

自组网 TCP Vegas 协议研究^{*})

拱长青^{1,2} 王庆辉¹ 王光兴¹

(东北大学信息科学与工程学院 沈阳 110004)¹ (沈阳航空工业学院计算机学院 沈阳 110034)²

摘要 对自组网内的 TCP 拥塞控制算法进行了仿真比较研究。主要研究了自组网环境下 TCP Vegas 拥塞控制算法的性能表现,并尝试对 Vegas 算法的拥塞避免机制进行改进。改进的主要措施是对拥塞避免阶段网络通信状况进行细致分解,并根据不同的网络状况采取相应的措施;进而提出了 Vegas1 和 Vegas2 两种改进方案。与 Reno、SACK、Vegas 等算法的仿真对比结果表明,无论在 TCP 吞吐量方面,还是在 TCP 段的传送效率方面,改进后的 Vegas2 算法都优于其他算法。

关键词 TCP, 自组网, Vegas, 拥塞避免

Analysis of TCP Vegas Performance over Mobile Ad Hoc Networks

GONG Chang-Qing^{1,2} WANG Qing-Hui¹ WANG Guang-Xing¹

(School of Information Science & Engineering, Northeastern University, Shenyang 110004)¹

(College of Computer, Shenyang Institute of Aeronautical Engineering, Shenyang 110034)²

Abstract TCP Congestion Control Algorithms are simulated and analyzed in mobile ad hoc networks. The main research work concentrates on the performance analysis of TCP Vegas algorithms. Based on this simulation research work, the improved congestion control algorithms of TCP Vegas are presented. The primary improvement measures are to subdivide the congestion avoidance period into some more small subsection, to divide network conditions into more particular state; then to take special countermeasure for special conditions. Then the Vegas1 and Vegas2 algorithms can be formed. Simulation result shows that the Vegas2 algorithm is superior to other algorithms on the average throughput.

Keywords TCP, Mobile ad hoc networks, Vegas, Congestion avoidance

1 引言

自组网(Ad hoc 网络)的研究与应用在近几年逐渐为人们所重视,相应的技术发展也不断深入。由于自组网有可能会与现行的其他网络相连,如接入固定的 Internet 之中,而 Internet 中又大量地利用 TCP 协议进行信息传输,因此有必要考虑 TCP 协议应用于自组网时的性能表现问题。事实上,仅在自组网内部,也有可能涉及使用 TCP 协议进行可靠的文件传输操作。

关于 TCP 协议在自组网中的应用问题,近期的研究才稍多一些^[1~7]。文[1]和文[2]将 TCP 协议在自组网中的应用研究总结得较为全面,认为网络拓扑的不断变化和路由的更新使得自组网的 TCP 端到端控制十分困难,性能因而下降较多。文[3~7]中分别有一些对几个常见拥塞控制算法的改进措施,或者是提出了较新的拥塞控制算法。

本文研究的出发点是基于传输层内处理,即只是利用在传输层收集的信息进行该层次的传输控制处理,不涉及其它层次的信息使用,重点研究 Vegas 算法的改进。目前一些文献认为 SACK 算法的性能在自组网中有较好表现^[3],进而专注于 SACK 算法的改进。而本文通过仿真研究比较发现这样的事实,TCP 包的大小对各种拥塞控制算法的性能有较大影响,在很多情况下 SACK 算法的性能不及 Vegas 算法。例

如,虽然 65536Byte 的 TCP 数据包使得 Vegas 算法无法工作,包接收率极低;但是 512Byte 的 TCP 包却使 Vegas 算法性能达到最优。因此,本文在 Vegas 算法基础上对其进行了改进尝试。

2 Vegas 拥塞控制算法介绍

2.1 文[8]中的 Vegas 拥塞控制算法

该算法能较好地预测网络带宽使用情况,其公平性和效率都较好。但是 Vegas 与其他算法在竞争带宽方面处于劣势^[9]。

Vegas 算法根据 expected rate(期望的吞吐量)和 actual rate(实际吞吐量)的差值 delta 来估计网络中可用的带宽。若实际吞吐量和期望吞吐量数值很接近,delta 的值较小,可以认为网络没有发生拥塞;若实际吞吐量远小于期望吞吐量,delta 的值较大,则网络很有可能出现拥塞状况。据此,Vegas 算法可以更新拥塞窗口,以保证网络传输的正常运行。

定义 $expected\ rate = cwnd / base\ rtt$;

$actual\ rate = cwnd / rtt$;

