基于 OpenSync 的 TTE 时钟同步控制软件设计方案

（版本 1.0）

OpenTSN 开源项目组

2022 年 05 月

版本历史

版本	修订时间	修订内容	文件标识
1.0	2022.5	初版编制	OpenTSN3.4

目录

1. 缩略语 4
2. 项目概述 5
2.1. 设计目标 5
2.2. 实现思路 5
3. 总体设计 6
3.1. 程序架构 7
3.2. 处理流程 8
3.2.1. 主函数流程 9
3.2.2. 超时处理函数流程 10
3.2.3. 报文处理函数流程 11
3.3. 数据结构 12
3.3.1. struct tte_sync_context 12
3.3.2. struct timer_list_node 13
附录 A. 数据 PCF 协议格式 15
附录 B. OpenSync 协议格式 16
附录 C. AS6802 同步参数 17
C.1. CM 同步参数 17
C.2. SM 同步参数 18

1. 缩略语

本文包含以下缩略语：

AS6802 时间触发以太网的时钟同步协议

GMII (Gigabit Media Independent Interface) 千兆媒体独立接口

HCP (Hardware Control Point) 硬件控制点

LID (Local IDentification) 本地标识

Syn_clk (Synchronization clock) TSN 交换机同步时钟，该时钟与网络时钟进行同步

Local_cnt (Local time) TSN 交换机本地时间，该时间只在 TSN 交换机中被使用，用于计算 PTP/PCF 报文在 TSN 交换机中驻留时间

MAC (Media Access Control) 媒体访问控制，或称物理地址、硬件地址

MHz (Mega Hertz) 兆赫兹，时钟频率单位

ms (millisecond) 毫秒

ns (nanosecond) 纳秒

OpenTSN (Open Time-sensitive Networking) 开源时间敏感网络项目

OpenSync (Open Synchronization) 开源同步框架

PCF (Protocol Control Frame) 协议控制帧

PKT (PacKeT) 报文

PTP (Precision Time Protocol) 精准时间协议

RAM (Random Access Memory) 随机访问存储器

RC (Resource Constraint) 资源约束流量

s (second) 秒

SMAC (Source Media Access Control) 源物理地址

ST (Scheduled Traffic) 预定流量，即时间敏感流量

sync (synchronization) 主时钟节点发送给从时钟节点的同步报文

TC (Transparent Clock) 透明时钟

TSMP (Time Sensitive Management Protocol) 时间敏感管理协议，详见文档“OpenTSN 时间敏感管理协议（TSMP）简介”

TSN (Time-Sensitive Networking) 时间敏感网络

μs (microsecond) 微秒

2. 项目概述

本文档是介绍基于 OpenSync 实现时间触发以太网 TTE 中的时钟同步协议 AS6802，文档分为缩略语，项目概述以及总体设计三个部分。

2.1. 设计目标

基于可编程时钟同步实现框架 OpenSync 实现单进（线）程的 AS6802 时间同步控制程序，实现以下目标：（1）该程序既可以运行于集中式控制器，也可以运行于分布式节点上；（2）在集中式架构下实现时，能够在单线程中实现时钟同步中所有角色的处理逻辑。

2.2. 实现思路

将 AS6802 时间同步实现过程中所有的操作划分为两类：消息触发操作和超时触发操作。程序顶层是循环监测这两类操作，当收到报文时调用指定的函数对报文进行处理，否则判断是否有超时发生，超时则调用超时处理函数。

（1）消息触发操作

程序采用非阻塞的方式接收报文，即收不到报文时不等待直接返回，返回之后进行超时的判断；收到报文则对报文进行相应的处理。

（2）超时触发操作

程序使用的超时判断是通过获取当前时间与超时计时器起点时刻的差值，循环判断是否超时，超时则执行超时处理函数。

除了对上述两种操作触发方式的实现，如果要支持不同的时间同步角色，还需要设计每设备的同步状态上下文，用于记录每个设备的同步状态和信息。当超时触发或者消息触发时，相应的处理函数依据当前的状态进行操作，而后更新状态。

3. 总体设计

OpenSync 为实现时钟同步协议提供了编程接口，包括报文的接收发送，OpenSync 协议的解析与构造，本地时钟的修改与校正等。基于 OpenSync 框架开发时钟同步协议能够仅通过软件编程不同的协议控制程序，不需要对硬件进行修改。

