

基于 PTPd2 的精密时钟同步软件实现方法

陶稳静¹, 陆 阳^{1,2}, 卫 星^{1,2}, 贾向利¹

(1. 合肥工业大学 计算机与信息学院, 合肥 230601;

2. 安全关键工业测控技术教育部工程研究中心, 合肥 230601)

摘 要: 针对软件实现方式在协议栈网络驱动层获取时间戳, 易受协议栈延时和抖动影响导致同步精度较低的问题, 提出一种精密时钟同步软件实现方法。基于开源代码 PTPd2 采用纯软件方式实现 IEEE 1588 协议, 通过 Linux 系统的内核函数在网络驱动层获取报文时间戳, 相比网络时间协议在应用层获取时间戳可有效避免协议栈的干扰, 同时利用 Wireshark 抓包软件捕获报文经过介质访问控制(MAC)层的时间, 在 PTPd2 中补偿报文从网络驱动层传输到 MAC 层产生的延时和抖动, 提高了时钟同步精度。实验结果表明, 当主-从时钟设备直接相连时, 合理设置 P 、 I 值和同步周期并补偿出边界时间, 时钟同步精度可达 $19\ \mu\text{s}$, 可满足大部分分布式控制系统的要求。

关键词: 精确时间协议; 时钟同步; 软件实现; 出边界时间; 时钟偏差

中文引用格式: 陶稳静, 陆阳, 卫星, 等. 基于 PTPd2 的精密时钟同步软件实现方法[J]. 计算机工程, 2019, 45(3): 47-53, 59.

英文引用格式: TAO Wenjing, LU Yang, WEI Xing, et al. Software implementation of precision clock synchronization based on PTPd2[J]. Computer Engineering, 2019, 45(3): 47-53, 59.

Software Implementation of Precision Clock Synchronization Based on PTPd2

TAO Wenjing¹, LU Yang^{1,2}, WEI Xing^{1,2}, JIA Xiangli¹

(1. School of Computer Science and Information Engineering, Hefei University of Technology, Hefei 230601, China;

2. Engineering Research Center of Safety Critical Industry Measure and Control Technology, Ministry of Education, Hefei 230601, China)

[Abstract] Aiming at the problem that the software implementation obtains the timestamp in the network driver layer of the protocol stack, which is susceptible to the delay of the protocol stack and the jitter, the synchronization accuracy is low. A precise clock synchronization software implementation method is proposed. Based on open source code PTPd2, this paper uses pure software method to implement the IEEE 1588 protocol, and uses the kernel function of Linux system to get the message timestamp in the network driver layer, it can effectively avoid the interference of protocol stack compared with Network Time Protocol(NTP) that get timestamp in application layer, at the same time, Wireshark packet capture software is used to capture the time when the message passes the Media Access Control(MAC) layer and compensate the delay and jitter generated by the message transmitted from the network driver layer to the MAC layer in PTPd2, thereby it improves the clock synchronization accuracy. Experimental results show that when the master-slave clock device is directly connected, the P 、 I value and the synchronization period are reasonably set and the boundary time is compensated, and the clock synchronization accuracy can reach $19\ \mu\text{s}$. This method can meet the requirements of most distributed control systems.

[Key words] Precision Time Protocol (PTP); clock synchronization; software implementation; outbound time; clock deviation

DOI: 10.19678/j.issn.1000-3428.0050662

0 概述

随着信息技术的迅速发展, 时钟同步技术成为分布式测控系统中的关键技术, 该技术对时钟同步精度的要求越来越高。传统的网络时间协议

(Network Time Protocol, NTP) 在 2 000 km 范围的广域网内同步精度在 10 ms 级, 局域网内的同步精度在 1 ms 级^[1], 这种毫秒级的精度已经无法适用对精度要求更高的场合, 而同步精度在亚微秒级的 IEEE 1588 协议则可以满足需求^[2]。

基金项目: 国家重点研发计划专项(2016YFC0801405, 2016YFC0801804)。

作者简介: 陶稳静(1994—), 女, 硕士研究生, 主研方向为时钟同步技术; 陆 阳, 教授、博士生导师; 卫 星, 副教授; 贾向利, 硕士研究生。

收稿日期: 2018-03-07

修回日期: 2018-05-09

E-mail: taowenjing123789@163.com

IEEE 1588 协议有 3 种实现方式^[3]:基于支持精确时间协议(Precision Time Protocol, PTP)的芯片在物理层获取时间戳的硬件实现方式;基于支持 PTP 的嵌入式微控制器在媒体独立接口(Media Independent Interface, MII)处获取时间戳的硬件实现方式;基于精确时间协议进程(Precision Time Protocol daemon, PTPd)的软件实现方式。文献[4]基于第 1 种方式使用芯片 DP83640 在物理层获取时间戳,同步精度在 100 ns 左右;文献[3, 5]基于第 2 种方式使用 ARM 芯片 STM32F407 进行微电网的时钟同步,同步精度在亚微秒级;文献[6]基于第 3 种方式在一台计算机中安装虚拟机,利用虚拟网络实现主-从时钟的通信,同步精度在 30 μ s;文献[7]基于第 3 种方式将 PTPd 移植到嵌入式设备 ARM11 (S3C6410)中实现 PTP,同步精度在 10 ms 级。软硬件实现 PTP 的区别在于获取时间戳的方式不同,硬件实现是在底层获取时间戳,需要硬件辅助,精度较高,但在实际应用中需要改变原有网络中的硬件设备,增加了成本;软件实现是在协议栈获取时间戳,精度稍低,但是实现简单,节约成本。

