

中国科学院研究生院

博士学位论文

低轨通信星座自主网络管理关键技术研究

作者姓名: 江玉洁

指导教师: 梁旭文 研究员

中国科学院上海微系统与信息技术研究所

学位类别: 博士

学科专业: 通信与信息系统

培养单位: 中国科学院上海微系统与信息技术研究所

二〇一二年八月

Research on Key Techniques of Autonomous Network
Management for LEO Satellite Communication System

By

Jiang Yujie

A Dissertation Submitted to
Graduate University of Chinese Academy of Sciences
In partial fulfillment of the requirement
For the degree of
Doctor of Communication and Information System

Shanghai Institute of Microsystem And Information Technology

August, 2012

致 谢

随着本文即将完成,历时六年在职攻读博士学位的学习生活也将落下帷幕,在此,衷心感谢所有关心和帮助我的人。

首先,由衷地感谢我的导师梁旭文研究员。梁老师严谨求实的治学态度,精益求精的科研精神,锐意进取的工作作风,宽厚坚实的理论功底,都深深影响着我,使我在论文的研究过程中丝毫不敢懈怠和马虎,本文的完成离不开梁老师的热心关怀和悉心指导,在此谨向梁老师致以最诚挚的感谢!

感谢通信技术室主任刘会杰研究员。作为我工作部门的直属领导,刘主任给予我攻读博士学位极大的支持,无论从工作安排上、论文选题的指导下都给我提供了很大的帮助。刘老师渊博的学识、严谨的治学态度和敬业精神给我留下了深刻的印象,他言传身教,使我不仅在知识上得到了长进,独立科研能力也得到了锻炼和提高。在此向刘老师致以衷心的感谢!

特别感谢通信技术室副主任龚文斌博士。作为我工作项目的领导,龚博士时常关心我的工作发展、论文进度,是我真正的良师益友,同时龚博士在通信、电子领域渊博的知识,在工程项目上丰富的经验,都使我受益匪浅。感谢龚博士对我论文提出的宝贵意见,在此致以深深的谢意!

感谢帅涛、任前义,吴杲、姚晔、张炜,陈晓挺、贾铂奇,沈洪兴、姜泉江、马陆、漆庄平、金凤、何小苑、张亮、吕源、姜兴龙、梁广等同事在学习、工作中共同创造的和谐工作氛围以及多年来给予我的各种帮助。感谢我的妹妹江潇在本文软件仿真工作中对我的帮助。

感谢微系统所研究生部余翔波老师在我的论文研究工作中不厌其烦的关心。

感谢我的父母,感谢他们这么多年来对我的养育之恩。感谢我的公婆对我孩子的照顾,使我免去了后顾之忧。

最后,感谢我的丈夫陈辰,感谢他对我工作的支持、生活的照顾,在无数个为了论文奋斗的不眠之夜,他的鼓励和帮助是让我坚持下去的动力!感谢我可爱的女儿,她是上天赐予我跋涉途中最美的礼物!

低轨通信星座自主网络管理关键技术研究

江玉洁 (通信与信息系统)

指导老师：梁旭文 研究员

摘要

本文从低轨通信卫星星座系统自主管理的需求出发，对星座网络自主管理关键技术进行了研究，包括自主管理架构选择、星上自主位置管理策略、分布式网络故障诊断技术、自主路由更新技术等。主要创新点有：

- ✓ 针对自主管理架构，提出了基于约束条件的分布式分簇算法 (CDCA)，该约束条件规定同轨道的相邻节点组成簇的核心成员，充分利用了同轨道链路的稳定特性，通过将卫星层次化分簇的方法，提高了管理星簇的稳定性；
- ✓ 提出了基于簇的星上自主多 HLR 位置管理策略，并针对网络拓扑变化造成的簇内更新信令开销大的问题，提出移动 agent 与指针转发相结合的簇内位置更新策略；并给出了通过多 HLR 相互协作的方式进行位置查询的方案；理论分析了位置更新和位置查询的网络开销及网络侧的寻呼时延；
- ✓ 以系统级故障诊断理论为基础，提出一种适合于星上处理的分布式自主网络故障识别算法 (DSFD)，使得每个节点都可独立进行自识别，而无需获得整网节点状态，从而降低网络故障检测开销，缩短诊断时延。
- ✓ 提出了考虑切换和时延的权值路由算法，既能保证路由选择的优先级，又兼顾了网络流量的平衡，同时，采用节点实时状态与权值路由表相结合的方式选择路由，具备了一定程度的自主性；针对发生链路故障后的路由更新问题，以权值路由策略为基础，提出了一种基于移动 agent 的多层次扩散路由更新算法，在不增加过多的星上计算量的同时，提高时延性能，降低路由更新开销。

关键词 低轨卫星，自主网络管理，分簇，位置管理，网络故障诊断，路由算法

Research on Key Techniques of Autonomous Network Management for LEO Satellite Communication System

Jiang Yujie (Communication and Information System)

Directed by: Prof. Liang Xuwen

Abstract

For the demand of autonomous operation on LEO satellite constellation communication system, several key techniques on autonomous network management are researched in this dissertation, including framework selection of autonomous management, strategy of location management, network fault diagnosis algorithm, and independent routing algorithm. The key contributions are:

- ✓ A distributed clustering algorithm based constraints(CDCA) is proposed, which can enhance the stability of cluster by hierarchical clustering design of making the adjacent nodes in the same orbit as core cluster members.
- ✓ A multi-HLR location management strategy based on cluster is proposed, which discusses the location updating strategy of combining mobile agent and pointer forwarding and location query strategy of multi-HLR collaboration. The network side overhead and paging delay of the location management strategy are analyzed theoretically.
- ✓ A distributed network fault diagnosis algorithm for on-board processing on satellite (DSFD)is proposed, which based on system-level fault diagnosis theory(SLD). Each satellite node can detect itself independently without the whole network states, which reducing the test overhead and shortening the diagnosis delay.
- ✓ A weighted routing algorithm considering handover and time delay is proposed, which can give consideration to both the priority of routing selection and the network flow balance. Meanwhile, it is autonomous to a certain extent that nodes' real-time status and weight routing sheet are combined for routing. On account of route updating after inter-link fault occurring, a multi-layer diffusing route updating algorithm with mobile agent is also proposed,which can improve the performance of time delay and update overhead without increasing computing buget.

Keywords: LEO satellite, Autonomous network management, Clustering, Location management, Network fault management, Routing algorithm

目 录

致 谢	I
摘 要	II
ABSTRACT	III
目 录	IV
第一章 绪论	1
1.1 研究背景及意义	1
1.1.1 低轨通信卫星星座系统概述	1
1.1.2 星座网络自主管理的需求	2
1.1.3 问题的提出	3
1.2 自主管理体系结构	5
1.3 论文主要研究内容	6
1.3.1 自主管理分簇算法	6
1.3.2 星上位置管理技术	7
1.3.3 自主故障检测与识别技术	7
1.3.4 路由算法及自主路由更新技术	8
1.4 论文结构安排	8
第二章 星座网络自主管理簇架构的研究.....	10
2.1 引言	10
2.2 分簇算法国内外研究现状	10
2.3 星座网络拓扑分析	12
2.3.1 低轨卫星星座网络模型	12
2.3.2 星间网络拓扑分析	13
2.4 基于约束条件的分布式分簇算法	15
2.4.1 约束条件	15
2.4.2 管理簇的划分	17
2.4.3 管理簇的形成	23
2.5 仿真与分析	26
2.6 小结	29
第三章 星座自主运行中的位置管理策略研究.....	30
3.1 引言	30
3.2 位置管理概述	30
3.2.1 地面移动蜂窝系统中的位置管理	30
3.2.2 低轨移动通信星座系统中的位置管理	32
3.3 星座自主运行时的位置管理策略	33
3.3.1 基于卫星波束覆盖的动态位置区划分	33
3.3.2 基于簇的多 HLR 位置管理构架	34
3.3.3 基于簇的多 HLR 位置更新策略	36

3.3.4 多 HLR 协作的位置查询策略.....	46
3.4 性能分析	48
3.5 仿真与分析	49
3.6 小结.....	53
第四章 基于 SLD 的分布式网络故障诊断技术研究.....	54
4.1 引言	54
4.2 系统级故障诊断算法概述.....	54
4.2.1 系统级故障诊断理论	54
4.2.2 国内外研究现状.....	56
4.3 基于 SLD 的分布式卫星网络故障识别算法.....	59
4.3.1 卫星网络建模.....	59
4.3.2 故障测试过程.....	60
4.3.3 M-概率的分布式测试模型	60
4.3.4 小概率忽略的诊断规则.....	64
4.3.5 分布式故障识别算法	65
4.4 仿真分析	71
4.4.1 诊断完全率和诊断正确率.....	71
4.4.2 诊断开销与诊断时延	73
4.5 小结.....	74
第五章 基于移动 AGENT 的星上自主路由更新技术研究.....	75
5.1 引言	75
5.2 路由技术概况.....	75
5.2.1 卫星网络路由技术	75
5.2.2 移动 agent 路由技术	77
5.3 改进的基于静态拓扑快照的路由策略.....	79
5.3.1 静态拓扑快照分析	79
5.3.2 基于快照序列的路由策略.....	81
5.3.3 考虑切换和时延的权值路由算法	82
5.4 基于移动 AGENT 的路由更新算法	92
5.4.1 故障链路对路由性能的影响分析	93
5.4.2 基于移动 agent 的多层扩散路由更新算法	94
5.4.3 基于移动 agent 的系统设计	96
5.4.4 仿真分析	98
5.5 小结.....	101
第六章 总结与展望	102
参考文献	104
作者攻读博士学位期间发表的论文	115
作者简历	116

第一章 绪论

在下一代移动通信系统中，能够提供全球覆盖的卫星移动通信系统正成为重要组成部分。相对于中高轨卫星系统，低轨（LEO）卫星系统^{[1][2][3]}具有卫星功率小、路径损耗低、传播时延短、易于实现手持个人通信和系统稳定性高等优势，非常适合移动通信。未来 LEO 卫星系统有效的窄带信息接入能力以及宽带卫星系统的传输语音、数据、视频业务等多媒体高质量综合业务信息能力，不仅可作为地面网络的有效支持补充，且具有地面有线传输媒体乃至地面无线传输媒体无法媲美的三维无缝隙覆盖和广域连接的独特魅力，因此它必将在未来信息高速公路中充当重要的角色和发挥卓越的作用。世界上已运行的低轨卫星系统包括提供实时数据和语音业务的 Iridium 系统、Globalstar 系统、以短数据传输为主的 Orbcomm 系统等。值得注意的是新铱星公司的“下一代铱星”（Iridium Next）计划，它将是一个基于 IP 的宽带网络，采用最新的卫星技术和无线通信技术，可以提供超过现有服务的扩展服务——从高带宽数据传输到增强的语音和短信服务，还支持特定的天基应用。近年来国内也一直致力于低轨通信星座系统相关技术的研究，包括星座组网技术、星间链路技术、切换技术、路由交换技术、星地一体化管理技术等等，取得了一定的成果，对我国低轨通信卫星星座系统的建设有很大的推动作用。随着航天技术的发展，卫星上处理能力也日渐提高，星座系统自主运行^[4]和管理正逐渐成为未来的发展趋势。

1.1 研究背景及意义

1.1.1 低轨通信卫星星座系统概述

十一五期间论证提出的低轨通信星座系统的基本组成如图 1.1-1 所示，包括空间、运控和应用^[2]三个部分。空间段采用（24,3,1）的 Walker 轨道方案实现对全球中低纬度的连续覆盖，并且可扩展为（48,6,3）构型的极轨星座来实现全球覆盖。本文以（24,3,1）构型的星座为研究对象，即卫星总数为 24、轨道面数为 3 且相邻轨道面邻近卫星之间的相位因子为 1，轨道高度取 1450km，轨道周期约 114 分钟。应用系统是指各类用户终端，包括手持机、车(机、船)载站、武器平台终端、航天载体终端、便携站和固定站等；运控系统负责对空间系统和应用系统的控制，包括网络操作控制分系统、应用管理分系统、安全保密分系统、信关站、遥测遥控站(TT&C)以及连接它们的运行支持网络(OSN)等。

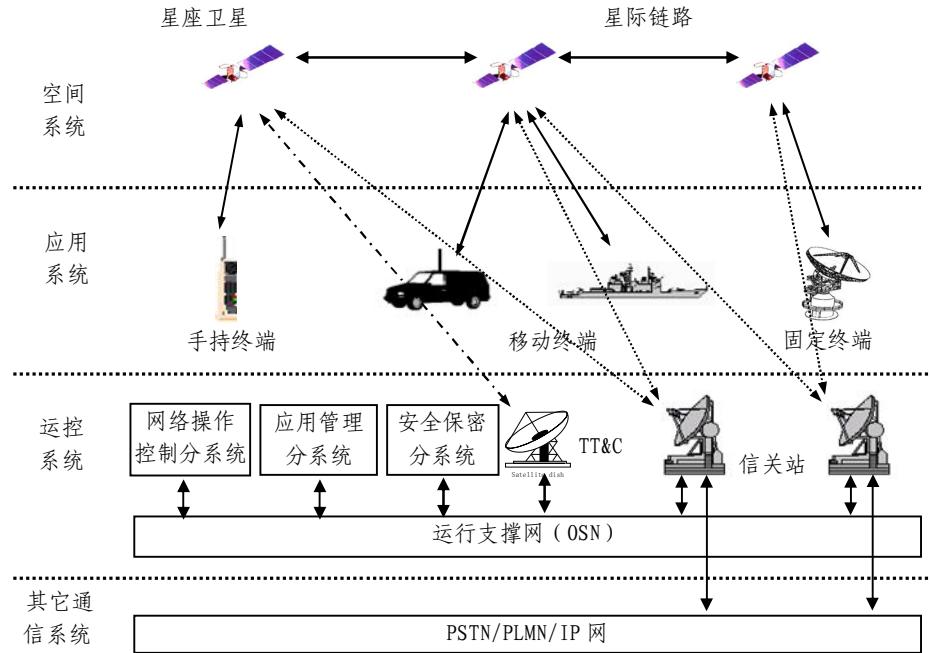


图 1.1-1 星座卫星通信系统的组成

低轨通信星座系统支持的业务种类繁多，主要包括语音、数据、图像等，其中语音业务速率为 2.4~9.6 kbps，数据和图像 64~384 kbps，另外还可能包括高速数据业务以及未来可能接入的 internet 业务。

星座卫星之间通过星间链路构成通信网络，为用户提供移动服务。与地面移动通信网络相比较，星座卫星相当于地面基站的角色，当然，卫星网络拓扑的时变性是不同于地面网络的特征。

1.1.2 星座网络自主管理的需求

目前卫星的测控与管理均是由地面系统完成。传统的模式是采用单星测控，即当卫星过境国内地面站时，通过测控信道与地面站交互遥控指令和遥测参数。然而随着星座系统的增加以及组成星座的卫星数目的不断增加，地面控制的成本和复杂程度将越来越高。另外，针对我国的国情，境外布站的难度较大，因此卫星通常需要运行至可见境内地面站时才能传输测控信息，时延较大。有研究者提出，通过星间链路传输测控信息以达到缩短测控时延的目的，但这无疑增加了星间链路的传输负担，加重了网络负载。

正是认识到以地面测控为主的传统模式在运行和管理中存在的缺陷，各国都在积极发展航天器的自主运行技术，尤其是星座系统的自主运行能够大大降低星座运行成本、减小系统风险，是一种必然的发展趋势。实现星座自主运行涉及诸多问题，其中，自主管理是星座自主运行需要解决的关键问题之一。

星座系统不依赖于地面站而进行自主管理可以大大减轻地面测控负担、降低星座运行成本、提高卫星的生存能力。对于国内来说，发展星座系统自主管理能力不仅可

以减少测控站数量，解决境外布站难题，同时还能提高系统战时的抗摧毁性，具有重要的军事意义。

星座系统的自主管理可以是一段时间内的自主，在条件允许或者需要的时候可以将管理的结果和状态下传给地面站，降低地面站的管理成本，同时克服地面站管理模式信息传输时延大的缺点；也可以是一部分管理功能的星上自主操控，与地面站共同完成整个系统的管理，有效减轻地面管理的负担。其自主管理的模式可以灵活多变，以适应不同的应用需求。

1.1.3 问题的提出

对于星座卫星通信系统来说，主要业务为语音、数据等，由星座卫星网络、地面网络与地面站共同为用户提供服务。当前针对星座网络的自主管理技术的研究还不是很多，因此当星座卫星系统不依赖于地面站和地面网络自主运行时，星上的自主管理要满足星上自主监测、自主决策和自主控制的需求，尚存在很多有待解决的问题。本文从星座网络自主运行最基本的管理需求出发，对亟待解决的几个基本问题进行了分析，如图 1.1-2 所示：

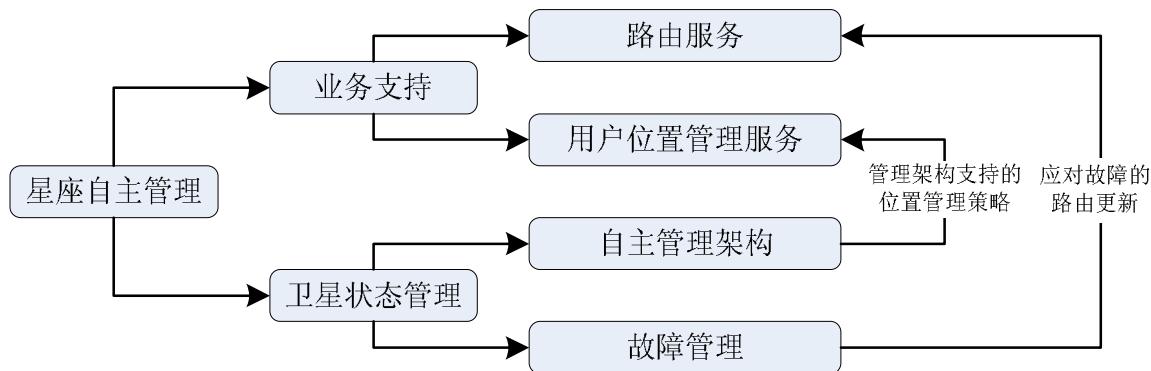


图 1.1-2 星座网络自主管理基本需求

星座自主管理需要具备的两个基本功能为：管理卫星状态和支持业务运行。从卫星状态管理的需求出发，首先必须具备合适的自主管理架构，以提高管理效率；其次，必须对卫星网络进行实时的故障监控，及时地发现网络中存在的故障以便对故障隔离恢复从而不影响正常的业务通信。从业务运行支持的需求出发，星座网络需要提供支持移动通信的用户位置管理服务和路由服务，其中，用户位置管理服务要求星座网络存储用户的位置更新消息、用户的呼叫信息、提供位置查询、寻呼用户的服务，这些都离不开自主管理架构的支持，而路由服务则必须能够应对网络故障，才能保证星座自主运行模式下用户的 QOS 要求。

从上述分析可知，这几部分内容是相辅相成、承上启下、有机结合的，下面分别对它们存在的问题进行详细论述。

(1) 管理体系结构问题

体系结构是网络管理的基础，不同的管理架构带来的管理效率也不同。传统的集

中式网络管理^[5]采用管理者-代理模式，网络负载严重不平衡，不适用于带宽受限的网络，同时管理数据传输的时延太大，也不适合实时性要求较高的移动通信星座系统。采用分布式管理^[6]可以有效地提高负载和时延性能，分布式管理多用于大规模网络场合或动态网络场合如 ad hoc 网络等，目前多采用分层次的网络管理结构，将管理应用分派给各级代理完成，从而减轻中央管理者的负担，这种结构比较适合多层次卫星网络。对于本文所研究的单层卫星网络，对等式管理结构相对来说更加适合自主管理的需求，但是卫星网络拓扑的时变性会引起管理结构的不稳定。合理的划分卫星网络管理区域，使得管理结构稳定可靠并能减少管理开销是研究自主管理体系结构需要关注的问题。

(2) 用户位置管理问题

低轨通信卫星星座系统的主要业务是为用户提供端到端的话音通信服务。一次成功的话音通信过程包括用户的接入、寻呼、通话接续、挂断等等操作。其中，对用户的寻呼依赖于对用户位置的管理。通常，用户位置管理由地面系统完成，当开始一段话音通信之前，首先要根据源用户呼叫的目的用户 ID 在地面系统进行查询，地面系统管理着所有用户的当前位置、卫星轨道运行情况、用户终端被卫星覆盖的情况等，根据查询的结果可以确定话音通信的目的卫星。在失去地面支持时，用户的位置管理由卫星自主完成。卫星的高速移动使得用户所处的卫星覆盖区不断变化，这会引起用户位置小区切换的判别模糊问题，同时造成频繁的位置更新，使得位置管理变得困难。另外卫星的移动性还造就了用户位置管理的分布性特点，怎样进行全网统一的管理也是必须要进行研究的问题。

(3) 故障诊断问题

在失去地面支持的情况下，准确并及时的发现卫星故障是星座自主管理的一个重要任务。完整的故障管理包括故障的检测、诊断、恢复等过程，传统的非星上自主的做法是由地面接收卫星的遥测数据，利用地面强大的计算资源和存储资源根据设计的专家系统对卫星某个分系统进行地面诊断。前述章节提到过，若采用单星测控，诊断时延过大，若利用星间链路测控，则会占用大量的网络资源。将故障诊断过程搬到星上自主进行，则面临星上资源受限、存在故障时的路由连接失效等问题。因此，设计星上自主故障诊断策略时，需考虑通信开销、路由失效、诊断复杂度等因素。

(4) 路由更新问题

通常，为了降低星上处理的复杂度，路由计算功能由地面站完成，根据静态拓扑结果生成合理的路由表，上传至卫星。当拓扑异常或发生更新变化时，再由地面上传新的路由表供星上查询使用。然而，当星座系统想要高效自主运行时，路由更新的任务就必须由星上承担。尤其是当卫星网络拓扑发生异常情况时，要求星座及时进行路由的更新，从而避免造成大量星间数据丢失。当这些功能由星上自主完成时，不得不

考虑由此带来的星上计算量和存储量增加的问题，另外完全照搬地面计算路由的过程，则会与系统设计的初衷相背驰，是不可行的。因此必须寻找合理的解决途径以满足系统需求。

综上，在不受地面站干预的情况下，星座要自主地维持整个网络持续高效并稳定可靠地运行，需要自成管理体系，完成用户位置的自主管理、卫星的自主决策，通过对网络的实时监控，及时发现网络故障，应对故障影响，保证关键服务。

网络性能的优劣直接影响用户的服务质量。因此，本文着眼于通信星座的网络性能，对星座系统进行自主管理，主要包括位置管理、故障管理、路由管理等等。

1.2 自主管理体系结构

星座卫星网络的分布范围广、拓扑动态变化等特性决定了其网络管理无法由单一的管理系统承担。因此可以选择网络中的若干卫星节点作为管理者，每个管理者负责管理网络的一部分（即管理簇^[6]），同时管理者只存储其所辖管理簇的 MIB^[8]，对整个网络的管理（例如对不同管理簇之间星间链路的管理）由不同管理者共同完成，即管理者通过协作^[34]的方式对网络进行管理，这就是星座卫星网络自主管理的基本思想。这种管理结构我们称之为对等式管理结构^[6]，如图 1.2-1 所示。

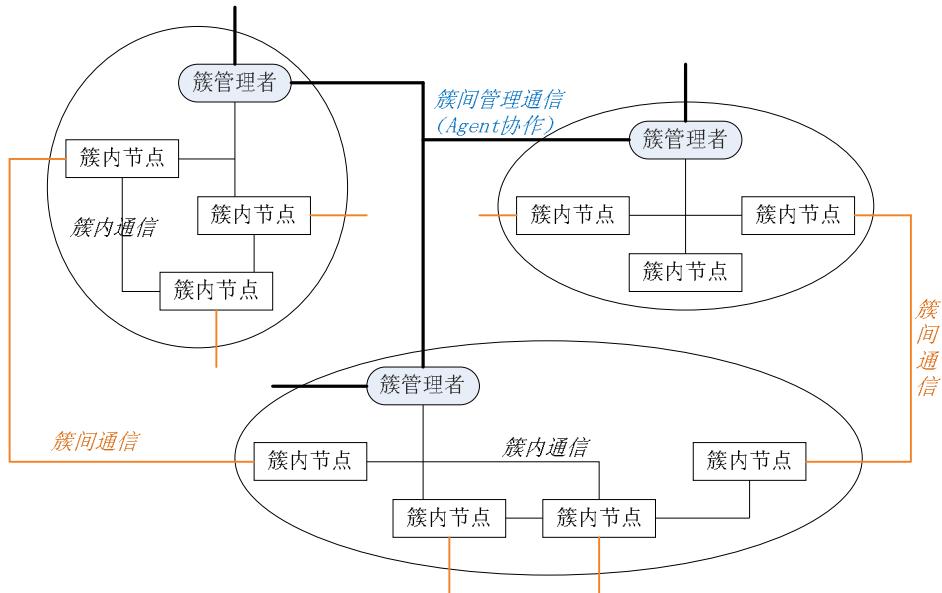


图 1.2-1 星座自主管理体系结构

在整个网络的生命周期中或一段时间内，网络管理的任务由网络管理系统独立承担，不需要人参与到网络管理的活动中。星座卫星网络自主管理具有以下几个特点：

- ✓ 星座网络自主管理系统是一个分布式的网络管理系统，整个网络中存在若干个管理者，而不是像集中式网络管理系统那样只有一个网管中心；
- ✓ 星座网络自主管理系统分为若干个簇，每个簇包含一个管理者，簇与簇之间

是对等的，网络中最高管理者即为簇管理者；

- ✓ 各管理者可以通过协作来完成单一管理者不能独立完成的管理任务。
- ✓ 簇管理者对本簇内的节点进行管理，交互管理信息，簇与簇之间的管理信息交换需通过簇管理者相互之间的通信来完成；
- ✓ 簇管理者之间的通信链路是一种逻辑关系上的链路，并不表示它们之间一定存在星间链路；
- ✓ 节点之间的业务通信可通过不同簇内的成员之间的星间链路进行；
- ✓ 星座网络自主管理系统实施管理的活动中不需要人的参与，整个管理过程所涉及的操作由系统自主完成；

星座网络自主管理采用分布式管理，根据一定的原则将整个网络划分为不同的管理簇，每个卫星在某个时刻只属于一个管理簇。相对其他网络管理技术来说，卫星网络采用自主管理具有以下几点优势：

- ✓ 星上进行分布式自主管理，减少了网管信息的传输量和传输时延，从而提高网管效率；
- ✓ 星上可以实时地进行故障诊断和故障处理，减少了故障处理时延；
- ✓ 网络管理不受卫星覆盖范围限制的影响，实时性强。

本论文对自主管理关键技术的研究均是基于这样的自主管理架构下开展的。

1.3 论文主要研究内容

低轨通信卫星星座网络的自主管理是未来发展趋势，卫星网络独有的移动速度快、时延大、拓扑周期变化、网络资源受限等特点，使得现有的地面无线网络的管理架构不适用于卫星网络。同时，星上设备远远高于地面设备的可靠性要求以及航天级器件远远低于商业级器件的处理能力现状，使得目前的自主组网技术、基于庞大数据库支持的故障检测与诊断技术、复杂度高的路由算法等技术很难在星上实现。因此，这些方面都要针对低轨通信星座网络的特点进行设计。论文主要研究了低轨通信星座网络在自主管理时需要解决的一些关键技术，下面简单介绍一下本文在以下几个方面所做的研究工作。

1.3.1 自主管理分簇算法

根据低轨通信星座网络拓扑特征以及承载业务的特点，确定分布式管理模式。在此基础上，提出利用成簇方式建立星座自主管理的体系结构。

本文分析了低轨通信卫星网络拓扑的特点，包括建链方案、建链时长以及静态拓扑划分方式等，提出了适合于低轨通信星座网络的管理簇划分算法。衡量一个分簇算法的优劣主要有以下几个标准：簇结构的稳定性、簇的数量、负载均衡度及簇维护开销等。因此，本文在设计低轨通信星座网络的管理分簇算法时充分考虑了这些因素，

利用了同轨道卫星节点之间链路的稳定长久性，提出了管理簇划分的约束条件——同轨面的一组相邻卫星节点作为一个簇结构的固定组成。基于该约束条件的分簇算法能有效降低簇维护成本、缩短管理时延、减小管理开销、均衡负载。本文采用图论对卫星网络进行了建模，并引入管理约束度这个重要参数，论述了簇管理者的选择策略以及管理簇的划分原则。对算法的具体实现过程即簇的建立和保持过程进行了详细描述，并通过与传统分簇算法的仿真比较结果说明本算法用于自主管理体系结构时的有效性。

1.3.2 星上位置管理技术

针对本文低轨通信星座系统为个人移动通信服务的业务特点，提出网络在自主管理时需要解决的问题：如何摆脱用户位置管理对地面站的依赖性。本文在基于无地面站支持的前提下对星上位置管理技术进行了研究。

本文着重于网络侧的位置管理，目标是通过星上自主位置管理，查找到用户当前时刻位于哪颗星覆盖区。星上位置管理面临的问题包括卫星处理能力有限、HLR 与 VLR 相对位置不断变化、卫星相对于用户的高速运动使得位置更新次数频繁等等。针对这些问题，利用本文提出的管理簇结构，提出多 HLR 的分布式位置管理架构，研究了多 HLR 协作的位置管理流程。研究了簇间更新和簇内更新策略，提出 VLR 的簇归属不变化时不发起位置更新，有效减少了由于卫星运动造成的用户频繁位置更新所带来的网络级更新开销。同时利用移动 agent 与指针转发相结合的簇内更新策略，解决由于卫星运动造成的指针失效问题。本文还论述了整网数据库维护的问题，研究了过期位置信息的处理策略、用户位置寻呼策略等。对策略的代价进行了理论分析，并与传统策略进行了仿真分析比较，结果表明本文策略在寻呼时延、位置管理开销等性能上有很好的表现。

1.3.3 自主故障检测与识别技术

本文主要针对网络故障的自主检测和识别方法进行研究，以系统级故障诊断理论为基础，结合低轨星座系统网络自主运行的需求，提出一种适合于星上处理的分布式自主网络故障识别算法（DSFD），使得每个节点都可独立进行自识别，而无需获得整网节点状态，这种算法能够大大降低网络故障检测开销，同时缩短诊断时延，并且降低星上计算量，提高可靠性。

论文采用图论对卫星网络进行建模，采用 SLD 测试流程，在建模基础上研究了 M-概率的分布式测试模型。通过对测试任务的设计，使得测试图符合小概率忽略的诊断规则。由于小概率忽略的规则会导致诊断结果部分误判，为了解决该问题，论文研究了测试图预修正、二次判决后处理等风险消减措施，达到正确识别故障的目的。论文详细描述了故障诊断流程，并对性能与概率参数的关系进行了理论分析，对算法的有效性进行了证明。仿真实验和分析表明该算法能够有效地提高故障诊断的正确率。

和完全率，同时在诊断开销和诊断时延方面也较同类算法性能有明显优势。

1.3.4 路由算法及自主路由更新技术

传统的卫星网络路由算法是采用最短链路时延和静态拓扑快照相结合的方式，不具有自主性，对卫星网络实时链路状态没有很好的适应性，同时无法应对链路发生故障时的路由更新。针对这两种情况，本文对基于静态拓扑序列的路由算法进行了改进，考虑到时延和切换对路由代价的综合影响，研究了权值路由策略。在保持路由表不变化的同时，数据交换时结合节点当前的链路状态，采用简单的星上自主判决策略，使得这种基于静态拓扑序列的路由能够在一定程度上平衡流量、缓解拥塞。论文通过仿真对算法在时延抖动、流量平衡、拥塞预防等方面的性能进行了分析。

本文还对基于移动 agent 的路由策略进行了研究，提出了利用移动 agent 进行局部动态更新的路由算法，来解决发生链路故障后的路由表更新问题。论文对移动 agent 的管理流程、迁移策略进行了研究，并依据移动 agent 的迁移路径比较结果给出权值路由更新的计算方法。论文对基于移动 agent 的系统进行了设计，使得算法在实现上更具有实际意义。该算法在能够应对网络拓扑异常变化的同时，不增加星上的计算量，减小路由更新的扩散范围。论文对算法的网络开销、更新时延以及路由更新后交换性能进行了仿真分析。

1.4 论文结构安排

本论文在背景预研所确定的24/3/1 构型低轨卫星网络前提下，研究了网络自主管理中的几个关键技术。主要思路是解决自主管理中存在的问题，同时要兼顾卫星网络资源受限的特点，满足语音业务时延要求，使算法具备可实现性。

论文的结构安排如下：

第一章主要是对研究的背景和意义进行了介绍，分析了星座网络自主管理的需求，引出本文所要研究的问题。并对自主管理的体系结构进行了概括性总述，后续的研究在此基础上开展。

第二章从自主管理体系结构出发，提出采用分簇的方式进行分布式管理。首先介绍了分簇算法的国内外研究现状，分析了它们对星座网络的适用性。详细分析了星座网络的拓扑特性，在此基础上提出了基于约束条件的分布式分簇算法，并对网络建模、簇划分原则、簇形成过程进行了详细描述，给出了伪代码实现过程及仿真分析结果。

第三章首先对位置管理技术进行了概述，比较了地面移动系统位置管理与低轨星座系统位置管理的不同之处，进而提出了星座网络自主位置管理的解决方案，提出了移动agent与指针转发相结合的簇内位置更新策略，并对其中采取的位置更新流程、数据库维护策略进行了详述，通过数学方法分析了该策略的位置管理代价，并对其性能进行了仿真验证。

第四章首先介绍了系统级故障诊断理论及其国内外研究现状，说明其对星座网络的适用性。在此基础上提出星上自主网络故障识别的算法，对测试任务进行了设计，提出M-概率的测试模型，理论分析了忽略小概率事件对诊断结果的影响，进而提出消除影响的措施。最后与其它算法进行了仿真分析比较。

第五章首先对网络路由技术进行了概述，说明移动agent用于路由算法的优势。接着对网络静态拓扑快照进行了描述，推导并仿真了切换对时延抖动的影响，说明传统算法的缺陷，进而在静态拓扑序列路由基础上提出了改进的考虑时延和切换因素的权值路由算法。并以此算法为标本，提出移动agent局部路由更新算法来解决星座网络自主管理时发生链路故障后的路由更新问题。最后对算法的性能进行了仿真分析。

第六章总结全文，并对今后的研究工作方向给出了建议。

第二章 星座网络自主管理簇架构的研究

2.1 引言

对星座进行高效的自主管理在系统自主运行中具有至关重要的意义。卫星网络具有资源受限、星间传输时延大、网络拓扑周期时变等特点，使得星座系统自主管理的难度增加。传统的集中式管理在星座网络管理中存在管理时延大、负载不均衡、可靠性低的缺点，无法满足当前星座系统自主管理的需求。分布式管理模式的自主性、自适应性、响应快速性则能很好地解决集中式管理模式所存在的问题。在分布式管理模式下，利用成簇方式建立星座自主管理的体系结构，对管理的自主性、高效性、实时性等均有好处。

低轨通信星座系统具备不同于一般移动网络的特点：1) 星座内节点之间的拓扑关系是有规律的；2) 同轨道节点之间的连接关系是稳定不变的；3) 卫星之间的连通性不是简单地由可见性和发射功率来决定，其拓扑关系、通信距离很大程度上依赖于星间链路及天线的设计；4) 星间链路带宽资源受限；5) 业务数据实时性要求高。

本文针对低轨通信星座系统的特点，研究了自主管理模式下的管理星簇划分算法——基于约束条件的分布式分簇算法（CDCA），该算法具有以下优点：1) 充分利用了同轨道链路的稳定特性，约束同轨道的相邻节点组成簇的核心成员，通过将卫星层次化分簇的方法，提高了管理星簇的稳定性；2) 除非节点失效，否则不改变簇管理者及簇内核心成员，从而减小了拓扑变化时簇维护造成的包开销；3) 以轨道面的数量作为划分簇时的依据，使得网络内的管理簇数目稳定，网络负载更加均衡；

2.2 分簇算法国内外研究现状

分簇的概念最初是基于 adhoc 网络^[9]而提出的，是为了解决由于 adhoc 网络的动态性以及变化的不规律等特点带来的网络路由问题。后来随着 MANET 的发展，分簇算法的研究获得了迅速的发展^{[17][19][21][22][23][24]}。簇（cluster）对于分布式网络中移动性管理的处理、路由机制的优化、更好的带宽利用、拓扑信息的综合等方面均有明显的优势^[17]。由于卫星的高速移动性，卫星网络拓扑的动态性、卫星网络的分布性、卫星网络管理的重要性，使得分簇管理在卫星网络中的应用研究得到发展^[31]。

早期的分簇算法有最小 ID 算法^{[10][11][12]}与最大连接度算法^{[13][14]}。最小 ID 算法选择 ID 最小的节点成为簇首节点，节点周期性地向其邻居节点(在其接收范围内的节点)广播 ID 值。每个节点比较自己与其直接邻居节点的 ID 值，若发现自己为 ID 值最小的节点，则自动成为簇首节点。最大连接度算法选择连接度最大的节点成为簇首，连接度指一个节点的直接邻居个数。这两种算法比较简单，但存在缺失公平性、吞吐量降低等缺点。后来出现了不少基于这两类算法的改进算法^{[15][16][17][18]}。

其中, 文献^[17]提出了应用于动态环境的 WCA 分簇算法, 是后来诸多权值算法的基础。该算法的主要思想是通过加权计算获得每个网络节点作为簇首的能力, 通过在全局范围内广播节点的权值来选择簇首, 根据是否能够在发射覆盖范围内接收到信号来决定节点的邻居关系, 同时簇结构需经多次迭代确定。

后来的研究大多都是基于该算法进行改进或应用^{[19][20][21][22][23][24]}, 其中文献^[19]提出经典 DWCA 算法是对 WCA 算法的改进, 不再全局范围广播自己的权重而只是在一跳范围内广播, 大大减小了网络开销, 同时使簇成员邀请二跳成员加入簇, 减少了簇的数量, 另外还对权重参数做了修改, 使其更适合于实际应用场景, 并对两者的性能做了比较。文献^[20]则从另一个方面对 WCA 算法进行了改进, 针对 WCA 算法中频繁簇首选举占用过多带宽的缺点, 提出了 WCA-L 算法, 对簇首选举过程启动的条件进行了约束, 在至少存在两个簇首为邻居的情况下才重启簇首选举, 从而降低簇保持过程的开销。文献^[24]在簇的维持阶段引入了移动性预测来帮助簇的管理, 可以有效降低系统中节点能量的消耗, 提高了数据传输效率和簇的稳定性;

上述各种类 WCA 算法主要应用于 adhoc 网络, 大多簇首选举时的关注点在于节点的能量损失和移动性等, 而在无线混合网络中, 无线资源的紧张越来越得到重视, 以提高无线资源的利用为目标的分簇算法成为 WMNs (wireless mesh networks) 领域研究的重点。文献^[25]对应用于混合网络的层次化分簇算法进行了优化, 并对权值进行本地计算, 限制 adhoc 网络到骨干网络之间的传输性形成有效的信息分发, 从而减少簇首选举中的通信代价。文献^[26]基于论述了簇的最优建立问题, 并分别讨论了簇与簇之间不相交和存在一定交叉覆盖的情况下算法的优化。

同 adhoc 网络类似, 在传感器网络中, 无线路由策略、数据传输效率、节点能量损耗问题也是关注的焦点, 因此该领域也是分簇算法蓬勃发展的沃土。文献^{[27][28][29]}针对传感器网络提出了改进的分簇算法。

分簇算法除了上述提及的优点, 还因为其层次化的结构, 在网络管理方面发挥着作用。文献^[30]将分簇算法应用到移动 adhoc 网络 (MANET) 的管理上, 提出了一个利用令牌机制与最小节点标识相结合, 适用于层次性管理结构的基于图的 MANET 簇生成算法, 以及对 MIB (management information base) 的建议性改进, 为 MANET 的网络管理提出了一种新方法。

文献^{[31][32][32][34]}是面向卫星网络管理的分簇算法研究。文献^[31]在文献^[30]的基础上提出了一种适用于卫星综合网络的星簇生成算法, 将令牌机制与权值相结合, 提高了网络管理的灵活性, 并充分考虑了星间存在的单向链路和空间星簇生成的自主性。文献^[32]提出了星地网络管理的结构, 通过对卫星网络分簇管理能够有效地缩短和减少地面管理站与卫星节点代理之间的通信延迟和开销。文献^[33]针对对卫星综合信息网, 提出了基于延迟的动态管理域划分算法, 降低了卫星综合信息网特有的高延迟特性的影

响，并针对管理星簇的生成算法进行了验证。文献^[34]则针对单层卫星网络提出了基于链路时延的管理域划分算法 ADSSNMD，并在此基础上进行了改进，并对改进后的性能进行了仿真，证明了算法在保持管理域稳定性方面的有效性。

在实际卫星应用中，为了达到更好的星间通信效果，异轨链路节点之间的连通时刻、时长都是经过预先设计的，通常由卫星节点之间的相对运动速度、方位角、俯仰角、天线的波束宽度、天线的跟踪能力、天线安装角度等因素决定。并且，这种拓扑关系是有规律性的，并随时间周期变化。同时，由于节点之间的通信距离、管理开销所造成的功耗在卫星网络中相对于通信业务的甚至卫星平台的功耗来说几乎可以忽略，因此在设计星座网络分簇算法时，应充分利用星座网络本身固有的特点，修正簇首能力指数的计算方法，使提出的分簇算法更加适用于星座网络的自主管理。

2.3 星座网络拓扑分析

管理架构的设计需要考虑星间链路的时延、延迟变化率以及路由策略等因素，而这些因素与卫星网络拓扑具有密切关系，因此研究管理簇的划分需要确定星座系统的拓扑结构。

2.3.1 低轨卫星星座网络模型

低轨卫星通信系统采用 Walker 星座，星间链路为 Ka 频段信号，每个卫星节点通过四条星间链路与相邻四个节点进行通信，其中，包括 2 条同轨面链路和 2 条异轨面链路。图 2.3-1 是 (24,3,1) 的 Walker 星座示意图。

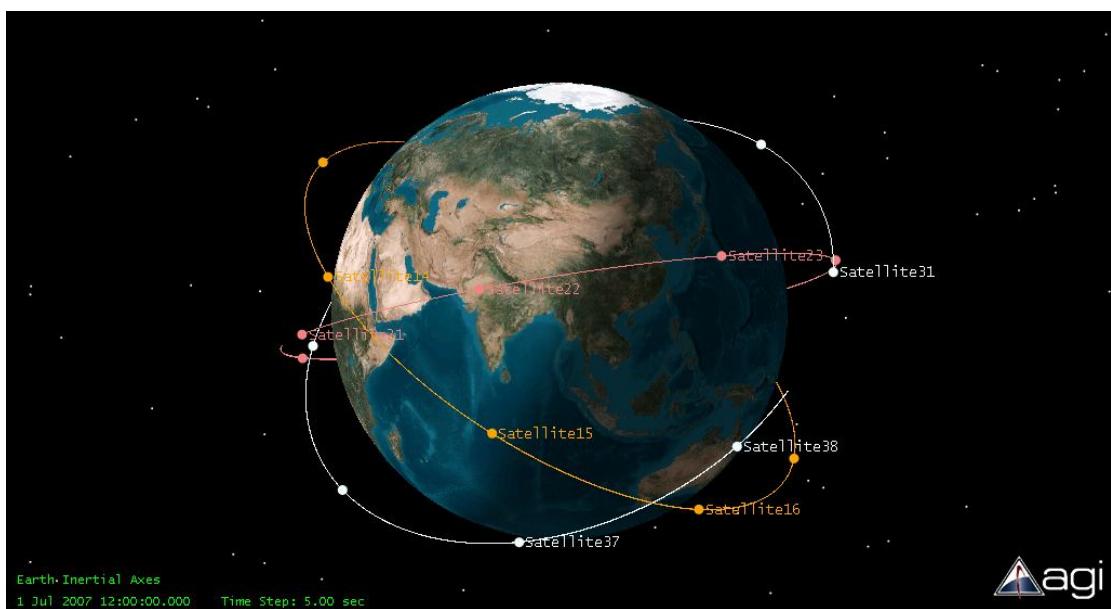


图 2.3-1 (24,3,1) 的 Walker 星座示意图

该星座由 24 颗卫星组成，均匀分布在 3 个轨道面上，相邻轨道相位因子为 1，轨道倾角为 30 度。相邻轨道的相邻卫星的相位差为：

$$360^\circ \times F/N = 15^\circ$$

(式 2.3-1)