总体设计从程序的架构、程序的主要流程以及核心数据结构三个部分进行阐述。

3.1. 程序架构

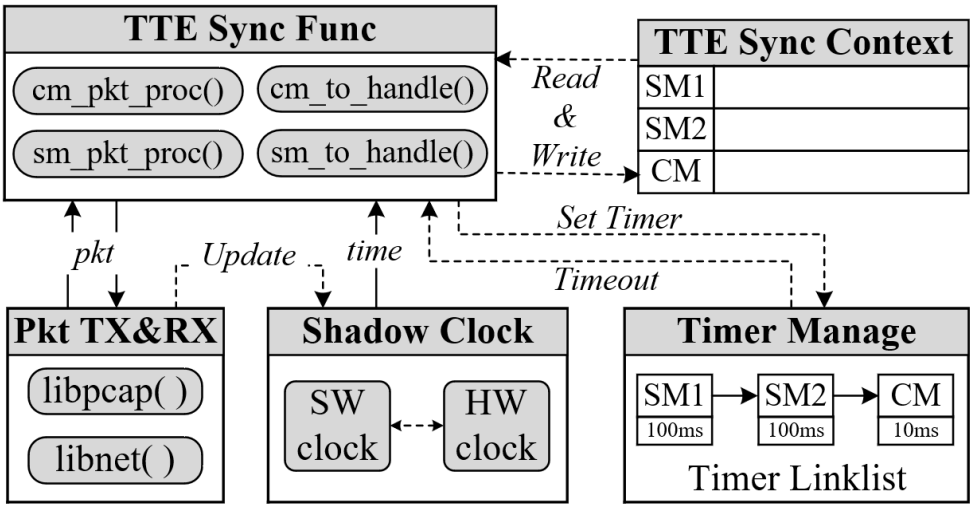


图 1 程序整体架构图

程序按照功能逻辑可以分为五个部分，其中两部分与具体时间同步协议相关，余下三个部分与具体协议无关，包括 OpenSync 框架中提供的编程接口（接收、发送报文）和定时器管理机制。

（1）TTE Sync Func

AS6802 时间同步处理函数，包括用于处理超时触发的函数与消息触发的函数。按照时钟角色可以分为 SM 和 CM（本次实现暂不包括 SC）。每个函数都是根据触发的条件和当前设备上下文中记录的状态完成同步相关的操作，并更新上下文中的状态。

（2）TTE Sync Context

AS6802 时间同步上下文，上下文中定义每个设备在同步过程中需要使用的所有数据。当程序运行于分布式节点上，只需要一条上下文；如果是在集中式控制器上，需要处理多个设备的数据时，上下文需要构成一个表，每条表项对应一个设备。在同步处理函数执行时，除了当次的触发条件，还需要查询对应设备的上下文信息。

（3）Pkt TX&RX

OpenSync 提供的接收报文接口，可以使用 OpenSync API 提供的接口进行报文的接收与发送。

（4）Shadow Clock

该部分维护软件系统时间与硬件接收时间戳的映射关系，这种映射关系不具有确定性，只能作为模糊的参考时间，不能作为精确的硬件时间。OpenSync API 提供获取该参考时间的接口。OpenSync 在接收报文的过程中会为每个报文记录硬件的接收时间戳，所以在收到报文的时刻会同时更新 Shadow Clock 的映射关系。

（5）Timer Manage

定时器管理，每个定时器对应一个设备，当程序对应多个设备时，各个定时器之间以链表的形式组合。主函数不断地轮询该链表，判断是否超时，如果超时则根据定时器 ID 查询对应设备的上下文，然后调用对应的超时处理函数。

3.2. 处理流程

由于程序是单进程单线程的实现方式，所以逻辑比较复杂，在设计的过程中为了展现清晰的逻辑结构，增强代码的可读性，采用了分层设计，本部分主要介绍主函数的流程以及超时处理函数、消息处理函数的通用处理流程。

