



浙江工业大学

# 计网课程设计实验报告

题目： 计算机网络课程设计

组长姓名 张坤

组员姓名 缪健

提交日期 2021 年 1 月 14 日

## 目录

|   |    |
|---|----|
| 一、 任务一 .....  | 3  |
| 二、 任务二 .....  | 7  |
| 1. 在交换机上实现 VLAN 配置 .....                                      | 7  |
| 2. 基于 Console 控制台登录配置路由器，学习路由器配置相关命令 .....                    | 10 |
| A. 基本配置 .....   | 10 |
| B. 接口配置 .....   | 11 |
| 3. 基于 packet tracer 构建网络环境，分别进行静态路由配置和基于 RP 的动态路由配置。<br>..... | 11 |
| 1) 静态路由配置 .....   | 11 |
| 2) 动态路由配置 .....   | 14 |
| 三、 任务三 .....  | 18 |
| 1. 运行环境 .....   | 18 |
| 2. 设计目标 .....   | 18 |
| 3. 核心思想 .....   | 18 |
| 4. 流程图 .....  | 18 |
| 5. 关键问题,核心代码 .....  | 19 |
| 6. 运行界面 .....   | 20 |
| 7. 抓包分析 .....   | 21 |
| 1) 第一次抓包 .....  | 22 |
| 2) 第二次: .....   | 22 |
| 3) 第三次 .....  | 22 |
| 至此三次握手完毕 .....  | 22 |
| 4) 第四次 .....  | 22 |
| 5) 第五次 .....  | 23 |
| 下面是四次挥手 .....   | 23 |
| 7) 第一次挥手: .....   | 23 |
| 8) 第二次 .....  | 23 |
| 9) 第三次 .....  | 23 |
| 10) 第四次 .....   | 23 |
| 8. 实验遇到的问题: .....   | 23 |
| 1 服务器只能连接一个客户端: .....   | 23 |
| 2 服务器无法返回数据给客户端 .....   | 23 |
| 9. 实验总结 .....   | 23 |

## 一、 任务一

1. 常用网络命令 ipconfig, ping, netstat, tracert, arp, telnet 的功能
  - 1) ipconfig: 用来显示主机内 IP 协议的信息, 包括以太网, 以太网配置器, PPP 适配器等信息

```
以太网适配器 以太网:

    连接特定的 DNS 后缀 . . . . . :
    本地链接 IPv6 地址. . . . . : fe80::5dd9:546:72bf:805%10
    IPv4 地址 . . . . . : 10.136.5.242
    子网掩码 . . . . . : 255.255.240.0
    默认网关. . . . . : 10.136.0.1

PPP 适配器 My VPN Link:

    连接特定的 DNS 后缀 . . . . . :
    IPv4 地址 . . . . . : 10.200.42.33
    子网掩码 . . . . . : 255.255.255.255
    默认网关. . . . . : 0.0.0.0
```

如上,可以从中查看本机的 IP 地址,子网掩码和默认网关等等

- 2) ping: 用来检测两台主机相互通讯是否成功,需要多少时间.以 ping [www.baidu.com](http://www.baidu.com) 为例

```
C:\Users\zk>ping www.baidu.com

正在 Ping www.a.shifen.com [36.152.44.95] 具有 32 字节的数据:
来自 36.152.44.95 的回复: 字节=32 时间=28ms TTL=56
来自 36.152.44.95 的回复: 字节=32 时间=28ms TTL=56
来自 36.152.44.95 的回复: 字节=32 时间=29ms TTL=56
来自 36.152.44.95 的回复: 字节=32 时间=29ms TTL=56

36.152.44.95 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 = 28ms, 最长 = 29ms, 平均 = 28ms
```

显然可以 ping 通 [www.baidu.com](http://www.baidu.com),一共发送四个数据包,用时约 28ms

- 3) netstat: 帮助我们了解网络的整体使用状况,可以显示当前正在活动的网络连接详细信息

```
C:\Users\zk>netstat

活动连接

 协议 本地地址           外部地址           状态
TCP    10.200.42.33:6072   17.57.145.9:5223   ESTABLISHED
TCP    10.200.42.33:6073   40.119.211.203:https ESTABLISHED
TCP    10.200.42.33:6118   40.90.189.152:https ESTABLISHED
TCP    10.200.42.33:6127   .:https            CLOSE_WAIT
TCP    10.200.42.33:6196   118.178.135.232:https ESTABLISHED
```

以部分信息为例,其显示结果包含:协议名称,本地地址,外部地址,状态.

还有 netstat -e, netstat -n 等命令.如下

(c) 2020 Microsoft Corporation. 保留所有权利。

```
C:\Users\zk>netstat -e
接口统计
```

|        | 接收的       | 发送的       |
|--------|-----------|-----------|
| 字节     | 614284273 | 441312624 |
| 单播数据包  | 7082967   | 13731094  |
| 非单播数据包 | 5754      | 9326      |
| 丢弃     | 2816      | 0         |
| 错误     | 0         | 0         |
| 未知协议   | 0         |           |

```
C:\Users\zk>
```

- 4) tracert: 判定数据包到达目的主机所经过的路径、显示数据包经过的中继节点清单和到达时间.以 tracert [www.baidu.com](http://www.baidu.com) 为例

```
C:\Users\zk>tracert www.baidu.com
```

通过最多 30 个跃点跟踪  
到 www.a.shifen.com [36.152.44.96] 的路由:

|   |       |       |       |                |
|---|-------|-------|-------|----------------|
| 1 | 2 ms  | 2 ms  | 2 ms  | 10.200.0.1     |
| 2 | 15 ms | 6 ms  | 4 ms  | 111.0.79.21    |
| 3 | 4 ms  | 3 ms  | 3 ms  | 221.183.76.205 |
| 4 | 29 ms | 28 ms | 31 ms | 221.183.42.61  |
| 5 | 36 ms | 35 ms | 35 ms | 221.183.59.54  |
| 6 | *     | *     | *     | 请求超时。          |
| 7 | 32 ms | 31 ms | 32 ms | 182.61.216.72  |
| 8 | 28 ms | 28 ms | 28 ms | 36.152.44.96   |

跟踪完成。

其连接过程如上,一共跟踪八个跃点

Tracert 同样也有后续命令

- i. -d:指定不将 IP 地址解析到主机名称
- ii. -h maximum\_hops:指定跃点数以跟踪 target\_name 的主机的路由
- iii. -j host-list:指定 tracert 使用程序数据报所采用的路径中的路由器接口列表
- iv. -w timeout:等待 timeout 为每次回复所指定的毫秒数

