其他协议

@M了个J
李明杰

https://github.com/CoderMJLee

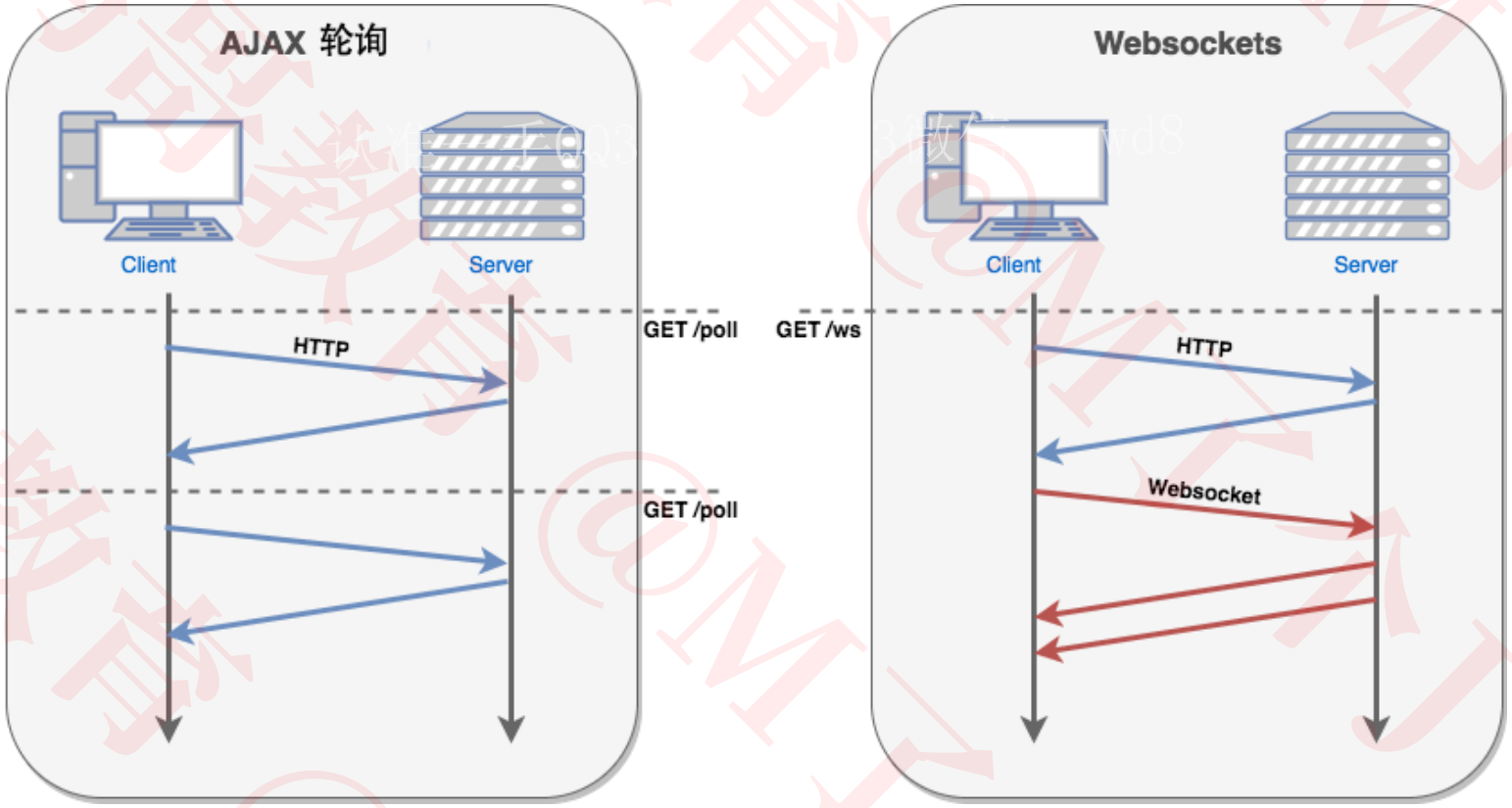
https://space.bilibili.com/325538782

小码哥教育 SEEMYGO

实力IT教育 www.520it.com

HTTP vs WebSocket

- HTTP请求的特点：通信只能由客户端发起。所以，早期很多网站为了实现推送技术，所用的技术都是轮询
  - 轮询是指由浏览器每隔一段时间（如每秒）向服务器发出HTTP请求，然后服务器返回最新的数据给客户端
  - 为了能更好的节省服务器资源和带宽，并且能够更实时地进行通讯，HTML5规范中出现了WebSocket协议