本文在分析 IEEE 1588V2 同步原理的基础上,利用 Linux 系统内核开源的特点,改进时间戳获取方式,在介质访问控制(Media Access Control, MAC)层获取时间戳以补偿主-从时钟的出边界时间,以提高同步精度。此外,对影响同步精度的其他因素,如 P 、 I 值、同步周期、中间设备等进行了分析和实验验证。

1 IEEE 1588V2 协议与 PTPd 协议

IEEE 1588V2 协议由 IEEE 组织在 2008 年发布,相比 IEEE 1588V1 协议,IEEE 1588V2 协议的主要改进有以下 3 个方面:1)简化报文格式以节省网络带宽;2)增加端延时测量方式测量节点之间的链路延迟;3)增加 E2E(End-to-End)和 P2P(Peer-to-Peer)2 种透明时钟模型,降低链路非对称性对时钟同步精度的影响。

1.1 IEEE 1588 同步原理

PTP 的工作原理是首先利用最佳主时钟(Best Master Clock, BMC)算法选择一个精确的时钟源作为最佳主时钟,然后交换报文信息,利用本地时钟同步(Local Clock Synchronization, LCS)算法,周期性地同步和修改网络中的所有时钟^[8]。主-从时钟同步原理如图 1 所示。首先主时钟周期性地发送 Sync 报文,系统记录 Sync 报文的精确发送时间 t_1 ;接着主时钟发送 Follow_Up 报文,该报文携带 Sync 报文

发送的精确时间 t_1 ,从时钟接收到 Sync 报文,并记录接收时间 t_2 ;然后从时钟发送 Delay_Req 报文,并记录发送时刻 t_3 ,主时钟接收到 Delay_Req 报文,并记录接收时刻 t_4 ;最后主时钟将时间信息 t_4 放入 Delay_Resp 报文中发送给从时钟,当从时钟接收到 Delay_Resp 报文时,一个同步周期结束。

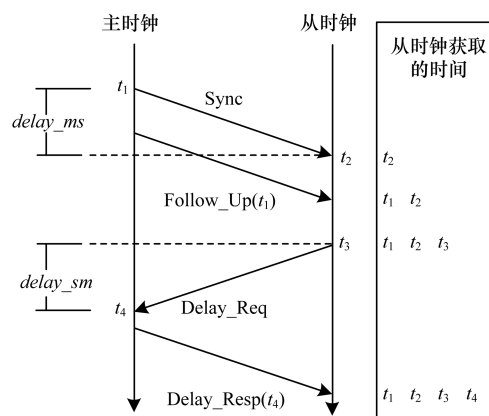


图 1 主-从时钟同步原理

定义 t_1 时刻主-从时钟之间的时钟偏差为 $offset$ (假设一个同步周期内主-从时钟频率一致),主-从链路延迟为 $delay_ms$,从-主链路延迟为 $delay_sm$,则有:

$$t_2 - t_1 = delay_ms + offset \quad (1)$$

$$t_4 - t_3 = delay_sm - offset \quad (2)$$

协议假定主-从时钟的往返时间是对称的,即 $delay_ms = delay_sm = delay$,因此,根据式(1)、式(2)有:

$$offset = (t_2 - t_1 + t_3 - t_4) / 2 \quad (3)$$

$$delay = (t_2 - t_1 + t_4 - t_3) / 2 \quad (4)$$

最终从时钟可根据主-从时钟偏差 $offset$ 和链路延迟 $delay$ 调整本地时钟与主时钟保持同步。

在 IEEE 1588V2 中改进了 Sync 报文的发送速率,最高可达 1 000 次/s,而第 1 版中是不大于 1 次/s^[9],但是同步周期太短也会造成网络拥堵,因此,在一般情况下都设置为 1 s 或 2 s。协议规定 Delay_Req 报文的发送周期是 Sync 报文的 1 倍~32 倍,因为链路延迟一般变化不大,所以 Delay_Req 的发送周期可以设置比 Sync 稍长,以节省网络带宽。

1.2 PTPd 功能部件描述

PTPd 是 PTP 的纯软件实现,其最早是由美国 Case Western Reserve University 的 2 名工程专业学生历时 6 个月所开发^[10],PTPd 现在由全球最大的开源软件开发平台 SourceForge 进行更新和维护。目前 PTPd 有 2 个版本,PTPd1 支持 IEEE 1588V1,PTPd2 支持 IEEE 1588V2^[11],PTPd2 支持的开发平台有 Linux、FreeBSD、OpenBSD、NetBSD、Solaris 以

及各种嵌入式系统等。本文采用 PTPd2.1.0 代码, 在 Linux 系统上实现 PTP。

图2所示为 PTPd 的功能部件联系图, 主要由7个部分组成, 分别是协议引擎、报文处理、网络层实现、时间戳获取、时钟伺服模块、定时器以及 BMC 算法。

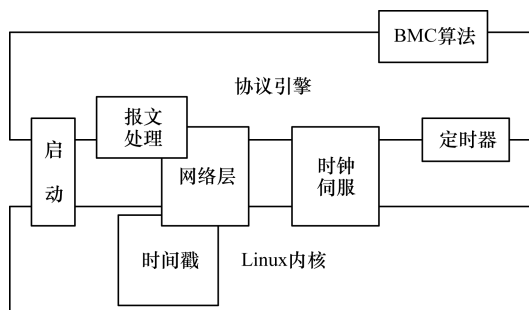


图2 PTPd 功能部件示意图

PTPd 各功能部件详述如下:

