网络编程

1. 网络编程入门

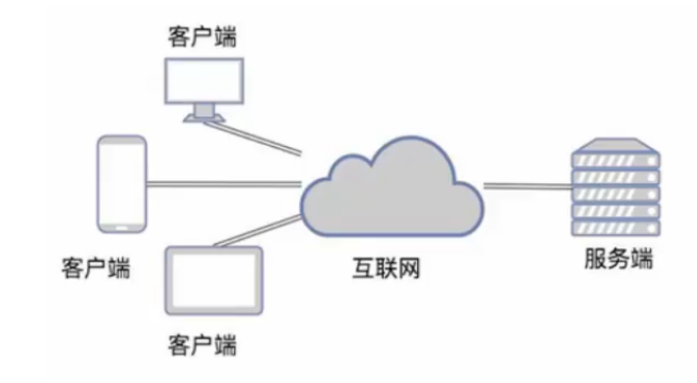
1.1 网络编程概述

> 计算机网络

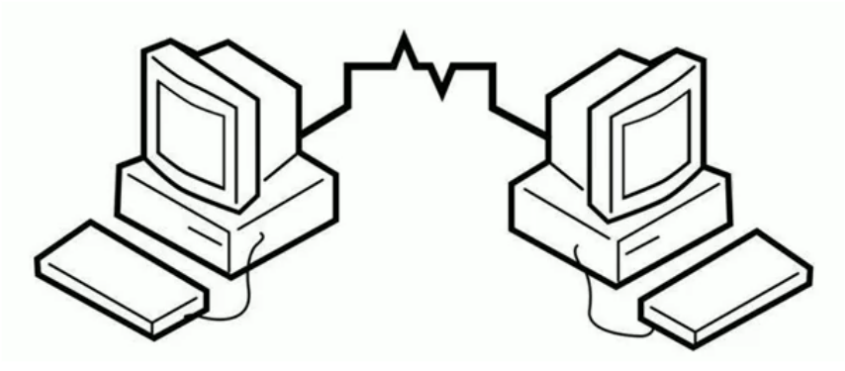
- 是指将地理位置不同具有独立功能的多台计算机及其外部设备，通过通信线路连接起来，在网络操作系统，网络管理软件及网络通信协议的管理和协调下，实现资源共享和信息传递的计算机系统

> 网络编程

- 在网络通信协议下，实现网络互连的不同计算机上运行的程序间可以进行数据交换



1.2 网络编程三要素



> IP地址

- 要想让网络中的计算机能够互相通信，必须为每台计算机指定一个标识号，通过这个标识号来指定要接收数据的计算机和识别发送的计算机，而IP地址就是这个标识号。也就是设备的标识

端口

- 网络的通信，本质上是两个应用程序的通信。每台计算机都有很多的应用程序，那么在网络通信时，如何区分这些应用程序呢？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应用程序了。也就是应用程序的标识

协议

- 通过计算机网络可以使多台计算机实现连接，位于同一个网络中的计算机在进行连接和通信时需要遵守一定的规则，这就好比在道路中行驶的汽车一定要遵守交通规则一样。在计算机网络中，这些连接和通信的规则被称为网络通信协议，它对数据的传输格式、传输速率、传输步骤等做了统规定，通信双方必须同时遵守才能完成数据交换。常见的协议有UDP协议和TCP协议

1.3 IP地址

IP地址：是网络中设备的唯一标识

IP地址分为两大类

- IPv4：是给每个连接在网络上的主机分配一个32bit地址。按照TCP/IP规定，IP地址用二进制来表示，每个IP地址长32bit，也就是4个字节。例如一个采用二进制形式的IP地址是
  “11000001 0101000 00000010 1000010”，
  这么长的地址，处理起来也太费劲了。为了方便使用，IP地址经常被写成十进制的形式，中间使用符号"."分隔不同的字节。于是上面的IP地址可以表示为"192.168.1.66"。IP地址的这种表示法叫做“点分十进制表示法”，这显然比1和0容易记忆得多
- IPv6：由于互联网的蓬勃发展，IP地址的需求量愈来愈大，但是网络地址资源有限，使得IP的分配越发紧张。为了扩大地址空间，通过IPv6重新定义地址空间，采用128位地址长度，每16个字节一组，分成8组十六进制数，这样就解决了网络地址资源数量不够的问题

常用命令：

- ipconfig：查看本机IP
- ping IP地址：检查网络是否连通

特殊IP地址

- 127.0.0.1：是回送地址，可以代表本机地址，一般用来测试使用

1.4 InetAddress 的使用

为了方便我们对IP地址的获取和操作，Java提供了一个类`InetAddress`供我们使用

InetAddress：此类表示Internet协议(IP)地址

方法名	说明
static InetAddress getByName(String host)	确定主机名称的IP地址。主机名称可以是机器名称，也可以是IP地址
String getHostName()	获取此IP地址的主机名
String getHostAddress()	返回文本显示中的IP地址字符串

1.5 端口

端口：设备上应用程序的唯一标识

端口号：用两个字节表示的整数，它的取值范围是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

1.6协议

协议：计算机网络中，连接和通信的规则被称为网络通信协议

UDP协议

- 用户数据报协议(User Datagram Protocol)
- UDP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，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。

