

復旦大學

数据通信与计算机网络 Lab4: 网络层

王傲

15300240004

数据通信与计算机网络

COMP130017. 01

指导教师：肖晓春

**目录:**

0. 目录	.....	2
1. IP 协议	.....	3
1.1	.....	3
1.2	.....	5
1.3	.....	6
2. ping	.....	7
2.1	.....	7
2.2	.....	9
2.3	.....	9
3. traceroute	.....	10
3.1	.....	11
3.2	.....	12
3.3	.....	13
3.4	.....	14
3.5	.....	15
4. 研究与讨论	.....	16
4.1	.....	16
4.2	.....	16
5. 参考资料	.....	18

## 1. IP 协议

实验步骤：

实验主机为 MacBook Pro Mid 15'，Wireshark 版本为 2.4.1，使用 VMware Fusion  
10.0.1 装载 Windows 10 1709。服务器操作系统为 Windows Server 2012 R2。

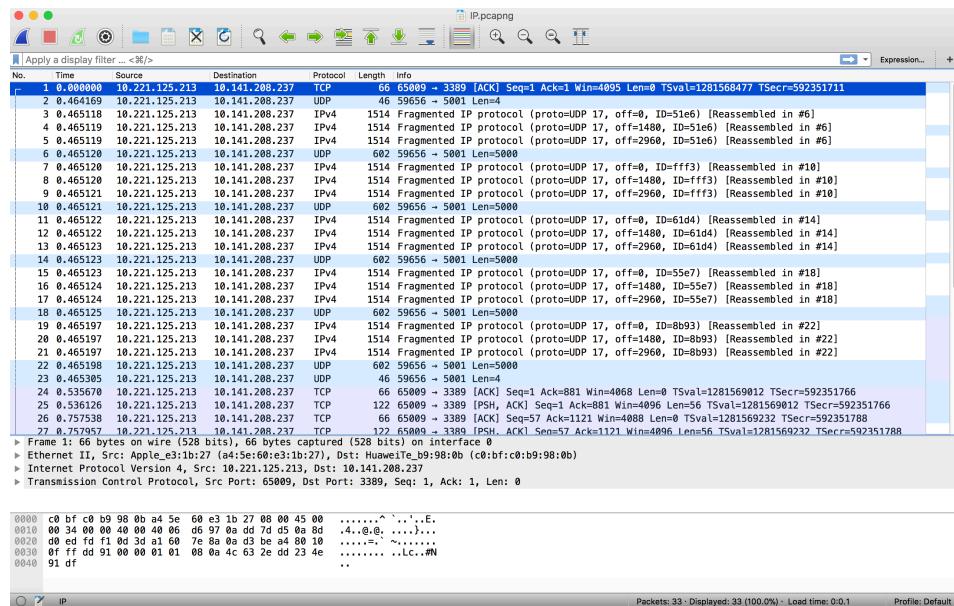
主机和服务器均连接在复旦校园网内部，主机 IP 地址为 10.221.125.213，服务器 IP 地址为 10.141.208.237。由于 DHCP 协议的原因，重新连接后主机的 IP 地址可能会发生改变。

使用 PCATTCP 进行相互通讯，分析 IP 协议的工作方式（分片与重组）。

实验分析：

**1.1 观察分析IP分组中片偏移量(Fragment offset)字段的含义，复习标志字段(Flags)各位所代表的含义，指出哪位是表示还有分片，并观察各个IP分组中的该字段是否如此。**

捕获的包内容如下：



在 IP 协议中，IP 报文可能会因为超过某个链路的 MTU 而导致分片，所以 IP 协议需要有一种将分片的报文重新组装的机制。在 IP 协议中，用 16 bit 的标识 (identifier) 确定报文属于同一个原报文；使用标志 (flags) 来判断是否分段；使用片偏移量 (offset) 来确定数据起始位置。

这里，13 bit 的片偏移量表示分片后的报文的起始字节在原报文中的序号：

```

▼ Internet Protocol Version 4, Src: 10.221.125.213, Dst: 10.141.208.237
  0100 .... = Version: 4
  .... 0101 = Header Length: 20 bytes (5)
▶ Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
  Total Length: 1500
  Identification: 0xffff3 (65523)
▼ Flags: 0x01 (More Fragments)
  0... .... = Reserved bit: Not set
  .0.. .... = Don't fragment: Not set
  ..1. .... = More fragments: Set
  Fragment offset: 1480
  Time to live: 63
  Protocol: UDP (17)
  Header checksum: 0xf137 [validation disabled]
  [Header checksum status: Unverified]
  Source: 10.221.125.213
  Destination: 10.141.208.237
  [Source GeoIP: Unknown]
  [Destination GeoIP: Unknown]
Reassembled IPv4 in frame: 10

```

比如这个 IPv4 的报文，片偏移量为 1480，表示这个报文的数据起始字节在原报文中的序号是 1480，即第 1480 个。根据标识可以确定属于同一原分组的分组，再根据片偏移量和数据长度，就可以将分片的报文重新进行组装。

IPv4 协议的标志字段 (flags) 有 3 bit，第一位预留不使用，置为 0；第二位是 DF (Don't Fragment) 位，设为 1 时表明路由器不能对上层数据包分段；第三位是 MF (More Fragments) 位，如果产生分片，则除最后一个分片分组的 MF 位为 0 外，其余分片分组的 MF 位为 1。MF 位设为 1 表示还有分片。

```

▼ Internet Protocol Version 4, Src: 10.221.125.213, Dst: 10.141.208.237
  0100 .... = Version: 4
  .... 0101 = Header Length: 20 bytes (5)
▶ Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
  Total Length: 1500
  Identification: 0x8b93 (35731)
▼ Flags: 0x01 (More Fragments)
  0... .... = Reserved bit: Not set
  .0.. .... = Don't fragment: Not set
  ..1. .... = More fragments: Set
  Fragment offset: 1480
  Time to live: 63
  Protocol: UDP (17)
  Header checksum: 0x6598 [validation disabled]
  [Header checksum status: Unverified]
  Source: 10.221.125.213
  Destination: 10.141.208.237
  [Source GeoIP: Unknown]
  [Destination GeoIP: Unknown]
Reassembled IPv4 in frame: 22

