移动低占空比传感网中基于多信标消息的低时延邻居发现算法

梁俊斌[1,2]，周翔[1,2]，马方强[1,2]，蒋婵[1,2]，何宗键[3]

（1. 广西大学计算机与电子信息学院，广西 南宁 530004；2. 广西大学广西多媒体通信与网络技术重点实验室，广西 南宁 530004；
3. 奥克兰大学网络研究中心，奥克兰 1142）

摘 要：邻居发现可以使网络中的节点通过简单的信息交互彼此发现对方，适用于新型的移动低占空比传感网（MLDC-WSN）。然而，由于 MLDC-WSN 中节点具有可随机移动、睡眠等特性，使网络拓扑频繁发生改变，导致部分节点需要花费很多的能量和时间才能发现邻居。如何使网络中的全部节点实现快速邻居发现是目前研究的难点问题。为了解决这个难题，提出了一种新的基于多信标消息的低时延邻居发现算法，节点通过发送一种简短的信标消息来寻找自己的邻居，并且通过调整信标消息发送的时刻及发送的次数，从而获取较低的邻居发现时延。最后，通过定量分析和仿真实验发现，与已有算法相比，该算法能够在 MLDC-WSN 中以更小的能耗、更低的时延和更大的概率发现全部的邻居节点。

关键词：移动低占空比传感网；邻居发现；低时延；信标消息

中图分类号：TP393

文献标识码：A

doi: 10.11959/j.issn.1000−436x.2019139

Low-latency neighbor discovery algorithm based on multi-beacon message in mobile low-duty-cycle sensor network

LIANG Junbin[1,2], ZHOU Xiang[1,2], MA Fangqiang[1,2], JIANG Chan[1,2], HE Zongjian[3]

1. School of Computer and Electronics Information, Guangxi University, Nanning 530004, China
2. Guangxi Key Laboratory of Multimedia Communications and Network Technology, Guangxi University, Nanning 530004, China
3. Network Research Center, the University of Auckland, Auckland 1142, New Zealand

Abstract: Neighbor discovery enables nodes in the networks to discover each other through simple information interaction, which was suitable for the new mobile low duty cycle sensor network (MLDC-WSN). However, because the nodes in MLDC-WSN can move randomly and sleep, the network topology was changed frequently, which results in that some nodes need a lot of energy and time to find their neighbors. How to realize fast neighbor discovery for all nodes in the network was a difficult problem in current research. To solve this problem, a new low-latency neighbor discovery algorithm based on multi-beacon messages was proposed. In this algorithm, the nodes were discovered by sending a short beacon message through their neighbor nodes, and by adjusting the time and frequency of beacon message sent, a lower neighbor discovery delay was obtained. Eventually, through quantitative analysis and simulation experiments, it is found that compared with existing algorithms, this algorithm can find all neighbor nodes in MLDC-WSN with less energy consumption, lower latency and greater probability.

Key words: mobile low-duty-cycle sensor network, neighbor discovery, low latency, beacon message

---

收稿日期：2018–12–26；**修回日期**：2019–05–17

基金项目：国家自然科学基金资助项目（No.61562005，No.61762010）；广西自然科学基金资助项目（No.2018GXNSFBA281169）；广西高等学校千名中青年骨干教师培育计划基金资助项目（桂教人（2017）49 号）

Foundation Items: The National Natural Science Foundation of China (No.61562005, No.61762010), The Natural Science Foundation of Guangxi Province (No.2018GXNSFBA281169), The Cultivation Plan for Thousands of Young and Middle-Aged Backbone Teachers in Guangxi Higher Education School (Guangxi Education People (2017) No.49)

1　引言

移动无线传感器网络（MWSN, mobile wireless senor network）是由大量具有移动特性且资源（能量、通信、存储、计算等）有限的传感器节点采用无线多跳通信方式自组织而成的网络，已经被广泛应用于科学探索、目标跟踪等领域[1-2]。然而，在真实场景中，由于节点存在大量的空闲监听时段，导致能量消耗过大。目前，低占空比的工作模式逐渐成为节点节省能量的重要方式之一。具有低占空比工作模式的移动无线传感器网络，被称为移动低占空比传感网（MLDC-WSN, mobile low-duty-cycle wireless senor network）。近年来，MLDC-WSN 越来越受到国内外学者的关注。

在 MLDC-WSN 中，由于节点往往通过随机移动来监测目标事件，使网络拓扑结构不断改变，导致节点需要花费较多的能量和时间来发现其邻居节点[3-4]。此外，节点采用低占空比工作模式，即在一个工作周期内，节点绝大部分时间处于睡眠状态，仅极小部分时间处于苏醒状态，这使节点间需要等待较长时间来发现彼此[5]。因此，如何低能耗、低时延地发现邻居节点是一项具有挑战性的工作，这直接关系到 MLDC-WSN 的寿命和网络工作的保障[6-7]。

目前，在已有的低占空比网络研究中，邻居节点相互发现的方式主要是通过设计一定的调度算法来使一对节点同时唤醒并进行消息传递，如基于 birthday 邻居发现协议[8]、异步发现协议[9]、disco[10]、U-connect[11]算法等，这些算法虽然确保了一对节点能够在限定时延内完成彼此发现，却需要 2 个节点等待较长的时间以同时被动式苏醒，然后进行相互发现。但是，如果节点间可以共享自身睡眠调度表，则可以通过主动苏醒的方式来实现低时延、低能耗的相互发现。

基于上述原理，本文提出了一种基于多信标的低时延邻居发现算法，该算法中节点通过信息交互获取彼此下次苏醒的时刻，采用主动苏醒的方式来进行相互发现，分为如下 2 个过程：节点可以在睡眠时间段内发送信标（beacon）消息，与邻居节点共享下次的苏醒时刻；苏醒的邻居节点接收 beacon 消息，调整下次发送 beacon 消息的时间，实现低时延的双向邻居发现。例如节点 a 在苏醒时间段内收到另一节点 b 在睡眠时间段内发送的信标消息，获知节点 b 是其邻居节点；然后，节点 a 在节点 b 苏醒的时间段内发送信标消息，使节点 b 发现节点 a，从而实现双向的邻居发现。

本文的主要贡献包括 2 个部分：1) 基于信标消息的邻居发现方式，提出了一种基于多信标消息的低时延邻居发现算法 ready-go；2) 通过理论分析和仿真实验，将已有算法与提出算法从网络能耗、发现概率、发现时延等性能方面进行了对比和分析。

