

**计算机网络课程设计报告**



学 号 202103151406

姓 名 谢君艺

班 级 软件工程06

提交日期 2022 年 12 月 31日

目录

[1](#_Toc123123366)

[一． 课程的目的和任务 3](#_Toc123123367)

[1.1课程目标 3](#_Toc123123368)

[1.2 课程思政 3](#_Toc123123369)

[1.3 课程任务 3](#_Toc123123370)

[二． 课程的基本内容和要求 4](#_Toc123123371)

[2.1常用网络命令 4](#_Toc123123372)

[2.1.1 ipconfig 4](#_Toc123123373)

[2.1.2 ping 9](#_Toc123123374)

[2.1.3 netstat 11](#_Toc123123375)

[2.1.4 tracert 14](#_Toc123123376)

[2.1.5 arp 16](#_Toc123123377)

[2.1.6 telnet 17](#_Toc123123378)

[2.2交换机与路由器 19](#_Toc123123379)

[2.2.1交换机命令学习及交换机初始化设置 19](#_Toc123123380)

[2.2.2交换机及VLAN配置 22](#_Toc123123381)

[2.2.3路由器命令学习和路由器登录方法 29](#_Toc123123382)

[2.2.4静态路由配置 35](#_Toc123123383)

[2.2.5动态路由配置（OSPF）并融入综合网络情景设计 43](#_Toc123123384)

[2.3 网络编程 53](#_Toc123123385)

[2.3.1 设计思想 53](#_Toc123123386)

[2.3.2 实验环境 53](#_Toc123123387)

[2.3.3 主要函数 54](#_Toc123123388)

[2.3.4程序流程图 54](#_Toc123123389)

[2.3.5 难点解决 54](#_Toc123123390)

[2.3.6 测试结果截图 55](#_Toc123123391)

[2.3.7 实验结果分析 55](#_Toc123123392)

[2.3.8 程序源代码 56](#_Toc123123393)

[三． 实验总结与心得体会 58](#_Toc123123394)

1. **课程的目的和任务**

**1.1**课程目标

【课程目标1】 理解计算机网络体系结构和工作原理，掌握常用的网络命令的功能，分析命令执行结果，运用常用网络命令进行网络测试与故障检测，得到合理有效结论。

【课程目标2】 能够进行网络组网设计、配置交换机和路由器，能够按照实验方案实施仿真实验，采集和整理数据，并对实验数据进行分析、处理和解释。

【课程目标3】具备网络编程能力，能够进行简单网络编程、掌握设计协议分析程序的方法。

【课程目标4】具备自主学习和创新意识，有不断学习、适应发展的能力。

**1.2** 课程思政

1. 构建方案在实践基础上，要大胆创新，勇于尝试，不断优化迭代，精益求精。
2. 尊重科学，实事求是。对于所设计的方案，能够通过实验去加以求证，理论与实践并重。
3. 团结协作、勇于担当，具有责任心与使命感。

**1.3** 课程任务

任务一、运用网络命令进行网络故障检测分析。掌握计算机网络相关命令原理及应用。常用网络命令ipconfig, ping, netstat, tracert, arp, telnet的功能；在windows环境下使用上述网络命令进行网络状态监测和跟踪，给出相应的截图和对结果的解释。

任务二、基于交换机命令学习及交换机初始化设置的特定任务需求，构建系统方案、实施实验、处理数据并分析实验结果。基于交换机VLAN设置的特定任务需求，构建系统方案、实施实验、处理数据并分析实验结果。基于路由器端口、路由器命令学习和路由器登录方法的特定任务需求，构建系统方案、实施实验、并验证。基于静态路由、动态路由配置的特定任务需求，构建系统方案、实施实验、处理数据并分析实验结果。

(1) 安装packet tracer，在packet tracer仿真环境下，熟悉交换机命令、交换机初始化配置；

(2) 在交换机上实现VLAN配置；

要求：创建三个VLAN，给出拓扑，查看VLAN信息

(3) 基于Console控制台登录配置路由器，学习路由器配置相关命令；

(4) 基于packet tracer构建网络环境，分别进行静态路由配置和基于RIP的动态路由配置，有余力同学设计基于OSPF的动态路由配置。能够进行综合集成网络情景设计更佳。

