(19)国家知识产权局

(12)发明专利

(10)授权公告号 CN 116156593 B
(45)授权公告日 2024.03.26

(21)申请号 202310145593.X

(22)申请日 2023.02.01

(65)同一申请的已公布的文献号
申请公布号 CN 116156593 A

(43)申请公布日 2023.05.23

(73)专利权人 深圳市华普微电子股份有限公司
地址 518055 广东省深圳市南山区西丽街道西丽社区留新四街万科云城三期C区八栋A座3001房3002房、3003房、3004房、3005房
专利权人 无锡泽太微电子有限公司

(72)发明人 范观平 阮庆瑜

(74)专利代理机构 深圳市深联知识产权代理事务所(普通合伙) 44357
专利代理师 黄立强

(51)Int.Cl.
H04W 40/02(2009.01)
H04W 40/00(2009.01)
H04W 40/32(2009.01)
H04W 84/18(2009.01)
H04L 45/48(2022.01)

(56)对比文件
CN 105050149 A,2015.11.11
CN 106255134 A,2016.12.21
CN 108632940 A,2018.10.09
KR 20090097608 A,2009.09.16
US 2009059816 A1,2009.03.05
CN 103428899 A,2013.12.04
WO 2015108521 A1,2015.07.23

审查员 吕源

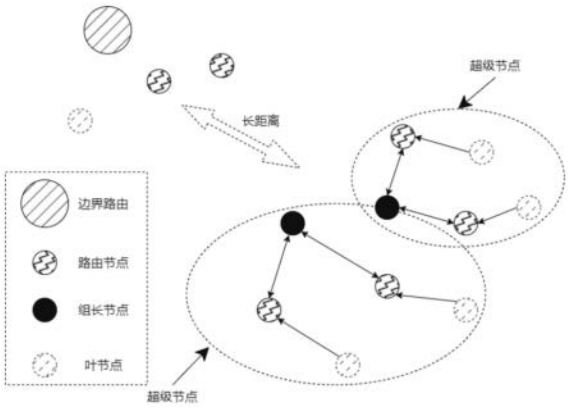
权利要求书3页 说明书6页 附图4页

(54)发明名称

一种无线自组网的方法

(57)摘要

本发明公开了一种无线自组网方法,包括如下步骤:步骤1、所有节点上电完成初始配置后,通过邻居发现机制,获取邻居的基本信息,所述基本信息包括获取节点在网络中的角色;步骤2、根据不同的角色寻求附近的邻居节点,并与该节点的父节点建立连接;步骤3、与邻居中相互关联的节点组成一个超级节点,同时确定组长节点,所述组长节点提前验证组内各节点的合法性,并检测路由好坏;步骤4、当所有节点都完成入网并验证合法之后,组长节点将丢弃其身份,作为网络中普通的路由节点,完成后续网络的各项功能,完成自组网。本发明解决大规模部署无线节点时,入网时间较长的问题,尤其是加速一些边缘节点的入网速度,从而减少网络的部署时间。



CN 116156593 B

1.一种无线自组网的方法,其特征在于,包括如下步骤:

步骤1、所有节点上电完成初始配置后,通过邻居发现机制,获取邻居的基本信息,所述基本信息包括获取节点在网络中的角色;

步骤2、根据不同的角色寻求附近的邻居节点,并与该节点的父节点建立连接;

步骤3、与邻居中相互关联的节点组成一个超级节点,同时确定组长节点,所述组长节点提前验证组内各节点的合法性,并检测路由好坏;

步骤4、当所有节点都完成入网并验证合法之后,组长节点将丢弃其身份,作为网络中普通的路由节点,完成后续网络的各项功能,完成自组网。

2.根据权利要求1所述的一种无线自组网的方法,其特征在于:所述节点在网络中的角色包括:

叶节点,所述叶节点寻求附近的路由节点;

若发现邻居存在一个路由节点,则直接与其建立路由关系;

若发现邻居有多个路由节点,则利用常规算法找到最佳的路由节点作为父节点;

若发现邻居中不存在路由节点,则节点进入睡眠,等待下一次唤醒后再重复此过程。

3.根据权利要求2所述的一种无线自组网的方法,其特征在于:所述节点在网络中的角色还包括:

路由节点,寻找附近的叶节点或路由节点;

若附近只有叶节点,则与其建立路由关系,并保存路由表;

