文章编号:1007-5321(2020)02-0094-09 **DOI**:10.13190/j.jbupt.2019-112

基于 Linux 系统的 LEO 卫星动态路由协议研究与实现

王 程[1], 徐 玭[1], 张素兵[2], 王力权[3], 王卫东[1]

(1. 北京邮电大学 电子工程学院, 北京 100876; 2. 中国电子技术标准化研究院, 北京 100007;
3. 中国电子科技集团公司 第五十四研究所, 石家庄 050081)

摘要: 基于虚拟拓扑算法的思想,提出了一种基于网络状态的低轨(LEO)卫星动态路由协议. 通过预测卫星周期运动来划分快照,按照每个快照内的拓扑预计算路由. 根据卫星节点的实时状态动态调整网络拓扑,并重计算路由,以提高卫星网络的应急能力,增强网络的抗毁性. 在 NS3 仿真平台验证了协议的正确性,还在 Linux 系统实现了该路由协议,针对划分的 4 个模块在实现功能时的难点提出了解决方案. 在 Linux 系统中对实现的协议进行了功能测试和性能测试,验证了协议的路由功能. 与传统的基于虚拟拓扑算法相比,该协议在时延、丢包率和吞吐量性能上有所提升.

关 键 词: 低轨卫星; 虚拟拓扑; 静态路由; 动态路由; Linux

中图分类号: TN927 + .2 **文献标志码**: A

Research and Implementation of Dynamic Routing Protocol for LEO Satellites Based on Linux System

WANG Cheng[1], XU Pin[1], ZHANG Su-bing[2], WANG Li-quan[3], WANG Wei-dong[1]

(1. School of Electronic Engineering, Beijing University of Posts and Telecommunications, Beijing 100876, China;
2. China Electronics Standardization Institute, Beijing 100007, China;
3. The 54th Research Institute, China Electronics Technology Group Corporation, Shijiazhuang 050081, China)

Abstract: Based on the idea of virtual topology algorithm, a network state based dynamic routing protocol for low earth orbit (LEO) satellites that combines static routing, and dynamic routing is proposed. It firstly divides snapshot based on predictable satellite periodic motion into pre-calculate optimal routing, then dynamically adjusts network topology to recalculate routing according to real-time status of satellite nodes, so that it improves satellite network emergency capability and survivability. In addition to verifying correctness of the protocol on NS3 platform, the routing protocol is implemented on Linux operating system (OS). The solution is useful for solving difficulties of the modules in implementing the function, and the functional test and performance test are carried out on the Linux OS to verify the performance of routing modules. The proposed routing protocol improves the performance of the delay, packet loss rate, and throughput compared to the traditional virtual topology routing algorithm.

Key words: low earth orbit satellites; virtual topology; static routing; dynamic routing; Linux

收稿日期: 2019-06-11
基金项目: 国家自然科学基金项目(61801033)
作者简介: 王 程(1992—), 男, 讲师.
通信作者: 张素兵(1973—), 男, 高级工程师, E-mail: zhangsub@cesi.cn.

卫星通信作为当今世界通信的重要组成部分，具有不受地理和自然条件限制、覆盖范围广、通信距离远、通信容量大和通信质量好等一系列优点[1]，在民用、商用和军用等领域受到广泛关注. 其中，由于低轨(LEO, low earth orbit)卫星系统具有通信时延小、传输损耗低、可实现完整全球连续覆盖、能够提供小型化和多样化终端等优势[2]，在全球范围内得到大力发展. 但是，单颗 LEO 卫星覆盖范围小，用户可视时间短，网络拓扑高度动态变化，节点与链路故障率高，因此 LEO 卫星网络需要稳定、抗毁、鲁棒的路由技术[3]；同时，网络流量分布不均以及用户需求多样性要求 LEO 卫星网络能够采取有效的路由技术，满足不同业务传输的服务质量(QoS, quality of service)需求，提高网络传输效率. 因此，路由技术研究是提高 LEO 卫星网络系统性能的关键环节.

根据卫星网络拓扑结构的不同，路由算法可分为单层、双层、多层卫星网络路由技术. 单层卫星网络路由技术包括基于虚拟拓扑的路由算法、基于覆盖域划分的路由算法、基于数据驱动的路由算法和基于虚拟节点的路由算法. 笔者将针对基于虚拟拓扑的路由算法展开研究.

基于虚拟拓扑算法的基本思想是，利用卫星系统的周期性和星座结构的可预测性，将卫星系统一个周期内的动态拓扑划分为一系列时间片内的静态拓扑，每个时间片内可以采用最短路径算法找到最优路径. 虚拟拓扑的概念最早由 Werner[4] 提出，随后，其他学者及研究人员在此基础上进行了深入研究. 例如，Li 等[5] 提出了基于队列状态的动态路由算法，该算法基于虚拟拓扑预测传播时延并加以动态调整，从而平衡网络负载；Qi 等[6] 利用卫星星座的可预测性构建空间网络，提出消息转发规则，确立适当路由，增加消息传递率并减少端到端的延迟；He 等[7] 利用虚拟拓扑思想建立网络系列时空图，表示沿时间维度的 2 个节点连通性，在此基础上实现了广播路由和调度算法；Jia 等[8] 利用虚拟拓扑方法解决了多个卫星为地面站下载数据过程中时间接触窗口不能充分利用的问题；Tang 等[9] 基于快照拓扑表示，提出一种星间链路重新分配的方法，以优化卫星网络快照的路由性能.