WebSocket

- WebSocket，是基于TCP的支持**全双工**通信的应用层协议
  - 在2011年由IETF标准化为RFC 6455，后由RFC 7936补充规范
  - 客户端、服务器，任何一方都可以主动发消息给对方

- WebSocket的应用场景很多
  - 社交订阅、股票基金报价、体育实况更新、多媒体聊天、多玩家游戏等

HTTP vs WebSocket

- WebSocket和HTTP属于平级关系，都是应用层的协议
  - 其实TCP本身就是支持全双工通信的（客户端、服务器均可主动发消息给对方）
  - 只是HTTP的“请求-应答模式”限制了TCP的能力

- WebSocket使用80（ws://）、443（wss://）端口，可以绕过大多数防火墙的限制
  - ws://example.com/wsapi
  - wss://secure.example.com/wsapi

- 与HTTP不同的是，WebSocket需要先建立连接
  - 这就使得WebSocket成为一种有状态的协议，之后通信时可以省略部分状态信息
  - 而HTTP请求可能需要在每个请求都额外携带状态信息（如身份认证等）

WebSocket – 建立连接

- WebSocket需要借助HTTP协议来建立连接（也叫作握手，Handshake）
  - 由客户端（浏览器）主动发出握手请求

```
GET /chat HTTP/1.1
Host: server.example.com
Upgrade: websocket
Connection: Upgrade
Sec-WebSocket-Key: dGhlIHNhbXBsZSBub25jZQ==
Origin: http://example.com
Sec-WebSocket-Protocol: chat, superchat
Sec-WebSocket-Version: 13
```

```
HTTP/1.1 101 Switching Protocols
Upgrade: websocket
Connection: Upgrade
Sec-WebSocket-Accept: s3pPLMBiTxaQ9kYGzzhZRbK+xOo=
Sec-WebSocket-Protocol: chat
```

- Connection必须设置Upgrade
  - 表示客户端希望连接升级
- Upgrade必须设置websocket
  - 表示希望升级到WebSocket协议
- Sec-WebSocket-Version
  - 表示支持的Websocket版本
  - RFC 6455要求使用的版本是13

WebSocket – 建立连接

- Sec-WebSocket-Key是客户端生成的随机字符串，比如例子中的dGhlIHNhbXBsZSBub25jZQ==

- 服务器接收到客户端的Sec-WebSocket-Key后，会进行以下操作

① Sec-WebSocket-Key加上一个固定的GUID值（258EAFA5-E914-47DA-95CA-C5AB0DC85B11）

- dGhlIHNhbXBsZSBub25jZQ==258EAFA5-E914-47DA-95CA-C5AB0DC85B11

② 将①的结果进行SHA-1摘要计算

- b37a4f2cc0624f1690f64606cf385945b2bec4ea

③ 将②的结果进行Hex To Base64编码

- s3pPLMBiTxaQ9kYGzzhZRbK+xOo=

④ 将③的结果做为Sec-WebSocket-Accept响应头的值，返回给客户端

- 如此操作，可以尽量避免普通HTTP请求被误认为WebSocket协议

WebSocket – 使用

- WebSocket体验和演示
  - https://www.websocket.org/echo.html

- W3C标准化了一套WebSocket API，可以直接使用JS调用

```
let ws = new WebSocket('wss://example.com')
```

WebService

- WebService，译为：Web服务，是一种跨编程语言和跨操作系统平台的远程调用技术标准

- WebService使用场景举例
  - 天气预报、手机归属地查询、航班信息查询、物流信息查询等
  - 比如天气预报，是气象局把自己的服务以WebService形式暴露出来，让第三方程序可以调用这些服务功能
  - http://www.webxml.com.cn/zh_cn/index.aspx

