浙江理工大学学报，2019，41(5)：650-656
Journal of Zhejiang Sci-Tech University
DOI:10.3969/j.issn.1673－3851(n).2019.05.014

基于改进卡尔曼滤波的实时以太网时钟同步优化算法

史仲渊[1]，张　华[1,2,3]，王旭浩[1]
(1.浙江理工大学机械与自动控制学院，杭州 310018；2.浙江大学能源工程学院，杭州 310007；
3.浙江盾安人工环境设备股份有限公司，杭州 310053)

摘　要：为提高分布式控制系统的时钟同步精度，分析了影响时钟同步精度的主要因素，对时钟同步过程建立了含控制量的数学模型，提出了基于改进卡尔曼滤波的实时以太网时钟同步优化算法。针对传统时钟同步算法中驱动层加盖的时间戳精度低、链路延时抖动严重等不足，通过补偿软件时间戳精度，使驱动层获取时间戳更接近物理层获取的时间戳；采用对卡尔曼增益分段的方法改进卡尔曼滤波算法，并使用改进的卡尔曼滤波算法修正时钟偏差值，使时钟偏差测量过程更趋平稳。实验发现，当主从时钟经路由器相连，调整合适卡尔曼滤波参数并补偿软件时间戳精度后，时钟同步精度可达 14 μs。

关键词：实时以太网；时钟同步；卡尔曼滤波；链路延时抖动；卡尔曼增益
中图分类号：TN274　　**文献标志码**：A　　**文章编号**：1673-3851 (2019) 09-0650-07

Real-time ethernet clock synchronization optimization algorithm based on improved Kalman filter

SHI Zhongyuan[1], *ZHANG Hua*[1,2,3], *WANG Xuhao*[1]
(1.Faculty of Mechanical Engineering & Automation, Zhejiang Sci-Tech University, Hangzhou 310018, China;
2.College of Energy Engineering, Zhejiang University, Hangzhou 310007, China;
3.Zhejiang Dun'an Artificial Environment Co., Ltd., Hangzhou 310053, China)

Abstract: In order to improve the clock synchronization accuracy of distributed control system, the main factors affecting the clock synchronization accuracy were analyzed, and the mathematical model containing control quantity was established for the clock synchronization process. Besides, a real-time ethernet clock synchronization optimization algorithm based on Kalman filtering was proposed. The timestamp stamped by the driver layer in the traditional clock synchronization algorithm has low precision, and link delay jitter is serious. For the problems, through compensating the precision of the software timestamp, the timestamp obtained by the driver layer was closer to that obtained by the physical layer. And the Kalman filter algorithm was improved based on piecewise Kalman gain, and the improved Kalman filtering algorithm was used to correct the clock deviation so that the clock deviation measurement process became more stable. It was found that the clock synchronization accuracycould reach 14 μs when the master and slave clocks were connected by the router, the appropriate Kalman filtering parameters were adjusted and the software timestamp precision was compensated.

Key words: real-time ethernet; clock synchronization; Kalman filter; link delay jitter; Kalman gain

收稿日期：2019－01－30　　**网络出版日期**：2019－05－05
基金项目：国家自然科学基金项目(U1609205，51675488，51307151)；浙江省自然科学基金项目(LY18E070006，LY18E050016)
作者简介：史仲渊(1993－　)，男，浙江象山人，硕士研究生，主要从事机电控制以及自动化方面的研究。
通信作者：张　华，E-mail：zhanghua@zstu.edu.cn

0　引　言

实时以太网凭借高传输速度、高带宽等优点，已成为运动控制领域的重要技术。而同步运动控制作为运动控制领域的关键技术，同时也是实时以太网通讯技术实现的中心环节，目前主要存在分布式时钟同步和同步控制策略两个问题[1]。对于分布式时钟同步问题，传统的时间同步技术网络协议(Network time protocol，NTP)采用应用层同步，精度在 10 ms 到 100 ms 之间[2]。2002 年推出的网络测量和控制系统的精密时钟同步协议标准(IEEE1588 precision clock synchronization protocol，IEEE1588)，通过确定主从时钟状态、交换同步报文等方案，实现亚微秒级的时钟同步[3]。2008 年修订的精确时钟同步协议第二版，引入了透明时钟，改进了边界时钟逐级时钟传递方式，进一步提高了时钟同步精度[4]。IEEE1588 协议的主要实现方式有两种：一种是使用支持 IEEE1588 协议的芯片或嵌入式微控制器在物理层加盖时间戳的硬件实现方式；另一种是使用软件在网络驱动层获取时间戳的软件实现方式。朱望纯等[5]使用专用端口物理层(Port physical layer，PHY)芯片 DP83640 在物理层加盖时间戳，实现时钟偏差在 100 ns 左右的时钟同步。该方法可通过硬件辅助获得最高精度的时间戳，但未解决链路非对称延时等问题，且在批量生产中生产成本较高。陶稳静等[6]采用基于开源代码 PTPd2 的纯软件方式，在网络驱动层获取时间戳，并对该时间戳进行补偿，实现时钟偏差在 30 μs 左右的时钟同步。但该方案在使用时由于时钟偏差测量过程噪声较大，且未消除链路非对称延时干扰，因而同步精度不佳，而卡尔曼滤波算法可大幅度降低测量噪声干扰，使测量过程更平稳。但目前已有的一些卡尔曼滤波算法[7]建模过程复杂，且未消除野值对滤波过程的影响，滤波精度有待提高。

