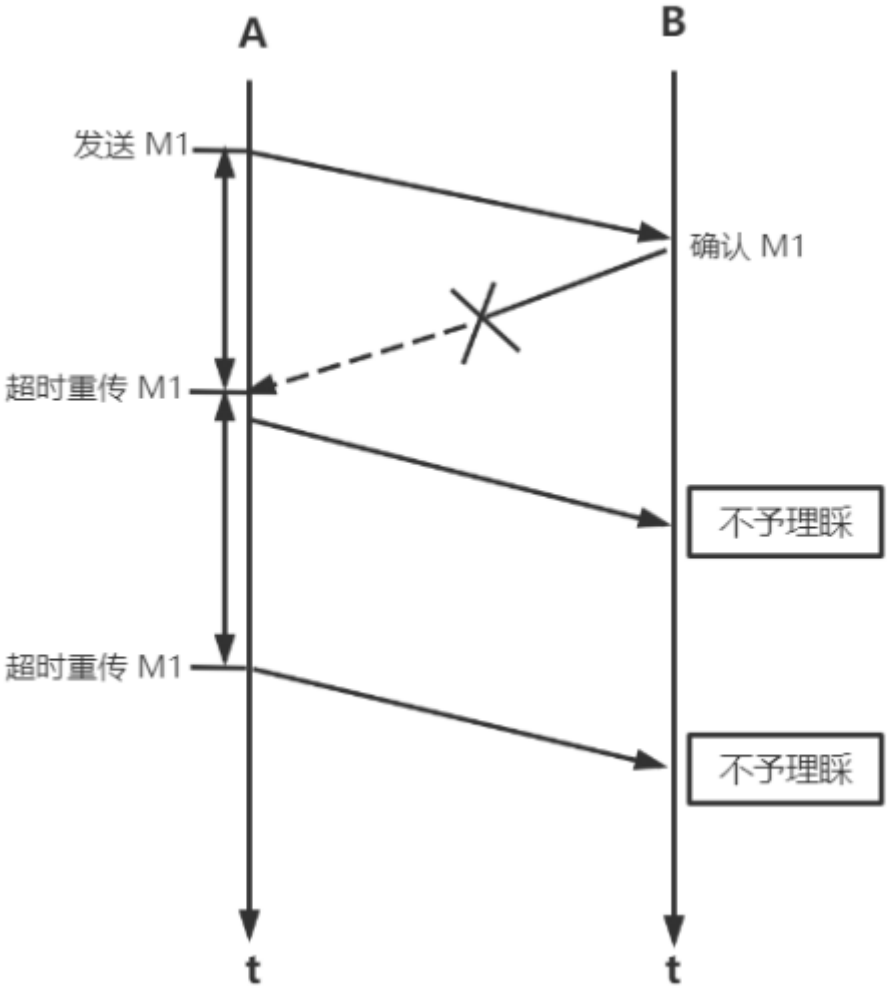
第五章习题

5-08. 为什么说**UDP**是面向报文的，而**TCP**是面向字节流的？
发送方 UDP 对应用程序交下来的报文，在添加首部后就向下交付 IP 层。UDP 对应用层交下来的报文，既不合并，也不拆分，而是保留这些报文的边界。接收方 UDP 对 IP 层交上来的 UDP 用户数据报，在去除首部后就原封不动地交付上层的应用进程，一次交付一个完整的报文。发送方TCP对应用程序交下来的报文数据块，视为无结构的字节流（无边界约束，课分拆/合并），但维持各字节。

