“第 5 章 网络层”习题解答

5.01 试说明运输层在协议栈中的地位和作用。运输层的通信和网络层的通信有什么重要的区别？为什么运输层是不可缺少的？

答： 运输层在协议栈中处于 IP 层之上，应用层之下。它的作用是向应用层的通信实体提供服务。它可以把应用层的通信实体交给的数据报文，通过网络层传输到目的主机，并上交给目的主机的应用层对等通信实体。

运输层通信和网络层的通信的重要区别是：运输层通信实现了两个主机中的进程间的数据传输，而物理层实现了两个主机间的数据传输。

在实际的计算机网络通信中，实现的通常是应用实体（进程）间的通信。而在一个主机中经常有多个应用进程同时分别与另一个主机中的应用进程进行通信，因此设立运输层可以方便的实现主机间应用实体间的通信复用户与分用。另外，由于网络层（IP 层）提供了无连接的不可靠的通信服务，因此当应用层要求可靠的通信服务时，仅靠网络层提供的服务是不能满足其要求的，此时可在运输层设立可靠的进程间的通信功能，以满足这一要求。从以上分析可见，运输层是不可缺少的一个网络层次。

5.02 网络层提供数据报或虚电路服务对上面的运输层有何影响？

答：网络层供数据报服务对上面的运输层的影响是：数据报是一种无连接不可靠的通信的服务，它不能保证所传输按的数据正确性和及时性，因此，运输层若想向上提供可靠的或实时性通信服务，必须增添相应的功能。

网络层提供虚电路服务对上面的运输层的影响是：虚电路服务是一种面向连接的可靠的通信服务，它可以较好的保证所传输按的数据正确性和及时性，因此，运输层若想向上提供可靠的或实时性通信服务时，可直接利用这些功能。这样网络层的设计和实现就变得比较简单。当由于虚电路服务在通信前，需先建立通信节点间的虚电路，并在通信完成后，还需释放所建立的虚电路，由于建立和释放虚电路都需要一定的时间和占用一定的网络通信资源，因此对于一些随机性或突发性高的通信应用，其通信效率是不高的。

5.03 当应用程序使用面向连接的 TCP 和无连接的 IP 时，这种传输是面向连接的还是无连接的？

答：当应用程序使用面向连接的 TCP 和无连接的 IP 时，就应用程序来讲，这种传输即不是面向连接的，也不是无连接的。因为，一种通信方式的确定，主要是指对等层间的通信实体所采用的形式。一个主机中的应用程序是要与另一个主机中的应用程序实现对等通信，其通信方式取决于应用层所采用的通信方式。而面向连接的 TCP 是在运输层通信实体间的所采用的通信方式， 而无连接的 IP 是在网络层通信实体间的所采用的通信方式。

5.05 试举例说明有些应用程序愿意使用不可靠的 UDP，而不愿意采用可靠的 TCP。

答：由于 UDP 采用无连接和面向报文的的通信方式，没有拥塞控制，支持一对一或一对多的交互通信，且首部开销小等特点，因此特别使用于一些要求可靠性和准确性较小及时性较强的应用通信。如 IP 电话、实时视频会议和远程网络仿真终端通信等。

在 IP 电话和实时视频会议系统中，传输的是语音数据和视频数据。对于语音通信和视频通信，强调的是数据传输的连续性和适当稳定的速率，容许一定的数据差错。而 UDP 而可以很好的满足这些要求。如果采用采用可靠的 TCP 协议，尽管能保证其传输数据的正确性，但由于建立连接和释放连接以及为保证可靠传输等带来的较大的通信开销，且通信的实时性也不一定能得到很好的保证。

对于远程网络仿真终端通信，仿真终端和服务器之间需要及时的交流的信息，特别是仿真终端需要把用户的操作请求及时的发送给服务器，而这些请求信息往往是很短的（一个或几个字符）。UDP 可以把这些短的信息打包在一个 UDP 报文段中进行及时的发送，并且可以一次传输就能可把这些信息提交给服务器，以便服务器对其进行识别和处理。如果采用采用可靠的 TCP 协议，尽管能保证其传输数据的正确性，但由于建立连接和释放连接以及为保证可靠传输等带来的通信开销，相对这些短的信息是较大的，而且 TCP 通信是面向字节流的，有时还会把这些短的信息分成多个报文段进行传输，这更降低了信息传输的实时性。

5.08　为什么说 UDP 是面向报文的，而 TCP 是面向字节流的?