任务三、基于网络编程的特定任务需求进行socket编程，设计处理流程、数据结构、实施实验并分析实验结果。

编程要求：捕获本机网卡的IP包，对捕获的IP包进行解析。要求必须输出以下字段：版本号、总长度、标志位、片偏移、协议、源地址和目的地址。

要求有详细的说明文档，包括程序的设计思想、工作流程、关键问题、程序注释和对捕获包的解析截图。

编程语言不作要求，可使用自己熟悉的C、C++、java或C#等。

1. **课程的基本内容和要求**

# 2.1常用网络命令

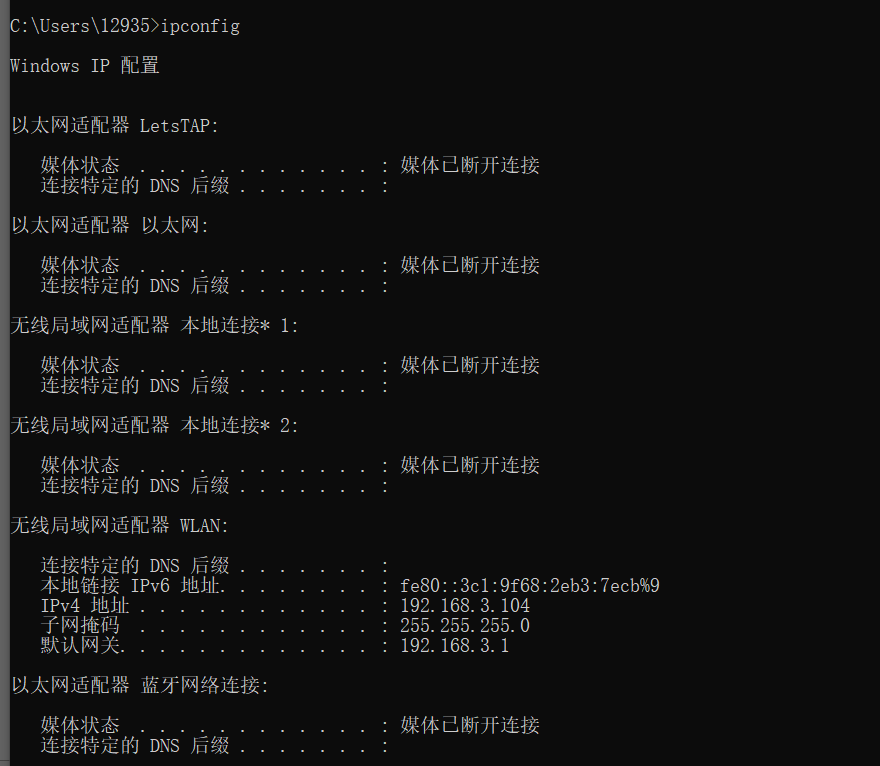
## 2.1.1 ipconfig

功能：显示各种TCP/IP信息包括IP地址，子网掩码，缺省网关值、虚拟机和无线网卡等的基本信息，并刷新或释放DNS和DHCP的缓存或内容。

命令：ipconfig

作用：显示各个接口的连接状况和相关信息（IP 地址、子网掩码等）

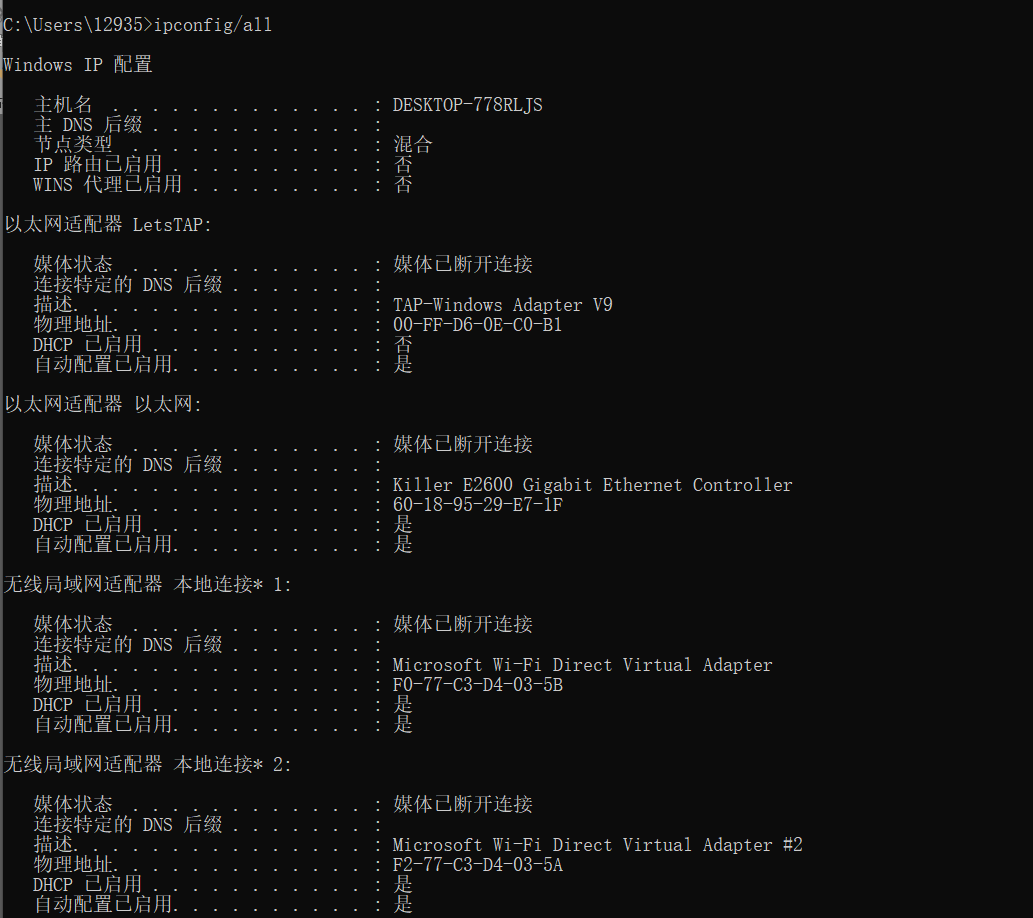
命令截图如下：



命令ipconfig/all

作用：显示设备TCP/IP的详细信息

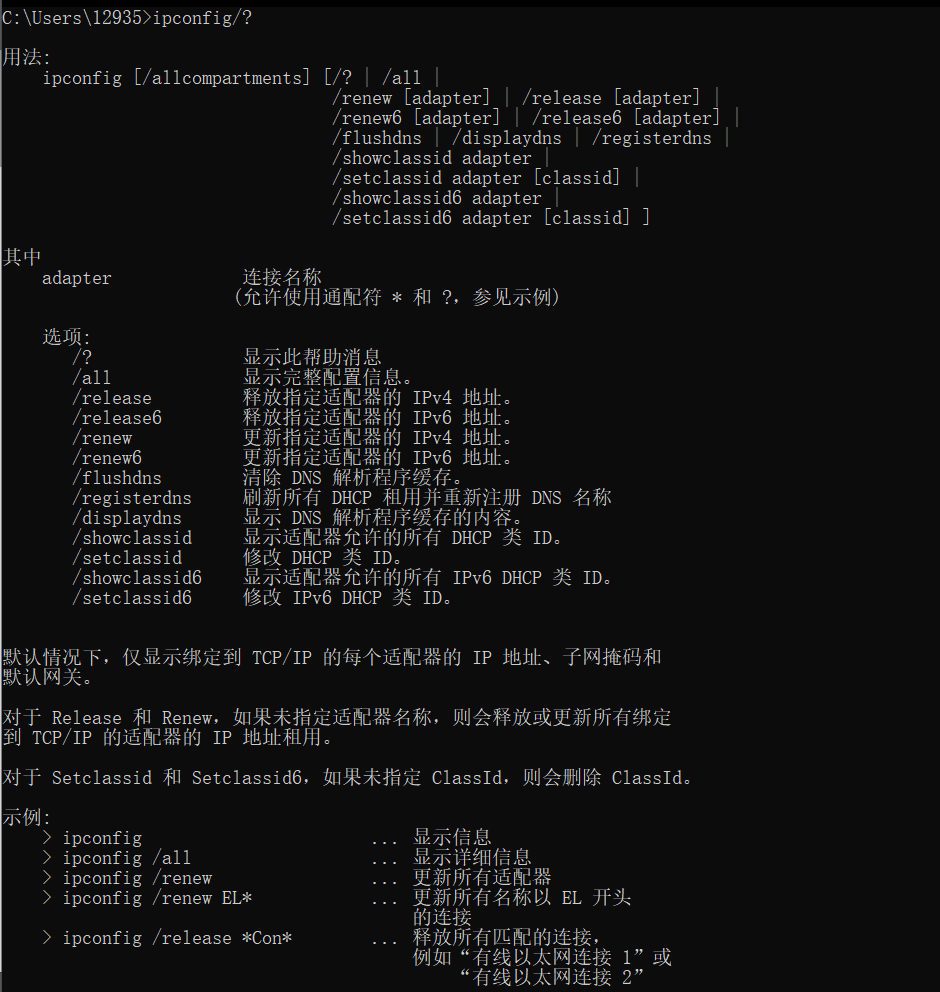
命令截图如下：



命令：ipconfig/？

作用：显示帮助，即用法

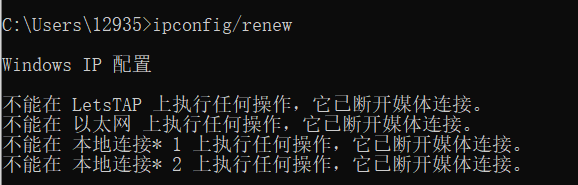
命令截图如下：



命令：ipconfig/renew

作用：手动向服务器刷新请求

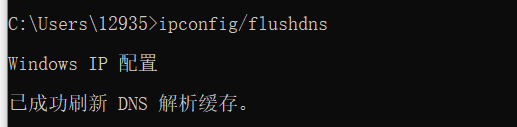
命令截图如下：



命令：ipconfig/flushdns

作用：刷新DNS解析缓存

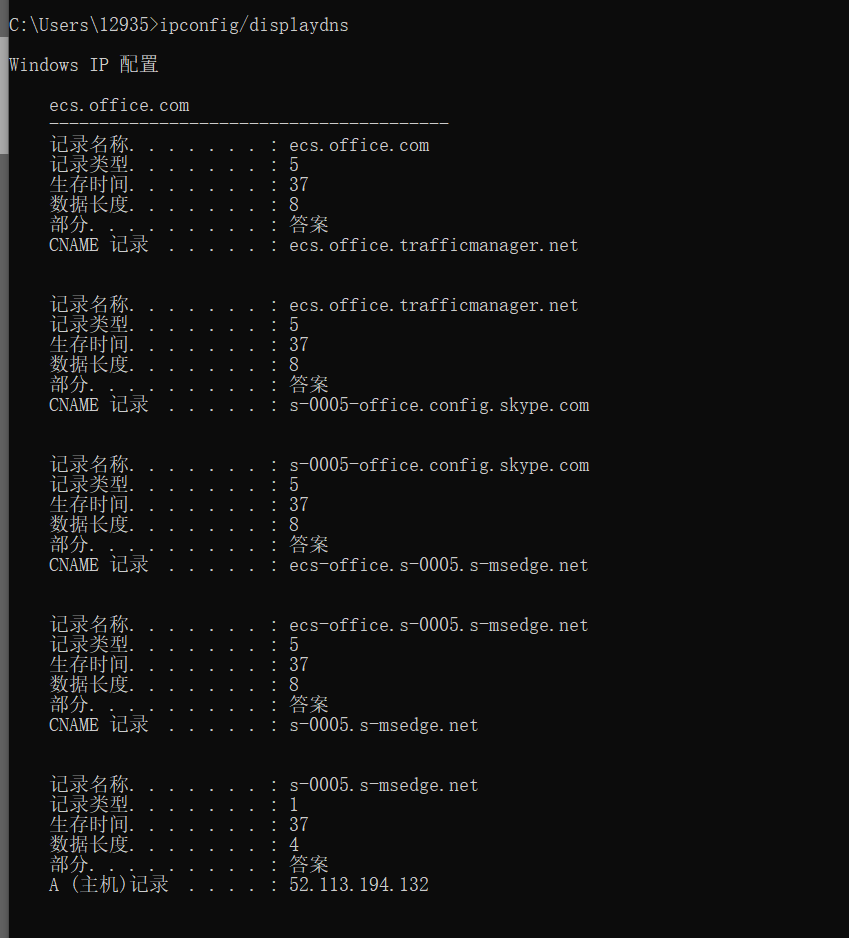
命令截图如下：



命令：ipconfig/displaydns

作用：显示dns内容

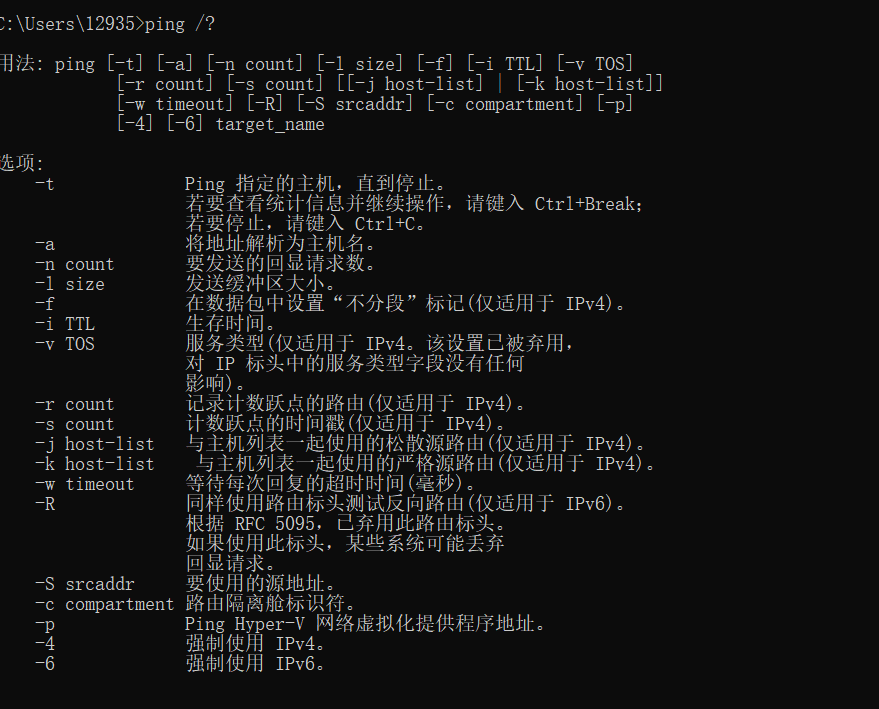
命令截图如下：



## 2.1.2 ping

功能：一种网络诊断工具，利用ICMP用于测试TCP/IP参数是否设置正确等

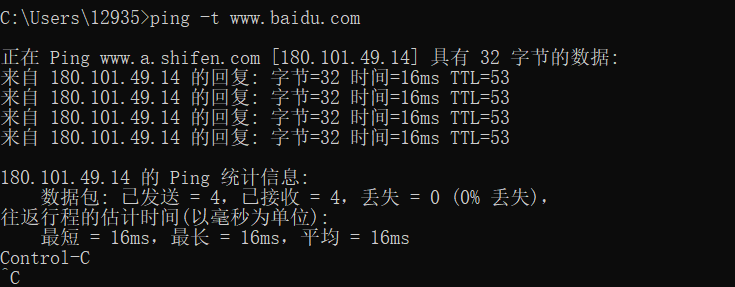
用法：如ping/？命令截图所示：



命令：ping -t

作用：Ping 指定的主机，直到停止。要查看统计信息并继续操作输入Ctrl+Break；要停止，输入 Ctrl+C

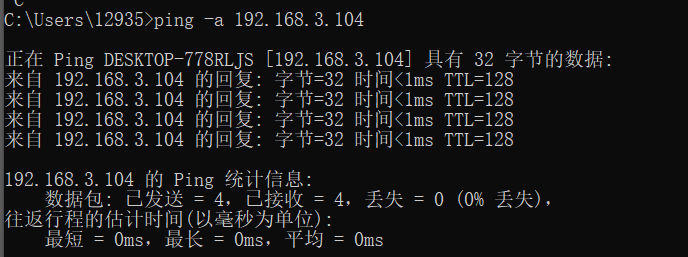
命令截图如下：



命令：ping -a

作用：将地址解析为计算机名

命令截图如下：



命令：ping -n count

作用：发送count个要求回复的请求

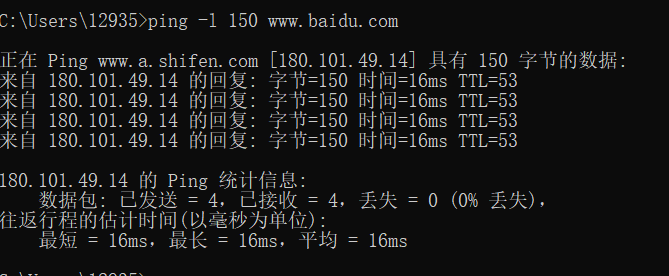
命令截图如下：



命令：ping -l size

作用：发送缓存区大小的包

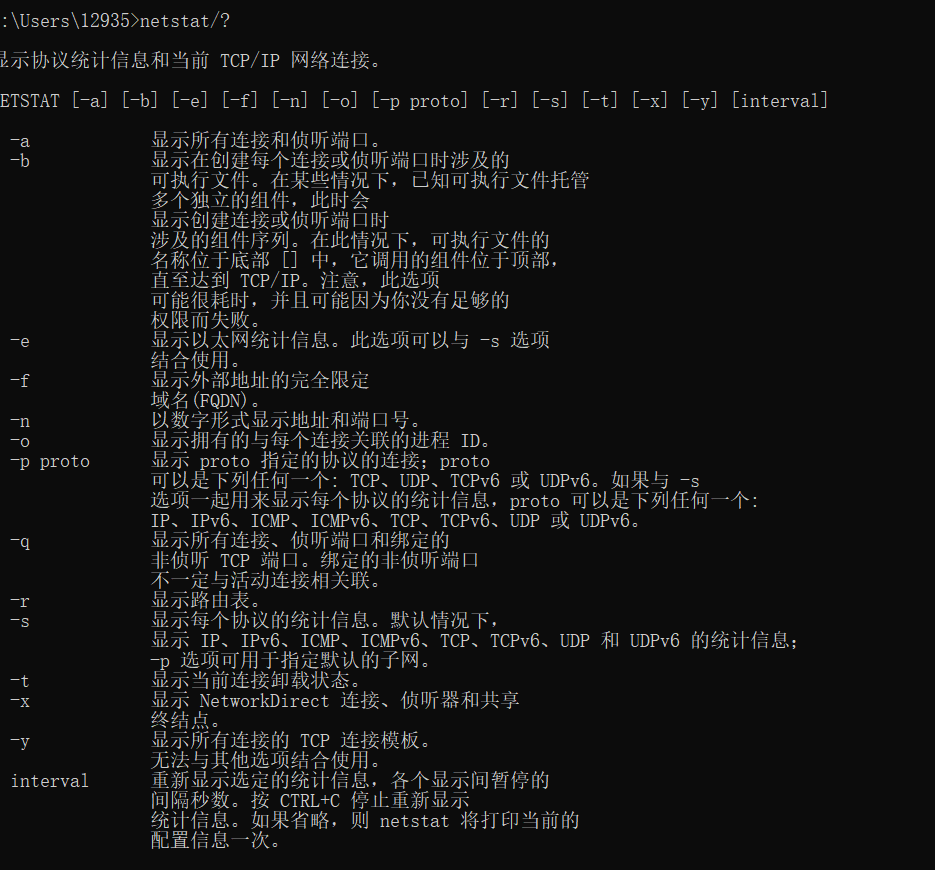
命令截图如下：



## 2.1.3 netstat

功能：用于监控TCP/IP网络，可以显示路由表、实际网络连接以及接口状态信息，让用户知道哪些网络连接正常运作。

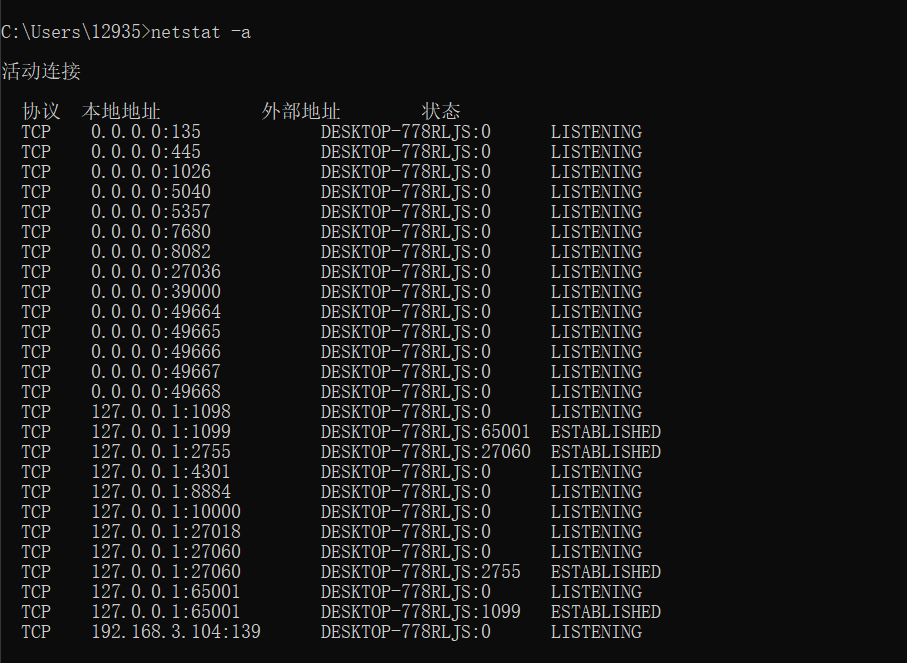
用法：如下图netstat/？指令截图所示：



命令：netstat -a