5-17. 在停止等待协议中，如果收到重复的报文段时不予理睬（即悄悄地丢弃它而其他什么也没做）是否可行？试举出具体的例子说明理由。



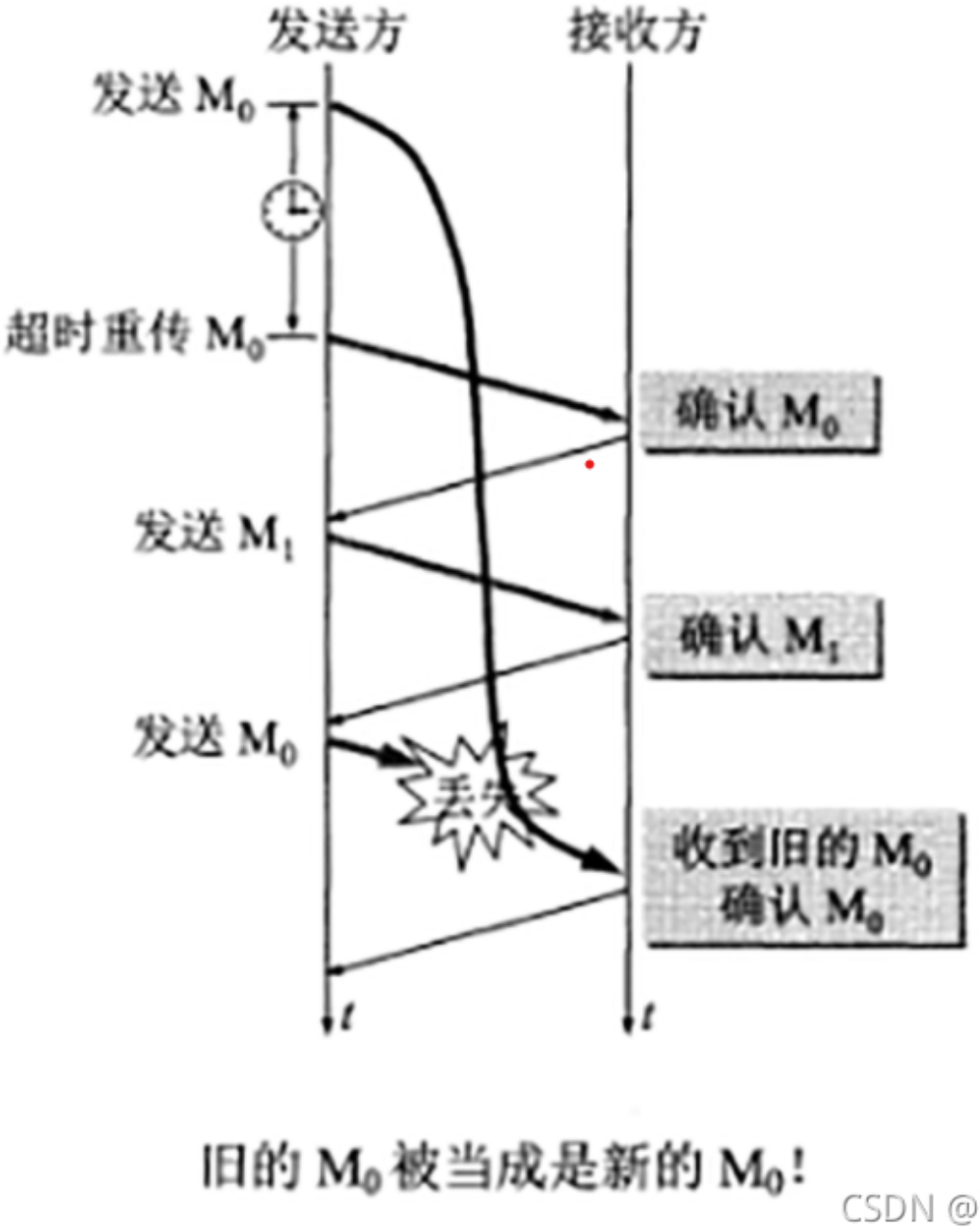
A 发送报文段 M1，B 收到后发送确认，但这个确认丢失了。

A 发送报文段 M1，B 收到后不予理解。这就导致 A 再次超时重传报文段 M1。

B 收到重复的报文段都不予理睬，A 就一直超时重传报文段 M1，将会导致 A 的资源浪费呀~。

可见，如果收到重复的报文段时不予理睬是不行的。

5-18. 假定在运输层使用停止等待协议。发送发在发送报文段**M0**后再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。于是收下**M0**，并发送确认。显然，接收方后来收到的**M0**是重复的，协议失败了。试画出类似于图**5-9**所示的双方交换报文段的过程。



5-21. 使用连续 **ARQ** 协议中，发送窗口大小是 **3**，而序列范围 **[0, 15]**，而传输媒体保证在接收方能够按序收到分组。在某时刻，接收方，下一个期望收到序号是 **5**。试问：
（**1**）在发送方的发送窗口中可能有出现的序号组合有哪几种？
（**2**）接收方已经发送出去的、但在网络中（即还未到达发送方）的确认分组可能有哪些？说明这些确认分组是用来确认哪些序号的分组。

（1）在接收方，下一个期望收到的序号是 5。这表明序号到 4 为止的分组都已收到。若这些确认都已到达发送方，则发送窗口最靠前，其范围是 [5, 7]。
假定所有的分组都丢失了，发送方都没有收到这些确认。这时，发送窗口最靠后，应为 [2, 4]。因此，发送窗口可以是 [2, 4]，[3, 5]，[4, 6]，[5, 7] 中的任何一个。

（2）接收方期望收到的序号 5 的分组，说明序号为 2，3，4 的分组都已收到，并且发送了确认。对序号为 1 的分组的确认肯定被发送方收到了，否则发送方不可能发送 4 号分组。可见，对序号为 2，3，4 的分组的确认有可能仍滞留在网络中。这些确认是用来确认序号为 2，3，4 的分组的。

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** 有 **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文件长度，而运输层、网络层和数据链路层所使用的首部开销共 **66** 字节，链路的数据率为 **10 Mb/s**，试求这个文件所需的最短发送时间

（ 1 ） L 的最大值是 2^{32}=4GB 字节

因为TCP报文 序号 字段——占 4 字节（32位）。TCP 连接中传送的数据流中的每一个字节都编上一个序号。序号字段的值则指的是本报文段所发送的数据的第一个字节的序号。 L是让求文件的长度，即TCP报文 序号 字段最多可以表示的字节的总数.