- 5) arp: 查看本地计算机或另一台计算机的 ARP 高速缓存中的当前内容.  
仅输入 arp 命令会提示如下

```

C:\Users\zk>arp
显示和修改地址解析协议(ARP)使用的“IP-网物理”地址转换表。

ARP -a inet_addr eth_addr [if_addr]
ARP -d inet_addr [if_addr]
ARP -s (inet_addr) [(eth_addr) [if_addr]] [-w]

-a          通过询问当前协议数据，显示当前 ARP 项。
             如果指定 inet_addr，则只显示指定计算机
             的 IP 地址和物理地址。如果不止一个网络
             接口使用 ARP，则显示每个 ARP 表的项。
-g          与 -a 相同。
-v          在详细模式下显示当前 ARP 项。所有主地址
             和物理地址上的列都可用显示。
inet_addr   指定 Internet 地址。
-if_addr    指定 if_addr 指定的网络接口的 ARP 项。
-d          删除 inet_addr 指定的主机。inet_addr 可
             以是通配符 *，以删除所有主机。
-s          添加主机并且将 Internet 地址 inet_addr
             与物理地址 eth_addr 相关联。物理地址是用
             连字符分隔的 6 个十六进制字节。该项是永久
             的。
eth_addr    物理地址。
if_addr     如果存在，此项指定地址转换表应该建立的接口
             的 Internet 地址。如果不存在，则使用第一
             个可用的接口。

示例：
> arp -a 192.55.85.212 00-00-00-00-00-00... 添加静态项。
> arp -a          ... 显示 ARP 表。

```

那么以 arp -g 为例

```

C:\Users\zk>arp -g

接口: 10.136.5.242 --- 0xa
Internet 地址      物理地址      类型
10.136.0.1         2c-9d-1e-0d-b3-55 动态
10.136.15.255      ff-ff-ff-ff-ff-ff 静态
224.0.0.22         01-00-5e-00-00-16 静态
224.0.0.251        01-00-5e-00-00-fb 静态
224.0.0.252        01-00-5e-00-00-fc 静态
239.255.255.250    01-00-5e-7f-ff-fa 静态
255.255.255.255    ff-ff-ff-ff-ff-ff 静态

```

后文太长不放了,现在可以看出目前本机的 arp 高速缓存如上

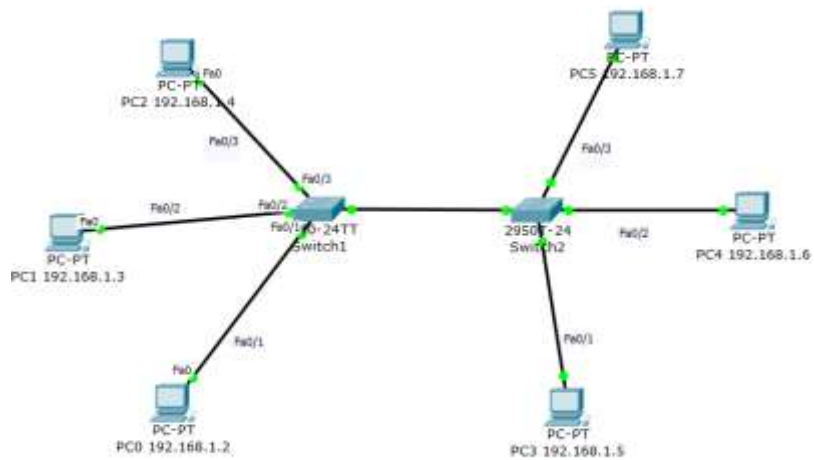
同样 arp 也有后续命令:

- i. -a:通过询问 TCP/IP 显示当前 arp 项。如果制定了 inet\_addr，则只显示指定计算机的 IP 地址和物理地址。
  - ii. -g: 通过询问 TCP/IP 显示当前 arp 项。如果制定了 inet\_addr，则只显示指定计算机的 IP 地址和物理地址。
  - iii. inet\_addr:以带点的十进制标记指定 Internet 地址
  - iv. -d: 删除由 inet\_addr 指定的项
  - v. -s:在 arp 缓存中添加项, 将 Internet 地址 inet\_addr 和物理地址 eth\_addr 相关联。
  - vi. 物理地址由以连字符分隔的 6 个十六进制字节给定。使用带点的十进制标记指定 internet 地址，在超市到期后项自动从缓存删除。
- 6) telnet:远程控制服务器,允许用户登录进入远程主机系统,远程操作等等.其常用命令行如下:
- i. open: 使用 openhostname 可以建立到主机的 Telnet 连接。
  - ii. close: 使用命令 close 命令可以关闭现有的 Telnet 连接。

- iii. display：使用 display 命令可以查看 Telnet 客户端的当前设置。
- iv. send：使用 send 命令可以向 Telnet 服务器发送命令。支持以下命令：
  - v. ao：放弃输出命令。
  - vi. ayt：“Are you there”命令。
  - vii. esc：发送当前的转义字符。
  - viii. ip：中断进程命令。
  - ix. synch：执行 Telnet 同步操作。
  - x. brk：发送信号。