作用：显示所有socket，对运行的进程状态进行显示

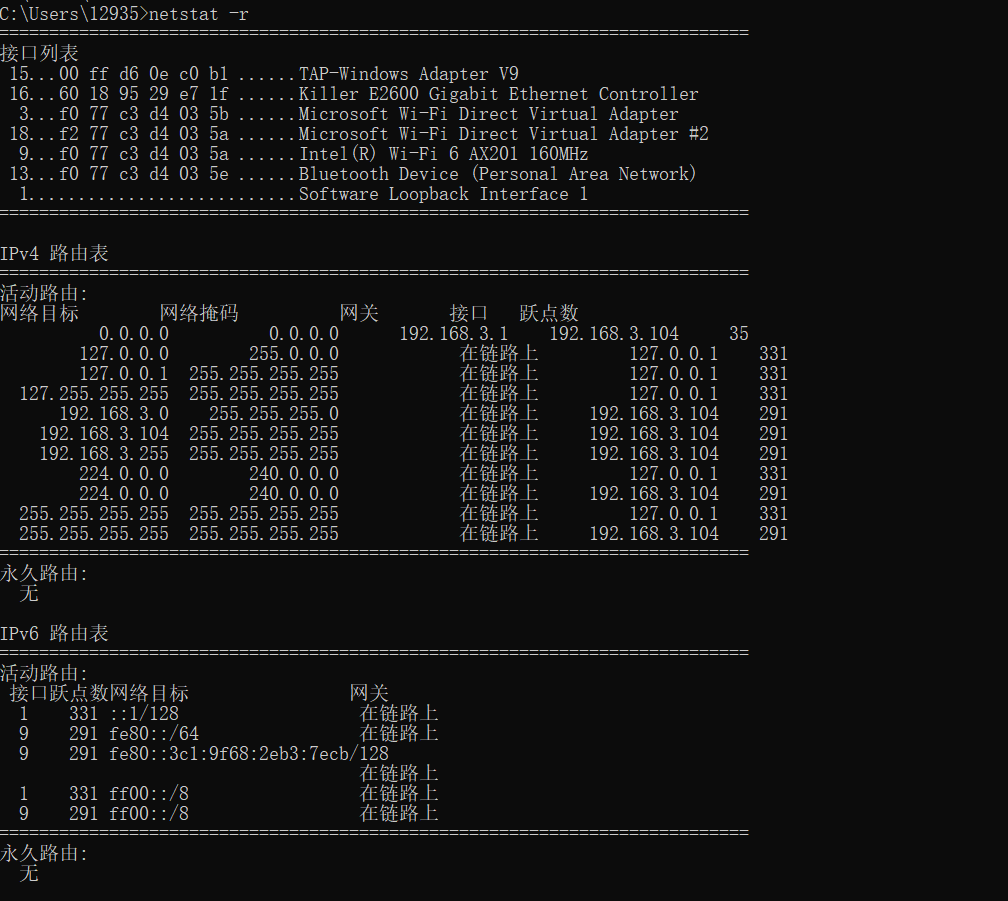
命令截图如下：



命令：netstat -r

作用：显示核心路由表

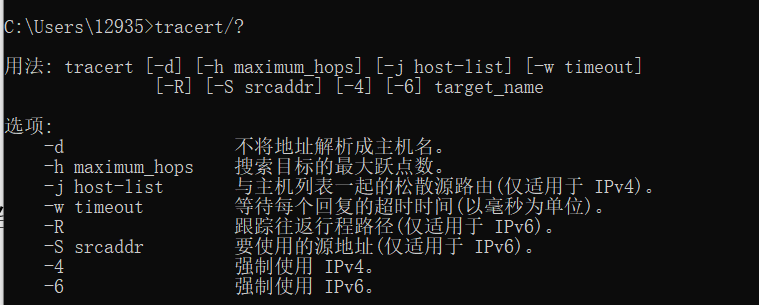
命令截图如下：



## 2.1.4 tracert

功能：是一个简单的网络诊断工具，可以列出分组经过的路由节点，以及它在IP 网络中每一跳的延迟

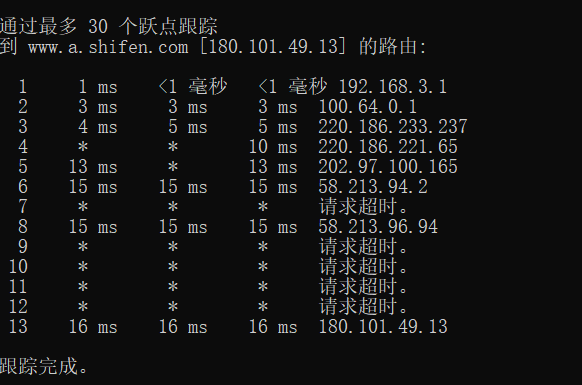
用法：如tracert/？指令截图所示：



命令：tracert -d

作用：不将地址解析成主机名

命令截图如下：



其中，从左到右的 5 条信息分别代表了“生存时间”（每途经一个路由器结点自增 1）、“三次发送的 ICMP 包返回时间”（共计3个，单位为毫秒 ms）和“途经路由器的 IP 地址”（如果有主机名，还会包含主机名）

命令：tracert-h maximum\_hops

作用：指定搜索目标的最大跃点数

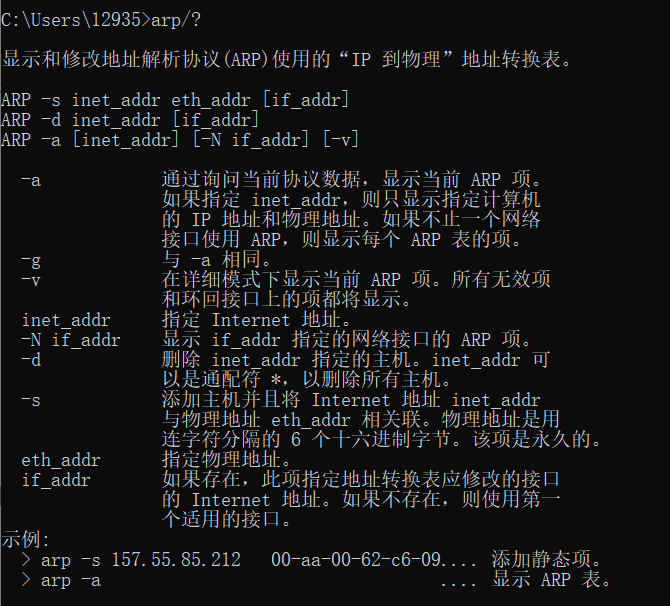
命令截图如下：



## 2.1.5 arp

功能：显示或修改ARP协议使用的IP地址和MAC地址的转换表

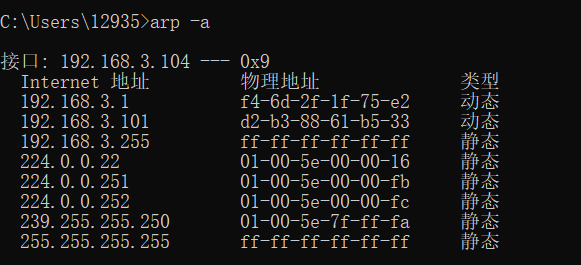
用法：如arp/？指令截图所示：



命令：arp-a

作用：显示每个接口的ARP表的项

命令截图如下：

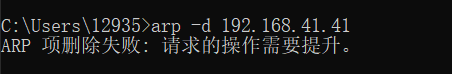


命令：arp -d

作用：删除 inet\_addr 指定的主机。inet\_addr 可以是通配符 \*，以删除所有主机

（权限需要是管理员）

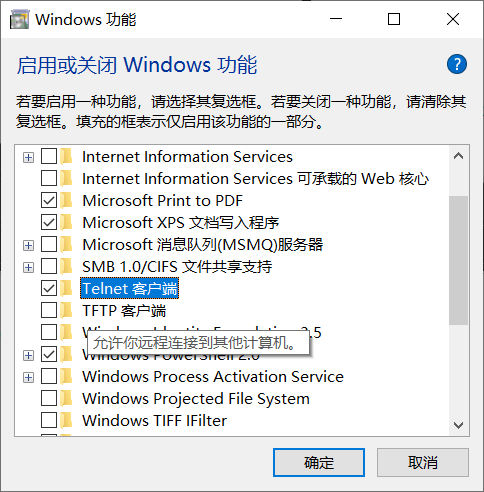
命令截图如下：



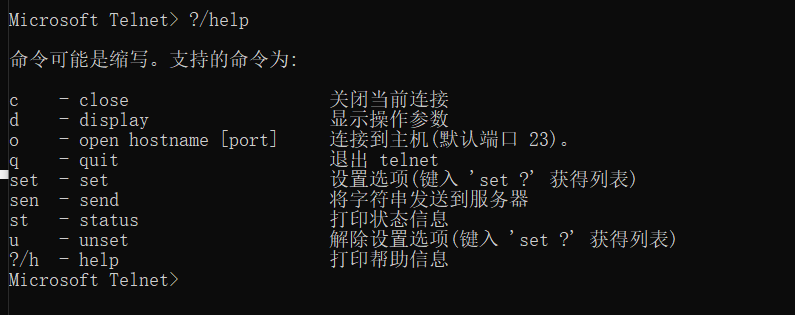
## 2.1.6 telnet

功能：用于远程登录到网络中的计算机，并以命令行的方式远程管理计算机。

注意：先开启telnet客户端才能使用



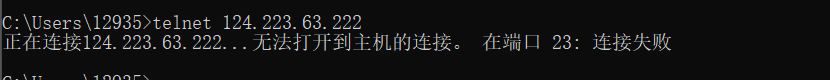
用法：如？/help指令所示：



命令：telnet ip地址

作用：远程连接

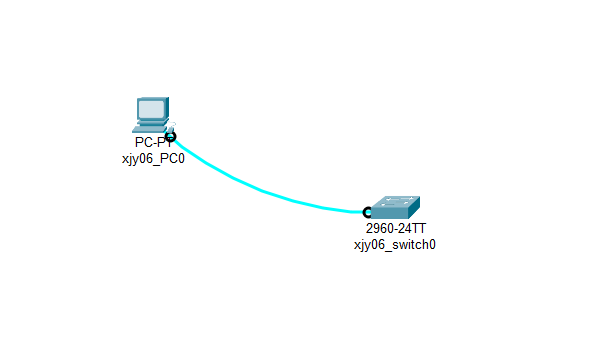
命令截图如下：



# 2.2交换机与路由器

## 2.2.1交换机命令学习及交换机初始化设置

**(1)安装packet tracer**



**(2) 熟悉交换机命令**

交换机相关命令：

用户模式： Switch>

特权模式 ： Switch#

全局配置模式： Switch(config)

端口模式 ： Switch(config-if)#

交换机常用的配置命令行：

enable 进入特权模式（一般简写为 en）

config t 进入全局配置模式（一般简写为conf t）

interface fa 0/1 进入交换机某个端口模式（一般简写为int fa0/1）

exit 返回到上级模式（一般简写为ex）

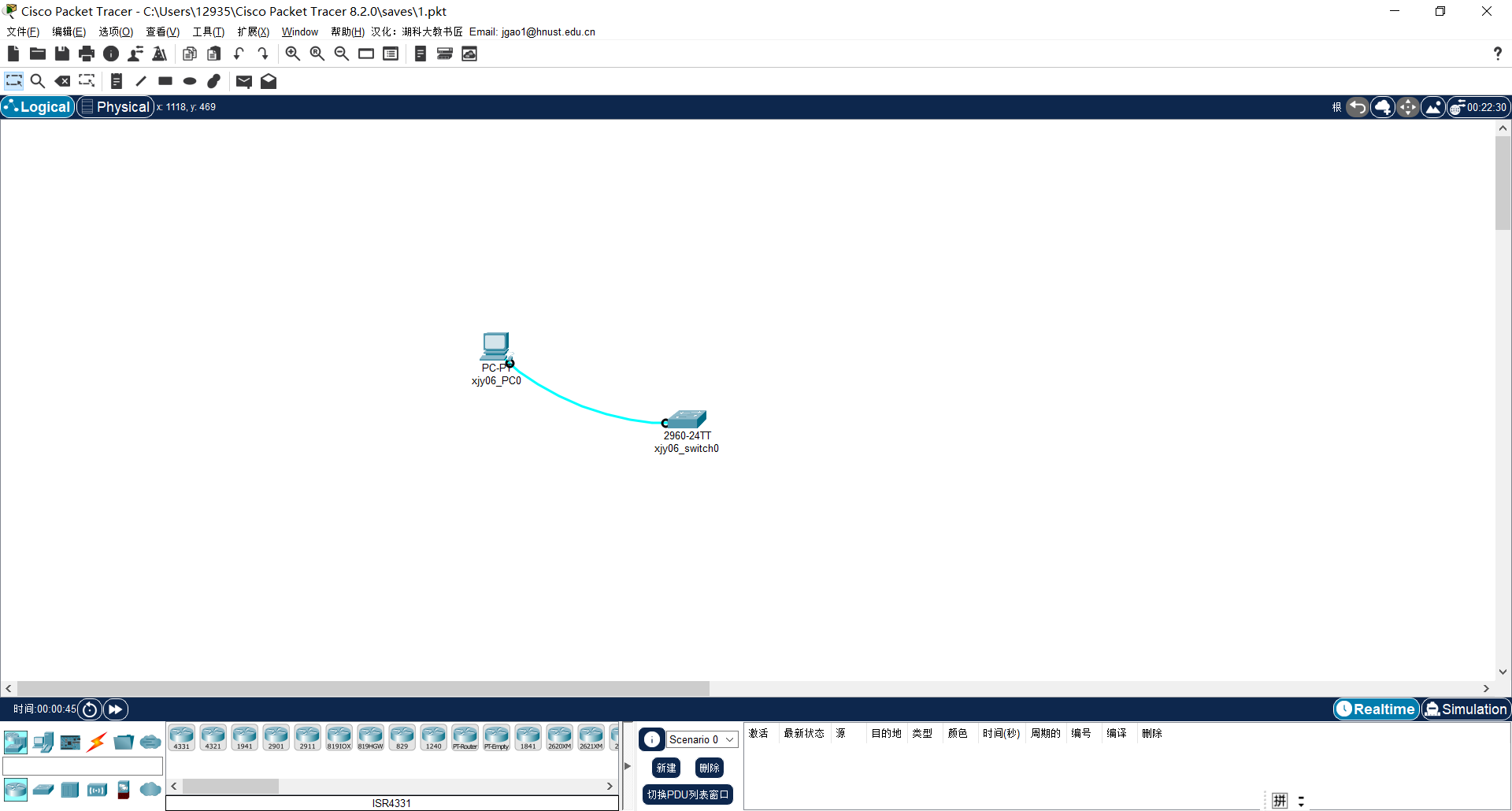
end 从全局以下模式返回到特权模式(一般简写为en)

