低轨道(LEO)宽带卫星网络最短路由优化算法*

王凯东 [1,2**] 易克初 [2] 田 斌 [2]

(1. 西安电子科技大学计算机学院; 2. 综合业务网国家重点实验室, 西安 710071)

摘要 宽带卫星网络不仅能提供全球范围内的覆盖而且能提供广泛多样的不同种类和服务质量的数据通信业务. 由低轨道(LEO)卫星簇组成的星座网络由于具有较短的端到端的时延以及更宽的带宽的优点, 越来越为人们所重视. 但针对 LEO 卫星星座网络而特别设计的简单有效的路由算法却是急待解决的问题. 从 LEO 卫星星座网络抽象出一种新颖的三维球面网格拓扑结构, 即蜘蛛型拓扑网络 (SWTN), 并提出了基于 SWTN 的分布式分组路由算法. 本算法具有非常低的星上计算复杂度以及不需要星上路由表的特点, 非常适合星上处理. 通过仿真可以得到本路由算法的性能.

关键词 分组路由 低轨道卫星星座 宽带卫星网络 蜘蛛型网络拓扑 星上交换

卫星通信已经在我们生活的地球上获得了广泛的应用. 这是由于卫星能覆盖非常广泛的地理区域(包括陆地、空中、海洋, 以及条件恶劣的区域像高山、河流、森林、沙漠以及南北极区域等等), 并且能提供不同种类和服务质量的宽带业务, 因此卫星网络可以为政府、军队、企业和我们每一个人提供广泛多样的不同种类的数据通信业务. 无论是发达国家还是发展中国家, 卫星通信网络都是非常有吸引力的, 是能改变整个国家和社会的信息技术之一. 在下一代将因特网(NGI)的发展中, 卫星网络(包括LEO和GEO卫星)都将扮演非常重要的角色. 这是因为[1]: (a)卫星可以覆盖地球大部分的区域, 包括城市、乡村以及偏远或者难以到达的地区; (b)卫星通信可以具有灵活的按需分配带宽的能力; (c)为了达到最大的

2004-04-08 收稿, 2005-04-18 收修改稿
* 国家自然科学基金(批准号: 60172029), 空间微波技术国防科技重点实验室基金(514730201)
** E-mail: wkd@mail.xidian.edu.cn

资源利用率目的, 对于不可预测带宽的业务需求, 卫星可以作为备用信道; (d)卫星可以使新用户方便快捷地接入到因特网中, 将网络的扩充变得十分容易; (e)卫星可以成为下一代因特网的安全阀. 当地面网络出现拥塞或者某段光纤链路出现故障等问题时, 通过接入卫星信道可以很容易得到解决. (f)对于某些新的应用领域, 例如数字地球、远程教育、远程医疗、远程娱乐等等, 通过卫星都可以变成现实.

但是要将宽带卫星通信网络系统和基于 IP 分组的因特网非常好地融合在一起, 即能在宽带卫星网络上路由 IP 分组包, 还需要解决一些技术难题. 由于 LEO 卫星簇围绕地球高速公转, 因此 LEO 卫星之间相互链接的拓扑形状是随时间不断动态变化的, 并不是静止不动的, 需要特别针对 LEO 卫星星座网络, 设计出一种简单的、高效的分组路由算法. 通过此路由算法, 使分组能从发送方开始, 一跳一跳地穿过 LEO 卫星星座, 然后顺利到达接收方, 这是解决 LEO 宽带卫星网和因特网能很好融合在一起的关键技术之一.

近年来, 已有文献[2~15]研究有关LEO卫星网络路由算法, 但大多是基于面向连接的网络结构[2~7], 例如基于ATM的星上交换网络. 这些文献中的路由算法大多是在路由建立初始阶段计算的. 即路由的计算是在地面交换中心集中按照路由请求消息计算得到相应的路由表, 然后配置到每一颗卫星中去, 星上的交换机仅仅按照路由表来转发分组. 由于卫星公转和地球自转引起的网络拓扑的变化, 并不能保证其初始最优路由以后也是最优的. 为了解决这个问题文献[8]提出了所谓的“路径切换(path handover)”的方法. 但这些面向连接的路由算法通过使用路径切换来得到最短路由的方法却因为需要大量的路由表信息在卫星之间更新数据而变得不太可行.

目前, 因特网使用已经是大众所必需的工具之一, 并且下一代因特网也在紧锣密鼓的研制之中, 因此不管是公共商业领域还是军事领域都要求将IP技术和卫星网络紧密结合, 即在卫星上能够进行路由面向无连接的IP分组. 由于LEO卫星簇围绕地球公转, 而且地球本身也在自转, 所以其卫星网络拓扑结构是随时间变化的, 因此必须针对这种随时间变化的特殊的网络拓扑研究设计出相应的路由算法来. 面向无连接的分组路由算法问题又分为集中式和分布式算法两类. 对于集中式算法方面, 文献[9]中的Darting 算法用于解决卫星网络中非常繁重的拓扑消息更新问题, 但是与扩展的Bellman-Ford算法[10](这是一种距离向量协议修订版本)比较, Darting算法有相同的端到端的时延, 却带来了很多倍的时间开销代价. 前面提过, 集中式计算路由表然后更新到卫星中去的算法会带来非常大的时间开销和带宽开销, 另外也不可能在同一时刻同时将全部LEO卫星的路由表及时更新, 因此在星上路由表的更新切换过程中分组有时会找不到其最短路由. 文献[12]提出了基于地理位置的面向无连接的分布式分组路由算法, 但由它得到的路

由并不是全局最优路由，而是局部最优路由的组合. 文献[13]论述了用一个双向扭曲曼哈顿街道网(Manhattan street network)[14]拓扑结构来模拟卫星星座图，由此想得到分组最短路由，但因为卫星网络拓扑结构是随时间变化的复杂三维立体球面网格拓扑结构，而曼哈顿街道网拓扑仅仅是一个矩形的二维网格拓扑结构，因此用简单的曼哈顿街道网不能很直观地模拟出卫星星座这种复杂的三维立体球面网格结构的特点；而文献[15]从另外一个角度比较详细地论述了一种面向无连接的分布式的分组路由算法，对曼哈顿街道网的拓扑结构做了加权处理，即在链路上通过加权的方法来区分不同纬度的卫星链路，这种给曼哈顿街道网路径加权的方法来模拟卫星星座虽取得了比较好的效果，但其算法还是比较繁琐，其星上路由计算和处理的负担还是有些过重.

在本文，我们在LEO卫星星座网络拓扑中提出一种新颖的网络拓扑模型SWTN (spiderweb topological network)，并相应给出了基于SWTN的面向无连接的分组路由优化算法. 本路由算法的路由选择是基于源地址和目的地址之间路径的最小传播时延为准则. 本算法是分布式的算法，即下一跳路由的选择都是基于当前时刻(时间片)LEO卫星中的分组来决定的，与其他LEO卫星和分组无关. 本算法并没有因为卫星星座网络拓扑随时间的变化而有LEO卫星之间路由表信息交换的开销. 本文提出的算法也考虑了拥塞避免和链路毁坏的情况下的次优路径. 由于本算法是基于IP数据报分组的，因此并没有像在面向连接的网络中路由算法中有已建立链路连接路由表信息的切换开销. 本路由算法的实现简单高效，仅仅通过一次比特位的减法运算即可得到下一跳的路由，另外星上没有保存有大量路由表信息. 和其他相关的文献[12,13,15]相比，本算法极大地减轻了星上处理的运算开销和宝贵的星上存储单元，也便于硬件高速实现.

1 蜘蛛型拓扑网络(SWTN)

由极轨道(或近极轨道)LEO卫星组成的星座覆盖了整个地球. 此星座有N个轨道面，每个轨道面由M个LEO卫星组成，卫星之间通过星际链路(ISL)互联. 星际链路有两种形式[16]：在每个卫星轨道面内有面内星际链路(intraplane ISL)，在相邻轨道面间有面间星际链路(interplane ISL). 在每一轨道面内，每颗卫星都与同轨道的在其前后的相邻卫星分别建立链路. 由于轨道面内相邻卫星几乎没有相对运动，因此面内星际链路的距离以及卫星天线的指向基本保持恒定. 而相邻轨道面间的星际链路，由于卫星之间相对位置的变化，其距离、方位角和仰角都是时变的，因此面间星际链路的距离是时变的. 此外，在面间星际链路中还有一类特殊链路需要特别考虑. 上升轨道面和下降轨道面之间称为缝隙(seam)，此时LEO卫星运动轨迹相反，因此称为缝隙星际链路(cross_seam ISL). 相对与其他星际链路(面内和面间星际链路)，缝隙星际链路具有距离和方位角度实时高速变化

和链路周期动态切换的特点. 这样有$N*M$个LEO卫星构成一个星座. 我们用$M*N/N/0°$来表示此星座[17].

在 $M*N/N/0°$组成的星座网络中, 处于极点区域的卫星间隔密度很大, 卫星之间相对运动速度很快, 特别是越过极点前后的面间星际链路的天线方向会相反, 由于技术上的一些困难以及实际地理环境并没有太多人在极点区域活动, 不要求做面间星际链路(interplane and cross_seam ISL)路由, 只要求做面内星际链路(intraplane ISL)路由.

例如, 一个大的LEO卫星星座网络如图 1 所示, 其中N=12 个轨道面, 每轨道M=24 颗LEO卫星, 共 288 颗LEO卫星; 另一个小的LEO卫星星座网络如图 2 所示, 其中N=6 个轨道面, 每轨道M=12 颗LEO卫星, 共 72 颗LEO卫星. 由于所有的LEO卫星都是极轨道(或准极轨道)卫星, 并且有上升面和下降面的划分, 此种星座称为walker star constellation或者称为π-constellation[16], 如图 3 所示.

