# 计算机网络

1. **OSI七层模型**

应用层 ：为应用程序提供服务，并管理应用程序之间的通信（SMTP、HTTP、FTP）

表示层 ：处理数据的标识问题，比如编码、格式转化、加密解密等

会话层 ：负责建立管理和断开通信连接，实现数据同步

传输层 ：端到端传输数据，同时处理传输错误、控制流量等（TCP UDP）

网络层 ：地址管理、路由选择（IP协议）

数据链路层 ：数据分割成帧，mac寻址、差错校验、信息纠正等。（以太网）

物理层 ：利用传输介质为数据链路层提供物理连接

发送端从应用层 → 物理层 打包发送

接收层从物理层 → 应用层 解析获取

1. **路由器工作在哪一层**

网络层。 路由器是连接因特网中各局域网、广域网的设备，它会根据信道的情况自动选择和设定路由，以最佳路 径，先后顺序发送信号。

路由和交换机最主要的区别就是交换机发生在OSI参考模型第二层（数据链路层），而路由发生在第三 层（网络层）。 这一区别决定了路由和交换机在移动信息的过程中需要使用不同的控制信息，所以两者实现各自功能的 方式是不同的

1. **DNS解析过程**

**知识点：**

根、顶级、权威、具体服务器，每一级都返回下一级的服务器地址

**迭代查询：**

1. 浏览器检查自身缓存，有无解析此域名对应的ip
2. 没有就去，操作系统缓存hosts文件中查询，因为可能被更改
3. 没有请求本地域名服务器（LDNS）解析域名（一般在城市某处，距离不会太远）
4. 还没有的话，就去根DNS域服务器查询，此时会给出顶级域名服务器
5. 顶级域名服务器查询，此时会给出权威域名服务器的地址，这是网站注册的域名服务器
6. 权威域名服务器的地址最后找到服务器地址
7. 去NameServer查询，根据映射关系表找到目标IP,返回给LDNS（LDNS缓存域名及IP）
8. .LDNS解析结果返回用户（缓存到系统缓存中），域名解析结束
9. **UDP和TCP之间的区别**

**UDP协议 ：**

1. 是面向无连接的，不需要在正式传递数据之前先连接起双方
2. UDP 协议只是数据报文的搬运 工，不保证有序且不丢失的传递到对端，
3. 并且UDP 协议也没有任何控制流量的算法

**TCP协议：**

1. 无论是建立连接还是断开连接都需要先需要进行握手
2. 在传输数据的过程中，通过各种算法保证数 据的可靠性（有序不丢失）

**总的来说 ：**

UDP 相 较于 TCP 更加的轻便，一般可以用于直播、即时通讯、即时游戏等

TCP的问题就是相比 UDP 来说不那么的高效

1. **TCP建立连接-三次握手**

起初，两端都为 CLOSED 状态。在通信开始前，双方都会创建 TCB。 服务器创建完 TCB 后便进入 LISTEN 状态，此时开始等待客户端发送数据

**第一次握手：**

客户端向服务端发送连接请求报文段。该报文段中包含自身的数据通讯初始序号。请求发送后，客户端 便进入 SYN-SENT 状态

**第二次握手：**

服务端收到连接请求报文段后，如果同意连接，则会发送一个应答，该应答中也会包含自身的数据通讯 初始序号，发送完成后便进入 SYN-RECEIVED 状态

**第三次握手：**

当客户端收到连接同意的应答后，还要向服务端发送一个确认报文。客户端发完这个报文段后便进入 ESTABLISHED 状态，服务端收到这个应答后也进入 ESTABLISHED 状态，此时连接建立成功

1. **TCP连接要三次握手，两次就可以连接啊？**

防止出现失效的连接请求报文段被服务端接收的情况，从而产生错误。 客户端发送了一个连接请求 A，但是因为网络原因造成了超时，这时 TCP 会启动超时重传的机制再次发 送一个连接请求B。此时请求顺利到达服务端，服务端应答完就建立了请求，然后接收数据后释放了连接。

假设这时候连接请求 A 在两端关闭后终于抵达了服务端，那么此时服务端会认为客户端又需要建立 TCP 连接，从而应答了该请求并进入 ESTABLISHED 状态。但是客户端其实是 CLOSED 的状态，那么就会 导致服务端一直等待，造成资源的浪费