本文对实时以太网下分布式控制系统中时钟同步算法进行改进，提出一种基于改进卡尔曼滤波的时钟同步优化算法。首先在驱动层获取时间戳；再通过 Wireshark 软件获取报文从驱动层到物理层的驻留时间，对时间戳进行补偿；然后通过改进卡尔曼滤波降低测量噪声干扰，提高时钟同步精度；最后通过对比实验，验证该算法的有效性。

1　IEEE1588 同步原理及影响同步精度的因素

1.1　时钟同步原理

IEEE1588 的同步原理是先利用最佳主时钟算法(BMC)[8]选择更精确的时钟源作为最佳主时钟，再通过本地时钟同步算法(LCS)周期性地交换报文信息，计算时钟偏差和传输延时，最后根据时钟偏差修正从时钟的本地时钟状态来完成时钟同步[9]。具体同步过程如图 1 所示。

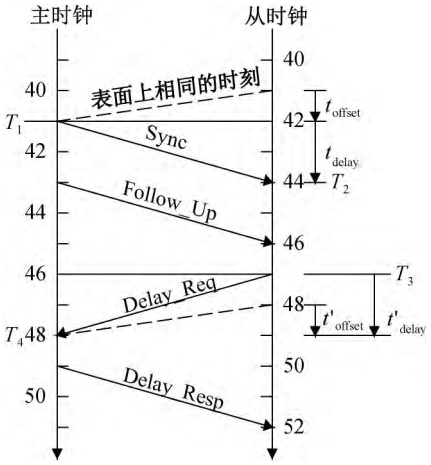

图 1　IEEE1588 时钟同步过程

IEEE1588 的同步步骤如下：

a)主时钟端先于 T_1 时刻向从时钟端发送同步报文(Sync Message)，再发送带有该时间戳 T_1 的跟随报文(Follow_Up Message)。

b)从时钟端在收到 Sync 报文的同时，记录收到报文的精确时间 T_2，并于 T_3 时刻向主时钟端发送延迟请求报文(Delay_Req Message)。

c)从时钟端接收来自主时钟端的 Follow_Up 报文，提取其中精确时间戳 T_1。

d)主时钟端在收到 Delay_Req 报文后，记录收到报文的精确时间 T_4，并向从时钟端发送带有 T_4 的延迟响应报文(Delay_Resp Message)。

e)从时钟端收到 Delay_Resp 报文，提取其中记录的精确时间戳 T_4。

f)定义 t_{offset} 为 T_1 时刻的时钟偏差，t_{delay} 为主时钟端到从时钟端的传输延时，t'_{offset} 为 T_3 时刻的时钟偏差，t'_{delay} 为从时钟端到主时钟端的传输延时，利用时间戳之间的关系，如图 1 所示，可推导出公式：

$$T_2 - T_1 = t_{offset} + t_{delay} \tag{1}$$

$$T_4 - T_3 = t'_{delay} - t'_{offset} \tag{2}$$

假设链路对称[10](即 $t_{delay} = t'_{delay}$)，短时间内主从时钟时钟频率不变($t_{offset} = t'_{offset}$)，可由式(1)—式(2)得时钟偏差和传输延时的计算公式：

$$t_{delay} = (T_2 - T_1 + T_4 - T_3)/2 \tag{3}$$

$$t_{offset} = (T_2 + T_3 - T_1 - T_4)/2 \tag{4}$$

g)根据计算所得的 t_{offset} 和 t_{delay}，调整从时钟设备的本地时钟，完成一次时钟同步过程。

1.2 时钟同步精度的影响因素

IEEE1588 同步协议能实现亚微秒级的时钟同步,但在实际实现过程中,还需注意影响时钟同步精度的主要因素如时间戳获取位置、网络链路不对称、同步时间间隔等。

a)时间戳获取位置。T_1 等时间戳是报文离开或接收的精确时间,又是时钟偏差的计算单元,直接决定了同步精度。而时间戳标记的精度,主要由获取位置决定。报文在应用层生成(处理),然后经过一系列封包(解包),通过网卡发送(接收)。期间,操作系统响应延时、协议栈处理延时、逐层传递的抖动延时等误差使传递给对方时钟的时间戳与真实时间有所偏差[11]。因此,时间戳的产生地点越接近底层,精度越高。若采用专用芯片加盖时间戳,可于媒体独立接口(MII)或者 PHY 处生成时间戳[12],虽提高了时间戳精度,但增加了生产成本。若采用软件方式,最理想的时间戳生成地点则是网络驱动层。

