分类号：TP915.3　　单位代码：10422
密　级：　　学　　号：200210724

山东大学

硕士学位论文

Shandong University Master's Thesis

论文题目：ad hoc 网络环境下的 TCP 性能分析及改进

作 者 姓 名　王伟伟
专　　业　通信与信息系统
指导教师姓名
专业技术职务　张晓敏 副教授

2005 年 4 月 22 日

原 创 性 声 明

本人郑重声明：所呈交的学位论文，是本人在导师的指导下，独立进行研究所取得的成果。除文中已经注明引用的内容外，本论文不包含任何其他个人或集体已经发表或撰写过的科研成果。对本文的研究作出重要贡献的个人和集体，均已在文中以明确方式标明。本声明的法律责任由本人承担。

论文作者签名：王伟伟　　日　期：2005.4.22

关于学位论文使用授权的声明

本人完全了解山东大学有关保留、使用学位论文的规定，同意学校保留或向国家有关部门或机构送交论文的复印件和电子版，允许论文被查阅和借阅；本人授权山东大学可以将本学位论文的全部或部分内容编入有关数据库进行检索，可以采用影印、缩印或其他复制手段保存论文和汇编本学位论文。

(保密论文在解密后应遵守此规定)

论文作者签名：王伟伟　导师签名：张吹亚　日　期：2005.4.25

摘 要

ad hoc 网络是一种自组织网络，它不用任何通信基础设施的支持就可在几个移动节点间通信。由于 ad hoc 网络具有组网灵活，可有效利用资源等特点，它在军事通信、应急通信、商业应用环境等方面将有广阔的应用前景。然而同时由于其面临高误码率、电池容量有限、易被监听等问题，需要对其不断地进行改进和完善。

ad hoc 网络作为一种新型网络，还存在着很多问题，其大规模的实现、实用化还有很多的研究工作要做。由于 ad hoc 网络中的多跳无线传输和网络拓扑变化，TCP 性能较之有线网络严重恶化。如何解决 TCP 应用于 ad hoc 网络时面临的性能恶化是当前亟待解决的问题。现在提高 ad hoc 网络的 TCP 性能的主要研究方向集中在对 MAC 层、网络层、传输层机制的改进，还有的改进在原有体系结构的基础上又增加了一层。

TCP 的核心是拥塞控制机制。目前的 TCP 版本主要有 Tahoe、Reno、NewReno、SACK 和 Vegas 几种，还有一种针对无线网络的改进被称为 DA。Tahoe TCP 的拥塞控制机制是分三个阶段实现的，即慢启动、拥塞避免、快速重传。后几种 TCP 均是基于 TCP Tahoe 的，只不过都进行了改进，使 TCP 能更有效的运行或者适合某种特定的网络状态。Reno TCP 是当前在互联网中使用的最广泛的 TCP 版本。它在 Tahoe 的基础上增加了快速恢复阶段。NewReno TCP 是对 Reno TCP 做了小的改进，它解决了 Reno TCP 在快速恢复阶段多包丢失的问题。选择确认（SACK）通过接收方对数据进行选择性确认来提高 TCP 的性能，它能较好地解决一个窗口中丢失多包的问题。DA 只是接收方的机制，它是专门为共享信道的无线网络做的改进。Vegas TCP 有着完全不同于前五种 TCP 的拥塞控制机制。Vegas 主要应用三种技术来提高吞吐量减少丢失。

由于 TCP 是为有线网络设计的，它有效运行的前提是假设丢包都是由于网络拥塞造成的。TCP 把包的丢失作为网络发生了拥塞的标志信号，每检测到有包丢失，TCP 就启动拥塞控制机制，减少发送端的发送速率，从而减轻了拥塞程度。但在 ad hoc 网络中，包的丢失大多不是由网络拥塞造成的。无线信道的高误码率、节点移动造成的路由中断、MAC 层的冲突等等都会造成包的丢失，这时 TCP 不

必要的启动拥塞控制，使得TCP的吞吐量显著下降。为了使TCP更好的适应ad hoc网络，需要进行改进。

NS 是 Network Simulator 的首字母缩写，它是由 LBNL(Lawrence Berkeley National Laboratory)的网络研究小组开发的仿真工具。NS 是一种可扩展、易配置、可编程的事件驱动的网络仿真软件。本文中的仿真都是基于NS2 的。首先对 ad hoc 网络中各种 TCP 版本的性能进行了仿真，并分析了 ad hoc 网络的多跳和移动特性对 TCP 性能的影响，并比较了各种版本的 TCP 性能，找出了各种条件下最适合于 ad hoc 网络的 TCP。接着对适合无线网络的 DA 机制应用于 ad hoc 网络的性能表现做了进一步的分析并提出了对其的改进。然后对已发表的文献中的各种已提出的改进 ad hoc 网络中 TCP 性能的方案进行了总结并做出了自己的评价，其中包括对 MAC 层的改进，对网络层的改进，对传输层的改进，综合的改进以及其它方面改进。当前的改进大多都集中在网络层以上，且改进的方案大多都增加了算法的复杂性，这对能源有限的移动节点很不利。多跳 ad hoc 网络中 TCP 性能下降的主要原因是 MAC 层的暴露站问题和二进制退避机制引起的 TCP 的不稳定性，因此本文对 ad hoc 网络的 MAC 层进行改进，首先是针对 ad hoc 网络中由于多跳特性造成的 TCP 的不稳定性提出了自己的改进，并仿真并分析了改进结果。然后对多跳 ad hoc 网络中存在的信道锁定造成的不公平性提出了自己的改进方案，并用仿真分析验证了此改进不但可以改进 ad hoc 网络中有多条连接时的不公平性问题，而且还可以改善 TCP 的不稳定性问题，提高了吞吐量，这两种改进都比较简单，没有增加移动节点的能源消耗和网络负荷，适合移动性不大的 ad hoc 网络。

关键词： ad hoc 网络，TCP，拥塞控制，NS

ABSTRACT

Ad hoc network is a self-organized system independent of communication infrastructure. On one hand, ad hoc networks have foreground in military communication, urgency affairs and commercial environment in the future because of its virtues such as flexibility and high efficiency in utilizing the resources. On the other hand, ad hoc networks need to be improved because of its high bit error rate, poor security and so on.

As a new network, ad hoc network still has many problems. A lot of work is to be done for the realization and utility of ad hoc network. The performance of TCP degrades seriously because of the transmission through multi-hop wireless media and the transformation of the network topology. How to solve the problem of TCP is one of the main problems facing to ad hoc networks. At present the researches mainly lies on the modification of MAC, Network, and Transport layer. There also is a scheme that adds a new layer on the base of old architecture.

The core of TCP is congestion control mechanism. TCP has different variants, which are Tahoe, Reno, NewReno, SACK, Vegas. There also is an improved scheme for wireless networks called DA. The congestion control of Tahoe TCP includes three phases: Slow Start, Congestion Avoidance and Fast Retransmission. The latter five TCP are all based on Tahoe, with modification to which. Those modifications make TCP run more efficiently. Reno is the most popular version used in the networks. It adds a Fast Recovery phase to Tahoe TCP. NewReno made a small modification to Reno, which can solve the multi-loss problem in the Fast Recovery phase. SACK (Selective ACK) can improve the performance of TCP through receiver selectively acknowledging the data. DA is the mechanism of TCP receiver, which makes TCP accommodate to the wireless networks. Vegas has a quite different congestion mechanism, which has mainly three technologies to decrease losses and increase the throughput.

As we know, TCP is designed for wired networks, and it assumes that any loss is

caused by congestion happening in the network. When a loss is detected, TCP will invoke congestion mechanism to slow the sending rate at the sender and thus alleviate the congestion. But in ad hoc networks, most loss is not caused by the congestion, but by the high bit error rate of the wireless link, the route breakage caused by mobility of the nodes, the collision in the MAC layer and so on. Therefore the unnecessary congestion control will degrade the performance of TCP. Therefore TCP needs to be improved to adapt to the ad hoc networks.

NS(Network Simulator) is developed by the network research group of LBNL (Lawrence Berkeley National Laboratory). NS is an extendable, configurable and programmable network simulation tool. All the simulations in the paper is based on NS2. First, I simulate different TCP variants in order to know the effect of multi-hop and mobility of the nodes on the performance through analyzing the results of the simulations of the performance of TCP over ad hoc networks. The comparisons of different TCPs are also made to find the best TCP variant under various network conditions. Second, I analyze the performance of DA and improve it. Then I generalize the proposed improved TCP schemes and evaluate them, which include modifications on MAC, network, transport layers and so on. Most of the proposed schemes centered on the layers above network layer, and nearly each of them add complexity to arithmetic, which is disadvantageous to mobile node. The main reason that results in degrading of TCP performance is that exposed node and back off mechanism of MAC layer. Therefore I make improvement on the MAC layer. First I propose a new scheme to solve the instability of TCP over multi-hop ad hoc networks and verify it by simulation results. Second, I Aim at to solve the unfairness of at least two TCP connections over ad hoc networks. The simulation results demonstrate that it not only improved the fairness, but also alleviate the instability of TCP, enhancing the performance. Both of the two schemes are simple without increasing the consumed power and the spending of the network. The improved schemes work well in the less mobile ad hoc networks.

Keywords: ad hoc networks; TCP; congestion control; NS

引言

目前，很多研究致力于开发基于自组织（self-organization）的新一代移动通信技术和信息服务。随着 ad hoc 网络技术的出现，这种系统将会成为最热门的话题。实际上，许多基本的技术问题还未解决，真正的商业应用仍有待于进一步的实践。

移动 ad hoc 网络始于 20 世纪 70 年代，至今已经有成熟的路由、移动 IP、组播等技术，并能够使用基于公开密钥系统（RSA）的数字签名来确保网络的安全性，但仍有一些问题没有解决。

目前从事 ad hoc 网络研究的机构主要有 Internet 工程任务组 IETF、IEEE 及 DARPA（国际高级研究计划局）。IETF 于 1997 年成立了专门的研究组——ad hoc 网络组，针对 ad hoc 网络开发基于 IP 协议的路由机制并解决与网络层相关的技术问题。1999 年 1 月，RFC 2501 详细给出了 ad hoc 网络的应用场合、特征和性能要求，并在 2000 年下半年公布了一系列移动 ad hoc 网络路由协议草案，如 DSDV、AODV、TORA、DSR、OLSR 等。IEEE 通信分会在 2000 年底成立了专门的移动 ad hoc 技术分委员会。美国 DARPA 资助的 SUO SAS(Small Unit Operation Situation Awareness System)系统工作在 20MHz~2.5GHz 的频段，带宽为 500kHz~20MHz，自适应数据率为 16kbit/s~4Mbit/s。高移动环境下的由 100 个实验单元组成的现场实验已于 2002 年春开始。此外，瑞士联邦工学院 Termin Nodes 项目组正在研究和实现大规模移动自组网。我国对移动 ad hoc 网络的研究和开发起步较晚，目前专门从事移动 ad hoc 网络研究的科研机构不多，与国外的研究还有较大差距，且处于理论探讨阶段，尚未形成任何专利成果。

由于移动 ad hoc 网络具有使用灵活、组网迅速、鲁棒性强的特点，可以广泛应用于军事作战、突发事件、家庭、学校及公司等多种领域，具有巨大的市场潜力。

在军事上，ad hoc 网络已成为一种对抗敌方有意破坏行为的有效办法。由于各节点的对等性，网络中不存在目标突出、易于暴露、易受攻击的中心，敌方不易发现我指挥中心，难以进行摧毁性的攻击。1981 年美国就为海军特遣部队开发了一种高频自组网——HF-ITF。1989 年又投巨资研发的士兵综合作战系统——

“陆地勇士”就是 ad hoc 网络技术的成功应用，并在阿富汗反恐战争中得到成功的应用。

在民用方面，ad hoc 网络技术可用于抢险救灾等紧急突发事件中，通过快速组建 ad hoc 网络，即时传输险情、位置等第一手资料。Ad hoc 网络可以在特定地点提供交互式服务，如在博物馆，可以通过 ad hoc 网络实现购买电子门票、认证电子门票、馆内全程导游等服务；通过 ad hoc 网络还可以在公司各个部门间快速建立无线网络会议，方便地实现与会者的多方协作。

为了将基于 TCP 协议的上层应用程序无缝移植到移动 ad hoc 网络中去，同时将 ad hoc 网络与传统 Internet（主要以 TCP 为传输层协议）进行无缝融合，研究移动 ad hoc 网络中的 TCP 性能非常必要。

TCP 是一种面向连接、提供可靠字节流服务的网络传输层协议，其核心是拥塞控制机制。由于 TCP 是为有线网络设计的，它有效运行的前提就是网络中由于信道而发生的丢失很少，几乎可以忽略。TCP 把丢失作为发生了拥塞控制的指示，每当检测到有丢失时，便启动拥塞控制机制。目前的 TCP 版本主要由 Tahoe、Reno、NewReno、SACK、Vegas 几种，其中 Tahoe 是最早的版本，Reno、NewReno、SACK 均基于其上并有所改进。Vegas TCP 有着完全不同的拥塞控制机制。其中 Tahoe TCP 通过慢启动、拥塞避免和快速重传实现了拥塞控制；Reno 在此基础上又实现了快速恢复；NewReno 又解决了 Reno 在快速恢复阶段多包丢失的问题；SACK 通过选择确认来提高 TCP 的性能；Vegas 主要应用三种技术来提高吞吐量减少丢失。

ad hoc 网络作为一种新型网络，存在着很多问题，其大规模的实现、实用化还有很多的研究工作要做。由于 ad hoc 网络中的多跳无线传输和网络拓扑变化，TCP 性能较之有线网络严重恶化。如何解决 TCP 应用于 ad hoc 网络时面临的性能恶化是当前亟待解决的问题。现在提高 ad hoc 网络的 TCP 性能的主要研究方向主要集中在对 MAC 层、网络层、传输层机制的改进，还有的改进在原有体系结构的基础上又增加了一层。对 MAC 的改进可减轻 TCP 的不稳定性问题，但没有完全消除不稳定性的影响，且已提出的对 MAC 层的改进方案较其它层少。对 MAC 层以上层次的改进虽然可提高 TCP 的吞吐量，但都会增加网络开销和复杂性。未来的研究方向是根据网络实际情况，在复杂度和吞吐量之间更好地折衷。我们有理由相信随着研究的深入，困扰 ad hoc 网络的问题终将得到解决，ad hoc

网络将有更加广阔的前景。

本文试图通过不同仿真情形来进一步分析TCP不同版本在ad hoc网络中应用时的性能表现及其原因，并在此基础上提出改进机制并加以验证。

第一章 无线 ad hoc 网络简介

无线 ad hoc 网络是由一组自治的无线节点或终端相互合作而形成的独立于固定的基础设施并采用分布式管理的网络，是一种自存在、自组织和自管理网络。这些节点都是可移动的，可以动态地创建一个无线局域网。

与传统无线网络不同的是，ad hoc 网络没有固定的网络体系结构和管理支持，所有节点分布式运行，当移动节点加入或离开网络时，网络拓扑结构会自动改变。所有节点具有路由器功能，负责发现和维护到其他节点的路由，向邻节点发送或转发包。

