

实验七：简易端口扫描工具的实现

——1801090006 郭盈盈

2020.04.14

一、实验题目

请依据实验文档中的内容，编写一个简易的端口扫描工具，并测试该工具是否有效。

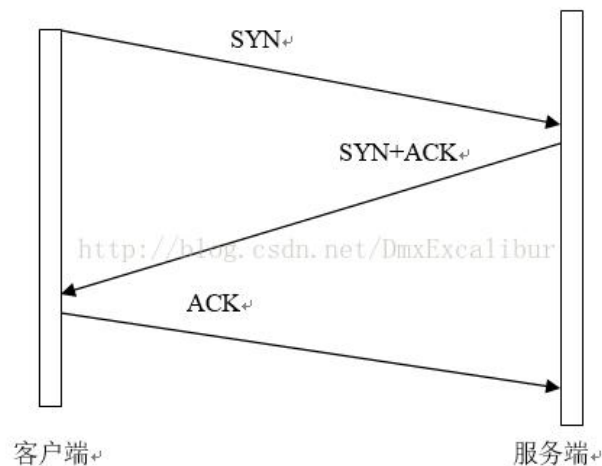
本次实验编程语言不限，必须要有界面，不要提交整个工程文件。

提交时请上传实验报告、源代码（.cpp、.py 或 .java 文件等）。

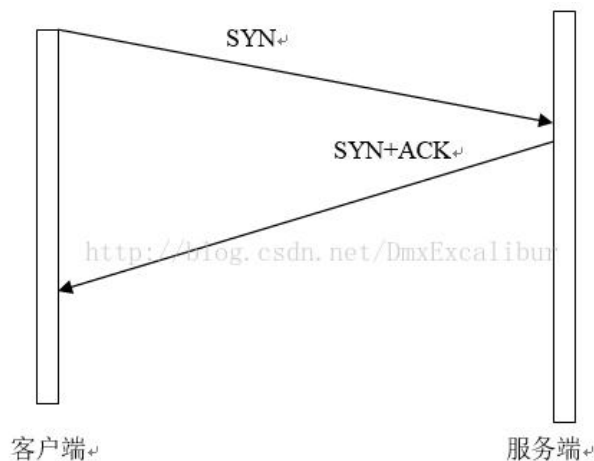
二、相关知识

我们知道完成一次 TCP 连接需要完成三次握手才能建立。端口扫描正是利用了这个原理，通过假冒正常的连接过程，依次向目标主机的各个端口发送连接请求，并根据目标主机的应答情况判断目标主机端口的开放情况，从而分析并对一些重要端口实施攻击。

端口扫描的方式有两种，一种称为完整扫描，一次连接过程如下图所示：



另一种扫描方式称为半开扫描，出于欺骗的目的，半开扫描在收到服务端的应答信号（SYN+ACK）后，不再发送响应信号（ACK）。一次连接过程如下图所示：



算法分析

利用 `java.net.Socket` 类建立 socket 连接,如果无法与指定的 IP 和端口建立连接,将会抛出 `IOException`。我们用 try-catch 对这个 `IOException` 异常进行捕获,以判断是否成功与指定的 IP 端口建立连接。如果成功建立了连接,说明指定 IP 的指定端口已经开放;如果程序抛出了一个 `IOException` 异常被我们捕获,则说明指定的 IP 没有开放指定的端口。扫描指定端口段则是利用循环不断与服务器的指定端口进行连接,供我们判断是否开放。

三、实验步骤与结果分析

代码

扫描连接实现

在得到端口和建立 socket 之前一定要判断端口的合法性,因为端口的范围是在 1~65535,如果我们去建立范围外端口的连接就是没必要的,而且是不可行的。既然是起始端口和终止端口,那么就要有个大小顺序问题,也就是判断它们的大小。

```
104         startPort = Integer.parseInt(tfBegin.getText()); //获得起始端口号
105         endPort = Integer.parseInt(tfEnd.getText()); //获得终止端口号
106         if(startPort<1||startPort>65535||endPort<1||endPort>65535){
107             //检查端口是否在合法范围1~65535
108             System.out.printf("端口范围必须在1~65535以内!");
109             return null;
110         }else if(startPort>endPort){ //比较起始端口和终止端口
111             System.out.println("端口输入有误! 起始端口必须小于终止端口");
112             return null;
113         }
```

