OpenTSN网络控制器使用手册

（版本1.0）

OpenTSN开源项目组

2020年12月

目录

[OpenTSN网络控制器使用手册 1](#_Toc63676806)

[**1 引言** 3](#_Toc63676807)

[**2运行环境** 3](#_Toc63676808)

[**3 文件说明** 3](#_Toc63676809)

[3.1 ARP应用 3](#_Toc63676810)

[3.2 通用函数库 4](#_Toc63676811)

[3.3 网络初始化应用 5](#_Toc63676812)

[3.3 状态监测应用 5](#_Toc63676813)

[3.4 PTP时间同步应用 6](#_Toc63676814)

[**4 编译和运行步骤** 6](#_Toc63676815)

[4.1 编译 6](#_Toc63676816)

[4.2 运行 7](#_Toc63676817)

[**5 组网示例** 8](#_Toc63676818)

[5.1组网拓扑 8](#_Toc63676819)

[5.2配置文本 8](#_Toc63676820)

[5.3运行顺序 13](#_Toc63676821)

[**附录一：Libxml2库安装教程** 14](#_Toc63676822)

[**附录二：问题记录** 14](#_Toc63676823)

**1 引言**

本文档为TSN集中控制器使用手册，主要描述控制器运行环境、文件说明、编译和运行步骤、组网示例，用户可以参考该文档使用网络控制器。

网络控制器主要功能包含ARP代理、网络初始配置、状态监测和PTP时间同步，所有功能需要与硬件配合才能实现。

**2运行环境**

网络控制器的运行环境为Linux系统，需要安装以下库支撑程序运行

* 需要安装libpcap和libnet库，用于收包和发包
* 需要安装libxml2库，用于进行xml文件解析（xml库中需要支持连续解析多个xml文本，会在提供的压缩包中包含xml库文件，用户可以进行安装和使用）
* Linux设备需要把网口开启混杂模式

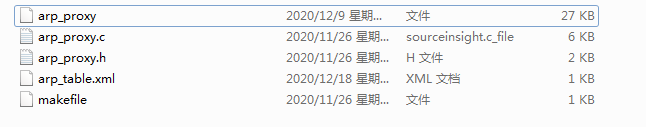
**3 文件说明**

网络控制器包含的源文件如下图所示，arp文件夹下存放的是arp应用程序，cnc\_api文件下存放通用基础库，cnc\_ptp文件下存放PTP时间同步应用程序，net\_init存放网络初始化应用程序，state\_monitor文件夹存放状态监测应用程序。



## 3.1 ARP应用

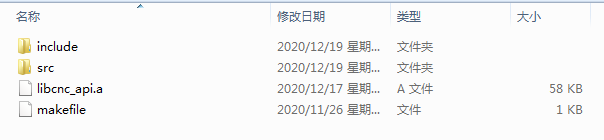
arp文件夹存放ARP代理应用程序，用于响应ARP请求报文，包含的文件如下图所示



* arp\_proxy：进行编译后生成的可执行文件
* arp\_proxy.c：arp代理的源文件，所有的函数在该文件中实现
* arp\_proxy.h：arp代理的头文件，arp数据结构定义在该文件中
* arp\_table.xml：用于存放arp表项
* makefile：可以进行编译的文件

## 3.2 通用函数库

cnc\_api文件夹为基础函数库，包含include文件夹（主要存放通用函数的api头文件）和src文件夹，包含的文件如下图所示



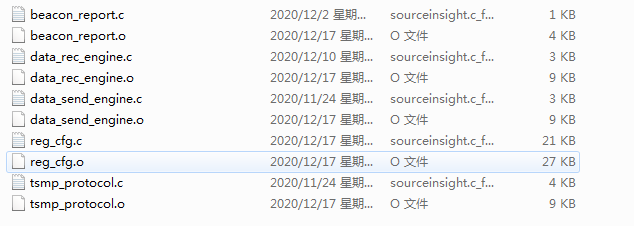
* include：主要存放通用函数的API头文件和通用的数据结构定义
* src：通用API的实现
* libcnc\_api.a：编译基础函数生成的动态库文件
* makefile：可以进行编译的文件

include文件夹中包含的文件



* beacon\_report.h：芯片上报报文的数据结构定义
* cnc\_api.h：通用API的头文件
* reg\_cfg.h：芯片配置和HCP配置报文的数据结构定义
* tsmp\_protocol.h：tsmp协议的数据结构定义