此星座主要覆盖中低纬度地区，其相位关系直接影响到各个卫星之间的位置关系，以及连通关系。下面表 2.3-1 中给出了 24 星 Walker 星座的相位关系。我们对星座中轨道面及卫星进行编号，三个轨道面分别称为轨道面 1、轨道面 2 和轨道面 3，同一轨道面里的卫星用 1~8 的子编号区分，例如，Sat27 代指轨道面 2 中的 7 号卫星。

表 2.3-1 24 星各星之间相位关系

卫星编号	纬度幅角 (°)	卫星编号	纬度幅角 (°)	卫星编号	纬度幅角 (°)
Sat11	0	Sat21	15	Sat31	30
Sat12	45	Sat22	60	Sat32	75
Sat13	90	Sat23	105	Sat33	120
Sat14	135	Sat24	150	Sat34	165
Sat15	180	Sat25	195	Sat35	210
Sat16	225	Sat26	240	Sat36	255
Sat17	270	Sat27	285	Sat37	300
Sat18	325	Sat28	330	Sat38	345

相对于极地轨道星座，倾斜轨道星座的优点是避免了相对运行轨道之间的缝。在卫星网络中存在两种类型的星间链路 (inter-satellite links ISLs)：同轨星间链路 (intra-plane ISLs) 和异轨星间链路 (inter-plane ISLs)。每颗卫星可与同轨道前后卫星建立两条稳固的同轨星间链路，与相邻轨道面的可见卫星最多可达 6 颗，用于接入异轨星间链路的天线只有两幅，可建立两条异轨星间链路。考虑星座网络的拓扑结构时，同轨邻星的链路非常稳定，而异轨的邻星链路是动态变化的根据两星之间距离、方位角、俯仰角的变化率以及天线的跟踪扫描特征确定异轨星间链路的建立。本文对于采用何种方式的异轨建链方案能够达到最好的网络性能不作研究，直接采用在 STK 中分析得出的方案。

2.3.2 星间网络拓扑分析

同一轨道面卫星位置相对固定，因此同轨星间链路是永久性链路。而异轨面的卫星由于相对位置发生变化，不仅链路的长度会时刻变化，且会关闭。例如铱星系统，当卫星进入极地地区时会暂时关闭异轨星间链路；相反方向运动轨道面的卫星由于相对运动速度太快，也会关闭链路。另外，由于卫星的不断运动导致两个卫星从可见变为不可见时，或者由于卫星姿态的变化使得两颗卫星从可通信变为不可通信时也会使得异轨链路断开。因此异轨链路的变化会引起卫星网络拓扑结构发生变化，且这种变化是由卫星轨道模型决定的，因此，在设计管理簇生成算法时，首先需要分析卫星网

络的拓扑特征。

前面已经提出 24/3/1 低轨卫星可建立两条同轨星间链路和两条异轨星间链路，在 STK 中分析各种建立异轨链路的方案，根据卫星的运行轨道以及天线的跟踪、扫描采用如表 2.3-2 所示的建链方案。

表 2.3-2 异轨卫星建立链路方案

轨道面 1——轨道面 2	轨道面 1——轨道面 3	轨道面 2——轨道面 3
Sat 11——Sat 27	Sat 11——Sat 32	Sat 21——Sat 37
Sat 12——Sat 28	Sat 12——Sat 33	Sat 22——Sat 38
Sat 13——Sat 21	Sat 13——Sat 34	Sat 23——Sat 31
Sat 14——Sat 22	Sat 14——Sat 35	Sat 24——Sat 32
Sat 15——Sat 23	Sat 15——Sat 36	Sat 25——Sat 33
Sat 16——Sat 24	Sat 16——Sat 37	Sat 26——Sat 34
Sat 17——Sat 25	Sat 17——Sat 38	Sat 27——Sat 35
Sat 18——Sat 26	Sat 18——Sat 31	Sat 28——Sat 36

考虑到距离变化、方位角变化速率、俯仰角变化率、天线跟踪等诸多因素对异轨链路建立的影响，低轨通信星座通常采用接续式的异轨星间链路^[35]，即在不同的轨道之间均保持至少有一条轨间链路，而且轨间链路具有接续性质，即前一条链路中断之前，后续的异轨星间链路已经建立，依次类推。图 2.3-2 表示高度为 1450km 的(24,3,1) walker 星座在一个轨道周期内，第 1 轨道和第 2 轨道的异轨星间链路的通断持续情况。

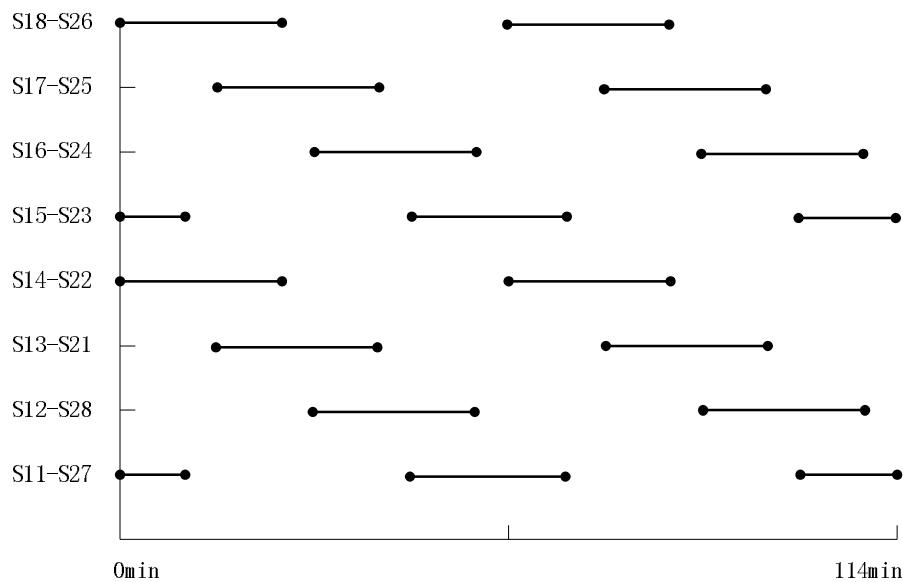


图 2.3-2 星座网络异轨链路建立情况示意图

各个卫星节点对之间的具体建链时间如表 2.3-3 所示。

表 2.3-3 一个轨道周期内各异轨星间链路建立时间段

1~ 2	建链时间	1~ 3	建链时间	2~ 3	建链时间
S11-S27	00:25-00:57	S11-S32	00:01-00:33	S21-S37	00:20-00:52
	01:22-01:54		00:58-01:30		01:17-01:49
S12-S28	00:10-00:42	S12-S33	00:15-00:47	S22-S38	00:06-00:38
	01:07-01:39		01:12-01:44		01:03-01:35
S13-S21	00:00-00:28	S13-S34	00:00-00:04	S23-S31	00:00-00:23
	00:53-01:25		00:29-01:01		00:48-01:20
	01:50-01:54		01:26-01:54		01:45-01:54
S14-S22	00:00-00:14	S14-S35	00:00-00:18	S24-S32	00:00-00:09
	00:39-01:11		00:43-01:15		00:34-01:06
	01:36-01:54		01:40-01:54		01:31-01:54
S15-S23	00:25-00:57	S15-S36	00:01-00:33	S25-S33	00:20-00:52
	01:22-01:54		00:58-01:30		01:17-01:49
S16-S24	00:10-00:42	S16-S37	00:15-00:47	S26-S34	00:06-00:38
	01:07-01:39		01:12-01:44		01:03-01:35
S17-S25	00:00-00:28	S17-S38	00:00-00:04	S27-S35	00:00-00:23
	00:53-01:25		00:29-01:01		00:48-01:20
	01:50-01:54		01:26-01:54		01:45-01:54
S18-S26	00:00-00:14	S18-S31	00:00-00:18	S28-S36	00:00-00:09
	00:39-01:11-		00:43-01:15		00:34-01:06
	01:36-01:54		01:40-01:54		01:31-01:54

根据图 2.3-2 所示的异轨链路通断情况对该星座系统的拓扑连接关系进行分析可知，将任意一条异轨链路连接发生变化称为一个拓扑更新，则一个轨道周期内共有 96 次拓扑更新。由图中还可以看出，该星座网络中一对异轨节点在一个轨道周期内存在两个时间段可以建立链路，大约为 32min。由于 walker 星座的轨道对称性，S18 和 S26 这一节点对与 S14 和 S22 节点对的链路建立时间段相同，同样的情况也在其它节点对中发生，因此可将一个轨道周期内的拓扑更新次数简化为 48 次。

2.4 基于约束条件的分布式分簇算法

2.4.1 约束条件

星座网络内所有节点都是对等的，都肩负着接入、交换、路由等功能，只有性能上的差异，没有功能上的不同。对这种平面式结构采用适当的分簇算法构造分布式的拓扑，相互邻近的一组节点构成一个簇，可以提高管理的实时性，降低并平衡整网的管理负载，并通过分布式协作完成网络的自主管理。

衡量一个分簇算法的优劣主要有以下几个标准：簇结构的稳定性、簇的数量、负载均衡度及簇维护开销等。因此，设计低轨通信星座网络的管理分簇算法时应充分考虑这几个问题，结合该网络的特点提出有针对性的解决方案。

低轨通信星座网络具备不同于一般移动网络的特点：1) 星座内节点之间的拓扑

关系是有规律的；2) 同轨道节点之间的连接关系是稳定不变的；3) 卫星之间的连通性不是简单地由可见性和发射功率来决定，其拓扑关系、通信距离很大程度上依赖于星间链路及天线的设计；4) 星间链路带宽资源受限；5) 业务数据实时性要求高。

本文针对低轨通信星座网络的特点，提出了管理簇划分的约束条件——同轨面的一组相邻卫星节点作为一个簇结构的固定组成，以提高管理簇的稳定性。在此基础上，我们来分析簇数量、簇大小的选择考虑因素。

文献^[17]提出采用负载均衡因子 LBF 来表示簇结构的负载均衡能力：

$$LBF = \frac{n_c}{\sum_i (x_i - \mu)^2} \quad (\text{式 2.4-1})$$

其中， n_c 表示簇的数量， x_i 表示簇 i 的势(即簇 i 包含的节点数量)， $\mu = (N - n_c)/n_c$ ，表示簇管理者管理的成员数量的平均值(N 表示网络中节点的总数量)。 LBF 的值越高，系统的负载均衡能力越好。

由 (式 2.4-1) 说明，系统若需好的负载分布，应使簇的大小尽量平均。

另一方面，簇的大小具体值选择与管理时延的需求有关。为了使管理者能以最快的速度收集所管簇内的网管信息且网管操作指令能尽快地发布给被管卫星节点，被管卫星与簇管理者之间的通信时延应尽可能短。根据本文的约束条件，可采用以簇管理者为中心，其同轨的 x 跳邻居节点为核心节点的簇组成方式。由此，其管理者与被管节点之间最长的时延为：

$$T = x \cdot \frac{d_{\text{intra}}}{c} + \frac{d_{\text{inter}}(t)}{c} \quad (\text{式 2.4-2})$$

其中， c 表示光速， d_{intra} 表示同轨相邻节点之间的距离， $d_{\text{inter}}(t)$ 表示异轨相邻节点之间的距离，是随位置不同而变化的，计算如下：

$$d_{\text{intra}} = \sqrt{2}R \sqrt{1 - \cos\left(\frac{360}{M}\right)} \quad (\text{式 2.4-3})$$

$$d_{\text{inter}}(t) = \sqrt{2}R \sqrt{1 - \cos\left(\frac{360}{2 \times N}\right)} \times \cos(lat(t)) \quad (\text{式 2.4-4})$$

上式中， N 代表轨道面个数， M 代表每个轨道面内卫星个数， $R = r + h$ ， r 为地球半径， h 为轨道高度， $lat(t)$ 表示 t 时刻的纬度。

根据系统对管理时延的需求，可对管理者与被管节点之间的最长时延进行约束，使

$$T_{\max} < \delta \quad (\text{式 2.4-5})$$

其中， δ 为管理时延门限值。

根据以上分析，基于本文提出的约束条件下的簇个数选择可遵循与轨道面成正比

的关系，这样可以保证网络内簇的个数比较稳定，同时由于约束条件的存在会使得各簇的大小比较平均。簇内包含的节点个数选择主要依赖于 x 值的选取，可根据系统管理时延的要求进行选取，同时 x 值的大小决定了簇大小的基线。

对于本文的研究对象——高度 1450km 的 (24,3,1) walker 星座来说，可选取每个轨道面包含 1 个簇管理者， x 值为 2 来构成簇。

2.4.2 管理簇的划分

2.4.2.1 簇管理者选择策略

(1) 管理指数

低轨通信星座网络中的节点众多，但并不是每个节点都有能力成为簇管理者，完成相应的管理任务。管理者的选择策略与网络的拓扑结构、系统的管理需求密切相关。

根据前述章节的网络拓扑分析可知，低轨通信星座系统采用接力的方式进行星间链路的建链，因此，一个节点总是和相邻轨道面上的固定节点建立通信链路，当然，两条异轨链路并不总是同时建立。这样就存在节点在某些时刻拥有 2 条星间链路，另一些时刻存在 3 条星间链路或 4 条星间链路的情况。一般地，在建立一个簇时，希望簇内节点与簇管理者之间的路由跳数尽可能少，因此，簇管理者能够同时建立越多的星间链路就显得尤为重要。

作为簇管理者，其处理的任务要比一般节点多，在选择时，尽可能选择计算能力较强的节点。同时，簇管理者需要比一般节点存储更多的管理信息，建立大容量的管理信息数据库，因此节点的存储能力也是选择的依据之一。

另外，建设星座系统时，通常不会一步到位，而是逐步进行的。也就是说，星座卫星的发射时间并不一致。那么，节点的寿命也是选择簇管理者时需要考虑的一个因素。

综上所述，这里引入管理指数来表征节点成为管理者的能力。设卫星节点 v_i 的管理指数为 W_i ，则有：

$$W_i = w_1 \cdot T_i + w_2 \cdot C_i + w_3 \cdot S_i + w_4 \cdot E_i \quad (\text{式 2.4-6})$$

其中 T_i 为节点在一个轨道周期内同时建立 4 条星间链路的总时长， C_i 为节点的计算能力， S_i 为节点的存储能力， E_i 为节点的寿命。 w_1 、 w_2 、 w_3 、 w_4 分别为上述各种因素的权值。

(2) 管理约束度

根据节点的管理指数计算结果作为簇管理者选择的依据，这点与经典的 DWCA 算法是一致的，然而不加约束条件的分布式分簇算法对簇首的选择策略并不适合星座网络的拓扑特点。这是因为，从星间网络拓扑的分析来看，可以建链的节点对总是固定的，并且是按照轨道卫星的排列顺序依次建链，不加约束条件的话，可能导致划分出

的簇的位置分布非常集中、数量非常多，不利于自主管理。

因此除了采用前述的管理指数对节点的管理能力进行表征，这里再引入一个参数——管理约束度 CP_i ，该参数表示节点被选择为簇管理者的优先级。

每个节点的管理约束度 CP_i 的初始值均为 0，在簇划分的过程中根据相邻轨道建簇的情况改变 CP_i 的值，每当一个节点加入某个簇后，应修改其管理约束度 CP_i 约束度为 (CP_i+1) 。选择簇管理者时总是在 CP_i 值最小的约束条件下进行，这样能够保证簇管理者的一跳邻居节点数量在当前条件下总是最大的，从而使平均管理时延较小。 CP_i 值的具体修改过程在下一小节中进行描述。

2.4.2.2 管理簇划分算法

根据前述对低轨通信星座网络特点的分析，针对约束条件，这里提出一种管理簇的结构，在这种结构中，将节点的身份定义为以下四种：游离节点、簇管理者、簇内核心成员、簇内一般成员。其中，游离节点表示该节点不属于任何簇，其余三种则分别代表了节点在簇内的不同地位。簇的结构表示为图 2.4-1 所示：

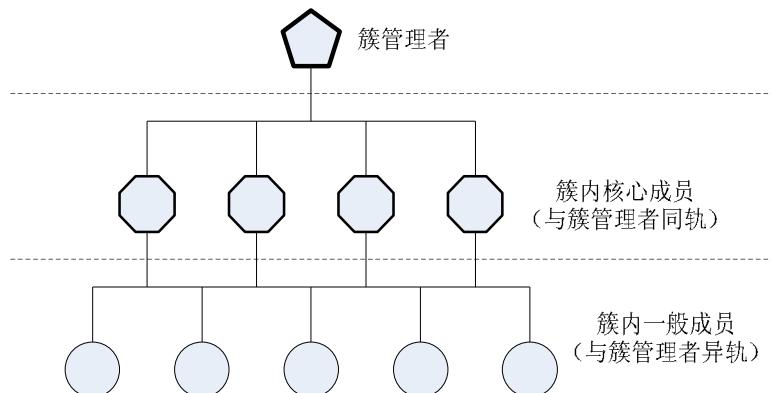


图 2.4-1 管理簇结构示意图

如图所示，与簇管理者链路距离较近的几个同轨道节点成为簇内核心成员，核心成员是簇内的稳定元素，它们与簇管理者之间的连接关系在正常情况下是恒定不变的。簇内一般成员是与簇管理者或簇内核心成员相连的节点，它们与簇的关系随网络拓扑的变化而变化，是簇内的不稳定元素。

在讨论簇的划分之前，先对算法中涉及的一些参数作以下定义。

首先对空间卫星网络建模。从网络的角度来看，空间卫星网络的核心元素包括卫星节点和星间链路，我们用图表示某时刻网络的拓扑结构。由于网络中星间链路是双向的，因此可用一个无向图 $G = \langle V, E \rangle$ 来表示空间卫星网络，其中非空节点集 V 表示网络中的卫星节点集合；连接节点的边集 E 表示在网络中存在的星间链路。

定义 2.1 在网络 $G = \langle V, E \rangle$ 中，给定节点对 $u, v \in V$ ， $\langle u, v \rangle$ 或 $\langle v, u \rangle$ 表示节点 u 和节点 v 之间的边。在空间卫星网络中， $\langle u, v \rangle$ 或 $\langle v, u \rangle$ 表示卫星节点 u 与卫星节点 v 之间存在的星间链路，这里的“存在”定义为无论在当前时刻是否连通，只要在一个轨道周期内存在连通时间的星间链路。通过对网络中存在的边赋值来表示星间

链路在某一时刻 t 的连通性, $E_t < u, v > = 0$ 表示卫星节点 u 与卫星节点 v 之间的星间链路在 t 时刻是断开的, $E_t < u, v > = 1$ 表示卫星节点 u 与卫星节点 v 之间的星间链路在 t 时刻是连通的。

定义 2.2 将节点集合 V 划分为若干个子集 V_1, V_2, \dots, V_n , $V = V_1 \cup V_2 \cup \dots \cup V_n$, 其中 V_n 定义为轨道面 n 上的卫星节点集合。

在上述簇结构的基础上对星座网络进行划分, 划分过程分为 3 个阶段:

第一阶段: 在每个轨道面分别划分出一个核心簇;

第二阶段: 将剩余游离节点划分入已存在的核心簇;

第三阶段: 对发生拓扑连接变化的簇内一般节点重新划分其簇归属;

下面分阶段阐述。

第一阶段:

设 V_c 为网络中的需要进行划分的卫星节点集合, 初值为网络的卫星节点集合 V 。

设节点的 $CP_i=0$, $v_i \in V$ 。

Stage1(1): 在 V_n 包含的节点中选择簇管理者, n 初始值为 1;

在 V_n 中管理约束度最低的节点中选择管理指数最大的节点作为簇管理者。若 v_i 满足 $CP_i = \min(CP_j, v_j \in V_n)$ 且 $W_i = \max(W_j, v_j \in V_n)$, 则选择 v_i 作为簇管理者, 若满足该条件的节点不止一个, 则选择 ID 号小的卫星作为簇管理者, 例如, 当 v_i 与 v_j 均满足前述条件, 且 $i < j$, 则选择 v_i 为簇管理者。

Stage1(2): 以 V_n 中选出的簇管理者为基础, 根据约束条件选择簇内核心节点;

使 V_n 中与 v_i 间隔两跳以内的节点加入以 v_i 为簇管理者的分簇 $CL_t(v_i)$, 成为簇内核心节点, 即 $CL_t(v_i) = \{v_i\} + \{v_j | j \in \{i-2, i-1, i+1, i+2\}, v_j \in V_{o1}\}$, $V_c = V_c - CL_t(v_i)$ 。

Stage1(3): 根据 V_n 中划分的核心簇, 修改相关节点的管理约束度, 并修改 n 的值为 $n+1$;

查询整个轨道周期内与 $CL_t(v_i)$ 中节点之间存在边的节点, 修改其管理约束度, 如下: 当存在 $v_j \notin CL_t(v_i)$ 且 $< v_j, v_p > \in E (v_p \in CL_t(v_i))$ 时, 修改 $CP_j = CP_j + 1$ 。

Stage1(4): 重复步骤 1~3, 直到 $n=N$ 结束;

该步骤结束时, 所有轨道面中均包含有一个簇管理者。

第二阶段:

将当前 V_c 中的节点划分到已存在的簇中, 成为簇内一般节点, 划分原则如下:

Stage2(1) 设 v_r 是剩余节点, v_r 需加入与其连通的簇, 即若加入簇 $CL_t(V_i)$, 须满足 $E_t < v_r, v_j > = 1 (v_j \in CL_t(v_i))$;

Stage2(2) 剩余节点优先选择异轨簇加入, 即若存在 $E_t < v_r, v_j > = 1 (v_j \in CL_t(v_i))$, 且 $V_a \neq V_b (v_r \in V_a, v_j \in V_b)$, 则优先选择簇 $CL_t(V_i)$ 加入, $CL_t(V_i) = CL_t(V_i) + \{v_r\}$;

Stage2(3) 当剩余节点 v_r 与两个簇管理者之间均存在链路时，则计算 v_r 与两个簇管理者之间的跳数，选择跳数小的簇加入；当 $v_r \in V_c$ ，设 v_r 与 $v_i (v_i \in CL_t(v_i))$ 之间的跳数表示为 $hop< v_r, v_i >$ ，当 $E_t < v_r, v_j > = 1 (v_j \in CL_t(v_i))$ 时计算 $hop< v_r, v_i > = 1 + hop< v_j, v_i >$ ；

Stage2(4) 当剩余节点 v_r 与两个簇管理者之间的跳数均相同时，选择与 v_r 之间的剩余连通时长最大的核心节点的归属簇加入； v_r 与簇内核心节点之间的剩余时长可表示为 $Tres< v_r, v_j > (v_j \in CL_t(v_i))$ ，该值可通过查询本地拓扑表获得；

Stage2(5) v_r 入簇， $V_c = V_c - v_r$ ；

Stage2(6) 对剩余的节点重复 Stage2(1)~Stage2(5)，直至 $V_c = \Phi$ ，划分过程结束。

下面对算法进行证明。

设上述算法一共划分了 m 个管理簇： CL_1, CL_2, \dots, CL_m ，则有定理如下：

定理 在网络 $G = \langle V, E \rangle$ 中，经算法 CDCA 所划分的管理簇 CL_1, CL_2, \dots, CL_m 是节点集 V 的一个划分，且 $CL_1 \cup CL_2 \cup \dots \cup CL_m = V$ ， $CL_1 \cap CL_2 \cap \dots \cap CL_m = \Phi$ 。

证明：首先证明管理簇的不重叠性。即证明，若存在节点 $v \in CL_i$ 则 $v \notin CL_j (i \neq j)$ 。这里从节点的属性出发，分别从 3 个层面证明。

第一，假设节点 v 在簇 CL_i 中的属性是簇管理者或簇内核心节点，则根据步骤 **Stage1(3)** 和 **Stage1(4)** 可知，当选定一个簇管理者或者节点已经成为簇内核心节点后，下一个簇管理者及簇内核心节点会从不同的轨道面中产生，因此节点 v 不可能成为另一个簇的管理者或簇内核心节点；根据 **Stage2** 可知，已经选为簇管理者的节点不可能成为簇内一般节点；

第二，假设节点 v 在簇 CL_i 中的属性是簇内一般节点，根据 **Stage2**，节点 v 不可能成为簇管理者或簇内核心节点；因此这里只需证明节点 v 不可能成为另一个管理簇的簇内一般节点即可，采用反证法，假设 $v \in CL_i$ 且 $v \in CL_j$ 。根据步骤 **Stage2(4)**，可得 $Tres< v, v_a > (v_a \in CL_t(v_i)) = Tres< v, v_b > (v_b \in CL_t(v_j))$ ，根据前述章节对网络拓扑的分析可知，异轨节点之间采取接续连通方式，因此在某一时刻同一节点的两条异轨链路的剩余连通时长不可能相同，这与假设矛盾。

由此可得，若节点 $v \in CL_i$ 则 $v \notin CL_j (i \neq j)$ ，即 $CL_1 \cap CL_2 \cap \dots \cap CL_m = \Phi$ 。

再次，证明管理簇的完整性。即证明 $CL_1 \cup CL_2 \cup \dots \cup CL_m = V$ 。

由步骤 **Stage2(6)** 可知，划分过程结束时， $V_c = \Phi$ 。由此可知对于划分前任意节点 $v \in V$ ，必然存在 $v \in CL_i (i=1,2,\dots,m)$ ，故 $v \in CL_1 \cup CL_2 \cup \dots \cup CL_m$ ，可得 $V \subseteq CL_1 \cup CL_2 \cup \dots \cup CL_m$ 。又由 CL_i 定义可知， $CL_1 \cup CL_2 \cup \dots \cup CL_m \subseteq V$ ，由此， $CL_1 \cup CL_2 \cup \dots \cup CL_m = V$ 。

第三阶段：

初始的管理簇划分之后，当新的拓扑时刻到来，节点之间的拓扑连接也会随之变

化，这可能导致簇内一般节点与原来簇之间的连接断开，从而引起簇的变化。这就需要重新对节点划分簇的归属。过程如下：

Stage3(1) 当新的拓扑时刻到来后，簇内一般节点 v 离开原来的簇 CL_i ，即 $CL_i = CL_i - \{v\}$ ， $V_c = V_c + \{v\}$ ；

Stage3(2) 对 V_c 中节点进行划分，划分过程如上述 **Stage2(1)~Stage2(6)**，直至 $V_c = \Phi$ ；

2.4.2.3 (24, 3, 1) 低轨星座网络分簇过程描述

图 2.4-2~图 2.4-6 显示了(24,3,1)低轨星座网络采用上述管理簇划分算法的过程。在划分之前网络拓扑结构如图 2.4-2 所示，其中实线表示当前拓扑时刻连通的星间链路，虚线表示当前拓扑时刻断开但在轨道运行周期内存在的星间链路。

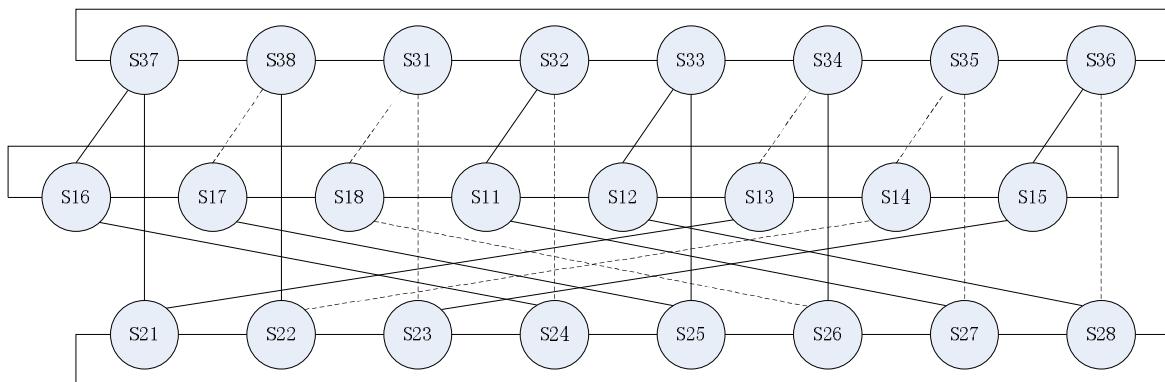


图 2.4-2 某个拓扑区间 t 卫星节点初始状态

首先选择轨道 1 中管理指数最大的节点作为第一个簇管理者，为简单起见，这里假设各节点的计算能力、存储能力、寿命均相同，管理指数的大小仅与节点同时建立 4 条星间链路的总时长有关，各节点在一个轨道周期内同时建立 4 条星间链路的总时长见表 2.4-1。

表 2.4-1 一个轨道周期内各节点同时建立 4 条星间链路的总时长

轨道面 1	建链时长 (min)	轨道面 2	建链时长 (min)	轨道面 3	建链时长 (min)
S11	16	S21	16	S31	21
S12	54	S22	16	S32	16
S13	16	S23	20	S33	54
S14	46	S24	17	S34	18
S15	16	S25	16	S35	21
S16	54	S26	16	S36	16
S17	16	S27	20	S37	54
S18	46	S28	17	S38	18

根据表中所示，第一个簇管理者选择 S12，则根据步骤 1，S11、S13、S14、S18 加入以 S12 为管理者的簇 $CL_i(S12)$ ，如图 2.4-3。同时，修改轨道 2 和轨道 3 上相关节点的管理约束度，即修改与簇 $CL_i(S12)$ 中节点之间存在异轨链路的节点管理约束度

为 1，包括节点 S21、S22、S26、S27、S28 以及 S31、S32、S33、S34、S35。

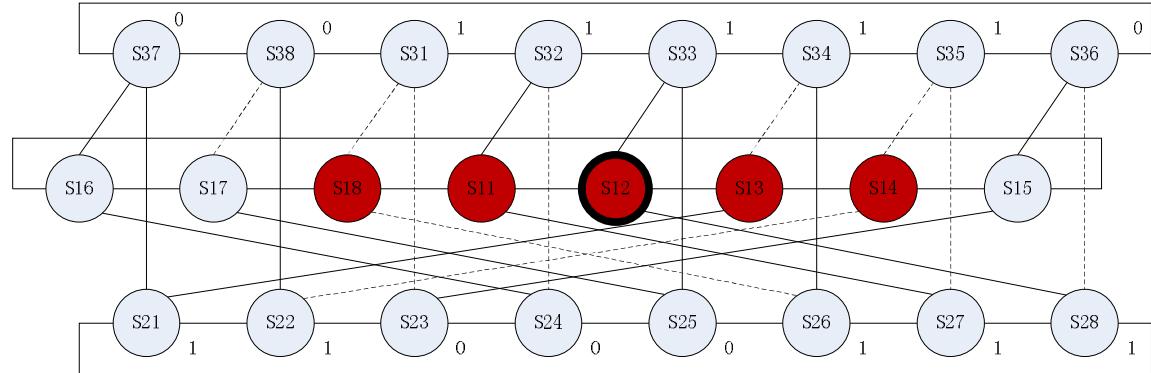


图 2.4-3 第一个簇产生过程

在轨道 2 中首先选择管理约束度最低的节点作为候选簇管理者，包括 S23、S24、S25，再选择管理指数最高的节点作为第二个簇的管理者，即 S23，并使节点 S21、S22、S24、S25 加入簇 $CL_t(S23)$ ，同时修改轨道 3 和轨道 1 中相关节点的管理约束度，如图 2.4-4 所示。

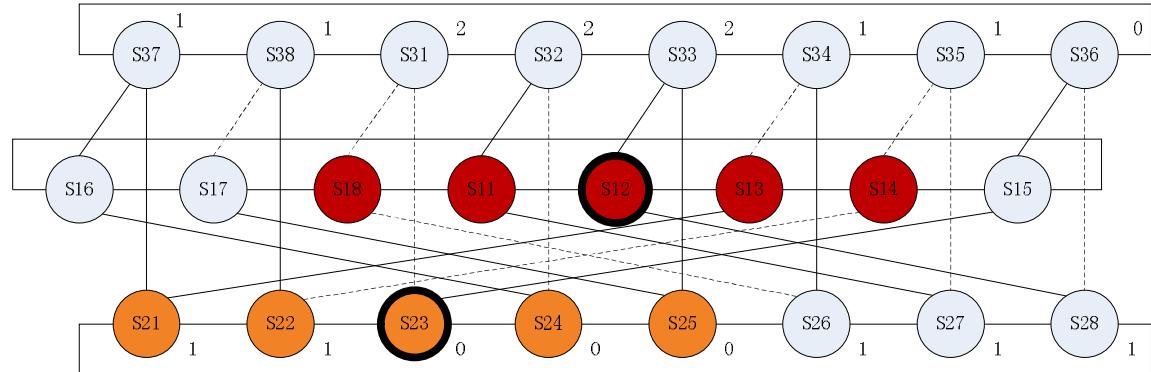


图 2.4-4 第二个簇产生过程

在轨道 3 中选择管理约束度最低的节点作为候选簇管理者，此时只有一个节点管理约束度为 0，即 S36，因此选择 S36 作为第三个簇管理者，并使节点 S34、S35、S37、S38 加入簇 $CL_t(S36)$ ，如图 2.4-5 所示。

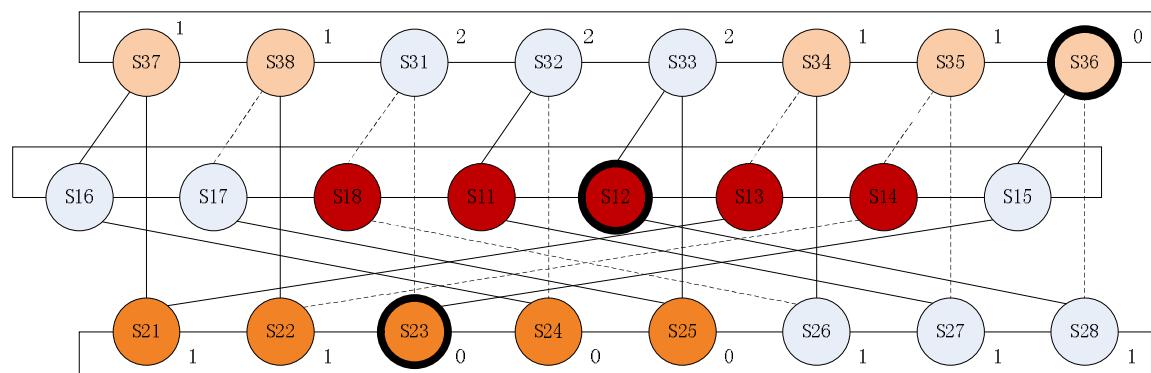


图 2.4-5 第三个簇产生过程

在上述过程之后，网络中存在 3 个簇，对应 3 个轨道面。此后根据步骤 6 对剩余未入簇的节点进行划分，根据跳数与剩余时长的计算比较，剩余节点 S16、S15、S17、S26、S27、S28、S31、S32、S33 入簇结果如图 2.4-6 所示。

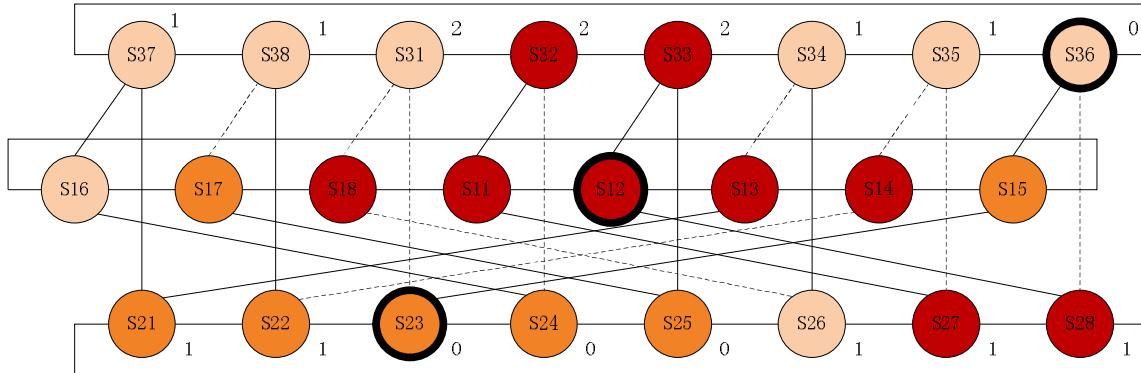


图 2.4-6 剩余节点入簇过程

2.4.3 管理簇的形成

上一小节对簇的划分过程进行了描述，侧重于划分的规则及策略，没有涉及到实现过程。本文中，簇是为卫星网络的自主管理而服务的，在实际应用中，簇的形成过程需要通过节点之间的消息交互来自主完成，本节描述簇形成过程中涉及的消息交互以及节点自身的处理流程。

簇的形成通常分为两个阶段：簇的建立和簇的保持。下面分阶段阐述本论文所提出的管理星簇形成算法。

(1) 管理星簇的建立

簇建立过程是指确定簇管理者，确定节点簇归属的初始过程。该阶段需要根据每个节点广播的管理指数来选择簇管理者，并通过节点之间的同轨或异轨关系选择簇内核心成员，主要利用的是同轨道节点的相对固定连接关系来保持管理簇的稳定性。

不同于传统的 DWCA 算法，本文提出的 CDCA 算法在初始选择簇管理者进行广播的方式不是全网广播，而是采用按照轨道面按照顺序进行广播的方式。下面针对 (24,3,1) walker 星座网络描述其管理簇形成过程。

簇建立的步骤如下：

Step 1: 轨道面 1 上的每个节点向同轨的所有节点发送管理声明 **Manage_Inform**: <*my_id*, *weight*, *limit_degree*>，其中包含了它的 ID 号和管理指数 W_i 以及管理约束度 CP_i ；

Step 2: 收到同轨道所有节点的管理声明后，首先比较同轨面上节点的管理约束度，在管理约束度最小的前提下再比较管理指数，管理指数最大的节点以簇管理者的身份向其同轨的邻居节点发送成簇通知 **CH_Inform**: <*Cid*>；

Step 3: 收到成簇通知的节点向簇管理者发送成簇应答消息 **ACK_CH_message**: <

$\text{my_id}, \text{Cid}$; 同时向距离其一跳的同轨游离邻居节点（核心成员）转发收到的成簇通知 CH_Inform , 同样，收到该成簇通知的节点向簇管理者发送应答消息，此时建立的簇称为初始簇；

Step 4: 簇管理者收到所有 4 个应答消息后向全网发布成簇广播 Cluster_Declare : $\langle \text{Cid}, \text{Kid1}, \text{Kid2}, \text{Kid3}, \text{Kid4} \rangle$, $\text{Kid1}, \text{Kid2}, \text{Kid3}, \text{Kid4}$ 表示簇内核心节点的 ID 号；

Step 5: 所有收到 Cluster_Declare 的游离节点，查询本星是否存在边 $\langle u, v \rangle$ ，其中 u 为本节点， v 为 Cid 所在簇内节点，若存在，则修改自己的管理约束度 $\text{CP}=\text{CP}+1$ ；

Step 6: 对轨道面 2 上收到 Cluster_Declare 的游离节点，重复 Step1~4，所有轨道面上游离节点重复 Step5；

Step 7: 对轨道面 3 上收到 Cluster_Declare 的游离节点，重复 Step1~4，所有轨道面上游离节点重复 Step5；

Step 8: 收到轨道面 3 的 Cluster_Declare 的游离节点，选择当前时刻与其连通的异轨链路节点所在的簇加入，若两个异轨链路均连通，则计算与每个簇管理者之间的跳数，选择跳数最小的簇加入，若跳数相同，则选择当前时刻该链路剩余连通时长较长的簇加入，若不存在异轨链路，则选择与其连通的同轨链路节点所在的簇加入。入簇申请表示为 Register_message : $\langle \text{my_id}, \text{Cid} \rangle$, 向相应的簇管理者发送；

Step 9: 收到入簇申请的簇管理者返回入簇申请应答 $\text{ACK_register_message}$: $\langle \text{Cid} \rangle$ 给申请节点；

最后，当星座网络中没有游离状态的节点时，管理簇建立过程完成。

下面采用伪代码方式对算法的执行过程进行描述，在此之前先定义一些变量如表 2.4-2 所示。

表 2.4-2 算法执行代码变量定义

变量名称	含义	备注
<code>PRESENT_ORBIT</code>	当前成簇轨道面	初值为 1
<code>Neighbor_SO</code>	同轨面邻居节点集合	
<code>Neighbor_next_SO</code>	同轨面一跳邻居节点集合	
<code>Neighbor_next_DO</code>	异轨面一跳邻居节点集合	
<code>N</code>	轨道面个数	
<code>status</code>	节点状态	包含四种状态：簇管理者（ <code>CH</code> ）；簇内核心节点（ <code>COR</code> ）；簇内一般节点（ <code>COM</code> ）；游离节点（ <code>L</code> ）；初值为 <code>L</code>
<code>Cid_k</code>	<code>id</code> 为 k 的节点所属簇的管理者 <code>id</code>	

伪代码：

```

for(;;)
{
    if(my_orbit_id==PRESENT_ORBIT)
    {
        broadcast Manage_Inform<my_id, my_W, my_CP>;
        Wait;
        On receiving all Manage_Inform from my orbit
        {
            if(my_CP==min(CPk) && my_W==max(Wk)(k∈{sat_id of my orbit}))
            {
                if(my_id==min(id)(id∈{sat_id of max(Wk)}))
                {
                    broadcast CH_Inform<CID> to Neighbor_SO;
                    Status='CH';
                }
                Wait;
                On receiving 4 ACK_CH_message from its Neighbor_SO
                    broadcast Cluster_Declare: <CID, Kid1, Kid2, Kid3, Kid4>;
                }
            else
            {
                On receiving CH_Inform<CID>
                {
                    transmit ACK_CH_message<my_id, CID> to cluster_head;
                    if (cluster_head∈Neighbor_next_SO)
                        transmit CH_Inform<CID> to Neighbor_next_SO(except cluster_head);
                    status='COR';
                }
            }
        }
    }
    else
    {
        On receiving CH_Declare<CID, Kid1, Kid2, Kid3, Kid4>
        {
            if(exist <V_k, myself>)(k∈{CID, Kid1, Kid2, Kid3, Kid4})
                my_CP=my_CP+1;
            PRESENT_ORBIT=n+1;(CH_Declare from Orbit n)
            if (PRESENT_ORBIT==N+1)
                Jump from Zoop;
        }
    }
}
if(status=='L')
{
    if(E<V_k, myself>==1(Vk∈Neighbor_next_DO))
    {
        Caculate hop(my_id,CID);
        Caculate Tres<V_k, myself>;
    }
    if(number of k==1)
    {
        transmit Register_message<my_id, CIDk> to Sat_CIDk;
        status=='COM';
    }
    if(number of k==2)
    {
        if(hop(my_id,CID)==min(hop(my_id,CIDk))&&Tres(my_id,k)==min(Vk∈Neighbor_next_DO))
        {
            transmit Register_message<my_id, CIDk> to Sat_CIDk;
            status=='COM';
        }
    }
    if(number of k==0)
    {
        transmit Register_message<my_id, CIDp> to Sat_CIDp;Vp∈Neighbor_SO
        status=='COM';
    }
}
else if(status=='CH')
{
    On receiving Register_message<my_id, CID>
        transmit ACK_register_message<CID> to register;
}

```

图 2.4-7 分簇算法伪代码

(2) 管理星簇的保持

第二个阶段是星簇的保持，本文中启动簇保持过程的条件是节点的拓扑连接关系改变，这种情况下发起离簇动作的总是簇内一般节点。

簇保持过程如下：

- 簇内节点保存有拓扑变化的时刻表，当下一个拓扑时刻到来之前 2s 时，簇内一般成员节点检查与簇内核心成员节点的连接关系是否变化，若变化，则向簇管理者发送离簇声明 **Move_Declare: < my_id, CID >**；

- b) 簇管理者收到簇内节点发送的 Move_Declare 之后，返回应答消息 ACK_Move_message: < Cid >;
 c) 簇内节点收到 ACK_Move_message 后将自己的状态修改为游离状态；
 d) 游离节点在下一个拓扑时刻到来之后重新选择新簇加入，过程同前述 2.4.3.1 中 step8~step9；

表 2.4-3 对本算法中用到的消息类型进行了总结。

表 2.4-3 CDCA 中簇建立与保持过程中消息类型

消息	描述
Manage_Inform: < my_id, weight, limit_degree >	声明节点的 id、管理指数和约束度
CH_Inform: < Cid >	声明节点的 id 和簇管理者身份
ACK_CH_message: < my_id, Cid >	节点对 CH_Inform 的应答
Cluster_Declare: < Cid, Kid1, Kid2, Kid3, Kid4 >	声明核心簇的组成
Register_message: < my_id, Cid >	游离状态节点申请入簇的消息
ACK_register_message: < Cid >	簇管理者对节点入簇申请的应答
Move_Declare: < my_id, Cid >	节点向簇管理者发出的离簇通告
ACK_Move_message: < Cid >	簇管理者对节点离簇申请的应答