除此之外，针对各类优化目标，已有一些 LEO 卫星网络路由算法的研究成果. 例如，Wang 等[10] 针对我国卫星网络全球布站难度大，数据往往需通过境内信关站进行回传的问题，提出了一种目的节点泛洪的全局—局部算法，通过动态成本估计和概率选路的方法减少路由算法和流量平衡方法导致的不必要消耗；Pan 等[11] 针对卫星网络的周期性拓扑变化，直接利用地面路由协议将产生无休止的路由收敛问题，提出了一种专用于 LEO 卫星网络的路由协议，其拥有一种按需动态路由机制，用于应对链路故障和恢复引起的不规则拓扑变化；Fan 等[12] 针对传统路由信息协议(RIP, routing information protocol)，只考虑链路跳数，难以解决 LEO 卫星网络拓扑结构或全球用户分布不均匀造成的局部网络拥塞问题，提出了一种面向链路拥塞的 RIP，使用链路缓冲区域反馈机制和链路成本机制将产生拥塞链路的流量分配给其他空闲链路，以缓解网络拥塞并降低丢包率；Hussein 等[13] 提出了位置感知负载平衡方案，通过调整一些可调系数，在卫星网络的使用寿命和选择路线传播时延中权衡，保证卫星之间更好地分配业务，确保更好的吞吐量和更低的数据丢失；Li 等[14] 针对 LEO 卫星网络负载不均衡，局部热点地区负载大导致的部分节点拥塞问题，为 LEO 卫星网络设计了一种状态感知和负载平衡的路由模型，考虑了负载变化、连接和节点故障以及恢复等情况，通过对路由表的切换重置操作以及高效的最短路径算法进行动态更新，降低路由开销；Wang 等[15] 针对星间链路负载预测问题，提出了一种基于拥塞预测的负载均衡路由算法，建立了多目标优化模型，采用修正因子来调整路径成本，通过拥塞预测来预测卫星间链路的拥塞. 相较于丰富的理论研究方案，实际实现的方案数量较少.

基于虚拟拓扑的路由算法其优点在于路由协议开销较小，对卫星设备性能要求较低. 但是，预处理的特性也导致此类算法无法应对复杂多变的网络情况，比如节点或链路受损处理和流量负载控制等问题. 因此，笔者提出了一种基于网络状态的 LEO 卫星网络动态路由(NSDR, network state based dynamic routing)算法，通过对邻居节点连接状态定期检测以及节点负载信息交互动态检测网络拓扑，对异常情况(节点断开/恢复、节点拥塞/恢复等)及时感知并相应调整. 针对提出的 LEO 卫星路由算法，除在 NS3 平台进行模拟仿真外，更重要的是，在 Linux 系统中实现了该动态路由协议，验证了在实际网络中该路由算法的可行性，并针对 Linux 实现的难点提

出解决方案,实现了从仿真平台模拟到操作系统应用的跨越.

1 NSDR 协议与模拟

所提出的 NSDR 算法包括基于轨道特性的静态路由以及基于网络状态的动态路由,采用基于虚拟拓扑的路由算法将动态拓扑划分为系列快照,预处理计算每个节点的静态路由,然后在每个快照内根据实时探测的节点连通和负载情况动态更新拓扑表与路由表,得到基于网络状态的最优路径. NSDR 算法框架如图 1 所示.

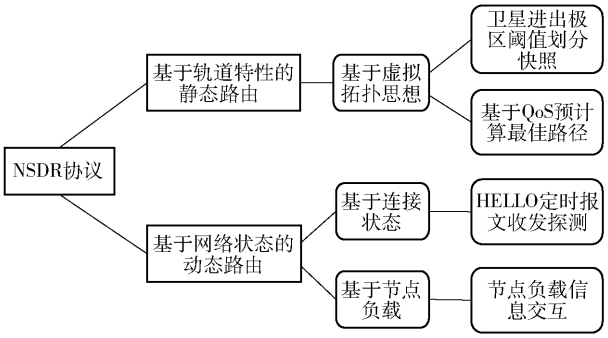


图 1 NSDR 算法框架

1.1 基于轨道特性的静态路由

根据虚拟拓扑路由算法的基本原理,将设定纬度阈值划分极区,在 LEO 卫星运动周期内,一旦有卫星进入或走出极区即产生一个拓扑快照,将卫星周期划分为若干个连贯的快照. 对每个快照内的静态拓扑,采用 Dijkstr 和深度优先搜索算法,计算出本节点到所有其他卫星节点的下一跳路径,以跳数优先选取路径,在跳数相同情况下根据 QoS 需求,考虑链路传播时延作为代价值挑选最优路径. 由此,得到每个快照内当前卫星节点到网络其他节点的初始化路由表,如图 2 所示. 快照切换时,对应调用每个快照内的路由表,减少星上路由开销.

bit0 1	32	63 64	95
自身节点地址	目的节点1地址	下一跳地址	
自身节点地址	目的节点2地址	下一跳地址	
…	…	…	
自身节点地址	目的节点N地址	下一跳地址	

图 2 路由表格式

1.2 基于网络状态的动态路由

LEO 卫星网络运行情况多变,面对突发的链路中断/恢复或负载超出/恢复负荷的情况,预处理得到的静态路由表已无法保证路由的准确性,因此需要增加动态检测机制,应对网络变化情况.

