# 实验八：端口扫描工具的优化

——1801090006 郭盈盈

## 一、实验题目

请结合本次实验的资料，进一步优化上次实验中的端口扫描工具，使其可以实现**域名转换 IP 地址**、**主机存活测试**功能。

本次实验**编程语言不限，必须要有界面**，不要提交整个工程文件。

提交时请上传**实验报告**、**源代码**（.cpp、.py 或.java 文件）。

## 二、相关知识

**域名转换 IP 地址**

一、域名

INTERNET 上用来寻找网站所用的名字，是 INTERNET 上的重要标识，在全世界，没有重复的域名，域名是**唯一**的。

二、IP 地址

给每个连接在 Internet 上的主机分配的一个 32bit 地址。

IP 地址是 IP 协议提供的一种统一的地址格式，它为互联网上的每一个网络和每一台主机分配一个逻辑地址，以此来屏蔽物理地址的差异。

三、关系

当用户访问某台主机时，必须先将域名“翻译”成对应的 IP 地址，然后才能通过 IP 地址与该主机联系，并且以后的所有通信都将用到 IP 地址这一“翻译”的过程，这就是所谓**域名解析**；反过来由 IP 地址得出域名地址的过程称为域名反向解析。

四、对应关系有两种类型

1、一对一

Internet 上 IP 地址是唯一的，一个 IP 地址对应着唯一的一台主机。给定一个域名地址能找到一个唯一对应的 IP 地址。

2、一对多

一台计算机提供多个服务，既作 www 服务器又作邮件服务器，IP 地址还是唯一，但可根据计算机提供的多个服务给予不同域名，一个 IP 地址对应多个域名。

五、域名解析实现

```
//将域名转换为IP地址
private void tranlaste() {
    InetAddress address = null;
    try {
        address = InetAddress.getByName(tfAddress.getText());
    }catch(UnknownHostException e) {
        e.printStackTrace();
        text.setText("转换失败！请输入正确的域名！");
    }
    tfIP.setText(address.getHostAddress());
}
```

在上述代码中，所需知识：

**InetAddress** 是 Java 对 IP 地址的封装，在 java.net 中有许多类都使用到了 InetAddress，包括 ServerSocket，Socket，DatagramSocket 等。InetAddress 类没有构造方法，不能用 new 来构造实例，通常用它提供的静态方法获取：

**（1）public static InetAddress getByName(String host)//host 可以是一个机器名，也可以是 ip 地址或 DNS 域名。**

（2）public static InetAddress getLocalHost()//获得本地主机这两个方法会产生 UnknownHostException 异常，应在程序中捕获。

另外，利用 InetAddress 对象获得 IP 地址或机器名的方法：

（1）public byte[] getAddress()//获得本对象的 IP 地址，存放在字节数组中。

（2）public String getHostAddress()//获得本对象的 IP 地址。

（3）public String getHostName()//获得本对象的机器名。

**主机存活测试（ICMP Scan）**

connect()函数默认是阻塞，扫描端口需要一定的时间，尤其是不开放的端口，对于不在线的主机这种扫描就变得没有意义了，测试主机是否在线节省了总体检测时间。主机存活测试即判断主机是否在线，即测试**是否可以 ping 通，调用 ping 即可**。

ping 的实现详见实验六（c++实现），此处用 Java 的方式实现。

**设定延时扫描**

一、必要性

利用 connect 进行端口扫描时，当连接一个开放的端口时需要的时间约为 0.05s，而不开放的端口就大约会消耗 21s，也就是说扫描一个不开放的端口时几乎会卡死 21 秒。给 connect()函数设定一个超时时间可以避免默认的多次重发，从而节约时间，提升扫描速度。

二、实现及必备知识

```
try {
    Socket socket = new Socket();        //建立连接
    socket.connect(new InetSocketAddress(host, currPort), Integer.parseInt(tfTime.getText()));
    socket.close();        //关闭连接
    return true;
}catch(IOException e) {
    return false;
}
```

以上代码中，，所需知识：

1. connect 函数通常用于**客户端建立 tcp 连接**,定义函数：

public void connect(SocketAddress endpoint,int timeout)
throws IOException

参数：

endpoint - SocketAddress

timeout - 以 timeout 为单位的超时值。

**2. InetSocketAddress 类，**此类用于实现 IP 套接字地址(IP 地址+端口号)，用于 socket 通信。

构造方法

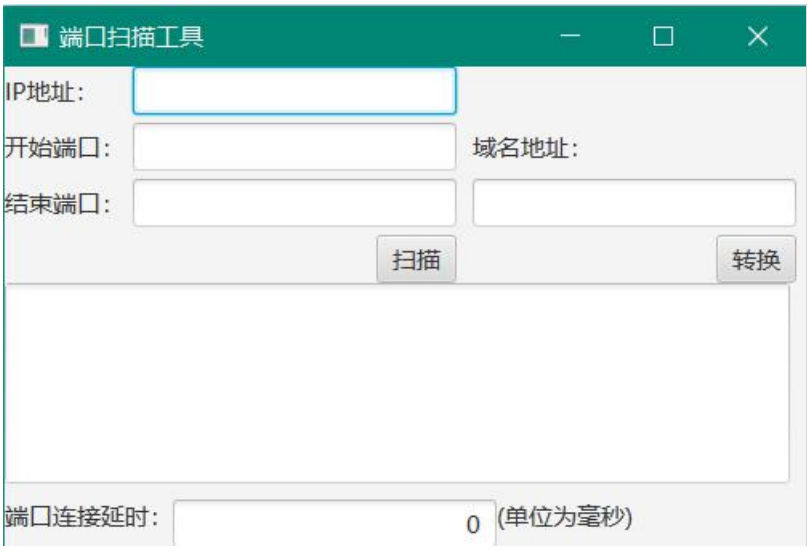
（1）InetSocketAddress(InetAddress addr, int port)//从 IP 地址和端口号创建套接字地址。

