**好文分享：**

1. 面渣逆袭：计算机网络六十二问，三万字图文详解！速收藏！

<https://mp.weixin.qq.com/s/yAlErlC09GnjaVvwUo3Acg>

1. **七层模型和五层模型**

1、物理层（建立、维护、断开物理连接）：网线、电缆、集线器 。这一层数据的单位称为比特（bit）

2、数据链路层（建立逻辑连接、进行硬件地址寻址、差错校验等功能）： arp，数据的单位称为帧（frame）

3、网络层（进行逻辑地址寻址，实现不同网络之间的路径选择）：ipv4、icmp，单位为数据包（packet）

4、传输层（定义传输数据的协议端口号，以及流控和差错校验）：tcp、udp，单位为数据段（segment）

5、会话层（建立、管理、终止会话）：rpc

6、表示层（数据的表示、安全、压缩）：jpeg、ascii，

7、应用层（网络服务与最终用户的一个接口，规定应用程序的数据格式）：http、ftp、smtp

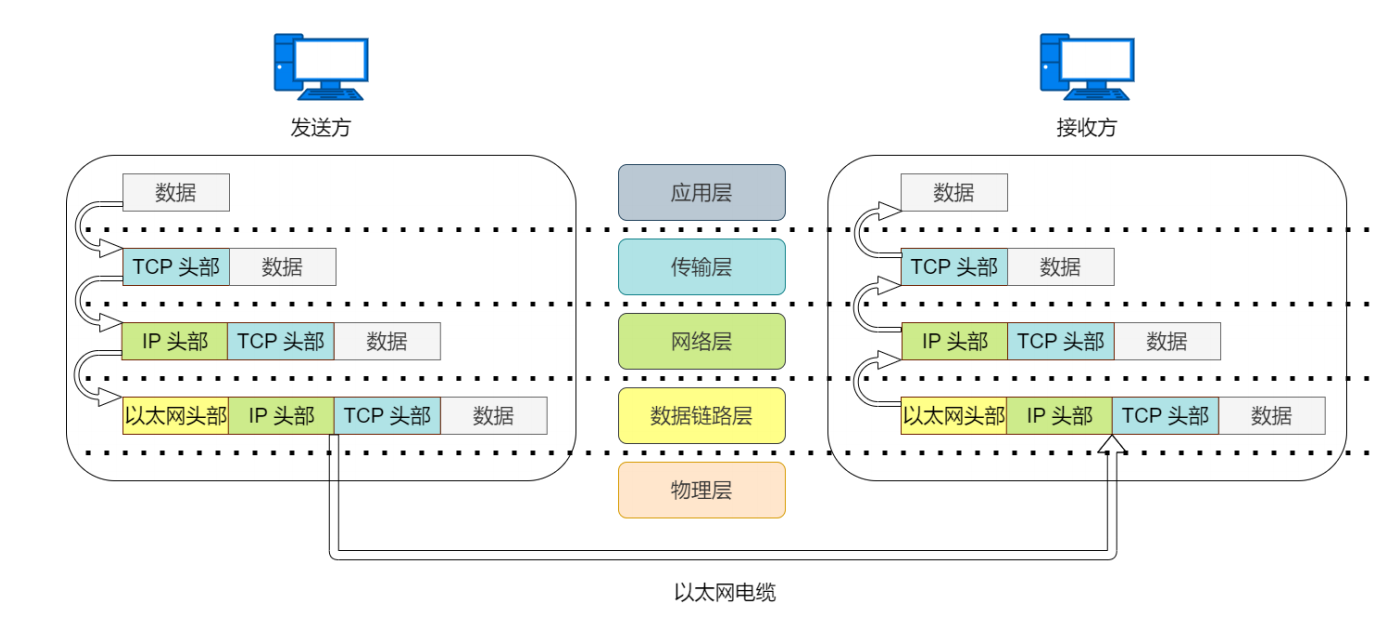
|  |  |  |  |
| --- | --- | --- | --- |
| OSI七层网络模型 | 五层体系结构 | TCP/IP四层概念模型 | 对应网络协议 |
| 应用层 (message) | 应用层 | 应用层 | HTTP、FTP、SMTP |
| 表示层 | Telnet、SNMP、 |
| 会话层 | RPC |
| 传输层 (segment) | 传输层 | 传输层 | TCP、UDP |
| 网络层 (packet) | 网络层 | 网络层 | IP、ICMP |
| 数据链路层 (frame） | 数据链路层 | 数据链路层 | FDDI,Ethernet,Arpanet,PPP,PDN |
| 物理层 (bit) | 物理层 | IEEE802.1A |