```

可以看见，这个分组中，DF 位没有被设置，MF 位被设为 1，表示已经分片，后面还有分组。

如果产生分片，则除最后一个分片分组的 MF 位为 0 外，其余分片分组的 MF 位为 1。前面的例子我们可以看见，MF 位为 1，表明后面还有分组。而最后一个分片分组，MF 位为 0，表明没有分组了：

```

▼ Internet Protocol Version 4, Src: 10.221.125.213, Dst: 10.141.208.237
  0100 .... = Version: 4
  .... 0101 = Header Length: 20 bytes (5)
  ► Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 588
    Identification: 0xffff3 (65523)
  ▼ Flags: 0x00
    0... .... = Reserved bit: Not set
    .0.. .... = Don't fragment: Not set
    ..0. .... = More fragments: Not set
    Fragment offset: 4440
    Time to live: 63
    Protocol: UDP (17)
    Header checksum: 0x1356 [validation disabled]
    [Header checksum status: Unverified]
    Source: 10.221.125.213
    Destination: 10.141.208.237
    [Source GeoIP: Unknown]
    [Destination GeoIP: Unknown]
  ► [4 IPv4 Fragments (5008 bytes): #7(1480), #8(1480), #9(1480), #10(568)]

```

1.2 为了能让接收端重组原始数据报，IP 使用首部的特殊字段对分片进行了编号。观察 IP 分组中标识字段，观察同一次发送的 IP 分组该字段是否相同。第二次发送的 IP 分组该字段是否与第一次相同。

IP 协议使用 16 bit 的标识 (identifier) 来确定属于同一个原分组的分片分组，属于同一个原分组的分片分组的标识号相同。再加上片内偏移和分组长度，就可以将分片分组的数据进行拼接。

IP 分组中的标识字段位置如下：

```

▼ Internet Protocol Version 4, Src: 10.221.125.213, Dst: 10.141.208.237
  0100 .... = Version: 4
  .... 0101 = Header Length: 20 bytes (5)
  ► Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 1500
    Identification: 0xffff3 (65523)
  ▼ Flags: 0x01 (More Fragments)
    0.... .... = Reserved bit: Not set
    .0.. .... = Don't fragment: Not set
    ..1. .... = More fragments: Set
    Fragment offset: 1480
    Time to live: 63
    Protocol: UDP (17)
    Header checksum: 0xf137 [validation disabled]
    [Header checksum status: Unverified]
    Source: 10.221.125.213
    Destination: 10.141.208.237
    [Source GeoIP: Unknown]
    [Destination GeoIP: Unknown]
    Reassembled IPv4 in frame: 10

```

同一次发送的 IP 分组的标识是相同的，表明属于同一个原分组：

3	0.465118	10.221.125.213	10.141.208.237	IPv4	1514	Fragmented IP protocol (proto=UDP 17, off=0, ID=51e6) [Reassembled in #6]
4	0.465119	10.221.125.213	10.141.208.237	IPv4	1514	Fragmented IP protocol (proto=UDP 17, off=1480, ID=51e6) [Reassembled in #6]
5	0.465119	10.221.125.213	10.141.208.237	IPv4	1514	Fragmented IP protocol (proto=UDP 17, off=2960, ID=51e6) [Reassembled in #6]
6	0.465120	10.221.125.213	10.141.208.237	UDP	602	59656 → 5001 Len=5000

第二次发送的 IP 分组的标识字段与第一次不同，需要加以区分：

3	0.465118	10.221.125.213	10.141.208.237	IPv4	1514	Fragmented IP protocol (proto=UDP 17, off=0, ID=51e6) [Reassembled in #6]
4	0.465119	10.221.125.213	10.141.208.237	IPv4	1514	Fragmented IP protocol (proto=UDP 17, off=1480, ID=51e6) [Reassembled in #6]
5	0.465119	10.221.125.213	10.141.208.237	IPv4	1514	Fragmented IP protocol (proto=UDP 17, off=2960, ID=51e6) [Reassembled in #6]
6	0.465120	10.221.125.213	10.141.208.237	UDP	602	59656 → 5001 Len=5000
7	0.465120	10.221.125.213	10.141.208.237	IPv4	1514	Fragmented IP protocol (proto=UDP 17, off=0, ID=ffff3) [Reassembled in #10]
8	0.465120	10.221.125.213	10.141.208.237	IPv4	1514	Fragmented IP protocol (proto=UDP 17, off=1480, ID=ffff3) [Reassembled in #10]
9	0.465121	10.221.125.213	10.141.208.237	IPv4	1514	Fragmented IP protocol (proto=UDP 17, off=2960, ID=ffff3) [Reassembled in #10]
10	0.465121	10.221.125.213	10.141.208.237	UDP	602	59656 → 5001 Len=5000

1.3 分析每个5008个字节的UDP数据报（每个数据报包含5000个字节的数据部分和8各字节的UDP首部），被分为了几个分段，为什么？（提示：以太网要求一次传输的长度不大于1514个字节，其中14字节时以太网帧的首部）

被分为了 4 段。

从原理上说，以太网的 MTU 为 1500 个字节，而此次发送的数据为 5000 个字节，相除后向上取整，是 4。

从实验结果来说，每次传输的分片分组，前三个分组的片内偏移分别为 0、1480、2960，长度均为 1480；最后一个分组的片内偏移为 4440，长度为 568（去掉首部的 20 个字节）：

```
▼ Internet Protocol Version 4, Src: 10.221.125.213, Dst: 10.141.208.237
  0100 .... = Version: 4
  .... 0101 = Header Length: 20 bytes (5)
▶ Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
  Total Length: 588
  Identification: 0xffff3 (65523)
  Flags: 0x00
    0.... .... = Reserved bit: Not set
    .0... .... = Don't fragment: Not set
    ..0. .... = More fragments: Not set
  Fragment_offset: 4440
  Time to live: 63
  Protocol: UDP (17)
  Header checksum: 0x1356 [validation disabled]
  [Header checksum status: Unverified]
  Source: 10.221.125.213
  Destination: 10.141.208.237
  [Source GeoIP: Unknown]
  [Destination GeoIP: Unknown]
▶ [4 IPv4 Fragments (5008 bytes): #7(1480), #8(1480), #9(1480), #10(568)]
```