b)网络链路不对称。在计算传输延时时,需假设链路对称。但在实际情况中,由于网络中负载和流量时刻变化,造成网络不确定性延时,很难保证两个方向上传输延时一致。此外,网络节点数的增加,路由器、中继器、集线器、交换机等网络中转设备的引入,进一步增大了传输延时,加重了链路不对称性对同步精度的影响。

c)同步时间间隔。主从时钟间频率漂移会随时间累积,加剧时钟偏差,因此理论上应尽可能减小同步时间间隔,但过短的同步间隔会占用过多的网络带宽,从而造成网络拥堵,降低同步精度。

2 时钟同步优化算法

2.1 时钟同步优化算法流程

时钟同步优化算法主要分为最佳主时钟算法、改进的本地时钟同步算法和改进的卡尔曼滤波算法三个部分。优化算法整体流程如图 2 所示,主要步骤如下:

a)系统启动后,服务器端新开子进程,重新设定 UDP 套接字建立与客户端的连接,父进程阻塞等待。

b)已接入的时钟设备根据最佳主时钟算法确定主从时钟并设置各时钟状态。

c)主从时钟间根据本地时钟同步算法交互报文数据,获取软件时间戳。

d)主时钟端测量出报文从驱动层到物理层的驻留时间,补偿软件时间戳,并以该时间戳为基础计算时钟偏差和传输延时。

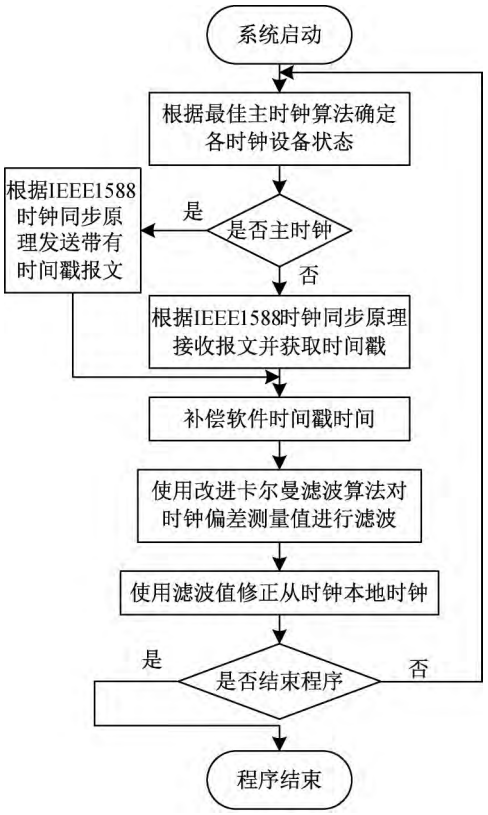

图 2 时钟同步优化算法整体流程

e)从时钟端使用改进卡尔曼滤波算法对计算所得的时钟偏差值进行滤波处理。

f)以滤波值修正从时钟端本地时钟,完成一次同步过程。

2.2 时间戳获取方式的改进

NTP 协议在应用层获取时间戳,同步精度仅达毫秒级。为获得更高的时间戳精度,可从网络驱动层获取时间戳。在 Linux 系统中,需先使用 setsocket 函数,设置 SO_TIMESTAMPNS 选项[13],使报文于驱动层封包时,在其附属数据中添加以纳秒为单位的时间值。在解包过程中,使用 CMSG_FRITHDR 和 CMSG_NETHDR 宏遍历附属数据,以附属数据成员 cmsg_type 是否为 SCM_TIMESTAMP 类型为条件,获取其中匹配的时间戳数据,存放于 timespec 结构体类型中。

2.3 补偿软件时间戳时精度

时间戳获取位置如图 3 所示,驱动层获取的软件时间戳,与物理层获取的硬件时间戳有一定偏差。可通过补偿报文从驱动层传输到物理层所经历的时间,提高软件时间戳精度。先令主时钟组播 Sync 报文和 Follow_Up 报文,使用 Wireshark 软件获得 Sync 报文到达物理层的时间 T_{m1} 和 Follow_Up 报

文中携带的驱动层时间戳 T_{m2}，所需补偿主时钟软件时间戳时间 $\Delta T_{master}=T_{m1}-T_{m2}$。同理可补偿从时钟的软件时间戳时间，使软件获取的时间戳更接近于硬件获取的时间戳。本文中从站时间戳于 MAC 层获取，与物理层时间戳近似，不需要补偿。

