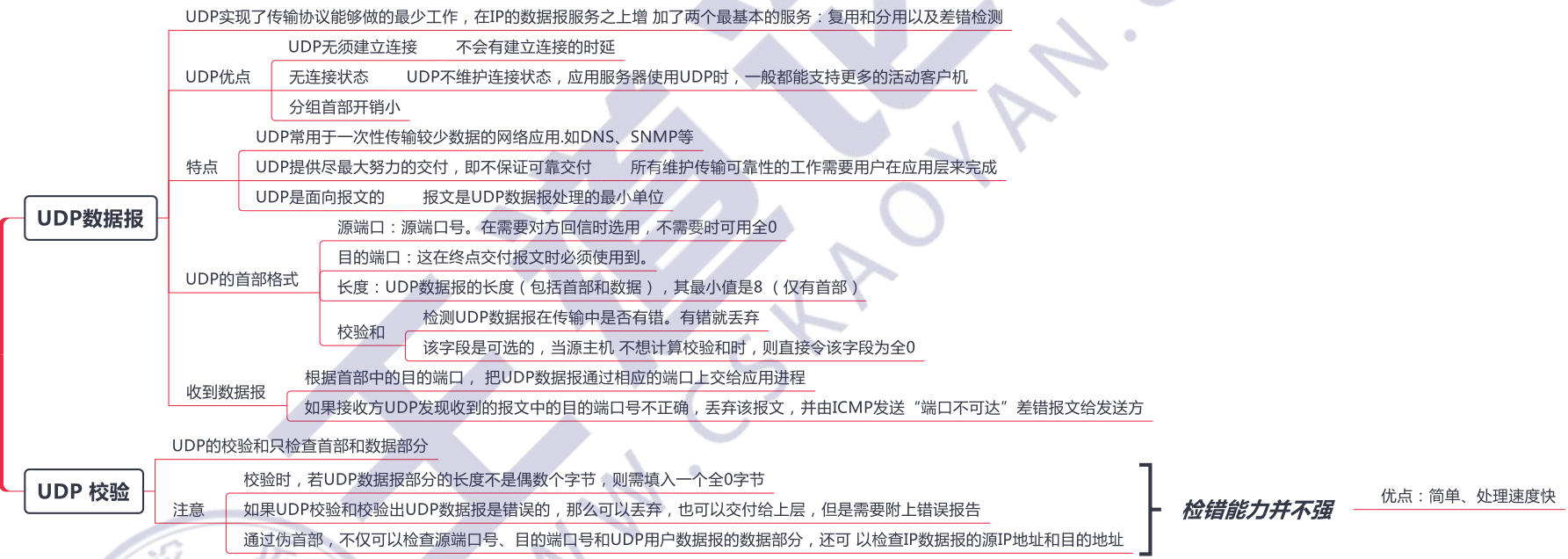


5.1传输层提供的服务



注：仅供王道VIP学员使用 严禁外部传播！

5.2 UDP协议



5.3 TCP协议（上）

TCP协议的特点

- TCP是在不可靠的IP层之上实现的可靠的数据传输协议，它主要解决传输的可靠、有序、无丢失和不重复问题
- TCP是面向连接的传输层协议
- 特点
 - 每条TCP连接只能有两个端点，每条TCP连接只能是点对点的（一对一）
 - TCP提供可靠的交付服务，保证传送的数据无差错、不丢失、不重复且有序
 - TCP提供全双工通信，允许通信双方的应用进程在任何时候都能发送数据，为此TCP连接的两端都设有发送缓存和接收缓存，用来临时存放双向通信的数据
 - TCP是面向字节流的

