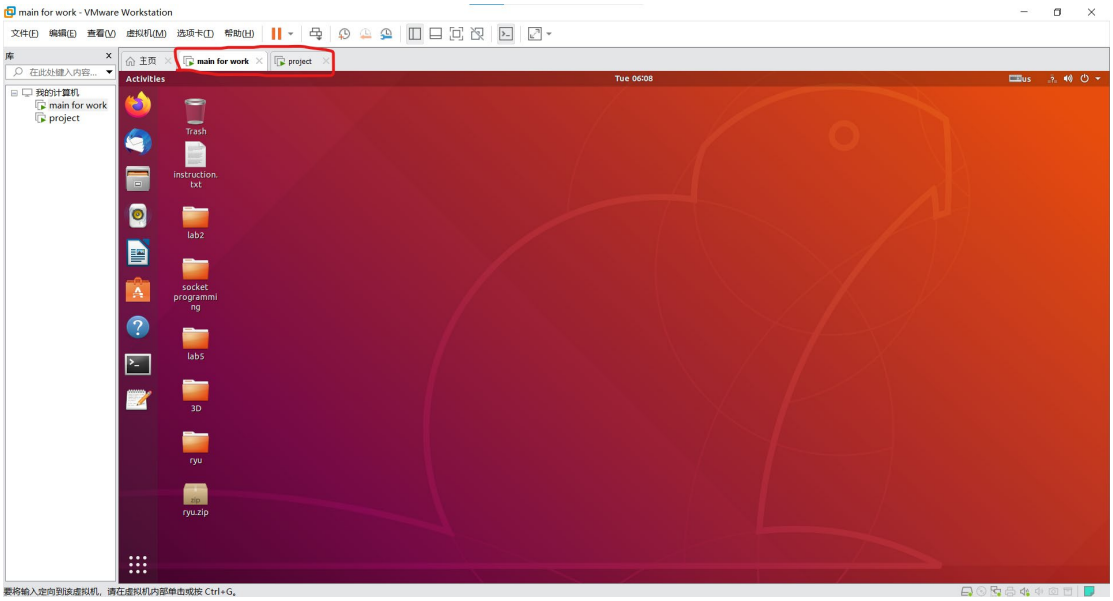
lab:vxlan

叶增渝 519030910168

一、网络设置

1.新建两个 Ubuntu 虚拟机：第一个虚拟机名为 main for work，第二个虚拟机名为 project



2.在第一个虚拟机 main for work 中，我们使用 Mininet 的默认启动命令创建两个 host 与一个 switch，并设置 h1 的 IP 地址为 10.0.0.1，设置 h2 的 IP 地址为 10.0.0.2，并将 s1 的 IP 地址设置为 10.0.0.101

```
spoilvoid@ubuntu: ~
File  Edit  View  Search  Terminal  Help
spoilvoid@ubuntu:~$ sudo mn
[sudo] password for spoilvoid:
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2
*** Adding switches:
s1
*** Adding links:
(h1, s1) (h2, s1)
*** Configuring hosts
h1 h2
*** Starting controller
c0
*** Starting 1 switches
s1 ...
*** Starting CLI:
mininet> h1 ifconfig h1-eth0 10.0.0.1 netmask 255.0.0.0
mininet> h2 ifconfig h2-eth0 10.0.0.2 netmask 255.0.0.0
mininet>
```

