ISSN 1000-9825, CODEN RUXUEW
Journal of Software, Vol.21, No.9, September 2010, pp.2289−2303
doi: 10.3724/SP.J.1001.2010.03684

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel/Fax: +86-10-62562563

传感器网络中基于树的最大生命精确数据收集*

梁俊斌[1,2], 王建新[1+], 李陶深[2], 陈建二[1]

[1](中南大学 信息科学与工程学院,湖南 长沙 410083)

[2](广西大学 计算机与电子信息学院,广西 南宁 530004)

Maximum Lifetime Algorithm for Precise Data Gathering Based on Tree in Wireless Sensor Networks

LIANG Jun-Bin[1,2], WANG Jian-Xin[1+], LI Tao-Shen[2], CHEN Jian-Er[1]

[1](School of Information Science and Engineering, Central South University, Changsha 410083, China)

[2](School of Computer and Electronics Information, Guangxi University, Nanning 530004, China)

+ Corresponding author: E-mail: jxwang@mail.csu.edu.cn

Liang JB, Wang JX, Li TS, Chen JE. Maximum lifetime algorithm for precise data gathering based on tree in wireless sensor networks. *Journal of Software*, 2010,21(9):2289−2303. http://www.jos.org.cn/1000-9825/3684.htm

Abstract: In multi-hop wireless sensor networks that contain a high density of nodes, precise data gathering makes nodes that are close to the sink incur a heavier workload, which depletes their energy faster and can easily cause a "hot spot" that would shorten the network lifetime. The problem of constructing a tree that has a maximum lifespan is NP-complete. An algorithm called MAXLAT can be used to solve this problem without the need for the location of nodes. MAXLAT starts from a tree whose root has the largest number of children. The nodes in the tree are classified into three subsets that go accordingly to their respective loads: bottleneck nodes, sub-bottleneck nodes, and rich nodes. Next, the MAXLAT continues to transfer descendants of high-load nodes to sub-trees of low-load nodes by coloring. When MAXLAT is terminated, it constructs a tree in which "bottleneck nodes" carry a lighter load. Simulation results show that the tree achieved by MAXLAT has a longer lifetime than trees created by previous algorithms.

Key words: wireless sensor network; data gathering; maximum lifetime; spanning tree

摘 要: 在节点密集部署的多跳传感器网络中,精确数据收集使得越靠近Sink节点的传感器节点需要承担越多的数据转发量,能量消耗很快,容易造成"热区",缩短了网络生命周期.为了最大化网络生命周期,需要构造生命周期最大的生成树,但这属于 NP 完全问题.无须知道节点的位置信息,提出一种算法 MAXLAT 来解决这个问题.算法以一棵 Sink 拥有最多孩子的生成树为基础,并根据节点负载的大小将树上节点分别定义为瓶颈节点、次瓶颈节点和富裕节点.然后,通过对所有节点进行着色,不断转移瓶颈节点的子孙,到富裕节点的子树上去.算法结束时,得到一棵

* Supported by the National Natural Science Foundation of China under Grant Nos.60873265, 60873188, 60903222 (国家自然科学基金); the National Basic Research Program of China under Grant No.2008CB317107 (国家重点基础研究发展计划(973)); the Program for Changjiang Scholars and Innovative Research Team in University of China under Grant No.IRT0661 (长江学者与创新团队发展计划)

Received 2008-08-01; Revised 2009-01-14; Accepted 2009-07-06

“瓶颈节点”负载较轻的生成树.实验结果表明,与目前已有算法相比,MAXLAT 构造的树具有更长的生命周期.

关键词: 无线传感器网络;数据收集;最大化生命周期;生成树

中图法分类号: TP393 文献标识码: A

无线传感器网络(wireless sensor network,简称 WSN)综合了传感器技术、嵌入式计算技术、分布式信息处理技术和通信技术,能够协作地实时监测、感知和采集网络分布区域内的各种环境或监测对象的信息,并对这些信息进行处理,从而获得详尽准确的信息,传送到需要这些信息的用户[1].无线传感器节点具有数量庞大、体积微小、通信和计算能力弱、电源能量有限且无法补充、能量异构等特点[2].

在实际应用(如环境监测、战场监视等)中,每个传感器会周期性地感知一定量的数据,并通过多跳路由传送到一个远方的 Sink 节点,称为数据收集(data gathering).在实际应用中,常见的数据收集具有以下两种模式[3]:

(1) 相关数据收集(correlated data gathering).每个传感器感知到的数据具有一定的相关性,则节点可以将接收到的数据和自己感知到的数据进行聚合,并以固定大小的数据包进行发送.在这种模式下,每个节点无论收到多少数据包,其发送出的数据大小是一样的,即数据发送独立于数据接收;

(2) 精确数据收集(precise data gathering).每个传感器感知到的数据相关性不强,则节点会接收长度不同的数据包,连同自己感知到的数据,只以低比例聚合或不聚合.在这种模式下,每个节点发送的数据与接收到的数据加上自己感知到的数据,在长度上呈线性关系.在精确数据收集中,越靠近 Sink 节点的中继节点要承担越多的转发数据量,能量消耗很快,容易造成“热区”,缩短网络生命周期[4].

为了最大化网络生命周期,需要构造生命周期最大的生成树.本文提出一种在多跳传感器网络中,适用于精确数据收集的构造最大化生命周期树算法 MAXLAT(maximum lifetime precise data gathering algorithm based on tree).MAXLAT 无须知道节点的位置信息,只需要了解节点的邻居信息.它首先构造一棵 Sink 拥有最多孩子的生成树,然后通过着色不断将这棵树上负载重的节点子孙转移到负载轻的节点的子树上去.最后构造出的生成树,瓶颈节点负载较轻.仿真实验表明,与目前已有的算法相比,MAXLAT 能构造生命周期更长的生成树.

本文第 1 节主要介绍数据收集协议的研究现状和相关工作.第 2 节是网络模型和问题描述.第 3 节阐述 MAXLAT 算法并进行分析.第 4 节进行模拟实验,对算法进行验证.第 5 节是总结.

1 相关工作

目前有很多工作对数据收集进行研究,根据网络中节点间的协作关系,主要分为 3 类:

(1) 基于簇的数据收集协议(cluster-based data gathering protocol)

基于簇的数据收集协议从所有节点中选择一部分作为簇首,其他节点选择最近的簇首加入它,从而形成一种 Voronoi 结构的网络.簇首负责收集簇内节点收集上来的数据,直接或多跳地传送到 Sink 节点.比较典型的基于簇的数据收集协议有 LEACH[5],HEED[6],EEUC[7],CEDA[8]等.基于簇的数据收集协议便于实现和管理,但普遍存在簇首分布不均匀、簇首负载过重、簇规模难以控制等缺点.

(2) 基于链的数据收集协议(chain-based data gathering protocol)

基于链的数据收集协议将网络中所有的节点构成一条链,并在链上选择一个节点作为头节点与 Sink 节点直接通信,链两端数据沿链传输到头结点.比较典型的基于链的数据收集协议有 PEGASIS[9],CCS[10],DRAEM[11]等.基于链的数据收集协议的优点是减少了分簇算法在簇重构过程中所产生的开销,节点采用小功率与最近邻居节点通信,从而降低了能量的消耗.但是,链中距离较远的节点会引起较大的数据延迟,而且唯一的头节点会使得它会成为严重的瓶颈.

(3) 基于树的数据收集协议(tree-based data gathering protocol)

基于树的数据收集协议将网络中所有节点组成一棵生成树,每个节点接收其孩子节点发送来的数据,连同自己感知到的数据一起发送给父节点.由于树结构是支持网络连通性的最小图结构[12],因此,基于树的数据收集协议能够有效保证网络的连通性和可靠性,具有保证 QoS、容易实现高效的能量管理等优点,是目前研究的热

点.比较典型的基于树的数据收集协议有 PEDAP(power efficient data gathering and aggregation protocol)和 PEDAP-PA[13],MLDGA(maximum lifetime data gathering algorithm)[14],MNL(maximum network lifetime)[15], EESR(energy efficient spanning tree)[16],IAA(iterative approximation algorithm)[17]等,他们都属于集中式算法,其中:PEDAP 和 PEDAP-PA,MLDGA,MNL 采用贪婪启发式算法构造生成树,既可以处理相关数据收集,也可以处理精确数据收集;而 IAA 采用近似算法构造生成树,只能处理相关数据收集.

