WSN 中虚拟时戳与父亲责任链时钟同步算法

赵有俊[1],曾子维[1],臧殿红[2]
ZHAO You-jun[1],ZENG Zi-wei[1],ZANG Dian-hong[2]

1.辽宁科技大学 计算机科学与工程学院,辽宁 鞍山 114051
2.中国石油大学 信控学院,山东 东营 257061
1.College of Computer Science & Engineering,Liaoning University of Science and Technology,Anshan,Liaoning 114051,China
2.College of Information and Control Engineering,China University of Petroleum,Dongying,Shandong 257061,China
E-mail:sdzhyj@126.com

ZHAO You-jun,ZENG Zi-wei,ZANG Dian-hong.Virtual timestamp and father-burden-chain clock synchronization algorithms for WSN.Computer Engineering and Applications,2007,43(19):167-170.

Abstract: Based on studying on clock synchronization in wireless sensor network,this paper first proposes virtual timestamp and father-burden-chain clock synchronization algorithms which are suitable for single-hop and whole networks respectively. Furthermore,for the clock synchronization algorithms to be robust and low energy consuming,a new method,searching father with children,is proposed,which uses two novel data structures,neighbor table and link tree. Experimental results show that the clock synchronization algorithm is very precise and can be applied in data collection and detection of large-scale wireless sensor networks.

Key words: Wireless Sensor Network(WSN);clock synchronization;virtual timestamp;father-burden-chain

摘 要:通过对无线传感器网络时钟同步算法的研究,提出了适用于单跳网的虚拟时戳时钟同步算法与适用于多跳网的父亲责任链时钟同步算法。为确保整个时钟同步的健壮性与同步过程低的能量消耗,进而提出了携子寻父算法,当利用邻居表构造的层次链路树发生断链时,其以较小的代价快速恢复父亲责任链。实验结果显示其具有较高的精度,适合于大规模无线传感器网络的数据采集与监测等应用。

关键词:无线传感器网络;时钟同步;虚拟时戳;父亲责任链
文章编号:1002-8331(2007)19-0167-04 **文献标识码**:A **中图分类号**:TN92;TP212

1 引言

无线传感器网络是一种新兴的数据采集和处理模式,集成了传感器、微处理器、无线通信模块的节点[1]以 Ad-Hoc 的方式组成大规模多跳的无线自组网。和传统网络相比其节点的处理能力有限,大多数的应用如数据采集、数据融合和数据通信等需由多个甚至整个网络中的所有节点协同完成;时钟同步是保障节点协同工作的重要的支撑技术。

传统的网络时钟同步已存在很多成熟的算法,比如 Internet 中的 NTP 同步算法[2],GPS[3]接收器采用的分布式同步算法等。但它们固有的诸如成本高的局限性,不适合无线传感器网络。由于节点受限,无线传感器网络的时钟同步算法要求具有低的能量开销、低的计算复杂度、良好的扩展性,并支持多跳以实现全网同步。此外,还要满足具体的精度要求。

根据无线传感器网络的特点,本文提出了虚拟时戳时钟同步算法,采用发送者同步模型,实现了低功耗简单的时钟同步,适用于单跳网络;并进一步提出了父亲责任链时钟同步算法,以实现多跳网络的时钟同步。为保证健壮性还提出了邻居发现、链路生成、携子寻父等一系列算法。

2 相关的时钟同步协议与算法

2.1 时钟同步的实现

时钟同步可以由软件或硬件实现。一般包括 3 个步骤:(1)监测是否有新的时钟同步事件发生;(2)估计远程的时钟;(3)校正本地时钟。时钟同步事件由时钟源节点根据具体的策略发出,可以是基于固定周期的,也可以是基于事件触发的。估计远程时钟是指要求同步的节点为响应同步,对远程时钟源的时钟进行估计,它必须根据收到的具体的信息计算出收到信息的延时;由它决定同步精度。本地时钟校正是指节点根据前面的信息对本地的时钟重设,较高精度的时钟同步要考虑校正时的处理延时。

从下文可以看出,本文提出的同步算法与此有很大的差别,传统的同步主要由接收节点对时钟估计,而虚拟时戳与父

基金项目:辽宁省教育厅科学技术研究基金(No.05L002)。
作者简介:赵有俊(1972-),男,讲师,主要研究方向:无线局域网、无线传感器网络、嵌入式网络;曾子维(1963-),男,副教授,主要研究方向:计算机网络技术(Ad Hoc 和传感器网络)、软件工程;臧殿红(1976-),女,讲师,主要研究方向:传感器检测技术、自动化装置。