（2）发送帧数等于数据字节/每帧的数据字节= 2^{32} /1460=2941758.422，需要发送 2941759 个帧。帧首部的开销是 66×2941759 = 194156094 字节。发送的总字节数 = 2^{32} +194156094 = 4489123390字节。
数据率 10 Mbilt/s = 1.25 MB/s = 1250000 字节/秒。
发送 4489123390 字节需时间为：4489123390/1250000 =3591.3 秒。
即 59.85 分，约 1 小时。

5-27. 一个 **TCP** 报文段的数据部分最多为多少个字节？为什么？如果用户要传送的数据的字节长度超过 **TCP** 报文字段中的序号字段可能编出的最大序号，问还能否用 **TCP** 来传送？

（1） 一个 TCP 报文段段的数据部分最多为 65495 字节，此数据部分加上 TCP 首部的 20 字节，再加上 IP 首部的 20 字节，正好是 IP 数据报的最大长度 65535。（当然，若 IP 首部包含了选择，则 IP 首部长度超过20字节，这时 TCP 报文段的数据部分的长度将小于 65495 字节。）

（2）如果数据的字节长度超过 TCP 报文段中的序号字段可能编出的最大序号，仍可用 TCP 传送。编号用完后可重复使用。但应设法保证不出现编号的混乱。

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延样本和重传时间都会偏小。试问：重传时间最后会减小到什么程度**?**

Karn 算法允许 TCP 能够区分开有效的和无效的往返时间样本，从而改进往返时间的估算。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。TCP 发送了报文段后，没有收到确认，于是超时重传报文段。但刚刚重传了报文段后，马上就收到了确认。显然，这个确认是对原来发送的报文段的确认。

但是，根据题意，我们就认为这个确认是对重传的报文段的确认。这样得出的往返时间就会很小。这样的往返时间最后甚至可以减小到很接近于零、

因此，上述的这种做法是不可取的。

5-35. 试计算一个包括**5**段链路的运输连接的单程端到端时延。**5**段链路程中有**2**段是卫星链路，有**3**段是广域网链路。每条卫星链路又由上行链路和下行链路两部分组成。可以取这两部分的传播时延之和为**250ms**。每一个广域网的范围为**1500km**，其传播时延可按**150000km** / **s**来计算。各数据链路速率为**48kb** / **s**，帧长为**960**位。

	WAN	WAN	WAN	卫星网	卫星网	合计
传播时延	10 ms	10 ms	10 ms	250 ms	250 ms	530 ms
发送时延	20 ms	20 ms	20 ms	20 ms	20 ms	100 ms

