中山大學
SUN YAT-SEN UNIVERSITY

本科生实验报告

Undergraduate Experiment Report

题目 Title: **计算机网络 (II) 实验报告**

院 系
School (Department): 数据科学与计算机学院

专 业
Major: 网络空间安全

学生姓名
Student Name: 金钰 王广烁 李晨曦

学 号
Student No.:

时间：二〇二〇年十二月三十日

Date: December 30th 2020

目录

插图目录

表格目录

第一章 背景

第二章 方案设计

2.1 课程作业目的

2.2 场景设计与预期要求

2.3 实现方案与技术细节

2.3.1 传统路由和基于 TCP 与 UDP 路由的实现方案

2.3.2 实现传统路由

2.3.2.1 搭建拓扑

首先找到 `classic/create_topo.py` 这个文件，在配置好 mininet 的主机 A 中运行

```
sudo mn --custom create_topo.py --topo mytopo \
--switch ovs --controller remote,ip={ip of B} --link tc
```

其中 `ip` 项是控制器主机 B 的 ip 地址。

在 B 主机中运行

```
ryu run --observe-links  \
ryu.app.rest_router ryu.app.gui_topology.gui_topology
```

这样一来整个拓扑已经搭建成功。浏览 `http://ip_of_B:8080` 可以查看拓扑图像如下：

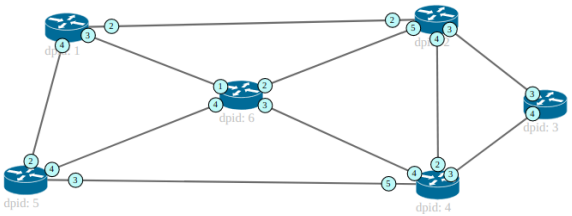

图 2-1 拓扑图

注意到这里没有画出主机的拓扑情况。

2.3.2.2 生成路由

首先找到 `classic/make_route.py`，在主机 B 上运行

```
python make_route.py
```

这会在控制器上创建基于最短路算法的静态路由。此时路由已经创建好，但是，由于主机的默认网关均未进行配置，要手动配置默认网关。

在 mininet 的命令行中运行

```
u1 ip route add default via 172.18.0.2 \
u2 ip route add default via 172.18.1.2 \
u3 ip route add default via 172.18.2.2 \
u4 ip route add default via 172.18.3.2 \
u5 ip route add default via 172.18.4.2 \
u6 ip route add default via 172.18.2.2 \
u7 ip route add default via 172.18.2.2 \
u8 ip route add default via 172.18.2.2
```

这样一来路由就搭建成功了。

2.3.2.3 测试路由

在 mininet 的命令行中运行

```
pingall
```

得到输出如下：

```
mininet> pingall
*** Ping: testing ping reachability
u1 -> u2 u3 u4 u5 u6 u7 u8
u2 -> u1 u3 u4 u5 u6 u7 u8
u3 -> u1 u2 u4 u5 u6 u7 u8
u4 -> u1 u2 u3 u5 u6 u7 u8
u5 -> u1 u2 u3 u4 u6 u7 u8
u6 -> u1 u2 u3 u4 u5 u7 u8
u7 -> u1 u2 u3 u4 u5 u6 u8
u8 -> u1 u2 u3 u4 u5 u6 u7
*** Results: 0% dropped (56/56 received)
```

这样一来就可以确定路由已经正常工作。

2.3.2.4 性能测试

在 mininet 的命令行中运行

```
u3 iperf3 -s -D
u6 iperf3 -s -D
u7 iperf3 -s -D
u8 iperf3 -s -D
u1 iperf3 -c u3 -t 20 > u1.tcp &
u1 iperf3 -c u6 -u -b 40M > u1.udp &
u2 iperf3 -c u7 -t 20 > u2.tcp &
u2 iperf3 -c u8 -u -b 40M > u2.udp &
```

