

传输层概述

- 传输层是主机才有的层次，路由器最多能够达到网络层
- 传输层，为运行在**不同主机上的进程**之间提供了逻辑通信，网络层提供**主机之间**的逻辑通信
- 只有主机的协议栈才有传输层和应用层

传输层的功能

提供**应用进程和进程之间的逻辑通信**

- 网络层为主机提供逻辑通信，传输层为主机上的进程提供端到端的逻辑通信

复用和分用

- 复用：发送方不同的应用进程都可同一个传输层协议传送数据（**传输数据**）
- 分用：接收方的传输层在剥去报文的首部后能够把这些数据正确交付到目的应用进程（**交付数据**）

差错检测

- 网络层中，只校验首部是否出差错而不检查数据部分
- 传输层需要对收到的报文进行差错检测

提供两种不同的传输协议

- 面向连接的TCP协议
- 无连接的UDP协议

传输层的号址和端口

- 传输层使用端口来标识主机中的应用进程
- 端口值具有本地意义，只是为了标志本计算机应用层中各个进程在和传输层交互时的层间接口
- 不同计算机的相同端口是没有联系的
- 端口号长度16bit，能标识65534个不同的端口号

- **端口的分类**
    - 服务端使用的端口号
      - 熟知端口号（0-1023）
      - 登记端口号（1024-49151）
    - 客户端使用的端口号（49152-65535）
- |     |        |      |     |      |      |      |
|-----|--------|------|-----|------|------|------|
| FTP | TELNET | SMTP | DNS | TFTP | HTTP | SNMP |
| 21  | 23     | 25   | 53  | 69   | 80   | 161  |

套接字

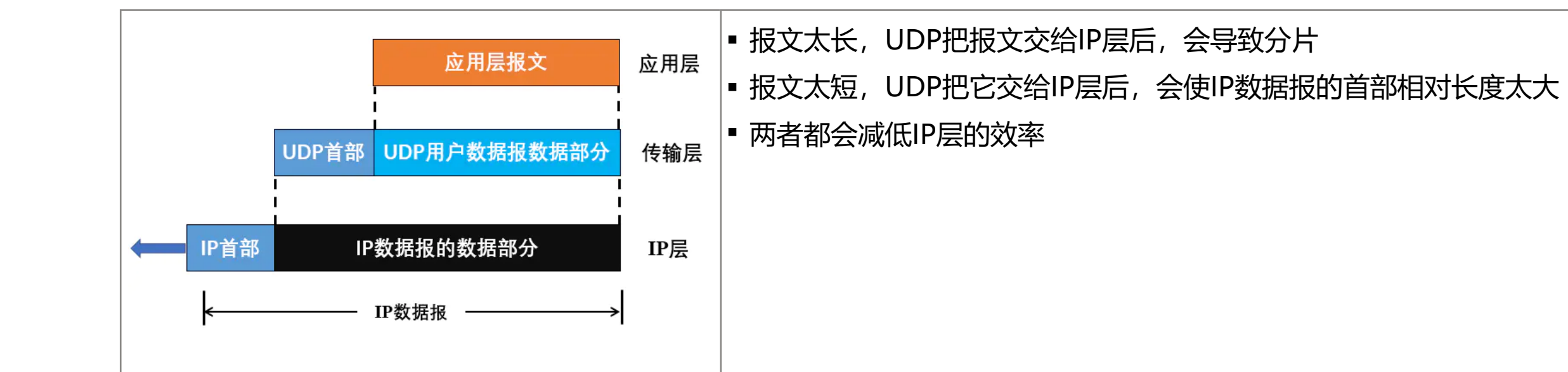
- 套接字Socket=（IP地址：端口号）
- 套接字唯一地标识网络中的一台主机和其上的一个应用（进程）