交换机与路由器的区别：

①工作所处的OSI层次不一样，交换机工作在OSI第二层数据链路层，路由器工作在OSI第三层网络层

②寻址方式不同：交换机根据MAC地址寻址，路由器根据IP地址寻址

③转发速不同：交换机的转发速度快，路由器转发速度相对较慢。



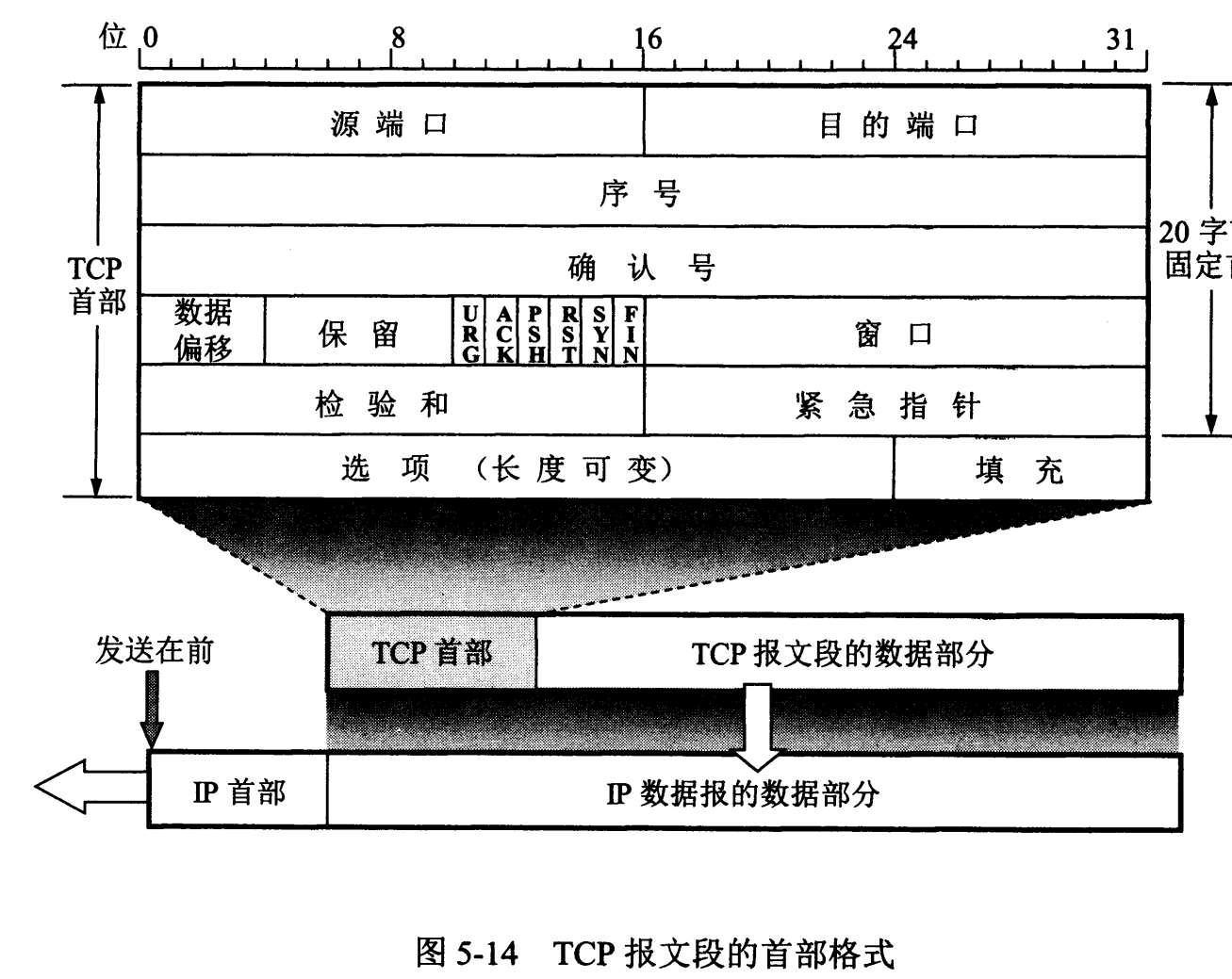
**二、TCP（传输控制协议）**

**TCP:** 提供一种面向连接的、可靠的传输层协议，即数据无误、数据无丢失、数据无失序、数据无重复到达的通信。在传送数据前必须建立连接，TCP连接是通过三次握手建立的。

**UDP:**是一种面向无连接的传输层协议，不会对自己提供的连接实施控制。适用于实时应用，例如：IP电话、视频会议、直播等。

1. **TCP报文结构图**

备注：1个汉字=2byte=2字节=16bit tcp首部长度最少20个字节。



**IP首部：**版本、首部长度、服务类型、全长、标识符生存时间、标志和分片偏移、32位源ip、32位目的ip

**序号seq ：**用于对字节流进行编号，例如序号为 301，表示第一个字节的编号为 301，如果携带的数据长度为 100 字节，那么下一个报文段的序号应为 401。

