

时间敏感网络（TSN）环网 组网操作手册

OpenTSN 开源项目组

2020 年 10 月

版本历史

版本	修订时间	修订内容	修订人	文件标识
1.0	2020.10.25	初版编制		OpenTSN
1.1	2020.10.30	代码下载地址更新		

目录

1. 概述	5
2. 组网环境组成	5
3. TSN 组网环境拓扑	5
3.1. 组网设备配置	6
3.2. 组网主要板卡说明	8
3.2.1. 环网 TSN 节点	8
3.2.2. TSNNic 测试仪	9
4. 代码下载	10
4.1. TSNNic 软件	10
4.2. 32 条 ST 流发包软件	11
4.3. TSN 分布式控制器软件	13
4.4. TSN 硬件代码	16
5. 硬件代码新建 TSN 工程	17
6. 硬件代码编译 TSN 项目	19
7. 硬件代码烧录 TSN 工程	20
8. 硬件代码上板调试	22
9. 测试案例	24
9.1. 测试 TSN 环网时间同步精度	24
9.1.1. 预期结果	24
9.1.2. 操作步骤	24
9.1.3. 测试结果	26
9.2. 流量测试	27

9.2.1. 预期结果	27
9.2.2. 操作步骤	28
9.2.3. 测试结果	31
9.3. Ping 报文测试	32
9.3.1. 预期结果	32
9.3.2. 操作步骤	32
9.3.3. 测试结果	33

OpenTSN

1. 概述

本文档为时间敏感网络（下文简称 TSN）环网组网的操作手册。详细介绍了关于 TSN 环网的拓扑组成环境以及组网的操作步骤，而后再通过详细的测试案例来验证了组网的正确性。

2. 组网环境组成

TSN 组网环境搭建包括：TSN 环网中各设备的软件、硬件配置环境，以及 TSN 环网的拓扑环境。

3. TSN 组网环境拓扑

TSN 环网组网环境拓扑如下图 3-1 所示，连线均为网线连接，图中箭头指向即为信号流向参考。

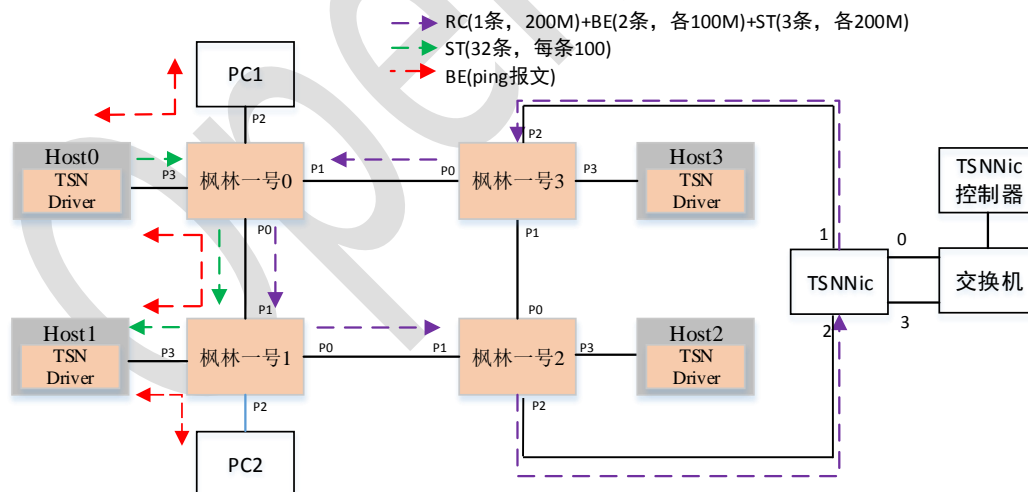


图 3-1 TSN 环网组网环境拓扑

3.1. 组网设备配置

TSN 环网中各设备的配置信息如下表 1 所示。

表 1 设备配置

序号	设备名称	配置信息	备注
1	终端 Host0	1) 连接在枫林一号 0 的 P3 口; 2) ubuntu 操作系统或安装虚拟机的操作系统; 3) 包含可执行 TSN 分布式控制器软件; 4) 包含可执行 32 条 ST 流发包软件。	
2	终端 Host1	1) 连接在枫林一号 1 的 P3 口; 2) ubuntu 操作系统或安装虚拟机的操作系统; 3) 包含可执行 TSN 分布式控制器软件。	
3	终端 Host2	1) 连接在枫林一号 2 的 P3 口; 2) ubuntu 操作系统或安装虚拟机的操作系统; 3) 包含可执行 TSN 分布式控制器软件。	
4	终端 Host3	1) 连接在枫林一号 3 的 P3 口; 2) ubuntu 操作系统或安装虚拟机的操作系统; 3) 包含可执行 TSN 分布式控制器软件。	
5	枫林一号 0	TSN 芯片节点 0, 详细信息见 2.3.1 节	需烧录 TSN 工程 文件
6	枫林一号 1	TSN 芯片节点 1, 详细信息见 2.3.1 节	需烧录 TSN 工程

序号	设备名称	配置信息	备注
			文件
7	枫林一号 2	TSN 芯片节点 2，详细信息见 2.3.1 节	需烧录 TSN 工程 文件
8	枫林一号 3	TSN 芯片节点 3，详细信息见 2.3.1 节	需烧录 TSN 工程 文件
9	PC1	1) Ping 报文测试终端； 2) MAC 配置：B0:83:FE:8B:68:1B； 3) IP 配置：192.168.1.56。	
10	PC2	1) Ping 报文测试终端； 2) MAC 配置：D8:CB:8A:D7:22:9E； 3) IP 配置：192.168.1.55。	
11	TSNNic 控制 器	1) 控制终端 ip：与 192.168.1.1 同网段； 2) 虚拟机 ubuntu； 3) 虚拟机 ip 地址配置为：192.168.1.30。	
12	TSNNic 测试 仪	1) ST/RC/BE 流的发送与接收； 2) 包含可执行的 TSNNic 软件； 3) 与 TSNNic 控制器连接要经过交换机。	

3.2. 组网主要板卡说明

3.2.1. 环网 TSN 节点

如下图 3-2、3-3 所示，是环形 TSN 网络拓扑中的交换节点，其对外接口以及载板的相关接口，均在图中有相应的标注及表格内有说明提示，并且要注意载板上 ID 号的拨码配置。

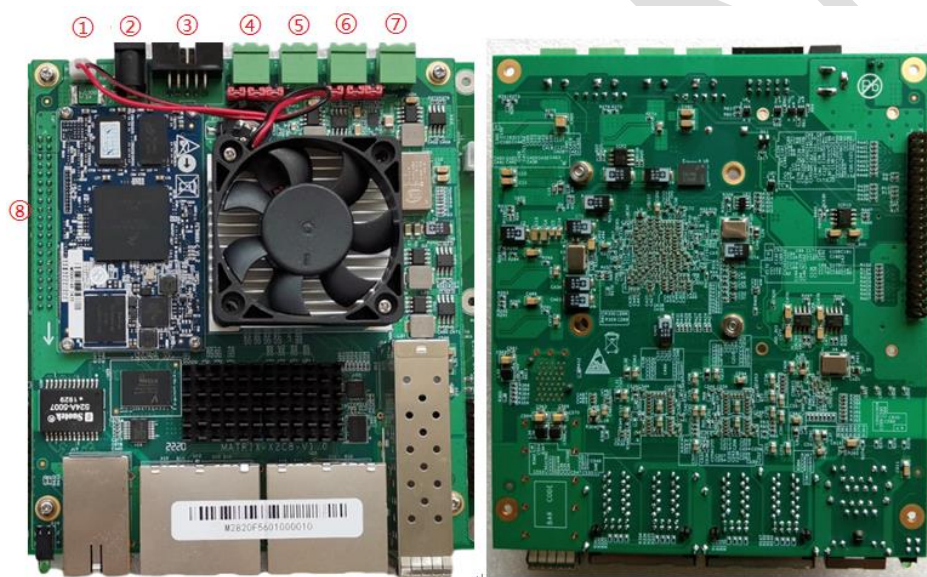


图 3-2 环网节点正反面

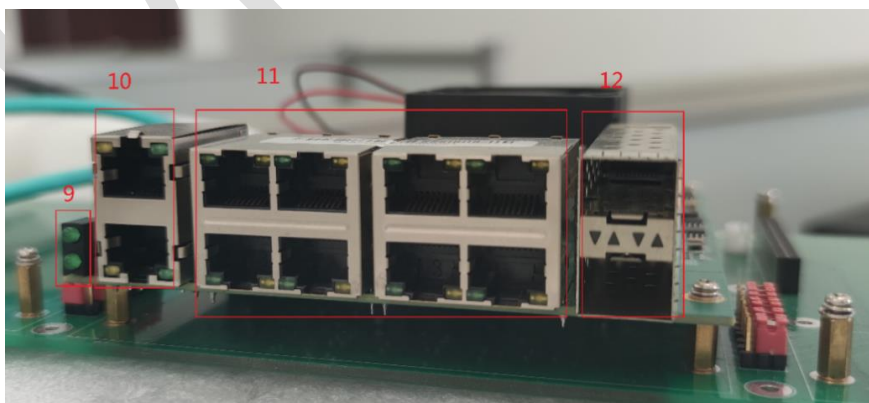


图 3-3 环网节点侧面

上图中标号 1-12 的接口，详细说明如表 2。

表 2 环网芯片节点接口说明

编号	器件位号	器件说明
图 3-1		
1	J11	12V 风扇插座，有防呆设计。
2	J2	12V 电源插座。注意此插座不能与底板上面的 12V 插座同时插上。
3	J3	JTAG 插座，有防呆设计。可用于 JTAG 边扫调试及 ASFlash 烧录。
4-7	J12/J13/J15/J16	RS485 插座，J12/J13 为 RS480 总线 0，
8	J1	2.54mm 网格插针
图 3-2		
9	D14	面板指示灯：上面 PWR(电源)、下面 SYS()
10	J17	ARM 卡接口:上面网口，下面串口
11	U12/U25	正对接口方向，8 个千兆口的对应编号为： 上 1 3 5 7 下 0 2 4 6
12	J14	上层是 SFP+0 号口，下层是 SFP+1 号口。

