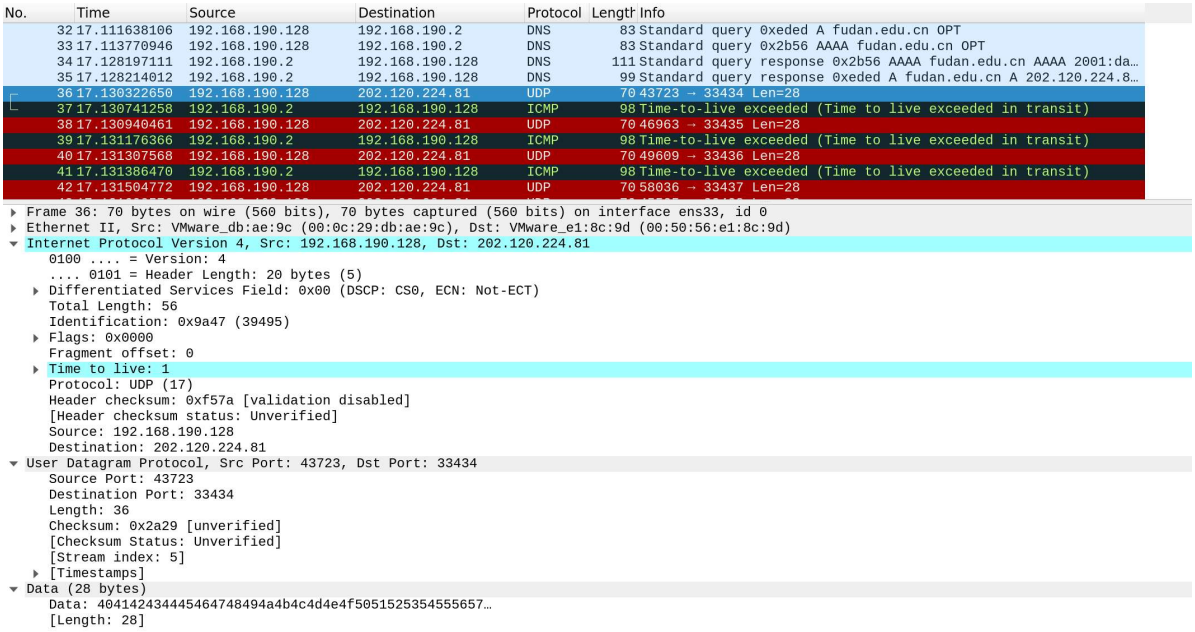
计算机网络 Lab5

一、实验任务

Part1

任务1：选择第一个发送的UDP包，观察：

第一个发送的UDP包：



1. 发送端的IP地址?

   192.168.190.128

2. 在IP header中，上层协议的数值是多少?

   17

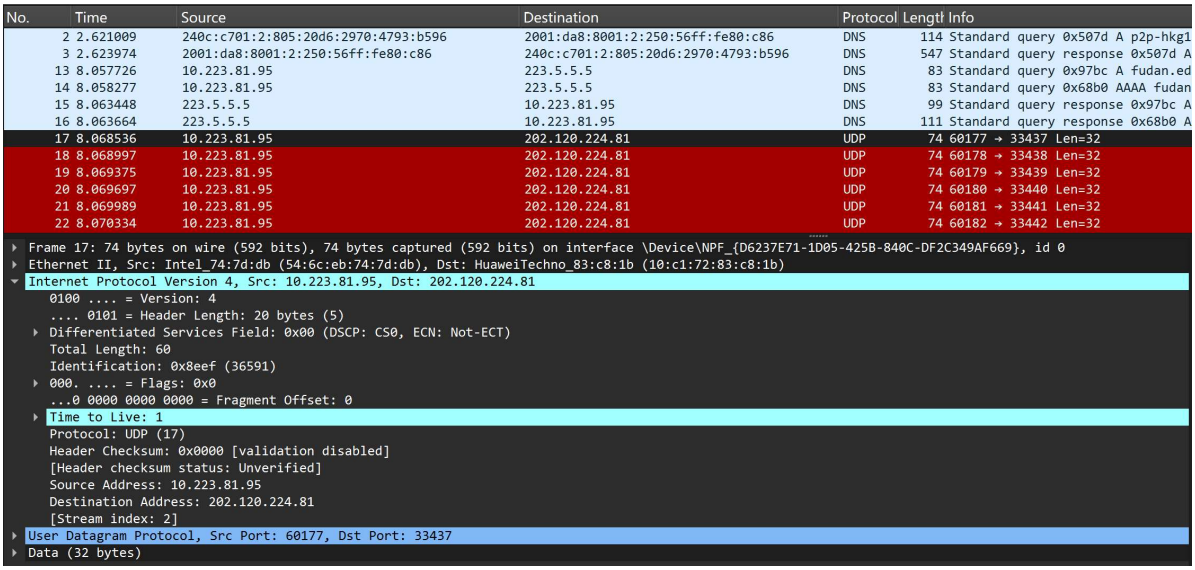
3. IP header有多少bytes? IP数据报数据载荷有多少bytes?

