第 33 卷 第 8 期 计 算 机 工 程 2007 年 4 月
Vol.33 No.8 Computer Engineering April 2007
·网络与通信· 文章编号：1000—3428(2007)08—0115—03 文献标识码：A 中图分类号：TP393

传感器网络随机睡眠节点调度算法研究及实现

何朝笋，石高涛，廖明宏
(哈尔滨工业大学计算机科学与技术学院，哈尔滨 150001)

摘 要：传感器网络是由大量形体较小、能源受限并且配置有计算能力和无线通信能力的传感器节点以 Ad Hoc 方式组成。传感器节点能源有限性、高密度配置的节点包冲突率高等问题使得节点调度成为必然。该文提出了一种基于动态邻居节点信息的随机睡眠调度机制，并将算法做成了一个可供重复使用的组件嵌入到 TinyOS 中。通过仿真实验，得到了验证。

关键词：传感器网络；节点调度；TinyOS；Tossim

Research and Implementation of Random-sleep Node Dissemination Algorithm for Sensor Network

HE Zhaosun, SHI Gaotao, LIAO Minghong
(School of Computer Science and Technology, Harbin Institute of Technology, Harbin 150001)

【Abstract】 Sensor network is composed of many small, low-power devices with sensing, communication and limited on-board processing capability. The limited power of node and high packet collision with dense node makes noke dissemination necessary. This paper gives a random-sleep node dissemination system based on dynamic neighboring node information, and embedded it into TinyOS as a reusable component. The system is proved through the simulation.

【Key words】 Sensor network; Node dissemination; TinyOS; Tossim

随着传感器技术、嵌入式计算技术和通信技术的日臻成熟，出现了具有感知、计算和通信能力的微型传感器。由微型传感器构成的传感器网络可以使人们在任何时间、地点和环境条件下获取大量详实而可靠的信息，因此被广泛地应用于国防军事、环境监测、交通管理、医疗卫生、反恐抗灾等领域。

在传感器网络中，节点一般都依靠电池供电，其能源非常有限，并且由于节点数量大，节点撒播出去后一般很难回收，因此对这些节点进行充电几乎是不可能的。所以，如何最大限度地节省能源的使用以延长网络寿命是传感器网络的关键问题。

目前从物理层到应用层有很多围绕着能源管理的都已在开展，在这些方法中，真正最大限度的节能方式是关闭节点，使节点睡眠。这种方法是可行并且是合理的，原因是大部分传感器网络都以高密度配置，一方面节点同时处于工作状态，会浪费大量不必要的能源，并且搜集到的数据会高度相关和冗余，另一方面由于同一区域中节点竞争信道也会带来过多的数据包冲突。

因此，目前一种广泛使用的减少能源消耗的办法就是调度节点使其轮流工作，尽可能多地关闭冗余节点，减少了不必要的能源消耗，以达到延长了网络寿命的目的。

虽然目前出现了许多节点调度算法，但这些算法的验证平台绝大多数都是用户自己设计实现的，而没有将算法在传感器网络操作系统 TinyOS 中加以验证，这就使算法只停留在理论上，与实际应用出现了严重的脱节，算法本身性能的好坏也就无法得到客观的评价。因此，将节点调度算法在传感器网络操作系统 TinyOS 中加以实现是一项紧迫的任务。

1 相关工作

目前有很多关于节点调度方面的研究工作在进行，这些工作大致可以分为以下两类：

(1)节点确定性睡眠。文献[2]中每个节点判断邻居是否可以管理该节点的覆盖范围，进而决定是否睡眠。文献[3]给出了一个基于探测的网络密度控制协议 PEAS。文献[4]等人将网络划分成虚拟的网格，在每个网格里面只保留一个工作节点。文献[5]研究如何使用最少节点达到全覆盖。文献[6]和文献[7]研究了传感器网络的 K 覆盖问题。

(2)节点随机睡眠。其基本思想是每个节点以概率 p 睡眠，以 1-p 的概率保持工作状态。这方面的研究集中在节点睡眠概率和节点感知半径、网络区域大小的关系方面。在这些研究中，每个点以固定的概率睡眠，因此适应性不好。

本文的研究属于第(2)种，在分析固定概率缺点的基础上提出了一种更加适应于传感器网络的随机调度模式。本文的创新之处在于，将提出的算法在 TinyOS 上进行仿真，并将此节点调度算法做成 TinyOS 库中的一个组件，可供其它应用程序使用。

