电 子 科 技 大 学

UNIVERSITY OF ELECTRONIC SCIENCE AND TECHNOLOGY OF CHINA

专业学位硕士学位论文

MASTER THESIS FOR PROFESSIONAL DEGREE

论文题目 天地一体化网络下基于机器学习的路由技术研究

专业学位类别 工 程 硕 士

学 号 201822220402

作 者 姓 名 罗泽耀

指 导 教 师 郭 伟 教 授

分类号 _____ 密级 _____

UDC [注1] _____

学 位 论 文

天地一体化网络下基于机器学习的路由技术研究

（题名和副题名）

罗泽耀

（作者姓名）

指导教师 郭 伟 教 授

电子科技大学 成 都

（姓名、职称、单位名称）

申请学位级别 硕士 专业学位类别 工 程 硕 士

工程领域名称 电子与通信工程

提交论文日期 2021.04 论文答辩日期 2021.05

学位授予单位和日期 电子科技大学 2021 年 06 月

答辩委员会主席 雷霞

评阅人 冯钢、章小宁

注 1：注明《国际十进分类法 UDC》的类

Research of Routing Technology based on Machine Learning in Integrated Satellite-Terrestrial Information Network

A Master Thesis Submitted to

University of Electronic Science and Technology of China

Discipline: **Master of Engineering**

Author: **Luo Zeyao**

Supervisor: **Prof. Guo Wei**

School: **National Key Laboratory of Science and Technology on Communication**

摘　要

天地一体化信息网络（Integrated Satellite-Terrestrial information Network，ISTN）是一种融合了天基网络、空基网络和地面网络的多层次异构一体化网络，随着人类对太空的不断探索和互联网应用需求的不断扩张，天地一体化将成为通信网络发展的趋势，对于天地一体化网络的研究一定程度上代表了我国信息科技发展的前沿水平。天地一体化网络由于具有覆盖面积广泛、节点高速动态变化、通信链路质量低、同时运行多种网络协议等特点，导致在网络架构设计、异构网络管理、资源配置、路由等方面都具有很大的难度。得益于近年来软件自定义网络（Software Defined NetWork，SDN）技术的快速发展，SDN 基于控制面和数据面分离的特点有效地提高了网络部署效率，同时人工智能和机器学习在解决工程问题上的广泛应用，也为天地一体化网络架构和路由算法研究提供了新的思路。

本文的主要工作分为以下三点：第一，由于天地一体化网络异构性和网络规模的不断扩大，现有的网络架构已无法实现灵活高效的网络部署，本文设计了一种基于 SDN 的天地一体化网络架构 SDN-ISTN。该架构遵循传统 SDN 中控制与数据分离的思想，设计了一种用于实现全网集中式管理和控制的控制器，并详细阐述了控制器的实现细节和部署方案。第二，基于 OPNET 和 STK 仿真软件搭建了基于 SDN 架构的天地一体化网络仿真平台，并提出了一种适用于该网络场景的拓扑控制机制。借鉴南向接口协议 OpenFlow 的原理，设计了一种适用于本场景中网络节点通信的数据交互协议，完成了控制器、卫星节点和地面终端节点的建模和功能实现，最后通过仿真验证了所提出的 SDN-ISTN 架构和拓扑控制机制的合理性。第三，提出了一种基于机器学习算法分支 Q-Learning 的一体化自适应路由算法 ISTN-QR(Integrated Satellite-Terrestrial Information Network based Q-Learning Routing Alogorithm)。得益于 SDN 架构下控制器集中式控制网络的特点，该算法改进了传统 Q-Learning 算法中分布式计算实现路由收敛时间长的缺点，同时结合 ISTN 天基网络中卫星轨道的运行特点，改进路由计算过程中 Q 值的更新方式和选路策略，提升了路由算法对于 ISTN 网络拓扑变化的适应性。仿真结果表明，ISTN-QR 路由算法相比于 FEQ-Routing 和传统 SDN 路由算法 Dijkstra，在高低流量负载情况下都具有更高的包递交率，同时在端到端传输时延、时延抖动方面也都有明显的性能提升，因此 ISTN-QR 算法整体上具有更好的可靠性和稳定性。

关键词：天地一体化信息网络，SDN，机器学习，Q-Learning，路由算法

ABSTRACT

ISTN (Integrated Satellite-Terrestrial information Network) is the integration of heterogeneous networks such as space, air and ground network. With the continuous exploration of space and expansion of Internet applications demand, the integration of satellite and ground will become the trend of the development of the communication network. A certain extent on the study of integration of satellite and ground network represents the forefront of the information technology development level. Due to the wide coverage area, quick speed and the variation of the characteristic parameter of networks, links of low quality, and multiple network protocols running at the same time, the integrated network of satellite and ground has great difficulties in network architecture design, heterogeneous network management, resource allocation, routing and so on. Thanks to the quick development of software-defined network technology in recent years, SDN effectively improves the efficiency of network deployment based on the separation of control plane and data plane. Meanwhile, with the wide application of artificial intelligence and machine learning in solving engineering problems, it also provides a new way of thinking for the research of network architecture and routing technology.

The main work of this dissertation consists of the following three points: Firstly, Due to the heterogeneity and the continuous expansion of the scale of the network, the existing network architecture can no longer achieve flexible and efficient network deployment，so this dissertation designed an architecture based on SDN in the ISTN. Based on the idea of separating control plane from data plane in traditional SDN network, we designed a controller for centralized management and control of the global network. And we elaborate the implementation details and deployment scheme of the controller in the following. Secondly, using OPNET and STK simulation software, we set up the integration of satellite and ground network simulation platform based on SDN architecture. And a topology control mechanism suitable for this network scenario is proposed. From south to the principle of interface protocols OpenFlow, we design a kind of communication protocol which is applicable to this scenario, then realize the ground terminal controller and satellite node modeling, finally we verified the rationality of the proposed integrated network architecture through the simulation. Thirdly, this dissertation proposes an integrated adaptive routing algorithm ISTN-QR (Integrated Satellite

Terrestrial Information Network based Q-Learning Routing Alogorithm) which based on Q-Learning, a branch of machine learning algorithm. Thanks to SDN controller in the centralized control of network architecture, the performance of the proposed algorithm has better performance than that of Q-Learning algorithm on the routing convergence time. At the same time, combining ISTN space-based network in the operation of the satellite orbit characteristics, we improve the way of routing computation of Q value in the process of updating and routing strategy. The simulation results show that ISTN-QR routing algorithm has higher packet delivery rate under high and low traffic loads, and also has significant performance improvement in end-to-end transmission delay and delay jitter, compared with FEQ-Routing and traditional SDN routing algorithm Dijkstra. Therefore, ISTN-QR algorithm has better reliability and stability on the whole.

Keywords: Integrated Satellite-Terrestrial Information Network (ISTN), Software Defined NetWork (SDN), Machine Learning (ML), Q-Learning, Routing Algorithm

目 录

图目录

表目录

缩略词表

英文缩写	英文全称	中文名称
AI	Artificial Intelligence	人工智能
AODV	Ad hoc On-Demand Distance Vector	按需距离向量路由
DT	Decision Tree	决策树
DTN	Delay Tolerant Networks	容忍延迟网络
FEQ-Routing	Full Echo Q-Routing	全回响 Q 路由
GEO	Geostationary Earth Orbit	地球同步轨道
ISL	Inter-Satellite Link	星间链路
ISTN	Integrated Satellite Terrestrial information Network	天地一体化信息网络
ISTN-QR	Integrated Satellite-Terrestrial Network based adaptive Q-Routing	天地一体化网络自适应 Q 路由
LEO	Low Earth Orbit	低地球轨道
MEO	Medium Earth Orbit	中地球轨道
ML	Machine Learning	机器学习
NM	Network Management	网络管理
OLSR	Optimized Link State Routing	优化链路状态路由
RL	Reinforcement Learning	强化学习
SDN	Software Defined Network	软件定义网络
SDN-ISTN	SDN based Integrated Satellite Terrestrial Information Network	基于 SDN 的天地一体化网络架构
SL	Supervised Learning	监督学习
SO	Satellite Orbit	卫星轨道
STL	Satellite-Terrestrial Link	星地链路
SVM	Support Vector Machine	支持向量机
UL	Unsupervised Learning	无监督学习

第一章 绪 论

1.1 研究背景与意义

无线移动通信在过去的几十年内经历了飞速发展，如今已经发展到了 5G 通信，地面通信网络的用户规模也呈指数型增长趋势，另外受限于地面网络的覆盖面积和网络通信容量，对于卫星通信网络的研究也在近 10 年开始受到重视，可以预测未来的全球通信网络将会是一个融合了传统地面移动通信网络、多层次轨道卫星网络、多飞行器高/低空通信网络等新型融合异构一体化网络。国外一些以美国为代表的发达国家从 2000 年左右就开始部署天地一体化网络的建设工作，一些国外研究机构和协会也相继提出了很多关于融合异构网络的协议体系，在国家政府和商业公司的大力支持下陆续诞生了转型通信架构（TCA）、一体化卫星通信计划（ISI）、国际海事卫星（INMARSAT）、铱星系统（Iridium）等商用天地一体化信息网络项目[1]。我国早在 2000 年的《中国的航天》白皮书就对建设新型天基信息网络提出了明确要求，2006 年沈荣骏院士提出了关于建设天地一体化航天互联网的构想[2]，一些研究机构和高校开始陆续在这领域进行深入研究。如今我国已将空天地一体化信息网络的研究和建设纳入了“十三五”规划中，张乃通院士[3]则进一步详细阐述了我国天地一体化信息网络的研究方向，并对天地一体化信息网络的一些概念、基本架构、协议体系和相关组网技术进行了综述。闵士权[4]指出了我国关于天地一体化网络未来的发展“三步走”步骤，实现天基、空基和地基网络之间从互补到协同最后到融合的研究步骤。

天地一体化网络是一个异构网络，具备网络覆盖面积广泛、节点高速动态变化、通信链路质量低、同时运行多种网络协议等特点，所以设计一个灵活组网、性能可靠，并且综合管理整体网络资源的天地一体化网络是一个十分复杂的研究问题。SDN 技术作为一种新型网络体系架构，通过将传统网络设备中控制层面和数据层面分离，简化了数据交换设备的设计复杂度，也有利于实现对全网络状态信息和拓扑情况的集中式管理，通过 SDN 控制面实现全网信息管理，同时灵活控制网络节点，可以构建一个信息传输可靠、灵活且可配置的天地一体化网络架构，由于控制层面具有全局视图和一定的控制能力，因此借助 SDN 可以改善天基网络中节点之间的协作能力和异构一体化网络系统的兼容性。

天地一体化网络中天基网络是非常重要的一个网络段，通常采用卫星星座的形式构建天基网络。随着卫星互联网规模的不断扩大，近年来低地球轨道星座已经成为全球各大企业和学术界的关注重点，随之陆续诞生了以 Iridium、OneWeb 和

StarLink 为代表的低轨卫星星座系统，其中铱星系统和 OneWeb 系统的具体卫星轨道参数见下表 1-1 所示。

表 1-1 Iridium 与 OneWeb 系统卫星参数

轨道高度	785km	1200 km
倾斜角	86.4°	87.9°
运行周期	100.13 min	110 min
最小仰角	8.2°	10.15°
轨道数	6	18
轨道卫星数	11	36
卫星总数	66	648

第一代 Iridium 铱星卫星系统于 20 世纪 90 年代由摩托罗拉公司发起建设，系统包含 66 颗在轨卫星和 6 颗备用卫星，它们均匀分布在 6 个低轨圆轨道面上，提供全球覆盖的实时语音服务[5]。第二代 Next Iridium 系统则进一步降低了卫星的平均造价和建设成本，并且提高了卫星的载荷能力，相比前一代系统，Iridium Next 具有更大的传输带宽，而且具有专属频率不和地面通信运营商网络相互干扰，因此可以提供更灵活的业务速率，提供多种业务需求。OneWeb 公司于 2012 年宣布部署 OneWeb 卫星互联网系统，经过多年的前期准备于 2019 年正式开展卫星发射任务，截止到 2019 年末，OneWeb 已经发射了 79 颗卫星，意味着 OneWeb 项目已经正式进入了部署阶段[6]。另外由 SpaceX 公司主导的 StarLink 系统已经完成了功能测试和初期系统部署任务。截止到 2020 年底，SpaceX 公司已经完成了 16 次卫星发射任务[7]，共发射了接近 1000 颗低轨卫星，系统已形成了初步的规模，配合地面关口站即可提供全球范围内的互联网业务。根据 StarLink 的建设成本和运营能力分析，可以预测到未来 StarLink 的主要商业模式：首先承担一部分的地面互联网的骨干流量，降低通信时延，提供更灵活的组网方式；然后是为蜂窝网不能覆盖到的地球偏远地区（包括极区、海洋、航空等）提供网络服务；然后是进一步演化成常规空间网络的基础设施，逐渐取代或者辅助太空领域中的中继卫星提供传输链路。

目前设计天地一体化网络方向的路由算法研究主要集中在某一网络段之上，很少综合其它网络段的网络和链路特性，协调设计一体化网络中的多网络协同路由算法。实际上在典型的天地一体化网络场景下，由于卫星对地面的覆盖面积较广和低轨卫星移动速度较快等原因，路由传输过程中对于潜在星地链路的选择问题和星间链路频繁断开与连接问题都使得天地一体化下路由算法设计不同于传统地面端到端路由机制那样简单，低轨卫星网络具有较高的动态变化特性，使得在融合

低轨卫星网络和地面网络的天地一体化中的路由研究问题变得更加复杂。随着人工智能的不断普及，机器学习称为近年来被广泛议论的话题，在路由算法方面，越来越多以强化学习为代表的机器学习算法被提出，并被广泛地应用在卫星通信网络、Ad Hoc、车联网等实际网络场景中，因此利用机器学习算法设计天地一体化网络中的新型路由算法具有广泛的应用前景。

1.2 国内外研究现状

1.2.1 天地一体化网络协议体系

1.2.1.1 CCSDS 协议体系

随着卫星网络和地面网络规模的不断增大，传统 IP 网络协议架构已不再适用于这种新型天地一体化网络场景，因此许多研究机构开始研究如何整合整个天地一体化网络下通信协议体系。空间数据系统咨询委员会（CCSDS）针对空间卫星网络环境，参考了现有地面网络协议和编码调制技术，并进行了标准化，提出了一种分层式的空间通信协议标准（SCPS），CCSDS 协议体系如下图 1-1 所示[8]，与 OSI 标准 7 层协议模型不同，CCSDS 协议体系当中有的协议是可跨层次描述的，比如 CCSDS 文件传输协议（CFDP）和 Proximity-1 协议。

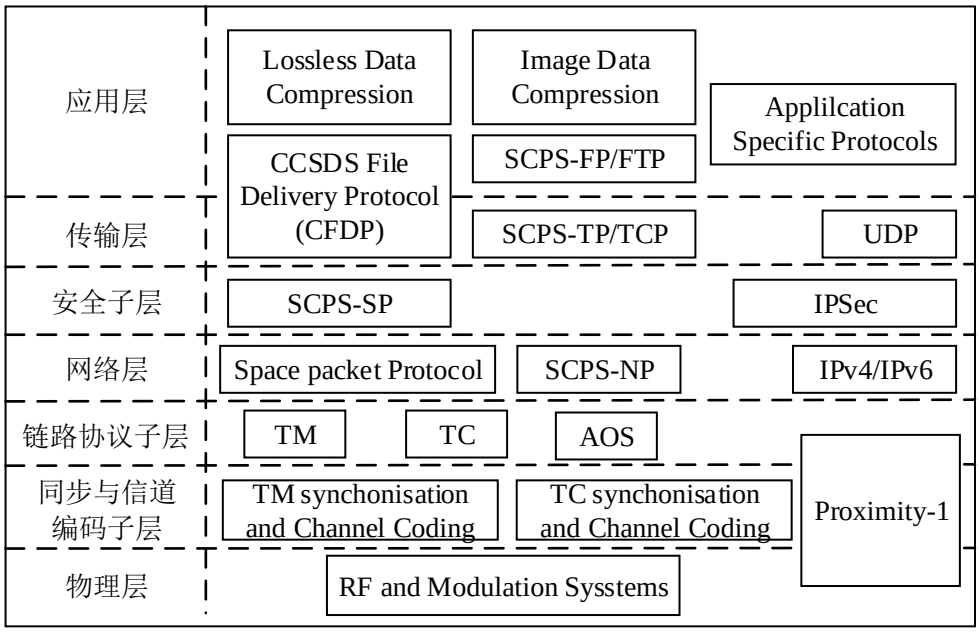

图 1-1 CCSDS 协议体系

CCSDS 数据链路子层和同步信道编码子层对应于 OSI 中的数据链路层，数据链路子层主要由 Telecommand(TC)、Telemetry(TM)、Advanced Orbiting System(AOS)

和 Proximity-1 协议组成，这些协议可以为星地通信的上行链路和下行链路数据传输提供多种链路服务。TM 依赖于 TM 同步和信道编码子层提供服务，包括帧定界、帧同步、错误控制编码等功能，TC 主要用于底层传输控制，通过将较大的数据包拆分成小块实现分段功能，TC 也具备一种称为 COP 的自动重传功能。另外 TC 依赖于底层 TC 同步和信道编码子层提供的同步和错误控制编码功能。

CCSDS 的网络层通常包含了三种协议标准，分别是 Space Packet Protocol（SPP）、SCPS-NP 和 IPv4/IPv6。他们主要负责网络寻址和路由功能，网络层协议由于不具备重传功能，所以不保障数据成功传递，需要高层传输协议保障服务质量。另外 CCSDS 规范的网络层协议中原不包含传统互联网中广泛使用的 IP 协议，但是为了将空间网络和传统地面网络融合，可以使用封装技术将 SCPS-NP 或 SPP 数据包封装成 IPv4/IPv6 能够承载的 IP 数据包。CCSDS 协议体系虽然提出了以 SCPS-SP、SCPS-TP 等为代表的传输控制协议，但是并没有要求强制使用这些协议，因为一些应用层协议（比如 CFDP）作为跨层次协议可以直接工作在网络层之上，不需要传输控制层协议提供服务也能保障可靠的端到端数据传输。

目前地面网络中使用最为广泛的网络层技术是 IP 协议，集合未来应用需求不断发展的趋势，CCSDS 基于 IP 技术提出了 IP over CCSDS 的协议标准，其主要的目的就是为未来天地一体化网络通信提供一种协议设计方案。目前的研究表明，基于 IP 协议的空间卫星网络已经可以和地面互联网之间进行互通，其中基于 IP over CCSDS 的协议体系则发挥了巨大的作用。IP over CCSDS 协议体系建议空间和地面网络使用一致的网络协议，其中空间网络运行 CCSDS 体系建议的链路层协议，包括 AOS、TC、TM 等链路子层协议，网络层在 CCSDS 链路层之上承载 IP 数据包已实现天基网络和地面网络的融合。

尽管基于 CCSDS 体系的通信体系已经受到广泛关注和研究，但是目前 CCSDS 并没针对路由问题提出可行解决方案，由于最初 CCSDS 提出协议解决方案的出发点是面向星地链路点到点的通信，而不直接适用于天地一体化网络中的异构节点之间的路由问题。即使后来提出了基于 IP 协议技术的 IP over CCSDS 协议体系可以解决天地一体化网络的灵活配置，但是仍然没有从根本上解决 CCSDS 协议体系在空间网络通信上的缺陷[9]。

1.2.1.2 DTN 协议体系

容忍延迟网络（Delay Tolerant Networks，DTN）的概念最早由美国推进实验室与 2003 年提出[10]，随后 Internet 研究任务组基于星际网络研究小组（Inter-planetary Networking Research Group，IPNRG）的研究内容开始研究 DTN 的通信协议，随后

经过 DTN 研究小组（DTNRG）的不断改进，于 2007 年提出了如下图 1-2 所示的 DTN 协议体系，用于解决星际网络中卫星通信时链路缺乏连接性的问题。

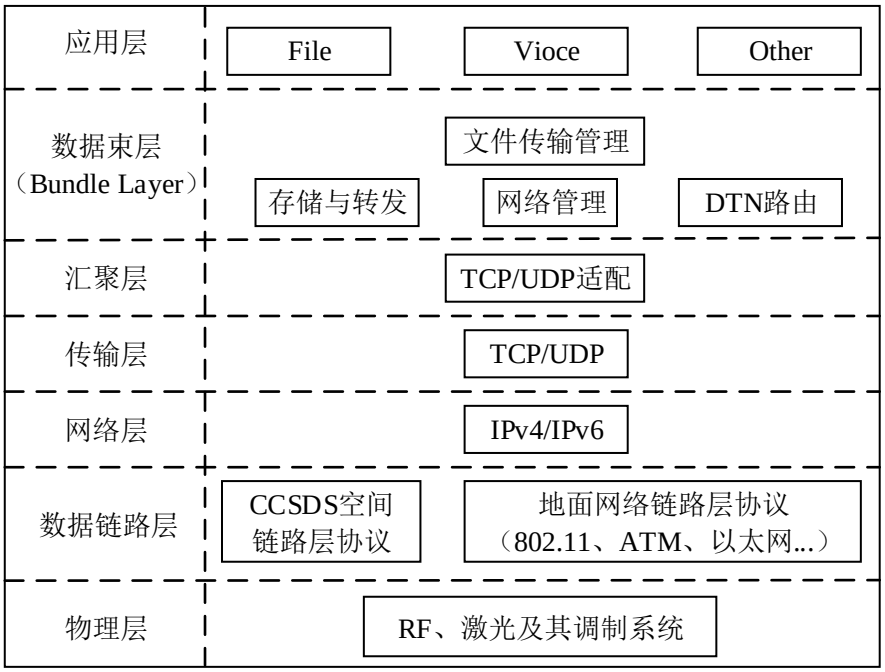

图 1-2 DTN 协议体系

在 DTN 体系当中，容忍延迟网络被描述成一种受限网络，因为网络中设备之间的通信有可能受限于通信距离、节点移动、不可预测的环境因素、有限的存储/计算能力等因素，而无法基于 Internet 网络体系结构提出解决方案。比如常见的地面移动网络、太空卫星网络、军事无线自组织网络、无线传感器网络等，都可以在具体的应用环境中被描述为 DTN，相比于传统 TCP/IP 网络，DTN 网络有以下特点：高时延，通常卫星网络通信的时延单位以秒为单位进行统计，而地面 Internet 中传输时延以毫秒作为统计单位；部署在太空、海上等极端环境中的 DTN 网络节点其计算资源和存储资源有限，在一定程度上限制了网络的性能；不对称的数据传输速率[10]，这意味着系统输入输出的流量存在巨大差异，通常在太空探测应用中普遍存在这种情况；间歇性的网络链接，例如卫星网络中星间链路的周期性断开和连接，无线传感器网络中传感器节点的随机性断开等情况；最后是 DTN 网络一般具备较低信噪比和较高误码率的传输特性，这是 DTN 网络环境带来的必然结果，一般在 DTN 太空通信网络中，由于环境导致的信号误码率可以达到 10%[10]。这些特点无疑使得传统 TCP/IP 协议技术无法直接适用于情况复杂且节点异构的网络环境中。在天地一体化网络中，天基网络具有网络延迟大、链路频繁中断、卫星轨迹周期可预判等特点，而相对于天基网络，地面网络中节点运动速度低而且网络拓扑

相对固定，另外空中网络作为地面网络的拓展，也基本具有地面网络的一些特点，因此使用 DTN 协议架构部署 ISTN 理论上是一种可行的方案。

在 DTN 协议体系中，需要重点关注数据束（Bundle）层协议，Bundle 层协议的主要作用是基于“存储+转发”的思想实现数据的可靠传输，Bundle 层针对异构网络中不同子网的不同协议架构，使用了统一的数据保管机制，从而屏蔽了底层网络不同带来的弊端，实现整体异构网络的全覆盖。Bundle 层运行在传输层之上，通常对于 DTN 节点的存储能力有非常高的要求。尤其是在深空通信网络环境中，卫星和飞船等节点的存储资源和网络链路资源非常有限，基于 DTN 协议的 bundle 层数据束有可能在某个传输连路上存储超时，最终导致节点内存资源耗尽，进而还会导致数据丢失，引发数据源节点进行超时重传，进而降低网络整体性能，因此可以看出，Bundle 层的设计对于 DTN 网络的性能表现至关重要。

基于 DTN 协议体系设计天地一体化网络被验证是一种可行的方案，基于 DTN 的路由算法研究也一直是一个热门话题。DTN 体系通过 Bundle 层协议屏蔽底层的不同网络体系，从而融合多种异构网络，这意味着在存在着多种异构子网的天地一体化网络场景中，可能同时运行着多种路由协议。比如适用于天基网络的快照路由、DRA 路由等，适用于空中集群飞行器网络的 EPI 路由，另外地面网络中 Ad Hoc 网络中常用的 AODV、OLSR 等路由算法，这些路由算法通常都运行在天地一体化网络对应子网内部，而 DTN 网关作为连接各个子网的重要节点，通常也运行着基于 DTN 协议的特殊路由协议，比如 CGR 路由[11]。对于 DTN 路由性能指标的评价不仅仅基于当前路径或下一跳节点的选择上，也可能取决于业务数据类型，另外 DTN 节点转发数据时，也可以根据网络状态和业务需求灵活选择其它备份路由协议，这些无疑给整体网络路由部署造成了较大的难度。因此在天地一体化网络中，基于 DTN 协议体系的多路由协议会造成许多实际部署上的问题，通常分层次的路由方式不是最优的解决方案。

1.2.1.3 SDN 架构体系

随着全球范围内地面和空间网络规模的不断扩大，卫星网络系统的规模也在不断扩张，异构网络融合和网络规模上升同时会给网络管理增加难度，天地一体化网络的组网难度也在陡然上升。

软件定义网络的概念伴随着互联网技术的不断进步而诞生，它将网络架构扁平化处理，实现了数据转发和网络控制分离。广义的 SDN 代表的是一种解决网络问题的思想，高效的 SDN 部署方案能够快速对网络路由、资源配置做出指示决策，提高网络整体工作效率。典型的 SDN 架构如下图 1-3 所示，一般将物理网络设备

分离成为控制平面、数据平面和应用层面，控制面的运行实体通常从网络交换机中抽离出来以 SDN 网络设备独立运行，以实现对整体网络信息的统一管理和对数据交换设备行为的集中式控制。而数据平面则可以运行在网络中各种转发设备、移动终端设备之上，由于工作在数据平面，大部分网络设备已经不需要进行独立的网络计算任务，因此 SDN 架构对于网路部署的成本控制也有较大贡献。

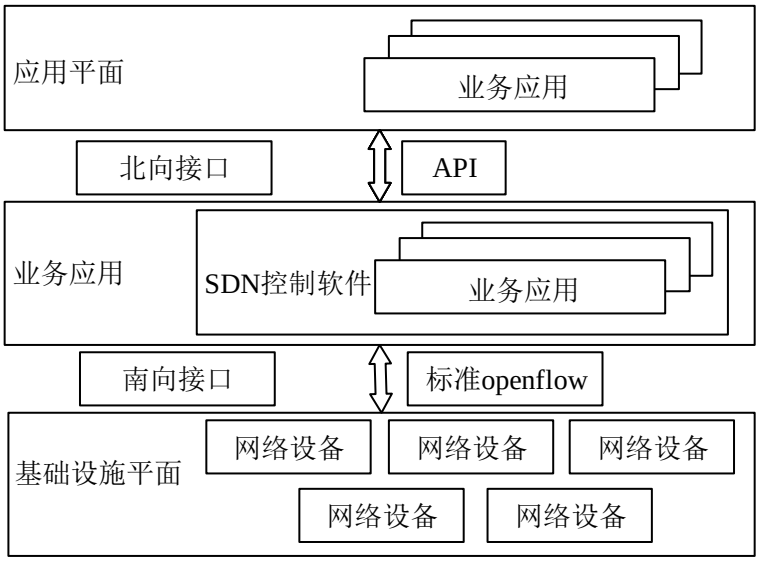

图 1-3 典型的 SDN 架构[12]

在传统分布式网络架构中，网络设备一般同时负责网络管理和数据转发，一旦网络中出现突发情况，对应的网络事件需要逐步通知才能扩散到整个网络，这种方式往往比较低效。在 SDN 中全网管理和监测功能全部由 SDN 控制器负责，网络设备通常只需要执行控制器下发的对应指令，这种分层的组网思想降低了全网部署和网络路由计算的难度。SDN 在实际地面网络当中已经被广泛使用，依靠先进且智能便捷的网络管理方式，给地面网络和空间网络设计和优化做出了巨大贡献。

在文献[12]中作者研究了常见SDN控制器的两种部署方式:层次型和扁平型，并结合空天地一体化网络架构和场景特点，选择了分层次部署 SDN 控制器实现一体化网络管理和控制的方案，然后通过分析 SDN 中多控制器的负载均衡原理和性能表现，提出了详细的层次型多控制器实现负载均衡的设计流程。最后作者在 Ubuntu+Cbench 软件测试平台验证了基于 SDN 的空天地一体化网络下层次型控制器部署方案的优势，实验结果表明相比于平面式多控制器部署，层次型的多控制器架构设计能有效减少数据面和控制面之间数据流的交互数量。

在文献[13]中作者基于 LEO 星座系统 Iridium 的网络和链路模型，研究了基于 SDN 架构的天基网络组网方式，然后分析了 LEO 卫星网络中典型的两种路由方案

虚拟节点和虚拟拓扑的优缺点，然后选择基于虚拟拓扑原理的路由方案，通过在SDN 控制面上部署 Dijkstra+DFS 路由算法，并且利用 SDN 控制器能实时获取全网拓扑信息的优势，基于时间片原理实时计算对应最短路径，然后通过 SDN 南向接口下发路由信息到转发节点。最后作者通过实验仿真验证了基于 SDN 的天基网络架构在通信时延和数据递交率性能上要由于传统的天基组网架构。

在文献[14]中，作者提出了一种结合 SDN 和 NFV 的天地一体化网络架构SERvICE，该架构主要由管理面、控制面和数据面三个层面构成，基于数据和控制分离的思想，卫星只需要简单执行来自控制面的策略。另外基于网络功能虚拟化（NFV）技术，该架构可以不受网络设备位置的限制，灵活地在卫星网络和地面网络中配置相关网络服务功能。SERvICE 基于 SDN 集中式控制网络的思想，通过将地面控制器、GEO 卫星、网络信息收集模块和转发策略模块部署在控制面，并将MEO/LEO 卫星、地面网关和相应交换机设备部署在数据面，该架构可以高效地实现天地一体化网络集中式管理，并部署对应卫星进行数据处理和转发。最后作者通过相关实验，表明文章提出的网络架构 SERvICE 在功能设计上是可行的，实现了天地一体化网络下基于该架构的灵活组网和细粒度 QoS 保证。

