一.网络编程入门

1.软件结构

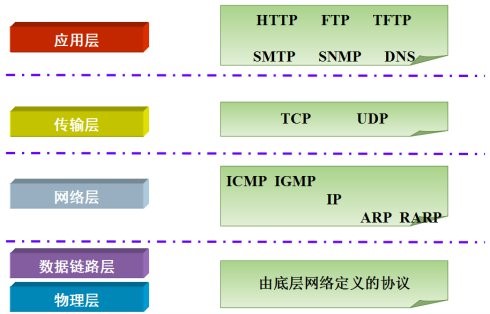
* C/S结构：全称为 Client/Server 结构，是指客户端和服务器结构
* B/S结构：全称为 Browser/Server 结构，是指浏览器和服务器结构

两种软件架构各有优势，但是无论哪种架构，客户端、浏览器与服务器的交互都离不开网络的支持。网络编程，就是在一定的协议下，实现两台计算机之间通信的程序。

2.网络通信协议

* 网络通信协议：是指要通信的计算机之间都必须遵守的规则，只有遵守这些协议规则，计算机之间才能进行通信。协议中对数据的传输格式、传输速率、传输步骤等做了统一规定，通信双方必须同时遵守这些规则，才能最终完成数据交换。其中最常用的就是TCP/IP协议。

* TCP/IP协议： 传输控制协议/因特网互联协议(Transmission Control Protocol/Internet Protocol)，是因特网最基本、最广泛的协议。它定义了计算机如何连入因特网，以及数据如何在它们之间传输的标准。TCP/IP协议是一个协议族，它的内部包含一系列用于处理数据通信的协议，并采用了四层的分层模型，TCP/IP协议中的四层分别是应用层、传输层、网络层和物理层(数据链路层)，每层分别负责不同的通信功能，并且每一层都呼叫它的下一层所提供的协议来完成自己的需求。



物理层：物理层是用于定义物理传输通道，通常是对某些网络连接设备的驱动协议，例如针对光纤、网线提供的驱动。

网络层：网络层是整个TCP/IP协议的核心，它主要用于将传输的数据进行分组，将分组数据发送到目标计算机或者网络。

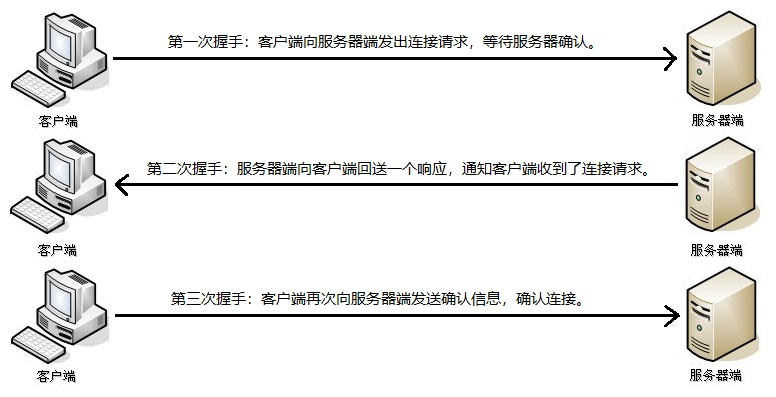
传输层：主要使网络程序进行通信，在进行网络通信时，可以采用TCP协议，也可以采用UDP协议。

应用层：主要负责应用程序的协议，例如HTTP协议、FTP协议等。

3.常用协议介绍

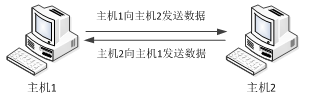
java.net 包中包含的类和接口，它们提供低层次的通信细节。我们可以直接使用这些类和接口，来专注于网络程序开发，而不用考虑通信的细节。java.net 包中提供了两种常见的用于网络程序的协议支持：

* TCP协议：传输控制协议(Transmission Control Protocol)，TCP协议是面向连接的通信协议，即传输数据之前，在发送端和接收端建立逻辑连接，然后再传输数据，它提供了两台计算机之间可靠无差错的数据传输。TCP协议中，在发送数据的准备阶段，客户端与服务器之间要进行三次交互，称为三次握手，以保证连接的可靠。整个交互过程如下图所示。