1) PTPd 协议引擎。主要运行 PTP 状态机, 由 protocol.c 实现, 完成协议端口状态之间的正确转换, 核心函数是 protocol(), 运行在一个永久循环中, 一般由外部中断或产生错误终止循环。

2) 报文处理。文件 msg.c 详细定义了函数 msgPack() 和 msgUnpack(), msgPack() 用于封装由于定时器超时而要发送的报文信息, msgUnpack() 则解析端口接收到的报文信息。

3) 网络层实现。文件 net.c 实现报文的网络传输, 由 findIface() 函数识别端口硬件地址和 IP 地址, netSelect() 检查是否在网络上接收到数据, 并且所有报文基于 UDP 传输, 根据不同的 UDP 端口号发送和接收报文。

4) 时间戳获取。精确时间戳在 Linux 内核中产生, 由 net.c 中的 recvmsg() 函数获取, 精度在纳秒级。

5) 时钟伺服模块。文件 servo.c 是同步系统的核心, 主要功能是计算主-从时钟偏差和链路延迟并且更新本地时钟。第2.2节详细阐述了本模块的作用。

6) 定时器。由 timer.c 实现, 主要功能是定义定时器中断时间, 根据报文类型设置定时间隔, 并依据定时器是否超时实时更新定时器。

7) BMC 算法。文件 bmc.c 详述了最佳主时钟算法, 依据状态决策算法结果返回各端口应有状态, 仅适用于普通时钟。

2 PTPd 系统设计

2.1 获取时间戳的方式及改进

获取精确时间戳是进行时钟同步的关键一步, 如图3所示, 可在 A、B、C 3 个位置获取时

间戳。

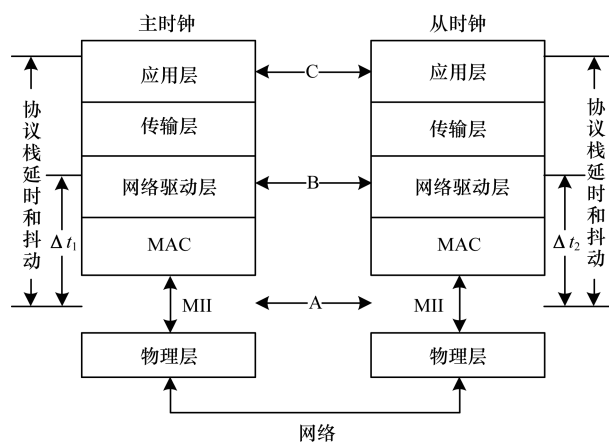


图3 时间戳的产生位置

NTP 在 C 点即应用层获取时间戳, 精度最低; PTP 可在 B 点或 A 点获取时间戳, B 点即网络驱动层, 同步精度比 NTP 高; A 点是在 MII 接口处获取, 同步精度最高, 达亚微秒级, 但是需要硬件辅助。因为报文从 C 点传输到 A 点会受协议栈的延迟和抖动的影响, 所以越靠近物理层获取时间戳, 同步精度越高。

PTPd 是在 B 点获取时间戳, 要获取报文到达网络驱动层的时间戳, 需要先设置 Linux 内核中的 socket 选项为 SO_TIMESTAMP, 但 SO_TIMESTAMP 选项返回微秒级精度, 而 SO_TIMESTAMPNS 选项返回纳秒级精度^[12], 因此, 首先采用 SO_TIMESTAMPNS 选项, 然后用 recvmsg() 函数接收网络报文, 报文到达网络驱动层的时间戳被存储为 SCM_TIMESTAMPNS 类型的附属数据, 使用 CMSG_FIRTHDR() 和 CMSG_NXTHDR() 宏遍历附属数据, 其中匹配 SCM_TIMESTAMPNS 类型的数据即为时间戳, 最后通过 timespec 结构体存储此时间戳。

PTPd 在 B 点获取时间戳可避免报文从 C 点传输到 B 点受协议栈延迟和抖动的影响, 因此, 时钟同步精度较高, 然而报文从 B 点传输到 MAC 层也会受到协议栈延迟和抖动的影响, 若分别在主-从时钟中补偿报文从 B 点传输到 A 点的时间则能提高同步精度。本文定义出边界时间为报文从网络驱动层传输到 MAC 层所经历的时间, 则计算出边界时间的步骤如下:

步骤1 主时钟首先组播 Sync 报文 S_j , 报文 S_j 到达主时钟 MAC 层的时间为 $T_{mac_S_j}$, 随后组播 Follow_Up 报文, Follow_Up 报文记录了报文 S_j 到达主时钟的网络驱动层的精确时间 $T_{ip_S_j}$ 。

步骤2 在主时钟中用 Wireshark 抓包软件捕获报文 S_j , 提取 $T_{mac_S_j}$, 如图4所示。

```

Interface id: 0 (eth0)
Encapsulation type: Ethernet (1)
Arrival Time: Jan 19, 2018 09:29:54.583098000 中国标准时间
[Time shift for this packet: 0.000000000 seconds]
Epoch Time: 1516325394.583098000 seconds
[Time delta from previous captured frame: 0.999940000 seconds]
[Time delta from previous displayed frame: 0.999940000 seconds]

```

图 4 报文到达 MAC 层的时间

步骤 3 在主时钟中用 Wireshark 抓包软件捕获跟随报文 S_j 的 Follow_Up 报文,提取 Tip_S_j ,如图 5 所示。

```

Precision Time Protocol (IEEE1588)
  > 1000 .... = transportSpecific: 0x8
    .... 1000 = messageId: Follow_Up Message (0x8)
    .... 0010 = versionPTP: 2
    messageLength: 44
    subdomainNumber: 0
  > flags: 0x0200
  > correction: 0.000000 nanoseconds
    ClockIdentity: 0xf0def1fffeb56fe7
    SourcePortID: 1
    sequenceId: 26
    control: Follow_Up Message (2)
    logMessagePeriod: 0
    preciseOriginTimestamp (seconds): 1516325394
    preciseOriginTimestamp (nanoseconds): 583063275

