DOI: 10.19650/j.cnki.cjsi.J2108423

无线传感器网络中延迟补偿的分布式时钟同步算法

方子涵[1],李 旦[1,2],蒋 鹏[1],刘彤宇[1],陆起涌[1,2]
(1. 复旦大学电子工程系 上海 200433; 2. 复旦大学义乌研究院 义乌 322001)

摘 要:时钟同步为无线传感器网络中的节点处理分布式任务提供了一个共同的时间参考,无线传感器网络需要精确的时钟同步来保持数据的一致性和协调性,但是同步过程中的不确定性延迟会严重干扰同步参数的计算。为了消除消息延迟的不确定性对时钟同步的影响,本文考虑高斯分布的消息延迟,建立了新的时钟同步模型,并基于卡尔曼滤波和分布式时钟同步算法,提出了一种能够有效处理延迟影响的时钟同步算法,该算法使用广播的单向通信方式,所有参数可以在接收端独立更新,另外考虑了以计算的全局时间为参考更新偏差来提升同步精度。使用 5 个节点进行 MATLAB 仿真实验和 nRF52832 硬件验证,在微秒级延迟下,本算法的同步精度可以维持在 10 μs,相较于其他算法,本算法维持了很好的同步精度和稳定性。

关键词: 分布式时钟同步;测量延迟;卡尔曼滤波;无线传感器网络
中图分类号: TP393 TH701 **文献标识码**: A **国家标准学科分类代码**: 510.99

Distributed clock synchronization for Kalman based delay estimation in wireless sensor networks

Fang Zihan[1], Li Dan[1,2], Jiang Peng[1], Liu Tongyu[1], Lu Qiyong[1,2]
(*1. Department of Electronic Engineering, Fudan University, Shanghai 200433, China;*
2. Yiwu Research Institute, Fudan University, Yiwu 322001, China)

Abstract: Clock synchronization provides a common time reference for different nodes in the wireless sensor networks to deal with distributed tasks. The wireless sensor networks require accurate clock synchronization to keep the consistence and coordination of data among nodes. However, the unpredictability of message delays in the synchronization process may affect the synchronization accuracy significantly. To eliminate the impact of delay fluctuation, the clock synchronization model is formulated, which considers the influence of delays with Gaussian distribution in the synchronization process. And a Kalman based delay estimation algorithm on distributed clock synchronization is proposed, which can effectively deal with the influence of delays. This algorithm utilizes one-way message exchange mechanism, and all parameters could be updated independently in one node. In addition, the calculated global time is considered as a reference to update the clock offset. In this way, the synchronization accuracy is improved. The MATLAB simulation and the nRF52832 experiment testbed with 5 nodes indicate that this algorithm restricts the synchronization error into 10 μs under microsecond delays. Compared with other algorithms, this method could achieve better synchronization accuracy and performance.

Keywords: distributed clock synchronization; delay measurement; Kalman filter; wireless sensor network

0 引 言

无线传感器网络(wireless sensor network, WSN)由大量分布在特定区域的传感器节点组成,由于其低成本、无线传输和自配置的特点而被广泛用于数据监控、目标跟踪和环境监测等场景[1-2]。这些应用场景通常需要节点间协同工作[3],例如网络中各节点任务的有序执行、距离估计、协议运行、协同休眠等[4],这就要求所有节点的时钟保持同步,即所有节点具有一致的全局时间。因此时钟同步一直是无线传感器网络中一个重要且具有挑战的问题。

在无线传感器网络中,时钟同步算法通常分为两类,一类是基于参考节点的时钟同步算法,另一类是分布式时钟同步算法。RBS (reference-broadcast synchronization)[5]算法是最早提出的基于参考节点的时钟同步

收稿日期:2021-08-16 Received Date: 2021-08-16

算法,该算法使用参考节点进行周期性广播,使单跳范围内节点接收参考信息后与周围节点交换记录的时钟戳,从而完成节点间的同步。TPSN(timing-sync protocol for sensor networks)[6]使用分层树结构将整个网络进行划分,每个节点与上一层参考节点进行周期性同步。FTSP(flooding time synchronization protoco1)[7]对同步精度进行了很大的优化,它采用动态选择参考节点的方式,每个节点对本地时钟的漂移和偏差进行线性回归估计来维持高精度并减少通信开销。但是无论 RBS、TPSN 还是 FTSP,这些通过参考节点完成的同步,都没有考虑参考节点和树状结构导致的误差累积,另外,如果参考节点出现故障,整个网络将花费很长时间重新完成同步[8],可能会极大影响网络性能。

因此目前广泛应用的是分布式时钟同步算法,无需参考节点,只需网络中所有节点维持一个全局的虚拟时钟,就可以获得更精确的同步时钟,提高了网络的鲁棒性和可扩展性[9]。分布式时钟同步中经典的方法是 ATS(average timesynch)算法[10],它采用两次平均的分布式时钟同步算法,即每个节点根据自己与发送节点的时钟差调整偏差值,使得所有节点的时钟收敛到一个全局的虚拟时钟,但是 ATS 算法没有考虑节点与节点之间通信延迟所产生的影响。文献[11]提出的 RoATS(robust average timesynch)算法对 ATS 算法进行了优化,可以通过截断和容错,在使用配对通信的情况下抵抗延迟影响,但是其考虑的是双向时钟同步,且漂移率的更新无法在单个节点独立进行。双向时钟同步可以准确估计无线传输延迟,但节点能耗较大[12],本文使用广播的单向通信方式,可以极大降低能量消耗[12]。文献[13]推导出时钟偏差和漂移率的一般模型,并应用卡尔曼滤波器捕捉其时变行为,从而满足了所需的误差范围,但仍然需要发送节点作为参考,网络中不同区域的同步精度可能相差很多。文献[14]中每个节点基于卡尔曼滤波器,使用其他节点的状态估计来预测全局时钟,并使用投票算法来调整自己对于虚拟全局时钟的估计,但文献[14]仅考虑了传输延迟的影响。

