



# 合肥工业大学

## 计算机与信息学院

### 实验报告

课    程：计算机网络系统实训

姓    名：孙琳

学    号：2018211958

专业班级：计算机科学与技术 18-2 班

日    期：3 月 8 日

# 目录

路由器配置实验 .....	3
一、实验目的.....	3
二、实验环境与设备.....	3
三、实验内容.....	3
四、实验步骤.....	3
1. 路由器接口的配置 .....	3
2. 静态路由的配置 .....	18
3. 设置默认路由 .....	24
4. 动态路由协议 RIP 的配置 .....	27
五、实验总结.....	33

# 路由器配置实验

## 一、实验目的

1. 认识路由器的端口、型号
2. 掌握路由器的路由配置
3. 理解网络互联的基本原理

## 二、实验环境与设备

本实验在 PC 机上利用模拟软件 Packet Tracer V6 进行操作。

## 三、实验内容

1. 路由器接口的配置
2. 静态路由配置
3. 默认路由配置
4. 动态路由配置

本次实验的主要任务是了解路由器的基本设置，与网络之间的连接关系。通过这次的实验很好的掌握了各个网段之间的，各个路由器下的 pc 的连接情况。通过对静态，默认，动态路由配置，使得各个路由器下的 PC 相互通信。

## 四、实验步骤

下面的步骤按照实验指导书中的步骤严格进行，并对实验中碰到的问题及解决方案进行介绍。

### 1. 路由器接口的配置

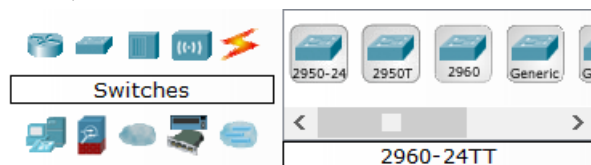
此处我按照实验指导书给定的拓扑结构图进行创建。

需要注意的是路由器、交换机、终端节点机的种类不能选错，并且各个序号对应特定位置也要看清，如上面三个路由器是以 Router0、Router2、Router1 的顺序摆放的，如果没注意到这一点，且其他结构又不做出相应改变的话，也会得到错误的拓扑结构，后面的实验都得不到正确的结果。

路由器选择 Router-PT



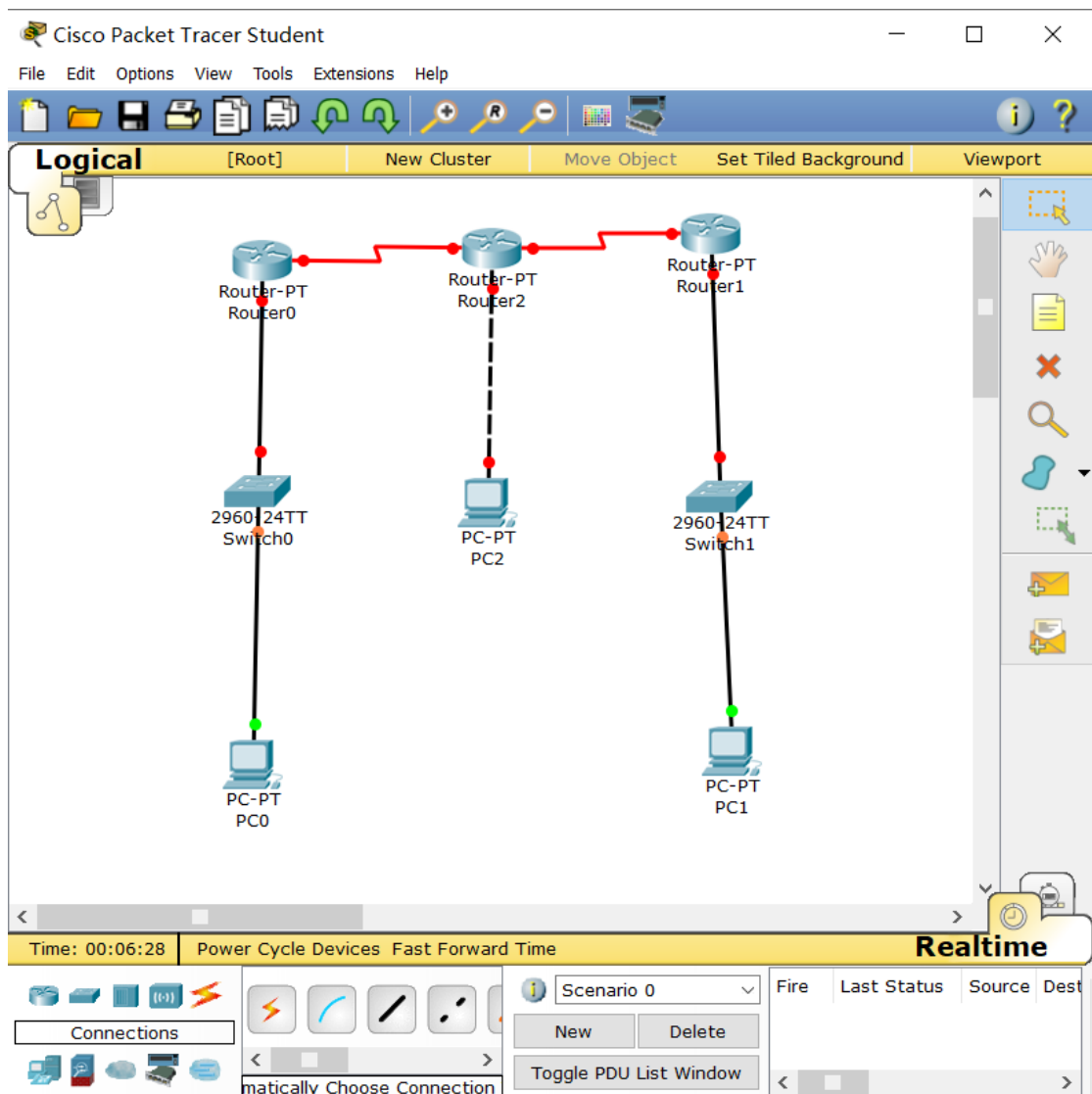
交换机选择 2960-24TT



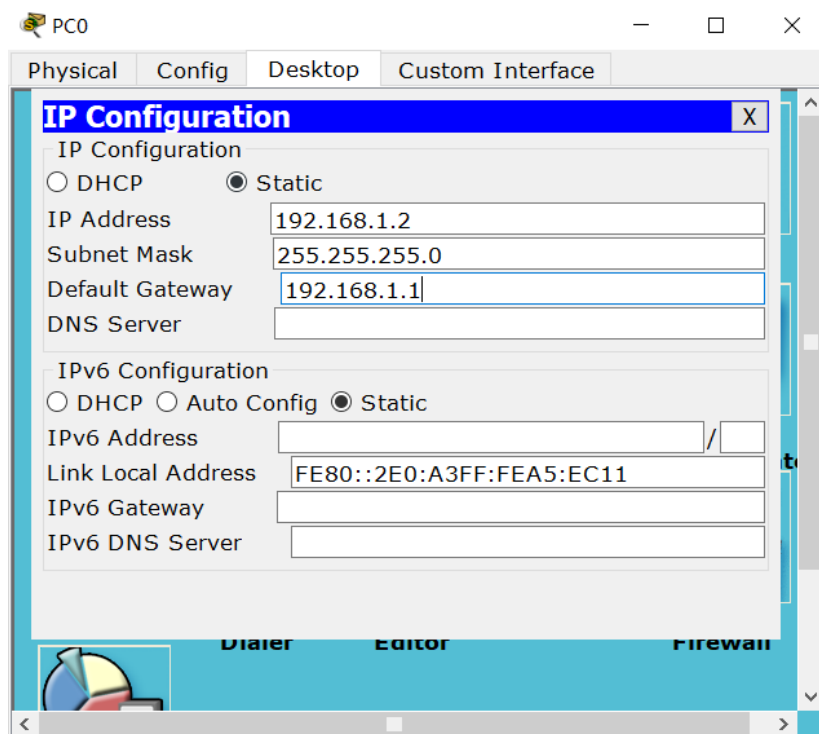
终端节点机选择 PC-PT



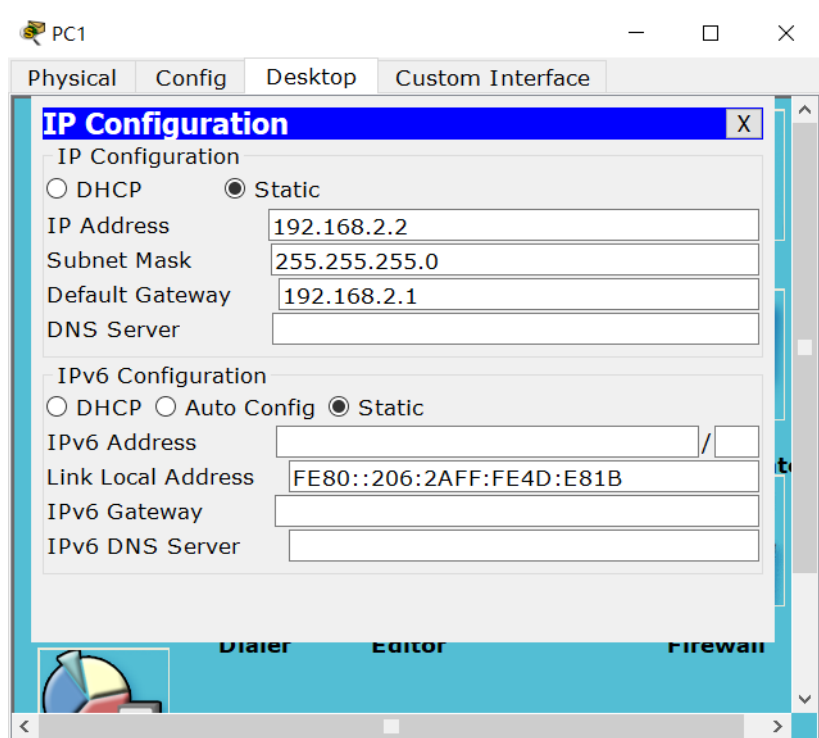
连线后得到如下拓扑结构图



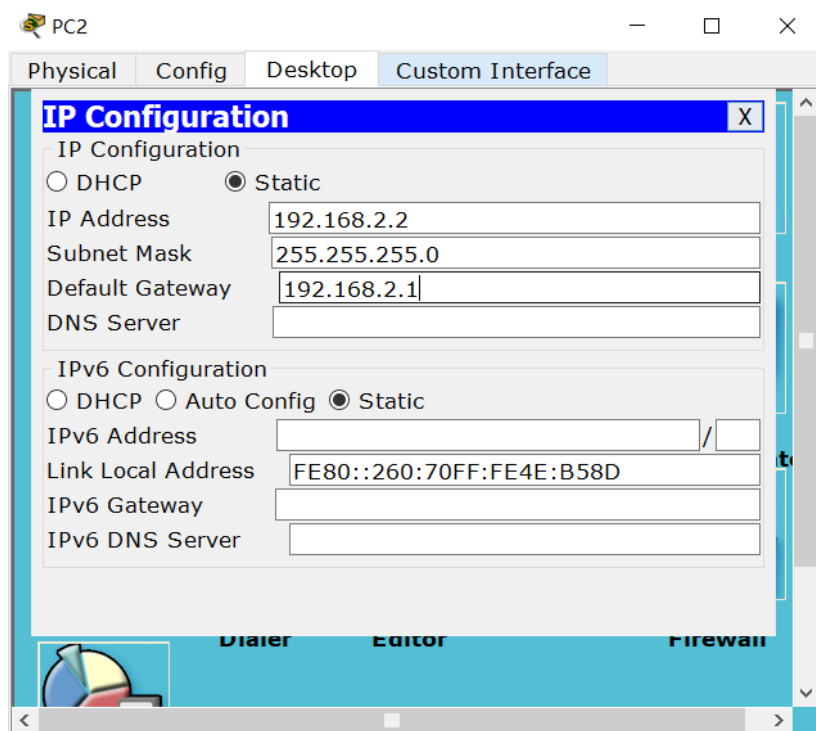
可以看到此时除了交换机与 PC 机之间是一个橙色一个绿色的节点，  
为了使橙色变绿  
我们要对三个主机进行对应的配置，  
PC0 配置如下：



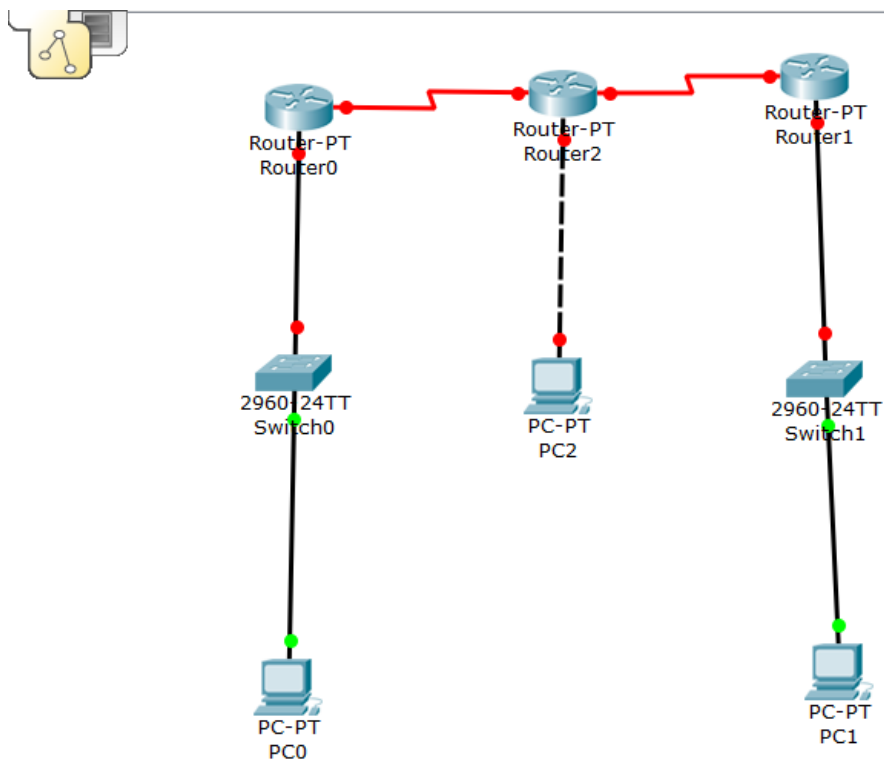
PC1 配置如下：



PC2 配置如下：



可以看到配置好后，交换机与 PC 机之间变成了双绿色节点。



接下来继续为每个路由器的各个接口配置，该系统中默认交换机不用配置。

Router0 的 F0/0 端口配置如下：

The screenshot shows the configuration window for Router0, specifically the 'Config' tab for the 'FastEthernet0/0' interface. The 'Port Status' section has a red box around the 'On' checkbox, which is checked. Other settings include 'Bandwidth' set to '100 Mbps', 'Duplex' set to 'Auto', 'MAC Address' as '0090.0CDD.C724', 'IP Address' as '192.168.1.1', 'Subnet Mask' as '255.255.255.0', and 'Tx Ring Limit' as '10'. The 'Equivalent IOS Commands' section at the bottom shows the commands for enabling the interface and the line protocol.

Router0

Physical Config CLI

**FastEthernet0/0**

Port Status ☒ On

Bandwidth ☒ 100 Mbps ☐ 10 Mbps ☐ Auto

Duplex ☒ Half Duplex ☐ Full Duplex ☒ Auto

MAC Address 0090.0CDD.C724

IP Configuration

IP Address 192.168.1.1

Subnet Mask 255.255.255.0

Tx Ring Limit 10

Equivalent IOS Commands

```
%LINK-5-CHANGED: Interface FastEthernet0/0, changed state to up
%LINEPROTO-5-UPDOWN: Line protocol on Interface FastEthernet0/0, changed state to up
```

需要注意的是，图中红框中的 On 必须选中，仅配置而不打开接口是无效的。

Router0 的 S2/0 端口配置如下：

The screenshot shows the configuration window for Router0, specifically the 'Config' tab for the 'Serial2/0' interface. The 'Port Status' section has a red box around the 'On' checkbox, which is checked. Other settings include 'Duplex' set to 'Full Duplex', 'Clock Rate' set to 'Not Set', 'IP Address' as '172.16.1.2', 'Subnet Mask' as '255.255.0.0', and 'Tx Ring Limit' as '10'. The 'Equivalent IOS Commands' section at the bottom shows the commands for configuring the interface and the line protocol.

Router0

Physical Config CLI

**Serial2/0**

Port Status ☒ On

Duplex ☒ Full Duplex

Clock Rate Not Set

IP Configuration

IP Address 172.16.1.2

Subnet Mask 255.255.0.0

Tx Ring Limit 10

Equivalent IOS Commands

```
Router(config)#interface Serial2/0
Router(config-if)#no shutdown

%LINK-5-CHANGED: Interface Serial2/0, changed state to down
Router(config-if)#
```

需要注意的是，此时软件自动生成的子网掩码是错误的，IP 地址 172.16.1.2 对应的子网掩码应该是 255.255.255.0 而不是 255.255.0.0。（S2/0 都会出错）

Router1 的 F0/0 端口配置如下：

The screenshot shows the configuration window for Router1, specifically the FastEthernet0/0 interface. The window has tabs for Physical, Config, and CLI. The Config tab is active, showing a tree on the left with categories: GLOBAL (Settings, Algorithm Settings), ROUTING (Static, RIP), and INTERFACE (FastEthernet0/0, FastEthernet1/0, Serial2/0, Serial3/0, FastEthernet4/0, FastEthernet5/0). The FastEthernet0/0 interface is selected. The configuration fields are as follows:

- Port Status: ☒ On
- Bandwidth: ☐ 100 Mbps ☐ 10 Mbps ☒ Auto
- Duplex: ☐ Half Duplex ☐ Full Duplex ☒ Auto
- MAC Address: 0040.0B09.3390
- IP Configuration:
  - IP Address: 192.168.2.1
  - Subnet Mask: 255.255.255.0
- Tx Ring Limit: 10

Below the configuration fields, there is a section for Equivalent IOS Commands:

```
%LINK-5-CHANGED: Interface FastEthernet0/0, changed state to up
%LINEPROTO-5-UPDOWN: Line protocol on Interface FastEthernet0/0, changed state to up
```

Router1 的 S3/0 端口配置如下：

The screenshot shows the configuration window for Router1, specifically the Serial3/0 interface. The window has tabs for Physical, Config, and CLI. The Config tab is active, showing a tree on the left with categories: GLOBAL (Settings, Algorithm Settings), ROUTING (Static, RIP), and INTERFACE (FastEthernet0/0, FastEthernet1/0, Serial2/0, Serial3/0, FastEthernet4/0, FastEthernet5/0). The Serial3/0 interface is selected. The configuration fields are as follows:

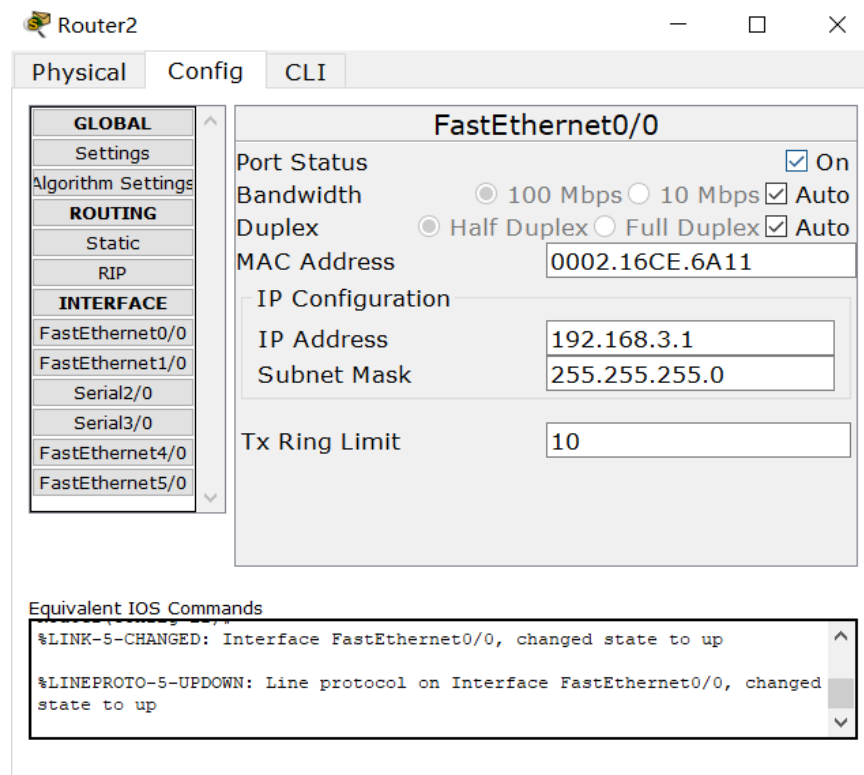
- Port Status: ☒ On
- Duplex: ☐ Full Duplex
- Clock Rate: 2000000
- IP Configuration:
  - IP Address: 172.16.2.2
  - Subnet Mask: 255.255.255.0
- Tx Ring Limit: 10

Below the configuration fields, there is a section for Equivalent IOS Commands:

```
Router(config-if)#
Router(config-if)#exit
Router(config)#
Router(config)#interface Serial3/0
Router(config-if)#
```

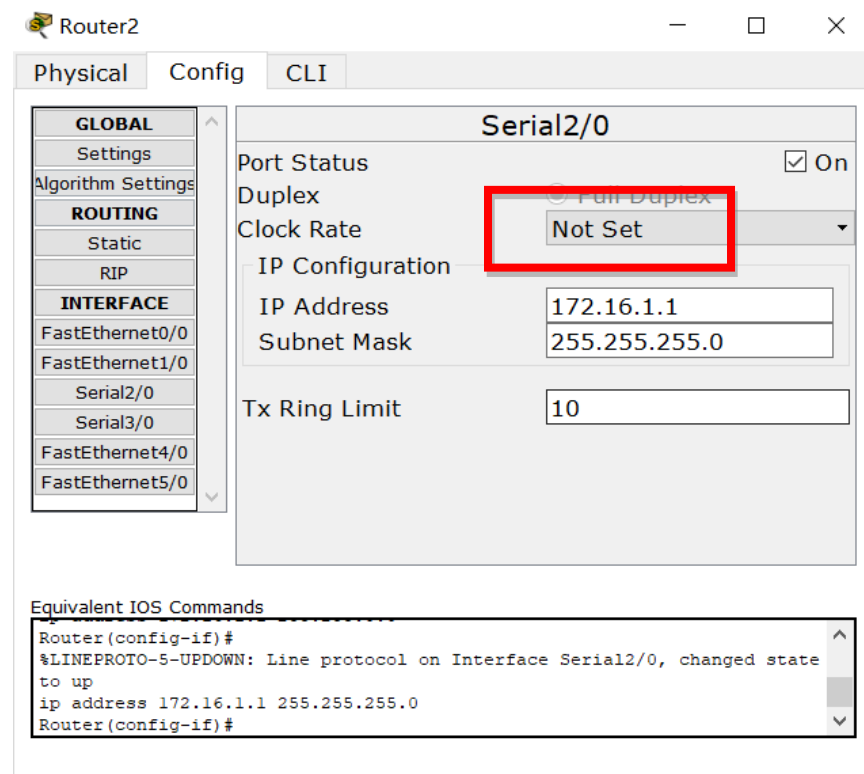


Router2 的 F0/0 的端口配置如下：



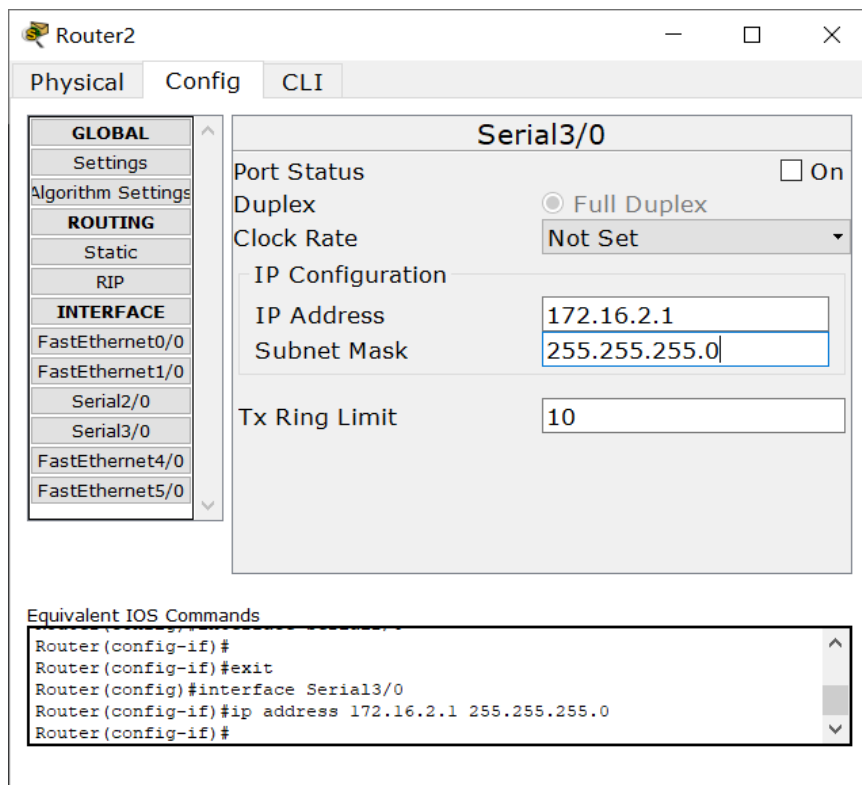
The screenshot shows the configuration window for Router2, specifically for the FastEthernet0/0 interface. The window has tabs for Physical, Config, and CLI. The Config tab is active, showing a tree view on the left with categories: GLOBAL, Settings, Algorithm Settings, ROUTING, Static, RIP, and INTERFACE. Under INTERFACE, FastEthernet0/0 is selected. The main configuration area for FastEthernet0/0 includes: Port Status (checked On), Bandwidth (radio buttons for 100 Mbps, 10 Mbps, and checked Auto), Duplex (radio buttons for Half Duplex, Full Duplex, and checked Auto), MAC Address (0002.16CE.6A11), IP Configuration (IP Address 192.168.3.1, Subnet Mask 255.255.255.0), and Tx Ring Limit (10). Below the configuration area, there is a section for Equivalent IOS Commands showing two lines: %LINK-5-CHANGED: Interface FastEthernet0/0, changed state to up and %LINEPROTO-5-UPDOWN: Line protocol on Interface FastEthernet0/0, changed state to up.

Router2 的 S2/0 的端口配置如下：

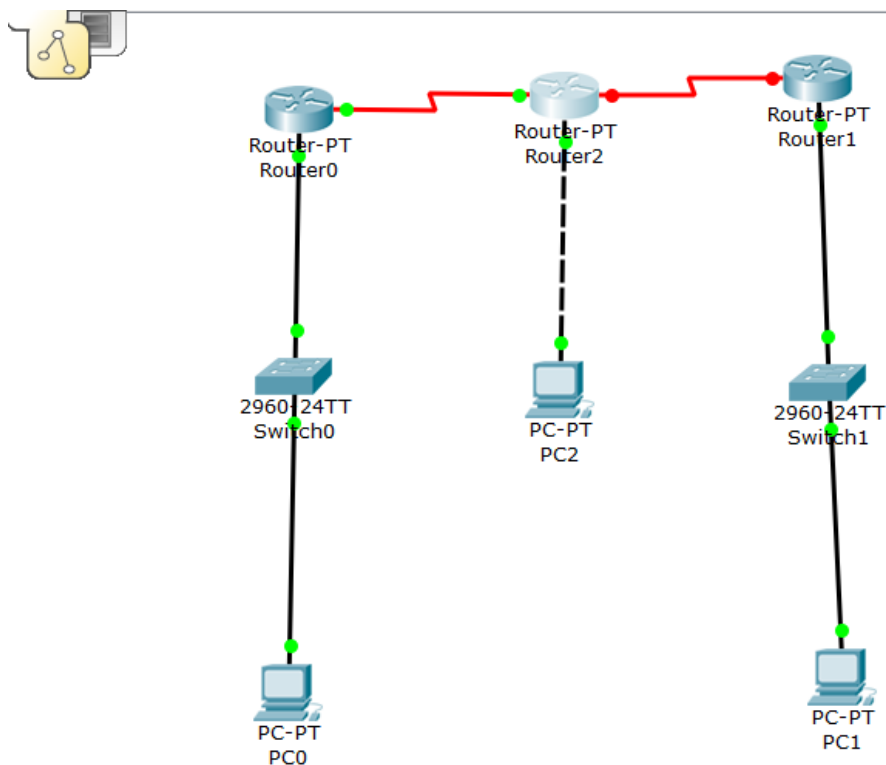


The screenshot shows the configuration window for Router2, specifically for the Serial2/0 interface. The window has tabs for Physical, Config, and CLI. The Config tab is active, showing a tree view on the left with categories: GLOBAL, Settings, Algorithm Settings, ROUTING, Static, RIP, and INTERFACE. Under INTERFACE, Serial2/0 is selected. The main configuration area for Serial2/0 includes: Port Status (checked On), Duplex (radio buttons for Half Duplex, Full Duplex, and Not Set, with the Not Set option highlighted by a red box), Clock Rate (Not Set), IP Configuration (IP Address 172.16.1.1, Subnet Mask 255.255.255.0), and Tx Ring Limit (10). Below the configuration area, there is a section for Equivalent IOS Commands showing three lines: Router(config-if)#, %LINEPROTO-5-UPDOWN: Line protocol on Interface Serial2/0, changed state to up, and ip address 172.16.1.1 255.255.255.0.