3.2.1. 主函数流程

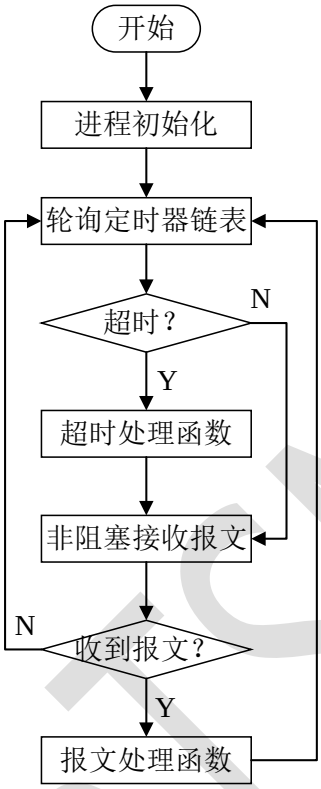


图 2 主函数处理流程图

（1）首先，进程初始化。初始化的内容包括每个设备上下文的数据结构（同步角色、同步参数等）、定时器的初始值、用于接收和发送报文的句柄；

（2）轮询定时器链表。获取当前时刻之后，与定时器链表中每个节点的定时器起始时刻做差，差值与定时器时长比较，如果超时，进入超时处理函数；如果所有的定时器均未超时，则尝试非阻塞接收报文；

（3）超时处理函数的步骤见 3.2.2 节；

（4）非阻塞接收报文。如果接收到报文，进入报文处理函数；否则，跳转至步骤（2）轮询定时器链表；

（5）报文处理函数的步骤见 3.2.3 节。报文处理结束之后跳转至步骤（2）。

3.2.2. 超时处理函数流程

当轮询定时器链表，监测到超时之后，进入超时处理函数。根据超时的定时器节点信息可以获取这个定时器的 ID，定时器 ID 与设备 ID 一一对应，从而可以根据这个 ID 读取该设备的上下文信息，其中也包括了当前设备的同步状态，接下来就是进入该状态下的超时处理。超时处理的动作主要包括同步状态跳转、设置新的定时器或者发送报文等。

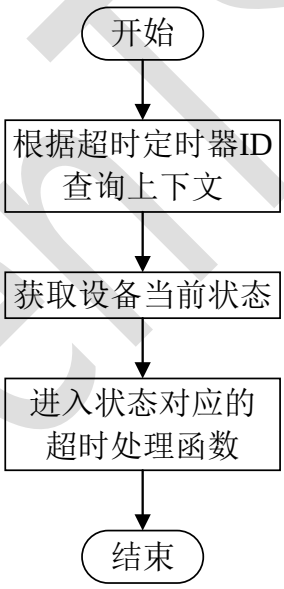


图 3 超时处理函数流程图

3.2.3. 报文处理函数流程

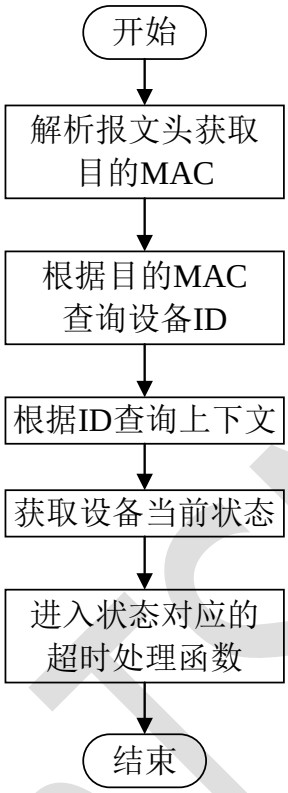
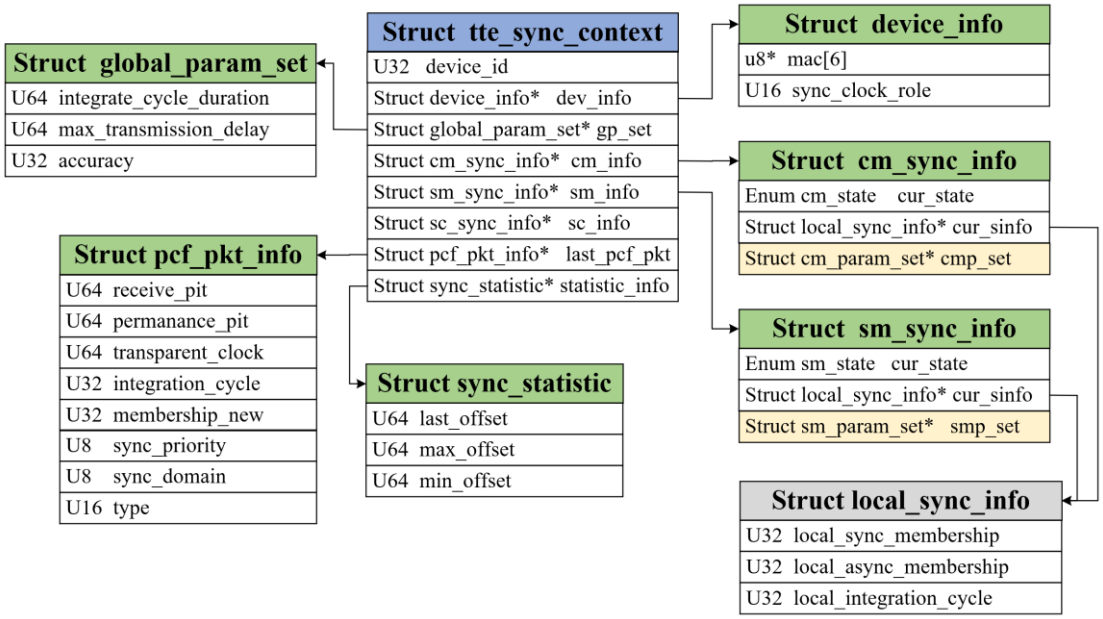


