计算机网络核心知识点

关注微信公众号“大厂offer宝典”，获取更多面试技巧及资料

计算机网络：

计算机网络

什么是网络协议，为什么要对网络协议分层 *

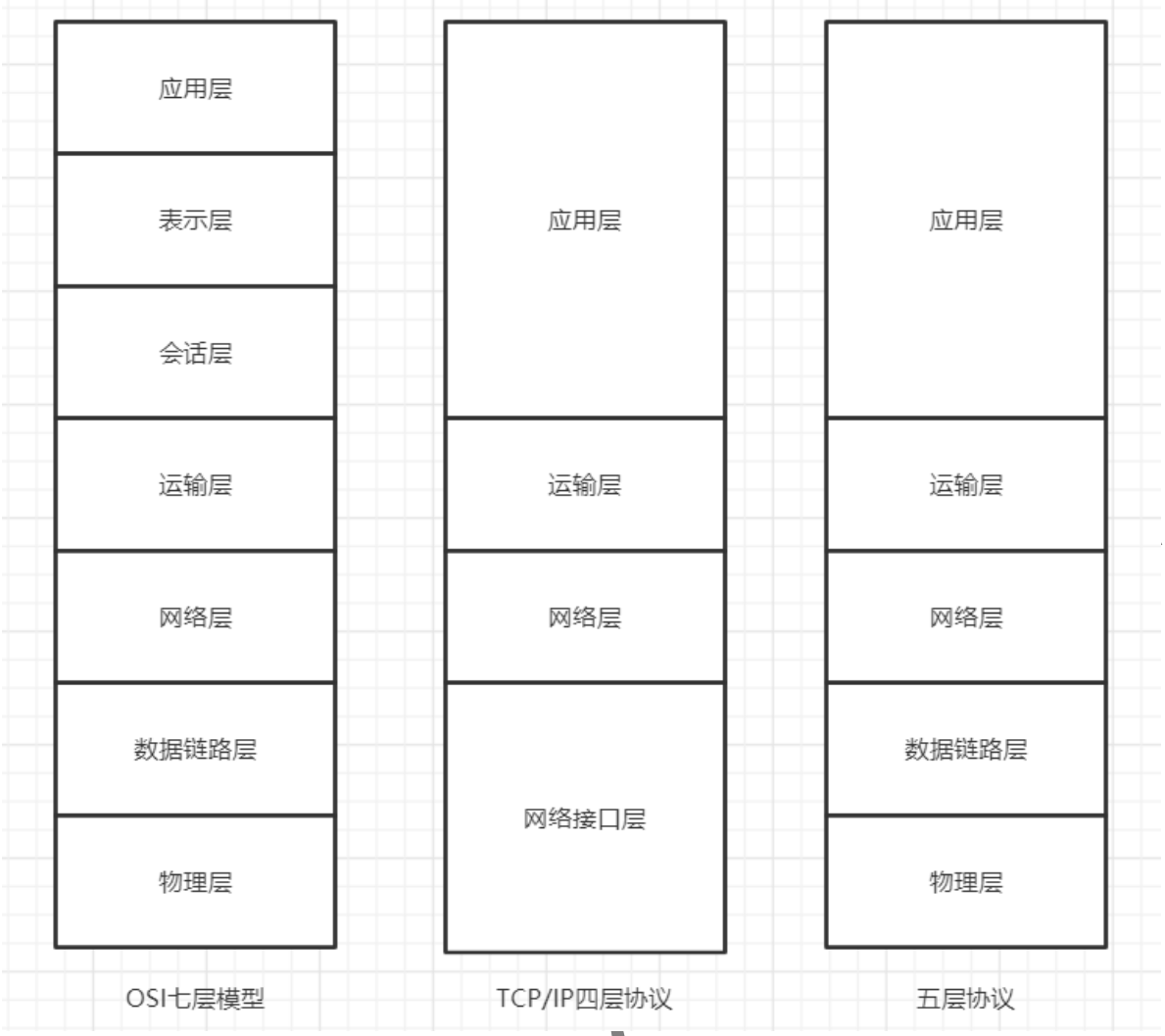
网络协议是计算机在通信过程中要遵循的一些约定好的规则。

网络分层的原因：

- 易于实现和维护，因为各层之间是独立的，层与层之间不会收到影响。
- 有利于标准化的制定

计算机网络的各层协议及作用 * * *

> 计算机网络体系可以大致分为一下三种，七层模型、五层模型和TCP/IP四层模型，一般面试能流畅回答出五层模型就可以了，表示层和会话层被问到的不多。



- 应用层

  应用层的任务是通过应用进程之间的交互来完成特定的网络作用，常见的应用层协议有域名系统DNS，HTTP协议等。

- 表示层

  表示层的主要作用是数据的表示、安全、压缩。可确保一个系统的应用层所发送的信息可以被另一个系统的应用层读取。

- 会话层

  会话层的主要作用是建立通信链接，保持会话过程通信链接的畅通，同步两个节点之间的对话，决定通信是否被中断以及通信中断时决定从何处重新发送。。

- 传输层

  传输层的主要作用是负责向两台主机进程之间的通信提供数据传输服务。传输层的协议主要有传输控制协议TCP和用户数据协议UDP。

- 网络层

  网络层的主要作用是选择合适的网间路由和交换结点，确保数据及时送达。常见的协议有IP协议。

- 数据链路层

  数据链路层的作用是在物理层提供比特流服务的基础上，建立相邻结点之间的数据链路，通过差错控制提供数据帧（Frame）在信道上无差错的传输，并进行各电路上的动作系列。 常见的协议有SDLC、HDLC、PPP等。

- 物理层

  物理层的主要作用是实现相邻计算机结点之间比特流的透明传输，并尽量屏蔽掉具体传输介质和物理设备的差异。

URI和URL的区别 *

- URI(Uniform Resource Identifier)：中文全称为统一资源标志符，主要作用是唯一标识一个资源。
- URL(Uniform Resource Location)：中文全称为统一资源定位符，主要作用是提供资源的路径。

有个经典的比喻是URI像是身份证，可以唯一标识一个人，而URL更像一个住址，可以通过URL找到这个人。

X

DNS的工作流程 * * *

DNS的定义：DNS的全称是domain name system，即域名系统。DNS是因特网上作为域名和IP地址相互映射的一个分布式数据库，能够使用户更方便的去访问互联网而不用去记住能够被机器直接读取的IP地址。比如大家访问百度，更多地肯定是访问www.baidu.com，而不是访问112.80.248.74，因为这几乎无规则的IP地址实在太难记了。DNS要做的就是将www.baidu.com解析成112.80.248.74。

DNS是集群式的工作方式还是 单点式的，为什么?