2　相关工作

目前，可以根据网络节点苏醒的方式，将已经存在的邻居发现算法分为被动式邻居发现和主动式邻居发现 2 种方式。其中，被动式邻居发现算法是指每个节点按照预先设定好的睡眠调度表有规律地苏醒，从而进行相互发现，如 ENDP（efficient neighbor discovery protocol）、Disco、U-connect、C-tours quorum 等算法；主动式邻居发现算法是指苏醒节点通过获知邻居节点下次苏醒时间，主动调整自身苏醒时间，从而进行相互发现，如 Searchlight、Nihao、Griassdi、SPND（energy saving selectively proactive neighbor discovery）、GBND（group-based neighbor discovery）等算法。

2.1　被动式邻居发现算法

早期，Dutta 等[10]提出了 Disco 发现算法，将中国剩余定理（CRT, Chinese remainder theorem）应用到占空比传感网中来实现邻居节点的快速发现。具体步骤如下：每个节点选取 2 个素数，将其中一个作为自身的工作周期，如果节点自身的计时器能够整除其工作周期，则节点苏醒。该算法保证了任意 2 个节点能够在一定的时限内相互发现，但未考虑节点因时钟偏移而导致发现时延过大的问题。

针对 Disco 算法存在的问题，Kohvakka 等[12]提出了一种同步低占空比传感网中高效邻居发现协议（ENDP）。该协议的基本思想是：1) 节点相互发现，构建邻居集合；2) 节点工作一段时间后，苏醒节点在两跳通信范围内发送信标消息，其中，信标消息包括节点有效负载和同步信息；3) 节点先进行时钟同步，再根据节点负载信息选择剩余能量大的苏醒节点进行数据传输。该算法虽然保证了节点在限定时延内的发现概率，平衡了节点的能量负载，但是依然存在平均发现时延较高的问题。

在 Disco 算法的基础上，Kandhalu 等[11]提出了 U-connect 邻居发现算法。与 Disco 算法相似，该算法首先选择一个素数作为节点工作周期，并将连续 N 个时隙构建成一个 $q\times q$ 的网格矩阵。接着，节点从矩阵中选取第 $i(1\leqslant i\leqslant q)$ 列和第 $j(1\leqslant j\leqslant q)$ 行的一半时隙作为自身苏醒时间段，从而提高了节点间同时苏醒的概率，降低了节点发现时延。但是，该算法明显增大了节点的占空比，使空闲监听的能耗增大。

针对 U-connect 算法存在的不足，陈良银等[13]提出了一种新的 C-torus quorum 发现算法，可以应用在对称和非对称网络环境下进行邻居节点的发现。具体过程如下：首先，节点选择一个 $h\times w$ 网格矩阵的元素总数 n 作为工作周期，其中 h 或 w 为素数；其次，从网格矩阵中任意选取某一列 $c(1\leqslant c<h)$，再从这一列中选取某一行 $r(1\leqslant r\leqslant w)$；再次，在该行的 c 列元素后选取 $\frac{w}{2}$ 个元素，如果没有 $\frac{w}{2}$ 个元素，则返回 r 行的第一个元素继续选择；最后，节点将 $w-1$ 个元素作为苏醒时间段，再进行邻居节点的发现。该算法基于 CRT 理论，保证了在一定时限内任意 2 个节点必然能够相互发现。但在最坏情况下的限定发现时延较大。

从上述几种发现算法可以看出，网络节点均采用被动式邻居发现的方式，通过一定的策略来降低 2 个节点同时苏醒的等待时间。这种方式的优点是任意 2 个节点间不需要信息交互，但也存在任意 2 个节点间可能需要等待很长的时间才能完成相互发现的缺点。

2.2 主动式邻居发现算法

Bakht 等[14]提出了 Searchlight 算法，主要利用 beacon 消息来实现网络节点间的快速双向发现，其基本思想是：首先，利用节点苏醒时刻之间的偏移量来设计 beacon 消息的发送时刻；然后，通过拓展被动苏醒时间段的长度，使节点在被动苏醒时间段的前后一部分时间也主动苏醒；最后，当所有节点都有相同的占空比时，通过计算节点的苏醒偏移量来选择采用确定性方法或概率性方法，提高算法在平均情况下的性能。但是，为了保证该算法的发现概率和时延基本呈线性增长的关系，各个节点间需要相同的占空比。

在 Searchlight 算法的基础上，针对节点时间段槽位不对齐等问题，Qiu 等[15]提出了一种新的邻居发现协议 Nihao，可以通过发送更多的信标消息来减少空闲监听。该协议基于“多说少听（TMLL, talk more listen less）”的原则，节点将一个占空比生命周期划分为 n 个时间段，且每隔 m 个时间段节点发送一个 beacon 消息，从而实现与邻居节点的快速发现。由于在每个生命周期内发送多个 beacon 消息，该协议会导致较高的通信能耗，且未考虑节点时钟偏移等问题。

针对 Nihao 协议存在的不足，Kindt 等[16]针对网络节点时钟不同步的情况，提出了一种新的 Griasdi 协议，节点之间通过相互合作的方式来降低平均双向发现时延。该协议的具体过程是节点在苏醒时间段内发送或接收包含有自身下次苏醒时刻的 beacon 消息，获得该 beacon 消息的邻居节点将适当地调整发送 beacon 消息的计划，以便在其对应的邻居节点下次苏醒时刻发送 beacon 消息。该协议使节点充分利用发送 beacon 消息来减少与邻居节点双向发现的平均时延，但是它在限定时延内的发现概率不高。

在 Griasdi 协议的基础上，梁俊斌等[17]提出了一种低能耗的主动式邻居发现（SPND）算法，通过对移动节点通信范围内的邻居节点进行划分，来减少节点主动苏醒的次数，从而降低网络能耗。该算法的具体步骤是：首先，根据节点移动速度预测下一时刻自身邻居集合；然后，节点将自身通信范围划分为 $[0,R-\Delta S]$ 和 $[R-\Delta S,R]$ 这2 个区域，这样对应的邻居节点集合被划分为 G_1 和 G_2 这 2 个集合；最后，节点在下一次苏醒时刻，只需要与集合 G_2 中的节点相互发现即可。该算法虽然降低了网络通信能耗，但却依然无法保证较低的邻居发现时延。

此外，Chen 等[18]提出了一种基于群组的发现（GBND）方法，节点之间通过建立一个时间表参考机制来加快相互发现的速度。该方法的基本思想是：节点 B 首先使用传统双向发现方式获知节点 A 的唤醒时刻；然后，节点 B 主动将已知节点 C 的唤醒调度表发送给节点 A，使节点 A 可以间接地发现节点 C，从而将相互发现的节点构成一个集合；最后，节点 B 有选择性地将集合中的部分节点的睡眠调度表分享给新加入的节点。但是，该算法在最坏情况下的节点发现时延较大，且成功率不高。

