Shanghai Jiao Tong University

Computer Networks Lab4

Author: *LIU Qiaoan 520030910220*

Course: *Spring 2022, CS3611: Computer Networks*
Date: *May 16, 2022*

Setup

In this section, I write LAN1.py and LAN2.py to construct 2 LANs. Set all the links bandwidth as 10Mbps, the packet loss rate as 0%, and the time delay as 5ms. Note that both LANs in the two VMs are behind NAT, so h1/h2 cannot communicate with h3/h4 diretly. Then, we need to use VXLAN to build a tunnel between these two LANs [1].

The IP address of 2 VMs are:

- In VM1:
  enp0s8: 192.168.56.101
- In VM2:
  enp0s8: 192.168.56.102

After configure of IP address, I find that I cannot ping 10.0.0.6/10.0.0.3/10.0.0.4 from 10.0.0.5/10.0.0.1/10.0.0.2. Then I start to build the VXLAN. The first step is build bridge in LAN and assign the IP address of enp0s8 to it. The second step is to build the VXLAN tunnel and set the remote IP address.

Homework 1

After read some articles in the website [2], I find that after construct the bridge in both LANs, we need to connect them. Since we have combine the bridge with the network card enp0s8, we just need to use two lines of commands.

- In VM1:
  ```
  sudo ovs-vsctl add-port s1 vxlan0 -- set interface vxlan0
  type=vxlan options:remote_ip=192.168.56.102
  ```
- In VM2:
  ```
  sudo ovs-vsctl add-port s2 vxlan0 -- set interface vxlan0
  type=vxlan options:remote_ip=192.168.56.101
  ```

In this two commands, we add VXLAN between server and bridge of other LAN. Then we use ping to verify, see in Figure 1.

```
mininet> h1 ping 10.0.0.3
PING 10.0.0.3 (10.0.0.3) 56(84) bytes of data.
64 bytes from 10.0.0.3: icmp_seq=1 ttl=64 time=23.0 ms
64 bytes from 10.0.0.3: icmp_seq=2 ttl=64 time=26.6 ms
64 bytes from 10.0.0.3: icmp_seq=3 ttl=64 time=28.6 ms
64 bytes from 10.0.0.3: icmp_seq=4 ttl=64 time=26.9 ms
64 bytes from 10.0.0.3: icmp_seq=5 ttl=64 time=24.8 ms
64 bytes from 10.0.0.3: icmp_seq=6 ttl=64 time=26.6 ms
64 bytes from 10.0.0.3: icmp_seq=7 ttl=64 time=29.0 ms
64 bytes from 10.0.0.3: icmp_seq=8 ttl=64 time=26.5 ms
64 bytes from 10.0.0.3: icmp_seq=9 ttl=64 time=32.0 ms
64 bytes from 10.0.0.3: icmp_seq=10 ttl=64 time=24.8 ms
64 bytes from 10.0.0.3: icmp_seq=11 ttl=64 time=27.9 ms
64 bytes from 10.0.0.3: icmp_seq=12 ttl=64 time=34.4 ms
^C
--- 10.0.0.3 ping statistics ---
12 packets transmitted, 12 received, 0% packet loss, time 11039ms
rtt min/avg/max/mdev = 23.019/27.632/34.403/3.013 ms
```

Figure 1: VXLAN is constructed successfully

Homework 2

In this section, I ping 10.0.0.6 from 10.0.0.5 in VM1, and use Wireshark to monitor the interfaces s1 and enp0s8.

For s1, the result is seen in Figure 2.