5 段链路的传播时延=250 * 2+（1500/150000）*3*1000=530ms

5 段链路的发送时延=960/（48 * 1000）*5*1000=100ms

所以5段链路单程端到端时延=530+100=630ms

5-36. 重复 **35** 题，但假定其中的一个陆地上的广域网的传输时延为 **150 ms**。

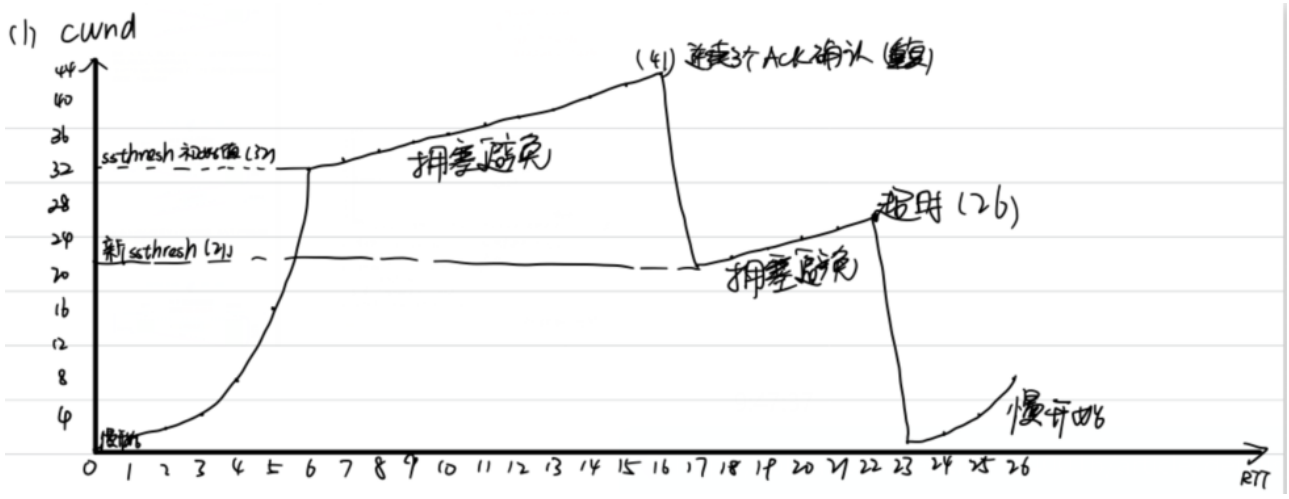
	WAN	WAN	WAN	卫星网	卫星网	合计
传播时延	10 ms	10 ms	10 ms	250 ms	250 ms	530 ms
发送时延	150 ms	20 ms	20 ms	20 ms	20 ms	230 ms

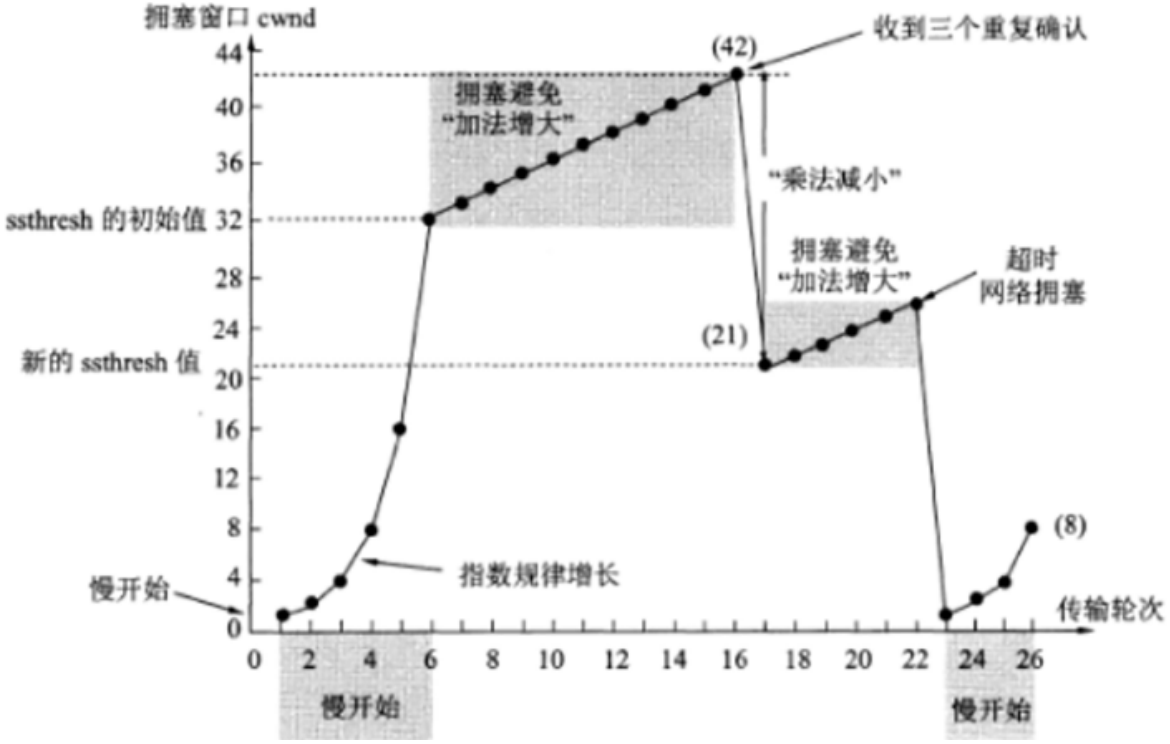
可见总共需时：760 ms

5-39. **TCP** 的拥塞窗口 **cwnd** 大小与传输轮次 **n** 的关系如表 **5.1** 所示：
（**1**）试画出如教材中图 **5-25** 所示的拥塞窗口与传输轮次的关系曲线。
（**2**）指明 **TCP** 工作在慢开始阶段的时间间隔。
（**3**）指明 **TCP** 工作在拥塞避免阶段的时间间隔。
（**4**）在第 **16** 轮次和第 **22** 轮次之后发送方是通过收到三个重复的确认还是通过超时检测到丢失了报文段？
（**5**）在第 **1** 轮次，第 **18** 轮次和第 **24** 轮次发送时，门限 **ssthresh** 分别被设置为多大？
（**6**）在第几轮次发送出第 **70** 个报文段？
（**7**）假定在第 **26** 轮次之后收到了三个重复的确认，因而检测出了报文段的丢失，那么拥塞窗口 **cwnd** 和门限 **ssthresh** 应设置为多大？