四者加起来后，正好是 5008 个字节。

所以被分为了 4 段。

## 2. ping

实验步骤：

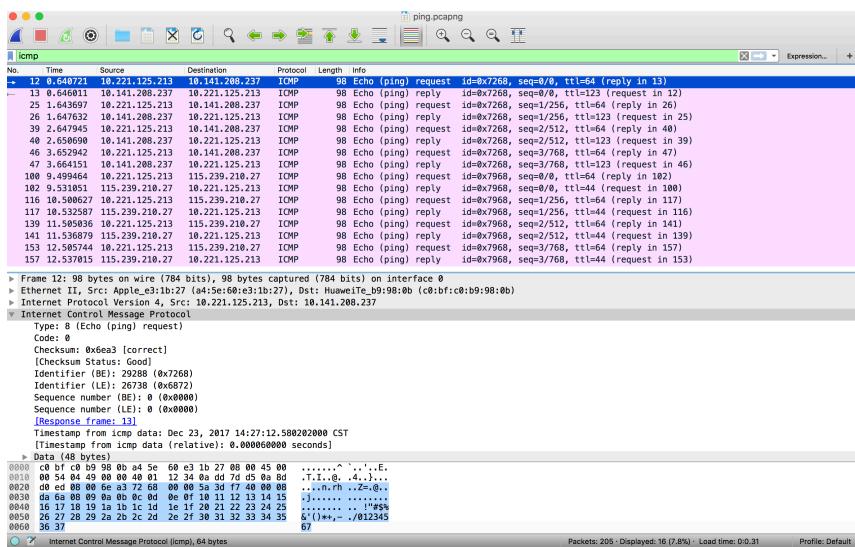
ping的运作原理是向目标主机传出一个ICMP echo要求数据包，并等待接收echo回应数据包。程序会按时间和成功响应的次数估算丢失数据包率（丢包率）和数据包往返时间（网络时延，Round-trip delay time）。使用Wireshark捕获包进行分析。

主机IP地址为10.221.125.213，本地服务器IP地址为10.141.208.237。

实验分析：

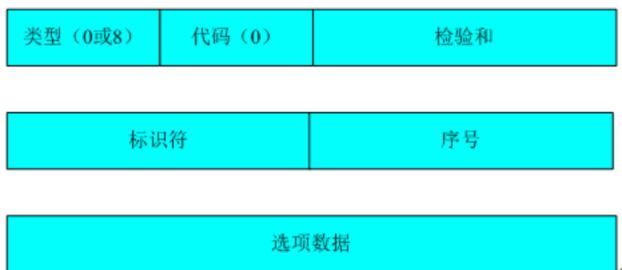
**2.1 观察 ping 命令捕获的分组，对于每个目的端，发送 4 个回应请求分组并接受 4 个 ICMP 回应应答分组，分析 ICMP 请求分组和 ICMP 应答分组字段的含义，写出回应请求和回应应答的 ICMP 类型号分别是什么。**

分别 ping 了 10.141.208.237 和 www.baidu.com，每个 IP 四次，结果如下：



前八个分组为对 10.141.208.237 的请求和响应，后八个分组为对 www.baidu.com 的请求和响应。

ping 的 ICMP 的报文格式如下：



首先查看 ICMP 的请求分组：

```

▶ Internet Protocol Version 4, Src: 10.221.125.213, Dst: 10.141.208.237
▼ Internet Control Message Protocol
  Type: 8 (Echo (ping) request)
  Code: 0
  Checksum: 0x6ea3 [correct]
  [Checksum Status: Good]
  Identifier (BE): 29288 (0x7268)
  Identifier (LE): 26738 (0x6872)
  Sequence number (BE): 0 (0x0000)
  Sequence number (LE): 0 (0x0000)
  [Response frame: 13]
  Timestamp from icmp data: Dec 23, 2017 14:27:12.580202000 CST
  [Timestamp from icmp data (relative): 0.000060000 seconds]
▶ Data (48 bytes)

```

首先，是 8 bit 的类型，这里值为 8，表示 ping 的请求（0 表示 ping 的应答）；然后，是 8 bit 的代码（code），不同的值指明发送失败的不同原因，这里发送成功，没有设置：

```

Code
0 = net unreachable;
1 = host unreachable;
2 = protocol unreachable;
3 = port unreachable;
4 = fragmentation needed and DF set;
5 = source route failed.

```

再然后，是 16 bit 的校验和；校验和之后，是 16 bit 的标识符和 16 bit 的序号。这里特别注意一下，Wireshark 将标识符和序号分别表示为 BE 和 LE，然而捕获的包中标识符和序号只有一个。出现这种情况的原因是不同操作系统的字节顺序不一样，有**大端** (Big Endian) 和**小端** (Little Endian) 之分，这里 Wireshark 将它们全部标识了出来以作区分。再之后就是 ICMP 的数据部分。从 Wireshark 的捕获结果来看，ICMP 的数据中包含了时间戳。

对应这个 ping 请求的响应分组内容如下：

```

▶ Internet Protocol Version 4, Src: 10.141.208.237, Dst: 10.221.125.213
▼ Internet Control Message Protocol
  Type: 0 (Echo (ping) reply)
  Code: 0
  Checksum: 0x76a3 [correct]
  [Checksum Status: Good]
  Identifier (BE): 29288 (0x7268)
  Identifier (LE): 26738 (0x6872)
  Sequence number (BE): 0 (0x0000)
  Sequence number (LE): 0 (0x0000)
  [Request frame: 12]
  [Response time: 5.290 ms]
  Timestamp from icmp data: Dec 23, 2017 14:27:12.580202000 CST
  [Timestamp from icmp data (relative): 0.005350000 seconds]
▶ Data (48 bytes)

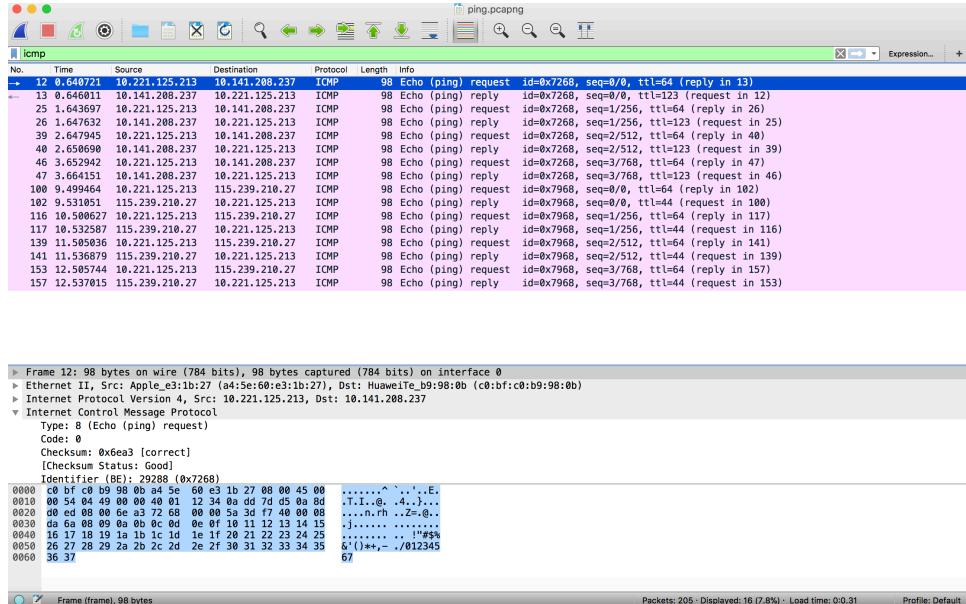
```