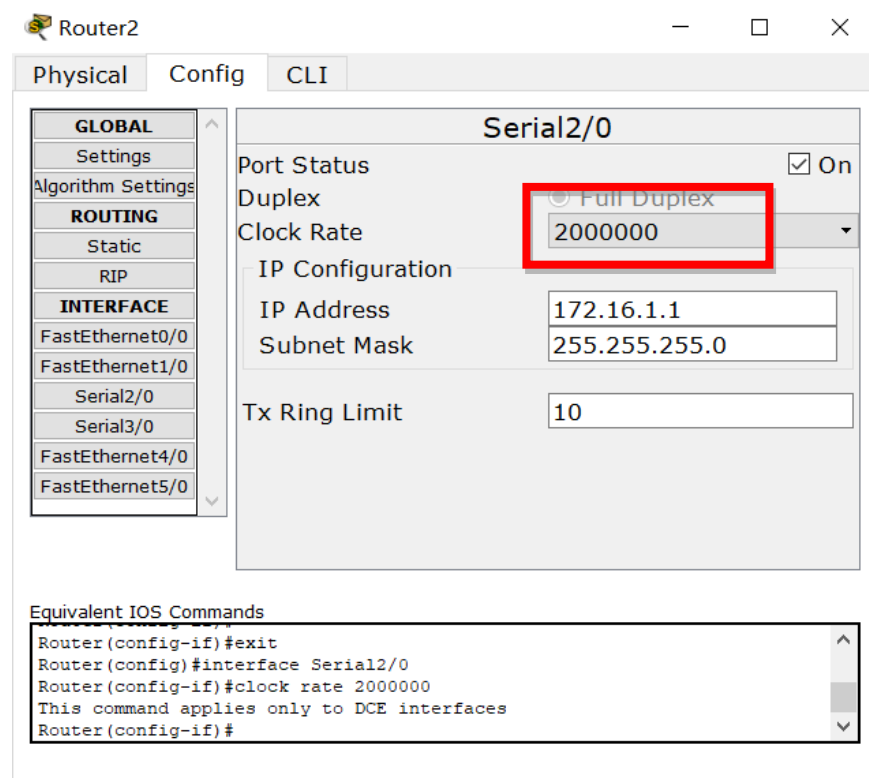
Router2 的 S3/0 的端口配置如下：



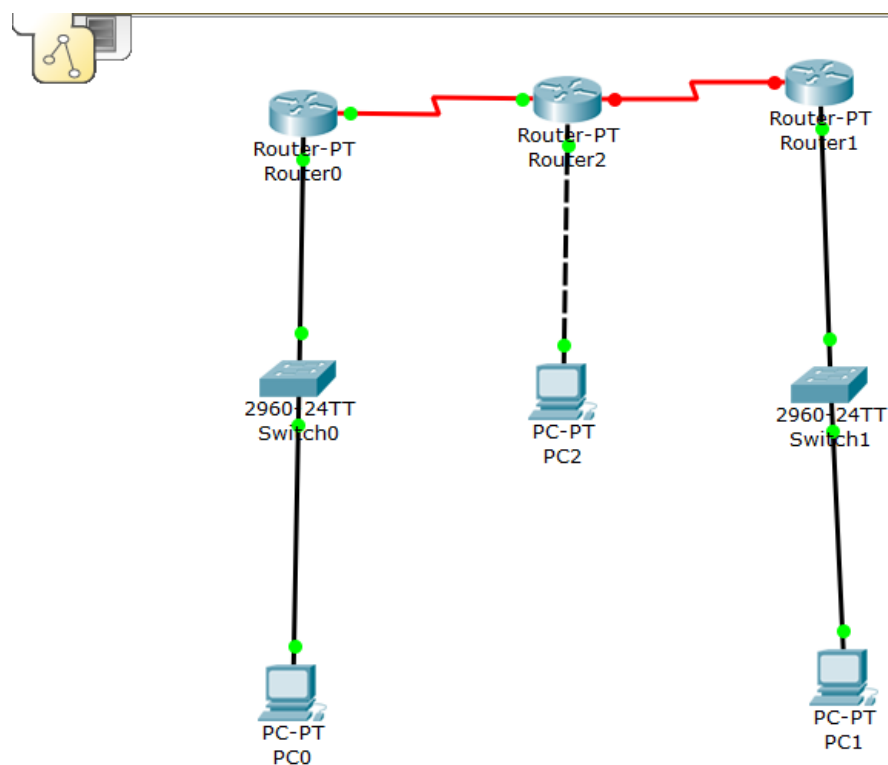
严格按照指导书配置完成后，得到的拓扑结构图是这样的：



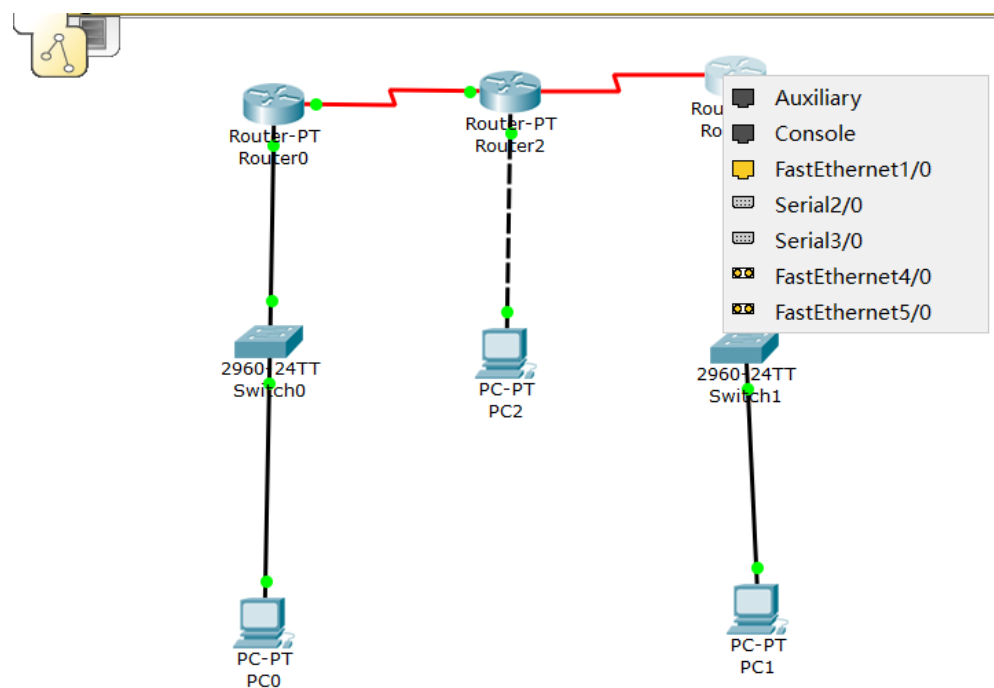
可以看到还是有红色节点，在仔细检查比对后，我觉得问题可能是出在有的端口的时钟频率速度是 Not Set，检查后我对没有设置的部分端口的时钟频率速度都统一设置为 2000000 如下（其余截图略）。



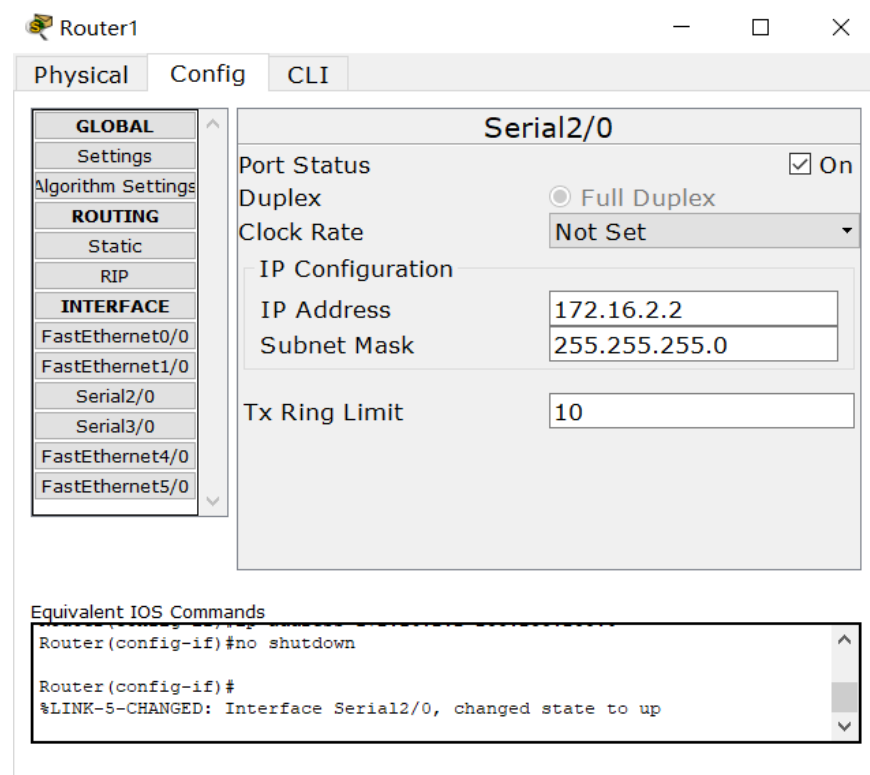
但是问题还是没有解决，仍旧是红点状态，问题出在哪呢？



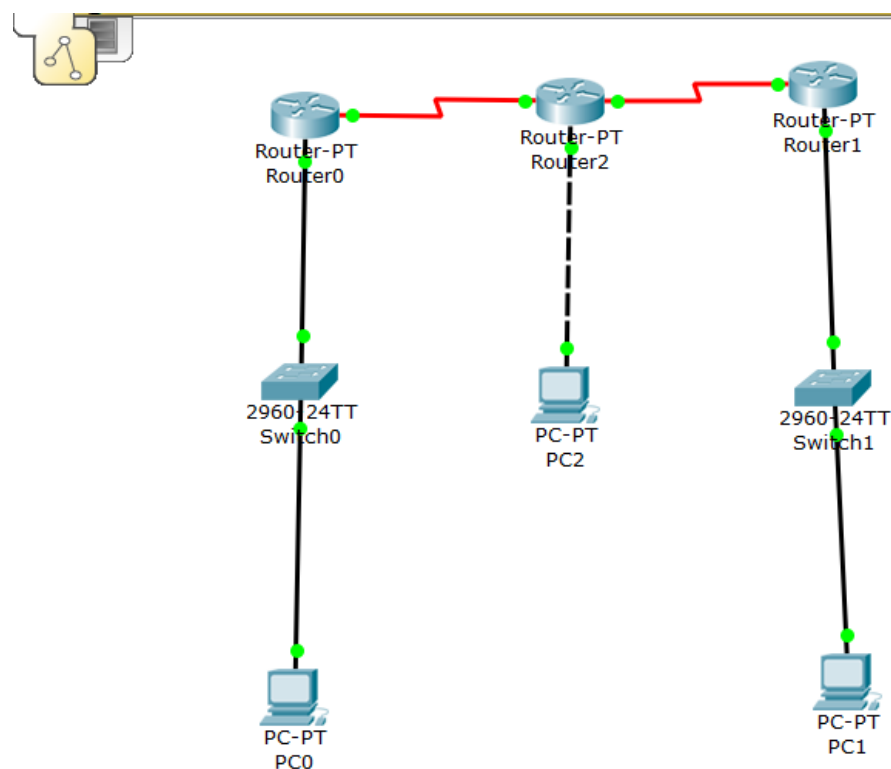
尝试了无数种办法后，我终于找到了问题所在，那就是断开线头再连的时候，会提示连接哪个端口，这个时候我将最右侧的 Router2 与 Router0、1 已经连接的端口进行比对，发现原来此处路由器左侧的端口是 Serial2/0，右侧的端口是 Serial3/0!所以问题出在 Router1 应当配置的端口应该是 Serial2/0!



于是我重新配置了 Router1 的 S2/0 端口，并且解除了对 S3/0 端口的配置。

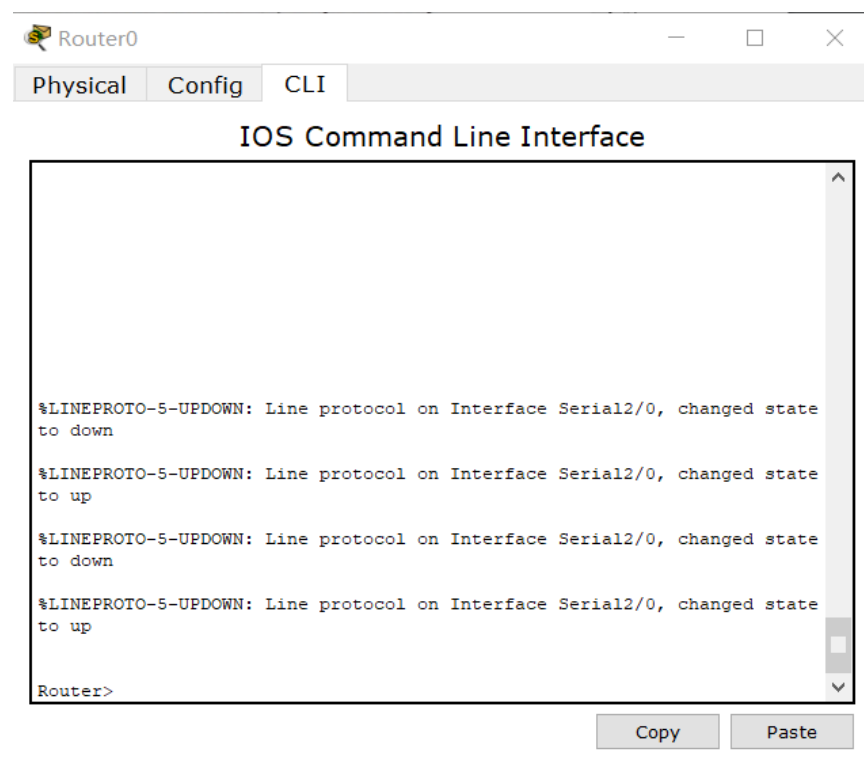


成功解决问题，所有端口全部变绿！



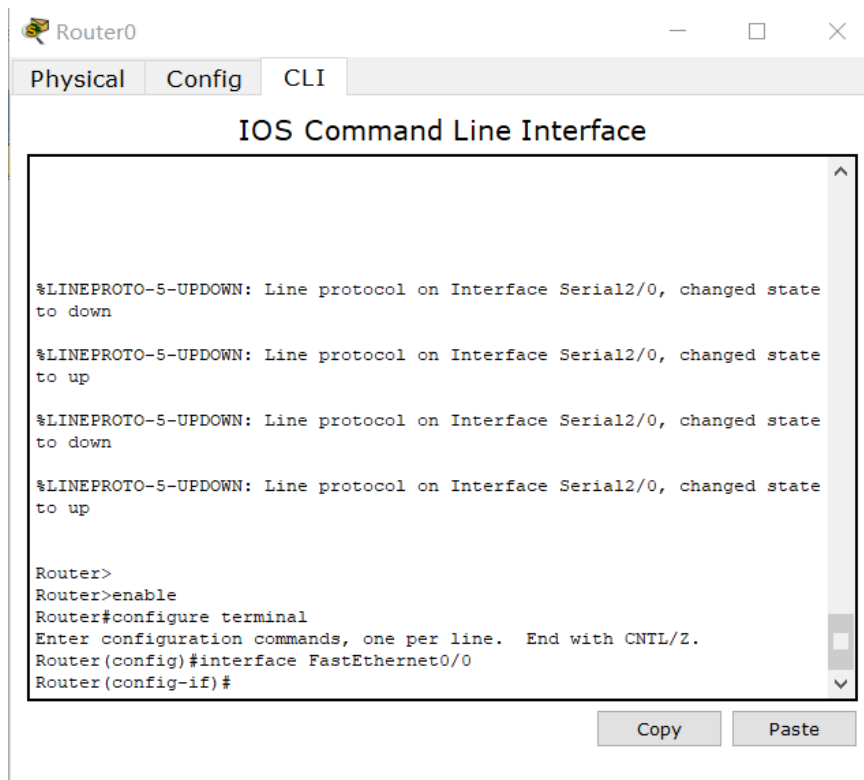
接下来查看路由器的路由表，需要使用指令 `show ip route`。

首先查看 Router0 的路由表，我们打开 Router0 的命令行接口（Command Line Interface = CLI），此时命令行是 Router>

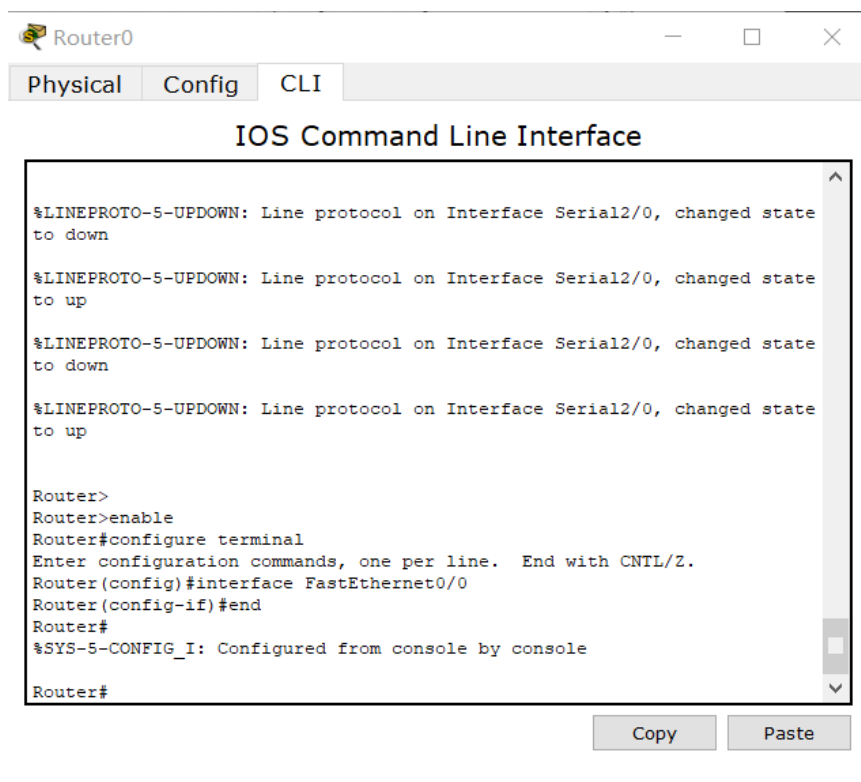


但是我们需要的是 Router(config-if)#, 最简单的办法就是先打开到 Config, 选择任意一个端口, 这时候可视化对应这边的命令行就是进入了

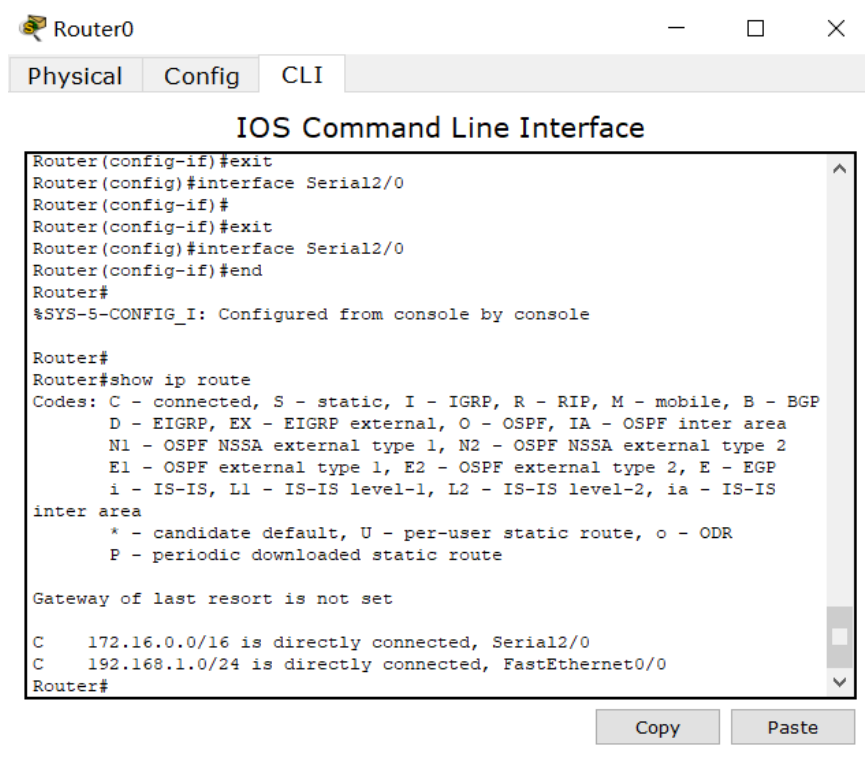
“Router(config)#interface 选择的端口名”, 如下图所示



此时再输入 end 结束, 再回车, 将进入到 Router#模式如下:



上面这种调整方式是我试出来的最快捷的进入 Router#的方法了，接下来我们在命令行输入 show ip route，显示出 Router0 的路由表如下：



The screenshot shows the Router0 CLI interface with the 'CLI' tab selected. The command history shows the configuration of interface Serial2/0. The output of the 'show ip route' command is displayed, showing the routing table with two entries: 172.16.0.0/16 and 192.168.1.0/24.

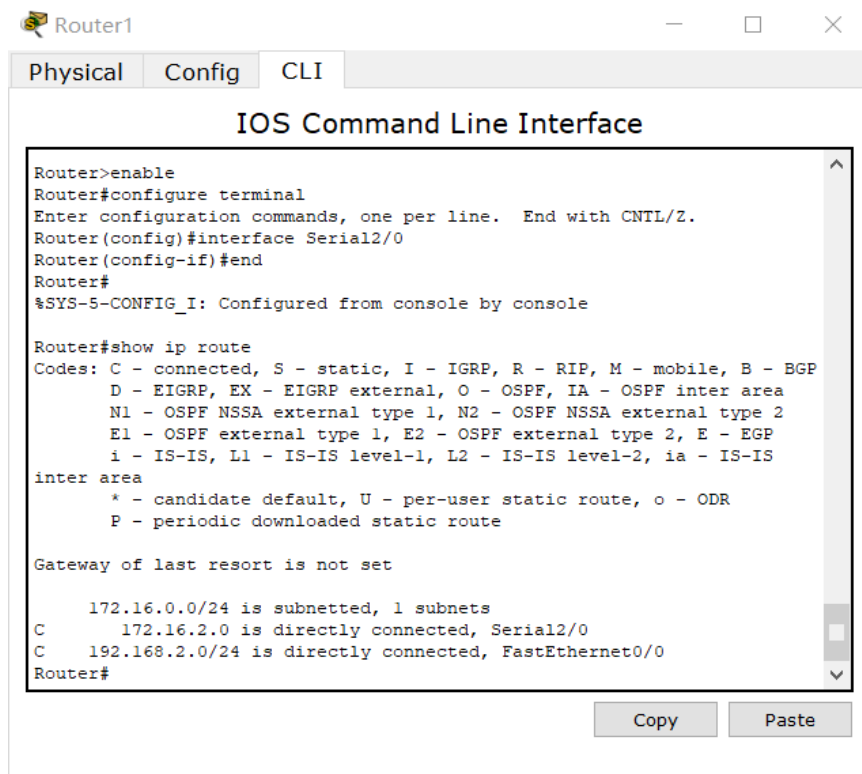
```
Router0
Physical Config CLI
IOS Command Line Interface
Router(config-if)#exit
Router(config)#interface Serial2/0
Router(config-if)#
Router(config-if)#exit
Router(config)#interface Serial2/0
Router(config-if)#end
Router#
%SYS-5-CONFIG_I: Configured from console by console

Router#
Router#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, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, ia - IS-IS
       inter area
       * - candidate default, U - per-user static route, o - ODR
       P - periodic downloaded static route

Gateway of last resort is not set

C      172.16.0.0/16 is directly connected, Serial2/0
C      192.168.1.0/24 is directly connected, FastEthernet0/0
Router#
```

同理 Router1 的路由表如下：



The screenshot shows the Router1 CLI interface with the 'CLI' tab selected. The command history shows the configuration of interface Serial2/0. The output of the 'show ip route' command is displayed, showing the routing table with three entries: 172.16.0.0/24, 172.16.2.0, and 192.168.2.0/24.

