中国科学技术大学计算机学院

计算机网络实验报告

实验三

T C P

学　　号：PB17081504

姓　　名：廖洲洲

专　　业：计算机科学与技术

指导老师：张信明

中国科学技术大学计算机学院

2019年12月4日

一、 实验目的

1、捕获从计算机到远程服务器的大量TCP传输

2、根据获得的跟踪结果对TCP传输机制作一些必要的分析，加深对TCP协议的理解

二、 实验原理

1、Wireshark（前称Ethereal）是一个网络封包分析软件。网络封包分析软件的功能是抓取网络封包，并尽可能显示出最为详细的网络封包资料。 Wireshark使用WinPCAP作为接口，直接与网卡进行数据报文交换，监听共享网络上传送的数据包，但不能对其进行修改或者控制。

2、TCP是因特网运输层的面向连接的可靠的运输协议。

三、 实验环境

1、 硬件

Lenovo XiaoXin CHAO7000
Intel® Core ™ i7-8500U CPU 1.80GHZ
内存：8G
显卡：NVIDIA GEFORCE 940MX

2、 软件

wireshark

四、 实验过程

1）实验步骤

a） 下载alice.txt

b） 访问 http://gaia.cs.umass.edu/wireshark-labs/TCP-wireshark-file1.html

c） 捕获从计算机到远程服务器的大量TCP传输

2) 实验分析

一、 在我得到的跟踪文件中：

1) 客户端(你的计算机)的IP地址和TCP端口号？

```
114.214.254.247    128.119.245.12    TCP    1514 58342 → 80 [ACK]
114.214.254.247    128.119.245.12    TCP    1514 58342 → 80 [ACK]
114.214.254.247    128.119.245.12    TCP    1514 58342 → 80 [ACK]
```

IP地址：114.214.254.247

TCP端口号：58342

2) 服务器端的IP地址和TCP端口号？

同由上图：

IP地址：128.119.245.12

TCP端口号:80

二、 在下载好的捕获报文中：

3) 客户端(你的计算机)的IP地址和TCP端口号？

```
192.168.1.102    128.119.245.12    TCP    62 1161 → 80 [SYN]
128.119.245.12   192.168.1.102     TCP    62 80 → 1161 [SYN,
192.168.1.102    128.119.245.12    TCP    54 1161 → 80 [ACK]
```

IP地址：192.168.1.102

TCP端口号：1161

4) SYN segment：

a) 用于初始化TCP连接的序列号是多少？

```
1161 → 80 [SYN] Seq=0 Win=16384 Len=0 MSS=1460 SACK_PERM=1
80 → 1161 [SYN, ACK] Seq=0 Ack=1 Win=5840 Len=0 MSS=1460 SACK_PERM=1
1161 → 80 [ACK] Seq=1 Ack=1 Win=17520 Len=0
```

序号为0

b) Segment内哪部分决定了该segment是一个SYN segment？

```
Flags: 0x002 (SYN)
   000. .... .... = Reserved: Not set
   ...0 .... .... = Nonce: Not set
   .... 0... .... = Congestion Window Reduced (CWR): Not set
   .... .0.. .... = ECN-Echo: Not set
   .... ..0. .... = Urgent: Not set
   .... ...0 .... = Acknowledgment: Not set
   .... .... 0... = Push: Not set
   .... .... .0.. = Reset: Not set
 > .... .... ..1. = Syn: Set
   .... .... ...0 = Fin: Not set
   [TCP Flags: ··········S·]
```

TCP报文段中有6比特的标志字段，其中的SYN比特被置1时说明该报文是SYN报文。

5) SYNACK segment(服务器→客户端)

a) 序列号是多少？

```
128.119.245.12      192.168.1.102       TCP      62 80 → 1161 [SYN, ACK] Seq=0 Ack=1
192.168.1.102       128.119.245.12      TCP      54 1161 → 80 [ACK] Seq=1 Ack=1 Win=1
```

