电 子 科 技 大 学

UNIVERSITY OF ELECTRONIC SCIENCE AND TECHNOLOGY OF CHINA

硕士学位论文

**MASTER THESIS**



论文题目 **SCPS-TP协议中拥塞控制关键技术研究**

学科专业 **计算机应用技术**

学 号 **201221060427**

作者姓名  **刘斌**

指导教师  **罗光春** **教授**

分类号 密级

UDC注1

学 位 论 文

**SCPS-TP协议中拥塞控制关键技术研究**

（题名和副题名）

**刘斌**

（作者姓名）

指导教师 **罗光春 教授**

**电子科技大学 成 都**

（姓名、职称、单位名称）

申请学位级别  **硕士** 学科专业 **计算机应用技术**

提交论文日期 论文答辩日期

学位授予单位和日期 **电子科技大学** **年 月 日**

答辩委员会主席

评阅人

注1：注明《国际十进分类法UDC》的类号。

|  |
| --- |
| **RESEARCH ON Congestion Control**  **of SCPS-TP** |

A Master Thesis Submitted to

University of Electronic Science and Technology of China

Major: **Computer Application Technology**

Author: **LiuBin**

Advisor: **Luo GuangChun Professor**

School: **School of Computer Science & Engineering**

独创性声明

本人声明所呈交的学位论文是本人在导师指导下进行的研究工作及取得的研究成果。据我所知，除了文中特别加以标注和致谢的地方外，论文中不包含其他人已经发表或撰写过的研究成果，也不包含为获得电子科技大学或其它教育机构的学位或证书而使用过的材料。与我一同工作的同志对本研究所做的任何贡献均已在论文中作了明确的说明并表示谢意。

作者签名： 日期： 年 月 日

论文使用授权

本学位论文作者完全了解电子科技大学有关保留、使用学位论文的规定，有权保留并向国家有关部门或机构送交论文的复印件和磁盘，允许论文被查阅和借阅。本人授权电子科技大学可以将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、缩印或扫描等复制手段保存、汇编学位论文。

（保密的学位论文在解密后应遵守此规定）

作者签名： 导师签名：

日期： 年 月 日

# 摘 要

目前天基和地基网络的通信协议并没有统一的设计和建设，这在星地和星间通信中尤其明显。各地面通信系统在互联网的迅速发展下，已基本全部采用TCP/IP，以支持各种各样的应用要求，实现了地面通信协议的大统一。而各卫星系统在任务多样复杂、缺乏统一标准的情况下，往往采用各自独立开发的通信协议，不能很好地与地面应用网络相结合，天地各网络间的通信也必须通过网间的协议翻译转换来实现。 SCPS 协议在上世纪末提出，其研究目的是在兼容地面互联网的基础上支持可靠的高速率数据传输，支持并行任务多节点之间网络路由功能选择，并与地面具有良好互操作性等。SCPS协议解决了空间数据通信中存在的诸多问题，如：信号强度弱、信道噪声大、传输时延大、带宽时延积大、多普勒频移较大以及链路容易中断等问题。经过多年的发展，SCPS 协议在空间通信网络中得到了广泛的应用，同时也暴露出了一些亟待解决的问题。

本文首先总结了各国在SCPS-TP协议的应用以及改进方面所做出的工作。接着详细描述了SCPS-TP协议对于传统TCP协议拥塞控制算法的改进。通过NS2模拟仿真，指出了Vegas协议中包突发现象以及过早进入拥塞避免阶段对于传输性能的影响，在拥塞避免阶段，Vegas cwnd线性增长，不能尽快利用带宽。为解决上述问题，本文第三章基于Vegas算法进行了两方面的改进：针对慢启动阶段包突发现象，使得Vegas算法过早结束慢启动阶段，本节算法采用了每个RTT增加一半cwnd的策略。针对拥塞避免阶段，窗口增长速度不能尽快充分利用带宽，本节算法采用了自适应的窗口增长策略。通过NS2的对比仿真实验，可以看出，在短时间内，改进后的Vegas算法吞吐量要明显高于传统Vegas算法。第四章，针对慢启动阶段Vegas协议在慢启动阶段不能有效利用带宽，参照TCP Westwood算法，采用带宽估计算法有效利用带宽。针对Vegas不能分辨链路拥塞还是路哟改变，采用了路径监视的策略。更加精准地反映了网络状态，提高了卫星资源利用率。仿真结果表明，新算法在系统吞吐率上较Vegas算法有明显提高，并表现出了较好的公平性。

最后，基于CCSDS AOS标准和SCPS-TP 协议搭建了空间网络数据传输的一个系统。通过TCP分段技术，使得空间内部网络能够使用传统TCP协议拥塞控制算法，而空间链路间则使用改进的Vegas算法。这样，减少了不必要损耗，并且能更快速地传输文件。在系统中传输文件，尤其是小文件，我们可以看出，改进后的算法要优于传统Vegas算法。

**关键词**：CCSDS，SCPS-TP，Vegas算法，拥塞控制

# ABSTRACT

At present, the communication protocol of the space-based and ground-based network design and construction are not unified, which is particularly evident in the satellite and inter satellite communication. The rapid development of the Internet the ground communication system, has been basically all use TCP/IP, to support the application demands of all kinds, the realization of a large unified ground communication protocol. Each satellite system in the complex task and the lack of unified standard case, often using a communication protocol developed independently, not very good with ground application network combination, communication world between the various networks must also protocol translation from the network to realize. SCPS protocol is presented at the end of last century, the purpose of the research is to support high rate data transmission reliable based on compatible ground to the Internet, support multi node parallel tasks between network routing function selection, and has good interoperability and ground. The SCPS protocol solves many problems exist, such as spatial data communication: weak signals, channel noise, large transmission delay, bandwidth delay product is large and easy to interrupt the question big link, Doppler frequency shift. After years of development, the SCPS protocol has been widely used in space communication network, but also exposed some problems to be solved.

This paper first summarizes the various countries in the SCPS-TP protocol's application and improvement to which aspects of the work. Then a detailed description of the SCPS-TP protocol to improve the traditional TCP congestion control algorithm. The simulation by NS2, points out that the Vegas protocol packet burst phenomenon and premature enters the congestion avoidance phase to affect transmission performance, in congestion avoidance phase, Vegas cwnd linear growth, not as soon as possible the use of bandwidth. In order to solve the above problem, in the third chapter, based on the Vegas algorithm is improved in two aspects: in the slow start phase packet burst phenomenon, so that the Vegas algorithm the premature end of the slow start phase, this section algorithm uses each RTT half of the increase in the CWnd strategy. According to the congestion avoidance phase, the window growth does not as soon as possible to make full use of bandwidth, this section algorithm adopts adaptive window growth strategy. Through simulation, NS2 can see, in a short period of time, the Vegas algorithm improved throughput are significantly higher than the traditional Vegas algorithm. The fourth chapter, aiming at the slow start phase Vegas protocol can not in slow start phase, the effective use of bandwidth, according to TCP Westwood algorithm, using bandwidth estimation algorithm is effective use of bandwidth. In view of the Vegas can not distinguish the link congestion or road yo change, using path monitoring strategy. To more accurately reflect the state of the network, improves the utilization rate of resource satellite. The simulation results show that, the new algorithm in system throughput than Vegas algorithm is improved obviously, and showed better fairness.

Finally, the CCSDS AOS standard and the SCPS-TP protocol based on building a system of space data transmission network. Through the TCP segment technology, make the space inside the network to use traditional TCP congestion control algorithm, and the space link between the use of the improved Vegas algorithm. Thus, to reduce unnecessary losses, and more rapid transmission of documents. Transfer files in the system, especially the small files, we can see that the improved algorithm is superior to the traditional Vegas algorithm

Key Words：CCSDS，SCPS-TP，TCP Vegas，Congestion Control

目 录

[摘 要 I](#_Toc409641935)

[ABSTRACT II](#_Toc409641936)

[第一章 绪论 1](#_Toc409641937)

[1.1 本文研究背景与意义 1](#_Toc409641938)

[1.2 国内外研究现状 2](#_Toc409641939)

[1.3 本文研究内容 5](#_Toc409641940)

[第二章 SCPS-TP协议拥塞控制算法分析 6](#_Toc409641941)

[2.1 TCP协议拥塞控制概述 6](#_Toc409641942)

[2.1.1 慢启动阶段(Slow start） 6](#_Toc409641943)

[2.1.2 拥塞避免阶段(Congestion avoidance） 7](#_Toc409641944)

[2.1.3 快速重传阶段(Fast retransmit） 7](#_Toc409641945)

[2.1.4 快速恢复阶段(Fast recovery） 7](#_Toc409641946)

[2.2 SCPS-TP对于TCP协议改进 8](#_Toc409641947)

[2.2.1 数据丢失的改进 9](#_Toc409641948)

[2.2.2 网络拥塞的识别和处理 10](#_Toc409641949)

[2.2.3 链路恶化处理 10](#_Toc409641950)

[2.2.4 链路中断 11](#_Toc409641951)

[2.2.5 误码的识别和处理 12](#_Toc409641952)

[2.2.6 带宽容量有限的改进 12](#_Toc409641953)

[2.2.7 针对长往返延迟的改进 16](#_Toc409641954)

[2.2.8 链路不对称的改进 16](#_Toc409641955)

[2.2.9 SCPS-TP的扩展能力 17](#_Toc409641956)

[2.3 目前基于SCPS-TP的改进策略 19](#_Toc409641957)

[2.4 本章小结 22](#_Toc409641958)

[第三章 对Vegas拥塞控制算法的改进 23](#_Toc409641959)

[3.1 Vegas拥塞控制算法概述 23](#_Toc409641960)

[3.2 对Vegas协议的改进 24](#_Toc409641961)

[3.2.1 慢启动阶段改进 24](#_Toc409641962)

[3.2.2 拥塞避免阶段改进 25](#_Toc409641963)

[3.3 模拟方案设计 28](#_Toc409641964)

[3.3.1 NS2仿真工具介绍 28](#_Toc409641965)

[3.3.2 具体实验步骤 29](#_Toc409641966)

[3.4 仿真结果与分析 30](#_Toc409641967)

[3.4.1 NS2拓扑结构及代码修改 30](#_Toc409641968)

[3.4.2 NS2仿真结果与分析 33](#_Toc409641969)

[3.5 本章小结 36](#_Toc409641970)

[第四章 基于动态带宽估计拥塞算法改进 37](#_Toc409641971)

[4.1 卫星网络中TCP-vegas算法中存在的问题 37](#_Toc409641972)

[4.1.1 Vegas慢启动阶段不能有效利用带宽 37](#_Toc409641973)

[4.1.2 Vegas算法不能区分链路拥塞还是路由改变 38](#_Toc409641974)

[4.2 基于带宽估计的慢启动算法改进 39](#_Toc409641975)

[4.3 基于路径监视的BaseRTT更新算法 41](#_Toc409641976)

[4.4 仿真结果与分析 43](#_Toc409641977)

[4.4.1 代码编写 43](#_Toc409641978)

[4.4.2 运行结果分析 45](#_Toc409641979)

[4.5 本章小结 47](#_Toc409641980)

[第五章 系统设计与实现 48](#_Toc409641981)

[5.1 需求背景与设计目标 48](#_Toc409641982)

[5.1.1 需求背景 48](#_Toc409641983)

[5.1.2 设计目标 49](#_Toc409641984)

[5.2 关键技术实现 49](#_Toc409641985)

[5.2.1 IPv4/IPv6 over CCSDS实现 49](#_Toc409641986)

[5.2.2 AOS调度算法与实现 51](#_Toc409641987)

[5.3 软件设计实现 52](#_Toc409641988)

[5.3.1 系统总体设计 52](#_Toc409641989)

[5.3.2 主要工作流程 55](#_Toc409641990)

[5.4 系统功能测试 60](#_Toc409641991)

[5.4.1 系统配置及编译 60](#_Toc409641992)

[5.4.2 系统功能改进及运行结果 62](#_Toc409641993)

[5.5 本章小结 65](#_Toc409641994)

[第六章 结论与展望 66](#_Toc409641995)

[6.1 本文总结 66](#_Toc409641996)

[6.2 展望 66](#_Toc409641997)

[致 谢 68](#_Toc409641998)

[参考文献 69](#_Toc409641999)

[攻读硕士学位期间取得的成果 72](#_Toc409642000)

# 绪论

## 本文研究背景与意义

目前天基和地基网络的通信协议并没有统一的设计和建设，这在星地和星间通信中尤其明显。各地面通信系统在互联网的迅速发展下，已基本全部采用TCP/IP，以支持各种各样的应用要求，实现了地面通信协议的大统一。而各卫星系统在任务多样复杂、缺乏统一标准的情况下，往往采用各自独立开发的通信协议，不能很好地与地面应用网络相结合，天地各网络间的通信也必须通过网间的协议翻译转换来实现。随着空间通信的日益发展，各国空间组织为解决空间协议标准等不统一问题，成立了空间数据系统咨询委员会(CCSDS,The Consultative Committee for Space Data Systems)。

CCSDS现在有11个成员、28个观察员、140个非正式成员，已经发展成为世界各空间组织相互支持的技术协商性机构，CCSDS一直致力于研究天地各通信网络协议的统一与推广。其任务是负责开发和采纳适合于航天测控和数传系统的各种通信协议和数据处理规范，以适应航天器复杂化的发展趋势，以及满足空间资源和平利用、加强国际合租的需要。CCSDS的相关工作反映了世界空间数据系统的最新技术发展动态。成立30年来它推出了一整套技术建议书，其中一部分已直接转化为国际标准化组织(ISO)的正式国际标准，并已广泛地应用于各国际空间团体，如美国国家航空航天局（NASA，National Aeronautics and Space Administration）喷气推进实验室、美国的Lockheed Martin导弹与航天公司、ESA欧洲空间技术中心等。CCSDS标准不仅为实现开放互联的国际空间数据系统网奠定了技术基础，而且还反映了世界空间数据系统的最新技术发展动态。

20世纪80年代早期，CCSDS发布了分包遥测协议的国际标准，它利用称为源包的可变长度数据单元以更有效的方式传送遥测数据。这个标准被许多太空项目用于共享太空船上和地面的数据处理设备。随后，CCSDS又制定了另外一个关于遥控数据传输的国际标准，这个标准用来从地面向航天器传送分包遥控(TC，Telecomm and Packet)来发送命令。通过一个可变长度的非持续数据流来传送TC包，以便从地面向航天器上的各种设备和子系统发送命令。

20世纪80年代后期，为了满足高级在轨系统如国际空间站的要求，CCSDS发布了第3个标准高级在轨系统(AOS，Advanced Orbiting System)。AOS在分包遥测的基础上发展而来，将分包遥测标准扩展到能传送各种类型的在线数据(如音频和视频数据)，它在星—地、地—星、星—星链路上都可以使用。

空间探索的深入发展迫切要求与之适应的空间通信网络的构建。随着空间探测对象的不断增加和探测范围的扩大，空间探索越来越需要不同国家之间的通信合作，这使得传统点对点的通信模式逐步发展为一点对多点甚至多点间的网络通信模式。同时，地面Intemet的发展，特别是覆盖范围的扩展，客观上要求空间节点的支持。这两点原因使得天地一体化网络成为未来网络发展的重要趋势。与传统的地面Intemet网络不同，空间通信网络是涵盖卫星、航天器、空间站以及各类探测传感器等通信载体与地面网络融合的复杂的一体化异构型网络。而且，空间通信网络由于空间环境的限制，存在着传播延时长、误码率高、带宽不对称、连接非持续性、网络拓扑结构动态变化等特点。针对上述特性，CCSDS于1999年规范了一套空间通信协议(Space CommunicatiOll Protocol Specification，SCPS)SCPS以地面网络普遍应用的TCP/IP四层分层结构为模型，在局部兼容Intemet基础上，为适应空间网络特性而进行了适当的剪裁与扩充。经过了7年的发展，SCPS在空间通信网络中得到了广泛的应用，同时也暴露出了一些亟需解决的问题。2006年初，CCSDS对SCPS部分协议进行了修订。SCSP-TP协议对TCP协议进行扩展，提高了航天通信网络链路中TCP协议的传输性能。

本文将围绕空间网络协议和空间通信的链路特性展开讨论，介绍SCPS-TP标准相关协议，并对SCPS-TP中拥塞控制关键技术进行改进。

## 国内外研究现状

目前， SCPS 发展 空间数据系统咨询委员会 CCSDS 制定了分组遥测协议(TM)、分组遥控协议 (TC)、高级在轨系统协议(AOS)。后来 CCSDS 对这 3 个协议重构，制定了空间分 组协议(SPP)、 TM、 TC、 AOS 空间数据链路协议(SDLP)、 TM/TC 同步与信道编码 (SCC)协议等，用来替代以前的标准。 1999 年， CCSDS 制定了空间通信协议规范(SCPS)， SCPS 针对空间传输环境 特性，修改和扩展了 TCP/IP 协议栈，制定了网络协议 SCPS-NP、安全协议 SCPS-SP、 传输协议 SCPS-TP 和文件协议 SCPS-FP，其对应文档为：

1） CCSDS713.0-B-1：空间通信协议规范-网络协议， SCPS-NP；

2） CCSDS713.5-B-1：空间通信协议规范-安全协议， SCPS-SP；

3） CCSDS714.0-B-1：空间通信协议规范-传输协议， SCPS-TP；

4） CCSDS717.0-B-1：空间通信协议规范-文件协议， SCPS-FP。

以上 4 个文件是由美国国防部(DoD)和 NASA 牵头， CCSDS 研究制定的空 间通信协议规范 SCPS 建议书，定义了一系列与地面互联网功能相对应且可以交互运行的协议，并已于 1999 年 5 月被国际标准化组织 ISO 采纳为国际标准。

SCPS 协议以 Internet 协议为基础，针对空间通信网络通信的特点进行了适 当的修改和扩充，主要解决了地面 Internet 协议在空间通信中面临的一系列问题， 如传输时延大、信号电平弱、信道噪声大、多普勒频移大、空-地通信频繁中断等 问题。 SCPS-NP、 SCPS-SP、 SCPS-TP 和 SCPS-FP 在底层协议（数据链路层、物 理层）的支持下，构成了完整的网络模型，实现了空—地、星间的端到端连接， 支持高效、可靠的空间数据传输，一方面适应了空间通信这一特殊环境；另一方 面，又可与因特网协议兼容和互操作， 其中 SCPS-TP 在使用 SCPS 协议时是必须 使用的，而其它三个协议可以用 TCP/IP 协议栈中的协议代替。

在国际上，CCSDS获得各个主要机构和组织的普遍认同，同时采用 CCSDS标准的航天任务也在增加，其范围涵盖各类卫星航天器。欧洲航天局（ESA）明确规定今后航天器全部采用 CCSDS 标准，美国宇航局（NASA）为支持所有采用深空网 DSN（Deep Space Network）的任务而开发的多任务高级操作系统 ANMOS必须采用 CCSDS 标准，这也就意味着以后所有 DSN 新任务都必须使用 CCSDS标准从 20 世纪最后几年开始，CCSDS 的重要贡献是将地面的因特网引入到空间通信中。美国启动了星际互联网（IPN）研究，计划将地面互联网引入到空间，这是美国国防高级研究计划局（DARPA）牵头，喷气动力实验室（JPL）主要负责研发，星际互联网体系主要由主干网络、接入网络、近距网络、星座及编队网络构成，包括地球、轨道和深空三个区域，整套星际互联网体系结构使用的协议就是 CCSDS建议。从 2002 年开始，NASA 也开始涉及此项研究，提出“OMNI”计划，其目的是验证因特网通信技术在空间通信里的可行性。迄今为止，已经在 CCSDS 注册过的空间任务已经达到 300 项，其中包括卫星、航天飞机、国际轨道空间站及运载火箭等，并且有超过 100 种的兼容 CCSDS 建议的支持空间任务的软硬件产品，比如：航天器平台等硬件设施和地面支持辅助系统的软件和硬件等等[1]。CCSDS 标准的最初是以民用为目的，但其带来的效益和成果引起各国军方高度重视，发展迅速，典型例子有美国国家导弹防御体系中的天基红外系统使用的航天器就采用 CCSDS 标准[4,5]目前国外 CCSDS 技术发展主要在两个方面：第一，高级在轨系统（AOS），整个系统采用开放式互连架构，能够统一星上数据流，并趋向标准化、硬件模块化、软件集成化，在国际空间站中得到应用；第二，由美国国防部（DOD）以及美国宇航局共同研究开发的 SCPS（Space Communication Protocol Specification）协议。