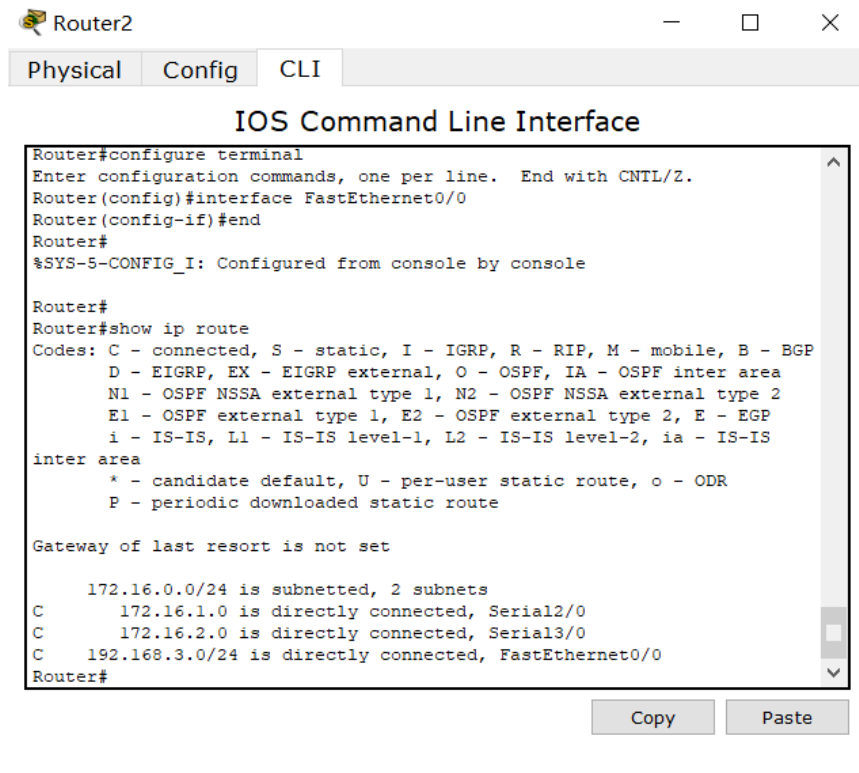
```
Router1
Physical Config CLI
IOS Command Line Interface
Router>enable
Router#configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
Router(config)#interface Serial2/0
Router(config-if)#end
Router#
%SYS-5-CONFIG_I: Configured from console by console

Router#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, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, ia - IS-IS
       inter area
       * - candidate default, U - per-user static route, o - ODR
       P - periodic downloaded static route

Gateway of last resort is not set

      172.16.0.0/24 is subnetted, 1 subnets
C      172.16.2.0 is directly connected, Serial2/0
C      192.168.2.0/24 is directly connected, FastEthernet0/0
Router#
```

同理 Router2 的路由表如下：



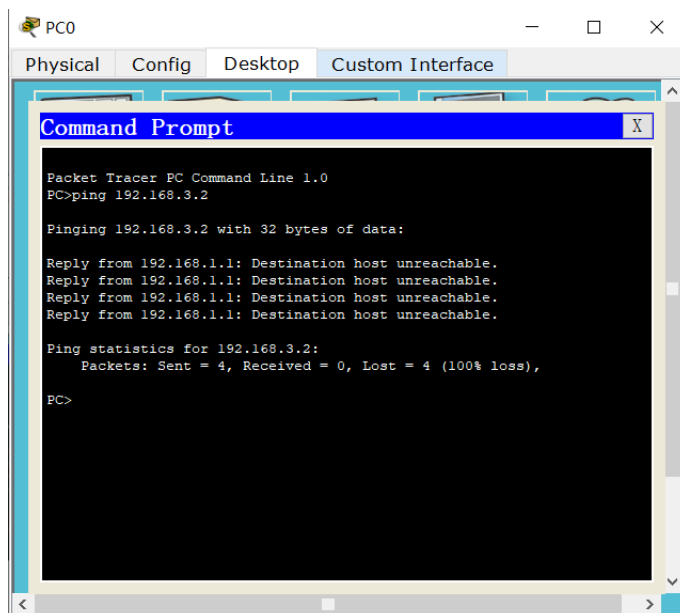
C 表示直连路由

这些路由器显示的路由信息将用于与后面的实验结果进行比较。

接下来，我们测试主机之间的连通性，总结下来基本方法就是：

以测试 PC0 与 PC2 之间的连通性为例，就是在 PC0 上试着 ping PC2 的 ip 地址，或者在 PC2 上 ping PC0 的 ip 地址，如果 ping 通，就证明连通，反之亦反。

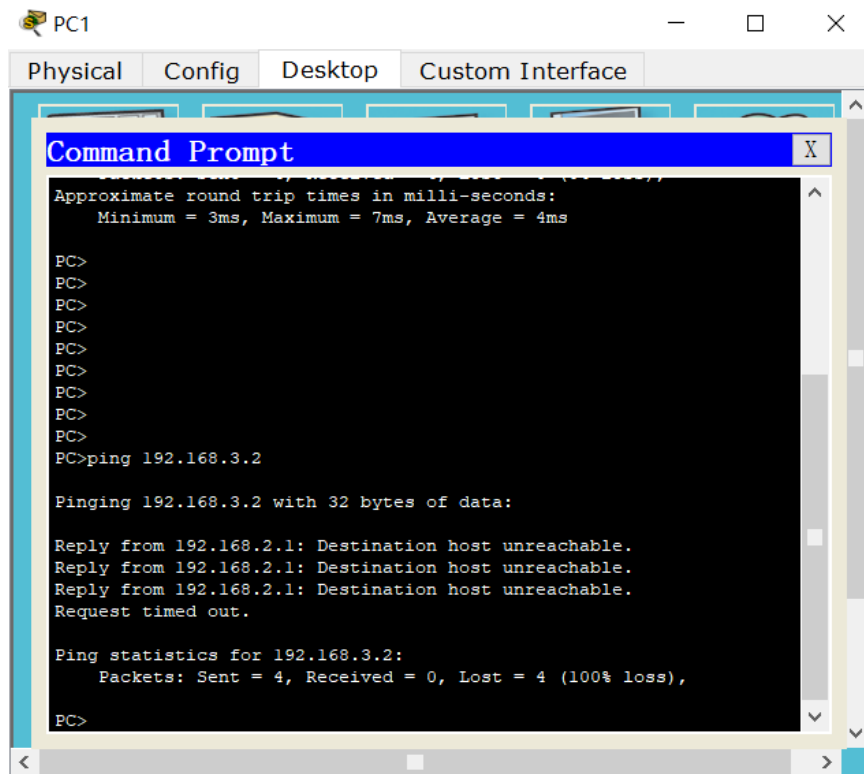
下面首先测试 PC0 与 PC2 之间的连通性：





可以看到是 100% loss, 说明不连通。

同理测试 PC1 与 PC2 的连通性:



```
PC1
Physical Config Desktop Custom Interface
Command Prompt
Approximate round trip times in milli-seconds:
  Minimum = 3ms, Maximum = 7ms, Average = 4ms

PC>
PC>
PC>
PC>
PC>
PC>
PC>
PC>
PC>
PC>
PC>ping 192.168.3.2

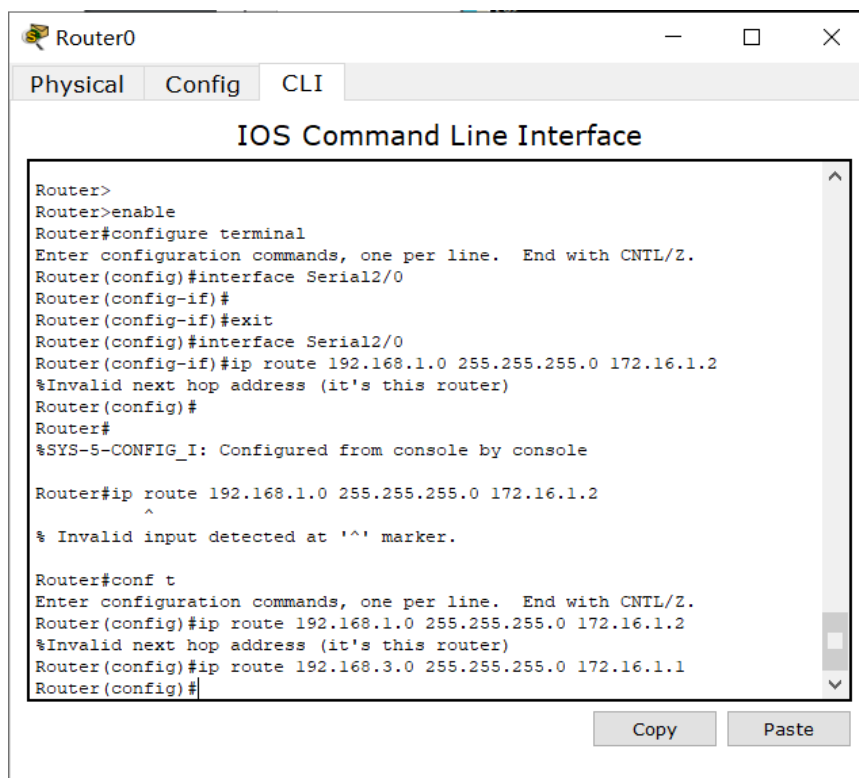
Pinging 192.168.3.2 with 32 bytes of data:

Reply from 192.168.2.1: Destination host unreachable.
Reply from 192.168.2.1: Destination host unreachable.
Reply from 192.168.2.1: Destination host unreachable.
Request timed out.

Ping statistics for 192.168.3.2:
    Packets: Sent = 4, Received = 0, Lost = 4 (100% loss),
PC>
```

也是 100% loss, 说明 PC1 与 PC2 也不连通。

分析原因很简单: 不在同一个网段。



```
Router0
Physical Config CLI
IOS Command Line Interface

Router>
Router>enable
Router#configure terminal
Enter configuration commands, one per line. End with CNTL/Z.
Router(config)#interface Serial2/0
Router(config-if)#
Router(config-if)#exit
Router(config)#interface Serial2/0
Router(config-if)#ip route 192.168.1.0 255.255.255.0 172.16.1.2
%Invalid next hop address (it's this router)
Router(config)#
Router#
%SYS-5-CONFIG_I: Configured from console by console

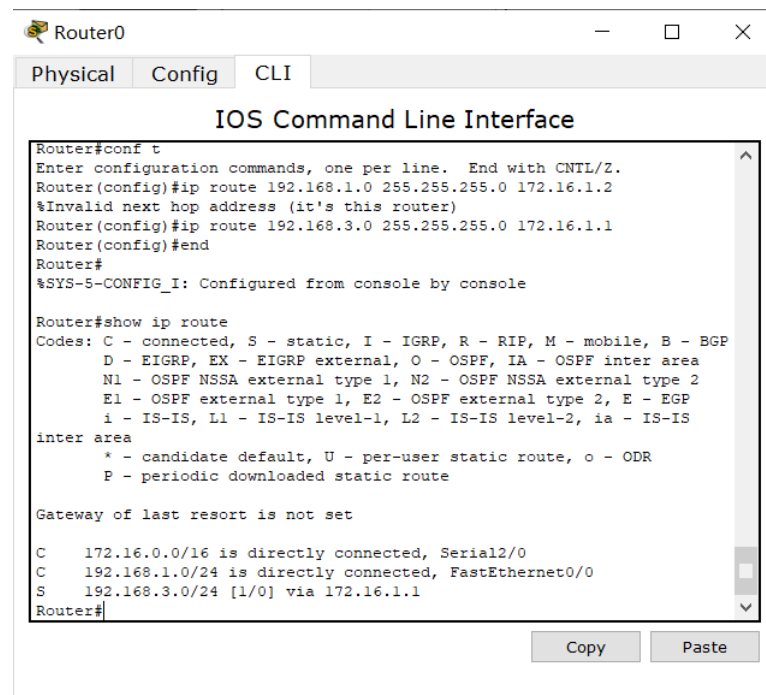
Router#ip route 192.168.1.0 255.255.255.0 172.16.1.2
^
% Invalid input detected at '^' marker.

Router#conf t
Enter configuration commands, one per line. End with CNTL/Z.
Router(config)#ip route 192.168.1.0 255.255.255.0 172.16.1.2
%Invalid next hop address (it's this router)
Router(config)#ip route 192.168.3.0 255.255.255.0 172.16.1.1
Router(config)#
```

## 2. 静态路由的配置

首先进入到 Router(config)# 的模式下，我们在 Router# 下输入 conf t 即可，然后再输入如下命令：Router(config)#ip route 192.168.3.0 255.255.255.0 172.16.1.1。

然后查看 Router0 的路由表进行检查



```
Router0
Physical Config CLI
IOS Command Line Interface
Router#conf t
Enter configuration commands, one per line. End with CNTL/Z.
Router(config)#ip route 192.168.1.0 255.255.255.0 172.16.1.2
%Invalid next hop address (it's this router)
Router(config)#ip route 192.168.3.0 255.255.255.0 172.16.1.1
Router(config)#end
Router#
%SYS-5-CONFIG_I: Configured from console by console

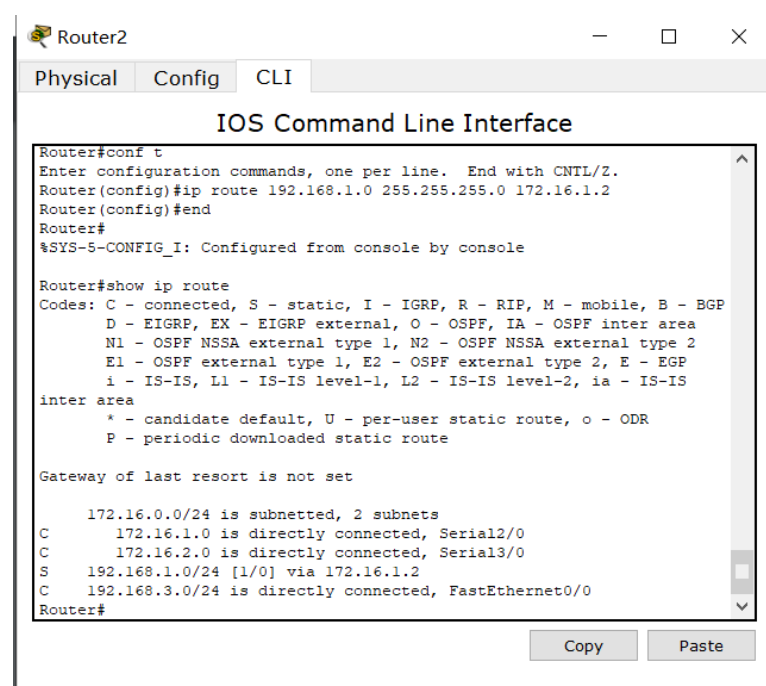
Router#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, ia - IS-IS
       inter area
       * - candidate default, U - per-user static route, o - ODR
       P - periodic downloaded static route

Gateway of last resort is not set

C    172.16.0.0/16 is directly connected, Serial2/0
C    192.168.1.0/24 is directly connected, FastEthernet0/0
S    192.168.3.0/24 [1/0] via 172.16.1.1
Router#
```

发现对比之前的路由表多了一条静态路由 S

同理在 Router2 上也添加一条静态路由并查看路由表：



```
Router2
Physical Config CLI
IOS Command Line Interface
Router#conf t
Enter configuration commands, one per line. End with CNTL/Z.
Router(config)#ip route 192.168.1.0 255.255.255.0 172.16.1.2
Router(config)#end
Router#
%SYS-5-CONFIG_I: Configured from console by console

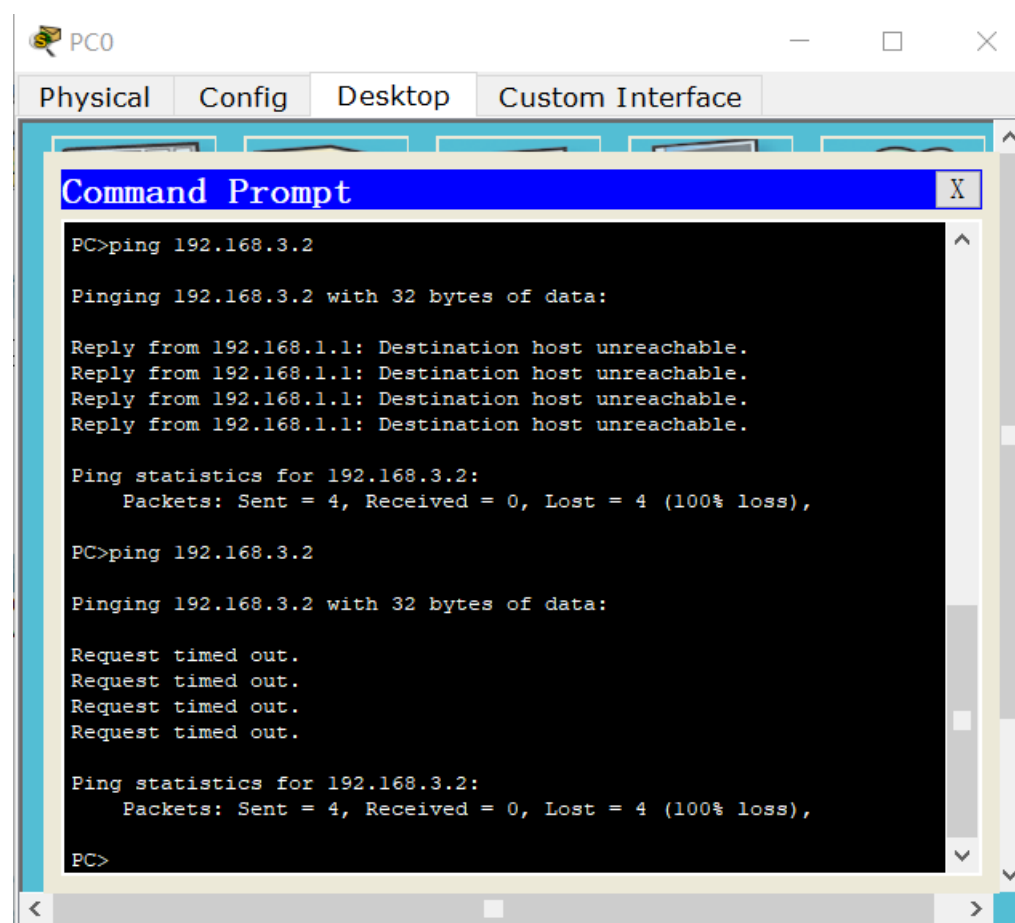
Router#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, ia - IS-IS
       inter area
       * - candidate default, U - per-user static route, o - ODR
       P - periodic downloaded static route

Gateway of last resort is not set

      172.16.0.0/24 is subnetted, 2 subnets
C      172.16.1.0 is directly connected, Serial2/0
C      172.16.2.0 is directly connected, Serial3/0
S    192.168.1.0/24 [1/0] via 172.16.1.2
C    192.168.3.0/24 is directly connected, FastEthernet0/0
Router#
```

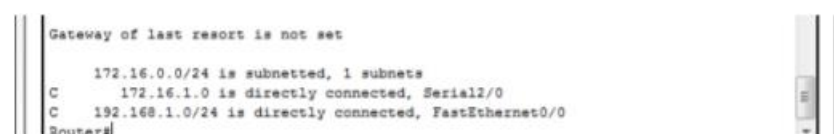
完成两条静态路由的配置后，我们重新测试主机之间的连通性：

PC0 与 PC2 连通性如下：

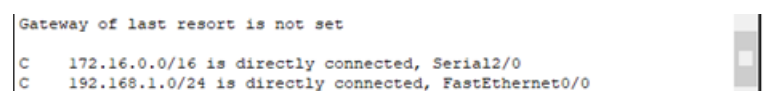


发现居然还是不连通，这一定是出错了。于是我从头开始逐一检查，发现了原因是 Router 的直连路由不对：

指导书上第一步配置好后 Router0 的路由表应当是：

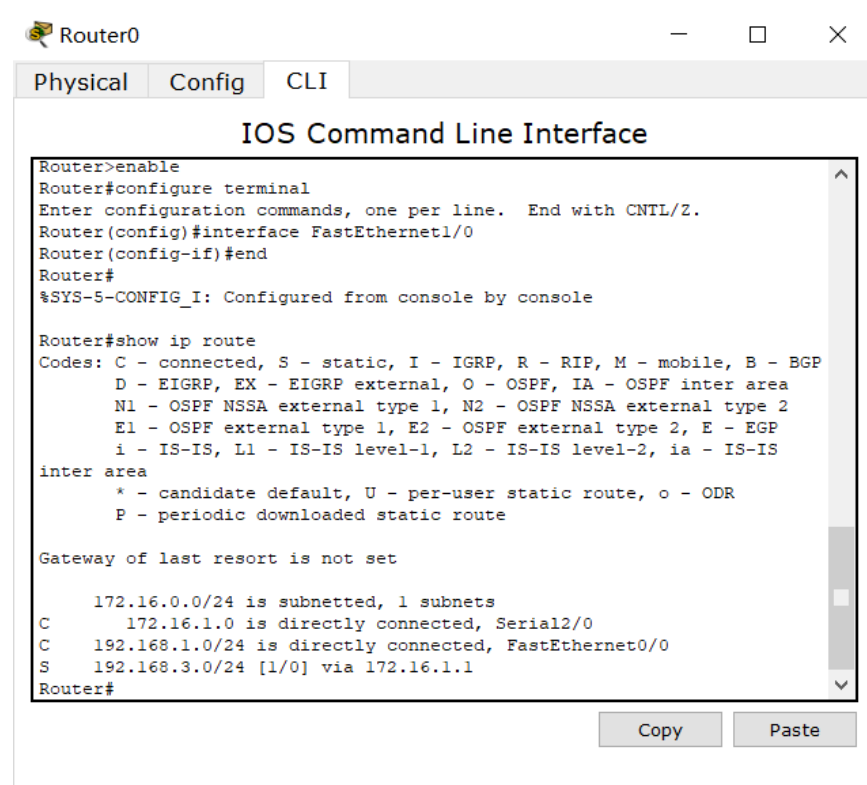


对比我的



虽然都是两条直连路由，但是仔细观察就会发现我的少了一句“172.16.0.0/24 is subnetted, 1 subnets”，且第一个直连路由是 C 172.16.0.0/16 而不是指导书里面的 C 172.16.1.0。也就是说我将本来应当作为子网的 172.16.0.0/16 直接作为了直连路由？

就这样，我就在这卡了很久，始终找不到问题的原因，甚至重新进行了一遍实验都不行，最后无奈之下，我重启了思科的这个模拟器，然后再次试着查看了一下 Router0 的路由表，居然解决了？！



The screenshot shows the Router0 CLI window with the 'CLI' tab selected. The title bar says 'Router0'. The window contains the following text:

```
Router>enable
Router#configure terminal
Enter configuration commands, one per line. End with CNTL/Z.
Router(config)#interface FastEthernet1/0
Router(config-if)#end
Router#
%SYS-5-CONFIG_I: Configured from console by console

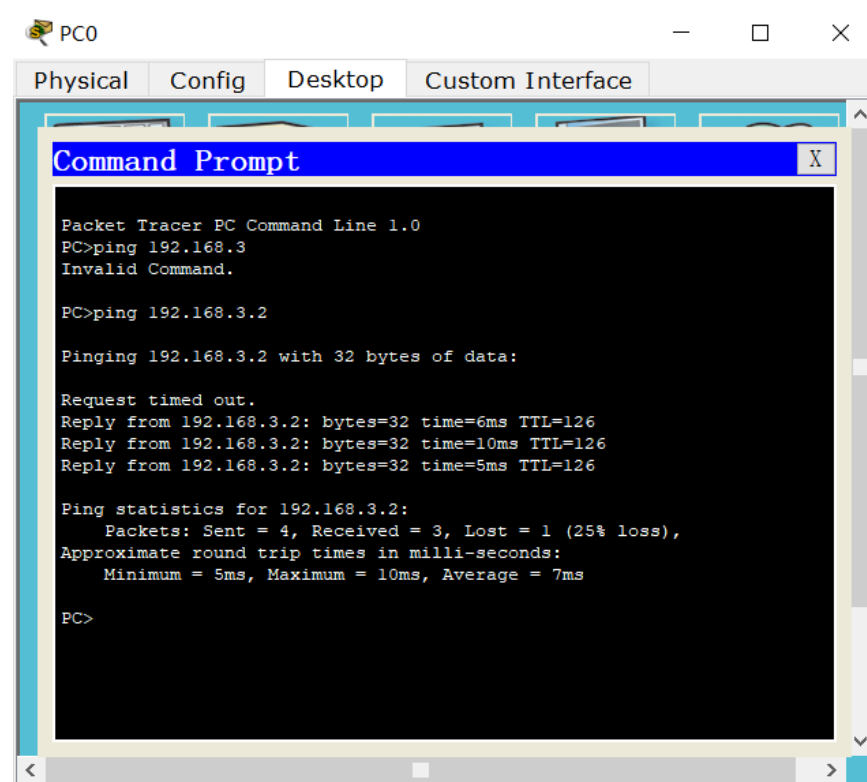
Router#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, ia - IS-IS
       inter area
       * - candidate default, U - per-user static route, o - ODR
       P - periodic downloaded static route

Gateway of last resort is not set

      172.16.0.0/24 is subnetted, 1 subnets
C       172.16.1.0 is directly connected, Serial2/0
C    192.168.1.0/24 is directly connected, FastEthernet0/0
S    192.168.3.0/24 [1/0] via 172.16.1.1
Router#
```

At the bottom right of the window are 'Copy' and 'Paste' buttons.

接下来测试 PC0 与 PC2 的连通性，



The screenshot shows the PC0 Command Prompt window with the 'Desktop' tab selected. The title bar says 'PC0'. The window contains the following text:

```
Packet Tracer PC Command Line 1.0
PC>ping 192.168.3
Invalid Command.

PC>ping 192.168.3.2

Pinging 192.168.3.2 with 32 bytes of data:

Request timed out.
Reply from 192.168.3.2: bytes=32 time=6ms TTL=126
Reply from 192.168.3.2: bytes=32 time=10ms TTL=126
Reply from 192.168.3.2: bytes=32 time=5ms TTL=126

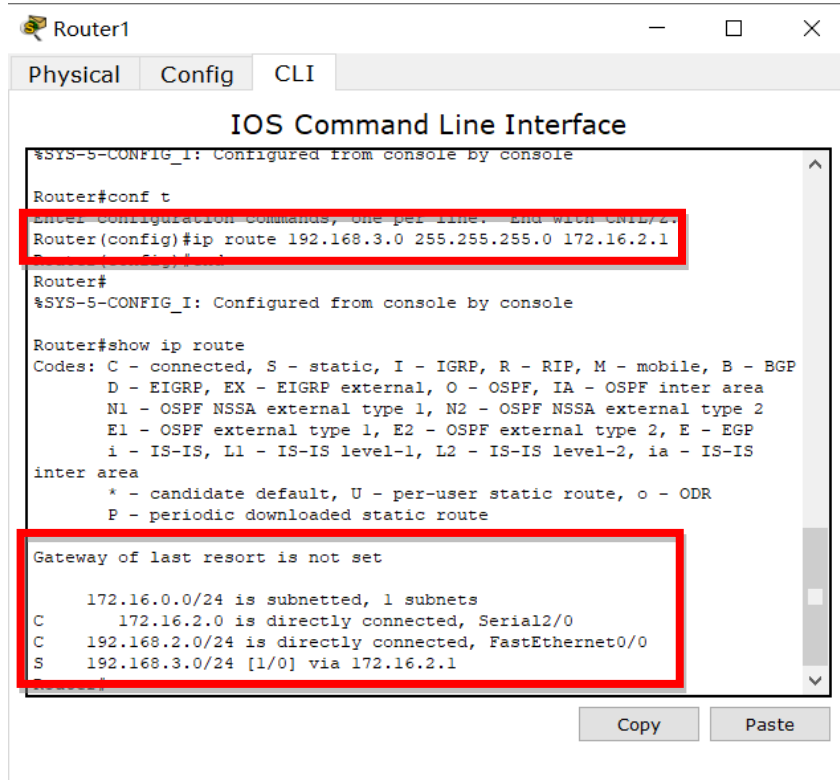
Ping statistics for 192.168.3.2:
    Packets: Sent = 4, Received = 3, Lost = 1 (25% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 5ms, Maximum = 10ms, Average = 7ms

PC>
```

问题解决了，自然也就连通了。

接下来继续对 Router1 和 Router2 添加静态路由以实现 PC1 和 PC2 的连通性：

在 Router1 中添加静态路由并查看路由表：



The screenshot shows the Router1 CLI interface with the 'CLI' tab selected. The title bar says 'Router1'. The main window is titled 'IOS Command Line Interface'. The command history shows the following commands and outputs:

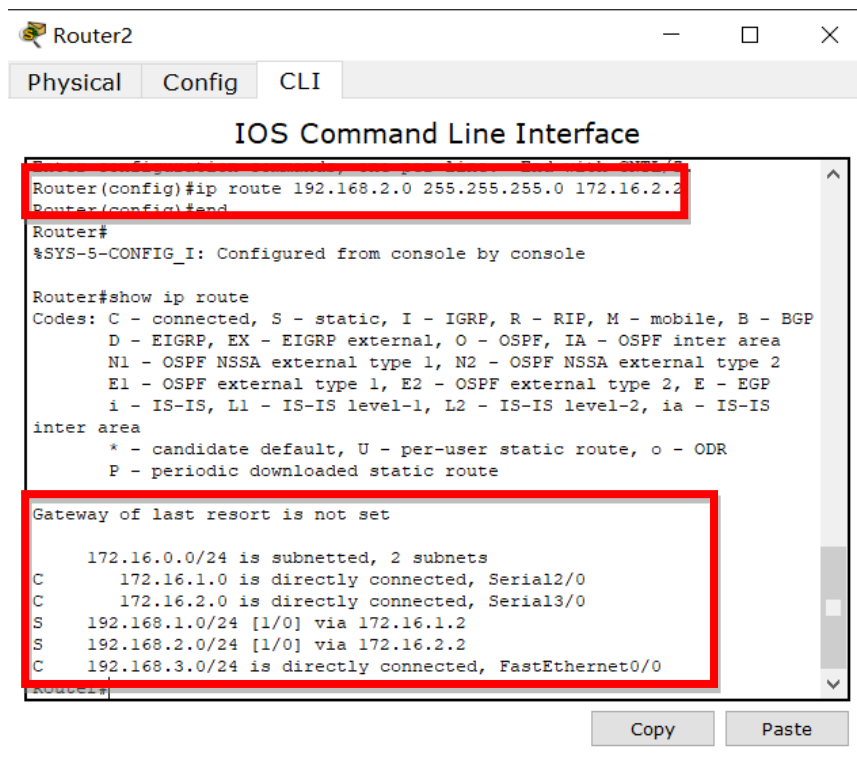
```
%SYS-5-CONFIG_I: Configured from console by console
Router#conf t
Enter configuration commands, one per line. End with CNTRL-Z.
Router(config)#ip route 192.168.3.0 255.255.255.0 172.16.2.1
Router(config)#end
Router#
%SYS-5-CONFIG_I: Configured from console by console
Router#show ip route
Codes: C - 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, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, ia - IS-IS
       inter area
       * - candidate default, U - per-user static route, o - ODR
       P - periodic downloaded static route

Gateway of last resort is not set

      172.16.0.0/24 is subnetted, 1 subnets
C        172.16.2.0 is directly connected, Serial2/0
C    192.168.2.0/24 is directly connected, FastEthernet0/0
S    192.168.3.0/24 [1/0] via 172.16.2.1
```

At the bottom of the window, there are 'Copy' and 'Paste' buttons.