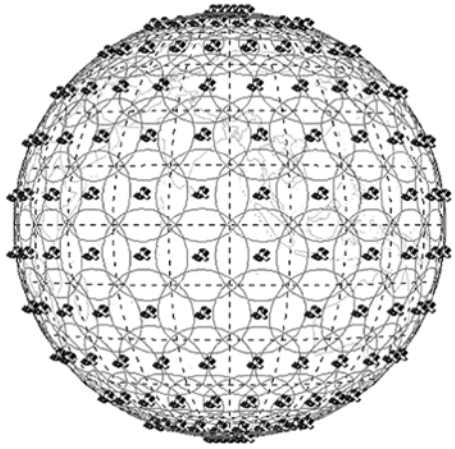

图 1　由 288 颗极轨道 LEO 卫星组成的大星座网络(N=12, M=24)

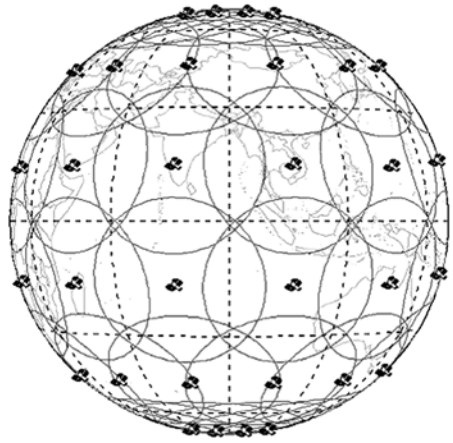

图 2　由 72 颗极轨道 LEO 卫星组成的小星座网络(N=6, M=12)

已有一些文献[12,13,15]使用了不同网络拓扑结构来进行最短路由的寻找, 但大多不简便实用. 本文提出了一种新型的网络拓扑结构, 称为蜘蛛型拓扑网络(spiderweb topological network, 简称SWTN), 使用SWTN 这种拓扑可以很容易并且很简单地找出最短路由来. 其拓扑如图 4 所示. 我们将图 1 的立体星座结构压扁拓扑成 2 张类似蜘蛛网形状的平面网络结构(分别从N极和S极看). 此蜘蛛型拓扑网络中每一个节点有一颗卫星(太空路由器或称交换机). 我们可以想象图 1 的立体星座是某种超级蜘蛛织成的球面蜘蛛网, 为了便于分析和理解, 我们将此球面蜘蛛网压扁拓扑成两张背靠背叠加在一起的普通蜘蛛所织成的平面蜘蛛网络. 这样超级蜘蛛在立体球面蜘蛛网络上能尽快地并且尽可能简单地寻找到已

捕获的猎物(目的地址), 即怎样才能找到一条在球面蜘蛛网上已捕获猎物(目的地)的最短路由的问题, 就变成了普通蜘蛛在我们日常所见的平面蜘蛛网络上寻找最短路由的问题. 此时的平面蜘蛛网是由 2 张普通的蜘蛛网背靠背的叠加而成, 因此蜘蛛要从一个平面到另一个平面去吃猎物必须通过其所在平面的蜘蛛网的边沿(赤道). 本文将源(source)称为蜘蛛, 而目的地(destination)称为猎物. 这里我们提出基于蜘蛛型拓扑网络的分组最短路由准则, 这是本文后面的算法实现所要遵循的准则.

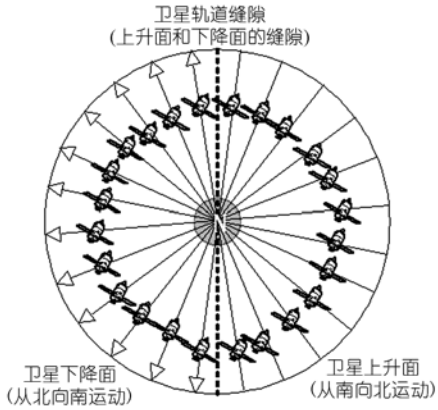

图 3 π 型星座北极透视图

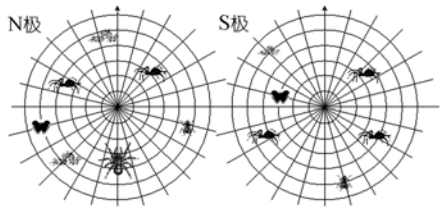

图 4 蜘蛛型拓扑网络(SWTN)

请注意, SWTN本质是一张非常规则的三维球面网格. 将此三维球面网格压扁成两张背靠背的平面SWTN的目的是为了便于分析理解. 我们可以给SWTN中的节点进行地址编码, 因此可以在具有寻址能力的SWTN中进行相关的路由计算. 此外, 尽管真实的极轨道LEO卫星星座中各卫星轨道并不是真正地穿过极点的圆形轨道, 而是卫星轨道都朝着某个方向略微有点偏移而且是略微有点椭圆型的轨道(以避免各颗LEO卫星在极点附近相互碰撞), 但这并不影响此星座网络的抽象. 如果一个LEO卫星星座可以用$M^*N/N/0°$[17]来表示, 那么此星座就是一个三维球面的网格, 即可以抽象成一个很规则的蜘蛛型拓扑网络(SWTN)模型. 本文所提出的基于SWTN的分组路由算法必定适用于此π-constellation (或称 walker star constellation)[16]. 而由非极轨道LEO卫星构建的星座称为rosette constellation (或称walker delta constellation)[16]. 本文所提出的基于SWTN的分组路由算法不适用于非极轨道LEO星座网络.

2 SWTN 的寻址方式和编码方案

2.1 LEO 卫星星座的一些必要的基本概念

1) 地球自转周期. 众所周知, 我们居住的地球是太阳系中的一颗行星. 地球

本身具有自西向东自转的周期. 地球自转一圈的周期为 23 小时 56 分 4.09 秒[18].

2) LEO卫星轨道和星座. 本文涉及的LEO卫星轨道都是越过南北极点区域的圆形极轨道. 有关卫星的轨道和周期可由Kepler第 1, 2, 3 定律确定[17,18]. 一个极轨道卫星星座网络由N个轨道面(每个轨道面由M个极轨道LEO卫星均匀分布在地球上空)组成. 例如, 图 1 所示, N=12, M=24, 取 15°作为分割单位; 图 2 所示, N=6, M=12, 取 30°作为分割单位. 也可以取经度和纬度分割不相同的单位, 例如经度取 30°, 纬度取 15°.

3) 南北极面. 以赤道为中心将地球分为南北 2 个极面. 图 4 是将图 1 拓扑成两张平面的背靠背蜘蛛型拓扑网络. 图 5 为从某个极点看上去的平面拓扑图, 这里我们将北极作为观测点, 本文后续的描述也是在北极透视. 注意图 5 是 2 张平面 SWTN 背靠背的叠加在一起的拓扑图型. 类似地, 图 6 中的 SWTN 是从图 2 的星座网络抽象出来的.

4) 星际链路和邻居卫星. 由于 LEO 星座是通过星际链路互联成网络, 因此每一颗卫星都有相应的邻居卫星. 定义: 具有星际链路的 2 颗 LEO 卫星互称邻居. 星际链路分为简单结构和复杂结构 2 类, 如图 5~8 所示. 对于简单结构, 每颗 LEO 卫星有 4 颗邻居卫星(包含 cross-seam ISL): 轨道面内有 2 个邻居, 一个为内

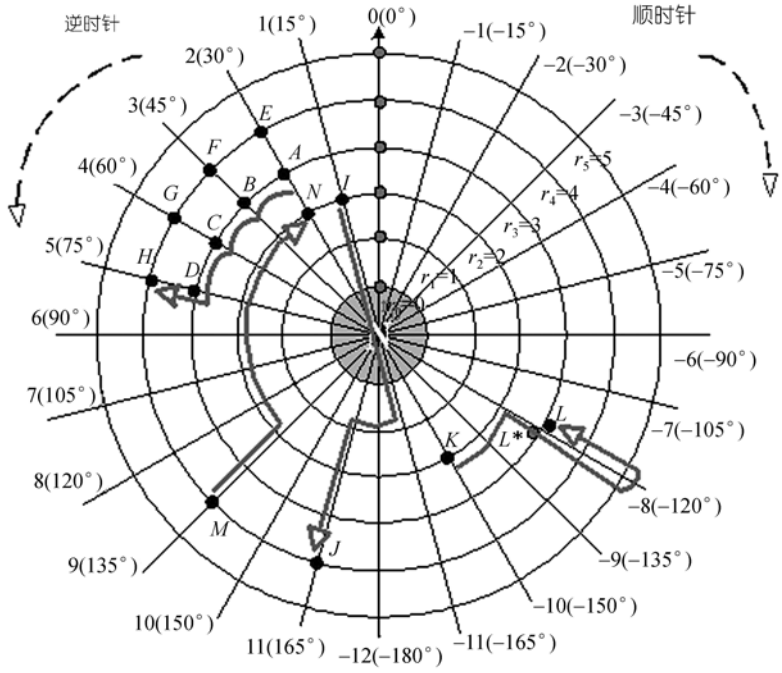

图 5　SWTN 的寻址方式和编码方案(N=12, M=24)

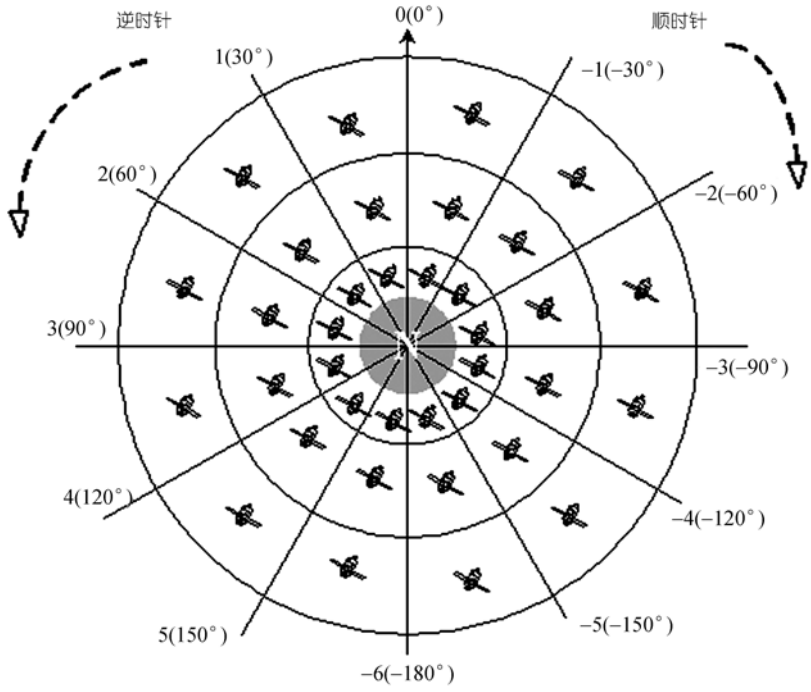

图 6 SWTN 的北极点透视图(N=6, M=12)

邻居(靠近极点), 另一个为外邻居; 邻近的轨道面也有 2 个邻居, 一个为顺时针邻居, 一个为逆时针邻居. 对于复杂结构, 每颗 LEO 卫星有 8 颗(当有 cross-seam 时则有 7 颗)邻居卫星: 轨道面内有 4 个邻居, 为内邻居 1 和内邻居 2, 外邻居 1 和外邻居 2; 邻近的轨道面也有 4 个邻居, 为顺时针邻居 1 和顺时针邻居 2, 逆时针邻居 1 和逆时针邻居 2. 但邻近缝隙的卫星为 7 条 ISL, 并没有顺时针邻居 2 或逆时针邻居 2. 请注意, 在第 1 节已经说明, 在 75°到 90°纬度区域的极点区域内, 没有面间星际链路(interplane ISL), 仅有面内星际链路(intraplane ISL).

