

低轨卫星网络中高效资源利用的组播算法

程连贞 刘 凯 张 军

(北京航空航天大学新航行系统民航重点实验室 北京 100083)

摘 要 为了解决低轨卫星网络中现有典型源组播算法的信道资源浪费问题,提出了一套单核共享树组播算法,即核心群合并共享树(CCST)和加权 CCST(w -CCST)算法. CCST 算法包括动态近似中心(DAC)选核方法和核心群合并组播路径构建方法. DAC 方法根据组成员在网络中的分布情况自适应选择最优核;在核心群合并方法中,以核节点作为初始核心群,通过核心群和剩余组成员的最短路径方法逐步扩展直至整棵组播树构建完成,从而使得组播树的树代价最小,大大提高了网络的传输带宽利用率和传输效率. 在 w -CCST 算法中,可以通过调整加权因子来适度增大树代价、降低端到端传播时延以满足某些端到端时延要求苛刻的实时组播业务. 最后,通过仿真与其它算法进行了性能对比,仿真结果说明 CCST 组播树的平均树代价比其它组播树显著降低,平均端到端传播时延比其它组播树稍高; w -CCST 算法的平均端到端传播时延性能好于 CCST 算法,树代价性能稍差,说明使用加权因子可以在组播树的树代价和端到端传播时延性能之间作折中.

关键词 卫星 IP 网络;低轨;组播;共享树;选核

中图法分类号 TP393

Multicast Routing Algorithms with High Bandwidth Efficiency for LEO Satellite Networks

CHENG Lian-Zhen LIU Kai ZHANG Jun

(CNS/ATM Laboratory of CAAC, Beihang University, Beijing 100083)

Abstract To resolve the channel resources waste problem of the typical source-based multicast routing algorithm in low earth orbit (LEO) satellite IP networks, this paper proposes a new core-based shared tree algorithm called the core-cluster combination shared tree (CCST) algorithm and its improved version (i. e. the w -CCST algorithm). The CCST algorithm consists of the dynamic approximate center (DAC) core selection method and the core-cluster combination multicast route construction method. The DAC method can adaptively select the optimal core node according to group distribution in the network. And the core-cluster combination method takes core node and its nearest group member in hops as initial core-cluster, and expands it to construct entire multicast tree with the lowest tree cost stepwise by a shortest path scheme between newly-generated core-cluster and surplus group members, which can greatly improve transport bandwidth utilization and transport efficiency. In the w -CCST algorithm, a weighted factor is used to make tradeoff between tree cost and end-to-end propagation delay. Therefore, tree cost can be increased a bit and meanwhile end-to-end propagation delay is decreased slightly to meet strict end-to-end delay requirements of some real-time multicast applications by adjusting the weighted factor. Performance of the CCST and w -CCST algorithms is compared with several

收稿日期:2006-03-24;修改稿收到日期:2007-03-04. 本课题得到国家自然科学基金(60532030,10577005)和航天科技创新基金资助.

程连贞,女,1978年生,博士研究生,主要研究方向为移动通信网络的路由协议和信道接入协议. E-mail: clz_tea@sina.com. 刘 凯,男,1973年生,博士,副教授,主要研究方向为无线个人通信、移动 Ad Hoc 网络、无线接入、无线传感器网络和天空地一体化网络. 张 军,男,1965年生,博士,教授,博士生导师,主要研究领域为航空电信网、天空地一体化网络、空中交通管理和卫星通信导航等.

other algorithms. And simulation results show that average tree cost of the CCST multicast tree is greatly lower than that of multicast tree of other algorithms, and its average end-to-end propagation delay is a bit higher than that of others, while average end-to-end propagation delay of the ω -CCST multicast tree is lower than that of the CCST multicast tree.

Keywords satellite networks; LEO; multicast; shared tree; core selection

1 引言

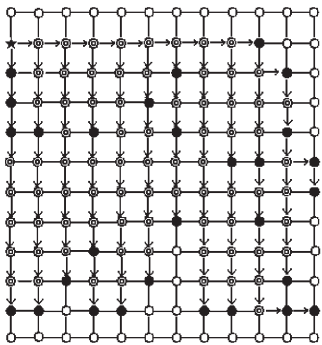
卫星网络覆盖范围广和下行链路的广播特性非常适合组播通信. 与高轨(GEO)和中轨(MEO)卫星网络相比,低轨(LEO)卫星网络的信息传输时延很小,适合转发实时多媒体业务^[1-2],不过星上资源稀少的特点使得其应用受到限制.

组播技术有效地解决了单点发送、多点多跳接收和多点发送、多点多跳接收的问题,即源节点如何尽最大可能沿公共共享路径向多个目的节点同时发送相同的信息数据,这些数据只在分支节点处被复制,在每条链路上只转发一次,从而最大程度地利用了网络资源. 组播算法通常分为源(source-based)组播算法和共享树(Shared Tree, ST)组播算法两类,其中源算法生成的源组播树是以源节点为根的最短路径树,时延性能较好;共享树算法中,以组播组中某(几)个成员节点为根(称为核),生成一棵整个组播组共享的组播树,所得代价性能较好,时延性能不如源组播树^[3]. 如果某节点属于网络内 L 个组播组,每个组有 M_i 个源,源算法使用源地址 S_i 和组地址 G (即 (S_i, G))标识组播会话,组播路由表需要为每组中每个源生成和存储 1 条记录(entry),所以此节点在组播路由表中需要至少 $\sum_{i=1}^L M_i$ 条记录;共享树组播算法使用 $(*, G)$ 标识组播会话,每个组播组只需 1 条记录,所以此节点只需在组播路由表中保存 L 条记录. 由此可知,与源组播算法相比,共享树算法所需存储空间小,可扩展性(scalability)好.