在 Router2 中添加静态路由并查看路由表：



The screenshot shows the Router2 CLI interface with the 'CLI' tab selected. The title bar says 'Router2'. The main window is titled 'IOS Command Line Interface'. The command history shows the following commands and outputs:

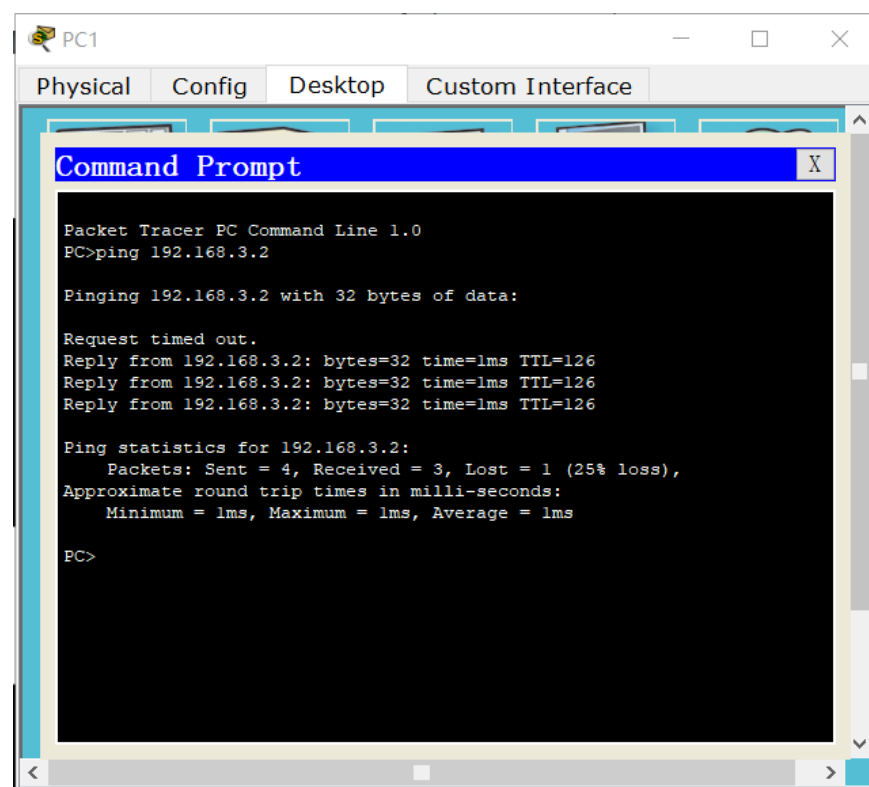
```
%SYS-5-CONFIG_I: Configured from console by console
Router(config)#ip route 192.168.2.0 255.255.255.0 172.16.2.2
Router(config)#end
Router#
%SYS-5-CONFIG_I: Configured from console by console
Router#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - 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, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, ia - IS-IS
       inter area
       * - candidate default, U - per-user static route, o - ODR
       P - periodic downloaded static route

Gateway of last resort is not set

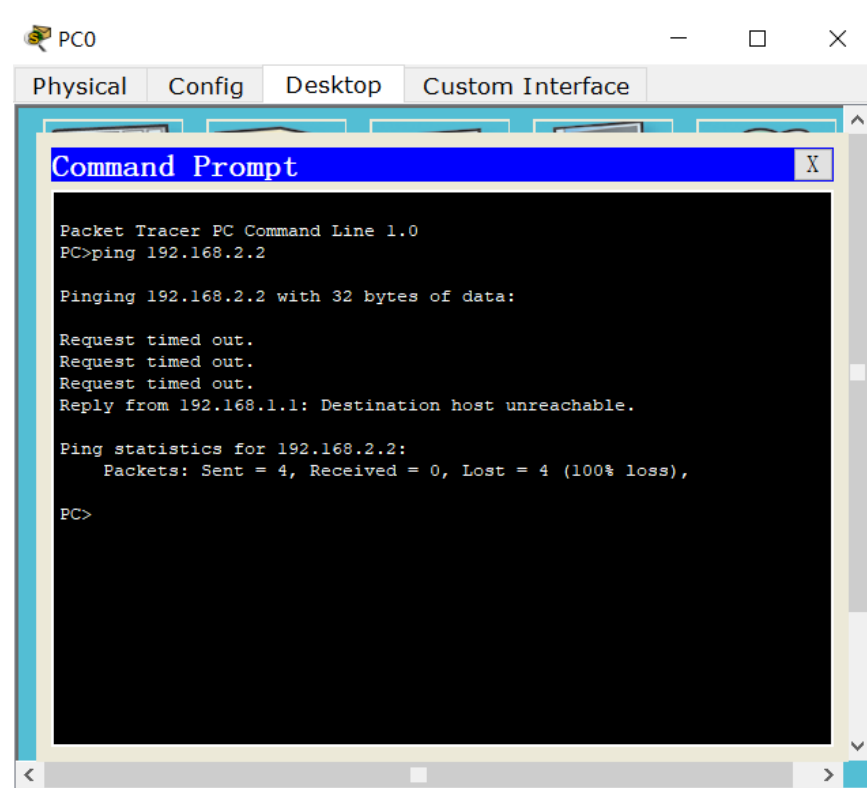
      172.16.0.0/24 is subnetted, 2 subnets
C        172.16.1.0 is directly connected, Serial2/0
C        172.16.2.0 is directly connected, Serial3/0
S    192.168.1.0/24 [1/0] via 172.16.1.2
S    192.168.2.0/24 [1/0] via 172.16.2.2
C    192.168.3.0/24 is directly connected, FastEthernet0/0
```

At the bottom of the window, there are 'Copy' and 'Paste' buttons.

然后测试 PC1 与 PC2 的连通性：



上面已经知道 PC0 与 PC2 互通，PC1 与 PC2 互通，接下来测试 PC0 与 PC1：



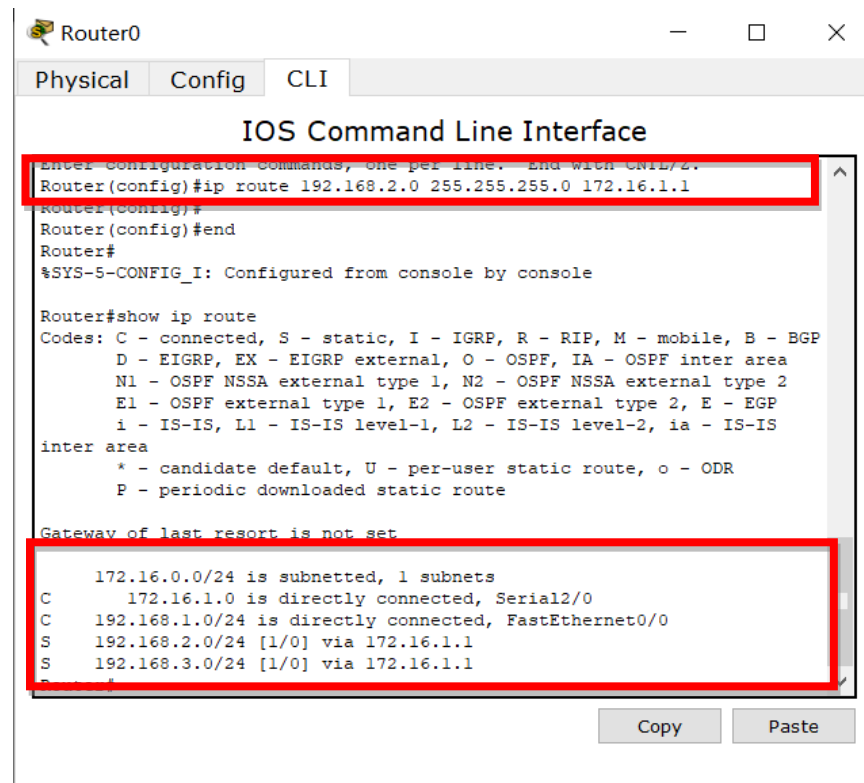
不连通，究其原因原因是因 Router0 的路由表中不存在到达 PC1 所在网络

(192.168.2.0) 的路由，查看 Router2 的路由表可知，其中不存在到达 PC0 所

在网络（192.168.1.0）的路由，因此 PC0 与 PC1 的连通性为不通。

为了使之连通，我们要对 Router0 与 Router1 中添加静态路由并查看路由表：

Router0 中添加静态路由并查看路由表：



The screenshot shows the Router0 CLI interface with the 'CLI' tab selected. The title bar says 'Router0'. The main window is titled 'IOS Command Line Interface'. The command prompt is 'Router(config)#'. The command 'ip route 192.168.2.0 255.255.255.0 172.16.1.1' has been entered and is highlighted with a red box. Below the command, the prompt changes to 'Router(config)#end' and then 'Router#'. The command 'show ip route' is entered, and the output is displayed. The output shows the routing table for Router0, including the static route for 192.168.2.0/24 via 172.16.1.1. The output is also highlighted with a red box. The 'Copy' and 'Paste' buttons are visible at the bottom right.

```
Router0
Physical Config CLI
IOS Command Line Interface
Enter configuration commands, one per line. End with CNTRL-Z.
Router(config)#ip route 192.168.2.0 255.255.255.0 172.16.1.1
Router(config)#
Router(config)#end
Router#
%SYS-5-CONFIG_I: Configured from console by console

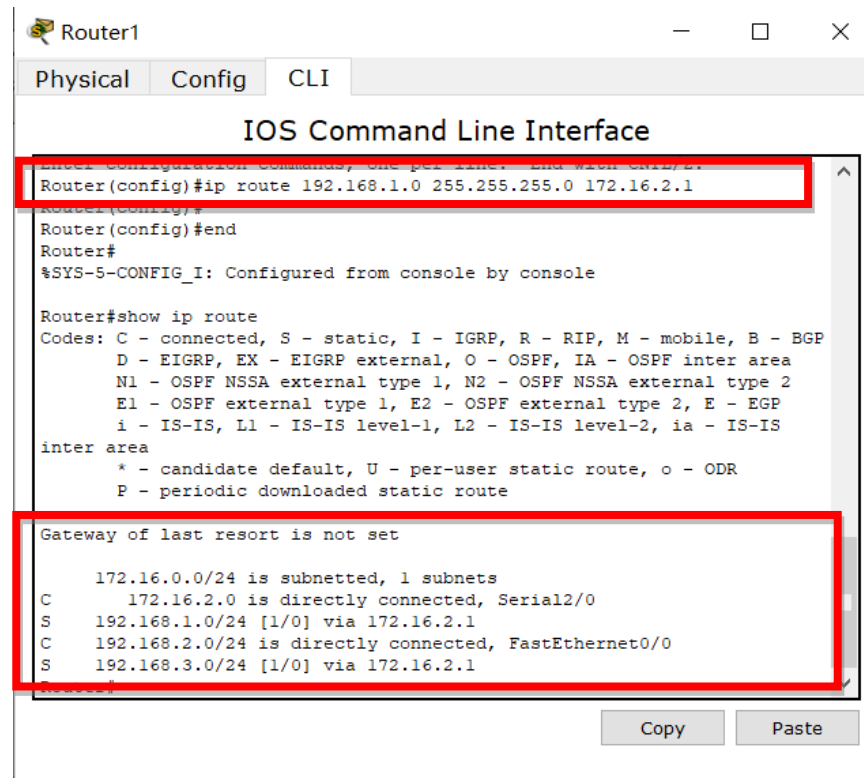
Router#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, ia - IS-IS
       inter area
       * - candidate default, U - per-user static route, o - ODR
       P - periodic downloaded static route

Gateway of last resort is not set

      172.16.0.0/24 is subnetted, 1 subnets
C      172.16.1.0 is directly connected, Serial2/0
C      192.168.1.0/24 is directly connected, FastEthernet0/0
S      192.168.2.0/24 [1/0] via 172.16.1.1
S      192.168.3.0/24 [1/0] via 172.16.1.1

Copy Paste
```

Router1 中添加静态路由并查看路由表



The screenshot shows the Router1 CLI interface with the 'CLI' tab selected. The title bar says 'Router1'. The main window is titled 'IOS Command Line Interface'. The command prompt is 'Router(config)#'. The command 'ip route 192.168.1.0 255.255.255.0 172.16.2.1' has been entered and is highlighted with a red box. Below the command, the prompt changes to 'Router(config)#end' and then 'Router#'. The command 'show ip route' is entered, and the output is displayed. The output shows the routing table for Router1, including the static route for 192.168.1.0/24 via 172.16.2.1. The output is also highlighted with a red box. The 'Copy' and 'Paste' buttons are visible at the bottom right.

```
Router1
Physical Config CLI
IOS Command Line Interface
Enter configuration commands, one per line. End with CNTRL-Z.
Router(config)#ip route 192.168.1.0 255.255.255.0 172.16.2.1
Router(config)#
Router(config)#end
Router#
%SYS-5-CONFIG_I: Configured from console by console

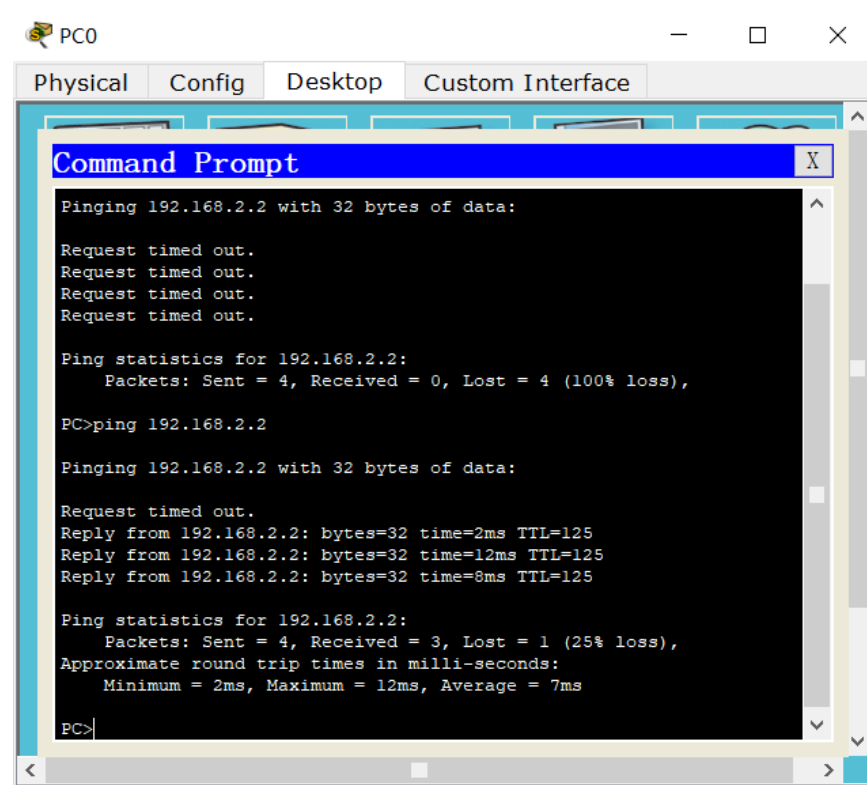
Router#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, ia - IS-IS
       inter area
       * - candidate default, U - per-user static route, o - ODR
       P - periodic downloaded static route

Gateway of last resort is not set

      172.16.0.0/24 is subnetted, 1 subnets
C      172.16.2.0 is directly connected, Serial2/0
S      192.168.1.0/24 [1/0] via 172.16.2.1
C      192.168.2.0/24 is directly connected, FastEthernet0/0
S      192.168.3.0/24 [1/0] via 172.16.2.1

Copy Paste
```

然后测试 PC0 与 PC1 的连通性：



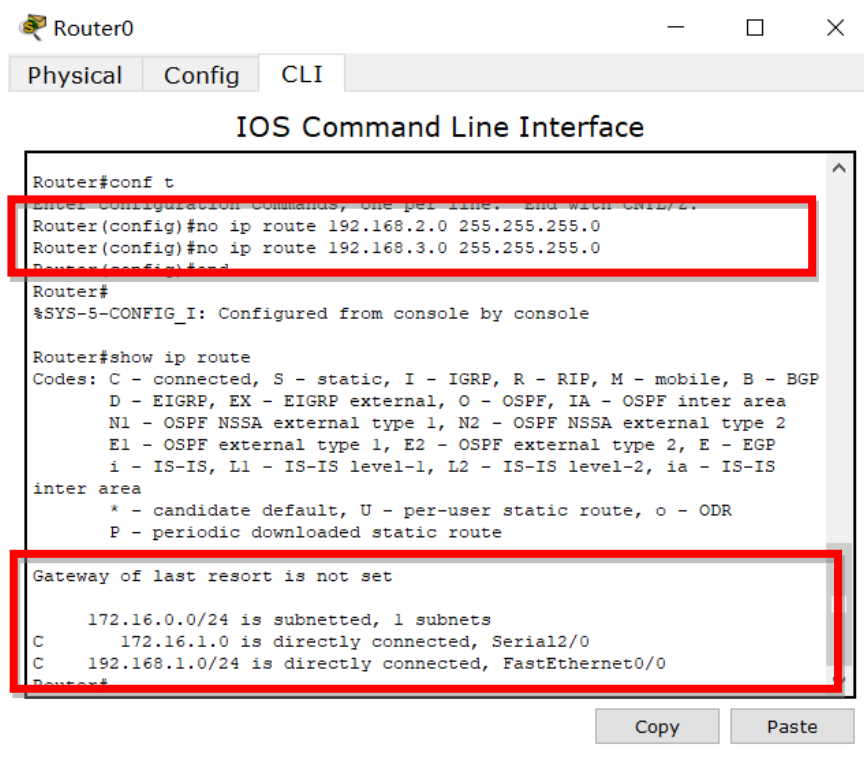
此时已经连通。

### 3. 设置默认路由

之所以要设置默认路由，是因为 PC0 想要访问 PC1 和 PC2 所在网络，需要在 Router0 中添加两条静态路由。这两条静态路由的下一跳 IP 地址相同，并且 Router0 所在网络只有一条通路连接其它网络。这种情况下，如果使用默认路由，则 Router0 只需设置一条默认路由就可使 PC0 可访问 PC1 和 PC2，这样 Router0 的路由表将更加简单。



首先要删除 Router0 的静态路由，然后再查看路由表检查是否删除成功：



The screenshot shows the Router0 CLI interface with the 'CLI' tab selected. The title bar says 'Router0'. The main window is titled 'IOS Command Line Interface'. The command history shows the following sequence of commands and outputs:

```
Router#conf t
Enter Configuration Commands, one per line. End with CNTRL-Z.
Router(config)#no ip route 192.168.2.0 255.255.255.0
Router(config)#no ip route 192.168.3.0 255.255.255.0
Router(config)#end
Router#
%SYS-5-CONFIG_I: Configured from console by console

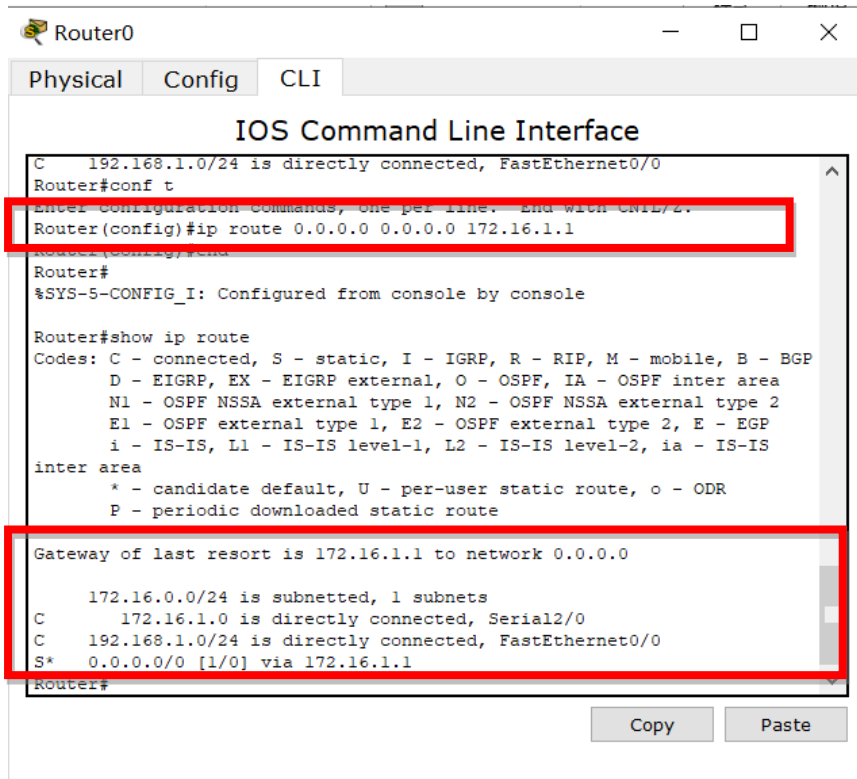
Router#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, ia - IS-IS
       inter area
       * - candidate default, U - per-user static route, o - ODR
       P - periodic downloaded static route

Gateway of last resort is not set

      172.16.0.0/24 is subnetted, 1 subnets
C       172.16.1.0 is directly connected, Serial2/0
C       192.168.1.0/24 is directly connected, FastEthernet0/0
Router#
```

Two red boxes highlight the configuration commands and the resulting routing table output. The 'Copy' and 'Paste' buttons are visible at the bottom right.

然后在 Router0 中添加默认路由并查看路由表：



The screenshot shows the Router0 CLI interface with the 'CLI' tab selected. The title bar says 'Router0'. The main window is titled 'IOS Command Line Interface'. The command history shows the following sequence of commands and outputs:

```
C       192.168.1.0/24 is directly connected, FastEthernet0/0
Router#conf t
Enter Configuration Commands, one per line. End with CNTRL-Z.
Router(config)#ip route 0.0.0.0 0.0.0.0 172.16.1.1
Router(config)#end
Router#
%SYS-5-CONFIG_I: Configured from console by console

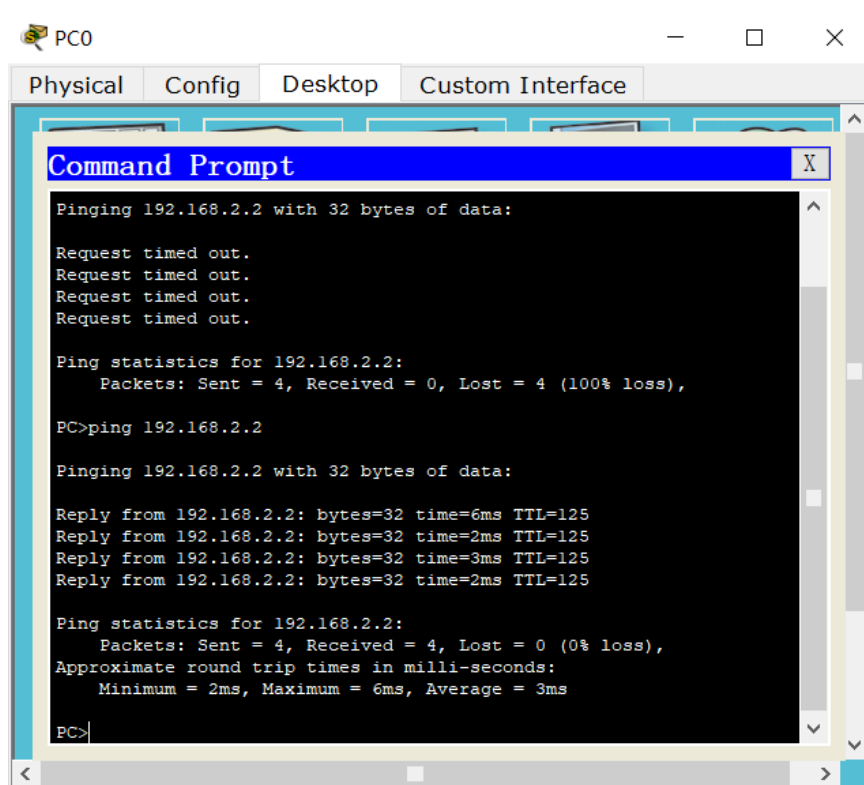
Router#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, ia - IS-IS
       inter area
       * - candidate default, U - per-user static route, o - ODR
       P - periodic downloaded static route

Gateway of last resort is 172.16.1.1 to network 0.0.0.0

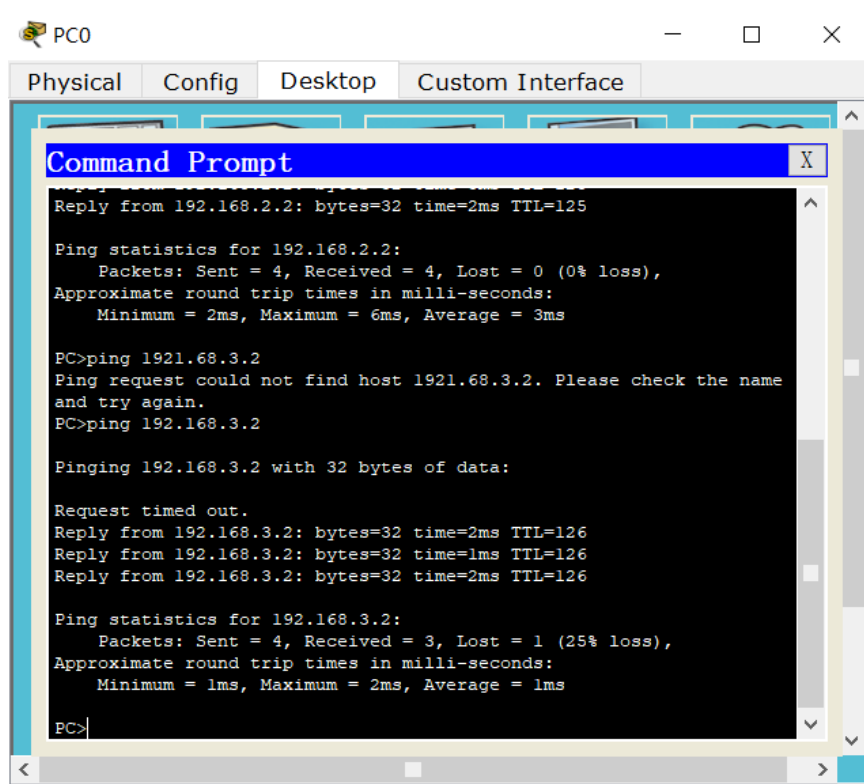
      172.16.0.0/24 is subnetted, 1 subnets
C       172.16.1.0 is directly connected, Serial2/0
C       192.168.1.0/24 is directly connected, FastEthernet0/0
S*     0.0.0.0/0 [1/0] via 172.16.1.1
Router#
```

Two red boxes highlight the configuration command and the resulting routing table output. The 'Copy' and 'Paste' buttons are visible at the bottom right.