Switchport access vlan x 将端口接入已经设置好的vlan

Switch mode trunk 将其连接模式改为trunk

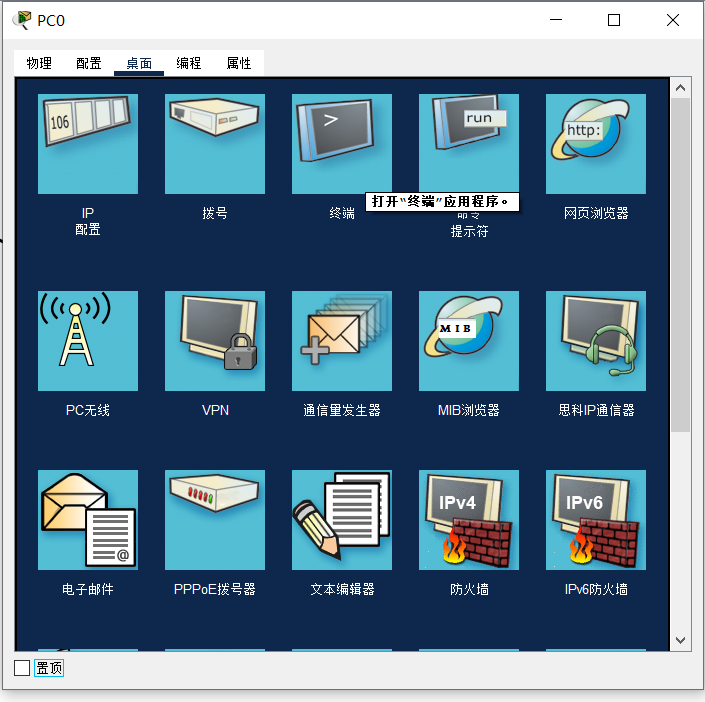
**(3) 交换机初始化配置**

**1.控制台连线**

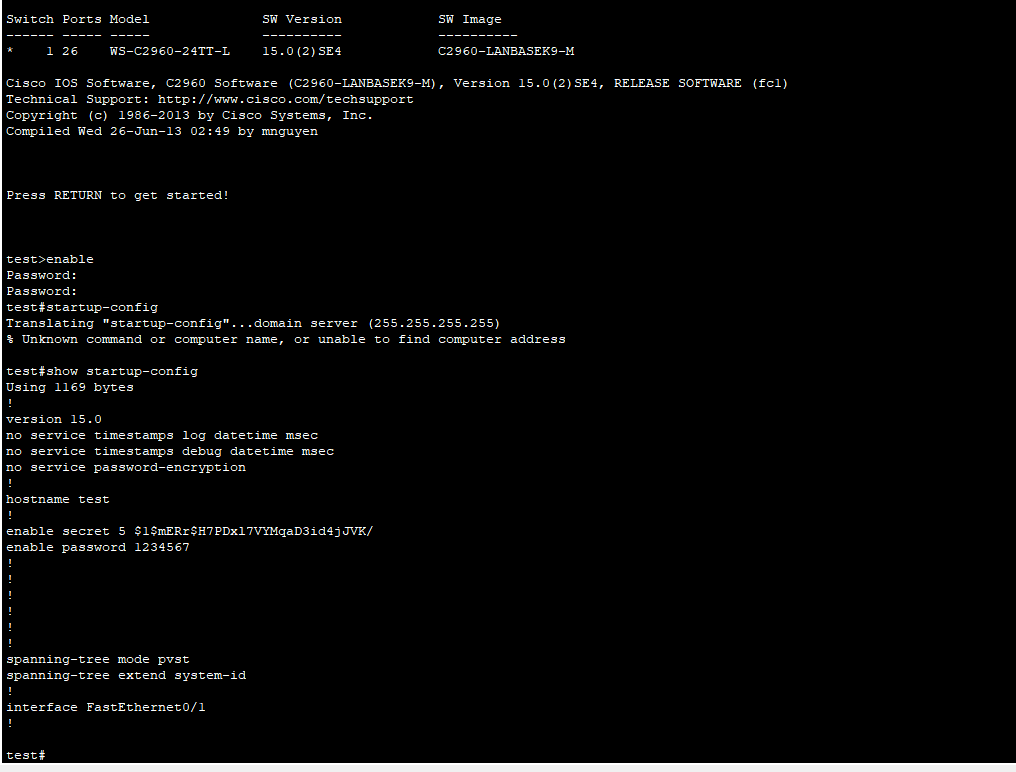


其中，pc0的rs232 连接 switch0 的console

**2.终端选择**



**3.交换机初始化配置**

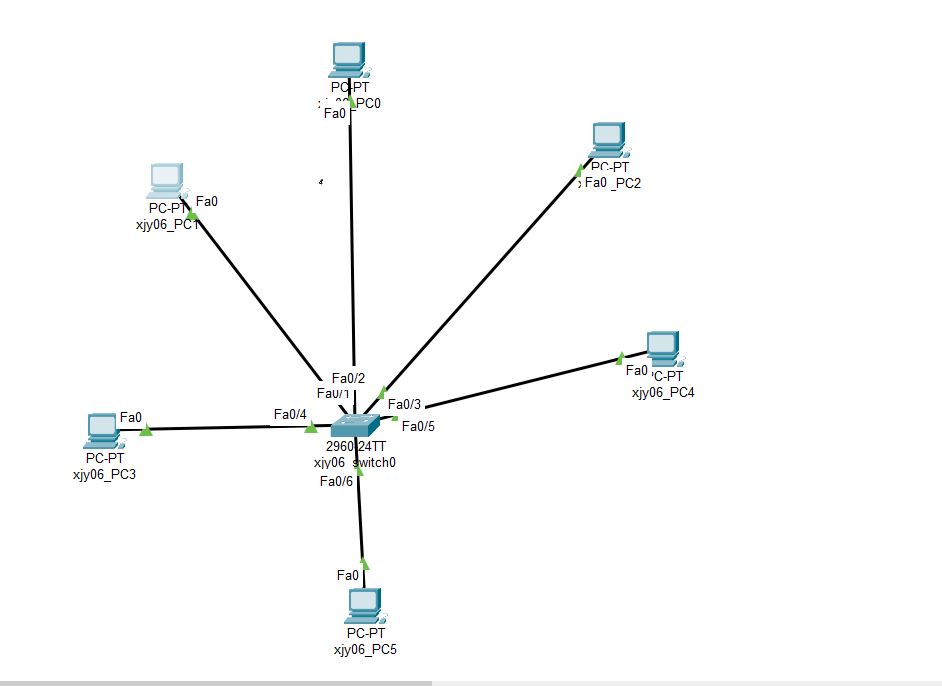


进入enable后，输入setup，初始化配置行用户名和密码的配置，特权模式密码以及IP端口的配置，然后reload，进行show startup-config命令展示配置信息。

## 2.2.2交换机及VLAN配置

**(1) 初始配置及测试**

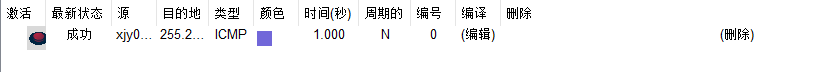
连线，端口为Fa/ pc号+1



如图所示，图中的六个主机分别对应交换机开放的六个主机，并且VLAN都是默认1

下面由主机xjy06\_pc0进行广播





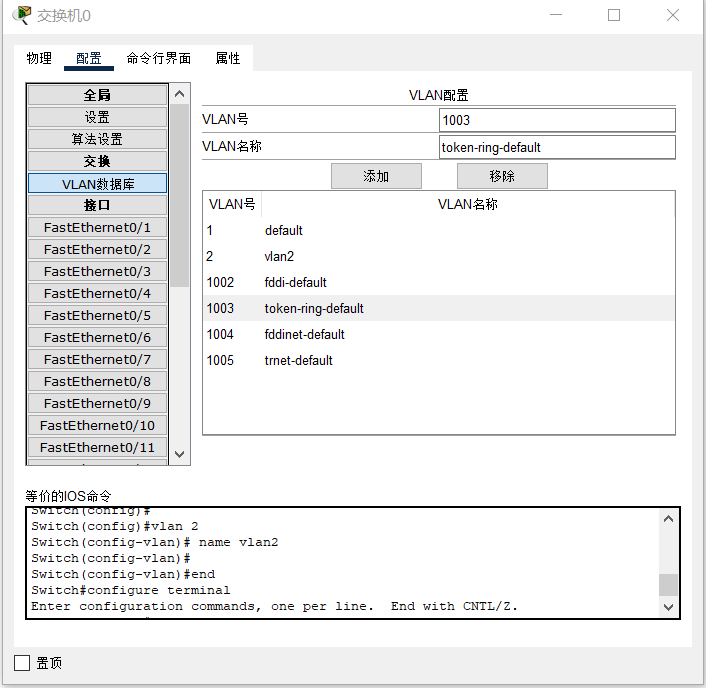
测试结果是正确的，所有pc都能收到IP数据报

**(2) 划分VLAN**

**1.新建vlan2，3**

此处的实现将面板实现和命令行实现都列出，

该实现的面板实现为：



同理，增加vlan3.