- 事实上，WebService完全可以用普通的Web API取代（比如HTTP + JSON）
  - 现在很多企业的开放平台都是直接采用Web API

WebService – 核心概念

- SOAP（Simple Object Access Protocol），译为：简单对象访问协议
  - 很多时候，SOAP = HTTP + XML
  - WebService使用SOAP协议来封装传递数据

- WSDL（Web Services Description Language），译为：Web服务描述语言
  - 一个XML文档，用以描述WebService接口的细节（比如参数、返回值等）
  - 一般在WebService的URL后面跟上?wsdl获取WSDL信息
    - ✓比如：http://ws.webxml.com.cn/WebServices/WeatherWS.asmx?wsdl

RESTful - 简介

- REST的全称是：REpresentational State Transfer
  - 译为“表现层状态转移”

- REST是一种互联网软件架构设计风格
  - 定义了一组用于创建Web服务的约束
  - 符合REST架构的Web服务，称为RESTful Web服务

RESTful – 实践建议

- URL中使用名词（建议用复数形式），不使用动词
  - 推荐：/users、/users/6
  - 不推荐：~~/listUsers~~、~~/getUser?id=6~~、~~/user/list~~、~~/user/get?id=6~~

- 使用HTTP的方法表达动作

	GET：查询	POST：创建	PUT：更新	DELETE：删除
/users	查询所有的用户	创建一个用户	更新所有用户的信息	删除所有的用户
/users/6	查询id为6的用户	405 Method Not Allowed	更新id为6的用户的信息	删除id为6的用户

- 一个资源连接到其他资源，使用子资源的形式
  - GET /users/6/cars/88
  - POST /users/8/cars

RESTful - 实践建议

- API版本化
  - mj.com/v1/users
  - mj.com/v2/users/66

- 返回JSON格式的数据

- 发生错误时，不要返回200状态码

HTTPDNS

- HTTPDNS是基于HTTP协议向DNS服务器发送域名解析请求
- 替代了基于DNS协议向运营商Local DNS发起解析请求的传统方式
- 可以避免Local DNS造成的域名劫持和跨网访问问题
- 常用在**移动互联网**中（比如在Android、iOS开发中）

优势点	XX云HTTPDNS	运营商Local DNS
高速	**高速访问** 接入节点覆盖国内Top17运营商、东南亚及北美，解析精准，访问迅速	**访问缓慢** 用户跨网访问、解析异常问题
安全	**防劫持** 绕开运营商Local DNS，无劫持，防止DNS被污染拦截	**广告劫持** 域名解析结果被指向广告页面、插入第三方广告
智能	**精准调度** 精确识别来源请求，访问导向最准确节点	**解析转发** 自身不进行域名递归解析，而把请求转发给其他运营商
可靠	**负载均衡** 一个IP三地集群容灾，秒级自动故障切换，服务提供99%以上的SLA	**异常故障** 缓存服务器运维环境参差不齐，时有故障