PEDAP 首先构造一棵只有 Sink 节点的树,然后将树外节点和树上节点间的通信代价作为权值,每次挑选一个节点加入当前树,直至所有节点加入树.PEDAP 能够均衡网络中总的能量耗费,延长网络中最后一个死亡节点的生命周期.但是,它没有考虑节点的剩余能量,容易导致能量少的节点死亡.PEDAP-PA 考虑了节点剩余能量,能做到能量感知,延长了第 1 个死亡节点的生命周期.由于 PEDAP 和 PEDAP-PA 采用贪婪启发式策略,在构建生成树时,每个节点均作为叶子加入,并不知道自己和父节点在最终生成树上的子孙数量,这造成最终生成树上各子树的节点数不均衡.长时间使用一棵生成树,会使子孙较多的瓶颈节点能量消耗过快,缩短网络生命周期.因此,算法规定固定 100 轮重新计算并更换树的结构.显然,这个规定过于经验化,并不合理.

MLDGA 对 PEDAP-PA 进行了改进,在权值设计上考虑了当前树的生命周期最大化和节点间的通信代价.但是与 PEDAP-PA 一样,其最终生成树也不均衡.为了减少“瓶颈节点”的能量消耗,采用动态路由方案,即每轮数据收集都使用不同的树,但频繁地更换树结构会增加网络中相应的通信和计算开销.

MNL 是针对查询驱动(query-driven)的数据收集协议,它以节点的最小剩余能量最大化为目标设计权值,构造节点剩余能量最大的生成树.当查询一个接一个地到达时,每次根据网络中各节点的能量状况构造一棵生成树,这相当于动态路由.MNL 构造树的思想与 MLDGA,PEDAP-PA 相近,其生成树并不均衡.

EESR 也是对 PEDAP-PA 进行改进,但其主要研究合适的更换树结构的频率.它指出,在网络中更换树结构是需要额外通信代价的,更换的频率过高或过低都会影响网络生命周期.最后,它通过实验分析得到结论,如果生成树上节点的最大负载越小,则更换树结构的频率可以越低,以节省额外费用.

IAA 针对相关数据收集,研究如何构造近似最优的最大化生命周期数据收集树.算法从一棵任意树开始,采用近似算法对度比较大的节点进行调整,最后得到一棵能量均衡的最小度生成树.但是,它无法用于精确数据收集以解决“热区”问题.

本文借鉴 IAA 的部分分析思想,针对精确数据收集,提出一种新的基于着色的方法构造瓶颈节点负载较轻的生成树.网络采用这种生成树进行数据收集,能有效缓解“热区”问题,延长网络生命周期.

2 系统模型和问题描述

第 1 节有些工作,如 LEACH,PEGASIS,PEDAP-AP 等,都假设网络是单跳的(single-hop network),即每个节点都能与网络中的任何节点(包括 Sink 节点)直接通信,这样的好处是能够有效均衡网络的能量消耗.但是在实际应用中,网络区域广大,能量有限的节点不太可能与任何节点直接通信[18].节点发送的数据包,一般要经过若干中继节点,多跳地传送到 Sink 节点,这就是多跳网络(multi-hop network).鉴于实际应用中的需要,我们研究的对象是多跳的无线传感器网络.

2.1 网络模型

n 个传感器节点随机地分布在一个面积为 $M\times M$ 平方米的正方形区域 A 内,存在 1 个 Sink 节点.整个传感器网络组成一个连通的无向图 $G(V,E)$,其中:V 是节点集合,$V=\{v_0,v_1,v_2,\ldots,v_n\}$,$|V|=n+1$.$v_0$ 是 Sink 节点,$v_1,v_2,\ldots,v_n$ 是传感器节点;E 是 G 中边的集合.如果两个传感器节点 v_i 和 v_j 相互处于对方的通信半径内,则边$(v_i,v_j)\in E$.$|E|=m$ 为边的数量.网络具有如下性质[19]:

(1) 网络是连通的静态网络,节点部署后不再移动;

(2) 节点的类型可以有多种,节点的初始能量是异构的,而且不能补充.

为了简化问题和便于讨论,假设节点采用固定发射和接收功率,发射 1 bit 数据需要的能量是 E_t,接收 1 bit

数据需要的能量是 E_r[17,20].假定网络中有其他一些拥塞控制策略,以避免数据在传输过程中发生拥塞和重传.

2.2 相关定义

本文主要研究精确数据收集问题,不失一般性,假设中继节点不对数据进行聚合.对于一棵树 T,树上的每个节点周期性地产生 l bit 数据,连同孩子节点发送给自己的数据,全部发送给父节点[20].为了描述方便,我们首先给出以下定义:

定义 1(轮(round)). 这是从所有传感器节点收集一次数据并传送到 Sink 节点的过程,不管它持续的时间是多少[13].

定义 2. 假设一个节点 v_i 在树 T 上有 t 个孩子,他们各自发送来长度为 $l_1,l_2,\ldots,l_t$ 的数据,v_i 自己感知到的数据长度为 l bit,则 v_i 在一轮中接收的总数据量为 $\left(\sum_{i=1}^{t} l_i\right)=lS(T,v_i)$,而 v_i 发送的数据量具有函数关系:

$$f(l_1,l_2,\ldots,l_t,l)=l(S(T,v_i)+1),$$

其中,$S(T,v_i)$是节点 v_i 在树 T 上子孙的数量,$i=0,\ldots,n$.如果节点是叶子节点,则 $f(l)=l$.

定义 3. 对于一个节点 v_i,需要接收所有孩子节点发送来的数据,然后连同自己感知到的数据一起发送给父节点.由于节点感知和计算能耗可忽略不计,因此,一个节点在一轮中的能量耗费为

$$C(T,v_i)=E_r\left(\sum_{i=1}^{t} l_i\right)+f(l_1,l_2,\ldots,l_t,l)E_t=lS(T,v_i)E_r+l(S(T,v)+1)E_t=lS(T,v_i)(E_r+E_t)+lE_t.$$

定义 4(节点的生命周期(node lifetime)). 节点 v_i 在一棵树 T 中存活的轮数是节点的生命周期.存活指节点 v_i 的剩余能量 $E(v_i)>0$.节点的生命周期可定义为 $L_{node}(T,v_i)=\left\lfloor \frac{E(v_i)}{lS(T,v_i)(E_r+E_t)+lE_t} \right\rfloor$,$i=0,\ldots,n$.

定义 5(树的生命周期(tree lifetime)). 树 T 中第 1 个节点死亡时,该节点存活的轮数是树的生命周期.定义为

$$L_{tree}(T)=\min_{i=0,\ldots,n}\{L_{node}(T,v_i)\}.$$

定义 6(网络生命周期(network lifetime)). 网络中第 1 个节点死亡时,该节点存活的轮数[15]是网络生命周期.

定义 7(最优树). 最优树是根据当前网络中所有节点能量水平构造的一棵生成树,其生命周期比网络中其他生成树的生命周期都要大.定义为 $T_o=\{T \mid L_{tree}(T)=\max_{T^*\in TS(G)} L_{tree}(T^*)\}$,$T^*$ 是网络 G 中任意一棵生成树,$TS(G)$是图 G 中生成树的集合.

定义 8(瓶颈节点). 当前树中最早消耗完能量的节点,其生命周期与所在树的生命周期相当.

2.3 问题描述

传感器网络一般部署在比较危险或人类很难进入的区域,传感器节点往往是无法替换的.因此,我们希望网络能工作尽可能长的时间.对于网络中利用树进行数据收集的方式,根据目前已有文献的成果可分为以下 3 类:

(1) 网络只使用一棵树进行数据收集.此时,网络生命周期等于当前树的生命周期,IAA 属于这种方式.这种方式的优点是可以节省更换树结构所需的额外通信费用,缺点是树上的瓶颈节点容易死亡;

(2) 网络每轮重新构造一棵树进行数据收集.此时,网络生命周期等于在多棵树中存活的生命最短的那个节点的生命周期,MLDGA 和 MNL 属于这种方式.这种方式的优点是能够均衡网络中所有节点的能量消耗,缺点是频繁地更换树结构会增加大量额外的通信费用;

(3) 网络根据一定的频率重新更换树结构,PEDAP-AP 和 EESR 属于这种方式.此时,网络生命周期也是多棵树相互作用的结果.这种方式是方式(1)和方式(2)的一个平衡,优点是能在均衡所有节点能量消耗的同时,尽量减少更换树结构所需的额外通信费用;它的缺点是准确地更换树结构的频率很难获得.

