1 网络编程

1.1 OSI七层模型

- 应用层
- 表示层
- 会话层
- 传输层 TCP UDP
- 网络层 IP
- 数据链路层
- 物理层

TCP/IP概念层模型：链路层、网络层（LVS负载均衡 IP）、传输层（TCP、UDP）、应用层 HTTP

1.2 一次完整的HTTP请求的完整过程

①建立 TCP 连接（之前可能还有一次DNS域名解析）

②客户端向服务器发送请求命令

③客户端发送请求头信息

④服务服务器应答器

⑤返回响应头信息

⑥服务器向客户端发送数据

⑦服务器关闭 TCP 连接

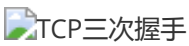
1.3 TCP、UDP的区别?

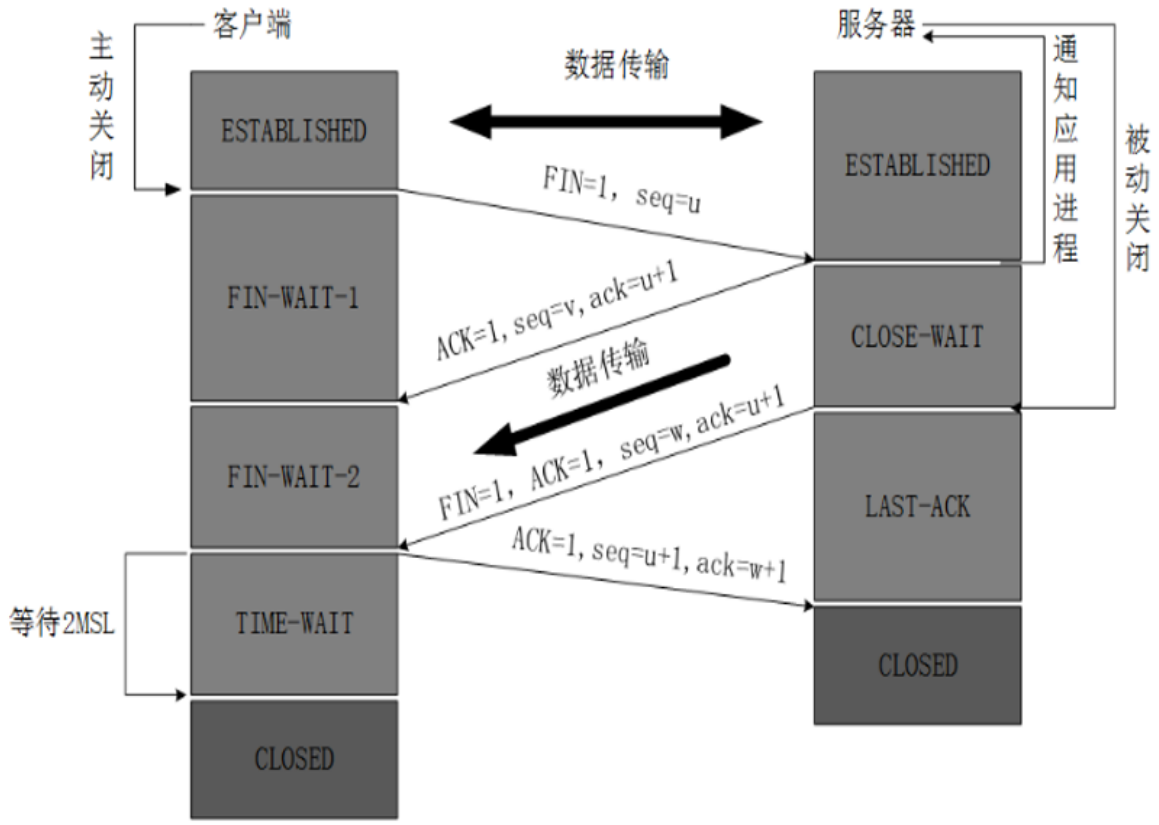
1. TCP面向连接,提供可靠传输服务，保证数据准确性；UDP 是无连接的 尽最大努力交付，不保证可靠交付。
2. UDP具有较好的实时性，工作效率比TCP高，适用于对高速传输和实时性有较高的通信。
3. 每一条TCP连接只能是点到点的;UDP支持一对一，一对多，多对一和多对多的交互通信。
4. TCP对系统资源要求较多，UDP对系统资源要求较少。