2.5 仿真与分析

本文对 CDCA 在星座网络中的性能作了仿真，并与 DWCA^[19]作了比较。仿真场景采用轨道高度为 1450km、轨道倾角为 30 度的 (24,3,1) walker 星座。仿真时长取一个轨道周期，这么做是由于星座网络的拓扑结构变化随轨道周期变化，因此一个轨道周期内的性能基本可以代表整个星座正常运行状态时的性能。

根据前述章节，衡量一个分簇算法优劣的标准包括簇结构的稳定性、簇的数量、负载均衡度、成簇及簇维持的网络开销等等。在特定的 24 星星座模型下，由于节点数量有限，本文的分簇算法在簇的数量和负载均衡度方面与传统算法不具有明显差异，因此本文主要从簇结构的稳定性、成簇过程中的网络开销、成簇时延等几个方面说明 CDCA 的性能。

(1) 簇结构的稳定性

对自主管理的星座网络来说，簇结构的稳定性直接影响管理开销的多少，簇结构越稳定，管理开销越少。这里对地面传统 DWCA 分簇算法应用在我们的星座网络中的性能进行了仿真，并与本文提出的 CDCA 算法进行了比较，如图 2.5-1 所示：

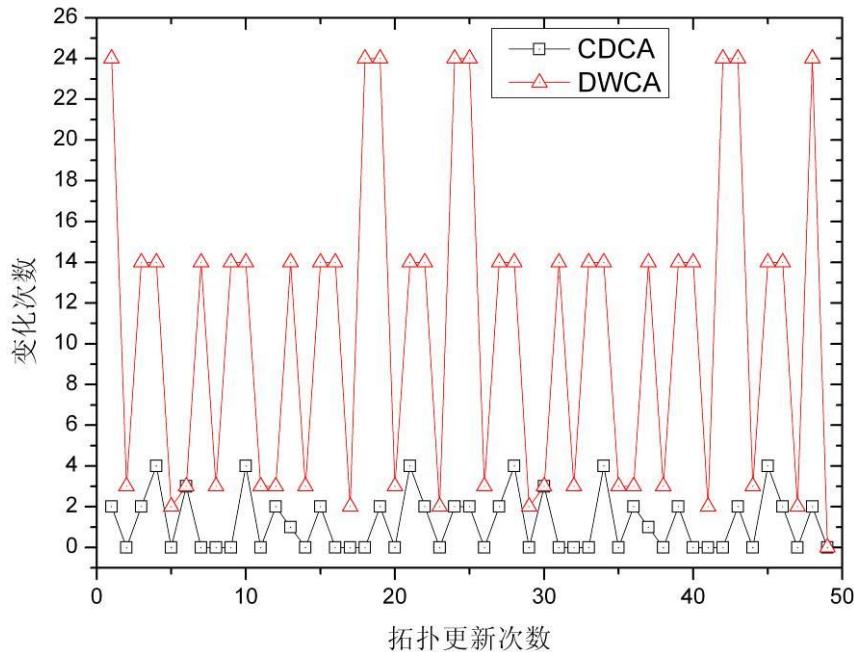


图 2.5-1 CDCA 簇结构稳定性

上图是两种算法在一个卫星轨道周期内的节点状态变化次数的比较，横坐标表示拓扑更新次数，一个轨道周期内共有 48 次。节点状态变化是指节点的身份发生变化、节点的隶属簇关系发生变化两种。节点状态变化次数越多，簇的结构越不稳定。统计表明，一个轨道周期内，DWCA 节点状态变化次数平均 526 次，而 CDCA 为 60 次，明显优于 DWCA 算法。

这是由于 DWCA 算法需要实时监测节点自身的管理权重变化，当管理权重的变化使得当前簇首不满足权重最大的条件时，就需要对簇首进行重新选择，整个簇结构就需要重新建立。DWCA 中管理权重与节点当前状态有关，当拓扑结构发生变化时，节点的管理权重将随着星间链路的断开或连通以及连通剩余时长发生变化，带来的后果是节点状态的不断变化。而本文的 CDCA 算法由于将同轨面相邻的一组节点作为簇的固定组成，簇的主要构成不随拓扑变化而变化，仅仅簇内一般节点需要改变其簇归属，因而节点状态变化次数较少。

(2) 网络开销

图 2.5-2 是 CDCA 和 DWCA 两种算法在网络开销方面的性能比较，仿真了簇建立过程和簇保持过程中所产生的包个数总和，其中簇保持过程的网络开销对应一个轨道周期内每次拓扑更新时刻所产生的包数量。

图中第一个拓扑区间的网络开销较大，是由于簇建立过程发生在该区间，而随后的拓扑区间只需进行簇保持，因而开销较小。由仿真图可以看出，在初始的簇建立过程中，CDCA 和 DWCA 所产生的包开销基本在一个量级，CDCA 略少于 DWCA，这是由于 CDCA 在进行簇管理者选择时只进行定向的消息广播，DWCA 则向所有当前连通的邻居节点进行广播。在随后的簇保持过程中 DWCA 的开销则明显大于本文的 CDCA 算

法, 这与前面节点状态变化次数的仿真结果是相一致的, 节点状态的变化意味着需要产生簇维持的包开销。

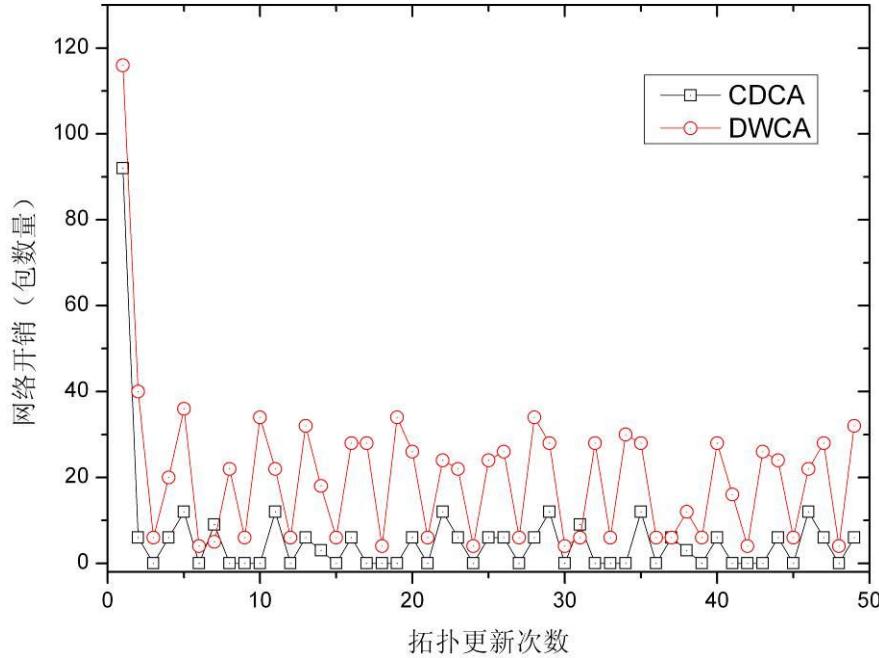


图 2.5-2 CDCA 的网络通信开销

(3) 成簇时延

本文对簇的形成过程在每个拓扑区间所占用的时间进行了仿真, 如图 2.5-3 所示。图中第一个拓扑区间为簇建立过程, 而随后的拓扑区间为簇保持过程。成簇的时延包括了成簇过程中消息交互的传输时延、星上处理时延等。成簇时延过大会影响正常的网络管理的实时性。

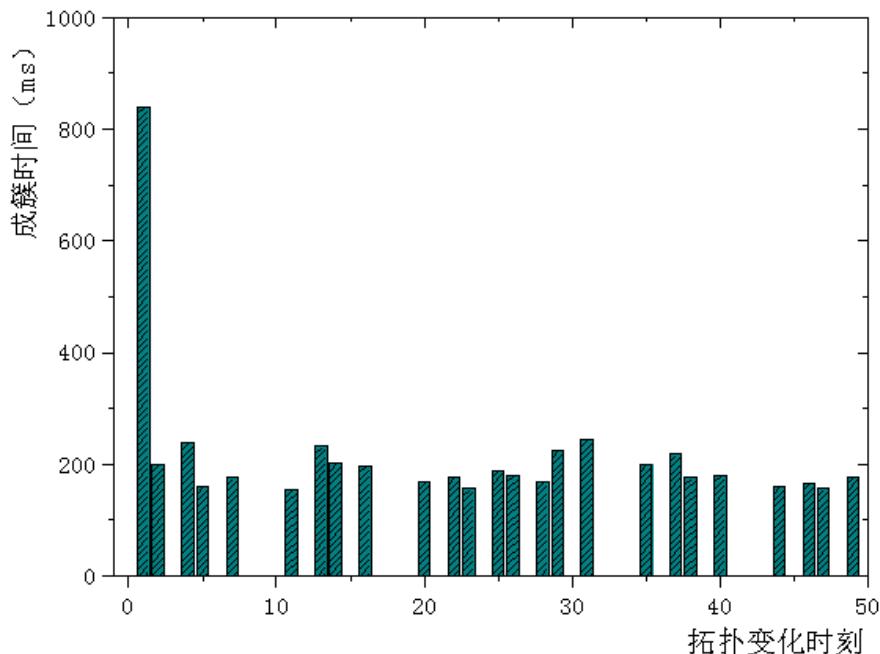


图 2.5-3 CDCA 算法成簇时延

由图中可以看出, CDCA 算法的初始簇建立过程所占用的成簇时间约为 840ms 左右。

右，在每个拓扑更新区间切换前后所进行的簇维持时间平均约为 180ms。由此可见，成簇时延相对于一个拓扑持续的时间来说是比较小的，对管理信息的传输影响不会很大。

2.6 小结

本文对应用于星座网络自主管理的分簇算法进行了研究，尤其对于链路带宽资源受限、业务实时性要求高的通信星座网络来说，分簇管理是对其进行分布式管理的基础。本文在深入分析低轨通信星座网络特性的基础上，根据卫星的实际运动模式与通信特性提出了基于约束条件的分布式分簇算法 CDCA。该算法充分利用了网络本身所固有的同轨链路稳定的特性，使所建立的簇结构具有更好的稳定性，从而有效降低簇维护所造成的网络开销，减轻了网络负担，使得分簇管理在星座网络中的应用更加实用，为星座网络的高效自主管理奠定了基础。仿真分析也表明，与其它分簇算法比较，CDCA 具有更好的簇稳定性和更少的维护开销；同时成簇过程所占用的时间并不会影响到正常的网络管理。

第三章 星座自主运行中的位置管理策略研究

3.1 引言

在低轨卫星星座系统中，卫星相对于地面做高速运动，因此卫星与用户终端之间的可视时间非常短，大约为十几分钟。由于卫星的高速运动和用户的移动性，**LEO** 卫星网络为了能发现移动终端的接入点，将呼叫传递到移动用户，必须对用户的位置信息进行记录，即具有位置管理的功能。

根据前面章节 1.1 中图 1.1-1 所表示的星座卫星通信系统的组成，通常位置管理由地面段的运控系统完成。然而，本文所讨论问题的前提是在星座网络自主管理模式下进行，即在没有地面站支持的情况下完成对用户业务的支持。星上实现用户位置管理受卫星资源和能力的限制，较难满足全网用户容量的需求，因此这里的星上位置管理适用于自主运行周期短、用户容量低的特殊工作模式，或者星上管理境外用户、地面管理境内用户的混合管理工作模式。为了简化问题，本文在研究位置管理策略时屏蔽了应用场景，仅关注无地面站支持条件下的星上自主位置管理策略。

位置管理相关研究工作主要围绕着无线侧的移动台位置更新和终端寻呼，以及网络侧位置注册和呼叫传递两个方面进行的。无线侧的位置管理策略主要围绕位置区划分、寻呼策略展开研究。对于网络侧的位置管理，主要的研究围绕位置管理数据库网络结构展开。本文主要针对网络侧的位置管理进行研究。

本章针对星座自主运行的特点，提出了基于簇的多 **HLR** 自主位置管理策略，并引入移动 **agent** 的概念，与传统指针转发策略相结合，从而解决仅仅采用传统指针转发时由卫星网络拓扑变化造成的指针失效问题，同时有效地降低位置更新的信令开销。

3.2 位置管理概述

3.2.1 地面移动蜂窝系统中的位置管理

移动通信系统中，用户的位置管理方案主要有动态和静态两种。地面移动蜂窝系统大多采用固定位置区策略，将覆盖范围划分为多个固定不变的位置区(**Location Area**)来作自动位置管理。每个 **LA** 包含若干蜂窝小区，移动终端运动到不同的 **LA** 中就需要进行位置更新，对于到一个移动终端的呼叫，需同时轮询当前 **LA** 中的所有小区。固定位置区方案如图 3.2-1 所示。

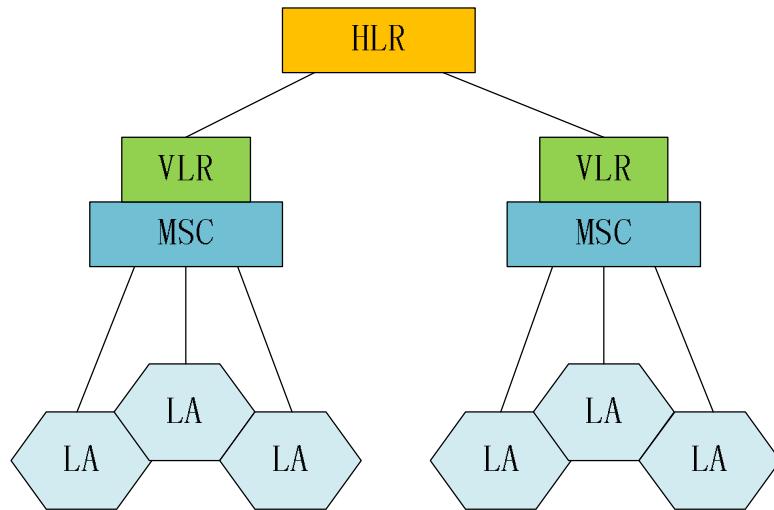


图 3.2-1 静态位置区方案

动态位置管理方案^[42]中则不存在明确位置区划分和边界,位置更新过程依赖于移动用户的呼叫和移动模型。现有动态更新策略主要有三种:基于时间^[36]、基于运动^{[37][38]}和基于距离^[39]。

位置更新方法结合了周期性位置更新和过位置区边界时的位置更新 2 种。每当移动台检测到它在跨越 LA 时就产生一次位置更新,不过基于静态位置区的方案在这种策略下很容易产生乒乓效应。如果网络和移动台之间在一段固定的时间内(比如 3h)都没有通信(即移动台处于空闲模式),移动台也应产生位置更新,以便在系统数据库失败的情况下恢复用户位置数据。

位置信息的管理通过网络数据库进行,采用两级数据库(即归属位置寄存器 HLR 和访问位置寄存器 VLR^[40])的模式。

基于这样分等级的垂直管理的网络结构,存在诸多问题:简单的位置区划分必然导致移动管理控制信令分布的不均匀,而无线资源本身是均匀的,信令集中在位置区边界,无线资源就不能有效充分利用。此外,固定而不重叠的位置区可能导致移动台在相邻的 LA 间来回运动引发乒乓效应,导致不必要的位置更新。同样地,用户在两个 VLR 之间(或边界)来回运动,也会导致 HLR 频繁地对这两个 VLR 进行插入用户数据和删除用户数据的操作。这些不可避免的震荡,必然对其上一级节点造成不必要的信令负担,浪费无线资源或信令带宽。

因此,针对网络侧的位置管理,主要围绕位置管理数据库网络结构展开,一类是基于现有网络结构(HLR/VLR)的集中式方案,一类是采用全新结构的分布式方案,相关研究策略包括:指针转发、局部锚点以及动态分层等^[41]。

传统的集中式方案通常采用两级数据库结构^{[41][43]},配置有两种类型的数据库:归属数据库 HDB(Home Database)和访问数据库 VDB(Visitor Database),HDB 在系统中通常被称为归属位置寄存器 HLR(Home Location Register),VDB 通常被称为访问位置寄

存器 VLR(VisitorLocation Register)。HLR 存储归属网络中所有用户的相关信息，包括接入权限、密码和用户位置等。每个 VLR 则存储它所关联的位置区所登记的所有用户的数据。只要移动终端 MT(Mobile Terminal)移动到新的位置区 LA(LocationArea)，新 VLR 就会向其 HLR 报告 MT 的新位置信息；同时 HLR 通知旧 VLR 删除 MT 的信息。当有呼叫时，通过查找 HLR 中的信息，传递呼接到 MT 所在的 VLR，根据 MT 的状态建立呼叫。在两级数据库管理策略的基础上，衍生了另外一些优化数据库结构，以达到减小位置管理开销和提高系统性能的目的，包括用户位置缓存^[44]、用户文档复制^[45]、前向指针^[46]、本地缓存^[47]等方案。

任何一种位置管理策略都是以降低位置管理代价为目的的，文献^[48]便提出了一个通用的位置管理代价估计模型，适合于不同的位置管理算法及应用场景，对位置管理的优化研究提供了分析基础。

近年来，随着技术的发展与更新换代，越来越多的新技术与位置管理相结合，从而提高位置管理的代价性能，文献^[49]采用模糊逻辑技术降低位置管理的代价；文献^[50]将移动数据库技术用于位置管理，充分应用了 Agent 技术的移动性和前向指针法更新开销小的优点，在数据库开销相同的情况下，其它网络条件不变的情况下，使操作本地化，减少了消息传输的距离，从而减少了网络的整体开销，使位置管理系统得到了改进，提高了性能。

另外，随着网络时代的爆发，越来越多的研究将目光转向下一代及下下代移动通信网络中位置管理技术^[51]。

3.2.2 低轨移动通信星座系统中的位置管理

低轨卫星是未来卫星移动通信系统的一个重要的发展方向。**LEO** 卫星轨道高度在一般为 700-1500km，单颗 LEO 卫星对地面移动终端服务时间很短，大约在 10 分钟。在地面蜂窝移动通信系统中，影响位置管理技术的主要因素是移动终端运动特征，而在 LEO 卫星网络中，位置管理方案的选取则主要取决于卫星的运动，所以不能照搬地面蜂窝移动通信系统的位置管理技术，必须对其进行改进以适用于 LEO 卫星网络。

LEO 卫星网络中的位置管理同样涉及位置区设计、位置更新、位置寻呼以及位置信息数据库管理等几个方面的问题。

LEO 卫星系统中的位置区划分设计方案有很多，包括有基于用户地理位置的静态位置区^[52]、基于卫星波束覆盖的位置区^[53]、基于地面站覆盖的位置区、同时基于地面站和卫星覆盖的位置区^[54]几种，也有类似地面移动网络中研究较多的基于用户地理位置的动态位置区划分^[60]。

不同的位置区划分带来的位置更新和寻呼代价也不同，针对 LEO 移动通信网络的位置管理研究，主要从位置区设计及相应的位置更新策略^{[55][56][57]}、寻呼策略^[58]、

位置数据库管理策略^{[59][60][61][62]}等几个方面进行了研究，以降低位置更新信令开销、寻呼代价为目的，提出了优化改进的方案。

本文主要关注网络层的位置管理策略研究，这方面的研究旨在通过位置数据库结构的设计来降低位置管理的信令开销。文献^[59]提出了基于前向指针和区域注册的动态 LEO 位置管理策略，该策略通过动态调整区域小区个数，使得在移动终端的移动特性和切换失败概率不同的情况下位置管理代价最优化；文献^[60]提出了一种在卫星上面设置 VLR 的数据库管理方案，降低了国外用户呼叫建立时延以及位置管理的开销。文献^[61]提出设置多个 HLR 的位置管理方案，降低位置管理的信令开销，同时也考虑了数据库的容灾功能。

3.3 星座自主运行时的位置管理策略

3.3.1 基于卫星波束覆盖的动态位置区划分

LEO 星座自主运行时，将失去地面站的支持，位置管理不得不由卫星网络自主完成。同前所述，用户位置管理通过定义位置区的大小和地点实现，当移动终端超出了位置区的范围，则发起位置更新，网络就会相应地更新其位置信息，这样才能达到在有呼叫到达时通过正确的位置区寻呼用户的目的。

位置区的划分可分为静态位置区和动态位置区两种，在 LEO 星座系统中，静态位置区主要是基于地面站覆盖区的固定位置区，而动态位置区划分则可分为基于用户地理位置的动态位置区、基于卫星波束覆盖的动态位置区、基于卫星波束和地面站覆盖的动态位置区。显然，静态位置区划分方法并不适合自主运行的星座系统。而基于用户地理位置和基于卫星波束和地面站覆盖的动态位置区划分方法均需在地面站支持下进行用户的寻呼，同样也不适合本文的研究环境。因此，本文采用基于卫星波束覆盖的动态位置区划分方法，在此基础上，如何降低位置更新开销是优化设计的目标。

基于卫星波束覆盖的动态位置区划分如图 3.3-1 所示。

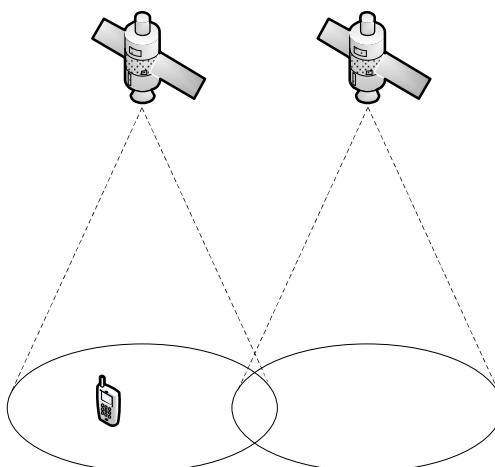


图 3.3-1 基于卫星波束覆盖的动态位置区

由于卫星相对于用户终端的移动速度很高，即使用户固定位置，实时覆盖它的卫星或卫星波束仍然会发生频繁的变化，尤其是波束的切换相对于卫星的切换更加频繁，从寻呼方便的角度出发，可按照波束变化进行位置注册，从降低开销的角度出发，可按照卫星变化进行位置注册。考虑到卫星信道资源的宝贵，本文采取基于卫星变化的位置注册策略，即当覆盖移动终端的卫星发生变化时，即发起一次位置注册操作。

在此基础上，位置区内的寻呼策略可采用文献^[63]中的结合时间和移动的位置更新方案计算用户位于卫星波束的概率，再通过基于概率的分组寻呼方案^[64]对用户进行位置寻呼，从而降低寻呼的开销。

本文关注的是网络侧的位置管理问题，主要研究星座网络如何自主进行位置数据库的管理，以及怎样正确查找到当前覆盖用户的卫星，从而寻呼到用户等。

3.3.2 基于簇的多 HLR 位置管理构架

本文提出星上位置管理的方案，即在星上设置 HLR 和 VLR 的数据库管理方案，可作为星座系统自主运行时的用户位置管理解决方案，同时也可解决境外用户访问境内地面站 VLR 的长距离信令传输问题。

传统的数据库管理均是在地面进行，然而卫星是具有移动性的，照搬由地面站完成的位置管理策略显然并不合适，具体表现在以下两个方面：

(1) 地面位置管理中的 HLR 和 VLR 的设立是基于地理位置的，它们之间的绝对位置和相对位置均是不变的，而卫星之间的绝对位置和相对位置均是在不断变化的，设置 HLR 和 VLR 需考虑卫星移动性对它们之间通信造成的影响；

(2) 星上若采用传统单 HLR 的集中式位置管理方式，则会引起网络负载的不均衡，同时仍然无法解决 VLR 与 HLR 之间的长距离信令传输问题；

从另一方面来说，卫星的移动是具有规律的，这使得网络拓扑关系也是有规律变化的，利用这种规律性来设置星上的 HLR 和 VLR 可以有效解决上述问题。

通过本文在前述 2.3.2 节中对网络拓扑的分析可知，同轨卫星上的相对位置是不变的，利用这一特点本文在 2.4 节提出了基于约束条件的分布式分簇算法，通过簇对网络进行管理，通过该算法划分的簇结构同样适用于本章的位置管理。因此，在此基础上本文提出一种基于簇的多 HLR 位置管理策略，其数据库架构如图 3.3-2 所示：

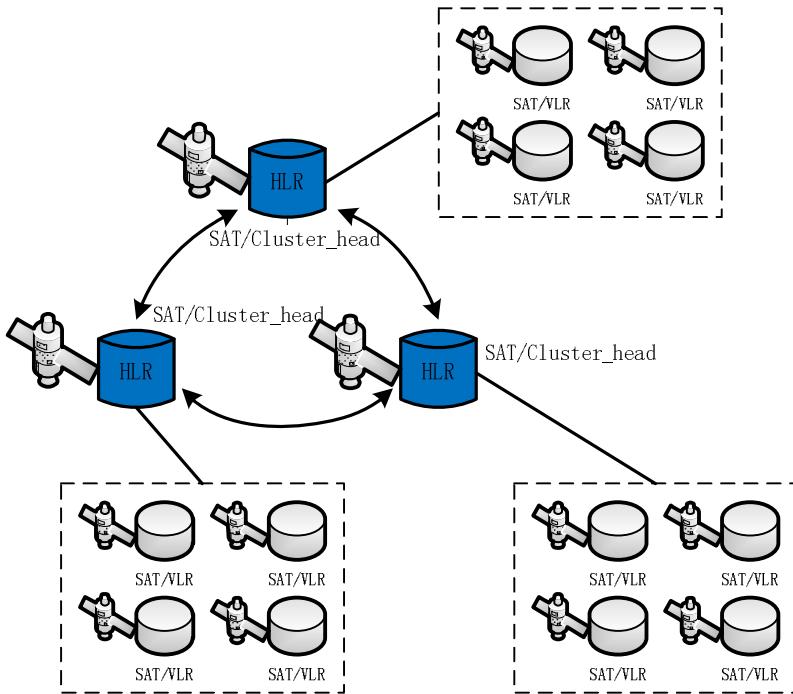


图 3.3-2 基于簇的多 HLR 位置管理架构

由于卫星数量有限，并且每颗卫星波束覆盖的面积较大，同时卫星之间的相对位置可能随时间变化，因此这里在每颗卫星上均设置一个 VLR，管理本星波束覆盖下的用户，而不再像地面网中采用多位置区共用一个 VLR 的方案。

如图 3.3-2 所示，簇管理者扮演 HLR 的角色，而簇内其它卫星节点则作为 VLR，特别地，簇管理者除了承担 HLR 的作用，同时还作为本地 VLR 使用。需要位置查询时则通过簇与簇之间的协作完成。由此可知，正常状态下，簇管理者的数量即为网络中 HLR 的数量；根据前一章簇的划分策略，簇管理者是稳定不变的，而 HLR 是与卫星绑定的，因此它不会随意在卫星之间移动，是稳定的数据库，这为 HLR 的管理提供了便利，避免了由于卫星移动而造成的数据库迁移问题；但另一方面，就 HLR 中存储的用户信息来说，它又是一个动态的数据库，这是由于网络中存在多个 HLR，而卫星网络是动态变化的，这就使得 HLR 服务的用户对象也在不断变化。

本方案中，使当前为用户提供服务的卫星成为该用户的 VLR，因此用户的 VLR 是随着卫星的移动而不断变化的，这也造成每颗卫星上的 VLR 中存储的用户信息是动态变化的，也就是说，星上 VLR 也是一个动态数据库，这一点上同地面位置管理是类似的。不同于地面策略的是，星座网络的簇管理者既是 HLR 又是 VLR。

为了后续章节描述方便，这里采用 SAT/HLR 表示星上设置的位置归属寄存器，SAT/VLR 表示星上设置的位置访问寄存器，SAT n/HLR k 表示编号 n 的卫星上设置的编号 k 的 HLR，SAT n/VLR n 表示编号 n 的卫星上设置的编号 n 的 VLR。

3.3.3 基于簇的多 HLR 位置更新策略

由于卫星运动速度相对地面用户来说非常快，地面用户的移动速度可忽略。下图是地面上某一固定位置用户在 16 小时内被卫星覆盖的情况以及覆盖卫星的归属簇变化情况示意图。

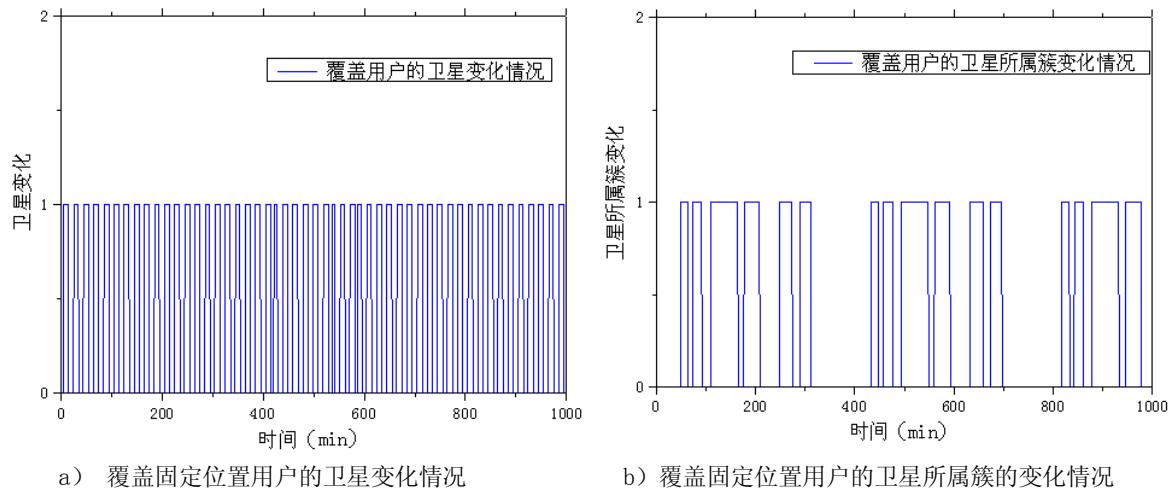


图 3.3-3 固定位置用户在 16 小时内被卫星覆盖的情况

图 3.3-3 中横坐标为仿真时间，纵坐标值发生一次变化，代表覆盖用户的卫星发生了变化（图 a）或者代表覆盖用户的卫星所属簇发生了变化（图 b）。从图上可以看出，覆盖用户的卫星所属簇发生变化的次数远远少于卫星发生变化的次数，说明当覆盖用户的卫星发生变化时，其归属簇不一定发生变化。

根据这一特点，本文提出了基于簇的位置管理策略，能够有效地减少位置更新信令开销。同时，由于存在多个 HLR，本文将该策略称为基于簇的多 HLR 位置更新策略。其更新原则如下：

只在用户服务 VLR 的归属簇改变时，发起一次 VLR 到 HLR 的位置更新操作；否则，只进行簇内位置更新，采用指针转发策略，不必发起 VLR 到 HLR 的位置更新操作。

下面分别描述簇间位置更新流程和簇内位置更新流程。

3.3.3.1 位置更新流程

根据用户服务 VLR 的归属簇改变时刻，可以将簇间位置更新分为两种情况：

(1) VLR 对用户服务期间归属簇不变

VLR 对用户服务期间归属簇不变，这种情况下发生簇间位置更新主要是由于为用户服务的 VLR 发生了变化，且该 VLR 的归属 HLR 发生了变化。

当用户进入一个卫星管理簇时，选择第一个为它服务的 SAT/VLR 向本簇的簇管理者即 HLR 发起位置更新，更新策略采用簇内更新流程。若为用户服务的下一个 SAT/VLR 仍然处于本簇内，则进行簇内位置更新。若为用户服务的下一个 SAT/VLR

进入一个新簇时，则当前 SAT/VLR 向新的簇管理者即 SAT/HLR 发起位置更新，之后重复上述过程。下面采用一个具体例子来说明其详细过程，如图 3.3-4 所示。

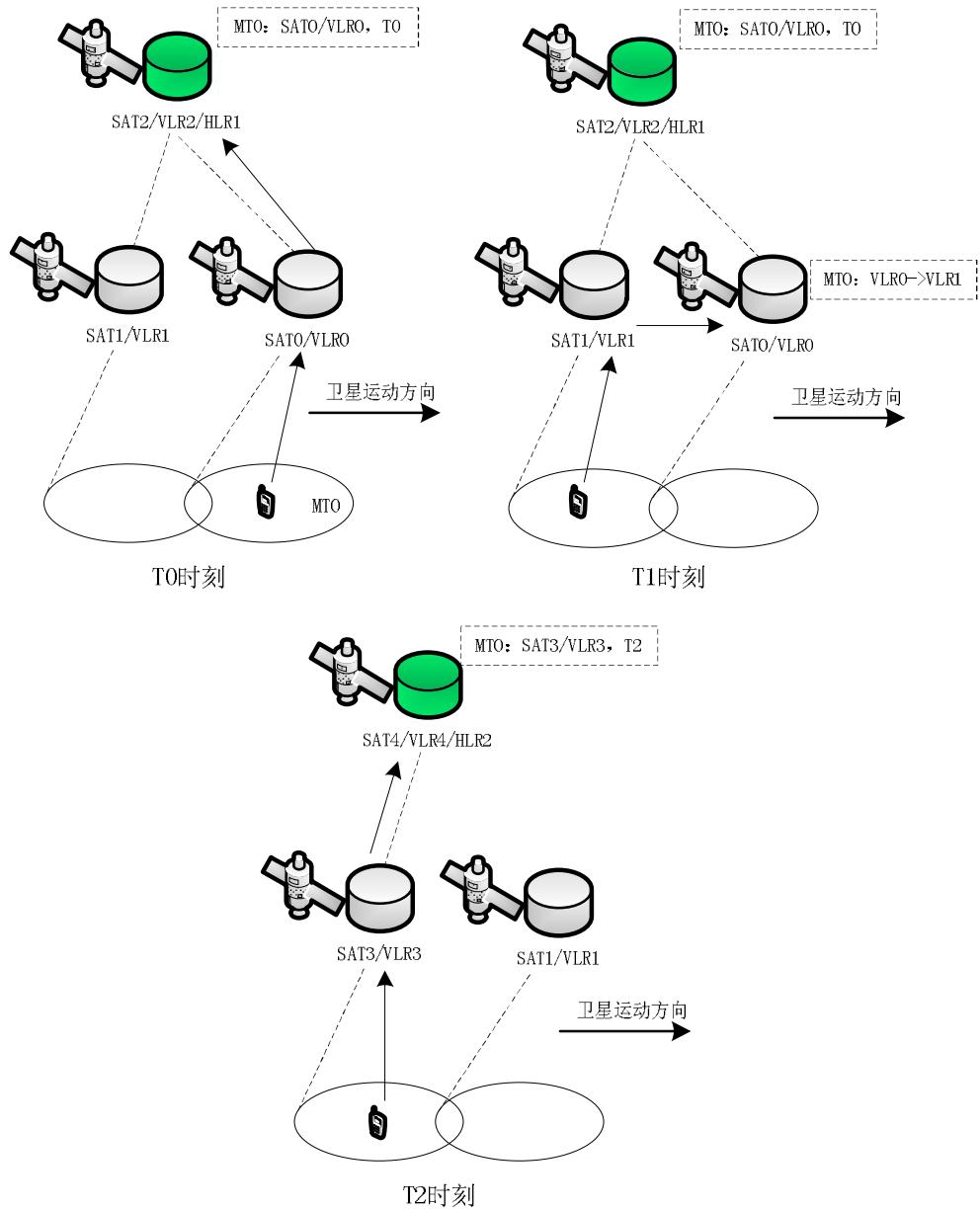


图 3.3-4 VLR 归属簇不变时的簇间位置更新流程

进行簇间更新的前提是，用户收到的卫星信息中包含该卫星当前时刻所属管理簇的情况，基于此，用户才能告知卫星网络它前一时刻进行位置更新的 SAT/VLR 归属簇，卫星网络据此可判断是否需要进行簇间更新。

以图 3.3-4 为例，簇间更新流程如下：

a) T0 时刻，用户 MT0 首次位置更新

T0 时刻，用户被 SAT0/VLR0 覆盖，MT0 进行位置更新，SAT0/VLR0 向 HLR1 发送位置更新；

b) T1 时刻，用户 MT0 的服务 SAT/VLR 改变，归属 HLR 不变

T1时刻，随着卫星的移动，用户 MT0 被 SAT1/VLR1 覆盖，但是 SAT1/VLR1 和前一时刻为 MT0 服务的 SAT0/VLR0 同属一个 HLR 管理，即 MT0 仍然处在簇 HLR1 中，此时则只需进行簇内更新即可，而无需再向 HLR 发起位置更新；

c) T2时刻，用户 MT0 的服务 SAT/VLR 改变，归属 HLR 同时改变

T2时刻，用户 MT0 被 SAT3/VLR3 覆盖，同时 SAT3/VLR3 的归属管理簇变为 HLR2，那么此刻 SAT3/VLR3 就需要向新的 HLR 即 HLR2 发起位置更新；

(2) VLR 对用户服务期间归属簇改变

随着卫星网络拓扑的变化，为用户服务的 SAT/VLR 会离开原来的簇加入新的簇，也就是说 SAT/VLR 的归属簇即 SAT/HLR 发生变化，但在这个过程中为用户服务的 SAT/VLR 仍旧是当前的 SAT/VLR 没有发生变化，这种情况下用户不必重新发起位置更新，只需由当前的 SAT/VLR 发起一次簇间位置更新即可，如图 3.3-5 所示。

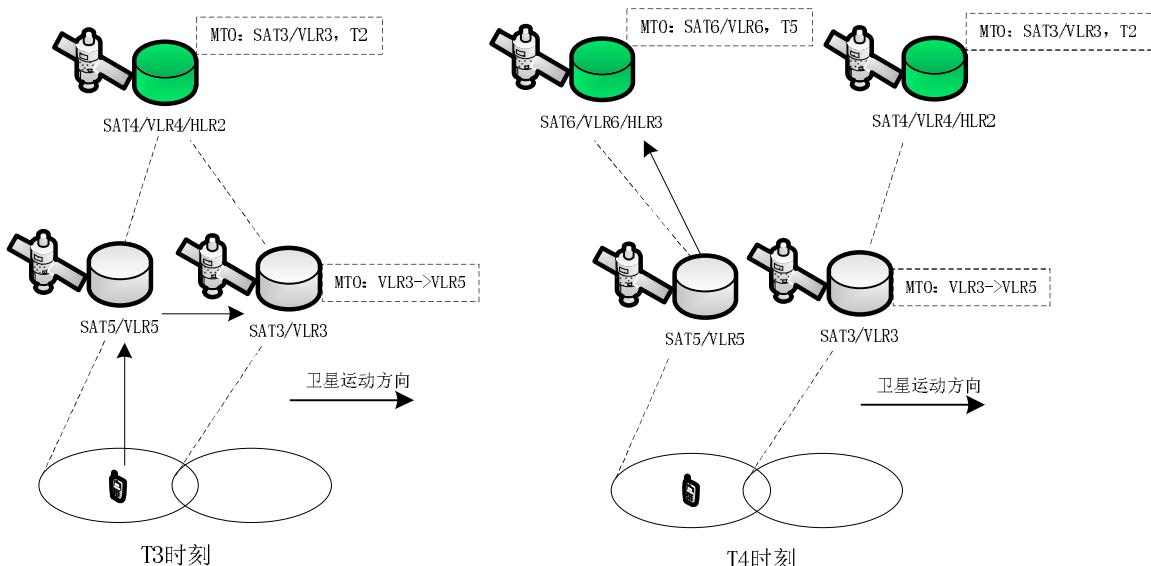


图 3.3-5 VLR 归属簇改变时的簇间位置更新流程

描述：

a) T3 时刻，用户 MT0 的服务 SAT/VLR 改变，归属 HLR 不变

T3 时刻，用户 MT0 被 SAT5/VLR5 覆盖，与其在前一时刻即 T2 时刻的 SAT3/VLR3 同属一个 SAT/HLR 即 SAT4/HLR2，因此只需进行簇内更新即可；

b) T4 时刻，用户 MT0 的服务 SAT/VLR 不变，归属 HLR 改变

T4 时刻，用户 MT0 仍然被 SAT5/VLR5 覆盖，但其归属簇发生了变化，其 SAT/HLR 从 SAT4/HLR2 变为 SAT6/HLR3，这时就需要进行簇间位置更新，由 SAT5/VLR5 向 SAT6/HLR3 发起簇间位置更新；

另外由图中可以看出，承担了簇内的 HLR 作用的卫星，同时还作为本地的 VLR 使用，如图中 SAT4 既是网络中的一个 HLR，又是本星上的 VLR，表示为 SAT4/VLR4/HLR2。

3.3.3.2 移动 agent 与指针转发相结合的位置更新策略

在基于簇的星上位置管理架构基础上，进一步分析可知：

1) 由于卫星的快速移动，为用户服务的 VLR 不断变化，若每次变化 VLR 均向管理其的 HLR 进行位置更新，可能会出现大量位置更新信息短时间失效的情况，造成一定的开销浪费；

2) 基于簇结构的 HLR 设置方式下，可采用指针转发策略对簇内用户位置更新情况进行管理，从而降低更新开销。但由于卫星网络中的 VLR 位置是随时间而移动的，因此指针存在失效的风险。

针对上述存在的问题，本文提出一种移动 agent 与指针转发相结合的位置更新策略，利用智能 agent 的移动性，将移动 VLR 上的指针迁移到与 HLR 具有相对稳定连接关系的 VLR 上，从而避免指针失效问题，同时减少了位置更新的网络开销。

(1) 传统的 K 步指针转发位置管理策略

如图 3.3-6 所示，当 MT 进入一个新的 VLR，按照传统策略，它应该发送位置更新消息到 VLR，并由 VLR 将更新消息转发到 HLR。但根据指针转发的思想，只建立一个由旧的 VLR 到目前 VLR 的指针，这样就减少了由 VLR 到 HLR 的长途信令。

当需要寻呼该 MT 时，首先找到该 MT 所属 HLR，然后查找到 HLR 里所记录的旧 VLR，之后再利用旧 VLR 和当前 VLR 之间的指针查找到 MT。同样，在进入下一个新的 VLR 时，也重复此过程，即取消由当前 VLR 发送到 HLR 的位置更新消息，而建立一条上一 VLR 到当前 VLR 的指针。这样重复 K 次，可建立一条 K 步的指针，称为 K 步指针转发策略。而寻呼时，按照这 K 步指针来寻找 MT。当 K 到达一个阀值时，就进行传统位置更新，既删除之前所建立的 K 步指针，发送更新消息到 VLR，然后将位置消息更新到 HLR，之后再重复之前建立 K 步指针的步骤。图中以 K 为 3 为例，当寻呼时，按照指针链表：HLR→VLR1→VLR2→VLR3→VLR4 可以查找到用户。

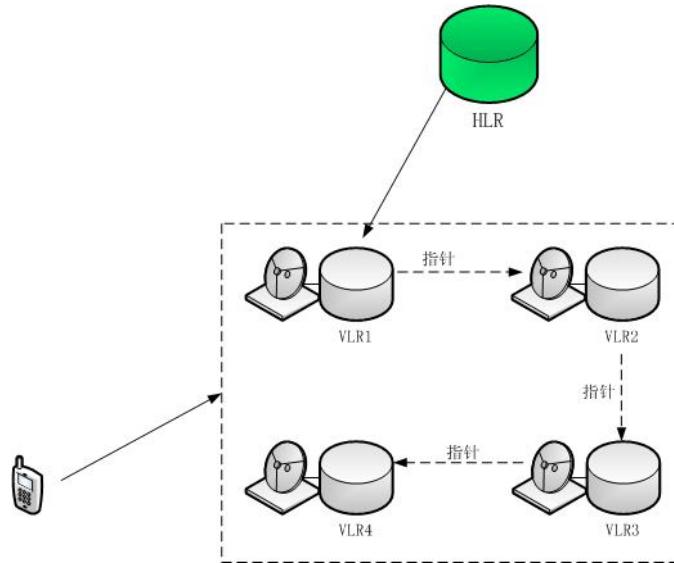


图 3.3-6 K 步指针转发的位置管理策略

从上面的描述可以看出，K 步指针转发策略在一定情况下可以有效减少位置更新的开销，因为没有了频繁的 VLR 到 HLR 的信令传输。但由于指针链的长度，寻呼时延会增大。

(2) 移动 agent 与指针转发相结合的位置更新策略

通过上一小节对指针转发策略的描述，利用指针可以有效减少位置更新的开销，这对于卫星网络有限的带宽资源来说非常适用，但是由于卫星的移动性会对指针的时效产生一定的影响，以图 3.3-6 为例进行分析。

