Communications and Network, 2009, 1, 108-113
doi:10.4236/cn.2009.12017 Published Online November 2009 (http://www.scirp.org/journal/cn).

An Optimal Algorithm for Over-the-Air Reprogramming Protocol in WSNs

Ting WEN, Zhi LI*
School of Electronics and Information, Sichuan University, Chengdu, China
**Corresponding author*
E-mail: {*sunny.wenting, dr.elitelee*}*@gmail.com*
Received September 20, 2009; *accepted October* 21, 2009

Abstract

The over-the-air reprogramming has played an important role in Wireless Sensor Networks (WSNs), and has been widely researched. But the problems of low efficiency and high energy consumption have brought new challenges to the research. This paper presents an optimal algorithm, it is targeted to address the problem of relatively low efficiency and high energy consumption in over-the-air reprogramming in WSNs. Simulation results demonstrate that the optimal algorithm can improve the efficiency of over-the-air reprogramming in WSNs, and make the energy consumption in network not only lower but more balanced.

Keywords: Over-the-Air Reprogramming, WSNs, Optimal Algorithm

1. 引 言

无线传感器网络(Wireless Sensor Networks. WSNs)是由部署在监测区域内大量的低成本微型传感器节点组成，并通过无线通信方式形成的一个多跳的，自组织网络系统[1]。WSNs 架构灵活，能高分辨率遥感数据并具有自适应机制，因此在国防、环境监测，家庭自动化，医疗，灾区或战场信息的收集和精准农业[2]等很多领域具有广阔的应用前景和极高的应用价值。

在实际应用中，传感器节点一旦部署在指定区域后，将会长时间的在无人职守的情况下工作。而我们知道，随着时间的推移，环境和用户需求会发生变化，这就可能导致传感器节点中的已有应用程序无法完成当前的任务。因此，对 WSNs 进行再编程（即：当传感器节点部署在指定区域后对其进行应用程序的更新）是十分必要的[3]。

目前，WSNs 空中再编程（over-the-air reprogramming in WSNs）的研究主要涉及两个方面[4]：

- 代码分发协议（Code Distribution Protocol），即解决代码在网络中如何繁殖的问题。
- 可靠的通信协议（Reliable Transport Mechanism），即在通信过程中，如何保证所有节点都能无差错地接收代码的问题。

现有的WSNs空中再编程方法主要有：XNP[5]，Deluge[6]，MOAP[4]，MNP[7]等。但是在代码分发协议上，目前性能出色的MNP仅考虑了节点请求者的数量，且每次只能从相邻节点中选出一个作为转发节点。因此对整个网络而言，现在的WSNs再编程方法转发轮数多，花费时间长（如在室外7×7的网络拓扑中，用最大功率传输33KB代码，MNP需要25分钟），节点能量消耗大，效率低。

针对上述问题，本文从 WSNs 空中再编程的代码分发协议上考虑，提出了一种优化算法。该算法通过相邻节点间的距离来确定转发节点，并动态调节转发节点的发射功率，实现了网络中多个相邻节点以不同功率的同时转发，减少了转发轮数，加快了空中代码分发的速度,节点能耗降低，效率明显提高。

2. 优化算法描述

本文所讨论的采用优化算法的 WSNs 网络模型是基于以下假设：

- 节点数目众多且分布均匀。
- 节点全部采用全向天线，且发射功率可调。
- 在 WSNs 空中再编程过程中，网络通信质量良好，拓扑结构不变。

同时，对节点作如下定义：

- 转发节点：通过代码分发协议选出的向下一跳发送代码镜像的节点。
- 候选转发节点：指成功接收了代码段镜像的节点。此时它只具有成为转发节点的可能性。只有在选取机制中获胜的候选转发节点才能成为转发节点。
- 邻居候选转发节点：若两个以最大功率通信的候选转发节点间出现通信重叠区域，则它们为邻居候选转发节点。
- 未知节点：未接收相应代码段镜像的节点。
- 公共节点：位于候选转发节点通信重叠区域内的未知节点。