1) 链路中断/恢复,节点超载/恢复

每个节点维护一个邻居表,记录与每个邻居节点的连接与负载状态,如图 3 所示. 邻居节点编号为卫星编号,更新最新消息编号防止过期报文信息失效. 邻居连接状态使用 0 表示断开,1 表示连接;负载状态使用 0 表示超载,1 表示负载情况正常.

bit0 1	2 65	66 67	68 69
邻居节点1编号	邻居节点1最新消息编号	邻居之前连接状态(0/1)	邻居当前连接状态(0/1)
…	…	…	…

图 3 邻居表格式

节点定期获取当前的负载情况,负载情况以 1 s 内链路带宽利用率为衡量标准,依据“平滑加窗法”考虑当前时间点和前一段时间点的负载流量信息计算节点负载,以此来平滑突发数据的峰谷值,使数据更加可靠. 式(1)为根据“平滑加窗法”计算的数据包长度,式(2)为带宽利用率的计算公式.

$$S = 0.7 \times S' + 0.2 \times S'' + 0.1 \times S''' \quad (1)$$

$$Q = S/W \quad (2)$$

其中:S 为“平滑加窗法”后的 1 s 内节点平均转发和发送的数据包长度,S'、S''、S'''为当前 1 s 内、前 1 s 内、前 2 s 内发送和转发的数据包总长度,W 为节点带宽(bit/s),Q 为本地带宽利用率.

将获取的负载信息写入 HELLO 探测报文,定时发送,探测邻居节点连接状态. 若在规定时间内未收到邻居节点回应,则认为邻居节点断开,将邻居表中该邻居节点连接状态置 0;若探测得到回应,则表明为连接,将邻居表中连接状态置 1. 提取接收报文中邻居节点负载状况,判断是否超过设定阈值. 判断标准为:若 $Q \leqslant \alpha\%$,即未超过负载阈值,将连接状态置 1,其余则超过阈值,认为该邻居节点“断开”,连接状态置 0,将阈值设置为 80%.

邻居表中当前状态与之前状态产生差异时触发修改网络拓扑,重新计算路由.

2) 防环路广播机制

由于邻居连接状态以及负载状态为本节点与邻居节点小范围内交互,根据拓扑更新重新计算路由仅局限于下一跳更新. 为防止出现环路情况,需要增加广播机制,当发现邻居表中前后状态不一致需要更新路由表时,将这一变化链路(链路两端节点以及之前/当前状态)信息广播至全网,告知其余节

点同步更新拓扑，重计算路由路径.

1.3　NS3 仿真验证

NS3 是一款面向网络系统的离散事件仿真软件. 下面利用该仿真平台验证 NSDR 协议的正确性，并与基于队列状态的动态路由（QSDR，dynamic routing based on queue state ）算法[5] 以及最短路径优先（SPF，shortest path first）算法进行仿真对比，其中链路传播时延、链路带宽、数据包大小等参数统一设置为表 1 中的值，QSDR 算法的其余参数设置同参考文献[5].

表 1　NS3 仿真参数设置

卫星系统参数	数值	仿真参数	数值
卫星数目	24	链路传播时延/ms	20
轨道面数目	6	链路带宽/（Mbit·s^{-1}）	25
轨道倾角/（°）	86.4	数据包大小/byte	1 024
轨道高度/km	780	队列长度/packet	200
		数据流传输速率/（Mbit·s^{-1}）	0.5 ~ 1

仿真时在 NS3 平台搭建类铱星 LEO 卫星系统，采用极轨轨道，极区纬度阈值为南北纬 70°，卫星数目为 24 颗，6 个轨道面，带宽设置为 30 Mbit/s，其余参数设置如表 1 所示.

图 4 为 LEO 卫星系统二维场景示意图，图中仅标注半球平面视图. 网络中每个节点向全网发送数据流，并接收其余节点发送至本节点的数据流. 改变链路数据传输速率大小，使得数据传输速率逐渐逼近链路带宽，模拟网络正常以及拥塞的情况，测试传输速率不同情况下，网络的传输时延、丢包率以及吞吐量的变化. 由图 5 可知，所提出的 NSDR 算法在虚拟拓扑算法的基础上能够平衡网络负载情况，减小网络时延，降低丢包率，增加网络吞吐量. 此外，NSDR 算法的数据包传输时延低于 QSDR 算法，其原因在于 QSDR 算法在全局链路时延的基础上进行路由选择，换路可能导致较长的传输时延；NSDR 算法的丢包率略高于 QSDR 算法，其原因在于所提出的 NSDR 算法策略考虑一跳范围内的邻居节点负载，有可能在邻居负载均处于较高状态时发生丢包；综合考虑时延和丢包率，NSDR 算法的吞吐量仿真结果略低于 QSDR.