建立与服务器指定端口的连接,就要用到 `java.net.Socket` 类了,首先我们来看看它的构造方法。

`Socket()`: 通过系统默认类型的 `SocketImpl` 创建未连接套接字。

`Socket(InetAddress address, int port)`: 创建一个流套接字,并将其连接到指定 IP 地址的指定端口号。

`Socket(InetAddress address, int port, InetAddress localAddr, int localPort)`: 创建一个套接字,并将其连接到指定远程端口上的指定远程地址。

`Socket(Proxy proxy)`: 根据不管其他设置如何都应使用的指定代理类型(如果有),创建一个未连接的套接字。

`Socket(SocketImpl impl)`: 创建带有用户指定的 `SocketImpl` 的未连接 `Socket`。

`Socket(String host, int port)`: 创建一个流套接字,并将其连接到指定主机上的指定端口号。

`Socket(String host, int port, InetAddress localAddr, int localPort)`: 创建

一个套接字并将其连接到指定远程主机上的指定远程端口。

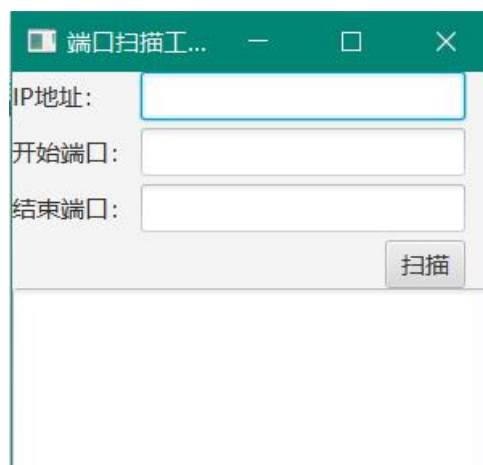
可以看到我们有很多种构造方法，目前只需要关心第二种构造方法 `Socket(InetAddress address, int port)` 即可，因为我们并不需要与服务器运行在端口的服务进行交互，所以我们只要建立连接，然后关闭连接即可。即：“`Socket s = new Socket(address, port);`”。

下面就是我们的核心算法了。循环指定端口段的所有端口，对所有端口建立连接。连接成功后我们就算完成了当前循环的任务，然后调用 `close()` 方法关闭连接。如果成功建立连接，就不会执行到 `catch` 里面，而是执行到下面的语句；如果不能连接上去就会抛出一个异常被我们捕获，程序就会运行到 `catch` 里面，执行 `catch` 里面的语句；最后继续下一个循环。代码如下：

```
81 private boolean isConnect(int currPort) {
82     String host = tfIP.getText();
83     try{
84         InetAddress address = InetAddress.getByName(host);
85         //转换类型
86     }catch(UnknownHostException e){
87         System.out.println("无法找到 "+ host);
88         return false;
89     }
90     try {
91         Socket socket = new Socket(host,currPort);           //建立连接
92         socket.close();                                       //关闭连接
93         return true;
94     }catch(IOException e) {}
95     return false;
96 }
97 }
```

还有计算耗时的算法，就是计算扫描实现的代码的运行时间，即在代码两端加入 `System.currentTimeMillis()`；获取开始运行的时间和结束运行的时间，最后算出差值即是耗时。

