# 论文题目：IEEE1588高精度时钟同步技术的改进与实现

**摘要**

**Abstract**

# 第一章 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。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[1]。

现阶段网络系统逐渐庞大复杂，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。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。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。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，超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。

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。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。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存在无规律的跳变，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。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。

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。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。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，该版本是针对局域网里多播环境制定。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。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。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。

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。Giorgi et.al45.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。同时在[93]中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。作者74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。Exel et al95.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。Anantha K. Karthik [78]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。Puttnies H.等人[59]介绍了提高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。

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。

孔祥瑞针对PTP中的非对称延时链路进行了分析，提出了一种基于三层BP神经网络的时钟偏差补偿方案，将主从时钟偏差输入到神经网络模型，通过输出结果来补偿时钟偏差，从而提高PTP非对称延时链路的同步精度。

史仲渊描述了影响同步精度的主要因素，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抓包工具了补偿软件时间戳，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。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。

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。

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。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。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。Vitesse公司产出了一种内置有CPU和IEEE l588协议的交换芯片——VSC7460，运行相应的应用程序即可进行时钟同步，方便调试。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。

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MSTP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。

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。

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。

## 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，在其基础上研究以太网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，通过其系统驱动对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。同时在计算链路延时时，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间偏差来计算频率偏差并以此补偿时钟的频率漂移，使得从时钟较稳定的运行。此外，考虑当网络中流量增大，负荷严重时会导致往返路径延时的严重非对称，获取错误的时间偏差，极大程度的影响了同步的精度，引入非对称延时校正算法避免该情况的发生。

### 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 1588v2.0协议的基本概念，包括IEEE 1588v2.0时钟模型、同步原理以及报文的类型与格式，分析了IEEE 1588v2.0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。

# 第二章 IEEE 1588精确时钟同步协议

IEEE 1588协议进行同步的概念之一为IEEE 1588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。两个自由振荡的时钟的时间基准一般都有偏差，IEEE 1588协议是通过时钟间的报文交换，利用报文里携带的时间戳来进行时钟同步。

## 2.1 IEEE 1588时钟模型

### 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。

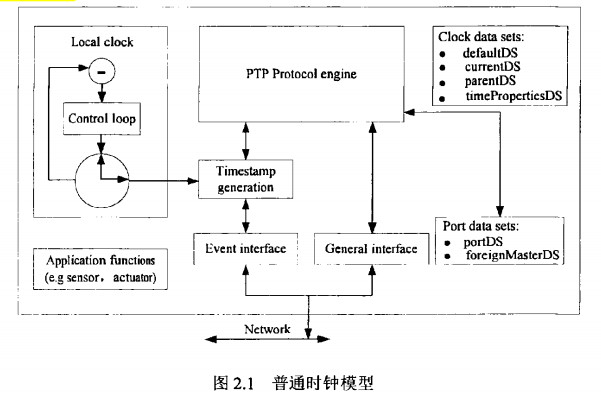


图2.1 普通时钟模型

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，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。

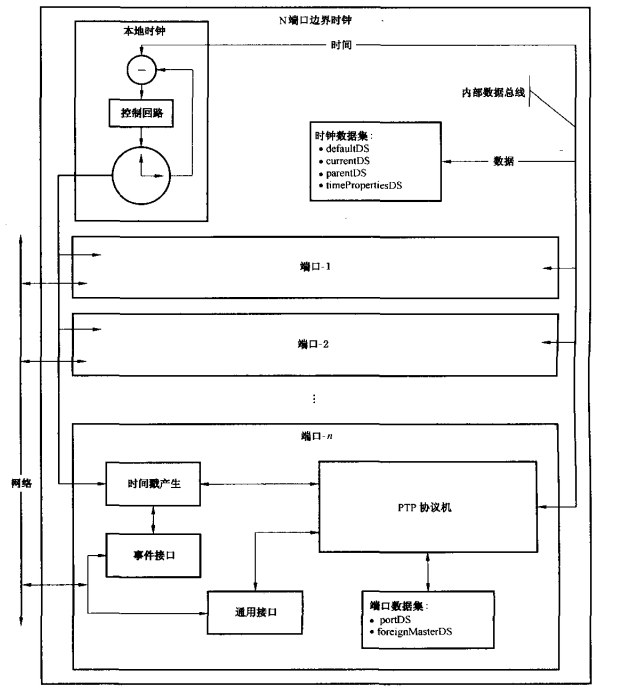


图2.2 边界时钟模型

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**

### 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。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，详细内容将在下节介绍。

