计算机网络大作业

使用 windump 收集物理网络上流量并进行数据分析

无 34　董凯杰　2013010572

无 34　刘晨　　2013010572

2015/12/19

目录

1 数据收集及预处理

1.1 数据收集

由于现在均为自动获取的 IP 地址，因此每次登陆所用的 IP 地址均不同，需要查询本地 IP 地址后再进行 windump。采集时间为 2015 年 12 月 7 日，共 15 分钟。运行 windump 命令行格式如下：

windump –i 3 –s 80 –w traffic.pac host local IP

在采集过程中，主机进行了如下工作：

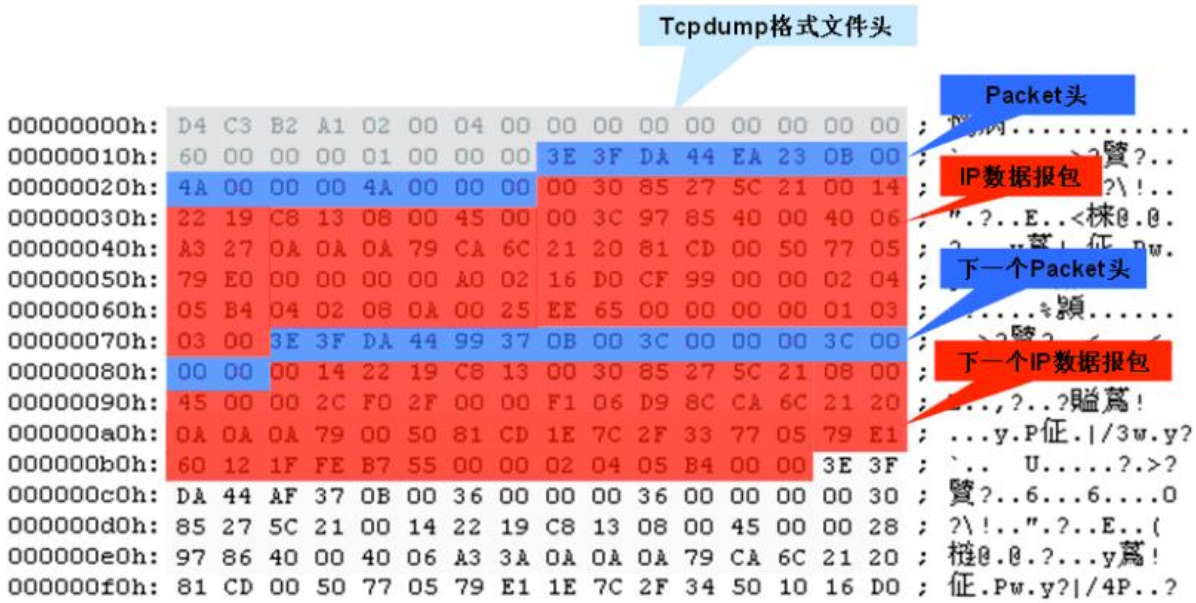
- 浏览校内外网站
- 观看视频网站
- 视频聊天
- 传送文件
- 观看直播网站

1.2 预处理

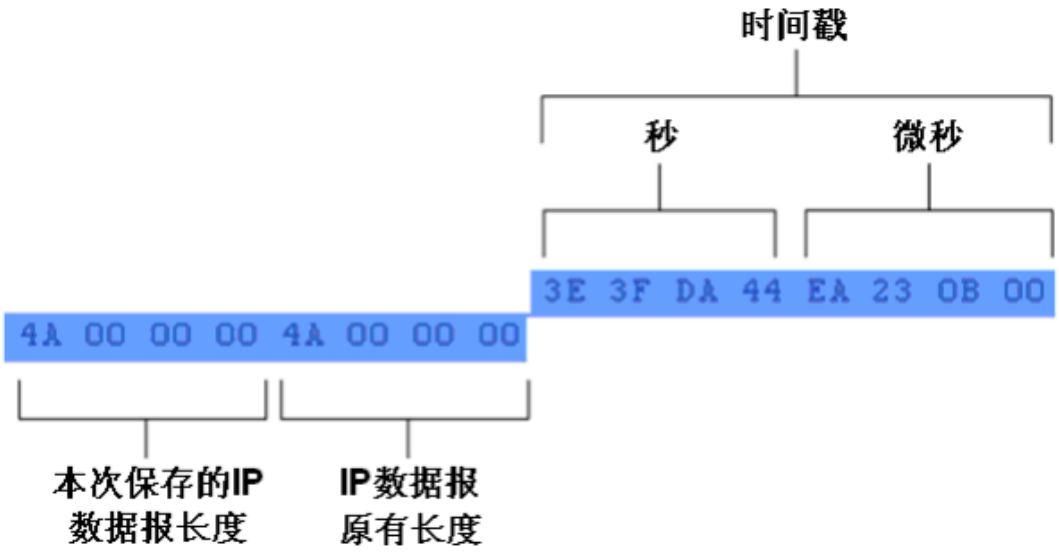
使用 windump（Linux 下使用 tcpdump）抓取下的文件是标准的二进制文件，可以使用 UltraEdit 等工具打开。分析后发现其文件分为三部分：

1. Tcpdump 格式文件头；
2. Packet 头；
3. IP 数据报包。

其中 Tcpdump 格式文件头长度为 24 个字节， Packet 头部分长度为 16 个字节。下图就是各部分内容的示意图。

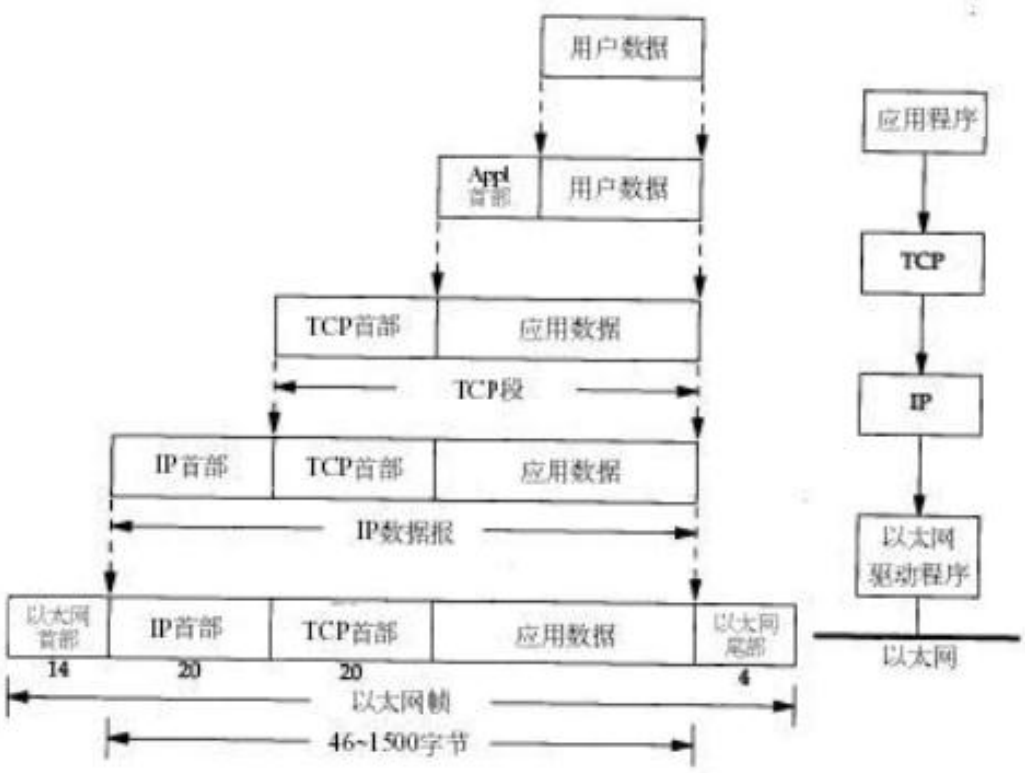


而 Packet 头各部分组成如下，其中第 9—12 个字节为本次保存的 IP 数据报长度，不过要注意的是，此部分所有数据均是颠倒后再进行保存的，因此如图所示的 4A 即是最低位。



因此预处理阶段，首先需要跳过 24 个字节的 Tcpdump 格式文件头，然后依次从 Packet 头中读取中该帧 IP 数据报的长度，再跳过这 16 个字节的 Packet 头，最后对 IP 数据报进行分析解读。

数据发送时是按照如下图所示自上而下，层层加码；数据接收时是自下而上，层层解码。



最下面一层的头部，也就是数据链路层的信息，包含 14 个字节。包括 6 个字节的目的 MAC 地址和 6 字节的源 MAC 地址以及 2 字节的类型字段，表指定网络层所用的协议类型。常见的如 IPv4:0x0800，ARP:0x0806，IPv6:0x86DD 等。而本次实验主要分析与 TCP、UDP 相关的 IPv4 协议，因此只需筛选出协议类型为 0x0800 的即可。

其次需要分析网络层的 IP 包头，主要格式如下所示：

<table>
<tr><td>版本(4)</td><td>首部长度(4)</td><td>服务类型(8)</td><td colspan="2">数据报总长(16)</td></tr>
<tr><td colspan="3">分组 ID(16)</td><td>标记(3)</td><td>段偏移量(13)</td></tr>
<tr><td colspan="2">生存时间(8)</td><td>高层协议(8)</td><td colspan="2">首部校验和(16)</td></tr>
<tr><td colspan="5">源 IP 地址(32)</td></tr>
<tr><td colspan="5">目的 IP 地址(32)</td></tr>
</table>

本次实验需要记录数据报总长，标记中的 MF 字段，段偏移量，高层协议，源 IP 地址以及目的 IP 地址。

其中数据报总长是包含 IP 头部的数据报的总长度，而在第一问对总数据量进行统计时，需要对扣除 IP 头部的数据报总长度进行累加。高层协议记录的是传输层使用的协议编号，常见的有 ICMP:1，IGMP:2，TCP:6，UDP:17 等。本次实验就需要对协议编号为 6 和 17 的进行统计分析。对于该数据报是否分段的判断需要用到 MF 和段偏移量，如果二者同时为 0，则这是一个未分片的完整的数据报；如果 MF 为 0 而偏移量不为 0，则这是一个分片了的数据报的最后一个片段；如果 MF 不为 0，则为一个分片了的数据报中间的某个片段。

最后需要分析传输层协议的包头，选取 TCP 协议，其主要格式如下所示：

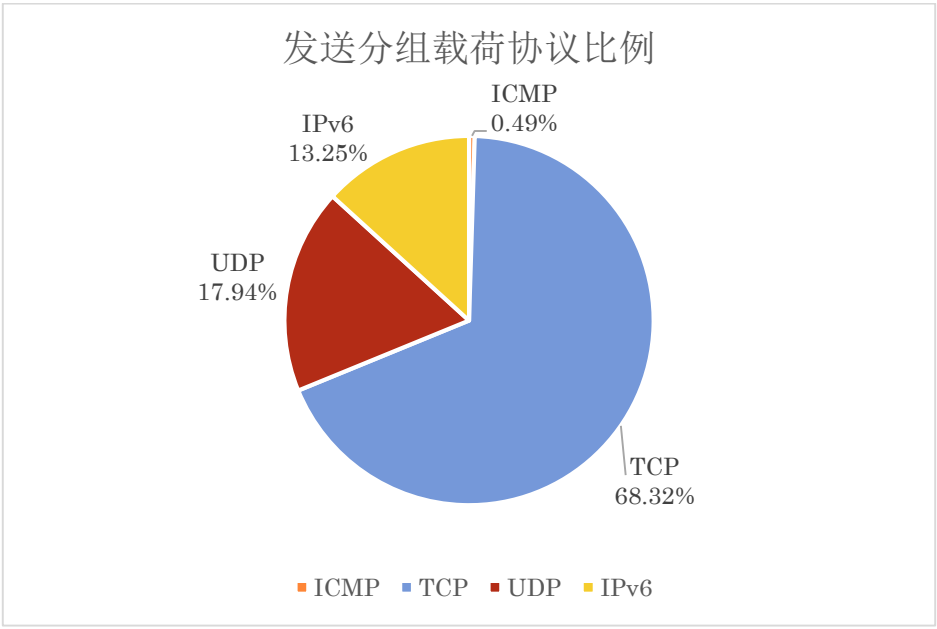
<table>
<tr><td colspan="2">源端口(16)</td><td colspan="2">目的端口(16)</td></tr>
<tr><td colspan="4">TCP 序号(32)</td></tr>
<tr><td colspan="4">捎带的确认(32)</td></tr>
<tr><td>首部长度(4)</td><td>保留(6)</td><td>Flag(6)</td><td>窗口尺寸(16)</td></tr>
<tr><td colspan="2">TCP 校验和(16)</td><td colspan="2">紧急指针(16)</td></tr>
</table>

