**实验6 传输层实验**

**传输层实验发送方初始序号（sequence number）记录表，请填写：**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **实验名称** | **实验班号** | **学号** | **姓名** | **发送方初始序号** |
| TCP协议基本分析 | 周五上午 | ZY1906812 | 李泽霖 | 4272716823 |
| 滑动窗口机制和窗口侦查机制分析 | 周五上午 | ZY1906812 | 李泽霖 | 326667532 |
| 慢启动、拥塞避免及拥塞处理和超时重传机制实验 | 周五上午 | ZY1906812 | 李泽霖 | 3150675336  468841012 |
| 快重传和快恢复实验 | 周五上午 | ZY1906812 | 李泽霖 | 1945360164 |
| 糊涂窗口综合症和Nagle算法分析实验 | 周五上午 | ZY1906812 | 李泽霖 | 1219823660  3988224377 |

1、根据2.6中步骤3回答：TCP的连接和建立采用的是：三次握手 方式，PCA是 主动打开方(C) ，PCB是 被动打开方(S) 。先点击发送再点击接收，会出现什么问题？为什么？

**答：**

**先点击发送再点击接受会导致连接失败，而且如果没有应用进程被动打开，那么主动打开的应用进程无法建立连接。**

**因为C/S模式下，若Server没有启动，那么Client就连接不上。服务器和客户端的结合需要看TCP状态机，必须存在被动打开方才能够连接。**

2、根据2.6中步骤5，结合预习报告，分析TCP连接的建立过程，根据TCP建立过程的三个报文，先填写下表：

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名称 | 第一条报文 | 第二条报文 | 第三条报文 |
| 报文序号 | 3 | 4 | 5 |
| Sequence Number | 4272716823 | 4276788842 | 4272716824 |
| Acknowledgement Number | 0 | 4272716824 | 4276788843 |
| Ack | 0 | 1 | 1 |
| Syn | 1 | 1 | 0 |

3、根据2.6中步骤6回答：

TCP连接建立时，其报文首部与其它TCP报文不同，有一个“Option”字段，它的作用是什么，值为多少？结合IEEE802.3协议规定的以太网最大帧长度分析此数据是怎样得出的。

答：

**Option字段的值中包含一个最大报文段长度（Maximum segment size，MSS），取C/S两方承载的MSS中取较小的值。MSS应用于数据传送阶段，在本实验中得到的MSS值是1460 字节。**

**IEEE802.3协议规定的以太网最大帧长度为1518字节。**

**MSS＝以太网最大帧长度－以太网帧头和帧尾（18）－IP首部固定长度（20）－TCP首部固定长度（20）＝1518－18－20－20＝1460**

4、根据2.6中步骤7：结合预习报告，分析TCP连接的释放过程，选择TCP连接撤消的四个报文，将报文信息填入下表。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 字段名称 | 第一条报文 | 第二条报文 | 第三条报文 | 第四条报文 |
| 报文序号 | 529 | 533 | 537 | 538 |
| Sequence Number | 4273072164 | 4276788843 | 4276788843 | 4273072735 |
| Acknowledgement Number | 4276788843 | 4273072735 | 4273072735 | 4276788844 |
| Ack | 1 | 1 | 1 | 1 |
| Fin | 1 | 0 | 1 | 0 |

5、根据2.6中步骤8：分析TCP数据传送阶段的前8个报文，将报文信息填入下表。

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 报文序号 | 报文种类  (发送/确认) | 序号字段 | 确认号字段 | 数据长度 | 被确认报文序号 | 窗口 |
| 6 | 发送 | 4272716824 | 4276788843 | 1400 |  | 5840 |
| 7 | 发送 | 4272718224 | 4276788843 | 1460 |  | 5840 |
| 8 | 发送 | 4272719684 | 4276788843 | 1460 |  | 5840 |
| 9 | 确认 | 4276788843 | 4272718224 | 0 | 6 | 8400 |
| 10 | 发送 | 4272721144 | 4276788843 | 1460 |  | 5840 |
| 11 | 发送 | 4272722604 | 4276788843 | 1460 |  | 5840 |
| 12 | 确认 | 4276788843 | 4272719684 | 0 | 7 | 11680 |
| 13 | 发送 | 4272724064 | 4276788843 | 1460 |  | 5840 |

