TSN 网络控制器(TSNLight3.4)设计方案

（版本 1.0）

OpenTSN 开源项目组

2022 年 04 月

版本历史

版本	修订时间	修订内容	修订人	文件标识
1.0	2022.4.12	1.初版编制	开源项目组	OpenTSN3.4

目录

1 引言

1.1 编写目的

本文档是 OpenTSN 控制器 TSNLight3.4 的设计文档，主要描述了 OpenTSN 控制器 TSNLight3.4 实现的功能、工作流程、核心数据结构以及各模块的详细设计。

编写本文档目的是定义和说明 OpenTSN 控制器 TSNLight3.4 的设计方案和实现细节。

1.2 术语定义

- TSN：Time Sensitive Networking，时间敏感网络；
- PTP：Precision Timing Protocol，精确同步时钟协议；
- MID：Management Identity Document，管理身份标识；
- TSMP：Time Sensitive Management Protocol，时间敏感管理协议；
- TSNLight：OpenTSN 控制器；
- TSNInsight：可视化网络管理软件；
- Qbv：时间感知整形调度；
- Qch：队列循环转发调度；
- ST 流：时间敏感流量；

1.3 引用文档

（1）《OpenTSN 控制架构规范》；

2 TSNLight 3.4 概述

2.1 简介

OpenTSN 控制器 TSNLight3.4 采用单进程单线程的方式实现，实现网络初始化、网络配置以及在网络运行状态检测网络状态的功能。

相比于 TSNLight3.3 版本，TSNLight3.4 主要修改的内容如下：

（1）时间同步功能从 TSNLight 中剥离，由 OpenSync 软件实现；

（2）TSMP 协议报文类型及报文格式发生变化；

（3）引入 MID 概念，MAC 地址基于 MID 生成；

（4）引入 TSMP 转发表概念，TSMP 帧的寻址依赖于 TSMP 转发表；

（5）增加 TSNLight 与 TSNInsight 的通信。

2.2 总体架构

TSNLight3.4 整体架构如图 2-1 所示，包括通用基础库和功能模块两部分内容。功能模块主要实现网络初始化、基础配置、本地规划配置和网络状态检测等应用功能；通用基础库主要实现南北向通信接口等基本功能。

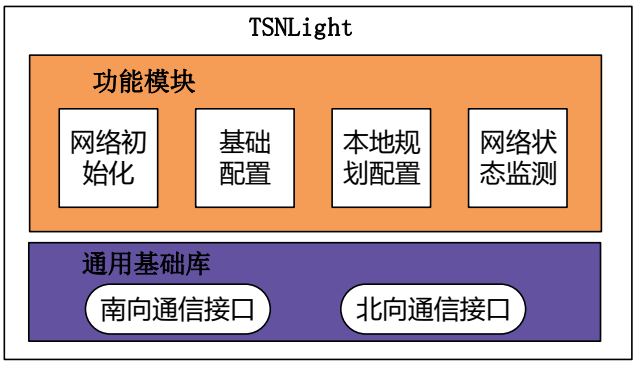

图 2-1 TSNLight3.4 系统架构

TSNLight 3.4 与硬件平台的南向通信接口遵循 TSMP 协议规范，与上层应用（如 TSNInsight）的北向通信遵循 TSNLight 的动态通信接口规范。

TSNLight 3.4 所示实现的网络初始化、基础配置、本地规划配置和网络状态检测等应用功能，简单介绍如下：

（1）网络初始化

网络初始化主要实现网络接口初始化，包含数据接收初始化、数据发送初始化。在初始化时，需要指定网卡的名称（以命令行输入的形式获取），以及设置过滤规则。其中数据发送使用 raw_socket 的方式，数据接收使用 libpcap 的方式。

（2）基础配置

基础配置主要是通过解析 tsnlight_init_cfg.xml 文本获取基础配置信息，然后配置硬件控制点 HCP 寄存器地址空间，完成 TSN 网络控制通路的初始化。