可以看见，响应分组的报文格式与请求分组几乎一模一样，只不过开始的 8 bit 的类型值为 0，表示是应答分组。

请求的 ICMP 类型号为 8，应答的 ICMP 类型号为 0。

## 2.2 在显示过滤器中输入“icmp”，过滤出 ICMP 分组。

过滤出的 ICMP 分组内容如下：



## 2.3 比较本地 ping 命令和远程 ping 命令的 ICMP 请求分组、ICMP 响应分组中“生存时间”（Timetolive）字段的区别，思考生存时间的含义。

对于 ping 10.141.208.237（连在复旦校园网内的服务器）：

ICMP 请求分组的 TTL 字段：

```
▼ Internet Protocol Version 4, Src: 10.221.125.213, Dst: 10.141.208.237
  0100 .... = Version: 4
  .... 0101 = Header Length: 20 bytes (5)
  ▶ Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
  Total Length: 84
  Identification: 0x0449 (1097)
  Flags: 0x00
  Fragment offset: 0
  Time to live: 64
  Protocol: ICMP (1)
```

ICMP 响应分组的 TTL 字段：

```
▼ Internet Protocol Version 4, Src: 10.141.208.237, Dst: 10.221.125.213
  0100 .... = Version: 4
  .... 0101 = Header Length: 20 bytes (5)
  ▶ Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
  Total Length: 84
  Identification: 0x1809 (6153)
  Flags: 0x00
  Fragment offset: 0
  Time to live: 123
  Protocol: ICMP (1)
```

对于 ping www.baidu.com:

ICMP 请求分组的 TTL 字段:

```
▼ Internet Protocol Version 4, Src: 10.221.125.213, Dst: 115.239.210.27
  0100 .... = Version: 4
  .... 0101 = Header Length: 20 bytes (5)
  ► Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 84
    Identification: 0xa87 (6791)
  ► Flags: 0x00
    Fragment offset: 0
    Time to live: 64
  Protocol: ICMP (1)
```

ICMP 响应分组的 TTL 字段:

```
▼ Internet Protocol Version 4, Src: 115.239.210.27, Dst: 10.221.125.213
  0100 .... = Version: 4
  .... 0101 = Header Length: 20 bytes (5)
  ► Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 84
    Identification: 0xa87 (6791)
  ► Flags: 0x00
    Fragment offset: 0
    Time to live: 44
  Protocol: ICMP (1)
```

可以看见，本地 ping 指令和远程 ping 指令的请求分组的 TTL 均是 64，但是本地 ping 指令的响应分组的 TTL 是 123，而远程 ping 指令的响应分组的 TTL 却只有 44。

出现这种现象的原因是因为 TTL 表示分组在网络中剩余的生存时间，防止分组如同幽灵般永远在网络内游荡。TTL 字段具体的含义是可以经过的路由器的数量，每经过一个路由器，TTL 的值减去一；当 TTL 为 0 时，路由器必须丢弃这个分组。

所以，发出 ping 指令时，还没有经过路由器，所以两者的请求分组的 TTL 是相同的；而收到响应分组时，来自本地服务器的分组经过的路由器少，TTL 的值就大，而来自远程服务器的分组经过的路由器多，TTL 字段的值就小。

### 3. traceroute

实验步骤:

由于使用的是 Mac，因此使用 traceroute 尝试连接本地服务器和远程服务器（www.baidu.com），使用 Wireshark 抓包并分析。

主机 IP 地址为 10.221.125.213，本地服务器 IP 地址为 10.141.208.237。

实验分析:

traceroute 命令用 IP 协议中的生存时间（TTL）字段和 ICMP 错误消息来确定从一个主机到网络上其他主机的路由。每次发送三个 TTL 相同的分组测量 RTT，TTL 不断增加直

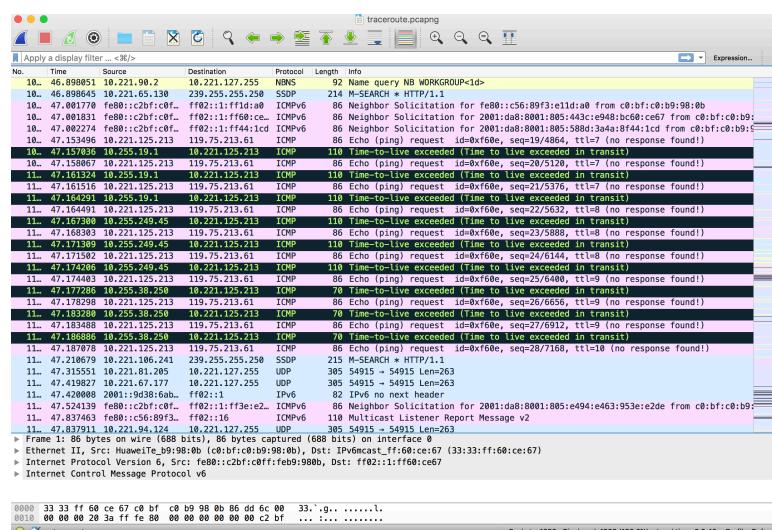
至到达目的 IP。首先，traceroute 送出 TTL 是 1 的 IP 数据包到目的地，当路径上的第一个路由器收到这个数据包时，它将 TTL 减一。此时，TTL 变为 0，所以该路由器会将此数据包丢掉，并送回一个 ICMP time exceeded 消息（包括发 IP 包的源地址，IP 包的所有内容及路由器的 IP 地址），traceroute 收到这个消息后，便知道这个路由器存在于这个路径上，接着 traceroute 再送出另一个 TTL 是 2 的数据包，发现第二个路由器，这样一直进行下去。traceroute 每次将送出的数据包的 TTL 加 1 来发现另一个路由器，这个重复的动作一直持续到某个数据包抵达目的地。当数据包到达目的地后，该主机则不会送回 ICMP time exceeded 消息，由于 traceroute 通过 UDP 数据包向不常见端口(30000 以上)发送数据包，因此会收到 ICMP port unreachable 消息，故可判断到达目的地。