**确认号ack：**期望收到的下一个报文段的序号。例如 B 正确收到 A 发送来的一个报文段，序号为 501，携带的数据长度为 200 字节，因此 B 期望下一个报文段的序号为 701，B 发送给 A 的确认报文段中确认号就为 701。

**数据偏移** ：指的是数据部分距离报文段起始处的偏移量，实际上指的是首部的长度。

**URG:** 紧急标志位，为1表示有紧急数据，应尽快传送。

**确认 ACK** ：当 ACK=1 时确认号字段有效，否则无效。TCP 规定，在连接建立后所有传送的报文段都必须把 ACK 置 1。

**PSH：**催促标志位，为1表示会立即创建一个报文并发送，接收端应立即返回结果。

**RST：**复位标志位，为1表示直接丢弃缓存区的包发送RST包，接收端收到RST包无须发ack包确认。

**同步 SYN** ：在连接建立时用来同步序号。当 SYN=1，ACK=0 时表示这是一个连接请求报文段。若对方同意建立连接，则响应报文中 SYN=1，ACK=1。

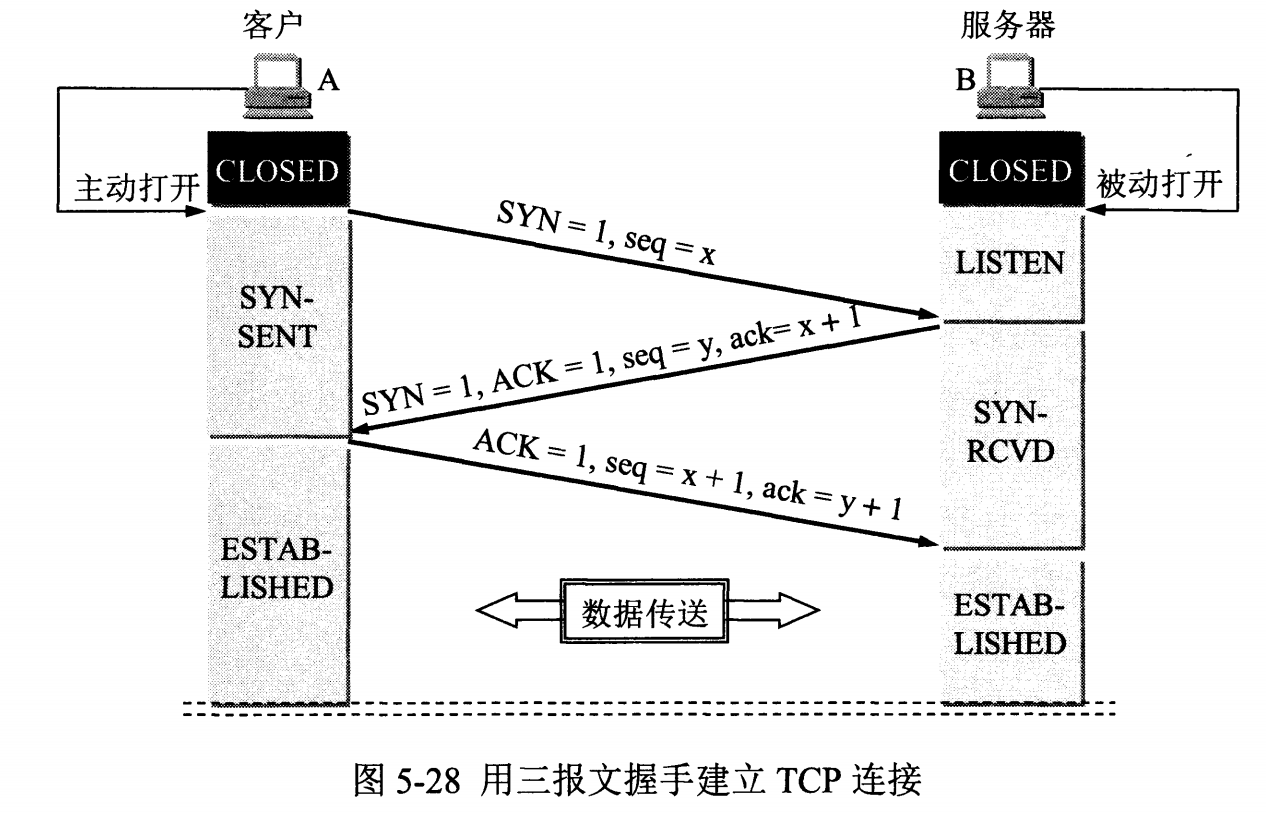
**终止 FIN** ：用来释放一个连接，当 FIN=1 时，表示此报文段的发送方的数据已发送完毕，并要求释放连接。