SCPS 协议在上世纪末提出，其研究目的是在兼容地面互联网的基础上支持可靠的高速率数据传输，支持并行任务多节点之间网络路由功能选择，并与地面具有良好互操作性等。SCPS 协议解决了空间数据通信中存在的诸多问题，如：信号强度弱、信道噪声大、传输时延大、带宽时延积大、多普勒频移较大以及链路容易中断等问题。经过 7 年的发展，SCPS 协议在空间通信网络中得到了广泛的应用，同时也暴露出了一些亟待解决的问题。2006 年，CCSDS 对 SCPS 部分协议进行了修订。

2006 年重新审定的 SCPS-TP 协议增加了明确的拥塞提示功能，解决了空间通信中 TCP 协议将信道误码导致的丢包误认为网络拥塞导致的问题。此外，在原先标准性能选项之外，增添了 SCPS-TP 扩展性能选项，通过为不同厂商提供不同的独特标识符，以方便厂商开发适应各自应用环境的、具有各自不同特点的产品。在数据系统方面，国际空间站采纳 CCSDS 提出的 AOS 标准，并根据不同航天任务的需要采用四种分类业务，其中包括：复用业务、封装业务、路径业务、位流业务：遥测数据、高速率有效载荷数据、视频数据等大业务量数据通过路径业务传输；部分高速率有效载荷数据及语音数据通过位流业务传输，在设计初期，为实现端到端的通信以及应用业务，还采用因特网业务，该业务用于电子邮件传输、远程数据库下载及一些控制交互指令的传送。

SCPS的应用情况:

对于 CCSDS-SCPS，自1999 年以来，标准化的 SCPS 作为一种开源协议被美国 NASA 喷气实验室(JPL)应用于每一次航天任务。在商用领域，不同厂商也 推出了多种基于SCPS协议产品。美国MITRE公司作为 NASA 非盈利的长期合作伙伴，基于 CCSDS-SCPS 协 议，在 1997 年就成功开发了 CCSDS-SCPS/TCP 网关，实现了CCSDS-SCPS 协议 到 TCP 之间的协议转换，客服了 TCP 协议不适合空间通信信道的不足，并以此实 现了空间 Internet 原型。 2000 年，美国环球科技 公司推出了基于 SCPS 的商用软件 SkipWare，并得到 军用空间网络市场的青睐，其主要的硬件平台为 TurboIP 平台，该平台在 Internet over Satellite 技术方面取 得了重大突破，是一种基于 SCPS-TP 的增强型网关，可以 无缝的融入到现有 Internet，并能与 TCP/IP 实现互通，并具智能拥塞控制、数据速 率控制、选择性否定确认(SNACK)、快速启动等功能。 2004 年，加拿大 Xiphos 技术公司推出了采用 SCPS-TP 的 XipLink 型优化网关， 大大提高了网页响应和下载速率，增加了并发连接数目。 2005 年，该方案先后为沙特、突尼斯等运营商构建上下行分别为 2 Mbps 和 45 Mbps 的一体化卫星网络。之后， Xiphos 还推出更高速率和小型化的 产品。2004 年， LTI 公司也推出了 Mini Accelerator 3000E、 3000S 等基于 SCPS-TP 系列产品，并成功应用在 2005 年美国第十一届 Combined Endeavor 军事演习。 2005 年， Avtec 系统公司也推出了一种 SCPS 通信协议网关，其最大特点在于 其速率可高达 155 Mbps。

就国内而言，作为国内仅有的两个 CCSDS 观察员，中国空间技术研究院和中 国科学院空间科学与应用研究中心多次以观察员身份参加 CCSDS 每年组织的年 会，积极参与 CCSDS 组织的相关工作。中科院还在承担的―实践五号‖卫星的研 制任务中在国内首先采用了 CCSDS 协议。

神州飞船、探测一、二号、实践系列等卫星的成功在轨体现了采用 CCSDS 标准的优越性。2008年4月在西昌发射成功的我国第一颗数据中继卫星“天链一号01星”在数据链路层使用了CCSDS 协议，“天链一号01星”的成功发射提高了神七的测控和通信覆盖能力。更值得关注的是，2008 年6月23日，中国国家航天局 CNSA 已成为 CCSDS 组织第 11 个正式成员，这也意味着中国航天管理机构已承认 CCSDS 空间数据协议体系为空间数据协议标准，中国航天研究机构将普遍采用 CCSDS 协议实现天-地间的数据传输与处理。

但由于多种原因，CCSDS 标准在我国测控与通信领域并被全面采用，而其中对 SCPS协议的研究就更加稀少。

## 本文研究内容

本文首先对SCPS-TP对于TCP传统拥塞控制改进进行了介绍，然后对国内外针对空间网络所进行的拥塞算法改进进行了综述。最后，以Vegas算法为指导基础，提出了自己的改进方式。

本文各章节安排如下：

第一章阐述了课题背景、研究意义以及国内外研究现状。

第二章对SCPS-TP 对传统TCP协议的改进进行了分析，描述了空间网络中长时延和高误码其传输性能产生的影响，介绍了空间网络通信中所使用的传输控制协议。

第三章先对Vegas拥塞控制算法的原理进行了介绍，然后总结了研究人员对其做出的一些改进工作，并指出了不足之处。针对慢启动阶段包突发现象，使得Vegas算法过早结束慢启动阶段，本节算法采用了每个RTT增加一半cwnd的策略。针对拥塞避免阶段，窗口增长速度不能尽快充分利用带宽，采用自适应的窗口增长策略。然后对于NS2仿真工具进行了介绍，最后使用NS2仿真工具对算法进行了验证。

第四章对慢启动阶段Vegas协议在慢启动阶段不能有效利用带宽，参照TCP Westwood算法，采用带宽估计算法有效利用带宽。针对Vegas不能分辨链路拥塞还是路哟改变，采用了路径监视的策略。最后，使用NS2对改进的算法进行了验证。

第五章使用基于IP over CCSDS的软件系统对其进行运行校验。首先对IP/IPv6 over CCSDS和AOS调度算法的实现进行了详细描述。接着从系统总体设计和主要工作流程两个方面介绍了软件的设计实现。然后介绍了本文的网络系统。对其中网关程序的主要功能，软件模块以及工作线程分别进行了深入的分析，最后采用支持跨平台的FileZilla来检验对比最终效果。

第六章对本文做出总结，并对下一步工作进行展望。

# SCPS-TP协议拥塞控制算法分析

空间网络通信双方传输距离遥远，链路上受到星体干扰，电磁等各种外界因素的影响，在实际的数据传输过程中，传输性能将受到严重的挑战。拥塞控制算法作为TCP传输协议的一部分，对数据传输效果有着举足轻重的影响。为此，如何对现有的TCP协议拥塞控制算法进行优化，使其能够在空间 网络的链路环境下有较好的传输效果，是目前的一个重要研究方向。

## TCP协议拥塞控制概述

传统的TCP拥塞控制策略包含4个基本阶段[10]：慢启动阶段、拥塞避免阶段、快速重传阶段以及快速恢复阶段。为了更好的描述这四个阶段，本文首先对其中涉及的几个主要参数进行介绍，具体如表2-1所示。

表2-1 拥塞控制机制主要参数

|  |  |
| --- | --- |
| 参数 | 含义 |
| 发送窗口（win） | 发送端实际发送的窗口值大小，win =  min{cwnd，rwnd}。 |
| 接收窗口（rwnd） | 接收端能够接收的窗口值大小。 |
| 拥塞窗口（cwnd） | 发送端能够发送数据量的最大值。可以用来衡量当前网络的拥塞程度。 |
| 初始窗口（IW） | TCP连接建立后，发送端进行数据传输时拥塞窗口的初始值。 |
| 最大报文段长度（Max Segment Size，MSS） | TCP数据包传输的最大数据分段长度，一般默认取值为1460Bytes。 |
| 慢启动门限（ssthresh） | TCP拥塞控制由慢启动阶段进入到拥塞避免阶段的阀值。 |
| 往返时延（Round Trip Time，RTT） | 发送端从发送报文出去到接收到返回的ACK确认包所经历的时间 |
| 超时重传（Retransmit Time Out，RTO） | 数据包从发送到失效所花费的时间。用来识别数据包的丢失，以及网络是否产生拥塞。 |

TCP在完成建立连接的三次握手后，发送端主要对拥塞窗口值进行维护，它是由网络状况而决定的。

### 慢启动阶段(Slow start）

通常情况下，TCP协议不能准确获取数据传输开始阶段的网络状况。所以，为了合理地使用网络带宽，在传输开始阶段，采用的是一种探测机制，逐步增大拥塞窗口值，使网络尽快地达到饱和状态。在TCP拥塞控制算法中，慢启动算法主要作用于数据传输的开始阶段以及数据丢失的重传阶段。

慢启动算法中，发送端的初始拥塞窗口值cwnd被初始化为1个数据包。于是发送端首先发送一个数据包，接收端成功收到后返回该数据包的ACK，发送端收到ACK确认帧后，增大拥塞窗口为2。接着发送端发送2个数据包，当其收到接收端反馈的2个ACK确认帧时，增大拥塞窗口值为4。依此类推，每收到一个ACK确认帧，cwnd = cwnd + 1，这样每过一个RTT，cwnd都将增加一倍，拥塞窗口以指数形式增长。当拥塞窗口值达到慢启动门限或者发生了丢包时，TCP将结束慢启动阶段，从而进入到拥塞避免阶段。

### 拥塞避免阶段(Congestion avoidance）

慢启动算法可以使TCP的拥塞窗口值迅速地到达一个理想的状态。但是如果一直如此增长，网络将很容易产生拥塞，十分不利于带宽的充分利用。因此，进入拥塞避免阶段后，为了防止网络数据流量的过快增长致使拥塞。拥塞窗口以线性形式增长。接收端每收到一个ACK时，拥塞窗口cwnd = cwnd + 1/cwnd。这样，就能以较为平稳的增长速度接近网络带宽的上限。

### 快速重传阶段(Fast retransmit）

不论是慢启动阶段还是拥塞避免阶段，拥塞窗口值cwnd一直是增长的状态，这必然会导致网络负荷不断加重，最终导致网络拥塞，引起丢包。通常情况下，发送端只有在定时器超时后才会对丢失的数据包重新发送，并不能及时地对丢失的数据包进行处理。为了处理这种情况的发生，TCP采用可快速重传算法。

当通信双方的网络发生了丢包时，会被默认的当作拥塞来处理。如果接收端检测出了超时或者是收到了三个重复的 ACK 确认帧，表明数据发送端和接收端之间的链路上已经产生了拥塞，有数据包已经丢失。此刻 TCP 协议立即进入到快速重传阶段。首先，TCP 将慢启动门限值设置为当前拥塞窗口的一半，降低通信网络上的数据流量，用来减轻网络的拥塞程度，同时不必等待定时器超时而是立即重传丢失的数据包，提高数据的传输效率。

### 快速恢复阶段(Fast recovery）

快速恢复算法就是：TCP拥塞控制策略在对已经丢失的数据包快速重传后，马上进入到拥塞避免阶段，而不是慢启动阶段。接着，快速恢复算法将控制数据的发送，直到收到一个非重复的ACK为止。采用快速恢复算法能让发送端的拥塞窗口值cwnd处于较为合理的状态，使网络的吞吐量保持在一个较高的水平，有利于带宽的充分利用。

传统拥塞控制机制如图2-1所示：



图2-1传统拥塞控制机制

## SCPS-TP对于TCP协议改进

SCPS-TP为空间通信网络提供端到端的数据传输。它根据可靠性要求，将传输服务分为：完全可靠、最大可靠、最小可靠三种方式。SCPS-TP支持基于优先级的处理，支持无连接多播和面向数据包的应用，能与TCP、UDP协议产品互操作。SCPS-TP对应Internet传输协议中的TCP和用户数据报协议UDP，但对TCP协议作了一些扩展：减少TCP启动必需的握手信号，加快数据传输启动；扩大TCP窗口尺寸，适应空间通信往返时延大的要求；具有往返时延测量功能；具有序列号回传保护功能；具有选择性被动确认功能(SNACK)；具有包头压缩功能，以节省开销；具有数据速率控制功能，减少出现拥塞的概率。2006年CCSDS对SCPS-TP做了修改，增加了拥塞提示功能，用来解决空间通信中TCP协议将信道误码导致的丢包误认为网络拥塞的问题。并添加扩展性能选项，厂商可以指定自己不同的独特标识符，方便开发适应各自应用环境的协议产品。SCPS-TP协议在后面会有详细介绍。

为了克服空间链路对TCP性能的影响，SCPS-TP协议针对空间通信中误码率高、传输时延长、链路带宽不对称、间断性连接等问题，改进了标准TCP协议中的部分机制。下面详细介绍SCPS-TP改进方法。

SCPS-TP是在TCP、UDP基础上，为了适应空间通信环境中误码率高、时延大、噪声干扰以及信道不对称等问题而提出的。它提供完全可靠、尽力可靠和最小可靠三种服务(其中完全可靠服务由TCP提供，尽力可靠由做了部分更改的TCP提供，最小可靠由UDP提供)；在时延、带宽和误码多种不利情况下有效工作；可在空间网络环境下有效工作；支持优先级；支持无连接多播；支持面向分组的应用(改进措施见图2-2。

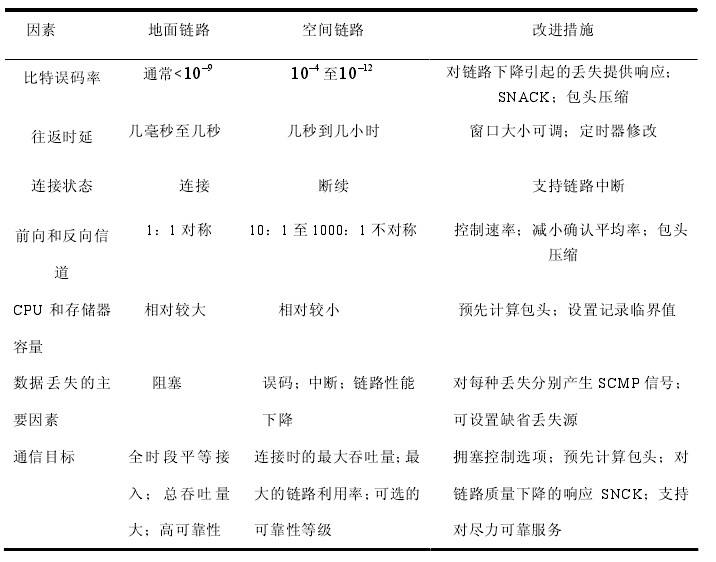


图2-2SCPS-TP的改进措施

### 数据丢失的改进

在空间通信中，导致数据丢失的三个原因是网络拥塞、误码、链路中断。SCPS-TP提高TCP性能的关键在于识别数据丢失原因并进行相应的处理操作。SCPS-TP对数据丢失原因的识别方法和相应处理操作如下。

优化TCP吞吐量的一个办法是标记丢失包并且只做适度的处理。在空间通信环境中最少有三种情况可能导致数据丢失：网络拥塞、链路恶化、链路中断。SCPS-TP对这三种情况采取各不相同处理方式。SCPS-TP有两种机制用来确定数据分组丢失的原因：一是在协议中设置一个参数，它的值由网络管理信息数据库或基于路径的应用程序来设定；二是协议层间信令的传递。在第一种机制中，和标准的TCP一样，在数据分组丢失而没有任何的明确信息时，SCPS-TP设置了一个默认的假设。TCP默认数据丢失的原因是网络拥塞，而SCPS-TP默认的是由网络控制者设置的一个参数，这个参数也可以由以不同线路为基础的应用程序来设定。在一些特殊的网络中，带宽是足够大的，这时候SCPS-TP把数据丢失的原因默认为链路错误。第二种机制是对分组丢失数据的做明确的信号标记，表明数据丢失。

### 网络拥塞的识别和处理

SCPS-TP利用RFC3168定义的主动队列管理（Active Queue Management，AQM）和显式拥塞通知机制（explicit congestion notification，ECN）来识别网络是否处于拥塞状态。AQM检查队列是否将要溢出，以此判断网络是否即将发生拥塞。在拥塞即将出现时，ECN在尚未丢失数据报文的情况下通知SCPS-TP发送端网络即将出现拥塞。ECN路由器在数据报文的IP首部中设置特殊标志并把数据报文向前发送，在收到有特殊标志的数据报文后，接收端SCPS-TP。收到该拥塞信号的SCPS-TP使用Vegas拥塞控制算法来降低丢包率并设置较大的通信窗口，快速恢复网络。

SCPS-TP采用TCP的Vegas拥塞控制机制来使拥塞的损失降到最小。TCP为了使吞吐量尽量接近最佳值，使发送窗口的尺寸接近链路的带宽和时延的乘积BDP(Bandwidth Delay Product)。TCP Vegas拥塞控制机制不是通过调整接收窗口的大小来约束拥塞窗口的大小。TCP Vegas拥塞控制机制通过自己调节来控制边界，即使在窗口很大的时候也不过度使用链路去找饱和点来避免拥塞发生。在空间链路中，因为长时延和能源有限使得从丢失状态中恢复过来是很耗资源的。所以特别需要找到一种更优化的解决办法来替代在TCP中处理数据丢失的办法。TCP Vegas解决的办法是：时拥塞窗口的回复远远慢于标准TCP；在每次吞吐量增加后对下次的吞吐量进行估计，适当调整窗口以避免数据丢失。

SCPS-TP采用TCP Vegas中对慢启动机制的修改。不像TCP那样在每次单向传输后就将拥塞窗口加倍，而是在一个RTT(往返时间)后才将拥塞窗口加倍。SCPS-TP设置一个拥塞窗口的临界点，当到达这个点时，拥塞窗口就由指数增长阶段进入线性增长阶段，SCPS-TP判断在这一点后就进入拥塞避免，并且吞吐量的增加也逐渐减小。另外，当拥塞窗口的大小达到带宽和时延的乘积的时候，SCPS-TP也进入拥塞避免阶段。带宽和时延的乘积是网络连接中的一个应用参数，就像TCP中接收窗口大小一样。TCP Vegas拥塞避免机制不停的测算吞吐量，判断拥塞窗口是否超过临界点进而使网络的联接、数据传输更顺畅。

如果算法对数据丢失的缺省判断为拥塞，或者接收端接收到ECN(Explicit Congestion Notification)网络都认为是发生了拥塞，就会调用拥塞控制机制来应对包(packet)丢失。任何转发路由或者目的端主机都可以发送ECN，同时可以带有ICMP信息或者在包头中带有ECN(中介绍)。

### 链路恶化处理

在空间链路中误码是导致数据丢失的主要原因，而如果链路规划得当，拥塞虽不能避免但可以减少。在这种情况下，SCPS-TP不把拥塞作为数据丢失时的缺省原因，不把调用拥塞控制机制去处理数据包丢失作为缺省操作。为了避免对链路资源的过度占用，SCPS-TP协议使用开环令牌速率控制机制来完成在特定速率情况下测量输出业务。对于每一个终端、端节点，每个链路的速率是个可配置的参数。这个速率参数存储在可全局访问的路由结构中。

对于链路恶化出现误码的情况，SCPS-TP采用以下四个阶段的不同操作来应对，它们分别是：起始确认、信令、反馈信息和终止确认。下行链路对起始确认的误码处理通过空间链路的接收端即地面站来执行。接收端可以请求链路层根据所接收数据包的数量来丢弃那些存在差错(如CRC差错)的信息。接收端中存有一个接收差错数据包的带权值的平均数，一旦这个平均数超过了阈值，网络就会被认为转到了链路恶化状态。接收端网络层维护一个稳定的基于最短最近算法的目的缓冲区，由恶化链路中传输的数据包会存在这个缓冲区内。当接收端网络转到恶化链路状态后，接收端开始把存在历史差错信息的ICMP消息附送到缓冲区所包含的目的地去。每个目的地都会使用确认信息中的TCP选项来通知其相应的SCPS-TP数据源有关链路差错的信息。在ICMP信息更新消息以前，目的端的SCPS-TP协议会对链路的恶化状态保持两个RTT时间左右。