亲责任链时钟同步算法则主要由发送节点完成对同步时间差的估计。

2.2 现有的无线传感器网络时钟同步算法：

无线传感器网络时钟同步算法主要有以下 3 类：

(1)基于发送者同步模型的算法

例如 DMTS 算法[4]、FTSP 算法[5]。其原理如图 1 所示：主节点竞争信道成功后，给同步广播分组包打上时间戳 T_0，在发送广播分组前，主节点首先发送前导码和同步码，以便于接收点同步，根据发送的信息位个数 n 和发送每个比特位所需时间 t，可以估计出前导码和同步码的发送时间为 nt。接收节点收到前导码和同步码后，记录本地时间 T_1，在开始处理接收到的数据包时，记录本地时间 T_2。电磁波的传播时延忽略不计，则全部的时延为发送前导码和同步码的时延加上 T_2 和 T_1 的差，即接收节点的时钟调整为：$T_0+nt+(T_2-T_1)$。可见此类算法只通过单个广播时间分组，便能同步单跳广播域内的所有节点。

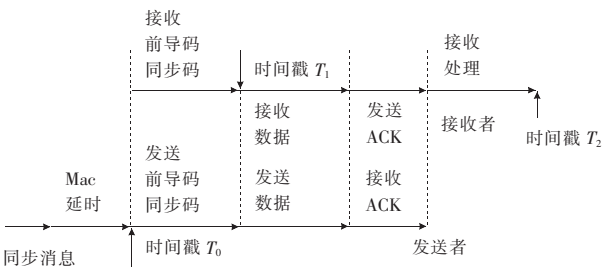

图 1 基于发送者同步模型原理图

(2)基于发送者-接收者交互同步算法

例如 TPSN 算法[6]等。原理同传统的 NTP 时间同步协议[2]，它采用双向通信，可以把传送时延、传播时延和接收时延的影响降低 50%。相邻两个节点通过交换两个消息实现同步，如图 2 所示：等待时钟同步的客户机 C 向同步服务器 S 发送同步请求并记录此时的本地时间 T_1，S 收到该包后记录本地时间 T_2，经过一段时间间隔后发送同步回应包，并附带本地时刻 T_3 以及 T_2，C 收到后记录本地时间 T_4；d 为往返时延，δ 为时间偏差。通过同步信令的交互，可以求出以下的时间 $T_2=(T_1+d+\delta)$；$T_4=(T_3+d-\delta)$。则可以计算出以下数值 $\delta=\frac{(T_2-T_1)-(T_4-T_3)}{2}$，$d=\frac{(T_2-T_1)+(T_4-T_3)}{2}$，节点 C 依次将它的时间同步到节点 S。

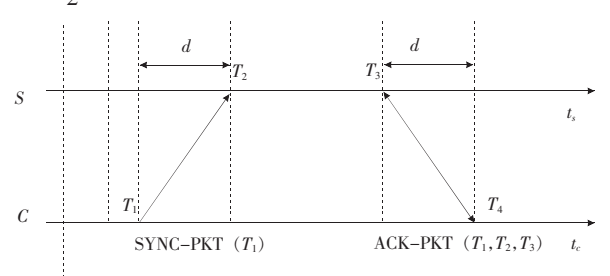

图 2 基于发送者-接收者交互模型原理

(3)基于两个接收者交互同步的算法

例如 RBS(Reference Broadcast Synchronization)[7]，即参考广播同步，需要第三方的广播信息包协助，具体的策略为，第三方发送一个广播信标，需要同步的两个节点接收者 1 和接收者 2 都能接收到这个数据包，并分别记下自己收到这个同步数据包的时间 T_1、T_2，其中接收者 2 就可以根据 T_2、T_1 的差来调整自己的时钟和接收者 1 的时钟保持同步，$T=T+T_2-T_1$，此类算法忽略了传播延时。

3 虚拟时戳与父亲责任链同步算法概述

3.1 虚拟时戳与父亲责任链同步算法简述

如同前文所述的同步算法，虚拟时戳时钟同步算法也是单跳的；然后通过父亲责任链时钟同步算法逐级实现全网同步。整个同步策略包括以下几个步骤：

步骤 1 建立一个稳定的树状层次链路关系，SINK 节点是层次树的根，然后逐渐建立第 1 层，第 2 层，…，第 n 层子树。具体采用的策略是邻居发现和链路形成算法。