3.2.2. TSNNic 测试仪

如下图 3-4 所示，测试仪正面有 4 个数据网口（0、1、2、3）、1 个管理网口（MGMT）、1 个复位按钮（RST）以及 4 个 led 灯，根据 TF 卡内存放的软件达到相应的功能，其中 1 口为发送端，2 口为接收端。

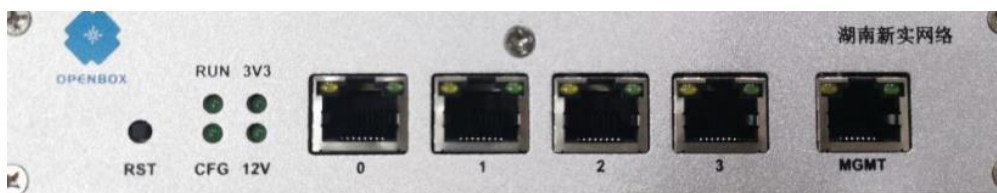


图 3-4 TSNNic 测试仪正面

测试仪背面有 JTAG 接口、USB 接口、COM 串口、开关和电源接口。



图 3-5 TSNNic 测试仪背面

4. 代码下载

4.1. TSNNic 软件

下载码云上 openTSN/bin/TSNNic/硬件/目录下的 BOOT.bin，如图 4-1 所示，下载网址为 <https://gitee.com/opentsn/openTSN2.0/tool>。

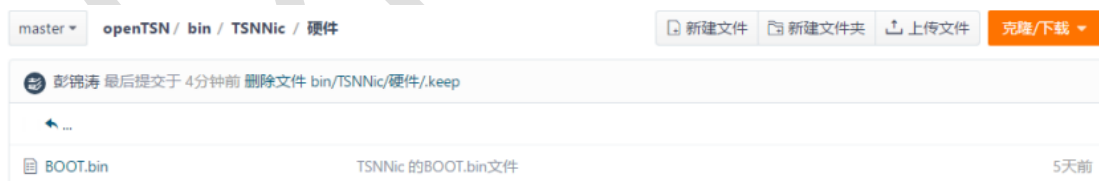


图 4-1 TSNNic 软件下载地址

将 BOOT.bin 拷贝到 openbox_s4 中的 TF 卡的 mnt 目录下，如下图 3-1 所示。

- 1) 步骤一：将 linux 系统设备将 ip 地址设置与 openbox_s4 处于同一网段；
- 2) 步骤二：在 linux 系统中使用 scp 进行拷贝文件到 openbox_s4 中；

```
scp BOOT.bin root@192.168.1.18:/mnt/
```

其中 BOOT.bin 为需要拷贝的文件，root 表示 openbox 中的用户名，192.168.1.18 表示 openbox 中的管理网口的 IP，/mnt/表示拷贝到 openbox 中的/mnt/目录下。

- 3) 步骤三：输入密码“123123”。

4.2. 32 条 ST 流发包软件


名称	修改日期	类型	大小
 send	2020/10/27 21:45	文件	37 KB
 send_32_pkt_fun.c	2020/10/27 21:45	C 文件	67 KB

图 4-2 32 条 ST 流发包软件目录

send 为可执行文件，send_32_pkt_fun.c 为源文件。

该软件仅是一个发包程序，周期性发送 32 条流，每个报文发送 100 次后程序退出，发送的报文格式固定。

该软件的运行环境为 linux 设备，需要安装 libnet 库，运行的步骤如下：

1) 步骤一：使用 ifconfig 查看网口的名称；

```
root@ubuntu:/home/joejiang# ifconfig
enp0s17 Link encap:以太网 硬件地址 08:00:27:1d:a9:23
        inet 地址:192.168.1.30 广播:192.168.1.255 掩码:255.255.255.0
        inet6 地址: fe80::bdd0:b73f:bcb4:ad1/64 Scope:Link
        UP BROADCAST RUNNING MULTICAST MTU:1500 跃点数:1
        接收数据包:6942548 错误:0 丢弃:363 过载:0 帧数:0
        发送数据包:27935 错误:0 丢弃:0 过载:0 载波:0
        碰撞:0 发送队列长度:1000
        接收字节:496975567 (496.9 MB) 发送字节:2774311 (2.7 MB)

lo       Link encap:本地环回
        inet 地址:127.0.0.1 掩码:255.0.0.0
        inet6 地址: ::1/128 Scope:Host
        UP LOOPBACK RUNNING MTU:65536 跃点数:1
        接收数据包:174 错误:0 丢弃:0 过载:0 帧数:0
        发送数据包:174 错误:0 丢弃:0 过载:0 载波:0
        碰撞:0 发送队列长度:1
        接收字节:15472 (15.4 KB) 发送字节:15472 (15.4 KB)
```

图 4-3 使用步骤一

2) 步骤二：使用 root 权限进行编译，执行 ./send 100000 enp0s17；

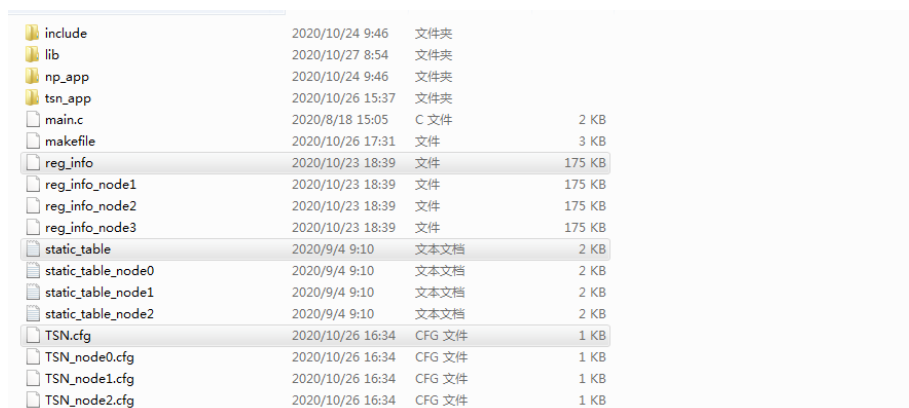
其中 100000 表示发送周期，为 100000us，用户可以自己定义，
enp0s17 是网卡的名称，用户根据需要修改为自己的网卡名称；

```
root@ubuntu:/mnt/hgfs/local_mange/test_hr_tsnsoft/send_fun# gcc send_32_pkt_fun.c -o send
root@ubuntu:/mnt/hgfs/local_mange/test_hr_tsnsoft/send_fun# ./send 100000 enp0s17
count = 1
count = 2
count = 3
count = 4
count = 5
count = 6
count = 7
count = 8
count = 9
count = 10
count = 11
count = 12
count = 13
count = 14
count = 15
count = 16
count = 17
count = 18
count = 19
count = 20
count = 21
count = 22
count = 23
count = 24
count = 25
count = 26
```

图 4-4 使用步骤二

3) 步骤三：程序执行完后自动退出；

4.3. TSN 分布式控制器软件



include	2020/10/24 9:46	文件夹	
lib	2020/10/27 8:54	文件夹	
np_app	2020/10/24 9:46	文件夹	
tsn_app	2020/10/26 15:37	文件夹	
main.c	2020/8/18 15:05	C 文件	2 KB
makefile	2020/10/26 17:31	文件	3 KB
reg_info	2020/10/23 18:39	文件	175 KB
reg_info_node1	2020/10/23 18:39	文件	175 KB
reg_info_node2	2020/10/23 18:39	文件	175 KB
reg_info_node3	2020/10/23 18:39	文件	175 KB
static_table	2020/9/4 9:10	文本文档	2 KB
static_table_node0	2020/9/4 9:10	文本文档	2 KB
static_table_node1	2020/9/4 9:10	文本文档	2 KB
static_table_node2	2020/9/4 9:10	文本文档	2 KB
TSN.cfg	2020/10/26 16:34	CFG 文件	1 KB
TSN_node0.cfg	2020/10/26 16:34	CFG 文件	1 KB
TSN_node1.cfg	2020/10/26 16:34	CFG 文件	1 KB
TSN_node2.cfg	2020/10/26 16:34	CFG 文件	1 KB