## 二、 任务二

1. 在交换机上实现 VLAN 配置  
拓扑结构如下

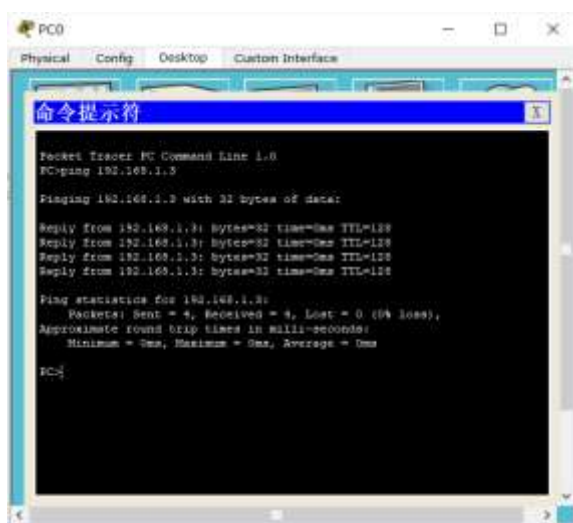


PC 的 IP 配置如下



其余如图不赘述

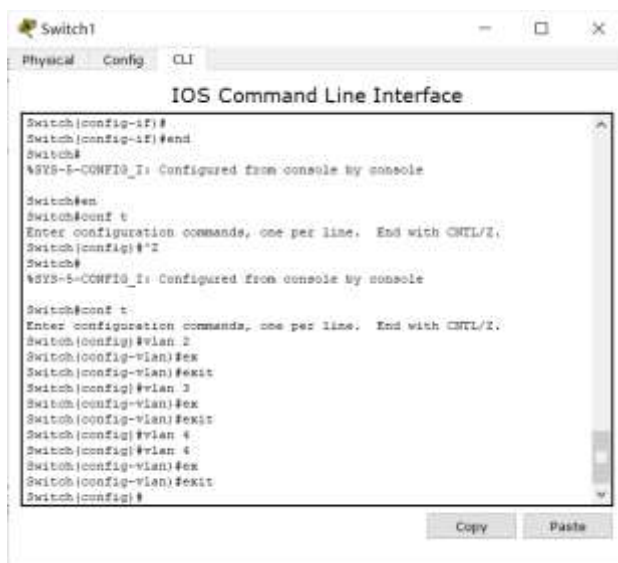
先尝试 ping 命令如下



4 个 PC 均可通讯

对交换机使用命令行如下

先新建三个 VLAN



把 vlan 配置给线路如下:



```
Switch1
Physical Config CLI
IOS Command Line Interface
Switch(config-vlan)#exit
Switch(config)#vlan 4
Switch(config-vlan)#exit
Switch(config)#int fa 0/1
Switch(config-if)#sw
Switch(config-if)#switchport access vlan 2
Switch(config-if)#exit
Switch(config)#int fa 0/2
Switch(config-if)#sw
Switch(config-if)#switchport access vlan 3
Switch(config-if)#exit
Switch(config)#int fa 0/3
Switch(config-if)#sw
Switch(config-if)#switchport access vlan 4
Switch(config-if)#exit
Switch(config)#
```

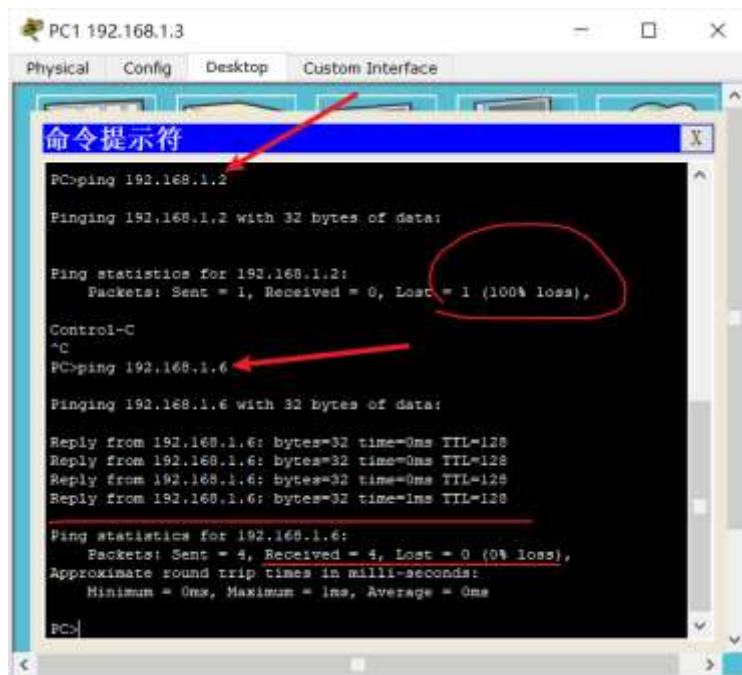
最后配置两交换机之间的线路 0/24

```
Switch1
Physical Config CLI
IOS Command Line Interface
Switch(config-if)#sw
Switch(config-if)#switchport access vlan 3
Switch(config-if)#exit
Switch(config)#int fa 0/3
Switch(config-if)#sw
Switch(config-if)#switchport access vlan 4
Switch(config-if)#exit
Switch(config)#int fa 0/24
Switch(config-if)#sw
Switch(config-if)#switchport mode trunk
Switch(config-if)#
%LINEPROTO-5-UPDOWN: Line protocol on Interface FastEthernet0/24,
changed state to down
%LINEPROTO-5-UPDOWN: Line protocol on Interface FastEthernet0/24,
changed state to up
end
```

Show VLAN 如下

```
Switch1
Physical Config CLI
IOS Command Line Interface
Fa0/15 Fa0/16, Fa0/17, Fa0/18, Fa0/19 Fa0/20, Fa0/21, Fa0/22, Fa0/23 Gig0/1, Gig0/2
2 VLAN0002 active Fa0/1
3 VLAN0003 active Fa0/2
4 VLAN0004 active Fa0/3
1002 fddi-default act/unsup
1003 token-ring-default act/unsup
1004 fddinet-default act/unsup
1005 trnet-default act/unsup
VLAN Type STATE MTU Parent RingNo BridgeNo Stp BrgMode Transl
Trans
1 enet 100001 1500 - - - - - 0
0
2 enet 100002 1500 - - - - - 0
0
3 enet 100003 1500 - - - - - 0
0
--More--
```

对另一台交换机使用同样命令行,不再赘述  
那么测试 ping 命令查看六台 PC 是否可以通讯  
192.168.1.3 ping 192.168.1.2 如下



不可 ping 通,因为主机处于不同的 VLAN 中  
而 192.168.1.3 ping 192.168.1.6 则可以,因为处于相同 VLAN 中

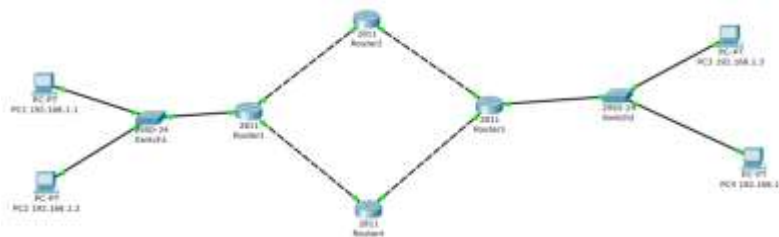
2. 基于 Console 控制台登录配置路由器,学习路由器配置相关命令
  - A. 基本配置
    - i. router>enable /进入特权模式
    - ii. router#conf t /进入全局配置模式
    - iii. router(config)# hostname xxx /设置设备名称
    - iv. router(config)#enable password /设置特权口令
    - v. router(config)#no ip domain lookup /不允许路由器缺省使用 DNS 解析命令
    - vi. router(config)# Service password-encrypt /对所有在路由器上输入的口令进行暗文加密
    - vii. router(config)#line vty 0 4 /进入设置 telnet 服务模式
    - viii. router(config-line)#password xxx /设置 telnet 的密码
    - ix. router(config-line)#login /使能可以登陆
    - x. router(config)#line con 0 /进入控制口的服务模式
    - xi. router(config-line)#password xxx /要设置 console 的密码
    - xii. router(config-line)#login /使能可以登陆

