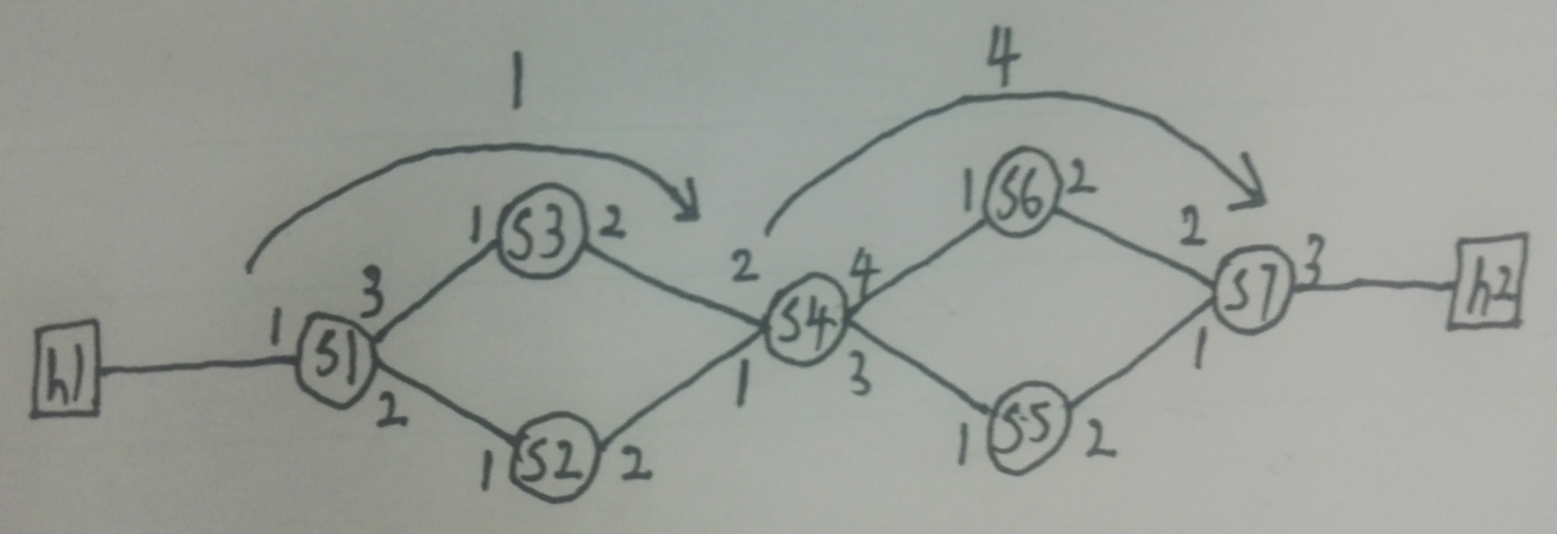
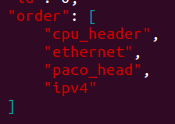
本实验在paco协议验证实验的基础上进行，因此实验资源是相同的。

实验拓扑如下图，与验证实验相同

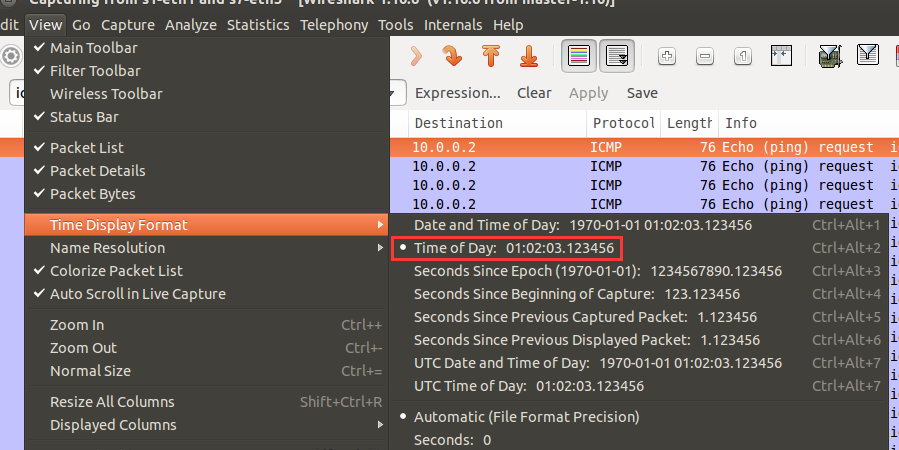


实验步骤：

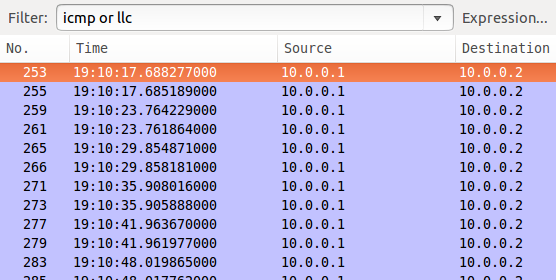
1. 运行build.sh编译p4程序，编译成功后，编辑paco.p4：搜索order，将order下面的内容改成如下图所示，如果已经一样，则不用修改