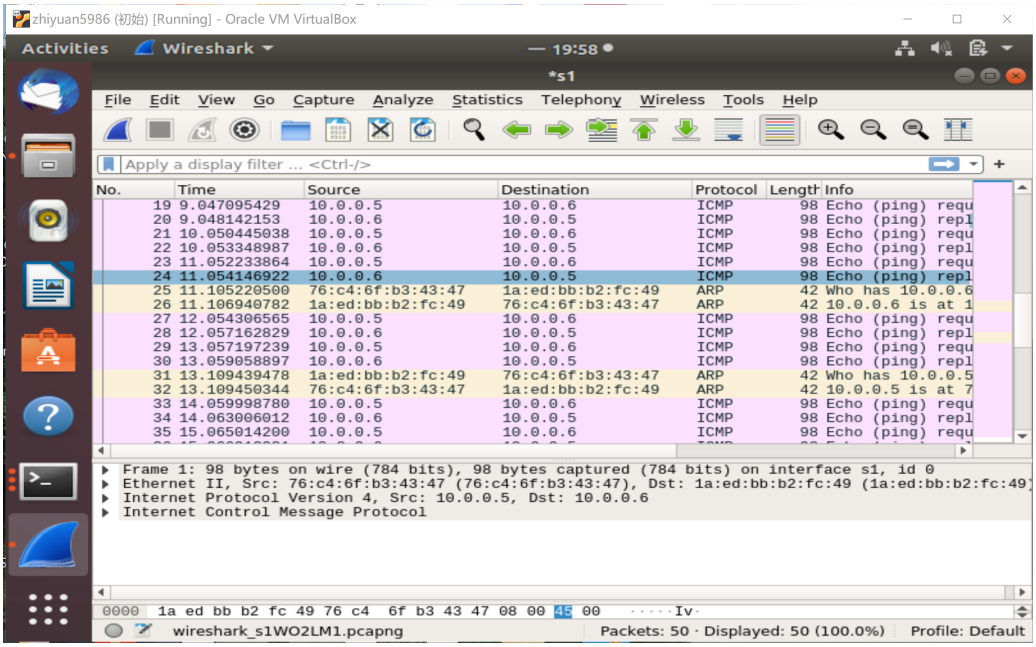

Figure 2: The packet passing s1

From Figure 2, we can see that Internet Control Message Protocol (ICMP), Internet Protocol Version 4 (IPv4) and Address Resolution Protocol (ARP) are used.

For enp0s8, the result is seen in Figure 3.

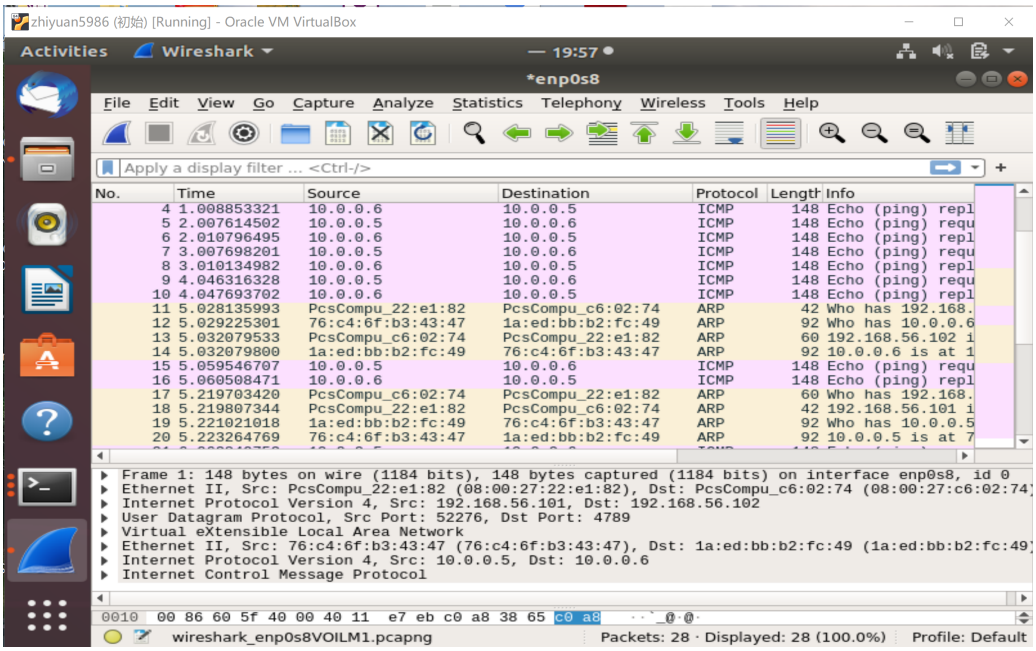

Figure 3: The packet passing enp0s8

From Figure 3, we can see that Internet Control Message Protocol (ICMP), Internet Protocol Version 4 (IPv4), Address Resolution Protocol (ARP) and User Datagram Protocol (UDP) are used.