在文献[15]中，作者针对目前天地一体化网络中网络拓扑高速变化、卫星移动性、高延迟和网络拓展性等问题，提出了一种基于 SDN 的融合了空间网络和地面网络的复合架构 SD-STIN，该架构基于软件定义网络和边缘计算技术，实现了灵活组网和高效的全网管理。另外文章对 SD-STIN 架构下移动管理、资源分配和路由方案等领域问题进行了研究，并介绍了一些潜在的解决方案。

在文献[16]中作者基于信息中心网络和 SDN 提出了一种新型天地一体化信息融合网络架构，根据低中高轨道卫星的特征实现了对应的功能，并解释了星上路由器的功能划分和流表设计原理。文中针对该一体化网络架构下卫星网络中星间路由提出了一种基于空间网络划分的动态自适应分布式路由算法，然后详细阐述了星地协同实现内容传输方案的基本思想和工作流程。最后通过实验仿真验证了基于 SDN 架构的信息中心网络和天基网络融合的天地一体化网络架构在完成星地协同数据传输方面带来的性能增益。

在文献[17]中作者研究了基于 SDN 的空间网络、移动通信网络和互联网之间融合的一体化网络架构和实现，文章分析了实际的组网需求和现状并提出了一种基于“天网地网”形式的 SDN 一体化网络组网架构，并详细阐述了组网原理和运行流程。文章特别地针对了异构网络中控制器的部署实现问题提出对各网络控制器的改造解决方案，并通过实验仿真了基于 OPNET 平台搭建的 SDN 一体化网络仿真架构的可行性。

综合以上关于 SDN 的天地一体化网络架构的研究文献，可以看出目前大多数研究侧重于对 ISTN 的架构设计和网络的统一控制管理的实现，具体主要体现在 SDN 架构下控制器对全网的集中式管理和控制方式，对于原有网络协议的一体化融合改进方式，以及在一些具体网络场景下的实际应用研究等方面，但是很少有文献针对 SDN 架构下 ISTN 网络中协同路由算法进行深入研究。另外对于天地一体化网络，作为核心网络段的卫星网络通常会承担大量的流量转发任务，考虑到卫星设备有限的硬件处理能力和卫星网络部署成本，基于 SDN 架构的 ISTN 组网方式需要着重考虑天基网络段中星座选型、SDN 控制器部署和功能划分、卫星交换机实现以及全网资源分配等问题。

1.2.2 传统卫星网络路由算法

路由方案无法适用于所有的网络场景，在天地一体化网络下的路由算法应该针对天地一体化网络自身的特点，结合网络架构和业务特点进行优化。要提出一种适用于天地一体化网络场景的路由算法，可以从以下几个方面进行思考和研究。

首先要考虑卫星网络拓扑频繁变化的特性，天基网络是天地一体化网络中的骨干网络段，管理和维护卫星拓扑对于路由算法的性能至关重要，应该结合卫星星座的周期性和卫星轨迹的可预测性来设计对应路由算法，另外还需要综合考虑卫星节点上有限的计算存储能力。其次，为了防止卫星拓扑变化产生的链路切换导致呼叫丢失，路由算法也可以结合拓扑变化周期规律和流量预测算法有效避免呼叫丢失的情况，提高通信质量。第三，路由算法一般可以基于链路传播时延或者路由跳数等准则计算最短路由路径，从而降低端到端传输时延和提高链路资源利用率。第四，卫星星座中卫星轨道和卫星的分布一般是均匀对称的，即轨道间隔和同轨道卫星间隔基本固定，设计路由算法时可以结合该特点进一步优化选路策略，提高星间链路的使用效率。另外还需要避免部分星间链路出现拥塞，同时部分星间链路长时间空闲的情况，即应该考虑实现负载均衡的自适应路由算法。此外，为了更好地实现天基网络和地面网络的融合，也可以尝试将一些现有的地面网络路由算法应用到天地一体化网络场景中，例如地面 IP 网络中常用的 OSPF、BGP 等路由算法，MANETs 网络中常用的 AODV、OLSR 等路由算法[18]，并探讨分析一些现有路由算法在天地一体化网络中的适用性。

传统的天基网络路由算法根据其设计思想可以大致分为三类：基于虚拟节点、基于虚拟拓扑和动态自适应的路由算法。

虚拟节点指将星座卫星所覆盖的地球区域进行统一划分，可以将一颗卫星对应的覆盖区域定义为一个虚拟节点，它拥有一个独立的逻辑地址，然后使覆盖该区

域的卫星拥有两个地址，一个唯一的物理地址和一个可动态适应的逻辑地址，该逻辑地址与地面逻辑地址相对应，这样就实现了卫星网络拓扑的动态变化特性，从而可以在该“静态拓扑”上进行对应路由算法计算。虚拟节点路由算法中一般数据包需要包含源地址和目的地址的地理位置信息，转发节点可以根据源地址和目的地址可以获得对应的逻辑地址，从而进行路由表更新或查询，因此一般基于虚拟节点的路由方案可以采取分布式计算的部署形式。

经典的以虚拟节点为基础设计的路由算法即 DRA[19]。理论上基于虚拟节点思想为基础的路由算法是可行的，但实际也存在一些缺陷。在近极轨星座网络当中，例如 Iridium 铱星系统的轨道倾斜角为 87.6°，所以其反向缝两侧轨道之间的间距要比其他轨道之间的间距大很多，导致难以建立稳定的异轨星间链路，而虚拟节点的要求星座网络节点整体呈均匀分布且对称，才能建立星座与地面区域覆盖的逻辑地址一一对应。因此基于虚拟节点的路由算法理论上只能适用于极轨星座网络，而且考虑到如果有部分卫星节点出现故障或者星间链路断开，但路由算法也无法第一时间获取到相应网络信息，由此也可能造成路由信息滞后或者失效。

虚拟拓扑指的是将卫星星座系统的一个运行周期 T 按照等时间间隔△T 划分成若干个时间片，然后在每个时间片上建立一个星座网络虚拟拓扑，然后可以在每个虚拟拓扑上计算对应的最优路由路径。一般基于虚拟拓扑方案的路由计算都是在地面上提前计算好，然后将路由表配置在卫星节点上，卫星节点根据时间间隔，选择对应拓扑下的路由表进行查表并完成数据转发。

经典的以虚拟拓扑为基础设计的路由算法有 M-DSPA[20]、快照序列[21]、FSA[22]等算法。这些基于星座运行周期等间隔划分时间片的路由算法基本上都是根据对应虚拟拓扑来计算相应最优路径的，一般采用的算法是 Dijkstra 最短路径算法，采用链路传输时延作为路径权重评价标准，从而计算出源到目的节点之间的最短路径，生成对应路由表。这种计算方法简单有效，但是没有考虑实际网络 QoS 要求，所有需要进一步结合 QoS、网络负载、链路利用率等指标改进路由算法形成一种多目标优化路由方案，如 DT-DVTR 算法[23]，这些算法则是基于前文虚拟拓扑路由方案，并结合了具体网络场景，针对特定优化目标提出的路由方案。但是虚拟拓扑方案通过分时间片计算最优路径的这一基础方案，导致以其为基础的路由算法存在致命的缺陷，这要求大量的网络节点需要存储每一个时间片对应的虚拟拓扑结构下的路由表，假设网络节点规模为N，则每张路由表的空间复杂度为 $O(N)$，而假设星座运行周期 T 被等时间间隔ΔT拆分成 K 个时间片，则每个卫星需要预先存储的路由表其空间复杂度为 $O(K{\cdot}N)$，显然这对卫星节点的存储计算能力有非常高的要求。另外分析虚拟拓扑方案的抗毁性能，如果网络运行期间部分节点或者链路出现

故障，网络拓扑将发生变化，但是由于虚拟拓扑算法是提前基于星座运行周期更新路由表的，不能动态地根据网络故障情况修改路由表，因此虚拟拓扑路由算法也不具备抗毁性，在实际算法设计中，应该考虑基于虚拟拓扑思想，将星座拓扑按时间片划分，但是实际的路由计算可以交给卫星节点动态根据当前网络情况动态计算并更新路由表，这样可以使其具备一定的抗毁性，但是代价是提高了数据传输时延，整体来说这种方案的性能提升不大，而且不能解决虚拟拓扑路由方案的本质缺陷。

虚拟节点和虚拟拓扑思想本质上都是在屏蔽卫星网络的动态拓扑变化给路由算法带来的麻烦，但是动态自适应的路由算法则从另外的角度来解决路由问题。动态自适应路由方案一般通过卫星节点之间周期性交换网络信息，进而获取到全网拓扑信息计算路由表。这种方案可以有效获取网络信息，实时响应节点失效或链路拥塞等网络故障，提高了整体网络的自主运行能力[24]。

典型的卫星动态自适应路由算法有 DARTING[25]、CEMR[26]、LAOR[27]等，这些路由算法大部分都参考了地面网络经典路由算法中基于链路状态、距离向量等方法来进行网络路径发现。例如 LAOR 算法参考了地面 Ad Hoc 网络中 AODV 协议的算法思想，LAOR 是一种基于位置信息辅助的按需路由算法，算法通过将 Iridium 星座上的拓扑按运行周期划分用于确定星间链路生存时间，卫星节点可以根据包含路径传输代价的信令报文获取链路信息，因此可以获取到传输时延、链路拥塞状态等情况。节点通过这种方法进行拓扑发现和路由发现，然后计算对应路径并更新路由表项。通常动态自适应路由可以有效发现并响应网络故障，对网络的稳定运行提供了保障，但是这一类路由算法要求卫星节点周期性或者根据拓扑变化进行路由发现，而路由发现期间的数据包则一直排队等待路由收敛过程结束，并且要求卫星节点也具备和地面终端一样的计算能力才能完成如此复杂的逻辑计算，因此自适应路由对于传输时延性能有一定程度上的损失，并且对卫星节点的计算能力也提出了较高的要求。目前大多数的卫星网络动态自适应路由算法都基于地面网络路由算法而来，因此没有充分考虑卫星网络的结构特点和链路特性，由于算法的复杂度较高，实际在星上处理资源严格受限的环境中难以实现。

综合以上研究情况可以得出，目前关于天地一体化网络中路由算法的研究主要分为两种类型，一种是基于地面路由算法，将卫星网络作为地面网络的数据中继点来设计路由；另一种则是从卫星网络的角度出发，只关注卫星网络中路由算法研究，而没有考虑所提出路由算法对于融合地面网络时的适用性。大部分现有卫星网络路由算法都只是基于天基网络特点进行设计，而传统地面路由算法由于网络协议体系、链路环境、拓扑情况、物理设备性能限制等因素也无法直接适用于天基网络，导致目前在新型的天地一体化网络场景中还有没有形成一种被广泛客观认同

且行之有效的路由方案。

1.2.3 机器学习与路由算法

随着无线网络通信技术的不断发展，越来越多的研究人员开始研究结合机器学习（Machine Learning，ML）算法来解决网络通信领域中的一些技术问题，比如传输信道、频谱管理、资源分配、移动接入管理和网络路由等[28]。本文重点研究了天地一体化网络中路由算法设计与实现，所以主要关注基于机器学习的路由算法研究领域，本节将简单阐述机器学习的基本思想与算法分类，以及在网络路由领域的一些研究现状。

近年来人工智能（Artificial Intelligence，AI）一直是学术界和社会应用领域讨论的热门话题，其中机器学习作为人工智能领域的一个分支，也是研究的热门学科。机器学习是一门融合了数理统计、凸优化、计算机、电子硬件等多门学科的交叉性优势学科，利用机器学习算法，我们可以利用计算机强大的计算和存储能力可以从已有数据和认知信息中自动学习新的知识，在一些特定的研究领域，机器学习算法甚至可以无需明确编程，完全基于自身算法模型不断学习从数据和信息中学习，进而改进自身模型。机器学习算法来源于科学家对计算机思考能力的不断探索，众所周知人类的大脑是非常先进的“计算机”，我们可以依据以往积累的经验和知识对当前环境做出适当反应，于是科学家们开始探索是否可以将计算机变得和人类大脑一样不断学习并做出反应。人工智能学科便在这个探索的课程中诞生了，其中机器学习便是实现人工智能的一种重要形式，当一个系统在不断变化的环境中学习，并不断适应这些新的环境时，这个系统便不但积累学习经验，最终形成一个智能系统。

通过输入一些数据到系统中，机器学习算法会依照对应的数据分析模型挖掘出特定的信息，然后依据这些信息对机器模型进行修正，这个学习的过程称为训练。机器学习按照系统学习的方式主要可以分为监督学习、无监督学习和强化学习。无论是哪种学习模式，机器学习算法对数据（即训练集）的依赖度都非常高，当输入的数据中包含的有效信息越多，最后训练出的系统其性能也就越好。不同学习算法针对数据的统计方法也有所不同，所以可以检测和识别出不同的工作模式，这也增强了算法模型对新数据及其包含信息的识别能力。

1.2.3.1 监督学习

在应用于解决实际工程问题时，监督学习（Supervised Learning，SL）是最为普遍的一类学习算法，其主要的用途是进行分类和预测。监督学习的核心方法就是从已知的输入数据和信息当中学习经验，然后输入已知的现有模型，可以建立一种

输入输出数据的映射关系并不断校正算法模型，因此监督学习需要一些外在辅助系统，可以将输入数据划分为训练集和测试集，前者的作用是训练该模型，一般输入训练数据会有对应的输出集合，系统从输入和输出集合之间的映射关系中学习某种模式，然后将这种模式应用到测试数据集中进行测试，检验该模式对于当前模型的适用性。目前广泛被使用的监督学习算法有决策树算法、贝叶斯算法和支持向量机算法。

决策树（Decision Tree，DT）算法通常用于数据分类，通常包含一个根节点、若干内部节点和若干叶节点[29]，每一个叶节点表示一种决策结果，非叶节点则表示对于某种属性的测试，从根节点逐步测试到叶子节点，其中非叶子节点会根据特定的测试函数 $f(x)$进行测试判决，决策树算法学习过程就是为了训练出一种可以根据特定属性离散集合将输入数据对象进行分类的决策树模型。

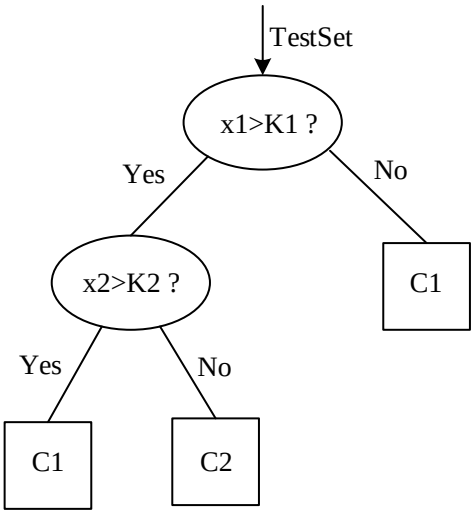

图 1-4 决策树示意图

一个简单的决策树如上图所示，假设数据集 TestSet 中的数据对象包含 x1 和 x2 两个测试属性，叶节点中的{C1，C2，C3}表示决策结果。当数据对象从根节点开始进行属性测试，首先根据 x1 属性判断是否大于 K1，不满足此测试条件的数据对象被划分到了 C1 类，满足该测试条件的对象继续进行第二个测试条件，当属性 x2 不满足大于 K2 的条件时，则划分到 C2 类，否则划分到 C3 类集合当中。从决策树算法工作流程中可以看出，决定了决策树算法性能的关键步骤就是对数据对象的最优属性划分，属性划分越准确，基于该数据集进行学习后得到的分类结果就越准确。

支持向量机（Support Vector Machine，SVM）一般可以处理分类和线性回归问题，SVM 在样本空间中定义了一个超平面，从而将不同的样本对象划分到对应的类别中，而样本数据在输入训练之前需要根据其属性特征转换成一个特征向量。

SVM 的核心是一个核函数，它表示样本对象根据其属性与 SVM 划分的特征空间之间的映射关系。针对样本对象的划分形式可以分为线性和非线性，线性划分对象即可以直接映射在 SVM 超平面上，对于不可线性划分的样本空间，通常需要将特征空间依据样本属性数量提升到一个更高维的特征空间，所以首先需要根据训练样本，提取适合的属性集合，然后找到合适的映射关系，从而推导出和函数的具体形式。通常将基于训练样本的核函数展开获得的 SVM 最优解展开式称为支持向量展式（support vector expansion）[29],通常核函数的定义隐式地表征了 SVM 对应的特征空间，所以核函数的定义是否准确，决定了 SVM 的分类性能。

贝叶斯学习是一种依赖概率统计学科进行决策分类的学习算法，通常用于样本数据的多分类任务。它依据一个样本的先验概率分布，获得一个后验分布，然后随着学习样本数据的增加，可以将之前的后验分布作为先验分布。可以看出，贝叶斯学习的先验选择很关键，因为先验分布直接影响了后验分布的结果，所以当先验分布选择不正确时，贝叶斯分类器的预测将会出错。朴素贝叶斯学习是一种简单实现、性能良好的学习方法，它基于较少的训练数据就可以基于概率表模型进行训练，并对新的数据样本进行预测，理论上朴素贝叶斯要求输入样本的特征属性相互独立，即条件概率独立，但是实际应用中即使样本集不满足此条件，贝叶斯分类也可以通过其他的一些方法保证性能。比如引入拉普拉斯修正方法，假设样本属性和类别之间均匀分布，即引入额外的先验分布。

1.2.3.2 无监督学习

非监督学习（Unsupervised Learning，UL）由于已知的输入与输出之间的映射关系，系统不能根据已有知识进行自我学习，所以非监督学习只能挖掘输入数据中的规律。一般来说，无监督学习没有明确的学习目标，由于输入数据没有已知的标签，也不能根据训练集的数据结果获得量化后的学习效果，但是无监督学习可以通过提取数据的若干特征，从而发现样本对象集合的性质。一般无监督学习的常见用途是聚类任务，聚类试图将样本集中的样本划分为若干不相关联的子集，形成一系列的样本簇，最终达到样本分类的效果。最常见的无监督学习算法就是 K-均值，首先 K-均值会确定用于划分数据样本空间的样本簇的个数 K，然后在样本空间中确定 K 个重心，对数据集 D=$\{x_1, x_2, x_3, \ldots, x_k\}$，考虑聚类结果的簇划分 C=$\{C_1, C_2, C_3, \ldots, C_k\}$，一般对于簇的划分可以通过如下所示的一些评价指标来衡量聚类结果的性能。

DBI 指数：

$$\mathrm{DBI}=\frac{1}{k}\sum_{i=1}^{k}\max_{j\neq i}\left(\frac{\mathrm{avg}\left(C_i\right)+\mathrm{avg}\left(C_j\right)}{d_{\mathrm{cen}}\left(C_i,C_j\right)}\right) \tag{1-1}$$

簇内样本距离平方差：

$$\mathrm{E}=\sum_{i=1}^{k}\sum_{x\in C_i}\left\|x-\mu_i\right\|_2^2 \tag{1-2}$$

$$\mu_i=\frac{1}{\left|C_i\right|}\sum_{x\in C_i}x \tag{1-3}$$

一般来说，DBI 指数越小，表示对数据样本对应的簇划分越合理，聚类的性能越好。在 K-均值算法中，还可以通过簇划分的最小化平方误差 E 来评价算法模型的性能，E 值越小，表示每个样本与簇内部样本均值之间的距离越小，即簇内样本的特征相似度越高[29]。

1.2.3.3 强化学习

强化学习（Reinforcement Learning，RL）作为机器学习的一个分支，与前两种学习方式有较大区别，它基于一种自主学习框架，系统可以通过这个学习框架和外界环境进行交互，通过执行特定的动作，系统转移到某种状态，然后从环境中获取由于此动作进行状态转移的反馈值，进行修正自身框架。一般来说，与监督学习和非监督学习相比，强化学习更适合解决网络路由问题，因为强化学习不需要根据现有训练数据集进行模型学习和测试，强化学习算法的学习过程与网络中数据包的不断路由决策过程非常相似，因此诞生了许多以强化学习为理论基础的路由算法。

文献[30]中作者提出了融合机器学习的分布式路由算法 ELMDR，算法基于神经网络极限学习机（Extreme Learning Machine，ELM）模型对 LEO 网络场景进行流量预测，通过对即将到来的流量负载情况进行预测，再基于产生的移动智能体收集到网络信息进行路由决策，最后相邻的卫星节点再根据路由决策动态调整自身的流量负载情况。

文献[31]首次研究了将基于强化学习分支的 Q-Learning 算法模型用于解决网络路由问题，并提出了 Q-Routing 路由算法。Q-Routing 通过综合评价多种网络参数质量来更新对应的 Q 值表项，以实现智能体与网络环境之间的信息交互，进而学习新的知识并训练算法模型，路由算法首先经历前期的探索阶段，此阶段实现智能体对网络节点和链路情况的探索，最终实现路由收敛，然后进入路由开发阶段，智能体利用当前的学习经验对数据包的转发行为做出决策。Q-Routing 算法模型的训练过程是分布式实现的，即多智能体之间同时运行路由算法，智能体仅仅基于自

身收集的局部信息进行学习，因此对于 Q-Routing 路由算法，收敛时间是一个重要的性能指标。作者在一个简单的不规则网络拓扑中验证了 Q-Routing 算法的正确性，并对算法的收敛性与网络流量负载之间的关系进行了分析。

随后有不少的学者基于 Q-Routing 算法的一些性能缺陷进行了改进，主要包括 FEQ-Routing[32]、PQ-Routing[33]和 DRQ-Routing[34]等。这些算法主要是针对算法动态网络拓扑、动态负载变化和网络故障等情况的适应性以及算法收敛时长等方面进行相应的改进。文献[32]提出了智能体 Full Echo 全回响寻路策略，使得算法在动态网络负载环境中快速收敛，提高了整体路由的时延性能。文献[33]则通过保存之前学习到的部分网络经验进行流量预测，然后动态调整路由算法的选路策略，以实现算法在多种网络流量负载情况中具有稳定的性能表现。文献[34]中提出了基于双向探索过程的 DRQ-Routing 算法增加了路由的反向探索方式，智能体不仅仅可以通过传统前向探索过程中获取的前向邻居节点信息更新本地 Q 值，反向探索是通过目的节点主动向源节点发送探索信息，使得经过的中间节点都可以利用反向邻居节点信息更新 Q 值，即一次路由过程中所需的学习信息可以拆分成前向探索和反向探索获取，同时进行两种探索模式，从而有效降低路由算法的学习收敛时间。以上几种基于 Q-Routing 改进的路由算法虽然有一定程度上的性能提升，但是仿真结果仅仅是在简单拓扑网络中完成的验证，并且具体算法仿真是采用分布式实现的，如果将 Q-Routing 直接应用到基于 SDN 的天地一体化网络场景中，以上一些路由算法的改进思想将不再适用于集中式管理的 SDN 网络架构中，因此需要结合具体天地一体化网络场景特点，对基于 Q-Learning 的路由算法进行进一步的改进。

综合以上对监督学习、无监督学习和强化学习三种类型机器学习算法的基础原理和特点分析，可以看出强化学习作为一种通过与外界环境不断交互，然后不断学习并训练算法模型的算法，是最适合应用于解决网络路由问题的算法分支。通过对强化学习算法进行建模，可以有效地将网络中的各种对象实体转换成马尔科夫决策过程模型，进而实现路由问题求解，具体的算法设计与实现则涉及到对网络中环境模型与智能体对象的划分、状态集合与动作集合的设计以及奖励函数定义等问题。如上文强化学习基础模型分析可知，引入强化学习算法用于解决实际应用问题时，一般有两种解决方案，一种是基于现有模型的，一种是基于无模型的。而在实际解决网络路由问题时，不同网络架构、网络规模和应用业务类型会导致路由面临不同的问题，很难从对应环境中收集到足够有效的先验知识，所以基于模型的强化学习很难应用在天地一体化网络场景下，目前大多数的研究也都采用的是基于无模型的强化学习理论。

1.3 本文研究内容与主要工作

本文首先分析了天地一体化网络相关技术的国内外研究现状，针对几种天地一体化网络协议体系进行了阐述，并重点分析了基于 SDN 架构的天地一体化网络的技术路线，并对目前采用 SDN 建设一体化网异构网络的研究进展进行了综述，然后提出了一种典型的基于 SDN 的天地一体化网络架构，并详细阐述了相关涉及细节。接着利用 OPNET 和 STK 仿真平台搭建了上述基于 SDN 的天地一体化网络仿真平台，实现了网络基本的通信功能。最后参考了机器学习和强化学习算法的分支 Q-Learning 的基本算法思想，提出了一种适用于上述网络场景的协同一体化路由算法 ISTN-QR，并通过 OPNET 仿真平台对路由算法的性能表现进行了仿真和分析。本文的主要工作具体有以下几点：

（1）提出了一种基于 SDN 架构的天地一体化网络场景，并基于该架构完成了控制器的功能设计和实现；

（2）基于 OPNET 和 STK 仿真软件搭建了基于 SDN 架构的天地一体化网络平台，设计了一种适用于该场景的网络交互协议，实现了基本的网络通信功能；

（3）结合强化学习算法分支 Q-Learning 的基本思想，对上述天地一体化网络进行建模，提出了一种适用于上述场景的一体化协同路由算法，通过实验仿真验证了该路由算法的性能提升。

1.4 本文组织结构

全文共分为五章，具体的结构安排如下：

第一章为绪论，主要叙述了当前天地一体化网络相关技术的研究现状。研究了当前典型天地一体化网络协议架构的优缺点，并分析了几种典型星座卫星系统的特点。研究了目前天基网络中常见路由算法的特点，并基于其算法思想对相关路由技术进行了分类阐述。最后研究了目前机器学习和路由算法相结合的相关技术，并进行了综述。

第二章主要基于传统 SDN 架构的技术特点，提出了一种基于 SDN 的天地一体化网络架构，并对该架构中控制器的设计细节和部署方案进行了详细阐述。然后基于一种典型 SDN-ISTN 网络场景，研究和分析了相关的卫星切换流程。

第三章是对第二章架构设计的仿真验证，通过 OPNET 和 STK 仿真平台搭建了第二章提出的天地一体化网络仿真场景，详细阐述了相关的设计细节，实现了基本的网络功能并仿真验证了上述架构设计的正确性。

第四章首先分析了强化学习和 Q-Learning 的基本算法模型，并基于 Q-Learning 算法思想提出了一种适用于天地一体化网络场景的新型一体化协同路由算法

ISTN-QR，并基于上述仿真场景对该路由算法进行了仿真验证，结果显示 ISTN-QR 路由算法相比于 SDN 网络 Dijkstra 路由算法，在端到端时延、时延抖动和传输可靠性等方面有一定程度的提升。

第五章总结全文的工作，并对下一步的研究工作进行了展望。

第二章 基于 SDN 的天地一体化网络架构研究

本章首先阐述了经典 SDN 网络架构和设计原理，以及控制器和交换机内部的基本工作原理。然后结合天地一体化网络特点和 SDN 数据与控制分离的思想，设计了一种基于 SDN 的天地一体化网络架构，并详细阐述了 SDN 控制器的设计原理。接着基于该网络架构研究了一种典型天地一体化网络场景，分析了该场景中整体网络的工作流程。最后研究了该场景中服务卫星的切换策略，分析了基于 SDN 控制器的卫星切换流程。本章为后文中搭建天地一体化网络场景和设计路由算法提供了理论研究基础。

2.1 SDN 架构概述

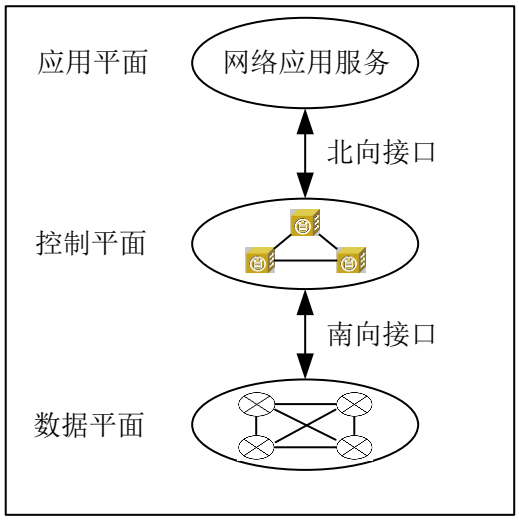

图 2-1 软件定义网络逻辑架构

传统 SDN 逻辑架构如上图 2-1 所示，其中 SDN 控制器工作在控制平面，一般起到承上启下的作用，即对上层相应的网络应用通过北向接口提供服务，对下层的物理交换机中数据转发通过南向接口提供相应指导操作。

SDN 通常具备控制与数据转发分离、转发平面和控制平面之间开放接口和逻辑集中控制等特征[35]。控制与数据分离使得 SDN 可以同时在转发和控制层面同步演进，这种去耦合的思想极大促进了 SDN 的广泛推广。同时分离成两个独立的网络层次之后，还需要高效的交互接口，即南向接口可以屏蔽控制器与交换机信息交互的实现细节，同时提供一些编程接口，方便控制器在逻辑层面实现对全网络的集中式控制，转发器则专注于底层数据转发的物理实现。

SDN 中南向接口协议是 SDN 网络设计中的一个关键部分，目前普遍使用的是

OpenFlow 协议，SDN 控制器与物理交换机之间的信息交互一般通过 Packet_in 和 Packet_out 数据包实现，控制器发送请求报文给网络节点用以收集网络信息，然后以控制报文的形式下控制指令，请求报文和控制报文的协议封装通常会根据具体应用做一些定制化改进。

2.1.1 SDN 控制器

可以看出，SDN 控制器就像传统计算机中的操作系统一样，在 SDN 整体架构中起到至关重要的作用，控制器管理和控制整体网络的运行，SDN 的整体工作效率也受到 SDN 控制的影响。一般 SDN 控制器通过将传统网络交换设备的控制平面抽离到集中化控制的 SDN 控制平面，然后通过南向接口下发控制指令。SDN 控制器的功能划分一般包括网络链路发现、拓扑管理、策略定制和流表项下发等[36]。

