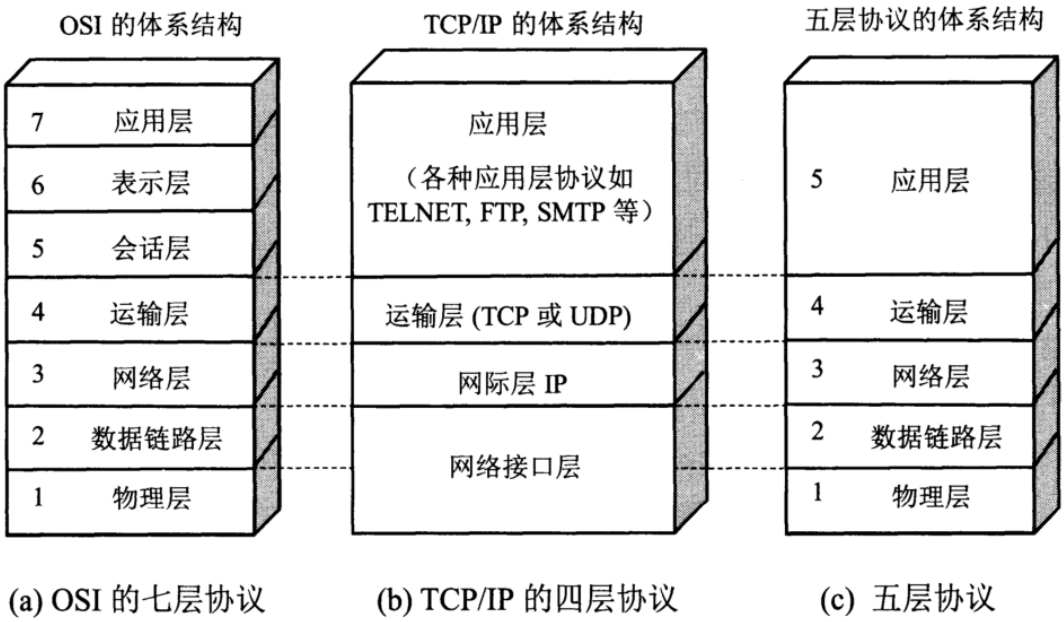
概括



七层网络体系结构各层的主要功能：

- 应用层：为应用程序提供交互服务。在互联网中的应用层协议很多，如域名系统DNS，支持万维网应用的HTTP协议，支持电子邮件的SMTP协议等。
- 表示层：主要负责数据格式的转换，如加密解密、转换翻译、压缩解压缩等。
- 会话层：负责在网络中的两节点之间建立、维持和终止通信，如服务器验证用户登录便是由会话层完成的。
- 传输层：从主机精确到进程 TCP UDP
- 网络层：选择合适的路由和交换结点 主机点到点通信
- 数据链路层：数据链路层通常简称为链路层。将网络层传下来的IP数据报组装成帧，并再相邻节点的链路上传送帧。
- `物理层`：实现相邻节点间比特流的透明传输，尽可能屏蔽传输介质和通信手段的差异。

报文 应用层数据信息

段 传输层 tcp是报文段 udp是整个报文 受到MSSMaxitum Segment Sizex限制

数据报 网络层 可以分为分片 受到MTU Maximum Transmission Unit 链路层的大小限制

帧 链路层 数据报传到链路层分片或者一整个数据报叫做分组 将分组封装成帧

应用层

HTTP常见的状态码有哪些

	类别	原因短语
1XX	Informational（信息性状态码）	接收的请求正在处理
2XX	Success（成功状态码）	请求正常处理完毕
3XX	Redirection（重定向状态码）	需要进行附加操作以完成请求
4XX	Client Error（客户端错误状态码）	服务器无法处理请求
5XX	Server Error（服务器错误状态码）	服务器处理请求出错

