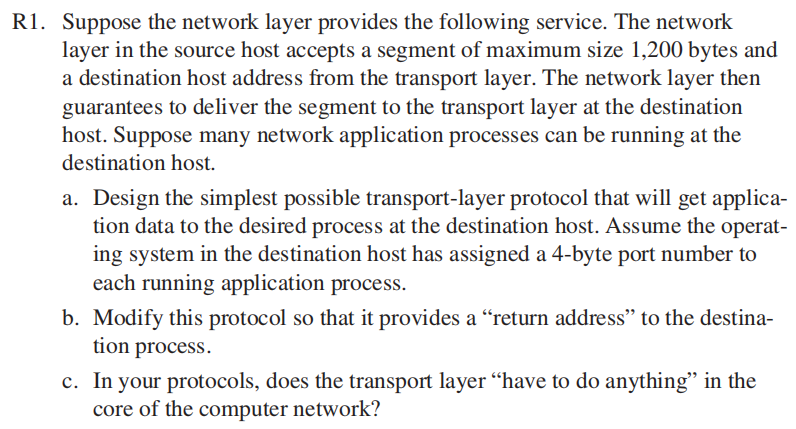
注：英文题目来自第八版教材，答案是根据第七版中文教材做的，所以可能会有所出入或者对不上号。未用第七版英文题目进行更正，因为第八版的新题也很有价值，可以日后研究。

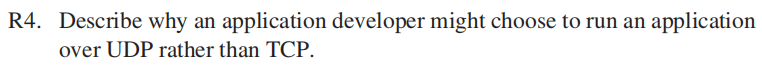


解：

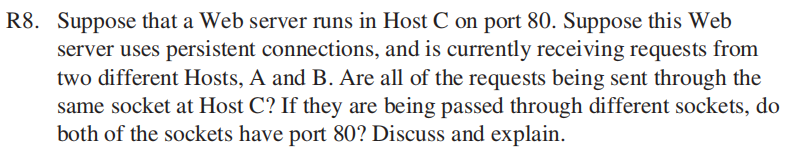
a.设计的最简单运输层协议将包括两部分：目的端口号和报文。因为该协议只要使应用程序数据到达位于目的主机所需要的进程就可以了，它将会在下发到源主机的网络层时加上目的主机的IP地址，保证了主机之间的逻辑通信。到达目的主机的网络层后会把该运输层报文段提取上交到运输层，运输层根据目的端口好在本机的网络进程中找到有相同的端口号的目的进程，并交付应用数据。

b.为了提供让目的进程返回的地址，在需要加上一个源端口号字段。这样目的进程就能通过把源端口号设置为目的端口号，进而向源进程传输数据。

c.它并不需要做“任何事”。目前它只提供交付数据的功能，不具备诸如拥塞控制、确保数据完整性等功能。

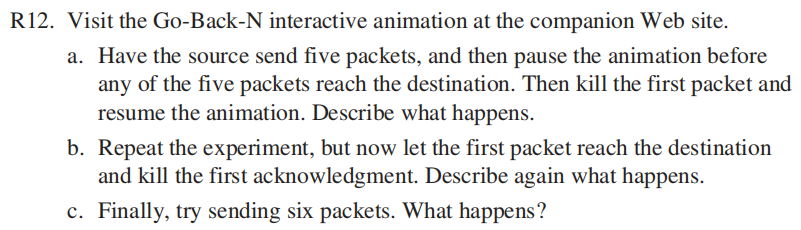


解：这个问题问的是在什么情况下，UDP的优点明显盖过TCP的缺点。UDP协议是一种极为简化的运输层协议，它不提供不必要的服务，大概就是在IP协议上加上源和目的地的端口后等信息，所以它可以随时地，以任何速率向其他端系统发送数据。TCP协议有许多优良特性，它确保数据完整性，提供拥塞控制，但这些特性会增加端到端通信的时延。如果一个应用程序需要提供实时服务，而且能够容忍一定的分组丢失，那么UDP协议是更好的选择。



