TSN 集中控制器设计文档

（版本 1.1）

OpenTSN 开源项目组

2021 年 5 月

版本历史

版本	修订时间	修订内容	文件标识
1.0	**2021.4.23**	**该版本为接口规范文档初始版本，主要包含 OpenTSN 交换机数据平面抽象、OpenTSN 交换机配置流程、OpenTSN 交换机配置格式**	**TSNLight3.0 单进程 集中式控制器**
1.1	**2021.5.17**	**该版本使用单进程单线程实现，使用状态机描述整体的状态；增加与上层应用通信的接口。**	

目录

1 设计目标.....5
2 总体设计.....5
2.1 系统结构.....5
2.2 工作流程.....6
2.3 整体流程.....9
2.3.1 整体流程.....9
2.3.2 超时处理函数流程.....11
2.4 整体数据结构.....12
3 详细设计.....错误!未定义书签。
3.1 控制器初始化与网络配置.....错误!未定义书签。
3.1.1 功能描述.....错误!未定义书签。
3.1.2 处理流程.....错误!未定义书签。
3.1.3 数据结构.....错误!未定义书签。
3.1.4 编程接口.....错误!未定义书签。
3.2 时间同步.....错误!未定义书签。
3.2.1 功能描述.....错误!未定义书签。
3.2.2 处理流程.....错误!未定义书签。
3.2.3 数据结构.....错误!未定义书签。
3.2.4 编程接口.....错误!未定义书签。
3.3 网络运行状态.....错误!未定义书签。
3.3.1 功能描述.....错误!未定义书签。
3.3.2 详细流程.....错误!未定义书签。
3.3.3 数据结构.....错误!未定义书签。
3.3.4 编程接口.....错误!未定义书签。
4 TSNLight 与上层应用交互内容.....错误!未定义书签。
4.1 交互流程.....错误!未定义书签。
4.2 交互信息.....错误!未定义书签。
附录 1:组网示例.....13

附录 2:CNC_API 13
附录 2.1 数据接收相关 API 13
附录 2.2 数据发送相关 API 14
附录 2.3 TSMP 协议相关 API 15
附录 2.4 配置芯片和 HCP 相关 API 15
附录 2.5 上报相关 API 17

1、设计目标

该方案对集中控制器进行修改，主要修改的内容包含以下内容，增加与上层控制应用通信的北向协议模块，通过代理与上层应用通信；使用状态机描述工作流程，并详细描述每个状态具体功能；使用单进程单线程的形式，方便以后移植到嵌入式 CPU 中。TSNLight3.0 可以在 OpenTSN2.0 和 OpenTSN3.0 工程中使用。