200 404 403(收到拒绝) 400(语法错误) 500

301 ：(永久移动) 请求的网页已永久移动到新位置。 服务器返回此响应(对 GET 或 HEAD 请求的响应)时，会自动将请求者转到新位置。

302：(临时移动) 服务器目前从不同位置的网页响应请求，但请求者应继续使用原有位置来进行以后的请求。

HTTP 常用的请求方式

方法	作用
GET	获取资源
POST	传输实体主体
PUT	上传文件
DELETE	删除文件
HEAD	和GET方法类似，但只返回报文首部，不返回报文实体主体部分
PATCH	对资源进行部分修改
OPTIONS	查询指定的URL支持的方法
CONNECT	要求用隧道协议连接代理
TRACE	服务器会将通信路径返回给客户端

GET请求和POST请求的区别

使用上的区别:

- URL 和BODY
- 数据长度
- 安全 地址栏可见

本质区别

GET请求是幂等性

HTTP请求报文和响应报文的格式

请求报文格式:

1. 请求行（请求方法+URI协议+版本）
2. 请求头部

3. 空行
4. 请求主体

响应报文:

1. 状态行（版本+状态码+原因短语）
2. 响应首部
3. 空行
4. 响应主体

Http1.1与1.0的区别

1.长连接 请求头Connection： keep-alive

2 流水线，客户端先进先出变为服务器端先进先出，客户端无需等待回应即可发第二个请求，但是服务器依然要按顺序

3.节约带宽 先发请求头回复100 ， 再发请求体

4.请求头增加host字段 关于缓存的字段

http1.0和2.0的区别

1.多路复用，一个连接中并发多个请求或回应，通过数据帧的stream id来区分不同请求

2.头部压缩

3.二进制抛弃ascll 采用头信息帧和数据帧

4.服务器推送 把css js等资源主动发送到本地并缓存

https与http区别

1.安全性

2.端口80和443

3.三次握手后增加ssl/tls握手 Secure Socket Layer Transport Layer Security

4.ca获取安全证书

加密，摘要算法，安全证书解决了窃听，篡改，冒充风险

https过程

前置准备:服务器的公钥交给CA，CA使用秘钥加密，生成证书

tcp三次握手

客户端访问服务器，给出协议版本号、Client random，以及客户端支持的加密方法。(client hello)

服务器确认双方使用的加密方法，并给出数字证书、Server random。(server hello)

客户端使用CA公钥解密证书获取服务器公钥并加密key返回服务器

服务器私钥解密，这样客户端和服务器同时知道Client random，Server random，key生成对称加密的对话秘钥，开始通信

Cookie和Session的区别

- 作用范围不同，Cookie 保存在客户端（浏览器），Session 保存在服务器端。

- 存取方式的不同，Cookie 只能保存 ASCII，Session 可以存任意数据类型，一般情况下我们可以在 Session 中保持一些常用变量信息，比如说 UserId 等。
- 有效期不同，Cookie 可设置为长时间保持，比如我们经常使用的默认登录功能，Session 一般失效时间较短，客户端关闭或者 Session 超时都会失效。
- 隐私策略不同，Cookie 存储在客户端，比较容易遭到不法获取，早期有人将用户的登录名和密码存储在 Cookie 中导致信息被窃取；Session 存储在服务端，安全性相对 Cookie 要好一些。
- 存储大小不同， 单个 Cookie 保存的数据不能超过 4K，Session 可存储数据远高于 Cookie。

cookie session token

1. cookie 客户端保存发送请求携带
2. session 服务器内存保存一份，发给客户端一份，客户端发请求时通过cookie携带，服务器数据库验证是否一致
3. token 服务器生成发给客户端一份，客户端发请求携带，如果是jwt服务器只需要拿秘钥重新计算signature验证签名即可，否则也是类似session在数据库验证