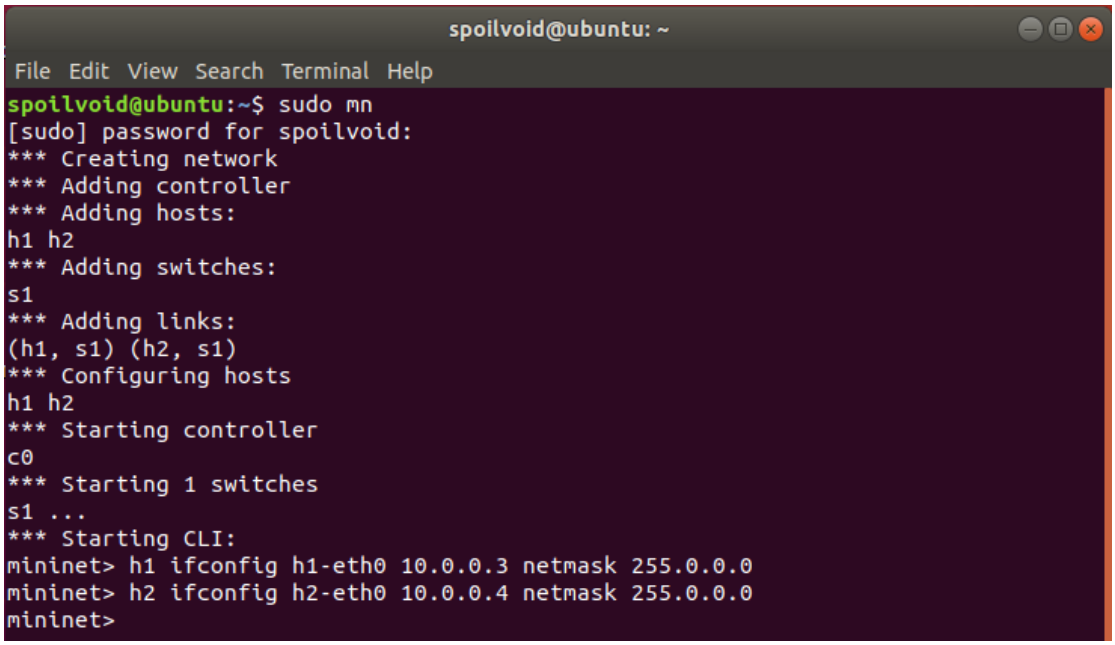
```
spoilvoid@ubuntu: ~
File  Edit  View  Search  Terminal  Help
spoilvoid@ubuntu:~$ sudo ifconfig s1 10.0.0.101/8 up
[sudo] password for spoilvoid:
spoilvoid@ubuntu:~$
```

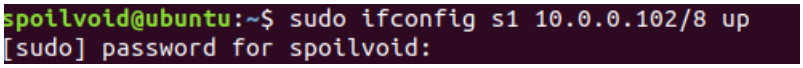
通过 ifconfig 命令可以看到 IP 地址已经得到修改

```
mininet> h1 ifconfig
h1-eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.0.0.1  netmask 255.0.0.0  broadcast 10.255.255.255
        inet6 fe80::fccd:7cff:fe53:2c30  prefixlen 64  scopeid 0x20<link>
        ether fe:cd:7c:53:2c:30  txqueuelen 1000  (Ethernet)
        RX packets 90  bytes 10075 (10.0 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 12  bytes 936 (936.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

mininet> h2 ifconfig
h2-eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.0.0.2  netmask 255.0.0.0  broadcast 10.255.255.255
        inet6 fe80::70bd:8ff:fea5:dbc6  prefixlen 64  scopeid 0x20<link>
        ether 72:bd:08:a5:db:c6  txqueuelen 1000  (Ethernet)
        RX packets 90  bytes 10075 (10.0 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 12  bytes 936 (936.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

```
s1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.0.0.101  netmask 255.0.0.0  broadcast 10.255.255.255
        inet6 fe80::a2:5cff:fecf:ec44  prefixlen 64  scopeid 0x20<link>
        ether 02:a2:5c:cf:ec:44  txqueuelen 1000  (Ethernet)
        RX packets 4  bytes 224 (224.0 B)
        RX errors 0  dropped 12  overruns 0  frame 0
        TX packets 51  bytes 5976 (5.9 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

3.在第二个虚拟机 project 中，我们使用 Mininet 的默认启动命令创建两个 host 与一个 switch，并设置 h1 的 IP 地址为 10.0.0.3，设置 h2 的 IP 地址为 10.0.0.4，并将 s1 的 IP 地址设置为 10.0.0.102



