno 类似，是在 MANET 仿真下的测试[10-11]，采用 Reno 做为背景流。测试的重点在于观察 Veno 的窗口变化和流量大小，并与 Reno 进行比较，看看是否在 MANET 环境下 Veno 的性能与 Reno 比较有所提高。

测试可以按照在有拥塞的 MANET 环境下 Reno 和 Veno 是否产生竞争（即是否有背景流），是否误码，这几种情况来进行测试。由于误码大小和拥塞（或网络容量）可以在 TCL 程序中设置，可以分成两种情况进行测试，一种是单独对两种协议进行测试，即无背景（不竞争）方式，另一种是将两种协议放在一起进行测试，即有背景（竞争）方式。

对测试的分析集中在对到窗口的变化情况和流量大小变化情况的讨论，因此，主要监控对象是窗口、平均流量和重传率。并通过分析生成窗口变化图、平均流量图和重传率图来观察 Veno 相对于 Reno 的变化。

2.1.1 测试方案 1

这是 Reno 和 Veno 不同时传输（无背景）的情况，只需要生成 3 个节点（两个无线节点 0 和 2，一个中转节点 1），中间节点 1 处于移动状态中，在同样的环境下分别对 Reno 和 Veno 进行测试。测试模型如图 1。

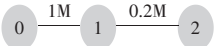

图 1 无背景情况的 Veno 测试方案

可以使用 err 来设置不同的错误率（此处的错误率不同于前面的传输层误包率，而是在 UMTS 的物理层的误包率），通过设定中转节点 1 和移动终端 2 之间的链路容量和序列大小来控制系统达到拥塞的时间，如：容量都比较大的时候，Reno 和 Veno 需要很长时间才能达到拥塞，在指定监控时间内就有可能还没有发生拥塞；容量都较小的时候（如 0.2M），达到拥塞的时间就比较短，这样可以出现多次拥塞现象。

2.1.2 测试方案 2

这是 Reno 和 Veno 同时运行在一条链路（有背景）上的情况，此时需要生成 9 个节点（4 个无线收发节点，4 个无线固定中转节点，一个移动中转节点），在同样的环境下分别对 Reno 和 Veno 进行测试。采用同时传输两种协议有两个目的，一个是使网络上存在背景，更符合一条链路多个信道的真实情况；二是在同样的环境下，使 Reno 和 Veno 产生容量竞争，在竞争情况下，比较 Veno 是否相对于 Veno 性能有所提高。

测试模型如图 2，与测试方案 1 不同之处是实现了两个TCP 节点的同时监控。

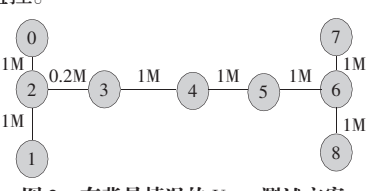

图 2 有背景情况的 Veno 测试方案

2.2 Veno 在 MANET 下的测试

按照前面的分类，将测试分为以下几种情况分别进行测试。

2.2.1 无背景、有拥塞、无误码情况下的测试

此时采用测试方案 1。设置发送终端 0 和中转节点 1 之间

的容量为 1M,中转节点 1 和接收终端 2 之间的容量为 0.2M,节点 1 以 15 m/s 的速度移动,此时 Reno 和 Veno 将在较快时间内达到拥塞。

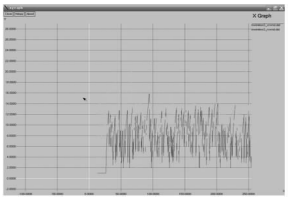

图 3　无背景、无拥塞、无误码时 Reno 与 Veno 窗口对比图

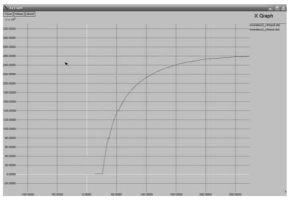

图 4　无背景、有拥塞、无误包时 Reno 与 Veno 吞吐量对比图

设置误包率为 0,对 Reno 和 Veno 各进行一次测量,测试时间 600 s,红色代表 Veno,绿色代表 Reno。测得 Reno 吞吐量为 288 000 b/s,Veno 吞吐量为 275 000 b/s,窗口对比图如上图:通过对 CWND 窗口的进行监控,可以看到,Veno 的周期要比 Reno 长。Reno 流量稍大于 Veno 的吞吐量。在没有竞争,有拥塞但无丢包率的情况下,出现 Reno 较大或者 Veno 较大都是有可能的,这同测试的环境有关系。这种情况在实际的 MANET环境中应该出现得较少,因为 MANET 环境中均存在高误码率。

2.2.2　无背景、有拥塞、有误码情况下的测试

设置发送终端 0 和中转节点 1 之间的容量为 1M,中转节点 1 和接收终端 2 之间的容量为 0.2M,即有拥塞情况,节点 1 以 15 m/s 的速度移动,此时 Reno 和 Veno 将在较快时间内达到拥塞。此时设置不同的错误率进行测试。

(1)将错误率设为 0.01(统计上相约为传输层误包 0.01),对 Reno 和 Veno 各进行一次测量,测试时间 600 s,测得 Reno 吞吐量为 258 000 b/s,Veno 吞吐量为 255 000 b/s,得到窗口图,吞吐量图对比图如下(以 b/s 为单位)。