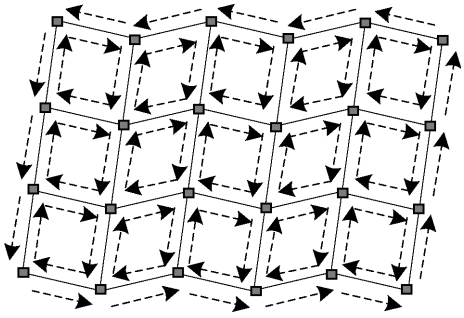
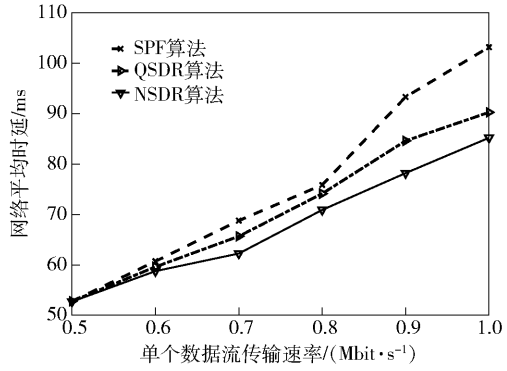
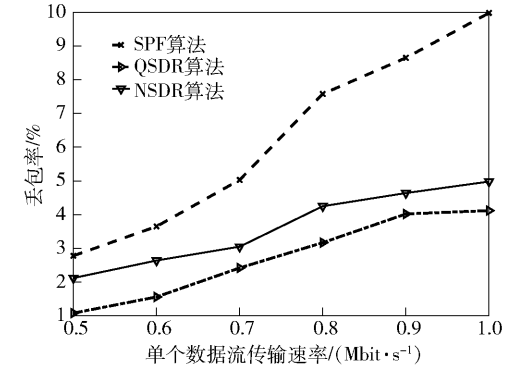


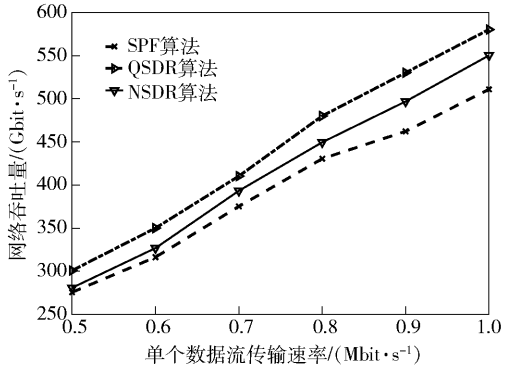
图 4　NS3 仿真场景示意图



(a) 时延



(b) 丢包率



(c) 吞吐量

图 5　NS3 平台仿真验证

2　基于 Linux 系统的路由协议实现

2.1　框架结构

通用的网络协议已经在 Linux 系统的内核空间实现，所以实现路由协议的一种方法是通过修改内核代码使 Linux 操作系统使用设计的路由协议作为

默认的路由协议. 该方法从理论上是可行的,但代价较大[16]. 因此,采用在用户空间实现协议的方法,既可以利用 Linux 系统提供的现有网络功能(TCP/IP),与内核空间相互独立,不会对内核空间产生过强的依赖,也不会影响系统的正常运行.

本方案实现路由协议的主要工作就是根据拓扑信息和链路信息计算出正确的路由路径共享到内核路由表,内核转发模块根据路由表逐跳转发数据包. 基于 Linux 系统的路由协议实现整体框架如图 6 所示.

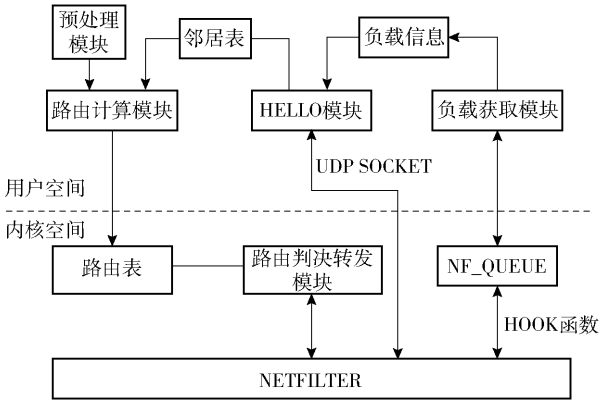


图 6 基于 Linux 系统的路由协议实现框架

预处理模块负责根据卫星系统参数、卫星在轨编号计算快照和拓扑表,预计算出最优路由表存储备用. 路由计算模块检测快照变化和拓扑变化,负责根据拓扑表计算路径,维护路由表. 负载获取模块负责获取本地负载,写入 HELLO 报文. HELLO 模块定时收发 HELLO 报文,探测邻居连接和负载状态,维护邻居表,邻居表前后状态变化触发修改拓扑表.

2.2 关键问题与解决方案

现有的基于 Linux 系统的路由实现方案大多应用于无线自组网,如文献[17-19]中所述,分别实现了自组网络中的 AODV、OLSR、DSR 协议. 应用场景以及协议设计不同,使得 LEO 卫星场景的路由协议在实现机制上将遇到困难. 除此之外,Linux 系统中现有的路由体系结构都是按照有线网路由协议(RIP、OSPF 等主动路由协议)的工作方式实现[20],在实现过程中针对所提出的 NSDR 算法中网络状态探测的需求要做相应调整.

