****

**专业学位硕士学位论文**

**题 目 IEEE1588高精度时钟同步技术的改进与实现**

**作 者 王逸凡 完成日期 2021 年 3 月 10 日**

**专业（领域） 电子与通信工程**

**研 究 方 向 电子信息系统设计**

**指 导 教 师 彭良福教授、刘彬工程师**

**学 院 电子信息学院**

**授予学位日期 年 月 日**

原创性声明

本人郑重声明：本学位论文成果是本人在西南民族大学读书期间在导师指导下取得的，论文成果归西南民族大学所有。除了文中特别加以标注和致谢的地方外，论文中不包含其他人已经发表或撰写过的研究成果，也不包含为获得西南民族大学或其他教育机构的学位或证书而使用过的材料。与我一同工作的同志对本研究所做的任何贡献均已在论文中作了明确的说明并表示谢意。

作者签名：

日期：　　　年　 月　 日

学位论文版权使用授权书

本人同意学校根据《中华人民共和国学位条例暂行实施办法》等有关规定保留本人学位论文并向国家有关部门或资料库送交论文纸件或电子版本，允许论文被查阅和借阅；本人同意西南民族大学可以将论文的全部或者部分内容编入有关数据库进行检索，可以采用影印、缩印或者其它复制手段和汇编学位论文，同时授权中国学术期刊电子杂志社等单位将本论文收录到《中国优秀硕士学位论文全文数据库》、《中国博士学位论文全文数据库》等数据库，并通过网络向社会公众提供信息服务。

导师签名：　　 　　 　 作者签名：

日期：　　　 年　 月　 日 日期：　　　年　 月　 日

# 摘 要

第三次工业革命的推动加快了信息化产业的发展，彻底改变了社会的运作模式，分布式系统架构已经成为当今众多领域的主要发展方向，系统中各个节点分工协作共同完成复杂的任务。为了保证任务运行和数据采集与交互的顺序，各节点需要有相同的时间基准。通过卫星授时的方式可以得到纳秒级的授时精度，但设备建设成本高，并且卫星信号易受干扰，不适合大规模的使用。传统的NTP协议的同步精度最高为毫秒级，已经无法满足现在电信、工控等领域的精度要求，而IEEE 1588协议可以达到亚微秒级的同步精度，故可以在该协议的基础上优化细节将同步精度提高到纳秒级以满足当今各领域的同步要求。

本文对IEEE 1588协议的基础概念进行了深入分析，在此基础上研究了影响同步精度的因素，针对这些因素，以AM3358和AR8031\_AL1A为核心设计并实现了基于IEEE 1588协议的时钟同步模块，利用Linux系统下的PTP硬件时钟子系统和套接字设置选项使得应用程序能获得PTP报文的硬件时间戳，并采用卡尔曼滤波消除同步过程中的系统误差，同时将滤波后的时钟偏差做为PI控制器的输入，通过计算的频率的补偿值来调整本地时钟频率。通过设置不同的主从时钟、不同的中间设备进行测试实验。实验表明，在内设铷原子钟的时频同步分析仪作为主时钟，时钟同步模块作为从时钟的情况下，主从时钟在网线直连状态下的同步精度可以达到±8ns左右。

针对非对称延时对同步精度的严重影响，本文使用DAC模型来检测并补偿时钟偏差的离群值，提高了高网络负载下的系统稳定性及时钟同步精度。此外，为了进一步优化DAC模型，本文提出一种时钟模型线性估计的方法，该算法通过估计当前同步周期时间戳的数值来对时钟偏差进行估计，在解决DAC模型弊端的同时，保证了时钟同步的精度。

**关键词：**IEEE 1588；时钟同步；非对称路径延时；PTP硬件时钟

# Abstract

The promotion of the third industrial revolution accelerated the development of the information industry and completely changed the operation mode of society. Distributed system architecture has become the main development direction in many fields today, and each node in the system cooperates to complete complex tasks together. In order to ensure the sequence of task operation and data collection and interaction, each node needs to have the same time base. Nanosecond-level timing accuracy can be obtained through satellite timing, but the equipment construction cost is high, and satellite signals are susceptible to interference, which is not suitable for large-scale use. The synchronization accuracy of the traditional NTP protocol is at the highest millisecond level, which can no longer meet the accuracy requirements of the current telecommunications, industrial control and other fields, while the IEEE 1588 protocol can reach sub-microsecond synchronization accuracy, so the details can be optimized on the basis of this protocol. The synchronization accuracy is improved to nanosecond level to meet the synchronization requirements in various fields today.

This paper analyzes the basic concepts of IEEE 1588 protocol in depth, and studies the factors that affect synchronization accuracy on this basis. Aimin-g at these factors, AM3358 and AR8031\_AL1A are used as the core to design and implement a clock synchronization module based on IEEE 1588 protocol, using Linux The PTP hardware clock subsystem and socket setting options under the system enable the application to obtain the hardware time stamp of the PTP message, and use Kalman filtering to eliminate system errors in the synchronization process, and at the same time use the filtered clock deviation as PI The input of the controller adjusts the local clock frequency through the calcu-lated frequency compensation value. Test experiments by setting up different master-slave clocks and different intermediate devices. Experiments show that when the time-frequency synchronization analyzer with the built-in rubidium atomicclock is used as the master clock and the clock synchronization module is use-d as the slave clock, the synchronization accuracy of the master-slave clock in the state of direct connection of the network cable can reach about 8ns.

Aiming at the serious impact of asymmetric delay on synchronization accuracy, this paper uses the DAC model to detect and compensate for outliers of clock deviation, which improves the system stability and clock synchronization accuracy under high network load. In addition, in order to further optimize the DAC model, this paper proposes a method of linear estimation of the clock model. This algorithm estimates the clock deviation by estimating the value of the current synchronization period timestamp. While solving the shortcomings ofthe DAC model, it ensures the accuracy of clock synchronization.

**Keywords:** IEEE 1588; clock synchronization; path delay of asymmetry; PTP hardware clock

目 录

[摘 要 3](#_Toc66985712)

[Abstract 4](#_Toc66985713)

[第1章 绪论 8](#_Toc66985714)

[1.1 研究背景及意义 8](#_Toc66985715)

[1.2 国内外研究现状 9](#_Toc66985716)

[1.3 论文主要工作和章节安排 12](#_Toc66985717)

[1.3.1 论文的主要工作 12](#_Toc66985718)

[1.3.2 论文的章节安排 13](#_Toc66985719)

[第2章 IEEE 1588精确时钟同步协议 14](#_Toc66985720)

[2.1 IEEE 1588时钟模型 14](#_Toc66985721)

[2.1.1 普通时钟 14](#_Toc66985722)

[2.1.2 边界时钟 15](#_Toc66985723)

[2.1.3 透明时钟 16](#_Toc66985724)

[2.2 IEEE 1588协议报文 18](#_Toc66985725)

[2.2.1 PTP报文类型和格式 18](#_Toc66985726)

[2.2.2 PTP报文封装方式 21](#_Toc66985727)

[2.3 IEEE 1588同步原理 22](#_Toc66985728)

[2.3.1 基本原理描述 22](#_Toc66985729)

[2.3.2 延时请求响应测量机制 24](#_Toc66985730)

[2.3.3 对等延时测量机制 26](#_Toc66985731)

[2.4 时钟同步精度影响因素分析 27](#_Toc66985732)

[2.4.1 非对称路径延时对同步精度的影响 27](#_Toc66985733)

[2.4.2 时间戳获取位置对同步精度的影响 28](#_Toc66985734)

[第3章 时钟同步模块的系统设计 30](#_Toc66985735)

[3.1 时钟同步模块整体框架 30](#_Toc66985736)

[3.2 主控芯片的选择 31](#_Toc66985737)

[3.2.1 AM3358简介 31](#_Toc66985738)

[3.2.2 AM3358硬件时钟的支持 32](#_Toc66985739)

[3.3 以太网PHY模块硬件设计 33](#_Toc66985740)

[3.4 PTP软件平台设计 34](#_Toc66985741)

[3.4.1 Linux操作系统 34](#_Toc66985742)

[3.4.2 Linux PTP简介 36](#_Toc66985743)

[3.4.3 PTP硬件时钟 37](#_Toc66985744)

[3.4.4 硬件时间戳的获取 40](#_Toc66985745)

[第4章 时钟同步模块的算法设计 42](#_Toc66985746)

[4.1 卡尔曼滤波算法 42](#_Toc66985747)

[4.1.1 时钟状态的建模 42](#_Toc66985748)

[4.1.2 滤波过程 43](#_Toc66985749)

[4.2 时钟频率补偿算法 45](#_Toc66985750)

[4.3 时钟同步精度测试 46](#_Toc66985751)

[4.3.1 标准符合测试 47](#_Toc66985752)

[4.3.2 时钟同步模块同步精度测试 49](#_Toc66985753)

[第5章 非对称路径延时校正算法 52](#_Toc66985754)

[5.1 非对称路径延时校正模型 52](#_Toc66985755)

[5.1.1 模型描述 52](#_Toc66985756)

[5.1.2 理论验证与同步精度测试 55](#_Toc66985757)

[5.2 时钟模型线性估计算法 57](#_Toc66985758)

[5.2.1 算法描述 57](#_Toc66985759)

[5.2.2 算法验证与同步精度测试 61](#_Toc66985760)

[第6章 总结与展望 63](#_Toc66985761)

[6.1 总结 63](#_Toc66985762)

[6.2 展望 64](#_Toc66985763)

[致谢 65](#_Toc66985764)

[参考文献 67](#_Toc66985765)

# 绪论

## 1.1 研究背景及意义

如今信息技术迅速发展，网络系统规模日益增大，分布式网络的部署越来越广泛。在现代的分布式系统中，各节点能够独立地或彼此联合地执行复杂的任务，由于分布式网络的特性，各个节点通过可靠的共享通信介质来进行实时的数据交换，而时间是交互数据过程中不可或缺的重要部分，时钟同步也是许多应用程序工作的前提[1]。但是各节点通常只会维护本地的时钟，并且这些本地时钟在运行时会因不同的时钟频率、不同的温度等因素导致彼此的时钟出现差异，所以必须要对各节点进行时钟同步的操作，否则，拥有不同时间标准的各节点在进行通信时会导致交互数据的延时，若时间偏差过大，会使得系统发生故障，甚至导致崩溃[2]。

现阶段网络系统逐渐庞大复杂，系统分布化的程度也越来越高，对时钟同步的精度要求也逐步提高。因此，时钟同步技术作为节点通信的绝对必要项也受到了广泛地关注，电力、航天、导航、工业物联网、通信等诸多领域都需要使用此技术来确保网络中各节点的实时性和确定性[3]。在电力和一些能源供应领域，时钟同步对电网稳定性地控制和停电分析至关重要。在智能工厂中，生产线上相互协作的生产机器人必须要精确同步，当多个电机移动同一个机械负载时，时钟同步起着决定性作用[4]。此外，在基于时分多址（TDMA）的通信中，每个设备被给予独立的传输时隙，设备之间同步的精度越高，就能越精确地依附时隙，从而更好地利用带宽[5]。目前，我国目前处于全球5G发展的第一梯队，在5G技术及产业发展上取得了骄人的成绩，5G已经正式商用的同时也催生了许多新的业务，比如大规模物联网、VR/AR、自动驾驶等，这些新业务可能会具备高精度同步需求。由此可见，高精度时钟同步技术作为5G同步网络的大规模部署与实现的基础支撑，将在同步网络中起着举足轻重的作用[6]。

目前的分布式通信网络主要采用卫星的授时信号作为时钟基准，以此来满足通信的时钟同步需求，而地面同步组网只是作为备用方式以防止卫星接收机出现故障或卫星信号不稳定的情况[7]。卫星授时具有同步时间短、精度高，使用方便等优点，但缺点也很明显，例如某些环境场景难部署，抗干扰能力差等，而且如今网络的部署规模越来越大，必须要考虑卫星接收设备和运维的成本问题[8]。同时，如今卫星信号受到干扰导致异常的次数越来越多，卫星信号被攻击的情况也时有发生。最近，欧洲的伽利略导航系统部分卫星播发的时间信息出现异常跳变，导致其系统时间发生异常，持续了3小时之久，该异常会影响伽利略系统的定位、导航和授时性能，因此，若时钟同步完全依赖于卫星授时，将会带来极大的安全隐患[9]。因此，在卫星信号降质的情况下，通过建立的高精度的地面同步网络进行传递高精度的时钟频率信号，在现有的技术基础上优化细节提升同步精度，以此保证通信网内各节点的同步要求。卫星授时和地面同步网这两种时钟同步方式将会形成“天地互备”的关系，两者相互共存，相互补足[10]。

在选择时钟同步技术时，需要考虑其能够实现的精度、网络建设的成本、同步算法的实现难度等方面。目前，已经较为广泛地投入应用的时钟同步技术主要有网络时间协议（NTP）和IEEE 1588协议。NTP协议采用软件来实现，其实现精度较低，无法达到现在网络时钟同步的精度要求[11]。而一些支持IEEE1588协议的设备，其时钟同步精度可以达到±30ns，并且基于IEEE 1588协议的地面同步组网在4G网络时代已有部署经验[12]。鉴于IEEE 1588协议的成熟、安全可靠、应用规模大、成本相对较低的特点，研究基于IEEE 1588协议的时钟同步技术更为合适。

## 1.2 国内外研究现状

IEEE 1588简称为PTP（Precision Time Protocol）协议，即精确时钟协议。该协议自被发布以来，已经发展出三个版本，IEEE 1588-2002（IEEE 1588v1.0）版本是最初发布的版本，主要用于局域网里多播的通信方式[13]。IEEE 1588-2008（IEEE 1588v2.0）版本应对的是网络拓扑结构更为复杂的网络环境，该版本也是目前受到设备支持和应用最广泛的版本[14]。欧洲核子研究组织（CERN）提出了一种新型的时钟同步技术—白兔子（White Rabbit）技术，该技术以千兆以太网为基础，使用同步以太网（SyncE）技术实现时钟频率同步，使用IEEE 1588v2.0实现时间同步，同时使用全数字双混频鉴相器（DDMTD）提高时间戳精度，可达到200ps的精度[15]。然而，该技术最大的缺点是在终端节点和交换机上需要昂贵的硬件支持。WR技术中的一些概念被引入了IEEE 1588-2019草案，例如物理层相位同步、DDMTD技术等，IEEE1588-2019（IEEE 1588 v2.1）版本已于2020年6月发布，该版本可以和1588-2008版本系统一起运行，但有一定的限制，并且不再兼容1588-2002版本[16, 17]。

自IEEE 1588协议发布之日起，吸引了各大组织、研究院对其进行研究和开发，涉及了电信网络、自动控制以及工业控制等多个领域。G.Giorgi等人提出了一种将PTP和卡尔曼滤波相结合的方法，以补偿各种不确定性带来的误差。作者强调了时钟偏差和时钟漂移估计的准确性、时间戳交换的间隔和从时钟的稳定性对同步精度的影响，并分析了这些因素的影响和相互作用[18]。同时提出了一种由两种卡尔曼滤波器组成的组合算法。第一种卡尔曼滤波器具有检测离群值的功能，如果参考的时间源发生故障，当前同步产生的离群值过多，则启用第二种卡尔曼滤波器进行处理。该方法提高了鲁棒性，取得了较好的精度[19]。N.Kero等人分析了可能对PTP准确性产生负面影响的因素，并使用硬件时间戳实现了100ns-1µs的精度。作者使用卡尔曼滤波器作估计，通过彼此独立的路径将几个时钟源连接到从时钟。然后，比较各时间源与从时钟的时钟偏差，通过从时钟上的控制器选择精度较高的时钟源作为当前主时钟，其他时钟源作为备用时钟，由此解决时钟源或路径发生故障的问题，提高了同步的可靠性[20]。Reinhard Exel等人研究了用于同步的PI控制器的参数，指出时钟伺服对于同步的重要性，将基于加法器的时钟或压控振荡器与PI控制器结合使用。作者研究了影响时钟控制的变量，并在此基础上证明了合理的PI控制器参数是减小钟差的关键[21]。A.K.Karthik等人提出了一种提高PTP对非对称时延的鲁棒性的方法，解决了在传输路径的延迟未知的情况下，时钟偏移和频率漂移估计的建模问题和网络攻击[22]。Puttnies H.等人介绍了一种提高PTP同步精度的PTP-LP方法，并表明该方法对分组延迟的变化具有很强的鲁棒性。PTP-LP使用精确的硬件时间戳来估计时钟偏移和漂移。在时钟稳定性和分组时延的不同分布方面，将PTP-LP与标准PTP在不同条件下进行了比较。结果表明，当使用稳定的HW时钟且网络中存在未知的、不可忽略的分组延迟时，PTP-LP达到最优效果[23]。