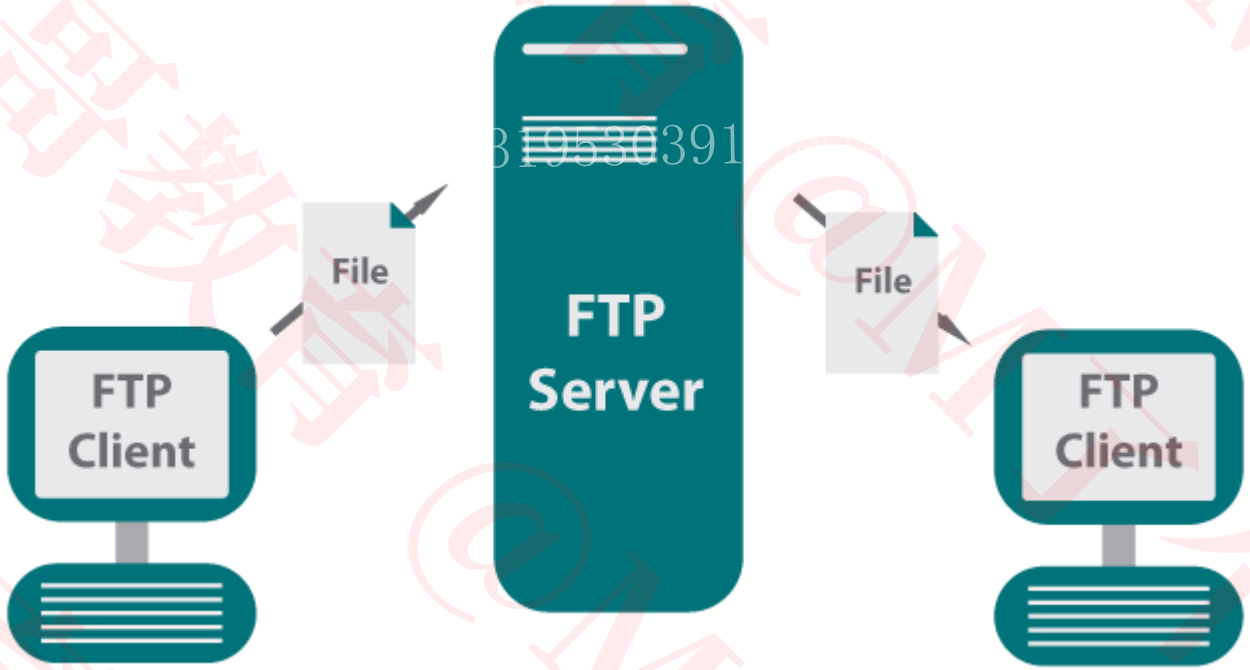
HTTPDNS – 使用

- 市面上已经有现成的解决方案
  - 腾讯云：https://cloud.tencent.com/product/httpdns
  - 阿里云：https://help.aliyun.com/product/30100.html

- 移动端集成相关的SDK即可使用HTTPDNS服务

FTP

- FTP（**F**ile **T**ransport **P**rotocol），译为：文件传输协议，RFC 959定义了此规范，是基于TCP的应用层协议
- 在RFC 1738中有定义，FTP的URL格式为：**ftp://[user[:password]@]host[:port]/url-path**