该实现的命令行实现为：

创建一个新的 VLAN

Switch>enable //进入特权模式

Switch#config terminal //进入全局配置

Switch(config)#vlan 3 //new 一个新的vlan

Switch(config-vlan)#name vlan3 //命名

Switch(config-vlan)#end

Switch#show vlan brief //查看 vlan 信息

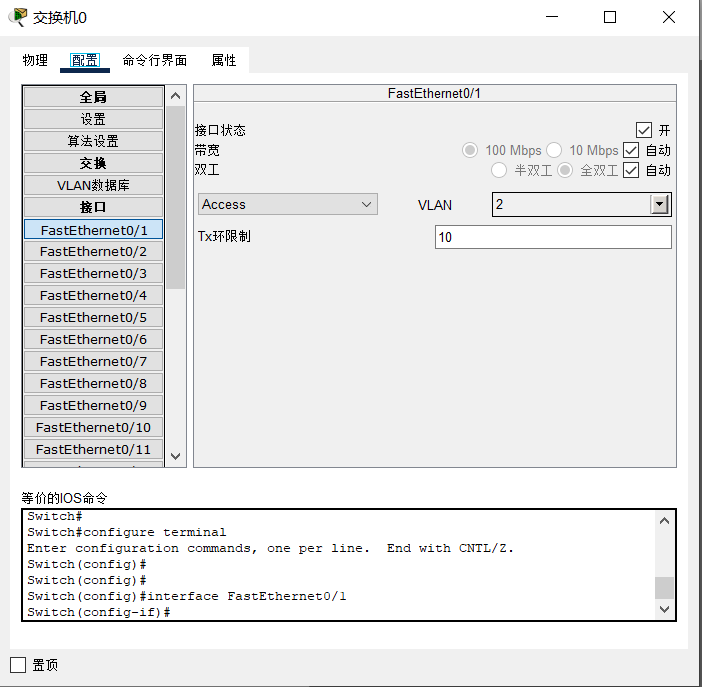


**2.将端口加入VLAN**

本文将端口fa1，2加入vlan2，fa5，6加入vlan3

此处的实现将面板实现和命令行实现都列出，

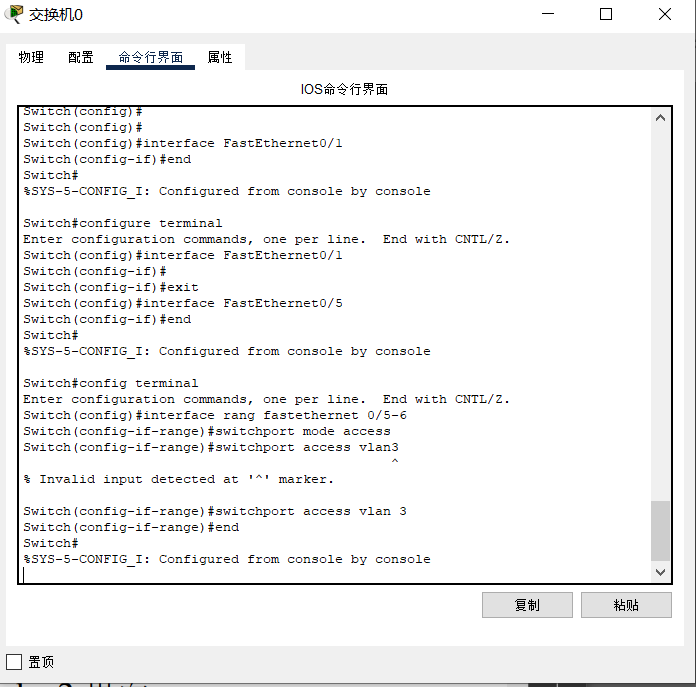
该实现的面板实现为：



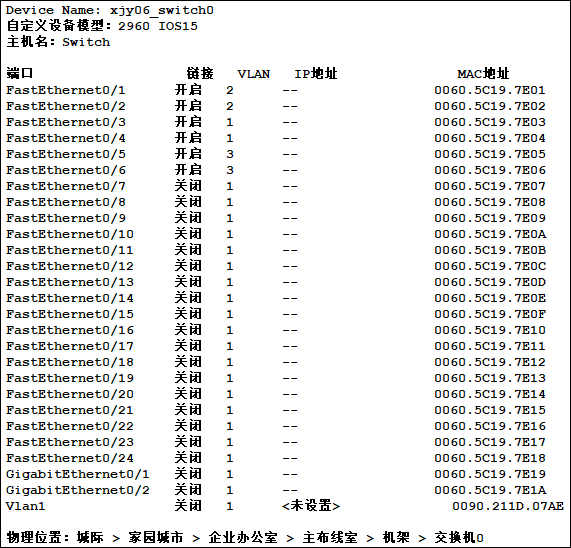
同理将f2 加入 vlan2

该实现的命令行实现为：

此处将5，6加入 vlan3



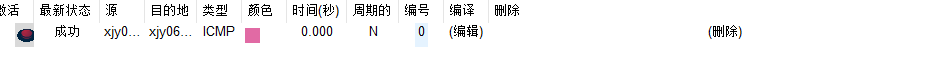
此时的端口状态



**(3) 测试新的vlan**

依旧从pc0发送ip数据报，发现不同的vlan之间不互通。

Pc0 -> pc1



Pc0 ->pc2



## 2.2.3路由器命令学习和路由器登录方法

**(1) 路由器相关命令**

进入：config terminal/memory/network

配置网络时常采用的命令：copy 和 load

1. 标识：hostname 标识名
2. 启动标识：banner 启动标识
3. 接口：interface 端口号
4. 密码：line 0 6

login

passwd

enable password/secret

1. 接口：

1）配置端口

interface 端口号

clock rate 时钟速率（64000）在来配置serial串口的，起同步作用，这个时钟频率和bandwidth带宽相关，单位是位每秒bps，带宽多大就多大

bandwidth 设置带宽（缺省 56）在串口中配置

media-type 介质类型 在以太网口上 fa

no shutdown //不关闭

write memory

2）检验端口

show interfaces

show controllers

6． 查看邻居路由

show cdp interface

show cdp neighbors [detail]

show cdp entry routerA

7． IP Address 配置

Ip address 网络地址 掩码

Ip host 主机名 address

Ip domain-lookup nsap

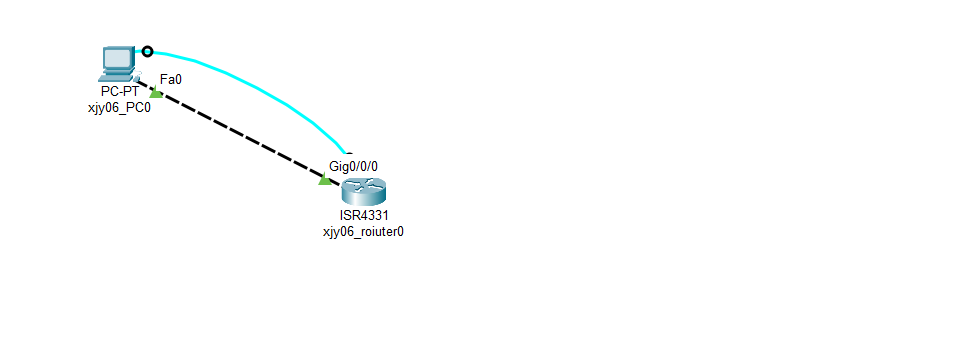
Show hosts

Ping 主机名/IP 地址

**(2) 基于console路由器基础配置**

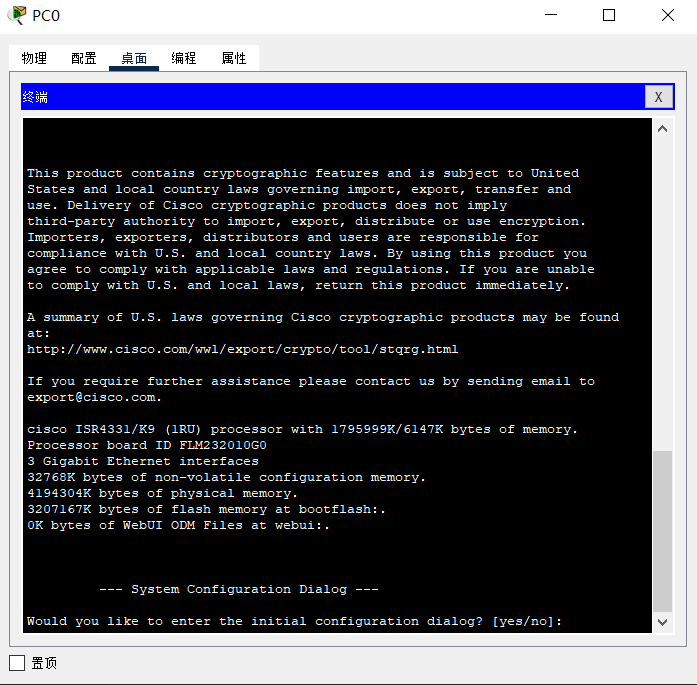
**1．接线**

利用一台主机一台路由器做简单的接线，利用控制台线将它们相连。

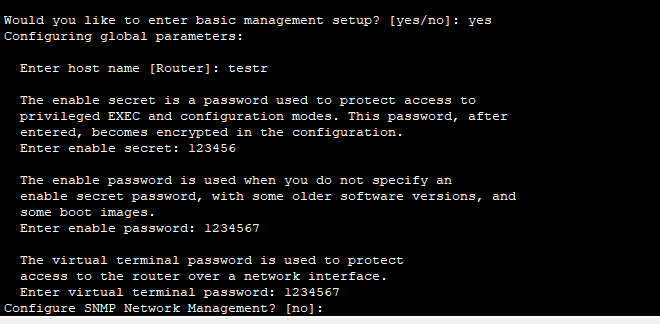


**2．配置**

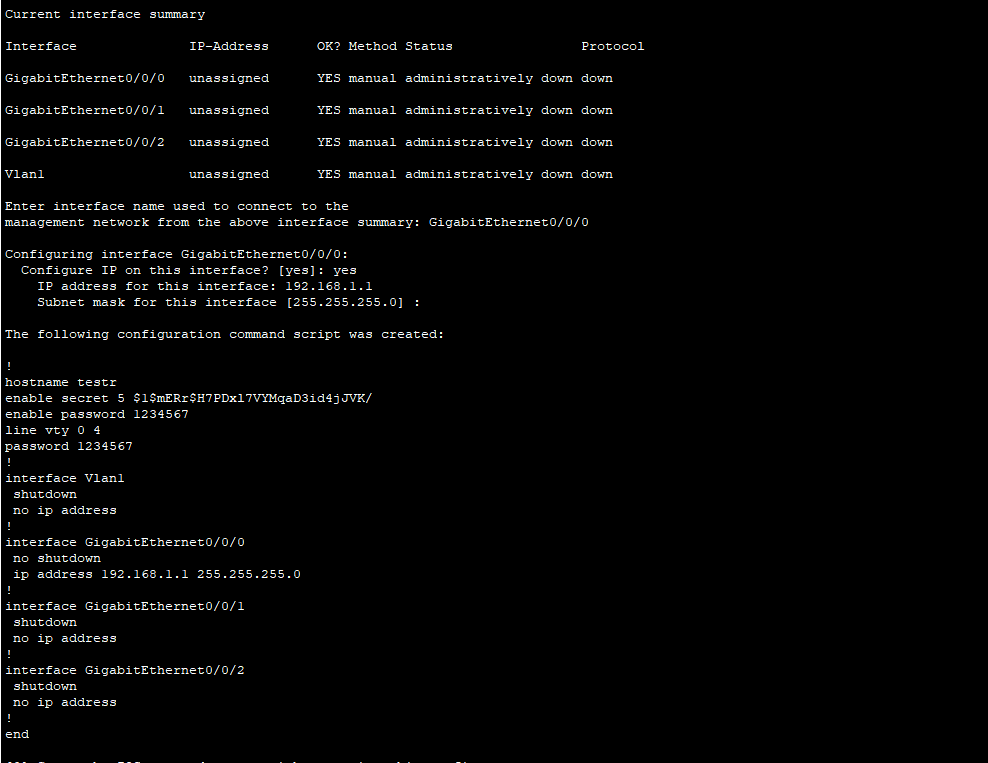
进入pc0的终端



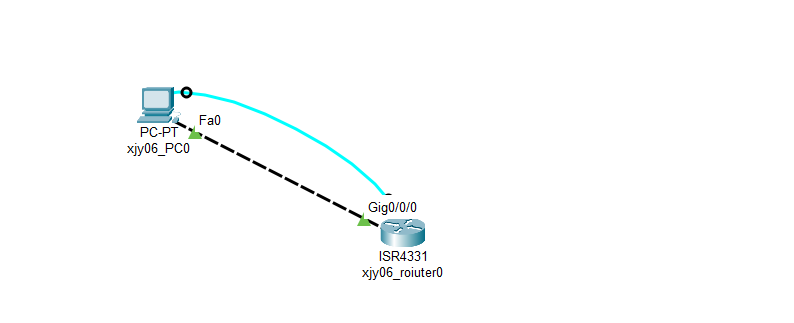
依旧是路由器名以及相关密码的填写



然后是端口ip

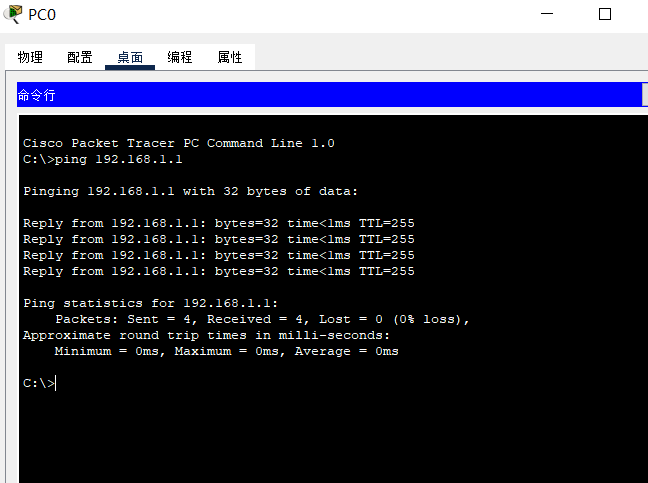


**3．更新并测试**



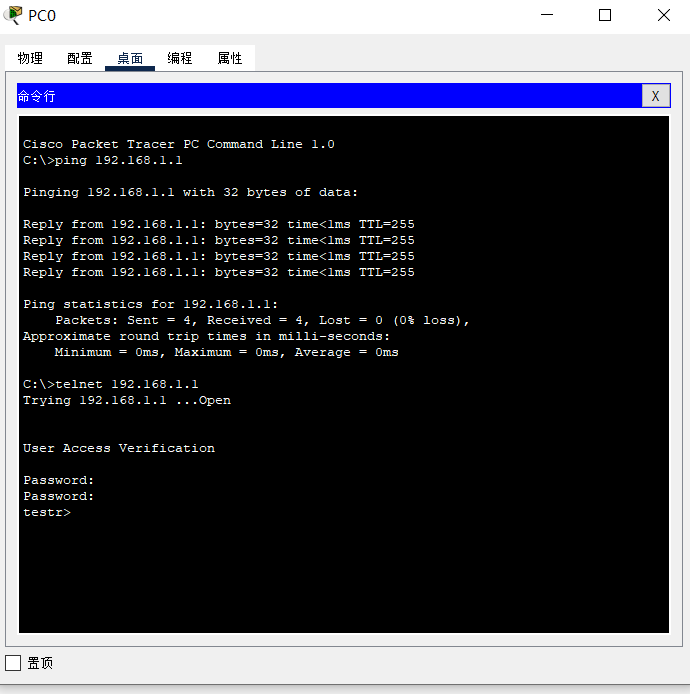
首先设置pc0的ip地址为192.168.1.x（因为路由器配置的ip是192.168.1.1）

