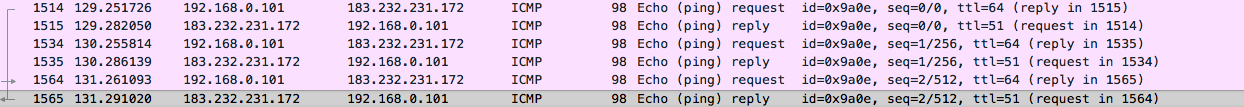
 网络协议分析

**ping的实现**

2019/4/30

1. **ICMP实现原理**

使用系统命令ping百度，抓取icmp Request/Response报文



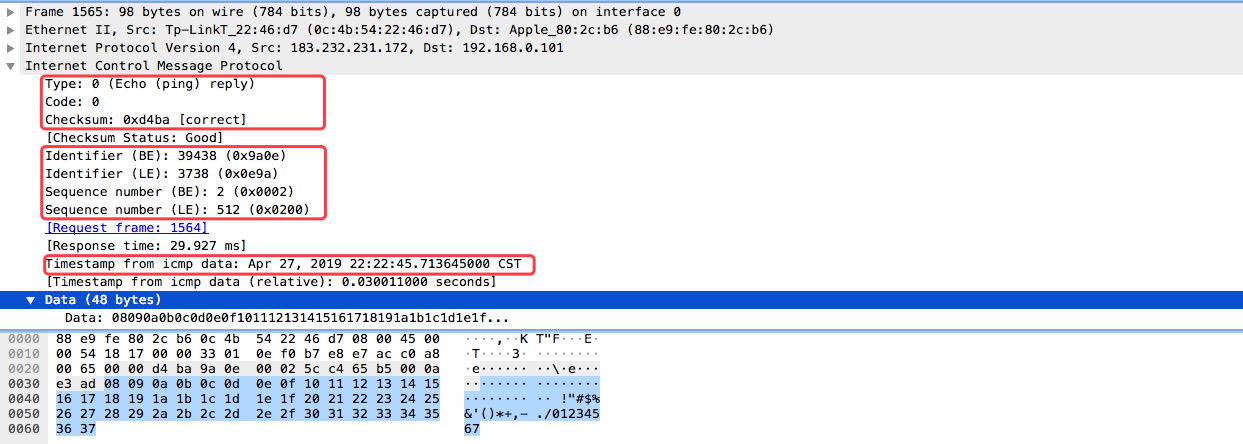
**分析icmp报文，可知：**

* ICMP 消息使用IP头作为基本控制。ip头部为20字节，包含源地址、目的地址等信息
* icmp Reques/Responset消息头部为8字节，后接data部分。

**icmp Reques/Responset包结构如下：**

* type字段，值为0表示icmp response，值为8表示icmp Request
* code字段，值通常为0
* checksum，报文首部加数据的检验和
* identifier，请求标识符，在类unix系统下为[icmp](https://www.baidu.com/s?wd=icmp&tn=SE_PcZhidaonwhc_ngpagmjz&rsv_dl=gh_pc_zhidao) [Id](https://www.baidu.com/s?wd=Id&tn=SE_PcZhidaonwhc_ngpagmjz&rsv_dl=gh_pc_zhidao)entifier的内容就是[ping](https://www.baidu.com/s?wd=ping&tn=SE_PcZhidaonwhc_ngpagmjz&rsv_dl=gh_pc_zhidao)的进程号pid
* sequence number，当前请求标识符下报文的序号
* timestamp from icmp data：发送当前identifier和sequence number的icmp Request报文的时间
* data：数据部分，icmp Request中自定义，icmp Response中回应相同的data

|  |  |  |
| --- | --- | --- |
| Type | Code | checksum |
| identifier | | sequence number |
| (timestamp from icmp data) data | | |



