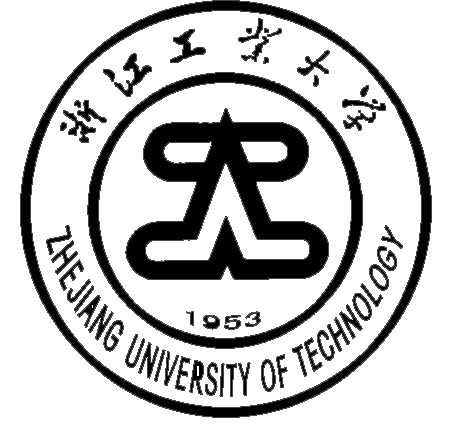


**计算机网络课程设计报告**



学 号

姓 名

班 级

**目录**

**一．课程的目的和任务------------------------------------------------------3**

**二． 课程的具体要求和内容-------------------------------------------------3**

**2.1具体要求----------------------------------------------------------------3**

**2.2实验内容----------------------------------------------------------------4**

**1. 任务一 常用网络命令 -------------------------------------------------4**

**2. 任务二 交换机与路由器 --------------------------------------------13**

**3. 任务三 网络编程 ----------------------------------------------------28**

**三． 总结------------------------------------------------------------------35**

**一．课程的目的和任务**

本课程要求学生在“计算机网络原理”理论学习基础上，验证和加深对计算机网络概念的理解。通过本课程的实践，培养学生独立思考、综合分析和动手实践的能力。

通过本课程的学习，达到如下目标：

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，能够对命令的功能进行解释，分析命令执行结果，得到合理有效结论。

2. 掌握Packet Tracer软件的操作方法，能够使用该软件模拟组网、配置交换机、路由器，能够按照实验方案实施仿真实验，采集和整理数据。

3. 具备网络编程能力，能够设计抓包程序获取数据包，结合相关协议对数据包各个字段的含义进行分析、处理和解释，获取有效结论。

1. **课程的具体要求和内容**

**2.1具体要求**

**任务一：常用网络命令**

1. 写出常用网络命令 ipconfig, ping, netstat, tracert, arp, telnet 的功能；
2. 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：交换机与路由器**

(1) 安装 packet tracer（自行网络搜索），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。

要求：静态路由配置拓扑中至少 4 个路由器；RIP 动态路由配置中源站和目的站之间设置两条跳数 不同的路径，通过 RIP 配置后查看选择的是哪条路径。

要求写出相应的步骤，给出截图和文字说明。

**任务三：网络编程**

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

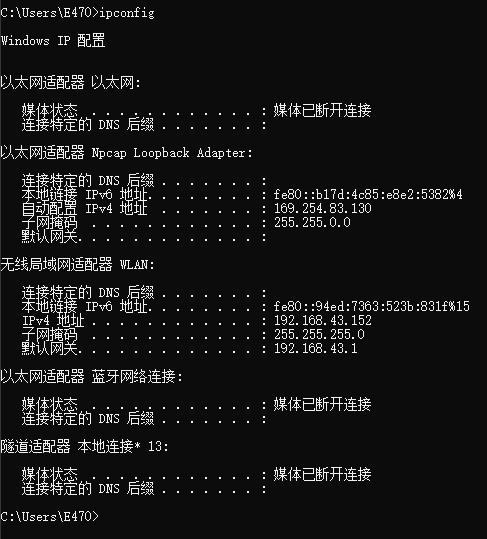
的关联、总结和心得体会等。

编程语言不作要求，可使用自己熟悉的 C、C++、java 或 C#等

**2.2实验内容**

**任务一：常用网络命令**

1. ipconfig: 显示Windows IP 配置.



-图1-

分析:

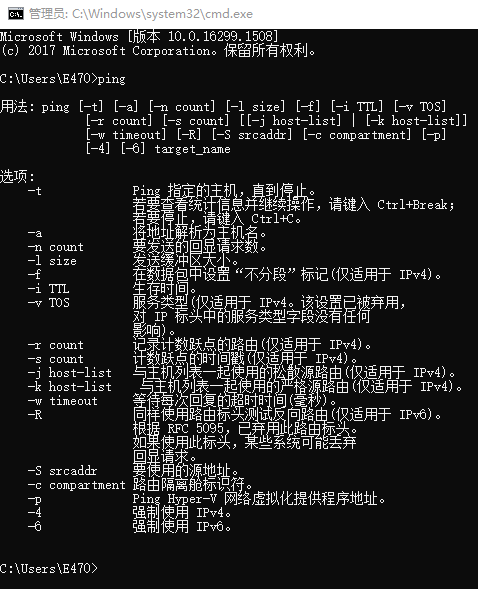
(1)使用不带参数的ipconfig命令可以得到以下信息：IP地址、子网掩码、默认网关.

(2)使用ipconfig /all可得到更多信息：主机名、DNS服务器、节点类型、网络适配 器的物理地址、主机的IP地址、子网掩码以及默认网关等,可以知道本地网络是否配置正常.

2.ping: 用来检查网络是否通畅或者网络连接速度.

ping是Windows、Unix和Linux系统下的一个命令,也属于一个通信协议，是TCP/IP协议的一部分.利用“ping”命令可以检查网络是否连通，可以很好地帮助我们分析和判定网络故障.ping命令主要基于ICMP.ping回答了两个基本的问题：是否有连接？连接的质量如何？

(1)ping



-图2-

分析:使用ping命令，可以看到ping命令的许多可使用的参数.

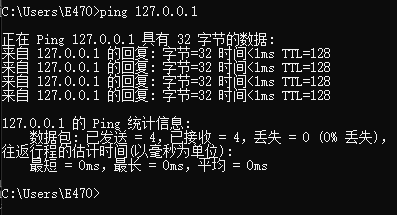
(2)ping+远程IP



-图3-

分析：ping远程IP地址，检查网络连接是否正常.如图也可换成IP地址.119.75.217.109是百度的一个IP地址，在连接网络与断开网络的情况下分别ping这个IP地址，得到两种不同的结果，在网络连接时成功ping通，而在网络断开时无法访问目标主机.

(3)ping+127.0.0.1



-图4-

分析：

ping环回地址，验证在本地计算机上是否正确地安装了TCP/IP协议.

发送的数据包从cmd的shell发到了kernel的tcp/ip模块，然后从tcp/ip模块返回到cmd下，就看到ping成功了.也表示本机tcp/ip配置没问题.

(4)ping+本机IP



-图5-

分析：检查本机网卡是否正常

ping127.0.0.1和ping本机的过程是不一样的，ip输出函数先检查地址是不是环回地址

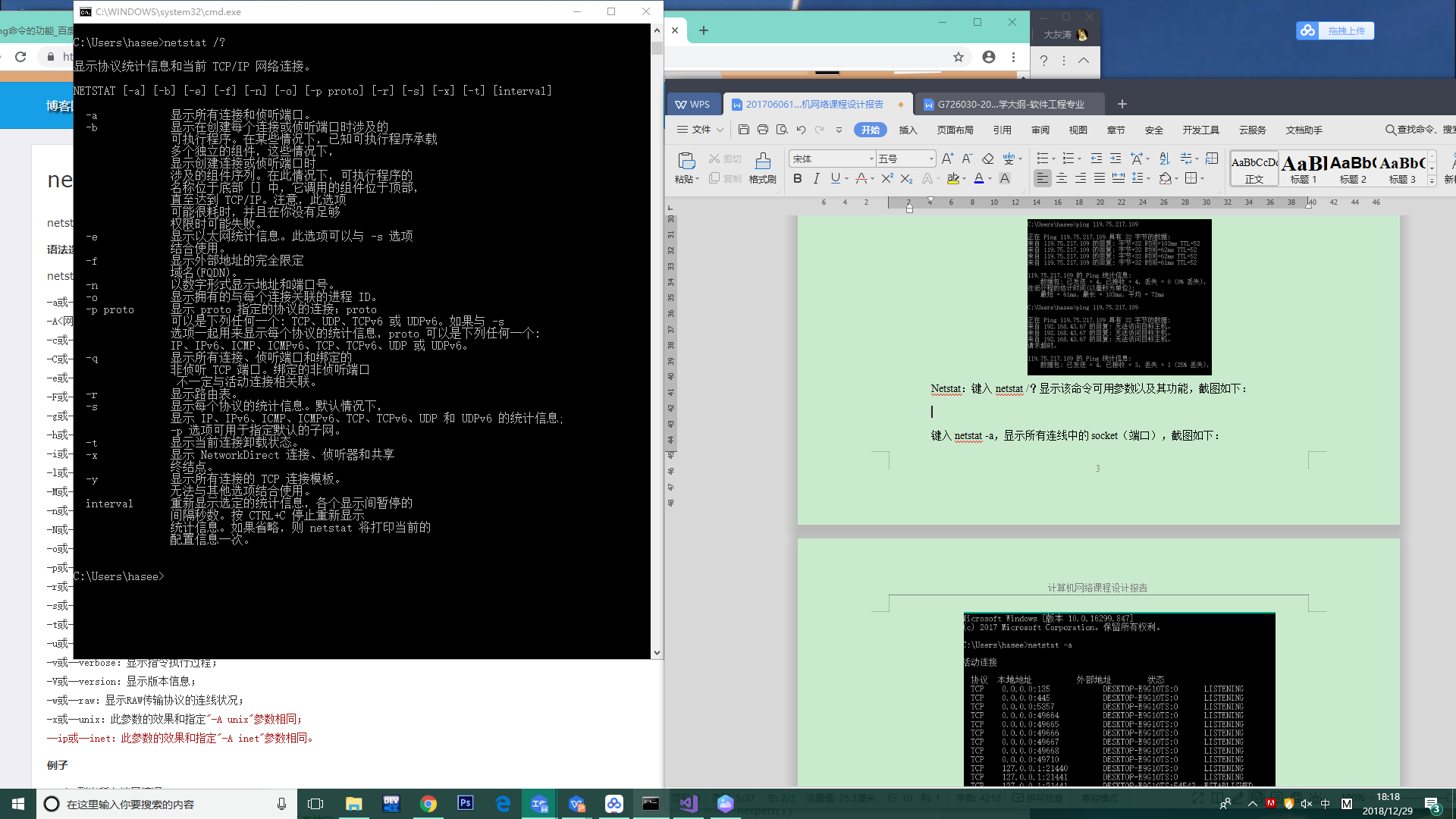
1. 如果是环回地址,直接交给环回驱动程序处理,返回ip输入函数
2. 如果不是环回地址,检查是不是广播或者多播地址
3. 如果不是广播或者多播地址,才检查是不是本机地址,如果是本机地址 则交给环回驱动程序处理，环回驱动程序返回给ip输入函数

从上面可以看出 ping127.0.0.1，数据包是不经过网卡的 ，ping本机则是需要经过网卡的.

3.netstat: netstat是在内核中访问网络及相关信息的程序，它能提供TCP连接，TCP和UDP监听，进程内存管理的相关报告.

netstat命令的功能是显示网络连接、路由表和网络接口信息，可以让用户得知有哪些网络连接正在运作.使用时如果不带参数，netstat显示活动的TCP连接状态列表.

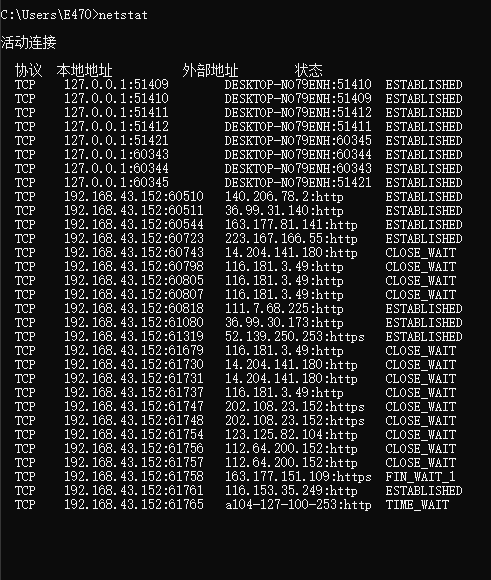
1. netstat /?



