申请上海交通大学硕士学位论文

# 基于 IEEE1588 的时钟同步精度研究


**论文作者** _____

**学　　号** _____

**导　　师** _____


**专　　业** 控制科学与控制工程

**答辩日期** 2016 年 5 月

Submitted in total fulfillment of the requirements for the degree of Master
in Control Science and Control Engineering

# On clock synchronization based on IEEE1588

School of Electronic Information and Electrical Engineering
Shanghai Jiao Tong University
Shanghai, P.R.China

May, 2016

# 上海交通大学
# 学位论文原创性声明

本人郑重声明：所呈交的学位论文，是本人在导师的指导下，独立进行研究工作所取得的成果。除文中已经注明引用的内容外，本论文不包含任何其他个人或集体已经发表或撰写过的作品成果。对本文的研究做出重要贡献的个人和集体，均已在文中以明确方式标明。本人完全意识到本声明的法律结果由本人承担。

学位论文作者签名：_____

日　期：_____年_____月_____日

# 上海交通大学
# 学位论文版权使用授权书

本学位论文作者完全了解学校有关保留、使用学位论文的规定，同意学校保留并向国家有关部门或机构送交论文的复印件和电子版，允许论文被查阅和借阅。本人授权上海交通大学可以将本学位论文的全部或部分内容编入有关数据库进行检索，可以采用影印、缩印或扫描等复制手段保存和汇编本学位论文。

本学位论文属于

**保　密** □，在 _____ 年解密后适用本授权书。

**不保密** □。

(请在以上方框内打 ✓)

学位论文作者签名：_____　　指导教师签名：_____

日　期：_____年 ____月 ____日　　日　期：_____年 ____月 ____日

# 基于 IEEE1588 的时钟同步精度研究

## 摘 要

时钟同步问题来源于工业中分布式系统的发展和实时性任务的需求，核心问题是通过同步方法来实现各个子系统之间的时钟一致性，传统的 GPS 同步方法和 NTP(网络时钟同步协议) 等协议由于价格昂贵、同步精度差等原因满足不了工业需求，IEEE1588 精确时钟同步协议协议是目前时钟同步领域最好的同步方法，不过其在实际使用中由于链路延时等因素影响难以达到纳秒甚至亚纳秒的同步精度以及无法保证从时钟的稳定性。

本文从统计分析的角度研究了链路延时、时钟伺服系统及时钟频率漂移补偿的相互关系，以提高 IEEE1588 时钟同步系统精度及稳定性，主要研究内容如下：

(1) 基于链路延时的统计特性，研究了动态阈值法和滑动时间窗的实时监控算法，以提高同步精度；

(2) 在时钟伺服系统中，针对链路时延的阶跃突变干扰采用多模型 PID 控制策略，以优化从时钟的稳定性，减少从时钟的频繁抖动；

(3) 对于时钟频率漂移问题，通过监测 PTP 报文样本是否发生链路堵塞等来进行报文过滤，只选取正常链路传输的 PTP 样本来进行校正，从而提高频率偏差计算的准确度；

(4) 通过 stateflow 时钟同步仿真系统进行仿真分析。仿真结果表明不仅时钟同步系统对多种链路延时抖动的适应性得到了提高，而且整个系统的同步精度和稳定性也得到了提高。

**关键词：**时钟同步 统计算法 频率补偿 时钟伺服系统 stateflow

# On clock synchronization based on IEEE1588

## ABSTRACT

Time synchronization problem originates from the requirement of developing distributed system and real-time tasks. The core problem is to achieve the time consistency among all the subsystems through time-synchronization method. The traditional GPS method and NTP protocol are not able to meet current industrial demand due to their high-cost and unpredictable message latency. Currently, IEEE1588 protocol is the best choice for time-synchronization problem, However, it often cannot achieve the precision level of sub-microsecond or nanosecond and it's not able to guarantee the stability of slave clock in the industrial reality.

From the perspective of mathematical statistics, the objective of this paper is to promote a systematically statistical method to improve the time-synchronization precision and stability. The contributions are shown in the following.

(1) Through the mathematical analysis for transmitting message latency, I decomposed latency into inherent jitter, temporary latency mutation and persistent latency change. In order to improve accuracy, dynamic threshold method and fixed-window real-time detection method are promoted.

(2) In clock servo system, multi-model PID controller strategy is promoted for improving the slave-clock stability when step-mutation occurs. So slave clock system will become more smooth due to the less jitter.

(3) By filtering PTP messages based on the detection of link blocking, qualified PTP messages will be choosed for calcuation of master-to-slave frequency offset.

(4) Based on stateflow time-synchronization simulation system, all the promoted methods above get validated and verified. The simulation results show us those methods above can improve the adaptive ability of time-synchronization system for multiple types of latency changes, and the synchronization precision and stability of whole system as well.

**KEY WORDS:** clock synchronization, statistical method, frenquency compensation, clock servo, stateflow

# 目 录

# 第一章 概述

随着当前网络通信与计算机技术的不断发展，在工业领域中越来越多的基于网络的分布式系统得到了应用。为保证分布式网络系统能够实现精确的数据采集、运行控制等实时性任务，工业中对于各个分布式节点的时间同步精度提出了极为严格的要求，尤其是在大多数以工业以太网为基础的控制系统中，已经逐渐对同步精度提出了微秒级甚至纳秒级别的要求。但是，由于实际测控设备之间固有的时钟差异和时钟数据在网络传输中的随机延时，而且现场设备会由于温度变化、电磁干扰、振荡器老化等多种原因而使得自身时钟并不稳定，使得时间误差随时间不断累积，严重破坏了如变电站、电力监控系统等对时钟同步严格的工业应用。因此，一种更加有效且高精度的时钟同步技术成为测控系统中的关键技术之一。

## 1.1 工业领域中时钟同步的技术现状

时钟同步即保证工业系统内各个节点的时钟能够保持一致，即维护一个全局一致的物理或逻辑时钟，使得系统各节点中与时间有关的信息、时间及行为有一个全局一致的解释[1]。常说的时钟同步，主要包含了时刻同步和时间间隔一致。时刻即表示连续流逝时间的某一个瞬间，而时间间隔是指两个时刻之间的间隔长度。所以，时钟同步主要包括频率同步和相位同步两个方面：

- 频率同步：指各从时钟自身运行的计数器频率保持相同平均速率。通过频率同步可以保持所有节点以相同的计数频率运行，从而极大减小时钟误差根源，降低了累积时钟误差。频率同步必须依靠接受连续的时钟信息才能计算出真实的频率并对自身作出调整。
- 相位同步：指对频率的积分。通过将时钟的相位量化，赋予数值表示，就是时刻。相位同步可以接受离散的时间信息来调整本地时钟。

所以说，时钟同步主要是完成对时和守时这两个功能，所谓“对时”，就是通过不定期的对表操作，来将本地时钟的相位与远程节点相位进行同步；守时就是保持频率一致，尽可能的让本地节点和远程节点的时间间隔偏差在一个允许的范围内。

在工业领域时钟同步技术不断发展，主要使用 GPS、包时钟同步以及精确时钟同步等时钟同步技术。

### 1.1.1 GPS 同步技术

如超高压变电站等工业系统对于时间同步精度的要求非常高，例如变电站运行人员实时监控电网运行，一旦发生事故后的故障分析，这些都需要统一的时间基准。而以全球定位系统 GPS 作为时间基准，可以提供非常高的同步精度（达到 50ns），而且这种方式不依赖通络的数据负载，直接传递频率和相位信息，让整个系统保持极高的同步性能。然而，这种方式也存在自身弊端。

- GPS 不适用于室内节点，存在难以选址和安装的问题，这使得 GPS 同步技术的人力成本增大。
- GPS 设备本身价格昂贵，而且人们一般会选取高昂的晶振用作备用时钟，以保持系统的安全性。而且若 GPS 设备大量使用，可想而知后期的更换升级的费用也会十分高昂。
- 由于美国政府从未对 GPS 信号的质量及使用期限给予任何承若何保证，而且美国政府还具有对特定地区 GPS 信号严重降质处理的能力，所以大量使用 GPS 会存在严峻的国防安全隐患。

### 1.1.2 包时钟同步技术

所谓包时钟同步技术 ToP(Timing over Packet)，就是利用分组网络来传递时间信息，而所传递的时间报文也有多种。

- NTP(Network Time Protocol) 协议全称是网络时间协议，主要用来使互联网上计算机保持时间同步，一般而言，NTP 可以提供 1-50ms 的可靠时间源。另外，SNTP(Simple Network Time Protocol) 简单网络时钟同步协议也是较为主流的时间同步协议，它能提供接近 1ms 的同步精度。然后这两种协议比较容易受到网络突发报文的影响而使得同步精度降低。
- PTP(Precise Time Protocol) 精确时间同步协议，来自于 IEEE1588 协议标准[2]。该协议第一版于 2002 年推出，具备亚微秒级的同步精度，允许同时传递频率和相位信息。第二版于 2008 年公布，缩短并统一了报文长度，并且引入了透明时钟机制。具备更高的时间同步精度。

结合对当前所有同步技术的综合分析可以看出，由于 GPS 存在国防安全隐患和使用成本高昂等因素，难以大规模应用这种同步技术，而 NTP 协议主要应用于互联网中保持计算机时间同步，其精度只能达到毫秒级别，同样不适用于对同步精度要求非常高的工业系统中。而 PTP 同步技术从实现的简单性、精度要求和稳定性方面都能够很好的满足现如今分布式系统对时间同步精度的要求，因此，本文也选取 PTP 为主要研究对象。

### 1.1.3 精确时钟同步技术

应用于网络测量和控制系统的精确时钟同步技术是基于 IEEE1588 标准，也称作 PTP 协议（Precision Clock Synchronization Protocol）。

自 1995 年以太网中数据传输速度从 10Mb/s 提高到 100Mb/s 后，计算机网络领域一致在尝试解决网络设备时钟同步问题，随后开发出了一套同步协议——网络时间协议 NTP(Network Time Protocol) ——来提高网络设备的同步性能。虽然在不断发展中，NTP 协议同步精确度不断提高达到 200$\mu$s，但是，面对同步精度要求越来越高的网络实时通信、测量仪器与工业控制领域，NTP 难以达到它们的要求。

随后，网络精密时钟同步委员会于 2002 年在 IEEE 标准委员会的支持下，推出了 IEEE1588 时钟同步标准。该标准推出后迅速在 Ethernet/IP 等基于以太网的总线中得到采用。该协议采用了主从时钟方案，利用报文传递时间信息，基于网络链路的对称性和延时测量技术，来实现主从时钟的频率、相位精确同步。

在 2008 年，该委员会继续发布了 IEEE1588 第二版。在第二版标准中，实现了亚微秒级的时钟同步，可以用于军事和实际的测试测量中。而且提供了更为灵活的同步周期、改进了时间戳的表示方法、解决了主时钟容错性能差的问题、提高了网络拓扑变化的适应性。另外，该版本中还引入了透明时钟，增加了端延时机制，有助于明显提高同步精度。

PTP 协议主要针对分布式网络系统中的精确时钟同步问题，它能够将系统内部各个独立时钟同步到一个统一的时钟标准上，而且消耗的网络和本地资源非常有限，并带来较高的时钟同步精度。如果硬件时钟能够提供硬件级别的时间戳标记功能，那么其同步精度能够达到亚微秒级别。而且 PTP 协议实现起来成本很低，也非常方便后期的维护工作。PTP 协议与其他时钟同步协议如 NTP/SNTP 相比，具备如下几个特点[4]：

- PTP 协议适用于局域网中支持组播报文发送的网络通信技术，也非常适合在以太网中实现。
- 可以实现亚微秒级别的同步精度，相比之下，NTP/SNTP 最多只能达到毫秒级别。
- PTP 协议所需要的网络资源和计算资源非常少，所以在实现起来消耗的成本不高。

随着 IEEE1588 协议的不断发展与实践，不仅在测试与测量控制、工业自动化领域广泛应用了该协议，而且随后的军事应用、分组通信和电力系统等多个领域也开始逐步将该同步方法应用进去。在国外，2015 年 3 月在 TI 推出的高性能多核架构 KeyStone 中[5]，提供了两个硬件功能来支持 IEEE1588：记录时间戳，发送同步脉冲。其中 KeyStone1 支持 two step 的时间戳模式，同时也能支持 1588 协议中规定的 PTP 报文解析，在国内，2014 年 12 月，OPWILL 发布了基于 PTN 协议分析仪的 IEEE1588v2PTP 和 SyncE 测试选件，为运营商提供关键的网络定时和频率同步测试服务来分析全新的 IEEE1588v2

PTP 和同步以太网（SyncE）协议，从而为运营商的移动回传和 LTE 测试提供超高性价比测试解决方案。

## 1.2 精确时钟同步技术应用现状

根据 IEEE1588 协议内容，系统时钟同步精度应该能够达到亚微秒级[2]。然而，在实际的应用中，即使严格按照协议标准的内容来实现，仍然难以达到这样的同步精度。例如 Padova 大学曾严格按照协议标准，实现了一套完整的 IEEE1588 同步系统[3]，但是在运行中发现同步精度最高只有 100 微秒，和预期的同步精度还有很大的差距，更何况在工业现场中，更加复杂的工业环境使得同步精度更加难以保证。

IEEE1588 协议的同步原理是采用了主从时钟通过时间戳报文来传递时间信息，并通过估计链路延时来计算最终的时间误差并予以校正。然后，通过对 1588 协议的深入分析及对现场应用情况的深入了解，可以发现以下因素可能严重破坏 1588 实际运行的同步精度。

(1) 时间戳精确度

IEEE1588 协议的核心就是通过主从时钟对外发布包含时钟信息的报文来实现所有从时钟向主时钟同步。例如在同步过程中，主时钟会对外发送 sync 报文，该报文中标记的时间戳在理想情况下应该是报文离开主时钟的一瞬间的时间，但是在实际运行中，由于很多时间戳是在物理层之上标记的，报文被标记时间戳后仍会在协议栈中有短暂滞留，这样导致了发送时间戳的不准确，最终破坏了从时钟的同步计算精度[25]。

(2) 链路不对称性

通过分析 1588 协议可以看出，从时钟在计算主从时钟偏差时会假设报文传输的往返路径对称，即 Master To Slave Delay 和 Slave To Master Delay 相等，并据此来求得最终的 offset 值。严格来讲，这种假设并不成立。首先由于网络传输过程要经过中间节点如交换机等设备，报文在进入这些设备的协议栈会发生排队和堵塞的情况，而这些情况所导致的链路延时变动是无法预知的；另外，报文传输的拓扑结构也并非一成不变的，一旦拓扑结构发生变化，就意味着往返路径可能有极大的不同。这些都是实际运行中链路固有的不对称性表现[34]，所以，如果只是简单按照 1588 协议中假设路径对称，那么势必会带来不可忽视的误差。

(3) 网络延时固有抖动

由于 1588 协议中的报文需要在网络环境下进行传输，而在复杂的网络环境下，报文传输的时间一定会带有随机的抖动漂移，也就是说，即使网络负载良好，拓扑结构也不发生变动的情况下，网络延时仍然会有自身的随机抖动。如果对于这些抖动不进行处理，那么就会导致从时钟不断处于波动之中，甚至有可能导致从时钟系统不稳定。

(4) 主从时钟源晶振漂移

首先，IEEE1588 协议的主从时钟偏差的计算方式是假定主从时间偏差值在短时间内是保持不变的。然而真实的情况是，主时钟在发送 Sync 报文时的偏差值 Offset_1 与从时钟最后接收 Delay_Resp 报文时的偏差值 Offset_2 是不完全一致的，所以最终计算出来的 offset 并不完全等于真实的 Offset_2[4]。所以这会导致同步精度变差。

(5) 从时钟校正策略

常规的对从时钟校正策略有直接校正和 PI 控制器校正，其中，直接校正能够快速对从时钟校正，在最短时间内实现时钟同步，然后，一旦 offset 值计算中存在较大偏差，那么从时钟将处于不断的快速波动当中，这会严重破坏从时钟的同步性能，甚至可能导致从时钟无法稳定。PI 控制器是一种较为典型而简单的控制方法，能够保证从时钟较为处于较为稳定的状态，但是，简单的 PI 控制器并不能考虑到延时的波动对从时钟带来的影响，所以也无法达到很好的控制效果。

当前针对上述所问题，工业界对 IEEE1588 协议[2] 进行了深入研究并取得了一定的研究成果。

(1) 时间戳准确度研究现状

为了提高时间戳准确度，尽量使得报文中的时间戳直接来自物理层以可以减少报文在协议栈的停留，2007 年推出了首个可以在以太网收发器上集成 IEEE1588 协议的芯片 － DP83640[4]，该芯片能够将时间戳标记点从原来的应用层下移到数据链路层和物理层之间，从而降低网络流量及协议栈对报文传输延时的影响。又比如，2015 年 3 月 TI 推出的 Keystone 架构也同样能够支持物理时间戳标记[5]。

(2) 链路不对称性研究现状

针对链路延时中的不对称性问题，当前主要的解决方法有通过类似透明时钟来记录报文在中间节点的传输时间，并在从时钟处减去传输时间来计算主从偏差。但这种方式在 2008 年才提出，目前并未得到广泛应用，它不仅会受时间戳精度影响，而且现有大多数同步系统仍然采用的是早期的边界时钟。所以当前并不具备很好的应用价值。在 C. Lei 的文章 [57] 中采用了自适应滤波算法，该算法能较好的进行滤波，但由于不能对历史纪录进行实时分析，从而无法检测持久性的时延变化。

(3) 网络延时固有抖动研究现状

Jayesh Chhapekar 在 2012 年发表的一篇文章中对 IEEE1588 数据包的包延时抖动进行数学分析和建模，并以此来观察数据包延时的概率是如何变化的[8]。但是在真实的工业环境中，由于网络的复杂性和时变性，很难用某个概率分布来拟合固有抖动，所以也很难取得良好的消除抖动效果。文献 [9] 中设计了运用卡尔曼滤波算法来计算主从时钟偏差同样有助于减少固有抖动带来的影响。

(4) 从时钟校正策略研究现状

当前普遍方法是采用 PI 控制器进行校正，如文献 [10] 中对 PI 控制器参数选择进行了评估，但是没有详细研究具体的应用方法。另外由于网络的时变性，无法用固定的参数来控制从时钟，那样会带来很差的自适应性和鲁棒性。除此之外，文献 [11] 对 PID 控制器在时钟同步中的应用进行深入研究，并且采用了经验法来对 PID 参数进行整定，但该文缺陷是没有把系统参数发生变化情况下的参数调整考虑进去。文献 [12] 利用模糊控制算法来估算时钟偏差，这种方法适合于时钟同步周期较大时，但是如果同步周期太大，网络的不确定性干扰加剧，使得模糊控制的效果变差。