**窗口** ：窗口值作为接收方让发送方设置其发送窗口的依据。之所以要有这个限制，是因为接收方的数据缓存空间是有限的。

**16位校验和：**由发送端填充，接收端对 TCP 报文段执行 CRC 算法以检验 TCP 报文段在传输过程中是否损坏。注意，这个校验不仅包括 TCP 头部，也包括数据部分。这也是 TCP 可靠传输的一个重要保障。

**16 位紧急指针：**它和序号字段的值相加表示最后一个紧急数据的下一字节的序号。因此确切地说，这个字段是紧急指针相对当前序号的偏移，不妨称之为紧急偏移。TCP 的紧急指针是发送端向接收端发送紧急数据的方法。

**2.TCP三次握手**



**3、TCP建立过程状态机转变：**

（1）一开始建立连接之前，服务端和客户端的状态都为CLOSED。

（2）服务端创建socket后开始监听，变为LISTEN状态，等待客户的连接请求。

（3）客户端向服务端发送请求:，第一次握手(SYN=1，ACK=0，seq=x) 的连接请求报文，客户端的状态变为SYN\_SENT。

（4）服务端收到客户端的报文，如果同意建立连接，则向客户端发送第二次握手（SYN=1、ACK=1、seq=y、ack=x+1）的报文，此时服务端的状态变为SYN\_RCVD。

（5）客户端收到服务端的连接确认报文后，还要向服务端发出确认报文，第三次握手(ACK=1，seq=x+1，ack=y+1)，客户端状态变为ESTABLISHED，服务端收到客户端的ACK后也变为ESTABLISHED。

（6）B收到A的确认后，连接建立。至此，TCP3次握手完成，建立连接！

1. **TCP 握手为什么是三次，为什么不能是两次？不能是四次？**

三次握手完成两个重要的功能，既要双方做好发送数据的准备工作（双方都知道彼此已经准备好），也要允许双方就初始序列号进行协商，这个序列号在握手过程中被发送确认。

**为什么不能是两次？**

为了防止服务器端开启一些无用的连接增加服务器开销。

防止已失效的连接请求报文段突然又传送到了服务端，因而产生错误。

如果改成两次握手，可能会发生死锁现象。客户端给服务端发送了一个连接请求，服务端收到了这个分组，并发送了确认应答分组。按照两次握手的协议，服务端认为已经成功地建立了，可以开始发送数据分组。可是客户端的应答分组在传输中被丢失的情况下，将不知道服务端是否已经准备好，不知道服务端建立什么样的序列号，客户端甚至怀疑服务端手否收到自己的连接请求分组。在这种情况下，客户端认为连接还未建立成功，将忽略服务端发来的任何数据分组，只等待连接确认应答分组。而服务端在发出的分组超时后，重复发送同样的分组,这样就形成了死锁。

**所以我们需要“第三次握手”来确认这个过程：**

通过第三次握手的数据告诉服务端，客户端有没有收到服务器“第二次握手”时传过去的数据，以及这个连接的序号是不是有效的。若发送的这个数据是“收到且没有问题”的信息，接收后服务器就正常建立 TCP 连接，否则建立 TCP 连接失败，服务器关闭连接端口。由此减少服务器开销和接收到失效请求发生的错误。

