无线传感器网络节点周期性休眠时间同步研究*

孙新伟,申兴发,张能贵
(杭州电子科技大学 计算机学院,浙江 杭州 310018)

摘要:为了实现无线传感器网络节点的有效时间同步,采用了 OMNeT ++ 仿真工具,并改进了 TPSN 算法,提出了一种无线传感器网络节点周期性休眠时间同步算法。算法分为发现及收集子节点信息和节点同步两个阶段,实现了对无线传感器网络节点的实时精确时间同步。仿真结果表明,该时间同步算法能实现对网络节点十分准确的同步。

关键词:无线传感器网络;时间同步;周期性休眠
中图分类号:TP393.01 **文献标识码**:A **文章编号**:1001 - 4551(2010)05 - 0075 - 03

Time synchronization study of periodic sleep for wireless sensor networks

SUN Xin-wei, SHEN Xing-fa, ZHANG Neng-gui
(Institute of Computer Application, Hangzhou Dianzi University, Hangzhou 310018, China)

Abstract: Aiming at achieving the effective time synchronization in wireless sensor network node, OMNeT ++ simulation tools were used, and TPSN algorithm was improved, a wireless sensor network nodes periodically sleep time synchronization algorithm was divided into discovery and collection of child nodes of information and nodes synchronize. Wireless sensor network nodes were realized in real-time precision time synchronization. The simulation results show that the time synchronization algorithm can achieve very accurate synchronization of network nodes.

Key words: wireless sensor network; time synchronization; periodic sleep

0 引 言

时间同步是任何分布式系统的一个重要基础,也是无线传感器网络的一项基础支撑技术。在无线传感器网络的应用中,传感器节点采集的数据如果没有空间和时间信息是无任何意义的。准确的时间同步是实现传感器网络自身协议的运行、数据融合、TDMA 调度、协同睡眠、定位等的基础[1-6]。近年来,经过各国学者的不懈努力,至今已经提出了许多时间同步算法[7-9]。纵观这些算法主要可以分为以下 3 类:基于接收者 - 接收者(receiver-receiver)的同步算法、基于成对(pair-wise)同步的双向同步算法和基于发送者 - 接收者(sender - receiver)的单向同步算法。TPSN 是基于双向成对同步的,采用在 MAC 层进行时间标记的方法,减少了关键路径的长度,因而减小了传输延迟的不确定性,达到了较高的同步精度。但该算法每个下层节点都要与它的上层节点进行双向成对,消息开销较大。在无线秒表的实际应用中,无线传感器网络具有以下特性:节点用电池供电,并且需要长期工作,所以采用周期性休眠工作方式;要求节点唤醒的同步性;硬件资源有限;通信能力弱,需要通过多跳传播消息;节点位置固定不变,属于静态网络。由于节点休眠状态时钟精度相对较低和漂移很大,造成各个节点之间的时间不同步,偏差比较大,所以网络节点唤醒并不同步,容易造成多余的等待和能量浪费。而且由于唤醒的不同步,网络中存在着休眠和工作两种状态节点,如何实现全网的再同步也是迫切需要解决的问题。

本研究结合双向成对机制和泛洪广播时间同步的思想,对 TPSN 算法进行改进,设计一种无线传感器网络节点周期性休眠时间同步算法。

收稿日期:2009 - 10 - 10
基金项目:浙江省自然科学基金资助项目(Y107701)
作者简介:孙新伟(1981 -),男,安徽太和人,主要从事计算机软件与理论方面的研究. E-mail: sxwei021@sohu.com

1 时间同步算法设计

双向成对同步机制采用简单的双向往返的方法测量成对节点间的时间偏移和传播延迟,而泛洪广播时间同步考虑到在特定时间范围内节点时钟晶振频率是稳定的,对时钟漂移进行了线性回归分析,两种机制各有所长。因此该算法(长时间休眠时间同步算法 LSTS)采用的是这两种机制的结合。

算法分为两个阶段:

(1) 发现及子节点收集阶段。该阶段主要是通过泛洪广播的方法给网络中每个节点赋予一个层次号,并使每个节点通过信息的收集获得它的下层子节点号。首先选取根节点(一般选 Sink 节点)并赋予层次号 0,然后由它开始广播分层消息,每个接收到该分层消息的节点取出消息包中的层次号,并判断自己的层次号,如果还没有层次号便将自己的层次号设为这个层次号加 1;如果已有层次号,且层次号比消息包中的层次号小 1,便将这个节点号加入到自己的下层邻节点表中。重复这个过程直至所有节点都赋予一个层次号,至此整个网络形成了一个以第 0 层节点为根的层次数,且每层非叶子节点都有了自己广播域内下层节点号的信息。