根据节点是否移动，可以将无线 ad hoc 网络分为移动 ad hoc 网络和传感器网络。在移动 ad hoc 网络中，各个节点都可以自由移动。事实上，很多文献常常把无线 ad hoc 网络等同于移动 ad hoc 网络。在传感器网络中，各个无线节点静态的随机分布于某一区域。传感器负责收集区域内的声音、电磁或地震信号等多种信息，将他们发送到网关节点。网关具有更大的处理能力，能够进一步处理信息，或有更大的发送范围，可以将信息发送至某个大型网络，使远程用户能够检索到该信息。本文中的 ad hoc 网络主要是指移动 ad hoc 网络。

1.1 无线 ad hoc 网络的发展[1]

无线 ad hoc 网络的历史可以追溯到 1972 年由美国国防部发起的分组无线网络(Packet Radio Network, PRNET)项目，到了 80 年代演化为生存自适应网络(Survival Adaptive Radio Network, SURAN)项目。这些项目为战场上的士兵、坦克、飞机等移动节点提供了基于自组织、有较强生存能力和抗干扰能力的分组无线网络。

90 年代初，无线 ad hoc 网络得到了长足发展，主要是因为笔记本电脑的大量应用和各种基于无线和红外技术通信设备的广泛出现，为无线 ad hoc 网络的应用提供了广阔的空间。90 年代中后期以来无线 ad hoc 网络的标准化活动推动了它的发展，其中 IETF(Internet Engineering Task Force)的移动 ad hoc 网络(MANET)工作组负责对路由协议进行标准化工作。IEEE 802.11 工作组对基于冲突避免的媒体接入协议进行了标准化，部分解决了隐藏节点问题，使利用笔记本电脑上的 802.11 PCMCIA 无线网卡组建无线 ad hoc 网络成为可能。

1.2 无线 ad hoc 网络的优点[2]

（1）灵活性

无线 ad hoc 网络支持随时随地的通信，并允许自发组建移动网络。同时因为无线 ad hoc 网络不需要基础设施，所以网络建设的成本会大大降低，布网也将变得非常迅速，这对应付突发事件和缓解热点地区的通信压力十分有利。

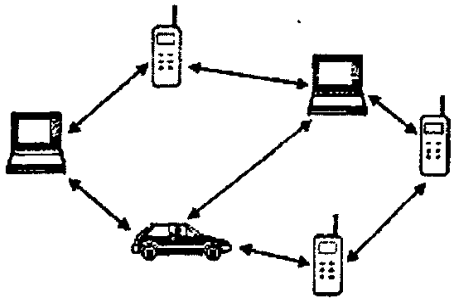


图 1.1 无线 ad hoc 网络结构

（2）抗干扰

无线 ad hoc 网络是一个多跳的网络，因此通信过程可能需要若干次中继才能完成，而中继节点可以灵活地选择路由，避开干扰严重的地区，因此，相对于传统的蜂窝结构而言，无线 ad hoc 网络可以在一定程度上消除建筑物阴影造成的影响，同时可以解决城区周边地区信号质量差的问题。

（3）有效的资源利用

无线 ad hoc 网络可以更有效的利用无线资源。由于每个用户的覆盖范围变得非常有限，这样频率的复用也成倍地提高了，从而增加了容量。

1.3 无线 ad hoc 网络面临的问题[3]

无线 ad hoc 网络面临的问题有：

（1） 有限覆盖范围问题

一方面，较短的传输距离使路由的更新跟不上移动带宽所带来的拓扑结构的变化；另一方面，过多的中继可能使路由变得很脆弱。

（2） 无线媒质的广播特性

无线媒质带来的问题常见的有隐藏节点和暴露节点问题。

（3） 传输错误带来的丢包

无线传输的差错率要远远高于有线传输，TCP 这样的高层协议在这样的链路上工作，效率会大大降低。

（4） 移动带来的路由改变

路由的改变对无线 ad hoc 网络的影响比对传统蜂窝要大。目前的路由协议在这方面还不很健全。

（5） 移动带来的丢包

由于移动带来的路由改变而不得不重新选路，以及设备存储容量的限制等原因，无线 ad hoc 网络中的节点在移动过程中的丢包率会明显增加。

（6） 电池的限制

移动设备的电池容量十分有限，限制了设备的复杂度，从而对移动设备组成的无线 ad hoc 网络也产生了一定的影响。

（7） 潜在的频繁网络分区问题

无线 ad hoc 网络在工作过程中，可能会形成若干分区，频繁的分区会对网络的连接造成影响。

（8） 安全性问题

采用中继方式，使数据很容易受到第三方的监听。

1.4 移动 ad hoc 网络的应用

移动 ad hoc 网络潜在的应用很多，可分为以下几类：

（1）移动会议：在室外临时环境中，工作团体的所有成员可以通过 ad hoc 方式组成一个临时网络来协同完成一项大的任务，或协同完成某个计算任务。在室内办公环境中，办公人员携带的包含 ad hoc 收发器的 PDA 可以通过无线方式自动从台式机上下载电子邮件，更新工作日程表。

（2）家庭联网：通过移动联网的方式把办公室的办公环境延伸到家庭，必要时在家庭办公。或者利用我们随身携带的个人无线 ad hoc 设备与装备了 ad hoc 收发器的家庭电器通信，自动完成开锁、开灯、打开娱乐设备、调节空调等操作。

（3）紧急服务：由于停电或其他灾害出现，网络基础设施遭到破坏时，组建一个 ad hoc 网络帮助紧急救援人员完成必要的通信工作。

（4）传感器网络：最近，人们开始关注大量分布的传感器协调工作问题，传感器可以工作在危险的环境（如化学有害物质泄漏现场），通过在传感器上装备位置指示器、ad hoc 收发器等，将传感器所在现场的信息传送到危险现场以外，避免救援人员进入现场，收集和辨别事故信息。

（5）个人与网络：通过 ad hoc 网络把个人通信、娱乐、办公等设备联网，这些

设备可以或不需要与 Internet 相连，但在执行用户的某项活动时肯定需要彼此通信，在这种情况下，移动性不是主要问题。

（6）军事无线通信：在现代化战场上，各种军事车辆之间、士兵之间、士兵与军事车辆之间都需要保持密切的联系，以完成集中统一指挥，协调作战。这样的通信网络是一种典型的 ad hoc 网络。据报道，在最近的伊拉克战争中，移动 ad hoc 网络得到有效的应用。

（7）其它商业应用：如未来装备 ad hoc 收发设备的机场预约和登机系统可以自动地与乘客携带的个人无线 ad hoc 设备通信，完成目前的换登机牌等手续。如商场内商品 RF 标签，廉价的 RF 标签可以通过无线接口由 ad hoc 设备动态刷新。顾客若携带手持设备可以很容易地找到某种商品和价格。

1.5 无线 ad hoc 网络的发展前景

无线 ad hoc 网络作为一种新颖的移动计算机网络，它既可以作为一种独立的网络而存在，也可以作为当前固定基础设施网络的一种补充形式。它的发展趋势主要为：（1）民用领域将逐步扩大。从 ad hoc 技术开始民用以来，也由于一些 ad hoc 的产品面世，基于 ad hoc 技术的移动终端以及网络设备已经成为一些大通信公司研发的重点。（2）ad hoc 网络的应用范围逐步扩大。目前的 ad hoc 网络的某些技术主要是针对特定的环境，因此要提出一种能适应于任何环境下的相关协议。另外 ad hoc 网络在军事通信、应急通信、商业应用环境等方面将有广阔的应用前景。

第二章 TCP 的拥塞控制

TCP 是事实上的传输层标准协议[4]。TCP 是一个面向连接的端到端的可靠的协议，是为了适应支持多网络应用的多层次协议设计的。TCP 为连接在不同但相互连接的计算机通信网络上的主机计算机的对等进程提供可靠的进程互操作通信。TCP 很少依赖下层的通信协议的可靠性。TCP 总是假设它可能从低层协议获得的为低可靠性的数据报服务。原则上，TCP 能够在分组交换或电路交换等较广范围内的通信系统上进行操作。

TCP 的核心是拥塞控制机制。由于路由器的流量过载引起网络拥塞，即数据流量超过了路由器的缓冲处理能力时，即是发生了拥塞，这时路由器就会丢弃数据报文段。发生网络拥塞以后，除了重传丢失的数据报文段以外，TCP 还要降低它的传输速率，以减少网络中的数据流量，从而使得路由有足够的时间来处理转发数据报文段。然后 TCP 逐渐增加传输速率来探测网络的容量，以避免拥塞的再次发生。

目前在 Internet 中实际使用的拥塞控制基本是建立在 TCP 的窗口控制基础之上的。网络层的路由器所起的作用比较小。TCP 基于窗口的端到端的拥塞控制对于 Internet 的稳定性起到了关键性作用。TCP 协议使用一种加性增加乘性减少 AIMD(additive increase multiplicative decrease)的拥塞控制算法[5]。两个主机之间使用 TCP 传送的数据单元叫做报文段，其中对数据报文段的确认为 ACK。发送方维持着一个拥塞窗口，发送方的发送速率是根据其窗口大小来调控的：当发送方发现窗口内的一个报文段发生丢失，则认为这个丢失是由网络拥塞造成的，随后发送方就将窗口减小，以减小发送速率，从而避免拥塞的加重；如果这个窗口中的报文段没有发生丢失，则表明目前网络状况良好，发送方将窗口尺寸加大，进而增大了报文段的发送速率。基本的 TCP 拥塞控制是通过控制一些重要参数的改变和 3 个阶段来实现的。

TCP 用于拥塞控制的主要参数有：拥塞窗口（cwnd），通告窗口（awin），发送窗口（win），慢启动阈值（ssthresh），往返时间（RTT），超时重传计数器（RTO）和快速重传阈值（tcprexmtthresh）。

（1）拥塞窗口（cwnd）：一个 TCP 状态参量，代表着一个 TCP 允许发送的最大

数据量。在任意一个给定的时刻，TCP 不会发送序号大于最大确认序号和 cwnd、awin 中较小者的数据。

（2）接收端通告窗口（awin）：接收方能处理的最多的数据数量。

（3）发送窗口（win）：去 cwnd 和 awin 中较小的一个，决定着发送速率。

（4）慢启动门限（ssthresh）：窗口指数增加的最大值。

（5）往返时间（RTT）：从报文段发送到接收到对其的确认的时间间隔的平均值。

（6）重传超时（RTO）：从报文段发送后可认为段丢失的时间间隔。

（7）快速重传阈值（tcprexmtthresh）：可认为是有数据丢失时所收到的重复的 ACK 的数目。本文中为 3。

到目前为止，TCP 主要有 Tahoe、Reno、NewReno、SACK、vegas 和 TCP+DA 这几种。其中 Tahoe 的拥塞控制是最基本也是最原始的机制，Reno, NewReno, SACK, TCP+DA 均基于 Tahoe 上，并在其上有所改进。而 Vegas TCP 则有着一种完全不同的拥塞控制机制。下面分别介绍。

2.1 Tahoe 拥塞控制机制[6]

Tahoe 的拥塞控制主要有三个阶段：慢启动、拥塞避免、快速重传。

2.1.1 慢启动阶段

由于端系统不能预测网络资源的使用情况，如果在建立一个新的连接时向网络中发送大量的数据包，容易导致耗尽路由器的资源，使网络吞吐量急剧下降。为了避免新建立的连接产生的突发通信量超过网络的承受能力，TCP 使用了慢启动算法来逐渐探测网络的可得带宽。当建立新的 TCP 连接时，首先拥塞窗口被初始化为一个报文段的大小（缺省值为 512 或 536 字节）。发送方每收到一个 ACK 确认时，将 cwnd 增加一个报文段的发送量，这样，cwnd 随 RTT 呈指数级增长：1 个，2 个，4 个，8 个...，直到拥塞窗口 cwnd 达到慢启动门限 ssthresh。其实慢启动这个名词完全误导了，因为这个过程一点都不慢，且这时 cwnd 是指数增长的。发送方向网络中发送的数据量将急剧增加。

2.1.2 拥塞避免阶段

当 cwnd 达到慢启动门限 ssthresh 时，TCP 就进入了拥塞避免阶段。这时当一个窗口中的所有的报文段都得到了相应的确认时，cwnd 才加 1。直到 cwnd 到达了接收方通告窗口 awin，或有报文段丢失了。

2.1.3 快速重传

TCP 有两种方法来检测丢失，一种是当从报文段发送开始到了重传计时器超时，即时间超过了 RTO，都没有收到相应的确认，就认为报文段丢失了；另一种方法是收到 3 个重复的 ACK，也可认为序号为此 ACK 序号的报文段丢失了。对任何丢失的报文段，发送方都认为发生了网络拥塞，因此除了重传之外，设 ssthresh=min(cwnd/2, awin)；cwnd=1；重新进行慢启动，减少发送速率，避免拥塞加重。由于收到 3 个重复的 ACK 就可重传丢失的段，不用等待一个 RTO，所以这时的重传又称为快速重传。

以 Tahoe 为基础的基本 TCP 拥塞控制的算法如下。

初始化：

win=min(cwnd, awin)

cwnd=1;

ssthresh=65535bytes(缺省值)；/

当新确认包 ACK 到达时：

if(cwnd<ssthresh)

/Slow Start/

cwnd=cwnd+1；

else

/congestion avoidance/

cwnd=cwnd+1/cwnd;

超时：

ssthresh=max(2, min(cwnd/2, awin));

cwnd=1.

2.2 Reno TCP

Reno TCP 是当前在互联网中使用的最广泛的 TCP 版本。它的基本的拥塞控制机制与 Tahoe TCP 相同，只是做了一个小的改进。它在 Tahoe 的基础上增加了快速恢复阶段。当启动快速重传算法重传了丢失的段之后，如果 TCP 重新进入慢启动阶段，将会使拥塞窗口减为 1，重新进行探测网络带宽，从而严重降低了网络吞吐量，因此快速恢复算法在快速重传之后转去执行拥塞避免算法，避免了过大的减少发送窗口而导致的网络性能下降。

快速恢复算法和快速重传算法描述如下：如果连续收到三个重复的 ACK，设

慢启动门限 ssthresh 为当前窗口的一半，但不能小于 2 个报文段的大小，设 cwnd 为 ssthresh 加上三个报文段的大小；重传丢失的包；对再收到的每个重复的 ACK，cwnd 再加 1；如果收到了一个新的 ACK，设置 cwnd=ssthresh， 重新进行拥塞避免算法。

2.3 NewReno TCP[7]

NewReno TCP 是对 Reno TCP 做了小的改进，它解决了 Reno TCP 在快速恢复阶段多包丢失的问题。在 Reno TCP 的快速恢复阶段，若收到一个新的 ACK 就跳出快速恢复阶段，在 NewReno TCP 中若在快速恢复阶段收到的这个新的 ACK 没有对在此期间发送的所有数据进行确认，就称其为部分 ACK，NewReno 就认为在紧跟着这个部分 ACK 序号后的那个报文段丢失了，重传之。直到在快速恢复阶段发送的所有的报文段都得到了确认为止。这样就减少了超时重传的次数。

2.4 SACK TCP[8]

作为 TCP 的一个选项，选择确认（SACK）通过接收方对数据进行选择性确认来提高 TCP 的性能，它能较好地解决一个窗口中丢失多包的问题。当接收到的数据不连续时，接收方发送承载有 SACK 选项的确认（ACK）来通知发送方哪些数据已正确接收或正在缓存区中排队。一个 SACK 选项最多包括 3 个 SACK 块，每个块表示一组已经被接收到的序列号连续的报文段（每一段包含固定数量的字节），用左边界（即该块中第一个字节的序列号）和右边界（即最后一个字节的序列号+1）来标识，末尾序列号为左边界-1 的块和初始序列号为右边界的报文段则已丢失，需要重传。SACK 选项的第一块表示最近收到的一组连续报文段，其它的块则用于报告先前收到的一些连续段的信息。发送方可根据 SACK 选项信息只重传真正丢失的段，而不重传那些序号大于丢失段且已被正确接收的段。这大大减少了 TCP 的无谓重传，同时也使得发送方不用等待重传计时器超时就可以恢复多个丢失的数据。