## B. 接口配置

- i. router(config)#int s0 /进入接口配置模式 serial 0 端口配置 (如果是模块化的路由器前面加上槽位编号, 例如 serial0/0 代表这个路由器的 0 槽位上的第一个接口)
  - ii. router(config-if)#ip add xxx.xxx.xxx.xxx xxx.xxx.xxx.xxx /添加 ip 地址和掩码
  - iii. router(config-if)#enca hdlc/ppp 捆绑链路协议 hdlc 或者 ppp 思科缺省串口封装的链路层协议是 HDLC 所以在 show run 配置的时候接口上的配置没有, 如果要封装为别的链路层协议例如 PPP/FR/X25 就是看到接口下的 enca ppp 或者 enca fr
  - iv. router(config)#int loopback /建立环回口(逻辑接口)模拟不同的本机网段
  - v. router(config-if)#ip add xxx.xxx.xxx.xxx xxx.xxx.xxx.xxx /添加 ip 地址和掩码给环回口在物理接口上配置了 ip 地址后用 no shut 启用这个物理接口反之可以用 shutdown 管理性的关闭接口
3. 基于 packet tracer 构建网络环境, 分别进行静态路由配置和基于 RP 的动态路由配置。要求: 静态路由配置拓扑中至少 4 个路由器; RP 动态路由配置中源站和目的站之间设置两条跳数不同的路径, 通过 RIP 配置后查看选择的是哪条路径要求写出相应的步骤, 给出截图和文字说明。

### 1) 静态路由配置

拓扑结构如下



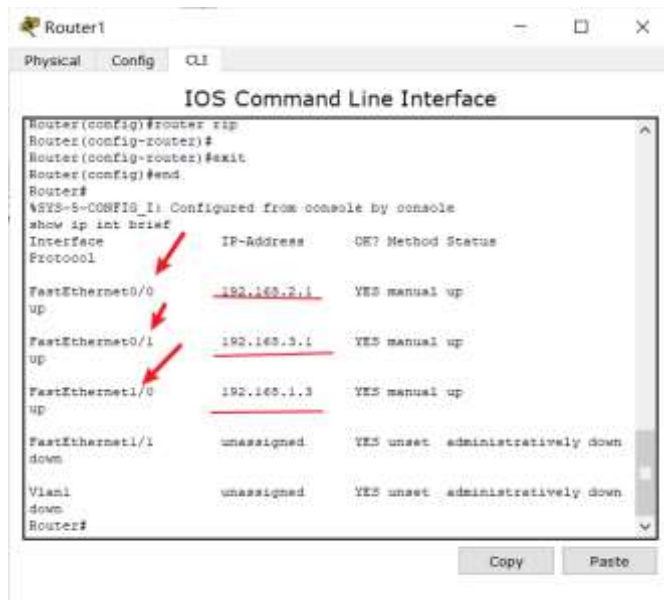
首先设置 PC 的 IP 地址如下,以 PC1 为例:



接着设置四个路由器的接口如下,以 Router1 为例



需要对四个路由器设置,用命令 `show ip int brief` 查看如下



接着设置静态路由

```

Router(config)#ip route 192.168.6.0 255.255.255.0 192.168.3.2
Router(config)#ip route 192.168.7.0 255.255.255.0 192.168.2.2
Router(config)#ip route 192.168.8.0 255.255.255.0 192.168.2.2

```

用命令行 `do show ip route` 查看

```
Router1
Physical Config CLI
IOS Command Line Interface
Enter configuration commands, one per line. End with CTRL/Z.
Router(config)#do
Router(config)#do show ip route
show ip route
% Invalid input detected at '' marker.
Router(config)#do show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, X - EGP
       * - candidate default, U - per-user static route, o - ODR
       P - periodic downloaded static route
Gateway of last resort is not set.
C    192.168.1.0/24 is directly connected, FastEthernet1/0
C    192.168.2.0/24 is directly connected, FastEthernet2/0
C    192.168.3.0/24 is directly connected, FastEthernet3/0
S    192.168.4.0/24 [1/0] via 192.168.3.2
S    192.168.5.0/24 [1/0] via 192.168.3.2
```

对其余三台路由表同样设置

接着尝试 ping 命令

一开始是无法 ping 通的,但是第二次就可以了

此处忘记截图了

PC1 ping PC3

```
PC1 192.168.1.1
Physical Config Desktop Custom Interface
命令提示符
Packet Tracer PC Command Line 1.0
PC>ping 192.168.1.3
Invalid Command.

PC>ping 192.168.1.3
Pinging 192.168.1.3 with 32 bytes of data:

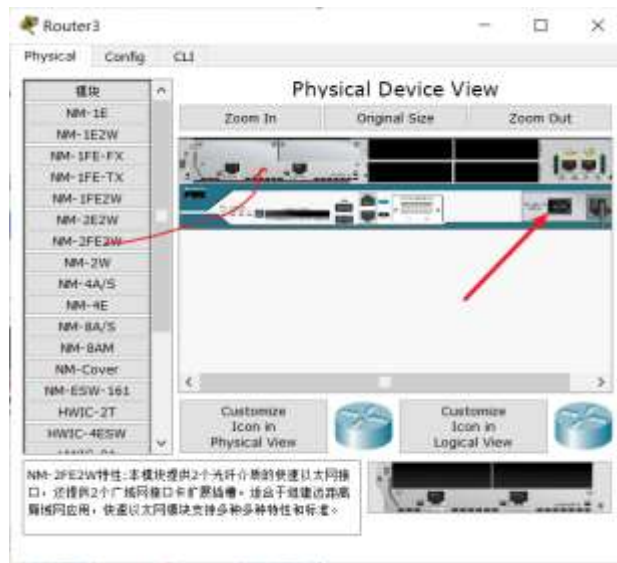
Reply from 192.168.1.3: bytes=32 time=0ms TTL=255
Reply from 192.168.1.3: bytes=32 time=0ms TTL=255
Reply from 192.168.1.3: bytes=32 time=0ms TTL=255
Reply from 192.168.1.3: bytes=32 time=0ms TTL=255

Ping statistics for 192.168.1.3:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 0ms, Maximum = 0ms, Average = 0ms

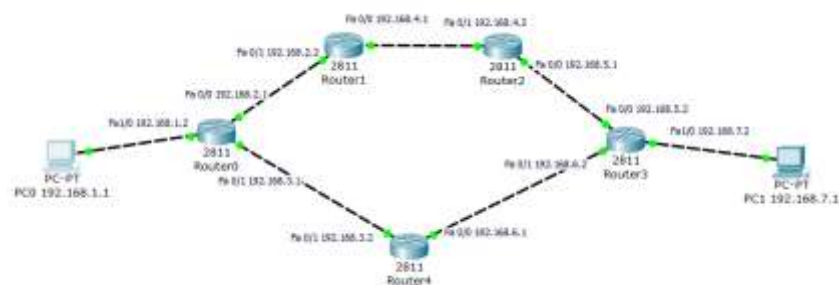
PC>
```