序列号是0

b) ACK的值是？服务器是如何决定该值的？

ACK是1，服务器填充进报文段的确认号是服务器期望从客户端收到的下一字节的序号。

c) Segment内哪部分决定了该segment是一个SYNACK segment？

```
Flags: 0x012 (SYN, ACK)
   000. .... .... = Reserved: Not set
   ...0 .... .... = Nonce: Not set
   .... 0... .... = Congestion Window Reduced (CWR): Not set
   .... .0.. .... = ECN-Echo: Not set
   .... ..0. .... = Urgent: Not set
   .... ...1 .... = Acknowledgment: Set
   .... .... 0... = Push: Not set
   .... .... .0.. = Reset: Not set
 > .... .... ..1. = Syn: Set
   .... .... ...0 = Fin: Not set
   [TCP Flags: ·····A··S·]
```

TCP报文段中有6比特的标志字段，其中的ACK和SYN比特都被置1时说明该报文是SYNACK报文

6) 包含HTTP POST命令的TCP segment的序号是？

```
Sequence number: 1    (relative sequence number)
[Next sequence number: 566    (relative sequence number)]
Acknowledgment number: 1    (relative ack number)
0101 .... = Header Length: 20 bytes (5)
Flags: 0x018 (PSH, ACK)
  000. .... .... = Reserved: Not set
  ...0 .... .... = Nonce: Not set
  .... 0... .... = Congestion Window Reduced (CWR): Not set
```

```
0030  44 70 1f bd 00 00 50 4f  53 54 20 2f 65 74 68 65   Dp····PO ST /ethe
0040  72 65 61 6c 2d 6c 61 62  73 2f 6c 61 62 33 2d 31   real-lab s/lab3-1
0050  2d 72 65 70 6c 79 2e 68  74 6d 20 48 54 54 50 2f   -reply.h tm HTTP/
0060  31 2e 31 0d 0a 48 6f 73  74 3a 20 67 61 69 61 2e   1.1··Hos t: gaia.
0070  63 73 2e 75 6d 61 73 73  2e 65 64 75 0d 0a 55 73   cs.umass .edu··Us
0080  65 72 2d 41 67 65 6e 74  3a 20 4d 6f 7a 69 6c 6c   er-Agent : Mozill
0090  61 2f 35 2e 30 20 28 57  69 6e 64 6f 77 73 3b 20   a/5.0 (W indows;
00a0  55 3b 20 57 69 6e 64 6f  77 73 20 4e 54 20 35 2e   U; Windo ws NT 5.
```

序号是1

7) 把包含HTTP POST命令的TCP segment作为第一个segment：

a) 那么最开始的6个segment（包含包含HTTP POST命令的TCP segment）的序列号分别是多少？

```
 619 1161 → 80 [PSH, ACK] Seq=1 Ack=1 Win=17520 Len=565
1514 1161 → 80 [PSH, ACK] Seq=566 Ack=1 Win=17520 Len=1460
1514 1161 → 80 [ACK] Seq=2026 Ack=1 Win=17520 Len=1460
1514 1161 → 80 [ACK] Seq=3486 Ack=1 Win=17520 Len=1460
1514 1161 → 80 [ACK] Seq=4946 Ack=1 Win=17520 Len=1460
1514 1161 → 80 [ACK] Seq=6406 Ack=1 Win=17520 Len=1460
```

分别为：1，566，2026，3486，4946，6406

b) 6个segment分别是什么时间发送的？

```
0.026477  192.168.1.102  128.119.245.12  TCP  619 1161 → 80 [PSH, ACK] Seq=1
0.041737  192.168.1.102  128.119.245.12  TCP  1514 1161 → 80 [PSH, ACK] Seq=56
0.054026  192.168.1.102  128.119.245.12  TCP  1514 1161 → 80 [ACK] Seq=2026 Ac
0.054690  192.168.1.102  128.119.245.12  TCP  1514 1161 → 80 [ACK] Seq=3486 Ac
0.077405  192.168.1.102  128.119.245.12  TCP  1514 1161 → 80 [ACK] Seq=4946 Ac
0.078157  192.168.1.102  128.119.245.12  TCP  1514 1161 → 80 [ACK] Seq=6406 Ac
```