Pc0尝试ping路由器，ping通



**4．telnet登录测试**

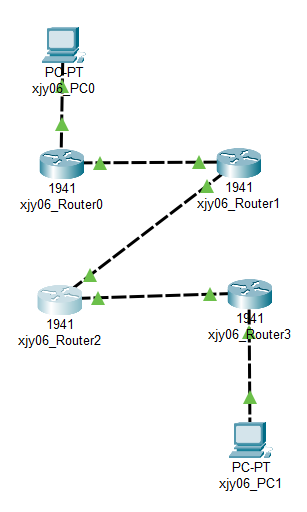
输入之前创建的密码，登录成功



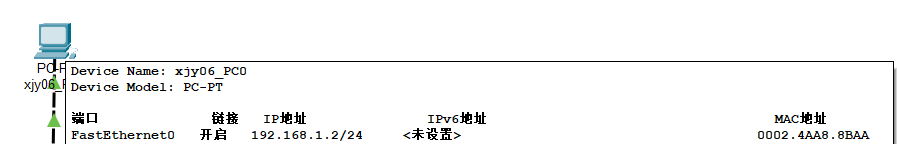
## 2.2.4静态路由配置

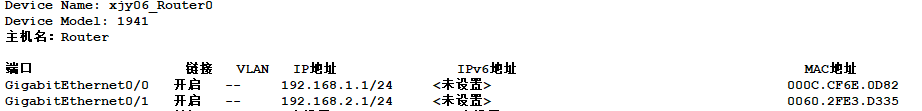
**(1) 线路配置及ip设置**

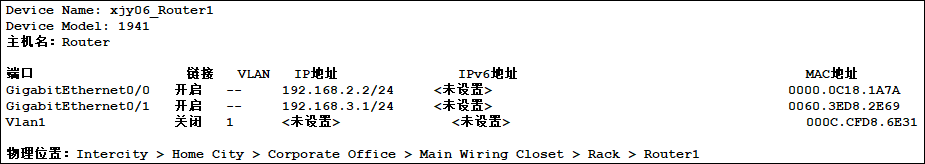
首先是进行线路配置，并把端口打开，如下图所示：

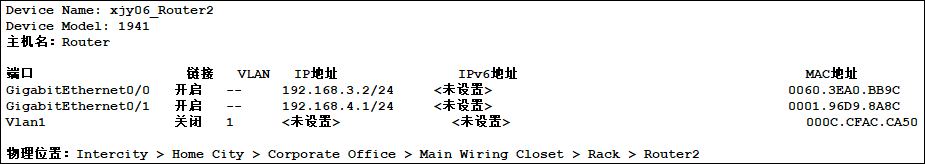


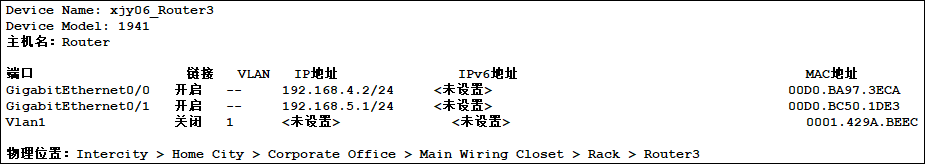
拥有四个路由器，两台主机用于静态路由的配置，其中连接的线为交换线，各主机，端口的ip地址如下图所示：

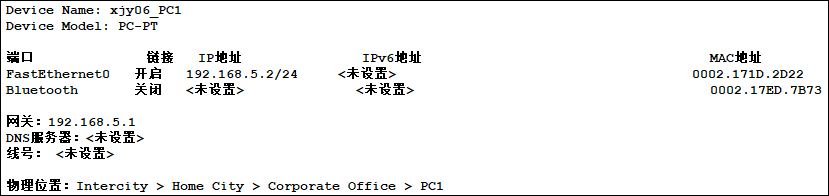






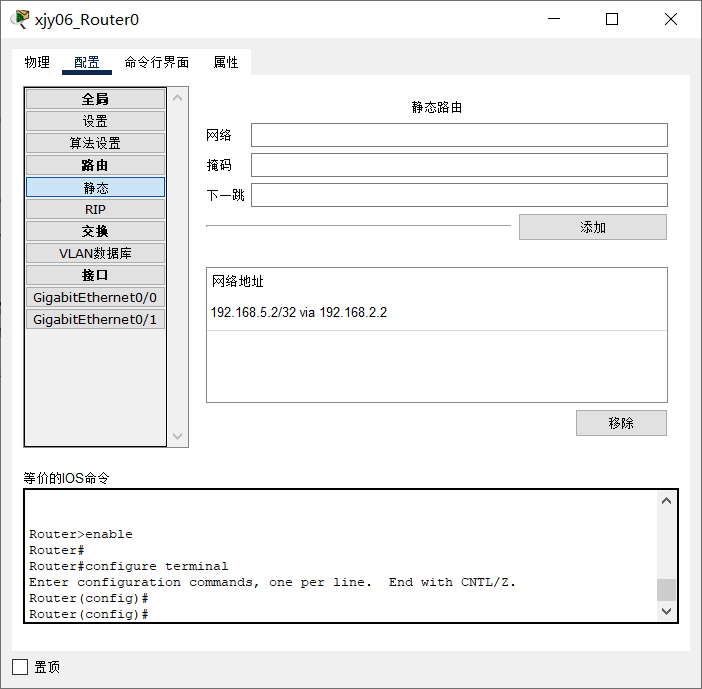


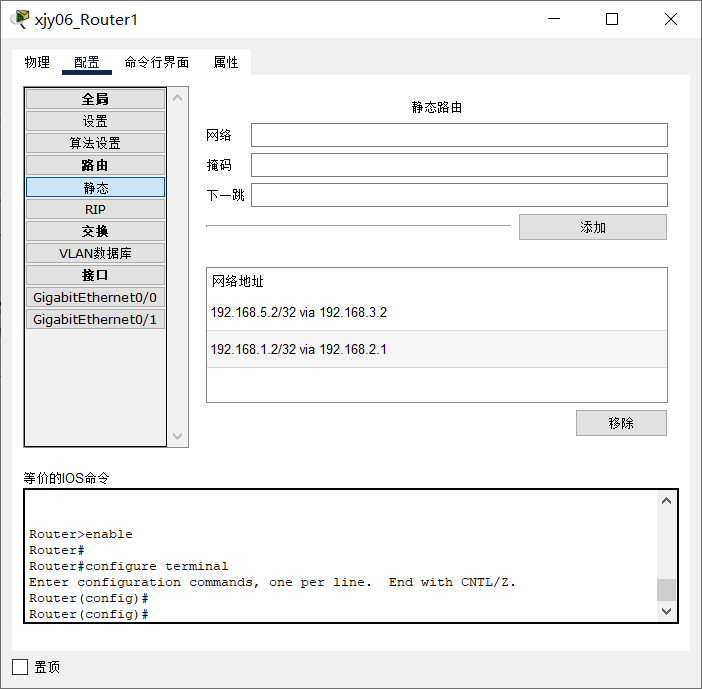


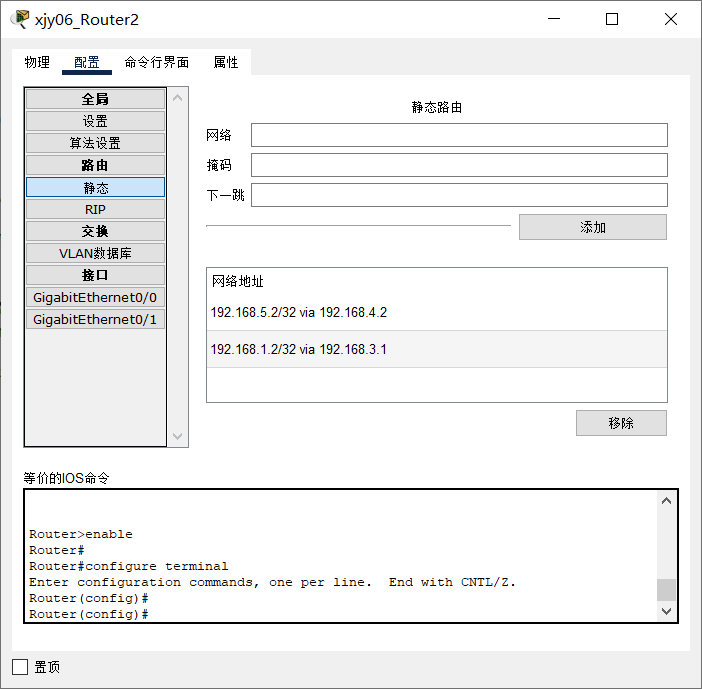


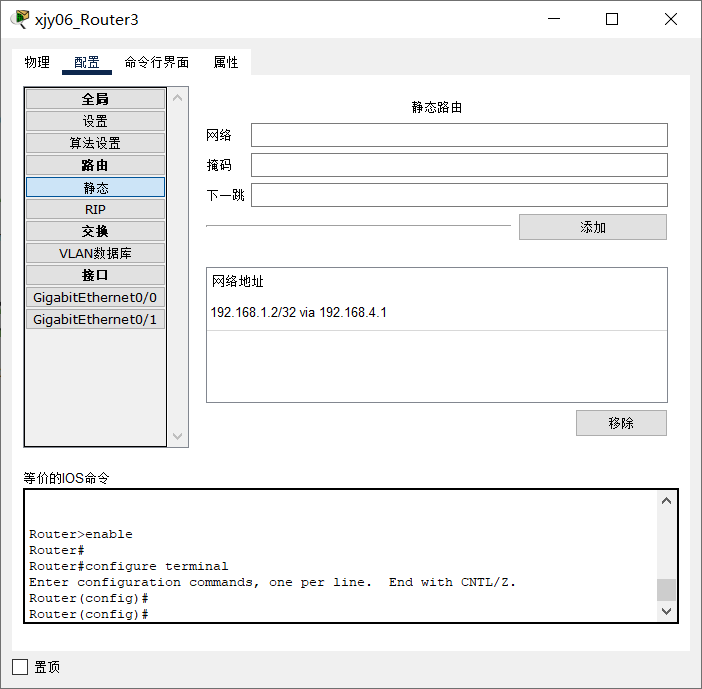
**(2) 静态路由设置**

对于将xjy06\_PC1（192.168.5.2），以及xjy06\_PC0（192.168.1.2）作为目的地址，此刻设置子网掩码为255.255.255.255以代表特定主机路由，并对四个路由的下一跳端口进行依次设计，如下图所示：