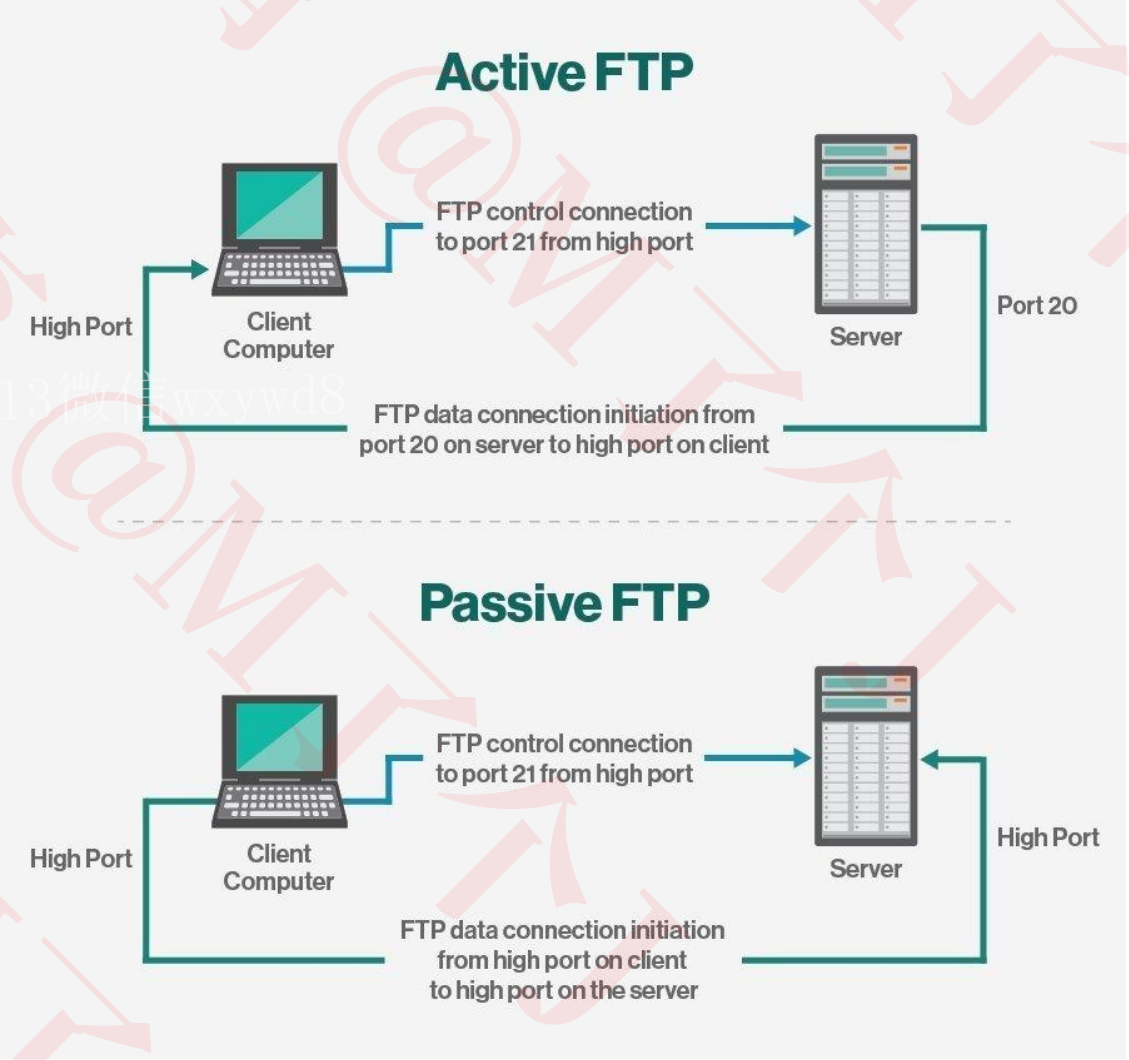
FTP – 连接模式

- FTP有2种连接模式：主动（Active）和被动（Passive）

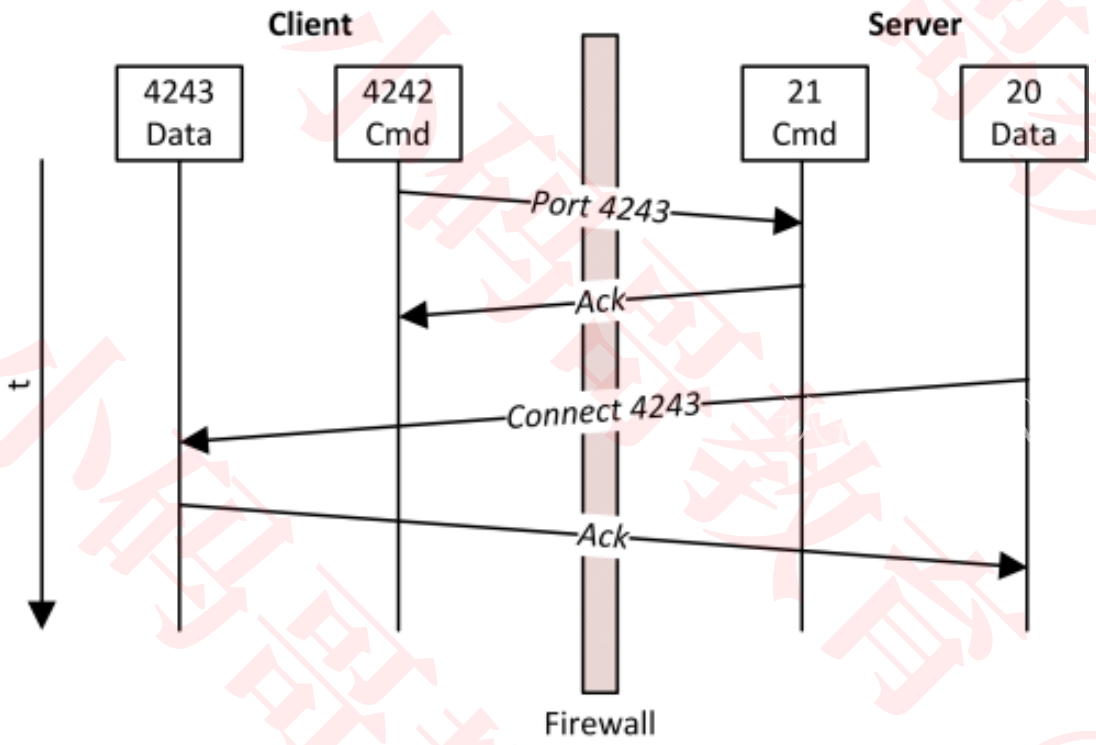
- 不管是哪种模式，都需要客户端和服务器建立2个连接

① 控制连接：用于传输状态信息（命令，cmd）