2、总体设计

2.1 系统结构

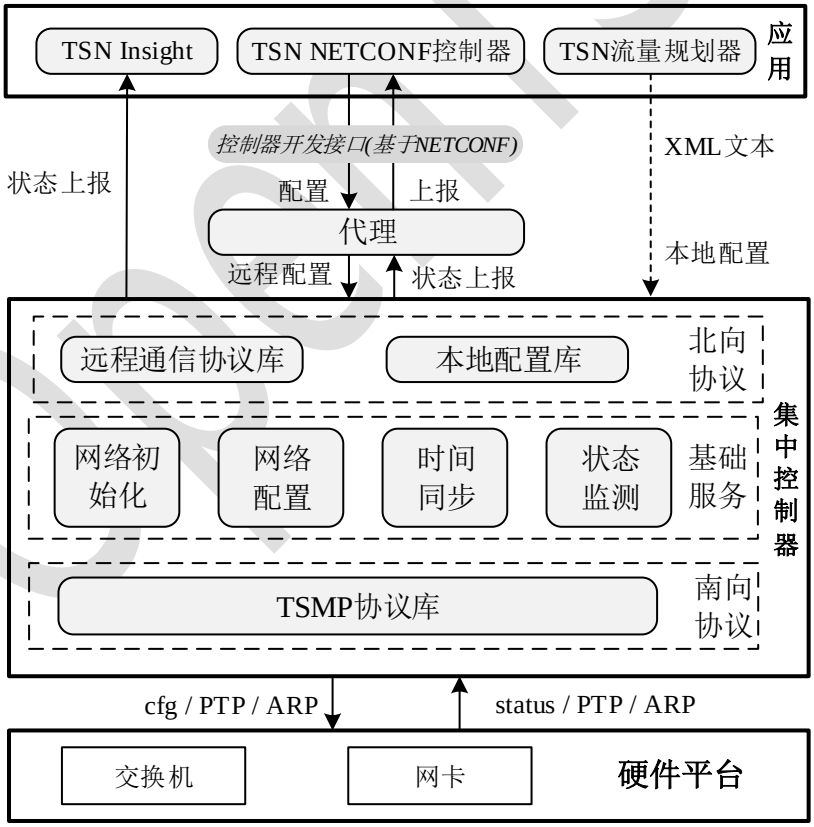


图 2-1 TSN 集中控制器系统结构

TSNLight 为 TSN 网络的本地管理器（集中控制器），完成对 TSN 网络初始化、配置、时间同步以及状态监测的功能。TSN 代理作为上

层应用与集中控制器通信的桥梁，负责通信协议转换。上层应用可以根据用户的流量进行离线规划，最后把配置信息封装成数据报文的形式，通过代理传输到集中式控制器，也可以直接生成一个 xml 文本，由集中控制器直接读取该文本获取离线规划配置信息。

控制器总体使用一个进程实现，在进程中使用 libpcap 非阻塞式接收报文，并且每次只接收一个报文。当在超时时间内没有接收到报文时，跳出该函数，根据当前状态执行超时处理函数；当接收到报文时，根据当前所处的状态调用不同的回调函数对报文进行处理。

