一种用于大规模无线传感器网络的时钟同步算法

郝 纲 庄 毅
(南京航空航天大学计算机科学与技术学院 南京 210016)

摘 要 针对经典的时钟同步算法在大规模无线传感器网络中存在的同步精度低与能量消耗高等问题,提出一种基于簇-树结构的无线传感器网络时钟同步算法。首先,建立一棵以网关为根节点、簇首为子节点的生成树来减少网络中节点同步时的累计跳数;然后,在该生成树的基础上采用簇间双向的 SRS 和簇内单向的 ROS 同步机制进行同步,在保证同步精度的前提下减少网络同步所需的消息数量。实验结果表明,相比传统的 RBS 和 TPSN 算法,提出的簇-树结构同步算法可使网络的平均同步精度保持在更高的水平,并有效地降低网络同步时节点的能耗。

关键词 无线传感器网络,时间同步,精度,能耗,跳数
中图法分类号 TP393 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2015.12.041

Time Synchronization Algorithm for Large-scale Wireless Sensor Networks

HAO Gang ZHUANG Yi
(College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing 210016, China)

Abstract Concerning the problem that typical time synchronization algorithms used in large-scale wireless sensor network have a low accuracy and high energy consumption, this paper proposed a time synchronization algorithm based on cluster-tree structure. First, the algorithm establishes clusters-based spanning tree to reduce the hop count of synchronization, and then uses two-way SRS in inter-cluster and one-way ROS in intra-cluster to reduce the number of messages required for the network synchronization. The experimental results show that compared with the RBS and TPSN algorithm, the proposed algorithm can keep the network synchronization precision in higher level, and effectively reduce energy consumption of sensor nodes.

Keywords Wireless sensor network, Time synchronization, Accuracy, Energy consumption, Hop count

1 引言

近些年来随着微电子和传感器技术的日渐成熟,大规模无线传感器网络凭借其容错能力强、检测范围广等优点取得了广泛的应用。但网络中较多的节点数目、较大的信息流量以及较高的同步跳数等问题对大规模无线传感器网络同步时的节点精度和能量消耗提出了新的要求。

2 相关工作

自从 Jeremy Elson 和 Kay Romer 等人在 2002 年首次提出无线传感器网络时间同步这一研究课题以来,该领域的研究人员已陆续提出了多种经典的时间同步算法,这些算法总体上可以分为以下两种。

1)基于 RRS(Receiver-Receiver Synchronization)的同步机制,其典型代表为参考广播同步算法 RBS(Reference Broadcast Synchronization)[1,2]。该算法很好地排除了发送节点的发送时间和接收时间对时钟同步精度造成的影响,但当网络规模较大时需要进行大量的信息交换,节点的能量消耗比较大。

2)基于 SRS (Sender-Receiver Synchronization)的同步机制,该机制又可以细分为基于单向消息传输的发送-接收同步和基于双向消息交换的成对同步两类,其典型的代表分别为 FTSP(Flooding Time Synchronization Protocol)[3] 和 TPSN (Time Synchronization Protocol for Sensor Network)[4]。TPSN 算法虽然具有较高的时钟同步精度,但当网络中节点密度较大时,节点的同步跳数会明显增加,从而对同步精度造成较大影响;而基于单向消息传输的 FTSP 算法通过将通信时延进行精细划分并分别进行处理来减小时延对同步精度的影响,虽然提高了网络同步的精度,但由于采用洪泛的方式,网络的能量消耗较大。

此外,Kyoung-lae Noh 等人将 RBS 和 TPSN 进行结合,提出了 PBS (Pairwise Broadcast Synchronization)[5]时钟同步算法。该算法采用一种新的 ROS(Receiver-Only Synchronization)同步机制,可在不影响同步精度的基础上减少网络通信开销,但它需要在网络中部署两个超级节点使其通信范围覆盖网络中其他所有节点,因此该算法主要应用在规模较小的单簇网络中。

文献[6]和文献[7]中提出当 WSN 规模增大时,网络中

到稿日期:2014-12-07 返修日期:2015-03-18 本文受国家自然科学基金青年科学基金项目(61202351),江苏省普通高校研究生科研创新计划资助项目(CXZZ13_0171)资助。

郝 纲(1989—),男,硕士生,主要研究方向为无线传感器网络,E-mail:capehg@163.com;庄 毅(1956—),女,教授,主要研究方向为计算机网络、分布计算。

节点的同步跳数也会随之增加，从而使网络的同步精度随跳数的累计而有所降低，并带来较大的能量开销。综合以上分析可知，经典时钟同步算法并不能很好地适用于规模较大的WSN。为此，本文在充分考虑TPSN和PBS算法优势的基础上，提出了一种用于大规模WSN的簇-树结构时间同步算法CTTS(A Time Synchronization Algorithm based on Cluster-Tree)。该算法结合SRS和ROS两种机制进行网络同步，并采用二次回归方法对节点同步时的偏移量进行优化，在保证同步精度的情况下有效降低节点的能量开销。