图 4-5 软件目录

该文件的目录结构如上图所示：

- 1) include 文件夹存放的是公共头文件；
- 2) lib 文件夹存放的是与平台相关的文件，编译生成与平台相关的库文件；
- 3) np_app 文件夹存放是 np 芯片的软件（没有使用到）；
- 4) tsn_app 文件夹存放的是 TSN 控制器主要的应用；
- 5) main.c 为主函数；
- 6) reg_info 硬件需要配置的寄存器和表项，用户可以根据自己需要更改该文件，用于对硬件进行不同的配置，具体的含义如下图所示，为了便于操作，存放四个 reg_info_xx 文件，分别为四个节点的硬件配置文件，但程序只会读取 reg_info，因此需要把后缀去掉，默认为 0 号节点。例如需要运行一号节点时，

则需要把目前的 reg_info 重命名为 reg_info_node0，把 reg_info_node1 重命名为 reg_info；

```

type: register // 类型为单个寄存器，每个[]中只有一个类型
inject_slot_period: 100 // 注入周期
submit_slot_period: 100 // 提交周期
slot_length: 6 // 时间槽长度 us
port_type: // 端口类型，0代表文接口，1代表终端口
qbw_out_qos: // 工作模式，0代表qbw，1代表qwb
report_type: // 上报类型
report_en: // 上报使能，0代表不上报，1代表上报
report_period: // 上报的周期，只能为1 (ms) ~ 1000 (ms)
rc_regulation_value: 100 // rc流阈值，bufid小于100时丢弃
be_regulation_value: 150 // be流阈值，bufid小于150时丢弃
unmap_regulation_value: 100 // 未映射流阈值，bufid小于100时丢弃
}

type: forward_info // 类型为转发表
0 // 空槽和固定flowid，空槽用0代表输出端口号，输出端口采用bitmap的形式，该内容表示flowid为0的报文从0号端口转发
1 // flowid为1的报文从0号和1号端口转发
2 256 // flowid为2的报文从2号端口（主机口）转发
}

type: inject_time_table // 类型为注入时间表
1 0 // 前两位表示该表项有效，0表示无效，1表示有效；中间的2代表注入地址，最后一个0代表注入时间槽，该内容表示该表项有效，注入地址为2的报文在第0个时间槽注入
1 16 // 该内容表示该表项有效，注入地址为16的报文在第1个时间槽注入
1 32 // 该内容表示该表项有效，注入地址为32的报文在第2个时间槽注入
0 48 // 该内容表示该表项无效
}

type: submit_time_table // 类型为提交时间表
1 0 // 与注入时间表类似，该内容表示该表项有效，提交地址为2的报文在第0个时间槽提交
1 16 // 该内容表示该表项有效，提交地址为16的报文在第1个时间槽提交
1 32 // 该内容表示该表项有效，提交地址为32的报文在第2个时间槽提交
0 48 // 该内容表示该表项无效
}

type: gate_out // 类型为门控表
depth: 0 // 门控表配置的深度为0
gate_out_0: fc fa f9 f8 fc fa f9 f8 // 0号端口门控，每个端口有8个队列，fc为16进制，表示0和1号队列关闭，其他所有队列打开，fa表示0号2号队列关闭，其他队列打开，
// f9表示1和2号队列关闭，其他队列打开，f8表示0、1和2号队列关闭，其他队列打开
gate_out_1: fc fa f9 f8 fc fa f9 f8 // 1号端口门控
gate_out_2: fc fa f9 f8 fc fa f9 f8 // 2号端口门控
gate_out_3: fc fa f9 f8 fc fa f9 f8 // 3号端口门控
gate_out_4: fc fa f9 f8 fc fa f9 f8 // 4号端口门控
gate_out_5: fc fa f9 f8 fc fa f9 f8 // 5号端口门控
gate_out_6: fc fa f9 f8 fc fa f9 f8 // 6号端口门控
gate_out_7: fc fa f9 f8 fc fa f9 f8 // 7号端口门控

```

图 4-6 reg_info 文件

7) static_table.txt 文件为流映射需要的文件，具体含义如下图所示，与 reg_info 文件相同，也存放四个文件 static_table_xx.txt 文件，分别对应四个节点，默认为 0 号节点，但程序只会读取 static_table.txt，因此需要把后缀去掉。例如需要运行一号节点时，则需要把目前的 static_table.txt 重命名为 static_table_node0.txt，把 static_table_node1.txt 重命名为 static_table.txt；

```

1 {
2   ID:0//表项ID
3   src_ip:c0a80105//源端IP地址
4   dst_ip:c0a80104//目的端IP地址
5   src_port:1388//源端口
6   dst_port:1770//目的端口
7   protocol_type:11//ip协议类型
8   flow_type:0//TSNtag中的报文类型
9   flow_id_TSNtag:0005//静态流量使用flowID, 每条静态流分配一个唯一flowID
10  seq_id:0000 //用于标识每条流中报文的序列号
11  frag_flag:0 //用于标识分片后的报文头尾
12  frag_id:0//用于表示当前分片报文在原报文中的分片序号
13  inject_addr:01//注入地址
14  submit_addr:0//提交地址
15  pkttype:0//metadata中的报文类型
16  md_inject_addr:01//metadata中的注入地址
17  outport:0000//输出端口号
18  lookup_en:1//查表使能
19  frag_last:0//用于标识分片后的报文头尾, 当是报文的最后一片时, 置为1
20  reserve:0//预留位
21 }

```

图 4-7 static_table.txt 文件

- 8) TSN.cfg 文件为软件的一些基础配置文件，具体含义如下所示，与 reg_info 文件相同，也存放四个文件 TSN_xx.cfg 文件，分别对应四个节点，默认为 0 号节点，但程序只会读取 TSN.cfg，因此需要把后缀去掉。例如需要运行一号节点时，则需要把目前的 TSN.cfg 重命名为 TSN_node0.cfg，把 TSN_node1.cfg 重命名为 TSN.cfg。

```

Multicast_IMAC=4096 //时间同步报文sync组播地址
HOST_IMAC=2 //本地的imac地址,
M_s_Flag=0 //时间同步主从配置, 0代表从, 1代表主
net_interface=enp0s17 //网络的名称, 用于libpcap抓包时监听的网口,
endpoint_port=2 //终端连接的端口号
LOCAL_MAC1=b0:83:fe:8b:68:1b//终端1的目的mac地址
LOCAL_IP1=192.168.1.56 //终端1的目的ip地址
//如果ARP请求报文的ip为192.168.1.56, 则回复的响应报文的源mac地址为b0:83:fe:8b:68:1b
LOCAL_MAC2=00:00:00:18:60:92//终端2的目的mac地址
LOCAL_IP2=192.168.1.12 //终端2的目的ip地址
LOCAL_MAC3=00:00:00:18:60:93//终端3的目的mac地址
LOCAL_IP3=192.168.1.13 //终端3的目的ip地址
LOCAL_MAC4=00:00:00:18:60:94//终端4的目的mac地址
LOCAL_IP4=192.168.1.14 //终端4的目的ip地址

```

图 4-8 TSN.cfg 文件

该软件的运行环境为 linux 设备，需要安装 libnet 和 libpcap，运行的步骤如下：

- 1) 步骤一：使用 ifconfig 查看网口的名称；修改 TSN.cfg 文件中 net_interface 选项的值为 enp0s17；

```
root@ubuntu:/home/joejiang# ifconfig
enp0s17 Link encap:以太网 硬件地址 08:00:27:1d:a9:23
        inet 地址:192.168.1.30 广播:192.168.1.255 掩码:255.255.255.0
        inet6 地址: fe80::bdd0:b73f:bc4:ad1/64 Scope:Link
        UP BROADCAST RUNNING MULTICAST MTU:1500 跃点数:1
        接收数据包:6942548 错误:0 丢弃:363 过载:0 帧数:0
        发送数据包:27935 错误:0 丢弃:0 过载:0 载波:0
        碰撞:0 发送队列长度:1000
        接收字节:496975567 (496.9 MB) 发送字节:2774311 (2.7 MB)

lo      Link encap:本地环回
        inet 地址:127.0.0.1 掩码:255.0.0.0
        inet6 地址: ::1/128 Scope:Host
        UP LOOPBACK RUNNING MTU:65536 跃点数:1
        接收数据包:174 错误:0 丢弃:0 过载:0 帧数:0
        发送数据包:174 错误:0 丢弃:0 过载:0 载波:0
        碰撞:0 发送队列长度:1
        接收字节:15472 (15.4 KB) 发送字节:15472 (15.4 KB)
```