   `IPheader`有20bytes,数据载荷有36bytes

4. 该IP数据报是否分片?

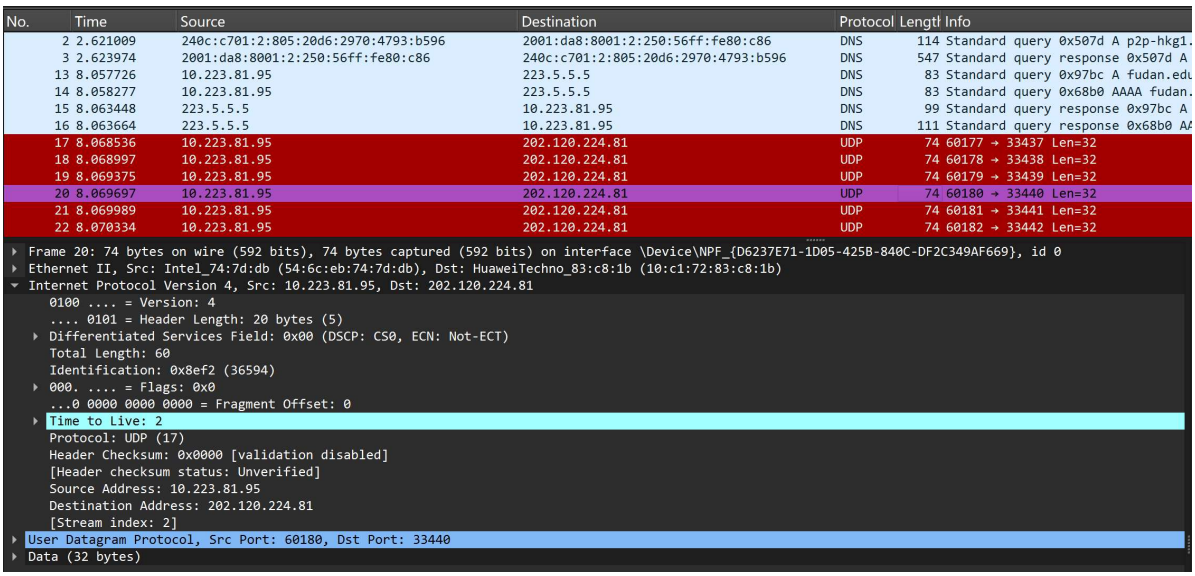
   "Fragment offset" 为 0，没有分片

任务2：观察连续的UDP包（穿插其他包），观察：



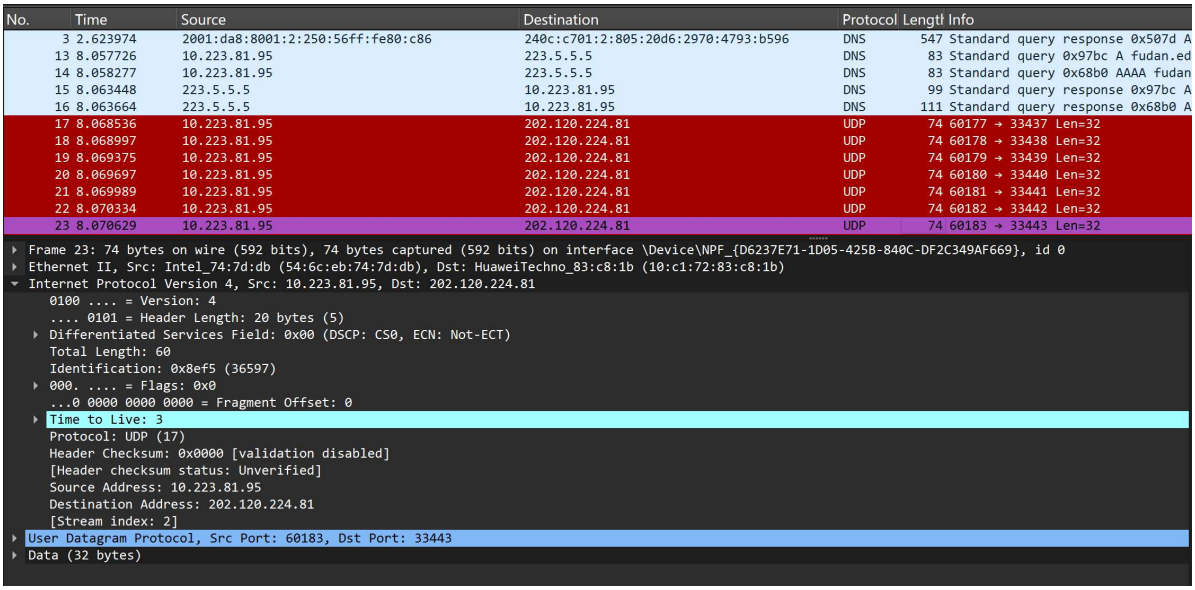
No.	Time	Source	Destination	Protocol	Length	Info
2	2.621009	240c:c701:2:805:20d6:2970:4793:b596	2001:da8:8001:2:250:56ff:fe80:c86	DNS	114	Standard query 0x507d A p2p-hkg1
3	2.623974	2001:da8:8001:2:250:56ff:fe80:c86	240c:c701:2:805:20d6:2970:4793:b596	DNS	547	Standard query response 0x507d A
13	8.057726	10.223.81.95	223.5.5.5	DNS	83	Standard query 0x97bc A fudan.ed
14	8.058277	10.223.81.95	223.5.5.5	DNS	83	Standard query 0x68b0 AAAA fudan
15	8.063448	223.5.5.5	10.223.81.95	DNS	99	Standard query response 0x97bc A
16	8.063664	223.5.5.5	10.223.81.95	DNS	111	Standard query response 0x68b0 A
17	8.068536	10.223.81.95	202.120.224.81	UDP	74	60177 → 33437 Len=32
18	8.068997	10.223.81.95	202.120.224.81	UDP	74	60178 → 33438 Len=32
19	8.069375	10.223.81.95	202.120.224.81	UDP	74	60179 → 33439 Len=32
20	8.069697	10.223.81.95	202.120.224.81	UDP	74	60180 → 33440 Len=32
21	8.069989	10.223.81.95	202.120.224.81	UDP	74	60181 → 33441 Len=32
22	8.070334	10.223.81.95	202.120.224.81	UDP	74	60182 → 33442 Len=32

```
▸ Frame 17: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface \Device\NPF_{D6237E71-1D05-425B-840C-DF2C349AF669}, id 0
▸ Ethernet II, Src: Intel_74:7d:db (54:6c:eb:74:7d:db), Dst: HuaweiTechno_83:c8:1b (10:c1:72:83:c8:1b)
▾ Internet Protocol Version 4, Src: 10.223.81.95, Dst: 202.120.224.81
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
  ▸ Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 60
    Identification: 0x8eef (36591)
  ▸ 000. .... = Flags: 0x0
    ...0 0000 0000 0000 = Fragment Offset: 0
  ▸ Time to Live: 1
    Protocol: UDP (17)
    Header Checksum: 0x0000 [validation disabled]
    [Header checksum status: Unverified]
    Source Address: 10.223.81.95
    Destination Address: 202.120.224.81
    [Stream index: 2]