traceroute 有一个固定的时间等待响应(ICMP TTL 到期消息)。如果这个时间过了，它将打印出一系列的\*号表明：在这个路径上，这个设备不能在给定的时间内发出 ICMP TTL 到期消息的响应。然后，traceroute 给 TTL 记数器加 1，继续进行。

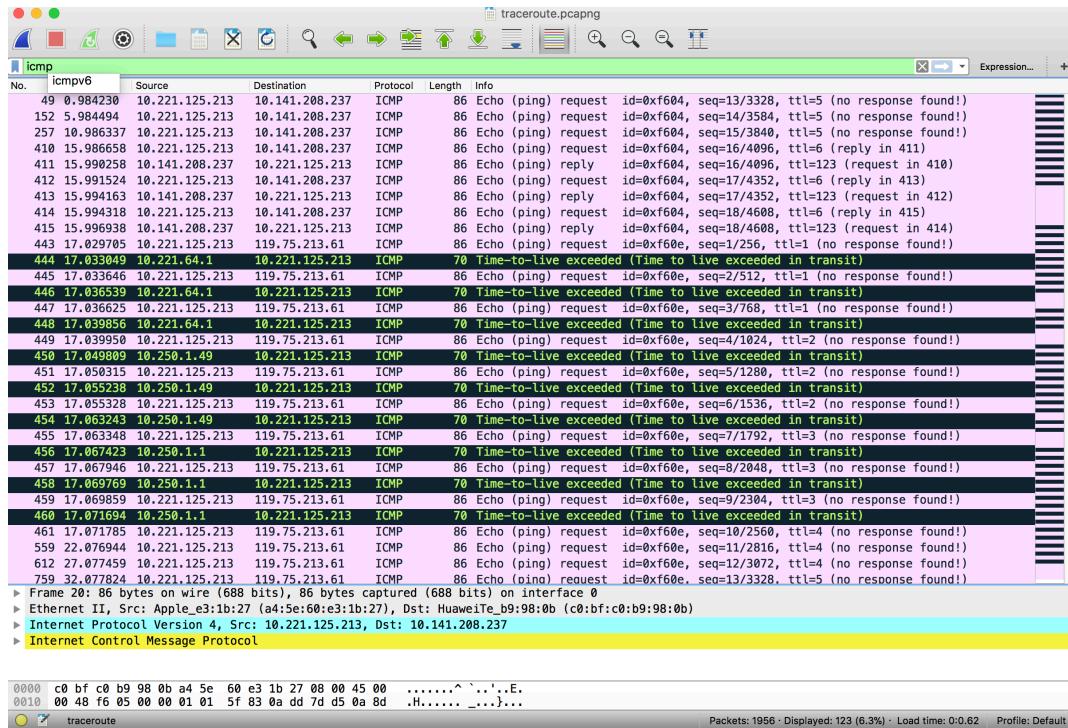
特别注意，默认情况下，traceroute 是向目的地址的某个端口（大于 30000）发送 UDP 数据报，tracert 是向目的地址发出 ICMP 请求回显数据包。为了便于展示结果，使用 traceroute 指令时加上-I 参数，使得 traceroute 使用 ICMP 回应取代 UDP 资料信息，结果与 tracert 类似。

### 3.1 观察 traceroute 命令捕获的分组，在显示过滤器中输入“icmp”，过滤出 ICMP 分组。

捕获的分组如图：



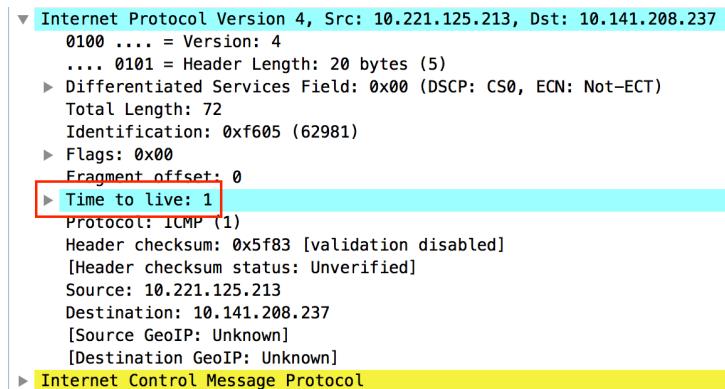
使用过滤器后的 ICMP 分组:



可以看出，每次 ICMP 请求以 ping 的 request 的方式发出，然后收到 ICMP 的超时分组，从而获得路由器信息。

### 3.2 观察 traceroute 命令（本地/远程）发送的第一个探测分组，IP 首部中生存时间字段被置为 1。

发送到本地服务器的第一个探测分组:



事实上每次探测有三个 TTL 相同的分组:

20 0.913223	10.221.125.213	10.141.208.237	ICMP	86 Echo (ping) request id=0xf604, seq=1/256, ttl=1 (no response found!)
21 0.931940	10.221.64.1	10.221.125.213	ICMP	70 Time-to-live exceeded (Time to live exceeded in transit)
22 0.932987	10.221.125.213	10.141.208.237	ICMP	86 Echo (ping) request id=0xf604, seq=2/512, ttl=1 (no response found!)
28 0.941160	10.221.64.1	10.221.125.213	ICMP	70 Time-to-live exceeded (Time to live exceeded in transit)
29 0.941285	10.221.125.213	10.141.208.237	ICMP	86 Echo (ping) request id=0xf604, seq=3/768, ttl=1 (no response found!)
30 0.944878	10.221.64.1	10.221.125.213	ICMP	70 Time-to-live exceeded (Time to live exceeded in transit)