图 4-9 操作步骤一

- 2) 步骤二：使用 root 权限进行编译，执行 make TYPE=TSN_FPGA；

```
root@ubuntu:/mnt/hgfs/local_mange/test_hr_tsnsoft/tsnsoft20200825_final_20200903_2_NP_上传码云# make TYPE=TSN_FPGA
gcc -g -o ./lib_src/buf.o -c ./lib_src/buf.c -ln -lpthread -lpcap -lnet -I ./include
gcc -g -o ./lib_src/quene.o -c ./lib_src/quene.c -ln -lpthread -lpcap -lnet -I ./include
gcc -g -o ./lib_src/msg.o -c ./lib_src/msg.c -ln -lpthread -lpcap -lnet -I ./include
gcc -g -o ./lib_src/data_rec_engine.o -c ./lib_src/tsn_data_rec_engine.c -ln -lpthread -lpcap -lnet -I ./include
gcc -g -o ./lib_src/data_send_engine.o -c ./lib_src/tsn_data_send_engine.c -ln -lpthread -lpcap -lnet -I ./include
gcc -g -o ./lib_src/timer.o -c ./tsn_app/timer/timer.c -ln -lpthread -lpcap -lnet -I ./include
gcc -g -o ./lib_src/basic.o -c ./lib_src/basic.c -ln -lpthread -lpcap -lnet -I ./include
gcc -g -o ./lib_src/schedule.o -c ./lib_src/schedule.c -ln -lpthread -lpcap -lnet -I ./include
gcc -g -o ./lib_src/tools.o -c ./lib_src/tools.c -ln -lpthread -lpcap -lnet -I ./include
gcc -g -o ./lib_src/table_operations.o -c ./lib_src/table_operations.c -ln -lpthread -lpcap -lnet -I ./include
#gcc -g -o ./lib_src/timer.o -c ./tsn_app/timer/hw_timer.c -ln -lpthread -lpcap -lnet -I ./include
ar -rc libhxl.a ./lib_src/buf.o ./lib_src/quene.o ./lib_src/msg.o ./lib_src/data_rec_engine.o ./lib_src/data_send_engine.o ./lib_src/ba
s.c.o ./lib_src/
schedule.o ./lib_src/tools.o ./lib_src/timer.o ./lib_src/table_operations.o
makefile:64: "the value is TSN_FPGA"
gcc -g -D TSN_FPGA -o local_mange.o -c ./tsn_app/local_mange/local_mange.c -ln -lpthread -lpcap -lnet -I ./include #local_mange
gcc -g -D TSN_FPGA -o regroup.o -c ./tsn_app/pkt_regroup/regroup.c -ln -lpthread -lpcap -lnet -I ./include #regroup
gcc -g -D TSN_FPGA -o ptp.o -c ./tsn_app/ptp/ptp.c -ln -lpthread -lpcap -lnet -I ./include #ptp
gcc -g -D TSN_FPGA -o map_service.o -c ./tsn_app/flow_map/map_service.c -ln -lpthread -lpcap -lnet -I ./include
gcc -g -o arp_reply.o -c ./tsn_app/arp_reply/arp_reply.c -ln -lpthread -lpcap -lnet -I ./include
ar -rc libtsn.a local_mange.o regroup.o ptp.o map_service.o arp_reply.o
gcc -g -D TSN_FPGA -o hx main.c -I ./libhxl -I ./libtsn -ln -lpthread -lpcap -lnet -I ./include
root@ubuntu:/mnt/hgfs/local_mange/test_hr_tsnsoft/tsnsoft20200825_final_20200903_2_NP_上传码云#
```

图 4-10 操作步骤二

- 3) 步骤三：完成步骤二后会生成 hx 的可执行文件，执行 ./hx。

4.4. TSN 硬件代码

下载码云上 [openTSN2.0/src](https://gitee.com/opentsn/openTSN2.0/src) 目录下的 soft_code，如图 5-2 所示，

下载网址为 https://gitee.com/opentsn/openTSN2.0/src/soft_code。

5. 硬件代码新建 TSN 工程

新建工程的具体操作步骤如下：

- 1) 打开 quartus （版本要求是不低于 **2018.1** 版本）， 点击 file->NewProjectWizard， 选择好路径以及芯片， 如下图 5-2 所示。

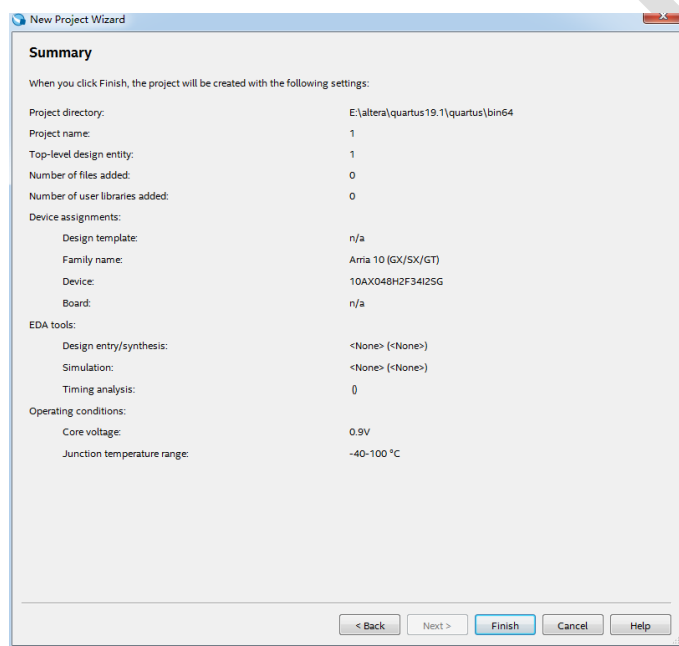


图 5-1 新建 TSN 项目文件

- 2) 打开 setting 面板，将下载的 TSN 硬件代码以及需要的 IP 核文件添加到工程，并设置工程顶层。

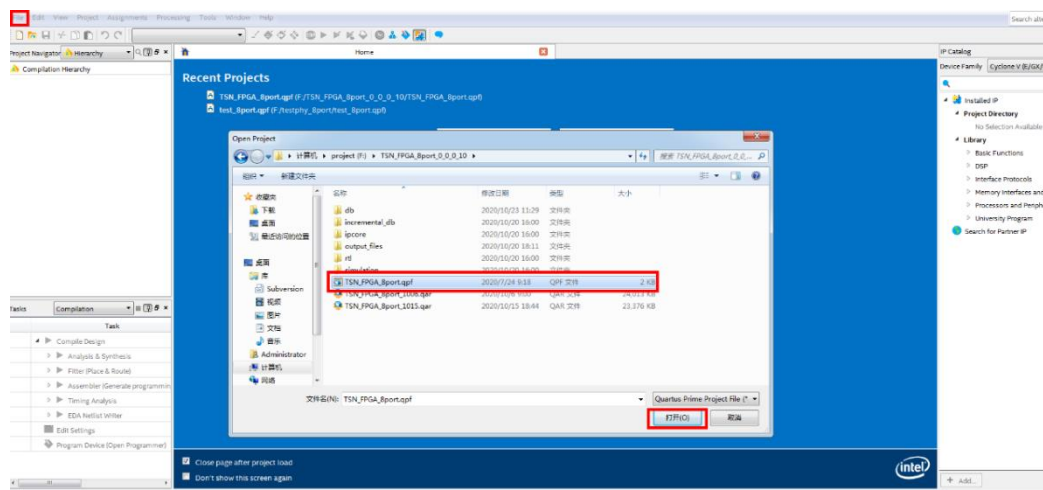


图 5-2 打开 TSN 项目文件

- 2) 完全打开 TSN 项目后，点击工具栏的编译按钮，进行工程的完整编译。如下图 6-3 所示。

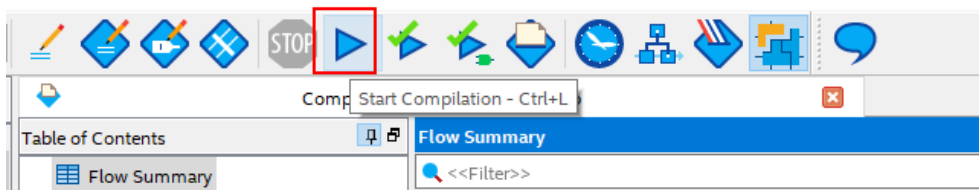


图 6-3 完整编译 TSN 工程

- 3) 编译不报错，并且所有警告都熟知且警告不影响调试，查看 RTL 图看编译后的综合情况，没问题就可以将代码烧录到芯片内部，进行上板调试。

7. 硬件代码烧录 TSN 工程

TSN 工程烧录的具体操作步骤如下：

- 1) 使用烧录线依次连接枫林一号 0--枫林一号 3 开发板，4 个模块逐一烧录。

- 2) 点击 tools->programmer->addfiles, (如果需要固化程序, 可生成.jic 文件) 添加编译完成的 TSN_FPGA_8port.sof 文件。如下图 7-4 所示。