▸ User Datagram Protocol, Src Port: 60177, Dst Port: 33437
▸ Data (32 bytes)
```



No.	Time	Source	Destination	Protocol	Length	Info
2	2.621009	240c:c701:2:805:20d6:2970:4793:b596	2001:da8:8001:2:250:56ff:fe80:c86	DNS	114	Standard query 0x507d A p2p-hkg1.
3	2.623974	2001:da8:8001:2:250:56ff:fe80:c86	240c:c701:2:805:20d6:2970:4793:b596	DNS	547	Standard query response 0x507d A
13	8.057726	10.223.81.95	223.5.5.5	DNS	83	Standard query 0x97bc A fudan.edu
14	8.058277	10.223.81.95	223.5.5.5	DNS	83	Standard query 0x68b0 AAAA fudan.
15	8.063448	223.5.5.5	10.223.81.95	DNS	99	Standard query response 0x97bc A
16	8.063664	223.5.5.5	10.223.81.95	DNS	111	Standard query response 0x68b0 AA
17	8.068536	10.223.81.95	202.120.224.81	UDP	74	60177 → 33437 Len=32
18	8.068997	10.223.81.95	202.120.224.81	UDP	74	60178 → 33438 Len=32
19	8.069375	10.223.81.95	202.120.224.81	UDP	74	60179 → 33439 Len=32
20	8.069697	10.223.81.95	202.120.224.81	UDP	74	60180 → 33440 Len=32
21	8.069989	10.223.81.95	202.120.224.81	UDP	74	60181 → 33441 Len=32
22	8.070334	10.223.81.95	202.120.224.81	UDP	74	60182 → 33442 Len=32

```
▸ Frame 20: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface \Device\NPF_{D6237E71-1D05-425B-840C-DF2C349AF669}, id 0
▸ Ethernet II, Src: Intel_74:7d:db (54:6c:eb:74:7d:db), Dst: HuaweiTechno_83:c8:1b (10:c1:72:83:c8:1b)
▾ Internet Protocol Version 4, Src: 10.223.81.95, Dst: 202.120.224.81
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
  ▸ Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 60
    Identification: 0x8ef2 (36594)
  ▸ 000. .... = Flags: 0x0
    ...0 0000 0000 0000 = Fragment Offset: 0
  ▸ Time to Live: 2
    Protocol: UDP (17)
    Header Checksum: 0x0000 [validation disabled]
    [Header checksum status: Unverified]
    Source Address: 10.223.81.95
    Destination Address: 202.120.224.81
    [Stream index: 2]
▸ User Datagram Protocol, Src Port: 60180, Dst Port: 33440
▸ Data (32 bytes)
```