解：A和B的请求会通过80端口找到服务器进程, 就这里而言它们通过为与C的相同套接字, 这个套接字具有端口80。

当它们与服务器进程建立连接的时候，服务器进程会单独为它们分配套接字，通过专门的套接字响应客户端的请求。这两个套接字就不具有80端口了。

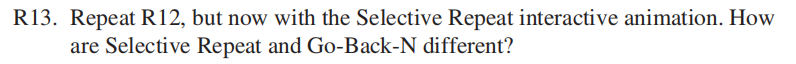


解：

a. 丢包造成一定时间后，所有五个包都被重传。

b. 由于Go-Back-N使用累积数据，ACK的丢失没有触发任何重传。

c. 发送方无法发送第六个分组，因为发送窗口大小固定为5。

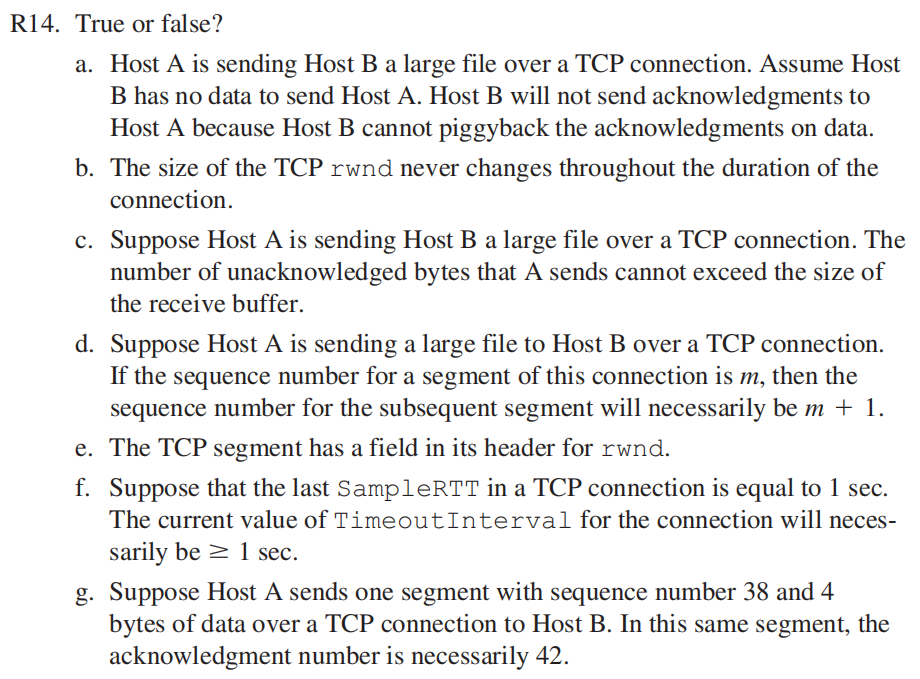


解：

a. 分组丢失时，接收的4个分组被缓冲接收器。完成后超时，发送方重新发送丢失的数据包，接收方传送缓冲的数据包数据包以正确的顺序应用。

b. 接收器为丢失的ACK发送了重复ACK。

c. 发送窗口大小固定为5时，发送方无法发送第六数据包当分组丢失时，返回-N重传所有的分组，而选择性地重传所有的分组。仅重发丢失的数据包。在丢失确认的情况下，选择性重复发送重复ACK，并作为返回-N使用的累积确认，不需要重复ACK。



解：

a. 错误。主机B将不向主机A发送确认这句话违背了一个可靠数据传输协议的基本原则。首先要明确为了确保可靠的数据传输，接收方向发送方发送确定报文是协议的一部分。再分析推导出这一错误结论的原因:主机B不能随数据捎带确认。书本上提出捎带的概念时，给的是一个具体场景，发送方要发送一个字符，接收方要返回该字符进行回显，所以顺便把确认信息放入到发给发送方的数据的报文段中。要区分清楚确认和捎带确认之间的关系。

b. 错误。rwnd=RcvBuffer-[LastByteRcvd-LastByteRead]。rwnd表示的是接收窗口的大小，这取决于接收方应用程序从缓存中读取数据的速率和发送方发送的速率，是会变化的。