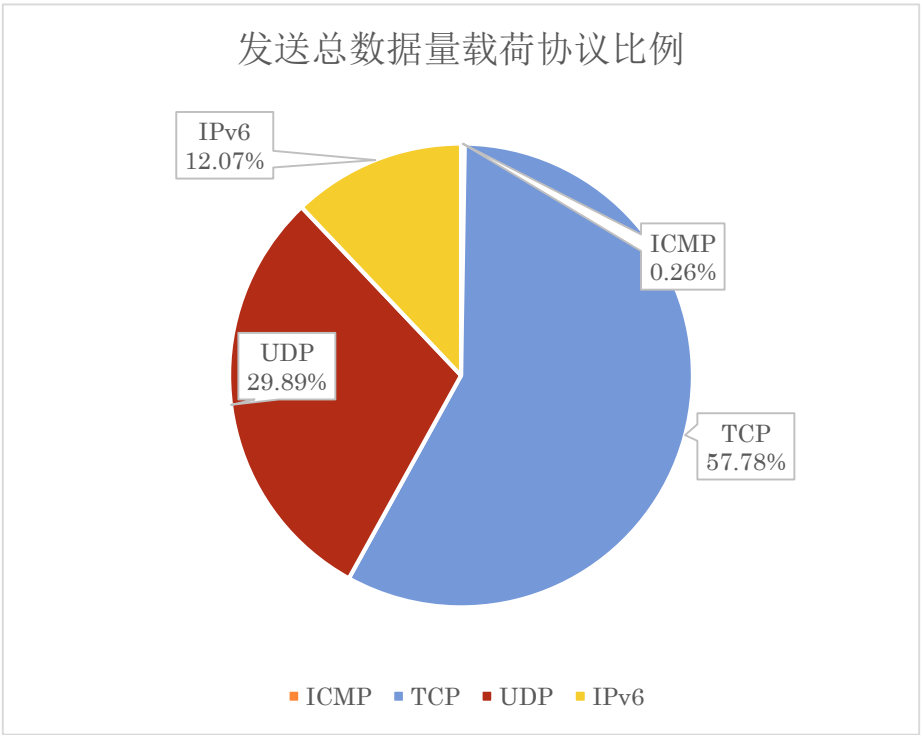
本次实验针对 TCP 协议需要记录源端口和目的端口的信息，以及 6 位的控制位。而 UDP 协议相比 TCP 协议本身比较简单，只需记录源端口和目的端口的信息即可。

2 数据分析

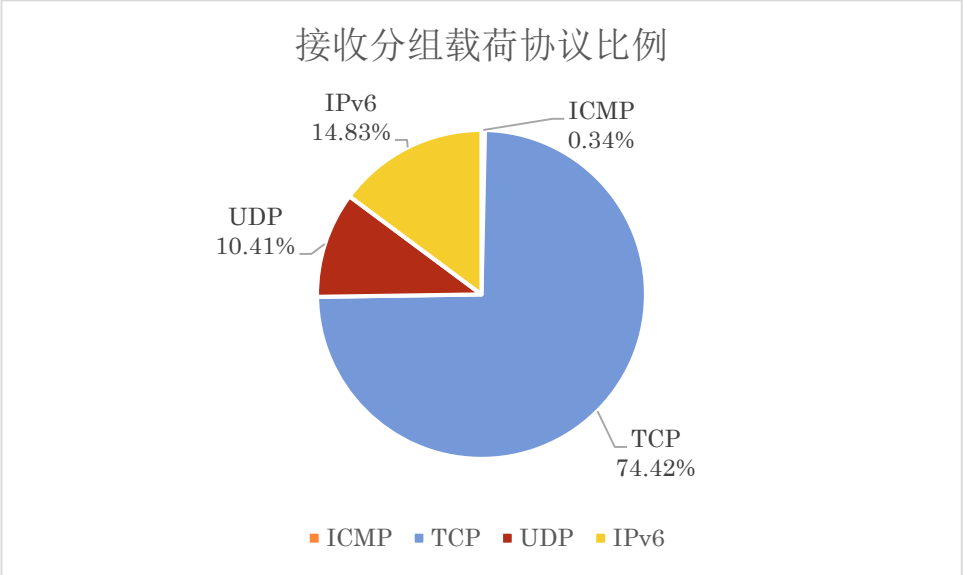
2.1 给出 IP 分组携带不同协议的载荷的饼图，分别按分组数和总数据量进行统计

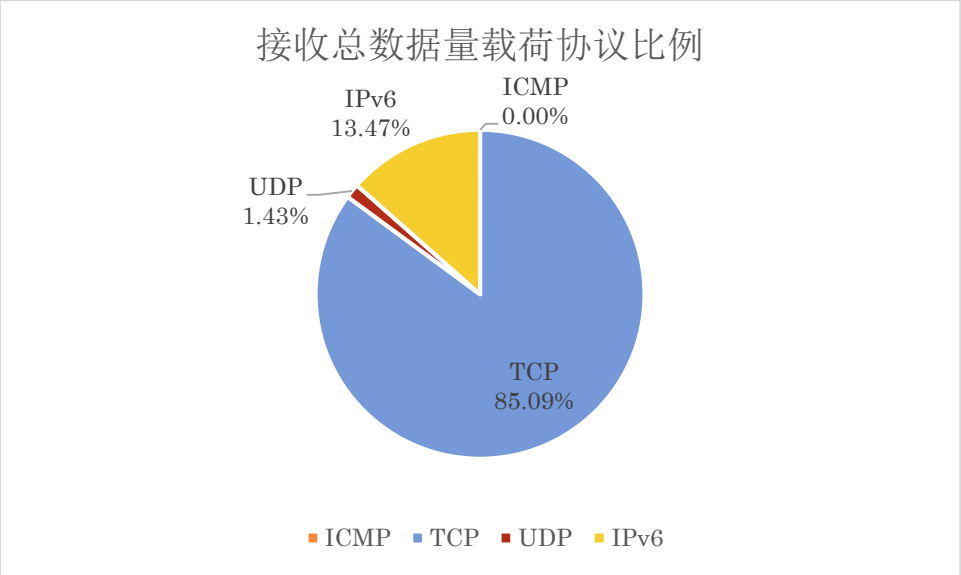
发送端 IP 分组携带不同协议的载荷情况如下两图所示：





接收端 IP 分组携带不同协议的载荷情况如下两图所示：





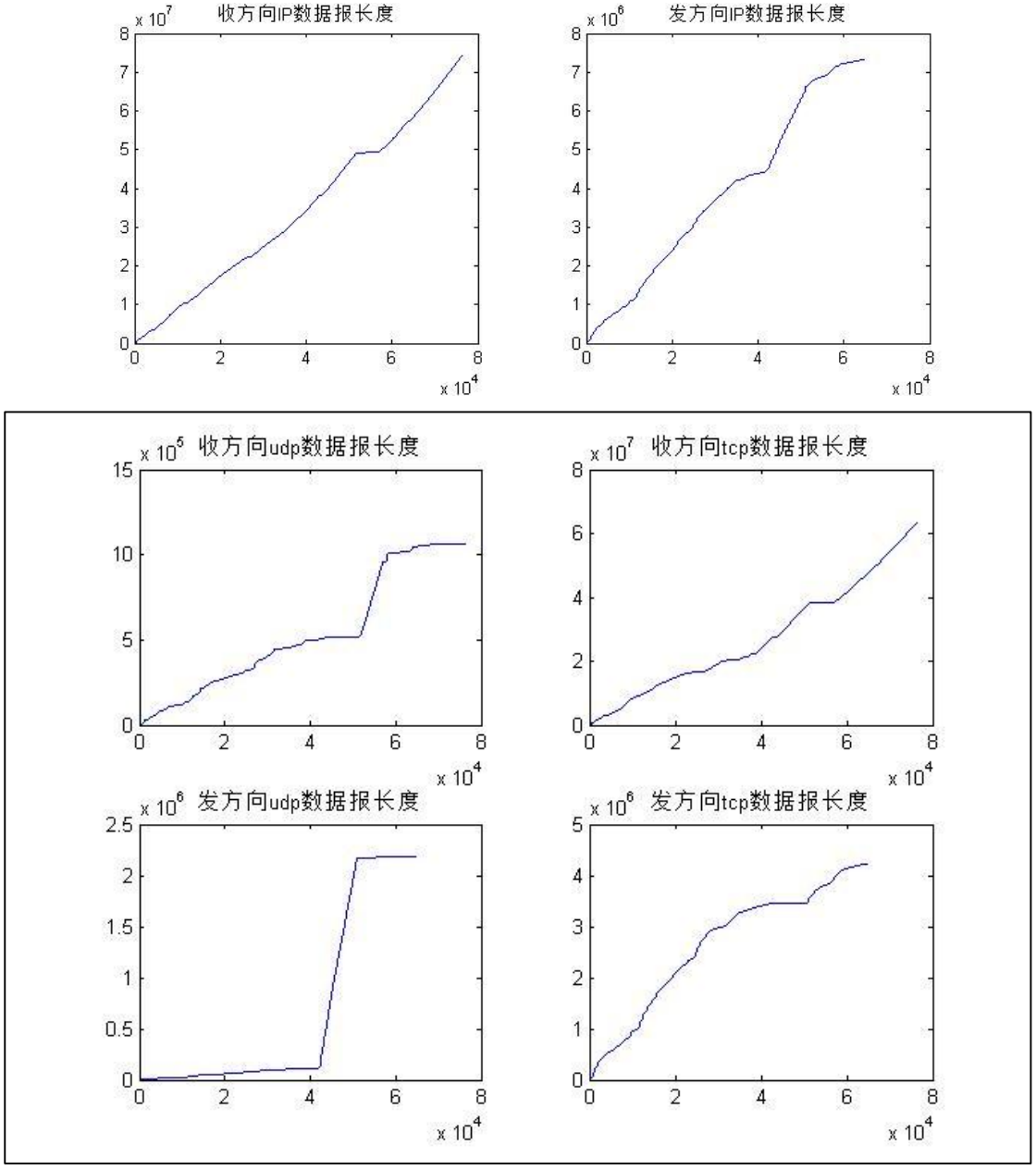
2.2 有多少 IP 分组是片段（fragment）？有多少 IP 数据报被分片？载荷为 TCP 和 UDP 的分别有多少比例的 IP 数据报被分片？

发送方向上分段的分组数： 0
发送方向上被分片的数据报数量： 0
发送方向上ICP载荷的数据报中被分片的比例： 0.000000
发送方向上UDP载荷的数据报中被分片的比例： 0.000000