**为什么不是四次？**

简单说，就是三次挥手已经足够创建可靠的连接，没有必要再多一次握手导致花费更多的时间建立连接。

**5、三次握手中每一次没收到报文会发生什么情况？**

**第一次握手服务端未收到SYN报文**

服务端不会进行任何的动作，而客户端等待一段时间后会重新发送SYN报文，如果仍然没有回应，会重复这个过程，直到发送次数超过最大重传次数限制，就会返回连接建立失败。

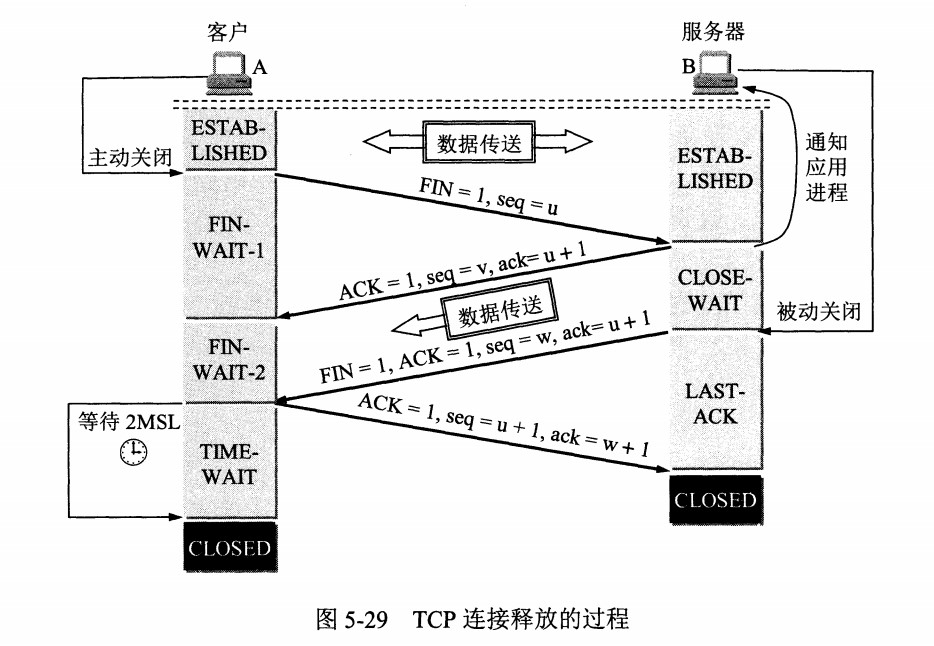
**第二次握手客户端未收到服务端响应的ACK报文**

客户端会继续重传，直到次数限制；而服务端此时会阻塞在accept()处，等待客户端发送ACK报文

**第三次握手服务端为收到客户端发送过来的ACK报文**

服务端同样会采用类似客户端的超时重传机制，如果重试次数超过限制，则accept()调用返回-1，服务端建立连接失败；而此时客户端认为自己已经建立连接成功，因此开始向服务端发送数据，但是服务端的accept()系统调用已经返回，此时不在监听状态，因此服务端接收到客户端发送来的数据时会发送RST报文给客户端，消除客户端单方面建立连接的状态。

**6、TCP断开连接四次挥手过程：**



（1）客户端发送TCP连接请求的报文，其中报文中包含了seq序列号，是由发送端随机生成的，并且还将报文中的FIN字段置为1，表示需要断开TCP连接。

（2）服务端会回复服务端发送的TCP断开请求报文，其中包含seq序列号，是由回复端随机生成的，而且会产生ACK字段，ACK字段数值，是在服务端发过来的seq序列号基础上加1进行回复的，一边服务端收到信息时，知晓自己的TCP断开请求已经得到了验证。

（3）服务端在回复完服务端的TCP断开请求后，不会马上就进行TCP连接的断开，客户端会先确保断开前，所有传输到服务端的数据是否已经传输完毕，一但确认传输数据完毕就会将回复报文的FIN字段置为1，并产生随机seq序列号。

（4）客户端收到客户端的TCP断开请求后，会回复客户端的断开请求，包含随机生成的seq字段和ACK字段，ACK字段会在客户端的TCP断开请求的seq基础上加1，从而完成客户端请求的验证回复。

（5）至此TCP断开的四次挥手过程完毕。

**7、四次挥手的原因：**

客户端发送了 FIN 连接释放报文之后，服务器收到了这个报文，就进入了 CLOSE-WAIT 状态。这个状态是为了让服务器端发送还未传送完毕的数据，传送完毕之后，服务器会发送 FIN 连接释放报文。

**8.为什么连接的时候是三次握手，关闭的时候确是四次握手？**

因为当服务端收到客户端的SYN连接请求报文后，可以直接发送SYN+ACK报文。其中ACK报文是用来应答的，SYN报文是用来同步的。

关闭连接时，当服务端收到FIN报文时，很可能并不会立即关闭SOCKET，所以只能先回复一个ACK报文，告诉客户端，你发的FIN报文我收到了。只有等到我服务端所有的报文都发送完了，我才能发送FIN报文，因此不能一起发送，故需要四步握手。

**9.为什么TIME\_WAIT状态需要经过2MSL（最大报文段生存时间）才能返回到CLOSE状态？**

虽然按道理，四个报文都发送完毕，我们可以直接进入CLOSE状态了，但是我们必须假想网络是不可靠的，有可能最后一个ACK丢失。所以客户端TIME\_WAIT状态就是用来重发可能丢失的ACK报文。在客户端发送出最后的ACK回复，但该ACK可能丢失。服务端如果没有收到ACK，将不断重复发送FIN片段。所以客户端不能立即关闭，他必须确认服务端接收到了ACK。客户端会在发送出ACK之后进入到TIME\_WAIT状态。客户端会设置一个计时器，等待2MSL是两倍的最大报文段生存时间。如果直到2MSL，客户端都没有再次收到FIN，那么客户端推断ACK已经被成功接收，则结束TCP连接。

**10.如果已经建立了连接，但客户端突然出现故障了怎么办？**

TCP还设有一个保活计时器，客户端如果出现故障，服务器不能一直等下去，白白浪费资源。服务器没收到一次客户端的请求后都会重新复位这个计时器，若两小时还没有收到客户端的任何数据，服务端就会发送一个探测报文段，以后每隔75分钟发送一次，若一连发送10个探测报文任然没反应，服务端就认为客户端出了故障，接着就关闭连接。

1. **TCP，UDP区别，为什么可靠和不可靠？**

**TCP的优点：** 可靠（有序、无差错、不丢失、不重复），稳定 TCP的可靠体现在TCP在传递数据之前，会有三次握手来建立连接，而且在数据传递时，有确认、窗口、重传、拥塞控制机制，在数据传完后，还会断开连接来节约系统资源。

**TCP的缺点：** 慢，效率低，占用系统资源高，易被攻击 TCP在传递数据之前，要先建连接，这会消耗时间，而且在数据传递时，确认机制、重传机制、拥塞控制机制等都会消耗大量的时间，而且要在每台设备上维护所有的传输连接，事实上，每个连接都会占用系统的CPU、内存等硬件资源。 而且因为TCP有确认机制、三次握手机制，这些也导致TCP容易被人利用，实现DOS、DDOS、CC等攻击。

