第37卷 第14期
Vol.37 No.14

计 算 机 工 程
Computer Engineering

2011年7月
July 2011

·专栏·

文章编号：1000—3428(2011)14—0027—04 文献标识码：A 中图分类号：TP393

基于簇头冗余的WSN不等规模分簇路由

张 擎[a]，柴乔林[b]

(山东大学 a. 艺术学院；b. 计算机科学与技术学院，济南 250100)

摘 要：在采用多跳通信方式的分簇无线传感器网络中，接近sink点的簇头负担较重，容易过早耗尽能量而失效，造成网络分割，降低网络传输质量。针对该问题，在研究已有工作的基础上提出一种结合不等分簇和冗余簇头机制的路由算法。经仿真验证，该算法在延长网络生命周期、节约网络能耗和提高数据传输可靠性方面有较好的性能。

关键词：无线传感器网络；不等规模分簇；动态簇头冗余；能量高效；路由算法

Unequal Scaled Clustering Routing for WSN Based on Redundancy of Cluster Headers

ZHAGN Qing[a], CHAI Qiao-lin[b]

(a. School of Fine Arts; b. School of Computer Science & Technology, Shandong University, Jinan 250100, China)

【Abstract】In multi-hop Wireless Sensor Network(WSN) using clustered routing algorithm, the nearer to the sink node, the more load is put on the headers. As a result, headers near the sink node will quickly lose their energy and become unavailable which leads to network-partitioning and lower data transmission quality. In order to solve this problem, based on the analysis of existing researches, a new routing algorithm is proposed, which is based on both Unequal scaled Clusters and Redundancy of Headers(UCRH) algorithm. According to simulation results, UCRH algorithm overwhelms in prolonging network's lifetime, reducing energy cost and supporting reliable data delivery.

【Key words】Wireless Sensor Network(WSN); unequal scaled clustering; dynamic redundancy of cluster headers; energy efficient; routing algorithm

DOI: 10.3969/j.issn.1000-3428.2011.14.007

1 概述

分簇路由具有拓扑管理方便、能量利用高效、数据融合简单等优点，成为当前WSN重点研究的路由技术。研究表明，LEACH[1]、TEEN[2]等经典协议要求网络中所有节点都可以直接与*sink*节点通信，这样的限制大多数情况下是不能满足的。由于能量和节点传输距离的限制，远离*sink*点的节点需要借助其他节点，以多跳通信方式传送数据。这种WSN称为多跳WSN。在此类网络中，簇头不但要负责簇内数据的传输和处理还要担负对簇间过路数据的传输。越接近*sink*点的簇头由于更多的过路数据而更快地耗尽能量，造成*sink*点附近的Hot-Spot问题[3]。这一问题会破坏网络连通性和关键区域的数据采集，同时降低网络传输的可靠性，引入耗费巨大的簇重组过程。

迄今为止，已有很多分簇路由协议通过在接近*sink*点的地区产生更多更小规模的分簇来解决Hot-Spot问题。其中较为经典的是EEUC[3]协议和USC[4]协议。目前，基于以上经典算法，也出现了很多改进的研究成果。但这些研究没有考虑到WSN节点分布不均匀这一特点，只依靠节点与*sink*点的远近程度来控制分簇规模，依然在很大程度上造成负载不均衡。另外，分簇形成后，没有良好的机制在簇头负担过重或出现拥塞时给予及时监测和补救，造成簇头单点能量快速消耗的同时也降低了网络传输的可靠性。有部分研究提出簇头冗余机制[5-6]，通过簇头轮换实现维护簇头能量，处理簇头数据拥塞或故障。簇头轮换的实现分为集中式和分布式2种。在集中式方式中，由簇头收集和维护能量信息，并根据当前能量情况主动提出簇头轮换；在分布式方式中，由簇内所有节点维护当前簇头和冗余簇头的能量信息，并对簇头轮换做出判断。这2种方式各有其缺点。集中式要求簇头实时获取和计算冗余簇头的能量，给繁忙的簇头带来了额外负担，在出现拥塞时这种方式更显得笨拙。分布式方式中簇员不断交互更新信息，额外消耗了大量节点能量和带宽。当前集中式和分布式簇头轮换机制皆以节点能量作为冗余簇头轮换的依据，在簇头节点出现数据拥塞或突发故障时冗余节点的反应总会出现延迟，无法做到实时快速处理。

