第十五节 可靠传输与流量控制

一、课程目标

掌握教材 5.6-5.7。重点掌握发送窗口和接受窗口、超时重传、流量控制。

二、课程内容

【可靠传输】

1、TCP 连接的两个端点各自维护两个窗口

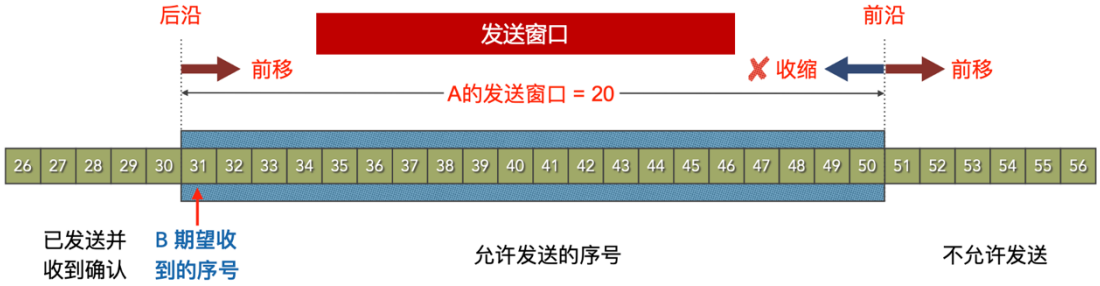
■ 发送窗口：准备发送的数据和已发送但未收到确认的数据。

■ 接收窗口：按序到达但未被应用程序接收的数据、不按序到达的数据

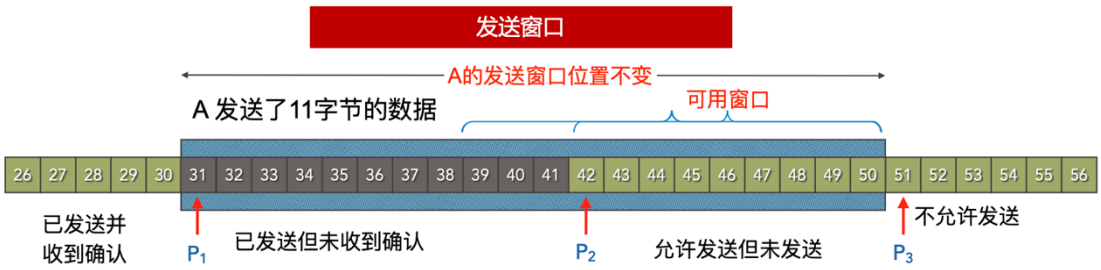
*注意，TCP 连接是全双工，因此通信双方一共维护上述四个窗口（各两个）。

2、TCP 的可靠传输机制用字节的序号进行控制。TCP 所有的确认都是基于序号而不是基于报文段的。

3、窗口：发送方 A 收到接收方 B 的确认报文段（窗口值 20 字节，确认号 21），则发送方可以构造发送窗口如下所示：



随后，发送方 A 继续发送 11 字节数据，但由于第 31 字节数据尚未收到确认，并不改变发送窗口大小及位置。发送窗口如下所示：



其中，P3-P1=A 的发送窗口(又称为通知窗口)；P2-P1=已发送但尚未收到确认的字节数；P3-P2=允许发送但尚未发送的字节数(又称为可用窗口)。此时，接收方 B 的接受窗口如下所示，由于第 34 字节缺失，接收方 B 仍发送确认 33：

关于发送接收的几点注意事项：

（1）A 的发送窗口内的序号都已用完，由于未收到确认，必须停止发送。

（2）A 的发送窗口并不总是和 B 的接收窗口一样大。

（3）未按序到达的数据应临时存放在接收窗口中。

（4）接收方必须有**累积确认**的功能，这样可以减小传输开销。

（5）发送窗口后沿的变化情况有两种，即不动（没有收到新确认）和前移（收到新确认），不可能后退。发送窗口前沿可能向后收缩（接收方通知缩小发送窗口），但 TCP 标准不建议如此操作，容易乱。