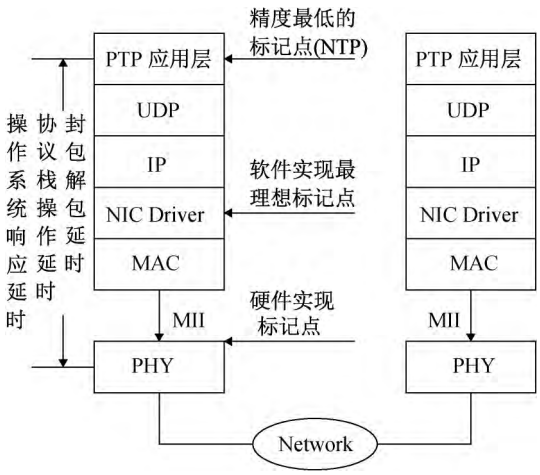

图 3　时间戳获取位置

2.4　时钟偏差的卡尔曼滤波过程

假设同步时间间隔为 T_0，用 $T'_{offset}(n)$ 表示时刻 nT_0 处时钟偏差的真实值，$T_{offset}(n)$ 表示时刻 nT_0 处的测量值，有观测模型如下：$T_{offset}(n)=T'_{offset}(n)+v(n)$。式中，$v(n)$ 表示时钟偏差噪声，可假设为零均值，方差为 σ_v^2 的白噪声。再假设主时钟的频率为 f，无噪声影响。记时刻 nT_0 处从时钟频率为 $f(n)$，频率漂移为 $f'(n)$，正常情况下，频率漂移 $f'(n)$ 可表示为：$f'(n)=a(n)+w(n)$。式中 $a(n)$ 为晶振正常漂移，受环境及晶振老化的影响，$w(n)$ 为随机漂移噪声，为零均值，方差为 σ_w^2，独立于 $v(n)$ 的白噪声[14]。正常工况下，$a(n)$ 为常值，则 $T'_{offset}(n)$ 可写成如下积分形式：$T'_{offset}(n)=\int_0^x(f(n)-f)\mathrm{d}\tau+\theta_0$。离散化后得从时钟的状态方程和观测方程如式(5)：

$$\begin{cases}\begin{bmatrix}T'_{offset}(n+1)\\ f(n+1)-f\end{bmatrix}=\begin{bmatrix}1 & T_0\\ 0 & 1\end{bmatrix}\begin{bmatrix}T'_{offset}(n)\\ f(n)-f\end{bmatrix}+\\ \qquad\begin{bmatrix}0.5T_0^2\\ T_0\end{bmatrix}a(n)+\begin{bmatrix}0.5T_0^2\\ T_0\end{bmatrix}w(n)\\ T_{offset}(n+1)=\begin{bmatrix}1 & 0\end{bmatrix}\begin{bmatrix}T'_{offset}(n)\\ f(n)-f\end{bmatrix}+v(n)\end{cases}\tag{5}$$

定义在时刻 $(n+1)T_0$ 处的系统状态 $\boldsymbol{X}(n+1)$ 为从时钟的时钟偏差真实值和时钟频率，即：$\boldsymbol{X}(n+1)=\begin{bmatrix}T'_{offset}(n+1)\\ f(n+1)-f\end{bmatrix}$。$\boldsymbol{Y}(n+1)$ 为对于状态的观测信号，即 $\boldsymbol{Y}(n+1)=[T_{offset}(n+1)]$，状态转移矩阵 $\boldsymbol{A}=\begin{bmatrix}1 & T_0\\ 0 & 1\end{bmatrix}$，控制量矩阵和噪声驱动矩阵 $\boldsymbol{B}=\boldsymbol{\tau}=\begin{bmatrix}0.5T_0^2\\ T_0\end{bmatrix}$，观测矩阵 $\boldsymbol{H}=[1\quad 0]$，式(5)可改写为式(6)，并对系统状态进行卡尔曼滤波。

$$\begin{cases}\boldsymbol{X}(n+1)=\boldsymbol{A}\boldsymbol{X}(n)+\boldsymbol{B}a(n)+\boldsymbol{\tau}w(n)\\ \boldsymbol{Y}(n+1)=\boldsymbol{H}\boldsymbol{X}(n)+v(n)\end{cases}\tag{6}$$

a) 对 $(n+1)T_0$ 时刻的状态预测，如式(7)：

$$\begin{cases}\boldsymbol{X}(n+1\mid n)=\boldsymbol{A}\boldsymbol{X}(n\mid n)+\boldsymbol{B}u(n)\\ \boldsymbol{P}(n+1\mid n)=\boldsymbol{A}\boldsymbol{P}(n+n)\boldsymbol{A}^{\mathrm{T}}+\boldsymbol{Q}\end{cases}\tag{7}$$