5) 逻辑小区. 如图 5 和 6 所示, SWTN 是由一组(M/4 个)同心圆和一组从圆心出发的射线(2N 条)组成(惟一确定)的. M/4 个同心圆和 2N 条射线的相交将(北极面和南极面)两个大圆面分割成了一簇有规则的并且大小成比例的扇形(环)面区域, 每个大圆面有 N*M/2 小区, 共有 N*M 小区, 这里我们称这些扇型区域为逻辑小区(logic cell). 特别要注意的是, 它是按照 2N 条射线等分 360°经度和 M/4 个同心圆等分 90°纬度来划分逻辑小区的, 即全部逻辑小区的经度距离相等, 纬度距离也相等, 因此这些逻辑小区是大小成比例的、扇型的小区.

6) 逻辑节点. SWTN 上的逻辑小区可以抽象为一组有规则的离散点, 称之

为逻辑节点(logic node). 逻辑节点和逻辑小区有一一映射的关系. 请注意, 图 5 和 6 中的南北两极极点不包含在逻辑节点中, 即没有 LEO 卫星和极点对应.

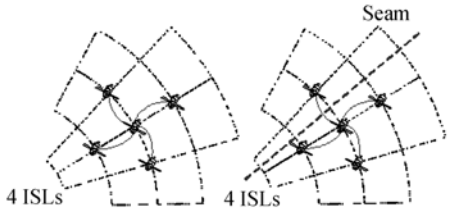

图 7　星际链路 ISL(简单结构)

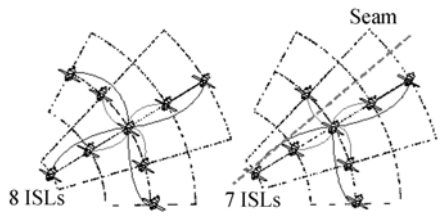

图 8　星际链路(复杂结构)

7) 切换和时间片. 如图 9 所示, 由于真实的 LEO 卫星在太空中围绕地球高速运动. 在某一瞬时刻, 每一颗 LEO 卫星都落在某一个确定的逻辑小区中, 而逻辑小区又有一定的面积(面积大小不相等, 但成比例, 它是由等大小的经度和纬度构成的), 因此每一颗 LEO 卫星滑过某一逻辑小区都有相等的并且是确定的时间片. 在此确定的时间片内各颗 LEO 卫星的下行覆盖区域(footprint)都投影(对准)到相对应的逻辑小区, 这样在时间片内 LEO 卫星、逻辑小区、逻辑节点它们三者之间有一一对应的关系, 而在时间片之间, LEO 卫星必须要动态切换来更新和逻辑小区的映射关系. 切换分为(轨道)面内切换和(轨道)面间切换两种方式(如图 9 所示): (a) 面内切换, 由于每一卫星轨道都有 M 颗卫星依次顺序地周而复始地绕着过南北极点区域的极轨道运行, 因此有相应的面内切换时间片(intraplane time slice), 令其为 T1. 在此面内切换时间片 T1 内, 各颗 LEO 卫星都对应此卫星轨道上各个逻辑小区, 而在时间片和时间片之间卫星需要切换到下一个逻辑小区. 面内切换仅仅是各轨道内部 LEO 卫星和逻辑小区的动态切换和映射关系. 在 SWTN 中实际上就是半径(同心圆)间的切换, 而不是角度间切换. (b) 面间切换, 由于地球周而复始地自西向东自转, 而 N 条卫星轨道都是极轨道, 因此就有相应的面间切换时间片(interplane time slice), 令其为 T2. 在此面间切换的时间片 T2 内, 各轨道的 LEO 卫星都对应某个角度的逻辑小区, 而当面间切换时刻到达时, 各轨道的 LEO 卫星就切换到下一个相邻角度的逻辑小区. 由于地球自转而形成的面间切换时间片在 SWTN 中实际上是角度间的切换,

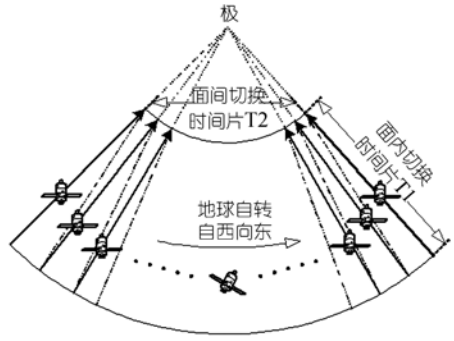

图 9　逻辑小区内卫星轨迹

而不是半径(同心圆)间的切换.

星座网络中的每一颗 LEO 卫星内部都有一个定时器(timer), 用来计时这两类时间片. 当一个时间片结束, 另一个时间片到来时, 各颗 LEO 卫星必须切换到下一个相应的逻辑小区, 即各颗 LEO 卫星将各自的覆盖区域(footprint)投影到下一个相应的逻辑小区上. 在本文, 星座网络中所有 LEO 卫星的定时器都是同步工作的, 但在真实的卫星星座网络中, 各颗 LEO 内部的定时器并不能保证非常严格的同步工作, 可能会有误差累积现象. 因此卫星地面控制中心应该周期性地消除此同步误差累积现象, 使各颗 LEO 卫星内部的定时器的同步误差被限定在一定的允许范围内. 当前, 任何真实的卫星系统都需要地面控制中心及时地控制并调节卫星的某些参数, 例如轨道高度、时钟同步等等. 因此本文提出的两个时间片的概念和切换的概念是合理的.

在每一个时间片中, LEO 卫星、逻辑小区(logic cell)、逻辑节点(logic node)三者之间的关系是一一映射的. 如

$$Logic\ Node \xleftarrow{\text{一一映射}} Logic\ Cell \xleftarrow[\text{一一映射}]{\text{时间片切换动态更新}} LEO\ Satellites \qquad (1)$$

所示, LEO 卫星通过两种时间片的切换来动态更新与逻辑小区(*Logic Cell*)的一对一映射, 而逻辑小区(Logic Cell)又和逻辑节点(Logic Node)有一对一的映射关系. 在如图 5 和 6 所示的 SWTN 上, 逻辑节点是一组有序的离散的点集合, 因此可以容易地编成二进制比特码并进行简单有效的加减和逻辑比较运算, 而此运算的结果又可以映射到相应的逻辑小区和 LEO 卫星中.

2.2 SWTN 的逻辑节点寻址方式及其编码

下面给出逻辑节点(logic node)的寻址方式及其编码: 绝对地址和相对地址.

(a) 绝对地址.

在如图 5 和 6 所示的 SWTN 星座网络上的每一逻辑节点都可以用经过圆心的半径和角度以及极面惟一确定.

对于半径, 我们使用正整数表示各个同心圆的半径, 正整数最大取值为 $R_{\text{most}}=[(M/4)-1]$ (请注意, 这里 M 必定是 4 的倍数, 其解释见附录 A). 这样 $R_0=0$, $R_1=1$, $R_2=2$, $R_3=3$, …, $R_{\text{most}}=[(M/4)-1]$(其中 $R_0=0$ 表示最小半径区域). 其编码方案采用 UpperBound($\log_2[M/4]$)位 2 进制比特表示. 例如, 图 5 所示 $M=24$, 则

$$R_{\text{most}}=5,\ \text{UpperBound}(\log_2[M/4])=3(\text{bits});$$

图 6 所示 $M=12$, 则

$$R_{\text{most}}=2,\ \text{UpperBound}(\log_2[M/4])=2(\text{bits}).$$

它们各自的半径编码方案如表 1 所示. 若 $M=36$, 则

$$\text{UpperBound}(\log_2[M/4])=4(\text{bits}).$$

对于角度, 我们规定了基准线为 0°(本文选择经过格林威治天文台的经度线

作为基准 0°). 从基准线逆时针(东经经度)旋转角度为正值, 顺时针(西经经度)旋转角度为负值. 对应于任何逻辑节点的角度可以用一个 UpperBound($\log_2[2N]$)位 2 进制补码来表示[$-N$, $N-1$]区间内的整数. 例如, 图 5 所示 N=12, 则

$$\text{UpperBound}(\log_2[2N])=5(\text{bits});$$

图 6 例子中 N=6, 则

$$\text{UpperBound}(\log_2[2N])=4\ (\text{bits}).$$

各自的角度编码方案如表 2 所示.