当确认信息段中出现历史差错选项的时候，SCPS-TP的发送部分不会立即启用拥塞控制机制来应对丢包。也就是说，当历史差错选项出现的时候，发送端不会立即减小拥塞窗口的大小，也不会重置重传计时器。这个对丢包的处理机制，可以用常规方式从多重确认信息以及超时数值中检测丢失包的信息，而该机制将维持工作直到接收到一个不包含历史差错信息选项的确认信息。

### 链路中断

在处理链路中断过程中，有类似处理误码的四个阶段：识别、通知、响应、终止识别。识别链路中断的方法，是和具体链路有关的。通常，地面站检测到载波同步信号丢失或者检测到接收信号强度低于某一门限值时，认为链路中断。一旦地面站（或航天器）检测到链路中断，它会向链路同一侧的所有主机发送“链路中断”SCMP报文，它包括丢失包的SCPS-TP首部。发送端SCPS-TP对链路中断信号的响应是：进入持续模式，停止发送新数据，停止重传定时器，周期发送嗅探报文检查链路是否恢复。SCPS-TP在收到链路恢复SCMP报文或者嗅探报文顺利到达接收端并被确认时，回到正常工作模式。发送端SCPS-TP会根据先前收到的链路中断SCMP报文中的SCPS-TP首部中的序号，来决定数据流从何处恢复传输。

链路中断是指通信连接的瞬间中断，造成中断的原因可能是卫星临时脱离地面站的视野，就是说当网络拓扑节发生变化时设备的切换或其他短时间终端的现象。与处理链路恶化一样，SCPS-TP对链路中断的处理也分为四个阶段：起始确认、信令、反馈信息和终止确认。确定一个链路是否有中断出现靠的是判断机制而与链路本身无关。在通常情况下，链路中断在地面站处得到确认，即通过接收信号强度低于某一个阈值来判断。一旦地面站或航天器发现了链路中断，它会在接收该信息的链路上发送一个联络中断ICMP消息到任意终端。ICMP消息的产生有抵达业务来触发，并含有生成该消息的TCP包头部分。SCPS-TP发送端对链路中断信号的响应是进入持续模式，持续发送周期性探测数据包。SCPS-TP并不重复性地设置超时、进行重传以及重置重传计时器。相反的，除非遇到特殊的探测数据包，SCPS-TP会挂起计时器并中止重传机制。当SCPS-TP从地面站处接收到链路重建ICMP消息或者某个探测数据包的到达获得确认时，SCPS-TP就会退出持续模式。当链路重建时，SCPS-TP发送端就能从TCP头部的序列号来推断该数据流从什么位置来继续传送数据包，而该TCP头部是在原始的链路中断ICMP信息中所携带的。

### 误码的识别和处理

空间通信中导致数据丢失的主要原因是误码。SCPS-TP将数据丢失的默认处理方式设置为针对误码的处理方式。

SCPS-TP针对误码的识别和处理过程分为四个阶段：识别、通知、响应、终止识别。空间链路的接收终点即地面接收器执行下行链路的识别操作。地面接收器从链路层获取接收到的受损帧的个数（如通过CRC校验）。地面接收器维护一个最近接收分组目的地址的缓存器，并根据受损帧总数计算受损帧加权流动平均值。当该平均值超过门限值时，地面接收器进入链路出错状态，接收器的网络层调用空间网络控制报文协议SCMP（类似于TCP/IP中的ICMP协议）向使用该空间链路的终端主机发送标示为“链路已出错”的SCMP报文。终端主机接收到该SCMP报文后，立即通过ACK报文（报文首部包含链路出错选项）向其对应的SCPS-TP数据源接收到该ACK后，不启动拥塞控制，也不减小拥塞窗口，发送速率和重传间隔时间保持不变，直到发送端接收到未带有链路出错信息的ACK时，进入终止识别阶段。

### 带宽容量有限的改进

针对空间链路带宽容量有限而使网络性能受限的情况，SCPS-TP提供两种改善机制：头部压缩技术和选择性否定确认(SNACK)选项。

SCPS-TP的头部压缩技术在传输层实现端到端操作。SCPS-TP头部压缩将头部减少了大约50%，极大节约了确认报文的开销。SCPS-TP头部压缩技术将连接期间不变化的信息进行摘要处理。经过压缩的SCPS-TP头部是可变长的，但必须包含连接标示符、用以指明哪些可选字段被使用和哪些标志位被置位的比较域，还要包含校验和。SCPS-TP仅压缩SCPS-TP头部而不压缩IP头部。

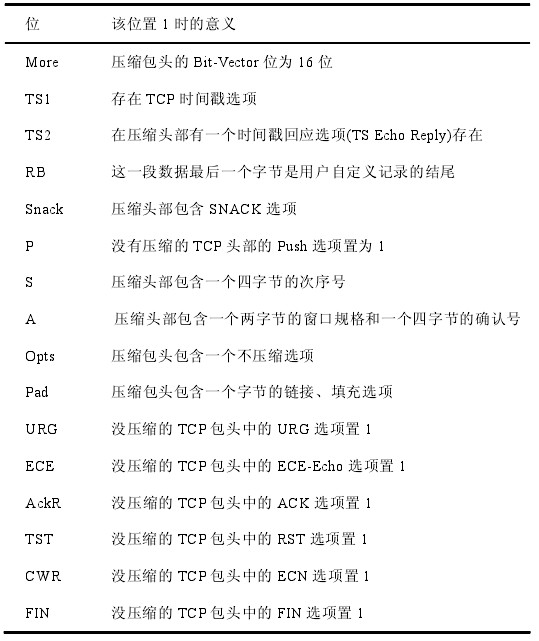


图2-3 压缩包头选项说明

SCPS-TP应对带宽受限的一个重要措施就是端到端TCP头部压缩。SCPS-TP的头部压缩策略和TCP/IP的头部压缩是基本一致的。TCP/IP的包头压缩方法是通过在链路层上基于下一跳信息来执行的，链路层上保留了发送和接收TCP/IP包头的组成部分。压缩器通过分配连接ID和首次通信连接所保存的TCP/IP发送包头来进行初始化。后续的包头仅由前一个TCP包头变化而来。包头压缩所采用的方法是综合会话过程中不变的元素、以连接标识符代替端口号，并去掉与传输分组无关的信息。头部压缩和解压缩由端系统执行，数据链路层并不执行这些操作。

SCPS-TP 协议具有可容忍丢失TCP头部压缩功能，可在传输层完成端到端得数据传输工作，能够容忍丢包以及连接变化。SCPS-TP包头压缩后大概可以降低TCP包头大小的50%左右，大大节省了反馈信息段的开销。图2-4SCPS-TP包头压缩后的SCPS-TP头部。

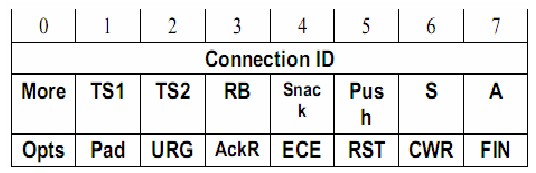


图2-4SCPS-TP头部

传统的TCP使用ACK报文进行确认，一个ACK报文最多能够识别接收缓冲区中的一个缺失数据块。SNACK允许SCPS-TP接收方通知发送方在接收方队列有一个或一个以上的缺失数据块。这样，SNACK选项可以加速丢失数据的重传并避免了发送方达到窗口极限。

SCPS-TP对TCP的一个重要的扩展就是SNACK(Selective Negative Acknowledgment)。SNACK选项集合了SACK和NAK机制的优点，为适应空间通信环境而提出的。SACK选项提供接收端已经接收到并被排序的数据中非连续数据块的信息。SACK选项并没有改变原TCP包头中的ACK序列号的独立性。SACK的目的就是让数据发送端利用SACK选项所提供的附加信息来更有效的重传丢失的数据报。SACK选项包含一个成对的长度为16比特的可变长度链表。链表中每一对代表接收端的一个数据块。在空间通信环境下，SACK选项会产生两个难题：SACK选项是基于高速网络环境的，因此它没有高效利用比特位；由于TCP包头限制在只能用于小于40个八位组，因此SACK选项在时间戳选项被使用时，它只能标注最多三个洞。NAK选项是为卫星环境设计的，它比SACK更能有效利用带宽。但NAK不是积极确认而是一种消极确认。NAK也未改变TCP包头的内容，并不能影响传统TCP的ACK数据段的部分。

Option Type为强制添置位。添加选项域的第一个字节。选项类别包含一个公认的SNACK选项的类别数。目前暂定为十进制数21。

Option Length为强制添置位。它位于选项域的第二个字节位。这个字段的数字表示SNACK选项的整个八位组的长度。当SNACK的Bit-Vector位未使用时，Option Length应该固定为6。当SNACK的Bit-Vector位被使用时，Option Length应该固定为8。

Holel Offset为强制添置位，位于选项域的第三个和第四个八位组。它包含了出现第一个洞与ACK序列号的相对位置。

Holel Size为强制添置位，位于选项域的第五个和第六个八位组。它为双方互相协调的最大报文段(MSS)位所标注的连续洞的个数。

SNACK Bit-Vector是一个可选项，只有当接收端缓存区出现的洞为非连续时，才会应用，它以“0”“1”来区分是否有非连续的洞。举例说明：

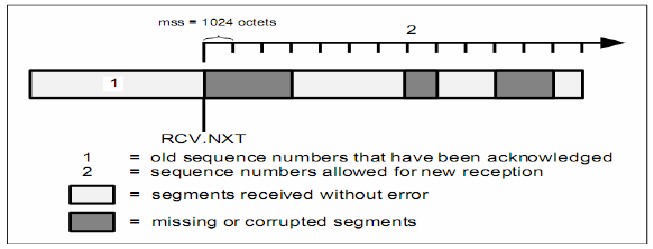


图2-5接收情况

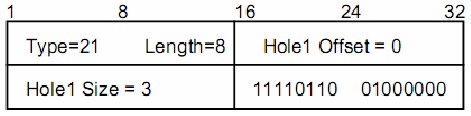


图2-6ACK选型添加情况

从图2-6中可以看出，缓冲区在RCV.NXT后面有连续三个洞，后面还有两个不连续的洞。因为有非连续的洞出现，SNACK Bit-Vector位被添加，因此Length为8。Kind是SNACK的识别公共号，应为15。RCV.NXT下一位就有洞，Holel Offset置为0，HolelSize为3。再如下面的例子(如图2-7)：

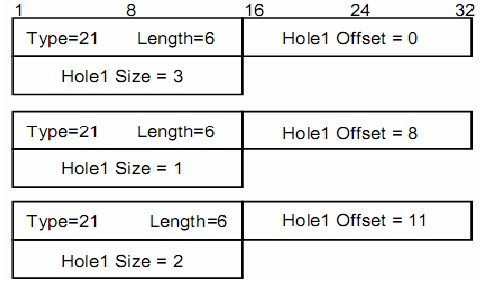


图2-7ACK选型添加示例

SNACK不仅可以在受限方面发挥作用，在应对数据丢失方面也是非常有用的。SNACK允许接收端通知 SCPS-TP协议发送关于接收端乱序队列中的多个缺失或错误数据段的信息。而在不具备选择性确认信息的情况下，TCP协议只可以利用ACK序号来识别接收端中至多一个缺失数据段。利用简单的累积确认以及快速重传算法，TCP协议能够有效地恢复每个窗口的单一丢包。但由于接收端必须通过接收新数据来增加ACK序号，因此TCP协议需要最小一个RTT的时间来发送信号来告知该乱序队列中每个额外的缺失数据段。SNACK选项可以对接收端缓存空间中的多个缺失数据段进行标识。通过快速提供更多的有关丢失数据段的信息，SNACK选项能够加快数据恢复的速度以及避免发送端窗口受限，因此允许在等待获知丢失数据段信息时耗尽管道容量。在出现丢包时持续传输数据的能力是极其重要的，尤其当丢包是由误码而非拥塞引起时。在这种情况下，当确知是使用拥塞控制处理丢包时，SNACK在保持通信管道容量饱和使用方面具有特殊的效力，并允许在丢包恢复过程中数据传输能够持续工作在满负荷的瓶颈状态下。

### 针对长往返延迟的改进

SCPS-TP中，通过两个技术措施解决长往返延迟(RTT)对数据传输所带来的影响：窗口缩放和定时器修改。

窗口缩放功能允许SCPS-TP处理多于64K字节的未收到应答的数据。在数据丢失主要是由误码造成的情况下，采用更大的窗口尺寸有助于提高网络性能，这样就可以在恢复丢失期间不断地传送新的数据。

SCPS-TP修改了TCP定时器的超时时间，允许往返延迟从秒级增加到小时级，符合深层空间传输往返延时实际情况。在建立连接时，SCPS-TP将重传定时器设置成一个合适的往返延时估计值，从而避免在连接的开始阶段就出现多次重传超时。另外，SCPS-TP协议利用TCP时间戳选项，有助于比较精确地估计在有分组丢失情况下的RTT值。

在目前的航天测控网络中，网络的拥塞一般还不是很严重，因此我们可以作出一个合理的假设：即网络的RTT变化相对较为缓和，即使有一定程度的拥塞，一段时间内RTT的增加也是连续的；与此相对，如果RTT的变化较为突然和不连续，则必定是网络拓扑结构发生变化所致。

在SCPS-TP的发送端可以观测到RTT的变化，但这种变化的连续或突然是十分模糊的概念，不好界定；因此，仅凭具体的变化数值不易区分RTT增大的原因是由于路径变化导致的传播时延增加，还是网络发生拥塞使时延增加。因此，当航天测控网络的拓扑结构发生改变时，SCPS-TP中应用Vegas拥塞避免机制的性能会明显下降 。

### 链路不对称的改进

TCP依赖一串确认信息来记录从发送端发送的数据。在航天器与地面站之间的通信链路是很特别的，它在一个方向是很高的配置而在反方向上却是相对很低的配置。这种配置在遥测和遥感中是很常见和很典型的。如果卫星遥测信道所传送的数据并不是都很可靠，那么TCP确认信息所产生的业务量就不可以较为轻松的在反向信道上传输。为了解决这个问题，SCPS-TP协议取消了对每间隔一个接收数据段就产生一个TCP确认信息的限定。当存在乱序队列时，SCPS-TP协议还放松了对每个接收数据段进行确认的要求。相反，SCPS-TP接收端对确认信息进行时延处理，延迟时间是RTT估计时间相关联的可配置的时间周期。

严重的时间延迟肯定将会从多个方面影响TCP的性能。基于特定的确认频率、重传计时器的值以及所使用的窗口的大小，快速重传算法可能会失效。当链路中同时有多个连接时，ACK的传输线路就可能出现拥塞，这时TCP的拥塞机制就会降低数据传输率和ACK的发送频率。但在这种情况下即使拥塞控制机制降低数据传输速率，SCPS-TP也会保持ACK的发送频率。SCPS-TP应对不对称链路的另一个办法是包头压缩。关于包头压缩在本章的后面将会提到。

信道的宽度不对称性导致反向链路上TCP的ACK报文拥塞，进而使TCP通信的链路利用率降低。SCPS-TP采用两种改善方法：降低确认应答频率、速率控制。

TCP协议规定每接收一个数据报文时向发送端至少回应一个确认报文，并且规定当出现接收数据报文乱序时对每个数据报文立即给予确认，SCPS-TP取消了以上两项规定，而是改成了SCPS-TP接收方以固定速率给予确认，减轻了对确认信道的带宽压力。

SCPS-TP采用开环速率控制机制以避免过度占用链路带宽。开环速率控制机制允许在每个端点上基于每个SCPS-TP连接，配置传输速率参数。该机制在整个传输过程中把分组的发送速率控制在用户指定值并限制突发传输数据的尺寸。

### SCPS-TP的扩展能力

SCPS容量选项设置在类似TCP头部中SYN字节的位置，包括：选项类型、选项

长度、容量选项字节向量。

选项类型：必须设置并且应该占用选项的第一个字节；设置值是十进制的20

选项长度：必须设置并且应该占用选项的第二个字节；设置值是十进制的4

容量选项：必须设置并且应该占用选项的第三个字节；指示SCPS容量链接有效

连接ID：必须设置并且应该占用选项的第四个字节；如果Com字节被设置为1，连接表示符应设置为非0

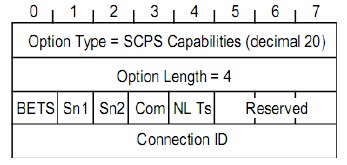


图2-8 SCPS-TP容量选项

扩展容量时会被重新使用的SCPS-TP容量选项在SYN(TCP包会有SYN字段被设置)字段上标识两次或两次以上。这些扩展选项的长度不允许为4，以便与‘标准’SCPS-TP容量选项区分开。就像以前说过的，在SYN字段上的第一个SCPS容量选项的长度必须是4允许扩展容量的目的是：允许对这种运行形式感兴趣的不同厂商或社团能够制定各自不同的环境。每个对此感兴趣的厂商或社团都会得到一个不同的标识扩展容量信号的第一部分是出现在第二个SCPS-TP容量选项。这个选项具有标准TCP选项格式。这个选项的类型是SCPS-TP容量(20)，这个长度是可用的，但不能为4(长度的要求是基于前面提到的SCPS容量选项)。

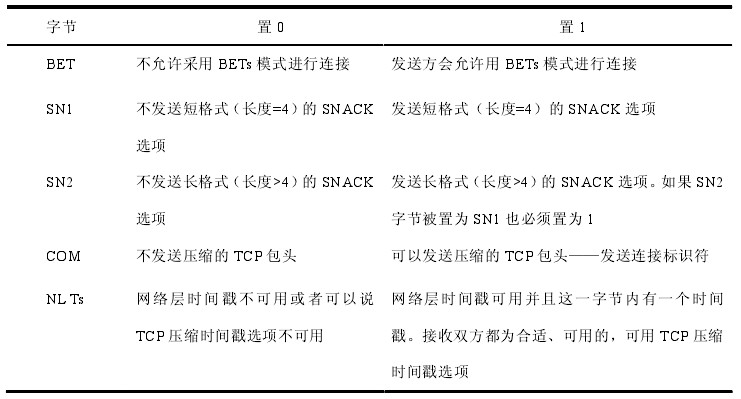


图2-9 SCPS-TP容量选项说明

上表说明：

(1)如果TCP两个连接端点发送BETs字节值为1，则此次连接采用BETs模式执行。

(2)如果SCPS-TP连接双方都使用压缩包头,这时网络层的时间戳(NLTs)是有效的，越界时间戳会被设置在网络层头部的时间戳域内。

选项长度：这个域内的值的扩展是将SCPS-TP扩展容量选项中等于编号字节(octet)1和字节(octet)2的内容进行扩展。扩展SCPS-TP容量选项的长度域类似于常规TCP选项的长度域。长度域的值可以不等于4，以确定执行能够理解扩展选项不至于把扩展容量的SCPS-TP选项和‘标准’的SCPS-TP容量选项弄混淆。



图2-10 扩展容量信号开端

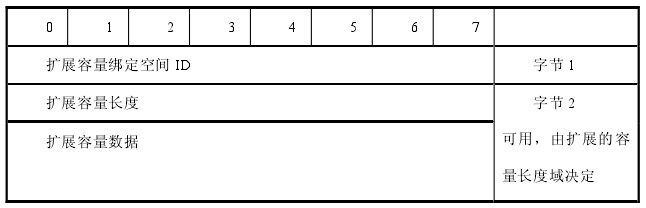


图2-11 扩展容量格式

## 目前基于SCPS-TP的改进策略

随着CCSDS发布了SCPS协议后，空间链路研究人员的目光从改进TCP协议转到了改进SCPS协议。其中主要是针对SCPS-TP改进。虽然SCPS是针对太空的特殊环境(相对于地面通信环境)进行改进的，但是太空中的不同应用场景，还是会存在差异。SCPS协议针对的是太空应用场景中存在的公共特性来改进而产生的。但是太空中不同的应用场景还有存在自己独有的特性，针对这些独有的特性，协议还具有改进的空间。