## 1.3 本文研究内容

在 IEEE1588 协议的实际应用过程中，通过深入研究分析发现了协议本身存在的多种问题，例如未处理网络延时的不对称性及固有抖动，这使得 IEEE1588 协议在实际应用中无法达到非常理想的同步精度。

### 1.3.1 网络传输时延的随机性和不确定性

在硬件提供高精度时间戳的情况下，网络传输延时误差成为了影响同步精度的最主要来源。因此，本文也着重考察对网络延时问题的解决方法。文章中将网络时延拆分成固有抖动漂移、暂时性时延突变和持久性时延变化，分别分析各种变化原因，并且从数学统计方法的角度出发，对固有抖动漂移阐述基于最小二乘的时延估计方法，把历史时延数据作为样本序列，从而减小时延抖动或突变对同步精度带来的不良影响；对暂时性时延阶跃突变，采用动态阈值法通过限制阶跃噪声的大小来防止从时钟系统出现过大的振荡；对持久性时延变化使用基于固定滑动时间窗实时检测的方法来及早发现持久性变化，并立即通过样本失效手段来快速消除持久性变化对样本序列及时延估计所带来的波动。

### 1.3.2 从时钟校正的控制策略

为了提高从时钟稳定性，一般会采用时钟伺服系统来对从时钟进行控制，一般而言，会使用 PID 控制器来控制从时钟校正量，不过，通过对链路延时的分析，可以发现其中存在一种阶跃性延时。当一个进入稳态的从时钟遇到了阶跃性延时信号时，就会发生一定的抖动并慢慢收敛，这会导致从时钟的频繁调整，从而影响到整个系统的时钟变化[10,11]。因此文中将采用了多模型 PID 控制系统，通过对误差信号进行监控，当信号发生较大跳变时调整控制器以提高从时钟系统稳定性。

### 1.3.3 统计方法在晶振源漂移及透明时钟中的应用

由于 IEEE1588 中默认了一个假设：即认为主从时间偏差值在短时间内是保持不变的。而在实际情况下，由于主从时钟晶振不一致，这会导致主从时钟便差在实时变化中。这使得难以得到最真实的主从偏差值而无法精确校正。所以，文中将会从统计方法角度出发，针对主从时钟源漂移问题采用多种优化方法，以对从时钟晶振漂移进行补偿从二减小主从偏差的波动。另外，由于 IEEE1588v2 中引入了透明时钟的概念，为了能够将本文的统计方法应用到透明时钟中，将做出相关差异分析，并阐述处理方案。

### 1.3.4 算法仿真与验证

在文章后面，本文在深入研究采用的多种时延统计优化算法的前提下，通过在自己搭建的基于 stateflow 的 matlab 仿真平台上，分别测试了上述所提两种算法。根据最终试验结果表现来观察所提算法对时钟同步系统的稳定性、鲁棒性和自适应性等当面的优化。

## 1.4 论文安排

本文的其它章节安排如下，

第二章介绍 1588 同步协议的运行原理，同时在最后介绍 1588 协议在实际运行中遇到的一些不可避免的问题，及对于这些问题的已有研究成果。

第三章从网络传输延时的角度来进行分析，并且利用结合数学统计与实时检测相结合的方法来保证延时数据的稳定性、准确性和可靠性。在该章节会首先对传输延时进行数学模型的建立，通过深入分析其特性可得到数学模型，然后依据模型特性，基于相关的数学统计方法，例如最小二乘处理、固定时间窗等，采用符合模型特性的数学统计优化算法。其中，本文还会应用滑动时间窗延时实时检测方案，用以区分暂时性延时和持久性延时，并在此基础上采用不同的处理方案。从而保证从时钟的稳定性。

第四章从统计角度出发，首先针对时钟伺服系统进行分析，然后为了应对其中的链路延时阶跃突变，文中应用了基于多模型 PID 控制的控制策略，以提高从时钟的稳定性；另外，针对主从时钟源晶振漂移问题，阐述了相关的解决方案，以对从时钟晶振漂移进行频率补偿来减小主从偏差的波动。同时还将对统计算法在透明时钟的应用作出分析，并采用相关的实际应用方案。另外，该章还会利用自己搭建的时钟同步系统仿真平台，并在该平台上对本文中所提到的统计优化算法进行验证。根据仿真结果得出结论，并总结其中的不足之处。

第五章对全文进行总结和展望。

# 第二章 时间序列样本统计方法

在本章中，将依据对 IEEE1588 时钟同步原理及协议中可能存在的一些应用问题进行分析，并梳理关于这些问题的当今研究理论成果来作一个全面的分析。

## 2.1 IEEE1588 时钟同步原理

IEEE1588 系统是由 PTP 设备和非 PTP 设备结合而成的分布式网络化系统。IEEE1588 协议主要介绍的是系统中的实时 PTP 设备之间的相互同步方案，使得所有的从时钟能够与自己的主时钟同步，而所有主时钟又能够和同一个 Grand Master 时钟同步，最终达到整个系统中所有时钟保持同步。

首先，每个 PTP 时钟会有多个端口，当系统启动时，会立即开始 PTP 网络建立过程。该过程中，每个端口会对外发送 Announce 报文，同时通过检查所收到的 Announce 报文中的相关信息来判定哪个端口适合成为主时钟，被判定为主时钟的端口则会定期对外发送 Sync 报文。如果，一个判定为主时钟的端口收到了一个来自于更好的主时钟的 Sync 报文，那么该主时钟则会立即将自身的主时钟状态切换为从时钟状态。当然，如果一个处于从时钟状态的端口认为自己比当前主时钟更加好，那么它可以将自己设置为主时钟状态，并对外发送 Sync 报文。当所有端口通过互相比较时钟特性并建立了各自的主从状态后，则意味着主从秩序完成建立。如果此时新加入了一个时钟，那么该时钟首先会等待来自某一主时钟的 Sync 报文，若在规定的时间内没有收到任何主时钟的 Sync 报文，那么该时钟则将自身设置为主时钟，直到发现一个更好的主时钟。

随后，则会进行时钟同步过程。该过程主要是主从时钟之间发送报文，使得每个从时钟计算出自身与主时钟之间的时钟偏差，并且对自身进行校正从而实现同步。首先主时钟会对外界进行周期性的 Sync 报文发送，对于采用硬件时间戳的设备将直接在 Sync 报文内记录发送时间，否则的话将再发送 Follow_up 报文来进行发送。当从时钟收到 Sync 报文后，则能够接受到发送时间戳并记录当前的接收时间戳。同时，从时钟自身也会周期性向主时钟发送 Delay_Req 报文，会通过接受主时钟回传的 Delay_Resp 报文来共同计算出链路延时 $T_{delay}$ 和主从偏差 $T_{offset}$。最终利用 $T_{offset}$ 对从时钟进行校正而实现同步。在图2–1中，可以看到较为完整的时钟同步过程。下面，对时钟同步过程及相关计算做简明扼要地介绍。

首先，主时钟会周期性对外发送 Sync 报文，当该报文离开主时钟时会记录时间戳为 t_1。

然后，当从时钟接收到 Sync 报文时，标记接收时间为 t_2。此时，从时钟即获得了 Sync 报文的发送与接收时间戳 t_1 和 t_2。那么假设主从时间偏差为 $T_{offset}$，主从时钟间网络传输时间为 $T_{delay}$，则可以得到下面式子：

$$T_{offset} + T_{delay_1} = t_2 - t_1 \tag{2–1}$$

另外，从时钟周期性向主时钟发送 Delay_Req 报文，假设发送时间为 t_3，当主时钟收到该报文时，会立即记录下 Delay_Req 报文的接收时间 t_4，并把该接收时间 t_4 存入 Delay_Resp 报文中传递回该从时钟。那么此时，从时钟会得到 t_3 和 t_4 两个时间，可得到下面式子：

$$T_{delay_2} - T_{offset} = t_4 - t_3 \tag{2–2}$$

此时，根据协议标准中，将往返传输延时视为对称，即：

$$T_{delay_1} = T_{delay_2} = T_{delay} \tag{2–3}$$

所以，结合上面几个式子可以得到如下结果：

$$T_{delay} = \frac{(t_2 - t_1) + (t_4 - t_3)}{2} \tag{2–4}$$

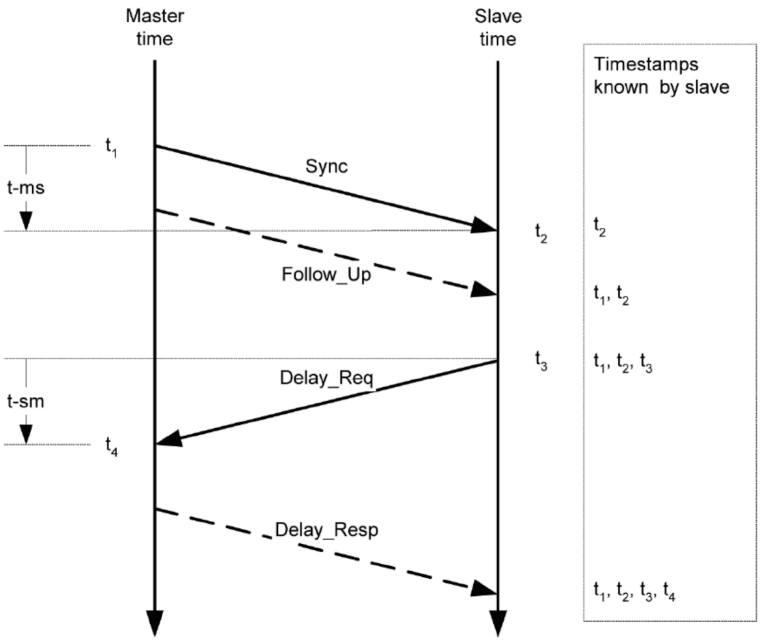


图 2–1 时钟同步算法流程图

Fig 2–1 Method of Time Sync

$$T_{offset} = \frac{(t_2 - t_1) - (t_4 - t_3)}{2} \tag{2–5}$$

通过上面的计算方法，可以得到主从时钟之间的相位偏差 $T_{offset}$。然后，还需要计算主从时钟的时钟频率偏差才能实现完整的时钟同步。

为了计算时钟频率偏差，认为主时钟是按固定周期向从时钟发送 Sync 报文，那么，对主时钟而言，第 (k-1) 个 Sync 报文和第 (k) 个 Sync 报文之间的发送间隔[6] 为:

$$\Delta T_{master} = T_{master(k)} - T_{master(k-1)} \tag{2–6}$$

其中，$T_{master(k)}$ 表示第 k 个 Sync 报文的主时钟发送时间。

对于从时钟而言，两个 Sync 报文的时间间隔[6] 为:

$$\Delta T_{slave} = T_{slave(k)} - T_{slave(k-1)} \tag{2–7}$$

其中，$T_{slave(k)}$ 表示第 k 个 Sync 报文的在从时钟处的接收时间。由此得到主从时钟的频率偏差。

$$\Delta f = \frac{\Delta T_{master}}{\Delta T_{slave}} \tag{2–8}$$

因此，只需要调节从时钟本地频率扩大 $\Delta f$ 倍就可以实现主从时钟频率一致。

## 2.2 同步精度的影响因素

### 2.2.1 PTP 报文传输周期

在同步过程中，最基本的是 $T_{offset}$ 值和 $T_{delay}$ 值的计算，直接影响同步结果的准确性和稳定性，以及最初 PTP 网络建立的快速性。所以，协议相关的时间周期和超时事件如下。

- Announce 报文周期：该周期为 $2^{AnnounceInterval}$，协议默认 Announce 报文默认发送间隔为 2s，采用多播方式。当从时钟接收到该报文，若端口为 Slave 状态，则会更新本地数据集。
- Sync 报文周期：SyncInterval 参考值为 0.5s-2s。每个 Master 端口会周期性对外发送 Sync 报文，以触发系统中同步过程。如果 SyncInterval 取值过大，则会使的从时钟偏离太大而失去同步效果；若该周期太小，则会严重加剧网络流量负载，从时钟处于不断的调整波动之中。
- Delay_Req 报文周期：MinDelayReqInterval 参考值为 1s-32s，默认 1s。该报文主要用来更新链路延时 $T_{delay}$ 值。如果周期过小，则会给主时钟增加过多的负载压力，

加剧网络流量负载；若周期过大，则有可能导致上一次的 $T_{delay}$ 与当前实际 $T_{delay}$ 值出现严重偏差，从而导致 $T_{offset}$ 值偏差较大，严重破坏同步精度。理想的周期应该在区间 [0，$2^{MinDelayReqInterval+1}$] 中均匀分布。

### 2.2.2 链路延时不对称

如 (2-4)、(2-5) 所示，当计算主从时钟偏差 $T_{offset}$ 时，需要先获取链路传输延时 $T_{delay}$。在协议中该延时的计算方法是直接假设前后两次传输延时相等，从而通过取平均计算出 $T_{offset}$。然而实际上，前后两次的传输路径往往并不对称，即式 (2-3) 并不成立，而导致往返的链路传输延时不一致的因素[55] 如下。

- 排队堵塞：当报文在传输中经过中间交换机或路由器时，由于网络负载的不可预知性，无法知道报文在中间交换机上的排队时间，当网络流量良好时，报文可以直接传递过去；而当网络负载较大，可能导致很长的报文排队时间。另外，即使报文可以不用排队，由于协议栈的存在，报文仍然需要经过解包和打包的过程，而这两个过程的消耗时间都与操作系统的调度和协议栈处理过程有关。因此，报文传递过程中穿越交换机时排队和堵塞时间的随机性会导致延时不对称[46]。
- 传输抖动：在网络系统中，所有信息的传递过程所消耗的时间会存在固有的抖动，这是无法避免的。
- 网络拓扑结构变化：当拓扑结构突然发生变化，报文传输的路径也就不同了，这必然导致往返链路时延完全不同。
- $T_{delay}$ 与当前真实延时不匹配：因为 $T_{delay}$ 的计算依靠 Delay_Req 报文周期，而 $T_{offset}$ 的计算又是依靠 Sync 报文周期，两者一般并不匹配。所以会导致在计算 $T_{offset}$ 所使用的 $T_{delay}$ 是过去的值，与当前真实的 $T_{delay}$ 值可能并不一致。

上述几种现象的存在，都会导致往返的两次延时不一致，所以说，在高同步精度的要求下，真实的传输延时绝对不能简单的假设相等。

### 2.2.3 主从时钟源频率漂移

由于主时钟在发送 Sync 报文时的偏差值 $T_{offset1}$ 与从时钟最后接收 Delay_Resp 报文时的偏差值 $T_{offset2}$ 是不完全一致的，所以最终计算出来的 $T_{offset}$ 并不完全等于真实的 $T_{offset2}$。而且，在实际工业环境中，由于主时钟一般会采用更为稳定、精度较高的设备，而从时钟则一般精度较低，晶振源容易发生漂移现象，从而导致主从时钟之间的频率无法保持一直，这个因素会持续不断的使得从时钟偏离主时钟，因此，本文将对从时钟频率进行补偿，从而降低从时钟漂移现象对同步精度带来的破坏。

### 2.2.4 时间戳的精确度

对于很多采用软件时间戳方式的设备，报文在协议栈中传递所带来的延时会导致物理层之上的时间戳往往不能真实反映报文的发送时间或接收时间，从而导致时间戳[53]。这也导致报文封装和解包过程产生偏差，直接破坏最终的时钟同步精度。本交换机项目中，采用的是支持硬件时间戳的设备，即设备能够直接在物理层为报文添加时间戳，从而保证了时间戳的精确度，因此，在本文将会忽略此因素。

## 2.3 本章小结

本章首先介绍了 IEEE1588 时钟调频调相的同步原理，并且对其中与本文相关的时钟同步过程进行了分析，另外，还分析了 ptp 报文传输过程消耗的时间对同步精度的影响，表明了封装中发生的排队堵塞现象和传输中的链路变化等均会严重破坏最终的时钟同步精度。

# 第三章　基于统计的链路延时误差优化方法

本章在分析链路延时实际影响因素的基础上，针对链路延时建立完整细致的数学模型，并采用统计的方法，利用从时钟的时延历史样本来研究真实链路延时的估计方法。

## 3.1　链路延时不对称性的影响因素

### 3.1.1　网络传输抖动

在网络系统中，所有信息的传递过程所消耗的时间会存在固有的抖动，这是无法避免的，也是引起链路延时不对称性的主要原因。通常认为网络流量随机过程符合马尔可夫模型或者泊松分布。其中，马尔可夫模型表示对历史具有有限的记忆，或者说在已经知道现在的前提下，其将来并不依赖于过去；而泊松过程中的随机变量（单位时间呼叫达到的次数）是独立且服从自相似分布的[12]：

$$P(X_k = n) = \frac{n\lambda\Delta t e^{-\lambda\Delta t}}{n!}, n \geq 0; \tag{3–1}$$

然而通过试验测量分析认为，网络流量序列在很宽的时间尺度内存在突发现象，或者说，Internet 网络流量序列符合自相似特性[13–16]，即样本均值的方差比样本个数的倒数减小得更慢。另外，文献 [17] 对于 NTP 时间包全程延时作了大量研究，文献 [18] 利用两主机之间通过传送带时间标签的 UDP 包测量全程延时，得到了相同的结论，即认为 Internet 网络流量及延时具有一定随机抖动。而以太网是构成 Internet 网的主要部分，所以有理由认为以太网网络流量及延时也具备随机抖动。

### 3.1.2　网络拓扑结构变化

一般的拓扑结构如图3–1所示[2]：

假如同步系统中由于环路结构的快速重配置等原因导致链路的拓扑结构发生改变，将导致报文传输的往返延时的不同，而这也导致计算出来的延时值与实际值有很大偏差。然而，在现有的 IEEE1588 协议中，并没有很好的去探测链路结构的变化并针对性做出处理。这也正是下文中所针对的一个重点研究内容。

### 3.1.3 delay 与当前真实延时不匹配

因为 delay 的计算依靠 Delay_Req 报文周期，而 offset 的计算又是依靠 Sync 报文周期，两者一般并不一致。所以会导致在计算 offset 所使用的 delay 是过去的值，与当前真实的 delay 值可能并不一致。所以，这会导致多数时候计算所使用的 delay 值并不一定等于真实的 delay 值，从而出现误差。

### 3.1.4 传输链路堵塞

在真实的工业网络环境中，网络负载总是在实时变化的。一旦交换机内部负载过大，队列缓冲区堆积了较多的待转发报文时，新进入的同步报文则会保持在队列末尾进行等待，直到交换机把前面的所有报文都转发处理了才轮到该同步报文转发。而在这个等待过程中所消耗的时间完全取决于网络负载情况和交换机内部队列的密度，这种时间对于从时钟而言是完全无法预知的。所以，很有可能前后两次的往返报文在穿越中间