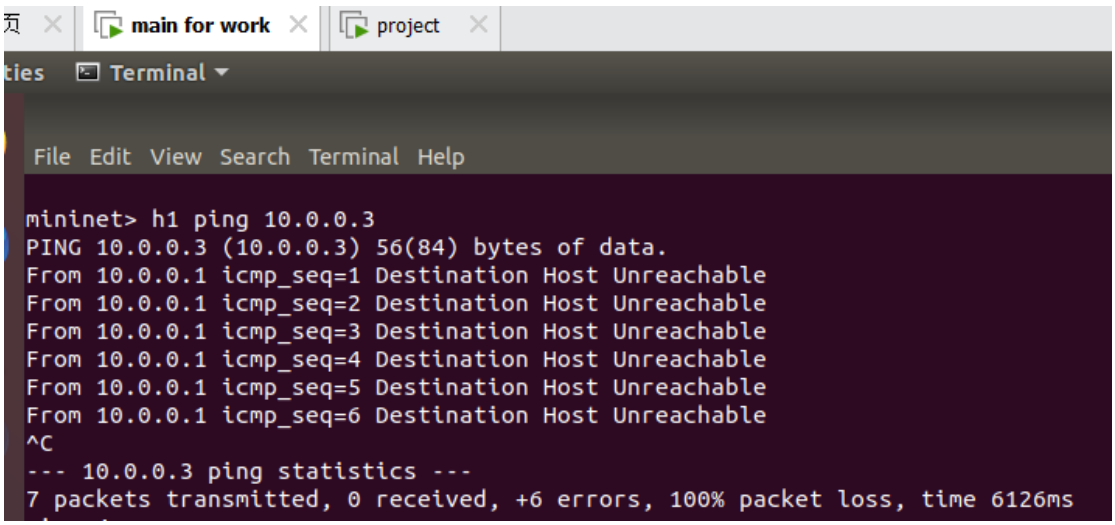


通过 ifconfig 命令可以看到 IP 地址已经得到修改

```
mininet> h1 ifconfig
h1-eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.0.0.3  netmask 255.0.0.0  broadcast 10.255.255.255
        inet6 fe80::c422:50ff:feea:c3f1  prefixlen 64  scopeid 0x20<link>
        ether c6:22:50:ea:c3:f1  txqueuelen 1000  (Ethernet)
        RX packets 35  bytes 3770 (3.7 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 11  bytes 866 (866.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

mininet> h2 ifconfig
h2-eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.0.0.4  netmask 255.0.0.0  broadcast 10.255.255.255
        inet6 fe80::a039:6eff:fece:6ad4  prefixlen 64  scopeid 0x20<link>
        ether a2:39:6e:ce:6a:d4  txqueuelen 1000  (Ethernet)
        RX packets 36  bytes 3856 (3.8 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 11  bytes 866 (866.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

```
s1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.0.0.102  netmask 255.0.0.0  broadcast 10.255.255.255
        inet6 fe80::78d0:87ff:fef5:b24c  prefixlen 64  scopeid 0x20<link>
        ether 7a:d0:87:f5:b2:4c  txqueuelen 1000  (Ethernet)
        RX packets 63  bytes 3628 (3.6 KB)
        RX errors 0  dropped 11  overruns 0  frame 0
        TX packets 75  bytes 8076 (8.0 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

