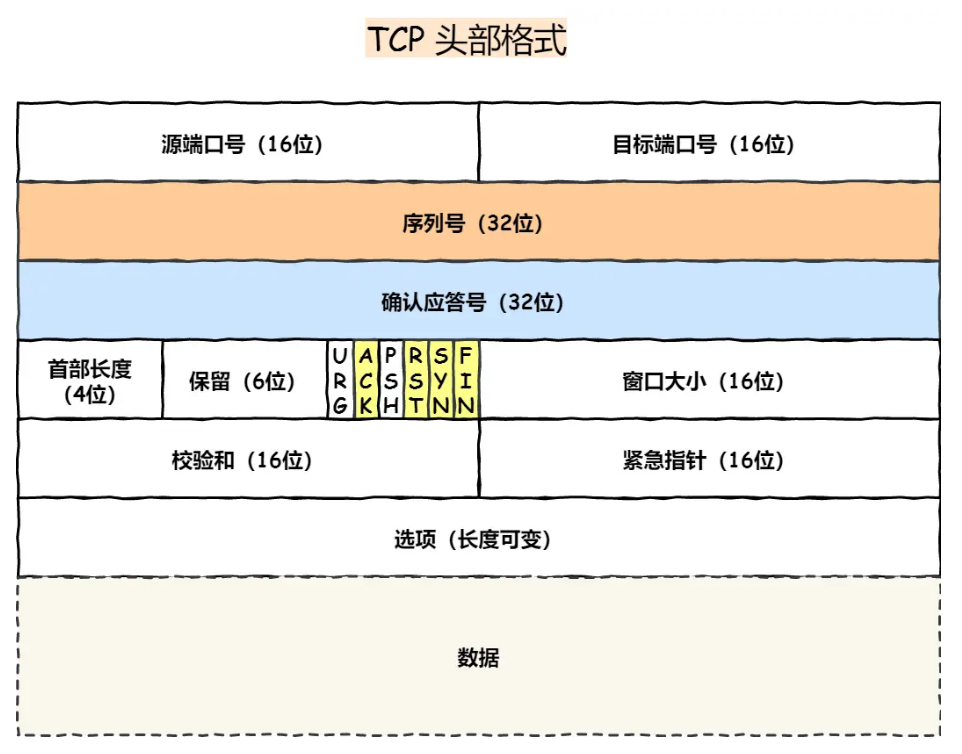
1. TCP基本认识
2. TCP头格式



序列号：在建立连接时由计算机生成的随机数作为初始值，通过SYN包传给接收端主机，每发送一次数据就累加一次该数据字节数的大小。用来解决网络包乱序的问题

确认号：下一次期望收到的数据的序列号，发送端收到这个确认号后可以认为在这个序号之前的数据都已经正常接收。用来解决丢包的问题。

控制位：

SYN：该位为1表示希望建立连接，并在其序列号字段进行序列号初始值的设定。

ACK：该位为1时确认应答的字段变为有效，TCP规定除了最初建立连接的SYN包外该位必须为1

FIN：该位为1时表示今后不会再有数据发送，希望断开连接。

RST：该位为1时表示TCP连接中出现异常必须断开连接

1. 为什么需要TCP协议？TCP工作在哪一层？（传输层）

IP层是不可靠的，不保证网络包的交付、按序交付和数据的完整性

TCP是一个工作在传输层的可靠数据传输的服务，保证接收端的网络包是无损坏、无间隔、非冗余和按序的。

1. 什么是TCP

TCP是面向连接的、可靠的、基于字节流的传输层通信协议

面向连接：一对一才能连接（不能一对多）

可靠的：无论网络链路出现什么变化都可以保证一个报文一定能够达到接收端

字节流：通过TCP传输的消息可能会被操作系统分组成多个TCP报文，如果接收方不知道消息的边界是无法读出一个有效的消息的。且TCP报文是有序的，当前一个TCP报文没有收到时，后面的报文会被放到缓存中，但也不能给应用层处理，同时对重复的TCP报文会自动丢弃。