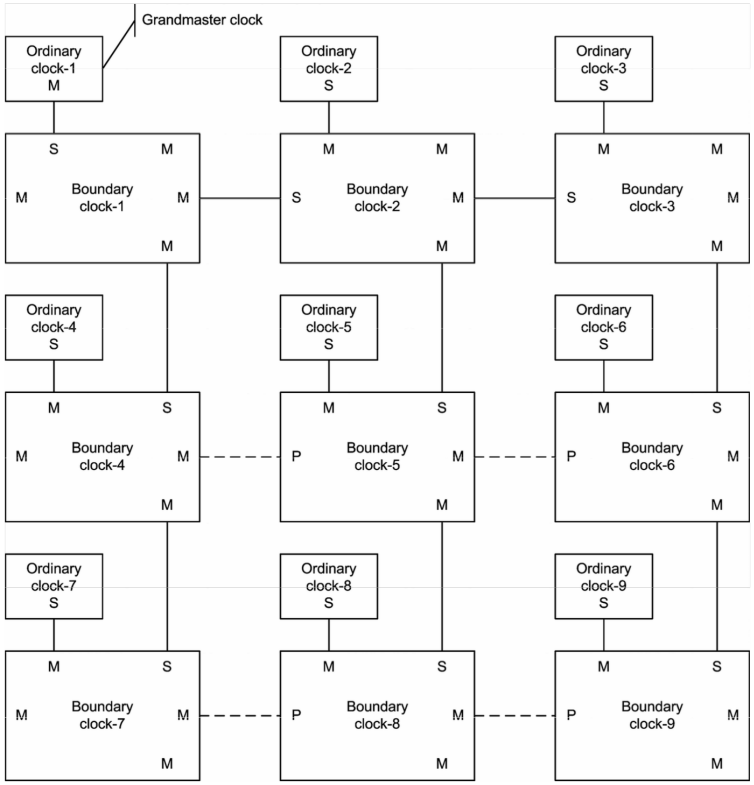


图 3–1 分布式同步系统结构图

Fig 3–1 The Structure of distributed sync system

某个交换设备时，所消耗的等待时间会有很大差异，而正是这种差异的存在，即报文堵塞时间的不可预知性和随机性，导致了链路延时的不对称性。当报文进入中间交换设备时，会先转发操作。然而，该转发操作并非瞬间完成的。当报文进入交换设备，会依赖于该交换设备的协议栈并得到解包和打包处理。所谓解包即交换设备根据自身协议栈把报文层层解析得到用户数据，打包即重新把用户数据封装并向协议栈底层传递。这个过程称之为堵塞时间，即交换设备空闲，但由于协议栈的存在导致报文必须驻留的解析时间。所以，在报文得到处理到真正离开中间设备，中间必须经过一段堵塞时间。而堵塞过程中的解包和打包操作，均需要操作系统分配 CPU 来进行处理。而操作系统的调度是完全随机的，无法提前预知操作系统的调度策略，也无法预知该解析过程所真实需要耗费的时间。尽管这些时间非常微小，但对于想要达到亚微秒同步精度的同步系统而言，这样的时间偏差不可忽视。

## 3.2 链路延时数学建模

如前文所述，报文的传输过程所受到的影响因素主要有以下几点：延时固有抖动会导致延时发生固有的小幅波动；拓扑结构变化或网络重配置将使得链路传输延时与之前的延时会发生明显变化，即新延时不再与旧有延时有任何关系；链路传输堵塞的发生取决于网络状况，网络良好时，该延时时间非常短，但当网络负载加剧，则延时会突然剧烈增大为 T1，直到网络恢复，因此可以认为这种情况导致的延时增大往往是暂时性的。基于此，可以用下列数学模型[33] 来进行描述链路传输延时：

$$T_{all}(i) = T_{pure}(i) + T_{queue}(i) + T_{jitter}(i) i = 1\ 2\ 3... \tag{3–2}$$

其中，我们使用 i 表示同步过程中的第 i 次报文传输，$T_{all}$ 代表单次链路传输总延时，$T_{pure}$ 代表在不考虑任何堵塞和抖动的情况下，在固定传输路径上的延时，$T_{queue}$ 代表在该次链路传输过程中，发生的堵塞等导致的总延时，$T_{jitter}$ 表示该次传输中存在的抖动延时。为保持简洁，用下面公式代替：

$$T(i) = T_p(i) + T_q(i) + T_j(i) \tag{3–3}$$

### 3.2.1 固定拓扑结构下的纯粹链路延时

对于 $T_p$，用来表示在固定的拓扑结构下，不考虑堵塞和抖动等外界干扰的纯粹链路延时。容易知道，只要链路保持不变，那么 $T_p$ 就不会发生改变，也就是说，假设网络配置情况等保持一定的概率为 $P_{path\_stable}$，那么 $T_p$ 就为常量，即

$$T_p(i+1) = T_p(i) \qquad probability : P_{path\_stable} \tag{3–4}$$

$$T_p(i+1) \neq T_p(i) \qquad probability : (1 - P_{path\_stable}) \tag{3–5}$$

正常来说，网络重配置的概率非常小，也就是说 $P_{path\_stable}$ 值非常小。但是，如果一旦发生了网络重配置，那么未来的延时与过去的延时将很有可能出现很大变化。

基于上述特性，在此将拓扑结构变化延时定义为持久性时延变化。

### 3.2.2 链路传输堵塞

对于 $T_q(i)$，用来表示报文在传输过程中经过中间转换设备时的驻留时间。正常情况下，即当网络中流量负载良好时，$T_q(i)$ 会保持一个微小的驻留值，接近为零。但是，一旦网络负载突然加剧，那么 $T_q(i)$ 会瞬间增大，而且网络负载越大，则 $T_q(i)$ 会有越明显的突变。这直接导致的后果的前后几次测量的 delay 值有明显变化，从而影响到从时钟的精度。所以，假设网络流量繁忙的概率为 $P_{net\_busy}$，那么可以用如下公式来描述堵塞延时：

$$T_p(i) = S(i) \cdot W(i) \tag{3–6}$$

$$S(i) = 1 \qquad probability : P_{net\_busy} \tag{3–7}$$

其中，$S(i)$ 是一个开关量，等于 1 的概率为 $P_{net\_busy}$，或者说，当网络繁忙时，$S(i)$ 便为 1；反之，当网络空闲时，$S(i)$ 便为 0，即堵塞延时 $T_q(i)$ 也为 0。$W(i)$ 则表示真实的堵塞延时。不过一般而言，网络繁忙持续时间较短，所以在多数时间 $T_q(i)$ 取值接近零，偶尔会发生阶跃性突变。

基于上述特性，在此将报文堵塞延时定义为暂时性延时突变。

### 3.2.3 链路传输延时固有抖动

对于 $T_j(i)$，用来表示传输延时的固有抖动成分。该抖动主要基于 $T_p$，度量真实链路传输时间对平均时间的偏差值。从宽时间序列角度来看，对于随机变量 $T_j(i)$ 的期望值有如下公式[33]：

$$ET_j(i) = 0 \tag{3–8}$$

这里，假设 $T_j(i)$ 的方差值为 $\sigma^2$。$T_j(i)$ 都是独立同分布。

### 3.2.4　链路传输延时模型分析

上文中已经从数学模型角度，建立了多个方程来描述链路传输延时 T。由于本文的根本出发点是利用历史延时数据作为样本来共同估计当前真实延时，即从时钟端会通过收集并保存历史时延样本，然后利用这些历史时延样本数据，采用结合最小二乘法、动态阈值法和滑动固定时间窗实时检测三种方法来共同消除上述因素对延时造成的误差。

下面从历史延时数据样本角度切入，来对上文中所建立传输延时数学模型进行分析并采用相应的解决方法。

#### 3.2.4.1　链路传输固有延时 $T_j(i)$

由于该固有延时是以太网流量中存在的随机因素，且可以认为 $T_j(i)$ 是小幅波动的干扰噪声。这种干扰噪声会导致最终的传输延时处于小幅随机波动中。一般而言，针对此类随机变量最好的处理方法就是依靠历史样本进行滤波，因此，在下文中将采用基于最小二乘法的滤波法，通过对历史延时样本进行线性回归，得到一条直线并以此来估计当前时延，从而滤除此类随机干扰噪声。

#### 3.2.4.2　链路传输堵塞 $T_q(i)$

若实际运行中突发网络负载加剧，则 $T_q(i)$ 会由零突变增大，且取决于网络负载压力。不过，网络负载处于不断变化中，而且多数时候是网络畅通的。由于 $T_q(i)$ 在阶跃增大后，会随着网络负载减小而恢复到旧有延时，因此，将这种延时变化定义为暂时性变化。也就是说，即使发生了网络负载加剧，旧有延时样本仍然有效，只是由于网络负载的出现导致传输延时出现了小幅阶跃信号。因此，在下文中采用了基于动态阈值法，用来消除历史延时样本中突变数据对时延估计结果的影响。

#### 3.2.4.3　纯粹链路延时 $T_p(i)$

在实际运行中，当网络重配置导致拓扑结构变化时，会使的 $T_p$ 与之前的延时不一致，而且一般来讲不会恢复到之前的延时。这也就是说，拓扑结构变化直接导致了链路传输延时的持久性变化。由于采用的数学统计方法中需要依赖旧有延时来估计当前真实延时，所以，持久性变化的发生会导致旧有延时样本彻底失效。但如果在实际运行中，系统不能及时检测出持久性变化的发生，那么依旧采用已失效的旧有延时来进行同步计算，一定会导致误差大大增大。因此，在下文中采用了滑动固定时间窗实时检测法，用来实时检测持久性变化，以实现尽快检测并迅速使旧有样本失效。

## 3.3 固有时延抖动

根据上文知道，固有抖动 $T_j(i)$ 属于随机变量，而且可以认为在宽时间序列上，固有抖动的期望值 $E(T_j(i))$ 为零。所以，针对这种特性的随机干扰噪声，采取任何一种统计方法都能够得到比较好的去噪效果。文章在此处选用了最小二乘法，通过对多个历史延本数据进行线性回归，可以得到一条良好的直线，而可以利用这条直线来对当前延时进行估计。

之所以采用最小二乘法，还有一个重要的原因是可以得到某时间段内时延回归直线的斜率，在下文的滑动固定时间窗实时检测法中，将利用该直线斜率的变化来进行对持久性时延变化的实时检测。

在从时钟处，会不断收集离散的时延样本数据 ($t_i$, $D_i$)(i=1, 2, 3 $\cdots$)，其中，$D_i$ 表示在从时钟 $t_i$ 时刻获得的时延值。为了进行最小二乘线性回归，取离当前最近的 m 个样本值来进行处理，并取基 1，t，进行线性回归拟合

$$f(t) = a_0 + a_1 t \tag{3–9}$$

使得方差值

$$E(a_0, a_1) = \sum_{i=0}^{m} (f(t_i) - D_i)^2 = \sum_{i=0}^{m} (a_0 + a_1 t_i - D_i)^2 \tag{3–10}$$

能够达到极小值，因此，$a_0$ 和 $a_1$ 需要满足

$$\frac{\partial E}{\partial a_k} = 2 \sum_{i=0}^{m} t_i^k (a_0 + a_1 t_i - D_i) = 0 \qquad (k = 0, 1) \tag{3–11}$$

上面的式子可以简化为如下形式:

$$a_0 \sum_{i=0}^{m} t_i^k + a_1 \sum_{i=0}^{m} t_i^{k+1} = \sum_{i=0}^{m} t_i^k D_i \qquad (k = 0, 1) \tag{3–12}$$

将该方程组转化为矩阵表示，可以得到如下的形式:

$$\begin{bmatrix} m+1 & \sum_{i=0}^{m} t_i \\ \sum_{i=0}^{m} t_i & \sum_{i=0}^{m} t_i^2 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \end{bmatrix} = \begin{bmatrix} \sum_{i=0}^{m} D_i \\ \sum_{i=0}^{m} t_i D_i \end{bmatrix} \tag{3–13}$$

将之前在从时钟所收集到的距当前最近的 m 个时延样本数据代入上述式子中，即利用方差最小原理，可以得到 $a_0$ 和 $a_1$ 两个参数，因此便得到了一条拟合直线: f(t) = $a_0$ + $a_1$。此时就可以用该直线来估计当前时刻的真实时延时，而且，由于固有抖动噪声的随机性，通过该方法会显著消除时延抖动所造成的影响。

针对链路延时固有抖动，由上文数学模型可以知道该抖动时属于随机变量，所以选择的思路是采用数学统计方法，把历史时延样本作为基础，通过对这些时延样本建立线性回归模型并得到一条拟合直线，通过该直线可以讲随机扰动噪声造成的影响大大减下。

但是，对于同步系统中发生的暂时性突变，例如由堵塞等导致的暂时性的时延突变，若简单最小二乘法，那么时延的突变会导致回归直线的斜率快速增大，即导致计算出来的主从偏差也快速变化。当这种暂时性突变恢复正常，回归直线也逐渐恢复正常。这样容易导致从时钟经常处于快速振荡状态，非常不利于从时钟系统的正常工作。

所以，在下面将针对这种暂时性时延突变所带来危害进行限制，防止其突变造成主从偏差跟随性突变从而严重破坏从时钟鲁棒性。

## 3.4 暂时性时延变化的动态阈值法

由上文描述可以看出，单纯的最小二乘法确实可以有效的消除链路传输延时中的固有抖动部分，然而，一旦同步系统中发生由堵塞等导致的暂时性突变时，那么由最小二乘法回归得到的拟合直线也会随之有明显的突变，而这将导致相应的从时钟进行较大幅度的错误校正，而且，由于这种突变一般是暂时性的，所以，等网络负载恢复正常，拟合直线也渐渐恢复正常。然而，如果系统中频繁发生此类暂时性变化，那么势必会导致从时钟快速振荡。

因此，下文中针对暂时性时延变化作出深入研究，并且采用动态阈值法来应对此类暂时性变化的发生。

### 3.4.1 暂时性时延变化特征研究

对于堵塞等因素导致的暂时性时延变化 $T_q(i)$，通过上文的数学模型分析可知，该时延变化因素是偶然性的，而且一般持续时间比较短，不过一旦出现，往往会导致时延出现较大的波动。可以将此类误差定义为暂时性阶跃噪声，一方面表示其持续时间较短，另一方面表示堵塞现象一旦发生，很有可能导致延时出现较大波动。

对从时钟而言，是采用上述最小二乘法来对历史时延进行线性回归估计，所以，如果出现了此类暂时性阶跃噪声，则会使的若干样本值相对旧有样本值之间，出现较大涨幅。而这种突变的涨幅会导致回归直线的斜率逐渐上升，则直接导致最终计算出来的延时值明显增大，从而导致 offset 值出现暂时性的大幅波动，虽然说后续随着网络负载的逐渐减轻而使的时延值及回归直线斜率逐渐恢复，但是，从从时钟稳定性角度出发，如果网络负载变化比较频繁，那么从时钟则会不断处于大幅波动之中，这对于从时钟系统稳定性极为不利。

### 3.4.2 动态阈值法分析

为了防止暂时性时延突变造成从时钟进入大幅波动甚至不稳定等现象，需要对时延变化进行一定的限制，或者说，通过历史时延的平均波动来衡量当前时延值带来的波动，如果当前时延值波动处于可接受阈值范围内则默认有效，若超过该阈值，则应该将其限制在阈值范围内。从而保持从时钟不会因为暂时性时延突变而出现快速大幅波动等不稳定现象。

对于阈值的选取，不应该将其设置为一个固定值，因为在网络环境不断变化的工业现场中人们无法提前预知堵塞等因素会导致的时延波动。所以，如果想要合理的判断当前时延的真实波动特性，最好的方法便是基于历史时延波动情况。如果当前时延波动与历史波动相比，处于一个有限范围内，那么可以认为当前时延带来的波动是正常的；反之，如果当前波动显著超过历史波动，那么应该对当前波动进行一定的限制，从而防止其对从时钟稳定性造成危害。

### 3.4.3 动态阈值法实现

根据上述分析可知，动态阈值的选取需要实时根据历史时延波动数据而得。所以，为了能够计算历史波动情况，将首先创建一个固定时间窗，将每次的测量时延值保存其中；然后，通过时间窗内的历史时延值来计算时延波动阈值；最后，利用该阈值来衡量当前时延的抖动情况，并依据衡量结果来得到估计时延值。

下面给出一个含有测量时延和估计时延的固定时间窗，见表4–3：

表 3–1 动态阈值法第 k 个固定时间窗

Table 3–1 Mixed Time Window of Dynamic Threshold Method

| 时间 | $t_{k1}$ | $t_{k2}$ | $\cdots$ | $t_{k(m-1)}$ | $t_{km}$ |
|---|---|---|---|---|---|
| 当前测量时延 | $d_{k1}$ | $d_{k2}$ | $\cdots$ | $d_{k(m-1)}$ | $d_{km}$ |
| 动态阈值法时延估计值 | $\overline{D_{k1}}$ | $\overline{D_{k2}}$ | $\cdots$ | $\overline{D_{k(m-1)}}$ | $\overline{D_{km}}$ |

其中，当前为时间 $t_{ki}$，此时测量时延为 $D_{ki}$，为了了解该时延的波动性，需要先计算历史时延的波动标准差，此时，假设之前的波动标准差为 $\sigma_i$，该标准差可以直接对从 $t_{k1}$ 时刻到 $t_{k(i-1)}$ 时刻之间的估计时延，依据标准差公式计算而得。此处不作赘述。然后，可以依据该标准差 $\sigma_i$ 来选取动态阈值，依据下式来进行选取：

$$T_i = \alpha * \sigma_i \tag{3–14}$$

其中，$T_i$ 表示在 $t_{ki}$ 根据历史时延数据实时计算出来的当前动态阈值，可以认为通过设定参数 $\alpha$ 来调节阈值的范围，通常可取值为 1 - 3。

然后，还需要引入截止函数来判断当前波动是否超出动态阈值的范围，并根据判断结果做出相应处理方法。将采用如下截止函数，当波动超过动态阈值时则直接取该阈值来计算当前实验估计值。

$$G(d_i) = \begin{cases} d_i, & |d_i| < \alpha * \sigma_i \\ \alpha * \sigma_i, & d_i \geq \alpha * \sigma_i \\ -\alpha * \sigma_i, & d_i \leq -\alpha * \sigma_i \end{cases} \qquad \alpha > 0 \tag{3–15}$$

基于上面的动态阈值与截止函数，在 $t_{ki}$ 时刻，可以利用第 k 个固定时间窗历史数据及当前测量延时 $d_i$ 得到如下当前时延估计值:

$$\overline{D_{ki}} = \gamma * G(d_i) + \overline{D_{k(i-1)}}; \qquad 1 \leq i \leq m \tag{3–16}$$