2.1. 转发节点生成算法

在网络中，首先由基站节点向网络中广播程序代码的镜像。在基站节点的通信范围内，成功接收了代码段镜像的节点便成为了候选转发节点。

候选转发节点要升级成为转发节点，需要经过以下过程：

1）候选转发节点在随机时间段 k 内，以最大功率 P_m 不间断地向网络广播一定数量的“ Discovery ”消息，此消息的覆盖半径设为 R_m 且包括以下重要信息：

P_ID：网络节点将要更新的应用程序版本；

S_ID：代码段镜像的标号；

SID：发送此“ Discovery ”消息的节点网络 ID；

T_Req：请求节点的总数。T_Req 的值初始为零，若在时间段 k 内，T_Req=0，说明此节点周围的节点已经全部接收到了相应的代码段镜像。

2）未知节点接收到“ Discovery ”消息后，根据 P_ID 和 S_ID 来决定是否接收新的代码镜像。若不接收，则保持“ 沉默 ”；若要接收，则会向网络广播发送“ Request ”消息，此消息包括以下重要信息：

SID：发送“ Discovery ”消息的候选转发节点 ID；

O_ID：发出此“ Request ”消息的节点网络 ID；

P0：接收功率；

T_Req：与“ Discovery ”消息中的 T_Req 为同一值。

3）候选转发节点根据“ Request ”消息中的 SID 来判断是否为公共节点并记录其数量 Np（初始值为零）和请求节点的总数 T_Req。

4）若此时有公共节点，候选转发节点则根据无线传输模型中，发送数据所消耗的能量 $E(d)$ 与距离 d 的关系[8]及功率和能量的关系，由“ Request ”消息中的 P_0，计算出与公共节点间的距离，并记录其最小值 d_{min} 和对应的公共节点网络 ID，通过下式：

$$d' = d_{min} + R_m \tag{1}$$

5）随后，候选转发节点通过“ Distance ”消息把自己的网络 ID，d' 和与 d' 所对应的邻居候选转发节点的网络 ID 通知给其通信区域内的公共节点，并让公共节点广播此消息。

此后，每个候选转发节点把接收到的 d' 和自己计算的此距离求平均后得到 d，作为自己与某个邻居候选转发节点的最终距离，记录为 d。

例如图 1 所示，A,B 为邻居候选转发节点，C,D 分别是距离 A,B 最近的公共节点，则对 A 有：

$$d_{min} = d_{AC} \tag{2}$$

$$d'_A = d_{min} + R_m \tag{3}$$

同理可得 d'_B，因此求得 A，B 两节点间的距离 d：

$$d_{AB} = (d'_A + d'_B)/2 \tag{4}$$

6）候选转发节点（如节点 A）对 Np 的值进行判断：

a) 若 Np 为零，表明 A 以最大功率 P_m 进行通信时，没有邻居候选转发节点，此时，A 将升级为转发节点，其转发功率为 P_m，覆盖半径为 R_m。

b) 若 Np 不为零，即是说在 A 附近有邻居候选转发节点。通过记录的距离 d（$d < 2R_m$），建立一个关于邻居候选转发节点信息的动态向量表，例如有 A,B,C 三个候选转发节点，且 A 与 B,C 都有通信重叠区域，则 A 所记录的动态向量表由表 1 所示：

表 1. 节点 A 的动态向量表示意.

Table 1. The dynamic vector set of node A.

候选转发节点地址	距离
A,B	d_{AB}
A,C	d_{AC}

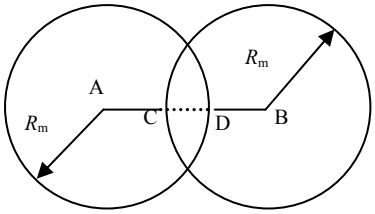

图 1. 邻居候选转发节点距离计算示例.

Figure 1. Calculating *d* between candidate forwarding nodes.