TCP报文段

TCP传送的数据单元称为报文段

作用

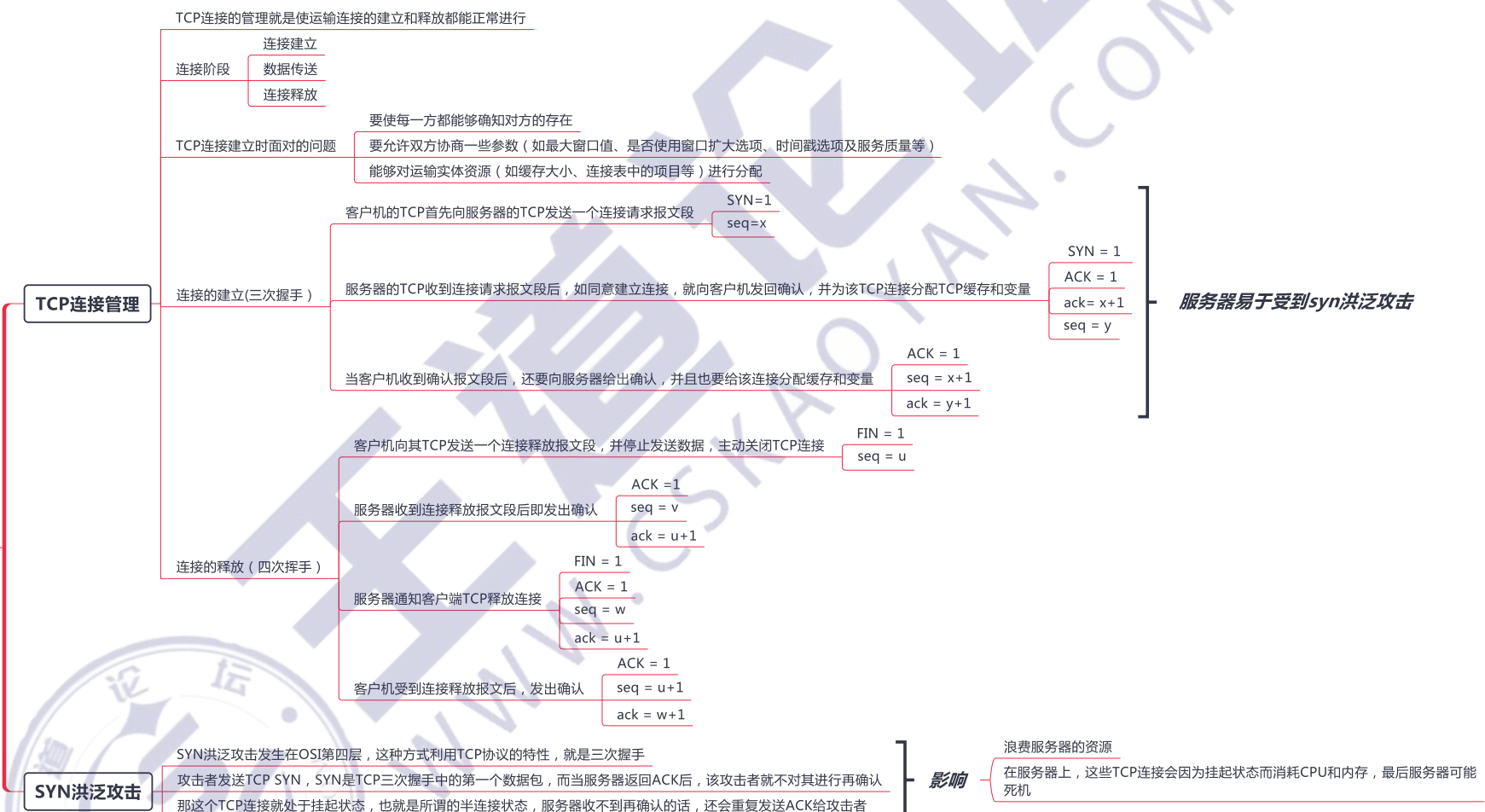
- 运载数据
- 建立连接、释放连接和应答

字段意义

- 源端口和目的端口字段：各占2B，端口是运输层与应用层的服务接口，运输层的复用和分用功能都要通过端口实现
- 序号字段
 - 占4B，TCP是面向字节流的（即TCP传送时是逐个字节传送的），所以TCP连接传送的数据流中的每个字节都编上一个序号
 - 序号字段的值指的是本报文段所发送的数据的第一个字节的序号
- 确认号字段：占4B，是期望收到对方的下一个报文段的数据的第一个字节的序号
- 数据偏移（即首部长度）：占4位，它指出TCP报文段的数据起始处距离TCP报文段的起始处有多远
- 保留字段：占6位，保留为今后使用
- 紧急位URG：URG=1时，表明紧急指针字段有效。它告诉系统报文段中有紧急数据，应尽快传送（相当于高优先级的数据）
- 确认位ACK
 - ACK=1时确认号字段才有效
 - ACK=0时，确认号无效
- 推送位PSH（Push）：接收TCP收到PSH=1的报文段，就尽快地交付给接收应用进程，而不再等到整个缓存都填满后再向上交付
- 复位位RST（Reset）：RST=1时，表明TCP连接中出现严重差错（如主机崩溃或其他原因），必须释放连接，然后再重新建立运输连接
- 同步位SYN：SYN=1表示这是一个连接请求或连接接收报文
- 终止位FIN（Finish）：用来释放一个连接。FIN=1表明此报文段的发送方的数据已发送完毕，并要求释放传输连接
- 窗口字段：占2B，表示允许对方发送的数据量，单位为字节
- 校验和：占2B，校验和字段检验的范围包括首部和数据两部分
- 紧急指针字段：占16位，指出在本报文段中紧急数据共有多少字节（紧急数据放在本报文段数据的最前面）
- 选项字段：长度可变，TCP最初只规定了一种选项，即最大报文段长度
- 填充字段：使整个首部长度是4B的整数倍