其中，$\gamma$ 用来表示当前测量时延在当前估计时延 $\overline{D_{ki}}$ 中所占的比重，一般可以取:

$$\gamma \approx (0.85 - 0.9) \tag{3–17}$$

最后，每一轮时间窗的第一个延时估计值 $\overline{D_{k1}}$ 是取上一轮时间窗所有时延估计值的平均值，即:

$$\overline{D_{k1}} = \frac{1}{m} * \sum_{i=1}^{m} \overline{D_{(k-1)(i)}} \tag{3–18}$$

### 3.4.4　动态阈值法优缺点分析

利用上述动态阈值法，可以通过在固定时间窗内，通过计算历史时延值来得到一个动态阈值，然后利用该动态阈值来衡量当前测量时延的波动影响。若链路中发生较为严重的堵塞延时，导致时延值突然大幅增大，那么可以通过该动态阈值方法可以有效防止时延值大幅突变对从时钟造成的快速振荡甚至不稳定，从而使得从时钟有更佳优良的稳定性。

但是，此方法的前提是该时延突变必须为暂时性突变，也就是说这种突变只是短时间内的，随着系统继续运行该突变又会逐渐减小并恢复正常。只有在这样的前提下，才可以通过动态阈值限定方法，在这个突变的短时间内把时延突变对从时钟造成的影响尽量减小。

然而，如果同步系统中发生的是持久性突变，例如拓扑结构变化，那么就意味着真实时延值与历史时延值已经完全不同，如果此时仍然采用上述动态阈值法强制把时延

限制在一个有限范围内，那么很有可能导致数据完全偏离真实时延，出现难以想象的错误。

基于此，下面继续针对同步系统中可能发生的持久性时延变化采用一种实时检测法，该方法能够快速发现持久性突变的发生，并且立即通过使历史样本失效的方法来刷新时间窗并重新进行时延估计。

## 3.5 持久性延时变化的滑动时间窗实时检测法

### 3.5.1 持久性延时变化特征研究

由上文可以看出，基于最小二乘法和动态阈值法可以有效的消除固定抖动噪声和暂时性阶跃噪声，改善了同步系统的精确性和鲁棒性。但是，如果发生由网络重配置等因素导致的拓扑结构变化时，就会导致链路传输时延的持久性变化，如果仅仅采用上述两种方法是无法有效应对此类变化的。

根本原因在于，一旦发生持久性变化，则意味着历史延时样本均已经失效，不应该继续用于后续的时延估计中。然而，上述两种方法均依赖于历史延时样本，而且这两种方法均无法探测出持久性变化的发生。这样带来的直接后果就是：在时延已经完全变化后，同步系统仍然继续使用错误的历史延时样本数据去估计当前真实的延时，所计算出来的估计时延值一定会与真实时延存在很大偏差，而且这样的偏差要一直持续到右足够多的新样本替代旧有时延样本后才能逐渐消除。

显然，这会严重破坏系统的同步精度和稳定性。因此，文中应用了一种基于固定滑动时间窗的实时检测法。下面对该方法进行详细的介绍。

### 3.5.2 滑动时间窗实时检测法

这里所谓的持久性变化主要是指一旦发生了网络拓扑结构等变化，那么新的链路传输延时将与旧有延时样本完全不一致，这也就意味着，旧有累积延时样本已经不能作为计算新延时的基础数据了。否则，大量失效的延时样本数据会导致当前依靠最小二乘法所得到的时延估计出现非常严重的偏差。

所以，下面会上述最小二乘法所得到拟合直线斜率角度入手，来实时检测持久性变化的发生。

当从时钟收到同步报文时，会立即依靠历史数据和最小二乘法来建立一条回归直线。那么考虑这样一种情况，如果网络发生了重配置导致链路传输延时，那么此刻之后的新的延时数据就会与之前的样本数据有明显偏差，而此时仍然在使用最小二乘线性回归方法计算回归直线。那么可想而知，此刻之后的回归直线斜率会发生明显突变，假设

网络重配置导致链路变得更长，那么得到的延时也就比之前更大，对应的回归直线斜率也会不断的向上提升，一直到历史实验样本全部被新的时延数据替代，那时的新的回归直线会恢复到接近水平。而在这个过程中，由于直线斜率的先增后减，导致其估计出来的时延值非常不准确，而且会导致从时钟发生一次大幅波动。

所以，理想的状态应该是，当发生拓扑结构突变，链路延时回归直线开始提升时，就能够尽快检测并发现这种现象，然后立即把历史时延数据失效化。从而，新的时延数据不会被历史样本影响而导致从时钟要经过长时间的波动直到新的时延完全替代旧有时延样本。于是，通过立即把历史时延数据失效的方法可以让从时钟尽快得到校正。

### 3.5.3 滑动时间窗实时检测法实现

通过上述介绍，下面将目标着眼于回归直线斜率的变化情况，并通过对历史斜率的实时检测从而快速发现持久性变化的发生，并且采取使历史样本数据立即失效的方法来快速校正从时钟。

为了能够实时检测回归直线的历史斜率变化情况，下面会采用滑动时间窗方法来实时检测并计算斜率变化。

表 3–2 在当前 $t_c$ 时刻的滑动时间窗

Table 3–2 Slide Time Window at current time $t_c$ in Real-Time Monitor Method

| 时间 | $t_{c1}$ | $t_{c2}$ | $\cdots$ | $t_{c(m-1)}$ | $t_{cm}$ |
|---|---|---|---|---|---|
| 当前测量时延 | $d_{c1}$ | $d_{c2}$ | $\cdots$ | $d_{(cm-1)}$ | $d_{cm}$ |
| 当前回归直线斜率 | $\overline{K_{c1}}$ | $\overline{K_{c2}}$ | $\cdots$ | $\overline{K_{c(m-1)}}$ | $\overline{K_{cm}}$ |

在表3–2中，该时间窗最后一列数据为当前值。对应当前时刻为 $t_c$，此时测量得到的时延值为 $d_c$，同时可以得到当前回归直线的斜率值 $\overline{K_c}$。由于如果当前同步系统并非发生持久性变化，那么该回归直线的斜率值应该是接近零并处于上下小幅抖动之中；而如果同步系统发生拓扑结构等持久性时延变化，那么斜率值则会处于一个不断上升或者不断下降的趋势之中。其中，不断上升意味着新的拓扑结构带来了更长的链路传输延时，反之，不断下降则意味着新拓扑结构带来了更短的链路传输延时。

那么，为了检测回归直线斜率的变化，下面从该滑动时间窗内的所有斜率样本值角度出发，通过探究这些斜率样本值的波动方差来决定是否发生持久性变化。即利用下面

的式子来计算时间窗内的斜率均值 $E_c$ 和方差 $D_c$:

$$E_c = \frac{1}{m}\sum_{i=1}^{m}(\overline{K_{ci}}) \tag{3–19}$$

$$D_c = \frac{1}{m}\sum_{i=1}^{m}(\overline{K_{ci}} - E_c)^2 \tag{3–20}$$

所以，在 $t_c$ 时刻可以计算出时间窗内的历史斜率波动方差为 $D_c$，根据上文分析，若该方差接近零，则表示保持小幅振荡，此时可以认为没有发生持久性变化；但如果斜率方差值超过一定范围，那么可以认为同步系统中发生了持久性变化，为了使系统快速适应这种变化，则需要立即使历史样本失效。因此，采用下面函数来处理这种过程:

$$R_c = \begin{cases} 0, & D_c < \omega * \overline{DT} \\ 1, & D_c \geq \omega * \overline{DT} \end{cases} \tag{3–21}$$

$$\omega \approx (1\ 1.5) \tag{3–22}$$

其中，$\omega$ 为可调节参数，也可以称为容忍度，即对于回归直线斜率变化的容忍程度，正常情况下，容忍度越小，则能够越快发现持久性变化。$\overline{DT}$ 为平均方差。若当前时间窗内所计算出来的波动超过了限定阈值，那么就认为发生了持久性变化，进行失效处理，即把所有旧有样本全部删除，并重新开始同步。从而保证从时钟对持久性变化的适应性和快速响应性。

除此之外，对于滑动时间窗的实现，首先在上文中假设该时间窗长度为 m，然后以最新测量时刻 $t_c$ 为该滑动时间窗的末尾一组数据，向前获取前 (m-1) 组数据加入该滑动时间窗内部。所以，每次新收到一次同步报文，便把该报文的数据加入到该滑动时间窗的最后一组，与此同时，将第一组数据从该时间窗内移除。所以整体看来，该时间窗是处于不断前行的一个过程，因此成为滑动时间窗。采用这样的方法可以时刻保持当前历史时延数据最新，才能够最快的检测出最真实的回归直线斜率的变动情况。

### 3.5.4 滑动时间窗实时检测法小结

首先，该方法主要是用来应对同步系统链路中发生的网络重配置等导致的拓扑结构变化。由于文章是从统计角度入手，所以每次同步计算都要严重依靠历史样本数据。而如果一旦发生了上述的持久性变化，那么就意味着历史样本数据已经完全失去效果了，如果继续使用则会导致严重误差且长时间无法正确校正。所以，本方法通过对时延的变化情况着手，通过实时检测最小二乘法中拟合直线的斜率变化，来快速判断同步系统中是否发生了持久性变化，一旦发现，便立即将历史样本失效，从而能够减小这种变化对同步系统造成的危害。

## 3.6 本章小结

本章从统计角度出发，对链路传输过程中可能导致时延变化的多个因素进行详细而深入的分析，并且利用延时模型的特性，提供了一个系统性的同时应对多种链路延时波动因素的方法，即采用动态阈值法和滑动时间窗实时检测等来简便有效的应对固有抖动、暂时性时延突变和持久性时延变化等因素。这些方法都是基于历史实验样本数据，互相补充，能够完整而高效的应对复杂工业环境中同步系统链路延时的多种状况。不仅提高了从时钟的快速响应性和同步精度，还大大提高了从时钟的稳定性，使得同步系统能够更好应对拓扑结构变化和堵塞等现象

# 第四章　基于 PID 的时钟同步校正策略及仿真

本章从时钟伺服角度进行分析，并采用相关 PID 控制方法来提高从时钟系统稳定性。另外，为了能够计算得到准确的 offset，还将通过对从时钟晶振频率进行补偿以使得主从时钟频率尽可能保持一致来消除 offset 波动所带来的问题。其次，本章节还将介绍如何将稳中提到的统计方法应用到透明时钟中，由于透明时钟与边界时钟的根本差异，需要对统计方法进行调整才能更好的应用到透明时钟中。最后，本节将对文中采用的方法进行仿真，并对仿真结果进行分析总结。

## 4.1　时钟伺服系统

所谓时钟同步，最核心一环自然是从时钟根据计算数据对本地进行校正来实现同步，即时钟伺服系统。通常 IEEE1588 协议中采用的是 PI 控制策略，即利用 PI 控制器来对从时钟进行校正，而不是直接利用 offset 进行校正。这种做法能够一定程度提高从时钟的同步精度和稳定性，但它也存在如 PI 参数固定等缺陷而导致同步系统无法有效应对复杂多变的工业网络环境，这严重破坏了从时钟系统的稳定性。因此，为了提高从时钟校正过程的实时性和快速性，需要设计一套比较良好的时钟伺服系统来对从时钟进行校正。

### 4.1.1　通用 PTP 时钟伺服系统介绍

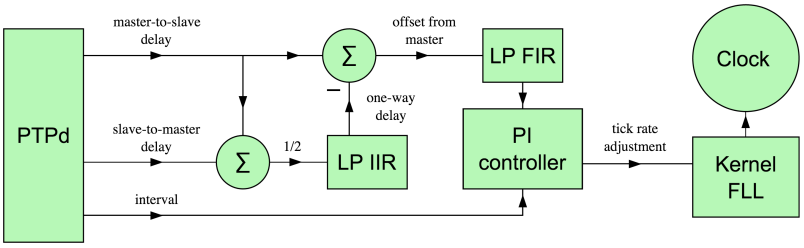


图 4–1 PTP 时钟伺服系统

Fig 4–1 The structure of PTP clock servo system

一种常见的 PTP 时钟伺服系统如图4–1所示[58]。从左到右表明了数据从 PTP 协议引擎逐渐流向从时钟。在该系统中，PTP 协议引擎通过周期性获取主从偏差和从主偏差，

将这两个偏差值通过均值滤波及低通滤波器获取到当前 offset 值。然后，将当前 offset 值与采样周期数据共同传递给 PI 控制器，由 PI 控制器来对从时钟进行校正以实现主从同步。该 PI 控制器由比例（P）和积分（I）两个环节共同构成，其中，比例环节可以用来消除主从时钟间的相位偏差，而积分项则可以用来消除系统的稳态误差，即消除主从时钟间的频率差[58]。

上述这样一套较为完整的时钟伺服系统可以较好的实现时钟同步，而且，PID 控制器由于具有方法简单、鲁棒性好和可靠性高等特点，在实际控制系统设备中得到广泛的运用，而且 PID 控制是在工业过程控制中应用最为广泛的一种控制方法。PID 控制器的基本结构由比例单元 P、积分单元 I 和微分单元 D 组成，分别对应目前误差、历史累计误差及未来误差，通过 Kp，Ki 和 Kd 三个参数的设定。PID 控制器主要适用于线性且动态特性不时变的系统，在不清楚受控系统特性的情况下，一般认为 PID 控制器是最适用的控制器。它是工业控制应用中常见反馈回路部件，会把收集的数据与一个参考值比较，并把该差值用于新的输入值计算中，这个新输入值的目的是让系统数据达到或者保持在参考值。另外，它还可以根据历史数据来调整输入值以提高系统准确度和稳定度。

### 4.1.2 基于统计方法的 PID 控制器原理

上文中采用的统计方法是通过在从时钟处累计 PTP 报文及链路延时样本，通过滤波、阈值限制与动态检测方法以提高链路延时精确度，从而得到更为精确的链路延时数据样本。不过，如果直接拿计算得到的主从偏差去进行从时钟校正，那么很有可能会破坏从时钟的稳定性。对于时钟同步系统，我们不仅仅希望主从时钟的时间偏差很小，更希望从时钟的波动很小，即具备良好的稳定性。或者说即使链路中发生了突然性的大幅波动，仍希望从时钟能够不会产生明显的波动，保持平稳运行。

所以，为了提高同步系统的稳定性和鲁棒性，这里采用了 PID 控制器，来对整个同步系统进行控制，以保证从时钟的稳定性和鲁棒性。假设控制器的输入偏差为 Error(k)，对应的控制器输出为 Output(k)，那么根据 PID 控制器原理，可以得到如下计算公式：

$$Output(k) = K_p * Error(k) + K_i * \sum_{i=1}^{k} Error(i) + K_d \Delta Error(k) \tag{4–1}$$

$$Output(k) = K_p * [\frac{1}{T_d} * \sum_{0}^{k} Error(i) + T_d * \frac{dError(k)}{dk}] \tag{4–2}$$

$$\Delta Output(k) = K_p(E(k) - E(k-1)) + K_i E(k) + K_d(E(k) - 2E(k-1) + E(k-2)) \tag{4–3}$$

$$\Delta Output(k) = Output(k) - Output(k-1) \tag{4–4}$$

其中，Error(k) 为主从时间偏差。希望达到的目的是使的 Error(k) 趋近于零，也就是使的主从时间差尽可能小。这里采用公式 (4-3) 来进行分析，假设当前时刻为 k，那么可以通过 Error(k)、Error(k-1)、Error(k-2) 来计算得出当前的控制量 $\Delta Output(k)$。所以，为了获取当前控制量，必须获得连续三个主从偏差值 offset。根据上面可以知道，通过三个连续偏差值 Error(k)、Error(k-1)、Error(k-2)，结合 PID 控制器可以计算出所需要的控制量，将该控制量作用于从时钟即可达到时钟同步的效果。如图4–2。

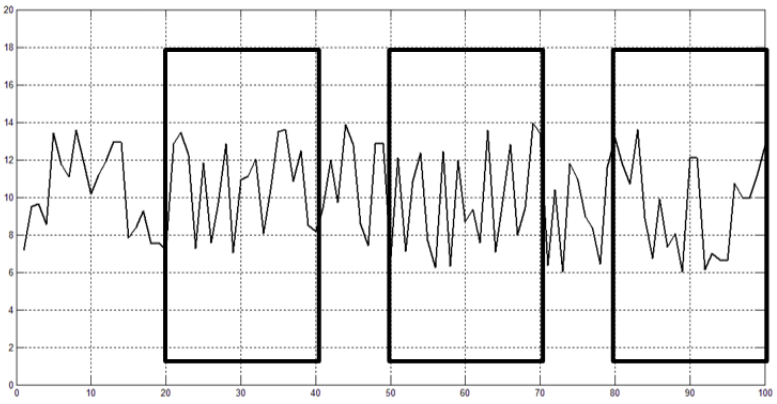


图 4–2 时间窗连续取值

Fig 4–2 The time-window value figure

那么，这里存在一个问题，就是如果链路延时发生了阶跃噪声，如果继续使用常规的 PID 控制，那么可能会导致控制量出现较大的变化，从而引发从时钟的不稳定。可以参考图4–3，使用一个长度为 10 的滑动时间窗来获取上面的连续偏差值，当发生阶跃噪声时，滑动窗内的取值也会发生较大的波动，不过，对于这种阶跃噪声，最初时无法判断是暂时性突变还是持久性变化，所以，为了保证从时钟的稳定性，这里要调整控制结构。

图4–3为检测阶跃噪声的取值方法，通过结合当前时延和时间窗内的历史时延数据来计算出阶跃噪声。

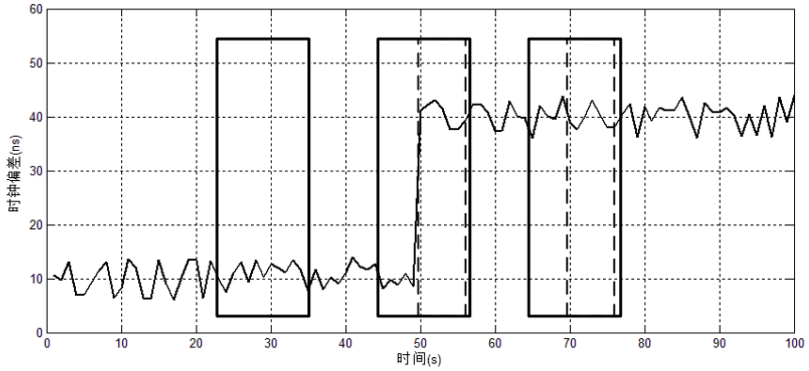


图 4–3 滑动时间窗阶跃噪声检测

Fig 4–3 The sliding-window step-signal detection figure

### 4.1.3 PID 控制器设计

基于上文分析，为了应对时延的阶跃噪声，可以对时延采用上文的方法来进行连续检测，以减小阶跃噪声对从时钟系统稳定性的破坏，这里采用了一种多模型 PID 控制系统，即通过检测主从时间偏差的变化情况来选择合适的控制器，从而保证从时钟能达到尽可能的稳定。