随后测试 PC0 与 PC1 的连通性：



然后是 PC0 与 PC2 的连通性：



结果是都连通。

#### 4. 动态路由协议 RIP 的配置

在此之前先把三个路由器中的静态路由和默认路由全部删除，具体过程如下：

首先分别查看三个路由器 Router0、Router1、Router2 的路由表，确认哪些是除了直连路由之外的静态路由和默认路由。

Router0:

```
Gateway of last resort is 172.16.1.1 to network 0.0.0.0

    172.16.0.0/24 is subnetted, 1 subnets
C       172.16.1.0 is directly connected, Serial2/0
C       192.168.1.0/24 is directly connected, FastEthernet0/0
S*      0.0.0.0/0 [1/0] via 172.16.1.1
```

有一条默认路由

Router1:

```
Gateway of last resort is not set

    172.16.0.0/24 is subnetted, 1 subnets
C       172.16.2.0 is directly connected, Serial2/0
S       192.168.1.0/24 [1/0] via 172.16.2.1
C       192.168.2.0/24 is directly connected, FastEthernet0/0
S       192.168.3.0/24 [1/0] via 172.16.2.1
```

有两条静态路由

```
Gateway of last resort is not set

    172.16.0.0/24 is subnetted, 2 subnets
C       172.16.1.0 is directly connected, Serial2/0
C       172.16.2.0 is directly connected, Serial3/0
S       192.168.1.0/24 [1/0] via 172.16.1.2
S       192.168.2.0/24 [1/0] via 172.16.2.2
C       192.168.3.0/24 is directly connected, FastEthernet0/0
```

Router2:

有两条静态路由

确认好要删除的对象后，接下来就是挨个删除：

删除 Router0 的默认路由后的路由表：

```
Gateway of last resort is not set

C       172.16.0.0/16 is directly connected, Serial2/0
C       192.168.1.0/24 is directly connected, FastEthernet0/0
Router0#
```

删除 Router1 的默认路由后的路由表：

```
Gateway of last resort is not set

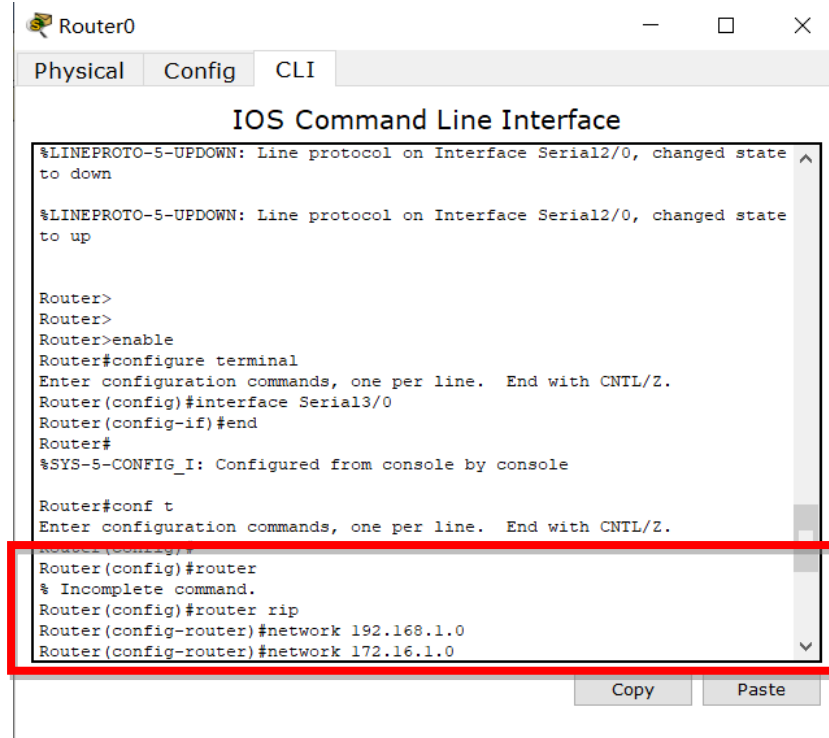
    172.16.0.0/24 is subnetted, 1 subnets
C       172.16.2.0 is directly connected, Serial2/0
C       192.168.2.0/24 is directly connected, FastEthernet0/0
Router1#
```

删除 Router2 的默认路由后的路由表：

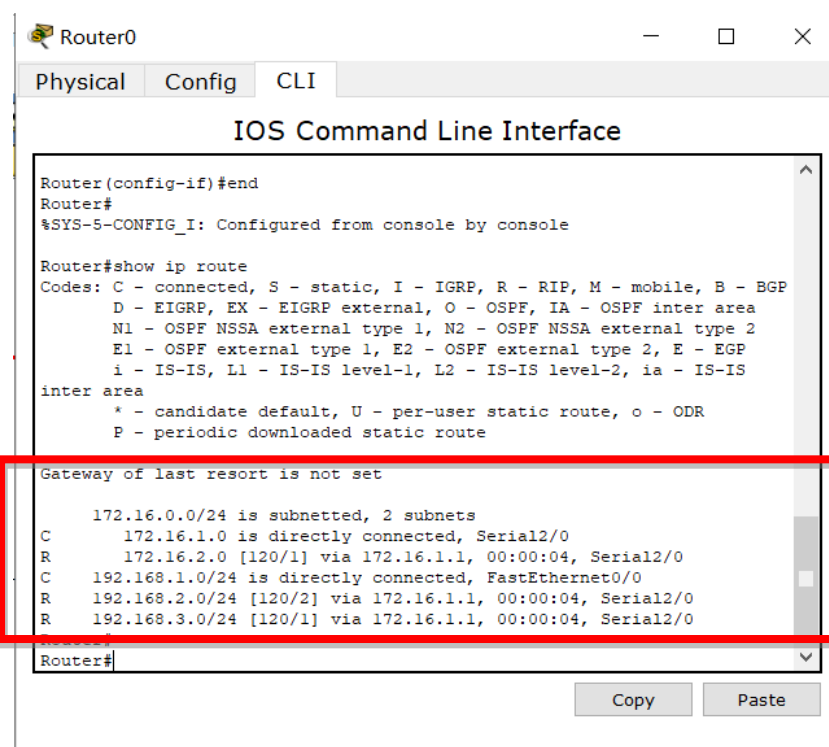
```
Gateway of last resort is not set

172.16.0.0/24 is subnetted, 2 subnets
C    172.16.1.0 is directly connected, Serial2/0
C    172.16.2.0 is directly connected, Serial3/0
C    192.168.3.0/24 is directly connected, FastEthernet0/0
Router#
```

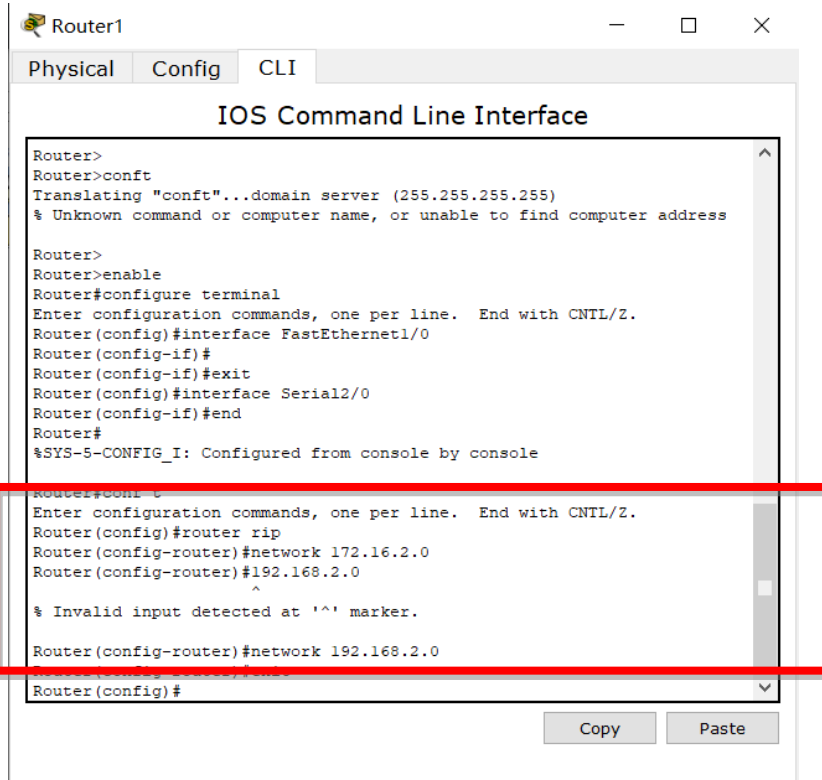
此时路由表中皆仅有直连路由 C，然后我们在 Router0 上配置 RIP 协议，



查看其路由表：

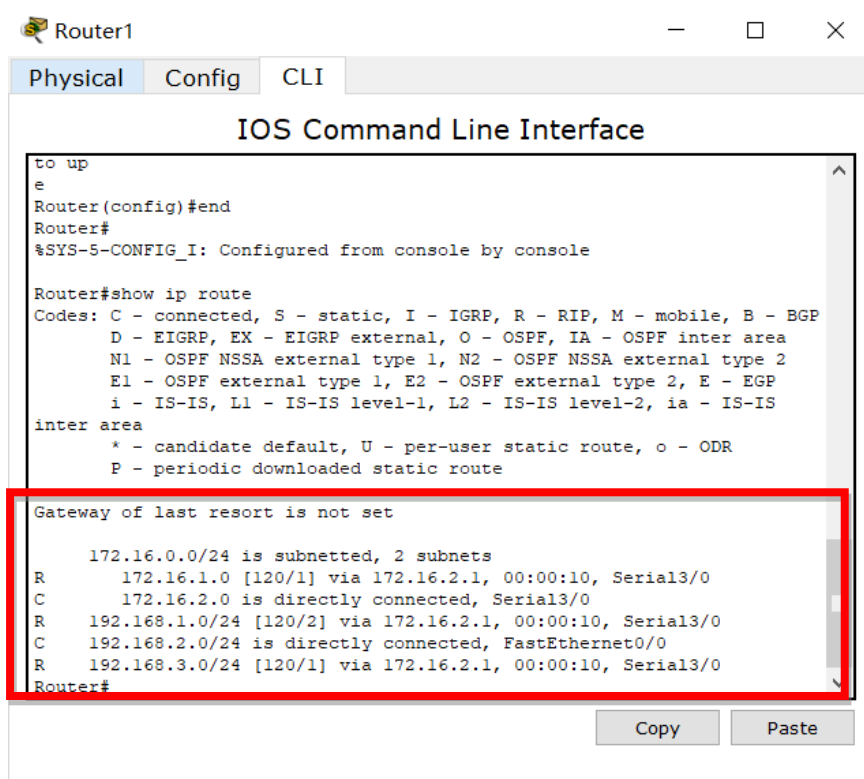


同理，在 Router1 上配置 RIP 协议，命令如下：



```
Router1
Physical Config CLI
IOS Command Line Interface
Router>
Router>conf t
Translating "conf t"...domain server (255.255.255.255)
% Unknown command or computer name, or unable to find computer address
Router>
Router>enable
Router#configure terminal
Enter configuration commands, one per line. End with CNTL/Z.
Router(config)#interface FastEthernet1/0
Router(config-if)#
Router(config-if)#exit
Router(config)#interface Serial2/0
Router(config-if)#end
Router#
%SYS-5-CONFIG_I: Configured from console by console
Router#conf t
Enter configuration commands, one per line. End with CNTL/Z.
Router(config)#router rip
Router(config-router)#network 172.16.2.0
Router(config-router)#network 192.168.2.0
^
% Invalid input detected at '^' marker.
Router(config-router)#network 192.168.2.0
Router(config-router)#
Router(config)#
```

查看其路由表：

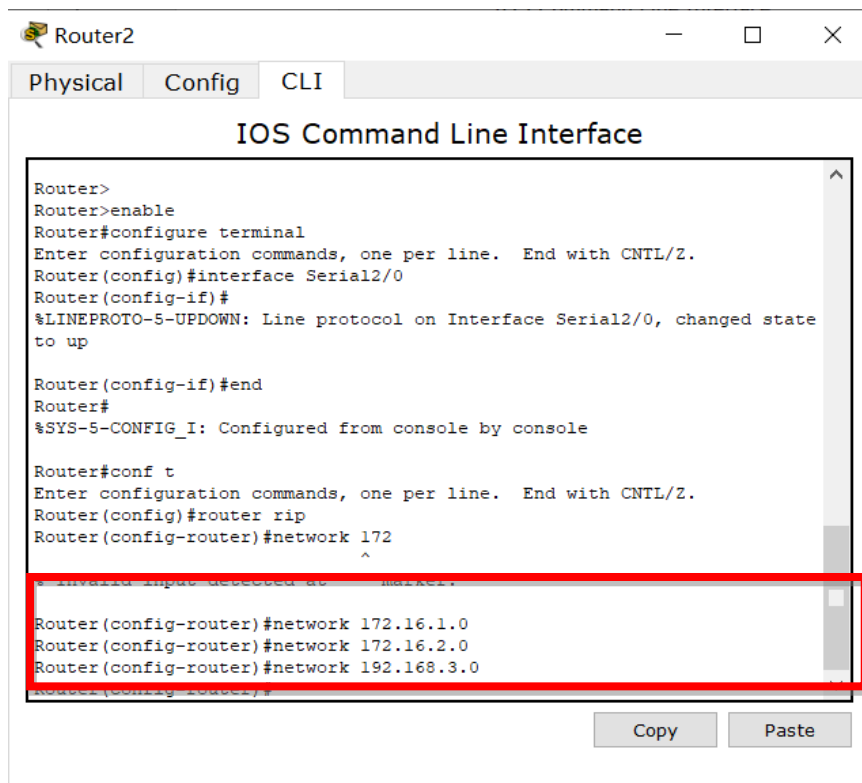


```
Router1
Physical Config CLI
IOS Command Line Interface
to up
e
Router(config)#end
Router#
%SYS-5-CONFIG_I: Configured from console by console
Router#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, ia - IS-IS
       inter area
       * - candidate default, U - per-user static route, o - ODR
       P - periodic downloaded static route

Gateway of last resort is not set

      172.16.0.0/24 is subnetted, 2 subnets
R       172.16.1.0 [120/1] via 172.16.2.1, 00:00:10, Serial3/0
C       172.16.2.0 is directly connected, Serial3/0
R       192.168.1.0/24 [120/2] via 172.16.2.1, 00:00:10, Serial3/0
C       192.168.2.0/24 is directly connected, FastEthernet0/0
R       192.168.3.0/24 [120/1] via 172.16.2.1, 00:00:10, Serial3/0
Router#
```

同理，在 Router2 上配置 RIP 协议，命令如下：

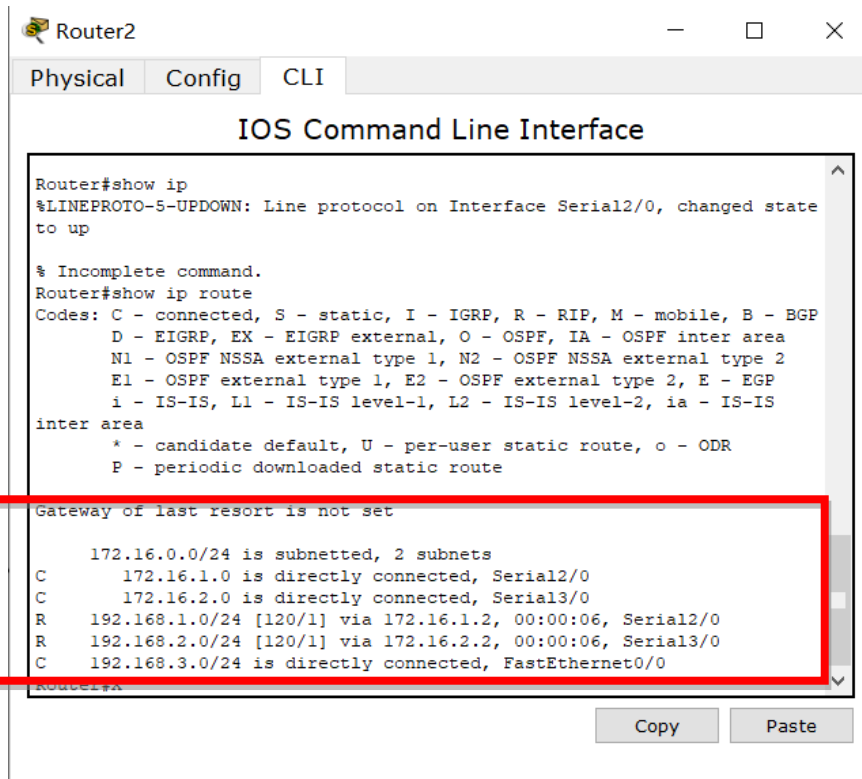


```
Router>
Router>enable
Router#configure terminal
Enter configuration commands, one per line. End with CNTL/Z.
Router(config)#interface Serial2/0
Router(config-if)#
%LINEPROTO-5-UPDOWN: Line protocol on Interface Serial2/0, changed state to up

Router(config-if)#end
Router#
%SYS-5-CONFIG_I: Configured from console by console

Router#conf t
Enter configuration commands, one per line. End with CNTL/Z.
Router(config)#router rip
Router(config-router)#network 172
^
Router(config-router)#network 172.16.1.0
Router(config-router)#network 172.16.2.0
Router(config-router)#network 192.168.3.0
Router(config-router)#
```

查看其路由表：



```
Router#show ip
%LINEPROTO-5-UPDOWN: Line protocol on Interface Serial2/0, changed state to up

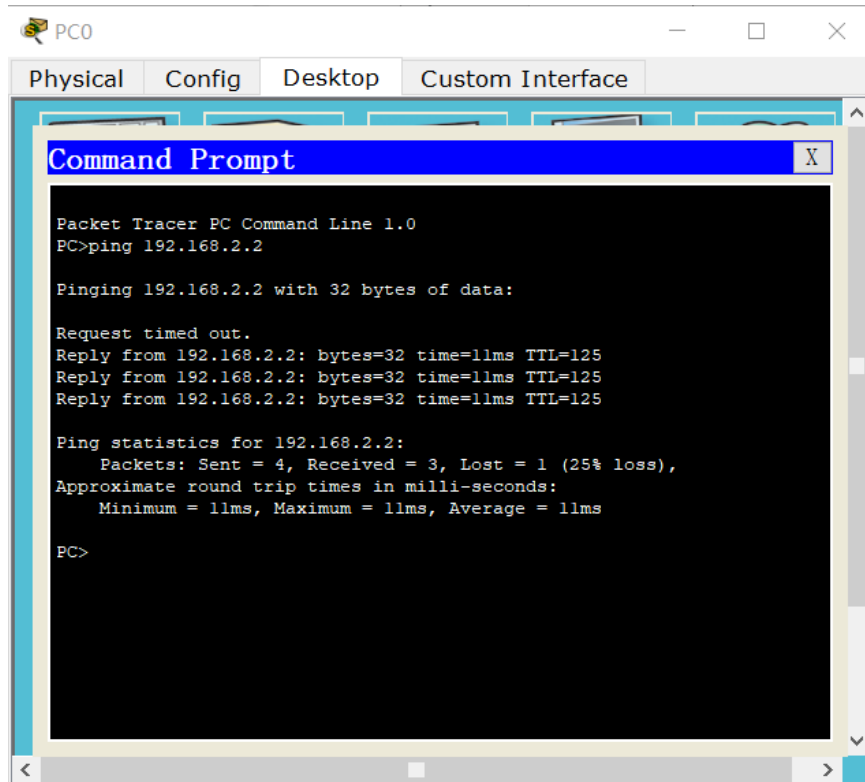
% Incomplete command.
Router#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
       D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
       N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
       E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, ia - IS-IS
       inter area
       * - candidate default, U - per-user static route, o - ODR
       P - periodic downloaded static route

Gateway of last resort is not set

      172.16.0.0/24 is subnetted, 2 subnets
C        172.16.1.0 is directly connected, Serial2/0
C        172.16.2.0 is directly connected, Serial3/0
R        192.168.1.0/24 [120/1] via 172.16.1.2, 00:00:06, Serial2/0
R        192.168.2.0/24 [120/1] via 172.16.2.2, 00:00:06, Serial3/0
C        192.168.3.0/24 is directly connected, FastEthernet0/0
Router#
```

做完这些之后，我们测试一下主机之间的连通性：

首先，在 PC0 处 pingPC1，以此检查 PC0 与 PC1 之间的连通性：



```
PC0
Physical Config Desktop Custom Interface
Command Prompt
Packet Tracer PC Command Line 1.0
PC>ping 192.168.2.2

Pinging 192.168.2.2 with 32 bytes of data:

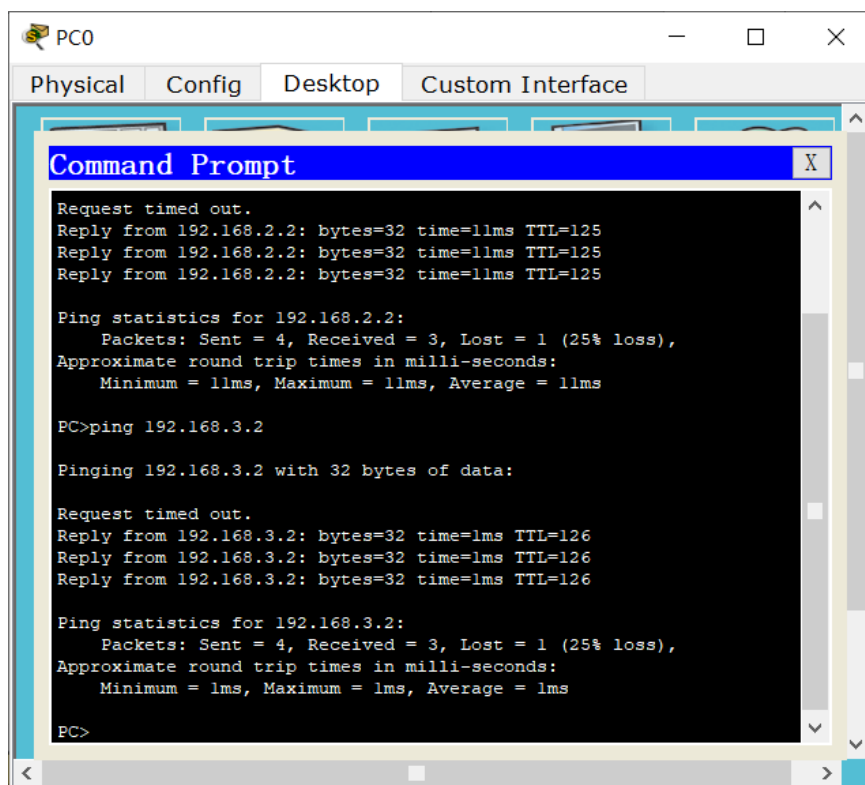
Request timed out.
Reply from 192.168.2.2: bytes=32 time=11ms TTL=125
Reply from 192.168.2.2: bytes=32 time=11ms TTL=125
Reply from 192.168.2.2: bytes=32 time=11ms TTL=125

Ping statistics for 192.168.2.2:
    Packets: Sent = 4, Received = 3, Lost = 1 (25% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 11ms, Maximum = 11ms, Average = 11ms

PC>
```

结果是连通的。(25% loss)

然后在 PC0 处 pingPC2，以此检查 PC0 与 PC2 之间的连通性：



```
PC0
Physical Config Desktop Custom Interface
Command Prompt
Request timed out.
Reply from 192.168.2.2: bytes=32 time=11ms TTL=125
Reply from 192.168.2.2: bytes=32 time=11ms TTL=125
Reply from 192.168.2.2: bytes=32 time=11ms TTL=125

Ping statistics for 192.168.2.2:
    Packets: Sent = 4, Received = 3, Lost = 1 (25% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 11ms, Maximum = 11ms, Average = 11ms

PC>ping 192.168.3.2

Pinging 192.168.3.2 with 32 bytes of data:

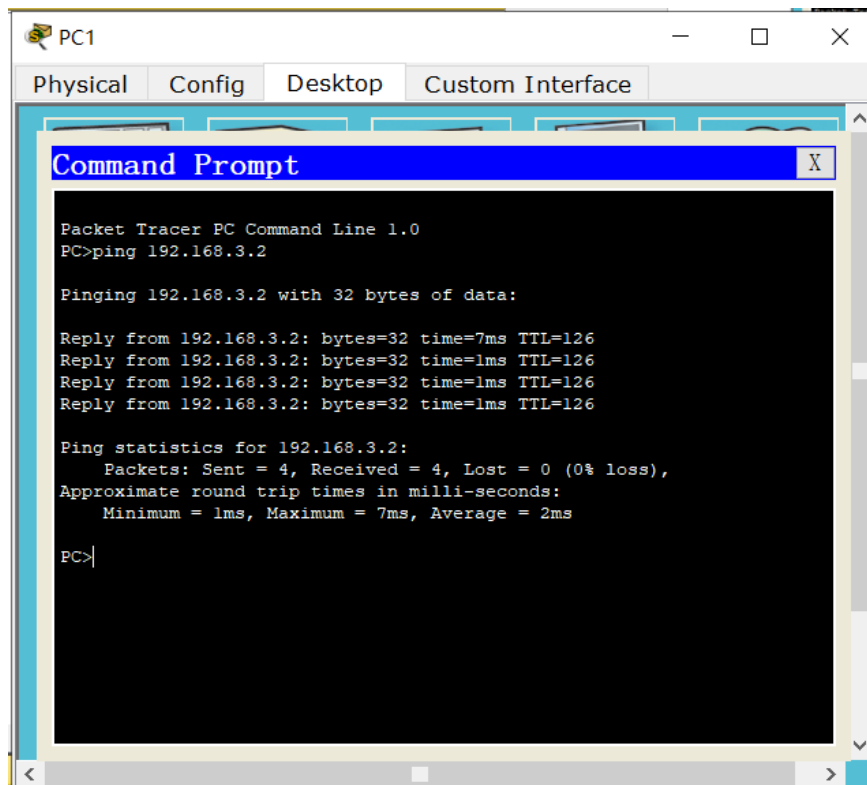
Request timed out.
Reply from 192.168.3.2: bytes=32 time=1ms TTL=126
Reply from 192.168.3.2: bytes=32 time=1ms TTL=126
Reply from 192.168.3.2: bytes=32 time=1ms TTL=126

Ping statistics for 192.168.3.2:
    Packets: Sent = 4, Received = 3, Lost = 1 (25% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 1ms, Maximum = 1ms, Average = 1ms

PC>
```

结果是连通的。(25% loss)

然后我们在 PC1 处 ping PC2, 以此检查 PC1 与 PC2 之间的连通性。



结果是连通的。(0% loss)



## 五、实验总结

由于做这个实验的时候已经离学完计算机网络过去了一个漫长的暑假，计算机网络作为一个理论性极强的课程，几个月不看，感觉对概念里面的细节就会遗忘一些，于是在验收完后，回到实验室我在自己的笔记本上下载好了 Cisco Packet Tracer Student。然后把实验涉及到的定义在网上检索，重新复习了一遍，最后再从头开始做了一遍这个实验并且截图记录，写下了这次的实验报告，这一切让我受益匪浅，对计算机网络里面路由的这部分知识理解又加深了许多，理解了之后记得也更牢固了。

下面我对实验里面涉及到的重要的部分做一些记录，说一些自己的理解：

根据路由器学习路由信息、生成并维护路由表的方法包括直连路由、静态路由和动态路由。

### 1) 直连路由

对路由器而言，无须任何路由配置，即可获得其直连网段的路由。直连路由是指路由器接口直接相连的网段的路由。

### 2) 静态路由

静态路由是一种特殊的路由，由网络管理员采用手工方法，在路由器中配置而成。

缺点是不能动态的反映网络拓扑，当网络发生变化的时候，管理员必须手动的改变路由。但这同时也使得可以精确控制路由选择，改进网络性能。优点是不要动态路由协议参与，这将减少路由器的开销，为重要的应用保证带宽。

如果出于安全的考虑想隐藏网络的某些部分或者管理员想控制数据转发路径也可以使用静态路由，小网络也可以配置静态路由，因为便捷。

### 3) 默认路由

是指路由器在路由表中如果找不到到达目的得具体路由时，最后会采用的路由，默认路由通常会在出口网络中使用，目的 IP 地址和掩码都为 0.0.0.0 的路由为默认路由。