```

图 5 报文到达网络驱动层的时间

步骤 4 通过下式计算主时钟的出边界时间 Δt_1 :

$$\Delta t_1 = \left[\sum_{j=1}^n (T_{mac_S_j} - Tip_S_j) \right] / n \quad (5)$$

其中, n 为报文数。

步骤 5 从时钟按照上述相同方式计算从时钟的出边界时间 Δt_2 。

根据上述方法分别计算出主-从时钟的出边界时间 Δt_1 和 Δt_2 , 主时钟发送 Sync 报文, 将主时钟的出边界时间 Δt_1 补偿到 Sync 报文到达主时钟的网络驱动层的时间, 即第 1.1 节所述同步原理的 t_1 中, $t_1 = t_1 + \Delta t_1$, 并将补偿后的时间 t_1 装入 Follow_Up 报文的 preciseOriginTimestamp 字段中发送给从时钟; 从时钟发送 Delay_Req 报文, 将从时钟的出边界时间 Δt_2 补偿到 Delay_Req 报文到达从时钟的网络驱动层的时间, 即第 1.1 节所述同步原理的 t_3 中, $t_3 = t_3 + \Delta t_2$, 并将补偿后的时间 t_3 保存在从时钟中。最后从时钟根据获取的时钟信息 $t_1 \sim t_4$ 进行时钟同步。本文方法补偿主-从时钟的出边界时间, 达到接近 A 点获取时间戳的效果, 因此, 相比在 B 点获取时间戳的时钟同步精度更高。

此外, 文献[13]采用 gettimeofday() 来获取系统时间, 并补偿内核获取该函数的时间来提高同步精

度, 但并不适用于本文系统。文献[14]在 Windows 系统上实现 PTP 也仅能达到亚毫秒级, 由于 Linux 系统下的 clock_gettime() 和 clock_settime() 函数返回纳秒级精度, 因此本文系统采用该函数来获取设定系统时间, 可进一步提高时钟同步精度。

2.2 时钟伺服模块

图 6 所示为 PTPd 系统的时钟伺服模块, 该模块主要由无限冲击响应 (Infinite Impulse Response, IIR) 滤波器、有限冲击响应 (Finite Impulse Response, FIR) 滤波器以及比例积分 (Proportional Integral, PI) 控制器组成。

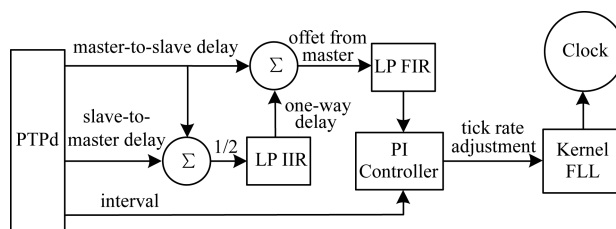


图 6 时钟伺服模块

该模块的作用是由从时钟根据第 1.1 节 IEEE 1588 同步原理计算的主-从时钟偏差 $offset$ 和链路延迟 $delay$ 校正本地时钟。文献[15]采用卡尔曼滤波算法对时钟偏差、时钟漂移变化率进行估计, 并根据估计值修正从时钟, 但算法较复杂, 不易实现。本文分别利用 FIR 和 IIR 滤波器过滤主-从时钟偏差和链路延迟以减小报文传输延时抖动产生的误差, 并使用 PI 控制器计算从时钟相对于主时钟的频率补偿值, 从而补偿系统时钟频率, 比例积分控制器的 $Drift$ 与 adj 如下:

$$Drift = \sum_{i=1}^m \frac{X_i}{I} \quad (6)$$

$$adj = \frac{X_m}{P} + Drift = \frac{X_m}{P} + \sum_{i=1}^m \frac{X_i}{I} \quad (7)$$

其中, I 为积分系数, X_i 为第 i 个同步周期的主-从时钟偏差值, P 为比例系数, $Drift$ 为 m 个同步周期的时钟偏差累积值, adj 为第 m 个同步周期计算的频率补偿值。最终从时钟根据式 (6)、式 (7) 计算出 adj 并调用 Linux 系统内核函数 adjtimex() 补偿系统频率, 实现从时钟与主时钟的时钟频率一致。

2.3 影响时钟同步精度的因素分析

影响时钟同步精度的因素分析如下:

1) 获取时间戳的位置

在一般情况下, 各时钟设备的通信报文甚至同一种报文从应用层到物理层经历的时间各不相同, 其主要受操作系统和协议栈的延时及抖动的影响,

因此, 在最靠近物理层的位置获取时间戳, 时钟同步精度最高。本文在无硬件支持的条件下, 尽量补偿主-从时钟的出边界时间, 最终提高同步精度。

2) 时钟晶振的不稳定性^[16]

主-从时钟的时钟晶振不可能保持完全一致, 因此, 从时钟需要不断地进行频率补偿来与主时钟同步, 在时钟同步系统中, 选择晶振稳定可靠的时钟以及进行频率补偿至关重要。

3) 链路延迟的不对称性

如 1.1 节讨论的同步原理假定链路延迟是对称的, 而实际链路延迟是不对称的。文献[17]采用加权平均求链路延迟来减小不对称, 但对本文系统效果不明显, 而且网络中间设备如路由器、交换机等会加大链路的不对称性, 因此, 可以考虑使用边界时钟和透明时钟降低链路不对称性。