答案是集群式的，很容易想到的一个方案就是只用一个DNS服务器，包含了所有域名和IP地址的映射。尽管这种设计方式看起来很简单，但是缺点显而易见，如果这个唯一的DNS服务器出了故障，那么就全完了，因特网就几乎崩了。为了避免这种情况出现，DNS系统采用的是分布式的层次数据数据库模式，还有缓存的机制也能解决这种问题。

DNS的工作流程

> **主机向本地域名服务器的查询一般是采用递归查询，而本地域名服务器向根域名的查询一般是采用迭代查询。**
>
> 递归查询主机向本地域名发送查询请求报文，而本地域名服务器不知道该域名对应的IP地址时，本地域名会继续向根域名发送查询请求报文，不是通知主机自己向根域名发送查询请求报文。迭代查询是，本地域名服务器向根域名发出查询请求报文后，根域名不会继续向顶级域名服务器发送查询请求报文，而是通知本地域名服务器向顶级域名发送查询请求报文。
>
> 简单来说，递归查询就是，小明问了小红一个问题，小红不知道，但小红是个热心肠，小红就去问小王了，小王把答案告诉小红后，小红又去把答案告诉了小明。迭代查询就是，小明问了小红一个问题，小红也不知道，然后小红让小明去问小王，小明又去问小王了，小王把答案告诉了小明。

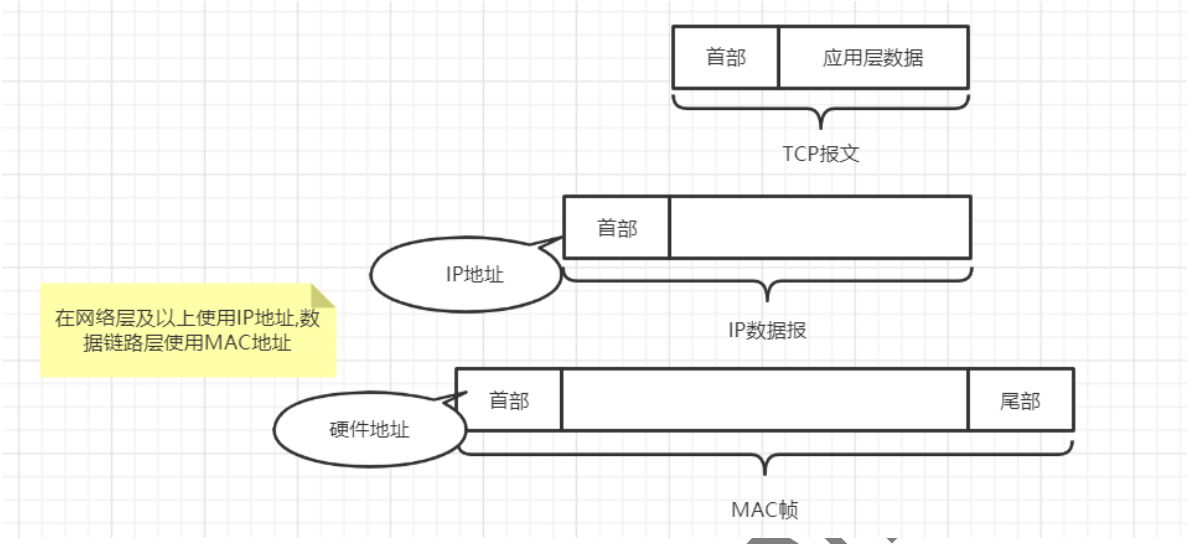
1. 在浏览器中输入www.baidu.com域名，操作系统会先检查自己本地的hosts文件是否有这个域名的映射关系，如果有，就先调用这个IP地址映射，完成域名解析。
2. 如果hosts文件中没有，则查询本地DNS解析器缓存，如果有，则完成地址解析。
3. 如果本地DNS解析器缓存中没有，则去查找本地DNS服务器，如果查到，完成解析。
4. 如果没有，则本地服务器会向根域名服务器发起查询请求。根域名服务器会告诉本地域名服务器去查询哪个顶级域名服务器。
5. 本地域名服务器向顶级域名服务器发起查询请求，顶级域名服务器会告诉本地域名服务器去查找哪个权限域名服务器。
6. 本地域名服务器向权限域名服务器发起查询请求，权限域名服务器告诉本地域名服务器www.baidu.com所对应的IP地址。
7. 本地域名服务器告诉主机www.baidu.com所对应的IP地址。

了解ARP协议吗? * *

ARP协议属于网络层的协议，主要作用是实现从IP地址转换为MAC地址。在每个主机或者路由器中都建有一个ARP缓存表，表中有IP地址及IP地址对应的MAC地址。先来看一下什么时IP地址和MAC地址。

- IP地址：IP地址是指互联网协议地址，IP地址是IP协议提供的一种统一的地址格式，它为互联网上的每一个网络和每一台主机分配一个逻辑地址，以此来屏蔽物理地址的差异。
- MAC地址：MAC地址又称物理地址，由网络设备制造商生产时写在硬件内部，不可更改，并且每个以太网设备的MAC地址都是唯一的。

数据在传输过程中，会先从高层传到底层，然后在通信链路上传输。从下图可以看到TCP报文在网络层会被封装成IP数据报，在数据链路层被封装成MAC帧，然后在通信链路中传输。在网络层使用的是IP地址，在数据据链路层使用的是MAC地址。MAC帧在传送时的源地址和目的地址使用的都是MAC地址，在通信链路上的主机或路由器也都是根据MAC帧首部的MAC地址接收MAC帧。并且在数据链路层是看不到IP地址的，只有当数据传到网络层时去掉MAC帧的首部和尾部时才能在IP数据报的首部中找到源IP地址和目的地址。



网络层实现的是主机之间的通信，而链路层实现的是链路之间的通信，所以从下图可以看出，在数据传输过程中，IP数据报的源地址(IP1)和目的地址(IP2)是一直不变的，而MAC地址(硬件地址)却一直随着链路的改变而改变。

ARP的工作流程(面试时问ARP协议主要说这个就可以了)：

1. 在局域网内，主机A要向主机B发送IP数据报时，首先会在主机A的ARP缓存表中查找是否有IP地址及其对应的MAC地址，如果有，则将MAC地址写入到MAC帧的首部，并通过局域网将该MAC帧发送到MAC地址所在的主机B。
2. 如果主机A的ARP缓存表中没有主机B的IP地址及所对应的MAC地址，主机A会在局域网内广播发送一个ARP请求分组。局域网内的所有主机都会收到这个ARP请求分组。
3. 主机B在看到主机A发送的ARP请求分组中有自己的IP地址，会像主机A以单播的方式发送一个带有自己MAC地址的响应分组。
4. 主机A收到主机B的ARP响应分组后，会在ARP缓存表中写入主机B的IP地址及其IP地址对应的MAC地址。
5. 如果主机A和主机B不在同一个局域网内，即使知道主机B的MAC地址也是不能直接通信的，必须通过路由器转发到主机B的局域网才可以通过主机B的MAC地址找到主机B。并且主机A和主机B已经可以通信的情况下，主机A的ARP缓存表中寸的并不是主机B的IP地址及主机B的MAC地址，而是主机B的IP地址及该通信链路上的下一跳路由器的MAC地址。这就是上图中的源IP地址和目的IP地址一直不变，而MAC地址却随着链路的不同而改变。
6. 如果主机A和主机B不在同一个局域网，参考上图中的主机H1和主机H2，这时主机H1需要先广播找到路由器R1的MAC地址，再由R1广播找到路由器R2的MAC地址，最后R2广播找到主机H2的MAC地址，建立起通信链路。

有了IP地址，为什么还要用MAC地址？ * *

> 简单来说，标识网络中的一台计算机，比较常用的就是IP地址和MAC地址，但计算机的IP地址可由用户自行更改，管理起来相对困难，而MAC地址不可更改，所以一般会把IP地址和MAC地址组合起来使用。具体是如何组合使用的在上面的ARP协议中已经讲的很清楚了。
>
> 那只用MAC地址不用IP地址可不可以呢？其实也是不行的，因为在最早就是MAC地址先出现的，并且当时并不用IP地址，只用MAC地址，后来随着网络中的设备越来越多，整个路由过程越来越复杂，便出现了子网的概念。对于目的地址在其他子网的数据包，路由只需要将数据包送到那个子网即可，这个过程就是上面说的ARP协议。
>
> 那为什么要用IP地址呢？是因为IP地址是和地域相关的，对于同一个子网上的设备，IP地址的前缀都是一样的，这样路由器通过IP地 址的前缀就知道设备在在哪个子网上了，而只用MAC地址的话，路由器则需要记住每个MAC地址在哪个子网，这需要路由器有极大的存储空间，是无法实现的。
>
> IP地址可以比作为地址，MAC地址为收件人，在一次通信过程中，两者是缺一不可的。