tcp规定，在连接建立后所有传送的报文段都必须把ack置1

5.3 TCP协议（中）



5.3 TCP协议（下）

TCP可靠传输（实现机制）

- 序号 TCP首部的序号字段用来保证数据能有序提交给应用层，序号建立在传送的字节流之上
- 确认 TCP首部的确认号是期望收到对方的下一个报文段的数据的第一个字节的序号
TCP默认使用累计确认，即TCP只确认数据流中至第一个丢失字节为止的字节
- 重传 TCP每发送一个报文段，就对这个报文段设置一次计时器。计时器设置的重传时间到期但还未收到确认时，就要重传这一报文段
冗余ACK 再次确认某个报文段的 ACK,而发送方先前已经收到过该报文段的确认
当收到对于某个报文段的3个冗余ACK，可以认为该报文段已经丢失。这时发送方可以立即对该报文执行重传

TCP流量控制

- 匹配发送方的发送速率与接收方的读取速率
- 流量控制机制 基于滑动窗口协议的流量控制机制
实现方法 接收方根据自己接收缓存的大小，动态地调整发送方的发送窗口大小（接收窗口 $rwnd$ ），限制发送方向网络注入报文的速率
发送方根据其对当前网络拥塞程序的估计而确定的窗口值，这称为拥塞窗口 $cwnd$ 其大小与网络的带宽和时延有关
- 流量控制的区别 传输层：定义端到端用户之间的流量控制
数据链路层：定义两个中间的相邻结点的流量控制
- 窗口大小的区别 传输层：滑动窗口可以动态变化
数据链路层：滑动窗口不能动态变化

拥塞控制：防止过多的数据注入网络，保证网络中的路由器或链路不致过载

- 拥塞控制与流量控制的区别 相同点：它们都通过控制发送方发送数据的速率来达到控制效果
区别 拥塞控制是让网络能够承受现有的网络负荷，是一个全局性的过程，涉及所有的主机、所有的路由器，以及与降低网络传输性能有关的所有因素
流量控制是点对点的通信量的控制，即接收端控制发送端，它所做的是抑制发送端发送数据的速率，以便使接收端来得及接收

- 窗口 接收窗口 $rwnd$ ：接收方根据目前接收缓存大小所许诺的最新窗口值，反映接收方的容量
拥塞窗口 $cwnd$ ：发送方根据自己估算的网络拥塞程度而设置的窗口值，反映网络的当前容量
- 发送窗口的上限值 = $\min[rwnd, cwnd]$

TCP拥塞控制

- 慢开始 每经过一个传输轮次（即往返时延RTT），拥塞窗口 $cwnd$ 就会加倍
令拥塞窗口 $cwnd = 1$ 即一个最大报文段长度MSS，每收到一个对新报文段的确认后，将 $cwnd$ 加1
使分组注入网络的速率更加合理
慢开始一直把拥塞窗口 $cwnd$ 增大到一个规定的慢开始门限 $ssthresh$ （阈值），然后改用拥塞避免算法
- 实现机制 拥塞避免 发送端的拥塞窗口 $cwnd$ 每经过一个往返时延RTT就增加一个MSS的大小，而不是加倍
 $cwnd$ 按线性规律缓慢增长（即加法增大），而当出现一次超时（网络拥塞）时，令慢开始门限 $ssthresh$ 等于当前 $cwnd$ 的一半（即乘法减小）
- 快重传 当发送方连续收到三个重复的ACK报文时，直接重传对方尚未收到的报文段，而不必等待那个报文段设置的重传计时器超时
- 快恢复 发送端收到连续三个冗余ACK（即重复确认）时，执行“乘法减小”算法，把慢开始门限 $ssthresh$ 设置为出现拥塞时发送方 $cwnd$ 的一半
把 $cwnd$ 的值设置为慢开始门限 $ssthresh$ 改变后的数值，然后开始执行拥塞避免算法（“加法增大”），使拥塞窗口缓慢地线性增大
- 网络拥塞的处理 发送方检测到超时事件的发生（未按时收到确认，重传计时器超时），就要把慢开始门限 $ssthresh$ 设置为出现拥塞时的发送方的 $cwnd$ 值的一半（但不能小于2）
然后把拥塞窗口 $cwnd$ 重新设置为 L 执行慢开始算法