**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 | 例如周三上午 |  |  |  |
| 滑动窗口机制和窗口侦查机制分析 |  |  |  |  |
| 慢启动、拥塞避免及拥塞处理和超时重传机制实验 |  |  |  |  |
| 快重传和快恢复实验 |  |  |  |  |
| 糊涂窗口综合症和Nagle算法分析实验 |  |  |  |  |

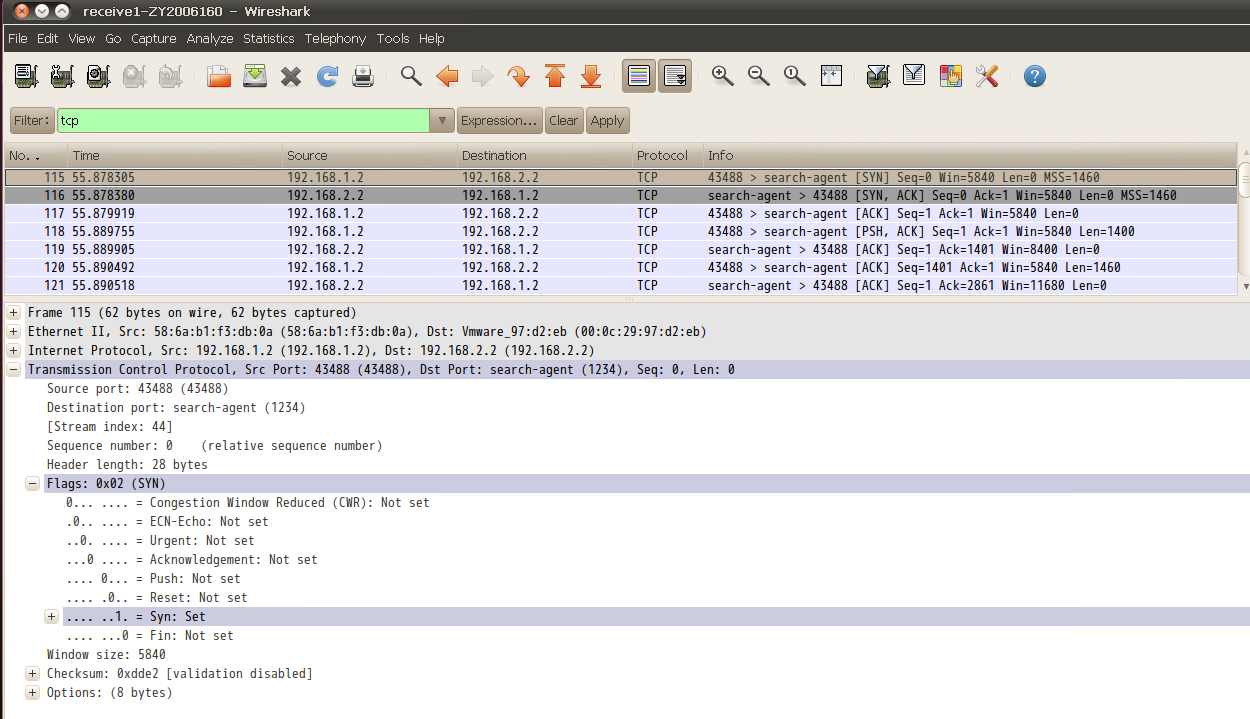
1、根据2.6中步骤3回答：TCP的连接和建立采用的是：客户端/服务器 方式，PCA是 客户端 ，PCB是 服务器 。先点击发送再点击接收，会出现什么问题？为什么？