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。此处介绍该透明时钟处理Sync报文的过程，如图2.4所示。

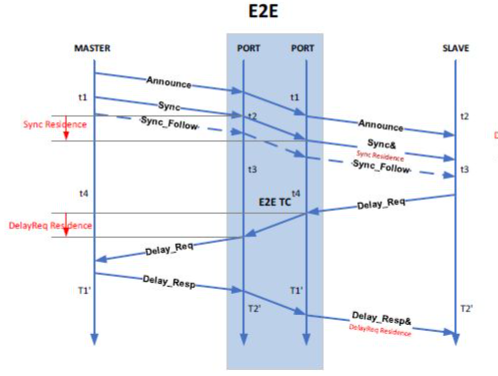


图2.4 端到端透明时钟处理Sync报文的过程

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。

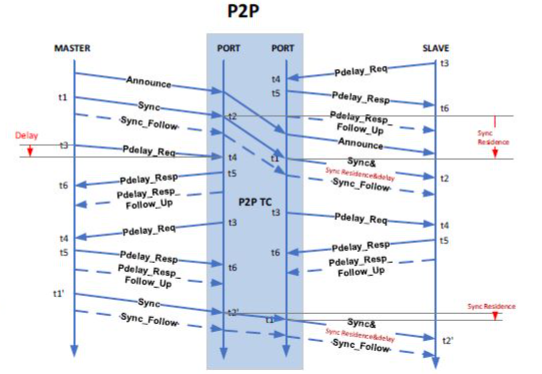


图2.5 点到点透明时钟处理Sync报文的过程

Figure 2.5 Process of Peer-to-peer Transparent Clock dealing Sync message

P2PTC将主从时钟间的路径分为两个链路，并参与计算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。

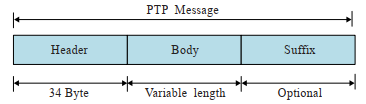


图2.6 PTP报文的组成

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。

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报文的主体结构见表2.1。

表2.1 Sync和Delay\_Req报文的主体结构

|  |  |  |  |
| --- | --- | --- | --- |
| 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报文的主体结构分别见表2.2和表2.3。

表2.2 Pdelay\_Req报文主体结构

|  |  |  |  |
| --- | --- | --- | --- |
| 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

|  |  |  |  |
| --- | --- | --- | --- |
| 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通用报文头

|  |  |  |  |
| --- | --- | --- | --- |
| 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。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。