综上所述，现存的主动式邻居发现算法虽然能够在平均时延和能耗方面取得较好的平衡，但是依

然存在在最坏情况下发现时延较大、限定时延下发现概率不高等问题。

3　系统模型和问题描述

首先对将要使用到的术语进行定义，然后给出带有信标消息的 MLDC-WSN 模型及问题描述。

3.1　相关定义

定义 1　beacon 消息。节点周期性地广播给邻居节点的信标信息。该信息包含的数据量较小，常用于传感器节点间的相互发现。

定义 2　单向邻居发现[19]。一个节点接收到另一个节点发送的 beacon 消息，从而获知该节点为自己的邻居节点。

定义 3　双向邻居发现。2 个节点彼此收到了对方发送的 beacon 消息，从而实现双向的邻居发现。

3.2　系统模型

假设一个边长为 $L\times L$ 的矩形区域部署着 n 个移动传感器节点，每个节点均采用低占空比工作模式，通信半径均为 R。将 MLDC-WSN 视为一个无向图，即 $G_{\text{net}}=\{V,E\}$，其中，V 为节点的集合，E 为节点间通信边的集合。此外，MLDC-WSN 具备以下特点。

1) 所有节点的能量有限且不可再生。

2) 所有节点采用随机路点（RWP, random way point）模型进行移动[20]。

3) 所有节点大部分时间处于睡眠（sleep）状态，关闭除去定时器以外所有其他功能模块，只在少部分时间内保持活动（active）状态来进行事件感知和数据传输。

3.3　问题描述

在传统的 MLDC-WSN 中，节点处于 active 状态下才能完成事件感知和数据传输，如图 1 所示，其中 T 表示工作周期。在第 2、9、16 个时间段内，节点 i 处于 active 状态，可以主动发送或接收数据，而在其余时间段内将无法发送或接收数据。

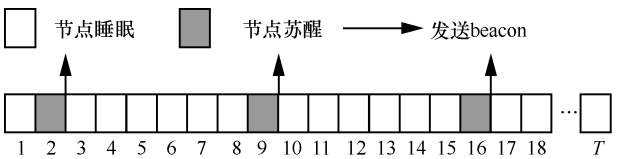

图 1　传统的网络模型示意

因此，如果节点 i 与节点 j 在某个时刻 t 处能够相互发现，则需要满足 2 个条件。

1) 在某一时间段 Δt 内，节点 i 与节点 j 之间的距离 d_{ij} 始终不超过通信半径，即 $d_{ij}\leqslant R$。

2) 在时间段 Δt 内，节点 i 与节点 j 需要获知彼此下次苏醒时刻，从而进行相互发现。

由上述 2 个条件可知，任意 2 个节点从可以相互通信的时刻 t_0 开始到完成相互发现的时刻 $t+\Delta t$ 为止，它们相互发现的时延 δ 为

$$\delta=t+\Delta t-t_0 \tag{1}$$

在一个工作周期 T 中，节点的占空比 DC 为节点苏醒的时间段加上节点发送信标消息所占的时间之和与 T 的比值，即 $\text{DC}=\dfrac{t_{\text{active}}+\sum t_{\text{beacon}}}{T}$。从而可以获得节点在一个工作周期内的能耗为 $E_{\text{cost}}=\text{DC}\times T\times H$，其中 H 为单位时间内节点的平均能耗。

因此，可以将 MLDC-WSN 中低时延的邻居发现问题归纳为以最小能耗 E_{cost} 实现发现时延 L 最小化的问题，即

$$\{\min E_{\text{cost}},\min L\} \tag{2}$$

如果给定节点的生命周期 T，则可以将问题转化为以最小低占空 DC 实现发现时延 L 最小化的问题，结合式(2)，可以表示为

$$\{\min \text{DC},\min L\} \tag{3}$$

传统 MLDC-WSN 很难获得较小的节点邻居发现时延，如何使网络中的全部节点以低能耗实现快速的邻居发现是本文研究的课题。

针对上述问题，本文首先改进了 MLDC-WSN 模型的设计，然后提出了一种新的基于多信标消息的低时延邻居发现算法 ready-go。

4　算法设计

4.1　算法的基本思想

在 Nihao 算法中，节点通过增加 beacon 消息的发送次数来减少自身苏醒的次数，从而实现能耗的降低；Griassdi 算法通过一种互助的方式，使节点在实现了单向邻居发现后，能尽快地实现双向邻居发现。在这 2 种算法思想的基础上，本文提出了一种基于多信标消息的低时延邻居发现算法即通过持续广播快速响应的方式来实现低时延和低能耗的邻居发现。“持续广播”是指节点在开始进行邻居发现时，在一段连续时间内的每个时间段的开始

片段广播一个 beacon 消息，该消息中将包含发送节点在这段连续时间之后的第一次苏醒时刻。而“快速响应”则指发送节点 i 周围的节点 j 在接收到 beacon 消息后，主动调整自身下一次 beacon 消息的发送时间，从而正好使节点 i 在苏醒时刻能够接收到节点 j 发送的 beacon 消息，实现低时延的双向邻居发现。

4.2 算法的详细设计

ready-go 算法基于 TMLL 原则进行设计，即节点可以在任意时间段（包括睡眠时间段和苏醒时间段）中发送一个 beacon 消息，而对应的邻居节点只能在苏醒时刻才能接收该 beacon 消息。在低占空比网络中，由于每个节点都拥有自己的占空比，使节点间具有不同的睡眠调度方式，且所有节点的占空比均在同一范围内。假设节点可以预先获取所有节点中最小占空比的值，则可以将 ready-go 算法分为 2 个过程：ready 过程和 go 过程。ready 过程如图 2 所示，go 过程如图 3 所示。