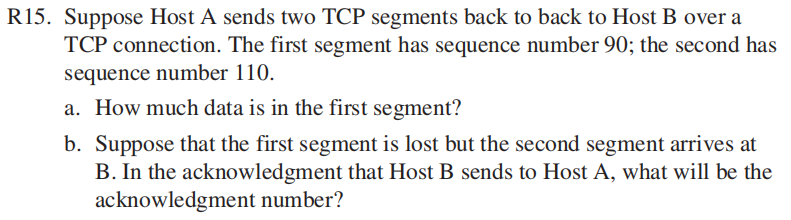
c. 正确。讲的就是TCP的流量控制。

d. 错误。序号是根据TCP数据的字节流决定的，而不是建立在报文序列之上。后继报文段的序号应该是m+n，而n是最大报文段长度。

e. 错误。TCP报文段有接收窗口字段，rwnd存放在接收窗口字段中。

f. 正确。TimeoutInterval=EstimatedRTT+4\*DevRTT。

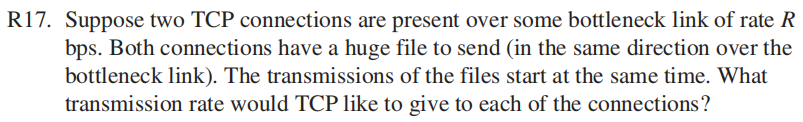
g. 错误。概念性错误，确认号是期待接收方发送的序号。



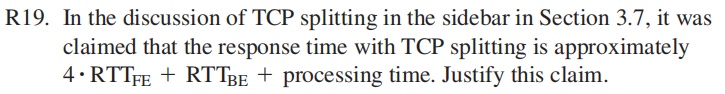
解：

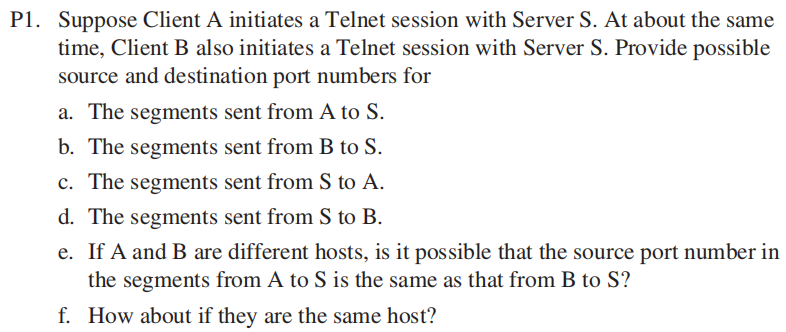
a. 110-90=20个字节。

b. 90。在建立连接的时候主机B就知道要先接收90。



解：这里涉及到2条TCP连接的公平性问题，很难保证两条TCP连接分配均等的传输速率。但是2条TCP连接的速率之和是会在R/2～R之间浮动的。至于哪条快一点，哪条慢一点是不确定的。



