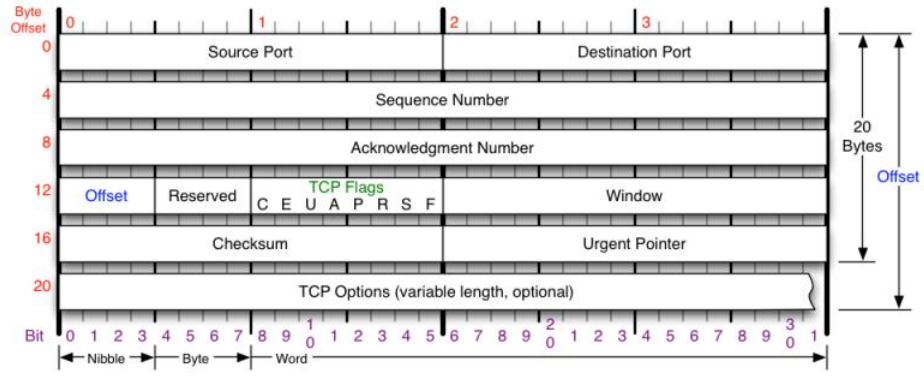
TCP 和拥塞控制

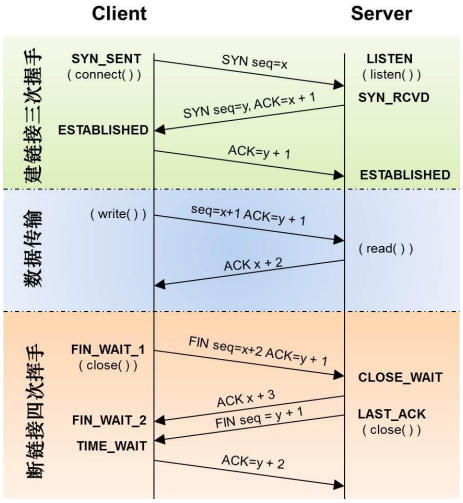
一、TCP 基础

1. Header



- Sequence Number 包的序号（Seq），增加数为包的长度
- Acknowledgement Number 确认序号（Ack）
- Window 窗口大小（接收缓冲区大小）
- TCP Flag 包的类型

2. 状态机



(1) FLAGS:

① SYN：同步序列编号（Synchronize Sequence Numbers）

② ACK：确认序号有效

③ FIN：释放一个连接（Finish）

(2) Entries & Parameters:

① MSL: Maximum Segment Lifetime

② SYN flood: SYN 包攻击

③ tcp_tw_reuse: TIME_WAIT 重用

④ tcp_tw_recycle: TIME_WAIT 回收

⑤ tcp_max_tw_buckets: TIME_WAIT 最大数量

3. TCP 重传

(1) 超时重传（Timeout Retransmit）

① sender 死等 ACK，超时重传

(2) 快速重传（Faster Retransmit）

① receiver 发送最后一个 ACK

② 收到 3 个重复的 ACK，则重传该包

(3) SACK（Selective Acknowledgment）

① Header 中增加 SACK 选项，记录接收端收到的数据包的 SeqNum 范围

② Parameters: tcp_sack

(4) D-SACK（Duplicate Selective Acknowledgment）

① SACK 中记录重复的 SeqNum 范围（ACK 范围大于于 SACK 范围）

② Parameters: tcp_dsack

4. TCP 的 RTT（Round Trip Time）

(1) 经典算法（RFC792）

① Smoothed RTT:

SRTT = (α * 历史 SRTT) + ((1- α) * RTT)

② Retransmission TimeOut:

RTO = min [UBOUND, max [LBOUND, (β * SRTT)]]

③ α = [0.8, 0.9]

④ β = [1.3, 2.0]

⑤ UBOUND = 最大 timeout 时间

⑥ LBOUND = 最小 timeout 时间

(2) Karn / Partridge 算法

① 忽略重传的 RTT

② 发生重传时，对 RTO 翻倍

(3) Jacobson / Karels 算法

① SRTT = 历史 SRTT + α *(RTT – 历史 SRTT)