若附近既有叶节点又有路由节点,则尝试与没有父节点的孤儿节点建立路由关系,并且选择最佳的路由节点建立联系,两个路由节点之间进行比较,子节点较多的则为组长节点;

若子节点一样多,则先上电的为组长节点,或者利用其他的一些方法选取组长节点,只需保证组长节点尽可能多的保存和管理组内节点的信息,若路由节点本身未与任何一个组长节点建立联系,则尝试与邻居中的某个较优的组长节点建立联系;

若未发现有组长节点,则让自己成为组长节点;

若与某个组长节点关联以后,则放弃与其他路由节点的关联。

4.根据权利要求2所述的一种无线自组网的方法,其特征在于:所述节点在网络中的角色还包括:

组长节点,则无需再次判断是否存在邻居,根据组长节点的定义,其邻居必定存在;

一个路由节点在成为组长节点之后,会尽最大可能尝试与邻居中的孤儿节点进行关联,包括孤儿叶节点和孤儿路由节点;

其中,组长节点不与另一个组长节点相关联。

5.根据权利要求1所述的一种无线自组网的方法,其特征在于:所述组成超级节点过程中,当边界路由或中心节点传递过来的入网信息到达超级节点之后,所述超级节点内的所有节点迅速回应中心节点的请求,加速节点的入网。

6.根据权利要求2所述的一种无线自组网的方法,其特征在于:所述的常规算法包括递归算法和非递归算法;

所述递归算法采用如下计算策略:

1)定义节点在网络中的角色,明确这个角色的功能,由于递归的特点是问题和子问题

都会调用函数自身,所以这个角色的功能一旦确定还需要找寻问题与子问题的递归关系;

2)寻找问题与子问题间的递推公式,由于问题与子问题具有相同解决思路,只要子问题调用上述定义好的节点在网络中的角色问题即解决;

3)将寻找问题与子问题间的递推公式用代码表示出来补充到步骤1)定义的节点在网络中的角色中;

4)根据问题与子问题的关系,推导出时间复杂度,如果发现递归时间复杂度不可接受,则需转换思路对其进行改造。

7.根据权利要求6所述的一种无线自组网的方法,其特征在于:所述非递归算法的计算策略为:

借助栈来实现,用栈保留当前结点的祖先信息;

由后序遍历的特点可知,当第三次搜索到某结点时才访问该结点,而第一次和第二次搜索到某结点时,压入栈而不访问,所以需要一个标志位来判断当前结点是否已经遍历完了其左子树和右子树,规定tag=0时不访问,tag=1时访问;

当搜索到某结点不为空时,要先遍历其左子树,因而将tag赋值为0;

若当前指针结点为空,则弹出栈顶元素,根据tag判断遍历完左子树还是右子树;若遍历完左子树,则将该元素二次压入栈,tag赋为1,遍历其右子树;若遍历完右子树,即第三次搜索到,则此时栈中元素正好是从根节点到该结点的路径经过的元素,所以应该在第三次搜索到该结点时判断该结点数据域值是否与要查找的元素值相同;若相同,则栈中元素依次出栈;若不相同,则继续查找;

而不管第三次搜索到的元素是否为所查找的元素,其所在的双亲的左子树已经遍历完,所以要将其指向设为空。

8.根据权利要求7所述的一种无线自组网的方法,其特征在于:所述的非递归算法具体步骤如下:

初始化空栈,指针currentNodePtr指向根结点;

当当前结点currentNodePtr不为空或栈不为空时,循环执行以下操作:

1) 若当前指针不为空,执行以下操作:

定义SelemType型变量e,其数据域nodePtr赋值为当前指针currentNodePtr,标志位tag赋值为0;

将e压入栈;

currentNodePtr指向其左子树;

2)否则,即当前树结点为空时,执行下列操作:

定义SelemType型变量e,弹出栈顶元素并赋给e;

currentNodePtr指向$e_{nodePtr}$,t赋值为e_{tag};

若t=0,即遍历完左子树,则使e_{tag}为1,将e压入栈;

否则,即遍历完右子树,完成以下操作:

若当前指针的整型数值等于x,则输出该元素,当栈不为空时,循环执行弹出栈顶元素并输出;

当前指针置空,即currentNodePtr。

9.根据权利要求1所述的一种无线自组网的方法,其特征在于:步骤4中,当一个超级节

点形成以后,其中的所有路由节点都传递网络配置信息给下一个超级节点,以此往后不断地拓展网络,直至所有部署的节点都已入网完成;