**（1）SC**PS-TP 协议运行时，可以收集到的重要网络状况指标值是 RTT，根据 RTT 的不同变化状况，本章将网络状况分为三种：网络正常、网络拥塞和网络拓扑结构变化。

用BP神经网络预测：

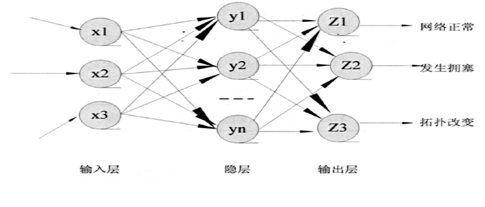


图 2-12 算法策略

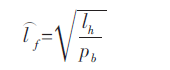
实现本算法的伪代码为：

|  |
| --- |
| rtt =(rtt1+rtt2)/2  将rtt作为 BP 神经网络的一个输入量  将 rtt1 作为 BP 神经网络的另一个输入量  通过 BP 神经网络预测网络状况  If(网络状况为正常)  base\_rtt=rtt  else if (网络状况为拥塞)  {  正常进入 Vegas 算法减小拥塞窗口  }  else if (网络状况为拓扑结构变化)  {  Vegas 算法 增大拥塞窗口  } |

**（2）对SCPS参数进行优化**

对链路的空间通信帧长的优化

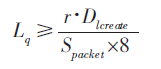
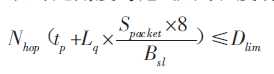
在SRARQ 协议下，最佳帧长为



其中lh代表的是数据帧头部的开销，Pb代表的是误码率。

队列长度的优化

空间网络中，中轨卫星、低轨卫星等空间飞行器与地面站结点之间的链路，因空间飞行器轨道运行而发生切换，产生分组丢失。空间网络应用期望尽可能低的分组丢失率。

（1）（2）

队列长度应该满足上面2个式子，其中，Lq为平均队列长度，r 为上层应用流量速率，Dlcreate为链路建立的时延，Spacket为分组大小，单位Byte。，tp为分组传播时延，Bsl为链路带宽，Nhop为链路经过的跳数，Dlim为应用对端到端时延的约束要求。

**（3）基于路径的 SCPS-TP 流量控制的改进**

针对的情况：在地面站—通信终端卫星之间的通信时，当由直接通信转变为地面站—中继卫星—通信终端卫星时，即地面站和通信终端卫星由直接连接链路变为间接连接链路时，路径发生变化，这时RTT的时间就会边长。如果根据vegas算法，就会默认为是发生了网络拥塞，就会减少拥塞窗口，与实际情况不符。根据这个情况，对vegas算法改进。

关键在于判断出何时链路发生了变化，然后用新的RTT替换vegas中的Base\_RTT。

1):判断链路发生了变化：在SCPS-NP协议中SCPS-NP 包头中的 Hop Count 字段，是记录包在网络中经过跳数的计数器，Hop Count 字段的值在中继通信时要比直接通信增加 1。发送端可以通过检查所收到的 ACK 包中 Hop Count 字段值的改变来判断路径中卫星个数是否变化，进而决定是否需要更新 BaseRTT

2）更新BaseRtt

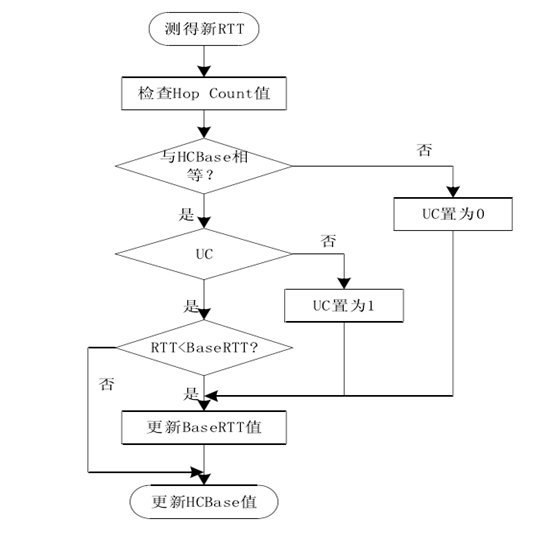


图2-14 算法流图

Vegas 改进机制更新时延基准 BaseRTT 的操作流程如图 3-19。SCPS-TP 连接建立后,在发送端记下当前的 Hop Count 值，作为比较的基准 HCBase。RTT 是发送端通过对某时刻窗口内所有包的往返时间统计得到的，每隔一段时间得到一次。当每次得到新的 RTT 时，检查此时 ACK 包中的 Hop Count 值是否等于 HCBase。如果不等，表明路径发生了变化，需要更新 BaseRTT。也可以进一步判断跳数是否增加，只在跳数增加时才进行下一步。记此时的 RTT 为 RTT0，RTT0的观测期间为 T0，由于路径变化发生在 T0期间，所得到的 RTT0不能完全反映当前的路径时延，置标志位 UC(Update is Complete)为 0。还需再过一个完全在路径变化后的观测期间 T1，将观测得到的 RTT1作为新的 BaseRTT。最后将 HCBase 更新为当前的 Hop Count 值。改进后的 Vegas 机制在路径发生切换时，可以及时准确地更新 BaseRTT 值，并根据后面持续不断的新 RTT 值来更新拥塞窗口大小，不会出现盲目减小的状况。

**（4）SCPS-TP中慢启动改进**

Vegas 拥塞控制策略的cwnd 增加方法在大时延空间链路上显得保守缓慢，其探测带宽的灵敏性使得它较早地对网络状况作出响应， 从而过早结束慢启动进入拥塞避免阶段，极大降低平均吞吐量。

综合考虑Vegas 和Hybla 及相关改进型TCP 的拥塞控制策略优点的基础上，提出新的用于卫星通信链路的拥塞控制策略， 称之为Veghy 策略。Veghy 策略在慢启动阶段，cwnd 借鉴Hybla 协议按比例因子ρ 的方式增加，比例因子ρ 为实际RTT 与基准值RTT0的比值，ρ = RTT /RTT0。；而在拥塞避免阶段同Vegas 方法，根据测得的RTT 值并利用α 和β判断链路带宽利用情况， 从而控制cwnd 的增或减。

实现的伪代码：

|  |
| --- |
| If （cwnd < ssthresh） /\* 慢启动阶段\*/  cwndi+1 = cwndi+ρ2 -1 ;  Else /\* cwnd >= ssthresh ，拥塞避免阶段\*/  If （Δ\*RTTmin<α） /\* 带宽未充分利用\*/  cwndi+1 = cwndi+1 ;  Else If （Δ\*RTTmin >β） /\* 带宽已超负荷\*/  cwndi+1 = cwndi-1 ;  Else /\*α≤ Δ\*RTTmin ≤β，带宽得到充分利用\*/  cwndi+1 = cwndi;  When（DupAck || TimeOut） /\* 当（多）包丢失或超时\*/  cwndi+1 = cwndi \* 3/4; /\* 拥塞窗口减少1/4 \*/ |

其中Δ是实际吞吐量与预期吞入量的差值，实际吞吐量：cwnd/RTT，期望吞吐量：cwnd/RTTBase 。

## 本章小结

本章首先对TCP协议的拥塞控制进行了介绍，传统的拥塞控制机制主要包括了慢启动、拥塞避免、快速重传和快速恢复这个4个阶段。然后分析了SCPS-TP协议对于传统TCP协议进行的改进；最后，介绍了目前对于SCPS-TP改进的方法。其中TCP Vegas协议修改是本文的重点研究内容，第三章和第四章将对其进行更详细的介绍。

# 对Vegas拥塞控制算法的改进

## Vegas拥塞控制算法概述

Brakmo和Peterson于1995年提出一种用RTT测量网络状况的拥塞控制算法，称之为TCP Vegas。Vegas算法工作原理是通过计算预期吞吐量和实际吞吐量之间的差值来评测当前的网络可用带宽。与TCP Reno协议相比，Vegas在下面3个方面有了改进：

1. 新的慢启动机制

与TCP Reno相比，TCP Vegas不仅希望能有效地利用可用频宽，而且也尽量避免因为传送太快而发生数据包丢失的情况。故在慢启动（slow-start）阶段，为了检测及避免数据包丢失，TCP Vegas减慢了cwnd增加的速度。Vegas 修改了TCP 的slow-start算法：cwnd的值大约经过2个RTT后才增加1倍。与TCP Reno不同的是，Vegas根据预期传送率与实际传送率之间的差异值（Diff）来调整ssthresh（慢开始门限）的值。当TCP Vegas检测到网络开始有队列产生时，Vegas就由慢开始（Slow-start）进入拥塞避免（Congestion Avoidance）阶段。

1. 新的重传机制

在TCP Vegas 中，Vegas改进了重传的机制，每当收到一个Duplicate ACK时，Vegas会特别留意RTT的值以检查是否有超时发生。若是，则不需要等待3个Duplicate ACK，Vegas就会直接重送此数据包。此外，当丢失的数据包被重送并返回新的ACK时，Vegas也会特别注意第一个或第二个返回的ACK，以检测随后的数据包是否已超时，并重送丢失的数据包。

1. 新的拥塞避免机制

TCP Vegas拥塞算法描述见式（3-1）：

 （3-1）

其中， RTTbase表示的是发送端与接收端之间RTT的最小值。通常情况下是TCP连接建立后发送第一个数据包经历的RTT，而cwnd（congestion window） 表示当前拥塞窗口值的大小，即发送的数据包数；RTTcur 表示当前计算出的RTT。

如果期望吞吐量和实际吞吐量之间差值很小，说明拥塞并没有发生，那么Vegas算法就会增大拥塞窗口值以充分利用带宽，如果期望吞吐量与实际吞吐量的差值很大，说明网络很有可能已经发生了拥塞。Vegas算法就会减小拥塞窗口以降低网络的拥塞程度。根据上述的思想，Vegas 算法就能保证数据传输的正常运行。

调整拥塞窗口的算法可描述见式（3-2）：

 （3-2）

δ=Diff×RTT，代表的是路由器中缓存的数据包个数，α和ρ是定义的两个阈值，通常的取值为1和3或者2和4，α触发发送速率的提升；ρ触发发送速率的降低。

## 对Vegas协议的改进

在慢启动阶段由于窗口快速增长，导致TCP包突发现象产生。这种现象导致TCP Vegas在慢启动阶段过早的进入拥塞避免阶段，这样将会使cwnd过早的从指数增长变为线性增长，不能够充分利用网络带宽。此外，网络资源和用户的竞争可能会随着时间的推移而不可预知，而传统的TCP Vegas不能够反应这种变化。为了更快速地反应网络变化，窗口调整算法应该变得比原有的算法更加积极。我们的算法将解决这些问题。

根据对TCP Vegas的分析，本文将从两个方面来提高TCP Vegas协议的性能。1：提高拥塞窗口的增长速率；2：调整α和β，以提升TCP Vegas的传输性能。TCP Vegas协议有三个阶段：慢启动（slow start），拥塞避免（CA），快速重传阶段。本文将讨论慢启动阶段和拥塞避免阶段的改进。

### 慢启动阶段改进

Vegas 计算一个额外的数据δ，并且每隔一个RTT来倍增当前的cwnd，当δ大于γ的时候，Vegas算法结束慢启动阶段，进入拥塞避免阶段。问题在于，当cwnd应该倍增的时候，Vegas发送端在接收到一个ACK的时候，连续发送两个包，这将会导致在短时间内发送双倍数量的包，这种激增将会导致排队队列暂时变长，最终产生δ偏移，使得δ过早的高于γ。作为结果，Vegas过早的结束指数增长的慢启动阶段，使得性能受到影响。

如何使得包平滑的传输将是解决δ计算偏移的关键。本节的算法将对Vegas算法做出两个简单的改变：首先，拥塞窗口将在每个都进行调整而不是每隔一个RTT。其次，每个RTT，窗口值增加1.5倍，而不是原先的两倍。这样的算法修改将首先解决包突发的问题，平缓δ计算偏移的现象。其次，新算法将会更快的使得算法达到最大可利用带宽。表3-1中表明新算法cwnd增长更快。

表3-1每个RTT中Vegas和NVegas的cwnd对比

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| RTT | 0 | 1 | 2 | 3 | 4 | 5 | 6 |
| Vegas | 2 | 2 | 4 | 4 | 8 | 8 | 16 |
| NVegas | 2 | 3 | 4.5 | 6.75 | 10.13 | 15.19 | 22.78 |

下面是慢启动阶段的伪代码：

|  |
| --- |
| if(cwnd\_ < ssthresh\_) { // slow-start  if(delta > v\_gamma\_) {//进入慢启动阶段  // slow-down a bit to ensure  // the net is not so congested  ssthresh\_ = 2;  cwnd\_-=(cwnd\_/8);  if(cwnd\_<2)  cwnd\_ = 2.;  v\_incr\_ = 0;  } else  v\_incr\_ = cwnd/2;  } |

### 拥塞避免阶段改进

链路的吞吐量可以由下式（3-3）推出，可以发现吞吐量受到cwnd和RTT的影响。本文拥塞避免阶段改进策略将基于测量的RTT值，和计算的Diff值，根据Diff处于不同的值域决定计算策略。cwnd的增长速率会得到修订，α和β的值也会根据Diff所处值域的不同而动态变化。

（3-3）

原始的Vegas拥塞避免阶段的策略可由式（3-2）得到。对于每一次TCP Vegas的连接，其期望的吞吐量在每一次连接初始化就决定了。当一个ACK数据包接收成功，第一个实际吞吐量由RTT和前一个数据包大小所推得。Diff值根据（3-1）和（3-2）推得（3-4）

 （3-4）

其中，BaseRTT是之前测量到的最小RTT，RTT(k)是在时间k时测得的RTT值。在连接建立的初试节点，如果测得的RTT接近BaseRTT，根据式子（3-2），满足条件一，cwnd与吞吐量都将会快速增长。RTT(k)会因为链路拥塞而不断改变。链路拥塞越经常发生，RTT(k)就越增大。其中有两个重要的操作参数α和β影响cwnd的变化。在图（3-1）中可以看出对于TCP Vegas拥塞避免阶段有三个条件：当diff<α成立，吞吐量处于一个很低的水平，这时候吞吐量将会增长。当diff处于α和β之间，吞吐量将会保持在一个稳定水平。当diff大于β时，吞吐量将会降低。

图3-1

这样三个条件的划分在面对长延时网络特别是卫星链路的时候不是很灵活。最主要的原因是，当面对大的RTT值是，cwnd的增长值不能得到及时更新。因此，我们提出了一个新的条件分割，由四个部分组成，由图（3-2）所示：

图3-2

之所以划分成四个区域的主要原因是为了更加明了的却分当前网络的拥塞状况。我们分别这样定义四个区域的划分：

Area1：这一区域表示网络带宽高度可用，因此cwnd将会迅速增长。

Area2：网络带宽即将耗尽，因此cwnd的增长将会缓慢下来。

Area3：这时候表明网络连接稳定，cwnd将会保持稳定。

Area4：网络将会发生拥塞，cwnd将会减小以避免网络拥塞以及丢包。

这样四个区域的划分将会更加有利于网络的稳定。下面将给出四个区域分别对应的cwnd、α和β变化策略伪代码：

|  |
| --- |
| if(cwnd>=ssthresh)//congestion avoidance stage  {  if(diff<alpha)//Area1  {  P=P+1;  cwnd+=(beta-diff)\*P;  alpha+=4;  beta+=10;  }  else if(alpha<diff<(alpha+beta)/2)//Area2  {  p=0;  cwnd+=(alpha+beta)/2-diff;  if(beta>alpha+10)  beta-=10;  }  else if((alpha+beta)/2<diff<beta)//Area3  {  P=0;  if(alpha>4)  alpha-=4;  if(beta>alpha+2)  beta-=2;  }  else(beta<diff)//Area4  {  P=0;  if(diff-beta>cwnd)  cwnd-=(diff-beta);  else  cwnd=2;  }  } |

与传统TCP Vegas算法相比，上述算法有如下优势：1、cwnd在不同的链路条件下动态调整；2、参数α和β动态调整；3、cwnd的增长速率动态调整，为此我们引入了一个参数P，在不同的值域下P取不同的值，保证增长速率的变化。

在Area1时，cwnd因为P的不断增长而急剧的变大。当Area2时，cwnd降低增长速率，同时减小α和β之间的差值。当Area3时，cwnd达到稳定，α和β之间的差值也将达到最小。当Area4时，cwnd急剧的减小。

在这一部分，我们将分析在有更多可用带宽的情况下，改进的Vegas策略与传统的Vegas算法在拥塞避免阶段分别需要多少时间来达到稳定阶段。这里，我们考虑一种简单的情况，当前平衡状态下的窗口大小是W，新平衡情况下的窗口大小是nW。

根据公式3-2可以知道，Vegas算法在每个RTT都会线性的更新cwnd。很明显可以知道，这将会用（n-1）\*W个RTT才会充分的利用新的可用带宽。而另一方面，当新的Vegas算法探测到新的可用带宽的时候，拥塞窗口的增长方式将会按照公式。如果新策略用i个RTT来达到新的平衡状态的话，那么我们将会得到下公式：

我们可以做如下假设，δ在这个短暂的时期内趋近于零。上式可以计算如下：



可以看出来，这个值将远小于（n-1）\*W。

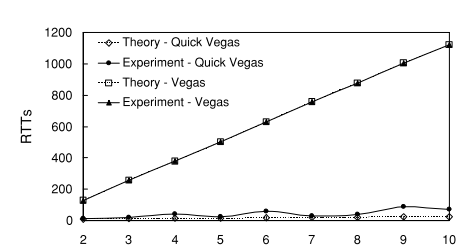


图3-3算法到达平衡的时间

在拥塞避免阶段，包突发现象因为链路瓶颈的存在而趋于平滑。Vegas算法也在这个阶段线性的增长，这将会导致包突发现象大大减少。图3-4表示的是无论是从实验上来看还是从理论上来看，新的Vegas策略都将会更快到达平衡阶段。

## 模拟方案设计

对于改进的Vegas算法性能分析，可以通过网络模拟工具进行实现。比较常见的工具有NS2、OPNET，其中NS2是一款开源软件，并且NS2有卫星网络模拟的扩展，这对本文的实验有很大帮助。

除了NS2等网络模拟工具，比较常见的做法是将算法方案写入linux kernel module，NS2中关于Vegas相关代码是参考linux内核，修改起来也相对比较方便。最后，我们还可以在已有的空间通信协议规范传输协议 SCPS-TP开源代码中对Vegas进行修改。

本节将先实现第一个目标：即在NS2中对Vegas改进算法进行实现。本文使用的NS版本是2.3.5。

### NS2仿真工具介绍

Network Simulator 2是一种以C++及OTcl写的非连续-事件引发(discrete-event driven)及物件导向(object oriented)的网络模拟器，乃由 VINT project 所开发。NS-2提供了一个整合性的环境，实现在单一模拟器中提供多种不同层次的细部明细，其目的在支援网络研究及教育，如协定设计、流量研究等。传统做模拟的通常需事先写程式码来实做一个演算法，亦可能要很多机器，甚至需要有比较多设备的实验室、开发测试平台等等.；而利用NS-2，我们可以比以前更容易模拟一个演算法，首先建立自己的情境模拟及需要的可能网络状况，再设定相关参数、通讯协定─经由NS2执行而得出一个输出档， 再透过一些软件如nam、gnuplot、和awk等进一步的分析，比起传统的做法容易，且能省下不少时间。

NS2使用C++和Otcl作为开发语言。NS2可以说是Otcl脚本解释器，它包含了仿真事件调度器，网络组件对象库和网络建设示库。事件调度的仿真时间，并且激活事件队列中的当前事件，执行一些相关的事件，网络组件通过数据包的传输与彼此沟通，但它不消耗仿真时间。都需要花时间来处理网络仿真组件分组必须使用事件调度。首先，它发送一个事件的数据包，然后等待此事件将回来，做下一步处理。事件调度器的另一个用途是定时。出于效率的考虑，数据信道和控制信道的实现是分开的。为了减少分组和事件的处理时间，基本网络组件对象事件调度器和数据通道是用C++编写，这些对象通过映射OTcl解释器可见。