4.此时我们从第一台虚拟机的 h1 ping 另一台虚拟机的 h1,发现无法 ping 通



5.然后我们对第一台虚拟机 main for work 设置网桥（我们 VMWare 的物理网卡为 ens33，所以我们设置 ens33 而非 eth0，由于之前设置过了，这里才会有 exist 的提示），并将其 IP 设置为 192.168.56.127

```
spoilvoid@ubuntu:~$ sudo ovs-vsctl add-br br1
ovs-vsctl: cannot create a bridge named br1 because a bridge named br1 already e
xists
spoilvoid@ubuntu:~$ sudo ifconfig ens33 0 up
spoilvoid@ubuntu:~$ sudo ovs-vsctl add-port br1 ens33
ovs-vsctl: cannot create a port named ens33 because a port named ens33 already e
xists on bridge br1
spoilvoid@ubuntu:~$ sudo ifconfig br1 192.168.56.127/24 up
```

6. 同理，然后我们对第二台虚拟机 project 设置网桥（我们 VMWare 的物理网卡为 ens33，所以我们设置 ens33 而非 eth0，由于之前设置过了，这里才会有 exist 的提示），并将其 IP 设置为 192.168.56.128

```
spoilvoid@ubuntu:~$ sudo ovs-vsctl add-br br1
ovs-vsctl: cannot create a bridge named br1 because a bridge named br1 already exists
spoilvoid@ubuntu:~$ sudo ifconfig ens33 0 up
spoilvoid@ubuntu:~$ sudo ovs-vsctl add-port br1 ens33
ovs-vsctl: cannot create a port named ens33 because a port named ens33 already exists on bridge br1
spoilvoid@ubuntu:~$ sudo ifconfig br1 192.168.56.128/24 up
```

7.最后使用 vxlan 命令创建 overlay 的 network

对第一台虚拟机 main for work

```
spoilvoid@ubuntu:~$ sudo ovs-vsctl add-port s1 vxlan0 -- set interface vxlan0 ty
pe=vxlan options:remote_ip=192.168.56.128
spoilvoid@ubuntu:~$
```

使用 ifconfig 检查得

```
spoilvoid@ubuntu:~$ ifconfig
br1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.56.127  netmask 255.255.255.0  broadcast 192.168.56.255
        inet6 fe80::20c:29ff:fec9:7883  prefixlen 64  scopeid 0x20<link>
        ether 00:0c:29:c9:78:83  txqueuelen 1000  (Ethernet)
        RX packets 171  bytes 11562 (11.5 KB)
        RX errors 0  dropped 736  overruns 0  frame 0
        TX packets 48  bytes 5736 (5.7 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

ens33: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet6 fe80::f2c1:4346:877a:53cb  prefixlen 64  scopeid 0x20<link>
        inet6 fe80::7a25:c401:e4d5:15a0  prefixlen 64  scopeid 0x20<link>
        inet6 fe80::ef3f:e1c8:3d17:e8ce  prefixlen 64  scopeid 0x20<link>
        ether 00:0c:29:c9:78:83  txqueuelen 1000  (Ethernet)
        RX packets 1473  bytes 111964 (111.9 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 421  bytes 36971 (36.9 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 2959  bytes 223707 (223.7 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 2959  bytes 223707 (223.7 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

s1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.0.0.101  netmask 255.0.0.0  broadcast 10.255.255.255
        inet6 fe80::a2:5cff:fecf:ec44  prefixlen 64  scopeid 0x20<link>
        ether 02:a2:5c:cf:ec:44  txqueuelen 1000  (Ethernet)
        RX packets 6  bytes 336 (336.0 B)
        RX errors 0  dropped 12  overruns 0  frame 0
        TX packets 54  bytes 6240 (6.2 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

s1-eth1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet6 fe80::f45f:90ff:fe45:322  prefixlen 64  scopeid 0x20<link>
        ether f6:5f:90:45:03:22  txqueuelen 1000  (Ethernet)
        RX packets 12  bytes 936 (936.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 92  bytes 10245 (10.2 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