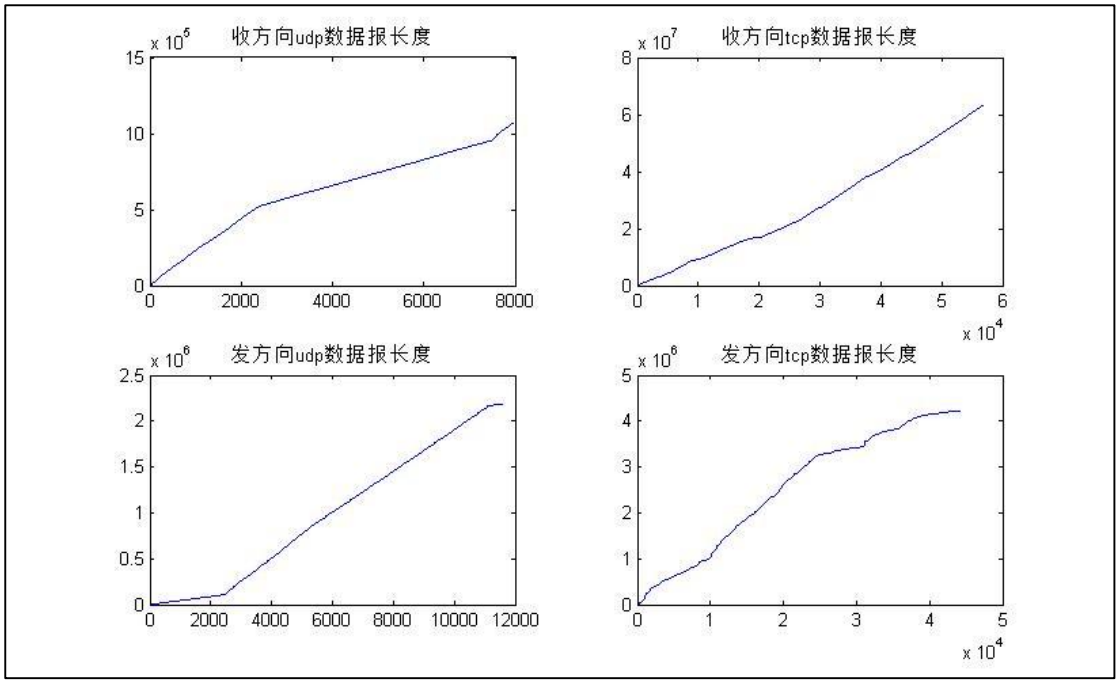
接收方向上分段的分组数： 0
接收方向上被分片的数据报数量： 0
接收方向上ICP载荷的数据报中被分片的比例： 0.000000
接收方向上UDP载荷的数据报中被分片的比例： 0.000000

本次实验收集到的 IP 数据报均未被分片。

2.3 给出 IP 数据报长度的累计分布曲线，并分别比较载荷为 TCP 和 UDP 的 IP 数据报长度的累计分布。

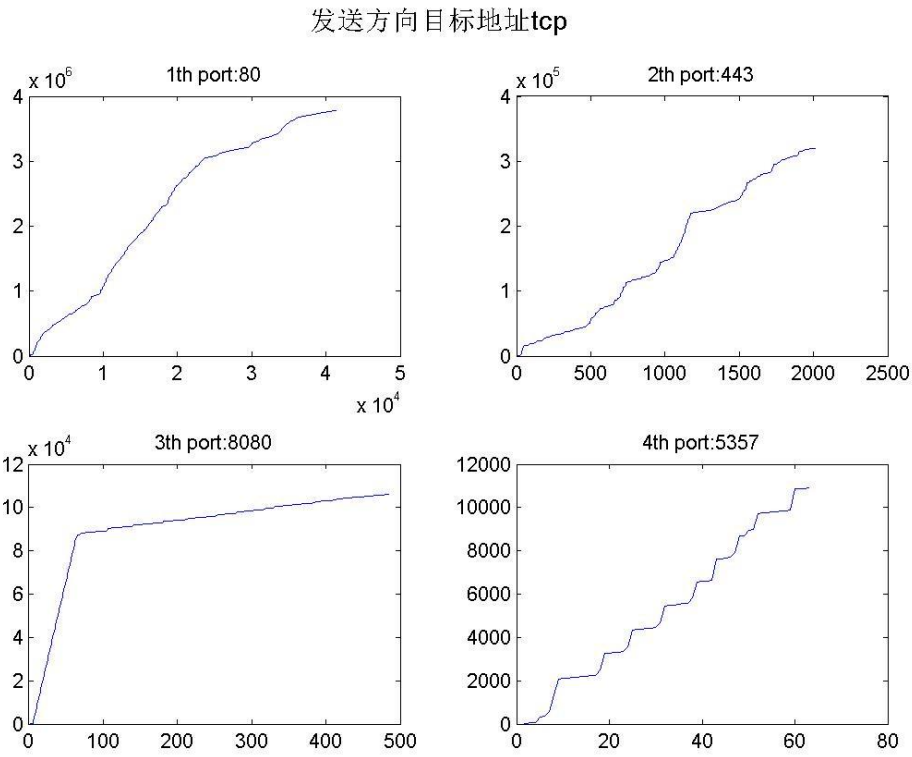


此图上下横坐标分别为收发两方向总数据包

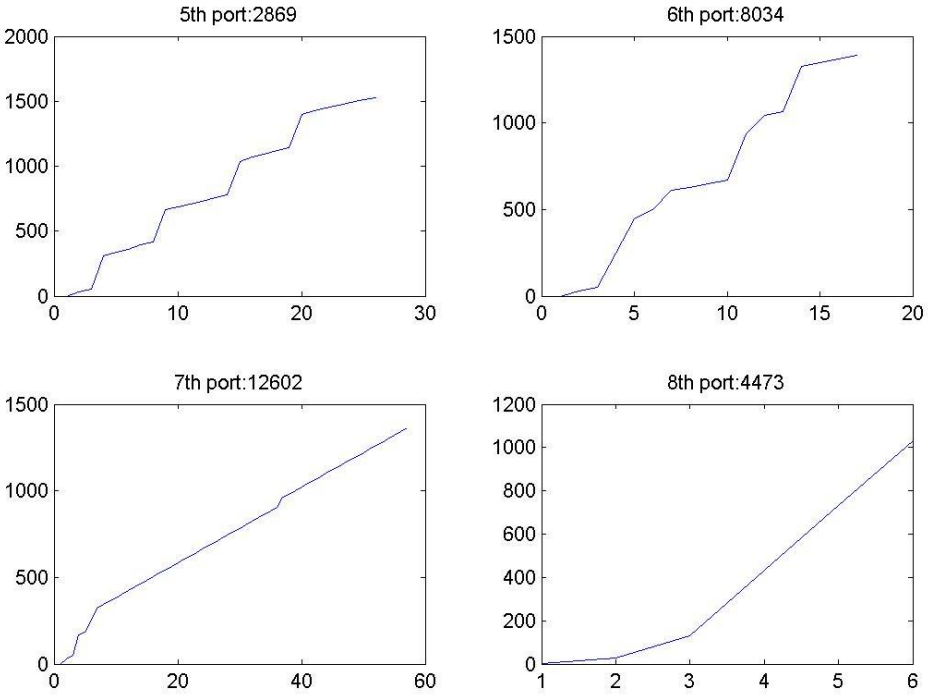


此图横坐标为各自包数量

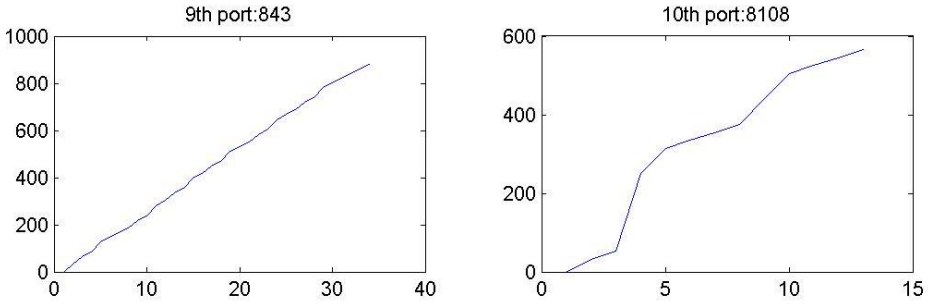
2.4 分别对 TCP 和 UDP 的 traffic 给出端口分布的直方图，比较前 10 名端口上数据报长度的累计分布曲线。