3 簇-树结构时钟同步模型

本文提出的簇-树结构同步模型中共包含网关(Sink Node，SN)、簇首(Cluster Head，CH)和簇内成员(Cluster Member，CM) 3类节点，如图1所示。假设每个传感器节点具有唯一的ID，其中网关除了和普通节点具有相同的传输半径外，还具有更多的能量，它作为生成树的根节点控制整个簇首生成树的建立过程，并维护一张记录其直接子节点ID的信息表；簇首作为生成树中的支路节点，维护着一张记录其父节点、子节点、簇内成员节点ID以及在生成树中所处深度的信息表；每个簇内成员节点属于且仅属于一个簇，并维护一张记录其所属簇的簇首节点ID的信息表。

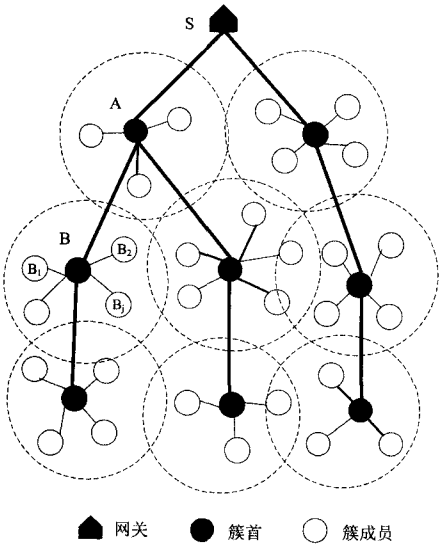


图1 簇-树结构同步模型

在该同步模型中，网络节点的同步首先由网关发起，并沿着生成树的通路从已经完成同步的低深度簇首向未同步的高深度簇首进行[8]，同步过程分为簇间同步和簇内同步两个阶段。其中簇间同步采用SRS同步机制，按照自顶向下的原则逐步实现不同深度的簇首的同步；而簇内成员节点的同步采用ROS机制，通过对其所属簇的簇首与上一层簇首同步过程的监听完成自身的同步。

图2为模型中节点同步时序，其中节点A、B分别为图1中具有父子关系的簇首，B_j为簇首B内第j个簇内成员节点。假设在第i轮同步中簇首A在本地时钟$T_{1,i}^{(A)}$向簇首B发送同步开始信息syn_start，该消息中包含时间戳$T_{1,i}^{(A)}$；簇首B在本地时间$T_{2,i}^{(B)}$收到该消息，并在其本地时间$T_{3,i}^{(B)}$向簇首A发送应答消息start_ack，该消息中包含$T_{1,i}^{(A)}$、$T_{2,i}^{(B)}$和$T_{3,i}^{(B)}$ 3个时间戳；簇首A在收到消息start_ack后的$T_{5,i}^{(A)}$时刻向簇首B发送消息syn_over，该消息中包括时间戳$T_{3,i}^{(B)}$、$T_{4,i}^{(A)}$和$T_{5,i}^{(A)}$，则根据文献[4]中的SRS机制可以得到模型中簇首B与簇首A的时钟偏移δ_i，如式(1)所示，其中d为传播距离产生的时延。

$$\delta_i=\frac{(T_{4,i}^{(A)}-T_{2,i}^{(B)})+(T_{1,i}^{(A)}-T_{3,i}^{(B)})}{2} \tag{1}$$

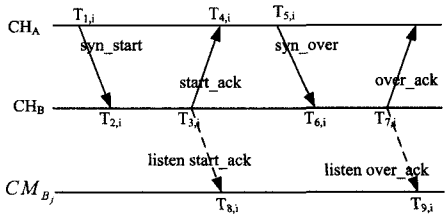


图2 节点同步时序

虽然PBS算法利用了节点的广播特性，在保证同步精度的情况下降低了节点的能耗，但该算法要求待同步节点处于两个参考节点共同的通信范围内，并没有考虑处于单个参考节点通信范围内的传感器节点，如1图中的簇首B的成员节点B_j(该类节点处于B的通信范围而不处于A的通信范围)。为了通过ROS方式实现该类节点的同步，本文采用增加一次消息发送的方式对PBS算法进行改进。即当簇首B完成同步后，在$T_{7,i}^{(B)}$时刻向簇首节点A发送一次是结束应答消息over_ack，该消息中包含上一个消息start_ack到达簇首A的时间，簇首B内各成员节点监听到over_ack后，根据消息中包含的时间戳建立如下方程，如式(2)所示。

$$\begin{aligned}T_{4,i}^{(A)}&=T_{3,i}^{(B)}+d+\theta_i\\T_{8,i}^{(B_j)}&=T_{3,i}^{(B)}+d+\theta_i\end{aligned} \tag{2}$$