若候选转发节点的动态向量表中某一距离值小于 d_6，表明邻居候选转发节点间的通信重叠区域较大。此时，让能量较低的节点在此轮中退出竞争并在代码镜像转发开始后进入休眠态以节省能量，而让另一个节点升级成转发节点。

实验中我们发现，当两个候选节点同时转发代码时，它们之间的距离越近（公共节点越多），网络再编程的效率就越低，当距离小于 d_6 时，再编程效率非常明显地降低。因此，从再编程效率的角度出发，我们在距离小于 d_6 的节点间采取低能量的节点在本轮转发中休眠的策略，拉开了转发节点间的距离。

7）随后，网络中的转发节点重复以上步骤，更新动态向量表，使记录的所有的距离值 $2R_m > d \geq d_0$。

2.2. 转发节点调整发射功率

通过上述算法后，生成的转发节点若使用原来的发射功率，那么处于通信重叠区域中的未知节点由于同时有多个节点要向自己发送数据，最终将导致接收代码镜像不成功。为了解决这以问题，此时我们采取调整转发节点发射功率的办法，具体步骤为：

1）转发节点在自己的动态向量表中，找出最短距离 d_m，把发射功率调整到 P_1（P_1 的覆盖半径是 $d_m/2$，在实际中，选择与 P_1 最相近的功率），然后在网络中广播“Power”消息，通知相邻的转发节点，自己将以功率 P_1 进行代码镜像的转发。

2）若转发节点随后接到了其他节点发来的“Power”消息，它将收到的最小发射功率与自己计算出来的 P_1 做比较，选出最小值 P_{min} 做为其发射功率，若未接到“Power”消息，节点则将采用 P_1 功率射。

此后，在同一轮中生成的转发节点，便可以同时以不同的功率进行代码镜像的转发，而“被迫”休眠的节点，则会在下一时段重新参与转发节点的竞选；从节能的角度考虑，转发过代码镜像的节点要休眠一段时间，不会连续两次参与转发节点的竞争。

为了便于理解，特举例说明优化算法的过程。如图 2 所示。

已知 A,B,C,D,E,F 为 6 个候选转发节点，分别以这 6 个节点为圆心的圆为等圆，表示候选转发节点以最大功率通信时的覆盖区域，且半径都为 R_m。节点 A-B，A-C，B-C，C-D，D-E,A-F 间的距离分别为 d_1，d_2，d_3，d_4，d_5，d_6 且 $d_6 < d_0 < d_4 < d_2 < d_1 < d_3 < d_5$。

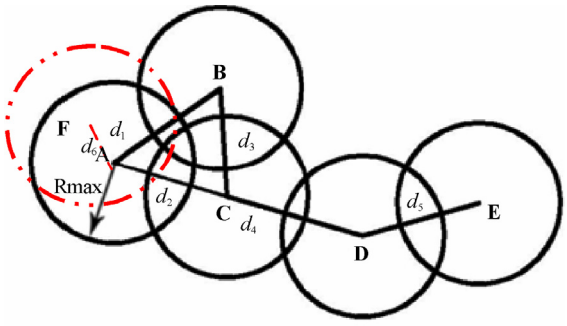

图 2. 优化算法的过程示例

Figure 2. Process of the optimal algorithm.

由图 2 知，圆与圆之间的公共部分就是重叠区域，由优化算法知，由于只有距离 d_6 小于 d_0，所以此时 A 和 F 中能量较低的节点（若为 F）进入休眠态，而节点 A 就升级为转发节点。

随后，A,B,C,D,E 这 5 个转发节点根据动态向量表的记录，调整各自的发射功率，使 5 个节点能同时转发代码镜像，避免对各自区域中的公共节点造成影响。

3. 仿真及分析

采用 TinyOS[9]的仿真平台 TOSSIM[10]对以上优化算法进行仿真对比实验。转发的代码镜像大小约为 11KB(分成 4 段,每段包含 128 个数据包)，节点的初始功率均为 P_m。在仅有代码分发协议不同，其他协议不变的条件下，将本文所述的优化算法与 MOAP 和 MNP 机制的代码分发协议 Ripple[4]和 Sender Selection Protocol[7]作比较。