答：UDP 是面向报文，主要是指 UDP 把应用层每次提交的信息作为一个独立完整的报文信息打包在用户数据报中，在接收方 UDP 又用户数据报的中的数据部分一次上交给相应的应用层软件。

而 TCP 是面向字节流的，主要是 TCP 把应用层每次提交的信息，以字节为单位先存储在一个数据缓冲区中。在建立了通信连接后，就把缓冲区中的数据根据接收窗口的大小，以字节为单位取出其中的一部分数据打包在一个报文段中进行传输。在接收方，每收到一个报文段，就进行相应的处理，并把报文段中正确的数据提交给接收缓冲区中，然后再适当的时候，再以字节为单位提交给应用层软件。

5.09 端口的作用是什么？为什么端口号要分为三种?

答：端口的作用是：采用统一的格式对一个计算机中的不同应用进程进行标识，目的是在应用进程与相应的运输进程进行交互时，提供一个确定的层间接口。

运输层协议将端口号划分为熟知端口、登记端口号和客户端口号三类。其中：

熟知端口的值一般为 0 ∽ 1023。IANA 把这些端口号指派给了 TCP/IP 最重要的一些应用程序。

登记端口号的为 1024 ∽ 49151，为没有熟知端口号的应用程序使用的。使用这个范围的端口号必须在 IANA 登记，以防止重复。

客户端口号：数值为 49152 ∽ 65535，留给客户进程选择暂时使用。当服务器进程收到客户进程的报文时，就可知道客户进程所使用的端口号。通信结束后，这个端口号可供其他客户进程以后使用。

5.13 一个用户数据报 UDP 的数据字段为 8192 字节。在链路层使用以太网传送。试问应当划分为几个 IP 数据报片？说明每一个 IP 数据报片的数据字段的长度。

答：一个用户数据报 UDP 的数据字段为 8192 字节，其用户数据报的长度为：（8192+8）资金额=8200 字节。IP 数据报的长度为：（8200+20）字节=8220 字节。若在链路层使用以太网传送，由于以太网的帧中的数据段长度最长为 1500 字节，因此应当划分为 6（8220/1500 取整）个 IP 数据报片。其前 5 个 IP 数据报片的数据字段的长度分别为 1500 个字节，最后一个 IP 数据报片的数据字段的长度为 720 个字节。

5.14　一个 UDP 用户数据报的首部的十六进制表示是：06 32 00 45 00 1C E2 17。试求源端口、

目的端口、用户数据报的总长度、数据部分长度。这个数据报是从客户发送给服务器，还是从服务器发送给客户？使用 UDP 的这个服务器程序是什么？

答：一个 UDP 用户数据报的首部的十六进制表示是：06 32 00 45 00 1C E2 17，根据 UDP 用户数据报首部的规定，其源端口号为 06 32，目的端口号为 00 45，用户数据报的总长度为 00 1C，数据部分长度为 00 14。

由于其源端口号为 06 32，目的端口号为 00 45，将其转换成十进制后，源端口号为 1585，目的端口号为 69。以此可见这个数据报是从客户发送给服务器。使用 UDP 的这个服务器程序是 TFTP。

5.15 使用 TCP 对实时话音数据的传输有没有什么问题？使用 UDP 传送数据文件时会有什么问题？

答：**使用 TCP 对实时话音数据的传输的问题是：**由于 TCP 是一个面向连接的可靠的通信协议，其建立连接和释放连接，以及为保证可靠传输等需要采取比较的算法，由此带来了较大的通信开销，通信的实时性不能得到很好的保证。