庄晓燕提出了基于时钟群的主时钟选择算法，并表明在主时钟失效时，该算法能够降低对后续时钟同步的精度影响。同时，针对环境对时钟漂移的影响，提出了基于二阶卡尔曼滤波器的同步算法，在同步过程中能够自适应地选择时钟模型[24]。

孔祥瑞对PTP中非对称路径延时进行了详细分析，提出了一种基于三层BP神经网络的时钟偏差补偿方案，首先将主从时钟偏差输入到神经网络模型，通过该模型的输出结果来调整时钟偏差，以此来提高PTP非对称路径延时的同步精度[25]。

史仲渊详细描述了影响同步精度的主要因素，并提出了频率补偿方案，以此减小由频率漂移引起的同步误差。利用WireShark软件抓取报文到达MAC层的时间戳，以此来补偿软件时间戳，使其更接近于硬件时间戳，减少因网络协议栈延时引起的同步误差。此外，引入了改进的卡尔曼滤波算法，对时钟偏差进行滤波的同时，减小离群值对卡尔曼滤波精度的影响，提高了同步的鲁棒性[26]。

陶稳静采用FIR和IIR滤波器对主从时钟偏差和链路延时进行滤波，并且引入PI控制器补偿主从时钟间的频率偏差，实现时钟同步。作者设计了两种方法来实现IEEE 1588协议。网络层实现方法是基于开源代码PTPd2采用纯软件的方式来实现；MAC层方法采用硬件方式实现，采用STM32F407作为主芯片，DP83848为物理层芯片，由硬件自动获取报文到达MII／RMII接口的时间戳，以此实现时钟同步。最终网络层实现方法的同步精度可以达到微秒级，MAC层实现方法可以达到纳秒级[27]。

除了对IEEE 1588协议的学术研究，国内外的很多公司和企业也生产了支持IEEE 1588协议的产品。

美国国家半导体公司早在2007年就生产出一款支持IEEE 1588协议的PHY芯片—DP83640，它可以在物理层检测协议报文，获取并且更新时间戳，可以达到20ns以内的精度[28]。More Than IP 公司推出了支持IEEE 1588协议的MAC模块，该模块可以在MAC层获取时间戳。美国Microsemi公司生产出一种高融合度的时钟卡设备，该设备支持同步以太网和IEEE 1588v2协议，以达到频率同步和时间同步[29]。Renesas公司通过自研的时间设备获取PTP报文时间戳，并将其同步到PTP硬件时钟子系统的时间戳单元，提高报文时间戳的精度，从而提升时钟同步精度[30]。Vitesse公司产出了一种内置有CPU和IEEE l588协议的交换芯片，运行相应的应用程序即可进行时钟同步，方便调试[31]。目前Texas Instruments、Intel、Boardcom、Mavell等大公司也都相继推出了支持IEEE 1588协议和多路恢复时钟的PHY芯片[32]。

国内，华为针对城轨交通时钟同步应用提出了一种基于IEEE 1588v2的增强型地铁传输系统。该系统在地铁控制中心部署时钟服务器，外接卫星接收机接收来自GPS/北斗卫星系统的高精度时钟信号，通过PTP或1PPS+TOD时间接口以及SyncE时钟接口给传输设备授时，传输设备通过PTP接口为每个基站提供高精度时钟，从而实现全网高精度时钟同步[33]。

中兴通讯基于自研芯片，在PCS层进行PTP时间戳的处理，提高了时间戳的精度；采用时钟化的相位检测技术，以低频时钟获得较高精度的时间，并且对设备系统内的时间分发误差进行优化。综合以上技术，消除了PTP计算和系统分发引入的误差，设备的同步精度达到±5ns[34]。

北京酷鲨科技有限公司生产的P66/P68 1588时间同步单板支持基本的IEEE 1588协议，支持OC、BC或GC三种时钟模式，同时支持GNSS和PTP输入。该单板可以串接在现有数据链路中串接后能提供低延时、全透明的GE线速数据透传通道。由此特点，普通设备想要提高授时精度，只需在设备上行链路添加该板卡，板卡可以线性透传业务报文，其业务不受影响，从时钟可以穿过普通网络与主时钟达到100ns的同步精度[35]。

深圳市夏光时间技术有限公司的产品——SyncEdge时频同步分析仪是一款便携式时频同步专用测试仪表，内置GPS/BDS接收机以及铷原子钟，包括NTP，PTP等时间接口，可满足3G/4G/5G网络高精度时频同步的测试需求[36]。

## 1.3 论文主要工作和章节安排

### 1.3.1 论文的主要工作

IEEE 1588v2.0相较于IEEE 1588v1.0更加完善，同时也增加了多种新功能，例如改进了报文的数据结构，简化了报文格式、引入透明时钟模型、增加了新的网络路径延时测量方式等等。本论文主要针对IEEE 1588 v2.0协议，在其基础上研究网络通信中的时钟同步技术，提高网络系统中节点之间的同步精度。通过对影响时钟同步性能因素的分析，充分利用Linux系统内核开源的优势，通过其类驱动程序对PTP硬件时钟的抽象，获取当以太网报文通过MII/RMII接口时的时间戳，避免网络协议栈中的未知延时。同时在计算时钟偏差等时钟状态值时，加入滤波算法，以此降低报文在传输过程中引入的误差；并且使用PI控制器通过当前的时钟偏差来补偿时钟的频率偏差，提高时钟同步的精度。此外，当网络中流量增大，负荷严重时会导致往返路径延时的严重非对称，使得时钟偏差值过大或过小，极大程度的影响了同步的精度，故引入非对称延时校正算法保证该情况下时钟同步的精度。

### 1.3.2 论文的章节安排

第一章，介绍了论文的研究背景和意义，通过查阅大量的国内外相关的参考文献，阐述了论文的主要研究内容，结合实际情况，提出了一些方法来提高时钟同步的精度。

第二章，阐述了IEEE 1588协议的基本概念，包括IEEE 1588时钟模型、同步原理以及报文的类型与格式，分析了IEEE 1588v2协议两种测量路径延时的方式，最后介绍了影响时钟同步精度的因素以及避免的方法。

第三章，描述了时钟同步模块的整体设计，包括硬件平台的元器件的选型以及时钟同步软件平台的搭建，详细介绍了PTP硬件时钟子系统和用户空间获取硬件时间戳的方法。

第四章，介绍了时钟同步模块的算法设计，采用卡尔曼滤波消除同步过程中的系统误差，同时使用PI控制器计算频率补偿值，以此来调整本地时钟频率。设置不同的实验环境进行同步精度测试。

第五章，分析了非对称延时对同步精度的影响，提出了两种非对称路径延时校正算法，以此来抑制由高网络负载引起的非对称延时，最后通过实验测试验证算法的有效性。

第六章，总结了本文的研究成果以及展望进一步的工作和研究。

# IEEE 1588精确时钟同步协议

## 2.1 IEEE 1588时钟模型

IEEE 1588v2协议进行同步的概念之一为IEEE 1588v2时钟。在同步网络中使用几种不同类型的时钟，能够构建十分复杂的网络，以此满足不同配置的需求。IEEE 1588v2的时钟可分为三类：普通时钟，边界时钟，透明时钟[37]。

### 2.1.1 普通时钟

普通时钟（Ordinary Clock, OC）的时钟模型如图2.1所示，普通时钟可以位于网络的末端或网络的起始处，并且只存在一个PTP端口。

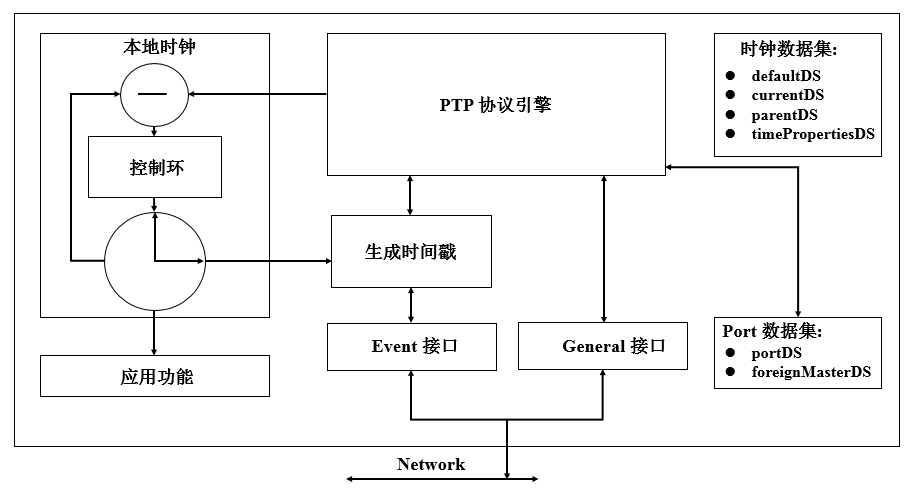


图2.1 普通时钟模型

Figure2.1 Model of OC

普通时钟的PTP端口包含事件接口（Event interface）和通用接口（General interface），事件接口用来处理PTP的事件报文和报文的时间戳信息，通用接口处理通用报文用于管理时钟设备状态。普通时钟既可以充当主时钟，也可以充当从时钟。在同一网络中，可以有多个主时钟和从时钟，但必须确定一个最高级别的时钟作为最佳主时钟，使其他时钟与之同步，这个过程通常是由最佳主时钟算法（Best Master Clock, BMC）完成的。由该算法选举出来的最佳主时钟作为普通时钟来使用，通常使用卫星信号对其进行授时，故该时钟拥有极高质量的时钟信号，可以为网络中其他时钟设备提供高质量的时间基准。

### 2.1.2 边界时钟

若存在多个从时钟，各个从时钟需要和最佳主时钟进行联系，因此在PTP时钟中引入了类似交换机网络的概念——边界时钟（Boundary Clock, BC），其时钟模型如图2.2所示。边界时钟拥有多个PTP端口，且每个PTP端口都和普通时钟的PTP端口类似。

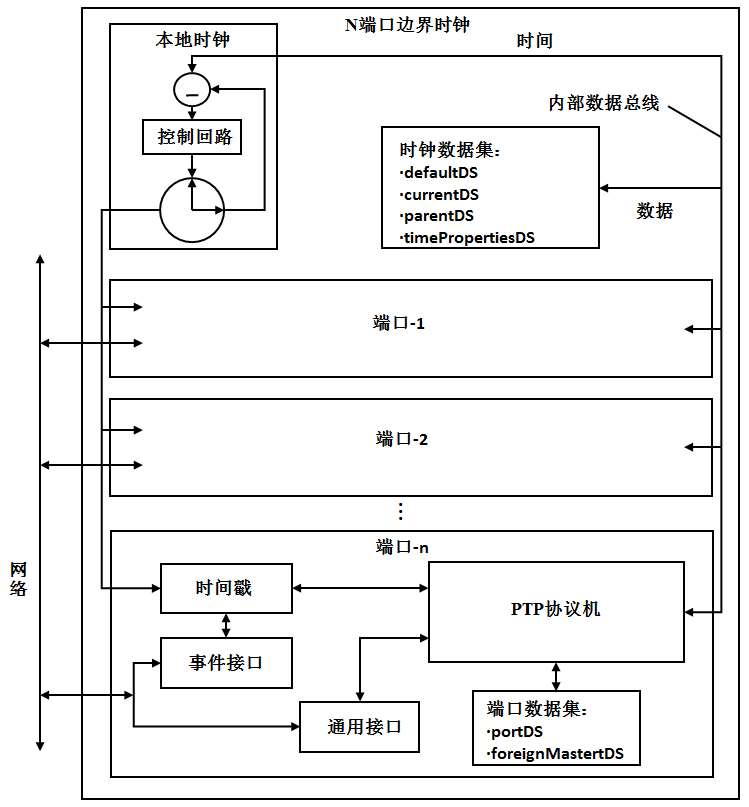


图2.2 边界时钟模型

Figure2.2 Model of BC

边界时钟的主要作用就是将它的上行设备与其下行的多个从属设备同步，实现不同PTP网络域的桥接，使得这些从属设备能获得整个PTP网络的最佳主时钟的时基，同步示意图如图2.3。边界时钟先作为PTP从时钟与最佳主时钟同步，然后，作为主时钟通过多个端口与其从属时钟同步，实现将高质量的时间基准从最佳主时钟传送到其它时钟。由此方式，可以建立较复杂的网络，满足不同的网络拓扑需求。



图2.3 边界时钟同步示意图

Figure2.3 Diagram of BC Synchronization

### 2.1.3 透明时钟

IEEE 1588v2协议引入了透明时钟来替代实现交换机功能的边界时钟[38]。透明时钟不会进行时钟同步，对于PTP拥有报文直接转发，对于事件报文将通过测量PTP报文在透明时钟内部的驻留时间，并将其添加到PTP报文里的校正域中，从时钟通过报文到达时间戳校正字段中的信息来确定报文延时，以此保证同步的精确性。由此，在进行网络拓扑时，并不强制在网络中使用边界时钟。透明时钟可分为端到端透明时钟(E2ETC, end-to-end transparent clock)和点到点透明时钟(P2PTC, peer-to-peer transparent clock)，针对点到点透明时钟，IEEE 1588v2协议增加了一种对等延时测量机制，以此计算任意两“点对点”时钟的路径延时，详细内容将在2.3小节介绍。

端到端透明时钟实现PTP报文透传，不会进行时钟同步。对于事件报文，测量其在内部的传输时间，修正时间戳[39]。此处介绍该透明时钟处理Sync报文的过程，如图2.4所示。



图2.4 端到端透明时钟处理Sync报文的过程

Figure2.4 Process of E2ETC Processing Sync Message

当Sync报文进入E2E透明时钟时记录报文的进入时间，离开时记录离开时间，将两个时间差作为Sync报文在时钟内部的驻留时间，并将其添加到报文的校正域中，以此确认准确的传输延时。

点到点透明时钟除了可以测量事件报文在其内部的驻留时间，并且可以计算与其相连时钟的路径延时。此处同样以Sync报文为例介绍点到点透明时钟对其处理过程，如图2.5所示。



图2.5 点到点透明时钟处理Sync报文的过程

Figure 2.5 Process of P2PTC Processing Sync Message

P2PTC将主从时钟间的路径分为两个链路，并参与计算与其相连时钟的路径延时，在转发Sync报文前将报文在自身内部的驻留时间与上行链路的路径延时之和添加到Sync报文的校正域中，随后将其转发给从时钟。

## 2.2 IEEE 1588协议报文

### 2.2.1 PTP报文类型和格式

PTP协议是基于主从时钟间同步报文的交互来实现的，PTP报文是同步过程中至关重要的部分。主从时钟记录同步报文的发送时间和接收时间，并将记录的时间戳放入报文对应的字节中。PTP报文是由报文头，报文主体和报文扩展字段等部分组成，PTP报文的组成格式如图2.6。