从而得到簇首B簇成员节点B_j和与簇首节点A之间的时钟偏移，如式(3)所示。在该同步过程中成员节点B_j通过监听的方式完成与簇首A的时钟同步，而不用发送任何消息，有效减少了网络同步时消息的交换数量。

$$\theta_i=(T_{8,i}^{(B_j)}-T_{4,i}^{(A)}) \tag{3}$$

本文将SRS和ROS两种机制结合，很好地解决了同步模型中簇首和簇内成员两类节点的时钟同步问题。但在任意第i轮同步过程中，簇首B采用SRS方式与簇首A进行同步时其时钟的调整量依据式(1)，而簇首B内成员节点通过ROS方式与簇首A进行同步的调整量依据式(3)，即网络中到同一个簇的簇首与簇内成员节点在同步过程中的精度存在一定差异。

为此，本文采用二次回归方法解决算法在SRS和ROS两种机制转换过程中精度不一致的问题。

令y_i代表节点在第i轮同步后的时钟的实际调整量，δ_i为节点在第i轮同步过程中通过SRS或ROS方式得到的调整量，定义如式(4)。

$$J_n=\sum_{i=1}^{n}(y_i-\delta_i)^2 \tag{4}$$

则$\forall i\in[1,n]$，当J_n达到最小值时，可求得满足要求的y_i。为此，假设在整个同步过程中时钟调整量y_i可以由同步轮数i的二次方程所表示，如式(5)所示，其中a_n、b_n、c_n分别为二次方程中的系数项。

$$y_i=a_ni^2+b_ni+c_n \tag{5}$$

令$h_n=[a_n \quad b_n \quad c_n]$，则当$\frac{\partial J_n}{\partial h_n}=0$时，可得到一组满足要求的系数方程组。对该偏导数在$\hat{h}_{n-1}$进行二次泰勒级数展开，如式(6)所示。

$$\frac{\partial J_n}{\partial h_n}=\frac{\partial J_n}{\partial h_n}\bigg|_{h_n=\hat{h}_{n-1}}+\frac{\partial^2 J_n}{\partial h_n^2}\bigg|_{h_n=\hat{h}_{n-1}}(h_n-\hat{h}_{n-1}) \tag{6}$$

对式(6)进行简单变换后可以得到每轮迭代后的 h_n，如式(7)所示。

$$h_n=\hat{h}_n-(\frac{\partial^2 J_n}{\partial h_n^2}\bigg|_{h_n=\hat{h}_n})^{-1}(\frac{\partial J_n}{\partial h_n}\bigg|_{h_n=\hat{h}_n}) \tag{7}$$

求得每轮同步中二次方程的系数通解，并根据同步轮数 i 得到节点的时钟调整量 y_i，从而实现簇首和簇内成员节点同步时在 SRS 和 ROS 两种机制间的平滑过渡。

4 算法实现

CTTS 算法的实现分为簇-树拓扑建立和时钟同步两个过程。在拓扑建立过程中，首先进行网络分簇确定簇首和簇内成员节点，然后在网关和簇首间完成簇-树网络拓扑的建立；时钟同步分为簇间同步和簇内同步两个阶段，在簇间同步的过程中完成簇内成员节点的同步。

4.1 簇-树拓扑建立

算法在簇-树网络拓扑的建立过程中采用经典的 LEACH[9] 低功耗算法来实现分簇。LEACH 中所有节点自动生成一个 0-1 间的随机数，本地随机数小于设定阈值的节点当选簇首，并广播自己成为簇首的消息，其余的节点根据接收到的消息信号强度判断距离该簇头节点的距离[10]，并按照就近原则选择要加入的簇。

当网络中的各节点都确定了自己的簇首或簇成员身份后，CTTS 算法开始以网关为根节点、簇首为子节点构建一棵簇首生成树。首先，网关以父节点的身份向所有簇首发送建树消息 construct_msg，如果某个尚未加入到生成树中的簇首收到该消息，表明该簇首和网关间不需要经过其他节点的转发就可以直接进行双向通信，则将该簇首节点作为网关的孩子加入到生成树中，并分别对父节点和子节点维护的信息表进行更新。然后，新加入的各簇首作为父节点继续发送 construct_msg 消息，并不断重复以上过程，直到支路上的最新加入的簇首维护的信息表中孩子节点始终为空，则认为该簇首节点为叶节点，支路的建树过程结束。