表 1　SWTN 的半径编码方案

M=24						M=12		
R_0	R_1	R_2	R_3	R_4	R_5	R_0	R_1	R_2
000	001	010	011	100	101	00	01	10

表 2　SWTN 的角度编码方案

	大星座(N=12)编码方案			小星座(N=6)编码方案		
逆时针角度	[0°, 15°)	0	00000	[0°, 30°)	0	0000
	[15°, 30°)	1	00001			
	[30°, 45°)	2	00010	[30°, 60°)	1	0001
	[45°, 60°)	3	00011			
	[60°, 75°)	4	00100	[60°, 90°)	2	0010
	[75°, 90°)	5	00101			
	[90°, 105°)	6	00110	[90°, 120°)	3	0011
	[105°, 120°)	7	00111			
	[120°, 135°)	8	01000	[120°, 150°)	4	0100
	[135°, 150°)	9	01001			
	[150°, 165°)	10	01010	[150°, 180°)	5	0101
	[165°, 180°]	11	01011			
顺时针角度	(0°, −15°)	−1	11111	(0°, −30°)	−1	1111
	(−15°, −30°)	−2	11110			
	(−30°, −45°)	−3	11101	(−30°, −60°)	−2	1110
	(−45°, −60°)	−4	11100			
	(−60°, −75°)	−5	11011	(−60°, −90°)	−3	1101
	(−75°, −90°)	−6	11010			
	(−90°, −105°)	−7	11001	(−90°, −120°)	−4	1100
	(−105°, −120°)	−8	11000			
	(−120°, −135°)	−9	10111	(−120°, −150°)	−5	1011
	(−135°, −150°)	−10	10110			
	(−150°, −165°)	−11	10101	(−150°, −180°)	−6	1010
	(−165°, −180°)	−12	10100			

此外, 还需要用 1 bit 来区分南北极面, 用 0 表示北极面, 1 表示南极面.

通过以上分析, 就可将 SWTN 上的任一逻辑节点用(极面, 半径, 角度)三元组惟一确定, 即绝对地址. 这样, 在 SWTN 上的单播路由问题就变成在具有地址坐标的逻辑节点间寻找最短通路的问题.

(b)相对地址.

由于 SWTN 具有循环结构特性, 因此在 SWTN 任何角度上的节点都可以看成是这个网络的基准线 0°(中心). 为了计算简单、表示方便, 我们这里就给出了相对地址的概念. 设源节点和目的节点的绝对地址分别为($pole_{\text{from}}$, R_{from}, θ_{from})和($pole_{\text{to}}$, R_{to}, θ_{to}), 我们将源节点作为基准, 令其地址为(0, 0, 0), 则目的节点相对于源节点的相对地址($pole$, R, θ)表示为以源节点为基准的相对距离(相对地址). 相对地址表示如下:

$$
\begin{aligned}
&Pole = Pole_{\text{to}} \oplus Pole_{\text{from}},\\
&R = R_{\text{to}} - R_{\text{from}},\\
&\theta = MOD_{2N}(\theta_{\text{to}} - \theta_{\text{from}}) = \begin{cases} (\theta_{\text{to}} - \theta_{\text{from}}), & (-N) \le (\theta_{\text{to}} - \theta_{\text{from}}) \le (N-1);\\ (\theta_{\text{to}} - \theta_{\text{from}}) - 2N, & N \le (\theta_{\text{to}} - \theta_{\text{from}});\\ (\theta_{\text{to}} - \theta_{\text{from}}) + 2N, & (\theta_{\text{to}} - \theta_{\text{from}}) < (-N). \end{cases}
\end{aligned} \tag{2}
$$

这里的取模 $2N$ 的意思是指θ 的取值范围在[$-N$, $N-1$]之间.

其中, $Pole$ 的取值为 0 或 1, 0 表示源节点和目的节点在相同极面; 1 表示在不同极面. R 的取值范围为[$-R_{\text{most}}$, R_{most}]之间, 若 $R>0$ 则表示目的地址在源地址的外环上, 若 $R<0$ 则表示目的地址在源地址的内环上, 若 $R=0$ 则表示目的地址和源地址在相同的环上. θ 由于是模 $2N$ 的运算, 因此取值范围还在[$-N$, $N-1$]之间, θ 为正表示目的节点在源节点的逆时针方向, θ 为负表示目的节点在源节点的顺时针方向.

同样也可以将目的节点作为基准, 则源节点相对于目的节点的相对地址($pole$, R, θ)就表示为以目的节点为基准的相对距离(相对地址), 则(2)式前后两项就要颠倒一下. 本文我们以源节点作为基准, 采用(2)式.

给出相对地址的目的是在 SWTN 中任何逻辑节点都可以使用相同的路由算法进行路由计算, 而不管各逻辑节点在 SWTN 中的绝对位置.

3 基于 SWTN 的分布式分组路由准则

本节只考虑 SWTN 未出现星际链路拥塞或损毁情况, 链路出现拥塞或损毁情况将在 4.3 小节讨论.

(A)在 SWTN 中经过的节点数要尽量少, 即分组经过的路由器的跳数要尽量少. 这是满足现有的 IP 路由协议中的最短路径优先规则. 即蜘蛛应经历尽量少的蛛网节点去吃掉猎物.

在一个真实的随时间动态变化的 LEO 星座网络中, 由于每一颗 LEO 卫星和逻辑小区、逻辑节点通过两类时间片切换方法(面内切换时间片和面间切换时间片)来相互一一映射. 但是必须要考虑如下情况: 当一个分组在某一星际链路(ISL)上传输时(ISL 存在一定的时延), 此时发生时间片的切换, 这会更新 LEO 卫星和

逻辑小区新的映射关系, 造成分组到达的下一跳不是其所想要去的下一跳 LEO 卫星. 这种现象在任何一种星座路由算法中都会存在. 由于出现这种情况是随机发生的, 因此在一个动态时变的星座网络中分组最短路由的总跳数相应地是一个估计值. 其估计值如下.

假设在地球和 LEO 卫星都静止的一个星座网络中, 分组最短路由的总跳数为:

1)若分组的源地址和目的地址在同一极面上, 则穿过极点和不穿过极点的分组总跳数分别为

$$total\ hops_{\text{in same plane}} = \begin{cases} |R|+|\theta|+one\ hop_{\text{up}}+one\ hop_{\text{down}}, & \text{不穿过极点;} \\ (N-|\theta|+1)+(R_{\text{from}}+R_{\text{to}})+one\ hop_{\text{up}}+one\ hop_{\text{down}}, & \text{穿过极点.} \end{cases} \tag{3}$$

2)若分组源和目的地址分别在不同极面上, 需要借助共轭节点概念. 定义: 南北两个极面的角度和半径相同的逻辑节点互称为对方的共轭节点. 若 $R\geq 0$, 即 $R_{\text{to}}\geq R_{\text{from}}$, 则应找到目的节点的共轭节点($pole^{*}_{\text{to}}$, R_{to}, θ_{to}), 此时分组路由应从源节点到目的节点的共轭节点加上同一轨道面的共轭节点到目的节点; $R<0$,即 $R_{\text{to}}<R_{\text{from}}$, 则应找到源节点的共轭节点($pole^{*}_{\text{from}}$, R_{from}, θ_{from}), 此时分组路由就是从同一轨道面的源节点到共轭节点加上源节点的共轭节点到目的节点. 由此得到分组总跳数为

$$total\ hops_{\text{in different plane}} = total\ hops_{\text{in same plane}} + 2(R_{\text{most}} - R_{\text{max}}), \tag{4}$$

这里 $R_{\max}$ 表示源节点和目的节点较大的外环半径.

将(3)式中两项相减得到$|\theta|-N/2-R_{\min}-1\geq 0$(或<0), 其中 Rmin 代表源节点和目的节点之间内环半径. 由此可判断分组路由是否应该还是不应该穿过极点. 这是下一节分组路由算法的主要依据.

(B)在相同跳数的条件下, (地理)距离尽量短. 这也是 SWTN 应用在 LEO 卫星星座网络的要求. 由于是低轨道卫星星座网络, 蜘蛛网络中的各个节点都对应某颗 LEO 卫星. 在星际链路中, 面内路由(intraplan ISL)由于距离相同, 因此传播迟延也相同; 面间路由(interplan ISL)由于在不同纬度上距离不相同, 因此传播迟延不相同. 高纬度的面间路由的传播迟延要比低纬度面间路由的短. 即蜘蛛应尽量在靠近蛛网中心的内圆(即高纬度)上经过, 而尽量不要在外圆上经过, 这样蜘蛛就能在尽量短的时间内抵达猎物.

(C)由于蜘蛛大脑简单, 路由选择算法也应尽可能简单. 不同于地面网络节点, 在卫星上由于星上计算、星上存储、星上处理等等都是非常宝贵的资源, 因此星上分组路由算法应尽可能简单并容易实现.

4 基于 SWTN 的分布式分组路由算法

4.1 分组在地面站

(a) 发送分组: 计算绝对地址.

发送分组的卫星地面站必须事先知道自己以及目的分组所在地面卫星站的经纬度. 当有分组需要发送时, 计算出发送和接收地面站各自相应的绝对地址, 填入到卫星分组头中, 发送到当前时刻本地面站相对应的 LEO 卫星中.

(b) 接收分组: 比较绝对地址.

地面站接收从 LEO 卫星转发下来的分组, 从卫星分组头中得到分组目的地址, 比较是否和本地面站的绝对地址相同(减法运算), 若是则接收本分组并交给高层处理, 否则丢弃不用.

4.2 分组在 LEO 星座网络中

首先考虑, 分组所在的 LEO 卫星和分组目的地址在同一极面.

(a) 计算相对地址, 即距离.

在 LEO 星座网络中, 每一颗 LEO 卫星的绝对地址($pole_{\text{satellite}}$, $R_{\text{satellite}}$, $\theta_{\text{satellite}}$)都是通过时间片概念来动态地映射到 SWTN 中相应逻辑节点的绝对地址. 一旦有分组到来, 将采用(2)式把当前时刻 LEO 卫星的逻辑节点和分组中的目的节点的绝对地址变换成基于当前时刻LEO卫星逻辑节点的相对地址($pole$, R, θ). 其中 R 的取值范围为$[-R_{\text{most}}, R_{\text{most}}]$, 若 $R>0$ 则表示目的地址在当前时刻 LEO 卫星所在的逻辑节点的外环上, 若 $R<0$ 则表示目的地址在当前时刻 LEO 卫星所在的逻辑节点的内环上, 若 $R=0$ 则表示目的地址和当前时刻 LEO 卫星所在的逻辑节点在相同的环上. 而θ 的取值范围在$[-N, N-1]$之间, 若$\theta = 0$ 表示当前时刻 LEO 卫星所在的逻辑节点和目的节点在同一轨道面上, 若$\theta > 0$ 表示目的节点在以当前时刻 LEO 卫星所在的逻辑节点为基准的逆时针方向, 若$\theta < 0$ 表示目的节点在以当前时刻 LEO 卫星所在的逻辑节点为基准的顺时针方向.