## 2) 动态路由配置

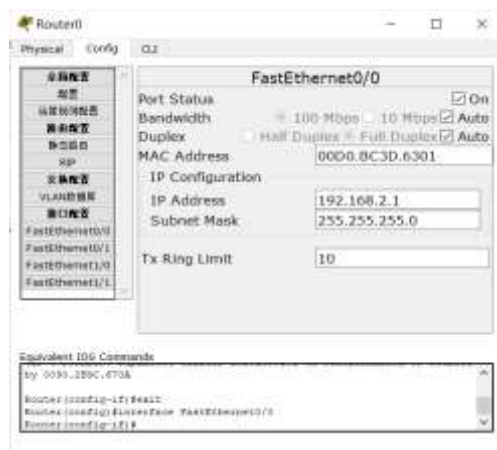
首先增加部分路由器端口,先关闭电源,再把 NM-2FE2W 拖拽到如图所示位置



拓扑结构如图:



除了命令行外可以在 config 界面直接设置  
以 Router0 为例:





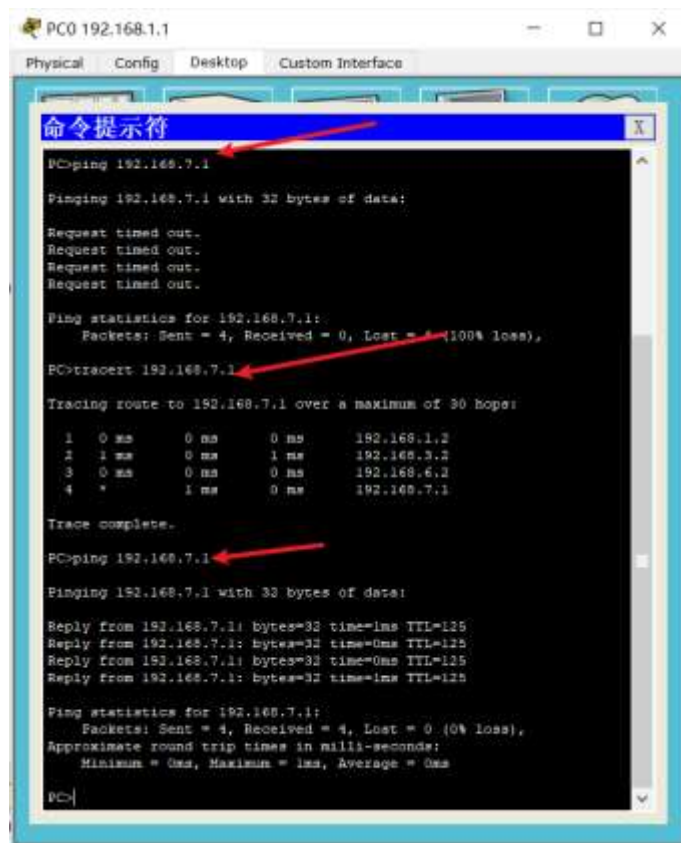


接着在 config 界面给路由器配置 RIP 路由,以 Router0 为例:



其余不赘述

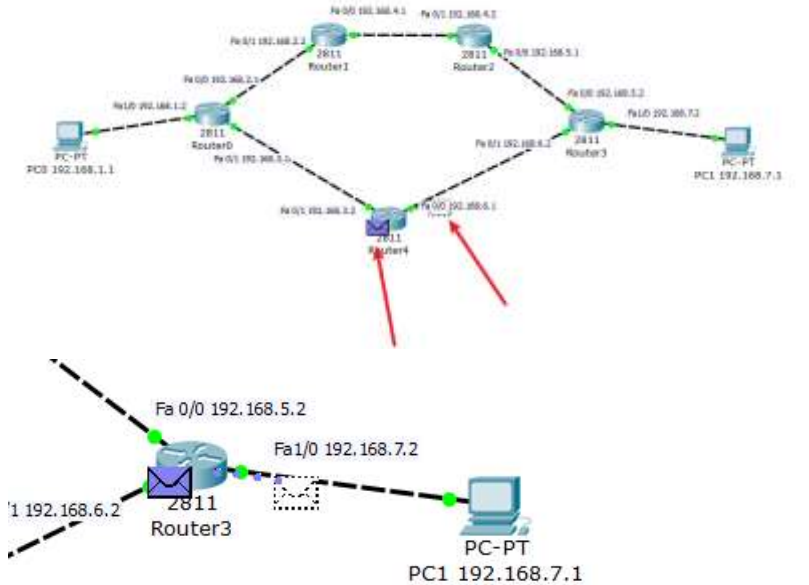
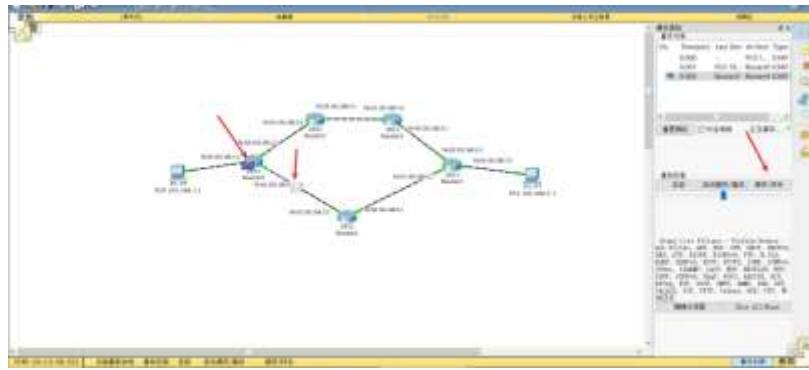
那么全部设置成功后,第一次 ping 会失败,因为此时会建立线路如图



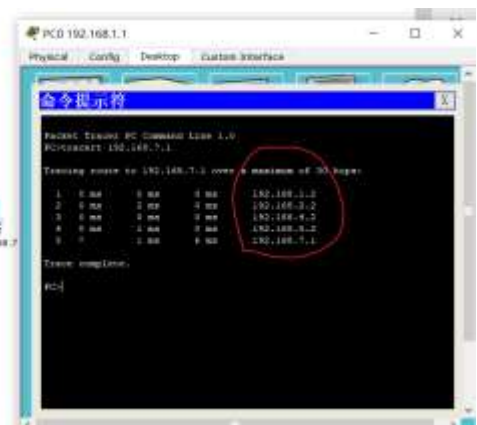
那么输入 `tracert 192.168.7.1` 命令后可以看到路径是沿着下面的路发送 ip 数据包的