对于方式(1),如果当前树的生命周期最大,网络生命周期就可以最大;对于方式(2),如果每轮更换的都是根据节点当前能量水平构造的生命周期最大的树,则节点的能量消耗较小且均衡.因此,即使经过在多棵树上进行工作,节点仍能最大限度地保存能量.这样,网络生命周期比网络使用一般的树时要长;对于方式(3),PEDAP-AP采用的是固定频率,EESR 采用的是动态频率.方式(3)实际上是方式(2)的特例,无论采用何种频率,一棵生命周期最大的树在其工作周期内肯定是最能保存节点能量的.每次更换的都是最大生命周期树的网络,生命周期也会比更换一般树的网络要长.

可以看到,无论网络采取哪种方式进行数据收集,要想延长网络生命周期都需要构造生命周期最大的树.由于网络中可能存在多个树结构,找到一棵生命周期最大的树的问题可以表示为

$$\max L_{tree}(T) \tag{1}$$

根据定义 4 和定义 5,公式(1)可以表达为

$$\max L_{tree}(T)=\max \min_{i=0,\ldots,n}\left\{\left\lfloor \frac{E(v_i)}{lS(T,v_i)(E_r+E_t)+lE_t}\right\rfloor\right\} \tag{2}$$

由于 E_r 和 E_t 是常量无法调节,只有 $S(T,v_i)$可以调节,是主要的优化目标.为了简化表达,将 $l(E_r+E_t)$从分母中提取出来,对公式(2)进行等价变换:

$$\max L_{tree}(T)\Leftrightarrow \max \min_{i=0,\ldots,n}\frac{E(v_i)}{S(T,v_i)+c} \tag{3}$$

其中,$c=E_t/(E_r+E_t)$.显然,这属于一种负载均衡问题,能量($E(v_i)$)越大的节点,应承担的子孙($S(T,v_i)$)就越多;反之,则越少.文献[15]通过对最小集合覆盖问题(the minimum set cover problem)进行归约,证明了这个问题是 NP 完全的.在通常情况下,构造一棵这样的最优树是很困难的.因此,我们提出一种新的算法 MAXLAT 来解决这个问题,以下是其详细的描述.

3 算法 MAXLAT 的设计

根据公式(3),节点应承担与其能量成比例的子孙,而瓶颈节点是承担的子孙数量超过了合理比例的节点.因此,算法 MAXLAT 以一棵 Sink 拥有最多孩子的生成树为基础,不断转移瓶颈节点的子孙到非瓶颈节点的子树上去,最后得到一棵瓶颈节点负载较轻的生成树.

3.1 算法MAXLAT描述

首先,在网络 G 中构造一棵 Sink 拥有最多孩子的生成树 T,构造步骤如下:

(1) 初始时,树上只有 Sink 节点,它是树的第 0 层节点;

(2) Sink 节点将通信半径内所有 1 跳邻居加入树中作为自己的孩子,这些节点也是树的第 1 层节点;

(3) 按照剩余能量从高到低的次序,树上第 h–1 层的每个节点每次任意挑选一个未加入树的邻居节点加入树中作为自己的孩子,这个新加入树的节点是树的第 h 层节点.循环执行,直至所有树的第 h–1 层节点的邻居都被加入树中.这里,$h\geqslant 2$ 为一个正整数.

步骤(3)迭代地执行,h=2,3,…,直至网络中所有节点加入树中,形成一棵 Sink 拥有最多孩子的生成树 T.直观地看,树 T 上拥有较多子树,这有利于子树间节点的转移.而且,如果节点能够均衡地分布到这些子树上,将有利于减轻瓶颈节点的负载.

在网络 G 中得到树 T 后,将以树 T 为基础进行优化.所谓优化,就是转移瓶颈节点的子孙到非瓶颈节点的子树上去,以减小树上瓶颈节点的子孙数量,即 $S(T,v_i)$的值.但是,$S(T,v_i)$在公式(3)中是式子的分母,并不容易进行调整.因此,将公式(3)转换为如下等价的形式:

$$\max L_{tree}(T)\Leftrightarrow \min \max_{i=0,\ldots,n}\frac{S(T,v_i)+c}{E(v_i)} \tag{4}$$

定义 $r(T)=\max_{i=0,\ldots,n}\frac{S(T,v_i)+c}{E(v_i)}=\max_{i=0,\ldots,n} r(T,v_i)$,称为树 T 的反生命周期,$r(T,v_i)$是树 T 上节点 v_i 的反生命周期.这

样,求解 max $L_{tree}(T)$的问题就转化为求解树 T 反生命周期最小化的问题,即

$$\max L_{tree}(T)\Leftrightarrow\min r(T) \tag{5}$$

此外,要减少树上瓶颈节点子孙的数量,首先要明确哪些节点属于瓶颈节点.为此,引入一个任意的常数$\varepsilon>0$代表瓶颈节点的反生命周期与树的反生命周期的最大差距.然后,根据树上节点 $v_i(i=0,1,\ldots,n)$的反生命周期 $r(T,v_i)$与 $r(T)$的差距,将所有节点划分到以下 3 个不相交的集合 V_1,V_2 和 V_3:

(1) $V_1=\{v_i:r(T)-\varepsilon<r(T,v_i)\leqslant r(T)\}$.在这个集合中,节点的反生命周期与 $r(T)$很接近,属于瓶颈节点;

(2) $V_2=\{v_i:r(T)-\varepsilon-1/E(v_i)<r(T,v_i)\leqslant r(T)-\varepsilon\}$.在这个集合中,节点的反生命周期很接近于瓶颈节点的反生命周期,如果它们的子孙增加一个(节点的反生命周期增加 $1/E(v_i)$),就会变为瓶颈节点.因此,称这个集合中的节点为次瓶颈节点;

(3) $V_3=V-V_1-V_2$.这个集合中的节点负载较轻,即使增加一个子孙节点也不会成为瓶颈节点,称为富裕节点.

为了在生成树 T 中转移瓶颈节点的子孙到非瓶颈节点的子树上,首先设定每个节点有一个颜色变量 *color*,以表示节点的状态.我们规定节点有 3 种颜色,分别是 Red,Yellow 和 Green.当一个节点 v_i 需要被着色时,首先计算该节点的反生命周期 $r(T,v_i)$,然后根据其属于哪个集合以及父节点的颜色来确定:

(1) 如果这个节点属于 V_1,则着色为 Red;

(2) 如果这个节点属于 V_2,则先查看父节点的颜色:如果父节点为 Red,则该节点着色为 Red;否则,该节点着色为 Yellow;

(3) 如果这个节点属于 V_3,则先查看父节点的颜色:如果父节点不是 Green,则该节点着色为父节点的颜色;否则,该节点着色为 Green.

对于节点的着色,是在对树的一次深度优先遍历中完成的.为了实现对树 T 中一棵以节点 x 为根的子树上所有节点进行着色,定义一个 *Color*(*T*,*x*)函数,具体描述如下.

```
Function Color(T,x)
1.  if (x 是 Sink 节点) x.color="Green";
2.  else {      根据 x 的反生命周期 r(T,x)与 r(T)的差距,判断 x 属于 V1,V2 或 V3 中的哪个集合并相应更新 V1,V2 或 V3 的内容;
3.              根据节点 x 的集合属性及父节点的颜色对 x 进行着色;
4.  }
5.  For (x 在树 T 中的每个孩子 y)
6.      Color(T,y);
```

注意,在对一棵树进行着色后,由于节点能量的异构,颜色为 Green 的子树中可能会包含颜色为 Red 或 Yellow 的子树;颜色为 Yellow 的子树中可能会包含颜色为 Red 的子树;颜色为 Red 的子树中不会包含其他颜色的子树.