### 具体实验步骤

NS2在其完整的仿真过程中，主要包含以下几个步骤：首先是修改源代码，然后编写Tcl脚本，最后就是分析仿真出来的结果，接下来对这三个部分做更加详尽的描述：



图3-4网络模拟的过程

（1）修改源码：在实际的仿真过程中，使用者根据需求决定是否需要修改源代码。NS2源代码一般是由C++语言来编写，这就要求使用者具备C++编程基础以及相应的调试知识。值得特别关注的是，因为NS2有C++和OTcl两种编程语言而组成，所以在修改C++代码时，要对相应的OTcl代码也进行修改。

（2）Tcl/OTcl仿真代码编写：这一步是进行NS2仿真实验时不可缺少的，绝大多数NS2的仿真实验实际上就是通过编写Tcl脚本来控制仿真实验中模拟事件的开始和结束过程。所以，使用者需要了解Tcl脚本语言的相关知识。

（3）仿真结果分析：这一步可以体现出仿真实验的实际效果，如果与预期效果不符，还需要重新修改后再进行仿真，因此，这一步非常有必要。想要正确分析出仿真结果，使用者需要了解NS2中Trace文件的结构，同时可以使用常用的辅助工具对结果文件进行分析，最后对得到的分析数据制成图表等。

## 仿真结果与分析

### NS2拓扑结构及代码修改

NS2拓扑结构如图3-5示。其中路由器R1和Send端在一个空间局域网。路由器R2与Recive端在另一个空间局域网。两个局域网之间采用的是传统有线网络的链路环境，即低时延，低误码。而路由器R1与路由器R2之间则是长时延，高误码的空间链路环境。



图3-5网络拓扑结构

在上述的拓扑结构中，路由器R1到路由器R2之间的链路仿真参数如表3-1所示。在无特殊说明的情况下，链路的默认仿真参数如表3-1所示，其中链路的带宽为10Mb，时延为300ms，误码率(单位：byte)为10-5，数据传输方向为Send端发往Recive端。

NS2中关于Vegas部分代码，Vegas.cc修改:

|  |
| --- |
| Void NewVegasTcpAgent::recv(Packet\* pkt,Handler)  {  if(rtt>0) {//只列出和slow start和congestion void两个阶段相关的代码  /\* if there's only one pkt in transit, update  \* baseRTT  \*/  if(rtt<v\_baseRTT\_ || rttLen<=1)  v\_baseRTT\_ = rtt;    double expect; // in pkt/sec  // actual = (# in transit)/(current rtt)  v\_actual\_ = double(rttLen)/rtt;  // expect = (current window size)/baseRTT  expect = double(t\_seqno\_-last\_ack\_)/v\_baseRTT\_;    // calc actual and expect thruput diff, delta  int delta=int((expect-v\_actual\_)\*v\_baseRTT\_+0.5);  if(cwnd\_ < ssthresh\_) { // slow-start  if(delta > v\_gamma\_) {  // slow-down a bit to ensure  // the net is not so congested  ssthresh\_ = 2;  cwnd\_-=(cwnd\_/8);  if(cwnd\_<2)  cwnd\_ = 2.;  v\_incr\_ = 0;  } else  v\_incr\_ = cwnd\_/2;  }  } else { // congestion avoidance  if(delta>v\_beta\_) {    P=0;  if(delta-v\_beta\_>cwnd\_)  v\_incr\_=v\_beta\_-delta;  else  cwnd\_=2;    } else if(delta<v\_beta\_&&delta>(v\_beta\_+v\_alpha\_)/2)  {  P=0;  if(v\_alpha\_>4)  v\_alpha\_-=4;  if(v\_beta\_>v\_alpha\_+2)  v\_beta\_-=2;    }  else if(delta>v\_alpha\_&&delta<(v\_beta\_+v\_alpha\_)/2)  {  p=0;  v\_incr\_+=(v\_alpha\_+v\_beta\_)/2-delta;  if(v\_beta\_>v\_alpha\_+10)  v\_beta-=10;  }  else // current rate is cool.  {  P=P+1;  v\_incr\_+=(beta-diff)\*P;  v\_alpha\_+=4;  v\_beta\_+=10;  }  }  } // end of if(rtt > 0)  } |

关于实时吞吐量，本节写了一个awk脚本分析tr文件，awk文件如下：

|  |
| --- |
| #!/usr/bin/awk  BEGIN {  bytes\_recvd = 0;  throughput = 0;  interval = 1;  current\_time\_instance = 0;  nxt\_time\_instance = current\_time\_instance + interval;  }  {  action = $1;  time = $2;  from = $3;  to = $4;  type = $5;  pkt\_size = $6;  flow\_id = $8;  src = $9;  dst = $10;  sequence\_n0 = $11;  pkt\_id = $12;  if (time < nxt\_time\_instance)  {  if (action == "r")  {  bytes\_recvd = bytes\_recvd + pkt\_size;  }  }  else {  current\_time\_instance = nxt\_time\_instance;  nxt\_time\_instance += interval;  throughput = bytes\_recvd / current\_time\_instance;  printf("%d %dn",current\_time\_instance, throughput/1024);  }  }  END {  } |

将其保存为insth.wak，用法为awk –f insth.wak file1.tr > out1.dat，out1.dat就是我们想要的实时吞吐量数据文件。

### NS2仿真结果与分析

#### 3.3.3.1 初始窗口改进仿真

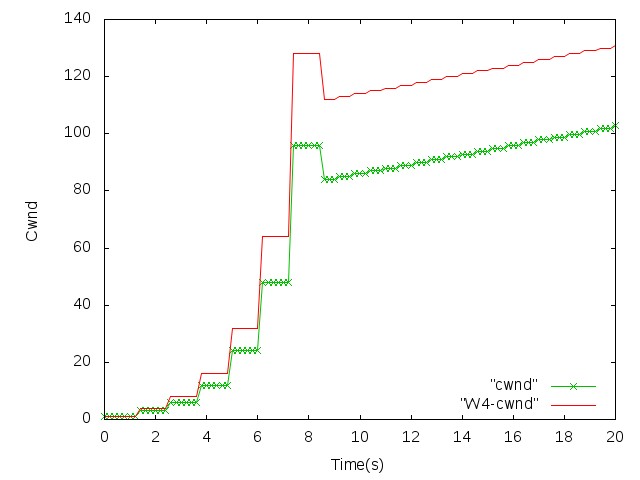


图3-6拥塞窗口对比

图3-6示的是拥塞窗口值增长情况对比。由于初始窗口的改进是针对长时延而进行的，为了更好的显示出改进后的效果，本文是在延时为300ms，无误码的理想状态下仿真的。红色曲线（上面）代表的是改进后的Vegas算法窗口值变化，绿色曲线（下面）代表的是传统Vegas算法的窗口值变化，改进后的Vegas算法初始窗口值为4。从图中可以看出，采用增大初始窗口策略后，相同时间段，拥塞窗口值要更大。这是因为增大初始窗口后，相同的时间内，发送端能接收到更多TCP ACK，从而使得拥塞窗口值更大。

图3-6显示的是算法吞吐量情况对比。数据传输的吞吐量是衡量协议性能的主要技术指标之一，同样，它是在延时300ms，无误码的链路环境下仿真的，红色曲线（上面）代表的是改进后的Vegas算法吞吐量变化，绿色曲线（下面）代表的是传统Vegas算法的吞吐量变化。从图中可以看出，在同一时刻，改进后的拥塞控制算法吞吐量要高于传统Vegas算法的吞吐量。在20秒时，改进算法的吞吐量可达到505kbps，而传统的Vegas算法只能达到445kbps。这是因为单位时间内，发送端的拥塞窗口值更大，能发送更多的数据包。从而使得网络的吞吐量得到了提高。

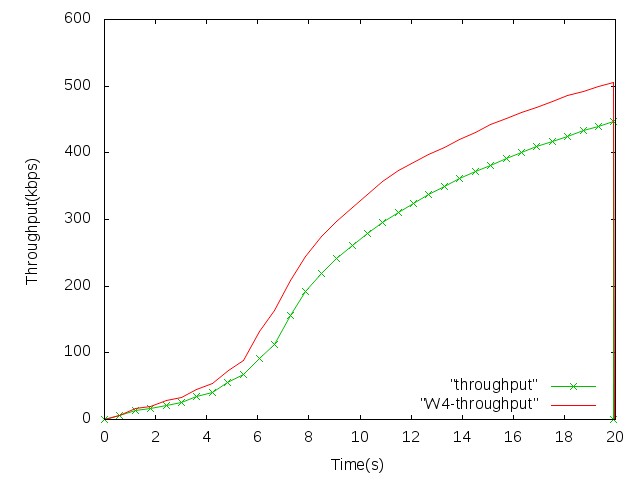


图3-7吐量对比

通过上述的仿真对比以及分析，可以得出:在短连接TCP以及小数据传输的情形下，采用扩大初始窗口的策略，能够有效的提高其传输效率，更好的充分利用带宽。

#### 3.3.3.2 加速拥塞窗口增长仿真

图3-7显示的是拥塞窗口值增长情况对比。同样的，加速拥塞窗口增长的改进方案是针对长时延而进行的，为了更好的显示出改进后的效果，本文是在延时为300ms，无误码的理想状态下仿真的。红色曲线（上面）代表的是改进后的Vegas算法窗口值变化，绿色曲线（下面）代表的是传统Vegas算法的窗口值变化，改进后的Vegas算法初始窗口值为4。从图中可以看出，采用加速拥塞窗口增长策略后，在大约24秒时，就能够到达最大拥塞窗口值；而传统的Vegas算法却需要花费112秒的时间，才能达到。这是因为传统的Vegas算法在进入到拥塞避免阶段后，它是线性的增长方式增加窗口值,即每收到一个TCP ACK，增加1 / cwndcur；而采用了加速窗口策略后，在拥塞避免阶段，它先是以加速的增长方式扩大窗口值，即每收到一个TCP ACK 增加cwndopt / 16 / cwndcur。接着当最佳拥塞窗口值cwndopt与cwndcur相差不大时，再以传统Vegas的增长方式进行，即1 / cwndcur。这样就大大地缩短了在拥塞避免阶段所花费的时间。

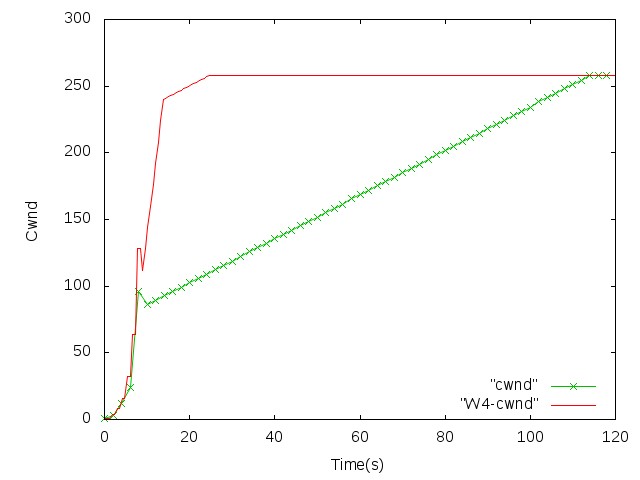


图3-8拥塞窗口对比

图3-8显示的是算法吞吐量情况对比。同样都是在延时300ms，无误码的链路环境下仿真的，红色曲线（上面）代表的是改进后的Vegas算法吞吐量变化，绿色曲线（下面）代表的是传统Vegas算法的吞吐量变化。从图中可以看出，在同一时刻，改进后的拥塞控制算法吞吐量要高于传统Vegas算法的吞吐量。在100秒时，改进算法的吞吐量可达到将近1300kbps，而同一时刻，传统的Vegas算法只能达到1000kbps。这是因为采用了加速拥塞窗口增长策略后，发送端能增加更多的拥塞窗口值，从而发送了更多的数据包。使得网络的吞吐量得到了提升。

通过上述的仿真对比以及分析，可以得出:在短连接TCP以及小数据传输的情形下，采用加速拥塞窗口增长的策略，能够有效的提高其传输效率，更好的充分利用带宽。

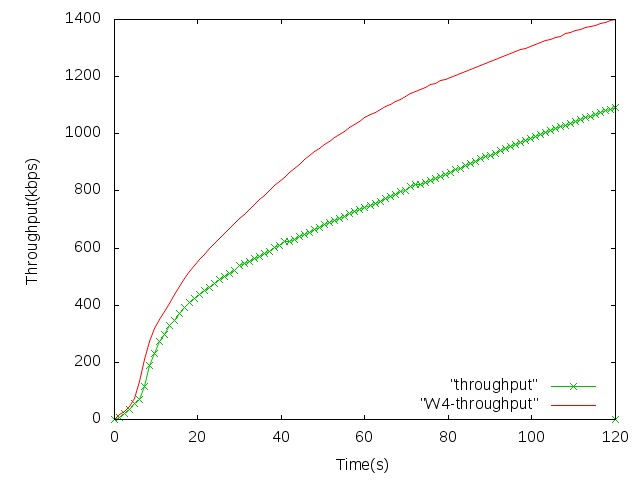


图3-9吞吐量对比

## 本章小结

本章首先对Vegas拥塞控制算法的原理进行了介绍，然后总结了研究人员对其做出的一些改进工作，并指出了不足之处。针对慢启动阶段包突发现象，使得Vegas算法过早结束慢启动阶段，本节算法采用了每个RTT增加一半cwnd的策略。针对拥塞避免阶段，窗口增长速度不能尽快充分利用带宽，本节算法采用了自适应的窗口增长策略。本文采用NS2对所做的工作进行仿真，因此介绍了NS2仿真的工作流程，紧接着对仿真环境进行详细描述。通过仿真可以看出，新算法优于原有Vegas算法。

# 基于动态带宽估计拥塞算法改进

在通信网络中，TCP Vegas 在网络利用率、稳定性、公平性、流量、丢包率表现出比TCP Reno更好的性能。因而TCP Vegas通常用在具有大带宽时延积（BDP，Bandwidth Delay Product）网络中，尤其是空间通信环境。然而，在慢启动阶段突发的数据传送将使TCP Vegas算法提前进入拥塞避免阶段，此时的cwnd（Congestion Window）远未达到可用网络带宽的有效值。同时，TCP Vegas不能有效区分链路改变和拥塞。在本节中提出了一种基于带宽估计和路由监视的BaseRTT更新的新算法，拥塞水平估计得更加精确，在大宽带时延积网络中新算法性能表现更加优秀。

## 卫星网络中TCP-vegas算法中存在的问题

### Vegas慢启动阶段不能有效利用带宽

TCP Vegas慢启动阶段，当RTT值较小（如在几百ms或几十ms），BDP（Bandwidth Delay Product）相对较小，经过多次RTT进入拥塞避免阶段。但是RTT值越大，例如高达500ms以上，BDP是比较大的，拥塞避免阶段阈值也相应增加。假定初始cwnd为1，RTT的网络往返延迟值，可用有效带宽为Bw（单位pkts/s），可以得到，慢启动阶段的持续时间是：

 4-1

由上式可见，慢启动阶段需要的时间Timess与 RTT\*log2RTT成正比。长时延空间信道环境下， RTT 值可以达到秒级，假设 RTT=1s， Bw=2Mbps， 1个 segment 的大小 MSS 为 1Kbyte，那么利用上面公式，可得该条件下慢启动阶段所需时间约为 16s。可见，在长时延信道条件下，Vegas 算法达到网络可利用有效带宽所需的时间较长，这对提升网络的吞吐性能是不利的。

虽然Vegas 算法可以很成功的预测网络的拥塞状况，但是在慢启动阶段突发的数据传输会导致Vegas 算法提前进入拥塞避免阶段，然而此时 cwnd 并未达到网络可利用的有效带宽值。在这种情况下，当 RTT比较小时对网络性能的影响并不显著，但当 RTT 很大时，这种影响是非常大的。时延值越长，BDP 越大，提前进入拥塞避免阶段对网络性能带来的影响就越打。在拥塞避免阶段，cwnd 每个 RTT 增幅为 1。从进入拥塞避免阶段到达 到最佳拥塞控制窗口所需时间为：

 4-2

如果，Bw=8Mbps，RTT=0.5s，MSS=2Kbyte，那么最佳拥塞窗口大小应该是250，然而Vegas算法会在cwnd达到128时提前进入拥塞避免阶段。在拥塞避免阶段会多需要122个RTT才会达到最佳拥塞窗口。这对传输是极为不利的。

本文提出了一种借鉴TCP Westwood的带宽估计算法，来提高SCPS-TP中Vegas算法在慢启动阶段的性能。

### Vegas算法不能区分链路拥塞还是路由改变

Vegas算法因为采取了新的拥塞避免机制，可以比较准确地测量网络的拥塞状况，所以相比于Reno算法，TCP Vegas算法更加适用于卫星网络。但是Vegas算法以RTT为主要参数来控制发送窗口的变化，而卫星通信网的链路拓扑结构是高动态实时变化的，这会造成RTT的非拥塞原因增大。Vegas算法本身并没有能力识别RTT的增大是由网络拥塞造成的还是由路径变化造成的。因此如果是路径的改变导致RTT的增加Vegas算法也会减小发送窗口，这样不仅会浪费宝贵的链路带宽资源，同时也会导致了TCP协议的性能下降。

在卫星网络中，传播时延随着卫星之间距离以及传输路径的变化而变化，通信距离每增加1000km，会带来额外的13.3ms的往返时延。以具有全球覆盖能力的极轨道星座为例，星座参数设置如图（4-1）所示。由于卫星之间高速的相对运动以及两极地区星间链路的断开与重建，其网络拓扑结构变化频率达平均3～4min/次。此外，由于卫星星座覆盖缝隙的存在(即反向运动卫星之间不建立星间链路)，使得网络通信时延存在较大突变。



图4-1极轨道星座参数

在不考虑卫星星上处理时延的情况下，A地至B地之间通信的RTT是实时变化的，其中最大RTT为328.6ms(通信终端位于覆盖缝隙两侧时)，最小RTT为33.6ms，即由于通信路径距离变化引起的往返时延差值高达295ms。

一般情况下，Vegas算法认为当前测得的往返时延与最小往返时延的差值是排队时延。在卫星网络中，往返时延与最小往返时延的差值中，很大一部分是由于传播时延的变化引起的。大的传播时延差值可能会隐藏掉排队时延对往返时延的影响，成为决定往返时延的主要因素，致使Vegas算法不能准确预测网络的拥塞状况。此外，卫星间距离变化范围的不同导致Vegas算法的性能也不同。卫星轨道高度越高，往返时延RTT的变化范围越大，Vegas算法的性能也就越差。

当卫星网络拓扑结构改变使得端到端通信路径长度变化范围增大时，当前往返时延与最小往返时延的传播时延差值增加，而这种增加与网络的拥塞状况无关，相反地，发送终端应该根据带宽时延积的增大，相应地增大拥塞窗口。但是，在Vegas算法中，发送终端会误认为往返时延增大是由于网络状况恶化引起的，进而减小拥塞窗口。可见，在卫星网络中，由于网络拓扑结构动态变化使得往返时延变化较大，导致Vegas算法存在不稳定性。