SDN 控制器可以通过底层交换设备互相发送 Hello 报文实现以实现邻居发现，进而通过专属链路获取全网拓扑和链路信息。交换机将自己已建立的链路情况通过南向接口上报到控制面，由控制器统一指导完成链路发现工作。SDN 网络整体拓扑信息可以由控制器统一监控和管理，主要是通过对 SDN 交换机设备的信息进行收集和计算来实现的。策略定制指控制器基于全网拓扑信息和业务分析，统一生成并下发给交换机的流表的策略形式。流表是控制平面和数据平面之间信息交互的关键形式，控制器主要利用流表实现对交换机的指令传达，所以高效的流表下发策略通常影响 SDN 的工作效率。而流表项的下发一般分为主动下发和被动下发两种情况，主动下发指交换机中收到需要匹配流表项的数据之前，SDN 控制器主动下发对应流表项；被动式下发则通常是交换机请求控制器下发相应流表项时，控制器才进行流表下发。可以看出，主动形式的流表下发通常比被动式下发更智能，能够有效减少数据处理的匹配时间，而且还可以减少数据平面和控制平面之间的信令交互频率，降低网络开销。然而主动式流表下发也需要控制器需要更灵活地收集和处理网络信息，主动监视网络中的链路变化、流表过期等资源变化情况，并及时计算和更新对应流表信息，才能保障网络高效稳定持续运行。

2.1.2 SDN 交换机

SDN 交换机是 SDN 中实际完成数据转发任务的物理设备，与传统网络架构中的网络设备（交换机或者路由器）相似，SDN 交换机在收到数据包后，也是基于数据包中的一些关键字段，获取到该数据包的网络信息，然后与自身流表项进行匹配。当获取到匹配信息时，SDN 控制机就基于匹配项执行对应数据处理行为。但是与传统网络转发设备根据自身和周边环境生成流表项的行为不同，SDN 交换机

的流表项全部由上层 SDN 控制器统一生成下发，所有的复杂网络逻辑计算都由 SDN 控制器实现，因此 SDN 交换机只需要专注实现和控制器的交互逻辑和转发逻辑，其实现复杂度相比传统网络交换设备将大幅降低。

SDN 交换机与控制器的数据交互行为一般通过 SDN 架构中南向接口实现。目前较为广泛应用的南向接口协议是 OpenFlow 协议[37-38]，它提供了一种 SDN 控制层和数据层面之间数据流传输标准，从而解决控制层如何将数据流下发给交换机和数据层面如何将上层所需信息上传给控制面的问题，实现了真正意义上的 SDN 层次分工，集中式管理的功能。

SDN 交换机的转发逻辑如上文所述，与传统网络交换设备无较大差异，在具体实现上可以参考传统交换设备的模块化设计。一般 SDN 交换机需要在交换模式、背板设计、缓冲机制和数据转发等方面进行合理设计[36]。

SDN 交换机的数据交换模式定义了数据包的转发速度和一些处理延时指标，规范的交换机设计通常会给用户提供一些选项用于对交换模式进行初始化设置。数据包在交换机内部的传输过程可以通过背板设计来描述，一般需要指定设备 IO 端口、数据存储/中继单元等。常见的 SDN 交换机内部数据传输采用共享总线机制，即数据包经过对应裁决单元指定传输端口后，会直接进入对应端口的缓冲队列当中排队，等待处理。高效合理的缓冲机制可以避免数据包因不能被及时发送而丢弃的情况。

与传统交换机相似，SDN 交换机中的数据转发策略可以通过流表的形式抽象描述传统路由器中的路由表、交换机中的转发表等机制。SDN 交换机中的流表更新依据控制器的相应指令，而交换机通常需要进行存储更新和查询操作，另外也可以根据流表对应的时间戳检测流表项是否过期，然后主动请求控制器更新相应流表项。

2.2 基于 SDN 的天地一体化网络架构研究

2.2.1 天地一体化网络架构设计

相比于地球同步轨道（Geostationary Earth Orbit，GEO）卫星和中地球轨道（Medium Earth Orbit，MEO）卫星，低地球轨道（Low Earth Orbit，LEO）卫星网络具备更低的传输时延和功耗、覆盖面更广、部署成本更低等优点，因此越来越多的研究开始逐渐基于 LEO 天基网络建设天地一体化网络场景，在地面网络部分，传统地面 IP 网络技术已经发展非常成熟的阶段，同时地面网络设备在计算存储能力、部署成本等方面都明显要优于卫星设备，因此设计基于 SDN 的天地一体化网络（ISTN）架构需要综合考虑天基网络和地面网络的特点，星间链路（Inter-Satellite

Link，ISL）和星地链路（Satellite-Terrestrial Link，STL）之间的差异，重点考虑 SDN 控制器的部署方案和对全局网络的管理方式，提出一种方便有效实现 ISTN 组网和管理的基于 SDN 的 ISTN 网络架构。

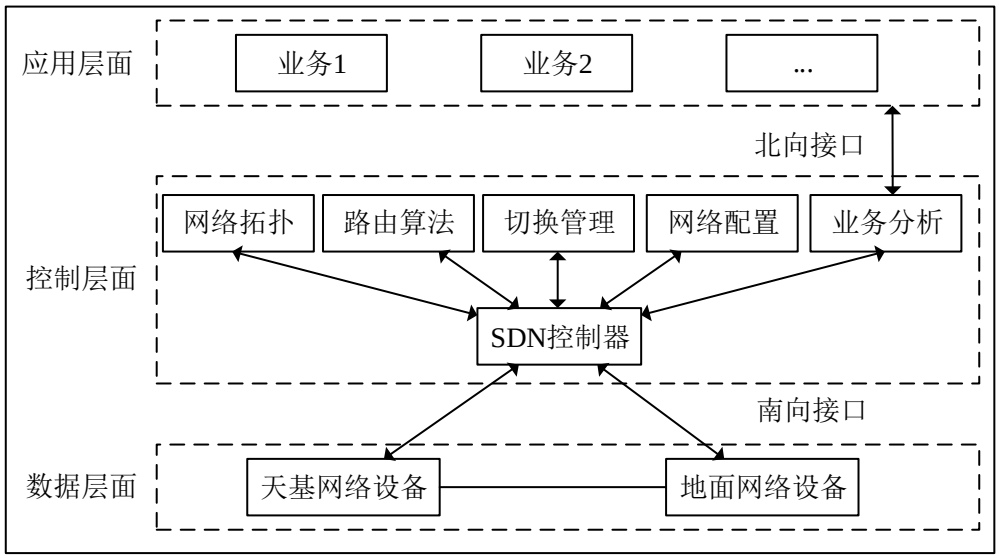

图 2-2 基于 SDN 的天地一体化网络架构

如上图 2-2 中基于 SDN 的 ISTN 架构图所示，SDN 控制器运行在定制网络设备上，经过南向接口对异构网络交换机的工作模式进行统一管理和控制。数据层面中由天基网络设备和地面网络设备构成，一般可以包含骨干卫星、接入卫星、地面终端用户等设备构成，其中主要具备数据转发能力的是低轨卫星。应用层面则是对具体业务应用的抽象描述，通常需要提供给用户可编程式接口，以实现特定业务的配置。

2.2.2 SDN 控制器设计

控制层面中主要包含 SDN 控制器，作为 SDN-ISTN 的控制实体，控制器中一般实现了网络拓扑控制、路由算法、切换管理、网络配置、业务分析等模块。

路由算法则根据网络拓扑实现对应路由算法的计算，并生成对应的路由表，以流表的形式下发到数据层面转发设备。

切换管理模块主要实现对地服务卫星的切换功能，具体工作原理将在本章 2.4 节详细阐述。

网络配置模块是对 SDN 资源管理功能的抽象描述，主要完成全网配置的初始化，实现对 ISTN 中节点链路发现和网络带宽等资源的分配。

业务分析模块则通过北向接口对来自应用层的业务模型进行解析，将业务流进行标准化，根据转换成底层转发设备可以理解的业务流请求信息。

网络拓扑控制模块经过对 ISTN 动态网络拓扑进行管理，可以利用时间分片或逻辑地址分片的方式将 ISTN 网络划分成多个拓扑，具体工作原理将在第三章网络结构设计中详细阐述。

需要说明的是，以上控制层功能模块之间并不是完全独立的，通常有些模块之间需要相互配合共同实现一些功能。例如控制器执行网络配置模块功能后，随即会更新其它模块的初始值，完成 SDN 控制面初始化。而网络拓扑模块和切换管理模块需要交换网络信息完成卫星切换功能，网络拓扑中管理者低轨星座网络中星间链路和星地链路的连接情况，当发生卫星切换时，通常指星地链路连接情况的更新，因此切换管理模块执行完毕时，还需要通知网络拓扑模块更新对应星地链路信息。而路由算法的执行也需要依据网络拓扑模块提供的整体网络拓扑信息和业务分析模块提取的业务数据包信息，本文将在第四章详细阐述新型路由算法的设计细节。

接下考虑控制器的部署问题，在天地一体化网络领域，一般天基网络段作为骨干网络，承担了大量的流量转发任务，通常基于 SDN 架构的天地一体化网络考虑将控制器部署在地面或者部分边缘卫星节点上，以实现天网地网的一体化网络融合。

常见的控制器部署方案有平面式和集中式两种类型[39]。平面式的部署方案中一般存在多个 SDN 控制器共同管理整个网络，所有的控制器在逻辑上都属于全局控制器，管理着全局网络信息，因此处于同一层次。当网络中拓扑信息发生变化或者局部节点链路出现异常情况时，局部控制器首先获取网络变化信息并将该信息同步到其它控制器中。集中式的部署方案中控制器的模型则相对比较简单，通常采用层次化的控制模型，可以将全网中的控制器划分到不同的层次，相对底层的控制器一般管理局部网络区域，较上层的控制通过管理底层控制器以实现全网信息管理和控制。平面式的部署方案通常其控制器部署规模随着网络规模线线性增长，而且同级控制器之间需要考虑数据同步保证网络信息的一致性，当网络出现节点或链路故障时，每个控制器都需要进行相应更新，因此在信息同步过程中，过多的交互指令会增加整个网络的负载，甚至引起网络拥塞。集中式部署方案中由于采用分层式的逻辑架构，可以采用较少的控制器实现对全网络的控制，能有效减少信令开销。但是如果将所有的控制功能模块都交给全局控制器实现，则底层控制器明显没有太大作用，反而在交互过程中会因底层控制器的存在而增加了报文交互时延，不利于网络高效运行和控制。一种较为简单有效的集中式部署模型就是全网只存在一个全局控制器，全局控制器通过直接与底层交换机进行信息交换实现网络信息收集和指令流表下发功能。

2.3 基于 SDN 的天地一体化网络场景

结合前文中对于现有天地一体化网络和 SDN 架构设计的研究，可以将基于 SDN 集中式管理控制的天地一体化网络（ISTN）场景描述如下图 2-3 所示，场景主要由地面网络段和卫星网络段构成，另外通过将控制器以集群的形式部署在地面骨干网络中，利用中间设备和专属通道与卫星和地面设备进行数据交互，从而实现对全网的拓扑信息维护管理和相关指令下发。

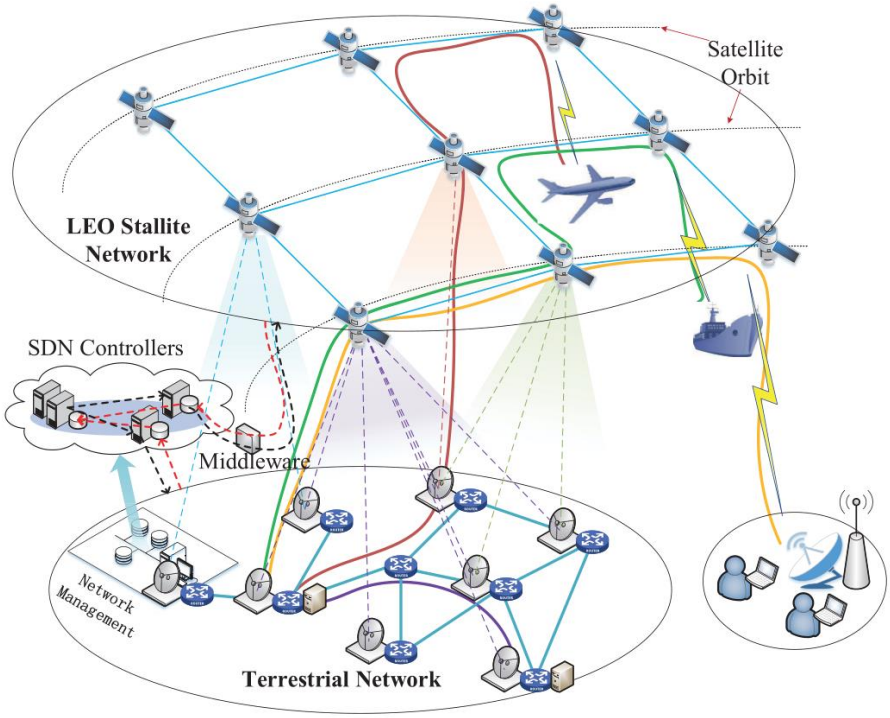

图 2-3 典型基于 SDN 的天地一体化网络场景[40]

在上图所示场景中，天基网络段基于 LEO 星座网络实现卫星网络通信，低轨卫星在确定的卫星轨道（Satellite Orbit，SO）上周期运动，卫星之间则可以利用同轨或异轨星间链路 ISL 进行数据传输，卫星与地面网络设备通过星地链路 STL 实现通信，一般卫星的可见性和星地数据传输受到了仰角等参数的限制，但如本文第一章关于低轨星座系统的特点可知，大多数的星座系统保障了多颗卫星同时覆盖同一地面区域，并且卫星多波束技术的发展也保障了星地链路的通信质量。ISTN 中天基网络一般用于数据业务接入、数据流传输，相比高轨道卫星网络，LEO 的传输时延较低，满足 ISTN 中普通数据业务的 QoS 要求。

在地面网络中包含地面终端节点和地面站节点，如图中网络管理（Network Management，NM）模块所示，SDN 控制器可以通过部署在部分地面站上以实现对 ISTN 的全网管理和控制，控制器通过专属数据流协议（DataFlowProtocol）传递对

应的网络管理/控制命令和流表信息，地面站和卫星节点则被抽离成 SDN 架构中的数据转发器（Switch），直接执行控制器下发的对应的数据转发动作。

2.4 卫星切换策略研究

由于天地一体化网络中低轨卫星相对地面移动速度较快，一颗低轨卫星一般与地面接入终端之间的有效通信时间有限，因此终端需要频繁与上空服务卫星之间进行链路切换。基于 SDN 控制的 ISTN 场景中，终端用户和 SDN 控制器都会有类似的与上空低轨卫星之间发生卫星切换的情况，因此设计一种高效的卫星切换策略可以为 ISTN 中低时延、高可靠的通信提供保障。

参考传统地面移动通信网络中用户切换原理，可以将卫星对地切换过程也可以分成硬切换和软切换两类[41]。硬切换指地面用户与新的服务卫星建立链路之前需要断开与当前服务卫星之间的通信；而软切换则会切换过程当中同时建立和新旧服务卫星的通信链路，当完成呼叫交接后再断开与原服务卫星之间的通信链路。硬切换相对软切换来说，会一定程度上降低呼叫成功率，但是其实现原理更简单，也更节省卫星信道资源，因此本文基于硬切换原理实现卫星切换策略。

在本章前文提出的 ISTN 场景中，低轨卫星所覆盖的地面区域中网络设备会发生两种类型的切换，分别是星内波束间切换和卫星间切换。星内波束间的切换通常是由于移动用户在卫星覆盖区域内的覆盖小区之间的切换，实际上是链路层上的切换过程。而卫星间切换则通常是指卫星的高速移动使得地面终端用户需要频繁地与上空服务卫星之间的星地链路。本文研究的 ISTN 场景主要考虑卫星间切换情况，即终端与不同卫星之间星地链路的切换。

常用的星地链路切换准则主要有以下几种，分别是最近卫星准则、最强信号准则、最小负荷准则和最长可视时间准则[42]。最近卫星准则一般指地面终端在与服务卫星之前的通信链路发生切换时，优先选择最近的潜在服务卫星并建立新的星地通信链路，一般这种方法需要终端收集上空卫星的一些位置信息，包括经纬度、海拔、仰角等，然后判断是否进行卫星切换。最强信号准则和最小负荷准则通常会基于卫星接收信号强度或者卫星负载情况来判断是否进行切换，一般地面用户根据接收到的卫星的信令报文获取到当前服务卫星与潜在服务卫星之间的负载情况和链路信号强度，然后选择服务信号更强或者负载更低的卫星进行切换。最长可视时间准则会综合考虑卫星的运动轨迹和速度、位置信息和对地覆盖面积等参数，然后综合这几种参数来评价潜在服务卫星的可服务时长，进而选择目标切换卫星。

与最近卫星准则相比，最长可视时间准则通常是以牺牲切换时延为代价来减少卫星切换次数，因此需要根据具体场景需求选择切换方案。而最近卫星准则也意

味着卫星仰角更大，即在同等的发射功率和大尺度衰落环境中具有更优的信干噪比，所以星地链路的通信质量更优。

本文综合考虑了各种切换策略的优缺点、实验仿真平台条件以及 ISTN 中星地协同路由设计中对传输时延的敏感性，最终选择使用基于最近卫星准则下的硬切换方案实现 ISTN 场景中的星地链路切换功能。

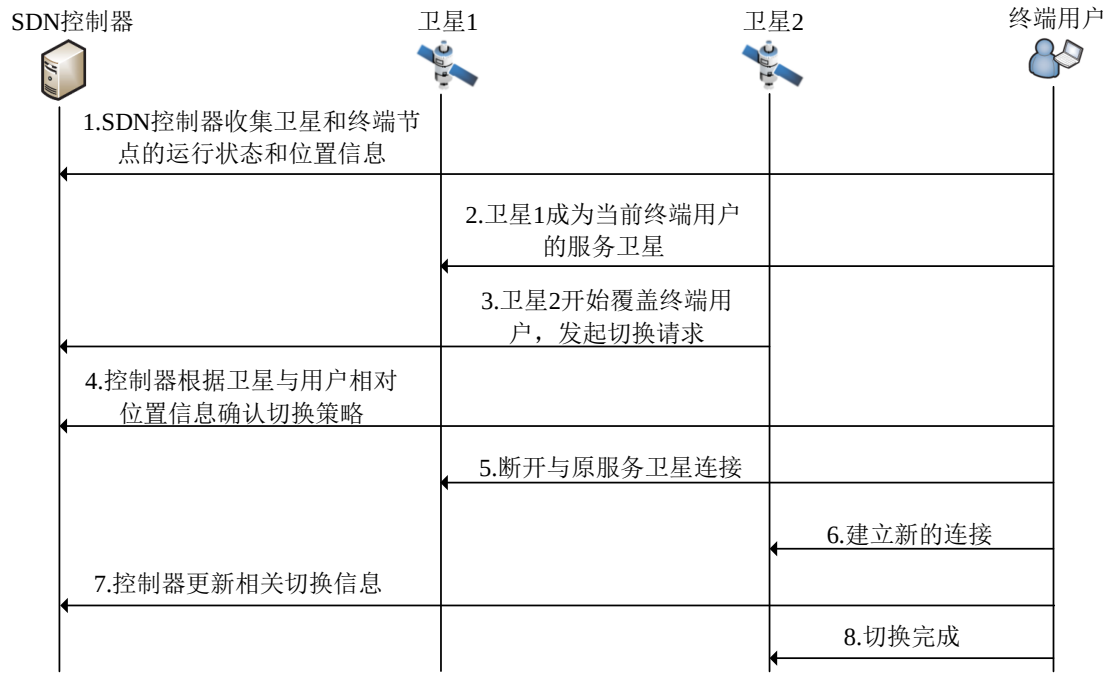

图 2-4 基于 SDN 控制器的卫星切换流程

如上图 2-4 所示 ISTN 中的典型卫星切换流程，基于 SDN 控制器的全局控制能力，所有的卫星切换都可以控制器的指导下完成，使得 ISTN 可以有效地实现星地之间数据连续传输。网络正常运行过程中，SDN 控制器会收集卫星 1、卫星 2 和终端用户的运行状态和位置信息，用于实时管理卫星切换；假设目前卫星 1 与终端用户建立了星地链路，在此运行期间，卫星 2 逐渐运动至终端用户上空区域并开始覆盖终端用户，卫星 2 会开始尝试与终端用户通信并申请星地链路切换，申请经过终端用户转交到控制器，然后控制器执行切换策略相关计算任务，判断卫星 2 是否符合切换条件。如果符合切换条件，控制器则下发允许切换指令，终端首先与原服务卫星断开连接，然后确认回复卫星 2 的服务申请，并建立新的连接，最后上报切换相关信息至控制器并完成此次卫星切换流程。

以上通过 SDN 控制器集中式管理卫星切换的方案中，控制器作为最后切换的决策者，承担着重要地位。在整个用户和卫星通信的过程中，控制器都需要接收来自卫星和地面终端的定期信息汇报，以便控制器监视卫星位置信息和负载情况。这

种集中式的切换管理方案简单有效，同时也减轻了卫星在切换过程中的计算资源开销，非常符合资源受限的低轨卫星星座。

2.5 本章小结

本章首先分析了典型 SDN 架构的特点和控制器、交换机的工作方式，然后根据天地一体化网络中各网络段特点，提出了一种基于 SDN 集中式控制的天地一体化网络架构，并对其中 SDN 控制器的内部功能和部署方式进行了研究和设计。最后基于此架构提出了一种典型天地一体化网络场景，然后针对此场景研究了常见卫星切换策略，并详细分析了基于 SDN 控制器实现卫星切换的整个工作流程。

第三章 基于 OPNET 的天地一体化网络仿真平台设计

本章主要基于前一章中提出的 SDN 的天地一体化网络架构，利用 OPNET 和 STK 软件搭建了 ISTN 网络仿真平台。在天地一体化网络结构设计中，结合了卫星星座网络的运行特点，提出了一种基于 SDN 的拓扑控制机制，最后通过仿真验证了上述网络架构和所提出的拓扑控制机制的可行性。

3.1 STK 和 OPNET 仿真软件介绍

STK（Satellite Tool Kit）卫星仿真工具包是由美国 Analytical Graphics（AGI）公司研发设计的一款专用于卫星通信领域分析软件。本文主要使用 STK 对低轨卫星轨道建模，搭建一个基于准确参数的星座轨道模型，然后导入到 OPNET 中实现天地一体化网络建模。

STK 作为专业的卫星仿真工具，在卫星轨道参数模拟和计算、星上收发机和传感器模拟、对地覆盖情况模拟等方面有着非常关键的优势，STK 提供了非常齐全的卫星模型，非常方便用户进行调用或者二次修改。因此采用 STK 对低轨星座建模要比采用其他网络仿真软件进行自行开发星座相关模型要高效准确得多。基于 STK 的卫星星座建模根据提供的一些轨道参数生成可视化轨道和卫星模型，一般可以通过二维或三位动画显示来查看建模的效果，另外可也以利用 STK 提供的卫星覆盖区域可视化功能和多窗口数据分析功能来验证卫星建模和轨迹建模是否正确。总体来说，STK 是一款功能相关完备的卫星网络仿真工具，但是 STK 也有一些自身的缺陷，由于 STK 不提供网路协议仿真模块，所以它不能直接进行卫星网络协议仿真和验证[43]。

OPNET 是由 OPNET 公司于 1987 年发布的专业网络仿真平台，他能够为通信网络和分布式系统提供全面的模拟仿真开发环境，OPNET 基于离散事件原理来仿真各种模拟系统的行为并分析其性能。OPNET Modeler 提供了相关网络工具的定制化功能，包括网络协议栈设计、数据包格式自定义和网络性能指标收集等，使得 OPNET 可以非常方便地构建大型仿真系统。与其他网络仿真软件（例如 NS2、OMNeT++、Qualnet、MATLAB 等）相比，OPNET 具备模型可拓展、可视化界面、模块可移植、提供标准协议模型库、仿真架构分层等众多优势功能，几乎适用于所有的网络协议和场景仿真。因此本文将选择采用 OPNET 搭建天地一体化网络仿真场景，通过自定义网络设备模型和协议实现来分析验证本文所提出的 ISTN 架构和路由算法的适用性。

每个 OPNET 工程可以由一个或多个仿真场景组成，场景可以看成是工程的子集。OPNET 模型具有如下图 3-1 所示的三个模型层次，包括网络域、节点域和进程域，三个层次相互嵌套以实现设计者的要求。

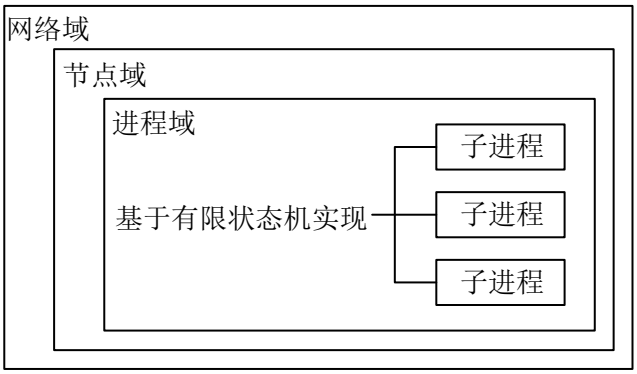

图 3-1 OPNET 仿真模型层次

OPNET 的仿真设计过程如下图 3-2 所示。

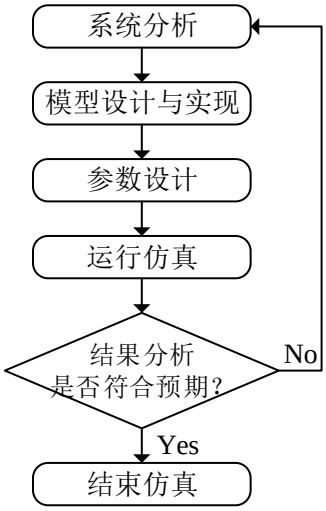

图 3-2 OPNET 仿真设计流程

节点域中的网络设备主要由一些标准库提供的节点模型构成，用户也可以根据具体实际需求自定义节点模型。节点模型是对网络设备抽象，可以根据实际需要定义一些接口供用户方便配置网络节点的属性，从而控制具体每一个 Module 的工作状态。由于 OPNET 在卫星网络仿真场景中没有提供标准卫星模型，卫星网络中节点的高速移动使得网络拓扑动态变化，OPNET 中提供的标准模型库的节点模型和一些路由算法模块多是以 TCP/IP 协议栈建模的，无法直接应用于卫星网络中，所以需要自己根据研究方向定制卫星节点模型，另外结合 AGI STK 软件导入卫星星座轨迹模型。

天地一体化网络中的卫星节点和地面节点应该基于相同的协议栈，地面网络

中的终端节点和信关站节点与卫星节点一样，采用定制化的节点模型。由于星间通信和星地通信是无线网络通信机制，OPNET 中也提供了 Pipeline 管道机制对无线通信进行建模，无线通信建模基于广播机制，建模的本质是对于无线链路的特性描述。无线链路相比于有线链路比较特殊，因为它在网络场景中不可见，OPNET 中提供了管道机制来模拟无线链路的相关特性描述，主要包括 14 个阶段：确定收发信机、计算传输时延、链路关闭(闭包)、信道匹配、计算发信机天线增益、计算传播时延、计算收信机天线增益、计算收信机功率、模拟背景噪声、干扰噪声、计算信噪比、计算误比特率、错误分布、纠错[44]。其中链路关闭阶段是在发信机将数据包发送之后用于确定某个收信机是否能正常收到数据包，这个阶段如果不能正常通过，将不能继续执行接下来的所有阶段，无线链路将会被关闭。

OPNET 的三层仿真模型机制可以有效还原真实物理环境中的网络、设备和协议等，保证网络仿真数据的真实性，而且利用仿真平台提供的多种工具和 STK 工具包，理论上可确保基于 OPENT 和 STK 搭建本文网络仿真场景的方案的可行性。

3.2 一体化网络结构设计

3.2.1 星座轨道设计

星座通信系统通常由若干颗规则分布在预定轨道上的卫星共同组成，常见的星座构造主要分为倾斜轨道星座和近极轨道星座两种类型。倾斜轨道星座通常部署 P 个轨道面，每个轨道面运行 N 颗卫星，由于轨道面间距相同，因此分布均匀，一般又称为 Walker 星座；近极轨星座是对极轨星座的改进，由于极轨星座中轨道是倾斜角为 90°的圆轨道，在轨道面较多的情况下可能会发生极区碰撞，因此将调整了倾斜角，诞生了近极轨星座，如前文所述典型的系统是 Iridium 铱星。需要注意的是，铱星星座中 1 号和 6 号轨道面上的卫星相向运动，而且两轨道面的间距相比其他轨道面之间的间距较大，因此存在反向缝区域，在反向缝之间很难建立星间链路通信，因此实际部署近极轨星座需要考虑反向缝链路断开情况。一般确定星座系统中的轨道的形状、大小和卫星的特征等需要提供如下图 3-3 所示几个参数[45]：

1.轨道半长轴（Semimajor Axis）a，即卫星轨道与地心之间的距离；

2.轨道偏心率（Eccentriity）e：对于圆轨道，偏心率为 0；

3.轨道平面倾角（Orbit Inclination）i：即轨道平面和赤道平面的夹角；

4.近地点幅角（Argument of Perigee）ω：对于圆轨道，ω=0；

5.右升交点赤经（Right Ascension of Ascending Node，RAAN）Ω：即春分点到地心的连线与右升交点连线之间的夹角；

6.卫星初始位置 $\omega+v$：v表示真近点角，即卫星初始时刻在轨道上的幅角，$\omega+v$表示卫星初始时刻与地心的连线和右升交点连线之间的夹角；

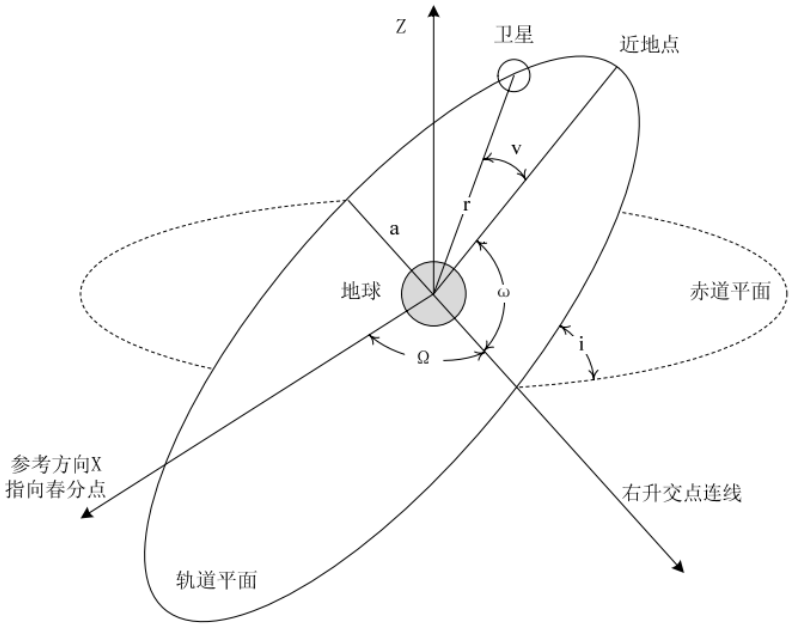

图 3-3 卫星轨道相关参数示意图[45]