性能测试的结果在 `u1.tcp, u1.udp, u2.tcp, u2.udp` 这四个文件中。

2.3.3 实现基于 UDP 和 TCP 选择的 SDN 路由

2.3.3.1 搭建拓扑

首先找到 `classic/create_topo.py` 这个文件，在配置好 mininet 的主机 A 中运行

```
sudo mn --custom create_topo.py --topo mytopo \
--switch ovs --controller remote,ip={ip of B} --link tc
```

其中 `ip` 项是控制器主机 B 的 ip 地址。

在 B 主机中运行

```
ryu run --observe-links  \
ryu.app.rest_router ryu.app.gui_topology.gui_topology \
ryu.app.ofctl_rest
```

这样一来整个拓扑已经搭建成功。实际上，这里的拓扑和传统路由时的拓扑没有区别。

2.3.3.2 生成路由

首先找到 `sdn-tcp-udp` 文件夹中的 `make_classic_route.py` 和 `make_sdn_route.py`，在主机 B 上运行

```
python make_classic_route.py \
python make_sdn_route.py
```

这会在控制器上创建基于最短路算法的静态路由和基于 UDP 和 TCP 的 SDN 路由。

此时路由已经创建好，但是，由于主机的默认网关均未进行配置，要手动配置默认网关。

在 mininet 的命令行中运行

```
u1 ip route add default via 172.18.0.2
u2 ip route add default via 172.18.1.2
u3 ip route add default via 172.18.2.2
u4 ip route add default via 172.18.3.2
u5 ip route add default via 172.18.4.2
u6 ip route add default via 172.18.2.2
u7 ip route add default via 172.18.2.2
u8 ip route add default via 172.18.2.2
```

这样一来路由就搭建成功了。

2.3.3.3 测试路由

在 mininet 的命令行中运行

```
pingall
```

得到输出如下：

```
mininet> pingall
*** Ping: testing ping reachability
u1 -> u2 u3 u4 u5 u6 u7 u8
u2 -> u1 u3 u4 u5 u6 u7 u8
u3 -> u1 u2 u4 u5 u6 u7 u8
u4 -> u1 u2 u3 u5 u6 u7 u8
u5 -> u1 u2 u3 u4 u6 u7 u8
u6 -> u1 u2 u3 u4 u5 u7 u8
u7 -> u1 u2 u3 u4 u5 u6 u8
u8 -> u1 u2 u3 u4 u5 u6 u7
*** Results: 0% dropped (56/56 received)
```

这样一来就可以确定路由已经正常工作。

2.3.3.4 性能测试

这里进行了两次性能测试, 一次测试和传统路由相同, 限制 udp 的速率为 $40Mbp/s$, 一次把 udp 的速率限制到 $100Mbp/s$。

在 mininet 的命令行中运行

```
u3 iperf3 -s -D
u6 iperf3 -s -D
u7 iperf3 -s -D
u8 iperf3 -s -D

u1 iperf3 -c u3 -w 1M -t 20 > u1-40.tcp &
u1 iperf3 -c u6 -u -b 40M > u1-40.udp &
u5 iperf3 -c u7 -w 1M -t 20 > u5-40.tcp &
u5 iperf3 -c u8 -u -b 40M > u5-40.udp &

u1 iperf3 -c u3 -w 1M -t 20 > u1-100.tcp &
u1 iperf3 -c u6 -u -b 100M > u1-100.udp &
u5 iperf3 -c u7 -w 1M -t 20 > u5-100.tcp &
u5 iperf3 -c u8 -u -b 100M > u5-100.udp &
```

第一次性能测试的结果在 `u1-40.tcp, u1-40.udp, u5-40.tcp, u5-40.udp` 这四个文件中。

第二次性能测试的结果在 `u1-100.tcp, u1-100.udp, u5-100.tcp, u5-100.udp` 这四个文件中。

2.4 组员的分工与工作

1[1]

2.5 实验中遇到的挑战与问题及其解决方法

第三章 实验分析

第四章 总结

参考文献:

[1] LONG J, SHELHAMER E, DARRELL T. Fully convolutional networks for semantic segmentation[C] // Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2015 : 3431 – 3440.