说一下ping的过程 * *

ping是ICMP(网际控制报文协议)中的一个重要应用，ICMP是网络层的协议。ping的作用是测试两个主机的连通性。

ping的工作过程：

1. 向目的主机发送多个ICMP回送请求报文
2. 根据目的主机返回的回送报文的时间和成功响应的次数估算出数据包往返时间及丢包率。

路由器和交换机的区别？ *

	所属网络模型的层级	功能
路由器	网络层	识别IP地址并根据IP地址转发数据包，维护数据表并基于数据表进行最佳路径选择
交换机	数据链库层	识别MAC地址并根据MAC地址转发数据帧

TCP与UDP有什么区别 * * *

	是否面向连接	可靠性	传输形式	传输效率	消耗资源	应用场景	首部字节
TCP	面向连接	可靠	字节流	慢	多	文件/邮件传输	20~60
UDP	无连接	不可靠	数据报文段	快	少	视频/语音传输	8

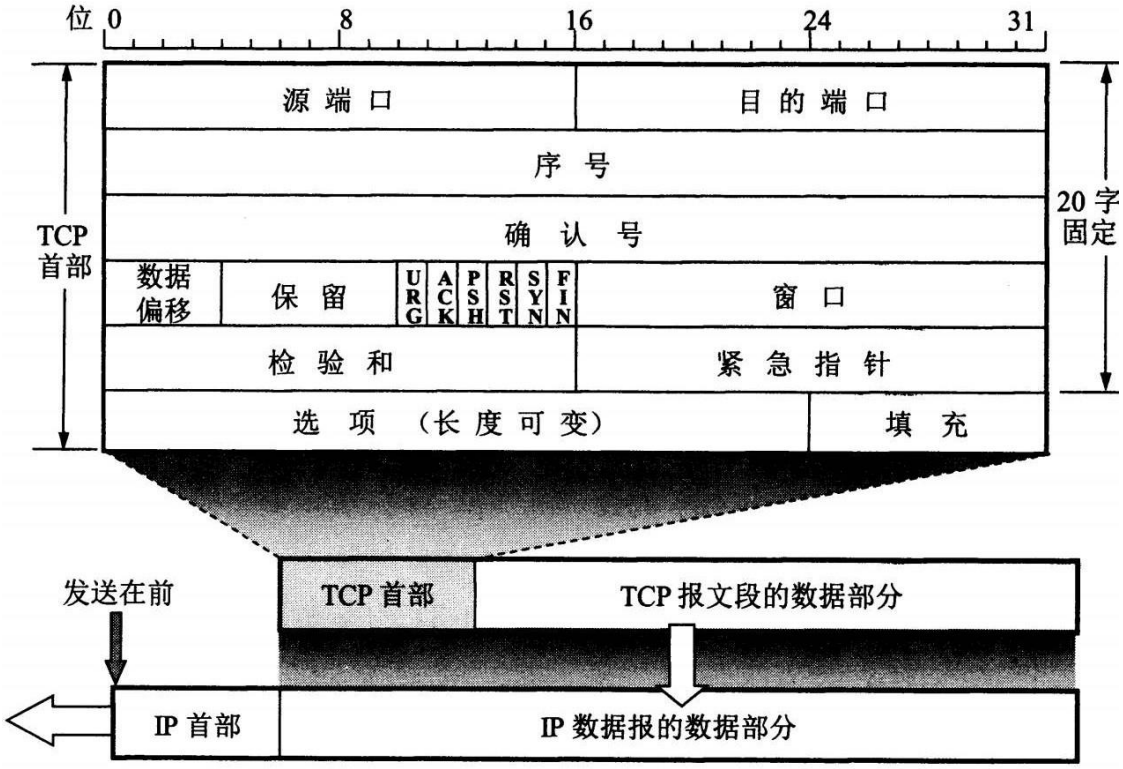
> 有时候面试还会问到TCP的首部都包含什么

- TCP首部(图片来源于网络)：

  前20个字节是固定的，后面有4n个字节是根据需而增加的选项，所以TCP首部最小长度为20字节。

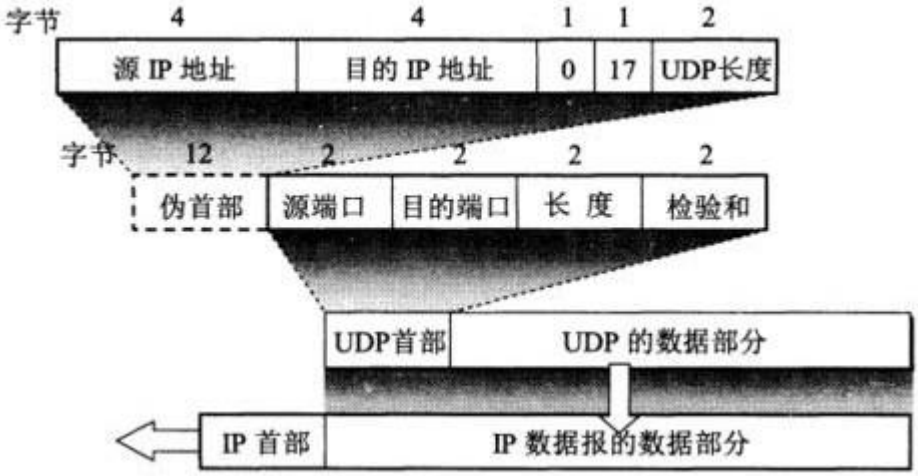
TCP面向字节流的含义：不保证接收方应用程序受到的数据块与发送方应用程序发送的数据块，具有对应大小的关系，但是他们的字节流是一致的。

UDP面向数据报的含义：UDP对于应用层下来的报文，不合并也不拆分，而是保留报文的边界交给IP层，由IP层进行分片



- UDP首部(图片来源于网络):