(b) 若$\theta = 0$(例如图 5 中从 A 到 E), 表示当前卫星和目的节点在同一轨道上, 此时应做轨道面内下一跳路由:

1) 若 $R>0$,表示当前卫星在内环, 目的节点在外环, 因此本分组向外环方向做下一跳;

2) 否则, 若 $R<0$, 表示当前卫星在外环, 目的节点在内环, 因此本分组向内环方向做下一跳;

3) 否则, 必定 $R=0$, 此时表示当前卫星就是目的节点卫星, 因此下一跳就是卫星向地面接收站发送(地面接收站可以成功接收本分组)本分组.

(c) 若$|\theta| - N/2 - R_{\min} - 1 < 0$ (例如图 5 中从 A 到 H 和从 M 到 N), 根据第 3 节的

准则(B), 应在高纬度路由. 此时:

若当前卫星在极点区域内, 由于极点区域内没有面间路由, 因此本分组向外环方向做下一跳. 请注意, 由于各颗 LEO 卫星内部有一个定时器(timer), 因此在任何时刻能很容易知道其自身是否处于极点区域.

若 $R\geq 0$ 且当前卫星不在极点区域(这表示当前卫星在内环或者相同的环上); 或者若 $R<0$ 且当前卫星的内邻居在极点区域, 这两种情况分组需要做轨道面间路由: 若 $\theta>0$, 表示目的节点在以当前时刻 LEO 卫星所在的逻辑节点为基准的逆时针方向, 因此本分组逆时针做下一跳; 若 $\theta<0$, 表示目的节点在以当前时刻 LEO 卫星所在的逻辑节点为基准的顺时针方向, 因此本分组顺时针做下一跳. 请注意, $R<0$ 表示当前卫星在 SWTN 中的半径 $R_{\text{satellite}}\geq 0$, 因此它一定有内邻居, 而且当前卫星也能容易地知道其内邻居是否处于极点区域(由于各颗 LEO 卫星内部有一个定时器). 例如, 图 5 的 SWTN 中, 当前卫星若 $R_{\text{satellite}}=1$, 则其内邻居一定在极点区域; 图 6 的 SWTN 中, 借助于定时器计时面内切换时间片(intraplane time slice)的一半, $R_{\text{satellite}}=1$ 的当前卫星也能很容易获知其内邻居是否当前处在极点区域.

若 $R<0$ 且当前卫星的内邻居不在极点区域, 表示当前卫星在外环, 目的节点在内环, 因此本分组向内环方向做下一跳.

(d) 若 $|\theta|-N/2-R_{\min}-1\geq 0$ (例如图 5 中从 I 到 J), 表示需要越过极点跳数比不越过极点的跳数要少, 根据第 3 节的准则(A), 当前分组需朝着圆心(极点)方向向内做下一跳 Intraplane 路由.

(e)若当前分组所在的 LEO 卫星和分组目的地址不在同一极面上, 需要找到对应的共轭节点(例如图 5 中从 K 到 L, 其中 L^*为 L 对应的共轭节点).

若 $R\geq 0$, 即 $R_{\text{to}}>R_{\text{satellite}}$, 则先将目的节点映射到当前时刻LEO卫星所在的极面上的共轭点($pole^{*}_{\text{to}}$, R_{to}, θ_{to}), 然后应用上面(a)~(d)的情形去做分组的下一跳路由.

若 $R<0$, 则 $R_{\text{to}}<R_{\text{satelllite}}<R_{\text{most}}$, 本分组向外环方向做下一跳. 即若 $R_{\text{satelllite}}<R_{\text{most}}$, 做($pole_{\text{satellite}}$, $R_{\text{satellite}}$, $\theta_{\text{satellite}}$)到($pole_{\text{satellite}}$, $R_{\text{satellite}}+1$, $\theta_{\text{satellite}}$)的下一跳; 若 $R_{\text{satelllite}}=R_{\text{most}}$, 则分组的下一跳应越过本极面到另一极面去, 做($pole_{\text{satellite}}$, $R_{\text{satellite}}$, $\theta_{\text{satellite}}$)到($pole^{*}_{\text{satellite}}$, $R_{\text{satellite}}$, $\theta_{\text{satellite}}$)的下一跳.

(f) 对于复杂结构的星际链路, 如图 8 所示.

对于复杂结构的 ISL 的 LEO 星座中分组的下一跳处理也很简单, 仅仅是将(a)~(e)中稍加改动即可. 即, 先减法运算得到基于当前 LEO 卫星的相对地址, 然后看($pole$, R, θ)正负和大小来判断本分组下一跳的方向. 当判断好本分组的下一跳方向后, 此时需要判别是否走复杂结构中邻居 2 的星际链路. 若走面内星际链路, 则$|R|$是否大于等于 2? 若是则走相应的邻居 2 链路, 否则走邻居 1 链路. 若走面间星际链路, 则需判断是否经过缝隙链路? 若是则走邻居 1 链路, 否则还需判

断$|\theta|$是否大于等于 2? 若是则走邻居 2 链路, 否则走邻居 1 链路.

4.3 星际链路拥塞或损毁与分组次优路由

由于星际链路可能会出现拥塞或者损毁情况, 本算法也考虑分组的次优路径, 且次优路径的计算也是简单有效的.

若最短路径的下一跳为从$|\theta|$到$|\theta|-1$, 且此 ISL 发生拥塞或者损毁, 若$|R|>0$, 则可选择次优路径的下一跳从$|R|$到$|R|-1$. 同理, 若最优路径的下一跳为从$|R|$到$|R|-1$, 但此 ISL 发生拥塞或者损毁, 若$|\theta|>0$, 则可选择次优路径的下一跳从$|\theta|$到$|\theta|-1$.

举例说明, 如图 5 所示, 对于星际链路$|\theta|$到$|\theta|-1$ 拥塞或者损毁的情况, 从 A 到 H 的最短路径为 $ABCDH$, 若星际链路 CD 拥塞或者损毁, 则次优路径为 $ABCGH$; 若星际链路 BC 或者 AB 拥塞或者损毁坏, 则次优路径分别为 $ABFGH$ 和 $AEFGH$. 同理, 对于星际链路$|R|$到$|R|-1$ 拥塞或者损毁的情况, 从 E 到 D 的最优路径为 $EABCD$, 若星际链路 EA 拥塞或者损毁, 则次优路径为 $EFBCD$.

请注意, 一颗卫星若有四条星际链路, 从任何一个方向进来的分组, 只可能有两个方向的出口, 一个是最短路径的下一跳, 另一个就是次短路径的下一跳, 而若从第 3 个方向出去在 SWTN 中相应就会绕了一个弯子, 容易发生自环路由现象, 因此本文的算法不走这第 3 条路.

尽管在星座中存在许多条次优路径, 但本文仅给出一种寻找次优路径的方法, 并没有过多地考虑多条次优路径的算法. 事实上, 在星座中寻找多条次优路径的算法是需要认真考虑的另一关键技术. 本文主要研究分组在星座中的最短路由.

4.4 阻止自环路由现象

如果一个分组路由算法是分布式的并且没有任何机制保证, 例如路由记录(routing recording)机制, 那么此“盲目”的转发分组路由算法可能会发生分组路由自环现象. 因此, 本节分析基于 SWTN 的分布式分组路由算法在星座网络中是否会发生自环现象.

幸运的是, 本文提出的基于 SWTN 的分布式分组路由算法天生对自环现象具有免疫力, 不需要附加某种机制来阻止自环现象发生. 这是由于 SWTN 是一张非常规则的三维球面网格, 因此可以用(*pole*, *radius*, *angle*)三元组对 SWTN 的逻辑节点进行地址编码并进行相应的加减运算. SWTN 中的相对地址不但反映了任意两个逻辑节点的相对距离, 而且也反映了这两个逻辑节点的相对位置. 因此, 基于 SWTN 的分布式分组路由算法必定不会发生自环现象. 这也说明本分组路由算法的简单性和鲁棒性.

4.5　解决具有相位差的星座网络中的分组路由问题

本文提出的 SWTN 是一张非常规则的三维网格, 因此非常适合没有相位差的极轨道LEO卫星星座网络. 由于星座网络是由人来设计的, 因此可以事先设计成不带有相位差的星座网络, 这样本文提出的基于 SWTN 的分组路由算法可以应用得非常好.

但有些情况下, 一个星座网络中的 LEO 卫星具有相位差, 那么需要考虑本文提出的分组路由算法适用于此星座网络的情况. 一般情况下, 相邻轨道间卫星的相位差是相等的, 即 $Phasing$ =360°/2M. 一个具有相位偏差的星座网络如图 10 所示, 其中 N=6, M=12, $phasing$=15°.

在 2.1 小节的 7)里有时间片和切换的概念. 对于一个没有相位差的 SWTN (即 $phasing$=0°), 星座网络中所有的 LEO 卫星同时切换. 而对于一个有相位差的 SWTN($phasing$≠0°), 则星座网络中的卫星并不是同时切换的: 对于面间切换时间片, 所有卫星都是同时切换的; 而对于面内切换时间片, 具有相同相位的那些卫星同时切换, 不同相位的卫星的切换是不同步的. 一般情况下, 这个不同步时间等于半个面内切换时间片时间, 如图 10 所示.