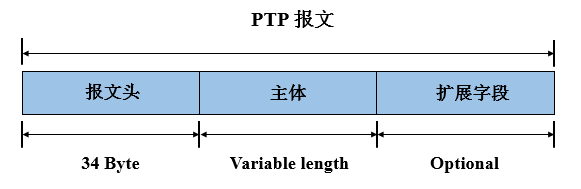


图2.6 PTP报文的组成

Figure2.6 Composition of PTP Message

PTP报文分为两大类：事件报文和通用报文，事件报文在发送和接收时都会生成准确的时间戳，通用报文不要求生成时间戳。事件报文集中包含：Sync报文，Delay\_Req报文，Pdelay\_Req报文和Pdelay\_Resq报文；通用报文集中包含：Announce报文，Follow\_up报文，Delay\_Resp报文，Pdelay\_Resp\_Follow\_up报文，Management报文和Signaling报文[40]。

Sync，Delay\_Req，Follow\_Up和Delay\_Resp报文用于生成和传递使用延时请求响应测量机制的普通时钟和边界时钟所需的时间戳信息，Sync报文和Delay\_Req报文的结构见表2.1。

表2.1 Sync和Delay\_Req报文结构

Table2.1 Structure of Sync and Delay\_Req Message

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 比特 | 八位位组 | 偏移 |
| 报文头 | 0-7 | 34 | 0 |
| originTimestamp | 0-7 | 10 | 34 |

Pdelay\_Req，Pdelay\_Resp和Pdelay\_Resp\_Follow\_Up报文用于测量使用对等延时测量机制的两个PTP端口之间的路径延迟，该路径延迟用于校正Sync和Follow\_Up报文中的时间戳信息。使用对等延时测量机制的时钟能够使用Sync和Follow\_Up消息中的时间戳信息以及已测量的路径延迟进行同步。Pdelay\_Req和Pdelay\_Resp报文的结构分别见表2.2和表2.3。

表2.2 Pdelay\_Req报文主体结构

Table2.2 Structure of Pdelay\_Req Message

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 比特 | 八位位组 | 偏移 |
| 报文头 | 0-7 | 34 | 0 |

|  |  |  |  |
| --- | --- | --- | --- |
| originTimestamp | 0-7 | 10 | 34 |

|  |  |  |  |
| --- | --- | --- | --- |
| reserved | 0-7 | 10 | 44 |

表2.3 Pdelay\_Resp报文主体结构

Table2.1 Structure of Pdelay\_Resp Message

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 比特 | 八位位组 | 偏移 |
| 报文头 | 0-7 | 34 | 0 |
| requestReceiptTimestamp | 0-7 | 10 | 34 |
| requestingPortIdentity | 0-7 | 10 | 44 |

在BMC算法中，Announce报文携带本地时钟的时钟级别、时钟精度等时钟属性信息，用于建立同步的层次结构。

Management报文处理对时钟的运维信令，主要用于查询或更新时钟本地的的PTP数据集。

Signaling报文作为备用报文，用于除以上报文用途外的其它目的的信息传递。例如，改变主从时钟的同步时间间隔。

PTP报文头对于所有的PTP报文都是通用的，其具体结构见表2.4。

表2.4 PTP报文头

Table2.1 Header of PTP Message

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 比特 | 八位位组 | 偏移 |
| messageType | 0-3 | 0.5 | 0 |
| transportSpecific | 4-7 | 0.5 | 0 |
| versionPTP | 0-3 | 0.5 | 1 |
| reserved | 4-7 | 0.5 | 1 |
| messageLength | 0-7 | 2 | 2 |
| domainNumber | 0-7 | 1 | 4 |
| reserved | 0-7 | 1 | 5 |
| flagField | 0-7 | 2 | 6 |
| correctionField | 0-7 | 8 | 8 |
| reserved | 0-7 | 4 | 16 |
| sourcePortIdentity 10 | 0-7 | 10 | 20 |
| sequenceId | 0-7 | 2 | 30 |
| controlField | 0-7 | 1 | 32 |
| logMessageInterval | 0-7 | 1 | 33 |

PTP通用报文头主要包含了：

1. 报文类型（MessageType）：不同的值代表不同的PTP报文。
2. 传输细节（transportSpecific）：根据映射不同的传输协议填入不同的值。
3. PTP版本号（versionPTP）：IEEE 1588v1版本此处值为1，v2版本此处值为2。
4. PTP域序列号（domainNumber）：对于普通时钟和边界时钟，此处值为各时钟维护的本地数据集中的相应数值。
5. 校正域（correctionField）：报文在透明时钟内部的驻留时间，点到点透明时钟的路径延时以及非对称延时的补偿值填入此处。
6. 源端口号（sourcePortIdentity）：对应发送端口的相关属性。
7. 序列号（sequenceId）：用来区分同一发送端口的多条同类型报文。
8. 控制域（controlField）：根据报文类型的值来取值。
9. 对数报文时间间隔（logMessageInterval）：包括Annonuce报文，Syn-c报文以及Delay\_Req报文的对数时间间隔，对应的取值是以2为底的对数。例如，此处值为0，则，表示当前报文发送速率为1秒1次。

### 2.2.2 PTP报文封装方式

IEEE 1588v2协议规定PTP报文可以采用不同的链路传输协议，例如IEEE802.3/Ethernet，UDP/IP等[41]。时钟同步过程中两时钟发送的PTP报文的封装方式必须相同，否则彼此无法响应。

目前PTP报文的封装方式主要为UDP/IPv4封装和IEEE802.3/Ethernet封装。

PTP报文的UDP/IPv4封装流程如下：

1. 应用程序在应用层填充PTP报文，随后发送到下一层。
2. 在传输层为PTP报文添加UDP报文头，将其封装成UDP报文。
3. 在网络层添加IPv4的IP报头，形成IP报文。
4. 在数据链路层将报文封装成以太网帧后进行传输。

以UDP/IPv4方式封装PTP报文的示意图如图2.7。

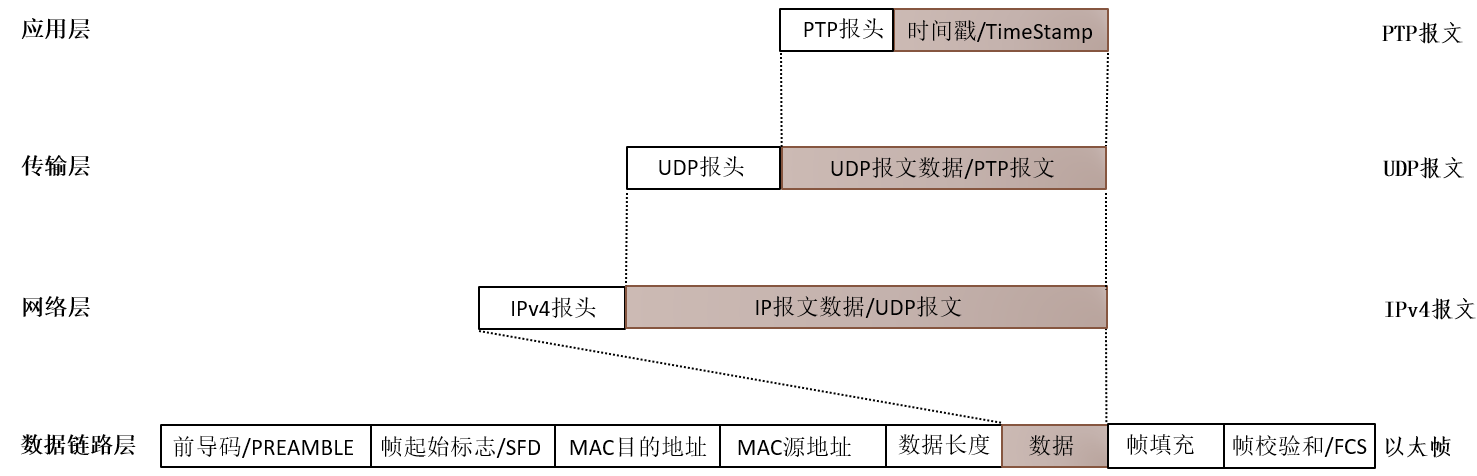


图2.7 PTP报文的UDP/IPv4封装

Figure2.7 UDP/IPv4 Encapsulation of PTP Message

PTP报文的IEEE802.3/Ethernet封装方式较为简单，应用程序在应用层将PTP报文填充完毕后，直接发送到数据链路层将其封装成以太网帧，该以太网帧只能在同一个局域网通信。该封装方式示意图如图2.8。

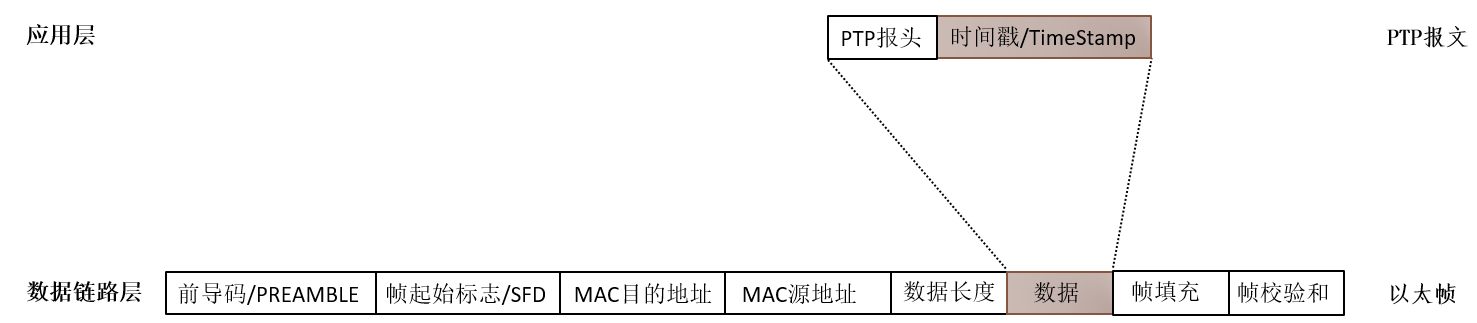


图2.8 PTP报文的IEEE802.3/Ethernet封装

Figure2.8 IEEE802.3/Ethernet Encapsulation of PTP Message

报文解析模块对接收到的以太网数据帧进行解析，检测其中是否存在PTP报文。首先检测以太网帧类型域值，若该值为0x0800，则判断该报文为UDP/IPv4封装方式的报文，随后再检测IPv4报文类型域，若该值为0x11则说明上层数据为UDP数据报文；紧接着检测UDP报文中的目的端口号，若为319/320，则判断此报文为PTP报文。若以太网帧类型域值为0x88F7，则判断接收到的报文是以IEEE802.3/Ethernet的方式来封装的PTP报文。

## 2.3 IEEE 1588同步原理

### 2.3.1 基本原理描述

IEEE 1588协议在实现时钟同步之前需要确定最佳主时钟以及时钟的主从关系。

最佳主时钟可以通过静态配置或BMC算法动态选定，该算法将根据时钟的级别、精度等来确定所在网络的最佳主时钟，同时，若网络发生变化或当前最佳主时钟的属性发生改变，可由此算法重新选择另一个时钟作为最佳主时钟[42]。BMC算法包括两个部分，分别是数据集比较算法（Data-Set Comparison Algorithm, DSCA）和状态决定算法（State-Decision Algorithm）。数据集比较算法通过比较两时钟端口对应的时钟属性，决出时钟质量较好的时钟；状态决定算法根据数据集比较算法决出的时钟属性，确定整个网络中各时钟端口的状态，生成时钟间的主从关系。

时钟的主从关系可通过BMC算法来确认，也可人工配置。在确认主从关系后，整个PTP网络形成了以最佳主时钟为根节点的树状网络结构，如图2.9所示。



图2.9 PTP主从关系示意图

Figure2.9 Diagram of PTP Master-Slave Relationship

在确定了最佳主时钟和时钟间的主从关系后，主从时钟间开始时间同步。主从时钟周期性的交换PTP报文，从时钟通过报文交换获得的时间戳来计算路径延时以及与主时钟的时钟偏差，以此偏差更新本地时钟，达到与主时钟同步的目的[43]。同步示意图如图2.10。



图2.10 IEEE 1588同步示意图

Figure2.10 Diagram of IEEE 1588 Synchronization

由图2.10所示，IEEE 1588协议的同步过程分两个阶段：

1. 路径延时的测量：此阶段测量主从时钟之间报文的传输延时。从时钟通过与主时钟的报文交换保存报文收发的时间戳，由此计算主从时钟报文往返的路径延时。假设在当前的同步过程中，主从时间偏差不变，且报文往返的路径延时相等，由此可以计算出主从时钟单向的路径延时。
2. 时钟偏差的测量：通过路径延时测量阶段获得主时钟到从时钟的单向路径延时，使用主时钟发送SynC报文时获得的主时钟发送时间戳和从时钟接受时间戳，可以计算得到主从时钟偏差，以此调整从时钟时间。

显然，在IEEE 1588v2时钟同步协议中，路径延时的获取是计算主从时钟偏差的前提，IEEE 1588v2协议定义了两种路径延时测量机制：延时请求响应测量机制和对等延时测量机制[44]。

### 2.3.2 延时请求响应测量机制

延时请求响应测量机制的原理如图2.11所示，具体的实现过程如下：

1. 主时钟向从时钟发送Sync报文，同时记录Sync报文的发送时间*t1*；从时钟接收Sync报文，并记录报文到达从时钟的时间*t2*。
2. 主时钟在发送Sync报文后，继续发送携带有*t1*时间戳信息的Follow\_up报文给从时钟。
3. 从时钟向主时钟发送Delay\_Req报文，并记录报文的发送时刻*t3*；主时钟接收Delay\_Req报文，记录报文的接收时刻*t4*。
4. 主时钟在接收Delay\_Req报文并记录接收时间后，将携带有接收时间*t4*的Delay\_Resp回复给从时钟。



图2.11 延时请求响应测量机制

Figure2.11 Delay Request-response Measurement Mechanism

由以上步骤，从时钟获得了*t1~t4*四个时间戳，由此可以计算主从路径延时。设报文由主到从的路径延时为*delay\_ms*，从到主的路径延时为*delay\_sm*，并假设在当前同步过程中，主从时钟的偏差不会发生变化，保持为*offset*，则：

 (2.1)

 (2.2)

假设主从时钟之间报文往返路径延时相同，即：

 (2.3)

记主到从单向路径延时为*delay*，根据式（2.1）和（2.2）可以计算：

 (2.4)

 (2.5)

至此，从时钟可以根据计算出的时钟偏差*offset*调整本地时钟，完成与主时钟的同步。

延时请求响应测量机制也称端到端延时测量机制，该机制不会测量网络中每段路径的延时，只测量主端到从端整个链路的报文传输延时。端到端延时测量机制最大的优势是，能在存在普通交换机或路由器的网络中正常运行，具有普适性和通用性。IEEE 1588v2协议中引入了透明时钟，当网络中使用端到端透明时钟，需要配套使用延时请求响应测量机制。

### 2.3.3 对等延时测量机制

对等延时测量机制用于测量任意两个时钟间路径延时，两时钟不分主从，互相发送Pdelay报文，测量各自的单向路径延时，图2.12展示了时钟B作为对等延时测量发起者所进行的延时测量过程。



图2.12 对等延时测量机制

Figure2.12 Peer Delay Measurement Mechanism

具体步骤如下：

1. 时钟A向时钟B发送Sync报文，并记录报文的发送时间*T1。*时钟B接收Sync报文，并记录报文的到达时间*T2*。
2. 时钟A在发送完Sync报文后，紧接着发送携带有*T1*时间戳信息的Follow\_up报文给时钟B，此时时钟B已有*T1*，*T2*时间戳。
3. 时钟B向时钟A发送Pdelay\_Req报文，并记录报文的发送时间*t1*；时钟A接收Pdelay\_Req报文，并记录报文的接收时间*t2*。
4. 时钟A向时钟B发送Pdelay\_Resp报文，并记录发送时间*t3*；时钟B接收该报文，记录接收时间*t4*。
5. 时钟A在发送Pdelay\_Resp报文后，紧接着向从时钟发送携带*t3*时间戳信息的Pdelay\_Resp\_Follo-w\_up报文。