1) 路由表的操作与维护

路由表的更新由快照切换和网络拓扑变化触发,快照切换由定时器计时按快照序号更新,网络拓扑根据网络状态变化,具有随机性. 由于卫星网络链路较长,传输时延较高,在新旧快照切换时,若仍有数据包按照旧快照路径传输,则可能导致数据包传输错误或丢失. 为避免该情况产生,本方案采用双路由表形式,预存先后 2 个快照路由表. 当快照更新定时器触发时,先修改 2 个路由表的优先级,高/低优先级路由表中分别存储当前/上一快照下的路由信息,当传输完成后清空低优先级路由表,替换为下一快照下的路由信息,完成路由表更新.

快照和网络状态检测以及网络拓扑修改的功能均在用户空间实现,而路由表由内核空间维护,因此要实现用户空间操作内核路由表的功能. 采取在内核空间增添双路由表的方式,不修改默认路由,并为双路由表设立优先级 1 和 2,用户空间计算或更新路由路径,利用 ip route add 或 ip route del 对双路由表进行切换和增删改查,这样可以方便用户空间对内核路由表的操作,避免错误修改内核默认路由表,从而提高路由准确性和效率.

2) 节点负载获取

Linux 系统中采用 Netfilter 框架对数据包过滤、处理和转发,负载获取模块利用 Netfilter 实现负载提取并计算,此方法需要考虑用户空间和内核的交互以及数据包处理流程和对应操作.

本方案采用在 NF_IP_POST_ROUTING 挂载点注册自定义的 HOOK 函数,通过 HOOK 函数返回 NF_QUEUE,将数据包排队送入用户空间进行处理,用户空间创建访问队列,设置其数据拷贝内容为数据包长度(bit),记录数据包长度进行负载计算,处理完成后将数据包送回内核,否则数据包可能发生错误或丢失.

3) 邻居节点状态探测

邻居节点状态探测由 HELLO 模块实现,实现过程中需要考虑定时收发 HELLO 报文实现方式、定时发送和超时机制的时间设置、根据连接和负载状态修改邻居表的优先级设置等问题.

定时收发报文,探测邻居节点状态是 2 个节点间网络进程的通信,要求计时回复,在传输必要内容包含节点编号、消息编号、负载大小的前提下,数据包大小尽可能小,进程间操作次数尽可能少,经对比,采用 UDP SOCKET 的通信方式效果最佳,相较于 TCP SOCKET 等其他 SOCKET 方式,流程框架简单. 利用 UDP SOCKET 自定义发送内容按需调整数据包的大小,利用 sleep( ) 函数设定发送间隔,利用

setsocket( )函数设定计时,若 send_timeout 时间内本地客户端未收到邻居服务器端返回的确认消息,即认为节点断开. 由于数据包在星间链路传输时间(距离/传输速度)约 20 ms,HELLO 报文发送间隔设置为 30 ms,超时时间设置为 1 s,稍大于数据传输往返时间.

节点连接/断开以及节点超载/恢复情况导致网络前后状态不一致时,邻居表的变化会触发改变拓扑,然后进行路由重计算. 若 HELLO 报文探测得到 2 个节点连接,但其中某个节点超载,则认为这 2 个节点“断开”,将邻居表置 0,此时需要设置标识防止邻居探测将网络拓扑恢复为 1,而置 1 操作实际只能由负载恢复正常后才能触发.

4) 多任务处理机制

NSDR 算法中每个模块的功能需要同步进行,互不干扰,因此可以采用多线程操作. 对多个线程需要使用的变量,如拓扑表(预处理模块、路由计算模块共用)、负载(负载获取模块、HELLO 模块共用)等设置为全局变量,供需要的模块应用或操作. 在此过程中,要避免线程之间对全局变量同时操作产生错误. 例如,HELLO 模块中 UDP SCOCET 采用客户/服务器模式,客户端接收邻居节点确认消息,判断连接状态,服务器端接收邻居节点主动发送的 HELLO 报文,获取节点负载,判断负载状态,2 个线程可同步修改邻居表,该部分逻辑判断较多,情况较复杂,相较于采用互斥锁的方式多次加锁解锁,增加条件变量判断,容易产生错误,本方案选取设置标识 flag 判断简化流程.

3　基于 Linux 系统的路由协议测试

由于 LEO 卫星星座数量较多,难以实际模拟真实卫星场景,基于 Linux 系统的动态路由协议测试将在小型拓扑中实现,分为路由功能测试和性能测试,以检查路由协议各模块能否正常工作,根据网络情况能否正确计算路由,并在实现功能的基础上对路由协议性能进行评估.

测试设置 5 个网络节点,即采用 5 台笔记本电脑,节点编号为 0 ~ 4,组成一个小型的模拟卫星网络,包括轨道内链路和轨道间链路. 为避免外部网络环境变化造成影响,将每个节点的网络带宽设置为上行 0.8 Mbit/s,下行 10 Mbit/s,利用 Ubuntu 系统基于 Linux 内核平台运行路由协议.

3.1　功能测试

NSDR 算法基于 LEO 星座的拓扑结构设计,事先依据纬度阈值划分星座周期内的快照,记录完整周期内各个快照的拓扑,实时计算序号调用对应快照拓扑图. 在实现和测试时针对小范围拓扑,自定义了 2 个快照内 5 个节点的拓扑结构,存储在用户空间. 功能测试分为以下 3 部分.