本文参考了 Iridium 星座系统的参数特征，利用了 STK 仿真软件设计如下图 3-4 所示的 ISTN 中天基网络 LEO 卫星轨道模型：

表 3-2 LEO 星座轨道参数

低轨卫星数	66
圆形轨道平面数	6
每轨道平面卫星数	11
轨道平面倾斜角 i	86.4°
轨道高度 h	780 km
轨道半长轴 a	7158.14 km
偏心率 e	0
右升交点赤经 Ω	0°
近地点幅角 ω	0°

通过定义 6 条轨道面参数和轨道面上卫星对应参数，如上表 3-2 所示，可以在 STK 三维视图中观察如下图所示三维 LEO 星座模型，然后可以将星座模型文件导入到 OPNET 网络场景中，生成低轨星座网络拓扑。

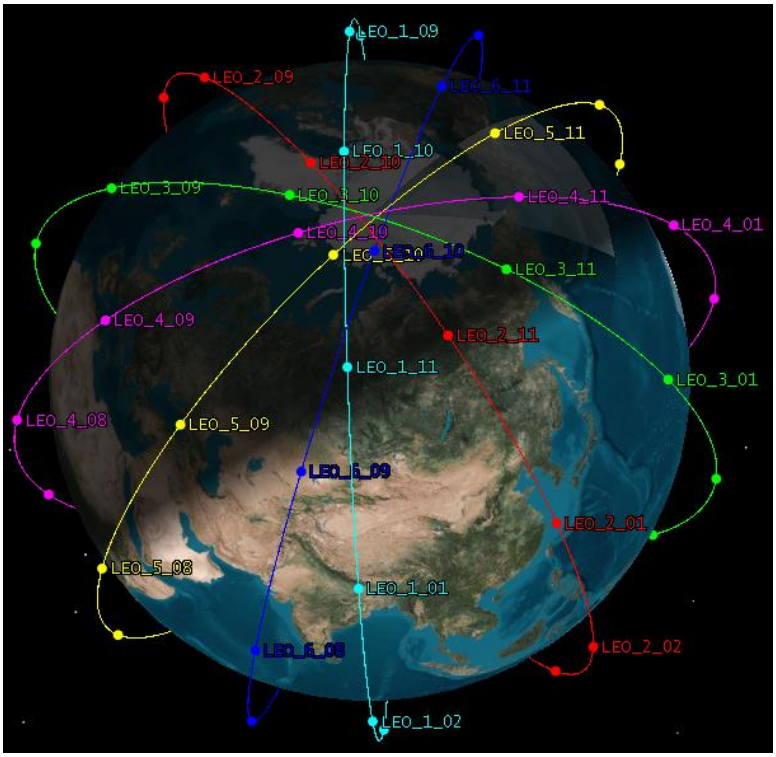

图 3-4 STK 星座三维视图

3.2.2 网络场景搭建

利用前文中 STK 仿真软件设计的低轨卫星星座轨道模型，导入到 OPNET 网络场景中，生成一体化网络中的天基网络段，然后部署一个地面控制器节点和若干地面终端节点，生成如下图 3-5 所示的天地一体化网络拓扑。

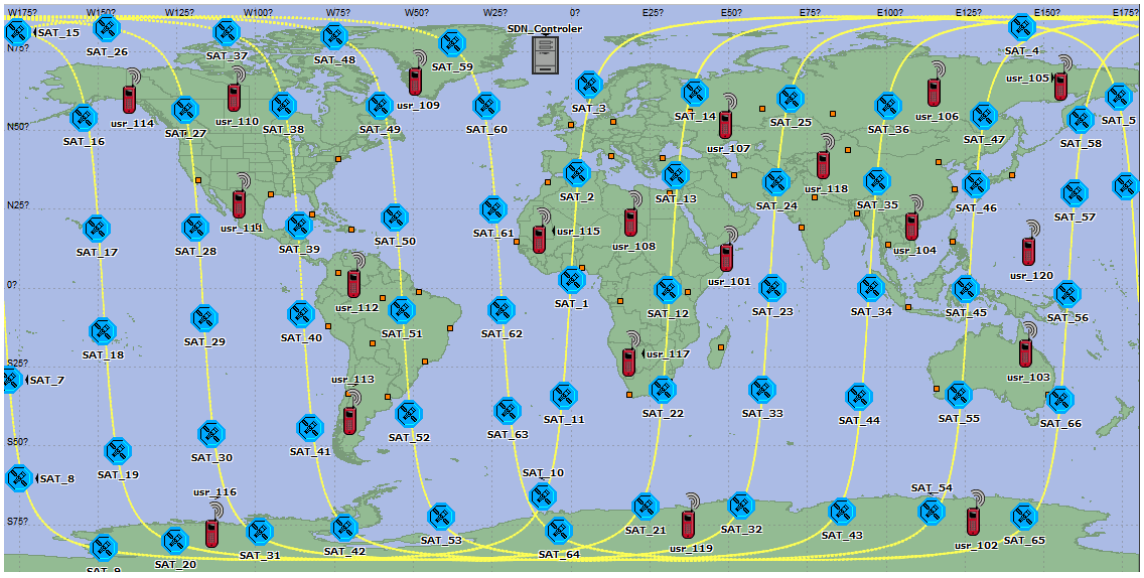

图 3-5 天地一体化网络场景示意图

如上图所示，通过前文中对控制器、卫星交换机和地面终端设备的设计，搭建了基于 SDN 架构的融合地面网络和卫星网络的一体化网络仿真场景。整个网络场景中主要天基网络和地面网络两个部分构成，天基网络段包含 66 颗低轨卫星，均匀分布在 6 个圆形轨道上周期运行，地面网络段则包含一个 SDN 控制器用于全网集中式管理控制和 20 个地面终端节点用于生成和收集业务数据。为了简化仿真过程中地面网络部分的复杂度，场景中的地面终端节点采用固定位置的部署方式，但是由于低轨卫星相对地面高速移动，整体网络中卫星间的数据交互和地面设备和卫星设备的数据交互都存在间断性，因此一体化网络的拓扑依旧保持高动态变化特性。

一体化网络场景中卫星交换机和地面终端设备采用统一编址的方式，方便控制器管理和维护相关数据。对于卫星交换机，SAT_1 至 SAT_11 节点分别对应 1 号轨道上的 11 颗低轨卫星，SAT_12 至 SAT_22 节点分别对应 2 号轨道上的 11 颗低轨卫星，剩余的节点同理可得，全部 66 颗低轨卫星节点的编址方式直接提取节点名称中整数序列号，即 001 至 066。对于地面终端，usr_101 至 usr_120 节点的编址方式与卫星节点相似，即 101~120。SDN_Controler 作为全网控制器节点，采用独立编址的形式，实现对全网络的集中式控制与管理。

3.2.3 拓扑控制方案设计

由于天地一体化网络中的低轨卫星相对地面的高速移动以及星间链路的频繁切换，导致了整体 ISTN 网络的拓扑变化也非常频繁，这增加了 ISTN 中的网络拓扑管理和控制方案的部署难度。结合前文中设计的 SDN-ISTN 网络架构，本小节利用控制器集中式管理网络的优势和 ISTN 中卫星网络的运行特点，研究了一种适用于该场景的自适应拓扑控制方案，具体可以从以下两个方面分析。

3.2.3.1 自适应 Hello 报文发送机制

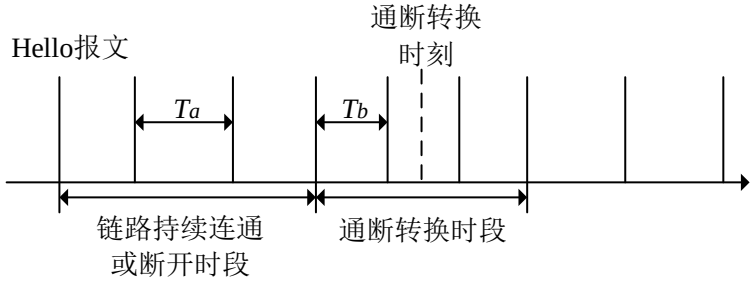

图 3-6 自适应 Hello 报文发送机制

对于卫星节点之间的链路发现过程，通常采用周期性地发送 Hello 报文实现，

如果发送周期过长，则无法及时发现链路变化，如果发送周期过短，则会导致产生大量的控制报文，从而影响链路的有效利用率。本文提出了自适应的 Hello 报文发送机制，从而有效实现网络拓扑控制过程中链路发现的功能，具体原理如上图 3-6 所示。结合前文中快照序列实现动态拓扑发现的原理，已知异轨星间链路的切换相对比较频繁，以链路切换周期（即快照中一个时间片的周期 T）作为异轨星间链路的持续连通或断开时间段，在这段时间内，Hello 报文以正常周期 T_a 发送，减少链路资源的占用；而当处于链路通断转换时段时，则降低 Hello 报文的发送周期，以较小的发送周期 T_b 加快 Hello 报文发送，实现链路连通和断开的快速确认。对于同轨星间链路，在网络正常运行情况下，其链路连接情况相对比较稳定，因此在整个卫星网络运行周期中其链路切换情况是可预测的，可以将 Hello 报文的发送周期调整到卫星拓扑时间片 T，从而在实现链路通断发现和保证链路资源利用率之间取得动态平衡。

3.2.3.2 基于 SDN 的拓扑控制机制

在 OpenFlow 协议中，SDN 控制器基于链路发现协议（Link Layer Discovery Protocol，LLDP）进行链路发现，其中交换机之间的链路发现机制可以通过前文中提出的自适应 Hello 报文发送机制进行优化，而控制器可以通过控制报文收集网络链路信息，通过相应的数据分析来生成网络拓扑结构。

如前文关于 SDN 控制器的设计原理所述，控制器主要通过网络拓扑控制模块实现对 ISTN 网络的拓扑信息收集和管理。在网络初始运行阶段，控制器与交换机之间的通信流程如下图 3-7 所示。

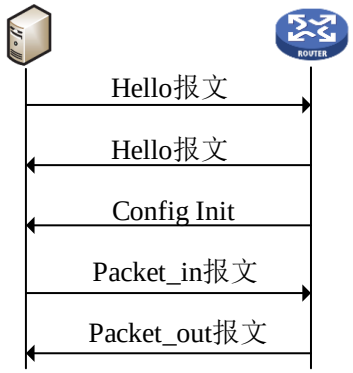

图 3-7 控制器与交换机通信流程

在网络链路信息收集过程中，控制器通过 Packet_out 和 Packet_in 报文与交换机进行信息交互，实现拓扑发现，具体的拓扑控制机制工作流程如下图 3-8 所示。

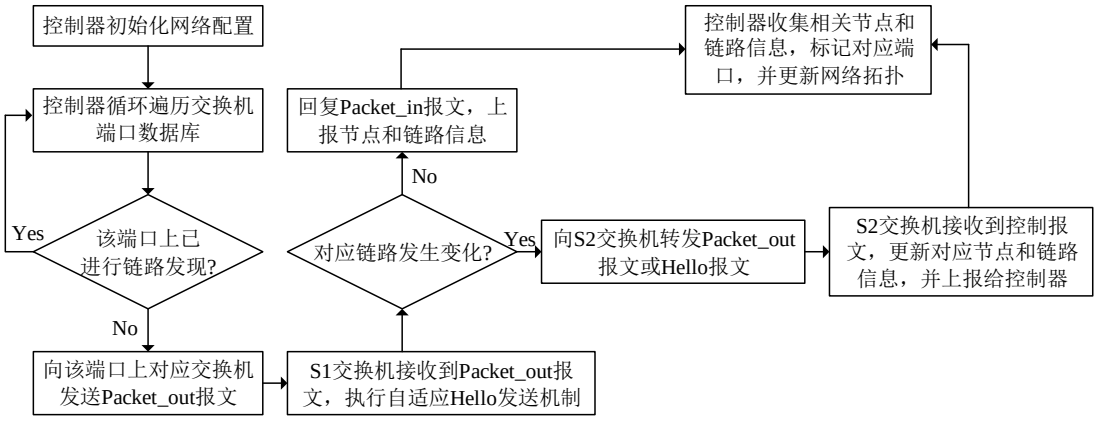

图 3-8 拓扑控制机制工作流程

控制器周期性地向全网络的交换机的所有端口发送 Packet_out 报文，然后等待交换机回复包含本节点信息和链路信息的 Packet_in 报文。另一方面，交换机基于自适应 Hello 报文发送机制发现链路切换时，也会主动上报相关切换信息到控制器。具体步骤如下所述：

第一，控制器与交换机按照上图 3-7 所示流程建立通信，相互同步地址、坐标等节点信息；

第二，开始拓扑发现过程，控制器首先遍历交换机数据库对应的端口信息，如果对应端口上没有相应的链路信息，则生成对应 Packet_out 报文并发送到对应的交换机 S1 上，S1 获取到该信息后，查询自身链路数据库，如果对应端口上的链路信息有效，则直接回复 Packet_in 报文，否则转发 Packet_out 报文到邻居交换机 S2，

第三，S2 更新对应端口上的链路信息，并通过 Packet_in 报文上报自身节点信息和对应端口上的链路信息。

第四，控制器接收到 Packet_in 报文后，查询交换机 S1~S2 对应的链路信息并更新到数据库中，进而更新全网拓扑信息。

为了避免控制器对交换机上某端口对应链路情况的重复发现，拓扑发现过程中控制器可以对重复端口进行标记，因此避免传输重复的控制报文，也能减少处理相关信息的时间，实现更快速的动态拓扑更新，同时也降低了控制器的负载。

3.2.4 数据交互流程分析

接下来分析在 SDN-ISTN 网络场景中典型的仿真数据包处理流程，并分析基于 SDN 架构的天地一体化网络设计是否符合软件定义网络技术中控制与转发分离的思想，进而从理论上验证场景设计的合理性。

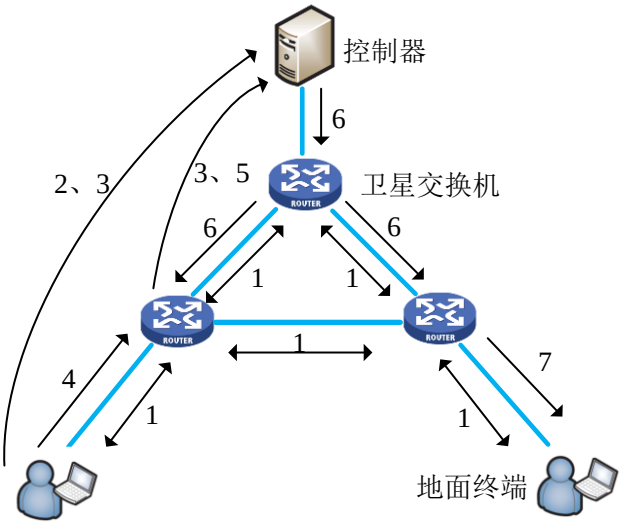

图 3-9 仿真数据包交互流程

本文设计的 ISTN 场景中仿真数据包的处理流程如上图 3-9 所示，其中各步骤的具体处理内容如下：

(1)卫星节点之间通过 Hello I 型报文进行星间链路发现，卫星节点与地面终端节点之间通过 Hello II 型报文进行星地链路发现。

(2)卫星交换机和终端节点通过 Hello 报文向控制器上报自身节点信息和链路情况，控制器根据上报信息获取全网拓扑信息。

(3)随着低轨卫星相对地面不断移动，地面终端与服务卫星之间发生链路切换，切换完成后，卫星节点和终端通过 Hello 报文上报相关切换信息。

(4)终端节点根据配置生成相应的业务数据包，并通过星地链路上传到服务卫星交换机上。

(5)卫星交换机根据业务数据包相应字段，查询转发流表项，若匹配成功，则直接根据匹配项转发该数据包；若匹配失败，则生成 Packet_in 包，向控制器申请对应流表项更新，将数据包插入到转发队列中等待匹配转发。

(6)控制器接收到 Packet_in 包，通过预设静态路由转发表或调用路由模块计算获得相应的转发表项，生成流表数据包 Packet_out，然后下发到对应卫星交换机上。

(7)目的卫星交换机获取到业务数据包，如果目的地址是本卫星节点，则进行相关业务统计，否则将业务数据包下发到地面终端节点。

3.3 SDN 控制器设计

3.3.1 控制器节点模型

在本文提出的天地一体化网络场景中，控制器作为一体化网络架构中的中心

控制节点被部署在地面基站中，从而实现对全局网络的管理和控制。根据上一章中对控制器的设计研究，控制器中的核心功能主要分为网络拓扑管理、路由算法实现、卫星切换、网络参数初始化和业务统计分析这几个模块，另外控制器与卫星交换机和用户终端之间的信息交互通过 Hello 报文、Packet_in 和 Packet_out 报文实现。在本文的仿真设计当中，控制器作为天地一体化网络中的核心设备，其节点模型设计如下图 3-10 所示。

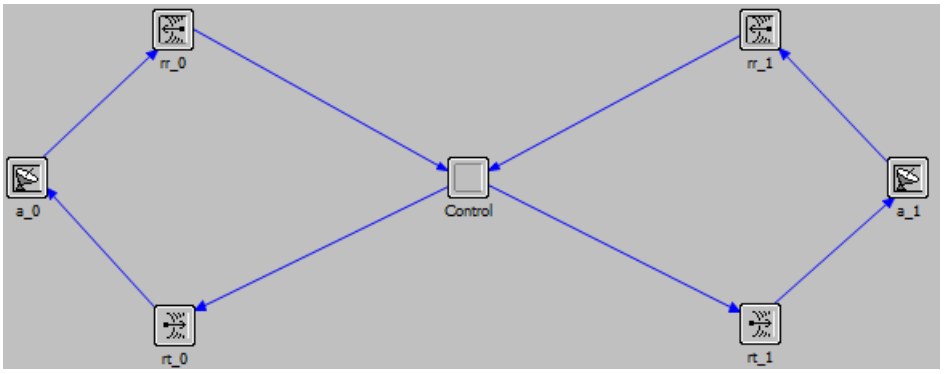

图 3-10 SDN 控制器节点模型

控制器的节点模型较为简单，通过两对全向无线收发设备分别与卫星交换机和地面终端进行数据收发，数据通信主要是用于和交换机建立连接的 Hello I 型报文和用于信息交互的流表信息数据包。核心处理模块 Control 则根据接收到的数据，实时监测和管理网络拓扑信息，另外还持续进行路由计算和相应流表信息下发。

3.3.2 控制器进程模型

控制器的核心进程模块是 Control 模块，其具体处理模型如下图 3-11 所示。

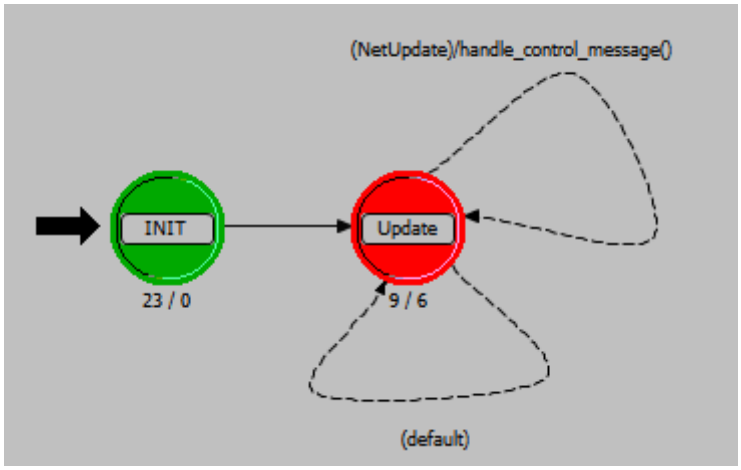

图 3-11 Control 进程模型

模块中的 INIT 状态主要用于初始化控制器的节点属性和配置参数，然后创建

一些用于维护与交换机通信的数据结构，并注册业务统计量，最后设置自中断和包流中断功能。Update 状态中则实现了控制器大部分的功能模块，如前文所述，由于控制器模块内部的功能模块之间存在一定的联系，因此具体的功能实现过程中，采用纯代码实现功能函数调用的方式要比采用有限状态机的状态转移形式更方便，因此下面将详细介绍控制器实现过程中涉及到的数据结构和函数方法的具体作用。

3.3.2.1 相关数据结构

- 网络节点信息结构体：

```
typedef struct node_information{
    int     address;
    Objid   obj;
    int     sat;
    double  lat;
    double  lon;
    double  alt;
    double  x;
    double  y;
    double  z;
    List    nbr_list;
} node_information;
```

node_information 结构体用以保存 ISTN 场景中的卫星交换机节点信息，其中的 address 字段和 sat 字段标识卫星交换机的地址，Objid 对象是 OPNET 平台提供的一种特殊对象类型，用以获取和访问该网络节点对象的数据结构；另外 lat、lon 和 alt 字段分别保存卫星的经纬度和海拔信息，x、y 和 z 字段保存卫星的直角坐标系信息；最后的 List 对象是 OPNET 平台提供的链表类型，nbr_list 对象表示控制器节点通过链表形式存储和管理 ISTN 网络中的交换机设备信息。

- 星间链路相关信息结构体：

```
typedef struct SdnInformation{
    int     address;
    int     isl_num;
    List    isl_list;
} SdnInformation;
```

SdnInformation 结构体用以保存卫星交换机已知星间链路的信息，address 字段

标识卫星交换机的网络地址，isl_num 字段标识目前已发现并注册的星间链路的数量，链表对象 isl_list 则保存了具体星间链路的相关时延和代价信息，链表长度可以通过 isl_num 标识和更新；

- 接入终端信息结构体：

```
typedef struct SdnClient{
    int   address;
    int   sat;
} SdnClient;
```

SdnClient 结构体用于以保存 ISTN 网络中已注册地面终端节点的信息，address 字段标识地面终端用户的网络地址，sat 字段标识目前服务该终端用户的卫星的网络地址。

- 邻居交换机信息结构体：

```
typedef struct nbr_inf{
    int        address;
    double   delay;
    double   load;
    double   time;
}nbr_inf;
```

nbr_inf 结构体用于保存每个卫星交换机节点的邻居节点信息，其中 address 字段标识邻居节点地址，delay 字段标识传输时延，load 字段标识邻居交换机的负载情况，time 字段标识该结构体信息的新鲜度。在本文所提出的场景中，卫星交换机最多与同轨前后两颗卫星和异轨左右两颗卫星建立星间链路，因此邻居表的空间复杂度为 $O(4N)=O(N)$。

另外上文中提出的前三种结构体分别以 List 对象 net_manage_list、sdn_node_list 和 sdn_client_list 三种链表对象的元素形式被注册和管理，由于 OPNET 提供 C 语法形式实现具体代码逻辑，因此对于链表对象的访问只能通过轮询方式实现，数据访问的时间复杂度为 $O(N)$。

3.3.2.2 相关函数接口

控制器进程模型中 Update 状态的状态转移条件 NetUpdate 可以由自中断和接收包流中断触发，自中断用于控制器主动进行网络信息收集和流表下发，而接收包流中断指的是控制器接收到来自卫星交换机或者地面用户终端的数据包。Update 状态中可以根据终端类型和数据包类型调用对应的功能函数，实现前文所述的控

制器功能模块，主要涉及的函数及其功能描述如下。

- handle_control_message()：获取控制消息，并根据消息类型进行相应处理。
- get_address_by_name()：根据节点名称统一对网络设备进行编址。
- receive_datapacket_handle()：控制器接收到数据包，根据数据包类型调用对应处理函数。
- handle_hello_message()：处理接收到的 Hello 数据包，根据 Hello 包类型与对应交换机建立连接，然后生成并发送回复消息。
- handle_packet_in()：解析接收到的 Packet_in 数据包，根据对应字段提取交换机的请求信息。
- send_packet_out()：根据请求报文中各层包头信息，计算交换机申请的匹配表项更新，并填充到 Packet_out 包中对应字段中，下发给交换机。
- manager_update_net_node_information()：根据网络设备地址获取对应节点的 objid，维护更新全网络的设备节点信息。
- node_information_insert_manager()：获取到卫星交换机的注册请求后，创建节点信息对象并初始化，注册到控制器的网络拓扑信息数据库中。
- client_information_insert_manager()：获取到终端用户的注册请求后，创建节点信息对象并初始化，注册到控制器的网络拓扑信息数据库中。
- sdn_manage_list_entry_get(int address)：根据输入网络地址查询并获取对应节点的信息数据结构。
- get_dest_node_information()：获取业务数据包其目的节点的相关信息。
- check_nbr_valid()：检查交换机邻居节点数据是否过期。
- ckeck_packet_valid()：检查接收到的数据包是否有效。
- client_sat_change_handle(Packet* pkptr)：实现卫星切换，根据卫星和终端用户之间的相对位置判断是否执行服务切换。
- routing_algorithm()：路由算法实现模块，根据网络拓扑数据库和业务数据包信息执行路由算法计算。
- calculate_route_table()：根据路由算法执行结果生成路由表，并生成对应的转发流表信息。

3.4 卫星网络设计

3.4.1 卫星节点模型

本文所提出的基于 SDN 架构的天地一体化网络场景中，低轨卫星作为 SDN 交换机，在逻辑架构上既要与控制器之前实现控制面和数据面上的信息交互，在物理

网络拓扑中，还要与其他卫星节点和地面终端设备分别建立星间链路通信和星地链路通信。因此参考了传统 IP 网络协议栈中分层次分模块实现网络功能的思想，设计了如下图 3-12 所示的卫星节点模型。

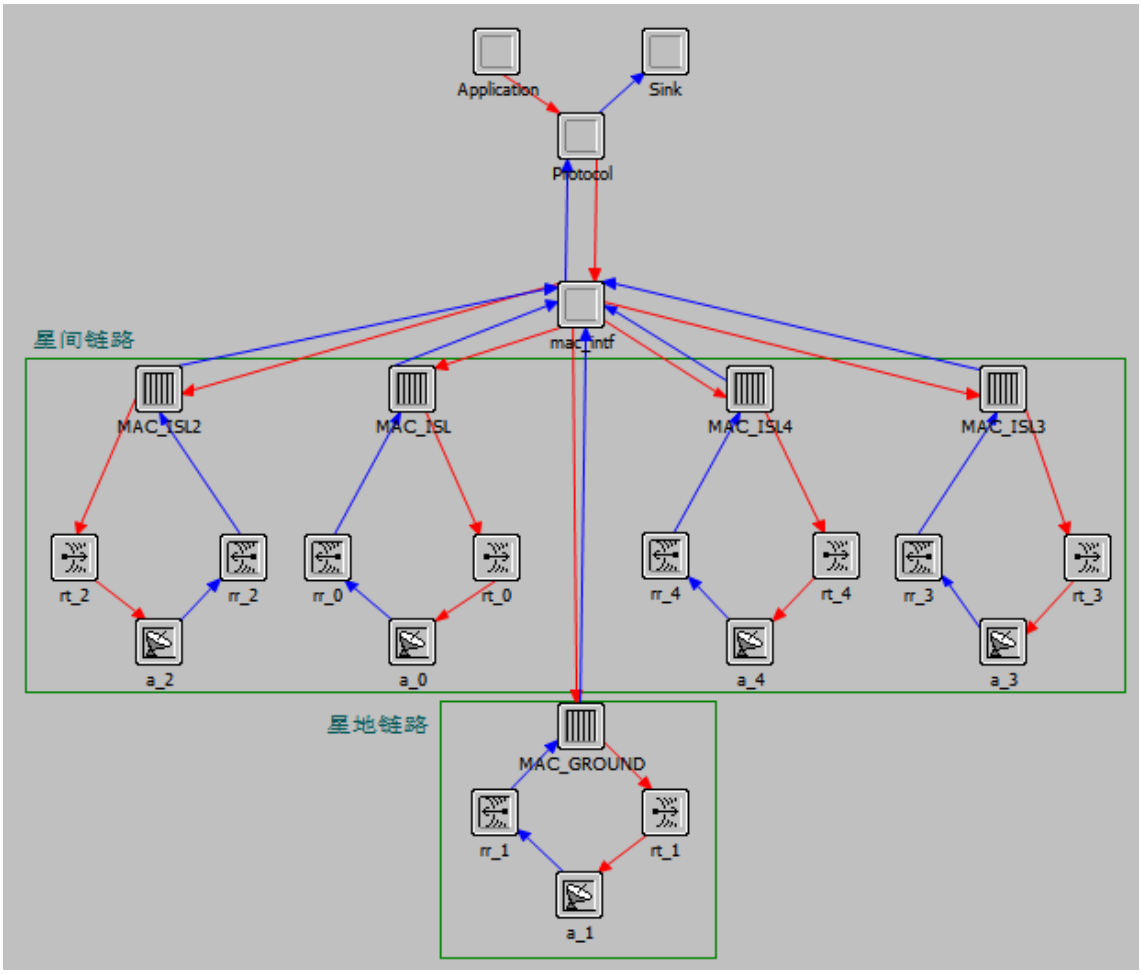

图 3-12 卫星节点模型

如上图所示，卫星节点主要由业务源模块、无线通信收发模块、Protocol 模块、mac_intf 模块、MAC_GROUND 模块和 MAC_ISL 模块构成。业务源模块主要用于生成星上业务数据，并完成相应数据包字段的封装，Sink 模块则用于实现一些星上统计信息的收集和分析；低轨卫星上有四对用于星间链路通信的无线收发信机，它们分别与卫星同轨道的两颗邻居卫星以及异轨道的两颗邻居卫星通信，由于采用近极轨道星座建模，1 号轨道和 6 号轨道之间存在反向缝，无法建立星间链路；另外卫星还有一对用于星地链路通信的收发信机，通常卫星交换机通过这对收发信机接收地面 SDN 控制器和用户终端的流表信息和业务数据。MAC_GROUND 和 MAC_ISL 是链路层的实现部分，主要实现数据包的封装和解封，另外在链路层周期广播 Hello 报文用于邻居发现；mac_intf 模块是链路层和网络层的接口，主要实