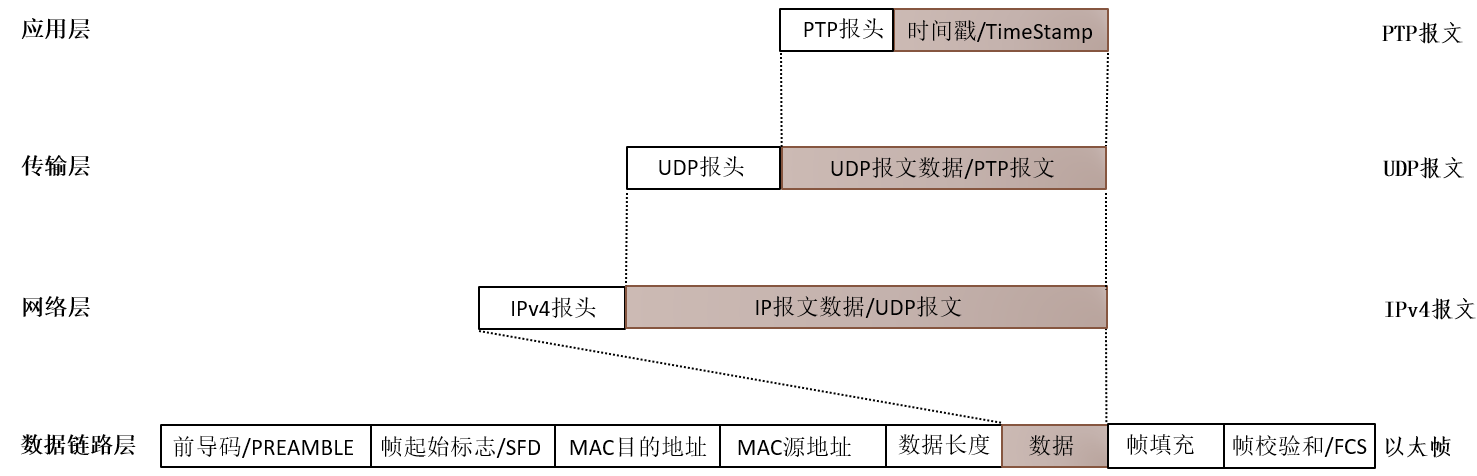


图2.7 PTP报文的UDP/IPv4封装

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。

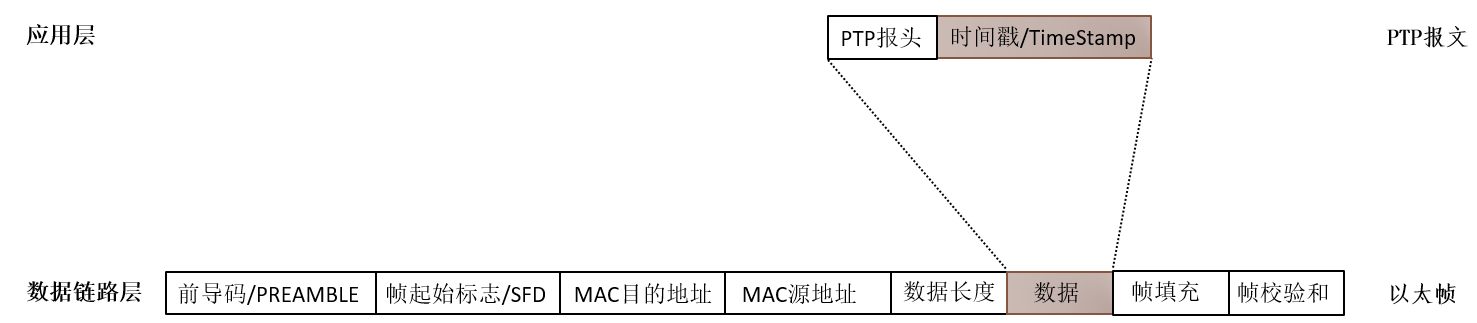


图2.8 PTP报文的IEEE802.3/Ethernet封装

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。首先检测数据帧中的以太网帧类型域值，若该值为0x0800，则该报文为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。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。

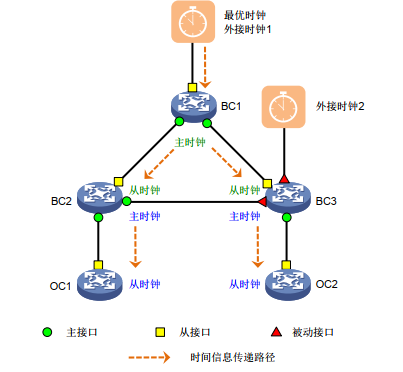


图2.9 PTP主从关系示意图

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，同步示意图如图2.10。

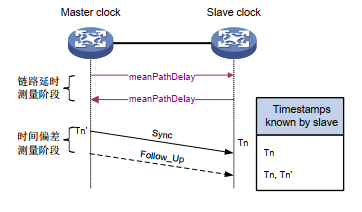


图2.10 IEEE 1588同步示意图

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 1588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。

### 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，同时记录发送Sync报文时主时钟的本地时间*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 延时请求响应机制原理图

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

具体步骤如下：

1. 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*。
2. 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*。
3. 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~t4*四个时间戳，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，在得到时钟B接收到Sync报文的时间*t2*和时钟A发送Sync报文的时间*t1*后，根据公式（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 时钟同步性能影响因素分析

### 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，若在同步过程中，这些因素造成报文的实际往返延时相差过大，以PTP协议中计算单向路径延时的方法将会很大程度降低同步的精度。

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。

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报文在协议栈顶部生成，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产生较大的延时波动。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。从而影响时间戳的精度。

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，MAC控制器在检测到PTP事件报文时触发中断，将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，然而需要专门的硬件支持。



# 第三章 时钟同步模块的设计与实现

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。

## 3.1 时钟同步模块整体框架

使用硬件平台实现IEEE 1588v2协议有三种方案可供选择：

（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，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。

