计算机研究与发展
Journal of Computer Research and Development
https://crad.ict.ac.cn
63(9): 2424−2441, 2026

FlyPTP: 一种跨平台易部署的 TSN 时间同步实现方法

李文超　孙寅涵　余天赐　谭良峰　马高辉　孙志刚
（国防科技大学计算机学院　长沙　410073）
（liwenchao18@nudt.edu.cn）

FlyPTP: An Implementation Method for TSN Time Synchronization with Cross-Platform and Easy Deployment

Li Wenchao, Sun Yinhan, Yu Tianci, Tan Liangfeng, Ma Gaohui, and Sun Zhigang
（*College of Computer Science and Technology*, *National University of Defense Technology*, *Changsha* 410073）

Abstract　Time sensitive networking (TSN) enhances traditional Ethernet with technologies such as high-precision time synchronization and time-aware shaping, delivering end-to-end deterministic data transmission services characterized by low latency, low jitter, and high reliability. It finds extensive applications in aerospace, industrial control, vehicle networks, and other scenarios. However, the inability of a large number of existing Ethernet end nodes to upgrade for TSN time synchronization has become a bottleneck restricting TSN deployment and application. To address this, FlyPTP method is introduced. Its core innovation decouples time synchronization from end-node architecture through a cooperative design, using lightweight software on endpoints and dedicated hardware in access switches. The theoretical validity of FlyPTP method is proven, and the hardware logic of FlyPTP and related software is developed based on FPGA. Experiments are designed to deploy FlyPTP on various heterogeneous end nodes. The results show that all tested end nodes can be deployed without modifying existing software and hardware, with a synchronization accuracy reaching 85 ns. This verifies the capability of FlyPTP to achieve TSN time synchronization for heterogeneous end nodes without altering their existing architectures. By separating time synchronization functions from the complex and diverse hardware/software environments of end nodes, FlyPTP provides a flexible and scalable solution for upgrading traditional Ethernet to TSN, overcoming the limitations of hardware dependency and kernel modification in conventional methods. This not only reduces the technical threshold for TSN deployment but also ensures compatibility and performance consistency across different platforms, laying a solid foundation for the large-scale application of TSN in industrial Internet and critical infrastructure scenarios.

Key words　time sensitive networking; time synchronization; PTP; cross platform; easily deployable

摘　要　时间敏感网络（time-sensitive networking, TSN）在传统以太网的基础上，增加高精度时间同步和时间感知整形等技术，为端到端提供低延时、低抖动和高可靠的确定性数据传输服务，在航空航天、工业控制、车载网络中具有广泛的应用场景。现有大量以太网端节点难以升级支持 TSN 时间同步，是制约 TSN 部署应用的瓶颈。为此，提出一种 TSN 时间同步方法 FlyPTP，其核心思想在于解耦时间同步功能与端节点固有架构，通过部署在端节点上的软件与接入交换机上的硬件相结合，实现平台无关的 TSN 时间同步。该方法从理论上被证明有效，并基于 FPGA 开发了相应的硬件逻辑及相关软件。通过设计实验，在多种异构端节点上部署 FlyPTP。结果表明，所有被测端节点无需修改现有软硬件即可完成部署，同步精度达到 85 ns，验证了 FlyPTP 在不修改现有架构的情况下，实现异构端节点 TSN 时间同步的能力。

收稿日期：2025-06-06；**修回日期**：2026-04-24
通信作者：孙志刚（sunzhigang@nudt.edu.cn）

关键词 时间敏感网络；时间同步；精确时间协议；跨平台；易部署

中图法分类号 TP393

DOI: 10.7544/issn1000-1239.202550459 **CSTR:** 32373.14.issn1000-1239.202550459

在当今智能化发展的浪潮中，工业控制、航空电子、汽车电子等分布式实时系统的上层应用不断丰富，基于总线的通信网络已难以满足应用对带宽资源的需求[1-2]。以太网凭借其良好的兼容性和高带宽的特点，获得了工业界的广泛关注。但传统的以太网只提供尽力而为的服务，不能满足实时系统对网络端到端低延时、低抖动以及零丢包率的确定性要求，为此，IEEE 802.1 工作组研究并制定了时间敏感网络（time sensitive networking, TSN）技术标准[3]。TSN 在传统以太网的基础上，引入高精度时间同步、时间感知整形等核心机制，旨在为 2 层网络构建端到端的确定性数据传输通路。

实现这一端到端确定性传输的前提，是网络中的所有节点，包括核心的交换机与海量的终端设备，都必须遵循统一、精确的时间基准。因此，在端侧设备实现 TSN 高精度时间同步，是打通端到端确定性传输“最后一公里”的关键，也是实现时间感知整形、帧抢占等高级 TSN 功能的基石。

实现精确时间协议（precise time protocol, PTP）[4]定义的时间同步功能，是 TSN 实现确定性传输的基础。TSN 端节点通常采用软硬件协同的方式实现高精度的 PTP 时间同步[5-8]。其中，硬件负责为同步报文标记进出端口的时间戳，软件则基于时间戳执行同步算法，生成时钟控制指令以调整本地时钟的相位与频率，最终实现全网节点的时间一致性。

然而，在推动传统以太网向 TSN 演进的过程中，一个关键挑战在于：现有存量以太网端节点因其高度异构的硬件架构与软件生态，普遍缺乏升级支持 TSN 级高精度时间同步的能力，这严重制约了 TSN 技术的规模化部署与应用。

在端节点实现 TSN 时间同步的难点，体现在 2 个方面：

1）架构耦合。在传统 TSN 时间同步的实现中，时间同步功能深度依赖于端节点操作系统内核所提供的服务，导致其与特定操作系统内核及硬件驱动紧密耦合。因此，在面向闭源系统、老旧组件或高度碎片化的软件生态时，常因无法修改内核或缺乏驱动支持，而使 TSN 时间同步的移植与适配变得异常困难。

2）部署困难。TSN 要求端节点具备在数据链路层或物理层标记硬件时间戳的能力。然而，存量端节点的网络接口硬件设计高度异构，为支持硬件时间戳功能，往往需要对特定型号的网卡进行芯片功能扩展或定制化驱动开发。这种“1 对 1”的硬件改造与适配缺乏通用性，研发成本高昂，难以在规模庞大的存量网络中实施。

针对上述挑战，现有研究工作主要遵循 2 条技术路径，但均存在局限性：

主流方法如 Linux PTP[5]，通过利用 Linux 内核的 PTP 硬件时钟框架来实现高精度同步，但其实现与 Linux 内核及特定网卡驱动深度耦合，难以移植至非 Linux 或闭源操作系统。另一类研究致力于在嵌入式场景实现轻量级 PTP[6-7]，或采用模块化设计以增强可移植性[8]。然而，这些方法通常在硬件层面依赖特定的硬件加速模块或专用 PHY 芯片，在软件层面则绑定特定的实时操作系统或协议栈，其跨平台能力依然受限，无法普适性地解决异构端节点的升级问题。

与现有研究相比，本文工作的核心创新在于提出了一种根本性的解耦思路：TSN 时间同步功能无需与端节点复杂多变的软硬件架构深度捆绑。基于此，我们提出了一种名为 FlyPTP 的 TSN 时间同步方法。其核心思想是：将时间同步的关键硬件功能从端节点内部架构中剥离出来，将其部署在端节点外侧，并通过设计统一的时间同步软硬件通信接口和通告帧，复用端节点现有的网络通路，在用户态软件与外侧硬件逻辑之间传递时间同步所需的所有信息，实现异构端节点跨平台易部署的 TSN 时间同步升级。

本文的主要贡献可以总结为 3 个方面：

1）针对异构端节点 TSN 时间同步升级的难题，本文提出了 FlyPTP 方法，其核心在于将时间同步功能从端节点固有的软硬件架构中解耦。本文设计了跨平台的统一通信接口及通告帧 *FlyMsg*，复用现有网络通路传递时间同步信息，实现了平台无关的 TSN 时间同步。此外，本文从理论上证明了该方法的有效性，表明其同步精度与端节点内置硬件方法等效。

2）为验证所提方法的可行性，本文基于 FPGA 平台完成了 FlyPTP 的原型实现。该原型系统包括轻量

化的时间同步硬件逻辑 FlyPTPH, 以及具备完整 TSN 时间同步功能的软件 FlyPTPS, 共同构成了一个完整且可跨平台部署的时间同步解决方法。

3)设计实验, 在多种异构端节点上部署 FlyPTP 原型系统。实验结果表明, 在不修改端节点现有操作系统、驱动及硬件的情况下, FlyPTP 能够达到 85 ns 的同步精度, 并可与现有 TSN 系统无缝互联, 充分验证了其实现异构端节点 TSN 时间同步的能力。

1 研究背景

1.1 TSN 节点的构成

TSN 是新一代确定性以太网技术, 其网络核心组成主要包括 TSN 交换机和 TSN 端节点两大部分。各节点依托标准化的 TSN 协议簇, 协同实现低延时、低抖动和高可靠的确定性数据传输服务。

TSN 交换机是 TSN 网络的中枢节点, 负责实现时间敏感流量的确定性调度和转发。其核心功能包括亚微秒级的时间同步、流量整形与调度和帧抢占与冗余管理。这要求 TSN 交换机能够同时支持多种 TSN 协议, 例如时间同步协议 802.1AS[9] 和流量整形 802.1Qbv[10] 等。

TSN 交换机通常采用可编程的 FPGA 或专用 ASIC 芯片实现, 以满足纳秒级的硬件时间戳标记和实时流量处理需求。相较于基于 ASIC 的固定功能方法[11], 基于 FPGA 的 TSN 交换机[12-15] 凭借 IP 核复用等技术, 能够显著缩短开发周期, 并且支持 TSN 协议的灵活扩展和更新, 是当前主流的实现方法。

TSN 端节点是生产或消费 TSN 流量的设备, 其硬件平台支持 PTP 功能(即硬件时间戳标记功能)的程度存在差异, 如图 1 所示。

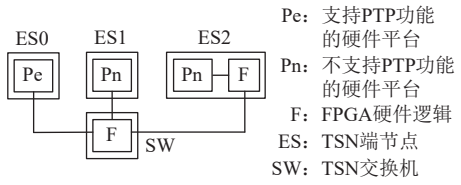


Fig. 1 Example of TSN node composition

图 1 TSN 节点构成示例

具体可划分为 3 类平台:

1)Pe 型硬件平台

Pe(PTP enable)型硬件平台是指原生支持 PTP 功能的硬件平台, 典型如搭载 Intel I210 网卡的端节点, 如图 1 中的 ES0。此类平台构成的端节点可以直接在 MAC 或 PHY 层标记硬件时间戳, 仅需增加时间同步软件即可实现 TSN 时间同步。

2)硬件逻辑固定的 Pn 型硬件平台

Pn(PTP not enable)型硬件平台是指不支持 PTP 功能的硬件平台。对于硬件逻辑固定的 Pn 型硬件平台, 其硬件逻辑不具备可编程性, 典型如采用“CPU + ASIC”架构的端节点, 如图 1 中的 ES1。由于硬件逻辑固定, 无法通过修改端节点本身使其支持硬件时间戳, 因此无法通过内部改造升级 PTP 功能, 需要在其外部增加额外的硬件逻辑才能升级 PTP 功能。

3)硬件可编程的 Pn 型硬件平台

此类硬件平台本身不支持 PTP 功能, 但其硬件逻辑具备可编程能力, 典型如“CPU + FPGA”架构的端节点, 如图 1 中的 ES2。尽管不具备原生的 PTP 能力, 但可通过扩展或修改其硬件逻辑, 例如添加 PTP 的 IP 核来升级支持硬件时间戳功能。

对于 ES2 这类具备硬件可编程能力的 Pn 型端节点而言, 采用 FPGA 实现时间同步硬件逻辑是可行且必要的技术路径。其必要性主要体现在 2 个方面: 首先, FPGA 具备快速原型开发与验证的能力, 能够适应 TSN 标准持续演进和以太网平滑过渡到 TSN 的升级需求。相较于 ASIC, FPGA 支持在硬件层面灵活部署和更新 IP 核, 便于迭代优化。其次, FPGA 可编程逻辑能够弥补 Pn 型平台原生缺失的关键硬件组件, 包括高精度硬件时钟、MAC 或 PHY 层时间戳标记模块以及对应的时钟控制寄存器接口, 从而为高精度时间戳的生成与捕获提供物理基础。