Description.

- Internet Control Message Protocol (ICMP). ICMP is part of the Internet protocol suite as defined in RFC 792. ICMP messages are typically used for diagnostic or control purposes or generated in response to errors in IP operations (as specified in RFC 1122). ICMP errors are directed to the source IP address of the originating packet [3].
- Internet Protocol Version 4 (IPv4). It is one of the core protocols of standards-based internetworking methods in the Internet and other packet-switched networks. IPv4 uses a 32-bit address space which provides 2^{32} unique addresses, but large blocks are reserved for special networking purposes [4].
- Address Resolution Protocol (ARP). The Address Resolution Protocol (ARP) is a communication protocol used for discovering the link layer address, such as a MAC address, associated with a given internet layer address, typically an IPv4 address [5].
- User Datagram Protocol (UDP). It is one of the core members of the Internet protocol suite. It has no handshaking dialogues, and thus exposes the user's program to any unreliability of the underlying network; there is no guarantee of delivery, ordering, or duplicate protection [6].

Finding. I find that enp0s8 uses one more protocol, which is UDP. And the frame size of enp0s8 is 148, which is exactly 50 bytes more than the frame size of s1 that is 98. We will see soon that it's quite important that VXLAN adds 50 bytes to the datagram.

Homework 3

First, I test the bandwidth between 192.168.56.101 and 192.168.56.102. Set 192.168.56.101 as client and 192.168.56.102 as server. The screenshots of them are shown in Figure 4 and Figure 5.

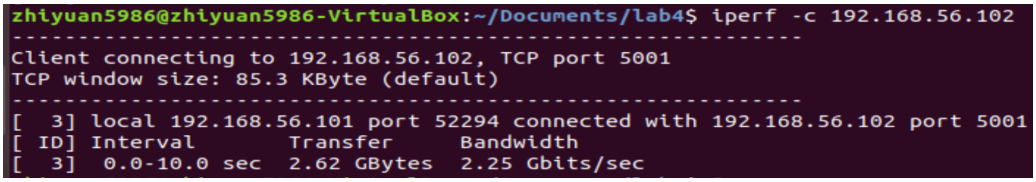

Figure 4: The screenshot of 192.168.56.101

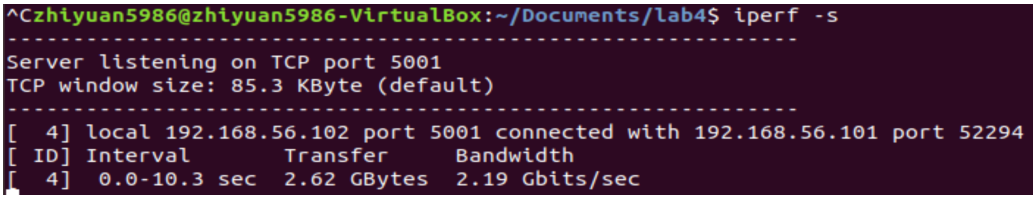

Figure 5: The screenshot of 192.168.56.102

We can see that the bandwidth between 192.168.56.101 and 192.168.56.102 is about 2.19 Gbits/sec.

Second, I test the bandwidth between 10.0.0.1/10.0.0.2/10.0.0.5 and 10.0.0.4. However, when I test the bandwidth between 10.0.0.5 and 10.0.0.4, I find that it's so small and there is no packet received by 10.0.0.4. See in Figure 6.