(2) 同步阶段。该阶段的主要任务是从根节点开始逐层同步网络中的节点,使它们达到整体同步。在该阶段,由根节点的 time_sync 包发起,当接收到这个包,第 1 层的节点发起与根节点的双向消息交换。在发起消息交换之前,为最小化无线信道冲突,每个节点都要等待一个随机时间。一旦接收到根节点的回复消息,它们用双向成对同步的方法计算它们之间的偏移和传播延迟,并调整自身时钟到根节点的时钟,计算式如下:

$$\delta = \frac{(T_2 - T_1) - (T_4 - T_3)}{2};$$

$$d = \frac{(T_2 - T_1) + (T_4 - T_3)}{2} \tag{1}$$

式中 T_1—子结点的发送时间;T_2—根结点的接收时间;T_3—根结点的发送时间;T_4—子结点的接收时间。

由于节点唤醒的不同步性,为了保证第一层的所有节点都能够实现与根节点的同步,在根节点定义了一个数组向量,用来标记该同步周期与根节点完成同步的第一层节点。若发现其子节点未实现同步,则周期性地重发同步消息。算法消息交换如图 1 所示。

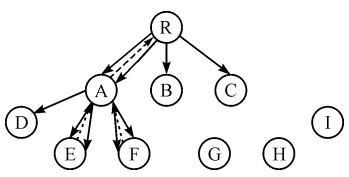

图 1 算法消息交换示意图

根节点 R 在同步周期广播同步消息,唤醒状态的子节点 A 实现与根节点同步,未唤醒的节点 B 与 C 则未完成同步,根节点 R 周期性广播同步消息,实现子节点 B、C 与根节点的同步。第 1 层节点与根节点同步后,各自选取自己广播域表中的子节点,重复以上同步过程,直至网络中所有节点都达到同步。算法在 MAC 层为发送节点和接收节点标记时间戳,排除了发送时间、访问时间和接收时间的影响,提高了算法的精度。

对于网络中新加入的节点,可以通过向周围节点发送广播的方式得到一个层次号,具体方法是新加入节点发送广播请求消息,所有接收到该请求消息的节点都返回一个它所在的层次号,新加入节点在收到应答消息后,比较它们的层次号,选层次号最小的加 1 作为自己的层次号,并把层次号最小的节点作为自己的父节点,并告诉父节点已成为其子节点。

2 算法性能仿真

为了检验和比较 LSTS 算法的性能,采用 OMNeT ++ 仿真工具对算法进行仿真实验。OMNeT ++ 是一种基于组件的模块化的开放的网络仿真平台,作为离散事件仿真器,具备强大完善的图形界面接口。

2.1 OMNeT ++ 建立仿真模型

仿真环境中部署 7 个节点,建立 3 层网状结构,每个节点分别对应一个确定的层次号,并且能够接收对应父子节点的消息。假设整个网络的工作周期为 T_{circle},固定为 10 s,其中节点休眠时间为 T_{sleep},T_{sleep}在 0 ~ 10 s 之间取值;第 i 节点相对根节点的时钟偏移为 $Offset[i]$,$Offset[i]$为 0 到最大偏移设定值 T_{offset}之间产生的随机数;第 i 节点的丢包概率为 $P[i]$,$P[i]$为 0 ~ P_{lost}(最大丢包设定的概率值)之间产生的随机数;父节点同步消息重发的周期设定为 1 s,整个网络完成同步的消息数为 M_{sgcount}。

2.2 LSTS 算法性能分析

仿真模型通过对节点的偏移值、丢包概率和网络休眠时间 3 个因素的改变,统计完成全网同步需要发送的消息数。通过实验环境测试了这 3 个因素对算法性能的影响力。具体结果如图 2 ~ 图 5 所示。

在周期 10 s、最大偏移值为 10 s、丢包概率在 0.2 ~0.4 之间随机产生的情况下网络休眠时间同消息数量的关系如图 2 所示,说明了休眠时间占同步周期的比重越大,全网完成时间同步需要越多的消息开销。

在周期 10 s、网络休眠时间 5 s、丢包概率在 0.2 ~ 0.4 之间随机产生的情况下节点最大偏移值同消息数量的关系如图 3 所示,说明了节点之间偏移值与消息开销的关系,当偏移值接近周期值时,消息开销反而减少。在周期 10 s、最大偏移值为 5 s、网络休眠时间 5 s 的情况下节点丢包概率同消息数量的关系如图 4 所示,说明了丢包概率小于一定值时,消息开销变化很小,当超过该值时,消息开销明显增加。与 TPSN 相比,LSTS 算法减小了消息开销,尤其是节点偏移越大时越明显,在 LSTS 算法增加了对节点下一周期的偏移补偿后,降低了系统能量开销,可以获得更高的同步精度。在假设消息传输时延 10 ms、供电电压 3 V 的条件下,对应节点偏移与系统能耗的关系如图 5 所示。