当路由表中的所有路由都选择失败的时候，为使得报文有最终的一个发送地，将使用默认路由。

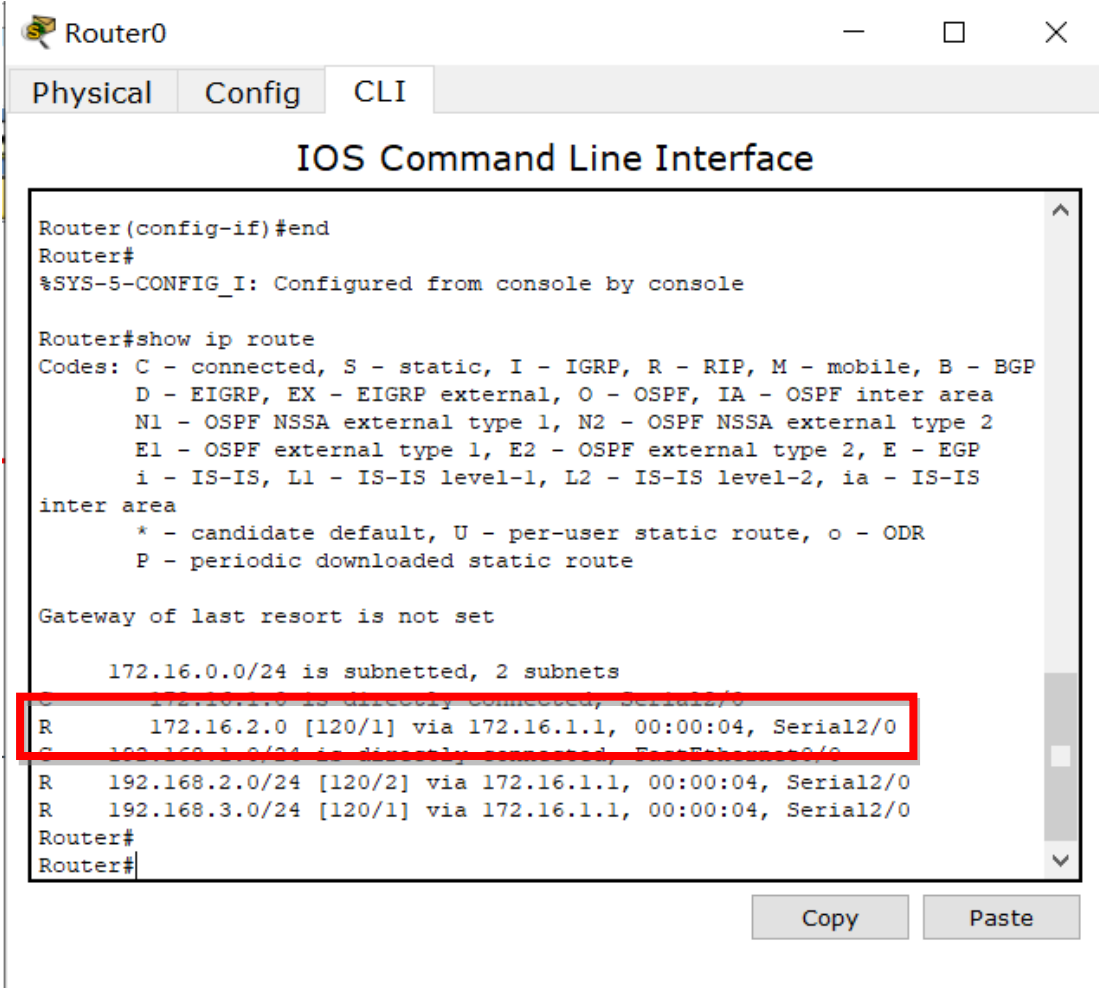
#### 4) 动态路由

也就是由路由协议动态建立的路由，在本实验中具体体现为路由信息协议 RIP，是内部网关协议 IGP 中最先得到广泛使用的协议。RIP 是一种分布式的基于距离向量的路由选择协议，是因特网的标准协议，其最大优点就是实现简单，开销较小。此外我们还学过 OSPF，BGP 动态路由协议。

其特点有：无需管理员手工维护，减轻了管理员的工作负担；占用了网络带宽；在路由器上运行路由协议，使路由器可以自动根据网络拓扑结构的变化调整路由条目；网络规模大、拓扑复杂的网络

验收时，老师要对我对路由表里面做一些解释，当时回答的不是很好，在此我将进行补充解释：

以我们做实验的截图为例，下面是实验第四部分，配置好 RIP 协议后的 Router0 的路由表：



```
Router0
Physical Config CLI
IOS Command Line Interface

Router(config-if)#end
Router#
%SYS-5-CONFIG_I: Configured from console by console

Router#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, E - EGP
       i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, ia - IS-IS
       inter area
       * - candidate default, U - per-user static route, o - ODR
       P - periodic downloaded static route

Gateway of last resort is not set

    172.16.0.0/24 is subnetted, 2 subnets
C       172.16.1.0 is directly connected, Serial2/0
R       172.16.2.0 [120/1] via 172.16.1.1, 00:00:04, Serial2/0
C       192.168.1.0/24 is directly connected, FastEthernet0/0
R       192.168.2.0/24 [120/2] via 172.16.1.1, 00:00:04, Serial2/0
R       192.168.3.0/24 [120/1] via 172.16.1.1, 00:00:04, Serial2/0
Router#
Router#
```

首先我们明确，C 为直连路由，S 为静态路由，R 表示该路由是通过 RIP 协议学习而来。然后我们以红框部分为例：

“172.16.2.0” 是目的网络；“ [120/1] ” 是管理距离/度量值 (Metric)；  
“via 172.16.1.1” 是达目的网络的下一跳路由器的 IP 地址；“00:00:04” 是指路由器最近一次得知路由到现在的时间；“Serial 2/0” 是指到达下一跳应从哪个端口出去。其中，管理距离 (AD) 用来表示路由的可信度，路由器可能从多种途径获得同一路由，路由表中管理距离值越小，说明路由的可靠程度就越高；度量值是 (metric) 某一个路由协议判别到达目的网络的最佳路径的方法。当路由器有多个路径到达某一目的网络，路由协议判断哪一条是最佳的放到路由表中，路由协议会给每一条路径计算出一个数值，这个数值就是度量值，没有单位。度量值越小，路径越佳。不同的路由协议定义的度量值方法不用，选出的最佳路径可能也不一样。

# 合肥工业大学

## 《计算机网络系统实践》报告

设计题目：广播通信设计

学生姓名：孙淼

学    号：2018211958

专业班级：计算机科学与技术 18-2 班

2021 年 3 月

# 目录

第 1 类 基于套接字的网络编程.....	2
设计 1.3 广播通信设计 .....	2
一、设计要求.....	2
二、开发环境与工具.....	5
三、设计原理.....	5
四、系统功能描述及软件模块划分.....	10
五、设计步骤.....	11
1. 服务器模块核心代码.....	12
2. 登陆/注册模块核心代码.....	14
3. 用户模块核心代码.....	16
4. 聊天室模块核心代码.....	18
5. 好友/群组添加模块核心代码.....	22
6. 好友/群组聊天模块核心代码.....	22
六、关键问题及其解决方法.....	29
1. 找不到合适的数据结构.....	29
2. 好友私聊时信息丢失.....	29
3. 多个好友/群组面板的分布.....	29
4. 万恶的空指针.....	30
七、设计结果.....	32
八、软件使用说明.....	37
九、参考资料.....	38
十、验收时间及验收情况.....	38
十一、设计体会.....	40

# 第 1 类 基于套接字的网络编程

## 设计 1.3 广播通信设计

### 一、设计要求

- 掌握广播通信技术；
- 了解基于 Winsock API 的消息机制和编程应用方法；
- 了解 Windows SDK 编程架构。

在上述三条的前提下，完成 WinSock API 编程，实现局域网消息广播的实用程序；最后完善上述程序，使用网络广播知识制作一个可用的局域网聊天室软件。对于我所设计和实现的系统，现陈述如下：

通过所学的网络广播知识为基础，设计出了一个可用的局域网聊天室软件，其具体功能如下：

- 新用户第一次使用聊天室需要注册；
- 注册过的用户信息会自动存储在数据库之内，用户的好友和群组信息也会自动存入，下一次可以直接登陆，好友和群组信息也会加载进去；
- 两个用户之间可以通过好友信息搜索来添加好友，开始私聊；
- 多个用户之间可以通过群组信息搜索来加入群聊，一起聊天；
- 考虑到实践设计的要求，该软件以充足的图形界面实现，并具有足够的美观性。

通过查阅资料，下面我将简述单播、广播、多播的区别和联系，以及各自的优缺点和适应范围：

#### 单播：

单播是客户端与服务器之间的点到点连接。“点到点”指每个客户端都从服务器接收远程流。仅当客户端发出请求时，才发送单播流。单播是在一个单个的发送者和一个接受者之间通过网络进行的通信。

单播的优点：

- ✓ 服务器及时响应客户机的请求；

✓ 服务器针对每个客户不通的请求发送不通的数据，容易实现个性化服务。

单播的缺点：

- ✗ 服务器针对每个客户机发送数据流，服务器流量=客户机数量×客户机流量，在客户数量大、每个客户机流量大的流媒体应用中服务器不堪重负。
- ✗ 现有的网络带宽是金字塔结构，城际省际主干带宽仅仅相当于其所有用户带宽之和的 5%。如果全部使用单播协议，将造成网络主干不堪重负。

单播适用范围：

单播方式适合用户较少的网络

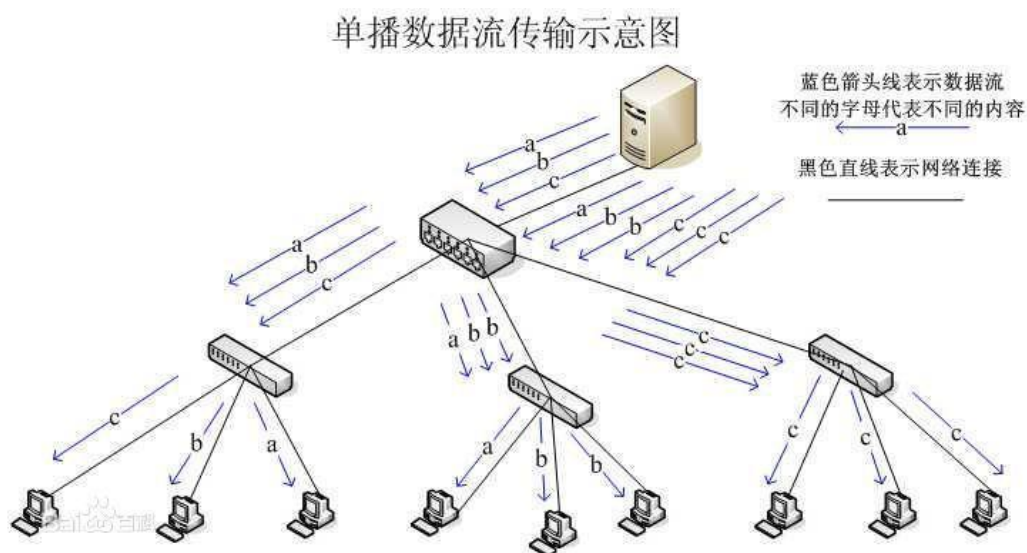


图 1-1 单播示意图

广播：

在同一个以太网环境下，一台主机如果需要同时向所有主机进行通信，这里其实主机不必要知道所有各点的地址。他只要向发送目的地址为广播地址，那么所有主机都会收到这台主机所发送的报文。

广播的优点：

- ✓ 网络设备简单，维护简单，布网成本低廉；
- ✓ 由于服务器不用向每个客户机单独发送数据，所以服务器流量负载极低。

广播的缺点：

- ✗ 无法针对每个客户的要求和时间及时提供个性化服务；

- ✗ 网络允许服务器提供数据的带宽有限，客户端的最大带宽=服务总带宽；
- ✗ 广播禁止在 Internet 宽带网上传输。

广播适用范围：

广播方式适合用户稠密的网络

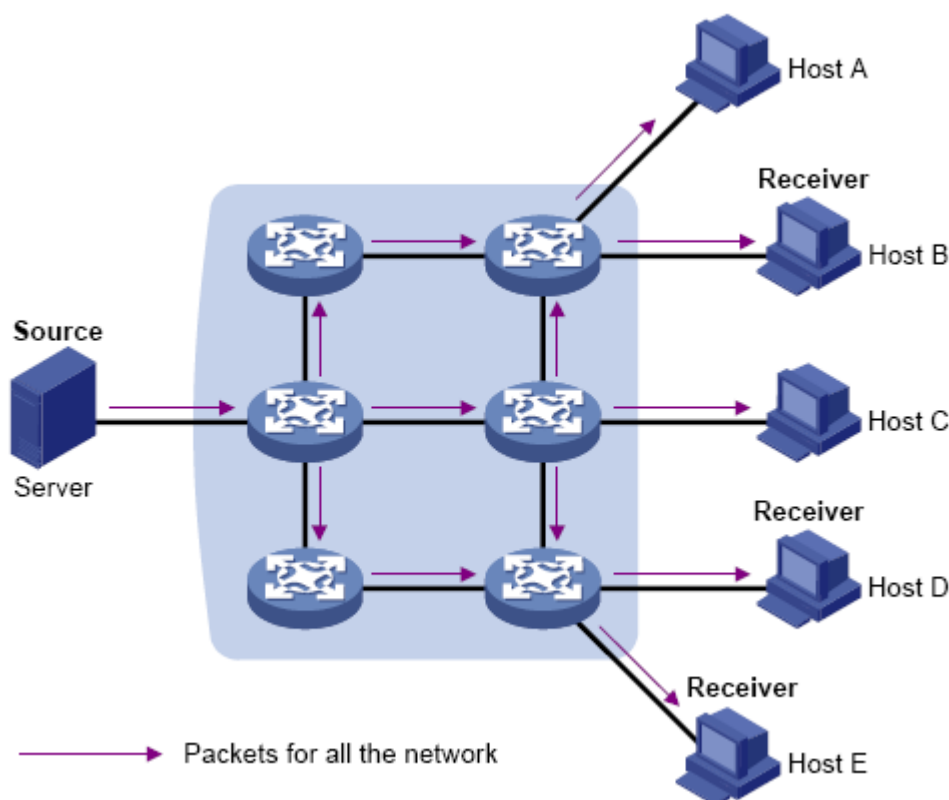


图 1-2 广播示意图

**多播（组播）：**

组播顾名思义就是在同组成员才能收到此数据报文，也就是说当我发送一个数据目的地址是这个组的组播地址的时候，所有在这个组的成员都会收到这个发送的内容。组播地址不能是源地址。

多播的优点：

- ✓ 需要相同数据流的客户端加入相同的组共享一条数据流，节省了服务器的负载。具备广播所具备的优点；
- ✓ 由于组播协议是根据接受者的需要对数据流进行复制转发，所以服务端的服务总带宽不受客户接入端带宽的限制。IP 协议允许有 2 亿 6 千多万个（268435456）组播，所以其提供的服务可以非常丰富；
- ✓ 此协议和单播协议一样允许在 Internet 宽带网上传输。



多播的缺点：

- ✘ 与单播协议相比没有纠错机制，发生丢包错包后难以弥补，但可以通过一定的容错机制和 QOS 加以弥补；
- ✘ 现行网络虽然都支持组播的传输，但在客户认证、QOS 等方面还需要完善。

多播适用范围：

综上所述，单播方式适合用户较少的网络，而广播方式适合用户稠密的网络，当网络中需求某信息的用户量不确定时，单播和广播方式效率很低。

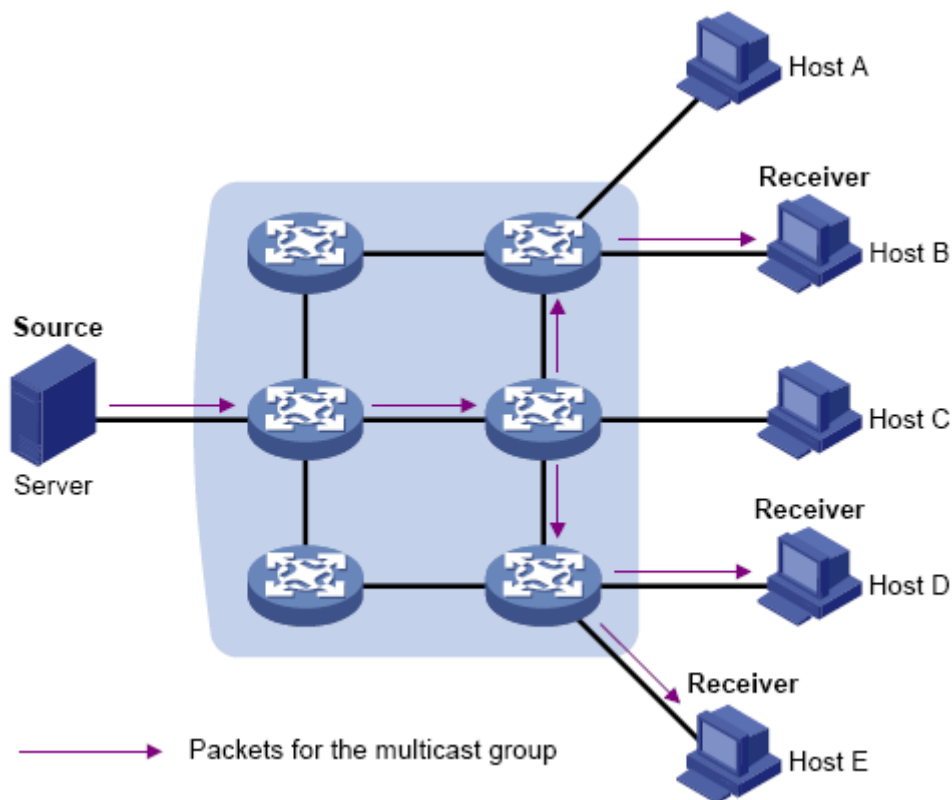


图 1-3 多播（组播）示意图

## 二、开发环境与工具

开发环境：eclipse 集成开发环境、Java 语言

工具：Intel®Core™i7-8750H 处理器@ 2.20GHz、64 位 Windows 10 操作系统

## 三、设计原理

该软件设计原理主要涉及的计算机网络知识有是:UDP 协议、TCP 协议、单播、组播。

在 Java 中, 要想实现 TCP 编程, 就需要 socket, 通常也称作“套接字”, 用于描述 IP 地址和端口, 是一个通信链的句柄。应用程序通常通过“套接字”向网络发出请求或者应答网络请求, 对 socket 的本身组成倒是比较好理解。既然是应用通过 socket 通信, 肯定就有一个服务器端和一个客户端。所以它必然就包含有一个对应的 IP 地址。另外, 在这个地址上 server 要提供一系列的服务, 于是就需要有一系列对应的窗口来提供服务。所以就有一个对应的端口号 (Port)。

Server 端所要做的事情主要是建立一个通信的端点, 然后等待客户端发送的请求。典型的处理步骤如下:

- 1) 构建一个 ServerSocket 实例, 指定本地的端口。这个 socket 就是用来监听指定端口的连接请求的。
- 2) 重复如下几个步骤:
  - a) 调用 socket 的 accept() 方法来获得客户端的连接请求。通过 accept() 方法返回的 socket 实例, 建立了一个和客户端的新连接。
  - b) 通过这个返回的 socket 实例获取 InputStream 和 OutputStream, 可以通过这两个 stream 来分别读和写数据。
  - c) 结束的时候调用 socket 实例的 close() 方法关闭 socket 连接。

那么在服务器端就是:

```
1. public class TestTCPSend {
2.     public static void main( String [] args) {
3.         ServerSocket serverSocket=null;
4.         Socket socket=null;
5.         String msg= "hello client,I am server.." ;
6.         try {
7.
8.             //构造 ServerSocket 实例, 指定端口监听客户端的连接请求
9.             serverSocket= new ServerSocket( 8080 );
10.            //建立跟客户端的连接
11.            socket=serverSocket.accept();
12.
13.            //向客户端发送消息
```

```

14.      OutputStream os=socket.getOutputStream();
15.      os.write(msg.getBytes());
16.      InputStream is=socket.getInputStream();
17.
18.      //接受客户端的响应
19.      byte [] b= new byte [ 1024 ];
20.      is.read(b);
21.      System.out.println( new String (b));
22.
23.  } catch (IOException e) {
24.      e.printStackTrace();
25.  } finally {
26.      //操作结束, 关闭 socket
27.      try {
28.          serverSocket.close();
29.          socket.close();
30.      } catch (IOException e) {
31.          e.printStackTrace();
32.      }
33.  }
34.  }
35. }

```

客户端的请求过程稍微有点不一样:

- 1) 构建 Socket 实例, 通过指定的远程服务器地址和端口来建立连接。
- 2) 通过 Socket 实例包含的 InputStream 和 OutputStream 来进行数据的读写。
- 3) 操作结束后调用 socket 实例的 close 方法, 关闭。

于是服务端代码就是:

```

1.  public class TestTCPReceive {
2.      public static void main( String [] args) {
3.          Socket socket=null;
4.          try {
5.              //对服务端发起连接请求
6.              socket= new Socket( "localhost" , 8080 );
7.
8.              //接受服务端消息并打印
9.              InputStream is=socket.getInputStream();
10.             byte b[]= new byte [ 1024 ];
11.             is.read(b);
12.             System.out.println( new String (b));

```

```

13.
14.         //给服务端发送响应信息
15.         OutputStream os=socket.getOutputStream();
16.         os.write( "yes,I have received you message!" .getBytes());
17.     } catch (IOException e) {
18.         // TODO Auto-generated catch block
19.         e.printStackTrace();
20.     }
21. }
22. }

```

在 Java 中，socket 的 UDP 的 API 并没有包含在 socket API 中，而是通过 DatagramSocket 和 DatagramPacket 来实现的。

DatagramSocket 只允许数据报发送给指定的目标地址，而 MulticastSocket 可以将数据报以广播的方式发送到多个客户端。所以 DatagramSocket 适合用于设计好友私聊，MulticastSocket 适合用于设计多人群聊。DatagramSocket 类是基于 UDP 协议进行通信的包装类，实现两个客户端通过 UDP 协议通信，可以使用 DatagramSocket 和 DatagramPacket 类，其中 DatagramSocket 构造方法如下表 1 所示：

表 3-1 DatagramSocket 类的构造方法

DatagramSocket()	构建一个数据报套接字，绑定到本地主机的任何可用的端口
DatagramSocket(int port)	构建一个数据报套接字，绑定到本地主机的指定端口
DatagramSocket(int port, InetAddress laddr)	创建一个数据报套接字，绑定到指定的本地地址
DatagramSocket(SocketAddress bindaddr)	创建一个数据报套接字，绑定到指定的本地套接字地址

若分为客户端和服务端，那么在客户端：

- 1) 实例化 DatagramSocket 类(带上指定端口)，创建客户端；
- 2) 准备数据，数据是以字节数组发送的；
- 3) 打包数据，使用 DatagramPacket 类 + 服务器地址+ 端口；
- 4) 发送数据；
- 5) 关闭连接。

也就是说客户端实例如下：

```

1. public static void main(String[] args) throws IOException {

```

```

2.      // 1.创建服务端+端口
3.      DatagramSocket client = new DatagramSocket(614);
4.
5.      // 2.准备数据
6.      String msg = "I love computer network ";
7.
8.      byte [] data = msg.getBytes();
9.
10.     // 3.打包（发送的地点及端口）
11.     DatagramPacket packet = new DatagramPacket(data, data.length, new InetS
        ocketAddress("127.0.0.1", 8888));
12.
13.     // 4.发送资源
14.     client.send(packet);
15.
16.     // 5.关闭资源
17.     client.close();
18. }

```

相应的，在服务器端：

- 1) 实例化 DatagramSocket 类+指定端口
- 2) 准备接收的字节数组，封装 DatagramPacket
- 3) 接受数据
- 4) 解析数据
- 5) 关闭连接

实例如下：

```

1.  public static void main(String[] args) throws IOException {
2.      // 1.创建服务端+端口
3.      DatagramSocket server = new DatagramSocket(1219);
4.
5.      // 2.准备接收容器
6.      byte[] container = new byte[1024];
7.
8.      // 3.封装成包 new DatagramPacket(byte[] b,int length)
9.      DatagramPacket packet = new DatagramPacket(container, container.length);
10.
11.     // 4.接收数据,使用 DatagramSocket 的实例的 receive( DatagramPacket ) 方法进行接收
12.     server.receive(packet);
13.

```

```

14. // 5.分析数据
15. byte[] data = packet.getData();
16. int length = packet.getLength();
17. String msg = new String(data, 0, length);
18. System.out.println(msg);
19. server.close();
20. }

```

MulticastSocket 有点像 DatagramSocket, 事实上 MulticastSocket 是 DatagramSocket 的一个子类, 当要发送一个数据报时, 可以使用随机端口创建一个 MulticastSocket, 也可以在指定端口创建 MulticastSocket。MulticastSocket 提供了如下表 2 所示的构造方法。

表 3-2 MulticastSocket 的构造方法

MulticastSocket()	使用本机默认地址、随机端口来创建 MulticastSocket 对象
MulticastSocket(int portNumber)	使用本机默认地址、指定端口来创建对象
MulticastSocket(SocketAddress bindaddr)	使用本机指定 IP 地址、指定端口来创建对象

他们的 API 都很多, 在此不表, 在后面的代码分析部分我会着重介绍。

## 四、系统功能描述及软件模块划分

我通过所学的网络广播知识为基础, 设计出了一个可用的局域网聊天室软件, 其具体功能描述如下:

- 新用户第一次使用聊天室需要注册;
- 注册过的用户信息会自动存储在数据库之内, 用户的好友和群组信息也会自动存入, 下一次就可以直接登陆, 好友和群组信息也会加载进去;
- 两个用户之间可以通过好友信息搜索来添加好友, 好友间可以私聊;
- 多个用户之间可以通过群组信息搜索来加入群聊, 一起聊天;
- 考虑到实践设计的要求, 该软件以充足的图形界面实现, 并具有足够的美观性。

我将该聊天室软件模块划分为如下 6 个模块:

### 1. 服务器模块(Server)

服务器模块运行于后台监听登陆/注册模块的信息。

当等待到登录/注册模块传来的账号密码时，与从服务器自带的哈希表 user 中读取的已注册的用户信息进行端口号、用户名、密码核对，如果该用户已存在且选择的是登陆，则登陆成功，进入用户主页面，并把数据库中该用户中的好友、群组信息显示在主页面上；如果该用户不存在，则需要注册。

## 2. 登陆/注册模块(loginOrRegister)

登陆/注册模块主要与服务器模块和用户模块互联。

该模块提供一个登陆/注册界面，使用者在界面中输入用户名与密码，然后登录/注册模块会把这些数据稍作处理后发送给服务器。服务器根据一定的逻辑对这些数据进行判断，并且根据判断的结果进行相应的动作，详见上面服务器模块的介绍。

此外当新用户注册时，此模块还会在数据库新建该用户的信息文件。

## 3. 用户模块(Member)

用户模块与登陆/注册模块和聊天室模块互联。

该模块负责管理用户的信息，信息借助于聊天室模块来显示，当老用户登陆成功时，用户模块负责将数据库内的老用户好友和群组信息加载显示到聊天室模块中；当用户新添加好友或群组时，用户模块将新好友和群组的信息添加到该用户对应的数据库中。

## 4. 聊天室模块(MainFrame)

聊天室模块与用户模块以及其余所有模块相联。

聊天室模块主要刷新并显示用户的好友和群组并支持进入聊天，并提供添加好友，添加群组的功能。

## 5. 好友/群组聊天模块(friendDialogFrame/groupDialogFrame)

好友/群组聊天模块与聊天室模块相联。

好友聊天以单播实现，用户 IP 地址皆为 127.0.0.1，聊天支持发送文字；群组聊天以多播实现，群组 IP 地址皆为 224.0.0.1，聊天支持发送文字。

## 6. 好友/群组添加模块(addFriend/addGroup)

用于添加好友/群组。添加好友需要提供好友用户名、端口号、好友 IP 地址；添加群组需要提供群组名和群组 IP 地址。

# 五、设计步骤

由于该软件模块较多，模块自身有函数，模块之间也有函数关系，所以这部分主要是单独介绍各个模块自身的函数情况，并在合适的地方总结模块之间的相互关系。

在这部分的最后我将总结所有函数之间的关系。

## 1. 服务器模块核心代码

```
1.      public void run(){
2.  try{
3.      BufferedReader in=new BufferedReader(new InputStreamReader(socket.getInp
      utStream()));
4.      BufferedWriter out=new BufferedWriter(new OutputStreamWriter(socket.getO
      utputStream()));
5.      String login_reg;
6.      while(true){
7.          login_reg=in.readLine();
8.          //登陆/注册模块传来的信息以@分段
9.          String info[]=login_reg.split("@");
10.         //分段结果大于三段
11.         if(info.length>3){
12.             out.write("no@no@用户名或密码非法\n");
13.             out.flush();
14.             //分段结果小于三段
15.         }else if(info.length<3){
16.             out.write("no@no@请输入完整信息\n");
17.             out.flush();
18.             //分段结果等于三段
19.         }else{
20.             //具体是注册模块时
21.             if(info[0].equals("register")){
22.                 //老用户注册
23.                 if(user.containsKey(info[1])){
24.                     out.write("no@register@用户已存在\n");
25.                     out.flush();
26.                     //新用户第一次注册
27.                 }else{
28.                     String port[]=info[2].split("#");
29.                     out.write("yes@"+port[1]+"@欢迎\n");
30.                     out.flush();
31.                     user.put(info[1],info[2]);
32.                 }
33.                 //具体是登陆模块时
34.             }else if(info[0].equals("login")){
```



```

35.         //老用户登陆
36.         if(user.containsKey(info[1])){
37.             if(user.get(info[1]).equals(info[2])){
38.                 String port[]=info[2].split("#");
39.                 out.write("yes@" +port[1]+"@欢迎\n");
40.                 out.flush();
41.                 //老用户密码错误时
42.             }else{
43.                 out.write("no@login@密码错误\n");
44.                 out.flush();
45.                 System.out.println("注册成功 4");
46.             }
47.             //新用户直接登录
48.         }else{
49.             out.write("no@login@用户名不存在请注册\n");
50.             out.flush();
51.         }
52.     }
53. }
54. }
55. }catch(Exception e){
56.     System.out.println(e);
57. }