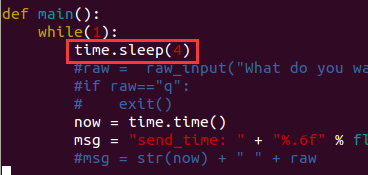
1. 运行run\_demo.sh搭建实验环境
2. 等mininet启动换成后在mininet的CLI中输入xterm h1 h2会分别弹出主机h1和主机h2的控制台
3. 以管理员身份（命令行输入sudo wireshark）启动wireshark。然后选择监听端口（测试延时路径的两端），比如本实验要测试数据包从s1到s7的时间，则需要同时监听s1的1端口（s1-eth1）和s7的3端口（s7-eth3），选择好端口后，开始监听，然后在过滤栏中输入“icmp or llc”并应用。然后对wireshark进行如下图所示的配置



1. 在主机h2上运行reveive.py,在主机h1上运行send.py，如果成功，几秒后h2的控制台会有信息输出。同时在wireshark中将会看到抓包结果。如下图所示



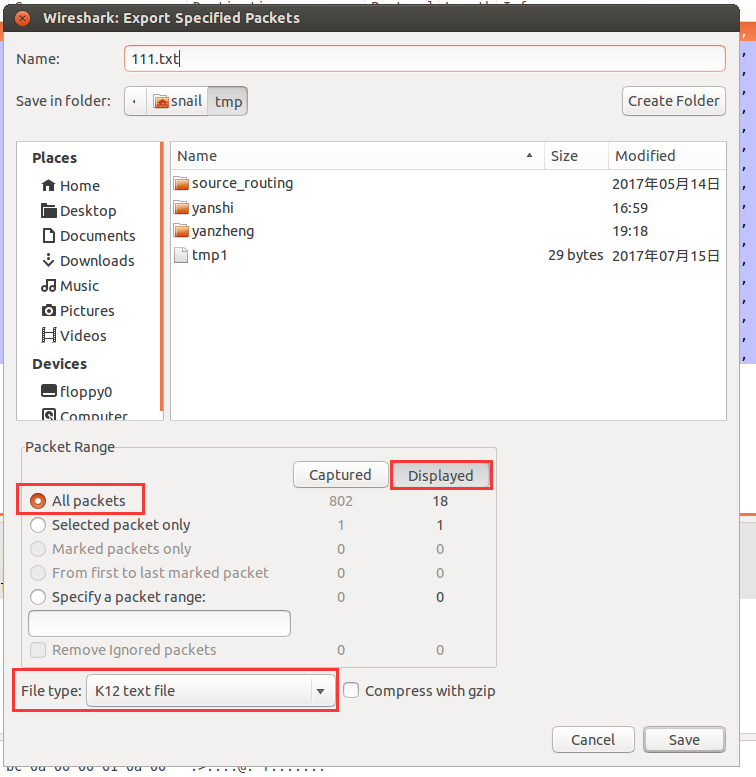
图中时间相近的两次抓取结果为一次传输的数据包分别在入端口和出端口被抓取的结果，以此可以分析延时（注：有时出端口的抓取结果会显示在入端口抓取结果的上面）。图中可以看到每两次传输大概会有6秒的间隔，在send.py脚本中可以设置每两次传输的时间间隔，如下图中的time.sleep(4)为延时4秒（注：发包用的是python库scapy，用send.py中的方法发包时，默认有2秒延时，因此实验看到的是4+2=6秒延时，也就是说，不设置延时的话，也会有2秒的延时，至于原因我也不太清楚，如果需要了解，可以自行阅读scapy官网）



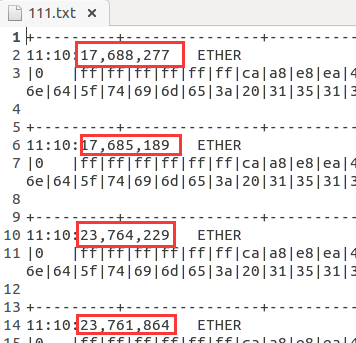
1. 导出实验结果

点击wireshark菜单栏的file->Export specified Packets

然后按照如下图选择，之后选择要保存的路径以及文件名，点击save



之后就会输出如下图所示的内容，利用文本中的时间信息即可编程自动分析延时



（7）完成测试，按照paco验证实验清理实验环境即可