时钟同步中的消息延迟由 5 部分组成:1)发送处理延迟;2)媒体访问延迟;3)传输延迟;4)无线电传播延迟;5)接收处理延迟[15]。这些消息延迟会使得发送端传输的同步数据包不能及时被接收端处理,这将导致发送和接收时间戳的不对应,会很大程度上干扰漂移率和偏差的估计[9,11],而无线传感要求长时间维持同步并具有动态鲁棒性,延迟影响将使同步过程变得很困难。已有的分布式时钟算法虽然考虑了消息延迟,但采用了较为理想的假设,通常假设消息延迟是确定不变的。在实际中,发送处理延迟、接收处理延迟和媒体访问延迟是最不确定的且很难进行估计[5],这极大地增加了延迟对同步精度的影响,且不确定延迟会比同步精度大几个数量级[3]。文献[7]中详细介绍了时钟同步过程中可能出现的延迟及影响因素,由于系统调用开销和操作系统负载,发送处理延迟和接收处理延迟可能高达 100 ms,而媒体访问延迟表示数据传输开始的到真正从信道发出的等待时间,根据当前信道状态,通常在 10~500 ms 波动。但是由于消息延迟可以看成独立的随机过程,因此根据中心极限法则,消息延迟假设为高斯分布[5,16]。

由于时钟模型中消息延迟的不确定性以及时钟参数的时变性,本文考虑了时钟同步过程中的高斯分布延迟,提出了新的时钟模型,将卡尔曼滤波应用到同步过程中来跟踪网络的动态参数变化,并充分利用其去噪特性来估计延迟的真实值,设计了一种估计时钟动态参数的时钟同步算法(Kalman based delay estimation algorithm on distributed clock synchronization, KBDDCS)。其次本文考虑节点能量和网络数据通信量的限制,采用广播的单向通信方式,使节点自适应接收间隔的变化,且所有参数可以在接收端独立更新。最后本文采用计算的全局时间作为参考更新偏差,提高了同步精度。理论和实验表明,在考虑延迟干扰的情况下本算法能够维持很好的同步精度。

1　理论分析

1.1　传统的分布式时钟同步模型

在传统的分布式时钟同步模型中,对于每个节点 i,其时钟数据是通过对已知频率的晶振产生的脉冲进行计数来获得的,相对于真实时间,该时钟数据可以表示为一个简单的时间变化函数,即节点认为的本地时间 $C_i(t)$ 可以建模为真实时间 t 的线性函数如式(1)所示。

$$C_i(t) = \alpha_i t + \beta_i \tag{1}$$

其中,α_i 为硬件时钟漂移率,表示在给定时间内时钟会变快或者变慢多少个计数值,对于理想的时钟,硬件时钟漂移率 $\alpha_i = 1$;β_i 表示偏差,代表了上电时本地时间与真实时间的初始偏差。

对于节点 i,其真实时间 t 是无法获知的,因此无法计算 α_i 和 β_i 的值。然而本地时间 $C_i(t)$ 可以表示成相对另一个时钟 $C_j(t)$ 的线性关系,即:

$$C_i(t) = \frac{\alpha_i}{\alpha_j} \cdot C_j(t) + \left(\beta_i - \frac{\alpha_i}{\alpha_j}\beta_j\right) = \alpha_{ij} \cdot C_i(t) + \beta_{ij} \tag{2}$$

其中,α_{ij} 表示相对漂移率;β_{ij} 表示相对偏差。

因此可以设置一个虚拟参考时钟 $C_v(t)$,即:

$$C_v(t) = a_v t + b_v \tag{3}$$

其中,a_v 表示虚拟时钟的漂移率,b_v 表示虚拟时钟的偏差。每个节点都可以通过比较自己的本地时钟与虚拟

参考时钟的差距估计相对参数 α_{iv} 和 β_{iv}，使得 $\alpha_{iv} \cdot C_i(t) + \beta_{iv}$ 的估计与虚拟时钟 $C_v(t)$ 相同。使用逻辑时钟 $L_i(t)$ 来表示使用本地时钟估计的虚拟参考时钟，$1/\hat{a}_i(t)$ 表示估计的相对漂移率 α_{iv}，$\hat{b}_i(t)$ 表示估计的相对偏差 β_{iv}。由此可得传统的分布式时钟同步算法[9-11]中，采用的时钟同步模型为：

$$L_i(t) = \frac{1}{\hat{a}_i(t)} C_i(t) + \hat{b}_i(t) \tag{4}$$

1.2 估计延迟的时钟同步模型

消息延迟在无线传输中是不可避免的，这将导致实际接收时间相对于发送时间延迟了 $d_i(t)$，即接收时间为 $C_i(t + d_i(t))$ 而发送时间为 $C_j(t)$，对于接收节点来说，相当于其本地时间与发送节点的本地时间的差值增加了 $d_i(t)$，但发送节点和其他未收到数据的节点的时钟则还是正常运行的，此时使用式(4) 无法进行接收时间的准确描述。使用逻辑时钟 $C_i(t + d_i(t))$ 进行同步将会给漂移率估计带来很大影响，导致各节点无法完成同步，证明如下。

根据式(1)可以得到：

$$t = \frac{C_i(t) - \beta_i}{\alpha_i} = \frac{C_j(t) - \beta_j}{\alpha_j} \tag{5}$$

假设 T_i 表示节点 i 的同步周期，将式(5) 进行变形可以得到：

$$\alpha_{ij} = \frac{\alpha_i}{\alpha_j} = \frac{C_i(t) - C_j(t - T_i)}{C_j(t) - C_j(t - T_j)} \tag{6}$$

假设 $interval_i = C_i(t + d_i(t)) - C_i(t - T_i)$，如果没有消除接收时间 $C_i(t + d_i(t))$ 的延迟影响，则相对漂移率 α_{ij} 的计算为：

$$a_{ij} = \frac{interval_i}{C_j(t) - C_j(t - T_j)} = \frac{C_i(t) + C_i(d_i(t)) - C_i(t - T_i)}{C_j(t) - C_j(t - T^j)} = \frac{\alpha_i \Delta t + C_i(d_i(t))}{\alpha_j \Delta t} \tag{7}$$