图 4 报文处理函数流程图

当非阻塞的报文收到一个报文时，首先解析报文头获取该报文的目的 MAC，根据目的 MAC 可以查询对应设备 ID，从而能够访问该设备对应的上下文信息，其中包括了设备当前的同步状态，接下来进入状态对应的报文处理。报文处理函数是将收到的报文、当前状态作为输入，输出的动作包括设置定时器、发送报文或者状态跳转等。

3.3. 数据结构

3.3.1. struct tte_sync_context



该结构体用于存储 AS6802 同步上下文，包含 8 个属性。

（1）设备标识 device id，唯一标识网络中的设备，用于可以根据报文的目的 MAC 或者定时器 ID 进行检索，从而能够访问对应设备的上下文；

（2）设备信息 dev_info，记录与设备相关的信息:设备 MAC 地址和在同步过程中的时钟角色；

（3）同步全局参数 gp_set，对于 AS6802 时间同步，有些参数是全局的，即所有的设备都会使用同样的参数，集成周期、最大传输延时和精度；

（4）3 个与时钟同步角色相关的结构体指针，分别指向 CM、SM、SC 对应的同步信息结构体，由于每个设备在同步过程中只

能担任一种角色，所以当时钟角色确定之后，其他 2 个指针会被初始化未空指针，不被使用；

（5）接收到的报文信息 last_pcf_pkt, 用于记录最新接收到报文的摘要信息，这一部分和报文相关，报文按照标准描述的格式如下，该结构体属性的设置也参照该格式；

	0-15			16-31
0	Integration_cycle			
32	Membership_new			
64	Reserved			
96	Sync_priority	Sync_domain	Type	Reserved
128	Reserved			
160	Transparent_clock			
192				

（6）统计信息 statistic_info，记录该设备在同步过程中计算的最大相位偏差、最小相位偏差以及最新一次集成周期的相位偏差，该信息用于分析设备的同步情况。

3.3.2. struct timer_list_node

该结构体是构成定时器链表的节点，包含三个属性：设备标识 device_id 用于将定时器与设备绑定，每个设备只能设置一个计时器，在进行超时处理时才能够提前查询到对应的上下文；定时器起始点 timer_start ，定时机制在获取当前时间后，用当前时间与该值作差，差值与定时器 timer_length 比较，如果前者大于则判断为超时，进入超时处理函数；否则判断定时器未超时，等待下一次超时检查。

Struct timer_list_node
U32 device_id
U64 timer_start
U64 timer_length

附录A. 数据 PCF 协议格式

TSN 交换机接收/发送的 PCF 报文格式如图 F-2 所示，PCF 报文中关键字段的说明如表 F-2 所示。

<table>
<tr><td></td><td colspan="2">0-15</td><td colspan="2">16-31</td></tr>
<tr><td>0</td><td colspan="4">Integration_cycle</td></tr>
<tr><td>32</td><td colspan="4">Membership_new</td></tr>
<tr><td>64</td><td colspan="4">Reserved</td></tr>
<tr><td>96</td><td>Sync_priority</td><td>Sync_domain</td><td>Type</td><td>Reserved</td></tr>
<tr><td>128</td><td colspan="4">Reserved</td></tr>
<tr><td>160</td><td colspan="4" rowspan="2">Transparent_clock</td></tr>
<tr><td>192</td></tr>
</table>