2 随机睡眠调度机制

在传感器网络中，节点被撒播在环境中，每个节点随机醒来，然后开始工作。假设每个节点的工作过程分为周期为

基金项目：哈尔滨工业大学校基金资助项目(HIT 2002.74)；国家自然科学基金资助重点项目(60533110)
作者简介：何朝笋(1980－)，男，硕士生，主研方向：无线传感器网络；石高涛，博士生；廖明宏，博士、博导
收稿日期：2006-05-17 **E-mail：**hezs2000@163.com

T 的轮(round)，在每一轮的开始进行节点调度，之后根据调度结果使工作的节点进行感知，其它节点睡眠。

2.1 固定概率调度

固定概率调度是指每个节点以预先设置的相同概率 p=β来进行睡眠，其中β是一个[0,1]之间的实常数。这种调度方式比较简单，基本没有引入任何负载，但是这种调度方式只考虑了节点自身，没有考虑到整个网络的覆盖和连通性，因此这种方法不具有可扩展性和适应性。

2.2 动态邻居信息概率调度

在这种调度方式中，在每一轮开始时，每个节点都向外发送一个短消息，这个短消息中包含了节点编号等节点基本信息，网络中的其它工作节点在收到消息后，将这个节点加入到自己的邻居表中，每个节点统计自己的邻居节点个数，以此来决定自己在本轮是工作还是睡眠。上面提到的邻居节点是指处于节点通信半径之内的节点。

考虑另外一种邻居节点，节点感知半径内的节点。如果一个节点其感知半径内的邻居节点个数 k 满足 k≥3，则这个节点睡眠后，可以认为这个节点的睡眠对网络的覆盖是没有影响的。

如图 1 所示，A 和 B 互为邻居节点，并且位于彼此的感知区域内，根据几何知识，节点 A 和节点 B 的感知圆环所形成的圆心夹角 $\theta = 2\arccos(d/2R)$，因为 $0 < d < R$，所以 $2\pi/3 \le \theta < \pi$，因此，节点 A 的感知区域若被其邻居节点瓜分至少需要 3 个邻居节点。

若考虑感知区域外的邻居节点，则所需的节点数目还会更少。所以，当 k 为 3 时，所有邻居节点个数小于 3 的节点都将处于工作状态，在这种情况下，网络的覆盖率会非常高。

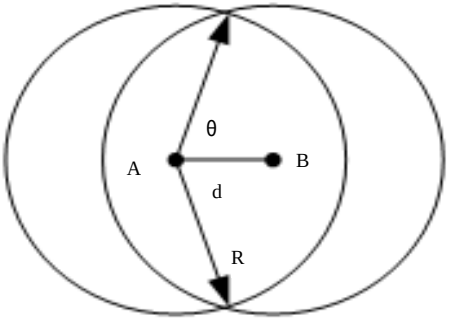

图 1 最少邻居节点个数分析

上面，给出了邻居节点的两种不同定义，这是因为一般节点的感知半径比通信半径小得多，并且节点的通信半径可调。Zhang 和 Wang 已经证明，当通信半径不小于感知半径的 2 倍时，全覆盖的网络是连通的。假定在算法中节点通信半径是感知半径的 2 倍。

至此，给出每一轮开始时每个节点上运行算法的大概过程，如下所示：

```
1  if(节点处于工作状态)
2     then 节点向外发送一个短消息；
3  产生一个定时器；
4  while(定时器没有被触发)
5     节点接收短消息，并且统计收到的消息个数 n；
6  n=n/4；
7  k=3；
8  if(n>=k)
9        then 以概率 p=(n-k)/n 关闭节点；
10 调度结束后，处于工作状态的节点开始正常工作；
```

由于已经假定节点通信半径是感知半径的 2 倍，因此由均匀分布可知，节点收到短消息的个数除以 4 就是节点感知半径内的邻居节点个数。

3 算法在 TinyOS 中具体实现过程

(1)进入 TinyOS 系统的根目录，然后进入 tos/lib 目录中，在下面建立一个名为 RandomSleep 的文件夹。在此目录下，将给出上述动态邻居信息概率随机睡眠机制的具体实现，并将它作为一个组件供应用程序使用。