基于此，本文提出一种综合考虑节点位置、局部节点密度和节点能量的不等规模分簇算法，在此基础上设计一种高效的动态簇头冗余路由机制来保护簇头节点能量，保障数据传输质量。

2 网络模型和冗余簇头

2.1 网络模型

传感器网络具有以下特征：

(1)具有唯一ID的传感器节点随机部署在感知区域，由于部署的随机性，区域内节点分布并不均匀。

(2)网络中有一个与有线网络直接相连，有持续的电量供应的*sink*点。*sink*点知道网络初始部署情况。

(3)节点无法自由移动。

(4)除*sink*点外的其他节点具有相同的处理和通信能力，传输距离有限，最大传输距离为R。节点在网络中的地位平

作者简介：张 擎(1982－)，女，博士研究生，主研方向：网络与分布式技术；柴乔林，教授

收稿日期：2010-12-24 **E-mail：**zq-abby@163.com

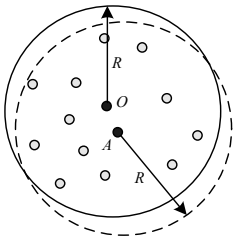
等，连接对称，在相互通信范围之内的节点互为邻居节点。

(5)节点可根据接收信号强弱估计与邻居节点间的距离。

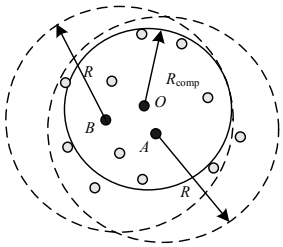
(6)各节点初始能量相同，记为 $P_{\max}$。节点能量有限，能够知道自己的剩余能量。收发单位比特的能耗参照文献[7]中的模型。

2.2 冗余簇头

本文引入冗余簇头这一概念描述路由机制。冗余簇头是指分簇过程完成后，除了簇头节点之外，能与所有簇内节点进行直接通信的节点。如图 1(a)所示，节点 A 为簇 O 中的一个冗余簇头。针对冗余簇头，有一点需要说明，即在节点都以 R 为分簇半径进行分簇时，冗余簇头存在的概率并不大。但随着分簇半径的减小，冗余簇头存在的概率有所增加。这是因为分簇规模变小，簇内节点在位置上更加聚集，而冗余簇头是以距离 R 判断是否与簇内节点可直接通信，因此，冗余簇头出现的可能性有所增加，越接近 *sink* 点，分簇规模越小，冗余簇头出现越频繁。如图 1(b)所示，当簇头节点 O 的分簇范围缩小为 R_{comp} 时，簇内节点 B 也成为冗余簇头。文献[6]证明有，当分簇半径缩小到 $R/2$ 时，簇内节点是全联通的，即除了簇头节点外，簇内所有节点都是冗余簇头。因此，在接近 *sink* 点的区域内保证形成存在冗余簇头的簇结构是可行的。