cwnd	1	2	4	8	16	32	33	34	35	36	37	38	39
n	1	2	3	4	5	6	7	8	9	10	11	12	13
cwnd	40	41	42	21	22	23	24	25	26	1	2	4	8
n	14	15	16	17	18	19	20	21	22	23	24	25	26

（1） 拥塞窗口与传输轮次的关系曲线如下图所示：





（2）慢开始时间间隔：[1, 6] 和 [23, 26]

（3）拥塞避免时间间隔：[6, 16] 和 [17, 22]

（4）在第 16 轮次之后发送方通过收到三个重复的确认，检测到丢失了报文段，因为下一个轮次的拥塞窗口减半了。

在第 22 轮次之后发送方通过超时，检测到丢失了报文段，因为下一个轮次的拥塞窗口下降到 1了。

（5）在第 1 轮次发送时，门限 ssthresh 被设置为 32，因为从第 6 轮次起就进入了拥塞避免状态，拥塞窗口每个轮次加 1。

在第 18 轮次发送时，门限 ssthresh 被设置为发生拥塞时拥塞窗口 42 的一半，即 21。

在第 24 轮次发送时，门限 ssthresh 被设置为发生拥塞时拥塞窗口 26 的一半，即 13。

（6）第 1 轮次发送报文段 1。（cwnd = 1）

第 2 轮次发送报文段 2， 3。（cwnd = 2）

第 3 轮次发送报文段 4 ~ 7。（cwnd = 4）

第 4 轮次发送报文段 8 ~ 15。（cwnd = 8）

第 5 轮次发送报文段 16 ~ 31。（cwnd = 16）

第 6 轮次发送报文段 32 ~ 63。（cwnd = 32）
第 7 轮次发送报文段 64 ~ 96。（cwnd = 33）
因此第 70 报文段在第 7 轮次发送出。
（7）检测出了报文段的丢失时拥塞窗口 cwnd 是 8，因此拥塞窗口 cwnd 的数值应当减半，等于 4，而门限 ssthresh 应设置为检测出报文段丢失时的拥塞窗口 8 的一半，即 4。

5-41. 用 **TCP** 传送 **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。再设发送端和接收端的起始序号分别选为 **100** 和 **200**，试画出类似于教材中图 **5-31** 的工作示意图。从连接建立阶段到连接释放都要画上

要传送的 512 B 的数据必须划分为 6 个报文段传送，前 5 个报文段各 100 B，最后一个报文段传送 12 B。下图是双方交互的示意图。下面进行简单的解释：

【----- 进行三报文握手 -----】
报文段 **#1**：A 发起主动打开，发送 SYN 报文段，除以 SYN-SENT 状态，并选择初始序号 seq = 100。B 处于 LISTEN 状态。
报文段 **#2**：B 确认 A 的 SYN 报文段，因此 ack = 101（是 A 的初始序号加 1）。B选择初始序号 seq = 200。B 进入到 SYN-RCVD 状态。
报文段 **#3**：A 发送 ACk 报文段来确认报文段 #2，ack = 201（是 B 的初始序号加 1）。A 没有在这个报文段中放入数据。因为 SYN 报文段 #1 消耗了一个序号，因此报文段 #3 的序号是 seq = 101。这样，A 和 B 都进入了 ESTABLISHED 状态。
【----- 三报文握手完成 -----】