1. **TCP断开连接-四次挥手**

TCP 是全双工的，在断开连接时两端都需要发送 FIN 和 ACK

**第一次握手：**

若客户端 A 认为数据发送完成，则它 需要向服务端 B 发送连接释放请求

**第二次握手：**

B收到连接释放请求后，会告诉应用层要释放 TCP 链接。然后会发送 ACK 包，并进入 CLOSE\_WAIT 状 态，此时表明 A 到 B 的连接已经释放，不再接收 A 发的数据了。但是因为 TCP 连接是双向的，所以 B 仍旧可以发送数据给 A

**第三次握手：**  
 B如果此时还有没发完的数据会继续发送，完毕后会向 A 发送连接释放请求，然后 B 便进入 LAST-ACK 状态

**第四次握手：**

A 收到释放请求后，向 B 发送确认应答，此时 A 进入 TIME-WAIT 状态。该状态会持续 2MSL（最大段 生存期，指报文段在网络中生存的时间，超时会被抛弃） 时间，若该时间段内没有 B 的重发请求的 话，就进入 CLOSED 状态。当 B 收到确认应答后，也便进入 CLOSED 状态

**注意：**

通过延迟确认的技术（通常有时间限制，否则对方会误认为需要重传），可以将第二次和第三次握 手合并，延迟 ACK 包的发送

1. **客A 进入 TIME-WAIT等待 2MSL 时间后才进入 CLOSED？**

为了保证 B 能收到 A 的确认应答。若 A 发完确认应答后直接进入 CLOSED 状态，如果确认应答因为网 络问题一直没有到达，那么会造成 B 不能正常关闭

1. **TCP如何解决数据丢包或报文顺序不对的问题？**

暂时决定不会问

1. **TCP如何实现流量控制的**

通过滑动窗口和拥堵窗口实现的。 滑动窗口主要是用于接收方，保证接收方能够接受数据。接收方通过报文告知发送方当前接收窗口剩余 大小，发送窗口根据该值变化大小滑动窗口（待发送区）发送报文。 拥堵窗口，主要用于网络，防止过多的数据拥堵网络，避免负载过大的情况

1. **Post和Get有什么区别？**

从本质上说，Post和Get都取决于http，使用哪个方法与应用层传输没有必然的联系。HTTP没有要求， 如果是POST，数据就要放在BODY中。也没有要求GET，数据（参数）就一定要放在URL中而不能放在 BODY中

**细节上有一些区别：**

1. Get能请求缓存，但是Post不可以
2. Post支持更多编码类型
3. Get回退无害，Post会再次提交
4. Get能被保存为书签，Post不可以
5. 由于浏览器Url有限制，所以Get的长度受限，但Post不受限（因为都在Body里）
6. Post相对Get更加安全
7. **Http常用首部**

**通用：**

1. cache-control ： 控制缓存行为
2. connection ： 连接的性质，比如keep-alive
3. user-Agent ：用户信息
4. Date ：报文创建时间

**请求：**

1. Referrer Policy : 表示来源的（浏览器所访问的前一个页面）,可以用于辅助检测crsf攻击， 一般浏览器的默认值是no-referrer-when-downgrade，意思是https降级http的时候不传 原地址
2. Accept : 能正确接收的媒体类型
3. Accept-XX(Accept-Charset/Accept-Encoding/Accept-Language):能正确接收的xx
4. Expect :期待服务端的指定行文
5. If-Match :两端资源标记比较
6. If-Modified-Since : 比较时间 未修改返回304 Not Modified
7. If-None-Match :比较标记 未修改返回304 Not Modified

**响应：**

1. Location : 重定向到某个location
2. Server : 服务器名字
3. Age :响应存在时间
4. Accept-Ranges ：可以接受的范围类型
5. **http请求中connection=keep-alive 的意义在哪里**