(a)分簇范围 R 中的冗余簇头节点



(b)分簇范围缩小后簇内冗余簇头节点

图 1 分簇前后的冗余簇头节点

3 算法描述

3.1 不等规模分簇算法

分簇算法中采用不等规模分簇的思想，综合考虑节点位置和局部密度信息确定节点的分簇半径，并依据节点的剩余能量和分簇半径内冗余簇头的数量竞选簇头。具体算法如下：

Step1 网络初始化。*sink* 点向整个网络发送信号，通知所有节点网络中距离其最远和最近的节点与其相隔的距离，分别记为 $d_{\max}$ 和 $d_{\min}$。所有节点通过感知接收信号的强度估算自己距 *sink* 点的距离。节点 S 到 *sink* 点的距离记为 $d(S,sink)$。分别用 $P_{\max}$ 和 P_{now} 记录节点的初始能量和现在的剩余能量。$P_{\min}$ 为系统变量，节点能量降低为 $P_{\min}$ 时视其为无效节点。

Step2 信息交互。网络中每个节点向一跳邻居节点发送自己的信息数据 Info_MSG。Info_MSG 中包含本节点 ID 和剩余能量。节点收到所有邻居节点的 Info_MSG 后，维护邻居列表 ALIST，并向所有邻居发送包含本节点所有邻居节点 ID 的列表 AIDLIST。至此，网络中所有节点获得了以下信息：包含邻居节点剩余能量和 ID 的列表 ALIST，针对每一个邻居的邻居节点 ID 列表。

Step3 计算分簇半径和冗余簇头数。首先，所有节点统计每个邻居节点的邻居节点数，如果把 R 距离范围的面积看作统一的单位面积，则可以把此单位面积内的节点数，也就是节点一跳范围邻居个数作为节点的局部密度。记任意节点 S 的邻居节点(包括 S 节点)中最多的邻节点个数为 $NR_{\max}$，最少的邻节点个数为 $NR_{\min}$，S 节点的邻节点个数为 NR。虽然根据已获得信息可以一步计算出分簇半径，但为了易于理解，将分 2 步描述对分簇半径的计算。

(1)利用节点位置信息，采用式(1)计算初始分簇半径，记为 R^1_{comp}。

$$R^1_{\text{comp}}=(1-c\times\frac{d_{\max}-d(S,sink)}{d_{\max}-d_{\min}})\times R \tag{1}$$

(2)利用节点的局部密度信息对初始分簇半径进行调整，得到最终分簇半径 R_{comp}。

$$R_{\text{comp}}=(1-\alpha\times\frac{NR-NR_{\min}}{NR_{\max}-NR_{\min}})\times R^1_{\text{comp}} \tag{2}$$

其中，$d(S,sink)$ 是节点 S 到 *sink* 点的距离；c 和 α 是介于(0,1)之间的数。在式(1)中，$d(S,sink)$ 在 $d_{\min}$ 和 $d_{\max}$ 之间变化，$(1-c)R\leqslant R^1_{\text{comp}}\leqslant R$ 且 R^1_{comp} 随 $d(S,sink)$ 的减小而减小，越接近 *sink* 点的节点分簇半径越小。在式(2)中，当节点局部密度 NR 介于周围节点的最大局部密度 $NR_{\max}$ 和最小局部密度 $NR_{\min}$ 之间时，$(1-\alpha)R^1\leqslant R_{\text{comp}}\leqslant R^1$ 且 R_{comp} 随 NR 的增大而减小，局部节点密度大的节点的分簇半径小。这样，在接近 *sink* 点越近的区域产生越多分簇结构的基础上进一步缩小密度大的节点的分簇范围，从而使簇头节点的负载更加均衡合理。

由于已经维护了邻居节点的所有邻节点信息并确定了分簇半径 R_{comp}，因此很容易对冗余节点进行判断。首先确定位于本节点 R_{comp} 半径范围内的邻节点，记为簇内节点。然后逐一查询这些簇内节点的邻居列表，如果某簇内节点的邻居列表中包含所有这些簇内节点，则此簇内节点为冗余节点。记冗余节点的个数为 n_{RHN}。