分别为：

0.026477, 0.041737, 0.054026, 0.054690, 0.077405, 0.078157

c) 每个segment对应的ACK分别是什么时间被接收的？

0.023172	128.119.245.12	192.168.1.102	TCP	62 80 → 1161 [SYN, ACK] Seq=0 Ack=1 Win=5840 Len=0
0.053937	128.119.245.12	192.168.1.102	TCP	60 80 → 1161 [ACK] Seq=1 Ack=566 Win=6780 Len=0
0.077294	128.119.245.12	192.168.1.102	TCP	60 80 → 1161 [ACK] Seq=1 Ack=2026 Win=8760 Len=0
0.124085	128.119.245.12	192.168.1.102	TCP	60 80 → 1161 [ACK] Seq=1 Ack=3486 Win=11680 Len=0
0.169118	128.119.245.12	192.168.1.102	TCP	60 80 → 1161 [ACK] Seq=1 Ack=4946 Win=14600 Len=0
0.217299	128.119.245.12	192.168.1.102	TCP	60 80 → 1161 [ACK] Seq=1 Ack=6406 Win=17520 Len=0
0.267802	128.119.245.12	192.168.1.102	TCP	60 80 → 1161 [ACK] Seq=1 Ack=7866 Win=20440 Len=0

这是服务器发向客户端的确认报文，故得到确认报文的接收时间为：0.053973,0.077294,0.124085,0.169118,0.217299,0.267802

d) 每个segment对应的RTT值是多少?

输入表格计算：（时间单位（s））

segment	序列号	发送时间	收到ACK时间	RTT
1	1	0.026477	0.053973	0.027496
2	566	0.041737	0.077294	0.035557
3	2026	0.054026	0.124085	0.070059
4	3486	0.05469	0.169118	0.114428
5	4946	0.077405	0.217299	0.139894
6	6406	0.078157	0.267802	0.189645

e) 当ACK被接收后， EstimatedRTT是多少？（注：假定对于第一个segment，RTT= EstimatedRTT ）

根据：

$$EstimatedRTT = (1 - \alpha) \cdot EstimatedRTT + \alpha \cdot SampleRTT$$

取α为0.125，计算得（时间单位（s））

segment	序列号	发送时间	收到ACK时间	RTT	EstimatedRTT
1	1	0.026477	0.053973	0.027496	0.027496
2	566	0.041737	0.077294	0.035557	0.028503625
3	2026	0.054026	0.124085	0.070059	0.033698047
4	3486	0.05469	0.169118	0.114428	0.043789291
5	4946	0.077405	0.217299	0.139894	0.05580238
6	6406	0.078157	0.267802	0.189645	0.072532707

8) 6个segment的长度分别是多少？

```
  619 1161 → 80 [PSH, ACK] Seq=1 Ack=1 Win=17520 Len=565
 1514 1161 → 80 [PSH, ACK] Seq=566 Ack=1 Win=17520 Len=1460
 1514 1161 → 80 [ACK] Seq=2026 Ack=1 Win=17520 Len=1460
 1514 1161 → 80 [ACK] Seq=3486 Ack=1 Win=17520 Len=1460
 1514 1161 → 80 [ACK] Seq=4946 Ack=1 Win=17520 Len=1460
 1514 1161 → 80 [ACK] Seq=6406 Ack=1 Win=17520 Len=1460
```

565，1460，1460，1460，1460，1460

9) 在整个追踪过程中，可利用的最小的缓存空间是多少？接收方缓存空间的缺乏会限制发送方吗？

```
    .... .... 0... = Push: Not set
    .... .... .0.. = Reset: Not set
  > .... .... ..1. = Syn: Set
    .... .... ...0 = Fin: Not set
    [TCP Flags: ·····A··S·]
  Window size value: 5840
  [Calculated window size: 5840]
  Checksum: 0x774d [unverified]
  [Checksum Status: Unverified]
  Urgent pointer: 0
```

可利用最小的缓冲空间（Win）为5840字节。服务器端的Win在慢慢变大，最后变到62780字节，因此没有限制发送端。

最后的窗口大小：

```
    .... .... ..0. = Syn: Not set
    .... .... ...0 = Fin: Not set
    [TCP Flags: ·····A····]
  Window size value: 62780
  [Calculated window size: 62780]
  [Window size scaling factor: -2 (no window scaling used)]
  Checksum: 0x44a8 [unverified]
  [Checksum Status: Unverified]
  Urgent pointer: 0
```