No.	Time	Source	Destination	Protocol	Length	Info
3	2.623974	2001:da8:8001:2:250:56ff:fe80:c86	240c:c701:2:805:20d6:2970:4793:b596	DNS	547	Standard query response 0x507d A
13	8.057726	10.223.81.95	223.5.5.5	DNS	83	Standard query 0x97bc A fudan.ed
14	8.058277	10.223.81.95	223.5.5.5	DNS	83	Standard query 0x68b0 AAAA fudan
15	8.063448	223.5.5.5	10.223.81.95	DNS	99	Standard query response 0x97bc A
16	8.063664	223.5.5.5	10.223.81.95	DNS	111	Standard query response 0x68b0 A
17	8.068536	10.223.81.95	202.120.224.81	UDP	74	60177 → 33437 Len=32
18	8.068997	10.223.81.95	202.120.224.81	UDP	74	60178 → 33438 Len=32
19	8.069375	10.223.81.95	202.120.224.81	UDP	74	60179 → 33439 Len=32
20	8.069697	10.223.81.95	202.120.224.81	UDP	74	60180 → 33440 Len=32
21	8.069989	10.223.81.95	202.120.224.81	UDP	74	60181 → 33441 Len=32
22	8.070334	10.223.81.95	202.120.224.81	UDP	74	60182 → 33442 Len=32
23	8.070629	10.223.81.95	202.120.224.81	UDP	74	60183 → 33443 Len=32

```
▸ Frame 23: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface \Device\NPF_{D6237E71-1D05-425B-840C-DF2C349AF669}, id 0
▸ Ethernet II, Src: Intel_74:7d:db (54:6c:eb:74:7d:db), Dst: HuaweiTechno_83:c8:1b (10:c1:72:83:c8:1b)
▾ Internet Protocol Version 4, Src: 10.223.81.95, Dst: 202.120.224.81
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
  ▸ Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 60
    Identification: 0x8ef5 (36597)
  ▸ 000. .... = Flags: 0x0
    ...0 0000 0000 0000 = Fragment Offset: 0
  ▸ Time to Live: 3
    Protocol: UDP (17)
    Header Checksum: 0x0000 [validation disabled]
    [Header checksum status: Unverified]
    Source Address: 10.223.81.95
    Destination Address: 202.120.224.81
    [Stream index: 2]
▸ User Datagram Protocol, Src Port: 60183, Dst Port: 33443
▸ Data (32 bytes)
```

> 注：上述截图为虚拟机外windows主机的wirshark截图，因此源IP和报文总长度与任务一的截图不同。这是由于经过了NAT网关的转换。NAT模式下虚拟机无法接受到目的地址fudan.edu.cn传来的已经到达的数据包，导致无法显示和测量，但实际上数据报是成功的发送到目的地了的，在外面的主机是可以正常查看的。尝试了换用**桥接模式**，发现虚拟机无法连上互联网，更加无法完成实验。

