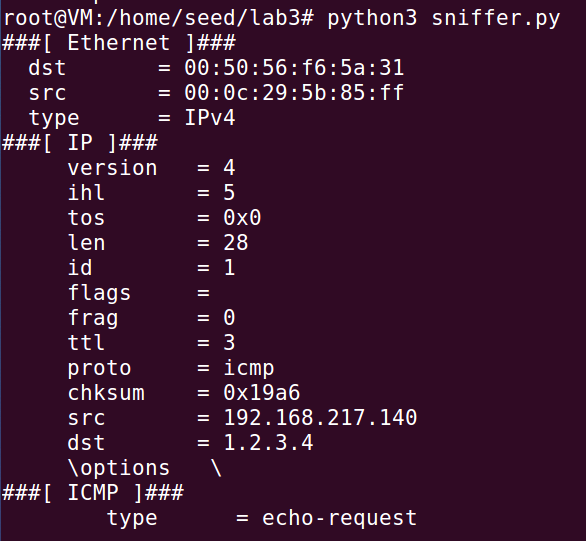
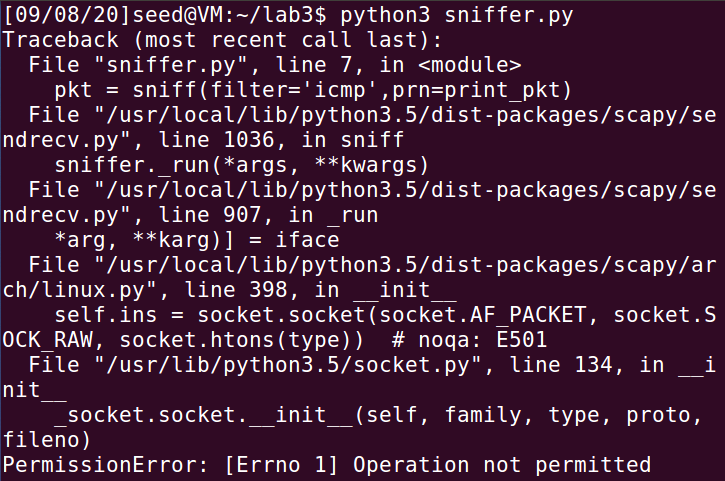
# Packet Sniffing and Spoofing Lab

**Task1.1A**

**在**root和非root情况下分别运行sniffer.py，并尝试输出报文的信息。可见程序必须在root权限下运行，普通用户会被限制。截获的报文为ICMP报文。





**Task1.1B**

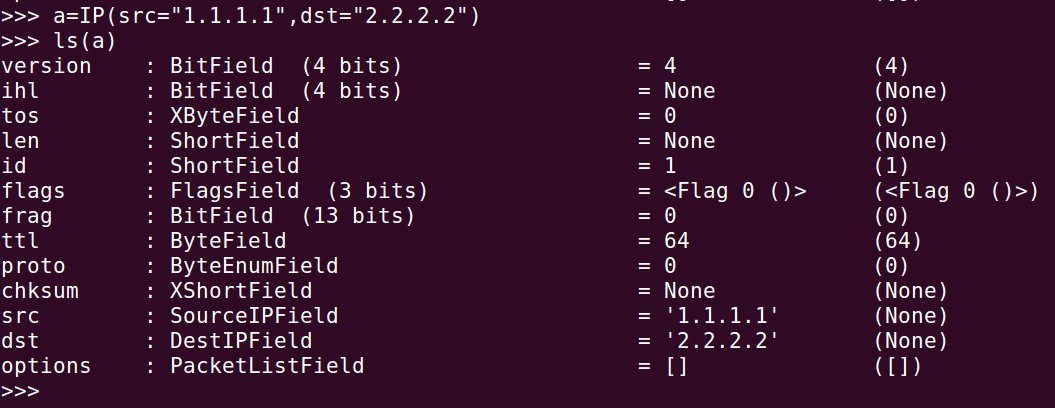
filter='icmp'