地面网络中,传统的源组播算法如距离向量组播路由协议(DVMRP)、开放式组播最短路径优先(MOSPF)等在报文周期性广播时需要消耗大量的网络传输带宽资源;共享树组播算法如核基树(CBT)、稀疏模式的协议无关组播(PIM-SM)等在高动态网络中需要节点之间频繁交互信息以便及时、准确地获得当前的连接情况,所以会产生很大的交换开销. 由于卫星网络的星上资源和处理能力极其有限,不

适合使用上述算法,并且卫星运动具有可预见性、周期性和规则性等特点,都使得卫星网络需要专用、简单、高效的组播算法. 目前,已提出的适于低轨卫星网络的组播算法主要有 MRA(Multicast Routing Algorithm)^[4]、RST(Rectilinear Steiner Trees)^[2]等. 其中 MRA 属于源算法,旨在使端到端传播时延最小;RST 算法使用整数线性规划方法为组播组在网络中寻找最短树长 RST 树,以构建总带宽使用最少的组播树.

组播的基本问题是找到一棵代价低的组播树,即如何尽量共享传输路径. 在 MRA 中,源和中间卫星节点使用数据报路由算法(DRA)^[4]逐跳分布式计算当前节点到每个目的节点可能的下一跳路由方向并标记为高、低优先级(high- and low-priority),然后将下一跳路由方向相同的目的节点合并为一个节点子集,以能够覆盖所有目的节点为目标,一般可得到几种子集组合,选取节点子集数最少的那个组合,并依据下一跳路由方向连接其子集节点至下一节点. 如此,目的节点集合逐渐被分化为子集,直至所有子集为空,则整棵组播树构建完成. 然而,由于 MRA 只利用下一跳路由方向信息合并路径,这些信息随组播路径的扩展逐渐分散,导致构建的组播路径也相应逐步分散,使得组播树的树代价很高,浪费了大量的网络资源. 定义树代价为组播树的根(源或核)到目的节点的所有链路代价的总和,若每条链路的代价是 1,树代价即为组播树上总链路数,它反



★ 源节点; ● 组播成员节点; ⊙ 中间卫星节点; ○ 一般卫星节点

图 1 MRA 组播树实例

映了对网络信道资源的占用情况. 图 1 是一个颇具代表性的 MRA 组播树实例, 一个 30 节点组播组均匀分布在网络中, 采用 12×12 的栅格结构表示简化的低轨卫星网络拓扑, 即 144 颗卫星、264 条链路, 星际链路长度视实际卫星网络初始对齐时刻情况而设置^[1,4]. 可以看出, 图中的 MRA 组播树占用了 77 个中间节点、106 条链路, 即占用了网络中大部分的中间节点和链路, 很多链路并没有被树上节点共享.

为了解决 MRA 的资源浪费问题, 本文提出了核心群合并共享树(Core-cluster Combination Shared Tree, CCST)算法, 包括动态近似中心(Dynamic Approximate Center, DAC)选核方法和核心群合并组播路径构建方法. DAC 方法首先基于逻辑位置形成的虚拟、静态的网络拓扑动态选择组播组的中心位置, 然后以最靠近中心位置的成员节点作为核节点. 在核心群合并方法中, 以核节点作为初始核心群, 通过核心群和剩余组成员的最短路径方法逐步扩展直至整棵组播树构建完成, 旨在使树代价最小, 提高信道资源利用率和传输效率, 同时增强算法可扩展性. 另外, 为了支持某些对端到端传播时延要求严格的组播业务, 我们提出了加权核心群合并共享树(the weighted CCST, 简称 w -CCST)算法, 以在树代价和端到端传播时延性能之间作折中.

本文第 2 节介绍低轨卫星网络的星座模型; 第 3 节详细描述 CCST 算法, 主要介绍 DAC 动态核位置选择方法和核心群合并组播路径构建机制; 第 4 节详细介绍对核心群合并方法进行了改进的 w -CCST 算法; 第 5 节用仿真评估了算法性能; 最后总结全文并给出相关结论.

2 低轨卫星网络的星座模型

本文假设低轨卫星网络由 P 个轨道组成, 每个轨道上有 S 颗卫星, 是 $P \times S/P/0^\circ$ 的极轨道星座类型, 并且每颗卫星与相邻卫星之间有 4 条星际链路(ISLs)相互连接, 其中 2 条是轨间星际链路, 另外 2 条是轨内星际链路, 不使用逆向轨间星际链路. 所有轨内星际链路的长度相等且固定不变, 轨间星际链路的长度随卫星地理位置的改变而变化. 另外, 由于极轨道星座在两极附近的卫星过于密集, 常常关闭一些轨间星际链路, 假定所连接的两颗卫星的星下点纬度都大于 75° 时, 关闭该轨间星际链路.

(w -)CCST 算法基于逻辑位置概念、使用 DRA 算法^[4]计算两节点之间的端到端传播时延最短路径

以构建组播树. 假设网络内的卫星是初始对齐的, 在某个初始时刻将整个星座的轨道按照一定的次序编号, 使用 p 标识轨道编号 ($0 \leq p \leq P-1$); 并将轨道内的卫星依次编号, 使用 s 标识卫星编号 ($0 \leq s \leq S-1$), 则 (p, s) 可以表示一个卫星节点, 以它们当前的地理位置为中心, 按照覆盖范围将整个星座覆盖区划分为逻辑位置, 此时得到的低轨卫星网络拓扑如图 2 所示. 以静态的逻辑位置作为一个虚拟卫星节点, 则形成稳定、规则的栅格状卫星网络拓扑结构. 在路由计算时, 将逻辑位置作为跳, 则星上路由表只与静态的逻辑位置相关, 不涉及卫星高速运动导致的网络拓扑快速变化对路由计算产生的复杂影响, 屏蔽了卫星的移动性, 简化了星上路由. 卫星移出逻辑位置时必须移交路由表, 移出的卫星发送它的路由表到轨道内移进的后续卫星, 并从轨道内前序卫星接收新的路由表.