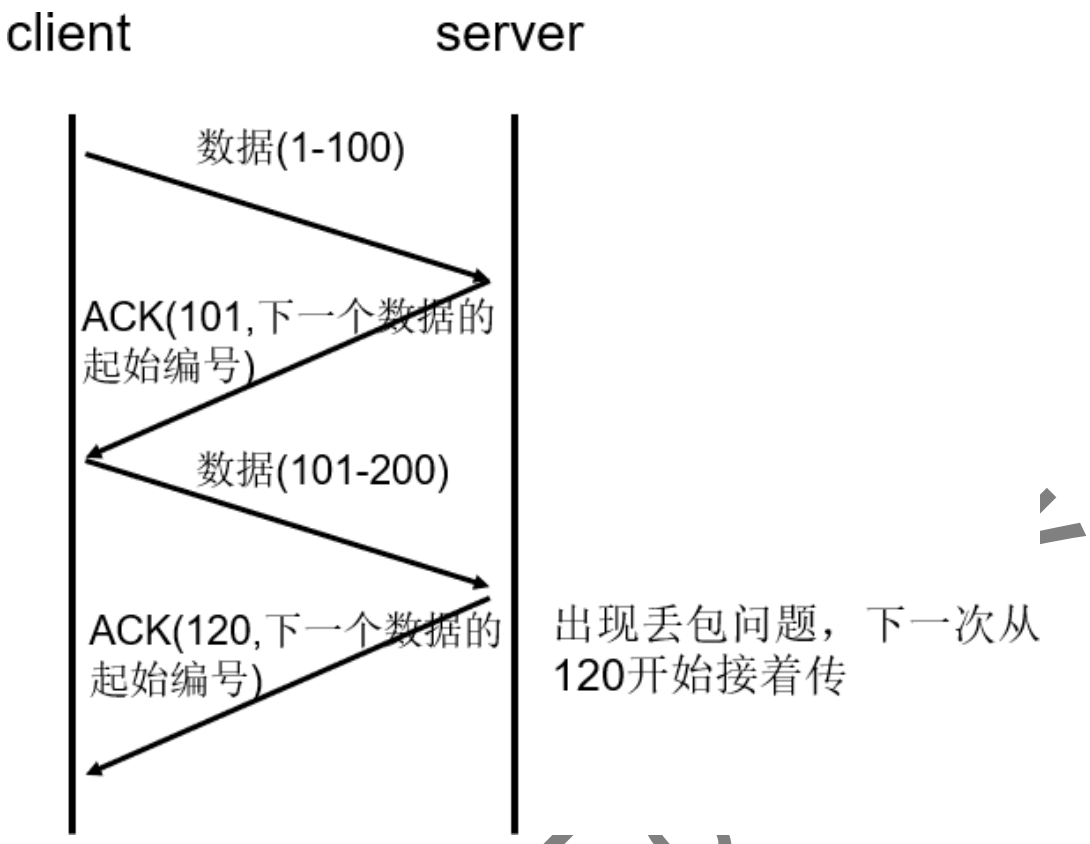
  UDP的首部只有8个字节，源端口号、目的端口号、长度和校验和各两个字节。



TCP协议如何保证可靠传输 * * *

主要有校验和、序列号、超时重传、流量控制及拥塞避免等几种方法。

- 校验和：在发送算和接收端分别计算数据的校验和，如果两者不一致，则说明数据在传输过程中出现了差错，TCP将丢弃和不确认此报文段。
- 序列号：TCP会对每一个发送的字节进行编号，接收方接到数据后，会对发送方发送确认应答(ACK报文)，并且这个ACK报文中带有相应的确认编号，告诉发送方，下一次发送的数据从编号多少开始发。如果发送方发送相同的数据，接收端也可以通过序列号判断出，直接将数据丢弃。如果



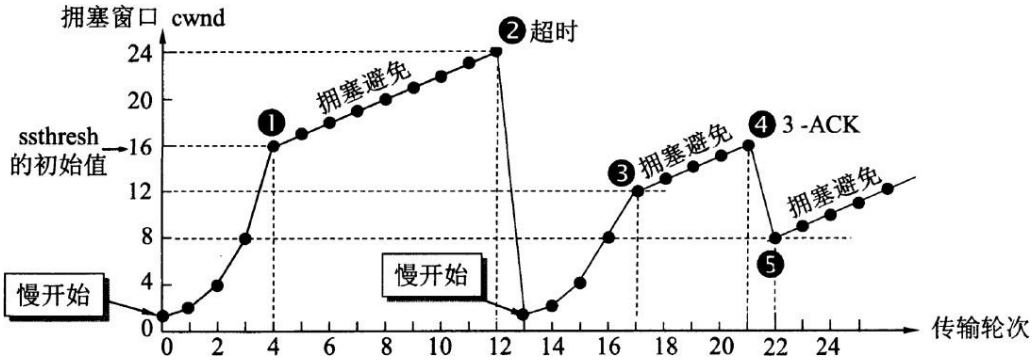
- 超时重传：在上面说了序列号的作用，但如果发送方在发送数据后一段时间内（可以设置重传计时器规定这段时间）没有收到确认序号ACK，那么发送方就会重新发送数据。

  这里发送方没有收到ACK可以分两种情况，如果是发送方发送的数据包丢失了，接收方收到发送方重新发送的数据包后会马上给发送方发送ACK；如果是接收方之前接收到了发送方发送的数据包，而返回给发送方的ACK丢失了，这种情况，发送方重传后，接收方会直接丢弃发送方冲重传的数据包，然后再次发送ACK响应报文。

  如果数据被重发之后还是没有收到接收方的确认应答，则进行再次发送。此时，等待确认应答的时间将会以2倍、4倍的指数函数延长，直到最后关闭连接。

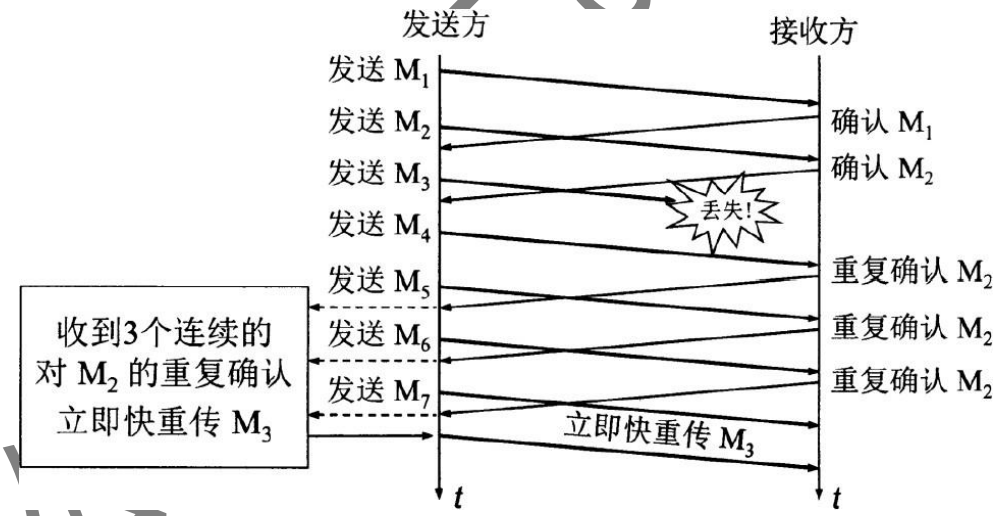
guide的ARQ协议

- 流量控制：如果发送端发送的数据太快，接收端来不及接收就会出现丢包问题。为了解决这个问题，TCP协议利用了滑动窗口进行了流量控制。在TCP首部有一个16位字段大小的窗口，窗口的大小就是接收端接收数据缓冲区的剩余大小。接收端会在收到数据包后发送ACK报文时，将自己的窗口大小填入ACK中，发送方会根据ACK报文中的窗口大小进而控制发送速度。如果窗口大小为零，发送方会停止发送数据。
- 拥塞控制：如果网络出现拥塞，则会产生丢包等问题，这时发送方会将丢失的数据包继续重传，网络拥塞会更加严重，所以在网络出现拥塞时应注意控制发送方的发送数据，降低整个网络的拥塞程度。拥塞控制主要有四部分组成：慢开始、拥塞避免、快重传、快恢复，如下图(图片来源于网络)。



这里的发送方会维护一个拥塞窗口的状态变量，它和流量控制的滑动窗口是不一样的，滑动窗口是根据接收方数据缓冲区大小确定的，而拥塞窗口是根据网络的拥塞情况动态确定的，一般来说发送方真实的发送窗口为滑动窗口和拥塞窗口中的最小值。