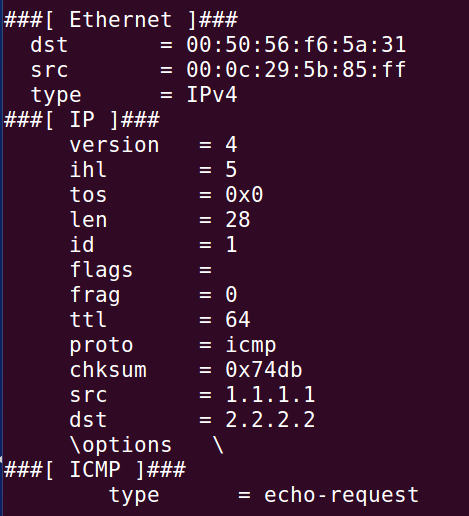
filter='tcp and host IP and port 23 '

filter='net 128.230.0'

**Task 1.2: Spoofing ICMP Packets**

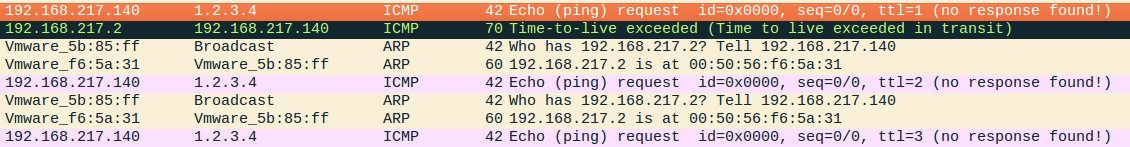
构造封装在IP报文中的ICMP报文，并发送出去。如图使用scapy构造自己定义源地址和目的地址的IP报文a，可以见到截获的报文的源地址显示为定义的“1.1.1.1”，目的地址定义为“2.2.2.2”。



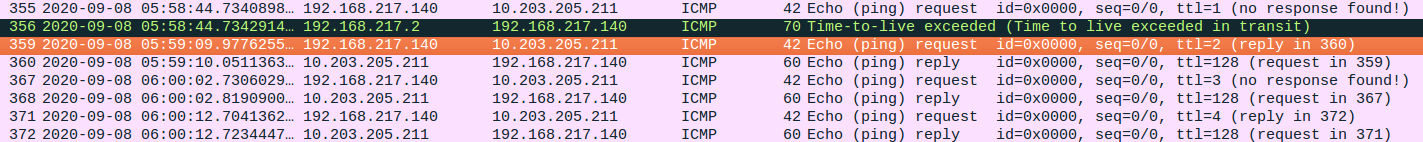


**Task 1.3: Traceroute**

依次设置发送报文的TTL为1、2和3，观察嗅探的结果。当TTL为1时，转发路由器192.168.127.140返回一个ICMP超时报文；而TTL大于1时，没有回应，因为该地址是无效地址。如果我们在目的地址中填入有效地址，则随着TTL的增大，会显示出整个的路径。

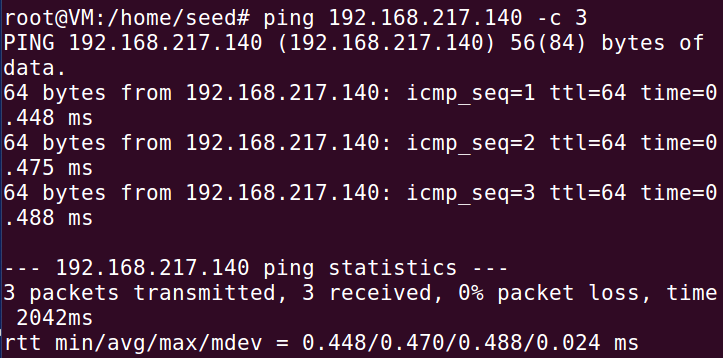


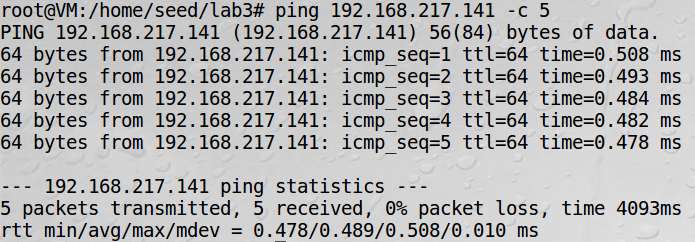
如图所示，尝试向校园网内地址10.203.205.211发送ICMP报文，发现在TTL为2时，就收到了目的地址的回应，说明源地址和目的地址之间只有一台路由器。