此外,优化的主要操作是把节点的子孙进行转移到一棵 Green 子树上,有 2 种节点可能会需要转移自己的子孙:一种是 Red 的瓶颈节点,对其转移子孙是优化的主要目标;另一种是 Yellow 的次瓶颈节点.当瓶颈节点的 Red 子树旁出现一棵 Green 子树时,瓶颈节点可以直接转移自己的子孙到这棵 Green 子树上去.而当瓶颈节点的 Red 子树旁出现的是一棵 Yellow 子树时,希望该 Yellow 子树能首先转移一个节点到另一棵 Green 子树上,从而使自身变为一棵 Green 子树.这样,瓶颈节点才可能转移自己的子孙.为了实现以上操作,定义一个转移函数 *Transit*(*T*,*x*),用于在树 T 中对一个节点 x 进行子孙的转移,具体描述如下所示.

```
Function Transit(T,x)
1.  for (x 在树 T 中的每一个与 x 颜色相同而且 visited 属性为"false"的孩子节点 y)
2.  {   if (y 是一个叶子节点)
3.      {   for (y 在图 G 中的每一个邻居节点 z)
```

```
4.          {   if (z 不属于 x 在树 T 中的子孙)
5.              {   case z.color==“Green”
6.                      Exchange(y,z);
7.                      return true;
8.                  case z.color==“Yellow”
9.                      在树 T 中沿 z 的父节点向上找到 z 所属的“Yellow”子树的最高层根节点 z_root;
10.                     if (z_root.visited==“false”)
11.                     {   z_root.visited=“true”;
12.                         if (Transit(T,z_root) && z.color==“Green”)
13.                         {   Exchange(y,z);
14.                             return true;
15.                         }
16.                     }
17.             }
18.         }
19.     } else {if (Transit(T,y)) return true;}
20. }
21. return false;       //没有转移任何节点
```

Function *Exchange*(*y*,*z*)

1. *w*=*y*.*parent*;
2. *y*.*parent*=*z*;
3. 更新树 *T* 上 *w* 及 *w* 和 *y* 的所有祖先节点 *v* 的 *S*(*T*,*v*)值,并使 *v*.*visited*=“*false*”;
4. 沿 *y* 的父节点向上找到根节点 v_0 的某个孩子 v_{c1},执行 *Color*(*T*,v_{c1}).
5. 沿 *w* 的父节点向上找到根节点 v_0 的某个孩子 v_{c2},如果 $v_{c2}\neq v_{c1}$ 执行 *Color*(*T*,v_{c2}).

为了防止节点在被循环递归的执行转移操作从而造成死锁,给每个节点设置一个变量 *visited*,初始时每个节点的 *visited* 值均为 *false*.如果某个节点被执行转移操作,则其 *visited* 值设为 *true*.只有当转移操作成功,节点的子孙数量发生了改变,其 *visited* 值才恢复为 *false*.否则,该节点在迭代过程中不用再执行转移操作.

转移函数 *Transit*(*T*,*x*)对以 *x* 为根的子树进行遍历,对于遇到的任何一个与 *x* 颜色相同而且 *visited* 属性为 *false* 的叶子节点 *y*,首先在图 *G* 中寻找属于节点 *y* 的邻居但不属于 *x* 子孙的任意节点 *z*.限定 *y* 与 *x* 颜色相同是因为当进行 Yellow 节点的子孙转移时,其子树中可能包含有 Red 节点,而对 Red 节点的处理主要在算法主程序中完成.此外,要求 *y* 的 *visited* 属性为 *false* 是为了防止算法在转移节点过程中,由于递归操作而把其他已经处于递归状态的节点转移.

如果 *z* 是一个 Green 节点,则将 *z* 作为 *y* 新的父节点.假设 *w* 是 *y* 原来的父节点,则更新树 *T* 上 *w* 及 *w* 和 *y* 的所有祖先节点的 *S*(*T*,*v*)值.然后,对以 v_o 的某个孩子为根且包含 *y* 和 *w* 的所有子树重新进行着色.

如果 *z* 是一个 Yellow 节点,则从 *z* 出发沿其父节点向上查找 *z* 所属的 Yellow 子树的最高层根节点 z_{root}.如果 z_{root} 没有被执行过转移子孙的操作,即:z_{root}.*visited*=“*false*”,则递归地调用 *Transit*(*T*,z_{root}),尝试将 z_{root} 的一个子孙转移到另一棵 Green 子树上去.如果成功,将使 z_{root} 及其 Yellow 子树变为 Green,*z* 变为 Green.这样,就能将 *z* 作为 *y* 的新父节点.由于相关子树发生了改变,因此更新树 *T* 上 *w* 及 *w* 和 *y* 的所有祖先节点的 *S*(*T*,*v*)值并使这些节点的 *visited* 值设置为 *false*.因为 Yellow 子树只可能包含在 Green 子树当中,因此将这些祖先节点的 *visited* 值设置为 *false* 不会影响其他 Yellow 子树上节点 *visited* 值的设置.然后,对以 v_o 的某个孩子为根且包含 *y* 和 *w* 的所有子树重新进行着色.

MAXLAT 从一棵 Sink 拥有最多孩子的生成树 T 开始,首先计算树 T 的反生命周期并对树 T 进行着色.然后,将 V_1 中节点按反生命周期以降序排列,并从反生命周期最大节点 x 开始进行转移操作 $Transit(T,x)$.如果转移操作成功,即 $Transit(T,x)=true$ 且 V_1 不为空,则重新对 V_1 中节点进行排序和转移操作;如果转移操作成功,即 $Transit(T,x)=true$ 但 V_1 为空,则说明按当前的标准,已经没有瓶颈节点,则重新计算树 T 的反生命周期,定义新的瓶颈节点及执行优化操作.如果转移操作失败,即 $Transit(x)=false$,则取 V_1 中的下一个节点 y 开始进行转移操作 $Transit(T,y)$.若 y 不存在,则说明当前的瓶颈节点无法继续优化,算法结束.算法 MAXLAT 的具体描述见算法 1.

命题 1. 算法 MAXLAT 至多执行 $O(n^2(1+1/\varepsilon E_{\min}))$轮迭代,并且每轮迭代均能够在多项式时间内结束.

证明:算法 MAXLAT 第 1 步构造一棵 Sink 拥有最多孩子的生成树 T.每次挑选一个节点加入树 T 中,执行 n 次后,n 个节点全部被加入树 T 中完成操作.第 2 步~第 16 步是迭代地对树 T 进行优化,如果迭代无法对任何瓶颈节点进行优化,算法直接结束.如果迭代能成功执行,每轮迭代会使一个"瓶颈节点"的反生命周期减少 $1/E(v_i)$,则只需经过至多$\lceil \varepsilon E(v_i)\rceil$轮迭代其反生命周期可以小于或等于 $r(T)-\varepsilon$.因此,使当前所有瓶颈节点的反生命周期都小于或等于 $r(T)-\varepsilon$,最多需要迭代的轮数是

$$\sum_{v_i\in V_1}\lceil \varepsilon E(v_i)\rceil \leqslant \sum_{v_i\in V_1}(\varepsilon E(v_i)+1)=\varepsilon\sum_{v_i\in V_1}E(v_i)+|V_1| \tag{6}$$

算法 1. MAXLAT.

```
1. 构建一棵 Sink 拥有最多孩子的生成树 T;
2. Ischanged←true;
3. while (Ischanged)
4. {    对树 T 进行遍历,使每个节点 v.visited=false,并计算树的反生命周期 r(T);
5.      Color(T,v0);
6.      while ((V1<>NULL) && (Ischanged==true)
7.      {   Ischanged←"false";
8.          将 V1 中的节点按反生命周期以降序排列;
9.          for (V1 中每一个节点 x)
10.         {   if (Transit(T,x))
11.             {   Ischanged←true;
12.                 break;
13.             }
14.         }
15.     }
16.}
```

对于任意一个瓶颈节点 v_i,均有 $r(T)-\varepsilon<(S(T,v_i)+c)/E(v_i)\leqslant r(T)$,因此,$(r(T)-\varepsilon)E(v_i)<S(T,v_i)+c$.对于任何一个节点,其子孙数量 $S(T,v_i)<n$.因此,对于所有瓶颈节点有

$$(r(T)-\varepsilon)\sum_{v_i\in V_1}E(v_i)<\sum_{i\in V_1}(S(T,v_i)+c)<n^2+cn\Rightarrow\sum_{v_i\in V_1}E(v_i)<(n^2+cn)/(r(T)-\varepsilon) \tag{7}$$

又因为$(r(T)-\varepsilon)E_{\min}|V_1|\leqslant(r(T)-\varepsilon)\sum_{v_i\in V_1}E(v_i)<n^2+cn$,

$$\Rightarrow|V_1|<(n^2+cn)/(r(T)-\varepsilon)E_{\min} \tag{8}$$

将公式(7)和公式(8)代入公式(6),有

$$\Rightarrow\sum_{v_i\in V_1}\lceil \varepsilon E(v_i)\rceil<(n^2+cn)\varepsilon/(r(T)-\varepsilon)+(n^2+cn)/(r(T)-\varepsilon)E_{\min}=(n^2+cn)(1+\varepsilon E_{\min})/(r(T)-\varepsilon)E_{\min} \tag{9}$$

注意到 $r(T)\leqslant\lceil(n+c)/E_{\min}\rceil$,联合公式(9)可知,算法使当前所有瓶颈节点的反生命周期都小于或等于 $r(T)-\varepsilon$ 需要 $O(n(\varepsilon E_{\min}+1))$轮迭代.此时,树 T 的反生命周期 $r(T)$至少降低ε.根据新的 $r(T)$,算法又会定义新的瓶颈节点并继续迭代地对树进行优化.