请写出TCP数据部分长度的计算公式。数据传送阶段第一个报文的序号字段值是否等于连接建立时第三个报文的序号？

**答：**

**TCP数据部分长度 = IP总长度字段值－IP首部长度字段值×4－TCP首部长度字段值×4**

**等于。**

6. 根据3.6.1中“ 滑动窗口机制和窗口侦查机制分析”步骤6回答：

1. 分析数据发送部分的前几条报文，描述发送方发送窗口的变化，并解释为什么？

**答：**

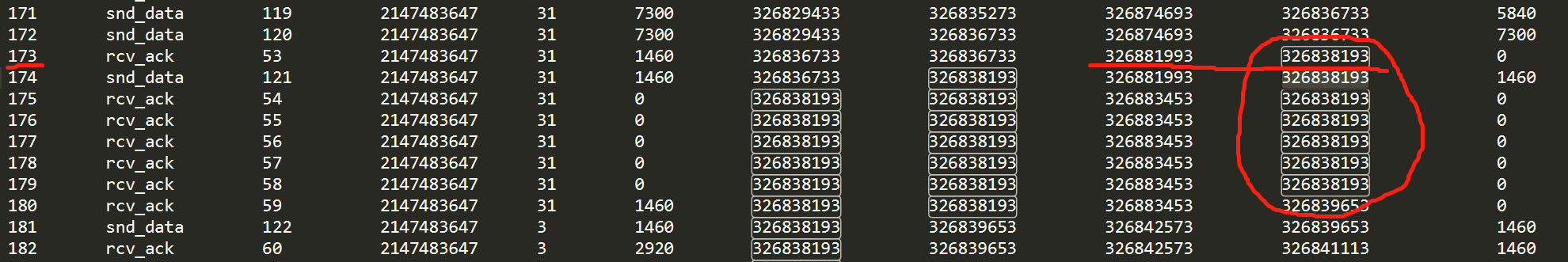
**接收方每一个ack中的窗口大小字段每次递增2920，即rwnd每次递增2920。**

**因为数据发送部分前几条报文时处于慢启动状态，每收到一个ack，发送方cwnd就会加1。**

**发送窗口=min[cwnd, rwnd]=cwnd，因此发送窗口的大小随着cwnd增长。**

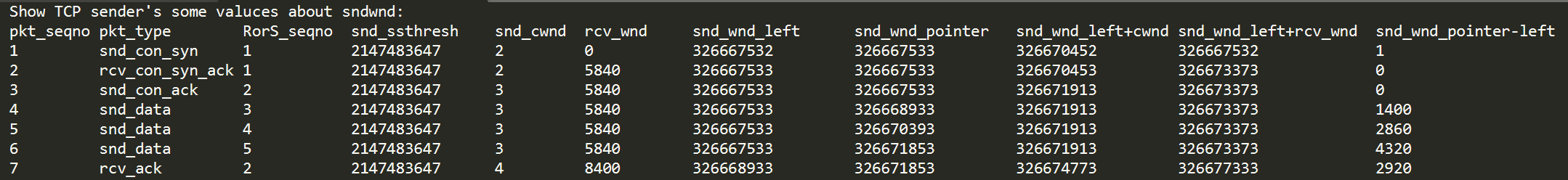
1. 指出从哪个序号的报文能够看出接收端开始休眠，并解释理由。

**答：**



**序号为173的报文接收端开始休眠。可以看到从序号173之后的报文，右边沿不再移动，表明接收方在窗口范围外的可用缓存被已确认的数据占据着，说明接收端的应用程序进程已经不再从缓存中读取数据了，即表明其已开始休眠。**

1. 分析文件send2-组座号-tcpsndwnddata.txt，选中三次握手连接建立后的前4条报文记录（3条DATA报文、1条ACK报文，序号为4、5、6、7），记下发送方发送窗口的相关值（rcv\_wnd , snd\_wnd\_left , snd\_wnd\_point , snd\_wnd\_left+cwnd , snd\_wnd\_left+rcv\_wnd , (snd\_wnd\_point- left)）。按下表分析计算接收方（及发送方）的窗口的相关值。