若将所有这些 HLR 和 VLR 搬移到卫星上实现，即 T1 时刻用户在 VLR1 的覆盖下，T2 时刻用户在 VLR2 的覆盖下，T3 时刻用户在 VLR3 的覆盖下，T4 时刻用户在 VLR4 的覆盖下，按照传统指针转发策略，网络建立了“HLR→VLR1→VLR2→VLR3→VLR4”这样的指针链表，但实际上，由于卫星的运动，星上 HLR 和 VLR 的相对位置是在不断变化的，因此可能会导致下述情况：在 T4 时刻建立了 VLR3→VLR4 的指针，然而在时刻 T4 之前由于拓扑的变化，VLR2 与 VLR1 和 VLR3 之间的链路已经断开，此时指针 VLR1→VLR2、VLR2→VLR3 已经失效，若依旧按照这样的链表对用户进行位置查询显然无法完成任务。

另一方面，星间链路的建立与卫星之间的距离、俯仰角、方位角、相对运动速度均有关系，需要满足一定条件才能建立链路，因此接力为用户服务的两个 VLR 之间并不一定存在星间链路，可能无法直接建立它们之间的指针，需通过其他节点进行路由，这样同样会引起信令的长距离传输，失去指针转发策略的意义。

通过上述例子分析表明，由于卫星之间的相对位置变化，导致传统指针转发策略无法适应星座网络。为了解决这个问题，可利用本文第二章中提出的簇结构特点即“同轨链路节点之间的相对位置是稳定不变的”来对指针转发的策略进行改进。

根据 2.4 节中对星座网络的分簇规则和簇划分的结果，每个簇由簇管理者、簇内核心节点、簇内一般节点组成。这里同样基于这样的簇结构对 HLR 和 VLR 进行设置，它们之间的关系如图 3.3-7 所示。

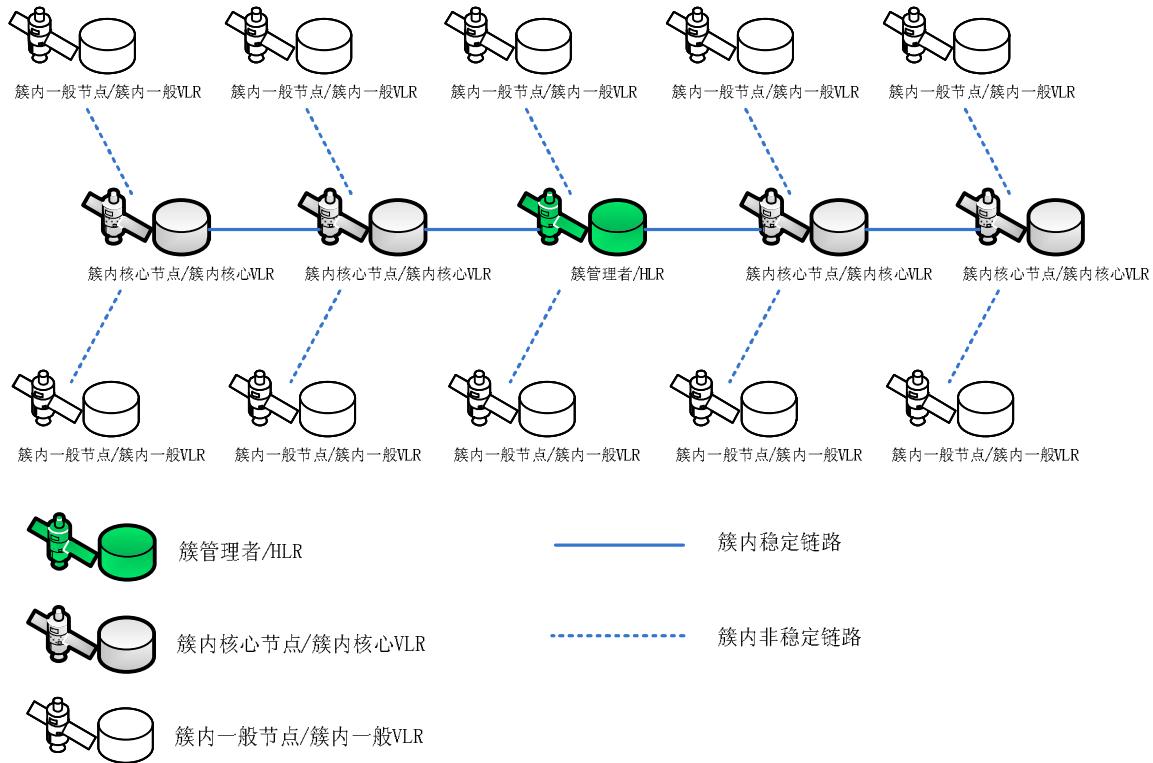


图 3.3-7 基于簇的 HLR 和 VLR 设置情况

簇管理者作为簇内的 HLR，同时也是核心 VLR，簇内其它核心节点也作为核心 VLR，簇内一般节点作为一般 VLR，由 2.4 节可知，它们之间具备以下关系：

- 核心 VLR 与 HLR 的相对位置关系是稳定不变的；
- 一般 VLR 与 HLR 的相对位置关系是随网络拓扑的变化而变化的；
- 一般 VLR 与相邻核心 VLR 之间存在一条星间链路；

由此可知，前述的指针失效问题主要存在于一般 VLR 与核心 VLR 之间，而核心 VLR 与 HLR 之间的指针则不会失效。

针对这一特点，本文提出建立簇内指针链表的原则： $\text{HLR} \rightarrow \text{核心 VLR} \rightarrow \text{一般 VLR}$ 。即当采取多步指针转发策略时，HLR 总是首先指向核心 VLR，再通过核心 VLR 指向一般 VLR。

为了实现这一思路，本文提出利用移动 agent 来建立指针、更新指针以及消解指针等。针对位置管理的问题，移动 agent 具有几个优点：

- 能够自由地在节点之间移动，访问不同节点上 VLR 的信息；
- 能够根据获得的知识，自主决定下一步动作；
- 可以大幅减少网络上的信令开销；

结合移动 agent 的优点，本文提出一种移动 agent 与指针相结合的位置管理策略，保证簇内指针的有效性。

发起簇间更新的条件是：当前覆盖用户的 SAT/VLR 的归属 HLR 发生改变。满足该条件，则启动簇间位置更新。

发生簇内更新的条件是：一是当前覆盖用户的 SAT/VLR 发生了改变，二是 SAT/VLR 的归属 HLR 未发生改变。二者同时满足，则启动簇内更新。

为了有效地建立指针，用户与卫星之间的消息交互必不可少。在星座网络自主运行时，用户收到卫星的广播消息中包含了本卫星的簇特性，包括本星在簇中的成员类型、所属簇管理者、相邻核心成员等等，如下所示：

本星编号	簇成员类型	当前归属簇管理者编号	相邻核心成员编号
------	-------	------------	----------

簇成员类型指簇管理者、簇内核心成员、簇内一般成员三种类型。当簇成员类型为簇管理者时，“相邻核心成员编号”字段无意义。

用户发起的位置更新消息应包含如下内容：

用户号码	当前服务SAT/VLR编号	波束号	上一个服务SAT/HLR编号	上一个核心服务SAT/VLR编号	更新时刻
------	---------------	-----	----------------	------------------	------

用户根据收到的卫星广播消息，可以获知当前服务 SAT/VLR 编号，若与记录中的当前服务 SAT/VLR 编号不同，则用户发起位置更新消息。位置更新消息的字段中，“上一个服务 SAT/HLR 编号”为记录中“当前归属簇管理者编号”，“上一个核心服务 SAT/VLR 编号”则需根据上一个服务 SAT/VLR 的相关信息而定，若上一个服务 SAT/VLR 为簇内一般成员，则其记录中的相邻核心成员便是上一个核心服务 SAT/VLR，若上一次服务 SAT/VLR 为簇内核心成员，则其本身便是上一次核心服务 SAT/VLR。该字段决定了移动 agent 的迁移路径。

该策略描述如下：

一、 簇间位置更新

覆盖用户 MT 的 SAT/VLR 收到 MT 的位置更新消息，根据消息内容可判断 SAT/VLR 的归属 HLR 与前一为用户 MT 服务的 HLR 是否一致，若不一致，则进行簇间位置更新。或者，当前为用户 MT 服务的 SAT/VLR 在服务期间其归属 HLR 发生了改变，也进行簇间位置更新。

- (1) SAT/VLR 首先在数据库中建立或更新用户 MT 的位置更新记录，然后产生一个移动 agent，其目的地址为 SAT/VLR 的当前归属 HLR；
- (2) 移动 agent 的迁移路线依据节点路由表，同时，下一跳节点限定在簇内核心节点中进行选择；
- (3) 移动 agent 记录每个经过的节点 VLR 编号，将其压入堆栈，并携带前行，源 VLR 记为第一个，当前经过的 VLR 为末尾一个；

(4) 移动 agent 经过 VLR 时, 访问本地数据库, 检查核心 VLR 上是否存在属于 MT 的指针或位置更新记录, 若无则核心 VLR 在本地建立“本 VLR→末尾 VLR”的指针, 并记录 agent 本次迁移路径; 若有则删除旧的记录或指针, 建立新的指针;

(5) 移动 agent 经过 HLR 时, 检查其上是否存在属于 MT 的指针或位置更新记录, 若无则建立新的一条用户 MT 的位置信息: (末尾 VLR, 用户 MT 发送位置更新消息的时刻), 若有则更新用户 MT 的位置信息; 更新结束, 移动 agent 销毁;

二、簇内位置更新

覆盖用户 MT 的 SAT/VLR 收到 MT 的位置更新消息, 根据消息内容可判断 SAT/VLR 的归属 HLR 与前一为用户 MT 服务的 HLR 是否一致, 若一致, 则进行簇内位置更新。

(1) SAT/VLR 产生一个移动 agent, 其目的地址为上一次核心服务 SAT/VLR; 若 SAT/VLR 与上一次核心服务 SAT/VLR 为同一 VLR, 则删除旧的 MT 指针, 建立新的 MT 的位置更新记录, 本次更新结束;

(2) 同上(2);

(3) 同上(3);

(4) 移动 agent 经过核心 VLR 时, 检查核心 VLR 上是否存在属于 MT 的指针;
 ①若无则核心 VLR 在本地建立“本 VLR→末尾 VLR”的指针;
 ②若有则进一步检查前次 agent 迁移路径中是否包含上一次核心服务 SAT/VLR;
 ➤ 若不包含, 则删除旧指针, 建立新指针, 继续前行, 直到到达上一次核心服务 SAT/VLR, 移动 agent 销毁, 本次更新结束;

若前行过程中经过 HLR 时, 更新用户 MT 的位置信息内容, 不改变“用户 MT 发送位置更新消息的时刻”, 移动 agent 销毁, 本次更新结束;

➤ 若包含, 则删除旧指针, 建立新指针, 移动 agent 销毁, 本次更新结束;

下面通过图 3.3-8~

图 3.3-9 来说明该策略的执行流程。

如图 3.3-8 所示, 假设在 t0 时刻, SAT6/VLR6 服务用户 MT, 其归属 HLR 为 SAT3/HLR1, 与前次服务 HLR 对比发生了改变, 则根据上述策略, 指针建立的步骤如下:

- a) SAT6/VLR6 发送一个移动 agent, 目的地为 SAT3/HLR1;
- b) 移动 agent 经过 SAT2/VLR2 时, 在其上建立 MT 的指针 VLR2→VLR6;
- c) 移动 agent 继续前进至 SAT3/HLR1, 在 HLR1 上更新用户 MT 的位置信息: (VLR2, t0), 表示用户 MT 在本簇的首次位置更新时刻为 t0, 为其服务的核心 VLR 为 VLR2, 更新结束, 移动 agent 销毁;

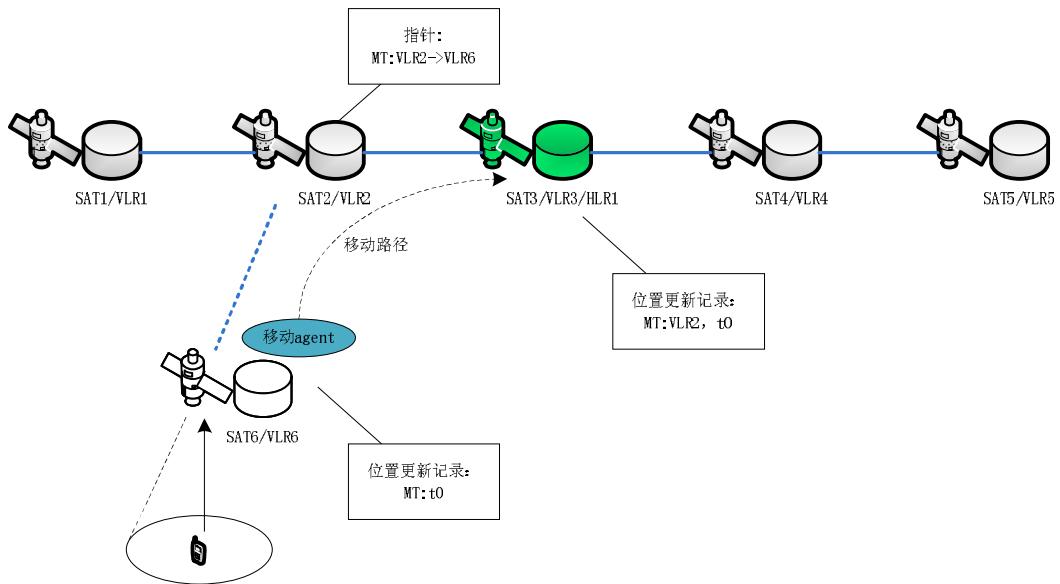


图 3.3-8 服务 HLR 发生变化时指针建立过程

如

图 3.3-9 所示，假设在 t_1 时刻，SAT7/VLR7 收到用户 MT 发起的位置更新消息，其归属 HLR 为 SAT3/HLR1，与 t_0 时刻相比没有发生改变，则建立指针和删除指针的步骤如下：

- a) SAT7/VLR7 发送一个移动 agent，目的地址为上一核心服务 VLR，即 SAT2/VLR2；
- b) 移动 agent 经过 SAT1/VLR1 时，检查其上是否存在 MT 的记录或指针，查询结果为无，则在其上建立 MT 的指针 $VLR1 \rightarrow VLR7$ ；
- c) 移动 agent 继续前进至 VLR2，检查 VLR2 上是否有 MT 的位置更新记录，查询结果为有，进一步检查前次 agent 迁移路径中是否有上一核心服务 VLR 即 SAT2/VLR2，查询结果为有，则修改旧的记录为新的指针 $VLR2 \rightarrow VLR1$ ，且本次移动 agent 迁移目的地为 VLR2，因此更新结束，移动 agent 自行销毁；

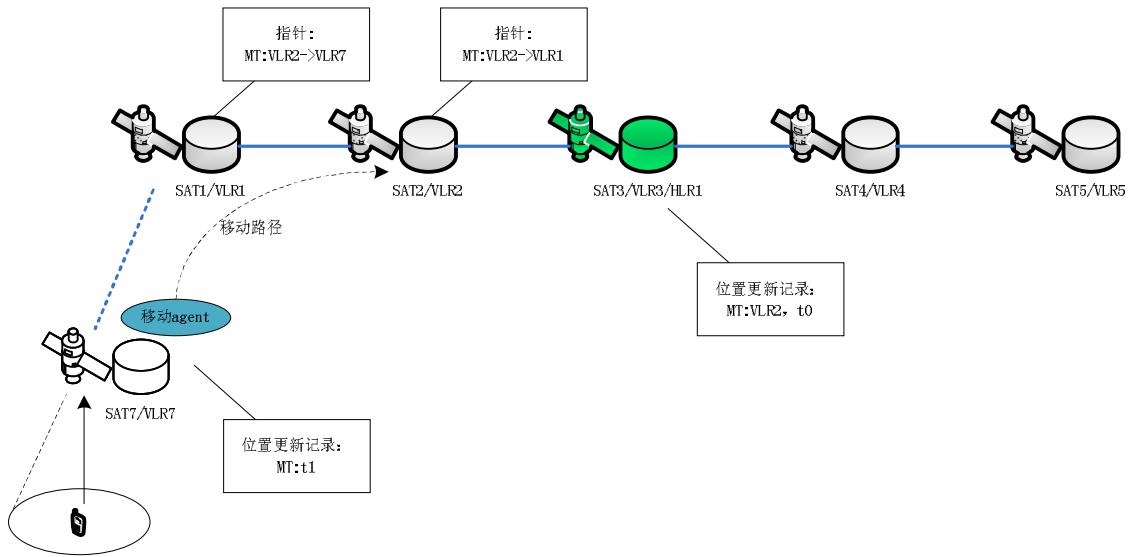


图 3.3-9 服务 HLR 不发生变化时指针建立过程

由以上过程可以看出，在进行簇内一般更新时，移动 agent 在向 HLR 行进的过程中发现经过的核心 VLR 上存在用户 MT 的位置更新记录，则无需再向 HLR 进行更新，从而减小了更新代价。

3.3.3.3 位置管理数据库的更新与维护

从前述的位置更新策略看，当在进行簇间位置更新时，只是向新的 HLR 进行了位置登记，并未对旧的簇内的 HLR 上用户的位置记录以及 VLR 上旧的指针进行删除处理；同时在进行簇内更新时也只是在经过的 VLR 上进行指针或记录的更新，本次更新中没有涉及的 VLR 上仍然存在有旧的指针或记录。由此可见，经过一段时间的运行，网络中的 HLR 以及 VLR 上均会存在较多无效更新记录，导致数据库越来越庞大，在有限的星上存储空间条件下必然导致溢出。因此，必须对无效的过期指针及过期的位置更新记录进行定期或不定期的删除。

根据本文的位置更新策略，通过簇内的指针，总能正确地查询到当前的用户位置，而不会受到无效指针的影响。然而，虽然正确性不会受到影响，但由于不同的 HLR 可能都保留了同一用户在不同时刻的位置更新记录，如果不对无效的记录进行删除，则呼叫某用户时，均需查询所有 HLR，导致呼叫时延增加。为了减小呼叫时延，也可使得网络内同一时刻只存在一个 HLR 中包含用户位置更新记录，这要求实时地对旧记录进行删除，需要额外的更新开销和更新时延，同时，更新时延可能会导致查询错误。因此本文采用 3.3.4 节中双簇寻呼的方式，使得在多个 HLR 用户记录的条件下缩短寻呼时延，同时 HLR 中旧记录的删除可延时进行。

位置更新过程中，任一卫星节点都可能成为用户的服务 VLR，均会存储用户当前的位置更新记录，而指针的保存则只限于簇内核心节点上，HLR 上还需保存簇内用户的本次入簇位置更新记录。这三种用户记录分别如下：

所有 VLR 上的用户当前位置更新记录：

(VLR, 用户 MT 发送位置更新消息的时刻)

所有 HLR 上的用户本次入簇位置更新记录：

(VLR, 用户本次入簇时刻)

所有核心 VLR 上的指针：

本 VLR → 新的 VLR

对于不同的用户记录类型，数据库的存储方式和处理方式不同：

(1) 所有 VLR 上，用户位置更新记录的删除条件：(当前时刻-用户 MT 发送位置更新消息的时刻) $\geq \alpha T_{prd}$ (T_{prd} 为一个轨道周期时间)；

(2) 所有 HLR 上，用户本次入簇位置更新记录的删除条件：(当前时刻-用户本次入簇时刻) $\geq \beta T_{prd}$ (T_{prd} 为一个轨道周期时间)；

(3) 所有核心 VLR 上，指针的删除条件：按照存储顺序定期删除，先入的先删。

从 VLR 开始工作起，每隔 γT_{prd} 时间删除 N 条指针 (T_{prd} 为一个轨道周期时间)。

3.3.4 多 HLR 协作的位置查询策略

当一个呼叫到达卫星时，网络需要进行位置查询。位置查询过程与位置更新的策略紧密相连，由上述位置更新过程可知，在一个簇内进行位置查询相对比较容易，通过查询被呼叫 MT 所在 HLR，可以得知被呼叫 MT 的第一个服务 SAT/VLR，然后通过其上存储的指针，即可找到被呼叫 MT。

另一方面，从上述位置更新过程的描述看，当寻呼某个用户时，网络需要通过 HLR 查询到用户的当前 VLR，但是可能由于旧的数据库尚未更新，用户的位置更新记录可能在多个簇中存在；另外由于存在多个 HLR，位置查询并不能一步到位查询到用户当前所属 HLR，轮询查找多个 HLR 的时延又过长，为此，本文提出多 HLR 协作的位置查询策略。

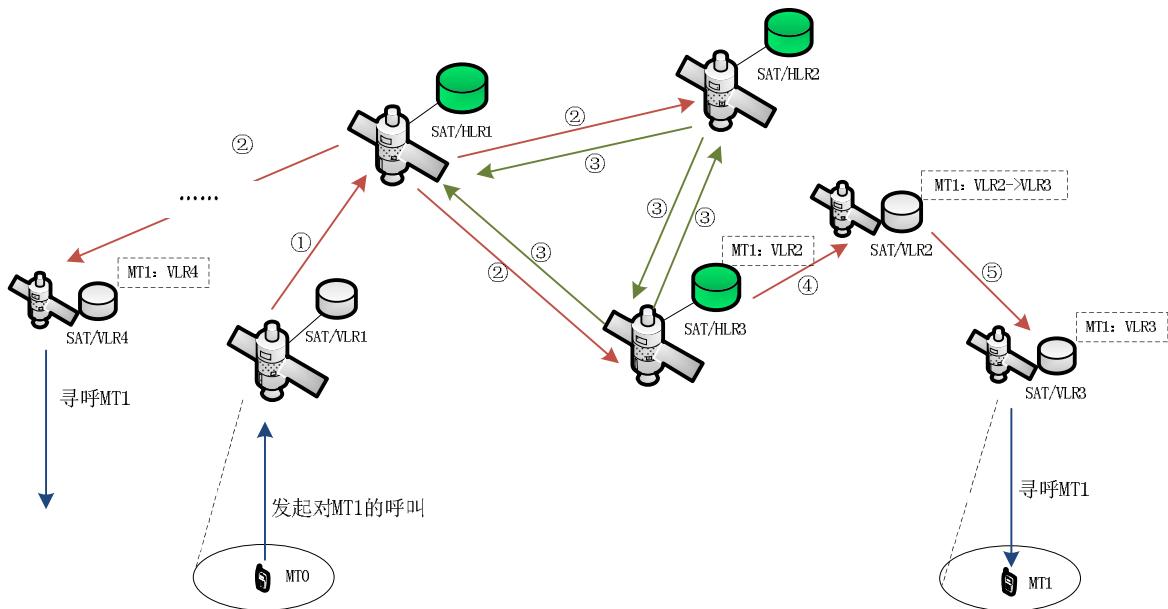


图 3.3-10 多 HLR 协作的位置查询策略

当一个移动用户发起一次呼叫时，卫星网络通过位置查询确定当前为被叫用户服务的 SAT/VLR，从而对被叫用户发起寻呼。本文着重解决网络侧的位置管理问题，为了描述方便，本文定义位置查询的过程为：从主叫服务 SAT/VLR 收到一次新的呼叫开始，到查询到被叫服务 SAT/VLR 结束。

如图 3.3-10 所示，用户 MT0 发起对 MT1 的呼叫，当前为 MT0 服务的 SAT/VLR 为 SAT/VLR1，位置查询过程如下所示：

- (1) SAT/VLR1 发送请求位置查询消息到其所属的簇管理者即 SAT/HLR1；
- (2) SAT/HLR1 发送请求位置查询协作消息分别到网络中另外两个簇管理者即 SAT/HLR2 和 SAT/HLR3，其中该消息中包含 HLR1 中查询到的 MT1 的位置信息，表明 MT1 在 HLR1 中的位置更新时间；
同时，SAT/HLR1 在本簇内查询 MT1 的位置更新记录，若存在，则发送寻呼消息到 MT1 在本簇内查询到的当前服务 SAT/VLR；
- (3) SAT/HLR2 和 SAT/HLR3 收到位置查询协作消息后，同时向其它 HLR 发送本地位置查询结果，表明 MT1 在本 HLR 中的位置更新时间；
- (4) 各 SAT/HLR 比较网络中所有的 HLR 中 MT1 的位置信息，若本 HLR 中 MT1 的位置更新时间距离当前时间最近，则表明其为当前 MT1 的服务 SAT/HLR，图 3.3-10 中 SAT/HLR3 为 MT1 的当前所属的 HLR，说明为 MT1 服务的 SAT/VLR 处在 SAT/HLR3 的管理之中；
- (5) SAT/HLR3 发送寻呼消息到被叫用户 MT1 在本簇内第一个服务 SAT/VLR 即 SAT/VLR2，SAT/VLR2 则根据存储的指针链将该寻呼消息发送到 MT1 的当前服务 SAT/VLR 即 SAT/VLR3；

3.4 性能分析

本文提出的位置管理策略可以解决星座网络自主运行时的用户位置管理问题，同时能够在卫星网络资源有限的条件下降低管理开销，缩短寻呼时延。下面本文对位置管理开销和寻呼时延性能进行理论分析。

位置管理开销包含位置更新开销和位置查询开销。

下述分析中的参数设定如表 3.4-1 所示。

表 3.4-1 参数设定一览表

符号	意义
i	当前 VLR 与 HLR 之间的跳数
p(i)	用户的服务 VLR 与 HLR 之间的跳数为 i 的概率
m _i	与 HLR 之间的跳数为 i 的 VLR 上存在的用户 MT 记录
p(m _i =0)	与 HLR 之间的跳数为 i 的 VLR 上不存在用户 MT 记录的概率
O ₁	移动 agent 在 1 跳内的传输开销
O _{cc(i)}	用户的服务 VLR 与 HLR 之间的跳数为 i 时的簇间位置更新开销
O _{cn(i)}	用户的服务 VLR 与 HLR 之间的跳数为 i 时的簇内位置更新开销
O _{cc}	簇间位置更新开销
O _{cn}	簇内位置更新开销
O _{hh(m,n)}	HLRm 到 HLRn 的传输开销
O _{inq}	位置查询开销
Δ T	1 跳传输时延均值
H _{hh(m,n)}	HLRm 到 HLRn 的连接跳数
T _{pag}	网络寻呼时延
P _{sh}	主叫用户与被叫用户处于同一 HLR 管理下的概率

根据 3.3.3.2 节中的策略描述，一次簇间位置更新的开销可表示为：

$$O_{cc}(i) = O_1 \cdot i \quad (式 3.4-1)$$

一次簇内位置更新的开销可表示为：

$$O_{cn}(i) = O_1 \cdot p(m_i = 0) + O_1 \cdot p(m_{i-1} = 0 | m_i = 0) + \dots + O_1 \cdot p(m_1 = 0 | m_2 = 0) \quad (式 3.4-2)$$

由 (式 3.4-1)，总的簇间位置更新开销可表示为：

$$O_{cc} = \sum_i \{p(i) \cdot O_{cc}(i)\} = \sum_i \{p(i) \cdot O_1 \cdot i\} \quad (式 3.4-3)$$

由(式3.4-2),总的簇内位置更新开销可表示为:

$$O_{cn} = \sum_i \{p(i) \cdot O_{cn}(i)\} = \sum_i \left\{ p(i) \cdot O_1 \cdot \left[p(m_i = 0) + \sum_{j=2}^i p(m_{j-1} = 0 | m_j = 0) \right] \right\}$$
(式 3.4-4)

位置查询的开销表示为:

$$O_{inq} = 2O_{cn} + \sum_{m, n} O_{hh}(m, n)$$
(式 3.4-5)

寻呼时延可表示为:

$$T_{pag} = p_{sh} \cdot \Delta T \cdot 2 \sum_i p(i) \cdot i + (1 - p_{sh}) \cdot \Delta T \cdot \left\{ 2 \sum_i p(i) \cdot i + 2E[H_{hh}(m, n)] \right\}$$
(式 3.4-6)

3.5 仿真与分析

本文的位置管理策略基于分布式的数据库结构,其特点主要体现在下面两个方面:

- 采用多 HLR 处理;
- 采用移动 agent 与指针转发相结合的 HLR 到 VLR 的寻址方式;

针对这两点改进,本文进行了仿真,并与传统两级 HLR/VLR 位置管理策略在位置更新开销、寻呼开销、寻呼时延性能上进行了比较。

(1) 位置更新开销

针对星座网络自主运行状态下,某固定用户在 10 小时内网络侧的位置更新开销性能仿真结果,如图 3.5-1 所示。这里,位置更新开销以 O_1 为单位,表示“经过 1 跳传输所需开销”,对于本文的位置更新策略来说, O_1 指“一个移动 agent 所需传输的字节数”,对于传统两级 HLR/VLR 位置管理策略来说, O_1 指“一条位置更新消息所需的字节数”,由于移动 agent 的内容主要有用户号码、目的地址、本次位置更新时刻、历史路径等,与传统的位置更新消息开销基本相同,因此这里采用相同的单位量来表示总的位置更新开销。

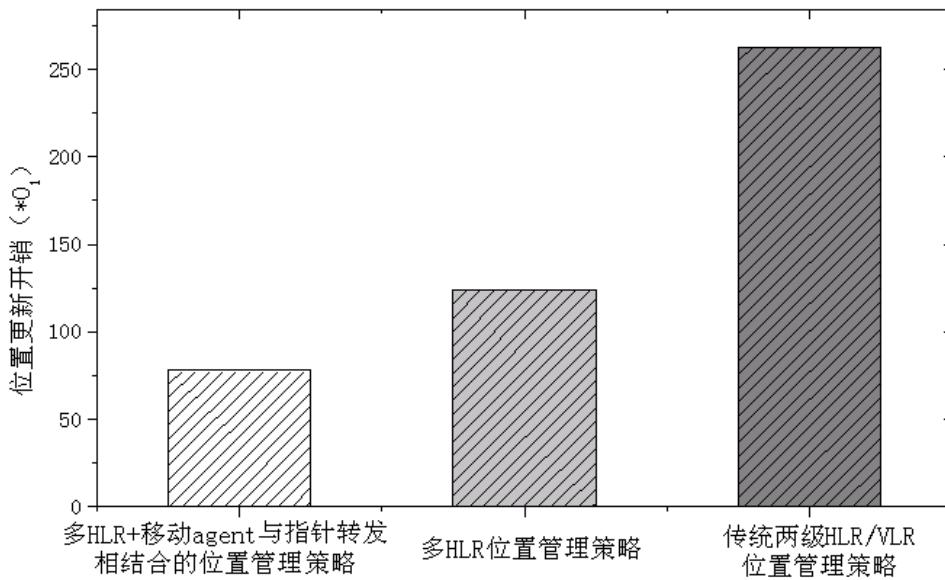


图 3.5-1 位置更新开销性能仿真比较结果

从仿真结果来看，传统两级 HLR/VLR 位置管理策略的位置更新开销比较大，采用多 HLR 位置管理策略，位置更新开销明显优于传统策略，而在多 HLR 基础上再采用移动 agent 与指针转发相结合的策略，则能进一步降低位置更新开销。这是因为传统两级 HLR/VLR 位置管理策略采用的是集中式管理方式，每一次当用户的服务 VLR 发生改变时，VLR 均需向中央 HLR 进行位置更新消息的传递，常常需经过多跳转发和频繁更新。而多 HLR 位置管理策略采用分布式管理方式，通过设置多个 HLR 来减少 VLR 向 HLR 传递位置更新消息的跳数，从而降低全局开销。在多 HLR 基础上再采用移动 agent 与指针转发相结合的簇内位置更新策略，使得服务 VLR 的归属簇没有发生变化时，不必将位置更新消息传递到最终的 HLR，而是利用指针的方式将消息传递到核心服务 VLR 即可，因此能够进一步降低开销，不过由于移动 agent 参与了指针的转移，后者对性能的提高贡献有限。仿真结果基本与前述章节的分析吻合，并验证了本文策略的有效性。

(2) 位置查询开销

图 3.5-2 为采用不同策略时网络中一次位置查询的平均开销，仿真时由呼叫用户与被叫用户的距离随机选取。

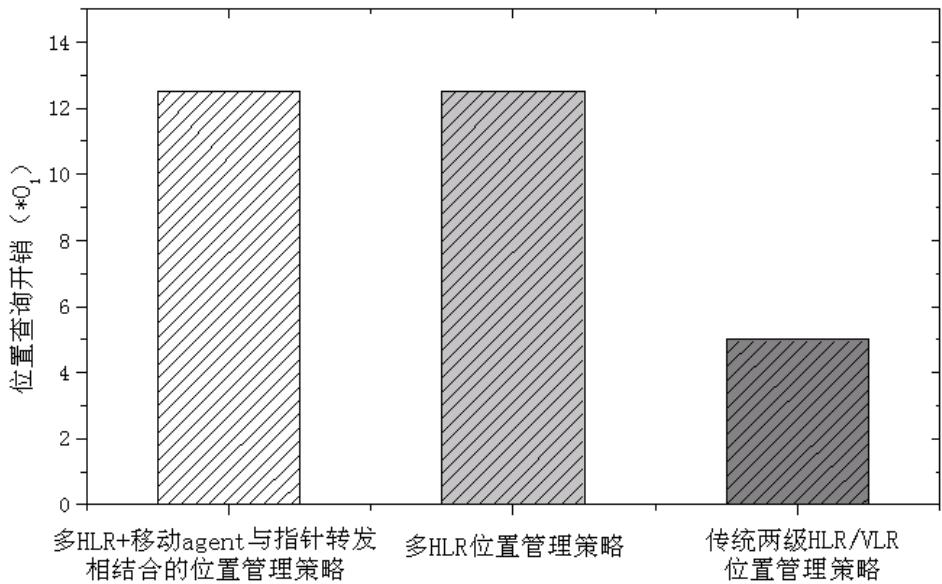


图 3.5-2 位置查询开销性能仿真比较结果

在位置查询开销上，采用多 HLR 的两种位置管理策略，利用指针进行簇内位置查询所经历的路径与一般的 HLR 到 VLR 的位置查询路径相同，因此两者的位置查询开销也相同，而传统两级 HLR/VLR 位置管理策略，由于其查询不必经由多个 HLR 的信息交换过程，因此其开销相对前两者较小。

(3) 寻呼时延

图 3.5-3 为不同用户距离下的平均寻呼时延仿真结果。由于网络侧的寻呼时延与位置查询代价成正比，根据上述结果，这里只对传统两级 HLR/VLR 位置管理策略、多 HLR+ 移动 agent 与指针转发相结合的策略进行比较。

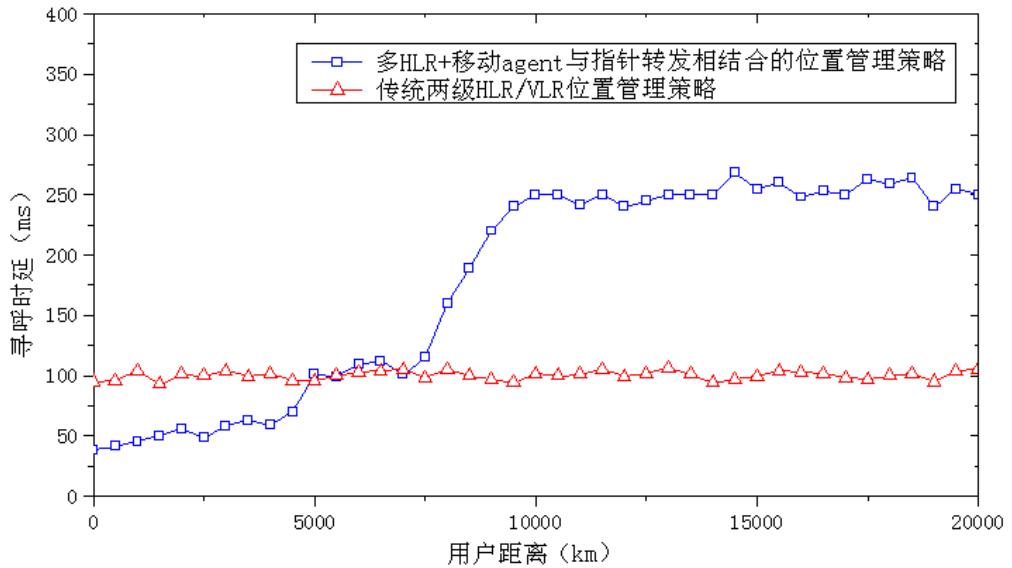


图 3.5-3 不同距离的用户之间网络平均寻呼时延仿真结果

由图可以看出，在两用户距离较近时，本文策略明显优于传统两级 HLR/VLR 位置管理策略。根据本文的寻呼策略，当用户同属一个簇管理时，寻呼时延较小，而分

属不同簇管理时，寻呼时延较大，仿真结果也表明了这一点。仿真中采用的多波束天线模型如图 3.5-4、图 3.5-4 所示。

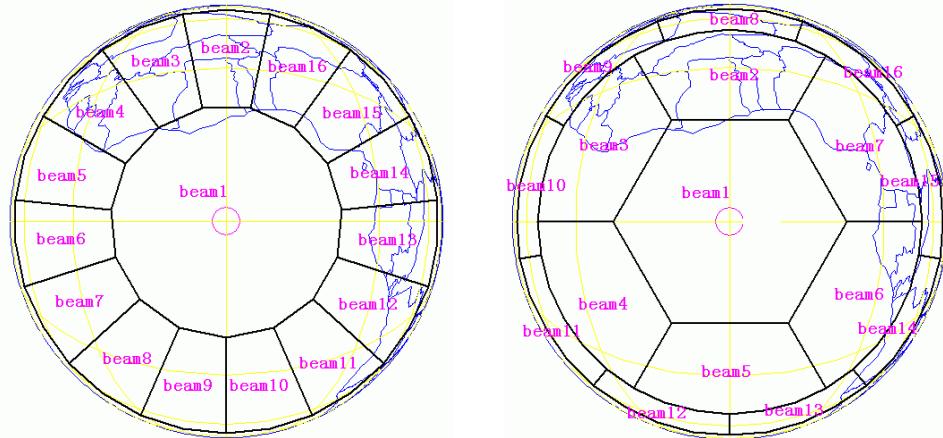


图 3.5-4 卫星多波束接收天线赋形图 图 3.5-5 卫星多波束发射天线赋形图

接收天线波束覆盖范围 26° 、 53° ，发射天线波束覆盖范围 26° 、 46° 、 53° 。求得接收天线波束的地表覆盖跨度(从卫星下点向外辐射方向)分别为 729Km、2256Km；发射天线波束的地表覆盖跨度分别为 729Km、1051Km、1205Km。据此可计算整个卫星波束对地表覆盖跨度约为 3700km 左右。根据本文的星座模型，两个不同卫星的波束重叠范围约在 1000km 左右。本文对不同距离的两个用户被卫星覆盖的情况进行了统计，如表 3.5-1 所示。这里，用户位置随机选取多组进行统计，概率取平均值。

表 3.5-1 任意两个不同距离用户被卫星覆盖情况统计表

两用户距离	两用户被同一卫星覆盖的概率	两用户服务 VLR 属同一 HLR 的概率
0~2000km	85%	98%
2000~4000km	48%	91%
4000~7000km	0	91%~56%
7000~10000km	0	56%~5%
10000~20000km	0	5%

对仿真结果分析表明，当两用户相距在 2000km 以内时，其平均寻呼时延均在 50ms 左右，从上表可知这时两用户处于同一卫星覆盖范围内的概率较大，因而其服务 VLR 属同一簇管理的概率也较大，因此其寻呼时延较小。而用户距离在 2000~4000km 之间时，它们被同一卫星覆盖的概率逐渐减小，但由于其距离近，覆盖它们的卫星通常为相邻卫星，故而其服务 VLR 属同一簇管理的概率仍然较大，其平均寻呼时延均在 50ms~70ms 之间。当两用户相距在 4000km 到 7000km 之间时，它们被不同卫星覆盖的概率几乎为 0，并且随着距离增加，它们的服务 VLR 同属一个 HLR 管理的概率逐渐减小，这段距离内，仿真的平均寻呼时延在 100ms 左右，而当距离在 7000~10000km 时，它们的服务 VLR 同属一个 HLR 管理的概率急剧减小，

这也造成在这段距离内仿真的平均寻呼时延急剧增加。当用户距离超过 10000km 之后，它们的服务 VLR 同属一个 HLR 管理的概率非常小，因此其平均寻呼时延约在 250ms。

3.6 小结

本章针对星座网络自主运行的特殊应用背景，对用户的位置管理策略进行了研究。传统的位置管理研究，无论是地面移动通信系统还是星座移动通信系统，其位置管理寄存器均是基于地球固定位置的地面站来实现的。对于本文的研究背景，需要在无需地面站支持的条件下完成整个星座网络的运行管理，这势必要求位置管理由星上自主完成。本章在深入分析了地面移动网络现有的位置管理策略以及低轨通信星座位置管理已有的研究成果的基础上，结合星座网络拓扑特点，提出了基于簇结构的星上自主多 HLR 位置管理策略。该策略主要针对网络侧的位置管理问题，不涉及无线侧的位置更新与寻呼问题，分别描述了簇间和簇内更新流程，尤其是针对簇结构的特点，提出了簇内采用移动 agent 与指针转发相结合的位置更新策略，可以有效减少位置更新的信令开销；同时对这种更新模式下的位置查询策略进行了研究，提出了有效的数据库更新与维护方法。最后通过理论分析和仿真结果证明，本策略可行有效，位置管理开销优于传统策略用于自主运行星座网络时的性能；寻呼时延则在两近距离用户通信时优于传统策略，这也较符合大多用户均为境内用户的实际情况。

第四章 基于 SLD 的分布式网络故障诊断技术研究

4.1 引言

低轨卫星组成的星座系统，其业务主要包括实时语音通信、短数据以及未来可能接入的 internet 服务等，星间链路的加入大大提高卫星网络连通性和灵活性，满足了语音通信的实时性要求，并使未来互联网的既接入成为可能，同时由于星间链路的接入，也使整个网络更为复杂。网络性能的优劣直接影响用户的服务质量，为了提高网络可靠性，必须对卫星网络进行故障管理。由于星座系统包含节点数量众多，而境外建测控站的难度较大，加之星地链路具有一定的时延，使得由地面进行故障管理的方式不能满足网络实时性管理的要求。同时，星座系统的自主运行可以降低卫星系统运行成本、减小系统风险，已经成为一种发展趋势。因此，开展星座自主网络故障管理的研究具有重要意义和应用价值。

系统级诊断是一种有效的故障识别方法。通过识别节点状态，可对故障节点进行隔离，或者对故障链路进行隔离，实现一定程度的网络容错。该方法假设节点间可以进行相互测试，测试结果的集合称为系统症状。节点状态为正常或故障，节点状态集合称为系统状态。诊断即从系统症状判别出系统状态的过程。星上设备受空间环境及辐射的影响较易发生计算错误等软故障，这类故障通常不易通过遥测参数识别出来，但有时又会对业务的正常通信造成极坏的影响。针对这种情况，系统级故障诊断方法是一种比较适合的解决方案。

本文主要以系统级故障诊断理论为基础，针对网络故障的自主检测和识别方法进行了研究，提出一种适合于星上处理的分布式自主网络故障识别算法（DSFD），使得每个节点都可独立进行自识别，而无需获得整网节点状态，这种算法能够大大降低网络故障检测开销，同时缩短诊断时延，并且降低星上计算量，提高可靠性。

4.2 系统级故障诊断算法概述

4.2.1 系统级故障诊断理论

系统级诊断(System-Level Diagnosis, SLD)理论^[65]是由 Preparata, Metze 和 chien 等人于 1967 年首次提出的，为解决一对一通信的有线系统中存在多个故障时的自诊断难题而提出的，后来，这种技术的应用拓展到大规模并行系统、计算机网络和集成电路等实际领域，并且相关研究也在无线通信网络中展开。

系统级诊断的基本原理是假设系统各单元间可以进行互相测试，通过分析测试结果诊断单元的状态。因为测试单元本身也可能是故障的，所以这种单元间的测试结果

可能是不可靠的。因此，必须对测试结果进行分析才能正确地定位故障单元。

系统级诊断通常采用图来建模。设一个诊断系统 S ，用无向图 $G(S)=(V, E)$ 对其进行建模：其中系统节点的集合为 $V=\{v_0, v_1, \dots, v_{N-1}\}$ ，数量为 N ； $G(S)$ 中无向边的集合为 E ，表示系统节点间的互联关系。如果 S 中节点 v_i 和 v_j 之间存在直接的联系或通信路径，则图 $G(S)$ 中存在边 $e_{ij}=v_i v_j \in E$ 。节点 $v_i \in V$ 的状态可以是正常 (Fault-Free) 或故障 (Faulty)，即 $v_i \in \{\text{fault-free (0)}, \text{faulty (1)}\}$ 。