```
s1-eth2: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet6 fe80::5027:1cff:fedc:f59d  prefixlen 64  scopeid 0x20<link>
        ether 52:27:1c:dc:f5:9d  txqueuelen 1000  (Ethernet)
        RX packets 12  bytes 936 (936.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 92  bytes 10245 (10.2 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

virbr0: flags=4099<UP,BROADCAST,MULTICAST>  mtu 1500
        inet 192.168.122.1  netmask 255.255.255.0  broadcast 192.168.122.255
        ether 52:54:00:55:cf:4f  txqueuelen 1000  (Ethernet)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

vxlan_sys_4789: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 65000
        inet6 fe80::e0ba:4dff:fe01:4a1a  prefixlen 64  scopeid 0x20<link>
        ether e2:ba:4d:01:4a:1a  txqueuelen 1000  (Ethernet)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

对第二台虚拟机 project 同理设置

```
spoilvoid@ubuntu:~$ sudo ovs-vsctl add-port s1 vxlan0 -- set interface vxlan0 type=vxlan options:remote_ip=192.168.56.127
```

通过 ifconfig 检查得

```
spoilvoid@ubuntu:~$ ifconfig
br1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.56.128  netmask 255.255.255.0  broadcast 192.168.56.255
        inet6 fe80::20c:29ff:fe33:e72e  prefixlen 64  scopeid 0x20<link>
        ether 00:0c:29:33:e7:2e  txqueuelen 1000  (Ethernet)
        RX packets 13811  bytes 695043 (695.0 KB)
        RX errors 0  dropped 400  overruns 0  frame 0
        TX packets 147  bytes 14374 (14.3 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

ens33: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.78.138  netmask 255.255.255.0  broadcast 192.168.78.255
        inet6 fe80::a1fb:c0bd:5a23:ce31  prefixlen 64  scopeid 0x20<link>
        inet6 fe80::538e:54c2:2e2b:78e5  prefixlen 64  scopeid 0x20<link>
        inet6 fe80::d724:9af5:1ab0:a045  prefixlen 64  scopeid 0x20<link>
        ether 00:0c:29:33:e7:2e  txqueuelen 1000  (Ethernet)
        RX packets 19715  bytes 1260592 (1.2 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 8572  bytes 532006 (532.0 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 37047  bytes 2679750 (2.6 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 37047  bytes 2679750 (2.6 MB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

s1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.0.0.102  netmask 255.0.0.0  broadcast 10.255.255.255
        inet6 fe80::78d0:87ff:fef5:b24c  prefixlen 64  scopeid 0x20<link>
        ether 7a:d0:87:f5:b2:4c  txqueuelen 1000  (Ethernet)
        RX packets 63  bytes 3628 (3.6 KB)
        RX errors 0  dropped 11  overruns 0  frame 0
        TX packets 75  bytes 8076 (8.0 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

s1-eth1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet6 fe80::c889:47ff:fec9:ef8f  prefixlen 64  scopeid 0x20<link>
        ether ca:89:47:c9:ef:8f  txqueuelen 1000  (Ethernet)
        RX packets 30  bytes 1944 (1.9 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 157  bytes 15598 (15.5 KB)
```

```
s1-eth2: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet6 fe80::8ceb:baff:fe93:afa5  prefixlen 64  scopeid 0x20<link>
        ether 8e:eb:ba:93:af:a5  txqueuelen 1000  (Ethernet)
        RX packets 17  bytes 1286 (1.2 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 163  bytes 15738 (15.7 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

vxlan_sys_4789: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 65000
        inet6 fe80::fa:1aff:fef8:7377  prefixlen 64  scopeid 0x20<link>
        ether 02:fa:1a:f8:73:77  txqueuelen 1000  (Ethernet)
        RX packets 40  bytes 2620 (2.6 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 40  bytes 2620 (2.6 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

8.我们从 10.0.0.1 ping 10.0.0.102，发现能够成功 ping 通