当所有节点都完成入网并验证合法之后,组长节点将丢弃其身份,作为网络中普通的路由节点,完成后续网络的各项功能。

10.根据权利要求1所述的一种无线自组网的方法,其特征在于:在所述组网的过程中,组长节点具有提前验证组内各节点的合法性的效果,并检测路由好坏,即对其组内的节点进行特定的测试,各个组长检查其成员,达到批量测试和认证节点的目的,从而为快速组网成功提供一定的基础。

一种无线自组网的方法

技术领域

[0001] 本发明属于无线自组网术领域,更具体地说,尤其涉及一种无线自组网的方法。

背景技术

[0002] 目前,广域无线自组网在需要大规模部署无线网络节点的时候,往往需要从中心节点出发,一级级的往外连接和扩展可配置的节点,受限于节点的入网速度,当所有节点都入网成功时,所需要的时间是非常可观的。假设每个节点的入网时间为秒级,当需要在户外部署具有千个以上节点的大规模网络时,则需要近几十分钟甚至更长的时间,且随着节点的增多,需要部署的节点距离中心节点或边界路由越来越远,或其他可能出现的一些错误等,都会延长节点入网的总时间,尤其是一些边缘节点,因此,我们提出了一种无线自组网的方法。

发明内容

[0003] 本发明的目的是为了解决现有技术中存在的缺点,而提出的一种无线自组网的方法,解决大规模部署无线节点时,入网时间较长的问题,尤其是加速一些边缘节点的入网速度,从而减少网络的部署时间。

[0004] 为实现上述目的,本发明提供如下技术方案:

[0005] 一种无线自组网的方法,包括如下步骤:

[0006] 步骤1、所有节点上电完成初始配置后,通过邻居发现机制,获取邻居的基本信息,所述基本信息包括获取节点在网络中的角色;

[0007] 步骤2、根据不同的角色寻求附近的邻居节点,并与该节点的父节点建立连接;

[0008] 步骤3、与邻居中相互关联的节点组成一个超级节点,同时确定组长节点,所述组长节点提前验证组内各节点的合法性,并检测路由好坏;

[0009] 步骤4、当所有节点都完成入网并验证合法之后,组长节点将丢弃其身份,作为网络中普通的路由节点,完成后续网络的各项功能,完成自组网。

[0010] 优选的,所述节点在网络中的角色包括:

[0011] 叶节点,所述叶节点寻求附近的路由节点;

[0012] 若发现邻居存在一个路由节点,则直接与其建立路由关系;

[0013] 若发现邻居有多个路由节点,则利用常规算法找到最佳的路由节点作为父节点;

[0014] 若发现邻居中不存在路由节点,则节点进入睡眠,等待下一次唤醒后再重复此过程。

[0015] 优选的,所述节点在网络中的角色还包括:

[0016] 路由节点,寻找附近的叶节点或路由节点;

[0017] 若附近只有叶节点,则与其建立路由关系,并保存路由表;

[0018] 若附近既有叶节点又有路由节点,则尝试与没有父节点的孤儿节点建立路由关系,并且选择最佳的路由节点建立联系,两个路由节点之间进行比较,子节点较多的则为组

长节点；

[0019] 若子节点一样多，则先上电的为组长节点，或者利用其他的一些方法选取组长节点，只需保证组长节点尽可能多的保存和管理组内节点的信息。若路由节点本身未与任何一个组长节点建立联系，则尝试与邻居中的某个较优的组长节点建立联系；

[0020] 若未发现有组长节点，则可以让自己成为组长节点；

[0021] 若与某个组长节点关联以后，则放弃与其他路由节点的关联。

[0022] 优选的，所述节点在网络中的角色还包括：

[0023] 组长节点，则无需再次判断是否存在邻居，根据组长节点的定义，其邻居必定存在；

[0024] 一个路由节点在成为组长节点之后，会尽最大可能尝试与邻居中的孤儿节点进行关联，包括孤儿叶节点和孤儿路由节点；

[0025] 其中，组长节点不与另一个组长节点相关联。

[0026] 优选的，所述组成超级节点过程中，当边界路由或中心节点传递过来的入网信息到达超级节点之后，所述超级节点内的所有节点迅速回应中心节点的请求，加速节点的入网。