**4号报文（sender----data---->receiver）**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | rcv\_wnd | snd\_wnd\_left | snd\_wnd\_pointer | snd\_wnd\_left+cwnd和  snd\_wnd\_left+rcv\_wnd | snd\_wnd\_point- left |
| 发送方发出报文 | 5840 | 326667533 | 326668933 | 326671913  326673373 | 1400 |
| 发送窗口右边沿 | =min( snd\_wnd\_left+cwnd, snd\_wnd\_left+rcv\_wnd ) = 326671913 | | | | |
|  | 通告的接收窗口 | 接收窗口左边沿 | 接收窗口指针 | 接收窗口右边沿 | 在接收缓存中的数据量（即未确认的数据） |
| 接收方接到DATA前 | 5840 | 326667533 | 326667533 | 326673373 | 0 |
| 接收方接到DATA后 | 5840 | 326667533 | 326668933 | 326673373 | 1400 |

**5号报文（sender----data---->receiver）**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | rcv\_wnd | snd\_wnd\_left | snd\_wnd\_pointer | snd\_wnd\_left+cwnd和  snd\_wnd\_left+rcv\_wnd | snd\_wnd\_point- left |
| 发送方发出报文 | 5840 | 326667533 | 326670393 | 326671913  326673373 | 2860 |
| 发送窗口右边沿 | =min( snd\_wnd\_left+cwnd, snd\_wnd\_left+rcv\_wnd ) = 326671913 | | | | |
|  | 通告的接收窗口 | 接收窗口左边沿 | 接收窗口指针 | 接收窗口右边沿 | 在接收缓存中的数据量（即未确认的数据） |
| 接收方接到DATA前 | 5840 | 326667533 | 326668933 | 326673373 | 1400 |
| 接收方接到DATA后 | 5840 | 326667533 | 326670393 | 326673373 | 2860 |

**6号报文（sender----data---->receiver）**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | rcv\_wnd | snd\_wnd\_left | snd\_wnd\_pointer | snd\_wnd\_left+cwnd和  snd\_wnd\_left+rcv\_wnd | snd\_wnd\_point- left |
| 发送方发出报文 | 5840 | 326667533 | 326671853 | 326671913  326673373 | 4320 |
| 发送窗口右边沿 | =min( snd\_wnd\_left+cwnd, snd\_wnd\_left+rcv\_wnd ) = 326671913 | | | | |
|  | 通告的接收窗口 | 接收窗口左边沿 | 接收窗口指针 | 接收窗口右边沿 | 在接收缓存中的数据量（即未确认的数据） |
| 接收方接到DATA前 | 5840 | 326667533 | 326670393 | 326673373 | 2860 |
| 接收方接到DATA后 | 5840 | 326667533 | 326671853 | 326673373 | 4320 |

**7号报文（receiver ----ack----> sender）**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | 通告的接收窗口 | 接收窗口左边沿 | 接收窗口指针 | 接收窗口右边沿 | 在接收缓存中的数据量（即未确认的数据） |
| 接收方发出ACK | 8400 | 326668933 | 326671853 | 326677333 | 2920 |
|  | rcv\_wnd | snd\_wnd\_left | snd\_wnd\_pointer | snd\_wnd\_left+cwnd和  snd\_wnd\_left+rcv\_wnd | snd\_wnd\_point- left |
| 发送方接到ACK后 | 8400 | 326668933 | 326671853 | 326674773  326677333 | 2920 |
| 发送窗口右边沿 | =min( snd\_wnd\_left+cwnd, snd\_wnd\_left+rcv\_wnd ) = 326674773 | | | | |
| 发送方接到ACK前 | 5840 | 326667533 | 326671853 | 326671913  326673373 | 4320 |
| 发送窗口右边沿 | =min( snd\_wnd\_left+cwnd, snd\_wnd\_left+rcv\_wnd ) = 326671913 | | | | |