步骤 2 采用虚拟时戳同步算法，从 SINK 节点开始由每棵子树的根节点发起单跳间时钟同步，它是基于发送者同步模型的。不同之处是：虚拟时戳算法并不要求根节点和自己的子节点有相同的时钟，但保证同父亲的兄弟节点间有相同的时钟，由父节点维护父子的相对时钟。

步骤 3 采用父亲责任链算法实现全网时钟同步。虚拟时戳算法并没有使父子节点具有一致的时钟，因此必须由父节点维护父子的时钟差。当某个节点要转发孩子节点的信息时，它根据时钟差把孩子的时钟修改为自己的时钟并上报给父亲，以此类推，直至 SINK 节点，即层间的时间调整是在交换信息时由父亲节点负责的，这样就形成一个由叶子节点到 SINK 节点的父亲责任链，从而完成时钟的同步。动态调整时钟同步的目的是减少整个网络的开销。由于存在时钟漂移，全网需要周期同步，在通信量相对较小的无线传感器网络中，这样同步是一个很好的折衷，本文的结语中对此也作了解释，并指出可根据具体应用作相应的改进。

步骤 4 非 SINK 节点检查自己与父亲的链接状态，如果父亲节点失效，则执行携子寻父算法，带领自己的子树寻找最优的父亲节点，并采用基于两个接收者交互同步的同步算法与自己的新兄弟进行时钟同步，调整为新兄弟节点的时钟，同时调整和自己子树的相对时钟差，其子树的时钟不必调整。

步骤 5 新加入节点在周围节点找到层次最小的邻居节点作为自己的父亲，并和自己的兄弟执行兄弟同步算法。

3.2 相关数据结构

为建立和维持一个稳定的层次链路树，每个节点需要维护以下的数据结构：1 个邻居表、1 个变量、3 个指针。

1 个邻居表：邻居表是一个一维数组，每个单元是一个四元组，其中第一行为自己的信息，*link* 域为-1。四元组的数据如下：$N(node\_id,son\_num,lay\_num,link)$，其中 *node_id*：为节点的唯一标志，传感器网络中不同节点的 *node-id* 是唯一的。*son_num*：节点的儿子个数，以便优化层次子树，防止某些子树过于庞大。*lay_num*：节点所在的层数。SINK 节点为 0 层，依次类推。*link*：同类节点的下一个节点的位置，为方便查询。

1 个变量：*time_offset* 用于存储和儿子的时间差，即父子的相对时钟。

3 个指针：*father*，*son*，*other*。*father*：指向自己的父亲节点，对应的父亲节点的 *link* 域为-2。*son*：指向自己的儿子节点，*son* 指向节点的 *link* 域指向该节点的一个同父兄弟，依次类推，最后的一个兄弟的 *link* 为 0。如果没有儿子节点 *son* 指针为-1。*other*：指向其它的邻居节点的指针，断链携子寻父时使用，*other* 指针所指域的最后一个节点的 *link* 值也为 0。

4 具体算法详述

4.1 层次链路树生成

(1)邻居发现算法

网络部署完毕或者网络定期更新时执行。通过 IEEE 802.11 DCF(分布式协调)的 CSMA/CA 与退避策略[8],节点首先侦听信道,尽可能收集信息。把收到的邻居信息例如 node_id 填入邻居表。当信道空闲一定的时间间隔(如一个 DIFS[8])后根据退避策略竞争信道发送自己的信息。邻居表的第零行为自己的信息。以下各行是邻居信息。同时置 *father*,*son*,*other* 指针的值为-1。邻居节点的 *son_num* 为 0,其它的除了 *node_id* 域之外都为-1。这样就形成了一个初始的邻居表。注意此时节点发送的是一个邻居发现包,只包含 *node_id* 信息,是一个广播数据包。