[0027] 优选的，所述的常规算法包括递归算法和非递归算法；

[0028] 所述递归算法采用如下计算策略：

[0029] 1)定义节点在网络中的角色，明确这个角色的功能，由于递归的特点是问题和子问题都会调用函数自身，所以这个角色的功能一旦确定还需要找寻问题与子问题的递归关系；

[0030] 2)寻找问题与子问题间的递推公式，由于问题与子问题具有相同解决思路，只要子问题调用上述定义好的节点在网络中的角色问题即解决；

[0031] 3)将寻找问题与子问题间的递推公式用代码表示出来补充到步骤1)定义的节点在网络中的角色中；

[0032] 4)根据问题与子问题的关系，推导出时间复杂度，如果发现递归时间复杂度不可接受，则需转换思路对其进行改造。

[0033] 优选的，所述非递归算法的计算策略为：

[0034] 借助栈来实现，用栈保留当前结点的祖先信息；

[0035] 由后序遍历的特点可知，当第三次搜索到某结点时才访问该结点，而第一次和第二次搜索到某结点时，压入栈而不访问，所以需要一个标志位来判断当前结点是否已经遍历完了其左子树和右子树，规定tag＝0时不访问，tag＝1时访问；

[0036] 当搜索到某结点不为空时，要先遍历其左子树，因而将tag赋值为0；

[0037] 若当前指针结点为空，则弹出栈顶元素，根据tag判断遍历完左子树还是右子树；若遍历完左子树，则将该元素二次压入栈，tag赋为1，遍历其右子树；若遍历完右子树(即第三次搜索到)，则此时栈中元素正好是从根节点到该结点的路径经过的元素，所以应该在第三次搜索到该结点时判断该结点数据域值是否与要查找的的元素值相同；若相同，则栈中元素依次出栈；若不相同，则继续查找；

[0038] 而不管第三次搜索到的元素是否为所查找的元素，其所在的双亲的左子树已经遍历完，所以要将其指向设为空。

[0039] 优选的,所述的非递归算法具体步骤如下:

[0040] 初始化空栈,指针currentNodePtr指向根结点。

[0041] 当当前结点currentNodePtr不为空或栈不为空时,循环执行以下操作:

[0042] 1)若当前指针不为空,执行以下操作:

[0043] 定义SelemType型变量e,其数据域nodePtr赋值为当前指针currentNodePtr,标志位tag赋值为0;

[0044] 将e压入栈;

[0045] currentNodePtr指向其左子树。

[0046] 2)否则,即当前树结点为空时,执行下列操作:

[0047] 定义SelemType型变量e,弹出栈顶元素并赋给e;

[0048] currentNodePtr指向enodePtr,t赋值为etag;

[0049] 若t=0,即遍历完左子树,则使e.tag为1,将e压入栈;

[0050] 否则,即遍历完右子树,完成以下操作:

[0051] 若当前指针的整型数值等于x,则输出该元素,当栈不为空时,循环执行弹出栈顶元素并输出;

[0052] 当前指针置空,即currentNodePtr。

[0053] 优选的,步骤4中,当一个超级节点形成以后,其中的所有路由节点都可以传递网络配置信息给下一个超级节点,以此往后不断地拓展网络,直至所有部署的节点都已入网完成;

[0054] 当所有节点都完成入网并验证合法之后,组长节点将丢弃其身份,作为网络中普通的路由节点,完成后续网络的各项功能。

[0055] 优选的,在所述组网的过程中,组长节点具有提前验证组内各节点的合法性的效果,并可以检测路由好坏,即对其组内的节点进行特定的测试,各个组长检查其成员,达到批量测试和认证节点的目的,从而为快速组网成功提供一定的基础。

[0056] 本发明的技术效果和优点:本发明提供的一种无线自组网的方法,与现有技术相比,本发明旨在解决大规模部署无线节点时,入网时间较长的问题,尤其是加速一些边缘节点的入网速度,从而减少网络的部署时间;

[0057] 其次,本该方法利用某些无线自组网(如Wi-Sun FAN)邻居发现的特性,实现相邻节点的预先入网连接;一些待入网的节点,尤其是离中心节点较远的,在等待入网邀请的过程中,先与同在等待入网的邻居节点建立本地连接关系,组成一个待入网的超级节点,超级节点可视为一个整体,当超级节点入网时则表示组成超级节点的所有节点都入网成功;在超级节点中,各个节点的网络配置参数一致,且入网成功后,其中的所有节点都能传递网络的基本信息,从而邀请下一个节点或超级节点入网,以此加速网络的组建;

