

计算机网络课程实验报告

实验一

姓名		院系	人工智能与自动化学院	学号	
实验地点	科技楼 12 楼		实验日期	2022 年 4 月 28 日	
实验目的:					
1. 掌握常用网络命令的使用方法; 2. 熟悉和掌握网络管理、网络维护的基本内容和方法					
实验内容:					
<p>(1) ARP: 该命令可用于显示和修改 IP 地址与物理地址之间的转换表</p> <p>ARP -s inet_addr eth_addr [if_addr] ARP -d inet_addr [if_addr] ARP -a [inet_addr] [-N if_addr]</p> <ul style="list-style-type: none">-a 显示当前的 ARP 信息, 可以指定网络地址-d 删除由 inet_addr 指定的主机. 可以使用* 来删除所有主机.-s 添加主机, 并将网络地址跟物理地址相对应, 这一项是永久生效的。 <pre>PS C:\Users\1> arp -a 接口: 10.19.205.131 --- 0xb Internet 地址 物理地址 类型 10.19.207.254 00-74-9c-92-27-42 动态 10.19.207.255 ff-ff-ff-ff-ff-ff 静态 224.0.0.22 01-00-5e-00-00-16 静态 224.0.0.251 01-00-5e-00-00-fb 静态 224.0.0.252 01-00-5e-00-00-fc 静态 239.255.255.250 01-00-5e-7f-ff-fa 静态 255.255.255.255 ff-ff-ff-ff-ff-ff 静态 PS C:\Users\1> arp -a 10.19.207.254 接口: 10.19.205.131 --- 0xb Internet 地址 物理地址 类型 10.19.207.254 00-74-9c-92-27-42 动态 PS C:\Users\1></pre>					
<p>(2) Ipconfig: 该诊断命令显示所有当前的 TCP/IP 网络配置值。该命令在运行 DHCP 系统上的特殊用途, 允许用户决定 DHCP 配置的 TCP/IP 配置值。</p> <p>ipconfig [/? /all /release [adapter] /renew [adapter] /flushdns /registerdns /showclassid adapter /setclassid adapter [classidtoreset]]</p> <p>/all 产生完整显示。在没有该开关的情况下 ipconfig 只显示 IP 地址、子网掩码和每个网卡的默认网关值。</p>					

无线局域网适配器 WLAN:

```
连接特定的 DNS 后缀 . . . . . :  
IPv6 地址 . . . . . : 2001:250:4000:8231:48c2:5f40:7397:bffc  
临时 IPv6 地址 . . . . . : 2001:250:4000:8231:20c9:8dee:5869:57e6  
本地链接 IPv6 地址 . . . . . : fe80::48c2:5f40:7397:bffc%11  
IPv4 地址 . . . . . : 10.19.205.131  
子网掩码 . . . . . : 255.255.248.0  
默认网关 . . . . . : fe80::274:9cff:fe92:2742%11  
10.19.207.254
```

以太网适配器 蓝牙网络连接:

```
媒体状态 . . . . . : 媒体已断开连接  
连接特定的 DNS 后缀 . . . . . :
```

PS C:\Users\1> █

(3) Nbtstat: 该诊断命令使用 NBT (TCP/IP 上的 NetBIOS) 显示协议统计和当前 TCP/IP 连接。该命令只有在安装了 TCP/IP 协议之后才可用。

nbtstat [-a remotename] [-A IP address] [-c] [-n] [-R] [-r] [-S] [-s] [interval]

参数

- a remotename 使用远程计算机的名称列出其名称表。
 - A IP address 使用远程计算机的 IP 地址并列出名称表。
 - c 给定每个名称的 IP 地址并列出 NetBIOS 名称缓存的内容。
 - n 列出本地 NetBIOS 名称。“已注册”表明该名称已被广播 (Bnode) 或者 WINS (其他节点类型) 注册。
 - R 清除 NetBIOS 名称缓存中的所有名称后, 重新装入 Lmhosts 文件。
 - r 列出 Windows 网络名称解析的名称解析统计。在配置使用 WINS 的 Windows 2000 计算机上, 此选项返回要通过广播或 WINS 来解析和注册的名称数。
 - S 显示客户端和服务会话, 只通过 IP 地址列出远程计算机。
 - s 显示客户端和服务会话。尝试将远程计算机 IP 地址转换成使用主机文件的名称。
- interval 重新显示选中的统计, 在每个显示之间暂停 interval 秒。按 CTRL+C 停止重新显示统计信息。如果省略该参数, nbtstat 打印一次当前的配置信息。

```
WLAN:
节点 IP 地址: [10.19.205.131] 范围 ID: []
```

NetBIOS 本地名称表

名称	类型	状态
LAPTOP-LENOVO-Y<20>	唯一	已注册
LAPTOP-LENOVO-Y<00>	唯一	已注册
WORKGROUP	<00> 组	已注册

```
本地连接* 3:
节点 IP 地址: [0.0.0.0] 范围 ID: []
```

缓存中没有名称

```
本地连接* 12:
节点 IP 地址: [0.0.0.0] 范围 ID: []
```

NetBIOS 本地名称表

名称	类型	状态
LAPTOP-LENOVO-Y<20>	唯一	已注册

```
PS C:\Users\1> nbtstat -S
```

```
以太网 2:
节点 IP 地址: [0.0.0.0] 范围 ID: []
```