```

以上是对登陆/注册模块传来的信息进行分段、处理并返回给登陆/注册模块的关键代码，主要是看用户是登陆还是注册，当传来的信息符合是三段的基本要求时：

若是注册，则根据登陆/注册模块传来的信息分成：

- 老用户注册，此时不允许重复注册，回传“no@register@用户已存在\n”；
- 新用户注册，注册成功，回传 “yes@” +port[1]+ “@欢迎\n” ；

若是登陆，则根据发来的信息分成：

- 老用户登陆，登陆成功，回传 “yes@” +port[1]+ “@欢迎\n” ；
- 老用户登陆，密码错误，登陆失败，回传 “no@login@密码错误\n” ；
- 新用户登陆，此时不允许直接登陆，需要先注册，回传 “no@login@用户名不存在请注册\n”

与服务器模块相联的模块就是登陆/注册模块，下面进行介绍：

## 2. 登陆/注册模块核心代码

```
1.      //按下登陆按钮
2.  login_but.addMouseListener(new MouseAdapter(){
3.      public void mouseClicked(MouseEvent m){
4.          try{
5.              //将用户登陆输入的信息处理后送到服务器端
6.              String msg="login@"+usr_name_t.getText()+"@"+usr_pswd_t.getText(
              )+"\n";
7.              out.write(msg);
8.              out.flush();
9.              //接收服务器端对消息处理后回传的信息
10.             BufferedReader in=new BufferedReader(new InputStreamReader(socke
                t.getInputStream()));
11.             msg=in.readLine();
12.             String info[]=msg.split("@");
13.             if(info[0].equals("yes")){
14.                 username=usr_name_t.getText();
15.                 userport=Integer.valueOf(info[1]).intValue();
16.                 out.close();
17.                 in.close();
18.                 frame.dispose();
19.                 login=true;
20.             }else{
21.                 usr_name_t.setText(info[2]);
22.             }
23.         }catch(Exception e){
24.             System.out.println(e);
25.         }
26.     }
27. });
28. //按下注册按钮
29. register_but.addMouseListener(new MouseAdapter(){
30.     public void mouseClicked(MouseEvent m){
31.         try{//将用户注册输入的信息处理后送到服务器端
32.             String msg="register@"+usr_name_t.getText()+"@"+usr_pswd_t.getTe
                xt()+"\n";
33.             out.write(msg);
34.             out.flush();
35.             //接收服务器端对消息处理后回传的信息
36.             BufferedReader in=new BufferedReader(new InputStreamReader(socke
                t.getInputStream()));
37.             msg=in.readLine();
38.             String info[]=msg.split("@");
```

```

39.         if(info[0].equals("yes")){
40.             username=usr_name_t.getText();
41.             userport=Integer.valueOf(info[1]).intValue();
42.             out.close();
43.             in.close();
44.             frame.dispose();
45.             //建立新用户的数据库文件
46.             createFile(username);
47.             login=true;
48.         }else{
49.             usr_name_t.setText(info[2]);
50.         }
51.     }catch(Exception e){
52.         System.out.println(e);
53.     }
54. }
55. });

```

以上要与上面服务器端的处理一起来看，对比第一段服务器端的处理过程，我们可以知道：

当是注册时，回传给登陆/注册模块的信息分为以下情况：

- 老用户注册，回传 “no@register@用户已存在\n”，将在登陆/注册模块的用户名栏显示“用户已存在”；
- 新用户注册，回传 “yes@” +port[1]+ “@欢迎\n”，这在登陆/注册模块中将创建新用户的数据库文件；

当时登陆时，

- 老用户登陆，回传 “yes@” +port[1]+ “@欢迎\n”，此时将获取用户名和端口号；
- 老用户登陆，回传 “no@login@密码错误\n”，这将在登陆/注册模块的用户名栏显示“密码错误”；
- 新用户登陆，回传 “no@login@用户名不存在请注册\n”，这将在登陆/注册模块的用户名栏显示“用户名不存在”；

以上是登陆/注册模块与服务器模块的互联关系，同时登陆/注册模块还与用户模块互联（互联的媒介就是用户数据库），这里先用流程图来表示一下登陆/注册模块与服务器模块的互联关系：

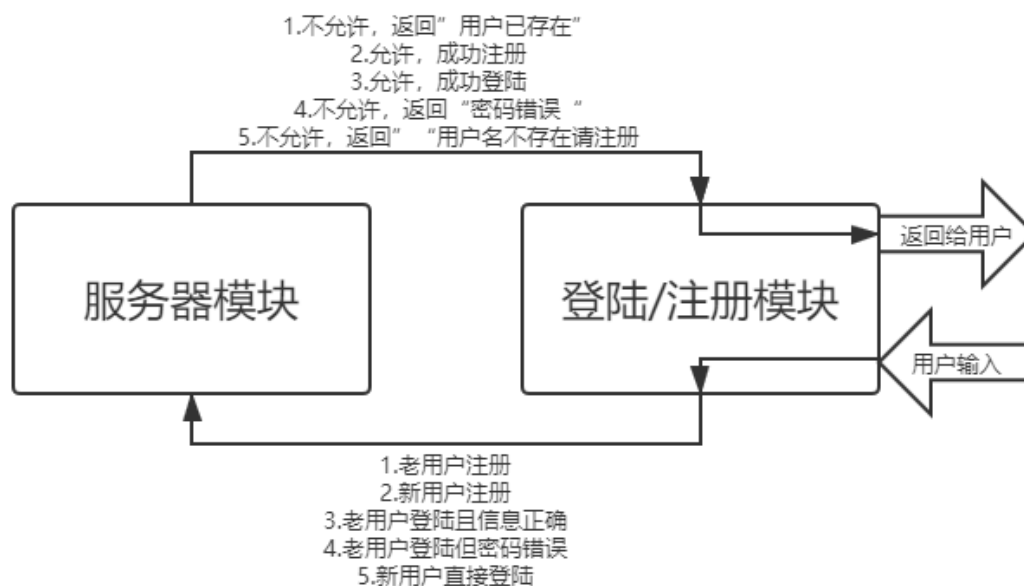


图 5-1 登陆/注册模块与服务器模块互联关系流程图

与登陆/注册模块相联的模块是用户模块，下面进行介绍：

### 3. 用户模块核心代码

用户模块的核心代码有两作用，一部分是与添加好友/群组模块相联（但是添加好友/群组模块实际上也是聊天室模块的一部分），另一部分是与聊天室模块相联，我将逐个介绍。

与添加好友/群组模块相联的代码部分是：

```

1. //当用户新添加一个好友时，将好友信息更新到数据库内该用户数据库的好友文件 friends.txt
   中
2. void updata_data(friend f){
3.     try {
4.         BufferedWriter bwrite=new BufferedWriter(new FileWriter(".\\userdata
        \\"+username+"\\friends.txt",true));
5.         bwrite.write((f.friend_inetAddress+"@"+f.friend_name+"@"+f.friend_po
        rt).substring(1));
6.         bwrite.newLine();
7.         bwrite.close();
8.     } catch (IOException e) {
9.         e.printStackTrace();

```

```

10.     }
11. }
12.
13. //当用户新添加一个群组时，将群组信息更新到数据库内该用户数据库的群组文件 groups.txt
    中
14. void updata_data(group g){
15.     try {
16.         BufferedWriter bwrite=new BufferedWriter(new FileWriter(".\\userdata
            \\ "+username+"\\groups.txt",true));
17.         bwrite.write((g.group_inetAddress+"@"+g.group_name).substring(1));
18.         bwrite.newLine();
19.         bwrite.close();
20.     } catch (IOException e) {
21.         e.printStackTrace();
22.     }
23. }

```

以上这部分将在聊天室模块使用添加好友/群组功能时生效，也就是将用户新添加的好友/群组信息写入到用户数据库中。方便下次登陆时，聊天室模块可以直接载入这些信息，而不需要重新添加这些好友和群组。

与聊天室模块相联的代码部分是：

```

1.     void load_data(){
2.     try{
3.         //将数据库内的好友信息载入 friend_array 中
4.         File fp=new File(".\\userdata\\"+username+"\\friends.txt");
5.         InputStreamReader read=new InputStreamReader(new FileInputStream(fp),"GB
            K");
6.         BufferedReader bfrear=new BufferedReader(read);
7.         String text=bfread.readLine();
8.         while(text!=null){
9.             String friend_info[]=text.split("@");
10.            friend_array.add(new friend(friend_info[0],friend_info[1],friend_inf
                o[2]));
11.            text=bfread.readLine();
12.        }
13.        bfrear.close();
14.        //将数据库内的群组信息加载到 group_array 中
15.        fp=new File(".\\userdata\\"+username+"\\groups.txt");
16.        read=new InputStreamReader(new FileInputStream(fp),"GBK");
17.        bfrear=new BufferedReader(read);
18.        text=bfread.readLine();
19.        while(text!=null){

```

```

20.      String group_info[]=text.split("@");
21.      group_array.add(new group(group_info[0],group_info[1]));
22.      text=bfread.readLine();
23.  }
24.  bfread.close();
25. }catch(Exception e){
26.     System.out.println(e);
27. }

```

这部分就是对上部分 updata\_data 函数结果的使用，也就是 load\_data，将之前存储好的已添加的好友/群组信息直接加载到聊天室模块中去显示。

这部分的流程图如下图 5-2 所示：

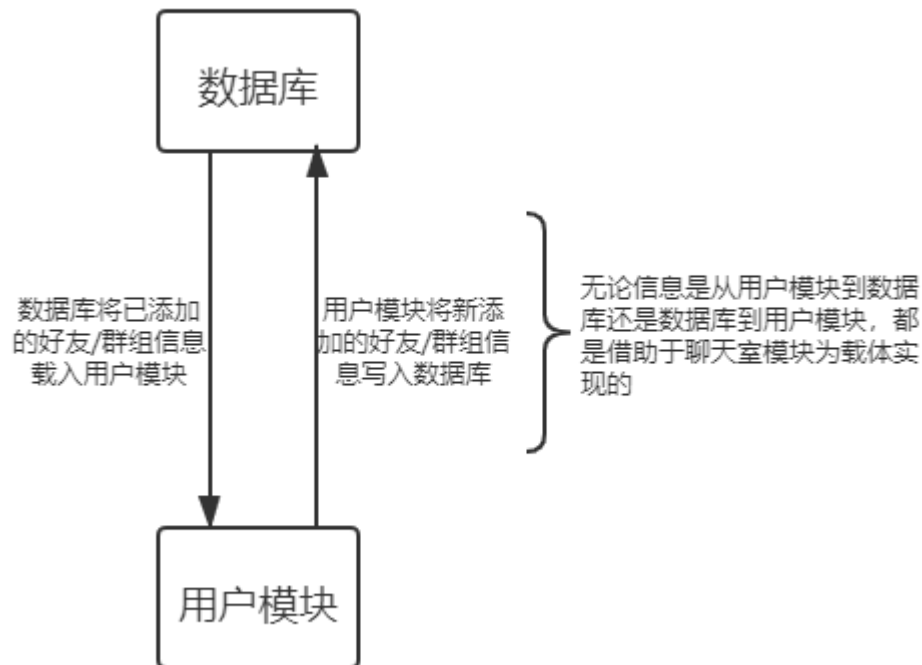


图 5-2 数据库与用户模块互联关系流程图

与用户模块相联的模块的是聊天室模块，下面进行介绍：

#### 4. 聊天室模块核心代码

为了减少不必要的篇幅，此处我将四个主要功能按钮的重载函数不表，聊天室模块的核心代码分为两部分，一部分是借助好友/群组信息来新建好友/群组显示面板的代码，另外一部分是聊天室的四个主要功能按钮的代码，前一部分代码如下：

```

1. //根据一个好友的信息，在 panel 上显示出一个好友控件

```

```

2. void draw_panel(friend f){
3.     //好友的 IP 地址
4.     JLabel address=new JLabel("          好友 IP 地
        址:"+f.friend_inetAddress);
5.     address.setBounds(0, 40*num, 200, 40);
6.     address.setBorder(javax.swing.BorderFactory.createLineBorder(new java.a
        wt.Color(0, 0, 0)));
7.     add_panel.add(address);
8.     //好友的端口号
9.     JLabel name=new JLabel("          好友用户名:"+f.friend_name);
10.    name.setBounds(200, 40*num, 175, 40);
11.    name.setBorder(javax.swing.BorderFactory.createLineBorder(new java.awt.
        Color(0, 0, 0)));
12.    add_panel.add(name);
13.    //与好友聊天按键
14.    JButton enter_but=new JButton("开始私聊");
15.    enter_but.setBounds(375, 40*num, 124, 40);
16.    add_panel.add(enter_but);
17.    num++;
18.    add_panel.revalidate();
19.    enter_but.addMouseListener(new MouseAdapter(){
20.        public void mouseClicked(MouseEvent m){
21.            open_dialog.put(f.friend_name, new friendDialogFrame(f,username
                ,enter_but,datagramsocket));
22.            //enter_but.setEnabled(false);
23.        }
24.    });
25. }
26. //根据一个群组的信息，在 panel 上显示处一个群组控件
27. void draw_panel(group g){
28.     //好友的 IP 地址
29.     JLabel address=new JLabel("          群聊 IP 地址
        "+g.group_inetAddress);
30.     address.setBounds(0, 40*num, 200, 40);
31.     address.setBorder(javax.swing.BorderFactory.createLineBorder(new java.a
        wt.Color(0, 0, 0)));
32.     add_panel.add(address);
33.     //好友的端口号
34.     JLabel name=new JLabel("          群聊组名称:"+g.group_name);
35.     name.setBounds(200, 40*num, 175, 40);
36.     name.setBorder(javax.swing.BorderFactory.createLineBorder(new java.awt.
        Color(0, 0, 0)));
37.     add_panel.add(name);
38.     //与好友聊天按键

```

```

39.     JButton enter_but=new JButton("进入群聊");
40.     enter_but.setBounds(375, 40*num, 125, 40);
41.     add_panel.add(enter_but);
42.     num++;
43.     add_panel.revalidate();
44.     enter_but.addMouseListener(new MouseAdapter(){
45.         public void mouseClicked(MouseEvent m){
46.             if(enter_but.isEnabled()){
47.                 new groupDialogFrame(g,username,enter_but);
48.                 //enter_but.setEnabled(false);
49.             }
50.         }
51.     });

```

这部分就是根据好友和群组的基本信息在聊天室主页面上加载出好友和群组的显示面板，主要还是 UI 设计代码为主，主要功能是进入聊天，可以看到好友面板上有进入好友私聊的按钮，群组面板上有进入群组群聊的按钮，通过这两个按钮可以进入好友/群组聊天模块。

另外一部分，四个主要按钮的功能代码如下（重载函数隐去）：

```

1.     //好友按钮
2.     friend_but.addMouseListener(new MouseListener(){
3.         @Override
4.         public void mouseClicked(MouseEvent arg0) {
5.             num=0;
6.             add_panel.removeAll();
7.             init_panel_f(member.friend_array);
8.             add_panel.repaint();
9.         }
10.    });
11. //群聊按钮
12.    group_but.addMouseListener(new MouseListener(){
13.
14.        @Override
15.        public void mouseClicked(MouseEvent e) {
16.            num=0;
17.            add_panel.removeAll();
18.            init_panel_g(member.group_array);
19.            //frame.revalidate();
20.            add_panel.repaint();
21.        }
22.    });

```

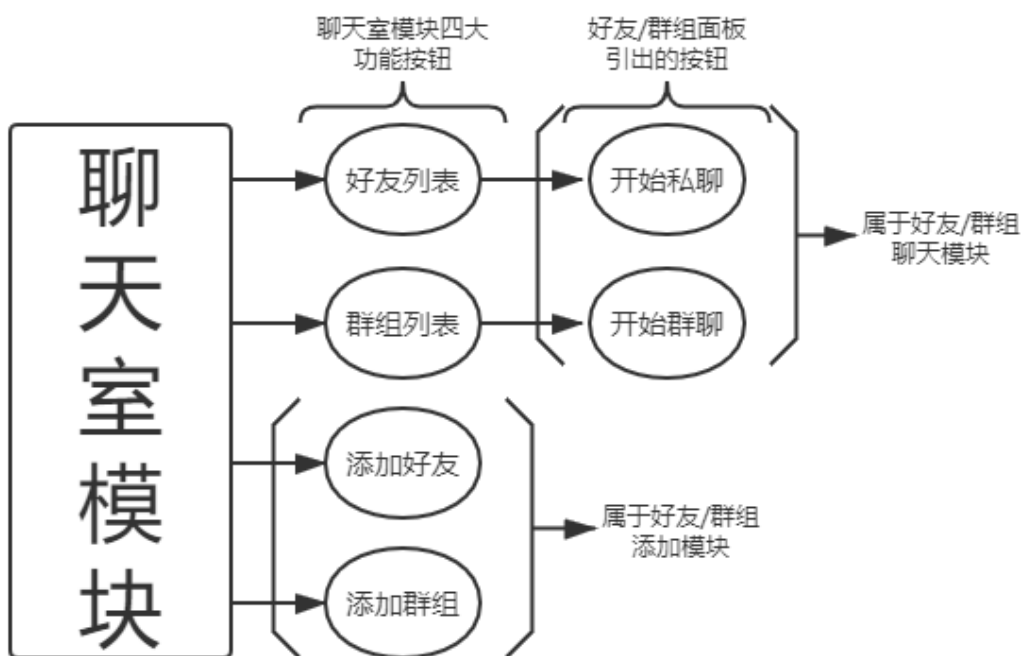


```

23. //添加好友按钮
24.     new_friend_but.addMouseListener(new MouseAdapter(){
25.         public void mouseClicked(MouseEvent m){
26.             new addFriend(member);
27.         }
28.     });
29. //添加群组按钮
30.     new_group_but.addMouseListener(new MouseAdapter(){
31.         public void mouseClicked(MouseEvent m){
32.             new addGroup(member);
33.         }
34.     });

```

可以看到，好友列表按钮就是通过之前用户模块载入的 friend\_array 表来刷新并且画出新的好友面板；群组列表按钮就是通过之前用户模块载入的 group\_array 表来刷新并且画出新的群组面板；添加好友/群组按钮是通过下面要介绍的好友/群组添加模块来实现的。



所以聊天室模块的流程图就如下图 5-3 所示。

图 5-3 聊天室模块流程图

根据上面的介绍，我们知道其余的与聊天室模块相联的是好友/群组添加模块和好友/群组聊天模块，下面进行介绍：

## 5. 好友/群组添加模块核心代码

好友添加模块

```
1. add_friend.addMouseListener(new MouseAdapter(){
2.     public void mouseClicked(MouseEvent m){
3.         friend f=new friend(ip_t.getText(),friendname_t.getText(),friendport_t.getText());
4.         member.updata_data(f);
5.         member.friend_array.add(f);
6.         frame.dispose();
7.     } });
```

群组添加模块

```
1. add_group.addMouseListener(new MouseAdapter(){
2.     public void mouseClicked(MouseEvent m){
3.         group g=new group(ip_t.getText(),groupname_t.getText());
4.         member.updata_data(g);
5.         member.group_array.add(g);
6.         frame.dispose();
7.     } });
```

这部分的 UI 设计代码就不列出了，所以上述是好友/群组添加按钮的事件代码，由于之前的模块设计比较全面，类之间的关系也很清晰，所以到了后面部分代码就很简单。主要功能就是往数据库里面更新用户的好友/群组信息，同时更新 friend\_array 和 group\_array 的信息，方便刷新好友/群组列表时可以显示出来。

## 6. 好友/群组聊天模块核心代码

无论是好友聊天还是群组聊天，核心代码都是信息发送按钮的回调函数，同理按钮的其余重载函数就不表了：

好友私聊信息发送按钮：

```
1.         //好友私聊发送按钮
2. end_but.addMouseListener(new MouseListener(){
3.
4.     @Override
```

```

5. //点击发送按钮时，将用户名以及输入框中的文本作为数据单播给好友，然后将输入框中的文本清空
6. public void mouseClicked(MouseEvent arg0) {
7.     byte[] send_message;
8.     DatagramPacket send_packet;
9.     try{
10.         send_message=(username+"@"+friendname+"@"+input_t.getText()).getBytes();
11.         send_packet=new DatagramPacket(send_message,send_message.length,inetAddress,port_num);
12.         datagramsocket.send(send_packet);
13.         text_area.setText(text_area.getText()+username+": "+input_t.getText()+"\n");
14.         input_t.setText("");
15.     }catch(IOException e){
16.         System.out.println("error happen in the sender "+username);
17.     }
18. }
19. });

```

群组群聊信息发送按钮：

```

1. //群组群聊发送按钮
2. send_but.addMouseListener(new MouseListener(){
3.
4.     @Override
5.         //点击发送按钮时，将用户名以及输入框中的文本作为数据组播给群成员，然后将输入框中的文本清空
6.         public void mouseClicked(MouseEvent arg0) {
7.             byte[] send_message;
8.             DatagramPacket send_packet;
9.             try{
10.                 send_message=(username+": "+input_t.getText()).getBytes();
11.                 send_packet=new DatagramPacket(send_message,send_message.length,inetAddress,port_num);
12.                 multicastsocket.send(send_packet);
13.                 input_t.setText("");
14.             }catch(IOException e){
15.                 System.out.println("error happen in the sender "+username);
16.             }
17.         }
18.     });
19. //接收线程，程序监控群里是否有人发送消息，如果有，则取出，同时显示在消息框中

```

```

20.    Thread receive_thread=new Thread(){
21.        public void run(){
22.            byte receive_msg[]=new byte[1000];
23.            DatagramPacket receive_packet=new DatagramPacket(receive_msg,100
0);
24.            while(true){
25.                try{
26.                    multicastsocket.receive(receive_packet);
27.                    byte[] receive_message=receive_packet.getData();
28.                    text_area.setText(text_area.getText()+new String(receive
_message,0,receive_packet.getLength())+"\n");
29.                }catch(Exception e){
30.                    System.out.println("error happen in the receiver "+usern
ame);
31.                }
32.            }
33.        }
34.    };
35.    receive_thread.start();
36. }

```

可以看到好友私聊是通过 datagramsocket 单播实现的，而群组群聊是通过 multicastsocket 多播实现的。需要注意的是，由于是组播，群聊还需要运行一个接收线程，用于监控群里是否有人发送消息，如果有，则取出，同时显示在消息框中。

完成上述所有的介绍后，我们不难得到上述模块的函数的总表和函数之间的总体关系：

表 5-1 所有模块内关键函数汇总表

所属模块	函数名	功能
服务器模块 Server	void ChatRoom.Server.Client.run()	处理登陆/注册模块传来的信息并回传
	void ChatRoom.Server.run()	服务器模块的界面设计
	void ChatRoom.loginOrRegister.login_reg()	登陆/注册模块的界面设计
	void ChatRoom.loginOrRegister.login_reg().new MouseAdapter() {...}.mouseClicked(MouseEvent m) JButton register_but -	注册时，负责发送和返回信息给服

登陆/注册模块 loginOrRegister	ChatRoom.loginOrRegister.login_reg()	服务器端信息的函数
	void ChatRoom.loginOrRegister.login_reg().new MouseAdapter() {...}.mouseClicked(MouseEvent m) JButton login_but - ChatRoom.loginOrRegister.login_reg()	登陆时，负责发送和返回信息给服务器端信息的函数
	void ChatRoom.loginOrRegister.createFile(String username)	注册成功时，新建用户信息数据库的函数
用户模块 Member	void ChatRoom.Member.load_data()	老用户登陆时，将数据库中该用户的信息载入动态数组，便于后续聊天室模块调用
	void ChatRoom.Member.updata_data(friend f)	用户添加好友时，将新好友信息写入该用户数据库
	void ChatRoom.Member.updata_data(group g)	用户添加群组时，将新群组信息写入该用户数据库
	ChatRoom.MainFrame.MainFrame(Member member)	聊天室模块的界面设计
	void java.awt.Component.addMouseListener(MouseListener l) JButton ChatRoom.MainFrame.friend_but	聊天室好友列表按钮事件函数
	void java.awt.Component.addMouseListener(MouseListener l) JButton ChatRoom.MainFrame.group_but	聊天室群组列表按钮事件函数
	void java.awt.Component.addMouseListener(MouseListener l) JButton ChatRoom.MainFrame.new_friend_but	添加好友按钮事件函数
	void	添加群组按钮

聊天室模块 MainFrame	java.awt.Component.addMouseListener(MouseListener l) JButton ChatRoom.MainFrame.new_group_but	钮事件函数
	void ChatRoom.MainFrame.deal_msg(String rece_msg)	接受私聊单播传来的信息
	void ChatRoom.MainFrame.init_panel_f(ArrayList<friend> f_r)	根据好友动态数组通过draw_panel初始化好友列表面板
	void ChatRoom.MainFrame.init_panel_g(ArrayList<group> g_r)	根据群组动态数组通过draw_panel初始化群组列表面板
	void ChatRoom.MainFrame.draw_panel(friend f)	根据有一个好友的信息，在panel上显示出一个好友的信息面板
	void ChatRoom.MainFrame.draw_panel(group g)	根据有一个群组的信息，在panel上显示出一个群组的信息面板
	ChatRoom.addFriend.addFriend(Member member)	添加好友模块界面设计
	void java.awt.Component.addMouseListener(MouseListener l) JButton ChatRoom.addFriend.add_friend	添加好友按钮界面打开事件函数
	ChatRoom.addGroup.addGroup(Member member)	添加群组模块界面设计
	void java.awt.Component.addMouseListener(MouseListener l) JButton ChatRoom.addGroup.add_group	添加群组事件界面打开函数

	void java.awt.Component.addMouseListener(MouseListener l) JButton enter_but - ChatRoom.MainFrame.draw_panel(friend)	进入好友私聊按钮事件函数
	void java.awt.Component.addMouseListener(MouseListener l) JButton enter_but - ChatRoom.MainFrame.draw_panel(group)	进入群组群聊按钮事件函数
好友/群组聊天模块	void ChatRoom.friendDialogFrame.dialog_frame(JButton enter_but)	好友私聊界面设计函数
	void java.awt.Component.addMouseListener(MouseListener l) JButton exit_but - ChatRoom.friendDialogFrame.dialog_frame(JButton)	好友私聊退出按钮函数
	void java.awt.Component.addMouseListener(MouseListener l) JButton send_but - ChatRoom.friendDialogFrame.dialog_frame(JButton)	好友私聊信息发送按钮事件函数
	void ChatRoom.groupDialogFrame.dialog_frame(JButton enter_but)	群组群聊界面设计函数
	void java.awt.Component.addMouseListener(MouseListener l) JButton exit_but - ChatRoom.groupDialogFrame.dialog_frame(JButton)	群组群聊退出按钮函数
	void java.awt.Component.addMouseListener(MouseListener l) JButton send_but - ChatRoom.groupDialogFrame.dialog_frame(JButton)	群组群聊信息发送按钮事件函数
	void ChatRoom.groupDialogFrame.dialog_frame(...).new Thread() {...}.run()	接收线程函数，程序监控群里是否有人发送消息，如果有，则取出，同时显示在消息框中

好友/群组添加模块	ChatRoom.addFriend.addFriend(Member member)	添加好友界面设计函数
	void java.awt.Component.addMouseListener(MouseListener l) JButton ChatRoom.addFriend.add_friend	添加好友按钮事件函数
	ChatRoom.addGroup.addGroup(Member member)	添加群组界面设计函数
	void java.awt.Component.addMouseListener(MouseListener l) JButton ChatRoom.addGroup.add_group	添加群组按钮事件函数