1. 根据文件send2-组座号-tcpsndwnddata.txt中发送方的发送窗口相关值进行分析，接收方开始休眠后，描述接收窗口的变化，指出窗口收缩、窗口合拢、窗口张开对应的开始报文序号，并记下send2-组座号-tcpsndwnddata.txt文件中的对应报文的数值记录（pkt\_seqno，pkt\_type，..，……）。

**答：**

**接收方开始休眠后，接收窗口的右边沿不再移动，而左边沿和指针不断右移，导致窗口合拢，此时接收窗口不断变小，直至成为零窗口。**

**接收方结束休眠后，应用程序进程重新开始读取缓存数据，接收窗口的右边沿向右移动，导致窗口张开，此时接收窗口重新变大，发送方重新开始发送数据。**

**【窗口收缩】：未出现**

**【窗口合拢】：**

pkt\_seqno 167

pkt\_type rcv\_ack

RorS\_seqno 52

snd\_ssthresh 2147483647

snd\_cwnd 31

rcv\_wnd 7300

snd\_wnd\_left 326829433

snd\_wnd\_pointer 326829433

snd\_wnd\_left+cwnd 326874693

snd\_wnd\_left+rcv\_wnd 326836733

snd\_wnd\_pointer-left 0

**【窗口张开】：**

pkt\_seqno 180

pkt\_type rcv\_ack

RorS\_seqno 59

snd\_ssthresh 2147483647

snd\_cwnd 31

rcv\_wnd 1460

snd\_wnd\_left 326838193

snd\_wnd\_pointer 326838193

snd\_wnd\_left+cwnd 326883453

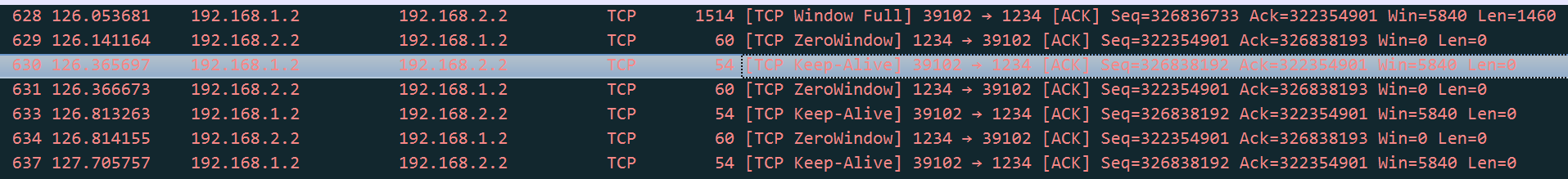
snd\_wnd\_left+rcv\_wnd 326839653

snd\_wnd\_pointer-left 0

7. 根据3.6.1中“ 滑动窗口机制和窗口侦查机制分析”步骤7回答：

写出窗口侦查开始的报文序号，窗口侦查报文数据长度、窗口侦查报文发送的时间规律。

**答：**



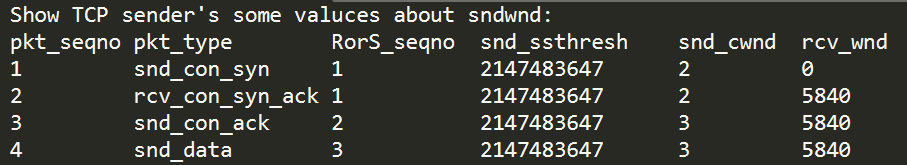
**开始的报文序号为630，报文数据长度为0，**

**时间规律：相邻的窗口侦查报文发送时间间隔翻倍增加。**

8. 根据3.6.2中“ 慢启动、拥塞避免及拥塞处理和超时与重传机制分析”步骤9回答：

1. 选中第一条发送数据的报文记录，记下其ssthresh和cwnd值是多少？为何为此值？按发送窗口的计算公式计算出当前的发送窗口snd\_wnd值。并记下此时的发送窗口左边沿snd\_wnd\_left值并计算出此时的发送窗口右边沿。