**UDP的优点：** 快，比TCP稍安全。 UDP没有TCP的握手、确认、窗口、重传、拥塞控制等机制，UDP是一个无状态的传输协议，所以它在传递数据时非常快。没有TCP的这些机制，UDP较TCP被攻击者利用的漏洞就要少一些。但UDP也是无法避免攻击的，比如：UDP Flood攻击……

**UDP的缺点：** 不可靠，不稳定。因为UDP没有TCP那些可靠的机制，在数据传递时，如果网络质量不好，就会很容易丢包。

1. **在不抓包的情况下，如果判断视频是使用了TCP还是UDP?**

**方法一：**弱网情况下，画面、声音卡顿是否出现前边的内容，应该是udp。

**方法二：**重新连接网络或者切换网络，如果内容未出现前边的内容，应该是udp。

现在常用的做法是使用http3.0，基于QUIC的UDP协议。QUIC 有⾃⼰的⼀套机制可以保证传输可靠性的。当某个流发⽣丢包时，只会阻塞这个流，其他流不会受到影响。

1. **说说 WebSocket 与 Socket 的区别？**

**Socket：** 其实就是等于 IP 地址 + 端口 + 协议。Socket 是一套标准，它完成了对 TCP/IP 的高度封装，屏蔽网络细节，以方便开发者更好地进行网络编程。

**WebSocket ：**是应用层通信协议。是一个持久化的协议，它是伴随 H5 而出的协议，用来解决 http 不支持持久化连接的问题。

**三、http**

**一、返回码**

|  |  |
| --- | --- |
| 2XX（成功） | 200:成功 |
| 3XX（重定向） | 300:多种选择 301:永久重定向 302:临时重定向 303:强制将post转成get |
| 4XX（客户端） | 400:错误请求 401:当前请求需要认证 403:禁止访问 404:访问的资源不存在 |
| 5XX（服务端） | 500:服务器内部错误 501:功能尚未实施 502:网关错误 503:服务不可用 504:网关超时 |

502：网关错误， 通俗点说就是和web服务器通信失败。错误的原因：

1. 网络不通，不能访问web服务器。 有可能断网、开启了防火墙等，可以通过ping命令来定位
2. web服务器没有启动， 可以通过查看日志来定位这个问题，或者查看端口是否启动
3. web服务器请求太多，响应不了请求，这个表现是有时好有时不好，可以通过查看web服务器日志来定位
4. **浏览器http全流程，包括域名解析、定位主机等**

**简单版：**域名解析 --> 发起TCP的3次握手 --> 建立TCP连接后发起http请求 --> 服务器响应http请求，浏览器得到html代码 --> 浏览器解析html代码，并请求html代码中的资源（如js、css、图片等） --> 浏览器对页面进行渲染呈现给用户。

**详细版：**

第一步：在浏览器输入地址，回车。

第二步：浏览器检测url的合法性，查找域名的 IP 地址：

浏览器分解url、生成http请求消息、解析这个域名，获取ip地址（浏览器会首先查看本地硬盘的 hosts

文件查找 ip 地址，没有则发出一个 DNS请求到本地 DNS 服务器，如果还是没有，则会一层一层往上

请求到根dns服务器）

第三步：浏览器向 web 服务器发送一个 HTTP 请求：

拿到域名对应的 IP 地址之后，浏览器会以一个随机端口（1024<端口<65535）向服务器的 WEB 程序

（常用的有 httpd,nginx 等）80 端口发起 TCP 的连接请求（

tcp模块1.创建套接字(socket) 、2.连接服务器、3.发送数据；

IP模块发送请求：生成IP头部、mac头部；

MAC模块发送请求：生成生成报头、校验序列、电信号；

PHY模块发送请求：网卡中的PHY模块会将通用电信号转换成网络传输所需的格式，通过网线发送

出去。经过网络转发后，最终到达服务器。）

客户端向服务器发起 http 请求的时候，会有一些请求信息，请求信息包含三个部分：

| 请求方法 URI 协议/版本

| 请求头(Request Header)

| 请求正文：

第四步：这个连接请求到达服务器端后，进入到网卡、进入到内核的 TCP/IP 协议栈（用于识别该连接请求，解

封包），最终到达 WEB 程序，建立了 TCP/IP的连接。

第五步：服务器根据内部逻辑处理请求，返回一个 HTTP 响应。

第六步：本地接收服务器响应，浏览器显示 HTML。

vue 实例有一个完整的生命周期，也就是从开始创建、初始化数据、编译模版、挂载 Dom -> 渲染、更

新 -> 渲染、卸载等一系列过程，我们称这是 Vue 的生命周期。

第七步：断开连接：TCP 四次挥手，连接结束。

1. **http优化**

资源内联：既然每个资源的首次访问都会存在握手等rtt损耗，那么越少数量的资源请求就越好，例如在一个html中src访问css，不如直接将其这个css集成到html中