2.5 TCP+DA（Delayed ACK）[9]

DA 只是接收方的机制，它是专门为共享信道的无线网络做的改进。DA 规定接收方收到两个满尺寸的报文段时，才发回一个 ACK。同时两个 ACK 之间的时间间隔不能超过一个门限值，否则重传那个重复的 ACK。由于 TCP 中的 ACK 为累积的，所以 DA 机制不会对 TCP 的可靠性有任何不好的影响。其发送方可以是任何版本的 TCP。

2.6 Vegas TCP[10]

Vegas TCP 有着完全不同于前五种 TCP 的拥塞控制机制。Vegas 主要应用三种技术来提高吞吐量减少丢失。第一种技术使重传丢失的包更加及时。第二种技术使 TCP 具有预知拥塞的能力，并且合理的调整它的传输速率。最后一项技术改变了 TCP 的慢启动机制以避免在初始时应用慢启动来探测可得带宽时丢失包。

2.6.1 更及时的重传技术

首先，Vegas 在每个包发出时读取并记录包的发送时间。当一个 ACK 到达时，Vegas 重读时钟，并用这个时间和先前的相关包的时间戳计算 RTT。Vegas 接着用这个更精确的 RTT 估计在下面两种情形下决定重传：

1）当一个重复的 ACK 到达时，Vegas 检查是否当前时间和相关包的时间戳之间的差值比超时值大，如果是，Vegas 不用等待收到 3 个重复的 ACK 就立即重传该包。

2）当一个新的 ACK 到达时，如果它是重传后的第一或第二个 ACK，Vegas 又会检查是否从该包发送到当前的时间大于超时值。如果是，Vegas 接着重传这个包，这甚至不用等一个重复的 ACK 就可捕捉到重传前丢失的包。

换句话说，Vegas 把收到的某个 ACK 看作检查是否已发生超时的标志。由于它只在很少的情况下检查超时，开销很小。即使能把激发快速重传的重复 ACK 数目由 3 减到 2 或 1，并不表示它能导致许多不必要的重传，因为它已对包失序发送的情况作了假设。

新的重传机制的目标不只是减少检测丢失的时间，把重复 ACK 的数目由 3 减到 2 或 1——这只是很小的节省，还在于即使没有第二个或第三个重复的 ACK 也可检测到丢失。Vegas 中仍保留了重传超时机制以防止新机制没能检测到包的丢失。

2.6.2 合理地调整传输速率

Vegas 测量和控制连接中的额外数据的数量，额外数据表示如果连接用的带宽接近网络的可得带宽时不会发送的数据。Vegas 的目标是保持网络中适当数量的额外数据。显然，如果连接发送了过多数量的额外数据，将会导致拥塞，还有，如果连接发送了较少的额外数据，它对临时增加的网络带宽无法做出快速反应。Vegas 拥塞避免行为不仅基于丢失的包，还基于对网络中额外数据数量的估计。

现在详细描述这个算法。首先，定义一个给定连接的 Base_RTT 为测得的最

小的往返时间。通常是在连接产生的包使路由器排队之前，TCP 连接上传输的第一个包的 RTT。如果假设连接没溢出，那么期望的吞吐量应为：Expected=cwnd/Base_RTT。其次，Vegas 计算当前实际的发送速率 Actual。Actual=cwnd/RTTa，其中，RTTa 为当前的往返时间。在每一个往返时间内算一次。再次，Vegas 比较 Actual 和 Expected，计算 diff=Expected-Actual。并由（1）式调整窗口，即

$$cwnd = \begin{cases} cwnd+1 & if\ diff < \alpha \\ cwnd-1 & if\ diff > \beta \\ cwnd & if\ \alpha \le diff \le \beta \end{cases} \qquad (1)$$

直觉上，实际吞吐量离期望吞吐量越远，网络中的拥塞越多，这就意味着发送速率应该降低。β 门限激活这种减少，另一方面，实际吞吐量离期望吞吐量太近，连接有没利用可得带宽的危险。α 激活这种增加。总体的目标是使网络中的额外数据保持在 α 和 β 之间。

因为这个算法，如前所述，比较实际与期望吞吐量的差和 α 、β 门限。这两个门限以单位 KB/s 来定义。但是，以连接占据了网络多少 extra 缓存来考虑的话，可能更精确。例如，在一个 BaseRTT 为 100ms，报文段大小为 1KB 的连接中，如果 α =30KB/s 且 β =60KB/s，那么我们可把 α 看作是说，连接需要占据网络中的至少 3 个 extra 缓存，把 β 看作是说，它应占据网络中不多于 6 个的缓存。

2.6.3 改进的慢启动机制

由于当连接开始时，没有安全窗口大小信息。如果这个初始的门限窗口值太小了，指数增加会太早停止，通过使用线性增加需要很长时间才能达到最优的拥塞窗口大小。结果，吞吐量下降了。另一方面，如果门限窗口设置得太大，拥塞窗口会增加直到超过了可得带宽，导致瓶颈路由处的与可得缓存相一致的丢失；这些丢失随着网络带宽的增加而增长。

需要的是一种方法来发现一个连接的还没导致这种丢失的可得带宽。为了达到这个目的，把拥塞控制检测机制包含的慢启动机制做了较小的改进。为了能检测并避免在慢启动时发生拥塞，Vegas 只在每隔一个 RTT 时允许指数增长。在之间，用的窗口大小保持不变，这样可得期望和实际的数据率的比较结果。当实际的数据率落在了期望数据率以下一个路由器缓存时，即额外数据量是 1 时，Vegas 从慢启动模式转变到线性增/减模式。

第三章 对 ad hoc 网络中 TCP 的性能仿真分析

3.1 网络仿真平台 NS 简介

本文中的仿真结果基于带有 Carnegie Mellon University 的 MONARCH 项目的扩展[11]的 Lawrence Berkeley National Laboratory(LBNL)[12]的 NS2 网络仿真器。NS 作为一个事件驱动的网络仿真平台，是美国南加州大学，施乐 PARC 实验室和加州大学伯克利共同合作的一个项目。它的源代码是开放的，目前还在不断地充实、完善。在它的当前版本 NS2 中已经实现了许多常用的网络构件，例如，网络传输协议 TCP 和 UDP；信息源模型 FTP，Telnet, Web, CBR 和 VBR；路由器队列管理机制 Drop Tail, RED 和 CBQ；路由计算法 Dijkstra 等等。

3.1.1 NS 开发背景

NS 是 Network Simulator 的首字母缩写，它是由 LBNL(Lawrence Berkeley National Laboratory)的网络研究小组开发的仿真工具。NS 是一种可扩展、易配置、可编程的事件驱动的网络仿真软件。NS 支持许多基本的协议，如 TCP 协议、一些路由协议和多点发送协议等。

LBNL 的网络仿真软件的开发始于 1990 年 5 月对 S.Keshav 的 REAL 网络仿真程序的修改。1991 年夏天，对仿真描述语言进行了修改，称为 TCPSIM，1994 年 12 月，Mccane 用 C++重写了 TCPSIM，称为 NS。

NS 是一个功能强大的仿真工具，目前，已经有 50 多个国家的 600 多个研究机构正在使用它。NS 软件可以从互联网上下载。

NS 仿真器由 10 万行 C++代码、7 万行 Otcl 代码、3 万行测试代码和 2 万行文本文档构成。NS 能在大多数 UNIX 平台下运行，包括 Free BSD、Linux 和 Sum Solaris。在 Window 95/98/NT 平台下工作不是很稳定，某些仿真工作能正常进行，某些则可能发生这样或那样的错误。

3.1.2 NS 的体系结构

NS 是用 C++和 Otcl(麻省理工学院开发的一种面向对象扩展的 Tcl 脚本语言，Tcl 本身是一种脚本语言）语言编写的。简单的从一个使用者的角度来看，它是一个面向对象的 Tcl 脚本语言的解释程序，这个解释程序有一个仿真事件调度器、一个网络构件（如节点、链路、队列等）对象库和一个用于搭建网络的管理模块

（指这些模块用于连接网络构件）库。也就是说，要使用 NS 来建立一个“网络”并在其上进行仿真，需要写一个 Otcl 脚本程序，在程序中初始化一个事件调度器、用网络构件库中的网络构件和管理模块提供的连接建立网络拓扑并通过时间调度器确定数据源传送数据包的启动和停止时间。

NS 使用 Otcl[13]和 C++[14]两种语言，是为了适应两方面的要求，一方面，许多协议的仿真要求有效地操纵字节、包头、实现某种算法，对运行时的速度要求很高，这就需要 C++这种系统编程语言来实现。另一方面，很多情况下，只需要轻微地改动某些参数或配置，Otcl 这种脚本语言就比较合适。

NS 是事件驱动的，事件调度是它的一个重要部件。在 NS 中事件是一种包（Packet），包中有独特的标号、启动时间和一个指针指向处理事件的对象。时间调度器负责监视仿真时间，在事件队列中的时间定义和响应时间启动这个事件，调用负责处理该时间的对象。

在 NS 中，事件调度器和基本的网络构件对象是使用 C++编写和编译的，这些 C++对象通过一个用 Otcl 编写的连接模块和 Otcl 对象联系起来，从而可以在 Otcl 解释器中用简单的 Otcl 脚本语言控制这些对象，调用 C++对象定义的方法和变量。一般情况下，在解释器中生成一个 Otcl 对象时，相应的生成一个 C++对象，NS 就是通过这种方式达到实用和效率的统一。

3.1.3 NS 运行平台

NS 选择平台为 Linux，因为 Linux 是一种免费系统，可安装于微机上，网上有很多软件包都支持此平台。Linux 被称为 Unix 的 PC 版，是迄今唯一能够在 PC 机硬件平台上为广大用户免费提供多任务、多进程功能的操作系统。

3.1.4 NS 软件包的组成

NS 软件包 ns-allinone 中的主要组成部分如下：

Bin：有常用的工具 ns, xgraph, nam 等的连接。

Nam: 动画显示工具。

Ns: NS 语言的核心部分。

Otcl: otcl 软件包

Tcl: tcl 语言软件包。

Tclcl: tcl 库文件。

Tk: tk 工具包。

Xgraph: 绘图显示工具。

3.1.5 如何使用 NS 进行仿真

我们通过 NS 解释程序调用仿真器，NS 解释程序是 otslsh 命令的 shell 扩展，一个仿真是用一个 otcl 脚本程序定义的。脚本程序使用 simulator 类作为与 NS 的接口。通过此类中定义的方法，可以定义网络的拓扑结构，指定网络的业务源端和业务接收端，还可以通过 simulator 类中的函数进行仿真程序，并收集数据。

在仿真程序的第一步是取得 simulator 类的一个实例。在 NS 中创建类的对象用 new 命令，删除对象实例用 delete 命令。

进行仿真，必须从生成拓扑结构开始。NS 仿真器支持以下几种拓扑结构产生方法：

(a) 使用 GT-ITM 拓扑结构生成程序。

(b) 使用 Tiers 拓扑结构生成程序。

(c) 其它拓扑结构生成程序。

(d) 手工生成网络的拓扑结构。

本课题采用的方法是手动生成网络的拓扑结构。网络的拓扑结构可以通过指定三个基本的拓扑单位来实现：nodes、links、agents。Simulator 类中有创建和确认这些拓扑单位的方法。

Simulator 类中的 node 函数用来创建节点，并给每个节点自动分配一个唯一的地址。

Simplex-link 函数用来在两个节点之间建立单向连接。Duplex-link 用来建立两个节点之间的双向连接。

Agents 是驱动仿真程序的对象。它可以被想象为在节点上运行的进程，节点可以是端主机或路由器。网络业务源端和业务接收端、动态路由模型以及各种不同的协议模块都是 agents 的实例。通过实例化一个 agent 类的子类对象可以创建一个 agent，agent/type 表示子类对象，其中 type 表示是哪一种 agent。

一旦创建了一个 agent，系统也自动为其分配一个唯一的句柄，用 simulator 类中 attach-agent 函数将这个 agent 与一个节点相连。在一个指定的节点上，每一个 agent 都被自动的赋予一个唯一的端口号，这个端口号类似于 TCP 或 UDP 端口号。

在仿真中，一些 agent 必须有相连的业务源，然而也有一些 agent 能够生成自

身的数据。例如，TCP 类型的 agent 需要有 FTP 或 Telnet 业务源相连，而恒速率的 CBR(constant bit-rate)agent 则可以自己产生数据。使用 attach-source 和 attach-traffic 将业务源连接到 agent 上。

每一个对象都有一些可以被指定的配置参数，这些配置参数是此对象的实例变量。这些参数在 NS 启动时被赋予缺省值，在 NS 软件包的 ns/tcl/lib 子目录中有一个启动文件 ns-default.tcl，此文件中有 NS 中各个类的缺省参数值。也可通过 set 命令来得到缺省参数值。也可以将一个类的所有对象的缺省值改变。

在 NS 中，用 simulator 类中的方法 at 调度事件，使得 otcl 进程能够在仿真事件的任意时间点被调用。这些 otcl 调度提供了一个灵活的仿真机制——它们可以用于 start 或 stop 网络业务。通过 run 函数来开始仿真，直至没有要处理的事件。仿真程序也可以用 stop 命令来终止，或是通过 tcl 的标准 exit 命令来退出脚本程序。

3.1.6 用 NS 进行无线网络仿真的一般步骤

用 NS 进行仿真的基本步骤如下：

（1） 产生事件调度器：产生一个仿真实例；用 at 调度时间；用 run 命令开始调度。

（2） [可选]打开跟踪机制：打开一个 nam 文件或 trace 文件跟踪链路上的包。

（3） 产生需要进行仿真的网络：配置网络参数，如路由协议、物理层协议、路由器排列长度等；创建节点；确定网络拓扑等。

（4） 产生传输层连接：设定连接的源端和目的端，并选择传输 agent 的类型，建立连接。

（5） 业务生成：确定连接上的传输业务类型，如 FTP，CBR 等；把业务附加到一个 agent 上。

（6） 传输应用层数据：用 at 调度时间来确定何时进行业务的传输。

（7） 开始仿真：调用命令 run 使仿真开始。

3.2 多跳 ad hoc 网络中的 TCP 性能仿真分析

ad hoc 网络是一种自组织无线网络，与传统的带固定设备的无线网络相比，其显著特点是网络中没有固定的通信基础设施，网络中的所有节点均可移动，且每个节点既作为终端又作为路由器，可提供包的存储转发功能。

TCP 协议是目前广泛使用的网络传输控制协议，它是为有线网络设计的，它

基于慢启动，拥塞避免，快速恢复和快速重传的拥塞控制和丢包恢复可有效地适应于有线传输。ad hoc 网络的一些本质特性使 TCP 应用于其中时面临一些问题。传统的 TCP 认为所有的包丢失都是由网络拥塞造成的。若有丢失的包，TCP 会减少窗口来降低发送速率，从而避免或减轻网络拥塞。但是在 ad hoc 网络中，许多因素都会导致包的丢失。这些原因主要有：