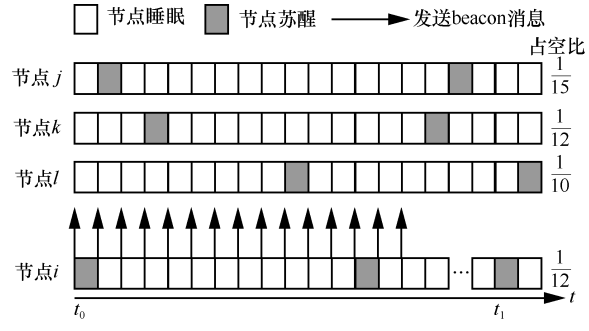

图 2 ready 过程

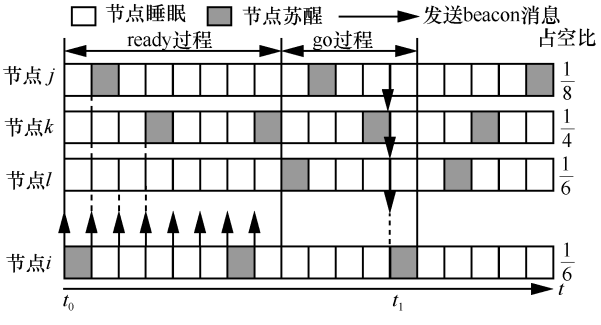

图 3 go 过程

在 ready 过程中，如果 MLDC-WSN 中任意一个节点 i 在某一时刻 t_0 处苏醒并进行邻居发现工作，那么节点 i 首先需要根据已经获取的节点占空比最小值 Γ 计算出需要连续发送 beacon 消息的次数 $N=\frac{1}{\Gamma}$。接着，发送节点 i 在之后的 N 个时间段内的开始时刻发送一个 beacon 消息，寻找自己的邻居节点。如图 2 所示，假设网络中节点占空比的最小值为 $\Gamma=\frac{1}{15}$，则节点 i 将从 t_0 开始的 15 个时间段内，在每个时间段的开始时刻发送一个 beacon 消息，用于寻找自己的邻居节点。同时，由于 Γ 是节点中最小的占空比值，在网络通信可靠的情况下，在 N 个时间段内，能够保证节点 i 周围通信范围内的节点（如 j、k、l 等）都能收到它发送的 beacon 消息。beacon 消息包含发送节点 i 在 N 个时间段后的第一次苏醒时刻 t_1 的信息。

在 go 过程中，节点 j、k、l 在收到邻居节点 i 的 beacon 消息后，会动态调整自身下一次发送 beacon 消息的时间表，以保证在节点 i 苏醒的时间段内发送 beacon 消息，从而进行双向的邻居发现。

为了使示意图更加清晰，节点占空比设置的值均稍大于实际情况中的占空比值。从图 3 中可以发现，收到节点 i 发送的 beacon 消息的节点 j、k、l，会根据 beacon 消息中包含的 t_1 时刻的信息，调整自身 beacon 消息发送的时刻，使发送的消息正好能被节点 i 在 t_1 时刻接收到。节点 i 处理完 t_1 时刻接收到的 beacon 消息后，构建在 t_1 时刻的邻居集合，完成从 t_0 时刻到 t_1 时刻的双向邻居发现过程。

此外，考虑到在实际情况中节点间存在一定的时钟偏移的情况，ready-go 算法通过为每个节点增加一次 beacon 消息的发送来进行解决，即节点 i 在 ready 过程中发送 $N+1$ 次 beacon 消息；同时，节点 j、k、l 在发送 beacon 消息时间段的开始与结束时刻，均发送一个 beacon 消息。

具体解决方案如图 4 所示，在节点存在时钟偏移的情况下，节点 i 发送的 beacon 消息仍然能被通信范围内的其他节点接收到。首先，在 ready 过程中，节点 i 增加一次 beacon 消息发送的次数；然后，在 go 过程中，为了使接收到 beacon 消息的节点 j、k、l 所发送的 beacon 消息能被节点 i 收到，节点 j、k、l 需要在 t_1 时刻时间段的开始时刻和结束时刻均发送一个 beacon 消息，以保证节点 i 在 t_1 时刻时间段内能够收到来自节点 j、k、l 发送的 beacon 消息，从而使节点间以较低的时延完成双向邻居发现。具体的算法描述如算法 1 所示。

算法 1 ready-go 算法

输入 MLDC-WSN 中每一个节点 i

输出 每个节点 i 的邻居结合 G^i

1)for (每一个节点 i) {

2)　do 获取节点占空比最小值 Γ

3)　do 计算连续发现 beacon 消息的次数 $N+1$

4)　while ($t=t_0+(N+1)\Delta t$){

5)　　do 节点 i 在一个时间段的开始时刻广播一个 beacon 消息

6)　　if (节点 j 苏醒收到 beacon 消息) {

7)　　do 节点 j 调整下一次发送 beacon 消息的时刻

8)　　} end if

9)　} end while

10)　if ($t=t_1$) {

11)　　节点 j 发送 beacon 消息

12)　　节点 i 苏醒接收节点 j 发送的 beacon 消息

13)　　节点 i 将节点 j 加入到邻居集合

14)　} end if

15) } end for

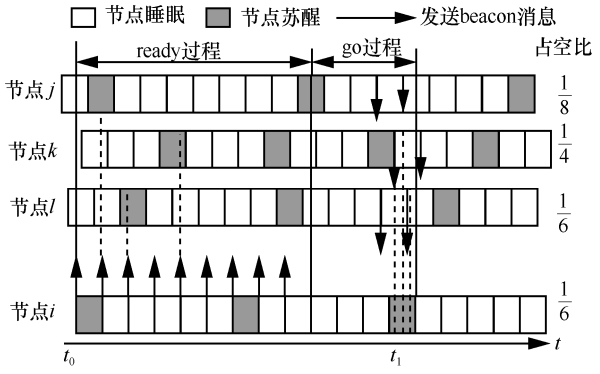

图 4　ready-go 算法示意

4.3　性能分析

4.3.1　节点发现时间表

在 ready-go、Nihao 和 Griassdi 算法中，每个节点苏醒的时间和发送 beacon 消息的时间是相互独立的。本节将使用与 U-connect 算法中相似的模型来定义 ready-go 算法中的发现时间表。

ready-go 算法中节点 i 的发现时间表被定义为 2 个二进制函数：$\psi L(i,t)$ 和 $\psi B(i,t)$，分别代表时刻 t 的苏醒时间段和发送 beacon 消息的时间段，即

$$\psi L(i,t)=\begin{cases}1, \text{节点苏醒}\\ 0, \text{节点睡眠}\end{cases}$$
$$\psi B(i,t)=\begin{cases}1, \text{节点发送一个beacon消息}\\ 0, \text{节点错误}\end{cases} \tag{4}$$

因此，节点之间的邻居发现可以通过 $\psi L(i,t)$ 和 $\psi B(i,t)$ 来定义。如果节点 i 和节点 j 在通信范围内，那么节点 i 可以单向发现邻居节点 j，可定义为 $\psi L(i,t)=1$，$\psi B(i,t)=1$；如果节点 i 和节点 j 在通信范围内，那么节点 i 和节点 j 可以实现双向发现邻居，可定义为

$$\exists t_1,t_2,\ \text{使}\psi L(i,t_1)=\psi B(j,t_1)\wedge\psi L(i,t_2)=\psi B(j,t_2) \tag{5}$$