(2)层次链路树生成算法

首先 SINK 节点广播一个层次链路树初始化数据包,数据包包含发送者的 *node_id* 和 *lay_num*。SINK 节点的 *lay_num* 值为 0。其相邻的节点在收到信息后等待一个延时间隔(比如一个 DIFS)后再转发层次链路树初始化数据包,并把接收到的 *lay_num* 加 1 记录到自己的 *lay_num* 域。依次类推。节点如果多次收到这个数据包,则只保留最先得到的(也就是 *lay_num* 值最小的)节点的信息,而且只转发一次此数据包。这是为了保证不会产生或者减少"内爆"(implosion)和"重叠"(overlap)现象[9]。节点转发数据后继续侦听信道,接收邻居信息,填充邻居表中各个节点的 *lay_num*。当所有的邻居节点的 *lay_num* 都不为-1,或者在信道空闲 2 个 DIFS 后(即初始化数据包已经广播到两跳之外),竞争信道同自己的邻居节点之间建立 *father* 域关系。具体做法:发送节点 *a* 组播建立 *father* 域数据包给邻居节点中 *lay_num* 最小但不是-1(*lay_num*=-1 意味着节点在发送完邻居发现包后突然失效)的邻居节点,要求建立 *father* 域联系,在这些回应的节点中选择一个 *son_num* 域最小的节点 *b* 为自己的 *father* 域节点。即把 *father* 的值由-1 改为其选中节点在自己邻居表中的行号的值。同时告诉节点 *b* 被选为自己的 *father* 域及自己的信息。*b* 节点收到后,根据 *son* 指针指向的节点的 *link* 指针找到最后的 *link* 为-1 的节点,并把其 *link* 的值改为 *a* 节点所在邻居表的位置数。如果 *son*=-1,则令 *son* 等于 *a* 在 *b* 邻居表中的位置数。如图 3 所示:节点 *a* 和节点 *e* 在第二跳的位置,*e* 在竞争中首先获得信道并和节点 *d* 形成 *father* 域关系。此后节点 *a* 竞争到信道它可和节点 *b* 或 *d* 建立 *father* 域关系,由于此时节点的 *d* 的 *son_num* 域数为 1,而节点 *b* 的 *son_num* 为 0,所以节点 *a* 和 *b* 建立 *father* 域关系 *a* 成为节点 *b* 的 *son* 域。同时相互修改自己的邻居表中的信息,这样就保证了节点均衡,防止某些节点过度耗能而提前失效。

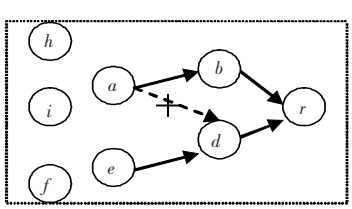

图 3　邻居表 *father* 及 *son* 域的形成

other 指针的形成:*father* 和 *son* 域完成后,在邻居表中自上至下查找 *link* 域为-1 的节点,并把发现的第一个节点的行号作为 *other* 指针的值,然后依次填充的 *link* 域的值为下一个符合条件的节点。如果无此条件的节点则 *other* 值保持-1(即此节点只有孩子和父亲没有兄弟)。

4.2　虚拟时戳与父亲责任链时钟同步算法:

(1)单跳虚拟时戳同步算法

层次链路树形成以后,由 SINK(第 0 层)发起时钟同步。具体的单跳虚拟时戳算法如下:子树根节点根据自己 *son* 及其 *son_num* 域的信息,广播一个同步信息包,并采用 tdma[10]的方式为每一个儿子分配一个时隙,协商后开始单跳间的时钟同步,先发送一个准备消息,每个儿子在自己的时隙里做准备就绪应答,根节点在收到所有儿子节点的准备就绪信号后,发送同步数据包,数据包包括前导码 101010…,具体位数根据实际的硬件而定,接着是同步码 1111…,具体的位数也是根据情况而定,然后是 8 个 bit 的触发码 00000000,最后是 8 个 bit 位的虚拟时戳 11111111。收到触发码和虚拟时戳后儿子节点使自己的计时器清零。根节点打上的虚拟时戳 11111111 是其发送完同步信息后的本地时钟,但没有把它发送到儿子节点,而是记录在变量 *time_offset* 中,为后面的父亲责任链同步算法所用。由此可以看出这个同步时戳是虚拟的,因为它只是使所有儿子节点的时钟同时为 0,但对于父亲的时钟儿子们不得而知。父子的时钟差由父亲节点维护。

(2)多跳的父亲责任链时钟同步

所有子树根节点(包括 SINK 节点)执行完虚拟时戳同步算法后,在有数据转发时动态执行父亲责任链时钟同步算法。具体算法如下:节点 *b* 收到儿子节点 *s* 要求转发的数据包;节点 *b* 转发时处理数据包的产生时间戳 T_1,令 $T_1=T_1+time\_offset$,并转发给自己的父节点 *f*,要求其继续转发,依次类推逐级累加直到 SINK 节点为止。