使用 UDP 传送数据文件时的问题是：由于 UDP 是一个无连接的不可靠的，面向报文的通信协议，当传送的文件过大时，UDP 只能把其中的一部分打包在一个用户数据报中，而不得不丢弃一部分数据。再者在数据报的传输过程中也会产生错误，而产生错误后，接收端就会把它丢弃。

5.16 在停止等待协议中如果不是用编号是否可行？为什么？

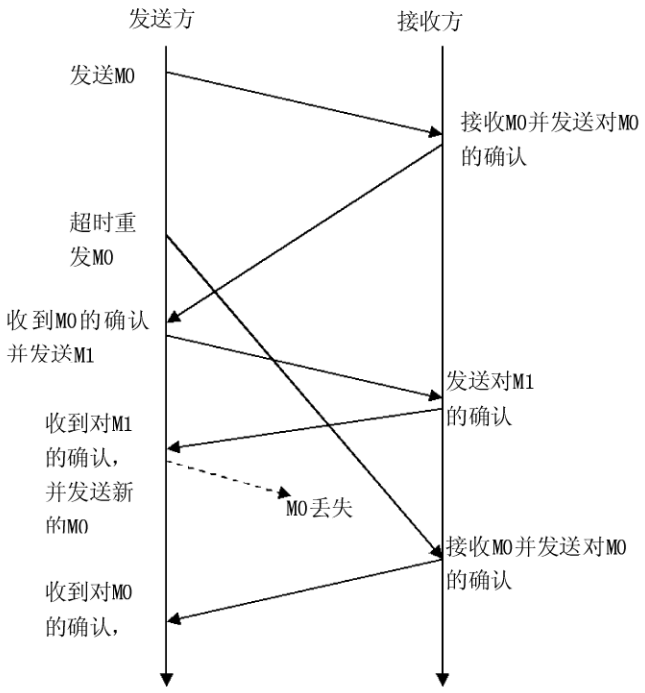
答：在停止等待协议中如果不是用编号是不可行的。因为，当发送方发送了一个数据分组后，就启动一个计时器并等待接收方的确认信息，如果在计时器到时，还未收到接收方的确认信息，就会重发刚发送过的数据分组。这样接收方有可能收到两个一样的数据分组，有数据分组中没有编号，接收方就会把它们当作两个不同的数据分组进行接收。

5.17 在停止等待协议中，如果收到重复报文时不予理睬是否可行？试举出具体例子说明理由。

答：在停止等待协议中，如果收到重复报文时不予理睬是不可行的。例如：当发送方发送了数据分组 M1 后，就启动一个计时器并等待接收方的确认信息。接收方在接收到第一个数据分组 M1 后就向发送方发送了一个确信息，但这个确认信在途中被丢失了，这样发送方在计时器到时，不会收到接收方的确认信息，就会重发刚发送过的数据分组 M1。当接收方收到两个一样的数据分组 M1 后，如果不向发送方发送确认信息，这样就会使发送永远得不到接收方关于数据分组 M1 的确认，因而就会继续不断发送数据分组 M1。

5.18 假定在运输层使用停止等待协议。发送方在发送报文段 M0 后，在设定的时间内未收到确认，于是重传 M0，但 M0 又迟迟不能到达接收方。不久，发送方收到了迟到的对 M0 的确认，于是就发送下一个报文段 M1，不久就收到了对 M1 的确认接着发送方发送新的报文段 M0，但这个新的 M0 在传送过程中丢失了。正巧，一开始就滞留在网络中的 M0 现在到达接收方。接收方无法分辨 M0 是旧的。于是就接受下 M0，并发送确认。显然，接收方后来收到的 MO 是重复的，协议失败了。试画出双方交换报文段的过程图。

答： 根据题意可画出收发双方交换报文段的过程示意图如下：



5.20 在连续 ARQ 协议中，若发送窗口等于 7，则发送端在开始时刻连续发送 7 个分组。因此，在每一个分组发出后，都要设置一个超时计数器。现在计算机里只有一个硬时钟。设这 7 个分组发出的时间分别为 T0,T1,T2,T3,T4,T5,T6，且 Tout 都一样大。试问如何实现这 7 个超时计数器。