UDP协议

- 用户数据报协议UDP（User Datagram Protocol）= IP的数据报服务 + （复用，分用，差错检测）

特点

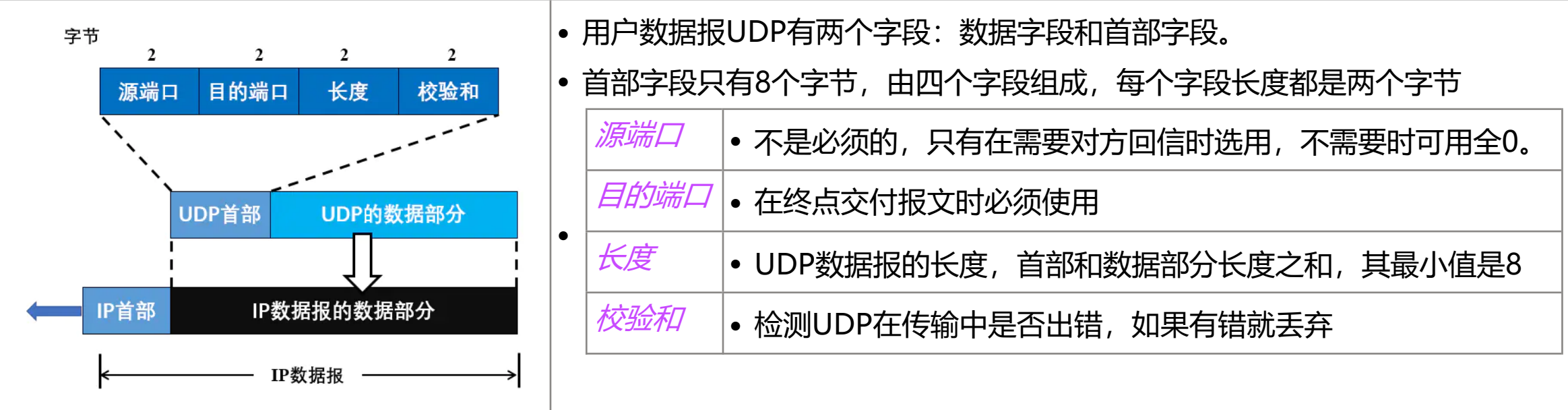
- UDP是无连接的
- UDP使用**尽最大努力交付**，不保证可靠交付
- UDP**没有拥塞控制**，有利于实时应用（视频，语音）
- UDP分组首部开销小，TCP有20b的首部开销，UDP仅有8b的开销
- UDP是**面向报文的**，报文是UDP数据报处理的最小单位



考点

- UDP校验和校验出UDP数据报是错误的，可以丢弃，也可以交付给上层，但是要附上错误报告

UDP首部格式



UDP基于端口的分用步骤

- 当运输层P层收到UDP数据报时，就根据首部中的目的端口，把UDP数据报通过相应的端口交付给上层应用进程
- 如果接收方UDP发现收到的报文的端口号不正确（即不存在对应于该端口号的应用进程），即丢弃该报文
- 然后由网际控制报文协议ICMP发送“端口不可达”差错报文给发送方。

UDP校验

- UDP检验和提供差错检测功能。在计算校验和时，要在UDP用户数据报之前增加12字节的伪首部
- 伪首部既不向下传送也不向上递交，只是为了计算校验和



- **各字段内容**
  - 源IP地址和目的IP地址：和IP数据一样，各占4个字节。
  - 伪首部第3个字节是全零。
  - 协议字段：UDP协议的协议字段值是17。
  - UDP长度：UDP用户数据报长度，首部长度和数据部分长度之和。