（1）传输介质的某些本质特性会导致包的丢失

易于出错的无线链路会引入随机误码和突发误码。随机误码会导致一个窗口中的单包或多包丢失，突发误码会导致一个窗口中的多包丢失。对单包丢失，TCP 会启动快速重传和快速恢复；多包丢失会导致重传超时，TCP 重新启动慢启动。这两种情况均会使 TCP 的吞吐量下降。

共享介质会引起多用户数据传输之间的竞争，在共享无线信道中，隐藏站和暴露站问题会使得传输失误。

（2）路由变化会导致包的丢失

路由最优化和恢复会使一些包丢失或乱序。在高度移动的环境下，移动节点的位置变化频繁，为了保证所用路由为最佳的，路由协议需要频繁地更新路由，这种路由变化会导致包的丢失和乱序。

路由中断会导致多包丢失和乱序。路由中断后，如果路由协议不能及时发现和恢复路由，中断的连接使多包丢失和乱序。

由于传统的 TCP 不能区分包丢失的原因是网络拥塞还是无线链路易出错的特性或是路由中断，它会不恰当地进行拥塞控制、拥塞避免和超时重传，从而导致吞吐量的下降。

为了验证以上理论分析结果，在本次仿真中，设计了一种网络环境，为了与以后的仿真相区别，称此次仿真环境为实验 1。为了考察 ad hoc 网络的多跳性对 TCP 性能的影响，在实验 1 中，网络拓扑结构为静态的由 8 个点组成的链式拓扑，如图 3.1 所示。在此实验中，所有节点通过半双工的无线链路通信，节点能量的最大传输范围是 250 米，每个节点的接口带宽为 2Mbps，队列的缓存为 50 个包，包的大小为 1000 字节，且对列为优先丢尾（PriDropTail）的。仿真的拓扑结构为静态的由 8 个点组成的链式拓扑，如图 3.1 所示。两邻近点之间的距离为 200 米，换句话说，只有相邻的节点能直接通信。邻节点之间的距离相同使得仿真中各点地位平等。

图 3.1 实验 1 仿真拓扑结构

MAC 的协议使用 IEEE802.11 DCF 模式，仿真中使用的路由协议为动态源路由协议 DSR(Dynamic Source Routing)[15]，仿真只在选定的一对发送端和接收端间建立一条 TCP 连接，业务流为 FTP，测量整个 120 秒仿真时间内成功接收的包数。并计算出吞吐量。本文中的吞吐量均为仿真运行 5 次后得出的吞吐量的平均值。

为了比较各个 TCP 的性能，用 Reno、NewReno、SACK、DSACK(SACK+DA，作为 TCP+DA 的代表)、Vegas 重复实验 1，分别得出各个 TCP 在相同的网络环境下的吞吐量。

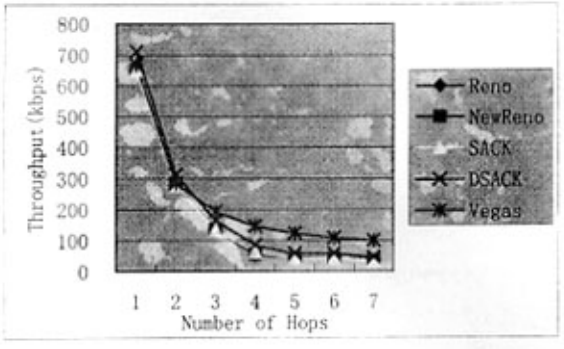


图 3.2 跳数对各个 TCP 吞吐量的影响

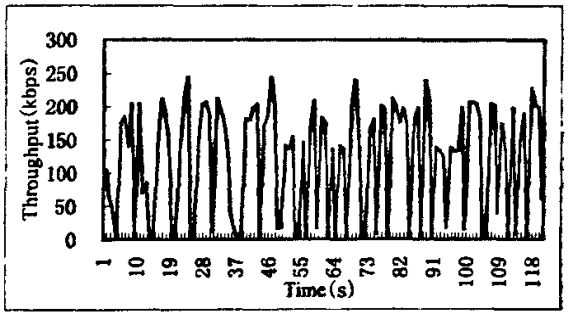
hops	Reno	NewReno	SACK	DSACK	Vegas
1	650.19	650.19	650.19	710.37	667.79
2	277.01	277.01	277.01	311.21	284.25
3	136.39	136.39	133.35	163.84	193.67
4	51.81	55.27	62.12	84.12	144.91
5	45.13	42.22	41.40	56.42	123.10
6	41.90	43.40	41.17	56.73	107.70
7	36.11	33.77	33.12	45.13	98.48

表 3.1 不同跳数下各个 TCP 的吞吐量 (kbps)

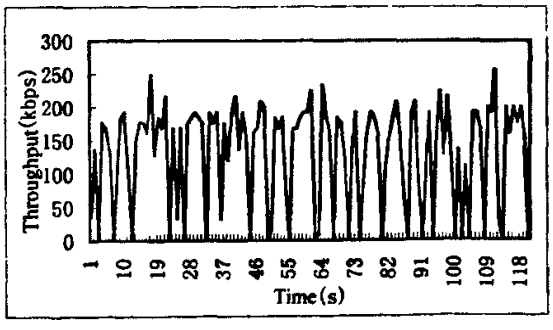
其结果如图 3.2 所示。此图表明随着跳数的增加，各种 TCP 的吞吐量都有所下降，当跳数较大以后，吞吐量又趋于平稳；当跳数较小时，Reno、NewReno、SACK、Vegas 这几种没有 DA 机制的 TCP 的吞吐量基本相同，而运用了 DA 机制的 DSACK 的吞吐量明显偏大；随着跳数的增大，Vegas 的吞吐量与其它 TCP

的吞吐量的差距明显增加，Vegas 表现出最佳的性能。这些结果可由表 3.1 更加明显的表现出来。

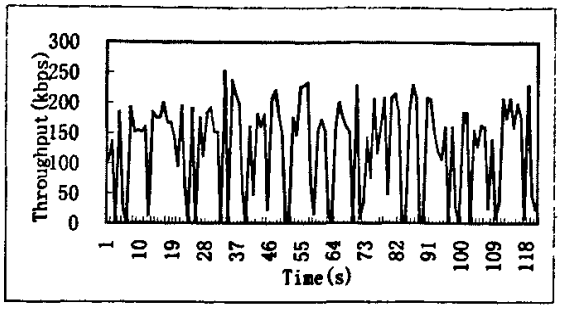
为了解释仿真结果，我们在仿真中建立一条节点 1 到节点 4 的 3 跳 TCP 连接，通过读取每一秒的时间内正确收到的包的数量来计算其吞吐量。其结果如图 3.3 所示，图 3.3 包含 5 个图，分别为各种不同的 TCP 版本在相同的最大窗口和包大小的情况下，在 1 点到 4 点这个 3 跳连接上的吞吐量。图中的 w 代表最大窗口数，s 代表包的大小，即包所包含的字节数。从图中可以看出，Reno、NewReno、SACK、DSACK 的吞吐量中有多个时间点的值达到或接近 0，这被称为 TCP 的不稳定性。从而，当跳数较小时，Reno、NewReno、SACK TCP 的吞吐量相差不大。这种不稳定性与窗口大小和包的大小有关，比较图 3.3（1）和图 3.4（1）可看出窗口越大，这种不稳定性表现得越严重；同样，比较图 3.3（1）和图 3.4（2），可以看出包的大小对不稳定性的影响。



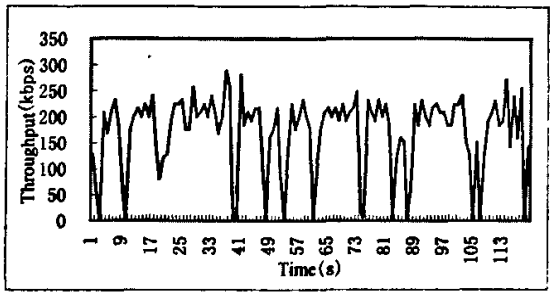
（1）Reno w=32 s=1000B



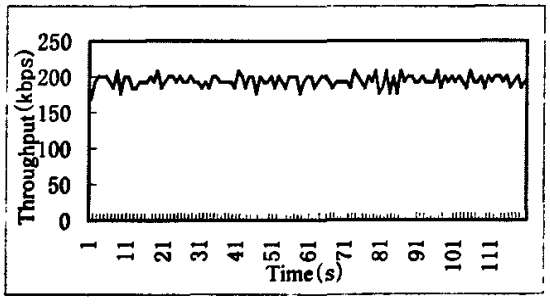
(2) NewReno w=32 s=1000B



(3) SACK w=32 s=1000B

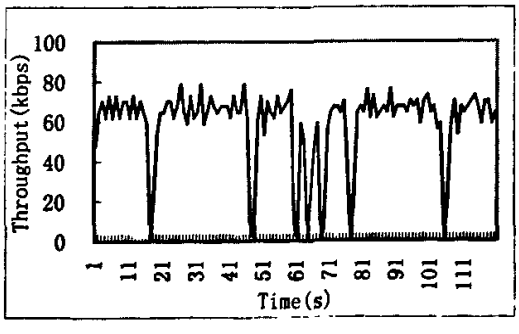


(4) DSACK w=32 s=1000B

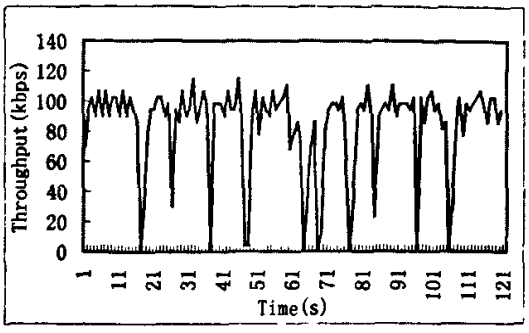


(5) Vegas w=32 s=1000B

图 3.3 点 1 到点 4 连接的各种 TCP 的实时吞吐量



(1) Reno w=8 s=1000B



(2) Reno w=32 s=512B

图 3.4 窗口和包的大小对 TCP 性能的影响

不稳定性问题根源于 MAC 层的 802.11 协议[16]，由于暴露站的问题和冲突使得中间站无法到达它的下一跳，同时 MAC 层的随机退避机制造成的信道捕获问题使之更糟。大的包和多的连续包的发送都增加了中间节点无法到达下一跳的概率。这会使节点退避后重试，多次失败后，它会宣布链路中断。此时会引发漫长而复杂的路由发现和恢复过程，而 TCP 发送端也会停止发送，从而导致了吞吐量的降低。同时 MAC 层的随机退避机制总是有利于成功传输的连接，从而造成信道捕获问题，这也会使 TCP 的性能不稳定。在跳数较小时，如 1 时，数据冲突和暴露站问题不是很严重，因此不会有不稳定性。随着跳数的增加，有更多的中间节点会影响到连接，节点无法到达下一节点的概率越来越大，因此吞吐量性能随着跳数的增加显著降低。而 DSACK 由于减少了将近一半的 ACK 的数量，多跳时，减少了由于回传的 ACK 发生冲突的次数，中间节点无法到达下一跳的概率减少，从而会有比 Reno、NewReno、SACK 较好的性能；跳数较少时，由于 ACK 数量的减少节省了网络资源，可用于有用数据的传输，吞吐量也较高。Vegas TCP 的窗口增减策略是保守的且它有一个“突发避免”策略[17]，而且由于它能快速检测丢失，因此它在多跳时没有不稳定性。

3.3 ad hoc 网络的移动性和高丢包率对 TCP 性能的影响

3.3.1 节点的移动速率的影响

在本次仿真中，设置网络的拓扑结构为在 30 个在 670*670 正方形区域内移动的节点，此次仿真称为实验 2。节点的移动基于“random waypoint”移动模型。在这种移动模型中，节点随机选择一个目标节点并向之移动，到达目标节点后暂停一段时间，接着再随机选择一个目标节点并向之移动，如此反复下去直到仿真

停止。其它的设置与实验 1 的仿真设置基本相同。

首先对 4 种不同的平均移动速度进行仿真：2m/s，10m/s，20m/s，30m/s，其暂停时间均为 0。对每个平均速度 v，节点的移动速度在 0-2v 区间内均匀分布。对每个平均速度运行 10 个不同的移动形式，共需要 40 个移动形式文件。

用 Reno，NewReno，SACK，DSACK，Vegas 重复实验 2，分别得出不同的 TCP 在不同的节点平均移动速度下的吞吐量曲线变化图。如图 3.5 所示。

很明显，随着平均移动速度的增加吞吐量明显下降。这有几方面的原因，首先，当移动速度增加时，路由的中断和形成过程会越来越频繁地发生，这在长时间内会导致大的链路中断概率，会增加重传次数和路由开销，减少有效的数据传输时间。其次，当路由中断后，如果路由协议不能及时地恢复路由或发现新路由，会发生重传超时。TCP 会启动慢启动，接着进行拥塞避免，减少了窗口大小，从而导致了吞吐量的下降。

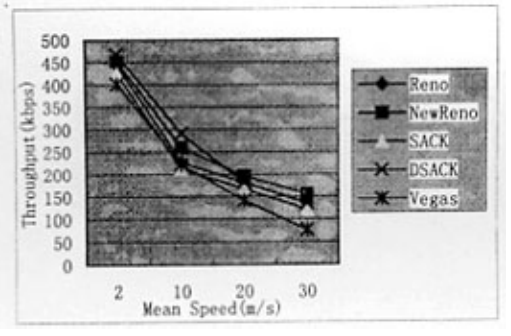


图 3.5 不同的节点移动速度下各种 TCP 的吞吐量

我们还能发现，在各种移动速度下，Reno，NewReno，SACK 的吞吐量有较小的差别。NewReno 有比 Reno 较高的吞吐量，这是由于 NewReno 在不增加开销的前提下，对多包丢失作出快速反应，减少发生超时重传的次数。SACK 对多包丢失也可快速恢复，但在本文的仿真结果中，SACK 并没有表现出多大的优越性，这是由于在 ad hoc 网络中，延迟带宽积(delayed bandwidth product)很小[18]，SACK 的选择性确认机制对吞吐量的提高不大，相反的，由于在 TCP 头中的 SACK 选项带来的开销和由于在发送端和接收端增加的额外的处理程序会使得 SACK 的吞吐量下降。在平均移动速度较小时，DSACK 的吞吐量最大，这是由于 DSACK 减少了将近一半的 ACK 的数目。减少了 MAC 层的冲突，同时节省了网络资源。但随着移动速度的加快，DSACK 的由于减少 ACK 数带来的好处已不能弥补由于

SACK 增加的开销带来的弊端。

Vegas 的吞吐量随着移动速度的增大而显著降低，这表明 Vegas 运用于 ad hoc 网络有新的问题。其中主要的是由重新寻找路由引起的问题。由于 Vegas 根据最小的往返时间 base_RTT 来更新窗口，对 base_RTT 的估计的准确性就很重要，而当 ad hoc 网络中的节点移动速度快时，重新寻找路由就会很频繁，这就会导致 base_RTT 的不准确性，从而引起吞吐量的严重减少。如果新路由的 RTTa 比 base_RTT 小，base_RTT 会更新为新路由的 RTT，这不会对吞吐量有何影响。但是，新路由的 RTTa 通常比 base_RTT 大，这时 base_RTT 不会更新，结果，发送端会认为发生了拥塞而减少窗口大小。但实际上，根据(1)、(2)式，Vegas 的窗口应该增加，以使 Diff 在α和β之间，因此，重新寻找路由会严重的影响 Vegas 的性能。