1) 网络正常运行测试

各个节点处于正常状态(相邻节点连通且无过载)时,验证各节点连通、邻居状态获取、路由表更新 3 个功能. 通过 ping 命令检查网络中各个节点的连通性;通过查看邻居表信息,检查邻居连通状态及负载信息是否成功获取;通过查看内核路由表,检查路由模块的路由信息写入是否正常. 如图 7(a)所示,节点 1 在网络正常情况下能够正确计算得到本节点到全网其余节点的路由表,其中第 1 个 IP 地址为目的节点地址,第 2 个 IP 地址为下一跳 IP 地址.

2) 网络节点断开/恢复连接运行测试

网络中存在单个或者多个邻居节点断开/恢复连接时,路由模块应及时发现并更新拓扑和邻居表,触发重新计算路由. 如图 7(b)(c)所示,验证得到断开和恢复 2/3 个邻居节点的情况下,NSDR 算法能够正确更新计算路由.

3) 网络节点拥塞/恢复正常运行测试

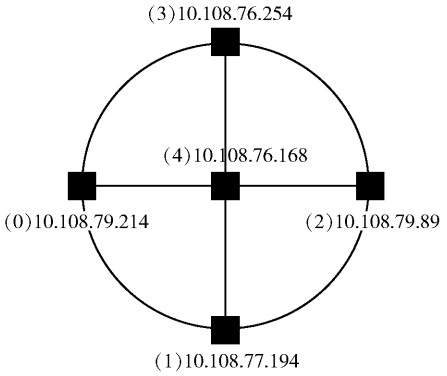
网络中某节点发现邻居节点拥塞(节点负载超过带宽的 80%)时,即认为到该邻居节点的链路断开,此时需要更新邻居表,重新计算路由. 本方案中对于节点拥塞的处理机制与节点断开相同,因此仅验证某节点在 1 个邻居节点拥塞后又恢复的情况. 如图 7(d)所示,拥塞触发路由重计算功能能够正常运行.

测试结果表明,在 Linux 系统实现的动态路由协议能够正确计算路由,并且能够基于时间动态切换快照,针对突发网络状况修改拓扑计算路由.

3.2　性能测试

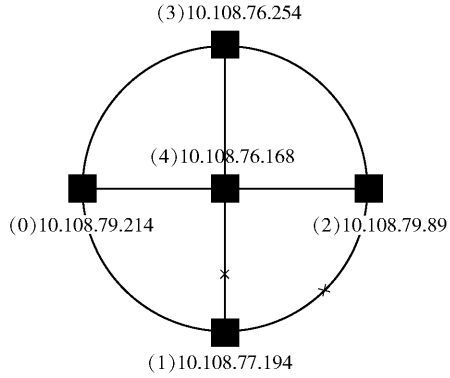
下面针对基于 Linux 系统实现的动态路由协议测试时延、丢包率和吞吐量 3 个方面的性能.

测试时,限制 Linux 系统带宽为 800 kbit/s,利用客户/服务器模式自动发送 2 000 个 UDP 包,每隔 10 ms 发送 1 个,传输速率变化范围为 760 ~ 800 kbit/s,对发送/接收数据包的时间计时,按照式(3) ~式(5)计算得到端到端时延 T、丢包率 D 和吞



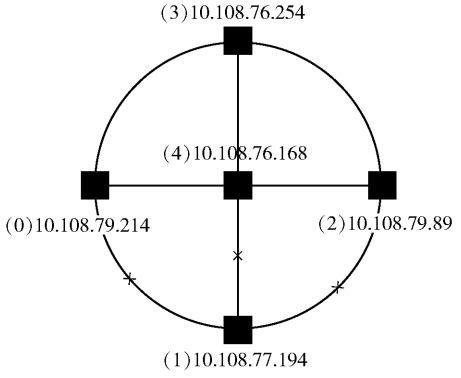

```
huiwen@huiwen-virtual-machine:~$ ip route list table 1
10.108.76.168 via 10.108.76.168 dev ens33
10.108.76.254 via 10.108.76.168 dev ens33
10.108.77.194 via 10.108.77.194 dev ens33
10.108.79.89 via 10.108.79.89 dev ens33
10.108.79.214 via 10.108.79.214 dev ens33
```

(a) 网络正常时节点1的路由表




```
huiwen@huiwen-virtual-machine:~$ ip route list table 1
10.108.76.168 via 10.108.79.214 dev ens33
10.108.76.254 via 10.108.79.214 dev ens33
10.108.77.194 via 10.108.77.194 dev ens33
10.108.79.89 via 10.108.79.214 dev ens33
10.108.79.214 via 10.108.79.214 dev ens33
huiwen@huiwen-virtual-machine:~$ ip route list table 1
10.108.76.168 via 10.108.76.168 dev ens33
10.108.76.254 via 10.108.76.168 dev ens33
10.108.77.194 via 10.108.77.194 dev ens33
10.108.79.89 via 10.108.79.89 dev ens33
10.108.79.214 via 10.108.79.214 dev ens33
```