由于 $C_i(d_i(t)) > 0$，因此计算的 a_{ij} 比其真实值要大，另外，由于 $d_i(t)$ 在每次同步时很可能发生变化，所以 Δt 无法预先通过历史信息获知，从而消除延迟影响。文献[9,11] 证明了相对漂移率计算错误对时钟同步收敛性的影响，在本文中，考虑式(18) 推导出的时钟估计迭代式和式(25) 中卡尔曼滤波的状态更新步骤，不正确的 α_{ij} 将导致卡尔曼滤波的过程中估计的 $\hat{d}_i(t)$ 比真实值要大，则更新接收时间后的 $C_i^+(t)$ 将比真实值要小，而这会使得下一次同步时 $interval_i$ 增大，在下一次同步的计算中导致 α_{ij} 继续增大，在后续同步过程中，此影响将逐步累加，导致整个网络无法收敛。

为了解决上述问题，需要在每次同步后及时消除接收端消息延迟的不确定性给同步过程带来的影响，更新本次接收时间的记录值，将其表示为 $C_i^+(t)$，即：

$$C_i^+(t + d_i(t)) = C_i(t + d_i(t)) - \hat{d}_i(t + d_i(t)) \tag{8}$$

使用 $\hat{d}_i(t)$ 表示估计的延迟影响，$C_i^+(t)$ 表示修正延迟影响后的本地时钟值，$\hat{b}_i^+(t)$ 表示补偿后的时钟偏差，即：

$$\begin{aligned} C_i^+(t) &= C_i(t) - \hat{d}_i(t) \\ \hat{b}_i^+(t) &= \hat{b}_i(t) + \frac{\hat{d}_i(t)}{\hat{a}_i(t)} \end{aligned} \tag{9}$$

修正后的逻辑时钟为：

$$L^i(t) = \frac{1}{\hat{a}_i(t)} \cdot C_i^+(t) + \hat{b}_i^+(t) \tag{10}$$

式(10)即为本文提出的新模型。对于各个节点，其硬件时钟参数 α_i 和 β_i 是无法获知的，所以同步的目标是，每个节点通过接收相邻节点发送的信息，对 $\hat{a}_i(t)$ 和 $\hat{b}_i(t)$ 进行调整，从而最终使得补偿后的逻辑时钟在同一时刻 t 均收敛到同一虚拟时钟值 $C_v(t)$，即：

$$\forall i \in N, \lim_{t \to \infty} \frac{\alpha_i}{\hat{a}_i(t)} = a_v \tag{11}$$

$$\forall i \in N, \lim_{t \to \infty} \frac{\beta_i}{\hat{a}_i(t)} + \hat{b}_i^+(t) = b_v \tag{12}$$

$$\forall i \in N, L_i(t) = C_v(t) = a_v t + b_v \tag{13}$$

2 同步算法

在上述时钟同步模型中，由于虚拟时钟 $C_v(t)$ 并非真实存在，而是网络内节点协商的结果，所以此时钟的漂移率和偏差并非定值且无法事先计算，虚拟时钟参数的不确定性对同步算法提出了更高的要求。另外，由于消息延迟存在很大的不确定性，如何在接收端动态分离出正确的时间进行参数计算也是一个很大的挑战。由文献[5,16]可知，延迟 $d_i(t)$ 服从高斯分布，为了跟踪时钟参数的变化，本文选择卡尔曼滤波进行状态估计，主要包括对延迟 $d_i(t)$、相对漂移率 $\hat{a}_i(t)$ 和相对偏差 $\hat{b}_i(t)$ 的估计。

分布式时钟同步通过节点间不断的交换时钟数据和当前估计的参数来实现，为了使最终所有节点的逻辑时钟收敛到一致的全局时钟，各节点需要定期进行时钟参数的更新，本算法采用时分复用系统，使用时隙对一个同步周期进行划分，每个节点在一个周期中的特定时隙发送数据，其他时隙则处于接收状态，在收到同步数据时立刻记录自己的本地时间，同步过程仅在接收到数据后进行。

2.1 状态方程

本文所提出算法的主要思想是基于各节点的周期性的收发信息交换，结合发送数据和接收数据，使用动态加权平均进行时钟参数的估计。网络内的节点均保持自己

对虚拟时钟的参数估计,当接收到其他节点广播的时钟数据后,使用卡尔曼滤波迭代更新,以维持与虚拟时钟的同步性。

卡尔曼滤波使用递归方式,分为预测和更新两个步骤,只需知道当前测量值和上一次迭代的时间估计就可以估计当前状态。但是由于采用分布式和广播的单向通信方式,节点的接收迭代间隔不是固定的,无法预先设置,因此需要设计一种能够自适应接收间隔的迭代方式,使得卡尔曼滤波能够及时与观测数据对应,以维持高精度的参数估计。为了完成预测步骤,需要知道当前时间相对于上一次迭代(即上一次接收)的时间间隔,因此发送节点需要发送自己上一次迭代的接收校正值 C_j_last,以便计算时间间隔。

假设在时间 t 时节点 j 发送其本地时间 $C_j(t)$,C_j_last 代表节点 j 在之前的同步校正后的接收时间,则 $C_j(t) - C_j\_last$ 表示节点 j 两次同步的时间间隔。节点 j 在接收到数据时立刻记录它的接收时间 $C_i(t + d_i(t))$,并且在更新参数后存储 $C_i\_last = C_i^+(t + d_i(t))$。根据式(10)、(13)可以得到:

$$C_i^+(t) = \hat{a}_i(t) \cdot (C_v(t) - \hat{b}_i^+(t)) \tag{14}$$

将式(12)代入式(14)可得节点 i 校正后的本地时间为:

$$C_i^+(t) = \hat{a}_i C_v(t) + (\beta_i - \hat{a}_i b_v) \tag{15}$$

由于节点仅在接收到数据时才会进行同步过程,因此 $C_j^+(t) = C_j(t)$ 且 $C_i\_last = C_i^+(t - T_i^j)$,其中 T_i^j 表示节点 i 记录的相邻两次同步过程的本地接收时间之差,因此根据式(1)、(15) 可以得到:

$$\frac{C_i^+(t) - C_i\_last}{\hat{a}_i(t)} = C_v(t) - C_v\_last = \frac{C_j^+(t) - C_j\_last}{\hat{a}_j(t)} \tag{16}$$