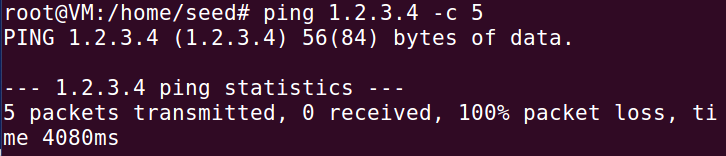
**Task 1.4: Sniffing and-then Spoofing**

设置两台VM虚拟机在同一个LAN中，A(192.168.217.140)与B(192.168.217.141)相互可以ping通。

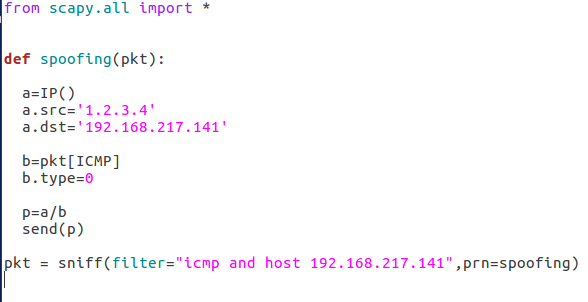




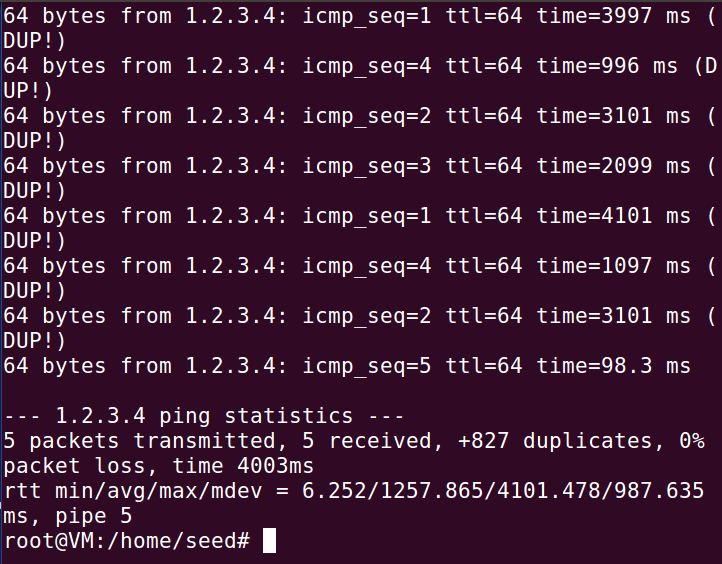
B向一个并不存在的地址1.2.3.4进行ping，结果是不可能收到回应。



在A中编程实现如果B发送ICMP报文，则发送源地址是1.2.3.4的ICMP回复报文。ICMP报文构造的时候，应保持标识符、序号和数据段的一致，这样才能实现回复报文与请求报文的对应。同时，ICMP回复报文的类型值为0，而请求报文的类型值为8，因此构造的ICMP报文的类型值需要修改。