总结上述函数，按钮事件函数以按钮名代替，不难得到下面的总函数关系流程图 5-4：

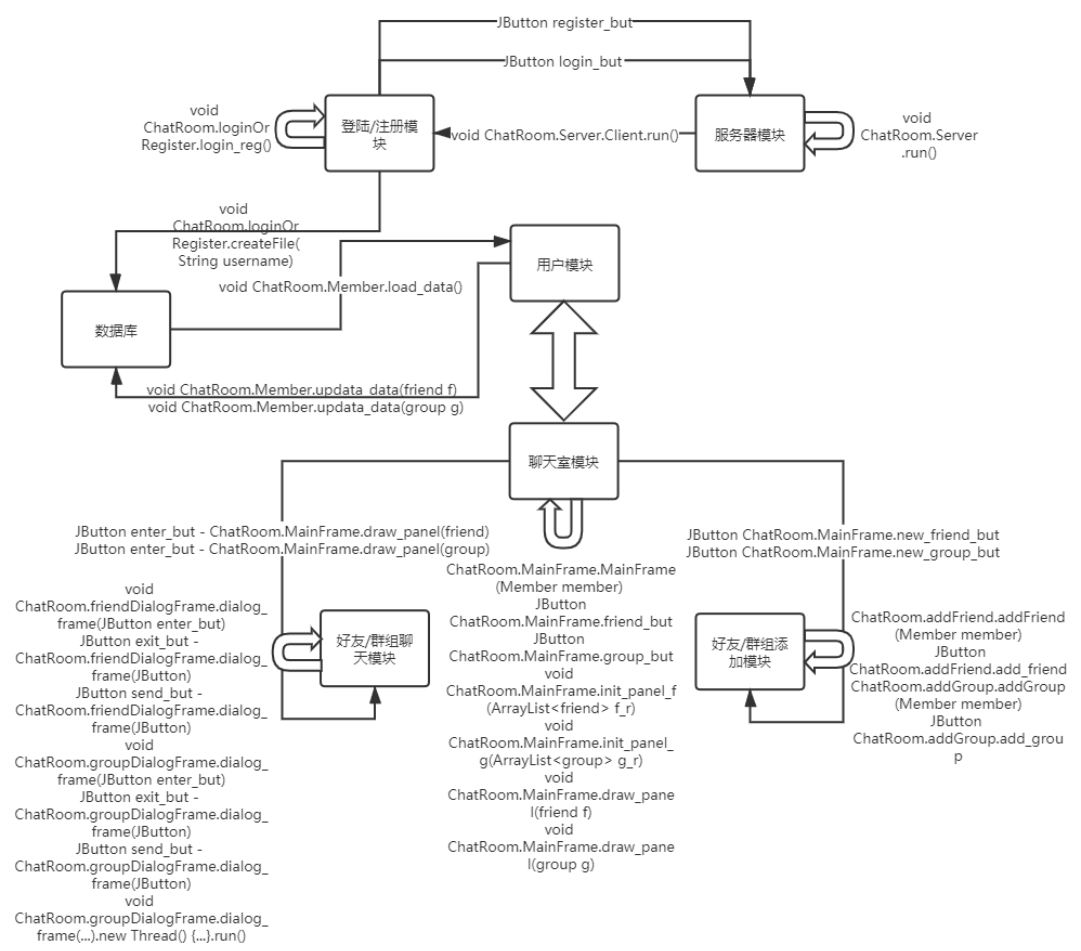


图 5-4 函数总体关系图



## 六、关键问题及其解决方法

### 1. 找不到合适的数据结构

因为我选择的课设题目是聊天室，所以在登陆页面需要很多信息的传输、查验、存储，刚开始这些信息总是被我搞得很乱，后面我在网上看了很多相关的信息，顺带也算是把数据结构复习了一下，因为用户最主要的信息就是用户名、密码、端口、IP 地址，群组主要的信息是群组名字、IP 地址，都是一个以上且存在唯一的对应关系，所以选择用 HashMap 无疑是很好的选择，对于多段信息的用户，可以选择常用的方法：将密码和端口融合到一起以特殊符号#分割，到时候再用 split 处理。

### 2. 好友私聊时信息丢失

在收到好友的消息时候，在聊天时打入大段字的时候，总会出现输出信息不全的情况，后来查阅资料我才知道，得到好友的消息时，我们的处理过程必须一次执行完，然后才进行下次的读入，这样能防止上一次数据被冲刷掉。在 java 中，我们可以利用 synchronized 关键字指明一个函数在执行过程中不能被打断。

```
1.     synchronized void deal_msg(String rece_msg){
2.     String msg[]=rece_msg.split("@");
3.     System.out.println(msg[2]);
4.     if(open_dialog.containsKey(msg[0])){
5.         open_dialog.get(msg[0]).text_area.setText(open_dialog.get(msg[0]).text_a
        rea.getText()+msg[0]+":"+msg[2]+"\\n");
6.         System.out.println(msg[2]);
7.     }else{
8.         System.out.println("????");
9.     }
```

### 3. 多个好友/群组面板的分布

以前没有设计过这种聊天室软件，第一次设计还是遇到很多问题的，其中就有用户界面怎么设计的好看的问题，我的方法就是不断的试，数字不断的改然后运行看怎么样，直到看的顺眼，偶尔还需要计算计算把控件对齐，这样会好看很多。

聊天室主界面设计的时候，多个好友/群组怎么放位置呢，想来想去还是和 QQ 一样，线性排列比较好看，怎么实现呢？那时候我突发奇想，用一个递增的

num 来乘上控件的宽度，这样就能实现每多一个控件，他的顶部位置就是“num\*已有的控件数”的宽度，如下：

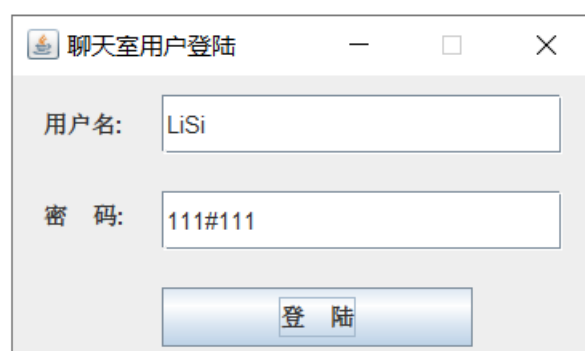
```
1.      JLabel address=new JLabel("                群聊 IP 地址  
      "+g.group_inetAddress);  
2.  address.setBounds(0, 40*num, 200, 40);  
3.  address.setBorder(javax.swing.BorderFactory.createLineBorder(new java.awt.Co  
      lor(0, 0, 0)));  
4.  add_panel.add(address);  
5.  //好友的端口号  
6.  JLabel name=new JLabel("                群聊组名称:"+g.group_name);  
7.  name.setBounds(200, 40*num, 175, 40);  
8.  name.setBorder(javax.swing.BorderFactory.createLineBorder(new java.awt.Color  
      (0, 0, 0)));  
9.  add_panel.add(name);  
10. //与好友聊天按键  
11. JButton enter_but=new JButton("进入群聊");  
12. enter_but.setBounds(375, 40*num, 125, 40);  
13. add_panel.add(enter_but);  
14. num++;
```

#### 4. 万恶的空指针

在噩梦的 3 月 12 号下午，出现了 Java 中令人闻风丧胆的空指针错误，而且也不显示错误行号，这让我检查了整整 4 天，终于在 15 号下午查出了原因。最后我选择的办法是推倒出错的部分重写，最后调试的时候都已经不抱希望了，结果居然跑通了，那一刻真是心情舒畅，然后赶紧趁着思路清晰在 16 号写完了实验报告。

控制台 ✕ Add\_F.java Add\_G.java \*Client.java friend.ja

Client (6) [Java 应用程序] E:\JDK\bin\javaw.exe (2021年3月12日 下午6:04:48)  
[java.lang.NullPointerException](#)



出现空指针问题之后，我在网上找到了引起空指针错误的所有致因，一个一个排除：

- i. 字符串变量未初始化
- ii. 接口类型的对象没有用具体的类初始化，比如：

```
Map map // 会报错
```

```
Map map = new Map(); //则不会报错了
```

- iii. 当一个对象的值为空时，你没有判断为空的情况。
- iv. 字符串与文字的比较，文字可以是一个字符串或 Enum 的元素，如下会出现异常

```
String str = null;

if (str.equals ( "Test" ) ) {

//这里的代码将不会被触发，因为会抛出

java.lang.NullPointerException 异常。

}
```

- v. 优先使用 String.valueOf ( ) 方法代替 toString ( )  
当程序代码需要对象的字符串表示形式时，请避免使用该对象的 toString 方法。如果你的对象的引用等于 null，NullPointerException 则会抛出，使用静态 String.valueOf 方法，该方法不会抛出任何异常并打印“null”
- vi. class 被声明了类型， 默认 class = null；这样在调用 class 中方法的时候系统只能给你个空指针异常， 给其实例化就好了：class = new Class();
- vii. 返回 null，方法的返回值不要定义成为一般的类型，而是用数组。这样如果想要返回 null 的时候就能避免许多不必要的 NullPointerException

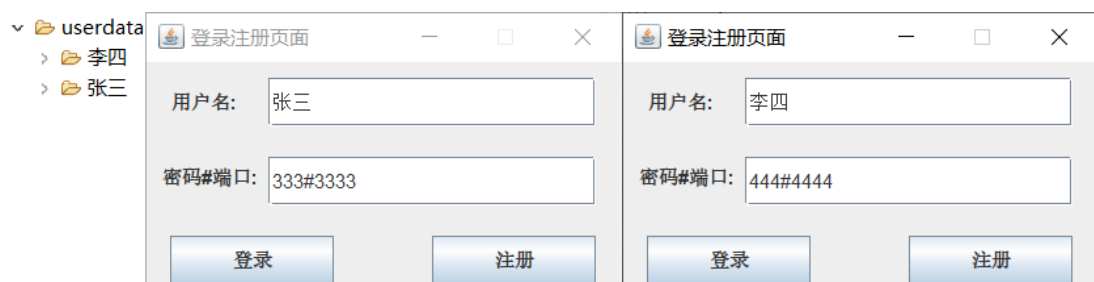
最可怕的是，这些可能的问题居然都可能存在，因为这些方法我都使用过了，但是最后发现原因是在 loginOrRegister 模块中，我对注册模块的信息传输打错了 Register，导致传到 Server 端的信息走不同任何一个路，后来我在 Server 端多加了一条路，也就是 Register 和 login 都不是的情况，这才检查出来。就这个小错误花了我四天时间，回想起来还是欲哭无泪。

## 七、设计结果

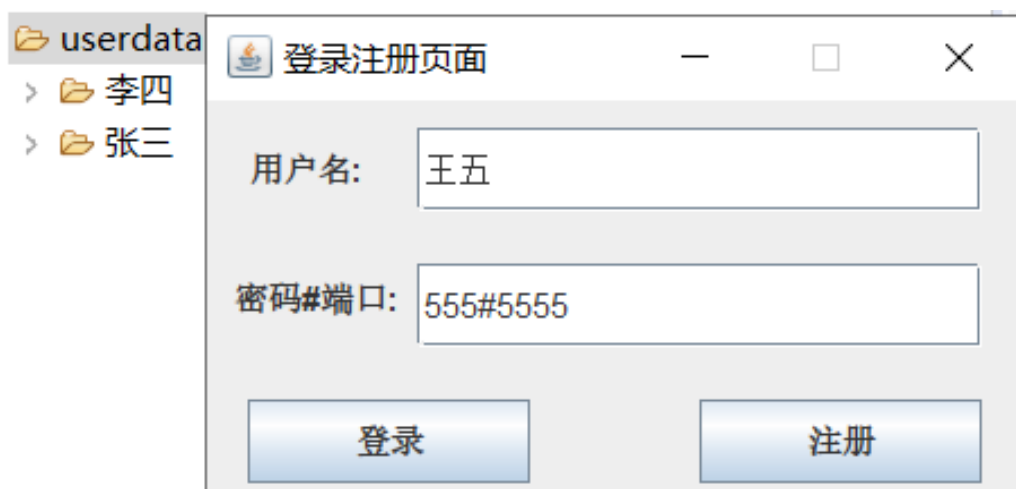
- 1) 运行服务器 Server.java，待弹出服务器窗口则说明服务器启动完毕；



- 2) 运行客户端 Member.java，此时将弹出用户登陆/注册界面，注意密码栏内容应该是“密码#端口”的格式，新用户需要选择注册，老用户直接登陆即可，可以看到用户数据库里有张三和李四的信息，所以我们直接按照正确的用户名、端口、密码登陆即可；



为了测试注册功能，我们再新注册一个用户王五：



注册成功后，刷新，可以看到数据库中有新用户王五出现。



3) 登陆/注册成功后将弹出聊天室主界面，界面上有四个按钮：好友列表、群组列表、添加好友、添加群组。

- 点击“好友列表”按钮可以刷新查看用户的好友情况，按下好友后的“开始私聊”按钮可以开始与好友私聊；
- 点击“群聊列表”按钮可以刷新查看用户的群组情况，按下群组后的“开始群聊”按钮可以开始群组内群聊；
- 点击添加好友按钮可以打开添加好友的页面；
- 点击添加群组按钮可以打开添加群组的页面；

由于张三李四是老用户，已经互相加好友了，所以在他们的好友列表可以看到彼此信息：



然后我们用王五的账号来加张三好友：



添加后，刷新王五的好友列表：

用户王五主界面

好友列表

群聊列表

添加好友

加入群聊

好友IP地址:127.0.0.1

好友用户名:张三

开始私聊

- 4) 好友私聊页面内，在输入框可以输入自己要说的话，之后点击“发送”按钮即可发给好友。点击“退出”按钮即可退出好友私聊界面；
- 我们测试一下互为好友的张三和李四聊天：

与李四的私聊页面

与李三的私聊页面

张三:李四，最近好吗？  
李四:还行吧，在做算机网络课设，很有挑战性

输入框

退出 发送

张三:李四，最近好吗？  
李四:还行吧，在做算机网络课设，很有挑战性

输入框

退出 发送

- 5) 群组群聊页面内，在输入框可以输入自己要说的话，之后点击“发送”按钮即可发进群聊。点击“退出”按钮即可退出群组私聊界面。
- 为了测试群组功能，我们需要让张三、李四、王五都加入到群组“计算机网络群”里面：

用户张三主界面

用户李四主界面

用户王五主界面

好友列表

群聊列表

添加好友

加入群聊

好友IP地址:127.0.0.1

好友用户名:李四

开始私聊

加入群组

请输入群组信息

群组IP地址: 224.0.0.1

群组名称: 计算机网络群

加入

好友列表

群聊列表

添加好友

加入群聊

好友IP地址:127.0.0.1

好友用户名:张三

开始私聊

加入群组

请输入群组信息

群组IP地址: 224.0.0.1

群组名称: 计算机网络群

加入

好友列表

群聊列表

添加好友

加入群聊

好友IP地址:127.0.0.1

好友用户名:张三

开始私聊

加入群组

请输入群组信息

群组IP地址: 224.0.0.1

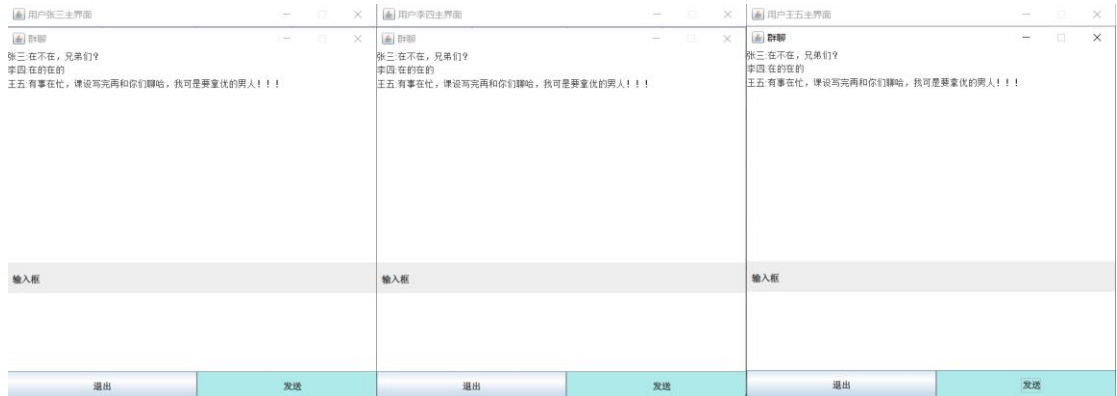
群组名称: 计算机网络群

加入

都加入后，我们刷新三位用户的群组列表，可以看到计算机网络群已经作为他们加入的群组显示出来了：



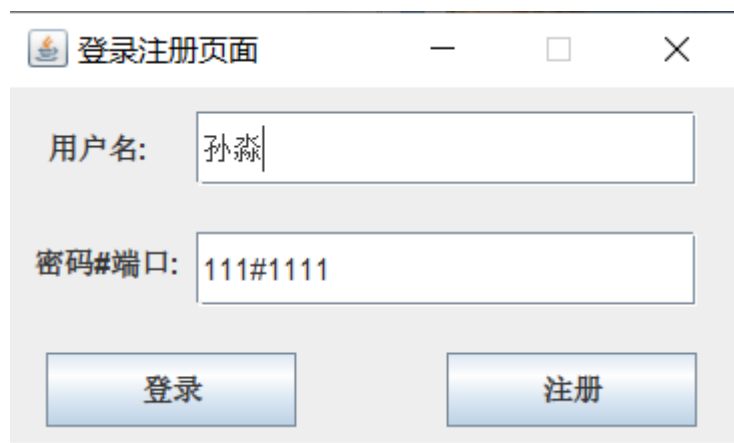
接下来让他们都进入群聊，测试是否能一起群聊：



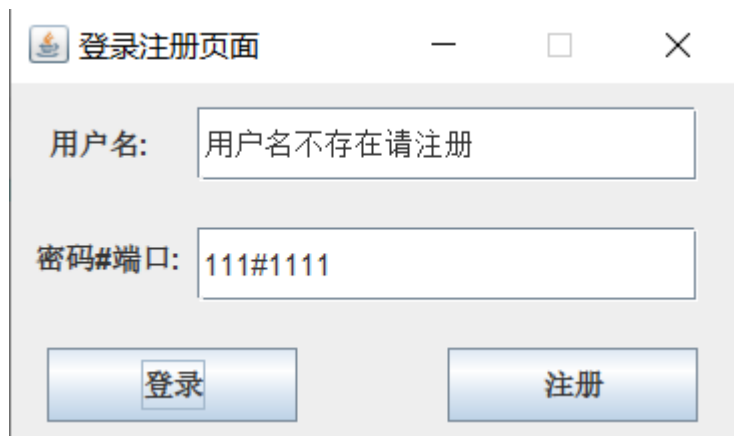
结果如上图所示，群聊功能完全完成。

下面补充一些没介绍的功能：

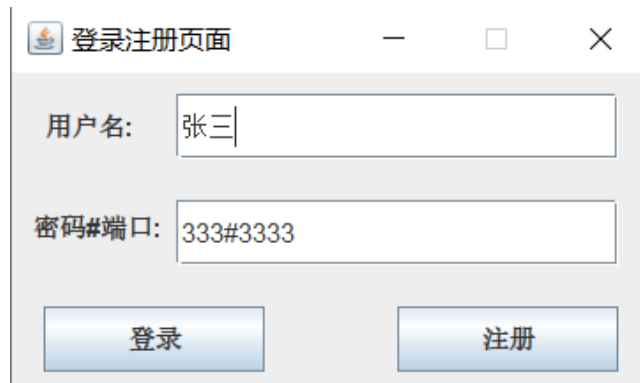
倘若我们在登陆界面登陆一个未注册的用户：



则系统会提示“用户名不存在”



倘若我们注册一个已有的用户：



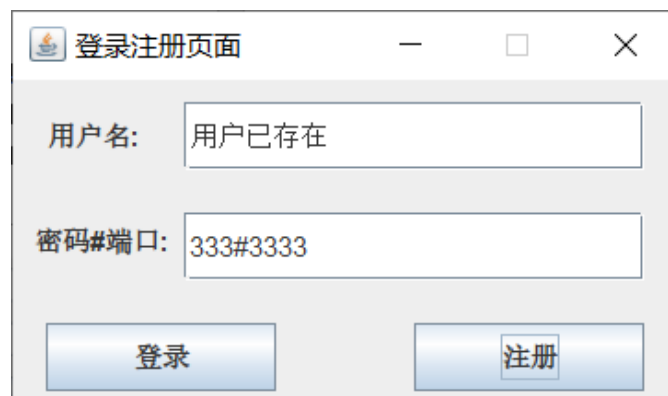
登录注册页面

用户名: 张三

密码#端口: 333#3333

登录 注册

则系统会提示“用户已存在”



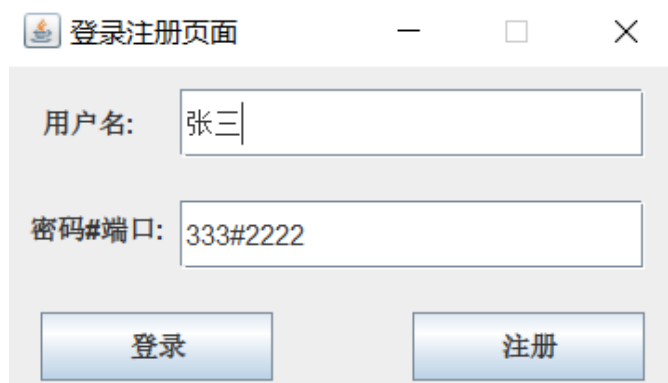
登录注册页面

用户名: 用户已存在

密码#端口: 333#3333

登录 注册

倘若我们输入已有用户的密码错误:



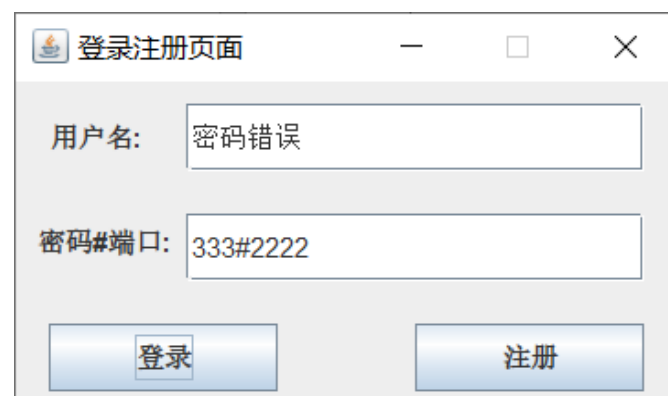
登录注册页面

用户名: 张三

密码#端口: 333#2222

登录 注册

则系统会提示“密码错误”



登录注册页面

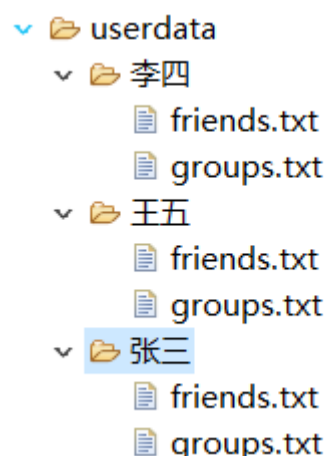
用户名: 密码错误

密码#端口: 333#2222

登录 注册



根据上述运行之后产生的用户数据库如下所示：



我们根据之前的测试可知，张三和李四是好友，那么打开张三的 friends.txt 文件可以看到：

```
1 127.0.0.1@李四@4444
2
```

打开李四的 friends.txt 文件可以看到：

```
1 127.0.0.1@张三@3333
2
```

同时，三位用户都在群组“计算机网络群”中，我们打开三位的 groups.txt 文件，显示都如下：

```
1 127.0.0.1@计算机网络群
2
```

## 八、软件使用说明

- 1) 运行服务器 Server.java，待弹出服务器窗口则说明服务器启动完毕；
- 2) 运行客户端 Member.java，此时将弹出用户登陆/注册界面，注意密码栏内容应该是“密码#端口”的格式，新用户需要选择注册，老用户直接登陆即可；
- 3) 登陆/注册成功后将弹出聊天室主界面，界面上有四个按钮：好友列表、群组列表、添加好友、添加群组。
  - 点击“好友列表”按钮可以刷新查看用户的好友情况，按下好友后的“开始私聊”按钮可以开始与好友私聊；
  - 点击“群聊列表”按钮可以刷新查看用户的群组情况，按下群组后的“开始群聊”按钮可以开始群组内群聊；

- 点击添加好友按钮可以打开添加好友的页面;
  - 点击添加群组按钮可以打开添加群组的页面;
- 4) 好友私聊页面内, 在输入框可以输入自己要说的话, 之后点击“发送”按钮即可发给好友。点击“退出”按钮即可退出好友私聊界面;
  - 5) 群组群聊页面内, 在输入框可以输入自己要说的话, 之后点击“发送”按钮即可发进群聊。点击“退出”按钮即可退出群组私聊界面。
  - 6) 添加好友界面, 需要分别输入好友 IP 地址、好友用户名、好友端口。需要注意的是, 由于是一机测试, 输入的 IP 地址的范围必须是 127.0.0.1--->127.255.255.254, 即回环地址;
  - 7) 添加群组界面, 需要分别输入群组 IP 地址、群组名字。需要注意的是, 输入的 IP 地址的范围必须是 224.0.0.1--->239.255.255.255, 即组播地址。

## 九、参考资料

- 
- [1]谢钧, 谢希仁. 计算机网络教程[M]. 人民邮电出版社:, 201409. 350.
  - [2]王林. 基于 C/S 模式的高并发局域网聊天系统设计[D]. 合肥工业大学, 2020.
  - [3]毕娜. 局域网聊天室系统的设计与实现[J]. 福建电脑, 2016, 32(05)
  - [4]陈健苇. 基于 WinSock 的局域网聊天室设计与实现[J]. 数字技术与应用, 2013(02)
  - [5]李竹林, 李丹霞. 基于 Winsock 的局域网聊天系统的设计与实现[J]. 河南科学, 2012, 30(10)
  - [6]黄海芳, 宋筱媛. 基于 UDP 组播的局域网聊天室设计[J]. 福建电脑, 2008(04)

## 十、验收时间及验收情况

验收时间: 3月17日上午9:50 - 10:15

验收情况: 通过, 但是遇到一些小问题自己设计的时候由于经验不足没有考虑到, 待老师点拨之后才想到, 下面我将介绍验收情况, 并且将验收时的问题及解决过程说明如下:

验收开始时, 郑老师先是要求我展示程序功能: 服务器的打开、客户端的注册/登陆、好友的添加、群组的添加、好友的私聊都没有任何问题, 直到群组的群聊的时候, 突然发不出来了。

情况如下图所示：



报错是：java.net.SocketException: error setting options;

于是我开始查阅资料看是什么原因导致的，网上资料很多，来回确认之后我恍然大悟，原来是没连网就进行配置，结果报错！和老师确认之后，老师指出组播地址需要 web 配置，这真是“纸上得来终觉浅，绝知此事要躬行”，确定了导致错误的原因，我打开手机热点，电脑再连上之后，果然就可以开始群聊了。



此次验收收获很大，感谢老师指出我因为经验不足而设计不好的地方！

## 十一、设计体会

在为时 8 天的设计过程中，我遇到了许多问题，终于在 16 号晚上完成了课设，心里十分舒畅。

由于之前许多实验都是不需要图形界面的，所以我一直使用的是自己更熟练的 C++ 编写程序，于是这次刚开始我也是选择 C++ 作为语言，但是开始设计之后，我就发现不对劲了，一是这次设计聊天室软件是需要很多界面的，比如登陆/注册界面、聊天室界面、添加好友/群组界面、好友/群组聊天界面，这些都用 Qt 设计的话很麻烦，二是局域网聊天室涉及 TCP/UDP 传输的，我从来没有在 C++ 中用过这些相关的库，但是我在初次接触 Java 时就学习过套接字相关的知识，计算机网络课时内还布置过一次套接字相关的实验，这些为这次课设都打下过基础。

考虑到上面提到的情况，我及时选择换成 Java 来完成这次课设，节省了大量时间。

此外，设计一个较复杂的程序是需要好好从头分析的，对于这次课设而言，服务器和客户端自然是必不可少的，然后考虑到局域网聊天室的功能，那么私聊和群聊也是需要的，私聊是与好友之间进行的，那么好友从何而来？所以就要有添加好友的功能。群聊是在群组内进行的，那么群组从何而来？所以就要有添加群组的功能。好友和群组不可能是直接全部打开的，所有要有一个好友和群组信息的载体，于是聊天室主界面也就应运而生了，聊天室需要根据不同用户载入不同的信息，如何区分不同用户？所以登录/注册模块就出现了，就是按照这样的思路，我完成了对这次课设的需求分析。这些想清楚之后，整个课设就会变得简单起来。

这次设计最难熬的时候是在 3 月 12 号下午，出现了 Java 中令人闻风丧胆的空指针错误，而且也不显示错误行号，这让我检查了整整 4 天，终于在 15 号下午查出了原因。最后调试的时候都已经不抱希望了，结果居然跑通了，那一刻真是心情舒畅，然后赶紧趁着思路清晰在 16 号写完了实验报告。

以前遇到这种很久都找不出原因的 bug，我一般都会让朋友帮我看看，有时候就能看到我没注意的错误（毕竟当局者迷，旁观者清），但是因为大三周围的朋友都在准备考研，而我准备出国读研，所以也不好意思麻烦他们帮我看看。这

是我找 bug 最久的一次，也绝对是我人生宝贵的财富，以后遇到 bug，我肯定都会想到这次 4 天才找到的 bug，也就不会再有畏难的心理了。