802.1AS 时间同步设计方案

（版本 1.0）

OpenTSN 开源项目组

2021 年 12 月

版本历史

版本	修订时间	修订内容	修订人	文件标识
1.0	2021.12.29	1.初版编制	李军帅	OpenTSN

目录

1 引言

1.1 编写目的

本文档是基于 802.1AS 中 P2P 同步模式设计的时间同步方案，描述了在 P2P 同步的处理流程、报文格式、同步时钟计算以及同步时钟配置等。并且根据主从时间偏差，提出一种关于频率补偿的方式。并且定义与 TSNInsight 通信的方式，以及传输的内容。

编写本文档目的是具体说明在集中式控制下 802.1AS 中 P2P 同步模式实现方式的设计方案和实现细节。

1.2 术语定义

- P2P：peer to peer，点到点
- TSN：Time Sensitive Networking，时间敏感网络；
- PTP：Precision Timing Protocol，精确同步时钟协议；
- TSMP 帧：符合 TSMP 协议规范的帧；
- TSN 控制器：运行 TSN 网络管理软件、具有配置管理 TSN 网络设备能力的 TSN 设备；
- GM：GrandMaster，主时钟

1.3 引用文档

《OpenSync API 设计》

《OpenTSN 时间敏感管理协议（TSMP）规范》

2 总体设计

2.1 设计目标

为了在 OpenTSN 中支持 802.1AS 的 P2P 时间同步，设计开发 P2P 时间同步方案，使各节点的全局时钟与主节点保持一致。该 P2P 时间同步方案既可以用于集中模式下时间同步，又可以在分布式模式下进行时间同步。

P2P 时间同步应用使用一个独立的进程实现。该进程只有一个线程，通过轮询处理超时事件和接收处理报文，实现时间同步和与 TSNInsight 通信的功能。

2.1 总体架构

如图 2-1 所示，时间同步的总体架构包含报文收发模块、定时器管理模块、报文接收模块、时间同步处理函数以及时间同步的 context。其中报文收发模块、报文发送模块、定时器管理模块为基础模块，在该架构下任何同步方式都需要基础模块；而同步处理模块和同步 context 则根据不同的同步方式，模块的内容也不相同，图中为 802.1AS 的 P2P 时间同步。

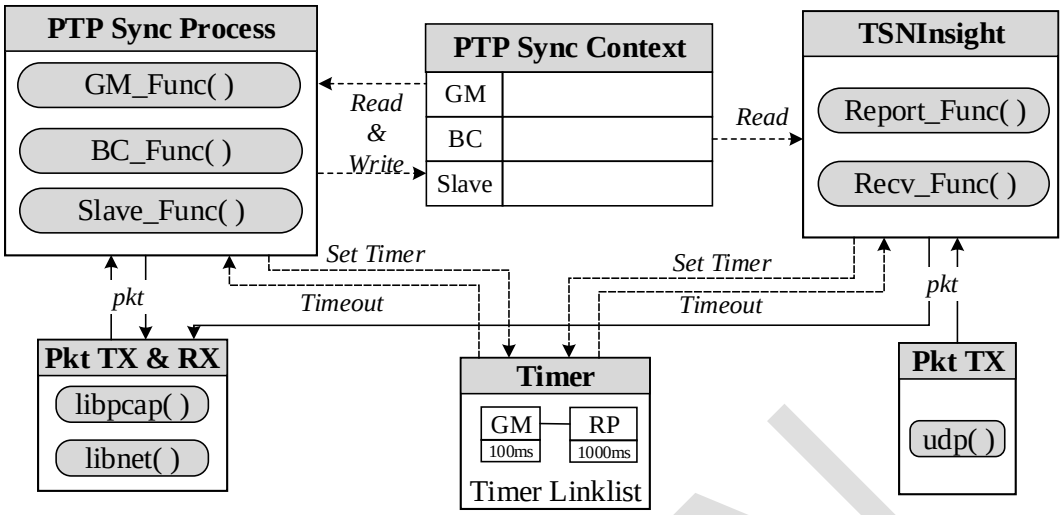

图 2-2 时间同步架构

Pkt TX&RX：报文收发模块，该模块通过 libpcap 捕获报文，然后把报文发送到同步处理模块进行处理，捕获的报文包含两部分，分别为时间同步报文和与 TSNInsight 通信的报文；该模块通过 libnet 发送报文，把时间同步处理模块生成的报文发送到网络中，该模块的发送报文只有同步报文。