1. IP数据报中哪些字段不断变化，哪些保持不变？

   不断变化：Identification、TTL。

   其他字段保持不变

2. 为什么有些字段不断变化，为什么有些不变？

   变化：

   Time to Live：每次没有达到目的地址会增加TTL值

   Identification：为单独确定每一个包的标识，应该不同包的ID互不相同

   不变：

   Header Length：由协议已经确定 保持不变

   Total Length：在发送的时候由参数确定，保持不变

   Protocol：UDP协议没有改变 保持不变

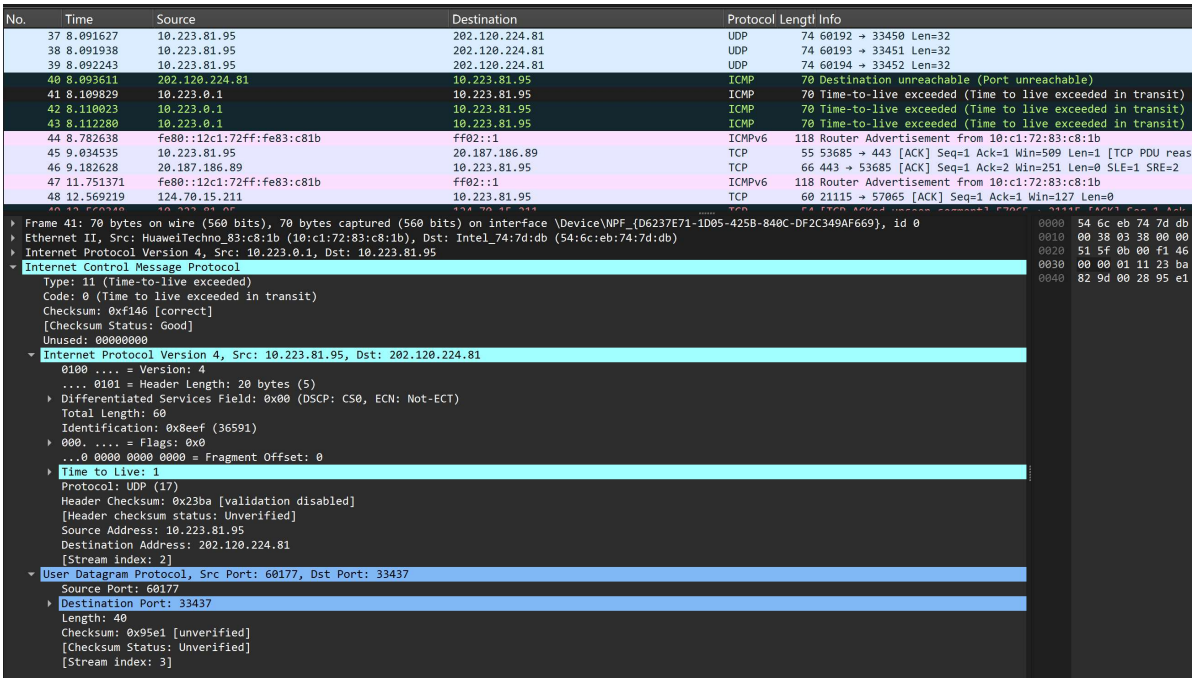
   Flags：分别标志Reserved bit， Don't Fragement和more Fragment， 连续UDP包中是否分片并没有改变 因此保持不变

   Source Address/Destination Address：两边地址都在traceroute命令发出时已经确定， 不会改变

3. 列出连续IP数据报中的标识序列。

   36591-36611

任务3：观察收到的第一个TTL-exceeded replies，观察：



1. 标识字段与TTL字段分别是多少？

   标识字段：36591

   TTL字段：1

2. 收到的所有TTL-exceeded replies中，这两个字段是否不变？为什么？

   会发生变化。

   TTL 值的变化：

   在 **TTL-exceeded replies** 的序列中，发送方会逐步增加原始数据包的 TTL（例如从 1 开始，然后逐渐递增），以测试通过的跳数，直到到达目标。因此，在返回的 ICMP TTL 超时回复消息中， TTL 值会发生变化。

   标识值的变化：

   IP 标识值（ID）是为每个 IP 数据包生成的唯一标识，以便在数据包分片时可以重新组装。不同的 **TTL-exceeded replies** 是不同路由器生成的 ICMP 报文，所以每个报文的 IP ID 值是不同的，由相应的路由器独立生成。

Part2

任务1：定制化拓扑

要求：

（1）定制化上述拓扑，并将脚本文件命名为customized_topo.py，提交文件中需包含该python文件

启动命令：

```
sudo mn --custom customized_topo.py --topo mytopo --link tc
```

（2）利用iperf验证端到端带宽，并截图，下面提供了各主机间端到端带宽的参考范围：

- H1 – H2: 10Mbps with ~12ms latency

```
mininet> iperf h1 h2
*** Iperf: testing TCP bandwidth between h1 and h2
*** Results: ['9.53 Mbits/sec', '11.8 Mbits/sec']
```

```
mininet> h1 ping h2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
64 bytes from 10.0.0.2: icmp_seq=1 ttl=64 time=33.4 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=26.8 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=25.4 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=25.0 ms
64 bytes from 10.0.0.2: icmp_seq=5 ttl=64 time=24.8 ms
64 bytes from 10.0.0.2: icmp_seq=6 ttl=64 time=25.0 ms
64 bytes from 10.0.0.2: icmp_seq=7 ttl=64 time=25.4 ms
^C
--- 10.0.0.2 ping statistics ---
7 packets transmitted, 7 received, 0% packet loss, time 6008ms
rtt min/avg/max/mdev = 24.755/26.528/33.419/2.879 ms
```