在生成树建立过程中，为避免已经成功加入某条支路中的簇首重复收到 construct_msg 消息，算法为每个簇首节点设置一个初始值为 TRUE 的标记 construct_flag，如果节点第一次收到 construct_msg 消息，则对消息进行解析，并立即设置标记 construct_flag 为 FALSE，表明该阶段将拒绝接收其他簇首发送的 construct_msg 消息。同时，由于簇首生成树的建立过程始终是在网关和簇首间进行的，因此在此过程中设置各簇内成员节点拒绝接收任何消息。以下为模型建立的主要步骤：

Step1　网关节点首先设置自己的 depth 为 0，然后向所有簇首广播 construct_msg 消息，该消息中包含自身的 ID 和所处深度 depth。

Step2　某个簇首第一次收到 construct_msg 消息时，首先设置 construct_flag 为 FALSE，然后通过对该消息的解析得到自己的父亲节点 ID 和自己在生成树中的深度 depth，并对自己维护的信息表进行更新。

Step3　簇首完成对消息 construct_msg 的处理后，随即向外发送一条应答消息 construct_ack，该消息中包含自身和其父节点 ID 值。

Step4　簇首收到消息 construct_ack 后，首先用自己的 ID 与消息中包含的父节点 ID 值进行对比，判断该消息是来自其孩子节点；如果是，则对该消息进行解析，得到其孩子节点，并对自己维护的信息表进行更新；如果为否，则直接丢弃该消息。

Step5　已经成功加入到生成树支路中的簇首在发送 construct_ack 消息完后，用自己的 ID 和深度值对 construct_msg 消息进行更新，并以父节点身份向外转发 construct_msg 消息。

Step6　当其他簇首收到转发的 construct_msg 后，首先检查标记 construct_first 的值，如果为 TRUE，则转到 Step2；否则，直接丢弃该消息。

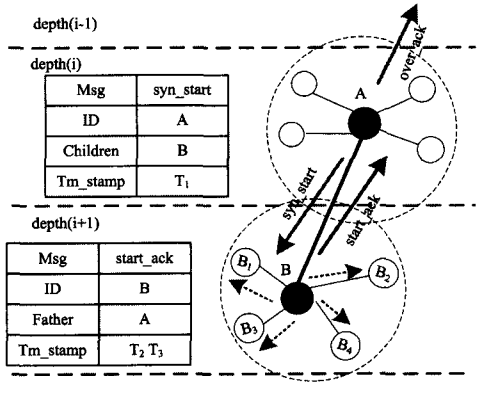
Step7　如果支路中的簇首确认自己为叶节点，则设置标记 leaf_flag 为 TRUE，并发送 construct_over 消息通知父节点该支路上生成树建立结束，该消息中包含自身和父节点 ID。

Step8　父节点收到该消息后用自身和其父节点 ID 对 construct_over 消息进行更新，并沿着该支路继续向其父节点转发该消息。

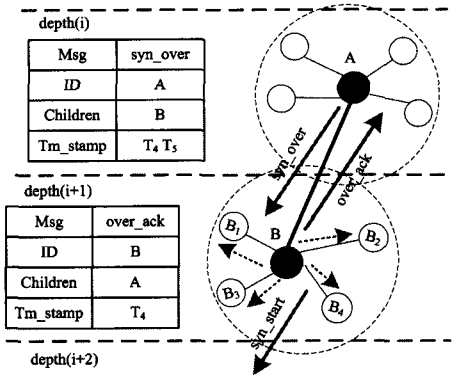
Step9　当网关收到其所有直接子节点发送的 construct_over 消息后，该簇首树的建立过程结束。

4.2 时钟同步

由于算法在完成簇-树的建立之后，在生成树的任一支路上具有父子关系的簇首可以直接进行通信，因此在此类节点间采用双向 SRS 机制实现簇间时钟同步；同时，由于簇间的同步是从生成树中已经完成同步的低深度簇首向需要同步的高深度簇首进行的，且在该过程中簇内成员节点可以接收到其所属簇的簇首发出的所有同步消息，因此算法采用 ROS 机制，通过单向监听的方式实现簇内时钟同步。图 3 为簇-树模型中具有父子关系的簇首节点间同步时的消息交换过程。



(a)Start 消息交换过程



(b)Over 消息交换过程

图 3　WSN 中节点的消息交换过程

为了便于算法描述，本文用 CH_i 表示在生成树中深度 $depth$ 值为 i 的簇首，CM_j^i 表示簇首 CH_i 内的第 j 簇成员节点。因为簇首的同步是沿着生成树的通路自顶向下进行的，所以假设在簇首 CH_{i+1} 与 CH_i 进行同步时，CH_i 节点已经完成与上一层父节点 CH_{i-1} 同步。下文以生成树中的一条支路为例来说明算法的具体实现。

Step1　首先 CH_i 节点向深度为 $i+1$ 的簇首 CH_{i+1} 广播开始同步消息 syn_start，该消息中包括 CH_i 的 ID 和发送该消息的时间戳 T_1。