(4) **nslookup**: 允许主机向指定的 DNS 服务器查询某个 DNS 记录。如果没有指明 DNS 服务器, **nslookup** 将把查询请求发向默认的 DNS 服务器。

nslookup 的一般格式是:

```
nslookup -option1 -option2 host-to-find dns-server
```

ipconfig 命令用来显示你当前的 TCP/IP 信息, 包括: 你的地址、DNS 服务器的地址、适配器的类型等信息。如果, 要显示与主机相关的信息用命令:

```
ipconfig/all
```

如果查看 DNS 缓存中的记录用命令:

```
ipconfig/displaydns
```

要清空 DNS 缓存, 用命令:

```
ipconfig /flushdns
```

```
PS C:\Users\1> nslookup www.baidu.com
服务器: dns.hust.edu.cn
Address: 202.114.0.131

非权威应答:
名称: www.a.shifen.com
Addresses: 182.61.200.7
           182.61.200.6
Aliases: www.baidu.com
```

(5) Netstat: 显示协议统计和当前的 TCP/IP 网络连接。该命令只有在安装了 TCP/IP 协议后才可以使⤵用。

```
netstat [-a] [-e] [-n] [-s] [-p protocol] [-r] [interval]
```

参数

-a 显示所有连接和侦听端口。服务器连接通常不显示。

-e 显示以太网统计。该参数可以与 -s 选项结合使用。

-n 以数字格式显示地址和端口号（而不是尝试查找名称）。

-s 显示每个协议的统计。默认情况下，显示 TCP、UDP、ICMP 和 IP 的统计。-p 选项可以用来指定默认的子集。

-p protocol 显示由 protocol 指定的协议的连接；protocol 可以是 tcp 或 udp。如果与 -s 选项一同使用显示每个协议的统计，protocol 可以是 tcp、udp、icmp 或 ip。

-r 显示路由表的内容。

Interval 重新显示所选的统计，在每次显示之间暂停 interval 秒。按 CTRL+B 停止重新显示统计。如果省略该参数，netstat 将打印一次当前的配置信息。

```
PS C:\Users\1> netstat -as
```

IPv4 统计信息

```
接收的数据包                = 243128
.....
```

IPv6 统计信息

```
接收的数据包                = 14427
.....
```

ICMPv4 统计信息

```
.....
```

ICMPv6 统计信息

```
.....
```

IPv4 的 TCP 统计信息

```
主动开放                    = 3039
被动开放                    = 1676
失败的连接尝试              = 427
重置连接                    = 167
当前连接                    = 22
接收的分段                  = 255357
发送的分段                  = 142553
重新传输的分段              = 0
```

IPv6 的 TCP 统计信息

主动开放	= 341
被动开放	= 0
失败的连接尝试	= 98
重置连接	= 39
当前连接	= 6
接收的分段	= 13979
发送的分段	= 3444
重新传输的分段	= 0

IPv4 的 UDP 统计信息

接收的数据报	= 7803
无端口	= 329
接收错误	= 0
发送的数据报	= 6719

IPv6 的 UDP 统计信息

接收的数据报	= 823
无端口	= 0
接收错误	= 0
发送的数据报	= 695

(6) Ping: 验证与远程计算机的连接。该命令只有在安装了 TCP/IP 协议后才可以使⤵用。

ping [-t] [-a] [-n count] [-l length] [-f] [-i ttl] [-v tos] [-r count] [-s count] [[-j computer-list] | [-k computer-list]] [-w timeout] destination-list

参数

- t Ping 指定的计算机直到中断。
- a 将地址解析为计算机名。
- n count 发送 count 指定的 ECHO 数据包数。默认值为 4。
- l length 发送包含由 length 指定的数据量的 ECHO 数据包。默认为 32 字节；最大值是 65,527。
- f 在数据包中发送“不要分段”标志。数据包就不会被路由上的网关分段。
- i ttl 将“生存时间”字段设置为 ttl 指定的值。
- v tos 将“服务类型”字段设置为 tos 指定的值。
- r count 在“记录路由”字段中记录传出和返回数据包的路由。count 可以指定最少 1 台，最多 9 台计算机。
- s count 指定 count 指定的跃点数的时间戳。
- j computer-list 利用 computer-list 指定的计算机列表路由数据包。连续计算机可以被中间网关分隔（路由稀疏源）IP 允许的最大数量为 9。
- k computer-list 利用 computer-list 指定的计算机列表路由数据包。连续计算机不能被中

间网关分隔（路由严格源）IP 允许的最大数量为 9。

-w timeout 指定超时间隔，单位为毫秒。

destination-list 指定要 ping 的远程计算机。

较一般的用法是 ping -t www.zju.edu.cn

```
PS C:\Users\1> ping www.zju.edu.cn

正在 Ping www.zju.edu.cn.w.cdngslb.com [2001:da8:20d:40d5:3::3fe] 具有 32 字节的数据:
来自 2001:da8:20d:40d5:3::3fe 的回复: 时间=11ms
来自 2001:da8:20d:40d5:3::3fe 的回复: 时间=12ms
来自 2001:da8:20d:40d5:3::3fe 的回复: 时间=11ms
来自 2001:da8:20d:40d5:3::3fe 的回复: 时间=10ms

2001:da8:20d:40d5:3::3fe 的 Ping 统计信息:
    数据包: 已发送 = 4, 已接收 = 4, 丢失 = 0 (0% 丢失),
往返行程的估计时间(以毫秒为单位):
    最短 = 10ms, 最长 = 12ms, 平均 = 11ms
```