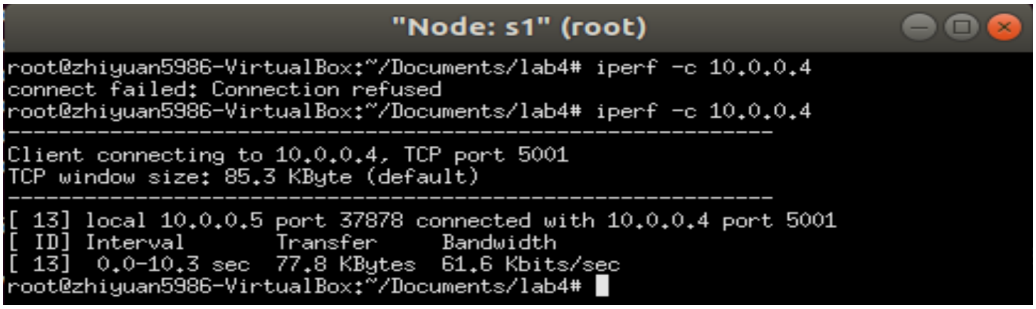

Figure 6: The screenshot of s1

After read hints and search on the Internet, I find that it's caused by maximum transmission unit (MTU). I find the MTU by `ifconfig br1`, see in Figure 7.

Since ping is an application layer command, we know that a datagram length is consist of 20 bytes datagram header, 20 bytes TCP segment header and application message. And we need 50 bytes for VXLAN in this lab. So the MSS of application message is: $1500-20-20-50=1410$. Then I add an extra parameter `-M` to specify the MSS of application layer message in the terminal. See the result in Figure 8, Figure 9, Figure 10 and Figure 11.

We can see that the bandwidth between 10.0.0.1/10.0.0.2/10.0.0.5 and 10.0.0.4 are all around 10 Mbits/sec, that's because the bandwidth is limited by the links of Mininet.

```
zhiyuan5986@zhiyuan5986-VirtualBox:~/Documents/lab4$ ifconfig br1
br1: flags=4098<BROADCAST,MULTICAST>  mtu 1500
        ether 08:00:27:22:e1:82  txqueuelen 1000  (Ethernet)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 397  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Figure 7: Find MTU

```
"Node: h1"
root@zhiyuan5986-VirtualBox:~/Documents/lab4# iperf -c 10.0.0.4 -M 1410
WARNING: attempt to set TCP maximum segment size to 1410, but got 536
------------------------------------------------------------
Client connecting to 10.0.0.4, TCP port 5001
TCP window size: 85.3 KByte (default)
------------------------------------------------------------
[ 13] local 10.0.0.1 port 37456 connected with 10.0.0.4 port 5001
[ ID] Interval       Transfer     Bandwidth
[ 13]  0.0-11.5 sec  11.6 MBytes  8.48 Mbits/sec
root@zhiyuan5986-VirtualBox:~/Documents/lab4#
```

Figure 8: The screenshot of h1

```
"Node: h2"
root@zhiyuan5986-VirtualBox:~/Documents/lab4# iperf -c 10.0.0.4 -M 1410
WARNING: attempt to set TCP maximum segment size to 1410, but got 536
------------------------------------------------------------
Client connecting to 10.0.0.4, TCP port 5001
TCP window size: 85.3 KByte (default)
------------------------------------------------------------
[ 13] local 10.0.0.2 port 42414 connected with 10.0.0.4 port 5001
[ ID] Interval       Transfer     Bandwidth
[ 13]  0.0-10.2 sec  12.6 MBytes  10.4 Mbits/sec
root@zhiyuan5986-VirtualBox:~/Documents/lab4# S
```

Figure 9: The screenshot of h2

```
root@zhiyuan5986-VirtualBox:~/Documents/lab4# iperf -c 10.0.0.4 -M 1410
WARNING: attempt to set TCP maximum segment size to 1410, but got 536
------------------------------------------------------------
Client connecting to 10.0.0.4, TCP port 5001
TCP window size: 85.3 KByte (default)
------------------------------------------------------------
[ 13] local 10.0.0.5 port 47198 connected with 10.0.0.4 port 5001
[ ID] Interval       Transfer     Bandwidth
[ 13]  0.0-10.7 sec  14.6 MBytes  11.5 Mbits/sec
root@zhiyuan5986-VirtualBox:~/Documents/lab4# S
```

Figure 10: The screenshot of s1

