对比：

打电话

邮寄信件

A picture containing text

Description automatically generated

邮件上地址很重要，那么网络通信包如何定位？

IP IP地址分配

域名 域名解析

www.baidu.com

<http://182.61.200.7/>

网络数据如何邮寄？

封包 和 解包

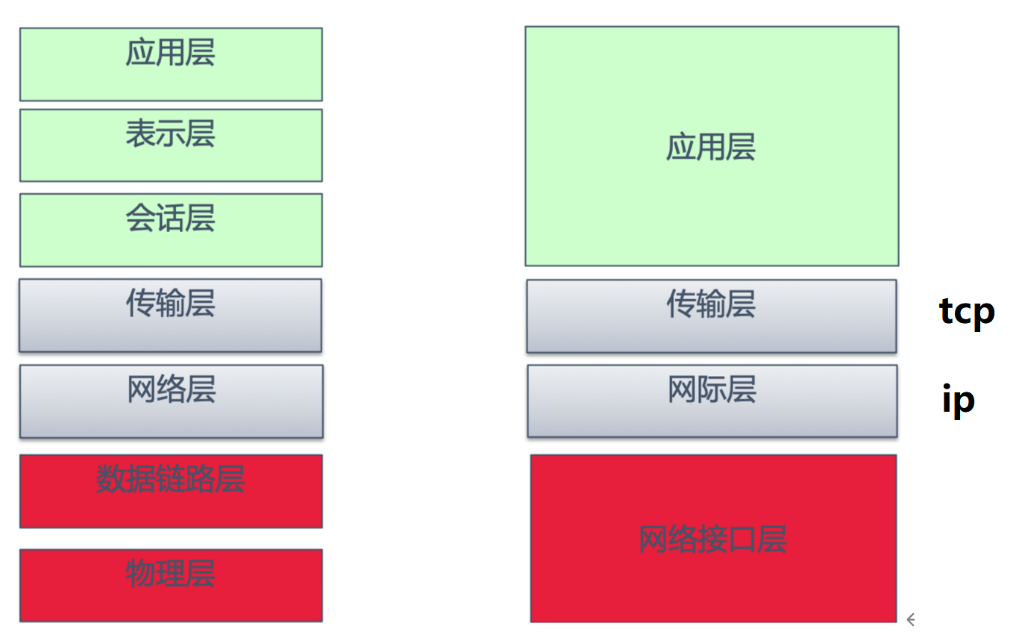
专业的人 做 专业的事，就是类和对象的设计思想

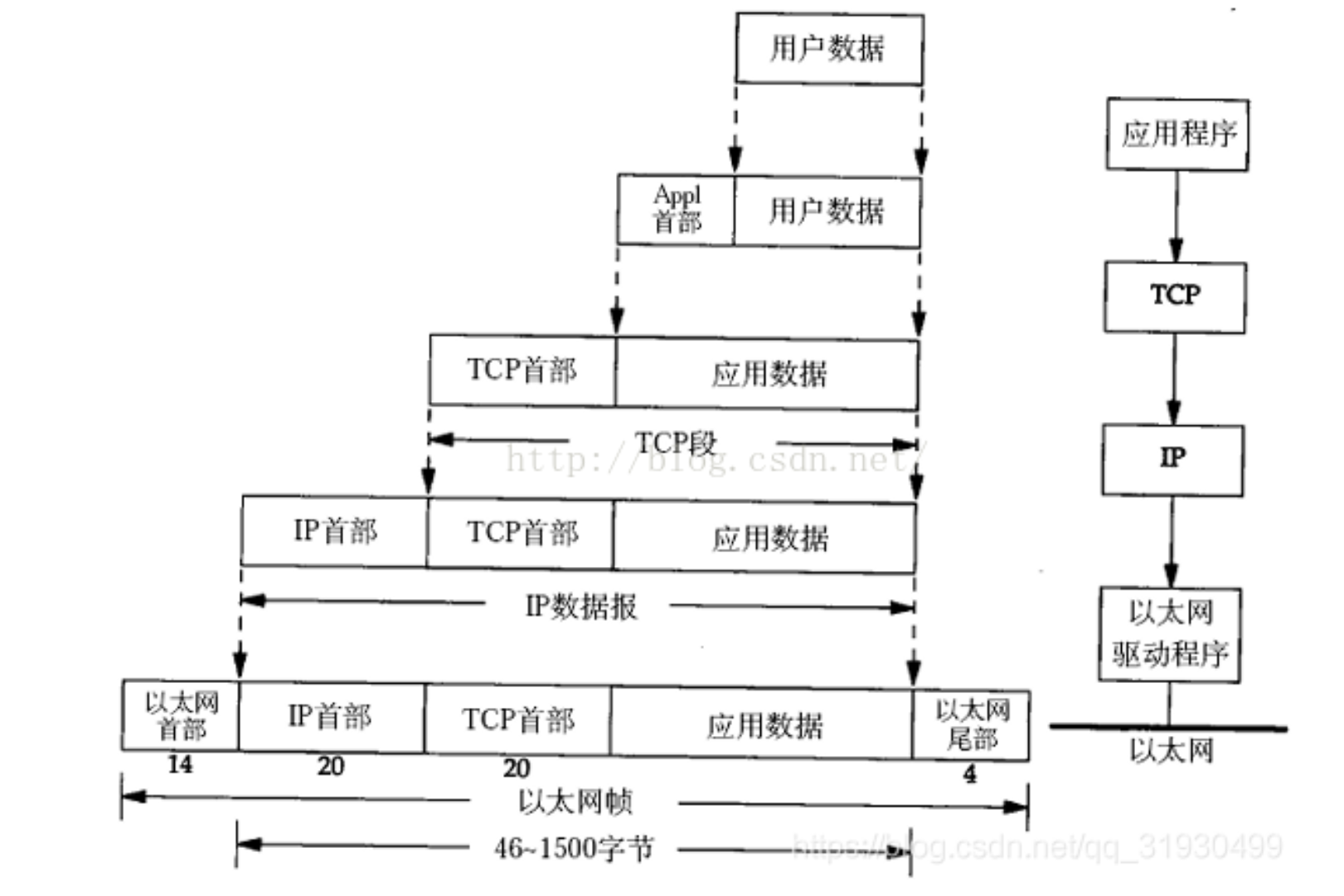
OSI七层网络模型

A picture containing table

Description automatically generated

TCP/IP四层网络模型





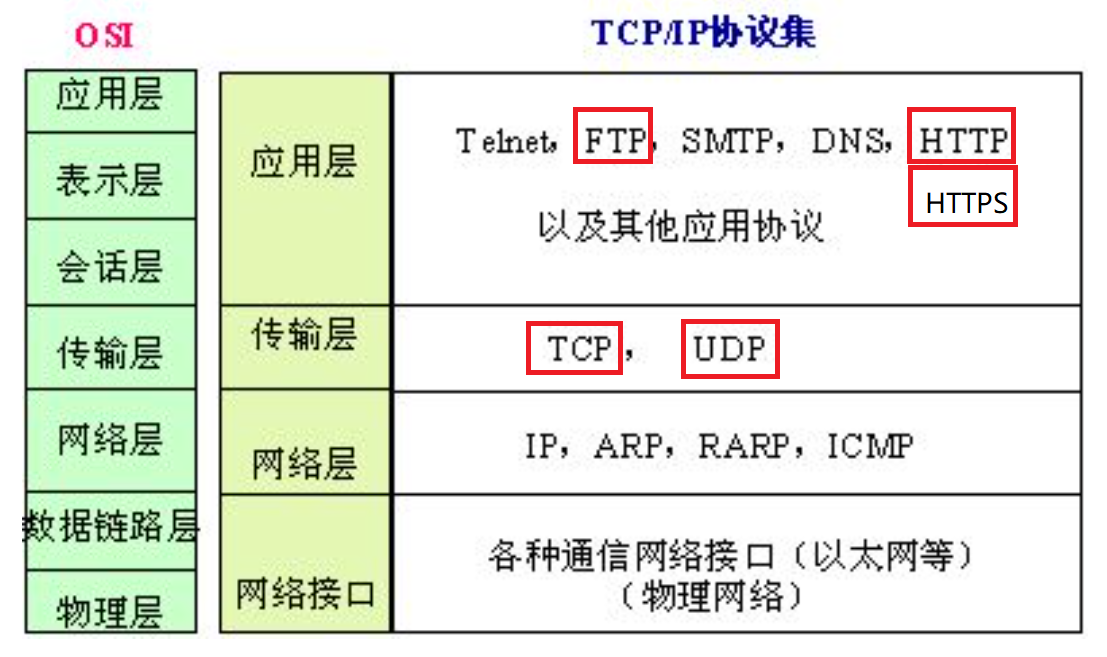
**网络协议之TCP/IP协议包**

<https://blog.csdn.net/qq_31930499/article/details/109261186>

**TCP和UDP的不同**

<https://www.cnblogs.com/shouhu/p/12163238.html>