图 F-2 PCF 报文（不包括以太网头）格式

表 F-2 PCF 报文中关键字段说明

名称	位宽 bit	说明
Integration_Cycle	32	集成周期，标识设备在时间上位于当前集群周期中的第几个集成周期，系统未同步时值为 0。当集成周期到达配置的上限值后会从 0 重新计数。
Membership_New	32	对应于 SM 的成员向量，每个比特位标识一个 SM 设备的同步情况，当为“1”时代表对应的 SM 设备完成了同步。
Sync_priority	8	同步优先级，标识网络中设备参与同步过程的优先级，CM 设备只处理与其相同同步优先级的 PCF 帧，SM 设备会处理与其相同以及更高同步优先级的 PCF 帧。
Sync_Domain	8	同步域，网络中处于相同的同步域的设备才可能进行 PCF 帧通信，不同同步域设备发来的 PCF 帧会被丢弃。
Type	4	标识 PCF 帧的具体类型，CS 帧的值为 0x4， CA 帧的值为 0x8，IN 帧值为 0x2。
Transparent Clock	64	透明时钟，用于记录 PCF 帧在转发过程中的时延；其低 16bit 数值单位为 2^{-16}，高 48bit 数值单位为 ns；控制器在构造 PCF 协议报文时将该字段填为 0。

附录B. OpenSync 协议格式

OpenSync 采用 Mac-in-Mac 的方式，

确保 OpenSync 帧能够在交换模块中能够根据 MAC 地址进行正常转发。OpenSync 帧类型用 TSMP 的类型标识。

32 个字节的 Sync 头包含 16 字节的 MAC/TSMP 头，以及 16 字节的时间信息。格式定义见 OpenTSN 的定义，主要包含以下几个域：

偏移量	名称	
0-5	目的 MAC	与时间同步帧的目的 MAC 一致
6-11	源 MAC	与时间同步帧的源 MAC 一致
12-13	长度/类型	TSMP 协议的类型（ff01）
14-15	TSMP 类型	标识 OpenSync 的 TSMP 子类型
16-23	接收/发送 PIT	使 Syn_clk 标记的 64 位帧在网络接口接收（Rx_pit）和发送（Dispatch_pit）时刻
24-31	接收发送时间戳 TS	使用 local_cnt 标记的 64 位时间戳

附录C. AS6802 同步参数

C.1. CM 同步参数

参数名称	数据类型	定义
observation_window	u64	CM 角色的压缩函数在收集阶段的观察窗口时间
max_observation_window	u64	CM 角色的压缩函数在收集阶段的全部观察窗口总时间
calculation_overhead	u64	CM 角色的压缩函数在计算阶段的开销时间
cm_dispatch_delay	u64	CM 角色的 PCF 帧派发延时
cm_scheduled_receive_pit	u64	CM 角色计划接收 IN 帧的时间点，可离线计算
cm_listen_timeout	u64	CM_INTEGRATE 状态之后等待的时间
cm_ca_enabled_timeout	u64	CM 角色在 CM_CA_ENABLED_TIMEOUT 状态继续转发 CA 帧的持续时间
cm_wait_4_in_timout	u64	CM 角色在冷启动阶段等待 1 个 pcf_membership 域高有效位数 IN 帧的等待时间
cm_integrate_tsync_thrld	u32	CM_INTEGRATE 状态转移到 CM_SYNC 状态的 IN 帧中 pcf_membership 域的最小有效位数
cm_integrate_twait_thrld	u32	CM_INTEGRATE 状态转移到 CM_WAIT_4_IN 状态的 IN 帧中 pcf_membership 域的最小有效位数
cm_wait_threshold_sync	u32	CM_WAIT_4_IN 状态下收到的不在接收窗口内的 IN 帧中 pcf_membership 域的最小有效位数
cm_unsync_tsync_thrld	u32	CM_UNSYNC 状态转移到 CM_SYNC 状态的 IN 帧中 pcf_membership 域的最小有效位数
cm_unsync_ttentative_thrld	u32	CM_UNSYNC 状态转移到 CM_TENTATIVE_SYNC 状态的 IN 帧中 pcf_membership 域最小有效位数
cm_tentative_sync_thrld_async	u32	CM_TENTATIVE_SYNC 状态下判定异步结团时 local_async_memberhip 的最小有效位数