3.3.2 暂停时间的影响

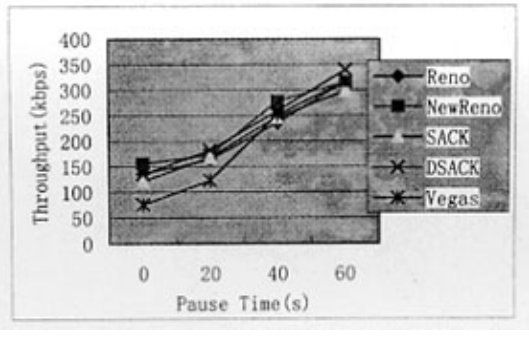


图 3.6 不同暂停时间下各种 TCP 的吞吐量

接着，以 4 种不同的暂停时间来运行实验 2：0s，30s，60s，90s，节点的平均移动速度均为 30m/s，每种暂停时间用不同的移动形式运行 10 次，取其吞吐量的平均值。

图 3.6 显示了不同的暂停时间下，不同的 TCP 的平均吞吐量。很明显，随着暂停时间的增长，吞吐量也随之增长。这很容易理解，较长的暂停时间表示了移动节点的高稳定性和较小的链路中断，在有较小中断的情况下，发送端可发送较多的包。

3.3.3 链路丢包率的影响

最后，以 4 种不同的链路丢包率运行实验 2：0.1，0.05，0.01，0.001。节点均静止，因此没有链路中断的影响。对每种丢包率用不同的节点位置仿真 10 次，

取其吞吐量的平均值。

图 3.7 所示为不同的链路丢包率下各种 TCP 的平均吞吐量。从中可看出，当丢包率增大时，各 TCP 的吞吐量随着丢包率的变化趋势与其随着节点平均移动速度的变化趋势基本相同，性能均有所下降，这是因为丢包率的增加会增大重传和超时的次数，使传输效率降低。Reno，NewReno，SACK，DSACK 吞吐量之间的差别也可由 3.3.1 中这四吞吐量之间的差别原因来解释。根据 2.6.2 的(1)式，Vegas 在丢包率较高时 cwnd 减得多，增得少。因此吞吐量下降最为严重。

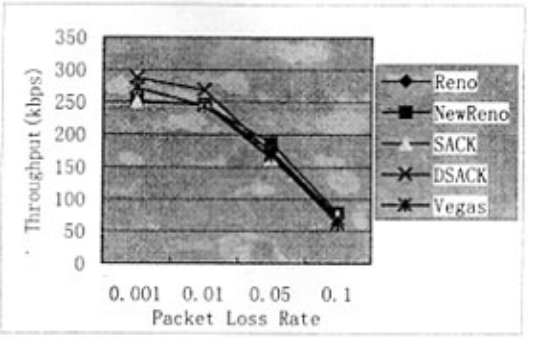


图 3.7 不同的丢包率下各种 TCP 的吞吐量

3.4 ad hoc 网络中的 DA 机制性能评估及改进

从以上的仿真结果中容易看出，DA 机制在无线 ad hoc 网络中有较好的性能表现，为了更加清楚地了解 DA 机制，设计了实验 3。实验 3 的网络拓扑结构和基本参数设置与实验 1 基本相同。由前面的仿真结果可知，TCP Reno、NewReno、SACK 有着基本相同的拥塞控制机制，且性能表现也近似相同。因此在本次仿真中，选用 TCP SACK 作为有相同的拥塞机制的几种 TCP 的代表，同时由于 TCP Vegas 的完全不同的拥塞控制机制，也对 Vegas+DA 的情况作了分析。

3.4.1 DA 机制对多跳连接的性能改进

图 3.8 是不同的跳数下 SACK，SACK+DA，Vegas，Vegas+DA 的吞吐量变化图，很明显，随着跳数的增加四种 TCP 的吞吐量均下降。其原因在 3.2 中已有描述，即由于 MAC 层的暴露站和冲突问题，SACK 有不稳定性问题，使得快速重传和超时重传的次数均较大，传输效率降低。Vegas 虽没有不稳定性问题，但根据（1）、(2)式，跳数的增多带来的冲突的增多和往返时间的加长也会使 Vegas 的窗口减少。从而吞吐量下降。SACK+DA、Vegas+DA 相对于 SACK、Vegas 吞吐量的增加量表现出 DA 机制由于减少冲突和节省资源所带来的性能的改善。具体

可见图 3.3(4)。这从表 3.2 中可更清楚地看出。表 3.2 中 SACK+DA 相对于 SACK 的吞吐量增长率为 9.26%-36.27%，Vegas+DA 相对于 Vegas 的增长率为 9.29%-23.06%，不及 SACK+DA 的增长率高，因为 Vegas 没有不稳定性问题，MAC 层的暴露站问题和冲突对 Vegas 的影响没有对 SACK 的影响严重。

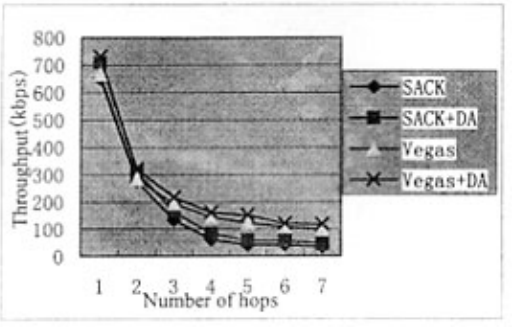


图 3.8 不同跳数时的运用 DA 机制的 TCP 的吞吐量改进

跳数	SACK	SACK+DA	增长率	Vegas	Vegas+DA	增长率
1	650.19	710.37	9.26%	667.79	729.80	9.29%
2	277.01	311.21	12.35%	284.25	319.99	12.57%
3	133.35	163.84	22.87%	193.67	217.28	12.19%
4	62.12	84.12	35.42%	144.91	160.84	11.00%
5	41.40	56.42	36.27%	123.10	151.48	23.06%
6	41.17	56.23	36.59%	107.70	120.57	11.95%
7	33.12	45.13	36.27%	98.48	117.07	18.87%

表 3.2 使用 DA 和不使用 DA 的 TCP 不同跳数下的吞吐量比较

3.4.2 最大窗口的影响

窗口大小代表着发送速率，窗口太小会使得链路带宽得不到充分利用，窗口太大会加剧 MAC 层的冲突问题[6]。为了找出最适合的最大窗口数，本文以 3 跳连接为例，对最大窗口为 1，2，4，8，16，32 的情况分别进行了仿真，结果如图 3.9 所示。从图中可看出，由于 Vegas 的拥塞控制机制，不管最大窗口如何变化，Vegas 发送端总是根据实际情况将窗口保持在一个平衡点上，因此吞吐量不随最大窗口变化。SACK 在窗口为 1 或 2 时，吞吐量与 Vegas 差不多，随着窗口加大，冲突加重，吞吐量随之下降。

SACK+DA、Vegas+DA 在最大窗口为 1 时，吞吐量非常小，这是因为当发送

端在发完一个包后，等待相应的 ACK 来发下一个包，而由于接收端运用了 DA 机制，此时接收端正在等待下一包的到来，直到等待时间超过了一个给定值，本文中为 100ms，接收端发回 ACK，这时发送端才可发下一个包，这样每要发一个包就要先等待 100ms，因此运用了 DA 机制的 TCP 连接在最大窗口为 1 时的性能很差。而最大窗口大于或等于 2 时，SACK+DA 和 Vegas+DA 的吞吐分别比相同最大窗口下的 SACK 和 Vegas 的吞吐量大。因此在多跳 ad hoc 网络中，接收端运用了 DA 机制的 TCP 的最大窗口最好设置在 2-4 内。

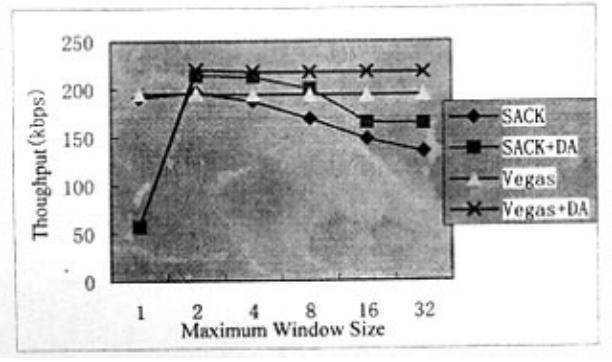


图 3.9 最大窗口对吞吐量的影响

3.4.3 小文件的传输时延

在这部分中，我们考察 DA 机制对小文件传输时延的影响，即从文件开始发送到全部接收到所需要的时间。在仿真中设业务流 FTP 只有 32 个包，最大窗口设为 4，观察 SACK，SACK+DA，Vegas，Vegas+DA 对这个小文件的传输时延。结果如图 3.10 所示。

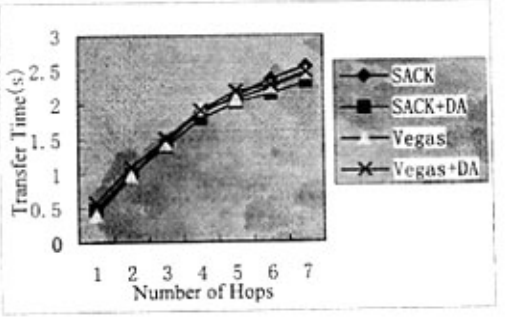


图 3.10 不同跳数下小文件的传输时间

从图中可以看出，Vegas+DA 在各种跳数下的传输时延均比 Vegas 的时延长，在跳数较小(如 1、2)时，SACK+DA 的传输时延也比 SACK 的时延长。这表明

DA 机制不适合小文件的传输，特别是对实时性有要求严格的传输。这是因为 TCP 的窗口均从 1 开始，如前所述，运用了 DA 机制的接收端在收到第一个包之后不立即发回 ACK，而是等待下一个包的到来，而此时发送端也在等待接收 ACK 来增加窗口，直到等待超时。接收端这时会发回 ACK，发送端收到这个 ACK 后，才把窗口增加，接着继续发包。这会使得慢启动非常慢，从而造成了较大的时延，这对大文件的影响不是很大，而对小文件来说，这种时延相对来说就已很大。

比较 SACK 和 SACK+DA 可看出，当跳数较大时，SACK+DA 的传输时延又比 SACK 小，这是由于随着跳数的增多，SACK 的不稳定性越来越严重，SACK+DA 的优势越来越明显地表现出来。

因此，在多跳 ad hoc 网络中，TCP 的吞吐量随着连接跳数的增加而降低，在接收端用 DA 机制可以减少冲突、节省资源。通过分析比较仿真结果，得出结论，DA 机制确实能够提高 ad hoc 网络的吞吐量，但对小文件的传输时延比较大，不适合小文件的实时性业务。同时在多跳 ad hoc 网络中，最大窗口最好设在 2-4 内。

3.4.4 对 DA 机制改进

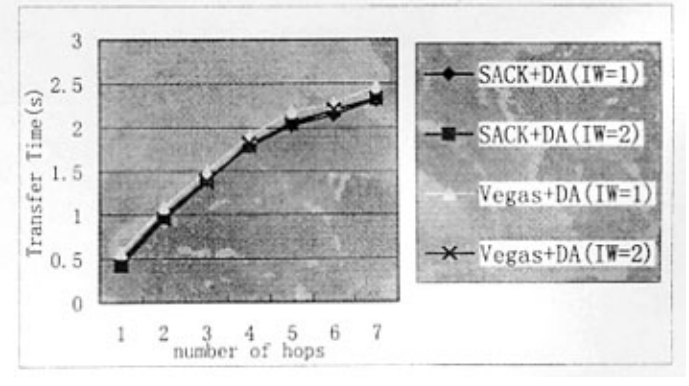


图 3.11 增大初始窗口对 DA 机制的改进

如 3.4.3 中所述，DA 机制不适合小文件的实时性业务，这是因为运用带 DA 机制的 TCP 传输小文件时有较大的时延，而这归根结底是由于 TCP 发送方的初始窗口为 1。

为了修正 DA 机制的这个缺点，设初始窗口为 2，再进行上述仿真，得出在不同跳数下，使用带和不带 DA 机制的 TCP 传输小文件的传输时延，其结果如图 3.11 所示。图中 IW 为初始窗口。从图中可看出对于 Vegas+DA 来说，在跳数由 1 到 7 之间变化时，IW 为 2 时的传输时延均比 IW 为 1 时的传输时延小；而对于

SACK+DA 来说，当跳数较小时，IW 为 2 时的传输时延确实比 IW 为 1 时的传输时延要小，而随着跳数大，IW 为 2 也不能减少 DA 机制的传输时延了，这是因为 SACK TCP 在多跳环境中有不稳定性，且窗口的增大会加重这种不稳定性。IW 为 2 使得发送方在开始时就以相对较大的发送速率进行发送，缺少了探测带宽的过程，更加剧了 TCP 的不稳定性，使得吞吐量在较多的时间点上达到或接近 0，这样由于这个原因引起的时间延迟比由于 DA 在慢启动时等待超时的延迟更严重，所以在跳数较大时，IW 为 2 的传输时延比 IW 为 1 时的传输时延还要偏大。但对 Vegas TCP 来说，不存在不稳定性问题，使初始窗口为 2 对小文件的传输时延减少的较为明显。

第四章 已提出的 ad hoc 网络中 TCP 改进综述

由前面的性能分析可知，ad hoc 网络的一些本质特性使 TCP 应用于其中时面临一些问题。目前各国都在致力于提高 ad hoc 网络中的 TCP 性能。

近几年对 ad hoc 网络中的 TCP 性能的改进主要是对传统的 MAC 层、网络层和传输层进行修改使之更适合 ad hoc 网络，还有的改进是在原有网络的基础上增加一层，添加一些功能来适应 ad hoc 网络的复杂性。以下对各改进机制分别加以描述。

4.1 MAC 层的改进

在 802.11 中，每一个正确接收的数据包都要想发送方作出正面的确认，如果在一个特定的时间内没收到确认，发送方重传这个包。连续的重传尝试由二进制指数退避算法间隔开在连续重传了预定的次数（即重传门限）后，发送方放弃重传并丢弃该包，并向上层报告路由失效。文[19]提出，在一个由 30 个节点在 300m*1500m 的区域内随机移动而形成的 ad hoc 网络内，在不同的链路误码率的情况下，适当的增加重传门限都会增加 TCP 的吞吐量，且误码率越高，连接的跳数越多，吞吐量增加的幅度越大。这种改进是针对 ad hoc 网络的移动性问题所进行的改进，这种改进有时会导致发生路由中断时，会增加其检测时延。

4.2 网络层的改进

4.2.1 基于竞争的路径选择 COPAS[20]

COPAS(Contention-based Path Selection)的目的是解决 MAC 层的信道捕获问题造成的吞吐量的下降。MAC 层的不公平性和信道捕获会使一个节点总是可以接入信道，发送数据。而邻近的其它节点即使开始发送数据的时间早，也很长时间无法接入信道，从而大大降低了 TCP 的吞吐量。COPAS 采用了两个新的路由技术来平衡网络中的竞争，即运用分离的前向和后向链路分别传输两个方向上的 TCP 报文段，以此来减少两个方向上 TCP 报文段的冲突。同时 COPAS 运用了动态竞争平衡技术，这种技术连续地监视网络竞争，并且选择有最小竞争的路由来避免信道捕获情况的发生。COPAS 虽可提高吞吐量，但是其算法复杂，实现较困难。

4.2.2 对路由失效指示响应的路由协议的改进[21]