$delta = expected\ rate - actual\ rate$ 。

上式中,cwnd 为当前拥塞窗口的大小,base rtt 是该连接上最小的一次 RTT 记录,而 rtt 是当前的测得的 RTT 值。

改变数据发送速率,也就是调整拥塞窗口的算法可描述

^{*})基金项目:国家高技术研究发展计划项目(863-708-2-4)。拱长青 博士研究生,主要研究方向为无线网络;王庆辉 博士研究生,主要研究方向为无线网络;王光兴 教授,博士生导师,主要研究方向为计算机网络。

如下:

```
if  $\delta < \alpha$            $cwnd + 1$ ;
if  $\delta > \beta$            $cwnd - 1$ ;
if  $\alpha \leq \delta \leq \beta$    $cwnd$  不变。
```

具体实现时,一般取值为 $\alpha=1, \beta=3$ 。

2.2 Vegas A——文[10]中对 Vegas 算法的改进

上述算法中 α 和 β 值固定不变,拥塞控制效果受到限制,算法对网络的适应性不是很强。文[10]中提出了改进的 Vegas A 算法,使 α 和 β 值可以自动调整,能更好地改善拥塞控制机制,自动适应网络状况的变化。算法的改进表现于拥塞避免阶段,此部分算法的主要思想和伪代码可整理如下:

```
当  $\alpha \leq \delta \leq \beta$  时
  若(新的吞吐量 > 上一个 RTT 内的吞吐量)
    则有:  $cwnd = cwnd + 1$ ;  $\alpha = \alpha + 1$ ;  $\beta = \beta + 1$ ;
  若(新的吞吐量 <= 上一个 RTT 内的吞吐量)
    则令: 窗口增量为零;
当  $\delta < \alpha$  时
  若( $\alpha > 1$  && 新的吞吐量 > 上一个 RTT 内的吞吐量)
    则有:  $cwnd = cwnd + 1$ ;
  若( $\alpha > 1$  && 新的吞吐量 <= 上一个 RTT 内的吞吐量)
    则有:  $cwnd = cwnd - 1$ ;  $\alpha = \alpha - 1$ ;  $\beta = \beta - 1$ ;
  若( $\alpha == 1$ )
    则有:  $cwnd = cwnd + 1$ ;
当  $\delta > \beta$  时
  则有:  $cwnd = cwnd - 1$ ;  $\alpha = \alpha - 1$ ;  $\beta = \beta - 1$ ;
其它情况时
  则令: 窗口增量为零;
```

3 本文对 Vegas 算法的改进

3.1 Vegas 1 算法

Vegas 1 算法的改进思想源于 Vegas A 算法, α 和 β 值也可以自动调整,但是拥塞窗口的调整策略与 Vegas A 算法有所不同。改进的目标是使算法更加均衡平稳,具有更大的自适应性。关于 Vegas 1 算法在卫星网络内的性能表现,作者已另行撰文详细探讨;本文集中考察 Vegas 1 算法在自组网中的性能。

改进措施之一是在慢启动阶段,每一个 rtt 时间内拥塞窗口调整一次,希望算法的拥塞控制反应更加灵敏。

改进措施之二是避免拥塞窗口增加过于冒进,在拥塞避免阶段,将 Vegas A 算法中所有涉及 $cwnd + 1$ 的行为改变为 $cwnd + 1 / cwnd$ 。仿真结果表明,修改后算法的 RTT 很小;但是由于太谦逊,使得吞吐量下降较多。

为进一步改善性能,并设法将吞吐量提高,采取的措施是不要将窗口降得太草率,每当需要降低窗口时仅仅降低 $1 / cwnd$,同时减少参数降低的次数。实现此目标的算法伪代码如下:

```
当  $\alpha \leq \delta \leq \beta$  时
  若(新的吞吐量 > 上一个 RTT 内的吞吐量)
    则有:  $cwnd = cwnd + 1 / cwnd$ ;  $\alpha = \alpha + 1$ ;
     $\beta = \beta + 1$ ;
  若(新的吞吐量 <= 上一个 RTT 内的吞吐量)
    则令: 窗口增量为零;
当  $\delta < \alpha$  时
  若( $\alpha > 1$  && 新的吞吐量 > 上一个 RTT 内的吞吐量)
    则有:  $cwnd = cwnd + 1 / cwnd$ ;
  若( $\alpha > 1$  && 新的吞吐量 <= 上一个 RTT 内的吞吐量)
    则有:  $cwnd = cwnd - 1 / cwnd$ ;
  若( $\alpha == 1$ )
    则有:  $cwnd = cwnd + 1 / cwnd$ ;
当  $\delta > \beta$  时
  若(新的吞吐量 <= 上一个 RTT 内的吞吐量-)
    则有:  $cwnd = cwnd - 1$ ; if( $cwnd < 2$ ) {  $cwnd = 2$  };
     $\alpha = \alpha - 1$ ;  $\beta = \beta - 1$ ;
  若(新的吞吐量 > 上一个 RTT 内的吞吐量)
    则令: 窗口增量为零;
```