解：我认为该断言是合理的。它给出的是一条包含重要时延参数的公式。在实际情况中，某些参数可能可以省略，但是在省略之前还是要经过考虑的。

解：假定A向S的源端口号为x，B向S的源端口号为y

1. A向S的源端口号为x，目的端口号为23。

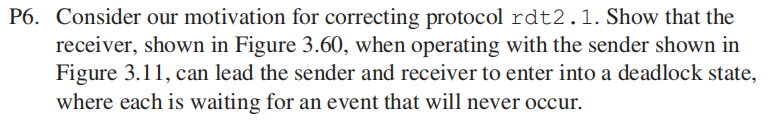
b. B向S的源端口号为y，目的端口号为23。

c. S向A的源端口号为23，目的端口号为x。

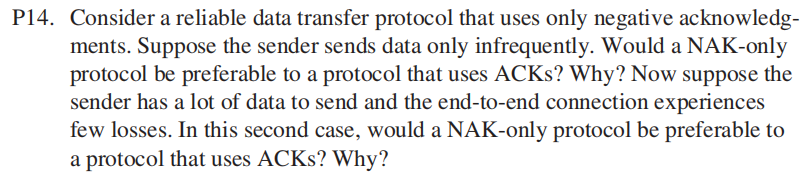
d. S向B的源端口号为23，目的端口号为y。

e. x，y可能相同。

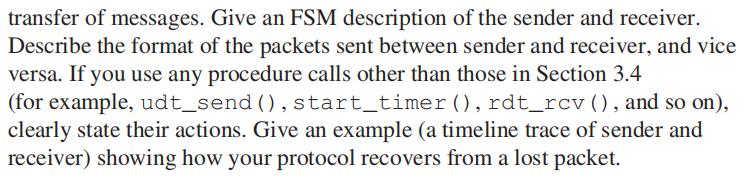
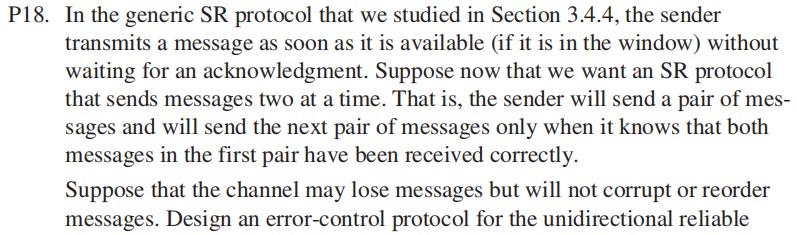
f. 不可能。



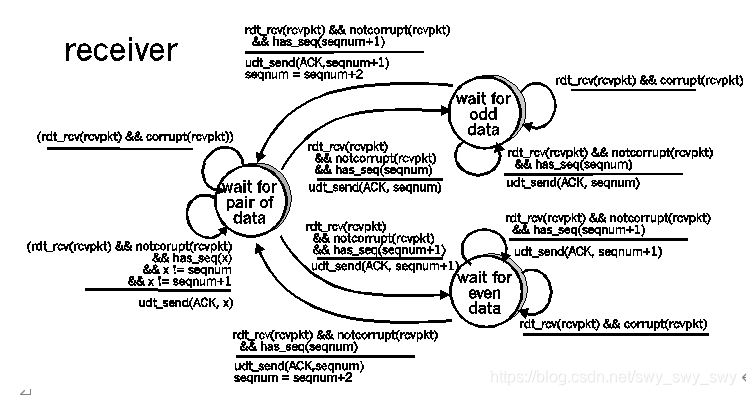
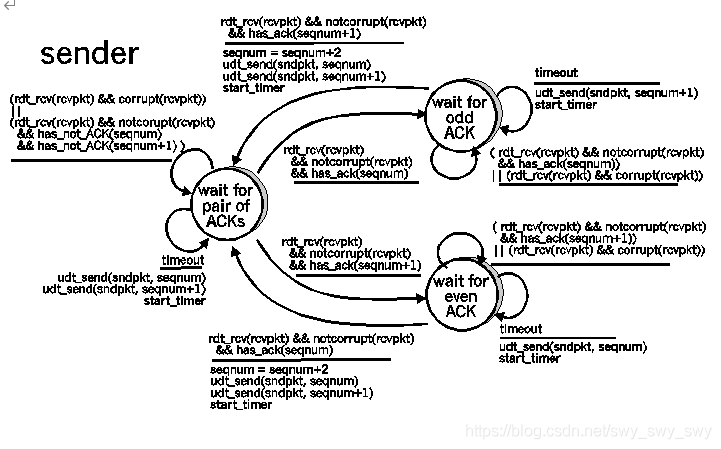
解：如果rdt2.1发送方正处于“等待来自上层的调用0”，接收方处于“等待来自下层的0”，发送方发送序号为0的分组，而接收方正确接收并向发送方发送ACK；此时发送方处于“等待ACK或NAK 0”，接收方处于“等待下层的1”，如果此ACK损坏，发送方重发序号0的分组，而接收方会发送NAK，这将导致一个死循环；其实此接收方并没有标注初始状态，如果发送方初始状态为“等待来自上层的0”，接收方初始为“等待下层的1”，也会导致上述死锁。