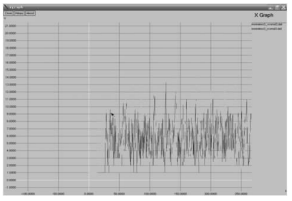

图 5　无背景、有拥塞、有误码,错误率 0.01 时 Reno 与 Veno 窗口对比图

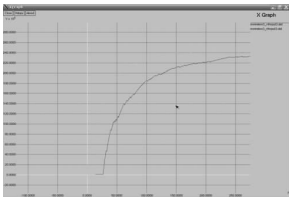

图 6 无背景、有拥塞、有误码,错误率 0.01 时 Reno 与 Veno 总流量对比图

可见,从吞吐量图上看,Reno 的流量要大一些,Reno 性能要比 Veno 要好,这是在误包率比较小的情况下会出现的,因为等待出现拥塞的时间长,Reno 的窗口 CWND 上升速度比 Veno 快一倍,而误码率相当小,导致 Reno 在大部分时间都处于一个大的窗口,而 Veno 的窗口小一些,使 Reno 的性能比较好。另外,在有误码的环境下,两种协议的窗口值,吞吐量比无误码环境中明显降低。

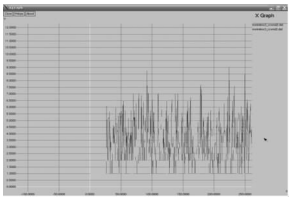

图 7　无背景、有拥塞、有误码,错误率 0.05 时 Reno 与 Veno 窗口对比图

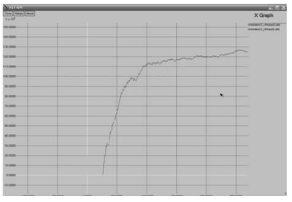

图 8　无背景、有拥塞、有误码,错误率 0.05 时 Reno 与 Veno 总流量对比图

(2)将误码率设为 0.05,使误码率增大(统计上相约为传输层误包 0.05),对 Reno 和 Veno 各进行一次测量,测试时间 600 s 得到以下结果。

由于误包率增大,流量图变得不规则起来,跟误包率 0.01 时相比,窗口值变得越来越小,观察窗口图,两者都在 2~5.5 之间变化起伏。测得 Reno 吞吐量为 128 000 b/s,Veno 吞吐量为 132 000 b/s。总体上看,窗口值和吞吐量都明显降低,相比错误率 0.01,两者的的吞吐量均下降了。但 Veno 窗口值略大于 Reno 的窗口值,Veno 此时的优势逐渐体现了出来由此可见,随着误包率的增大,Veno 的性能比 Reno 更好。

2.2.3　有背景、有拥塞、有误码情况下的测试

这是网络出现最多的情况,在有背景的情况下测试 Veno,采用方案 2,最符合真实情况。实验测试了典型误包率分别为 0.01 和 0.05 的情况,这样的设置与 MANET 环境的情况比较接近。

(1)将误包率设为 0.01(统计上相约为传输层误包 0.01),对 Reno 和 Veno 各进行一次测量,测试时间 600 s,测得 Reno 吞吐量为 36 000 b/s,Veno 吞吐量为 41 000 b/s,得到窗口图,吞吐量图对比图如下(以 b/s 为单位)。

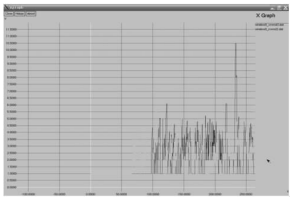

图 9　有背景、有拥塞、有误码,错误率 0.01 时 Reno 与 Veno 窗口对比图

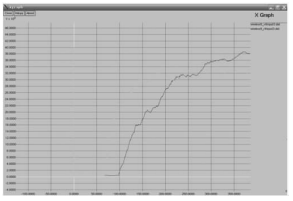

图 10　有背景、有拥塞、有误码,错误率 0.01 时 Reno 与 Veno 总流量对比图

在有背景流,误码率很小的情况下,Veno 的性能逐渐体现出来,与 Reno 相比,吞吐量稍微大一些。而在没有背景时,即 2.2.2 节的第一次测量中,Veno 比 Reno 吞吐量小。两次结果比较,同样的错误率情况下,在有背景时,Veno 的性能优势是非常明显的。分析其原因,由于背景增加了拥塞的程度,导致慢启动的增多,而 Veno 在拥塞时有更长的周期,相应的流量就会增大。

(2)将误包率设为 0.05(统计上相约为传输层误包 0.05),对 Reno 和 Veno 各进行一次测量,测试时间 600 s,测得 Reno 吞吐量为 17 000 b/s,Veno 吞吐量为 11 000 b/s,得到窗口图,吞吐量图对比图如下(以 b/s 为单位)。

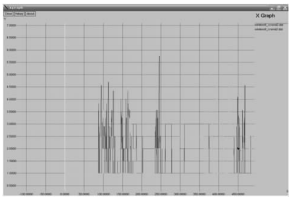

图 11　有背景、有拥塞、有误码,错误率 0.05 时 Reno 与 Veno 窗口对比图

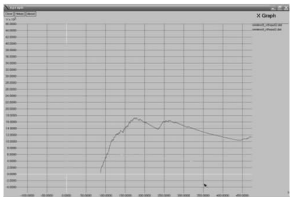

图 12　有背景、有拥塞、有误码,错误率 0.05 时 Reno 与 Veno 总流量对比图

在有背景流,误包率增大的情况下,Veno 的性能优势比较明显,误包率越大,尽管流量减少,但 Veno 相对 Reno 性能会增加。随着误包率的增加,两种协议的吞吐量急剧下降。分析其原因,由于背景增加了拥塞的程度,导致慢启动次数增多,所以使网络中两种协议的兼容性降低。

(下转 110 页)