(3)小结

虚拟时戳算法保证了同根节点有一致的时钟,且误差很小,误差来源理论上只有传播延时,在无线传感器网络中节点间隔一般只有几米到几十米,它们距父亲节点的距离大多在 10 m 左右,传播延时忽略不计不会影响精度。父亲责任链算法修正层间的时钟差,动态维护全网的时钟同步,它类似世界时区时间差,无信息交换时的时钟同步是无意义的。

此外采用分时而不是基于竞争的虚拟时戳算法的目的是使根节点有更大的优先权,从而减少因竞争信道而带来的不确定性,保证所有的儿子节点准备就绪后,只经过一次就可以同步完所有儿子,在具体实现上可采用低复杂度的算法,参见第 5 章,实验比较与结论。

4.3　携子寻父及其它支撑算法

(1)链路检查算法

节点侦听信道,如 *father* 域节点占用信道,则链路正常,返回。否则竞争获得信道后向父节点发送 *hello* 数据包,如果回应则返回。否则,侦听信道后再次发送 *hello* 信息,不成功后执行携子寻父算法。注意两次 *hello* 数据包发送间隔大于一个发送最大数据包分组的时间,因为无线传感器网络大都采用周期睡眠策略,这样可以消除隐藏终端带来的影响[11]。

(2)携子寻父算法

当某个节点失效后以其为根的子树必须寻找新的父亲以完成到 SINK 节点的数据传输,此子树的根找到父亲后再和新的兄弟执行兄弟时钟同步算法。具体的算法如下:首先查找 *other* 是否为-1,如果为-1,说明其为一棵孤立的子树,不再应答孩子的数据转发与其它有关信息交换请求,算法失败返回,并等待网络拓扑重构。如果 *other* 不为-1 则在其 *link* 域中找

lay_num 比自己小 1 的节点,交换信息,如果此节点父亲没有失效则把它作为自己的父亲。并查找新的兄弟,执行兄弟时钟同步算法。如果没有符合条件的则找一个 *lay_num* 值和自己相等的节点并交换信息,如果此节点不和自己同父(因为上述的数据结构不能保证这样的结点和自己有不同的父亲)并且其父亲节点没有失效,则把其作为自己的父节点,然后把自己的 *lay_num* 加 1,且告诉自己的子孙节点逐级加 1。然后找到新的兄弟进行时钟同步。如果上述条件都不满足,则等待一段时间,然后在 *other* 域中查找满足如下条件的节点:①*lay_num* 值和自己相同,②其 *father* 域和自己不同,③其父节点没有失效。找到后设其为父亲,交换信息双方修改邻居表信息。否则不再应答孩子的数据转发与其它有关信息交换请求,算法失败返回,并等待网络拓扑重构。此段算法的目的在于处理如下情况如图 4,如果节点 *b* 失效且节点 *d* 首先执行寻父算法,但是节点 *c* 不是它的邻居而是 *e* 的邻居,此时节点 *d* 携子寻父失败,它必须等待节点 *e* 寻父(把节点 *c* 设为父亲)成功后,才能把 *e* 设为自己的父亲节点,所以节点 *d* 等待一段时间以便节点 *e* 能够建立稳定的父子关系后,自己再把它设为父节点。

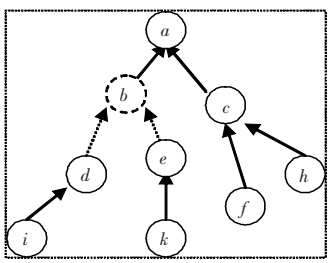

图 4 携子寻父算法图示

(3)兄弟同步算法

子树寻父成功后必须和自己的兄弟保持同步,具体采用 RBS 策略,首先父亲发送一广播数据包,当其中的同步特征码到达时,同步的两个孩子节点分别记下自己的时间,然后要求同步方接收兄弟的时间,并根据两者的时间差调整自己时钟,具体参考文献[7]。注意算法中同时修改自己与孩子的相对时钟。