src文件夹中包含的文件



* beacon\_report.c：解析芯片上报报文和HCP上报报文的通用函数实现
* beacon\_report.o：编译beacon\_report.c生成的目标文件
* data\_rec\_engine.c：数据接收通用函数实现
* data\_rec\_engine.o：编译数据接收文件生成的目标文件
* data\_send\_engine.c数据发送通用函数实现
* data\_send\_engine.o：编译数据发送文件生成的目标文件
* reg\_cfg.c：寄存器配置通用函数实现
* reg\_cfg.o：编译寄存器配置文件生成的目标文件
* tsmp\_protocol.c：tsmp协议通用函数实现
* tsmp\_protocol.o: 编译tsmp协议文件生成的目标文件

## 3.3 网络初始化应用

net\_init文件夹存放网络初始化应用程序



* init：编译网络初始化进行生成的可以执行文件
* init\_cfg\_xml.xml：初始配置的xml文本，根据该文本进行网络探测
* makefile：make该文件生成可以执行文件init
* net\_init.c：主函数实现以及解析和进行初始配置的函数实现
* net\_init.h：网络初始配置的头文件
* offline\_plan\_xml.xml：离线规划的xml文本，用于进行离线规划配置
* parse\_offline\_plan\_file.c: 解析离线规划文本各进行离线规划配置的函数实现
* parse\_offline\_plan\_file.h：离线规划配置的头文件。

## 3.3 状态监测应用

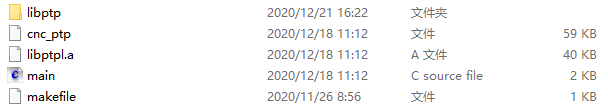
state\_monitor文件夹存放状态监测应用程序



* makefile：make该文件生成可执行文件monitor
* monitor：生成的可执行文件
* state\_monitor.c：状态监测模块的主函数和关键函数的实现
* state\_monitor.h：状态监测的头文件
* topolopy\_info\_xml.xml：拓扑信息的xml文本

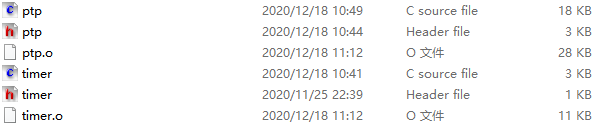
## 3.4 PTP时间同步应用

cnc\_ptp文件夹为PTP时间同步进程的实现



* libptp：主要存放时间同步相关函数的API头文件和数据结构定义以及时间同步相关函数API的实现
* cnc\_ptp：生成的可执行文件
* Libptpl.a：编译PTP时间同步函数生成的动态库文件
* main：时间同步主函数
* makefile：make该文件生成可执行文件cnc\_ptp

libptp文件夹中包含的文件



* Ptp.c：PTP同步处理模块的关键函数实现
* Ptp.h：PTP同步处理的头文件
* Ptp.o：编译时间同步处理文件生成的目标文件
* Timer.c：定时器模块的关键函数实现
* Timer.h：定时器处理的头文件
* Timer.o：编译定时器处理文件生成的目标文件

**4 编译和运行步骤**

## 4.1 编译

* 首先编译cnc\_api文件夹下的库文件，在cnc\_api文件夹下执行

make clean

make

* 编译net\_init文件夹下的网络初始化进程

make clean

make

* 编译state\_monitor文件夹下的状态监测进程

make clean

make

* 编译arp文件夹下的arp代理进程

make clean

make

* 编译cnc\_ptp文件夹下的ptp时间同步代理进程

make clean

make

## 4.2 运行

注：在root权限下执行

* 网络初始化进程（网络接口名为linux网络接口名字，使用ifconfig可以查看）

在net\_init目录下执行：./init 网络接口名

本设备的网络接口名为enp0s17，因此在执行时输入为 ./init enp0s17



* 运行状态监测进程

在state\_monitor目录下执行：./monitor 网络接口名

* 运行arp代理进程

在arp目录下执行：./arp\_proxy 网络接口名

* 运行PTP时间同步进程

在cnc\_ptp目录下执行：./cnc\_ptp

**5 组网示例**

## 5.1组网拓扑

下图为组网示例，用户可以参照该示例搭建网络拓扑。



该网络拓扑中有四个节点，控制器与节点1连接的端口为P2，PC1与节点1连接的端口为P3，节点3与PC2连接的端口为P3。

节点1的P0、P1、P2为协作模式，P3为非协作模式，默认情况下为非协作模式。（端口采用bitmap的形式，每一位都代表一个端口的类型，0代表协作模式，只接收映射过的报文，1代表非协作模式，可以接收所有报文 如上图拓扑所示，1号节点的端口类型为248，二进制为1111 1000，说明P0、P1、P2端口为协作模式，其他端口为非协作模式）；节点2的P0、P1、P2和P3都为协作模式；节点3的P0、P1、P2为协作模式，P3为非协作模式；节点2的P0、P1、P2和P3都为协作模式。