(b) 节点1断开/恢复2个邻居时的路由表




```
huiwen@huiwen-virtual-machine:~$ ip route list table 1
10.108.77.194 via 10.108.77.194 dev ens33
huiwen@huiwen-virtual-machine:~$ ip route list table 1
10.108.76.168 via 10.108.76.168 dev ens33
10.108.76.254 via 10.108.76.168 dev ens33
10.108.77.194 via 10.108.77.194 dev ens33
10.108.79.89 via 10.108.79.89 dev ens33
10.108.79.214 via 10.108.79.214 dev ens33
```

(c) 节点1断开/恢复3个邻居时的路由表

```
检测到4号节点发生拥塞
拥塞节点ip地址为10.108.76.168
拥塞节点当前负载为82893byte
占带宽百分比为80.95%
```

```
检测到4号节点恢复正常
拥塞节点ip地址为10.108.76.168
拥塞节点当前负载为78848byte
占带宽百分比为77%
```

```
huiwen@huiwen-virtual-machine:~$ ip route list table 1
10.108.76.168 via 10.108.79.89 dev ens33
10.108.76.254 via 10.108.79.214 dev ens33
10.108.77.194 via 10.108.77.194 dev ens33
10.108.79.89 via 10.108.79.89 dev ens33
10.108.79.214 via 10.108.79.214 dev ens33
huiwen@huiwen-virtual-machine:~$ ip route list table 1
10.108.76.168 via 10.108.76.168 dev ens33
10.108.76.254 via 10.108.76.168 dev ens33
10.108.77.194 via 10.108.77.194 dev ens33
10.108.79.89 via 10.108.79.89 dev ens33
10.108.79.214 via 10.108.79.214 dev ens33
```

(d) 节点4拥塞时节点1的路由表

图7 各种网络状态下节点1的路由表

吐量 O. 其中,Δt_i 为每个接收到的数据包传输时间;N_I 和 N_A 分别为理想/实际接收的数据包数量;R_N在数值上等于 N_A,按照实际接收数据包顺序进行编号;R_T 为接收到的数据包总字节数;t_L 和 t_F 表示

最后一个数据包接收时间和第 1 个数据包发送时间.

$$T = \frac{\sum_{i=R_N} \Delta t_i}{N_A} \tag{3}$$

$$D = \frac{N_I - N_A}{N_I} \tag{4}$$

$$O = \frac{R_T \times 8/1\,000}{t_L - t_F} \tag{5}$$

测试时,保证虚拟拓扑策略和动态路由算法仅在策略上设置不同,其余参数相同. 值得注意的是,这里得到的测试结果是基于 Linux 系统在测试网络环境下的性能,与 NS3 模拟的卫星网络场景不同,两者结果不能进行横向比较.

所实现的 NSDR 算法在链路断开时能够及时感知并切换链路. 在 Linux 平台进行的数据传输实验结果可得,链路断开时,SPF 算法无法进行链路切换,数据包全部丢失,而 NSDR 算法能够在 35 ms 左右的时间内进行收敛,重新切换路径,减少丢包,恢复传输.

由图 8 可知,NSDR 算法相较于传统的 SPF 算法可以在网络负载增大时平衡网络负载,减小时延,降低丢包率,增加吞吐量. 这是由于随着每个节点的总传输速率逐渐增大且接近带宽时,会出现数据包处理不及时,产生排队时延的情况,队列缓冲区没有足够的空间而导致链路拥塞和数据包丢失. 相较于传统的 SPF 算法,NSDR 算法能够在节点产生拥塞之前进行分流处理,降低网络时延,减少网络丢包,从而提高网络吞吐量,在整体性能上有所提升.