总之, TSN 网络以标准的 TSN 协议簇为基础, 通过交换机和端节点的共同协同, 构建了端到端的确定性数据传输体系。其确定性传输依赖于全网节点的时间一致性。时间同步技术则为全网提供了统一的时间基准, 是 TSN 实现确定性传输的基石。

1.2 TSN 时间同步原理

TSN 时间同步由 IEEE 802.1AS[9] 标准定义, 该标准是广义精确时间协议(generalized precision time protocol, gPTP)的一种特定配置。而 gPTP 本身, 是 IEEE 1588 精确时间协议 PTP 在 2 层局域网中的简化和优化版本。因此, 802.1AS 可被视为 PTP 协议簇中专门为 TSN 体系定制的一个子集。与更为通用的 PTP 相比, gPTP 强制使用硬件时间戳, 简化了报文类型, 并定义了适用于 TSN 的“点到点”链路延时测量机制, 从而能够满 TSN 对亚微秒级同步精度的苛刻要求。

gPTP 采用图 2 所示的主从式逐级的同步模型, 通过构建逻辑上的时间同步树, 使同步报文沿着时间同

步树传播，确保所有节点与主时钟节点（grandmaster，GM）的时钟对齐。

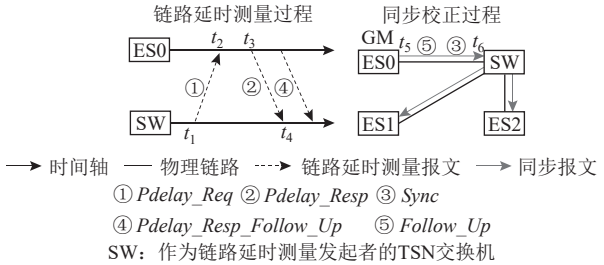


Fig. 2 TSN time synchronization principle

图 2 TSN 时间同步原理

TSN 时间同步由链路延时测量和同步校正这 2 个独立的周期性过程组成，其原理如下：

1）链路延时测量

gPTP 的链路延时测量采用“点到点”的模式，通过相邻节点之间交互链路延时测量报文，测量网络中 2 个相邻节点传输报文的延时以及 2 个节点之间的时钟频率比值。

以图 2 的链路延时测量过程为示例说明其工作原理，链路延时测量的发起者和响应者分别为 SW 和 ES0。首先，SW 发送延时测量请求报文 *Pdelay_Req*，记录发送时间戳 t_1，ES0 接收到请求报文并记录接收时间戳 t_2。

随后，ES0 发送延时测量响应报文 *Pdelay_Resp*，该报文携带有时间戳 t_2，ES0 记录其发送的时间戳 t_3，SW 接收到响应报文并记录接收时间戳 t_4，SW 通过解析 *Pdelay_Resp* 报文，获得时间戳 t_2。

最后，ES0 发送携带有 t_3 时间戳的报文 *Pdelay_Resp_Follow_Up*，SW 接收并解析出时间戳 t_3。此时，SW 便获得了全部链路延时计算所需的时间戳 t_1~t_4。SW 通过作差求平均的方法计算得到 SW 与 ES0 之间的链路延时测量值 d，计算公式如式（1）。

$$d = [(t_4 - t_1) + (t_2 - t_3)]/2 \text{。} \tag{1}$$

2）同步校正

同步校正由 TSN 中的 GM 发起，GM 周期性地发送同步报文 *Sync* 和 *Follow_Up* 至其下游的从节点，再由该节点转发同步报文至其下游节点，以此循环直至网络中所有从节点都接收到同步报文。所有从节点周期性地接收到同步报文并修正时钟。

从节点根据同步报文中携带的 GM 时间戳修正自身时钟的相位和频率。时钟的相位是指时钟的时间值；时钟频率是指时钟的快慢，即单个时钟脉冲信号增加的时间值。

以图 2 的同步校正过程为示例说明其工作原理，GM 为 ES0，其余节点均为从节点。首先，ES0 向 SW 发送同步报文 *Sync*，记录发送的时间戳 t_5，SW 接收到 *Sync* 并记录接收时间戳 t_6。随后，ES0 发送携带有 GM 基准时间戳 t_5 的报文 *Follow_Up*，SW 接收并解析 *Follow_Up*，SW 根据 GM 基准时间戳校正节点的本地时钟。至此，完成 1 轮时钟同步校正。时钟偏差校正值 *offset* 的计算如式（2）。

$$offset = (t_6 - t_5) - d \text{。} \tag{2}$$

从节点 SW 在接收到同步报文 *Sync* 和 *Follow_Up* 后，向其下游的从节点 ES1 和 ES2 进行转发。ES1 和 ES2 在接收到同步报文后，根据其中携带的 GM 时间戳修正自身时钟偏差，至此完成一轮全网所有节点的时间同步校正。

1.3 软件时间戳存在精度瓶颈

现有网络中存在大量仅支持软件时间戳的 Pn 型硬件平台构成的端节点。使用软件时间戳进行时间同步存在精度瓶颈，其时间同步精度通常在毫秒至微秒级，难以满足 TSN 亚微秒级同步的严苛要求。

软件时间戳由操作系统内核或用户态协议栈标记，其精度受多级非确定性因素制约。首先，软件时钟依赖于 CPU 时钟计数器，其频率易受电源管理和温度漂移影响。其次，通用操作系统的非实时调度机制会导致中断响应时间波动，引入软件时间戳的误差。最后，报文从网卡到达应用层需经历中断响应、直接内存访问、协议栈解析等过程，这些过程中的延迟抖动会带来软件时间戳的误差，进而影响时间同步精度。

以 Linux 操作系统为例，使用软件时间戳和开源时间同步软件 Linux PTP[5] 进行测试。结果如图 3 所示，其时间同步的时钟偏差值主要集中在 10~100 μs，

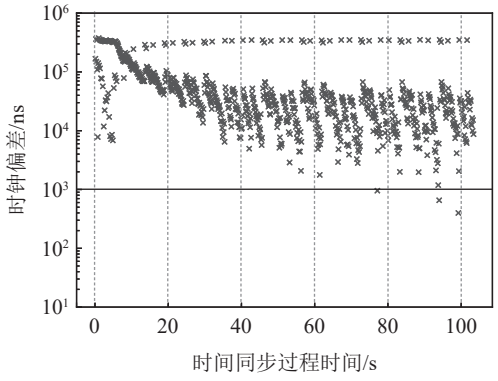


Fig. 3 The software timestamp cannot meet the TSN synchronization accuracy requirements

图 3 软件时间戳无法达到 TSN 同步精度要求

无法满足 TSN 对于亚微秒级时间同步精度的要求。由此可以看出,端节点使用软件时间戳无法满足传统以太网向 TSN 升级的要求。

1.4 存量端节点升级硬件时间戳的挑战

TSN 要求硬件时间戳在 MAC 或 PHY 层直接被捕获,以消除协议栈和中断处理引入的时间误差,进而实现高精度的时间同步。然而,现有 Pn 型端节点缺乏硬件时间戳的原生支持,需通过深度改造升级硬件时间戳功能。

对于硬件逻辑固定的 Pn 型端节点,无法通过修改端节点本身,使其支持硬件时间戳。而对于硬件可编程的 Pn 型端节点,可以改造其硬件逻辑,使其升级硬件时间戳功能。然而,升级过程面临存量端节点异构性带来的挑战,具体体现为以下 2 个层面:

1)硬件层面,需在网卡中部署硬件时钟模块,并针对不同厂商的网卡架构,定制化设计 MAC 或 PHY 层的时间戳标记逻辑。例如,Intel 网卡需通过扩展 PHY 芯片的功能实现时间戳标记[16],而 Realtek 等网卡则需改造其寄存器接口[17]。此类硬件异构性导致研发成本剧增,且难以实现通用化的部署。

2)软件层面,需修改驱动程序,以适配不同操作系统内核的中断处理机制及网络协议栈,实现时间戳从内核空间到用户空间的透传,以及对硬件时钟的控制。然而,闭源系统因无法修改内核源码,导致功能适配受限;而不同开源系统内核服务的实现存在显著差异,进一步加剧了开发和适配的难度。

综上所述,硬件平台与操作系统的异构性导致 TSN 时间同步升级需针对不同平台重复开发和适配,制约部署效率。因此,TSN 时间同步需要具备一定的跨平台部署能力,以降低规模化升级的成本和门槛。

2 FlyPTP 方法原理

本节首先构建 TSN 时间同步的通用实现模型,揭示 TSN 时间同步实现与端节点固有架构在内核层面耦合的问题,并分析出关键耦合点。然后,提出基于通告帧的 FlyPTP 实现机制,其核心思想是通过复用现有网络通路传递时间戳和时钟控制指令,从而实现时间同步功能与端节点固有架构的解耦。最后,理论证明了 FlyPTP 方法的有效性。

2.1 TSN 时间同步通用实现模型

TSN 时间同步通用实现模型是对各类 TSN 节点时间同步的功能抽象与体系化描述,揭示了各功能模块之间信息交互的关系。如图 4 所示,该模型包括 4 个功能模块和 3 个通道。

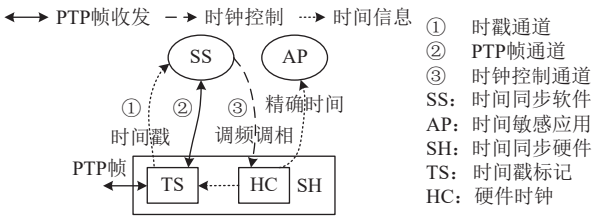


Fig. 4 Implementation model of TSN time synchronization

图 4 TSN 时间同步通用实现模型

功能模块分别是时间同步软件(synchronization software, SS)、时间敏感应用(time sensitive applications, AP)、时间戳(time stamp, TS)标记和硬件时钟(hardware clock, HC)。其中,TS 和 HC 共同构成时间同步硬件(synchronization hardware, SH)。3 个通道分别是时戳通道、PTP 帧通道和时钟控制通道。

TSN 时间同步通用实现模型各组件的介绍如下:

1)时间同步软件。时间同步软件负责 gPTP 时间同步协议的执行和计算,是时间同步实现的核心,通常在用户空间部署。时间同步软件接收时间戳和 PTP 帧,执行 gPTP 协议定义的链路延时测量和同步校正计算,根据计算结果调整硬件时钟的相位和频率,并向其他同步节点发送 PTP 帧。

2)时间敏感应用。时间敏感应用是指基于精确时间触发的应用程序,是时间同步功能的具体应用。例如基于门控的时间感知整形技术,通过硬件时钟发送的精确时间控制端节点在确定的时刻发送或接收流量,从而实现 TSN 对于流量的规划调度。

3)时间戳标记。时间戳标记负责记录 PTP 帧进出以太网口的精确硬件时间戳,并将时间戳通过时间戳通道送往时间同步软件,将 PTP 帧通过 PTP 帧通道送往时间同步软件。时间戳标记通常在以太网端口的 MAC 或 PHY 层实现。

4)硬件时钟。硬件时钟模块是实现 TSN 高精度时间同步的基础,负责维护时间信息,向时戳标记模块提供硬件时间戳,为时间敏感应用提供精确时间。TSN 时间同步要求硬件时钟具有可调节的能力,包括调节时钟的相位和频率。该模块根据时间同步软件发送的时钟控制指令,调整其相位和频率,以维护全网时钟的一致性。

5)时戳通道。时戳通道负责在时间戳标记和时间同步软件之间传递时间戳。现有时戳通道通常采用描述符回写机制实现,需要网卡驱动程序适配描述符环。PTP 帧进出网口的精确硬件时间戳由内核空间的描述符接收环和发送环传递,时间同步软件