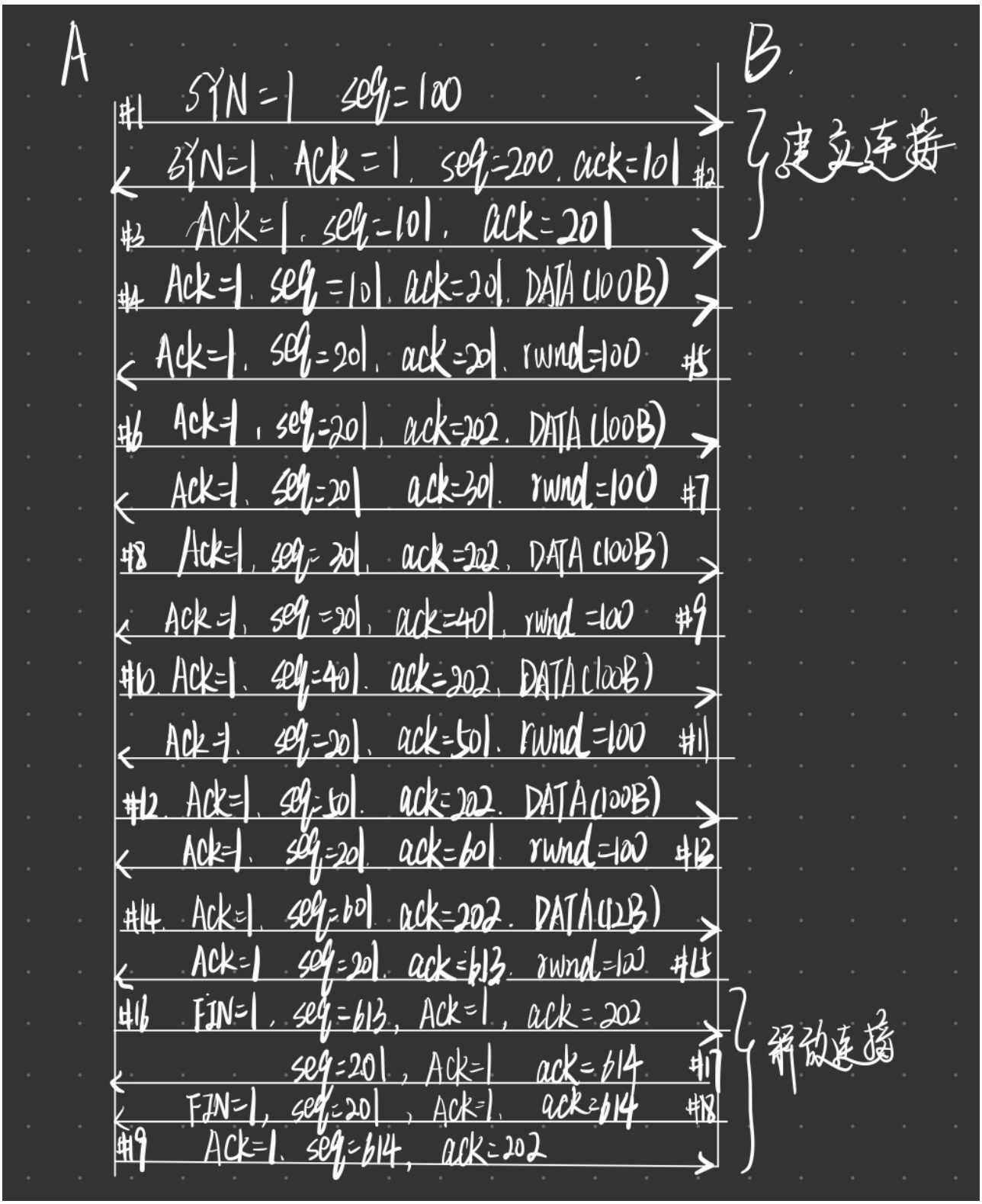
【----- 开始数据传送 -----】
报文段 **#4**：A 发送 100 字节的数据。报文段 #3 是确认报文段，没有数据发送，报文段 #3 并不消耗序号，因此报文段 #4 的序号仍然是 seq = 101。A 在发送数据的同时，还确认 B 的报文段 #2，因此 ack = 201。
报文段 **#5**：B 确认 A 的报文段 #4。由于收到了从序号 101 到 200 共 100 字节的数据，因此在报文段 #5 中，ack = 201（所期望收到的下一个数据字节的序号）。B 发送的 SYN 报文段 #2 消耗了一个序号，因此报文段 #5 的序号是 seq = 201，比报文段 #2 的序号多了一个序号。在这个报文段中，B 给出了接收窗口 rwnd = 100。
从报文段 **#6** 到报文段 **# 13** 都不需要更多的解释。到此为止，A 已经发送了 500 字节 的数据。值得注意的是，B 发送的所有确认报文都不消耗序号，其序号都是 seq = 201。
报文段 **#14**：A 发送最后 12 字节的数据，报文段 #14 的序号是 seq = 601。
报文段 **#15**：B 发送对报文段 #14 的确认。B 收到从序号 601 到 602 共 12 字节的数据。因此，报文段 #15 的确认号是 ack = 613（所期望收到的下一个数据字节的序号）。
需要注意的是，从报文段 #5 一直到 报文段 #15，B 一共发送了 6 个确认，都不消耗序号，因此 B 发送的报文段 #15 的序号仍然和报文段 #5 的序号一样，即 seq = 201。
【-----数据传送完毕-----】
【-----进行四报文挥手------】
报文段 **#16**：A 发送 FIN 报文段。前面所发送的数据报文段 #14 已经用掉了序号 601 到 612，因此报文段 #16 序号是 seq = 613。A 进入 FIN-WAIT-1 状态。报文段 #16 的确认号 ack = 202。