10)追踪文件中有重传的segment吗？你需要检查什么可以找到重传的segment？

没有重传的报文。检查发送端是否发送过两个具有相同Seq序号的报文，若发送方发送过相同序号的报文，即有重传，即检查Seq序号是否严格

单调递增即可。

11)接收者在一个ACK中通常确认多少数据？你能识别出接收方正在压缩其他接收段的情况吗？

通常确认1460字节。随机取一端进行计算

```
Ack=2026
Ack=3486
Ack=4946
Ack=6406
Ack=7866
Ack=9013
Ack=10473
Ack=11933
Ack=13393
```

ACK号	确认字节数
2026	
3486	1460
4946	1460
6406	1460
7866	1460
9013	1147
10473	1460
11933	1460
13393	1460

有，比如下下面的第52个确认报文确认了2352个字节

```
60 80 → 1161 [ACK] Seq=1 Ack=31237
60 80 → 1161 [ACK] Seq=1 Ack=33589
```

12)该TCP连接的吞吐量是多少（单位：bytes/s），并对计算方法加以说明。

第一次发送post报文段到最后一个ACK报文所花费时间为

```
0.026477    192.168.1.102    128.119.245.12    TCP    619 1161 → 80 [PSH, ACK] Seq=1 Ack=1 Win=17520 Len=565
0.041737    192.168.1.102    128.119.245.12    TCP    1514 1161 → 80 [PSH, ACK] Seq=566 Ack=1 Win=17520 Len=1460
202 5.455830    128.119.245.12    192.168.1.102    TCP    60 80 → 1161 [ACK] Seq=1 Ack=164091 Win=62780 Len=0
203 5.461175    128.119.245.12    192.168.1.102    TCP    784 80 → 1161 [PSH, ACK] Seq=1 Ack=164091 Win=62780 Len=730
```

5.455838-0.026477=5.4293(s)

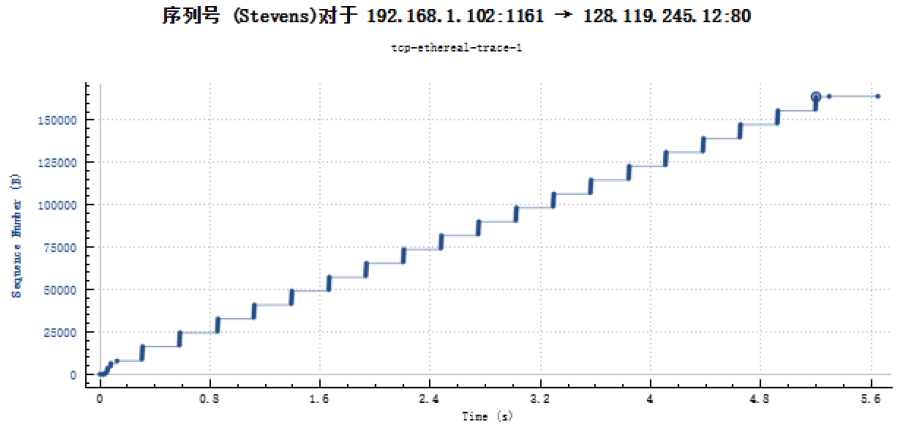
```
[ACK] Seq=1 Ack=164041 Win=62780 Len=0
[ACK] Seq=1 Ack=164091 Win=62780 Len=0
[PSH, ACK] Seq=1 Ack=164091 Win=62780 Len=730
```

总共发送164091-1=164090(bytes)