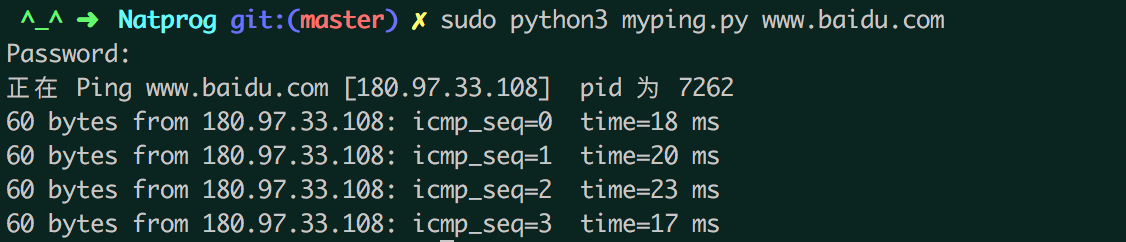
1. **python实现**

**设计思路**

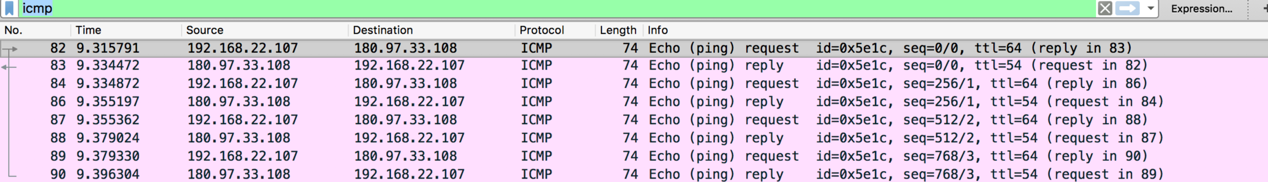
1. 获取需要ping的主机地址，创建原始套接字
2. 构造icmp Request数据报文，填充首部信息，在报文数据部分写入报文发送时间，计算校验和
3. 使用原始套接字向目的主机发送icmp Request
4. 套接字接收数据包，提取ip头部后面的数据部分，检验是否为icmp Response报文，比对identifier、sequence number与发送报文是否相同，相同则提取数据部分的时间信息，计算发送报文与接收回应之间的时间差，不同则丢弃报文，继续接收下一个。