其中：$\boldsymbol{X}(n+1|n)$ 为 $(n+1)T_0$ 时刻先验状态预测；$\boldsymbol{P}(n+1|n)$ 为先验估计协方差，是根据前次迭代结果的不可靠预测；$\boldsymbol{X}(n\mid n)$ 表示 nT_0 时刻后验状态预测；$\boldsymbol{P}(n+1|n)$ 是后验估计协方差，是 nT_0 时刻的最优预测。初始状态 $\boldsymbol{X}(0\mid 0)=[T_{offset}(0)\quad f(1)-f]^{\mathrm{T}}$，$\boldsymbol{P}(0\mid 0)$ 为任意非零值。$u(n)$ 是 nT_0 时刻控制量，有 $u(n)=a(n)+w(n)$。$\boldsymbol{Q}$ 为过程噪声协方差矩阵，且有 $\boldsymbol{Q}=\begin{bmatrix}\sigma_v^2T_0 & 0\\ 0 & \sigma_w^2T_0\end{bmatrix}$。

b) 对 $(n+1)T_0$ 时刻的状态更新，如式(8)—(10)：

$$\boldsymbol{K}(n+1)=\boldsymbol{P}(n+1\mid n)\boldsymbol{H}^{\mathrm{T}}[\boldsymbol{H}\boldsymbol{P}(n+1\mid n)\boldsymbol{H}^{\mathrm{T}}+\boldsymbol{R}]^{-1}\tag{8}$$

$$\boldsymbol{\varepsilon}(n+1)=\boldsymbol{Y}(n+1)-\boldsymbol{H}\boldsymbol{X}(n+1\mid n)\tag{9}$$

$$\begin{cases}\boldsymbol{X}(n+1\mid n+1)=\boldsymbol{X}(n+1\mid n)+\boldsymbol{K}(n+1)\\ \qquad\boldsymbol{\varepsilon}(n+1)\\ \boldsymbol{P}(n+1\mid n+1)=(\boldsymbol{I}-\boldsymbol{K}(n+1)\boldsymbol{H})\boldsymbol{P}(n+1\mid n)\end{cases}\tag{10}$$

其中：$\boldsymbol{K}(n+1)$ 表示 $(n+1)T_0$ 时刻的卡尔曼增益；测量噪声协方差矩阵 $\boldsymbol{R}$ 取决于与测量设备[15]相关的测量噪声协方差 σ_m^2；$\boldsymbol{\varepsilon}(n+1)$ 表示 $(n+1)T_0$ 时刻的新息。根据 nT_0 时刻滤波值和 $(n+1)T_0$ 时刻测量值，经过预测、更新，获得式(10)中 $(n+1)T_0$ 时刻系统状态。

2.5　改进的卡尔曼滤波算法

由于在时钟偏差、频率漂移的计算及测量过程中存在各种随机噪声，滤波后的系统状态中可能存在野值，严重降低数据的可靠性。而卡尔曼滤波算法对数据依赖性极强，极易受野值干扰而影响滤波精度。针对这一情况，本文采用对卡尔曼增益分段的方法来改进卡尔曼滤波算法，分段表达式可以表示为：

$$\boldsymbol{K}(n+1)=\begin{cases}\boldsymbol{P}(n+1\mid n)\boldsymbol{H}^{\mathrm{T}}[\boldsymbol{H}\boldsymbol{P}(n+1\mid n)\boldsymbol{H}^{\mathrm{T}}+\boldsymbol{R}]^{-1}, \text{无野值时}\\ m\times\boldsymbol{P}(n+1\mid n)\boldsymbol{H}^{\mathrm{T}}[\boldsymbol{H}\boldsymbol{P}(n+1\mid n)\boldsymbol{H}^{\mathrm{T}}+\boldsymbol{R}]^{-1}, \text{出现野值时}\end{cases} \tag{11}$$

当无野值时，卡尔曼增益不变；当野值出现时，通过影响新息，使新息方差增大，此时应调小卡尔曼增益，降低新息的影响，取 m 的范围为[0,1]。

而对于是否出现野值，可以新息方差平方根的倍数作为判断条件：

$$|\boldsymbol{\varepsilon}|\leqslant d\times\mathrm{sqrt}[|\boldsymbol{H}\boldsymbol{P}(n+1\mid n)\boldsymbol{H}^{\mathrm{T}}+\boldsymbol{R}|] \tag{12}$$

正常情况下，新息是符合零均值的高斯分布，本文中对实验数据进行统计，计算野值的出现概率约为 0.2%，因此取倍数 $d=2$。

3 实验及结果分析

实验使用计算机(PC)和嵌入式微控制器进行，PC 端处理器选择 Intel (R) Core (TM) i5-3317U，主频为 1.70 GHz，安装有 Ubuntu14.04LTS 系统。嵌入式微控制器采用 AM3359 工业通讯引擎，移植内核为 v4.14.79 的 Linux-RT 系统，同步控制系统实验平台如图 4 所示。系统启动后，PC 机和嵌入式端同时运行程序，由 BMC 算法决策出主时钟为 PC 机。通过设置不同 σ_v^2、σ_m^2，不同同步时间间隔，连接不同中间设备，及补偿软件时间戳等操作分别进行实验，分析不同参数及中间设备对滤波精度及时钟同步精度的影响。