session的痛点及解决方案

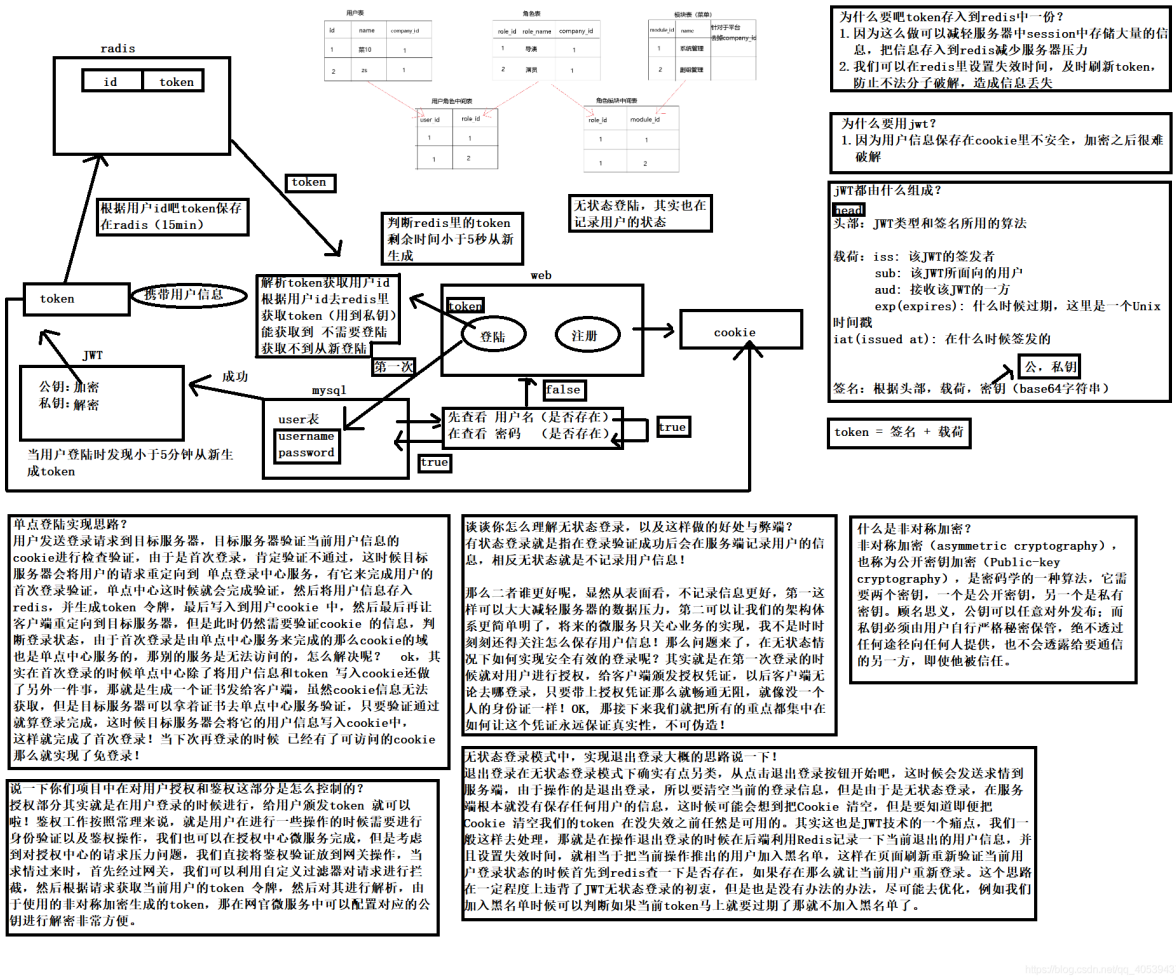
分布式如何保证发往的服务器中存有session

- session复制
- session粘连 只打给保存session的那个服务器
- session共享 将session存在redis 服务器从中获取

token的形式

可以看到 token 主要由三部分组成，中间用.分隔，使用base64编码

1. header：指定了签名算法
2. payload：可以指定用户 id，过期时间等非敏感数据
3. Signature: 签名，server存有秘钥，将1.2用.拼接起来，和秘钥三个一起签名生成摘要