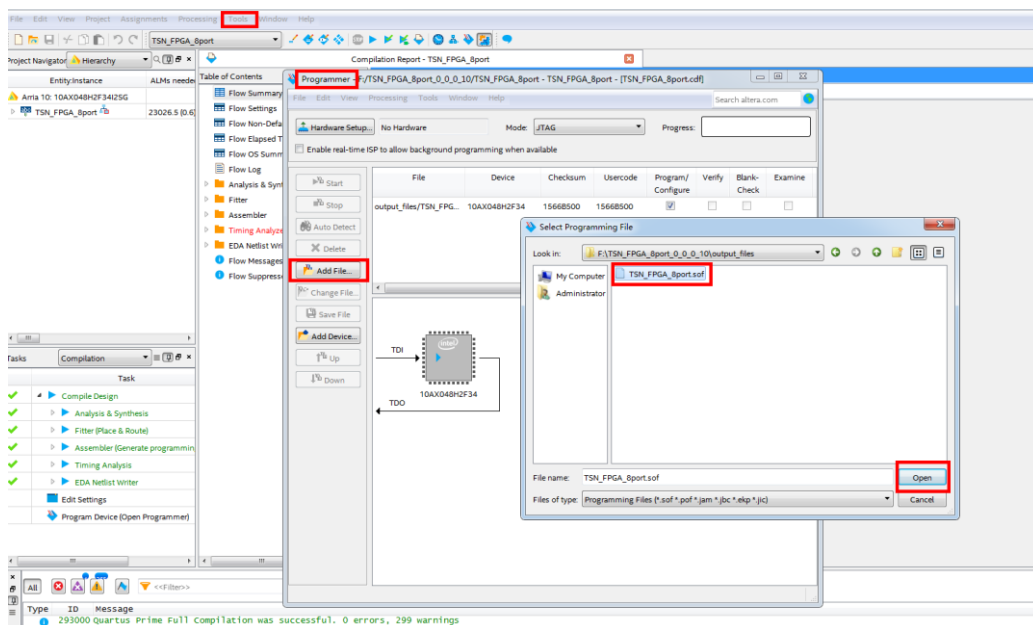


图 7-4 添加.sof/jic 文件

- 3) 选择烧录线的 USB 串口, 并选择 JTAG 模式烧录, 点击 start 开始烧录 TSN 工程。如下图 7-5 所示。

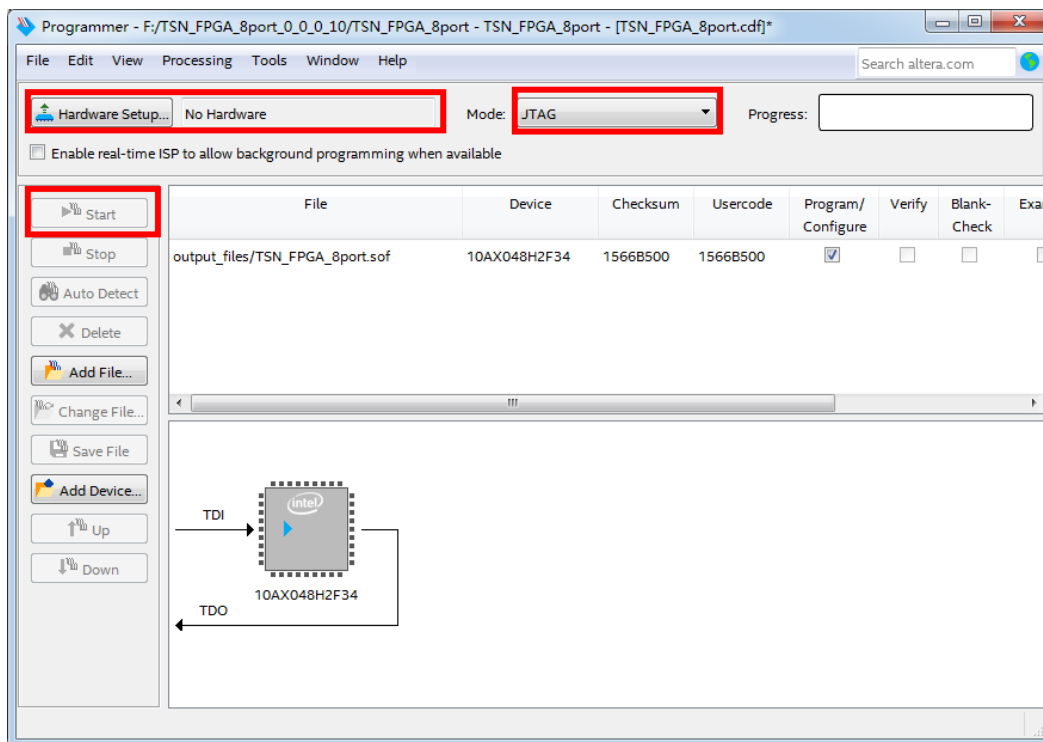


图 7-5 烧录 TSN 工程

8. 硬件代码上板调试

上板调试的大致操作步骤如下：

- 1) 点击 tools->signal tap logic Analyzer, 在触发信号栏, 选择需要调试的信号, 设置触发参数。如下图 8-6 所示。

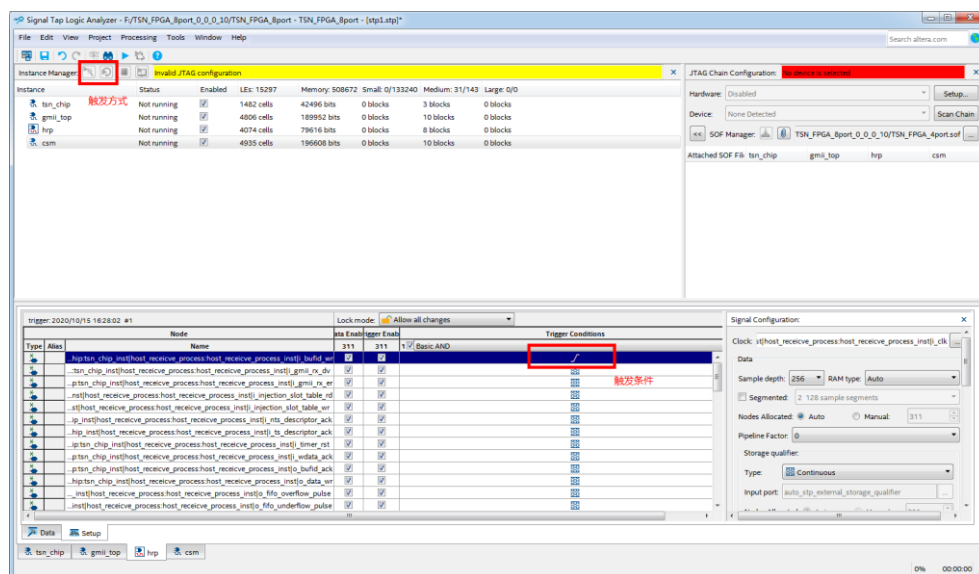


图 8-6 设置触发参数

- 2) 选择单步触发或连续触发，可以看到调试信号的具体数据。如下图所示 8-7 所示。

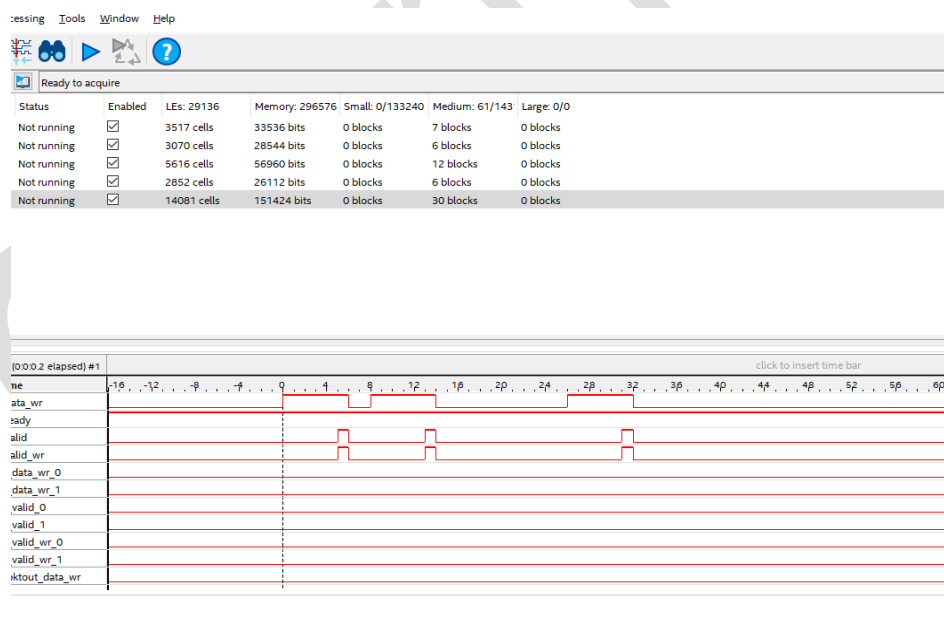


图 8-7 调试信号的具体数据

9. 测试案例

9.1. 测试 TSN 环网时间同步精度

9.1.1. 预期结果

枫林一号 3 芯片为主节点，其他三块芯片为从节点，记录三个从节点的 offset 值。在运行时间同步后，各个节点的终端界面上会将同步节点实时上报的同步信息参数 offset 值显示打印，offset 参数值应保持在在 50ns 以内。

9.1.2. 操作步骤

- 3) 搭建好 TSN 环网拓扑环境，按照 32 节配置好设备。
- 4) 确认各个芯片的配置表项，具体配置如表 3；

表 3 芯片的配置表项

表项	枫林一号 host3		枫林一号 host0	
	Flow_id	Outport	Flow_id	Outport
sync	4096	3	4096	257
req	0	256	0	2
resq1	1	1（64）	1	256
resq2	2	1（64）	2	1（64）
resq3	3	2（128）	3	0
ping 请求	98	1	98	256
ping 应答	99	256	99	2