[0058] 基于上述,本发明具有如下效果:

[0059] 1.彼此相距较远的节点可以实现类似并行入网的效果,加速偏远节点或边缘节点的入网速度;

[0060] 2.减少入网时与中心节点的交互流程,降低路径损耗带来的部分错误;

[0061] 3.实现多节点同时部署,并可进行批量测试。

附图说明

[0062] 图1为本发明叶节点启动流程图；

[0063] 图2为本发明路由节点启动流程图；

[0064] 图3为本发明组长节点启动流程图；

[0065] 图4为本发明利用超级节点进行大规模组网过程图。

具体实施方式

[0066] 为了使本发明的目的、技术方案及优点更加清楚明白，以下结合具体实施例，对本发明进行进一步详细说明。应当理解，此处所描述的具体实施例仅仅用以解释本发明，并不用于限定本发明。基于本发明中的实施例，本领域普通技术人员在没有做出创造性劳动前提下所获得的所有其他实施例，都属于本发明保护的范围。

[0067] 本发明提供了的一种无线自组网的方法，旨在解决大规模部署无线节点时，入网时间较长的问题，尤其是加速一些边缘节点的入网速度，从而减少网络的部署时间，具体如下：

[0068] 1.所有节点上电完成初始配置后，通过邻居发现机制，获取邻居的基本信息，如节点在网络中的角色(路由节点或叶节点)，链路质量等；

[0069] 2.如果节点本身为叶节点，则寻求附近的路由节点；如图1所示，若发现邻居存在一个路由节点，则直接与其建立路由关系；若发现邻居有多个路由节点，则利用一定的算法(包括但不限于链路质量)，找到最佳的路由节点作为父节点；若发现邻居中不存在路由节点，则节点进入睡眠，等待下一次唤醒后再重复此过程；

[0070] 3.如果节点本身为路由节点，寻找附近的叶节点或路由节点；如图2所示，若附近只有叶节点，则与其建立路由关系，并保存路由表；若附近既有叶节点又有路由节点，则尝试与没有父节点的孤儿节点建立路由关系，并且选择最佳的路由节点建立联系，两个路由节点之间进行比较，子节点较多的则为“组长节点”，若子节点一样多，则先上电的为组长节点，或者利用其他的一些方法选取组长节点，只需保证组长节点尽可能多的保存和管理组内节点的信息；若路由节点本身未与任何一个组长节点建立联系，则尝试与邻居中的某个较优的组长节点建立联系，若未发现有组长节点，则可以让自己成为组长节点；若与某个组长节点关联以后，则放弃与其他路由节点的关联；

[0071] 4.如果节点为组长节点，则无需再次判断是否存在邻居，根据组长节点的定义，其邻居必定存在；如图3所示，一个路由节点在成为组长节点之后，会尽最大可能尝试与邻居中的孤儿节点进行关联，包括孤儿叶节点和孤儿路由节点；组长节点不与另一个组长节点相关联；

[0072] 5.组长节点与邻居中相互关联的节点组成一个超级节点；当边界路由或中心节点传递过来的入网信息到达超级节点之后，超级节点内的所有节点迅速回应中心节点的请求，加速这些节点的入网，相当于这些节点被打包一起入网了，如图4所示，此操作减少了一下特定数据包的传输，可降低路径损耗引入的错误概率；当一个超级节点形成以后，其中的所有路由节点都可以传递网络配置信息给下一个超级节点，以此往后不断地拓展网络，直至所有部署的节点都已入网完成；当所有节点都完成入网并验证合法之后，组长节点将丢弃其身份，作为网络中普通的路由节点，完成后续网络的各项功能；

[0073] 6.在组网的过程中,组长节点具有一定的功能,可以提前验证组内各节点的合法性,并可以检测路由好坏,即对其组内的节点进行特定的测试,各个组长检查其成员,达到批量测试和认证节点的目的,从而为快速组网成功提供一定的基础。

[0074] 作为本实施例需要说明的是,步骤2中,一定的算法具体包括递归算法和非递归算法;

[0075] 所述递归算法采用如下计算策略:

[0076] 1)定义节点在网络中的角色,明确这个角色的功能,由于递归的特点是问题和子问题都会调用函数自身,所以这个角色的功能一旦确定还需要找寻问题与子问题的递归关系;

[0077] 2)寻找问题与子问题间的递推公式,由于问题与子问题具有相同解决思路,只要子问题调用上述定义好的节点在网络中的角色问题即解决;