-图6-

分析：这是netstat命令的所有可用参数.

(2)netstat



-图7-

分析：显示所有连连接和侦听窗口.

含义解释：

LISTEN - 侦听来自远方TCP端口的连接请求；

SYN-SENT - 在发送连接请求后等待匹配的连接请求；

SYN-RECEIVED - 在收到和发送一个连接请求后等待对连接请求的确认； ESTABLISHED - 代表一个打开的连接，数据可以传送给用户；

FIN-WAIT-1 - 等待远程TCP的连接中断请求，或先前的连接中断请求的确认；

FIN-WAIT-2 - 从远程TCP等待连接中断请求； CLOSE-WAIT - 等待从本地用户发来的连接中断请求； CLOSING - 等待远程TCP对连接中断的确认；

LAST-ACK -等待原来发向远程TCP的连接中断请求的确认； TIME-WAIT - 等待足够的时间以确保远程TCP接收到连接中断请求的确认； CLOSED - 没有任何连接状态.

该命令的一般格式为: netstat [-a][-e][-n][-o][-p Protocol][-r][-s][Interval]

命令中各选项的含义如下：

-a 显示所有socket，包括正在监听的

-c 每隔1秒就重新显示一遍，直到用户中断它

-i 显示所有网络接口的信息，格式“netstat -i”

-n 以网络IP地址代替名称，显示出网络连接情形

-r显示核心路由表，格式同“route -e”

-t 显示TCP协议的连接情况

-u 显示UDP协议的连接情况

-v 显示正在进行的工作

-p 显示建立相关连接的程序名和PID

-b 显示在创建每个连接或侦听端口时涉及的可执行程序

-e 显示以太网统计。此选项可以与 -s 选项结合使用

-f 显示外部地址的完全限定域名(FQDN)

-o显示与每个连接相关的所属进程 ID

-s 显示每个协议的统计

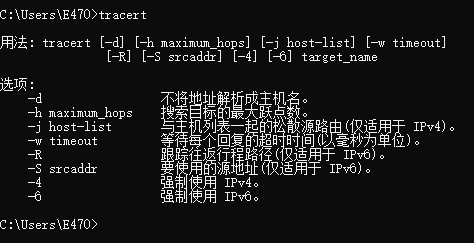
-x 显示 NetworkDirect 连接、侦听器和共享端点。

-y 显示所有连接的 TCP 连接模板。无法与其他选项结合使用。

interval 重新显示选定的统计，各个显示间暂停的间隔秒数.按 CTRL+C 停止重新显示统计.如果省略，则 netstat将打印当前的配置信息一次.

1. tracert: tracert（跟踪路由）是路由跟踪实用程序，用于确定 IP 数据包访问目标所采取的路径.tracert 命令用 IP 生存时间 (TTL) 字段和 ICMP 错误消息来确定从一个主机到网络上其他主机的路由.

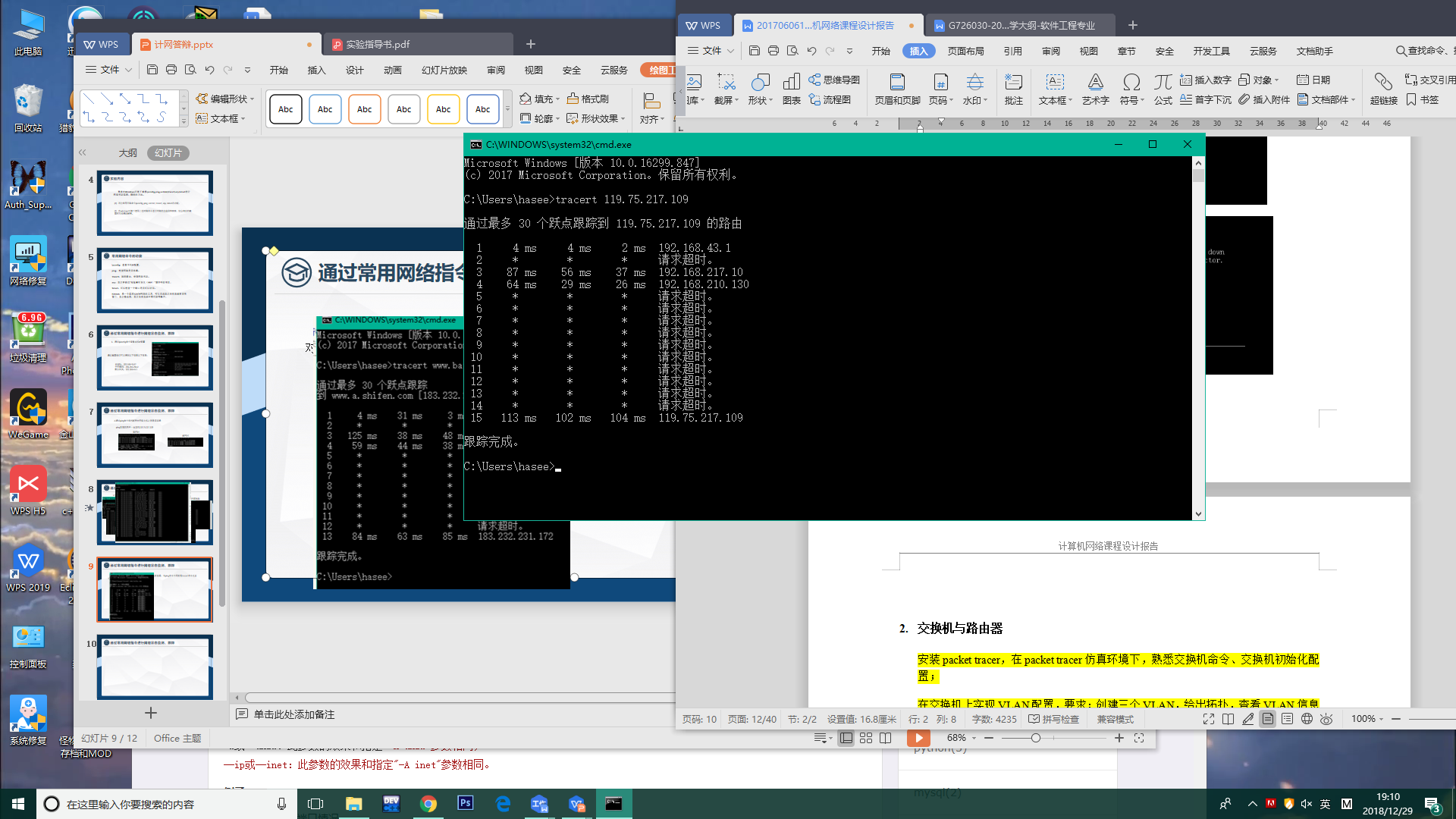
(1)tracert



-图8-

分析：这是tracert的所有可用参数.