其实虽然网络协议叫TCP/IP，并不是只表示TCP和IP，而是利用TCP/IP的协议统称。



网络拦截，网络解包导致

网络传输的不安全性，网络传输的加密

RSA 加密 (非对称加密，公钥/私钥加密)

Https Ssl/ssh

# [SSH加密原理](https://www.cnblogs.com/cn-leoblog/p/15665349.html)

<https://www.cnblogs.com/cn-leoblog/p/15665349.html>

git基于HTTPS/SSH加密，所以需要RSA密钥才能访问git server

**windows 网络编程大汇总**

<https://blog.csdn.net/baidu_16370559/article/details/104646624>

**MFC Class Library Overview**

https://docs.microsoft.com/en-us/cpp/mfc/class-library-overview?view=msvc-170

Windows有关网络编程的相关类见MSDN link:

**Win32 Internet Classes**

<https://docs.microsoft.com/en-us/cpp/mfc/win32-internet-classes?view=msvc-170>

**Windows Sockets Classes**

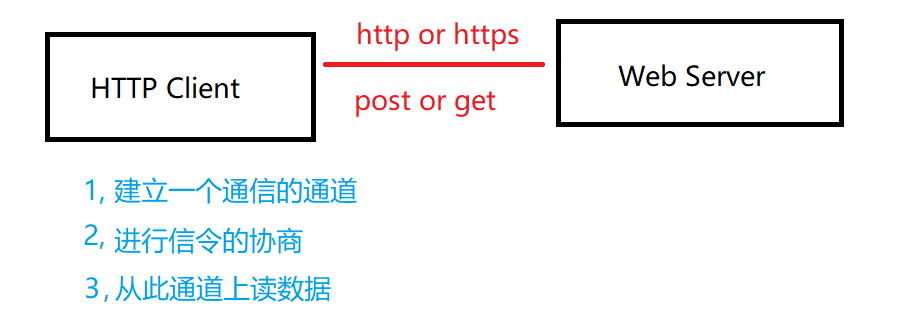
https://docs.microsoft.com/en-us/cpp/mfc/windows-sockets-classes?view=msvc-170

网络通信是存在于两台电脑之间，其中一台作为Server，另一台作为Client，通过本地网卡的自通信能力，也可以存在于同一台电脑的两个程序之间，同样也是一个程序作为Server，另一个程序作为Client。

当然由于Server和Client有许多相同的代码，为了代码的统一，也可以用同一个程序，但是它一定是同时具备有Server和Client的功能的。

HTTP Download，就是Browser的下载基础

基于Win32 Internet Classes



**Steps in a Typical HTTP Client Application**

<https://docs.microsoft.com/en-us/cpp/mfc/steps-in-a-typical-http-client-application?view=msvc-170>