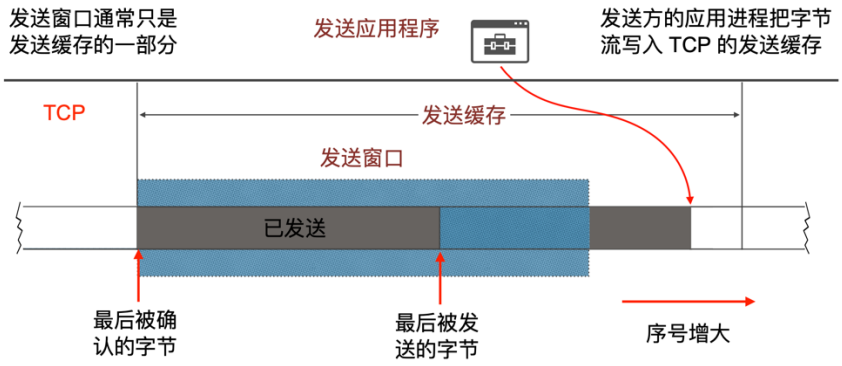
累积确认机制：

目的：发送方可以判断出来传输过程中是否存在乱序或者丢包的情况，从而决定是否进行超时重传或者快速重传。

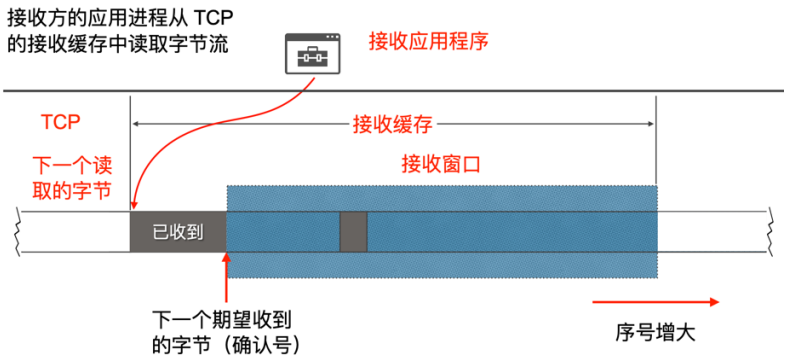
方法：累积确认依靠的是 TCP head 中的确认号 Acknowledgement 以及 ACK 标志位。Acknowledgement number = x+1 代表对于 x+1 之前的所有信息在接收方都已经确认接收完毕，接收方 B 期望发送方 A 发送 x+1 及其后面的消息，此乃累积确认，每次确认的都是前面所有的内容已经接收完毕。