Step4 选举簇头，生成簇结构。本算法在选举簇头时综合考虑节点的剩余能量和分簇半径中冗余簇头的个数。剩余能量大的节点在以前的任务中消耗较少，并且有更多的能力处理数据传输任务，被选择成为簇头有利于均衡负载和维持网络的稳定性。从整个网络看，接近 *sink* 点的区域传输任务重，容易出现簇头故障或数据拥塞，更需要冗余簇头发挥作用，越接近 *sink* 节点的区域越应提高冗余簇头这一分簇依据的重要性。为减小计算的复杂度，本算法采用一个较为简单的抉择方式。对任意节点 S，当 $d(S,sink)\leqslant\frac{1}{2}d_{\max}$ 时，即 S 与 *sink* 点的距离不大于最远节点与 *sink* 点距离的 1/2 时，只有 S 的冗余节点数 $n_{\text{RHN}}>0$ 时，S 才参与簇头竞争。相反，当 $d(S,sink)>\frac{1}{2}d_{\max}$ 时，所有节点参与竞争。具体分簇过程如下：

在分簇过程中，节点处于未定(idle)、簇头(head)和簇员(member)3 种状态。初始时节点的状态都为 idle。如果节点能够参与竞争且能量大于所有邻居节点，则宣布成为簇头，将状态置为 head，发送带有节点 ID 和分簇半径 R_{comp} 的簇头声明信息 HEAD_MSG。节点收到 HEAD_MSG，如果状态为 idle 并且自己与簇头距离小于 R_{comp}，则加入簇，置自身状态为

member 并发送包含本节点 ID 和簇头 ID 的加入消息 JOIN_MSG。簇头节点收到 JOIN_MSG 消息，若节点加入自己建立的簇，则将其插入簇成员列表 ML 中。若节点没有收到符合 R_{comp} 距离要求的 HEAD_MSG 消息，却收到邻节点发出的 JOIN_MSG 消息，则标记该邻节点的状态为 member，并同时设定时间 T(T 与节点剩余能量呈反比)，开始计时。当 T 时间到达时，如果一直未接收到符合 R_{comp} 距离要求的 HEAD_MSG 消息并且本节点是可以参与竞争的节点，则该节点成为簇头。如果本节点是不可以参与竞争的节点，则再等待 T 时间，在此时间内，如果自己的距离与 *sink* 节点的距离小于 $d_{max}/2$，则调整自己的分簇半径，使分簇半径内至少存在一个冗余簇头，等待结束仍然没有收到任何簇头消息，则宣布自己成为簇头。分簇过程如图 2 所示。上述过程结束后，所有节点都已处于 head 或 member 状态，分簇过程结束。

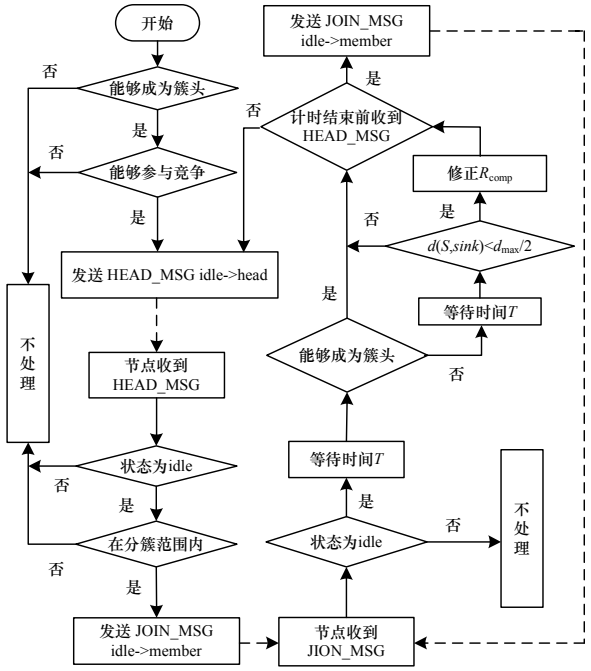


图 2　UCRH 算法簇结构生成过程

3.2　簇头冗余机制

本文提出一种不同于集中方式和分布方式的动态簇头冗余机制。新机制中簇头节点无须持续监测和计算其他节点的能量，大大减轻了簇头的负担。成员节点间也不需要交换信息以维护其他节点能量，最大程度地减小引入簇头冗余机制的额外消耗。针对数据拥塞和簇头故障，分别采用拥塞通知和故障监听的方式做到快速响应和处理。具体实现如下：