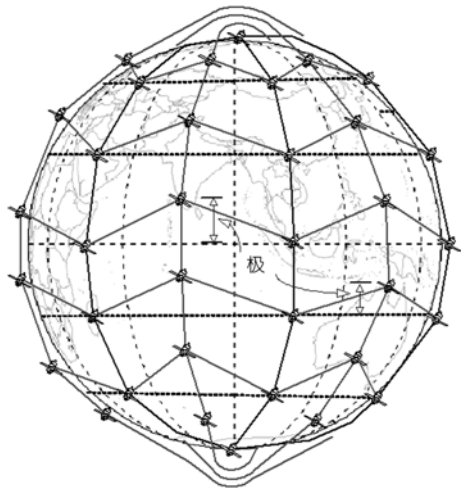

图 10　带有相位差的星座网络(N=6, M=12, $phasing$=15°)

对于具有相位差的星座网络, 将相同相位的那些卫星同时切换, 不同相位的卫星间存在一个切换时间差. 通过这种映射关系, 就将此星座网络映射到 SWTN 中. 而基于 SWTN 的分组路由算法不需要改动, 也适用于具有相位差的星座网络(仿真实验见下一节).

4.6　星上时间复杂度、计算复杂度、空间复杂度比较

星上处理、星上计算、星上存储复杂度是分组路由算法需要考虑的几个关键因素. 有许多算法用来确定网络中任意两点之间的最短路径. 但大多数算法由于使用连接矩阵来反映网络的拓扑, 因此时间复杂度非常高. 例如, Dijkstra最短路径算法的时间复杂度为$O(N^2)$, Bellman最短路由算法最坏情况时间复杂度为$O(N^3)$, 平均情况为$O(N\log(N))$, 这里N代表网络中的节点数目. 因为星座是由几百颗LEO卫星组成随时间变化的动态拓扑网络, 所以使用连接矩阵的这类最优路由算法其星上时间复杂度非常高. 另外尽管在文献[15]中, 仿真结果给出在Inter

Pentium III 450 MHz处理器中单个分组的下一跳选择只需花费 5 μs时间, 但其涉及许多乘法、除法、正弦函数和余弦函数的计算来得到加权曼哈顿街道网(WMSN)中相应的权值, 因此其星上计算复杂度代价较高.

所有关于计算分组最短路由的算法由于需要路由表, 会带来较高的星上存储复杂度, 因为星座由 $N\times M$ 颗 LEO 卫星组成. 此外, 由于 LEO 卫星不停地高速运动, 因此星上路由表需要定期的更新以适应星座网络的动态拓扑结构. 更新路由表主要分为两种方式: 预先存储路由表方式和周期性的计算路由表方式. 预先存储路由表方式是将各个更新周期时间片的路由子表都保存在各颗 LEO 卫星中一张大路由表内, LEO 卫星每次只取出当前更新时间片内路由子表的信息进行路由查表计算得到下一跳路由, 这会导致星上存储空间很大, 并且此方法不能对网络的拥塞或链路失效等情况及时做出反应. 另一种是由地面路由控制中心(routing control center)周期性地进行整个星座网络的路由计算, 并周期性地更新到星座网络中所有LEO卫星中. 但更新周期频率是一个矛盾, 更新频率过快会导致占用较大的卫星网络带宽资源, 可能会出现网络瓶颈; 更新频率过慢又会出现星上路由表陈旧而失效, 导致分组不能得到正确的路由. 总之, 路由表的定期更新机制一方面需要占用大量的、十分宝贵的星上存储资源, 另一方面有可能会占用一定的网络带宽资源.

本文提出的分组路由算法由于仅仅需要一次比特的减法和比较运算, 其星上计算量很小, 特别是, 本算法不需要任何星上路由表. 因此其星上时间复杂度、计算复杂度、空间复杂度都非常低, 特别适用于由 LEO 卫星簇组成的星座网络进行分组路由.

5 基于 SWTN 的分组路由算法具体编码方案实现举例

具体到编码实现时, 由于 LEO 卫星是在地球上空不停地高速运动, 因此应该设定两个时间片(时隙周期), 在此时间片内每一颗 LEO 卫星都一一对应 SWTN 中每一个逻辑小区, 也就一一对应每一个逻辑节点.

这样 SWTN 中各个逻辑节点都采用绝对地址(极面, 半径, 角度)来惟一表示(以格林威治天文台的经度为基准). 表 1 和表 2 为两个星座网络编码方案: 对于大星座网络(N=12, M=24), 采用 1bit 表示南北极面, 3bit 表示 M=24 的半径(表示范围为 0~5), 5bit 表示 N=12 的角度, 如图 11 所示, 一共需要 9 bit 来表示某一个逻辑节点. 对于小星座网络(N=6, M=12), 采用 1

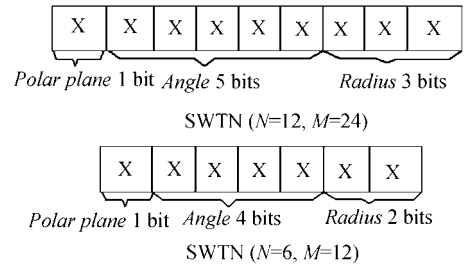

图 11 SWTN 中逻辑节点的地址编码表示

bit 表示南北极面, 2 bit 表示 M=12 的半径(表示范围为 0~2), 4 bit 表示 N=6 的角度, 如图 11 所示, 一共需要 7 bit 来表示某一个逻辑节点. 请注意, 不同的 N 和 M 值就会有不同的基于 SWTN 的编码方案.

此时具体编码实现都是用 2 进制比特来表示. 这样星上分组下一跳路由计算就很简单, 仅仅只需要做一次比特间的加减运算即可. 可以用硬件高速实现. 若某卫星有分组到来, 则:

(a)当前卫星所对应的逻辑节点和分组中目的节点用绝对地址来表示. 卫星分组头的编码格式如图 12 所示.

(b) 采用(2)式, 计算出相对地址, 用 2 进制表示. 其中大星座网络(N=12, M=24)的相对地址如

$$\begin{aligned}&Pole = Pole_{\text{to}} \oplus Pole_{\text{satellite}},\\&R = R_{\text{to}} - R_{\text{satellite}},\\&\theta = MOD_{(11000)}(\theta_{\text{to}} - \theta_{\text{satellite}}) = \begin{cases}(\theta_{\text{to}} - \theta_{\text{satellite}}),\ (10100) \le (\theta_{\text{to}} - \theta_{\text{satellite}}) \le (01011);\\(\theta_{\text{to}} - \theta_{\text{satellite}}) - 11000,\quad 01100 \le (\theta_{\text{to}} - \theta_{\text{satellite}});\\(\theta_{\text{to}} - \theta_{\text{satellite}}) + 11000,\quad (\theta_{\text{to}} - \theta_{\text{satellite}}) < (10100)\end{cases}\end{aligned} \tag{5}$$

所示, 小星座网络(N=6, M=12)的相对地址如

$$\begin{aligned}&Pole = Pole_{\text{to}} \oplus Pole_{\text{satellite}},\\&R = R_{\text{to}} - R_{\text{satellite}},\\&\theta = MOD_{(1100)}(\theta_{\text{to}} - \theta_{\text{satellite}}) = \begin{cases}(\theta_{\text{to}} - \theta_{\text{satellite}}),\ (1010) \le (\theta_{\text{to}} - \theta_{\text{satellite}}) \le (0101);\\(\theta_{\text{to}} - \theta_{\text{satellite}}) - 1100,\quad 0110 \le (\theta_{\text{to}} - \theta_{\text{satellite}});\\(\theta_{\text{to}} - \theta_{\text{satellite}}) + 1100,\quad (\theta_{\text{to}} - \theta_{\text{satellite}}) < (1010)\end{cases}\end{aligned} \tag{6}$$

所示. 其中极面符号是当前卫星逻辑节点和目的节点中的极面符号位异或得到的.

(c)若 $R \ge 0$ 则 $R_{\max}=R_{\text{to}}$, $R_{\min}=R_{\text{satellite}}$; 否则 $R_{\max}=R_{\text{satellite}}$, $R_{\min}=R_{\text{to}}$.

(d)若 $Pole$=1, 表示不在同一极面; 否则若 $Pole$=0, 表示在同一极面. 此时应按照本文第 4 节论述的路由算法进行本分组的下一跳.

还要说明一点, 具体的分组编码格式如图 12 所示, 这里借鉴了多协议标签交换(MPLS)的方法[19]. 即, 将进入到卫星星座网络域中的分组(不管是IP分组还是ATM信元都可以)打上一个垫片(shim), 即卫星分组头, 在卫星星座中的下一跳路由就通过这个垫片来高速处理, 而并没有改变原有分组的任何格式, 当分组从此卫星星座网络域出来时就将此垫片去掉, 即可得到原有的分组. 通过使用MPLS中垫片的方法, 卫星星座网络中不但可以路由标准的IP分组, 还可以交换ATM信元以及其他格式的分组包, 带来极大的灵活性. 注意在图 12 中的“ Type ”

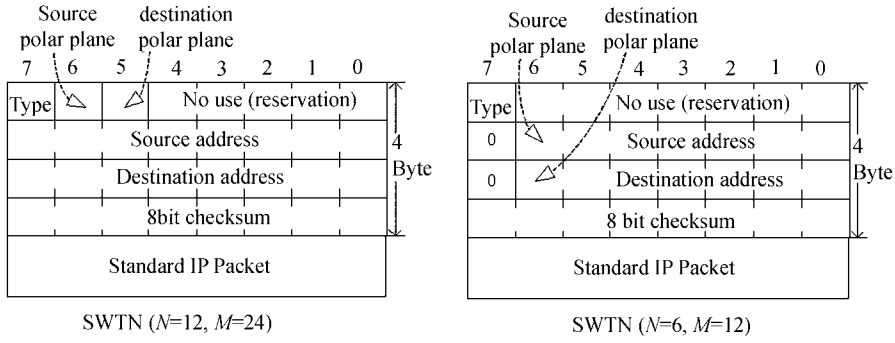

图 12 卫星分组头(4Bytes)

字段, 0 表示单播分组, 1 表示多播分组, 本文只考虑单播路由, 多播路由算法以后论述. 8bit 校验和字段是校验卫星分组头垫片中的信息, 例如可以是前 3 个字节信息的补码之和的补码.