设每个节点都能够对其邻节点（或称为物理邻节点）执行测试。系统中所有节点执行测试的集合称为测试任务，用测试图 $D(S)=(V, T)$ 表示。有向图 $D(S)$ 中边 $t_{ij}=(v_i, v_j) \in T$ 表示 v_i 对 v_j 执行的一个测试。任意节点 v_i 的被测者集合与其测试者集合可以分别用如下集合表示：

$$\Gamma(v_i)=\{v_j | t_{ij} \in T, j=0, 1, 2, \dots, N-1\}$$

$$\Gamma^{-1}(v_i)=\{v_j | t_{ji} \in T, j=0, 1, 2, \dots, N-1\}$$

图 4.2-1 是一个诊断系统 S 的系统图和测试图。

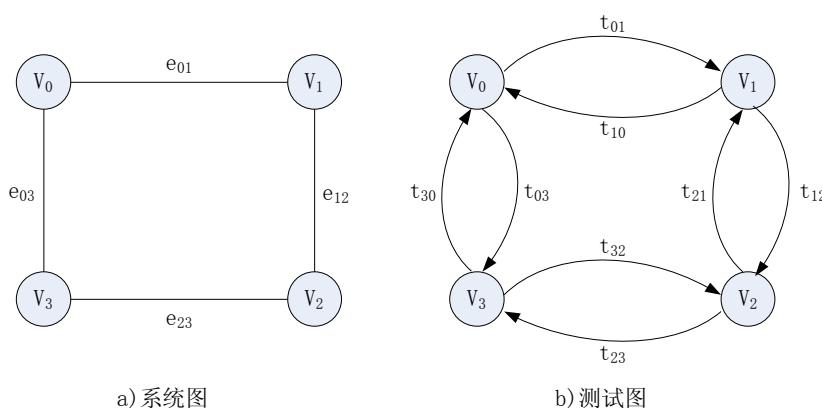


图 4.2-1 系统图与测试图

测试图的边值可表示测试 t_{ij} 的结果。 t_{ij} 的边值可表示为 $a_{ij}=\{0,1\}$ ，当 v_i 状态为正常且 v_j 状态也为正常时 $a_{ij}=0$ ；当 v_i 状态为正常且 v_j 状态为故障时 $a_{ij}=1$ ；当 v_i 状态为故障，无论 v_j 状态是否正常，测试结果均是不可靠的， a_{ij} 可能为 0 也可能为 1。

所有测试结果的集合称为系统的症状 (Syndrome)。与一个故障集 F 对应的症状集包含系统中由于故障集 F 的出现而产生的症状，记为 $S(F)$ 。为了区分正常和故障单元需要对症状进行分析 (deicode)，通过给定症状来识别故障单元集 F 的过程称为诊断 (Diagnosis)。假设 F 中的单元是故障状态，而其他节点是正常状态，这一现状与症状和产生该症状的测试无效模型是一致的。满足这种需求的故障集称为一致性 (consistent) 故障集。

系统级诊断的目的就是识别出这个一致性故障集。为了达到这个目的，系统级诊断需包括三个要素：

(1) 可诊断的条件

当一个系统中出现的故障数量不超过 t 时，系统可以保证能够识别出所有故障节点，那么这个系统被称为是 t -可诊断的 (t -diagnosable)^[67]。一个系统可保证诊断出的最大故障单元数量 t 称为可诊断性的度(degree of diagnosability)。一个 t -可诊断的充分条件： $n > 2t + 1$ ，而且每个节点必须被至少 t 个其他节点测试， n 是系统节点数量。这表明：系统结构的高连通性是实现较高的可诊断性的基础，因为诊断图(即测试图)中节点的最小入度需要大于 t 。

(2) 测试模型

如前所述，在节点间的相互测试中，当测试者为故障时，会产生不确定的测试结果而无法反映被测者的真实状态。即故障会影响测试结果的有效性，这种影响在系统级诊断理论中用测试无效模型 (Test Invalidation Model) 来描述。

一个故障单元测试另一单元所得到的测试结果比较复杂，不同模型对测试结果的解释是不同的。最常用的是对称无效模型(symmetric Invalidation Model)即 PMC^[65]模型和非对称无效模型(Asymmetric Invalidation Model)也称 BGM^[66]模型。在 PMC 模型中：一个正常单元测试另一单元时，如果被测单元为正常则相应结果为 0，如果被测单元为故障则结果为 1。另一方面，当一个故障单元测试另一单元时，相关结果任意，而与被测单元的真实状态无关。该模型也称为悲观模型。PMC 模型和 BGM 模型的区别在于：当测试单元和被测单元都为故障时，后者更为乐观，测试结果始终为 1。

(3) 诊断算法

基于系统产生的给定症状，决定系统中出现的故障集(假定故障集属于允许的故障集合家族)，即设计一个能够从测试结果中识别故障集的诊断算法。

诊断算法根据处理者的不同可分为集中式和分布式。集中式算法通常假设存在一个中心观测者执行诊断，它可以准确地接收所有测试结果，并基于症状诊断出单元状态。分布式则由系统中单元根据对邻单元的测试或从邻单元收集来的其他单元的信息，独立地执行诊断。根据测试任务分配方式的不同可分为固定测试任务或者非固定测试任务，固定测试任务通常是提前确定而且固定不变的，而非固定测试任务可以是随机的或者是自适应的，没有固定模式。

4.2.2 国内外研究现状

前一小节主要介绍了系统级故障诊断理论的概念、基本原理以及其具备的几大要素等。本节主要介绍系统级故障诊断算法在这几个方面的国内外研究现状，主要包括测试模型、处理模式、诊断算法等等。

系统级故障诊断理论在近几十年的发展过程中获得了很多研究成果，从最初的集中式诊断发展为分布式、自适应、在线诊断等多种特性的诊断算法，从固定不变的测试任务发展为自适应的测试任务模式，从固定测试图结构发展为概率测试模型等等。这其中本文重点介绍测试模型的发展和分布式诊断算法的研究现状以及在卫星网络

中的应用。

测试模型除了前一小节介绍的 **PMC** 模型和 **BGM** 模型外, 文献^[67]和文献^[68]分别在 **PMC** 模型和 **BGM** 模型的基础上提出了比较模型 (**Comparison Model**), 这种模型中, 测试图是无向的, 两个单元互相测试结果的比较值即为边值, 0 表示两个测试结果一致, 1 表示不一致。在这两个模型中, 实际上都假设存在一个第三方的比较器来进行结果的对比。文献^[69]对基于 **BGM** 模型的比较模型进行了扩展, 提出将比较器作为一个独立的单元, 模型中故障比较器会对比结果产生影响, 从而使这个比较模型更加具备普遍性。

文献^[70]对提出了基于比较模型的故障诊断算法, 并分别在诊断期间网络拓扑不变和诊断期间网络拓扑变化的条件下对故障诊断的能力进行了仿真比较, 为 **ad hoc** 网络故障诊断提供了可行的手段。

Maheshwari 和 **Hakimi**^[71]提出将概率论与系统级诊断相结合的思想, 为每个单元的可靠性分配一个先验值, 诊断的目标就是找到与症状一致的最可能的故障集。如果系统存在一个唯一的一致故障集, 且发生的概率大于 p , 则将这种系统称为 p - t -可诊断(p - t -diagnosable)系统。

基于概率的测试模型包含了测试者和被测者不同状态时所有可能的测试结果以及获得该结果的概率。文献^[72]提出一个通用的概率测试模型, 实际上是将所有的测试模型综合在一起, 假设无论测试者和被测试者的状态怎样, 所有的结果都以一定的概率出现。如表 4.2-1 通用概率测试模型所示, 其中 fs_i 表示测试者 v_i 的实际状态, fs_j 表示被测试者 v_j 的实际状态, a_{ij} 为 v_i 对 v_j 进行测试的结果, $P(a_{ij}|fs_i, fs_j)$ 表示测试结果为 a_{ij} 的概率。

表 4.2-1 通用概率测试模型

fs_i	fs_j	a_{ij}	$P(a_{ij} fs_i, fs_j)$
0	0	0	q_{ij}
0	0	1	$1-q_{ij}$
0	1	0	p_{ij}
0	1	1	$1-p_{ij}$
1	0	0	r_{ij}
1	0	1	$1-r_{ij}$
1	1	0	s_{ij}
1	1	1	$1-s_{ij}$

文献^[73]将概率测试模型与传统的 **PMC** 模型进行了结合, 提出了概率化的扩展 **PMC** 模型, 并针对不同的概率进行了诊断性能的仿真。

早期的诊断算法假设存在一个诊断中心, 它可以准确地接收所有测试结果, 并基

于全网的症状诊断出单元的状态。通常集中式诊断算法都假设每个节点的测试任务是提前确定而且固定不变的。文献^[74]在集中诊断方式的基础上，提出了分布式系统级诊断，系统中的每个节点通过对邻居节点的测试或通过邻居节点转发的方式获取其它节点的信息。其中，正常节点从邻节点可靠地接收测试结果，当它测试了该邻节点并认为其正常、可信，则信息沿着测试链可靠地传播到正常节点。该文提出的系统级诊断算法是完全分布式的。

上面提到的分布式诊断算法存在有故障识别率低、测试开销冗余大等缺点，为了克服这些缺点，Binahcini 等人提出自适应分布式系统级诊断算法(ADSD 算法)^[75]。该算法使每个节点在一个测试循环里必须仅被测试一次，测试由每个节点自适应发起并依赖于网络的故障情况，每个正常节点都能正确、独立地诊断出其余节点的故障情况。

在系统级故障诊断算法的发展中，ADSD 算法是一个重要的研究成果，基于该算法的扩展研究^{[76][77][78][79][80]}非常多。其中，文献^{[76][77]}提出层次化的 ADSD 算法(Hi-ADSD)，采用分层的方式进行节点间的测试，以减少诊断开销。文献^[78]是 Hi-ADSD 算法基础上提出的分布式故障诊断算法，在分层诊断基础上采用分簇方式缩短了诊断时延、降低了诊断开销。文献^[80]在分层诊断算法的基础上又提出了多层 ADSD 算法(ML-ADSD)，网络中的节点被平均分成 p 个簇组成网络的最低层，每个簇的簇首组成再上一层的簇，依此类推。簇内执行 ADSD 算法。该算法的特点是能够灵活地调整簇内节点数以及层数来改变算法的性能。其仿真分析表明，该算法具备更好的诊断时延性能。

上述诊断算法基于的测试模型主要是 PMC 模型或者类似模型，而基于概率测试模型的故障诊断算法是系统级故障诊断算法的另一大研究成果。文献^[81]提出的概率性算法首次对诊断正确和完全的极限进行了证明，指出系统正确且完全诊断的条件为系统测试数量需满足 $O(N \log N)$ (N 为节点总数)。文献^[82]对该方法进行了改进，使所需测试数量降低到 $O(N)$ 。文献^[83]提出采用多次重复测试的策略获得正确诊断。在每个测试连接上最少执行 $\log N$ 次测试，这种算法是基于投票机制的。文献^[84]定义了三种启发式故障分类方法，并实验比较了它们的诊断完全性和诊断正确性。

从系统级故障诊断算法的发展及研究成果上来看，正确而完全的诊断要求系统尽可能地具备高互联性和低故障率。然而在大多数应用场合，系统的互联结构只能是每个单元被一定数量的邻居单元所测试，为了克服 t -可诊断问题，提高诊断算法的实用性，近似正确和完全的概率性诊断算法成为重点研究的方向。

近年来，随着人工智能技术的发展，研究者利用神经网络来解决系统级故障诊断问题^{[85][86][87][88]}，文献^[85]提出了非对称比较模型，基于无向图测试来识别故障；文献^{[86][87]}均为其延伸研究，文献^[88]则利用神经网络通过部分节点症状进行自诊断，并通过仿真证明了算法对不同规模不同故障场景网络的适应性。

卫星网络的日渐复杂和对故障诊断越来越高的要求使得研究者开始关注系统级故障诊断理论在该领域内的应用。文献^{[89][90][91][92]}针对卫星网络的特点，在故障建模、测试模型、诊断算法等方面进行了研究，为 SLD 在卫星网络中的应用奠定了基础。上述针对卫星网络的研究主要侧重于诊断算法的诊断效果，因此均是基于传统的集中式处理方式。对于带宽受限、实时性要求高的移动卫星网络，需要更加关注算法在分布式、实时性等方面的需求。

4.3 基于 SLD 的分布式卫星网络故障识别算法

4.3.1 卫星网络建模

低轨通信星座系统采用 Walker 星座，星间链路为 Ka 频段信号，每个卫星节点包含 4 个端口，每个端口对应一条星间链路，每个卫星节点通过四条星间链路与相邻四个节点进行通信，其中，包括 2 条同轨面链路和 2 条异轨面链路。任一个卫星节点在系统中的理论拓扑图如图 4.3-1 所示：

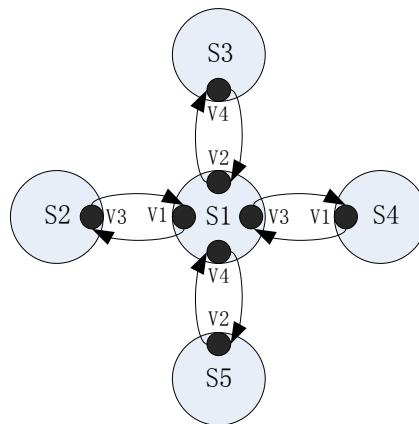


图 4.3-1 节点理论拓扑图

星座网络拓扑具备两个特点：动态性和规律性，这两个特点使我们可以将星座网络的拓扑结构划分为若干个静态拓扑快照，它们呈周期性变化。当处在某一个拓扑时刻时，节点与节点之间的通信可以且唯一映射为某一固定端口之间的通信，因此这里仍然采用图的形式对整个系统进行建模，用卫星节点表示图的顶点。系统可表示为 $G=(S, E)$ ，其中 $S=\{S_1, S_2, \dots, S_n\}$ 表示网络中端口的集合，由 n 个子集组成，其中子集可表示为 $S_i=\{V_{i,1}, V_{i,2}, V_{i,3}, V_{i,4}\}$ ，表示节点 S_i 上的端口集合。 E 表示有向边集合，对应了系统中的星间链路，如果节点 S_i 到 S_j 之间存在可以通信的星间链路，则图 G 中存在边 $e=(S_i, S_j) \in E$ ，可记为 $E_{i,j}$ 。

实际的网络拓扑结构是实时变化的，因此测试图可表示为 $D(t)=(S, T)$ ，表明 t 时刻的测试结果，其中 $t=\{t_1, t_2, \dots, t_m\}$ ，若整个网络运行周期可划分为 m 个拓扑快照，则 t_m 表示第 m 个拓扑时刻。设存在边 $(S_i, S_j) \in E$ ，则当 S_i 测试 S_j 得到结果为 x 时，测试值记为 $T_{i,j}(t)=x$ 。

4.3.2 故障测试过程

DSFD 算法采用分布式的处理可以使节点仅在获得与邻居节点互测结果的情况下，通过对局部故障测试图的判断，从而获得节点以及邻居节点的状态。这样避免了节点为获取整个网络的测试信息而导致的网络负载加重的情况。DSFD 算法的故障测试过程如图 4.3-2 所示：

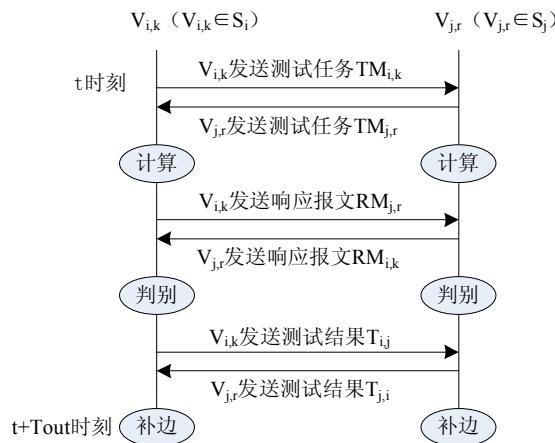


图 4.3-2 DSFD 算法的故障测试过程

如图中所示，测试过程可分为以下几个步骤：

- (1) 测试任务发起：网络中所有节点在 t 时刻通过相应的端口向自己的邻居节点（存在星间链路）发送测试任务，每个端口可以发送不同的测试任务，从而使各自获得的测试结果相独立，端口 $V_{i,k}$ 发送的任务为 $TM_{i,k}$ ；
 - (2) 测试数据回收：收到测试任务的端口 $V_{j,r}$ 执行其中的计算任务(星上代理执行)，然后返回携带执行结果的响应报文，端口 $V_{i,k}$ 收到的回执报文为 $RM_{i,k}$ ；
 - (3) 状态判断：发起测试的端口 $V_{i,k}$ 将收到的执行结果与预期结果进行对比，相同时，标识测试结果为 $T_{i,j}(t) = 0$ ，否则为 $T_{i,j}(t) = 1$ ；同时将该测试结果返回给被测端口 $V_{j,r}$ ；
 - (4) 测试过程结束：在($t+T_{out}$)时，每个节点将通过测试未发现的端口和边补上（在拓扑连接已知的情况下），并将测试值设为 2， T_{out} 为系统设置的时间阈值；
- 故障测试是网络故障诊断的基础，测试结果反映了网络的症状。

4.3.3 M-概率的分布式测试模型

卫星的故障原因多种多样，表现出来的链路通信状态也不同。由器件老化失效、电源故障、电磁干扰、机械故障等可能造成设备断电、电路破坏、天线失效等故障，直接导致卫星端口无法和其它卫星通信。由于空间辐射带来的单粒子翻转会导至信号翻转、计算错误等故障。空间频段干扰则会导致链路通信失败，但这种故障不是长期的，可能是一段时间或者在某个区域才会发生。概括起来，卫星网络中节点状态可分

为正常(0)、软故障(1)和硬故障(2)3种。硬故障是指端口失效，无法进行通信。软故障是指端口有效，但计算错误，也指星上代理计算错误。显然硬故障引起的测试结果是端口无关的，而软故障引起的测试结果则是端口相关的。这为制定诊断规则提供了依据。

节点间进行相互测试时，不同测试模型对测试结果的解释不同。例如系统级故障诊断理论里最常用的PMC模型中，一个正常节点测试正常节点时的结果为0，测试故障节点时的结果为1；而当一个软故障节点测试另一节点时，结果可能为0也可能为1。显然，这种模型下假设了正常节点总是能检测出故障节点，其测试的故障覆盖率为100%。而同样如BGM模型，假设故障节点测试故障节点的结果总是为0。基于这些假设的前提下，通常能够推导出正确的诊断结果，然而很多假设条件并不完全符合实际情况，在实现时，总是需要通过多次重复测试、提高测试任务的复杂性等使得实际情况无限逼近假设的模型。因此，与实际相结合更加合适的方式是采用概率性算法。

本文通过对测试任务的设计上来分析各种测试结果发生的概率，从而建立一个符合实际情况的概率测试模型。

大多数研究系统级诊断算法的文献对测试任务的描述都是停留在报文传递流程上，并未对测试任务与测试结果的概率之间的关系进行分析。本文在基于概率的通用测试模型基础上，采用随机测试任务方式，进行端口之间的有向测试，进一步分析测试结果的概率。

本文采用的测试任务主要是计算功能，假设测试任务Test assignment为一个映射算法，根据每次不同的输入约束，输出不同的计算结果，而这个计算结果的范围被限定为M个，这在计算机系统中很容易实现，比如限定输出结果为Nbit($M=2^N$)的二进制数。设测试任务的正确计算结果为A，端口 $V_{i,k}$ 计算结果表示为 $\Delta_{i,k}$ ，则节点的计算结果的各种条件概率如下：

$$P(\Delta_{i,k} = A | V_{i,k} = 0) = 1 \quad (\text{式 4.3-1})$$

$$P(\Delta_{i,k} = A | V_{i,k} = 1) = \frac{1}{M} \quad (\text{式 4.3-2})$$

$$P(\Delta_{i,k} = B | V_{i,k} = 1) = \frac{1}{M} \quad (\text{式 4.3-3})$$

其中，B表示任意一个计算结果。

如前节所述，若端口 $V_{i,k}$ 发起对端口 $V_{j,r}$ 的测试，若由端口 $V_{j,r}$ 返回的计算结果与端口 $V_{i,k}$ 的计算结果一致，则 $T_{i,j}=0$ ，若不一致则 $T_{i,j}=1$ 。

显然，一个正常节点测试一个正常节点的概率为：

$$P(T_{i,j} = 0 | V_{i,k} = 0, V_{j,r} = 0) = 1 \quad (\text{式 4.3-4})$$

一个正常节点测试一个故障节点的概率为：

$$\begin{aligned} P(T_{i,j} = 0 | V_{i,k} = 0, V_{j,r} = 1) &= P(\Delta_{i,k} = \Delta_{j,r} | V_{i,k} = 0, V_{j,r} = 1) \\ &= P(\Delta_{j,r} = A, V_{j,r} = 1 | \Delta_{i,k} = A, V_{i,k} = 0) \\ &= \frac{P(\Delta_{j,r} = A | V_{j,r} = 1) P(\Delta_{i,k} = A, V_{i,k} = 0)}{P(\Delta_{i,k} = A, V_{i,k} = 0)} = \frac{1}{M} \end{aligned} \quad (\text{式 4.3-5})$$

$$\begin{aligned} P(T_{i,j} = 1 | V_{i,k} = 0, V_{j,r} = 1) &= 1 - P(T_{i,j} = 0 | V_{i,k} = 0, V_{j,r} = 1) \\ &= 1 - \frac{1}{M} \end{aligned} \quad (\text{式 4.3-6})$$

一个故障节点测试一个正常节点的概率如下：

$$P(T_{i,j} = 0 | V_{i,k} = 1, V_{j,r} = 0) = \frac{1}{M} \quad (\text{式 4.3-7})$$

$$\begin{aligned} P(T_{i,j} = 1 | V_{i,k} = 1, V_{j,r} = 0) &= 1 - P(T_{i,j} = 0 | V_{i,k} = 1, V_{j,r} = 0) \\ &= 1 - \frac{1}{M} \end{aligned} \quad (\text{式 4.3-8})$$

一个故障节点测试一个故障节点的概率如下：

$$P(T_{i,j} = 0 | V_{i,k} = 1, V_{j,r} = 1) = \frac{1}{M} \quad (\text{式 4.3-9})$$

$$\begin{aligned} P(T_{i,j} = 1 | V_{i,k} = 1, V_{j,r} = 1) &= 1 - P(T_{i,j} = 0 | V_{i,k} = 1, V_{j,r} = 1) \\ &= 1 - \frac{1}{M} \end{aligned} \quad (\text{式 4.3-10})$$

端口与端口之间的测试图可表示为：

表 4.3-1 基于概率的测试模型

$V_{i,k}$ ($V_{i,k} \in S_i, k=1, \dots, 4$)	$V_{j,r}$ ($V_{j,r} \in S_j, r=1, \dots, 4$)	$T_{i,j}$	概率
0	0	0	1
0	1	0	p
0	1	1	1-p
1	0	0	p
1	0	1	1-p
1	1	0	p
1	1	1	1-p
0,1,2	2	2	1

2	0,1,2	2	1
注: $P=1/M$			

其中 $T_{i,j}$ 表示端口 $V_{i,k}$ 对 $V_{j,r}$ 的测试结果, $T_{j,i}$ 则表示端口 $V_{j,r}$ 对 $V_{i,k}$ 的测试结果。另外, 如图所示, 当测试双方任一方存在硬故障时, 测试结果均为 2。

传统的 SLD 算法需要获得整网的测试结果, 经过集中处理对所有端口进行统一的故障识别。这样做就需要将每个节点的测试结果汇集到一个统一的诊断中心, 由此带来的路由时延会造成诊断时延过长, 并且如果存在硬故障端口则可能由于路由问题导致诊断中心无法获得某些节点的测试结果, 同时这种集中式的处理方式也不利于卫星网络自主运行。因此, 本文提出了基于分布式处理方式的算法, 采用同一节点上端口与邻居端口的互测结果进行节点自身状态的诊断。

如 4.3.2 节故障测试流程的描述, 测试结束后, 每个节点上可以获得邻居端口的互测结果, 这个互测结果可表示为:

$$D_{i,j} = (T_{i,j}, T_{j,i})$$

$D_{i,j}$ 代表节点 S_i 上计算的 $V_{i,k}$ 对邻居端口 $V_{j,r}$ 的测试结果以及收到的节点 S_j 上邻居端口 $V_{j,r}$ 对 $V_{i,k}$ 的测试结果的组合结果。

$D_{i,j}$ 发生的概率可表示为:

$$P(D_{i,j}|V_{i,k}, V_{j,r}) = P(T_{i,j}|V_{i,k}, V_{j,r}) P(T_{j,i}|V_{i,k}, V_{j,r}) \quad (\text{式 4.3-11})$$

由此可建立 M-概率的分布式测试模型, 如表 4.3-2 所示。

表 4.3-2 M-概率的分布式测试模型

$V_{i,k}$ ($V_{i,k} \in S_i, k=1, \dots, 4$)	$V_{j,r}$ ($V_{j,r} \in S_j, r=1, \dots, 4$)	$D_{i,j}$ ($= (T_{i,j}, T_{j,i})$)	$P(D_{i,j} V_{i,k}, V_{j,r})$
0	0	(0, 0)	1
0	1	(0, 0)	p^2
		(0, 1)	$p(1-p)$
		(1, 0)	$p(1-p)$
		(1, 1)	$(1-p)^2$
		(0, 0)	p^2
1	0	(0, 1)	$p(1-p)$
		(1, 0)	$p(1-p)$
		(1, 1)	$(1-p)^2$
		(0, 0)	p^2
1	1	(0, 1)	$p(1-p)$
		(1, 0)	$p(1-p)$
		(1, 1)	$(1-p)^2$

		(1, 1)	$(1-p)^2$
0,1,2	2	(2, 2)	1
2	0,1,2	(2, 2)	1

注: $p=1/M$

表 4.3-2 是对单个节点上能够获得的端口互测结果的总结。

4.3.4 小概率忽略的诊断规则

根据表 4.3-2, 若 M 值足够大, 可得这些概率的极限为:

$$\lim_{M \rightarrow \infty} P(D_{i,j} = (0,0) | V_{i,k} = 0, V_{j,r} = 1) = \lim_{M \rightarrow \infty} p^2 = \lim_{M \rightarrow \infty} \frac{1}{M^2} = 0 \quad (\text{式 4.3-12})$$

$$\begin{aligned} \lim_{M \rightarrow \infty} P(D_{i,j} = (0,1) | V_{i,k} = 0, V_{j,r} = 1) &= \lim_{M \rightarrow \infty} p(1-p) \\ &= \lim_{M \rightarrow \infty} \frac{1}{M} \left(1 - \frac{1}{M}\right) = 0 \end{aligned} \quad (\text{式 4.3-13})$$

$$\begin{aligned} \lim_{M \rightarrow \infty} P(D_{i,j} = (1,0) | V_{i,k} = 0, V_{j,r} = 1) &= \lim_{M \rightarrow \infty} p(1-p) \\ &= \lim_{M \rightarrow \infty} \frac{1}{M} \left(1 - \frac{1}{M}\right) = 0 \end{aligned} \quad (\text{式 4.3-14})$$

$$\begin{aligned} \lim_{M \rightarrow \infty} P(D_{i,j} = (1,1) | V_{i,k} = 0, V_{j,r} = 1) &= \lim_{M \rightarrow \infty} (1-p)^2 \\ &= \lim_{M \rightarrow \infty} \left(1 - \frac{1}{M}\right)^2 = 1 \end{aligned} \quad (\text{式 4.3-15})$$

同样,

$$\begin{aligned} \lim_{M \rightarrow \infty} P(D_{i,j} = (0,0) | V_{i,k} = 1, V_{j,r} = 0) \\ = \lim_{M \rightarrow \infty} P(D_{i,j} = (0,1) | V_{i,k} = 1, V_{j,r} = 0) \end{aligned} \quad (\text{式 4.3-16})$$

$$= \lim_{M \rightarrow \infty} P(D_{i,j} = (1,0) | V_{i,k} = 1, V_{j,r} = 0) = 0$$

$$\lim_{M \rightarrow \infty} P(D_{i,j} = (1,1) | V_{i,k} = 1, V_{j,r} = 0) = 1 \quad (\text{式 4.3-17})$$

$$\begin{aligned} \lim_{M \rightarrow \infty} P(D_{i,j} = (0,0) | V_{i,k} = 1, V_{j,r} = 1) \\ = \lim_{M \rightarrow \infty} P(D_{i,j} = (0,1) | V_{i,k} = 1, V_{j,r} = 1) \end{aligned} \quad (\text{式 4.3-18})$$

$$= \lim_{M \rightarrow \infty} P(D_{i,j} = (1,0) | V_{i,k} = 1, V_{j,r} = 1) = 0$$

$$\lim_{M \rightarrow \infty} P(D_{i,j} = (1,1) | V_{i,k} = 1, V_{j,r} = 1) = 1 \quad (\text{式 4.3-19})$$

由此可见, 只要 M 值足够大, 该模型可简化为表 4.3-3:

表 4.3-3 $M \rightarrow \infty$ 的简化测试图

$V_{i,k}$ ($V_{i,k} \in S_i$, $k=1, \dots, 4$)	$V_{j,r}$ ($V_{j,r} \in S_j$, $r=1, \dots, 4$)	$D_{i,j}$ ($=\{T_{i,j}, T_{j,i}\}$)	$P(D_{i,j} V_{i,k}, V_{j,r})$
0	0	(0, 0)	1
0	1	(1, 1)	1
1	0	(1, 1)	1
1	1	(1, 1)	1
0,1,2	2	(2, 2)	1
2	0,1,2	(2, 2)	1

根据 $M \rightarrow \infty$ 的简化测试图，在建立诊断规则时，如果忽略小概率事件，则可降低测试模型在诊断推理时的不确定性。当然由此可能引起诊断错误的后果，但这一风险可在诊断过程中通过后处理的方式进行消减。据此可制定小概率忽略的诊断规则，如下：

规则 1 若 $D_{i,j} = (0, 0)$, 则 $V_{i,k}=V_{j,r}=0$;

规则 2 若 $D_{i,j} = (1, 1)$, 则 $V_{i,k}$ 和 $V_{j,r}$ 至少有一个为 1;

规则 3 若 $D_{i,j} = (1, 1)$, 且 $V_{i,k}=0$, 则 $V_{j,r}=1$;

规则 4 若 $D_{i,j} = (1, 1)$, 且 $V_{j,r}=0$, 则 $V_{i,k}=1$;

规则 5 若 $D_{i,j} = (2, 2)$, 则端口 $V_{i,k}$ 和 $V_{j,r}$ 中至少有一个为 2;

规则 6 若 $V_{i,k}$ 为非硬故障状态 (0 或 1), 则 S_i 节点上所有的非硬故障端口状态均与 $V_{i,k}$ 相同, 这是由于星上代理负责所有端口的计算任务, 因此同一节点上的非硬故障端口计算能力应该一致[5]。

由上述诊断规则可以看出, 在没有其它辅助判决信息的情况下, 硬故障端口状态无法被识别。因此本文算法主要是针对软故障状态识别问题的研究。

4.3.5 分布式故障识别算法

上述诊断规则制定的基础是建立在“将小概率测试结果忽略”的假设前提下, 而事实上小概率事件的发生会导致两个结果:

(1) 实际测试图与诊断规则的不符合, 如实际中小概率测试结果 $D_{i,j} = (0, 1)$ 或 $(1, 0)$;

(2) 实际测试图中存在虚假测试结果, 能够误导诊断结论, 如实际中 $D_{i,j} = (0, 0)$, 但对应的 $V_{i,k}=V_{j,r}=1$ 。

这两种情况均需在诊断过程中加以识别, 主要通过两个措施:

(1) 测试图预处理;

(2) 风险消减, 该措施包含在分布式故障诊断流程中。

4.3.5.1 测试图预处理

当两个互测端口中存在软故障端口时，可能出现的小概率测试结果中包含了前述诊断规则不能判别的情况，如表 4.3-4。

表 4.3-4 不符合诊断规则的小概率测试结果

$V_{i,k}$ ($V_{i,k} \in S_i$, $k=1, \dots, 4$)	$V_{j,r}$ ($V_{j,r} \in S_j$, $r=1, \dots, 4$)	$T_{i,j}$	$T_{j,i}$
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0

因此，在诊断开始前必须对测试图进行以下预处理：

- (1) 若 $T_{i,j}=0$ 且 $T_{j,i}=1$ ，则修改 $T_{i,j}=1$ ；
- (2) 若 $T_{j,i}=0$ 且 $T_{i,j}=1$ ，则修改 $T_{j,i}=1$ ；

这种预处理实际上是对小概率事件的纠正，能够消除不符合诊断规则的小概率结果，从而大大降低了测试结果的不确定性。

经过对测试图的预处理，每个节点可获得本地测试图如表 4.3-5 所示。

表 4.3-5 修正后的节点本地测试图

$V_{i,k}$	$V_{j,r}$	$D_{i,j}(T_{i,j}, T_{j,i})$	$P(D_{i,j} V_{i,k}, V_{j,r})$
0	0	(0, 0)	1
0	1	(0, 0)	q
0	1	(1, 1)	$1-q$
1	0	(0, 0)	q
1	0	(1, 1)	$1-q$
1	1	(0, 0)	q
1	1	(1, 1)	$1-q$
2	0,1,2	(2, 2)	1
0,1,2	2	(2, 2)	1
注：			
① $V_{j,r}$ 表示与 S_i 进行通信的一个邻居节点端口；			
② $D_{i,j}(T_{i,j}, T_{j,i})$ 表示 $V_{i,k}$ 与 $V_{j,r}$ 的互测结果；			
③ $q=1/M^2$ ；			

从上表可以看出，测试图的预处理并不能消除相互测试的两端口中包含有故障端口时测得 $D_{i,j}=(0, 0)$ 这种小概率事件所带来的虚假测试结果，根据诊断规则，一旦出

现这种虚假测试结果，其风险便是直接导致错误的诊断结论，因此仍然有必要在诊断过程中对这种情况加以识别和处理。

测试图同时也表明单次测试过程中无法识别出硬故障端口，测试结果为（2，2）的互测端口至少有一个为硬故障，但无法确诊。因此硬故障可通过星座网络不通拓扑连接下的测试结果比对来进一步识别。本文仅限于研究软故障引起的诊断问题，因此下述诊断流程中只论述了软故障的识别方法。

4.3.5.2 分布式故障诊断流程

不考虑系统中存在硬故障的情况下，端口的状态可以用节点的非硬故障状态统一表示，为了更好更简便地表达本算法的诊断流程，下面作几个状态定义：

- ✓ S_i 表示 S_i 上所有非硬故障端口的状态；
- ✓ $S_i (S_i)$ 表示 S_i 自身对 S_i 的状态判别结果，且代表了 S_i 上所有非硬故障端口的状态；
- ✓ $S_N (S_i)$ 表示 S_i 对邻居节点 S_N 的状态判别结果，且代表了 S_N 上所有非硬故障端口的状态；
- ✓ $S_i (S_N)$ 表示邻居节点 S_N 对 S_i 的状态判别结果，且代表了 S_i 上所有非硬故障端口的状态；

诊断的主要目的是识别出 S_i 上所有非硬端口的状态，识别的过程如图 4.3-3 所示。

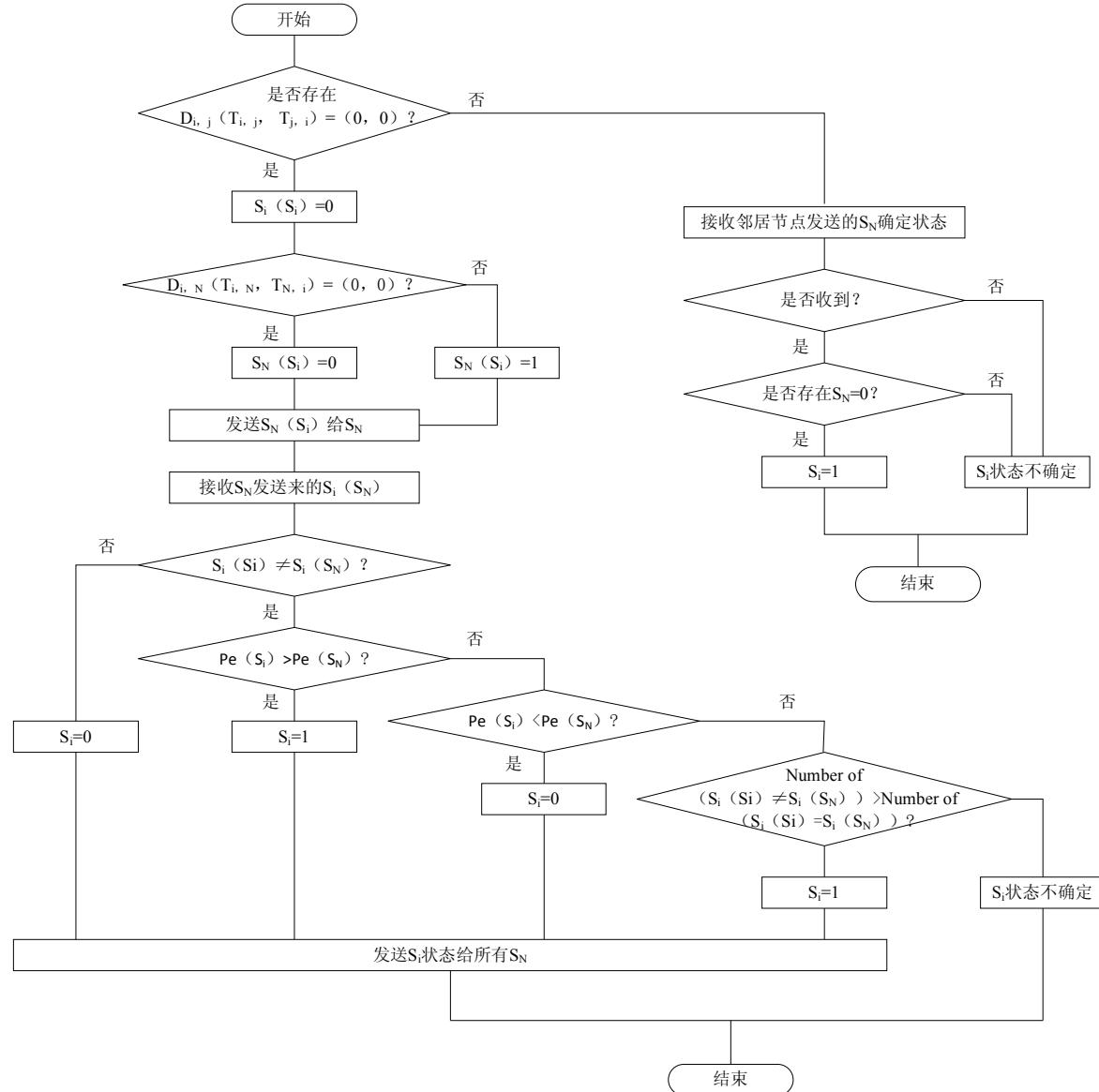


图 4.3-3 分布式故障诊断过程

当测试过程结束时,每个节点上至多获得 4 个 D 值,诊断过程包括以下几个步骤:

(1) 预判决

首先可以根据诊断规则进行预判决,获得端口的确定状态,便于互测端口状态的下一步推理。预判决的过程为:判断 4 个 D 值中是否存在 $D_{i,j}=(0, 0)$,若存在,则判断节点自身状态为正常,记为 $S_i (S_i) = 0$;

根据故障测试过程,同一个节点上不同端口与各自的邻居端口进行互测时选择的测试任务是随机的,互不相干的,因此 4 个 D 值的结果也是不相干的。设 4 个 D 值中存在 $(0,0)$ 的数量为 Nr 个,则本次判决错误的概率为:

$$P_e = \prod_{Nr} P(D_{i,j} = (0,0) | V_{i,k}, V_{j,r}) = q^{Nr} = \frac{1}{M^{2Nr}} \quad (Nr \in \{1,2,3,4\}) \quad (\text{式 4.3-20})$$

(2) 关联推理

以预判决时获得的状态确定的端口作为基础，根据诊断规则进行推理可得到邻居端口状态，推理过程为：当 $S_i(S_i)=0$ ，根据节点上的 D 值对邻居节点状态进行判决：若 $D_{i, N}(T_{i, N}, T_{N, i}) = (0, 0)$ ，则 $S_N(S_i)=0$ ，若 $D_{i, N}=(1, 1)$ ，则 $S_N(S_i)=1$ ；这称为初次判决。

该步关联推理的基础是基于“ $S_i(S_i)=0$ ”的假设前提下，因此本次判决错误的概率等于判决“ $S_i(S_i)=0$ ”错误的概率，即 P_e 。

(3) 数据交换

数据交换过程是进行小概率风险消减措施的前提步骤，也是分布式处理的重要过程。通过邻居端口之间对预判决和关联推理结果的数据交换，可以使节点获得更多的信息量，有助于故障的识别。具体如下： S_i 与邻居节点 S_N 交换各自的诊断结果，诊断结果形式如下：

节点	状态值	错误概率
S_i	$S_N(S_i)$	$P_e(S_i)$
S_N	$S_i(S_N)$	$P_e(S_N)$

事实上，这里隐含了一个条件，即只有判断自身状态为正常（0）的节点才能够获得对邻居节点的判决状态值。因此，能够与邻居节点交换数据的节点一定满足“ $S_i(S_i)=0$ ”的条件。

(4) 风险消减

小概率事件产生的欺骗性信息带来的直接风险便是误判决，风险消减过程便是尽可能地消除误判并识别出节点的真正状态。

数据交换的目的在于发现小概率事件带来的误判决，如果初次判决获得的节点状态与实际状态一致，则相邻节点互相判决的结果也应该是一致的。如果出现不一致的情况，则说明两者中必然存在误判的情况。本步骤便是根据发生错误概率的大小来纠正或减少误判结果，从而提高诊断的正确率。

由此说明，若存在 $S_i(S_i) \neq S_i(S_N)$ ，说明 S_i 和 S_N 中必有一个节点的判决是错误的，这时，必然是 $S_i(S_i)=0, S_i(S_N)=1$ ；若不存在 $S_i(S_i) \neq S_i(S_N)$ ，则不能说明它们之中存在误判决。

判决过程如下：

- 当 $S_i(S_i)=0$ 且 $S_i(S_i) \neq S_i(S_N)$ 时，首先判断 $P_e(S_i(S_i)=0)$ 与 $P_e(S_i(S_N)=1)$ 大小，若 $P_e(S_i) > P_e(S_N)$ ，则判决 $S_i=1$ ；若 $P_e(S_i) < P_e(S_N)$ ，则判决 $S_i=0$ ；若 $P_e(S_i)=P_e(S_N)$ ，则需进行进一步判断，在本节点 S_i 上判断 $\text{Number of } (S_i(S_i) \neq S_i(S_N))$ 是否大于 $\text{Number of } (S_i(S_i)=S_i(S_N))$ ，若大于，则确认 $S_i=1$ ，反之，则不确定 S_i 状态；

该判决是实际上是基于投票机制的，其判决错误的概率直接依赖于前述步骤的误判概率。

➤ 若不存在 $S_i (S_i) \neq S_N (S_N)$, 则确认 $S_i=0$;

下面分析该步骤发生错误的概率。该步骤选择信任前述判决结果，因此其发生错误的概率为：

$$P_e = P_e(S_i) \cdot \prod_{N_e} P_e(S_N), \quad (N_e \in \{1, 2, 3, 4\}) \quad (\text{式 4.3-21})$$

若经过上述判决步骤后 S_i 状态确定，则发送 S_i 状态（0/1）给 S_N ，否则不发送。

(5) 后判决

有些节点无法从获得的 D 值中直接确定状态，例如其获得的四个 D 值全部为(1,1)的情况，根据步骤 1，其无法获得 $S_i (S_i)$ 值，这种情况就需通过已确认状态的邻居节点对它的判决结果来判断，这里称之为后判决：若全部 $D_{i,k} = (1, 1)$ ($k=1, \dots, 4$)，则根据邻居节点发送的状态来判断，若存在 $S_N=0$ ，则 $S_i=1$ ；反之，则不确定 S_i 状态；

上述诊断流程的具体算法如图 4.3-4 所示。

```

Diagnosis Algorithm
{
    if(exist D==(0,0))
    {
        S_i(S_i)=0;
        Calculate P_e;
        for(; ;)
        {
            if(D_{i,N}=(1, 1))
                S_N(S_i)=1;
            else
                S_N(S_i)=0;
        }
        transmitting S_N(S_i) and P_e to S_N;
    }
    on receiving S_i(S_N) and P_e from S_N
    {
        if(S_i(S_i)==0)
        {
            if(exist S_i(S_i)≠S_i(S_N))
            {
                if(P_e(S_i(S_i))>P_e(S_i(S_N)))
                    S_i=1;
                else if(P_e(S_i(S_i))<P_e(S_i(S_N)))
                    S_i=0;
                else if(number of (S_i(S_i)≠S_i(S_N))>number of (S_i(S_i)==S_i(S_N)))
                    S_i=1;
            }
            else
                S_i=0;
        }
        if(S_i≠-1)
            transmitting S_i to S_N;
    }
    else
    {
        wait;
        on receiving S_N from S_N
        {
            if(S_N==0)
                S_i=1;
            if(S_N==1)
                S_i=0;
        }
    }
}