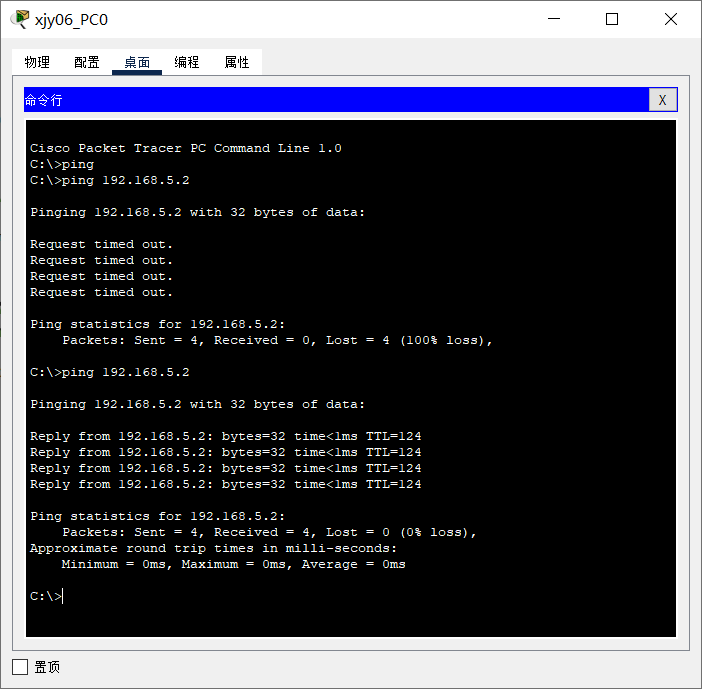




其中，实现指令为ip route 目的网络 掩码 { 网关地址 | 接口 }

**(3) 测试**

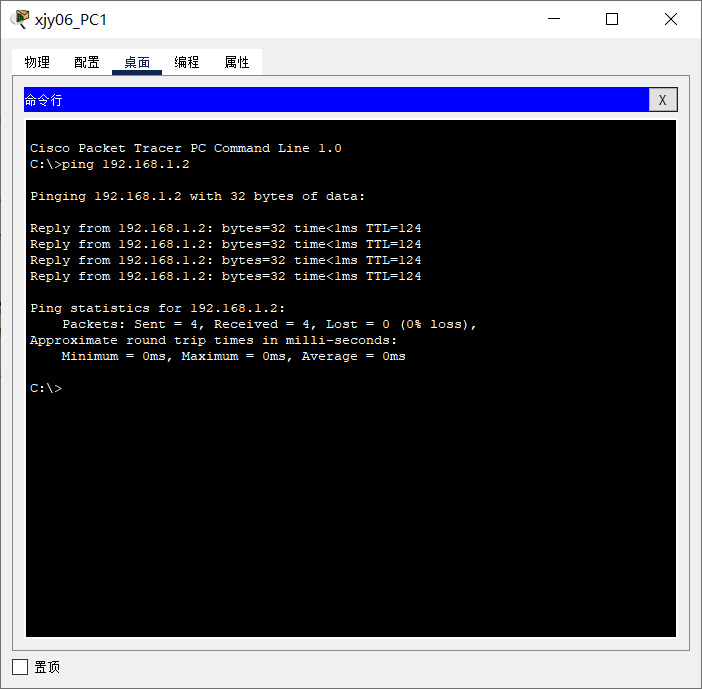
首先是pc0 ping pc1的操作，如下图所示：



第一次ping不通：事先路由不知道物理地址，需要arp，所以丢失

第二次ping通，说明配置完成

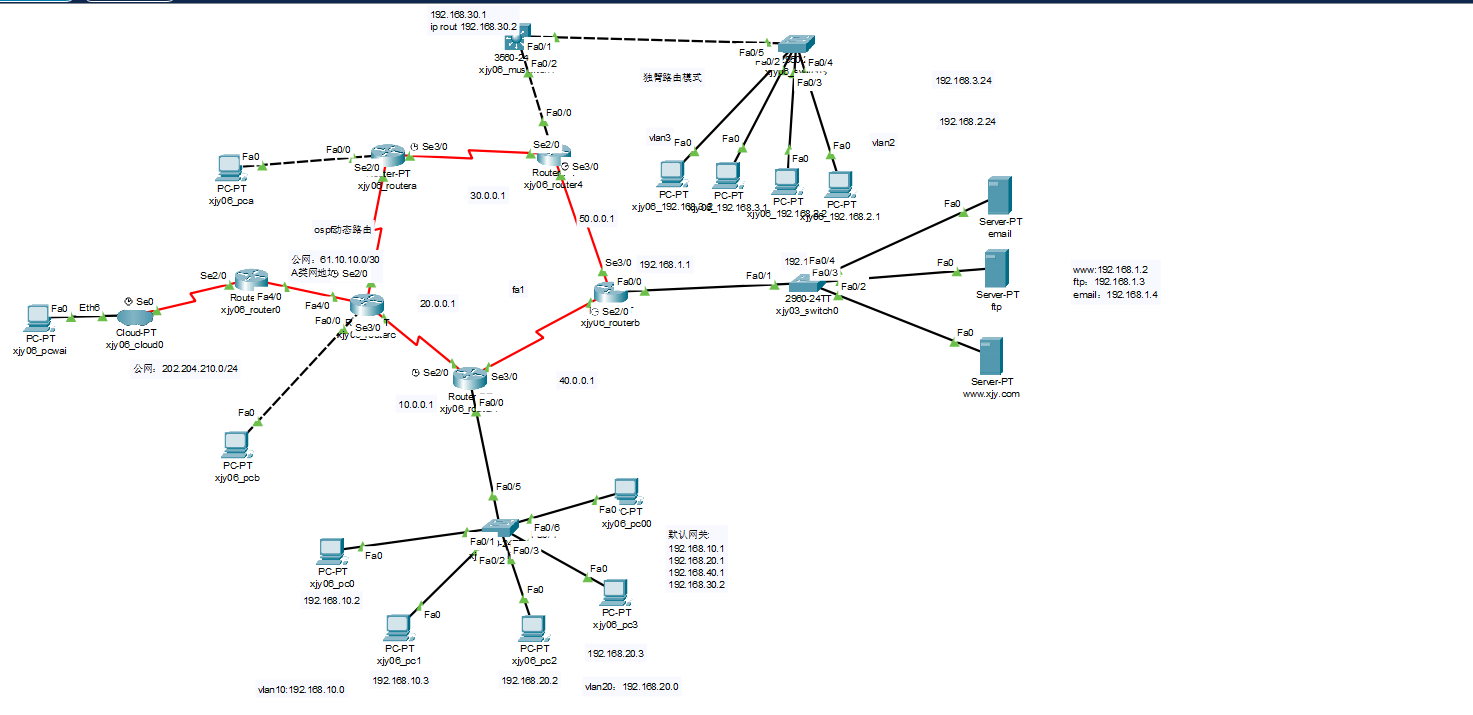
然后是进行反向ping的操作，如下图所示：



显然成功。

## 2.2.5动态路由配置（OSPF）并融入综合网络情景设计

**(1) 网络拓扑图**



**(2) 简单介绍**

利用PC，二层交换机，三层交换机，路由器，服务器，等设备设计小型企业网络，目前主要完成设计的是企业内局域网，xjy06\_pcwai和xjy06\_cloud0为尝试向企业局域网与外网联通的方案，暂未实现。

通过模拟现实场景中企业用户的上网需求。为企业划分经理部，公关部，财务部，开发部，产品部，人事部，运维部等七个部门，用于满足各部门间的通信需求。

同时企业内部设置了，www服务器，ftp服务器，email服务器，满足企业用户文件交流，邮件传输。

**(3) 主要实现的功能**

1、根据部门划分vlan (虚拟局域网)，个别部门划分一个的vlan

2、利用三层交换机通过单臂路由的模式实现不同两个部门间PC的相互通信

3、利用局域网路由器通过单臂路由的模式实现另外两个部门间PC的通信

4. 搭建企业内部web服务器、ftp服务器以及email服务器

5.通过ospf协议实现局域网内各个设备间的正常通信

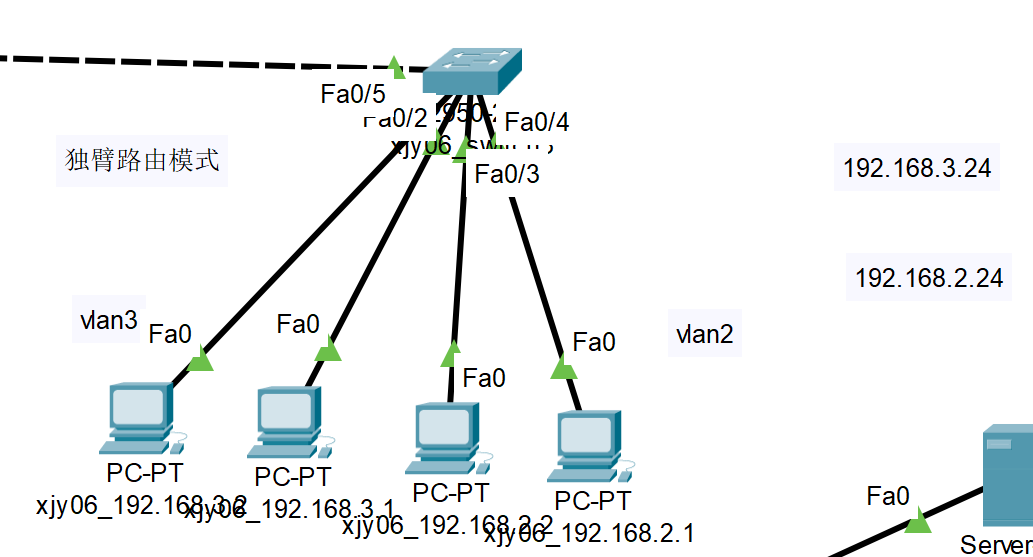
**(4) ip地址划分**

|  |  |  |  |
| --- | --- | --- | --- |
| **名称** | **Ip网段** | **默认网关** | **说明** |
| **Vlan2** | **192.168.2.0** | **192.168.2.24** | **财务部** |
| **Vlan3** | **192.168.3.0** | **192.168.3.24** | **开发部** |
| **Vlan10** | **192.168.10.0** | **192.168.10.1** | **运维部** |
| **Vlan20** | **192.168.20.0** | **192.168.20.1** | **产品部** |
| **三层交换机端口** | **192.168.30.0** | **192.168.30.1** | **重要端口** |
| **企业服务器** | **192.168.1.0** | **192.168.1.1** | **企业内网** |
| **Vlan40** | **192.168.40.0** | **192.168.40.1** | **人事部** |
| **无** | **192.168.80.0** | **192.168.80.2** | **经理部** |
| **无** | **192.168.70.0** | **192.168.70.2** | **公关部** |

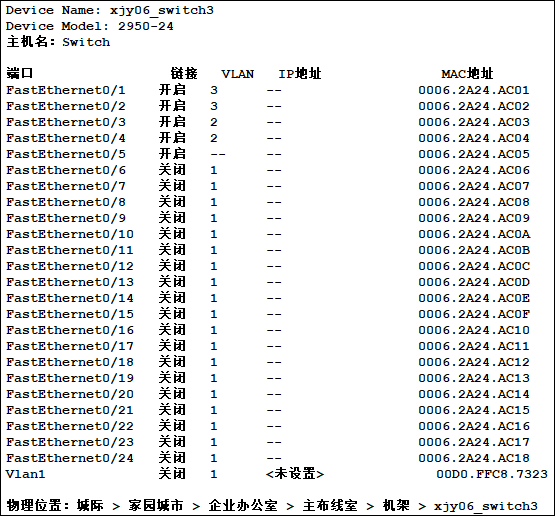
**(5) 单臂路由的实现**

1.以三层交换机所管vlan2，vlan3为例

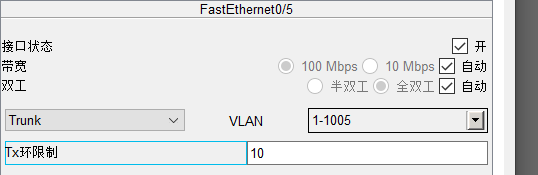
首先对交换机3



对端口划分各自vlan



并通过switchport mode trunk 将fa 0/5调整为vlan全通端口



同理对三层交换机的fa 0/1端口做此调整

下面在三层交换机配置逻辑接口

以vlan2为例子

En

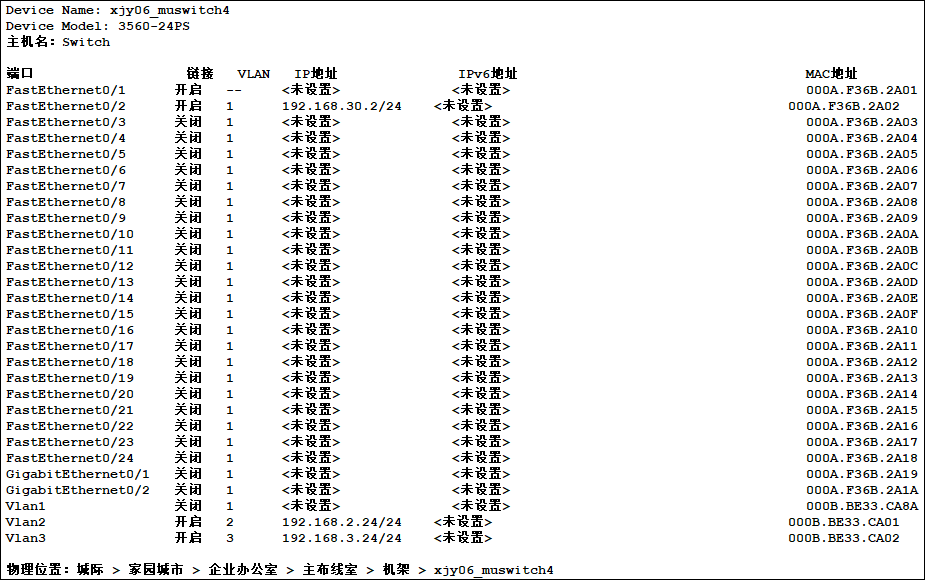
conf t

int vlan 2

ip address 192.168.2.24 255.255.255.0

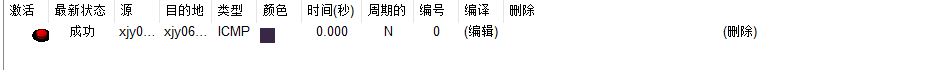
no shut

结果如下



下面我们对此局域网进行测试，截图如下所示，

由vlan3的192.168.3.1 发送 pdu 去vlan2 的192.168.2.1



可见成功

同理可配置路由器主导的单臂路由，vlan10，20，40区域

**(6) 通过ospf实现企业网内互通**

首先是指令，

以三层交换机为例

en

conf t

router ospf 2

network 192.168.2.0 0.0.0.255 area 0

创建ospf2的进程，并把192.168.2.0网段加入ospf行列中，实现即把所有的相邻网段加入ospf中

并按类似操作对路由器进行操作，所得结果截图如下所示：

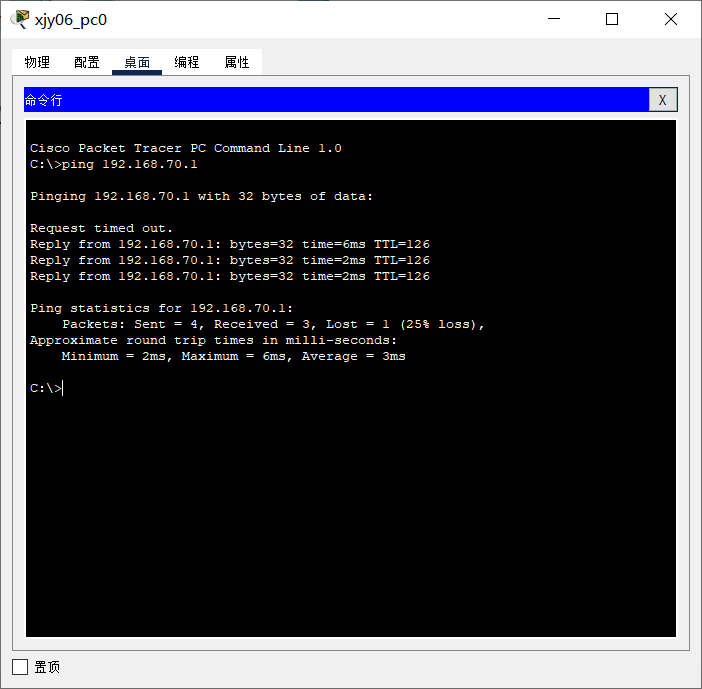


其中c代表直连，o代表需要跳转，via前为目的地，后为需要通关的端口，[ ]里的110是管理距离，后面的是cost，即路由穿越的那些中间网络的开销的累加，其默认公式为开销=10的8次方除以带宽，这是ospf选择路线重要的参考依据。、