**答：**

****

**ssthresh = 2147483647，cwnd = 3；**

**此时发送端未发现网络拥塞，因此ssthresh未被设置。慢启动开始时，cwnd被设置为2，而在建立TCP连接时收到了一个ACK报文，因此cwnd就会加1，变成3。**

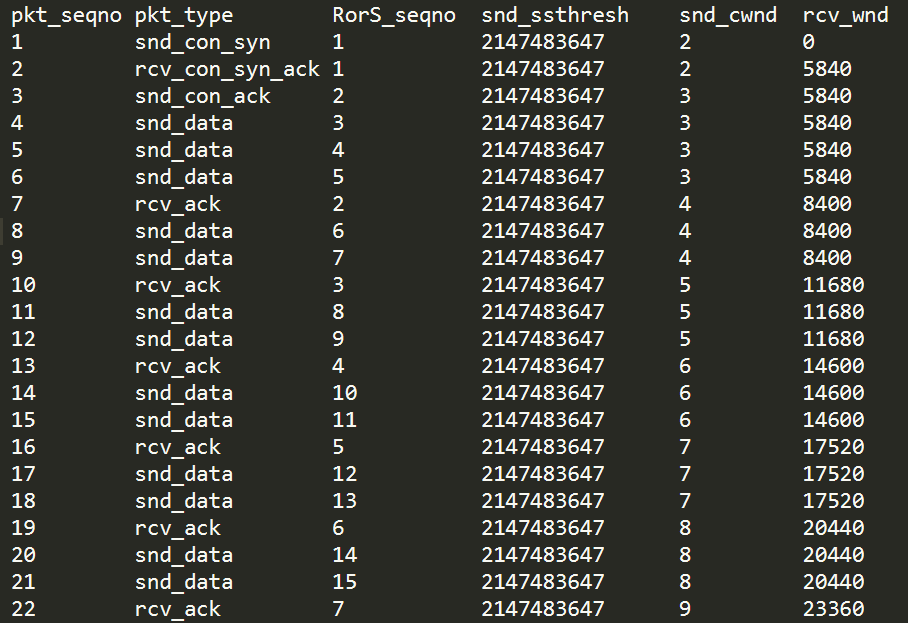
**snd\_wnd = min(snd\_cwnd, rcv\_wnd) = min(3×1460, 5840) = 4380**

**snd\_wnd\_left = 3150675337**

**snd\_wnd\_right = snd\_wnd\_left + snd\_wnd = 3150679717**

1. 在随后的发送数据报文中，ssthresh和cwnd值有何变化，呈何种规律？为什么呈此种规律？发送的报文是否可以验证这一规律？

**答：**



**Ssthresh一直不变，因为此时发送端未发现网络拥塞，因此ssthresh未被设置。**

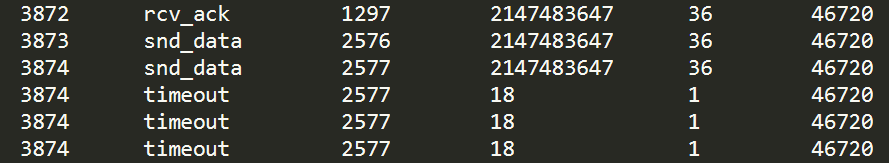
**此时处于慢启动状态，cwnd每收到一个ACK报文就会加1，一直到36后就不再增加。**