3.2 Vegas 2 算法

Vegas 2 算法的改进思想直接源于 Vegas 算法,拥塞避免阶段的算法仍分为三大类情况,但是每类里又有一些细化处理。

```
当  $\alpha \leq \delta \leq \beta$  时
  若(新的吞吐量 > 上一个 RTT 内的吞吐量)
    则有:  $cwnd = cwnd + 1$ ;
  若(新的吞吐量 <= 上一个 RTT 内的吞吐量)
    则令: 窗口增量为零;
当  $\delta < \alpha$  时
  则有:  $cwnd = cwnd + 1$ ;
当  $\delta > \beta$  时
  若(新的吞吐量 <= 上一个 RTT 内的吞吐量-)
    则有:  $cwnd = cwnd - 1$ ; if( $cwnd < 2$ ) {  $cwnd = 2$  };
  若(新的吞吐量 > 上一个 RTT 内的吞吐量)
    则令: 窗口增量为零;
```

4 仿真网络环境及仿真结果

4.1 仿真网络环境

仿真工具采用 NS2 仿真软件^[11]。

仿真环境设置如下:

物理层:在 $500m \times 500m$ 的区域内,设置 20 个移动结点;移动结点的平均移动速度由 setdest 工具随机生成场景时确定,共取八种。每个平均速度的场景要重复生成 20 次以平滑偶然因素的影响,然后对 20 种仿真结果取其平均值。无线传播模型使用 TwoRayGround,每个结点的有效通信距离为 250 米。

数据链路层:利用 NS 系统自带的 802.11 协议。

网络层:路由算法选为 DSDV 路由算法。

传输层:将 20 个移动结点分成 10 对,建立 10 个独立的 TCP 连接,每个 TCP 包的大小为 512 Byte。我们分别仿真比较了六种拥塞控制协议,分别是 Reno, SACK, Vegas, Vegas1, Vegas2, Vegas A 协议。

应用层:选择 FTP 作为传输层的数据来源。

仿真运行时间:100 秒。

4.2 仿真结果

算法性能比较以 TCP 连接上的吞吐量为依据。计算方法是统计 100 秒内全网络总的 TCP 包接收数目,然后除以 10 即得每个连接的平均 TCP 包接收数目。再考虑到包的大小和运行时间,得出平均每个连接的吞吐量;此值再对 20 个场景平均,就得到一种平均移动速度下每个连接上的平均吞吐量。最后以平均移动速度为横坐标,吞吐量为纵坐标,画出各种拥塞控制算法的吞吐量——平均速度曲线图。

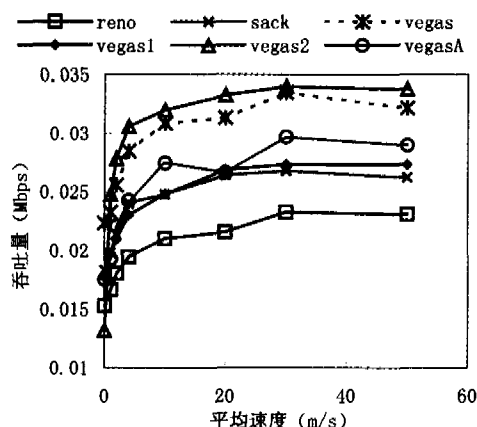


图1 吞吐量随平均速度的变化曲线

另外,为更深入地分析各种算法的优劣,还可以比较各种拥塞控制算法的传包效率,即收到 TCP 包的数目与发送 TCP

包的数目之比。分析这个比值的目的是为了更加有效地利用自组网珍贵的带宽资源,因为该值的大小反映了网络当中传输的废弃 TCP 包数量。虽然反映算法性能的最重要参数是吞吐量,但是传包效率也是必须考虑的指标。例如,如果一个算法是以传两个包丢一个包的代价最大限度地攫取带宽,那么该算法在自组网当中是不宜采用的。