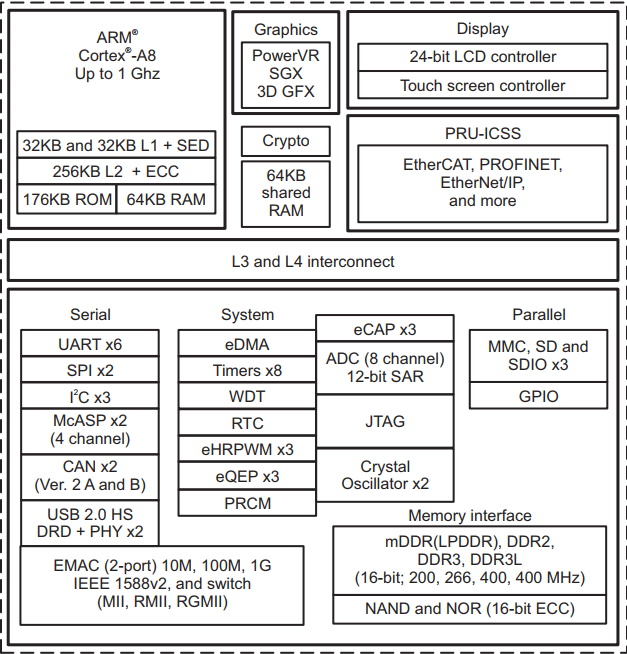
时钟同步模块框架设计如图3.1所示。



图3.1 时钟同步模块框架图

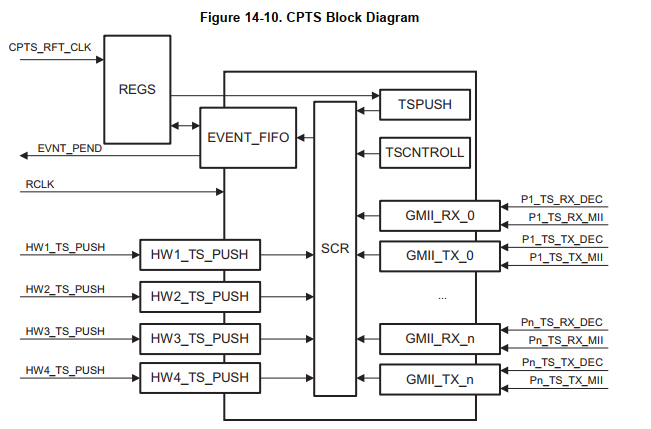
## 3.2 主控芯片的选择

### 3.2.1 AM3358简介



### 3.2.2 AM3358硬件时钟的支持

首先让我们了解一下Sitara SOC有哪些以太网接口，然后这些接口是否也会支持PTP的硬件时钟机制。因此，我们的第一个接口称为CPSW。这是我们传统的以太网MAC/PHY接口。它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。因此，我们在这里将其视为一个物理接口，它可以支持最多两个物理层或两个网络，正如我们所讨论的那样。对于CPSW，硬件时钟支持将由CPTS模块提供，我们在这里通过时钟指示的时钟支持展示了这一点。我们将仔细研究CPT。