```
mininet> h1 ping 10.0.0.102
PING 10.0.0.102 (10.0.0.102) 56(84) bytes of data.
64 bytes from 10.0.0.102: icmp_seq=1 ttl=64 time=7.81 ms
64 bytes from 10.0.0.102: icmp_seq=2 ttl=64 time=10.7 ms
64 bytes from 10.0.0.102: icmp_seq=3 ttl=64 time=11.5 ms
64 bytes from 10.0.0.102: icmp_seq=4 ttl=64 time=6.27 ms
64 bytes from 10.0.0.102: icmp_seq=5 ttl=64 time=0.567 ms
^C
--- 10.0.0.102 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4009ms
rtt min/avg/max/mdev = 0.567/7.379/11.501/3.902 ms
```

由此，前期设置完成

二、homework

1.我们在从一台虚拟机 ping 另一台虚拟机时，使用 wireshark 检测 s1

1	0.000000000	10.0.0.1	10.0.0.4	ICMP	98	Echo (ping) request id=0x0

发现仅仅只捕获到一个 ICMP 包，用于获取 ping 的地址是否可达，发现底层无连接不再发送

而使用 wireshark 检测 ens33

18	5.344441173	Vmware_ff:58:1d	Vmware_33:e7:2e	ARP	60	192.168.78.2 is at 00:50:5
19	6.368272834	Vmware_33:e7:2e	Broadcast	ARP	60	Who has 192.168.78.2? Tell
20	6.368273133	Vmware_ff:58:1d	Vmware_33:e7:2e	ARP	60	192.168.78.2 is at 00:50:5
21	7.049581376	10.0.0.1	10.0.0.4	ICMP	148	Echo (ping) request id=0x
22	7.051236133	10.0.0.4	10.0.0.1	ICMP	148	Echo (ping) reply id=0x
23	7.392918688	Vmware_33:e7:2e	Broadcast	ARP	60	Who has 192.168.78.2? Tell
24	7.392918937	Vmware_ff:58:1d	Vmware_33:e7:2e	ARP	60	192.168.78.2 is at 00:50:5
25	8.051479405	10.0.0.1	10.0.0.4	ICMP	148	Echo (ping) request id=0x
26	8.052457260	10.0.0.4	10.0.0.1	ICMP	148	Echo (ping) reply id=0x
27	8.416235271	Vmware_33:e7:2e	Broadcast	ARP	60	Who has 192.168.78.2? Tell
28	8.416235528	Vmware_ff:58:1d	Vmware_33:e7:2e	ARP	60	192.168.78.2 is at 00:50:5

发现除了平常的 ARP 包，还会循环地得到从源地址 10.0.0.1 发来的 ICMP 包与 10.0.0.4 发回的 ICMP 回应，说明是从物理网卡，以 overlay network 发出的。

2.

（1）由于 192.168.56.127 和 192.168.56.128 本身都是架在物理网卡上的网桥，所以对这两个 IP 直接使用本虚拟机进行 iperf 即可

将第二个虚拟机 project 作为 iperf 的服务端

```
spoilvoid@ubuntu:~$ iperf -s
------------------------------------------------------------
Server listening on TCP port 5001
TCP window size: 85.3 KByte (default)
```

使用第一个虚拟机 main for work 作为客户端进行测试

```
spoilvoid@ubuntu:~$ iperf -c 192.168.56.128
------------------------------------------------------------
Client connecting to 192.168.56.128, TCP port 5001
TCP window size: 85.3 KByte (default)
------------------------------------------------------------
[  3] local 192.168.56.127 port 49196 connected with 192.168.56.128 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0-10.0 sec   679 MBytes   569 Mbits/sec
```

（2）由于都需要与 10.0.0.102 通信，所以将第二个虚拟机中的 s1 作为服务端

```
mininet> s1 iperf -s
------------------------------------------------------------
Server listening on TCP port 5001
TCP window size: 85.3 KByte (default)
------------------------------------------------------------
```

第一个虚拟机 main for work 的 h1、h2、s1 分别作为客户端进行测试