报文段 **#17**：B发送确认报文段，确认号为 614，进入 CLOSE-WAIT 状态。由于确认报文段不消耗序号，因此报文段 #17 的序号仍然和报文段 #15 的一样，即 seq = 201

报文段 **#18**：B 没有数据要发送，就发送 FIN 报文段 #18，其序号仍然是 seq = 201。这个 FIN 报文会消耗一个报文。

报文段 **#19**：A 发送最后的确认报文段。报文段 #16 的序号是 613，已经消耗掉了。因此，现在的序号是 seq = 614。但这个确认报文段并不消耗序号。

【-----四报文挥手结束-----】

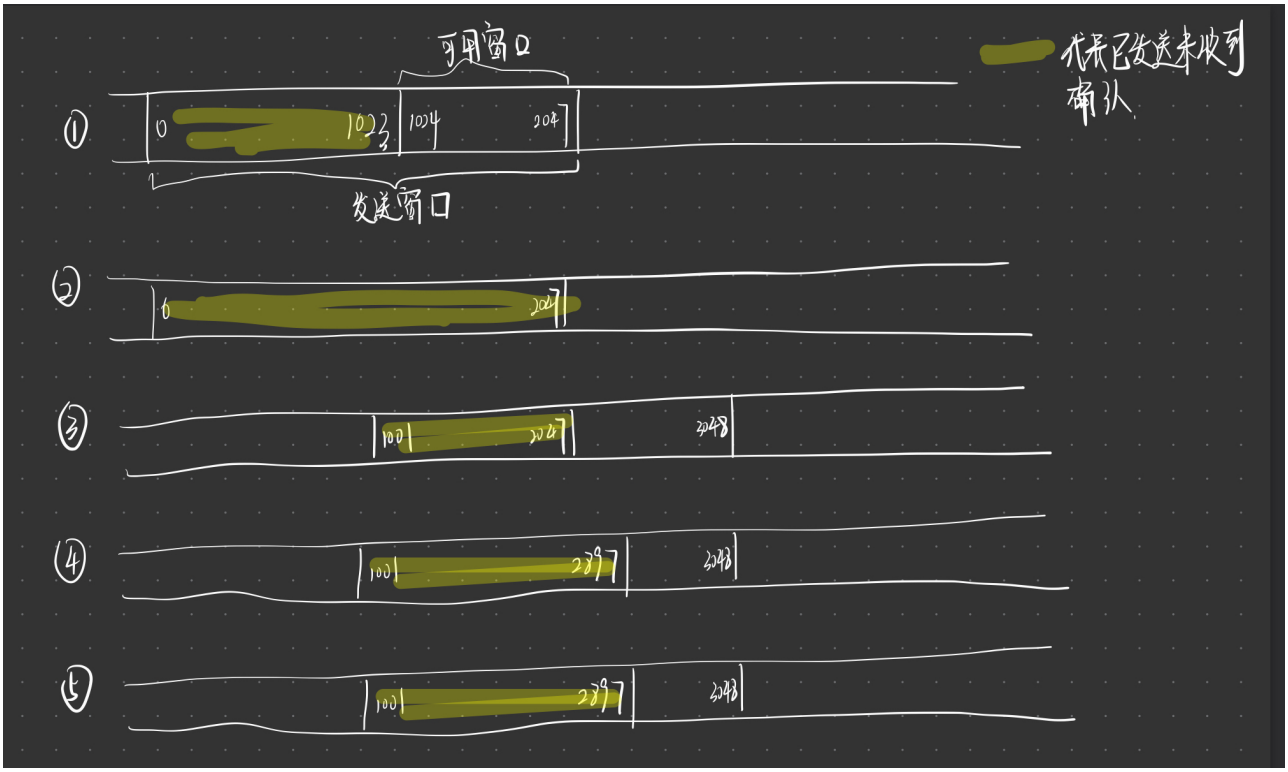


5-46. 试用具体例子说明为什么在运输连接建立时要使用三次握手。说明如不这样做可能会出现什么情况。
3 次握手完成两个重要的功能，既要双方做好发送数据的准备工作（双方都知道彼此已准备好），也要允许双方就初始序列号进行协商，这个序列号在握手过程中被发送和确认。
假定 B 给 A 发送一个连接请求分组，A 收到了这个分组，并发送了确认应答分组。按照两次握手的协定，A 认为连接已经成功地建立了，可以开始发送数据分组。可是，B 在 A 的应答分组在传输中被丢失的情况下，将不知道 A 是否已准备好，不知道 A 建议什么样的序列号，B 甚至怀疑 A 是否收到自己的连接请求分组，在这种情况下，B 认为连接还未建立成功，将忽略 A 发来的任何数据分组，只等待连接确认应答分组。而 A 发出的分组超时后，重复发送同样的分组。这样就形成了死锁。