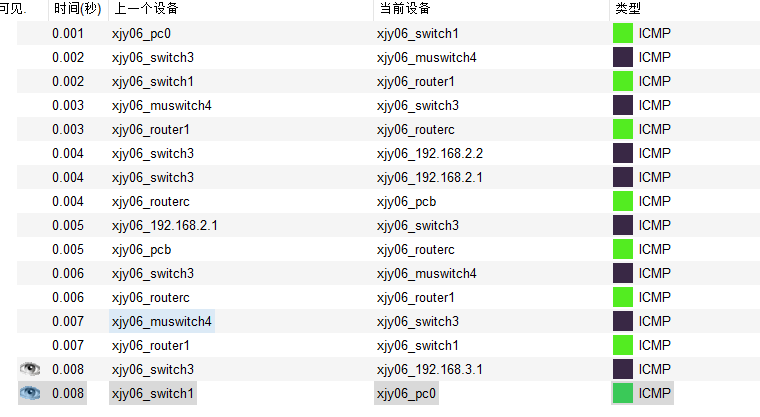
其它路由类似操作，后达成路由自学习，后即实现局域网互通。

**(7) 测试**

**P**c0 ping pcb



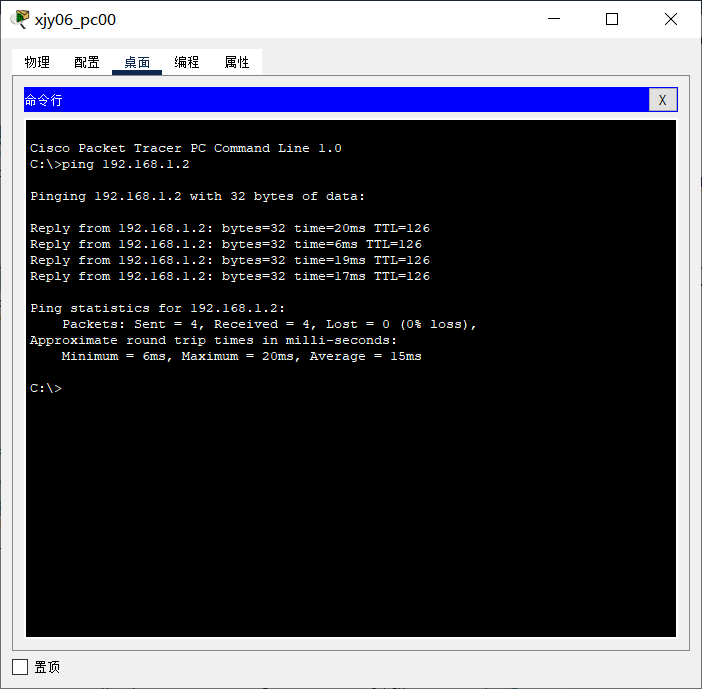
可见已经ping通，下面我们查看发送数据报时走的路线



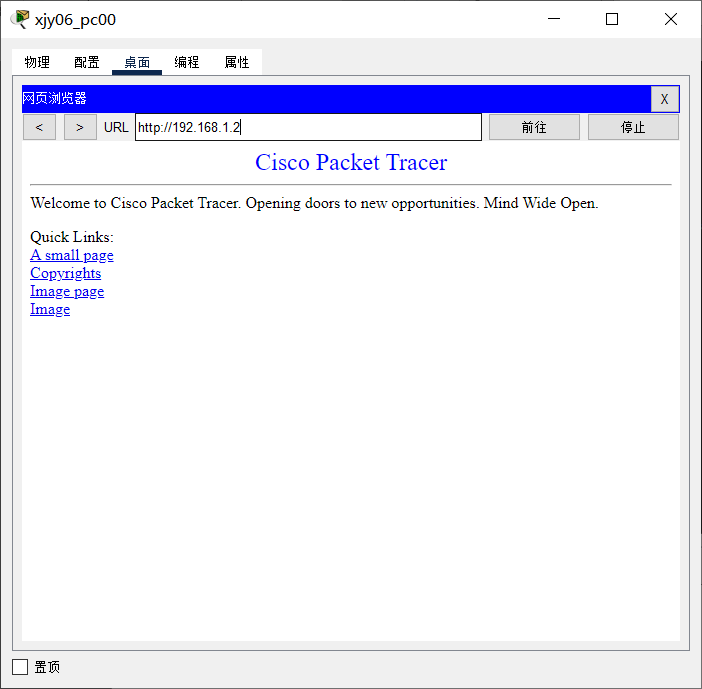
可以看到选择的路线是router1和routerc，显然是cost较小的路线，

可见配置成功

下面测试pc00 ping 服务器，



下面测试pc00访问网页



可见成功

# 2.3 网络编程

## 2.3.1 设计思想

利用java实现，安装winpcap和配置Jpcap.jar库文件，从管理员权限运行，利用jpcap的getdevicelist查看网卡列表，并通过opendevice函数绑定网卡，利用getpacket函数抓包，同时为了输出ip对ippacket进行4的锁定，接着对ippacket类型的各种属性进行输出，便能得到版本号、总长度、标志位、片偏移、协议、源地址和目的地址等内容。

## 2.3.2 实验环境

通讯程序编程语言：Java 编程软件: idea

抓包工具：wincap

硬件：

硬件环境： 设备名称 DESKTOP-778RLJS

处理器 11th Gen Intel(R) Core(TM) i7-11800H @ 2.30GHz 2.30 GHz

机带 RAM 16.0 GB (15.7 GB 可用)

系统类型 64 位操作系统, 基于 x64 的处理器

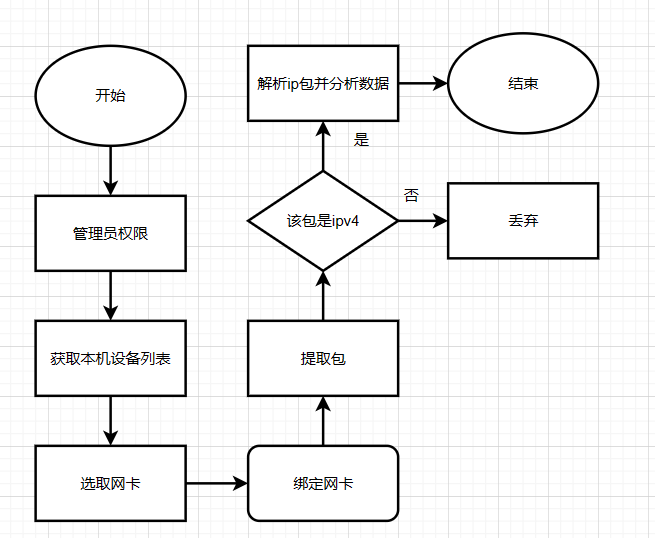
## 2.3.3 主要函数

用JpcapCaptor.getDeviceList()获取网卡列表，返回一个jpcap.NetworkInterface的数组

对网卡调用JpcapCaptor.openDevice()，返回一个JpcapCaptor类的对象

创建Packet类的对象用来接收抓到的包，利用jpcap.getPacket()函数

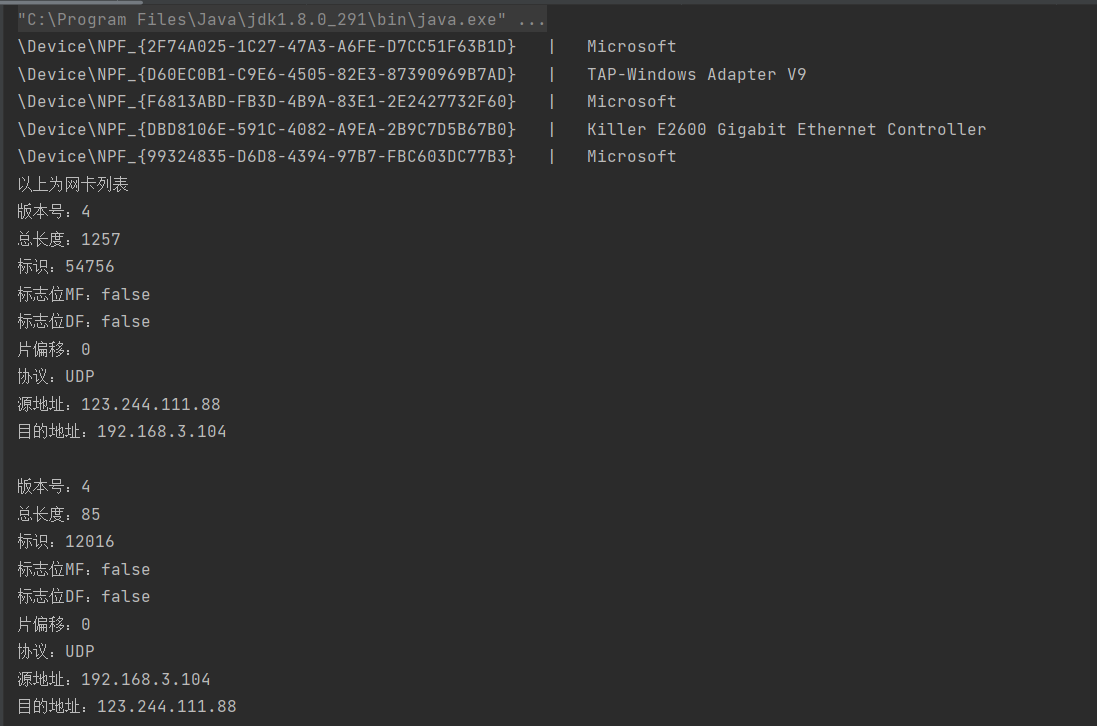
## 2.3.4程序流程图



## 2.3.5 难点解决

关于难点，首先就是如何绑定到本机网卡上，进行了很多次程序都报错，后来发现要管理员权限，然后就是在学习jpcap相关函数的时候，网络上的内容多且冗杂，后来经过不断努力，终于理解了函数的用法，以至后面的操作都水到渠成了。

## 2.3.6 测试结果截图



## 2.3.7 实验结果分析

首先是ip数据报的基本信息：

版本号：占4位，IP协议版本号, IPv4此字段值为4, IPv6此字段值为6

总长度：占16位，以字节为单位, 总长度包含IP的头部和数据部分, IP数据报

最大长度为65535字节, 但是最大不要超过MTU的长度

标识：占16位，唯一标识，一个数据报,可每发送一个数据报, 则该值加1, 如果数据报分片,则每个分片的标识都一样, 各个分片共享一。个标识号

标志位：占3位，最高位为0；最低位MF=1（0），后面还有分片（最后一个报

片）；DF=1（0），不能分片（可以分片）

片偏移：占13位，用以指出该分段的第一个数据字节在原始数据报中的偏移位

置(以8字节为单位),IP分片后每一个分组都具有自己的首部, 而且标志位相同,

但是片偏移值不同, 通过片偏移值接收端可以重新组装IP包

协议类型：占8位，指明IP层上承载的是哪个高级协议, 在分用的过程中, 协议

栈知道该交给上层的哪个协议处理, 如1为ICMP, 2为IGMP, 6为TCP, 17为UDP等.。

源地址：32位，发送方IP地址

目的地址：32位，接受方IP地址

依次对照实验结果，发现实验内的代码已经直接将内容输出出来了，例如协议类型的单独判断我们就不需要，然后就是对照基本信息查看是否出现差错，发现没有差错

## 2.3.8 程序源代码

import jpcap.JpcapCaptor;

import jpcap.NetworkInterface;

import jpcap.packet.IPPacket;

import jpcap.packet.Packet;

import java.io.IOException;

public class JpcapPacket{

public static void main(String[] args) {

//绑定网络设备

//获取网卡列表

NetworkInterface[] kk = JpcapCaptor.getDeviceList(); //一组NetworkInterface对象，可看作网卡数组

for (NetworkInterface n : kk) {

System.out.println(n.name + " | " + n.description);

}

System.out.println("以上为网卡列表");

JpcapCaptor jpcap = null;

// 限定抓取数据包的多少个字节

int temp = 1512;

try {

jpcap = JpcapCaptor.openDevice(kk[0], temp, true, 50); //获取网卡连接及获取包限时时间

} catch (IOException e) {

e.printStackTrace();

}

//抓包

int i = 0;

while (i < 5) {

Packet packet = jpcap.getPacket();

if (packet instanceof IPPacket && ((IPPacket) packet).version == 4) { //抓的是ipv4

i++;

IPPacket ip = (IPPacket) packet;

System.out.println("版本号："+ip.version);

System.out.println("总长度：" + ip.length);

System.out.println("标识：" + ip.ident);

System.out.println("标志位MF：" + ip.more\_frag);

System.out.println("标志位DF：" + ip.dont\_frag);

System.out.println("片偏移：" + ip.offset);

String protocol = ""; //用于确认协议

switch (new Integer(ip.protocol)) {

case 1:

protocol = "ICMP";

break;

case 2:

protocol = "IGMP";

break;

case 6:

protocol = "TCP";

break;

case 8:

protocol = "EGP";

break;

case 9:

protocol = "IGP";

break;

case 17:

protocol = "UDP";

break;

case 41:

protocol = "IPv6";

break;

case 89:

protocol = "OSPF";

break;

default:

protocol = "Other";

break;

}

System.out.println("协议：" + protocol);

System.out.println("源地址：" + ip.src\_ip.getHostAddress());

System.out.println("目的地址：" + ip.dst\_ip.getHostAddress());

System.out.println("");

}

}

}

}

1. **实验总结与心得体会**

本次实验让我更进一步了解和掌握了计算机网络的建立和拓扑结构，熟悉了各种网络指令，并对packet tracer的指令学习更加深入，出于学习不够深入的原因，并没有成功地把小型企业网配出来，虽然有想过把它配出来

实验全部做下来还是很需要时间的，真正完成之后，收获很多实用的知识