（2）InetSocketAddress(int port)//创建一个套接字地址，其中 IP 地址为通配符地址，端口号为指定值。

**（3）InetSocketAddress(String hostname, int port)//从主机名和端口号创建套接字地址。**

3. **getText()**从文本框中获取延迟时间（默认为 0），但**返回的值类型为 String** 所以需要 **Integer.parseInt()**将其转换为整型进行运算。

**基本界面**




大致分为 IP 地址与域名的转换以及端口的输入操作区域、文本显示的区域、端口延时设定区域 3 个部分。

1. IP 地址与域名的转换以及端口的输入操作区域

主要元件有：IP 地址的文本框（不可编辑）tfIP、开始端口的文本框 tfBegin、

结束端口的文本框 tfEnd、域名的文本框 tfAddress、扫描按钮 btScan(点击触发事件调用函数 connect)、转换按钮 btTranslate(点击触发事件调用函数 translate)以及对应的文字标签。

属性编辑

```
//关于端口与域名、IP间转换的面板
GridPane paneForIP = new GridPane();
paneForIP.setHgap(5);
paneForIP.setVgap(5);
paneForIP.add(new Label("IP地址："), 0, 0);
paneForIP.add(tfIP, 1, 0);
paneForIP.add(new Label("开始端口："), 0, 1);
paneForIP.add(tfBegin, 1, 1);
paneForIP.add(new Label("结束端口："), 0, 2);
paneForIP.add(tfEnd, 1, 2);
paneForIP.add(btScan, 1, 3);
paneForIP.add(new Label("域名地址："), 3, 1);
paneForIP.add(tfAddress, 3, 2);
paneForIP.add(btTranlaste, 3, 3);

paneForIP.setAlignment(Pos.CENTER_LEFT);//设置面板中的内容靠左
tfIP.setEditable(false);//设置IP框不可编辑
tfBegin.setAlignment(Pos.BOTTOM_RIGHT);//设置文本框中内容靠右
tfEnd.setAlignment(Pos.BOTTOM_RIGHT);
tfAddress.setAlignment(Pos.BOTTOM_RIGHT);
paneForIP.setHalignment(btScan, HPos.RIGHT);//设置该元件在水平方向靠右
paneForIP.setHalignment(btTranlaste, HPos.RIGHT);

btScan.setOnAction(e -> connect());//点击触发事件，调用connect
btTranlaste.setOnAction(e -> tranlaste());//点击触发事件，调用translate
```

2. 文本显示的区域

主要元件有：可多行显示的文本区域 text(不可编辑，通过触发事件改变显示的文本)

属性编辑

```
//关于信息显示的面板
text.setWrapText(true);
text.setEditable(false);//设置文本区域不可编辑
text.setPrefSize(490, 125);//设置文本区域宽为490，高为125
Pane paneForText = new Pane();
paneForText.getChildren().add(text);
```

3. 端口延时设定区域 3 个部分

主要元件有：设定延时扫描时间的文本框 tfTime(默认显示为 0，当点击扫描时会触发事件，读取该文本框中的信息，但注意类型转换，String 不能进行算术运算)以及对应的文字标签。

属性编辑

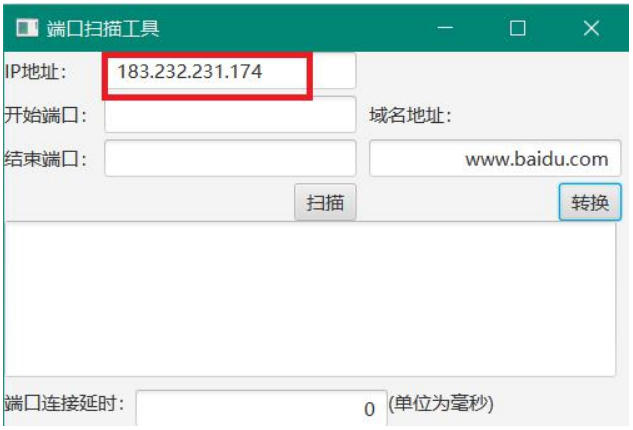
```
//关于延时扫描设定的面板
tfTime.setAlignment(Pos.BOTTOM_RIGHT);

HBox hbTime = new HBox();
hbTime.getChildren().add(new Label("端口连接延时： "));
hbTime.getChildren().add(tfTime);
hbTime.getChildren().add(new Label("(单位为毫秒)"));
```

## 三、实验步骤与结果分析

1. 当没有输入域名直接转换 ip 时，信息显示区域显示“主机不存在或者有防火墙！”。


2. 输入正确域名后点击转换按钮，IP 地址框内会显示对应的点分十进制的 IP 地址。



3.输入开始扫描端口和结束扫描端口，延时设定框内填入 200(ms)，点击扫描。


与上次实验相比，明显扫描速度明显大大减少，上次大约为 42s，而这次同样的端口号下，扫描时间为约为 0.4s。

## 四、实验收获与总结

在实现域名转换 IP 地址和主机存活测试时，java 与 c++有简易明显的互通之处，较好实现，而当实现设定延时扫描时，c++与 java 不用的是，c++需要用非阻塞通信来判断是否超时，而 java 有封装好的函数，只用 connect()直接调用即可，当时在研究 java 实现非阻塞通信时花了很久时间。