2.2 工作流程

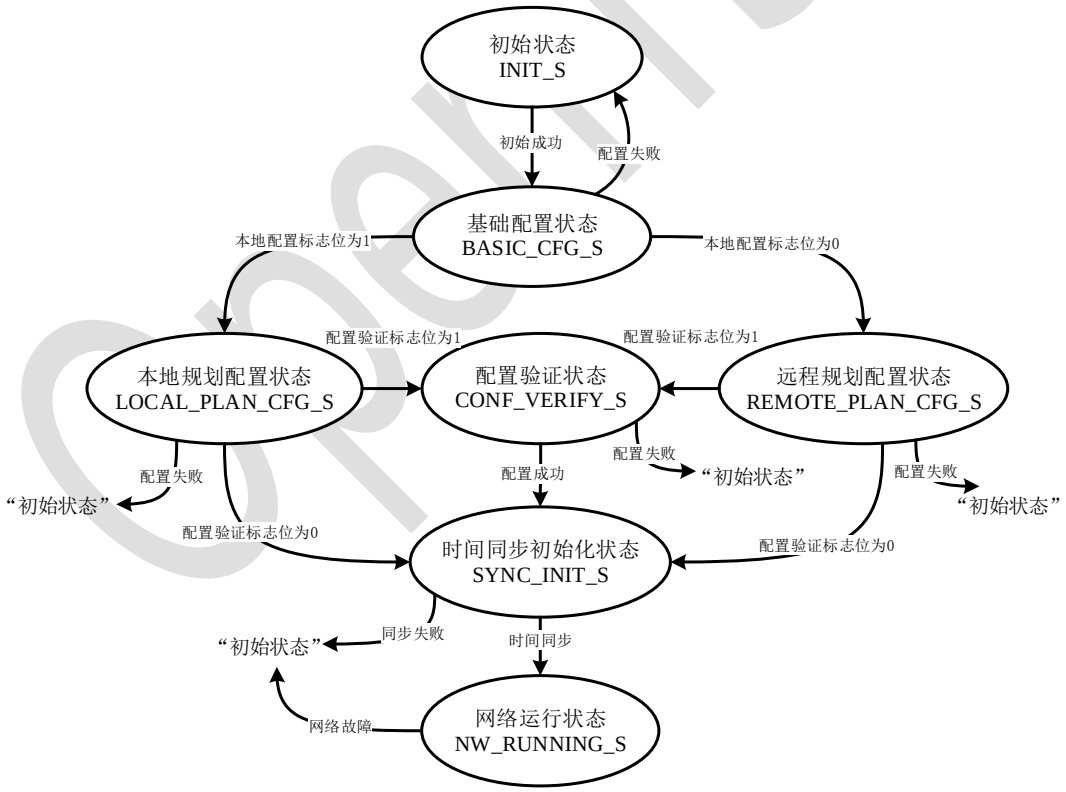


图 2-2 TSN 控制器工作状态图

工作流程分为七个状态，分别为初始状态、基础配置状态、本地规划配置状态、远程规划配置状态、配置验证状态、时间同步初始化

状态和网络运行状态。各个状态的具体运行流程在第三节详细设计中描述。

初始状态：完成各个变量的初始化，libnet 和 libpcap 接口的初始化,根据初始配置文本，填充初始配置数据结构。初始成功后跳转到基础配置状态。

表 2-1 初始状态跳转说明

当前状态	跳转条件	下一个状态
初始状态	初始结束	基础配置状态

基础配置状态：根据在初始状态获取的拓扑验证数据结构，完成节点基本参数的配置以及拓扑验证的功能。

表 2-2 基础配置状态跳转说明

当前状态	跳转条件	下一个状态
基础配置状态	基础配置成功、 本地配置标志位为 1	本地规划配置状态
	基础配置成功、 本地配置标志位为 0	远程规划配置状态
	基础配置失败、 net_run=0	初始状态

远程规划配置状态：接收离线规划配置信息，并存储在全局的配置信息数据结构中，然后按照接收的顺序对节点进行配置，每接收一个节点，配置一个节点。

表 2-3 远程规划配置状态跳转说明

当前状态	跳转条件	下一个状态
远程规划配置状态	远程规划配置成功、 配置验证标志位为 1	配置验证状态
	远程规划配置成功、 配置验证标志位为 0	时间同步初始化状态
	远程规划配置失败、 net_run=0	初始状态

本地配置状态：从本地的离线规划 xml 配置文本中获取配置信息，并对配置信息进行解析，存储在全局的配置信息数据结构中，按照文

本中节点出现的先后顺序进行解析，每次解析一个节点，配置一个节点。

表 2-3 本地配置状态跳转说明

当前状态	跳转条件	下一个状态
本地规划配置状态	本地规划配置成功、配置验证标志位为 1	配置验证状态
	本地规划配置成功、配置验证标志位为 0	时间同步初始化状态
	本地规划配置失败、net_run=0	初始状态