发送到远程服务器的第一个探测分组:

```
▼ Internet Protocol Version 4, Src: 10.221.125.213, Dst: 119.75.213.61
  0100 .... = Version: 4
  .... 0101 = Header Length: 20 bytes (5)
  ▶ Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 72
    Identification: 0xf60f (62991)
  ▶ Flags: 0x00
    Fragment offset: 0
  ▶ Time to live: 1
  Protocol: ICMP (1)
  Header checksum: 0xee6a [validation disabled]
  [Header checksum status: Unverified]
  Source: 10.221.125.213
  Destination: 119.75.213.61
  [Source GeoIP: Unknown]
  [Destination GeoIP: Unknown]
  ▶ Internet Control Message Protocol
```

同样的，每次探测有三个 TTL 相同的分组:

443	17.029705	10.221.125.213	119.75.213.61	ICMP	86 Echo (ping) request id=0xf60e, seq=1/256, ttl=1 (no response found!)
444	17.033049	10.221.64.1	10.221.125.213	ICMP	70 Time-to-live exceeded (Time to live exceeded in transit)
445	17.033646	10.221.125.213	119.75.213.61	ICMP	86 Echo (ping) request id=0xf60e, seq=2/512, ttl=1 (no response found!)
446	17.036539	10.221.64.1	10.221.125.213	ICMP	70 Time-to-live exceeded (Time to live exceeded in transit)
447	17.036625	10.221.125.213	119.75.213.61	ICMP	86 Echo (ping) request id=0xf60e, seq=3/768, ttl=1 (no response found!)
448	17.039856	10.221.64.1	10.221.125.213	ICMP	70 Time-to-live exceeded (Time to live exceeded in transit)

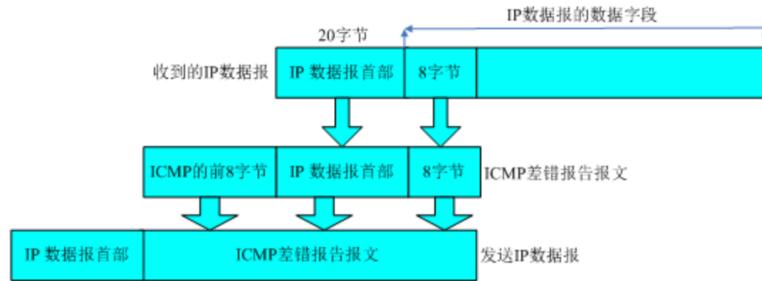
我们可以看见，不论是到本地服务器还是到远程服务器，首次探测的三个 ICMP 分组的 TTL 均是 1。这是 traceroute 的工作原理决定的。TTL 为 1，到达第一个路由器分组就会超时，路由器丢弃分组的同时发送 ICMP 超时报文，主机就可以根据收到的超时报文获得第一个路由器的信息。以此类推，不断递增 TTL 的值，就可以逐个获取路径上的路由器的信息，直至到达目的 IP 地址。

3.3 观察远程 traceroute 命令中，ICMP 生存超时报文分组，写出该分组中 ICMP 类型号是什么。

任意选择一个远程 traceroute 命令的 ICMP 生存超时报文分组:

```
▼ Internet Protocol Version 4, Src: 10.255.38.250, Dst: 10.221.125.213
  0100 .... = Version: 4
  .... 0101 = Header Length: 20 bytes (5)
  ▶ Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 56
    Identification: 0x0000 (0)
  ▶ Flags: 0x00
    Fragment offset: 0
    Time to live: 247
    Protocol: ICMP (1)
    Header checksum: 0x091a [validation disabled]
    [Header checksum status: Unverified]
    Source: 10.255.38.250
    Destination: 10.221.125.213
    [Source GeoIP: Unknown]
    [Destination GeoIP: Unknown]
    ▶ Internet Control Message Protocol
      Type: 11 (Time-to-live exceeded)
      Code: 0 (Time to live exceeded in transit)
      Checksum: 0xf4f1 [correct]
      [Checksum Status: Good]
  ▶ Internet Protocol Version 4, Src: 10.221.125.213, Dst: 119.75.213.61
  0100 .... = Version: 4
  .... 0101 = Header Length: 20 bytes (5)
  ▶ Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 72
    Identification: 0xf629 (63017)
  ▶ Flags: 0x00
    Fragment offset: 0
    Time to live: 1
    Protocol: ICMP (1)
    Header checksum: 0xee50 [validation disabled]
    [Header checksum status: Unverified]
    Source: 10.221.125.213
    Destination: 119.75.213.61
    [Source GeoIP: Unknown]
    [Destination GeoIP: Unknown]
  ▶ Internet Control Message Protocol
```

ICMP 超时报文的格式与上面相同，但包含数据不同：



在 ICMP 超时报文中，类型码的数值为 11，code 为 0，而数据部分，则是收到的 IP 报文的首部加上 IP 报文数据中的前 8 个字节。从 Wireshark 捕获的分组我们可以看出，ICMP 超时报文的格式和内容与之相符，包含 IP 报文的首部和一部分数据。

类型号为 11，表示超时。

### 3.4 观察分析 ICMP 请求分组中生存时间字段是如何变化的，ICMP 生存超时报文分组中生存时间字段的是如何变化。

ICMP 请求分组中 TTL 字段从 1 开始不断递增（每三个一组），ICMP 超时报文分组中 TTL 字段从一个较大的值（这里是 255）开始逐渐递减（也是每三个一组）。

ICMP 请求分组（每三个一组的 TTL 相同，这里不全放上来了）：

```

▼ Internet Protocol Version 4, Src: 10.221.125.213, Dst: 10.141.208.237
  0100 .... = Version: 4
  .... 0101 = Header Length: 20 bytes (5)
  ▶ Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
  Total Length: 72
  Identification: 0xf605 (62981)
  ▶ Flags: 0x00
  Fragment offset: 0
  ▶ Time to live: 1
  Protocol: ICMP (1)

▼ Internet Protocol Version 4, Src: 10.221.125.213, Dst: 10.141.208.237
  0100 .... = Version: 4
  .... 0101 = Header Length: 20 bytes (5)
  ▶ Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
  Total Length: 72
  Identification: 0xf608 (62984)
  ▶ Flags: 0x00
  Fragment offset: 0
  ▶ Time to live: 2
  Protocol: ICMP (1)

▼ Internet Protocol Version 4, Src: 10.221.125.213, Dst: 10.141.208.237
  0100 .... = Version: 4
  .... 0101 = Header Length: 20 bytes (5)
  ▶ Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
  Total Length: 72
  Identification: 0xf60b (62987)
  ▶ Flags: 0x00
  Fragment offset: 0
  ▶ Time to live: 3
  Protocol: ICMP (1)
  