```
mininet> h1 iperf -c 10.0.0.102 -M 536
------------------------------------------------------------
Client connecting to 10.0.0.102, TCP port 5001
TCP window size: 85.3 KByte (default)
------------------------------------------------------------
[  3] local 10.0.0.1 port 56130 connected with 10.0.0.102 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0-10.0 sec   197 MBytes   165 Mbits/sec
mininet> h2 iperf -c 10.0.0.102 -M 536
------------------------------------------------------------
Client connecting to 10.0.0.102, TCP port 5001
TCP window size: 85.3 KByte (default)
------------------------------------------------------------
[  3] local 10.0.0.2 port 35892 connected with 10.0.0.102 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0-10.1 sec   167 MBytes   139 Mbits/sec
mininet> s1 iperf -c 10.0.0.102 -M 536
------------------------------------------------------------
Client connecting to 10.0.0.102, TCP port 5001
TCP window size: 85.3 KByte (default)
------------------------------------------------------------
[  3] local 10.0.0.101 port 53602 connected with 10.0.0.102 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0-10.0 sec   293 MBytes   246 Mbits/sec
```

最终我们得到

IP1	IP2	Bandwidth
192.168.56.127	192.168.56.128	569Mbps
10.0.0.1	10.0.0.102	165Mbps
10.0.0.2	10.0.0.102	139Mbps
10.0.0.101	10.0.0.102	246Mbps

结论：相比之下，两个网桥之间的带宽比 Mininet 下的组件进行通信的带宽要大上不少。这可能是由于相比网桥的 vxlan 连接，与其内部的组件连接还需要经过额外的内部链路，即不仅要 intra 还要 inter，此内部链路成为了 bottleneck，导致了网络带宽的下降。

3.

（1）在第一个虚拟机 main for work 直接 ping 第二台虚拟机的 192.168.56.128

```
spoilvoid@ubuntu:~$ ping 192.168.56.128
PING 192.168.56.128 (192.168.56.128) 56(84) bytes of data.
64 bytes from 192.168.56.128: icmp_seq=1 ttl=64 time=1.41 ms
64 bytes from 192.168.56.128: icmp_seq=2 ttl=64 time=0.664 ms
64 bytes from 192.168.56.128: icmp_seq=3 ttl=64 time=0.724 ms
64 bytes from 192.168.56.128: icmp_seq=4 ttl=64 time=1.23 ms
64 bytes from 192.168.56.128: icmp_seq=5 ttl=64 time=0.693 ms
64 bytes from 192.168.56.128: icmp_seq=6 ttl=64 time=1.08 ms
64 bytes from 192.168.56.128: icmp_seq=7 ttl=64 time=0.938 ms
64 bytes from 192.168.56.128: icmp_seq=8 ttl=64 time=1.19 ms
64 bytes from 192.168.56.128: icmp_seq=9 ttl=64 time=1.10 ms
64 bytes from 192.168.56.128: icmp_seq=10 ttl=64 time=0.899 ms
64 bytes from 192.168.56.128: icmp_seq=11 ttl=64 time=1.28 ms
^C
--- 192.168.56.128 ping statistics ---
11 packets transmitted, 11 received, 0% packet loss, time 10057ms
rtt min/avg/max/mdev = 0.664/1.021/1.410/0.244 ms
```

（2）在第一个虚拟机 main for work 的 h1、h2、s1 分别 ping 10.0.0.102

```
mininet> h1 ping 10.0.0.102
PING 10.0.0.102 (10.0.0.102) 56(84) bytes of data.
64 bytes from 10.0.0.102: icmp_seq=1 ttl=64 time=4.97 ms
64 bytes from 10.0.0.102: icmp_seq=2 ttl=64 time=2.98 ms
64 bytes from 10.0.0.102: icmp_seq=3 ttl=64 time=1.20 ms
64 bytes from 10.0.0.102: icmp_seq=4 ttl=64 time=1.15 ms
64 bytes from 10.0.0.102: icmp_seq=5 ttl=64 time=1.17 ms
64 bytes from 10.0.0.102: icmp_seq=6 ttl=64 time=1.13 ms
64 bytes from 10.0.0.102: icmp_seq=7 ttl=64 time=1.04 ms
64 bytes from 10.0.0.102: icmp_seq=8 ttl=64 time=1.18 ms
64 bytes from 10.0.0.102: icmp_seq=9 ttl=64 time=1.15 ms
64 bytes from 10.0.0.102: icmp_seq=10 ttl=64 time=1.07 ms
^C
--- 10.0.0.102 ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 9015ms
rtt min/avg/max/mdev = 1.041/1.706/4.971/1.220 ms
```