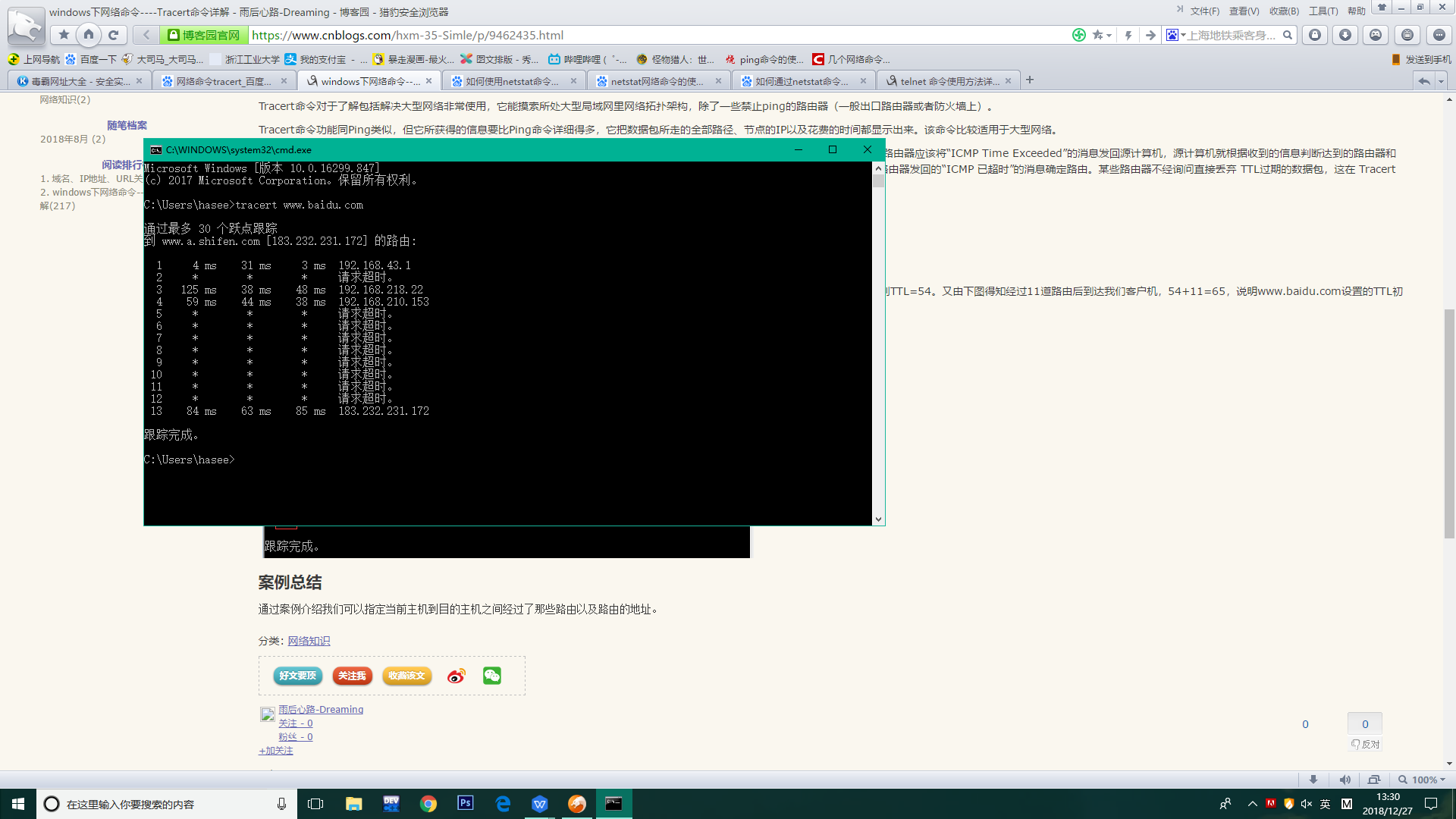
(2)tracert+空格+IP地址



-图9-

分析：获取网络状态，跟踪到该IP地址的路由选择路径

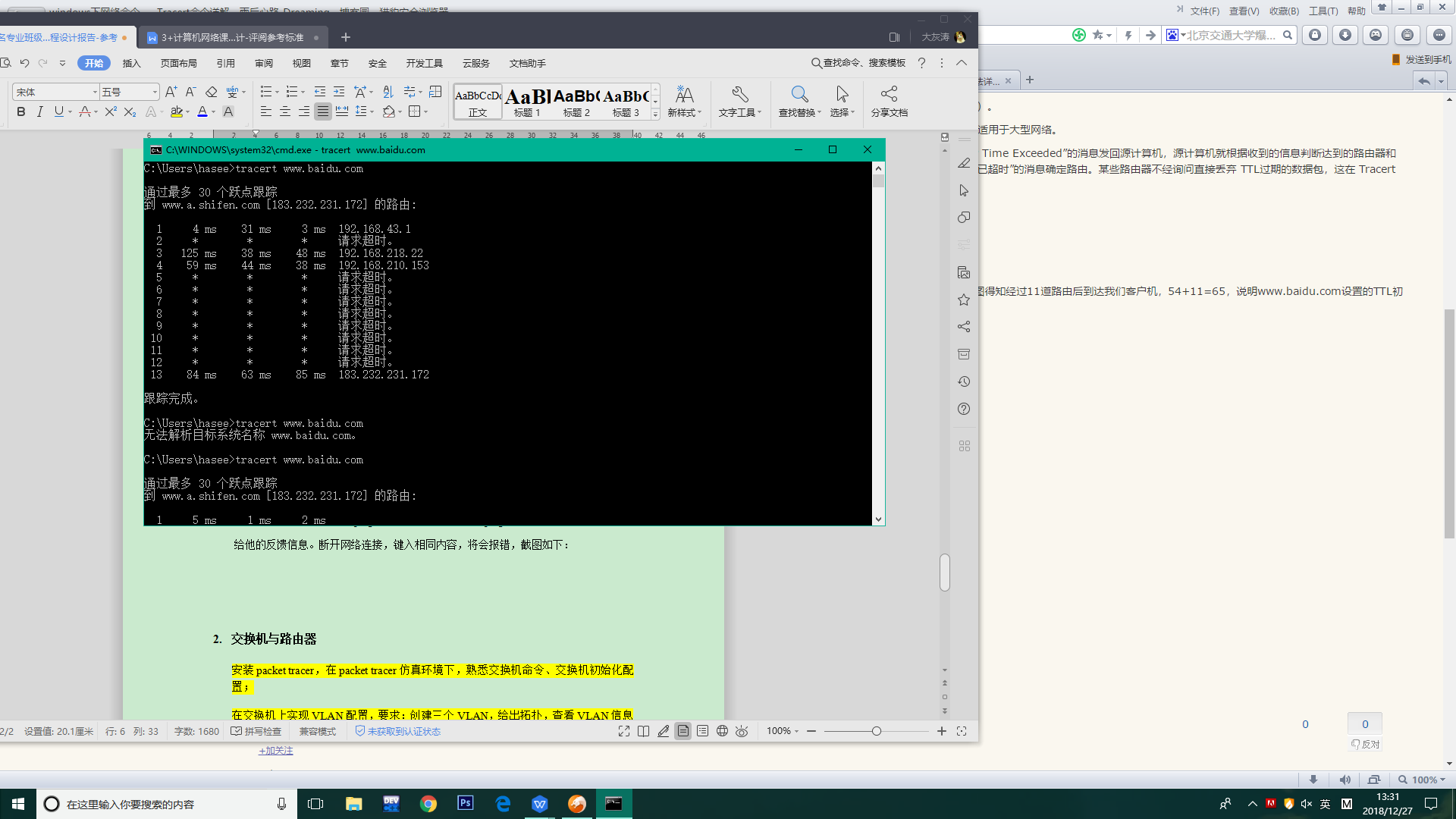
(3)tracert +一个网址



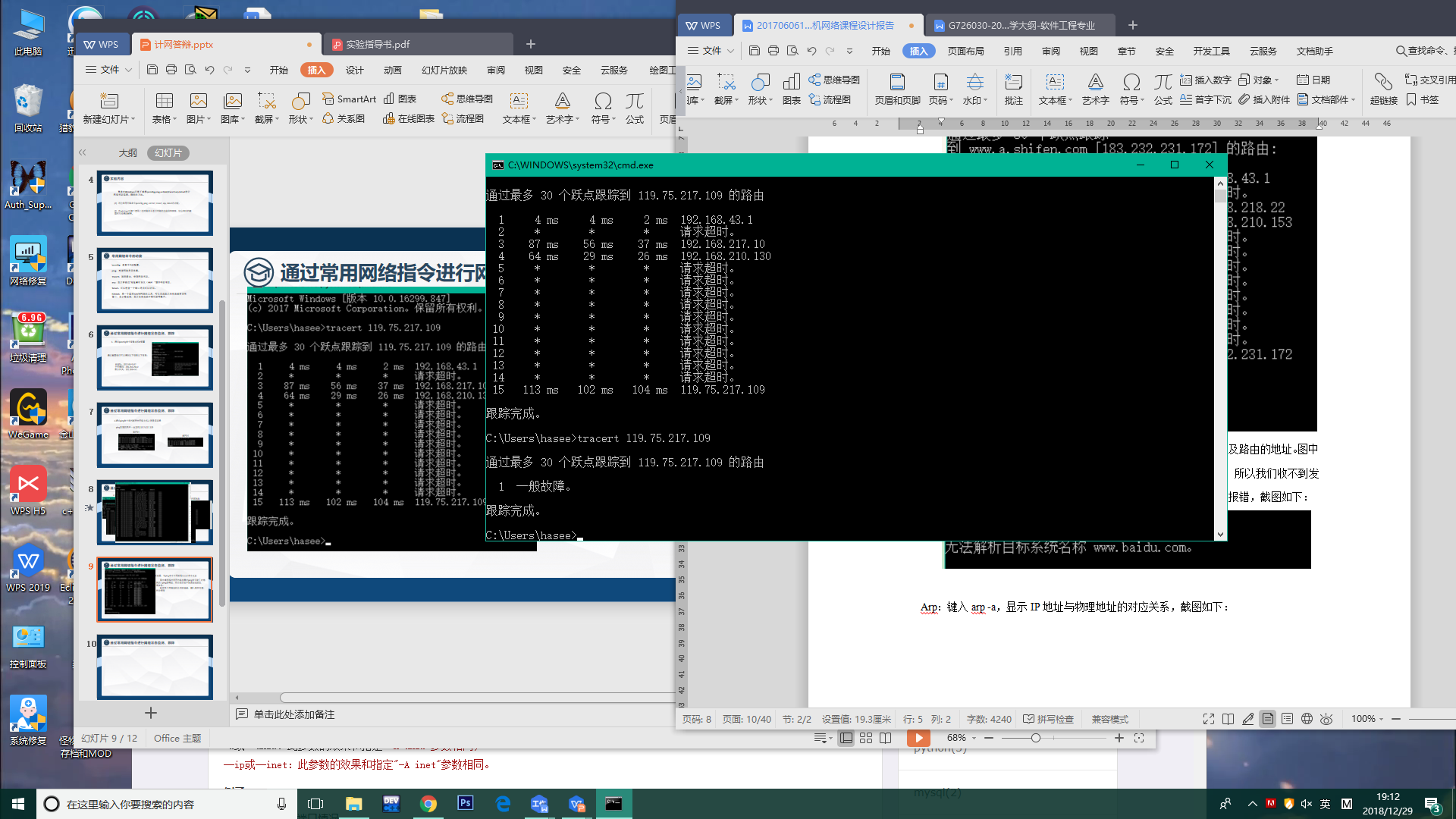
-图10-

分析：DNS解析会自动将其转换为IP地址并探查出途经的路由器信息。如这回在后面输入了百度的URL地址，可以发现共查询到13条信息.

由此我们可以知道当前主机到目的主机之间经过了那些路由以及路由的地址。图中请求超时是因为路由器对ping命令做了处理，关闭了ping的响应，所以我们收不到发给他的反馈信息。断开两个主机之间的连接，键入相同内容，将会报错，如图：



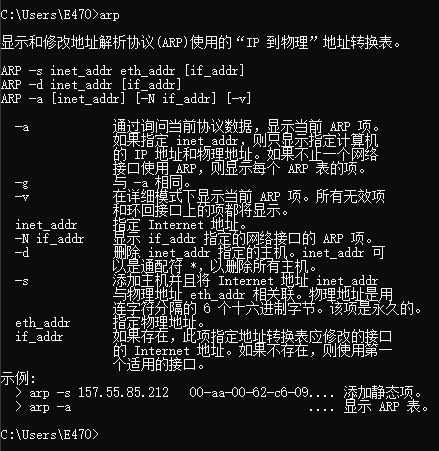
-图11-

-

-图12-

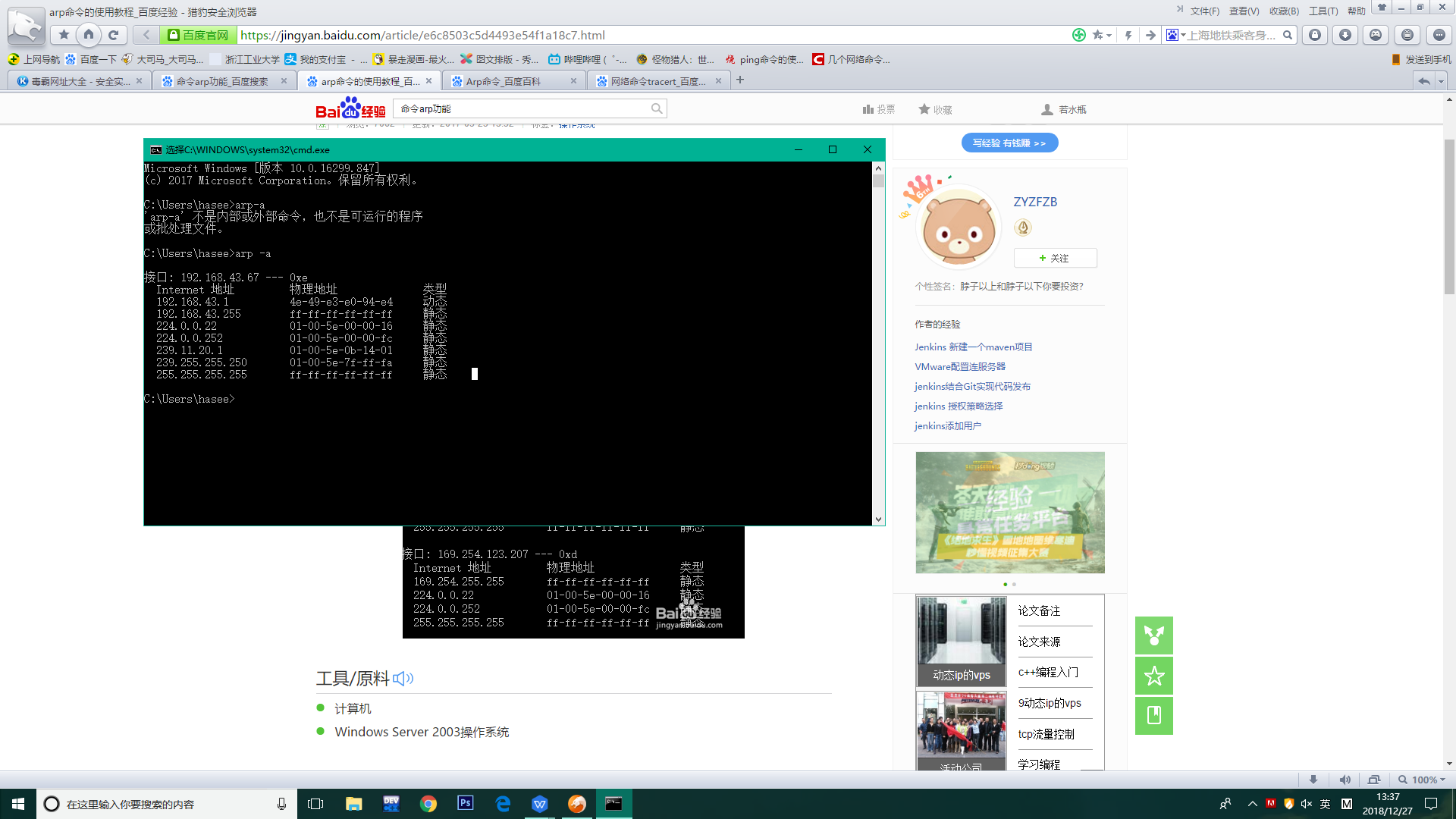
1. arp: 用来显示及修改特定IP地址的网卡地址.根据IP地址获取物理地址的一个TCP/IP协议.主机发送信息时将包含目标IP地址的ARP请求广播到网络上的所有主机，并接收返回（响应）消息，以此确定目标的物理地址；收到返回消息后将该IP地址和物理地址存入本机ARP缓存中并保留一定时间，下次请求时直接查询ARP缓存以节约资源.