**发送的报文可以验证这一规律。**

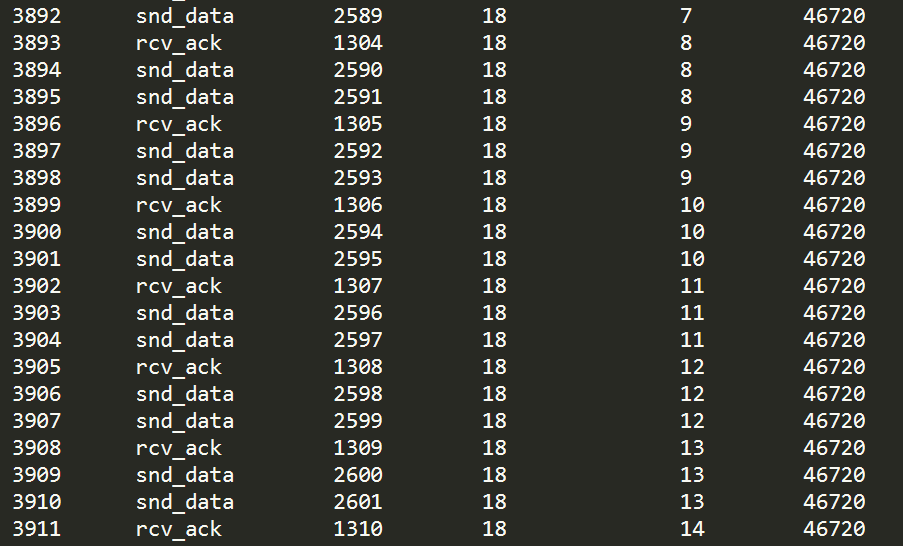
1. 指出ssthresh和cwnd值有突然变化的报文序号，为何会有这种变化？此后（直至结束）的ssthresh和cwnd值有何变化，呈何种规律？为什么？

**答：**

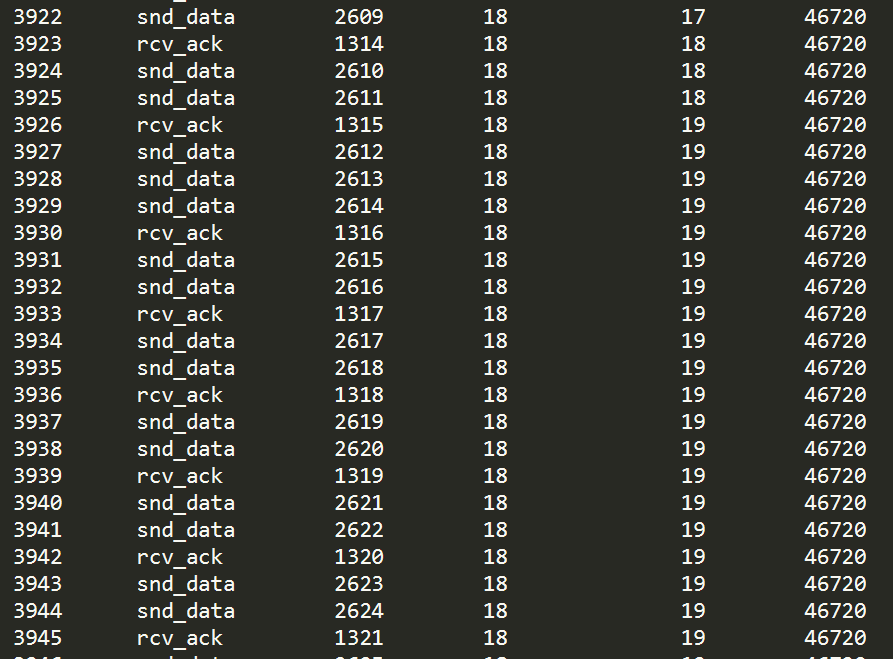
**ssthresh和cwnd值有突然变化的报文序号为3874；**



**因为路由器的e0/0端口断开，发送报文时设置的定时器超时，即发送端没有按时收到确认报文，发送端认为网络拥塞，ssthresh = max( cwnd/2, 2) = 18，并设置拥塞窗口cwnd = 1。**



**当cwnd ≤ ssthresh(18)时，使用慢启动算法，每收到一个ACK，cwnd就会加1.**



**当cwnd > ssthresh(18)时，使用拥塞避免算法，每收到cwnd个非重复的ACK报文，cwnd就会加1，再一直增加到36后就不再增加。**

9. 根据3.6.2中“ 慢启动、拥塞避免及拥塞处理和超时与重传机制分析”步骤10回答：

1. 对Wireshark截获的报文进行分析，分别记下TCP传输过程中发送第一条重传报文的序号，并请用截获报文的时间字段分别计算出在两种转发速率下第一条重传报文和其对应的原始发送报文的时间差、第二条重传报文和其对应的原始发送报文的时间差。分析同种速率下和两种转发速率下两次重传的时间差，结合题（2）对此现象做出解释。