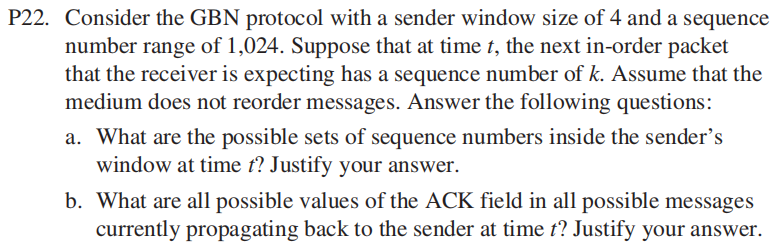


解：分组x丢失只能被接收方检测到，且只有x-1，x+1都被接收后。如果发送方在发送x之后隔较长时间才发送x+1，那么这段时间x将一直不会被重发。而当数据量较大且很少丢包时，用NAK协议发送的数据包的数量明显比ACK协议少。



解：

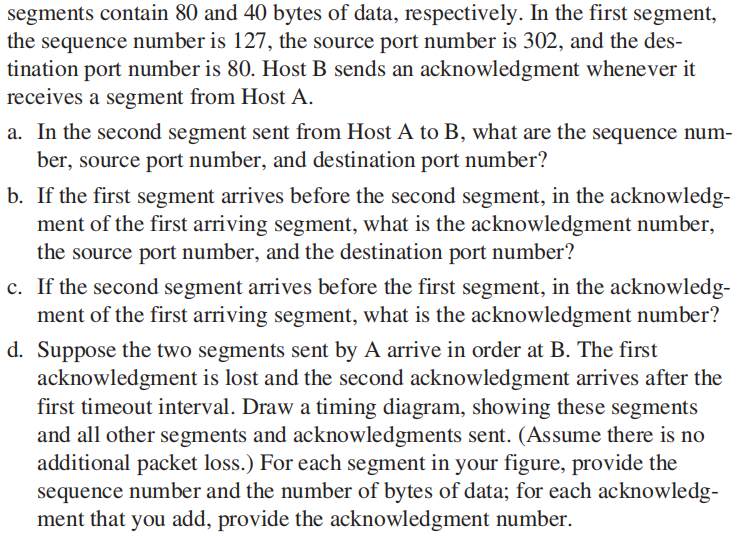
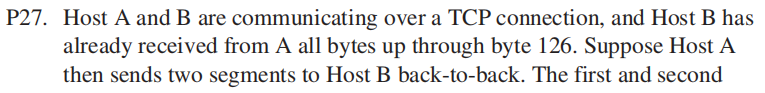




解：

a. 考虑两种极端情况：发送方发送k-4，k-3，k-2，k-1，接收方都完整得接收并发送ACK，但ACK全都未传到发送方，接收方的期待序号为k，而发送方窗口序号为[k-4,k-1]；如果ACK全都传回，则发送方更新base，其序号为[k,k+3]。因此序号可能是[k-4,k+3]。

b. 如果接收方期待k，则它一定将比k-1小的ACK发送出去了，如果要使发送方发送k-1，那么它至少已经接收到了k-5的ACK。因此正在传播回发送方的ACK序号可能是[k-4,k-1]