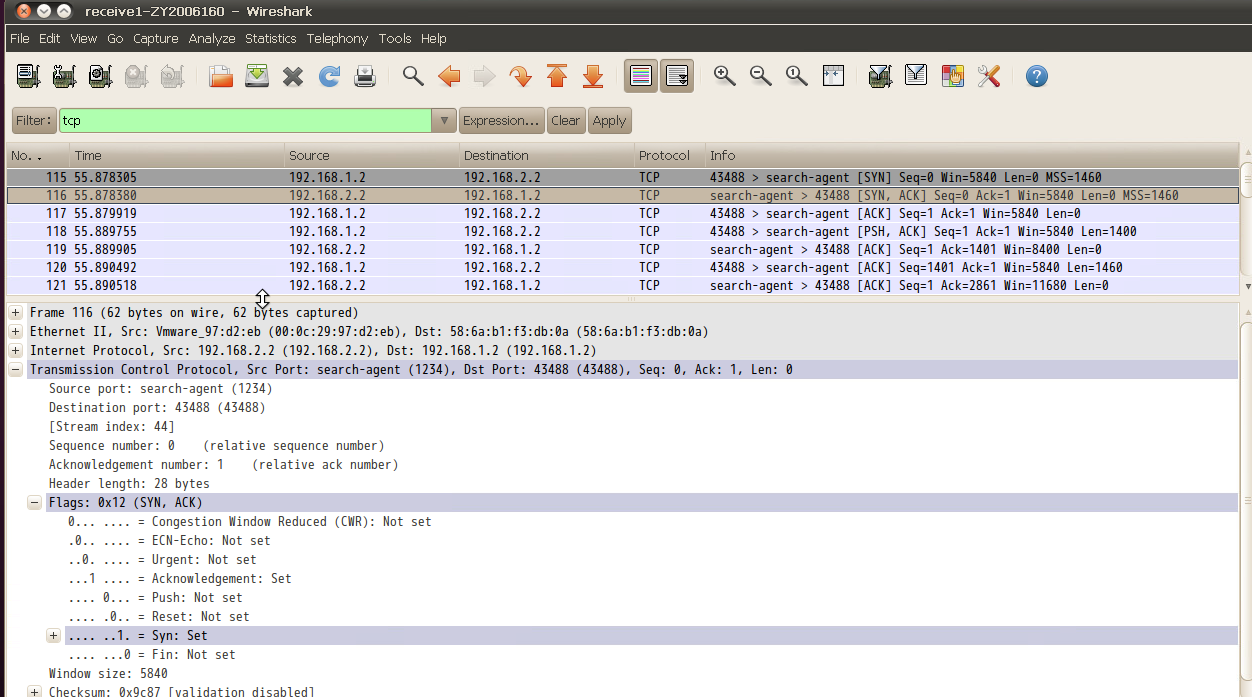
**答：**

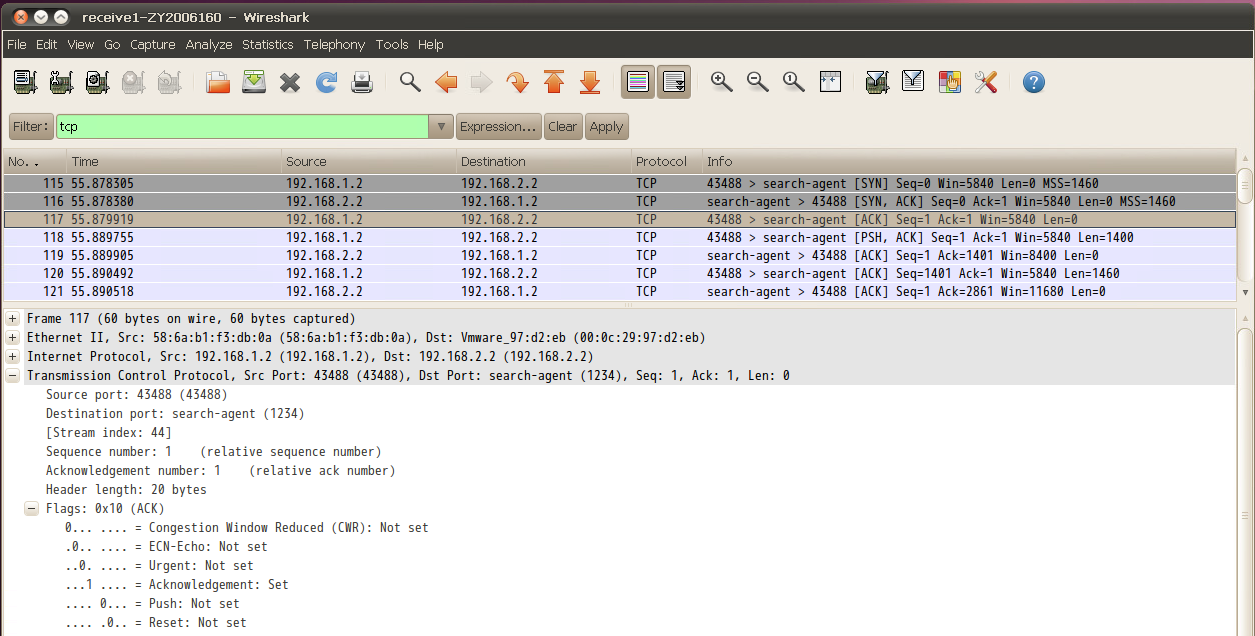
**先发送再接收会导致发送失败，因为C/S模式下，如果服务器未启动，则客户端就无法连接上服务器。**