(7) Route: 控制网络路由表。该命令只有在安装了 TCP/IP 协议后才可以使用。

route [-f] [-p] [command [destination] [mask subnetmask] [gateway] [metric costmetric]]

参数

-f 清除所有网关入口的路由表。如果该参数与某个命令组合使用，路由表将在运行命令前清除。

-p 该参数与 add 命令一起使用时，将使路由在系统引导程序之间持久存在。默认情况下，系统重新启动时不保留路由。与 print 命令一起使用时，显示已注册的持久路由列表。忽略其他所有总是影响相应持久路由的命令。

Command 指定下列的一个命令。

命令 目的

print 打印路由

add 添加路由

delete 删除路由

change 更改现存路由

destination 指定发送 command 的计算机。

mask subnetmask 指定与该路由条目关联的子网掩码。如果没有指定，将使用 255.255.255.255。

gateway 指定网关。

metric costmetric 指派整数跃点数（从 1 到 9999）在计算最快速、最可靠和（或）最便宜的路由时使用。

本机 ip 为 10.19.205.131，缺省网关是 10.19.207.254 假设此网段上另有一网关 10.111.142.254，现在想添加一项路由，使得当访问 10.13.0.0 子网络时通过这一个网关

IPv4 路由表

活动路由：

网络目标	网络掩码	网关	接口	跃点数
0.0.0.0	0.0.0.0	10.19.207.254	10.19.205.131	35
10.13.0.0	255.255.0.0	10.19.207.254	10.19.205.131	36
10.19.200.0	255.255.248.0		在链路上	10.19.205.131 291
10.19.205.131	255.255.255.255		在链路上	10.19.205.131 291
10.19.207.255	255.255.255.255		在链路上	10.19.205.131 291
127.0.0.0	255.0.0.0		在链路上	127.0.0.1 331
127.0.0.1	255.255.255.255		在链路上	127.0.0.1 331
127.255.255.255	255.255.255.255		在链路上	127.0.0.1 331
224.0.0.0	240.0.0.0		在链路上	127.0.0.1 331
224.0.0.0	240.0.0.0		在链路上	10.19.205.131 291
255.255.255.255	255.255.255.255		在链路上	127.0.0.1 331
255.255.255.255	255.255.255.255		在链路上	10.19.205.131 291

永久路由：

无

看到路由表已添加

(8) Tracert：该诊断实用程序将包含不同生存时间 (TTL) 值的 Internet 控制消息协议 (ICMP) 回显数据包发送到目标，以决定到达目标采用的路由。要在转发数据包上的 TTL 之前至少递减 1，必需路径上的每个路由器，所以 TTL 是有效的跃点计数。数据包上的 TTL 到达 0 时，路由器应该将“ICMP 已超时”的消息发送回源系统。Tracert 先发送 TTL 为 1 的回显数据包，并在随后的每次发送过程将 TTL 递增 1，直到目标响应或 TTL 达到最大值，从而确定路由。路由通过检查中级路由器发送回的“ICMP 已超时”的消息来确定路由。不过，有些路由器悄悄地下传包含过期 TTL 值的数据包，而 tracert 看不到。

tracert [-d] [-h maximum_hops] [-j computer-list] [-w timeout] target_name

参数

/d 指定不将地址解析为计算机名。

-h maximum_hops 指定搜索目标的最大跃点数。

-j computer-list 指定沿 computer-list 的稀疏源路由。

-w timeout 每次应答等待 timeout 指定的微秒数。

target_name 目标计算机的名称。

```
PS C:\WINDOWS\system32> tracert www.hust.edu.cn

通过最多 30 个跃点跟踪
到 www.hust.edu.cn [2001:250:4000:2000::245] 的路由：

  1      2 ms      2 ms      2 ms  2001:250:4000:8231::1
  2      2 ms      1 ms      2 ms  2001:250:4000:4980::1
  3      2 ms      1 ms      2 ms  2001:250:4000:4909::2
  4      2 ms      1 ms      1 ms  2001:250:4000:2000::245

跟踪完成。
PS C:\WINDOWS\system32>
```