6 实验仿真结果分析

为了验证基于 SWTN 的分组路由算法的正确性和性能, 本文用 OPNET 开发出三个 LEO 卫星星座网络系统. 其仿真实验模型如下:

1) 大星座网络: 如图 1 所示, 一个共有 288 颗 LEO 极轨道卫星围绕地球均匀组成的星座系统, 其中有 N=12 个轨道面, 每个面有 M=24 个卫星, 带有缝隙面星际链路, 卫星之间相隔 15.0°(经纬度).

2) 小星座网络: 如图 2 所示, 由 72 颗 LEO 极轨道卫星围绕地球均匀组成的星座系统, 其中有 N=6 个轨道面, 每个面有 M=12 个卫星, 带有缝隙面星际链路, 卫星之间相隔 30.0°(经纬度).

3) 带有相位差的星座网络: 如图 10 所示, 由 72 颗 LEO 极轨道卫星围绕地球均匀组成的星座系统, 其中有 N=6 个轨道面, 每个面有 M=12 个卫星, 带有缝隙面星际链路, 卫星之间相隔 30.0°(经纬度). 相邻轨道间的 LEO 卫星相位差为 15.0 °.

极点区域定义为 75°到 90°纬度区域. 卫星海拔高度(距地面)1680.860307 km, LEO卫星围绕地球极轨道高速运动, 这样卫星绕地球公转一圈的周期为 2 小时(即 7200 秒绕 1 圈, 5 分钟走纬度 15°, 10 分钟走纬度 30°). 通过卫星轨道周期公式得到 2 小时[18]. 另外地球自西向东周而复始地自转, 自转周期为 23 小时 56 分 4.09 秒. 这样两类时间片就为: 对于面内切换时间片T1, 大星座网络为 5 分钟切换一次面内逻辑节点地址, 小星座网络为 10 分钟切换一次; 对于面间切换时间

片T2, 大星座网络为 59 分 50.17041666666960 秒切换一次面间逻辑节点地址, 小星座网络为 119 分钟 40.34083333333920 秒切换一次. 本文给出了卫星有 4 个邻居的星座系统的仿真, 由于篇幅所限, 对于 8 个邻居的复杂星际链路结构星座系

表 3　各个城市(地区)的经纬度坐标及其绝对地址

城市(地区)	纬度	经度	绝对地址
西安(发送源)	34n15	108e52	(0,3,7)
上海	31n14	121e28	(0,3,8)
东京	35n42	139e46	(0,3,9)
新德里	28n36	77e12	(0,4,5)
莫斯科	55n45	37e35	(0,2,2)
夏威夷	21n17	157w42	(0,4,-11)
纽约	40n43	74w0	(0,3,-5)
悉尼	33s52	151e13	(1,3,10)
南极洲	80s30	140e0	(1,0,9)
开普敦	33s55,	18e22	(1,3,1)
布宜诺斯艾利斯	34s36	58w27	(1,3,-4)

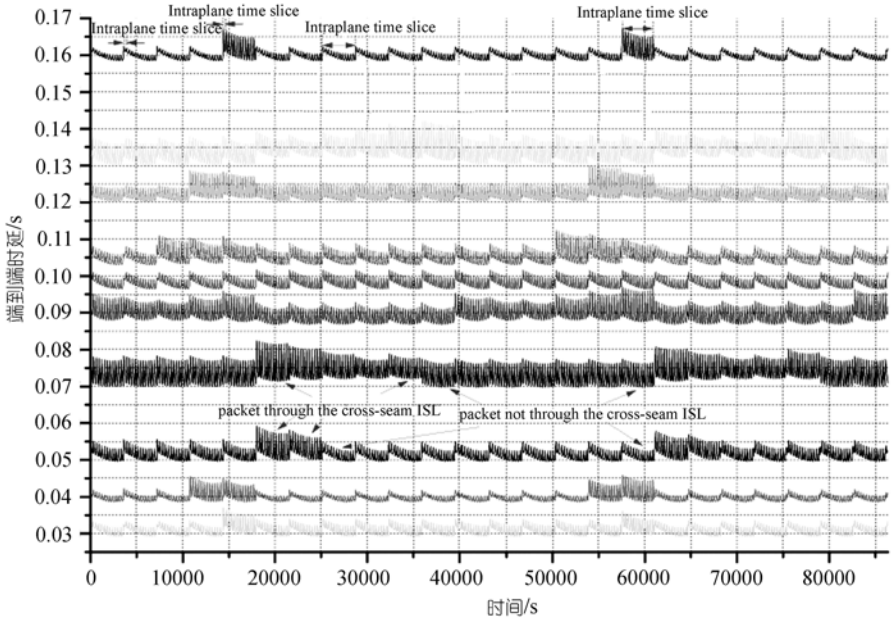

图 13　从西安发送分组到各城市(地区)的端到端的时延(SWTN N=12, M=24)

从下到上依次为: 上海、东京、新德里、莫斯科、夏威夷、纽约、悉尼、南极洲、开普敦、布宜诺斯艾利斯

统仿真没有给出, 但其结果基本相似, 只不过分组在星座的跳数变少.

仿真实验中, 发送端和接收端数据率以及 ISL 带宽都为 1,024,000bit/s. 星座网络中的分组由大小为 4096bits 的 IP 分组和 4 字节的卫星分组头组成.

本文给出了从西安发送IP分组到世界其他(10个)城市(地区)的分组时延随时间(仿真时间为 24 小时即一天)的变化情况, 如表 3 和图 13~15 所示. 请注意, 本文给出的在 LEO 星座网络中进行分组路由仿真实验, 是真正仿真了卫星围绕地球公转、而地球也在自转的真实的系统结果.

通过三个仿真实验, 可以看出本文提出的基于 SWTN 的分组路由算法非常适用于由极轨道(或准极轨道)LEO 卫星簇组成的星座网络.

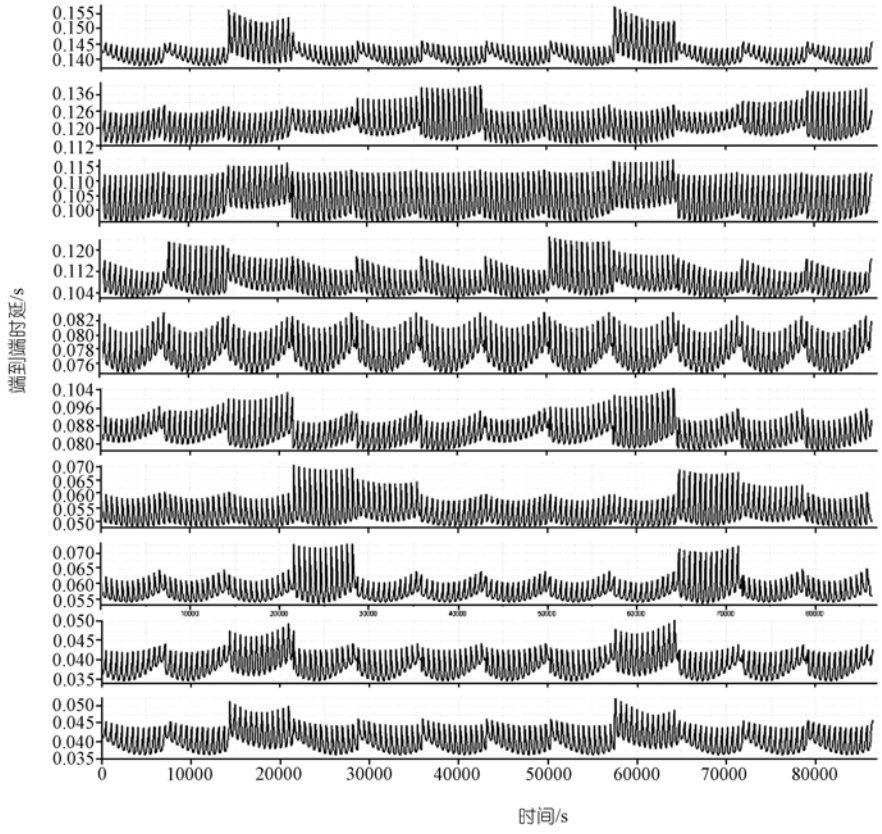

图 14 从西安发送分组到各城市(地区)的端到端的时延(SWTN, N=6, M=12)

从下到上依次为: 上海、东京、新德里、莫斯科、夏威夷、纽约、悉尼、南极洲、开普敦、布宜诺斯艾利斯

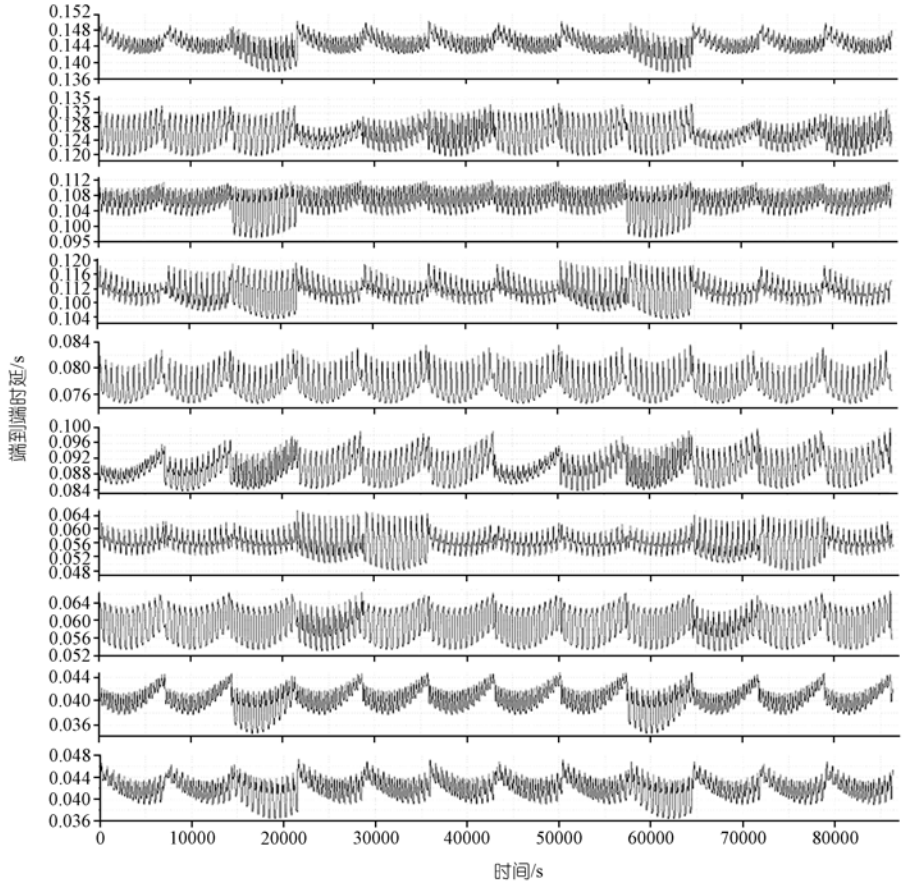

图 15　从西安发送分组到各城市(地区)的端到端的时延(带有相位差的 SWTN, N=6, M=12, *phasing*=15°)