假设最优树的反生命周期是 r^*,则算法 MAXLAT 结束时至多执行$\lceil (r(T)-r^*)/\varepsilon \rceil O(n(\varepsilon E_{\min}+1))$轮迭代.因为$\lceil (r(T)-r^*)/\varepsilon \rceil < \lceil ((n+c)/E_{\min}-0)/\varepsilon \rceil = \lceil (n+c)/\varepsilon E_{\min} \rceil$,其中,$E_{\min}=\min_{i=0,\ldots,n} E(v_i)$.因此,算法 MAXLAT 至多执行的迭代轮数$\lceil (r(T)-r^*)/\varepsilon \rceil O(n(\varepsilon E_{\min}+1) < \lceil (n+c)/\varepsilon E_{\min} \rceil O(n(\varepsilon E_{\min}+1)=O(n^2(1+1/\varepsilon E_{\min}))$.

在每轮迭代中,算法对所有瓶颈节点尝试执行转移操作.只要有一个瓶颈节点的子孙被成功转移,则算法进入下一轮迭代.转移操作有两种情况:(1) 当瓶颈节点的子孙能被直接转移时,迭代结束并进入下一轮;(2) 对某些瓶颈节点进行转移操作时,可能需要递归地对树 T 中一些 Yellow 节点进行转移操作,并对相关子树进行重新着色.着色只会改变子孙数量发生变化的节点的颜色,而其他节点不受影响.着色操作能够保证只将负载高的节点的子孙转移给负载低的节点,因此,转移操作不会产生新的瓶颈节点.

考虑任何一个递归的转移子孙操作,由于受 *visited* 属性的限制,对某个 Yellow 节点转移子孙的操作只会发生在不同子树的 Yellow 节点之间或 Yellow 节点与 Green 节点之间,算法不会发生循环递归.如果当前找到的一个叶子节点无法转移,则继续寻找下一个叶子节点;如果没能转移任何节点,递归结束并返回.如果成功转移一个子孙节点,则对相关子树进行深度遍历进行重新着色然后返回,这能在 $O(n)$时间内结束.转移成功后,只会将转移了子孙的 Yellow 节点的 *visited* 属性会变为 *false*,而其他没进行子孙转移的 Yellow 节点的 *visited* 属性不会发生改变.此外,节点 *visited* 属性值的变化与着色操作无关,因此着色后不会造成新的循环递归.又因为树 T 中的节点是有限的,因此递归最终一定会结束并返回.所以,对瓶颈节点的转移操作也一定能结束.每轮迭代只存在两个结果:(1) 成功转移一个瓶颈节点的子孙而进入下一轮迭代;(2) 无法优化任何瓶颈节点而中止算法.因此,每轮迭代均能够在多项式时间内结束. □

执行完算法 MAXLAT 后,得到的生成树就是一棵瓶颈节点负载较轻的树.以下我们将对算法的时间复杂度和优化程度进行分析.

3.2 算法MAXLAT的时间复杂度

对于算法 MAXLAT 第 1 步,算法结束时 n 个节点分批进行了排序操作,最多需要 $O(n\log n)$的时间.挑选 n 个节点加入树中,需要对图 G 中每个节点和每条边都查看一次,花费的时间为 $O(n+m)$.因此,算法第 1 步的时间复杂度为 $O(n\log n+m)$.

算法 MAXLAT 第 2 步~第 16 步是一个迭代的优化操作,如命题 1 所述,算法至多执行 $O(n^2(1+1/\varepsilon E_{\min}))$轮迭代.每轮迭代的主要操作是对以瓶颈节点为根的子树进行深度优先遍历(包括可能递归地对一些 Yellow 子树进行遍历),并查看子树上的叶子节点所依附的边以进行转移操作,时间复杂度为 $O(nm)$.每次转移后需要对相关子树进行重新着色,时间复杂度为 $O(n)<O(nm)$.因此,算法 MAXLAT 第 2 步~第 16 步的时间复杂度为

$$O(n^2(1+1/\varepsilon E_{\min}))O(nm)=O(n^3m(1+1/\varepsilon E_{\min})).$$

定理 1. 算法 MAXLAT 的时间复杂度为 $O(n^3m(1+1/\varepsilon E_{\min}))$.

证明:由以上分析,算法第 1 步的时间复杂度为 $O(n\log n+m)$,第 2 步~第 16 步的时间复杂度为 $O(n^3m(1+1/\varepsilon E_{\min}))>O(n\log n+m)$.因此,整个算法的时间复杂度为 $O(n^3m(1+1/\varepsilon E_{\min}))$. □

3.3 ε的取值与算法效果

从算法的时间复杂度看,ε越大,算法的执行时间越少,则ε取值是否可以无穷大呢?以下,我们就假设ε取无穷大值.由于网络中节点的数量和节点的能量都是有限的,因此当$\varepsilon \geqslant \lceil (n+c)/E_{\min} \rceil$时,树上所有节点的反生命周期都在 V_1 中.

算法的目标是在树 T 上找到属于 V_1 的瓶颈节点,并将这些瓶颈节点的子孙转移到以 V_3 节点为根的子树上去.由于ε取无穷大值后,树上所有节点均属于 V_1,没有属于 V_3 的节点.因此,树无法得到任何优化,算法退化到算法第 1 步时的结果.这样,算法效果比较差.

另一方面,如果ε取无穷小值,会使 V_1 的范围区间$(r(T)-\varepsilon, r(T)]$很小.此时,树上只有一个反生命周期最大的(或少数几个反生命周期相同且为最大的)节点属于 V_1,而树上其余的节点均属于 V_2 或 V_3.在 V_3 节点较多的情况下,生成树有更多的机会得到优化,算法效果会比较好.但是,此时算法的时间复杂度会很大.

综上所述,ε取值小,算法MAXLAT的效果会比较好,但是算法的时间复杂度大;ε取值大,算法MAXLAT的效果会比较差,但是算法的时间复杂度小.因此,ε的取值必须在算法的时间复杂度和效果之间平衡.其合理的取值,我们将通过实验进行分析.

3.4 算法MAXLAT的实现

由于在网络中构造最大化生命周期的精确数据收集树是 NP 完全的,计算量很大.而 Sink 节点拥有无限的能量和强大的计算能力,因此,算法 MAXLAT 在 Sink 节点上运行.MAXLAT 只需要知道网络中节点的邻居信息即可执行,无须知道节点的位置.网络部署完成后,Sink 节点广播一个消息,使所有节点分布地组成一棵初始的任意树[17],节点通过这棵树将自己的邻居信息和剩余能量报告给 Sink 节点.Sink 节点在获得所有节点的信息后,就可以利用 MAXLAT 计算生成树了.

MAXLAT 执行完后,Sink 节点向每个节点发送其必须的信息(如节点的父节点信息、节点在每一轮数据收集中发送和接收数据的时隙、需要接收几个孩子节点发送来的数据等).信息放在一个 k 字节的包中洪泛给每个节点.每个节点接收到这个包,立即将其广播出去,使其他节点也能收到.为了避免节点多次接收这个包,同时保证洪泛能够结束,广播之后节点停止接收该包一段时间.