具体的结构如图4–4。

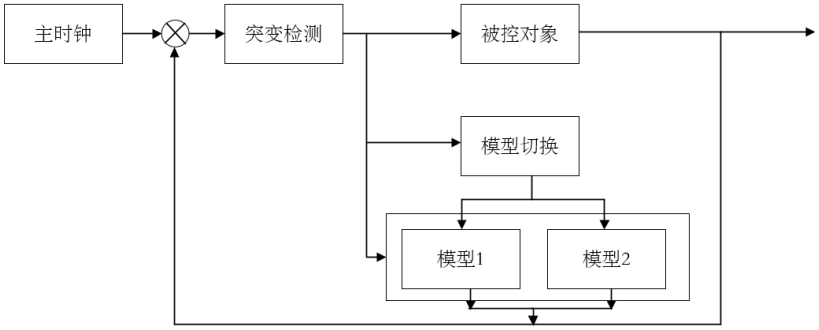


图 4–4 基于多模型 PID 控制器结构[70]

Fig 4–4 The multi-model PID controller

其中的原理是，在控制器前面，采用突变检测器，判断误差是否发生了阶跃噪声，如果没发生则使用正常的 PID 控制器；一旦检测发现阶跃噪声发生，则暂时使用自定义控制环节，保持偏差量不变，去对从时钟进行校正。同时继续等待一段时间，这段时间可以根据经验法去获取，如果误差重新发生阶跃恢复到原始状态或者稳定不变，那么就继续使用 PID 控制器。其中，突变检测器就是用来监测残差信号是否发生阶跃噪声，具

体的方法类似上一章节中的动态阈值法，通过在一定的时间窗内观测信号的变化情况，我们取时间窗长度为 10，如果发现当前信号处于突变中，那就切换控制模型，从 PID 控制模型切换到下列模型：

$$u(k) = u(k-1) \tag{4–5}$$

也就是使用历史数据来替代当前突变数据，从而保持从时钟的稳定性。当时间窗继续向前滑动，信号的抖动逐渐恢复。则可以恢复使用 PID 控制器来进行控制。

### 4.1.4　多模型 PID 控制器仿真及分析

首先，这里先对 PID 参数进行整定。所谓 PID 参数整定就是确定调节器的比例常数 $K_p$、$K_i$、$K_d$，一般而言，可以采用理论计算来进行参数整定，但是一般误差比较大。在当前工业实际中，较多的是采用工程整定法，例如经验法、衰减曲线法、临街比例带法和反应曲线法等等。这里采用经验试凑法来进行 PID 参数整定。首先，给定 $K_p$、$K_i$ 参数之，通过改变链路延时添加扰动，以观察控制曲线的形状，同时对 $K_p$、$K_i$ 进行调整直到同步曲线达到良好的同步效果；然后，在前面整定好的基础上再来调节 $K_d$ 微分参数，由于 $K_d$ 会抵制偏差变化，所以我们可以稍微减小 $K_p$、$K_i$ 然后继续调整到最佳值。经过了上面一步可以把 PID 控制参数整定好，然后，为了再仿真系统中测试上文中的多模型 PID 控制系统方法对阶跃噪声信号的优化效果，下面在时钟同步过程中，手动添加一个阶跃噪声变化量，使的从时钟接收报文出现一个较大的时滞，然后将对常规 PID 控制和多模型 PID 控制系统的仿真结果进行比较。

通过比较两个仿真曲线，可以看出，当使用多模型 PID 控制系统后，对于阶跃突变噪声信号有更好的适应性，从时钟也具有更好的稳定性。

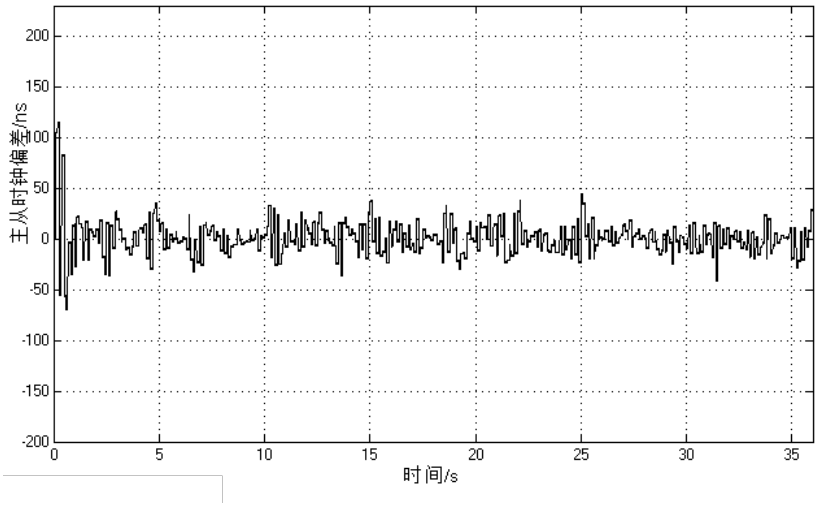


图 4–5 常规 PID 控制时延突变的同步曲线

Fig 4–5 The normal PID controller handling step signal of time-delay

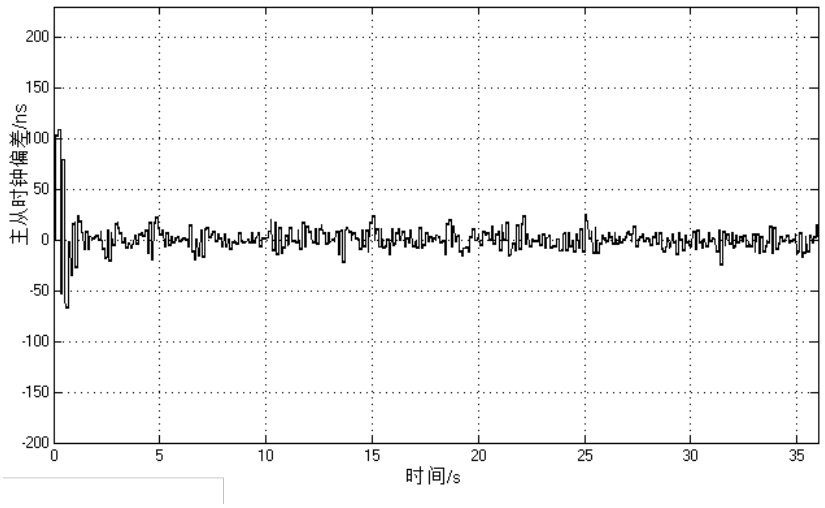


图 4–6 多模型 PID 应对阶跃噪声的同步曲线

Fig 4–6 The multi-model PID controller handling step signal of time-delay

## 4.2　统计方法在晶振源漂移补偿中的应用研究

由于在工业现场中采用的 PTP 时钟设备内部都是通过晶振来计时的，或者说内部时钟就是通过晶振的计数来实现的。然后，晶振本身物理特性比较复杂，一般用以下两个属性来进行描述：准确度和稳定度。其中准确度指系统误差，稳定度指随机误差。在 1588 同步协议中，晶振频率频偏都会影响到时钟的同步精度。IEEE1588 协议中假设在一次同步周期中，主从时钟的偏差会保持不变。然而实际上，虽然主时钟采用高稳时钟，稳定性较高，但是从时钟采用普通晶体振荡器，这使得从时钟频率与标准频率相比存在漂移，也就造成了主从时钟偏差一直处于变化中，也就是说，真实的情况并不满足协议中假设的那样主从偏差保持不变。

因此，由于由于从时钟晶振源的自身漂移，导致了主从时钟之间的偏差处于不断变化中。所以，应该采取手段来补偿从时钟晶振源的漂移，使得主从时钟晶振源保持一致，从而尽可能减小晶振源漂移造成的误差。

### 4.2.1　晶振频率对 1588 同步精度的影响

对于晶振频率误差，首先需要考虑其分辨率带来的影响。对于一个 20MHz 晶振源，其能分辨的最小时间为 50ns，对于低于该可分辨最小时间而言，就会被晶振自动忽略，从而引起误差。或者说假设当前真实时刻值为 340ns，而该晶振源表示的时间仍然为 300ns。所以，如果晶振源分辨率太差，则容易导致较大的误差。针对这种情况，需要在经济条件允许的范围内，尽可能提高分辨率以降低其不可识别度带来的偏差。

另外，晶振是物理设备，晶振源精度会同时受到外部环境和内部因素的共同影响，所以实际对外的输出频率是不断变化的。用 $f_{output}$ 表示晶振源对外输出的频率，$f_{standard}$ 表示标称频率，D 表示频率漂移程度，那么可以得到：

$$D = \frac{f_{output} - f_{standard}}{f_{standard}} \tag{4–6}$$

而在真实的 PTP 同步系统中，由于主时钟一般采用高稳时钟，所以可以认为主时钟的漂移非常小以至可以忽略，那么假设主从时钟频率漂移为 $f_{delta}$，计算方式如下：

$$f_{delta} = D * f_{standard} \tag{4–7}$$

也就是说，假设一次同步周期为 $T_{Sync}$，则一次 Sync 过程中主从时钟源的固有漂移值为：

$$T_{delta} = D * \frac{f_{delta} * T_{Sync}}{f_{standard}} \tag{4–8}$$

其中，$T_{delta}$ 就是一次同步周期内最大的时钟漂移。随着时间的流逝，这种漂移不会不会改善，而且会使得主从晶振源频率漂移越来越大，造成了漂移时间也越来越大，严重破坏了同步精度。

因此，必须对从时钟晶振漂移进行合适的补偿，使得每次同步周期中，从时钟的频率能够进一步靠近主时钟，从而减小频率漂移带来的不良影响。

### 4.2.2 统计方法在频率补偿中的应用

为了补偿从时钟频率，需要通过一种手段来测量主从时间的频率差值。这里采用的方法就是利用 Sync 报文的周期性发送，即主时钟是每隔 $T_m$ 时间段发送一次 Sync 报文，而从时钟也是每隔 $T_s$ 时间段收到一次 Sync 报文，假设主从时钟的频率是一致的，那么 $T_m$ 就应该接近于 $T_s$。然而实际上，由于从时钟频率漂移，使得从时钟的计数方式收到影响，导致 $T_s$ 并不等于 $T_m$，不过，正好可以利用这个特性来估算主从时钟的频率差值。

不过，要注意的是，$T_m$ 与 $T_s$ 不完全相等并不一定是从时钟漂移所带来的，还有可能是链路上传输的时间波动，导致 Sync 报文的发送和接收周期不一致。所以，不可以仅仅根据一次 Sync 报文的发送、接收周期来计算频率差值。针对这样的情况，下文采用了一种基于统计方法的频率补偿方法，下面进行详细介绍:

由于在上面的统计方法中，累积了很多的 Sync 报文样本，可以利用这些 Sync 报文来进行统计分析。

表 4–1 Sync 报文收发时间戳

Table 4–1 Timestamp for send and receive of Sync message

| Sync 次序 | 1 | 2 | 3 | ··· | n |
|---|---|---|---|---|---|
| 当前 Sync 报文发送时间 | $t_{s1}$ | $t_{s2}$ | $t_{s3}$ | ··· | $t_{sn}$ |
| 当前 Sync 报文接收时间 | $t_{r1}$ | $t_{r2}$ | $t_{r3}$ | ··· | $t_{rn}$ |

上面的表格可以很容易通过之前的样本数据得到，有了每个 Sync 报文的发送接收时间后，可以很简单的计算出 Sync 报文的发送接收周期波动值。

因此，能够得到主从时钟的周期样本，利用这些样本，可以采用均值法来计算主从时钟的周期，设主时钟平均周期为 $T_m$，从时钟平均周期为 $T_s$，那么可以得到主从时钟频率偏差为:

$$f_{delta} = \frac{1}{T_m} - \frac{1}{T_s} \tag{4–9}$$

利用 $f_{delta}$ 即可对从时钟频率进行校正。

表 4–2 Sync 报文收发周期

Table 4–2 Loop for send and receive of Sync message

| 编号 | 1 | 2 | 3 | ··· | n |
|---|---|---|---|---|---|
| Sync 报文主时钟发送周期 | $T_{m1}$ | $T_{m2}$ | $T_{m3}$ | ··· | $T_{mn}$ |
| Sync 报文从时钟接收周期 | $T_{s1}$ | $T_{s2}$ | $T_{s3}$ | ··· | $T_{sn}$ |

除此之外，我们可以对 Sync 报文采用上一章节中采用的动态检测方法，如果发现该时间段内的 Sync 报文发生暂时性时延突变，那么意味着内部的 Sync 报文传输时间会发生突变，那么我们可以选择“抛弃”这个时间段内的 Sync 报文，等到报文稳定传输后，再去计算频率差值。

## 4.3　统计方法在透明时钟中的应用

透明时钟是一种新的 PTP 设备类型，虽然它应用范围不如边界时钟更广，而且其本身需要成本更高，不过为了方法完整性，文中决定对于这类设备的同步优化进行分析。下面的分析主要针对透明时钟与边界时钟的差异性，来分析四种方法暂时性时延抖动优化方法，持久性时延变化优化方法，时延固有波动优化方法和主从时钟源频率补偿方法的调整。

### 4.3.1　透明时钟与边界时钟差异性

透明时钟 (Transparent Clock, TC) 是 1588v2 中引入的一种新的 PTP 设备类型，它的基本原理是通过内部驻留时间桥来测量事件消息在通过设备时的驻留时间，然后把驻留时间存储到事件消息的修正域中。当从时钟接收到该报文时，可以通过解析修正域而得知该报文在链路传输过程中所经历的传输时间。

透明时钟有两种工作机制。端对端透明时钟将转发所有符合网络规则的报文，同时测量出时间消息驻留时间，并把测量结果作为修正值累计到事件消息的修正域中；点对点透明时钟每个端口上额外添加一个延时测量模块，这个模块可以计算在该练路上，执行相同机制的对方端口之间的链路传输延时。所以说，点对点透明时钟不仅可以修正驻留时间，还能对端口之间的链路传输延时进行修正。不过，它只会对 Sync 报文和 Followup 报文进行处理，丢弃 DelayReq 和 DelayResp 报文。

依据上述透明时钟的原理，可以将其与边界时钟进行比较得到以下几个不同点：

- 首先，边界时钟完全无法测量链路传输延时，而透明时钟可以计算在设备内部的驻留时间，因此，即使报文在传输过程中，遭遇排队或堵塞等暂时性时延波动，从

时钟仍然能够通过减去修正域的值来消除其影响；

- 另外，假设链路发生了拓扑结构变化，所有报文的传输路径变动，依然可以利用透明时钟的特性消除因此带来的链路传输延时持久性变化。

### 4.3.2 统计方法在透明时钟中的调整与优化

由于边界时钟与透明时钟的固有差异，如果想把上述的统计方法应用到透明时钟，则需要做出一些调整。而有余透明时钟比边界时钟更高级，能够很好的衡量报文传输路径中消耗的延时，所以，如果在实际系统中采用了透明时钟，那么能够极大简化统计方法，也能够得到更好的同步精度。但是，目前工业应用中，透明时钟并没有完全替代边界时钟，一方面由于透明时钟成本更高，另一方面由于透明时钟的推出时间晚于边界时钟，工业中很多同步场合已经采用了边界时钟同步架构。

对于透明时钟，由于它能够较好的测量链路传输延时，那么可以作出以下调整。

#### 4.3.2.1 暂时性时延抖动

由于暂时性时延抖动主要源自中间设备内部的排队和堵塞，而透明时钟能够把这些时间添加到修正域中，因此即使设备发生排队现象，仍然能在从时钟处很好的消除这部分时间的变化，因此，如果一条链路中全部采用了透明时钟，那么可以不需要考虑此类时延抖动。不过，如果链路中同时采用了边界时钟与透明时钟，那么仍然有必要考虑方法实现；

#### 4.3.2.2 持久性时延变化

所谓持久性时延变化主要来自于链路拓扑结构的变化，一旦链路变化，那么报文的链路传输时间则会发生变化，而对于统计方法而言，这也就意味着之前的统计样本统统失效。在透明时钟中，会减去设备内部的时间波动，那么只有纯粹链路的传输时间变化会导致样本的移动，不过，一般而言，纯粹路径耗时非常微小，所以可以当作持久性时延变化不会对透明时钟有很大影响；

#### 4.3.2.3 时延固有波动

对于时延固有波动，无论透明时钟还是边界时钟都是无法消除的，所以，仍然要针对固有波动进行统计处理，以减小时延的波动造成从时钟源的波动，提高从时钟稳定性；

#### 4.3.2.4　主从时钟源频率补偿

主从时钟源与中间传输设备没有太大关联，无论是边界时钟还是透明时钟，都需要进行频率补偿。只不过，由于该方法是利用 Sync 报文的发送周期与接收周期的差异来进行优化的，所以会受到 Sync 报文传输时间的影响，如果 Sync 报文路径传输时间变长，那么接收周期变大，导致计算出现偏差。所以，对于透明时钟而言，可以忽略报文传输的时间变化，因此透明时钟在频率补偿时会有更好的精度。

## 4.4　基于 stateflow 的 PTP 仿真系统

为了能够对本论文中所阐述的优化方法进行验证，在实验室师兄的基础上，继续利用 stateflow 工具，搭建并优化了基于 IEEE1588 协议的时钟同步系统。为了能够实现仿真验证，该系统中创建了三台 PTP 时钟，包括两个普通时钟和一个边界时钟，其中一个普通时钟扮演 Grand Master 的角色，另一个时钟扮演从时钟角色。边界时钟在二者中间，与 GM 时钟进行同步，同时向从时钟发送 Sync 报文，最终达到三者处于同步。

### 4.4.1　PTP 仿真系统

下面仅对该仿真系统中比较重要的部分进行介绍，同时利用该系统来进行仿真验证上文所提方法。Simulink Stateflow 工具箱是 matlab 平台上的一种可视化建模工具，对于状态机、流程图等的创建，或者对 PTP 时钟同步系统之类事件驱动系统进行建模仿真具有非常良好的应用性。下面对其中与本文方法相关的一些模块进行简要地介绍，不作过多赘述。

#### 4.4.1.1　本地时钟模块

本地时钟模块：该模块为时钟模块提供本地时钟，对同步系统而言，就是要让从时钟的本地时钟模块能够在方法校正后实现与主时钟本地时钟模块保持一致，才算达到良好的同步效果。在实际系统中为晶振驱动，在仿真模型中，使用脉冲生成器来驱动，在每个脉冲上升沿处使的脉冲计数器加 pulseCount 值，另外设 rate 值来调节本地时钟精度。例如，使用的脉冲器为 100MHz，那么脉冲计数器 pulseCount 每次累加 10ns，如果用 rate = 100，那么表示每 100 个 10ns 对应 1 个时钟 clock，即 1 微秒。用 clockCount 来表示本地时钟，每产生一个 clock 就使 clockCount 加 1。另外，由于真实的时钟还存在频率偏差，为了模仿这种情况，再给 pulseCount 添加一个偏差值，frequencyOffset，即：

$$clockCount = (pulseCount + frequencyOffset)/rate \tag{4–10}$$

对于该频率偏差 frequencyOffset，应该用时钟伺服系统来进行校正。

#### 4.4.1.2 定时器模块

由于同步系统中存在很多中定时触发或超时事件，例如 Sync 报文的周期性发送，Announce 报文的超时等待事件等。所以，添加了这个模块，为系统中所有周期性或有超时发生的事件进行触发。