至此，时钟B获得了*T1*，*T2*以及*t1~t4*六个时间戳，对等延时测量机制与延时请求响应测量机制路径延时的计算方法类似，在假设时钟之间的往返路径延时相等的情况下，可以计算出时钟A到时钟B的路径延时*delay\_ab*为：

 (2.6)

时钟B在计算出*delay\_ab*后进入时钟偏差测量阶段，根据时钟B接收到Sync报文的时间*T2*和时钟A发送Sync报文的时间*T1*，可由公式（2.5）计算得出时钟偏差，实现时钟校准。

对等延时测量机制又称点到点延时测量机制，是IEEE 1588v2协议新增的一种延时测量机制。该延时机制可以计算网络中每段链路的报文传输延时，但前提是网络中的设备都需要支持IEEE 1588v2协议。例如，支持IEEE 1588v2协议的交换机在转发报文时会处理自身的排队延时，而普通的交换机不会处理对等延时消息，且不会响应报文。当网络中存在点到点透明时钟时，需要配套使用对等延时测量机制。

对等延时测量机制相比延时测量机制具有多个优点：（1）所有链路会定期测量延时，因此，在网络结构更改时，不用重新测量路径延时，可以直接使用已测得的与自身相连时钟的路径延时。（2）路径延时测量是一个独立的过程，可多次测量以提高延时测量精度。（3）当有多个从时钟时，无需担心主时钟响应Delay\_Request报文的能力，只需发送Sync和Follow\_up报文即可。

## 2.4 时钟同步精度影响因素分析

### 2.4.1 非对称路径延时对同步精度的影响

在IEEE 1588协议的同步原理中，两种路径延时的测量方法都是以假设报文往返传输延时对称的为前提而计算的，但在实际以太网中，这个假设是不成立的。网络协议栈延时抖动，报文经过交换机和路由器等网络中间件所产生的转发延时等因素都会导致报文往返传输的延时不对称[45]。正如公式（2.4），主从延时是以总路径延时平均值来计算的，若在同步过程中，这些因素造成延时异步，导致报文的实际往返延时相差过大，将会极大降低计算单向路径延时的准确性，从而降低同步的精度。

在网络通信中，网络协议栈会将通信模型中的数据流划分多个层级，常用的TCP/IP协议栈将网络模型划分为四个层级，从底层到顶层分别为：链接层、网络层、传输层和应用层。报文在应用层由相关应用程序填充完毕后，会逐步向下一层传输，每经过一层，就会在上一层报文的基础上添加报头和校验信息，这些操作会受到操作系统任务调度和处理器中断的影响，因此同步报文的发送延时具有不确定性，类似的，接收延时也无法准确估计。

在实际应用中，主从时钟一般会通过交换机或路由器进行连接，这些网络中间件接收到PTP报文后会对报文进行解析、缓存、寻路、再封装、排队转发等一系列操作，这些过程产生的处理延时会导致报文传输延时的不确定。当网络中数据流量过大，负载严重时，这些处理过程将会严重影响时钟同步的精度。

### 2.4.2 时间戳获取位置对同步精度的影响

IEEE 1588协议是根据交换携带时间戳信息的报文，由时间戳来完成主从路径延时和时钟偏差的计算，因此时间戳的精度会直接影响同步的精度[46]。

如图2.13所示，可以在网络协议栈的应用层、驱动层、和硬件层获取时间戳信息。

在应用层记录PTP报文的发送和接收时间是较为简单的方式，同时也是获取时间戳精度最低的方式，延时的波动有几百微秒。PTP数据包在协议栈顶部生成并发送，这些数据包从应用程序到物理接口遍历了整个软件堆栈，接收数据包也需要遍历类似的软件堆栈以返回到从时钟的PTP应用程序。PTP数据包在进行封包、解包、发送、接收一系列处理期间，逐层传递导致的协议栈处理延时以及流量负载会在主从两端产生较大的随机延时。同时，主从时钟设备在处理器能力、处理的数据包数量以及每个处理器上的不同软件加载等不同环境，PTP数据包被处理器移动队列的调度顺序和时间以及队列深度都有较大的不确定性，很大程度影响了同步的精度。

在网络驱动层获取时间戳信息将会是比较好的选择，由内核或中断服务例程提供。在此处获取的时间戳很大程度地避免了网络协议栈延时带来的不固定抖动，时间戳精度取决于中断延时和CPU的处理能力。

MAC层获取时间戳的方式需要有IEEE 1588协议的硬件时钟支持，该方式是在与PHY相连的MAC的物理接口上获取时间戳。MAC控制器在检测到PTP事件报文时触发中断，将PTP时间戳填入报文相应字段。相比于网络驱动层获取的时间戳，该方式的时间戳精度更高。

时间戳精度最高的获取位置是硬件层，因为在PHY芯片已经尽可能接近通信线路，在数据包进入和离开硬件时对其进行时间标记，可以获得几乎没有抖动的高精度时间戳，同样需要专门的硬件支持。



图2.13 时间戳的获取位置

Figure2.13 Position of Getting The Timestamp

# 时钟同步模块的系统设计

## 3.1 时钟同步模块整体框架

上文分析了时钟同步性能的影响因素，指出在应用层获取时间戳将很大程度地影响时钟同步精度。目前，通过纯软件实现IEEE 1588v2协议的同步精度已经无法满足当前各领域和应用场景的要求，借助硬件平台实现高精度的时钟同步尤为重要[47]。

使用硬件平台实现IEEE 1588v2协议有三种方案可供选择[8]：

（1）采用SOPC系统实现时钟同步功能，使用FPGA进行PTP数据包的收发和记录硬件时间戳，CPU进行协议分析和数学计算。FPGA记录的硬件时间戳精度较高，其硬件描述语言移植性较强，但开发周期长，信号时序匹配，仿真和软硬件结合的难度较大。

（2）采用微控制器来实现时钟同步功能，使用STM32等支持IEEE 1588v2协议的微控制器和PHY芯片。此方案可以在以太网控制器和PHY芯片之间的MII/RMII接口处获取时间戳，消除了网络协议栈的延时抖动。此方案需要对微控制器外围电路进行设计，电路设计的优劣会影响同步的精度。

（3）采用设计一体化、支持操作系统的高性能处理器来实现时钟同步，处理器包含支持IEEE 1588v2协议和硬件时间戳获取的以太网控制器。该方案不需要复杂的设计和调试，由于操作系统的存在，众多开源软件得以支持，并且系统中包含大量的库函数，使得开发效率大大提升。

由以上分析，考虑开发周期、难易程度，开发成本以及时钟模块的可扩展性和可维护性，选取方案（3）作为本文的设计方案。

时钟同步模块框架设计如图3.1所示。



图3.1 时钟同步模块框架图

Figure3.1 Frame of Clock Synchronization Module

## 3.2 主控芯片的选择

### 3.2.1 AM3358简介

AM3358是具有NEON协处理器的ARM Cortex-A8 32位精简指令集计算机处理器，支持高达1GHz的频率，包含mDDR、DDR2、DDR3控制器，在图像、图形处理、外设以及EtherCAT和PROFIBUS等工业接口选项方面得到了增强，并支持高级操作系统[25]。

AM3358微处理器包含众多子系统，其功能框图如图3.2。其中包括基于ARM Cortex-A8处理器的微处理器单元(MPU)子系统、可编程实时单元子系统和工业通信子系统(PRU-ICSS)与ARM内核彼此独立，允许单独操作和计时，以实现更高的效率和灵活性。PRU-ICSS支持附加外设接口以及EtherCAT、PROFINET、EtherNet/IP、PROFIBUS、以太网 POWERLINK、串行实时通信协议(Sercos)等实时协议。此外，凭借PRU-ICSS的可编程特性及其对引脚、事件和所有片上系统(SoC)资源的访问权限，该子系统可以灵活地实现快速实时响应、专用数据处理操作以及自定义外设接口，并减轻SoC其他处理器内核的任务负载。同时，时钟管理(PRCM)模块集成了15MHz至35MHz的高频振荡器，用于为各种系统和外设时钟生成参考时钟，可提供实时日期(年、月、日和星期几)和时间(小时、分钟和秒)信息。

AM3358微处理器可外接最多两个带集成PHY的USB2.0高速双角色器件（DRD）端口；两个工业千兆位以太网MAC（10、100和1000Mbps），每个MAC都支持MII、RMII、RGMII和MDIO接口，同时每个MAC和交换机可独立于其他功能运行并支持IEEE 1588v2精密时间协议。图3.2为AM3358芯片的功能框图。

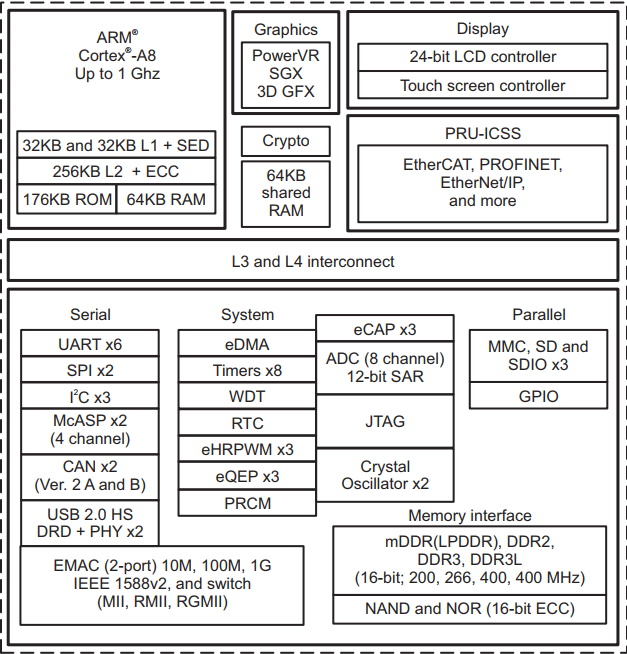


图3.2 AM3358功能框图

Figure3.2 AM3358 Functional Block Diagram

### 3.2.2 AM3358硬件时钟的支持

在Sitara系列的SOC（System on Chip）上存在一个称为CPSW的以太网MAC/PHY接口，该接口是一个三端口交换机，可以在双端口配置或交换机配置中使用，并且支持最多两个物理层或两个网络，CPSW或以太网交换机驱动程序遵循标准的Linux网络接口体系结构[48]。对于CPSW，硬件时钟的支持将由CPTS模块提供，其框图如图3.3所示。

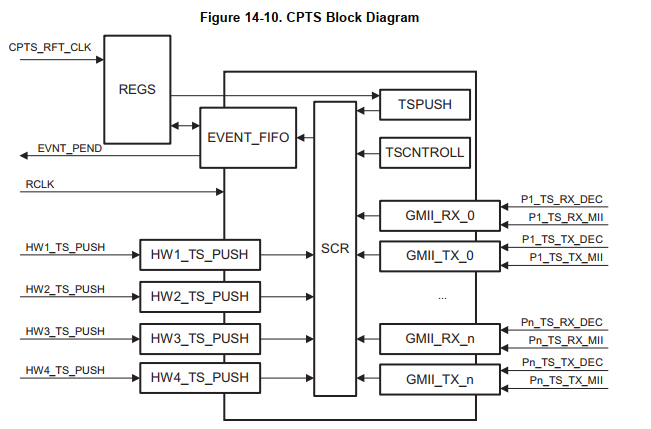


图3.3 CPTS模块框图

Figure 3.3 CPTS Block Diagram

在CPTS模块上有多个不同的时钟可供使用，这些时钟对于不同的SOC会有不同的选择。因此，需要确保并查看正在使用的SOC，确认SOC支持的时钟类型，并通过寄存器来配置所需时钟和使能各种其他的功能。图3.3中展示的CPTS\_RFT\_CLK寄存器位决定了使用的时钟为IEEE 1588v2时钟。

AM3358使用CPTS对事件进行时间戳标记，CPTS中包含一个EVENT\_FIFO，其中包含事件本身以及该事件发生的时间戳，同时会获得一个事件挂起通知，可用于确定读取这些事件和查看其时间戳并开始采取适当的操作的时间。

当数据包通过MII接口发送或接收数据时，CPTS模块的以太网端口会对每个数据包进行解码来判断数据包的类型，若为有效的时间同步数据包，则会产生以太网发送事件或以太网接收事件，将其加载到EVENT\_FIFO中进行处理，并包含事件记录的时间戳信息，该事件的时间戳信息由CPTS时钟产生。HW\_TS\_PUSH事件由硬件时戳输入产生，并将其加载到EVENT\_FIFO中进行处理，每个硬件时间戳输入都会与计时器相连，每隔一秒就会产生一个事件。此外，软件可以通过位和寄存器启动TS\_PUSH事件来获取当前时间戳值，事件中将返回时间戳值。在从EVENT\_FIFO中读取第一个时间戳之前，软件不能将第二个时间戳事件推入FIFO。

以上是CPTS获取事件时间戳不同方法。PTP是通过网络进行报文交互来同步的，故被CPTS模块判断为时间同步数据包，并根据数据包中报文类型域的值来判断当前PTP报文类型。

## 3.3 以太网PHY模块硬件设计

AM3358处理器包含两个千兆以太网MAC控制器，以RGMII接口形式与外部PHY芯片连接。PHY模块由AR8031\_AL1A芯片及其外围电路构成，如图3.4所示。AR8031\_AL1A是高度集成的数字信号收发器，在提供高性能的同时大幅降低了成本。AR8031\_AL1A为半/全双工10BASE-Te、100BASE-TX和1000BASE-T以太网提供物理层功能，以此通过标准的5类(CAT5)非屏蔽双绞线电缆发送和接收高速数据。表3.1展示了PHY芯片RGMII接口信号的相关信息。

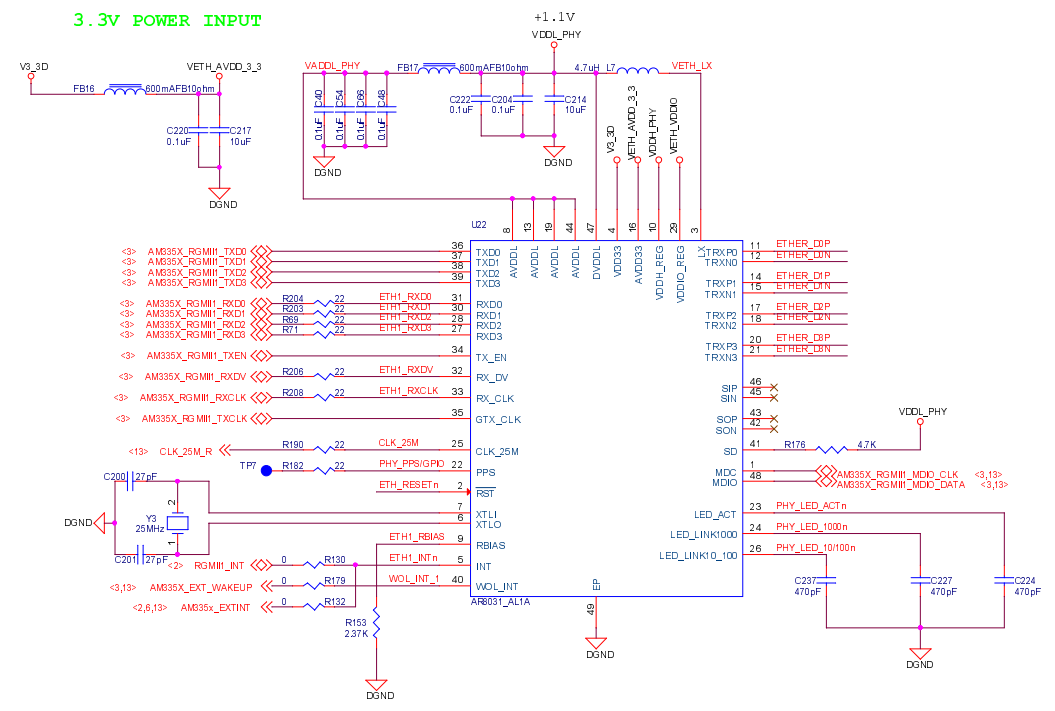


图3.4 PHY模块电路图

Figure3.4 Circuit Diagram of PHY Module

表3.1 信号类型和描述

Table3.1 Signal type and Description

|  |  |  |
| --- | --- | --- |
| 信号名 | 类型 | 描述 |
| GTX\_CLK | I | RGMII发送时钟 |
| RX\_CLK | I/O | RGMII接收时钟 |
| RX\_DV | I/O | RGMII接收数据有效 |
| RXD[3:0] | I/O | RGMII接收数据通道 |
| TX\_EN | I | RGMII发送使能 |
| TXD[3:0] | I | RGMII发送数据通道 |
| MDC | I | RGMII管理时钟 |
| MDIO | I/O | RGMII管理数据通道 |