实验过程中遇到的问题如何解决的？

实验二

姓名		院系	人工智能与自动化学院	学号	
实验地点	科技楼 12 楼		实验日期	2022 年 5 月 3 日	

实验目的：

1. 了解抗干扰编码原理。
2. 掌握海明编码和 CRC 编码的原理,能熟练计算。

实验内容：

实验内容：

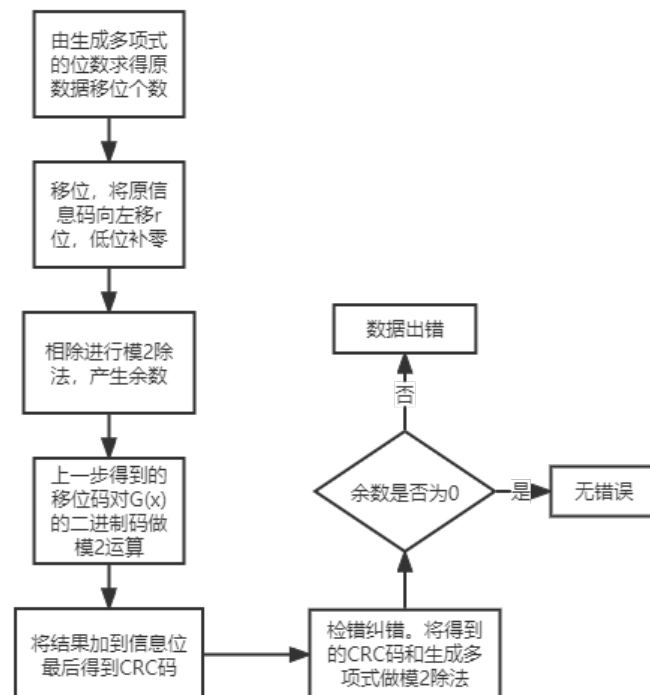
- 1、编程实现海明编码，包括编码过程和译码过程；要求有数据输入、结果输出界面；编程语言不限，开发平台不限，但要求功能完善，且有友好、功能完善的图形人机界面。
- 2、编程实现 CRC 编码，包括编码过程和译码过程；要求有数据输入、生成多项式输入、结果输出界面；编程语言不限，开发平台不限，但要求功能完善，且有友好、功能完善的图形人机界面。

思考问题：

- 1) CRC 编码和海明编码的主要区别有哪些？
- 2) 海明编码能纠多位错误吗？
- 3) CRC 编码的检错能力与生成多项式最高幂次数值有关系吗？

CRC 编码

1. 流程图



2. 代码

```
from tkinter import *

def division_process(data_bits, divisor):
    rem = [0 for x in range(len(divisor))] # To store the remainder
    data_with_pattern = [0 for x in range(len(data_bits) + len(divisor))] # Array of
    length sum of parameters lengths

    for i in range(len(data_bits)):
        data_with_pattern[i] = data_bits[i]
    for i in range(len(divisor)):
        rem[i] = data_with_pattern[i]
    # data_bits[i] -> data_with_pattern[i] -> rem[i]

    for i in range(len(data_bits)):
        # print('First data bit:', rem[0])
        # print('remainder:')
        if rem[0] == 1:
            for j in range(1, len(divisor)): # bit XOR
                rem[j-1] = rem[j] ^ divisor[j]
                # print(rem[j-1])
        else:
            for j in range(1, len(divisor)):
                rem[j-1] = rem[j]
                # print(rem[j-1])
            rem[len(divisor) - 1] = data_with_pattern[i + len(divisor)]
    return rem

def receiver(got, pattern_bits): # The receiver should do division also as the sender
does
    rem = division_process(got, pattern_bits)
    for k in range(len(rem)): # Check the remainder == 0 (looping bit by bit)
        if rem[k] != 0:
            return False
    return True

def crc_code(data_input, pattern_input):
    ''' Start of the program '''

    # print('Read a stream of bits for "DATA":')
    stream_data = data_input

    data = []
    for bit in stream_data:
        data.append(int(bit)) # Copy stream_data to data
        if(int(bit)!=0 and int(bit)!=1):
            print('Error Input!')
            exit() # Check Illegal Input

    ''' Data is read and put in data array now as integers '''

    # print('Read a stream of bits for "PATTERN":')
```

```

stream_divisor = pattern_input
# Pattern is used as Divisor
pattern = []
for bit in stream_divisor:
    pattern.append(int(bit))
    if (int(bit) != 0 and int(bit) != 1):
        print('Error Input!')
        exit() # Check Illegal Input

''' Divisor is read and put in pattern array now as integers '''

remainder = division_process(data, pattern)
# Divide the data with pattern

print('The remainder:')
for i in range(len(remainder) - 1):
    print('remainder[', i, '] =', remainder[i])

generated = []

# print('The generated code from CRC (Data will be sent from sender):')
for i in data:
    generated.append(i)
for i in range(len(remainder) - 1):
    generated.append(remainder[i])

# print(generated) # Sender
return generated

''' Receiver check the arrived data and decide if there is an error '''
arrived_data = [0 for j in range(len(data) + len(remainder) - 1)] # Receiver
# print('Read the data arrived to receiver to check if there is an error using CRC:')
received = input()
for k in range(len(received)):
    arrived_data[k] = int(received[k])

''' Data is being arrived to receiver '''

flag = receiver(arrived_data, pattern)

''' Computing the error percentage by comparing the generated bits in the sender
and arrived to receiver '''

count = 0
if len(generated) == len(arrived_data):
    for i in range(len(generated)):
        if (generated[i] ^ arrived_data[i]) == 1: # Two bits are different to each
other
            count += 1
    print('The percentage error = ', "{:.2f}".format(count/len(generated) * 100), '%')

def crc_decode(data_input, pattern_input, receive):