在浏览器中输入网址后执行的全部过程

1.域名解析

浏览器缓存，操作系统缓存，host表中找ip，本地域名服务器缓存，本地域名服务器迭代查询，返回ip并缓存

2.发http请求 分配端口tcp三次握手 路由选择协议到达并arp解析mac地址

3.服务器响应请求返回html文件

4.解析html同时请求资源

5.渲染

DDos攻击

多台肉鸡进行恶意对服务器进行恶意连接，如慢连接，畸形报文，synflood

传输层

TCP和UDP的区别

- 面向连接
- 一对多和一对一
- 可靠传输 流量控制拥塞控制
- 首部字节 8,20
- 有序无序
- 传输方式报文和流

- 应用场景

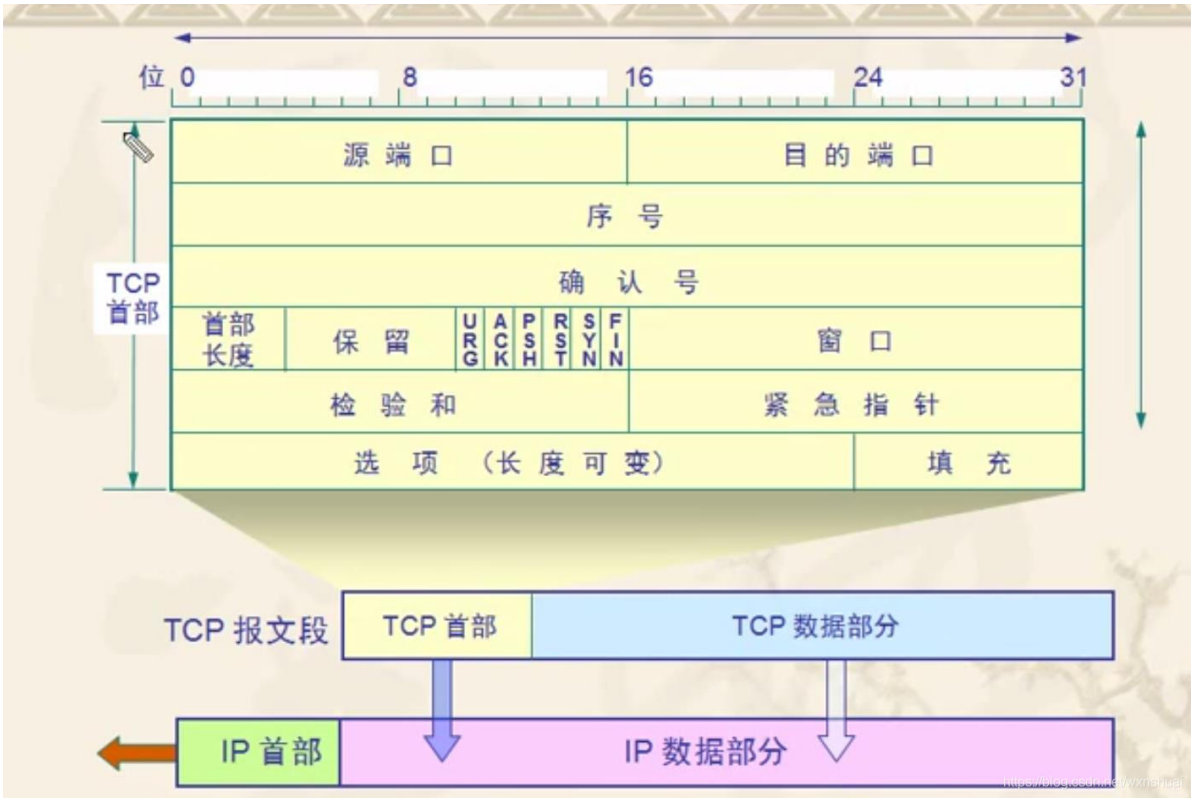
UDP 和 TCP 对应的应用场景是什么

tcp 可靠交付 smtp telnet ftp http

udp高效简单 dns 媒体广播

TCP和udp报文格式

20B

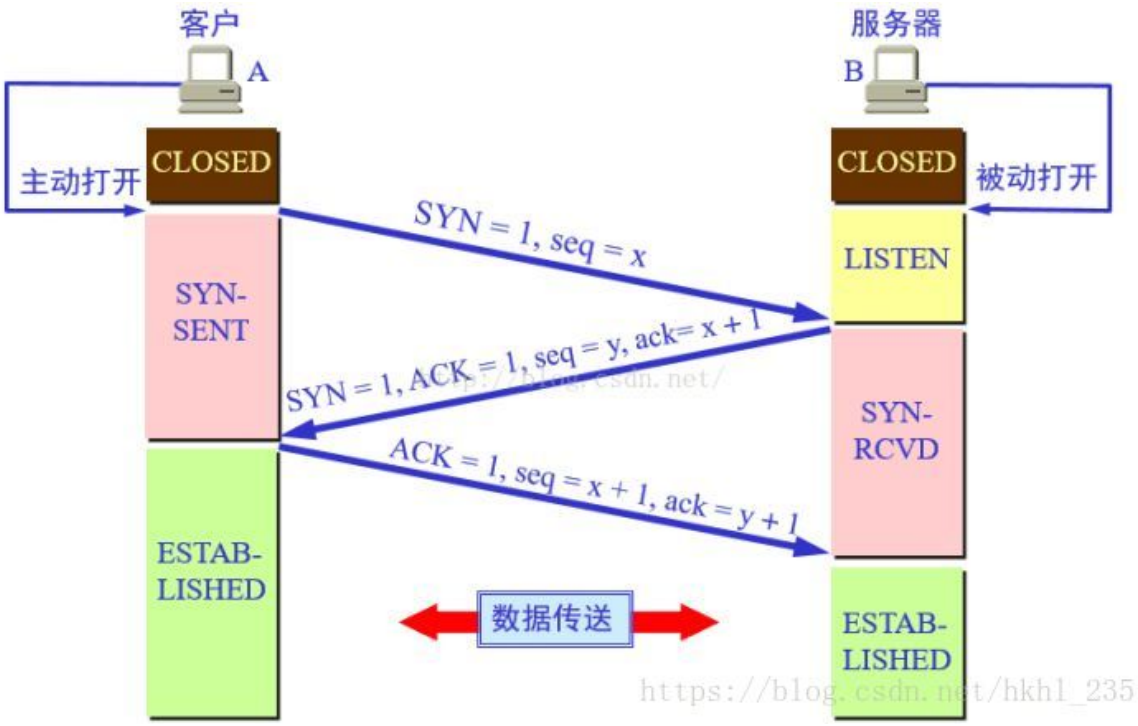


首部长度 指出TCP报文首部含选项时的长度，没有选项时为5，字长4位 ACK 为为1时，确认应答的字段变为有效。TCP规定除了在最初建立连接时候的SYN包之外该位必须设置为1 PSH 该位为1时，表示需要将收到的数据立刻上传给上层应用协议。