② 数据连接：用于传输文件和目录信息（data）



FTP – 主动模式



① 客户端打开一个随机的命令端口

- 端口号大于1024，假设为N
- 同时连接至服务器的命令端口21

② 客户端开始监听N+1数据端口

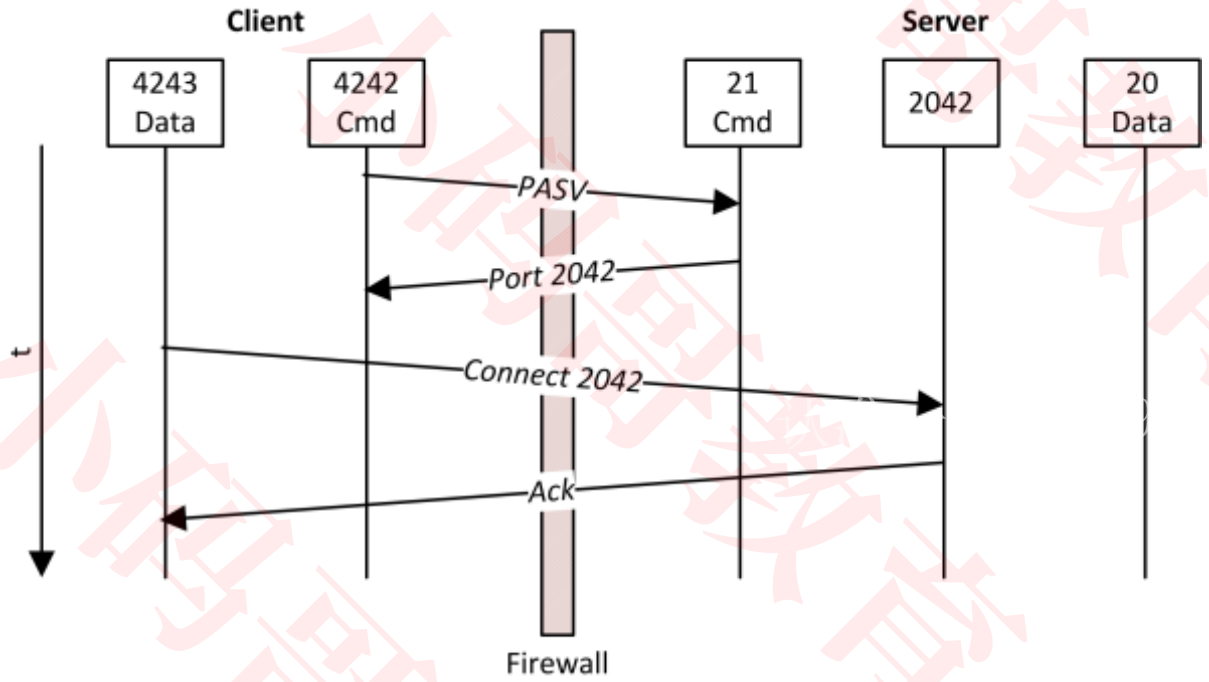
- 同时向服务器发送一个Port命令给服务器的命令端口21
- 此命令告诉服务器