[0078] 3)将寻找问题与子问题间的递推公式用代码表示出来补充到步骤1)定义的节点在网络中的角色中;

[0079] 4)根据问题与子问题的关系,推导出时间复杂度,如果发现递归时间复杂度不可接受,则需转换思路对其进行改造。

[0080] 进一步的,所述非递归算法的计算策略为:

[0081] 借助栈来实现,用栈保留当前结点的祖先信息;

[0082] 由后序遍历的特点可知,当第三次搜索到某结点时才访问该结点,而第一次和第二次搜索到某结点时,压入栈而不访问,所以需要一个标志位来判断当前结点是否已经遍历完了其左子树和右子树,规定tag=0时不访问,tag=1时访问;

[0083] 当搜索到某结点不为空时,要先遍历其左子树,因而将tag赋值为0;

[0084] 若当前指针结点为空,则弹出栈顶元素,根据tag判断遍历完左子树还是右子树;若遍历完左子树,则将该元素二次压入栈,tag赋为1,遍历其右子树;若遍历完右子树(即第三次搜索到),则此时栈中元素正好是从根节点到该结点的路径经过的元素,所以应该在第三次搜索到该结点时判断该结点数据域值是否与要查找的的元素值相同;若相同,则栈中元素依次出栈;若不相同,则继续查找;

[0085] 而不管第三次搜索到的元素是否为所查找的元素,其所在的双亲的左子树已经遍历完,所以要将其指向设为空。

[0086] 另外需要说民的是,所述的非递归算法具体步骤如下:

[0087] 初始化空栈,指针currentNodePtr指向根结点。

[0088] 当当前结点currentNodePtr不为空或栈不为空时,循环执行以下操作:

[0089] 1)若当前指针不为空,执行以下操作:

[0090] 定义SelemType型变量e,其数据域nodePtr赋值为当前指针currentNodePtr,标志位tag赋值为0;

[0091] 将e压入栈;

[0092] currentNodePtr指向其左子树。

[0093] 2)否则,即当前树结点为空时,执行下列操作:

[0094] 定义SelemType型变量e,弹出栈顶元素并赋给e;

[0095] currentNodePtr指向enodePtr,t赋值为etag;

[0096] 若t＝0,即遍历完左子树,则使e.tag为1,将e压入栈;

[0097] 否则,即遍历完右子树,完成以下操作:

[0098] 若当前指针的整型数值等于x,则输出该元素,当栈不为空时,循环执行弹出栈顶元素并输出;

[0099] 当前指针置空,即currentNodePtr。

[0100] 最后应说明的是:以上所述仅为本发明的优选实施例而已,并不用于限制本发明,尽管参照前述实施例对本发明进行了详细的说明,对于本领域的技术人员来说,其依然可以对前述各实施例所记载的技术方案进行修改,或者对其中部分技术特征进行等同替换,凡在本发明的精神和原则之内,所作的任何修改、等同替换、改进等,均应包含在本发明的保护范围之内。