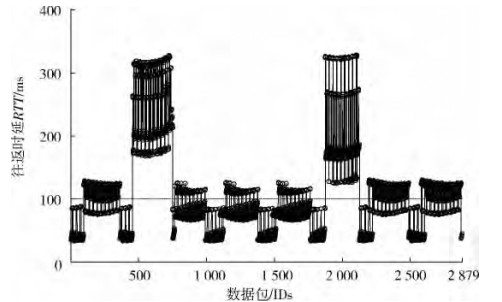


图4-2 A-B往返延迟情况

因此，如何在复杂的卫星链路中准确估计BaseRTT是本节的重要研究内容。

## 基于带宽估计的慢启动算法改进

我们的第一个改进是通过加入一个参数a，从而减小因为数据包激增造成排队时延陡然增大引起往返时延增大对网络的影响。

由下公式可知：diff是关于cwnd和RTT的 方程。

 4-3

而在实际网络应用中，cwnd与RTT的关系可以由下公式表示：

 4-4

 由上两个式子可得下公式，可以得出diff是关于cwnd的方程。

 4-5

由上面式子可以得出结论diff与cwnd和BaseRTT的取值有关，而与当前RTT取值无关。如果路由不发生变化，那么BaseRTT将会是一个常量。因此，只要确定cwnd，就可以得出diff的大小。在大BDP网络中，可以认为cwnd与RTT是线性关系，那么两次RTT之间的差值ΔRTT 可以由下列公式求得：

4-6

每次接收一个RTT，就计算一次当前a的值，如果路由不发生变化，那么a的取值将在一个小范围内波动，故本节算法采用加权平均算法来计算a的取值。当第一次计算出a的样本值a1，以后每计算出一次新的样本值，就通过下式对a的值进行加权更新：

 4-7

其中μ取值范围[0,1], μ越小，那么anew受到的aold的影响越小。在实际链路中，μ的取值与具体网络环境有关，这里μ取值为0.5。

发送端每收到一个RTT就对a进行一次计算，当观测结束后，发送端通过下公式对差值diff进行修正，从而减小由于数据包激增造成排队时延陡然增大引起往返时延增大对网络的影响：

4-8

我们在慢启动阶段第二个改进是通过借鉴TCP Westwood中带宽估计的算法，估计出当前网络环境下慢启动阶段所能达到最合适的cwnd。ssthresh在TCP协议中是对当前可用带宽的一种近似估计，因此 ssthresh最好设置为让链路中的传输数据量等于链路中的可用带宽，即链路被充分占用且不占用中间缓存。

如果源端成功接收一个数据分组的应答，则可知该数据分组已经被目的端成功接收，这就是TCP协议应答的机制。故测量链路带宽(Bandwidth)可通过在源节点来检测数据分组返回的速度。如果发送端在ti时刻收到来自接收端的ACK应答包，这表示已经正确传输了li长度的数据包。然后，我们可以通过下式来估计ti时刻链路带宽Bwi：

 4-9

发送端每次收到一个ACK就会对当前链路带宽进行估计，这样就会得到一个带宽样本值。然而这些样本值具有一定的随机性，不能够恰当反应当前链路的贷款状况。故本章将利用数字信号处理中的指数加权滤波器对样本带宽进行滤波，这样就会得到一个相对平稳的带宽估计值。滤波器的增益将会随着瓶颈链路容量变化而进行变化，其中增益会适应网络负载，通过上面的分析可以得到：

  4-10

其中，

 4-11

1/τ为低通算法的截止频率。

在慢启动阶段，每当发送端收到一个RTT就更新一次a，并计算当前带宽Bwi，如果观测器结束，就根据公式4-8计算一次diff的值，如果diff≤γ+0.5，那么cwnd 增加当前cwnd一半的值。如果diff>γ+0.5，此时，将当前链路的带宽估计值与当前cwnd进行比较，如果Bwi≤cwnd，那么就会结束当前的慢启动阶段，从而进入拥塞避免阶段。反之，就将继续执行当前慢启动算法。

以下是慢启动阶段的伪代码：

|  |
| --- |
| if(cwnd\_ < ssthresh\_) { // slow-start  anew=μ×aold+(1-μ)×ai  Bwi=(1-ηi)×Bwi+ηi×Bwi-1  diff =cwnd×(1-BaseRTT/(RTT-a×cwnd))  if(diff > v\_gamma\_+0.5) {  if(Bwi<cwnd){ v\_incr\_ = cwnd/2;}  else  {  ssthresh\_ = 2;  cwnd\_-=(cwnd\_/8);  if(cwnd\_<2)  cwnd\_ = 2.;  v\_incr\_ = 0;  }  } else  v\_incr\_ = cwnd/2;  } |

## 基于路径监视的BaseRTT更新算法

BaseRTT是Vegas算法中用来计算期望发送数据包数量的重要参数。因此，如果BaseRTT不准确的话，我们将不能准确估计当前的拥塞水平，也不能够充分利用当前带宽。在普通网络环境下，链路状态不经常发生改变，因此BaseRTT不需要自适应的去改变太多，然而在卫星网络环境下，重路由和链路变化经常发生。这种情况下，使用固定的BaseRTT将是不恰当的。在Vegas算法的拥塞避免阶段，BaseRTT在计算diff、cwnd中有重要作用，如果BaseRTT有错误，系统的表现将会大大受到影响。

如果路由的链路扩大，RTT变大，如果根据原本Vegas算法，BaseRTT不变的话，因为diff=cwnd（1-BaseRTT/RTT），所以diff也会变大。所以当跳数扩增非常大的时候，RTT的增长也会很大，所以导致diff的增幅也会很大，一旦diff>β,那么cwnd便会相应的减小。然而由于链路的扩大，链路中总缓存也会增大。造成这一切的原因是BaseRTT只减少确不增加，这在复杂的网络环境下会导致Vegas性能下降。

本节推荐的算法是监视每一个终端到达数据包的RTT。当TCP连接进入拥塞避免阶段，带宽尺寸将不会被估计，但是RTT的变化将会被监视。当网络拥塞发生的时候，RTT将会渐渐发生变化。然而如果链路条件变化，或者发生重路由，RTT将会发生剧烈变化。这种特点可以用来分辨当前是发生了拥塞还是链路发生了变化。

当TCP连接进入拥塞避免阶段，每当收到一个ACK就得到一个RTT。我们将其记做RTTnow，并将其与RTT的平局值RTTave进行比较，其中RTTave取最近40个RTT的平均值。如果RTTnow大于RTTave则将BaseRTT重置。下面是伪代码：

|  |
| --- |
| if(cwnd>=ssthresh)//congestion avoidance stage  {  if(delta>beta)  {  incr=delta/beta-1;  cwnd=cwnd-incr;  if(cwnd<2)  cnwd=2;  }  else if(deta<alpha)  {  incr=1-delta/alpha;  cwnd=cwnd+incr;  }  Else//current rate is cool  {  if(RTTnow>RTTave) BaseRTT=2\*RTO+BaseRTT;//reset BaseRTT  }  } |

上面代码中，其中RTO是超时重传（Retransmit Time Out，RTO）。如果当前RTT大于平均RTT的话，就重置BaseRTT的值。

另外，在代码中我们引入了一个变量incr，而incr的取值由公式4-12决定：

 4-12

TCP Vegas在拥塞避免阶段线性变化，cwnd改变的速率很慢，因此因此我们引入了incr，使得cwnd的改变更具有自适应性。可以根据当前网络的情况，改变cwnd改变的速率。如果δ大于β的话，表明当前带宽拥塞，因此δ越大，incr的值就越大，减小速率就越大。如果δ小于α的话，就表明当前网络带宽没有充分的利用，传输速率应该变快。δ越趋近于0，incr的值就愈大。 这样的话，cwnd就可以更快的达到稳定状态。

## 仿真结果与分析

### 代码编写

NS2代码编写主要包括两部分：

（1）基于带宽估计的慢启动代码

这段代码参考linux内核中linux-3.2.12/ net/ ipv4/ tcp\_westwood.c，westwood算法相关函数段的实现。下面是代码：

|  |
| --- |
| static u32 tcp\_westwood\_bw\_rttmin (const struct sock \*sk)  {  const struct tcp\_sock \*tp = tcp\_sk(sk);  const struct westwood \*w = inet\_csk\_ca(sk);  return max\_t (u32, (w->bw\_est \* w->rtt\_min) / tp->mss\_cache, 2);  }  static inline u32 westwood\_do\_filter (u32 a, u32 b)  {  return ((7 \* a) + b) >> 3;  }  static void westwood\_filter (struct westwood \*w, u32 delta)//带宽过滤器  {  /\* If the filter is empty fill it with the first sample of bandwidth \*/  /\* 如果是第一次得到带宽测量样本\*/  if (w->bw\_ns\_est == 0 && w->bw\_est == 0) {  w->bw\_ns\_est = w->bk / delta;  w->bw\_est = w->bw\_ns\_est;  } else { /\*已经有收到过测量样本了\*/  w->bw\_ns\_est = westwood\_do\_filter(w->bw\_ns\_est, w->bk / delta);  w->bw\_est = westwood\_do\_filter(w->bw\_est, w->bw\_ns\_est);  }  }  static void tcp\_westwood\_pkts\_acked (struct sock \*sk, u32 cnt, s32 rtt)  {  struct westwood \*w = inet\_csk\_ca(sk);  if (rtt > 0)  w->rtt = usecs\_to\_jiffies(rtt);  }  static struct westwood westwood\_update\_window (struct sock \*sk)//这个函数就是更新带宽估计值的，δ在慢启动阶段除了与gamma进行比较，同样也与bw\_est 进行比较  {  struct westwood \*w = inet\_csk\_ca(sk);  s32 delta = tcp\_time\_stamp - w->rtt\_win\_sx;  if (w->first\_ack) { /\* 是第一个ACK \*/  w->snd\_una = tcp\_sk(sk)->snd\_una;  w->first\_ack = 0;  }  if (w-rtt && delta > max\_t (u32, w->rtt, TCP\_WESTWOOD\_RTT\_MIN)) {  westwood\_filter(w, delta); /\* 更新带宽估计值\*/  w->bk = 0; /\*清零确认字节数\*/  w->rtt\_win\_sx = tcp\_time\_stamp; /\* 重设取样周期开始时间\*/  return w;//w->bw\_est就是我们想要的最后结果  }  return NULL;  }  void Vegas\_slow\_start(struct sock \*sk)  {  if(cwnd\_ < ssthresh\_) { // slow-start    v\_inc\_flag\_ = !v\_inc\_flag\_;  if(!v\_inc\_flag\_)  v\_incr\_ = 0;  else {  if(delta > v\_gamma\_) {  // slow-down a bit to ensure  // the net is not so congested    struct westwood\* bw=westwood\_update\_window (sk);  if(delta>bw->bw-est)  {  // slow-down a bit to ensure  // the net is not so congested  ssthresh\_ = 2;  cwnd\_-=(cwnd\_/8);  if(cwnd\_<2)  cwnd\_ = 2.;  v\_incr\_ = 0;  }  else  {  v\_incr\_=(rtt/v\_baseRTT\_)\*(rtt/v\_baseRTT\_);  }  } else  v\_incr\_ = (rtt/v\_baseRTT\_)\*(rtt/v\_baseRTT\_);//tcp Hybla  }  }  } |

（2）基于路径监视的BaseRTT更新算法

|  |
| --- |
| Void Vegas\_Cong(struct sock \*sk)  {  { // congestion avoidance  if(delta>v\_beta\_) {  /\*  \* slow down a bit, retrack  \* back to prev. rtt's cwnd  \* and dont incr in the nxt rtt  \*/  cwnd\_-=delta/v\_beta\_-1;  if(cwnd\_<2) cwnd\_ = 2;  v\_incr\_ = 0;  } else if(delta<v\_alpha\_)  // delta<alpha, faster....  v\_incr\_ = 1-delta/v\_alpha\_;  else // current rate is cool.  v\_incr\_ = 0;  if(RTTnow>RTTave) BaseRTT=2\*RTO+BaseRTT;//reset BaseRTT  }  if(RTTnow>RTTave) v\_baseRTT\_=2\*RTO+v\_baseRTT\_;//reset BaseRTT  }  } |

### 运行结果分析

本章所搭建的实验环境与3.4.1节一样，在此就不再重复描述。

图4-3显示的是拥塞窗口值增长情况对比。基于新策略对Vegas的改进，解决的是空间链路高误码所产生的不利影响。因此本章是在延时为300ms，误码率（单位：byte）为10-5的实验环境下进行仿真的。图中红色曲线（上面）代表的是改进后Vegas算法的窗口值变化，绿色曲线（下面）代表的是传统的Vegas算法窗口值变化。从图中可以看出，传统的Vegas算法窗口值波动幅度比较大，而改进后的算法相比之下较为稳定。这是因为在误码频繁造成链路丢包后，数据包将进行重传，此时发送端所得到的RTT将会增加，误认为链路吞吐量太大而产生了拥塞，不必要的降低了拥塞窗口。采用新策略后，发送端能够更加准确地判断出丢包的原因，丢包重传使得RTT增加，并不会让拥塞窗口值减小。因而其窗口值的变化处于相对稳定状态。

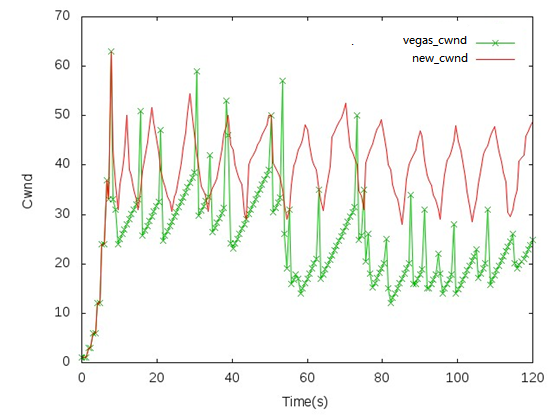


图4-3 拥塞窗口对比

图4-6显示的是算法吞吐量情况对比。同样，它是在延时300ms，误码率（单位：byte）为10-5的链路环境下仿真的，红色曲线（上面）代表的是改进后的Vegas算法吞吐量变化，绿色曲线（下面）代表的是传统Vegas算法的吞吐量变化。从图中可以看出，在同一时刻，改进后的拥塞控制算法吞吐量要高于传统Vegas算法的吞吐量。在100秒时，改进算法的吞吐量可达到220kbps左右，而传统的Vegas算法只能达到150kbps上下。改进后的拥塞窗口波动幅度处于相对稳定的状态，而传统Vegas算法窗口值波动较大。这样采用了新策略的Vegas能发送更多的数据包，从而使得网络的吞吐量得到了提高。

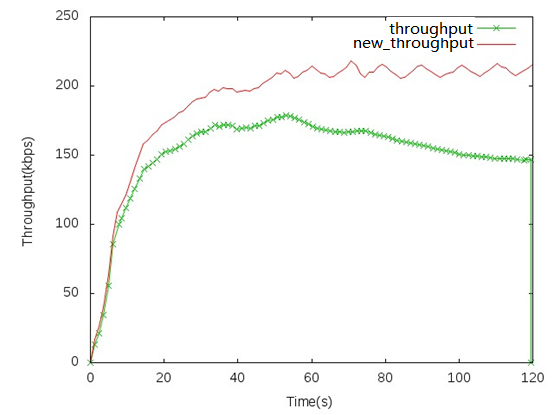


图4-4 吞吐量对比

## 本章小结

本章针对慢启动阶段Vegas协议在慢启动阶段不能有效利用带宽，参照TCP Westwood算法，采用带宽估计算法有效利用带宽。针对Vegas不能分辨链路拥塞还是路哟改变，采用了路径监视的策略。最后，使用NS2对改进的算法进行了验证。可以看出，改进的算法吞吐量要高于vegas算法的吞吐量。

# 系统设计与实现

## 需求背景与设计目标

### 需求背景

随着空间通信系统的发展，CCSDS建议逐渐成了空间通信的事实标准。特别是IP Over CCSDS Space Links提出以后，IP已经成为空间信息系统的标准上层应用。国外主要研究机构，如美国国家航空航天局，欧洲太空局等都利用CCSDS协议承载IP数据包来进行空间通信。在此背景下，我们和中科院空间科学与应用总体部联合，研究空间网络的IP化。

未来空间探测任务中信息量急剧增加，业务种类繁多复杂，各种信息的特点和对传输的要求差异大，信息源已经不是单点信息字，而是经过处理的信息集合，同时多用户信息源的信息需要通过同一物理信道传输。因此，未来空间探测任务对现有天地通信系统在支持综合业务、信息传送能力、信息传送容量和信息传送方式等方面提出了新的要求，采用基于标准协议的网络化天地通信系统已成为空间信息系统的发展方向。在我国的航天测控领域中，大量卫星测控数据传输仍采用的传统PCM测控传输帧方法，已经难以适应这些要求。尽管在神舟飞船、嫦娥探月卫星等任务中实验了CCSDS链路规范，但尚未形成完整的CCSDS标准的空间信息传输网络体系，尤其缺乏在IP融合技术、高性能传输技术等方面的研究成果。

目前我国迫切需要对天地通信网络的技术体制进行改进，适应未来航天器与地面系统之间任务信息传送的需求。特别是在未来我国的空间站建设任务中，天地通信系统除了要承载遥控遥测等传统业务以外，还需要开展电话语音通信、视频直播、文件传输以及天地互动的科学实验等应用业务。在这样的需求背景下，当前专用通信体制无法满足未来丰富业务通信的需求。因此，需要研究并构建空间的多航天器之间、航天器与空间通信卫星之间，以及与地面之间基于网络的通信系统及技术。2012年6月，神九任务中，开始测试了全新的“天地通信体制”，打破了以往“专线专用”的天地通信模式。采用IP体制，可以实现通信资源的调动。未来我国空间站建设中，对网络化天地通信的具体需求如下：

（1）支持基于IP的天地网络通信，支持可扩展的多种应用。未来的空间站需要支持可扩展的多种应用类型,并要求天地网络能够快速地实现与地面Internet、以及其他国际组织的系统进行互联。目前，专用的通信机制业务扩展能力差、数据传输能力不灵活、无法与其他标准的通信系统互通。因此，基于开放标准的IP技术和适用于卫星链路的CCSDS规范，构建基于网络的天地网络通信系统，将空间信息系统与地面网络互联，已成为国际和国内空间通信系统的发展方向。

（2）支持稳定、可靠、高效的天地网络数据传输。卫星等空间设备获取的数据载荷越来越多，要求天地网络通信的带宽更高、传输性能更强。由于天地链路具有长延迟、高误码、上下行带宽不对称和带宽资源有限等特点，在现有技术体系下构建基于IP的空间网络系统，存在一些技术难题，例如，TCP协议不会区分数据包丢失是由于传输错误还是由于拥塞造成的,丢包原因都被解释成网络拥塞，因此天地数据资源的可靠传输效率低下、实时视频和语音业务不流畅等问题。

### 设计目标

针对航天工程及其他空间应用中，信息量大，业务类型多，需要支持基于IP的网络通信等需求，实现基于IP的互联互通，保障网络的高可靠性、数据传输的高性能。IP over CCSDS网关实现了地面IP网络与空间网络的互联，使地面网络中的丰富应用扩展到了空间网络，并能够使用传输层性能增强技术来提高空间网络的数据传输效率。本章将设计实现一个基于IP over CCSDS网关系统，支持多种业务能力；基本构想是基于多层网络架构、CCSDS高级在轨系统（Advanced Orbiting System, AOS）规范、SCPS相关协议以及成熟的TCP/IP协议簇。

## 关键技术实现

### IPv4/IPv6 over CCSDS实现

IP over CCSDS能够在CCSDS的空间链路层协议上实现IP数据报的传递，将IP数据包作为与CCSDS源包同级的、标准的包格式，并在CA(Certificate Authority）建议书中规定。在空间链路子网（SLS,Space Link Sub-network）中传输的IP包必须使用其提供的多路复用业务或封装业务，并通过IP数据包的0-4bit的版本号的前3位来与其他数据包进行区分。

IP数据报通过CCSDS空间链路（空-空双向、空-地双向）协议进行传递的方法可以有三种选择：



图5-1 IP over CCSDS三种方式

（1）直接将IPv4、IPv6数据报置入一个或多个CCSDS数据链路帧之内或者以其他的CCSDS确认包来复用它们。这一选择使用了CCSDS AOS的虚拟信道包（VCP）、遥控VCP、遥控复用器接入点（MAP）包（MAPP）的传递业务；

（2）利用用户提供的串联数据流封装IPv4、IPv6数据报。将要传递的IPv4、IPv6数据报作为串行8位字节数据流置入CCSDS空间链路帧内。这一选择使用了CCSDS AOS虚拟信道接入（VCA）或遥控MAP接入（MAPA）传递业务；