根据式(11)调整式(16)可以得到:

$$\frac{C_i^+(t) - C_i^+(t - T_i^j)}{C_j(t) - C_j\_last} = \frac{\hat{a}_i(t)}{\hat{a}_j(t)} = \frac{\alpha_i}{\alpha_j} \tag{17}$$

对式(17)进行变换即可得到节点 i 校正后接收时间的迭代表达式为:

$$C_i^+(t) = C_i^+(t - T_i^j) + \alpha_i \left[\frac{C_j(t) - C_j\_last}{\alpha_j} \right] \tag{18}$$

本文考虑延迟的波动,为了准确地跟踪延迟变化,可以把式(18)写成如下形式:

$$C_i^+(t) = C_i^+(t - T_i^j) + \hat{d}_i(t - T_i^j) + \hat{a}_i(t - T_i^j) \left[\frac{C_j(t) - C_j\_last}{\hat{a}_j(t - T_j^*)} - \frac{\hat{d}_i(t - T_i^j)}{\hat{a}_i(t - T_i^j)} \right] \tag{19}$$

由于延迟只对接收节点有影响,本文假设每个接收端的延迟服从均值和方差相同的高斯分布。由于消息延迟是一个具有马尔科夫链性质的随机过程,而本文希望估计延迟的真实值,使用随机漫步模型可以描述延迟的独立性和随机性。由于卡尔曼滤波可以在高噪声环境中获得很准确的状态估计,因此虽然无法准确描述延迟的变化,但设置一个合适的波动范围,可以利用观测去修正由于预测不准导致的误差,从而找到延迟的真实值,即:

$$\hat{d}_i(t) = \hat{d}_i(t - T_i^j) + w_d(t) \tag{20}$$

其中,$w_d(t)$ 描述了延迟的波动范围,服从均值为 0,方差为 σ_d^2 的高斯分布。

根据式(3)、(11)所示,漂移率的估计趋向于定值,可以采用随机过程来描述漂移率的状态估计,其中 $w_a(t)$ 建议取一个比漂移率最大波动范围小很多的数值。

$$\hat{a}_i(t) = \hat{a}_i(t - T_i^j) + w_a(t) \tag{21}$$

由于节点只有接收到数据时才能进行迭代更新,因此在发送时隙只是维持参数的估计值,所以对于接收节点来说,如果此节点在上一次同步中为发送节点,即 $i = last\_send$,则节点 i 的迭代状态值停留在两次同步前。根据式(18),为了能够将状态值正确迭代到当前时刻,节点 i 需要发送节点 j 的上一次迭代的本地时钟校正值 C_j_last 也是两次同步之前的,即 $C_j\_last = C_j^+(t - T_j^i - T_j^k)$;否则应为上一次同步接收到另一节点 k 发送数据时的本地时钟值,即 $C_j\_last = C_j^+(t - T_j^k)$。为了与当前接收节点的状态值对应,发送节点 j 的上一次迭代的本地时钟值表示为:

$$C_j\_last = \begin{cases} C_j^+(t - T_j^k), & k \neq i, i \neq last\_send \\ C_j^+(t - T_j^i - T_j^k), & k \neq i, i = last\_send \end{cases} \tag{22}$$

本算法选择漂移率估计 $\hat{a}_i(t)$、校正后的本地时钟估计 $C_i^+(t)$、延迟估计 $\hat{d}_i(t)$ 作为状态参数,根据上面的描述,卡尔曼滤波的状态方程表示为:

$$\boldsymbol{X}_i(t) = \begin{bmatrix} \hat{a}_i(t) \\ \hat{C}_i(t) \\ \hat{d}_i(t) \end{bmatrix} = \begin{bmatrix} w_a(t) & 0 & 0 \\ 0 & w_C(t) & 0 \\ 0 & 0 & w_d(t) \end{bmatrix} + \begin{bmatrix} 1 & 0 & 0 \\ A_{21}(t) & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \hat{a}_i(t - T_i^j) \\ \hat{C}_i(t - T_i^j) \\ \hat{d}_i(t - T_i^j) \end{bmatrix} \tag{23}$$

$$A_{21}(t) = \frac{C_j(t) - C_j\_last}{\hat{a}_j(t - T_j^*)} - \frac{\hat{d}_i(t - T_i^j)}{\hat{a}_i(t - T_i^j)} \tag{24}$$

2.2　观测方程

观测方程作为实际观察到的测量值,用于对通过状态方程计算得到的估计值进行修正,使得修正后的状态估计更接近真实值。在可以直接观测到的发送节点的发送时间 $C_j(t)$ 和状态估计 $\boldsymbol{X}_j(t - T_j^*)$,以及接收节点的接收时间 $C_i(t + d_i(t))$ 和状态估计 $\boldsymbol{X}_i(t - T_j^*)$ 中,相对于

状态方程的3个变量,只有接收时间 $C_i(t+d_i(t))$ 可以作为观测量来对状态进行修正。但仅一个观测量对状态的修正不足以达到预期效果,而由于虚拟时钟的漂移率并非真实存在的因此无法直接观测到,所以本文考虑通过计算获得相对漂移率的观测量。

本文假设在短时间内,节点的硬件漂移率 α_i 为定值,由于每个节点的真实漂移率不一定相同,而考虑到节点广播的周期性以及相对漂移率定义 $a_{ij}=\alpha_i/\alpha_j$,结合式(17)可以写出根据发送节点参数计算的本节点的漂移率估计 $a_{oi}(t)$,式(26)可作为观测值对状态进行修正。

$$\hat{a}_{ij}=\frac{C_i(t+d_i(t))-\hat{d}_i(t-T_i^j)-C_i^+(t-T_i)}{C_j(t)-C_j^+(t-T_j)} \tag{25}$$

$$a_{oi}(t)=\hat{a}_{ij}\hat{a}_j(t-T_j^*) \tag{26}$$

由于漂移率估计和延迟估计是同步进行的,因此 $C_i(t+d_i(t))$ 是包含延迟的接收时间,而真实的接收时间近似等于 $C_i(t+d_i(t))-\hat{d}_i(t-T_i^j)$。由于假设接收端的延迟服从同一高斯分布,因此虽然 $\hat{d}_i(t-T_i^j)$ 表示上一次估计的延迟,此估计仍然可以对不同节点使用而不会导致很大的偏差,由于环境、仪器精度等在观测值产生的偏差能在卡尔曼滤波中被很好的抑制。