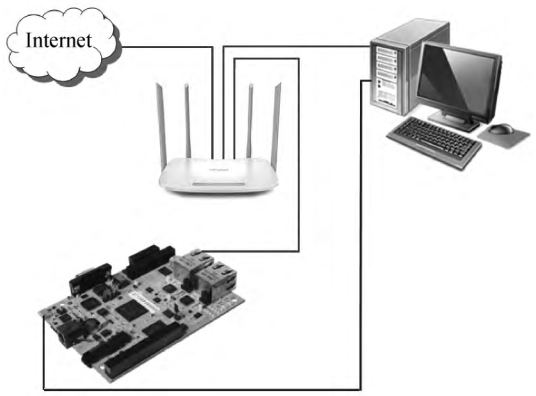

图 4 时钟同步控制系统实验平台

3.1 软件时间戳精度对同步精度的影响

保证同步时间间隔为 1.00 s，连接方式为通过路由器相连，无滤波的条件下，对补偿软件时间戳前后进行实验，记录 1000 个同步周期内时钟偏差的最大值、平均值及标准差，验证软件时间戳精度对同步精度的影响，实验结果见表 1。

表 1 软件时间戳补偿前后时钟同步精度表

是否补偿软件时间戳	最大时钟偏差/μs	平均时钟偏差/μs	标准差/μs
否	131.617	37.364	24.879
是	121.541	26.185	17.572

补偿软件时间戳后，最大时钟偏差为 121.541 μs，平均时钟偏差为 26.185 μs，相比于补偿前，均有明显下降，可见时间戳精度对时钟同步精度有较大影响，且补偿软件时间戳算法是合理的。

3.2 不同同步时间间隔对同步精度的影响

改变同步时间间隔，验证不同 T_0 对同步精度的影响。在无滤波，保持主从时钟端通过路由器相连情况下，分别记录同步时间间隔在 0.25、0.50、1.00、2.00、5.00 s 下的从时钟端的时钟偏差值，见表 2。

表 2 同步时间间隔与时钟同步精度表

同步时间间隔/s	最大时钟偏差/μs	平均时钟偏差/μs	标准差/μs
0.25	127.314	27.032	18.530
0.50	115.293	26.007	18.343
1.00	124.086	26.251	17.024
2.00	211.078	45.955	43.149
5.00	427.830	111.562	107.273

T_0 为 0.50～1.00 s 时，平均时钟偏差较小，同步精度较高。T_0 从 0.25 s 到 0.50 s，同步精度有所提高，这主要是因为主从时钟间时钟同步过于频繁，占用大量网络带宽，造成网络拥堵。而 T_0 从 1.00 s 到 5.00 s，平均时钟偏差逐渐增大，同步精度降低，是因为主从时钟频率漂移所造成的误差在整个同步间隔内累加，随 T_0 的增长而增大。综合考虑以上因素，推荐同步时间间隔为 1.00 s。

3.3 不同中间设备对同步精度的影响

保证无滤波、同步时间间隔为 1.00 s，改变主从时钟设备间连接方式，分析由于中间设备不同，导致链路不对称性加大，对同步精度的影响。其中，连接方式分别采用网线直接，经交换机、路由器连接，实验结果见表 3。

表 3 不同中间设备时钟同步精度表

中间设备	最大时钟偏差/μs	平均时钟偏差/μs	标准差/μs
网线直连	97.588	22.887	15.819
经交换机连接	99.234	24.321	16.293
经路由器连接	121.822	26.486	17.135

实验结果发现，网线直连的同步精度最高，经路由器连接的同步精度最低，这是因为路由器工作在网络层，根据 IP 地址寻址，交换机工作与数据链路层，根据 MAC 地址寻址，路由器较交换机更多一层封装，因此延时更长。

3.4　不同卡尔曼滤波参数对同步精度的影响

滤波参数 σ_v^2、σ_m^2 不同，滤波后所达到的同步精度不同。根据滤波原理，保证同步时间间隔为 1.00 s，连接方式为通过路由器相连，将 σ_v^2、σ_m^2 分别从 $10^{-7} \sim 10^{-3}$ 间取值，进行大量实验，选择最佳值。选取几组典型 σ_v^2、σ_m^2 值对比滤波精度，待滤波稳定后按滤波参数各周期时钟偏差值如图 5 所示。