(1)arp



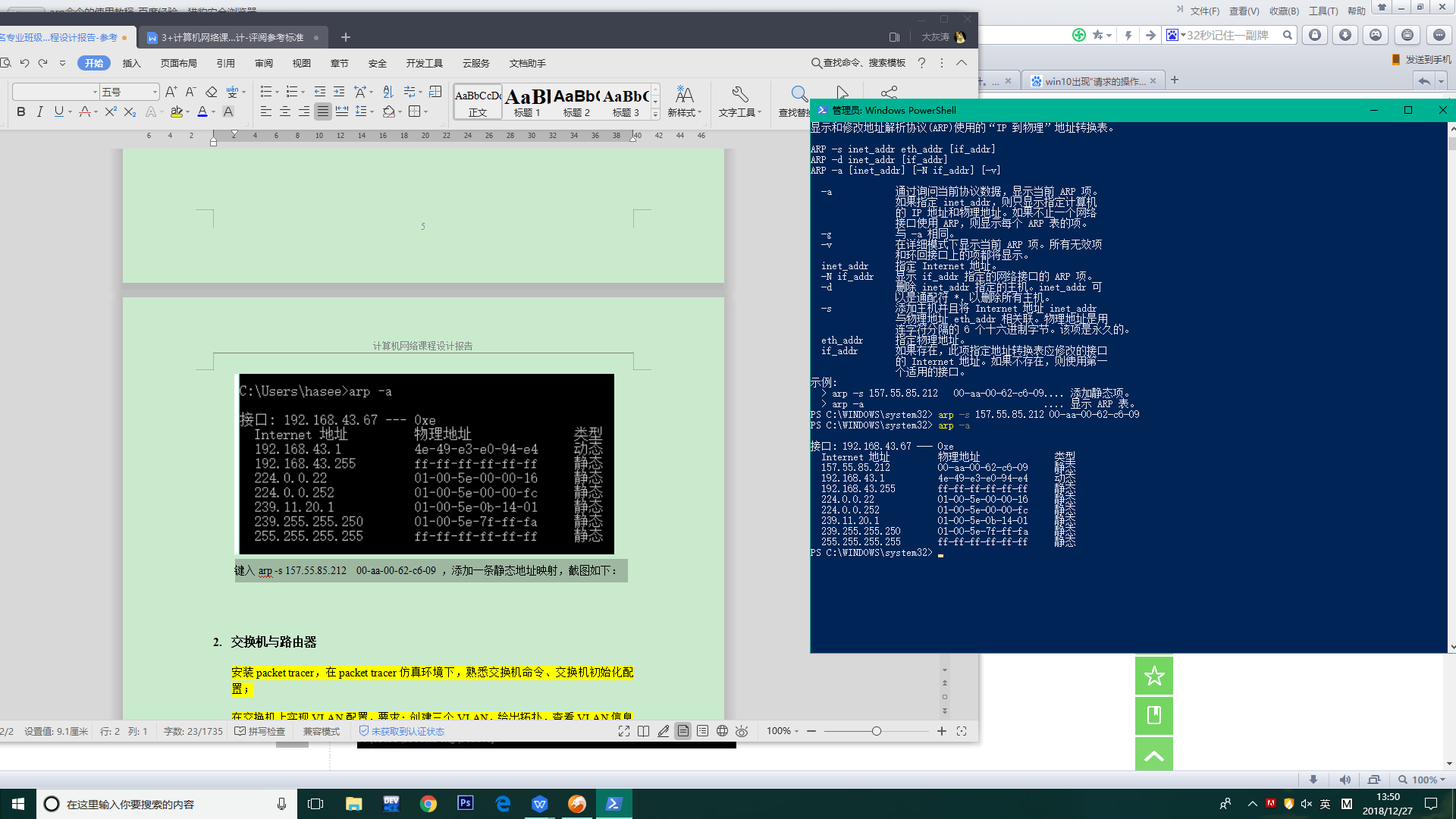
-图13-

1. 键入arp -a，显示IP地址与物理地址的对应关系



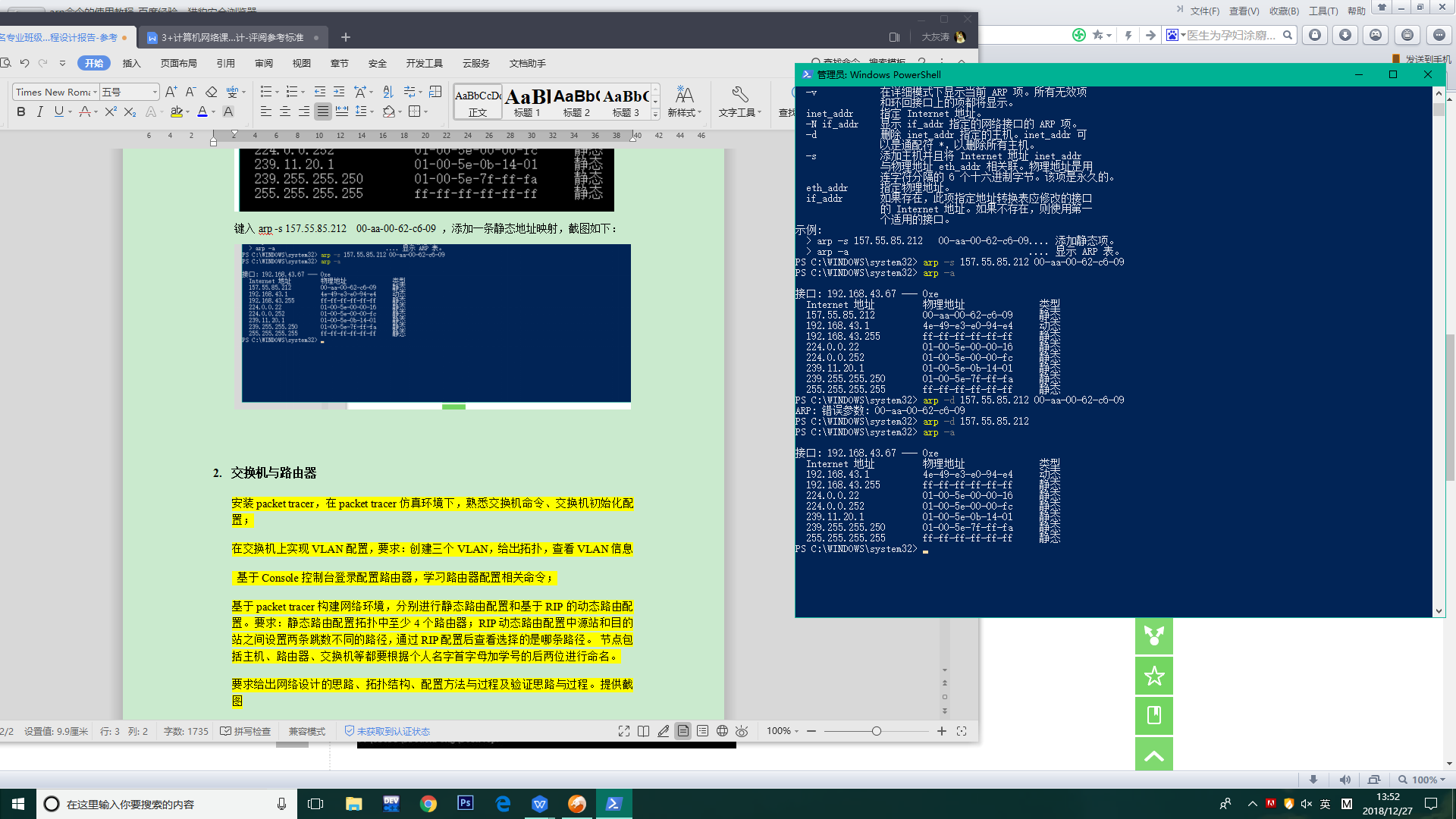
-图14-

1. 键入arp -s 157.55.85.212   00-aa-00-62-c6-09 ，添加一条静态地址映射



-图15-

1. 键入arp -d 157.55.85.212，来删除刚才加入的地址



-图16-

另外，当局域网中出现arp地址欺骗的时候，可以通过arp -d命令将本地存储的arp地址全部清空，重新获取.通过重新获取arp地址列表，可以解决网络突然掉线的问题.主要就是防止别的计算机模拟网关来欺骗本机电脑，本地的数据包数据都放到了别的计算机上，而没有发往网关，因此也就上不了网了.arp地址列表一清空，重新获取，恢复正常.

1. telnet: Telnet协议是TCP/IP协议家族中的一员，是Internet远程登陆服务的标准协议和主要方式.它为用户提供了在本地计算机上完成远程主机工作的能力.在终端使用者的电脑上使用telnet程序，用它连接到服务器.终端使用者可以在telnet程序中输入命令，这些命令会在服务器上运行，就像直接在服务器的控制台上输入一样.可以在本地就能控制服务器.要开始一个telnet会话，必须输入用户名和密码来登录服务器. telnet是常用的远程控制Web服务器的方法.

该命令的一般格式为: telnet [-a][-e escape char][-f log file][-l user][-t term][host [port]]