```

图 4.3-4 分布式故障诊断算法

4.4 仿真分析

本文对(24,3,1)的walker星座进行了仿真，端口共96个。如前述，单颗卫星在任何一个时刻与其它卫星之间的拓扑关系都可表示为图1所示，因此取任意拓扑时刻进行多次算法仿真获得的故障诊断结果不失一般性。

仿真的目的有两个：

- (1) 仿真算法的有效性，分析参数F%和M对算法性能的影响；
- (2) 比较本算法与其它算法在性能上的优劣；

4.4.1 诊断完全率和诊断正确率

算法的有效性通过诊断完全率C%和诊断正确率R%来衡量。设网络端口总数量为n，故障端口数量为f，未识别出状态的端口数量为u，状态识别错误的端口数量为e，则 $C\% = (n-u)/n$, $R\% = (n-u-e)/(n-u)$, 故障率 $F\% = f/n$ 。

仿真的输入参数除了网络拓扑外还包括故障率F%以及M值。前述的诊断规则已表明算法对硬故障无法确诊，因此仿真时主要方针本算法对于软故障的识别能力，设置故障端口的类型为软故障，含有故障端口的卫星在网络中服从均匀分布。仿真结果如图4.4-1和图4.4-2所示。

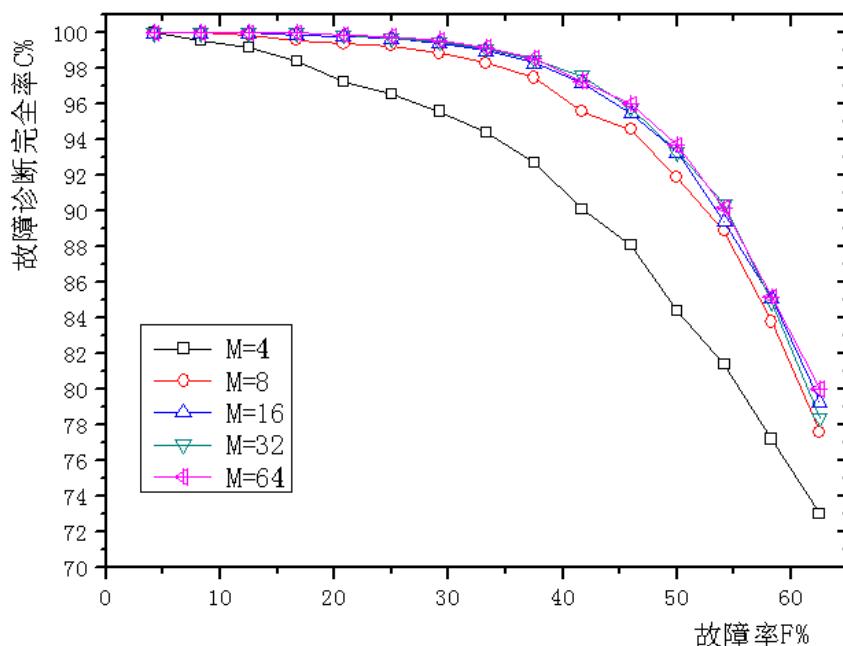


图 4.4-1 C%、M 及 F%之间的关系图

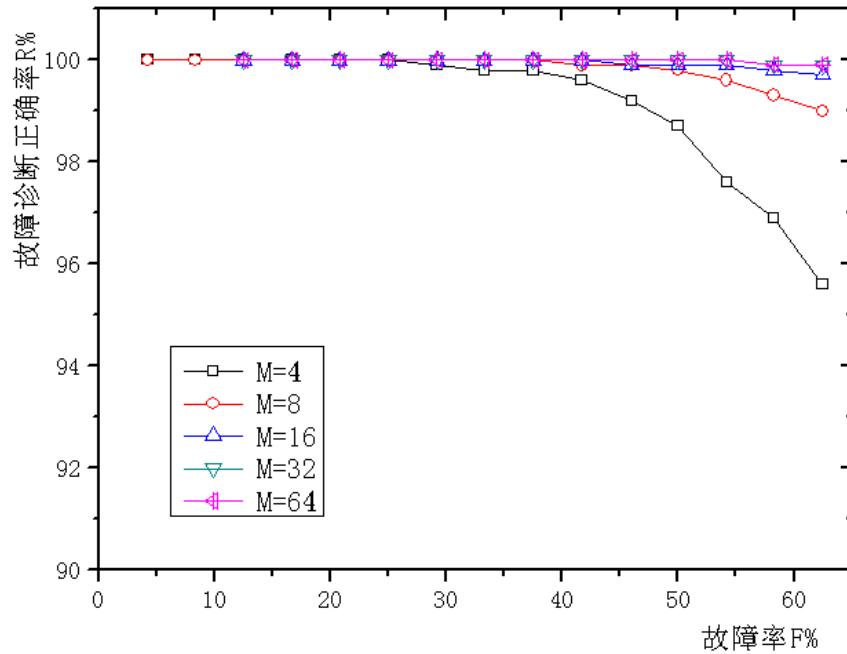


图 4.4-2 R%、M 及 F%之间的关系图

图 4.4-1 给出了不同 M 取值时软故障率与故障诊断完全率之间的关系，图 4.4-2 则表示不同 M 取值时软故障率与故障诊断正确率之间的关系。从仿真结果可以看出，本算法对软故障的诊断完全率非常高。当 $M \geq 16$ 时，软故障率 60% 以下故障诊断完全率均在 80% 以上，软故障率 30% 以下故障诊断完全率则在 99% 以上。同时从图中可以看出只要取合适的 M 值，故障诊断正确率也能得到保证， $M \geq 32$ 时，软故障率 50% 以下故障诊断正确率均为 100%，软故障率 60% 以下故障诊断正确率均为 99.9%。

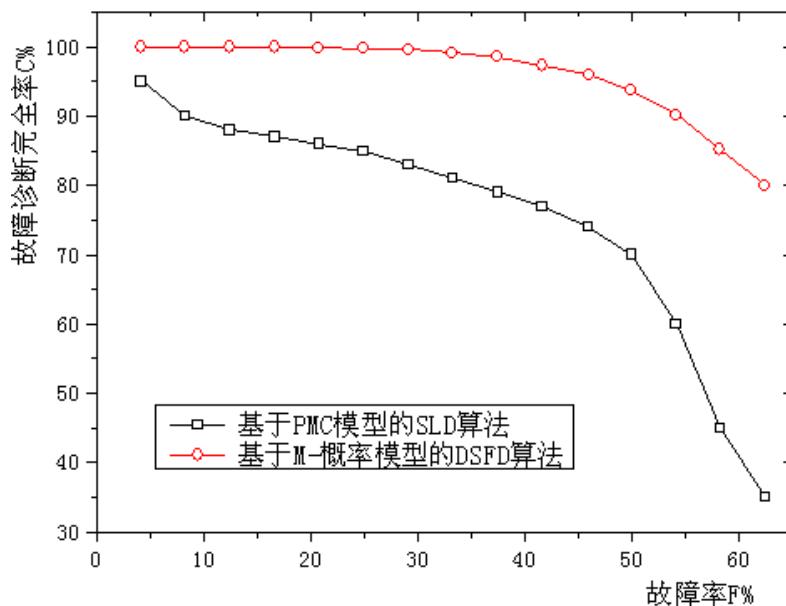


图 4.4-3 诊断完全率性能的比较图

图 4.4-3 是 DSFD 算法与基于 PMC 模型的 SLD 算法^[91]在诊断完全率上的比较。从图中可以看出，DSFD 算法的故障诊断完全率远远超出了基于 PMC 模型的 SLD 算法。

4.4.2 诊断开销与诊断时延

DSFD 算法与传统 SLD 算法^[91]、ADSD 算法^[75]在诊断开销和诊断时延上的性能比较如图 4.4-4 和图 4.4-5 所示。

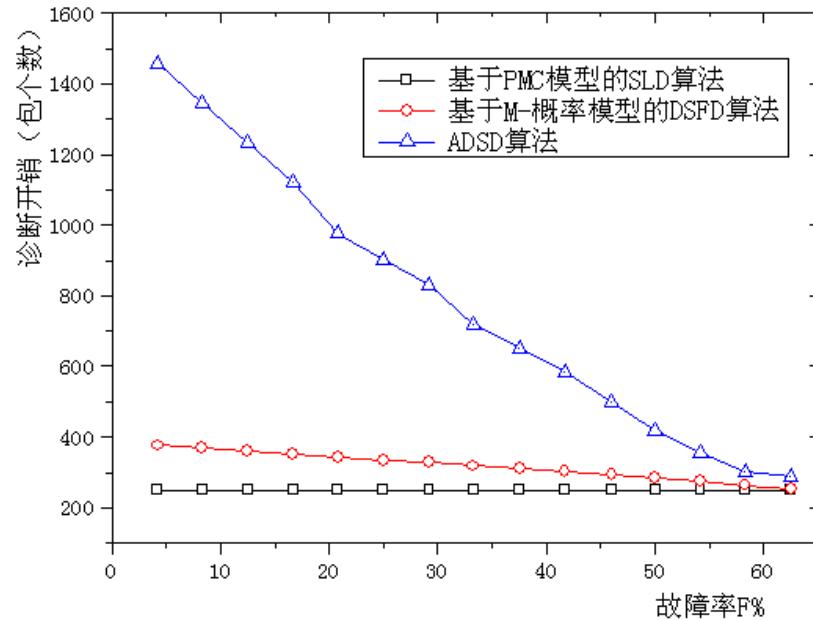


图 4.4-4 诊断开销比较图

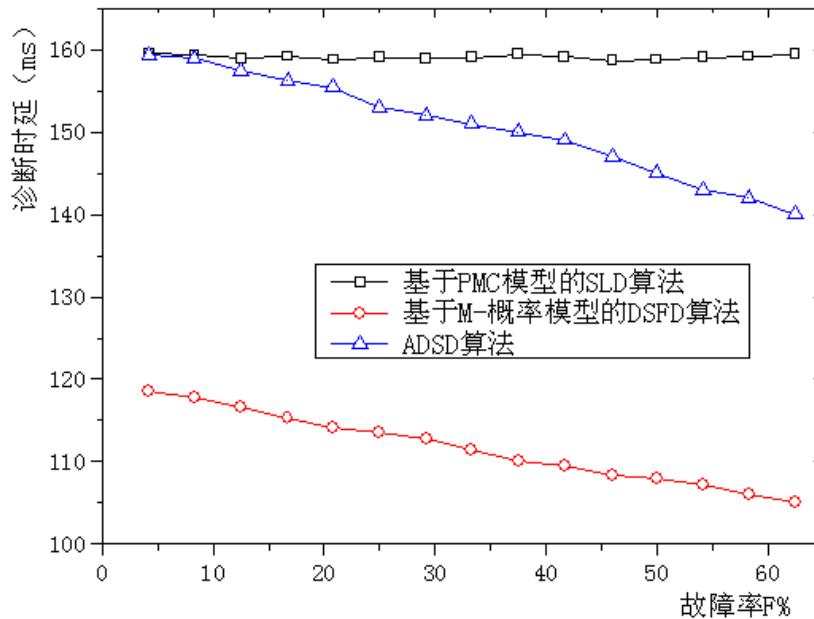


图 4.4-5 诊断时延比较图

由图 4.4-4 可以看出，DSFD 算法在诊断开销上多于传统 SLD 算法，少于 ADSD 算法，并随着故障率的增加而减少。这是由于传统 SLD 算法采用的是基于集中式处理的诊断方式，无需网络中所有节点都获得测试结果，而 ADSD 算法则是基于分布式处理的方式，并且要求每个节点都能对全网所有节点进行诊断，因而需交换大量的诊断消息。本文提出的 DSFD 算法同样是基于分布式处理方式，但它仅要求节点对局部网络

进行诊断，因此需交换的诊断信息量远远少于 ADSD 算法。

图 4.4-5 的诊断时延包括了测试过程和诊断过程，仿真时取 $T_{out}=55ms$ 。图示仿真结果表明 DSFD 算法的诊断时延性能远远优于传统 SLD 算法和 ADSD 算法。这是由于传统 SLD 算法的集中式处理方式需要获得所有节点的测试结果才能开始诊断，因此最长路由是影响其时延的主要因素。ADSD 算法要求每个节点均需获得其他正常节点的测试结果，最长路由也是影响它的主要因素，同时由于其诊断依赖于正常节点对测试结果的传输，因此其时延会随着故障率的增加而减少。本文的 DSFD 算法由于仅要求局部诊断，只需和相邻节点交换信息，因此诊断时延大大降低。

4.5 小结

本文基于系统级故障诊断理论，结合低轨星座系统网络自主运行的需求，提出了一种适合星上自主处理的分布式网络故障识别算法。本算法在 PMC 模型的基础上提出了 M-概率的分布式测试模型，通过小概率忽略的诊断规则，使得故障诊断完全率大大提高，采用分布式的处理方式有效地减小了诊断时延。本文通过仿真对算法的输入参数与性能之间的关系进行了分析，并与其它算法在诊断完全率、诊断时延、诊断开销等方面的性能进行了比较。仿真结果表明，只要设置合适的 M 参数，可以保证故障诊断的正确率，同时在保证诊断结果正确的条件下获得较高的故障诊断完全率。此外，通过分布式的诊断流程处理，算法在诊断开销上优于全局分布式诊断算法，在诊断时延上则优于集中式诊断算法。这说明本算法不仅故障识别率高，同时也较为适合卫星网络自主管理。

第五章 基于移动 agent 的星上自主路由更新技术研究

5.1 引言

路由管理是星座网络自主运行管理的一项重要内容。星座自主运行时采用的路由策略应对网络实时状态具有自主适应性，同时路由管理应自主地对发生故障后的网络路由作出及时反应。

路由算法的自主性通常需要大量的网络状态信息、复杂的星上计算处理来支持，这使得自适应路由算法在星间链路资源紧张、航天器件处理能力有限、可靠性要求高的卫星网络中的实现比较困难。传统的卫星网络路由策略通常采用地面预先计算、上传路由表至卫星的方法来解决星上处理能力有限的问题，而卫星网络的路由算法则采用传统的最短链路时延和静态拓扑快照相结合的方式。这种方法虽然使得星上处理比较简单，但也存在不具有自主性的缺点，怎样既兼顾星上处理能力又兼顾路由自主性是本章研究的一项内容。本文基于静态拓扑快照路由策略，分析了切换对时延抖动的影响，提出了考虑切换和时延的权值路由算法，该算法既能保证路由选择的优先级，又兼顾了网络流量的平衡，同时，采用节点实时状态与权值路由表相结合的方式选择路由，具备了一定程度的自主性。

上述算法是对快照序列路由策略的改进，本质上其路由表仍然是采用由地面预先计算好再上传至星上执行的方式，因此其不能很好地适应卫星网络发生故障时的路由更新需求。针对星座网络自主运行的特点，本章提出一种基于移动 agent 的多层扩散路由更新算法，来解决故障路由更新问题。该算法在能够应对网络拓扑异常变化的同时，不增加过多的星上计算量，并通过控制移动 agent 扩散层数来减小路由更新的扩散范围，从而降低路由更新带来的网络开销。

5.2 路由技术概况

5.2.1 卫星网络路由技术

卫星的高速运动造成 LEO 星座复杂的拓扑结构，这使得地面网络中的路由协议无法直接应用。根据 LEO 星座的特点，本节从网络拓扑结构、切换、星上处理能力、QOS 要求等几个方面分别介绍各类路由算法的国内外研究现状。

(1) 拓扑周期路由

卫星网络周期性路由策略是基于周期分割机制，根据卫星网络运行的周期性，把卫星轨道周期 T 划分为 n 个时间间隔，在时间间隔内，认为卫星网络拓扑结构固定不变。

有限状态自动机(FSA)模型^[93]是利用拓扑周期变化的典型路由算法。将每个系统周

期分为 n 个间隔的离散拓扑状态，在每个间隔内求解链路分配最优问题，该算法隐藏了星座拓扑时变性，成为后续很多路由算法的基础。

Werner 以 ATM 卫星网络为研究对象，提出了动态虚拟拓扑路由算法(DVTR)^[94]。采用类似 FSA 的方法将系统周期划分为一系列时间间隔，在每个固定的拓扑上进行“离线”路由计算，该算法需要对虚拟通道路径进行建立和维护。

Gounder 提出的基于快照序列路由算法^[95]中，每连通或断开一条星间链路，就认为形成一个新的拓扑快照，卫星网络动态拓扑表示为一系列拓扑快照的周期循环，，在每个快照对应的静态拓扑结构上，采用经典的算法(比如 Dijkstra 算法)离线计算路由。分组交换的实现只需在对应的时间间隔采用相应的路由表即可。

文献^[96]提出一种等长时间段快照序列路由改进算法，能够保证时间段内路径不中断，研究了不同时间段划分与丢包率、网络链路利用率、网络端到端延时的关系，该改进算法能够显著降低丢包率。

文献^[97]提出的概率路由协议 (Probabilistic routing protocol)，其基本思想是在一个新呼叫请求建立连接时，删除一部分呼叫生存期内可能被关闭或发生链路切换的星间链路。选择适当的目标概率函数和目标概率值，以最小化重路由和切换概率为目标计算路径。

文献^{[98][99]}采用集中式路由策略，在分割好的静态拓扑下采取 Dijkstra 算法计算路径，算法中以传输时延和队列时延作为衡量链路的标准，因此，网络需要周期性传递各星间链路的状态信息。

基于拓扑周期划分机制的路由策略将动态路由转变为一系列静态路由，有效地简化了路由算法复杂度，降低了通信开销。但这类策略需要每颗卫星存放各个离散间隔内的大量路由信息，对星载存储能力提出了较高的要求，同时对网络的实时信息变化适应性较差。

(2) QOS 路由

在满足不同业务的 QOS 要求方面，许多路由策略充分考虑了流量平衡、分组调度、剩余链路带宽等因素来避免链路的拥塞。

LEO 源路由算法^[100] (SRA, Source Routing Algorithm) 是由源节点发起路由的面向连接的按需路由算法，是一种分布式路由策略，需广播路由请求以支持路由的选择。文献^[101]提出了一种基于空间划分的动态源路由算法，引入逻辑位置的思想屏蔽了卫星移动性对路由选择的影响，仿真结果表明该算法能够在降低路由计算开销和交换开销的同时，保证了数据报的端到端传输时延要求。

Ekici 提出了分布式路由^{[102][103]} (DRA, Distributed Routing Algorithm)，对每个数据分组独立地选择路径，基于传播时延选择下一跳传输方向，处理简单，仅仅当发生链路拥塞后把分组转移到备份的路径上，但此策略对业务的 QOS 需求满足度不高。

文献^[104]提出 DLAR (Distributed Load-Aware Routing) 分布式路由算法，首先计算比当前卫星更接近目的卫星节点的卫星，在这些卫星中根据传播时延和队列时延选择下一跳节点。下一跳节点的计算依赖各卫星的当前位置以及系统确定的动态特征，因此，星上需要大量空间存储卫星各时刻位置。

文献^[105]提出了卫星切换动态概率路由优化策略，为了达到对网络资源的有效使用，当卫星切换时考虑带宽、业务分布等参数优化并重建原路径。文献^[106]假设单层 LEO 星座为双层星座，根据星际链路动态特征和流量分布，在各轨道面选择某卫星节点作为此轨道面的管理中心，在低轨卫星网络中采用双层星座的路由策略。

文献^[107]提出一种基于拓扑简化的分布式路由策略 (DRAST)，为降低星间链路切换频率，根据卫星运行周期和规律简化拓扑，卫星只需要存储拓扑表，结合拓扑特征在星上根据源卫星和目的卫星动态寻找最佳路径。在保证时延性和有效性基础上，降低路由协议的复杂度，但该策略要求每个卫星实时向邻居节点广播队列状态。

文献^[108]选择流量较低的链路作为最佳路径，此算法能有效避免拥塞，并最大化利用剩余的带宽资源。文献^[109]提出在路由中考虑分组的调度。文献^[110]中提出的 TCD (Traffic Class Dependent) 路由策略，在进行路由选择时充分考虑了星上队列和处理时延、星间链路长度、卫星业务的负载或者可用带宽资源等，以满足业务 QoS 要求，但该策略路由开销较高。

文献^[111]提出 SRED (Satellite Routing for end-to-end delay) 策略，采用 WFQ (Weighted Fair Queuing) 调度策略，其带宽分配取决于业务要求和网络可用资源，通过牺牲系统的吞吐量保证业务的时延性能。文献^[112]提出 HPSR (High Performance Satellite Routing) 策略，针对 SRED 的不足，使分配的带宽低于数据的平均速率，能获得较高的吞吐量。但这两种路由策略都要求星上具备较强的计算处理能力。

文献^[113]提出的 Destruction-resistant 路由算法，将星座网络进行分簇，各卫星节点仅仅了解本簇内的各星间链路的状态，全网链路状态信息通过各簇头传递。此算法通过分簇管理使路由高效化，但增加了簇维持带来的信令开销，算法执行的复杂度依赖于卫星网络的规模。文献^[114]在低轨卫星网络中应用改进的遗传算法，建立优化函数选择路径，此算法复杂度较大。

5.2.2 移动 agent 路由技术

移动 agent^[115]是具有移动性的智能 agent，它指能够自行决定在网络中的各个节点之间移动，代表其他实体进行工作的一种软件实体。可以根据具体情况，中断当前的执行，移动到另一设备上恢复运行，并及时地返回执行结果。移动 agent 的本质是将计算移动到数据端，直接在数据端进行本地处理，只返回最终结果，从而避免了大量中间数据在网络中的传输。因此在需要处理的数据量大、网络传输频繁、带宽不足的情况下，使用移动 agent 技术可以有效地节省网络负载、克服网络延迟。

近年来，随着移动 agent 技术的发展，这一理论、技术与其他的领域结合得到广泛应用，其中包括了移动 agent 在路由计算领域内的应用。移动 Agent 拓展了传统的计算模式，它不仅结合了 Agent 的自治、智能等特性，而且引入了移动的概念，极大的延伸了分布式计算的概念，非常适合动态路由的计算。

最早提出的 AntNet 算法^{[116][117][118]}，是后来众多移动 agent 路由算法的基础，它受蚁群系统(ACS: Ant Colony Systems)的启发而来，是一种基于移动 agent 的分布式自适应路由算法，其基本思想是，每一个移动 agent 建立一条从源节点到目的节点的路径，在建立路径时，移动 agent 收集经由的路径代价的大小信息以及网络中业务的数量信息，这些信息被一个反向移动 agent 携带返回，用以修改访问过的节点上路由表。文献^{[117][118]}均为 AntNet 算法的变形，对 AntNet 算法进行了一定的改进。

文献^[119]对 AntNet 算法在负载均衡方面的性能进行了仿真，并与 OSPF 路由协议进行了比较，证明了 AntNet 算法在应对网络变化时的有效性。

文献^[120]分析了 AntNet 算法中 agent 数量的增加对平均包延迟和网络吞吐量的影响，通过控制 agent 的生成率，达到负载感知的目的，从而提高性能。

文献^[121]针对网络规模的扩张造成的移动 agent 可测量性问题，提出将大型网络划分为几个小规模子网络再进行 AntNet 算法的执行的方法，避免移动 agent 的丢失和过期信息，该算法需在节点上维护两个路由表：本地路由表和超级路由表，超级路由表表示节点通向其它簇的路由。文献^[122]针对资源受限的网络提出对 AntNet 算法的改进。

近年来，也有不少研究将移动 agent 技术用于卫星网络动态路由中，如文献^[123]中提出基于 AntNet 的 LEO 卫星路由策略，收集各链路状态信息，在建立路由表时根据链路的状态计算在路径中被选链路的概率，当建立路径后，利用 Forward-Ant 和 Back-Ant 更新链路状态。此算法需要经过多次迭代才能建立稳定的路由表。

文献^[124]提出了一种分布式的 QoS 路由策略 (DQA)，该算法是对启发式蚂蚁算法的改进，实际上这里的 ant 即移动 agent，该策略充分考虑了切换对 QoS 的影响，根据每个节点的本地信息作路由判决，因此是分布式的，而且产生的路径是无环的，能够满足延时限制，同时避免链路拥塞。

文献^[125]针对单层卫星网络负载分布不均匀的特点，提出了一种适用于单层卫星网的基于移动 Agent 的动态路由算法(SDRA-MA)。该算法通过移动 Agent 在卫星节点间迁移，收集星际链路时延、卫星纬度等信息。当条件满足时，移动 Agent 往回迁移，并在每个中间卫星节点基于卫星地理位置计算所探测路径的代价、更新路由表。通过仿真证明该算法能够适应网络拥塞，实现负载平衡。

本文提出的算法正是利用了 agent 的移动性、自主性，将故障路由更新的复杂度降低，并减少路由更新开销。

5.3 改进的基于静态拓扑快照的路由策略

传统的星座路由策略是采用拓扑快照序列路由算法，利用卫星网络拓扑变化的规律性，将每个拓扑周期分成若干个静态拓扑快照，由地面计算生成的静态路由序列表上传至卫星，卫星按照拓扑变化的时间间隔进行路由表的更新，该算法有效降低了星上的计算复杂度，又充分利用了卫星网络拓扑特有的规律性。

但该算法同时又存在不能应对网络实时状态的缺点，因此，兼顾路由的自适应性与路由计算的复杂性是本节所要论述的主要问题。

5.3.1 静态拓扑快照分析

本文第二章 2.3 节对星座网络拓扑进行了定性的分析，主要是为了说明星座网络拓扑的特点，从而与分簇结构结合更紧密。本章主要侧重于路由算法，因此需要定量的分析，包括对拓扑快照的数量与时长、异轨链路的建链考虑等等。

卫星之间通过一定的星座关系构成的网络结构图称为 ISL 网络拓扑图。在卫星系统中，系统的 ISL 网络拓扑处于不断变化中，并没有精确模型。但是星座网络拓扑结构有其独特的拓扑特点，使得其路由策略有简化和变通的可能。如星座节点数固定、空间段运动周期性和星座运动的规律性和可预见性。

对于 (24, 3, 1) 的通信星座网络，每颗卫星有 4 条星间链路，即与同一个轨道面上的前后两颗卫星及左边和右边相邻轨道面上的两颗卫星构成星间链路。在正常情况下，卫星之间的相对位置关系在卫星运行过程中稳定不变。

根据给出的参数，在不考虑姿态偏差、轨道偏差等因素的情况下，同轨面相邻两星的距离、仰角和方位角在运行过程中几乎不改变。而卫星系统拓扑快照的产生是根据 ISL 的变化，而同轨卫星之间星间链路比较稳定，所以主要考虑异轨卫星之间的星间链路。

由于异轨卫星间的相对距离、方位角变化较为激烈，两颗异轨面邻星之间不能建立连续的星间链路。但根据天线的波束范围或机械式天线的转动能力，同时考虑相对距离、方位角的变化率等因素，在一个轨道周期内的某些时段，可以考虑和异轨卫星建立星间链路。建链的方式在 2.3 节中已经叙述，下面分析一下建链的条件。

表 5.3-1 一个轨道周期内 Sat11 和 Sat27 的相对运动参数表示了异轨面邻星相对运动特性的方位角变化速率、俯仰角变化速率以及距离变化速率等。

表 5.3-1 一个轨道周期内 Sat11 和 Sat27 的相对运动参数

参数	Sat11-Sat27		
时段	00:00~01:54	00:25~00:57	01:22~01:54
星间距离 R/km	4550.5~8164.4	4553~6840.7	4550.5~6898.1
方位角 $A/(\text{°})$	-66.9~66.9	-29.9~66.7	30.6~66.7
俯仰角 $Z/(\text{°})$	-16.9~-31.4	-26.1~16.4	-16.9~25.8

星间距离变化率/ (km/s)	-3.3~3.3	-2.9~2.9	-2.9~2.9
方位角变化率/ (°/s)	-0.078~0.078	0.078~0.0015	-0.078~0.006
俯仰角变化率/ (°/s)	-0.013~0.013	-0.012~0.012	-0.012~0.012

星间链路建立的条件考虑两点，一是星间距离，二是星间相对运动速度。根据目前设备研制水平及链路余量，考虑星间距离不超过 6900km 时可建立链路。而星间相对速度可由方位角变化率及俯仰角变化率反映，应在相对速度不太大的情况下建立链路。根据表中卫星对之间相对运动参数，分析得到 Sat11 在一个轨道周期内与 Sat27、Sat32 可分别在两个时间段建立链路。每个时段持续约 32 分钟。Sat11 与 Sat27 可以建立链路的第一个时间段为(以一个轨道周期 1 小时 54 分钟为参照)00:25~00:57 (单位：小时:分钟)，第二个时间段为 01: 22~01: 54；Sat11 与 Sat32 可以建立链路的时间段为 00:01~00:33 和 00: 58~01: 30。依此类推，可获得全部异轨链路在一个轨道周期内建立的时间段。

从表中可以看出，在异轨星间链路持续时间段内，卫星之间的方位角变化速率、俯仰角变化速率都很小，方位角和俯仰角范围都不大，根据现有的星间链路研制水平，天线完全能够跟踪上，具有可实现性。

如果忽略星间链路长度的影响，其网络拓扑在每一个时间间隔内是保持不变的，只有当星间链路关闭（或重新连接）或者卫星从其相邻轨道的左边运行至右边（或从其相邻轨道的右边运行至左边）时，网络拓扑才发生变化。为了叙述方便，把星座在一个时间间隔内保持不变的网络拓扑称为一张网络拓扑快照，每当增加或断开一条 ISL（星间链路），则认为产生一个新的“快照”。以此 24/3/1 星座为例，周期约为 114min。下面是用 STK 软件仿真得到的离散拓扑快照的结果，该图与本文第 2 章中表 2.3-3 是一致的。

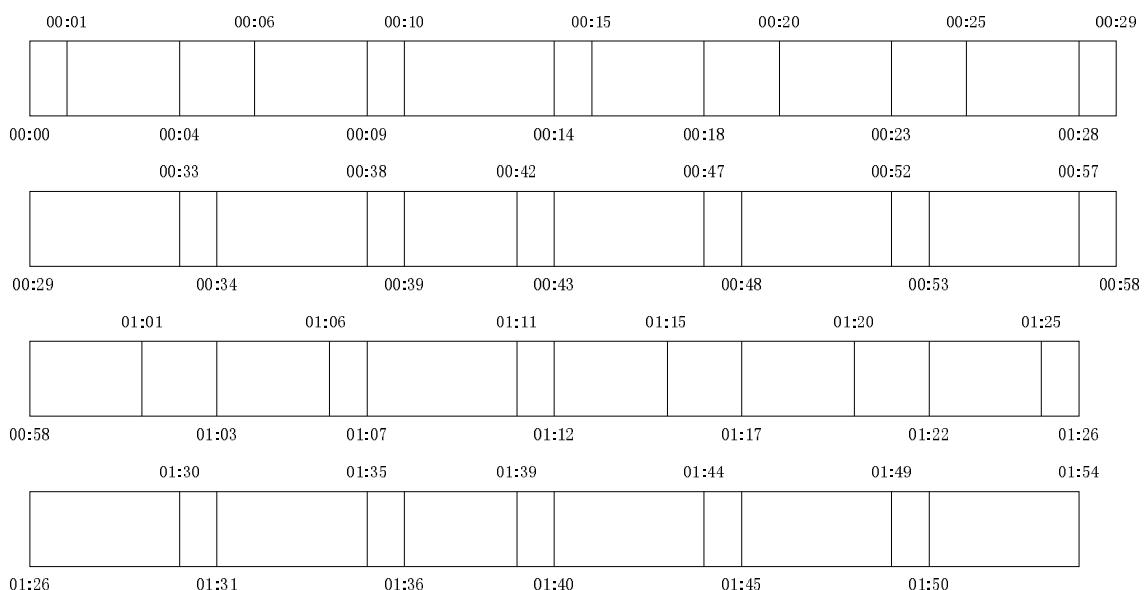


图 5.3-1 静态拓扑快照时间点分布图

根据快照拓扑的定义，在上图的时间点上都发生了星间链路的变化，所以都会产生新的快照。从图上可以看出，在一个轨道周期内，总共要产生 48 个拓扑快照。其中有 12 个快照持续 4 分钟，12 个快照持续 3 分钟，6 个持续 2 分钟，18 个快照持续 1 分钟。

因为考虑到地球自转，当一颗卫星回到地面上空的同一位置需经历一个卫星周期，该周期大概为 2 天，即 25 个轨道周期。所以，一个卫星周期的拓扑快照就是上面的一个轨道周期的拓扑快照的循环，总共需要 $25 \times 48 = 1200$ 个拓扑快照。

5.3.2 基于快照序列的路由策略

根据所使用的网络拓扑形式，LEO 卫星网络路由算法可以分为动态拓扑路由、静态拓扑快照序列路由两种。

动态拓扑路由算法依据实时的卫星星座拓扑进行路由，主要优点是对不同星座拓扑适应性强，能够对卫星失效、链路拥塞等情况做出及时反应；缺点是要求卫星节点之间频繁交换网络拓扑和链路信息，对星上处理能力要求较高，系统开销大。

对于卫星网络来说，节点的连接关系、传播延时可以预知。快照序列路由算法利用卫星网络拓扑结构的周期性，合理的将拓扑变化周期划分为一系列离散的时间片断，在每个时间片断内认为卫星网络的拓扑结构是固定不变的，从而将动态拓扑下的路由问题转换为静态拓扑下的路由问题，避免了卫星节点之间的拓扑信息交换，从而降低星上处理负担，减少信令开销；同时，由于卫星运行的可预测性，路由表可以预先在地面站计算后加载到卫星，卫星只需在时间段边界切换路由表即可，从而降低对星载 CPU 计算能力的要求。因此，考虑到卫星的工程可实现性，通常采用这种传统的静态拓扑序列路由策略。

星座系统中主要业务之一为语音业务，对提供服务的网络性能方面主要关注时延、通话质量、掉话率等等，其中时延性能是这种实时性强的业务最关注的指标，因此，传统的路由算法是以最短时延为计算准则的。

语音业务的时延一般指端到端时延，其定义为：从源用户到目的用户的传播时延、传输时延、排队时延和处理时延的总和。

传播时延：分组在卫星节点间的传播时间（距离/光速）。在上述星座中同轨星间链路距离约为 6000 km，异轨星间链路距离在 4900~6800 km 变化。近似认为各星间链路传播时延相等，约为 20 ms。

传输时延：业务分组大小和链路带宽的比值。

排队时延：指业务分组被交换到输出端口缓存区进行等候直到被输出的时延。

处理时延：卫星根据拓扑表和分组头部信息查找路由的时延。

由此可以看出，传输时延和排队时延与卫星网络链路的实时状态有关，采用离线计算路由方法时可不考虑它们对总时延的影响，而处理时延相对于卫星的处理能力来

说可以忽略不计，因此端到端时延的主要影响因素是传播时延，而传播时延实际上与路径的跳数是相对应的，最短时延路由算法就转化为寻找最小跳数的路径。

根据上述章节对网络拓扑快照的分析，卫星网络的拓扑结构就可以表示为一系列拓扑结构快照的循环，循环周期就是卫星回归周期。可以将卫星循环周期 T 分割成时间片 $[t_0, t_1], [t_1, t_2], \dots, [t_{n-1}, t_n]$ ，由地面信关站离线计算好每个时间片下的最短路由，并上传到卫星。

由于整网的路由表存储量太大，对星上设备的存储要求较高，我们采用局部路由表维护的策略，每个卫星节点仅维护与自己相关的一跳路由表，设星座系统中卫星的总数为 n ，将卫星节点从 1 到 n 进行编号， S_i 表示编号为 i 的卫星节点， $S_{i,N}$ 表示当前时刻 S_i 的邻居节点集合。那么，卫星节点 S_i 上存储的某一时间片 $[t_{n-1}, t_n]$ 内的路由表内容如下表所示：

表 5.3-2 路由表结构

目的节点 ID	下一跳节点 ID
$S_j (j \in \{1, 2, \dots, n\} \text{ 且 } j \neq i)$	$S_k (S_k \in S_{i,N})$

这种存储方式使卫星节点不必了解从源节点到目的节点的整个路由路径，只保存下一跳节点编号，便可以将星间数据正确路由，与整网路由表相比较存储量大大减少。

5.3.3 考虑切换和时延的权值路由算法

上一小节对传统的基于最短时延的快照序列路由策略进行了描述，在此基础上，本小节针对星座系统网络拓扑特有的动态性、规律性以及切换特性，提出考虑切换和时延的权值路由算法。

衡量路由算法性能优劣的参数包括端到端时延、掉包率、时延抖动等。选择路由策略的原则是在满足用户需求的基础之上尽量在这几个指标中达到一个平衡。

5.3.3.1 切换对时延抖动的影响分析

一次通话过程中涉及的切换过程可能包括波束间切换、星间切换和星间链路切换。其中，波束间切换是指用户在通信过程中由于卫星的移动造成用户从卫星一个波束的覆盖区切换到另一个临近波束的覆盖区，并不涉及分组业务的路由问题；星间切换是由源卫星或目的卫星改变所引起的，星间切换发生之后只需根据当前的路由表对新的分组业务采用新的路由即可，旧的分组业务可能会由于目的卫星改变而无法到达用户，从而引起掉包，但这种掉包情况不是由于路由本身的切换引起的；而星间链路切换是指在源卫星和目的卫星不改变的情况下由于星间拓扑发生改变而导致路由发生了切换，这种路由切换会导致掉包或时延加大等情况，是本文研究路由问题的重点。

因此，本文主要针对星间链路切换对语音通话性能的影响来研究合适的路由策略。下面来分析星间链路切换发生时对分组业务的影响。

当一个新的拓扑变化时刻到来时，网络中必定存在旧的星间链路断开或新的星间

链路建立的情况，如果某卫星节点上发生了星间链路切换，节点上本来需要通过旧的星间链路端口输出的分组数据包将面临三种情况：1) 分组尚未被交换到输出端口；2) 分组已经被交换到输出端口队列上等待传输；3) 在切换发生前分组已经通过旧的星间链路输出。

对于分组交换来说，这三种情况均可通过不同级别的备份存储策略对这些可能会丢失的分组进行重路由，然而重路由会造成一定的时延抖动。当然，也可在目的端采用排序策略对时延抖动进行平滑，但这必定引起总的时延增加；若考虑时延的影响，则可能会造成一定的掉包现象。下面我们来分析切换对时延抖动的影响。

时延抖动主要是由于切换导致路径不同使得分组到达目的端的时延不连续而引起的，前面论述过，本文只考虑端到端时延的主要组成——传播时延的影响，传播时延与节点之间的距离有关，同轨和异轨星间链路的距离计算分别如下：

$$d_{\text{intra}} = \sqrt{2}R \sqrt{1 - \cos\left(\frac{360}{M}\right)} \quad (\text{式 5.3-1})$$

$$d_{\text{inter}}(t) = \sqrt{2}R \sqrt{1 - \cos\left(\frac{360}{2 \times N}\right)} \times \cos(lat(t)) \quad (\text{式 5.3-2})$$

若源节点表示为 S ，目的节点表示为 V ，则 S 到 V 的路径表示为：

$\text{Path}(S, V, t) : S \rightarrow M_1 \rightarrow M_2 \dots \rightarrow M_k \rightarrow V$

其中， t 表示拓扑改变时刻， $M_1 \sim M_k$ 为中间卫星，表示该路径包含 $k+1$ 跳， $\text{Path}(S, V, t)$ 表示从 t 时刻到下一个拓扑改变时刻的时间片内 S 到 V 的路径。

t 时刻 S 到 V 的时延表示为：

$$D(S, V, t) = D(S, M_1, t) + \sum_{i=1}^{k-1} D(M_i, M_{i+1}, t) + D(M_k, V, t) \quad (\text{式 5.3-3})$$

下一个时刻 t' ，拓扑发生变化，假设 t' 时刻节点 M_r 和 M_{r+1} 之间的星间链路断开，则节点 M_r 上受此影响分组从 S 到 V 的时延表示为：

$$D(S, V, t') = D(S, M_1, t) + \sum_{i=1}^r D(M_i, M_{i+1}, t) + D(M_r, V, t') + \Delta T \quad (\text{式 5.3-4})$$

其中， ΔT 表示分组重路由处理时间。

则这两个不同的路径引起的时延抖动可表示为：

$$\begin{aligned} D_{\text{jitter}}(S, V, t \rightarrow t') &= |D(S, V, t) - D(S, V, t')| \\ &= |D(M_r, V, t) - D(M_r, V, t')| + \Delta T \end{aligned} \quad (\text{式 5.3-5})$$

由此可见，时延抖动主要是由 t 时刻和 t' 时刻不同拓扑下源节点 M_r 到目的节点 V 之间不同路径的时延差组成。

根据 5.3.1 节中对静态拓扑快照的分析，在每次拓扑改变时刻，仅有少数星间链路会发生断开或连通，大多数路径并不会受到影响。然而不同路由的选择造成的时延抖动的影响也有所不同，下面通过一个例子对此影响进行分析。

在拓扑区间[00:10, 00:14]，源卫星 S11 与目的卫星 S22 之间存在多条最小跳数路径，包括：

路径 1: S11→S12→S13→S14→S22

路径 2: S11→S12→S13→S21→S22

路径 3: S11→S32→S31→S38→S22

可以看出，影响路径持续性的异轨星间链路在路径 1 和路径 2 中为 1 条，在路径 3 中为 2 条。其中，路径 1 中链路 S14→S22 将在下一个拓扑时刻 00:14 到来时断开，而路径 2 和路径 3 在下个拓扑区间内仍然有效。

本文对采用不同路径在拓扑更新时刻的时延抖动进行了仿真。取拓扑变化时刻前后一段时间区间进行仿真，该区间取 160s，通常这为一次语音通话时长。包产生间隔为 0.8s，业务场景为理想情况下，仿真时，拓扑区间[00:10, 00:14]内路由选择为路径 1 和路径 2，拓扑更新后路由选择分别为路径 2 和路径 3，包时延性能如图 5.3-2 所示。

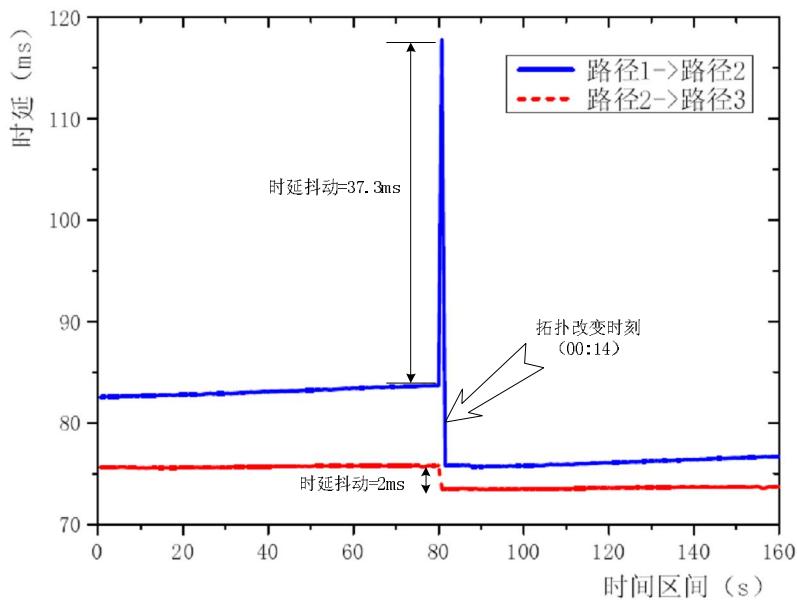


图 5.3-2 拓扑更新前后不同路由时延抖动性能的比较

由图可见，路径 1→路径 2 在拓扑更新后时延抖动非常大，而如果选择路径 2→路径 3，其时延抖动相对较小。这是由于，路径 1 在拓扑更新后失效，已经传输到节点 S14 上的分组必须通过其他路径到达目的节点，由此带来较大的时延抖动。而路径 2 在拓扑更新后仍然有效，拓扑更新前已经发出的分组仍然可以通过原有的链路传输到目的节点而不会产生大的时延抖动，图中较小的时延抖动是拓扑更新后源节点发出的分组切换到新的路由所造成的。