HTTP 是基于 TCP 的，每一个 HTTP 请求都需要进行三次握手。如果一个页面对某一个域名有多个请 求，就会进行频繁的建立连接和断开连接。所以HTTP 1.0 中出现了 Connection: keep-alive ，用于 建立长连接。Keep-Alive 模式更加高效，因为避免了连接建立和释放的开销。但是，长时间的TCP连接 容易导致系统资源无效占用，配置不当的keep-alive有时比重复利用连接带来的损失还更大。所以，正 确设置keep-alive timeout时间非常重要

1. **http请求中cache-control有哪些参数可以设置**
2. Public :表示任何缓存都可以缓存响应
3. private :表示响应仅供单个用户使用，不得由共享高速缓存存储。私有缓存可以存储响应
4. no-cache : 强制缓存在发布缓存副本之前将请求提交到源服务器以进行验证
5. no-store: 缓存不应存储有关客户端请求或服务器响应的任何内容
6. **http状态码**

**1XX: 通知**

1. 100 Continue 客户端应重新发请求
2. 101 Switching Protocols 改用协议 http换到https或者http1.1换到2.0之类

**2XX：成功**

1. 200 OK 操作成功
2. 201 Created按照客户端请求创建了一个新资源
3. 202 Accepted 请求无法或不被实时处理
4. 204 No Content 请求成功，但是报文不含实体的主体部分
5. 205 Reset Content 请求成功，但是报文不含实体主体部分，要求客户端重置内容
6. 206 Partial Content 进行范围请求

**3XX：重定向**

1. 301 Moved Permanently永久性重定向，资源已经被分配到了新的URL
2. 302 Found 临时重定向，资源临时分配了URL 实际上发部分客户端把它当成
3. 303处理 303 See Other 表示资源存在另一个URL。应用Get获取资源
4. 304 Not Modified 允许访问资源，但实体主体为空（客户端已经有此数据，不需要再次发 送）
5. 307 Temporary Redirect 临时重定向，资源临时分配了URL，但是希望客户端能够保持方 法不变请求新地址（解决302被当成303处理的问题）

**4XX：客户端错误**

1. 400 Bad Request 请求报文语法错误
2. 401 Unauthorized 发送的请求需要通过验证，客户端试图对一个受保护的资源操作但没有 认证证书
3. 403 Forbidden 请求资源存在但被拒绝，常用于一个资源只允许在特定时间段内访问（如果 不想透露可以谎报404）
4. 404 Not Found 找不到请求的资源
5. 405 Method Not Allowed 不支持的请求方法，比如只支持Get，但是收到了Post请求

**5XX：服务端错误**

1. 500 Internal Server Error 执行请求时发生错误（处理异常）
2. 501 Not Implemented 不支持此请求方法（和405区别在于，405是访问的资源不支持， 而501表示服务器不能操作此方法）
3. 502 Bad Gateway 代理与上行服务器之间出现问题
4. 503 Service Unavailable 服务器暂时处于超负荷或者维护中
5. **状态码返回204可能的原因**
6. 服务器拒绝请求返回 Get资源存在但表示是空的
7. 服务器通过这个响应代码告诉客户端：客户端的输入已被接受，但客户端不应该改变任何UI元素
8. **状态码204和205的区别**

204和205的区别在于205要求了重置！

用一个表单为例，如果提交后返回204，那么表单里的各个字段值不变，可以继续修改它们；但假如得 到的响应代码205，那么表单里的各个字段将被重置为它们的初始值

1. **状态码304的可能情况**

这个与浏览器的协商缓存有关。用时间的来说明：

当用户第一次请求资源A时，服务器会添加一个名为 Last-Modified 的响应头，这个头说明了A的 最后 修改时间 ，浏览器会把A的内容以及最后的响应时间缓存下来

当用户第二次请求A时，在请求中包含一个名为 If-Modified-Since 请求头，它的值就是第一次请求时 服务器通过 Last-Modified 响应头发送给浏览器的值，即资源A最后的修改时间

If-Modified-Since 请求头就是在告诉浏览器，我这里浏览器缓存的A最后修改时间是这个，你看看现 在A最后修改时间是不是这个，如果还是，那么就不用响应这个请求了，我会把缓存里的内容直接显示 出来