Step2　簇首 CH_{i+1} 收到该消息后，首先通过消息中的 ID 判读该消息是否来自其父节点。如果是，则解析出消息中的时间戳 T_1，并记录下其到达本地的时间 T_2；否则，直接丢弃该消息。

Step3　簇首 CH_{i+1} 处理完消息 syn_start 后，在 T_3 时刻发送应答消息 syn_ack，该消息中包含时间戳 T_1、T_2、T_3 和其父节点的 ID。

Step4　簇首 CH_{i+1} 的簇内成员节点 CM_j^i 监听到应答消息 syn_ack 后，随即记录下 start_ack 到达的本地时间 T_8。

Step5　簇首 CH_i 在 T_4 时刻收到来自簇首节点 CH_{i+1} 的应答消息 syn_ack 后，首先根据 syn_ack 中包含的父节点 ID 判断该消息是否来自其孩子节点。如果是，则对消息进行解析获取时间戳 T_1、T_2 和 T_3；否则，直接丢弃该消息。

Step6　簇首 CH_i 在 T_5 时刻向簇首 CH_{i+1} 发送同步结束消息 syn_over，该消息中包含时间戳 T_1、T_2、T_3、T_4 和消息发送时间戳 T_5。

Step7　簇首 CH_{i+1} 在收到 syn_over 消息后，根据消息中包含的时间戳关系和消息到达本地时间的 T_6，首先通过式(1)计算出自身与父节点 CH_i 的时钟偏移 δ_i。如果当前簇首为首轮同步，则取 $y_i=\delta_i$；否则通过式(7)，根据当前同步轮数 i 和前一次同步时节点的时钟调整通解 $\hat{h}_{n-1}$ 计算出 h_n，并将其回代到式(5)得到网络中簇首 CH_{i+1} 的时钟偏移 y_i，完成与父节点的时钟同步，如图 3(a)所示。

Step8　簇首 CH_{i+1} 完成同步后，在 T_7 时刻向其父节点回复同步结束 over_ack 应答，该消息中包含前一个应答 syn_ack 到达的时间 T_4。

Step9　簇首 CH_{i+1} 的簇内成员节点 CM_j^i 监听到应答消息 over_ack 后，从该消息中提取出 CH_{i+1} 发送的上一个应答消息 start_ack 到达簇首 CH_i 的时间 T_4，并根据式(2)计算出自己与簇首 CH_i 的时钟偏移 θ_i。如果当前簇内节点为首轮同步，则取 $y_i=\theta_i$；否则根据式(7)和 $\hat{h}_{n-1}$ 计算节点当前同步时的 h_n，最后通过式(5)得到 CM_j^i 与 CH_{i+1} 的时钟偏移 y_i，实现网络中簇内成员节点的时钟同步，如图 3(b)所示。

Step10　CH_{i+1} 在下一时刻用自己的本地时间和 ID 对 syn_start 进行更新，向其孩子节点 CH_{i+2} 转发该消息，并重复 Step1－Step9，直到生成树中叶节点完成同步为止。

Step11　同步过程结束。

5　仿真实验与结果分析

在 400m×400m 的方形区域内进行 8 组仿真实验，每组实验部署节点数目分别为 50，100，150，200，250，300，350 和 400，并分别运行 TPSN、RBS 和 CTTS 算法从节点同步精度和能量消耗两方面进行对比分析，实验中节点的传输半径设置为 15m，实验同步过程进行 10 轮。

5.1　WSN 同步跳数分析

图 4 示出采用不同算法时 WSN 的平均同步跳数。从图中可以看出，当网络中节点数目逐渐增大时，CTTS 算法的网络平均同步跳数明显少于 TPSN 和 RBS。图 5 表示当采用不同算法时，网络中同步跳数少于等于平均值的节点在整个 WSN 中所占百分比。通过该组对比实验可以得出，CTTS 算法建立的簇-树同步模型可以有效地减少网络的同步跳数，并使大多数节点处于较少的同步跳数内，很好地解决了大规模 WSN 中同步跳数过多的问题。

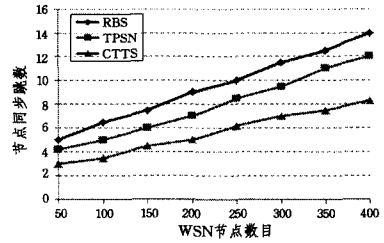


图 4　WSN 平均同步跳数

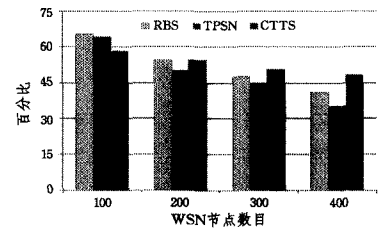


图 5　同步跳数不大于平均值的节点所占百分比

5.2　WSN 同步误差分析

表 1(a)、1(b)分别列出 WSN 节点数目为 50 和 400 时各算法的误差情况。通过对比可以发现，当网络规模较小时，采用不同算法的网络中节点最大误差均能维持在相对较低的水平；而当网络中节点数量增加到 400 时，虽然网络中节点最大误差均成倍增加，但在采用 CTTS 算法的网络中，同步误差小于等于平均值的节点所占比例仍保持在相对稳定的水平。