1. 慢开始：为了避免一开始发送大量的数据而产生网络阻塞，会先初始化cwnd为1，当收到ACK后到下一个传输轮次，cwnd为2，以此类推成指数形式增长。
2. 拥塞避免：因为cwnd的数量在慢开始是指数增长的，为了防止cwnd数量过大而导致网络阻塞，会设置一个慢开始的门限值ssthresh，当cwnd>=ssthresh时，进入到拥塞避免阶段，cwnd每个传输轮次加1。但网络出现超时，会将门限值ssthresh变为出现超时cwnd数值的一半，cwnd重新设置为1，如上图，在第12轮出现超时后，cwnd变为1，ssthresh变为12。
3. 快重传：在网络中如果出现超时或者阻塞，则按慢开始和拥塞避免算法进行调整。但如果只是丢失某一个报文段，如下图(图片来源于网络)，则使用快重传算法。



从上图可知，接收方正确地接收到M1和M2，而M3丢失，由于没有接收到M3，在接收方收到M5、M6和M7时，并不会进行确认，也就是不会发送ACK。这时根据前面说的保证TCP可靠性传输中的序列号的作用，接收方这时不会接收M5，M6，M7，接收方可以什么都不会，因为发送方长时间未收到M3的确认报文，会对M3进行重传。除了这样，接收方也可以重复发送M2的确认报文，这样发送端长时间未收到M3的确认报文也会继续发送M3报文。

但是根据快重传算法，要求在这种情况下，需要快速向发送端发送M2的确认报文，在发送方收到三个M2的确认报文后，无需等待重传计时器所设置的时间，可直接进行M3的重传，这就是快重传。(面试时说这一句就够了，前面是帮助理解)

4. 快恢复：从上上图圈4可以看到，当发送收到三个重复的ACK，会进行快重传和快恢复。快恢复是指将ssthresh设置为发生快重传时的cwnd数量的一半，而cwnd不是设置为1而是设置为为门限值ssthresh，并开始拥塞避免阶段。

TCP的三次握手及四次挥手 * * *

> 必考题

在介绍三次握手和四次挥手之前，先介绍一下TCP头部的一些常用字段。

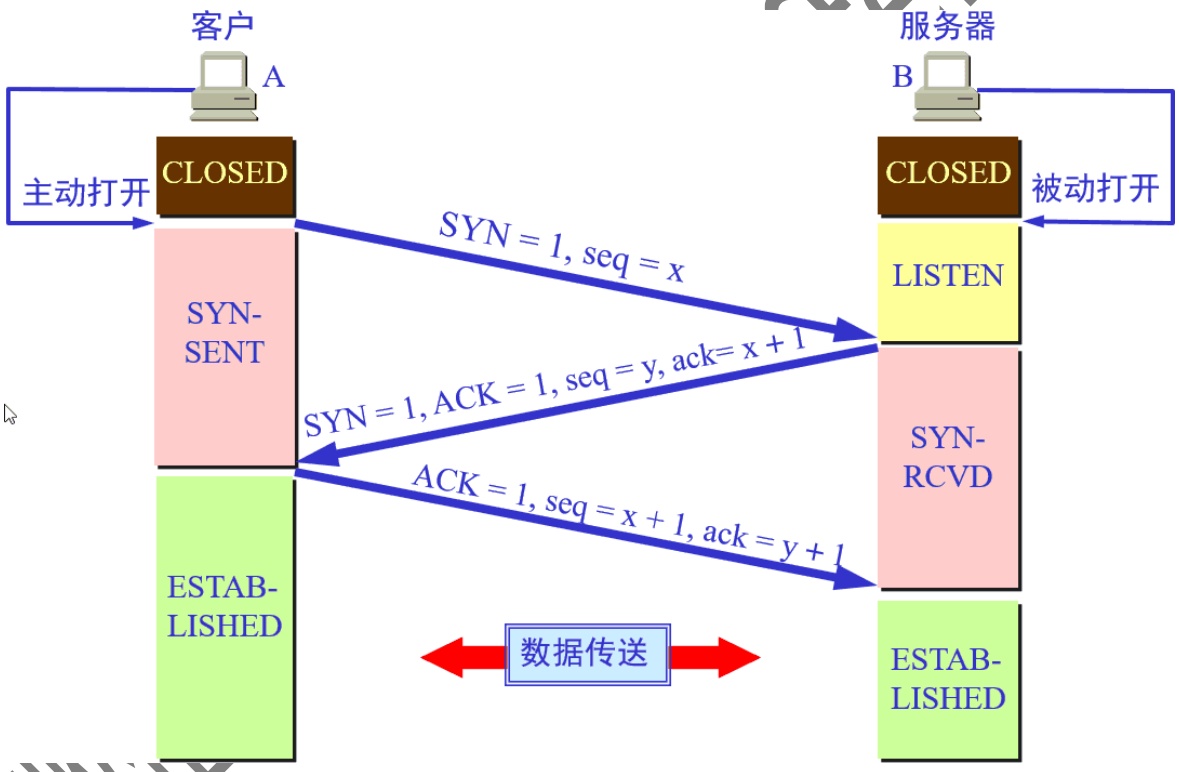
- 序号：seq，占32位，用来标识从发送端到接收端发送的字节流。
- 确认号：ack，占32位，只有ACK标志位为1时，确认序号字段才有效，ack=seq+1。
- 标志位：
  - SYN：发起一个新连接。
  - FIN：释放一个连接。
  - ACK：确认序号有效。

简述SYN攻击
SYN攻击即利用TCP协议缺陷，通过发送大量的半连接请求，占用半连接队列，耗费CPU和内存资源。
优化方式：
缩短SYN Timeout时间
记录IP，若连续受到某个IP的重复SYN报文，从这个IP地址来的包会被一概丢弃。

三次握手

> 三次握手的本质就是确定发送端和接收端具备收发信息的能力，在能流畅描述三次握手的流程及其中的字段含义作用的同时还需要记住每次握手时**接收端和发送端的状态**。这个比较容易忽略

先看一张很经典的图（图片来源于网络），发送端有CLOSED、SYN-SENT、ESTABLISHED三种状态，接收端有CLOSED、LISTEN、SYN-RCVD、ESTABLISHED四种状态。