完成三次握手，连接建立后，客户端和服务器就可以开始进行数据传输了。由于这种面向连接的特性，TCP协议可以保证传输数据的安全，不会丢失数据，所以应用十分广泛，例如下载文件、浏览网页等。

* UDP协议：用户数据报协议(User Datagram Protocol)，UDP协议是无连接通信协议，即在数据传输时，数据的发送端和接收端不建立逻辑连接。简单来说，就是当一台计算机向另外一台计算机发送数据时，发送端不会确认接收端是否存在，就会发出数据，同样接收端在收到数据时，也不会向发送端反馈是否收到数据。UDP协议以"数据报(Datagram)"作为数据传输的基本单位，但每次发送的数据都被限制在64kb以内，超出这个范围就不能发送了。整个交互过程如下图所示。



由于UDP协议的这种面向无连接性，导致使用UDP协议消耗资源小，通信效率高，可能会丢失数据。所以通常都会用于音频、视频和普通数据的传输，例如视频会议都使用UDP协议，因为这种情况即使偶尔丢失一两个数据包，也不会对接收结果产生太大影响，但是在传输重要数据时不建议使用UDP协议。

4.网络编程三要素

(1).协议

(2).IP地址：指互联网协议地址(Internet Protocol AddressIP)，IP地址用来给一个网络中的计算机设备做唯一的编号，分为以下两类

* IPv4：是一个32位的二进制数字，通常被分为4个字节，并且每个字节以十进制数字来表示，最终形成 192.168.65.100 的形式，其中每个字节都是0~255之间的十进制整数，那么最多可以表示42亿个IP地址。
* IPv6：由于IP地址的需求量愈来愈大，使得IP地址的分配越发紧张。为扩大地址空间，拟通过IPv6重新定义地址空间，采用一个128位的二进制数字，通常被分为8组，每组2个字节，并且将每组的2个字节以十六进制数字来表示，形成 ABCD:EF01:2345:6789:ABCD:EF01:2345:6789 的形式，号称可以为全世界的每一粒沙子编上一个地址，这样就解决了IP地址资源数量不够的问题。

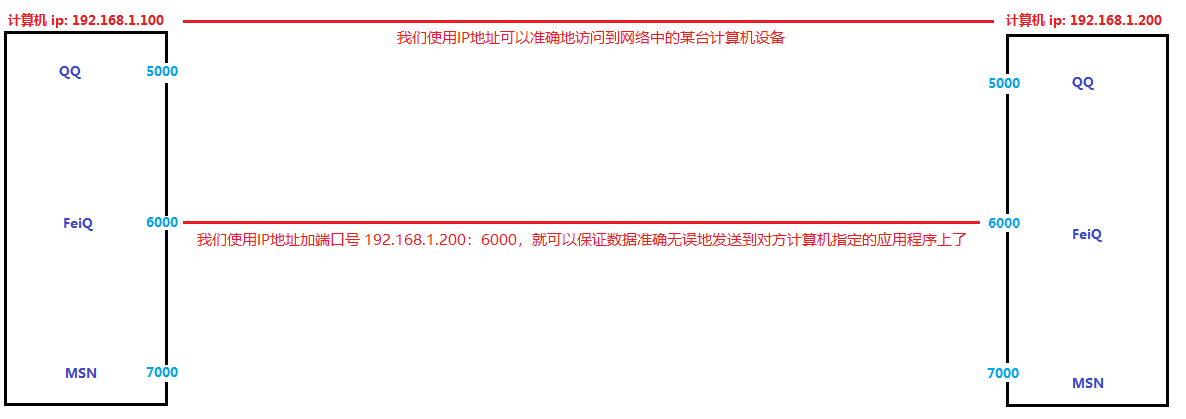
#.注意：

①.由于IP地址是数字组成的，不方便记忆，所以有了域名，通过DNS服务器可以将域名解析为相应的IP地址，所以域名就是文字形式记录的IP地址(如百度服务器的域名就是：www.baidu.com)。IP地址与域名是一对多的关系，一个IP地址可以对应多个域名，但是一个域名只有一个IP地址。

②.本机IP地址：127.0.0.1，域名为localhost，称为本机回送地址(Loopback Address)。主要用于网络软件测试以及本地机进程间通信，无论什么程序，一旦使用回送地址发送数据，立即返回，不进行任何网络传输。