4、缓存，与窗口对应。

- **发送缓存**：发送应用程序传送给发送方 TCP 准备发送的数据；TCP 已发送出但尚未收到确认的数据以及等待进入发送窗口的数据。



- **接收缓存**：按序到达的、但尚未被接收应用程序读取的数据；未按序到达的数据以及未进入到接收窗口的数据。



【超时重传】

5、超时重传时间的选择

原因：超时重传时间设置得太短，引起过多的不必要的重传，网络负荷增大。超时重传时间设置得过长，网络的空闲时间增大，网络传输效率降低。

6、TCP 自适应超时重传时间 RTO 计算方法：

（1）RTO 应略大于加权平均往返时间 RTT_S。

（2）**$RTO = RTT_S + 4 \times RTT_D$**（RFC 2988 建议），其中 RTT_D 是 RTT 偏差的加权平均值。

RTT_S 计算方法：

新的 $RTT_S = (1-\alpha) \times$ 旧的 $RTT_S + \alpha \times$ 新的 RTT 样本

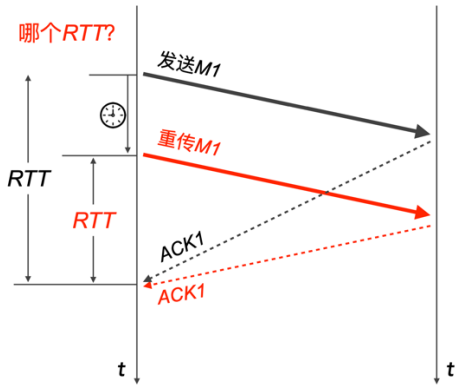
其中，RTT_S 为加权平均往返时间，首次 RTT_S 取第一次测量得到的 RTT 样本值。$0 \leqslant \alpha \leqslant 1$，若$\alpha$接近于零，表示 RTT 值更新较慢；若选择$\alpha$接近于 1，则表示 RTT 值更新较快。RFC 2988 推荐的α值为 1/8，即 0.125。

RTT_D 计算方法：

新的 $RTT_D = (1-\beta) \times$ 旧的 $RTT_D + \beta \times |RTT_S -$ 新的 RTT 样本$|$

其中，RTT_D 初始值取为测量到的 RTT 样本值的一半，$\beta \leqslant 1$，其推荐值是 1/4，即 0.25。

注意，RTO 计算方法细节及注意事项较多，并且实际实现各有不同，教材很难将计算方法完全详细讲解。例如，如何处理报文段重传与 RTT 测量结果？



TCP 采用 Karn 算法解决上述问题，在计算平均往返时延 RTT 时，只要报文段重传了，就不采用其往返时延样本。

进一步，TCP 采用修正的 Karn 算法，更新 RTO 计算方法。

新的 RTO = $\gamma \times$ 旧的 RTO

其中，系数γ的典型值是 2。当不再发生报文段的重传时，才根据报文段的往返时延更新平均往返时延 RTT 和超时重传时间 RTO 的数值。

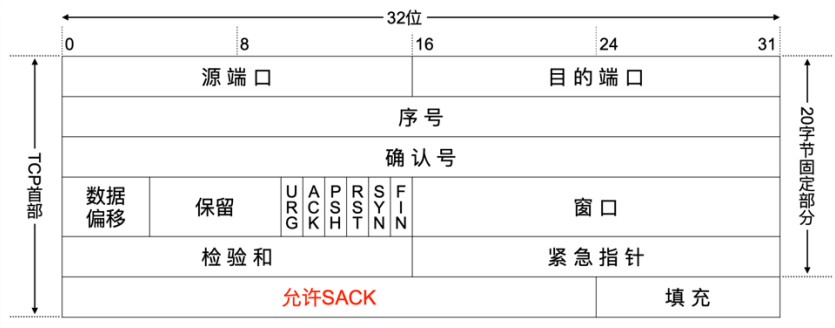
【选择确认】

7、选择确认 SACK (Selective ACK)

问题来源：若收到的报文段无差错，只是未按序号，中间还缺少一些序号的数据（**存在乱序和缺失**），那么能否设法只传送缺少的数据而不重传已经正确到达接收方的数据？

基本原理：从 Back-to-N 到选择确认 SACK (Selective ACK)。

处理流程：双方建立 TCP 连接时，在 TCP 首部的选项中加上“允许 SACK”的选项。首部选项的长度最大有 40 字节，指明一个字节块用掉 8 字节，因此在选项中最多只能指明 4 个字节块的边界信息(另一个字节用于指明是什么选项)。