(2)节点在每一轮开始时，都向外发送一个短消息，为了区别这个短消息和正常的数据信息，定义如下 RandomSleep.h 头文件：

```
enum {
    AM_RANDOMSLEEP_MSG = 249
};
typedef struct RandomSleep_Msg {
    uint8_t inf_type;
    uint8_t node_id;
} RandomSleep_Msg;
```

变量 inf_type 表示消息的类型，上层应用程序封装消息时在此变量中设定特定的值，节点接收消息时，查看此变量的值来看是何种类型的消息，如果是短消息则将 node_id 值加入自己的邻居节点表中，如果是正常的数据则表明此时节点已经处于稳定的工作状态，此时则调用 tos/lib/Route 中的多跳算法来实现数据的转发。

(3)定义了接口 OnOff，如下所示：

```
interface OnOff {
command result_t off();
event result_t requestOff();
        event result_t on();
}
```

接口 OnOff 通过调用系统底层的电源管理组件来实现对节点的开启和关闭。

(4)建立模块文件 RandomSleepM.nc，在这个模块中用到了接口 ReceiveMsg、Send 等，其中 Send 用来将封装好的消息发送出去，ReceiveMsg 用来通过 Radio 接收消息。在事件 ReceiveMsg.receive 中，加入了前面给出的算法，来统计感知半径内的邻居节点个数，在算法中使用了 OnOff 接口来控制节点的开启与关闭。

(5)又编写了模块配置文件 RandomSleep.nc，在此文件中将模块文件中使用到的 Send，ReceiveMsg 接口绑定到多跳路由组件上，以实现多跳路由机制。另外在模块配置文件中，用到了 RandomHPLOnOffM.nc 和 RandomOnOffC.nc 文件，它们都是对系统底层进行操作的模块。

(6)在 app 目录中编写上层应用程序 TestRandom Sleep。应用程序加入了对程序的运行时间的控制，同时使用了 RandomSleep 目录下我们编写的算法组件。

4 仿真实验

为了验证算法的优劣以及算法在 TinyOS 中实现的正确性与否，分别在 TinyOS 提供的仿真平台 Tossim[8]上和自己写的平台上进行了仿真。

Tossim 仿真平台同自己写的仿真平台有着不同之处，前

者在统计邻居节点个数时，只能通过计数接收到的短消息个数来获得邻居节点个数。

后者则不同，它可以通过判断一个节点是否在另一个节点的感知半径之内来决定它们是否为邻居节点。由于存在上面的不同之处，因此在仿真时，算法在统计邻居节点方法上有所不同。

在 Tossim 仿真中，其内部提供了两种 Radio 模型：simple 和 lossy。Simple 模型假设任意两个节点之间都能通信，这显然不符合实际。而 lossy 模型给出了一个定向图，根据图上的拓扑决定节点是否进行通信，并且两个可以通信的节点之间都有一个数据传输错误率，更加接近实际，因此用 lossy 模型来进行仿真。

先在 300*300 区域上随机产生 n 个节点的位置，然后根据上述节点位置信息，用 TinyOS 提供的工具 LossyBuilder 在 300*300 区域上产生节点间的数据传输错误率，并设定每个节点的通信半径为 50。分别以节点个数 n 等于 100，150，200，250，300 运行程序 10 次后，得到的结果如图 2 所示。