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 |  |  |  |
| Sequence Number |  |  |  |
| Acknowledgement Number |  |  |  |
| Ack |  |  |  |
| Syn |  |  |  |







3、根据2.6中步骤6回答：

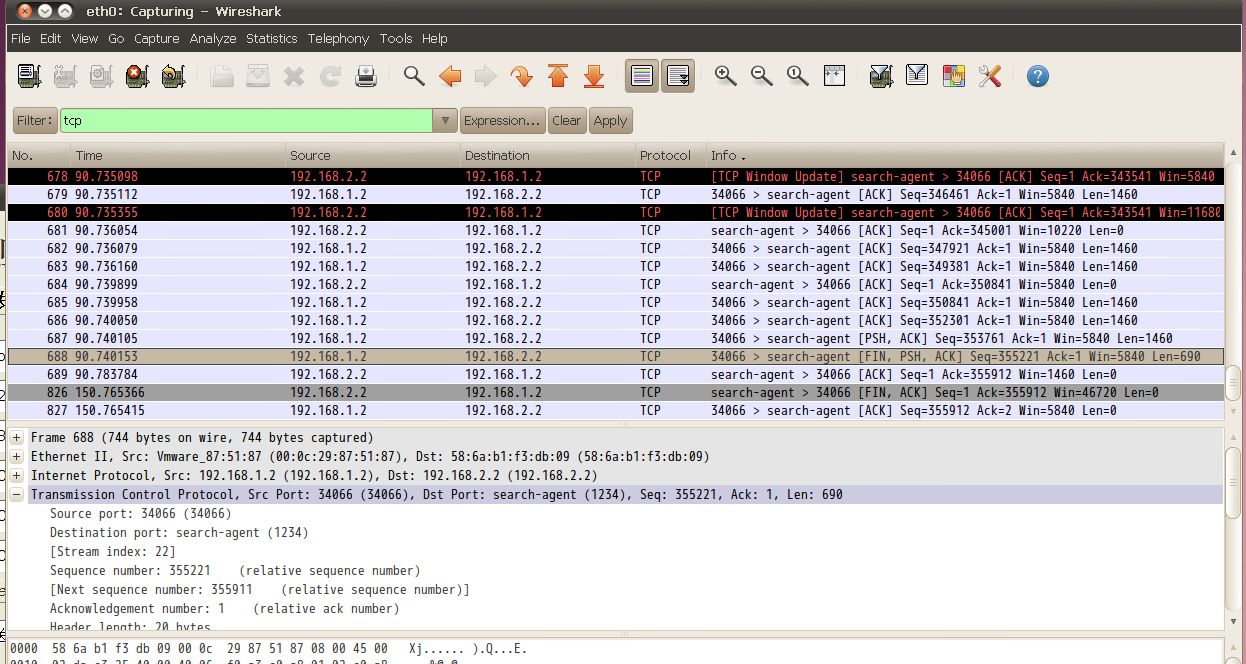
TCP连接建立时，其报文首部与其它TCP报文不同，有一个“Option”字段，它的作用是什么，值为多少？结合IEEE802.3协议规定的以太网最大帧长度分析此数据是怎样得出的。