分组M_i	分组发送时间T_i
M_0	T_0
$M_{1'}$	$T_{1'}$
$M_{0'}$	$T_{0'}$
M_2	T_2
M_4	T_4
M_5	T_5
M_6	T_6

答：在计算机只有一个硬时钟的情况下，可设置一个分组发送时间缓冲区，用于记录分组的发送时间，如右下表所示：发送方在连续发送完 7 个分组后，就不断的取出硬件时钟时间 T，并就算 T-Ti(i=0,1,2,…,6),然后用计算的结果与 Tout 进行比较，若大于 Tout，还能没有收到对应的分组 Mi 的确认，就说明该分组发生了超时。

5.21 假定使用连续 ARQ 协议，发送窗口大小是 3，而序号范围是[0，15]，而传输媒体保证接收方能按序收到分组。在某一时刻，假定接收方下一个期望收到的序号是 5，试问：

（1）在发送方的发送窗口中可能出现的序号组合有哪些种？

（2）在接收方已经发送出的，但在网络中的确认分组可能有哪些？说明这些确认分组是用

来确认那些序号的分组。

答：（1）在发送方的发送窗口中可能出现的序号组合有(2，3，4)、(3，4，5) 或(4，5，6)。

（2）在接收方已经发送出的，但在网络中的确认分组可能有 2、3，它们分别用来确认分组 2 和分组 3 的。

5.22 主机 A 向主机 B 发送一个很长的文件，其长度为 L 字节。假定 TCP 使用的 MSS 为 1460 字节。

（1）在 TCP 的序号不重复使用的条件下，L 的最大值是多少?

（2）假定使用上面计算出的文件长度，而运输层、网络层和数据链路层所用的首部开销共 66 字节，链路的数据率为 10Mb/s，试求这个文件发送所需的最短时间。

答：**（1）在 TCP 的序号不重复使用的条件下，L 的最大值是 2^{32}-1=4284967296。**

（2）假定使用上面计算出的文件长度，而运输层、网络层和数据链路层所用的首部开销共 66 字节，链路的数据率为 10Mb/s，则这个文件发送所需的最短时间是：((66×4284967296/1460)+ 4284967296)/10^7=448s。

5.23 主机 A 向主机 B 连续发送了两个 TCP 报文段，其序号分别是 70 和 100。试问：

（1）第一个报文段携带了多少个字节的数据？

（2）主机 B 收到第一个报文段后发回的确认号是多少？

（3）如果 A 发送的第二个报文段后发回的确认号是 180，试问 A 发送的第二个报文段中的数据有多少个字节？

（4）如果 B 收到第一个报文段丢失了，但第二个报文段到达恶劣 B。B 在第二个报文段到达后向 A 发回确认。试问这个确认号是多少？

答：（1）第一个报文段携带了(100-70)=30 字节的数据。

（2）主机 B 收到第一个报文段后发回的确认号是 71。

（3）如果 B 收到第二个报文段后发回的确认号是 180，试问 A 发送的第二个报文段中的数据有：9180-100）= 80 个字节。

（4）如果 A 发送的第一个报文段丢失了，但第二个报文段到达了 B。B 在第二个报文段到达后向 A 发回确认，这个确认号应是 70。

5.26 为什么在 TCP 首部中有一个首部长度字段，而在 UDP 的首部中就没有这个字段？

答：这是因为在 TCP 首部的长度中含有一个可选的字段，其长度是不定的，为了便于区别一个报文段的首部和数据部分，必须加入一个 TCP 首部的长度字段。而在 UDP 中，由于其用户数据报的首部长度是固定的（8 个字节），因此没有必要在其用户数据报中进行说明。

5.27 一个 TCP 报文段的数据部分最多为多少个字节？为什么？如果用户要传送到数据的字节长度超过 TCP 报文段中的序号字段所允许的的最大序号，问还能否用 TCP 来传送？

答：一个 TCP 报文段的数据部分最多为：（2^{16}-1）=64K 字节。这是因为发送方发送的报文段的最大数据长度有其最大的窗口值确定。在 TCP 中，规定最大的窗口值为 2^{16}-1。

