TCP：1.编号确认：

①. 序号(Seq)：本报文段所发送的数据第一个字节在整个字节流里的编号

一般会给第一个包的Seq，或明确说明初始值(ISN)

下一个报文段Seq = 当前报文段Seq + 当前报文段数据字度(单位：字节)

特殊标志位：SYN、FIN、各占1个序号

e.g. 一个SYN包(Seq = J, SYN=1)，确认后，下一个数据包长度为J+1

一个有100字节的FIN包(Seq = K)、下一个 Ack包应为 K+100+1 = K+101 (FIN 占个序号)

②. 确认号(Ack)：期望收到下一个报文段的第一个数据字节的序号

Ack = 已正确收到最后一字节的序号 + 1

如果接收方收到Seq=M，数据长度为L的报文段(无乱序)，则 Ack = M+L

若包含SYN、FIN，则也要+1， Ack = M+L+1

我发送的Ack = 收到的Seq + 收到的 Datalen

TCP流量控制

1. 接收窗口（rwnd）

由接收方在TCP报文头部窗口大小字段动态告诉发送方

接收方还能接收多少字节数据

发送方数据量不能超过接收方通告的rwnd。

2. 发送窗口：

大小 = min(接收方通告rwnd, 拥塞窗口cwnd)

定义发送方可发送的数据范围

3. 滑动机制：

接收方接收数据交给应用层，其缓冲空间释放，rwnd增大。

接收方通过后续Ack包将增大的rwnd告诉对方（发送方）

发送方收到新rwnd，向右移动发送窗口左边界，并根据新rwnd调整右边界，允许发送更多新数据

计算/判断窗口值　　窗口大小

1. 接收方在Ack包携带字段即当前rwnd。

2. 已发送数据量 ≤ min(rwnd, cwnd)

在流量控制中，通常cwnd很大，限制因素是rwnd。

3. 可用窗口大小：

min(rwnd, cwnd) − （最后发送字节序号 − 最后一个被确认的字节序号）

可用窗口 = 当前发送窗口大小 − 已发送但未收到Ack的数据量

4. 窗口变化：

Ack接收新数据，发送窗口左边界右移，rwnd允许、可用窗口增大。当收到接收方更新通告的rwnd增大，发送窗口右边界可能右移，可用窗口增大。通告rwnd小，发送方也必须减小发送窗口适应。

TCP 拥塞控制（4个算法）

1. 慢启动：（SS）

触发条件：连接开始，发生超时重传

cwnd = 1 * MSS（最大报文段长度，给定或可计算）

效果：cwnd 按指数个

结束条件：cwnd ≥ ssthresh（慢启动阈值）或检测到丢包

2. 拥塞避免（CA）

触发：慢启动到阈值时；以快恢复超出时。

每收到一个Ack，$cwnd = cwnd + \frac{(MSS * MSS)}{cwnd}$

效果：cwnd 线性个（每RTT增长1MSS）

e.g. cwnd = 10 MSS，每收到一个Ack，增长 $\frac{1}{10}$ MSS

10个Ack才增长1MSS，即1个RTT增加大约1MSS

结束条件：检测到丢包

3. 快重传（FR）

触发：发送方连续收到3个确认的Ack确认同一个序号（即收到更高序号分组，中间有分组丢失）

立即重传接收方重复Ack所指示的丢失的报文段（Seq = Ack）

进入快速恢复阶段

4. 快恢复（FRCV）

在外数据量：指已发送但尚未确认的数据量。

触发：快重传后，

计算：$ssthresh = \max\left(\frac{在外数据量}{2}, 2MSS\right)$

或 $ssthresh = \frac{cwnd}{2}$（更常见情况）

超时回到 cwnd = 1mss

3个DUP Ack回到 cwnd = ssthresh + 3MSS

cwnd = ssthresh + 3 MSS（收到3个DupAck，表明3个分组离开网络）（$ssthresh = \frac{cwnd}{2}$）

每收到一个重复Ack（Dup Ack）：cwnd = cwnd + 1MSS（认为1个分组离开网络）

当收到一个新确认的Ack（部分或全部修复3丢失数据的Ack）

退出快恢复状态

cwnd = ssthresh // 将cwnd设为新ssthresh值。

进入拥塞避免阶段

ssthresh只在超时或收到3个DUP Ack时更新

如果发生超时，退出快恢复，$ssthresh = \frac{cwnd}{2}$（至少2MSS）（$ssthresh = \frac{cwnd}{2}$），cwnd = 1 mss，进入慢启动阶段

（超时无论处于SS、CA，还是FRCV都要强制回SS）

所有cwnd，ssthresh默认以MSS为单位，报文也会给出MSS值

TCP 连接管理的双向释放

为什么要双向释放、好处，双向释放是怎么做的？

TCP是面向连接的可靠传输、连接建立释放需双向确认。

TCP连接通过4次握手确保可靠终止、释放采用双向释放：

1. 主动关闭方发送FIN报文段，请求关闭连接
2. 被动关闭方收到FIN，发送ACK确认，此时连接未完全关闭
3. 被动关闭方完成数据发送后，发送FIN报文段
4. 主动关闭方收到FIN，发送ACK进行确认，连接正式关闭。

双向释放好处是保证可靠性、及时释放资源防止异常终止。