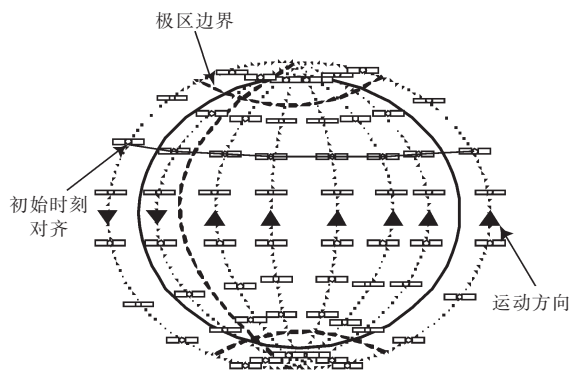


图 2 低轨卫星网络的星座模型

假设低轨卫星网络中某组播组共有 N 个成员节点, 记为 $G = \{g_0, g_1, g_2, \dots, g_{N-1}\}$, 其中 g_i ($i = 0, 1, \dots, N-1$) 代表成员节点的逻辑位置, 且 $g_i = (p_i, s_i)$.

3 CCST 组播算法

CCST 组播算法首先利用组成员的逻辑位置信息选择核位置, 然后使用核心群合并方法构建低树代价的组播树, 并且支持动态组成员关系的维护和更新.

3.1 核位置选择

在核共享树组播算法中, 由于核位置对组播路由的性能有很大影响, 所以核位置选择非常重要^[5-8]. 随着组成员的动态加入和退出以及节点移动导致的网络拓扑变化, 原本最优的核树可能会退化为随机核树^[7], 需要重新选择核位置并重建组播

路径,从而产生额外的网络开销并使得时延性能恶化,因此尽可能减少核移植和重建组播树的操作对于组播路由性能的持续优化至关重要。

3.1.1 相关工作

在文献[5]中,选核方法被分为4类,分别是随机(random)、基于拓扑(topology-based)、基于组(group-based)和基于性能的(performance-based)方案。随机方案从网络中随机选取一个节点作为核,相比其他方法,所得平均时延较大;基于组的方案基于网络拓扑和组成员位置选核;基于拓扑的方案基于大致的网络拓扑特征选核,其重新计算拓扑的时间粒度(time scale)相对较粗,对于高速移动的低轨卫星网络,很难获得其当前的、精确的拓扑结构,所以不适合使用这种方案;基于性能的方案从一个节点集合(接收节点、源节点或所有节点)中选择以某些性能度量综合加权后具有最优值的节点为核,这种方法过于复杂,不适合用于星上资源极其受限的低轨卫星网络中。综上可知,在这些方法中,基于组的方案最适于低轨卫星网络。

人们已经给出一些基于组的选核方法^[6-9],这些方法都是选取靠近组中心(group center)的节点作为特定中心组播树^[9]的根,特定中心组播树的总带宽使用性能居于最小 Steiner 树(以端到端时延为代价、最小化总带宽使用的最小生成树)和源树之间^[10],相比最小 Steiner 树,具有时延有界和易于实现的优点^[7-8]。文献[6-7]研究了选取中心位置的启发式算法。文献[8]提出的直径中心(Diameter Core, DC)方法将一个组播组中两成员节点之间最长的路径定义为直径,以靠近直径中心的成员节点作为核节点。文献[9]提出的最大中心树(Maximum-Centered Tree, MCT)方法的选核原则是核到其它成员节点的最大距离最小;平均中心树(Average-Centered Tree, ACT)方法按照核到所有组成员的平均距离最小的原则选核;直径中心树(Diameter-Centered Tree, DCT)方法以每个成员节点到两个最远成员节点的距离之和为直径,将最小直径的中点作为核节点。然而,上述方案需要频繁计算成员节点之间的链路距离,尤其当组播组数目较大时,需要进行大量的星上计算,这对于资源稀少的卫星是很重的负担,所以它们不适用于低轨卫星网络。

3.1.2 DAC 方法选核

设组 G 的中心位置的逻辑位置是组 G 中成员逻辑位置的平均值,记为 g_a , $g_a = (p_a, s_a)$, 其中 p_a ,

s_a 可由下式求得

$$p_a = \frac{1}{N} \left(\sum_{i=0}^{N-1} p_i \right) \quad (1)$$

$$s_a = \frac{1}{N} \left(\sum_{i=0}^{N-1} s_i \right) \quad (2)$$

g_a 与组 G 中成员节点之间距离的最小值记为 d_{\min} , 可从下式求得:

$$d_{\min} = \min\{d_i | d_i = \sqrt{(p_i - p_a)^2 + (s_i - s_a)^2}\} \quad (3)$$

式(3)中 $i=0, 1, 2, \dots, N-1$ 。将与 g_a 距离最近的逻辑位置记为 $g_u = (p_u, s_u)$, 其中 p_u, s_u 满足下式:

$$d_{\min} = \min\{(p_a - p_u)^2 + (s_a - s_u)^2\}^{\frac{1}{2}} \quad (4)$$

若式(4)有唯一解,则以其作为核位置,即 $g_c = g_u$; 若有多个解,则随机选取其中之一作为核位置。将核位置记为 g_c , $g_c = (p_c, s_c)$ 。

DAC 选核方法不需要复杂的星上计算,可以自适应选择最优的核节点,并且由于采用了逻辑位置形成的虚拟静态网络拓扑,只因组成员关系的动态变化触发核移植,会降低核移植频率,从而节省宝贵的星上资源。