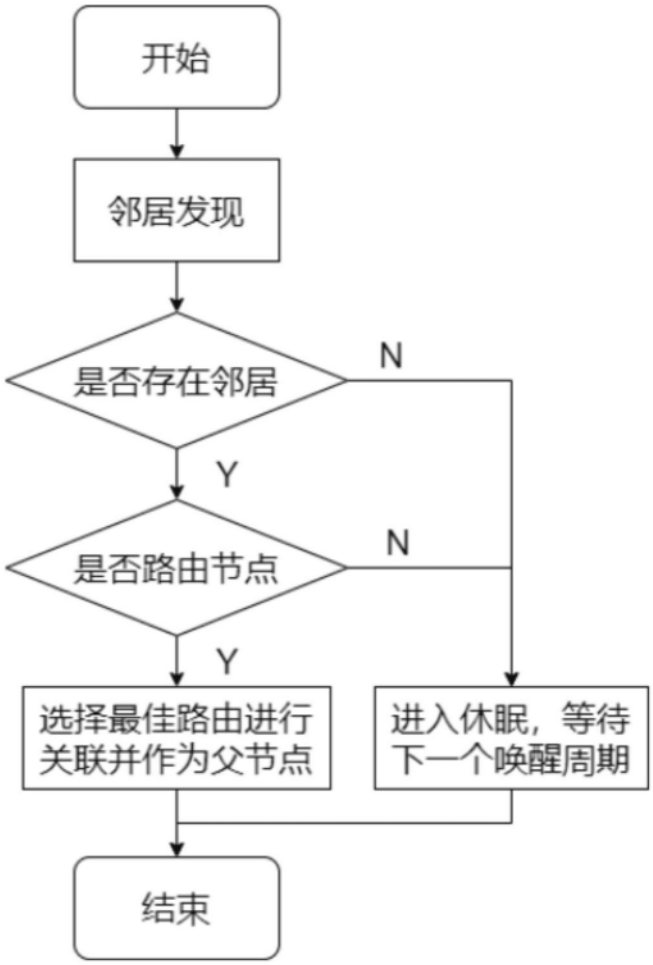

开始
邻居发现
是否存在邻居
N
Y
是否路由节点
N
Y
选择最佳路由进行关联并作为父节点
进入休眠，等待下一个唤醒周期
结束

图1

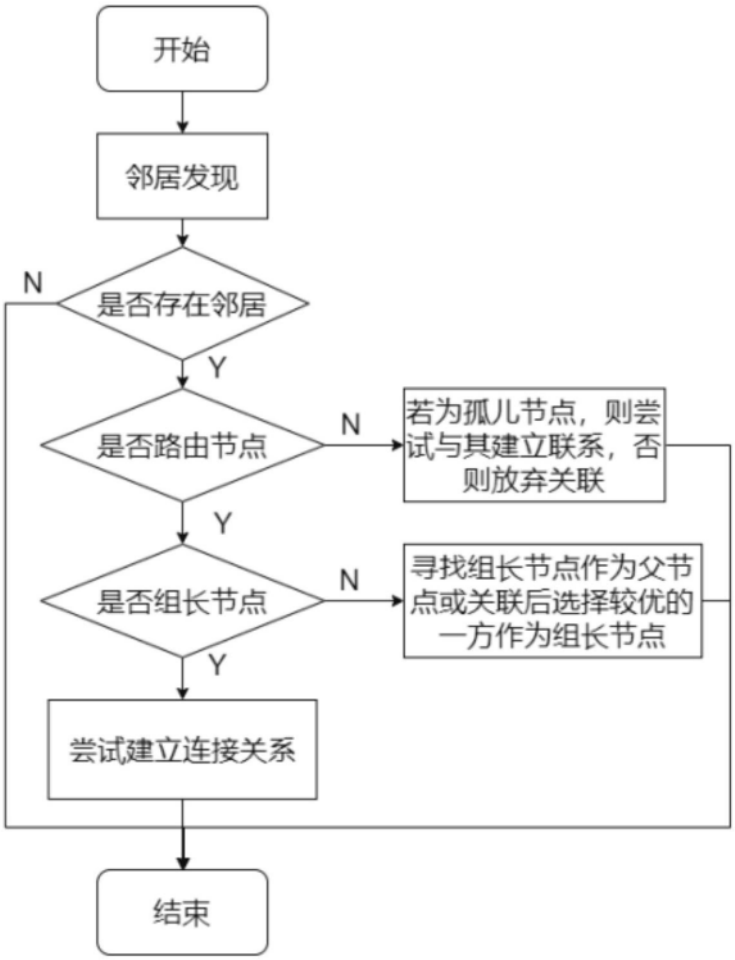

开始
邻居发现
是否存在邻居
N
Y
是否路由节点
N
若为孤儿节点，则尝试与其建立联系，否则放弃关联
Y
是否组长节点
N
寻找组长节点作为父节点或关联后选择较优的一方作为组长节点
Y
尝试建立连接关系
结束