选择节点接收数据时记录的本地时钟 $C_i(t+d_i(t))$ 和根据发送节点计算的漂移率估计值 $a_{oi}(t)$ 作为观测,则观测方程可以写为:

$$\boldsymbol{Y}_i(t)=\begin{bmatrix}a_{oi}(t)\\C_i(t+d_i(t))\end{bmatrix}=\begin{bmatrix}1&0&0\\0&1&1\end{bmatrix}\cdot\begin{bmatrix}\hat{a}_i(t)\\\hat{C}_i(t)\\\hat{d}_i(t)\end{bmatrix}+\begin{bmatrix}r_a(t)&0\\0&r_C(t)\end{bmatrix} \tag{27}$$

2.3 卡尔曼滤波

为了消除延迟的不确定性并跟踪时钟参数的动态变化,本文推导了式(23)的状态方程和式(27)的观测方程,使用卡尔曼滤波来估计相对漂移率、校正后的本地时钟和变化的消息延迟,因此可应用式(28)~(32)卡尔曼滤波的迭代过程来预测和更新系统状态,使得同步过程逐步收敛。

状态预测:$\boldsymbol{X}_i^-(t)=\boldsymbol{A}_i(t)\boldsymbol{X}_i(t-T_i^j)$ (28)

最小预测 MSE:$\boldsymbol{P}_i(t)=\boldsymbol{A}_i(t)\boldsymbol{M}_i(t-T_i^j)\boldsymbol{A}_i(t)^{\mathrm{T}}+\boldsymbol{W}_i$ (29)

卡尔曼增益:$\boldsymbol{K}_i(t)=\boldsymbol{P}_i(t)\boldsymbol{H}_i^{\mathrm{T}}(\boldsymbol{V}_i+\boldsymbol{H}_i\boldsymbol{P}_i(t)\boldsymbol{H}_i^{\mathrm{T}})^{-1}$ (30)

状态更新:$\boldsymbol{X}_i(t)=\boldsymbol{X}_i^-(t)+\boldsymbol{K}_i(t)(\boldsymbol{Y}_i(t)-\boldsymbol{H}_i\boldsymbol{X}_i^-(t))$ (31)

最小 MSE:$\boldsymbol{M}_i(t)=(\boldsymbol{I}-\boldsymbol{K}_i(t)\boldsymbol{H}_i)\boldsymbol{P}_i(t)$ (32)

通过之前的分析,进行卡尔曼滤波状态估计后,各接收节点的整体漂移率 $\alpha_i/\hat{a}_i(t)$ 已经处于几乎相同的数值,此时还需要对相对偏差 $\hat{b}_i(t)$ 进行调整,使得补偿偏差后节点间的逻辑时钟与虚拟时钟一致。

之前的算法大多采用发送节点的逻辑时钟来计算本节点需要补偿的偏差,这会使得同步过程中由于发送节点的变化带来过度调整,引起漂移率和同步误差的波动。为了加快收敛速度并保持参数估计的稳定性,减少参数估计过度调整导致的同步误差的增大,本文考虑以计算的全局时间为参考更新偏差,全局时间可以计算为发送节点和接收节点的逻辑时钟的加权平均值。

$$G_i(t)=\frac{w_i(t)L_i(t)+w_j(t)L_j(t)}{w_i(t)+w_j(t)} \tag{33}$$

选择将节点接收的数据包数量作为权重 $w_i(t)$,以此降低新加入节点对整个网络的影响,并使由于丢包导致的长期未接收数据的节点能够快速与网络其他节点取得同步。结合式(10)、(33)可以推出本节点偏差的估计为:

$$\hat{b}_i^+(t)=G_i(t)-\frac{1}{\hat{a}_i(t)}\cdot[C_i(t+d_i(t))-\hat{d}_i(t)] \tag{34}$$

本算法实现的具体步骤流程如下:

1)由于同步需要定期进行,因此可以假设每个节点按照自己的计数周期 T 进行周期性的广播,每个节点在自己的计数周期的特定时刻(如 $\frac{C_i(t)}{T}\in N^+$)发送数据包。

2)在节点 i 上电后,初始化卡尔曼滤波矩阵,设置 $\hat{a}_i(0)=1$,$\hat{C}_i(0)=C_i(0)$,$\hat{d}_i(0)=0$,随后判断本时隙是否为发送时隙。

3)处于发送时隙则发送包含序列号 ID_i 的数据包 $<ID_i,\hat{a}_i(t),\hat{b}_i(t),C_i(t),C_i^+(t-T_i^k),C_i^+(t-T_i^j-T_j^k)>$,并设置 $pkt\_send\_in\_last=1$。

4)如果处于接收状态则等待其他节点的数据包到来,并在接收数据后立即记录自己的接收时钟戳 $C_i(t)$。

5)如果节点 i 在上一次同步时是发送节点,即当节点 i 接收数据时 $pkt\_send\_in\_last=1$,则发送节点 j 的 $C_j\_last=C_j(t-T_j^i-T_j^k)$;否则 $C_j\_last=C_j(t-T_j^k)$。

6)如果初次收到节点 j 的数据,则设置 $\hat{a}_i(t-T_i^j)=1$,$\hat{C}_i(t-T_i^j)=C_i(t+d_i(t))$,$\hat{d}_i(t-T_i^j)=0$。

7)如果节点有对 $C_i^+(t-T_i)$ 的历史记录,则根据式(25)、(26)计算观测量 $a_{oi}(t)$,并进行后续迭代;否则设置 $C_i^+(t-T_i)=C_i(t)$,跳到步骤12)。