3.2 核心群合并方法构建组播树

假设已知组播组 G 的核节点 g_c , 为了易于表达后续内容,假定 $g_c = g_0$, 即 $G = \{g_c, g_1, g_2, \dots, g_{N-1}\}$ 。定义 $G_0 = \{g_1, g_2, \dots, g_{N-1}\}$ 。

3.2.1 相关概念和规则

(1) 最短路径

定义两节点间的各条路径中所经跳数最少的路径为最短路径。

(2) 最近距离路径

定义两节点间的各条路径中长度最短的路径为最近距离路径。

(3) 最近距离最短路径

如果两节点间的最短路径不止一个,则将长度最短的最短路径称为最近距离最短路径。

(4) 组播路径决策方法

对于点到多点的通信,假设需要构建网络中某节点 n_0 到节点集 H 的组播路径,首先计算 n_0 与 H 中各节点之间的最近距离最短路径并比较,选择其中的最近距离最短路径作为组播路径;然后将其上的全部节点构成一个节点集 K 。 H 除去 K 中节点后剩余的节点构成节点集 H' , K 到 H' 的组播路径选取方法如下:计算 K 中每个节点与 H' 各节点之间的最近距离最短路径并比较,选择其中的最近距离最短路径作为组播路径。

(5) 核心群(core-cluster)

在构建组播路径的过程中,将核节点和已生成组播路径上的中间节点、目的节点共同组成的节点集称为核心群,它的节点数随着组播路径的扩展不断增多.定义初始核心群 C_0 只包括核节点 g_c ,即 $C_0 = \{g_c\}$.

3.2.2 核心群合并方法

图 3 给出了核心群合并方法构建组播树的流程图.下面详细介绍构建过程:

1. 使用 DRA 计算集合 G_0 与初始核心群 C_0 之间的最近距离最短路径,得到属于 G_0 的目的成员节点 g'_1 和中间节点(记为 g'_{1a}, g'_{1b}, \dots),以它作为连接 g_c 和 g'_1 的组播路径,将 g'_1 和中间节点与 C_0 合并,得到新的核心群 $C_1 = \{g_c, g'_1, g'_{1a}, g'_{1b}, \dots\}$.令 $C'_0 = \{g_c, g'_1\}$, $i = 1$.

2. $G_i = G_{i-1} - C'_{i-1}$, C'_{i-1} 为 C_i 与 G 的交集.核心群 C_{i+1} 的具体产生方法是:计算 G_i 与 C_i 之间的最近距离最短路径,得到属于 G_i 的目的成员节点 g'_{i+1} 和中间节点(记为 $g'_{(i+1)a}, g'_{(i+1)b}, \dots$),以它作为新的组播路径,将 g'_{i+1} 、中间节点与核心群 C_i 合并后,得到 $C_{i+1} = \{g_c, g'_1, g'_{1a}, g'_{1b}, \dots, g'_{i+1}, g'_{(i+1)a}, g'_{(i+1)b}, \dots\}$.

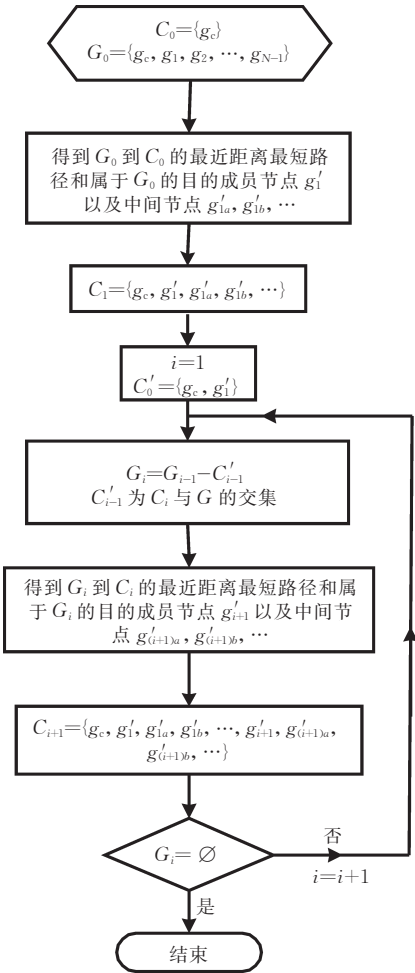


图 3 核心群合并方法构建组播树流程图

3. 令 $i=i+1$,继续步 2 直到 $G_i = \emptyset$,即组播组 G 内的成员节点全部合并到核心群内为止,此时整棵组播树全部构建完成.

3.3 动态组成员关系维护

定义下述 3 种操作以支持组成员的随机加入和退出.

成员加入.要加入某个组播组的卫星节点向所有邻节点广播发送包含组播会话 ID 的“加入请求”,如果邻节点当前不在此组播组的组播树上,则向其邻节点继续广播,直到到达这棵组播树的某个树上节点,随后沿组播树直接发送到核节点.核节点回复“加入确认”,并以当前组播树的树上节点集作为核心群,使用核心群合并方法计算新加入节点到核心群的最近距离最短路径作为新的组播路径.

成员退出.如果某个节点要退出组播组,它直接向核节点发送“退出请求”,核节点回复“退出确认”.如果退出成员是叶子节点,它和上游分支点之间的冗余路径及相关信息将被删除;如果是分支节点,它的信息将从成员列表中删除,但它仍需继续维护原有组播路径并转发组播分组.如果退出成员是核节点,并且组播组的安全性较高,则需进行核移植;否则仍可继续作为核节点,直到成员加入/退出次数达到门限值 δ 才触发核移植.

组播树更新.每次组成员加入或退出,核节点维护的成员更新计数器(Member Updating Counter, MUC)就加 1,直到计数器值超过 δ .此时触发组播树更新操作,使用 DAC 方法重新选择核节点并移交组信息到新核,然后重新构建组播树的组播路径.