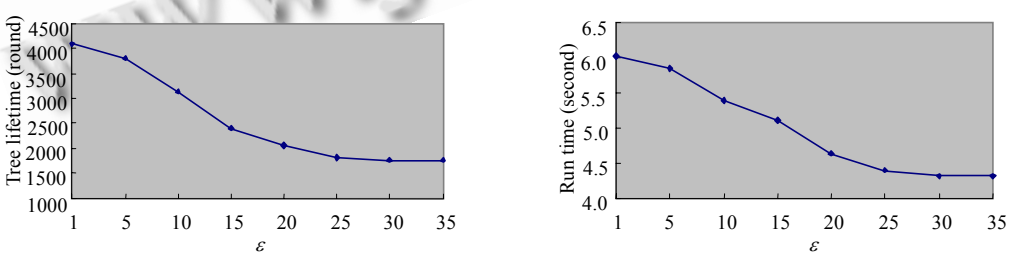
网络工作一段时间后,随着节点能量的耗尽或者有意外的损坏发生,节点会死亡;另一方面,可能会有新的节点被补充进来.因此,网络拓扑是会发生改变的,MAXLAT 会在网络拓扑改变时重新计算生成树.为了能够及时获知网络拓扑的改变,Sink 节点会在网络工作的过程中周期性地收集节点的邻居和剩余能量信息.

4 模拟实验

假设有 100 个节点随机分布在一个 $100\times100m^2$ 的正方形区域,Sink 节点位于这个区域的中心.每个节点的初始能量在[0.5J,1J]之间随机分布,节点的数据产生率为 128 bits/round.为了使所有算法能进行公平的对比,节点均采用固定发射功率,其中,节点的发送能耗大约是接收能耗的 2 倍,即 $E_t=2E_r$[21].E_r=50nJ/bit,所有节点的最大通信半径为 20m.由于典型的基于树的协议均在 Sink 上执行,他们在网络实现时所需的控制消息的能耗是几乎相等的.而且,当树结构更换不频繁时(如 IAA 只在网络部署和拓扑改变时更换树),相关的控制消息的能耗可以忽略不计.因此,我们主要关注树在数据收集时的通信能耗以及树的生命周期.实验的结果均是算法执行 20 次后的平均结果.

4.1 ε的合理取值

常量ε是用来界定瓶颈节点的关键常量,它的取值直接决定瓶颈节点的个数、树的优化程度和算法的执行时间.我们通过在同一棵任意树中对ε取不同的值进行优化,以考察ε的取值对 MAXLAT 性能的影响.图 1(a)是ε的不同取值对树的生命周期影响的结果,图 1(b)是算法在一台 Intel P4 2.8GHz CPU,1GB 内存的 PC 机上执行时所耗费的时间.



(a) Impact of ε's different values on tree lifetime
(a) ε的不同取值对树生命周期的影响

(b) Impact of ε's different values on run time
(b) ε的不同取值对执行时间的影响

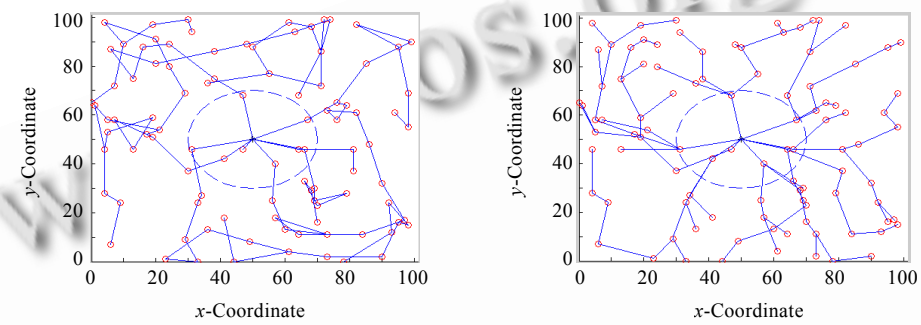
Fig.1 Impact of ε on performance of MAXLAT
图 1 ε对 MAXLAT 性能的影响

如图 1(a)所示,ε越小,树的生命周期越长;ε越大,树的生命周期则越短.这是因为在算法执行过程中,如果ε越小,V_1的范围越小,则属于 V_1的瓶颈节点的数量越少;与此对应,V_3的范围越大,属于 V_3的富裕节点越多.这样,在瓶颈节点及以它为根的子树周围,出现 Green 树的几率较大,容易实现瓶颈节点子孙的转移.瓶颈节点的负载得到减轻,树的生命周期就延长了.随着ε增大,树上有越来越多的节点被定义为瓶颈节点.而富裕节点数量减小,造成可优化几率减小.瓶颈节点的负载得不到有效减轻,树的生命周期就缩短了.

如图 1(b)所示,ε越小,算法的执行时间越长;ε越大,算法的执行时间越短.但是,由于算法在 Sink 节点上执行,而 Sink 节点的计算能力很强而且拥有无限的能量,因此算法的执行时间并不是我们关注的重点.我们主要关注算法如何获得生命周期更长的生成树,因此,取ε=1.下面的实验如不另加说明,则默认ε=1.

4.2 不同算法生成树对比

本次实验考察使用不同算法得到的生成树的情况.为了使结果清楚且容易比较,假定网络中所有节点的能量均为 1J.图 2(a)和图 2(b)分别是算法 MNL 和算法 MAXLAT 在同一网络拓扑得到的生成树的结果.



(a) Spanning tree of MNL
(a) 算法MNL的生成树

(b) Spanning tree of MAXLAT
(b) 算法MAXLAT的生成树

Fig.2 Comparison of different algorithms' trees

图 2 算法生成树对比

对比图 2(a)中算法 MNL 的生成树和图 2(b)中算法 MAXLAT 的生成树可以发现,其主要区别在于子树上的节点数量.图 2(a)中,算法 MNL 的生成树有 8 棵子树,树上节点数分别为 25,21,18,15,9,8,3,1;而图 2(b)中,算法 MAXLAT 的生成树有 8 棵子树,树上节点分别为 13,13,13,13,12,12,12,12.算法 MNL 得到的生成树中,各子树上节点数量差距较大,瓶颈节点有 24 个子孙,负载较重;而算法 MAXLAT 得到的生成树中,各子树上节点数量差距较小,瓶颈节点只有 12 个子孙,负载较轻.

造成这种区别的原因是,算法 MNL 在构造生成树时,决定树外节点加入树的权值主要考虑节点的剩余能量最大化.MNL 的贪婪性质使得树外节点倾向于选择当前负载最轻的树上节点作为父节点,而且节点加入树时均为叶子节点,并不能预测自身和当前父节点最后在树中的子孙数量.这导致最后构造出的生成树中,各子树上节点数量极不平衡,瓶颈节点的负载很重.为了克服算法 MNL 的缺点,算法 MAXLAT 首先构造一棵 Sink 拥有最多孩子的生成树,并以此作为基础.然后再根据节点的剩余能量和子孙数量进行综合考虑,不断转移负载重的节点的子孙到负载轻的节点上.最后得到的生成树,各子树上节点数量比较少而且较为均衡,有效地减轻了瓶颈节点的负载.

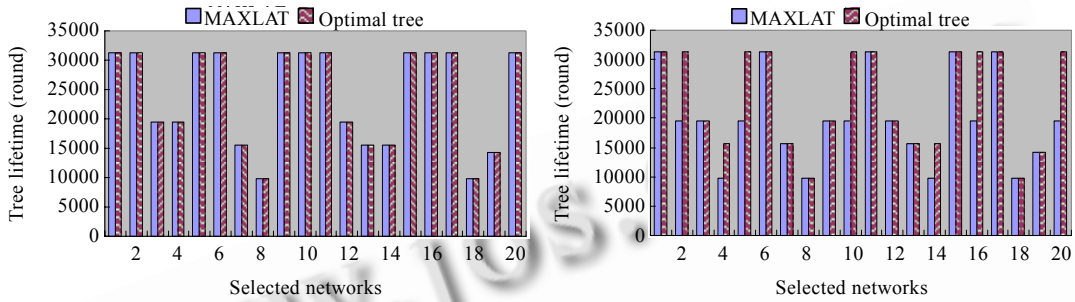
4.3 与最优树的生命周期对比

为了衡量算法的效果,将算法 MAXLAT 和最优树进行对比,观察各自树的生命周期.为了有统一的对比平台,随机产生 20 个网络拓扑,最优树通过枚举得到.由于在大规模网络中生成最优树是 NP 完全的,不太可能通过枚举得到,因此我们将网络中的节点数减少为 10 个,网络区域设置为 10×10 平方米,节点最大通信半径为 5m.分别设定ε=1 和ε=3,继续考察ε取值对算法结果的影响.这 2 种树的生命周期对比如图 3 所示.其中,图 3(a)是ε=1 时

的结果,图 3(b)是ε=3 时的结果.