  可以看到利用iperf测得的带宽在10Mbps左右，ping测得的RTT在24ms左右，即latency为12ms左右，符合预期。

- H2 – H4: <<16Mbps with ~22ms latency

```
mininet> iperf h2 h4
*** Iperf: testing TCP bandwidth between h2 and h4
*** Results: ['800 Kbits/sec', '850 Kbits/sec']
mininet> h2 ping h4
PING 10.0.0.4 (10.0.0.4) 56(84) bytes of data.
64 bytes from 10.0.0.4: icmp_seq=2 ttl=64 time=58.2 ms
64 bytes from 10.0.0.4: icmp_seq=3 ttl=64 time=47.5 ms
64 bytes from 10.0.0.4: icmp_seq=4 ttl=64 time=46.3 ms
64 bytes from 10.0.0.4: icmp_seq=5 ttl=64 time=45.6 ms
64 bytes from 10.0.0.4: icmp_seq=7 ttl=64 time=45.8 ms
64 bytes from 10.0.0.4: icmp_seq=9 ttl=64 time=45.8 ms
64 bytes from 10.0.0.4: icmp_seq=10 ttl=64 time=46.7 ms
^C
--- 10.0.0.4 ping statistics ---
10 packets transmitted, 7 received, 30% packet loss, time 9059ms
rtt min/avg/max/mdev = 45.647/47.983/58.234/4.227 ms
mininet>
```

可以看到利用iperf测得的带宽在800kbps左右，ping测得的RTT在44ms左右，即latency为22ms左右，符合预期。

- H3 – H4: 10Mbps with ~12ms latency

```
mininet> iperf h3 h4
*** Iperf: testing TCP bandwidth between h3 and h4
*** Results: ['9.54 Mbits/sec', '11.8 Mbits/sec']
mininet> h3 ping h4
PING 10.0.0.4 (10.0.0.4) 56(84) bytes of data.
64 bytes from 10.0.0.4: icmp_seq=1 ttl=64 time=36.4 ms
64 bytes from 10.0.0.4: icmp_seq=2 ttl=64 time=26.5 ms
64 bytes from 10.0.0.4: icmp_seq=3 ttl=64 time=25.1 ms
64 bytes from 10.0.0.4: icmp_seq=4 ttl=64 time=24.9 ms
64 bytes from 10.0.0.4: icmp_seq=5 ttl=64 time=26.2 ms
64 bytes from 10.0.0.4: icmp_seq=6 ttl=64 time=25.6 ms
64 bytes from 10.0.0.4: icmp_seq=7 ttl=64 time=25.2 ms
64 bytes from 10.0.0.4: icmp_seq=8 ttl=64 time=25.1 ms
^C
--- 10.0.0.4 ping statistics ---
8 packets transmitted, 8 received, 0% packet loss, time 7011ms
rtt min/avg/max/mdev = 24.945/26.881/36.356/3.620 ms
mininet>
```

可以看到利用iperf测得的带宽在10Mbps左右，ping测得的RTT在24ms左右，即latency为12ms左右，符合预期。

（3）通过`sudo mn --custom ./customized_topo.py --topo mytopo --test pingall --link tc`指令检验，并截图。描述一下出现的现象，并阐述一下原因。

```
*** Configuring hosts
h1 h2 h3 h4
*** Starting controller
c0
*** Starting 2 switches
s1 s2 ...(10.00Mbit 2ms delay 0.00000% loss) (20.00Mbit 10ms delay 0.00000% loss
) (20.00Mbit 2ms delay 10.00000% loss) (20.00Mbit 2ms delay 10.00000% loss) (10.
00Mbit 2ms delay 0.00000% loss) (20.00Mbit 10ms delay 0.00000% loss)
*** Waiting for switches to connect
s1 s2
*** Ping: testing ping reachability
h1 -> h2 h3 h4
h2 -> h1 h3 h4
h3 -> h1 h2 h4
h4 -> h1 h2 h3
*** Results: 0% dropped (12/12 received)
*** Stopping 1 controllers
c0
*** Stopping 5 links
.....
*** Stopping 2 switches
s1 s2
*** Stopping 4 hosts
h1 h2 h3 h4
*** Done
completed in 16.146 seconds
```

```
*** Configuring hosts
h1 h2 h3 h4
*** Starting controller
c0
*** Starting 2 switches
s1 s2 ...(10.00Mbit 2ms delay 0.00000% loss) (20.00Mbit 10ms delay 0.00000% loss
) (20.00Mbit 2ms delay 10.00000% loss) (20.00Mbit 2ms delay 10.00000% loss) (10.
00Mbit 2ms delay 0.00000% loss) (20.00Mbit 10ms delay 0.00000% loss)
*** Waiting for switches to connect
s1 s2
*** Ping: testing ping reachability
h1 -> h2 X h4
h2 -> h1 h3 h4
h3 -> h1 h2 h4
h4 -> h1 X h3
*** Results: 16% dropped (10/12 received)
*** Stopping 1 controllers
c0
*** Stopping 5 links
.....
*** Stopping 2 switches
s1 s2
*** Stopping 4 hosts
h1 h2 h3 h4
*** Done
completed in 37.298 seconds
```