由于使用UDP协议消耗资源小，通信效率高，所以通常都会用于音频、视频和普通数据的传输

- 例如：视频会议通常采用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。但是在使用UDP协议传送数据时，由于UDP的面向无连接性，不能保证数据的完整性，因此在传输重要数据时不建议使用UDP协议

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。在TCP连接中必须要明确客户端与服务器端，由客户端向服务端发出连接请求，每次连接的创建都需要经过“三次握手”
- 三次握手：TCP协议中，在发送数据的准备阶段，客户端与服务器之间的三次交互，以保证连接的可靠

  第一次握手：客户端向服务器端发出连接请求，等待服务器确认

  第二次握手：服务器端向客户端回送一个响应，通知客户端收到了连接请求

  第三次握手：客户端再次向服务器端发送确认信息，确认连接

- 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，所以应用十分广泛。例如上传文件、下载文件、 浏览网页等

2. UDP通信程序

2.1 UDP通信原理

UDP协议是一种不可靠的网络协议，它在通信的两端各建立一个Socket对象，但是这两个Socket只是发送，接收数据的对象

因此对于基于UDP协议的通信双方而言，没有所谓的客户端和服务器的概念

Java提供了DatagramSocket类作为基于UDP协议的Socket

2.2 UDP发送数据

> 发送数据的步骤

1. 创建发送端的Socket对象(DatagramSocket)

   `DatagramSocket()`

2. 创建数据, 并把数据打包

   `DatagramPacket(byte[] buf, int length, InetAddress address, int port)`

3. 调用DatagramSocket对象的方法发送数据

   `void send(DatagramPacket p)`

4. 关闭发送端

   `void close()`

2.3 UDP接收数据

> 接收数据的步骤

1. 创建接收端的Socket对象(DatagramSocket)

   `DatagramSocket(int port)`

2. 创建一个数据包，用于接收数据

   `DatagramPacket(byte[] buf, int length)`

3. 调用DatagramSocket对象的方法接收数据

   `void receive(DatagramPacket p)`

4. 解析数据包， 并把数据在控制台示

   `byte[] getData()`

   `int getLength()`

5. 关闭接收端

   `void close()`

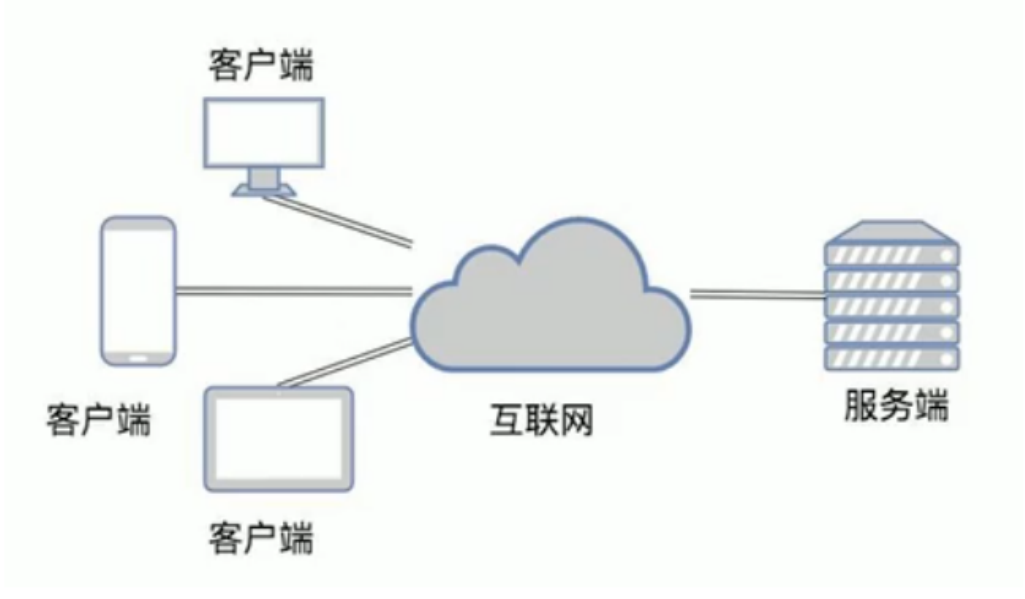
3. TCP通信程序

3.1 TCP通信原理

TCP通信协议是一种可靠的网络协议，它在通信的两端各建立一个Socket对象，从而在通信的两端形成网络虚拟链路，一旦建立了虚拟的网络链路，两端的程序就可以通过虚拟链路进行通信

Java对基于TCP协议的的网络提供了良好的封装，使用Socket对象来代表两端的通信端口，并通过Socket产生IO流来进行网络通信

Java为客户端提供了Socket类，为服务器端提供了ServerSocket类



3.2 TCP发送数据

> 发送数据的步骤

1. 创建客户端的Socket对象(Socket)

   `Socket(String host, int port)`

2. 获取输出流， 写数据

   `OutputStream getOutputStream0`

3. 释放资源

   `void close()`

3.3 TCP接收数据

> 接收数据的步骤

1. 创建服务器端的Socket对象(ServerSocket)

   `ServerSocket(int port)`

2. 监听客户端连接, 返回一个Socket对象

   `Socket accept()`

3. 获取输入流， 读数据,并把数据显示在控制台

   `InputStream getInputStream()`

4. 释放资源

   `void close()`

3.4 几个练习Demo 懒得写了

https://www.bilibili.com/video/BV1gb411F76B?p=355