4 w-CCST 组播算法

由于核心群合并方法的宗旨是使树上节点尽可能地共享组播路径,导致核与附近某些节点之间的组播路径可能要绕过一些不必要的中间节点,使得端到端传播时延增大.为了减轻乃至避免这种现象,考虑引入加权机制,在计算组播路径时使用加权因子 $w(0 \leq w \leq 1)$, w 值大则树代价性能相对较好, w 值小则端到端传播时延性能相对较好.

已知集合 G_i 中某节点 g_i ,假设依据最近距离最短路径原则,得到 g_i 到 C_i 的最近距离最短路径的链路数 l_i 以及 g_i 到 g_c 的组播路径链路数 l'_i ,可知 l_i 反映了构建新的组播路径时树代价的增长情况, l'_i 反映了核与目的节点之间的端到端传播时延性能.令 $W_i = l_i \times w + l'_i \times (1 - w)$,即 W_i 是 l_i 和 l'_i 以 w 加权

的代数和。

图 4 给出了使用加权核心群合并方法构建组播树的流程图,详细步骤如下。

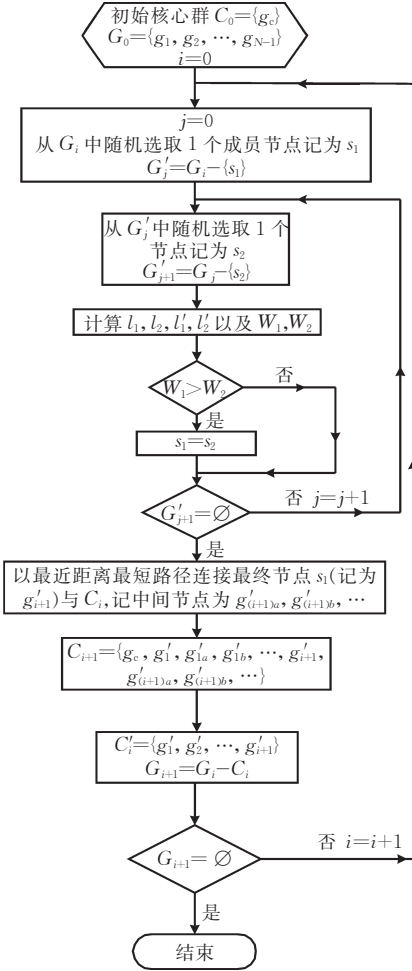


图 4 加权核心群合并方法构建组播树流程图

1. 初始化: 初始核心群 $C_0 = \{g_c\}$, $G_0 = G - C_0$, $i = 0$.
2. $j = 0$. 从集合 G_i 中随机选取一个成员节点记为 s_1 , $G'_i = G_i - \{s_1\}$.
3. 从集合 G'_i 中随机选取一个节点作为 s_2 , $G'_{i+1} = G'_i - \{s_2\}$, 计算将 s_1 和 s_2 分别以最近距离最短路径连接到 C_i 的组播路径增加的链路数 l_1 和 l_2 以及沿已有组播路径到 g_c 的链路数 l'_1 和 l'_2 , 然后根据下式计算 W_1 和 W_2 :

$$W_1 = l_1 \times w + l'_1 \times (1 - w) \quad (5)$$

$$W_2 = l_2 \times w + l'_2 \times (1 - w) \quad (6)$$

4. 比较 W_1, W_2 的大小, 若 $W_1 > W_2$, 令 $s_1 = s_2$; 否则不作任何处理. 然后转到步 5.

5. 判断 G'_{j+1} 是否为空集, 如果不为空, 则令 $j = j + 1$, 继续执行步 2 和步 3; 否则将最终得到的 s_1 节点(记为 g'_{i+1})以最近距离最短路径连接到核心群 C_i , 并将 s_1 和此路径上的中间节点(记为 $g'_{(i+1)a}, g'_{(i+1)b}, \dots$)与核心群 C_i 合并为新核心群 C_{i+1} , 即 $C_{i+1} = \{g_c, g'_1, g'_{1a}, g'_{1b}, \dots, g'_{i+1}, g'_{(i+1)a}, g'_{(i+1)b}, \dots\}$. $G_{i+1} = G_i - C'_i$, C'_i 为 C_{i+1} 中属于组 G 的非核成

员节点集合, 即 $C'_i = \{g'_1, g'_2, \dots, g'_{(i+1)}\}$.

6. 令 $i = i + 1$, 依次执行步 2~5, 直到集合 $G_i = \emptyset$ 为止, 此时整棵组播树全部构成。

5 性能仿真和评估

5.1 组播树实例和分析

定义使用 DAC 选核方法和 MRA 组播路径构建机制的扩展组播算法称为有核 MRA(Core-based MRA, 简称 CMRA), 对于同一组播组, CMRA 和 (w -)CCST 算法选择的核位置相同. 定义 DCT 选核方法^[9]和核心群合并组播路径构建机制组成的扩展组播算法为基于核心群的 DCT(Core-cluster-based DCT, 简称 CDCT)算法. 为了评估加权核心群合并方法的性能, 分别取加权因子 w 为 0.8 和 0.4, 将相应的 w -CCST 算法记为 w_1 -CCST 和 w_2 -CCST. 表 1 给出了本文用到的所有算法及其构成。

表 1 相关对比算法一览表

算法	选核方法	组播路径构建方法
MRA	无	MRA
CMRA	DAC	MRA
CCST	DAC	核心群合并
w_1 -CCST	DAC	加权核心群合并
w_2 -CCST	DAC	加权核心群合并
CDCT	DCT	核心群合并

图 5~图 8 是某个 30 节点均匀分布的组播组分别使用 CMRA、CCST、 w_1 -CCST 和 w_2 -CCST 算法生成的组播树实例. 表 2 以实际数据对上述实例的中间节点数和树代价性能作了对比, 可以看出 CCST 和 w_1 -CCST 的树代价和中间节点数最少, 可大大节约网络信道资源; 其中树代价最高、占用中间节点数最多的是 MRA, 其次是 CMRA; w_2 -CCST 的路径分布不如 CCST 和 w_1 -CCST 紧凑, 但明显好于 MRA 和 CMRA.