表 1　算法同步误差对比

(a)节点数目 50 个

算法	RBS	TPSN	CTTS
最大误差(μs)	38.2	27.3	29.1
最小误差(μs)	13.2	7.7	8.5
平均误差(μs)	33.7	22.5	24.2
小于等于平均误差节点所占百分比	62.1%	59.5%	55.0%

(b)节点数目 400 个

算法	RBS	TPSN	CTTS
最大误差(μs)	208.3	163.2	90.2
最小误差(μs)	15.5	10.2	11.3
平均误差(μs)	121.4	92.4	80.5
小于等于平均误差节点所占百分比	41.4%	45.1%	51.3%

图 6 反映了 CTTS 和 TPSN、RBS 算法中 WSN 平均时钟同步误差随节点数目增加而产生的变化趋势。从图中可以看出，相比 CTTS 算法，TPSN 和 RBS 误差增大趋势较为明显。因为在采用多跳的链式结构的算法中，当网络规模增大时节点的同步跳数会明显增加，由此带来通信的延迟和同步误差

的增大。而3种算法中CTTS的平均同步跳数最少，且该算法在网络同步过程中采用了二次回归方法对SRS和ROS间存在差异的调整量进行补偿，所以网络的平均同步精度仍能保持在较高的水平。

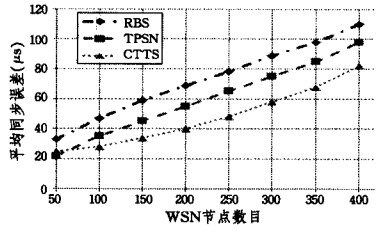


图6 WSN误差变化趋势

5.3 WSN能耗分析

网络在同步过程中，各节点均需要消耗一定的能量。为此，本文根据Heinemann等人提出的能耗模型[12]，对网络中各节点在发送和接收数据过程中的能耗进行分析。

$$E_S(k,d)=\begin{cases}k*E_{elec}+k*\varepsilon_{fs}*d^2, & d\leqslant d_0\\ k*E_{elec}+k*\varepsilon_{amp}*d^4, & d>d_0\end{cases} \tag{8}$$

$$E_R=k*E_{elec} \tag{9}$$

式(8)表示节点发送$k$bit数据时消耗的能量。式中E_{elec}表示节点物理硬件收、发单位数据所需的能量；ε_{fs}和ε_{amp}分别为两种信道模型下功率放大时的能耗参数；d表示发送节点和接收节点间的距离，当d小于等于阈值d_0时，采用自由空间模型，当发送距离d大于d_0时，采用多路径衰减模型。式(9)表示节点接收$k$bit数据的能耗。

图7为采用不同算法时，WSN所有节点在各同步轮次下的平均剩余能量对比图，可以看出CTTS的同步能耗明显低于RBS和TPSN。这是由于采用CTTS算法时，占网络中绝大多数节点的簇内成员节点不发送任何消息，只需通过式(9)直接计算其接收能耗，且这些成员节点离各自簇首的距离都在一跳范围之内，因此可采用式(8)的自由空间模型计算各簇首的发送能耗；TPSN算法中所有节点均需进行双向的信息发送和接收，即每一个节点均会有接收和发送能耗；而采用RBS算法的大规模网络在同步过程时，虽然有小部分节点只存在接收能耗，但由于其单一的网络拓扑结构，占网络大多数的双向通信节点须通过式(8)中的多路径衰减模型计算其同步时的发送能耗。

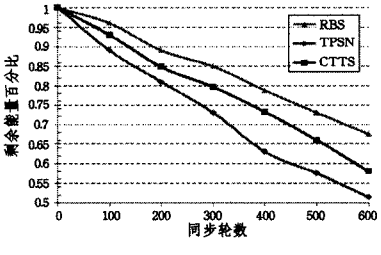


图7 WSN同步能耗

本文亦从网络同步过程中消息的复杂度上对CTTS算法的能耗进行分析。假设在WSN中传感器节点的数目为L，网络共进行N轮同步，则根据文献[5]可知RBS和TPSN算法进行N轮同步需要的最大消息数如式(10)所示。可以看出N_{TPSN}的大小与传感器节点的个数成正比，而N_{RBS}与节点个数的平方成正比。而当在以上算法应用于大规模WSN中时，网络需要交换大量的消息。