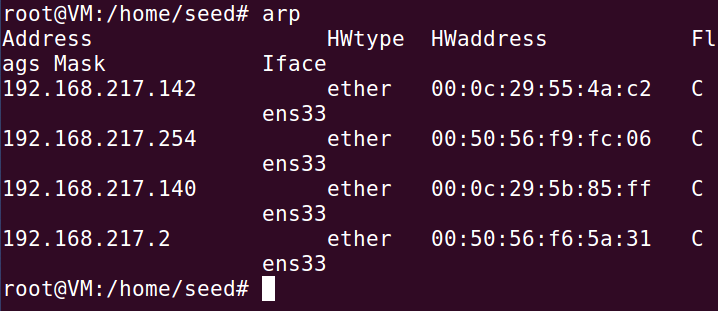


如图，B向1.2.3.4进行ping之后可以收到回复。

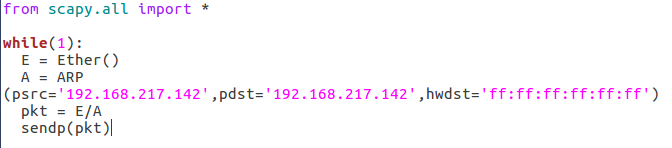


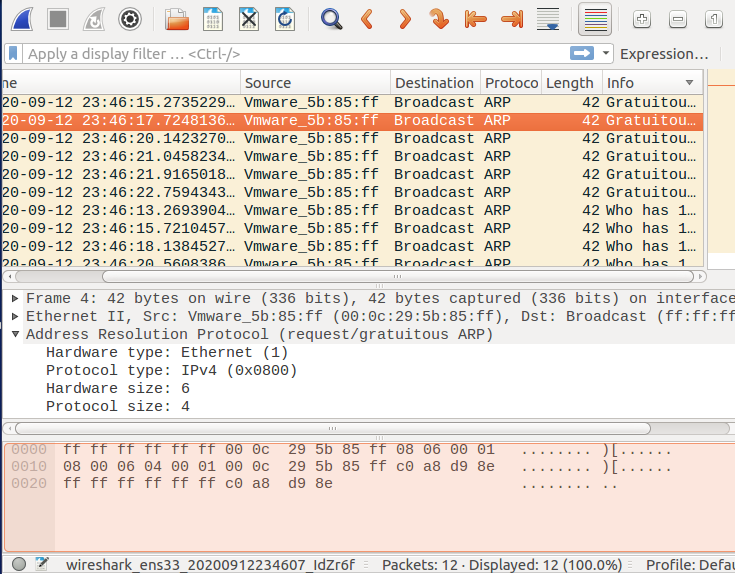
# ARP Cache Poisoning Attack Lab

设置三个VM虚拟机A(192.168.217.140)、B(192.168.217.141)、C(192.168.217.142)，A为攻击者，B为受害者。A发送arp信息使得B的arp缓存中记录的C的IP地址映射到A的硬件地址中，即IP地址192.168.217.142与00:0c:29:5b:85:ff对应。初始的B的ARP缓存如图。