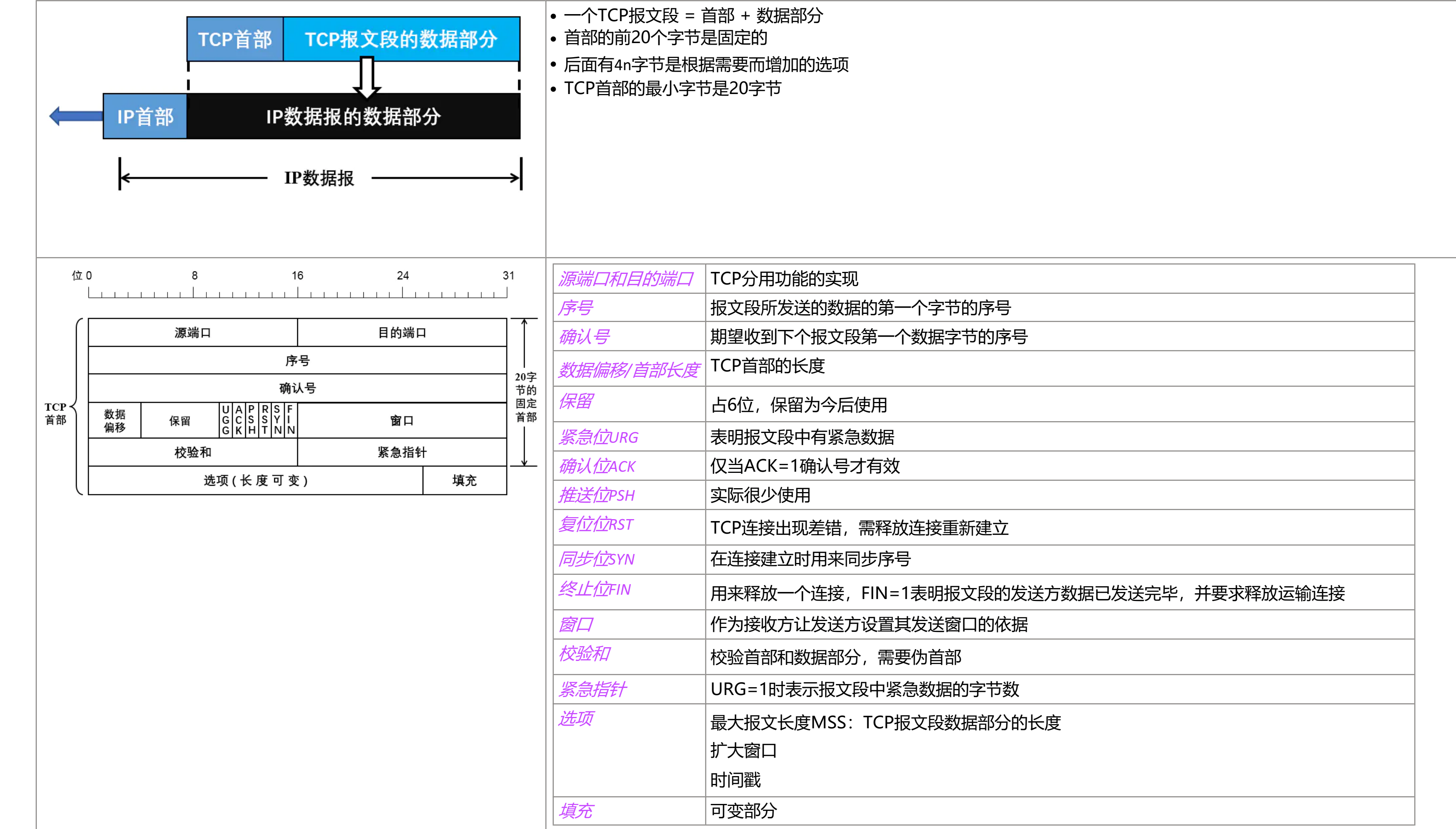
- **校验和过程**
  - 将校验和字段置位0。
  - 将伪首部和UDP用户数据报（首部和数据部分）看成是以16位为单位的二进制组成，依次进行二进制反码求和。
  - 将求和的结果的反码写入校验和字段。