$$\begin{aligned}N_{RBS}&=N+L(L-1)/2\\ N_{TPSN}&=2N(L-1)\end{aligned} \tag{10}$$

本文提出的CTTS算法中节点的同步过程只有簇首节点发送同步消息，又根据文献[9]可知网络分簇完成后簇首数量只占总节点数的25%～40%，而网络中占绝大多数的簇内成员节点在同步过程中不需要发送消息，只需通过单向监听的方式即可实现与簇首节点同步，所以CTTS算法同步过程需要交换的最大消息数可以通过式(11)粗略表示。

$$N_{CTTS}=(25\%\sim40\%)N_{TPSN} \tag{11}$$

即在相同的网络规模下，CTTS算法进行同步时所需交换的消息数目明显小于TPSN和RBS。由于WSN在时钟同步过程中的能量开销与节点发送的消息总量直接相关，因此本文通过对各算法在同步过程中交换的消息数目进行了实验对比。从图8可以看出，在网络规模增大的情况下，相同网络规模下节点进行同步时，CTTS算法的开销要小于RBS和TPSN，进一步证明了该算法可以很好地适用于能耗有限的大规模WSN。

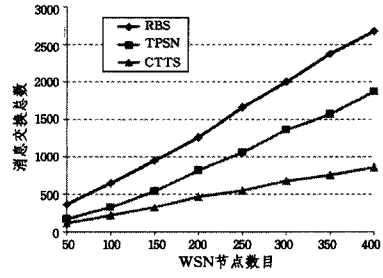


图8 WSN交换的消息总数

结束语 本文针对现有的经典时钟同步算法在大规模无线传感器网络中存在的不足，提出一种基于簇-树结构的时钟同步算法。该算法采用SRS和ROS两种同步机制实现全网节点的时钟同步，并引入了二次回归分析方法对节点同步时的调整量进行优化，在保证同步精度的前提下有效地减少了节点的能量开销。仿真实验结果显示，该算法可以很好地适用于对同步精度和能耗要求较高的大规模WSN。一方面，由于CTTS算法基于较为成熟的路由协议，因此本文没有对网络同步时节点间通信距离对WSN能耗的影响进行详细分析；另一方面，算法暂未考虑网络同步过程中簇首节点失效和新节点加入的情况。在下一步工作中将重点解决以上两个问题。

参考文献

[1] 徐朝农，徐勇军，李晓维. 无线传感器网络时间同步新技术[J]. 计算机研究与发展，2008，45(1)：138-145
Xu Chao-nong，Xu Yong-jun，Li Xiao-wei. New Time Synchronization Techniques for Wireless Sensor Networks [J]. Journal of Computer Research and Development，2008，45(1)：138-145

[2] Elson J，Girod L，Estrin D. Fine-grained network time synchronization using reference broadcasts[J]. ACM SIGOPS Operating Systems Review，2002，36(SI)：147-163

[3] Maróti M，Kusy B，Simon G，et al. The flooding time synchronization protocol[C]//Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems. ACM，2004：39-49

[4] Ganeriwal S，KumarR，Srivastava M B. Timing-sync Protocol for Sensor Networks[C]//Proceedings of the 1st International Conference on Embedded Networked Sensor Systems. Los Angeles，

USA,2003:138-149

[5] Noh K, Serpedin E, Qaraqe K. A new approach for time synchronization in wireless sensor networks: Pairwise broadcast synchronization[J]. IEEE Transactions on Wireless Communications, 2008, 7(9): 3318-3322

[6] Kim H, Kim D, YooS. Cluster-based hierarchical time synchronization for multi-hop wireless sensor networks[C]// 20th International Conference on Advanced Information Networking and Applications, 2006(AINA2006). IEEE, 2006, 2: 5

[7] Hu A, Servetto S D. Asymptotically optimal time synchronization in dense sensor networks[C]// Proceedings of the 2nd ACM International Conference on Wireless Sensor Networks and Applications. ACM, 2003: 1-10

[8] 田俊峰,温怀湘,温玉. 一种新的建立在簇结构上的时间同步算法[J]. 小型微型计算机系统,2010(3):20
Tian Jun-feng, Wen Huai-xiang, Wen yu. A Novel Time Synchronization Algorithm Based on Synchronizer of Clustering Architecture [J]. Journal of Chinese Computer System, 2010(3): 20

[9] Handy M J, Haase M, Timmermann D. Low energy adaptive clustering hierarchy with deterministic cluster-head selection[C]// 4th International Workshop on Mobile and Wireless Communications Network, 2002. IEEE, 2002: 368-372

[10] Gautam G C, Sharma T P, Katiyar V, et al. Time synchronization protocol for wireless sensor networks using clustering[C]// 2011 International Conference on Recent Trends in Information Technology (ICRTIT). IEEE, 2011: 417-422