4) 同步周期

同步周期越短, 说明主从时钟越频繁地进行时钟同步, 但同时也会占用更多的网络带宽, 因此, 同步周期并非越短越好, 应视网络情况而定。

3 时钟同步性能测试

实验在 2 台计算机上进行, 2 台 PC 均安装 Ubuntu14.04 LTS 系统, 其中, PC1 的处理器为 Intel(R) Core(TM) i5-2450M, PC2 的处理器为 Intel(R) Core(TM) i5-3210M, 主频均为 2.5 GHz。当 2 台 PC 上电启动后, 配置 PC1 的网卡地址: #ifconfig eth0 192.168.1.1 netmask 255.255.255.0; 配置 PC2 的网卡地址: #ifconfig eth0 192.168.1.2 netmask 255.255.255.0, 2 台 PC 同时运行 PTPd2 软件, 由最佳主时钟算法决策出一台 PC 为主时钟, 一台 PC 为从时钟。

本文分别配置 P 、 I 值、同步周期、出边界时间并连接不同中间设备进行实验, 以验证不同参数值及中间设备对同步精度的影响, 实验结果由从时钟输出主-从时钟偏差, Matlab 软件绘图。

3.1 不同 P 、 I 值对同步精度的影响

本节在同步周期为 1 s、未补偿出边界时间、主-从时钟设备通过网线直连条件下, 设置不同 P 、 I 值进行实验, 验证不同 P 、 I 值对同步精度的影响。

P 、 I 值为第 2.2 节讨论的比例积分控制器的比例系数和积分系数, 主要用于计算从时钟相对于主时钟的频率补偿值。不同的 P 、 I 值能达到的同步精度不同, 根据比例积分控制器原理, P 值和 I 值分别在 1~10、1~1 000 之间取值, 通过进行大量实验, 选取最佳 P 、 I 值。限于篇幅, 本文最终确定 3 组较为

典型的 P 、 I 值来对比同步精度, 实验结果如图 7~图 9 所示。

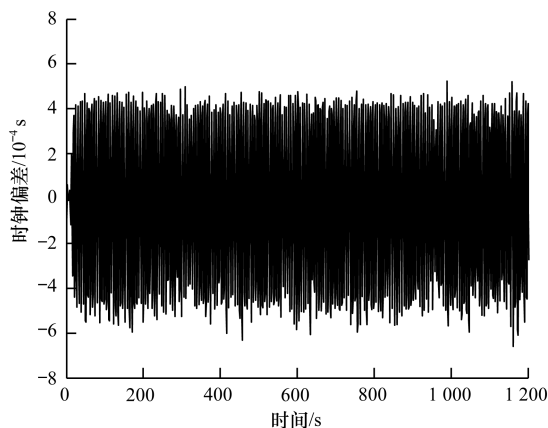


图 7 P 、 I 值为 1、1 时的同步精度实验结果

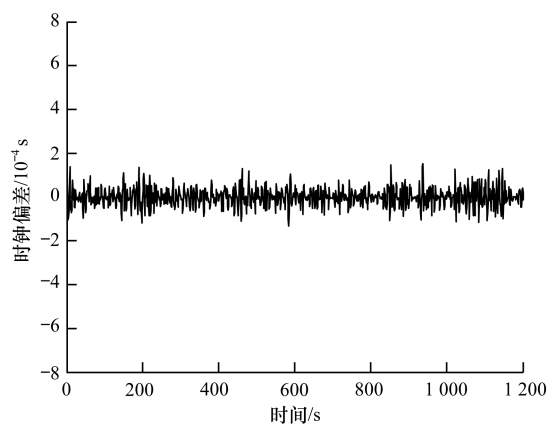


图 8 P 、 I 值为 2、16 时的同步精度实验结果

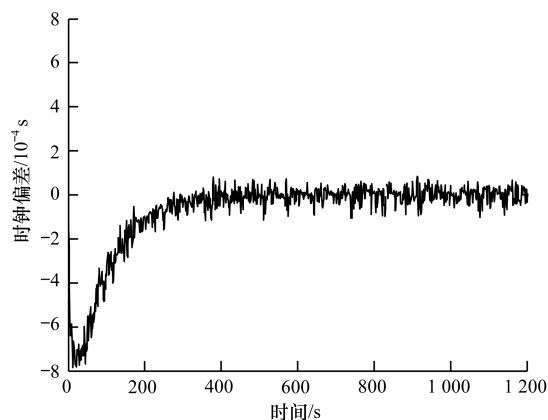


图 9 P 、 I 值为 10、1 000 时的同步精度实验结果

由图 7~图 9 可以看出, P 、 I 值为 1、1 时同步效果最差, 不仅时钟偏差较大而且抖动厉害; P 、 I 值为 10、1 000 是 PTPd2 系统默认值, 收敛速度较慢, 需要 5 min 以上才能稳定。本文先不考虑积分系数 I , 使 P 在 1~10 之间取值, 由第 2.2 节分析及实验结果可知, P 值越小响应速度越快, 但同时时钟偏差抖动也较大, 最终综合收敛速度和同步精度, 选定 P 为 2; 然

后加入积分系数 I , 使 I 在 $1 \sim 1\,000$ 之间取值, 先以 100 为梯度选取合适 I 值的范围, 再分别以 10 和 1 为梯度精确 I 值。通过实验可知, 当 I 为 16 时系统在 1 min 左右收敛, 并且同步精度较高。因此, P 、 I 值为 2、16 是较适合 PTPd2 系统的参数值, 表 1 为不同 P 、 I 值下的统计结果, 其中平均值是指主-从时钟偏差的绝对值平均值(下文同), 标准差反映了系统同步效果的稳定性, 因此当 P 、 I 值分别为 2 和 16 时, 同步精度约为 $30\ \mu\text{s}$ 。