现数据包的解析；Protocol 模块是网络层协议的抽象模块，主要进行与 SDN 控制器之间通信协议的封装和解析，维护和更新交换机中用于的数据转发的路由表。当与控制器建立通信之后，卫星交换机接收到来自的控制器的转发流表数据包，Protocol 模块通过解析该数据包后，将对应的转发表端口信息更新，然后用于该卫星转发器上的业务数据包的转发查表操作。可以看出，本文为了简化控制器的逻辑实现，适当地赋予了交换机一定程度的计算能力，方便卫星交换机实现一些用于星间链路通信的简单计算操作，从而降低了控制器的计算负载，但本质上交换机的行为还是受到控制器的指示，依旧遵循 SDN 控制转发分离的思想。

3.4.2 卫星进程模型

卫星节点中各进程模块遵循 OPNET 有限状态机机制来实现，下面详细介绍进程模块的实现细节。

卫星交换机中的 Mac 层协议模块有两种类型，首先介绍 MAC_ISL 进程模块，其进程状态转移模型如下图 3-13 所示。 MAC_ISL 模块中 INIT 状态用于初始化节点 Mac 地址、卫星最大通信距离，计算节点负载情况，初始化邻居链表，开启自中断；WAIT 状态则等待其它进程模块完成初始化；IDLE 即空闲状态，等待中断触发和数据流到达事件；GenerateHello 状态用于星间链路发现，模块会周期性地进入此状态，然后生成 Hello I 型数据包。HLYPacket 状态则表示接收到来自上层模块的数据包，然后直接将数据包根据转发表转发给对应的邻居卫星。PHYPacket 状态表示接收到来自邻居卫星的数据包，该模块需要通过 Seq 字段和 Count 字段检测数据包的有效性，如果有效则上交到 Protocol 模块，否则直接丢弃。

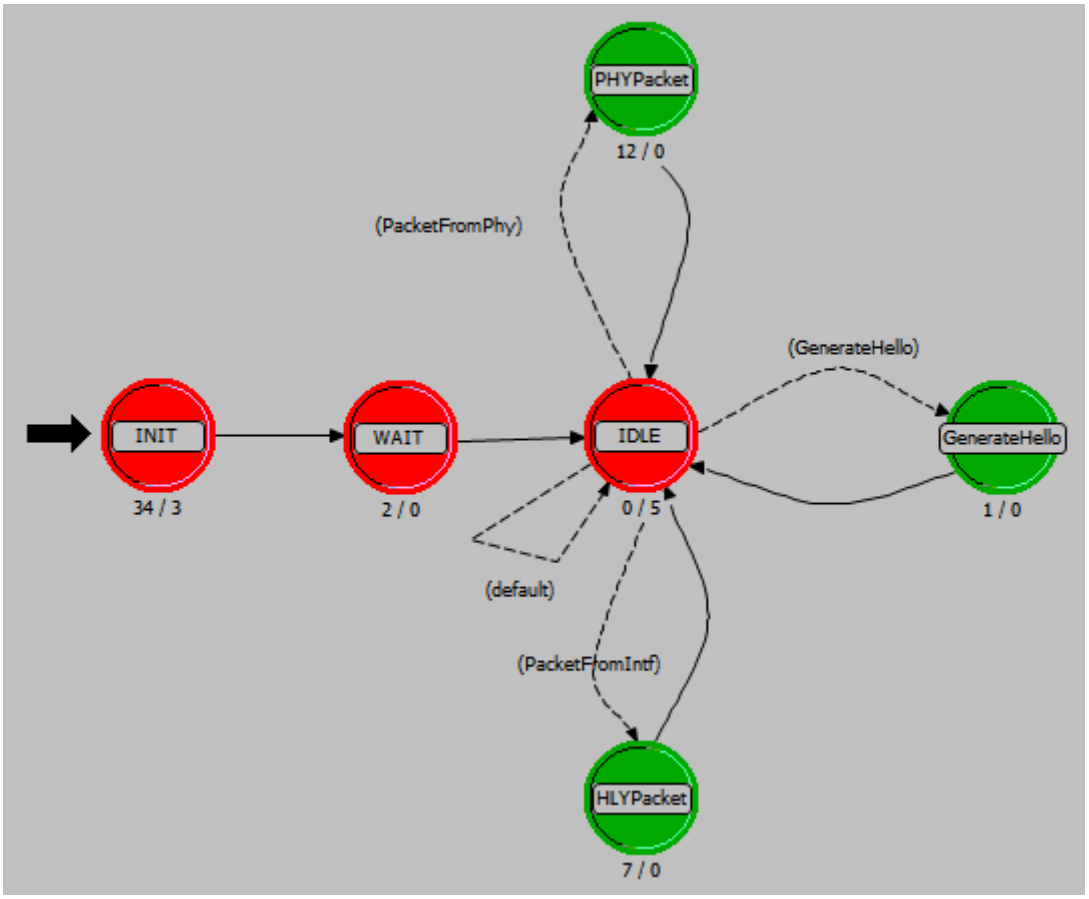

图 3-13 Mac 层进程模型

MAC_GROUND 模块与 MAC_ISL 模块运行类似的状态机模型，但是它们具体的数据处理对象和行为不同。MAC_GROUND 中的 GenerateHello 状态需要周期性生成 Hello II 型数据包用于星地链路发现，然后通过特殊的星地链路收发信机进行数据传输；PHYPacket 状态则表示接收到来自地面控制器或者终端设备的数据包。

mac_intf 模块则是为了方便 Mac 层和 Protocol 层之间数据传递而特意设计的一个逻辑转换层，具体的状态机设计如下图 3-14 所示。

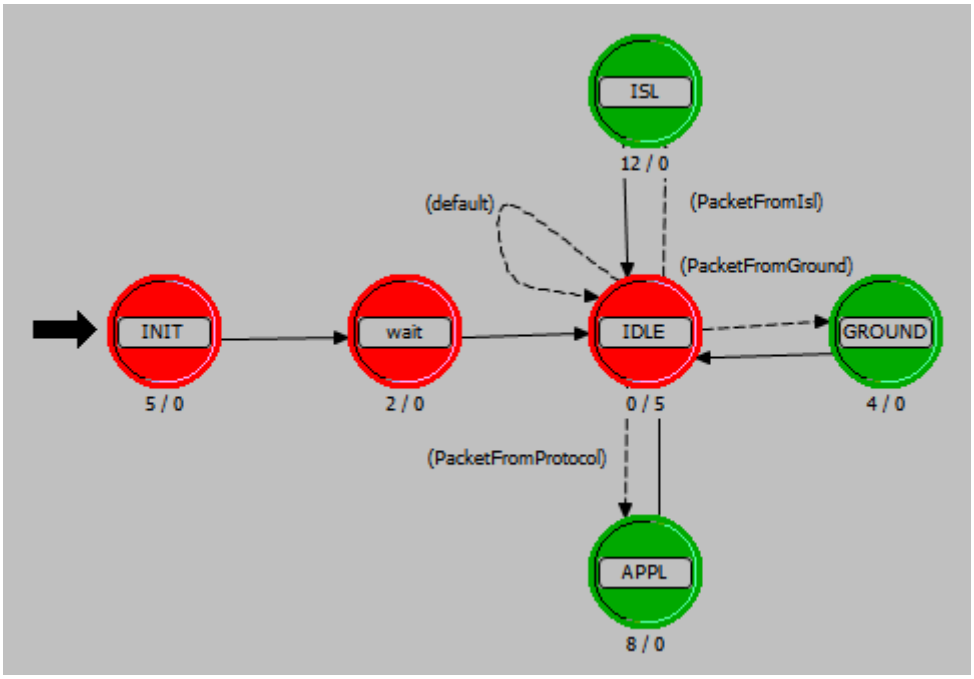

图 3-14 mac_intf 进程模型

mac_intf 模块中的 INIT 状态、WAIT 状态和 IDLE 状态与 MAC 模块中功能类似，用于完成必要的状态初始化。ISL 状态表示接收到来自星间链路的数据包，直接将数据包上交到上层模块；Ground 状态表示接收到来自地面控制器或者终端的数据包，直接上交给上层模块；APPL 状态表示接收到上层的数据包，一般是来自卫星生成的业务数据包 SatData 或者是卫星交换机用于申请更新转发表的 Packet_in 的数据包，模块可以根据转发表将对应的数据包下交到对应的 Mac 层转发队列中等待系统处理。

Protocol 模块是卫星交换机节点模型中的关键模块，如下图 3-15 所示，该模块主要实现了与控制器信息交互的逻辑处理、路由表的维护更新、生成应用业务解析和封装等功能。如前文天地一体化网络中 SDN 交换机的分析可知，交换机中管理维护一个转发表，控制器通过生成特定的转发流表信息控制卫星交换机的转发行为。另外为了增加 ISTN 场景中业务数据的多样性，使卫星交换机可以生成业务数据包 SatData，Protocol 模块需要对该数据包进行部分字段封装。在 Routing 状态中，根据待转发数据包与路由表中的转发字段的匹配结果，如果匹配成功，则将数据包下发到制定的端口，等待转发；如果不匹配，则生成和封装 Packet_in 数据包请求控制器更新对应的流表，然后等待 Packet_out 响应包，通过调用特定的函数对接收到的 Packet_out 数据包进行解析，获取到来自控制器的流表项，然后更新路由表，转发数据包。

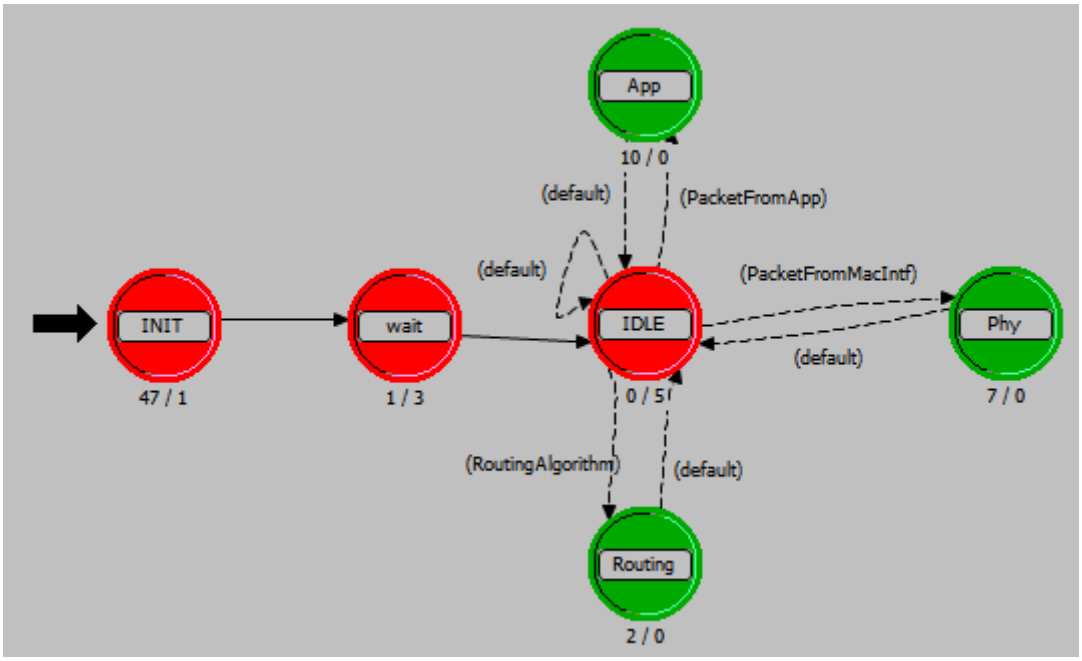

图 3-15 Protocol 进程模型

3.5 地面终端设计

3.5.1 终端节点模型

地面终端节点作为本文天地一体化网络场景中的地面接入用户，是实际业务数据端到端传输的源节点和目的节点，另外也通过与地面基站（SDN 控制器）建立连接，实现入网注册并受控制器的统一管理。地面终端节点模型设计如下图 3-16 所示，主要由一对收发信机、MAC_GROUND、mac_intf、Protocol、Application 和 Sink 模块构成。

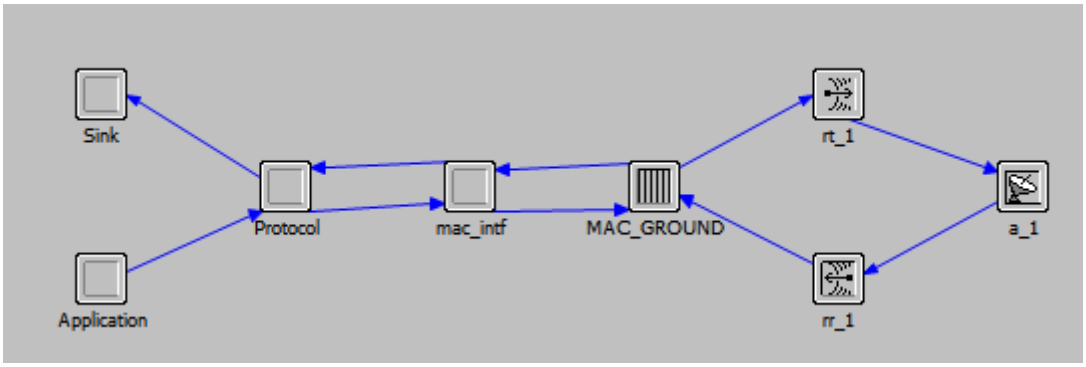

图 3-16 终端节点模型

其中收发信机用于与控制器、卫星交换机进行数据收发，OPNET 中可以通过定义天线的发送接收功率阈值来限制无线通信链路距离，因此本文设计的场景中，

通过配置节点模型的相关属性，使得距离较近的终端用户之间允许一跳范围内的数据传输。MAC_GROUND 模块是链路层协议的实现部分，主要用于星地链路的发现和维护，完成卫星切换过程中的相关信令交互，另外终端节点作为天地一体化网络中的接入设备，通过本模块与控制器之间的信令交互，完成入网注册，然后其卫星切换过程和业务数据转发行为受到控制器的集中式控制和管理。mac_intf 模块是链路层和网络层的接口，主要实现数据包的解析；Protocol 模块则主要实现网络层的功能，包括对应用层业务数据包的封装，以及根据控制器下发的转发流表生成并维护路由表项。与卫星交换机不同，业务数据包在地面终端上的转发依赖当前服务卫星的状态，在数据包通过星地链路上传到低轨卫星网络之前，如果存在尚未完成的卫星切换过程，则数据包需要在链路层转发队列中排队等待切换过程完成。Application 模块和 Sink 模块分别实现业务数据包的生成和收集，并计算一些相关的网络统计量。

3.5.2 终端进程模型

本文设计的地面终端节点模型采用和卫星交换机类似的模型设计，节点的工作行为从逻辑层面上来看与卫星交换机相似，都受到中央控制器的集中式控制和管理；从物理实现上来看，节点模型中各进程模块遵循 OPNET 有限状态机机制来实现，下面详细介绍进程模块的实现细节。

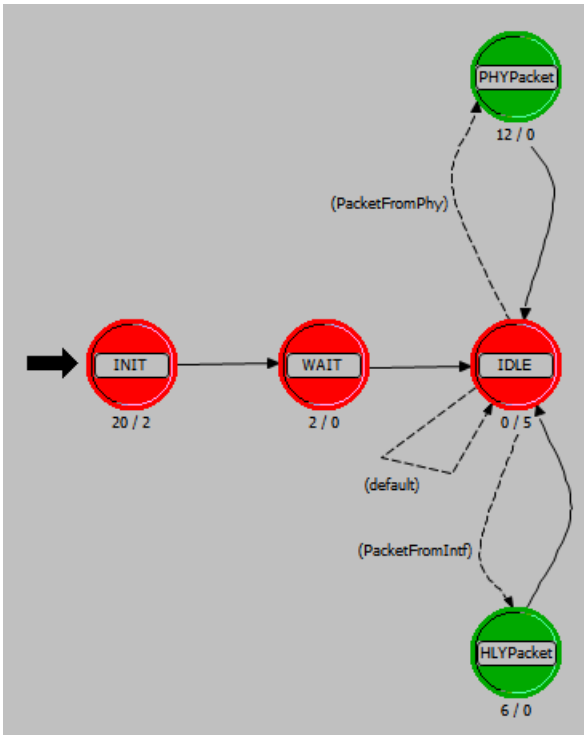

图 3-17 MAC_GROUND 进程模型

MAC_GROUND 进程模块的状态转移过程如上图 3-17 所示，INIT 状态用于获取节点的属性设置并初始化，主要的一个过程就是获取预设的卫星覆盖范围参数，用于确定当前场景下卫星交换机对地面的覆盖面积，然后终端根据该参数确定将来的卫星切换准则相关参数。WAIT 状态用于等待网络场景中其它进程模块的初始化，并启动进程自中断功能；IDLE 即空闲状态，等待中断触发和数据流到达事件；PHYPacket 状态标识接收到来自控制器或者卫星节点的数据包，首先需要包序列号或者 TTL 字段检测数据包的有效性，如果过期则直接丢弃，如果有效再继续根据数据包类型执行对应的操作。如果是来自卫星交换机下发的业务数据包，则根据目的地址决定是继续转发该数据包还是交给上层业务收集模块；如果是来自控制器的 Packet_out 包，则上交给 Protocol 模块执行相关解析操作。HLYPacket 状态标识接收到来自上层的包流，场景中主要存在业务数据包、Hello I 型和 Hello II 型，报文首先解析包类型，填充封装数据包中的对应字段，然后将该数据包插入到 Mac 层转发队列中等待转发。

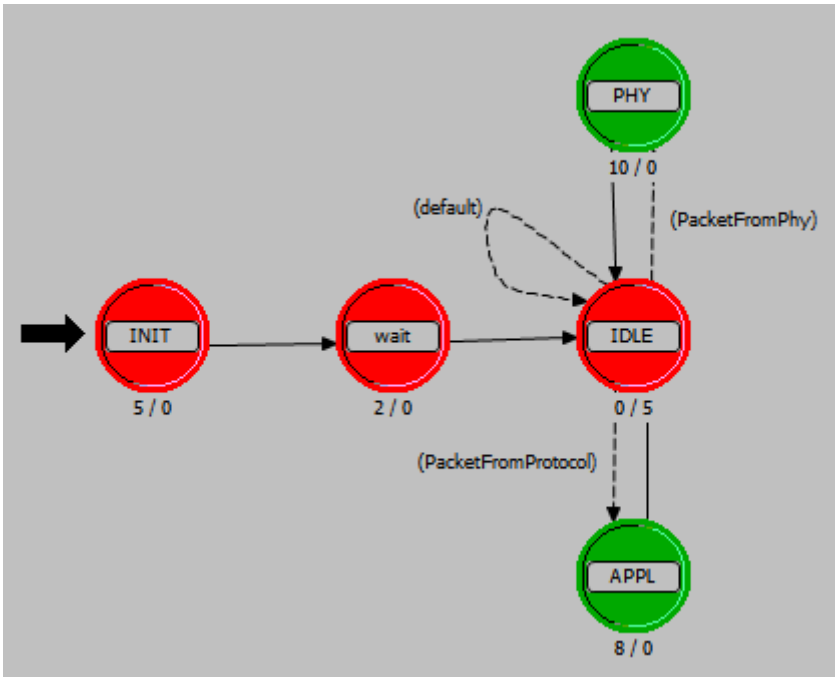

图 3-18 mac_intf 进程模型

mac_intf 模块则是为了方便 Mac 层和 Protocol 层之间数据传递而特意设计的一个逻辑转换层，具体的状态机设计如上图 3-18 所示。mac_intf 模块中的 INIT 状态、WAIT 状态和 IDLE 状态与 MAC 模块中功能类似，用于完成必要的状态初始化。PHY 状态表示接收到星地链路的数据包，直接上交到 Protocol 模块执行相应的包处理；APPL 状态表示接收到上层的数据包，一般是来自终端用户生成的业务

数据包 UsrData 或者是用于申请更新卫星切换的 Packet_in 的数据包，模块将数据包直接下发到 MacGround 模块中进行下一步处理。当该节点处于卫星切换过程时，由于需要从控制器获取用于切换的相关信息，在与控制器的信令交互过程中，业务数据包可以被异步下发到转发队列中等待卫星切换完成，然后执行业务数据转发。

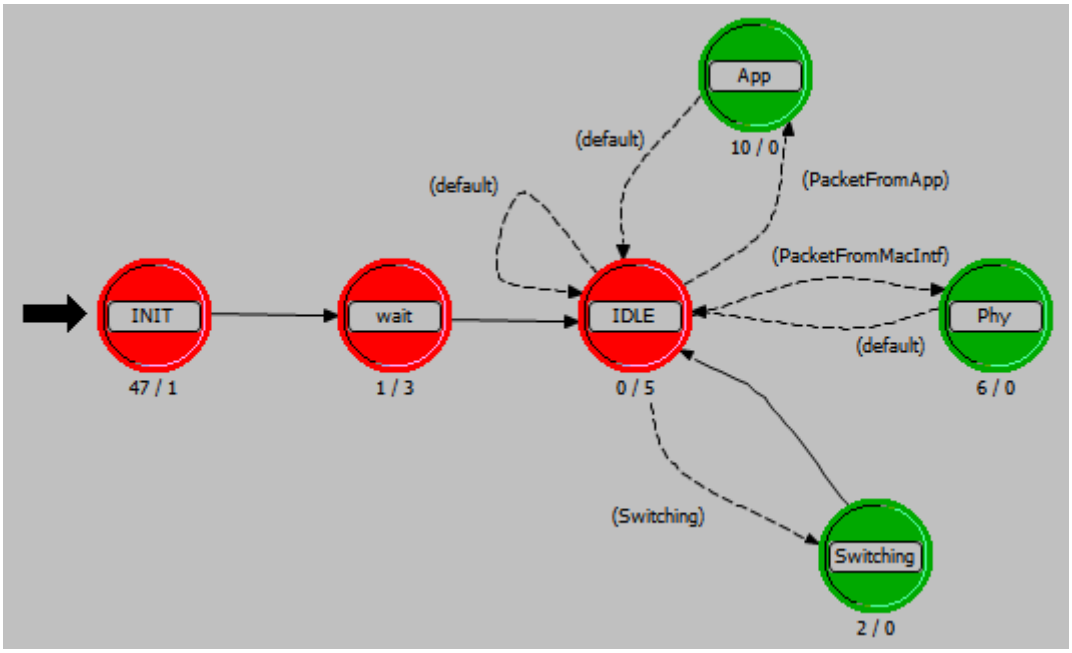

图 3-19 Protocol 进程模型

Protocol 模块的状态转移过程如上图 3-19 所示，本文设计的地面终端节点既是业务数据的收发节点，也是一体化网络中的数据转发节点，其行为受到控制器的集中式管理，具体与控制器或卫星节点的交互逻辑则由本 Protocol 模块内部实现。终端节点主要的逻辑功能包括星地链路发现与卫星切换、与控制器的信令交互与入网注册和业务数据收发等。其中 INIT 状态、WAIT 状态和 IDLE 状态与卫星节点 Protocol 模块中对应状态的功能相似，主要完成节点网络地址、路由表初始化和入网注册等。App 状态表示生成业务数据，该模块将根据用户的业务应用配置封装该数据包，本文设计的业务数据包中定义了预留的 T1~T12 字段可用于记录该数据包的转发路径，或者将直接将静态路由表填充到该字段中进行转发，这都可以根据控制器下发的转发策略来进行相应的配置，具体则在 App 状态和 Switching 状态中实现。Switching 状态用于与控制器的信令交互逻辑解析，终端节点通过 Hello I 型报文向控制器上传本节点的相关信息，完成入网注册，另外当需要进行卫星切换时，该模块接收到来自控制器的 Packet_out 包，通过解析相应字段确定切换准则并更新当前节点和服务卫星状态，最后再通过 Hello 报文回复切换相关结果信息。Phy 状态表示节点接收到业务数据，由于底层进行模块已经进行了有效性检查，所以直

接将该业务数据包上传到 Sink 收集模块进行相关统计量计算。

3.6 协议数据包设计

3.6.1 Hello 数据包

Hello 数据包主要用于两种情况，一种是 ISTN 网路场景中卫星节点和地面终端节点（SDN 交换机）用于邻居星间链路发现和星地链路发现的情况，另外一种情况用于交换机和部署在地面信关站（SDN 控制器）建立南向接口通道的情况。具体的 Hello I 型和 II 型数据包格式定义如下图 3-20 和 3-21 所示。

Seq (8bits)	FrameType (8bits)	Src (8bits)	Dest (8bits)
Load (8bits)	Count (8bits)		

图 3-20 Hello I 数据包格式

Seq (8bits)	FrameType (8bits)	Src (8bits)	Dest (8bits)
Load (8bits)	Flag (8bits)	Lat (8bits)	Long (8bits)
Alt (8bits)	Tsat (8bits)		

图 3-21 Hello II 数据包格式

Hello I 数据包由 Seq、Frametype、Src、Dest、Load、Count 等字段构成，其中 Seq 字段标识该数据包在仿真过程中的序列号；FrameType 字段标识该数据包的帧类型；Src 字段标识其源节点地址；Dest 字段标识其目的节点地址；Load 字段标识源节点的负载指数，主要反映其节点存储和计算能力；Count 字段则标志了该数据包的转发次数。

在用于低轨卫星交换机节点之间链路发现的情况中，各个交换机周期性主动广播 Hello 报文，报文中通过将 FrameType 字段设置为 1 表示是 Hello I 类数据包，然后初始化其它字段，其中 Dest 字段为缺省项，表示所有的节点均可接收该广播信息。当卫星节点 Sat2 接收到来自 Sat1 的 Hello I 数据包时，表明 Sat2 与 Sat1 之间存在可用的星间链路，Sat2 会更新自身的邻居管理数据库，记录用于与 Sat1 通信的链路端口信息和 Sat1 的地址、负载情况等信息。

在用于交换机与控制器建立南向通道的情况中，交换机通过洪泛的形式不断广告自身节点信息，直到控制器接收到该数据包，获取到与控制器的通信请求，然后控制器依据数据包中对应字段注册该交换机信息。交换机可以根据 Hello I 报文

中 FrameType 字段为 2 来识别该数据包是用于与控制器通信的 Hello 包，另外 Dest 字段也标识了该数据包的目的地是控制器，而不是缺省状态用于星间链路发现。在广播的过程中，Hello 报文不断被广播，可能会产生环形回路以及大量过期 Hello 数据包，因此交换机在接收到 Hello 数据包时，首先会根据包的序列号和转发次数来判断数据包是否有效，否则就销毁该 Hello 报文。

Hello II 型数据包由 Seq、FrameType、Src、Dest、Load、Flag、Lat、Long、Alt、Tsat 等字段构成，其中 Seq、FrameType、Src、Dest、Load 均与 Hello I 型数据包作用相似，Flag 字段标识请求卫星切换，Lat、Long、Alt 字段则标识源卫星节点目前的经纬度和海拔信息，Tsat 字段在卫星广播到地面终端的情况中保持缺省值。

Hello II 型数据包用于网络场景中星地链路的发现和切换情况。卫星节点会周期性地通过专用对地通信天线广播该数据包，用于发现星地链路和申请切换，地面用户接收到该数据包后，会根据卫星信息和控制器下发的切换策略确定是否统一切换卫星，并回复该数据包给对应卫星。如果允许用户切换服务卫星，则在切换流程最后，用户还需要与控制器上报相关信息，更新星地链路情况。

3.6.2 Packet_in 与 Packet_out 包

参考 OpenFlow 协议中定义的控制平面和数据平面的信息交互方式，本文根据 OPNET 仿真平台的自身特点和一些限制，自定义如下的 Packet_in 和 Packet_out 数据包格式。

从控制器的角度来看，控制器已经掌握了网络中所有交换机的情况和网络拓扑信息，即整体网络已完成初始化之后，控制器和交换机之间就会根据网路变化进行基于 Packet_in 和 Packet_out 数据包形式的信息交互。Packet_in 数据包主要用于交换机出现无匹配流表项的情况，Packet_out 数据包则主要用于控制器将对应流表策略下发到交换机中。

Seq (8bits)	FrameType (8bits)	Src (8bits)	Dest (8bits)
DataSrc (8bits)	DataDest (8bits)	Count (8bits)	

图 3-22 Packet_in 数据包格式

Packet_in 数据包的格式定义如上图 3-22 所示，由 Seq、FrameType、Src、Dest、DataSrc、DataDest 和 Count 字段构成。Seq、FrameType、Src、Dest 都与 Hello 包中对应字段作用相同，DataSrc、DataDest 字段则来源于交换机中无匹配流表项的

业务数据包，用于表征缺失或无效的转发流表项，而 Count 字段则用于统计该数据包在交换机中的转发次数，降低数据包广播开销。

Seq (8bits)	FrameType (8bits)	Src (8bits)	Dest (8bits)
FlowTable I (32bits)			
FlowTable II (32bits)			

图 3-23 Packet_out 数据包格式

Packet_out 数据包的格式定义如上图 3-23 所示，由 Seq、FrameType、Src、Dest、FlowTable 构成。其中 Src 字段标识控制器的网络地址，Dest 字段标识请求流表更新的交换机的地址，Packet_out 数据包的下发流程中，交换机可以根据 Dest 目的地址进行转发，保障流表策略的顺利下发。FlowTable 字段即用于存储控制器生成的流表信息，目标交换机可以根据特定的字符串解析方法获取对应的控制信息和转发策略。

3.6.3 业务数据包

业务数据包主要由地面用户终端设备生成和收集，为增加网络仿真中业务类型多样性，也允许卫星作为业务源节点生成相应的业务数据包。因此本场景中主要存在两种业务数据包 UsrData 和 SatData，其包格式定义如下图 3-24 和 3-25 所示。

Seq (8bits)	FrameType (8bits)	SrcClient (8bits)	DestClient (8bits)
Src (8bits)	Dest (8bits)	TTL (8bits)	CurrentHop (8bits)
T1 (8bits)	T2 (8bits)	T3 (8bits)	T4 (8bits)
T5 (8bits)	T6 (8bits)	T7 (8bits)	T8 (8bits)
T9 (8bits)	T10 (8bits)	T11 (8bits)	T12 (8bits)
Tsat (8bits)			

图 3-24 UsrData 数据包格式

Seq (8bits)	FrameType (8bits)	Src (8bits)	Dest (8bits)
TTL (8bits)	CurrentHop (8bits)	Tsat (8bits)	
T1 (8bits)	T2 (8bits)	T3 (8bits)	T4 (8bits)
T5 (8bits)	T6 (8bits)	T7 (8bits)	T8 (8bits)
T9 (8bits)	T10 (8bits)	T11 (8bits)	T12 (8bits)