-a 企图自动登录。除了用当前已登陆的用户名以外，与 -l 选项相同。

-e 跳过字符来进入 telnet 客户端提示。

-f 客户端登录的文件名

-l 指定远程系统上登录用的用户名。要求远程系统支持 TELNET ENVIRON 选项。

-t 指定终端类型。支持的终端类型仅是: vt100, vt52, ansi 和 vtnt。

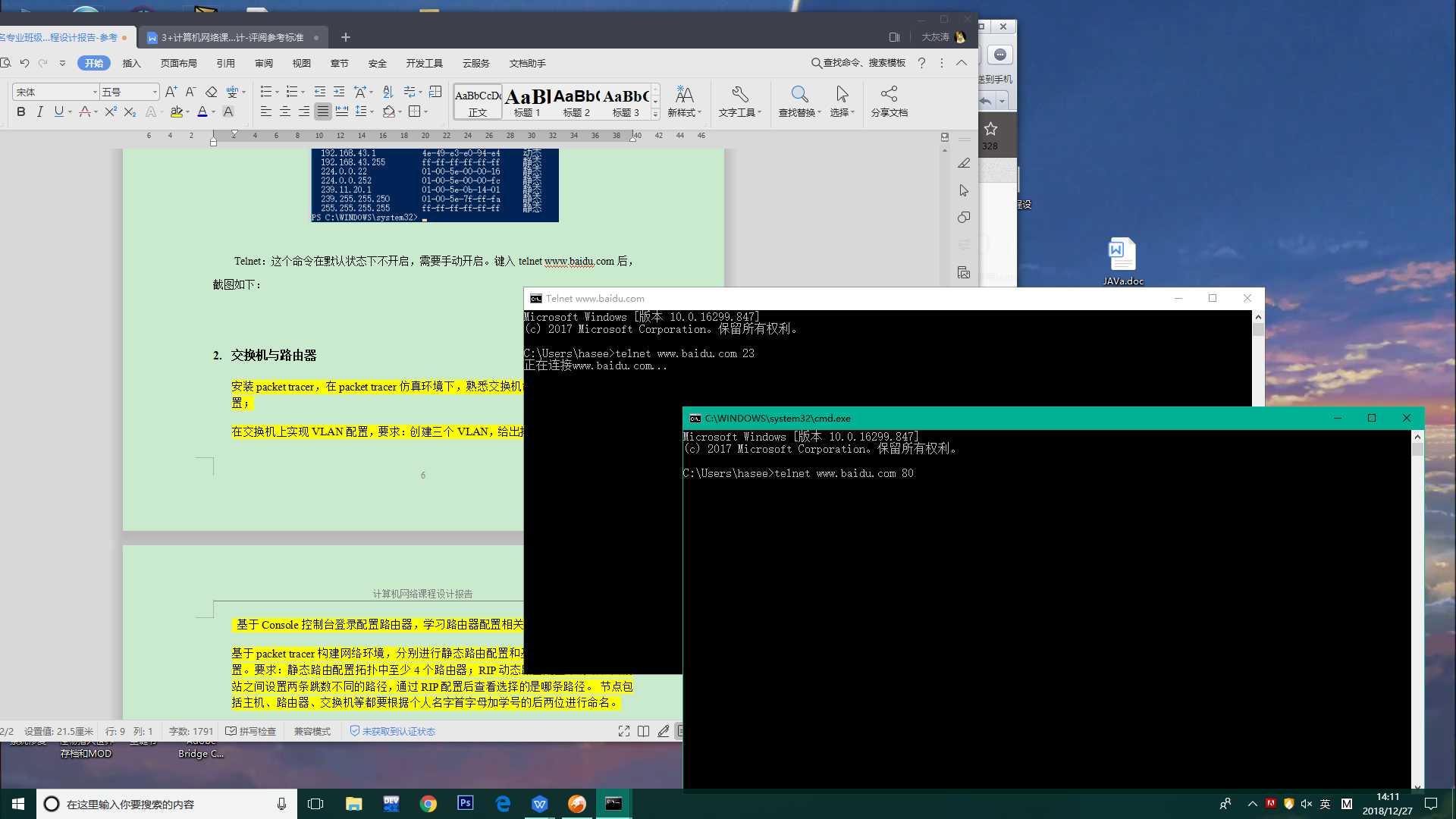
host 指定要连接的远程计算机的主机名或 IP 地址。

port 指定端口号或服务名。

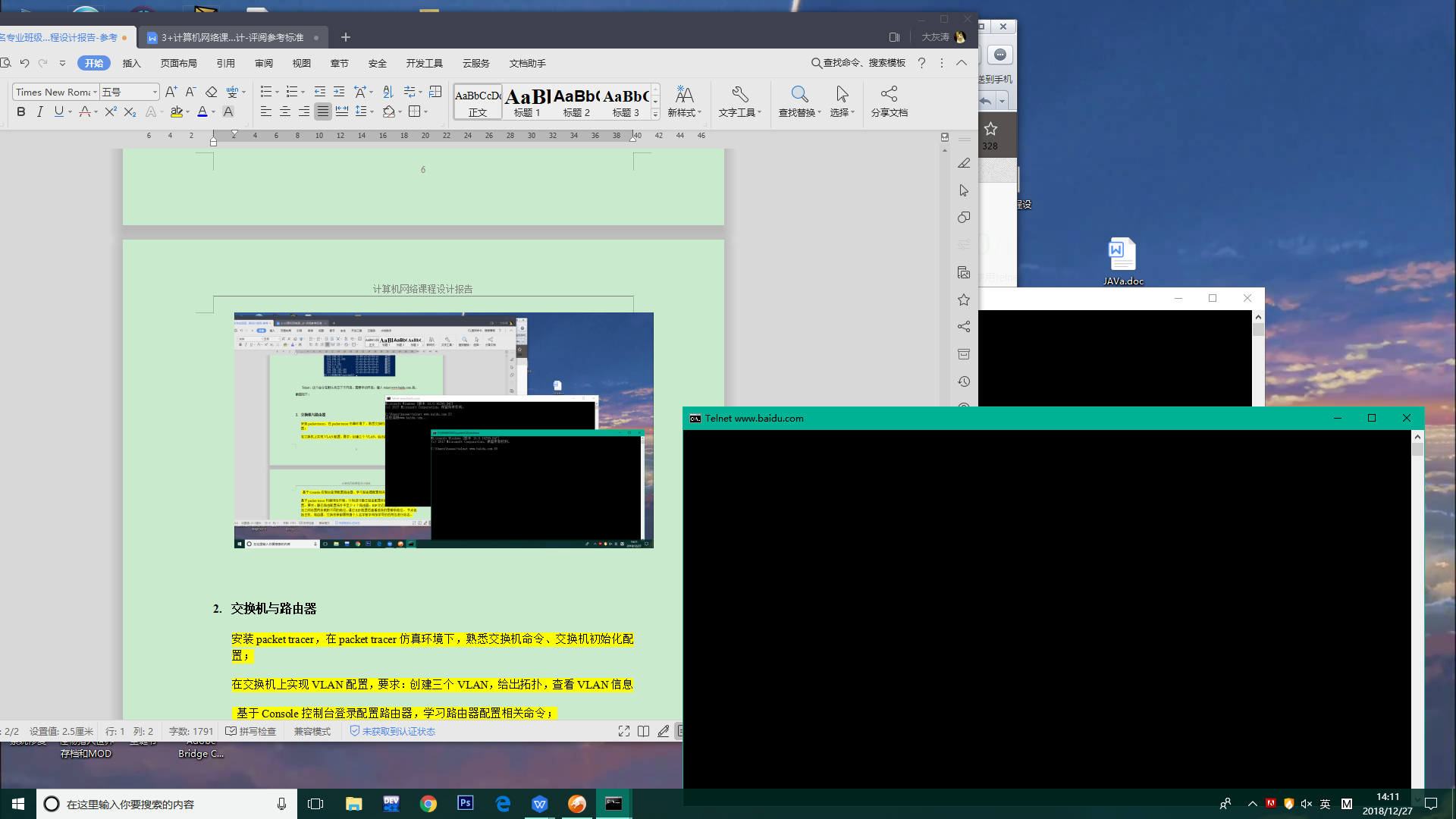
1. telnet命令在默认状态下不开启，需要手动开启. 键入telnet www.baidu.com，来完成测试端口号的操作：



-图17- 23号端口不开启



-图18-



-图19- 80号端口开启

**任务二：交换机与路由器**

1. 熟悉交换机命令、交换机初始化配置

交换机的命令行操作包括：

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

interface fa 0/1 进入交换机某个端口模式

exit 返回到上级模式

end 从全局以下模式返回到特权模式

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

1. 在交换机上实现 VLAN 配置

代码：

Switch>enable

Switch#conf t

Enter configuration commands, one per line. End with CNTL/Z.

Switch(config)#vlan 2

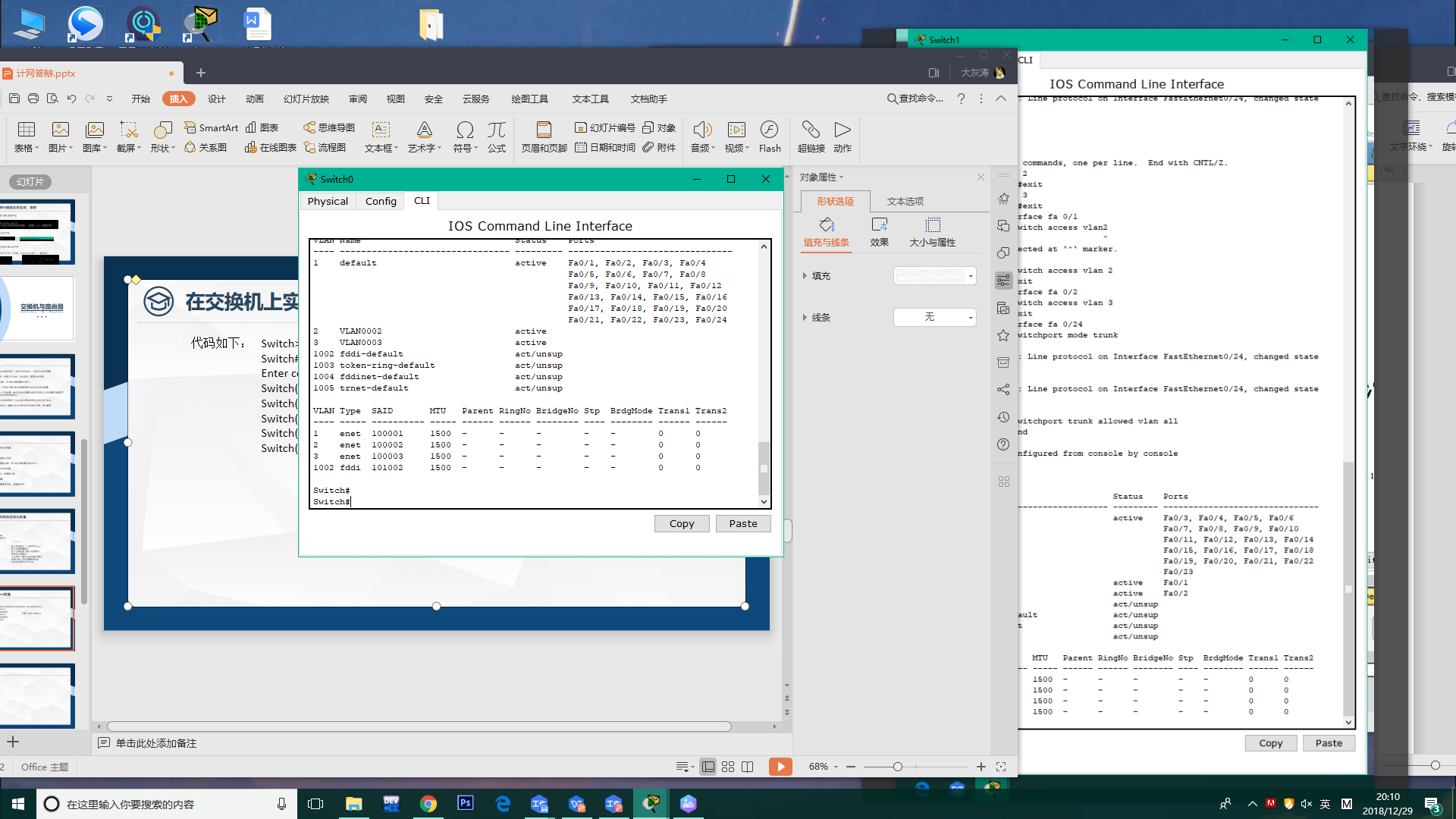
Switch(config-vlan)#exit

Switch(config)#vlan 3

Switch(config-vlan)#exit

Switch(config)#exit

通过命令show vlan可以看到创建成功.



-图20-

配置接口的代码：

Switch#enable

Switch#conf t

Enter configuration commands, one per line. End with CNTL/Z.