下图是 Announce 超时事件定时器模块作为参考，其他定时器模块与之非常类似。

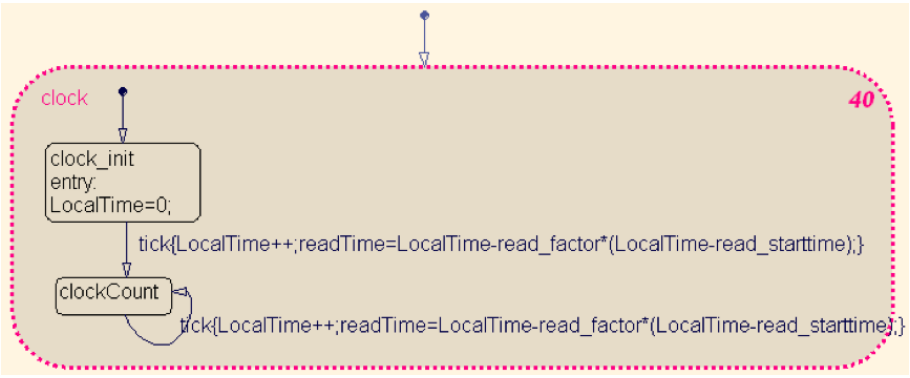


图 4–7 本地时钟仿真模型

Fig 4–7 The Simulation model of Local Clock

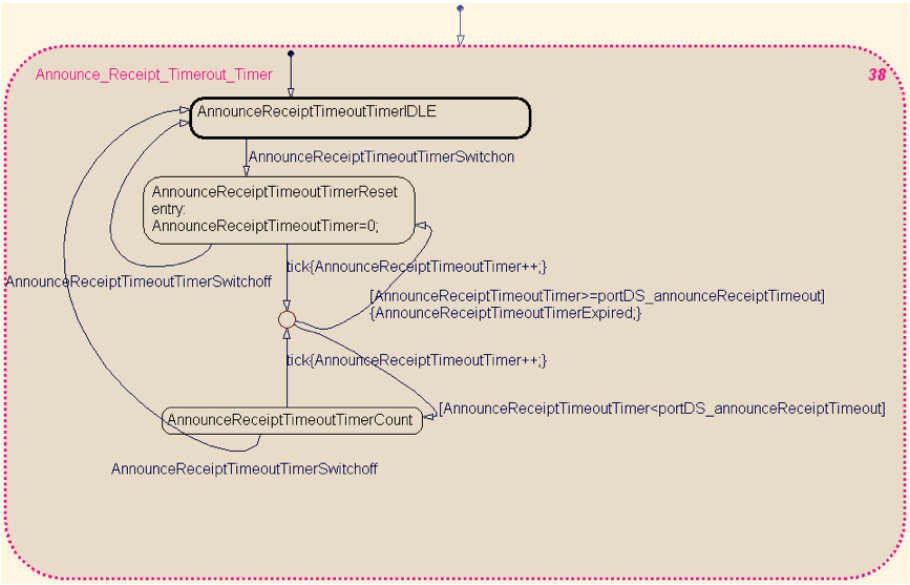


图 4–8 Announce 报文超时事件定时器仿真模型

Fig 4–8 The Timer model for Announce message receipt timeout

#### 4.4.1.3　报文收发模块

该模块主要包括每个时钟内部对外的收发报文。对于接收模块，它会时钟监听某外部信号，一旦有报文传入则会立即触发系统进入报文接受流程，该流程会讲所收到报文的数据全部存储到本地，同时向系统中发出报文接受完成信号。对于发送模块，会先把数据根据对应报文格式组装成对应的报文，把数据存入该报文中，然后从时钟对应的端口发出。

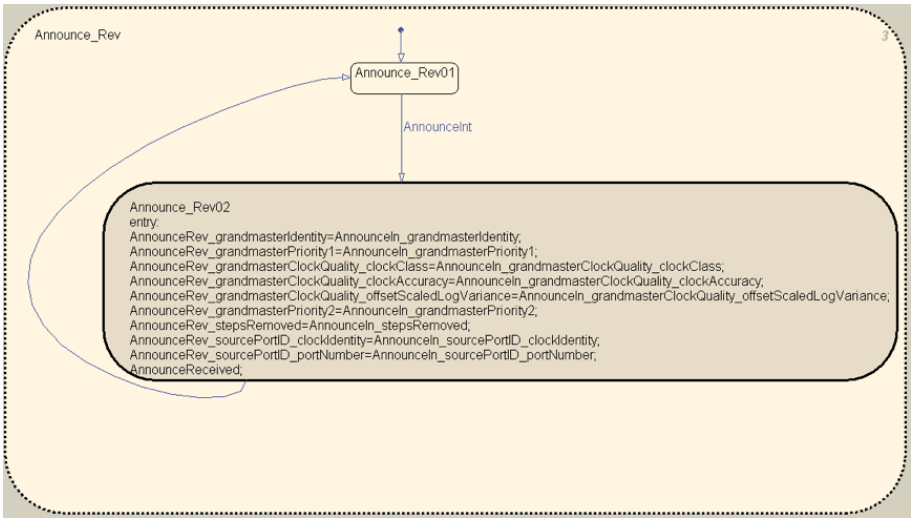


图 4–9 报文接收模块仿真模型

Fig 4–9 The model for receiving message

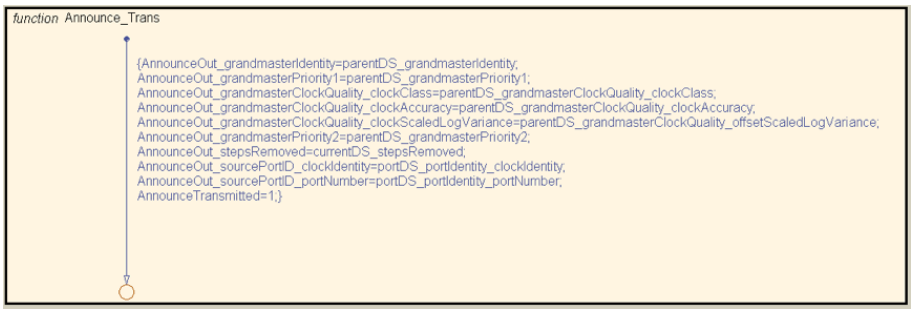


图 4–10 报文发送模块仿真模型

Fig 4–10 The model for sending message

### 4.4.2 算法仿真思路及指标分析

下面，为了能够在 stateflow 仿真平台上验证上述所提统计方法的有效性，下面将从以下几个角度来进行模拟与验证。

- 对于传输延时抖动，可以在传输中添加随机延时噪声来进行验证；
- 对于暂时性时延突变，可以对传输延时添加阶跃噪声，并观察对同步过程的影响；
- 对于持久性变化，可以在运行时链路传输中添加永久性固有延时，并观察结果；
- 对于控制策略，可以分别采取普通 PID 控制器校正和基于多模 PID 控制策略进行同步效果对比。

为了验证本文所提的优化方法，下面会在整个同步系统外部，通过监听主时钟和从时钟的时钟偏差曲线，从而可以知道该同步系统的同步性能。主要的比较指标有下面几点：

- 偏差值：通过观察主从时钟的偏差值，可以看出该同步系统的同步效果，这也能直接反映真实的同步精度。因此，可以以此来检测同步系统中链路延时抖动和暂时性抖动对同步精度造成的影响。
- 鲁棒性：当系统中发生持久性时延变化，那么会造成从时钟发生较大的抖动，所以，可以依据此来检测针对持久性时延变化的优化方法的效果，如果该优化方法能够使的从时钟的抖动减小，那么表示从时钟的鲁棒性得到了提高。
- 收敛速度及稳定性：可以通过观察主从偏差的波动值，并计算其收敛时间。尤其是在链路环境发生变化，即类似工业环境中的多种复杂因素导致系统不稳定，以此来观察系统的收敛速度和最终的稳定性，从而来判断本文采用的基于多模型的 PID 控制器的效果。

## 4.5 算法仿真及结果分析

为了验证上文中提到的最小二乘法，动态阈值法和实时动态检测方法的实际效果，下面一次对这三种方法进行验证。在验证中，可以通过认为模拟延时抖动、排队堵塞和拓扑结构变化等因素导致的时延变化，通过比较采用相关方法前后的同步曲线来得出相关结论。

下文中的同步曲线是指从时钟与主时钟之间的随时间变化的实时同步偏差曲线。

### 4.5.1 验证最小二乘法对延时抖动的优化

为了验证最小二乘法对延时抖动的优化效果，可以在从时钟计算链路传输延时时添加随机扰动，并模拟该扰动符合标准正态分布，得到下面两个仿真结果。首先，为了模

拟生产环境中的随机噪声，在传输链路中加入服从正态分布的随机影响因子，然后进行同步，可以得到下面的同步曲线，见对比图4–11与4–12。

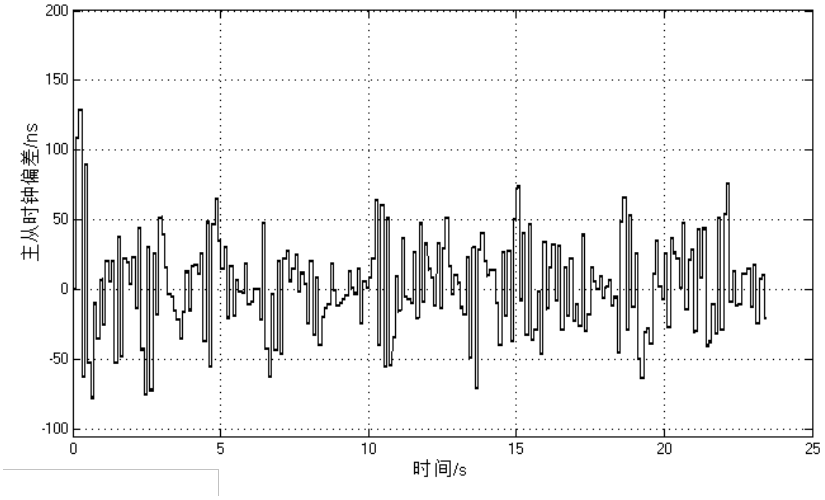


图 4–11 添加正态分布随机抖动的常规同步曲线

Fig 4–11 The Synchronization curve, within Normal Distribution random noise

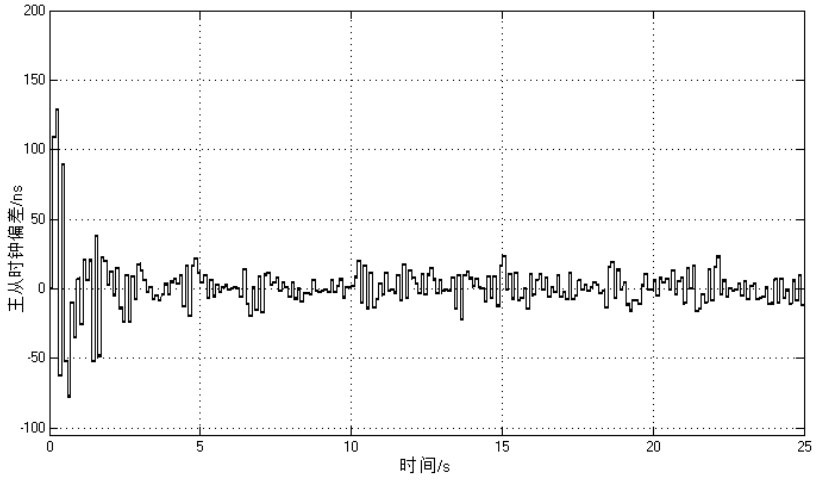


图 4–12 添加正态分布随机抖动且利用最小二乘处理时间窗长度为 10 的同步曲线

Fig 4–12 The Synchronization curve dealing with least square method, within Normal Distribution random noise. Time-window length is 10

首先，对比曲线4–11与4–12，可以看出，未做任何处理时的同步曲线在-80ns － +80ns之间波动，而采用最小二乘法进行样本处理后的同步曲线在-20ns － +20ns之间波动，这

说明最小二乘法处理后可以明显减小随机噪声对同步结果的影响，从时钟振动幅度减小，从时钟具有更好的同步效果。说明该方法能够很好的减小随机抖动对从时钟系统的影响。

另外，由于最小二乘法需要考虑时间窗的长度，即计算回归曲线所需要的最少的样本个数 m。为了了解样本个数对时钟同步的影响，下面再按下式给时间窗长度 m 赋值，

$$m = 30 \tag{4–11}$$

可以得到下面的同步曲线4–13。

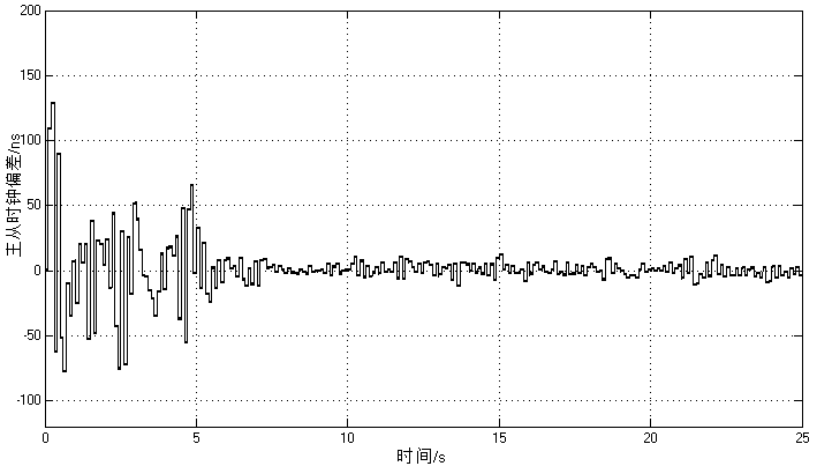


图 4–13 添加正态分布随机抖动且利用最小二乘处理时间窗长度为 30 的同步曲线

Fig 4–13 The Synchronization curve dealing with least square method, within Normal Distribution random noise. Time-window length is 30

通过对比曲线4–12与4–13，可以看出，时间窗长度 m 值为 10 时，收敛速度相对较快，在 3s 左右就进入收敛了。当长度 m 取值为 30 时，收敛速度相对较慢，直到 6s 左右才进入收敛。但是，可以看出，m 为 30 时的收敛后的精度比 m 为 10 时的收敛后精度更高，这是因为采用更多的样本，可以得到更为精确的回归直线，不过这需要花费更多的时间。

### 4.5.2 验证动态阈值法对暂时性时延突变的优化

首先，由于动态阈值法是针对一个固定时间窗内的样本进行数学处理，这里为了仿真，选定该时间窗长度为 10，即一次累计 10 条样本后，再来对对从 $t_{k1}$ 时刻到 $t_{k(10)}$ 时刻之间的估计时延来计算它们的标准差 $\sigma_k$，其中，k 是指当前的时间窗编号，从 1 开始递增。每次计算完一个时间窗后，就将 k 加 1，重新进入第 (k+1) 个时间窗继续计算。

为了模拟暂时性时延变化，在同步系统运行中间，突然加入一个时延突变，并保持一小段时间。并分别仿真出采用动态阈值法前后的时钟同步曲线。首先，为了模拟暂时性时延突变，在仿真接近 2 秒时，为链路传输延时添加一个阶跃噪声信号，并观察其同步曲线情况，同时取时间窗长度为 10，可以得到如下结果。见对比图4–14和4–15。

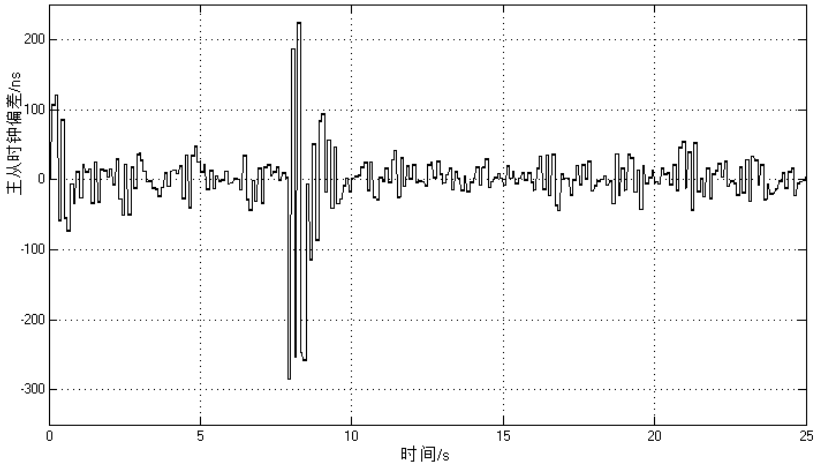


图 4–14 添加暂时性时延突变后的同步曲线

Fig 4–14 The Synchronization curve, within temporary time-delay mutation

然后，保持上面的阶跃噪声信号不变，采用文中采用的动态阈值法来进行优化，可以得到如下的结果如图4–15。

通过上面的对比曲线，可以看出，当我们在同步系统运行到 8s 时，添加一个时延突变后，在图4–14中，同步值立即跃变到-380ns - 320ns 之间，这也就表明真实同步系统中如果一旦遇到阶跃突变噪声，就会导致从时钟系统发生很大幅的阶跃波动，这会严重破坏从时钟系统的性能；而在图4–15中，通过采用动态阈值法，可以检测到从时钟发生的大幅的阶跃波动，并且立即将该波动限制到给定范围内以保证从时钟的稳定。

表 4–3 动态阈值法第 k 个固定时间窗

Table 4–3 Mixed Time Window of Dynamic Threshold Method

| 时间 | $t_{k1}$ | $t_{k2}$ | $\cdots$ | $t_{k9}$ | $t_{k(10)}$ |
|---|---|---|---|---|---|
| 当前测量时延 | $d_{k1}$ | $d_{k2}$ | $\cdots$ | $d_{k9}$ | $d_{k(10)}$ |
| 动态阈值法时延估计值 | $\overline{D_{k1}}$ | $\overline{D_{k2}}$ | $\cdots$ | $\overline{D_{k9}}$ | $\overline{D_{k(10)}}$ |