图 3-25 SatData 数据包格式

UsrData 数据包格式定义由 Seq、FrameType、SrcClient、DestClient、Src、Dest、TTL、CurrentHop、T1-T12 预留字段和 Tsat 字段组成，其中 Seq 表示数据包在仿真过程中的序列号，FrameType 字段标识数据包类型，SrcClient 和 DestClient 字段标识用户终端所生成业务数据的源地址和目的地址，TTL 字段规定数据包的生存时间，CurrentHop 字段记录数据包被转发次数，Tsat 标识源节点的服务卫星地址。T1-T12 作为业务数据的预留字段，可以用来记录业务数据的转发路径。SatData 数据包和 UsrData 包类似，由于低轨卫星和地面用户终端采用统一编址的方式，所以 UsrData 包中不需要 SrcClient、DestClient 字段重复指定源地址和目的地址。

3.7 仿真测试

前面几节内容介绍了在 OPNET 仿真平台中基于 SDN 架构的天地一体化网络场景设计实现原理，并详细分析了该场景中控制器、卫星交换机和地面终端节点之间数据包的交互流程。另外在本章 3.2 节中提出了自适应 Hello 报文发送机制，从而实现动态网络中的链路发现。针对同轨道星间链路，由于链路连接比较稳定，其对应节点的 Hello 发送周期 T_a 约等于其链路切换周期 T，对于异轨道星间链路，由于在数据转发过程中，卫星交换机采用异步转发的形式转发数据包，所以需要提前获取到异轨星间链路的通断情况，以便提前更新转发流表，因此对应节点的 Hello 发送周期 T_b 应尽可能小于 T_a，本文设 $T_b = T_a / 2$，一方面可保证节点及时发现星间链路通断情况并上报控制器进行转发流表更新，另一方面也可以有效限制转发 Hello 报文造成的网络开销。

基本的网络仿真场景如前文中图 3-5 所示，本节将通过实验仿真测试 ISTN 网络中控制器的功能实现是否正确，以及控制器与卫星节点和地面终端节点的数据交互是否符合理论分析，最终验证 SDN-ISTN 网络架构设计的合理性。

3.7.1 拓扑控制

仿真测试启动初期，卫星交换机和终端用户会通过 Hello 报文进行链路发现，并上报节点信息给 SDN 控制器，完成入网注册流程。典型的链路发现和节点信息上报情况如下图 3-26 所示。在仿真时间 T=3.00s 时，SDN 控制器获取到了来自卫星节点 Sat1 和终端用户 Usr101 的 Hello I 数据包，控制器通过解析对应字段可以获取并更新该节点的信息数据库和拓扑信息；另外卫星节点之间周期性地通过 Hello I 型报文进行链路探测，实现星间链路探测和维护；卫星交换机和地面终端节点之间定期通过 Hello II 数据包进行星地链路维护，图中仿真时间 T=3.568311s 时，地面终端 Usr101 节点获取到来自当前服务卫星 Sat23 的 Hello II 数据包，表明之前 Sat23 已经和 Usr101 节点建立星地链路，此时 Hello II 报文的交互过程则是星地链路的保活机制。

```
time:2.999453, SatNode 23 rcv Hello I Packet from SatNode 24
time:2.999453, SatNode 25 rcv Hello I Packet from SatNode 24
time:3.000000, SDN_Controler rcv Hello II Packet from Sat_1
time:3.000000, SDN_Controler rcv Hello II Packet from Usr_101
time:3.007568, SatNode 14 rcv Hello I Packet from SatNode 15
time:3.007572, SatNode 16 rcv Hello I Packet from SatNode 15
time:3.179687, SatNode 19 rcv Hello I Packet from SatNode 8
time:3.184632, SatNode 7 rcv Hello I Packet from SatNode 8
time:3.184635, SatNode 9 rcv Hello I Packet from SatNode 8
time:3.295655, SatNode 2 rcv Hello I Packet from SatNode 13
time:3.295655, SatNode 24 rcv Hello I Packet from SatNode 13
time:3.298184, SatNode 12 rcv Hello I Packet from SatNode 13
time:3.298184, SatNode 14 rcv Hello I Packet from SatNode 13
time:3.490823, SatNode 20 rcv Hello I Packet from SatNode 31
time:3.490823, SatNode 42 rcv Hello I Packet from SatNode 31
time:3.491547, SatNode 45 rcv Hello I Packet from SatNode 56
time:3.491998, SatNode 57 rcv Hello I Packet from SatNode 56
time:3.502287, SatNode 32 rcv Hello I Packet from SatNode 31
time:3.502293, SatNode 30 rcv Hello I Packet from SatNode 31
time:3.552963, SatNode 18 rcv Hello I Packet from SatNode 29
time:3.552963, SatNode 40 rcv Hello I Packet from SatNode 29
time:3.553759, SatNode 21 rcv Hello I Packet from SatNode 10
time:3.553952, SatNode 30 rcv Hello I Packet from SatNode 29
time:3.553953, SatNode 28 rcv Hello I Packet from SatNode 29
time:3.561708, SatNode 9 rcv Hello I Packet from SatNode 10
time:3.561712, SatNode 11 rcv Hello I Packet from SatNode 10

time:3.568311, UsrNode 101 rcv Hello II Packet, ServerSat:23
```

图 3-26 交换机信息上报过程

获取到全网所有节点的注册信息之后，SDN 控制器则开始初始化相关节点数据信息，生成对应的邻居图以表示全网拓扑信息和具体的链路信息。此时可以获取到实时的节点信息如下图 3-27 和 3-28 所示，由于篇幅原因，图中只显示了 Sat_1~Sat_6、Sat_60~Sat_66 和 Usr_101~Usr_106 的位置信息。

```
SDN_Controler completed NetWork Topo Init,All Nodes Info as below:
SatNode 1 info:latitude:-0.10, longitude:-0.24, altitude:780003.00,X_pos:7158064.53, y_pos:-30148.61, z_pos:-13098.62
 SatNode 2 info:latitude:32.55, longitude:2.07, altitude:780003.00,X_pos:6029856.28, y_pos:217546.79, z_pos:3851295.22
 SatNode 3 info:latitude:65.10, longitude:7.55, altitude:780003.00,X_pos:2987128.77, y_pos:396177.48, z_pos:6493001.88
 SatNode 4 info:latitude:81.16, longitude:155.90, altitude:780003.00,X_pos:-1003908.32, y_pos:449019.78, z_pos:7073154.71
 SatNode 5 info:latitude:49.07, longitude:175.61, altitude:780003.00,X_pos:-4676216.78, y_pos:359303.78, z_pos:5407667.31
 SatNode 6 info:latitude:16.44, longitude:178.70, altitude:780003.00,X_pos:-6863883.52, y_pos:155513.22, z_pos:2025311.57
```

图 3-27 控制器初始化节点和链路信息 a

```
SatNode 60 info:latitude:48.87, longitude:-26.33, altitude:780003.00,X_pos:4220110.21, y_pos:-2088814.49, z_pos:5391334.91
SatNode 61 info:latitude:16.23, longitude:-23.29, altitude:780003.00,X_pos:6312906.87, y_pos:-2716914.47, z_pos:2001137.39
SatNode 62 info:latitude:-16.43, longitude:-21.20, altitude:780003.00,X_pos:6401402.75, y_pos:-2482407.00, z_pos:-2024516.39
SatNode 63 info:latitude:-49.06, longitude:-18.13, altitude:780003.00,X_pos:4457496.22, y_pos:-1459759.27, z_pos:-5407291.24
SatNode 64 info:latitude:-81.17, longitude:1.37, altitude:780003.00,X_pos:1098389.08, y_pos:26344.20, z_pos:-7073317.16
SatNode 65 info:latitude:-65.12, longitude:150.04, altitude:780003.00,X_pos:-2609443.18, y_pos:1504083.71, z_pos:-6493651.27
SatNode 66 info:latitude:-32.56, longitude:155.48, altitude:780003.00,X_pos:-5488873.92, y_pos:2504314.46, z_pos:-3852225.38
UsrNode 101 info:latitude:61.83, longitude:-140.20, altitude:0.00,X_pos:-2313340.04, y_pos:-1927402.26, z_pos:5622651.51
UsrNode 103 info:latitude:61.30, longitude:-75.50, altitude:0.00,X_pos:766896.74, y_pos:-2965369.67, z_pos:5594558.40
UsrNode 102 info:latitude:35.41, longitude:-105.70, altitude:0.00,X_pos:-1406676.32, y_pos:-5004410.39, z_pos:3695642.02
UsrNode 104 info:latitude:1.15, longitude:-69.40, altitude:0.00,X_pos:2243642.23, y_pos:-5969113.41, z_pos:128008.82
UsrNode 105 info:latitude:-32.83, longitude:-139.30, altitude:0.00,X_pos:-4063174.83, y_pos:-3494881.71, z_pos:-3457895.85
UsrNode 106 info:latitude:-34.60, longitude:-97.20, altitude:0.00,X_pos:-658009.07, y_pos:-5208678.11, z_pos:-3621785.20
```

图 3-28 控制器初始化节点和链路信 b

3.7.2 卫星切换

```
time:8.992319, SatNode 8 rcv Hello I Packet from SatNode 9
Execute Dijkstra Algorithm, calculating route table...
from UsrNode 101 to UsrNode 102
  route path : Usr_101 --Sat_33 --Sat_32 --Sat_43 --Sat_54 --Usr_102
  SumHop is 5
time:9.000000, generate Paket out, fill into flow table, send it to SatNode 101
Execute Dijkstra Algorithm, calculating route table...
from UsrNode 102 to UsrNode 101
  route path : Usr_102 --Sat_43 --Sat_32 --Sat_33 --Sat_23 --Usr_101
  SumHop is 5
time:9.000000, generate Paket out, fill into flow table, send it to SatNode 102
time:9.000000, SDN_Controler rcv Hello II Packet from Sat_1
time:9.000000, SDN_Controler rcv Hello II Packet from Usr_101
time:9.005571, SatNode 43 rcv Hello I Packet from SatNode 32
```

图 3-29 控制器执行路由模块并下发流表

本节仿真场景中，地面终端只负责生成和收集业务数据包，终端节点通过星地链路将数据上传到卫星网络中并最终由服务卫星转发到目的终端节点，数据的路由转发主要发生在卫星网络中。通过在 SDN 控制器中主动计算对应端到端最短路径，并生成流表信息下发到转发路径上的对应卫星交换机上。本节采用了典型的 Dijkstra 最短路径算法实现路由，其中算法采用链路传播时延作为权重因子进行计算。路由计算在仿真控制平台上捕捉到的工作流程如上图 3-29 所示，T=9.00s 时控制器计算出从 Usr101 至 Usr102 的数据转发路径：Usr101-Sat33-Sat32-Sat43-Sat54-Usr102，另外计算出 Usr102-Usr101 的类似最短路由路径，Dijkstra 算法采用贪心算法计算端到端最短路径，可以看出 Usr101 与 Usr102 两节点之间计算出的最短路径均经历 5 跳完成数据路由，初步验证了 Dijkstra 算法在该仿真平台上的正确性。

随着仿真的持续进行，卫星节点相对地面高速移动，会造成网络拓扑的不断更新，因此上文所示三种 Hello 数据包的交互流程也会持续进行，用以更新全网信息。另外地面终端的服务卫星切换过程也在不断发生，通过 OPNET 控制台获取的典型的切换流程如下图 3-30 所示。

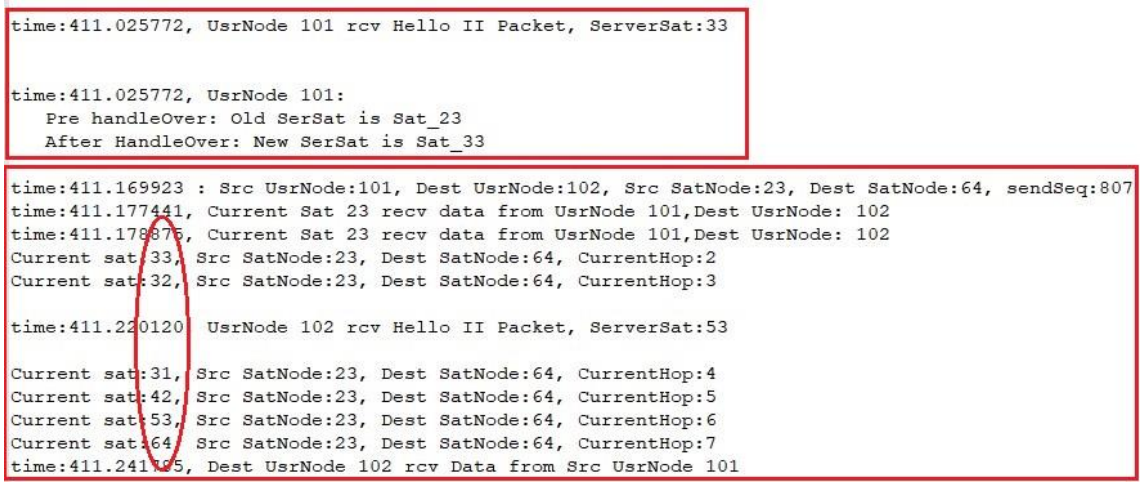

图 3-30 执行卫星切换流程并转发数据包

根据图 3-30 中第一个标记区域信息可知，在 T=411.025772s 时发生了终端节点 Usr101 节点的服务卫星切换过程，Usr101 节点的原服务卫星为 Sat23，此时 Usr101 接收到来自 Sat33 卫星的切换请求，Usr101 执行前文第二章提出的切换策略，并将相关切换信息上报到 SDN 控制器，最终完成服务卫星切换，Sat33 称为 Usr101 节点的新服务卫星。但是从图中可以看到，尽管此时 T=411.025772s 完成了服务卫星切换，但是由于在切换流程启动之前，Usr101 的一部分用户数据已经通过星间链路 Usr101~Sat23 上传到卫星 Sat23（T=411.177441）中，因此 Seq=807 的数据包仍然依靠原有流表策略进行转发，从图中第二个标记区域信息可以看出，该数据包的转发路径为 Usr101-Sat23-Sat33-Sat32-Sat31-Sat42-Sat53-Sat64-Usr102,整个传输时延约为 0.072s。

3.8 本章小结

根据前文第二章中基于 SDN 提出的天地一体化网络架构理论知识，本章利用 OPNET 和 STK 软件搭建了 SDN-ISTN 网络仿真平台。首先阐述了基于 STK 软件实现的低轨卫星星座的建模过程，利用 OPNET 搭建了天地一体化网络场景，并阐述了基于 SDN 的网络拓扑控制机制的相关原理，然后分析网络场景中典型的数据交互流程。然后对网络中的控制器、卫星交换机和地面终端这几种节点的设计细节和内部进程模型的实现过程和相关协议数据包的定义进行了详细阐述。最后仿真测试结果表明控制器与卫星交换机和终端节点可以正确地进行信令交互，实现掌握和管理全网拓扑信息，并顺利通过流表信息集中式控制卫星交换机的数据转发行为，因此仿真验证了基于 SDN 的天地一体化网络架构设计的可行性。

第四章 基于 Q-Learning 的 SDN 路由算法设计

在前一章搭建的 SDN-ISTN 仿真平台上，本章主要研究并设计了一种适用于天地一体化网络的自适应路由算法，该算法基于强化学习模型和 Q-Learning 的基础理论，通过将路由算法模块部署在 SDN 控制器上以实现端到端最优转发路径的计算，本章的路由算法设计和实现部分将主要从网络场景建模、Q 值更新和选路策略三个方面详细阐述该路由算法的原理，最后通过仿真验证了本文提出的路由算法相比于现有其他算法在路由性能上的提升。

4.1 强化学习和 Q-Learning 算法概述

4.1.1 强化学习模型

如本文第一章所述，强化学习作为一种启发式学习算法，由于其算法模型训练过程与网络路由中数据包的寻路过程具有相似的特征，因此强化学习被广泛应用在多种网络场景下用于解决路由问题。基本的强化学习模型如下图 4-1 所示，智能体（Agent）根据自身所处的状态（State）和从环境（Environment）中获取的奖励值（Reward），选择并执行对应地动作（Action），然后应用到环境当中，智能体通过不断地与环境交互，最终实现强化学习模型的构建。

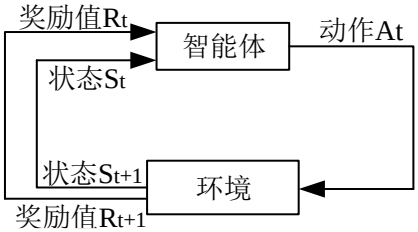

图 4-1 强化学习基础模型

在路由场景当中，智能体是一个路由节点，环境代表的是当前节点的邻居节点，智能体执行的动作表示选择环境中的某一个邻居节点传输数据包。奖励值是环境对智能体采取的动作的反馈值，用于智能体评价该动作的优劣程度。一般强化学习可以由一个四元组{*S*,*A*,*P*,*R*}描述，其中 *S* 表示一个状态集合，*A* 表示动作集合，*P* 表示状态转移概率矩阵，*R* 表示奖励函数。环境模型可以通过状态转移矩阵和对应奖励函数描述。一般强化学习问题的解决方案有两种，分别是基于有模型和基于无模型，基于有模型的强化学习中，智能体可以根据现有环境模型学习经验，从而不断改进自身的动作执行策略。而在无模型的强化学习中，智能体无法根据环境中的

先验知识改进自身策略，此时就不再需要状态转移概率矩阵 P。

假设 π_t 表示在时间 t 时智能体的动作执行策略，a_t 表示在时间 t 时的动作，s_t 表示在时间 t 时的状态，则可以定义在时间 t 时执行动作 a 从状态 s 转移到状态 s' 的条件概率如下所示：

$$\mathrm{P}(\mathrm{s},\mathrm{a},\mathrm{s}')=\mathrm{Pr}(\mathrm{s}'|\mathrm{s},\mathrm{a})=\mathrm{Pr}(\mathrm{s}_{t+1}=\mathrm{s}'|\mathrm{s}_t=\mathrm{s},a_t=\mathrm{a}) \tag{4-1}$$

同时也可以得到：

$$\left|\sum_{s'\in S}P(s'\mid s,a)\right|=1 \tag{4-2}$$

定义智能体在时间 t 时获取到的奖励值为 R_t，处于状态 s 的智能体在执行动作 a 后，会从环境中获取到一个奖励值，定义为 R(s,a)，智能体从环境中学习经验就是通过执行动作然后收集对应的奖励值，学习的最终的目的则是最大化全局奖励折扣值 G_t，所以智能体需要在一段时间内根据获得的奖励值来评价哪些动作是可取的，哪些动作是不可取的，并对相应的状态-动作对(s,a)的奖励值进行折扣统计。

一般对全局奖励折扣值的定义如下式所示[46]：

$$G_t=\lim_{h\to\infty}\frac{1}{h}\sum_{k=0}^{h}R_{t+k+1} \tag{4-3}$$

可以看出，要获取到最优的 G_t 值，需要智能体长期执行动作，并获取对应奖励值，算法执行时间越长，统计到的奖励值越多，算法学习程度就越高，强化学习模型就越准确。另外强化学习中的环境状态集合由于具有马尔可夫属性[47]，即下一个状态的转移只依赖于当前状态和即将执行的动作，因此状态转移不会受到时间 t 之前的其它状态-动作对(s,a)的影响，而且智能体在 t+1 时间获取的奖励值也不会受到 t 时间及以前的奖励值的影响。

4.1.2 Q-Learning 算法原理

Q-Learning 通过引入 Q 函数来描述对应动作 a 的价值，智能体可以根据从环境中获取的奖励反馈，然后使用 Q 函数独立地评价状态-动作对(s,a)的权值，进而更新智能体的动作执行策略。文献[48-49]中详细证明了，只要 Q-Learning 算法在所有状态下重复执行了所有的动作，算法将基于概率 1 收敛到最优动作执行策略。

Q-Learning 中采用了如下式所示的 Q 值更新函数，用于评价对应状态-动作对（s,a）在当前智能体中的价值。

$$\mathrm{Q}(s_t,a_t)=\mathrm{Q}(s_t,a_t)+\alpha\cdot\left[R_{t+1}+\xi\cdot\max_a\mathrm{Q}(s_{t+1},a)-\mathrm{Q}(s_t,a_t)\right] \tag{4-4}$$

化简可得下式：

$$Q(s_t,a_t)=(1-\alpha)\cdot Q(s_t,a_t)+\alpha\cdot\left[R_{t+1}+\xi\cdot\max_a Q(s_{t+1},a)\right] \tag{4-5}$$

其中$\alpha\in(0,1]$学习率因子，表示智能体对新学习知识的依赖程度，当 α=1 时，表示当前智能体的 Q 值更新只基于最新学习的知识。

R_t+1 表示智能体在 t 时刻执行动作 a，从环境中获取到的对应动作的奖励值。

参数因子$\xi\in(0,1]$，即折扣因子，表示智能体对当前执行动作集合中反馈的历史最优 Q 值的依赖程度，也可以理解成是对未来奖励值的衰减值。ξ越高，表示当前智能体 Q 值的更新越依赖以往当前的奖励值。

Q-Learning 的训练目的就是通过一系列的动作选择，智能体基于对应的一系列动作奖励值，逐渐积累经验并依靠 Q 函数转换成对应状态-动作对(s,a)的评估值,即 Q 值，最后依据 Q 值表找到最优的动作执行策略。整个 Q-Learning 算法训练过程就是智能体不断地更新 Q 值表中的 Q 值，然后依据新的 Q 值来判断对应状态下应该执行那个动作。

通过 Q 函数公式中的 maxQ(s_{t+1},a)项可以看出，Q-Learning 是一种离线策略模式的强化学习算法，离线策略模式指的是 Q 值的更新可以不依赖于智能体当前正在经历的训练过程。在路由场景中，当前节点 x 选择某一邻居节点 y 作为下一跳节点进行路由时，第一种策略就是节点 x 可以直接依赖之前的学习经验，根据当前 Q 值表中最优 Q 值对应的邻居节点进行数据转发，然后再更新 t+1 时刻的 Q 值表。第二种策略就是节点 x 可以基于选路策略随机选择下一跳节点，学习新的知识，即忽略之前的学习经验，智能体依靠依靠新的知识对外部环境进行学习，这意味着 Q 学习的训练过程可以是离线执行的，而数据路由过程可以依靠离线训练过程生成的 Q 函数表在线执行，即异步路由。

通过以上对 Q-Learning 的简要分析，可以看出算法的训练过程和网络路由过程具有相似的特性。但是 Q-Learning 本身也存在一些缺陷需要改进，在传统 Q-Learning 算法解决问题的场景中，智能体必须执行一系列动作值来不断获取反馈，从而修改自身的策略模型，这个训练过程的时间可能会由于智能体获取知识的策略不合理而变得特别长，反映在路由场景中，则会看到路由时延在算法执行前期或者整个过程中十分长。本质的原因在于，每一个执行 Q-Learning 的智能体都只能基于局部环境知识进行路由决策（即局部收敛），智能体需要一段时间的容错训练才能获取到最优的路由策略（即收敛时间过长），另外如果外界环境不断变化，智能体需要频繁与环境进行交互，获取变化信息，因此很可能智能体会一直处于算法探索阶段（即路由算法无法收敛）。

经过对 Q-Learning 算法优缺点的分析，可以看出，只要保证智能体有效获取网络拓扑变化的信息，基于 Q-Learning 的路由算法设计是一种可行的方案。前文中所提出的基于 SDN 架构的天地一体化网络中，由于控制和数据分离的特点，工作在控制平面的 SDN 控制器能够实时管理和控制全部网络信息，并利用全局网络拓扑信息进行路由计算，然后指导交换机进行数据转发。本文第三章已经验证了该架构的合理性，从理论上分析可知，将 Q-Learning 算法模块部署在控制器上进行路由计算，可以有效避免路由算法局部收敛和收敛时间过长等问题。

4.2 ISTN-QR 路由算法设计与实现

4.2.1 路由问题分析

一体化网络中路由算法的适用性。由于不同网络场景具有不同的网络特征，适应的路由算法也各有不同。在天地一体化网络下，网络拓扑频繁变化，如果采用基于时间片划分原理的静态路由，虽然可以解决拓扑变化和链路频繁切换的问题，但是这种算法要求网络节点具备较高的存储能力，因此新型路由算法应当结合一体化网络场景特征并且保证路由的适用性。

路由算法的效率问题。首先需要考虑路由算法对应天地一体化网络中如何利用马尔科夫决策过程对路由问题进行建模，同时合理利用 SDN 架构控制面的集中式管理优势，对传统 Q-Learning 算法中迭代更新 Q 值的过程进行优化，将提高路由算法在天地一体化网络中的计算效率。

路由算法性能评价标准。参考传统路由协议，基于强化学习的路由算法应该在传输时延、丢包率等指标上提供保证，另外由于强化学习是一种可以解决多目标优化问题的算法，在网络路由上也应该考虑解决负载均衡问题和具备一定的网络抗毁性，即在可靠性和稳定性能上应有一定的提升。

由于天地一体化网络中卫星与地面站之间呈点对区域性覆盖的信道特性，使得传统的卫星网络路由算法无法直接适用于天地一体化网络场景中，为了实现 ISTN 下卫星网络和地面网络之间的协同端到端路由，首先需要分析异构网络中链路和网络拓扑的一些特点。首先星地链路相对于星间链路和地面链路来说，会受到卫星网络和地面网络的同时约束，如果是以天基网络中卫星转发节点为骨干节点进行路由设计，那么也需要考虑地面终端与卫星网络之间数据传输的星地链路切换问题，从而保障 ISTN 协同路由的可靠性。其次，根据前文中基于 SDN 架构的 ISTN 网络场景设计，通过 SDN 控制器对 ISTN 网络的统一管理模式，可以实现路由转发过程中的路径选择统一配置，通过 SDN 控制器获取全局网络信息，调用路由算法模块进行相应的路由计算，生成路由转发流表，然后下发到工作在 SDN 数

据面的网络转发设备中。通常可以将网络转换成一个携带链路权重的图结构，权值的确定可以依据链路上的延迟性能、可靠性、稳定性和功率等指标，当需要度量多个指标的时候，路由问题就可以转换成一个多目标优化问题。

4.2.2 路由算法原理

4.2.2.1 网络建模

基于前一节对强化学习理论、Q-Learning 算法和基于 SDN 的天地一体化网络路由设计相关问题的分析，本节对 ISTN 网络进行算法建模，提出了基于 Q-Learning 的适用于天地一体化网络的协同自适应路由算法（下文简称 ISTN-QR），并详细阐述路由算法的具体实现原理。首先将前文搭建的天地一体化网络场景建模成一个连接图 $G=<V,E>$，其中 V 表示节点集合，E 表示链路集合，e(x,y)是 E 集合中的元素，表示节点 x 和节点 y 之间的通信链路，那么整体的天地一体化网络连接图结构化描述如下：

$$G(t)=G_s(t)\cup G_t\cup E_{st}(t) \tag{4-6}$$

其中 $G_s(t)=<V_s,E_s(t)>$表示卫星网络段的连接图，$G_t=<V_t,E_t>$表示地面网络段的连接图；V_s 表示整个网络中的节点集合；$E_{st}(t)$表示星地链路的集合。

本章第一节有提及到，强化学习模型具备马尔科夫性质，即处于当前状态下的智能体所执行的下一个动作决定了未来的状态转移，而与过去的状态无关。另外具备马尔科夫性质的一种模型就是马尔科夫链(Markov Chain，MC)。强化学习算法的理论基础是马尔科夫决策过程(Markov Decision Process，MDP)，与 MC 模型相似，智能体在与环境的交互过程中，状态转移和动作决策集合都是离散有限的。但是 MDP 模型相比 MC 模型，其状态的转移不仅与当前状态有关，还与即将执行的动作有关。因此参考强化学习基本模型中环境模型、动作、奖励和状态之间的联系和 Q-Learning 理论基础，可以根据前文提出的天地一体化网络中各对象的特征，给出如下定义：

（1）状态集合 S：定义每一个网络节点 V_i 即一种状态 S_i，因此可以将数据包在转发节点之间的每一次转发可以描述成强化学习模型中从一种状态转移到另一种状态的过程。

（2）动作集合 A：指除了数据包所在节点 V_i 以外所有网络节点的集合，即数据包在当前节点 V_i 中等待路由转发时，节点 V_i 根据特定选路策略从当前动作集合中选择最优的动作 A_i 并执行。动作集合表征的每一个智能体的潜在可执行动作集合，由于在 SDN 架构下控制与数据面分离的特点，实际的路由计算任务被提升到

了控制面，所以本算法中的智能体就是 ISTN 控制器。在 ISTN 网络中卫星节点作为 SDN 交换机，存在若干的邻居卫星节点，这些邻居节点构成了当前节点的动作集合，控制器作为网络全局控制中心，会根据特定的算法策略从该节点的动作集合选择最优动作，然后将其转换成交换机设备可理解的路由表，并下发到对应交换机上。

（3）奖励函数 R：奖励函数用于计算状态 S_i 下执行动作 A_i 带来的奖励反馈，它表示该状态-动作对在该节点上的价值权重，在天地一体化网络中可以理解成是对网络节点计算能力和节点之间传输链路质量的综合评价，所以 R 值越高，表示节点的计算能力和链路质量越好。这个奖励反馈通常是异步的，即在当前时间 T 时节点执行动作 A_i 对应的奖励反馈值将在下一时间点 T+1 时获取到。本文采用的奖励函数如下式所示：

$$R_{\mathrm{x}}\left(d,y\right)=\omega_1\cdot\left(1-T\right)+\omega_2\cdot\left(1-load\right) \tag{4-7}$$

其中 $y\in N^*(x)$，$N^*(x)$表示节点 x 的邻居集合，即 y 是 x 的一个邻居节点；$R_x(d,y)$表示节点 x 与邻居节点 y 通信时获取的反馈奖励值；T 表示两节点之间的传输时延；ω_1 表示时延参数的权重因子；$load$ 表示邻居节点的负载情况，反映存储能力和计算能力；ω_2 表示节点负载权重因子。可以看出，当参数因子 ω_1 和 ω_2 固定不变的情况下，通信链路 e(x,y)的反馈奖励值会随着传输时延和节点负载的上升而下降，即传输时延越高，节点负载越高，数据包选择该链路进行状态转移获取的奖励值越低。

由于一体化网络中，地面节点和卫星节点的计算能力和存储能力存在差异，另外星间链路和星地链路由于物理特性和通信距离等差异也很难进行一致性评价，因此在基于奖励函数计算反馈奖励值时，也需要根据链路类型调整参数因子以实现天地一体化协同路由算法。由于各卫星节点的队列存储能力相比地面设备较为有限，并且星间链路的传输时延要远小于星地链路，因此星际链路上的数据路由过程中更注重卫星节点存储能力的评价，因此参数 ω_1 应该比参数 ω_2 更大，而在星地链路上由于传输时延更大，并且地面设备存储能力更强，路由过程更注重节点间数据传输时延性能的评价，因此参数 ω_2 应该比参数 ω_1 更大。