3.1. 再编程的效率分析

1）令网络节点拓扑为 10×10 的网格，通过改变节点间距，分别记录采用三种协议时，网络节点全部成功接收代码所用的时间，仿真结果如图 3 所示。

由图 3 的仿真结果可看出，随着节点间距的增大，采用优化算法的网络再编程所需时间变换地最慢，当间距为 80ft 时，其再编程效率比使用 Sender Selection Protocol 时提高了约 32%。但我们注意到，当节点间距较小时，采用优化算法后的时间比采用 Sender Selection Protocol 协议时要长。这主要是因为节点间距较小时，转发节点生成算法使较多的节点“休眠”，空中再编程效率反而不如 Sender Selection Protocol,但随着节点间距的逐渐增大，优化算法通过调节

转发节点发射功率的办法使多个节点同时转发代码镜像，再编程的效率明显提高。另一方面，由于 Ripple 协议不支持代码段转发机制，而只能在节点全部接收代码后才能进行转发，因此，空中再编程的效率始终不如优化算法。

2）节点拓扑结构不变并固定节点间距为80ft。在不同时刻t，记录网络中成功接收代码的节点数占总节点数的比例，仿真结果如图4示。

从图 4 的仿真结果中我们看到，在任意时刻，较另外两种协议而言，采用优化算法时的成功节点比例都明显较高。同时可看到，在时间的前半段，采用优化算法时成功节点的增长比例很快，但随着时间的推移，其增长比例又逐渐减小。这主要是因为刚开始时，网络中未知节点比较多，采用优化算法时，不仅同一时刻网络中的转发节点较多，而且成功接收代码的节点数量也很多,因此较Sender Selection Protocol和Ripple协议而言，成功节点率的增加比例较大；而到了时间的后半段，增长比例的下降是由于网络中未知节点数已很少了的缘故。

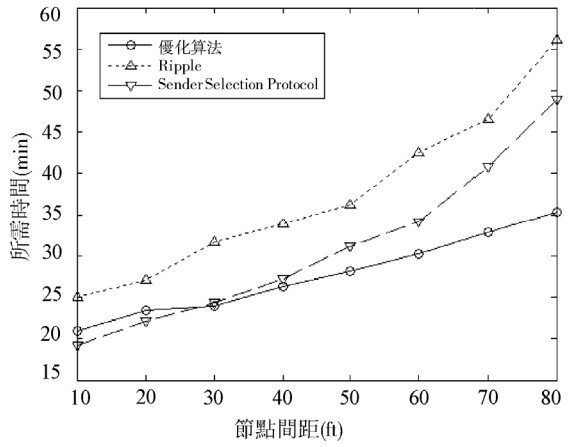

图 3. 再编程所需时间对比.

Figure 3. The contrast in completion time.

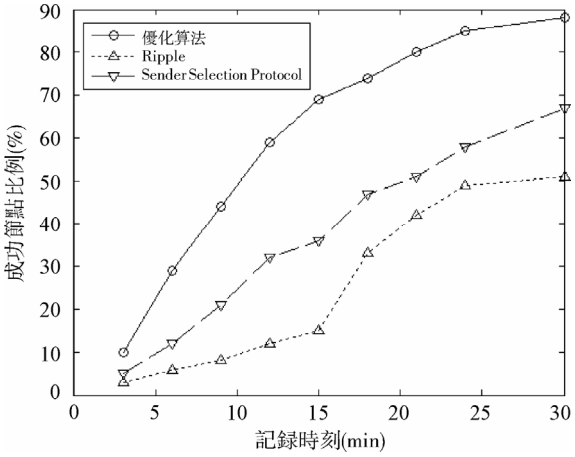

图 4. 再编程的成功节点率对比.

Figure 4. The contrast in proportion of "successful" nodes.