Host0 发出 ST0-ST31	14'dx	0	14'dx	256
NIC 注入流量 ST32	14'd200	4	14'd200	1
ST33	14'd3000	4	14'd3000	1
ST34	14'd6000	4	14'd6000	1
RC1	14'd9000	4	14'd9000	1
BE1	14'd12000	4	14'd12000	1
	枫林一号 host1		枫林一号 host2	
	Flow_id	Outport	Flow_id	Outport
sync	4096	256	4096	256
req	0	2	0	1
resq1	1	0	1	0
resq2	2	256	2	0
resq3	3	0	3	256
ping 请求	98	1	98	0
ping 应答	99	1	99	0
Host0 发出 ST0-ST31	14'dx	1	14'dx	1
NIC 注入流量 ST32	14'd200	4	14'd200	1
ST33	14'd3000	4	14'd3000	1
ST34	14'd6000	4	14'd6000	1
RC1	14'd9000	4	14'd9000	1
BE1	14'd12000	4	14'd12000	1
说明：Host0 发出 ST 流量的 flowID 为 100、512、1024、1536、2048、2560、3072、3584、4095、4608、5120、5632、6144、6656、7168、7680、8192、8704、9216、9728、10240、10752、11264、11776、12288、12800、13312、13824、14336、14848、15360、15872、除红色部分				

外，其他都为 512 的倍数。

- 5) 在终端 host0 的操作系统中，在 linux 界面利用命令“sudo su”获取权限，输入密码“123123”，再编辑命令进入到配置文件目录下，例如“cd/桌面/tsnsoft_final_20200903”，表项配置打开文件 reg_info 查看修改即可，确认表项无误保存退出，然后在目录下运行命令“./hx”，终端 host1、2、3 与终端 host0 操作类似，不再赘述（TSN 分布式控制器软件具体参看 3.3 节）；
- 6) 当四个终端的命令“./hx”都执行后，即可在各终端查看 offset 参数值的情况。

9.1.3. 测试结果

```

root@tsnswen-OptiPlex-3620:/home/tsnswen/桌面/tsnsoft20200903_Final_20200903_0
0000: 00 0f 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0010: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0020: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0030: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0040: 77 CE 00 00 00 00 00 00
*****PACKET*****
22221111 Import = 1
Count = 8
Packet Addr: 0x7fa1b0f08176
0 1 2 3 4 5 6 7 8 9 A B C D E F
0000: 00 0f 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0010: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0020: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0030: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0040: 00 3C 00 00 00 00 00 00
*****PACKET*****
22221111 Import = 1
Flag = 1 offset = 14
Count = 8
Packet Addr: 0x7fa1b0e00000
0 1 2 3 4 5 6 7 8 9 A B C D E F
0000: 00 0f 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0010: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0020: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0030: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0040: 00 24 00 00 00 00 00 00
*****PACKET*****
22221111 Import = 1
Count = 8
Packet Addr: 0x7fa1b0e00000
0 1 2 3 4 5 6 7 8 9 A B C D E F
0000: 00 0f 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0010: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0020: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0030: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0040: 00 19 00 00 00 00 00 00
*****PACKET*****
22221111 Import = 1
Count = 8
Packet Addr: 0x7fa1b0e00000
0 1 2 3 4 5 6 7 8 9 A B C D E F
0000: 00 0f 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0010: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0020: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0030: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0040: 00 00 00 00 00 00 00 00
*****PACKET*****
22221111 Import = 9
Flag = 0 offset = 29

```

图 9-1 枫林一号 0 同步结果

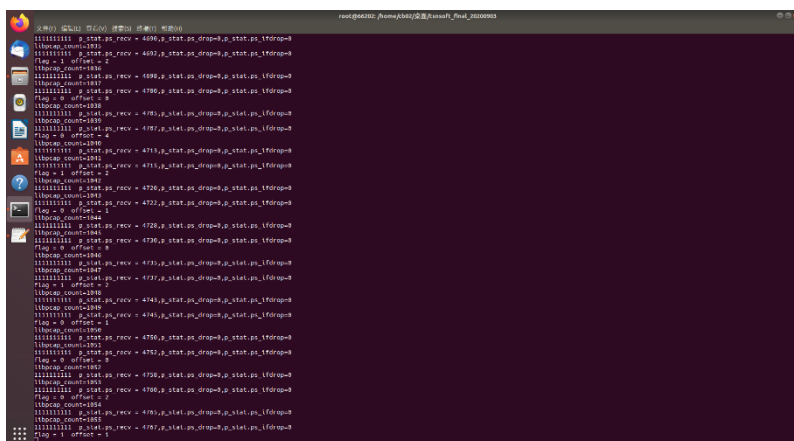


图 9-2 枫林一号 1 同步结果

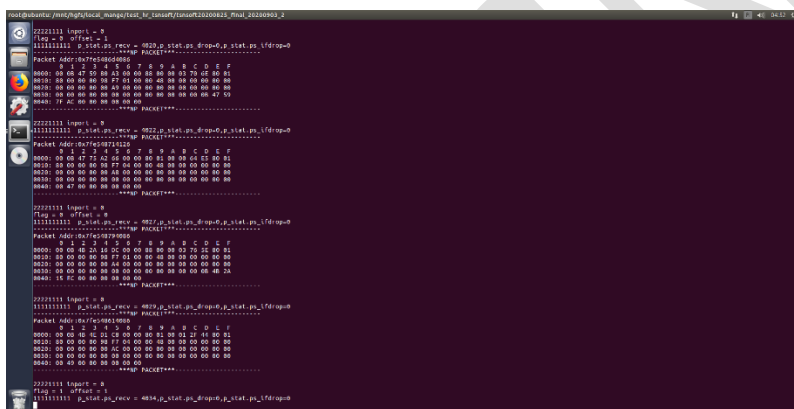


图 9-3 枫林一号 2 同步结果

从上图中看三个从节点的 `offset` 参数值结果均在 50ns 以内，符合预期。

9.2. 流量测试

9.2.1. 预期结果

在运行 TSN 同步后，利用 TSNNic 测试仪注入 1 条 200M 的 RC 流，1 条 100M 的 BE 流与 3 条各 200M 已映射的 ST 流，同时终端 host0 往终端 host1 发送 32 条 ST 流，每条流 100 个，在接收端要保

证 ST 流不存在丢包，且 timestamp_count 计数器为 3200，需要注意的是测试仪的注入流和 32 条流需同时注入。

9.2.2. 操作步骤

- 1) 按 TSN 环网时间同步测试案例配置表项，启动时间同步程序；
- 2) 测试仪控制终端进入虚拟机，在 linux 界面使用命令“sudo su”命令获取权限，然后进入到 TF 卡的 /mnt 目录下，再用命令“cd hgfs/nic/tsn_nic”，然后使用命令“./tester_ui”就可以到测试仪配置面板了；