```

ICMP 超时分组（仍然是每三个一组 TTL 相同）：

```

▼ Internet Protocol Version 4, Src: 10.221.64.1, Dst: 10.221.125.213
  0100 .... = Version: 4
  .... 0101 = Header Length: 20 bytes (5)
  ▶ Differentiated Services Field: 0xc0 (DSCP: CS6, ECN: Not-ECT)
  Total Length: 56
  Identification: 0x76c1 (30401)
  ▶ Flags: 0x00
  Fragment offset: 0
  Time to live: 255
  Protocol: ICMP (1)

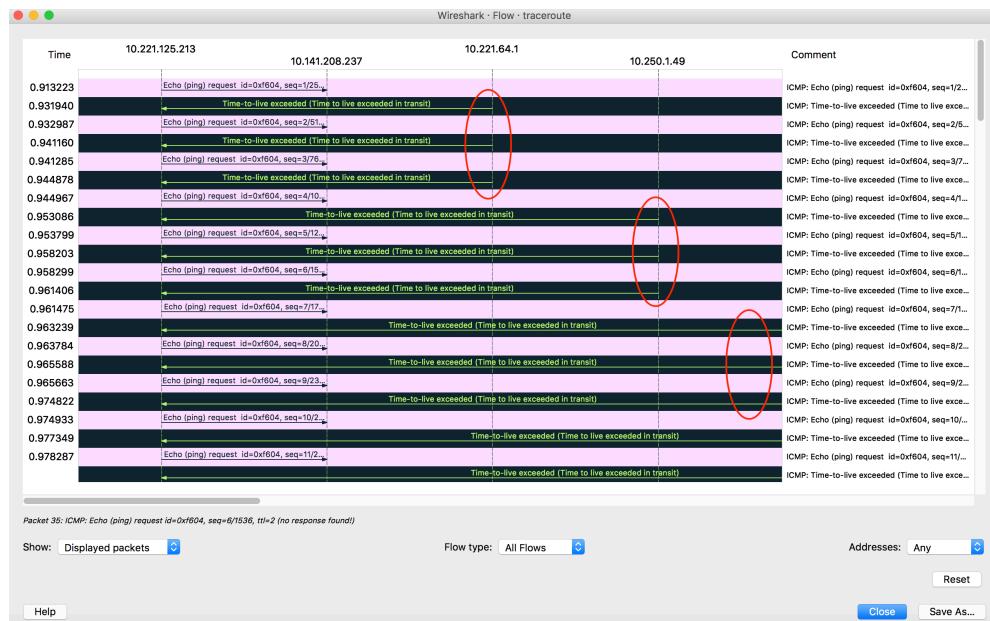
▼ Internet Protocol Version 4, Src: 10.250.1.49, Dst: 10.221.125.213
  0100 .... = Version: 4
  .... 0101 = Header Length: 20 bytes (5)
  ▶ Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
  Total Length: 56
  Identification: 0x02c6 (710)
  ▶ Flags: 0x00
  Fragment offset: 0
  Time to live: 254
  Protocol: ICMP (1)

▼ Internet Protocol Version 4, Src: 10.250.1.1, Dst: 10.221.125.213
  0100 .... = Version: 4
  .... 0101 = Header Length: 20 bytes (5)
  ▶ Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
  Total Length: 56
  Identification: 0x0000 (0)
  ▶ Flags: 0x00
  Fragment offset: 0
  Time to live: 253
  Protocol: ICMP (1)

```

出现这种情况的原因仍与 traceroute 的工作原理有关。随着 ICMP 请求指令中 TTL 的不断递增，可以逐个的获取路径上路由器的信息；每个路由器发送的超时 ICMP 报文的初始 TTL 都相同，为 255，随着经过的路由器的增多递减。

3.5 选择一个 ICMP 分组，使用“Statistics/FlowGraph”来观察，将该图截下来进行分析，并解释结果。



从图中可以看出，每三个超时的 ICMP 分组来自于同一个 IP 地址。与前面一样，这是 traceroute 的工作原理造成的。traceroute 每次发送三个 TTL 相同的 ICMP 请求分组，然后递增 TTL，到达某个路由器后 TTL 减为 0，路由器便将这三个分组丢掉，同时发送超时错误的 ICMP 响应分组，主机收到这三个分组后，就可以获得路径上路由器的信息，同时计算 RTT 时间。这就是上图产生的原因。

#### 4. 讨论与研究

##### 4.1 研究无类域间路由（Classless Inter Domain Routing, CIDR）和网络地址转换（Network Address Translation, NAT）在解决 IPv4 地址空间枯竭问题中所起的作用。

传统的对 IP 地址的划分（如 A 类、B 类、C 类）过于绝对，会造成 IP 地址的浪费。比如某个公司需要 50,000 个 IP 地址，如果使用多个 C 类地址的话就会不方便管理，如果使用一个 B 类地址的话就会有 15,000 个左右的 IP 地址被浪费。这无疑会加重 IP 地址的枯竭。

CIDR 对原来用于分配 A 类、B 类和 C 类地址的有类别路由选择进程进行了重新构建。CIDR 用 13–27 位长的前缀取代了原来地址结构对地址网络部分的限制（3 类地址的网络部分分别被限制为 8 位、16 位和 24 位）。在管理员能分配的地址块中，主机数量范围是 32–500,000，从而能更好地满足机构对地址的特殊需求，数量配置也更加灵活，降低对 IP 地址的浪费程度。同时，使用这种技术还可以减缓路由表的增长。

NAT 可以实现多个内部节点共享一个外部地址，内部可以有很多主机，每个主机有不同的内部地址，然而他们的外部地址都相同（比如复旦的校园网，外部地址只有几个，内部却可以支持成千上万的主机、服务器正常联网）。因此，NAT 可以极大的缓解 IP 地址枯竭带来的问题。事实上，IPv4 的地址早已经在 2011 年分配完毕，而 NAT 是使得 IPv4 能够使用至今的重要原因之一。