5-48 网络允许的最大报文段长度为 **128** 字节，序号用 **8** 位表示，报文段在网络中的寿命为 **30** 秒。求发送报文段的一方所能达到的最高数据率。

解：具有相同编号的报文段不应该同时在网络中传输，必须保证当序列号循环重复使用的时候，具有相同序列号的报文段已经从网络中消失。
序号用 8 位表示，序号范围是 $[0, 2^8 - 1]$，共 256 个序号。
报文段的寿命为 30 s，那么在 30 s 的时间内发送方发送的报文段的数目不能多于 255 个。
网络允许的最大报文段长度为 128 B，一共发送的比特数为 255 × 128 × 8 bit = 261120 bit。
每一条 TCP 连接所能达到的最高数据传输速率为 261120 bit / 30 s = 8.704 kbit/s。

5-61. 在本题中列出的 **8** 种情况下，画出发送窗口的变化。并标明可用窗口的位置。已知主机 **A** 要向主机 **B** 发送 **3 KB** 的数据。在 **TCP** 连接建立后，**A** 的发送窗口大小是 **2 KB**。**A** 的初始序号是 **0**。
(1) 一开始 **A** 发送 **1 KB**的数据。
(2) 接着 **A** 就一直发送数据，直到把发送窗口用完。
(3) 发送方 **A** 收到对第 **1000** 号字节的确认报文段。
(4) 发送方 **A** 再发送 **850 B**的数据。
(5) 发送方 **A** 收到 **ack = 900** 的确认报文段。
(6) 发送方 **A**收到对第 **2047** 号字节的确认报文段。
(7) 发送方 **A**把剩下的数据全部都发送完。
(8) 发送方 **A** 收到 **ack = 3072** 的确认报文段。



5-68. 在 **TCP** 的连接建立的三报文握手过程中，为什么第三个报文段不需要对方的确认？这会不会出现问题？

① 关于这个问题，还不能简单地用“是”或“否”来回答。
② 我们假定 A 是客户端，是发起 TCP 连接建立一方。现在假定三报文握手过程中的第三个报文段（也就是 A 发送的第二个报文段 —— 确认报文）丢失了，而 A 并不知道。这是，A 以为对方收到了这个报文段，以为 TCP 连接已经建立，于是就开始发送数据报文段给 B。
③ B 由于没有收到三报文握手中的最后一个报文段（A 发送的确认报文段），因此 B 就不能进入 TCP 的 ESTABLISHED 状态（“连接已建立”状态）。B 的这种状态可以叫做“半开连接”，即仅仅把 TCP 连接打开了一半。在这种状态下，B 虽然已经初始化了连接变量和缓存，但是不能接收数据。通常，B 在经过一段时间后（例如，一分钟后），如果还没有收到来自 A 的确认报文段，就终止这个半开连接状态，那么 A 就必须重新建立 TCP 连接。因

此，在这种情况下，第三个报文段（A 发送的第二个报文段）的丢失，就导致了 TCP 连接无法建立。

④ 但是，假定 A 在这段时间内，紧接着就发送了数据。我们知道，TCP 具有累计确认的功能。在 A 发送的数据报文段中，自己的序号也没有改变，仍然是和丢失的确认真的序号一样（丢失的那个确认帧不消耗序号），并且确认位 ACK = 1，确认号也是 B 的初始序号加 1。当 B 收到这个报文段后，从 TCP 的首部就可以知道，A 已确认了 B 刚才发送的 SYN + ACK 报文段，于是就进入了 ESTABLISHED 状态（“连接已建立” 状态）。接着，就接收 A 发送的数据。在这种情况下，A 丢失的第二个报文段对 TCP 的连接建立就没有影响。

⑤ 大家知道，A 在发送第二个报文段时，可以有两种选择：

（1）仅仅是确认而不携带数据，数据接着在后面发送。

（2）不仅是确认，而且携带上自己的数据。

⑥ 在第一种选择时，A 在下一个报文段发送自己的数据。但下一个报文的首部中仍然包括了对 B 的 SYN + ACK 报文段的确认，即和第二种选择发送的报文段一样。

⑦ 在第二种选择时，A 省略了单独发送一个确认报文段。

从这里也可以看出，A 发送的第二个仅仅是确认的报文段，是个可以省略的报文段，即使丢失了也无妨，只要下面紧接着就可以发送数据报文