运行`sudo mn --custom ./customized_topo.py --topo mytopo --test pingall --link tc`指令测试连通性，发现大多数情况下四个节点之间彼此都能ping通。但有时h1/h2与h3/h4之间会出现X，说明无法连通，这是因为S1与S2之间的链路有10%loss的丢包率，出现dropped是正常现象， 测试通过。

任务2：在虚拟终端上执行任务

利用`iperf`生成TCP流

- TCP Flow 1: 由h1按最大速率发向h3，持续时间为T=0sec~20sec
- TCP Flow 2: 由h2按最大速率发向h4，持续时间为T=10sec~30sec

要求：

（1） 利用python实现上述功能，并将脚本文件命名为host_iperf.py ，提交文件需包含该python文件。

运行命令：

```
sudo python host_iperf.py
```

（2）提交 Flow 1 和 Flow 2 带宽测试截图或文本文件，要求每0.5s测量一次。

```
------------------------------------------------------------
Client connecting to 10.0.0.3, TCP port 5001
TCP window size: 85.3 KByte (default)
------------------------------------------------------------
[  3] local 10.0.0.1 port 50444 connected with 10.0.0.3 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0- 0.5 sec   656 KBytes  10.7 Mbits/sec
[  3]  0.5- 1.0 sec   191 KBytes  3.13 Mbits/sec
[  3]  1.0- 1.5 sec  0.00 Bytes  0.00 bits/sec
[  3]  1.5- 2.0 sec  63.6 KBytes  1.04 Mbits/sec
[  3]  2.0- 2.5 sec  63.6 KBytes  1.04 Mbits/sec
[  3]  2.5- 3.0 sec   255 KBytes  4.17 Mbits/sec
[  3]  3.0- 3.5 sec  63.6 KBytes  1.04 Mbits/sec
[  3]  3.5- 4.0 sec   127 KBytes  2.09 Mbits/sec
[  3]  4.0- 4.5 sec   191 KBytes  3.13 Mbits/sec
[  3]  4.5- 5.0 sec  63.6 KBytes  1.04 Mbits/sec
[  3]  5.0- 5.5 sec  63.6 KBytes  1.04 Mbits/sec
[  3]  5.5- 6.0 sec  0.00 Bytes  0.00 bits/sec
[  3]  6.0- 6.5 sec  63.6 KBytes  1.04 Mbits/sec
[  3]  6.5- 7.0 sec  0.00 Bytes  0.00 bits/sec
[  3]  7.0- 7.5 sec  63.6 KBytes  1.04 Mbits/sec
[  3]  7.5- 8.0 sec  63.6 KBytes  1.04 Mbits/sec
[  3]  8.0- 8.5 sec  63.6 KBytes  1.04 Mbits/sec
[  3]  8.5- 9.0 sec   127 KBytes  2.09 Mbits/sec
[  3]  9.0- 9.5 sec   191 KBytes  3.13 Mbits/sec
[  3]  9.5-10.0 sec   127 KBytes  2.09 Mbits/sec
[  3] 10.0-10.5 sec  63.6 KBytes  1.04 Mbits/sec
[  3] 10.5-11.0 sec  0.00 Bytes  0.00 bits/sec
[  3] 11.0-11.5 sec  63.6 KBytes  1.04 Mbits/sec
[  3] 11.5-12.0 sec  0.00 Bytes  0.00 bits/sec
[  3] 12.0-12.5 sec  63.6 KBytes  1.04 Mbits/sec
[  3] 12.5-13.0 sec   191 KBytes  3.13 Mbits/sec
[  3] 13.0-13.5 sec   127 KBytes  2.09 Mbits/sec
[  3] 13.5-14.0 sec  63.6 KBytes  1.04 Mbits/sec
```

```
------------------------------------------------------------
Client connecting to 10.0.0.4, TCP port 5001
TCP window size: 85.3 KByte (default)
------------------------------------------------------------
[  3] local 10.0.0.2 port 39540 connected with 10.0.0.4 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0- 0.5 sec   384 KBytes  6.29 Mbits/sec
[  3]  0.5- 1.0 sec  29.0 KBytes   474 Kbits/sec
[  3]  1.0- 1.5 sec  63.6 KBytes  1.04 Mbits/sec
[  3]  1.5- 2.0 sec  63.6 KBytes  1.04 Mbits/sec
[  3]  2.0- 2.5 sec  63.6 KBytes  1.04 Mbits/sec
[  3]  2.5- 3.0 sec  0.00 Bytes  0.00 bits/sec
[  3]  3.0- 3.5 sec  63.6 KBytes  1.04 Mbits/sec
[  3]  3.5- 4.0 sec  0.00 Bytes  0.00 bits/sec
[  3]  4.0- 4.5 sec  0.00 Bytes  0.00 bits/sec
[  3]  4.5- 5.0 sec  63.6 KBytes  1.04 Mbits/sec
[  3]  5.0- 5.5 sec  0.00 Bytes  0.00 bits/sec
[  3]  5.5- 6.0 sec  0.00 Bytes  0.00 bits/sec
[  3]  6.0- 6.5 sec  0.00 Bytes  0.00 bits/sec
[  3]  6.5- 7.0 sec  63.6 KBytes  1.04 Mbits/sec
[  3]  7.0- 7.5 sec  0.00 Bytes  0.00 bits/sec
[  3]  7.5- 8.0 sec  0.00 Bytes  0.00 bits/sec
[  3]  8.0- 8.5 sec  63.6 KBytes  1.04 Mbits/sec
[  3]  8.5- 9.0 sec   127 KBytes  2.09 Mbits/sec
[  3]  9.0- 9.5 sec   127 KBytes  2.09 Mbits/sec
[  3]  9.5-10.0 sec  0.00 Bytes  0.00 bits/sec
[  3] 10.0-10.5 sec  63.6 KBytes  1.04 Mbits/sec
[  3] 10.5-11.0 sec  63.6 KBytes  1.04 Mbits/sec
[  3] 11.0-11.5 sec  63.6 KBytes  1.04 Mbits/sec
[  3] 11.5-12.0 sec  63.6 KBytes  1.04 Mbits/sec
[  3] 12.0-12.5 sec  0.00 Bytes  0.00 bits/sec
[  3] 12.5-13.0 sec  0.00 Bytes  0.00 bits/sec
[  3] 13.0-13.5 sec  63.6 KBytes  1.04 Mbits/sec
[  3] 13.5-14.0 sec  0.00 Bytes  0.00 bits/sec
```