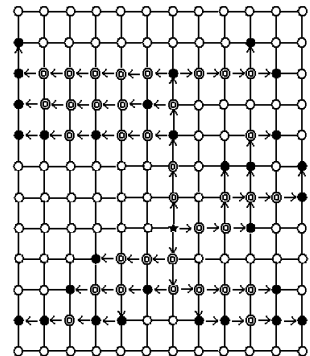


图 5 CMRA 组播树实例图

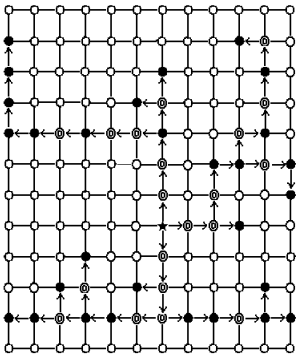


图 6 CCST 组播树实例图

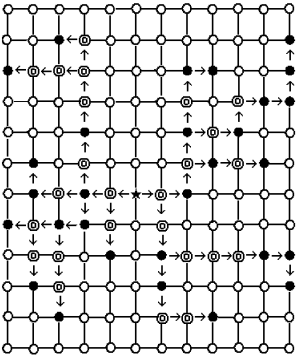


图 7 w_1 -CCST 组播树实例

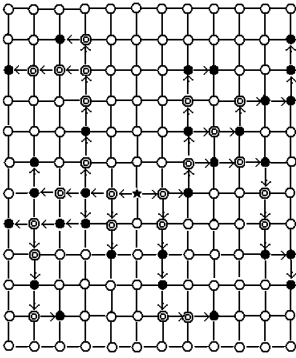


图 8 w_2 -CCST 组播树实例

表 2 相关组播树实例性能对比

算法	对应实例图	中间节点数	树代价
CCST	图 6	20	49
w_1 -CCST	图 7	20	49
w_2 -CCST	图 8	25	54
CMRA	图 5	35	64
MRA	图 1	77	106

对于均匀分布的同一组播组,DAC 与 DCT 方法选择的核位置比较接近,均位于组地理中心附近;CDCT 与 CCST 算法使用相同的路径构建机制,所得组播树形状和性能也颇为接近.对于非均匀分布的同一组播组,DAC 方法选择的核位置靠近组成员分布相对密集的区域,DCT 方法只依据成员节点之间的最大距离选核,不考虑组分布密度,所以两者所选核位置可能不同,此时的性能差异将在后续仿真

中进行考察.

5.2 仿真环境

假设低轨卫星网络由 12 个轨道组成,每个轨道上均匀分布 24 颗卫星.为了验证和评估 CCST 和 w -CCST 算法的性能,采用下述 3 种对比方案:(1) $(w-)$ CCST 算法与 MRA 对比;(2) $(w-)$ CCST 算法与 CMRA 对比;(3) CCST 与 CDCT 算法对比.表 3 给出了上述方案的详细对比.对于方案 1 和 2,采用均匀分布的组播组足够体现算法之间的性能差异,仿真的组成员数目从 8~30.方案 3 采用非均匀组播组,组成员以 1:5 的比例集中分布于网络中两个对角的 3×3 和 6×6 区域内,组成员数目从 6~36.

表 3 仿真对比方案

算法 1	算法 2	组分布情况	对比	仿真结果
$(w-)$ CCST	MRA	均匀,组成员数目 8~30	单核共享树与源组播树	图 8,图 9
$(w-)$ CCST	CMRA	均匀,组成员数目 8~30	选路方法(核心群合并与 MRA)	图 10,图 11
CCST	CDCT	非均匀,组成员数目 6~36	选核方法(DAC 与 DCT)	图 12

5.3 评估指标

定义以下 4 种归一化参数以有效评估算法性能:树代价的比值、树长(组播树上所有链路长度的总和)的比值、组播树上中间节点数的比值以及所有成员节点作为源时源到目的节点的平均端到端传播时延比值,分别记为 h, l, n 和 d ,其中 $d = \frac{1}{N} \sum_{i=0}^N d_i$, d_i 等于组中每个源节点到所有目的节点的端到端传播时延的平均值.

5.4 仿真结果

图 9 和图 10 给出了上述评估指标的仿真结果.其中 MRA 与 CCST、 w_1 -CCST、 w_2 -CCST 组播树的平均树代价比值 h 、树长比值 l 和中间节点数比值 n 基本上均大于 1,且随着组成员数目的增长呈线性增加趋势,说明 CCST、 w_1 -CCST、 w_2 -CCST 算法的树代价、树长和中间节点数性能显著优于 MRA,甚至 MRA 与 CCST 的平均中间节点数比值最高达 2.25. w_1 -CCST 算法的参数 h, l 和 n 的性能比较接近 CCST,明显好于 w_2 -CCST 算法;不过从图 10 可以看出,在所有的算法中,MRA 的平均端到端传播时延性能最好,CCST 算法的最差,即 CCST 算法以平均端到端传播时延为代价换取了最低树代价,同理,MRA 牺牲树代价获得了低时延;另外, w_2 -CCST 算法的平均端到端传播时延低于 w_1 -