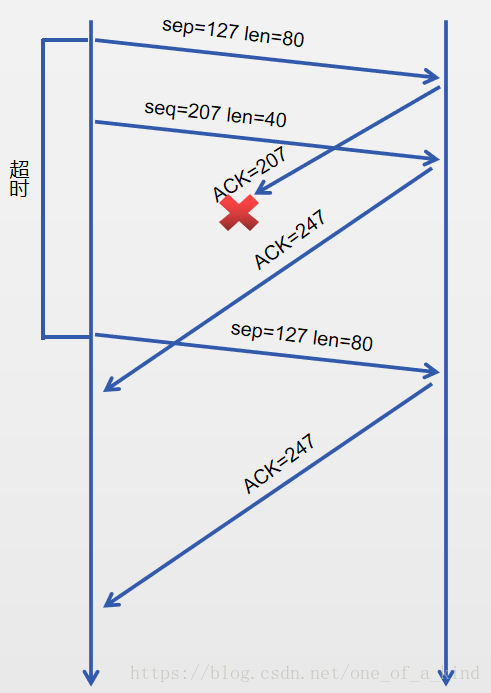


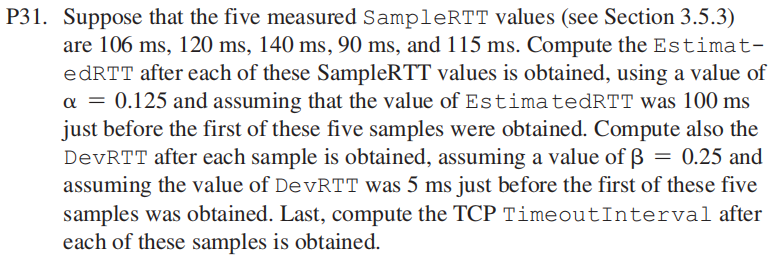
解：

a. 序号、源、目的端口号分别为 207、302、80。

b. 序号、源、目的端口号分别为 207、80、302。

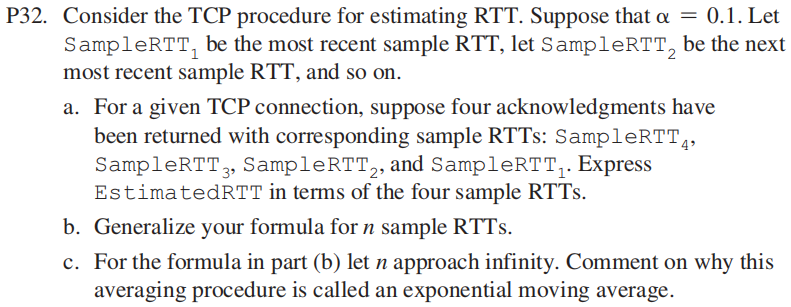
c. 127。

d. 



解：

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 迭代次数 | 旧ERTT | SRTT | 新ERTT | 旧DRTT | 新DRTT | TimeoutInterval |
| 1 | 100ms | 106ms | 100.75ms | 5ms | 5.75ms | 123.75ms |
| 2 | 100.75ms | 120ms | 103.16ms | 5.75ms | 9.125ms | 139.66ms |
| 3 | 103.16ms | 140ms | 107.765ms | 9.125ms | 16.05ms | 171.965ms |
| 4 | 107.765ms | 90ms | 105.54ms | 16.05ms | 16.48ms | 171.46ms |
| 5 | 105.54ms | 115ms | 106.72ms | 16.48ms | 14.725ms | 165.62ms |

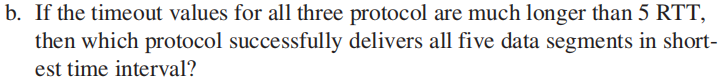
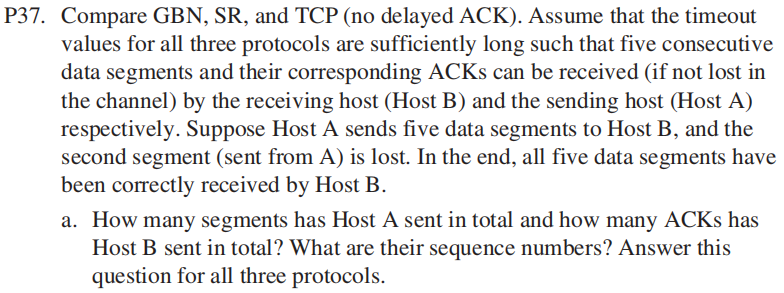