如图 3(a)所示,在小规模的网络中,当ε=1 时,算法 MAXLAT 生成树的生命周期,达到了最优树的生命周期水平.如图 3(b)所示,当ε=3 时,在部分网络拓扑中,算法 MAXLAT 生成树的生命周期,要比最优树的生命周期短.这是因为随着ε的增大,使得优化程度降低了,算法 MAXLAT 在部分网络拓扑中无法进行有效的优化,得到的生成树就比最优树要差.

由于在大规模的网络拓扑中无法在有限时间中枚举出最优树,因此对于大规模网络,我们将与其他数据收集算法进行对比,结果见第 4.4 节.



(a) Comparison when ε=1
(a) ε=1时的比较

(b) Comparison when ε=3
(b) ε=3时的比较

Fig.3 Comparison with optimal trees

图 3 与最优树对比

4.4 与不同算法的树生命周期对比

为了考察算法的可扩展性,分别检验 Sink 节点的位置以及网络中的节点密度对算法的影响,选择了以下 4 种场景:

(1) Sink 节点位于区域中心,坐标为(50,50),区域内有 100 个节点,简称为 *C*100(Sink center, 100 nodes);

(2) Sink 节点位于区域外面,坐标为(105,50),区域内有 100 个节点,简称为 *O*100(Sink outside, 100 nodes);

(3) Sink 节点位于区域中心,坐标为(50,50),区域内有 200 个节点,简称为 *C*200(Sink center, 200 nodes);

(4) Sink 节点位于区域外面,坐标为(105,50),区域内有 200 个节点,简称为 *O*200(Sink outside, 200 nodes).

选择典型的基于树的数据收集算法 PEDAP,PEDAP-AP,MLDGA,MNL 作为比较.图 4 是 4 个场景中各个算法生成树的生命周期对比.

如图 4(a)所示,在场景 1(*C*100)中,MAXLAT 的树生命周期为 4112 轮,PEDAP 为 1141 轮,PEDAP-AP 为 1460 轮左右,MLDGA 为 1804,MNL 为 1816 轮.这是因为 PEDAP 在构造生成树时,节点加入树并不考虑父节点的能量水平,使得生成树上能量少的节点容易死亡,因此树生命周期最短;PEDAP-AP 考虑了节点的剩余能量,树生命周期比 PEDAP 提高了 28%.MLDGA 和 MNL 都是对 PEDAP-AP 进行改进,它们分别从树上节点最小生命周期最大化以及最小能量耗费等方面定义树外节点加入树的权值,它们的树生命周期分别比 PEDAP-AP 提高了 23%和 24%.MAXLAT 的生成树上瓶颈节点的负载较轻,因此树生命周期最长,比 MNL 提高了 126%.

如图 4(b)所示,在场景 2(*O*100)中,MAXLAT 的树生命周期为 1932 轮,PEDAP 为 707 轮,PEDAP-AP 为 947 轮,MLDGA 为 1208 轮,MNL 为 1220 轮.相比于场景 1(*C*100),各算法的树生命周期均有所下降,MAXLAT 下降了 53%,PEDAP 下降了 38%,PEDAP-AP 下降了 35%,MLDGA 下降了 33%,MNL 下降了 32%.这是由于在多跳网络中,各节点的通信范围有限,当 Sink 节点在区域外面时,能与其直接通信的节点减少了.这造成各算法生成树上子树数量的减少,子树上的节点数量有所增多.因此,各算法生成树上的瓶颈节点负载加重了,造成它们生命周期的下降.由于 MAXLAT 能够有效均衡瓶颈节点的负载,虽然 MAXLAT 的树生命周期下降比率较大,但其树生命周期仍是最高的.

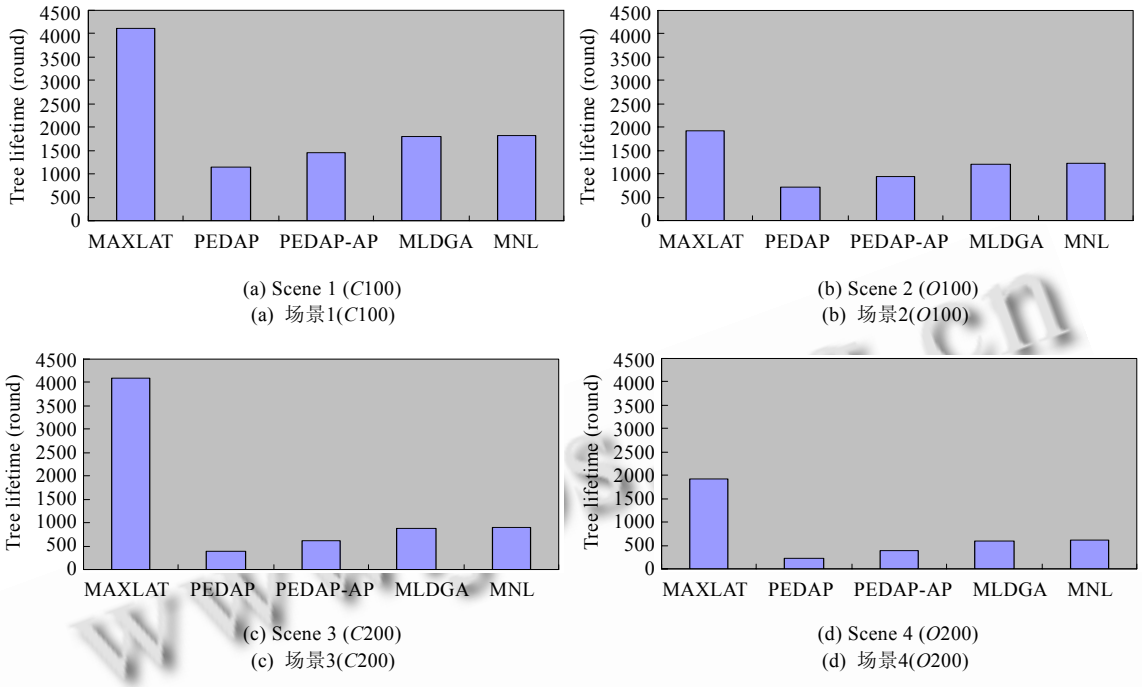


Fig.4 Comparison of tree lifetimes of different algorithms

图 4 不同算法的树生命周期比较

如图 4(c)所示,在场景 3(*C*200)中,采用 MAXLAT 的树生命周期达到 4098 轮,与场景 1(*C*100)时相差不大.这是因为,虽然网络中节点密度增加了,但有更多的节点能与 Sink 节点直接通信.这会生成较多的子树,经过算法优化后瓶颈节点的负载并没有明显增加.因此,MAXLAT 的树生命期变化不大.其他算法由于生成树上节点数量不均衡,随着网络中节点数量增多,瓶颈节点的负载有较大增加.它们的树生命周期比场景 1 时有大幅下降,其中:PEDAP 为 389 轮,下降了 66%;PEDAP-AP 为 612 轮,下降了 58%;MLDGA 为 882 轮,下降了 51%;MNL 为 903 轮,下降了 50%.

如图 4(d)所示,在场景 4(*O*200)中,采用 MAXLAT 的树生命周期为 1921 轮,PEDAP 为 228 轮,PEDAP-AP 为 393 轮,MLDGA 为 603 轮,MNL 为 618 轮.可以看出,所有算法的树生命周期比其他场景中都有所下降.但是,MAXLAT 的树生命周期与场景 2(*O*100)时相差不大,比场景 3(*C*200)时下降了 53%;而其他算法的树生命周期与场景 3(*C*200)时相比,PEDAP 下降了 41%,PEDAP-AP 下降了 36%,MLDGA 下降了 32%,MNL 下降了 31%,这与它们在场景 2 中较场景 1 时的生命周期下降比率非常接近.

从以上 4 个场景可以得到结论:在多跳的无线传感器网络中,Sink 节点放在区域的中心比放在区域外面能更有效地提高树的生命周期.Sink 节点的位置对 MAXLAT 的影响比对 PEDAP,PEDAP-AP,MLDGA 和 MNL 等算法的影响要大,但 MAXLAT 的树生命周期高于其他算法;另一方面,网络中节点密度的提高对 MAXLAT 的影响不大,只对 PEDAP,PEDAP-AP,MLDGA 和 MNL 有较大影响.因此,MAXLAT 特别适合于 Sink 节点放在区域的中心,并且密集地部署了大量传感器节点的应用.

