5-08 为什么说 UDP 是面向报文的，而 TCP 是面向字节流的？

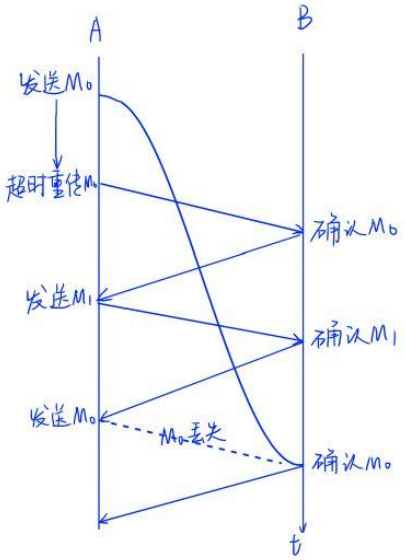
应用进程间使用 UDP 协议进行传输时，发送方的 UDP 对应用程序交下来的应用层**报文**，添加首部后就向下交付给网络层。在这过程中，既不合并，也不拆分，而是保留这些报文的边界，一次发送一个报文（即 UDP 数据报）。接收方的 UDP 收到网络层交付上来的 IP 数据报，去除首部后成为 UDP 用户数据报，直接交付上层的应用进程。因此说 UDP 是面向报文的。

应用进程间使用 TCP 协议进行传输时，虽然应用程序和 TCP 的交互是一次一个数据块，但 TCP 把它们仅仅看成是**一连串无结构的字节流**。TCP 并不关心应用进程一次把多长的报文发送到 TCP 的缓存中，而是根据接收方给出的窗口值和当前网络拥塞的程度来决定一个报文段应包含多少个字节。如果应用进程传送到 TCP 缓存的数据块太长，TCP 就可以把它划分短一些再传送。如果应用进程一次只发来一个字节，TCP 也可以等待积累有足够多的字节后再构成报文段发送出去。整个流程中，TCP 并不知道所传送的字节流的含义，也不保证接收方应用程序所收到的数据块和发送方应用程序所发出的数据块大小或数量相等。但 TCP 要保证接收方应用程序收到的字节流与发送方应用程序发出的字节流完全一样。

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不做）是否可行？试举出具体例子说明理由。

不可行。假设 A 向 B 发送报文后，B 发送给 A 的确认丢失，A 没有在计时器到期前收到 B 发送的确认。那么 A 就会重传数据报，如果 B 对重复收到的该数据报不予理睬，那么 A 就会不断重传该数据报，造成死循环，通信无法继续进行。

5-18 假定在运输层使用停止等待协议。发送方发送报文段 M_0 后在设定的时间内未收到确认，于是重传 M_0，但 M_0 又迟迟不能到达接收方。不久，发送方收到了迟到的对 M_0 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 M_0，但这个新的 M_0 在传送过程中丢失了。正巧，一开始就滞留在网络中的 M_0 现在到达接收方。接收方无法分辨 M_0 是旧的。于是收下 M_0，并发送确认。显然，接收方后来收到的 M_0 是重复的，协议失败了。
试画出类似千图 5-9 所示的双方交换报文段的过程。



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]，而传输媒体保证在接收方能够按序收到分组。在某一时刻，在接收方，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？

发送窗口大小为 3，因此发送方的发送窗口中一定有且只有 3 个元素。若接收方发送的确认号 5 被准确传送给发送方，那么窗口中序号为 5,6,7。考虑最坏情况，发送方没有收到接收方发送的确认号，并且发送前发送端窗口包含 2,3,4。

综上，发送方发送窗口中可能出现的序号组合为**2,3,4** 或 **3,4,5** 或 **4,5,6** 或 **5,6,7**。

(2)接收方已经发送出的、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。

若不采用累计确认，在网络中的确认分组可能有 2,3,4，分别用来确认 2,3,4 分组。

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)TCP 的序号不重复使用的条件下，L 的最大值是多少？

TCP 的序号范围是$[0,2^{32}-1]$，因此最多传送 2^{32} 个字节=4GB。因此 L_{max}=4GB。

(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 10 Mbit/s，试求这个文件所需的最短发送时间。（数据率为 10 Mbit/s 的 M 表示 10^6 而不是 2^{20}，一般只有数据大小时才采用二进制单位互化）

TCP 使用的 MSS 为 1460B，因此每次能够发送的最长数据部分为 1460B。4GB/1460B=2941758.4≈2941759 个数据报，因此底层传输的实际文件大小为 2^{32}+2941759*66=4489123390B。传输时间为 4489123390B/10 Mbit/s=3591.30s，大概需要 1h。

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？

TCP 数据报首部和 IP 数据报首部的固定长度为 20B，即最短需要 40B 的首部。而 IP 数据报的最大长度为 65535B，因此一个 TCP 报文段的数据部分最多有 65495 字节。