1. 什么是TCP连接？

连接：用于保证可靠性和流量控制维护的某些状态信息，这些信息的组合包括Socket、序列号和窗口大小称为连接

TCP连接需要客户端和服务端达成三个信息的共识：

Socket：由IP和端口号组成

序列号：用来解决乱序问题

窗口大小：用来做流量控制

1. 如何唯一确定一个TCP连接？

TCP四元组可以唯一确定一个连接，四元组包括：源地址、源端口、目的地址、目的端口。

源地址和目的地址的字段（32位）在IP头部中，作用是通过IP协议发送报文给对方主机

源端口和目的端口在TCP头部中，作用是告诉TCP协议应该把报文发给哪个进程

一个IP服务端监听了一个端口，它的TCP最大连接数理论上为：客户的IP数 \* 客户端的端口数，对IPV4，IP数为2^32 端口数为2^16

服务端实际的TCP连接数远不能达到理论上限，受到以下因素影响：

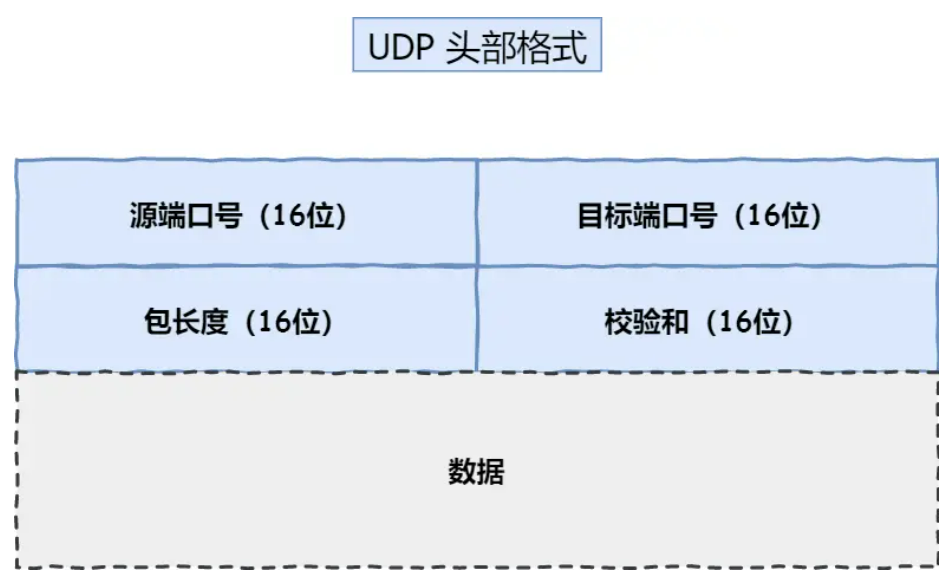
文件描述符限制：每一个TCP都是一个文件，如果文件描述符被占满会发送Too many open files，Linux对可打开的文件描述符有系统级、用户级、进程级的限制