表 1 不同 P 、 I 值下的同步精度分析结果

P 、 I 值	平均值/ μs	标准差/ μs	最大偏移值/ μs
1、1	299.269	342.190	-666.220
2、16	30.518	40.117	151.895
10、1 000	83.737	162.713	-783.914

3.2 不同同步周期对同步精度的影响

基于第 3.1 节实验结果, 在 P 、 I 值为 2 和 16、未补偿出边界时间、主-从时钟设备通过网线直连条件下, 设置不同的同步周期进行实验, 验证不同同步周期对同步精度的影响。

分别设置同步周期为 0.125 s、0.5 s、1 s、2 s 和 4 s, 统计结果如表 2 所示。

表 2 不同同步周期下的同步精度分析结果

同步周期/s	平均值/ μs	标准差/ μs	最大偏移值/ μs
0.125	27.697	35.929	-884.558
0.500	24.939	31.834	172.975
1.000	30.518	40.117	151.895
2.000	42.232	52.598	-154.193
4.000	194.549	220.523	526.492

由表 2 可知, 同步周期为 0.5 s 时, 精度最高, 但同步周期为 0.125 s 时精度变差, 主要是因为主-从时钟频繁进行时钟同步, 占用网络带宽, 造成网络拥堵; 而同步周期为 2 s 和 4 s 时的精度更差, 主要因为主-从时钟频率不一致, 同步周期越长, 累积误差越大, 因此, 折中考虑推荐同步周期设为 1 s。

3.3 补偿出边界时间对同步精度的影响

基于第 3.2 节实验结果, 在 P 、 I 值为 2 和 16、同步周期为 1 s、主-从时钟设备通过网线直连条件下, 对未补偿和补偿出边界时间这 2 种情况进行实验对比, 验证补偿出边界时间对同步精度的影响。

根据第 2.1 节所述方法分别计算主-从时钟的出边界时间, 主时钟的出边界时间是 34 090 ns, 从时钟的出边界时间是 20 076 ns。分别进行未补偿和补偿出边界时间实验, 实验结果如图 10、图 11 所示。

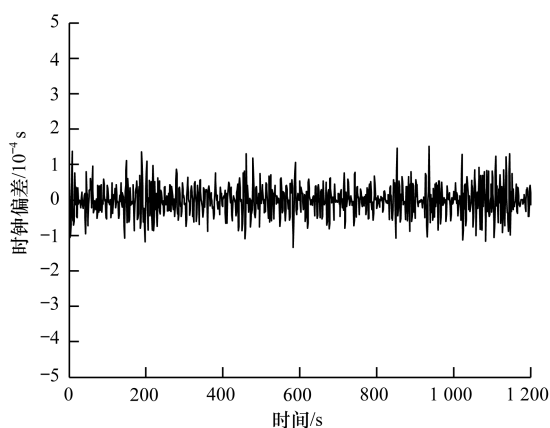


图 10 未补偿出边界时间实验结果

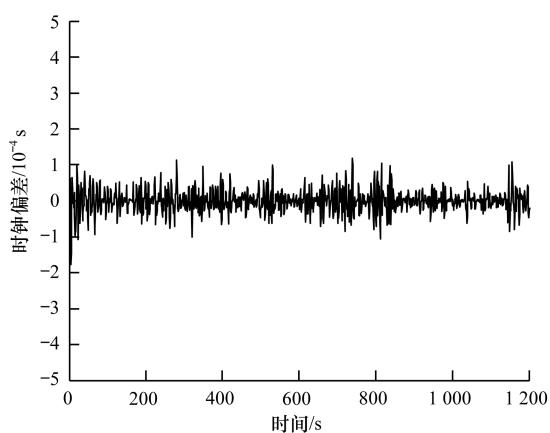


图 11 补偿出边界时间实验结果

补偿出边界时间后的主-从时钟偏差平均值为 $19.621\ \mu\text{s}$, 标准差为 $26.019\ \mu\text{s}$, 最大偏移值为 $-104.443\ \mu\text{s}$ 。从折线图和统计结果可知, 相比未补偿出边界时间, 同步精度提高了 $10\ \mu\text{s}$ 左右, 且时钟偏差抖动变小。实验结果表明, 补偿出边界时间可以提高时钟同步精度。

3.4 不同中间设备对同步精度的影响

基于第 3.3 节实验结果, 本文在 P 、 I 值为 2 和 16、同步周期为 1 s、补偿出边界时间条件下, 在主-从时钟设备间连接不同中间设备进行实验, 验证不同中间设备对同步精度的影响。

网络中的中间设备对计算链路延迟有一定影响, 即第 2.3 节所述的增加链路延迟的不对称性, 进而影响到同步精度, 分别配置中间设备为路由器、交换机、网线, 统计结果如表 3 所示。

表 3 不同中间设备下的同步精度分析结果 μs

中间设备	平均值	标准差	最大偏移值
路由器	30.277	37.944	169.339
交换机	24.736	31.765	-178.772
网线	19.621	26.019	-104.443

从表3可以看出,路由器的同步精度最差,因为路由器是三层设备,根据路由表寻址转发数据,交换机是二层设备,依据MAC地址转发数据,所以路由器的延时要比交换机长,且精度低。因此,增加网络元件会降低同步精度,建议使用透明时钟来减小误差,提高精度。