## 3.4 PTP软件平台设计

### 3.4.1 Linux操作系统

分布式系统功能的复杂化和多样化是现今分布式系统发展的主要方向，模块化和多任务趋势使得分布式系统都必须获得操作系统的支持才能实现越来越复杂任务要求。时钟同步模块使用的AM3358处理器能很好地兼容Android和Linux系统。Linux系统具有代码完全开源、系统版本更新快、可移植性好、模块化程度高、可维护性号、具有出色的网络性能等特点，Linux系统可以根据实际需要进行裁剪，系统中不需要的模块和功能可以不进行编译，这样可以大大减少操作系统所占的空间[49]。除了以上系统本身的优点，在Linux-3.0版本以后，Linux内核还添加了对于网络数据包硬件时间戳以及PTP硬件时钟的支持，更新的内核驱动程序抽象了基于CPTS的PTP支持，使得CPTS时钟映射为字符设备，该设备提供ioctl()、poll()和read()等方法，其中大部分功能都可以通过ioctl访问。

PTP协议是通过网络进行报文交互来实现的，Linux系统拥有出色的网络协议栈，同时拥有完善的网络通信机制[50]。在Linux系统中，网络设备与块设备和字符设备的操作不同，应用程序只能通过套接字来完成与网络设备接口的访问。Linux系统的网络设备驱动定义了四个层次，分别为网络协议接口、网络设备接口层、设备驱动功能层和网络设备与媒介层，如图3.5所示。这四层的作用如下：

1. 网络协议接口层向网络层协议提供透明的、统一的数据包收发接口。当上层协议需要发送或接收数据时，通过调用dev\_queue\_xmit()函数或netif\_rx()函数来完成数据包的收发。
2. 网络设备接口层向协议接口层提供一个统一的结构体net\_device用来描述网络设备的具体属性和操作。
3. 设备驱动功能层的作用是为了给上层的net\_device结构体定义具体的成员变量或函数，通过hard\_start\_xmit()函数驱动硬件完成相关操作。
4. 网络设备与媒介层为具体的网络适配器和传输媒介，是最底层完成数据包的发送和接收的实体。

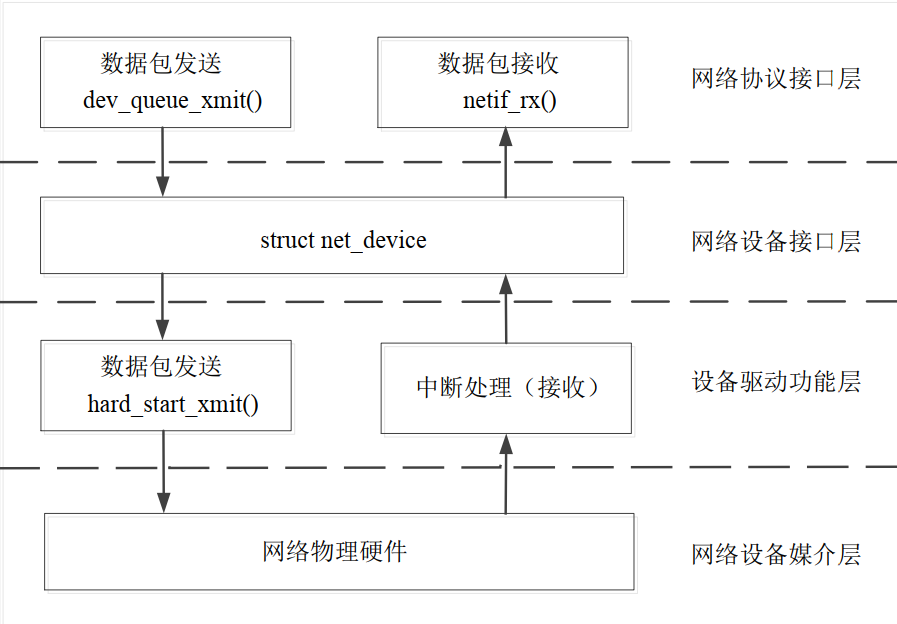


图3.5 Linux网络驱动设备结构图

Figure3.5 Structure Diagram of Linux Network Driver

生产AM3358的TI公司提供了软件开发套件（Software Development Kit），其中包括长期稳定的主线Linux内核支持、U-Boot引导加载程序支持、Linaro GNU Compiler Collection(GCC)工具链和兼容Yocto Project OE Core的文件系统等一系列嵌入式Linux设备的必要组件和工具，并包含有关软件基础结构及其详细信息以及适用于Sitara处理器的Code Composer Studio（CCS）集成开发环境（Integrated development environment，IDE），以便开发人员创建应用程序。

综上所述，选择Linux系统作为时钟同步模块开发的软件平台。

### 3.4.2 Linux PTP简介

Linux PTP是Linux系统的IEEE 1588v2协议的精确时钟协议（PTP）的实现。该软件通过Linux系统中SO\_TIMESTAMPING套接字选项来获取硬件和软件时间戳；通过使用clock\_gettime系列的函数调用（包括clock\_adjtimex系统调用）来支持Linux PTP硬件时钟（PHC）子系统；实现了边界时钟（BC），普通时钟（OC）和透明时钟（TC）；支持UDP/IPv4，UDP/IPv6和原始的以太网传输；支持IEEE 802.1AS-2011；采样模块化设计，提高开发效率，可轻松添加新的传输方式和时钟伺服器；实现单播操作；支持多种配置文件，包括：汽车以太网配置文件、默认的1588配置文件、电信配置文件G.8265.1，G.8275.1和G.8275.2；支持NetSync Monitor协议；支持绑定IPoIB和VLAN接口[51]。

Linux PTP的实现为有限状态机（FSM），其中状态的转换取决于收到的消息。除了实现基本的PTP协议要求外，最重要的是支持PTP硬件时钟，通过SO\_TIMESTAMPING套接字选项将来自PHC的带有时间戳的数据包提供给PTP堆栈，PTP堆栈计算适当的校正偏差并使用标准的POSIX时钟功能来调整PHC。Linux PTP的运行需要至少Linux-3.0版本的内核支持，通过内核驱动配置文件将PTP时钟的驱动编入内核。通过ethtool应用程序可以查看MAC是否支持硬件或软件时间戳，以及是否支持PTP硬件时钟。

在Linux系统下输入命令：ethtool –T eth0，此命令显示MAC是否支持硬件或软件时间戳。图3.6展示了MAC对硬件时间戳的支持，并被注册为PTP硬件时钟：0。

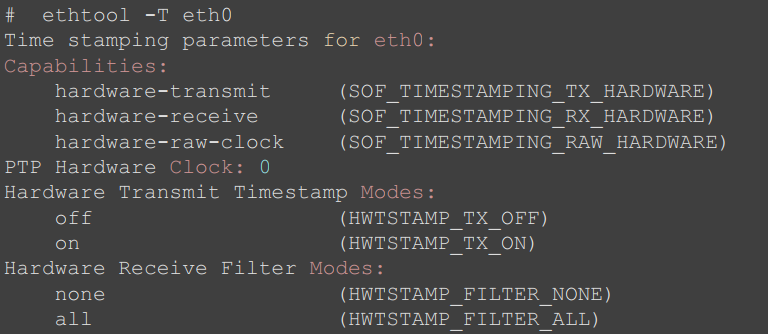


图3.6 PTP硬件时钟示意图

Figure3.6 Diagram of PTP hardware clock

Linux PTP主要提供了三个应用程序ptp4l、phc2sys和pmc。ptp4l是根据IEEE 1588v2标准的精确时钟协议的实现，可作为边界时钟（BC）和普通时钟（OC）。使用BMC为每个端口自动确定“从”和“主”的角色；phc2sys用于将正常的Linux系统时间与PTP硬件时钟同步，PTP硬件时钟本身由ptp4l应用程序与外部主时钟同步。对于phc2sys，“主”和“从”不是PTP协议定义的概念，而是相对于两个本地时钟的指代，通常PHC是“主”，Linux系统时间是“从”。pmc用于发送PTP管理查询的应用程序。该程序从名称和管理ID指定的标准输入操作中读取信息，通过选定的传输方式发送，并打印所有收到的消息。

### 3.4.3 PTP硬件时钟

为了在用户空间能够实现对硬件时钟的控制，Linux系统需要为此提供相应的接口。PTP硬件时钟子系统是Linux内核在3.0版本添加的，旨在为开发应用程序、时钟驱动程序、同步Linux系统时钟与外部时钟以及使用硬件时钟的辅助功能提供标准化接口的子系统。结合已支持的网络数据包时间戳的获取，对在Linux操作系统下开发PTP软件完成内核级的支持[52]。系统架构如图3.7所示。

Linux内核中存在一种能够实现有相似结构的驱动程序的共享代码，称为“类驱动程序”，该驱动程序拥有类似“面向对象”的程序模型，能够为特定时钟驱动提供内核接口，并为应用程序提供用户空间接口，处理与用户空间的所有事务；同时支持多个时钟驱动，满足在测试和开发多时钟系统过程中，对在单个系统中运行的多时钟进行性能比较。

“类驱动程序”的存在提高了开发效率，特定时钟驱动的编写只需要专注于硬件时钟的编程细节及其实现，驱动程序需要向PTP类时钟驱动注册其自身，特定时钟注册成功后，时钟驱动程序会通知类驱动程序为其创建一个字符设备，用户空间的应用程序可以使用标准的I/O控制方法来控制特定时钟。



图3.7 PTP硬件时钟子系统

Figure3.7 PTP Hardware Clock Subsystem

PTP硬件时钟子系统提供了用户空间应用程序接口（Application programming interface，API）和针对时钟驱动开发的内核API。

PTP时钟的驱动程序必须包括linux/ptp\_clock\_kernel.h头文件才能使用内核API，内核API提供三个函数。其中两个函数功能分别是允许时钟驱动向PHC子系统注册和从子系统注销，并且从中断服务程序调用第三个函数将注册或注销事件发送到PHC子系统。为了注册PTP硬件时钟，时钟驱动必须定义一个结构体PTP\_CLOCK\_INFO，结构体定义如图3.8。该结构体定义了时钟的功能和类接口，每个时钟必须实现类接口中的四个回调函数：adjfreq()、adjtime()、gettime64()和settime64()，因为它们是基本的时钟控制操作。setimer64()和gettimer64()函数提供报警管理，enable()函数用于控制外部事件的时间戳和周期性输出信号。

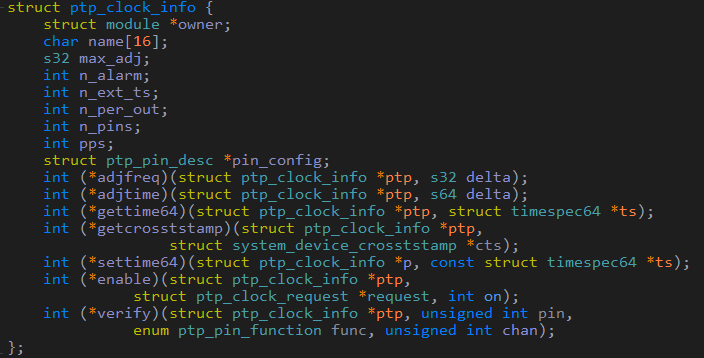


图3.8 结构体ptp\_clock\_info的定义

Figure3.8 Definition of structure ptp\_clock\_info

为了在用户空间对PTP硬件时钟进行编程，应用程序需要包含头文件linux/ptp\_clock.h，并打开映射硬件时钟的字符设备。该设备提供ioctl()、poll()和read()函数，可以访问大部分功能。字符设备提供的ioctls类似POSIX时钟API，表3.2中归纳了ioctls和类似的POSIX API。ptp\_clock\_adjfreq的参数是所需的频率补偿，单位为百万分之一(ppb)；用户程序可以使用ptp\_clock\_getcaps查询时钟的相关属性，在使用此ioctl后会返回一个结构体，该结构体包含时钟的最大频率调整、可编程报警的数量、外部时间戳通道的数量、可编程周期信号的数量以及时钟是否支持PPS回调。ptp\_clock\_feature\_request是剩余的辅助时钟特性总称，通过传递指定资源的ptp\_clock\_request结构体来启用或禁用相应功能，若启用了外部时间戳功能，应用程序可以通过阻塞read()和poll()来接收带时间戳的事件。

表3.2 PHC和POSIX系统调用

Table3.2 System Calls of PHC and POSIX

|  |  |
| --- | --- |
| PHC的ioctls | POSIX API |
| ptp\_clock\_adjfreq | adjtimex |
| ptp\_clock\_adjtime | none |
| ptp\_clock\_gettime | clock\_gettime |
| ptp\_clock\_settime | clock\_settime |
| ptp\_clock\_getcaps | none |
| ptp\_clock\_gettimer | timer\_gettime |
| ptp\_clock\_settimer | timer\_settime |
| ptp\_clock\_feature\_request | none |

当编写时钟驱动时，时钟驱动程序需要包含头文件linux/ptp\_clock\_kernel.h，并提供一个结构体ptp\_clock\_info来向类驱动注册本身。时钟驱动必须实现接口中的所有功能，否则驱动程序会从相应的函数返回错误标识。由于大多数硬件实现都将时间值作为两个32位寄存器访问的64位整数，因此驱动程序使用spin\_lock\_irqsave()和spin\_unlock\_irqrestore来防止并发访问。

Linux系统内核支持AM3358芯片，内核驱动包含了CPTS的驱动文件，通过驱动配置文件使能CPTS模块，将CPTS作为PTP硬件时钟。

### 3.4.4 硬件时间戳的获取

在Linux3.0版本后，Linux系统的网络堆栈已经支持硬件时间戳，为用户空间提供了API接口。主从时钟通过网络交换报文进行数据通信时，需要打开套接字socket，使用SIOCSHWTSTAMP控制命令为硬件时间戳配置设备驱动程序。由于并非所有硬件都能为每个接收到的报文添加时间戳，因此应用程序可以指定期望收到的报文类型。在socket中添加SO\_TIMESTAMPING选项支持软件/硬件时间戳的获取。如表3.3所示，SO\_TIMESTAMPING选项有多个设置选项，可通过setsockopt()函数来设置具体的选项获得相应的效果，此选项支持启用单独的发送或接收时间戳，同时允许在硬件时间戳不可用时使用软件时间戳作为后备。

表3.3 SO\_TIMESTAMPING设置选项

Table3.2 Option of SO\_TIMESTAMPING

|  |  |
| --- | --- |
| 设置选项 | 返回结果 |
| SOF\_TIMESTAMPING\_TX\_HARDWARE | 返回发送报文的硬件时间戳 |
| SOF\_TIMESTAMPING\_RX\_HARDWARE | 返回接收报文的硬件时间戳 |
| SOF\_TIMESTAMPING\_TX\_SOFTWARE | 若SOF\_TIMESTAMPING\_TX\_HARDWARE未使能或返回错误，则返回发送报文的软件时间戳 |
| SOF\_TIMESTAMPING\_RX\_SOFTWARE | 若SOF\_TIMESTAMPING\_RX\_HARDWARE未使能或返回错误，则返回接收报文的软件时间戳 |
| SOF\_TIMESTAMPING\_RAW\_HARDWARE | 返回原始硬件时间戳 |
| SOF\_TIMESTAMPING\_SYS\_HARDWARE | 返回已转为系统时间的硬件时间戳 |
| SOF\_TIMESTAMPING\_SOFTWARE | 返回系统的软件时间戳 |