即存在时刻 t_1、t_2，使在 t_1 时刻，节点 i 处于苏醒时间段，节点 j 处于 beacon 消息发送时间段；在 t_2 时刻，节点 i 处于 beacon 消息发送时间段，节点 j 处于苏醒时间段。

在 ready-go 算法中，节点的发现时间表可以定义为

$$\psi L(i,t)=\begin{cases}1, t=\dfrac{n}{\tau_i+1}, n\in\{0,1,\cdots,T\tau_i-1\}\\ 0, \text{其他}\end{cases}$$
$$\psi B(i,t)=\begin{cases}1, t\in\left\{0,1,\cdots,\dfrac{1}{\tau_{\min}}\right\}\cup\{t_1\}\\ 0, \text{其他}\end{cases} \tag{6}$$

其中，τ_i 表示节点 i 的苏醒时刻占空比，$\tau_{\min}$ 表示节点苏醒时刻占空比的最小值，T 表示节点 i 的工作周期，n 表示节点 i 在一个工作周期中的第 $n+1$ 次苏醒，t_1 表示节点 i 在连续发生 beacon 消息之后的第一次苏醒时刻。

在 Nihao 算法中，节点的发现时间表可以定义为

$$\psi L(i,t)=\begin{cases}1, [t]L<m\\ 0, \text{其他}\end{cases}$$
$$\psi B(i,t)=\begin{cases}1, \lceil t\rceil L=ma, a=0,1,\cdots,n-1\\ 0, \text{其他}\end{cases} \tag{7}$$

其中，L 表示 Nihao 算法中节点的最坏发现时延，为节点的工作周期长度，即 $L=T=mn$；$[t]L$ 表示 t 对 L 取模，$\lceil t\rceil L<m$ 表示节点每个工作周期的前 m 个时间段，ma 表示第 a 个 m 长度。

4.3.2　占空比时延乘积模型

通信能耗（E_{cost}）和发现概率（P）是低占空比传感网中邻居发现的重要衡量指标。将节点在一个工作周期内的数据接收/发送的数据量与传输 1 bit 数据的平均能耗的乘积作为该节点的通信能耗，即

$$E_{\text{cost}} = \text{DC} \times T \times H \times B \tag{8}$$

其中，DC 为节点一个工作周期内的占空比值，T 为节点的工作周期，H 为节点发送/接收 1 bit 数据的平均能耗，B 为传输带宽。

由文献[17]可知，节点 i 在时间 t 内被动苏醒发现邻居节点 k 的概率为

$$P_S^i(t) = 1 - \prod_{n-1}^{j=0}(1 - P_{ij}(t)) \tag{9}$$

其中，$P_{ij}(t) = \begin{cases} f(j,k,l), t \in [0, \delta_{\max}^{jk}) \\ 1, t \geqslant \delta_{\max}^{jk} \end{cases}$，表示节点 i、k 在 t 时间内相互发现概率；$f(j,k,l)$ 表示节点 i、k 能相互发现的概率；$\delta_{\max}^{jk}$ 表示节点 i、k 之间的最大发现时延。

针对式(8)，对于任意给定的节点，$T \times H \times B$ 通常为一个常数 c，即 $E_{\text{cost}} = \text{cDC}$；针对式(9)，可以通过计算概率密度函数 $p_S^i(t)$ 获得任意节点 i 发现所有邻居节点的期望时延，即

$$E_S^i(t) = \int_0^{\delta_{\min}^i} t p_S^i(t) \text{d}t$$

其中，$\delta_{\min}^i$ 表示节点 i 发现通信范围内的第一个邻居节点的时延。

经上述分析可以发现，占空比和发现概率分别是影响节点通信能耗和发现时延的关键因素，使占空比和最坏情况发现时延已经成为衡量邻居发现算法优劣的重要指标。

由于在 TMLL 模型下，节点的通信能耗主要发生在接收和发送消息中，因此，在给定工作周期 T 的情况下，节点的占空比 DC 包括苏醒时间段和睡眠时隙段发送 beacon 消息的时间，具体表示为

$$\text{DC} = \frac{1}{T}\left(\sum_{t=0}^{T-1}\psi L(i,t) + \alpha\left(\sum_{t=0}^{T-1}\psi B(i,t) - N_{\text{c}}\right)\right) \tag{10}$$

其中，N_{c} 为在一个工作周期中节点苏醒时发送 beacon 消息的时间隙个数，即 $\psi L(i,t) = \psi B(i,t) = 1$；$\sum_{t=0}^{T-1}\psi B(i,t) - N_{\text{c}}$ 是为了避免重复计算 beacon 消息发送个数；α 为节点发送一个 beacon 消息所需时间段长度的比例。

在 WSN 中，节点的发现时延和通信能耗往往是一对对立的指标，更低的时延意味着更多的能耗。为了方便比较算法性能，文献[11,15-16]提出了将占空比与时延的乘积作为评价邻居发现算法性能的指标。如果 2 个指标都小，则它们的乘积也小，算法性能更好。文中采用同样的评价模型将 ready-go 算法与其他典型的邻居发现算法进行性能对比，占空比和时延乘积 Λ 为

$$\Lambda = \text{DC} \times L \tag{11}$$

其中，L 表示最坏情况下的节点邻居发现时延。

下面将给出 ready-go 算法与 Nihao 算法的 DC 值与 L 值，在 ready-go 算法中，节点的最坏发现时延 L 可以表示为

$$L = \frac{2}{\tau_{\min}} + 1 \approx 2m + 1 \tag{12}$$

节点的占空比 DC 可以表示为

$$\begin{aligned} &\text{DC}_{\text{rgeady-go}} = \\ &\frac{1}{T}\left(\sum_{t=0}^{T-1}\psi L(i,t) + \alpha\left(\sum_{t=0}^{T-1}\psi B(i,t) - N_{\text{c}}\right)\right) = \\ &\frac{1}{T}\left(T\tau_i + \alpha\left(\frac{1}{\tau_{\min}} - 1\right)\right) = \tau_i + \frac{\alpha}{T\tau_{\min}} - \frac{2\alpha}{T} \end{aligned} \tag{13}$$

在 Nihao 算法中，节点的最坏发现时延 $L = T = mn$，节点苏醒时刻的占空比 $\tau_i = \frac{m}{T}$，则占空比 DC 为

$$\text{DC}_{\text{Nihao}} = \frac{m + \alpha(n-1)}{mn} \approx \frac{m + \alpha n}{mn} \tag{14}$$