分簇结束后，如果本簇内存在冗余簇头，则簇头节点发信息给冗余簇头进行任命，收到任命消息后的节点成为冗余簇头。簇头冗余机制包括以下 3 种情况。

(1)能量变化触发簇头轮换：在数据传输过程中，簇头无需进行任何监听或计算，每隔一段时间(簇头轮换间隔时间)，簇头向冗余簇头发送包含自己能量的信息。冗余簇头针对这一信息进行判断。如果簇头能量比自己当前的能量低，则开启倒计时，计时时间与自身当前能量呈反比。计时结束后，冗余簇头将成为新簇头，发送带有当前自身能量和 ID 的簇头轮换命令 ChHMsg，簇内所有节点收到 ChHMsg 命令后改变簇头 ID，完成簇头轮换。如果冗余簇头在计时结束前收到其他冗余簇头发来的 ChHMsg 命令，则结束计时，接受新簇头。新簇头产生后原簇头将成为冗余簇头。如果簇头节点的能量小于 P(能量低于 P 的节点不再担任簇头)，则马上广播退位信息，并成为普通簇成员。这时如果簇内存在冗余簇头则进行簇头轮换，如果簇内没有冗余簇头则通知 *sink* 点，发起重新分簇过程。能量不足 P 的冗余簇头需发送消息通知当前簇头将其在冗余簇头中除名，并成为普通簇员。

(2)数据拥塞触发簇头轮换：数据拥塞往往发生在簇头节点。但以往研究中单纯进行簇头轮换无法缓解数据拥塞，这是因为节点的处理能力相同，簇头轮换后仍然会在新簇头发生数据拥塞。这里，当簇头节点发生数据拥塞时(簇头节点根据输出数据缓冲队列中的数据量和收到重传包的数量判断是否出现拥塞)，在分簇范围广播拥塞信号。收到拥塞信号的冗余节点立刻担负起簇头的职责，并进行通知。成员节点将数据发送给距离自己最近的冗余簇头节点，以缓解簇头的负担。此过程持续到下一轮簇头轮换。

(3)簇头故障触发簇头轮换：簇头负担较重时容易出现故障而失效，这时如果不立刻进行处理，会严重影响数据传输，造成丢包和延迟。以往的簇头冗余机制对这种故障的反应速度较慢，故障被发现后触发重分簇过程。这里采取一种故障监听方案进行快速响应，避免进行簇重组。当一段时间中冗余簇头监听到有簇员传给簇头节点信息而簇头节点没有信息传出的时候，则判断簇头节点出现故障。冗余簇头立刻通过竞争进行簇头轮换，选举新簇头接管数据传输任务。

这里需要说明的是，在分簇过程中当节点与 *sink* 点距离大于 $d_{max}/2$ 时并未保证其所成簇内有冗余节点。这是因为距 *sink* 点越远，节点的簇间过路数据负载越小，另外，本算法在分簇时依据节点局部密度进一步缩小了分簇范围，限制了簇内节点个数，也一定程度上减少了簇内数据负担。所以，距 *sink* 点远的簇头节点负担较小，无须强加簇头冗余机制，而一定程度上增加分簇的重叠度。对于不存在冗余簇头的簇，簇头冗余机制将不再进行。

4　仿真结果和分析

仿真实验在网络生命周期、网络整体耗能和网络传输可靠性 3 个方面将本文提出的 UCRH 算法与经典的不等规模分簇算法——EEUC 算法，以及文献[6]中提出的基于集中式簇头轮换机制的 EERC 算法进行比较，验证 UCRH 算法的优越性。这里，网络生命周期指从网络开始运行到网络中出现因能量不足而失效的节点为止的时间。仿真中通过比较失效节点出现的时间和数量来衡量这一指标。