解：

a. EstimatedRTT=0.9(0.9(0.9(0.9EstimatedRTT+0.1SampleRTT1)+0.1SampleRTT2)+0.1SampleRTT3)+0.1SampleRTT4=0.94EstimatedRTT+0.93\*0.1SampleRTT1+0.920.1SampleRTT2+0.9\*0.1SampleRTT3+0.1SampleRTT4

b. 

c. 随着n趋于无穷，每一个RTT项的权重都呈0.9n的指数下降趋势。



解：

1. GBN：

A：首先发送分组12345，后来重发2345，总共9个分组。

B：首先发送ACK1111，后来发送2345，总共5个ACK。

SR：

A：首先发送分组12345，后来重发2，总共6个分组。

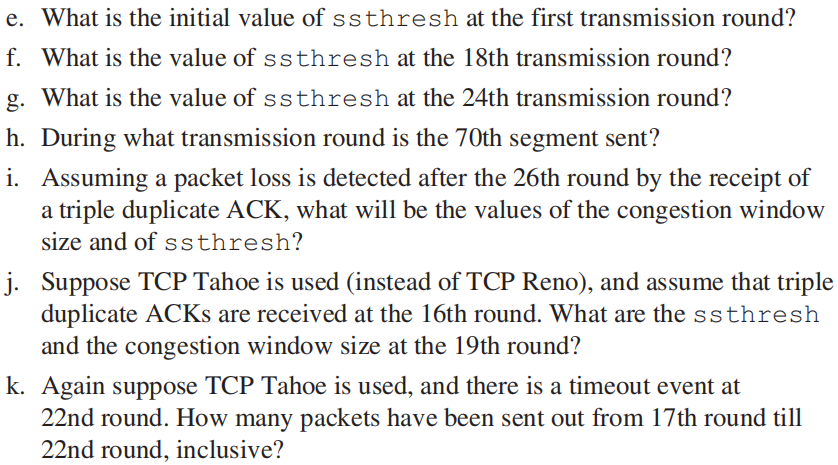
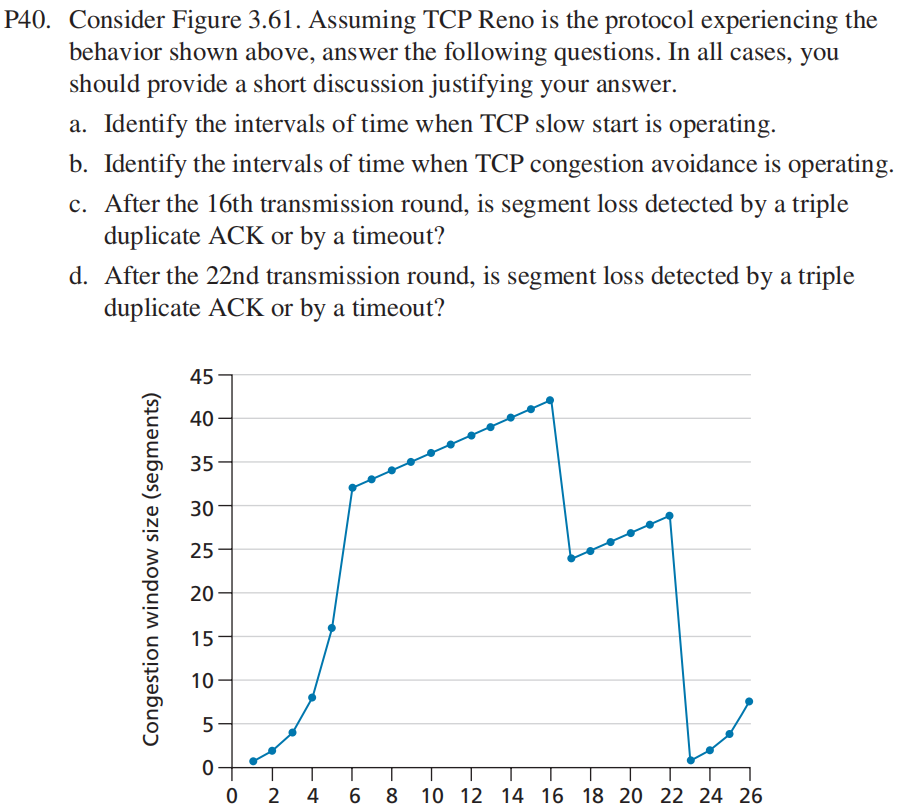
B：首先发送ACK1345，后来发送2，总共5个ACK。