因此，Nihao 算法的占空比时延乘积为

$$\Lambda_{\text{Nihao}} = \text{DC} \times L = \frac{m + \alpha n}{mn} mn = m + \alpha n \tag{15}$$

对于 ready-go 算法而言，为了便于比较，在不影响正确性的情况下，可以取 $\tau_i = \tau_{\min} = \frac{m}{T}$ 且 $T = mn$，则 $\text{DC}_{\text{ready-go}}$ 可以化简为

$$\begin{aligned} &\text{DC}_{\text{ready-go}} = \tau_i + \frac{\alpha}{T\tau_{\min}} - \frac{2\alpha}{T} = \tau_{\min} + \frac{\alpha}{T\tau_{\min}} - \frac{2\alpha}{T} = \\ &\frac{m}{mn} + \frac{n\alpha}{mn} - \frac{2\alpha}{mn} = \frac{m + \alpha(n-2)}{mn} \end{aligned} \tag{16}$$

则 ready-go 算法的占空比时延乘积为

$$\Lambda_{\text{ready-go}} = \text{DC} \times L = \frac{m+\alpha(n-2)}{mn}(2m+1) \quad (17)$$

将式(14)与式(13)相减可得 $\text{DC}_{\text{Nihao}} - \text{DC}_{\text{ready-go}} = \frac{2\alpha}{mn} > 0$，即 $\text{DC}_{\text{ready-go}} < \text{DC}_{\text{Nihao}}$。同时，将 2 种算法的最坏发现时延相除，可得 $\frac{L_{\text{ready-go}}}{L_{\text{Nihao}}} = \frac{2m+1}{mn} \approx \frac{2}{n}$。当 $n > 2$ 时，则有 $L_{\text{ready-go}} < L_{\text{Nihao}}$；当 $n \leqslant 2$ 时，则有 $L_{\text{ready-go}} \geqslant L_{\text{Nihao}}$。

经过上述分析，可以发现，当 $n > 2$ 时，有 $\text{DC}_{\text{ready-go}} < \text{DC}_{\text{Nihao}}$ 且 $L_{\text{ready-go}} < L_{\text{Nihao}}$，即 ready-go 算法在占空比和最坏情况下的发现时延都小于 Nihao 算法，使 $\Lambda_{\text{ready-go}}$ 明显优于 Λ_{Nihao}。实际上，当 $\tau_i > \tau_{\min}$ 时，ready-go 算法可以得到更优的最坏情况下节点发现时延值，以实现更低的占空比和时延乘积。

4.3.3　占空比时延乘积对比

将本文所提的 ready-go 算法与前文提到的典型邻居发现算法的占空比时延乘积进行对比。根据算法类型的不同，在计算中节点占空比和最坏情况下发现时延时，各算法尽可能使用相同的参数，具体对比情况如表 1 所示。

表 1　典型算法的占空比时延乘积对比

算法	参数	DC	L	Λ
C-torus quorum	n	$\frac{2}{n}$	n^2	$2n$
Disco	p_1、p_2	$\frac{1}{p_1}+\frac{1}{2}$	p_1+p_2	p_1+p_2
Searchlight	t	$\frac{2}{t}$	$\frac{t^2}{2}$	t
Nihao	m、n	$\frac{m+\alpha n}{mn}$	mn	$m+\alpha n$
ready-go	m、n	$\frac{m+\alpha(n-2)}{mn}$	$2m+1$	$\frac{(m+\alpha(n-2))(2m+1)}{mn}$
U-connect	q	$\frac{3}{2q}$	q^2	$\frac{3q}{2}$

表 1 中，n 为 C-tours quorum 算法中节点的工作周期的长度；p_1、p_2 为 Disco 算法中选取的 2 个素数，节点随机选取 2 个素数之一作为自己的工作周期；$T = q^2$ 为 U-connect 算法中节点的工作周期；$T = \frac{t^2}{2}$ 为 Searchlight 算法中节点的工作周期，即将节点的工作周期分解成一个宽是高两倍的矩阵；Nihao 算法中，节点工作周期为 $T = mn$，m、n 根据具体的网络场景调整，m 表示节点连续苏醒的时间段个数，n 表示节点 beacon 消息发送的次数；ready-go 算法中，m 等价于节点的工作周期 T 乘以节点的苏醒时刻占空比 τ_i，$n = \frac{1}{\tau_i}$。

4.3.4　多信标平均能耗

在 TMLL 模型中，节点发送 beacon 的时刻与节点的苏醒时刻是相互独立的。假设每个节点在一个工作周期内的苏醒时间段都相同，由式(10)可知，节点在睡眠时间段发送 beacon 消息的能耗为 $\alpha\sum_{t=0}^{|T|-1}\psi B(i,t)w$，节点在苏醒时发送信标的能耗为 $\alpha w N_{\text{c}}$，则节点在一个工作周期中发送 beacon 消息的能耗为

$$C_{\text{beacon}} = \alpha\sum_{t=0}^{|T|-1}\left(\psi B(i,t) + N_{\text{c}}\right)B \quad (18)$$

其中，B 为节点的传输数据带宽，$|T|$为一个工作周期中的时间段个数。

由于节点发送 beacon 消息的时隙只占一个时间段中的极少一部分。结合式(10)和式(18)可得，$C_{\text{non\_beacon}} \gg C_{\text{beacon}}$，其中 $C_{\text{non\_beacon}}$ 为节点在一个工作周期内其余工作的能耗（即 $C_{\text{non\_beacon}} = aB\left(\psi L(i,t) - N_{\text{c}}\right)$）。因此，在实际应用中，可以通过对多个工作周期内节点睡眠时刻发送 beacon 消息的能耗和苏醒工作时的能耗进行累加，求取平均能耗来替代节点的多信标平均能耗。

5　实验及对比分析

本节通过实验仿真来测试 ready-go 算法的性能，并将该算法的实验数据与前文提到的几种典型算法的数据进行对比。为了降低实验误差造成的影响，每组实验进行 1 000 次，数据取平均值。

5.1　实验环境

仿真实验基于 C++语言开发的一个仿真平台，在该平台上实现了 ready-go 算法及其他的典型算法。在实验中，假设 WLDC-WSN 部署在 500 m×500 m 的矩形区域内，由完全随机分布的 500 个移动低占空比节点组成，如图 5 所示。其中，黑色实心原点为传感器节点，R 为节点通信半径。假设节点的苏醒占空比值为 1%，每个时间段长度为