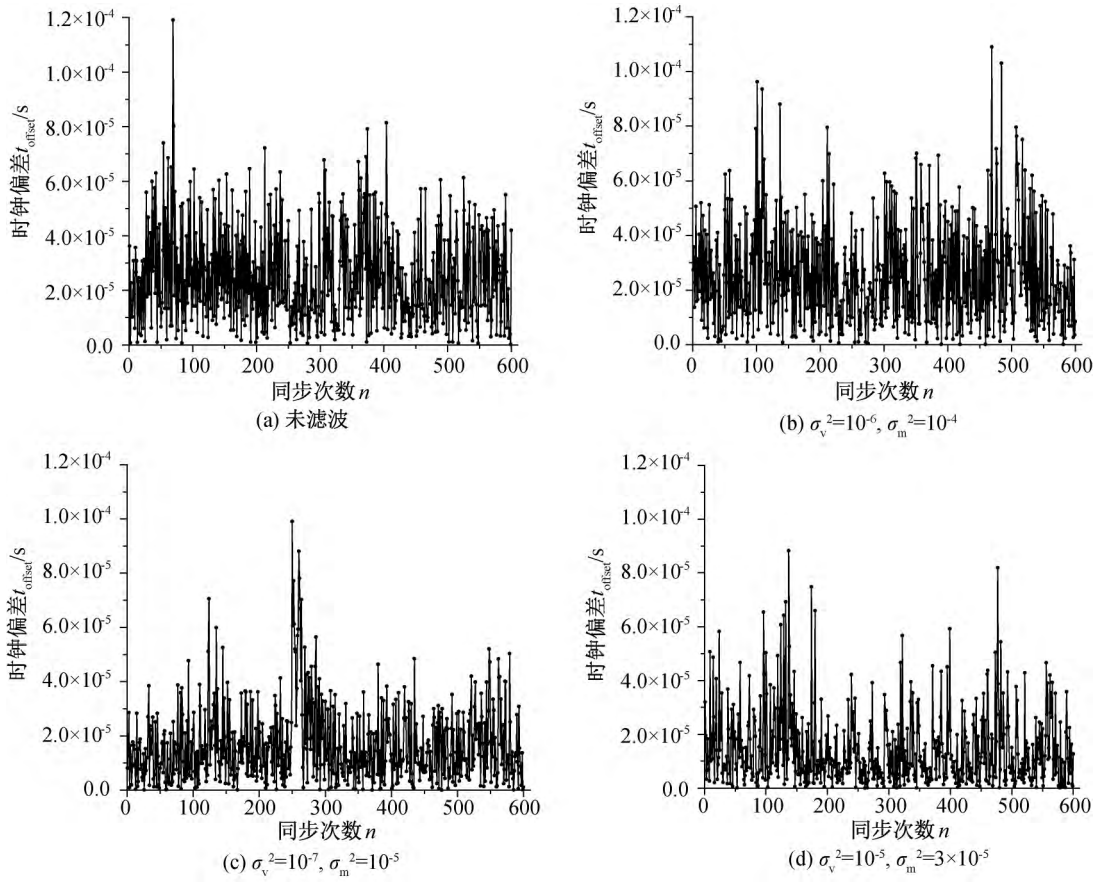

图 5　滤波前及不同卡尔曼滤波参数下时钟偏差

如图 5 所示，滤波前时钟偏差值较高且抖动严重，滤波后时钟偏差趋于平稳并有明显下降，各参数下最大时钟偏差和平均时钟偏差见表 4。对比图 5(b)—(d)滤波结果，在参数 $\sigma_v^2=10^{-6}$，$\sigma_m^2=10^{-4}$ 时，滤波效果不佳，稳定后平均时钟偏差为 24.883 μs。$\sigma_v^2=10^{-7}$，$\sigma_m^2=10^{-5}$ 时，滤波收敛较慢，约在 4 min 左右收敛，而滤波效果较好，各时刻时钟偏差值如图 5(c)所示，平均时钟偏差为 20.767 μs。经过大量实验，最终确定参数 $\sigma_v^2=10^{-5}$，$\sigma_m^2=3\times10^{-5}$。此时，滤波过程较平稳，滤波效果最好，收敛速度快，约在 1 min左右收敛，各时刻时钟偏差值如图 5(d)所示，稳定后最大时钟偏差为 91.384 μs，平均时钟偏差为 16.152 μs，相比于滤波前，平均时钟偏差降低了近 10 μs。

表 4　卡尔曼滤波参数与时钟同步精度表

是否滤波	滤波参数	最大时钟偏差/μs	平均时钟偏差/μs	标准差/μs
否	—	119.019	26.068	17.085
是	$\sigma_v^2=10^{-6}$，$\sigma_m^2=1\times10^{-4}$	109.014	24.883	17.971
是	$\sigma_v^2=10^{-7}$，$\sigma_m^2=1\times10^{-5}$	99.139	20.767	16.267
是	$\sigma_v^2=10^{-5}$，$\sigma_m^2=3\times10^{-5}$	91.384	16.152	15.369