TCP：

A：首先发送分组12345，后来重发2，总共6个分组。

B：首先发送ACK2222，后来发送6，总共5个ACK。

1. GBN、SR需要等待超时，而TCP使用快速重传，故TCP最快。



解：

a. [1, 6]和[23, 26]。

b. [6, 16]和[17, 22]。

c. 三个冗余ACK。

d. 超时。

e. 32。

f. 42/2=21。

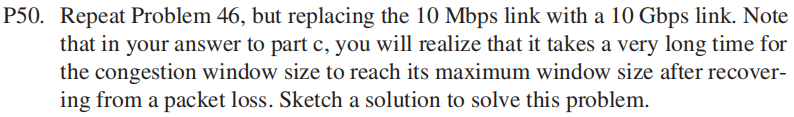
g. 29/2=14。

h. 分组1(p1)在传输轮回1(t1)中发送，p2～p3在t2中发送，p4～p7在t3中发送，p8～p15在t4，p16～p31在t5，p32～p63在t6，p64～p96在t7,因此分组70在第7个传输轮回内发送。

i. ssthresh=4，cwnd=ssthresh+3MSS=7。

j. 在第16个传输轮回时，ssthresh=cwnd/2=21，cwnd=1并进入慢启动状态，则在第19个传输轮回中，ssthreash=21，cwnd=4。

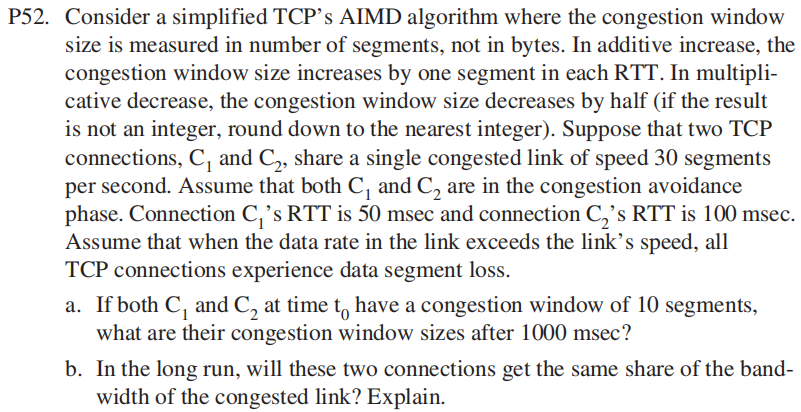
k. t17：1个，t18：2个，t19：4个，t20：8个，t21：16个，t22：21个，1+2+4+8+16+21=52个。



解：

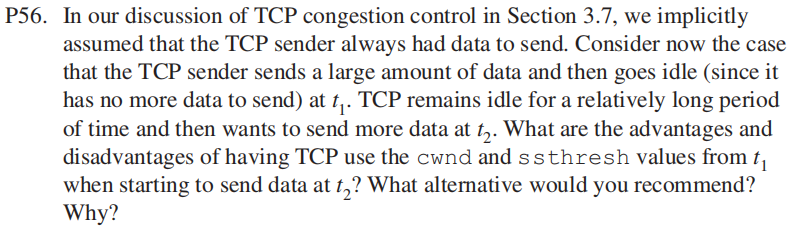
a. 都是1

b. 不



解：

此时传输的总量=W(2a+1)/2a，丢包率L=2a/w(2a+1)，不论吞吐量如何，需要的时间总为loga+12\*RTT。



解：

a.6RTT+15S/R

b.4.5RTT+15S/R

c.8.5RTT+15S/R