如果用户要传送的数据字节长度超过 TCP 报文段中的序号字段所允许的的最大序号，一般不能用 TCP 来传

送。

5.28　主机 A 向主机 B 发送 TCP 报文段，首部的端口号是 M，而目的端口号是 N。当 B 向 A 发送回信时，其 TCP 报文段的首部中的源端口号和目的端口号分别是多少？

答：主机 A 向主机 B 发送 TCP 报文段，首部的端口号是 M，而目的端口号是 N。当 B 向 A 发送回信时，其 TCP 报文段的首部中的源端口号应是 N，目的端口号应是 M。

5.29 在使用 TCP 传送数据时，如果有一个确认报文段丢失了，也不一定会引起与该确认报文段对应的数据的重传。试说明理由。

答：在使用 TCP 传送数据时，如果有一个确认报文段丢失了，也不一定会引起与该确认报文段对应的数据的重传。出现这种情况是完全可能的，例如当发送方连续发送了两个以上的报文段（M0,M1,M2…）后，如果 B 收到了第一报文段 M0 后，就发送一个对 M0 的确认，则这个确认在网络中被丢失了，而 B 接着收到了第二报文段 M1，就立即发送一个对 M1 的确认。这个对 M1 的确认在第一报文段 M0 未超时前就到达了发送方，这时发送方可根据对 M1 的确认判定收方肯定已收到了报文段 M0 和 M1，并且 M0 和 M1 都是正确的。因为如果收方没有收到 M0 或 M0 不正确，则按照非选择性重发 TCP 协议的规定，收方不可能发送对 M1 的确认信息，而只能等待 M0 的重发。

5.30 设 TCP 使用的最大窗口为 65535 字节，而传输信道不产生差错，带宽也不受限制。若报文段的平往返时间为 20ms，问所能得到的最大吞吐量是多少？

答：所能得到的最大吞吐量是：65535/20ms = 3276.75/ ms = 3276750b/s。

5.31 通信信道带宽为 1Gb/s，端到端传播时延为 10ms。TCP 的发送窗口为 65535 字节，问可能达到的最大吞吐量是多少？信道的利用率是多少？

答：通信信道带宽为 1Gb/s，端到端传播时延为 10ms。TCP 的发送窗口为 65535 字节，可能达到的最大吞吐量是：65535b/（(65535b×2）/1Gb/s+20ms）= 65535b/（131070b/1Gb/s+0.02s）

= 65535b/（1.31070×10^{-1}s+0.02s）= 65535b/ 0.02013107s = 3255415b/s。

信道的利用率是：(3255415b/s)/1Gb/s = 0.0033。

5.32 什么是 Karn 算法？在 TCP 的重传机制中，若不采用 Karn 算法，而是在收到确认时，都认为是对重传报文段的确认，那么由此得出的往返时间样本和重传时间都会偏小。试问：重传时间最后会减少到什么程度？

答：一般方法为：

第一次测量到一个往返时间 RTT ，以后每测量到一个新的 RTT 样本，就按下式重新计算一次加权平均往返时间 RTTS： 新的 RTTS = (1 – α)×(旧的 RTTS) + α ×(新的 RTT 样本)

式中，0α≤<1。RFC 2988 推荐的 α 值为 1/8，即 0.125。

TCP 根据计算的加权平均往返时间，再计算超时重传时间(RTO)，RTO = RTTS + 4 × RTTD ；RTTD 是 RTT 的偏差的加权平均值。RFC 2988 建议采用如下计算 RTTD：

第一次测量时，RTTD 值取为测量到的 RTT 样本值的一半。在以后的测量中，则使用下式计算加权平均的 RTTD： 新的 RTTD =(1–β)×(旧的 RTTD) + β×|RTTS–新的 RTT 样本|