节点1为主时钟，其他节点为从时钟，节点1与节点2时间同步报文路径为节点1--节点2，节点1与节点3的时间同步报文的路径为节点1--节点2--节点3；节点1到节点4的同步报文路径为节点1--节点4。同步的sync报文的flowID为4096。用户也可以修改转发表自定义转发路径。

PC1与PC2通信的数据流映射为BE流，路径为PC1--节点1--节点2--节点3--PC2，流量的flowID为10。

节点的配置顺序必须按照离控制器最近的原则依次进行配置，因此在写xml文本时，必须按照节点的配置顺序编写，先配置的节点写在前面。该示例中配置顺序为节点1--节点2--节点3--节点4。

## 5.2配置文本

arp\_table.xml文本构成如下表所示，该文本为arp代理文本，根据该文本构造arp响应报文。表项中的IP为查表索引，MAC为查表结果（MAC地址为源端向目的端通信的TSNtag，而不是目的端真实MAC地址，最后会在最后一跳查找重映射表将TSNtag还原为真实的MAC地址）。

|  |
| --- |
| <?xml version=**"1.0"** encoding=**"utf-8"**?>  <nodes>  <arp\_table>  <entry IP=**"192.168.1.64"** MAC=**"c0:04:00:00:00:00"**/>  <entry IP=**"192.168.1.68"** MAC=**"c0:04:80:00:00:00"**/>  </arp\_table>  </nodes> |

init\_cfg\_xml.xml文本构成如下表所示，该文本为网络初始配置文本，主要配置端口类型和转发表，配置目的为进行网络探测，是。（节点数目必须与实际的网络拓扑中节点数目相同）

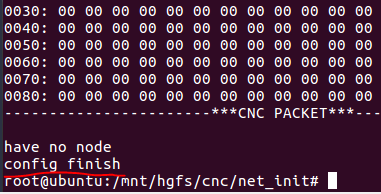
|  |
| --- |
| <?xml version=**"1.0"** encoding=**"utf-8"**?>  <nodes sync\_period=**"100"** master\_imac=**"1"** sync\_flowid=**"4096"**>  <!-- sync\_period表示同步周期，单位为ms，master\_imac表示主时钟的imac地址，sync\_flowid表示sync报文的flowID，在该示例中，同步周期为100ms，主时钟的imac地址为1，sync报文的flowID为4096 -->  <node imac=**"1"** port\_mode=**"248"**><!-- imac=1表示1号节点-->  <!-- port\_mode使用bitmap的形式，每一位都代表一个端口的类型，0代表协作模式，只接收映射过的报文，1代表非协作模式，可以接收所有报文，port\_mode="248"，248二进制为1111 1000，说明P0、P1、P2端口为协作模式，P3、P4、P5、P6、P7端口为非协作模式-->  <entry flowid=**"0"** outport=**"4"**/>  <!-- outport使用bitmap形式，共9bit，从高到低依次代表主机口、7、6、5、4、3、2、1、0号端口。每位的值为1表示向该端口转发，0表示不向该端口转发，如果多个位为1，则向多个端口转发。示例outport=4表示从设备的2号端口转发 -->  <entry flowid=**"1"** outport=**"256"**/>  <!-- 该节点的flowID为1，输出端口为256表示向该节点转发-->  <entry flowid=**"2"** outport=**"1"**/>  <!-- flowID为2的报文，也就是2号节点，输出端口为1表示向0号端口转发-->  <entry flowid=**"3"** outport=**"1"**/>  <!-- flowID为3的报文，也就是3号节点，输出端口为1表示向0号端口转发-->  <entry flowid=**"4"** outport=**"2"**/>  <!-- flowID为4的报文，也就是4号节点，输出端口为2表示向1号端口转发-->  </node>  <node imac=**"2"** port\_mode=**"252"**>  <entry flowid=**"0"** outport=**"2"**/>  <entry flowid=**"1"** outport=**"2"**/>  <entry flowid=**"2"** outport=**"256"**/>  <entry flowid=**"3"** outport=**"1"**/>  <entry flowid=**"4"** outport=**"1"**/>  </node>  <node imac=**"3"** port\_mode=**"248"**>  <entry flowid=**"0"** outport=**"2"**/>  <entry flowid=**"1"** outport=**"2"**/>  <entry flowid=**"2"** outport=**"2"**/>  <entry flowid=**"3"** outport=**"256"**/>  <entry flowid=**"4"** outport=**"1"**/>  </node>  <node imac=**"4"** port\_mode=**"252"**><!-- port\_mode使用bitmap的形式，0代表协作模式，只接收映射过的报文，1代表非协作模式，可以接收所有报文 -->  <entry flowid=**"0"** outport=**"1"**/>  <entry flowid=**"1"** outport=**"1"**/>  <entry flowid=**"2"** outport=**"2"**/>  <entry flowid=**"3"** outport=**"2"**/>  <entry flowid=**"4"** outport=**"256"**/>  </node>  </nodes> |