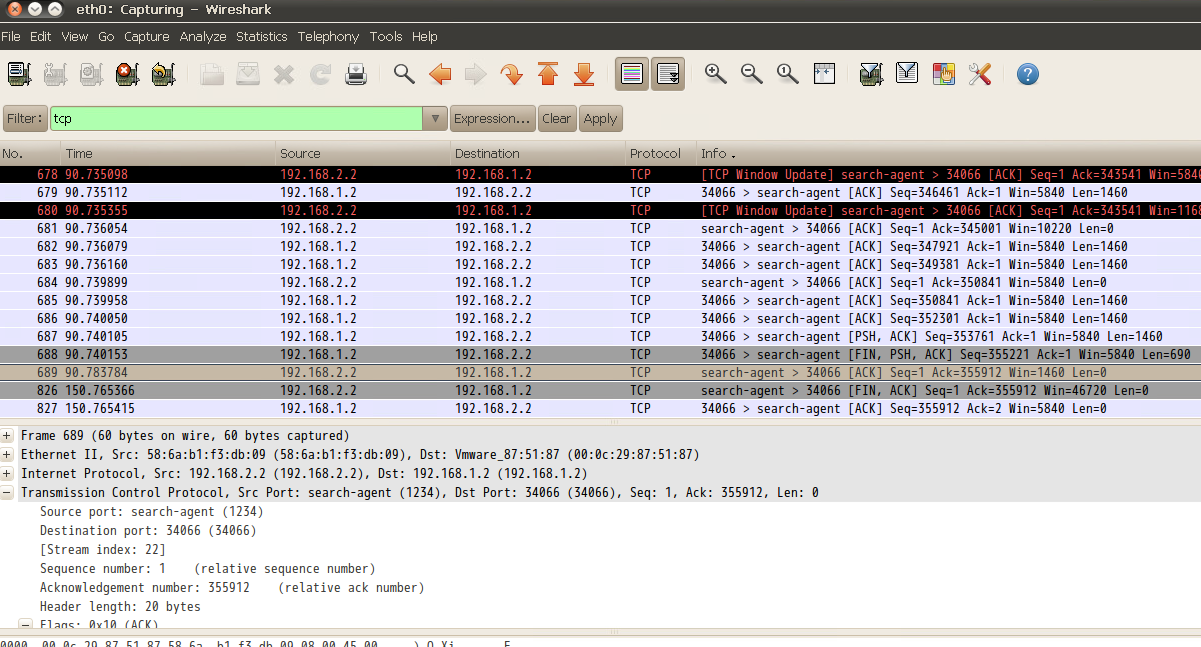
**答：**

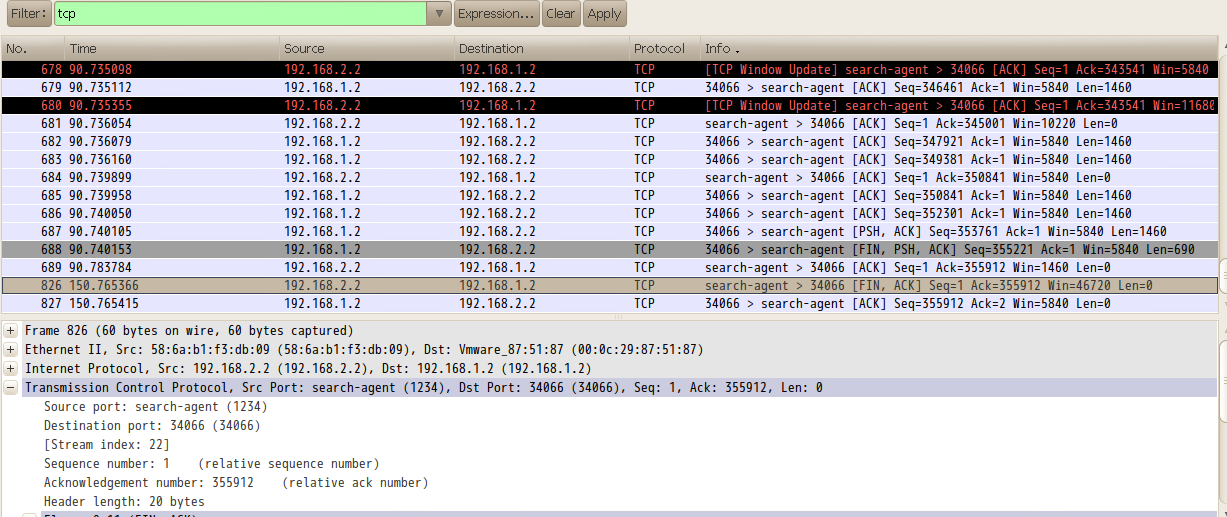
**Option字段用于协商双方的报文中数据段的长度，及MSS，取二者最小的。MSS被用于数据传输过程。值为1460字节。IEEE802.3规定以太网最大帧长度为1518字节，则MSS=以太网帧长度（1518）-以太网帧头帧尾长度（18）-IP首部长度（20）-TCP首部长度（20）=1460**