报文在被发出时，时间戳会包含在SO\_TIMESTAMPING字段中。接收端在接收到该报文时，可以通过函数recvmsg()从该报文的SO\_TIMESTAMPING字段中获取到时间戳，在recvmsg()中调用CMSG\_FIRSTHDR()和CMSG\_NXTHDR()宏遍历报文的附加数据确定时间戳的地址。当cmsg\_level的值等于SOL\_SOCK的值，同时cmsg\_type的值与SO\_TIMESTAMPING的值相等，表示时间戳被找到，并通过CMSG\_DATA()宏来获取。

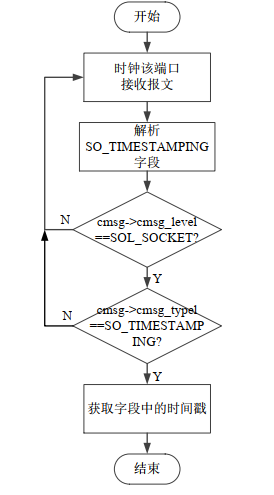


图3.9 接收报文时间戳的获取流程

Figure3.9 The Process of Obtaining The Timestamp of The Received Massage

时间戳获取的流程图如图3.9所示。recvmsg()函数根据设置选项返回对应的时间戳以及数据包数据。对于发送报文时间戳的获取，与接收报文时间戳的获取方式类似。当发送报文时，报文会被循环回socket的错误队列，在错误队列中报文会被添加时间戳信息，该时间戳同样通过recvmsg()函数获得，因此需要修改函数的标记位以满足对错误队列的处理。

# 时钟同步模块的算法设计

## 4.1 卡尔曼滤波算法

### 4.1.1 时钟状态的建模

卡尔曼滤波至今已经有60多年的历史，但它仍然是当今使用最频繁的滤波算法之一。卡尔曼滤波以其发明者鲁道夫·E·卡尔曼的名字命名，其主要特征是其计算量小，高效率的递归性质，以及它对含有高斯误差的一维线性系统具有优秀的估计效果。卡尔曼滤波的典型应用包括平滑噪声数据和提供参数的估计，其实际应用广泛，包括全球定位系统、锁相环电路、电力系统、目标跟踪、金融等众多领域[53]。在IEEE 1588v2时钟同步过程中，由于同步报文交换的不确定性因素，导致主从时钟偏差的测量值产生抖动，影响同步的精度，因此加入卡尔曼滤波算法，消除抖动误差，提高时钟同步精度[54]。

在使用卡尔曼滤波算法前，首先要对时钟的状态模型进行建模[]，建立时钟偏差、频率偏差和路径延时的状态方程和观测方程。

主时钟时钟源一般为高精度、高稳定性时钟，可将其作为理想的参考时钟，时钟模型为：

 (4.1)

从时钟由本地晶振时钟和计数器生成本地时间，其是时钟模型可表示为：

 (4.2)

其中，为从时钟的瞬时时钟斜率，为初始时钟偏差，为*t*时刻的干扰噪声。由此可得主从时钟偏差为：

 (4.3)

其中，，为主从时钟的瞬时频率偏差。

假设主从时钟的瞬时频率偏差为常量，则主从时钟偏差的离散模型为：

 (4.4)

其中，*Γ*为同步时间间隔*δ*内的时钟瞬时频率偏差，为每段同步周期内的噪声。由此可以得到时钟偏差的状态方程：

 (4.5)

记主从时钟频率偏差和路径延时的变化为随机过程，两者的状态方程为：

 (4.6)

 (4.7)

在式(4.5)、(4.6)和(4.7)中，、和分别是服从、和的高斯分布的噪声。

根据IEEE 1588v2协议同步原理，路径延时和主从时钟偏差可以表示为：

 (4.8)

 (4.9)

式中，表示第*n*次同步周期主从时钟报文交换获得的时间戳。根据连续两次同步周期所计算的时钟偏差可得频率偏差：

 (4.10)

在(4.8)、(4.9)和(4.10)中，、和分别是服从、和的高斯分布的噪声。其中，。

### 4.1.2 滤波过程

卡尔曼滤波是一种高效的递归滤波器，得到滤波对象的状态空间模型后即可对其进行滤波计算。

由式(4.5)、(4.6)和(4.7)可以建立时钟的状态转移方程，其中=为同步时间间隔。

 (4.11)

根据式(4.8)、(4.9)和(4.10)建立时钟的观测方程：

 (4.12)

由式(4.11)和(4.12)可得时钟模型的状态空间方程：

 (4.13)

其中，与为第*n*个同步周期的时钟状态向量和时钟观测向量，和分别为状态转移噪声和观测噪声。*A*为状态转移矩阵，H为观测矩阵：

，

设*Q*为状态转移噪声的协方差矩阵，*R*为观测噪声的协方差矩阵，且，。则有

，

卡尔曼滤波估计当前时钟参数主要分为预测和更新两个过程：

1. 预测时钟的当前状态

在*n*时刻根据式（4-），由*n-1*时刻的时钟状态最优估计预测*n*时刻的时钟状态。

 (4.13)

同时根据*n-*1时刻时钟状态估计的协方差及状态转移噪声，预测当前估计的协方差矩阵，由式（4-）计算得出：

 (4.14)

1. 更新时钟当前状态

在预测得出当前时钟状态的估计值及其协方差后，结合当前时钟状态的观测值以及观测噪声，由式（4-）和（4-）更新时钟的当前状态和协方差矩阵。

 (4.15)

 (4.16)

其中，为卡尔曼增益，通过式（4-）计算得出：

 (4.17)

综合以上过程，卡尔曼滤波的计算流程由图4.所示：



图4.1 卡尔曼滤波流程图

Figure4.1 Flowchart of Kalman filter

## 4.2 时钟频率补偿算法

时钟同步分为频率同步和相位同步。频率同步是指两个时钟的频率变化保持固定或步长一致，并保持固定的相位差；相位同步以频率同步为前提，指两个时钟的当前时间相同，即相位差恒为零。

在计算完主从时钟偏差后，从时钟需要调整本地时钟使之与主时钟同步。调整本地时钟的方法有两种：调整时钟的时间和补偿时钟频率。调整时钟的时间是通过当前计算的时钟偏差直接与时钟的时间来相加减；补偿时钟频率是根据当前时钟偏差计算频率偏差的补偿值，以此来调整从时钟的时钟频率。

目前，通常使用比例积分(PI)控制器来计算频率补偿值以此校正本地时钟频率[55]。其计算过程如下：

 (4.18)

 (4.19)

其中，*P*为比例系数，*I*为积分系数，为第*i*个同步周期的主从时钟偏差，*drift*为个同步周期的时钟偏差累加值，*adj*为第*n*个周期计算的频率偏差补偿值。由式(4.18)和(4.19)可以看出，*P*和*I*的取值对最后的频率偏差补偿值的计算起决定性作用，由于控制目标的设备模型、控制要求的不同，两者的取值需要考虑实际系统的运行情况，一般是通过经验估计并进行多次测试以确定合适的参数取值。

由于时钟同步模块运行Linux系统，故在计算出当前同步周期的频率偏差补偿值后，可以调用Linux的系统调用函数clock\_adjtime()根据补偿值调整时钟频率。在实际应用时，调整时钟的绝对时间和补偿时钟的频率两种调整本地时钟的方法是互相配合使用的，当主从时钟的时钟偏差过大，可以根据时钟偏差直接调整从时钟的绝对时间，使同步过程更快的趋于稳定。当时钟偏差较小，使用补偿时钟频率的方式来调整时钟，使得调整过程更加平滑、稳定。

## 4.3 时钟同步精度测试

在进行实验测试之前，需要统一设置PTP协议的运行参数，见表4.1，本文所有测试实验的PTP配置参数均以此表来设置。

表4.1 PTP协议配置参数

Table4.1 Configuration Parameters of PTP

|  |  |
| --- | --- |
| 名称 | 参数 |
| 通信方式 | 组播 |
| 承载协议 | IEEE802.3 |
| 延时测量机制 | E2E |
| 同步报文发送频率 | 1Hz |

### 4.3.1 标准符合测试

* + - 1. 标准符合测试部分的主时钟选用深圳市夏光时间技术有限公司的产品SyncEdge时频同步分析仪，该设备内设GPS/BD接收机以及设铷原子钟，铷原子钟频标短期的稳定度可达10-12量级，可作为时钟同步网的一级节点使用，同时支持标准的PTP协议，可通过千兆以太网网口与其它设备进行报文交互。

实验测试的从时钟使用的是搭载AM3358处理器和AR8031\_AL1A物理层芯片时钟同步模块，其底视图如下：

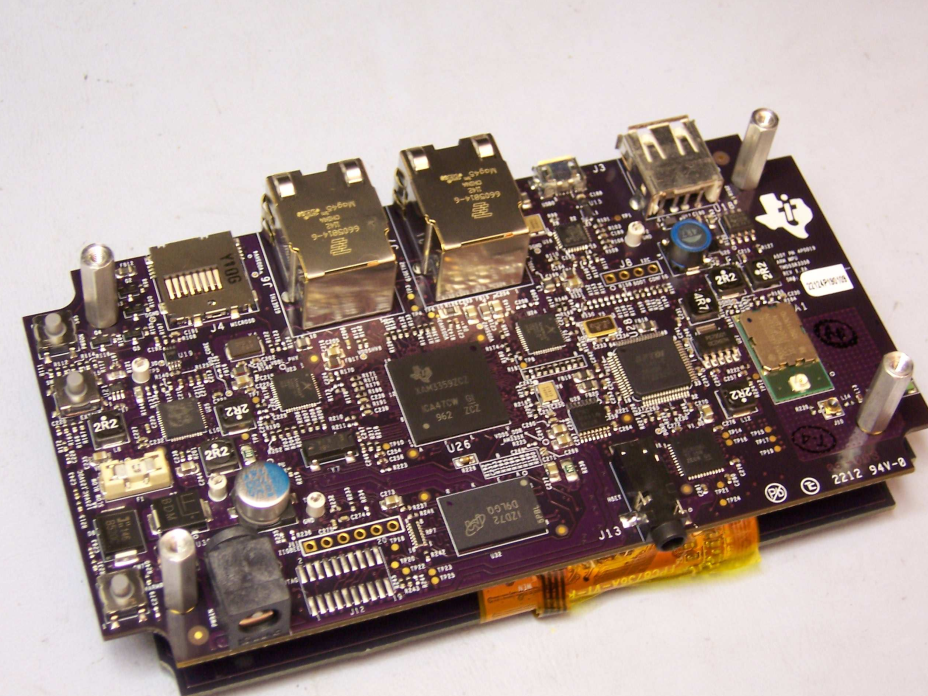


图4.2 时钟同步模块底视图

Figure4.2 Bottom View of Clock Synchronization Module

本小节的实验测试将时频同步分析仪与时钟同步模块通过网线直接相连，同时通过Micro usb数据线将时钟同步模块连接至计算机终端。设置时频同步分析仪工作于PTP主时钟模式，在计算机终端上将时钟同步模块设置为PTP从时钟模式，两者根据表4.1内容设置PTP配置参数。在时钟同步模块锁定后，从时钟模块会实时计算主从时钟偏差，通过Micro usb将时钟偏差保存至计算机终端，并使用MATLAB软件工具绘图展示。

在时钟同步模块上运行Linux PTP应用程序，使用PTP硬件时钟记录时间戳，设置*P*和*I*的值分别为0.4743和0.012，以标准的PTP协议进行同步测试，测试结果如图4.3。

D:\Desktop\夏光表No_kalman.emf

图4.3 标准符合测试的时钟同步精度

Figure4.3 Clock Synchronization Accuracy of The Standard PTP

由上图所示，由于时频同步分析仪时钟频率的高度稳定，主从时钟偏差的最大值为7ns，最小值为-8ns，时钟偏差均值和对应的标准差分别为1.96ns和2.54ns，实现了较高精度的时钟同步。

图4.4为采用卡尔曼滤波的时钟同步精度测试结果，卡尔曼滤波的状态转移噪声，观测噪声，为了便于描述，将引入了卡尔曼滤波的PTP时钟同步称为K-PTP（kalman-PTP）。最终测试的时钟偏差的最大值为8ns，最小值为-6ns，时钟偏差均值1.81ns，对应的标准差为2.32ns。

D:\Desktop\夏光表加kalman.emf

图4.4 K-PTP时钟同步精度测试

Figure4.4 Clock Synchronization Accuracy of K-PTP

由于主时钟的时钟频率的高度稳定性，从时钟计算的当前时钟偏差与卡尔曼滤波对其预测值相差较小，考虑时钟同步过程的偶然性，判断K-PTP对本小节实验测试的时钟同步精度影响较小。

### 4.3.2 时钟同步模块同步精度测试

本小节的测试采用两个时钟同步模块，其中一个作为主时钟，另一个作为从时钟，分别测试两者直连和通过以太网交换机相连情况下的时钟同步精度。主从时钟通过Micro usb分别与计算机终端相连，两者通过运行ptp4l应用程序来进行PTP时钟同步，同时设置*P*和*I*的值分别为0.5和0.0625。

主从时钟通过直连的方式相连接，分别测试引入卡尔曼滤波的PTP同步（K-PTP）和标准PTP同步的时钟同步精度，K-PTP的卡尔曼滤波参数，观测噪声，测试结果的对比图如图4.5所示。

D:\Desktop\板板直连.emf

图4.5 直连方式K-PTP和标准PTP时钟同步精度

Figure4.5 Clock Synchronization Accuracy of K-PTP and Standard PTP under Direct Connection

通过标准PTP测试的时钟偏差均值为22.53ns，对应的标准差为27.47ns；K-PTP的时钟偏差均值为18.02ns，标准差为23.87ns。

由于时钟同步模块中时钟频率的稳定度远低于铷原子钟，在同步过程中会引入影响较大的系统误差，故采用卡尔曼滤波消除系统噪声，提高时钟同步的精度。

主从时钟通过以太网交换机相连，同样分别测试K-PTP和标准PTP同步的时钟同步精度，K-PTP的卡尔曼滤波参数，观测噪声，测试结果的对比图如图4.6所示。

D:\Desktop\板板以太网.emf

图4.6 以太网连接方式K-PTP和标准PTP时钟同步精度

Figure4.6 Clock Synchronization Accuracy of K-PTP and Standard PTP under Ethernet Connection

主从时钟通过以太网连接的标准PTP测试的时钟偏差均值为29.04ns，对应的标准差为34.23ns；K-PTP的时钟偏差均值为22.61ns，标准差为27.53ns。

除了同步模块时钟频率的稳定度较低的因素，以太网相连的方式同样给系统引入了较强的系统噪声，在该情况下的卡尔曼滤波抑制噪声的效果更为明显。

# 非对称路径延时校正算法

## 5.1 非对称路径延时校正模型

### 5.1.1 模型描述

在实际应用中，为了拓扑网络结构，PTP网络中会加入路由交换设备，报文在网络路由交换设备中进行传输时常常会由于网络负载等因素产生延时抖动，这些网络设备会对网络中传输的数据包进行缓存、再转发等操作，当网络中流量过大，PTP报文会因为排队传输导致报文的接收时间戳信息异常，由于主从路径延时非对称，导致错误地计算主从时钟偏差，这些异常的时钟偏差严重偏离实际，称为离群值，将会严重影响时钟同步的精度[56]。为了解决这一问题，IEEE 1588v2协议定义了边界时钟和透明时钟代替普通的路由交换设备，边界时钟将整个大的网络分成多个小的局域网，在一定程度上降低了报文传输延时抖动。但随着网络拓扑结构逐渐复杂，主从层次逐渐增多，边界时钟所带来的降低延时抖动的影响越来越弱，时钟同步精度也因此而下降，且网络规模增大，必须考虑边界时钟大量使用带来的成本问题。因此需要一种在使用普通路由交换设备的情况下，降低因网络负载增大而产生的离群值对同步精度的影响的方法。