从下到上依次为: 上海、东京、新德里、莫斯科、夏威夷、纽约、悉尼、南极洲、开普敦、布宜诺斯艾利斯

7　结论

本文提出了一种分布式分组路由算法, 即基于 SWTN 的分组路由算法. 本算法适用于由极轨道 LEO 卫星簇组成的星座网络. 路由的判断是依据当前时刻分组所在的 LEO 卫星位置(绝对地址)和本分组自身携带的地址信息来决定的. 显而易见, 通过使用基于 SWTN 的分组最短路由准则, 由本路由算法得到的分组路由是最少跳数路径中的最短路由. 本分布式分组路由算法不会发生自环路由现象. 星上路由的计算不用查找路由表, 并且星上计算量非常小, 仅仅通过 1 次加减和比较运算即可得出相应的下一跳分组路由. 本算法可以同时处理简单结构和复

杂结构的星际链路星座分组路由, 以及具有相位差的卫星星座网络.

参 考 文 献

1 Akyildiz I F, Jeong S. Satellite ATM networks: A survey. IEEE Communications Magazine, 1997, 35(7): 30~44 [DOI]

2 Werner M. A dynamic routing concept for ATM-based satellite personal communication networks. IEEE Journal on Selected Areas in Communications, 1997, 15(8): 1636~1648 [DOI]

3 Werner M, Berndl G. Performance of optimized routing in LEO intersatellite link networks. IEEE Vehicular Technology Conference, 1997, v1: 246~250

4 Chang H S, Kim B W, Lee C G, et al. Topological design and routing for low-earth orbit satellite networks. IEEE Global Telecommunications Conference, 1995, v1: 529~535

5 Uzunalioglu H, Akyildiz I F, Bender M D. A routing algorithm for LEO satellite networks with dynamic connectivity. Wireless Networks, 2000, 6(3): 181~190 [DOI]

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

7 Mauger R, Rosenberg C. QoS guarantees for multimedia services on a TDMA-based satellite network. IEEE Communications Magazine, 1997, 35(7): 56~65 [DOI]

8 Uzunalioglu H, Akyildiz I F, Ye sha Y, et al. Footprint handover rerouting protocol for LEO satellite networks. Wireless Networks, 1999, 5(5): 327~337 [DOI]

9 Tsai K, Ma R. Darting: A cost effective routing alternative for large space-based dynamic topology networks. Journal of Engineering and Applied Science, 1995, v2: 682~686

10 Raines R A, Janoso R F, Gallagher D M, et al. Simulation of two routing protocols operating in a low earth orbit satellite network environment. Journal of Engineering and Applied Science, 1997, v1: 429~433

11 Chang H S, Kim B W. Performance comparison of static routing and dynamic routing in low-earth orbit satellite networks. IEEE Vehicular Technology Conference, 1996, v2: 1240~1243

12 Henderson T R, Katz R H. On distributed, geographic-based packet routing for LEO satellite networks. IEEE Global Telecommunications Conference, 2000, v2: 1119~1123

13 Wood L, Clerget A, Andrikopoulos I, et al. IP routing issues in satellite constellation networks. International Journal of Satellite Communications, 2001, 19(1): 69~92

14 Maxemchuk N. Routing in the Manhattan street network. IEEE Transactions on Communications, 1987, 35(5): 503~512

15 Ekici E, Akyildiz I F, Bender M D, et al. A distributed routing algorithm for datagram traffic in LEO satellite networks. IEEE/ACM Transactions on Networking, 2001, 9(2): 137~147

16 Walker J G. Satellite constellations. Journal of the British Interplanetary Society, 1984, 37(12): 559~572

17 Lo M W. Satellite-constellation design. Computing in Science & Engineering, 1999, 1(1): 58~66

18 Sellers J J. Understanding Space: An Introduction to Astronautics. McGraw-Hill Inc, 1996

19 Eric C R, Arun V, Ross C. Multiprotocol Label Switching Architecture. RFC 3031, January 2001

附录 A 解释 M 为何是 4 的倍数

一般来说, M 应该为 4 的倍数. 这是因为地球可以被对称地分割为南北半球和东西半球. 一个极轨道 LEO 星座网络最好是将 LEO 卫星均匀地分布在南北半

球和东西半球. 因此, 我们得到 M 应该为 4 的倍数. 这样在 SWTN 中, 南北极面的逻辑小区数目相等, 并且等间隔地均匀分布. M 为 4 的倍数的 SWTN 具有最简单的拓扑结构, 因此基于此 SWTN 的分组路由算法也相应简单.

若 M 不为 4 的倍数, 那么此 SWTN 就比 M 为 4 的倍数的 SWTN 复杂些. 可以进一步分为两类:

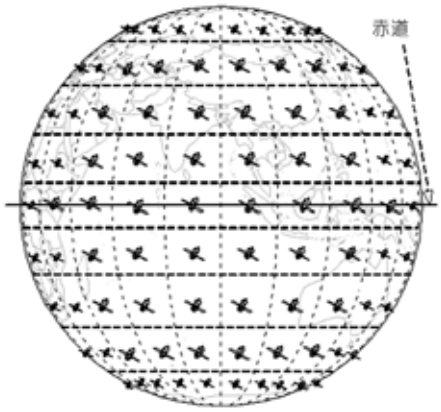

图 A1　由 216 颗极轨道 LEO 卫星组成的星座网络(N=12, M=18)

1) 若 M 模 4 余数为 2, 则此 SWTN 略微与 M 为 4 的倍数的 SWTN 有些不同. 如图 A1 所示, 此类 SWTN 在赤道环处有一条特殊的逻辑小区, 此逻辑小区一半区域属于北极面, 另一半区域属于南极面. 换句话说, 我们也可以认为在赤道环处这条特别的逻辑小区环带既属于北极面也属于南极面. 因此本文提出的分组路由算法也适用于此类 SWTN, 而不需要做任何修改.

2) 若 M 模 4 余数为 1 或者 3, 则此 SWTN 具有周期变化的最复杂的拓扑结构. 在北极面和南极面的逻辑小区的数目是不相等的, 时而南极面的逻辑小区数目比北极面的多, 时而北极面的逻辑小区数目比南极面多. 但是此类 SWTN 的周期变化特性是事先知道的, 各颗 LEO 卫星可以使用其内部的定时器来计时这种周期变化, 因此基于这类 SWTN 的分组路由算法应该做相应的修改, 必须考虑此类 SWTN 的周期变化特点.

总之, 由于卫星星座是事先由人精心设计的, 因此当构建一个由 N 个极轨道, 每个轨道有 M 颗 LEO 卫星组成的星座网络时, M 最好应为 4 的倍数.

附录 B　有关 SWTN 具有缝隙星际链路或者没有缝隙星际链路的情况

当星座网络没有缝隙星际链路时(例如 IRIDIUM 系统), 此 SWTN 就相应的是一个具有先天发育缺陷的三维球面网格. 例如, 若我们将图 5 所示的缝隙星际链路删除(从 0 到–1 和从 11 到–12), 就得到一个没有缝隙星际链路的 SWTN, 如图 B1 所示. 由于事先知道此星座网络没有缝隙星际链路, 因此我们也可以很容易地找到此类 SWTN 中任意两个逻辑节点的最短路由. 基于没有缝隙星际链路的 SWTN 的分组路由算法不同于带有缝隙星际链路的 SWTN 的分组路由算法, 必须做相应的修改.

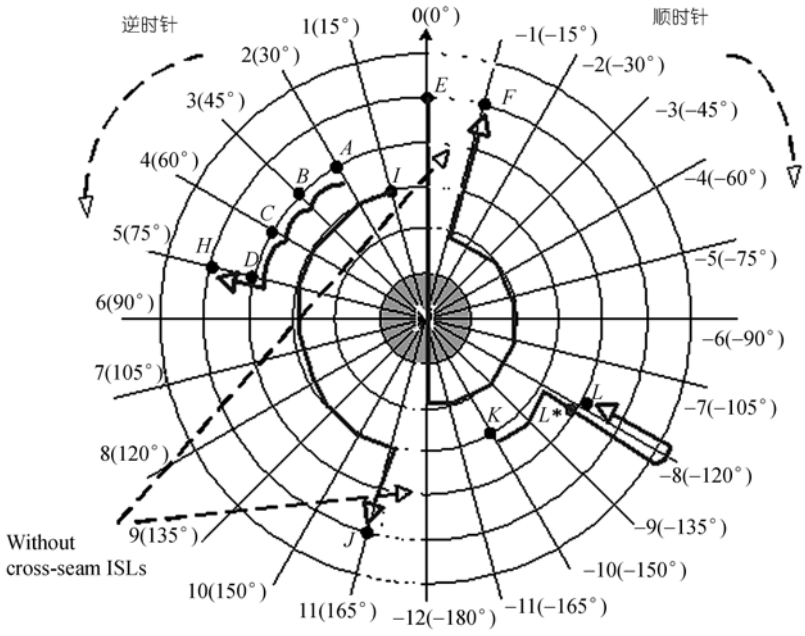

图 B1 不带有 cross-seam ISLs 的 SWTN (N=12, M=24)

本文主要研究适用于带有缝隙星际链路的极轨道 LEO 卫星星座网络的分组路由算法, 因为带有缝隙星际链路的星座网络比不带有缝隙星际链路的星座网络效率高.