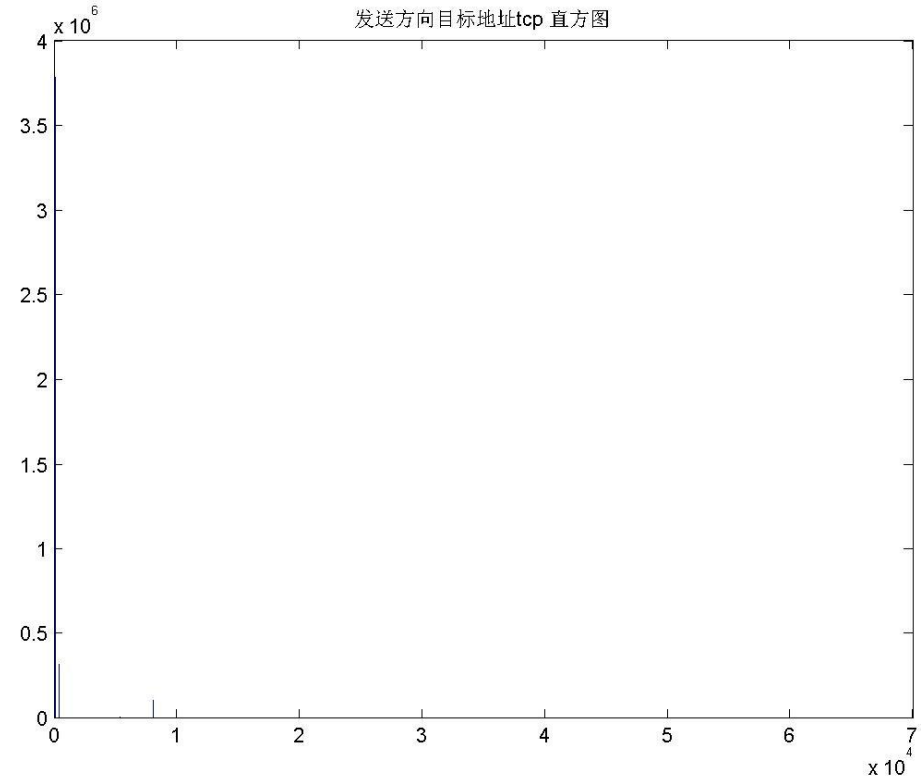


发送方向目标地址tcp

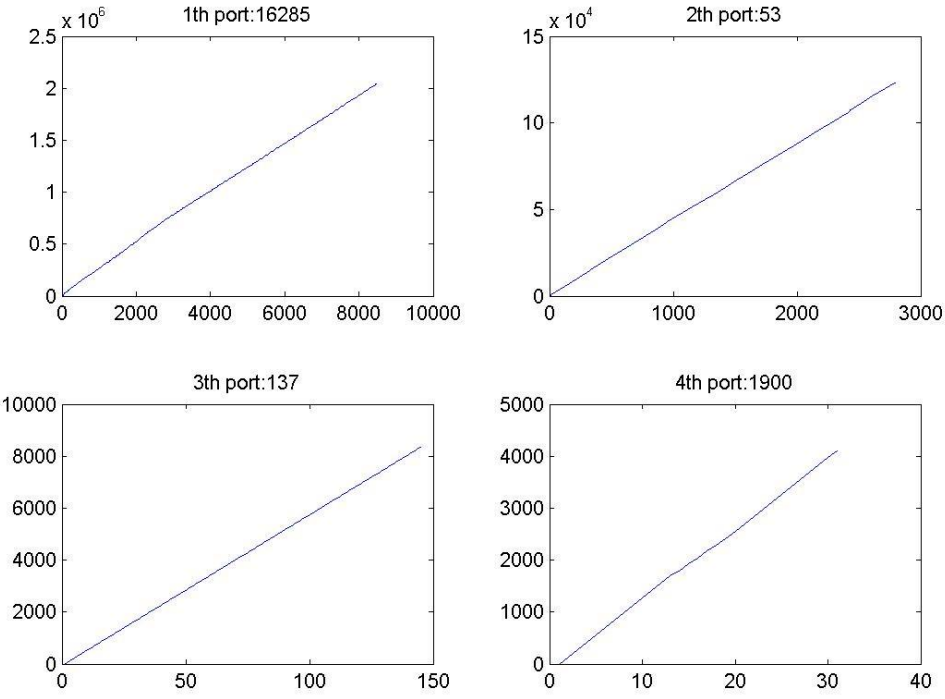


发送方向目标地址tcp

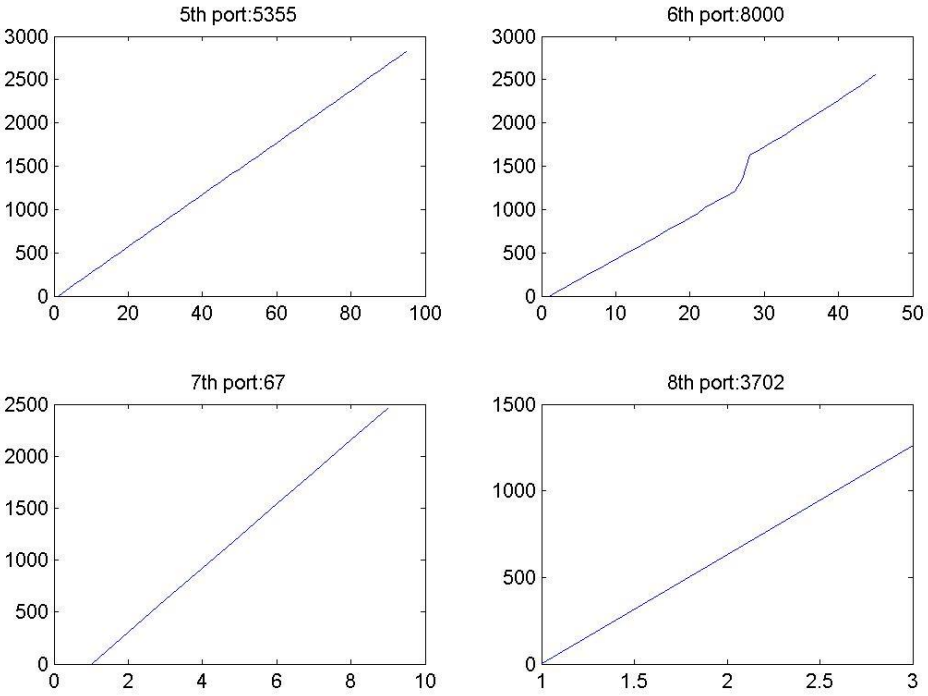




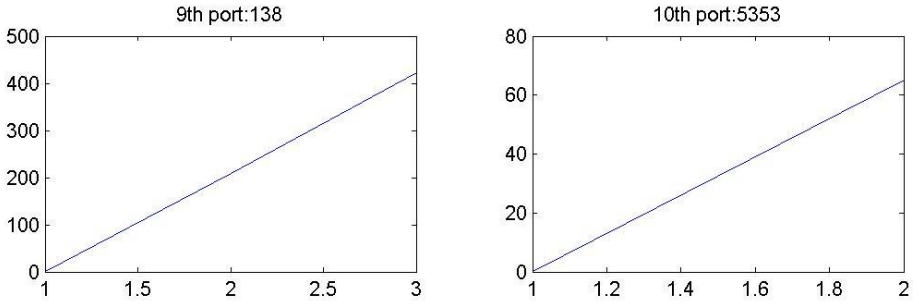
发送方向目标地址udp

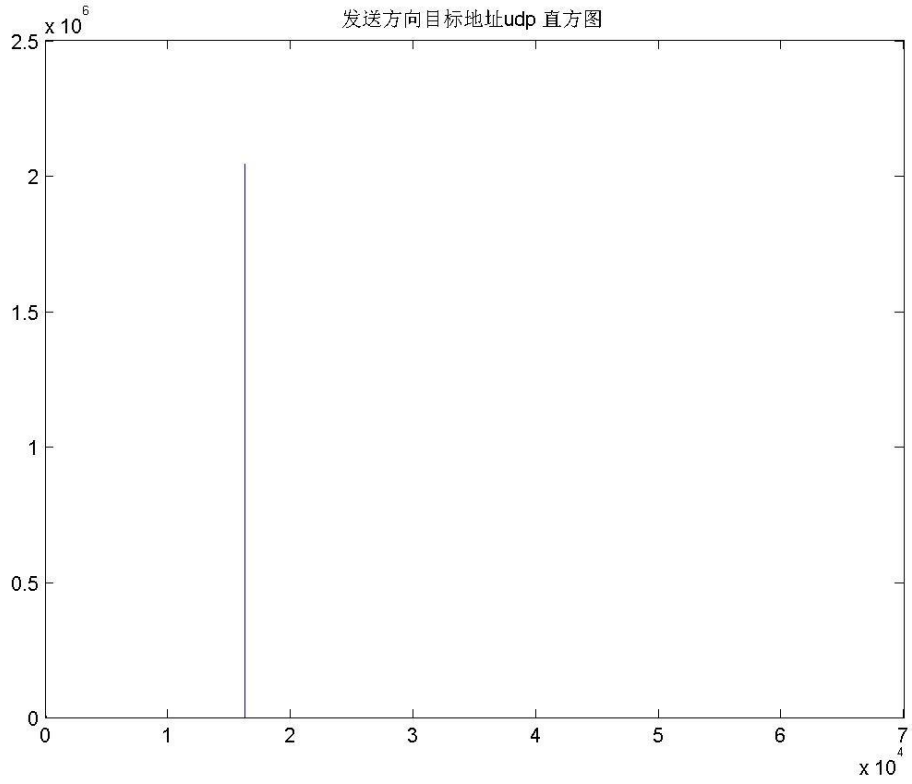


发送方向目标地址udp

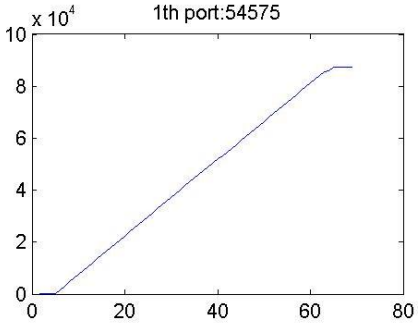


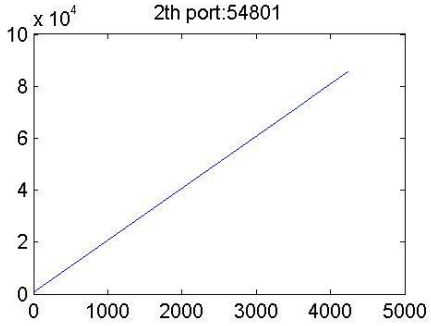
发送方向目标地址udp

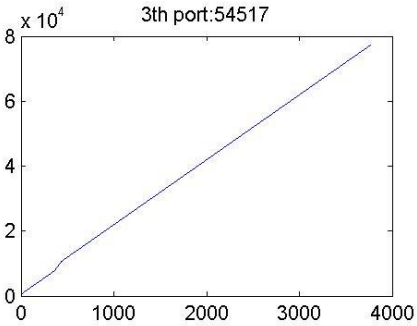


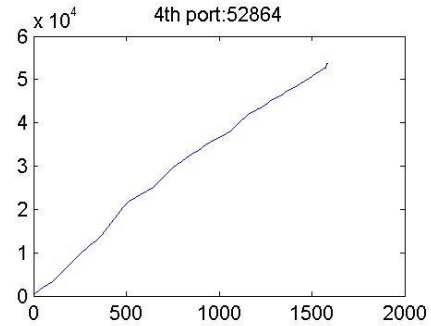

发送方向目标地址udp 直方图
x 10^6
2.5
2
1.5
1
0.5
0
0 1 2 3 4 5 6 7
x 10^4

发送方向原地址tcp

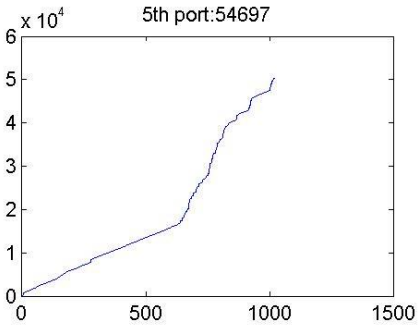

1th port:54575
x 10^4
10
8
6
4
2
0
0 20 40 60 80

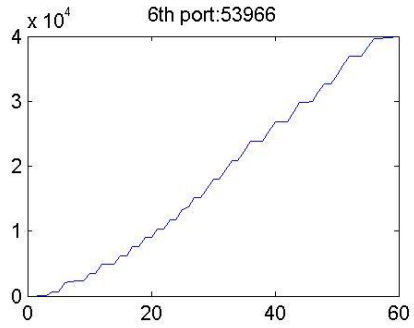

2th port:54801
x 10^4
10
8
6
4
2
0
0 1000 2000 3000 4000 5000

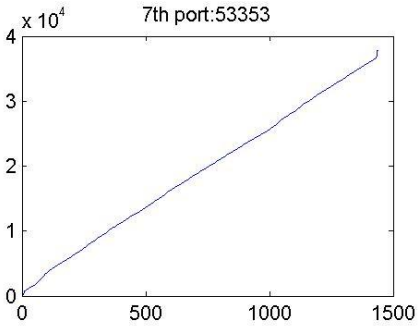

3th port:54517
x 10^4
8
6
4
2
0
0 1000 2000 3000 4000

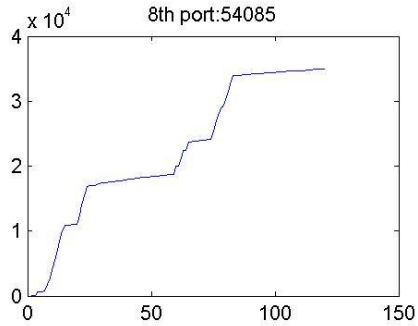

4th port:52864
x 10^4
6
5
4
3
2
1
0
0 500 1000 1500 2000

发送方向原地址tcp

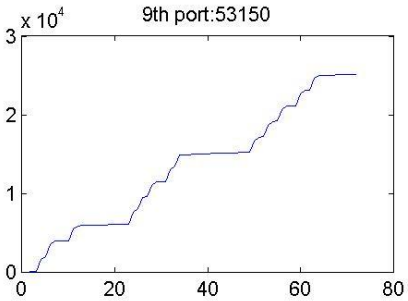


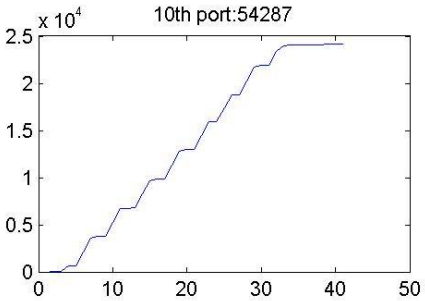


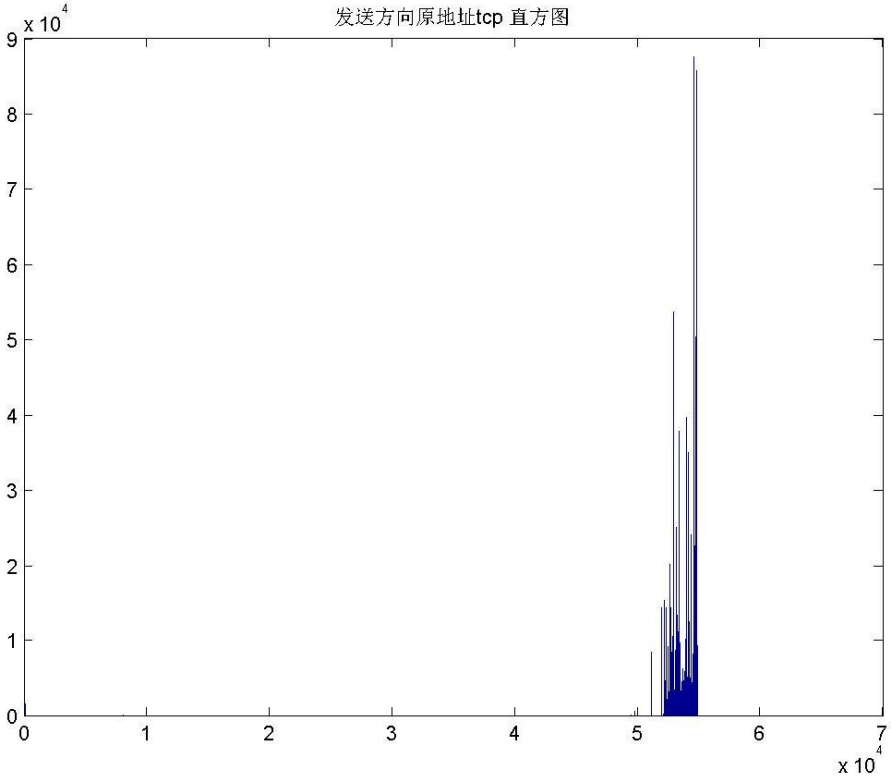




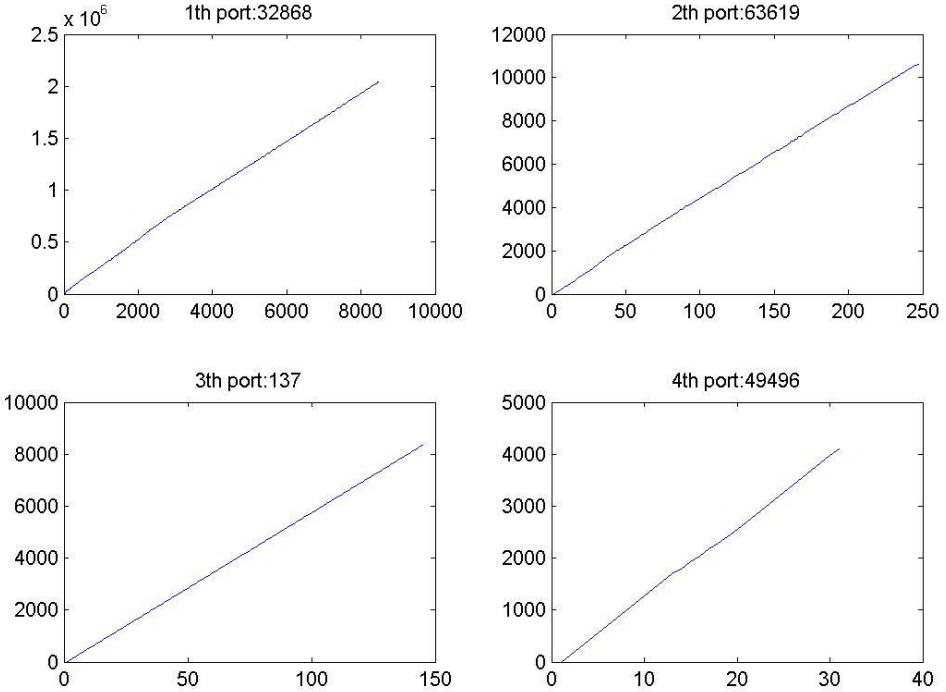
发送方向原地址tcp



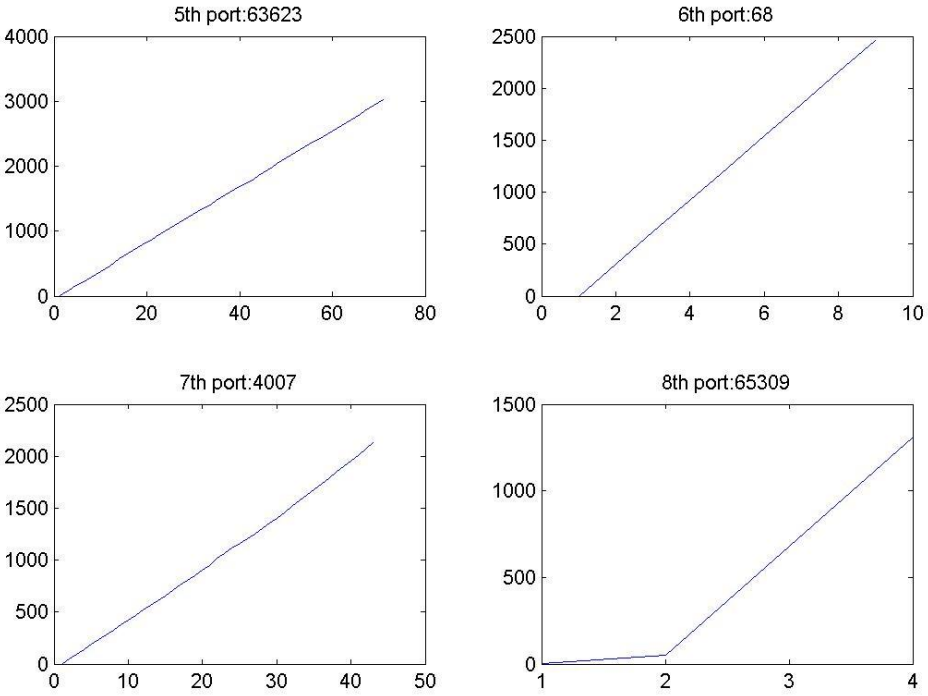



发送方向原地址tcp 直方图
x 10^4
x 10^4

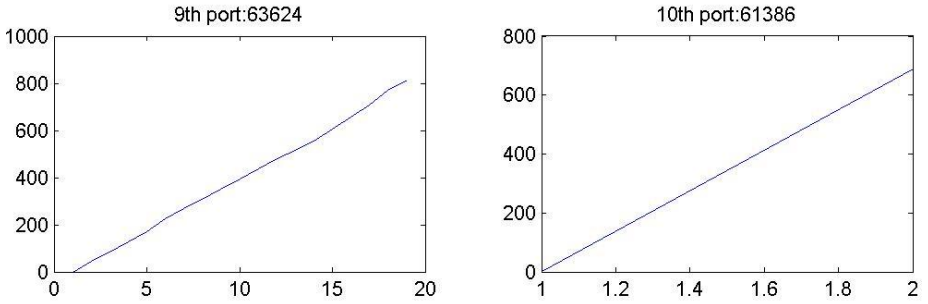
发送方向原地址udp

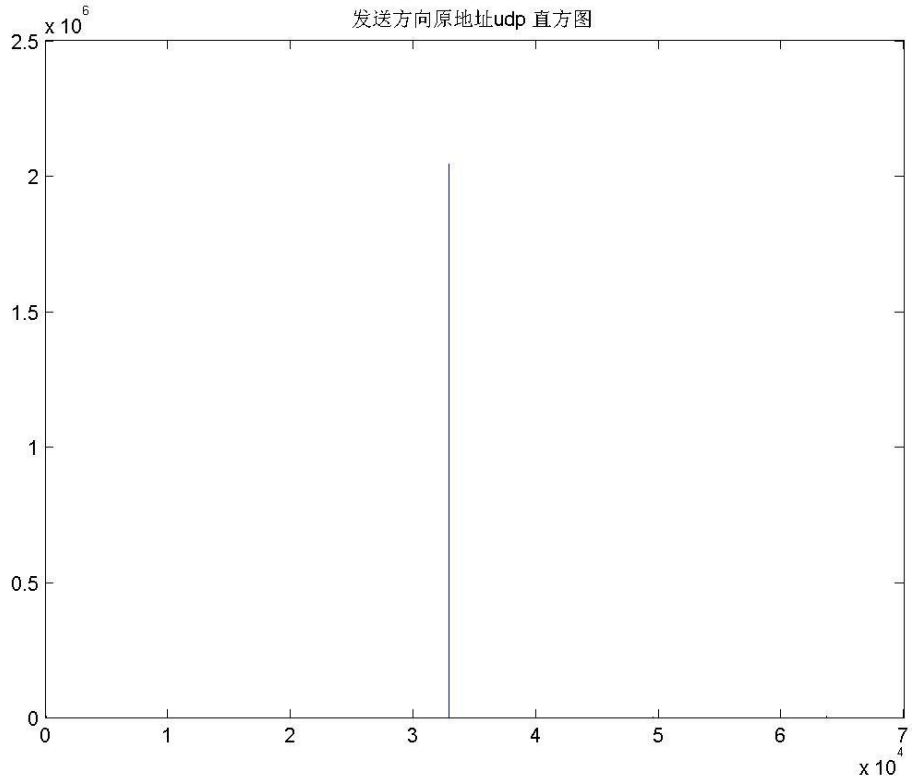

x 10^6
1th port:32868
2th port:63619
3th port:137
4th port:49496

发送方向原地址udp

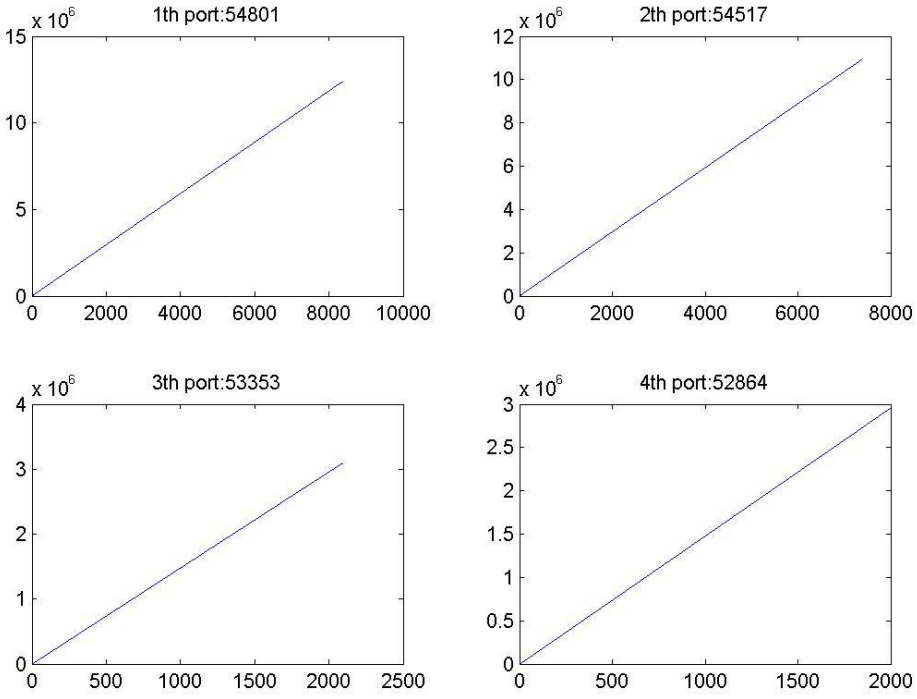


发送方向原地址udp

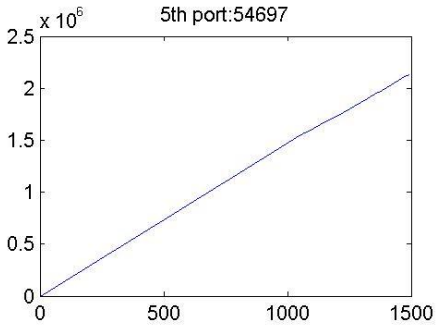