窗口界面实现



主要控件为 3 个可输入文本框, 1 个触发点击事件的按钮和 1 个显示结果的文本区域, 代码如下

```
28     private TextField tfIP = new TextField();
29     private TextField tfBegin = new TextField();
30     private TextField tfEnd = new TextField();
31     private Button btScan = new Button("扫描");
32     private TextArea text = new TextArea();
33     @Override
34     public void start(Stage primaryStage) throws Exception {
35         // TODO Auto-generated method stub
36         //Create UI
37         GridPane paneForIP = new GridPane();
38         paneForIP.setHgap(5);
39         paneForIP.setVgap(5);
40         paneForIP.add(new Label("IP地址: "), 0, 0);
41         paneForIP.add(tfIP, 1, 0);
42         paneForIP.add(new Label("开始端口: "), 0, 1);
43         paneForIP.add(tfBegin, 1, 1);
44         paneForIP.add(new Label("结束端口: "), 0, 2);
45         paneForIP.add(tfEnd, 1, 2);
46         paneForIP.add(btScan, 1, 3);
47
48         //Set properties for UI
49         paneForIP.setAlignment(Pos.CENTER_LEFT);
50         tfIP.setAlignment(Pos.BOTTOM_RIGHT);
51         tfBegin.setAlignment(Pos.BOTTOM_RIGHT);
52         tfEnd.setAlignment(Pos.BOTTOM_RIGHT);
53         paneForIP.setHalignment(btScan, HPos.RIGHT);
54
55         btScan.setOnAction(e -> connect());
56
57         text.setWrapText(true);
58         text.setEditable(false);
59         Pane paneForText = new Pane();
60         paneForText.getChildren().add(text);
61
62         BorderPane pane = new BorderPane();
63         pane.setTop(paneForIP);
64         pane.setCenter(paneForText);
65
66         Scene scene = new Scene(pane, 300, 250);
67         primaryStage.setTitle("端口扫描工具");
68         primaryStage.setScene(scene);
69         primaryStage.show();
70     }
```

操作及结果

在命令行窗口中输入 ping+百度域名获得 IP 地址

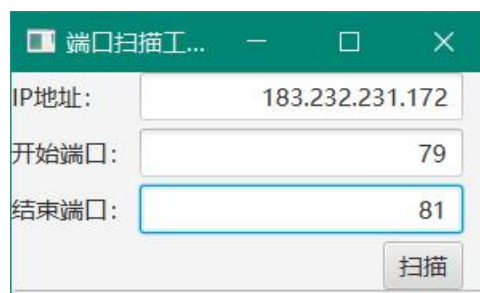
```
C:\WINDOWS\system32\cmd.exe
Microsoft Windows [版本 10.0.18362.778]
(c) 2019 Microsoft Corporation。保留所有权利。

C:\Users\Administrator>ping www.baidu.com

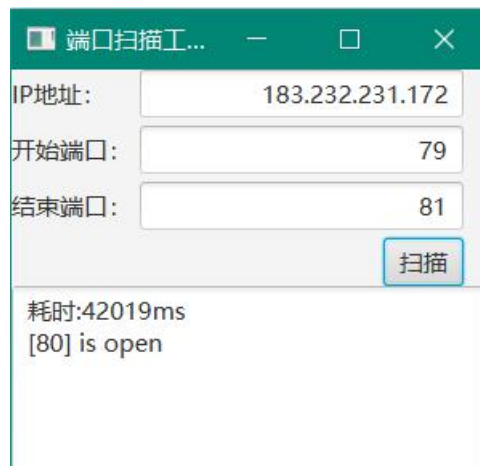
正在 Ping www.baidu.com [183.232.231.172] 具有 32 字节的数据:
来自 183.232.231.172 的回复: 字节=32 时间=11ms TTL=57
来自 183.232.231.172 的回复: 字节=32 时间=16ms TTL=57
来自 183.232.231.172 的回复: 字节=32 时间=11ms TTL=57
来自 183.232.231.172 的回复: 字节=32 时间=11ms TTL=57

183.232.231.172 的 Ping 统计信息:
    数据包: 已发送 = 4, 已接收 = 4, 丢失 = 0 (0% 丢失),
    往返行程的估计时间(以毫秒为单位):
        最短 = 11ms, 最长 = 16ms, 平均 = 12ms
```

在端口扫描工具中输入 IP 地址和开始扫描的端口号、结束扫描的端口号



点击扫描



可知 79 号端口到 81 号端口的扫描时间约为 42 秒，平均一个端口耗时 12 秒，这个速度太慢。

四、实验收获与总结

第一次在 java 中使用 `socket()`，通过编译运行了解了 `SeverSocket`, `Socket`, `NioSocket` 的用法。还有了解了端口扫描工具的工作原理，只是目前所实现的扫描时间过长，希望以后可以学到有别的办法解决。