```
"Node: h4"
root@zhiyuan5986-VirtualBox:~/Documents/lab4# iperf -s
------------------------------------------------------------
Server listening on TCP port 5001
TCP window size: 85.3 KByte (default)
------------------------------------------------------------
[  6] local 10.0.0.4 port 5001 connected with 10.0.0.1 port 37456
[ ID] Interval       Transfer     Bandwidth
[  6]  0.0-14.9 sec  11.6 MBytes  6.53 Mbits/sec
[  6] local 10.0.0.4 port 5001 connected with 10.0.0.2 port 42414
[  6]  0.0-12.7 sec  12.6 MBytes  8.36 Mbits/sec
[  6] local 10.0.0.4 port 5001 connected with 10.0.0.5 port 47198
[  6]  0.0-12.9 sec  14.6 MBytes  9.53 Mbits/sec
```

Figure 11: The screenshot of h4

Homework 4

Similar to Q3, now I use ping to send 20 pkts to test the RTT. First, the lantancy between 192.168.56.101 and 192.168.56.102 is about 0.7 ms, see in Figure 12.

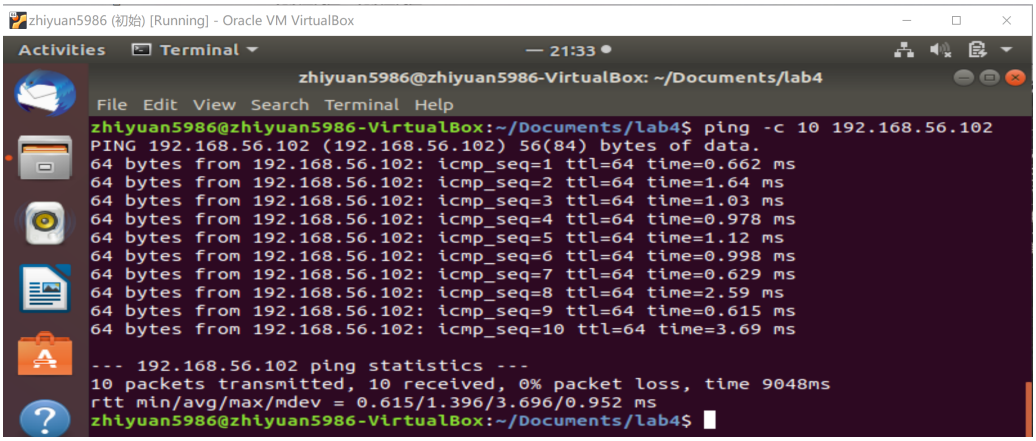

Figure 12: RTT between 192.168.56.101 and 192.168.56.102

Second, the lantancy between 10.0.0.5 and 10.0.0.4 is about 6.5 ms. That's because there is exactly one Mininet link between s1 and h4, which causes 5 ms delay.