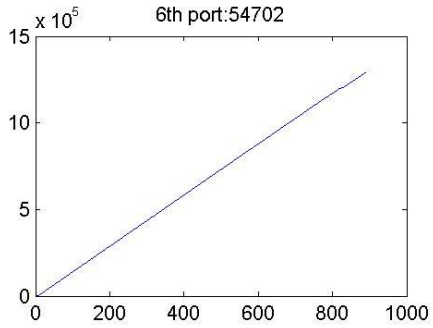


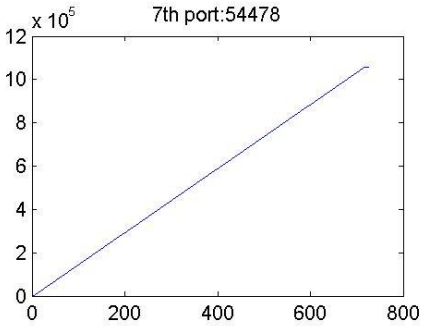
接收方向目标地址tcp

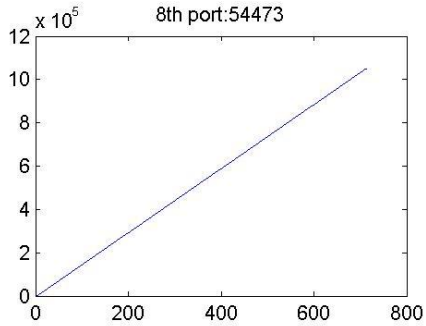


接收方向目标地址tcp

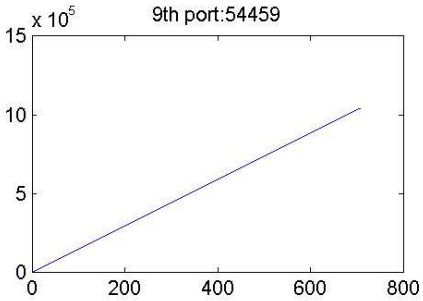


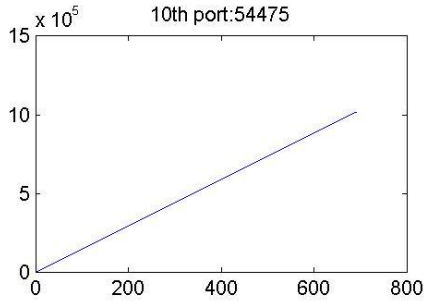


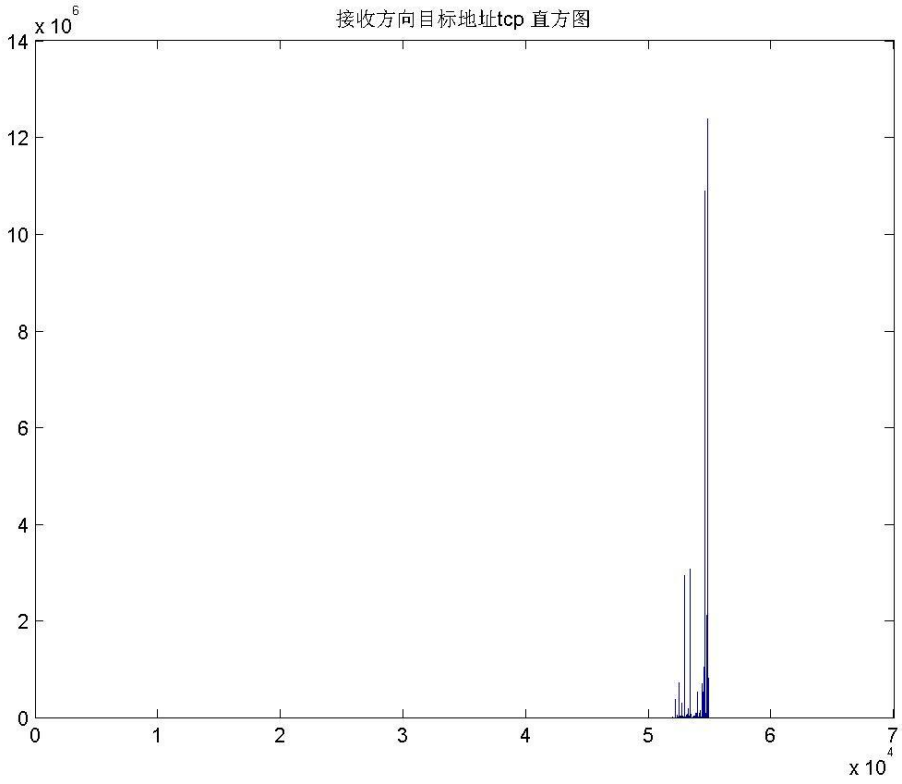




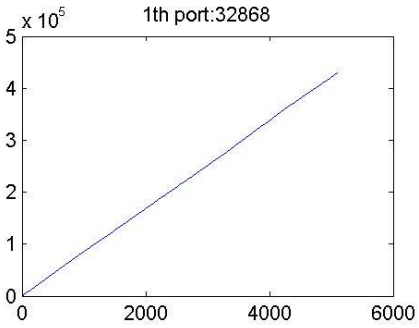
接收方向目标地址tcp

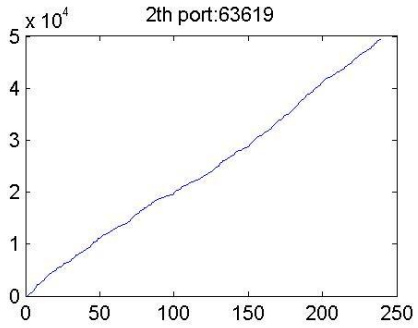


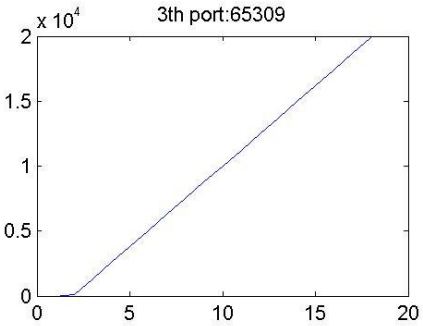


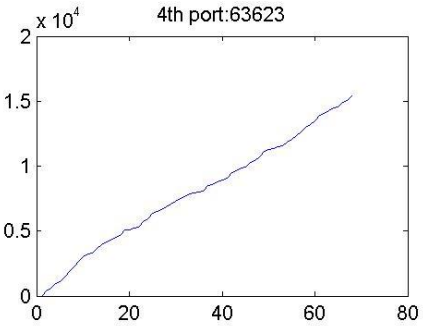


接收方向目标地址udp

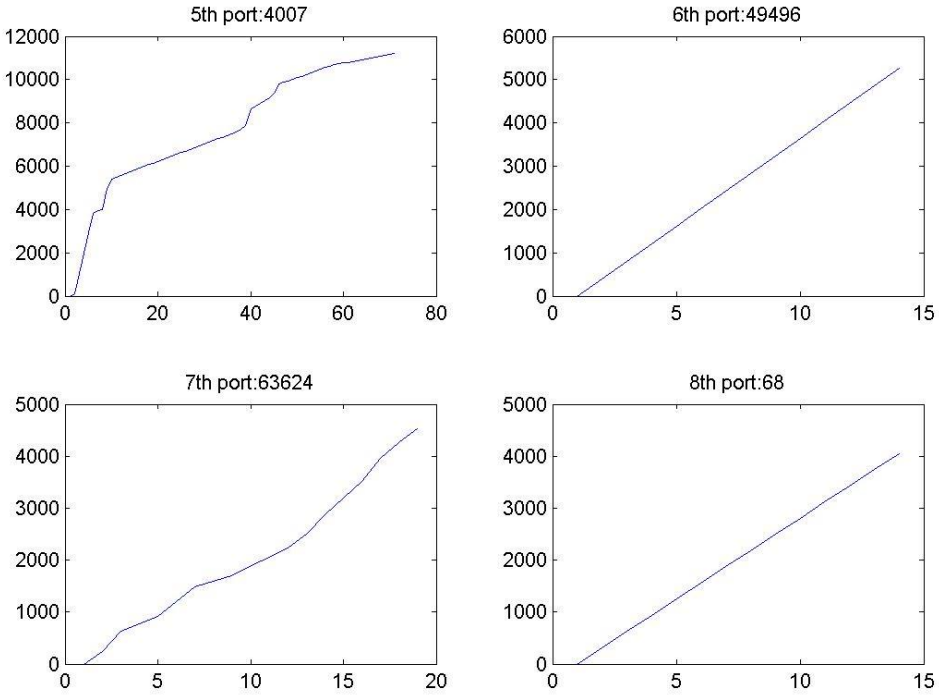




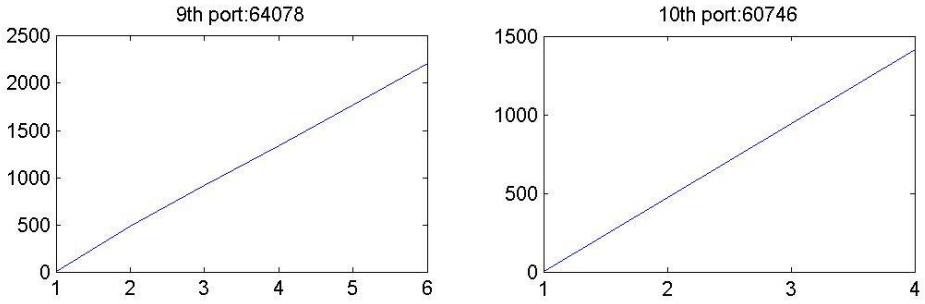


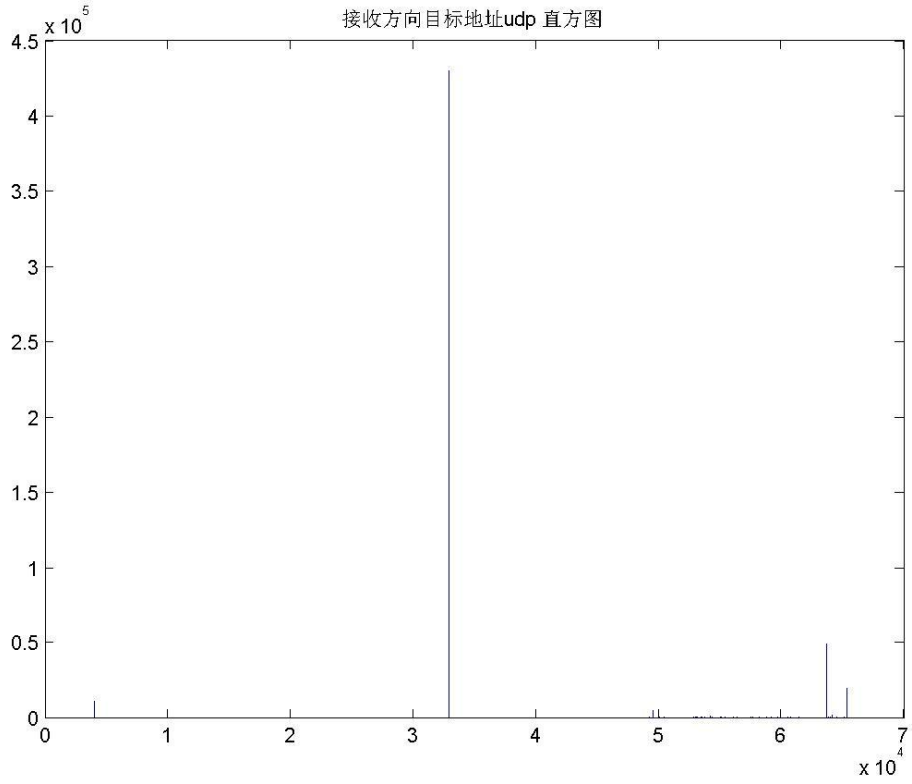


接收方向目标地址udp

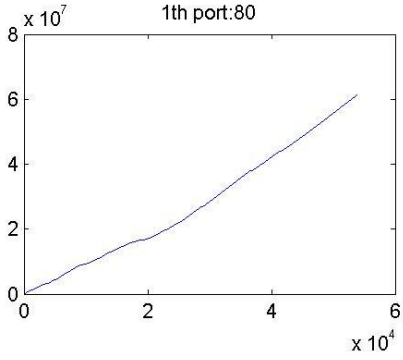


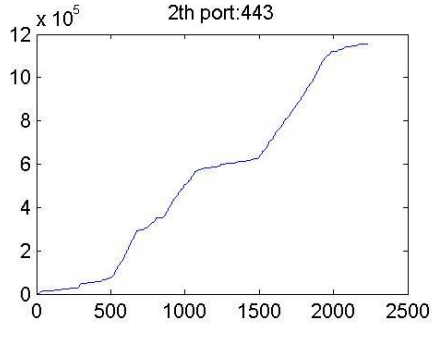
接收方向目标地址udp

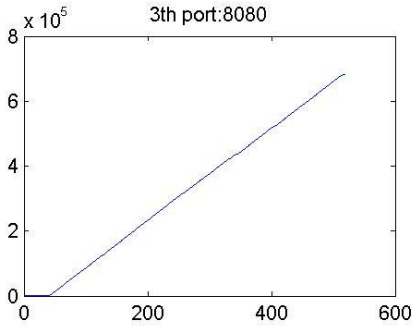


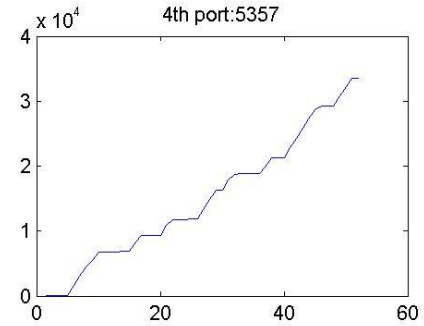

接收方向目标地址udp 直方图
x 10^5
4.5
4
3.5
3
2.5
2
1.5
1
0.5
0
0 1 2 3 4 5 6 7
x 10^4

接收方向原地址tcp

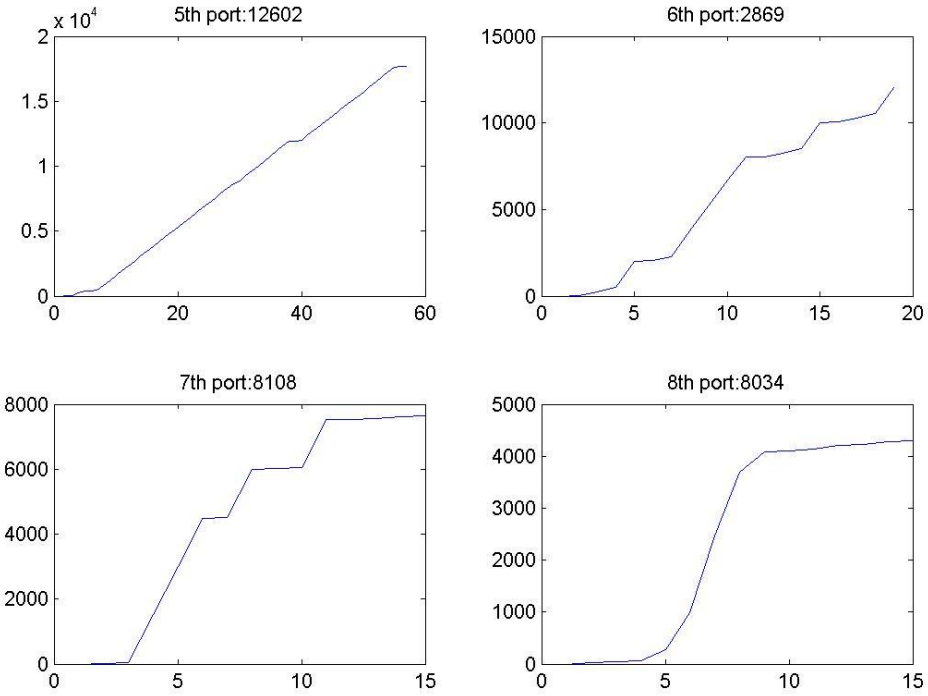

x 10^7
1th port:80
8
6
4
2
0
0 2 4 6
x 10^4


x 10^5
2th port:443
12
10
8
6
4
2
0
0 500 1000 1500 2000 2500

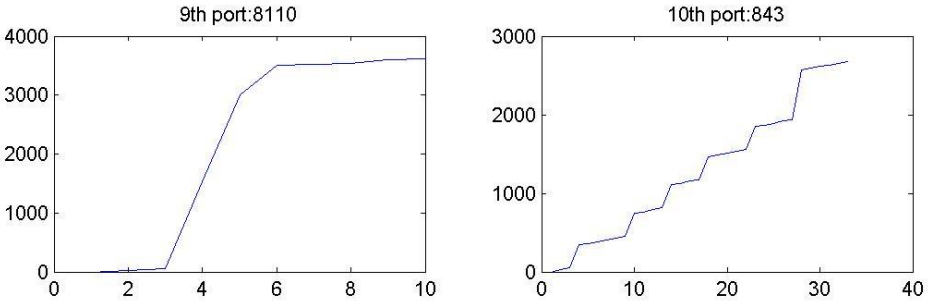

x 10^5
3th port:8080
8
6
4
2
0
0 200 400 600

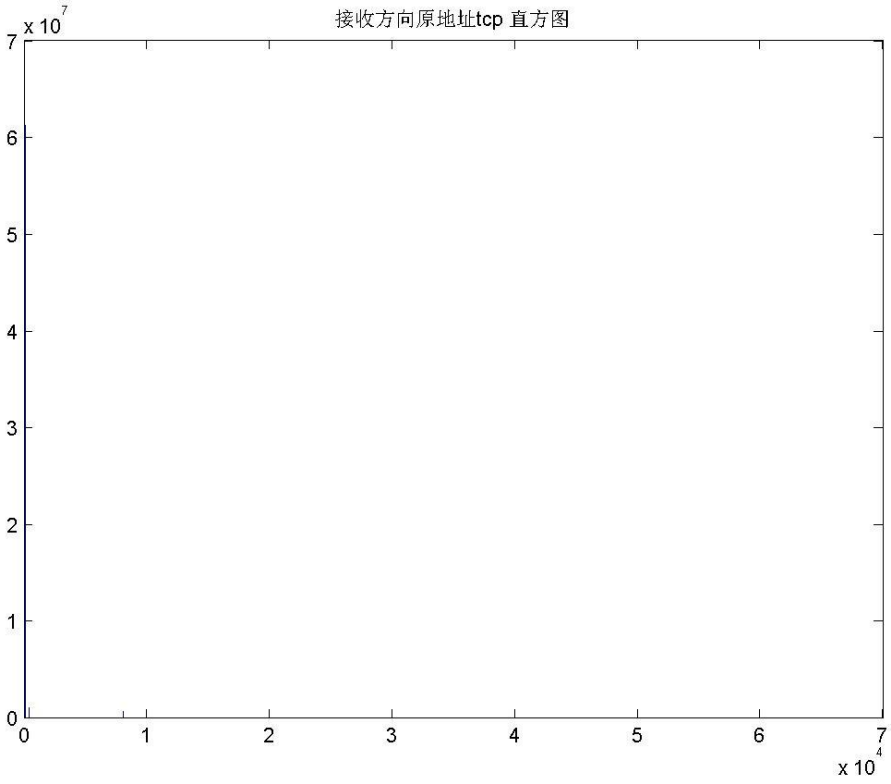

x 10^4
4th port:5357
4
3
2
1
0
0 20 40 60

接收方向原地址tcp

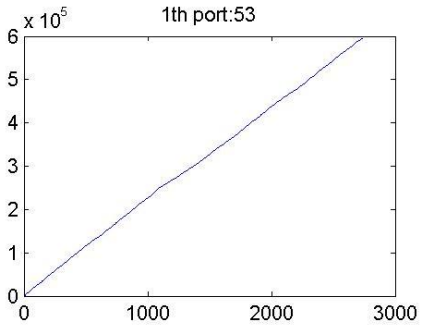


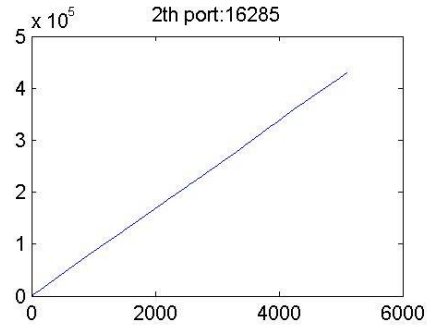
接收方向原地址tcp

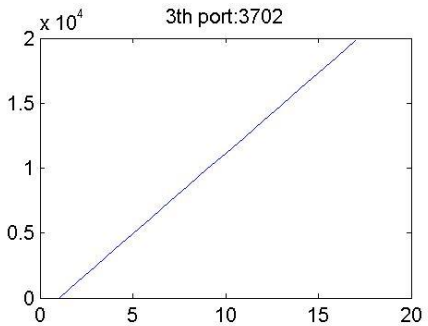


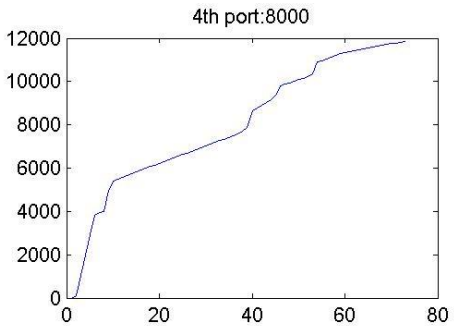

接收方向原地址tcp 直方图
x 10^7
7
6
5
4
3
2
1
0
0
1
2
3
4
5
6
7
x 10^4

接收方向原地址udp