通过开启套接字时间戳选项获取时间戳。

6）PTP 帧通道。PTP 帧通道负责在同步计算模块和时戳标记模块之间传递 PTP 帧。PTP 帧是一种特殊的以太网帧，因此 PTP 帧通道通常基于网络协议栈实现，由套接字将 PTP 帧提供给同步计算模块。

7）时钟控制通道。时钟控制通道负责向硬件时钟发送调整时钟相位和频率的指令。现有时钟控制通常采用内存映射 I/O 机制实现，需要网卡驱动程序适配内核空间的内存映射服务。时间同步软件调用内存映射服务，通过直接读写寄存器的机制，实现调频和调相参数的下发。

由上述分析可知，时戳通道和时钟控制通道的实现依赖于网络协议栈、描述符环、内存映射 I/O 等操作系统内核服务，是 TSN 时间同步功能与端节点固有架构紧密耦合的关键因素。

该耦合问题正是导致 1.4 节所述宏观挑战的根本原因。首先，由于时戳与时钟控制通道深度依赖于特定操作系统的内核服务，将其移植至异构平台时，必须重写驱动并适配内核，导致了“架构耦合”的挑战。其次，这种与底层平台的强绑定关系，使得时间同步功能无法独立于端节点软硬件环境而存在，必须为不同平台进行重复且定制化的开发，从而引发了大规模“部署困难”的问题。因此，破解 TSN 时间同步升级困境的关键在于如何将这 2 个通道从端节点的固有架构中解耦出来。

2.2 FlyPTP 实现机制

针对 TSN 时间同步实现过程中，时戳通道和时钟控制通道与端节点固有架构紧耦合的问题，本文提出基于通告帧的时间同步实现机制。FlyPTP 通过通告帧封装时间戳与时钟控制指令，并复用端节点现有网络通路，实现 FlyPTP 软硬件间的信息交互，从而将时间同步功能从端节点固有架构中解耦。

通告帧是一种特殊的以太网帧，其中携带的信息有 PTP 帧、时间戳和时钟控制指令。 当时间同步硬件收发 PTP 帧时，将收发的 PTP 帧封装为通告帧，再由通告帧携带时间戳送至时间同步软件，由此实现时戳通道。 时间同步软件进行同步计算后，发送通告帧至时间同步硬件，由通告帧携带调频调相的时钟控制指令，由此实现时钟控制。

对于 Pe 型硬件平台的端节点时间同步，如图 5 的 ES0，其时间同步硬件部署在硬件架构内部，时间戳和时钟控制指令的传递需要依赖于操作系统内核服务。Pe 型端节点的时间同步实现与其平台架构本身紧耦合，难以进行跨平台的功能移植。

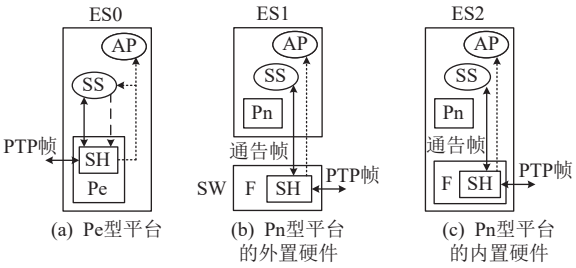


Fig. 5 TSN time synchronization implementation comparison

图 5 TSN 时间同步实现对比

对于 Pn 型硬件平台的端节点，其平台本身缺乏高精度时间同步能力，需要增加时间同步硬件，并实现时戳通道和时钟控制通道，才能实现时间同步功能。FlyPTP 方法将时间同步硬件部署在 Pn 型硬件平台外侧，并由时间同步硬件将收到的 PTP 帧封装为通告帧，通过通告帧实现时间戳和时钟控制指令在软硬件之间的传递。

根据 Pn 型硬件平台可编程能力的不同，时间同步硬件的部署位置也有所区别。对于硬件逻辑固定的端节点，如图 5 的 ES1，可将时间同步硬件部署在近端侧的 TSN 交换机上。时间同步硬件对网络侧收发 PTP 帧，对端侧收发通告帧。对于硬件可编程的端节点，如图 5 的 ES2，可将时间同步硬件部署在端节点上，对外收发 PTP 帧，对内收发通告帧。

FlyPTP 方法中的通告帧在时间同步软硬件之间的传递利用现有网络通路，支持时间戳和时钟控制指令在端节点中的透传。因此，基于通告帧实现的时戳通道和时钟控制无需修改现有端节点架构，通过将时间同步软硬件部署在用户空间和端节点硬件架构外侧，实现时间同步功能与端节点固有架构的解耦。

2.3 FlyPTP 有效性证明

FlyPTP 实现 Pn 型端节点时间同步的精度，与传统 Pe 型端节点的一致。具体表现为二者的同步精度仅与时间戳标记误差相关，与时间同步硬件的部署位置无关。

引理 1. 时钟偏差校正值的误差仅与时间戳标记误差相关。定理表达式如式（3）。

$$\Delta(offset) = \epsilon_{ts} - \epsilon'_{ts}, \tag{3}$$

其中 $\Delta(offset)$ 为时钟偏差校正值的误差，ϵ_{ts} 为 GM 节点发送同步报文时时间戳标记的误差，ϵ'_{ts} 为从节点接收同步报文时时间戳标记的误差。

证明.

首先，考虑误差的来源。对于 Pe 型端节点，时间同步过程中的误差来源于时间戳标记的误差。而对

于 Pn 型端节点，还需要考虑时间戳标记后，时间戳由链路传递至时间同步软件的链路延时抖动。

因此，我们假设 Pe 型端节点的时间同步硬件模块时间戳标记的误差为 ϵ_{hw}；Pn 型端节点上 FPGA 实现的时间同步硬件标记时间戳误差为 ϵ_{f}。Pn 型端节点的链路传输延时抖动为 Δ_{link}。Pe 型端节点的时间戳为 $t_i^*+\epsilon_{\mathrm{hw}}$，其中 t_i^* 为真实物理时间，Pn 型端节点的时间戳为 $t_i^*+\epsilon_{\mathrm{f}}$。

由链路延时测量的式(1)可得，引入误差后的 Pe 型端节点的链路延时为

$$d_{\mathrm{Pe}}=[(t_4^*+\epsilon_{\mathrm{hw0}})-(t_1^*+\epsilon_{\mathrm{hw0}})+(t_2^*+\epsilon_{\mathrm{hw1}})-(t_3^*+\epsilon_{\mathrm{hw1}})]/2\,, \tag{4}$$

$$d_{\mathrm{Pe}}=[(t_4^*-t_1^*)+(t_2^*-t_3^*)]/2=d^*\,。 \tag{5}$$

引入误差后，Pn 型端节点的链路延时为

$$d_{\mathrm{Pn}}=[(t_4^*+\epsilon_{\mathrm{f0}}+\Delta_{\mathrm{link}})-(t_1^*+\epsilon_{\mathrm{f0}})+(t_2^*+\epsilon_{\mathrm{f1}}+\Delta_{\mathrm{link}})-(t_3^*+\epsilon_{\mathrm{f1}})]/2\,, \tag{6}$$

$$d_{\mathrm{Pn}}=[(t_4^*-t_1^*)+(t_2^*-t_3^*)]/2+\Delta_{\mathrm{link}}=d^*+\Delta_{\mathrm{link}}\,。 \tag{7}$$

由时钟偏差校正值的式(2)可得，引入误差后 Pe 型端节点的时钟偏差校正值为

$$offset_{\mathrm{Pe}}=[(t_6^*+\epsilon_{\mathrm{hw0}})-(t_5^*+\epsilon_{\mathrm{hw1}})]-d_{\mathrm{Pe}}\,, \tag{8}$$

$$offset_{\mathrm{Pe}}=(t_6^*-t_5^*)-d^*+(\epsilon_{\mathrm{hw0}}-\epsilon_{\mathrm{hw1}})\,。 \tag{9}$$

引入误差后 Pn 型端节点的时钟偏差校正值为

$$offset_{\mathrm{Pn}}=[(t_6^*+\epsilon_{\mathrm{hw0}}+\Delta_{\mathrm{link}})-(t_5^*+\epsilon_{\mathrm{hw1}})]-d_{\mathrm{Pn}}\,, \tag{10}$$

$$offset_{\mathrm{Pn}}=(t_6^*-t_5^*)-d^*+(\epsilon_{\mathrm{f0}}-\epsilon_{\mathrm{f1}})\,。 \tag{11}$$

由式(9)和式(11)可知，*offset* 的误差仅来源于时间戳的标记误差 ϵ。 证毕.

需要指出的是，此证明所关注的时间戳标记误差 ϵ 是一个综合性的误差项，其已涵盖了由时钟晶振频率稳定度等设备自身因素所引入的时钟源误差。这是因为时间戳本质是本地硬件时钟的瞬时读数，任何影响时钟准确度的器件误差，最终都会体现在时间戳的数值上。对于程序运行而言，时间同步程序的核心职能是执行 gPTP 协议规定的同步计算。其可能引入的额外误差主要源于计算过程中的数值精度，如浮点数运算与舍入误差。然而，这部分误差仅与所选数据类型的表示精度相关，在不同计算平台上具有一致性，且其量级通常为皮秒或飞秒级，远小于时间戳标记的纳秒级误差。因此，程序运行误差对 Pn 型与 Pe 型之间同步的等效性比较不构成实质影响。

由引理 1 可知，FlyPTP 方法将时间同步硬件从端节点固有架构中剥离并部署至 FPGA，并不影响时间同步的精度，由此可得：

定理 1. FlyPTP 实现的 Pn 型端节点时间同步，与 Pe 型端节点时间同步具有等效性。

引理 1 的证明确立了 FlyPTP 方法的核心有效性。定理 1 表明，通过 FlyPTP 架构实现的 Pn 型端节点，在时间同步精度上可以达到与原生 Pe 型端节点同等的水平。这一结论从理论上确保了本文方法不因解耦架构而牺牲性能。这意味着，只要通过 FlyPTP 为 Pn 型端节点赋予了时间同步能力，那么该节点在同步性能上即可被视为一个标准的 TSN 端节点。因此，定理 1 不仅证明了同步精度的等效性，更从根本上论证了 FlyPTP 方法解决“架构耦合”与“部署困难”这一核心问题的有效性与正确性。

至此，FlyPTP 方法的有效性得证。

3 FlyPTP 总体设计

3.1 FlyPTP 设计概述

针对现有存量异构端节点在升级 TSN 时间同步时面临的“架构耦合”与“部署困难”两大挑战，本文提出了一种跨平台易部署的 TSN 时间同步方法 FlyPTP。本节将从设计理念、方法特点及与经典方法的对比 3 个方面，对 FlyPTP 进行整体性介绍。

FlyPTP 的设计理念是“功能解耦”与“路径复用”。该方法将时间同步功能从端节点复杂的固有架构(包括操作系统内核、驱动程序及硬件平台)中剥离出来，通过部署在用户空间的时间同步软件 FlyPTPS 与部署在端节点外侧或近端交换机上的轻量化硬件逻辑 FlyPTPH 相互协作，并复用端节点现有的网络数据通路传递时间戳与时钟控制指令，从而实现了时间同步功能与目标平台的高度解耦。

基于这一理念，FlyPTP 呈现出以下具体特点：

1)跨平台兼容性。通过定义统一的、平台无关的通告帧 *FlyMsg* 及软硬件接口，FlyPTP 屏蔽了底层 CPU 架构、操作系统和网卡芯片的差异，实现了“一次开发，多处部署”。

2)易部署性。FlyPTPH 采用轻量级硬件设计，资源开销小，可灵活集成于 FPGA 实现的端节点或接入交换机。对于端节点而言，无需修改其任何既有硬件、驱动程序或操作系统内核，即可通过以太网口等通用接口，以“即插即用”的方式获得 TSN 时间同步能力。

3)高精度同步。尽管硬件模块外置，但通过理论证明与精心设计，FlyPTP 能够为原本不支持硬件时间戳的 Pn 型端节点提供亚微秒的同步精度，完全

满足 TSN 对于时间同步的严格要求。

4）无缝互联。FlyPTP 的处理流程完全遵循标准的 gPTP 协议，对外通信标准的 PTP 帧，确保了其能够与现有 TSN 网络中的标准设备无缝互联与协同工作。