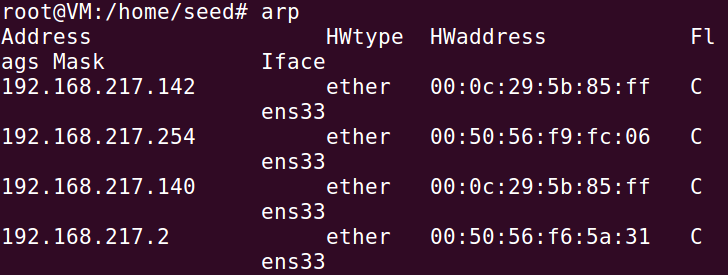


A持续向B发送ARP gratuitous报文，IP源地址和目的地址相同，目的硬件地址是ff:ff:ff:ff:ff:ff，源硬件地址是默认A的硬件地址。

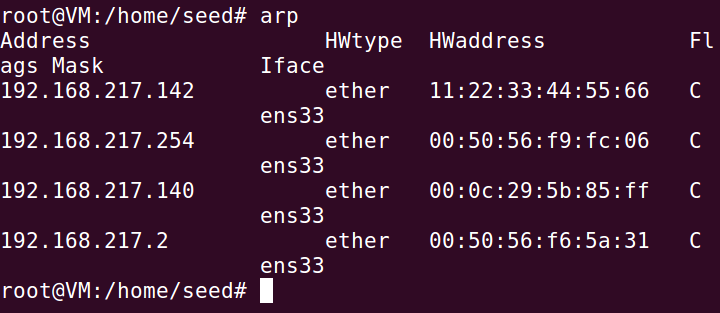




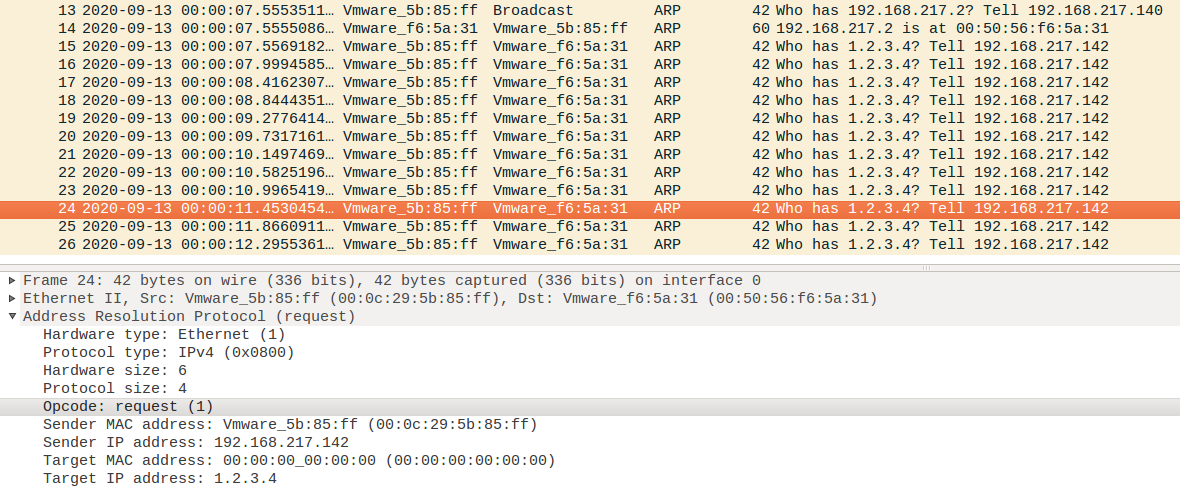
如图可见B的arp缓存中C的IP地址已经被映射到A的硬件地址中。

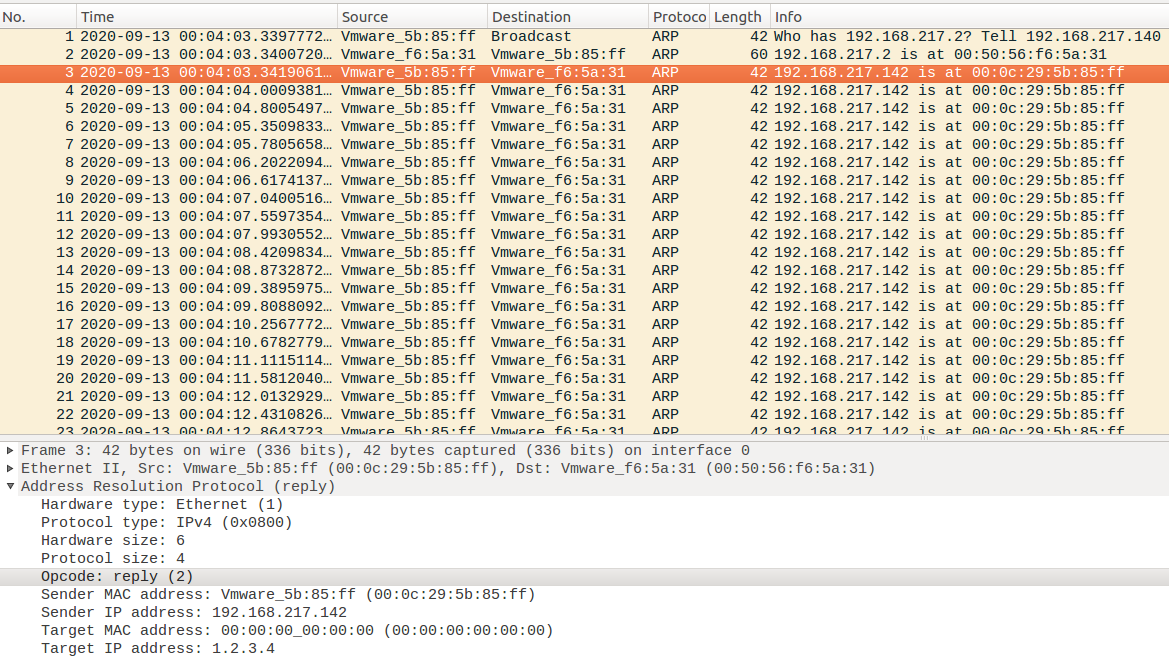