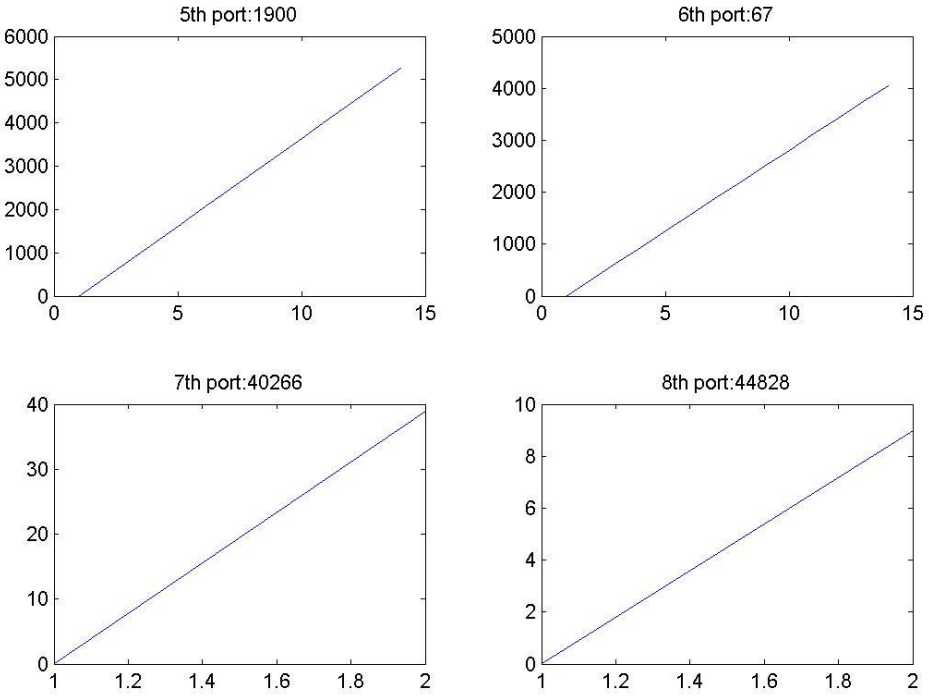

1th port:53
x 10^5
6
5
4
3
2
1
0
0
1000
2000
3000

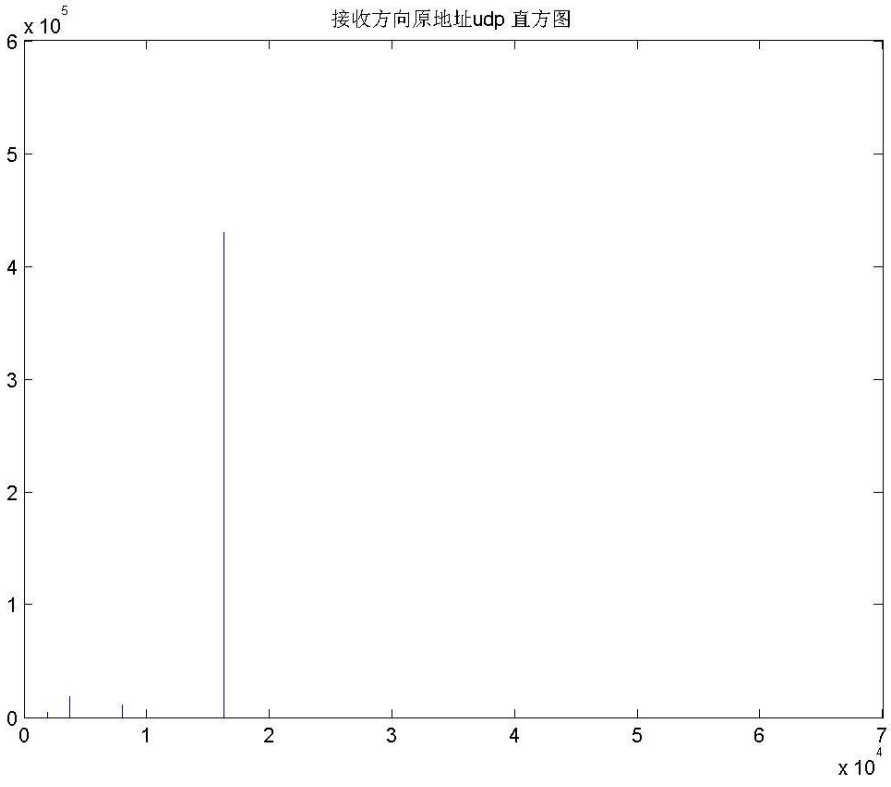

2th port:16285
x 10^5
5
4
3
2
1
0
0
2000
4000
6000


3th port:3702
x 10^4
2
1.5
1
0.5
0
0
5
10
15
20


4th port:8000
12000
10000
8000
6000
4000
2000
0
0
20
40
60
80

接收方向原地址udp


5th port:1900
6th port:67
7th port:40266
8th port:44828


接收方向原地址udp 直方图
x 10^5
x 10^4

2.5 对于载荷为 TCP 的报文，给出其中各个控制位出现的百分比。

收发双方向

URG： 0.0000005%

ACK： 83.0804255%

PSH： 7.2027765%

RST： 1.4142985%

SYN： 4.0689845%

FIN： 4.2335175%

收方向

URG： 0.0000005%

ACK： 84.2576675%

PSH： 7.5960355%

RST： 0.7192315%

SYN： 3.8443715%

FIN： 3.5826975%

发方向

URG： 0.0000005%

ACK： 81.5190885%

PSH： 6.6812105%

RST： 2.3361435%

SYN： 4.3668825%

FIN： 5.0966785%

3 分析与结论

3.1 协议载荷关系的讨论

按分组数进行统计，在发出的分组中 TCP 占到 68%，在收到的分组中则占 74%左右。按数据量进行统计，发出的数据中 TCP 占到了 57%，收到的数据中 TCP 占到了 85%。

首先 TCP 协议的占比均非常高的原因在于测试环境中主机的活动仅包含浏览网页、发送消息、看视频等等，而这些网络活动大部分使用的均是 TCP 协议。对于看视频，可能会使用 TCP 协议，也可能会使用 UDP 协议，而从本次

实验看来，现阶段的视频网站的传输应该使用的绝大多数是 TCP 协议。其次从数据量这个角度观察，发出的包含 TCP 协议的数据远小于收到的，这是由于在应用中发送和接收数据的不对称所致。例如，使用 HTTP 服务时，发送数据一般是对内容的请求，而接收数据才是真正的内容，因此接收的数据量会明显大于发送的数据量。

另外，在本次实验中，还进行了请求视频聊天的操作，因为仅仅是请求视频聊天，并未进行真正的视频聊天，因此对于视频聊天所使用的 UDP 协议的发送和接收数据量也存在着明显的不一致性。

最后，在全部的抓包过程中，还存在着大量 IPv6 协议，经过查询，大部分的 IPv6 协议均产生于看视频阶段。

3.2 IP 数据报分段的讨论

本次实验收集到的 IP 数据报均未被分片，一方面可能由于确实不存在过长的数据报，因此也就没有分段。另一方面也可能由于我们使用的抓包软件已经将分段的数据报整合了，因此在我们统计的过程中已经不存在分过段的数据报了。

3.3 数据报长度累计分布分析