[11] Mamun Q. A qualitative comparison of different logical topologies for wireless sensor networks[J]. Sensors, 2012, 12(11): 14887-14913

[12] Heinzelman W R, Chandrakasan A, Balakrishnan H. Energy-efficient communication protocol for wireless microsensor networks[C]// Proceedings of the 33rd annual Hawaii international conference on System sciences, 2000. IEEE, 2000(2): 10

(上接第 183 页)

(1)模型变换前的变换规则校验:模型变换是需要符合一些既定的要求和前提的,如果每次用户进行变换操作时,提供给用户所有菜单供选择,那么势必增加许多无谓的操作,向用户显示很多不友好的错误提示,似乎在对用户进行抱怨,从可用性角度来看,这是非常不好的。因此,在界面设计时,给出当下适用的变换规则并隐藏那些并不适用的规则(此将按钮置灰),以此来减少误操作及引导用户选择正确的规则,如图 3 所示。

(2)加入一些动画:在不增加程序的复杂度和软件的负担的前提下,增加适当的动画效果能有效地提升软件的趣味性,使软件功能更为饱满,很直观地提高用户体验。我们的工具在某些细节和关键之处加入了动画效果,例如创建与删除图例时有平移、旋转、淡入浅出等动画效果。模型变换时,图例之间的连线的撤销与连接都伴随着渐变的动画效果。除此之外,对话框的弹出、参数区的显示与隐藏切换等细节之处也都加入了适当的动画。

此外,我们还采用初学者模式和熟练者模式的区分、手势缩放操作等手段来克服移动设备自身的局限性,以提高可用性和用户体验。

结束语 本文介绍了一个基于人机交互设计的运行于安卓平台的计算机辅助软件需求分析工具,它相较于早期开发的基于 PC 端的工具[10]具有更好的易用性和用户体验,加入了对因果关系的形象描述,重点突出了人机交互设计的重要性。从笔者个人开发此工具的经历来看,由于在工具设计的初期没有充分采用人机交互设计方式,而是在开发过程中将其不断融入,导致界面比较单调枯燥,部分功能的用户体验仍然有待提升。因此,在接下来的工作中,除了继续扩展工具的功能以期进一步实现问题框架从理论到实践的过渡,利用人机交互设计来对原有功能不断进行完善也显得尤为迫切和必要。

参 考 文 献

[1] Al-Subaie H S F, Maibaum T S E. Evaluating the Effectiveness of a Goal-Oriented Requirements Engineering Method[C]// 4th International Workshop on Comparative Evaluation in Requirements Engineering, 2006. IEEE Computer Society, 2006: 8-19

[2] Kanakaraddi S G, Naragund J G, Chikaraddi A K. Active learning methods for teaching OOAD course[C]// 2013 IEEE International Conference in MOOC Innovation and Technology in Education (MITE). IEEE, 2013: 47-52

[3] Li J, Li Y F, Qing X, et al. Interface generation technology based on Concur Task Tree[C]// 2010 International Conference on Information Networking and Automation (ICINA). IEEE, 2010

[4] Jackson M. Software requirements and specifications: a lexicon of principles, practices and prejudices [M]. Boston: Addison-Wesley, 1995

[5] Jackson M. Problem frames: analyzing and structuring software development problems [M]. Boston: Addison-Wesley, 2001

[6] 李智,金芝. 从用户需求到软件规约:一种问题变换的方法[J]. 软件学报,2013,24(5):961-976
Li Zhi, Jin Zhi. From user requirements to software Specifications: An approach based on problem Transformation[J]. Journal of Software, 2013, 24(5): 961-976

[7] Berry M D. Software requirements and design: the work of Michael Jackson[J]. ACM SIGSOFT Software Engineering Notes, 2011, 36(2): 39-40

[8] 骆斌,冯桂焕. 人机交互软件工程视角[M]. 北京:机械工业出版社,2012:3-6
Luo Bin, Feng Gui-huan. Human-Computer Interaction: A Software Engineering Perspective [M]. Beijing: China Machine Press, 2012: 3-6

[9] 骆斌,冯桂焕. 人机交互软件工程视角[M]. 北京:机械工业出版社,2012:46-49
Luo Bin, Feng Gui-huan. Human-Computer Interaction: A Software Engineering Perspective [M]. Beijing: China Machine Press, 2012: 46-49

[10] 刘国源,万光海,庞柳,等. 基于问题架的计算机辅助需求工程工具的研发[J]. 计算机科学,2014,41(11):137-168
Liu Guo-yuan, Wan Guang-hai, Pang Liu, et al. Research and Development of Computer-aided Requirements Engineering Tool Based on Problem Frames[J]. Computer Science, 2014, 41(11): 137-168