Graphical user interface, text, application, email

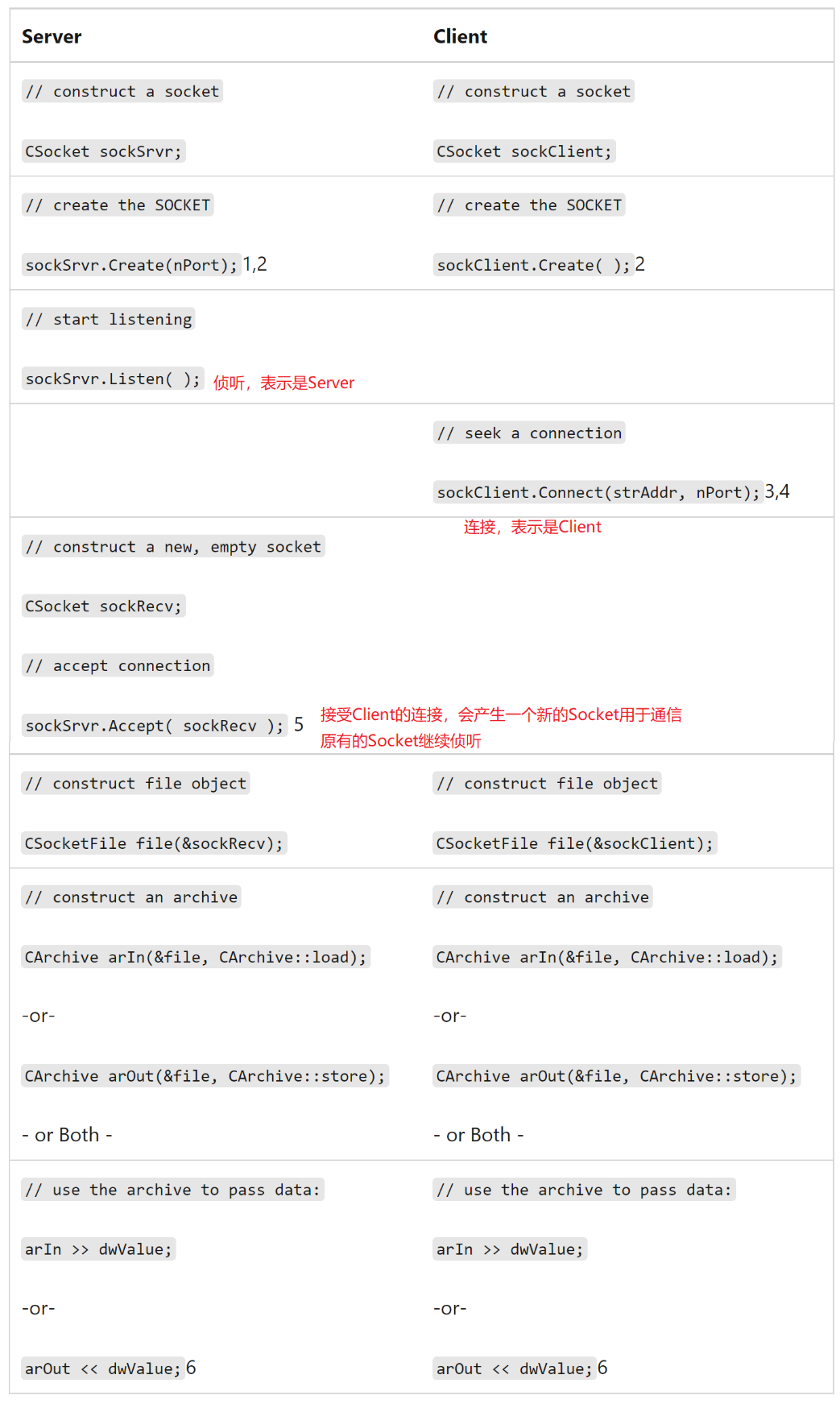
Description automatically generated

Chat Applicaton，就是WeChat，QQ等聊天程序的网络通信基础

基于Windows Sockets Classes

# Windows Sockets: Sequence of Operations

https://docs.microsoft.com/en-us/cpp/mfc/windows-sockets-sequence-of-operations?view=msvc-170