② DevRTT = (1- β) * DevRTT + β * (|RTT-SRTT|)

③ RTO= μ * SRTT + ∂ *DevRTT

④ α = 0.125

⑤ β = 0.25

⑥ μ = 1

⑦ ∂ = 4

5. TCP 滑动窗口

(1) sender 窗口（由 receiver 的 ACK，窗口向右移动）

① #1 已发送，已收到 ACK

② #2 已发送，未收到 ACK

③ #3 未发送，有空间

④ #4 未发送，无空间

(2) zero window

① receiver 缓冲区满，sender 隔段时间发送 ZWP（Zero Window Probe），让 receiver 发送 ACK 告知窗口大小

② Silly Window Syndrome

1) 数据塞不满 MTU（Maximum Transmission Unit），浪费带宽
2) 解决方案：
   a. David D Clark 算法：windowSize 小于某值，receiver 回复 windowSize（由 receiver 引起的）
   b. Nagle 算法：sender 将多个小包合并成一个大包发送（由 sender 引起的）

二、 TCP 拥塞控制

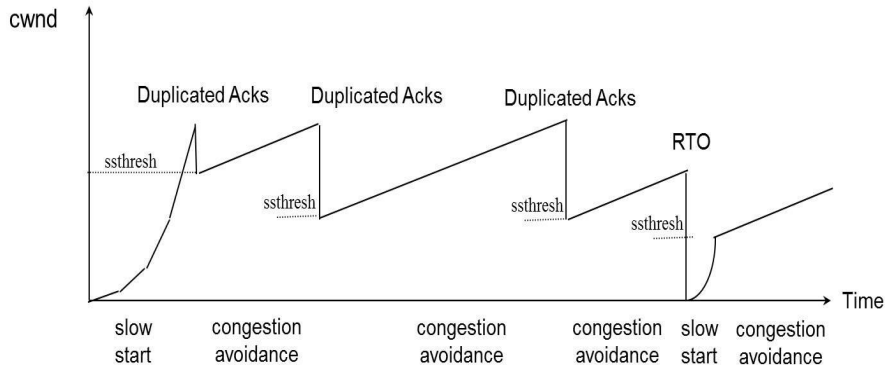
1. 拥塞控制算法

(1) 慢启动（Slow Start）
① 初始 cwnd（Congestion Window）=1
② 每收到一个 ACK，cwnd++（线性增长）
③ 每过一个 RTT，cwnd=2*cwnd（指数增长）
④ cwnd>=ssthresh（slow start threshold，一般为 65535 Byte）时，进入拥塞避免（Congestion Avoidance）

(2) 拥塞避免（Congestion Avoidance）
① 每收到一个 ACK，cwnd=cwnd+1/cwnd
② 每过一个 RTT，cwnd++

(3) 快速重传（Faster Retransmit）
① 发生快速重传（收到 3 个 duplicated ACK）时，cwnd/=2，ssthresh=cwnd
② 进入快速恢复（Fast Recovery）

(4) 快速恢复（Fast Recovery）
① cwnd=ssthresh+3*MSS（Maximum Segment Size）
② 重传丢失的包（根据 Duplicate ACK）
③ 再收到一个 Duplicated ACK，cwnd++
④ 收到新的 ACK 时，cwnd=ssthresh，进入拥塞避免



(5) TCP New Reno
① 没有 SACK 时，发生快速重传，进入 Fast Recovery
② 在 Fast Recovery 阶段，根据部分 ACK（Partial Acknowledgment）机制，继续恢复丢失的包，直到所有丢失的包被确认，再进入拥塞避免

(6) FACK（Forward Acknowledgment）
① SND.UNA：未被确认的最小序列号（发送窗口左端）
② SND.NXT：下一个即将发送的序列号
③ FACK：SACK 选项中 最高确认序列号
④ 未确认数据量（inflight data）：

inflight=SND.NXT−FACK

⑤ 丢包检测：

当 inflight 突然下降时，sender 认为 FACK 之前的包丢失，重传丢失的包