3.5 不同算法时钟同步精度对比

使用改进卡尔曼滤波算法,选取相同滤波参数进行实验,再列出文献[6]中采用不同算法,相同其他条件(同步时间间隔为1.00 s,连接方式为经路由器相连,补偿软件时间戳)下的时钟同步精度,见表5。

表5 不同算法下时钟同步精度表

算法	最大时钟偏差/μs	平均时钟偏差/μs	标准差/μs
PI算法	169.339	30.277	37.944
卡尔曼滤波算法	91.384	16.152	15.369
改进卡尔曼滤波算法	87.236	14.177	13.549

相对于PI算法,采用卡尔曼滤波算法实现的时钟同步无论是在最大时钟偏差、平均时钟偏差还是标准差上均有明显下降,可见卡尔曼滤波算法的优越性,在提高同步精度的同时,还提高了时钟偏差值测量的稳定性。使用改进卡尔曼滤波算法后平均时钟偏差为14.177 μs,相对于传统卡尔曼滤波算法,有近2 μs的提高,这是因为改进卡尔曼滤波修正了野值,大幅度提高了原野值点后的滤波稳定性,从而进一步提高了整体滤波精度。

4 结 论

本文提出并实现了一种基于改进卡尔曼滤波的时钟同步优化算法。通过对补偿软件时间戳前后进行实验,验证补偿时间戳方案的合理性;调整同步时间间隔进行实验,确定最佳同步间隔为1.00 s;连接不同中间设备进行实验,验证链路不对称性对同步精度的影响;通过大量实验,确定最佳卡尔曼滤波参数;对比不同算法下平均时钟偏差,验证改进卡尔曼滤波算法的优越性。实验结果表明,当主从时钟经路由器相连,采用该时钟同步优化算法后,时钟同步精度可达14 μs。

参考文献:

[1] 林梦云,马文礼,熊皑,等. 基于EtherCAT实时通信的电机驱动控制[J]. 微型机与应用,2017,36(10):1-4.

[2] Kikutani T, Yakoh T. A precise time synchronization method for real-time schedulers[J]. Electronics and Communications in Japan, 2018, 101(12): 21-29.

[3] Li D J, Wang G, Yang C J, et al. IEEE 1588 based time synchronization system for a seafloor observatory network[J]. Journal of Zhejiang University-Science C (Computers and Electronics), 2013, 14(10): 766-776.

[4] Xu W. A clock synchronization method based on IEEE 1588 and its implementation on Ethernet[J]. Journal of Chongqing University, 2008, 7(2): 145-150.

[5] 朱望纯,覃斌毅,王玉娟. 基于IEEE1588协议同步技术的研究[J]. 测控技术,2014,33(7):98-101.

[6] 陶稳静,陆阳,卫星,等. 基于PTPd2的精密时钟同步的软件实现方法[J/OL]. 计算机工程. (2018-06-21)[2019-03-26]. https://doi.org/10.19678/j.issn.1000-3428.0050662.

[7] 李进燕,朱征宇,刘琳,等. 基于简化路网模型的卡尔曼滤波多步行程时间预测方法[J]. 系统工程理论与实践,2013,33(5):1289-1297.

[8] 钱伟康,郭超,应怀樵. 基于Linux的IEEE1588精确时钟同步的实现[J]. 测控技术,2012,31(11):15-19.

[9] 章涵,冯冬芹,褚健. 基于路径加权反馈的工业环网时钟同步方法[J]. 浙江大学学报(工学版),2010,44(5):849-853.

[10] 苏建徽,陈亚园. 微电网精密时钟同步技术[J]. 电力自动化设备,2016,36(11):40-44.

[11] 覃斌毅,陈铁军,邱杰,等. 基于IEEE1588协议时钟同步精度影响因素的研究[J]. 计算机测量与控制,2014,22(10):3312-3315.

[12] 肖鲲,徐刚,袁海涛. 基于DP83640的IEEE-1588时钟同步协议在POWERPC体系上的实现方案和技巧[J]. 电气自动化,2012,34(6):71-73.

[13] 黄威然,楼佩煌,钱晓明. 基于实时以太网的网络化数控系统高精度时钟同步和短周期通信[J]. 计算机集成制造系统,2015,21(10):2668-2676.

[14] 李成龙,钟凡,马昕,等. 基于卡尔曼滤波和随机回归森林的实时头部姿态估计[J]. 计算机辅助设计与图形学学报,2017(12):159-166.

[15] 涂海峰,贾生伟,阳丰俊,等. 基于无迹卡尔曼滤波的巡飞弹气动参数在线辨识方法[J]. 航天控制,2018(5):14-18.

(责任编辑:康 锋)