CCST,从而验证了加权因子 ω 可以灵活调整 ω -CCST 算法的树代价和端到端传播时延性能。

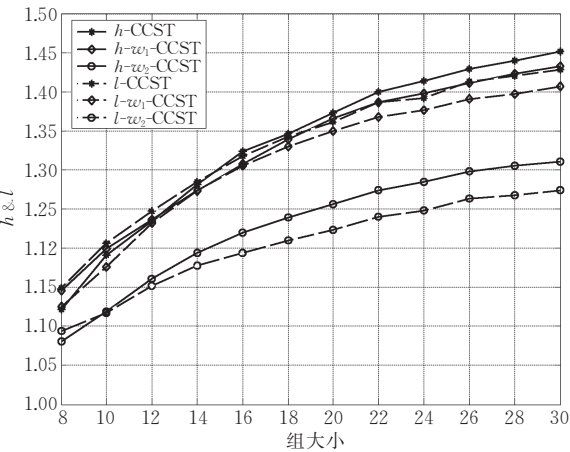


图 9 MRA 与 (ω) -CCST 树代价和树长性能对比

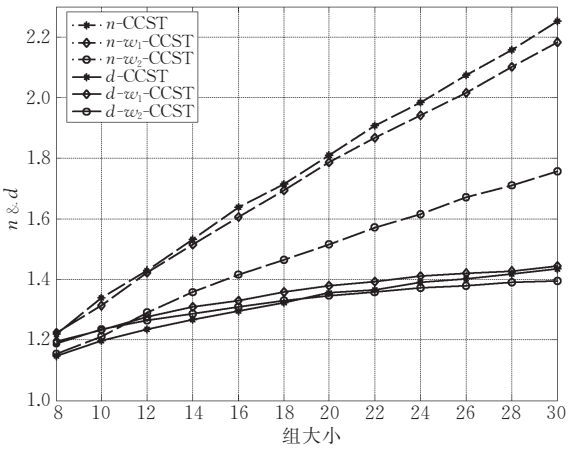


图 10 MRA 与 (ω) -CCST 中间节点数和端到端传播时延性能对比

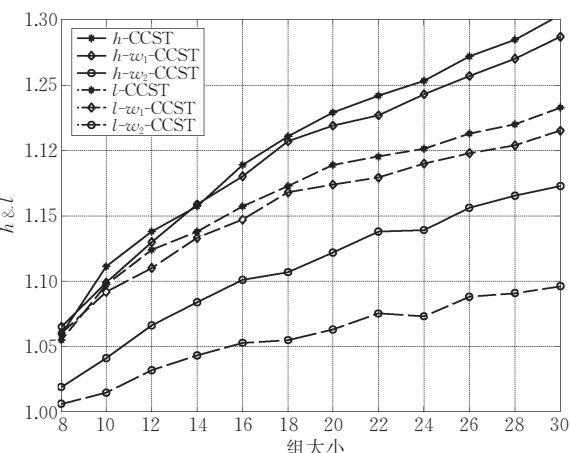


图 11 CMRA 与 (ω) -CCST 树代价和树长性能对比

从图 11 和图 12 可以看出,在树代价、树长和中间节点数性能方面,CCST 好于 ω -CCST 和 CMRA;在平均端到端传播时延性能方面,CMRA 最好。

ω -CCST 算法的树代价、树长和中间节点数性能也好于 CMRA,但端到端传播时延性能稍差。不过,由于 CMRA 和 CCST、 ω -CCST 均为共享树算法,因此它比 MRA 更接近这两种算法的性能。

从图 13 可以看出,非均匀组分布时,CDCT 和 CCST 算法的性能相近,它们的 h, l, n 性能对比曲线均基本稍高于 $y=1$, 所得 d 值大于 1, 说明 CCST 算法的性能略好于 CDCT, 可知 DAC 核位置方法略好于 DCT。

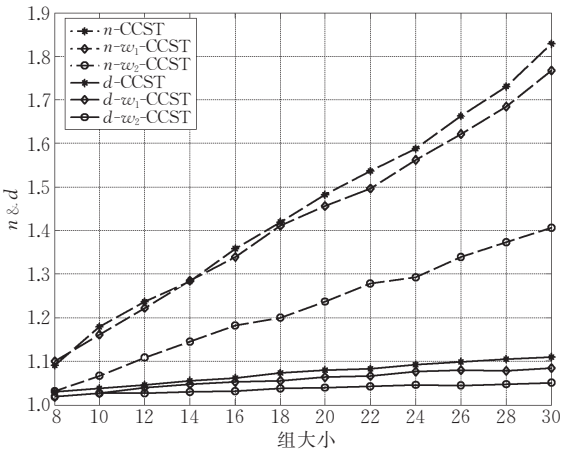


图 12 CMRA 与 (ω) -CCST 中间节点数和端到端传播时延性能对比

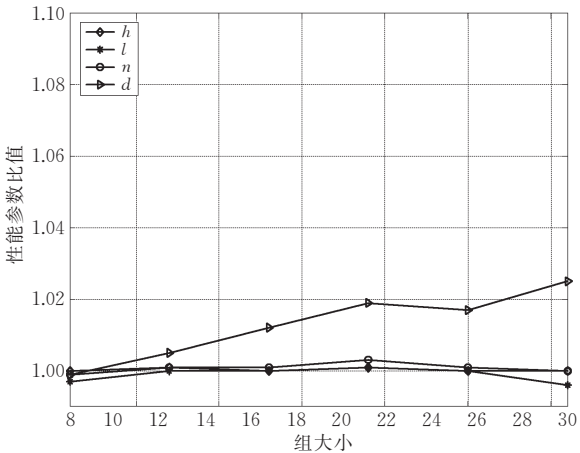


图 13 CDCT 与 CCST 算法的性能对比

综上所述,CCST 算法的树代价、树长尤其中间节点数性能比其它算法有显著改善,端到端传播时延性能稍差; ω -CCST 算法的加权因子可以在组播树的树代价和端到端传播时延性能之间作折中。

6 结 论

星上资源稀少的特性是低轨卫星网络支持多媒体组播应用时必须面临的挑战之一,因此星上资源

的高效利用非常重要. 组播技术提供了一种有效的解决手段, 因此低轨卫星网络的组播算法研究颇具价值. 为解决现有低轨卫星网络中典型源组播算法的信道资源浪费问题, 本文提出了一种单核共享树组播算法即 CCST 算法, 可以使所得树代价最小; 并在此基础上提出了 w -CCST 算法, 以支持某些有严格端到端时延要求的实时组播业务. 仿真结果说明, CCST 算法的树代价性能较其它算法有显著改善, 可以大大节省网络传输带宽资源, 提高传输效率, 不足的是端到端传播时延略高. 另外, w -CCST 算法可以在树代价和端到端传播时延性能之间作折中.