Timer Manage：定时管理模块，该模块向同步处理模块提供定时注册服务，如果超时，则执行注册的回调函数。目前在 PTP 时间同步中有两个定时服务，分别为定时发送同步报文和定时上报同步状态。

PTP Sync Process：同步处理模块，该模块按照时钟角色划分为多个时间处理函数，针对于 802.1AS 的 P2P 时钟角色分为主时钟 GM、边界时钟 BC 和从时钟 Slave。该模块可以设置定时器，用于触发时间同步，也可以处理时间同步报文，以完成时间同步的功能。

PTP Sync Context：同步 context 模块，该模块主要定义各个时钟角色使用到的数据结构。

TSNInsight：与 TSNInsight 通信模块，该模块的功能是与 TSNInsight 进行通信，定时上报同步状态，使用 UDP 发送，接收 TSNInsight 命令，使用 libpcap 捕获报文。

2.4 PTP 报文格式

PTP 的报文格式包含报文头和报文体，报文头的格式如下图所示。

Bits								Octets	Offset
7	6	5	4	3	2	1	0		
majorSdoId				messageType				1	0
minorVersionPTP				versionPTP				1	1
messageLength								2	2
domainNumber								1	4
minorSdoId								1	5
flags								2	6
correctionField								8	8
messageTypeSpecific								4	16
sourcePortIdentity								10	20
sequenceId								2	30
controlField								1	32
logMessageInterval								1	33

图 2-4 PTP 报文头格式

图 2-4 为 PTP 报文头格式，根据具体的实现，在报文体格式中，最重要的为 CorrectionField 字段，在实现时，该字段用于存储 t1（sync 在主时钟打的时间戳）和透明时钟（TC）的累加值。

Table 11-8—Sync message fields if twoStep flag is TRUE

Bits								Octets	Offset
7	6	5	4	3	2	1	0		
header (see 11.4.2)								34	0
reserved								10	34

Table 11-9—Sync message fields if twoStep flag is FALSE

Bits								Octets	Offset
7	6	5	4	3	2	1	0		
header (see 11.4.2)								34	0
originTimestamp								10	34
Follow_Up information TLV								32	44

图 2-5 PTP 报文中 sync 报文格式

图 2-5 为 sync 的报文格式，分为两种情况，一种是 twosep 为真时，在 sync 报文中不需要携带时间戳信息；另外一种为 twosep 为假时，需要携带时间等信息。在本设计中采用 twosep 为假的模式，但 originTimestamp 字段和 Follow_Up information TLV 字段设置为默认值 0。

2.5 处理流程

时间同步的总体架构使用一个单进程单线程的模式实现，通过轮询处理超时事件和接收处理报文实现时间同步应用的功能。

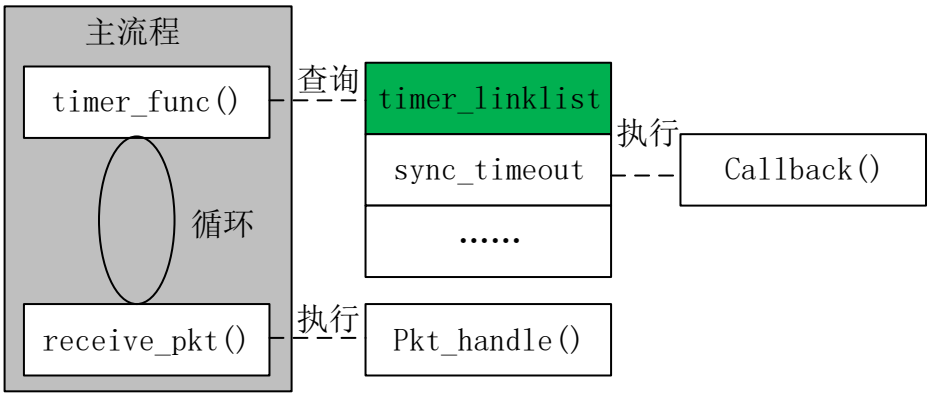

图 2-1 时间同步处理流程

如上图 2-1 所示，在主流程中依次轮询超时处理函数（timer_func）和接收报文处理函数（receive_pkt），在超时处理函数中，通过判断超时链表中具体时间确定是否超时，如果超时，则执行回调函数（callback）。在接收报文处理函数中，如果接收到报文，则执行报文处理函数。