|  |  |  |  |
| --- | --- | --- | --- |
|  | 第一个重传时间差 | 第二个重传时间差 | 第三个重传时间差 |
| 10Mbps | 0.229385 | 0.460059 | 0.919997 |
| 80Kbps | 4.164444 | 8.327595 | 16.652668 |

**答：**

**RTT = α × RTT\_old + (1－α) × RTT\_new α = 7/8**

**正常传输期间：RTO = β × RTT β > 1**

**超时重传期间：RTO =γ× RTO\_old γ= 2**

**同种速率下，每次重传的时间差为上一次的2倍；**

**两种转发速率下重传的时间差取决于各自的平均往返时延RTT，转发速率越高，平均往返时延越小，则重传时间差也越小。**

1. 对send3-组座号-tcprtodata.txt和send4-组座号-tcprtodata.txt文件进行分析。在正常传输时，两种速率下的TCP连接的RTO时间分别稳定在多少？在发生报文超时重传时，RTT和RTO值有何变化？特别是RTO值有何特定变化规律？为什么？继续正常报文传输时RTT和RTO值又有何变化？为什么？

**答：**

**10Mbps传输速率下，RTO时间稳定在230ms；**

**80Kbps传输速率下，RTO时间稳定在4163ms。**

**发生报文超时重传时，RTT不变，RTO双倍增大；因为RTO = γ × RTO\_old(γ= 2)，而在超时重传时没有收到ACK报文，因此RTT不变。**

**继续正常报文传输时RTT和RTO值会先变小然后重新趋于稳定；因为在超时重传期间，接收方把缓冲的数据给处理完了，刚开始能跟刚建立TCP连接时一样快速回复ACK报文。**

1. 课外：查阅相关资料，对RTT和RTO值的变化做出定量的计算和分析。建议阅读《用TCP/IP进行网络互联，第一卷》（第四版）中译本，[美]Douglas E. Comer，林瑶等译，电子工业出版社；其第13章中对RTT、RTO等的计算有比较详细的阐述，Linux下对此的实现相同。

10. 根据3.6.3中步骤1，继续上一节的实验，配置路由器，取消对接口的速率限制。请写出命令：

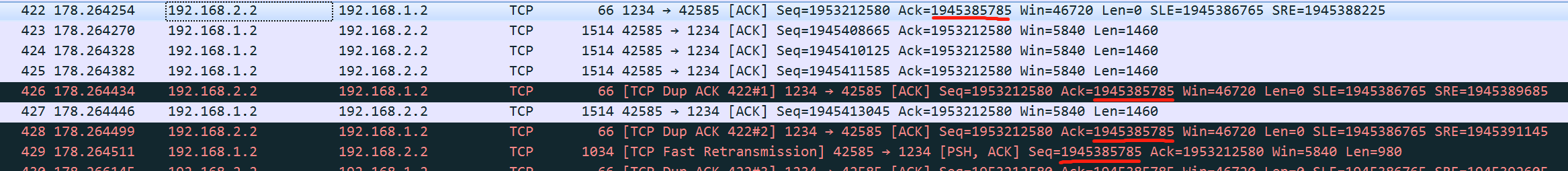
[Router-Ethernet0/0] undo qos lr outbound