（3）利用CCSDS封装业务来传递IPv4、IPv6数据报。将要传递的IPv4、IPv6数据报一一封装到CCSDS的封装包内，在一个或多个CCSDS空间数据链路帧中直接传递封装包。这一选择使用了CCSDS封装（ENCAP）业务。

由于（2）、（3）都是封装，因而可合并成两种选择如图2-7所示。

IP数据报

AOS

CCSDS数据报封装

方法1

方法2

CCSDS空间链路

图5-2 IP over CCSDS数据链路层实现

方法1只是用来传输非压缩的IPv4数据报。方法2是用来传输IPv4数据报或者IPv6数据报，压缩的和未压缩的都可以。

（1）CCSDS空间链路数据层协议能在CCSDS所有版本内直接支持IPv4数据报而无需任何中介融合层。CCSDS已经定义了一个支持IPv4报文在CCSDS空间链路上传递的业务表列。CCSDS数据链路协议支持两类IPv4报的传递，即属于主信道的和虚拟信道的。由于用户只能用CCSDS虚拟信道复用不同的数据包类型，因而仅关心虚拟信道的传递，主要目的就是推荐最合适于IP over CCSDS链路传递的业务。将非压缩的IPv4数据报直接置入CCSDS传递帧内，通过CCSDS空间数据链传递。此时，仅使用IPv4数据报头未压缩的IPv4数据报。IPv4数据报头的长度域是用于支持在传递帧之中来界定IPv4数据报。亦即在一个或多个CCSDS空间数据链路帧之内独立传递未经压缩的IPv4数据报或者以其他CCSDS确认包复用未压缩的IPv4数据报，使用了CCSDS AOS传递业务；

（2）将有压缩头的或非压缩的IPv4数据报或IPv6数据报利用CCSDS封装服务一一置入CCSDS封装包内。然后直接将封装包传递入一个或多个CCSDS空间数据链传递帧之内。通过CCSDS AOS使用CCSDS封装（ENCAP）业务。对于压缩的IPv4数据报头由于其长度域不再能用于界定数据报。因而，要在封装包头部提供各封装包的长度。由于IPv6数据报没有为CCSDS确认的包版本号（PVN），对于IPv6不管是否压缩本方法总能使用。

CCSDS空间链路数据层协议能在CCSDS所有版本内直接支持IPv4数据报而无需任何中介融合层，并将有压缩头的或非压缩的IPv4数据报或IPv6数据报利用CCSDS封装服务逐个置入CCSDS封装包内，从而完成空基网络和地基网络的无缝链接。

### AOS调度算法与实现

将因特网与空间网络结合以达到天地一体化网络通信的方案存在诸多问题需要研究。首先，由于因特网中的应用种类繁多、数据突发率高；而在原有AOS数据链路协议中，针对不同数据类型传输的虚拟信道有限，不同应用类型使用不同的虚拟信道传输，并且应用的带宽固化为某一定值。AOS协议的这些特点无法满足因特网中现有数据的特点。其次，采用AOS协议后，各数据系统的数据源类型繁多，其数据大多也是随机的产生，并且不同的数据对于完整性和实时性的要求差异大。如何有效地利用虚信道以及选择系统合适的虚信道调度方案，以达到传输数量更多、质量更高的信息，已经成为工程设计中的难点。

鉴于已有的流控算法，如令牌桶算法，不能直接支持AOS中的流量控制要求，并且结合AOS中虚拟信道的特点，提出了一种基于AOS虚拟信道的服务保障机制，通过将不同的数据按优先级分类，即不同优先级的数据使用不同的虚拟信道传输，然后对每个优先级的数据分别做流量控制，最后再采用优先级区分多级反馈队列调度算法对封装好的报文进行调度。

将基于AOS协议传输的数据分为不同的优先级。对优先级较高的数据在封装以及调度时提供一些特殊的质量保障机制，从而达到优先保证高优先级数据得到及时可靠传送，以提高整个系统的服务保障能力。

采取的质量保障机制包括数据进入AOS封装队列之前的流量控制以及基于AOS虚拟信道的调度算法。其中，流量控制算法，采用了关联令牌桶实现。

我们则是对虚拟信道调度算法进行了改进，鉴于已有的流控算法，如令牌桶算法，不能直接支持AOS中的流量控制要求，并且结合AOS中虚拟信道的特点，提出了一种基于AOS虚拟信道的服务保障机制，通过将不同的数据按优先级分类，即不同优先级的数据使用不同的虚拟信道传输，然后对每个优先级的数据分别做流量控制，最后再采用优先级区分多级反馈队列调度算法对封装好的报文进行调度。

将基于AOS协议传输的数据分为不同的优先级。对优先级较高的数据在封装以及调度时提供一些特殊的质量保障机制，从而达到优先保证高优先级数据得到及时可靠传送，以提高整个系统的服务保障能力。服务保障模型如图5-3所示：



图5-3 AOS服务保障机制模型

采取的质量保障机制包括数据进入AOS封装队列之前的流量控制以及基于AOS虚拟信道的调度算法。其中，流量控制算法，采用了关联令牌桶实现。

AOS应用场景中，应用层数据具有不同的优先级。由于不同虚信道传输的数据服务等级不同，而数据类型与虚信道一一对应，则相当于虚信道本身也就拥有了不同的服务等级。因此，对虚信道的调度也就成了为不同应用数据提供服务保障的措施之一。结合高优先级调度算法对于优先级高的任务的快速响应，以及多级反馈队列的优点，提出了拥有优先级区别的多级反馈调度算法。

由于虚信道和多级反馈调度队列均拥有优先级，故可以将二者一一对应起来。同时，对于时间片的分配也不能再采用原有分配原则，而是需要兼顾优先级和数据量的多少共同决定。拥有优先级区别的多级反馈队列模型如图2-9所示：



图5-4拥有优先级区分的多级反馈队列模型

其中，就绪队列的优先级规则是P(i)>P(i+1)，虚信道的优先级规则是Pa>Pb>Pc>Pd>…。为体现出虚信道的优先级差异，结合就绪队列的优先级情况，故将虚信道和就绪队列分别按照优先级从高到底的顺序依次对应起来。对于每个就绪队列，其时间片根据虚信道的数据量的实际情况给出。此时的调度规则是：

（1）各虚信道的数据依次进入对应的就绪队列末尾；就绪队列内部的数据按照先来先服务的原则调度；

（2）各就绪队列所分配的时间片由对应虚信道的数据量的多少决定。如果就绪队列内某虚信道的数据在时间片规定内未调度完毕，则将该虚信道对应数据转移至下一级就绪队列末尾；若该虚信道的数据在时间片规定时间内调度完毕，则继续调度当前就绪队列内其他虚信道的数据；

（3）仅当就绪队列i中没有数据需要被调度时，才调度就绪队列i+1中的数据。如果正在调度就绪队列i中的数据，而优先级高的就绪队列中出现数据，则立刻调度优先级高的就绪队列中的数据。

在采用优先级区分的多级反馈队列调度算法中，为了达到公平性，可以为“更繁忙”的虚信道分配更多的时间片。由于各数据源分系统产生数据包的速率可被认为是满足相互独立的泊松过程，那么各数据源产生的同类数据包的合成也就满足速率更高的泊松分布；同理，同一服务级别的多虚信道的数据合成也继续保持泊松过程所具备的特性。

## 软件设计实现

### 系统总体设计

系统总体功能如图5-5所示：



图 ‑5 CCSDS网关核心功能

系统可以被分为6个模块，其分别是AOS链路层模块、调度模块、总线通信接口模块、IP over CCSDS模块、物理链路接口模块和传输层性能增强模块。各个模块之间的链接关系如**Error! Reference source not found.**5-5所示：



图5-6模型链接关系

模块功能介绍：

（1）传输层性能增强模块

参考SCPS规范，设计了基于双网关的传输层增强技术方案，并实现的传输层增强网关软件。传输层增强网关软件实现了SCPS传输层协议和对特殊网络链路支持的拥塞控制算法，以最大限度地利用网络带宽。该网关对于终端用户完全透明，通过双网关技术，将端到端的连接分解为三段进行管理。

（2）AOS链路层模块

参照“AOS Space Data Link Protocol”和IP over CCSDS规范。首先，实现了AOS链路层的包服务、位流服务、虚拟信道访问服务、虚拟信道操作控制域服务、虚拟信道帧服务（VCDU的处理）、主信道帧服务、插入服务；其次，基于本课题的场景需求，设计并实现了协议数据单元转换功能模块和通信调度功能模块；并且，实现基于虚拟信道的流量控制功能。

另外，专用系统的数据包是VCDU，即原始AOS帧，因此网关还支持对原始帧的传输。网关中实现对所有虚信道（包括原始帧虚信道）的流控，原始帧Forward过程中也需要放到对应的服务上，而不能直接发送；对于Inwards的原始帧需要鉴别出来，并且不通过网关的AOS层，直接转发到内部网络。

（3）IP over CCSDS模块

为了支持空间信息系统内部和空间信息系统到地面Internet的端到端IP网络通信，网关中实现了IP over CCSDS封装功能。

（4）调度模块

负责数据来源和出口的通信和调度，通过分析数据包类型将数据解析或封装发送到对应的服务接口，如AOS层的包服务、VCDU服务接口，以及TCP增强处理子系统接口；或者实现总线数据单元的解析与封装并通信能力。

（5）总线通信接口模块

实现了面向空间信息系统总线的收发线程。天上网关与相连的NT卡通信，它将IP报文封装到FC\_AE\_1553B帧中，通过NT卡经由NC转发给相应的天上网闸，同时接收并解封来自NT卡的1553B帧。地上网关与地上网闸间建立两个TCP连接，分别转发上行和下行的网络数据。

（6）物理链路接口模块

收发线程，将AOS数据帧发送给对等的网关。

### 主要工作流程

软件系统系统由天上网关与地上网关两部分组成。网关系统的外部信息流如图5-7所示：

CCSDS网关

CCSDS网关

关

天地链路

NC卡

网闸

网闸

IP报文

FC\_AE\_1553数据包J

AOS数据帧

IP报文

VCDU

指令控制

NC卡

NC卡

NC卡

图5-6模型链接关系

（1）AOS传输帧构造

将IP数据报封装成AOS帧的过程。AOS数据帧构造过程如图所示。



图 5-7 AOS数据帧构造

多路复用过程

多路复用过程的功能是将变长的数据报包装到定长的复用协议数据单元M\_PDU中，即为每个由VCDU－ID参数所标识的VC构造各自的M\_PDU。该过程的处理对象可以是CCSDS允许传输的各种包，但在这里，只讨论数据为IP数据报的情况。对于IPv6的数据报，应该先使用包装业务将其包装在CCSDS包装包内，而对于Ipv4则不用，可以直接封装在M\_PDU内部，其PVN（版本号）是不一样的，以此来做区分。Ipv4的中版本号是四位，但在这里只使用前3位来标识。Ipv4数据报的PVN为“010”，而CCSDS包装包的PVN为“111”，因为包装包可以用来包装多种类型，包装包内的数据类型用协议标识字段来标识。由于传输帧的长度是一定的，所以对于超出M\_PDU长度的数据报要进行分割，与传统的分片不一样，分片过程必须生成数据报头部，而分割则不需要，因为在CCSDS信道中数据是有序的传送到目的地的M\_PDU格式如图所示。



图5-8 PDU包格式

该过程还有一个排队算法。针对同一主信道里不同虚拟信道上传输的帧，按一定的顺序把这些帧排成队列。这个排队算法可以根据需要自己定义，这个算法与虚拟信道的分配规则、发送速率、同步时间要求等有关，对于一些用来传输实时性要求比较高的数据的虚拟信道，可以给更高的优先级。一种典型的算法是轮询，首先把数据源分成若干种类，并按其数据源对传送的要求给定优先级。虚拟信道的复用机制一方面符合时分的特点，另一方面一定程度的满足了QoS的需求。为保持传输流的完整性，如果在一个帧发送时隙没有有效的帧时就生成一个空帧，并将其VCID（虚拟信道号）置为全1，该帧不参与计数。

插入及差错控制过程

面向物理信道上所有帧的这个过程主要用来完成插入业务和差错控制—包括帧头部差错控制和帧的差错控制。虚拟信道过程生成了基本的传输帧结构，而该过程用来填充其中的空部分。如果该虚拟信道不需要插入业务与差错控制，此处可以什么都不做。

帧编码过程

CCSDS主要有两种编码方案:R-S和Turbo。它们都提供了块级（分组）的纠错能力，比卷积编码的纠错能力要强许多。这是个可选的过程，主要是根据信道的误码率等来选择。如果启用了这个过程，那么在传输帧中，帧差错控制域的存在就显得多余，因为R-S提供了足够的发现错误的能力。在应用中经常将R-S编码与下面所要说的卷积编码构成级联编码。目前国内已有专门的CCSDS编码器，支持RS码，Turbo码，伪随机码以及卷积码。

bit转换过程

当传输的数据出现大量的连续‘0’或者连续‘1’的话不适应信道的传输特性。一个简单实用的方法就是用伪随机序列生成器生成一个固定长度的随机序列与VCDU/CVCDU的每位进行异或，这样得到的序列中1、0的分布将会变得比较均匀，在解码的时候同样使用这个序列进行每位异或即可。使用这个过程既可以加强原序列中0、1的随机性，同时增强了安全性以及保密性，经过这个过程得到的序列对不知道这个伪随机序列的人来说是不可识别的。

定界过程

该过程主要是同步标识的添加，是为了在接收端通过识别这个特定的比特位串来进行同步。通过给每个VCDU/CVCDU附加同步头标志前缀而形成CADU，并由VCDU伪随机序列生成器过程所确定的顺序产生一个连续和邻接的CADU流，从而形成PCA-PDU。每个CADU占一个物理信道上的同步时隙。同步标志的添加是在R-S与Turbo编码之后添加的，即它是不需要进行R-S或Turbo编码的。

其格式如图3-8所示。

图5-9VCDU/CVCDU附加同步头标志

卷积编码过程

卷积编码过程是对CADU的编码过程，主要有两种卷积编码来对CADU编码:基本卷积编码和收缩卷积编码。卷积编码提供的是比特级的前向纠错能力，适合纠正随机错误，在误码率较高的空间环境下，仅仅使用单个的编码方案可能不能获得足够的纠错效果，它一般情况下都与纠错力较强的R-S编码一起构成级联编码。

（2）传输层增强方案设计

SCPS在实际应用中，主要有3种具体实现方式：

1、端到端SCPS方式：在涵盖地面网络整个网络中，保证传输层与应用层端到端的链接，Internet网络层IP协议仅在卫星地面站处与空间网络层SCPS-NP进行互相转换，或者将SCPS-NP封装到IP的可选域；

2、层双网关方式：利用空间网络对分离的地面网络进行中继，保证应用层协议端到端的连接，地面Internet网络路由器交换机等商用设备不需要做任何改动，仅在地面网关进行TCP协议与SCPS-TP协议之间的转换，如下图所示；

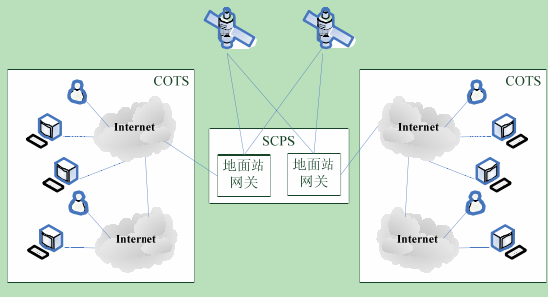


图5-10 传输层双网关方式结构

3、输层单网关方式：将地面终端分为Internet网络终端和支持SCPS的地面终端，利用单一地面站网关进行协议转换。

在空间探索领域，一般采用端到端的SCPS方式，在融合地面Internet与空间网络的一体化网络结构中，传输层双网关方式不需要对现有成熟的Internet网络设备与终端系统做过多的改动，且在空间通信链路能够应用SCPS-TP协议克服空间信道的通信制约。

其关键函数执行过程如下图所示：

图5-11传输层增强代码流程

## 系统功能测试

### 系统配置及编译

演示和应用场景抽象如图5-12所示，分为多个不同的网络，他们之间可能是路由相连接，我们的目的是使得整个网络是通的，也就是可以从PC1到PC2，在正常物理网络是通的情况下，我们在GW1和GW2上分别启动tun设备，使数据流经过tun设备，也就是经过GW1和GW2的数据分别需要经过他们的tun设备进行转发，也就需要在其上添加对应的路由，来实现经过tun转发。



图5-12网络连接抽象图

|  |  |  |
| --- | --- | --- |
| 设备名称 | 网卡号 | IP地址 |
| PC1（地面主机） | Eth0 | 192.168.1.111/24 |
| GW1（地面网关） | Eth0（左） | 192.168.1.123/24 |
|  | Eth1（右） | 10.1.0.1/24 |
| GW2（空间网关） | Eth1（左） | 10.1.0.2/24 |
|  | Eth0（右） | 10.2.0.1/24 |
| PC2（空间主机） | Eth0 | 10.2.0.2/24 |

表5-1 IP地址配置

路由配置如下：

|  |
| --- |
| root@GW1:~/gateway\_ground/bin# route -n  内核 IP 路由表  目标 网关 子网掩码 标志 跃点 引用 使用 接口  0.0.0.0 192.168.1.1 0.0.0.0 UG 0 0 0 eth0  10.1.0.0 10.1.0.1 255.255.255.0 UG 0 0 0 eth1  10.2.0.0 10.1.0.1 255.255.255.0 U 0 0 0 eth1  192.168.1.0 192.168.1.123 255.255.255.0 UG 0 0 0 eth0 |

|  |
| --- |
| root@GW2:~/gateway\_space/bin# route -n  Kernel IP routing table  Destination Gateway Genmask Flags Metric Ref Use Iface  0.0.0.0 10.1.0.1 0.0.0.0 UG 0 0 0 eth1  10.1.0.0 0.0.0.0 255.255.255.0 U 0 0 0 eth1  10.2.0.0 0.0.0.0 255.255.255.0 U 0 0 0 eth0  192.168.1.0 0.0.0.0 255.255.255.0 U 0 0 0 eth1 |

以下是网关启动配置：

|  |
| --- |
| 1. gateway\_ground\bin\control\_config.txt的内容介绍   config\_path=./ 配置文件的路径，这里是当前路径  gw\_addr=10.1.0.2 本端到对端可以达到的最远距离，这里是gw2的eth0  port=9999 端口  gateway\_ground\bin\global\_config.txt的内容介绍  used\_encrypt=false 关闭encryption  used\_scps=false 关闭scps  used\_errbit=false 关闭errbit  errbit\_m=1,9  errbit\_n=4,7  phy\_delay=0 关闭延时，开启为设定值n  peer\_gw\_ip=10.1.0.2 对端网关接收IP，本端可以到达，用来接收udp  kjz\_udp\_ip=10.2.0.2 天上ip  delay\_queue\_size=8100000  gateway\_space\bin\control\_config.txt的内容介绍  config\_path=./ 配置文件的路径，这里是当前路径  gw\_addr=10.1.0.1 本端到对端可以达到的最远距离，这里是gw1的eth1  port=9999  gateway\_space\bin\global\_config.txt的内容介绍  used\_encrypt=false 关闭encryption  used\_scps=false 关闭scps  used\_errbit=false 关闭errbit  errbit\_m=1,9  errbit\_n=4,7  phy\_delay=0 关闭延时，开启为设定值n  peer\_gw\_ip=10.1.0.1 对端网关接收IP，本端可以到达，用来接收udp  kjz\_udp\_ip=10.2.0.2 天上ip  delay\_queue\_size=8100000  运行软件  按照上面的信息配置好，运行gatewayd，添加路由，即可实现udp转发，如：  在GW1上运行gatewayd，之后添加路由，使通过seg\_tun\_dev到达天上网络，  route add -net 10.2.0.0/24 dev seg\_tun\_dev  在GW2上运行gatewayd，之后添加路由，使通过seg\_tun\_dev到达地上网络，  route add -net 192.168.1.0/24 dev seg\_tun\_dev |