由于 MAC 层共享信道的特征，如果节点在规定的重传次数内不能接入信道，

则该节点向数据包的源节点报告链路失效消息（这是假的路由失效消息），从而触发漫长的路由发现过程。为了避免这种不必要的路由发现过程，对路由协议改进如下：当某节点的路由协议收到来自链路层的到下一跳路由失效的指示时，并不马上向源节点发送路由出错消息，而是向对应的下一跳发送 Hello 消息，同时设置计时器。如果在计时器超时之前没有收到来自下一跳对 Hello 的应答，则重传 Hello 消息。如果计时器超时的次数超过了某个特定值，仍未收到应答消息，则判定到下一跳的链路失效，并向源节点发送路由出错消息。如果计时器的超时次数在到达设置的最大次数之前收到来自下一跳的应答消息，则复位计时器。并取消向源节点发送路由出错消息。这种改进能有效的排除由假的路由出错消息触发的路由发现过程，且对任何路由协议均适用，但在链路真发生中断时会延长路由失效的判定时间。

4.2.3 关于 TCP 的源路由 TSR[22]

当源节点依赖于邻节点提供的不随路由失效而更新的路由信息时，其选择无效路由的可能性会很高，无效的路由会引起连续的超时，从而导致 TCP 吞吐量降低。TSR(TCP-aware Source Routing)可以降低选择无效路由的概率。TSP 类似于 ELFN，但比其更有效。这个协议的目的就是在不改变终端系统的 TCP 协议栈的前提下，减少路由协议选择无效路由的可能性。具体操作为当源节点收到一个通知路由失效的消息时，TSR 并不从立即从路由缓存中选择一个替代路由，而是进入到一个保持状态，并周期性的发送探寻包来发现可靠的替代路由，同时，只有目的节点能对探寻包做出响应，当源节点收到探寻包的响应时，就沿第一条响应路由反向恢复传输。

当然，当源节点收到路由失效消息时，有可能选择的替代路由是有效的，这时 TSR 就会增加开销且降低链路的利用率，不过由于多包丢失对 TCP 的性能的影响是致命的，在大多数情况下，TSR 还是有效的。

4.3 传输层的改进

4.3.1 DA 机制[9]

前面已介绍过，DA(Delayed ACK)机制通过减少 ACK 的数量来减少发送数据在竞争信道时与 ACK 发生的冲突，节省网络资源。DA 只是收端的机制，它使 TCP 不是对每个收到的包都产生一个 ACK，而是对收到的每 2 个完整的包产生一个 ACK。另外，两个 ACK 的时间间隔不应超过一个门限值，否则重传 ACK。

DA 适合无线信道，但是对小文件的传输时延较大，且当最大窗口为 1 时，运用 DA 机制的 TCP 发送速度将会很慢。

4.3.2 乱序检测和响应 DOOR[23]

DOOR(Detection of Out-of Order and Response)目标是减少乱序到达的包对 TCP 吞吐量的影响。如在本文 3.2 中所述，在 ad hoc 网络中，由于节点移动，链路中断频繁导致的路由的重新发现和恢复不仅会引起大量包的丢失，还会引发频繁的乱序发送。此时 TCP 接收端就会发送重复的 ACK，接着接收端启动快速重传，使得发送速率降低，从而吞吐量下降。DOOR 不使用反馈机制就可检测到乱序 OOO(Out –Of-Order)的发生：在发送端，根据 ACK 的序号数判断 OOO；在接收端，在 TCP 头中加入两个字节的 TCP 选项字段判断 OOO，且若检测到 OOO，接收端应通知发送端，因为是由发送端对 OOO 做出响应。每当检测到有 OOO 发生时，发送端暂时使拥塞控制失效，并将状态变量保持一段时间 T1。如果检测到 OOO 时，TCP 发送端已经处于拥塞避免状态，应立即退回到进入拥塞避免以前的状态。原因如下：如果发送端是根据超时判断拥塞的，可能是由于链路中断引起的暂时性的传输中断。发送端能接收 ACK 表明中断已结束，TCP 可快速恢复为路由变化前的状态；如果发送端根据收到 3 个重复的 ACK 来判断发生了拥塞，可能是由于 OOO 或暂时的路由中断，同样 TCP 应恢复到以前的状态。这种方法通过检测 OOO 来区分路由中断和拥塞，不需要反馈信息，其复杂度和开销比基于反馈的方法均有所降低，但是由于基于反馈方法的信息直接来源于网络，所以 DOOR 不及基于反馈的方法精确。而且如果 OOO 不是由路由中断引起的，如在多径路由中，DOOR 就不适合了。

4.3.3 重新计算的 TCP—TCP-RC[24]

TCP-RC (TCP-ReComputing)在路由重建后，重新为 TCP 连接计算 cwnd 和 ssthresh，这样在路由变化时，它能根据当前 TCP 连接的容量适应性的调整 TCP 的传输速率。TCP-RC 引入了一个 TCP 连接参数，负载因子 ρ：

ρ =传输速率/当前 TCP 连接的容量。

通过保持 ρ 的稳定来实现 TCP-RC 降低突发业务流的可能性并避免了在高网络负载的情况下启动拥塞控制的作用。cwnd 和 ssthresh 是两个重要的 TCP 变量，它们控制着发送速率。每当路由改变时，就通过重新计算 cwnd 来保持 ρ 的稳定。

TCP-RC 对移动性大，路由频繁变化的 ad hoc 网络的 TCP 性能有显著改善，

但增加了计算量，这对能源受限的移动主机来说是不利的。

4.4 综合的改进

4.4.1 基于反馈的 TCP—TCP-F[25]

在 TCP(Feedback-based TCP)体系结构中，TCP 有两种状态：Snooze 和 Established。首先考虑一个已经建立好的 TCP 连接。开始时发送方的 TCP 处于 Established 状态，即正常状态。当某一个中间主机的网络层发现由于下一个主机的移动造成路由失败的话，就发送一个 RFN(route failure notification)给发送方。发送放在收到这个报文之后，就会进入 Snooze 状态。此时 TCP 完全停止发送任何报文段，把当前所有的定时器标记为无效，冻结所有的状态变量（比如重发定时器的值和 cwnd 的大小），并启动一个新的定时器——当该定时器超时时，TCP 发起新的路由查找请求。另一种使发送方返回 Established 状态的方法是 RRN(route reestablishment notification)报文。当先前任一转发过 RFN 报文的节点发现一条新的路由时，就向发送方发送一个 RRN 报文。当发送方收到 RRN 报文时，会发送当前窗口里的所有未被确认的分组。

通过上面简短的叙述，已经可以看出基于显式的报文通知的基本特点：1）围绕发送方的 TCP 层作一定的改动，把 TCP 分成至少两个状态——正常状态、重新计算路由状态；2）需要中间节点即时地发送一个路由失败通知给发送方；3）当发送方收到该通知后，都停止发送分组（一些特殊用途的分组除外，如探查路由是否重新建立的分组），有可能的话，还会改变一些参数如 cwnd 和 RTO 等。

这种改进减少了由于路由中断引起的包的丢失，从而避免了 TCP 不必要地触发拥塞控制，减少窗口。但由于它引入了 RRN 和 RFN，会增加系统开销，并且这种改进还要求中间节点保留状态信息。

4.4.2 明确路由失效通知：ELFN 技术[26]

ELFN(Explicit Link Failure Notification)与 TCP-F 的不同之处在于 ELFN 恢复路由的方法与 TCP-F 截然不同，它没有 RRN 这样的报文来通知发送方已经找到一条新的路由，而是让发送方定时发送分组，通过是否收到 ACK 来确定路由是否已经恢复。此外，ELFN 是与特定路由协议(DSR)有关的，ELFN 就是利用 DSR 协议里一种特殊的分组扩展而成的。这种方法可精确地通告路由中断，避免了不必要的拥塞控制，改善了 TCP 的性能，但增加了网络开销和复杂性。

4.4.3 限制拥塞窗口增加的 TCP—TCP/RCWE [27]

TCP/RCWE(TCP with Restrict Congestion Window Enlargement)基于对当前网络状态的估计来改进 TCP 在 ad hoc 网络中的性能，运用 TCP/RCWE 的发送端处于 3 种状态之一：正常、链路中断和拥塞。在正常状态下，发送端发送数据包；当收到明确链路失效消息 ELFN(Explicit Link Failure Notification)时，进入链路中断状态，此时停止发送数据且不启动拥塞控制机制。当有包丢失但没有收到 ELFN 时，节点进入拥塞状态，这时，不进行传统的 TCP 拥塞控制，而是利用 NSTATE 的值来区分包的丢失是由于网络拥塞还是由于无线链路的传输特性，从而做出不同的反应。NSTATE 可由下式求得：

$$NSTATE=\begin{cases} true & RTOnew \le RTOold \\ false & RTOnew > RTOold \end{cases}$$

式中，RTO_{new} 为当前 RTO 值，RTO_{old} 为前一 RTO 值。当 NSTATE 值为 false 时，拥塞窗口 cwnd 不增加；反之，若 NSTATE 值为 true 时，cwnd 根据所处状态为慢启动还是拥塞避免适当地增加。

4.4.4 带缓存容量和序列信息的 TCP—TCP-Bus [28]

TCP-Bus（TCP with BUffering capacity and Sequence information）中也有 TCP-F 体系中 RFN 和 RRN 相对应的控制报文：ERDN(Explicit Route Disconnection Notification)和 ERSN(Explicit Route Successful Notification)。与后者相比，TCP-Bus 对网络层作了更详细的设定：中间节点可以缓存分组，这样当找到新的路由时，可以采用类似于 SACK 的选项来减少不必要的重传，规定了控制报文的具体格式，同时还提供了两种可靠传输控制报文的机制。另外，TCP-Bus 也可以由发送方定时探查路由发生中断的节点（PN，Pivoting Node）获取 ERSN 报文。

4.4.5 分割 TCP—Split-TCP[29]

前面提到的 TCP-F 只能减少不稳定性，并不能消除多跳连接的不公平性问题。长的连接链路多且易于中断，冻结的可能性大；短的连接传输得快，易于独占共享链路。Split-TCP 通过把长的连接分成短的局部的段来提高公平性，在 2 个局部段之间用代理作为接口，当代理收到一个 TCP 报文段，缓存之，发送局部 ACK--LACK 来确认此包，并把此包发送到下一个局部段，如此进行下去，直到到达目的节点。这个方法没有改变 TCP 的端到端的确认机制，因为源节点在没有收到目的节点发来的累积的 ACK 之前不会将其缓存中的包清除。Split-TCP 把端

到端的可靠性和拥塞控制这两个传输层的功能分开了。Split-TCP 进一步减轻了不稳定性问题，但增加了算法的复杂性，同时增加了网络中传输的局部 ACK，增加了额外开销。

4.5 其它改进方案

4.5.1 链路信号强度代理—LSSA[30]

在 ad hoc 网络中，每个移动节点发射一个标识信号来表明或通知临近节点它的存在。LSSA(Link Signal Strength Agent)用同样的方法表明和通告 TCP 连接的存在。此方法引入了一个新层，叫做 LSSA 层，位于 TCP 和 IP 之间。当收到来自低层的信号强度指示，LSSA 将其封装成链路信号强度指示消息 LSSI，发送到源和目的节点。通过分析 LSSI 消息，源节点可监视当前 TCP 连接的状态，根据链路状态，源节点冻结其窗口，启动拥塞控制或是进行路由重建。此方法虽然改善了 TCP 的性能，却改变了 TCP/IP 的基本体系结构。

4.5.2 Ad hoc TCP—ATCP[31]

ATCP 在 TCP 和 IP 之间插入一层，此层称为 ATCP，它监听由 ECN(Explicit Congestion Notification)消息和 ICMP 的 Destination Unreachable 消息提供的网络状态信息。相应地，TCP 发送端进入相应的状态。收到 Destination Unreachable 消息时，TCP 接收端的状态冻结（TCP 接收端进入 persist 状态），直到新的路由建立，这期间发送端不发送任何包。当收到 ECN 时，发送端不用等待超时就可进行拥塞控制。

ATCP 层的功能只在 TCP 接收端激活，该层监测 TCP 和网络的状态，并做出相应的反应。ATCP 有四种状态：normal、congested、loss、disconnected。在 normal 状态下，ATCP 层不进行任何操作，是不可见的。当 ATCP 观察到 TCP 发送端收到 3 个重复的 ACK 或 TCP 的 RTO(Retransmission Time Out)将要超时，ATCP 使 TCP 进入 persist 状态，而其自身进入 loss 状态。在 loss 状态，ATCP 重传没得到确认的包。直到收到新的 ACK，TCP 退出 persist 状态，ATCP 返回 normal 状态。当收到 ECN 时，ATCP 进入 congested 状态，忽略重复的 ACK 和超时，TCP 这时进行拥塞控制。当收到 ICMP 的 Destination Unreachable 消息时，ATCP 使 TCP 进入 persist 状态，其自身进入 disconnected 状态，TCP 周期性的产生探测包，直到路由恢复，这时 TCP 退出 persist 状态，ATCP 返回 normal 状态。ATCP 保持了

端到端的语义，而且是透明的，即有和没有 ATCP 的节点可正常地建立连接。但是增加了网络开销和算法的复杂性。

第五章 提高 ad hoc 网络中 TCP 性能的改进

由第四章可知，当前的改进大多都集中在网络层以上。而从 3.1 可知，多跳 ad hoc 网络中 TCP 性能下降的主要原因是 MAC 层的暴露站问题和二进制退避机制引起的 TCP 的不稳定性，本部分中，将根据前面的仿真分析对 ad hoc 网络的 MAC 层进行改进，以提高多跳 ad hoc 网络中 TCP 的吞吐量并减轻其不公平性问题。

5.1 增加 MAC 层的最小竞争窗口

由 3.2 可知，在 ad hoc 网络中，TCP 吞吐量随着跳数的增加显著降低。这是由于吞吐量在多个时间点达到或接近 0，即 TCP 的性能不稳定。随着跳数的增加，不稳定性越来越严重。TCP 的不稳定性根源于 MAC 层的冲突和暴露站问题[3]。

目前的 ad hoc 网络的 MAC 层协议为 IEEE802.11[32]。IEEE 802.11 协议主要用于带有线骨干基础设施的无线局域网（WLAN）中。802.11 WLAN 是一个星型网络，接入点作为中心点，通信总是发生在接入点和节点之间。在这种 WLAN 中，所有节点都位于接入点的传输范围内。每当一个节点与接入点之间的传输开始时，其它所有的节点均已获知有数据在传，因为它们能监听到 RTS 或 CTS。这些 RTS/CTS 包含了指示传输时间的域。这样传输可有效地保留介质直到传输完毕。

而 ad hoc 网络是自组织网络，源节点不能检测到目的节点附近的数据传输，因此通常源节点发送一个 RTS 包来接入信道时，在目的节点附近的数据传输就会与 RTS 发生冲突。若源节点重试 7 次后仍没收到相应的 CTS，它就认为目的节点不在它的通信范围内，并产生一个链路失效指示。因此 MAC 层是产生错误的路由失效消息的根源所在。