经过上述分析说明，路由的持续性是影响时延抖动的重要因素。在设计路由策略时，应充分考虑链路切换对路由接续能力的影响，从而优化路由选择。

5.3.3.2 权值路由表及路由选择原则

一般地，传统路由表如表 5.3-2 所示，包含目的节点、响应端口号（对应下一跳节点 ID），对于每一个目的节点都有唯一的对应选择。然而，这种做法没有考虑到流量平衡的因素，没有充分利用网络资源，同时还会造成单条链路的负担加重从而导致拥塞。

考虑流量平衡的路由表设置也可采用概率路由表，如图 5.3-3 所示，路由表中包含了针对不同的目的节点的相邻节点选择概率值，在路由表中不同的目的节点 d 和每一个不同的相邻节点 n 都有一个概率值 P_{nd} 与之相对。该 P_{nd} 值表示当目的节点为 d 时选取相邻节点 n 为下一个路由节点的概率值。路由节点 k 将根据用户数据的不同目的节点，依据此值为其选择下一个行程节点。这种做法能够平衡流量以防止网络拥塞，然而，在网络空闲的情况下却会造成部分数据分组时延增加、网络资源没有充分利用的情况，这主要是概率路由使得部分分组没有选择最佳路由引起的。

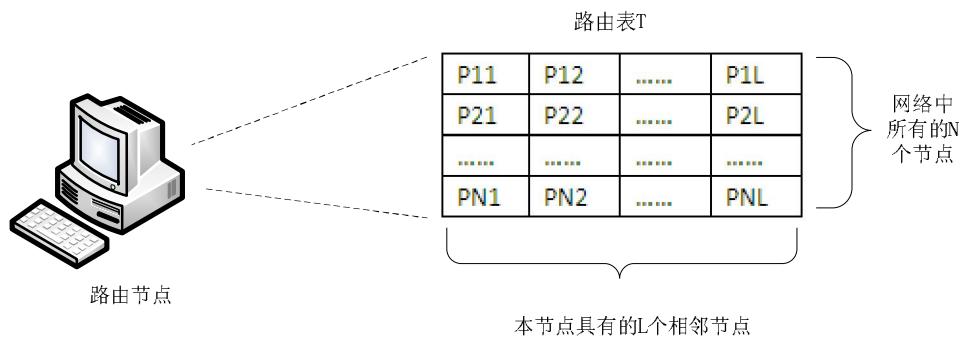


图 5.3-3 概率路由表

本文考虑一种权值路由表的方法，形式与概率路由类似，只是采用权值而不是概率来作为针对不同的目的节点的相邻节点选择度量工具。

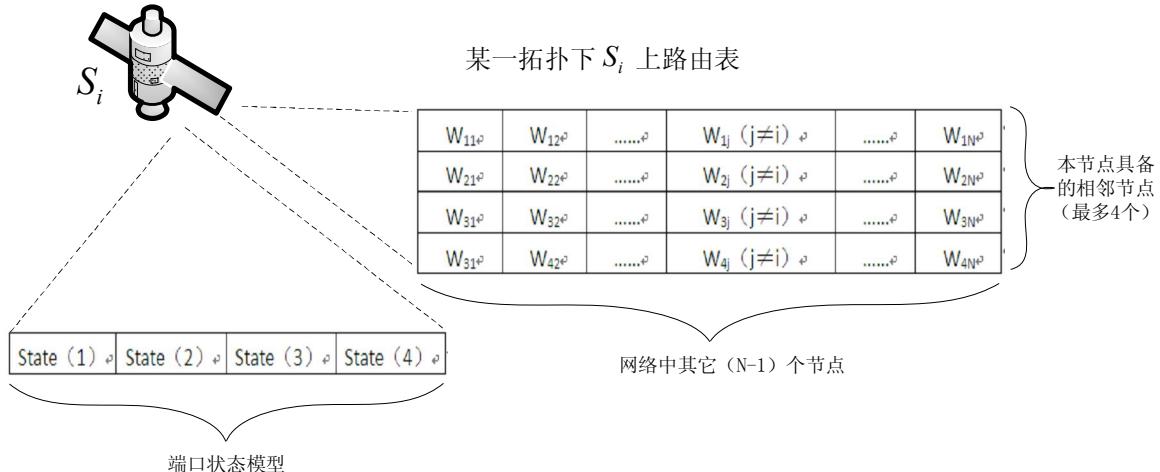


图 5.3-4 权值路由表

如图 5.3-4 所示,采用权值 W_{nd} 来表示节点 S_s 通过下一跳节点对应的端口 n 到达目的节点 S_d 的优先度, 在不同拓扑构型下 n 的取值不同, 对应不同的邻居节点, 权值 W_{nd} 的取值范围为[0, 1]。

这里权值代表了通过端口 n 到达目的节点的路径的优先级别,之所以采用权值而不是简单的顺序表示法来衡量其优先级,是因为顺序表示法无法预防拥塞,而只能在已经发生拥塞情况之后或者已经有拥塞倾向时才选择次优路由。采用权值表示法可以真实反映不同路由的代价,对代价相等的路由来说其公平性更好,同时它又可以兼顾概率路由能够进行流量平衡的优点。

为了克服静态拓扑序列路由不能及时对网络拥塞作出反应的缺点,本文引入一个模型: $state(n, t_k)$, 表示节点上每个端口在 $[t_k, t_{k+1}]$ 拓扑区间下的拥塞状态。

$$state(n, t_k) = \left(1 - \frac{q_n}{\sum_{n'=1}^{N_k} q_{n'}}\right) \times \frac{c_n}{\sum_{n'=1}^{N_k} c_{n'}} \quad (\text{式 5.3-6})$$

式中, q_n 表示通过端口 n 发往相邻节点缓冲区的数据包数量, c_n 表示通过端口 n 通往相邻节点的链路容量, N_k 表示拓扑区间 $[t_k, t_{k+1}]$ 下与 S_s 相连的星间链路数目。公式的前半部分为 q_n 与所有相邻节点缓冲区数据包总数的比值。后半部分表示 c_n 与所有相邻节点的链路容量总数的比值。该状态同时反映了链路的传输能力和节点缓冲区的即时状态,给出了一个即时的与队列等待时间有关的量化值。

由式可得, $state(n)$ 越小, 说明该端口越拥塞。路由选择时, 根据 $state(n, t_k)$ 与权值 W_{nd} 共同确定目的节点为 S_d 时的下一跳输出端口 n 。选择的原则在 5.3.3.3 中阐述。

5.3.3.3 基于动态选择参数的权值路由算法

本路由算法的目的是根据不同路径的代价计算出通过节点上每个端口到达不同目的节点时的分配权值,并确定不同权值路由的选择策略。

(1) 路径代价计算

首先,需要计算不同路径的代价。任何一条路径都是由多跳星间链路组成。设在拓扑区间 $[t_k, t_{k+1}]$ 下两个相邻卫星节点 S_i 、 S_j 之间的星间链路用于路由的代价为 $C(S_i, S_j, t_k)$ 。显然,传播时延是路由代价的主要因素,而传播时延主要是由两个卫星节点之间的距离决定。除此之外,通过前述章节的分析,切换是影响路由接续性的一个重要因素,映射到链路上来说,同轨星间链路为永久链接,异轨星间链路为暂时性链接,因此,减少路径上异轨链路的数量能够增加路由的接续性。

这里引入一个参数 λ 来表示链路的接续能力。 $C(S_i, S_j, t_k)$ 可表示为:

$$C(S_i, S_j, t_k) = D(S_i, S_j, t_k) \times \frac{1}{\lambda(S_i, S_j, t_k)} \quad (\text{式 5.3-7})$$

这里 $D(S_i, S_j, t_k)$ 表示拓扑 T_k 下相邻两节点之间的传播时延。 λ 取值范围为 [0, 1]，这里采用指数函数来表示参数 λ ，如（式 5.3-8）。

$$\lambda(S_i, S_j, t_k) = e^{-\left(1 - \frac{T_{res}(S_i, S_j, t_k)}{T_{tol}}\right)} \quad (\text{式 5.3-8})$$

λ 是与当前拓扑区间 $[t_k, t_{k+1}]$ 起始时刻 t_k 到该条星间链路断开时刻的剩余时间 $T_{res}(S_i, S_j, t_k)$ 与一个轨道周期总时间 T_{tol} 的比例呈指数关系的。

则源节点 S_i 与目的节点 S_d 之间的路径总代价可表示为：

$$\begin{aligned} TC_{s \rightarrow d}(t_k, p) &= D(S, M_1, t_k) \times \frac{1}{\lambda(S, M_1, t_k)} \\ &+ \sum_{i=1}^{hop(p)-2} \left\{ D(M_i, M_{i+1}, t_k) \times \frac{1}{\lambda(M_i, M_{i+1}, t_k)} \right\} \\ &+ D(M_{hop(p)-2}, V, t_k) \times \frac{1}{\lambda(M_{hop(p)-2}, V, t_k)} \end{aligned} \quad (\text{式 5.3-9})$$

其中， p 表示源节点 S_i 与目的节点 S_d 之间的第 p 条可能路径， $hop(p)$ 表示第 p 条路径的总跳数。

（2）有效路径的筛选准则

源节点 S_s 与目的节点 S_d 之间存在多条可达路径，如前所述，算法首先计算这些路径的代价，并使其参与到权值的计算中，最终共同确定每个端口上权值的大小。这时，参与计算的路径有效性对权值的大小起到重要的作用，这里的有效性指的是可选路径的代价应在优选路由可接受的范围之内。

为了获得有效路径，必须对多条可达路径进行筛选。筛选后获得 x 条路径参与后续的计算，显然，这 x 条路径的代价必须是所有可达路径中相对较小的。对于 x 的取值通常可采用两种方法，一是取固定值，这种方法比较简单，即将所有可达路径的代价从小到大排列，取前 x 个。但是对于源节点和目的节点之间距离比较近的情况来说，会导致明显大于最短时延的路径代价在参与计算时占有主导地位，例如相邻节点之间计算路由的情况，这显然不合理；二是根据源节点和目的节点之间的距离不同取不同的值，使得参与计算的路径数量与源节点和目的节点之间的距离成比例，以避免明显不合理的路径对计算结果的影响因子过大的情况，不过这种方法使得不同节点对的路由计算复杂度各不相同，同时在去除不合理路径时也无法采用统一的判别标准。

本文结合上述两种方式的优点，首先对 x 取固定值 N_{pre} ，再利用误差理论，对路径代价求加权均值及加权方差，来剔除 x 条路径中的不合理路径，从而在降低计算复

杂度的同时，抑制不合理路径的影响。

这里固定取 N_{pre} 条路径参与计算，且有

$$TC_{s \rightarrow d}(t_k, 1) < TC_{s \rightarrow d}(t_k, 2) < \dots < TC_{s \rightarrow d}(t_k, N_{pre}) \quad (\text{式 5.3-10})$$

引入两个加权因子 ω_p 和 Q_p ， ω_p 表示路径的大小排列次序的权重，可根据需要取一组合适的固定值。 Q_p 则表示了对 ω_p 的影响因子，表达路径代价与最小路径代价之间的关系，采用非线性函数表达，如式 (5.3-11)：

$$Q_p = 1 - \frac{1}{1 + e^{-\alpha}} \quad (\text{式 5.3-11})$$

其中，

$$\alpha = \frac{TC_{s \rightarrow d}(t_k, p) - \min(TC_{s \rightarrow d}(t_k, p))}{\min(TC_{s \rightarrow d}(t_k, p))} \quad (p = 1, 2, \dots, N_{pre}) \quad (\text{式 5.3-12})$$

根据误差理论，可计算源节点 s 与目的节点 v 之间粗筛后路径代价的加权平均值为：

$$\mu(TC_{s \rightarrow d}) = \sum_{p=1}^{N_{pre}} \omega_p Q_p TC_{s \rightarrow d}(t_k, p) \left/ \sum_{p=1}^{N_{pre}} \omega_p Q_p \right. \quad (\text{式 5.3-13})$$

加权方差：

$$\sigma(TC_{s \rightarrow d}) = \sqrt{\sum_{p=1}^{N_{pre}} \omega_p Q_p (TC_{s \rightarrow d}(t_k, p) - \mu(TC_{s \rightarrow d}))^2} \left/ (N_{tol} - 1) \sum_{p=1}^{N_{pre}} \omega_p Q_p \right. \quad (\text{式 5.3-14})$$

剔除不合理路径的原则是，如果存在

$$TC_{s \rightarrow d}(t_k, p) > \mu(TC_{s \rightarrow d}), \text{ 且 } |TC_{s \rightarrow d}(t_k, p) - \mu(TC_{s \rightarrow d})| > \sigma(TC_{s \rightarrow d}) \quad (\text{式 5.3-15})$$

则在后续计算中不包含第 p 条路径。

(3) 权值计算

权值计算首先按照各路径的代价计算每条路径所占的权重，再映射到端口，通过求和计算端口的权重。

设经过筛选，获得 N_{tol} 条合理路径。本算法设计代价越小的路径获得越高的权值，计算如下：

$$W_{s \rightarrow d}(t_k, p) = \left(1 - \frac{TC_{s \rightarrow d}(t_k, p)}{\sum_{p=1}^{N_{tol}} TC_{s \rightarrow d}(t_k, p)} \right) \left/ \sum_{p=1}^{N_{tol}} \left(1 - \frac{TC_{s \rightarrow d}(t_k, p)}{\sum_{p=1}^{N_{tol}} TC_{s \rightarrow d}(t_k, p)} \right) \right. \quad$$

(式 5.3-16)

路径权值 W_p 满足：

$$\sum_{p=1}^{N_{tol}} W_{s \rightarrow d}(t_k, p) = 1 \quad (\text{式 5.3-17})$$

将第 p 条路径下一跳节点对应的端口表示为 $\text{Next_hop}(p)$, 则对应端口权值为：

$$W_{nd}(t_k) = \sum_{\text{Next_hop}(p)=n} W_{s \rightarrow d}(t_k, p) \quad (\text{式 5.3-18})$$

端口权值 $W_{nd}(t_k)$ 仍满足：

$$\sum_n W_{nd}(t_k) = 1 \quad (\text{式 5.3-19})$$

(4) 路由选择

本算法最重要的特点便是星上路由动态选择，即结合端口的权值，根据当前端口的实时拥塞状态，确定分组的下一跳节点。实际上，是在固定权值的条件下，实时地选择路由，这与动态路由算法根本的区别在于，动态路由算法是实时地计算路由，而本文的算法是实时地计算路由选择参数。这也使得本算法能够在保证星上路由计算低复杂度的同时，兼顾链路的实时状态，从而避免拥塞、平衡流量。

这里引入参数 ρ , 用于衡量端口状态 $\text{state}(n, t_k)$ 与路由表中存储的端口权值 $W_{nd}(t_k)$ 相比的重要性。对于每个到达节点上的分组，其路由的选择通过一个选择指数表示：

$$E\eta_{nd}(t_k) = W_{nd}(t_k) + \rho \text{state}(n, t_k) \quad (\text{式 5.3-20})$$

选择指数综合考虑了权值和链路状态对路由选择的影响，每次分组到来时，首先查询每个端口的选择指数，选择指数最大的端口作为分组的输出端口。

参数 ρ 的取值应根据问题的特性而不同，如果取值太小，就无法体现端口状态 $\text{state}(n, t_k)$ 的作用；取值太大的话，权值路由表的作用也无法发挥。这两种情况都会导致性能的降低。

5.3.3.4 仿真分析

仿真模型中设置卫星节点星间链路缓冲区总容量为 40kB, 其中包括 4 个端口缓冲区，分别对应 4 条星间链路，平均每个缓冲区容量为 10kB。仿真主要针对语音业务，设置每次通话时间持续为 160s，通话期间平均 0.2s 产生一个包，分组长度为 272 个字节。星间链路带宽设置为 2Mbps。

背景流量设置为：仿真网络空闲状态时设置网络中 30% 的节点以 10kbps 速率发送报文，目的节点随机选择，30% 节点接收报文。仿真网络繁忙状态时设置为：网络

中 30%的节点以 1Mbps 速率发送报文，目的节点随机选择，30%节点接收报文。

在上述仿真条件下，分别对快照序列路由算法和本文提出的考虑时延和切换的权值路由算法进行仿真，参数 ρ 取 0.35，仿真设置 2 个场景分别从时延抖动、时延/掉包率两个方面进行性能比较。

仿真一：时延抖动性能。

仿真场景：网络空闲状态下，仿真源用户和目的用户之间一次 160s 的通话过程，160s 覆盖整个轨道周期时间，源节点固定，目的节点覆盖所有节点对。统计每一对节点对之间通信时的时延抖动值，并计算不同时延抖动值的分布情况。

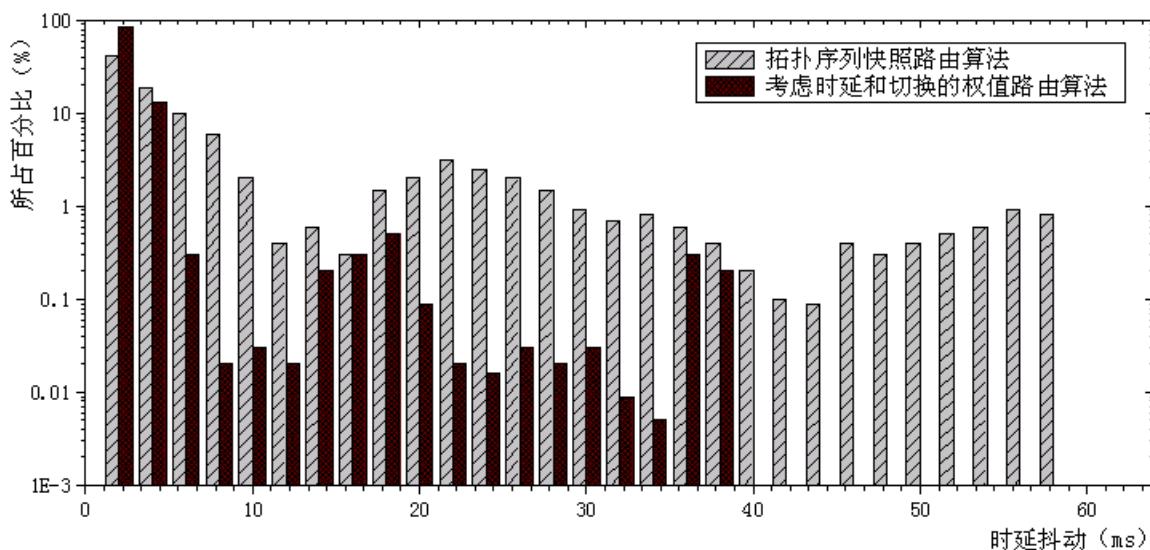


图 5.3-5 网络空闲条件下，一个 160s 通话中源节点 S11 到所有可能目的节点的时延抖动分布情况

图 5.3-5 中表示在一个轨道周期内，源节点为 S11，目的节点为网络中所有其它节点，一个 160s 语音通话中由切换带来的时延抖动分布情况图。对仿真数据进行统计，可得出两者的性能比较表：

表 5.3-3 时延抖动分布区间比较表

时延抖动值 (ms)	分布情况 (%)	
	快照序列路由算法	考虑时延和切换的权值路由算法
2ms 以下	42%	85%
4ms 以下	18%	13%
4ms~20ms	23%	1.4%
20ms~40ms	13%	0.6%
40ms~60ms	4%	0
60ms 以上	0	0

表 5.3-3 表明，快照序列路由算法的最大时延抖动大于 58ms，4ms 以下的时延抖动占 60%，而本文的考虑时延和切换的权值路由算法的最大时延抖动大于 38ms，同

时 4ms 以下的时延抖动占 98%。由此可见，本文算法的时延抖动性能明显比较优越。

同样的仿真过程，在网络繁忙时的时延抖动性能如图 5.3-6 所示。

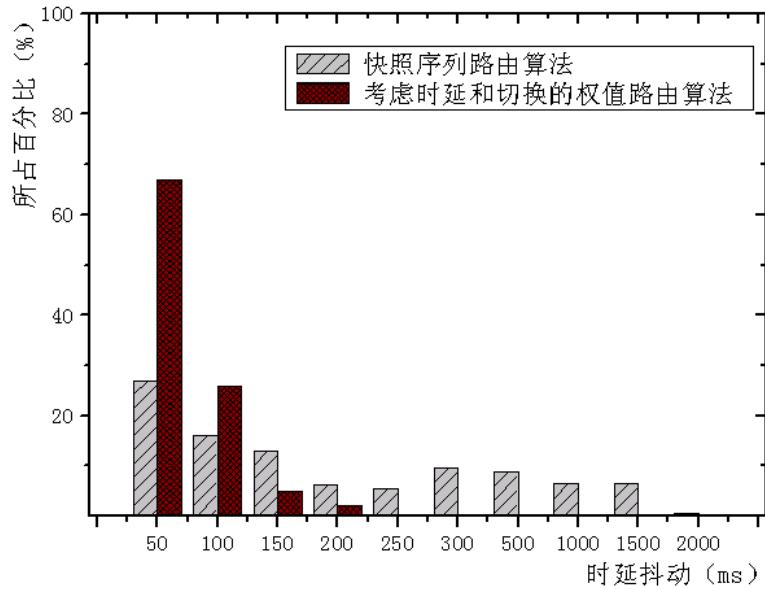


图 5.3-6 网络繁忙条件下，一个 160s 通话中源节点 S11 到所有可能目的节点的时延抖动分布情况

仿真结果显示了本文算法在网络繁忙条件下同样具备性能优势，虽然由于网络拥塞造成时延抖动的增加，但是与传统快照序列路由算法最大时延抖动达 2000ms 相比，本文算法最大时延抖动只有 200ms 左右。这主要是由于本文的权值路由算法可以根据网络拥塞状态对业务进行分流，使得次优甚至次次优路由承担了相当一部分业务转发任务，从而使拥塞带来的时延抖动增大情况得到缓解。

仿真二：网络繁忙下的时延性能及掉包率。

仿真场景：网络繁忙情况下两个固定位置用户进行语音通信，仿真 5 小时的源用户到目的用户的端到端时延。

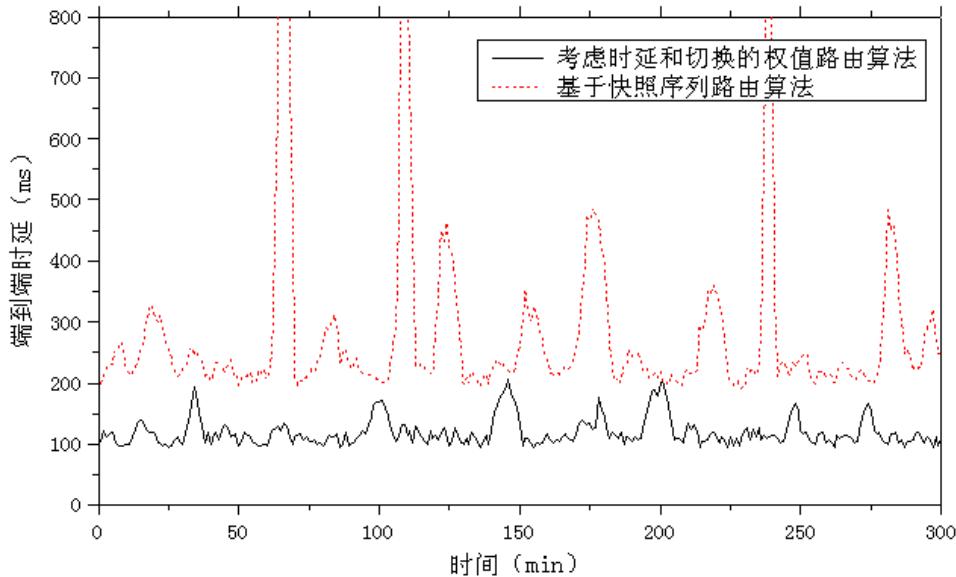


图 5.3-7 网络繁忙下的时延性能

由图可以看出，在网络繁忙条件下，本文算法的时延性能明显优于基于快照序列路由算法。基于快照序列路由算法由于没有考虑流量平衡机制，业务集中于一部分星间链路，引起较大时延抖动，导致最大时延达到 2s（图中未显示），并由此引起掉包（掉包率为 4.3%）。而本文算法采用权值路由表的方式对业务进行分流，并能够根据端口实时状态进行业务路由的调整，因此能够在网络繁忙的状态下保持较好的时延性能，从仿真结果看其时延保持在 200ms 以内，没有掉包现象。

5.4 基于移动 agent 的路由更新算法

星座自主运行时各种业务数据以及测控管理数据需要通过星间链路进行转发，在网络发生异常时必须及时更新路由，以避开故障链路，保证用户的服务质量。

本文提出的权值路由算法无需卫星进行路由计算，降低了星上处理复杂度，同时又通过权值分配结合动态选择的路由策略使得该算法在最佳路由和链路状态之间获得平衡。但是当星座在自主运行过程中出现链路故障时，它仍然存在不能对网络拓扑异常变化做出快速反应的缺点。

如果在链路故障后不对路由表进行更新，节点间的路由仍然按照原有的进行，则会导致两种情况：

(1) 受链路故障直接影响的节点由于链路总容量减小，导致分组的排队时延加长，从而引起比较高的掉包率；

(2) 受链路故障间接影响的节点由于无从获知故障信息，仍然按照原有的路由表操作，可能导致大量分组仍旧被交换到故障节点上的情况，最后使故障节点成为网络中的瓶颈，导致更长的交换时延和更高的掉包率。

为了避免上述情况，有必要对路由表进行更新。

这种路由表的存储方式虽然能够使卫星节点不必了解从源节点到目的节点的整

个路由路径，只保存下一跳节点编号，便可以将星间数据正确路由，与整网路由表相比较存储量大大减少，但是当自主运行期间发生星间链路异常情况时，星上必须自主更新每颗星上存储的路由表，以避开故障链路，降低数据的丢包率。这要求簇管理者获得整网的拓扑关系，进行计算，并将结果告知受影响的链路节点。这样做的结果是网络开销增加、星上计算代价增加，而星上的计算资源和存储资源以及链路带宽资源等通常都是受限的，因此这种做法不适合卫星网络的特点。另外，由于整网传输拓扑信息，时延较大，也不能满足路由更新实时性的需求。

因此自主路由管理便是为了解决上述问题而提出的，是本文的研究内容之一。本文提出采用移动 agent 进行局部路由探测的方法，进行路由的更新，能够在不显著增加星上计算量的情况下，又能保证链路异常时对静态拓扑路由进行更新，同时还要兼顾网络带宽受限的约束，有效降低路由更新带来的开销。

5.4.1 故障链路对路由性能的影响分析

首先分析一下故障链路对于节点路由的影响。假设节点 S_f 与 $S_{f'}$ 之间的星间链路出现故障，节点 S_f 上拓扑区间 $[t_k, t_{k+1}]$ 下路由表如表 5.4-1 所示：

表 5.4-1 节点 S_f 路由表

目的节点 端口 \ S _i	S ₁	S ₂	S _{i(i ≠ f)}	S _N
1	W ₁₁	W ₁₂	W _{1i}	W _{1N}
2	W ₂₁	W ₂₂	W _{2i}	W _{2N}
3	W ₃₁	W ₃₂	W _{3i}	W _{3N}
4	W ₄₁	W ₄₂	W _{4i}	W _{4N}

根据前述章节对权值路由算法的论证，某一拓扑区间下，某个节点通过某个端口到达某目的节点的权值实际上反映了该节点通过该端口到达目的节点的路由代价大小。因此，这里可以用权值来表示网络中任意一节点通过 S_f 到达目的节点的路径代价。将网络中节点上的权值扩展表示为 $W_{nd}(t_k, S_i)$ ，表示拓扑 t_k 时刻节点 S_i 上通过端口 n 到达目的节点 S_d 的权值。假设 S_f 上的故障端口为 f ($f \in \{1, 2, 3, 4\}$)，则受影响的权值项为不等于 0 的 $W_{fx}(t_k, S_f)$ 。

设 S_{h1} 为 S_f 的 1 跳邻居节点，则故障后 S_{h1} 通过邻居节点 S_f 到达目的节点 S_x 的代价可表示为：

$$C_{h1 \rightarrow f \rightarrow x} = \underbrace{W_{jx}(t_k, S_{h1})}_{j \leftrightarrow S_f} \cdot [1 - W_{fx}(t_k, S_f)] \quad (\text{式 5.4-1})$$

设 S_{h2} 为 S_f 的 2 跳邻居节点，则故障后 S_{h2} 通过邻居节点 S_{h1} 到达目的节点 S_x 的代价可表示为：

$$C_{h2 \rightarrow h1 \rightarrow x} = \underbrace{W_{rx}(t_k, S_{h2})}_{r \leftrightarrow S_{h1}} \cdot \left[1 - \underbrace{W_{jx}(t_k, S_{h1}) \cdot W_{fx}(t_k, S_f)}_{j \leftrightarrow S_f} \right] \quad (\text{式 5.4-2})$$

由此，则 S_f 的 H 跳邻居节点通过其 $(H-1)$ 跳邻居节点到达目的节点 S_x 的代价可表示为：

$$C_{H \rightarrow (H-1) \rightarrow x} = \underbrace{W_{ix}(t_k, S_{hH})}_{i \leftrightarrow S_{h(H-1)}} \cdot \left[1 - W_{fx}(t_k, S_f) \cdot \prod_{a=1}^H \underbrace{W_{ix}(t_k, S_{ha})}_{j \leftrightarrow S_{h(a-1)}} \right], (S_{h0} \leftrightarrow S_f) \quad (\text{式 5.4-3})$$

由上述分析可得，距离故障链路越近的节点受到的影响越大，距离较远的则受到的影响越小，据此，本文提出了基于移动 agent 的多层扩散路由更新算法，控制移动 agent 的扩散范围，仅对受故障影响较大的节点路由表项进行更新，从而降低路由更新的代价、减小链路故障影响、缩短路由更新时延等等。

5.4.2 基于移动 agent 的多层扩散路由更新算法

移动 agent 的扩散路径如图 5.4-1 所示。

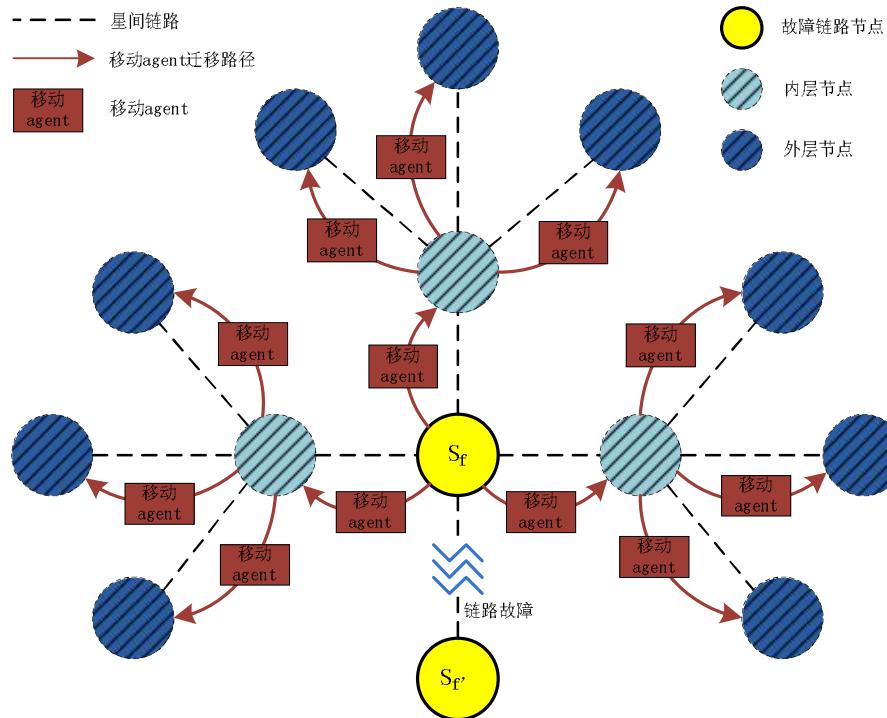


图 5.4-1 移动 agent 迁移过程示意图

图中只表示了 2 层扩散路径，即移动 agent 的生命周期为 2 跳，路由更新也只涉及故障节点的 2 跳邻居节点。本文的后续仿真都是基于 2 层扩散机制进行的，这是由于层数越多，网络开销越大，并且距离故障跳数越多的节点路由更新对网络性能的提升效果并不明显。

当链路出现故障时，由簇管理者发送消息给存在链路故障的节点，启动基于移动

agent 的多层扩散路由更新算法。

首先，作以下几个定义和约定：

定义节点 S_i 的 H 跳邻居节点集合为 $S_{i,H}$ ，约定下面的算法都是基于某一拓扑区间 $[t_k, t_{k+1}]$ ，并且在执行算法的过程中拓扑不发生变化。

移动 agent 的多层扩散路由更新流程如下：

(1) 存在链路故障的节点 S_f 生成一个其生命周期为 H 的初始移动 agent F_H ， H 表示移动 agent 迁移的跳数，初始移动 agent 的堆栈为空。

(2) 移动 agent 在当前节点上收集受链路故障影响的路由表项；

若移动 agent 为初始状态，设 S_f 上与故障链路对应的端口为 f ，则受链路故障影响的路由表项包括：

表 5.4-2 节点 S_f 上受链路故障影响的路由表项

目的节点	下一跳节点映射的端口	权值
S_x	f	$W_{fx} (\neq 0)$

移动 agent 查找当前节点上的路由表以及当前节点拓扑关系表，当表项中下一跳节点映射的端口为 f ，且其对应的权值 $W_{fx} \neq 0$ ，则将该路由表项对应的目的节点 ID、下一跳节点 ID、受损权值压入堆栈 $SS(x)$ ，堆栈结构如下：

表 5.4-3 堆栈 $SS(x)$ 初始值

下一跳节点	权值
$S_{f'}$	$W_{fx} (\neq 0)$

若移动 agent 不是初始状态，设当前节点为 S_j ，这里 $S_j \in S_{f,H}$ ，与 S_r ($S_r \in S_{f,H-1}$) 对应的端口为 j ，则其上受链路故障影响的路由表项包括：

表 5.4-4 节点 S_j 上受链路故障影响的路由表项

目的节点	下一跳节点映射的端口	权值
S_x	j	$W_{jx} (\neq 0)$

移动 agent 查找当前节点上的路由表以及当前节点拓扑关系表，当表项中目的节点为 S_x ，下一跳节点映射的端口为 j ，且其对应的权值 $W_{jx} \neq 0$ ，则将该路由表项对应下一跳节点 ID、受损权值压入堆栈 $SS(x)$ ，堆栈结构如下：

表 5.4-5 堆栈 $SS(x)$ 结构

下一跳节点	权值
$S_{f'}$	$W_{fx}(t_k, S_f)$
S_f	$W_{jx}(t_k, S_j), S_j \in S_{f,1}$
$S_r, S_r \in S_{f,1}$	$W_{rx}(t_k, S_r), S_r \in S_{f,2}$
.....
$S_p, S_p \in S_{f,H-1}$	$W_{qx}(t_k, S_q), S_q \in S_{f,H}$

(3) 移动 agent 收集完节点路由信息后, 通知路由节点更新路由, 且移动 agent 的生命周期 $H=H-1$;

当前节点为故障链路相连节点 S_f , 则目的节点为 S_x 的路由表项中权值更新为:

$$\begin{aligned} W_{fx}' &= 0 \\ W_{nx}' &= \frac{W_{nx}}{\sum_n W_{nx}}, \quad n \neq f \end{aligned} \quad (\text{式 5.4-4})$$

当前节点为 S_f 的 h 跳邻居节点 S_j , 则目的节点为 S_x 的路由表项中权值更新为:

$$W_{jx}' = \frac{C_{h \rightarrow (h-1) \rightarrow x}}{C_{h \rightarrow (h-1) \rightarrow x} + \sum_n W_{nx}(t_k, S_j)}, \quad n \neq j \text{ 且 } S_j \in S_{f,h} \quad (\text{式 5.4-5})$$

$$W_{nx}' = \frac{W_{nx}}{C_{h \rightarrow (h-1) \rightarrow x} + \sum_n W_{nx}(t_k, S_j)}, \quad n \neq j \text{ 且 } S_j \in S_{f,h} \quad (\text{式 5.4-6})$$

(4) 当前路由节点复制移动 agent, 并修改移动 agent 的迁移路径, 变化为多个新的移动 agent $F_{j \rightarrow r}$ (S_j 为当前节点且 $S_j \in S_{f,h}$, S_r 为 S_j 的邻居节点, 且 $S_r \notin S_{f,h-1}$)。新生成的移动 agent $F_{j \rightarrow r}$ 则携带堆栈 SS(x) 迁移至目的节点 S_r , 其生命周期仍然为 H 。

新的移动 agent 的数量在 1~3 内, 该数量与节点当前存在的星间链路个数相一致。

(5) 重复步骤 (2)、(3)、(4), 直到两种情况的发生, 才停止移动 agent 的复制与迁移: 一是, 当 $H=0$ 时; 二是, 当前节点 ID 与堆栈中下一跳节点有重复时。当这两种情况中任一发生时, 停止移动 agent 的复制与迁移, 通知当前节点更新路由表, 更新算法同步步骤 (3), 最后移动 agent 消亡;

该算法的优点包括: (1) 利用了移动 agent 的特性, 将管理者的计算任务转换为局部的路由信息收集任务, 大大减轻了管理者的计算负担; (2) 利用 agent 进行局部的路由信息收集, 可以不必得到整网的拓扑关系, 避免了簇管理者之间的拓扑信息交换, 有效降低了网络开销; (3) 该算法简单灵活, 能够应对网络拓扑异常变化的动态性, 非常适合资源受限、具备动态性、有自主性需求的卫星网络。

5.4.3 基于移动 agent 的系统设计

根据上述流程的描述, 基于移动 agent 的系统需包含 3 个模块: 路由节点 agent、路由 agent 发生器、路由移动 agent。路由节点 agent 和路由 agent 发生器属于服务由路由, 由节点本地生成, 并不会移动到其他节点。路由移动 agent 由路由 agent 发生器生成, 在节点中移动, 并在生命周期结束时消亡。

(1) 路由节点 agent

路由节点 agent 存在于网络中的每个路由节点, 是本地服务 agent, 负责与路由移动 agent 进行交互、监听路由更新信息并完成对节点路由信息的更新, 此服务 agent 由本地节点启动。外来的路由移动 agent 可以通过路由节点 agent 间接访问路由信息, 而不能直接访问, 这样可以防止恶意的外来 agent 修改信息。对本地节点路由信息的

更新是根据收到的路由移动 agent 发送的更新消息，由路由节点 agent 完成。

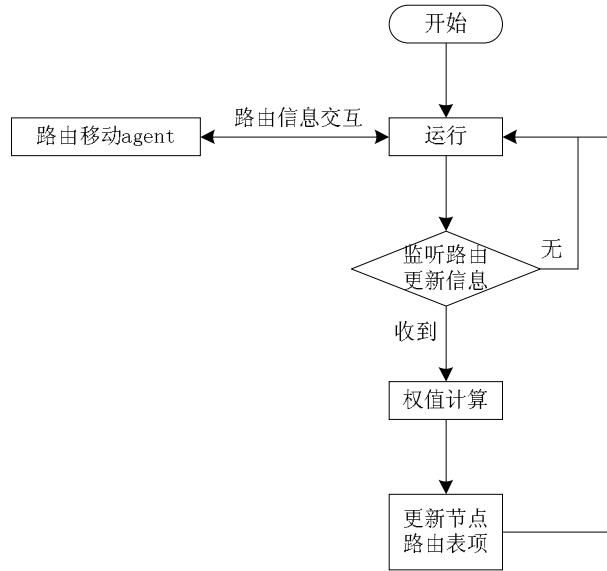


图 5.4-2 路由节点 agent

(2) 路由 agent 发生器

路由 agent 发生器在每一个需要产生路由 agent 的节点上运行，完成移动 agent 的初始化、生成、复制、参数修改等。

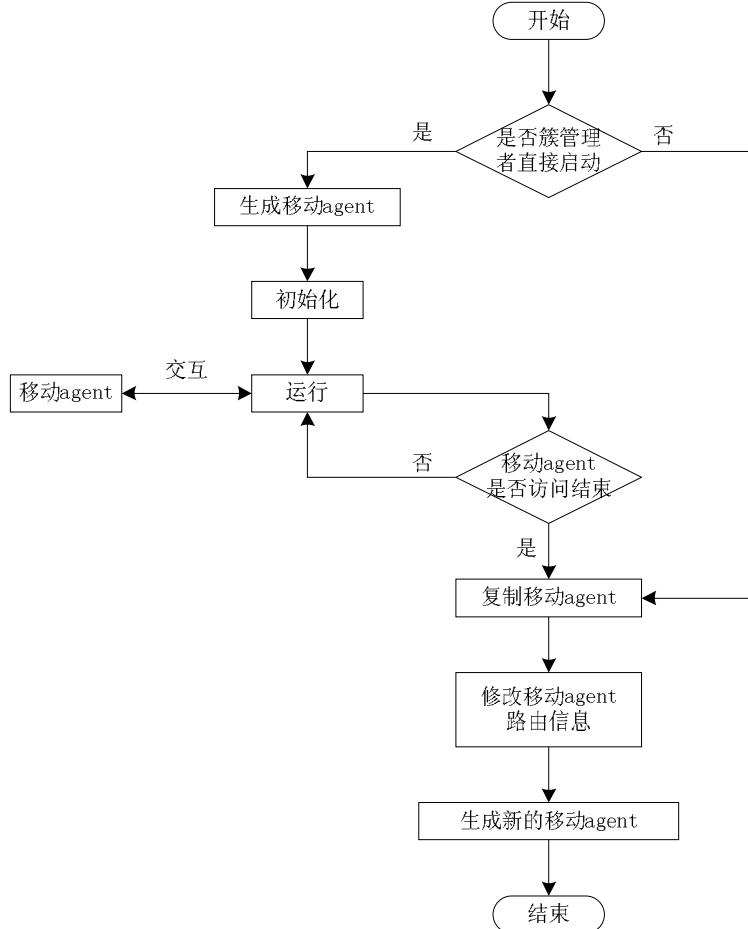


图 5.4-3 路由 agent 发生器

(3) 路由移动 agent 的实现

本算法中的路由移动 agent 主要功能为收集节点上受故障链路影响的路由表项。它通过访问路由节点 agent，根据 5.4.2 节中的规则，获得这些路由表项，并将相关信息压入相应的堆栈空间；同时，当路由移动 agent 收集信息结束后，将在前面行程中收集到的数据发送给当前节点，供当前节点更新路由信息之用。

路由移动 agent 的消亡根据当前的生命周期自主进行。

移动 agent 是一个包含代码、数据以及执行语境的软件包。在移动的过程中保存了它自身所有的状态。基于移动 agent 的系统必须拥有能够解读 agent 的解释模型和运行环境的支持。本文仅仅是为了解决故障路由自主更新的问题，而引入移动 agent 的概念，利用移动 agent 的自主性、通信性以及移动性达到降低路由更新代价的目的，对系统具体的实现并不涉及。

5.4.4 仿真分析

本节所提出的基于移动 agent 的多层扩散路由更新算法，主要为解决网络出现故障时的路由更新问题，因此仿真主要针对路由算法收敛情况、路由更新所需开销、更新路由后的时延性能等方面进行。比较的基础是传统的集中式路由更新算法，即路由更新需要收集网络中节点的拓扑信息，对路由进行重计算。

(1) 路由更新收敛时间

路由算法的收敛时间比较结果如图 5.4-4 所示。这里，算法的收敛时间指全网节点路由表重新计算更新所需的总时间，包含路由信息的交换时间、路由计算与处理的时间等等。

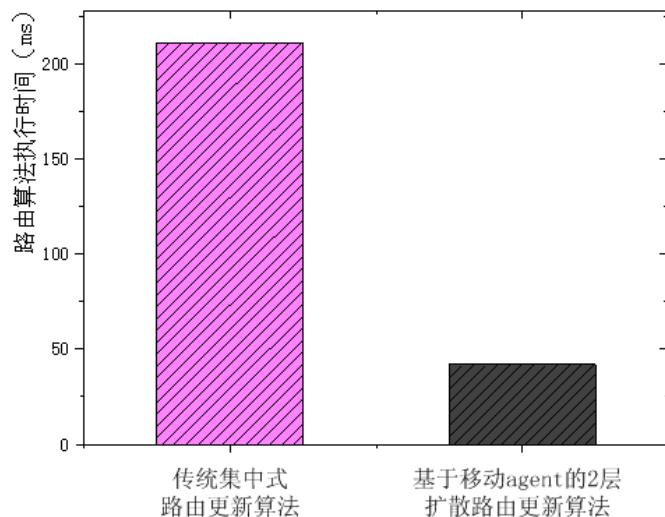


图 5.4-4 路由更新算法收敛时间比较图

从仿真结果来看，基于移动 agent 的 2 层扩算路由更新算法收敛地更快，这是因为它在进行路由更新时，只需更新故障节点附近的节点路由表，需收集和传输的路由信息比较少，传输的时延较小，计算处理的复杂度低，因此其总的更新时延较小。而