（3）本地规划配置

本地规划配置主要是通过解析 tsnlight_plan_cfg.xml 文本获取规

划工具输出参数配置信息，然后配置 TSN 交换核心或 TSN 网卡核心等用户自定义的硬件地址空间，实现 TSN 网络正确传输时间敏感流量和带宽预约流量。

对于规划信息配置，其配置的内容与用户应用流量及规划调度等相关，配置的寄存器地址是用户自定义的。所以关于规划信息配置，TSNlight 仅提供配置通道，由用户提供配置寄存器地址和配置内容。

（4）网络状态检测

TSNlight 通过周期性扫描网络设备节点，获取各个设备节点的基本状态信息，初步分析当前网络是否处于正常运行（未实现）。

3 总体设计

3.1 管理状态机

TSNLight3.4 采用单进程单线程的方式实现，其工作流程基于状态机进行控制。TSNLight3.4 共定义五个状态，分别为初始状态、基础配置状态、本地规划配置状态、时间同步初始状态和网络运行状态。各个状态机之间的跳转关系，如图 3-1 所示。

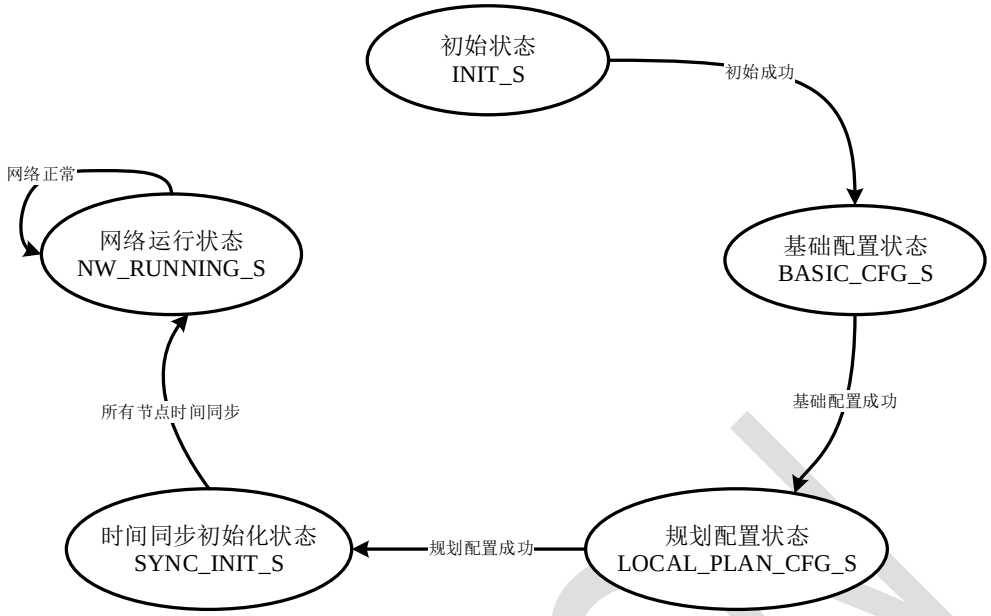

图 3-1 TSNLight3.4 管理状态机

各个状态的功能描述及状态跳转说明如下：

（1）初始状态

该状态的功能是完成接收和发送数据报文的初始化。

（2）基础配置状态

在该状态下，通过解析 tsnlight_init_cfg.xml 文本获取基础配置信息，完成各个节点的基础信息配置。该状态主要是打通整个网络的控制通路，使得所有控制报文能够正确转发。