假设发送端为客户端，接收端为服务端。开始时客户端和服务端的状态都是CLOSE。

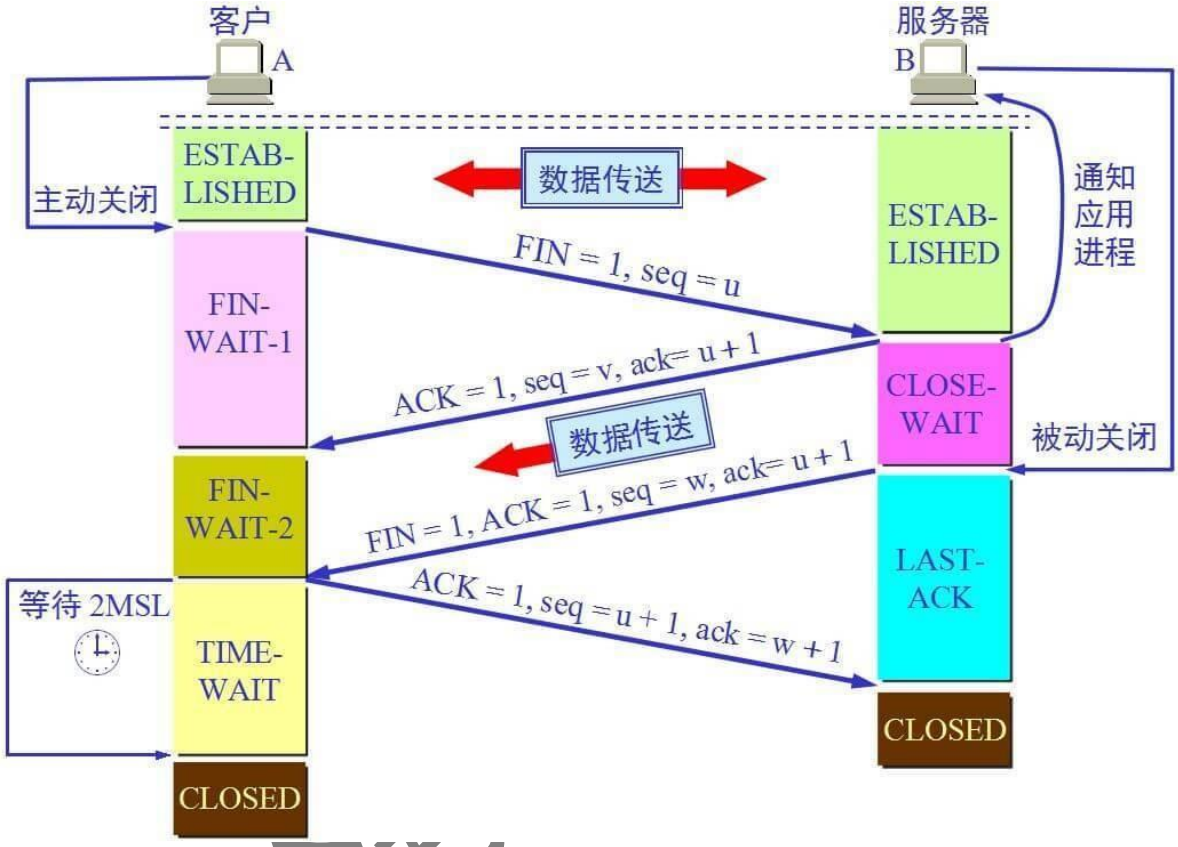
- 第一次握手：客户端向服务端发起建立连接请求，客户端会随机生成一个起始序列号x，客户端向服务端发送的字段中包含标志位SYN=1，序列号seq=100。第一次握手前客户端的状态为CLOSE，第一次握手后客户端的状态为SYN-SENT。此时服务端的状态为LISTEN
- 第二次握手：服务端在收到客户端发来的报文后，会随机生成一个服务端的起始序列号y，然后给客户端回复一段报文，其中包括标志位SYN=1，ACK=1，序列号seq=y，确认号ack=x+1。第二次握手前服务端的状态为LISTEN，第二次握手后服务端的状态为SYN-RCVD，此时客户端的状态为SYN-SENT。（其中SYN=1表示要和客户端建立一个连接，ACK=1表示确认序号有效）
- 第三次握手：客户端收到服务端发来的报文后，会再向服务端发送报文，其中包含标志位ACK=1，序列号seq=x+1，确认号ack=y+1。第三次握手前客户端的状态为SYN-SENT，第三次握手后客户端和服务端的状态都为ESTABLISHED。

在 TCP 连接的过程中，服务器的内核实际上为每个 Socket 维护了两个队列：
一个是还没完全建?连接的队列，称为 TCP 半连接队列，这个队列都是没有完成三次握手的连接，此时服务端处于 syn_rcvd 的状态；
另一个是建立连接的队列，称为 TCP 全连接队列，这个队列都是完成了三次握手的连接，此时服务端处于 established 状态；

> 需要注意的一点是，第一次握手，客户端向服务端发起建立连接报文，会占一个序列号。但是第三次握手，同样是客户端向服务端发送报文，这次却不占序列号，所以建立连接后，客户端向服务端发送的第一个数据的序列号为x+1。

四次挥手

和三次握手一样，先看一张非常经典的图（图片来源于网络），客户端在四次挥手过程中有ESTABLISHED、FIN-WAIT-1、FIN-WAIT-2、TIME-WAIT、CLOSED等五个状态，服务端有ESTABLISHED、CLOSE-WAIT、LAST-ACK、CLOSED等四种状态。最好记住每次挥手时服务端和客户端的状态。



假设客户端首先发起的断开连接请求

- 第一次挥手：客户端向服务端发送的数据完成后，向服务端发起释放连接报文，报文包含标志位FIN=1，序列号seq=u。此时客户端只能接收数据，不能向服务端发送数据。
- 第二次挥手：服务端收到客户端的释放连接报文后，向客户端发送确认报文，包含标志位ACK=1，序列号seq=v，确认号ack=u+1。此时客户端到服务端的连接已经释放掉，客户端不能像服务端发送数据，服务端也不能向客户端发送数据。但服务端到客户端的单向连接还能正常传输数据。
- 第三次挥手：服务端发送完数据后向客户端发出连接释放报文，报文包含标志位FIN=1，标志位ACK=1，序列号seq=w，确认号ack=u+1。
- 第四次挥手：客户端收到服务端发送的释放连接请求，向服务端发送确认报文，包含标志位ACK=1，序列号seq=u+1，确认号ack=w+1。

为什么TCP连接的时候是3次？两次是否可以？

不可以，主要从以下两方面考虑（假设客户端是首先发起连接请求）：

1. 假设建立TCP连接仅需要两次握手，那么如果第二次握手时，服务端返回给客户端的确认报文丢失了，客户端这边认为服务端没有和他建立连接，而服务端却以为已经和客户端建立了连接，并且可能向服务端已经开始向客户端发送数据，但客户端并不会接收这些数据，浪费了资源。如果是三次握手，不会出现双方连接还未完全建立成功就开始发送数据的情况。
2. 如果服务端接收到了一个早已失效的来自客户端的连接请求报文，会向客户端发送确认报文同意建立TCP连接。但因为客户端并不需要向服务端发送数据，所以此次TCP连接没有意义并且浪费了资

源。

为什么TCP连接的时候是3次，关闭的时候却是4次?

因为需要确保通信双方都能通知对方释放连接，假设客服端发送完数据向服务端发送释放连接请求，当客服端并不知道，服务端是否已经发送完数据，所以此次断开的是客服端到服务端的单向连接，服务端返回给客户端确认报文后，服务端还能继续单向给客户端发送数据。当服务端发送完数据后还需要向客户端发送释放连接请求，客户端返回确认报文，TCP连接彻底关闭。所以断开TCP连接需要客户端和服务端分别通知对方并分别收到确认报文，一共需要四次。

TIME_WAIT和CLOSE_WAIT的区别在哪?

默认客户端首先发起断开连接请求

- 从上图可以看出，CLOSE_WAIT是被动关闭形成的，当客户端发送FIN报文，服务端返回ACK报文后进入CLOSE_WAIT。
- TIME_WAIT是主动关闭形成的，当第四次挥手完成后，客户端进入TIME_WAIT状态。

为什么客户端发出第四次挥手的确认报文后要等2MSL的时间才能释放TCP连接?

MSL的意思是报文的最长寿命，可以从两方面考虑：

1. 客户端发送第四次挥手中的报文后，再经过2MSL，可使本次TCP连接中的所有报文全部消失，不会出现在下一个TCP连接中。
2. 考虑丢包问题，如果第四挥手发送的报文在传输过程中丢失了，那么服务端没收到确认ack报文就会重发第三次挥手的报文。如果客户端发送完第四次挥手的确认报文后直接关闭，而这次报文又恰好丢失，则会造成服务端无法正常关闭。

如果已经建立了连接，但是客户端突然出现故障了怎么办?