与 Linux PTP[5] 等经典方法相比，FlyPTP 的优势在于其跨平台的能力。传统方法深度依赖于特定的 PTP 功能网卡和操作系统内核服务（如描述符环、内存映射 I/O 等），导致其在异构、闭源或老旧平台上的移植和部署成本高昂。而 FlyPTP 通过架构层面的创新，从根本上避免了对端节点内核及驱动的修改，将复杂的适配工作收敛于自身统一的软硬件接口之内。这不仅显著降低了 TSN 升级的技术门槛和成本，更为实现传统以太网中海量异构端节点向 TSN 的平滑演进提供了一条通用、可行的技术路径。

在成本效益方面，FlyPTP 方法同样具备显著优势。诚然，为端节点附加外部硬件逻辑会引入一定的改造成本，但相较于传统升级路径，FlyPTP 实现了一种总体拥有成本更优的解决方法。首先，传统方法不仅需要升级或更换支持 PTP 的专用网卡硬件，其主要的成本在于为解决架构耦合问题而产生的、针对不同操作系统内核和网卡驱动的重复性软件适配开发。相比之下，FlyPTP 通过架构解耦，将硬件成本收敛于 2 种低成本场景：对于硬件可编程的端节点，FlyPTPH 可作为 IP 核直接集成，无需额外硬件；对于硬件固定的端节点，轻量化的 FlyPTPH 模块可集成于近端接入交换机，复用交换机中已有的 FPGA 资源，避免了为每个端节点独立增加硬件的成本。5.4 节的资源开销评估表明，FlyPTPH 所需的逻辑与存储资源极少，验证了其集成的低成本特性。更重要的是，在软件层面，FlyPTP 将传统方法中深度的内核修改工作，简化为仅需根据平台实现统一的通告帧 *FlyMsg* 收发接口，极大地降低了跨平台部署的软件开发和维护成本。因此，FlyPTP 在有效解决时间同步技术挑战的同时，也提供了一条经济可行的规模化部署路径。

3.2 总体架构

FlyPTP 时间同步总体架构如图 6 所示，分为 3 个部分：时间同步软件 FlyPTPS、时间同步硬件逻辑 FlyPTPH 和时间同步软硬件通信接口。FlyPTP 软硬件之间通过通告帧 *FlyMsg* 通信。

FlyPTPS 是 FlyPTP 的软件部分，是实现 gPTP 协议时间同步计算的核心，主要组成模块包括链路延时测量模块和同步校正计算模块。同步校正计算模块收发 PTP 同步帧，并下发调整硬件时钟频率和相位的时钟控制指令。链路延时测量模块收发 PTP 链路延时测量帧，并向同步校正计算提供链路延时测量值。此外，软件还提供了管理接口，负责配置下发和状态上报。

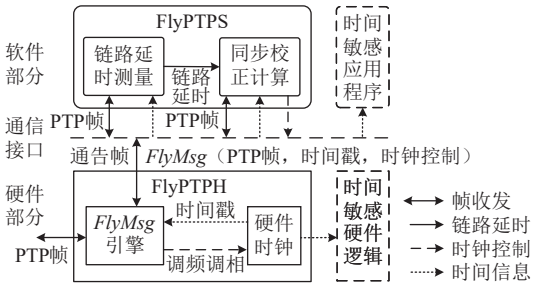


Fig. 6 FlyPTP time synchronization architecture

图 6 FlyPTP 时间同步架构

FlyPTPH 是 FlyPTP 的硬件部分，是实现 TSN 时间同步跨平台易部署的基础，主要组成模块包括 *FlyMsg* 引擎和硬件时钟。*FlyMsg* 引擎负责解析软件下发 *FlyMsg* 中的时钟控制指令，并将网络侧收发的 PTP 帧封装为 *FlyMsg*，其中携带有 PTP 帧进出该模块的时间戳。硬件时钟为 *FlyMsg* 引擎提供时间戳，并且根据时钟控制指令调整时钟的相位和频率。

FlyPTP 软硬件通信接口是实现跨平台兼容性的关键。该接口负责软件层面的 *FlyMsg* 解析和封装，为时间同步软件和时间敏感应用程序提供控制硬件时钟的接口调用，屏蔽了底层平台的差异，保障了跨平台行为的一致性。

此外，*FlyMsg* 是不出 FlyPTP 节点内部的软硬件通信帧，FlyPTP 对外收发标准的 PTP 帧。因此，FlyPTP 支持与现有 TSN 时间同步系统的无缝互联。

3.3 跨平台兼容性设计

FlyPTP 的跨平台兼容性依赖于通告帧 *FlyMsg* 驱动的时间同步功能与端节点固有架构的解耦，并通过统一的时间同步软硬件通信接口，实现异构端节点的 TSN 时间同步。本节将分别介绍 *FlyMsg* 帧和时间同步软硬件通信接口的设计。

1）*FlyMsg* 帧设计

FlyMsg 是 FlyPTP 软硬件通信的通告帧，携带有 PTP 帧、时间戳和时钟控制指令，实现了以下目标：

①端节点架构解耦式时间同步。时间同步软件 FlyPTPS 部署在用户空间，时间同步硬件逻辑 FlyPTPH 作为端节点硬件平台架构外侧部署的硬件逻辑，二者通过 *FlyMsg* 交互时间同步流程所需的信息，无需修改端节点现有的软硬件，即可实现端节点的 TSN 时间同步功能。

②复用现有网络通路。*FlyMsg* 作为 FlyPTP 的内部通信帧，通过端节点现有的以太网帧通路传递至用户空间。无需修改现有驱动程序和网络协议栈，不同平台仅需提供收发帧的函数，即可为用户空间的时间同步软件或时间敏感应用程序提供时间信息。

FlyMsg 有以下 2 种功能：一是传递时间同步过程中必要的信息，包括时间戳、时钟控制指令和 PTP 帧；二是为上层应用提供高精度的时间通告，例如，时敏应用程序需要精确的定时触发行为，FlyPTPH 可以通过发送 *FlyMsg* 为其提供精确的时间信息。

FlyMsg 帧作为软硬件间的唯一信息载体，其设计保证了跨平台行为的一致性。*FlyMsg* 具有固定的帧格式，由首部与载荷构成。其携带的 3 种成分，PTP 帧、时间戳和时钟控制指令，均以平台无关的字节序列形式封装在帧内。PTP 帧为标准格式；时间戳为统一的纳秒级整型数值；时钟控制指令则为对抽象寄存器地址与数据的操作码。无论底层硬件平台是何种架构，*FlyMsg* 帧的结构与内容解析逻辑完全一致，消除了因平台异构性带来的数据表示差异。

2)时间同步软硬件通信接口设计

时间同步软硬件通信接口旨在为上层应用提供统一的时钟控制接口和 PTP 帧透传接口，同时完成 *FlyMsg* 帧的封装、解析与平台适配。该接口可以有效地屏蔽平台差异性，并且简化时间同步应用的编程。

在不同软硬件平台应用 FlyPTP 时，首先需要进行平台适配，开发人员根据平台提供的以太网帧收发接口，完成 *FlyMsg* 的收发接口函数，实现 *FlyMsg* 从时间同步硬件逻辑到用户空间应用的传递。在此基础上，时钟控制接口和 PTP 透传接口通过调用 *FlyMsg* 收发接口函数，实现平台差异性的屏蔽，为上层提供统一的接口调用。

时钟控制接口为上层应用提供了对硬件时钟的状态感知与动态调整能力，是应用获取时间戳并控制时钟的桥梁。如表 1 所示，该接口通过向 FlyPTPH 发送读/写请求 *FlyMsg* 帧，为用户提供了控制与调整硬件时钟的 API 函数，例如获取时间、设置频率等。

PTP 帧透传接口实现了 FlyPTP 与标准 TSN 设备的协议级兼容性。如表 2 所示，该接口向用户提供收发 PTP 帧的 API 函数。上层应用通过该接口收发 PTP 帧，确保系统可无缝接入现有 TSN 网络。该接口接收来自 FlyPTPH 的 *FlyMsg* 封装帧，解析出其中的时间戳信息和 PTP 帧提供给上层应用；上层应用调用接口发送 PTP 帧，接口会将 PTP 信息封装为 *FlyMsg* 封装帧发送给 FlyPTPH。该接口通过封装和解析 *FlyMsg* 帧，实现了 PTP 帧在网络和应用程序之间的透传。在跨平台部署时，开发者仅需根据目标平台的网络编程接口，实现最底层的 *FlyMsg* 收发函数。在此之上，表 1 与表 2 所列的所有高阶接口功能均由 FlyPTP 软件库[18]统一实现，无需针对不同平台进行 2 次开发。

Table 1 Clock Control Interface Function List

表 1 时钟控制接口函数列表

API 函数名	功能描述	函数参数
get_clock_time( )	获取硬件时钟时间	时钟相位寄存器地址
get_clock_freq( )	获取硬件时钟频率	时钟频率寄存器地址
get_clock_offset( )	获取硬件时钟偏差	相位补偿寄存器地址
adj_clock_time( )	调整硬件时钟偏差	相位补偿寄存器地址；时钟偏差值
set_clock_time( )	设置硬件时钟时间	时钟相位寄存器地址；时钟相位
set_clock_freq( )	设置硬件时钟频率	时钟频率寄存器地址；时钟频率

Table 2 PTP Frame Transparent Interface Function List

表 2 PTP 帧透传接口函数列表

API 函数名	功能描述	函数参数
recv_ptp( )	等待接收并解析 PTP 帧	携带 PTP 帧的 *FlyMsg* 帧
send_sync( )	构造并发送 *Sync* 帧	携带 *Sync* 的 *FlyMsg* 帧
send_follow_up( )	构造并发送 *Follow_Up* 帧	携带 *Follow_Up* 的 *FlyMsg* 帧
send_pdelay_req( )	构造并发送 *Pdelay_Req* 帧	携带 *Pdelay_Req* 的 *FlyMsg* 帧
send_pdelay_resp( )	构造并发送 *Pdelay_Resp* 帧	携带 *Pdelay_Resp* 的 *FlyMsg* 帧

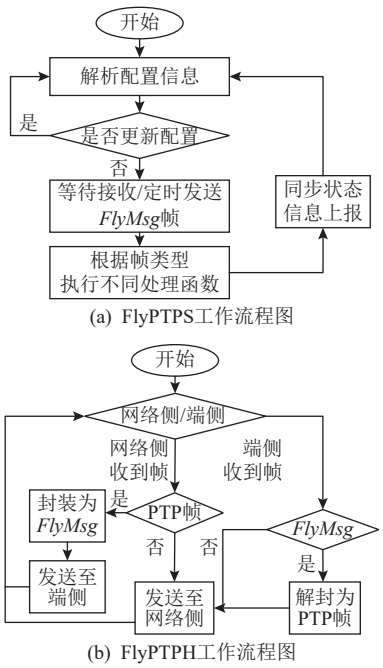
综上所述，FlyPTP 的跨平台兼容性是通过“接口标准化”与“适配局部化”来实现的。*FlyMsg* 帧作为统一的信息载体，定义了软硬件之间与平台无关的通信协议；而时间同步软硬件通信接口则通过抽象的 API 和可移植的适配函数，将底层异构平台在操作系统、网络栈及硬件驱动上的差异进行封装与屏蔽。具体而言，任何支持基础网络通信的平台，只需在其用户空间实现标准的 *FlyMsg* 帧收发函数，即可将平台上层的 FlyPTPS 与底层的 FlyPTPH 无缝连接，而 FlyPTP 的核心同步逻辑无需任何更改。这种设计确保了 FlyPTP 能够以一致的行为在不同的异构端节点上运行，从而实现跨平台兼容。

3.4 工作流程设计

FlyPTP 时间同步工作流程由 2 部分组成：一是软件 FlyPTPS 的工作流程，二是硬件 FlyPTPH 的工作流程。本节将分别介绍 FlyPTP 软硬件的工作流程。