✓ 客户端正在监听的数据端口N+1

✓ 并且已准备好从此端口接收数据

③ 服务器打开20号数据端口，并且创建和客户端数据端口（N+1）的连接

FTP – 被动模式



- 客户端通过两个随机的端口与服务器建立连接
  - 命令端口N
  - 数据端口N+1

① 客户端的命令端口N用于连接服务器的命令端口21

② 客户端通过命令端口N发送PASV命令给服务器的命令端口21

③ 服务器打开一个随机的数据端口P，并告知客户端该端口号P

④ 客户端数据端口N+1发起与服务器数据端口P的连接

邮件相关的协议

■ 发邮件使用的协议

□ SMTP（**S**imple **M**ail **T**ransfer **P**rotocol），译为：简单邮件传输协议

✓ 基于TCP，标准参考RFC 5321

✓ 服务器默认使用25端口，SSL/TLS使用465端口

■ 收邮件使用的协议

□ POP（**P**ost **O**ffice **P**rotocol），译为：邮局协议

✓ 基于TCP，最新版是POP3，标准参考RFC 1939

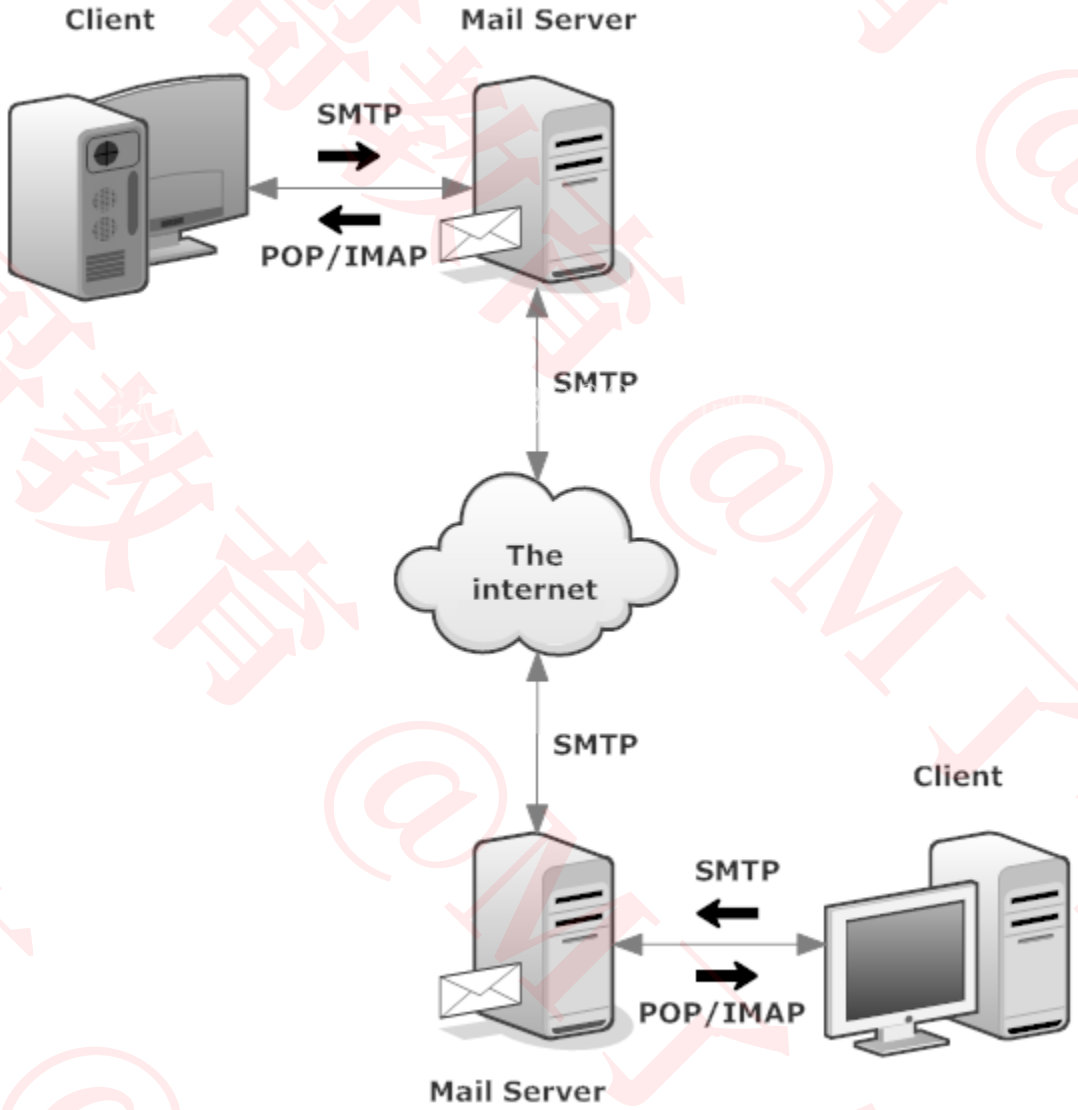
✓ 服务器默认使用110端口，SSL/TLS使用995端口