如果TCP连接已经建立，在通信过程中，客户端突然故障，那么服务端不会一直等下去，过一段时间就关闭连接了。具体原理是TCP有一个保活机制，主要用在服务器端，用于检测已建立TCP链接的客户端的状态，防止因客户端崩溃或者客户端网络不可达，而服务器端一直保持该TCP链接，占用服务器端的大量资源(因为Linux系统中可以创建的总TCP链接数是有限制的)。

保活机制原理：设置TCP保活机制的保活时间keepIdle，即在TCP链接超过该时间没有任何数据交互时，发送保活探测报文；设置保活探测报文的发送时间间隔keepInterval；设置保活探测报文的总发送次数keepCount。如果在keepCount次的保活探测报文均没有收到客户端的回应，则服务器端即关闭与客户端的TCP链接。

具体细节请看这篇博客TCP通信过程中异常情况整理。

HTTP 与 HTTPS 的区别 * * *

	HTTP	HTTPS
端口	80	443
安全性	无加密，安全性较差	有加密机制，安全性较高
资源消耗	较少	由于加密处理，资源消耗更多
是否需要证书	不需要	需要
协议	运行在TCP协议之上	运行在SSL协议之上，SSL运行在TCP协议之上

什么是对称加密与非对称加密 * *

- 对称加密

  对称加密指加密和解密使用同一密钥，优点是运算速度快，缺点是如何安全将密钥传输给另一方。常见的对称加密算法有DES、AES等等。

- 非对称加密

  非对称加密指的是加密和解密使用不同的密钥，一把公开的公钥，一把私有的私钥。公钥加密的信息只有私钥才能解密，私钥加密的信息只有公钥才能解密。优点解决了对称加密中存在的问题。缺点是运算速度较慢。常见的非对称加密算法有RSA、DSA、ECC等等。

  非对称加密的工作流程：A生成一对非堆成密钥，将公钥向所有人公开，B拿到A的公钥后使用A的公钥对信息加密后发送给A，经过加密的信息只有A手中的私钥能解密。这样B可以通过这种方式将自己的公钥加密后发送给A，两方建立起通信，可以通过对方的公钥加密要发送的信息，接收方用自己的私钥解密信息。

HTTPS的加密过程 * * *

X

上面已经介绍了对称加密和非对称加密的优缺点，HTTPS是将两者结合起来，使用的对称加密和非对称加密的混合加密算法。具体做法就是使用非对称加密来传输对称密钥来保证安全性，使用对称加密来保证通信的效率。

简化的工作流程：服务端生成一对非对称密钥，将公钥发给客户端。客户端生成对称密钥，用服务端发来的公钥进行加密，加密后发给服务端。服务端收到后用私钥进行解密，得到客户端发送的对称密钥。通信双方就可以通过对称密钥进行高效地通信了。

中间人攻击

但是仔细想想这其中存在一个很大地问题，就是客户端最开始如何判断收到的这个公钥就是来自服务端而不是其他人冒充的？

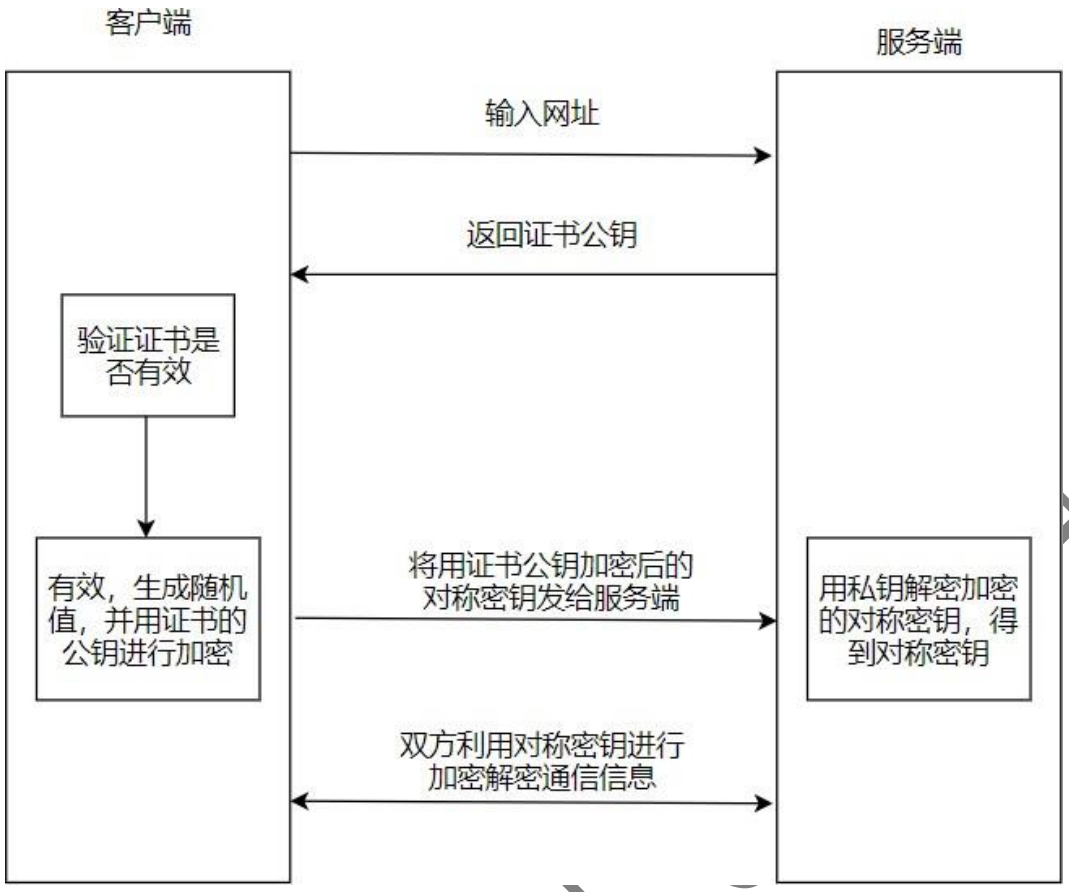
这就需要证书上场了，服务端会向一个权威机构申请一个证书来证明自己的身份，到时候将证书（证书中包含了公钥）发给客户端就可以了，客户端收到证书后既证明了服务端的身份又拿到了公钥就可以进行下一步操作了。

HTTPS的加密过程：

1. 客户端向服务端发起第一次握手请求，告诉服务端客户端所支持的SSL的指定版本、加密算法及密钥长度等信息。
2. 服务端将自己的公钥发给数字证书认证机构，数字证书认证机构利用自己的私钥对服务器的公钥进行数字签名，并给服务器颁发公钥证书。
3. 服务端将证书发给客服端。
4. 客服端利用数字认证机构的公钥，向数字证书认证机构验证公钥证书上的数字签名，确认服务器公开密钥的真实性。
5. 客服端使用服务端的公开密钥加密自己生成的对称密钥，发给服务端。
6. 服务端收到后利用私钥解密信息，获得客户端发来的对称密钥。
7. 通信双方可用对称密钥来加密解密信息。

上述流程存在的一个问题是客户端哪里来的数字认证机构的公钥，其实，在很多浏览器开发时，会内置常用数字证书认证机构的公钥。

流程图如下：



X

常用HTTP状态码 * * *

> 这也是一个面试经常问的题目,背下来就行了.