在思科界面中也可以看到





即 IP 数据包发送路径如上  
那么我们尝试断开下面的路段,继续查看数据包发送路径如下



### 三、 任务三

编程要求：两位同学一组，一位同学做 TCP 客户端，一位同学做 TCP 服务器。要求实现客户端和服务端端的文本通信，在客户端输入文本后，能够将该文本发送至服务器端正确显示。同样，在服务器端输入文本后，能够将该文本发送至客户端正确显示。完成程序后，要求对所编写程序产生的网络数据进行抓包分析。要求必须输出以下字段：TCP：源端口、目的端口、序号、确认号、标志位，IP：版本号、总长度、标志位、片偏移、协议、源地址和目的地址。对抓包结果进行解析，对所编写的程序进行理解，并解释关键语句的作用，以及其产生的数据包，并完成详细的说明文档。文档中内容包括所使用的实验软件和操作系统、程序的设计思想、流程图、关键问题和关键语句、程序注释和对捕获包的解析截图/与程序语句的关联、总结和心得体会等

#### 1. 运行环境

操作系统:WIN10

编译环境:IDEA 2020.3

处理器 Intel(R) Core(TM) i7-9750H CPU @ 2.60GHz 2.60 GHz

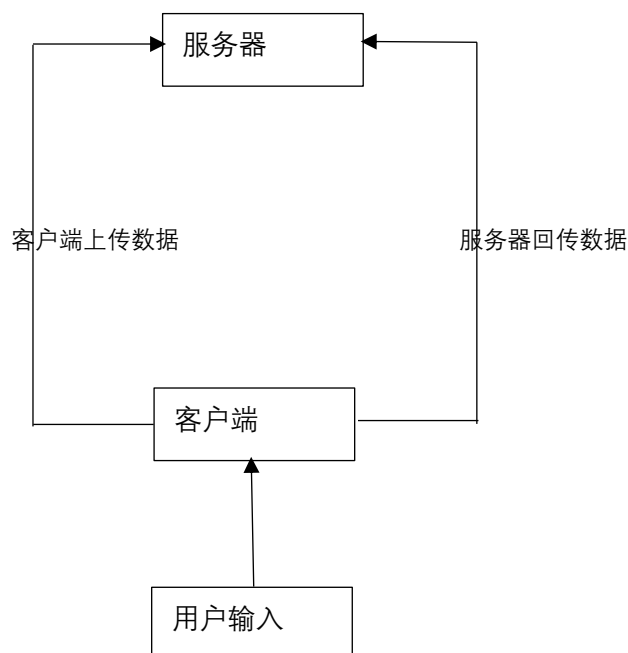
#### 2. 设计目标

基于 TCP 的 Socket 类和 Java 多线程实现类似 QQ 的多人聊天工具

#### 3. 核心思想

主要利用 ServerSocket 和 Socket 类连接程序,并依靠其 DataOutputStream 的 writeUTF 和 readUTF 函数, BufferedWriter 的 write 和 flush 函数传输数据

#### 4. 流程图



## 5. 关键问题,核心代码

- 1) 服务器怎么连接客户端:

经过网上多个博客的学习,服务器与客户端的连接主要依靠 ServerSocket 类和 Socket 类,如下

```
serverSocket = new ServerSocket(port: 8000);  
while (true) {  
    //等待客户端的连接  
    socket = serverSocket.accept();
```

首先用 serverSocket 用于等待客户端的连接,其 accept 函数会阻塞这个过程直到客户端连接.

- 2) 服务器怎么连接多个客户端:

主要依靠两个:while 循环和 Java 多线程

其中 while 循环是为了不停地接受客户端连接请求,而多线程是为了开启多个线程,接受客户端数据

```
serverSocket = new ServerSocket(port: 8000);  
while (true) {  
    //等待客户端的连接  
    socket = serverSocket.accept();  
    socketMap.put(socket, socketMap.size()+1);  
    int index=count++;  
    sendMessageToAll(message: "已上线", index);  
    //每当有一个客户端连接上来后,就启动一个单独的线程进行处理  
    new Thread(new Runnable() {  
        @Override  
        public void run() {  
            //获取输入流,并指定统一的编码格式  
            BufferedReader bufferedReader = null;  
            try {  
                bufferedReader = new BufferedReader(new InputStreamReader(socket.getInputStream(), charsetName: "UTF-8"));  
                //读取一行数据  
                //通过while循环不断读取数据  
                while ((message = bufferedReader.readLine()) != null) {  
                    //输出打印  
                    sendMessageToAll(message, index);  
                }  
            } catch (IOException e) {  
                e.printStackTrace();  
            }  
        }  
    }).start();
```

不过这会有一个弊端,就是服务器会卡在这里死循环,其 UI 界面会卡住.

- 3) 客户端怎么连接服务器:

客户端连接服务器需要端口号,IP 地址,代码如下:

```
IP=InetAddress.getLocalHost().getHostAddress();  
socket = new Socket(IP, port: 8000);
```

- 4) 客户端怎么上传数据给服务器
- 5) 当客户端连接上服务器后就可以传输数据,代码如下

首先依赖 BufferedWriter 类接受 Socket 的输出流,并将数据输入到 Socket 中,再 flush 出去

```
BufferedWriter bufferedWriter;  
  
IP=InetAddress.getLocalHost().getHostAddress();  
socket = new Socket(IP, port: 8000);  
//通过socket获取字符流  
bufferedWriter = new BufferedWriter(new OutputStreamWriter(socket.getOutputStream()));  
//获取服务器返回信息
```

```
bufferedWriter.write(str: message+"\n");
bufferedWriter.flush();
```

- 6) 服务器怎么回传数据给客户端

首先客户端需要在登录的时候开启一个线程,用于接收服务器传回来的数据:

```
//获取服务器返回值
new Thread(() -> {
    try {
        DataInputStream is= new DataInputStream(socket.getInputStream());
        String info = null;
        while((info=is.readUTF())!=null){
            TextContent.append(info);
        }
    } catch (IOException ioException) {
        ioException.printStackTrace();
    }
}).start();
```

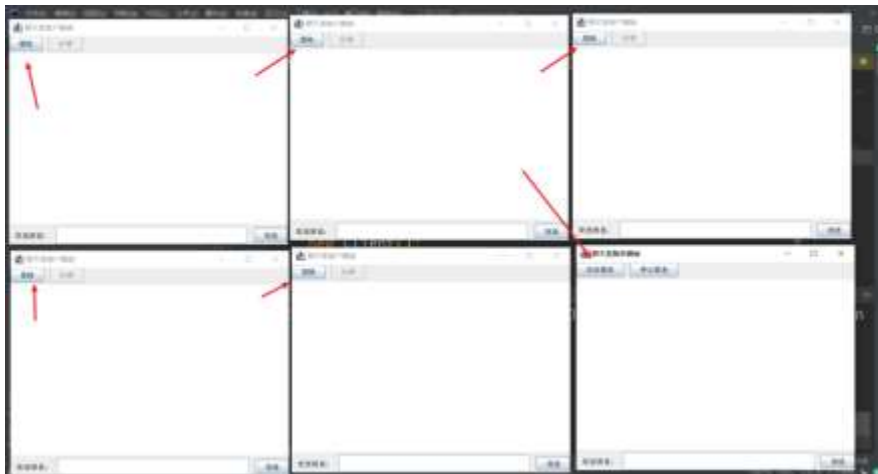
- 7) 其次服务器需要一个存放客户端 Socket 的数据,这里我使用了 Map 类,既存放了 Socket 数据,也存放了他们对应的客户端编号,用循环遍历和 DataOutputStream 类把数据输出到 Socket 中去

```
for(Map.Entry<Socket,Integer> entry:socketMap.entrySet()){
    DataOutputStream dataOutputStream = new DataOutputStream(entry.getKey().getOutputStream());
    dataOutputStream.writeUTF(str: "用/"+"index+": "+message+"\n"); //发送消息
    dataOutputStream.flush();
}
```

