网络编程概述

2018年8月27日 12:22

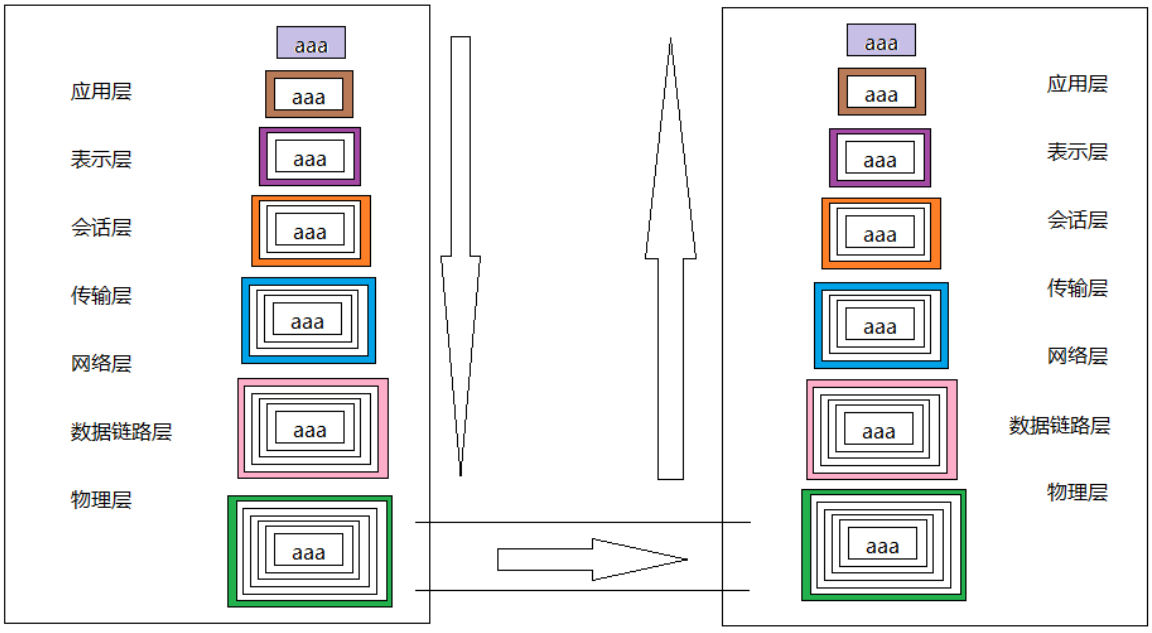
1. 网络编程概述

计算可以通过网络连接，组成计算机网络，计算机之间可以通过网络进行通信，传递信息。很多应用程序也都具有网络通信能力。而java也提供了开发网络程序的编程能力，这就称之为java的网络编程

2. 网络编程的基本概念 - 网络模型

OSI七层网络模型

物理层 数据链路层 网络层 传输层 会话层 表示层 应用层



3. 网络编程的基本概念 - 协议

网络中的计算机想要互相通信，必须遵循相同的沟通方式，需要提前约定，这样提前约定的沟通方式，称之为网络协议，由于网络是分层的，每层之间都有数据要传递，一般的协议都是为某一个层数据的通信来订立的，所以一般来说一个协议通常是归属于某一层的，每一个层也有若干的协议来约定通信规则

协议又可以分为公有协议 和 私有协议

公有协议是由国际化标准组织订立的，全世界的计算机都去遵循

应用层：HTTP HTTPS FTP SMTP POP3

传输层：TPC UDP

网络层：IP协议

私有协议是公司 组织 团队 个人 自己约定的协议 只在遵循该协议的小范围内起作用

4. 网络编程的基本概念 - IP

IP协议目前有两个版本：

IPV4:

0~255 : 0~255 : 0~255 : 0~255

其中如下网段的地址比较特殊，是内网地址：

10.0.0.0 - 10.225.225.255
172.16.0.0 - 172.31.255.255
192.168.0.0 - 192.168.255.255

特殊的ip：

本地回环地址：

127.0.0.1
0.0.0.0

广播地址

xxx.xxx.xxx.255

IPV6:

128位的二进制表示的地址，可以表示2^128个地址

5. 网络编程基本概念 - 端口

每个计算机 除了可以分配到一个IP以外 还会划分出2^16个端口

需要网络通信的软件 可以 来占用一个端口 通过 ip：端口 在指定ip的指定端口上进行通信

虽然ip只有一个，但是端口有很多，所以可以在一个ip上利用不同端口 实现同时进行多个通信的效果

0~65535

其中0~1024的端口是计算机预留的端口 普通程序不可以占用

其他端口应用程序随便占用，先到先得，同一时间一个端口只能有一个程序占用，所以用完后端口会被释放，其他程序才可以再次占用。

6. 主机名 域名 DNS服务器 Hosts文件

主机名

IP地址可以表示网络中的主机 但是ip不易记忆 所以一帮都会选择为当前主机 指定主机名

域名

主机名是可能重复的 为了防止在公网上主机名重复，有了域名的概念，域名需要统一到域名管理组织中注册，从而防止重复

DNS服务器

网络中有 DNS服务器中可以 帮我们将 主机名或域名翻译成对应ip

Hosts文件

可以在本地的Hosts文件中模拟DNS的功能

windows下：

C:\Windows\System32\drivers\etc\Hosts

Linux下：

/etc/hosts

7. 套接字编程 socket编程

为了能够使开发人员开发网络相关的程序，操作系统为开发者提供了网络编程的接口，通过这套接口可以开发基于网络层 和 传输层的 代码 从而实现网络通信。物理层 和 数据链路层 由操作系统负责，不需要开发人员关注，会话层 表示层 应用层 当中的需求 则需要开发人员根据需要自己来实现。

这套操作系统提供的网络编程的接口 称之为socket - 套接字编程

**套接字不是协议 只是一套编程接口 不要搞混

Java网络开发 - IP地址

2018年8月27日 16:12

1. 代表IP地址的类

继承结构：

java.net

类 InetAddress

在java中代表IP地址

重要方法：

static InetAddress	**getLocalHost**() 返回本地主机。
static InetAddress	**getByName**(String host) 在给定主机名的情况下确定主机的 IP 地址。
static InetAddress	**getByAddress**(byte[] addr) 在给定原始 IP 地址的情况下，返回 InetAddress 对象。

String	**getHostName**() 获取此 IP 地址的主机名。
String	**getHostAddress**() 返回 IP 地址字符串（以文本表现形式）。
byte[]	**getAddress**() 返回此 InetAddress 对象的原始 IP 地址。

2. InetSocketAddress

继承结构

java.net

类 SocketAddress

java.net

类 InetSocketAddress

代表socket通信过程中的IP地址

重要方法

构造方法摘要
InetSocketAddress(InetAddress addr, int port) 根据 IP 地址和端口号创建套接字地址。
InetSocketAddress(int port) 创建套接字地址，其中 IP 地址为通配符地址，端口号为指定值。
InetSocketAddress(String hostname, int port) 根据主机名和端口号创建套接字地址。

InetAddress	**getAddress**() 获取 InetAddress。
String	**getHostName**() 获取 hostname。
int	**getPort**() 获取端口号。

Java网络开发 - Socket编程 - UPD编程

2018年8月27日 16:33

1. UDP协议概述

UDP是TCP协议中非常重要和常用的通信协议，可以实现不可靠的网络通信

特点：

不需要创建连接

数据以独立的数据包的形式发送 每个数据包最大64KB

传输过程中 不保证数据一定可以到达 也不保证接受的到的数据包的顺序和发送时一致

速度比较快

~类似于飞鸽传书

在速度要求比较高 可靠性要求比较低 的场景下优先使用

2. java中的udp实现

继承结构

java.net

类 DatagramSocket

代表UDP通信的一个端

重要方法

构造方法摘要	
	DatagramSocket() 构造数据报套接字并将其绑定到本地主机上任何可用的端口。
	DatagramSocket(int port) 创建数据报套接字并将其绑定到本地主机上的指定端口。
	DatagramSocket(SocketAddress bindaddr) 创建数据报套接字，将其绑定到指定的本地套接字地址。

void	**send**(DatagramPacket p) 从此套接字发送数据报包。
void	**receive**(DatagramPacket p) 从此套接字接收数据报包。