状态码	类别
1XX	信息性状态码
2XX	成功状态码
3XX	重定向状态码
4XX	客户端错误状态码
5XX	服务端错误状态码

常见的HTTP状态码

1XX

- 100 Continue：表示正常，客户端可以继续发送请求
- 101 Switching Protocols：切换协议，服务器根据客户端的请求切换协议。

2XX

- 200 OK：请求成功
- 201 Created：已创建，表示成功请求并创建了新的资源
- 202 Accepted：已接受，已接受请求，但未处理完成。
- 204 No Content：无内容，服务器成功处理，但未返回内容。
- 205 Reset Content：重置内容，服务器处理成功，客户端应重置文档视图。

- 206 Partial Content：表示客户端进行了范围请求，响应报文应包含Content-Range指定范围的实体内容

3XX

- 301 Moved Permanently：永久性重定向
- 302 Found：临时重定向
- 303 See Other：和301功能类似，但要求客户端采用get方法获取资源
- 304 Not Modified：所请求的资源未修改，服务器返回此状态码时，不会返回任何资源。
- 305 Use Proxy：所请求的资源必须通过代理访问
- 307 Temporary Redirect： 临时重定向，与302类似，要求使用get请求重定向。

4XX

- 400 Bad Request：客户端请求的语法错误，服务器无法理解。
- 401 Unauthorized：表示发送的请求需要有认证信息。
- 403 Forbidden：服务器理解用户的请求，但是拒绝执行该请求
- 404 Not Found：服务器无法根据客户端的请求找到资源。
- 405 Method Not Allowed：客户端请求中的方法被禁止
- 406 Not Acceptable：服务器无法根据客户端请求的内容特性完成请求
- 408 Request Time-out：服务器等待客户端发送的请求时间过长，超时

5XX

- 500 Internal Server Error：服务器内部错误，无法完成请求
- 501 Not Implemented：服务器不支持请求的功能，无法完成请求

常见的HTTP方法 * * *

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

为了方便记忆，可以将PUT、DELETE、POST、GET理解为客户端对服务端的增删改查。

- PUT：上传文件，向服务器添加数据，可以看作增
- DELETE：删除文件
- POST：传输数据，向服务器提交数据，对服务器数据进行更新。
- GET：获取资源，查询服务器资源

GET和POST区别 * * *

安全是指请求方法不会破坏服务器上的资源。GET是「只读」操作，无论操作多少次，服务器上的数据都是安全的；POST是新增或提交数据,会修改服务器上的资源，所以是不安全的。

- 作用

  GET用于获取资源，POST用于传输实体主体

- 参数位置

  GET的参数放在URL中，POST的参数存储在实体主体中，并且GET方法提交的请求的URL中的数据做多是2048字节，POST请求没有大小限制。

- 安全性

  GET方法因为参数放在URL中，安全性相对于POST较差一些

- 幂等性

  GET方法是具有幂等性的，而POST方法不具有幂等性。这里幂等性指客户端连续发出多次请求，收到的结果都是一样的.

X

HTTP 1.0、HTTP 1.1及HTTP 2.0的主要区别是什么 * *

HTTP 1.0和HTTP 1.1的区别

- 长连接

  HTTP 1.1支持长连接和请求的流水线操作。长连接是指不在需要每次请求都重新建立一次连接，HTTP 1.0默认使用短连接，每次请求都要重新建立一次TCP连接，资源消耗较大。请求的流水线操作是指客户端在收到HTTP的响应报文之前可以先发送新的请求报文，不支持请求的流水线操作需要等到收到HTTP的响应报文后才能继续发送新的请求报文。

- 缓存处理

  在HTTP 1.0中主要使用header中的If-Modified-Since,Expires作为缓存判断的标准，HTTP 1.1引入了Entity tag，If-Unmodified-Since，If-Match等更多可供选择的缓存头来控制缓存策略。

- 错误状态码

  在HTTP 1.1新增了24个错误状态响应码

缓存策略：https://blog.51cto.com/u_14234228/2490237
1.x与2的区别https://www.cnblogs.com/lalalatianlalu/p/10944097.html

- HOST域

  在HTTP 1.0 中认为每台服务器都会绑定唯一的IP地址，所以，请求中的URL并没有传递主机名。但后来一台服务器上可能存在多个虚拟机，它们共享一个IP地址，所以HTTP 1.1中请求消息和响应消息都应该支持Host域。

- 带宽优化及网络连接的使用

  在HTTP 1.0中会存在浪费带宽的现象，主要是因为不支持断点续传功能，客户端只是需要某个对象的一部分，服务端却将整个对象都传了过来。在HTTP 1.1中请求头引入了range头域，它支持只请求资源的某个部分，返回的状态码为206。

HTTP 2.0的新特性

- 新的二进制格式：HTTP 1.x的解析是基于文本，HTTP 2.0的解析采用二进制，实现方便，健壮性更好。
- 多路复用：每一个request对应一个id，一个连接上可以有多个request，每个连接的request可以随机混在一起，这样接收方可以根据request的id将request归属到各自不同的服务端请求里。
- header压缩：在HTTP 1.x中，header携带大量信息，并且每次都需要重新发送，HTTP 2.0采用编码的方式减小了header的大小，同时通信双方各自缓存一份header fields表，避免了header的重复传输。
- 服务端推送：客户端在请求一个资源时，会把相关资源一起发给客户端，这样客户端就不需要再次发起请求。

HTTP1.0:不支持断点续传，如果请求被中断，则必须重新再次发送，对于大文件的下载和传输不友好，尤其是在网络不好的情况下
HTTP1.1:支持断点续传，利用HTTP消息头使用分块传输编码，将实体主体分块传输。垮掉了还能捡起来，很棒

Session、Cookie和Token的主要区别 * * *

HTTP协议是无状态的，即服务器无法判断用户身份。Session和Cookie可以用来进行身份辨认。

- Cookie

  Cookie是保存在客户端一个小数据块，其中包含了用户信息。当客户端向服务端发起请求，服务端会像客户端浏览器发送一个Cookie，客户端会把Cookie存起来，当下次客户端再次请求服务端时，会携带上这个Cookie，服务端会通过这个Cookie来确定身份。

- Session

  Session是通过Cookie实现的，和Cookie不同的是，Session是存在服务端的。当客户端浏览器第一次访问服务器时，服务器会为浏览器创建一个sessionid，将sessionid放到Cookie中，存在客户端浏览器。比如浏览器访问的是购物网站，将一本《图解HTTP》放到了购物车，当浏览器再次访问服务器时，服务器会取出Cookie中的sessionid，并根据sessionid获取会话中的存储的信息，确认浏览器的身份是上次将《图解HTTP》放入到购物车那个用户。

- Token

  客户端在浏览器第一次访问服务端时，服务端生成的一串字符串作为Token发给客户端浏览器，下次浏览器在访问服务端时携带token即可无需验证用户名和密码，省下来大量的资源开销。看到这里很多人感觉这不是和sessionid作用一样吗？其实是不一样的，但是本文章主要针对面试，知识点很多，篇幅有限，几句话也解释不清楚，大家可以看看这篇文章，我觉得说的非常清楚了。
  [cookie、session与token的真正区别]
  下面为了方便记忆，做了一个表格进行对比。

	存放位置	占用空间	安全性	应用场景
Cookie	客户端浏览器	小	较低	一般存放配置信息
Session	服务端	多	较高	存放较为重要的信息

如果客户端禁止 cookie 能实现 session 还能用吗? *

可以，Session的作用是在服务端来保持状态，通过sessionid来进行确认身份，但sessionid一般是通过Cookie来进行传递的。如果Cookie被禁用了，可以通过在URL中传递sessionid。

在浏览器中输入url地址到显示主页的过程 * * *

> 面试超高频的一道题，一般能说清楚流程就可以。

1. 对输入到浏览器的url进行DNS解析，将域名转换为IP地址。
2. 和目的服务器建立TCP连接
3. 向目的服务器发送HTTP请求
4. 服务器处理请求并返回HTTP报文
5. 浏览器解析并渲染页面

Servlet是线程安全的吗 *

Servlet不是线程安全的，多线程的读写会导致数据不同步的问题。