若用户要传送的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，可以使用 TCP 来传送。因为 TCP 报文段中的序号是循环使用的，一旦达到最大值 $2^{32}-1$ 后继续从 0 开始编号。因为 TCP 数据部分的最大上限为 65495B，远小于报文段的序号长度 2^{32}，因此不用担心循环使用中同一报文出现重复序号的情况。

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：重传时间最后会减小到什么程度？

TCP 协议规定发送方在规定时间内接收不到接收方返回的确认就认为发送数据丢失，并需要重传数据。但是重传时间的选择很难界定，选择不当会对传输效率造成很大影响。TCP 原先采用了一种自适应法，不断记录更新报文段的加权

平均往返时间 RTT_S 和超时重传时间 RTO。但是在重传机制下，重传后返回的确认难以识别为原先数据的确认还是重传数据的确认，因此对这种自适应算法影响较大。而 **Karn 算法**提出：在计算 RTT_S 时不计入重传报文段的 RTT，这样就能够得出较为准确的加权平均 RTT_S 和超时重传时间 RTO。

若不采用 Karn 算法，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 RTT_S 和 RTO 都会偏小。根据计算公式：

$$RTT_S = (1-\alpha) * RTT_S + \alpha * RTT$$

$$RTT_D = (1-\beta) * RTT_D + \beta * |RTT_S - RTT|$$

$$RTO = RTT_S + 4 * RTT_D$$

其中 RTT 为实际传输时间，如果收到确认时都认为是对重传报文段的确认，那么 RTT 就有可能比真实值小，因此 RTT_S 就会越来越小，收敛到往返时间。而 RTT_D 因此也会被计小，最终 RTO 收敛到往返时间。

5-35 试计算一个包括五段链路的运输连接的单程端到端时延。五段链路中有两段是卫星链路，有三段是广域网链路。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 250 ms。每一个广域网的范围为 1500km，其传播时延可按 150000 km/s 来计算。各数据链路速率为 48 kbit/s，帧长为 960bit。

每段广域网的传播时延=1500km/150000km/s=10ms

发送时延=960bit/48kbit/s=20ms

综上，单程端到端时延=10*3+20*5+250*2=630ms

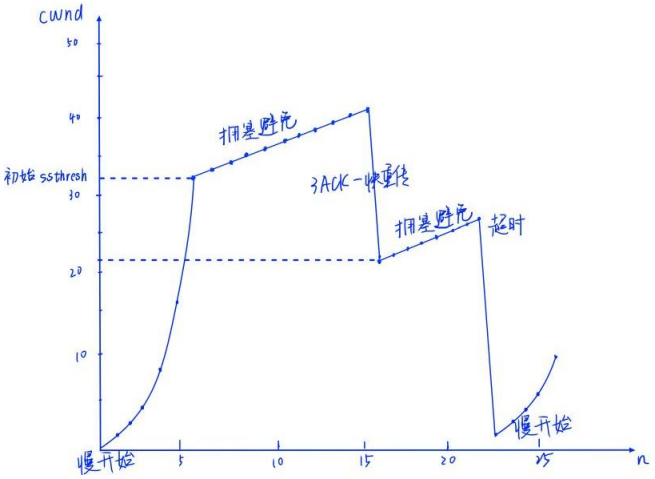
5-36 重复 5-35 题，但假定其中的一个陆地上的广域网的传输时延为 150ms。（传输时延是发送时延，而不是传播时延）

单程端到端时延=150+20*4+10*3+250*2=760ms

5-39 TCP 的拥塞窗口 cwnd 大小与传输轮次 n 的关系如下所示：

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)试画出如图 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。

传输轮次 1~6 和 23~26。

(3)指明 TCP 工作在拥塞避免阶段的时间间隔。

传输轮次 6~16 和 17~22。

(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?

第 16 轮次是三个重复确认；第 22 轮次是超时。

(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?

分别为 32,21 和 13。

(6)在第几轮次发送出第 70 个报文段?