图片懒加载：用到的时候再加载

服务器渲染：让服务端先将页面渲染好，再发送给客户端，也可以减少rtt的次数

1. **URL组成格式**



图中中括号是可选项

protocol 协议，常用的协议是http

hostname 主机地址，可以是域名，也可以是IP地址

port 端口 http协议默认端口是：80端口，如果不写默认就是:80端口

path 路径 网络资源在服务器中的指定路径

parameter 参数 如果要向服务器传入参数，在这部分输入

query 查询字符串 如果需要从服务器那里查询内容，在这里编辑

fragment 片段 网页中可能会分为不同的片段，如果想访问网页后直接到达指定位置，可以在这部分设置

1. **HTTP与HTTPS的基本概念和他们的区别**

1、HTTP的中文叫超文本传输协议，它负责完成客户端到服务端的一系列操作，是专门用来传输注入HTML的超媒体文档等web内容的协议，它是基于传输层的TCP协议的应用层协议。HTTP协议以明文方式发送内容，不提供任何方式的数据加密，如果攻击者截取了Web浏览器和网站服务器之间的传输报文，就可以直接读懂其中的信息。

2、HTTPS可以理解为HTTP+SSL/TLS （数字证书）的组合。HTTP协议以明文方式发送内容，不提供任何方式的数据加密，如果攻击者截取了Web浏览器和网站服务器之间的传输报文，就可以直接读懂其中的信息。

3、HTTP和HTTPS的区别

3.1 HTTP是超⽂本传输协议，信息是明⽂传输，存在安全⻛险的问题。HTTPS则解决 HTTP不安全的缺陷，在TCP和 HTTP⽹络层之间加⼊了 SSL/TLS 安全协议，使得报⽂能够加密传输。

3.2 HTTP连接建⽴相对简单， TCP三次握⼿之后便可进⾏ HTTP 的报⽂传输。⽽ HTTPS 在 TCP 三次握⼿之后，还需进⾏ SSL/TLS 的握⼿过程，才可进⼊加密报⽂传输。

3.3 HTTP的标准端口是80，而HTTPS的标准端口是443；

3.4 HTTP无需认证证书，HTTPS需要 CA（证书权威机构）申请数字证书来保证服务器的身份是可信的，相对安全；

3.5 HTTP的URL以http://开头，而HTTPS 的URL以https://开头；

4、HTTPS的工作原理

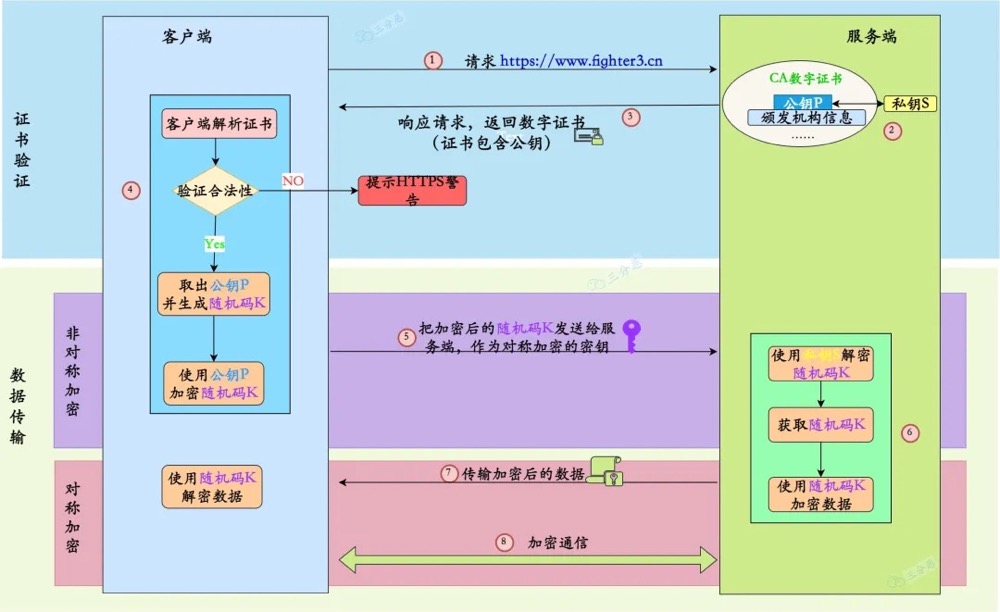
4.1 首先服务端给客户传输证书，证书包含了很多信息(比如说证书的颁发机构，证书的过期时间序列号、所有者信息、所有者公钥等)