Switch(config)#interface fa 0/1

Switch(config-if)#switch access vlan 2

Switch(config-if)#exit

Switch(config)#exit

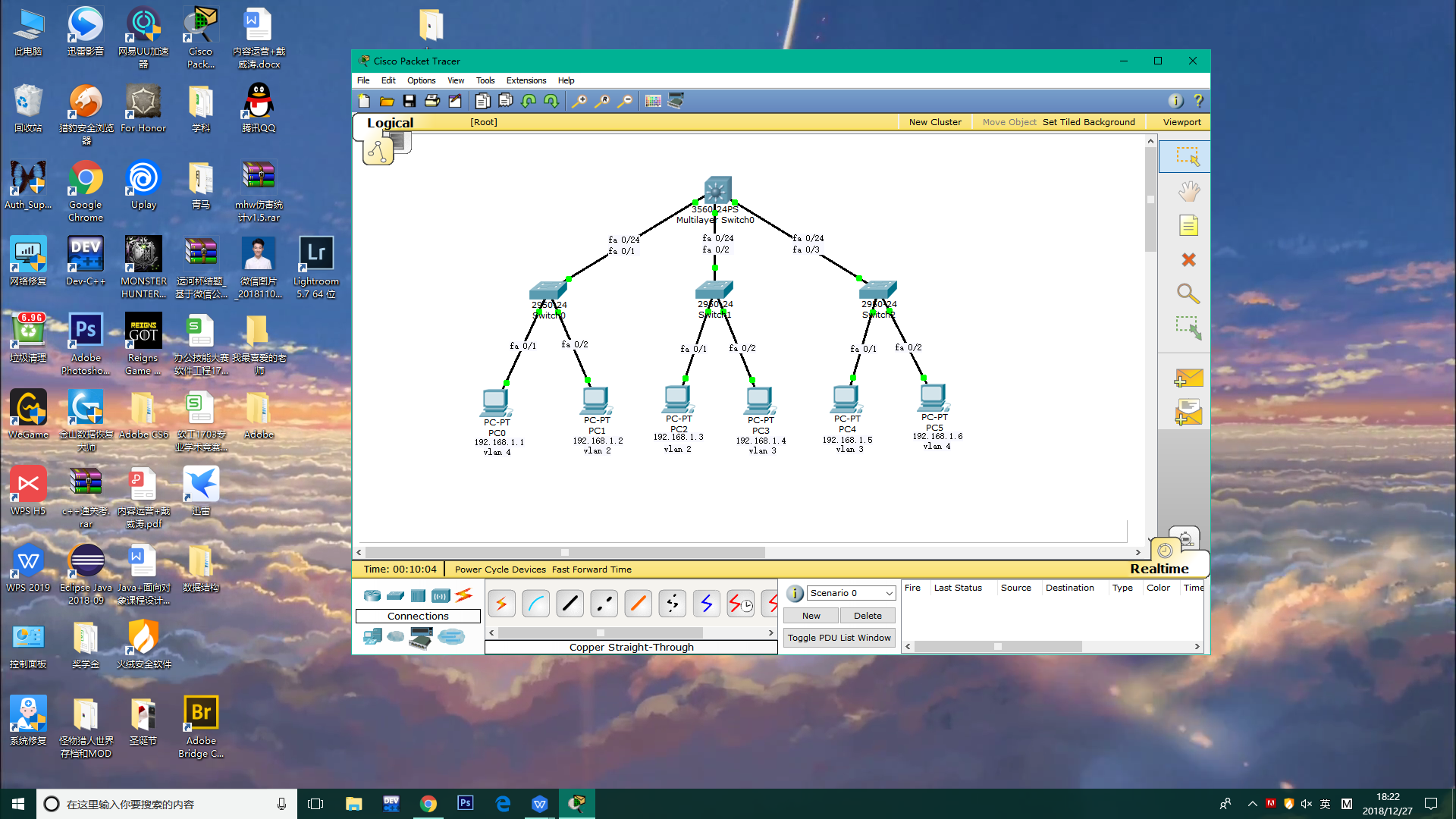
通过命令show vlan可以看到接口也配置完毕.

为了实现同一vlan的终端之间能够通信，三层交换机能建立对应vlan2、vlan3与vlan4的接口的实验目的. 在vlan配置时要完成的功能有两项：

1.创建vlan2、vlan3与vlan4.

2.将交换机端口以接入端口或者共享端口的方式分配给各个vlan，建立属于同一vlan的终端之间的二层交换路径和每一个vlan与三层交换机之间的二层交换路径.

网络拓扑结构：



-图21-

如果只要实现相同终端之间的通信，并不需要三层交换路径，如图：



-图22-

为PC分配IP地址

名称 相连的接口 IP地址

PC0 Fa0/1 192.168.1.1

PC1 Fa0/2 192.168.1.2

PC2 Fa0/1 192.168.1.3

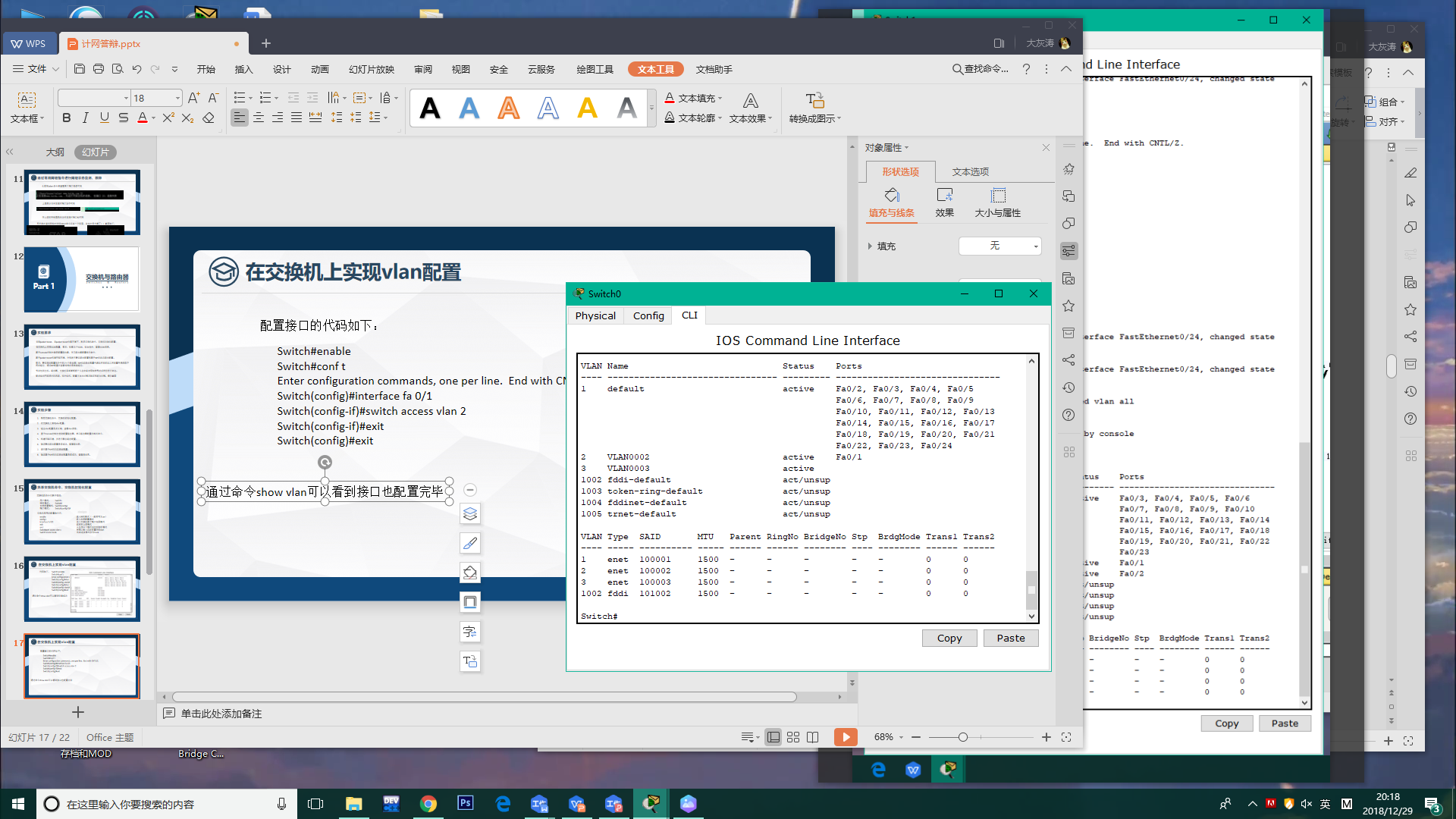
PC3 Fa0/2 192.168.1.4

PC4 Fa0/1 192.168.1.5

PC5 Fa0/2 192.168.1.6

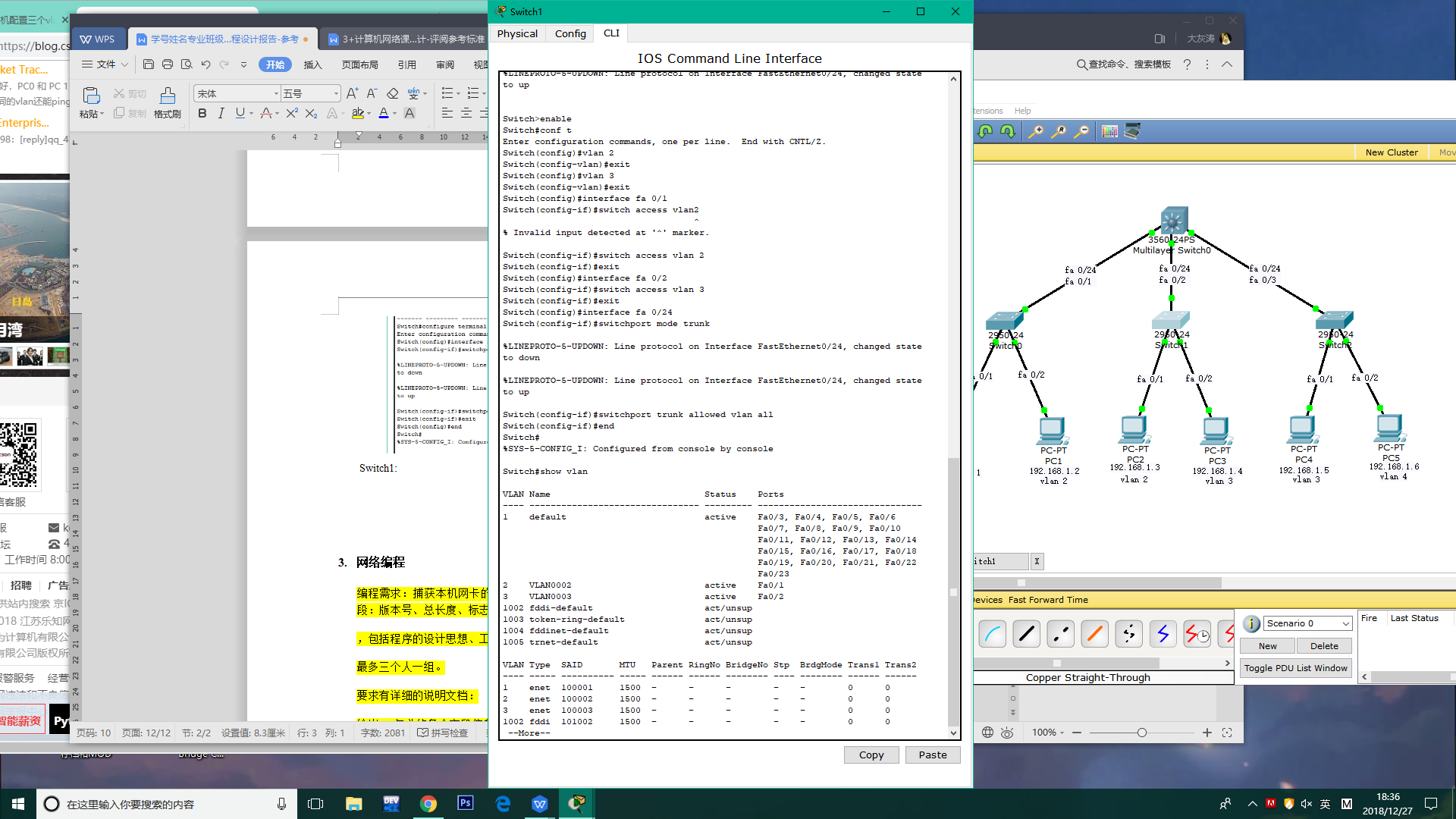
配置交换机的步骤为1.创建vlan 2.将交换机端口分配给vlan，如图：

Switch0：



-图23-

Switch1:



-图24-

Switch2:

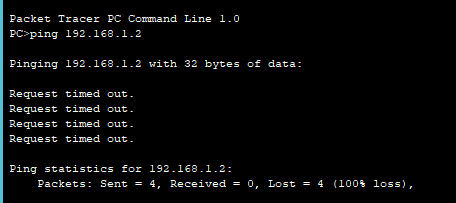


-图25-

验证：

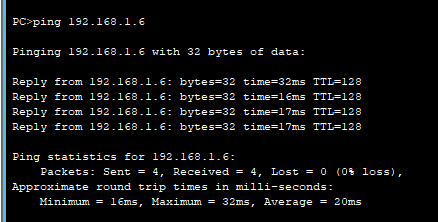
验证vlan配置是否正确，查看vlan信息，只需要不同vlan中的主机利用ping命令查看是否连通，相同valn中的主机利用ping命令查看是否连通即可验证。查看vlan信息则通过show vlan命令.