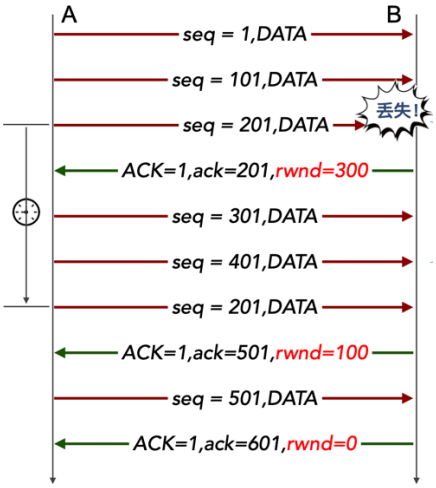


【流量控制】

8、流量控制（让发送方的发送速率不要太快，要让接收方来得及接收）

原因：两个应用程序通过 TCP 协议在网络中传输数据时，双方在硬件性能和软件性能上均可能存在差异，导致双方处理数据的速度不一致。当发送方的发送速度低于接收方接的处理速度时，不会出现问题。而当发送方的发送速度高于接收方的处理速度时，接收方会抛弃暂时无法“安置”的数据包。由于这些丢弃的数据包得不到确认，发送方会重新发送它们，直到他们被成功接收，造成资源浪费。TCP 流量控制就是确保发送方的发送速度不要超出接收方的处理能力。

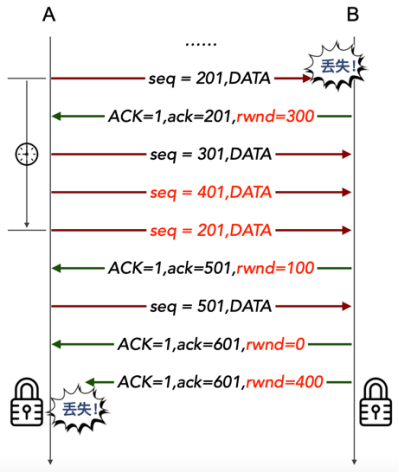
解决方法：TCP 接收方利用自己的接收窗口的大小来限制发送方发送窗口的大小。



9、死锁问题

现象原因：B 向 A 发送了零窗口的报文段后，B 的接收缓存又有了一些存储空间。B 向 A 发送 rwnd = 400 的报文段，但该报文段在传送过程中丢失。导致 A 一直等待收到 B 发送的非零窗口的通知；B 一直等待 A 发送的数据，双方陷入死锁。

解决办法：发送方设置**持续计时器**，在收到接收方的零窗口通知后，应启动该计时器。计时器超时后，发送方向接收方发送零窗口探测报文，接收方在确认这个探测报文段时给出现在的窗口值。若窗口仍然是零，发送方重置持续计时器。



10、TCP 传输效率问题（发送时机问题）

第一种机制：TCP 维持一个变量，它等于最大报文段长度 MSS。只要缓存中存放的数据达到 MSS 字节时，就组装成一个 TCP 报文段发送出去；

第二种机制：由发送方的应用进程指明要求发送报文段，即 TCP 支持的推送(push)操作、紧急数据 URG；

第三种机制：发送方的一个计时器期限到了，这时就把当前已有的缓存数据装入报文段(但长度不能超过 MSS)发送出去。

11、Nagle 算法（第四种机制，现实使用的，应对连续小字节交互场景）

（1）发送方先发送第一个数据字节，缓存后面到达的数据字节；

（2）发送方收到对第一个数据字符的确认后，把发送缓存中的所有数据组装成一个报文段发送出去，继续对随后到达的数据进行缓存；

（3）只有在收到对前一个报文段的确认后继续发送下一个报文段；当到达的数据已达到发送窗口大小的一半或已达到报文段的最大长度时，就立即发送一个报文段。

12、糊涂窗口综合症

现象：接收方的 TCP 缓冲区已满，接收方会向发送方发送窗口大小为 0 的报文；接收方的应用进程以交互方式每次只读取一个字节，接收方发送窗口大小为一个字节的确认报文，发送方发送一个字节的数据，接收窗口又满了，如此循环往复。

解决方法：接收方等待一段时间，使得接收缓存已有足够空间容纳一个最长的报文段，或者等到接收缓存已有一半空闲的空间；只要出现这两种情况之一，接收方就发出确认报文，并向发送方通知当前的窗口大小。

三、重点习题

P253：全部

四、参考资料