Socket的通信模型

Socket也叫套接字，是UC Berkley发明的一种网络通信机制，是一种基于缓存的通信模型

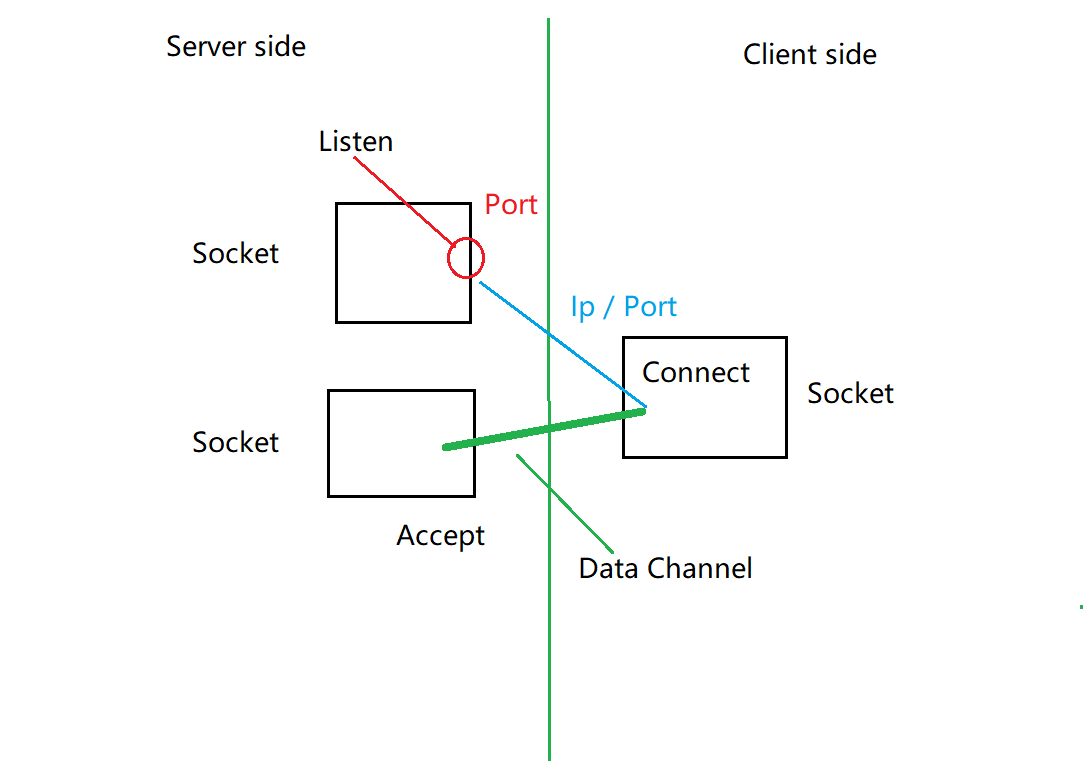
1，Server先启动，在某个端口Port侦听Listen，等待Client进行连接Connect。

2，Client后启动，通过Ip Address找到对应的Server，然后通过端口Port找到侦听的Socket，然后发起连接Connect请求。

3，Server收到连接Connect后，接收Accept请求，重新创建一个新的Socket，和Client的Socket建立数据通道Data Channel。

4，通过这条数据通道Data Channel，双方进行数据通信。

5，Server端原来的Socket继续侦听Listen。



Windows Socket是一个异步Socket，不会对主程序的调用阻塞，你可以想像它在底层使用了线程进行Socket相关的工作 （其实它在底层是使用了Windows的消息机制，因为Windows的消息机制从本质上就是异步的）

既然是异步的，那上层程序是如何知道Socket的状态，什么时候该发送数据，什么时候该接收数据呢？这里使用了一个叫Notify的通知机制，也叫Callback的回调机制

# Windows Sockets: Socket Notifications

https://docs.microsoft.com/en-us/cpp/mfc/windows-sockets-socket-notifications?view=msvc-170