5 实验比较与结论

5.1 实验配置与结果

本文采用 GAINS3[12]节点验证上述算法,并以此评价其性能。GAINS3 节点兼容伯克利大学的 MICA2[13]节点,处理器采用 ATMEL 公司的 ATmega 128L AVR[14]单片机,它内含 8 MHz 的时钟电路,实验使用其 16 位的定时计数器 1,采用 8 分频,计数间隔为 1 000,即计数器每 1 us 增加一个单位,每 1 ms 计数器清零,因此实验中的时钟可以精确到微秒级。其射频模块为 cc1000[15],实验采用的载波频率为 433 MHz,为便于实验比较,配置其传输速率为 19.2 kbps 与 38.4 kbps 两种情况;信道编码为曼彻斯特编码。虚拟时戳时钟同步数据包的采用 8 个字节,前导码和同步码各为 3 个,触发码和虚拟时戳各为 1 个字节。实验构造一个跳数为 10 的网络,每个节点配有 1 个感光传感器,设置为实验室白炽灯打开瞬间,各个节点对此事件计时,并向 SINK 节点汇报,设 SINK 节点为 0 号节点,1 号为第 1 跳,以此类推。实验结果如图 5 所示:纵坐标表示 1 跳到 10 跳平均时间差,单位为 us,其中上面是传输速率为 19.2 kbps 的结果,下面是 38.4 kbps 的结果。

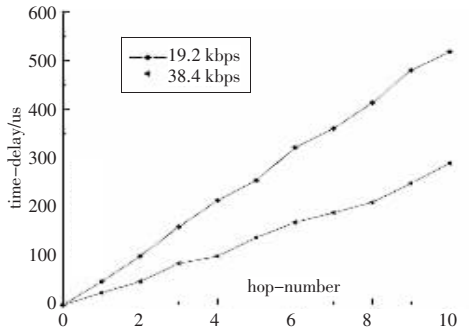

图 5 实验数据结果

5.2 实验结果分析与结论

从实验结果曲线可以看出,传输速率越高得到的时间差就越小,而且随着跳数的增加,基本呈线形增加趋势。同时还可以发现在传输速率为 19.2 kbps 时,每一跳的时间差大约增加 50 us 左右。而传输速率为 38.4 kps 时每一跳时间差增加大约为 25 us 左右,两者的比都为 2。因此如果把 19.2 kbps 的父亲结点的 *time_offset*(即父子相对时钟)在原有的基础上再加上 50 us,则可以保证每跳的延时可控制在 10 us 之内。而对于 38.4 kbps 的情况,相对时钟在原有的基础上再加上 25 us,也可以达到要求的效果。究其原因在于算法的虚拟时戳是打在 mac 层上的,而通过物理层调制发射与接收后就产生一个固定的延时,而且传输速率越高,延时越小。

ATmega128LAVR 单片机的指令多为单周期指令,工作在 8 MHz 时每条指令约执行 1/8 us,因为算法忽略处理延时,实验中数据包传送没有采用 CRC 算法,其正确性由孩子节点在分配给自己的 TDMA 时槽里应答准备消息包保证。子节点定时器清零前只做 8 次比较,故处理延时约为 1 us,另外,假设节点间距离为 100 m 传播延时为 100/(3×108)约为 0.3 us;此外传输速率为 19.2 kbps 时每 bit 的传输延时约为 52.08 us 所以总误差每跳在 53 us 左右,和实验结果基本相符。这个精度完全满足无线传感器网络的要求。算法采用较小的数据包与较小的计算量,减少了数据的收发量与处理开销,从而大量节约了的能量。

6 结语

本算法是在研究无线传感器网络自适应紧急上报与兴趣命令协议时提出的,因而不适合诸如目标跟踪、定位等应用。但适当改进即可胜任。如从 SINK 节点开始逐级把父子的相对时钟传给儿子节点,并由儿子节点修正自己的时钟,全网时钟就一致了。虚拟时戳同步的真正意义在于消除了基于发送者同步模型算法在传送时戳时误差的不确定性,并减少了信息量与计算量,节约了能量消耗。如果考虑传输延时,上述实验精度在可在 5 us 之内,调整值根据具体硬件获得。此外由于篇幅所限,本文省略了所有的算法伪码、数据帧格式、数据结构图例等内容,但这不影响对整个时钟同步算法的理解。

(收稿日期:2006 年 11 月)

参考文献:

[1] 任丰原,黄海宁,林闯.无线传感器网络[J].软件学报,2003,14(7):1282-1291.

[2] Mils D L.RFC1305,Network Time Protocol(Version 3) Specification, Implementation and Analysis[S],March 1992.

(下转 176 页)