TCP协议

TCP特点

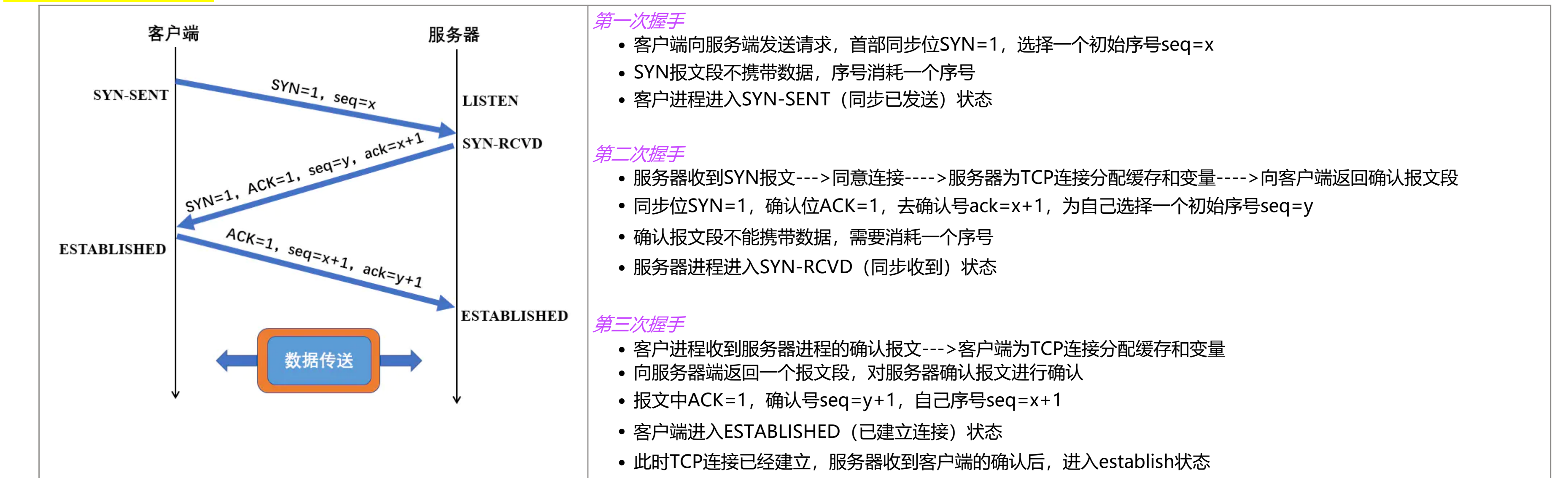
- TCP是**面向连接**的传输层协议
- TCP连接是一条**虚电路（逻辑连接）**，不是一条真正的物理连接
- 每条TCP连接**只能有两个端点**，每条TCP连接只能是点对点的
  - TCP连接的端点叫做套接字（socket）或插口
  - 套接字Socket=（IP地址：端口号）
  - TCP连接=（socket1,socket2）= {（IP1：prot1），（IP2：prot2）}
- TCP提供**可靠交付的服务**，保证数据**无差错，不丢失，不重复，并且按序到达**
- TCP提供全双工通信
  - 发送缓存：用于存放**准备发送的数据和已发送但未收到确认的数据**
  - 接受缓存：用于存放**按序到达但尚未被接受应用程序读取的数据和不按序到达的数据**
- TCP**面向字节流**，TCP把应用程序交下来的数据仅视作一连串的无结构的字节流

TCP报文段



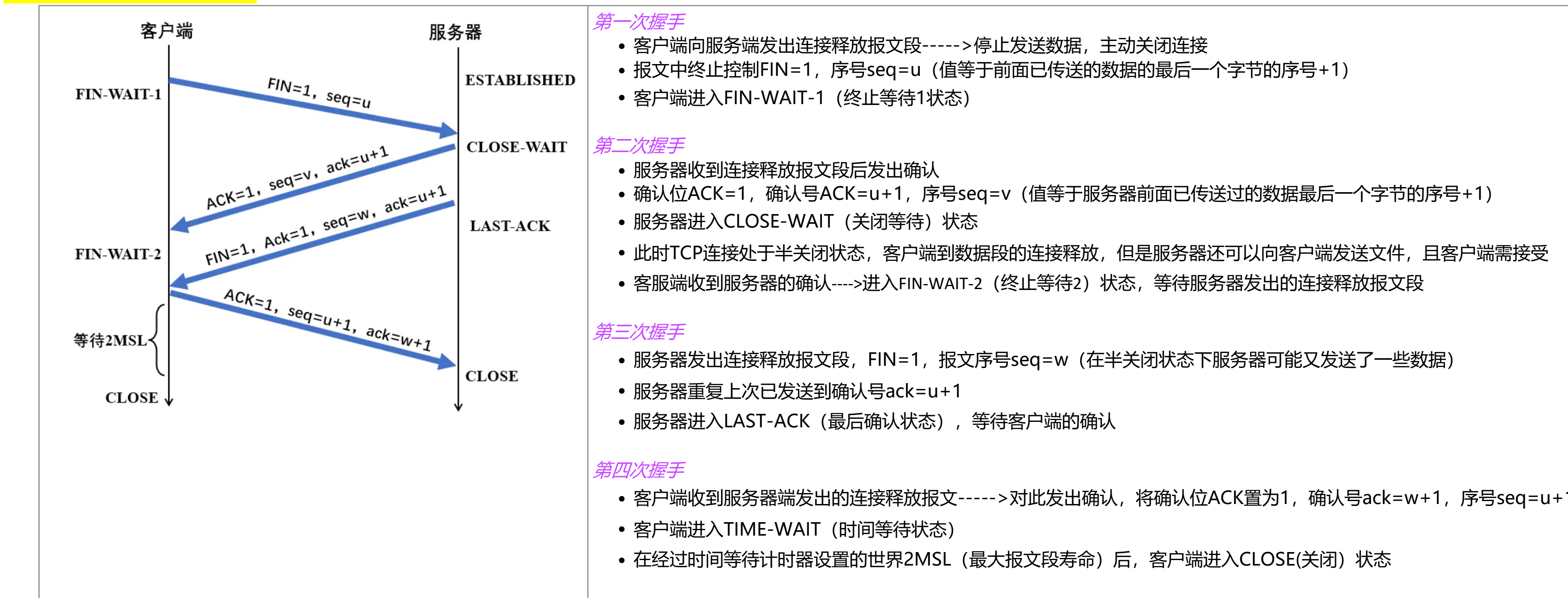
- **源端口和目的端口**：TCP分用功能的实现
- **序号**：报文段所发送的数据的第一个字节的序号
- **确认号**：期望收到下一个报文段第一个数据字节的序号
- **数据偏移/首部长度**：TCP首部的长度
- **保留**：占6位，保留为今后使用
- **紧急标志URG**：表明报文段中有紧急数据
- **确认标志ACK**：仅当ACK=1时确认号才有效
- **推送标志PSH**：实际很少使用
- **复位标志RST**：TCP连接出现差错，需释放连接重新建立
- **同步标志SYN**：在连接建立时用来同步序号
- **终止标志FIN**：用来释放一个连接，FIN=1表明报文段的发送方数据已发送完毕，并要求释放运输连接
- **窗口**：作为接收方让发送方设置其发送窗口的依据
- **校验和**：校验首部和数据部分，需要伪首部
- **紧急指针**：URG=1时表示报文段中紧急数据的字节数
- **选项**：最大报文长度MSS：TCP报文段数据部分的长度
- **扩大窗口**：时间戳
- **填充**：可变部分