以上为时间同步的简化流程，则具体的处理流程包含链路初始化、周期性发送 sync 报文、解析 sync 报文、发送配置报文。具体的处理流程如下图所示：

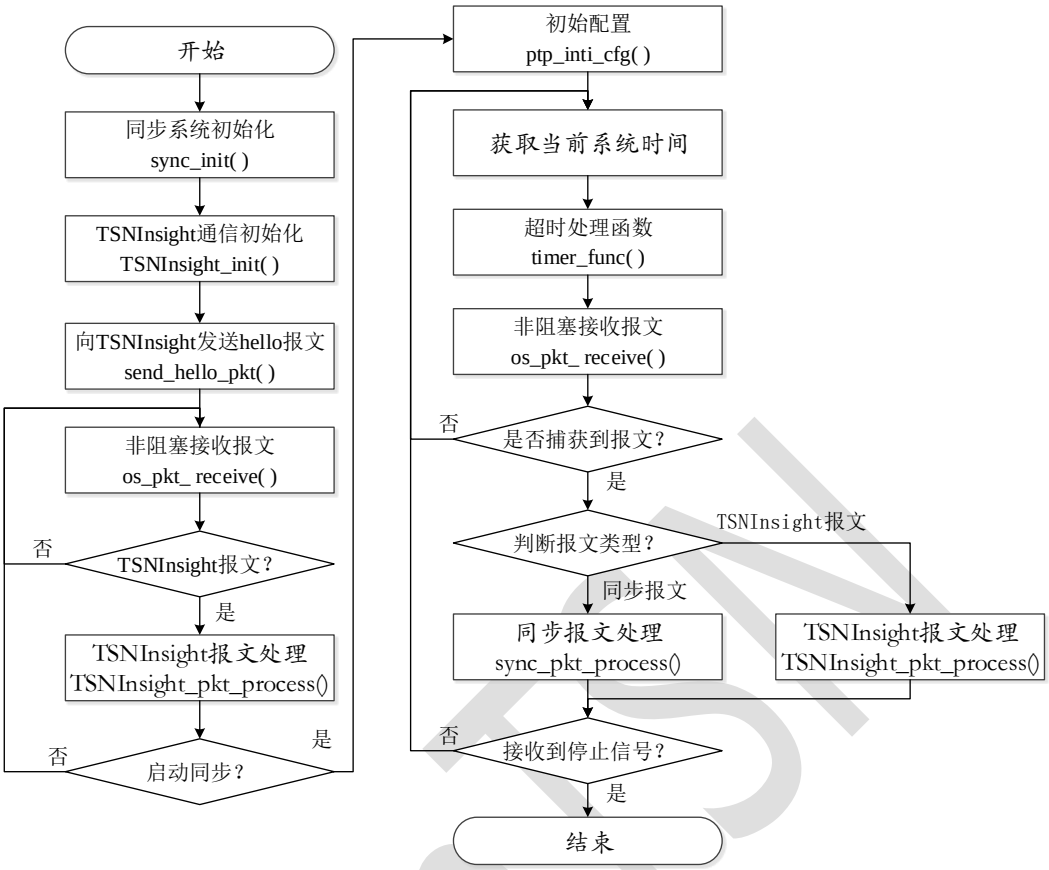

图 2-6 同步报文处理流程

（1）同步系统初始化：初始化同步系统，包括接收报文和发送报文接口初始化，超时链表的初始化以及链路信息的初始化。

（2）TSNInsight 通信初始化：初始化与 TSNInsight 通信的接口，主要初始化使用 UDP 通信的接口，以及通信的数据结构。

（3）向 TSNInsight 发送 hello 报文：PTP 应用在初始化完成后，首先向 TSNInsight 发送 hello 报文，通知 TSNInsight 目前 PTP 应用的 MID 以及角色。

