0 环境

两台虚拟机 *Ubuntu* 系统

1 配置

（1）配置文件 config.ini

```
# tsn context
deviceName=ens33 # device name
sendDevice=lo # send device

# directory
gclDir=/home/reptile/lede/TSN/config/gcl.xml
routesDir=/home/reptile/lede/TSN/config/routes.xml
configDir=/home/reptile/lede/TSN/config/config.ini
```

① 网卡

- 接收数据的网卡 *deviceName*
- 发送数据的网卡 *sendDevice*

查看网卡命令：*ifconfig*

② 目录

填写 **绝对路径**

门控列表目录（*gclDir*）和 *MAC* 地址目录（*routesDir*）均在源代码的 *config* 目录下

tips：运行时会要求输入该配置文件的目录（*configDir*），要是不想每次运行都要输入该目录，可以修改源代码的 *util/config* 目录下的 *ConfigSetting.cc*，输入该配置文件的绝对路径，再编译出来执行就不会要求输入目录了，示例如下

```
ConfigSetting::ConfigSetting() : m_delimiter(std::string(1, '=')),
m_commet(std::string(1, '#')) {
    // std::cout << "请输入配置文件 config.ini 的绝对路径: ";
    // std::cin >> this->m_filename;
    this->m_filename = "/home/reptile/lede/TSN/config/config.ini";
    // this->m_filename = "/tmp/config.ini";
    std::ifstream in(this->m_filename.c_str());
    if (!in) throw FileNotFoundException(this->m_filename.c_str());
    in >> (*this);
}
```

（2）MAC 地址表 routes.xml

① 网卡

配置文件中接收数据的网卡需要在文件中进行配置，比如 *ens33* 网卡，就在 *<filteringDatabase id="ens33"></filteringDatabase>* 的标签中进行配置

② MAC 地址匹配端口号

发送数据的设备的 *MAC* 地址需要在上述标签中注册，端口号则决定了采用哪一个端口进行管理，比如发送数据的设备的 *MAC* 地址是 00-0c-29-ad-5c-75，用端口 0 进行处理，那么相应的配置就是 *<individualAddress port="0" macAddress="00-0c-29-ad-5c-75"/>*

（3）服务端网卡

源代码 *test* 目录下 *test_sender* 中默认选择的网卡是配置文件中的 *deviceName*，也可根据实际情况进行修改

2 使用

前置准备：

```
sudo apt-get install libpcap-dev
sudo apt-get install libnet1-dev
```

（1）编译

- 库文件 *lib* 下 *make*
- 源代码根目录 *make* 和 *make test* 轮流几次

编译生成的结果在 *bin* 和 *bin/test* 目录下

（2）测试环境

AP 和 *Station* 构成的一个小型局域网

- *AP* 处理数据
- *Station* 分为服务端和客户端
    - 服务端发送数据
    - 客户端接收数据

（3）测试数据

<table>
<tr><td colspan="3">Dst MAC</td><td colspan="3">Src MAC</td></tr>
<tr><td>Proto</td><td>TCI</td><td>Type</td><td>Reserve</td><td>Seq</td><td>Type</td></tr>
<tr><td colspan="6">Data</td></tr>
</table>

1）序列号：Seq

2）优先级：TCI 中的 pcp 码

① 优先级 0：0x 00

② 优先级 1：0x 20

③ 优先级 2：0x 40

④ 优先级 3：0x 60

⑤ 优先级 4：0x 80

⑥ 优先级 5：0x a0

⑦ 优先级 6：0x c0

⑧ 优先级 7：0x e0

（4）测试

- 服务端：虚拟机 *A* 执行 *bin/test* 下的 *test_sender*
- *AP*：虚拟机 *B* 执行 *bin* 下的 *tsn_app*
- 测试工具：*wireshark*

① 服务端使用 *wireshark* 监测发送数据的网卡，可得到发送前的数据顺序

② AP 端直接使用 *wireshark* 检测发送数据的网卡（实际相当于客户端接受的数据），可得到 AP 处理后的数据顺序

3 任务

（1）门控列表 gcl.xml

① 熟悉代码

② 第一阶段：静态门控列表

读入已实现，在源代码根目录 *src/core/queue* 的 *GateControlList.cc* 和 *GateControlList.h*，所以 *test* 目录下的 *test_queue*（对应的可执行文件在 *bin/test* 下的 *test_queue*）可以直接读取 *gcl.xml* 进行测试

现在的门控列表是独立的功能，要求融入整体的功能中，同时要注意时间同步

③ 第二阶段：动态门控列表

根据实际的网络状况动态生成门控列表

（2）实时测试

① 熟悉代码

② 第一阶段：监测抓包

目前的测试采用的是 *wireshark* 监测网卡进行抓包，要求脱离 *wireshark*，自定义一个实时检测抓包程序

可参考源代码根目录 *test* 下的 *test_receiver*，即 *socket* 接收数据

③ 第二阶段：数据分析

对监测抓包到的数据进行分析，挖掘数据价值，按照评测标准比如时延、丢包率等实时绘图