配置验证状态：根据接收到的验证信息，对上报寄存器进行配置，用于获取需要验证的信息（上报信息），并把接收到的上报信息提交到上层应用，由上层应用判断是否配置成功。

表 2-4 配置验证状态跳转说明

当前状态	跳转条件	下一个状态
配置验证状态	配置验证成功	配置验证状态
	配置验证成功 net_run=0	初始状态

时间同步初始化状态：完成时间同步的功能，负责把网络中的主从时间偏差调整到一定范围内。

表 2-5 时间同步状态跳转说明

当前状态	跳转条件	下一个状态
配置验证状态	时间同步成功	网络运行状态
	时间同步成功 net_run=0	初始状态

网络运行状态：网络正常工作状态，通过对接收到的报文进行解析，进行相应的操作（时间同步、ARP 相应、状态监测）。

表 2-6 网络运行状态跳转说明

当前状态	跳转条件	下一个状态
网络运行状态	网路故障 net_run=0	初始状态

2.3 整体流程

2.3.1 整体流程

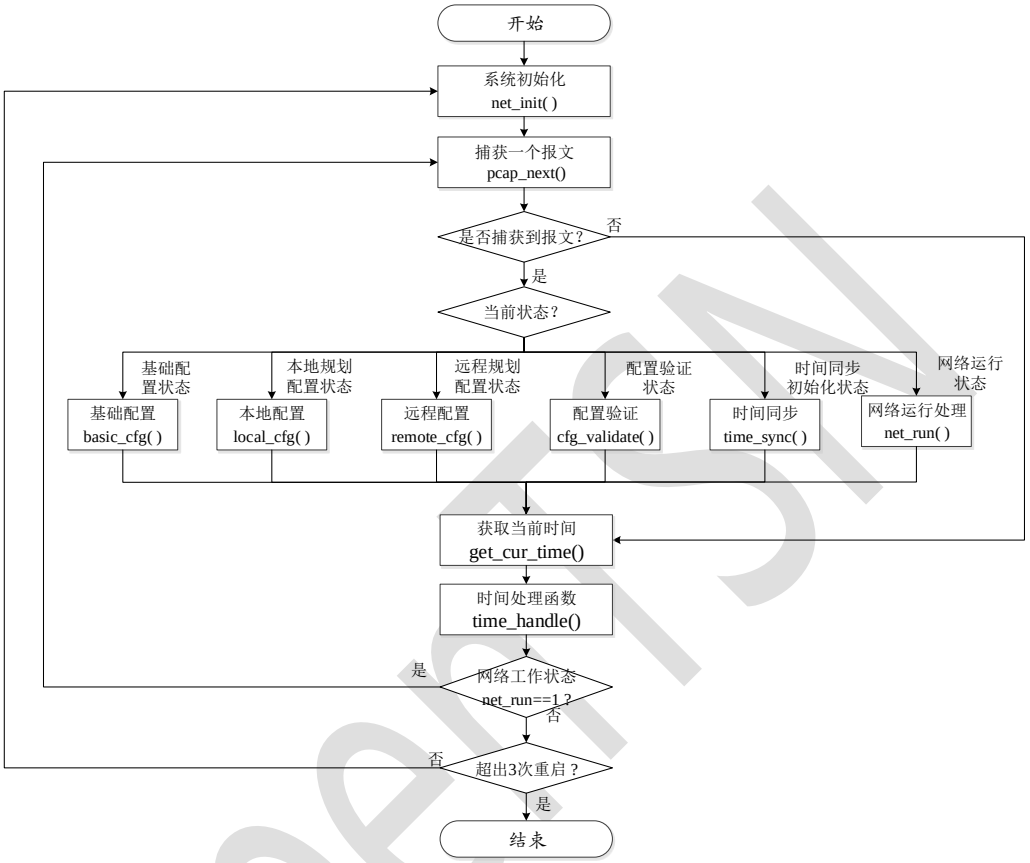


图 2-3 整体流程图

主函数伪代码：

表 2-7 主函数伪代码

```
int main（）
{
    init://goto 的标志位，网络重启
     net_init（）//网络初始化，初始化网络配置、时间同步、状态监测使用到的参数
     init_cfg( )//初始配置，先对控制器直连的节点进行默认配置，
```

是该节点能够上报

```
while（1）
{
  u8 *pkt = pcap_next(p_head);//非阻塞捕获一个报文，超时或者捕获到报文时往下执行

  if（pkt ！= NULL）//捕获到报文
  {
   switch（G_STATE）//根据状态判断需要进行的处理逻辑
  {
    case  BASIC_CFG_S：basic_cfg（p_head，pkt）；//网络基础配置；
    case  LOCAL_PLAN_CFG_S：本地规划配置；
    case  REMOTE_PLAN_CFG_S：远程规划配置；
    case  CONF_VERIFY_S：配置验证；
    case  SYNC_INIT_S：时间同步初始化；
    case  NW_RUNNING_S：时间同步，状态监测，动态配置；
   }
  }
  gettimeofday (time);//获取当前时间，用于判断是否超时
  time_handle(cur_state,time)；//根据本地时间判断是否超时
```

```
        if(work_run==1) //判断网络是否正常工作，该标志位在时间处理函数更改
            continue
        else
            goto:init;//跳转到初始状态

        if（restart_num>3）//判断重启是否超过三次
            break;
        else
            continue
    }
    return 0;
}
```