（4）非阻塞接收报文：该步骤接收 TSNInsight 发送的启动时间同步报文，用于开始时间同步。

（5）判断是否为 TSNInsight 报文：判断该报文是否为 TSNInsight

发送的报文。

（6）TSNInsight 报文处理：对 TSNInsight 的报文进行解析，获取 TSNInsight 的指令。

（7）判断是否启动同步：根据第（6）步解析的结果判断是否需要启动同步。

（8）初始配置：配置每个节点同步模式以及同步应用的 mac 地址。

（9）获取当前的系统：调用系统函数获取当前系统时间，用于对超时进行判断。

（10）超时处理：根据第（9）步获取的时间，遍历超时链表，判断是否超时，如果超时，则执行响应的处理函数，否则直接跳转到下一步。

（11）非阻塞接收同步报文：调用 opensync API 轮询接收同步报文。

（12）判断是否捕获到同步报文：如果捕获到报文，则对报文进行处理，否则跳转到第（9）步。

（13）判断报文类型：根据以太网类型判断报文为同步报文还是 TSNInsight 通信报文，然后执行不同的处理函数。

（14）同步报文处理：对接收到的报文进行解析，在 P2P 同步中，需要根据条件当前的时钟角色确定处理函数，例如 GM、BC 和 Slave。

（15）TSNInsight 报文处理：该步骤与第（5）步相同，对 TSNInsight

报文进行解析，获取命令信息。

（16）判断是否接收到停止信号：判断是否有停止信号，如果有，则程序退出，否则跳转到第（9）步。

2.6 关键数据结构

P2P 关键数据结构主要包含设备信息、时钟角色信息以及同步的状态，数据结构关系如下图所示。

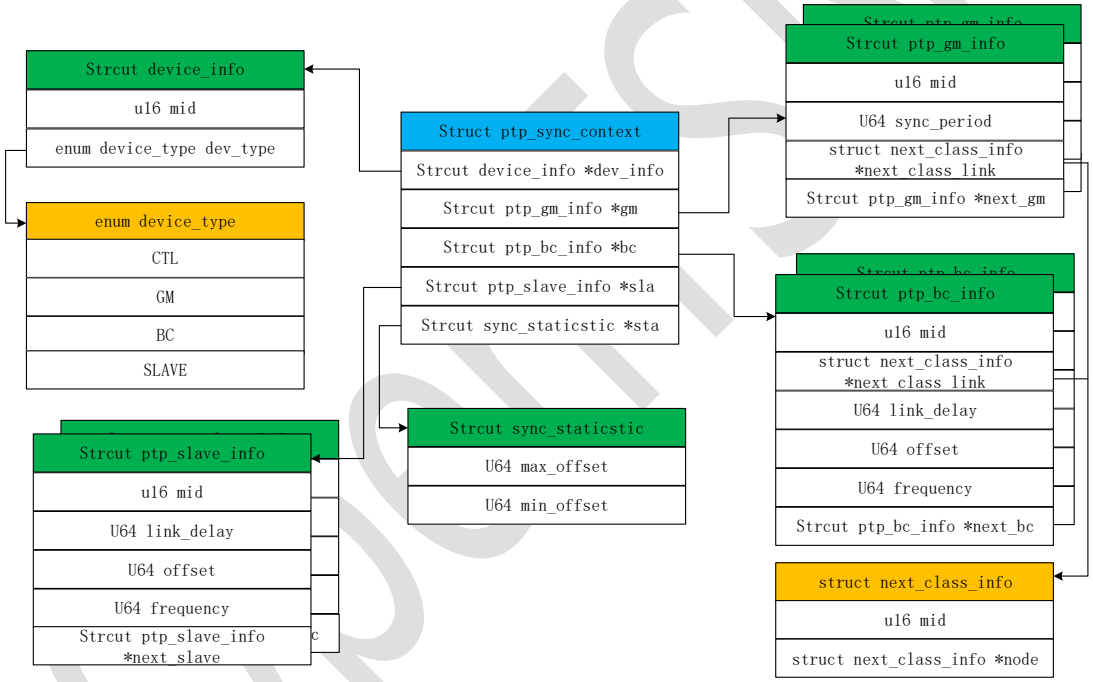

图 2-7 PTP 同步链路数据结构

下表为 P2P 的关键数据结构包含同步链路信息。

表 2-1 PTP 数据结构

```
//ptp同步节点信息
typedef struct
{
    device_info *dev_info;   //设备信息
    ptp_gm_info  *gm;     //主时钟数据结构
    ptp_bc_info  *bc;     //边界时钟数据结构
    ptp_slave_info  *slave;      //从数据结构
```

```
    ptp_staticstic  *sta;     //同步状态

}ptp_sync_context;

//设备信息
typedef struct
{
    u16 mid;   //设备的mid地址,
    enum device_type dev_type;//设备类型

}device_info;

//设备类型枚举
enum device_type
{
    PTP_CTL = 0,
    PTP_GM = 0,
    PTP_BC = 0,
    PTP_slave = 0,
};

//gm节点信息
typedef struct
{
    u16 mid;   //设备的mid地址
    u64 sync_period;//同步周期,单位为ns

    next_class_info *next_class_link;//下一等级的同步链路信息
    ptp_gm_info    *next_gm;  //可能存在多个gm的下一个gm
}ptp_gm_info;

//bc节点信息
typedef struct
{
    u16 mid;   //设备的mid地址
    next_class_info *next_class_link;//下一等级的同步链路信息
    u64 link_delay;       //链路延迟信息，单位ns，必须是8的倍数
    u64 offset;           //同步偏差，单位ns
    u64 frequency;        //频率偏差
    ptp_bc_info    *next_bc;  //使用链表把BC节点串起来
}ptp_bc_info;

//slave节点信息
```

```
typedef struct
{
    u16 mid;   //设备的mid地址
    u64 link_delay;       //链路延迟信息，单位ns，必须是8的倍数
    u64 offset;           //同步偏差，单位ns
    u64 frequency;        //频率偏差
    ptp_slave_info    *next_slave;  //使用链表把slave节点串起来
}ptp_slave_info;

//同步状态信息
typedef struct
{
    u64 max_offset; //本轮同步的最大offset，单位ns
    u64 min_offset; //本轮同步的最小offset，单位ns
}ptp_staticstic;
```