仿真实验采用 Omnet++3.2p1 仿真器，网络覆盖面积 600 m×600 m，距 *sink* 点最近的节点踞其距离约为 40 m，最远节点踞其距离约为 800 m。网络节点数目设置为 600 个，设置节点最大传输距离 R =60 m，数据包长度为 128 bit，控制包长 64 bit。节点的初始能量均为 10 000 J。收发单位比特的能耗参照文献[1]中的模型。取 $c=0.6$, $\alpha=0.8$。实验中，每隔 1 s 有 100 个随机节点构造数据包向 *sink* 点发送，每隔 10 s，有另外 40 个节点发送数据包，以造成突发数据。设 P=2 000 J，即当节点的能量降低到总能量的 1/5 时，视其为低能量节点，不再担当簇头。当节点能量降低为 100 J 时，视其为失效节点。

图 3 为 3 种算法失效节点出现时间和数量情况图。就失效节点出现的时间来看，UCRH 算法和 EERC 算法明显优于 EEUC 算法。分簇结束后，EEUC 算法没有采取任何保护簇头节点能量的机制，因此，死亡节点出现的时间最早。UCRH 算法根据节点密度进一步均衡簇头负载，并采用更加节省簇

头能量的簇头冗余机制，在保护关键节点能量方面表现最优。

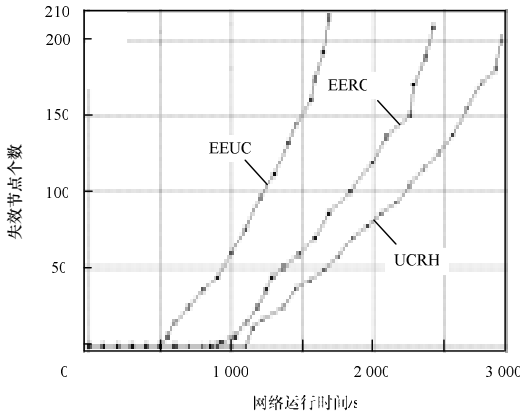


图 3 失效节点情况比较

图 4 为 3 种算法的网络整体耗能比较。UCRH 算法在节省网络整体耗能方面表现最优。在 UCRH 和 EERC 算法中，分簇阶段维护冗余簇头和数据传输阶段引入簇头冗余机制为网络增加了额外消耗。因此在网络初始运行阶段，UCRH 和 EERC 算法比 EEUC 算法消耗了更多的能量。但在数据传输开始后，特别是网络运行 10 s 产生突发数据后，由于 UCRH 算法和 EERC 算法通过簇头冗余机制减小出现拥塞和故障时的数据重传，因此在网络整体耗能方面体现出了优势，这种优势随着网络运行时间的推迟会越发明显。UCRH 算法避免了簇头向冗余节点发送冗余簇头列表，并且对拥塞和簇头故障处理更加有效，因此，在网络节能方面优于 EERC 算法。

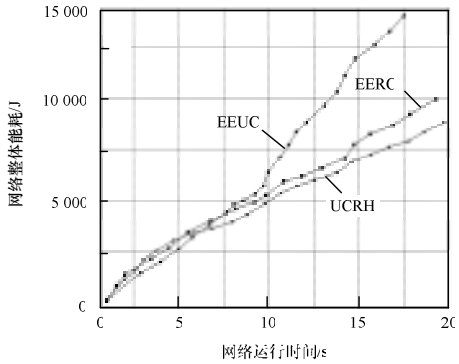


图 4 网络耗能情况比较

图 5 显示了 3 种算法在数据传输量(这里的数据传输量是指 *sink* 节点接收到的数据量)达到 1 000 个、5 000 个和 10 000 个数据包时统计的数据丢包率的比较。