offline\_plan\_xml.xml文本构成，该文本为离线规划生成的文本（如果没有离线规划器，用户可以自定义简单的流量），该文本中以节点（node）为单位，每个节点中包含gate\_table（门控表目前有4个，使用id进行区分）、register单个寄存器、forward\_table转发表（该转发表包含初始配置的转发表和数据流量的转发表），inject\_table注入时刻表、submit\_table提交时刻表、map\_table映射表、remap\_table重映射表。（节点数目必须与实际的网络拓扑中节点数目相同）

|  |
| --- |
| <?xml version=**"1.0"** encoding=**"utf-8"**?>  <nodes>  <node imac=**"1"**> <!-- 节点的imac为1 -->  <gate\_table id=**"0"**> <!-- 0号端口门控表 -->  <entry time\_slot=**"0"** state=**"254"**/>  <!-- gate-state表示门控状态，使用bitmap形式，共8位，每一位代表一个队列的门控状态，1表示门控打开，0表示关闭。示例gate\_state=254表示7、6、5、4、3、2、1号队列为打开状态，0号队列为关闭状态。0到7号队列分别为TS0、TS1、TS2、RC、PTP、BE、NMAC、重组队列 -->  <entry time\_slot=**"1"** state=**"255"**/>  <entry time\_slot=**"2"** state=**"255"**/>  <entry time\_slot=**"3"** state=**"255"**/>  </gate\_table>    <gate\_table id=**"1"**> <!-- 1号端口门控表 -->  <entry time\_slot=**"0"** state=**"255"**/>  <entry time\_slot=**"1"** state=**"255"**/>  <entry time\_slot=**"2"** state=**"255"**/>  <entry time\_slot=**"3"** state=**"255"**/>  </gate\_table>  <gate\_table id=**"2"**>  <entry time\_slot=**"0"** state=**"255"**/>  <entry time\_slot=**"1"** state=**"255"**/>  <entry time\_slot=**"2"** state=**"255"**/>  <entry time\_slot=**"3"** state=**"255"**/>  </gate\_table>  <gate\_table id=**"3"**>  <entry time\_slot=**"0"** state=**"255"**/>  <entry time\_slot=**"1"** state=**"255"**/>  <entry time\_slot=**"2"** state=**"255"**/>  <entry time\_slot=**"3"** state=**"255"**/>  </gate\_table>  <register>  <time\_slot>**4**</time\_slot><!-- 单位us -->  <inject\_slot\_period>**4**</inject\_slot\_period>  <submit\_slot\_period>**4**</submit\_slot\_period>  <qbv\_or\_qch>**1**</qbv\_or\_qch>  <rc\_regulation\_value>**4**</rc\_regulation\_value>  <be\_regulation\_value>**8**</be\_regulation\_value>  <unmap\_regulation\_value>**4**</unmap\_regulation\_value>  </register>  <forward\_table>  <entry flowid=**"0"** outport=**"4"**/>  <!-- outport使用bitmap形式，共9bit，从高到低依次代表主机口、7、6、5、4、3、2、1、0号端口。每位的值为1表示向该端口转发，0表示不向该端口转发，如果多个位为1，则向多个端口转发。示例outport=4表示从设备的2号端口转发 -->  <entry flowid=**"1"** outport=**"256"**/>  <entry flowid=**"2"** outport=**"1"**/>  <entry flowid=**"3"** outport=**"1"**/>  <entry flowid=**"4"** outport=**"2"**/>  <!-- 转发表上面的部分与init\_cfg.xml文本中相同，下面表示数据流量 -->  <entry flowid=**"10"** outport=**"1"**/>  <entry flowid=**"4096"** outport=**"1"**/>  <!-- imac为2的节点，flowID为10，outport为1，flowID为4096，outport为257-->  <!