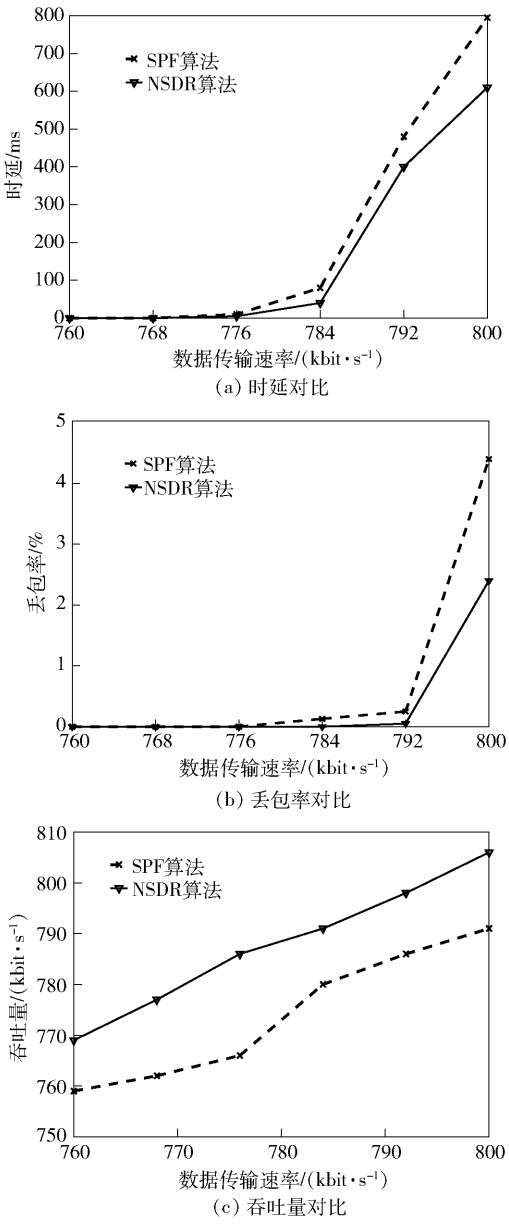


图 8　动态路由协议性能测试

4　结束语

所提出的 NSDR 算法基于虚拟拓扑的基本思想,实现了静态路由与动态路由的结合,利用卫星网络的规律性和周期性预计算网络拓扑和最优路由,减小卫星开销,在此基础上也能够应对网络节点突发状况,使卫星网络路由具有抗毁性. 除 NS3 验证路由协议的正确性之外,在 Linux 系统的实现与测试也验证了本协议实施的可行性. 针对在 Linux 系统实现过程中的关键问题(路由表的操作与维护、节点负载获取、邻居节点状态探测以及多任务处理机制),介绍了解决方案,为卫星网络路由协议的实现提供参考. 在 Linux 系统中对实现的路由协议进行的功能测试和性能测试结果表明,该路由协议能够完整地实现预设功能,且具有良好的性能,在时延、丢包率以及吞吐量方面相较于传统基于虚拟拓扑的算法有所提升.

参考文献:

[1] 王汝传. 卫星通信网路由技术及其模拟[M]. 北京: 人民邮电出版社, 2010: 8-9.

[2] 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.

[3] 刘子鸾. 卫星网络路由与流量控制关键技术研究[D]. 北京: 北京邮电大学, 2018.

[4] Werner M. A dynamic routing concept for ATM-based

satellite personal communication networks[J]. IEEE Journal on Selected Areas in Communications, 1997, 15(8): 1636-1648.

[5] Li Hezhong, Zhang Heteng, Qiao Liang, et al. Queue state based dynamical routing for non-geostationary satellite networks[C]// 2018 IEEE 32nd International Conference on Advanced Information Networking and Applications. Krakow: IEEE, 2018: 1-8.

[6] Qi Weijing, Hou Weigang, Guo Liang, et al. A unified routing framework for integrated space/air information networks[J]. IEEE Access, 2016, 4: 7084-7103.

[7] He Feng, Liu Qin, Lü Tao, et al. Delay-bounded and minimal transmission broadcast in LEO satellite networks [C]// IEEE International Conference on Communications. Kuala Lumpur: IEEE, 2016: 1-7.

[8] Jia Xiaohua, Lü Tao, He Feng, et al. Collaborative data downloading by using inter-satellite links in LEO satellite networks[J]. IEEE Transactions on Wireless Communications, 2017, 16(3): 1523-1532.

[9] Tang Zhu, Feng Zhenqian, Han Wei, et al. Improving the snapshot routing performance through reassigning the inter-satellite links [C]// 2015 IEEE Conference on Computer Communications Workshops. Hong Kong: IEEE, 2015: 97-98.

[10] Wang Yichen, Zhang Xuejun, Zhang Tao. A flooding-based routing algorithm for Ads-B packets transmission in LEO satellite network[C]// 2019 Integrated Communications, Navigation and Surveillance Conference (ICNS). Herndon: IEEE, 2019: 1-9.

[11] Pan Tian, Huang Tao, Li Xingchen, et al. OPSPF: orbit prediction shortest path first routing for resilient LEO satellite networks[C]// 2019 IEEE International Conference on Communications (ICC). Shanghai: IEEE, 2019: 1-6.

[12] Fan Weiqiang, Zhang Tao. A link congestion oriented LEO network routing protocol[C]// 2018 IEEE 18th International Conference on Communication Technology (ICCT). Chongqing: IEEE, 2018: 46-50.

[13] Hussein M, Abu-Issa A, Elayyan I. Location-aware load balancing routing protocol for LEO satellite networks [C]// 2018 International Conference on Advanced Communication Technologies and Networking (CommNet). Marrakech: IEEE, 2018: 1-7.

[14] Li Xu, Tang Feilong, Chen Long, et al. A state-aware and load-balanced routing model for LEO satellite networks[C]// 2017 IEEE Global Communications Conference. Singapore: IEEE, 2017: 1-6.

[15] Wang Houtian, Wen Guoli, Liu Naijin, et al. A load balanced routing algorithm based on congestion prediction for LEO satellite networks[J]. Cluster Computing, 2019, 22(4): 8025-8033. .

[16] 严雯. Linux 系统下 OLSR 路由协议研究及实现[D]. 成都: 电子科技大学, 2009.

[17] 谢世欢. Linux 系统上 AODV 路由协议的实现[D]. 成都: 电子科技大学, 2004.

[18] 陈炼. 基于 Linux 系统的 OLSR 路由协议研究与实现[D]. 重庆: 重庆邮电大学, 2017.

[19] 周帆. 动态路由协议(DSR)在 Linux 下的实现[D]. 西安: 西安电子科技大学, 2006.

[20] 谢世欢, 郭伟. 实现 Ad-hoc 按需路由协议的关键技术[J]. 计算机应用, 2006, 26(3): 517-518.
Xie Shihuan, Guo Wei. Key techniques on implementation of Ad-hoc on-demand routing protocols [J]. Journal of Computer Applications, 2006, 26(3): 517-518.