PSH为0时，则不需要立即传，而是先进行缓存。 RST 该位为1时，表示TCP连接出现异常，必须强制断开连接。 FIN 该位为1时，表示今后都不会再有数据发送，希望断开连接。当通信结束希望断开连接时，通信双方的主机之间就可以相互交换FIN位置为1的TCP段。(每个主机又对对方的FIN包进行确认应答以后就可以断开连接了。不过主机收到FIN设置为1的TCP段以后不必马上回复一个FIN包，而是可以等到缓冲区中的所有数据都因已成功发送而被自动删除之后再发。) 窗口 用来让对方设置发送窗口的依据（告诉对方自己能接受多少数据），2字节 检验和 检验和字段检验的范围包括首部和数据这两个部分。在计算检验和是，要在TCP报文段的前面加上12字节的伪首部 紧急指针 指出在本报文段紧急数据共多少个字节（紧急数据放在本报文段数据最前面）

8B

源端口号16位	目的端口号16位
udp长度	udp校验和
数据	

三次握手机制



syn=1同步报文 seq=x

server syn=1 同步报文，seq=y，ack=x+1 分配缓存和变量

seq=x+1,ack=y+1 分配缓存和变量

两次可以吗

导致过期连接生效 客户端不响应，而服务器一直等待

发消息没用，只有收到消息才能确认信息，

保证ack和seq递增维持可靠传输

最后一次ACK包丢失，会发生什么?