8)由于两次通信间隔的时隙数目是不确定的,因此状态方程参数需要实时更新,即在每次进行卡尔曼滤波

之前都需要通过式(24)计算 $A_{21}(t)$。

9)使用式(28)~(32)迭代更新卡尔曼滤波的状态量。

10)使用式(33)计算当前的全局时间 $G_i(t)$,并使用式(34) 估计时钟偏差 $\hat{b}_i^+(t)$。

11)更新接收时间 $C_i^+(t) = C_i(t) - \hat{d}_i(t)$,设置 $C_i\_last = C_i^+(t)$。

12)保存当前参数 $\hat{a}_i(t)$、$\hat{d}_i(t)$、$\hat{b}_i(t)$、$C_i^+(t)$,设置 $pkt\_send\_in\_last = 0$。

3　实验验证

为了展示算法的正确性和有效性,本文利用 1 000 次迭代的蒙特卡洛 MATLAB 仿真分析了算法在不同延迟情况下的效果。另外,本文将算法在真实的硬件实验平台上进行测试并与其他经典算法进行对比,实验系统包括 5 个 nRF52832[17] 蓝牙节点和一个监控节点,没有预定的参考节点。5 个节点使用相同的程序进行周期性发送并接收数据,采用伪周期的方式在自己的特定时隙发送当前时间以及其他时钟参数,其他节点在接收到数据后记录自己的当前时间并使用算法进行时钟参数的更新。使用同步误差来表征同步效果,err 表示各节点在同一时刻估计的全局时钟的差距。

$$err = \max_{i,j}(L_i(t) - L_j(t)) \tag{35}$$

3.1　仿真实验

假设延迟均服从均值 100 μs,标准差 33 μs 的高斯分布,即延迟有 99.97% 的置信度分布在[0,200] μs 内。为了展示算法效果,根据硬件产品说明书[17],设置各节点的硬件漂移率波动为 100×10^{-6},即 a_i 在[0.999 9,1.000 1]之间随机选择。设置同步周期 $T_i = 5$ s,硬件计数器频率为 16 MHz,因此 1 μs = 16 ticks,节点在 5 s 内的计数值大约为 8×10^7,因此设置各节点的初始偏移量在[0,80 000 000]之间随机选择。另外设置迭代参数的初始值 $\hat{a}_i(0) = 1$、$\hat{C}_i(0) = C_i(0)$、$\hat{d}_i(0) = 0$、$\hat{b}_i(0) = 0$,仿照硬件运行方式进行伪周期发送和接收,对各节点的行为进行分析。

图 1 所示为同步过程的收敛趋势,可以看出使用计算的全局时钟作为参考更新偏差比之前算法中使用发送节点作为参考更新偏差的同步误差和波动更小,由于降低了节点过度调整带来的误差,同步的稳定性有所提高。另外,所提出的算法能够使得延迟影响被大大削弱,漂移率和偏差能够被准确估计,使得同步过程在有延迟的情况下依然能够正常进行而不会显著损失精度。当消息延迟服从均值 100 μs,标准差 33 μs 的高斯分布时,同步精度可以维持在 10 μs。

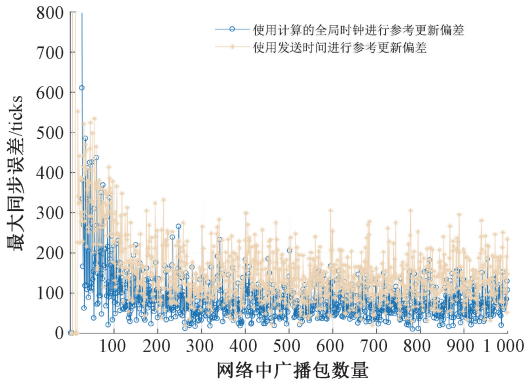


图 1　仿真实现中 5 个节点的同步误差

Fig. 1　Synchronization error among 5 nodes in simulation

维持延迟分布不变,图 2 所示为节点数量与收敛后平均同步误差的关系,由于分布式算法收敛后时钟参数的估计仍会有微小波动,因此节点增多可能导致过度调整,会稍微增大同步误差。

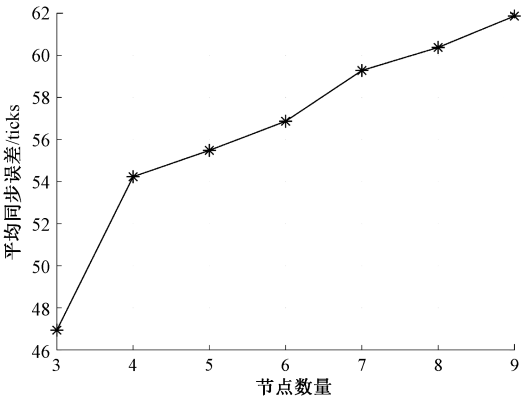


图 2　节点数量和同步误差的关系

Fig. 2　Relationship between numbers of nodes and synchronization error

3.2　不同延迟效果

本文使用 1 000 次蒙特卡洛仿真,展示了延迟波动的标准差 σ_d 与收敛后平均同步误差的关系,如图 3 所示。虽然随着延迟波动的增大,同步误差相应增大,但是同步效果并未过度受到延迟影响,由于卡尔曼滤波同时估计了本地时间和延迟波动,因此更好地保证了同步精度。对于微秒级延迟波动,即 σ_d<1 ms 时同步误差可以维持在 10 μs。

3.3　硬件测试

nRF52832 蓝牙芯片是由 Nordic 公司发布的 SoC 芯片,拥有 ARM Cortex-M4 32 位处理器,运行频率为 64 MHz,芯片内部有 5 个 32 位计数器,可以产生最大频