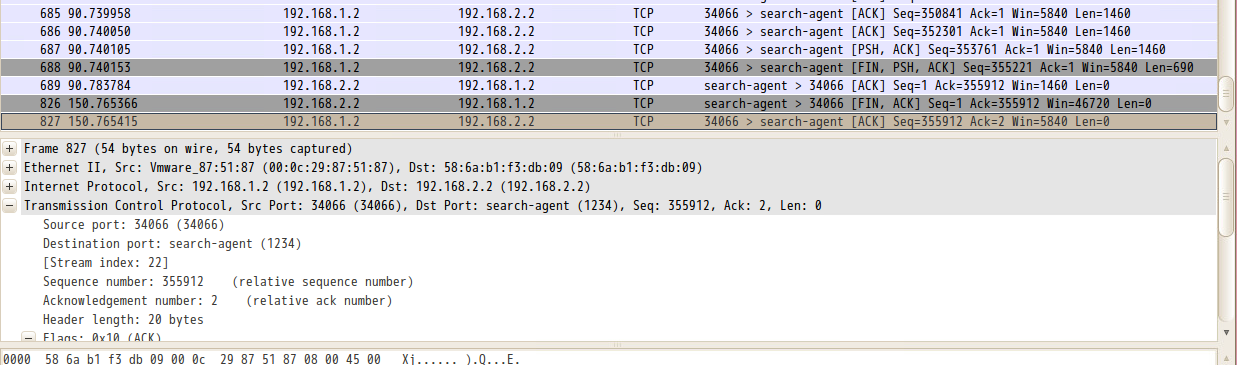
4、根据2.6中步骤7：结合预习报告，分析TCP连接的释放过程，选择TCP连接撤消的四个报文，将报文信息填入下表。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 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 |  |  |  |  |
| Sequence Number |  |  |  |  |
| Acknowledgement Number |  |  |  |  |
| Ack |  |  |  |  |
| Fin |  |  |  |  |









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 |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |

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报文长度-IP首部长度字段值\*4-TCP首部长度字段值\*4。等于**