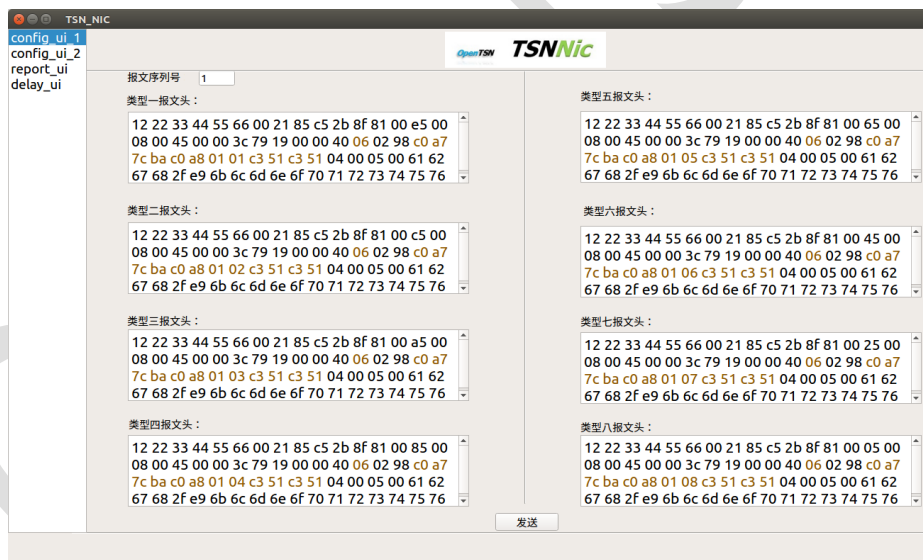


图 9-4 测试仪配置面板

- 3) Config_ui_1 配置界面配置：这个界面配置注入流的特征，主要是类型与 flow id 的配置，报文头前 6 个字节对应分类映射结果

TSNtag 的格式。即 5 条流配置完成后如下，配置完成后点击下方“发送”：

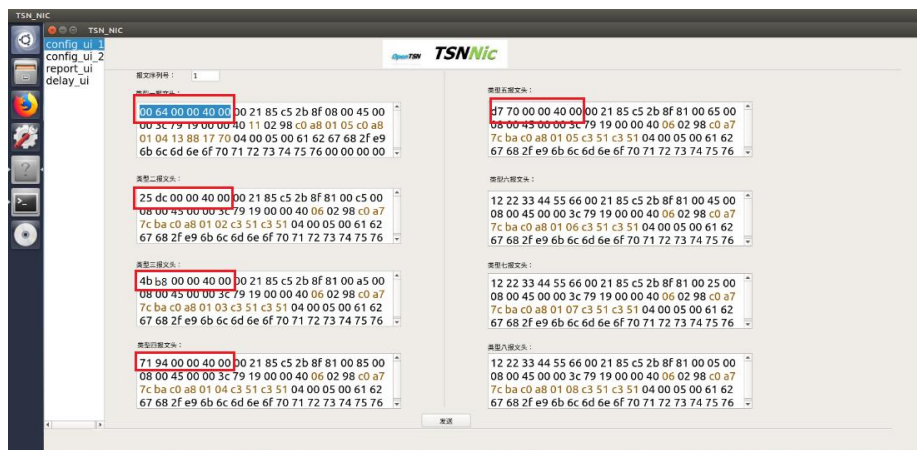


图 9-5 Config_ui_1 配置完成界面

4) Config_ui_2 配置界面配置：这个界面配置注入流的报文长度、速率、时间槽以及五元组等信息，长度 64—1518 字节，可自行设置不同长度进行测试；

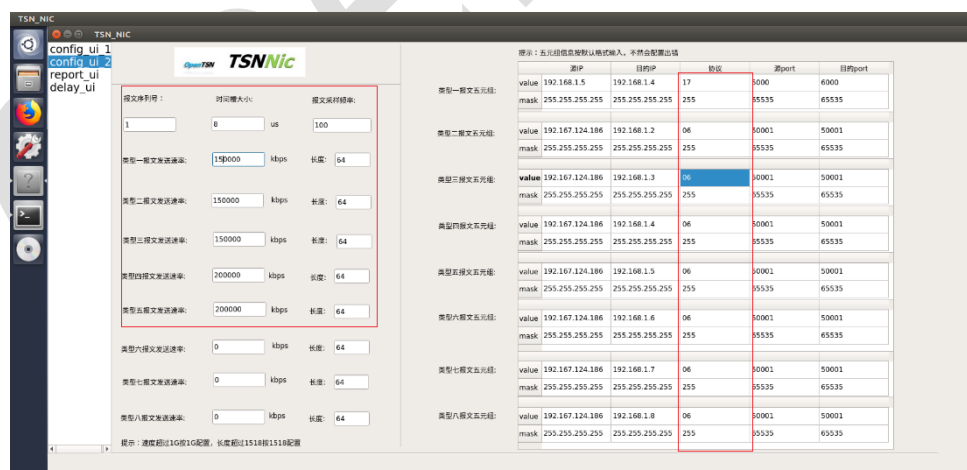


图 9-6 Config_ui_2 配置完成界面

- 5) Report_ui 测试界面：点击开始测试即可，需要查看丢包情况，点击停止测试即可。

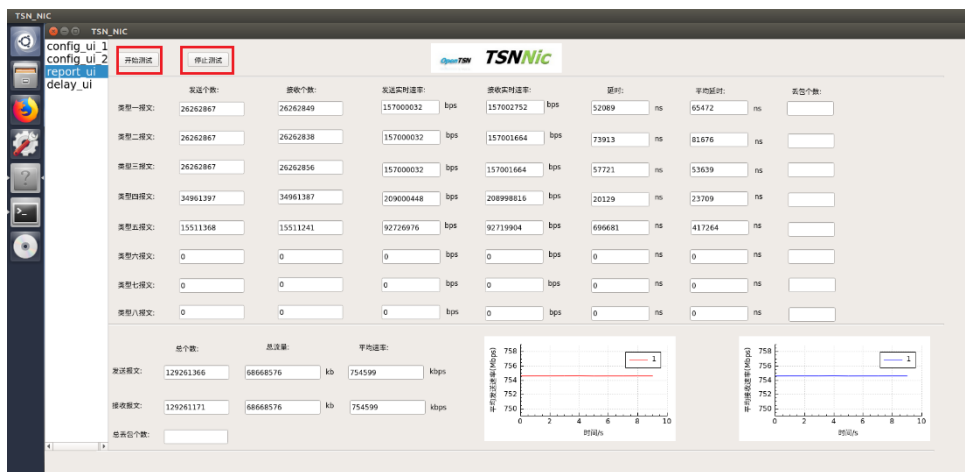


图 9-7 Report_ui 测试界面

- 6) 配置 32 条 ST 流：在终端 host0 的 linux 操作界面，进入到软件目录下，利用命令“gcc send_32_pkt_fun.c”进入流的配置；
- 7) 将注入和提交时间槽周期改为 1024，时间槽位 4us，0 号从机发生 32 条 ts 流，每条流发送 100；
- 8) 保存配置之后，运行命令“./a.out”即可进行测试。

9.2.3. 测试结果

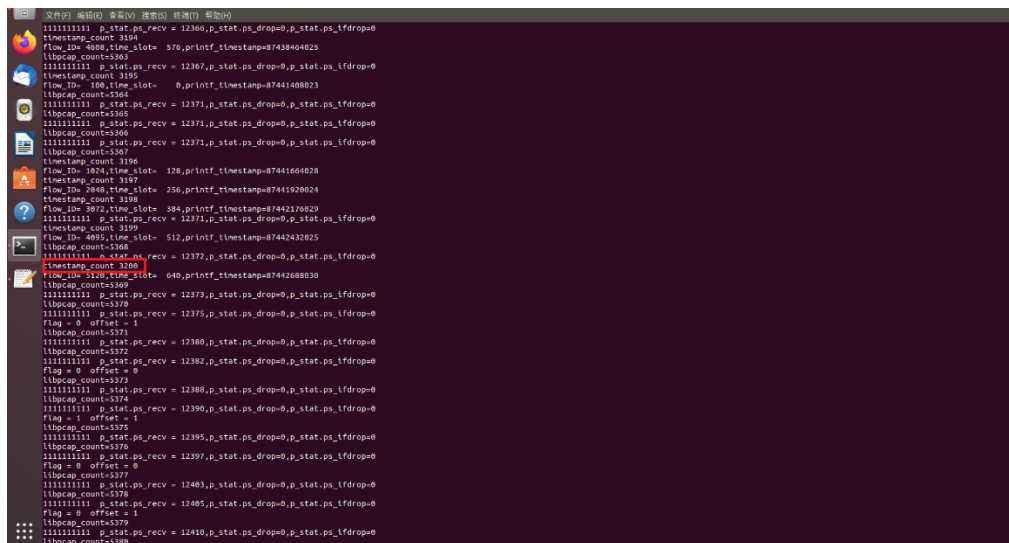


图 9-8 32 条流测试结果

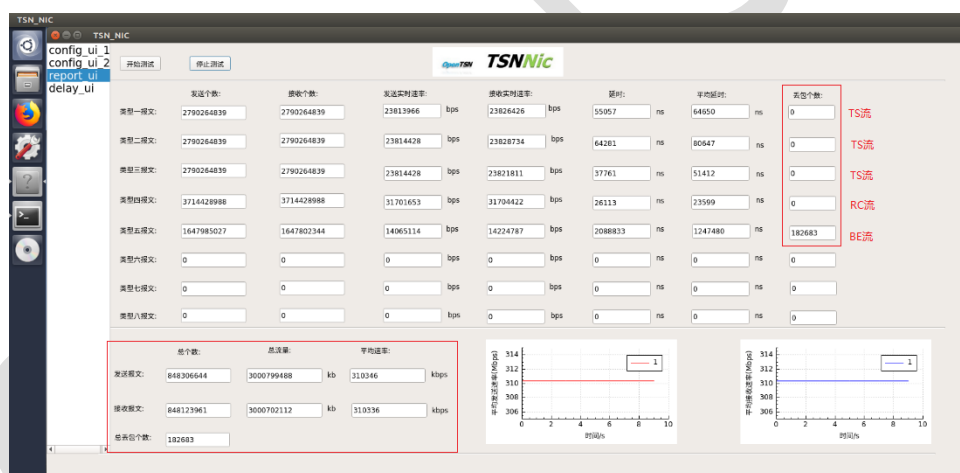


图 9-9 TSNNic 测试仪注入流测试结果

在 32 条 TS 流测试完成后，可以在 TSNNic 测试仪的 report_ui 界面点击停止测试，即可查看 nic 注入流的测试结果，从图 8-9 中看出，只存在 BE 流的丢包情况，符合预期

9.3. Ping 报文测试

9.3.1. 预期结果

选取字节长度为 200、700、1472 的报文进行测试，在 nic 发送超过 1G 的流量，可以看到 ping 报文会有请求超时发送，之后会恢复正常，因为把 ping 的报文映射为 BE 流，当速率大于 1G 时，会丢弃 BE 报文。

9.3.2. 操作步骤

- 1) 搭建好 TSN 环网拓扑环境，按照 2.2 节配置设备环境。
- 2) 在 PC1/PC2 端（注意 pc 端的 MAC 及 IP 配置）的 ping 报文界面，使用命令“ping 192.168.1.56 -t -l 700”，最后字符为报文长度