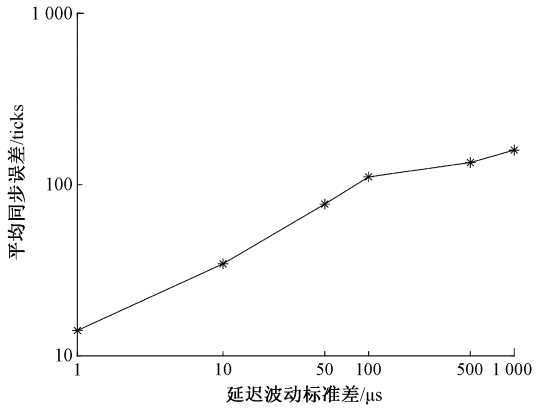


图 3 不同延迟分布的同步误差对比

Fig. 3 Comparison of synchronization error with different delay distributions

率 16 MHz 的外设时钟，芯片内置 2.4 GHz IEEE 802.15.4 无线电模块，可以运行在 2 Mbit/s 工作模式[17]。

利用 nRF52832 蓝牙芯片，搭建了实验验证平台如图 4 所示。实验平台由 5 个同步节点和 1 个监控节点组成，5 个节点使用相同的程序完成网络内的时钟同步，监控节点不参与时钟同步过程，仅用于获得其他节点的数据，并通过 USB 接口将数据传输到电脑进行显示。

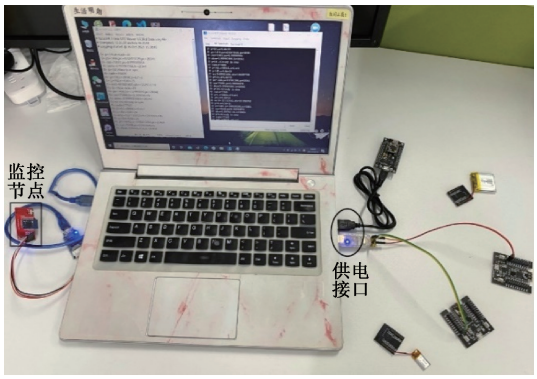


图 4 硬件实物图

Fig. 4 The hardware platform

硬件实验中设置各节点同步周期 $T_i = 5$ s，单个时隙 $T_s = 25$ ms，采用 2 Mbit/s 蓝牙低功耗模式。为了延长同步时间间隔，使用 1 个计数器和 1 个定时器级联作为节点的时钟计数模块，计数模块的时钟频率为 16 MHz，即 1 tick = 1/16 MHz = 62.5 ns。5 个节点在不同时刻随机启动，每个节点在发送前和接收后，记录当前本地时间用于同步过程的计算。

为了计算同步误差 err，设置了一个监控节点用于收集同一时刻其他 5 个节点的逻辑时钟值，5 个节点均处于监控节点通信范围内，接收到监控节点的询问后记录当前接收时间，并根据当前的漂移率估计 $\hat{a}_i(t)$ 和偏差估计 $\hat{b}_i(t)$ 计算当前的逻辑时钟值 $L_i(t)$，在下一次发送时将 $L_i(t)$ 一并广播发出，监控节点没有参与 5 个节点的同步过程。监控节点每 5 s 向 5 个节点询问当前逻辑时间，并通过 USB 连接到 PC 端，将各节点返回的逻辑时钟值以及其他时钟参数输出打印，本文通过监控节点收集到的数据进行分析。

图 5 所示为硬件中的测试结果，由于碰撞，在实际中测得了大约 5% 的丢包率，这在一定程度上影响了同步误差。对于丢包的影响将在后续文章中进行讨论。对比图 5 和 1，由于对消息延迟进行了估计和跟踪，因此网络内时钟可以有效收敛并维持全局时钟的统一，收敛后同步精度能够一直维持在预期水平，展示了算法很好的有效性。在延迟约为 100 μs 的情况下，同步精度可以维持在 10 μs。

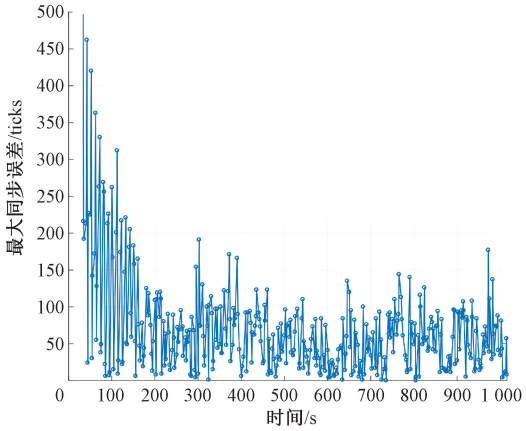
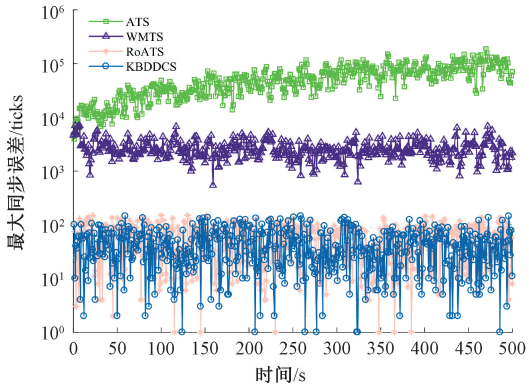


图 5 硬件实现中 5 个节点的同步误差

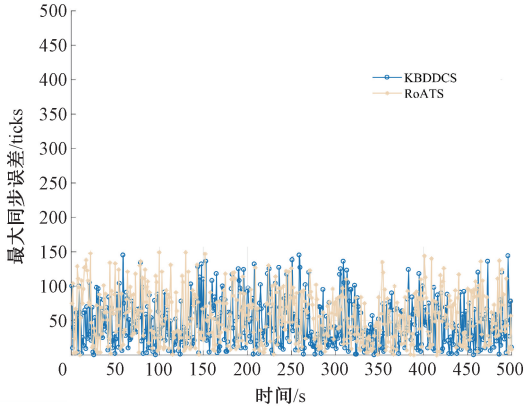
Fig. 5 Synchronization error of 5 nodes in hardware implement

3.4 不同算法比较

本算法与分布式网络中 WMTS(weighted maximum time synchronization)算法[9]、ATS 算法[10]和 RoATS 算法[11]进行了比较，图 6 所示为的同步收敛过程。ATS 由于没有考虑延迟影响，因此同步误差持续增大，WMTS 考虑了延迟影响，但由于缺少对延迟均值的先验知识，虽然漂移率可以估计的很准，但同步误差依然很大。本算法的同步精度与 RoATS 大致相同，但 RoATS 的漂移率补偿不能独立进行，需要收发双方成对通信来维持同步，即需要 3 个数据包才可以完成一个接收节点的完整同步更新过程，而本算法只需要一次广播即可完成通信范围内所有节点的更新。