如果在发送的ARP报文中将初始的源硬件地址自定义修改(如11:22:33:44:55:66)，则可以任意定义映射。



再尝试应用普通的ARP请求报文（op=1）和ARP回复报文（op=2）进行发送，结果并不能改变ARP缓存中的映射。



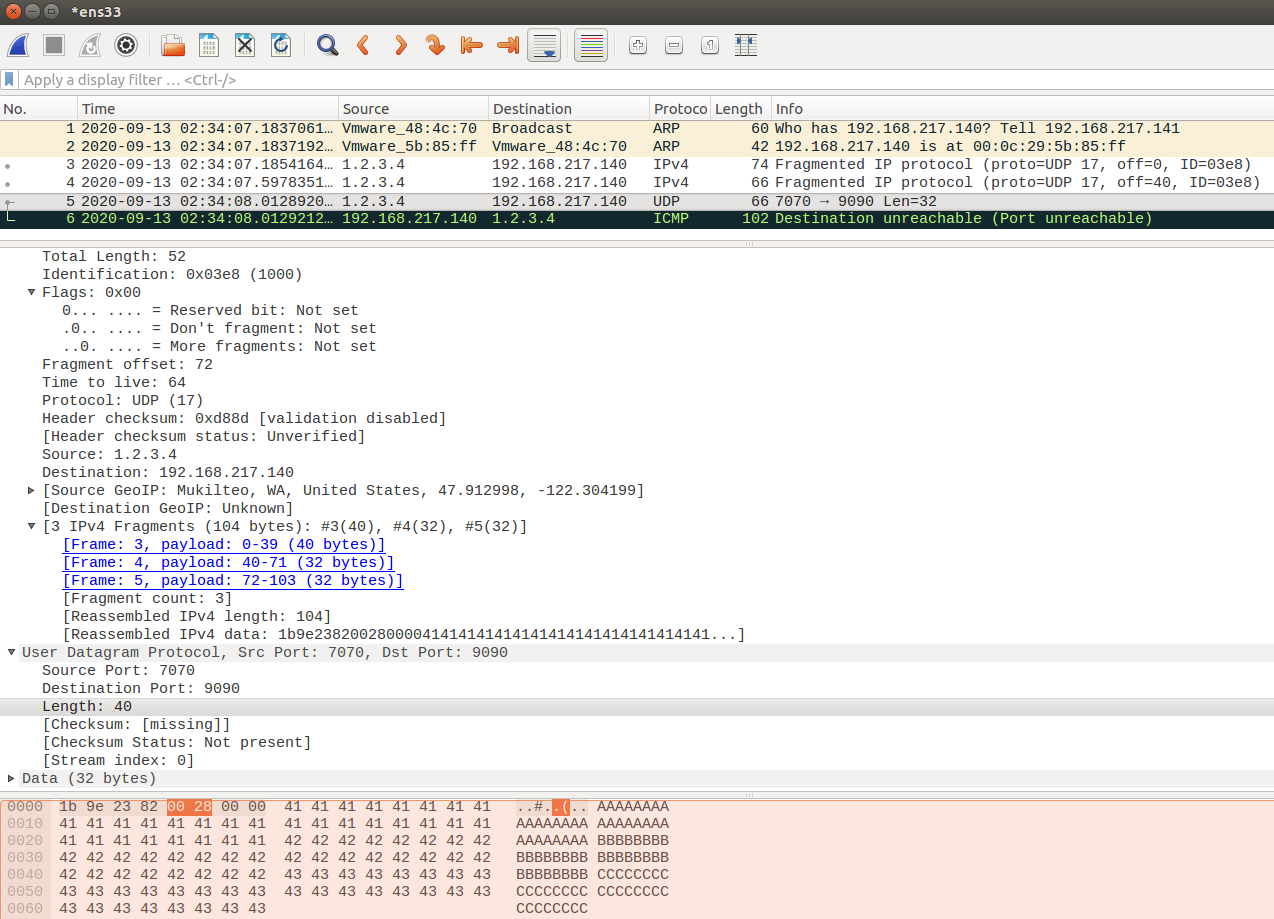


# IP Fragmentation

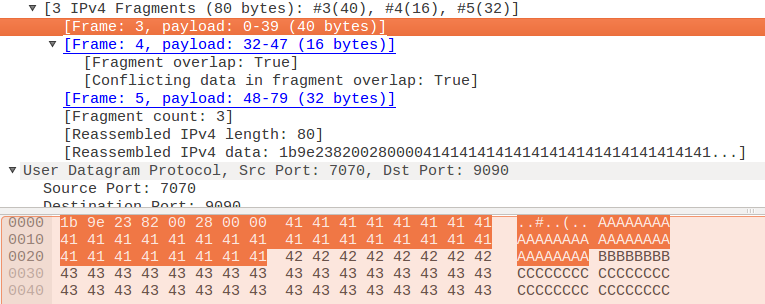
如图，构造三个IP报文分段，设置相应的标识、标志、偏移量和数据。三个分段的源地址、目的地址和标识都是一定的。