4.2 客户端进行证书的解析，若发现没问题，就生成一个随机值（私钥），然后用证书（也就是服务端发送的公钥）进行加密，并发给服务端

4.3 服务端使用私钥将这个信息进行解密，得到客户端的私钥，然后客户端和服务端就可以通过这个私钥进行通信了

4.4 服务端将信息进行对称加密，私钥正好直有客户端和服务端知道，所以信息就比较安全了

1. **系统从HTTP切换到HTTPS，需要从哪些方面进行测试？(待补充）**
2. 功能回归
3. 性能回归
4. 不同浏览器是否有证书问题
5. nginx配置检查
6. **HTTPS的工作流程**
7. 客户端发起 HTTPS 请求，连接到服务端的 443 端口。
8. 服务端有一套数字证书（证书内容有公钥、证书颁发机构、失效日期等）。
9. 服务端将自己的数字证书发送给客户端（公钥在证书里面，私钥由服务器持有）。
10. 客户端收到数字证书之后，会验证证书的合法性。如果证书验证通过，就会生成一个随机的对称密钥，用证书的公钥加密。
11. 客户端将公钥加密后的密钥发送到服务器。
12. 服务器接收到客户端发来的密文密钥之后，用自己之前保留的私钥对其进行非对称解密，解密之后就得到客户端的密钥，然后用客户端密钥对返回数据进行对称加密，酱紫传输的数据都是密文啦。
13. 服务器将加密后的密文返回到客户端。
14. 客户端收到后，用自己的密钥对其进行对称解密，得到服务器返回的数据。



1. **GET和POST的区别：**

1、GET 请求将信息放在 URL，POST 将请求信息放在请求体中。这一点使得 GET 请求携带的数据量有限，因为 URL 本身是有长度限制的，而 POST 请求的数据存放在报文体中，因此对大小没有限制。而且从形式上看，GET 请求把数据放 URL 上不太安全，而 POST 请求把数据放在请求体里想比较而言安全一些。

2、GET产生一个TCP数据包，对于GET方式的请求，浏览器会把http header和data一并发送出去，服务器响应200（返回数据）；

POST产生两个TCP数据包。浏览器先发送header，服务器响应100 continue，浏览器再发送data，服务器响应200 ok（返回数据）。

3、GET 请求能够被缓存，GET 请求能够保存在浏览器的浏览记录里，GET 请求的 URL 能够保存为浏览器书签。这些都是 POST 请求所不具备的。缓存是 GET 请求被广泛应用的根本，他能够被缓存也是因为它的幂等性和安全性，除了返回结果没有其他多余的动作，因此绝大部分的 GET 请求都被 CDN 缓存起来了，大大减少了 Web 服务器的负担。

1. **说下 HTTP/1.0，1.1，2.0 ，3.0的区别？**

**关键需要记住 HTTP/1.0 默认是短连接，可以强制开启，HTTP/1.1 默认长连接，HTTP/2.0 采用多路复用。HTTP/3主要有两大变化，传输层基于UDP、使用QUIC保证UDP可靠性。**

**HTTP/1.0**

默认使用短连接，每次请求都需要建立一个 TCP 连接。它可以设置Connection: keep-alive 这个字段，强制开启长连接。

**HTTP/1.1**

引入了持久连接，即 TCP 连接默认不关闭，可以被多个请求复用。

分块传输编码，即服务端每产生一块数据，就发送一块，用“流模式” 取代” 缓存模式”。

管道机制，即在同一个 TCP 连接里面，客户端可以同时发送多个请求。

**HTTP/2.0**

二进制协议，1.1 版本的头信息是文本（ASCII 编码），数据体可以是文本或者二进制；2.0 中，头信息和数据体都是二进制。

完全多路复用，在一个连接里，客户端和浏览器都可以同时发送多个请求或回应，而且不用按照顺序一一对应。

报头压缩，HTTP 协议不带有状态，每次请求都必须附上所有信息。Http/2.0 引入了头信息压缩机制，使用 gzip 或 compress 压缩后再发送。

服务端推送，允许服务器未经请求，主动向客户端发送资源。

**HTTP/3**

**主要有两大变化，传输层基于UDP、使用QUIC保证UDP可靠性。**

HTTP/2存在的一些问题，比如重传等等，都是由于TCP本身的特性导致的，所以HTTP/3在QUIC的基础上进行发展而来，QUIC（Quick UDP Connections）直译为快速UDP网络连接，底层使用UDP进行数据传输。

HTTP/3主要有这些特点：

使用UDP作为传输层进行通信

在UDP的基础上QUIC协议保证了HTTP/3的安全性，在传输的过程中就完成了TLS加密握手。

HTTPS 要建⽴⼀个连接，要花费 6 次交互，先是建⽴三次握⼿，然后是 TLS/1.3 的三次握⼿。QUIC 直接把以往的 TCP 和 TLS/1.3 的 6 次交互合并成了 3 次，减少了交互次数。

QUIC 有⾃⼰的⼀套机制可以保证传输的可靠性的。当某个流发⽣丢包时，只会阻塞这个流，其他流不会受到影响。