-- imac为3的节点，flowID为10，outport为256，flowID为4096，outport为256-->  <!-- imac为4的节点，flowID为10，outport为2，flowID为4096，outport为256-->  </forward\_table>    <inject\_table>  <entry time\_slot=**"0"** inject\_addr=**"1"**/>  <!-- 表示0号时间槽应该注入的报文存放在1的地址中-->  <entry time\_slot=**"1"** inject\_addr=**"2"**/>  </inject\_table>  <submit\_table>  <entry time\_slot=**"0"** submit\_addr=**"1"**/>  <!-- 表示0号时间槽应该提交的报文存放在1的地址中-->  <entry time\_slot=**"1"** submit\_addr=**"2"**/>  </submit\_table>  <map\_table>  <entry id=**"0"**>  <src\_ip>**192.168.1.11**</src\_ip>  <dst\_ip>**192.168.1.12**</dst\_ip>  <src\_port>**1111**</src\_port>  <dst\_port>**2222**</dst\_port>  <protocol\_type>**11**</protocol\_type><!-- 第0条映射表，最多有32条映射表，只对TS流进行映射，上面为报文的五元组，索引项，如果匹配中五元组，则查表的结果为下面的值-->  <flow\_type>**4**</flow\_type>  <flow\_id>**111**</flow\_id>  <inject\_addr>**1**</inject\_addr>  <submit\_addr>**2**</submit\_addr>  </entry>  <entry id=**"1"**>  <src\_ip>**192.168.1.13**</src\_ip>  <dst\_ip>**192.168.1.14**</dst\_ip>  <src\_port>**23**</src\_port>  <dst\_port>**45**</dst\_port>  <protocol\_type>**11**</protocol\_type>  <flow\_type>**1**</flow\_type>  <flow\_id>**12**</flow\_id>  <inject\_addr>**1**</inject\_addr>  <submit\_addr>**2**</submit\_addr>  </entry>  </map\_table>  <remap\_table>  <entry id=**"0"**>  <flow\_id>**13**</flow\_id>  <dmac>**33:33:33:33:33:33**</dmac>  <outport>**8**</outport>  <!-- outport使用bitmap形式，1表示向该端口转发，0表示不向该端口转发，如果多个位为1，则向多个端口转发-->  <!-- 重映射表、只在最后一跳起作用，flowID表示查表的索引，dmac和outport为查表的结果，该表项的含义为flowID为13的报文，替换目的mac为33:33:33:33:33:33，并且从3号端口输出-->    </entry>  <entry id=**"1"**>  <flow\_id>**14**</flow\_id>  <dmac>**33:33:33:33:33:33**</dmac>  <outport>**8**</outport>  </entry>  <entry id=**"2"**>  <flow\_id>**15**</flow\_id>  <dmac>**33:33:33:33:33:33**</dmac>  <outport>**8**</outport>  </entry>  </remap\_table>  </node>  <node imac=**"2"**>  ……  <node imac=**"3"**>  ……  <node imac=**"4"**>  ……  </nodes> |

## 5.3运行顺序

* 在net\_init文件夹下执行./init enp0s17（enp0s17表示网络接口名称，进行初始化配置和离线规划配置，出现cfg\_finish表示配置成功）



* 在arp目录下执行./arp\_proxy enp0s17(开启arp代理功能)
* 在cnc\_ptp目录下执行./cnc\_ptp enp0s17（开启时间同步）
* 在state\_monitor目录下执行./monitor enp0s17(进行状态监测)

**附录一：Libxml2库安装教程**

* 拷贝库文件夹到Linux机器中
* 进入libxml2-2.6.2文件夹目录下，在终端中打开
* 使用root权限执行以下命令

make distclean

./configure

make

make install

* 安装完成。

**附录二：问题记录**

* 执行./init enp0s17时，出现没有可执行权限，需要执行chmod 777 init，赋予init文件可执行权限