第一个分段需要包含udp头部的8个字节（包括源端口、目的端口和检验和），数据部分32字节，IP头部20字节，因此IP总长度为60字节；其他两个分段没有udp头部，因此总长为52字节，通过设置协议为17表示运送的是UDP数据段，在之后可以将三个拼装在一起。



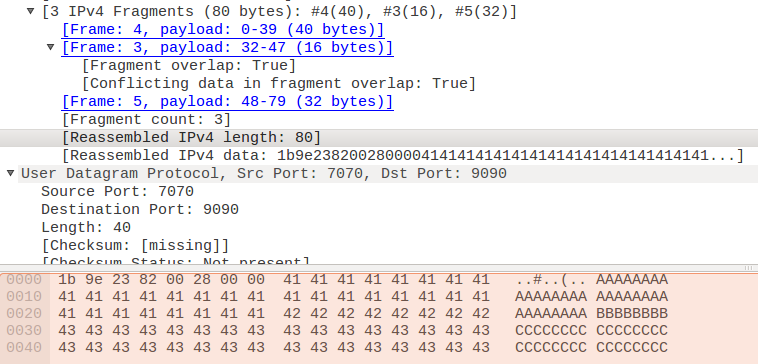
接收端接受如下。若设置自动拼装分片，则可以看到整个数据段为104字节，除去UDP头部的数据部分为96字节。



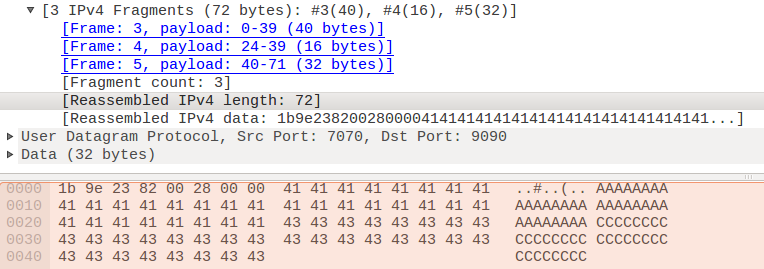
设置K=8，并且第二个分段的数据长度设置为16，共有8字节重合，而最终拼装的时候第二个分段的前8字节被抵消。