（3）规划配置状态

在该状态下，通过解析 tsnlight_plan_cfg.xml 文本获取与流量规划相关的配置信息。该状态只关心配置内容，而不关心具体的语义。

（4）时间同步初始化状态

在该状态下，TSNLight 开启时间同步服务，将网络中的主从时间

偏差调整到一定范围内。仅时间同步成功后才允许传输时间敏感流量。

（5）网络运行状态

在该状态下，完成网络状态监测和远程配置管理功能。若出现异常则结束。（目前未实现）。

3.2 工作流程

TSNLight3.4 的工作流程如图 3-2 所示。

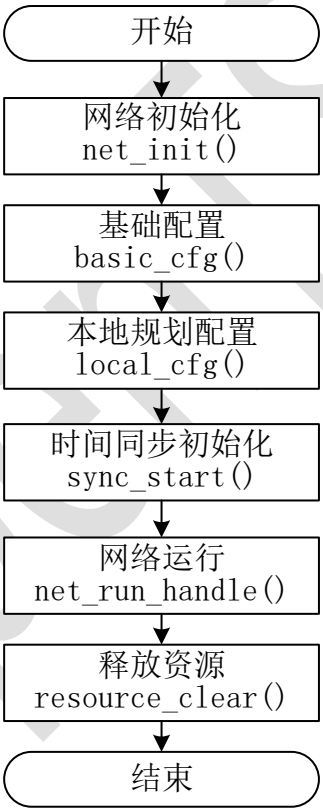

图 3-2 TSNLight3.4 工作流程图

TSNLight3.4 的工作流程详细描述如下：

（1）网络初始化：实现初始状态机功能。

（2）基础配置：实现基础配置状态机功能。

（3）本地规划配置：实现本地规划配置状态机功能。

（4）时间同步初始化：开启时间同步服务，等待全网时间同步。

（5）网络运行处理：实现网络运行状态机功能。

（6）释放资源：实现结束状态机功能。

3.3 对外通信接口

OpenTSN 控制器 TSNlight 对外接口支持动态通信接口和静态通信接口两种。

远程规划配置管理及状态展示等功能的实现依赖于动态通信接口，如 TSNlight 与 TSNInsight 的通信。关于动态通信接口详细定义及说明，请参考《OpenTSN 控制架构规范》文档“3.2 动态通信接口”所述。

基础配置和本地规划配置功能依赖于静态通信接口，如 TSNlight 配置硬件控制点 HCP 的 MID、TSMP 转发表等。静态通信接口采用的是静态 xml 文本实现方式。静态 xml 文本包含两个，分别是 tsnlight_init_cfg.xml 文本和 tsnlight_plan_cfg.xml 文本。关于静态通信接口详细定义及说明，请参考《OpenTSN 控制架构规范》文档“3.1 静态通信接口”所述。

3.4 核心数据结构

TSNLight 基于 TSMP 协议对 OpenTSN 交换机和 OpenTSN 网卡进行全集中式的管理控制，TSNLight 与 OpenTSN 交换机或 OpenTSN

网卡通信的控制报文均为 TSMP 帧，所以 TSMP 协议报文数据结构是 TSNLight 的核心数据结构。

关于 TSMP 协议报文类型及报文格式详细定义及说明，请参考《OpenTSN 控制架构规范》文档“5 TSMP 协议规范”所述。

表 3-1 核心数据结构

```
/*tsmp报文头部*/
typedef struct
{
    u8  dmac[6];   //dmac
    u8  smac[6];   //源mac
    u16 eth_type; /* 以太网类型，TSMP以太网类型为0xff01   */
    u8  type;      /* TSMP协议类型   */
    u8  sub_type; /* TSMP协议子类型   */
}__attribute__((packed))tsmp_header;

typedef enum
{
    MID_DISTRI = 0x01,    //地址分配
    NET_MANAGE = 0x02,     //网络管理
    NET_TETEMETRY = 0x03,   //网络遥测
    TUNNEL_ENCAPSULATION = 0x04,   //隧道封装
    TIME_ANNUNCIATE = 0x05,   //时间通告
    OPENSYNC = 0x06,   //OpenSync时间同步
}TSMP_TYPE;


//网络管理子类型
typedef enum
{
    GET_REQ = 0x01, //读请求
    SET_REQ = 0x02,     //写请求
    GET_RES = 0x03,   //读响应
    TRAP = 0x04,   //trap上报
}NET_MANAGE_SUB_TYPE;

//get_req子类型报文数据域
typedef struct
{
```

```
    u16  num;//数目
    u32  base_addr;//基地址
}__attribute__((packed))tsmp_get_req_pkt_data;

//set_req和get_res子类型报文数据域
typedef struct
{
    u16  num;//数目
    u32  base_addr;//基地址
    u32  data[0];//数值
}__attribute__((packed))tsmp_set_req_or_get_res_pkt_data;
```