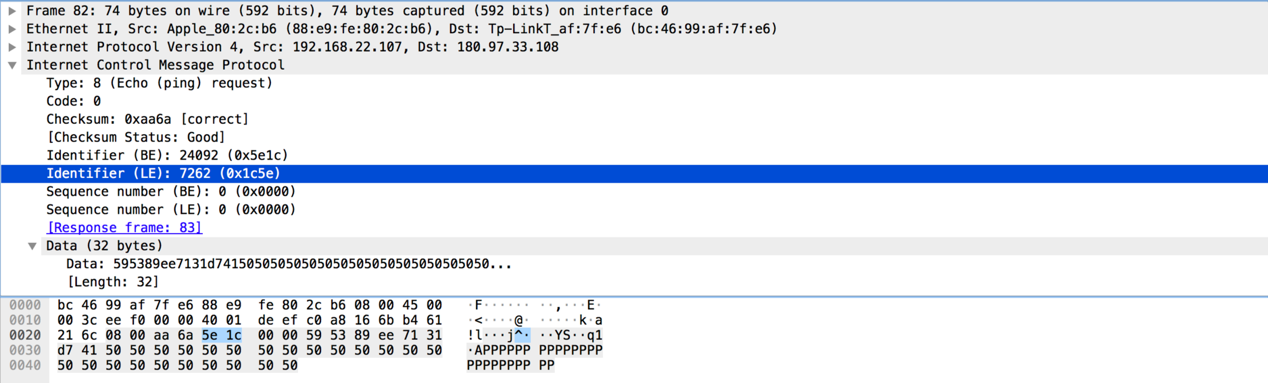
**实验结果截图**

使用sudo权限创建原始套接字socket.raw，ping百度，实验结果如下

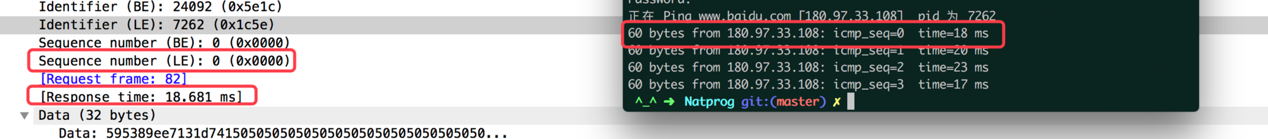


抓包可见当前发送的icmp数据包identifier字段与程序的pid相同





比对同一数据包wireshark抓包和ping.py得到的时延，误差在可接受范围内。



1. **实验代码及注释**
2. **import** argparse, socket
3. **import** time,struct,select,os
5. ICMP\_ECHO\_REQUEST = 8
6. ICMP\_ECHO\_RESPONSE = 0
7. ICMP\_CODE = 0
9. **def** ping(host):
10. icmp = socket.getprotobyname("icmp")
11. **try**:
12. ping\_socket = socket.socket(socket.AF\_INET, socket.SOCK\_RAW, icmp)
14. **except** socket.error:
15. **print**("ICMP messages can only be sent from processes")
16. **print**("create socket\_raw fail")
17. **print**("please runing ad root")
18. **raise** socket.error("create socket\_raw fail")
20. dest\_addr = socket.gethostbyname(host)
21. my\_ID = os.getpid() & 0xFFFF
22. **print**("正在 Ping {0} [{1}]  pid 为 {2}".format(host,dest\_addr,my\_ID))
24. **for** ping\_seq **in** range(0,4):
25. send\_ping\_request(ping\_socket,my\_ID,ping\_seq,dest\_addr)
26. pk\_size,seq,delay = receive\_ping\_response(ping\_socket,my\_ID,ping\_seq)
27. **if** delay > 0:
28. **print**("{} bytes from {}: icmp\_seq={}  time={} ms".format(pk\_size,dest\_addr,seq,int(delay\*1000)))
29. **else**:
30. **print**("icmp Request timeout")

33. **def** send\_ping\_request(ping\_socket,ID,ping\_seq,dest\_addr):
34. icmp\_packet = create\_icmp\_packet(ID,ping\_seq)
35. ping\_socket.sendto(icmp\_packet, (dest\_addr, 1))

38. **def** create\_icmp\_packet(ID,seq):
39. data\_checksum = 0
40. # 创建icmp Request的首部字段，checksum先为0
41. header = struct.pack("bbHHh", ICMP\_ECHO\_REQUEST, ICMP\_CODE, data\_checksum, ID, seq)
42. bytesInDouble = struct.calcsize("d")
43. data = (32 - bytesInDouble) \* b"P"
44. data = struct.pack("d", time.time()) + data
46. # 计算首部和数据部分的校验和，（将其填入）重新构造icmp Request首部报文
47. new\_checksum = checksum(header + data)
48. header = struct.pack("bbHHh", ICMP\_ECHO\_REQUEST, ICMP\_CODE, socket.htons(new\_checksum), ID, seq)
50. packet = header + data
51. **return** packet

54. **def** receive\_ping\_response(ping\_socket, ID, seq,timeout=2):
55. **while** True:
56. timeLeft = timeout
57. startedSelect = time.time()
58. whatReady = select.select([ping\_socket], [], [], timeout)
59. howLongInSelect = (time.time() - startedSelect)
61. **if** whatReady[0] == []:
62. # Timeout
63. **return** -1
65. timeReceived = time.time()
66. recPacket, addr = ping\_socket.recvfrom(1024)
67. # recPacket[0:20] 前20bytes为ip首部
68. # recPacket[20:28]后8bytes为icmp首部
69. icmpHeader = recPacket[20:28]
70. type, code, checksum, packetID, packetSeq = struct.unpack(
71. "bbHHh", icmpHeader
72. )
74. # type==3 为 destination unreachable
75. **if** type ==3 :
76. **print**("Destination Unreachable")
77. **return** -1
78. **if** type == ICMP\_ECHO\_RESPONSE **and** packetID == ID **and** packetSeq == seq:
79. bytesInDouble = struct.calcsize("d")
80. # 提取data中包含的报文发送时间信息
81. timeSent = struct.unpack("d", recPacket[28:28 + bytesInDouble])[0]
82. # 返回接收的数据包长度，序号，时延
83. **return** len(recPacket),packetSeq,timeReceived - timeSent
85. # 连续等待接收的总时间不能超过规定的时延限制
86. timeLeft = timeLeft - howLongInSelect
87. **if** timeLeft <= 0:
88. **return** -1

91. **def** checksum(data):
92. n = len(data)
93. m = n % 2
94. sum = 0
96. # data以每两个字节（十六进制）通过ord转十进制，第一字节在低位，第二个字节在高位
97. **for** i **in** range(0, n - m ,2):
98. sum += (data[i]) + ((data[i+1]) << 8)
99. **if** m:
100. sum += (data[-1])
102. #将高于16位与低16位相加
103. sum = (sum >> 16) + (sum & 0xffff)
104. # 如果还有高于16位，将继续与低16位相加
105. sum += (sum >> 16)
106. answer = ~sum & 0xffff
108. #主机字节序转网络字节序列（大端序）
109. answer = answer >> 8 | (answer << 8 & 0xff00)
110. **return** answer

113. **if** \_\_name\_\_ == '\_\_main\_\_':
114. parser = argparse.ArgumentParser(description='TCP Echo Client')
115. parser.add\_argument('host',  help='the host or ip address you want to ping', default='0.0.0.0', type=str)
116. args = parser.parse\_args()
118. ping(args.host)

班级：16272412

学号：16272203

姓名：黄萍萍