SDN Experiment 4

实验环境

与第二次实验相同

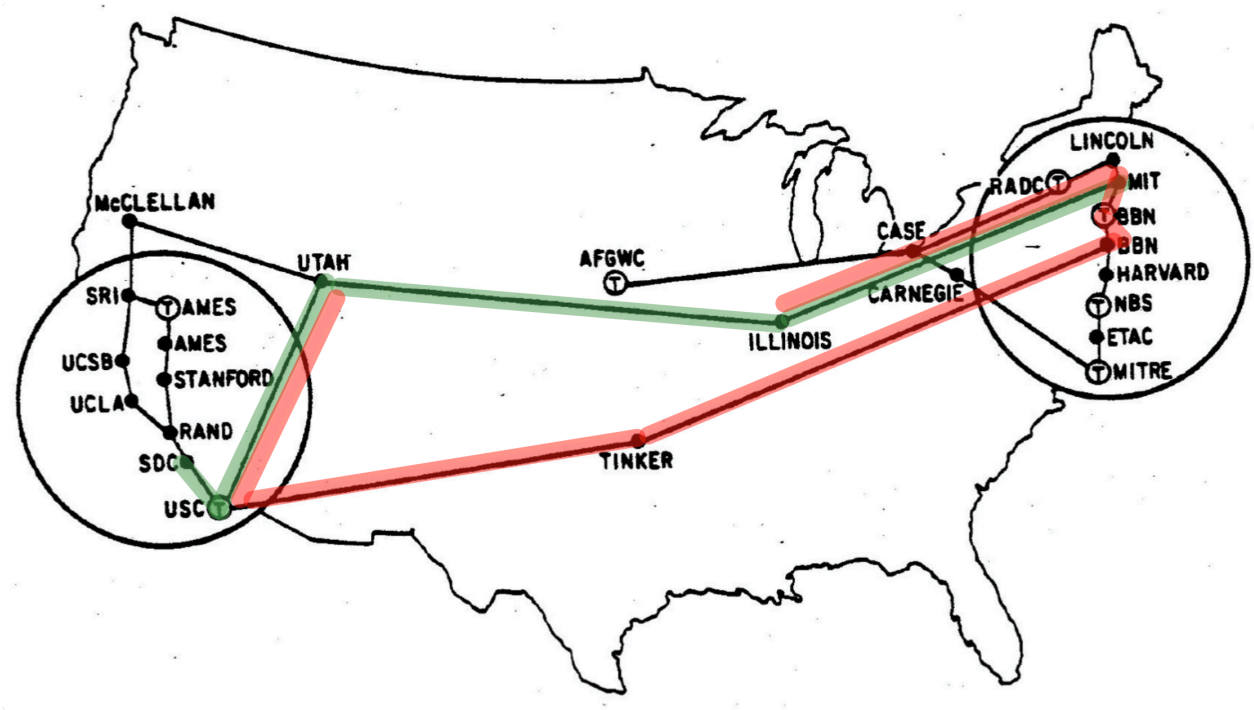
实验内容

题目

假如你是生活在1972年维护ARPAnet的网络管理员，在前面的实验中你学会了如何建立最短路径，下发了一条SDC到MIT跳数最少的路径（图中绿色的路径）。你的同事Bob某天接到了一个新的需求，要求UTAH到ILLINOIS之间的所有流量必须经过部署于TINKER的流量分析器以进行进一步研究，粗心大意的Bob没有检查当前的网络状态就很快下发了一条新的路径（图中红色的路径）。聪明又机智的你很快意识到Bob下发的流表很可能造成转发的环路。