附录 A. 802.1AS 点对点(P2P)同步工作原理

P2P 时间同步包含两部分，一部分为链路测量，用于测量相邻节点主从端口间的链路延迟，另外一部分为主从时钟同步，主时钟给从时钟进行授时。

P2P 的时钟类型包含 GM（主时钟）、BC（边间时钟）、TC（透明时钟）和 slave（从时钟），并且逐级进行同步。

P2P 的报文类型包含延迟测量报文（Pdelay_req、Pdelay_resp），同步报文包含 sync。

802.1AS 规定必须使用 P2P 进行延迟测量，下图为延迟测量的流程。

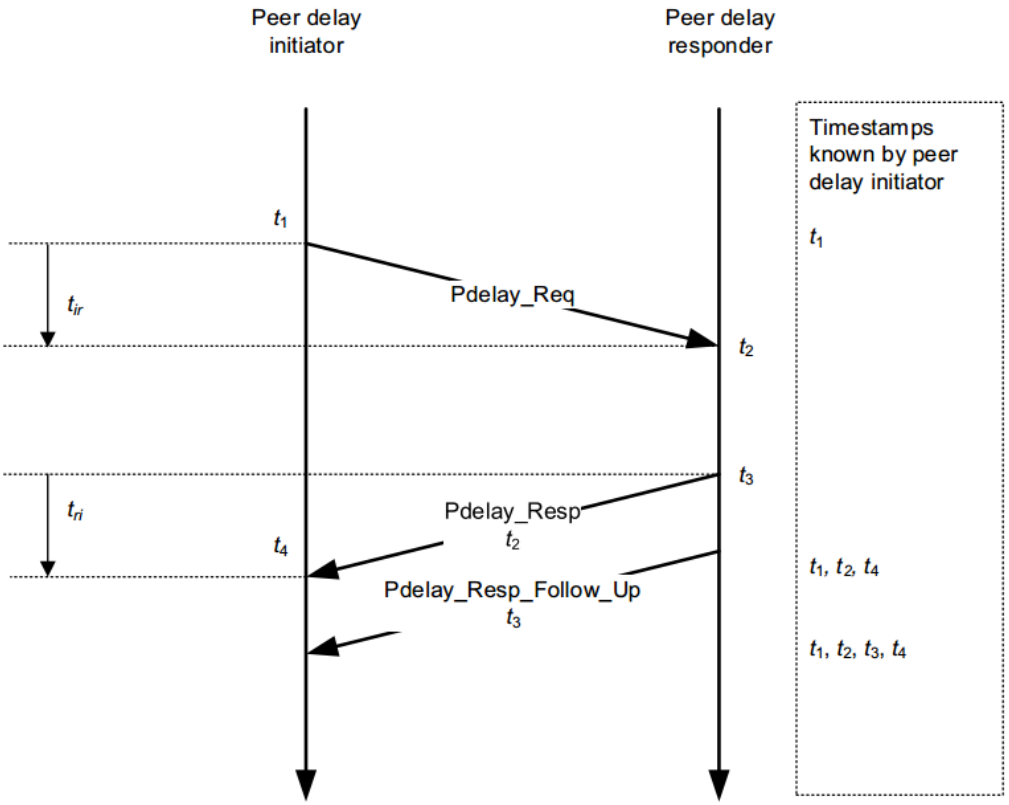

图 2-1 Peer-To-Peer 延迟测量报文交互示意图

第一步：Slave Port 周期性向对端发生 Pdelay_Req 请求端口链路延迟测量；

第二步：Master Port 响应链路延迟测量，返回 Pdelay_Resp 报文；

第三步：Master Port 发送 Pdelay_Resp_Follow_Up 报文携带时间戳 t3；