根据此可以syn flood攻击

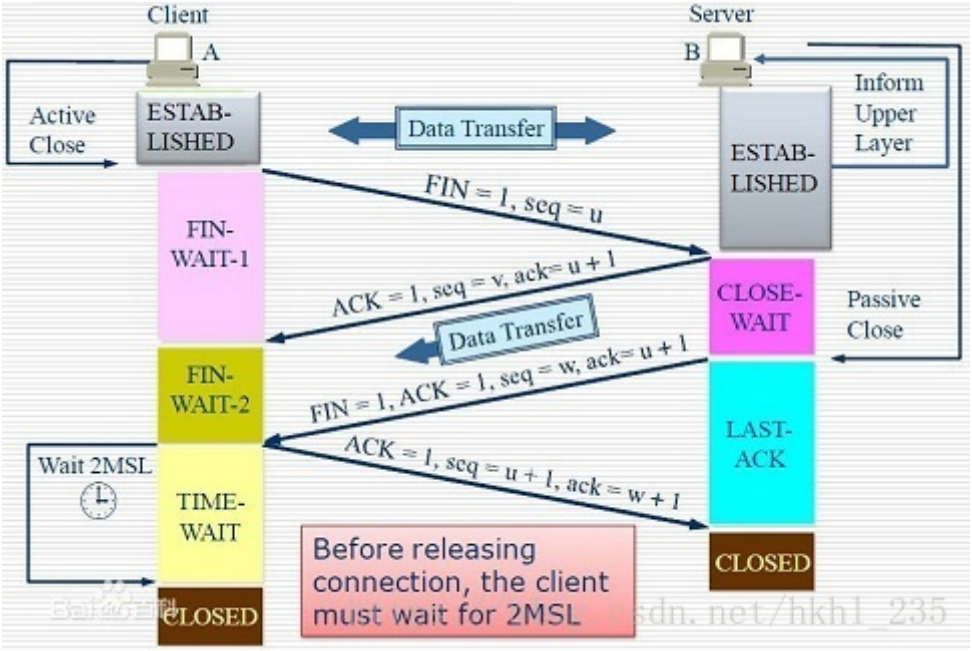
服务端:

- 第三次的ACK在网络中丢失，那么服务端该TCP连接的状态为SYN_RECV,并且会根据 TCP的超时重传机制，会等待3秒、6秒、12秒后重新发送SYN+ACK包，以便客户端重新发送ACK包。
- 如果重发指定次数之后，仍然未收到 客户端的ACK应答，那么一段时间后，服务端自动关闭这个连接。