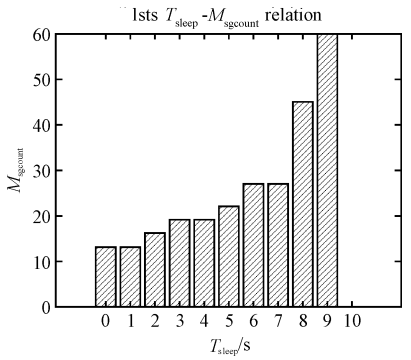

图 2　消息开销随休眠时间的增加的变化图

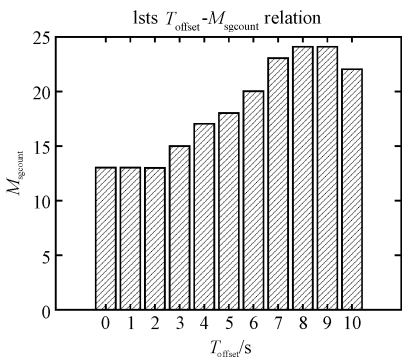

图 3　消息开销随节点偏移的增加的变化

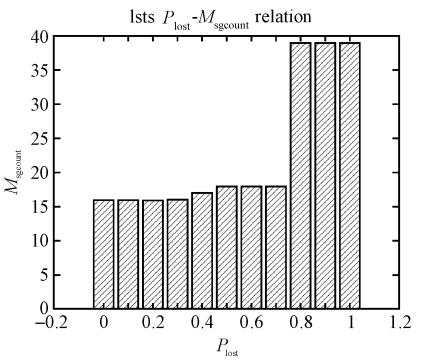

图 4　消息开销随丢包概率的增加的变化

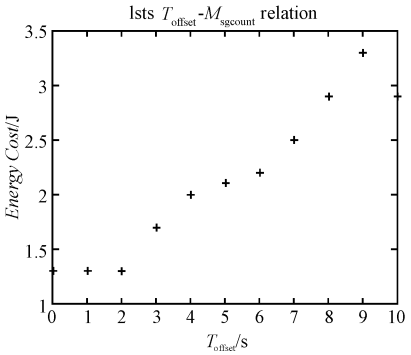

图 5　系统能耗随节点偏移的增加的变化

3　结束语

时间同步作为无线传感器网络应用的一项支撑技术,是无线传感器网络设计和应用的关键。而在具体实际应用中,网络长期处于休眠状态,如何实现全网的时间同步是一个重要的问题。针对这种情况本研究提出了一种将双向成对同步机制和泛洪广播时间同步机制相结合,并作了相应改进的时间同步算法 LSTS。与 TPSN 相比,该算法在增加了一定精度的情况下,满足了无线传感网节点周期性休眠时间同步的要求。仿真结果验证了该算法的可行性。

参考文献(References):

[1] ELSON J, GIROD L, ESTRIN D. Fine-grained Network Time Synchronization using Reference Broadcasts[C]//Proc of the Fifth Symp on Operating systems Design and Implementation(OSDI2002). New York: ACM Press, 2002: 147 – 163.

[2] GANERIWAL S, KUMAR R, SRIVASTAVA M B. Timing-Sync Protocol for Sensor Networks[M]. New York: ACM Press, 2003: 138 – 149.

[3] GANERIWAL S, KUMAR R, ADLAKHA S, et al. Network-wide Time Synchronization in Sensor Networks[EB/OL]. [2003 – 01 – 01]. http://www.ee.ucla.edu/ ~ram.

[4] ELSON J, ROMER K. Wireless Sensor Networks: A New Regime for Time Synchronization[C]. The First Workshop on Hot Topics in Networks(HotNets-I). New Jersey, USA: [s.n.], 2002.

[5] GREUNEN V, RABAEY J. Fine-Grained Network Time Synchronization using Reference Broadcast[C]//Proceeding 5th Symposium on Operating System Design and Implementation, Boston, MA: [s.n.], 2002: 147 – 163.

[6] ROMER K. Time Synchronization in Ad Hoc Networks [C]//ACM MobiHoc'01. Long Beach, CA: [s.n.], 2001: 173 – 182.

[7] SIVRIKAYA F, YENER B. Time synchronization in sensor networks: a survey[J]. **IEEE Networks**, 2004, 18(4): 45 – 50.

[8] SUNDARARAMAN B, BUY U, KSHEMKALYANI A D. Clock synchronization for wireless sensor networks: a survey [J]. **Ad-Hoc Networks**, 2005, 3(3): 281 – 323.

[9] KANG Guan-lin, WANG Fu-bao. DUAN Wei-jun. Survey on time synchronization for wireless sensor networks[J]. **Computer Automated Measurement & Control**, 2005, 13(10): 1021 – 1023.

[**编辑**:李　辉]