

HTTP协议版本

- HTTP/0.9
- 仅支持GET请求方式
- 仅能请求访问HTML格式的资源

- 只要不是1开头的都是Beta版本

- HTTP/1.0
 - 增加POST和HEAD请求方式
 - 支持多种数据格式的请求与访问
 - 支持cache缓存功能
 - 新增状态码、多字符集支持、内容编码等
 - 早期HTTP/1.0不支持keep-alive长连接，只支持串行连接
 - 后期HTTP/1.0增加Connection: keep-alive字段（非标准字段），开始支持长连接

- HTTP/1.1
 - 增加持久连接（默认开启Connection: keep-alive）
 - 增加管道机制（支持多个请求同时发送）
 - 增加PUT/PATCH/OPTION/DELETE等请求方式
 - 增加Host字段（指定服务器域名）-> 案例: 搜索百度 查看network
 - 增加100状态码（Continue）,支持只发送头信息
 - 增加身份认证机制
 - 支持传送内容的一部分和文件断点续传
 - 新增了24个错误状态码

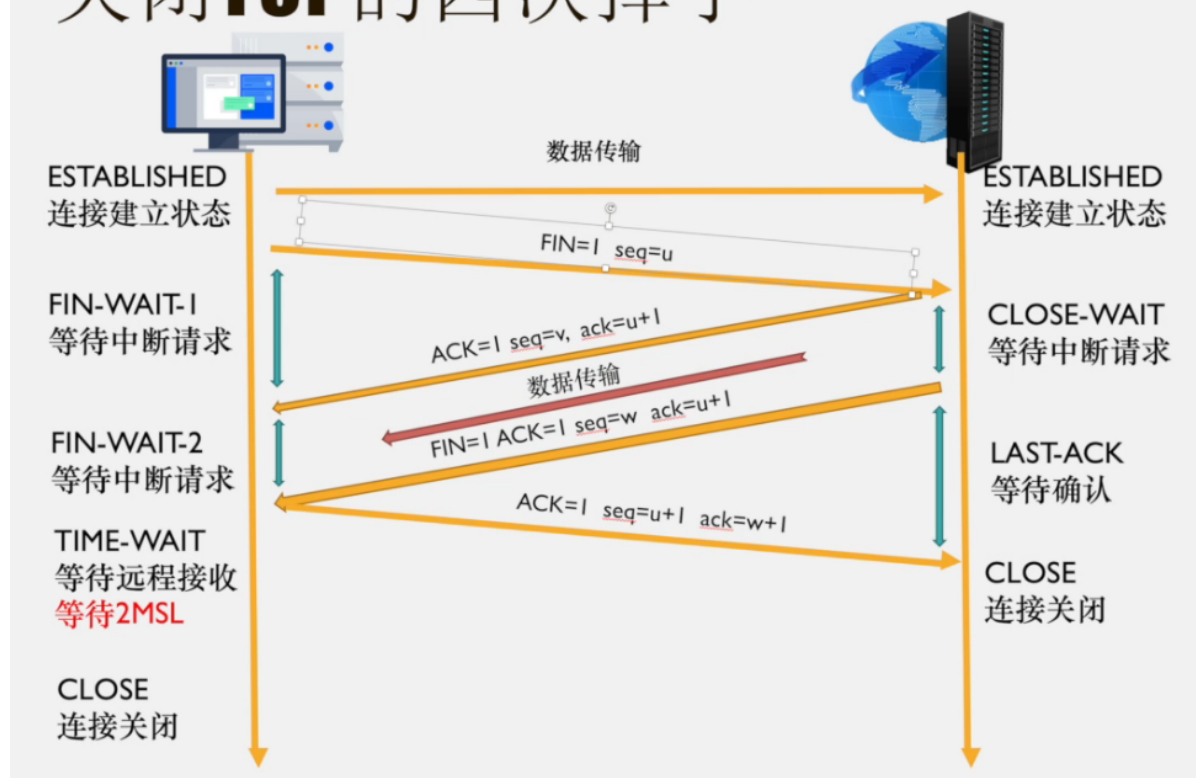
- HTTP/2.0
 - 增加双工模式（客户端同时发送多个请求，服务端同时处理多个请求）
 - 服务器推送（服务器会把客户端需要的资源一起推送到客户端，合适加载静态资源）
 - 头信息压缩机制（每次请求都会带上所有信息发送给服务端【HTTP协议不带状态】）
 - 二进制协议（头信息与数据体使用二进制进行传输）
 - 多工（先发送已处理好的部分，再响应其他请求，最后再解决没有处理好的部分）

关闭TCP

关闭TCP连接的前奏

- FIN: finish 关闭连接
- 状态:
 - FIN-WAIT-1 - 等待远程TCP的连接中断请求, 或先前的连接中断请求的确认
 - FIN-WAIT-2 - 从远程TCP等待连接中断请求
 - CLOSE-WAIT - 等待从本地用户发来的连接中断请求
 - LAST-ACK - 等待原来发向远程TCP的连接中断请求的确认
 - TIME-WAIT - 等待足够的时间以确保远程TCP接收到连接中断请求的确认
 - CLOSED - 没有任何连接状态