FlyPTPS 工作流程如图 7(a)和算法 1 所示。软件在每轮同步逻辑运行结束后检查是否更新配置，

若更新，则按照新的配置运行，因此 FlyPTPS 支持动态配置。FlyPTPH 工作流程如图 7(b)和算法 2 所示，主要完成端侧和网络侧以太网帧的分类，实现 *FlyMsg* 帧的封装和解析。



(a) FlyPTPS工作流程图

(b) FlyPTPH工作流程图

Fig. 7 Work flow chart of FlyPTP

图 7 FlyPTP 工作流程图

FlyPTPS 接收到 *FlyMsg* 帧后，解析 *FlyMsg* 帧中携带的信息，其中包括有时间同步计算所需的时间戳信息，以及 gPTP 协议规定交互的 PTP 帧。软件根据 *FlyMsg* 子类型的不同来执行不同的处理函数，例如同步帧解析和链路延时计算等。在每轮同步处理完成后，上报同步状态和链路延时测量状态。在 gPTP 时间同步计算的工作流程中，软件根据定时器信号定时发送 *Sync* 和 *Pdelay_Req* 帧，其中 *Sync* 只在配置为 GM 节点时定时发送。

FlyPTPH 具体分为 2 部分流程：一是以太网帧从网络侧到端侧的工作流程，二是以太网帧从端侧到网络侧的工作流程。

1)以太网帧从网络侧到端侧的流程。以太网帧由端侧 MAC 送入 FlyPTPH 后，首先，将以太网帧分类为 PTP 帧和非 PTP 帧。其中，非 PTP 帧被送至网络侧，PTP 帧被送至 *FlyMsg* 引擎。而后，硬件时钟模块记录 PTP 进入该模块时的时间戳，将时间戳和 PTP 帧封装在 *FlyMsg* 中送往端侧，至此，完成了以太网帧从网络侧到端侧的工作流程。

算法 1. FlyPTPS 工作流程。

输入：配置参数 *FlyConfig*，新配置 *NewFlyConfig*；

输出：同步状态参数 *SyncState*，链路延时测量状态参数 *LinkDelayState*。

① 初始化 *NewFlyConfig*←None, *SyncState*←None, *LinkDelayState*←None, *Sync* 发送定时器 *TimerSync*( )，*Pdelay_Req* 发送定时器 *Timer-Pdelay*( )；
② while 配置成功 do
③ if (*FlyPTPConfig* ≠ *NewFlyPTPConfig*) then
④ *FlyPTPConfig* ← *NewFlyPTPConfig*;
⑤ end if
⑥ if (*TimerSync*( )=1 & *FlyConfig.gm* =1) then
⑦ *FlyMsg_send*(*Sync*)；
⑧ end if
⑨ if (*TimerPdelay*( )= 1) then
⑩ *FlyMsg_send*(*Pdelay_Req*)；
⑪ end if
⑫ if (*FlyMsg_recv*( )= 1) then
⑬ *handler*(*FlyMsg*)；
⑭ end if
⑮ *report*(*SyncState*, *LinkDelayState*)；
⑯ end while

2)以太网帧从端侧到网络侧的流程。由上层应用发送的 *FlyMsg* 经端侧送至 FlyPTPH，首先经过 *FlyMsg* 引擎，引擎解析出 *FlyMsg* 中携带的时钟控制信息，并将其发往硬件时钟，校正时钟的相位和频率。再者，对于上层应用需要知道发送时间戳的 PTP 帧，如 *Sync* 和 *Pdelay_Req*，FlyPTPH 会重新封装一个 *FlyMsg*，携带 PTP 帧发出模块的时间戳送回端侧。最终，*FlyMsg* 引擎将解析出的 PTP 帧和非 PTP 帧一同发往网络侧。至此，完成了太网帧从端侧到网络侧的工作流程。

算法 2. FlyPTPH 工作流程。

输入：网络侧以太网帧 *msg_all*，端侧 *FlyMsg* 帧；

输出：网络侧以太网帧 *msg_all*，端侧 *FlyMsg* 帧。

① while 配置成功 do
② if (*mac_msg_recv*( )= 1) then
③ if (*ptp_recv*( )= 1) then
④ *send_FlyMsg*(*End_side*)；
⑤ end if
⑥ if (*ptp_recv*( )= 0) then
⑦ *send_msg*(*Net_side*)；
⑧ end if
⑨ end if
⑩ if (*end_msg_recv*( )= 1) then
⑪ if (*FlyMsg_recv*( )= 1) then

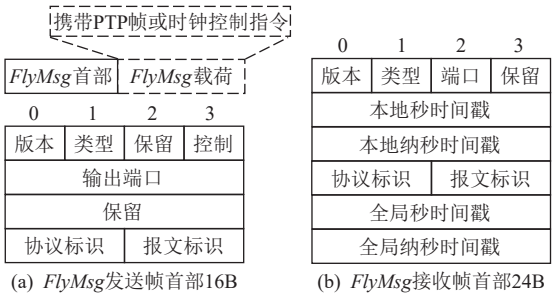
```
⑫        send_ptp(Net_side);
⑬      end if
⑭      if (FlyMsg_recv( ) = 0) then
⑮        send_msg(Net_side);
⑯      end if
⑰    end if
⑱ end while
```

4 FlyPTP 的原型实现

为验证 FlyPTP 方法实现端节点 TSN 时间同步的可行性及其在真实 TSN 系统中的同步性能，本文搭建了 FlyPTP 时间同步原型系统。该系统包含 3 个核心内容：*FlyMsg* 通告帧的详细格式、基于 FlyPTP 实现的时间同步软件 FlyPTPS，以及 FPGA 平台上实现的时间同步硬件逻辑 FlyPTPH。本节将分别介绍上述 3 个方面的设计与实现。

4.1 *FlyMsg* 帧格式

FlyMsg 帧格式如图 8 所示，由 16B 或 24B 首部和最长 1518B 的载荷组成。该载荷长度与标准以太网帧的最大传输单元 1518B 保持一致，其设计目的是确保 *FlyMsg* 能够完整地封装并透传一个最大尺寸的标准以太网帧。由于 PTP 帧本身是一种特殊的以太网帧，其长度不会超过此限制，因此该设计保证了任何合法的 PTP 帧均可被 *FlyMsg* 无损承载。



(a) *FlyMsg*发送帧首部16B (b) *FlyMsg*接收帧首部24B

Fig. 8 The format of the time announce frame *FlyMsg*

图 8 时间通告帧 *FlyMsg* 格式

FlyMsg 帧首部带有版本信息、子类型信息、硬件时间戳和输入输出端口信息等。*FlyMsg* 帧载荷带有 PTP 信息或时钟控制指令，其中，PTP 信息为标准 PTP 帧格式[4]，时钟控制指令为读写寄存器的地址和读写数据信息。

根据 *FlyMsg* 首部类型字段的不同，可将 *FlyMsg* 分为 *FlyMsg* 发送帧(16B 首部)和 *FlyMsg* 接收帧(24B 首部)。*FlyMsg* 发送帧是指从 FlyPTPS 发往 FlyPTPH 的帧；*FlyMsg* 接收帧是指从 FlyPTPH 发往 FlyPTPS 的帧。*FlyMsg* 发送帧分为读请求发送、写请求发送和封装 PTP 发送 3 个子类型；*FlyMsg* 接收帧分为读数据接收、响应接收和封装 PTP 接收 3 个子类型。各子类型 *FlyMsg* 帧的功能如下。

1)读请求发送帧：FlyPTPS 请求读取 FlyPTPH 硬件时钟寄存器中的数据。

2)写请求发送帧：FlyPTPS 请求改写 FlyPTPH 硬件时钟寄存器中的数据。

3)封装 PTP 发送帧：FlyPTPS 将 PTP 信息封装为 *FlyMsg* 发送帧的形式发送给 FlyPTPH。

4)读数据接收帧：FlyPTPH 回复 FlyPTPS 的硬件时钟寄存器数据。

5)响应接收帧：FlyPTPH 回复 FlyPTPS 的携带有时间戳的 *FlyMsg* 帧。

6)封装 PTP 接收帧：FlyPTPH 将接收到的 PTP 帧封装为接收帧的形式发送给 FlyPTPS。

4.2 FlyPTP 时间同步软件实现

本文基于 C++语言开发了时间同步软件 FlyPTPS，其基础功能代码约 2 710 行，软件架构如图 9 所示。

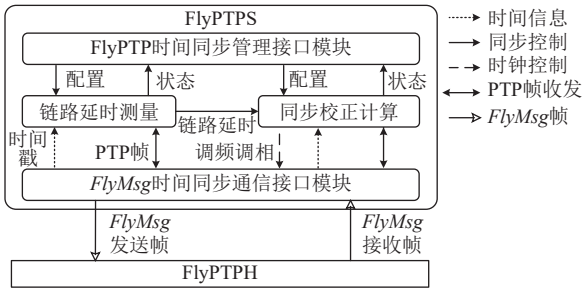


Fig. 9 FlyPTPS software module implementation

图 9 FlyPTPS 软件模块实现

FlyPTPS 核心是 gPTP 协议的实现，其主体由 2 个功能模块构成：

1)同步校正模块，作为软件核心，负责执行主从时钟同步。该模块通过 *FlyMsg* 通信接口收发 *Sync* 与 *Follow_Up* 帧，并获取精确的硬件时间戳，据此计算时钟偏差与频率比值。最终，通过调用 *FlyMsg* 通信接口，向 FlyPTPH 下发调频与调相指令，实现本地时钟与主时钟的闭环同步。

2)链路延时测量模块，为同步校正提供关键网络参数。该模块通过 *FlyMsg* 通信接口交互 *Pdelay_Req*、*Pdelay_Resp* 等报文，获取链路延时测量所需的 t_1~t_4 时间戳，并根据式(1)，计算相邻节点间的链路传播延时。

此外，FlyPTP 时间同步管理接口模块为上层应用提供配置与监控能力。用户可通过 *FlyConfig* 结构体

动态设置同步周期、主从角色等参数。软件运行期间，同步状态（如时钟偏差、GM 频率比）与链路延时测量状态等结果，将分别存入 *SyncState* 与 *LinkDelayState* 结构体，并通过该接口上报，支撑上层实现同步监测、故障诊断等管理功能。

FlyPTPS 通过底层的 *FlyMsg* 时间同步通信接口模块与 FlyPTPH 硬件交互。该模块将上述核心模块产生的 PTP 帧信息和时钟控制指令封装成标准的 *FlyMsg* 发送帧；同时，对接收到的 *FlyMsg* 接收帧进行解析，提取出时间戳与 PTP 帧信息传递给相应模块。此设计确保了软硬件间通信的平台无关性。

4.3 FlyPTP 硬件逻辑实现

FlyPTPH 是部署在 FPGA 上的轻量化时间同步硬件逻辑，负责产生高精度时间戳并与软件通信，其模块架构如图 10 所示。

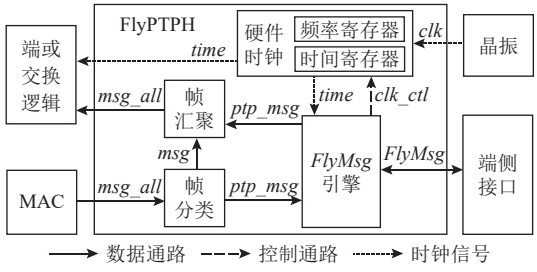


Fig. 10 FlyPTPH hardware module implementation

图 10 FlyPTPH 硬件模块实现

FlyPTPH 包含 4 个核心模块：

1）硬件时钟模块，该模块是系统的时间基准，由可读写的时间与频率寄存器构成。它接收板载晶振的脉冲信号进行计时，并可依据软件下发的时钟控制指令 *clk_ctl* 动态调整时钟的相位与频率。

2）*FlyMsg* 引擎，作为通信枢纽，负责 *FlyMsg* 帧的封装与解析。当 PTP 帧通过时，引擎会记录其进出的精确硬件时间戳并封装至 *FlyMsg* 帧头；同时，解析来自软件的 *FlyMsg* 帧，提取其中的时钟控制指令。