本次实验网络活动如下：浏览网页 0-4 看视频 4-6 视频聊天 6-8 传送文件 9-10 无操作 10-12 直播 12-15，活动后面的数字代表时间点。虽然累计分布曲线的横坐标是数据报个数而不是时间，因此并不能对时间进行详细分析，但通过分析数据报长度累计分布，依然可以找到几个特征变化段并进行分析。

1.在收发两个方向 ip 总数据报长度的累计分布曲线中，都可以看到在中部靠后的一些数据报的长度累积速度明显放缓，原因是此处所产生的无操作活动 10-12 所导致，之所以发生在中部的位置而不是按照时间顺序发生在 2/3 的位置，原因是由于 12-15 的看直播活动中有大量的数据报传输，导致了后 3 分钟的数据报量较大，把无操作过程的数据报挤到了中部位置。

2.从整体上看，tcp、tcp+udp 的数据报都呈现出收的数据量大于发的数据量的特点，这与一个普通计算机用户的使用行为相符，而 udp 则相反，发大于收，且有明显的阶梯，主要是进行了微信视频聊天活动所导致。

3.为分析 udp 为何发大于收，将在微信视频聊天活动阶段的两个用户的 udp 包抓出分析可以看到，用户之间并不是直接通信，而是将视频信息发送给腾讯位于深圳的某服务器的两个不同端口，再转发到用户手上。结合视频聊天的清晰度极低的特点可以得出两种可能：1.该服务器并不只是简单的转发，而是在服务器端对收到的视频进行了一定的压缩才进行转发，以达到减轻发送视

频用户端客户端运行压力以及保证接收端用户视频流畅的效果。2.发大于收的原因主要来自于 UDP 的不可靠性使得大量包的丢失。

4.UDP 平均长度收发都在 100 字节左右，TCP 为 1000 字节左右，这与除了视频聊天外 udp 大部分都是用于 dns 等服务有关。

3.4 端口数据报长度累计分布分析

整理端口数据可得如下结果（截取数量按第一名数量级截取）：

Tcp 发送方向目标地址前 4 名 80,443,8080,5357

Tcp 发送方向原地址前 10 名
54575,54801,54517,52864,54697,53966,53353,54085,53150,54287

Tcp 接收方向目标地址前 10 名：54801,54517,53353,52864，
54697,54702,54478,54473，54459,54475

Tcp 接收方向原地址前 6 名：80,443,8080,5357,12602,2869

Udp 发送方向原地址前 4 名 32868,63619,137,49496

Udp 发送方向目标地址前 7 名 16285,53,137,1900,5355,8000,67

Udp 接收方向目标地址前 5 名：32868,63619,65309,63623,4007

Udp 接收方向原地址前 4 名：53,16285,3702,8000

从整体上观察可以看到，接收方向的数据量较大，端口数量分布较广，且 udp 的数据报相对于 tcp 显著集中于几个端口中。另外，本机所接收和发送所用到的端口显著不如外网接收和发送的端口集中。

对于一些端口的数据报进行分析：

1、对于 udp 分析如下：

1) 出现最多的端口为 53 端口，即 DNS 服务器所开放的端口。另外通过分析包的目标地址可以看出，63623、63619、65309 为本机与学校 DNS166.111.8.28 通信的端口，由学校 DNS53 端口接收数据报。可以看到 DNS 对本机发送的包数量仅次于视频聊天所带来的包数量，可见在上网过程中 DNS 服务的重要性。