3.2. 再编程的能耗分析

传感器节点的绝大部分能量消耗在无线通信模块，尤其是在发送和接收态[11,12]。因此，节点收发包的总个数实际上与节点的能量消耗成正比。

在仿真中，令节点间距为20ft。通过把三种代码分发协议应用于不同的拓扑分布(实验中全部采用正方形拓扑，如7×7表示49个节点组成的正方形)，统计平均每个节点的收发包总数。实验结果如图5示。

由图 5 的仿真结果可知，节点数目越多，每个节点平均的收发包个数也越多。但是，当节点数目较少时，较其他两种协议而言，采用优化算法后，平均每个节点收发包个数较多。这主要是因为建立动态向量表时的通信在整个通信过程中所占的比例相对较高。

当节点增加到一定数量的时候（仿真中测得大约为 12×12 的网络拓扑，根据实验环境的差异，此值可能不同），采用优化算法后，每个节点平均收发包的数量开始低于使用另外两种协议时的收发包总个数。这是因为节点数目较多后，网络中同时转发的节点数量要明显高于使用其他两种协议时。转发节点数量的增多，使得同一时刻网络的再编程覆盖面增大了，提高了再编程的速度，所以在整个过程中，平摊到每个节点上的收发包总数就较少了。因此，从能耗的角度上说，优化算法比 Ripple 和 Sender Selection Protocol 更能节省网络能量。

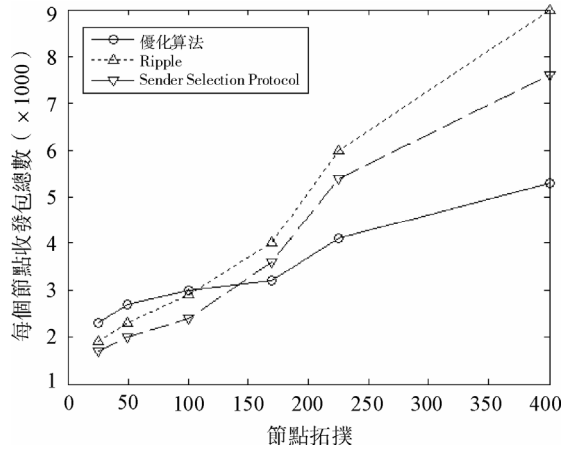

图 5. 再编程的总体能耗对比

Figure 5. The contrast in total energy consumption.

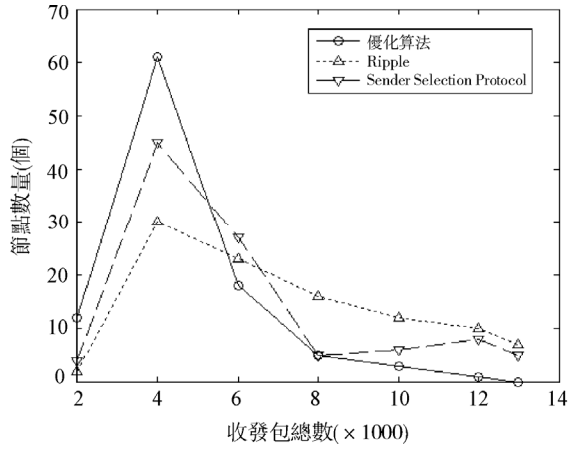

图 6.再编程节点能耗分布(10 × 10 拓扑分布).
Figure 6. The contrast in node energy consumption (10×10grid).

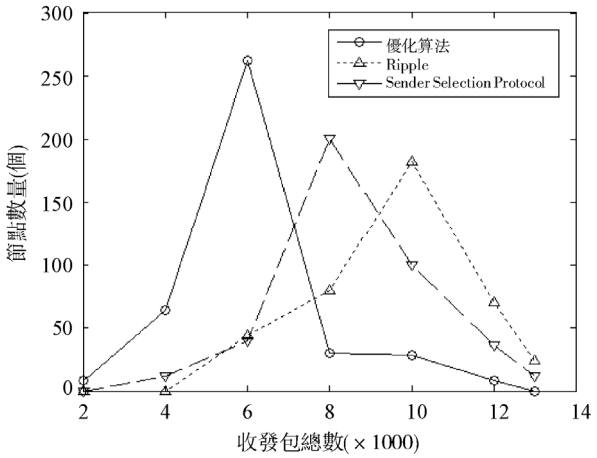

图 7.再编程节点能耗分布(20 × 20 拓扑分布).
Figure 7. The contrast in node energy consumption (20×20grid).