PC0 ping PC1



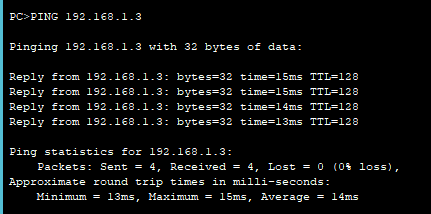
-图26-

PC0 ping PC5



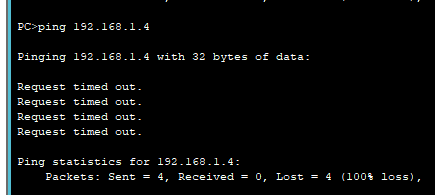
-图27-

PC1 ping PC2



-图28-

PC2 ping PC3



-图29-

**结论：可以得到只有相同vlan中的主机能相互ping通，与其他主机均不能ping通，原因是各种属于不同的vlan.**

1. 基于 Console 控制台登录配置路由器，学习路由器配置相关命令

设置特权模式密码，修改名字：

Router>en

Router #conf t

Router (config)#hostname R1

R1(config)#enable secret cisco //设置特权模式密码

R1(config)#exit

R1#exit

设置端口：

R1(config)#interface fa 0/0 //进入 fa 0/0 端口

R1(config-if)#ip address 192.168.1.1 255.255.255.0 //该端口配置相应的 IP 地址和子网掩码

R1(config-if)#no shutdown //开启端口

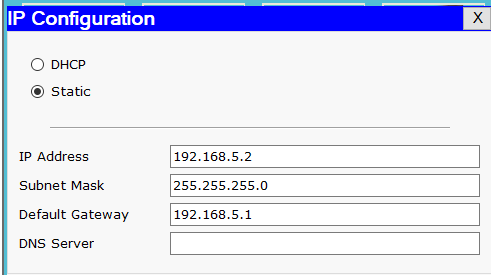
R1(config-if)#end

1. 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
2. 构建网络环境，并进行静态路由配置；

①设置两台主机

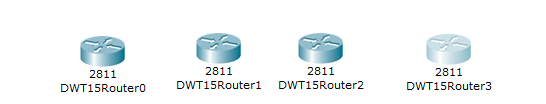


-图30-



-图31-

②设置四台路由器



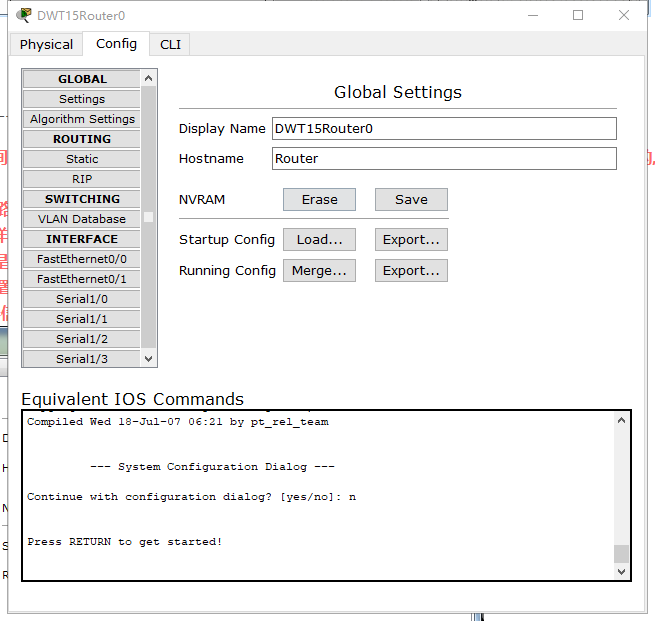
-图32-

关闭电源后，将要增加的串口模板拖动至相应位置，如图：



-图33-

之后开启电源，等待一会儿后看Config界面，如下图所示，则设置完毕：

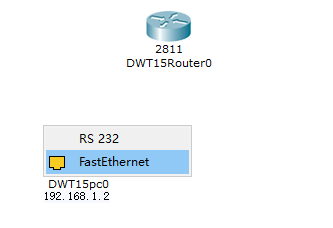


-图34-

同理设置其它三个路由器.

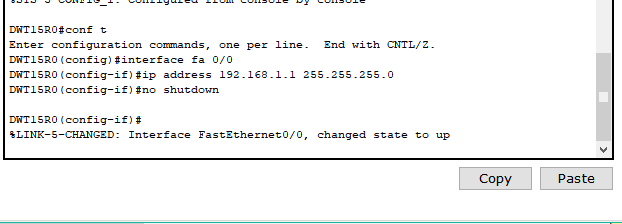
③连接主机与路由器

PC主机与路由器之间连接用双绞线，路由器之间用串口线.

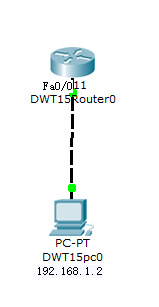


-图35-

连接到DWT15Rounter0的Fa0/0接口，随即设置相应接口的IP地址.



-图36-

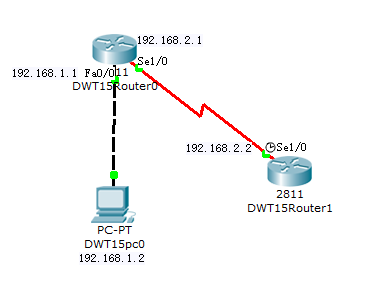


-图37-

设置成功后，用同样的方法设置DWT15pc1与DWT15Router3的连接.

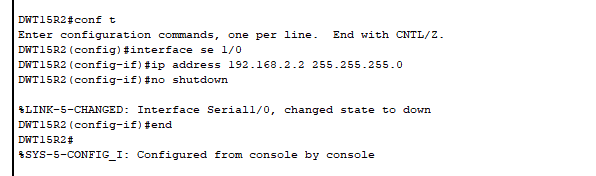
④连接路由器与路由器

路由器与路由器之间用串口线，在两个相互连接的路由器中有一个路由器的端口将会是DCE端口，另外一个就是DTE端口.



-图38-

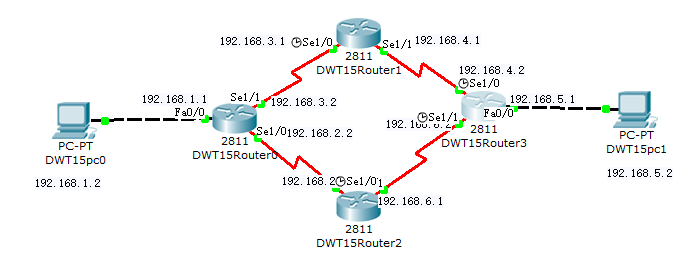
并且对相应的端口进行配置：



-图39-

同理连接其他路由器。

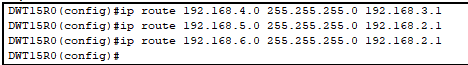
连接完毕后得到下面这个网络拓扑结构：



-图40-

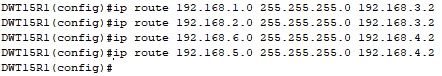
⑤配置静态路由表

DWT15Router0：



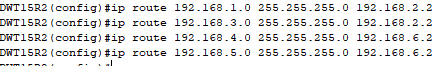
-图41-

DWT15Router1：



-图42-

DWT15Router2：



-图43-

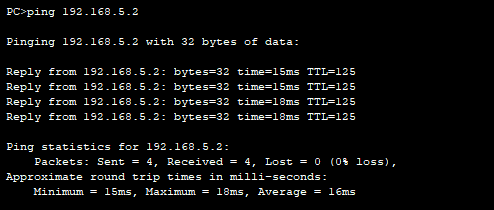
DWT15Router3：



-图44-

1. 验证静态路由配置是否成功，查看路由表；

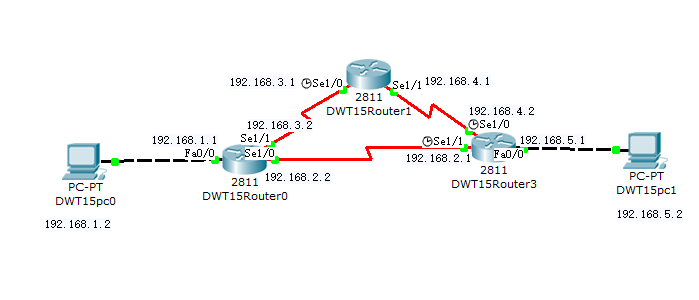
用主机0 ping 主机1成功，所以静态路由配置成功.



-图45-

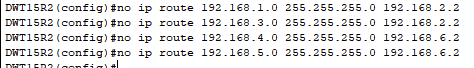
1. 进行基于RIP的动态路由配置；

①删除上个实验的路由表，为了有两条跳数不同的路径，在上一个实验的基础上减少一个路由器，配置方法同上一个实验.然后与DWT15Router0和DWT15Router1连接.于是有了如下网络拓扑结构：



-图46-

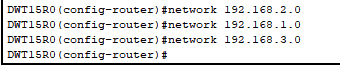
②删除原有路由表



-图47-

③配置的RIP动态路由，添加能直接到达的网段

DWT15Router0：



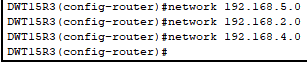
-图48-

DWT15Router1：



-图49-

DWT15Router3：

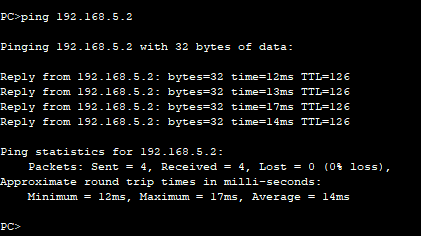


-图50-

1. 验证基于RIP的动态路由配置是否成功，查看路由表;

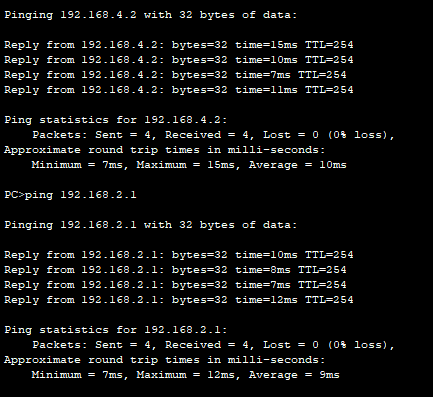
①验证网络是否连通

用PC0 ping PC1，成功，说明网络连通.



-图51-

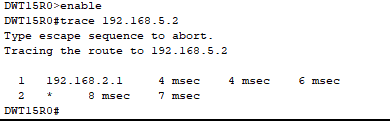
可以ping通192.168.4.2与192.168.2.1说明全网可达.



-图52-

②验证选路是否正确

利用trace命令可以看到在ping PC1时下一跳为192.168.5.2，即遵循RIP协议类型的矢量距离，选择了下面一条路，所以选路正确.