由于卡尔曼滤波的更新过程中需要当前时刻的时钟偏差观测值，若该值为离群值将会使得最后的估计值也会明显偏离实际，使得计算的最优估计值严重失真。故需要提高同步的鲁棒性，使其在网络负载流量较大的情况下，最后计算得出的时钟偏差更接近实际。

为了解决非对称延时的问题，采用一种非对称路径延时校正（Delay Asymmetry Correction, DAC）模型[57]。在非对称路径延时的情况下，该模型确定正确的主从钟时钟偏差来实现高精度同步，并且它不会修改标准内容，只对记录的时间戳和计算的时钟偏差进行处理，不会影响基本的报文交换。DAC模型分为两个测试阶段，在第一阶段，使用主从时钟非对称延时比率R，进行“R”测试；第二阶段用来更新通过“R”测试的时钟偏差值。DAC模型的基本原则是确保在网络中的高流量负载的情况下只使用良好的时钟偏差来正确地更新从属时钟。此外，DAC模型保存了通过测试阶段良好的时钟偏差值，若当前周期计算的时钟偏差为离群值，则从时钟使用保存的时钟偏差调整本地时钟。DAC模型的第一个测试阶段详细流程图如图5.1所示。



图5.1 DAC模型第一个测试阶段

Figure5.1 The First Test Step of The DAC Model

首先，主从时钟使用基于IEEE 1588v2协议标准中定义的延时请求响应测量机制交换PTP报文，计算时钟偏差。由于网络流量不可预测，并且会随着时间的推移而变化，同步过程中主要的延时变化会在正向路径(主到从)或反向路径(从到主)上发生。因此，使用非对称延时比率R来判断当前同步周期的正向和反向路径延时是否为非对称。

 (5.1)

式(5.1)中，*t1*~*t4*为延时请求响应测量机制中从时钟获得的时间戳，*offset*为主从时钟偏差，*delayms*和*delaysm*分别为正向路径延时和反向路径延时。非对称延时比率R的值表示当前正反向路径延时的对称情况，当时钟偏差相对于实际的传输延迟较小时，表示的效果和程度更优。

在获得当前周期的时钟偏差后，计算主从时钟路径的非对称延时比率*R*。若时钟偏差通过边界条件才有可能被认为是良好的偏差值。*R*的取值范围是基于在正常网络负载情况下，通过进行多次较长时间的实验测试收集的统计信息来确定的，图5.2为时钟同步模块在交换机连接状态和正常网络负载情况下的*R*值分布。

D:\Desktop\论文插图\R测试分布.emf

图5.2 R值的频率分布直方图

Figure5.2 Frequency Distribution Histogram of R Value

一旦当前时钟偏差没有通过“R”测试，表示当前的主从路径延时存在高度不对称，则使用先前存储的已经通过两次测试的偏差值替换当前的偏差值，从时钟根据替换的时钟偏差来调整本地时钟。

由于R边界条件的取值为实验测试的估计值，因此通过“R”测试的偏差值会有高于或低于预期偏差值的可能，这样的值同样会被认为是离群值，当使用该离群值调整从时钟时，主从同步精度可能会受到严重影响。故设定第二次测试，以保证用来调整本地时钟的时钟偏差值足够准确。图5.3为第二次测试流程图。



图5.3 DAC模型第二个测试阶段

Figure5.3 The Second Test Step of The DAC Model

如图所示，第二次测试的输入为通过“R”测试的第*t*个同步周期的时钟偏差值，随后令偏差值的预测值为：

 (5.2)

由式(5.3)设置当前时钟偏差第二次测试的边界条件，以确保被保存的偏差值的良好性，定义如下：

 (5.3)

式(5.2)中的为第*p*个通过两次测试的时钟偏差值。公式(5.3)定义的边界条件将当前计算的时钟偏差值与其预测值进行比较，若偏差值通过边界条件，则使用该偏差值调整本地时钟，同时保存通过该边界条件的最近十个周期的偏差值，对这些偏差值做滑动平均计算得出*saved*\_*offsett*，其计算如下：

 (5.4)

若未通过该边界条件，则使用*saved*\_*offsett*来代替当前周期计算的偏差值调整时钟。

综上所述，DAC两次测试目的都是为了确保从时钟使用良好的时钟偏差值来更新本地时钟，以此来保证非对称路径延时下的同步精度。

### 5.1.2 理论验证与同步精度测试

为了更好地模拟高网络负载、主从路径延时非对称的情况，使用以太网交换机连接两个时钟同步模块来测量时钟同步精度。为了验证DAC模型的有效性，在将其部署到时钟同步模块之前，先模拟高网络负载导致非对称延时的情况，测试主从时钟在该情况下的同步精度。

通过在主从时钟上运行不断向彼此发送数据包的应用程序，模拟高负载网络的情况，同时对正向路径引入70%的流量负载，反向路径引入30%的流量负载以模拟主从路径延时非对称的情况，实验测试结果如图5.4所示。

D:\Desktop\only_load.emf

图5.4 非对称延时下的时钟同步精度

Figure5.4 Clock Synchronization Accuracy under Asymmetric Delay

由上图可以看出，高网络负载导致的非对称延时严重影响了时钟同步的精度，必须解决此情况下的同步精度，提高系统的鲁棒性。

在标准PTP同步中引入DAC模型，设置*P*和*I*的值分别为0.5和0.0625进行时钟同步精度测试，引入DAC前后的对比测试结果如图5.5所示。

D:\Desktop\LoadAndDAC.emf

图5.5 DAC模型和标准PTP时钟同步精度对比

Figure5.5 Comparison of Clock Synchronization Accuracy of DAC Model and Standard PTP

D:\Desktop\DAC下的偏差值.emf

图5.6 DAC模型下的时钟同步精度

Figure5.6 Clock Synchronization Accuracy under DAC Model

DAC模型的引入有效抑制了非对称路径延时带来的影响，提高了高网络负载下的系统稳定性和时钟同步精度。DAC模型下测试的时钟偏差均值为23.73ns，对应的标准差为30.01ns。

## 5.2 时钟模型线性估计算法

### 5.2.1 算法描述

由5.1.2节的测试结果可以看出，DAC模型提高了主从时钟同步过程的鲁棒性，抑制了由高网络负载引起的非对称路径延时对同步带来的影响，在一定程度上保证了同步的精度。但由图5.6可以看出，在一段时间内连续出现了主从路径延时高度不对称的情况。对于非对称路径延时，DAC模型算法会使用保存的偏差值来替代当前周期计算的时钟偏差，故在非对称延时的持续时间内，出现了一段连续的*saved*\_*offset*值。在连续的高度不对称路径延时的情况下，这种以*saved*\_*offset*值替代当前偏差值的处理方式有失妥当，这也是DAC模型的弊端。

为了解决这一问题，本文提出一种时钟模型的线性估计算法（Linear Estimation of Clock Model, LEC），旨在主从路径延时非对称的情况下，计算准确的主从时钟偏差，提高同步精度。时钟模型一般是复杂且非线性的，LEC算法假设时钟在短时间内是线性的，通过从时钟由PTP报文交互获取的四个时间戳来实现从时钟模型准确的线性估计，并实时更新时钟斜率。

在对时钟模型进行线性估计前，先建立时钟的线性模型。考虑主从时钟遵循线性函数和：

， (5.5)

其中，*t*为主时钟的时间，将*M*(*t*)作为参考时钟；*S*(*t*)是*t*的线性函数，作为时钟斜率为*k*的从时钟模型，*θ*(0)为主从时钟初始偏差。

LEC算法的基础是从时钟通过PTP协议报文交互获得精确的硬件时间戳，对这些时间戳进行应用处理，通过主时钟作为参考时钟来估计从时钟线性模型的参数，包括时钟斜率*k*和截距*θ*(0)。获取在*n*个连续同步周期的主时钟的PTP时间戳和从时钟的PTP时间戳，并使用这些时间戳为从时钟模型的线性估计做铺垫，找出两个线性函数，一个在从时钟线性函数*S*(*t*)之上，另一个在*S*(*t*)之下，分别为*S*(*t*)的线性上届和线性下届函数，由两者的平均值估计从时钟的线性函数。图5.7展示了单个同步周期PTP时间戳和主从时间情况，可以看出点始终位于*S*(*t*)之上，同时点始终低于*S*(*t*)，LEC利用这一特性来准确估计*S*(*t*)。

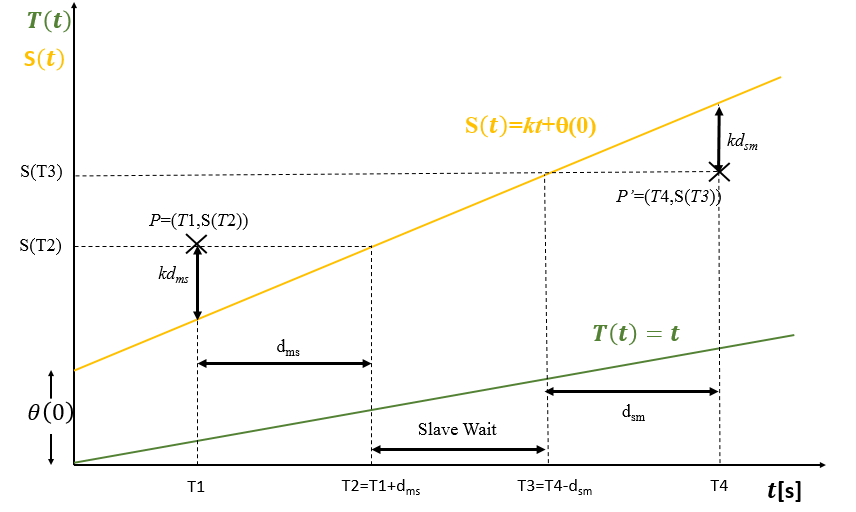


图5.7 一个同步周期内主从时钟的PTP时间戳和时间

Figure5.7 PTP Timestamp and Time of The Master and Slave in one Synchronization Period

时钟斜率*k*和截距*θ*(0)的确定是构造*S*(*t*)线性函数的必要条件，通过*n*个连续同步周期内的PTP时间戳，，，来确定从时钟线性函数。表示在第*n*个同步周期获取的PTP时间戳，*n*至少为2。对*S*(*t*)线性函数进行估计前，先找出*S*(*t*)的线性上届和下届函数。

*S*(*t*)的线性上届函数：使用连续两个正常的同步周期内获得的PTP时间戳，，，来定义两个点和，两者均始终位于*S*(*t*)之上。利用两点可以确定一个线性函数，即为*S*(*t*)的线性上届函数。

 (5.6)

其中，，。

*S*(*t*)的线性下届函数：函数的确认方式与线性上届函数相同，通过PTP时间戳，，，定义点和，两点始终位于*S*(*t*)之下。由此确定*S*(*t*)的线性下届函数为：

 (5.7)

其中，，。

至此，已经通过PTP时间戳确定了*S*(*t*)的线性上届和下届函数，如图5.8所示。

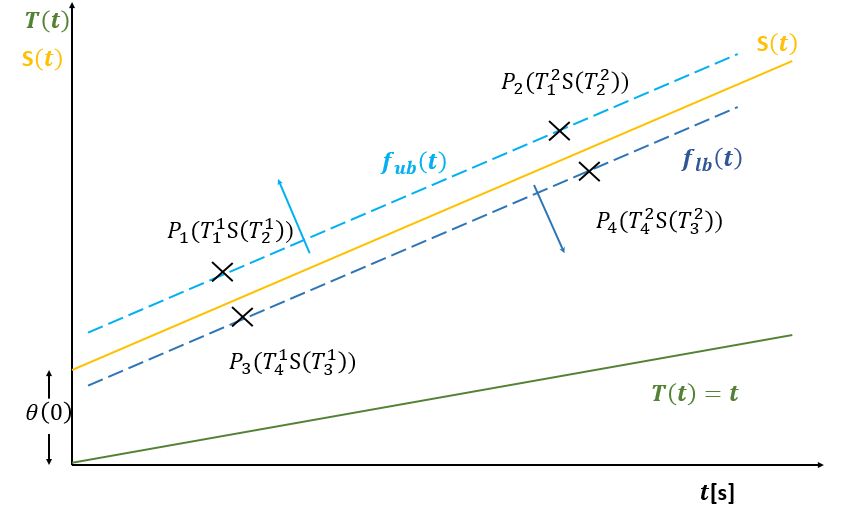


图5.8 从时钟模型的线性上届和下届函数

Figure5.8 Upper and Lower Linear functions of Slave Clock

令，，估计*S*(*t*)的线性函数：

 (5.8)

由网络中间件的排队延时转发引起的非对称路径延时主要影响了报文的发送或接收时间戳，导致通过PTP协议标准计算的路径延时严重失真，图5.9中展示了正向路径延时远大于反向路径延时的情况，可以看到点明显偏离了*S*(*t*)的线性上届函数。

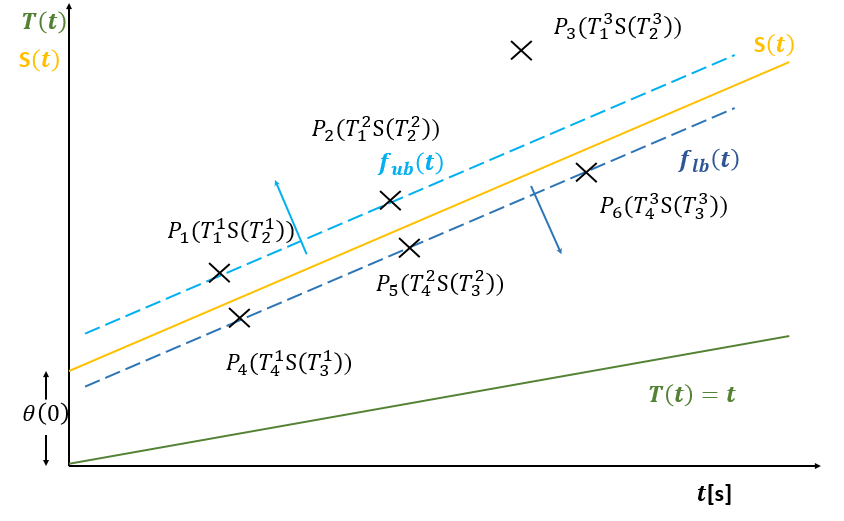


图5.9 非对称延时下的时间戳点

Figure5.9 Timestamp Point under Delay Asymmetry

假设前两个正常同步情况下的单向路径延时分别为*delay*1和*delay*2，两者均值为*delay*。当主从时钟路径延时高度不对称时，会导致同步报文到达从时钟和主时钟的时间*t*2和*t*4异常，从而无法准确计算当前同步周期的时钟偏差，但可以通过式(5.8)和*delay*计算得出当前周期同步报文发送和接收时间戳的估计。

 (5.9)

 (5.10)

其中和分别为当前同步周期*t*2和*t*4的估计。由式(5.9)和(5.10)可以估计出当前周期的同步报文时间戳，以此来校正时钟偏差，对正常同步情况下的时钟偏差进行准确估计。结合LEC算法后的主从时钟偏差计算流程图如下。