[Router-Ethernet0/1] undo qos lr outbound

11. 根据3.6.3中“快重传和快恢复算法分析”步骤6回答：

1. 请写出前三个重复ACK的报文序号。在第三个重复ACK报文到达后，发送报文发生什么变化？为什么？

**答：**



**前三个重复ACK的报文序号：422、426、428.**

**在第三个重复ACK报文到达后，发送方立即进行报文重传。**

**因为发送端连续收到3个重复的ACK报文时，即可认为某一段报文丢失并且网络仍能够进行正常报文传输，因此不必等待那个报文的定时器超时，而直接重传那个认为是丢失的报文段。**

1. 在前三个重复的ACK报文到达期间，ssthresh和cwnd有何变化？RTT和RTO有何变化？为什么？

**答：**

**第三个重复的ACK报文到达时，ssthresh = max( cwnd/2, 2 ) = cwnd/2 = 18/2 = 9；**

**cwnd = min( cwnd, 已发送的报文数－已接收但未确认的报文数＋重传的报文数＋3 ) = 17**

**RTT和RTO没有变化，因为重复的ACK确认和重传的报文不会影响RTT的值，且没有超时重传因此RTO也不变。**

1. 在第三个重复ACK报文后是否还有重复ACK报文到达？随之（直至传输结束），ssthresh和cwnd有何变化，呈何种规律？为什么有此规律？

**答：**

**还有重复ACK报文到达。**

**接收重复ACK报文期间，ssthresh不变，cwnd = min( cwnd, 已发送的报文数－已接收但未确认的报文数＋重传的报文数＋3 ).**

**没有重复ACK报文后，ssthresh不变，将cwnd缩小到ssthresh，并使用拥塞避免算法，发送方每收到cwnd个非重复的ACK报文，cwnd加1.**

12. 根据3.6.4中“糊涂窗口综合症和Nagle算法分析”步骤4回答：

1. 分别分析两次TCP传输过程中数据发送部分的前面几条报文，结合TCPTest程序参数设置，分析其数据长度的变化，并解释为什么？

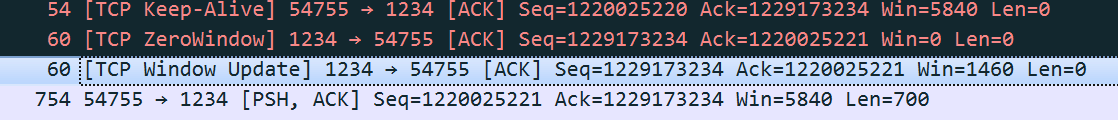
**答：**

**启用Nagle算法，除了带PUSH标志的报文外其他报文的数据长度都为MSS(1460)；因为发送缓存被设置为2000，每次写入字节数为800，会判断当前数据是否有发送缓存一半(1000)或一个MSS(1460)，因此写入第一个800后不会发送，写入第二个800后共1600，则发送一个MSS(1460)长度报文，还剩140，再写入第三个800后共940，不够缓存一半(1000)继续写入第四个800共1740，再发送一个MSS(1460)长度报文，还剩280.**

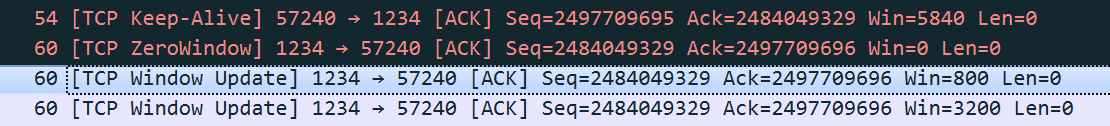
**不启用Nagle算法，数据长度为800，一有数据写入缓存就发送。**

1. 在两次TCP传输过程中，窗口通告为0后，窗口张开时通告的第一个窗口大小分别是多少？为什么？

**答：**



**启用Nagle算法，窗口张开时通告的第一个窗口大小为1460；因为会采用推迟确认技术，接收缓存的可用空间至少达到总空间的一半(50000)或一个MSS(1460)，才会进行窗口通告。**



**不启用Nagle算法，窗口张开时通告的第一个窗口大小为800；有小窗口则立即通告。**

1. 通过这两次TCP传输实验，比较分析Nagle算法的作用。

**答：**

**Nagle算法能有效避免糊涂窗口综合征的现象。**

**发送端通过推迟发送缓存中的数据，避免小数据的不断发送；**

**接收端通过延迟确认和窗口通告，同样能避免发送端发送小数据。**