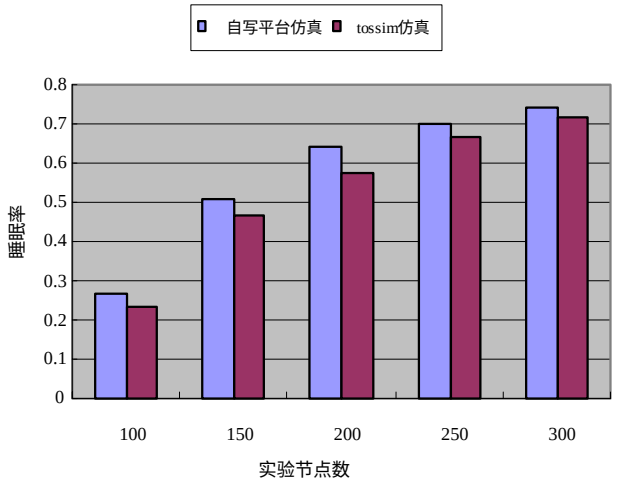

图 2 两个仿真平台下的睡眠节点数对比

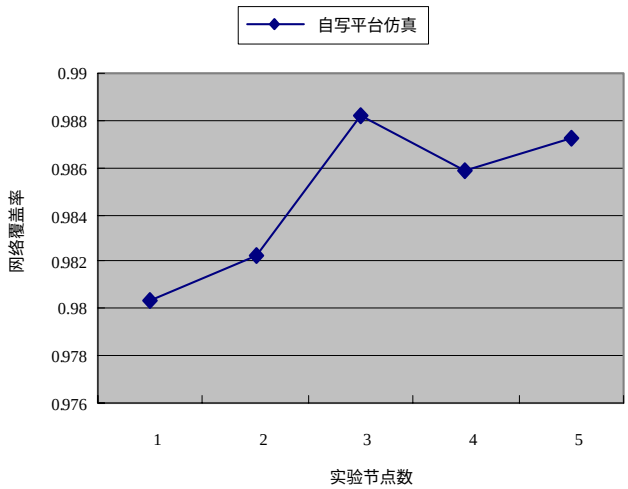

图 3 自写平台下网络的覆盖率

在用自己写的平台仿真时，将节点放置在 300*300 区域上，节点的通信半径为 50，感知半径为 25，同样分别以节点个数 n 等于 100，150，200，250，300 运行程序 10 次后，实验结果如图 2 所示。

由图 2 可以看出，两个实验平台下的节点睡眠率相差不大，这充分说明了我们在 TinyOS 下实现算法的过程是基本正确的，本文在前面算法中做出的假设也是正确的。

图 3 是自写平台下运行算法后网络的覆盖率，从图中可以看出，运行算法后，网络的覆盖率达到 98%以上，这充分说明了本文的算法保持了较高的网络的覆盖率。

目前我们还不能在 Tossim 仿真中获得网络覆盖率的分析，因为 Tossim 仿真和自写平台的仿真具有不同之处，在 Tossim 仿真中无法计算网络的覆盖面积，所以无法计算网络的覆盖率。

5 结论及后期工作

本文主要分析了传感器网络节点调度算法研究的现状，在分析固定概率调度算法存在的问题的基础上提出了一种随机睡眠调度算法，并且将算法做成了一个可供重复使用的组件嵌入到 TinyOS 中。通过仿真实验，认为该算法最大限度地减少了工作节点，保持网络有较高的覆盖率，同时延长了网络寿命。

本文对算法的验证是在仿真环境中进行的，一般在仿真环境下，许多实验条件都是比较理想化的，与实际应用中还是有一定的差距的。因此，我们后期的工作将主要集中在将算法在实际的节点上进行实验，以进一步分析算法的性能并加以改进。

参考文献

1 Hill J, Szewcyk R, Woo A, et al. System Architecture Directions for Networked Sensors[C]//Proceedings of the International Conference on Architectural Support for Programming Languages and Operating Systems. 2000.

2 Akyildiz I F, Su W, Sankarasubramaniam Y, et al. A Survey on Sensor[J]. IEEE Communication Magazine, 2002, 44(8): 102-114.

3 Ye F, Zhong G, Lu S, et al. PEAS: A Robust Energy Conserving Protocol for Long-lived Sensor Networks[C]//Proc. of the 23rd International Conference on Distributed Computing Systems. 2003: 28-37.

4 Xu Y, Heidemann J. Geography-informed Energy Conservation for Ad-hoc Routing[C]//Proc. of the 7th Annual ACM Conference on Mobile Computing and Networking. 2001: 16-21.

5 Zhang Honghai, Hou J C. Maintaining Sensing Coverage and Connectivity in Large Sensor Networks[R]. University of Illinois at Urbana Champaign, 2005.

6 Wang X, Xing G, Zhang Y, et al. Integrated Coverage and Connectivity Configuration in Wireless Sensor Networks[C]//Proc. of the First ACM Conference on Embedded Networked Sensor Systems. 2003: 28-39.

7 Huang C F, Tseng Y C. The Coverage Problem in a Wireless Sensor Network[C]//Proc. of the 2nd ACM International Conference on Wireless Sensor Networks. 2003: 115-121.

8 Levis P, Lee N. TOSSIM: A Simulator for TinyOS Networks[Z]. 2003. http://www.cs.berkeley.edu/~pal/pubs/nido.pdf.