图5.10 LEC算法的时钟偏差计算流程图

Figure 5.10 The Clock Deviation Calculation Flowchart of The LEC

如图5.10所示，算法以时刻从时钟模型的估计参数、和时刻与时刻单向路径延时的平均值以及时刻的时间戳端点为输入开始运行。在当前周期的同步过程完成后，计算出当前同步周期的主从时钟偏差，并计算非对称延时比率*R*，随后判断*R*的值是否满足边界条件，若不满足则将时钟偏差视作离群值，通过LEC算法估计同步报文到达从时钟的时间戳*t*2和Delay\_Req报文到达主时钟的时间戳*t*4，通过时间戳的估计值和计算当前同步周期时钟偏差估计，同时将当前同步周期的*t*1、*t*2及估计的*t*3、*t*4时间戳作为LEC算法进行线性上、下届函数估计的端点，结合上一同步周期的时间戳端点计算和保存当前同步周期的时钟斜率及其截距和单向路径延时的估计。

若满足边界条件，则计算时钟偏差的预测值并通过预测值设置边界条件，继续对当前周期的时钟偏差进行判断。预测值的计算及边界条件的设置如公式(5.2)和(5.3)，式(5.2)中的在此处表示为第*p*个调整从时钟的偏差值。若当前时钟偏差满足该边界条件，则表明当前时钟偏差足够准确，能够直接以此偏差值来调整时钟，同时计算并保存当前同步周期的单向路径延时和时钟斜率及其截距，否则将该时钟偏差视作离群值进行处理。

### 5.2.2 算法验证与同步精度测试

D:\Desktop\LEC.emf

图5.11 LEC算法和标准PTP时钟同步精度对比

Figure5.11 Comparison of Clock Synchronization Accuracy of LEC and Standard PTP Clock

D:\Desktop\AfterLEC.emf

图5.12 LEC算法下的时钟同步精度

Figure5.12 Clock Synchronization Accuracy under LEC

图5.11为引入了LEC算法的PTP同步和标准PTP同步的同步精度对比，测试环境和5.1.2节的实验环境相同。如图所示，LEC算法的引入同样较好的抑制了时钟偏差离群值对同步精度的影响，提高了系统的鲁棒性。此外，由图5.12可以看出，LEC通过估计当前同步周期时间戳的数值来对时钟偏差进行估计，弥补了DAC模型的弊端，到达了偏差均值23.48ns，标准差28.33ns的时钟同步精度。

# 总结与展望

## 6.1 总结

本文在深入理解IEEE 1588v2协议的基础上，设计了符合IEEE 1588v2协议的时钟同步模块，并通过卡尔曼滤波算法对时钟偏差、路径延时以及频率偏差进行滤波，同时使用时钟频率补偿算法补偿从时钟的时钟频率，到达较高的同步精度。此外，加入非对称路径延时校正算法，以此保证在高网络负载情况下的同步精度。本文的主要工作如下：

（1）首先介绍了IEEE 1588v2协议标准的研究背景以及国内外现状。紧接着分别阐述了IEEE 1588v2协议的基本概念以及同步原理。对时钟模型、协议报文和影响同步精度的因素进行了分析和描述。

（2）构建了以AM3358和AR8031\_AL1A为核心的时钟同步模块，详细介绍了AM3358处理器的功能以及对PTP报文时间戳的获取。搭建了PTP同步的软件平台，对Linux系统的PTP硬件时钟子系统以及硬件时间戳的获取方式进行了详细描述。

（3）由于报文在网络传输过程中会引入系统误差，影响时钟同步精度，因此本文采用卡尔曼滤波算法来消除误差，对时钟状态的建模和滤波过程进行了详细阐述，同时将滤波后的时钟偏差做为PI控制器的输入，通过计算的频率的补偿值来调整本地时钟频率。在不同的测试条件下，主从时钟同步精度可达纳秒级。

（4）最后对高网络负载情况下的时钟同步进行了讨论，通过以太网交换机连接两个时钟同步模块，并模拟了由高网络负载导致非对称延时的实验环境。分别对标准PTP和引入DAC模型的PTP进行实验测试，结果表明DAC模型有效抑制了非对称路径延时带来的影响，提高了高网络负载下的系统稳定性及时钟同步精度。但DAC模型在连续的高度不对称路径延时的情况下的处理方式略失妥当，故本文提出一种时钟模型线性估计（LEC）的方法来解决这一问题，LEC算法通过估计当前同步周期时间戳的数值来对时钟偏差进行估计，在解决DAC模型弊端的同时，保证了时钟同步的精度。

## 6.2 展望

本文对IEEE 1588v2协议的相关理论进行了深入的研究与讨论，设计并实现了符合协议标准的时钟同步模块。此外，针对非对称路径延时会对同步精度产生较大影响，本文也提出了PTP协议的改进方案，以此来保证非对称延时下的同步精度，取得了预期成果。但是，对以下几个方面还需要进一步研究：

（1）本文实验的网络结构为最简单的点对点结构，还需在更为复杂、规模更大的网络结构下进行研究，测试多跳网络下的时钟同步精度。

（2）由PI控制器进行时钟伺服需要确定合适的*P*、*I*值，两者的取值往往需要通过大量的测试实验来获得，故还需要研究更为稳定、智能的时钟伺服方案。

（3）采用支持IEEE 1588v2协议的PHY芯片，在物理层为报文加盖时间戳，进一步提高同步精度。

# 致谢

世间遗憾事，往往起始于再见二字。而世间幸运事，又往往在于之后真正再见之时**。**

在民大三年的求学生涯即将接近尾声，回顾三年的学习生活，走过的每一步都离不开关心和帮助我的老师和朋友们，每当想起他们，心中总是满怀着敬意与感激。

首先感谢我的在校导师彭良福老师。彭老师总是以高度的责任心关注着我研究课题中的每一个环节，以严谨的态度指导我学业中遇见的每一个问题，并想方设法地为学生创造良好的学习和生活条件。彭老师严谨的治学态度、平易近人的师长风范以及渊博的学术知识值得我铭记，是我终身学习的楷模。

同时还要感谢校外导师刘彬工程师。刘老师有着丰富的工程实践经验，在课题研究过程中，刘老师对出现的每一个问题都会悉心指导，对本课题的完成给予了极大的帮助。

感谢我的同门、同窗、实验室的小伙伴们，自晨起至深夜，我们并肩奋战，在学习、生活和工作中互帮互助、相互扶持。三年来的陪伴，让我们之间不仅是同学，更是亲密的朋友，这份珍贵的友谊将永远铭记。

感谢我的家人，是他们为我提供了精神和物质上的支持，让我能够心无旁骛地追逐目标，他们无私的爱与期望是我前进的动力，让我能乐观地面对生活中的任何难题。感恩我的家人，感恩他们为我付出的一切。

“路漫漫其修远兮，吾将上下而求索。”研究生阶段的生活告一段落，虽不知前路几许，但笃定未来可期。在即将踏上的工作之路上，将会一直谨记和合偕习，自信自强！

# 参考文献

[1] 王旭. 分布式测试系统时钟同步及触发研究与实现[D]. 中北大学, 2020.

[2] 王晋祺. 基于IEEE 1588v2协议的分布式测试系统时统系统研究[D]. 中北大学, 2019.

[3] 高志昊. 基于物理层时钟频率补偿的IEEE1588网络时钟同步研究与实现[D]. 桂林电子科技大学仪器仪表工程, 2017.

[4] IDREES Z, GRANADOS J, SUN Y, et al. IEEE 1588 for Clock Synchro-nization in Industrial IoT and Related Applications: A Review on Contribu-ting Technologies, Protocols and Enhancement Methodologies[J]. IEEE ACCESS, 2020,8: 155660-155678.

[5] PUTTNIES H, DANIELIS P, SHARIF A R, et al. Estimators for Time S-ynchronization—Survey, Analysis, and Outlook[J]. IoT

, 2020,1(2): 398-435.

[6] 胡昌军, 吕博, 缪新育. 5G同步组网架构及关键技术探析[J]. 信息通信技术与政策, 2020(04): 36-40.

[7] 刘艳亮, 张海平, 徐彦田, 等. 全球卫星导航系统的现状与进展[J]. 导航定位学报, 2019,7(01): 18-21.

[8] 王军. IEEE1588与同步以太网技术的研究与实现[D]. 西安电子科技大学, 2016.

[9] 中科院国家授时中心监测到伽利略卫星导航系统时间出现异常[EB/OL]. [2021-3-18]. http://news.sciencenet.cn/htmlnews/2020/12/450281.shtm?bsh\_bid=5572418383.

[10] 曾庆琦. 高精度网络时间同步的研究与实现[D]. 中国科学院大学(中国科学院武汉物理与数学研究所), 2020.

[11] 陈希, 滕玲, 高强, 等. NTP和PTP协议的时间同步误差分析[J]. 宇航计测技术, 2016,36(03): 35-40.

[12] 仲婧. IEEE 1588在智能变电站的应用及测试分析[J]. 信息通信, 2020(12): 91-95.

[13] IEEE Standard for a Precision Clock Synchronization Protocol for Networ-ked Measurement and Control Systems[Z]. 20021-154.

[14] IEEE Standard for a Precision Clock Synchronization Protocol for Networ-ked Measurement and Control Systems[Z]. 20081-300.

[15] LIPIŃSKI M, WŁOSTOWSKI T, SERRANO J, et al. White rabbit: a PTP application for robust sub-nanosecond synchronization, 2011.

[16] IEEE Standard for a Precision Clock Synchronization Protocol for Networ-ked Measurement and Control Systems[Z]. 20201-499.

[17] LANGER M, HEINE K, SIBOLD D, et al. A Network Time Security Ba-sed Automatic Key Management for PTPv2.1, Sydney, NSW, Australia, 2020.

[18] GIORGI G, NARDUZZI C. Performance Analysis of Kalman-Filter-Based Clock Synchronization in IEEE 1588 Networks[J]. IEEE Transactions on I-nstrumentation and Measurement, 2011,60(8): 2902-2909.

[19] GIORGI G, NARDUZZI C. A resilient Kalman filter based servo clock, 2013.

[20] KERÖ N, PUHM A, KERNEN T, et al. Performance and Reliability Aspe-cts of Clock Synchronization Techniques for Industrial Automation[J]. Proc-eedings of the IEEE, 2019,107(6): 1011-1026.

[21] EXEL R, RING F. Improved clock synchronization accuracy through opti-mized servo parametrization, Lemgo, Germany, 2013.

[22] KARTHIK A K, BLUM R S. Robust Clock Skew and Offset Estimation for IEEE 1588 in the Presence of Unexpected Deterministic Path Delay Asymme-tries[J]. IEEE Transactions on Communications, 2020,68(8): 5102-5119.

[23] PUTTNIES H, DANIELIS P, TIMMERMANN D. PTP-LP: Using Linear Pr-ogramming to Increase the Delay Robustness of IEEE 1588 PTP, 2018.

[24] 庄晓燕. 分布式测试系统时钟网络结构与同步算法研究[D]. 电子科技大学, 2013.

[25] 孔祥瑞. 基于IEEE1588协议的铁路时间同步技术研究[D]. 燕山大学, 2020.

[26] 史仲渊. 基于改进卡尔曼滤波算法的实时以太网时钟同步系统的研究[D]. 浙江理工大学, 2019.

[27] 陶稳静. 基于IEEE 1588协议的精确时钟同步系统的研究与设计实现[D]. 合肥工业大学, 2019.

[28] 苏朕轶. 基于SoC的IEEE1588时钟同步系统的设计与研究[D]. 西安电子科技大学, 2018.

[29] 祝托. 基于IEEE1588协议的从时钟同步技术研究[D]. 北京邮电大学, 2018.

[30] 焦洁. 非对称链路PTP时钟同步算法改进[D]. 西安理工大学通信与信息系统, 2015.

[31] 郭爱英. 时间触发以太网同步技术的研究与实现[D]. 电子科技大学, 2013.

[32] 李培基, 李卫, 朱祥维, 等. 网络时间同步协议综述[J]. 计算机工程与应用, 2019,55(03): 30-38.

[33] 陶源, 吴婷. 5G高精度时间同步组网方案研究[J]. 邮电设计技术, 2021(01): 77-82.

[34] 中兴通讯超高精度时间同步网，支撑5G商用规模化部署-中兴通讯技术（简讯）[EB/OL]. [2021-3-18]. http://www.zte.com.cn/china/about/magazine/zte-technologies/2020/3-cn/6/2.

[35] 1588PTP板卡P66/P68[EB/OL]. [2021-3-18]. https://www.coolshark.com/zhipin/qianrushijiejuefangan/P60yuanxingban/.

[36] SyncEdge时频同步分析仪[EB/OL]. [2021-3-18]. http://www.xgtime.com.cn/pro\_detail\_id\_17.html.

[37] 寇海洋. 基于IEEE1588协议的从时钟的设计与实现[D]. 大连理工大学, 2016.

[38] LEE S, KANG J, CHOI S S, et al. Design of PTP TC/Slave Over Seaml-ess Redundancy Network for Power Utility Automation[J]. IEEE Transactio-ns on Instrumentation and Measurement, 2018,67(7): 1617-1625.

[39] 王冠. IEEE1588v2时钟同步技术的研究与实现[D]. 武汉邮电科学研究院, 2012.

[40] 冀天宇. 基于IEEE1588高精度网络时钟同步技术的研究与应用[D]. 北京邮电大学, 2015.

[41] 邢永平. IEEE1588精密时钟同步协议的研究与实现[D]. 北京交通大学, 2013.

[42] TIEN N X, RHEE J M. A Novel Algorithm for Establishing a Balanced Sy-nchronization Hierarchy with Spare Masters (BSHSM) for the IEEE 1588 Pr-ecision Time Protocol[J]. Energies, 2017,10(10): 1469.

[43] SHKLIAREVSKYI I, SHKLIAREVSKYI O, DYADENKO E, et al. IEEE 1588 protocol profiles' comparative analysis according to different applicati-ons and standards[J]. 2016 IEEE International Symposium on Precision Cl-ock Synchronization for Measurement, Control, and Communication (ISPC-S), 2016: 1-4.

[44] 王旭东. 多功能IEEE1588网络时钟源的研发[D]. 华中科技大学, 2016.

[45] 邓德杨. 基于以太网交换机的IEEE1588V2时钟同步协议研究与实现[D]. 杭州电子科技大学, 2014.

[46] 陈金凤. IEEE1588网络时延测量研究与实现[D]. 中国科学院研究生院（国家授时中心）, 2013.

[47] 陈大峰. 基于IEEE1588的工业以太网实时同步技术的研究与开发[D]. 江南大学, 2013.

[48] LU Y, WANG D, ZHU S. Design of an embedded multi-antenna satellite data acquisition system based on ARM-Linux, 2018.

[49] 朱望纯, 钟震林, 覃斌毅. 嵌入式Linux设备的高精度IEEE 1588时钟同步实现[J]. 计算机测量与控制, 2014,22(05): 1619-1622.

[50] 宋宝华. Linux设备驱动开发详解[M]. 机械工业出版社, 2015.

[51] SHEREEN E, BITARD F, DÁN G, et al. Next Steps in Security for Time Synchronization: Experiences from implementing IEEE 1588 v2.1, 2019.

[52] COCHRAN R, MARINESCU C. Design and implementation of a PTP clo-ck infrastructure for the Linux kernel, 2010.

[53] FARAGHER R. Understanding the Basis of the Kalman Filter Via a Simp-le and Intuitive Derivation [Lecture Notes][J]. IEEE Signal Processing Ma-gazine, 2012,29(5): 128-132.

[54] GIORGI G. An Event-Based Kalman Filter for Clock Synchronization[J]. IEEE Transactions on Instrumentation and Measurement, 2015,64(2): 449-457.

[55] XU X, XIONG Z, SHENG X, et al. A New Time Synchronization Metho-d for Reducing Quantization Error Accumulation Over Real-Time Networks: Theory and Experiments[J]. IEEE Transactions on Industrial Informatics, 2013,9(3): 1659-1669.

[56] 符波行. 非对称通信链路中网络时间同步算法的研究与应用[D]. 华南理工大学, 2013.

[57] RAHMAN M A, KUNZ T, SCHWARTZ H. Delay Asymmetry Correction Model for Master-Slave Synchronization Protocols, 2014.