## 3.3 以太网模块硬件设计

## 3.4 PTP软件平台设计

### 3.4.1 Linux操作系统

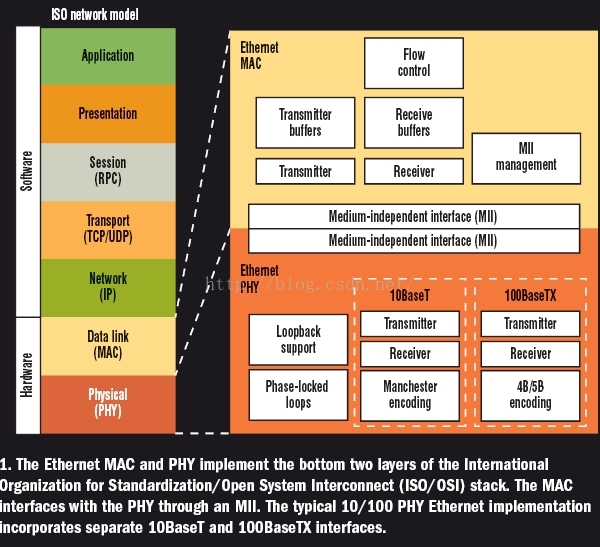
### 3.4.2 Linux PTP简介

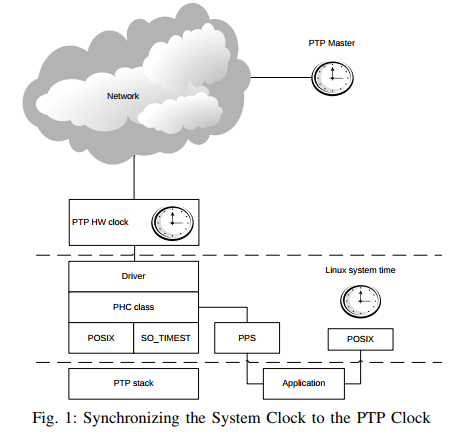
### 3.4.3 PTP硬件时钟

我们在前面的模块中讨论了单步过程。其中一个步骤就是同步能够发送准确的时间戳。因此，在这种情况下，同步将发送12个，不需要后续操作。我们可以在E2E机制和P2P两步点对点过程中看到这一点。点对点使用对等延迟响应后续消息，类似于我们之前看到的延迟响应后续消息。因此，可以在对等或端到端中完成两个步骤，它要求这些后续消息发送准确的时间戳，而不是可能生成的原始时间戳。因此，为了实现这一切，我们需要硬件时钟功能。我们需要能够在物理接口上为数据包添加时间戳。我们将在Sitara系列设备上看看我们对此的支持。因此，这是这张图的放大图。当然，如果你想跟上时间，你首先需要的是时钟。我们可以在CPT上使用几个不同的时钟，这将随着SOC的不同而不同。因此，您需要确保并检查您正在使用的SOC，以及它的时钟支持是什么。这将为您提供CPT时钟支持的分辨率。您将通过寄存器配置时钟，我们还将通过寄存器执行各种其他功能

但这里我们只是展示了一个寄存器位，它决定了您要使用哪个时钟。同样，在每个SOC上，您将有不同的选择。我们将使用CPT对事件进行时间标记。所以我们将有一个EVENT\_FIFO，它将包含事件以及该事件发生的时间戳。当然，您将获得一个事件挂起通知，您可以使用它来确定何时来阅读这些事件并查看它们的时间戳，然后开始采取适当的操作。当然，如果我们看的是PTP，对我们来说非常关键的事件之一是当通过MII接口接收或发送数据时。这里的输入表明，当我们接收和发送数据时，我们可以为放入EVENT\_FIFO的事件获取事件，该事件的时间戳由CPTS时钟派生。您还可以通过硬件事件生成事件。因此，我们可以将计时器与其中一个绑定在一起，例如，如果我们想这样做的话，每隔一秒就会获得一个事件。您也可以通过软件来实现，使用位和寄存器来执行软件推送。这是对CPT的一个很好的概述，以及获取事件时间戳的不同方法。对于PTP来说，最重要的显然是--当事物进入MII接口或离开MII接口时，我们能给它加时间戳吗？这样我们就可以用这些准确的硬件时间戳来计算PTP了吗？所以我们讨论了CPSW和CPTS。

我们了解了Sitara系列SOC如何支持CPSW和PRU-ICSS的MAC到PHY接口的硬件时钟。我们还讨论了这两个不同的时钟模块如何支持PTP协议，以便我们可以获得进出通过CPSW和PRU-ICSS连接的网络的消息的硬件时钟时间戳。





Linux PTP硬件时钟子系统的框图如图1所示。从顶部开始，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并使用标准POSIX时钟功能调整PHC。

### 3.4.4 硬件时间戳的获取

# 第四章 时钟同步算法设计与优化

## 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的最重要和最频繁的数据融合算法之一。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、无线电设备中的锁相环电路、电机系统、目标跟踪、金融等众多领域，卡尔曼滤波最著名的早期应用是在阿波罗号的导航计算机中，它把阿姆斯特朗送上了月球。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。

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[gie]，建立时钟偏差与时钟偏斜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）

故主从时钟偏差为：

 （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，为初始时钟偏差，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）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的观测方程可以表示为：

 （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的观测方程：

 （4-10）

在（4-8）、（4-9）、（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

由式（4-5）、（4-6）和（4-7）可以建立时钟的状态转移方程：

 （4-11）

根据式（4-8）、（4-9）和（4-10）建立时钟的观测方程：

 （4-12）

### 4.1.3 卡尔曼滤波优化

## 4.2 时钟频率补偿算法

## 4.3 非对称路径延时校正算法

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。为了解决这一问题，IEEE 1588v2协议定义了边界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。

# 第五章 测试验证与结果分析

## 5.1 PTP时钟同步精度测试

### 5.1.1 主从直连方式测试

夏光表无卡尔曼，有卡尔曼?

主从时钟直连无卡尔曼18.97，有18.0271,21.19

### 5.1.2 以太网交换机互连方式测试

imx6ull无卡尔曼27，有卡尔曼21

## 5.2 非对称路径延时校正算法验证

网络负载图only load，负载AND抗差卡尔曼，负载无kalmaAND时钟模型估计校正AND抗差卡尔曼延时23.6124

# 第六章 总结与展望

## 6.1 总结

## 6.2 展望

# 参考文献

[1]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.

# 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**。**

**校对报告**

当前使用的样式是 [中华人民共和国国家标准\_GBT\_7714-2015]

当前文档包含的题录共1条

有0条题录存在必填字段内容缺失的问题

所有题录的数据正常

**参考文献**

[1] 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.