上公式中符号 T 表示数据包从节点 x 传输到节点 y 的总体传输时延，具体的计算公式如下所示：

$$T=q+s-t \tag{4-8}$$

其中 q 表示数据包在节点 x 中的排队时延，一定程度上也可以反映节点 x 的计算能力；s 表示数据包在节点之间通信链路上的传播时延，在 ISTN 场景中，无

论是星间链路还是星地链路，数据包在链路传播上耗费的时延都要远大于排队时延；最后符号 t 的计算公式如下式所示：

$$t = \max_{z \in N^*(y)} R_y(d,z) \tag{4-9}$$

其中 $z \in N^*(y)$表示节点 z 属于节点 y 的邻居节点，$Ry(d,z)$表示数据包在节点 y 选择节点 z 作为下一节点转发到目的节点 d 的奖励值，即 t 值反映了当前节点 x 上的 R 值更新过程中会参考邻居节点 y 的再下一跳节点 z 与邻居节点 y 之间的奖励值，即该节点 y 与节点 z 之间的奖励值越大，则节点 x 与节点 y 之间的奖励值越大。

4.2.2.2 Q 值更新方式

用于路由计算的控制器作为强化学习模型中的智能体，可以实时根据全网拓扑信息计算路由表，在计算的过程中智能体将维护两个重要的数据结构，分别是用于记录奖励值的 R 值表(RTable)和用于整体评价链路和节点质量的 Q 值表(QTable)。

R 值表中的表项存储的是对应的两个网络节点之间通信链路的奖励值，更新方式如上式所示。

Q 值表中的值反映的是节点选择某动作并执行若干次以后，该节点对该动作带来的整体回报的评价，基本的更新方式如下式所示。

$$\mathrm{Q}_x(d,y) = (1-\alpha)\cdot \mathrm{Q}_x(d,y) + \alpha \cdot R_x(d,y) \tag{4-10}$$

公式中的x, y, d分别表示网络转发节点源节点、下一节点和目的节点，$R_x(d,y)$即上式表示的对应链路上反馈的奖励值，另外$\alpha \in (0,1]$，即算法学习率。可以看出$\mathrm{Q}_x(d,y)$的更新一方面既依赖于原来旧的$\mathrm{Q}_x(d,y)$值，另一方面也依赖于上一次数据转发获得的奖励反馈值，Q 值对两者的依赖程度取决于参数α。$\mathrm{Q}_x(d,y)$值的大小表示目的节点为 d 的数据包目前正处于节点 x 中，当前节点 x 选择从动作集合中选择邻居节点 y 作为下一跳转发节点的回报值，可以看出该 Q 值越大，节点 x 选择节点 y 作为下一跳节点的概率越高。

可以看出，上式中学习率α反映了 Q 值更新过程中对新鲜学习知识的依赖程度，α值越大时，Q 值的更新就越依赖从奖励函数中新学习到的知识。在 ISTN 卫星网络中，大部分卫星节点都存在 2 颗同轨邻居卫星和 2 颗异轨邻居卫星，在 ISTN 网络运行过程中，卫星节点之间的星间链路会随着卫星移动不断变化，由于同轨卫星在相同的预定轨道上同时移动，所以两卫星的相对运动是静止的，即星间链路不会发生变化，另外异轨卫星之间由于移动速度和方向的差异，会出现频繁的链路通断情况。根据以上对卫星网络特点的分析，可以看出由不同类型星间链路反馈的

$R_x(d,y)$值有不同的新鲜度，由此进行 Q 值更新时应该采取不用的学习率，具体的设计如下所示。

对于异轨星间链路，Q 值更新公式基本保持与上式不变：

$$\mathrm{Q}_x(d,y)=(1-\alpha_1)\cdot\mathrm{Q}_x(d,y)+\alpha_1\cdot R_x(d,y) \tag{4-11}$$

对于同轨星间链路，Q 值更新公式如下所示：

$$\mathrm{Q}_x(d,y)=(1-\alpha_2)\cdot\mathrm{Q}_x(d,y)+\alpha_2\cdot R_x(d,y) \tag{4-12}$$

其中参数α_2的计算公式如下所示：

$$\alpha_2=k\cdot(\mathrm{Q}_{avg}/\mathrm{Q}_{max})\cdot\alpha_1 \tag{4-13}$$

其中参数 $\alpha_1,\alpha_2,k\in(0,1]$，参数 k 用于控制学习率因子 α_2 对于因子 α_1 的依赖程度。Q_{avg} 表示当前节点 x 中所有邻居节点对应 Q 值的平均值，Q_{max} 表示目前节点 x 中对应目的节点 d 的所有 Q 值中的最大值，它们的计算公式如下所示：

$$\mathrm{Q}_{avg}=\frac{1}{N}\cdot\sum_{i=1}^{N}\mathrm{Q}_x(d,y_i),\forall y_i\in N^*(x) \tag{4-14}$$

$$\mathrm{Q}_{max}=\max(\mathrm{Q}_{avg}) \tag{4-15}$$

同轨星间链路所对应的 Q 值中，节点 x 选择的下一跳节点 y 与节点 x 处于同一轨道上，相比于异轨星间链路的频繁通断，同轨星间链路的通信相对稳定，选择该链路进行数据包路由而反馈的新链路信息相对较少，因此定义$0<\alpha_2\le\alpha_1\le1$，这保证数据转发过程中，算法模型可以从不同动作中获取反馈信息，并有针对性地更新对应 Q 值。与基本的 Q-Learning 算法中单步执行动作并更新对应 Q 值的方式不同，本文提出路由算法中，节点 x 采取某一个邻居节点y_i用以数据转发时，将会根据公式(4-11)更新对应的 Q 值，与此同时智能体也会根据公式(4-12)更新其它所有未被选中邻居节点对应的 Q 值。

以上根据不同链路特性对 Q 值更新采用不同策略的方式，有效提高了路由过程中对网络变化的适应性，当网路拓扑或网络流量负载快速变化时，当前的 Q 值表存储的 Q 值可能无法有效反映出合理的路由决策，因此需要采用特殊的动作执行策略（选路策略）来对网络环境进行探索，接下来将针对几种常见的选路策略进行分析并根据本文 ISTN 场景特点选择合适的策略。

4.2.2.3 选路策略设计

传统 Q-Learning 算法中由于多智能体进行模型训练过程中，每一个智能体都无法掌握全局网络环境信息，所以算法执行初期需要进行网络探索(exploration)，

然后进入网络信息利用(exploitation)阶段，因此传统 Q-Learning 的收敛时间由探索阶段和利用阶段的执行时间共同决定[50]。对于利用阶段，智能体在现有的有限学习经验中根据动作集合中各动作对应的 Q 值进行决策，如果现有学习经验不能客观全面反映全局网络信息，则基于此信息执行的动作很有可能陷入局部最优状态；对于探索阶段，智能体可以基于一些选路策略与外界环境进行信息交互，更有机会从全局网络环境中获取最优解决方案，探索阶段相比利用阶段有更大的改进空间以保证 Q-Learning 算法模型达到全局最优，因此探索阶段的选路策略将很大程度上决定了数据包路由过程中的性能。常用的选路策略有贪心策略、ε贪心策略、基于动作统计的策略和基于概率分布的策略四种[46]。

贪心策略指在算法执行过程中的每一次路由决策时，智能体都直接根据动作集合中各节点对应的 Q 值选择最大 Q 值的邻居节点作为路由下一跳节点。具体的路由动作执行准则如下式所示：

$$y^* = \arg \max_{y \in N^*(x)} \mathrm{Q}_x(d, y) \tag{4-16}$$

在传统分布式网络场景中，单一地执行贪心策略有一定概率会使智能体无法收敛到最优动作，因此算法执行前期学习到的 Q 值不能反映外部环境的真实情况，基于贪心策略将加剧这种误差带来的影响，智能体将仅仅基于局部网络信息执行局部最优的动作，即路由探索阶段陷入局部最优最终导致算法最终无法收敛到全局最优。

ε贪心策略则是针对贪心策略的缺点进行了一些改进，它允许路由算法在探索阶段以概率ε执行非最优 Q 值对应动作，智能体在每一次做路由决策时，可以通过一定概率下的随机探索过程获取并学习外部新的网络信息，这样可以有效避免持续陷入局部最优的情况。具体的路由动作执行准则如下式所示：

$$\mathrm{y}^* = \begin{cases} random(y) & random(0,1) \le \varepsilon \le 1 \\ \arg \max\limits_{y \in N^*(x)} \mathrm{Q}_x(d, y) & others \end{cases} \tag{4-17}$$

基于动作统计的策略，通常需要统计节点的所有动作的执行情况，比如每种动作的执行次数和成功次数，然后实时计算所有动作的成功概率。智能体可以从节点目前的可选动作集合 A 中选取最大概率的动作并执行路由。

$$P_x(d, y_i) = \frac{e^{\beta \cdot \mathrm{Q}_x(d, y_i)}}{\sum\limits_{y_i \in N^*(x)} e^{\beta \cdot \mathrm{Q}_x(d, y_i)}} \tag{4-18}$$

基于概率分布的策略，可以通过某种特定的概率分布描述动作集合中所有动

作的执行概率，智能体通过可以直接依据该概率分布选择并执行对应动作，比如 Boltzmann 分布是一种常用于路由场景的概率分布公式，其具体动作执行准则如上式所示。基于概率分布和基于动作统计的策略相似，由于不能严格地描述算法动作过程中对于动作集合的决策准则，另外在实际实现过程中，两种策略的复杂度较大，因此不适用于本文提出的路由场景。

基于以上对四种动作策略的分析，可以看出执行ε贪心策略的智能体通常可以避免算法无法收敛的情况，同时保证以较高效率选择并执行最优 Q 值对应动作，因此在分布式网络场景中可以获得较好的效果。但是在本文 ISTN 场景中，受益于 SDN 控制器带来的集中式网络管理和控制，路由算法的执行过程中可以大幅度降低网络探索的次数，并通过多次迭代计算的方式充分利用全局网络拓扑信息，从而计算出最优路径。

由于基于 SDN 的 ISTN 网络场景中路由不会陷入局部最优的情况，ε贪心策略也就不适用于本文提出的路由算法中，另外再考虑贪心策略在本文网络场景中的适用性。前文中提到由于分布式智能体无法掌握全局网络拓扑信息，贪心策略在分布式网络路由场景中可能导致算法无法收敛，但是本文中通过将执行路由算法的智能体部署在控制器上，将不会出现这种动作执行局部最优或算法无法收敛的情况。另外智能体基于最大 Q 值的准则执行对应动作可以有效降低了路由算法探索阶段的执行时间。取而代之的是，ISTN-QR 路由算法的时间复杂度主要体现在 Q 值迭代计算过程和路由表更新过程。

4.2.3 算法流程和复杂度分析

基于以上对 ISTN-QR 路由算法中奖励函数、Q 值更新函数以及选路策略的定义，可以将本文基于 SDN 架构的 ISTN 网络场景中路由算法的工作基本步骤描述如下表 4-1 所示：

表 4-1 ISTN-QR 路由算法执行流程

1. 初始化相关参数α_1、α_2、ω_1、ω_2 ，初始化奖励值 R 表；
2. 将 Q 值表中各元素赋 0，初始化路由表；
3. 控制器从网络中获取源节点到目的节点的路由更新请求；
4. 根据源-目的节点之间路由请求进行如下计算：
   4.1 设置源节点为初始状态x，目的节点为最终状态d；
   4.2 令当前状态$s = x$;
   4.3 根据公式（4-7）计算当前状态下邻居节点相关链路的奖励值；
   4.4 利用公式（4-16）对应选路策略执行特定动作，转移到下一状态s';
   4.5 根据公式（4-11 和 4-12）更新对应 Q 值；
   4.6 令当前状态$s = s'$;

4.7 如果$s == d$，执行步骤 5；否则，执行步骤 4.3；

5. 针对每对源-目的节点路由请求：

5.1 确认状态$x \to d$转移过程中执行的动作；

5.2 更新每一种状态下对应的路由表项；

5.3 当前关于 $x \to d$ 的路由计算完成；

6. 执行步骤 3；

以上关于 ISTN-QR 路由算法的执行流程主要阐述的是一次源节点到目的节点之间路由最短路径的计算过程，不同于分布式网络 Q-Learning 中智能体单步执行的原理，ISTN-QR 路由计算是在掌握全局网络拓扑信息的控制器上执行，因此如步骤 1 所述，关于状态转移过程中奖励值 R 的更新过程可以提前集中式计算并更新。另外针对每一对 $x \to d$ 状态转移过程中的路径计算流程，为了降低时间复杂度，步骤 4 和步骤 5 实际上也可以并发执行，因此可以基于上述流程得到 ISTN 网络场景中路由算法伪代码如下表 4-2 所示：

表 4-2 ISTN-QR 路由算法伪代码

Input：节点数量 N，节点集合 A，参数α_1、α_2、ω_1、ω_2等

Output：*QTable*，*RouteTable*

```
1.  //初始化 RTable 和 QTable
2.  init: RTable[N][N], QTable[N][N], RouteTable[N];
3.  for each i , j in N
4.      QTable[i][j] = 0, RTable[i][j] = -1;
5.  for each entry in A[i].nbr_list
6.      update RTable[i][entry]
7.  for each (x,d) in A：
8.      //确定初始状态和初始动作
9.      next_entry = A[x].nbr_list[random(0, A[x].nbr_list.size())];
10.     //迭代更新路由
11.     while next_entry != A[d]
12.         for each next_entry in entry.nbr_list   //执行选路策略
13.             if QTable[entry][next_entry] > MaxReward then
14.                 MaxReward = QTable[entry][next_entry];
15.                 MaxAction = next_entry;
16.             end if
17.         for each next_entry in entry.nbr_list   //更新 Q 值
18.             update QTable[entry][next_entry]
19.         next_entry = MaxAction;
20.     end while
21.     for i from 0 to N   //计算路由表
22.         if QTable[x][i] > MaxValue then
23.             MaxValue = QTable[x][i];
```

```
24.                 nextNode = i;
25.             end if
26.         RouteTable[0] = nextNode;
27.         while nextNode != A[d]
28.             for i from 0 to N
29.                 if QTable[nextNode][i] > MaxValue then
30.                     MaxValue = QTable[nextNode][i];
31.                     nextNode = i;
32.                 else if QTable[nextNode][i] == MaxValue then
33.                     rand = random(0,1);
34.                     if rand < 0.5 then
35.                         MaxValue = QTable[nextNode][i];
36.                         nextNode = i;
37.                     end if
38.                 end if
39.         end while
```

接下来分析 ISTN-QR 算法的时间复杂度和空间复杂度性能。对于时间复杂度，在本文采取的 ISTN-QR 算法中，网络节点规模为 N，算法执行初期对于相关数据结构进行初始化占用的时间复杂度为 $O(N^2)$，接着针对每一条源节点 x 到目的节点 d 的最短路径计算过程采用 DFS 深度优先遍历原理，因此基于 Q 值迭代更新的过程占用时间复杂度为 $O(4k \cdot log_2 N)$，其中 k 表示 Q 值表的迭代更新次数，定义 $K=4k$，即 $O(K \cdot log_2 N)$。然后基于当前 Q 值表计算节点(x,d)之间的最短路径，这部分的时间为 $O(N+N \cdot log_2 N)$。由于 QTable 和 RTable 等数据结构的初始化过程只执行一次，不纳入复杂度统计中，因此算法的时间复杂度主要是由 Q 值迭代更新和路由表计算两部分构成，因此要得到一条基于 ISTN-QR 算法计算的源节点 x 到目的节点 d 之间最优路径的计算时间复杂度为 $O(K \cdot log_2 N)+O(N+N \cdot log_2 N)$，即 $O((K+N)\ log_2 N)$。

对于空间复杂度，一般可以从算法运行临时占用存储空间、算法代码占用空间和输入输出数据占用空间三个方面综合衡量。本章提出的 ISTN-QR 路由算法实现中，SDN 控制器作为路由计算的实体网络设备，管理和维护了全部的网络节点和链路信息数据，具有较强的计算能力和存储能力，路由算法的评价指标中端到端传输时延是一个重要指标，因此路由算法采用以空间换时间的方式优化路由计算，从而提高路由时延性能。用于存储全网节点信息和链路信息的存储空间最大为 $O(N+4N)$，算法运行期间用于存储 RTable 和 QTable 的存储空间为 $O(2N^2)$，另外算法输出结果 RouteTable 占用存储空间为 $O(N)$，则可得本文 ISTN-QR 路由算法的空间复杂度为 $O(6N+2N^2)$，即 $O(N^2)$。

4.3 仿真与结果分析

4.3.1 仿真场景与参数设定

基于以上对 ISTN-QR 路由算法的设计原理和实现细节的阐述，本节基于前文提出的 SDN-ISTN 网络场景对 ISTN-QR 路和传统 Dijkstra 路由、FEQ-Routing 以及 Flooding 路由进行了性能仿真对比，具体的网络拓扑如下图 4-2 所示。

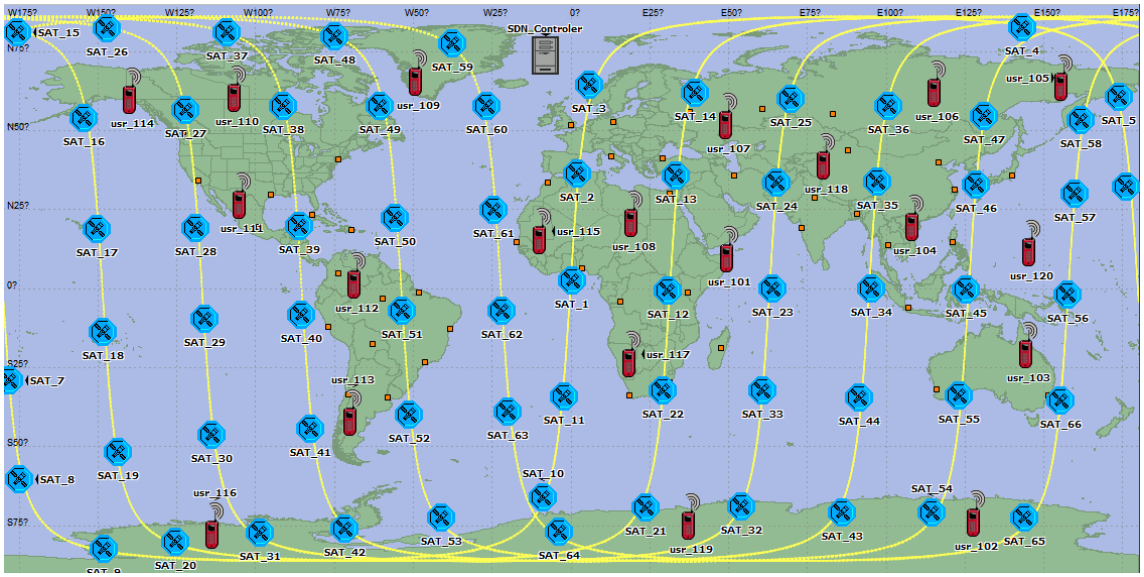

图 4-2 ISTN-QR 路由仿真场景示意图

在仿真场景中，由均匀分布在 6 个圆轨道面上的 66 颗低轨卫星构成天基网络段，地面网路段中部署了 20 个终端节点，另外部署了一个控制器节点以实现对全网的集中式管理和控制。通过将 ISTN-QR 路由算法实现模块部署在 SDN 控制器中，可以实时根据全网拓扑信息计算源节点-目的节点之间的最短路径，基于本章前文中路由算法的相关实现原理，本节的仿真场景中对于 ISTN-QR 算法的相关参数设置如下表 4-3 所示。

表 4-3 路由仿真相关参数设置

星间链路	ω_1	0.6
	ω_2	0.4
星地链路	ω_1	0.2
	ω_2	0.8
学习率α_1		0.8
折扣因子k		0.5
网络节点规模 N		88

数据包大小 Packet Size	1024 bits
信道速率 v	1Mbps
数据包生成速率 Packets Rate	20~300 pks/s
业务数据目的地址 Dest	random
仿真时间 Time	60 min

4.3.2 性能指标

为了分析本文提出的 ISTN-QR 路由算法在天地一体化网络中的性能表现，本文在仿真中收集和分析包递交率、端到端传输时延、时延抖动和路由开销四个指标，同时为了体现 ISTN-QR 算法的性能提升程度，本文选择以下几种路由算法作为性能参考对象：Dijkstra 最短路径路由、Flooding 洪泛路由和 FEQ-Routing 算法[32]，其中 Dijkstra 算法基于本文提出的 SDN-ISTN 架构实现集中式路由，Flooding 洪泛则基于广播的方式实现数据包的转发，FEQ-Routing 的实现方式则基于分布式实现。

1.包递交率（Packet Delivery Ratio，PDR）：表示应用层生成的业务数据包被成功转发到目的端的比例。另外随着网路流量负载的变化，网络对于数据包的承载能力也有所不同，这种能力也会反映在包递交率上。因此包递交率越高，表示路由算法的可靠性更高。

2.端到端传输时延（End-to-End Delay，EED）：本文 ISTN 网络场景中，数据包的端到端传输时延主要包含了节点处理、路由链路发现和计算、转发队列排队、链路传输等过程中产生的延迟。端到端时延性能反映了路由算法计算的转发路径的合理性，时延越小，说明路由算法性能越好。

3.时延抖动（Delay Jitter）：即业务数据流中相邻时刻的数据包传输时延之间的差值，时延抖动可以反映路由算法对于当前动态网络拓扑的适应性，时延抖动越小，表示路由算法性能越稳定。

4.路由开销（Routing Overhead）：定义路由开销为控制报文和冗余业务报文的转发量在仿真过程中占据总数据转发量的比重。

4.3.3 仿真结果与性能分析

为了展示 ISTN-QR 算法的路由计算和数据转发过程，我们通过追踪 Q 值表的更新状态来验证算法计算最短路径的过程。当仿真时间 T=28.00s 时，获取到了属于 Sat24 节点的 Q 值表状态如下图 4-3 所示。

```
from SatNode 24 to SatNode 11, route path : 13  2  1  11 SumHop is 5

time:28.000000, SatNode 24's Q_Value Table: Qvalue(Cur-Next):

 CurNode 1:  64.00(1-2)  100.00(1-11)  64.00(1-12)
 CurNode 2:  80.00(2-1)  51.20(2-3)  51.20(2-13)
 CurNode 3:  64.00(3-2)  40.96(3-4)  40.96(3-14)
 CurNode 4:  51.20(4-3)  32.77(4-5)  32.77(4-15)
 CurNode 5:  40.96(5-4)  32.77(5-6)  26.21(5-16)
 CurNode 6:  32.77(6-5)  40.96(6-7)  26.21(6-17)
 CurNode 7:  32.77(7-6)  51.20(7-8)
 CurNode 8:  40.96(8-7)  64.00(8-9)  40.96(8-19)
 CurNode 9:  51.20(9-8)  80.00(9-10)  51.20(9-20)
 CurNode 10:  64.00(10-9)  100.00(10-11)  64.00(10-21)
 CurNode 11:
 CurNode 12:  80.00(12-1)  51.20(12-13)  80.00(12-22)  51.20(12-23)
 CurNode 13:  64.00(13-2)  64.00(13-12)  40.96(13-14)  40.96(13-24)
 CurNode 14:  51.20(14-3)  51.20(14-13)  32.77(14-15)  32.77(14-25)
 CurNode 15:  40.96(15-4)  40.96(15-14)  26.21(15-16)  26.21(15-26)
 CurNode 16:  32.77(16-5)  32.77(16-15)  26.21(16-17)  20.97(16-27)
 CurNode 17:  32.77(17-6)  26.21(17-16)  32.77(17-18)  20.97(17-28)
 CurNode 18:  26.21(18-17)  40.96(18-19)  26.21(18-29)
 CurNode 19:  51.20(19-8)  32.77(19-18)  51.20(19-20)  32.77(19-30)
 CurNode 20:  64.00(20-9)  40.96(20-19)  64.00(20-21)  40.96(20-31)
 CurNode 21:  80.00(21-10)  51.20(21-20)  80.00(21-22)  51.20(21-32)
 CurNode 22:  100.00(22-11)  64.00(22-12)  64.00(22-21)  64.00(22-33)
 CurNode 23:  64.00(23-12)  40.96(23-24)  64.00(23-33)  40.96(23-34)
 CurNode 24:  51.20(24-13)  51.20(24-23)  32.77(24-25)  32.77(24-35)
 CurNode 25:  40.96(25-14)  40.96(25-24)  26.21(25-26)  26.21(25-36)
```

图 4-3 Q 值表状态示意图

由上图中打印信息可知，计算出的源节点 Sat24 到目的节点 Sat11 之间的数据转发路径为 Sat24-Sat13-Sat2-Sat1-Sat11，通过对 Q 值表中对应表项 Q 值的分析，可以看出算法选出的转发路径其对应 Q 值符合最优转发路径的准则，然后通过观察在仿真时间 T=938.093096s 时，序列号为 1965 的业务数据的转发过程如下图 4-4 所示。

```
from UsrNode 101 to UsrNode 102, route path : 23  34  45  55  54 , SumHop is 6
UsrNode 101: Old SerSat:12, after HandleOver, New SerSat:22
time:938.093096 : Src UsrNode:101, Dest UsrNode:102, Src SatNode:12, Dest SatNode:54, sendSeq:1965
time:938.101801, Current Sat 12 recv data from UsrNode 101,Dest UsrNode: 102
time:938.102458, Current Sat 12 recv data from UsrNode 101,Dest UsrNode: 102
time:938.102522, Current Sat 12 recv data from UsrNode 101,Dest UsrNode: 102
time:938.103259, Current Sat 12 recv data from UsrNode 101,Dest UsrNode: 102
Current sat:23, Src SatNode:12, Dest SatNode:54, CurrentHop:2
Current sat:34, Src SatNode:12, Dest SatNode:54, CurrentHop:3
Current sat:45, Src SatNode:12, Dest SatNode:54, CurrentHop:4
Current sat:55, Src SatNode:12, Dest SatNode:54, CurrentHop:5
Current sat:54, Src SatNode:12, Dest SatNode:54, CurrentHop:6
time:938.166202, Dest UsrNode 102 rcv Data from Src UsrNode 101
```

图 4-4 数据包转发过程示意图 a

上图显示了当前从源节点 Usr101 到目的节点 Usr102 的转发路径为 Usr101-Sat23-Sat34-Sat45-Sat55-Sat54-Usr102，根据随后的包转发流程中的打印信息可以获知，在 T=938.166202s 时，Seq=1965 号的数据包成功按照既定转发路径传输到目的节点 Usr102，我们通过获取网路拓扑信息，画出了该数据包的整个路由过程示意图如下图 4-5 所示。由于卫星网络中卫星节点分布均匀且具有规律性的移动轨迹等特点，分析图中的数据转发过程可以验证 ISTN-QR 路由算法计算得到的路径是具备最小跳数特征的最短路径。

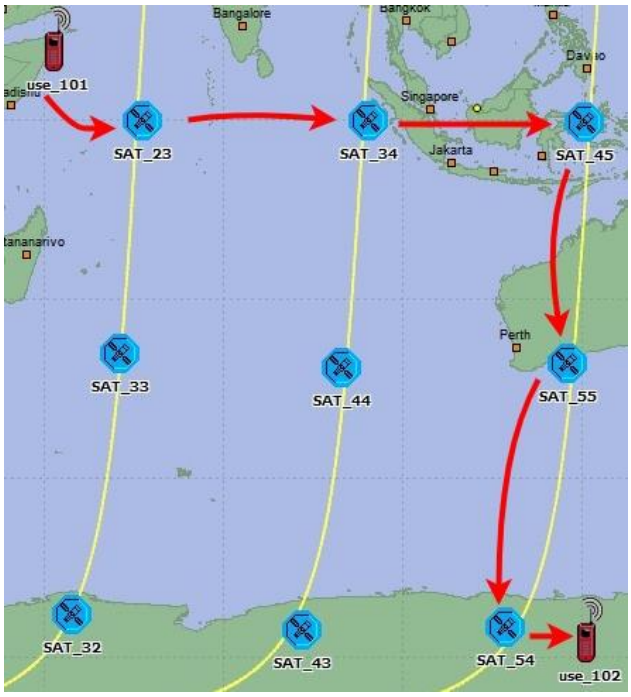

图 4-5 数据包转发过程示意图 b

另外为了验证本章所提出的 ISTN-QR 路由算法的收敛性，我们可以通过追踪该算法计算的转发路径是否是当前网络拓扑下最短路径来判断。在网络拓扑高速变化的天地一体化网络中，基于 SDN 控制器集中式计算端到端的转发路径，如果拓扑控制和链路发现过程中控制器不能将对应链路 Q 值收敛到最优值，则 ISTN-QR 算法则不能计算出端到端最短路径。已知 Flooding 路由总是具备最短端到端转发路径特征，因此通过比较 ISTN-QR 与 Flooding 路由两者的端到端路由跳数，可以验证 ISTN-QR 能够在拓扑动态变化的 ISTN 网络场景中实现收敛，具体仿真结果如下图 4-6 所示。

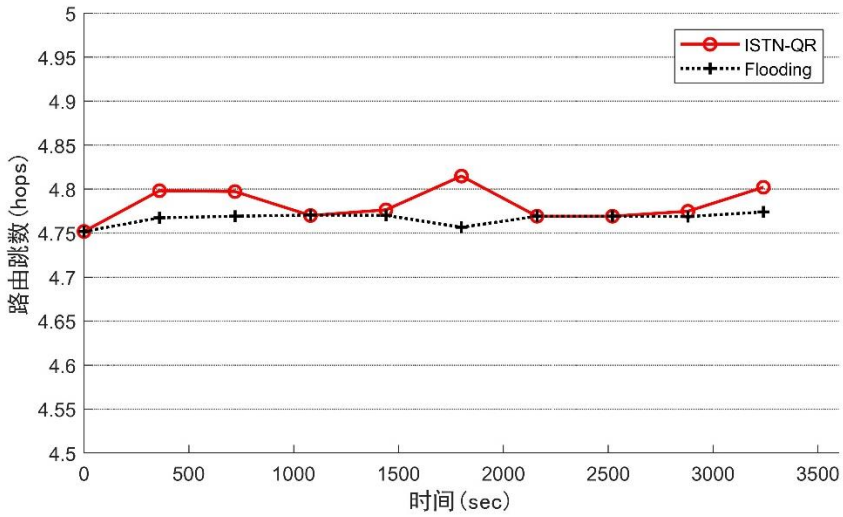

图 4-6 ISTN-QR 与 Flooding 路由转发跳数比较

通过调整仿真参数包生成速率(Packets Rate)以仿真路由算法在不同流量负载下的包递交率性能，并特别统计了 Packets Rate=50pks/s 和 Packet Rate=200pks/s 时业务数据包在收发两端的实际情况，两种分组产生速率分别表征低流量负载和高流量负载的网络环境，具体仿真结果如下图 4-7 和 4-8 所示。