现要求你运行VeriFlow工具，对上述两条转发路径进行检查，完成下面任务：

1. 打印出环路路径的信息
2. 进一步打印出环路对应的EC的相关信息

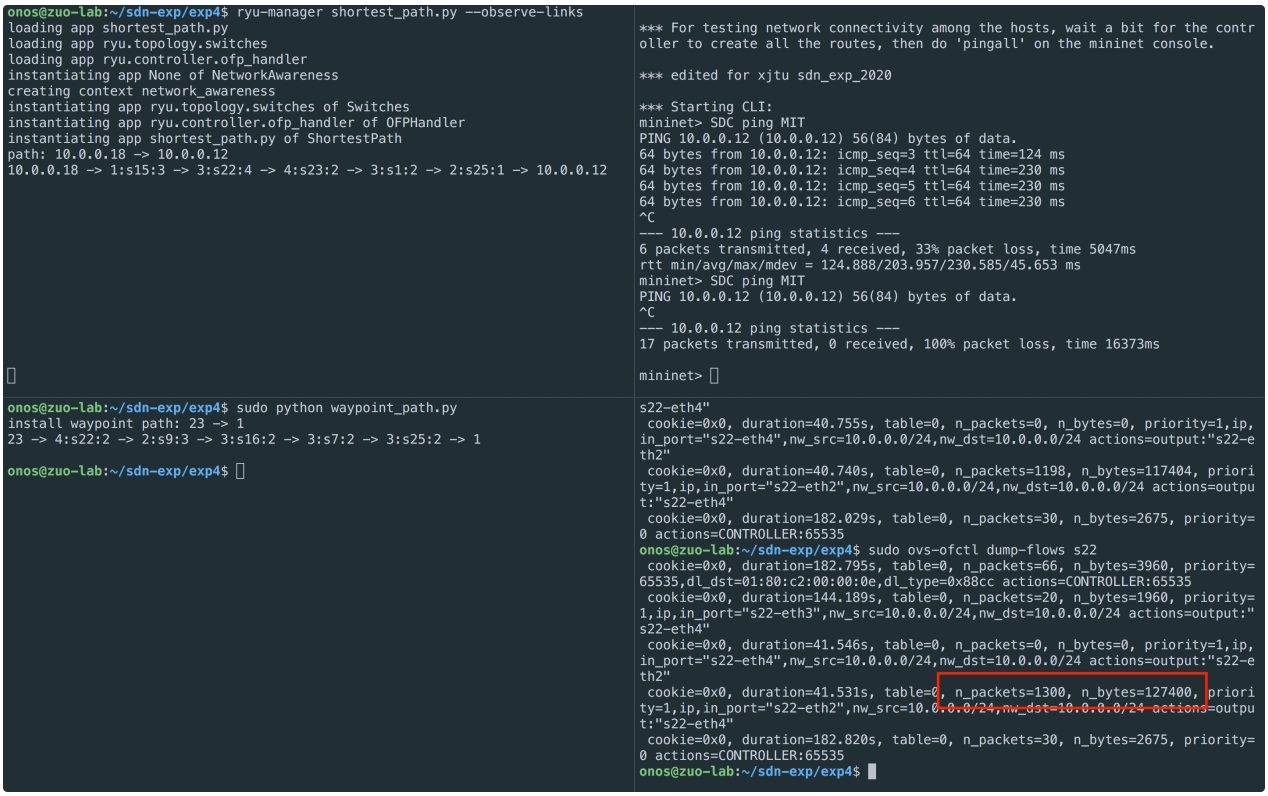


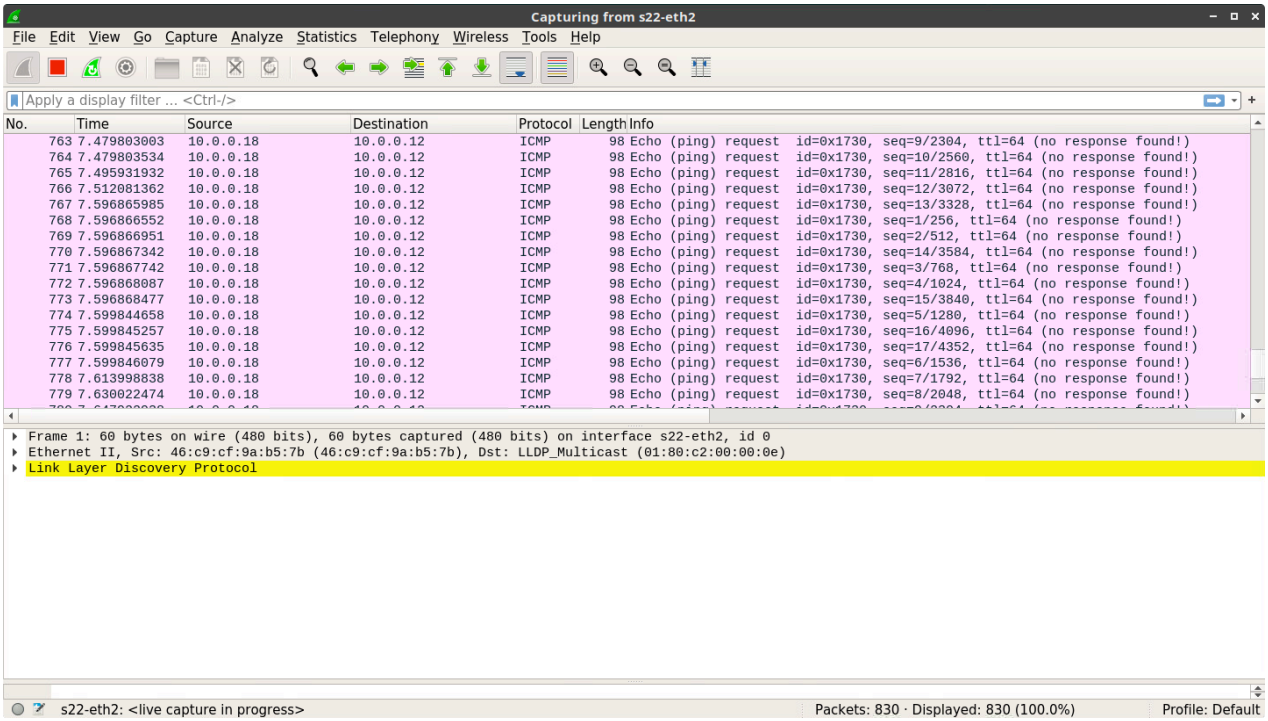
说明

- 如何观察转发的环路问题?

```
# 1. 启动拓扑
sudo python Arpanet19723.py
```

```
# 2. 启动最短路径的控制程序
ryu-manager ofctl_rest.py shortest_path.py --observe-links