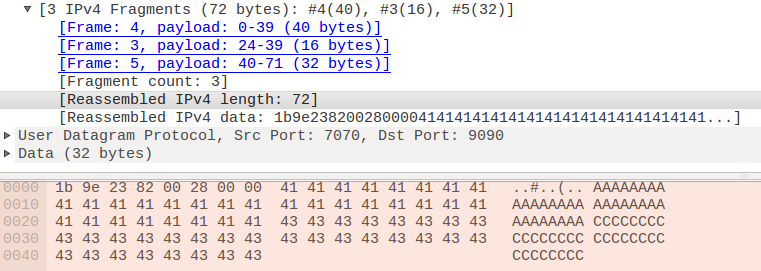


先发送第二个分段效果一样。

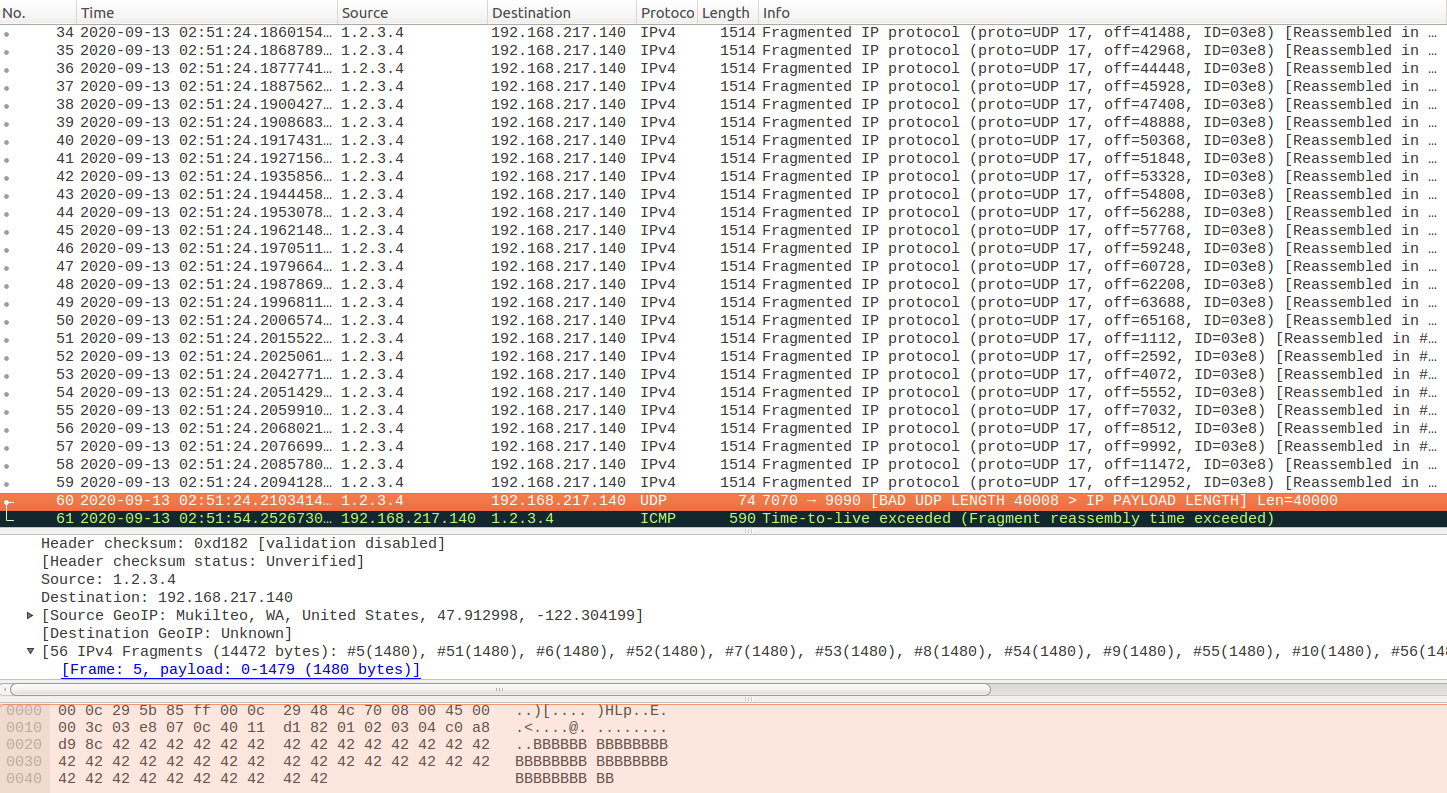


若第二个分段全部在第一个分段内，则它被被完全抵消，发送顺序无影响。





尝试发送80000字节的数据（40000个A和40000个B），在接收端收到56个分组。其中偏移值若超出65536，则取模。



若不发送第二个分段，则在一段时间后（约30秒），服务器会向客户端发送分段超时的ICMP报文。