2.3.2 超时处理函数流程

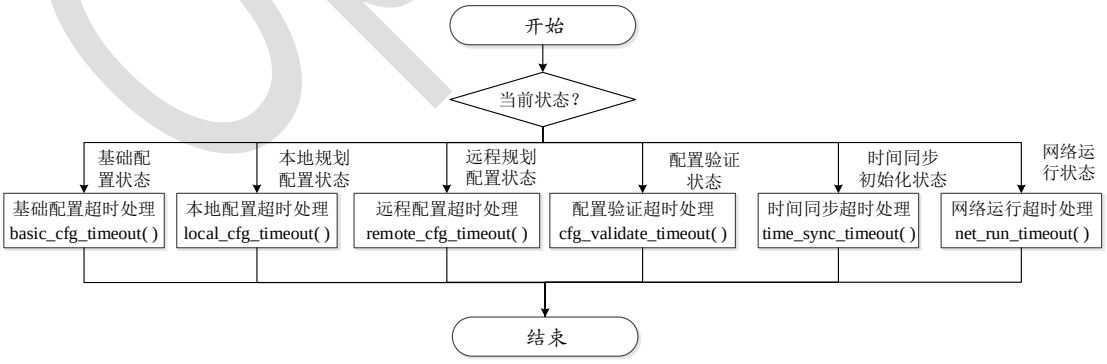


图 2-4 超时处理函数流程图

2.4 整体数据结构

网络数据结构用来表示网络中全部信息，使用节点表示每个单元，所有节点组合成一个网络。

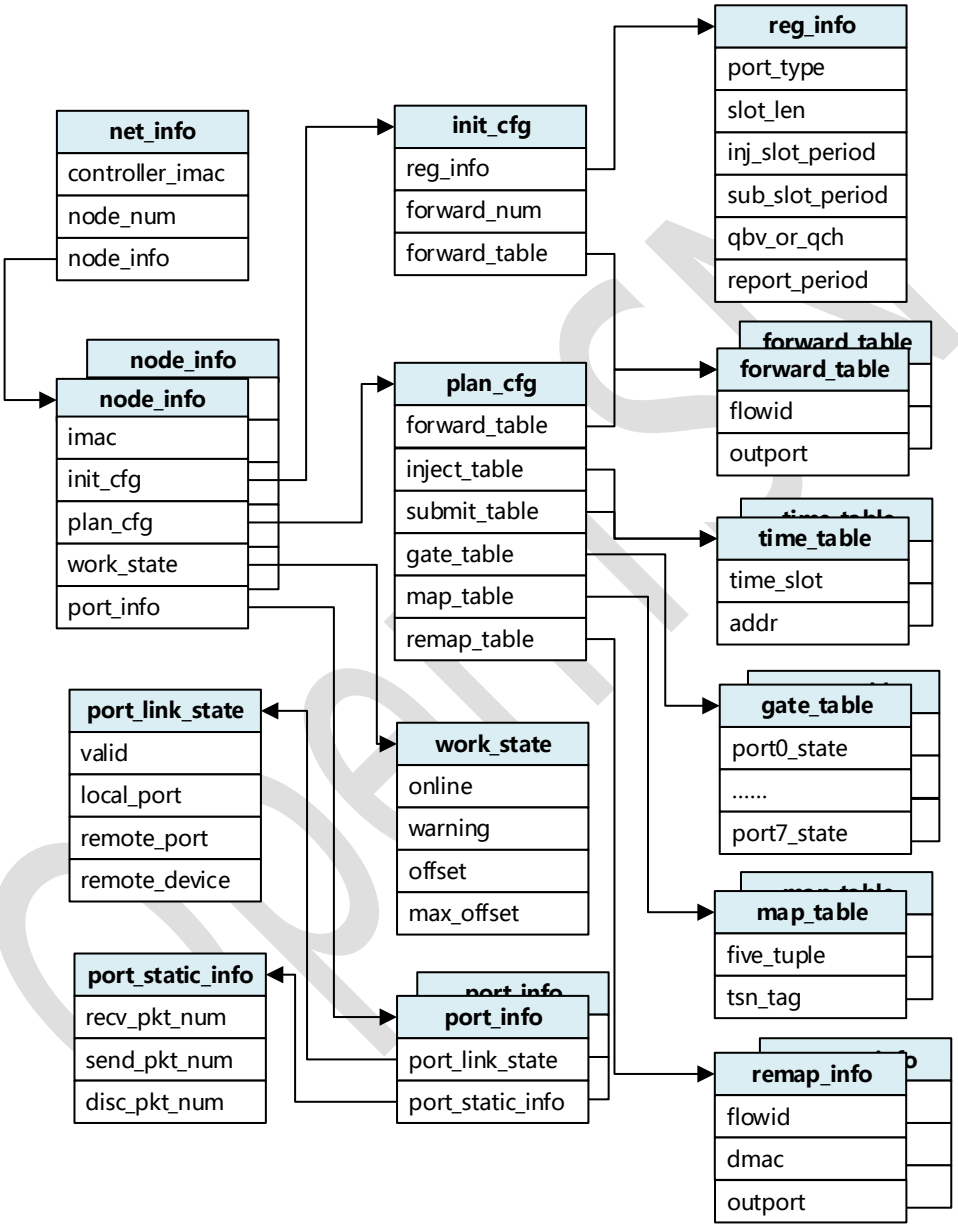