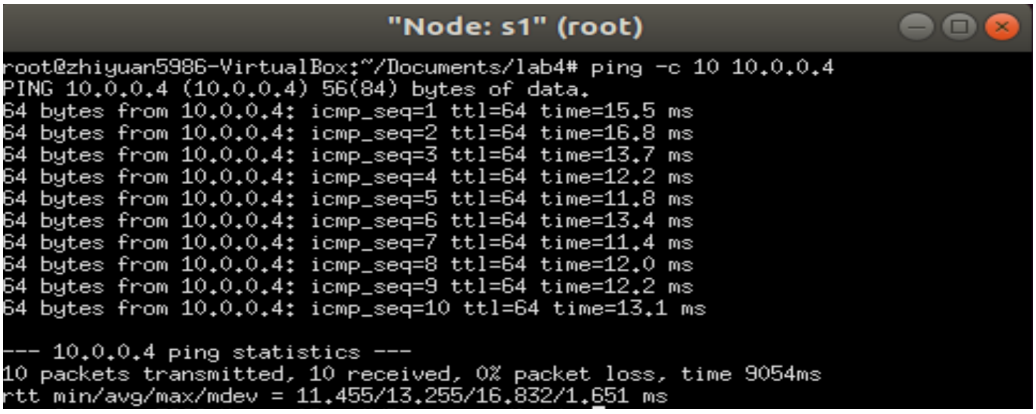

Figure 13: RTT between s1 and h4

And the lantancy between 10.0.0.1/10.0.0.2 and 10.0.0.4 is about 14.5 ms. That's because there are exactly two Mininet links between h1/h2 and h4, which cause 10 ms delay. See in Figure 14 and Figure 15.

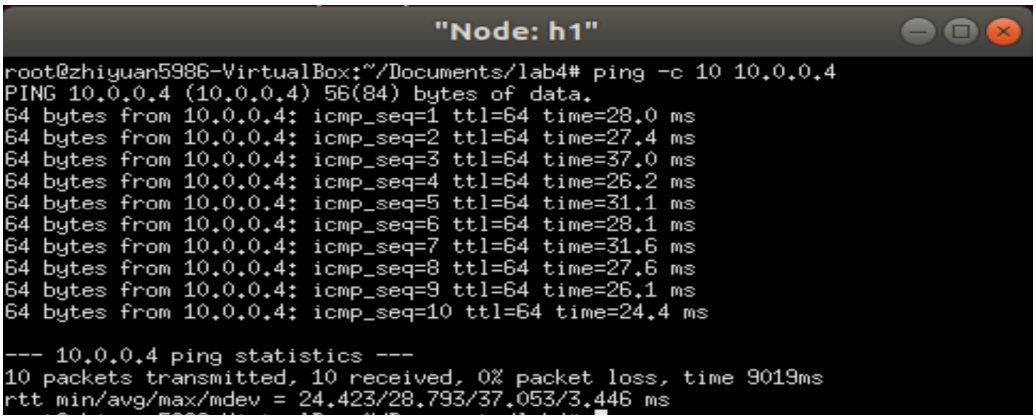

Figure 14: RTT between h1 and h4

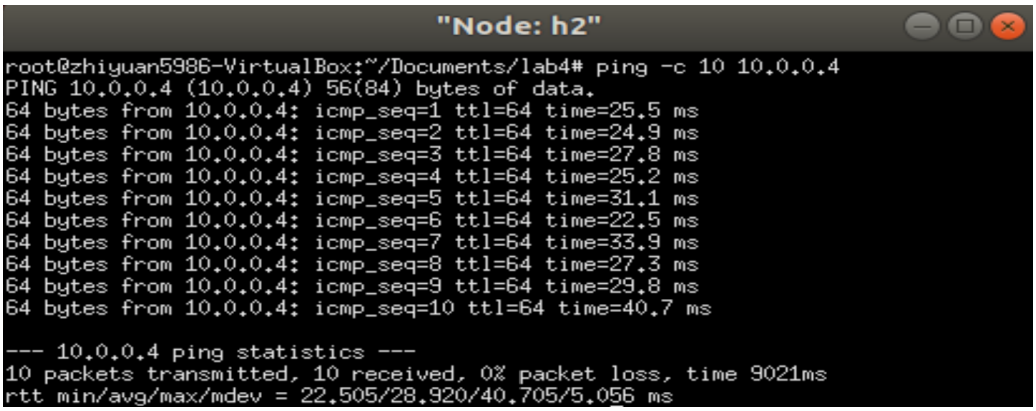

Figure 15: RTT between h2 and h4

Conclusion

In this lab, I connnect two LANs by VXLAN, and complete the tasks. After this lab, I understand the level of network more deeper, and find it subtle to build a link between two LANs. The limitation of MTU combine knowledge and pratice, which leaves a deep impression on me. Thanks to Prof.Jin and TAs.

References

[1] https://www.digitaltut.com/vxlan-tutorial/2

[2] https://blog.craftyun.cn/post/192.html

[3] https://en.wikipedia.org/wiki/Internet_Control_Message_Protocol

[4] https://en.wikipedia.org/wiki/IPv4

[5] https://en.wikipedia.org/wiki/Address_Resolution_Protocol

[6] https://en.wikipedia.org/wiki/User_Datagram_Protocol