# 3. 在拓扑中SDC ping MIT建立连接
mininet> SDC ping MIT

# 4. Bob下发从UTAH途经TINKER到达ILLINOIS的路径之后，你尝试SDC ping MIT失败
python waypoint_path.py

# 5. 查看路径上某一个交换机，如USC的流表，发现匹配某一条流表的数据包数目异常增加
# 也可打开wireshark观察该端口，发现不断增加的ICMP Request报文
sudo ovs-ofctl dump-flows s22
```





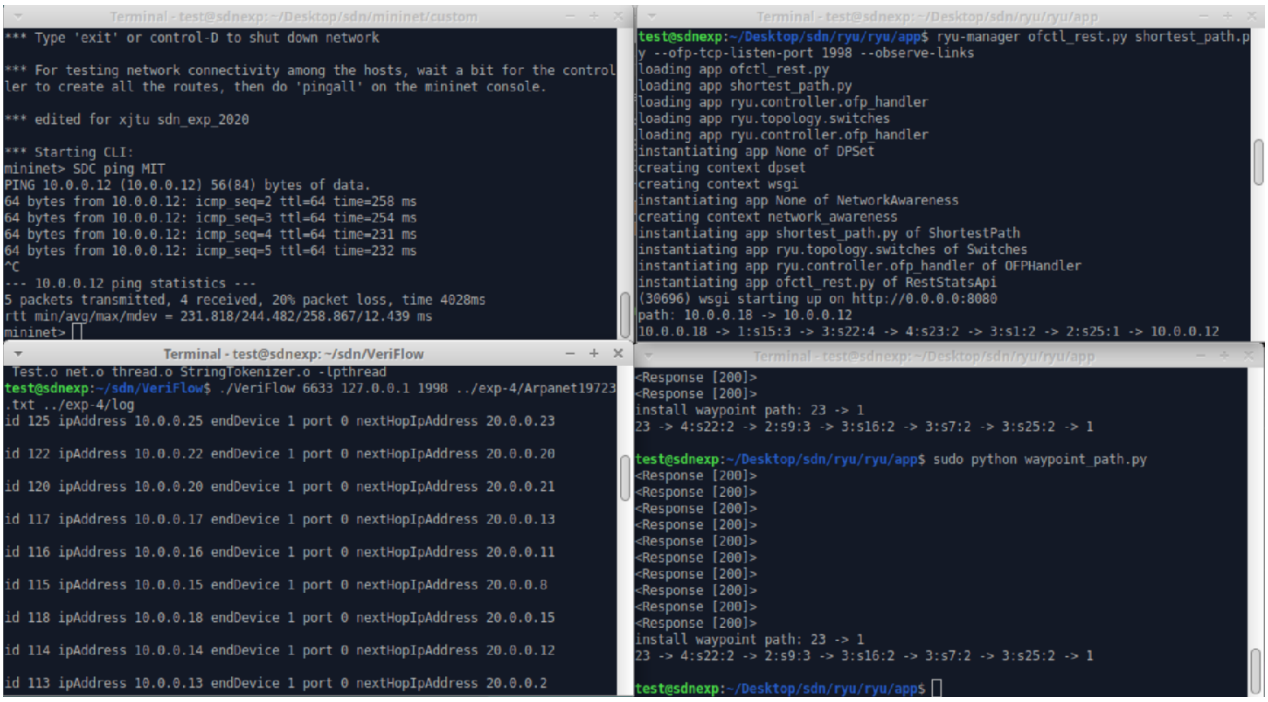
- 最短路径算法中为何使用rest api下发流表?