```

```

''' Start of the program '''

# print('Read a stream of bits for "DATA":')
stream_data = data_input

data = []
for bit in stream_data:
    data.append(int(bit)) # Copy stream_data to data
    if(int(bit)!=0 and int(bit)!=1):
        print('Error Input!')
        exit() # Check Illegal Input

''' Data is read and put in data array now as integers '''

# print('Read a stream of bits for "PATTERN":')
stream_divisor = pattern_input
# Pattern is used as Divisor
pattern = []
for bit in stream_divisor:
    pattern.append(int(bit))
    if (int(bit) != 0 and int(bit) != 1):
        print('Error Input!')
        exit() # Check Illegal Input

''' Divisor is read and put in pattern array now as integers '''

remainder = division_process(data, pattern)
# Divide the data with pattern

print('The remainder:')
for i in range(len(remainder) - 1):
    print('remainder[' + str(i) + ']' + ' = ' + str(remainder[i]))

generated = []

# print('The generated code from CRC (Data will be sent from sender):')
for i in data:
    generated.append(i)
for i in range(len(remainder) - 1):
    generated.append(remainder[i])

# print(generated) # Sender

''' Receiver check the arrived data and decide if there is an error '''
arrived_data = [0 for j in range(len(data) + len(remainder) - 1)] # Receiver
# print('Read the data arrived to receiver to check if there is an error using CRC:')
received = receive
for k in range(len(received)):
    arrived_data[k] = int(received[k])

''' Data is being arrived to receiver '''

flag = receiver(arrived_data, pattern)

return flag
''' Computing the error percentage by comparing the generated bits in the sender

```

```

and arrived to receiver '''

count = 0
if len(generated) == len(arrived_data):
    for i in range(len(generated)):
        if (generated[i] ^ arrived_data[i]) == 1: # Two bits are different to each
other
            count += 1
    # print('The percentage error = ', "{:.2f}".format(count/len(generated) * 100), '%')

def run1():

    a = str(inp1.get())
    b = str(inp2.get())
    print(a)
    print(b)
    c = crc_code(a, b)
    print('c', c)
    str1 = "".join([str(x) for x in c]) + "\n"
    print(str1)
    print(c)
    txt.insert(END, str1) # 追加显示运算结果
    inp1.delete(0, END) # 清空输入
    inp2.delete(0, END) # 清空输入

def run2(x, y):
    a = str(inp1.get())
    b = str(inp2.get())
    print(a)
    print(b)
    c = crc_code(a, b)
    s = '%0.2f+%0.2f=%0.2f\n' % (a, b, a + b)
    txt.insert(END, s) # 追加显示运算结果
    inp1.delete(0, END) # 清空输入
    inp2.delete(0, END) # 清空输入

root = Tk()
root.geometry('460x240')
root.title('CRC Code')
lb1 = Label(root, text='请输入数据与生成多项式')
lb1.place(relx=0.1, rely=0.1, relwidth=0.8, relheight=0.1)
inp1 = Entry(root)
inp1.place(relx=0.1, rely=0.2, relwidth=0.3, relheight=0.1)
inp2 = Entry(root)
inp2.place(relx=0.6, rely=0.2, relwidth=0.3, relheight=0.1)

# 方法-直接调用 run1()
btn1 = Button(root, text='RUN', command=run1)
btn1.place(relx=0.1, rely=0.4, relwidth=0.3, relheight=0.1)

# 方法二利用 lambda 传参数调用 run2()
btn2 = Button(root, text='方法二', command=lambda: run2(inp1.get(), inp2.get()))
# btn2.place(relx=0.6, rely=0.4, relwidth=0.3, relheight=0.1)

```

在窗体垂直自上而下位置 60%处起，布局相对窗体高度 40%高的文本框

```
txt = Text(root)
txt.place(rely=0.6, relheight=0.4)

root.mainloop()
```

3. 代码执行结果

设待传输数据为 **10010110**，校验码为 **1011**

编码过程执行结果：

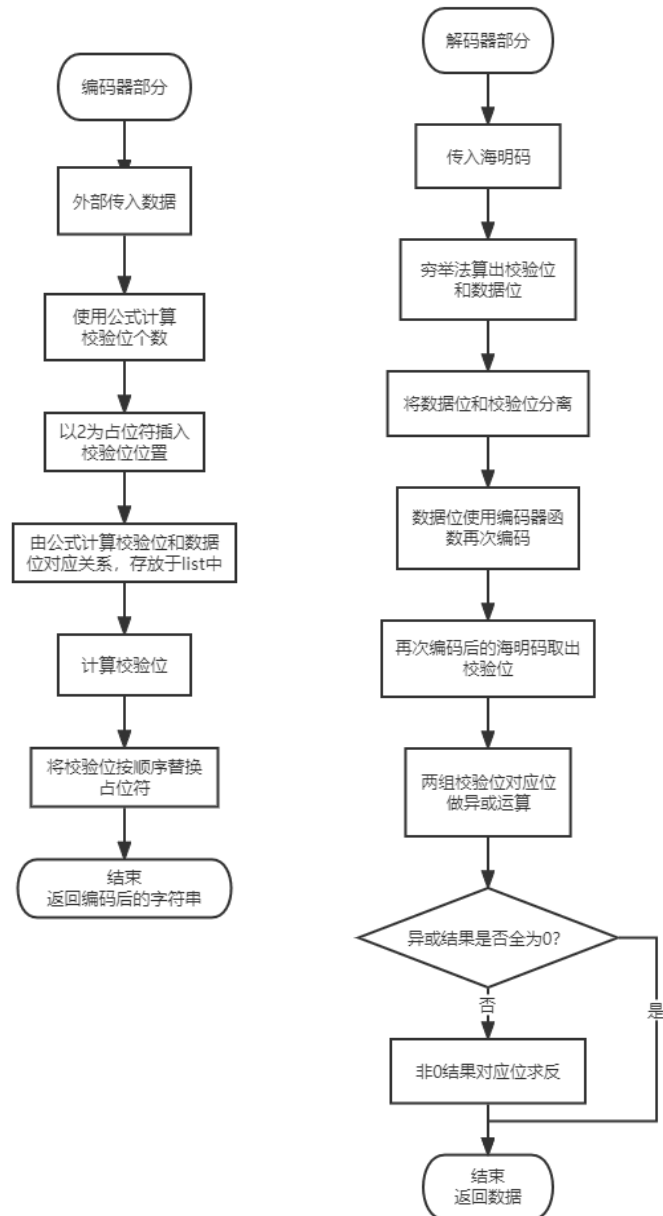
生成左移 3 位的原数据，值为 **[1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0]**，对校验码进行模 2 运算，得到三位余数后进行相加，得到结果 **[1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1]** 为 CRC 处理后的数据。