5 总 结

在本文中,我们针对精确数据收集问题,研究如何构建能量均衡的生成树,使得网络生命周期最大化.这是一个 NP 完全问题,通过简单的启发式方法或近似算法都无法很好地解决.我们通过对生命周期模型进行分析,提出一种新的算法 MAXLAT 来构造瓶颈节点负载较轻的生成树.网络使用这种生成树进行数据收集,节点能保存能量长时间的工作.仿真实验表明,算法有效延长了树的生命周期,尤其适用于网络节点密度较高且 Sink 节点

位于网络区域中心的应用.

另一方面,本算法是将网络中全局的信息集中到 Sink 节点统一进行计算,计算完成后还需要将树结构发送给网络中的每个节点,这些操作需要耗费一定额外的能量用于节点间的通信和协调.因此,下一步我们将研究只需要节点周围的局部信息就能构造出合适的生成树的分布式算法.

致谢 在此,我们向给予本文宝贵意见和建议的评审专家表示衷心地感谢.

References:

[1] Estrin D, Govindan R, Heidemann J, Kumar S. Next century challenges: Scalable coordination in sensor networks. In: Proc. of the 5th Annual ACM/IEEE Int'l Conf. on Mobile Computing and Networking (MOBICOM 1999). New York: ACM Press, 1999. 263−270. http://portal.acm.org/citation.cfm?id=313556

[2] Jian Q, Gong ZH, Zhu PD, Gui CM. Overview of MAC protocols in wireless sensor networks. Journal of Software, 2008,19(2): 389−403 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/19/389.htm [doi: 10.3724/SP.J.1001.2008.0389]

[3] Cristescu R, Beferull-Lozano B, Vetterli M. On network correlated data gathering. In: Proc. of the 23rd IEEE Conf. on Computer Communications (INFOCOM 2004). Piscataway: IEEE Press, 2004. 2571−2582. http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1354677

[4] Liu JN, Adler M, Towsley D. On optimal communication cost for gathering correlated data through wireless sensor networks. In: Proc. of the 12th annual Int'l Conf. on Mobile Computing and Networking (MOBICOM 2006). New York: ACM Press, 2006. 310−321. http://portal.acm.org/citation.cfm?id=1161089.1161124

[5] Heinzelman WR, Chandrakasan A, Balakrishnan H. Energy-Efficient communication protocol for wireless micro-sensor networks. In: Proc. of the 33rd Hawaii Int'l Conf. on System Sciences. Washington: IEEE Computer Society, 2000. 3005−3014. http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=926982

[6] Younis O, Fahmy S. HEED: A hybrid energy efficient distributed clustering approach for ad hoc sensor networks. IEEE Trans. on Mobile Computing, 2004,3(4):366−379. [doi: 10.1109/TMC.2004.41]

[7] Li CF, Ye M, Chen GH, Wu J. An energy efficient unequal clustering mechanism for wireless sensor networks. In: Proc. of the 2nd IEEE Int'l Conf. on Mobile Ad hoc and Sensor Systems (MASS 2005). Washington: IEEE Computer Society, 2005. 598−604. http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1542849

[8] Taherkordi A, Mohammadi R, Eliassen F. A communication-efficient distributed clustering algorithm for sensor networks. In: Proc. of the 22nd IEEE Int'l Conf. on Advanced Information Networking and Applications (AINA 2008). Washington: IEEE Computer Society, 2008. 634−638. http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4482986

[9] Lindsey S, Raghavendra CS. PEGASIS: Power efficient gathering in sensor information systems. In: Proc. of the IEEE Aerospace Conf. San Francisco: IEEE Computer Society, 2002. 1−6. http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1035242.

[10] Jung SM, Han YJ, Chung TM. The concentric clustering scheme for efficient energy consumption in the PEGASIS. In: Proc. of the 9th IEEE Int'l Conf. on Advanced Communication Technology (ICACT 2007). Phoenix Park: IEEE Computer Society, 2007. 260−265. http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4195130

[11] Lindsey S, Raghavendra C, Sivalingam KM. Data gathering algorithms in sensor networks using energy metrics. IEEE Trans. on Parallel and Distributed Systems, 2002,13(9):924−935. [doi: 10.1109/TPDS.2002.1036066]

[12] Khan M, Pandurangan G, Vullikanti A. Distributed algorithms for constructing approximate minimum spanning trees in wireless sensor networks. IEEE Trans. on Parallel and Distributed Systems, 2009,20(1):124−139. [doi: 10.1109/TPDS.2008.57]

[13] Tan HO, Korpeoglu I. Power efficient data gathering and aggregation in wireless sensor networks. SIGMOD Record, 2003,32(4): 66−71. [doi: 10.1145/959060.959072]

[14] Zhang Q, Xie ZP, Ling B, Sun WW, Shi BL. A maximum lifetime data gathering algorithm for wireless sensor networks. Journal of Software, 2005,16(11):1946−1957 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/16/1946.htm [doi: 10.1360/jos161946]

[15] Liang WF, Liu YZ, Online data gathering for maximizing network lifetime in sensor networks. IEEE Trans. on Mobile Computing, 2007,6(1):2−11. [doi: 10.1109/TMC.2007.250667]

[16] Hussain S, Islam O. An energy spanning tree based multi-hop routing in wireless sensor networks. In: Proc. of the IEEE Wireless Communications and Networking Conf. (WCNC 2007). Washington: IEEE Computer Society, 2007. 4383−4388. http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4225044

[17] Wu Y, Fahmy S, Shroff NB. On the construction of a maximum-lifetime data gathering tree in sensor networks: NP-Completeness and approximation algorithm. In: Proc. of the 27th IEEE Conf. on Computer Communications (INFOCOM 2008). Washington: IEEE Computer Society, 2008. 356−360. http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4509675

[18] Bhattacharjee S, Das N. Distributed data gathering scheduling in multi-hop wireless sensor networks for improved lifetime. In: Proc. of the 17th Int'l Conf. on Computer Theory and Applications (ICCTA 2007). Washington: IEEE Computer Society, 2007. 46−50. http://portal.acm.org/citation.cfm?id=1260320

[19] Kumar S, Lai TH, Balogh J. On *k*-coverage in a mostly sleeping sensor network. In: Proc. of the 10th Annual Int'l Conf. on Mobile Computing and Networking (MOBICOM 2004). New York: ACM Press, 2004. 144−158. http://portal.acm.org/citation.cfm?id=1023735

[20] Buragohain C, Agrawal D, Suri S. Power aware routing for sensor databases. In: Proc. of the 24th IEEE Conf. on Computer Communications (INFOCOM 2005). Piscataway: IEEE Press, 2005. 1747−1757. http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1498455

[21] Intanagonwiwat C, Govindan R, Estrin D. Directed diffusion: A scalable and robust communication paradigm for sensor networks. In: Proc. of the 6th Annual Int'l Conf. on Mobile Computing and Networking (MOBICOM 2000). New York: ACM Press, 2000. 56−67. http://portal.acm.org/citation.cfm?id=345920

附中文参考文献:

[2] 蹇强,龚正虎,朱培栋,桂春梅,无线传感器网络 MAC 协议研究进展.软件学报,2008,19(2):389−403. http://www.jos.org.cn/1000-9825/19/389.htm [doi: 10.3724/SP.J.1001.2008.0389]

[14] 张卿,谢志鹏,凌波,孙未未,施伯乐.一种传感器网络最大化生命周期数据收集算法.软件学报,2005,16(11):1946−1957. http://www.jos.org.cn/1000-9825/16/1946.htm [doi: 10.1360/ jos161946]

梁俊斌(1979−),男,广西南宁人,博士生,讲师,主要研究领域为无线传感器网络,计算机优化算法.

李陶深(1957−),男,博士,教授,主要研究领域为网络优化算法,分布式数据库,遗传优化设计,CAD 理论与应用.

王建新(1969−),男,博士,教授,博士生导师,主要研究领域为计算机网络,网络优化理论,计算机优化算法.

陈建二(1954−),男,博士,教授,博士生导师,主要研究领域为计算机网络优化理论,计算复杂性及优化,生物信息学.