(3).端口号：

网络的通信，本质上是两个进程(应用程序)之间的通信。我们使用IP地址可以准确地访问到每台计算机设备，但是每台计算机都有很多进程，那么在网络通信时，如何区分这些进程并将数据准确发送到相应的应用程序上呢？如果说IP地址可以唯一标识网络中的设备，那么端口号就可以唯一标识设备中的进程(应用程序)。



当我们打开一个应用程序时，操作系统就会为该进程分配一个随机的端口号，是用两个字节表示的整数，取值范围是0~65535。其中，0~1023之间的端口号用于一些知名的网络服务和应用，普通的应用程序需要使用1024以上的端口号。如果端口号被另外一个服务或应用所占用，会导致当前程序启动失败。常用的端口号： Mysql数据库：3306 Oracle数据库：1521 Tomcat服务器：8080

网络编程三要素：利用 "协议 + IP地址 + 端口号" 三元组合，就可以唯一标识网络中的一个进程了，那么进程间的通信就可以利用这个标识与其它进程进行交互。

二.TCP通信程序

基于TCP协议的TCP通信，能实现两台计算机之间的数据交互，通信的两端，要严格区分为客户端(Client)与服务器端(Server)。由于TCP协议是面向连接的通信协议，所以客户端和服务器端必须经过三次握手交互，才能建立逻辑连接实现通信。java.net 包中提供了两个类用于实现TCP通信程序：

(1). 客户端：java.net.Socket 类实现客户端。创建 Socket 对象，就是开启一个客户端，并向服务器端发出连接请求。客户端必须主动连接服务器端，连接成功后才能进行通信，服务器端不可以主动连接客户端。

(2). 服务器端：java.net.ServerSocket 类实现服务器端。创建 ServerSocket 对象，就是开启一个服务器端，并等待客户端的连接。服务器端程序需要事先启动，并且服务器端不会主动连接客户端，必须要等待客户端的连接。

三.综合案例

1.文件上传案例

客户端读取本地的文件，通过TCP通信程序把文件上传到服务器，服务器再把上传的文件保存到服务器硬盘上。具体实现步骤如下：

①.客户端使用本地的字节输入流，从客户端硬盘上读取要上传的文件

②.客户端使用网络字节输出流，把读取到的文件上传到服务器端

③.服务器端使用网络字节输入流，读取客户端上传的文件

④.服务器端使用本地字节输出流，把读取到的文件保存到服务器硬盘上

⑤.服务器端使用网络字节输出流，给客户端回写一个"上传成功"信息

⑥.客户端使用网络字节输入流，读取服务器端的回写的信息

#.注意：客户端、服务器端对本地硬盘文件的读写，需要使用自己创建的本地字节流对象。客户端与服务器端之间的文件读写，必须使用Socket中提供的网络字节流对象

(1).文件上传案例的阻塞问题

当我们运行上述客户端、服务器端代码时会发现，文件上传成功了，但是两个程序一直处于运行状态没有结束运行。原因如下：



原来是因为客户端调用read方法并没有读取到文件中的结束标记，则永远不会返回-1值，所以形成死循环。同样服务器端在使用read方法接收文件时，也永远读取不到文件结束标记，也永远不会返回-1值，也会形成死循环。导致虽然文件上传成功了，但是两个程序一直在执行死循环，并且两个死循环下面的代码也不会被执行，导致"上传成功"信息回写失败。解决方案：上传完文件，给服务器写一个结束标记



(2).文件上传案例的优化

a. 文件名称写死的问题：如果服务器端保存在硬盘上的文件路径写死，那么当客户端多次上传文件到服务器时，之前上传的文件由于重名会被替换掉，服务器硬盘上只会保留一个文件，建议使用系统时间优化，保证每次上传的文件名称唯一，代码如下：



b. 循环接收的问题：服务端保存一个文件后就关闭了，之后的客户端无法再上传，这是不符合实际的，可以使用使用循环改进，可以不断的接收不同客户端的文件，代码如下：



c.效率问题：服务端在接收大文件时，可能耗费几秒钟的时间，此时不能接收其他用户上传，所以，使用多线程技术优化，代码如下：



2.模拟B\S服务器