图2

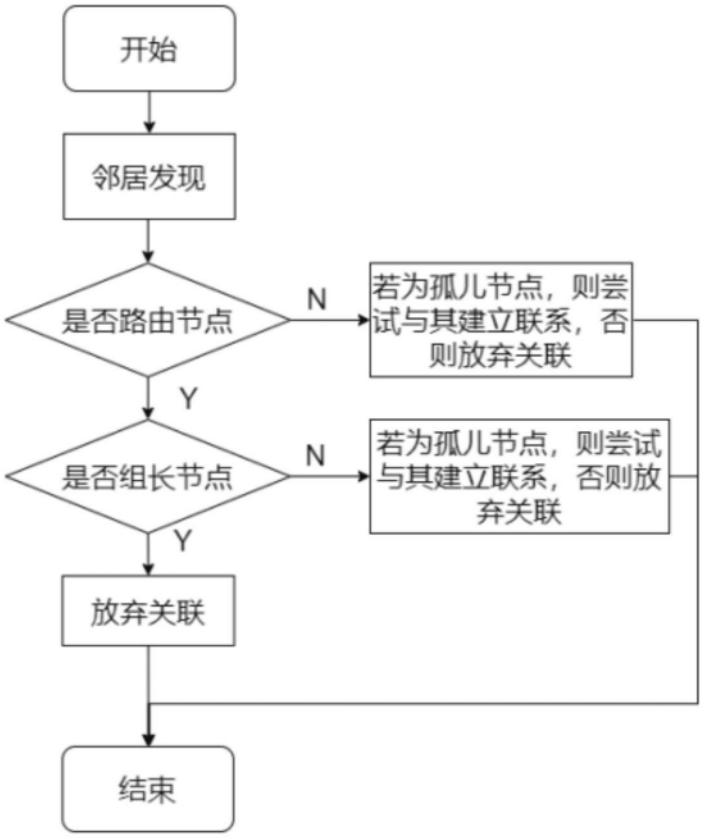

开始
邻居发现
是否路由节点
N
若为孤儿节点，则尝试与其建立联系，否则放弃关联
Y
是否组长节点
N
若为孤儿节点，则尝试与其建立联系，否则放弃关联
Y
放弃关联
结束

图3

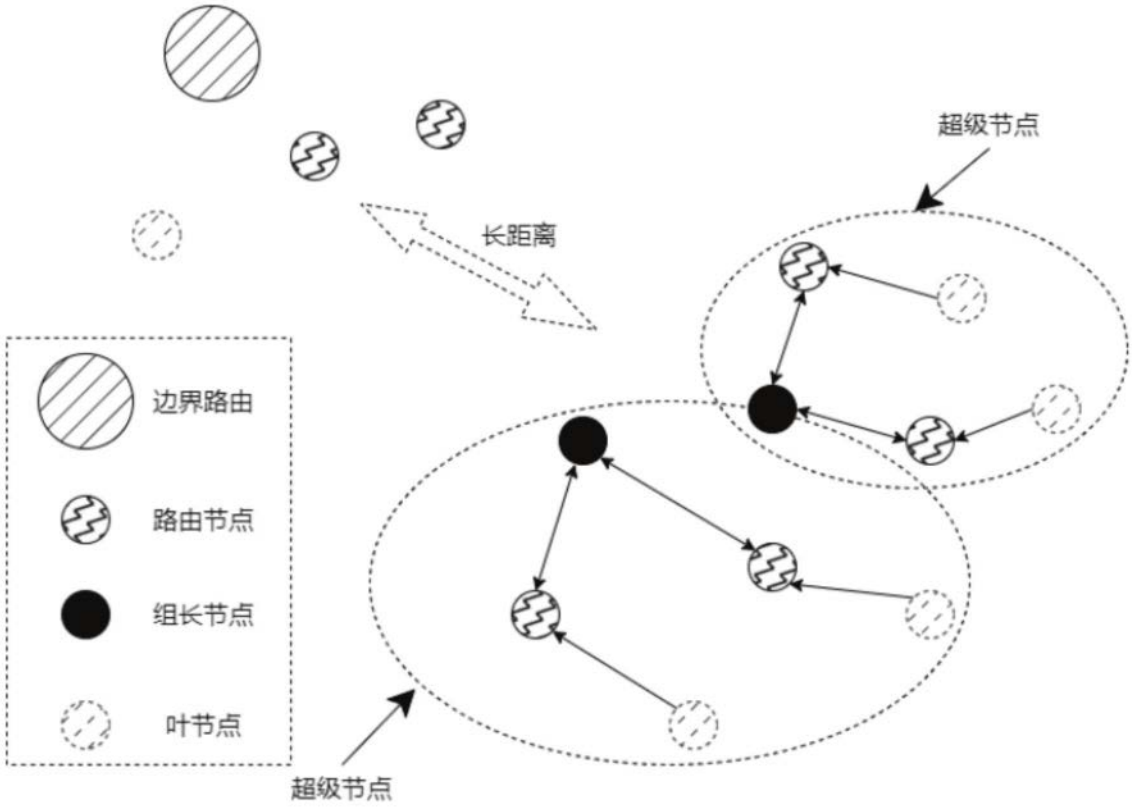


图4