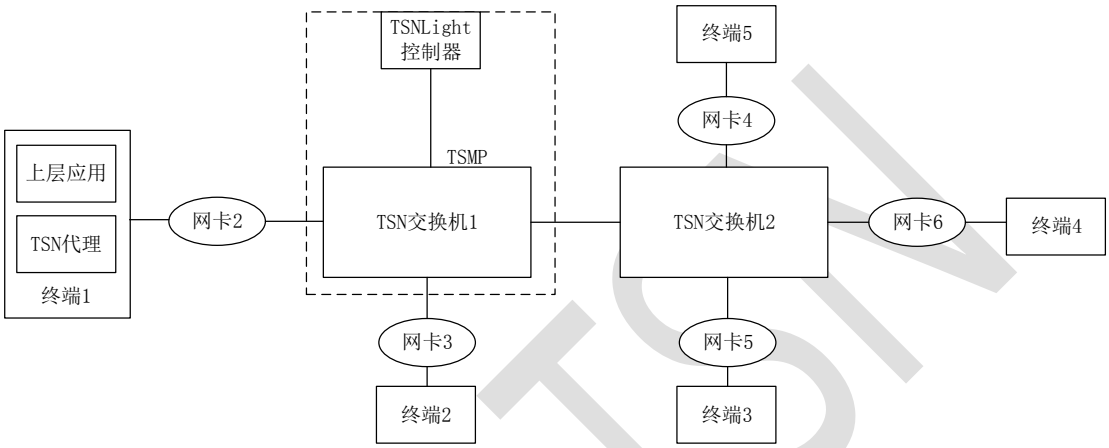


图 2-5 数据结构关系图

具体的数据结构的内容参考第三章详细设计中各个模块的数据结构。

附录 1、组网示例

组网示例如下图所示，控制器与 TSN 交换机直连，终端与 TSN 交换机之间需要连接一个网卡，上层应用和代理运行在其中一个终端上面。设计基于 OpenTSN3.0 版本