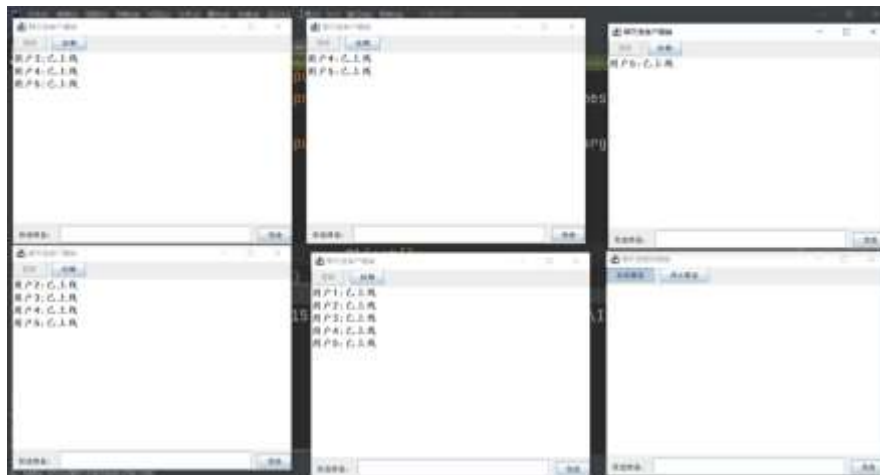
- 8) 阿巴巴巴

## 6. 运行界面

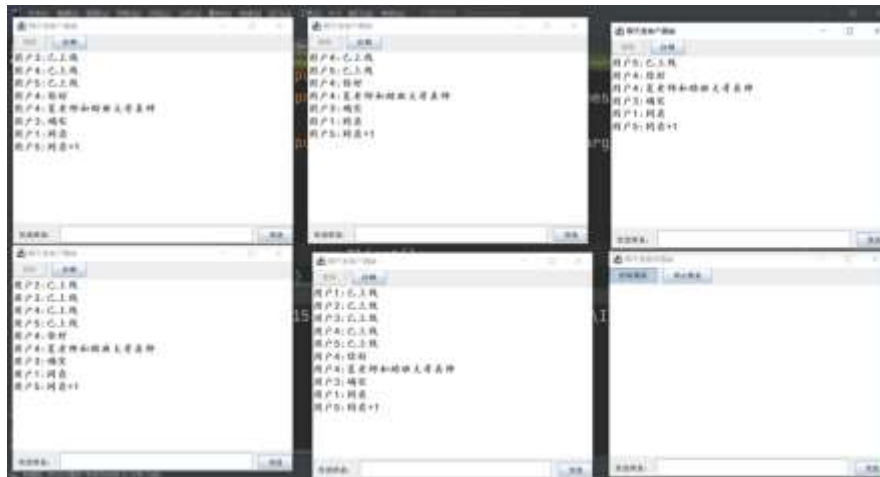
首先需要服务器运行服务,然后客户端登录



用户登录



然后用户发送数据:



## 7. 抓包分析

Wireshark packet capture analysis showing a TCP connection. The packet list shows a sequence of packets from 1 to 10, all from 10.200.0.1 to 10.200.0.2. The packet details pane shows the selected packet (10) with the following information:

Destination Port: 8000  
[Stream index: 0]  
[TCP Segment len: 0]  
Sequence Number: 16 (relative sequence number)  
Sequence Number (raw): 354436751  
[Next Sequence Number: 16 (relative sequence number)]  
Acknowledgment Number: 55 (relative ack number)  
Acknowledgment Number (raw): 357456164  
0101 ... = Header Length: 20 bytes (5)  
Flags: 0x010 (ACK)  
Window: 10233  
[Calculated window size: 2619648]  
[Window size scaling factor: 256]  
Checksum: 0x856f [unverified]  
[Checksum: 0x856f, 0x856f, 0x856f, 0x856f]

The packet bytes pane shows the raw data of the packet, which is a single byte (0x01) representing the ACK flag.

首先确认的是前三个包 1-3:三次握手.