3.5 同步效果综合分析

本文在主-从时钟设备通过网线直连条件下进行实验,但从时钟不进行时钟校准,并与第3.3节补偿出边界时间后的实验结果进行对比分析。分析结果如图12所示,从图12可以看出,当从时钟不调整本地时钟时,主-从时钟偏差会越来越大,其主要原因是主-从时钟的晶振频率不一致,因此,随着时间累积,误差逐渐增大。当从时钟进行时钟校准时,时钟偏差稳定在 $-105\text{ }\mu\text{s} \sim 100\text{ }\mu\text{s}$ 之间,并集中分布在零值附近。

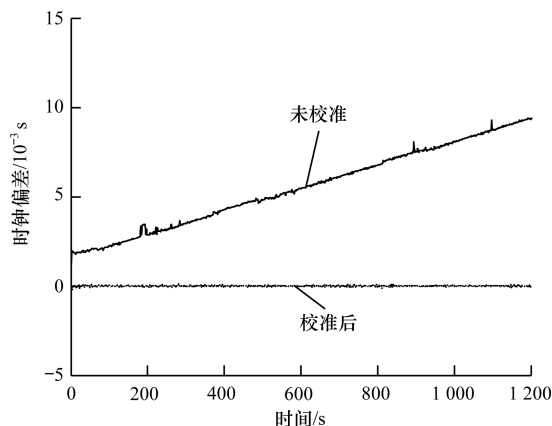


图12 从时钟未校准与校准后的时钟偏差对比

图13所示为进行时钟校准的主-从时钟偏差频率分布直方图,经分析可知,时钟偏差大致服从一个期望为 $\mu = 10 \times 10^{-9}\text{ s}$ 、标准差为 $\sigma = 2.601 \times 10^{-5}\text{ s}$ 的正态分布,即 $N(10 \times 10^{-9}, 6.769 \times 10^{-10})$ 。

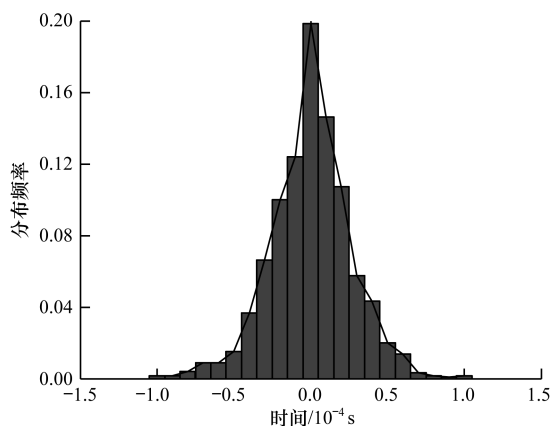


图13 时钟偏差频率分布直方图

通过综合分析可知,未进行时钟校准的从时钟

与主时钟的同步误差会随着时间累积达到毫秒级甚至秒级,而进行时钟校准的从时钟与主时钟的同步误差在 $100\text{ }\mu\text{s}$ 以内,平均误差为 $19\text{ }\mu\text{s}$,由此可以看出,本文系统的同步效果显著。

4 结束语

本文基于PTPd2代码在不需要硬件辅助的条件下实现IEEE 1588协议,通过实验选取最佳 P 、 I 值优化同步效果,并设置不同的同步周期进行实验,通过综合网络负载和同步精度,最终选定同步周期为 1 s ,分别补偿主-从时钟的出边界时间,以提高同步精度,增加网络元件,验证中间设备对同步精度的影响。实验结果表明,当最佳时钟同步精度为 $19\text{ }\mu\text{s}$ 时,本文方法可以满足大部分分布式控制系统的要求。

参考文献

- [1] Cisco Systems, Inc. Network time protocol; best practices [EB/OL]. (2008-12-17). [2018-03-07]. <https://www.cisco.com/c/en/us/support/docs/availability/high-availability/19643-ntpm.html>.
- [2] IEEE. Std 1588-2008 IEEE standard for a precision clock synchronization protocol for networked measurement and control systems; 978-0-7381-5400-8 [S]. New York, USA: IEEE Instrumentation and Measurement Society, 2008.
- [3] YANG X, ZHANG S, LI H. Research and implementation of precise time synchronization system of microgrid based on IEEE 1588 [C]//Proceedings of Advanced Information Management, Communicates, Electronic and Automation Control Conference. New York, USA: ACM Press, 2017: 258-261.
- [4] 钱美, 韩江桂, 尹为民. 船舶综合电力系统同步测量的实现[J]. 电力自动化设备, 2010, 30(11): 101-104.
- [5] 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.
- [6] 杜慧琦, 刘玥, 刘博, 等. IEEE1588v2 时钟同步技术在战术信息系统中的应用[J]. 计算机应用与软件, 2017, 34(11): 125-129.
- [7] 苏宇, 胡珩, 张涛, 等. 基于PTPd的精准时钟同步技术的研究[J]. 计算机技术与发展, 2016, 26(1): 175-180.
- [8] CHEN W, SUN J, ZHANG L, et al. An implementation of IEEE 1588 protocol for IEEE 802.11 WLAN [J]. Wireless Networks, 2015, 21(6): 2069-2085.
- [9] 陈翔. 精确时间同步协议的软件模型研究[D]. 昆明: 昆明理工大学, 2016.
- [10] CORRELL K, BARENDT N, BRANICKY M. Design considerations for software only implementations of the IEEE 1588 precision time protocol [C]//Proceedings of Conference on IEEE 1588 Standard for A Precision Clock Synchronization Protocol for Networked Measurement and Control Systems. Zurich, Switzerland: IEEE Press, 2006: 11-15.