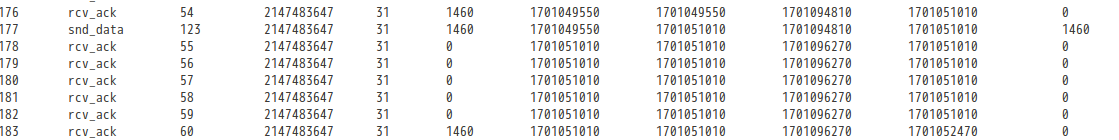
6. 根据3.6.1中“ 滑动窗口机制和窗口侦查机制分析”步骤6回答：

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：**

**发送窗口逐渐增大，此时发送方处于慢启动过程，因为发送窗口大小=min（接收窗口大小，拥塞窗口大小），慢启动中，拥塞窗口较小，且每收到一个ACK后，拥塞窗口大小增加一个MSS，因此发送窗口不断增大。**

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。



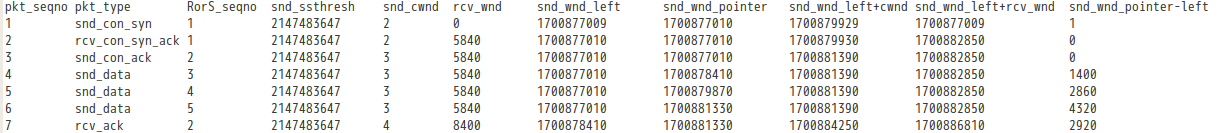
1. 分析文件send2-组座号-tcpsndwnddata.txt，选中三次握手连接建立后的前4条报文记录（3条DATA报文、1条ACK报文，序号为4、5、6、7），记下发送方发送窗口的相关值（rcv\_wnd , snd\_wnd\_left , snd\_wnd\_point , snd\_wnd\_left+cwnd , snd\_wnd\_left+rcv\_wnd , (snd\_wnd\_point- left)）。按下表分析计算接收方（及发送方）的窗口的相关值。

**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 |  |  |  |  |  |
| 发送窗口右边沿 |  | | | | |
|  | 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 |  |  |  |  |  |
| 接收方接到DATA后 |  |  |  |  |  |

**6号报文（sender----data---->receiver）**

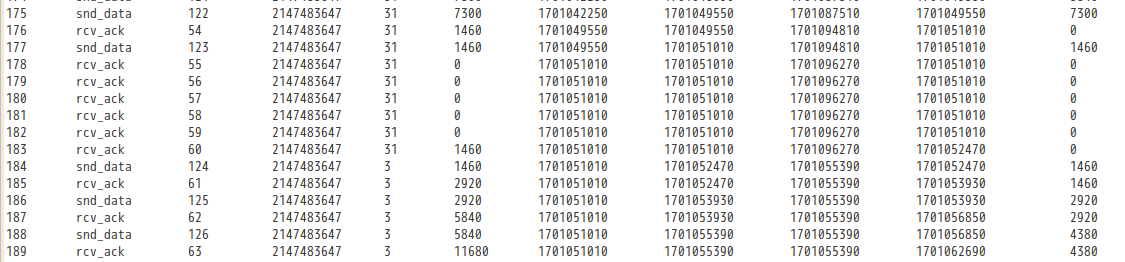
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | rcv\_wnd | snd\_wnd\_left | snd\_wnd\_pointer | snd\_wnd\_left+cwnd和  snd\_wnd\_left+rcv\_wnd | snd\_wnd\_point- left |
| 发送方发出报文 |  |  |  |  |  |
| 发送窗口右边沿 |  | | | | |
|  | 通告的接收窗口 | 接收窗口左边沿 | 接收窗口指针 | 接收窗口右边沿 | 在接收缓存中的数据量（即未确认的数据） |
|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

****

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

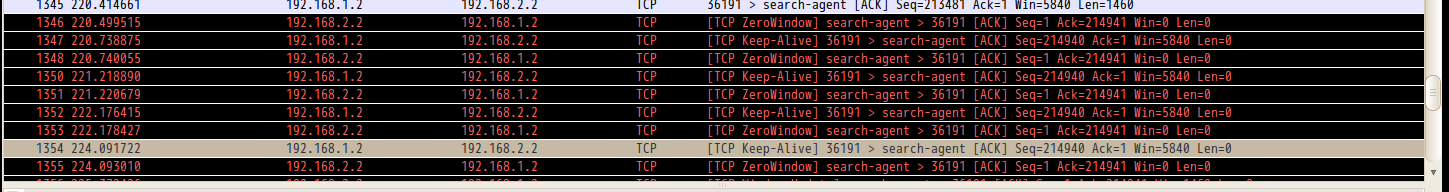
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | 通告的接收窗口 | 接收窗口左边沿 | 接收窗口指针 | 接收窗口右边沿 | 在接收缓存中的数据量（即未确认的数据） |
| 接收方发出ACK |  |  |  |  |  |
|  | rcv\_wnd | snd\_wnd\_left | snd\_wnd\_pointer | snd\_wnd\_left+cwnd和  snd\_wnd\_left+rcv\_wnd | snd\_wnd\_point- left |
| 发送方接到ACK后 |  |  |  |  |  |
| 发送窗口右边沿 |  | | | | |
| 发送方接到ACK前 |  |  |  |  |  |
| 发送窗口右边沿 |  | | | | |