cm_tentative_sync_thrld_sync	u32	CM_TENTATIVE_SYNC 状态下判定同步结团时 local_sync_memberhip 的最小有效位数
cm_tentative_sync_thrld	u32	CM_TENTATIVE_SYNC 状态转移到 CM_ SYNC 状态的 IN 帧中 pcf_membership 域的最小有效位数
cm_sync_thrld_async	u32	CM_ SYNC 状态下判定异步结团时 local_async_memberhip 的最小有效位数
cm_sync_thrld_sync	u32	CM_ SYNC 状态下判定同步结团时 local_sync_memberhip 的最小有效位数
cm_stable_thrld_async	u32	CM_ STABLE 状态下判定异步结团时 local_async_memberhip 的最小有效位数
cm_stable_thrld_sync	u32	CM_ STABLE 状态下判定同步结团时 local_sync_memberhip 的最小有效位数

C.2. SM 同步参数

参数名称	数据类型	定义
sm_local_membership	u32	当前 SM 设备分配的 membership 位
ca_round_trip	u64	CA 帧从发送到接收最终接收经历的延迟
sm_dispatch_pit	u64	SM 角色发送 IN 帧的时刻
smc_scheduled_receive_pit	u64	SM 和 SC 角色计划接收 IN 帧的时间点
sm_listen_timeout	u64	定义同步主控器从初始化进入 SM_INTEGRATE 状态之后等待的时间
sm_coldstart_timeout	u64	同步主控器派发 CS 帧的周期
sm_cs_offset	u64	同步主控器在一个 CS 帧固化之后、派发一个应答的 CA 帧之前等待的持续时间
sm_ca_offset	u64	同步主控器在一个 CA 帧固化之后、派发第一个 IN 帧之前等待的持续时间

sm_restart_timeout	u64	用在同步状态向异步状态跳转时，SM 在发送一个 CS 帧之前，尝试重新集成到以前系统的等待时间
sm_integrate_to_sync_thrld	u32	SM_INTEGRATE 状态转移到 SM_SYNC 状态的 IN 帧中 pcf_membership 域的最小有效位数
sm_integrate_to_wait_thrld	u32	SM_INTEGRATE 状态转移到 SM_WAIT_4_CYCLE_START 状态的 IN 帧中 pcf_membership 域的最小有效位数
sm_wait_thrld_async	u32	SM_WAIT_4_CYCLE_START 状态下判定异步结团时 local_async_memberhip 的最小有效位数
sm_unsync_to_sync_thrld	u32	SM_UNSYNC 状态转移到 SM_SYNC 状态的 IN 帧中 pcf_membership 域最小有效位数
sm_unsync_to_tentative_thrld	u32	SM_UNSYNC 状态转移到 SM_TENTATIVE_SYNC 状态的 IN 帧 pcf_membership 最小值
sm_tentative_sync_threshold_async	u32	SM_TENTATIVE_SYNC 状态下判定异步结团时 local_async_memberhip 最小有效位数
sm_tentative_sync_threshold_sync	u32	SM_TENTATIVE_SYNC 状态下判定同步结团时 local_sync_memberhip 的最小有效位数
sm_tentative_to_sync_thrld	u32	SM_TENTATIVE_SYNC 状态转移到 SM_SYNC 状态的 local_sync_memberhip 最小位数
sm_sync_threshold_async	u32	SM _SYNC 状态下判定异步结团时 local_async_memberhip 的最小有效位数
sm_sync_threshold_sync	u32	SM _SYNC 状态下判定同步结团时 local_sync_memberhip 的最小有效位数
sm_stable_threshold_async	u32	SM _STABLE 状态下判定异步结团时 local_async_memberhip 的最小有效位数
sm_stable_threshold_sync	u32	SM _STABLE 状态下判定同步结团时 local_sync_memberhip 的最小有效位数