β 是个小于 1 的系数，其推荐值是 1/4，即 0.25。

Karn 在计算平均往返时间 RTT 时规定：只要报文段重传了，就不采用其往返时间样本。因为报文段重传后，返回的确认不管是对第一次发送的报文段的确认，还是对重传的报文段的确认，都不能准确的反映网络的实际情况。这样得出的加权平均往返时间 RTTS 和超时重传时间 RTO 就较准确。

在 TCP 的重传机制中，若不采用 Karn 算法，而是在收到确认时，都认为是对重传报文段的确认，那么由此得出的往返时间样本和重传时间都会偏小。重传时间最后会减少到发送端会多次重复的发送一个分组。

5.33 假定在 TCP 连接建立时，发送方设定的超时重传时间 RTO=6 秒。

（1）当发送方收到对方的连接确认报文段时，测量出 RTT 的样本值为 1.5 秒。试计算现在的 RTO 值。

（2）当发送方发送数据报文段并收到对方的连接确认报文段时，测量出 RTT 的样本值为 2.5 秒。试计算现在的 RTO 值。

答：（1）当发送方收到对方的连接确认报文段时，测量出 RTT 的样本值为 1.5 秒。则现在的 RTO 值可以由下面的计算得出。

新的 RTTS = (1 − 0.125)× 6 + 0.125 × 1.5 = 5.4375（s）；

新的 RTTD =(1−0.25)×(0) + 0.25 ×|5.4375 − 1.5| = 0.984375（s）；

RTO = RTTS + 4 × RTTD = 5.4375 + 4 * 0.984375 = 9.375（s）。

（2）当发送方发送数据报文段并收到对方的连接确认报文段时，测量出 RTT 的样本值为 2.5 秒。则现在的 RTO 值可以由下面的计算得出。

新的 RTTS = (1 − 0.125)× 6 + 0.125 × 2.5 = 5.5625（s）；

新的 RTTD =(1−0.25)×(0) + 0.25 ×|5.5625 − 2.5| = 0.765625（s）；

RTO = RTTS + 4 × RTTD = 5.5625 + 4 * 0.765625 = 8.625（s）。

5.34 已知第一次测得的 TCP 的往返时间 RTT 是 30ms。接着收到了三个确认报文段用它们计算出的往返时间样本 RTT 分别是：26ms、32ms 和 24ms。设 α = 0.1，试计算每一次新的加权平均往返时间值 RTTS，并讨论所得出的结果。

答：已知第一次测得的 TCP 的往返时间 RTT 是 30ms。接着收到了三个确认报文段用它们计算出的往返时间样本 RTT 分别是：26ms、32ms 和 24ms。设 α = 0.1，

则根据三个确认报文段计算出的新的加权平均往返时间值 RTTS 分别为：

第一个新的 RTTS = (1 − 0.1)× 30 + 0.1 × 26 = 29.6（ms）；

第二个新的 RTTS = (1 − 0.1)× 30 + 0.1 × 32 = 30.2（ms）；

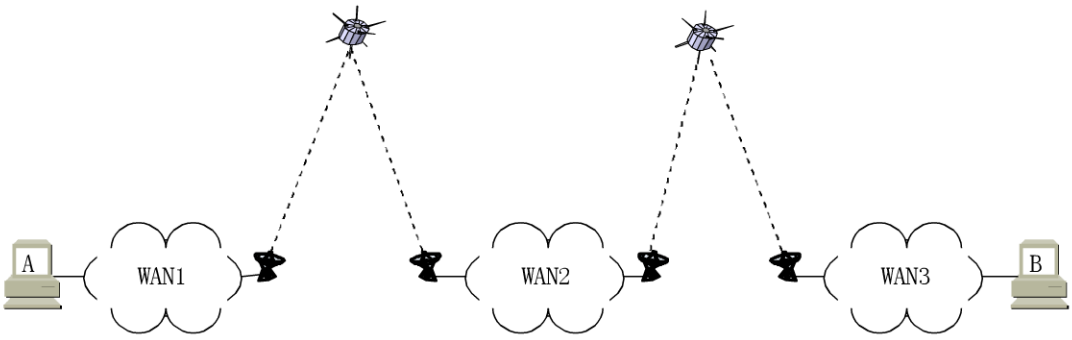
第三个新的 RTTS = (1 − 0.1)× 30 + 0.1 × 24 = 29.4（ms）。