图符 1-1 组网示例图

附录 2、CNC_API

cnc_api 为 API 库源代码文件，提供数据接收、数据发送、TSMP 协议、配置芯片和 HCP、上报等相关库函数供其他应用程序调用。

附录 2.1 数据接收相关 API

（1）数据报文接收初始化函数

函数定义	int data_pkt_receive_init(u8* rule);
输入参数	过滤规则字符串指针，示例：u8* rule= "ether[3:1]=0x01 and ether[12:2]=0xff01";
返回结果	成功返回 0，失败返回-1
功能描述	完成数据报文接收资源的初始化。包括 libpcap 句柄的初始化、打开网络设备、设置过滤规则等。

（2）数据报文接收处理函数(循环抓包）

函数定义	int data_pkt_receive_loop(pcap_handler callback);

输入参数	数据报文处理函数 callback
返回结果	成功返回 0，失败返回-1
功能描述	循环接收数据报文，并且将数据报文送给 callback 函数进行处理。

（3）数据报文接收处理函数(每次抓一个包）

函数定义	int data_pkt_receive_dispatch (pcap_handler callback);
输入参数	数据报文处理函数 callback
返回结果	成功返回 0，失败返回-1
功能描述	循环接收数据报文，并且将数据报文送给 callback 函数进行处理。

（4）数据报文接收处理函数(每次抓一个包,非阻塞)

函数定义	int data_pkt_receive_dispatch_1 (pcap_handler callback);
输入参数	数据报文处理函数 callback
返回结果	成功返回 0，失败返回-1
功能描述	循环接收数据报文，并且将数据报文送给 callback 函数进行处理。

（5）数据报文接收销毁函数

函数定义	int data_pkt_receive_destroy ( );
输入参数	无
返回结果	成功返回 0，失败返回-1
功能描述	完成数据报文接收资源的销毁

附录 2.2 数据发送相关 API

（1）数据报文发送初始化函数

函数定义	int data_pkt_send_init();
输入参数	无
返回结果	成功返回 0，失败返回-1
功能描述	完成数据报文发送资源的初始化。包括 raw scoket 句柄的初始化、指定网卡名称、原始套接字地址结构赋值等

（2）数据报文发送处理函数

函数定义	int data_pkt_send_handle(u8* pkt,u16 len);
输入参数	数据报文指针、数据报文长度
返回结果	成功返回 0，失败返回-1
功能描述	完成数据报文的发送处理

（3）数据报文发送销毁函数

函数定义	int data_pkt_send_destroy();
输入参数	无

返回结果	成功返回 0，失败返回-1
功能描述	完成数据报文发送相关资源的销毁

附录 2.3 TSMP 协议相关 API

（1）TSMP 报文头构造函数

函数定义	u8* build_tsmp_pkt(tsmp_sub_type type,u16 dimac,u16 pkt_len)
输入参数	tsmp 子类型，目的 imac，除 TSMP 头之外的报文长度
返回结果	成功返回偏移 TSMP 头之后的数据报文指针地址，失败返回 NULL
功能描述	完成 tsmp 报文内存空间的申请，TSMP 头的赋值。

（2）TSMP 报文头（源目的）地址互换函数

函数定义	int tsmp_header_switch_handle(u8* pkt,u16 len);
输入参数	tsmp 报文头指针、tsmp 报文长度
返回结果	成功返回去掉 TSMP 头之后的数据报文指针地址，失败返回 NULL
功能描述	完成 tsmp 报文头源目的 mac 的互换

（3）TSMP 报文头提取源 MAC 函数

函数定义	u16 get_simac_from_tsmp_pkt(u8* pkt,u16 len);
输入参数	tsmp 数据报文指针、tsmp 数据报文长度
返回结果	成功返回 imac 值（主机序），失败返回-1
功能描述	从 tsmp 报文头中取出源 mac 的 imac 字段，返回的 imac 是主机序