□ IMAP（**I**nternet **M**essage **A**ccess **P**rotocol），译为：因特网信息访问协议

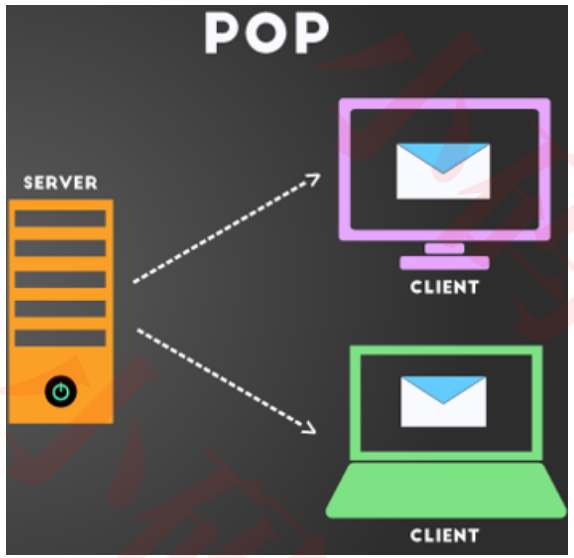
✓ 基于TCP，最新版是IMAP4，标准参考RFC 3501

✓ 服务器默认使用143端口，SSL/TLS使用993端口

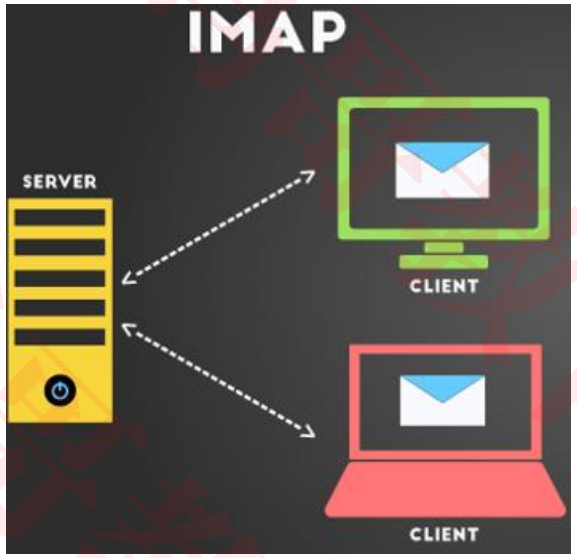
收发邮件的过程


Client
Mail Server
SMTP
POP/IMAP
SMTP
The internet
SMTP
Client
SMTP
POP/IMAP
Mail Server

POP vs IMAP



- POP的特点
  - 客户端连接服务器时，将会从服务器下载所有邮件
    - ✓ 可以设置下载完后，立即或一段时间后删除服务器邮件
  - 客户端的操作（比如删除邮件、移动到文件夹）**不会**跟服务器同步
  - 每个客户端都是独立的，都可以获得其自己的电子邮件副本



- IMAP的特点
  - 客户端连接服务器时，获取的是服务器上邮件的基本信息，并不会下载邮件
    - ✓ 等打开邮件时，才开始下载邮件
  - 客户端的操作（比如删除邮件、移动到文件夹）**会**跟服务器同步
  - 所有客户端始终会看到相同的邮件和相同的文件夹