1.4 TCP协议如何保证可靠传输?

1.5 TCP的握手、挥手机制?

SYN ACK


TCP三次握手



1.6 一次完整的HTTP请求的完整过程

①建立 TCP 连接（之前可能还有一次DNS域名解析）

②客户端向服务器发送请求命令

③客户端发送请求头信息

④服务服务器应答器

⑤返回响应头信息

⑥服务器向客户端发送数据

⑦服务器关闭 TCP 连接

1.7 RESTful

https://blog.csdn.net/x541211190/article/details/81141459

RESTful是目前最流行的 API 设计规范，用于 Web 数据接口的设计。

1.接口示例

- 传统URL请求格式：
  http://127.0.0.1/user/query/1 GET 根据用户id查询用户数据

  http://127.0.0.1/user/save POST 新增用户

  http://127.0.0.1/user/update POST 修改用户信息

  http://127.0.0.1/user/delete GET/POST 删除用户信息
- RESTful请求格式：
  http://127.0.0.1/user/1 GET 根据用户id查询用户数据

http://127.0.0.1/user POST 新增用户

http://127.0.0.1/user PUT 修改用户信息

http://127.0.0.1/user DELETE 删除用户信息

1.8 为什么Netty使用NIO而不是AIO?

原因：在Linux系统上，AIO的底层实现仍使用EPOLL，与NIO相同，因此在性能上没有明显的优势；Windows的AIO底层实现良好，但是Netty开发人员并没有把Windows作为主要使用平台考虑。

1.9 TCP的粘包/拆包原因及其解决方法是什么?

1.10 Netty的粘包/拆包是怎么处理的，有哪些实现?

1.11 同步与异步、阻塞与非阻塞的区别?

1.12 说说网络IO模型?

8、BIO、NIO、AIO分别是什么?

9、select、poll、epoll的机制及其区别?

10、说说你对Netty的了解?

11、Netty跟Java NIO有什么不同，为什么不直接使用JDK NIO类库?

12、Netty组件有哪些，分别有什么关联?

13、说说Netty的执行流程?

14、Netty高性能体现在哪些方面?

15、Netty的线程模型是怎么样的?

16、Netty的零拷贝提体现在哪里，与操作系统上的有什么区别?

17、Netty的内存池是怎么实现的?

18、Netty的对象池是怎么实现的?

19、在实际项目中，你们是怎么使用Netty的?

20、使用过Netty遇到过什么问题?

1.13 浅谈Http和Https有什么区别

- HTTP是不安全的：数据拦截、数据篡改、数据攻击

  HTTPS的安全需求：数据加密、身份验证 、数据完整性
- 数据加密算法

  对称加密：加解密只有一个密钥

  非对称加密：公钥加密，私钥解密
- HTTPS比HTTP多出的事情：
  - 请求https连接获取证书（公钥）CA证书机构

    - 客户端给服务器发送公钥加密的随机数密文
    - 客户端同时给服务器发送公钥加密的随机数+私钥的密文
    - 服务器根据公钥解出随机数，同时解出私钥
    - 客户端使用非对称加密进行数据传输，客户端使用公钥加密，服务器使用私钥解密。

1.14 get和post请求的区别

get和post从实现本质上讲都是http->TCP协议，是没有区别的。但是GET产生一个TCP数据包；POST产生两个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）；
- 而对于POST，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）。

他们的区别主要体现在以下几个方面：

- get请求用来从服务器上获得资源，而post是用来向服务器提交数据；
- GET参数通过URL传递，POST放在Request body中, 因此POST比GET更安全，因为GET参数直接暴露在URL上，所以不能用来传递敏感信息。
- get传输的数据要受到URL长度限制（最大长度是 2048 个字符）；而post可以传输大量的数据，上传文件通常要使用post方式；
- GET请求只能进行url编码，而POST支持多种编码方式。