参 考 文 献

- [1] Ekici E, Akyildiz I F, Bender M D. A multicast routing algorithm for LEO satellite IP networks. *IEEE/ACM Transactions on Networking*, 2002, 10(2): 183-192
- [2] Yang De Nian, Liao Wanjiun. On multicast routing using rectilinear Steiner trees for LEO satellite networks//*Proceedings of the IEEE Globecom 2004*. Dallas Texas, 2004: 2712-2716
- [3] Sahasrabudhe L H, Mukherjee B. Multicast routing algo-

rithms and protocols: A tutorial. *IEEE Network*, 2000, 14(1): 90-104

- [4] Ekici E, Akyildiz I F, Bender M D. A distributed routing algorithm for datagram traffic in LEO satellite networks. *IEEE/ACM Transactions on Networking*, 2001, 9(2): 137-147
- [5] Calvert K L, Zegura E W, Donahoo M J. Core selection methods for multicast routing//*Proceedings of the International Conference on Computer Communications Networks*. Las Vegas, Nevada, 1995: 638-642
- [6] Ali Shahzad, Khokhar Ashfaq. Distributed center location algorithm. *IEEE Journal on Selected Areas in Communications*, 1997, 15(3): 291-303
- [7] Thaler David, Ravishankar C V. Distributed center location algorithms: Proposals and comparisons//*Proceedings of the INFOCOM'96*. San Francisco, CA, 1996: 75-84
- [8] Raghavendra A D, Rai S, Iyengar S S. Multicast routing in internetworks using dynamic core based trees//*Proceedings of the 1996 IEEE 15th Annual International Phoenix Conference*. Scottsdale, 1996: 232-238
- [9] Wall David W. Mechanisms for broadcast and selective broadcast[Ph. D. dissertation]. Stanford University, 1980
- [10] Wei L, Estrin D. A comparison of multicast trees and algorithms. Computer Science Department, University Southern California; Technical Report USC-CS-93-560, 1993



CHENG Lian-Zhen, born in 1978, Ph. D. candidate. Her current research interests include routing protocols and multiple access protocols of wireless networks.

LIU Kai, born in 1973, Ph. D., associate professor. His research interests include wireless personal communications, mobile satellite networks, wireless access, WLAN,

wireless sensor networks and mobile ad hoc networks, especially in the area of distributed network architecture, wireless multiple access, QoS routing and communication protocol design.

ZHANG Jun, born in 1965, Ph. D., professor, Ph. D. supervisor. His research interests include the Integration of space, air and ground information networks, aeronautical telecommunication network, satellite navigation system in the aviation surveillance and new navigation system.

Background

Satellite networks are suitable to forward multicast traffics due to broadcast capability and world-wide coverage. Compared to geosynchronous earth orbit (GEO) and medium earth orbit (MEO) satellite networks, low earth orbit (LEO) satellite networks have much lower end-to-end propagation delay, so that they can support IP-based real-time multicast services, such as video conferences. However, they have to face the challenge of scarce onboard resources (bandwidth, memory and processing capability etc.).

Multicast routing is an important technique with bandwidth-efficiency. It disseminates packets from a source to all members of a multicast group. Instead of sending packets separately to each individual group member, a source just transmits one copy to its downstream branch node(s) along multicast tree, and every intermediate node and destination group member node receive the packet only from their upstream branch nodes, which results in that packets are transferred only once at each link of a multicast tree, and thus

transmission bandwidth on entire delivery path can be greatly reused and saved.

Multicast routing algorithms are used to determine multicast tree connecting source(s) and member nodes of a multicast group. And they are classified into source-based scheme and shared tree scheme. Typical source-based algorithms, such as DVMRP and MOSPF, are only appropriate to be applied in networks with a static or slowly changing network topology. And for typical shared tree algorithms such as CBT and PIM-SM, frequent information exchange will be induced by rapid movement of nodes and thus consume large amounts of bandwidth resources in a dynamic network. Therefore they are not suitable for satellite IP networks due to rapid satellite mobility and scarce onboard resources. Multicast routing algorithm(MRA) is a typical multicast routing algorithm for LEO satellite IP networks, while the created multicast trees have considerably high tree cost and consume large amounts of network transmission resources because only one-hop local information (i.e., next-hop directions) of current on-tree nodes to destinations are

employed to merge routes by it.

To resolve the channel resources waste problem of the MRA, a novel single-core shared tree algorithm called core-cluster combination-based shared tree(CCST) is proposed in this paper, which includes dynamic approximate center (DAC) core selection method and core-cluster combination multicast route construction scheme. And its improved version called the weighted CCST (w -CCST) algorithm is presented to make tradeoff between performance of tree cost and that of end-to-end propagation delay to support some real time multicast applications with strict end-to-end delay requirements.

This work is a part of projects of the National Natural Science Foundation of China under grant No. 60532030 and 10577005, and the Innovation Foundation of Aerospace Science, and Technology of China. These projects aim to develop fundamental theories and key technologies of the Integration of space, air and ground information networks. And satellite networks are indispensable integrant of it. The author's research area covers routing protocols of satellite networks.