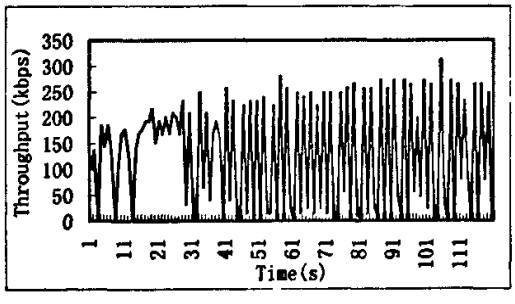
另外，节点在发送 RTS 之前先随机退避一段时间，退避的时隙数平均分布于[0，CWmin]，CWmin 为最小竞争窗口。IEEE 802.11 中的缺省值为 31。当节点发送了 RTS，但没收到 CTS 时，它就认为在目的节点处发生了冲突，为了避免再次发生冲突，节点先退避一段时间，退避时隙数平均分布于[0，2*CW]，CW 为竞争窗口。接着再重传 RTS，每重传一次，竞争窗口就加倍， 最多可重传 7 次。在 WLAN 中，只要有数据在传，所有的节点均可获知，RTS 的冲突主要是由于

多个站同时发送 RTS 所致。因此开始时的退避时间较小实际上可避免在网络中浪费时间。

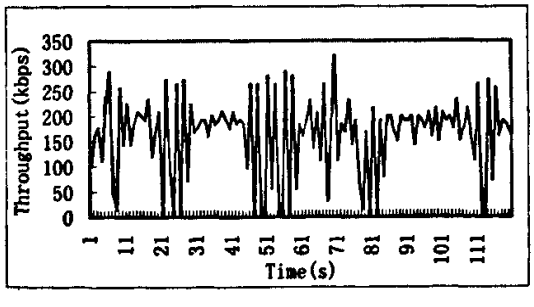
对于 ad hoc 网络来说，目的节点监听到网络中有数据在传输，因而在数据传输期间一直保持沉默，而不能对源节点发来的 RTS 做出反应。源节点对此一无所知，结果，发出的 RTS 在目的节点处因与其它的数据发生冲突而浪费掉。因此在 ad hoc 网络中退避时间太小不合理。

对此问题的解决方案是增大最小竞争窗口，使 RTS 传输间隔合理增加，同时使所有节点有平等的机会接入信道。因此当发出了 RTS 却收不到 CTS 时，就可以认为发生了冲突，退避较大的时间后重传 RTS，减少 RTS 由于冲突而浪费掉的概率，从而在很大程度上减少节点接入信道尝试失败 7 次的次数。进而减少产生错误路由失效信息的次数。本文中设计了实验 4，其网络拓扑结构和基本的网络参数设置与实验 1 的基本相同，只是为了更加明显的看出增加最小竞争窗口对 TCP 不稳定性的改进，在实验 4 中设包的大小为 1460 字节。其中的 TCP 为 NewReno。同时设置最小竞争窗口分别为 31、63、127、255、511，重复实验 4，通过分析比较仿真结果，得出最适当的 CWmin 值。

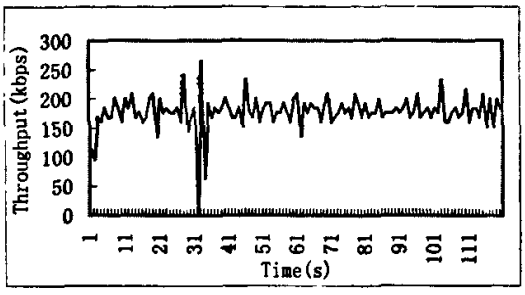
5.1.1 改进对 TCP 不稳定性的影响



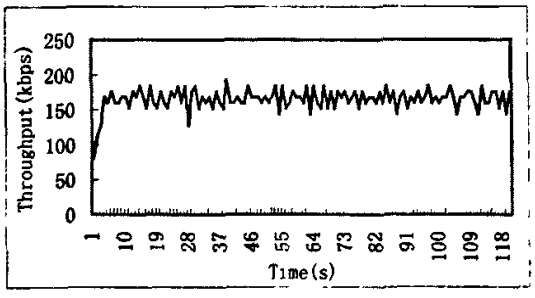
(1) CWmin=31



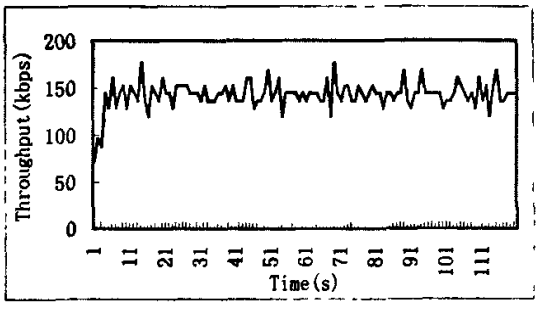
(2)CWmin=63



(3) CWmin=127



(4) CWmin=255



(5) CWmin=511

图 5.1 改变 CWmin 时 3 跳连接的吞吐量

为了直观的表现增大最小竞争窗口带来的性能改善，首先建立一条节点 1 到节点 4 的 3 跳连接，观察每一秒成功接收的包数，计算出吞吐量。其结果如图 5.1 所示。图 5.1 包括 5 个子图，分别为 CWmin 为 31、63、127、255、511 时节点 1 到节点 4 连接的吞吐量。比较图 5.1 的 5 个子图可看出，当 31<CWmin<255 时随着 CWmin 的增加，吞吐量到达或接近 0 的时间点越来越少；当 CWmin 大于或等于 255 后，吞吐量已没有达到或接近 0 的时间点，TCP 趋于稳定。但当 CWmin 为 511 时的平均吞吐量水平显然低于 CWmin 为 255 时的平均吞吐量水平。这是因为 CWmin 的值也不能过大，否则即使目的节点处的数据传输完毕后，源节点

仍在退避，这样只会浪费网络的传输时间。

5.1.2 改进对平均吞吐量的影响

计算出各种最小竞争窗口在各个跳数下的平均吞吐量，其结果如图 5.2 所示。

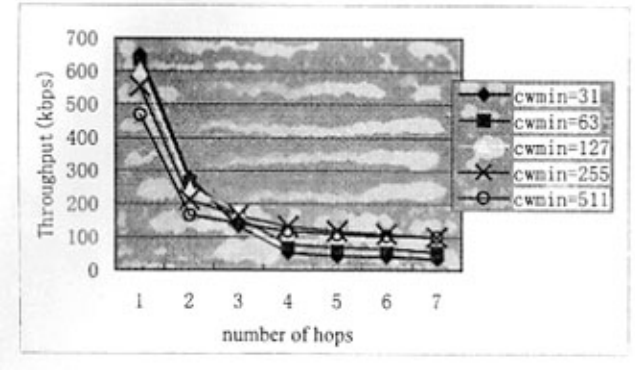


图 5.2 不同跳数下最小竞争窗口对平均吞吐量的影响

hops	cm=31	cm=63	cm=127	cm=255	cm=511
1	650.67	634.80	604.80	551.20	468.00
2	277.09	265.64	244.80	211.24	165.93
3	133.08	158.83	176.60	164.44	142.29
4	52.96	78.37	109.93	132.45	119.12
5	43.10	68.27	100.49	117.77	108.47
6	40.92	65.40	94.52	108.65	103.58
7	34.44	54.67	95.47	99.53	100.00

表 5.1 增加的 CWmin 值对各个跳数的连接的吞吐量的影响(kbps)

当跳数较小时（如 1、2），吞吐量随着 CWmin 值的增加的变化不大，但当 CWmin 的值过大（如 511）时，吞吐量降低的较大，因为在没有冲突的情况下增加 CWmin 的值只会浪费时间。当跳数大于 2 后，CWmin 增加后的吞吐量相对于原始的 CWmin(31)的吞吐量都有所提高。跳数小于或等于 3 时，CWmin=127 时吞吐量的增加较稳定；跳数大于 3 后，CWmin=255 时吞吐量在各种跳数下均有较稳定的增加。这可从表 5.1 中更加清楚地看出，表中 cm 表示最小竞争窗口 CWmin。因为初始退避时间虽不能太小，但也不能太大，否则即使目的节点处的数据传输完毕后，源节点仍在退避。综合总体的情况看，在本实验中，当 CWmin=127 时，性能表现较为稳定。因此在本文所述的 ad hoc 网络环境中最小

竞争窗口最好选为 127。

5.2 改进退避机制

由 3.2 可知，MAC 层的冲突和暴露站问题会使 TCP 不稳定。若一个节点多次尝试接入信道失败，它就认为链路中断，从而启动漫长而复杂的路由重建过程。导致数据不能快速传输，吞吐量下降。导致 TCP 性能不稳定的另外一个原因是，IEEE802.11 的退避机制使信道很容易被捕获，同时捕获问题也会导致多条 TCP 连接的严重的不公平性。针对这个问题，本文中提出了一种改进机制，并对改进后的 TCP 性能进行仿真分析，验证了其对 TCP 的性能确实有所改善。

5.2.1 TCP 连接的公平性分析

本部分，检验公平性问题。在 ad hoc 网络中，TCP 业务可能遭受严重的不公平性，即使在 2 个具有相同跳数的连接之间，也存在不公平性[33]。在我们的仿真试验中，甚至出现一条 TCP 连接被完全关闭的情况，即使该连接比竞争连接建立的早得多。这个问题源于多跳无线链路的 IEEE802.11 MAC 层。

本实验中的仿真环境和网络参数的设置如实验 4，即网络拓扑结构为静态的由 8 个点组成的链式拓扑，在此实验中，建立 2 个 TCP 连接。第一个是从节点 6 到节点 4，从 10.0s 开始传输，称为连接 1；第二个是从节点 2 到节点 3，在 30.0s 后开始传输，称为连接 2。整个实验在 120.0s 结束。此实验称为实验 5。

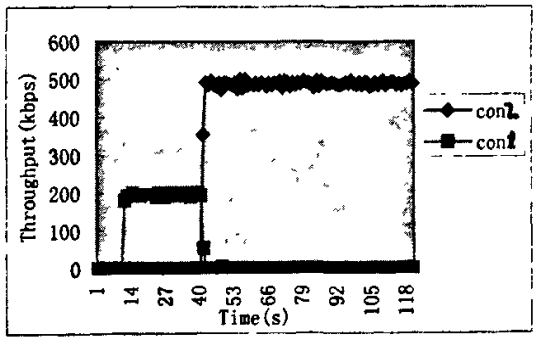


图 5.3 ad hoc 网络中严重的不公平性

仿真结果如图 5.3 所示，图中 con1 和 con2 分别代表连接 1 和连接 2。连接 1 是一条两跳 TCP 连接，它在 10.0s 传输开始后有大约 200kbps 的吞吐量。但是当连接 2 从 30.0s 传输开始后一直到仿真结束的时间内，连接 1 的吞吐量为 0，甚至没有使它重新开始的机会。此时，网络的总吞吐量完全属于连接 2——在从 30.0s 到 120.0s 的存在时间内大约为 500kbps。这就是严重的不公平性，即使连接 1 传

输开始的较早，仍然被完全关闭。

由于MAC层的退避机制和TCP的拥塞控制机制的相互影响而产生的信道捕获也严重的影响了TCP的性能，这不仅加重了TCP的不稳定性，而且导致了TCP的不公平性。问题的关键在于无线MAC协议。IEEE802.11使用的是指数退避机制：成功传输之间的退避用的是最小竞争窗口CWmin，而每失败一次竞争窗口就加倍。这样，退避总是对最后发送成功的站有利。在高度移动的网络中，节点常移出各自的通信范围，节点很少有机会捕获信道。捕获信道问题大多数出现在当网络节点是静止的或进行小的移动时，即节点较长时间的处于各自的通信范围之内。捕获问题的影响很严重，因为当节点在一段时间内无法接入信道，它们会产生路由错误包，对TCP业务来说，这使得没竞争得到信道的连接发生重传超时，从而吞吐量大大减少。在当前这一代在给定的一个〈源，目的〉对间用同一路由传输前向和反向业务的路由协议中，发生信道捕获的可能性很大。对TCP来说，这意味着前向的数据包和反向的ACK竞争接入同一共享介质，常常引起某一条连接的ACK报文段不能到达源节点，这样，TCP执行拥塞控制算法。可见，TCP数据包常捕获信道妨碍了ACK包到达源节点，甚至使得一条连接被关闭，且这个问题在有多个TCP流时会更严重。

5.2.2 改进公平性和性能

对这个问题的解决方法是把成功传输之间的时间间隔增大，使曾传输失败的连接有较大的机会接入信道。减少某一节点连续重试7次仍无法接入信道的概率，避免进行费时和复杂的路由发现，使TCP吞吐量稳定，并可使同一网络中的多个TCP连接的接入信道的机会平等。但成功传输之间的时间间隔也不能太大，否则浪费时间且使总体的吞吐量降低。因此，改进的方案是使成功传输的退避竞争窗口在两个较大的值之间循环。经过多次仿真试验，得出使竞争窗口在511和1023之间循环对TCP的信道捕获问题有所减轻，增加了TCP连接的公平性。改进后重复实验5，其仿真结果如图5.4所示。图中con1和con2分别代表连接1和连接2。由图可知，改进后，连接1和2的吞吐量都相对较稳定，没有出现一个连接完全关闭另一个连接的情况。

由前面的分析可知，此机制对TCP的不稳定性将有所缓解，同时还可提高多跳时TCP的吞吐量。为了证实，对实验4重新进行仿真。节点1到4的连接的吞吐量如图5.5所示。由图可看出，吞吐量已较稳定，没有降到或接近0的点。跳

数对平均吞吐量的影响如图 5.6 所示。图中 traditional 和 modified 分别代表传统的

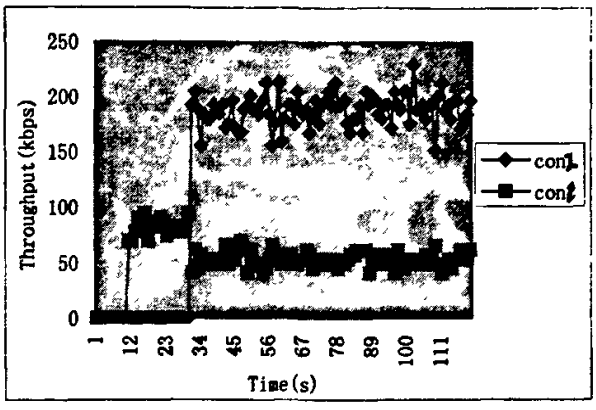


图 5.4 改进后 TCP 的公平性

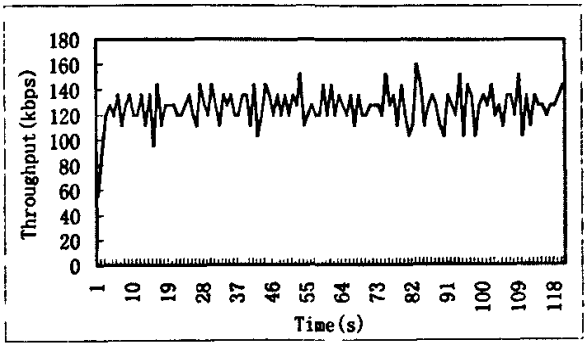


图 5.5 改进后节点 1 到 4 的吞吐量

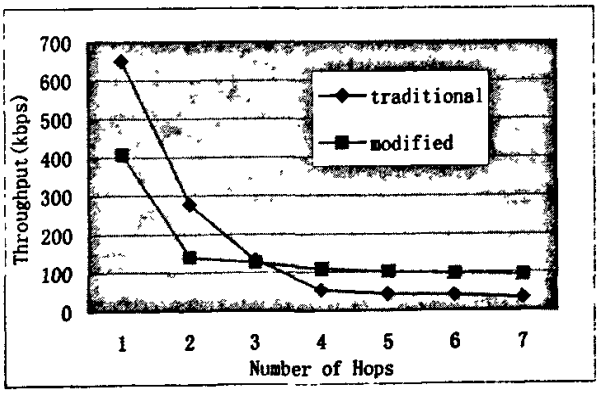


图 5.6 改进后不同的跳数下的平均吞吐量

和改进了的方案。当跳数较小时，改进的机制的平均吞吐量有所减小，这是因为跳数较少时，可认为信道捕获问题不大，而较大的竞争窗口会导致传输等待较长的时间。但是当跳数大于等于 3 后，改进的机制获得较大的平均吞吐量。在一般