最后得到的仿真运行结果可以总结成图 1、图 2 两个图来表示。

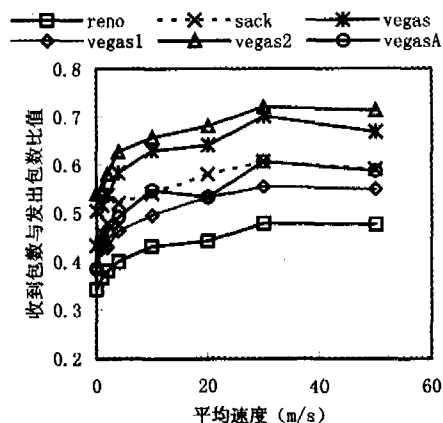


图 2 传包效率随平均速度的变化曲线

5 结果分析与结论

从以上两个图中可见,无论处于什么样的平均移动速度场景下,所仿真的六种算法之中 Vegas2 算法的性能都是最优的。

(1) TCP 包的大小影响拥塞控制算法的实施效率。这是自组网比较特殊的一个性质。原因在于结点移动频繁,较大的数据包不易传送成功,较小的数据包容易较快地传送到目的地,受拓扑变化的影响相对少些,因此易于传送成功。实验中发现,Vegas 算法对于 TCP 包的大小较为敏感,65536Byte 的 TCP 数据包使 Vegas 算法无法工作,包接收率极低;而 512Byte 的 TCP 包却又使得 Vegas 算法性能达到最优。其它几个算法也不同程度的受此影响,TCP 数据包较大时性能严重下降。

(2) Vegas1 算法和 Vegas A 算法在自组网中的性能不佳。在卫星网络中,这两个算法的表现不俗,原因是卫星网络的 RTT 很大,网络的规模及容量较大,因此参数 α 和 β 的变化是有效的,这种变化更多地挖掘出了卫星网络的潜能。而在自组网这样一个小容量小延迟网络内, α 和 β 的变化却阻碍了拥塞控制的有效实施。因此 Vegas1 算法

和 Vegas A 算法的性能反而逊于简洁的 Vegas 算法。

(3) Vegas2 算法在自组网中性能最优。繁简适度是其出色表现的基础。一方面,该算法没有 Vegas1 算法和 Vegas A 算法繁复, α 、 β 值固定为 1 和 3 不变,适应了自组网的浅容量和小延迟特点,变化快捷;另一方面,Vegas2 算法又较 Vegas 算法细致,更好地刻画了自组网实际的网络变化特征,因而 Vegas2 算法表现出了较好的网络拥塞控制能力。

(4) 平均速度对拥塞控制算法的影响有限。从两个仿真结果图可见,只有当结点的平均移动速度达到 50m/s 时,算法的效果才受到影响。本文的这个结果相信是由于结点移动区域设定较小和由此导致的通信跳数过少所致,今后的研究将深入探讨此问题。

参考文献

- 1 Al Hanbali A, Altman E, Nain P. A Survey of TCP over Mobile Ad Hoc Networks. <http://www-sop.inria.fr/maestro/personnel/Ahmad.Al-Hanbali/PUBLICATIONS/RR-5182.pdf>, 2004, 5
- 2 周建新,邹玲,石冰心. 无线网络 TCP 研究综述. 计算机研究与发展, 2004, 41(1): 53~59
- 3 姜海,程时昕,陈相宁. 无线 Ad Hoc 网中的 TCP SACK 与 TCP Vegas. 电路与系统学报, 2001, 6(1): 35~39
- 4 Holland G, Vaidya N. Analysis of TCP performance over mobile ad hoc networks. ACM Wireless Networks, 2002, 8(2): 275~288
- 5 Liu J, Singh S. ATCP: TCP for mobile ad hoc networks. IEEE Journal on Selected Areas in Communications, 2001, 19(7): 1300~1315
- 6 Kim D, Toh C K, Choi Y. TCP-BuS: Improving TCP performance in wireless ad hoc networks. Journal of Communications and Networks, 2001, 3(2): 1~12
- 7 夏巍,林亚平,李超. 基于专家控制的无线网络拥塞控制机制的研究. 通信学报, 2004, 25(1): 164~173
- 8 Brakmo L S, Peterson L L. TCP Vegas: End to End Congestion Avoidance on a global Internet. IEEE Journal on Selected Areas in Communications, 1995, 13(8): 1465~1480
- 9 罗万明,林闯,阎保平. TCP/IP 拥塞控制研究. 计算机学报, 2001, 24(1): 1~18
- 10 Srijith K N. Improving the Performance of TCP Vegas and TCP SACK: Investigations and Solutions. <http://www.srijith.net/publications/srijith-MSc-Thesis.pdf>, 2002
- 11 UCN/LBL/VINT. Network simulator-NS2. <http://www-mash.cs.berkeley.edu/ns>, 1995