（4）TSMP 报文内存地址释放函数

函数定义	void free_pkt(u8* pkt);
输入参数	tsmp 数据报文指针
返回结果	无
功能描述	对报文进行内存释放

附录 2.4 配置芯片和 HCP 相关 API

（1）芯片配置报文构造与发送函数

函数定义	int build_send_chip_cfg_pkt(u16 dimac,u32 addr,u8 num,u32 *data);
输入参数	dimac 配置设备的 imac 地址，addr 表示配置寄存器首地址，num 表示配置的数量,不超过 16 个，data 表示配置内容指针

返回结果	配置和发送成功返回 0，失败返回-1
功能描述	构建和发送芯片配置报文。

（2）HCP 配置报文构造与发送函数

函数定义	int build_send_hcp_cfg_pkt(u16 dimac,u32 addr,u32 *data,u16 num)
输入参数	dimac 配置设备的 imac 地址，addr 表示配置寄存器地址，data 表示配置内容指针，num 表示配置的数量。
返回结果	配置和发送成功返回 0，失败返回-1
功能描述	构建和发送 HCP 配置报文。

（3）芯片单个寄存器配置函数

函数定义	int cfg_chip_single_register(u16 dimac,chip_reg_info chip_reg)
输入参数	dimac 配置设备的 imac 地址，chip_reg 表示单个寄存器的值
返回结果	配置和发送成功返回 0，失败返回-1
功能描述	配置芯片的所有单个寄存器,并进行确认。

（4）芯片表项配置函数

函数定义	int cfg_chip_table(u16 dimac,u8 type,chip_cfg_table_info chip_cfg_table)
输入参数	dimac 配置设备的 imac 地址，chip_cfg_table 表示表项。
返回结果	配置和发送成功返回 0，失败返回-1
功能描述	配置芯片的表项，并进行确认函数。

（5）HCP 映射表配置函数

函数定义	int cfg_hcp_map_table(u16 imac,map_table_info *map_table)
输入参数	imac 配置设备的 imac 地址，map_table 表示映射表的数据结构。
返回结果	配置和发送成功返回 0，失败返回-1
功能描述	配置 hcp 映射表，并进行确认函数。

（6）HCP 重映射表配置函数

函数定义	int cfg_hcp_remap_table(u16 imac,remap_table_info *remap_table)
输入参数	dimac 配置设备的 imac 地址，remap_table 表示重映射。
返回结果	配置和发送成功返回 0，失败返回-1
功能描述	配置芯片的表项，并进行确认函数。

（7）64 位主机序转网络序函数

函数定义	u64 htonll(u64 value);
输入参数	64 位主机序数据
返回结果	64 位网络序数据
功能描述	64bit 主机序转网络序

（8）芯片单个寄存器主机序转网络序函数

函数定义	void host_to_net_single_reg(u8 *host);
输入参数	主机数据指针
返回结果	无
功能描述	芯片单个寄存器主机序转网络序

（9）芯片表项主机序转网络序函数

函数定义	void host_to_net_chip_table(u8 *host,u16 len);
输入参数	主机数据指针,数据长度
返回结果	无
功能描述	芯片表项主机序转网络序

（10）芯片单个寄存器信息打印函数

函数定义	void printf_single_reg(chip_reg_info *chip_reg);
输入参数	芯片寄存器信息地址指针
返回结果	无
功能描述	打印单个芯片寄存器信息

（11）芯片表项信息打印函数

函数定义	void print_chip_report_table(chip_report_table_info *report_entry);
输入参数	芯片上报表项信息结构体指针
返回结果	无
功能描述	打印芯片上报表项信息

附录 2.5 上报相关 API

（12）获取上报报文类型函数

函数定义	u16 get_chip_report_type(u8 *pkt,u16 len);
输入参数	上报报文指针，报文长度
返回结果	上报报文上报类型
功能描述	从上报报文中获取上报类型