继承结构

java.net

类 DatagramPacket

重要方法

构造方法摘要
DatagramPacket(byte[] buf, int length) 构造 DatagramPacket，用来接收长度为 length 的数据包。

byte[]	**getData**() 返回数据缓冲区。
int	**getLength**() 返回将要发送或接收到的数据的长度。
SocketAddress	**getSocketAddress**() 获取要将此包发送到的或发出此数据报的远程主机的 SocketAddress（通常为 IP 地址 + 端口号）。
void	**setSocketAddress**(SocketAddress address) 设置要将此数据报发往的远程主机的 SocketAddress（通常为 IP 地址 + 端口号）。
void	**setData**(byte[] buf) 为此包设置数据缓冲区。
void	**close**() 关闭此数据报套接字。

案例：实现UDP聊天案例

```
package cn.tedu.net.udp.chat;

import java.net.DatagramPacket;
import java.net.DatagramSocket;
import java.net.InetSocketAddress;
import java.util.Scanner;

public class ChatClient {
    public static void main(String[] args) {
        new Thread(new Sender()).start();
```

```
            new Thread(new Receiver()).start();
        }
    }

    /**
     * 聊天消息接受者
     */
    class Receiver implements Runnable{
        @Override
        public void run() {
            DatagramSocket ds = null;
            try {
                //1.创建接收端
                ds = new DatagramSocket(44444);
                while(true){
                    //2.接受数据包
                    byte [] data = new byte[1024];
                    DatagramPacket dp = new DatagramPacket(data, data.length);
                    ds.receive(dp);
                    //3.获取接收到的信息
                    String msg = new String(data,0,dp.getLength());
                    String ip = dp.getAddress().getHostAddress();
                    int port = dp.getPort();
                    System.err.println("==收到来自["+ip+":"+port+"]的消息，消息内容
                    为： “"+msg+"”========");
                }
            } catch (Exception e) {
                e.printStackTrace();
                throw new RuntimeException(e);
            } finally {
                //4.关闭接收端
                if(ds != null){
                    ds.close();
                }
            }
        }
    }

    /**
     * 聊天消息发送者
     */
    class Sender implements Runnable{
        @Override
        public void run() {
            Scanner scan = null;
            DatagramSocket ds = null;
            try {
```

```
            //1.创建发送端
            ds = new DatagramSocket();
            //2.创建控制台扫描器
            scan = new Scanner(System.in);
            while(true){
                //3.读取控制台 消息格式 [ip#端口#消息]
                System.out.println("===请输入要发送的消息，格式为[ip#端口#消息]：
                ================");
                String line = scan.nextLine();
                String attrs [] = line.split("#");
                String ip = attrs[0];
                int port = Integer.parseInt(attrs[1]);
                String msg = attrs[2];
                //4.发送数据
                DatagramPacket dp = new DatagramPacket(msg.getBytes(),
                msg.getBytes().length);
                dp.setSocketAddress(new InetSocketAddress(ip, port));
                ds.send(dp);
            }
        } catch (Exception e) {
            e.printStackTrace();
            throw new RuntimeException(e);
        } finally {
            if(ds!=null){
                ds.close();
            }
            if(scan!=null){
                scan.close();
            }
        }
    }
}
```

Java网络开发 - Socket编程 - TCP编程

2018年8月27日 16:38

1. TCP协议概述

TCP是TCP协议中非常重要和常用的通信协议，可以实现可靠的网络通信

特点：

需要创建连接 需要三次握手

底层建立的连接流 数据包以流的方式传递 没有传输数据量大小的限制

传输过程中 可以保证数据一定不会丢 也不会多 也可以保证 顺序的一致

速度比较慢

在可靠性要求比较高 速度要求比较低 的场景下优先使用