由上面的计算结果可看出：由于新的样本值的加权系数 α = 0.1，相对于旧的样本值的加权系数 0.9，要小的多，因此尽管新的样本值 波动较大，但对计算结果的影响并不大。

5.35 试计算一个包含五段链路的运输连接的单程端到端时延。其中有两段是卫星链路，三段是广域网链路。每条卫星链路由上行和下行链路组成。可以取卫星上下行的传播时延为 250ms。每

一个广域网的范围为 1500km，其传播时延可按 150000km/s 来计算。各数据链路的速率为 48kb/s，帧长为 960 位。

答：根据题意，可画出如下网络示意图。



由示意图可见，建立一个包含五段链路的运输连接的单程端到端时延应包括，连接帧的两端的发送和接收时延、在中转通信结点的帧的转发时延和通过五个网络段的传播时延。下面对各部分的实验进行分别计算。

（1）连接帧的两端的发送和接收时延 = (960b / 48000b/s)×2 = 0.04 s

（2）连接帧在中转通信结点中转发时延 = (960b / 48000b/s)×2 ×4 = 0.16 s

（3）通过五个网络段的传播时延 = (1500km / 150000km/s)×3 + 250ms = 0.03s + 0.25s = 0.28s

则建立一个包含五段链路的运输连接的单程端到端时延 = 0.04 s + 0.16 s + 0.28s = 0.48s。

5.37 在 TCP 的拥塞控制中，什么是慢开始、拥塞避免快重传和快恢复算法？这些算法都起什么作用？“乘法减少”和“加法增大”各用在什么情况下？

答：慢开始拥塞控制的具体做法如下：

在主机开始发送报文段时可先设置 cwnd = 1，即设置为一个最大报文段 MSS 的数值。

当收到一个对新的报文段的确认后，将 cwnd 加 1，即增加一个 MSS 的数值。

发送方每收到一个新报文段的确认就将 cwnd 加 1。

每经过一个传输轮次，拥塞窗口 cwnd 就加倍。

拥塞避免算法是对慢开始算法的一种改进，其思路是让拥塞窗口 cwnd 缓慢地增大，即每经过一个往返时间 RTT 就把发送方的拥塞窗口 cwnd 加 1，而不是加倍。

无论慢开始处理还是在拥塞避免处理，只要发送方判断网络出现拥塞，就要把慢开始门限（ssthresh）设置为出现拥塞时的发送方窗口值的一半（但不能小于 2）。然后把拥塞窗口 cwnd 重新设置为 1，执行慢开始算法。这样做的目的就是要迅速减少主机发送到网络中的分组数，使得发生拥塞的路由器有足够时间把队列中积压的分组处理完毕。

快重传算法规定：接收方每收到一个失序的报文段后就立即发出重复确认，而不要等待自己发送数据时才进行捎带确认。发送方只要一连收到三个重复的确认，就应当立即重传对方尚未收到的报文段，而不必继续等待重传计时器到期。这样可提高整个网络的吞吐量。

与快重传配合使用的还有快恢复算法。快恢复算法规定：当发送方连续的收到三个重复的确认时，发送方可认为网络很可能没有发生拥塞，就执行“乘法减小”算法，将慢开始门限 ssthresh 减半。此后就执行拥塞