  由于VeriFlow仅支持OpenFlow1.0，`shortest_path.py` 与 `waypoint_path.py` 中使用rest api更简便。rest api所用到的文件 `ofctl_rest.py` 位于路径ryu/ryu/app/中

- 如何使用VeriFlow

```
# 0. 从github下载VeriFlow并打上实验补丁
git clone https://github.com/samueljero/BEADS.git
cd BEADS
git am 0001-for-xjtu-sdn-exp-2020.patch

# 1. 编译VeriFlow
cd veriflow/VeriFlow
make clean all

# 2. 在自定义端口开启远程控制器，运行最短路程序
ryu-manager ofctl_rest.py shortest_path.py --ofp-tcp-listen-port 1024 --
observe-links

# 3. 运行VeriFlow的proxy模式
VeriFlow的proxy模式的cmd格式为：
VeriFlow <veriflow_port> <controller_address> <controller_port> <topology_file>
<log_file>
可用如下命令运行VeriFlow的proxy模式：
./VeriFlow 6633 127.0.0.1 1024 Arpanet19723.txt log_file.txt

# 4. 启动拓扑
sudo python Arpanet19723.py

# 5. 在拓扑中SDC ping MIT建立连接
mininet> SDC ping MIT
```

```
# 6. 下发从UTAH途经TINKER到达ILLINOIS的路径，在log文件中观察VeriFlow检测到的环路信息
python waypoint_path.py
```



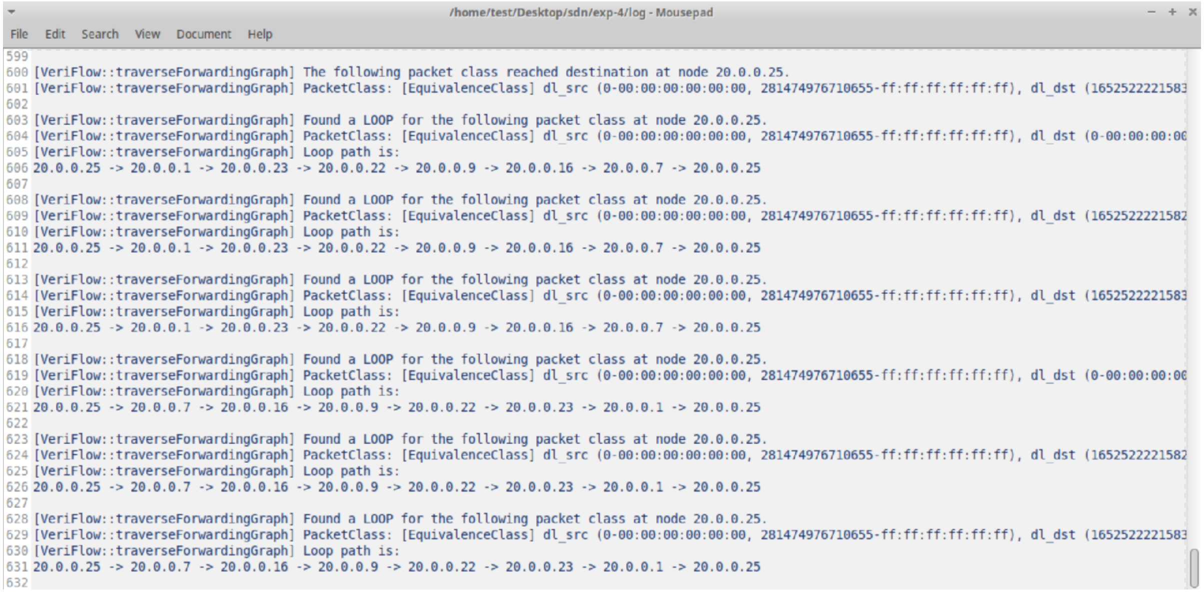
- VeriFlow的主要类或函数

```
# 1. VeriFlow::main()
VeriFlow的程序入口，规定了test模式和proxy模式的调用格式

# 2. VeriFlow::parseTopologyFile()
VeriFlow解析拓扑文件，建立网络模型的函数，规定了拓扑文件的格式

# 3. VeriFlow::handleVeriFlowConnection()
处理socket连接关系的函数，每个连接拥有两个单向通信线程，实现控制器和交换机之间的双向通信

# 4. OpenFlowProtocolMessage::process()
处理OpenFlow消息的入口函数，根据消息的类型调用相应的处理函数

# 5. OpenFlowProtocolMessage::processFlowRemoved()
处理OFPT_FLOW_REMOVED消息的函数

# 6. OpenFlowProtocolMessage::processFlowMod()
处理OFPT_FLOW_MOD消息的函数

# 7. EquivalenceClass
表示VeriFlow定义的等价类的数据结构，包括每个域的名称和存储的顺序

# 8. VeriFlow::verifyRule()
执行VeriFlow核心算法的函数，包括对等价类的划分、转发图的构造与不变量的验证

# 9. VeriFlow::traverseForwardingGraph()
```