```
mininet> h2 ping 10.0.0.102
PING 10.0.0.102 (10.0.0.102) 56(84) bytes of data.
64 bytes from 10.0.0.102: icmp_seq=1 ttl=64 time=5.15 ms
64 bytes from 10.0.0.102: icmp_seq=2 ttl=64 time=2.54 ms
64 bytes from 10.0.0.102: icmp_seq=3 ttl=64 time=1.19 ms
64 bytes from 10.0.0.102: icmp_seq=4 ttl=64 time=1.06 ms
64 bytes from 10.0.0.102: icmp_seq=5 ttl=64 time=1.28 ms
64 bytes from 10.0.0.102: icmp_seq=6 ttl=64 time=1.09 ms
64 bytes from 10.0.0.102: icmp_seq=7 ttl=64 time=1.06 ms
64 bytes from 10.0.0.102: icmp_seq=8 ttl=64 time=1.09 ms
64 bytes from 10.0.0.102: icmp_seq=9 ttl=64 time=1.41 ms
64 bytes from 10.0.0.102: icmp_seq=10 ttl=64 time=1.13 ms
64 bytes from 10.0.0.102: icmp_seq=11 ttl=64 time=1.18 ms
^C
--- 10.0.0.102 ping statistics ---
11 packets transmitted, 11 received, 0% packet loss, time 10023ms
rtt min/avg/max/mdev = 1.063/1.657/5.153/1.178 ms
```

```
mininet> s1 ping 10.0.0.102
PING 10.0.0.102 (10.0.0.102) 56(84) bytes of data.
64 bytes from 10.0.0.102: icmp_seq=1 ttl=64 time=8.93 ms
64 bytes from 10.0.0.102: icmp_seq=2 ttl=64 time=3.61 ms
64 bytes from 10.0.0.102: icmp_seq=3 ttl=64 time=1.20 ms
64 bytes from 10.0.0.102: icmp_seq=4 ttl=64 time=1.13 ms
64 bytes from 10.0.0.102: icmp_seq=5 ttl=64 time=1.11 ms
64 bytes from 10.0.0.102: icmp_seq=6 ttl=64 time=1.12 ms
64 bytes from 10.0.0.102: icmp_seq=7 ttl=64 time=1.06 ms
64 bytes from 10.0.0.102: icmp_seq=8 ttl=64 time=3.85 ms
64 bytes from 10.0.0.102: icmp_seq=9 ttl=64 time=1.20 ms
64 bytes from 10.0.0.102: icmp_seq=10 ttl=64 time=1.14 ms
^C
--- 10.0.0.102 ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 9014ms
rtt min/avg/max/mdev = 1.064/2.440/8.937/2.396 ms
```

最终我们得到

IP1	IP2	Latency
192.168.56.127	192.168.56.128	1.021ms
10.0.0.1	10.0.0.102	1.706ms
10.0.0.2	10.0.0.102	1.657ms
10.0.0.101	10.0.0.102	2.440ms

结论：相比之下，两个虚拟机网桥比 MininetMininet 下的组件进行通信的网络延迟要小。这可能是因为上述额外内部链路的影响；除此之外，我们在（1）中看到了在 Mininet 内部的 ICMP 包，等待超时可能也会产生一定影响；可能由于 vxlan 包的 vxlan 头部是包含在 UDP 内部的，所以更高层次的解包可能导致了第一个 packet 网络延时的明显增大。