第四步：Slave Port 根据 PTP 报文中时间戳计算链路延迟 PDelay = ((T4-T3)+(T2-T1))/2。

时间同步则是逐级进行同步，下图为三个相邻的 802.1AS 中时间同步的流程

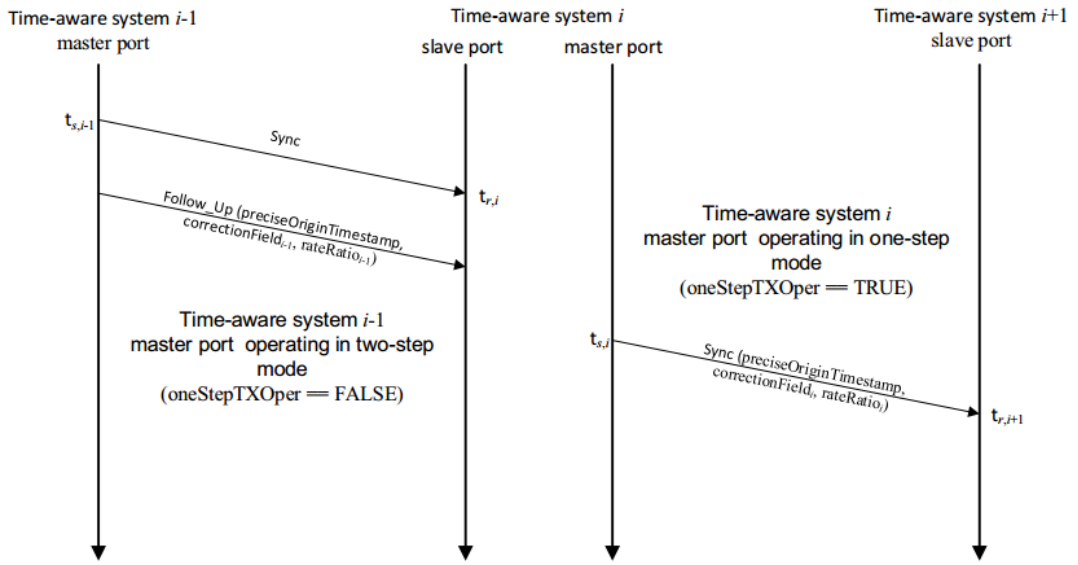

图 2-2 Peer-To-Peer 同步报文交互示意图

第一步：system i-1 中的 Master Port 周期性向下一级 system i 的 Slave Port 发出 Sync 同步报文；

第二步：system i-1 中 Master Port 发送 Follow_Up 报文，携带最初发送同步信息时的时间以及透明时钟；（当 oneStepTXOper == FALSE 时该步骤有效，否则该步骤无效）

第三步：system i 中 Master Port 周期性向下一级 system i+1 的 Slave Port 发出 Sync 同步报文。