服务器会获取 If-Modified-Since 值，与A当前的最后修改时间作对比，如果相同，则服务器返回304 状态码，表示A与浏览器上次缓存的相同，无需再次发送，浏览器可以显示自己的缓存页面，如果比对 不同，那么说明A已经改变，服务器会返回200状态码

1. **Http协议中的长短连接和长短轮询**

**短连接：**

所谓短连接，即连接只保持在数据传输过程，请求发起，请求建立，数据返回，连接关闭。它适用 于一些实时数据请求，配合轮询来进行新旧数据的更替

**长连接：**

长连接便是在连接发起后，在请求关闭连接前客户端与服务器都保持连接，实质是保持这个通信管 道，之后便可以对其进行复用

**短轮询：**

短轮询指的是在循环周期内，不断发起请求，每一次请求都立即返回结果，根据新旧数据对比决定 是否使用这个结果

**长轮询：**

长轮询是在请求的过程中，若是服务器端数据并没有更新，那么则将这个连接挂起，直到服务器推 送新的数据，再返回，然后再进入循环周期

1. **长短连接和长短轮询的区别**
2. 决定方式。一个TCP连接是否为长连接，是通过设置HTTP的Connection Header来决定的，而且 是需要两边都设置才有效。而一种轮询方式是否为长轮询，是根据服务器端的处理方式来决定的， 与客户端没有关系。
3. 实现方式。连接的长短是通过协议来规定和实现的。而轮询的长短，是服务器通过编程的方式手动 挂起请求来实现的
4. **Http为什么不安全？**
5. 数据以明文传递，有被窃听的风险
6. 接收到的报文无法证明是发送时的报文，不能保证完整性，因此报文有被篡改的风险
7. 不验证通信两端的身份，请求或响应有被伪造的风险
8. **http和https有什么区别？**
9. HTTP是超文本传输协议，信息是明文传输 ，HTTPS则是具有安全性的SSL加密传输协议
10. HTTP和HTTPS使用的是完全不同的连接方式，用的端口也不一样，前者是80，后者是443
11. HTTPS协议需要CA申请证书，一般免费证书比较少，因而需要一定费用
12. HTTP的连接很简单，是无状态的；HTTPS协议是由TSL协议进行了加密，比HTTP协议安全
13. **TLS及其握手功能**

https仍用http传输信息，但信息通过TLS进行了加密。 TLS作用于表示层

**两种加密技术：**

1. 对称加密：两边有相同的密钥，都知道如何加密解密
2. 非对称加密：数据公钥加密，私钥解密，私钥只有发出公钥的一方知道

对称加密问题在于如何让双方都知道密码，且不被其他人知道；非对称加密可以完美解决对称加密存在的问题

服务器将公钥发散出去。之后客户端创建一个密钥，用此公钥加密后发送给服务器，服务器用私钥解密，就能得知此密钥。双方都知道密码，之后就可以采用对称加密的方式进行数据传输了

**TLS握手：**

暂时不考

1. **Https的缺点？**
2. 通信两端都需要进行加密和解密，会消耗 大量的CPU、内存等资源 ，增加了服务器的负载
3. 加密运算和多次握手降低了访问速度
4. 在开发阶段，加大了页面调试难度 。由于信息都被加密了，所以用代理工具的话，需要先解密然 后才能看到真实信息
5. 用HTTPS访问的页面，页面内的外部资源都得用HTTPS请求，包括脚本中的AJAX请求
6. **HTTPS 的单向认证和双向认证**

**单向认证：**

1. 客户端保存着服务器的证书并信任该证书
2. https一般是单向认证，这样可以让绝大部分人都可以访问你的站点

**双向认证：**

1. 先决条件是有2个或者2个以上的证书，一个服务器证书，其它是客户端证书
2. 服务器保存着客户端的证书并信任该证书，客户端保存着服务器的证书并信任该证书。这样，在证 书验证成功的情况下即可完成请求响应
3. 双向认证一般用于企业应用对接（比如说堡垒机hh）
4. **http1.0和http2.0的区别是什么？**