译码过程执行结果

将生成的数据值与校验码进行模 2 运算，若余数为 0 则说明无错误。

海明码

1. 流程图



2. 代码

```
def encoder(userin):
```

```
    listin = list(userin)
```

```
    listincheck = list(userin)
```

```
    # print(listin)
```

```
    # print(len(userin))
```

```
    #确定校验位个数
```

```
    for i in range(0,len(userin)):
```

```

        if len(userin) <= 2**i - i - 1:
            checkbit = i
            # print(checkbit)
            break
    checkbitlist = []
    #用 2 作为占位符 插入校验位位置
    for i in range(0,checkbit):
        a = 2
        checkbitlist.append(a)
    for i in range(0,checkbit):
        listin.insert(2**i-1,checkbitlist[i])

    # print(listin)
    ablist = []
    #确定校验位和数据位对应关系 即校验位构成
    for i in range(0,len(listin)):
        if listin[i] != 2:
            a = i+1
            ab = bin(a)[2:].zfill(checkbit)#checkbit 个验证位
            ab = list(ab)
            ab.reverse()
            ablist.append(ab)
            # ab.reverse()
    # print(ablist)
    #计算校验位
    for i in range(0,checkbit):
        signal = 0 # 信号位 用于标志第一个 1
        for j in range(0,len(ablist)):
            if ablist[j][i] == &apos;1&apos;;:
                if signal == 0:
                    checkbitlist[i] = int(listincheck[j])
                    signal = 1
                else:
                    # if ablist[j][i] == &apos;1&apos;; and signal == 1:
                    checkbitlist[i] = int(checkbitlist[i])^int(listincheck[j])
    # print(checkbitlist)
    #校验位插入对应位置
    for i in range(0,checkbit):
        listincheck.insert(2**i-1,checkbitlist[i])
    str1 = "".join([str(x) for x in listincheck ]) + "\n"
    # print(str1)

    return str1

```

```

def decoder(deuserin):
    deuserinlist = list(deuserin)

    def checknumber(k):
        r = 0
        for i in range(0, k):
            if k <= 2 ** i - i - 1:
                r = i
                break
        return r

    k = 1
    while k + checknumber(k) != len(deuserin): # 计算有效位数
        k += 1
    r = checknumber(k)
    # print(r) # 校验位个数
    # print(k)
    datalist = []
    checklist = []
    temp = 0
    for i in range(0, len(deuserin)):
        if i == 2**temp-1:
            checklist.append(deuserin[i])
            temp = temp+1
        else:
            datalist.append(deuserin[i])
    # print(checklist)
    # print(datalist)

    reencoder = encoder(datalist)
    # print(reencoder)
    redatalist = []
    rechecklist = []
    temp = 0
    for i in range(0, len( reencoder)):
        if i == 2 ** temp - 1:
            rechecklist.append( reencoder[i])
            temp = temp + 1
        else:
            redatalist.append(reencoder[i])
    # print(rechecklist)
    glist = []
    for i in range (0,r):
        glist.append(int(checklist[i])^int(rechecklist[i]))

```



```

# print(glist)
glist.reverse()
str1 = "0b" + "".join([str(x) for x in glist])
error = int(str1,2)
print(error)
if deuserinlist[error - 1] == "1":
    deuserinlist[error - 1] = 0
else:
    deuserinlist[error - 1] = 1
# print(deuserinlist)
str1 = "".join([str(x) for x in datalist]) + "\n"
print(str1)
if error == 0:
    errorx = "无误" + "\n"
else:
    errorx = str(error) + "\n"
return errorx, str1
# code = input("请输入数据:")
# output = encoder(code)
# print(output)
# deuserin = input("请输入海明码:")
#
# decoder(deuserin)

from tkinter import *

def isbix(string):
    return string.count('&apos;1&apos;') + string.count('&apos;0&apos;') == len(string)

def run1():
    a = str(inp1.get())
    # b = float(inp2.get())
    # s = '&apos;%.2f+%.2f=%.2f\n&apos;' % (a, b, a + b)
    b = encoder(a)
    print(a)
    txt.insert(END, b) # 追加显示运算结果
    inp1.delete(0, END) # 清空输入
    inp2.delete(0, END) # 清空输入

def run2(x, y):
    # a = float(x)
    # b = float(y)
    # s = '&apos;%.2f+%.2f=%.2f\n&apos;' % (a, b, a + b)
    deuserin = str(inp2.get())

