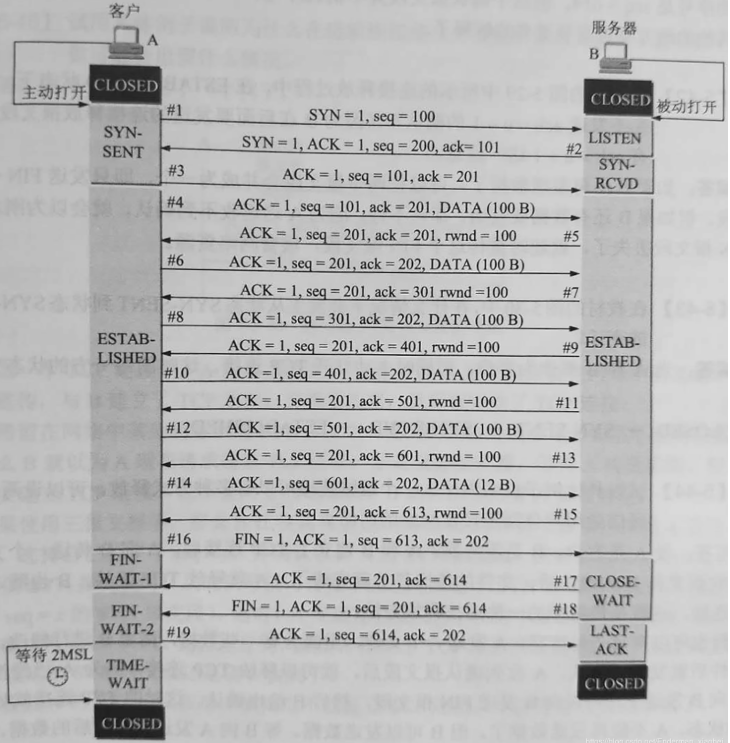
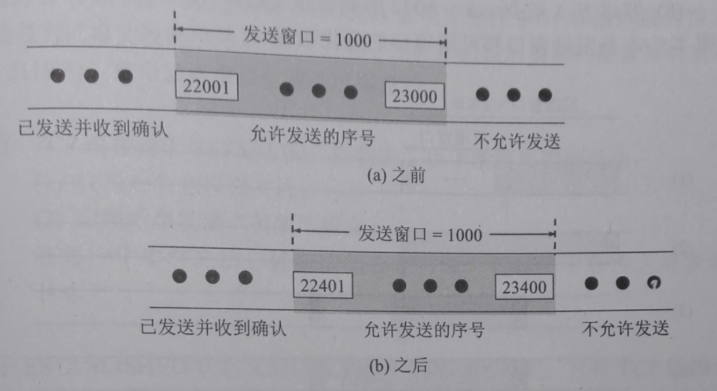
5-41用TCP传送512字节的数据。设窗口为100字节，而TCP报文段每次也是传送100字节的数据。再设发送方和接收方的起始序号分别选为100和200，试画出类似于图5-28的工作示意图。从连接建立阶段到连接释放都要画上。

* 【----- 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-----】



5-59 TCP连接使用1000字节的窗口值，而上一次的确认号是22001。现在收到了一个报文段，确认了字节22401。试用图来说明在这之前与之后的窗口情况。



5-60同上题。但接收方收到确认字节为22401的报文段时，其窗口字段变为1200字节。试用图来说明在这之前与之后的窗口情况。