避免算法（加法增大”），使拥塞窗口缓慢地线性增大。

5.40 TCP 在进行流量控制时是以分组的丢失作为拥塞的标志。有没有不是因拥塞而引起的分组丢失的情况？如有，试举出三种情况。

答：**TCP 在进行流量控制时是以分组的丢失作为拥塞的标志，也有不是因拥塞而引起的分组丢失的情况。如：**

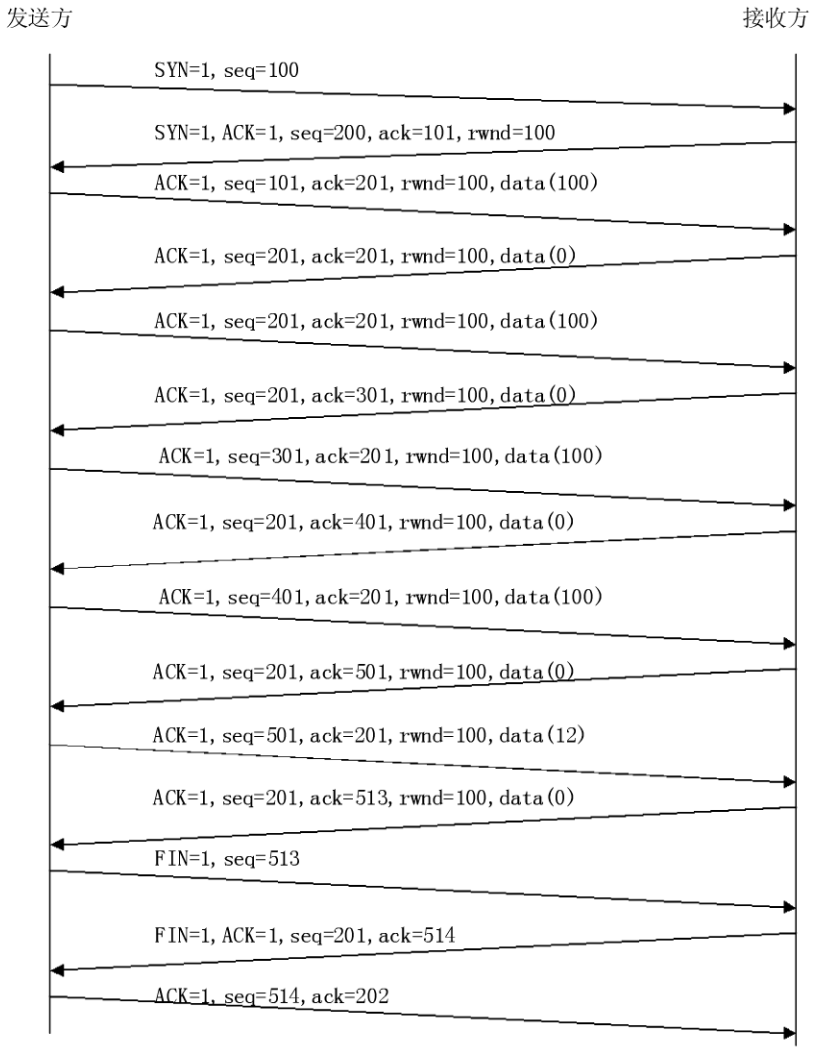
（1）当通信的物理链路因各种原因，而造成中断时，发送的分组就会中途丢失。

（2）当通信的物理链路收到干扰而造成分组中的路径控制信息，特别是目的地址的错乱时，发送的分组就会中途丢失。

（3）当采用的路由算法不能及时的更新路由表中的路由信息时，传送的数据分组若利用这些路由表中过时的路由信息时，就不能把分组传送到目的端，网络而不得不把它丢失。

5.41 用 TCP 传输 512 字节的数据。设窗口为 100 字节，而 TCP 报文段每次也是传送 100 字节的数据。再假定发送方和接收方的起始序号分别为 100 和 200，试画出其工作示意图（包括连接建立和释放）。

答：根据题意和用 TCP 传输原理，其工作示意图如右图：



5.45 解释为什么突然释放运输连接就可能会丢失用户数据，而使用 TCP 连接释放方法就可保证不丢失数据。

答：如果 A 向 B 发送了释放报文段后，就突然释放运输连接，则这个报文段可能会丢失，而 B 就不可能接收到这个释放报文段。当 B 还有数据继续向 A 发送时，由于 A 已经关闭了运输连接，则 B 向 A 发送这些数据报文段就会丢失。而使用 TCP 连接释放方法后，即使 A 向 B 发送的释放报文段丢失后，由于没有收到 B 的反馈信息，就不会释放运输连接。因此 B 向 A 发送的数据报文段，在到达 A 后，A 仍能进行接收。