3）帧分类与帧汇聚模块，二者协同处理网络数据。帧分类模块识别输入帧，将 PTP 帧导向 *FlyMsg* 引擎，非 PTP 帧则直接旁路。帧汇聚模块接收来自 *FlyMsg* 引擎的 PTP 帧与旁路的非 PTP 帧，将其合并后发送至网络设备逻辑，确保普通数据通行无阻。

在端节点或交换机中集成 FlyPTPH 时，需将其串联在 MAC 与网络设备逻辑（如网卡驱动或交换芯片）之间。上层应用通过以太网或 PCIe（peripheral component interconnect express）等通用接口与 FlyPTPH 交互 *FlyMsg* 帧。这种部署方式无需改动端节点原有硬件架构，实现了即插即用的时间同步能力。

5 性能评估

为验证 FlyPTP 实现异构端节点 TSN 时间同步的能力，本文搭建了测试验证环境。通过为原本不支持 TSN 时间同步的端节点部署 FlyPTPH 与 FlyPTPS，测试其新增时间同步功能后的 3 项关键指标：同步精度、与现有 TSN 系统的无缝互联能力，以及软硬件资源开销。实验结果验证了 FlyPTP 实现跨平台和易部署时间同步的能力。

5.1 实验设置

本文搭建 2 节点的网络拓扑，测试验证 FlyPTP 实现端节点时间同步的能力。实验环境如图 11 所示，由 2 台计算机和 1 块 FlyPTPH 硬件模块构成，并通过以太网接口连接。其中，端节点 1 由一台不支持 PTP 硬件时间戳的计算机与 FlyPTPH 硬件模块通过以太网直连组成，用于运行时间同步软件 FlyPTPS；另一台支持 PTP 硬件时间戳的端节点 2 运行 LinuxPTP。FlyPTPH 硬件逻辑基于 FPGA 模块实现。

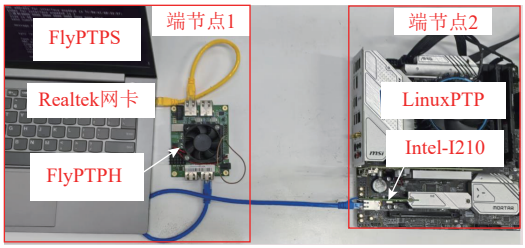


Fig. 11 FlyPTP end node time synchronization experimental environment

图 11 FlyPTP 端节点时间同步实验环境

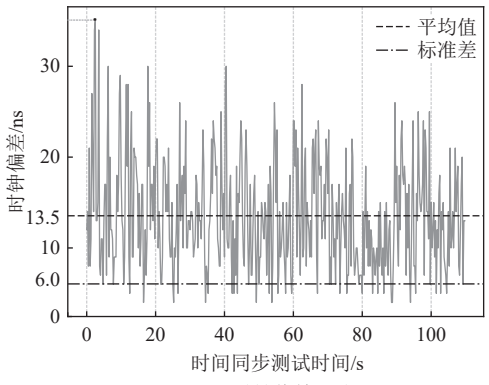
端节点 1 的 CPU 型号为 Intel core i7-1165G7，网卡型号为 Realtek-RTL8111，该网卡不支持 PTP 硬件时间戳，因此，端节点 1 在连接 FlyPTPH 之前不支持 TSN 时间同步。端节点 2 的 CPU 型号为 Intel core i7-12700F，网卡型号为 Intel-I210，该网卡支持 PTP 硬件时间戳，因此，端节点 2 是现有的支持 TSN 时间同步的端节点。端节点 1 和端点 2 均运行在 Linux 系统下。实验中 FlyPTPH 硬件逻辑使用 FPGA 实现，板卡型号为 Arria10-10AX048H，使用的 PHY 芯片型号为 VSC8504，板载晶振标称频飘率为 50 ppm。

5.2 同步性能

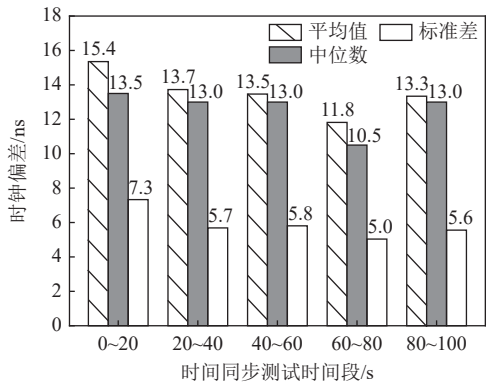
首先，测试原本不支持 TSN 时间同步的 Pn 型端节点 1，在采用 FlyPTP 方法赋予其时间同步能力后，

与原生支持 TSN 时间同步的 Pe 型端节点 2 之间相互进行时间同步的性能。同步的周期为 125 ms，用时钟偏差绝对值来评估。时钟偏差是指同一时刻在 2 个时钟示数差值的绝对值，同步精度是指观测时间内，2 个时钟偏差的最大值[19]。

测试结果如图 12 所示，图 12(a)展示端节点使用 FlyPTP 后，在 TSN 标准的 125 ms 同步周期下，稳定同步的一段时间内，时钟偏差值的观测情况。使用 FlyPTP 方法后，FlyPTPH 为时间同步软件提供了高精度的硬件时间戳，时钟偏差值稳定在 35 ns 以下，达到了 TSN 亚微秒级的同步精度要求。实验结果证明了 FlyPTP 可以实现端节点的 TSN 时间同步，并且能够与现有 TSN 时间同步系统无缝互联。



(a) 时钟偏差测试



(b) 时钟偏差统计

Fig. 12 TSN time synchronization test results of FlyPTP

图 12 FlyPTP 的 TSN 时间同步测试结果

观察不同测试时间段内时钟偏差值的统计结果，如图 12(b)所示，以直方图的形式展示时钟偏差的平均值、中位数和标准差。结果显示，在最初的 0~20 s 内，时钟偏差的平均值和标准差都最大，随着同步时间的增加，平均值和标准差均有所下降，平均值维持在 13.5 ns 左右，标准差维持在 5.5 ns 左右。实验结果证明了 FlyPTP 在 TSN 标准同步周期下，同步精度具有良好的收敛性和稳定性。

为了验证 FlyPTP 在多跳网络下的表现，我们设计了多点同步实验。本文在标准的 125 ms 周期下，向端节点 1 和端节点 2 之间增加多跳 TSN 交换机，而后测试 2 个端节点之间的时钟偏差值。测试结果如图 13 所示。由测试结果可知，随着同步节点跳数的增加，同步精度有所下降，但在 7 跳的端到端链式拓扑中，FlyPTP 实现的最大时间同步误差为 83 ns。该结果完全满足并远超 TSN 标准中广受认可的“7 跳 1 μs”的同步要求，证明了 FlyPTP 在规模化网络中的有效性。

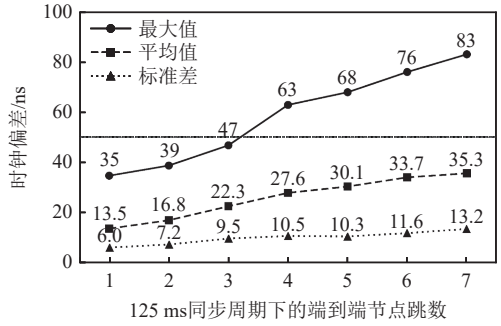


Fig. 13 Test results of FlyPTP under multi-hop nodes

图 13 FlyPTP 在多跳节点下的测试结果

然后，为了进一步验证 FlyPTP 端节点时间同步的性能，设置了 8 组实验，记录不同同步周期下时间同步的时钟偏差值。8 组实验的同步周期分别设置为 125 ms, 250 ms, 375 ms, 500 ms, 625 ms, 750 ms, 875 ms, 1s。每一组实验数据均是稳定运行 30 min 以上采集的数据，能够反映系统长时间运行的一般性能。

不同同步周期实验的结果如图 14 所示，由概率密度图展示不同同步周期下时间同步一段时间内时钟偏差绝对值的概率分布。概率密度图是一种用于展示实验数据概率分布的可视化工具，其峰值位置反映了数据取值较为集中的区域，宽窄则可以判断数据的离散程度。使用概率密度图可以反映一段时间内时间同步的稳定性以及时钟偏差主要集中的区域。

总体而言，各子图均呈现单峰的概率密度分布形态，表明在不同同步周期下，时钟偏差绝对值具有一个相对集中的取值范围。随着同步周期从 125 ms 延长至 1 s，分布曲线的峰值逐渐降低且向右平移，偏差值的最大值增加，数据分布更分散，说明同步周期越大，同步精度越低、同步稳定性越差。

在 TSN 标准的 125 ms 同步周期下，时钟偏差值的平均值为 13.5 ns，同步精度为 35 ns，时钟偏差值集中在 8~17 ns，标准差为 6.0。当同步周期从 125 ms 提高至 1 s 的过程中，时钟偏差的平均值和最大值均升

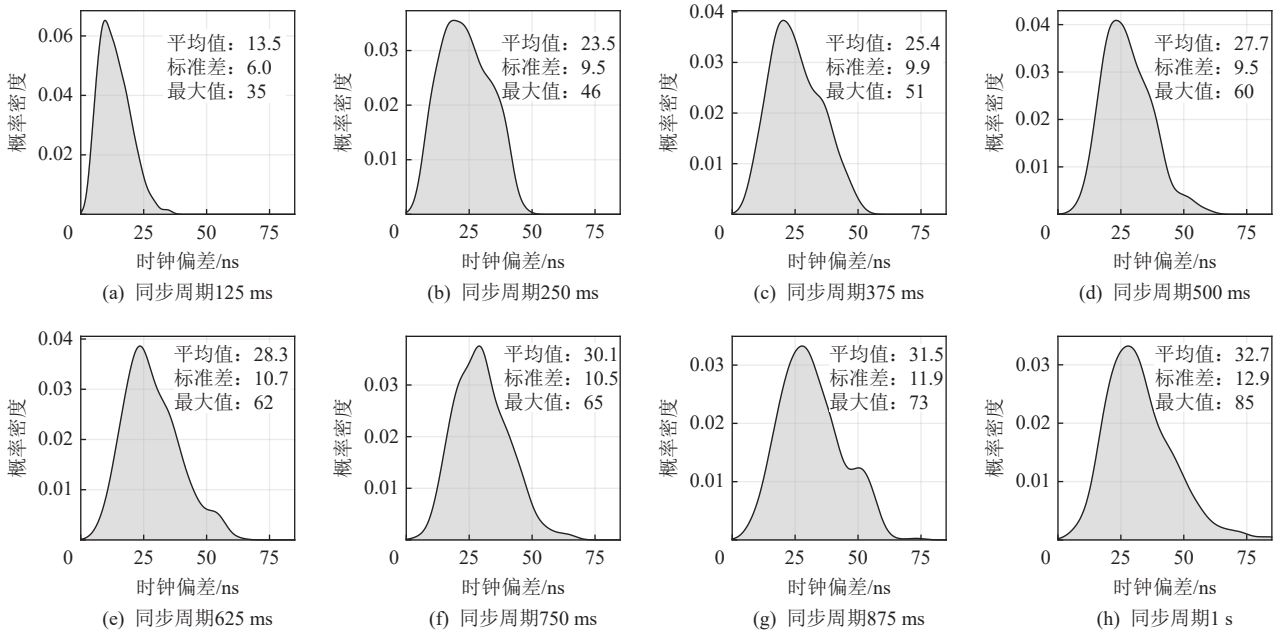


Fig. 14 Test results of FlyPTP at different time synchronization periods

图 14 FlyPTP 在不同时间同步周期的测试结果

高，标准差呈现升高的趋势。

当同步周期提升到 1 s 时，时钟偏差的平均值升高到 32.7 ns，时钟偏差值主要集中在 20~40 ns，同步精度升高至 85 ns。即便如此，其同步精度仍处于亚微秒级的范围之内。该结果说明 FlyPTP 在需要较低同步频率的场景应用时，依然能够满足 TSN 时间同步的精度要求。