```

```

s,y = decoder(deuserin)
txt.insert(END, y)
txt.insert(END, s)    # 追加显示运算结果
inp1.delete(0, END)   # 清空输入
inp2.delete(0, END)   # 清空输入

root = Tk()
root.geometry('460x240')
root.title('海明编码')

lb1 = Label(root, text='请输入数据和海明编码')
lb1.place(relx=0.1, rely=0.1, relwidth=0.8, relheight=0.1)
inp1 = Entry(root)
inp1.place(relx=0.1, rely=0.2, relwidth=0.3, relheight=0.1)
inp2 = Entry(root)
inp2.place(relx=0.6, rely=0.2, relwidth=0.3, relheight=0.1)

# 方法-直接调用 run1()
btn1 = Button(root, text='编码', command=run1)
btn1.place(relx=0.1, rely=0.4, relwidth=0.3, relheight=0.1)

# 方法二利用 lambda 传参数调用 run2()
btn2 = Button(root, text='译码',
               command=lambda: run2(inp1.get(), inp2.get()))
btn2.place(relx=0.6, rely=0.4, relwidth=0.3, relheight=0.1)

# 在窗体垂直自上而下位置 60% 处起，布局相对窗体高度 40%高的文本框
txt = Text(root)
txt.place(rely=0.6, relheight=0.4)

root.mainloop()

```

3. 代码执行结果

$n = 4$, $k = 3$ 时，求 1010 的海明码，编码得到 1010 对应的海明码为 1010010。译码时 $S_1 = P_1 \oplus D_1 \oplus D_2 \oplus D_4 = 0$ ； S_2 , S_3 类似……，若 $S_3S_2S_1 = 000$ 那么说明没出错；否则 $S_3S_2S_1$ 值为几就是那一位出错，例如 $S_3S_2S_1 = 001$ ，则说明第一位出错，直接将该出错位取反以纠错。

实验过程中遇到的问题如何解决的？	
------------------	--

问题 1: CRC 编码和海明编码的主要区别有哪些?

CRC 只能进行检错但是海明码可以进行纠错

问题 2: 海明编码能纠多位错误吗?

不能

问题 3: CRC 编码的检错能力与生成多项式最高幂次数值有关系吗?

有关

实验三

姓名		院系	人工智能与自动化学院	学号	
实验地点	科技楼 12 楼		实验日期	2022 年 5 月 3 日	

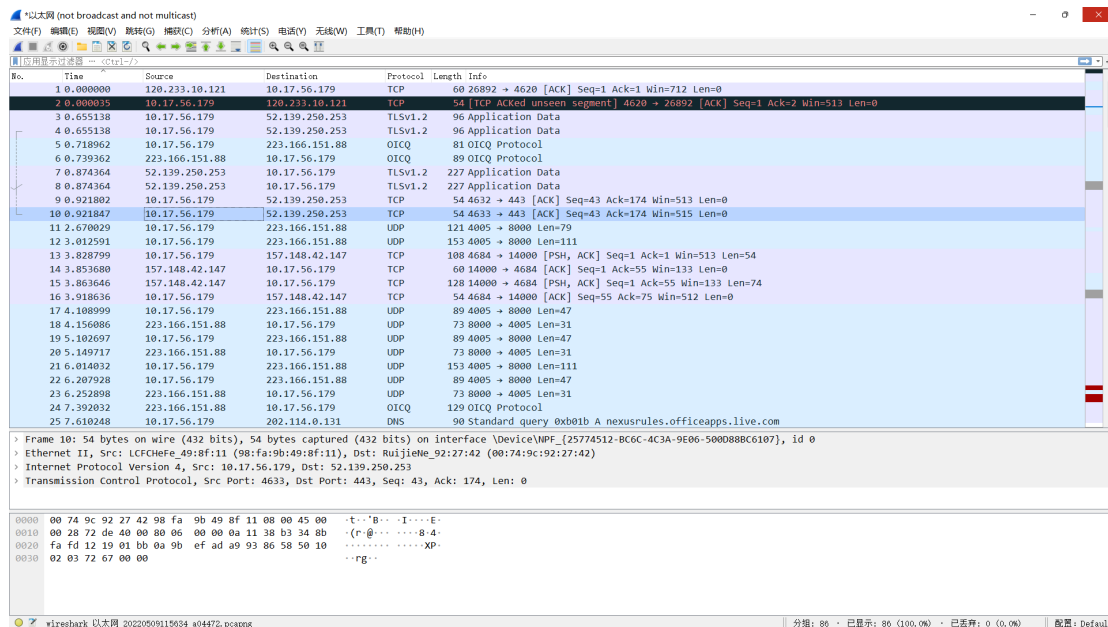
实验目的：

- 1、掌握抓包软件 Wireshark 的使用。
- 2、了解并掌握因特网中 MAC 帧、IP 数据报、TCP/UDP 段的字段及其含义。
- 3、了解并掌握三次握手、HTTP 协议的内容和功能。

实验内容：

以太网数据链路层协议分析实验

- 1、在 windows 中安装 Wireshark 软件。
- 2、配置包捕获模式为混杂模式，捕获网络中所有机器的数据包，当捕获到一定数量的数据包后，停止捕获，观察捕获到的数据包，并对照解析结果和原始数据包的具体字段，了解 MAC 地址字段、协议类型、数据来源等。