客户端：

客户端认为这个连接已经建立，如果客户端向服务端发送数据，服务端将以RST包（Reset，标示复位，用于异常的关闭连接）响应。此时，客户端知道第三次握手失败。

四次挥手



c->s ack=1 fin=1 seq=x,ack=y

s->c ack=1,seq=y,ack=x+1 发送数据

s->c ack=1 fin=1 seq=y+n ack=x+1

c->s ack=1 seq =x+1,ack=y+n+1

由于c不知道s是否收到ack，则需要等待2**MSL(Maximum Segment Life)**

因为s收到c需要msl 而如果超过这个时间没收到s会重新发送ack给c也需要msl时间

而且等待这么长时间可以让本连接产生的所有报文段消失，新连接中不会有旧请求报文段

如果s收到了 则s立刻断开连接，否则超时重传，确保一定能收到。

为什么连接的时候是三次握手，关闭的时候却是四次握手

因为三次握手的第二次不仅有ack还有syn

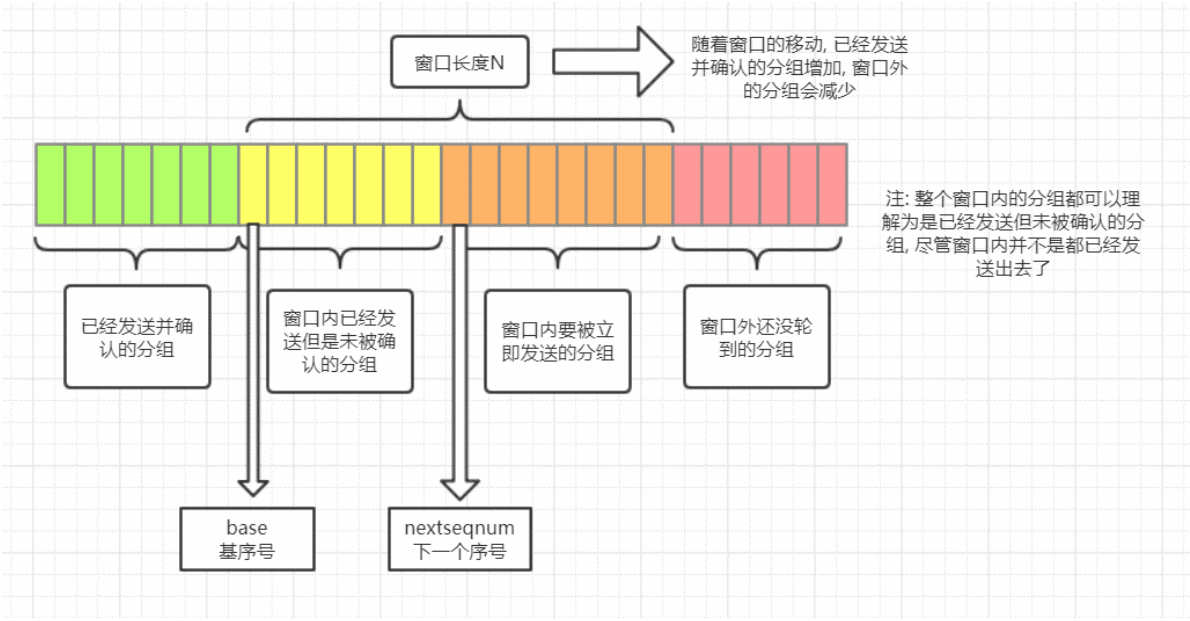
而四次挥手由于s要发数据，fin和ack分开做了两次