TCP连接（三次握手）



- **第一次握手**
  - 客户端向服务端发送请求，首部同步位SYN=1，选择一个初始序号seq=x
  - SYN报文段不携带数据，序号消耗一个序号
  - 客户进程进入SYN-SENT（同步已发送）状态
- **第二次握手**
  - 服务器收到SYN报文-->同意连接-->服务器为TCP连接分配缓存和变量-->向客户端返回确认报文段
  - 同步位SYN=1，确认位ACK=x+1，去确认号ack=x+1，为自己选择一个初始序号seq=y
  - 确认报文段不能携带数据，需要消耗一个序号
  - 服务器进程进入SYN-RCVD（同步收到）状态
- **第三次握手**
  - 客户进程收到服务器进程的确认报文-->客户端为TCP连接分配缓存和变量
  - 向服务端返回一个报文段，对服务器确认报文进行确认
  - 报文中ACK=1，确认号seq=y+1，自己序号seq=x+1
  - 客户端进入ESTABLISHED（已建立连接）状态
  - 此时TCP连接已经建立，服务器收到客户端的确认后，进入establish状态

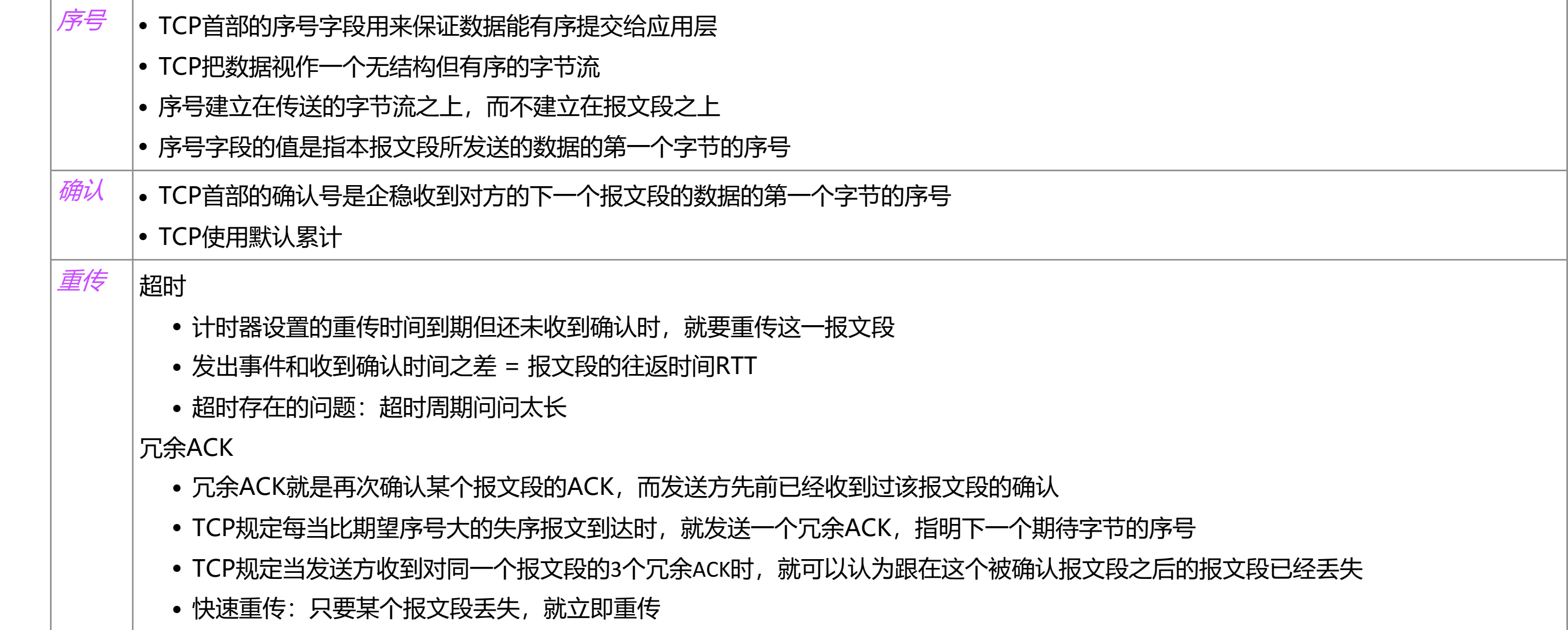
TCP连接释放（四次握手）



- **第一次握手**
  - 客户端向服务端发出连接释放报文段-->停止发送数据，主动关闭连接
  - 报文中终止控制FIN=1，序号seq=u（值等于前面已传送的数据的最后一个字节的序号+1）
  - 客户端进入FIN-WAIT-1（终止等待1状态）
- **第二次握手**
  - 服务器收到连接释放报文段后发出确认
  - 确认位ACK=1，确认号ACK=u+1，序号seq=v（值等于服务器前面已传送过的数据最后一个字节的序号+1）
  - 服务器进入CLOSE-WAIT（关闭等待）状态
  - 此时TCP连接处于半关闭状态，客户端到数据段的连接释放，但是服务器还可以向客户端发送文件，且客户端需接受
  - 客户端收到服务器的确认-->进入FIN-WAIT-2（终止等待2）状态，等待服务器发出的连接释放报文段
- **第三次握手**
  - 服务器发出连接释放报文段，FIN=1，报文序号seq=w（在半关闭状态下服务器可能又发送了一些数据）
  - 服务器重复上次已发送到确认号ack=u+1
  - 服务器进入LAST-ACK（最后确认状态），等待客户端的确认
- **第四次握手**
  - 客户端收到服务器端发出的连接释放报文-->对此发出确认，将确认位ACK置为1，确认号ack=w+1，序号seq=u+1
  - 客户端进入TIME-WAIT（时间等待状态）
  - 在经过时间等待计时器设置的世界2MSL（最大报文段寿命）后，客户端进入CLOSE(关闭)状态

TCP可靠传输

- TCP提供的可靠数据传输保证接收方进程从缓存区读出的字节流与发送方发出的字节流完全一样
- TCP使用校验（与UDP一致），序号，确认，重传来达到这个目的



TCP流量控制

- 流量控制的目的：使发送方的发送速率与接收应用程序的读取速率相匹配
- 判断网络拥塞的依据就是超时
- TCP利用滑动窗口机制实现对发送方的流量控制
- TCP的窗口单位是字节
- 发送窗口的上限值=min[rwnd, cwnd]
- **拥塞窗口Cwnd**：发送方根据当前网络拥塞程度估计而确认的窗口值
- **接收窗口rwnd**：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

TCP拥塞控制算法

- 拥塞控制是指防止过多的数据注入网络，保证网络中的路由拥塞或链路不致过载
- 拥塞往往表现为通信时延的增加