其次是服务器返回给所有客户端的一个数据和客户端返回的数据包 4-5(只登陆了一个客户端),

接着是客户端发向服务器的一个聊天记录(内容 111),然后服务器返回一个确认包 120-121,

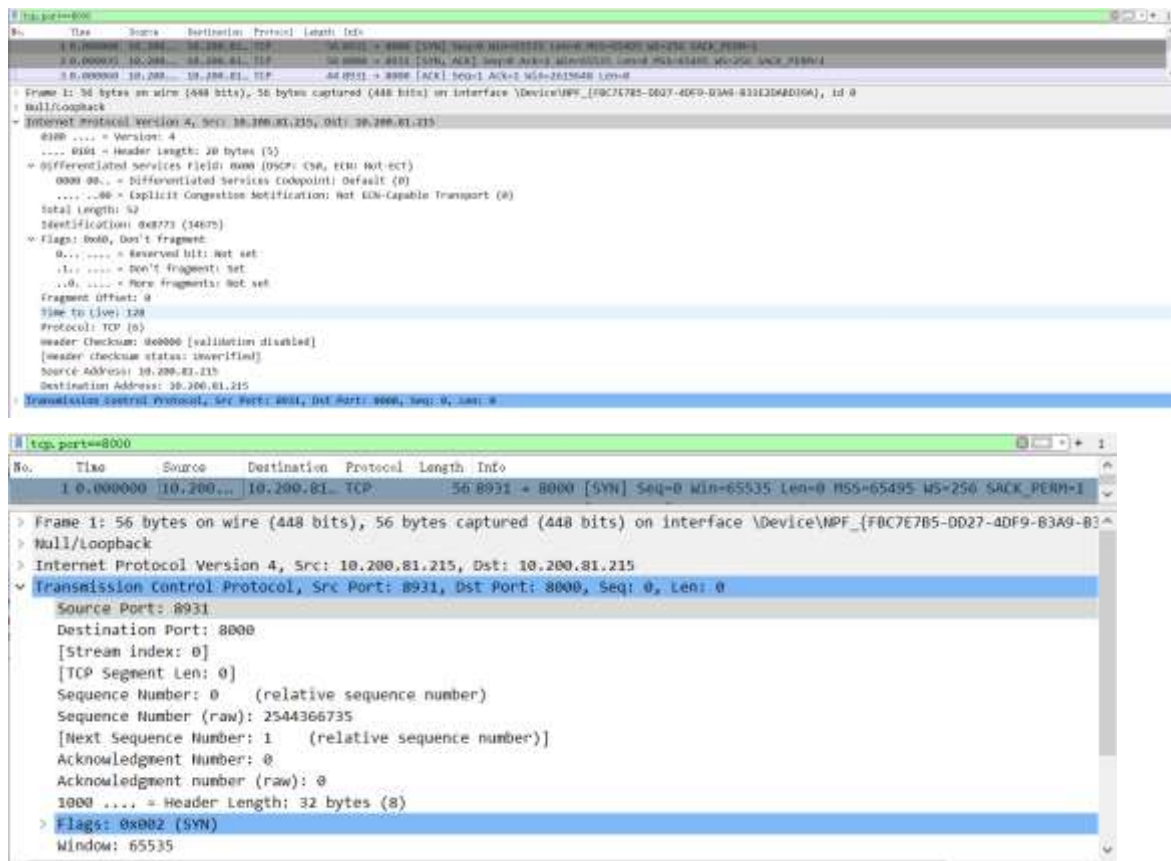
然后是服务器返回给所有客户端的一个数据和客户端返回的数据包 122-123(只登陆了一个客户端),

然后是服务器返回给所有客户端的一个数据(用于客户端提示下线)和客户端返回的数据包 166954-166954(只登陆了一个客户端),

最后客户端断开连接 166955-166958,对应四次挥手

## 1) 第一次抓包

截图如下,其余抓包截图不再放了



TCP: 源端口:8931;目的端口 8000;序列号: Seq=0;确认号:0;标志位:SYN=1,其余无;

IP: 版本号(Vers4);总长度(52);标志位 DF=1 片偏移 0 协议 TCP6 源地址(10.200.81.255),目的地址(10.200.81.255)

## 2) 第二次:

TCP: 源端口:8000;目的端口 8931;序列号: Seq=0;确认号:1;标志位:SYN=1,ACK=1 其余无;

IP: 版本号(Vers4);总长度(52);标志位 DF=1 片偏移 0 协议 TCP6 源地址(10.200.81.255),目的地址(10.200.81.255)

## 3) 第三次

TCP: 源端口:8000;目的端口 8931;序列号: Seq=1;确认号:1;标志位: ACK=1 其余无;

IP: 版本号(Vers4);总长度(40);标志位 DF=1 片偏移 0 协议 TCP6 源地址(10.200.81.255),目的地址(10.200.81.255)

至此三次握手完毕

那么第四次和第五次的报文分别为 1:客户端连接服务器后,服务器给客户端的一个响应,用于提示用户上线

2:客户端接收 4 后的回应.

## 4) 第四次

TCP: 源端口:8000;目的端口 8931;序列号: Seq=1;确认号:21;标志位:PSH=1ACK=1 其余无;

IP: 版本号(Version4);总长度(60);标志位 DF=1 片偏移 0 协议 TCP6 源地址(10.200.81.255),目的地址(10.200.81.255)

5) 第五次

TCP: 源端口: 8931;目的端口 8000;序列号: Seq=1;确认号:21;标志位:ACK=1 其余无;

IP: 版本号(Version4);总长度(40);标志位 DF=1 片偏移 0 协议 TCP6 源地址(10.200.81.255),目的地址(10.200.81.255)

6) 对应序号为 4-5,120-121, 166954-166954 的数据包和第四次第五次的是几乎一样的,这里就不赘述了,

#### 下面是四次挥手

7) 第一次挥手:

TCP: 源端口: 8931;目的端口 8000;序列号: Seq=1;确认号:35;标志位:FIN=1,ACK=1 其余无;

IP: 版本号(Version4);总长度(40);标志位 DF=1 片偏移 0 协议 TCP6 源地址(10.200.81.255),目的地址(10.200.81.255)

8) 第二次

TCP: 源端口: 8000;目的端口 8931;序列号: Seq=1;确认号:16;标志位: ACK=1 其余无;

IP: 版本号(Version4);总长度(40);标志位 DF=1 片偏移 0 协议 TCP6 源地址(10.200.81.255),目的地址(10.200.81.255)

9) 第三次

TCP: 源端口: 8000;目的端口 8931;序列号: Seq=1;确认号:16;标志位:PSH=1,ACK=1 其余无;

IP: 版本号(Version4);总长度(60);标志位 DF=1 片偏移 0 协议 TCP6 源地址(10.200.81.255),目的地址(10.200.81.255)

10) 第四次

TCP: 源端口: 8931;目的端口 8000;序列号: Seq=1;确认号:55;标志位: ACK=1 其余无;

IP: 版本号(Version4);总长度(40);标志位 DF=1 片偏移 0 协议 TCP6 源地址(10.200.81.255),目的地址(10.200.81.255)

至此四次挥手分析完毕

## 8. 实验遇到的问题:

1 服务器只能连接一个客户端:

解决方案:利用多线程,为每一个连接到服务器的客户端分配一个线程用于接收信息,详细过程见上文

2 服务器无法返回数据给客户端

原来以为需要客户端再建一个 Socket 用来连接服务器,但其实我上网查询资料后得知,服务器的 serverSocket 所创建的 Socket 在连接上客户端后,其实不但可以接收客户端的数据,还可以传输数据给客户端,只需要使用 DataOutputStream 把数据传回去即可,详细见上文;

## 9. 实验总结

通过这次实验,我通过上网查询到了许多原本不会的知识和技能,我学会了如何在思科软件上模拟路由器,交换机,PC 的实现,还学会了 Java 的一个重要的类 Socket 的使用,通过这个类的学习,我对 Socket 编程的工作原理有了更深的理解,还有 cmd 的命令,包括 ipconfig,ping, netstat, tracert,arp, telnet 等等,还有利用 Wireshark 软件对数据包进行抓取,并借这个软件的强大功能进行了 TCP 的三次握手,发送的数据包和四次挥手的分析,加深了我对 TCP 和 IP 的理解.以上种种,不但需要复习课本内容,理解其概念意义,还需要自行上网查询资料,查看不同的人的思路,即使前人已经把路铺好了,但是我还是走过许多的坑,被好多 bug 绊倒,但是依然完成了这次课设.感谢这次课设,加深了我对计算机网络的理解..