> Flow 1 和 Flow 2 带宽测试部分截图，具体文件保存至 flow1_10,flow2_10

（3）请描述一下出现的现象，并尝试解释一下原因。

整体上看，flow1和flow2都没有达到h1和h2的最大发送速率，只有在刚开始发送时比较接近。甚至在很多时间段出现了flow1/2带宽为0的情况。这主要是受到各链路的带宽、延迟和s1-s2链路10%丢包率的限制，网络链路上发生了丢包、拥塞或其他资源竞争情况。从时间上看，在0-10s和20s到30s这两段时间内，由于只有一个flow在使用s1-s2链路，Flow的带宽较稳定，维持在1-2Mbps之间。而在10-20s期间，Flow 1 和 Flow 2 是并行传输的，也就是说，两个流在这个时段共享网络带宽资源。在这种情况下，带宽的变化受到**单**

个流的带宽限制，并行流的带宽竞争以及**TCP 拥塞控制和流量调节**等因素的影响，导致带宽出现了明显的下降和抖动。

（4）尝试修改 Switch S1 和 Switch S2 之间链路的丢包率，重复任务二，观察并描述在不同丢包率下出现的现象，尝试利用所学的知识解释一下原因。

> 修改S1-S2之间的丢包率后测得的文件保存为flow1_x，flow2_x，x为丢包率

S1-S2:0%，在丢包率为0%流量较稳定,Flow 1的最大带宽可达约14.7 Mbits/sec，表现出一定的带宽稳定性，Flow 2带宽波动也较为平稳，最大带宽可达约27.1 Mbits/sec，且随着时间推移带宽较为均衡，没有显著的波动。

在测试期间，TCP流量几乎一直保持在接近最大带宽附近，只有在流量的起始和末尾阶段有较小的波动。表现出流量的“波动”是由TCP协议的“慢启动”和“拥塞避免”机制引起的。TCP在建立连接初期会使用较小的窗口，随着数据包的成功发送，它逐渐增加窗口大小，导致带宽逐步上升。

原因解释

丢包率为0%时，TCP连接的带宽较为稳定，因为没有丢包导致TCP重传和窗口缩小。TCP协议通过慢启动和拥塞控制机制适应网络的带宽，并动态调整窗口大小，从而在大部分时间内保持稳定的带宽。

S1-S2:20%，丢包率为20%时，和丢包率为10%的情况相比，出现了更频繁更大范围的带宽为0的停顿现象，并且绝大部分时间的速率低于1 Mbit/sec，流量间断较为显著，速率相对较低，最终的平均速率显著下降，只有100-300 Kbps

原因解释：

在丢包率为10%时，TCP的丢包检测和重传机制通常能够有效地恢复连接，导致速率出现波动，但总体影响较为温和。而在丢包率为20%时，由于更高的丢包率，TCP在恢复时需要更多的重传和调整，导致停顿和重传时间更长，速率显著下降。

在丢包率为20%的情况下，由于更多的数据包丢失，发送端的窗口会迅速收缩，TCP会暂停发送数据直到确认之前的数据包被成功接收，因此观察到更多的停顿时间，最终吞吐量显著下降。