### 系统功能改进及运行结果

本系统是基于IP over CCSDS的网关加速软件，其传输层增强运用的就是SCPS-TP协议。在第三四章，我们使用了NS2对改进的Vegas进行模拟，这一节，我们将基于SCPS-TP代码本身对其进行改进。

下表是SCPS\_RI\_1\_1\_144的文件夹列表：

|  |  |
| --- | --- |
| apps | SCPS套接字接口实例 |
| bin | 可执行代码目录 |
| docs | 用户手册 |
| FP | SCPS-FP协议源代码 |
| Include | SCPS协议头文件 |
| Lib | SCPS库文件 |
| NEW\_DOCS | 2006年之后新加入的用户手册 |
| source | 包含SCPS-TP、SCPS-SP、SCPS-NP的源代码 |

表5-2 SCPS文件夹列表

其中SCPS-TP协议源代码在source文件夹中。SCPS-TP支持Vegas、VJ和pure rate三种拥塞算法。默认情况下是pure rate。其中与Vegas协议紧密相关的有tp\_process.c和tp\_timer.c两个c文件，前者负责SCPS-TP的接受处理机制，而后者负责SCPS-TP定时器机制。其中SCPS-TP关于Vegas慢启动阶段和拥塞避免阶段的代码修改位于tp\_timer.c文件的tp\_TFVegas函数中。

|  |
| --- |
| void tp\_TFVegas( tp\_Socket \* s )  {  int cwinsegs;  int diff;  uint32\_t new\_cwnd;  struct timeval mytime;  int old\_state; mytime.tv\_sec = mytime.tv\_usec = 0;  if ( (s->rt\_route->rttcur) && (s->rt\_route->rttbest) )  {  cwinsegs =min( (s->snd\_prevcwnd / s->maxseg), max( (s->snd\_prevcwnd -  (s->snduna -  s->rtseq) ) / s->maxseg, 1 ) );  diff = cwinsegs - ( (cwinsegs \* s->rt\_route->rttbest) / s->rttcur);  diff = max( diff, 0 );  new\_cwnd = s->snd\_prevcwnd;  old\_state = (s->sockFlags & TF\_CC\_LINEAR);  if ( s->sockFlags & TF\_CC\_LINEAR )  {  if ( diff < s->VEGAS\_ALPHA )  {  new\_cwnd += s->maxdata;  /\* s->snd\_cwnd += s->maxdata; \*/  if ( new\_cwnd > 0x7fffffff )  {  new\_cwnd = 0x7fffffff;  }  }  if ( diff > s->VEGAS\_BETA )  {  if ( (int) new\_cwnd <= (int) s->maxdata )  {  s->sockFlags &= ~TF\_CC\_LINEAR;  /\* hit bottom - go exponential \*/  new\_cwnd = s->maxdata;  }  }  } /\* linear \*/  else{ /\* exponential \*/  if ( (s->VEGAS\_SS == 0) || ( (s->VEGAS\_SS |= 0) && (new\_cwnd == s->snd\_ssthresh) ) )  {  if ( diff > s->VEGAS\_GAMMA )  {  s->sockFlags |= TF\_CC\_LINEAR;  } else if ( new\_cwnd < 0x7fffffff )  {  new\_cwnd = (new\_cwnd << 1);  }  if ( s->VEGAS\_SS == 0 )  {  s->snd\_ssthresh = new\_cwnd;  }  }else  s->snd\_ssthresh = new\_cwnd;  /\* Help new connections "get off the ground"! \*/  if ( new\_cwnd == s->maxdata )  new\_cwnd = (new\_cwnd << 1);  if ( (s->rt\_route->sendpipe) && (s->rt\_route->sendpipe < new\_cwnd) )  {  new\_cwnd = s->rt\_route->sendpipe;  s->sockFlags |= TF\_CC\_LINEAR;  }else if ( s->app\_sbuff->max\_size < new\_cwnd )  {  new\_cwnd = s->app\_sbuff->max\_size;  s->sockFlags |= TF\_CC\_LINEAR;  }  }  if ( (int) new\_cwnd <= (int) s->maxdata )  {  s->sockFlags &= ~TF\_CC\_LINEAR; /\* hit bottom – go exponential \*/  new\_cwnd = s->maxdata;  }  s->snd\_cwnd += (new\_cwnd - s->snd\_prevcwnd);  if ( (int) (s->snd\_cwnd) < 0 )  {  s->snd\_cwnd = 0;  }  s->snd\_prevcwnd = max( new\_cwnd, s->maxseg );  s->snd\_cwnd = max( (int) (s->snd\_prevcwnd - (s->seqsent - s->snduna) ), 0 );  mytime.tv\_usec = s->rt\_route->rttcur;  s->rt\_route->rttcur = 0;  }  } |

上述代码中linear线性增长表示在CA（congestion avoid）阶段，exponential指数增长表示在slow-start阶段。为测试三、四章算法，本文分别根据三四章的算法在SCPS-TP中对于Vegas进行改写并运行。

为了验证数据的传输速率，本文采用FileZilla软件来统计验证改进算法的具体效果。FileZilla作为一个开放源代码的FTP软件，包含客户端和服务器端两个版本。FilZilla包含了一般FTP软件所有的功能，同时它不仅支持IPv4的数据传输，还支持IPv6。基于以上因素的综合考虑，决定在本文实验中使用FileZilla软件。PC-Send端和PC-Recive端使用的版本均为3.7.3。

传统Vegas算法在时延为200ms，误码率为10-7-10-5环境下的传输速率为100.4KB/s；在时延300ms，误码率为10-7-10-5环境下速率则降低到了87.0KB/s；在时延400ms，误码率为10-7-10-5环境下速率为75.2KB/s。在误码率恒定在10-7-10-5范围内，随着时延的增加，小文件的数据传输速率随之降低。

改进后的Vegas算法在时延为200ms，误码率为10-7-10-5环境下的传输速率为225.5KB/s；在时延300ms，误码率为10-7-10-5环境下速率则是182.6KB/s；在时延400ms，误码率为10-7-10-5环境下速率为152.0KB/s。对比之前传统Vegas，可以看出，在传输小文件时，改进后的Vegas算法传输速率最快。详细对比见表5-3所示。

表5-3 不同算法在不同环境下的传输速率对比

|  |  |  |  |
| --- | --- | --- | --- |
| 时延（ms） | 误码率 | 算法 | 传输速率（KB/s） |
| 200 | 10-7-10-5 | Vegas | 41.1 |
| 第三章Vegas | 200.4 |
| 第四章Vegas | 225.5 |
| 300 | 10-7-10-5 | Vegas | 30.7 |
| 第三章Vegas | 187.0 |
| 第四章Vegas | 182.6 |
| 400 | 10-7-10-5 | Vegas | 25.2 |
| 第三章Vegas | 175.2 |
| 第四章Vegas | 152.0 |

## 本章小结

为了验证第3,4章的研究成果，本章使用基于IP over CCSDS的软件系统对其进行运行校验。首先对IP/IPv6 over CCSDS和AOS调度算法的实现进行了详细描述。接着从系统总体设计和主要工作流程两个方面介绍了软件的设计实现。然后介绍了本文的网络系统。对其中网关程序的主要功能，软件模块以及工作线程分别进行了深入的分析，最后采用支持跨平台的FileZilla来检验对比最终效果。数据显示，使用了新算法的数据传输效果要远远高于未使用的数据传输效果。

# 结论与展望

## 本文总结

目前天基和地基网络的通信协议并没有统一的设计和建设，这在星地和星间通信中尤其明显。各地面通信系统在互联网的迅速发展下，已基本全部采用TCP/IP，以支持各种各样的应用要求，实现了地面通信协议的大统一。而各卫星系统在任务多样复杂、缺乏统一标准的情况下，往往采用各自独立开发的通信协议，不能很好地与地面应用网络相结合，天地各网络间的通信也必须通过网间的协议翻译转换来实现。 SCPS 协议在上世纪末提出，其研究目的是在兼容地面互联网的基础上支持可靠的高速率数据传输，支持并行任务多节点之间网络路由功能选择，并与地面具有良好互操作性等。SCPS协议解决了空间数据通信中存在的诸多问题，如：信号强度弱、信道噪声大、传输时延大、带宽时延积大、多普勒频移较大以及链路容易中断等问题。经过多年的发展，SCPS 协议在空间通信网络中得到了广泛的应用，同时也暴露出了一些亟待解决的问题。基于这种情况，本文在对TCP Vegas协议进行研究与分析的基础上，指出了其存在的问题与不足，然后对其进行了改进，最后使其应用于系统中，具体工作内容如下：

第一，TCP协议的拥塞控制进行了介绍，传统的拥塞控制机制主要包括了慢启动、拥塞避免、快速重传和快速恢复这个4个阶段。然后分析了SCPS-TP协议对于传统TCP协议进行的改进；最后，介绍了目前对于SCPS-TP改进的方法。

第二，首先对Vegas拥塞控制算法的原理进行了介绍，然后总结了研究人员对其做出的一些改进工作，并指出了不足之处。针对慢启动阶段包突发现象，使得Vegas算法过早结束慢启动阶段，本节算法采用了每个RTT增加一半cwnd的策略。针对拥塞避免阶段，窗口增长速度不能尽快充分利用带宽，本节算法采用了自适应的窗口增长策略。本文采用NS2对所做的工作进行仿真，因此介绍了NS2仿真的工作流程，紧接着对仿真环境进行详细描述。通过仿真可以看出，新算法优于原有Vegas算法。

第三，针对慢启动阶段Vegas协议在慢启动阶段不能有效利用带宽，参照TCP Westwood算法，采用带宽估计算法有效利用带宽。针对Vegas不能分辨链路拥塞还是路哟改变，采用了路径监视的策略。最后，使用NS2对改进的算法进行了验证。可以看出，改进的算法吞吐量要高于vegas算法的吞吐量。

第四，为了验证第3,4章的研究成果，本章使用基于IP over CCSDS的软件系统对其进行运行校验。首先对IP/IPv6 over CCSDS和AOS调度算法的实现进行了详细描述。接着从系统总体设计和主要工作流程两个方面介绍了软件的设计实现。然后介绍了本文的网络系统。对其中网关程序的主要功能，软件模块以及工作线程分别进行了深入的分析，最后采用支持跨平台的FileZilla来检验对比最终效果。数据显示，使用了新算法的数据传输效果要远远高于未使用的数据传输效果

## 展望

本文对上述几个问题做出了分析与研究，并给出了具体的算法和实验过程。然而目前还有许多问题尚未解决，等待进一步的分析与验证。总结如下：

1）SCPS-TP中SNACK选项的作用需要进一步的深入研究。

2）SCPS-TP中许多参数尚待优化。

3）SCPS-TP如何跨层管理将是下一步研究的重点。

# 致 谢

本论文是在我的导师罗光春教授的悉心指导下完成的。罗光春教授深厚的理论素养，丰富的实践知识，开拓了我的视野，使我的科研能力有了很大的提高；罗老师平易近人、求真务实、积极进取的人生态度也深深影响着我；在生活上，罗老师更是给予我无微不至的关怀。在此衷心感谢罗老师对我的关心和指导。

2012年9月，我加入新型网络与中间件教研室，在这三年里陈爱国副教授悉心指导我完成了教研室的科研工作，在学习和生活上都给了我很大的帮助和关心，在此向陈老师表示衷心感谢。

田玲副教授、秦科副教授、段贵多副教授对于我的科研工作和论文都提出了许多的宝贵意见，在此表示衷心的感谢。

在教研室期间，李春虎师兄和钟俊师兄都对我论文中的研究工作做出了很大的帮助，在此向他们表达我的感激之情。

最后，感谢我的家人，他们的理解和支持使我能够在学校专心完成我的学业。

# 参考文献

1. 樊秀梅，单志广，张宝贤，等. 容迟网络体系结构及其关键技术研究[J]. 电子学报，2008,36(1): 161-163.
2. 熊永平，孙利民，牛建伟，等. 机会网络[J].软件学报，2009, 20(1): 125-127.
3. 张俊宝. 延迟容忍网络路由协议研究[D]. 成都：电子科技大学，2012:1-30.
4. 王文柏. DTN网络中拥塞控制研究[D]. 合肥：中国科学技术大学，2011:2-20.
5. K. Fall. A delay-tolerant network architecture for challenged internets[C]. In: Proceedings of the ACM SIGCOMM Conference on Computer Communications, Karlsruhe, Germany, 2003:27-34.
6. 柏溢，王民北，丁大钊. 使用改进SNOOP协议改善无线链路TCP的性能[J]. 计算机工程与应用，2006,19:126-128.
7. S. Vangala,M A Labrador．Performance of TCP over Wireless Networks with the Snoop Protocol[C]．In：the 27th Annum IEEE Conference on Local Computer Networks，2002:600-601.
8. 陈宇，孟新，卞春江，等. 基于 SCPS-TP 的TCP 性能增强代理[J]. 计算机工程，2011,37:89-91.
9. Sallantin, Renaud．Initial spreading: A fast Start-Up TCP mechanism [C ]. Local Computer Networks (LCN), 2013 IEEE 38th Conference on,Sydney,Autralia, 2013:492-499
10. 刘光华，王辉. LEO卫星网络中TCP 协议性能研究[J]. 计算机工程， 2010, 36(14): 96-98.
11. W. Stevens. TCP Slow Start, Congestion Avoidance , Fast Retransmit , and Fast Recovery Algorithms [S].RFC 2001, 1997.
12. 黄科，TCP协议在空间通信中的应用研究[D]. 长沙：国防科学技术大学，2009:6-30.
13. Consulttative Committee for Space Data System. CCSDS710.0-G-0.4. Space Communication Protocol Specification(SCPS):Rationale,Requirement and Application Notes,Recommendation for Space Data System Standards. Draft Green Book[S], Washington,D.C.:CCSDS,1998.
14. Consulttative Committee for Space Data System. CCSDS713.0-B-1. Space Communication Protocol Specification (SCPS)-Network Protocol (SCPS-NP), Recommendation for Space Data System Standards. Blue Book[S],Washington,D.C.:CCSDS,1999.
15. Consulttative Committee for Space Data System. CCSDS713.5-B-1. Space Communication Protocol Specification (SCPS)-Security Protocol (SCPS-SP), Recommendation for Space Data System Standards. Blue Book[S],Washington,D.C.:CCSDS,1999.
16. Consulttative Committee for Space Data System. CCSDS714.0-B-2. Space Communication Protocol Specification (SCPS)-Transport Protocol (SCPS-TP), Recommendation for Space Data System Standards. Blue Book[S],Washington,D.C.:CCSDS,2006.
17. Consulttative Committee for Space Data System. CCSDS717.0-B-1. Space Communication Protocol Specification (SCPS)-File Protocol (SCPS-FP), Recommendation for Space Data System Standards. Blue Book[S],Washington,D.C.:CCSDS,1999.
18. C. Caini, R. Firrincieli, M. Marchese, et al. Transport layer protocols and architectures for satellite networks[J]. International Journal of Satellite Communications and Networking, July 2006, 25(1): 1~26.
19. 吴结，高随祥. 基于快速传输协议实现卫星TCP性能的改善[J].计算机应用，2006,26（7）：1563-1566.
20. C. Casetti, M. Gerla, S. Mascolo, et al. TCP Westwood: bandwidth estimation for enhanced transport over wireless links[J], Wireless Networks. 8 (5). 2002.467–479.
21. K N Srijith, L. Jacob, A L Ananda. TCP Vegas-A: Improving the Performance of TCP Vegas [J]. Computer Communications,2005,28:429-440.
22. 王建峰，黄国策，陈才强等. GEO卫星网中基于拥塞状态区分的TCP Vegas改进算法[J]. 计算机科学，2008 ,35(11):70-73.
23. 王云涛，方建安，张晓辉等. 基于TCP Vegas的网络拥塞控制改进算法[J] 计算机应用研究.，2009,26(12):4645-4647.
24. Wei Zhou,Wei Xing,Yongchao Wang. TCP Vegas-V: Improving the Performance of TCP Vegas[C]. Automatic Control and Artificial Intelligence (ACAI 2012), International Conference on ,xiamen,2012:2034-2039.
25. Rung-Shiang Cheng, Chun-Yu Ke. An Threshold-based Congestion Control Mechanism for Vegas TCP over Heterogeneous Wireless Networks.[C]. Wireless and Pervasive Computing (ISWPC), 2011 6th International Symposium on,Hongkong, 2011:1-6.
26. 王斌，陈元琰，冯伟等. TCP Vegas-b：TCP Vegas 改进算法[J]. 计算机工程与设计， 2011, 32(2):438-441.
27. N. Alipasandi, S. Jamali. An improvement over TCP Vegas by solving rerouting problem[J] Global Journal on Technology,2012(1).
28. Fuzhe Zhao, Shisong Xiao, Jixiong Chen. An Improved Vegas Scheme Based on Average Round Trip Time[C]. Intelligence Information Processing and Trusted Computing(IPTC), 2010 International Symposium on ,huanggang,2010:15-17.
29. 潘成胜，宣景朋，魏德宾等. 卫星网络中基于BaseRTT计算的TCPVegas算法改进[J].系统仿真学报，2012,24(6):1254-1258.
30. The Network Simulator - Ns-2. Website. http://www.isi.edu/nsnam/ns.
31. 徐雷鸣, 庞博, 赵耀. NS与网络模拟[M]. 北京: 人民邮电出版社, 2003.
32. 叶晓国，肖甫，孙力娟等. 基于 ns-2 的卫星网络仿真方法的研究[J]. 南京邮电大学学报（自然科学版）. 2009,29(1):79-82.
33. B. Radunovic． A Cross-layer Design of Wireless Ad－Hoc Networks［J］． IEEE Computer，2005，6( 12) : 78-84．
34. YinMin,TangYao,YuQuan. Cross-layer ideas in wireless network designs[C]. IEEE International SymPosium on Mierowave，Antenna， ProPagation and EMC Technologies for Wireless Communieations,2005,891-894.
35. V. Srivastava, M. Motani. Cross-Layer Design: a Survey and the Road Ahead[J]. IEEE Communications Magazine, 2005, 43(12):112-119.
36. 李蓉. 认知无线网络中跨层关键技术研究[D]. 北京 ：北京邮电大学，2012:20-35.
37. T. Raisinghani, S. Iyer. ECLAIR: Efficient cross layer architecture for wireless protocol stacks［M］．［s．l．］: TataInfotech Ltd，2004: 370-389.
38. Yingqun Yu,G.B. Giannakis. Cross-Layer Congestion and Contention Control for WireLess Ad HoeNetworks[J].IEEE Transaetions on wireless communications, 2008,7(1):37-42.
39. I.F.Akyildiz,Xudong Wang. Cross-Layer Design in Wireless Mesh Networks[J], IEEE Transactions on vehicular technology, 2008,57(2):1061-1076.
40. S. Primoz, K. Hamid. Aghajan, et al.Cross-Layer Optimization for High Density Sensor Networks:Distributed Passive Routing Decisions[J] .2004,ADHOC-NOW’04:266-279.
41. 袁银行. 无线网络基于跨层新策略的拥塞控制技术研究[D] 长沙：中南大学，2010:15-29.
42. S. Floyd. TCP and explicit congestion notification[J]. ACM Computer Communication Review, 1994, 24 (5): 10-23.
43. R. Ramani, A. Karandikar. Explicit Congestion Notification(ECN)in TCP over wireless networks[C]. Personal Wireless Communications,2000 IEEE International Conference on,Hyderabad,2000:495-499.
44. 蒋立正. IP over CCSDS 空间组网通信关键技术研究[D]. 北京：中国科学院研究生院，2009:20-55.
45. 潘成胜, 宣景朋, 魏德宾, 等. 卫星网络中基于 BaseRTT 计算的 TCP Vegas 算法改进[J]. 系统仿真学报, 2012, 24(006): 1254-1258
46. Consulttative Committee for Space Data System. AOS Space Data Link Protocol. Recommendation for Space Data System Standards, CCSDS732.0-B-2. Blue Book. Issue 2[S]. Washington, D.C.: CCSDS, 2006.

# 攻读硕士学位期间取得的成果

参与的项目

（1）2012年9月至2014年4月：\*\*\*技术研究