10 ms，其中，节点的苏醒占空比不包括节点在睡眠时间发送 beacon 消息的时隙。WLDC-WSN 中节点默认通信半径为 100 m，节点间信息失效时间设置为 5 000 个时间段，节点发送一个 beacon 消息的时长为 0.54 ms。根据路径损耗模型[21]，在距离 d 上计算发送 1 bit 数据的能量损耗为 $E_t = \varepsilon d^{\lambda} + E_{\text{ele}}$，接收 1 bit 数据的能量损耗为 $E_r = E_{\text{ele}}$。其中，$\varepsilon = 10^{-11}$，$\lambda = 2$，$E_{\text{ele}} = 50 \times 10^{-9}$ J。此外，为方便进行算法性能的对比，ready-go 算法与其他几种算法的实验参数保持一致。

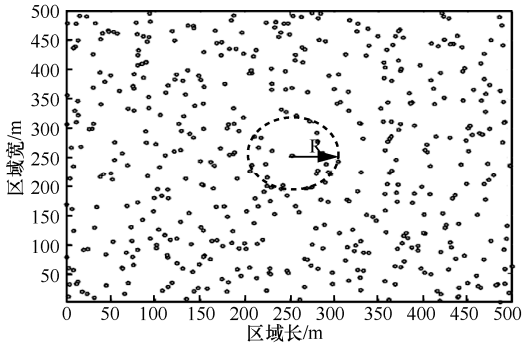

图 5 WLDC-WSN 分布示意

5.2 性能表现

本节分别从限定时延的发现概率、节点最坏情况下的发现时延、网络平均能耗等方面进行了测试，并将 ready-go 算法与其他典型邻居发现算法进行了对比。

5.2.1 限定时延的发现概率

在这组实验中，根据 4.3.2 节的发现概率模型，设定节点苏醒占空比为 1.4%。在给定限定时延发现概率的情况下，测试了 ready-go 算法的数据，并与 SPND、Nihao、Griassdi 这 3 种算法的限定时延发现概率进行了对比，结果如图 6 所示。

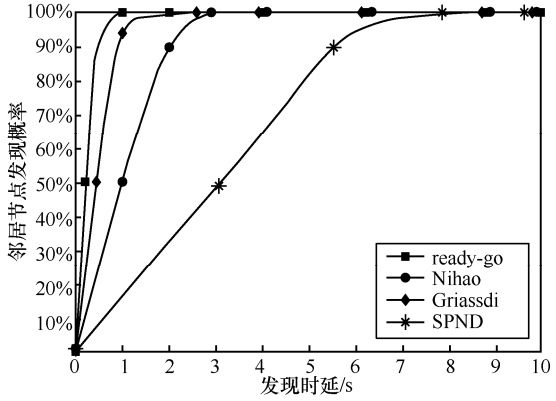

图 6 限定时延的发现概率

从图 6 中可以发现，根据算法限定时延的发现概率收敛到 100%的速度，将上述 4 种算法大致分为 3 个梯队：ready-go 算法和 Griassdi 算法属于第一梯队，这 2 种算法都采用了主动调整 beacon 消息发送时刻的策略，使节点能够在很低的发现时延下获得高的发现概率；第二梯队为采用了多 beacon 消息发送的 Nihao 算法，也能够使节点在较低的发现时延下得到高发现概率；SPND 算法由于 beacon 消息仅在苏醒时间段内发送，且节点未采用主动苏醒方式，导致其的概率收敛速度较慢，因此排在第三梯队。

经过上述对比可知，ready-go 算法由于采用了持续广播快速响应的思想，使节点能在最短的时间内找到所有的邻居节点，这使它的限定时延发现概率的收敛速度优于其他 3 种算法。

5.2.2 节点最坏发现时延

在这组实验中，通过取节点苏醒占空比值的范围为 0.2%~1.4%，测试了节点发现所有邻居节点的发现时延，实验结果如图 7 所示。

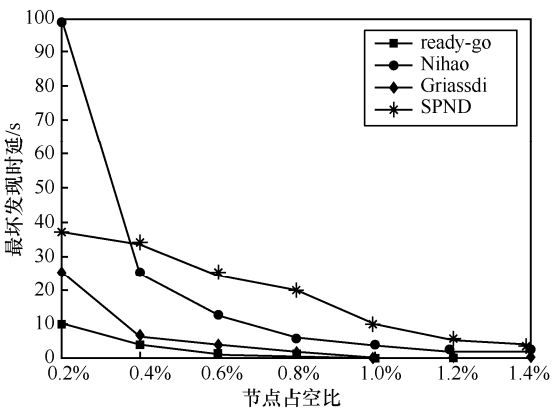

图 7 不同占空比下的发现时延

从图 7 中可以发现，随着节点苏醒占空比值的增大，SPND 算法的最坏发现时延明显大于其他 3 种算法的最坏发现时延，ready-go 算法和 Griassdi 算法能取得较小的最坏发现时延。同时，ready-go 算法与其他算法相比，在所有占空比取值情况下都能取得最低的最坏发现时延。因此，经过实验对比可知，采用共享苏醒时刻策略的 ready-go 算法，可以使本该处于睡眠状态的节点动态调整自身苏醒的时刻，从而增大了苏醒状态收到 beacon 消息的概率，降低了最坏发现时延。

5.2.3 网络平均能耗

在网络能耗的测试中，每次选取网络中任意一

个节点进行测试以评估网络能耗，并通过多次测试取平均值来减少实验误差。首先，设置节点的传输带宽为 $B = 256\ \text{kbit/s}$，则节点发送一次 beacon 消息的数据量为 $S_{\text{beacon}} = B\alpha$ 。

如图 8 所示，节点的占空比范围为 0.5%~5%，4 种邻居发现算法的通信能耗均随着节点苏醒占空比的增加而呈下降趋势。其中，SPND 算法的通信能耗明显高于其他 3 种邻居发现算法。而 ready-go 算法与 Nihao 和 Griassdi 算法相比，它的节点平均能耗的下降速率最大，且当节点的占空比大于 3.5% 时，ready-go 算法的平均能耗低于其他 3 种发现算法的平均能耗。这表明，随着节点占空比的增加，ready-go 算法发送 beacon 消息的次数会大幅度减少，使 ready-go 算法节点的平均能耗优于其他几种算法。当在较低占空比时，ready-go 算法的平均能耗虽然较高，但是节点发送信标的能耗只占一个苏醒时间段能耗的极小一部分，因此并不会对节点正常工作时的能耗造成过大的影响。