(下转第59页)

4 结束语

本文提出的服务链部署策略主要从用户和服务提供商两方面的角度考虑,在满足用户对服务时延和可用性要求的基础上,使运营商的资源消耗最小。同时,设计了基于遗传算法和禁忌搜索相结合的启发式算法,该算法保留了遗传算法的全局优化性能和禁忌搜索算法的记忆功能,可降低时间复杂度,提高部署效率,从而完成服务链的在线部署。实验结果表明,本文算法在保证时延和可用性的同时,提高服务链部署成功率和网络资源利用率。但本文在考虑服务质量时,仅运用到时延和可用性2个方面,没有考虑到数据传输速率和抖动等 QoS 参数,下一步将研究上述参数对服务部署的影响。

参考文献

- [1] Networkfunction virtualization white paper [EB/OL]. [2017-05-09]. [http://portal.etsi.org/NFV/NFV White Paper.pdf](http://portal.etsi.org/NFV/NFV%20White%20Paper.pdf).
- [2] European Telecommunications Standards Institute. Network function virtualization architectural framework [EB/OL]. [2017-05-02]. http://www.etsi.org/deliver/etsi_gs/NFV/001099/002/01.01.01_60/gs_NFV-002v010101p.pdf.
- [3] BHAMARE D, JAIN R, SAMAKA M, et al. A survey on service function chaining[J]. Journal of Network and Computer Applications, 2016, 75(C): 138-155.
- [4] BRONSTEIN Z, ROCH E, XIA J, et al. Uniform handling and abstraction of NFV hardware accelerators[J]. IEEE Network, 2015, 29(3): 22-29.
- [5] PALKAR S, LAN C, HAN S, et al. E2: a framework for NFV applications [C]//Proceedings of Symposium on Operating Systems Principles. New York, USA: ACM Press, 2015: 121-136.
- [6] MEHRAGHDAM S, KELLER M, KARL H. Specifying and placing chains of virtual network functions [C]//Proceedings of IEEE International Conference on Cloud Networking. Washington D. C., USA: IEEE Press, 2014: 7-13.
- [7] XIONG G, HU Y X, TIAN L, et al. A virtual service placement approach based on improved quantum genetic algorithm[J]. Frontiers of Information Technology and Electronic Engineering, 2016, 17(7): 661-671.
- [8] CLAYMAN S, MAINI E, GALIS A, et al. The dynamic placement of virtual network functions [C]//Proceedings of Network Operations and Management Symposium. Washington D. C., USA: IEEE Press, 2014: 1-9.
- [9] VIZARRETA P, CONDOLUCI M, MACHUCA C M, et al. QoS-driven function placement reducing expenditures in NFV deployments [C]//Proceedings of IEEE International Conference on Communications. Washington D. C., USA: IEEE Press, 2017: 1-7.
- [10] KIM T, KIM S, LEE K, et al. A QoS assured network service chaining algorithm in network function virtualization architecture [C]//Proceedings of IEEE International Symposium on Cluster, Cloud and Grid Computing. Washington D. C., USA: IEEE Press, 2015: 1221-1224.
- [11] KIM S, PARK S, KIM Y, et al. VNF-EQ: dynamic placement of virtual network functions for energy efficiency and QoS guarantee in NFV [J]. Cluster Computing, 2017, 20(3): 2107-2117.
- [12] CHEN Y M, PENG Y J. A QoS aware services mashup model for cloud computing applications [J]. Journal of Industrial Engineering and Management, 2012, 5(2): 457-472.
- [13] 谢凡, 李勇, 柳嘉强, 等. 基于 SDN-NFV 的移动核心网联动机制研究[J]. 计算机工程, 2017, 43(2): 144-149.
- [14] MISHRA M, SAHOO A. On theory of VM placement: anomalies in existing methodologies and their mitigation using a novel vector based approach [C]//Proceedings of IEEE International Conference on Cloud Computing. [S. l.]: IEEE Computer Society, 2011: 275-282.
- [15] 徐宁, 李春光, 张健, 等. 几种现代优化算法的比较研究[J]. 系统工程与电子技术, 2002, 24(12): 100-103.

编辑 索书志

(上接第53页)

- [11] 宋升金, 许永辉, 姜守达. 基于 PTPd 改进的高精密时钟同步实现[J]. 计算机工程与应用, 2013, 49(2): 72-75.
- [12] 朱望纯, 钟震林, 覃斌毅. 嵌入式 Linux 设备的高精度 IEEE1588 时钟同步实现[J]. 计算机测量与控制, 2014, 22(5): 1619-1622.
- [13] 徐立, 赵平. 基于 IEEE 1588 精确时钟同步软件实现[J]. 现代电子技术, 2010, 33(24): 47-50.
- [14] 吕青松, 吴宁. 基于 IEEE1588 协议的高精度网络时钟同步软件设计[J]. 南京师范大学学报(工程技术版), 2013, 13(4): 64-69.
- [15] 庄晓燕, 王厚军. 基于卡尔曼滤波器的 IEEE 1588 时钟同步算法[J]. 电子测量与仪器学报, 2012, 26(9): 747-751.
- [16] LI D, WANG G, YANG C, et al. IEEE 1588 based time synchronization system for a seafloor observatory network[J]. Journal of Zhejiang University-Science C, 2013, 14(10): 766-776.
- [17] 李学桥, 陈园, 梁爽. 基于 IEEE1588 协议的精确时钟同步算法改进[J]. 计算机工程与科学, 2011, 33(2): 42-45.

编辑 索书志