1+2+4+8+16+32=63

1+2+4+8+16+32+33=96

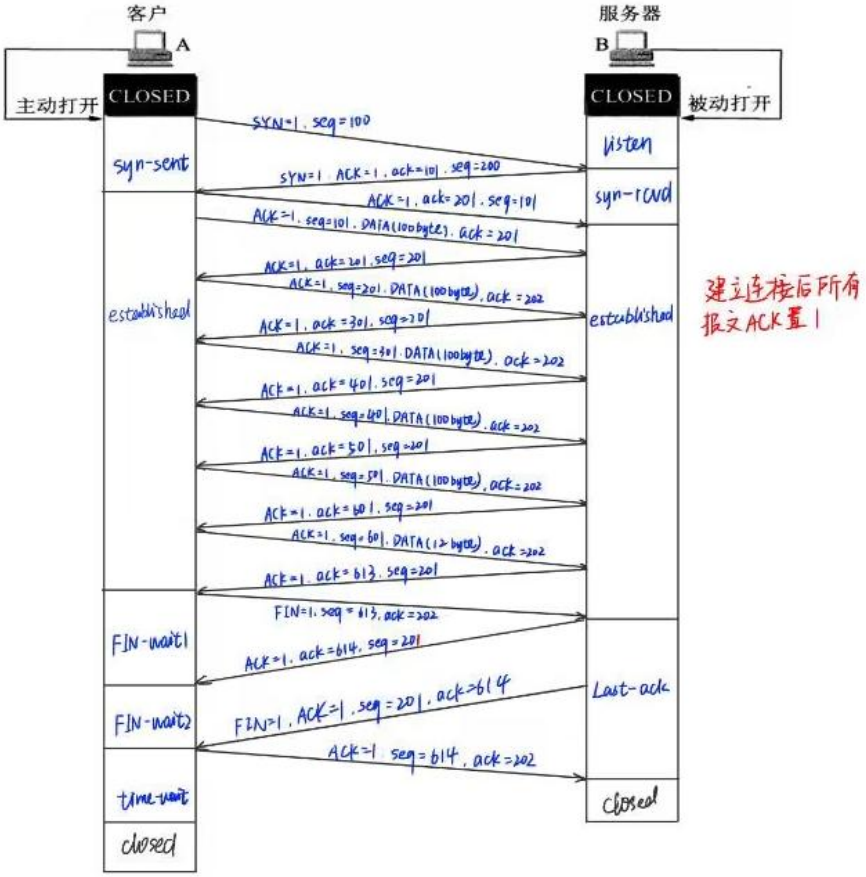
所以第 70 个报文段在第 7 轮次发出。

(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=cwnd/2=4

ssthresh=cwnd=4

5-41 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。再设发送方和接收方的起始序号分别选为 100 和 200,试画出类似于图 5-28 的工作示意图。从连接建立阶段到连接释放都要画上。



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报文握手。说明如不这样做可能会出现什么情况。

如果不使用 3 次握手，那么 A 发送出去但滞留在网络中延时很久才到达 B 的信号，可以在 A 和 B 已经关闭连接后直接建立，此后 B 一直等待与 A 的通信。但事实上 A 根本没想和 B 通信，因此导致 B 段资源一直被占用无法释放，降低了网络资源的利用率。

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 3KB 的数据。在 TCP 连接建立后，A 的发送窗口大小是 2KB。A 的初始序号是 0。

(1) 一开始 A 发送 1KB 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。

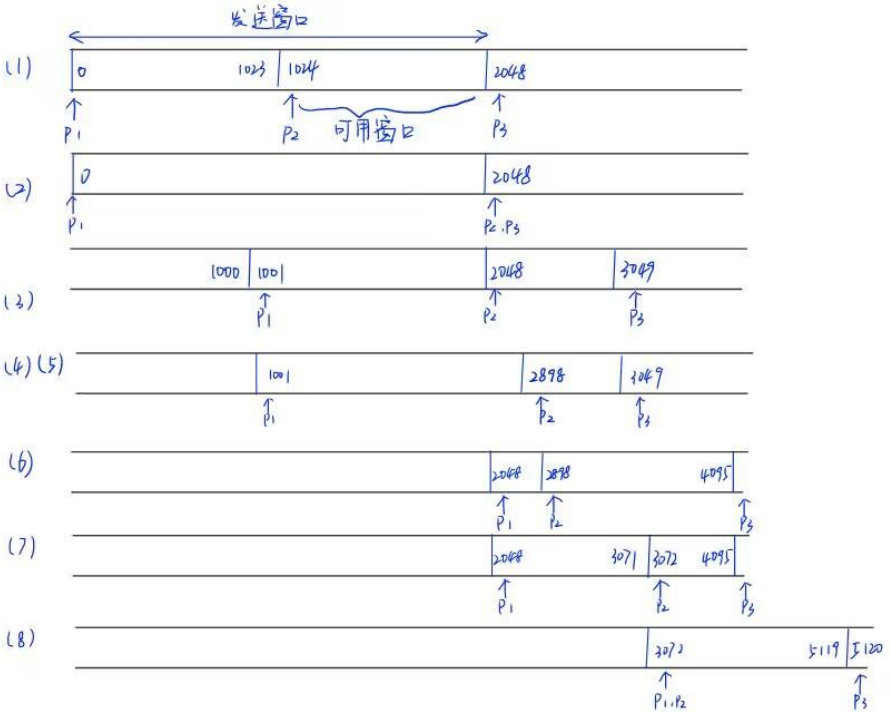
(4) 发送方 A 再发送 850B 的数据。

(5) 发送方 A 收到 ack=900 的确认报文段。（迟到的确认，可以忽略）

(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 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？

不会出现问题，因为发送方收到第二个握手报文时连接已经建立，即使第三个报文丢失，发送方传送数据给接收方，接收方也可以确认。