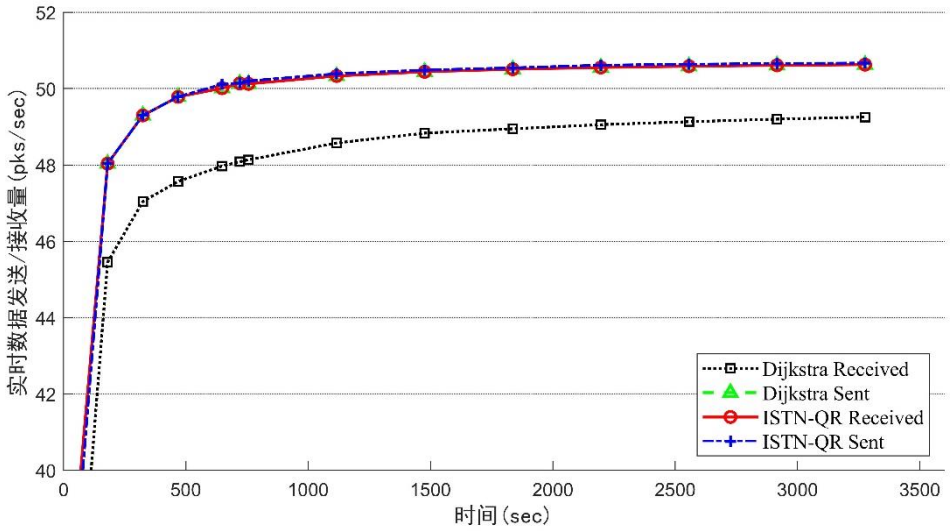

图 4-7 低负载时的端到端包传输量统计 a

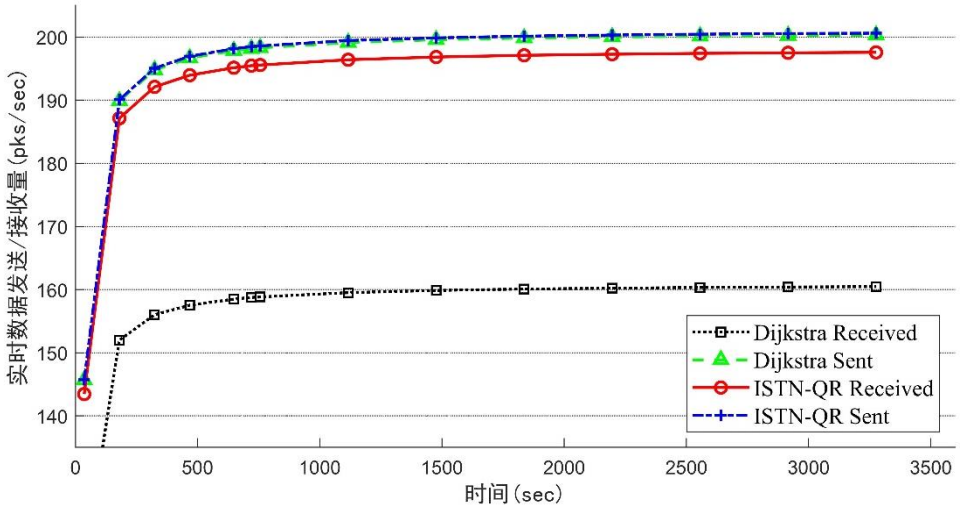

图 4-8 高负载时的端到端包传输量统计 b

上图 4-7 展示了包生成率为 50pks/s 时全网中数据收发两端的包收发情况，随着网络运行的逐渐稳定，ISTN-QR 算法和 Dijkstra 算法的包递交率均保持在 97%左右，ISTN-QR 算法性能略优于 Dijkstra 算法；图 4-8 展示了当包生成率为 200pks/s 时，网络处于高流量负载状态，受限于网络设备有限的计算存储能力，Dijkstra 算法的包递交率明显下降，大约维持在 80%左右，而 ITSN-QR 算法的包递交率性能下降较少，维持在 92%左右。为了观察路由算法的包递交率与网络流量负载之间

变化关系，接下来分别仿真并统计了对应包生成率条件下平均包递交率的变化关系，如下图 4-9 所示。

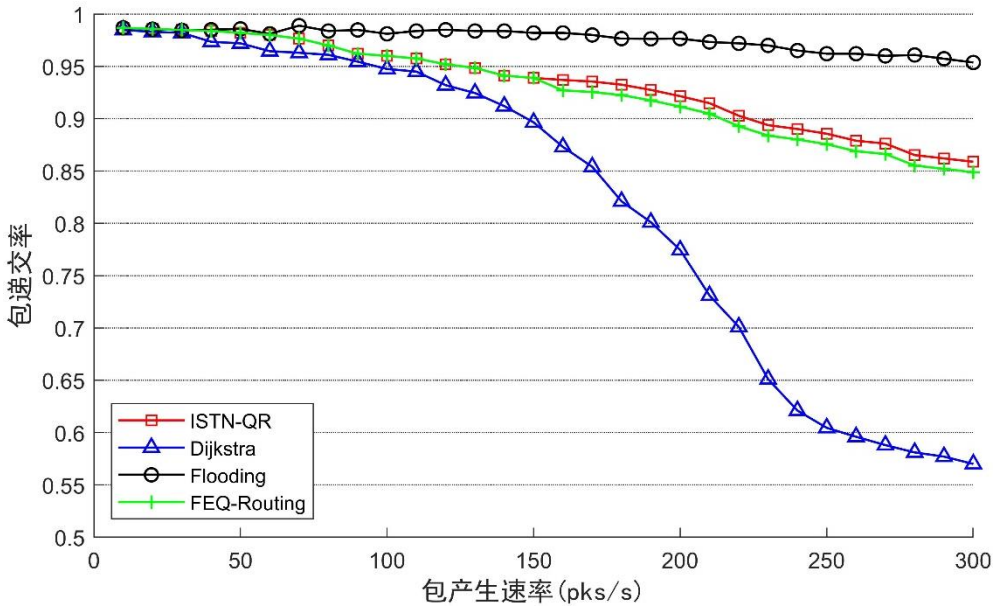

图 4-9 包递交率与网络流量负载的关系

从上图可以看出，在网络负载较低情况下，ISTN-QR 算法、Dijkstra、Flooding 和 FEQ-Routing 路由算法均能保持在较高的递交率水平上，其中 Flooding 洪泛采用广播扩散机制进行数据转发，能保证较稳定的包递交率。而随着流量负载的不断提升，一方面节点缓存队列逐渐拥塞，溢出的数据包被丢弃，另一方面在网络拥塞的情况下，控制器生成的转发路径在通过流表形式下发到交换机的过程中，无法实时适应网络拓扑的快速变化，导致路由失效。因此可以看出在网络负载较高时，Dijkstra 算法的递交率逐渐开始低于 ISTN-QR 路由算法，尤其是从 Packets Rate=150pks/s 开始，Dijkstra 算法的包递交率性能开始明显下降，而 ISTN-QR 算法虽然也有一定程度的下降，但性能损失较少。FEQ-Routing 采用的是与 ISTN-QR 类似的全回响路径探索方式，也可以有效适应动态网络拓扑变化并计算最优路径，因此递交率性能与 ISTN-QR 较为接近。另外 Flooding 洪泛路由的递交率随着网络负载提升并没有较明显的下降，但是洪泛广播机制会导致网络中出现大量冗余数据，大幅降低了网络资源的利用率。基于以上仿真结果可以分析出，ISTN-QR 路由算法整体上的包递交率性能较为稳定，相比于 Dijkstra 路由和 FEQ-Routing 路由均有一定程度上的提升。

对于路由算法，传输时延也一直是非常重要的性能衡量指标，结合前文中对于各路由算法的包递交率性能分析可以看出，当 Packets Rate=50pks/s 时，各路由算法的包递交率均处于较高水平，说明网络容量处于较高水平。因此接下来仿真统计

在包生成率 Packets Rate=50pks/s 时路由算法的端到端传输时延性能表现，具体仿真结果如下图 4-10 所示。

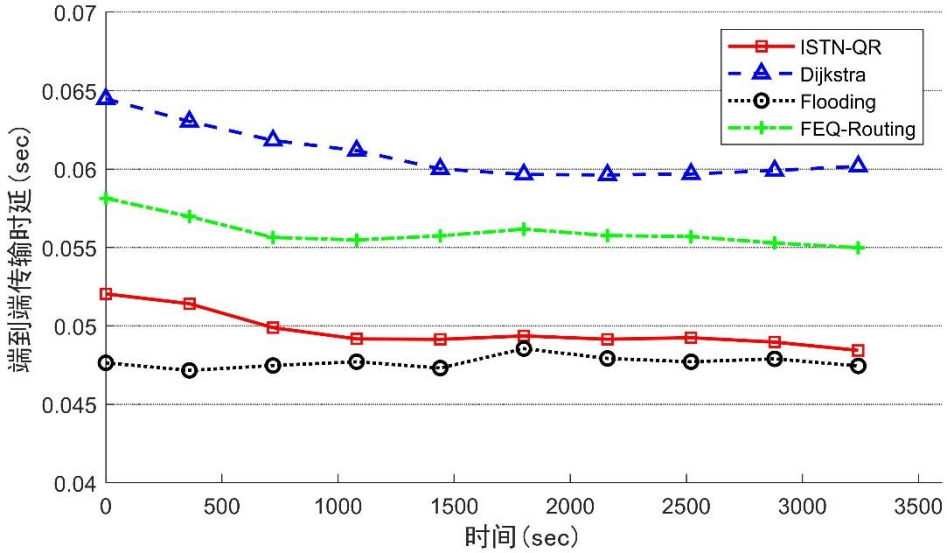

图 4-10 端到端传输时延性能比较

如上图可知，Flooding 采用洪泛机制可以保证数据包总是基于最优路径转发到目的终端，而且由于洪泛机制不需要较复杂的转发逻辑计算，所以包处理时延几乎可以忽略不计，因此 Flooding 路由具有最低的端到端时延性能。而 ISTN-QR 和 FEQ-Routing 分别采用集中式和分布式的实现方式，ISTN-QR 的最优路径收敛时间明显要低于 FEQ-Routing，而且 ISTN-QR 路由算法的整体传输时延性能要优于传统 Dijkstra 路由算法。由于在高动态变化的 ISTN 网络场景中，Dijkstra 算法计算的最短路径不一定是最优转发路径，从仿真结果来看，ISTN-QR 路由算法在整个仿真过程中得到端到端传输时延几乎稳定在 0.050s 以下，而 Dijkstra 算法随着仿真时间推移，逐渐稳定在 0.060s 左右，因此 ISTN-QR 路由算法相比 Dijkstra 算法在传输时延性能上有一定程度上的提升。根据本章前文中针对 ISTN-QR 路由算法的时间复杂度分析可知，ISTN-QR 算法的时间复杂度大概为$O((K+N)\cdot log_2 N)$，而对于 Dijkstra 算法，由于采用了局部贪心算法的方式逐次迭代从未选中节点集合中选择距离源节点最近的节点，本质上是一种 BFS 广度优先遍历算法，因此可以分析得知 Dijkstra 算法的时间复杂度为$O(N^2)$。因此可以看出，随着网络规模 N 的不断扩大，基于 Dijkstra 算法计算最短路径的时间成本将远远大于 ISTN-QR 算法，因此端到端传输时延性能要弱于 ISTN-QR 路由。结合以上结果分析，ISTN-QR 通过优化寻路策略和最短路径迭代计算方式，有效降低了端到端传输时延。

接下来分析两种路由算法的传输时延抖动性能，具体的仿真结果如下图 4-11 所示。

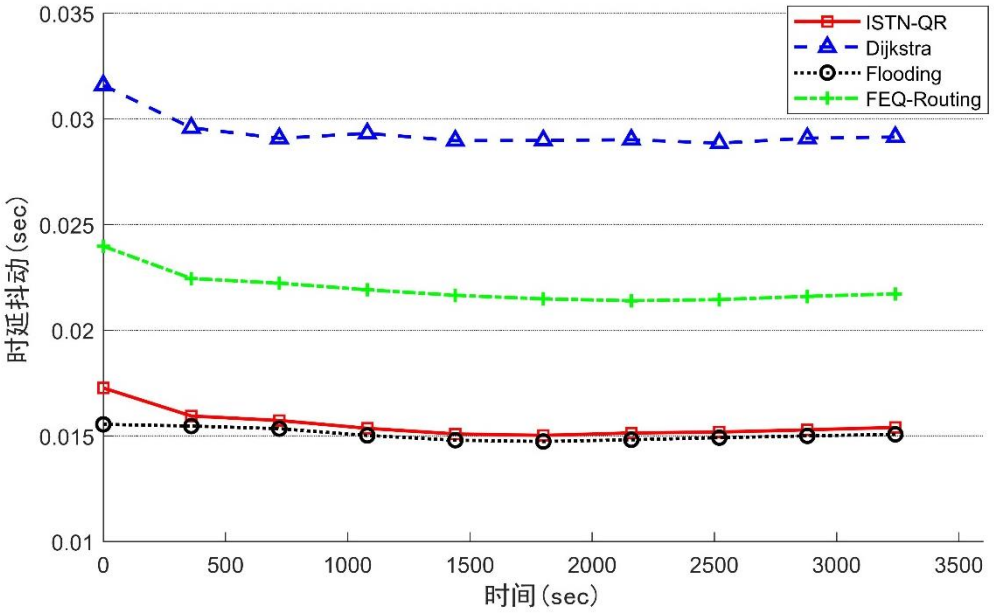

图 4-11 传输时延抖动性能比较

从上图中可以看出 ISTN-QR 算法的传输时延抖动在整个仿真过程中都相对低于 Dijkstra 算法和 FEQ-Routing 算法，这反映了基于 ISTN-QR 算法计算的转发路径相对于 Dijkstra 和 FEQ-Routing 的稳定性更好。仿真结果反映出时延抖动性能上的这种差异情况，其本质原因是 ISTN-QR 算法能够更好地适应天地一体化网络中网络拓扑的动态变化特性，基于 Q-Learning 算法的 ISTN-QR 路由算法在计算最短路径的时候，会综合当前最新网络信息和以往学习经验来更新 Q 值表，以实现对动态网络的适应。对于 FEQ-Routing，由于转发节点采用分布式实现方式，对于全局动态拓扑变化的信息获取不够及时，尤其是当卫星星间链路切换时，节点单次路径探索不能保证总是发现最优转发路径，因此业务数据传输时延抖动相较于 ISTN-QR 较大。而 Dijkstra 算法由于每次都仅仅基于当前网络邻接图计算出最短路径，在天地一体化网络这种高动态环境中，相比 ISTN-QR 算法没有体现出较好的适应性，因此计算出的转发路径其稳定性不如 ISTN-QR 算法。

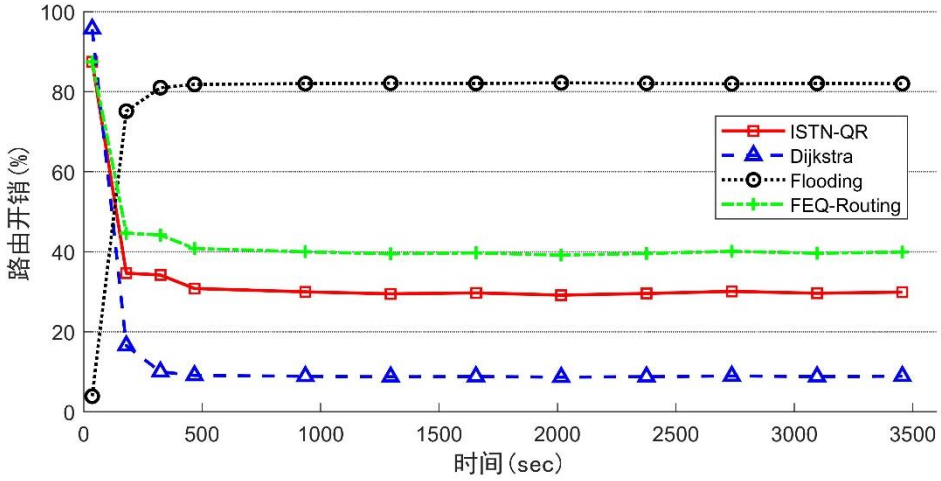

图 4-12 路由开销性能比较

最后分析仿真过程中路由开销性能情况，仿真结果如上图 4-12 所示。可以看出，Flooding 洪泛路由受限于广播机制的原理，导致在网络中需要转发大量的冗余数据，导致整体路由开销十分高，所以在天地一体化网络这种资源受限、复杂异构场景中不适用。FEQ-Routing 中全回响的路径探索方式导致，路由发现过程中会转发一部分冗余的控制报文，而对于以集中式实现的 ISTN-QR 算法，相邻节点之间的路径探索只需要一次控制报文转发流程即可完成，避免了冗余控制报文转发造成的额外开销。

另外由于 ISTN-QR 算法需要实时反馈网络拓扑变化信息给控制器，控制器执行路由算法需要利用当前的网络拓扑信息和一部分旧的卫星轨迹信息，所以 ISTN-QR 算法的路由更新会比 Dijkstra 算法更频繁，因此路由开销会比 Dijkstra 算法更高一些。分析结果可知，ISTN-QR 路由算法通过更频繁的路由更新方式来保证转发流表项的实时可用性，而 Dijkstra 的路由算法通过周期性地基于网络信息计算最短路径，并且通过控制报文逐跳更新对应交换机中的流表项，降低了路由开销。

结合以上仿真结果和分析可以看出，ISTN-QR 相比于基于分布式实现的 FEQ-Routing，在包递交率、传输时延、时延抖动和路由开销等指标上均有一定程度的性能提升。而 Flooding 洪泛路由虽然具有原理简单、路由可靠等特点，但是较高的路由开销使得 Flooding 不适用于天地一体化网络场景。另外与一样是基于 SDN 集中式控制实现的 Dijkstra 路由算法相比，ISTN-QR 算法通过损失部分路由开销性能，换取了更好的端到端传输时延性能、时延抖动和包递交率性能，ISTN-QR 算法能够更好地适应多种网络流量负载情况下的数据传输，相比于传统 Dijkstra 算法，ISTN-QR 算法具备更可靠的包传输效率和更稳定的传输时延性能。

4.4 本章小结

本章的主要内容是基于 Q-Learning 的算法模型对天地一体化网络进行 MDP 模型建模，然后对 ISTN-QR 算法中的迭代更新方式和选路策略设计进行详细阐述，并分析了 ISTN-QR 算法流程。

最后基于本文第三章所提出的天地一体化网络仿真平台对 ISTN-QR 算法和 Dijkstra 路由、Flooding 洪泛和 FEQ-Routing 算法进行了性能对比，验证了 ISTN-QR 路由算法在天地一体化网络场景中的适用性。

第五章 总结与展望

5.1 本文工作总结

本文以天地一体化网络路由协议为研究核心，主要研究了天地一体化网络体系架构，探讨了现有协议架构的优缺点，并提出了基于 SDN 的一体化网络架构，然后基于机器学习算法提出了一种适用于上述场景的一体化路由算法。本文具体完成的工作如下所述：

（1）阅读大量现有天地一体化网络架构相关文献，分析并研究了目前有关协议体系的研究现状。详细阐述了 CCSDS、DTN 和 SDN 等协议体系的优缺点和适用场景，最后采用 SDN 架构搭建天地一体化网络。

（2）分析传统 SDN 的设计思想，基于此设计了基于 SDN 的天地一体化网路架构，并对处于控制层面的 SDN 控制器的功能划分和部署方案进行了详细设计。

（3）使用 OPNET 和 STK 仿真软件，设计并实现了自定义 SDN 网络协议、相关节点模型和进程模型，然后搭建了基于 SDN 的天地一体化网络仿真平台，实现了天地一体化网络的基本通信功能。

（4）借鉴了目前传统卫星网络路由协议和基于机器学习的路由算法，提出了基于 Q-Learning 的适用于上述场景的一体化协同路由算法 ISTN-QR，并在前文中搭建的 SDN-ISTN 仿真平台上进行了算法实现，通过仿真对路由算法性能进行了测试对比。结果显示该算法相比于 Dijkstra、Flooding 和 FEQ-Routing 路由算法，在可靠性和稳定性等性能方面有一定程度提升。

5.2 研究展望

近年来天地一体化网络逐渐称为学术界的热门话题，本文仅仅只是从基础的网络架构和路由算法两个方面进行了初步研究。随着 SDN 技术的不断成熟，基于 SDN 架构的天地一体化网络还有很多改进的地方，随着未来网络规模的不断扩大，需要考虑控制层面中东西向的扩展性，其次还要考虑进一步标准化控制面和数据面之间南向接口的标准化问题，提高 SDN 网络的运行效率。另外本文提出的路由算法仅仅考虑了 LEO 卫星网络和地面网络融合场景，下一步可以考虑融合 MEO 和 GEO 卫星网络，增加地面异构网络的多样性，从而进一步提高路由算法的性能。

致 谢

珍贵的研究生学习生涯即将告一段落，回顾刚刚入学的时光仿如隔日。三年的研究生学习阶段让我收获了许多东西，也让我对自己未来人生道路有了更清晰的规划，这一切都离不开在这个阶段中我遇到的各位良师益友们。

首先要感谢我的导师郭伟教授，感谢郭老师为我提供了宝贵的科研条件和资源，郭老师在学术研究上认真严谨，在论文指导工作上认真负责，总是不厌其烦地帮我指出论文研究方面的问题，同时也感谢郭老师在生活中对我的关心。其次要感谢赵聪老师，赵老师在论文选题与研究工作的开展过程中都为我提供非常宝贵的意见，同时在撰写论文的过程中帮我指出了很多问题，在赵老师的指导下，我的论文工作才能顺利完成。另外还要感谢智能无线自组织网络团队的所有老师，他们在学习和生活上给予了很多的关心和支持。

感谢教研室的所有同学，在这三年里大家相互监督和支持，在科研和生活上给予了我非常多的帮助，愿大家未来前途一片光明。

感谢我的父母和妹妹在生活中对我无微不至的关怀，父母的养育之恩，我将用余生予以回报。

最后向所有评审老师表达最诚挚的谢意！

参考文献

[1] 沈学民，承楠，周海波.空天地一体化网络技术：探索与展望[J].物联网学报,2020,4(03):3-19

[2] 沈荣骏.我国天地一体化航天互联网构想[J].中国工程科学,2006(10):19-30

[3] 张乃通，赵康健，刘功亮.对建设我国“天地一体化信息网络”的思考[J].中国电子科学研究院学报, 2015, 10(03):223-230

[4] 闵士权.我国天地一体化综合信息网络构想[J].卫星应用,2016(01):27-37

[5] Iridium Commun.Inc. Iridium Global Network[EB/OL]. https://www.iridium.com, 2018

[6] OneWeb 组网发射恢复-联盟号送 36 颗卫星上天[J].卫星与网络, 2020(12):76

[7] 刘帅军，徐帆江，刘立祥. Starlink 第二代系统介绍[J].卫星与网络,2020(12):62-65

[8] K. Fall, S. Farrell. DTN:an architectural retrospective[J]. IEEE Journal on Selected Areas in Communications, 2008, 26(5):828-836

[9] 苏冰.空天地一体化网络路由协议的研究[D].成都：电子科技大学，2016

[10] 刘立祥.天地一体化网络[M].北京：科学出版社，2015

[11] C. Caini, R. Firrincieli. Application of Contact Graph Routing to LEO satellite DTN communications[J]. IEEE, 2012

[12] 杨虹.面向复杂空天地一体化网络的 SDN 控制器的研究[D].北京：北京交通大学，2018

[13] 朱思宇.基于 SDN 的天基网路由算法研究[D].哈尔滨：哈尔滨工业大学，2017

[14] T. Li, H. Zhou, H. Luo, et al. SERvICE: A software defined framework for integrated space-terrestrial satellite communication[J]. IEEE Transactions on Mobile Computing, 2017, 17(3): 703-716

[15] Y. Bi, G. Han, S. Xu, et al. Software defined space-terrestrial integrated networks: architecture, challenges, and solutions[J]. IEEE Network, 2019, 33(1): 22-28

[16] 李健.天地一体化信息网络中的星间路由与星地协同传输机制研究[D].合肥：中国科学技术大学，2020

[17] 李泽旺.软件定义一体化网络仿真平台研究与实现[D].成都：电子科技大学，2015

[18] A. Saika, R. E. Kouch, A. Najid, et al. Implementation and performances of the protocols of routing AODV and OLSR in the Ad hoc networks. IEEE, 2010

[19] E. Ekici, I. Akyildiz, M. D. Bender. A distributed routing algorithm for datagram traffic in LEO satelitte networks[J]. IEEE/ACM Transactions on Networking (TON), 2001

[20] M. Werner, C. Delucchi. ATM-based routing in LEO/MEO satellite networks with intersatellite links[J]. Selected Areas in Communications IEEE Journal on, 1997, 15(1):69-82

[21] V. V. Gounder, R. Prakash, H. Abu-Amara. Routing in LEO-based satellite networks[C]. Wireless Communications and Systems, 1999 Emerging Technologies Symposium. IEEE, 1999

[22] Hong, Seong, Chang, et al.FSA-based link assignment and routing in low-earth orbit satellite networks[J]. Vehicular Technology IEEE Transactions on, 1998

[23] M. Werner. A Dynamic Routing Concept for ATM-Based Satellite Personal Communication Networks[J]. IEEE Journal on Selected Areas in Communications, 1997, 15(8):1636-1648

[24] 肖阳.低轨卫星系统的位置管理及路由算法研究[D]. 成都：电子科技大学，2009

[25] K. Tsai, R. P. Ma. DARTING: a cost-effective routing alternative for large space-based dynamic-topology networks[C]. Proceedings of MILCOM '95. IEEE, 2002

[26] B. Jianjun, L. Xicheng, L. Zexin, et al. Compact explicit multi-path routing for LEO satellite networks[C]. Workshop on High Performance Switching & Routing. IEEE, 2005

[27] E. Papapetrou, S. Karapantazis, F. N. Pavlidou. Distributed on-demand routing for LEO satellite systems[J]. Computer Networks, 2007, 51(15):4356-4376

[28] Y. Sun, M. Peng, Y. Zhou, et al. Application of Machine Learning in Wireless Networks: Key Techniques and Open Issues[J]. IEEE Communications Surveys & Tutorials, 2019, P(99):1-1

[29] 周志华.机器学习 := Machine learning[M].北京：清华大学出版社，2016

[30] Zhenyu Na, P. Zheng, L. Xin, et al. Distributed Routing Strategy Based on Machine Learning for LEO Satellite Network[J]. Wireless Communications & Mobile Computing, 2018, 2018:1-10

[31] J. A. Boyan. A Distributed Reinforcement Learning Scheme for Network Routing[C]. International Workshop on Applications of Neural Networks to Telecommunications. 1993

[32] J. A. Boyan, M. L. Littman. Packet Routing in Dynamically Changing Networks: A Reinforcement Learning Approach[J]. Advances in neural information processing systems, 1993, 6

[33] Choi, Yeung. Predictive Q-routing: memory-based reinforcement learning approach to adaptive traffic control[C]. Proceedings of the 8th International Conference on Neural Information Processing Systems.MIT Press, 1995

[34] Kumar. Dual reinforcement Q-routing:an on-line adaptive routing algorithm[C]. Proceedings of the Artificial Neural Networks in Engineering Conference, 1997

[35] T. Rossi, M. D. Sanctis, E. Cianca, et al. Future space-based communications infrastructures based on High Throughput Satellites and Software Defined Networking[C]. IEEE International Symposium on Systems Engineering. IEEE, 2015

[36] Y. Miao, Z. Cheng, Z. Cui. Software defined integrated satellite-terrestrial network: A survey[C]. Proc.Int.Conf. Space Inf. Netw.Singapore: Springer, 2016, pp.16–25

[37] J. Bao, B. Zhao, W. Yu, et al. OpenSAN: A Software-defined Satellite Network Architecture[J].

Computer Communication Review: A Quarterly Publication of the Special Interest Group on Data Communication, 2014

[38] F. Hu, Q. Hao, K. Bao. A Survey on Software-Defined Network and OpenFlow: From Concept to Implementation[J]. Communications Surveys & Tutorials IEEE, 2014, 16(4):2181-2206

[39] 靳登匀.面向 One Web 卫星网络的 SDN 控制器设计与实现[D].哈尔滨：哈尔滨工业大学, 2019

[40] Q. Guo, R. Gu, T. Dong, et al.SDN-based End-to-end Fragment-aware Routing for Elastic Data Flows in LEO Satellite-Terrestrial Network[J]. IEEE Access, 2018:1-1

[41] B. Yang, Y. Wu, X. Chu, et al. Seamless Handover in Software-Defined Satellite Networking[J]. IEEE Communications Letters, 2016:1768-1771

[42] 张姗姗.基于软件定义卫星网络的低轨卫星切换策略研究[D].北京：北京邮电大学,2019

[43] 李睿，曾德贤.STK 用于网络环境下仿真的实现方法[J].航天控制, 2005(03):64-68

[44] 龙华. OPNET Modeler 与计算机网络仿真[M].西安：西安电子科技大学出版社, 2006

[45] 晏坚.低轨卫星星座网络 IP 路由技术研究[D].北京：清华大学, 2010

[46] Z. Mammeri. Reinforcement Learning Based Routing in Networks: Review and Classification of Approaches[J]. IEEE Access, vol.7, pp. 55916-55950, 2019

[47] M. L. Littman. Markov games as a framework for multi-agent reinforcement learning[M]. Morgan Kauffman Publishers, Inc. 1994

[48] C. J. C. H. Watkins. Learning from delayed rewards[D]. Lodon: University of Cambridge,1989

[49] C. J. C. H. Watkins, P. Dayan. Q-learning[J]. Machine Learning. 1992, 8(3-4): 279-292

[50] C. Wang, L. Zhang, Z. Li, et al. SDCoR: Software Defined Cognitive Routing for Internet of Vehicles[J]. IEEE Internet of Things Journal, 2018:1-1