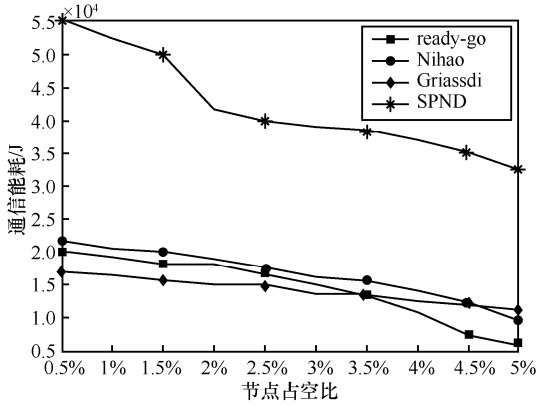

图 8　节点平均能耗

6　结束语

本文提出了一种基于多信标消息的低时延邻居发现算法 ready-go。该算法通过获取网络中节点苏醒占空比最小值，计算出持续发送 beacon 消息的次数；节点在开始进行邻居发现工作时，持续发送多个 beacon 消息，保证周围的潜在邻居节点都能收到，并通过调整自身 beacon 消息的发送时刻，从而快速实现双向的邻居发现。理论分析和仿真实验表明，ready-go 算法与其他典型算法相比，能获得更小的邻居发现时延。同时，节点能够实现限定时延的发现概率比现有典型算法的收敛速度更快。

参考文献：

[1] MENG T, WU F, CHEN G. Code-based neighbor discovery protocols in mobile wireless networks[J]. IEEE Transactions on Networking, 2016, 24(2): 806-819.

[2] GUO S, YANG Y, WANG C. DaGCM: a concurrent data uploading framework for mobile data gathering in wireless sensor networks[J]. IEEE Transactions on Mobile Computing, 2016, 15(3): 610-626.

[3] KOHVAKKA M, SUHONEN J, KUORILEHTO M, et al. Energy-efficient neighbor discovery protocol for mobile wireless sensor networks[J]. Ad Hoc Networks, 2009, 7(1):24-41.

[4] CHEN H, LOU W, WANG Z, et al. On achieving asynchronous energy-efficient neighbor discovery for mobile sensor networks[J]. IEEE Transactions on Emerging Topics in Computing, 2016,6(4):553-565.

[5] BOAVENTURA A S, CARVALHO N B. A low-power wakeup radio for application in WSN-based indoor location systems[J]. International Journal of Wireless Information Networks, 2013, 20(1):67-73.

[6] BURGHAL D, TEHRANI A, MOLISCH A. On expected neighbor discovery time with prior information: modeling, bounds and optimization[J]. IEEE Transactions on Wireless Communications, 2018, 17(1): 339-351.

[7] JEON W S, DWIJAKSARA M H, DONG G J. Performance analysis of neighbor discovery process in bluetooth low-energy networks[J]. IEEE Transactions on Vehicular Technology, 2017, 66(2):1865-1871.

[8] MCGLYNN M J, BORBASH S A. Birthday protocols for low energy deployment and flexible neighbor discovery in Ad Hoc wireless networks[C]// The ACM International Symposium on Mobile Ad Hoc Networking and Computing. ACM, 2001:137-145.

[9] CHEN S, RUSSELL A, JIN R, et al. Asynchronous neighbor discovery on duty-cycled mobile devices: integer and non-integer schedules[C]//The ACM International Symposium on Mobile Ad Hoc Networking and Computing. ACM, 2015: 47-56.

[10] DUTTA P, CULLER D. Practical asynchronous neighbor discovery and rendezvous for mobile sensing applications[C]//The ACM Conference on Embedded Network Sensor Systems. ACM, 2008: 71-84.

[11] KANDHALU A, LAKSHMANAN K, RAJKUMAR R. U-connect:a low-latency energy-efficient asynchronous neighbor discovery protocol[C]// International Conference on Information Processing in Sensor Networks. ACM,2010:350-361.

[12] KOHVAKKA M, SUHONEN J, KUORILEHTO M, et al. Energy-efficient neighbor discovery protocol for mobile wireless sensor networks[J]. Ad Hoc Networks, 2009, 7(1): 24-41.

[13] 陈良银，颜秉姝，张靖宇，等. 移动低占空比传感网邻居发现算法[J]. 软件学报, 2014(6): 1352-1368.
CHEN L Y, YAN B S, ZHANG J Y, et al. Neighbor discovery algorithm for mobile low duty ratio sensor networks [J]. Journal of Software, 2014 (6): 1352-1368.

[14] BAKHT M, TROWER M, KRAVETS R H. Searchlight:won't you be

my neighbor?[C]//The Annual International Conference on Mobile Computing and Networking. ACM, 2012:185-196.

[15] QIU Y, LI S, XU X, et al. Talk more listen less: energy-efficient neighbor discovery in wireless sensor networks[C]//The IEEE International Conference on Computer Communications. IEEE, 2016:1-9.

[16] KINDT P H, YUNGE D, REINERTH G, et al. Griassdi: mutually assisted slotless neighbor discovery[C]// ACM/IEEE International Conference on Information Processing in Sensor Networks. IEEE, 2017:93-104.

[17] 梁俊斌, 周翔, 李陶深. 移动低占空比无线传感网中低能耗的主动邻居发现算法[J]. 通信学报, 2018,39(4):45-55.
LIANG J B, ZHOU X, LI T S. Active neighbor discovery algorithms for low energy consumption in mobile low duty ratio wireless sensor networks [J]. Journal on Communications, 2018,39(4): 45-55.

[18] CHEN L, SHU Y, GU Y, et al. Group-based neighbor discovery in low-duty-cycle mobile sensor networks[J]. IEEE Transactions on Mobile Computing, 2016, 15(8): 1996-2009.

[19] CHEN L, BIAN K. Neighbor discovery in mobile sensing applications: a comprehensive survey[J]. Ad Hoc Networks, 2016, 48(9):38-52.

[20] YOON J, LIU M, NOBLE B. Random waypoint considered harmful[C]//The IEEE Computer and Communications Societies. IEEE, 2003: 1312-1321.

[21] LUO L, CAO Q, HUANG C, et al. EnviroMic: towards cooperative storage and retrieval in audio sensor networks[C]//International Conference on Distributed Computing Systems. IEEE, 2007: 34-44.

[作者简介]

梁俊斌（1979– ），男，广西南宁人，博士，广西大学教授，主要研究方向为无线传感器网络及分布式系统。

周翔（1995– ），男，湖北鄂州人，广西大学硕士生，主要研究方向为无线传感器网络。

马方强（1993– ），男，陕西咸阳人，广西大学硕士生，主要研究方向为无线传感器网络。

蒋婵（1980– ），女，广西合浦人，广西大学博士生，主要研究方向为无线传感器网络。

何宗键（1981– ），男，山东临沂人，博士，奥克兰大学网络研究中心研究员，主要研究方向为普适计算和无线移动网络。