-图53-

**任务三：网络编程**

1.设计思想

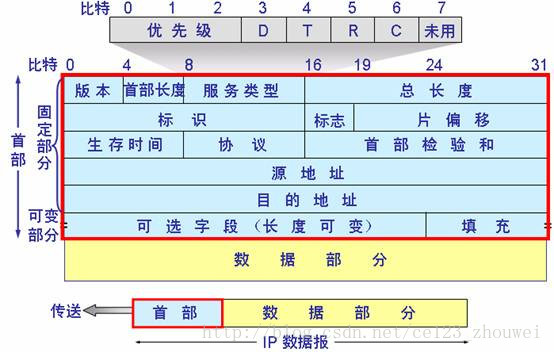
套接字:

在查阅网上的资料后，我知道了套接字这一概念。TCP用主机的IP地址加上主机上的端口好作为TCP连接的端点，这种端点就被叫做套接字（socket）

常用的套接字有三种，分别是流式套接字、数据报套接字与原始套接字。其中原始套接字允许对较低层次的协议直接访问，比如IP、ICMP协议。所以我们可以通过原始套接字来接受本机的IP包（包括IP头等信息头的最原始的数据信息）

IP数据包:

IP数据报分为两个部分，首部与数据，首部的前一部分是固定长度，共二十个字节，而在首部的固定部分后面是一些可选字段，其长度可变



-图54

按照这个结构，在实际编程时需要一个数据结构来表示IP数据报文的固定部分，定义如下：

typedef struct tIPPackHead

{

BYTE ver\_hlen; //IP协议版本和IP首部长度。高4位为版本，低4位为首部的长度(单位为4bytes)

BYTE byTOS; //服务类型

WORD wPacketLen; //IP包总长度。包括首部，单位为byte。[Big endian]

WORD wSequence; //标识，一般每个IP包的序号递增。[Big endian]

union

{

WORD Flags; //标志

WORD FragOf; //分段偏移

};

BYTE byTTL; //生存时间

BYTE byProtocolType; //协议类型，见PROTOCOL\_TYPE定义

WORD wHeadCheckSum; //IP首部校验和[Big endian]

DWORD dwIPSrc; //源地址

DWORD dwIPDes; //目的地址

BYTE Options; //选项

} IP\_HEAD;

1. 关键问题

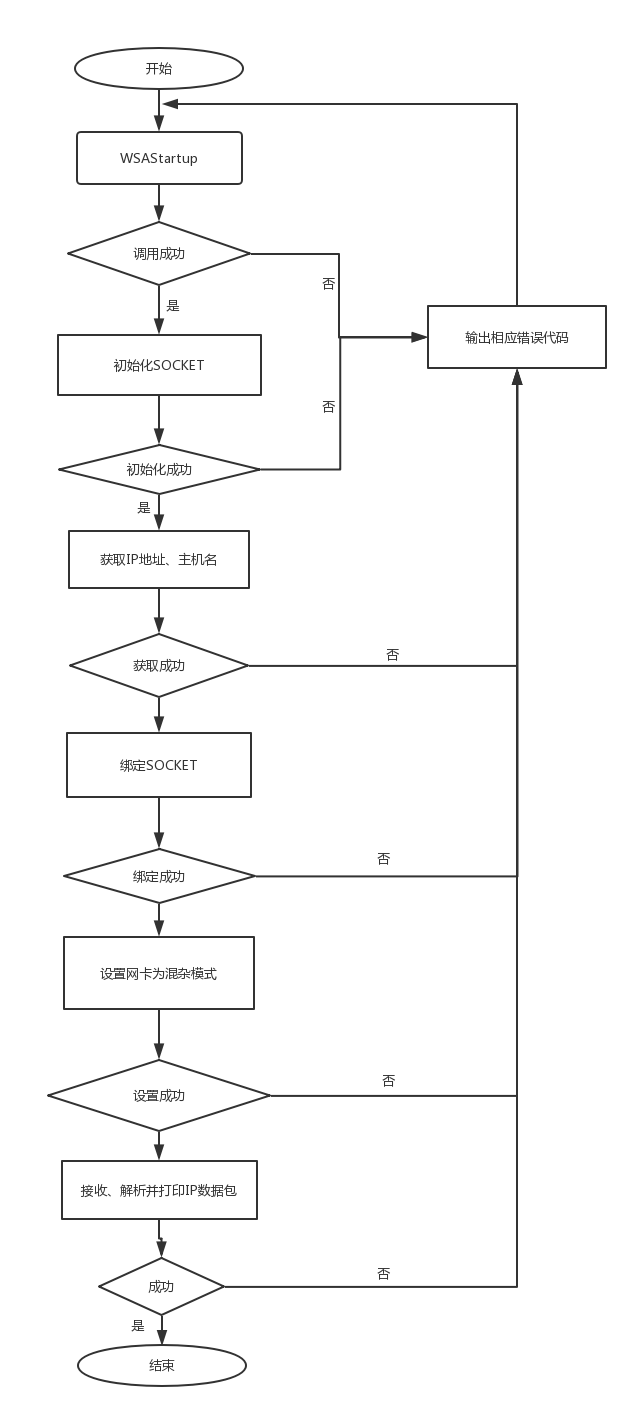
接下来使用套接字socket编程。基本流程为初始化套接字，然后监听数据包，解析数据包。

1. SOCKET sock=socket(AF\_INET,SOCK\_RAW,IPPROTO\_IP)，创建套接字，创建原始套接字，其原型为int socket(int domain, int type, int protocol)，参数protocol用来指明所要接收的协议包， IPPROTO\_IP代表IP协议， AF\_INET表示这是IPv4网络协议的套接字，SOCK\_RAW指的就是套接字类型为原始套接字

(2 ) 在通常情况下，网络通信的套接字程序只能响应与自己硬件地址相匹配的数据包或者是以广播形式发出去的数据包。对于其他形式的数据包，如果已经达到网络接口，但却不是发送到此地的数据包，网络接口在投递地址并非自身地址之后将不做响应，也就是说应用程序无法收取与自己无关的数据包。想要获取网络设备所有的数据包，就需要把网卡设置成混杂模式。

WSAIoctl(sock,IO\_RCVALL,&dwBufferInLen,sizeof(dwBufferInLen),&dwBufferLen,sizeof(dwBufferLen),&dwBytesReturned,NULL,NULL)用来把网卡设置为混杂模式。

1. recv(sock,buffer,65535,0)，该函数用于接收经过的IP包，其参数分别是套接字描述符，缓冲区的地址，缓冲区的大小
2. 工作流程



-图55-

1. 创建原始套接字后，IP头就会包含在接收的数据里面。然后，可以设置IP头操作选项。接下来是对socket的初始化工作：

/\*获取主机名\*/

char hostName[128];

if (-1 == gethostname(hostName, sizeof(hostName))) //，获取主机名，并判断是否获取成功

{

closesocket(sock); //关闭socket句柄，并释放相关资源。

cout << WSAGetLastError();//输出相应错误代码

system("pause");

return 0;

}

/\*获取本机IP地址\*/

struct hostent \* pHostent;

pHostent = gethostbyname(hostName);

//填充SOCKADDR\_IN结构的内容

sockaddr\_in addr;

addr.sin\_family = AF\_INET; //代表TCP/IP协议族。

addr.sin\_addr = \*(in\_addr\*)pHostent->h\_addr\_list[0]; //IP

addr.sin\_port = 8888; //端口，IP层端口可随意填

/\*绑定socket\*/

if (SOCKET\_ERROR == bind(sock, (sockaddr \*)&addr, sizeof(addr)))

//socket名称包含"协议， ip地址, 端口号"这三个要素， 而命名就是通过调用bind函数把socket与这三个要素绑定一起来 {

closesocket(sock);

cout << WSAGetLastError();

system("pause");

return 0;

}

填写sockaddr\_in的内容时，其地址值应填写为本机IP地址，本机IP地址可以通过gethostbyname()函数获取；端口号可以随便填写，但是不能与系统冲突；协议族应该填写为AF\_INET。使用htons()函数可以将无符号短整型的主机数据转换为网络字节顺序的数据。最后使用bind()函数将socket绑定到本机网卡上

绑定网卡后，需要使用WSAIoct1()函数把网卡设置为混杂模式，使网卡能够接收所有的网络数据，其关键代码如下：

u\_long sioarg = 1;

DWORD wt = 0;

if (SOCKET\_ERROR == WSAIoctl(sock, SIO\_RCVALL, &sioarg, sizeof(sioarg), NULL, 0, &wt, NULL, NULL))

//WSAIoctl(sock,IO\_RCVALL,&dwBufferInLen,sizeof(dwBufferInLen),&dwBufferLen,sizeof(dwBufferLen),&dwBytesReturned,NULL,NULL)用来把网卡设置为混杂模式

{

closesocket(sock);

cout << WSAGetLastError();

system("pause");

return 0;

}

如果接收到的数据包中的协议类型和定义的初始套接字匹配，那么接受到的数据就拷贝的套接字中。因此，网卡就可以接收所有经过的IP包

1. 接收数据包

程序中用recv()函数来接收经过的IP包。该函数有四个参数，第一个参数接收操作所用的套接字描述符；第二个参数接收缓冲区地址；第三个参数接收缓冲区大小，也就是要接受的字节数；第四个参数是一个附加标志，如果对所发送的数据没有特殊要求，直接设为0.因为IP数据包的最大长度是65535B，因此，缓冲区的大小不能小于65535B。设置缓冲区后，可利用循环来反复监听接收IP包，用recv()函数实现接收能力的代码如下：

cnt = 1;

char buf[65535];

int len = 0;

do

{

len = recv(sock, buf, sizeof(buf), 0);

//recv(sock, buffer, 65535, 0) 函数用来接收经过的IP包，参数分别为套接字描述符，缓冲区的地址，缓冲区的大小

if (len > 0)

{

DecodeIP(buf, len);//解析接收的数据包

}

} while (len > 0);

1. IP包的解析

在捕获IP数据包后，可以通过指针把缓冲区的内容强制转化为IP数据结构

IP\_HEAD iphead=\*(IP\*)buffer;

通过IPH解析解析IP头各个字段的代码：

/\*获取版本字段\*/

iphead.Version>>4;

/\*获取首部长度\*/

iphead.HarLen & 0x0f;

/\*获取服务类型字段中的优先级子域\*/

iphead.ServiceType>>5;

/\*获取服务类型字段中的TOS子域\*/

(iphead.ServiceType>>1)&0x0f;

/\*获取总长度字段\*/

iphead.TotalLen;

/\*获取标识字段\*/

iphead.ID;

/\*解析标志字段\*/

DF=(iphead.Flags>>14) & 0x01;

MF=(iphead.Flags>>13)&0x01;

/\*获取片偏移字段\*/

iphead.Frag0ff&0x1fff;

/\*获取生存时间字段\*/

iphead.TimeToLive;

/\*获取协议字段\*/

iphead.Protocol;

/\*获取首部校验和字段\*/

iphead.HdrChksum;

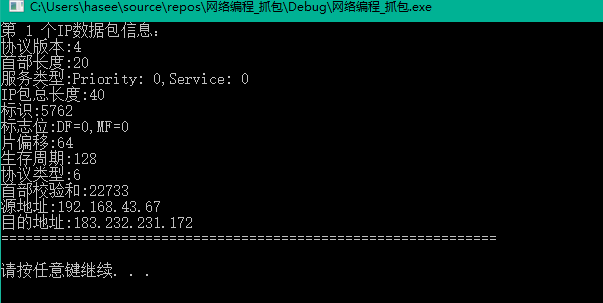
/\*解析源IP地址字段\*/

inet\_ntoa(\*(in\_addr\*)&iphead.SrcAddr);

/\*解析目的IP地址字段\*/

inet\_ntoa(\*(in\_addr\*)&iphead.DstAddr);

1. 运行截图



-图56-

**三、总结**

1.掌握了以下常用网络命令功能

ipconfig：查看本机IP配置。

ping：检查网络是否连通或者网络连接速度的命令。

tracert：追踪路由，检查网络状态。

arp：显示和修改“地址解析协议（ARP）”缓存中的项目。

telnet：可以检查一个端口是否可以访问。

netstat：是一个监控TCP/IP网络的工具，可以完成显示所有连接和侦听窗口，显示路由表，显示当前连接卸载状态等操作。

1. 掌握了路由器和交换机的区别
2. 得到以下结论：只有相同vlan中的主机能相互ping通，与其他主机均不能ping通，原因是各种属于不同的vlan

4.掌握了静态路由配置和基于 RIP 的动态路由配置.