5.3 跨平台测试

跨平台测试的核心目的在于验证 FlyPTP 为不具备原生 PTP 能力的异构端节点升级时间同步功能的有效性。如表 3 所示，实验所采用的 4 个端系统，其网卡均不支持硬件时间戳功能，属于 1.1 节所定义的 Pn 型硬件平台。因此，在部署 FlyPTP 之前，这些端节点均无法满足 TSN 同步要求。

Table 3 Heterogeneous End System Equipment Information

表 3 异构端系统设备信息

端系统标号	CPU 型号	网卡型号
端系统 1	Intel core i7-1165G7	Realtek-RTL8111
端系统 2	AMD Ryzen9-7940HX	Realtek-PCIe-GbE
端系统 3	Intel core i5-13500H	Intel-I219-V
端系统 4	Phytium D2000/8 E8C	LRES1050PF-SFP

在本实验中，端系统 1 与端系统 2 的部署模式对应于图 5 中的 ES1 类型，即通过近端侧的 FlyPTPH 硬件模块为其提供时间同步能力。作为对比，实验中作为同步对象的端节点 2 则使用了支持硬件时间戳的 Intel I210 网卡，其本身即为 Pe 型端节点，对应图 5 中的 ES0，并运行标准的 LinuxPTP 软件，用于模拟现有 TSN 网络中的标准节点。针对这 4 个端系统的实验，其时间同步周期均设置为 125 ms，并记录了稳定运行 30 min 以上的时钟偏差值。

为具体说明 FlyPTP 的易部署性与跨平台优势，我们记录了在不同端系统上的部署过程。FlyPTP 的核心软件 FlyPTPS 本身是平台无关的。部署时，唯一需要根据目标平台操作系统进行适配的，是 3.3 节中所述的 *FlyMsg* 收发函数的实现。该函数模块负责通过操作系统提供的原始套接字接口收发原始的 *FlyMsg* 以太网帧，其总代码量仅占 FlyPTPS 总代码量的 6.94%。在本实验涉及的同一 Linux 操作系统下，此部分代码可完全复用，无需任何修改。即使未来需要移植至其他操作系统，也仅需修改 *FlyMsg* 收发函数。相较于传统的需要为每种网卡开发专属内核驱动、深度修改协议栈的方法，FlyPTP 方法的移植适配工作量更少。

需要进一步说明的是，完整的 FlyPTP 方法包含时间同步软件 FlyPTPS 与时间同步硬件 FlyPTPH。其跨平台易部署性，不仅体现在软件的接口抽象上，也同样得益于硬件 FlyPTPH 的可复用设计。尽管在不同硬件平台（如不同类型的端节点或交换机）上部署时，FlyPTPH 的外围接口需根据目标硬件环境进行适配，但其核心时间同步逻辑是固定的，可作为标准

IP 核复用。在实际集成过程中,开发人员只需围绕该核心 IP 核,针对目标平台的网络通信接口和时钟接口进行连接,即可实现 FlyPTPH 的快速集成,具体接口见 4.3 节。这种“核心逻辑固化、接口灵活适配”的硬件设计模式,结合软件层的统一接口,共同构成了 FlyPTP 跨异构平台易部署的关键。

FlyPTP 实现异构端系统时间同步的测试结果如图 15 所示。小提琴图的上横线和中间横线分别标注时钟偏差的最大值(同步精度)和平均值。

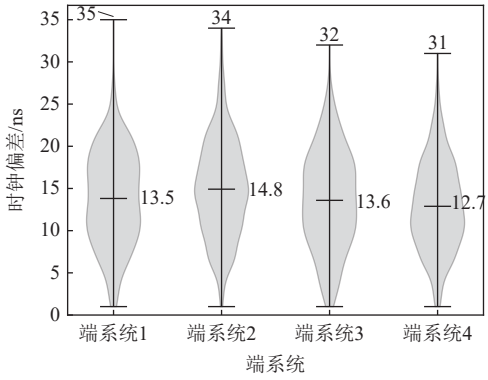


Fig. 15 Comparative chart of time synchronization of heterogeneous end systems implemented by FlyPTP

图 15 FlyPTP 实现异构端系统时间同步对比图

由小提琴图可以看出,不同端系统实现时间同步的精度相似,均维持在 35 ns 以下;数据分布主要集中段均在 10~20 ns。实验结果表明,尽管这些端系统硬件架构各异,但它们均实现了高度一致的亚微秒级同步精度。这充分证明了 FlyPTP 方法在实现跨平台兼容的同时,能够以极低的软硬件改造代价,将异构端节点无缝接入 TSN 网络。

5.4 资源使用

为进一步验证 FlyPTP 易部署的能力,对 FlyPTP 硬件和软件资源的开销进行统计。FlyPTPH 原型的资源开销如表 4 所示,其中,该原型的逻辑资源是查找表(look up tables, LUT),存储资源是块随机存取存储器(block random access memory, BRAM)。 FlyPTPH 原型总共使用了 7 604 个 LUT 和 44.81KB 的 BRAM,分别不超过实验所用 FPGA 总量的 4.14% 和 1.25%。

能满足 FlyPTPH 资源开销的最小低端 FPGA 型号为 Cyclone 10 系列的 10CL010YU256C8G,其单芯片成本低至 15 美元。资源开销数据证明了 FlyPTP 轻量化的硬件设计在实现时尽可能地节省了资源开销,有利于开发人员将 FlyPTPH 集成在硬件逻辑可编程的端节点或近端侧的 TSN 交换机上,并且能够满足低成本部署的需求。这一方面降低了单个节点的升级成本,另一方面也确保了在集成 FlyPTPH 时,不会过度占用节点原有的可编程逻辑资源。

Table 4 Resource Overhead of FlyPTPH

表 4 FlyPTPH 资源开销

类型	用量/KB	用量占总量的比例/%
LUT	7 604	4.14
BRAM	44.81	1.25

时间同步软件 FlyPTPS 在 Linux 操作系统下,Ubuntu 20.04 环境中运行,同步周期设置为 125 ms。在同等条件下运行 Linux PTP,并且无其他应用程序干扰的情况下,统计并对比二者运行时系统的 CPU 利用率,结果如表 5 所示。统计数据显示,Linux PTP 因深度依赖内核服务以实现高精度,导致其内核空间利用率显著高于 FlyPTPS。与之相对,FlyPTPS 将绝大多数操作置于用户空间完成,仅占用少量系统内核的网络数据传输和计算结果文件读写服务,其架构主动规避了对操作系统内核的依赖。CPU 利用率的对比印证了其在架构上的解耦优势,这种将同步功能主体移出内核的设计,是 FlyPTP 易于跨平台部署的关键所在。

Table 5 CPU Utilization Comparison of Time Synchronization Software

表 5 时间同步软件 CPU 利用率对比 %

时间同步软件名称	用户空间利用率	内核空间利用率	内核空间占比
FlyPTPS	0.53	0.01	1.8
Linux PTP	0.41	0.43	51.2

此外,本文进一步统计了 FlyPTP 原型系统运行过程中网络资源的使用情况。时间同步软件 FlyPTPS 运行时,每秒发送数据包 24 个,接收数据包 16 个,收发速率分别为 1.38 KBps 和 1.86 KBps。实验数据说明,FlyPTP 时间同步仅使用极少的网络资源,方便在网络资源受限的节点上部署。

6 相关工作

为应对 TSN 时间同步升级中存量端节点硬件架构与操作系统异构带来的挑战,现有研究致力于从软件架构优化、硬件设计创新及系统级解决方法等多个层面开发跨平台、高精度的实现方法。本文以主流方法 Linux PTP 的局限性为切入点,分析现有方法的实现机制并评估其跨平台能力,并进一步总结和分析了近期的研究进展。

Linux PTP 是当前 TSN 端节点时间同步的主流方法，其通过用户态守护进程调用内核 PTP 硬件时钟框架实现高精度同步[20]。PTP 硬件时钟框架提供统一接口控制硬件时钟的相位和频率，并支持从网卡驱动获取 MAC/PHY 层硬件时间戳。例如，Intel I210 网卡通过驱动注册时钟设备，Linux PTP 通过时钟系统调用读取时间戳[5]。然而，Linux PTP 与 Linux 内核高度耦合，导致其难以向非 Linux 平台移植，具体体现在：

1）内核依赖。需目标网卡驱动实现 PTP 硬件时钟接口，包括时间戳回传、时钟调节等回调函数。非 Linux 系统缺乏等效框架，移植需重构驱动逻辑。

2）协议栈绑定。Linux PTP 依赖 Linux 网络栈的时间戳选项捕获收发队列描述符中的时间戳，在非 Linux 系统移植过程中，需修改协议栈中断处理逻辑以支持时间戳透传。

3）实时性约束。Linux 内核的调度延迟和中断响应抖动影响同步精度，需通过补丁改造内核为硬实时系统，进一步增加其向非 Linux 系统适配的复杂度。

为降低跨平台适配成本，研究者提出了多种面向特定硬件平台的轻量化时间同步方法。Lao 等人[6]在嵌入式系统中实现了基于 FreeRTOS 的 PTP 守护进程 PTPd，利用 STM32 的 IEEE 1588 硬件加速模块将同步精度提升至 100 ns 以下。然而，该方法在硬件层面依赖特定 MCU 的硬件加速模块，需修改网络驱动以支持时间戳回传；在软件层面绑定 FreeRTOS 任务调度和 LwIP 协议栈，移植至其他操作系统需重写内核中断处理、协议解析和时钟控制接口，跨平台能力受限。András 等人[7]通过抽象硬件操作为适配层，实现了 FreeRTOS 微控制器主从模式 PTP 来实现 flexPTP，支持在 TM4C1294 和 STM32F7/H7 间移植。但其跨平台能力仍受限于 FreeRTOS 生态，硬件适配层需针对不同 MCU 定制开发。Pietro 等人[8]设计了模块化的 PTP 实现 PPSi，采用协议核心与运行环境解耦的架构。其通过定义函数接口抽象硬件时钟和网络操作，使协议状态机可跨平台复用，跨平台能力较强。然而，PPSi 仍需 White Rabbit 等专用 PHY 芯片提供底层时间戳支持，且时钟伺服算法与硬件强相关。

当前研究在硬件时间戳精度提升和芯片级集成方面取得了显著进展。Shao 等人[21]针对 PTP 与 PHY 时钟域不同带来的时间戳采样误差问题，提出了一种名为精细计数锁相环的重同步电路。该设计通过外部高精度相位测量单元来估计和补偿时钟域交叉引入的量化误差，将传统方法中 1 个 PHY 时钟周期的误差大幅降低。该方法在 FPGA 原型系统中展示了通过专用电路设计优化底层时间戳精度的有效性。在芯片设计方面，Chen 等人[22]和 Lan[23]等人专注于为工业物联网设计高集成度的 PTP 同步芯片。Chen 等人[22]的工作在 45 nm 工艺下设计了一款位于数据链路层的 PTP 协议芯片，通过补偿时钟偏移和移除过大环回延时等优化机制，实现了 434 MHz 的工作频率和 26 ns 的同步精度。Lan 等人[23]提出了一种带有动态频率补偿的 PTP 辅助时间戳硬件，并在 45 nm 工艺下实现了 370 MHz 的工作频率和 10.8 ns 的同步分辨率。这些 ASIC 设计方法旨在通过硬件集成减少协议栈延迟，提供稳定且低功耗的高精度同步能力，但其通常与特定的网络接口绑定，仍面临与端节点平台集成的挑战。

在系统架构层面，研究者也开始探索超越单个节点或局域网的大规模、高精度时间同步网络。Han 等人[24]提出了一种创新的国家级高精度时间同步系统架构。该架构在骨干网中采用分布式共视时间传递技术，在城域网中则采用基于时分多路复用的 PTP 时间传递。通过数据中心协调各区域时间服务器，该方法在跨越 2 150 km、3 个省份/城市的现场试验中，实现了端到端 36 ns 的同步精度。这项工作展示了复杂网络环境下实现广域高精度同步的可行性，为 TSN 在未来大规模工业互联网中的应用提供了系统级的参考。