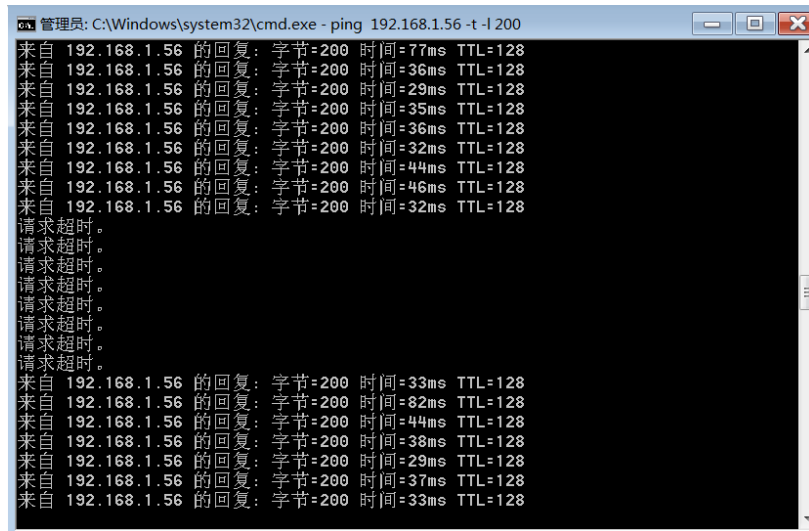


```
C:\Users\Administrator>ping 192.168.1.56 -t -l 700
正在 Ping 192.168.1.56 具有 700 字节的数据:
来自 192.168.1.56 的回复: 字节=700 时间=22ms TTL=128
来自 192.168.1.56 的回复: 字节=700 时间=25ms TTL=128
来自 192.168.1.56 的回复: 字节=700 时间=38ms TTL=128
来自 192.168.1.56 的回复: 字节=700 时间=25ms TTL=128
来自 192.168.1.56 的回复: 字节=700 时间=29ms TTL=128
```

图 9-10 ping 命令

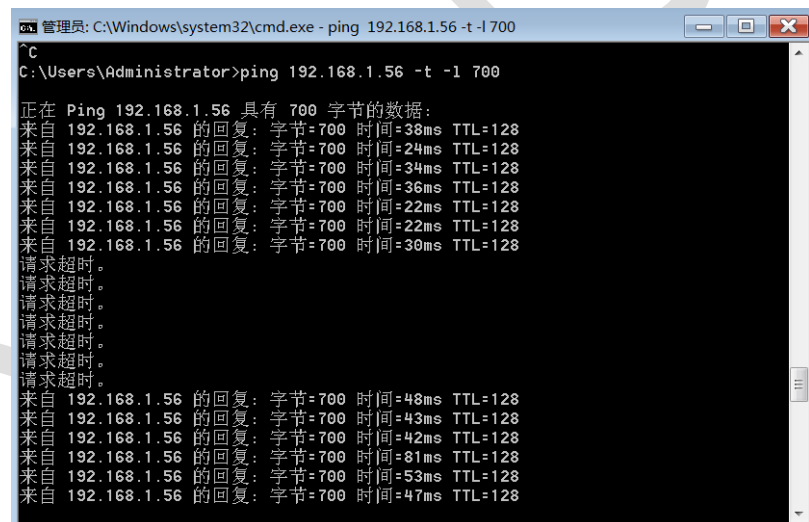
- 3) 在 ping 报文过程中，使用 nic 测试仪发送超过 1G 的流量，配置过程不再赘述；
- 4) 当 nic 测试仪流量发送完成，ping 报文又能恢复正常。

9.3.3. 测试结果



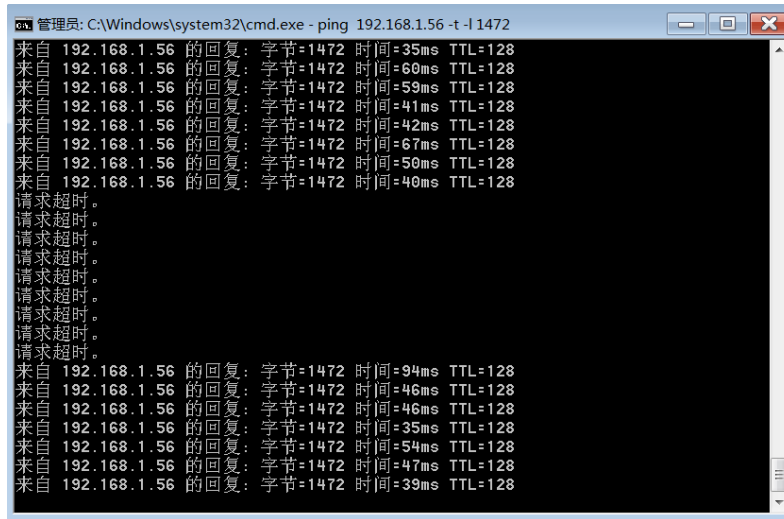
```
管理员: C:\Windows\system32\cmd.exe - ping 192.168.1.56 -t -l 200
来自 192.168.1.56 的回复: 字节=200 时间=77ms TTL=128
来自 192.168.1.56 的回复: 字节=200 时间=36ms TTL=128
来自 192.168.1.56 的回复: 字节=200 时间=29ms TTL=128
来自 192.168.1.56 的回复: 字节=200 时间=35ms TTL=128
来自 192.168.1.56 的回复: 字节=200 时间=36ms TTL=128
来自 192.168.1.56 的回复: 字节=200 时间=32ms TTL=128
来自 192.168.1.56 的回复: 字节=200 时间=44ms TTL=128
来自 192.168.1.56 的回复: 字节=200 时间=46ms TTL=128
来自 192.168.1.56 的回复: 字节=200 时间=32ms TTL=128
请求超时。
请求超时。
请求超时。
请求超时。
请求超时。
请求超时。
请求超时。
来自 192.168.1.56 的回复: 字节=200 时间=33ms TTL=128
来自 192.168.1.56 的回复: 字节=200 时间=82ms TTL=128
来自 192.168.1.56 的回复: 字节=200 时间=44ms TTL=128
来自 192.168.1.56 的回复: 字节=200 时间=38ms TTL=128
来自 192.168.1.56 的回复: 字节=200 时间=29ms TTL=128
来自 192.168.1.56 的回复: 字节=200 时间=37ms TTL=128
来自 192.168.1.56 的回复: 字节=200 时间=33ms TTL=128
```

图 9-11 ping 报文长度 200 结果



```
管理员: C:\Windows\system32\cmd.exe - ping 192.168.1.56 -t -l 700
^C
C:\Users\Administrator>ping 192.168.1.56 -t -l 700
正在 Ping 192.168.1.56 具有 700 字节的数据:
来自 192.168.1.56 的回复: 字节=700 时间=38ms TTL=128
来自 192.168.1.56 的回复: 字节=700 时间=24ms TTL=128
来自 192.168.1.56 的回复: 字节=700 时间=34ms TTL=128
来自 192.168.1.56 的回复: 字节=700 时间=36ms TTL=128
来自 192.168.1.56 的回复: 字节=700 时间=22ms TTL=128
来自 192.168.1.56 的回复: 字节=700 时间=22ms TTL=128
来自 192.168.1.56 的回复: 字节=700 时间=30ms TTL=128
请求超时。
请求超时。
请求超时。
请求超时。
请求超时。
请求超时。
请求超时。
来自 192.168.1.56 的回复: 字节=700 时间=48ms TTL=128
来自 192.168.1.56 的回复: 字节=700 时间=43ms TTL=128
来自 192.168.1.56 的回复: 字节=700 时间=42ms TTL=128
来自 192.168.1.56 的回复: 字节=700 时间=81ms TTL=128
来自 192.168.1.56 的回复: 字节=700 时间=53ms TTL=128
来自 192.168.1.56 的回复: 字节=700 时间=47ms TTL=128
```

图 9-12 ping 报文长度 700 结果



```
管理员: C:\Windows\system32\cmd.exe - ping 192.168.1.56 -t -l 1472
来自 192.168.1.56 的回复: 字节=1472 时间=35ms TTL=128
来自 192.168.1.56 的回复: 字节=1472 时间=60ms TTL=128
来自 192.168.1.56 的回复: 字节=1472 时间=59ms TTL=128
来自 192.168.1.56 的回复: 字节=1472 时间=41ms TTL=128
来自 192.168.1.56 的回复: 字节=1472 时间=42ms TTL=128
来自 192.168.1.56 的回复: 字节=1472 时间=67ms TTL=128
来自 192.168.1.56 的回复: 字节=1472 时间=50ms TTL=128
来自 192.168.1.56 的回复: 字节=1472 时间=40ms TTL=128
请求超时。
请求超时。
请求超时。
请求超时。
请求超时。
请求超时。
请求超时。
请求超时。
请求超时。
来自 192.168.1.56 的回复: 字节=1472 时间=94ms TTL=128
来自 192.168.1.56 的回复: 字节=1472 时间=46ms TTL=128
来自 192.168.1.56 的回复: 字节=1472 时间=46ms TTL=128
来自 192.168.1.56 的回复: 字节=1472 时间=35ms TTL=128
来自 192.168.1.56 的回复: 字节=1472 时间=54ms TTL=128
来自 192.168.1.56 的回复: 字节=1472 时间=47ms TTL=128
来自 192.168.1.56 的回复: 字节=1472 时间=39ms TTL=128
```

图 9-13 ping 报文长度 1472 结果

从上图中看 ping 报文的测试结果，符合预期。