关闭TCP的四次挥手



- 客户端进入 FIN-WAIT-2 的状态时, 就不会在向服务端发送请求, 但是能够接收到响应
- 1、客户端发送连接关闭报文 (此时已停止发送数据) (第一次挥手)
 - 报文首部: FIN=1 (序列号seq=u)
 - 此刻: 客户端进入终止等待1 (FIN-WAIT-1) 状态
- 2、服务器收到连接关闭报文, 并发送确认报文 (第二次挥手)
 - 报文首部: ACK=1 ack=u+1 (确认FIN) (序列号seq=v)
 - 此刻: 服务器进入关闭等待 (CLOSE-WAIT) 状态
 - 说明: 连接半关闭状态, 客户端没有数据要发送, 但服务器如果还要发送数据, 客户端依然需要接受

- 3、客户端收到服务器的确认请求后，客户端进入终止等待2（FIN-WAIT-2）状态
 - （服务器在这期间还要确认客户端所需要的数据是否真的发送完毕了，如果还没发送完，则继续发送数据）
- 4、服务器确认数据已发送完毕后，向客户端发送连接关闭报文（第三次挥手），服务器进入最后确认（LAST-ACK）状态
 - 报文首部：FIN=1 ACK=1 ack=u+1（确认上一次数据包）序列号seq=w
- 5、客户端收到服务器的连接关闭报文后，发出接收确认报文（第四次挥手），客户端进入时间等待（TIME-WAIT）状态
 - 报文首部：ACK=1 ack=w+1(确认上一次数据包) 序列号seq=u+1
- 6、服务器收到客户端发出的确认，立即进入TCP关闭状态（CLOSE），TCP连接结束
 - （TCP关闭，服务端要比客户端早一些）
- TIME-WAIT时长：2MSL Maximum Segment Lifetime 最大报文生存时间
- MSL的值根据不同的情况而不同，一般是30秒 1分钟 2分钟
- 目的：保证客户端发送的最后一个报文能够打到服务器，一旦报文丢失，服务器会认为，自己最后一次发送的FIN+ACK包，客户端并没有收到，此时，服务器会重新发送一次FIN+ACK包，而客户端可以在2MSL的TIME-WAIT时间内收到重新传输的FIN+ACK包，接着重新进行第四次挥手，并重启2MSL计时器。
- 为什么是四次挥手
 - 原因：第一次挥手的时候发送了FIN包，服务器接收到以后，表示客户端不再发送数据了，但还能接收数据。这时服务器先向客户端先发送确认包，并且确认自己是否还有数据没有发送给客户端，这个确认的阶段是CLOSE-WAIT，所以在终止等待1（CLOSE-WAIT）的开始和结束需要各发送一个包，状态开始时向客户端发送的包是确认收到来自客户端的FIN包，状态结束时向客户端发送的是确认数据已经完整发送，所以是四次挥手。
- TCP连接建立后，客户端突然出现故障？
- TCP保活计时器：客户端如果出现故障，服务器每收到一次客户端的请求后都会重新复位保活计时器，时间通常是2小时，若2小时还没有收到客户端的数据，服务器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个探测报文仍无反应，服务器就认为客户端出了故障，此时将关闭连接。
- TCP 保活计时器 在任何TCP连接时都是存在的

同源策略

- 比3次握手4次挥手都重要的概念

- 同源策略 Same-Origin-Policy(SOP)
- web浏览器只允许在两个页面有相同的源时，第一个页面的脚本访问第二个页面里的数据。
- 第一个页面: <http://test.jsplusplus.com/index.html> AJAX请求 ->
- 第二个页面: <http://study.jsplusplus.com/server.php>

- 同源策略 Same-Origin-Policy(SOP)
- 报错: Access to XMLHttpRequest at 'http://study.jsplusplus.com/server.php' from origin 'http://test.jsplusplus.com' has been blocked by CORS policy: No 'Access-Control-Allow-Origin' header is present on the requested resource.
- Access to XMLHttpRequest at: 请求跨域
- 地址: <http://study.jsplusplus.com/server.php>
- from origin 'http://test.jsplusplus.com'
- 请求来自源 <http://test.jsplusplus.com>
- has been blocked by CORS policy
- 被跨域资源共享策略(Cross-origin resource sharing)阻止
- No 'Access-Control-Allow-Origin' header is present on the requested resource
- 在请求的资源中没有发现'允许跨域'头信息



- 同源策略 Same-Origin-Policy(SOP)

- <http://www.jsplusplus.com>

- | | |
|---|-----|
| • http://study.jsplusplus.com | 不同源 |
| • http://www.jsplusplus.com:8080 | 不同源 |
| • https://www.jsplusplus.com | 不同源 |
| • http://www.jsplusplus.com/course/index.html | 同源 |
| • http://www.jsplusplus.com/index/course/index.html | 同源 |

- 源(域名): 协议 + 域名 + 端口

- 同源: 相同的协议 && 相同的域名 && 相同的端口
- 不同源 (跨域): 不同的协议 || 不同的域名 || 不同的端口

- 将[server.php](http://study.jsplusplus.com/server.php)放入test.jsplusplus.com/中

- 同源策略 Same-Origin-Policy(SOP)
- 跨域操作初试
- `server.php: header('Access-Control-Allow-Origin: *');`
- `server.php: header('Access-Control-Allow-Origin: http://test.jsplusplus.com');`
- 同源策略是浏览器的一个安全功能，不同源的客户端脚本在没有明确授权的情况下，不能读写对方资源。只有同一个源的脚本赋予dom、读写cookie、session、ajax等操作的权限。

- 同源策略 Same-Origin-Policy(SOP)

- 不受同源策略限制的项：

- 1、页面的超链接
- 2、重定向页面
- 3、表单的提交
- 4、资源引入 `script` `src/link` `href/img` `src/iframe` `src`

- 只要有js引擎的浏览器都使用同源策略。