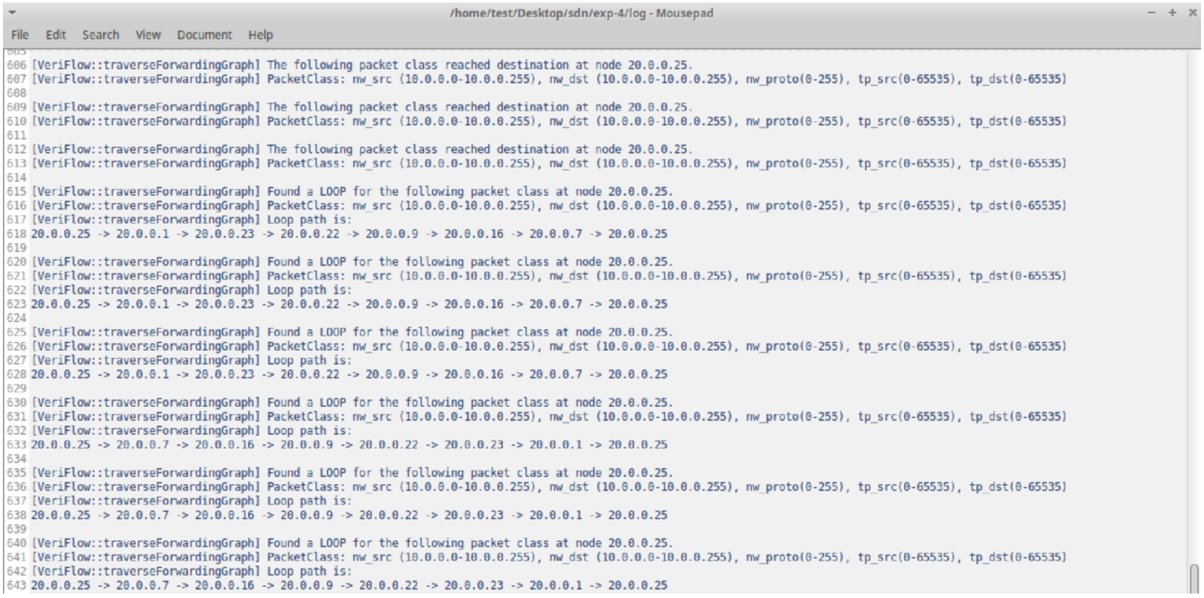
遍历某个特定EC的转发图，验证是否存在环路或黑洞

示例

- 环路路径的打印 本实验要求打印出环路的信息，包括出现环路的提示信息，EC的基本信息和环路路径上的IP地址 提示：traverseForwardingGraph函数中的visited为unordered_set，可改成有序的数据结构



- 相关数据包信息的打印 EC的基本信息显示为14个域的区间形式，为方便Bob查错，现简化EC信息的表示形式，仅从14个域中提取TCP/IP五元组作为主要信息显示 提示：在环路路径打印的基础上，修改EC的显示格式



参考

- 第二次实验参考自李呈的 network awareness 项目
- VeriFlow 的使用说明参考 README
- VeriFlow 相关论文，汇报视频请参考NSDI'13会议网站