(a)4种算法同步误差比较
(a) Synchronization error for 4 algorithms



(b) 本算法与RoATS算法同步误差放大图
(b) The synchronization error of DCCKTS and RoATS

图 6　经典算法的同步误差对比

Fig. 6　Comparison of the synchronization error with classical algorithms

4　结　　论

本文针对无线传感器网络中的时钟同步问题,考虑消息延迟的不确定性对同步过程的影响,在考虑高斯噪声影响的通信过程中,通过改进时钟模型,消除了延迟对后续同步过程的累加影响。为了准确地跟踪延迟变化,利用卡尔曼滤波对时钟同步参数进行估计,仅通过单向通信和有限的信息进行同步,维持了参数估计的准确性和同步精度。另外,本文使用发送节点和接收节点的逻辑时钟的加权平均值作为全局时钟进行同步,加快了收敛速度,提高了同步精度。仿真和硬件实验结果表明,本算法可以在微秒级延迟波动下将网络内的同步误差维持在 10 μs,相比于 WMTS、ATS 和 RoATS 算法,本算法 KBDDCS 能够在接收端独立更新的情况下显著提升同步精度。

参考文献

[1] 李凤保, 李凌. 无线传感器网络技术综述[J]. 仪器仪表学报,2005,26(8):559-561.
LI F B, LI L. Survey on wireless sensor network techniques[J]. Chinese Journal of Scientific Instrument, 2005, 26(8): 559-561.

[2] GEETHA D D, TABASSUM N. A survey on clock synchronization protocols in wireless sensor networks[C]. 2017 International Conference On Smart Technologies For Smart Nation (SmartTechCon), IEEE, 2017: 504-509.

[3] 王鈳, 万羊所, 唐晓铭,等. 不可靠 WSN 时钟同步网络化输出反馈 MPC 量化分析[J]. 仪器仪表学报, 2017, 38(7): 1798-1808.
WANG T, WAN Y S, TANG X M, et al. Unreliable WSN clock synchronization networked output feedback model predictive control quantitative analysis[J]. Chinese Journal of Scientific Instrument, 2017, 38(7): 1798-1808.

[4] 张建军, 魏炬熠, 魏振春. 一种能量有效的自适应 WSN 时间同步算法[J]. 电子测量与仪器学报,2016, 30(8):1220-1227.
ZHANG J J, WEI J Y, WEI ZH CH. Energy-effective and adaptive WSN time synchronization algorithm[J]. Journal of Electronic Measurement and Instrumentation, 2016, 30(8): 1220-1227.

[5] ELSON J, GIROD L, ESTRIN D. Fine-grained network time synchronization using reference broadcasts[C]. 5th Symposium on Operation Systems Design and Implementation (OSD1 02), 2002: 147-163.

[6] GANERIWAL S, KUMAR R, SRIVASTAVA M B. Timing-sync protocol for sensor networks [C]. Proceedings of the 1st International Conference on Embedded Networked Sensor Systems, 2003: 138-149.

[7] 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, 2004: 39-49.

[8] LIU C, PANG H, CAO N. Research on time synchronization technology of wireless sensor network[C]. 2017 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC), IEEE, 2017: 391-394.

[9] HE J, CHENG P, SHI L, et al. Time synchronization in WSNs: A maximum-value-based consensus approach[J]. IEEE Transactions on Automatic Control, 2013, 59(3): 660-675.

[10] SCHENATO L, FIORENTIN F. Average TimeSynch: A

consensus-based protocol for clock synchronization in wireless sensor networks [J]. Automatica, 2011, 47(9): 1878-1886.

[11] GARONE E, GASPARRI A, LAMONACA F. Clock synchronization protocol for wireless sensor networks with bounded communication delays[J]. Automatica, 2015, 59: 60-72.

[12] 汪付强, 曾鹏, 于海斌. 一种低开销的双向时间同步算法[J]. 仪器仪表学报, 2011, 32(6): 1357-1363.
WANG F Q, ZENG P, YU H B. A low overhead two-way time synchronization algorithm[J]. Chinese Journal of Scientific Instrument, 2011, 32(6): 1357-1363.

[13] HAMILTON B R, MA X, ZHAO Q, et al. ACES: Adaptive clock estimation and synchronization using Kalman filtering[C]. Proceedings of the 14th ACM International Conference on Mobile Computing and Networking, 2008: 152-162.

[14] KIRSCH F, VOSSIEK M. Distributed Kalman filter for precise and robust clock synchronization in wireless networks [C]. 2009 IEEE Radio and Wireless Symposium. IEEE, 2009: 482-485.

[15] PING S. Delay measurement time synchronization for wireless sensor networks[J]. Intel Research Berkeley Lab, 2003, 6: 1-10.

[16] WU Y C, CHAUDHARI Q, SERPEDIN E. Clock synchronization of wireless sensor networks[J]. IEEE Signal Processing Magazine, 2010, 28(1): 124-138.

[17] SEMICONDUCTOR N. nRF52832 product specification v1. 4[J]. Datasheet nRF52832, Oct, 2017: 1-553.

作者简介

方子涵,2019 年于东南大学获得学士学位,现为复旦大学硕士研究生,主要研究方向为无线传感器网络中的时钟同步问题。
E-mail: 19210720032@fudan.edu.cn

Fang Zihan received her B. Sc. degree from Southeast University in 2019. She is currently pursuing her M. Sc. degree at Fudan University. Her main research interest is the clock synchronization in wireless sensor networks.

陆起涌(通信作者),分别在 1988 年和 1993 年于复旦大学获得学士学位和硕士学位,现为复旦大学主任技师,主要研究方向为智能仪器仪表、物联网应用。
E-mail: lqyong@fudan.edu.cn

Lu Qiyong (Corresponding author) received his B. Sc. degree and M. Sc. degree both from Fudan University in 1988 and 1993, respectively. He is currently the chief technician at Fudan University. His main research interests include intelligent instruments and application of internet of things.