此外，令节点拓扑结构分别为 10×10 和 20×20 的网格并固定节点间距为 20ft。当网络再编程结束时，将节点按各自收发包的数量所在范围进行统计，实验结果如图 6，7 所示。

由图 6 和图 7 可以看出，当采用优化算法时，网络中超过一半的节点的收发包总数集中在一个区间(当网络为 10×10 和 20×20 的拓扑时，包总数分别集中在 2000 到 4000 间和 4000 到 6000 间)，当节点较多时，这种“集中”现象更加明显，且只有个别节点的收发包数较大；而采用 Sender Selection Protocol 和 Ripple 时，“集中”现象并不明显，且都存在较多的节点收发包总数很多的情况。因此，从能量的角度来说，使用优化算法，网络中节点的能量消耗将更加均衡。

4. 结束语

本文综合考虑了 WSNs 节点能量有限且分布范围广的特点，针对 WSNs 空中再编程技术，提出了一种优化算法。该算法通过同时选取多个相邻转发节点并动态调节转发节点发射功率的办法，优化了代码的转发路径. 仿真结果表明, 与传统的代码分发协议相比，此优化算法不仅明显地提高了网络再编程的效率，而且在降低网络能耗的同时，均衡了网络能耗。当节点数量较多,节点间距较大时，其优化效果更加明显。下一步工作的重点是：进一步完善优化算法中转发节点的生成算法，使转发节点具有更高的网络覆盖率。同时针对实际 WSNs 的通信环境和拓扑结构，研究可靠的传输机制，进一步提高 WSNs 空中再编程的效率。

6. References

[1] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, “Wireless sensor networks: A survey,” Computer Networks Journal, Vol. 38, No. 4, pp. 393–422, 2002.

[2] J. Gehrke and L. Liu, “Sensor-network applications,” IEEE Internet Computing, Vol. 10, No. 2, 2006.

[3] Q. Wang, Y. Y. Zhu, and L. Cheng, “Reprogramming wireless sensor networks: Challenges and approaches,” IEEE Network Magazine, May–June 2006.

[4] T. Stathopoulos, J. Heidemann, and D. Estrin, “A remote code update mechanism for wireless sensor networks,” Technology Report, CENS-TR-30, University of California, L.A., 2003.

[5] Crossbow Technology, Inc., Mote In-Network Programming User Reference Version 20030315, 2003, http://webs.cs.berkeley.edu/tos/tinyos-1.x/doc/Xnp.pdf.

[6] J. W. Hui and D. Culler, “The dynamic behavior of a data dissemination protocol for network programming at scale,” In Proceedings of the second International Conference on Embedded Networked Sensor Systems (SenSys 2004), Baltimore, Maryland, 2004.

[7] S. S. Kulkarni and L. M. Wang, “MNP: Multihop network reprogramming service for sensor networks,” Proceedings of the 25th IEEE International Conference on Distributed Computing Systems (ICSCS’05), 2005.

[8] J. P. Pan, “Topology control for wireless sensor networks [C],” Proceedings of the 9th Annual International Conference on Mobile Computing and Networking, San Diego, California, ACM Press, USA, pp. 286–299, 2003.

[9] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. E. Culler, and K. S. J. Pister, “System architecture directions for networked sensors,” In Proceedings of the 9th International Conference on Architectural Support for Programming

Languages and Operating Systems, Boston, MA, USA, pp. 93–104, November 2000.

[10] P. Levis, N. Lee, M. Welsh, and D. Culler, "TOSSIM: Accurate and scalable simulation of entire TinyOS applications," In Proceedings of the First ACM Conference on Embedded Networked Sensor Systems (SenSys), November 2003.

[11] J. Hill and D. Culler, "Mica: A wirleess platform for deeply embedded networks," IEEE Micro, Vol. 22, No. 6, pp. 12–24, 2002.

[1] L. M. Sun and J. Z. Li, "Wireless sensor network," Published by Tsinghua University, 2005.