如表 6 所示，现有跨平台实现方法需对目标系统的软硬件固有架构进行深度修改，此类修改导致高昂的适配成本，且对闭源系统或老旧固件难以实施。

Table 6 Cross-Platform Capability Comparison of TSN Time Synchronization Methods

表 6 TSN 时间同步方法跨平台能力对比

TSN 时间同步方法	跨平台的硬件依赖	跨平台的内核修改
Linux PTP[5]	依赖 PTP 功能网卡硬件支持	修改操作系统网络协议栈和实时内核
嵌入式 PTPd[6]	依赖 STM32 的 1588 硬件加速模块	修改 FreeRTOS 任务调度和 LwIP 驱动
flexPTP[7]	依赖 MCU 硬件架构无法扩展至 x86 平台	适配 LwIP 协议栈和 FreeRTOS 调度策略
PPSi[8]	依赖 White Rabbit 等专用硬件	需实现平台专用网络和时间操作接口
PTP ASIC[22-23]	依赖 GMII 等特定接口与 MAC/PHY 连接	与平台绑定，通常需要定制驱动
FlyPTP（本文）	依赖于轻量化的硬件逻辑 FlyPTPH	无需修改任何内核服务或驱动程序

通过对近期相关工作的梳理可以发现，TSN/PTP 时间同步的研究呈现出以下趋势和焦点：

1）硬件优化持续深入。研究不再满足于简单的

硬件时间戳支持,而是深入到时钟架构内部,通过锁相环等精密电路设计来克服时钟域偏差等固有误差源,追求皮秒到纳秒级的极致精度。

2)专用化与集成化。针对工业物联网等特定场景,研究者越来越多地采用 FPGA 固化逻辑或 ASIC[22-23]的方式来设计时间同步功能,以在保证高性能的同时,获得更低的功耗、更小的体积和确定的延迟。

3)跨平台与易部署需求凸显。尽管硬件优化和专用设计成果显著,但如 Linux PTP、嵌入式 PTP 等方法的跨平台能力不足问题依然存在。如何让高精度时间同步能力无缝适配现有大量异构的、非 TSN 能力的端节点,成为一个关键的工程挑战。

本文的 FlyPTP 方法将时间同步功能从目标系统的软硬件架构中解耦,在硬件平台外侧部署时间同步硬件逻辑,无需修改任何系统内核服务或驱动程序,易于在不同类型的平台间大规模的部署。

7 结束语

本文针对传统以太网升级 TSN 过程中,实现异构端节点 TSN 时间同步架构耦合、部署困难的问题,提出了一种 TSN 时间同步方法 FlyPTP。该方法通过解耦时间同步功能与端节点固有架构,实现了跨平台易部署的 TSN 时间同步功能。设计了跨平台兼容的通告帧 *FlyMsg* 以及平台无关的时间同步软硬件通信接口,为 FlyPTP 方法跨异构平台部署提供了支持。

然后,本文从理论上证明了 FlyPTP 方法的有效性。设计并实现了 TSN 时间同步软件 FlyPTPS 以及轻量化的时间同步硬件逻辑 FlyPTPH,搭建了 FlyPTP 原型系统。

最后,设计实验,测试了 FlyPTP 实现时间同步跨平台和易部署的能力。结果表明,使用 FlyPTP 方法无需修改现有端节点的软硬件架构,能够跨平台实现异构端节点的时间同步;与现有 TSN 时间同步系统的同步精度保持在 85 ns,达到了 TSN 对于时间同步精度的要求,并且能够稳定地无缝互联。实验证明了 FlyPTP 的能力。

下一步工作将基于 FlyPTP,进一步研究 TSN 时间同步管理控制的方法和应用,实现 TSN 多域容错时间同步的方法,以及 FlyPTP 时间同步与其他 TSN 关键功能的协同集成机制。

作者贡献声明:李文超负责方法整体设计并撰写论文;孙寅涵负责实验设计;余天赐负责硬件部分的实现;谭良峰、马高辉负责软件部分的实现;孙志刚提出指导意见并修改论文。

参 考 文 献

[1] Yan Jinli, Quan Wei, Jiang Xuyan, et al. Injection time planning: Making CQF practical in time-sensitive networking[C]//Proc of the 38th IEEE Int Conf on Computer Communications. Piscataway, NJ: IEEE, 2020: 616−625

[2] Jiang Xuyan, Quan Wei, Fu Wenwen, et al. OpenPlanner: An open-source traffic planning for time-sensitive networking[J]. Journal of Computer Research and Development, 2025, 62(5): 1307−1329 (in Chinese)
(姜旭艳, 全巍, 付文文, 等. OPenPlanner: 一个开源的时间敏感网络流量规划器[J]. 计算机研究与发展, 2025, 62(5): 1307−1329)

[3] Yang Xiangrui, Yan Jinli, Chen Bo, et al. FlexTSN: A flexible TSN switch implementation model[J]. Journal of Computer Research and Development, 2021, 58(1): 153−163 (in Chinese)
(杨翔瑞, 严锦立, 陈波, 等. FlexTSN: 一种灵活的 TSN 交换实现模型[J]. 计算机研究与发展, 2021, 58(1): 153−163)

[4] IEEE Standards Association. IEEE Std 1588: IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems[S]. Los Alamitos, CA: IEEE Computer Society, 2019

[5] Network Time Foundation. The Linux PTP project[EB/OL]. [2025-05-12]. http://linuxptp.sourceforge.net

[6] Lao Kaiyao, Yan Gangfeng. Implementation and analysis of IEEE 1588 PTP daemon based on embedded system[C]//Proc of the 39th Chinese Control Conf. Piscataway, NJ: IEEE, 2020: 4377−4382

[7] András W, Tamás K. Portable, PTP-based clock synchronization implementation for microcontroller-based systems and its performance evaluation[C]//Proc of the 14th IEEE Int Symp on Precision Clock Synchronization for Measurement, Control, and Communication. Piscataway, NJ: IEEE, 2021: 1−6

[8] Pietro F, Maciej L, Alessandro R, et al. PPSi − A free software PTP implementation[C]//Proc of the 8th IEEE Int Symp on Precision Clock Synchronization for Measurement, Control, and Communication. Piscataway, NJ: IEEE, 2014: 71−76

[9] IEEE Standards Association. IEEE Std 802.1AS−2020: IEEE Standard for Local and Metropolitan Area Networks—Timing and Synchronization for Time-Sensitive Applications[S]. Los Alamitos, CA: IEEE Computer Society, 2020

[10] IEEE Standards Association. IEEE Std 802.1Qbv−2015: IEEE Standard for Local and Metropolitan Area Networks–Bridges and Bridged Networks-Amendment 25: Enhancements for Scheduled Traffic[S]. Los Alamitos, CA: IEEE Computer Society, 2016

[11] Fu Wenwen, Liu Rulin, Quan Wei, et al. nPSA: A low-latency, deterministic switching architecture for TSN chips[J]. Journal of Computer Research and Development, 2023, 60(6): 1322−1336 (in

Chinese)
(付文文, 刘汝霖, 全巍, 等. nPSA: 一种面向 TSN 芯片的低延时确定性交换架构[J]. 计算机研究与发展, 2023, 60(6): 1322−1336)
[12] Quan Wei, Yang Xiangrui, Sun Zhigang, et al. Synchronized network model and its application for TSN[J]. Computer Engineering and Design, 2021, 42(4): 914−919 (in Chinese)
(全巍, 杨翔瑞, 孙志刚, 等. 面向 TSN 的同步网络模型及应用[J]. 计算机工程与设计, 2021, 42(4): 914−919)
[13] Zhou Zifan, Michael S, Yan Ying. Mapping TSN traffic scheduling and shaping to FPGA-based architecture[J]. IEEE Access, 2020, 8: 221503−221512
[14] Akram B, Takahiro H, Takaaki F. Hardware design and evaluation of an FPGA-based network switch supporting asynchronous traffic shaping for time sensitive networking[J]. IEEE Access, 2024, 12: 123149−123165
[15] Quan Wei, Fu Wenwen, Sun Zhigang, et al. HX-DS09: A customized low power time sensitive networking chip for high-end equipment[J]. Journal of Computer Research and Development, 2021, 58(6): 1242−1245 (in Chinese)
(全巍, 付文文, 孙志刚, 等. 枫林一号: 一款面向高端装备定制的低功耗时间敏感网络芯片[J]. 计算机研究与发展, 2021, 58(6): 1242−1245)
[16] Syed S, Wan A, Aeizaal A. Time-aware traffic shaper using time-based packet scheduling on Intel I210[J]. International Journal of Research and Engineering, 2018, 5(9): 2348−7852
[17] Xu Hengrui, Sun Chentong. Design and implementation of an AUTOSAR-based in-vehicle ethernet communication driver[C]//Proc of the 5th Int Conf on Electronic Information Engineering and Computer Technology. Piscataway, NJ: IEEE, 2025: 721−725
[18] Li Wenchao. FlyPTP[EB/OL]. [2025-06-03]. https://gitee.com/wen-chao-li2/fly-ptp
[19] Sun Yinhan, Sun Zhigang, Fu Wenwen, et al. Virtual Scoping: Lightweight and scalable TSN time synchronization accuracy measurement method[J]. Journal on Communications, 2024, 45(9): 1−13 (in Chinese)
(孙寅涵, 孙志刚, 付文文, 等. Virtual Scoping: 轻量级可扩展的 TSN 时间同步精度测量方法[J]. 通信学报, 2024, 45(9): 1−13)
[20] Richard C, Cristian M. Design and implementation of a PTP clock infrastructure for the Linux kernel[C]//Proc of the 4th IEEE Int Symp on Precision Clock Synchronization for Measurement, Control, and Communication. Piscataway, NJ: IEEE, 2010: 116−121
[21] Shao Chengjia, Xu Zhen, Harihara S. A novel resynchronization circuit to enable high-accuracy PTP timestamping[J]. IEEE Transactions on Instrumentation and Measurement, 2024, 73: 1−8
[22] Chen Yeeshao, Chu Yuansun, Lo Dawei, et al. Chip design of PTP time synchronization for industrial IoT[J]. IEEE Access, 2025, 13: 21966−21979
[23] Lan Yankai, Chen Yeeshao, Hou Tingchao, et al. Development board implementation and chip design of IEEE 1588 clock synchronization system applied to computer networking[J]. Electronics, 2023, 12(10): 2166−2185
[24] Han Liuyan, Lu Rongduo, Li Jiachen, et al. Novel national high-precision time synchronization system for mobile communication[J]. IEEE Access, 2025, 13: 18878−18889



Li Wenchao, born in 2000. PhD candidate. His main research interests include time-sensitive networking and time synchronization.
李文超, 2000 年生。博士研究生。主要研究方向为时间敏感网络、时间同步。



Sun Yinhan, born in 1997. PhD. His main research interests include time-sensitive networking and fault-tolerant time synchronization.
孙寅涵, 1997 年生。博士。主要研究方向为时间敏感网络、容错时间同步。



Yu Tianci, born in 2001. PhD candidate. His main research interests include the architecture of time-sensitive networking and FPGA design.
余天赐, 2001 年生。博士研究生。主要研究方向为时间敏感网络架构、FPGA 设计。



Tan Liangfeng, born in 2002. PhD candidate. His main research interests include time-sensitive networking and fault-tolerant time synchronization.
谭良峰, 2002 年生。博士研究生。主要研究方向为时间敏感网络、容错时间同步。



Ma Gaohui, born in 2002. PhD candidate. His main research interests include time-sensitive networking and Byzantine fault tolerance.
马高辉, 2002 年生。博士研究生。主要研究方向为时间敏感网络、拜占庭容错。



Sun Zhigang, born in 1974. PhD, professor. His main research interests include software-defined networking, time-sensitive networking, network architecture, and FPGA design.
孙志刚, 1974 年生。博士, 研究员。主要研究方向为软件定义网络、时间敏感网络、网络体系结构、FPGA 设计。