2) 以及另一个出现较多的 137 端口，用于在局域网中提供计算机的名字或 IP 地址查询服务。分析可以看到两端地址分别为本机地址 101.5.239.240 和子网掩码地址 101.5.239.255。

3) 还有 8000，4007 端口，经过分析发现为 oicq 服务，用于支持腾讯的相关服务。

4) 此外，还有两个数据量极大的 32868 与 16285 端口，在百度中并不能搜索到相关信息，通过提取相应的包可以发现，对方地址为腾讯在深圳某服务器，可见是微信视频聊天的相关端口。本机通过 32868 端口给腾讯服务器 16285 发送了 8469 个包共 2330kb 的数据，接收了 5100 个包共 603kb 的数据，这与上面所分析的上下行数据不对称相符。

2、对于 tcp 分析如下：

1) 80 端口：即为上网过程中超文本传输协议所开放的端口，查询数据量前四的 ip 所在地址，分别为辽宁省大连市 联通、北京市海淀区 合一信息技术北京有限公司（优酷） BGP 多线、甘肃省兰州市 教育网、重庆市 联通，而上网过程中则浏览了优酷、新浪等网页，由此猜测除了数据量排名第二的优酷外，这些网站都在不同位置放置了自己的镜像。

2) 可以看到，tcp 中有大量的 5 开头，五位数的端口，例如总数据量第二的 54801 为上述甘肃兰州教育网发往本机的端口，另外通过筛选该 ip 的包可以看到，该地址通过 80 端口向本机多个 54 开头的端口发送了大量消息，由此可见该类 ip 应该都是本机为支持超文本传输协议所开放端口。

3) 接着查询了 Tcp 发送方向目标地址前 4 名中的 443,8080,5357，发现均是用来支持 http 服务或是加密的 https 服务，且基本集中于这四个端口，不像本机的接收端口，数据分布在很多的 5 开头的五位数端口上。但是 https 服务的数量显著少于 http 服务的数量，可见在网页通信中，大部分数据主要还是通过 http 协议传输，而只有少部分重要信息需要通过 https 进行加密传输。

4) 此外，还有一些不知作用的端口，例如对方地址坐标为华中科技大学的 12602 端口也向本机发送了 22kb 的数据，情况不明。

3.5 控制位分布分析

在本次实验中并没有发生 URG 位被置于 1 的情况，这是由于在日常使用中通常不会发生需要紧急中断的情况。

收发两个方向上，ACK 的比例都十分高，说明传输过程中大部分都是 ACK 分组，是建立在握手机制上的信息传输。

只有百分 7 左右的 PSH 被置为一，说明只有少数数据被要求直接递交，可见浏览网页、看视频、看直播等动作大部分数据都不要求效率的极致。

收到的数据报 RST 置 1 的情况大于发送的数据报 RST 为 1 的情况，且仅为百分二，说明网络情况良好，且本地网络状况优于对方。

SYN 置 1 的比例在收发中都维持在百分四左右，但发大于收，可见并没有所有的握手的成功建立。

发方向的 FIN 比收方向的 FIN 置一的比例高了 1.5 个百分点，可见在本次实验中本地要求中断连接的情况比对方要求中断连接的情况更多，可能是由于中间的文件传输活动所导致。

4　工作分配

董凯杰：数据采集，初步整理，利用 Wireshark 初步分析结果，问题 1-2 的代码编写以及相应部分的实验报告撰写。

刘晨：问题 3-5 的代码编写，图表绘制，以及相应部分的实验报告撰写。

5　源文件及源代码

见文件夹中相关文件。