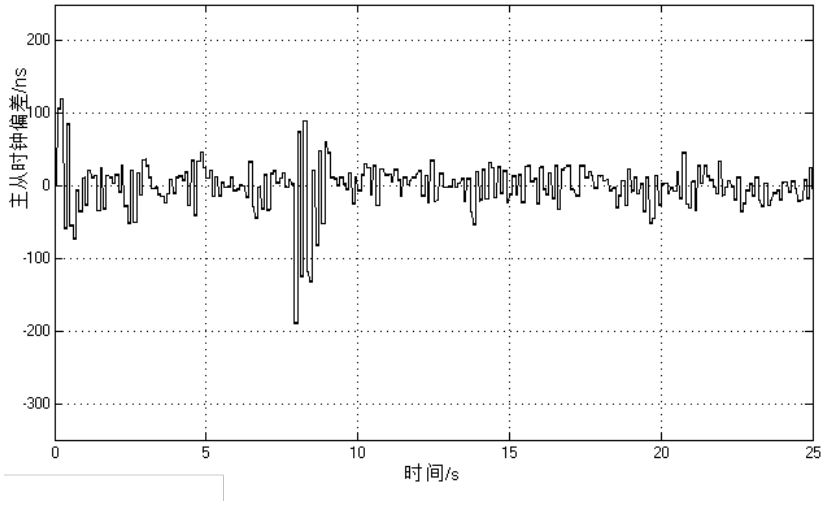


图 4–15 添加暂时性时延突变后，采用动态阈值法处理前后的同步曲线

Fig 4–15 The Synchronization curve with dynamic threshold value method, within temporary time-delay mutation

另外，在上一章中还提出过动态阈值的截止函数如下：

$$G(d_i) = \begin{cases} d_i, & |d_i| < \alpha * \sigma_i \\ \alpha * \sigma_i, & d_i \geq \alpha * \sigma_i \\ -\alpha * \sigma_i, & d_i \leq -\alpha * \sigma_i \end{cases} \qquad \alpha > 0 \tag{4–12}$$

在该函数中，采用了一个控制量 $\alpha$ 来调节该动态阈值的大小，为了了解该控制量 $\alpha$ 对时钟同步过程的影响，下面分别对 $\alpha$ 取下面值进行对比仿真。可以得到下面的仿真曲线。

$$\alpha = 3 \tag{4–13}$$

$$\alpha = 1.5 \tag{4–14}$$

对比曲线4–16和4–17可以看出，当 $\alpha$ 取值减小，则动态阈值降低，对于时延突变也更为敏感，而且一旦发生了时延突变，则会把同步值限制在一个较小的范围内部。因此，降低 $\alpha$ 值可以提高同步系统对暂时性时延突变的敏感度，但同时如果过于敏感则有可能会导致误检测，因此需要具体实践经验积累来进行调节。

通过上述对比仿真结果可以看出，利用动态阈值法对阶跃噪声进行优化后，从时钟的瞬间振荡明显减小，这意味着从时钟具备了更好的鲁棒性和稳定性，不会轻易随着网络负载变化而发生较大波动。

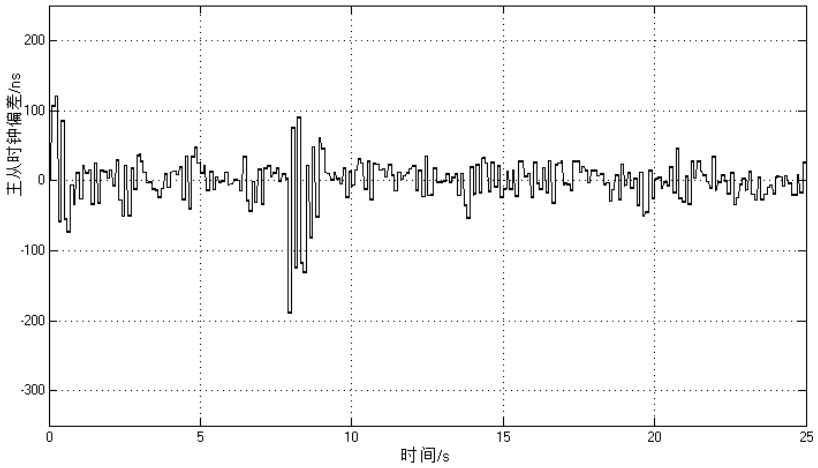


图 4–16 添加暂时性时延突变后，采用动态阈值法控制量为 3 时的同步曲线

Fig 4–16 The Synchronization curve with dynamic threshold value method, within temporary time-delay mutation, and control parameter equals to 3

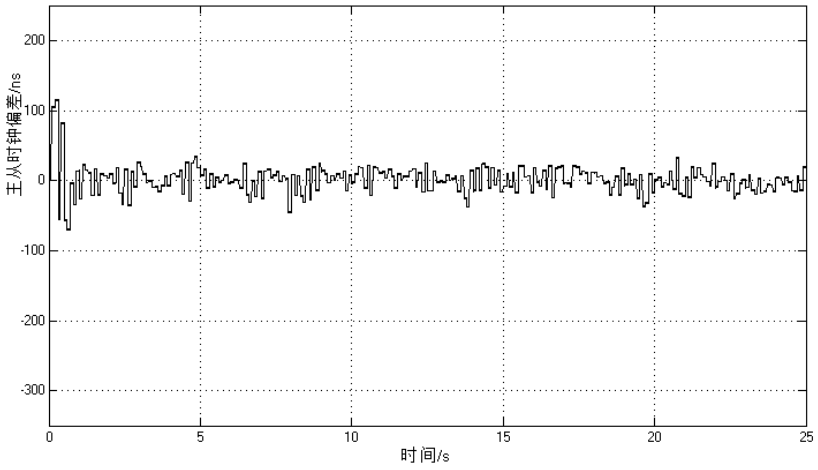


图 4–17 添加暂时性时延突变后，采用动态阈值法控制量为 1.5 时的同步曲线

Fig 4–17 The Synchronization curve with dynamic threshold value method, within temporary time-delay mutation, and control parameter equals to 1.5

### 4.5.3 验证实时动态检测法对持久性时延变化的优化

为了模拟持久性时延变化，在同步系统运行中间，加入一个恒定的时延变化，并保持不变。然后分别仿真出采用实时动态检测方法前后的时钟同步曲线。见对比图4–18与4–19。

为了模拟持久性延时变化，给链路延时添加固定的增量，并且保持不变，然后可以得到如下的同步曲线:

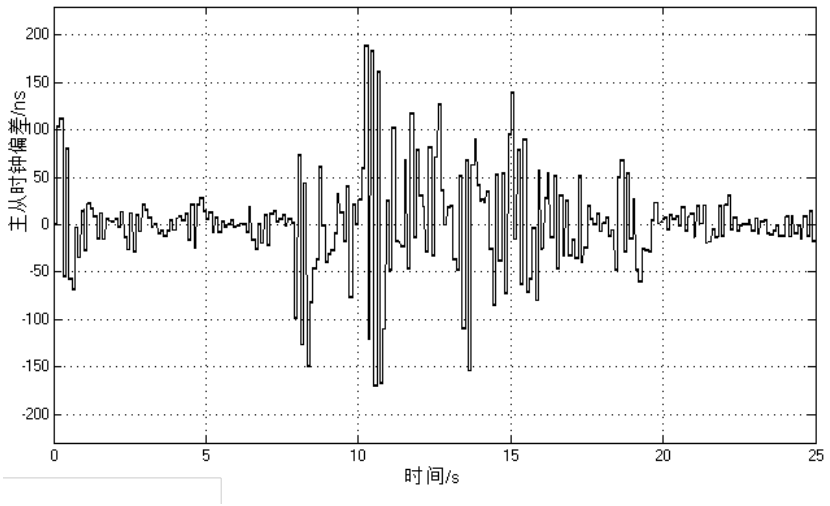


图 4–18 添加持久性时延变化后样本逐步覆盖的同步曲线

Fig 4–18 The Synchronization curve, within persistence time-delay change

然后，保持上面的固定增量，采用动态检测方法来进行处理，可以得到同步曲线4–19。

可以发现，在图4–18中，同步系统运行到 8s 时添加持久性时延变化，同步系统由于链路传输延时变化而随着慢慢变化，但由于从时钟系统并为检测到链路发生了重配置，所以还是按原始方法进行同步，这使得同步系统需要大概 10s 左右时间才能恢复正常，即在这 10s 内逐渐消除原始样本对新链路样本的影响。而在图4–19中，可以看到同步系统在 6s 左右的时间内即恢复正常，这是因为实时检测，发现样本朝某一个方向发生固定变化，因此使旧有样本立即失效，重新同步。

同时，在上一章中提出了滑动时间窗动态检测方程:

$$R_c = \begin{cases} 0, & D_c < \omega * \overline{DT} \\ 1, & D_c \geq \omega * \overline{DT} \end{cases} \tag{4–15}$$

$$\omega \approx (1 - 1.5) \tag{4–16}$$

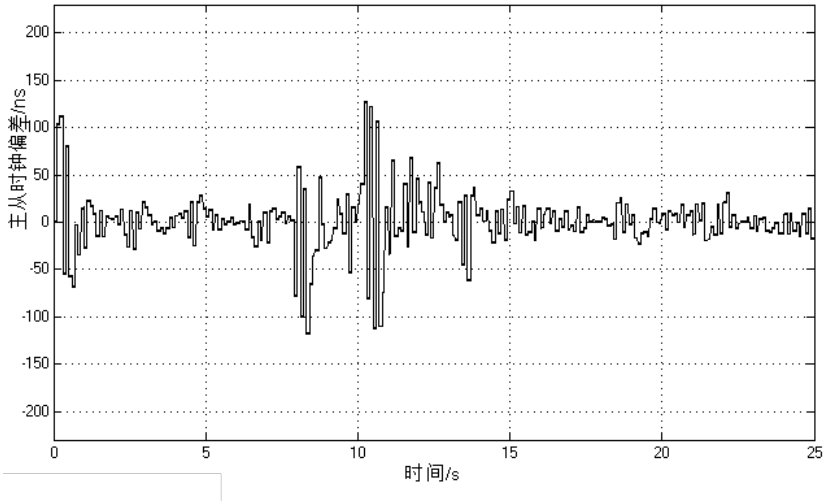


图 4–19 添加持久性时延变化后实时检测 w=1.5 的同步曲线

Fig 4–19 The Synchronization curve with real-time dynamic monitoring method, within persistence time-delay change, and w equals to 1.5

该方程中，$\omega$ 为对时延变化的容忍度，可以自己设定。为了了解容忍度 $\omega$ 对于时钟同步系统的影响，取

$$\omega = 1.2 \tag{4–17}$$

来进行仿真，可得到下面的仿真曲线。

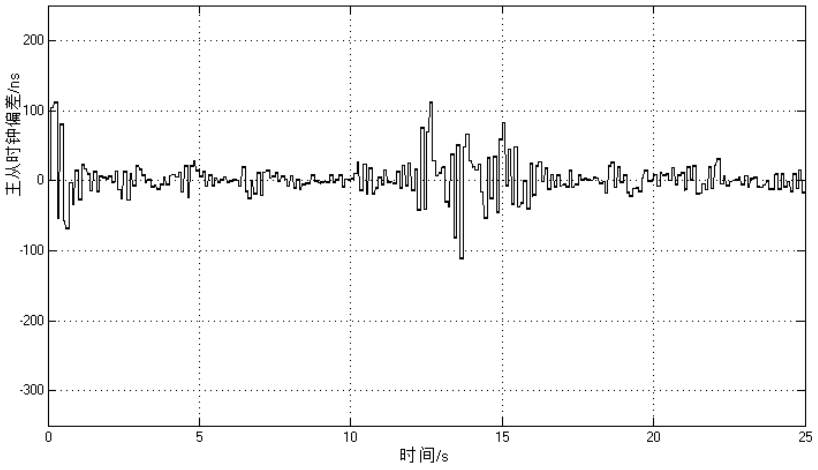


图 4–20 添加持久性时延变化后实时检测 w=1.2 的同步曲线

Fig 4–20 The Synchronization curve with real-time dynamic monitoring method, within persistence time-delay change, and w equals to 1.2

通过对比4–19和4–20可以看出，前者在 t=11s 左右才检测到持久性变化，而容忍度更低的后者在 t=8s 左右就已经检测到了。也就是说，当降低容忍度后，系统可以更早发现持久性变化，立即使旧有样本失效并重新进入新一轮同步，只需要更短的时间便可以重新恢复正常。

因此，利用实时动态检测法对持久性时延变化进行实时检测后，可以在系统中发生持久性时延变化后通过实时检测能够迅速发现该变化，然后采取相关措施后使得同步过程快速恢复正常，这提高了同步系统对外界变化的快速响应性。

### 4.5.4 验证统计策略对晶振源频率补偿的优化效果

为了在仿真系统中模拟从时钟晶振源漂移的情况，假设要模拟从时钟频率在不断衰减，即时间周期越来越长这样一种情况，那么可以给从时钟频率添加一个固定的衰减，使得从时钟的频率处于不断的小幅衰减中。然后，再利用上面采用的频率补偿法，不断的对从时钟频率进行补偿，以保持主从时钟源频率尽可能的一致。

下面，先给从时钟源频率加上一个固定的衰减值，并观测主从时钟偏差的变化情况，可以得到同步曲线4–21。

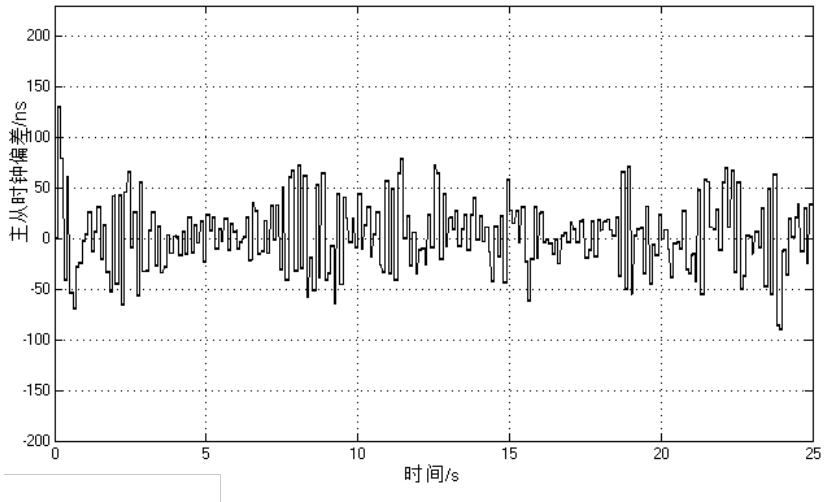


图 4–21 为从时钟源频率添加固定衰减后的同步曲线

Fig 4–21 The Synchronization curve, within persistence time-decrease in slave clock

可以看出主从时钟源的偏差处于不断的波动之中。然后，为从时钟频率进行补偿，可以得到同步曲线4–22。

通过上面的比较，可以看出，在图4–21中不使用频率补偿时，由于从时钟频率漂移导致主从时间差越来越大，需要校正的幅度也越来越大，时钟同步曲线在-80ns － 60ns

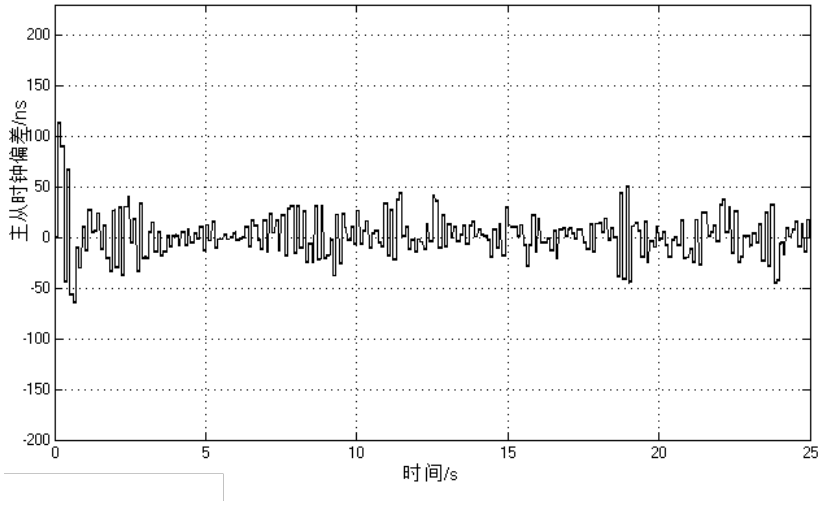


图 4–22 采用晶振源频率补偿算法后的同步曲线

Fig 4–22 The Synchronization curve by frequency compensation, within persistence time-decrease in slave clock

之间波动。但是当对从时钟频率进行补偿后，如图4–22，使得主从时钟的频率更加接近，削弱了从时钟漂移带来的偏差，时钟同步曲线在-50ns － 50ns 之间波动，幅度减小，更加的稳定。

## 4.6　本章小结

在本章节中，为了提高从时钟的稳定性，常规的做法是直接使用计算误差 offset 值对从时钟进行校正，不过那种做法会使得从时钟具有较大的波动，而且不能有效利用历史数据来进行调节。因此，本小结首先阐述了时钟伺服系统的概念，使用时钟伺服系统一般能够较好的提高同步系统的稳定性。然后，文中提到常规 PID 控制器校正，不过，为了应对延时阶跃噪声这种情况，文中又采用了一种多模型 PID 控制系统，即通过对控制误差进行检测，一旦发现控制误差发生阶跃变化，就切换控制器，以防止从时钟发生较大抖动，同时，在一定的观察时间段内，如果控制误差阶跃恢复到原始值，则立即切换回常规 PID 控制，否则，直到观测时间短结束，才切回常规 PID 控制。阶跃噪声信号会使得常规 PID 控制器控制后的从时钟发生频繁的调整，这会影响到依赖它的整个系统的抖动。而采用多模型 PID 控制系统可以把持续的波动转变成一个瞬时性的变化，这可以明显减小从时钟的频繁抖动，使整个系统更加稳定。

然后，对于常规的从时钟频率漂移补偿方法，一般都是通过记录一段时间内的 Sync 报文的发送和接收时间，以此来估算主时钟和从时钟各自的频率，并计算出二者的频率

差来进行校正。这种做法没有本质错误，只是不能很好应对 Sync 报文在链路传输过程中发生的变化，一旦用来计算的 Sync 报文在传输时发生了较大的延时，那么从时钟处接收到的 Sync 报文就会滞后，从而导致频率计算出错。而本节中利用第三章中的统计方法，在从时钟处累计了一定 Sync 报文样本，可以先通过第三章的动态监测方法对报文进行过滤，即如果发现 Sync 报文发生了暂时性时延突变或者持久性时延变化，那么就丢弃当前时间段内的 Sync 报文，直到发现 Sync 报文传输较为稳定，才利用这些报文的收发时间来计算频率差。相对现有方法，它可以得到更加准确、不受外界因素影响的频率差。然后，为了适应新的 PTP 设备－透明时钟，文中首先对比了边界时钟与透明时钟的区别，然后将文中的多种统计方法依次进行调整，优化，使得这些统计方法能够更好的适用于透明时钟。最后，文中通过自己搭建基于 stateflow 的 PTP 仿真平台，对于文中采用的统计方法进行了仿真验证，通过最终的仿真结果可以看出统计方法不仅能够很好的提高同步系统对多种时延变化的鲁棒性，更能提高整个系统的同步精度和稳定性。