的多跳网络中，特别是在节点数较多时，连接的跳数一般都大于 3。因此可认为改进的机制确实可提高 TCP 的性能。

本部分对退避机制进行的改进，减轻了信道捕获问题，提高了 TCP 吞吐量，同时改善了 TCP 连接的公平性。

5.3 小结

如前 4.1 所述，已提出的 MAC 层的改进是针对 ad hoc 的移动性带来的路由中断引起的 TCP 吞吐量恶化问题所进行的改进，本部分是针对 ad hoc 网络的多跳特性引起的 TCP 性能恶化问题对 MAC 层进行了改进，既提高了吞吐量，又比较简单，没有增加移动节点的能源消耗和网络负荷，同时还解决了多个连接之间的不公平性问题，比较适合移动性不大的 ad hoc 网络。但存在的不足是此改进对 ad hoc 网络的移动性带来的 TCP 性能的恶化没有多少改进。这是将来的工作。

第六章 总结与展望

本文首先介绍了 ad hoc 网络的特点、应用及其面临的问题。接着对各种版本的 TCP 的拥塞控制机制做了详细的研究，其中有 Tahoe、Reno、NewReno、SACK、Vegas，还有一种针对无线网络提出的 DA 机制。接着介绍了网络仿真工具 NS2 的开发背景，运行平台及其软件包的组成，并简单地介绍了 NS 的使用。并利用 NS 对各种版本的 TCP 在多跳和移动两种情况下分别进行了仿真分析，并分析产生这种结果的原因。从仿真结果中可以看出随着连接地跳数的增加和节点平均移动速度的加快，TCP 的吞吐量显著下降，只不过各种 TCP 下降的程度不一样，从而得出结论，在本文所设的网络环境下，Vegas 和 DA 机制较适合多跳网络，但 Vegas 不适合移动性较高的 ad hoc 网络，而 DA 机制又表现出了较好的性能。因此就对 DA 进行了更深一步的研究，得出结论除 Vegas TCP 的其它版本在使用 DA 机制时应选择适当的窗口大小，同时 DA 用来传小文件的时延相对来说较大，并针对此问题提出了改进并进行了验证。接着，对已提出的为提高 TCP 吞吐量而做出的对 MAC 层，网络层，传输层以及其他方面的改进进行了总结描述并做出了自己的评价，然后就提出了两种自己的改进，第一种可以解决 TCP 的不稳定性问题，从而提高吞吐量。第二种改进解决了 MAC 层的不公平性问题，同时又可减轻 TCP 的不稳定性问题。对这两种改进方案都用仿真工具验证了它们的有效性。

以上两种改进都是为了减少 ad hoc 网络的多跳特点带来的问题，而已证明 ad hoc 网络中节点的移动也会带来 TCP 性能的恶化，目前针对此问题的研究及改进相对较少，未来的工作就是要在已有工作的基础上继续研究，进一步解决以下问题：

1. 逐步提高无线链路的可靠性，减少由于高误码率带来的 TCP 性能的下降。
2. 优化 MAC 层的竞争接入机制，逐步减少 MAC 层的暴露站和隐藏站问题给 TCP 的不稳定性带来的影响。
3. 使路由协议更适合移动性网络，减少路由发现和路由重建的时延，从而使 TCP 的拥塞控制更有效地运行。
4. 针对 ad hoc 网络的移动性改进 TCP 自身的拥塞控制机制，从而减少由于

节点的移动而不必要的运行拥塞控制的次数。

5. 加强层与层之间的通信以及相互合作，使 TCP 区分网络拥塞和其它原因造成的丢包，提高吞吐量。

参考文献

[1] 王珍，刘飒，无线 ad-hoc 网络及其关键技术，电信科学，2003 年第 4 期，10~13。

[2] 梅辉，罗文茂，无线 ad hoc 网络，现代通信，2003 年第 9 期，6~8。

[3] 方旭明，移动 ad hoc 网络研究及发展现状，数据通信，2003 年第 4 期，15~23。

[4] W. R. Stevens，TCP/IP Illustrate, Volumn 1: The protocol， Addison Wesley, 1994

[5] M. Allman，V. Paxson，W. Stevens，RFC2581, TCP Congestion Control, 1999.04。

[6] W. Stevens，RFC2001, TCP Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery Algorithms，1997.01。

[7] S. Floyd，T. Henderson，RFC2582，The NewReno Modification to TCP's Fast Recovery Algorithm，1999.04。

[8] M. Mathis，S. Floyd，A. Romanow，RFC2018，TCP Selective Acknowledgment Options，1996.10

[9] R. Braden，RFC1122，Requirements for Internet Hosts -- Communication Layers，1989.10。

[10] Brakmo, L.S.，Peterson, L.L.，TCP Vegas: End to End Congestion Avoidance on a global Internet, IEEE journal ON SELECTED AREAS IN COMMUNICATIONS, vol.13, NO.8, 1995.10, 1465~1480。

[11] The cmu monarch project， http://www.monarch.cs.cmu.edu/

[12] K. Fall and K. Varadhan, ns Notes and Documentation, LBNL, Aug 1998. http://www-mash.cs.berkeley.edu/ns/

[13] （美）Brent B.Welch 著，韦尔奇，王道义，乔陶鹏等译，TCL/TK 组合教程（第二版），电子工业出版社，北京，2001.

[14] 郑莉，董渊，C++语言程序设计，清华大学出版社，北京，2001。

[15] Wei-Peng Chen; Hou, J.C ， Dynamic, ad-hoc source routing with connection-aware link-state exchange and differentiation，Global Telecommunications Conference, 2002. GLOBECOM '02. IEEE , Volume: 1 , 17-21 Nov. 2002。

[16] S. Xu, T. Saadawi, "Does the IEEE 802.11 MAC Protocol Work Well in Multi-hop Ad Hoc Networks", IEEE Communication Magazines, June 2001, 130~137。

[17] L. S. Brakmo, S. W. O'Malley, L. Peterson, "TCP Vegas: New Techniques for Congestion Detection and Avoidance", ACM SIGCOMM Computer Communication Review, London, Oct 1994, vol 24, issue 4, 24~35。

[18] J. Liu, S. Singh, ATCP: TCP for Mobile Ad Hoc Networks, IEEE Journal of Selected Areas in Communications, July 2001, 1300~1315。

[19]Gavin Holland, Nitin Vaidya, Impact of Routing and Link Layer on TCP Performance in Mobile Ad Hoc Networks, Wireless Communications and Networking Conference, 1999. WCNC. 1999 IEEE, 21-24 Sept.1999 vol.3 1323-1327.

[20]Carlos de M. Cordeiro, Samir R. Das, Dharma p. Agrawal. COPAS: Dynamic Contention-Balancing to Enhance the Performance of TCP over Multi-hop Wireless Networks[C]. Computer Communications and Networks, Eleventh International Conference, 2002.10.14-16: 382~387

[21]李云,陈前斌,隆克平等. 无线自组织网络中 TCP 稳定性的分析及改进[J]. 软件学报. 2003; 14(6): 1178~1186

[22]Jin-Hee Choi; Chuck Yoo; TCP-aware source routing in mobile ad hoc networks; Computers and Communication, 2003.（ISCC 2003）.Proceedings. Eighth IEEE International Symposium on , 30 June-3 July 2003 Pages: 69-74 vol.1

[23]Feng Wang, Yongguang Zhang. Improving TCP Performance over Mobile Ad-Hoc Networks with Out-of-Order Detection and Response[C]. Proceedings of the 3rd international symposium on Mobile ad hoc networking&computing. 2002.6: 217~225

[24]JianXin Zhou; BingXin Shi; Ling Zou, Improve TCP performance in ad hoc network by TCP-RC , Personal, Indoor and Mobile Radio Communications, 2003. PIMRC 2003. 14th IEEE Proceedings on , vol 1. 7-10 Sept. 2003 Pages: 216-220

[25]Dong Sun, Hong Man. ENIC-An Improved Reliable Transport Scheme for Mobile Ad Hoc Networks. Global[C] Telecommunications Conference, GLOBALCOM'01, IEEE, 2001.11.25-29(5): 2852~2856

[26]Holland G, Vaidya N. Analysis of TCP performance over mobile ad hoc networks[J]. Wireless Networks, 2002. 8(2): 275~288

[27]Muset gunes, Donald Vlahovic. The Performance of the TCP/RCWE Enhancement for Ad Hoc Networks[C]. Computers and Communications, ISCC 2002. Seventh

International Symposium, 2002.6; 1-4: 43~48

[28]Dongkyun Kim, C.-K.Toh, Yanghee Choi. TCP-Bus: improving TCP performance in wireless ad hoc networks[C]. Communications, 2000 IEEE International Conference, 2000.6.18-22(3): 1707~1713

[29]Swastik Kopparty, Srikanth V. Krishnamurthy, Michalis Faloutsos, Satish K. Tripathi. Split TCP for Mobile Ad Hoc Networks[C]. Global Telecommunications Conference, GLOBECOM '02, IEEE, 2001.11.17-21(1): 138-142

[30]Chengzhou Li, Symeon Papavassiliou. The Link Signal Strength Agent(LSSA) Protocol for TCP implementation in wireless mobile ad hoc networks[C]. Vehicular Technology Conference, VTC 2001 Fall, IEEE VTS 54th , 2001.10.7-11(4): 2528~2532

[31]Jian Liu, Suresh Singh. ATCP: TCP for Mobile Ad Hoc Networks[J]. Selected Areas in Communications, IEEE journal, 2001; 19(7): 1300~1315。

[32] IEEE STD 802.11, Wireless LAN Media Access Control (MAC) and Physical Layer (PHY) Specification, 1999.

[33]Rui Jiang, Vikram Gupta, Chinya V. Ravishankar; Interaction Between TCP and the IEEE802.11 MAC Protocol; DARPA Information Survivability Conference and Exposition, 2003. Procedings, Volume: 1, 22-24 April 2003 Pages 273-282 vol. 1

致 谢

首先感谢张晓敏副教授在我研究生阶段学习、生活中给予的谆谆教诲和悉心指导。正是张老师的教导和鼓励，使我明白了科学研究的内涵，使我初步掌握了科学研究的方法，并使我确定了从事科学研究的人生目标。张老师不仅为我指明了研究的方向，还在很多方面给我以启迪，为我的成长倾注了许多心血，她的教诲是我人生一笔宝贵的财富。

同时，我要感谢实验室的同学和师哥师姐、师弟师妹们，他们在我的学习和生活中给予了很大的帮助，他们将自己的设计开发经验无私的传授给我，使我获益匪浅。实验室里的学术讨论营造了良好的学术氛围，使我在科研方法上得到很多指导，开阔了视野。

感谢母校对我的教育和培养，我将永远记住在山东大学度过的这段美好时光。

最后，感谢我的家人对我学业上莫大的支持和鼓励。

攻读学位期间发表的学术论文目录

王伟伟，张晓敏； 《Ad Hoc 网络环境下的 TCP 性能分析》；
《计算机工程与应用》；2004 年 9 月，第 40 卷 第 27 期(总第 478 期)，154~156。

ad hoc网络环境下的TCP性能分析及改进

作者：王伟伟
学位授予单位：山东大学

参考文献(33条)

1. 王珍. 刘(友风) 无线Ad-hoc网络及其关键技术[期刊论文]-电信科学 2003(4)

2. 梅辉. 罗文茂 无线adhoe网络 2003(09)

3. 方旭明 移动Ad Hoc网络研究与发展现状[期刊论文]-数据通信 2003(4)

4. W R Stevens TCP/IP Illustrate,Volumn 1:The protocol 1994

5. M Allman. V Paxson. W Stevens TCP Congestion Control 1999

6. W Stevens TCP Slow Start,Congestion Avoidance,Fast Retransmit,and Fast Recovery Algorithms 1997

7. S Floyd. T Henderson The NewReno Modification to TCP's Fast Recovery Algorithm 1999

8. M Mathis. S Floyd. A Romanow TCP Selective Acknowledgment Options 1996

9. R Braden Requirements for Interact Hosts-Communication Layers 1989

10. Brakmo L S. Peterson L L TCP Vegas:End to End Congestion Avoidance on a global Intemet 1995(08)

11. The cmu monarch project

12. K Fall. K Varadhan ns Notes and Documentation 1998

13. Brent B Welch. 韦尔奇. 王道义. 乔陶鹏 TCL/TK组合教程 2001

14. 郑莉. 董渊 C++语言程序设计 2001

15. Wei-Peng Chcn. Hou J C Dynamic,ad-hoc source routing with connection-aware link-state exchange and differentiation 2002

16. S Xu. T Saadawi Does the IEEE 802.11 MAC Protocol Work Well in Multi-hop Ad Hoc Networks 2001(06)

17. L S Brakmo. S W O'Malley. L Peterson TCP Vegas:New Techniques for Congestion Detection and Avoidance 1994(04)

18. J Liu. S Singh ATCP:TCP for Mobile Ad Hoc Networks 2001

19. Gavin Holland. Nitin Vaidya Impact of Routing and Link Layer on TCP Performance in Mobile Ad Hoc Networks 1999

20. Carlos de M Cordeiro. Samir R Das. Dharma p Agrawal COPAS:Dynamic Contention-Balancing to Enhance the Performance of TCP over Multi-hop Wireless Networks 2002

21. 李云. 陈前斌. 隆克平. 吴诗其 无线自组织网络中TCP稳定性的分析及改进[期刊论文]-软件学报 2003(6)

22. Jin-Hee Choi. Chuck Yoo TCP-aware source routing in mobile ad hoc networks 2003

23. Feng Wang. Yongguang Zhang Improving TCP Performance over Mobile Ad-Hoc Networks with Out-of-Order Detection and Response 2002

24. JianXin Zhou. BingXin Shi. Ling Zou Improve TCP performance in ad hoc network by TCP-RC[会议论文] 2003

25. Dong Sun. Hong Man ENIC-An Improved Reliable Transport Scheme for Mobile Ad Hoc Networks.Global 2001

26. Holland G. Vaidya N Analysis of TCP performance over mobile ad hoc networks[外文期刊] 2002(02)

27. Muset gunes. Donald Vlahovic The Performance of the TCP/RCWE Enhancement for Ad Hoc Networks 2002

28. Dongkyun Kim. C -K Toh. Yanghee Choi TCP-Busimproving TCP performance in wireless ad hoc networks 2000

29. Swastik Kopparty. Srikanth V Krishnarnurthy. Michalis Faloutsos. Satish K.Tripathi Split TCP for Mobile Ad Hoc Networks 2001

30. Chengzhou Li. Symeon Papavassiliou The Link Signal Strength Agent(LSSA)Protocol for TCP

implementation in wireless mobile ad hoc networks 2001

31. Jian Liu. Suresh Singh ATCP:TCP for Mobile Ad Hoc Networks 2001(07)

32. IEEE STD 802.11,Wireless LAN Media Access Control(MAC)and Physical Layer(PHY)Specification 1999

33. Rui Jiang. Vikram Gupta. Chinya V Ravishankar Interaction Between TCP and the IEEE802.11 MAC Protocol 2003