传统的集中式路由更新算法需要收集全网节点的路由信息，交换的时延较大，计算处理的复杂度也高，因此整个算法收敛地较慢。

(2) 路由更新开销

仿真时长为一个卫星轨道周期，链路故障随机设置 1 条，路由更新算法的开销取多次仿真的平均值，本文算法与传统集中式路由更新算法的比较结果如图 5.4-5 所示。本文算法中 agent 压入堆栈的每一条受损路由信息大小为 17bit，而集中式路由更新算法中每个节点上的拓扑信息大小为 48*15bit，而计算完成后需发送给每个节点的路由更新信息大小为 48*23*5bit。

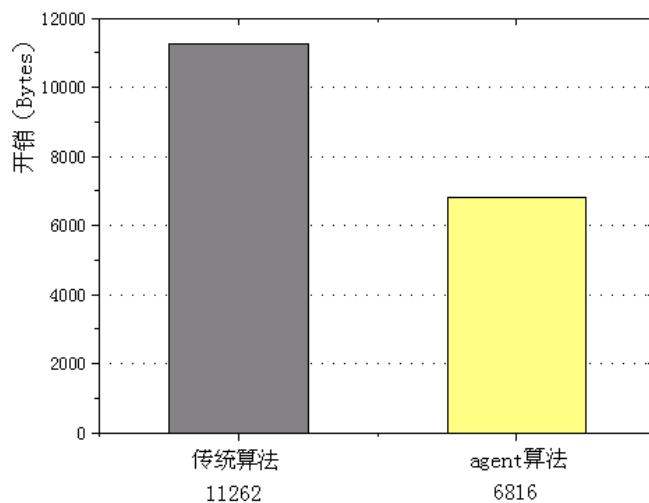


图 5.4-5 路由更新算法开销比较

从仿真结果看，传统集中式路由更新的平均开销大约是本文算法的 2 倍。由于异轨链路在整个轨道周期内并不是时刻连通的，因此，受其影响的路由相对于同轨链路发生故障的情况要少，图 5.4-6 和图 5.4-7 分别是一条异轨链路和一条同轨链路发生故障时，基于移动 Agent 的多层扩散路由更新算法的开销在每个拓扑区间的分配情况。

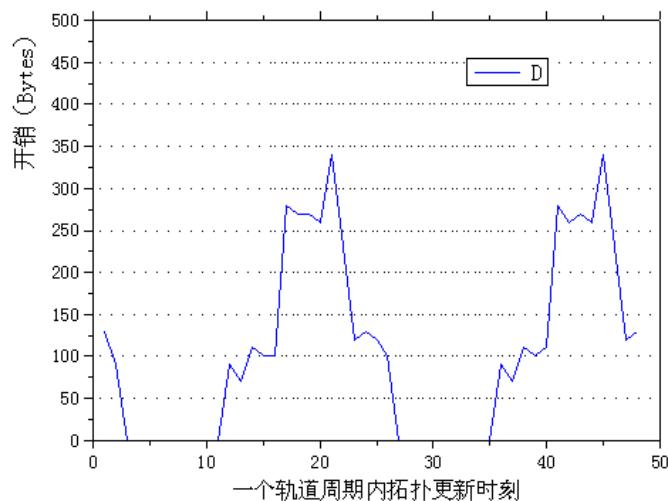


图 5.4-6 异轨链路故障时一个轨道周期内的路由更新开销

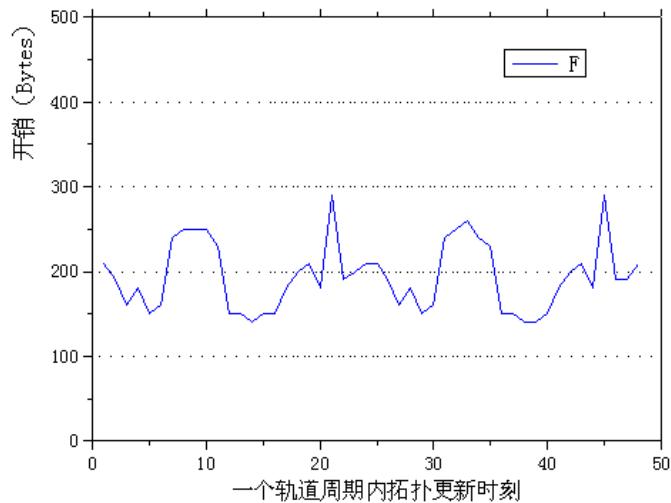


图 5.4-7 同轨链路故障时一个轨道周期内的路由更新开销

根据本文 2.3 节和 5.3 节中对星座网络拓扑的分析可知，任一条异轨链路在一个轨道周期内建立的时间段为 2 个，因此图 5.4-6 中两个时间段的开销为 0 实际上是由于在此期间该链路在空间上本身不连通、该时刻的节点路由表与此链路无关的结果。而相对而言，同轨链路由于是长久稳定连通链路，其受到故障的影响较大，因此更新的开销也较大。

(3) 发生链路故障后时延性能

为了证明本文算法的有效性，仿真与不进行路由更新的权值路由算法、进行全局更新的权值路由算法相比较。不进行路由更新即维持原有的路由表不变；进行全局更新即根据故障发生后的网络拓扑计算获得每个节点上的新路由表。仿真时，为了比较三者应对故障的性能，场景设置为：网络空闲，一条星间链路发生了故障；两个固定地理位置的用户通过星间链路发送报文，其源节点和目的节点不固定。仿真比较结果如图 5.4-8 所示，横坐标为时间，纵坐标为报文端到端时延。

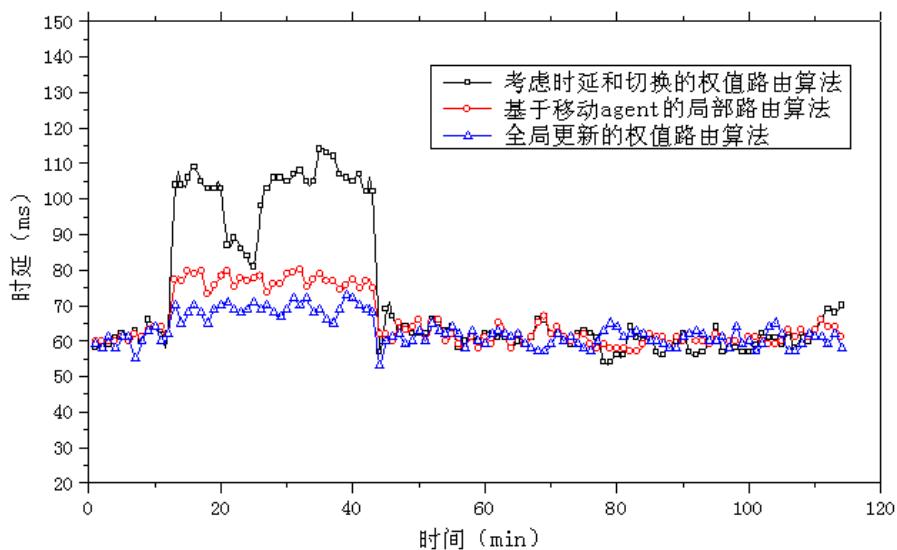


图 5.4-8 链路发生故障时的时延性能

图中时延曲线突跳区间为路由受到该故障链路影响的时间区间，这是由于仿真时用户位置固定，覆盖它们的卫星在不断变化，因此受到故障影响的时间只占整个轨道周期的一部分。由图可以看出，不进行路由更新的权值路由算法应对故障的性能表现较差，在网络空闲状态下时延陡增，这是由于业务分组在选择路由时仍旧遵循旧的路由表，造成故障节点无法应对越来越多的到达分组，从而引起较长的时延；基于移动 agent 的多层扩散路由更新算法，由于进行了路由更新，减小了故障链路对路由的影响，因此拥有较好的时延性能，同时，与全局更新的算法相比较，时延性能相差不大，这说明多层扩散的机制在限制路由更新开销的同时，能够保证最大限度地对受影响的路由表进行修正，这与本文设计此算法的初衷相吻合。

5.5 小结

本文基于快照序列路由策略，分析了切换对时延抖动的影响，提出了考虑切换和时延的权值路由算法，该算法既能保证路由选择的优先级，又兼顾了网络流量的平衡，同时，由于采用节点实时状态与权值路由表相结合的方式来选择路由，具备一定程度的自主性。综合时延和切换的影响，给出了路由代价的理论分析，并给出了算法的时延抖动、时延性能仿真结果，表明本算法在应对拥塞时的时延和时延抖动方面的性能表现良好。

针对发生故障后的路由更新问题，本文又提出了基于移动 agent 的多层扩散路由更新算法。该算法基于权值路由策略，通过移动 agent 收集和更新节点路由信息，能够在应对网络拓扑异常变化的同时，不增加星上的计算量，减小路由更新的扩散范围。对基于移动 agent 的系统进行了设计，使得算法在实现上更具有实际意义。通过仿真分析，证明算法在网络开销、更新时延有着良好性能，并通过路由更新前后时延的性能表现证明本算法的有效性。

第六章 总结与展望

本文针对低轨星座卫星通信系统开展研究工作,从星座网络自主管理的需求出发,以保证星座自主运行下的用户 QoS 为目的,研究了自主网络管理涉及的几个关键技术。

合理的体系结构是进行高效网络管理的基础;用户的位置管理是网络提供服务的前提;网络故障检测与诊断是保证网络可靠性的关键;自主路由策略及故障路由更新是提高网络性能的必要措施。星座网络有序良好的运行,必须对以上几个关键技术进行研究。本文在深入分析星座网络拓扑特征、自主运行特点的基础上,借鉴相关领域的研究成果,结合本文研究背景,在分簇管理结构、星上位置管理、自主网络故障诊断、自主路由更新等技术上进行了大量创新工作,取得了以下研究成果:

(1) 根据低轨通信星座网络拓扑特征以及承载业务的特点,确定利用成簇方式建立星座自主管理的体系结构,提出了基于约束条件的分布式分簇算法 (CDCA)。利用同轨道卫星节点之间链路的稳定长久性,提出了管理簇划分的约束条件——同轨面的一组相邻卫星节点作为一个簇结构的固定组成。基于该约束条件的分簇算法能有效降低簇维护成本、缩短管理时延、减小管理开销、均衡负载。采用图论对卫星网络进行了建模,并引入管理约束度这个重要参数,论述了簇管理者的选择策略以及管理簇的划分原则,并通过与传统分簇算法的仿真比较结果说明本算法用于自主管理体系结构时的有效性。

(2) 提出了基于簇的多 HLR 星上自主位置管理策略,摆脱了用户位置管理对地面站的依赖性,解决了星座网络自主运行时的位置管理问题。着眼于网络侧的位置管理,基于本文提出的管理簇结构,描述了 HLR 与 VLR 的位置更新流程,并提出移动 agent 与指针转发相结合的簇内更新策略,解决由于卫星运动造成的指针失效问题,同时该策略有效减少了由于卫星运动造成的用户频繁位置更新所带来的网络级更新开销。论述了整网数据库维护的问题,研究了过期位置信息的处理策略、用户位置寻呼策略等。对策略的代价进行了理论分析,并与传统策略进行了寻呼时延、位置管理开销等性能的仿真分析比较。

(3) 以系统级故障诊断理论为基础,结合低轨星座系统网络自主运行的需求,提出一种适合于星上处理的分布式自主网络故障识别算法 (DSFD),采用图论对卫星网络进行建模,提出了 M-概率的分布式测试模型以及小概率忽略的诊断规则,从而提高故障诊断完全率。针对误判,提出了测试图预修正、二次判决后处理等风险消减措施,以提高故障诊断正确率。该算法使每个节点都可独立进行自识别,而无需获得整网节点状态,能够大大降低网络故障检测开销,同时缩短诊断时延,并且降低星上

计算量，提高可靠性。文中给出了判决错误概率的理论分析，详细描述了故障诊断流程，并通过仿真对算法的有效性进行了证明。

(4) 基于静态拓扑快照路由策略，分析了切换对时延抖动的影响，提出了考虑切换和时延的权值路由算法，该算法既能保证路由选择的优先级，又兼顾了网络流量的平衡，同时具备一定程度的自主性。综合时延和切换的影响，给出了路由代价的理论分析，并给出了算法的时延抖动、时延性能仿真结果。针对发生故障后的路由更新问题，提出了基于移动 agent 的多层扩散路由更新算法。该算法基于权值路由策略，通过移动 agent 收集和更新节点路由信息，能够应对网络拓扑异常变化的同时，不增加星上的计算量，减小路由更新的扩散范围。对基于移动 agent 的系统进行了设计，使得算法在实现上更具有实际意义。最后给出了算法在网络开销、更新时延以及路由更新后时延性能上的仿真结果。

随着星座网络自主运行的需求发展，星座自主网络管理技术会受到越来越多的关注，本文涉及的几个方面的关键技术，仍然存在诸多问题待解决和进一步改进。下一步的工作为：

- (1) 针对自主管理体系结构，改进 CDCA 算法，优化算法应对故障的能力；
- (2) 针对星上自主管理模式，研究无线侧的位置管理策略，降低用户位置更新开销，减小星上数据库维护成本。
- (3) 改进 DSFD 算法，利用网络拓扑规律，提高硬故障诊断率，并将网络级的分布式 SLD 算法与卫星设备级故障诊断相结合，研究星上自主故障诊断技术。
- (4) 开展移动 agent 技术的平台开发，为移动 agent 在星上的实现奠定基础。

参考文献

- [1] 余金培, 杨根庆, 梁旭文, 现代小卫星技术与应用[M], 上海科学普及出版社, 2004.
- [2] 张更新, 张杭等, 卫星移动通信系统[M], 人民邮电出版社, 2001.
- [3] 吕海寰, 蔡剑明, 甘仲民等, 卫星通信系统[M], 人民邮电出版社, 1996.
- [4] 陈荔莹, 徐东宇, 赵振岩, 国外卫星星座自主运行技术发展综述[J], 航天控制, 2008, Vol.26, No. 2, pp92-96.
- [5] Todorova P. Network Management in ATM LEO Satellite Networks[C]//Proceedings of the 35th Hawaii International Conference on System Sciences. Big Island, USA, IEEE, 2002, pp.3907-3913
- [6] Jena philippe, Martin Flatin, Simon Znaty and Jean Pierre Hubaux. A Survey of Distributed Network and Systems Management Paradigms[J]. Journal of Network and Systems Management, Springer, vol. 7, pp. 9-26, 1999
- [7] 姜月秋, 陈冬松, 冯永新, 王光兴, 一个应用于卫星网网络管理的星簇生成算法[J], 小型微型计算机系统, 2004, Vol.25, No.5.
- [8] 郭楠, 分布式网络自管理模型及相关问题研究, 博士学位论文, 东北大学, 2005.
- [9] Charles E. Perkins. 2008. *Ad Hoc Networking* (1 ed.). Addison-Wesley Professional.
- [10] Baker, D.J., and Ephremides, A., “A distributed algorithm for organizing mobile radio telecommunication networks,” in 2nd International Conference on Distributed Computer Systems, Paris, France, IEEE, 1981, pp. 476-483.
- [11] Baker, D.J., and Ephremides, A., “The architectural organization of a mobile radio network via a distributed algorithm,” IEEE Transactions on Communications COM-29 11, 1981, pp. 1694–1701.
- [12] Ephremides, A., Wieselthier, J.E., and Baker, D.J., “A design concept for reliable mobile radio networks with frequency hopping signaling,” in Proceedings of IEEE, Vol. 75, 1, 1987, pp. 56-73.
- [13] MGerla, J Tzu-Chieh Tsai. Multicluster Mobile, Multimedia Radio Network[J]. ACM/Baltzer Journal of Wireless Networks, 1995, 1(3):225-265.
- [14] Parekh, A.K., “Selecting routers in ad-hoc wireless networks,” in Proceedings of the SBT/IEEE International Telecommunications Symposium, 1994, pp. 420-424.
- [15] Lin, C. R., and Gerla, M., “Adaptive Clustering for Mobile Wireless Networks,” IEEE Journal on Selected Areas in Communications, Vol. 15, 7, 1997, pp. 1265-1275.

- [16] S. Basagni, "Distributed clustering for ad hoc networks", in Proceedings of the 1999 International Symposium on Parallel Architectures, Algorithms, and Networks (I-SPAN'W), (Perth/Fremantle, Australia), pp. 310-315, IEEE Computer Society, June 23-25 1999.
- [17] MAINAK CHATTERJEE, SAJAL K. DAS and DAMLA TURGUT."WCA: A Weighted Clustering Algorithm for Mobile Ad Hoc Networks", 2002 Kluwer Academic Publishers. Manufactured in The Netherlands.Cluster Computing 5, 193–204, 2002
- [18] Chatterjee, M.; Sas, S.K.; Turgut, D.; "An on-demand weighted clustering algorithm (WCA) for ad hoc networks," Global Telecommunications Conference, 2000. GLOBECOM '00. IEEE , vol.3, no., pp.1697-1701 vol.3, 2000
- [19] Wonchang Choi; Miae Woo; , "A Distributed Weighted Clustering Algorithm for Mobile Ad Hoc Networks," Telecommunications, 2006. AICT-ICIW '06. International Conference on Internet and Web Applications and Services/Advanced International Conference on , pp. 73, 19-25 Feb. 2006
- [20] Bricard-Vieu.V, Nasser. N, and Mikou, N, "A Mobility Prediction-based Weighted Clustering Algorithm Using Local Cluster-heads Election for QoS in MANETs," Wireless and Mobile Computing, Networking and Communications, 2006, IEEE International Conference on. pp. 24–30
- [21] Chang Li; Yafeng Wang; Fan Huang; Dacheng Yang; , "A Novel Enhanced Weighted Clustering Algorithm for Mobile Networks," Wireless Communications, Networking and Mobile Computing, 2009. WiCom '09. 5th International Conference on, pp.1-4, Sept. 2009
- [22] Anitha, V.S.; Sebastian, M.P.; , "Scenario-Based Diameter-Bounded Algorithm for Cluster Creation and Management in Mobile Ad hoc Networks," Distributed Simulation and Real Time Applications, 2009. DS-RT '09. 13th IEEE/ACM International Symposium on, pp.97-104, 25-28 Oct. 2009.
- [23] Jahani, S.; Bagherpour, M., "A Clustering Algorithm for Mobile Ad hoc Networks based on Spatial Auto-Correlation", 2011 International Symposium on Computer Networks and Distributed Systems (CNDS),2011 , Page(s): 136 – 141
- [24] S.Muthuramalingam, R.RajaRam, Kothai Pethaperumal and V.Karthiga Devi.A Dynamic Clustering Algorithm for MANETs by modifying Weighted Clustering Algorithm with Mobility Prediction.International Journal of Computer and Electrical Engineering, Vol. 2, No. 4, August, 2010, pp:1793-8163

- [25] Brust, M.R.; Andronache, A.; Rothkugel, S.; , "WACA: A Hierarchical Weighted Clustering Algorithm Optimized for Mobile Hybrid Networks," Wireless and Mobile Communications, 2007. ICWMC '07. Third International Conference on, pp.23, 4-9 March 2007
- [26] Nizar Bouabdallah, Rami Langar, and Raouf Boutaba, Design and Analysis of Mobility-Aware Clustering Algorithms for Wireless Mesh Networks. IEEE/ACM TRANSACTIONS ON NETWORKING, 2010, VOL. 18, NO. 6, pp:1677-1690.
- [27] S. Bandyopadhyay and E. J. Coyle, "An energy efficient hierarchical clustering algorithm for wireless sensor networks," in IEEE INFOCOM, 2003, vol. 3, pp. 1713–1723.
- [28] O. Younis and S. Fahmy, "Distributed clustering in ad-hoc sensor networks: A hybrid, energy-efficient approach," in Proc. IEEE INFOCOM, 2004, pp. 629–640.
- [29] Zhang Jian-wu, Ji Ying-ying, Zhang Ji-ji, Yu Cheng-lei. A Weighted Clustering Algorithm Based Routing Protocol in Wireless Sensor Networks. 2008 ISECS International Colloquium on Computing, Communication, Control, and Management pp:599-602
- [30] 冯永新,王光兴,刘治国,姜月秋. 一个应用于移动 Ad Hoc 网络管理的簇生成算法 [J], 软件学报, 2003, Vol.14, No.
- [31] Yueqiu Jiang; Yongbing Liu; Yingyou Wen; Guangxing Wang. A clustering algorithm applied to the satellite networks management. Proceedings of the Fourth International Conference on Digital Object Identifier: 10.1109/PDCAT.2003. Page(s): 396 – 399.
- [32] Zhang Wenbo; Sun Peigen; Xu Haifeng; Dong Qian; , "Design and implement of management clustering primitives based on satellite networks," Advanced Computer Control (ICACC), 2010 2nd International Conference on, vol.2, pp.259-263, 27-29 March 2010
- [33] 闻英友,冯永新,王光兴.一种卫星综合信息网中的管理域划分策略及簇生成算法[J]. 小型微型计算机系统,2004,25(10):1742-1745.
- [34] 宋剑锋, 空间卫星网络自主管理及其协作模型研究. 博士论文(国防科大) 2007.4
- [35] 何家富等, 一种具有异轨星间链路的 Walker 星座网络拓扑与路由生成方案[J], 解放军理工大学学报 (自然科学版), Vol. 10, No.5 , Oct . 2009
- [36] Martin, E.; , "A Graphical Study of the Timer Based Method for Location Management with the Blocking Probability," Wireless Communications, Networking and Mobile Computing (WiCOM), 2011 7th International Conference on , pp.1-4, 23-25 Sept.2011

- [37] Keqin Li, Cost analysis and minimization of movement-based location management schemes in wireless communication networks: a renewal process approach, *Wireless Netw* (2011) 17:1031–1053
- [38] Xian Wang, Pingzhi Fan, Jie Li, Yi Pan. Modeling and Cost Analysis of Movement-Based Location Management for PCS Networks With HLR/VLR Architecture, General Location Area and Cell Residence Time Distributions. *IEEE Transactions on Vehicular Technology*. 2008, Volume 57, 3815-3831
- [39] Yi-hua Zhu, Victor C. M. Leung, Optimization of Distance-Based Location Management for PCS Networks, *IEEE TRANSACTIONS ON WIRELESS COMMUNICATIONS*, VOL. 7, NO. 9, SEPTEMBER 2008
- [40] Vergados, D.D. Panoutsakopoulos, A. Douligeris, C. Location Management in 3G Networks using a 2-Level Distributed Database Architecture. *IEEE 18th International Symposium on Personal, Indoor and Mobile Radio Communications. PIMRC* 2007. pp:1-5
- [41] Ian F. Akyildiz, Janise McNair, et al. Mobility Management in Next-Generation Wireless Systems. *Proceedings of the IEEE*. 1999, 8(87):1347-1384
- [42] Ian F. Akyildiz, Wenye Wang, A Dynamic Location Management Scheme for Next-Generation Multitier PCS Systems. *IEEE TRANSACTIONS ON WIRELESS COMMUNICATIONS*, VOL. 1, NO. 1, JANUARY 2002
- [43] Narendhiran K, Tafazoli R, Evans B. Evaluation of location tracking schemes for satellite UMTS[J]. *3G Mobile Communication Technologies*, 2000, 3: 381-386.
- [44] R.Jain, Y.B.Lin and S.Mohan. A caching strategy to reduce network impacts of PCS. *IEEE J.Select.Areas Commun*, Oct.1994, 12:1434-1444
- [45] Shivakumar N. and Widom J. User profile replication for faster location lookup in mobile environments. *Proc. ACM/IEEE MOBICOM'95*, 1995, 161-169
- [46] Jain.R. and Lin Y.B. Performance modeling of an auxiliary user location strategy in a PCS network. *Wireless Network*, 1995, 1:197-210
- [47] Joseph S.M.Ho and Ian F.Akyildiz. Local anchor scheme for reducing signaling costs in personal communications networks. *IEEE/ACM Transactions on Networking*, Oct.1996, 4(5):709-725
- [48] Martin, E.; Ling Liu; Weber, M.; Pesti, P.; Woodward, M.; , "Unified analytical models for Location Management costs and optimum design of location areas," *Collaborative Computing: Networking, Applications and Worksharing*, 2009. CollaborateCom 2009. 5th International Conference on
- pp.1-10, 11-14 Nov. 2009

- [49] Munadi, R.; Ismail, M.; Abdullah, M.; Misran, N.; , "Location management cost reduction using fuzzy logic in cellular radio network," Space Science and Communication (IconSpace), 2011 IEEE International Conference on , vol., no., pp.165-169, 12-13 July 2011
- [50] 刘玉利, 基于 agent 的移动用户位置管理技术研究, 硕士学位论文, 哈尔滨理工大学, 2007
- [51] Archan Misra, Member, IEEE, Abhishek Roy, Member, IEEE, and Sajal K. Das, Information-Theory Based Optimal Location Management Schemes for Integrated Multi-System Wireless Networks. IEEE/ACM TRANSACTIONS ON NETWORKING, VOL. 16, NO. 3, JUNE 2008
- [52] K.Narendhiran,R.Tafazolli, and B.G.Evans. New Methodology Defining Fixed Location Area in Mobile Satellite Communication System Vehicular Technology Conference,2001:315~317
- [53] Alexander Guntsch. Mobility Management in an Integrated GSM a Satellite PCN. Vehicular Technology Conference,1996,3(28):1830~1834
- [54] Janise McNair. Location Registration and Paging in Mobile Satellite Systems. IEEE.2000:232~237
- [55] Zhu ZHANG, Qing GUO, An IP mobility management scheme with dual location areas for IP/LEO satellite network, Zhejiang Univ-Sci C (Comput & Electron) 2012 13(5):355-364
- [56] Debabrata Sarddar, Soumya Das, Dipsikha Ganguly, A Time based Mobility Management Method for Leo Satellite Networks, International Journal of Computer Applications (0975 – 8887) Volume 42– No.2, March 2012
- [57] 刘丽华, 郭伟, 刘伟, LEO 卫星网基于时间和距离的位置更新策略, 通信技术, Vol.43, No.06,2010
- [58] Fan Hu; Lidong Zhu; , "Selective paging schemes for LEO satellite network based on dynamic location area," Communications, Circuits and Systems (ICCCAS), 2010 International Conference on , vol., no., pp.186-189, 28-30 July 2010
- [59] 李黎明, LEO 卫星通信网络中位置管理策略研究及仿真平台设计, 硕士学位论文, 电子科技大学, 2008
- [60] 张旭, 低轨卫星通信系统位置管理技术研究, 硕士学位论文, 电子科技大学, 2008
- [61] 李彬, 朱立东, 吴诗其. LEO 卫星网络多 HLR 位置管理策略研究. 西华大学学报, 2007, 26 (5) :42-45

- [62] Wang Jinglin; Cao Zhigang; , "Research on hierarchical location management scheme in LEO satellite networks," Future Computer and Communication (ICFCC), 2010 2nd International Conference on , vol.1, pp. 127-131, 21-24 May 2010
- [63] 张旭,朱立东,吴诗其.低轨卫星系统中结合时间和移动的位置更新策略.电讯技术,2008,48 (2) :57-60
- [64] 朱艺华,肖刚,史定华,高济.按概率分批寻呼的位置管理策略.通信学报, 2004, 25 (8) :82-88
- [65] F. P. Preparata, G. Metze, R. T. Chien, “On the connection assignment problem of diagnosable systems”, IEEE Trans. Electronic Computers, vol E C-16, 1967, pp 848-854.
- [66] F.Barsi,F.Grandoni and P.Maestrini.A Theory of Diagnosability of Digital Systems[J].IEEE Trans.on Computers.1976.C-25(6):583-593.
- [67] K.Y.Chwa and S.L Hakimi.Schemes for Fault-Tolerant Computing:A comparison of Modular Redundant and t-Diagnosable Systems[J], Information and Controls,1981,45(3):212-238.
- [68] M.Malek. A Comparison Connection Assignment for Diagnosis of Multiprocessor Systems[A], In Proc.of the 10th Symposium on Computer Architecture[C], La Baule,France,1980:31-36.
- [69] J.Maeng and M.Malek. A Comparison Connection Assignment for Self-Diagnosis of Multicomputer System[A]. In Proc. Of the 11th Fault Tolerant Computing Symposium[C].Portland,Maine,1981:173-175.
- [70] S Chessa,P Santi,Comparison based system-level fault diagnosis in ad-hoc networks[C]// Proc.20th IEEE Symp.on Reliable Distributed Systems,New Orleans:IEEE,2001,257-266.
- [71] S.N.Maheshwari and S.L.Hakimi. On Models for Diagnosable Systems and Probabilistic Fault Diagnosis[J]. IEEE Trans. On Computers, 1976, C-25(3): 228-236.
- [72] A.D.Friedman and L.Simoncini,System-level Fault Diagnosis[J],IEEE Trans.on Computer,1980,13:47-53.
- [73] 侯霞, 卫星网络故障检测与诊断相关技术的研究. 博士论文(中国科学院) 2005.6
- [74] J.G.Kuhl and S.M.Reddy.Distributed Fault Tolerance for Large Multiprocessor System[A].In Proc. Of the 7th Symposium on Computer Architecture[C],1980:23-30.
- [75] R.Bianchini and R.Buskens. Implementation of On-Line Distributed System-level Diagnosis Theory[J].IEEE Trans.on Computers, 1992, 41(5): 616-626.

- [76] Duarte, E.P., Jr.; Nanya, T.; , "Hierarchical adaptive distributed system-level diagnosis applied for SNMP-based network fault management," Reliable Distributed Systems, 1996. Proceedings., 15th Symposium on , pp.98-107, 23-25 Oct 1996
- [77] E. P. Duarte Jr., and T. Nanya, "A Hierarchical Adaptive Distributed System-Level diagnosis algorithm", IEEE Transactions on computers, Vol. 47, no. 1, January (1998).
- [78] M.S.Su,K.Thulasiraman, and A.Das. A Multi-level Adaptive Distributed Diagnosis Algoruthm for Fault Detection in A Network of Processors[A]. In Proc.of the 39th Annual Allerton Conf.on Communication,Control, and Computers,Allerton.IL,pp. 15-30, 2001.
- [79] Dongni Li, Cluster-Based System-Level Fault Diagnosis in Hierarchical Ad-Hoc Networks, 2007 International Conference on Computational Intelligence and security, 2007, pp1062-1066
- [80] Nishi Yadav, P.M. Khilar, Hierarchically Adaptive Distributed Fault Diagnosis in Mobile Adhoc Networks Using Clustering, 2010 5th International Conference on Industrial and Information Systems, pp 7-12, ICIIS 2010, Jul 29 - Aug 01, 2010, India
- [81] E.R.Scheinerman. Almost Sure Fault Tolerance in Random Graphs[J]. SIAM Journal on Computing, 1987, 16(6):1124-1134.
- [82] D.M.Bough,G.F.Sulllivan and G.M.Masson. Efficient Diagnosis of Multiprocessor Systems under Probabilistic Model[J]. IEEE Trans.on Computers, 1992, 41:1126-1136.
- [83] D.Fussel,S.Rangarajan. Probabilistic Diagnosis of Multiprocessor Systems with Arbitrary Connectivity[A]. In Proc.of the 19th Int.IEEE Symp.on Fault-Tolerant Computing[C], Chicago,Illinois,1989:560-565.
- [84] T.Bartha,E.Selenyi. On Classification Heuristics of Probabilistic System-level Fault Diagnostic Algorithms[A]. P.Kacauk and G.Kotsis. Distributed and Parallel Systems:From Instruction Parallelism to Cluster Computing[M], 2000
- [85] Elhadef, M. A Perceptron Neural Network for Asymmetric Comparison-Based System-Level Fault Diagnosis[C]. 2009 International Conference on Availability, Reliability and Security, 2009, Page(s): 265-272
- [86] Hui Yang, Elhadef, M. , Nayak, A. An Evolutionary Approach to System-Level Fault Diagnosis[C]. IEEE Congress on Evolutionary Computation, Chongqing, 2009, Page(s): 1406-1413
- [87] Elhadef, M. Solving the PMC-Based System-Level Fault Diagnosis Problem Using Hopfield Neural Networks[C]. 2011 IEEE International Conference on Advanced Information Networking and Applications, 2011, Page(s): 216 – 223

- [88] Elhadef, M. , Nayak, A. Comparison-Based System-Level Fault Diagnosis: A Neural Network Approach, IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, VOL. 23, NO. 6, JUNE 2012
- [89] X Hou,Z H Fan,L Li,F J Xu,C D She,G Hu.Algorithm of Fault Diagnosis for Satellite Network[C]///Proc.of the International Conference on Intelligent Mechatronics and Automation.Chengdu, China:IEEE,2004,594-598.
- [90] X Hou,Z H Fan,L Li,Z G Hong.Fault Diagnosis in Satellite Network by System-level Diagnosis[C]//Proc.of the 1st IFIP International Conference on Wireless and Optical Communications Networks. Muscat, Oman: IEEE, 2004, 151-154.
- [91] 侯霞, 范植华等, 基于系统级诊断理论的卫星网络故障识别算法. 软件学报. 2006, Vol.17,No.3, pp.388~395
- [92] 侯霞, 杨鸿波, 范植华, 一种改进的卫星网络故障诊断算法及其评估, 系统仿真学报, 2006, Vol.18, No.11, pp3172-3179
- [93] Chang H S, Kim B W, Lee C.G, et al. FSA-based Link Assignment and Routing in Low-earth Orbit Satellite Networks. IEEE Transactions on Vehicular Technology,1998,47(3):1037-1048.
- [94] Werner M, A Dynamic Routing Concept for ATM-based Satellite Personal Communication Networks. IEEE Journal on Selected Areas in Communications, Vol.15,No.8, pp. 1636-1648,October 1997.
- [95] Gounder, V.V.; Prakash, R.; Abu-Amara, H.; , "Routing in LEO-based satellite networks," Wireless Communications and Systems, 2000. 1999 Emerging Technologies Symposium, pp.22.1-22.6, 1999.
- [96] 王京林, 晏坚, 曹志刚, LEO 卫星网络快照序列路由算法优化[J], 宇航学报, Vol.30, No.5, September, 2009
- [97] Uzunalioglu, H.; , "Probabilistic routing protocol for low Earth orbit satellite networks," Communications, 1998. ICC 98. Conference Record. 1998 IEEE International Conference on , vol.1, pp.89-93 vol.1, 7-11 Jun 1998.
- [98] M. Mohorcic, A. Svilgelj, G. Kandus, and M. Werner, Performance evaluation of adaptive routing algorithm in packet-switched intersatellite link networks, Intern. J. of Satellite Commun., vol.20,pp.97-120,2002.
- [99] M. Mohorcic, M. Werner, A. Svilgelj, and G.. Kandus, Adaptive Routing for Packet-oriented Intersatellite Link Networks: Performance in Various Traffic Scenarios,IEEE Trans, Wireless Commun., vol.1,no.4,pp.808-818,October 2002.

- [100] Admela Jukan, Hoang Nam Nguyen and Harmen R.van, An approach to QoS-based for LEO Satellite Networks, 2000 International Conference on Communication Technology (ICCT 2000) IEEE . Vol .1(2000).ISBN 0-7803-6394-9:S 922-929, 21.08.2000-25.08.2000.
- [101] 宋学贵, 刘凯, 张军, 程连贞, 一种适于 LEO 卫星网络的动态源路由算法[J], 北京航空航天大学学报, Vol.32, No.12, December, 2006
- [102] Ekici E, Akyildiz I F, and Bender M D. Bender, Datagram routing algorithm for LEO satellite networks. In Proc, IEEE INFOCOM, vol.2, Mar,2000,pp.500-508.
- [103] Ekici E, Akyildiz I F, and Bender M D. Bender, A Distributed Routing Algorithm for Datagram Traffic in LEO Satellite Networks. IEEE/ACM Transactions on Networking, 2001,9(2):137-147.
- [104] Papapetrou, E.; Pavlidou, F.-N.; , "Distributed Load-Aware Routing in LEO Satellite Networks," Global Telecommunications Conference, 2008. IEEE GLOBECOM 2008. IEEE ,pp.1-5, Nov2008.
- [105] 王亮, 张乃通。LEO 网络中卫星切换的动态概率路由优化策略, 通信学报, 2002 年 9 月第 23 卷 第 9 期。
- [106] 白建军, 卢锡城, 彭伟, LEO 卫星网络中一种简洁的星上分布式路由协议, 软件学报, 2005 vol.16, No12.
- [107] 曾媛, 低轨卫星通信网络的星上交换和路由技术研究, 中国科学院研究生院博士学位论文, 2008
- [108] Kucukates, R.; Ersoy, C.; , "High performance routing in a LEO satellite network," Computers and Communication, 2003. (ISCC 2003). Proceedings. Eighth IEEE International Symposium on, pp. 1403- 1408, vol.2, June 2003.
- [109] Jun Sun and Eytan Modiano. Routing Strategies for Maximizing Throughput in LEO Satellite Networks, IEEE Journal On Selected Areas In Communications, 22(2):273-286,Feb 2004.
- [110] G. Kandus S. Kos M. Pustisek J. Bester A. Svilgelj, M. Mohorcic. Routing in ISL networks Considering Empirical IP Traffic. IEEE Journal On Selected Areas In Communications, 22(2):261-272, Feb 2004.
- [111] Qijie Huang; Boon Sain Yeo; Peng-Yong Kong; , "A routing algorithm to provide end-to-end delay guarantee in low Earth orbit satellite networks," Vehicular Technology Conference, 2004. VTC 2004-Spring. IEEE 59th , vol.5, pp. 2911- 2915, May 2004

- [112] Qijie Huang; Boon Sain Yeo; Peng-Yong Kong; , "An enhanced QoS routing algorithm for provision of end-to-end delay guarantee in low earth orbit satellite networks," Wireless Communications and Networking Conference, 2005 IEEE , vol.3, pp. 1485- 1490 Vol. 3, 13-17 March 2005.
- [113] Dongni Li; Xin Wang; Ya Meng; , "A Destruction-Resistant Routing Algorithm in Low Earth Orbit Satellite Networks," Wireless Communications, Networking and Mobile Computing, 2007. WiCom 2007. International Conference on , pp.1841-1844, 21-25 Sept. 2007.
- [114] Yiltas, D.; Zaim, A.H.; , "A Dynamic Routing Algorithm in LEO Satellite Systems Estimating Call Blocking Probabilities," Recent Advances in Space Technologies, 2007. RAST '07. 3rd International Conference on, pp.541-545, 14-16 June 2007
- [115] 王汝传, 徐小龙, 黄海平编著, 智能 AGENT 及其在信息网络中的应用[M]. 2006
- [116] Di Caro, G., Dorigo, M., AntNet: A mobile agents approach to adaptive routing [R]. Bruxelles: Universite Libre de Bruxelles, Technical Report IRIDIA, 1997.12-36.
- [117] Dorigo M. & Di Caro G., "AntNet: Distributed Stigmergetic Control for Communications Networks[J]," Journal of Artificial Intelligence Research, Number 9, pp. 317-365, 1999.
- [118] Baran, B.; Sosa, R.; , "A new approach for AntNet routing[C]," Computer Communications and Networks, 2000. Proceedings. Ninth International Conference on, pp.303-308, 2000
- [119] Strobbe, M.; Verstraete, V.; Van Breusegem, E.; Coppens, J.; Pickavet, M.; Demeester, P.; , "Implementation and evaluation of AntNet, a distributed shortest-path algorithm[C]," Telecommunications, 2005. advanced industrial conference on telecommunications/service assurance with partial and intermittent resources conference/e-learning on telecommunications workshop. aict/sapir/elete 2005. proceedings, pp. 320- 325, 17-20 July 2005
- [120] Arnous, R.A.; Arafat, H.A.; Salem, M.M.; , "Improving the load balancing within the data network via modified AntNet algorithm[C]," Information and Communications Technology, 2007. ICICT 2007. ITI 5th International Conference on , pp.189-195, Dec. 2007
- [121] Aman, S.S.; Akbarzadeh-T, M.-R.; Naghibzadeh, M.; , "A novel approach to distributed routing by super-AntNet[C]," Evolutionary Computation, 2008. CEC 2008.

- (IEEE World Congress on Computational Intelligence). IEEE Congress on , pp.2151-2157, June 2008
- [122] Hossain, M.S.; Elghobary, D.; El Saddik, A.; , "Ant based routing algorithms for resource constrained networks," Instrumentation and Measurement Technology Conference (I2MTC), 2010 IEEE , vol., no., pp.192-197, 3-6 May 2010
- [123] Wang Ping; Gu Xue-mai; Liu Gong-liang; , "Multi-QoS Routing for LEO Satellite Networks[C]," Advanced Communication Technology, The 9th International Conference on , vol.1, pp.728-731, 12-14 Feb.2007.
- [124] 许辉, 吴诗其, LEO 卫星网络中基于蚂蚁算法的分布式 QoS 路由[J], 计算机学报, Vol. 30 No. 3 Mar. 2007
- [125] 饶 元, 王汝传, 郑 彦. 一种基于移动 Agent 卫星网动态路由算法[J], 解放军理工大学学报, 2010, 11(3)

作者攻读博士学位期间发表的论文

- [1] 江玉洁, 姜兴龙, 梁旭文. 面向移动星座网络的自主管理分簇算法[J], 系统工程理论与实践, 2011, 11, (Ei 收录)
- [2] 江玉洁, 姚晔, 梁旭文. 分布式卫星网络故障诊断算法[J], 小型微型计算机系统, 已录
- [3] 江玉洁, 姚晔, 梁旭文. LEO 卫星网络中一种自适应权值路由算法[J], 计算机应用与软件, 已录
- [4] 江玉洁, 基于移动 agent 的 LEO 卫星网络多层扩散路由更新算法, 计算机工程, 已投
- [5] 姚晔, 江玉洁, 梁旭文. 卫星 CICQ 交换系统调度算法研究, 计算机工程, 2012
- [6] 宋海伟, 江玉洁, 龚文斌. 一种基于 agent 的卫星导航任务自主处理架构的研究[C], 第二届中国卫星导航学术年会, 2011

作者简历

姓 名 江玉洁
性 别 女
籍 贯 河南省
出生日期 1980 年 2 月

主要简历：

2006.09 – 2012.08 中国科学院上海微系统与信息技术研究所
通信与信息系统 工学博士

2004.04 – 至今 上海微小卫星工程中心 工作

2001.09 – 2004.04 西安电子科技大学
测试计量技术及仪器 工学硕士

1997.09 – 2001.07 西安电子科技大学
检测技术及仪器仪表 本科

参加研究课题情况：

2004.04–2005.07, CAPS 精码接收机的研制：负责接收机 DSP 软件开发工作，完成接口控制、辅助信号捕获同步过程、解帧、译码、导航解算等功能；参加大系统联试、测试任务；

2005.08–2006.12, CX02 星地演示验证系统、CX02 地面设备研制：负责收发信机 DSP 软件开发工作，其中地面设备参加了星地联试试验；

2007.01–2008.12, 星座预研项目：参加星座交换路由方案设计与论证、配合系统大总体进行技术协调、参与完成星座交换机的原理样机研制与验收；

2009.05–2011.12, 二代导航项目：参加有效载荷方案设计工作；负责导航任务处理器的研制和测试工作，并参加有效载荷电性联试；参与技改项目的方案设计、报告编写；参与有效载荷 IDS 协调；参与星间链路及自主导航方案论证；

2011.10–2012.08, 某低轨星座预研项目：负责星载交换机的方案设计、接口协调、测试方案论证，参与星间组网方案论证等工作；

2011.09–2012.08, 某科学实验卫星数传分系统：负责数传分系统的方案设计、技术协调、接口协调等工作，配合总体完成有关大系统接口协调等；

中国科学院上海微系统与信息技术研究所

学 位 论 文 独 创 性 声 明

本人声明所呈交的学位论文是我个人在导师指导下进行的研究工作及取得的研究成果。尽我所知，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，也不包含为获得中国科学院上海微系统与信息技术研究所或其它教育机构的学位或证书而使用过的材料。与我一起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了谢意。

研究生签名 _____ 日 期 _____

中国科学院上海微系统与信息技术研究所

学 位 论 文 使 用 授 权 声 明

本人完全了解中国科学院上海微系统与信息技术研究所有关保留、使用学位论文的规定，即研究所有权保留送交论文的复印件，允许论文被查阅和借阅；可以公布论文的全部或部分内容，可以采用影印、缩印或其他复制手段保存论文。本人电子文档的内容和纸质论文的内容相一致。保密的论文在解密后遵守此规定。论文的公布（包括刊登）授权中国科学院上海微系统与信息技术研究所人才教育处。

研究生签名 _____ 导师签名 _____ 日 期 _____