4.2 从 <http://www.traceroute.org> 上选择一个 traceroute 服务器，执行一次到自己本地机的 traceroute，再对那台机器作一次相反的 traceroute。记录两次操作的 traceroute 输出，评价两次路径的相似性。

使用的 traceroute 服务器为台湾的 HiNet (<http://traceroute.hinet.net/>)。尝试连接复旦的一个 IP 地址 202.120.224.92（主机所在的内网）。

使用台湾的 traceroute 服务器的原因是北京和上海的服务器无法正常使用。而且，在使用过程中，发现 traceroute 服务器只显示前 30 个结果。

连接结果如下：

### Traceroute Result (202.120.224.92) :

```
Type escape sequence to abort.
Tracing the route to 224.fudan.edu.cn (202.120.224.92)

 1 TPDB-3516.hinet.net (210.65.161.22) 0 msec 4 msec 0 msec
 2 TPDT-3011.hinet.net (220.128.1.146) 4 msec 4 msec 0 msec
 3 r404-s2.tp.hinet.net (220.128.1.182) 4 msec 0 msec 0 msec
 4 r12-hk.hinet.net (220.128.2.177) 20 msec 24 msec 20 msec
 5 r11-hk.hinet.net (220.128.3.153) 24 msec 24 msec 20 msec
 6 pacnet-hk-gw.hinet.net (211.22.33.53) 24 msec 16 msec 16 msec
 7 xe-0-0-0.0.gw2.hkg9.10026.telstraglobal.net (61.14.158.67) 16 msec 20 msec 28 msec
 8 61.8.59.38 152 msec 152 msec 152 msec
 9 101.4.117.149 192 msec 192 msec 192 msec
10 101.4.118.121 192 msec 192 msec 192 msec
11 101.4.115.9 188 msec 192 msec 192 msec
12 101.4.112.70 220 msec 220 msec 216 msec
13 101.4.116.117 216 msec 216 msec 220 msec
14 *
   101.4.117.29 216 msec *
15 101.4.115.173 216 msec * *
16 * * *
17 * * *
18 * * *
19 * * *
20 * * *
21 * * *
22 * * *
23 * * *
24 * * *
25 * * *
26 * * *
27 * * *
28 * * *
29 * * *
30 * * *
```

使用主机 traceroute 服务器的结果如下：

```
traceroute to traceroute.hinet.net (203.69.42.196), 64 hops max, 52 byte packets
 1 10.221.64.1 (10.221.64.1) 3.368 ms 3.327 ms 6.361 ms
 2 10.250.1.49 (10.250.1.49) 12.495 ms 7.531 ms *
 3 10.250.1.69 (10.250.1.69) 2.453 ms
 10.250.1.1 (10.250.1.1) 3.322 ms 3.813 ms
 4 * * *
 5 224.fudan.edu.cn (202.120.224.3) 19.461 ms 4.212 ms 4.952 ms
 6 * * *
 7 10.255.19.1 (10.255.19.1) 4.387 ms 4.662 ms 8.826 ms
 8 10.255.249.45 (10.255.249.45) 4.834 ms 3.404 ms 4.061 ms
 9 10.255.38.250 (10.255.38.250) 2.962 ms 3.058 ms 3.309 ms
10 * * 202.112.27.1 (202.112.27.1) 5.431 ms
11 * 101.4.115.174 (101.4.115.174) 152.275 ms 521.069 ms
12 101.4.117.30 (101.4.117.30) 524.412 ms 26.143 ms 23.977 ms
13 101.4.116.118 (101.4.116.118) 533.585 ms 265.556 ms 32.495 ms
14 101.4.112.69 (101.4.112.69) 28.871 ms 29.358 ms 29.267 ms
15 101.4.115.10 (101.4.115.10) 32.607 ms
101.4.117.162 (101.4.117.162) 33.681 ms
101.4.117.254 (101.4.117.254) 34.258 ms
16 101.4.118.122 (101.4.118.122) 32.065 ms 31.080 ms 31.619 ms
17 101.4.117.250 (101.4.117.250) 67.228 ms 66.811 ms 66.654 ms
18 61.8.59.37 (61.8.59.37) 70.179 ms 73.078 ms 81.265 ms
19 ae-0.pacnet-telstra.tokyjp03.jp.bb.gin.ntt.net (61.120.145.62) 165.871 ms
160.553 ms 206.624 ms
20 * ae-12.r02.tokyjp03.jp.bb.gin.ntt.net (61.120.145.61) 152.360 ms 143.500 ms
21 ae-10.r31.tokyjp05.jp.bb.gin.ntt.net (129.250.3.252) 153.163 ms 174.203 ms
178.995 ms
22 ae-9.r26.tokyjp05.jp.bb.gin.ntt.net (129.250.2.10) 173.314 ms
ae-10.r26.tokyjp05.jp.bb.gin.ntt.net (129.250.2.152) 146.136 ms
ae-9.r26.tokyjp05.jp.bb.gin.ntt.net (129.250.2.10) 140.910 ms
23 r404-s2.tp.hinet.net (211.72.233.90) 191.181 ms
xe-1.hinet.tokyjp05.jp.bb.gin.ntt.net (129.250.66.114) 221.918 ms
r404-s2.tp.hinet.net (211.72.233.78) 218.432 ms
24 r4104-s2.tp.hinet.net (220.128.7.178) 186.529 ms
r4104-s2.tp.hinet.net (220.128.10.234) 214.712 ms
r4104-s2.tp.hinet.net (220.128.7.178) 177.559 ms
25 tpdt-3011.hinet.net (220.128.3.126) 150.275 ms
tpdt-3011.hinet.net (220.128.3.182) 179.429 ms
tpdt-3011.hinet.net (220.128.7.14) 181.969 ms
26 tchn-3011.hinet.net (220.128.16.9) 199.853 ms 181.403 ms 162.145 ms
27 220-128-16-89.hinet-ip.hinet.net (220.128.16.89) 180.891 ms 180.113 ms 14
8.281 ms
28 211-22-229-25.hinet-ip.hinet.net (211.22.229.25) 140.761 ms 159.129 ms 14
```

可以看出，两者经过的路由器几乎没有重合。这是路由选择算法的原因。考虑到两台主机之间的距离和可选择的路由器，未出现重合并不难理解。

## 5. 参考资料

1. 《数据通信与计算机网络》课程课件
2. CSDN 技术博客