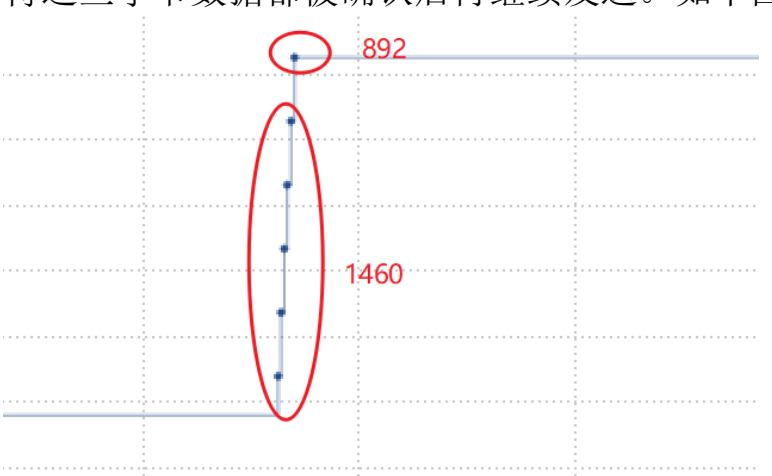
故吞吐量为164090/5.4293=30223（bytes/s）

13)使用时间序列图(Stevens)绘图工具查看从客户端发送到gaya .cs.umass.edu服务器的片段的序列号和时间图。您能确定TCP的

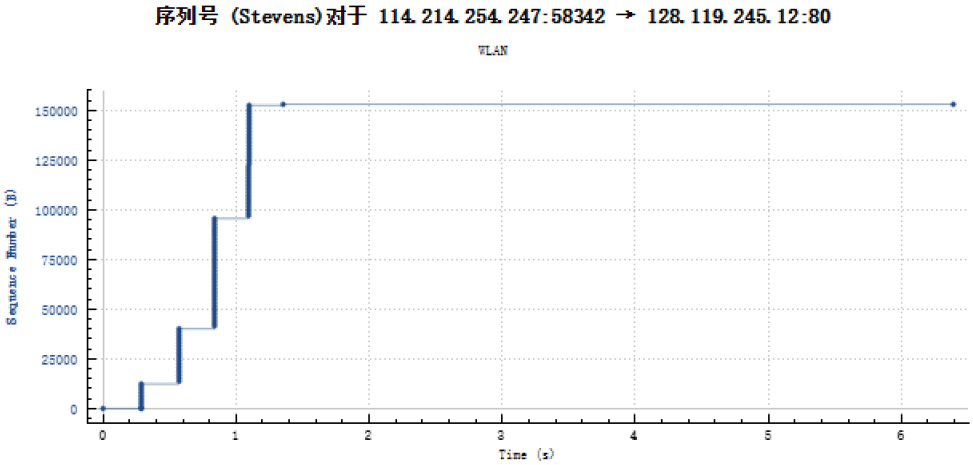
slowstart阶段在哪里开始和结束，以及在哪里避免拥塞吗?对测量数据与我们在文中研究的理想化TCP行为的不同之处进行评论。



慢启动在开始发送post报文时开始了。TCP的slowstart阶段的结束以及拥塞避免阶段的观察需要知道拥塞窗口的变化，着从时间序列图中不能直接观测到。但是，我们从图中发现，发送序列号是一直递增的，因此我们推测到TCP并没有发生丢包事件，因此不会进入拥塞避免阶段。
不同处：
在理想化TCP行为中，TCP开始时进入慢启动阶段，发生超时指示的丢包事件后，cwnd设置为1并且重新启动慢启动阶段，将ssthresh设置为cwnd/2。若cwnd=ssthresh，则结束慢启动转移到拥塞避免阶段。但是我们本次实验中没有发生丢包事件，因此一直处于慢启动阶段，没有没有进入拥塞避免和快速重传。同时我们的拥塞窗口一直固定为同一值，只要连续发送1460*5+892*1字节=8192字节的包就会暂停发送，等待这些字节数据都被确认后再继续发送。如下图



14)根据我的的跟踪文件回答问题13th.



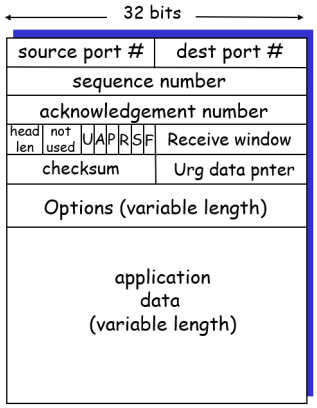
同样，慢启动在开始发送post报文时开始了。序列号一直递增，没有发生过丢包事件，不会进入拥塞避免阶段。

不同之处：我的一直处于慢启动阶段，并且拥塞窗口以指数的形式递增，并且再慢启动结束前就完成了所有数据的发送。

五、 实验总结

1. 本次实验我捕获从了计算机到远程服务器的大量TCP传输，根据获得的跟踪结果对TCP传输机制作了一些必要的分析，加深对了TCP协议的理解

2. 加深了对TCP报文结构的了解



3. 对TCP的三次握手机制、可靠性传输、拥塞控制进行了更加贴合实际的深入学习。

三次握手过程为