TCP可靠传输怎么实现

1. 检验和
2. seq和ack
3. 超时重传
4. 拥塞控制
5. 流量控制

滑动窗口

因为tcp面向字节流，发送的是拆分的数据块



拥塞控制

拥塞窗口和接收窗口

拥塞是网络环境状态的反映，接收是接收能力容量的反映

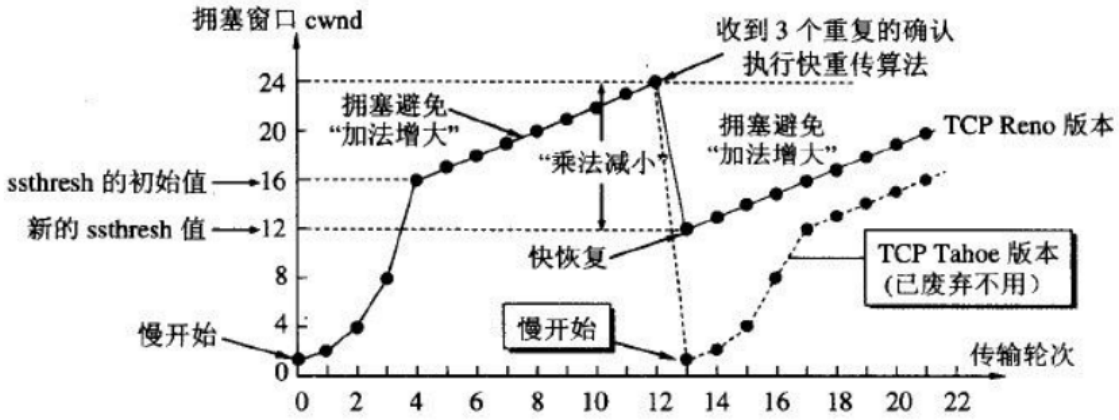


图 5-27　从连续收到三个重复的确认转入拥塞避免

慢开始 一开始报文段发送从1开始指数增长

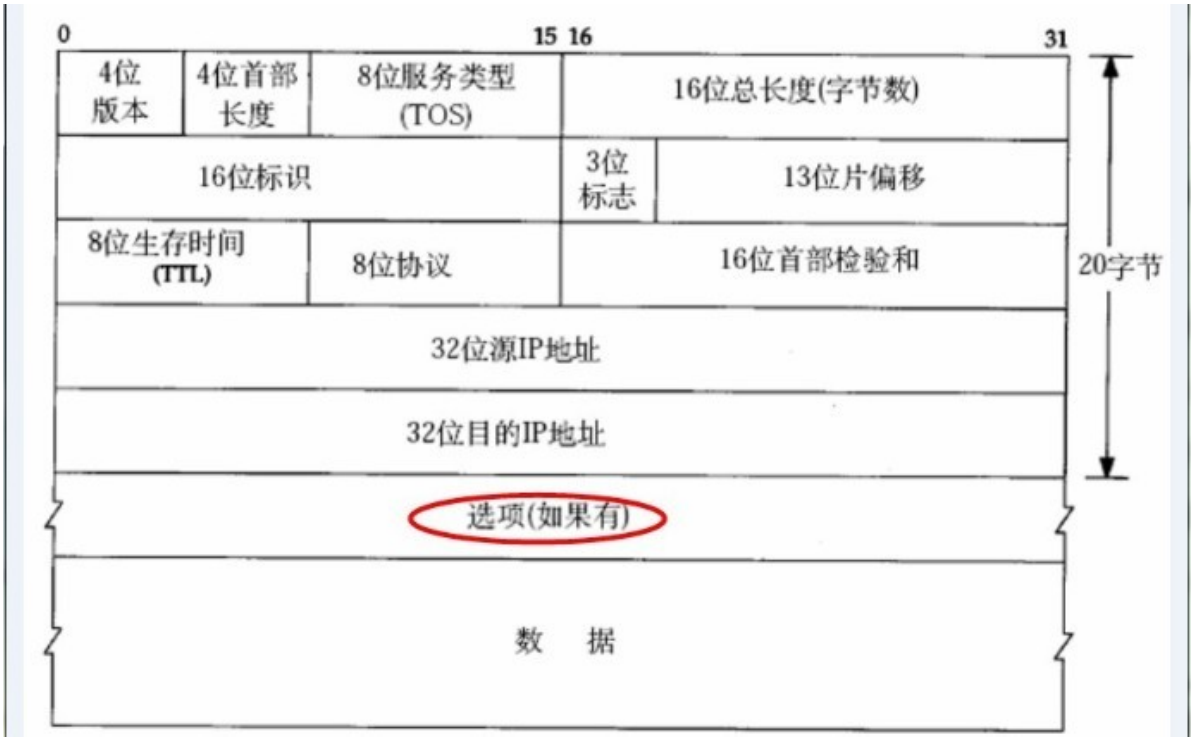
拥塞避免 到达拥塞避免预设值开始线性增长 到达拥塞窗口值后拥塞避免值减半

快重传 不等待重传计时 到达拥塞窗口后接收到三个冗余ack，即有数据块丢失后，立即重传

快恢复 从新设置的拥塞避免预设值线性增长

网络层

ip数据报格式


0
15 16
31
4位
版本
4位首部
长度
8位服务类型
(TOS)
16位总长度(字节数)
16位标识
3位
标志
13位片偏移
8位生存时间
(TTL)
8位协议
16位首部检验和
20字节
32位源IP地址
32位目的IP地址
选项(如果有)
数 据