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，..，……）。



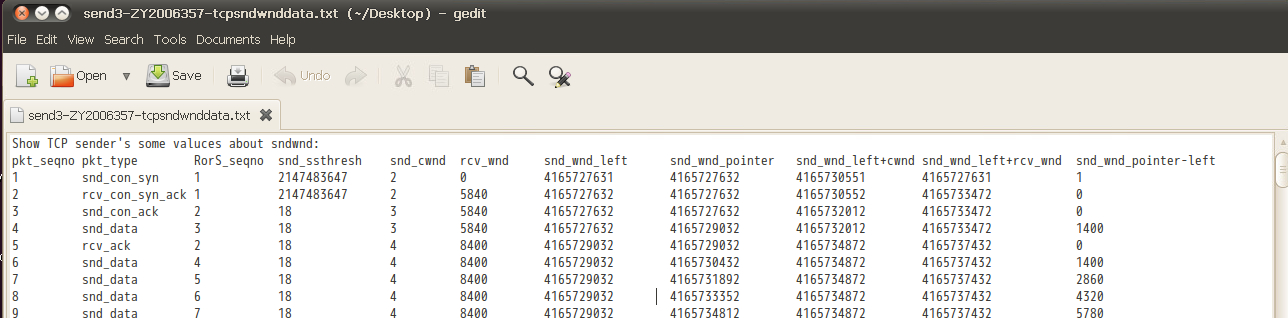
7. 根据3.6.1中“ 滑动窗口机制和窗口侦查机制分析”步骤7回答：

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。



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。



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？
2. 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？

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 |  |  |  |
| 80Kbps |  |  |  |

**答：**

**同种传输速率下，新的重传时间=旧的重传时间\*2。不同传输速率下，初始的重传时间=β\*RTT，可见传输速率越高，RTT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？
2. 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：

**答：**

**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：**

**发送端立即重传丢失报文。因为连续收到3个ACK后，发送方认为网络仍可以正常通信，因此不用等到该报文的定时器到时，而是直接认定该报文丢失，因此立即重传。**

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:**

**ssthresh=max(cwnd/2,2)=9，cwnd=min(cwnd，已发送的报文数-已接受但未确认的报文数+重传的报文数+3).RTT无变化，因为没有ACK报文，故无法计算RTT，RTO也不变化，因为不是因为超时重传引起的报文重发，因此RTO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：**

**有重复ACK收到。**

**重复ACK接收过程中：ssthresh不变化，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 )。**

**新的ACK收到后，ssthresh不变化，cwnd变为ssthresh，直接进入拥塞避免阶段，之后每收到cwnd个ACK后，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算法时，由于发送方发送缓存大小为2000字节，第一次写入800字节时，剩余1200字节，超过一半，则不发送，进行第二轮发送缓存填充，写入1600字节时，发送一个MSS大小（1460字节），剩余140字节，继续重复，直到写入到1740字节，又发送1460字节，此时剩余280字节。**

**如果不启用Nagle算法，则每次填入800字节时，都立即发送。**

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算法通告的窗口大小为1460字节，因为采用推迟通告技术，需要等到可用空间的大小达到一个MSS的大小才去通告。**

**不启用Nagle算法通告的窗口大小为800字节，因为有小窗口时则立即进行通告。**

1. 通过这两次TCP传输实验，比较分析Nagle算法的作用。

**答：**

**Nagle算法能够有效避免糊涂窗口综合征。发送方通过推迟发送缓存中的数据，避免小数据包不断发送；接收方通过延迟窗口通告，也能避免发送小型数据包。**