http2.0是对http1.0的改进，相较于http1.0更快更高效

1. http2.0实现了多路复用，用一个TCP进行连接共享，一个请求对应一个id，这样就可以发送多个请 求，接收方通过id来响应不同的请求，解决了http1.0队首阻塞和连接过多的问题。因为http2.0在同一 域名不论访问多少文件都只有一个连接，所以对服务器而言，提升的并发量是很大的
2. http2.0引入了二进制数据帧和流的概念，数据拆分成数据帧传输，并进行顺序标识，接收方收到数据 后按序组合即可获取正确数据。这样就可以并行传输了，解决了http1.0只能串行传输的问题
3. http2.0压缩头部，使用序号对头部编码，在两端备份索引表，通过对编码进行比较来判断是否需要传 输，减少了需要传输的大小。解决了http1.0中头部反复传输资源浪费的问题
4. http2.0中，服务器可以在客户端某个请求后，主动推送一些客户端一定需要的资源 。这样也能减少 请求的数目

当然http2.0也不是尽善尽美的，比如说在出现丢包的情况时，需要重新传输，后面的数据也就被阻塞 了，但是http1.0因为有多个连接，所以不会影响其他连接的传输。这样的话http2.0的性能反倒不如 http1.0了

但这个是TCP的问题了，要说改TCP也不太实际。不过也有解决方案，基于QUIC协议的http3.0就解决了 这个问题，需要简单描述一下吗

QUIC的简单描述：多路复用（ID识别）、纠错机制。

1. **http2.0 新特性**
2. **多路复用：**通过一个TCP连接传输所有数据。一个请求对应一个id，这样一个链接上可以有多个请求，每个连 接的请求可以随机的混杂在一起，接收方可以根据请求的id将请求再归属到各自不同的服务器端请 求里面
3. **二进制分帧层：**HTTP/2.0性能增强的关键，它改变了通信两端交互数据的方式，原先是以文本传输，现在要先对 数据进行二进制编码，再把数据分成一个一个的帧，接着把帧送到数据流中，最后对方接受帧并拼 接成一条消息，再处理请求
4. **首部压缩：**前面提到过的HTTP1.x的header带有大量信息，而且每次都要重复发送，HTTP2.0使用encoder来 减少需要传输的header大小，通讯双方各自cache一份header fields表，既避免了重复header的 传输，又减小了需要传输的大小
5. **服务器推送：**HTTP2.0支持服务器主动推送，简单地说就是一次请求返回多个响应，这也是一减少HTTP请求的 方法。服务器除了处理最初的请求外，还会额外push客户端一定会请求的资源，无需客户端发出 明确的请求
6. **http2.0存在的问题**

http2.0使用了多路复用，一般来说同一域名下只需要使用一个TCP连接。 但是当连接中出现丢包时，整个TCP都要开始等待重传，后面的数据也都被阻塞了。而http1.0可以开启 多个连接，只会影响一个，不会影响其他的。 所以在丢包情况下，http2.0的情况反而不如http1.0

1. **http3.0**

为了解决2.0丢包性能的问题，Google基于UDP提出了QUIC协议。 HTTP3.0中的底层支撑协议就是QUIC。所以http3.0也叫HTTP-over-QUIC

1. **QUIC协议**

UDP协议高效，但不可靠。QUIC基于UDP，在原来的基础上结合了tcp和http的精华使它可靠

**多路复用：**

HTTP2虽然是多路复用，但是TCP协议是没有这个功能的。QUIC 原始就包含此功能，并且传输的单个 数据流可以保证有序交付且不会影响其他数据流 其在移动端也会比TCP好，因为TCP基于IP+端口识别连接，不适合多变的网络环境，但是QUIC是通过 ID识别连接，不论网络如何变化，只要ID不变，就能迅速连上（实时手游就是这样实现的）

**纠错机制：**

假如说这次我要发送三个包，协议会算出这三个包的异或值并单独发出一个校验包，也就是总共发出了 四个包。 当出现其中的非校验包丢包的情况时，可以通过另外三个包计算出丢失的数据包的内容。 当然这种技术只能使用在丢失一个包的情况下，如果出现丢失多个包就不能使用纠错机制了，只能使用 重传的方式了

**0-RTT：**

通过使用类似 TCP 快速打开的技术，缓存当前会话的上下文，在下次恢复会话的时候，只需要将之前的 缓存传递给服务端验证通过就可以进行传输了