内存限制：每个TCP连接都要占用一定内存，操作系统内存是有限的，内存被占满会发生OOM（out of memory）

1. UDP和TCP的区别和应用场景

UDP不提供复杂的控制机制，利用IP提供面向无连接的通信服务

UDP的头部格式：



源端口和目的端口号：告诉UDP协议应该把报文发送给哪个进程

包长度：该字段保存了UDP首部的长度和数据长度之和

校验和：防止收到网络传输中受损的UDP包

TCP和UDP的区别：

1. 连接：

TCP是面向连接的传输层协议，传输前要建立连接

UDP不需要连接，立即传输数据

1. 服务对象：

TCP是一对一的两点服务

UDP支持一对一、一对多、多对多

1. 可靠性：

TCP提供可靠交付，数据可以保证无差错、不丢失、不重复、按序到达

UDP是尽最大努力交付

1. 拥塞控制、流量控制

TCP独有

UDP即使网络非常拥堵也不会影响发送速率

1. 首部开销

TCP首部较长，没有使用选项字段时是20个字节

UDP首部8字节

1. 传输方式

TCP流式传输，没有边界但保证顺序可靠

UDP是一个包一个包的发送，有边界但可能会丢包和乱序

1. 分片不同

TCP数据大于MSS会在传输层进行分片，目标主机收到后也在传输层组装，中途丢失分片只需重传这个分片

UDP数据大于MTU在IP层进行分片

1. TCP和UDP应用场景

TCP经常用于HTTP/HTTPS，FTP

UDP经常用于包总量较少的通信如DNS，SNMP

1. 为什么TCP有首部长度字段UDP没有：TCP有可变长的选项字段，UDP头部长度不会变化
2. 为什么UDP有包长度字段，TCP没有

TCP数据的长度=IP总长度 – IP首部长度 -TCP首部长度，UDP也可以这么计算

UDP包长度的原因：一种说法：为了将UDP首部补充至4字节的整数倍；另一种说法：如今UDP协议是基于IP协议发展的，当年不是

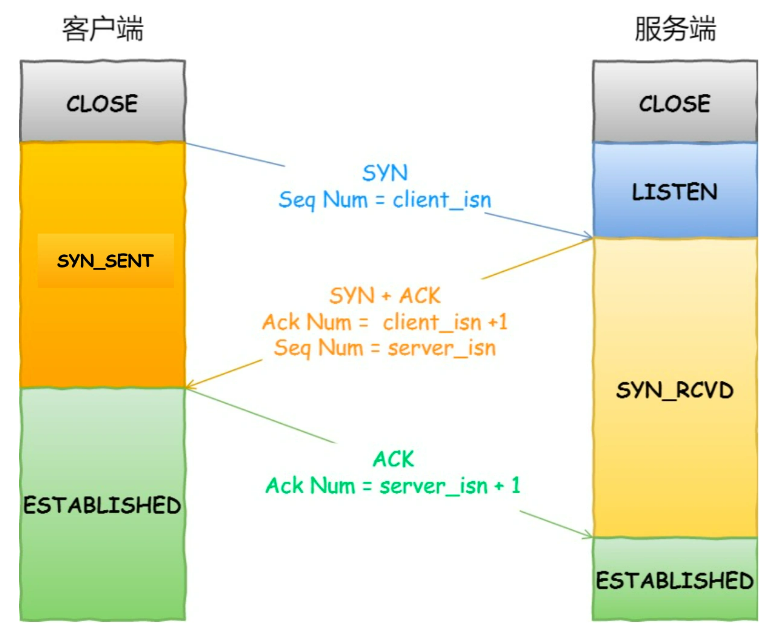
1. TCP和UDP可以共用一个端口吗？

可以，因为在IP包头中协议号字段表明了该数据包使用了TCP还是UDP，根据这个信息送往对应的模块处理，再根据端口号确定送给哪个应用程序，因此TCP和UDP的端口相互独立

1. 多个TCP服务进程可以绑定同一个端口吗？

两个TCP服务进程同时绑定的IP地址和端口相同会出错，IP地址不同端口相同不会出错（0.0.0.0除外，因为代表任意地址）

1. TCP连接建立
2. 三次握手过程



* 1. 一开始客户端和服务端都出于CLOSE状态，服务器先监听某个端口，出于LISTEN状态
  2. 客户端随机初始化序列号client\_isn，将SYN标志位置为1，将SYN报文发送给服务端，表示向服务端发起连接，该报文不含应用层数据，之后客户端进入SYN-SENT状态
  3. 服务端收到客户端SYN报文后，初始化自己的序号server\_isn，将确认号字段填入client\_isn+1，将SYN和ACK标志位置为1，发送个客户端，该报文也不包含应用层数据，之后服务端出于SYN-RCVD状态
  4. 客户端收到后还要向服务端回应最后一个应答报文，该应答报文首部ACK标志位置为1，确认号为server\_isn+1，最后把报文发送给服务端，这次报文可以携带客户端到服务端的数据，之后客户端出于ESTABLISHED状态
  5. 服务端收到回复后也处于ESTABLISHED状态

第三次握手是可以携带数据的

1. 如何在Linux系统中查看TCP状态：netstat -napt
2. 为什么是三次握手不是两次或四次：两次不够，三次就可以完成四次的工作

一方面三次握手才能保证双方具有接收和发送的能力

另一方面：

1. 阻止重复历史连接的初始化（主要原因）（客户端发了两次SYN，旧的先到达，服务器回复的确认号为旧的，客户端希望的是新的，这时会发送RST，如果有三次握手那么两端都不会建立连接（进入ESTABLISHED），如果没有三次握手，客户端的RST到达之前服务端建立了连接会造成资源浪费）
2. 同步双方的初始序列号

序列号的作用：接收方可以去除重复的数据、接收方可以根据数据包的序列号按序接收、可以标识发送的数据包中哪些已经被对方接收了

第一次握手发送客户端的序列号，第二次握手服务端确认了客户端的序列号，但需要第三次握手服务端才知道客户端确认了自己的序列号

4. 为什么每次建立TCP连接时初始化的序列号都要求不一样

1.防止历史报文被下一个相同的四元组的连接接收

2.为了安全性，防止伪造的相同序列号的TCP报文被接收

5. 初始序列号ISN是如何随机产生的：基于时钟递增

6. 既然IP层会分片，为什么TCP层还需要MSS

MTU：一个网络包的最大长度，以太网中一般1500字节

MSS：出去IP头和TCP头后一个网络包能容纳的TCP数据的最大长度

如果只用MTU分片，当一个IP分片丢失整个IP报文的所有分片都要重传