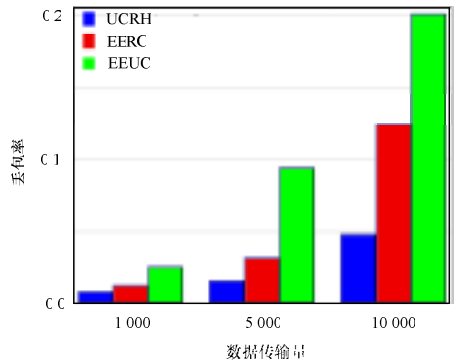


图 5 丢包率情况比较

从图 5 可以看出，3 种算法的丢包率均随着传输数据量的增大而有所增加，但 UCRH 算法的丢包率明显小于其他 2 个算法，而且随传输数据的增加，这种优势越发明显。由于没有采取任何补救措施，EEUC 协议的丢包率最高，EERC 算法通过动态簇头轮换机制一定程度上减少了丢包率。

本文提出的 UCRH 算法通过更加高效的措施对数据拥塞和簇头故障进行实时监控和有效处理，使丢包率控制在一个较低的水平，在数据传输可靠性方面较其他 2 个算法有明显优势。

5 结束语

本文分析当前多跳 WSN 分簇路由协议的研究现状和不足，提出一种结合不等规模分簇和簇头冗余机制的 WSN 分簇路由算法(UCRH 算法)。

算法在分簇阶段综合节点的位置和节点的局部密度信息控制节点的分簇半径。分簇结束后，提出一种更加高效的簇头冗余机制，在引入较少额外消耗的前提下保护簇头节点能量，实时有效地处理拥塞和簇头故障，减少消耗巨大的重新分簇过程。

目前，针对结合不等规模分簇和冗余簇头机制的 WSN 分簇路由算法，尚有很多问题有待探讨，如如何更加合理地控制分簇规模、进行高效的冗余簇头维护、选择更加合理的动态簇头冗余机制等。并且，算法需进一步对簇间路由机制的能量高效性和安全可靠性进行研究。

参考文献

[1] Heinzelman W, Chandrakasan A, Balakrishnan H. Energy-efficient Communication Protocol for Wireless Micro Sensor Networks[C]// Proc. of the 33rd Annual Hawaii International Conference on System Sciences. [S. l.]: IEEE Computer Society, 2000: 3005-3014.

[2] Manjeshwar A, Grawal D P. TEEN: A Protocol for Enhanced Efficiency in Wireless Sensor Networks[C]//Proc. of the 15th Parallel and Distributed Processing Symposium. San Francisco, USA: IEEE Computer Society, 2001: 2009-2015.

[3] Li Chengfa, Ye Mao, Chen Guihai, et al. An Energy-efficient Unequal Clustering Mechanism for Wireless Sensor Networks[C]// Proc. of the 2nd IEEE International Conference on Mobile Ad-hoc and Sensor Systems. Washington D. C., USA: IEEE Press, 2005: 7-10.

[4] Soro S, Heinzelman W B. Prolonging the Lifetime of Wireless Sensor Networks via Unequal Clustering[C]//Proc. of the 19th IEEE International Parallel and Distributed Processing Symposium. [S. l.]: IEEE Press, 2005.

[5] 彭绍亮, 李姗姗. CREED: 一种高可靠的能量高效的传感器网络分簇方法[J]. 计算机工程与科学, 2008, 30(4): 98-99.

[6] 张 擎, 曲德祥, 柴乔林. 基于完全图的能量高效可靠分簇算法[J]. 计算机工程, 2010, 36(5): 91-94.

[7] 郑少仁, 王海涛, 赵志峰, 等. Ad Hoc 网络技术[M]. 北京: 人民邮电出版社, 2005.

编辑 顾逸斐