常规的时钟伺服系统一般采用 PID 控制器方法来进行从时钟控制，不过这种方法不能很好的处理链路延时发生阶跃噪声这种情况，因此，为了应对链路延时中发生的暂时性时延突变与持久性时延变化中触发的阶跃噪声信号，文中采用的基于多模型 PID 控制系统，它会对校正残差进行突变监测，一旦发现阶跃噪声，则立即切换控制器以确保从时钟的稳定性。相对现有方法，它能减小从时钟的频繁抖动，使的从时钟系统运行更加平稳。而且，由于常规的从时钟频率漂移补偿方法都是通过一段时间内的 Sync 报文来计算主从频率差，但并没有考虑到 Sync 报文在链路传输过程中发生的时延变化。而本文采用的动态监测方法会对报文进行过滤，只选取稳定传输下的 Sync 报文来进行计算，可以得到更加准确、不受外界因素影响的频率差。

# 第五章 总结与展望

行文至此，文章已经细致地对时钟同步技术及 IEEE1588 协议的相关问题进行了深入的研究和探讨，同时也创新性地从数学统计角度入手，采用了动态阈值法、基于固定时间窗的实时动态检测算法等，用来处理报文链路传输时延变化。同时也对于主从时钟源晶振漂移进行了分析，并从统计角度给出了相关的解决方案，而且通过文末的仿真能发现这些算法不仅能够在一定程度上提高系统的同步精度，而且能提高同步系统的稳定性和鲁棒性，对于工业环境下复杂的网络环境有良好的动态适应性。

下面，对于文章进行总结和展望。

## 5.1 总结

首先，由于当前分布式系统等应用越来越广泛，而其中时钟同步问题也已经成为了束缚分布式系统发展和实现很多实时性需求的一大瓶颈。而为了达到工业系统中所需要的时钟同步精度，通过分析了解发现以往的时钟同步方法如 GPS、NTP、SNTP 等技术均由于各种原因，如国防安全性、时钟同步精度和稳定性等，而无法满足当前工业环境对时钟同步的需求。而 IEEE1588 精密时钟同步协议由于其自身的易部署性、价格适中、同步精度高的多种特性使得其成为了当今非常重要且应用广泛的时钟同步技术。

然后，通过深入了解发现，IEEE1588 时钟同步协议在实际运行中，并没有达到理想的微秒级甚至亚微秒级的时钟同步精度。于是，通过深入分析协议内容和实际运行背景，提炼出其中对同步精度破坏最大的两个因素：链路传输延时的随机性和变化性；从时钟校正策略。针对前者，文中创新性地从数学统计角度出发，把链路传输延时数据作为样本，并且根据其数学特性分别采用了动态阈值法和基于固定时间窗的实时检测算法等，对链路传输延时中的暂时性时延突变和持久性时延变化进行优化；然后，通过对主从时钟源晶振漂移进行了分析，采用了统计方法来进行补偿，另外还针对透明时钟与边界时钟的不同特性，分析了统计算法如何应用到透明时钟的场合中。最后，利用 stateflow 工具，在 matlab 平台上搭建了一套完整的时钟同步仿真系统。文中利用该系统来对上文所提算法进行验证，通过最后的结果可以看出所提算法不仅能够改善同步系统的时钟同步精度，而且还能够充分应对链路中发生的由于拓扑结构变化等因素导致的持久性时延变化，这是当前很多同步算法并不具备的。

相信在上面所作的工作和研究成果，能够为之后的研究者提供一条不一样的思路，而且这些仿真结果能够为他们提供一定的实验依据。

## 5.2 展望

在上文中，通过对 IEEE1588 时钟同步协议进行深入研究，发现了其中存在的严重问题，并且也采用了相应的解决方案，通过最后的仿真结果可以看出，这些方案确实能够改善同步系统的同步精度、稳定性和鲁棒性。

但是，这并不意味着这些方案已经没有缺陷了。

后续的研究者可以从以下两个方面着手，进一步对文中所提的算法进行优化和改进以继续提高工业同步系统的同步精度和稳定性等指标。

1. 由于文章中针对链路延时是从数学统计角度出发进行研究，所以，这意味着从时钟必须能够存储一定数量的样本数据。当然，虽然这些数据都非常小，每个样本仅仅只是存储一个时延样本值，不过，对于存储空间有限的工业交换机设备而言，这些空间仍然是宝贵的，所以，后面可以探讨如何更有效的存储样本数据，如何及时把过时样本数据清空以保证硬件层能够有能力存储这些样本值；
2. 对于时钟伺服系统，文中研究主要针对其中一个关键点，不过，要想使得从时钟系统稳定，那就必须搭建完整的时钟伺服系统，所以，如何搭建合理的时钟伺服系统以提高同步系统的稳定性和鲁棒性是一个重要的研究课题。

最后，随着分布式系统和实时性需求的不断发展，时钟同步将会越来越影响到所有系统性能的关键一环，而当前实现时钟同步最有效的便是 IEEE1588 协议，因此对该协议的研究具有很广阔和长远的研究价值。而对协议而言，越来越高的时钟同步精度和系统稳定性将是持久的研究课题。

# 参考文献


[1] 李本亮, 王厚军, 师奕兵。“基于 *PTP* 的无线分布式测试系统时钟同步研究”。电子科技大学学报，**2010**，*39*(4)：556–559。

[2] IEEE 1588. *Precision Clock Synchronization Protocol for Networked Measurement and Control Systems*. New York: Institute of Electrical and Electronics Engineers, **2008**.

[3] L. Benetazzol, C. Narduzzi and M. Stellini. “*Analysis of Clock Tracking Performances for a Software-only IEEE 1588 Implementation*”. In: *Instrumentation and Measurement Technology Conference*. Warsaw, Poland, 2007-05.

[4] 何一航。*IEEE1588* 高精度网络时钟同步研究与实现。上海，**2011**。

[5] 王金鑫。在 *KeyStone* 器件实现 *IEEE1588* 时钟方案。Texas Instruments，2013-10。

[6] 李学桥, 陈园, 梁爽。“基于 *IEEE1588* 协议的精确时钟同步算法改进”。计算机工程与科学，2011-10：42–45。

[7] S. Lv, Y. Lu and Y. Ji. “*An Enhanced IEEE 1588 Time Synchronization for Asymmetric Communication Link in Packet Transport Network*”. *IEEE Communication letters*, 2010-08: 764–766.

[8] C. Jayesh *et al.* “*Mathematical analysis & model of packet delay variation experienced by IEEE 1588 packets*”. In: *IEEE International Advance Computing Conference*, 2013-08: 347–350.

[9] 陈昕, 孙建华, 于振兴。“基于 *IEEE1588* 的无线传感器网络时钟同步方法”。系统工程与电子技术，**2014**，*36*(3)：564–574。

[10] 黄健，刘鹏，杨瑞民。“*IEEE1588* 精确时钟同步协议从时钟设计”。电子技术与应用，**2010**，(7)：94–97。

[11] 吴爽，张道农，胡啸。“*IEEE1588* 时钟同步系统中采用 *PID* 控制的可行性分析”。中国电机工程学会，**2013**：1220–1224。

[12] 邵恩。“基于 *FPGA* 的时钟同步控制系统研究与实现”。**2013**。

[13] V. Paxson and S. Floyd. “*Wide area traffic: The failure of possion modeling*”. *IEEE/ACM Translations on Metworking*, **1995**, *3*(3): 224–226.

[14] Willinger, M. S. Taqqu and R. Sherman. "*Self-similarity through high-variability: Statistical analysis of Ethernet LAN traffic at the source level*". *IEEE/ACM Translations on Networking*, **1997**, *5*(1): 71–86.

[15] W. E. Leland, S. T. Murad and W. Willinger. "*On the Self-Similar Nature of Ethernet Traffic（Extended Version）*". *IEEE/ACM Translations on Metworking*, **1994**, *2*(1): 1–15.

[16] J. Beran, R. Sherman and T. M. S. *Long-Range dependence in Variable-Bit-Rate Video traffic*. IEEE/ACM Translations on Metworking, **1995**.

[17] Q. Li and D. L. Mills. "*On the Long-Range dependence of Packet Round Trip Delays in Internet*". In: *IEEE ICC*, **1998**: 1185–1191.

[18] M. S. Borella. "*One Estimating Long Range Dependence of Network Delay*". *International Journal of Chaos Theory and Applications*, **2001**, *6*(4).

[19] 周祖德。基于网络环境的智能控制。北京：国防工业出版社，**2004**。

[20] J. X. X and X. Jing. "*Observer based learning control for a class of nonlinear systems with timevarying parametric uncertainties*". *IEEE Transactions on Automatic Control*, **2004**, *49*(2): 275–282.

[21] 王国敬，穆志纯。“基于网络控制系统平均时延的模糊控制器设计”。控制与决策，**2009**，*24*(8)：1214–1217, 1222。

[22] 王晓峰，吴平东，任长清。“基于 *TCP/IP* 网络的远程伺服控制系统中的动态模糊控制器”。计算机应用，**2002**，*22*(11)：83–85。

[23] 郑劭馨，薛薇，薛艳君。“基于改进的神经元 *PID* 的网络化控制系统”。控制工程，**2008**，*15*(3)：232–234, 260。

[24] D. Mohl and H. Weibel. "*Prestandard prototype implemantation of and end-to-end transparent clock*". In: *Proc. Conf. IEEE 1588*, 2006-10.

[25] 黎锐烽，曾祥君，李泽文。“*IEEE1588* 同步时钟网络时延误差的分析及修正”。电力系统自动化，**2012**，*36*(12)：82–87。

[26] 孙立宁，谢小辉，张峰峰。“基于神经网络的时延预测算法研究”。机器人，**2004**，*26*(3)：237–240。

[27] D. Liu, J. H. Du and Y. Zhao. "*Study on the time-delay of internet-based industry process control system*". In: *Proceedings of the 5th World Congress on Intelligent Control and Automation*. Hangzhou, **2004**: 1376–1380.

[28] C. Z. Huang, Y. Bai and X. J. Liu. "*Fuzzy PID control method for a class of networked cascade control systems*". *Computer and Automation Engineering*, **2010**: 140–144.

[29] A. Fadaei and K. Salahshoor. "*Design and implementation controller for networked control systems*". *ISA of a new fuzzy PID Transactions*, **2008**, *47*(4): 351–361.

[30] 陈鹏，戴连奎。"自适应 *Simth* 补偿器在基于 *IP* 的网络控制系统中的应用"。控制理论与应用，**2006**，*23*(1)：115–118。

[31] 任长清，吴平东，王晓峰。"基于互联网的液压远程控制系统延时预测算法研究"。北京理工大学学报，**2002**，*22*(1)：85–89。

[32] D. Srinivasagupta, H. Schattler and B. Joseph. "*Model predictive control: An algorithm for Timestamped control processes with random delays*". *International Journal of Computers and Chemical Engineering*, **2004**, *24*(8): 1337–1346.

[33] C. Iantosca, C. Heitz and H. Weibel. "*Synchronizing IEEE 1588 clocks under the presence of significant stochastic network delays*". In: *Conference on IEEE*, **2005**.

[34] 李超，徐启峰。"*IEEE1588* 协议延时不对等问题的修正"。电子测量与仪器学报，**2013**，*27*(10)：931–935。

[35] C. Iantosca, C. Heitz and H. Weibel. "*Measurement control and communication using IEEE 1588*". In: *Conference on IEEE 1588*. London: Springer-Verlag, **2006**.

[36] L. Sungwon, L. Seunggwan and H. Choongseon. "*An accuracy enhanced IEEE 1588 synchronization protocol for dynamically changing and asymmetric wireless links*". *IEEE Communication Letter*, **2012**, *16*(2): 190–192.

[37] 沈瑛，张翠芳。"基于 *BP* 神经网络的模型参考自适应控制"。西南交通大学学报，**2001**，*36*(5)：553–556。

[38] K. Lee *et al.* "*Ieee 1588-standard for a precision clock synchronization protocol for networked measurement and control systems*". In: *Conference on IEEE*, **2005**: 2.

[39] H. Weibel. "*High precision clock synchronization according to IEEE 1588 implementation and performance issues*". *Proc. Embedded World 2005*, **2005**.

[40] H. Cho *et al.* "*Precision time synchronization using IEEE 1588 for wireless sensor networks*". In: *Computational Science and Engineering, 2009. CSE'09. International Conference on*, **2009**: 579–586.

[41] D. Köhler. "*A practical implementation of an IEEE1588 supporting Ethernet switch*". In: *Precision Clock Synchronization for Measurement, Control and Communication, 2007. ISPCS 2007. IEEE International Symposium on*, **2007**: 134–137.

[42] W. Feng and S. Wenjie. "*Precise time stamping method for IEEE1588 clock synchronization message*". *Chinese Journal of Scientific Instrument*, **2009**, *30*(1): 162–169.

[43] S. Lee. "*An enhanced IEEE 1588 time synchronization algorithm for asymmetric communication link using block burst transmission*". *Communications Letters, IEEE*, **2008**, *12*(9): 687–689.

[44] M. Ouellette *et al.* "*Using IEEE 1588 and boundary clocks for clock synchronization in telecom networks*". *Communications Magazine, IEEE*, **2011**, *49*(2): 164–171.

[45] P. Ferrari *et al.* "*IEEE 1588-based synchronization system for a displacement sensor network*". *Instrumentation and Measurement, IEEE Transactions on*, **2008**, *57*(2): 254–260.

[46] T. Murakami and Y. Horiuchi. "*Improvement of synchronization accuracy in ieee 1588 using a queuing estimation method*". In: *2009 International Symposium on Precision Clock Synchronization for Measurement, Control and Communication*, **2009**.

[47] Y. Z. L. Zhi. "*Implementation and analysis of IEEE1588 precision clock protocol [J]*". *Electronic Measurement Technology*, **2009**, *4*: 019.

[48] 赵上林, 胡敏强, 窦晓波, *et al.* "基于 *IEEE1588* 的数字化变电站时钟同步技术研究 *[J]*". 电网技术, **2008**, *32*(21): 97–102.

[49] 于鹏飞 *et al.* "*IEEE 1588* 精确时间同步协议的应用方案". 电力系统自动化, **2009**, (13): 99–103.

[50] G. Ben-xuan *et al.* "*A study and analysis of high-accuracy network synchronization based on IEEE1588 [J]*". *Industrial Instrumentation & Automation*, **2006**, *4*: 20–23.

[51] A. Depari *et al.* "*Evaluation of timing characteristics of industrial ethernet networks synchronized by means of IEEE 1588*". In: *Instrumentation and Measurement Technology Conference Proceedings, 2007. IMTC 2007. IEEE*, **2007**: 1–5.

[52] 黄云水 and 冯玉光. "*IEEE1588* 精密时钟同步分析". 国外电子测量技术, **2005**, *24*(9): 9–12.

[53] X.-q. Li, Y. Chen and S. Liang. "*Improvement of Precise Time Synchronization Algorithm based on IEEE 1588*". In: *Proceeding of International Conference on Computer, Mechatronics, Control and Electronic Engineering*, **2010**: 70–73.

[54] P. Ferrari *et al.* "*Evaluation of timestamping uncertainty in a software-based IEEE1588 implementation*". In: *Instrumentation and Measurement Technology Conference (I2MTC), 2011 IEEE*, **2011**: 1–6.

[55] R. Zarick, M. Hagen and R. Bartos. "*The impact of network latency on the synchronization of real-world ieee 1588-2008 devices*". In: *Precision Clock Synchronization for Measurement Control and Communication (ISPCS), 2010 International IEEE Symposium on*, **2010**: 135–140.

[56] J. C. Eidson. *Measurement, control, and communication using IEEE 1588*. Springer Science & Business Media, **2006**.

[57] C. Lei and Z. Tianlin. "*An adaptive filtering algorithm with packet delay variation for IEEE1588 servo system*". In: *Electronic Measurement & Instruments (ICEMI), 2013 IEEE 11th International Conference on*, **2013**: 745–748.

[58] G. Giorgi and C. Narduzzi. "*Modeling and simulation analysis of PTP clock servo*". In: *Precision Clock Synchronization for Measurement, Control and Communication, 2007. ISPCS 2007. IEEE International Symposium on*, **2007**: 155–161.

[59] A. Mahmood and R. Exel. "*Servo design for improved performance in software timestamping-assisted WLAN synchronization using IEEE 1588*". In: *Emerging Technologies & Factory Automation (ETFA), 2013 IEEE 18th Conference on*, **2013**: 1–8.

[60] L. Cosart. "*Packet network timing measurement and analysis using an IEEE 1588 probe and new metrics*". In: *Precision Clock Synchronization for Measurement, Control and Communication, 2009. ISPCS 2009. International Symposium on*, **2009**: 1–6.

[61] G. B. F. D. C. Jian and J. Jianxiang. "*Study on single neuron compensation method in network synchronization [J]*". *Chinese Journal of Scientific Instrument*, **2006**, *12*: 003.

[62] H. Min *et al.* "*Timestamp based predictive robot control system over real-time industrial ethernet*". In: *Intelligent Robotics and Applications*. Springer, **2010**: 183–194.

[63] 郁振安. "基于神经网络的 *PID* 控制器". 电气自动化, **1995**, *17*(2): 8–9.

[64] 桂本烜 *et al.* "基于单神经元的网络同步补偿算法研究". 仪器仪表学报, **2007**, *27*(12): 1573–1577.

[65] 罗成汉. "基于 *MATLAB* 神经网络工具箱的 *BP* 网络实现". 计算机仿真, **2004**, *21*(5): 109–111.

[66] 高海兵 *et al.* “基于粒子群优化的神经网络训练算法研究”. 电子学报, **2004**, *32*(9): 1572–1574.

[67] 朱大奇, 史慧, *et al.* 人工神经网络原理及应用. 科学出版社, **2006**.

[68] 石慧, 王玉兰 and 翁福利. “*BP* 神经网络和模糊时间序列组合预测模型及其应用”. 计算机应用, **2011**, *31*(A02): 90–91.

[69] F.-f. LIAO and J. XIAO. “*Research on self-tuning of PID parameters based on BP Neural Networks [J]*”. *Acta Simulata Systematica Sinica*, **2005**, *7*: 047.

[70] N. Wang. “*A fuzzy PID controller for multi-model plants*”. In: *Machine Learning and Cybernetics, 2002. Proceedings. 2002 International Conference on*, **2002**: 1401–1404.

# 攻读学位期间发表的学术论文

[1] 第一作者. 统计方法在 IEEE1588 时钟同步协议中的应用. 化工自动化及仪表, 2015.07.