- 3、配置包捕获过滤器，只捕获特定 IP 地址、特定端口或特定类型的包，然后重新开始捕获包并分析。

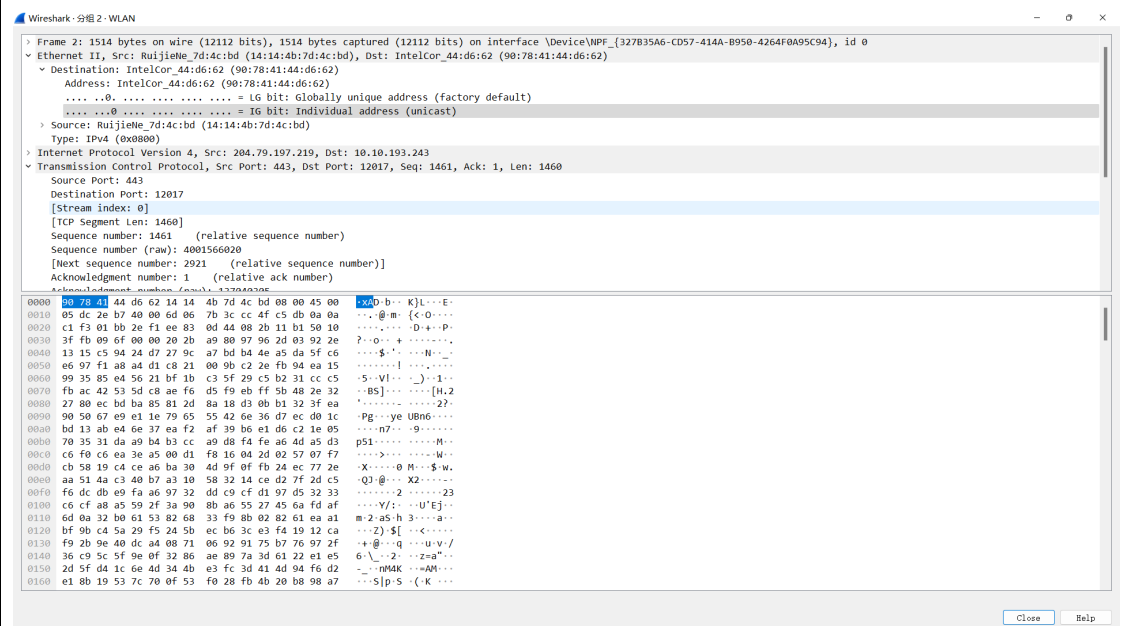
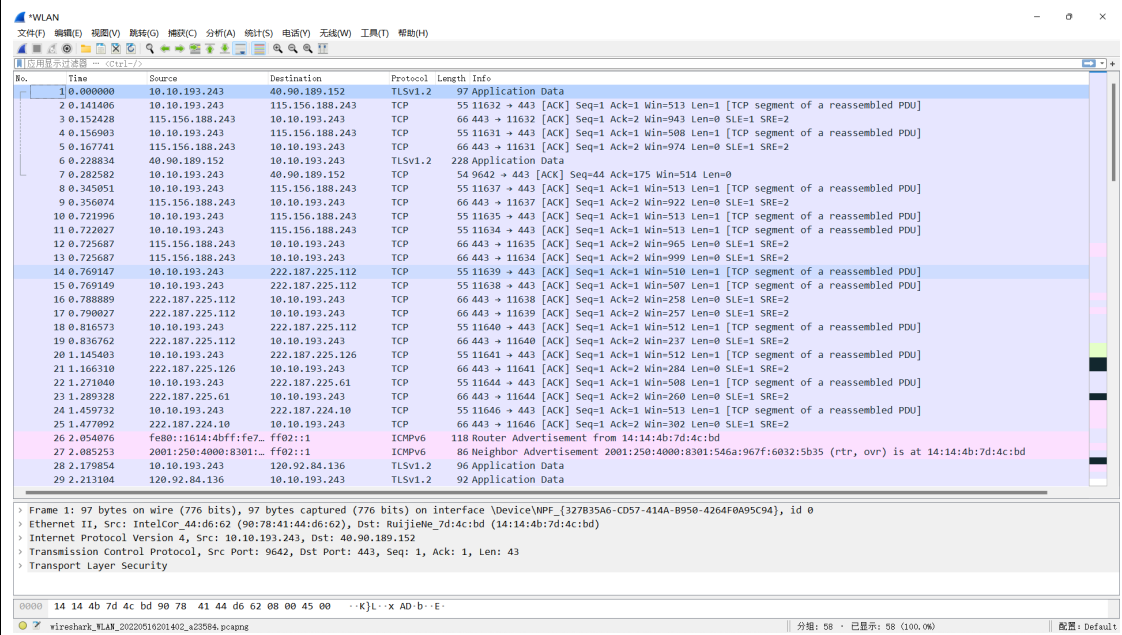
4. 以太网帧分析

（a）捕捉任何主机发出的 Ethernet 802.3 格式的帧（帧的长度字段 ≤ 1500 ），Wireshark 的 capture filter 的 filter string 设置为：ether[12:2] ≤ 1500 。观察并分析帧结构，802.3 格式的帧的上一层主要是哪些 PDU？是 IP、LLC 还是其它哪种？为什么？

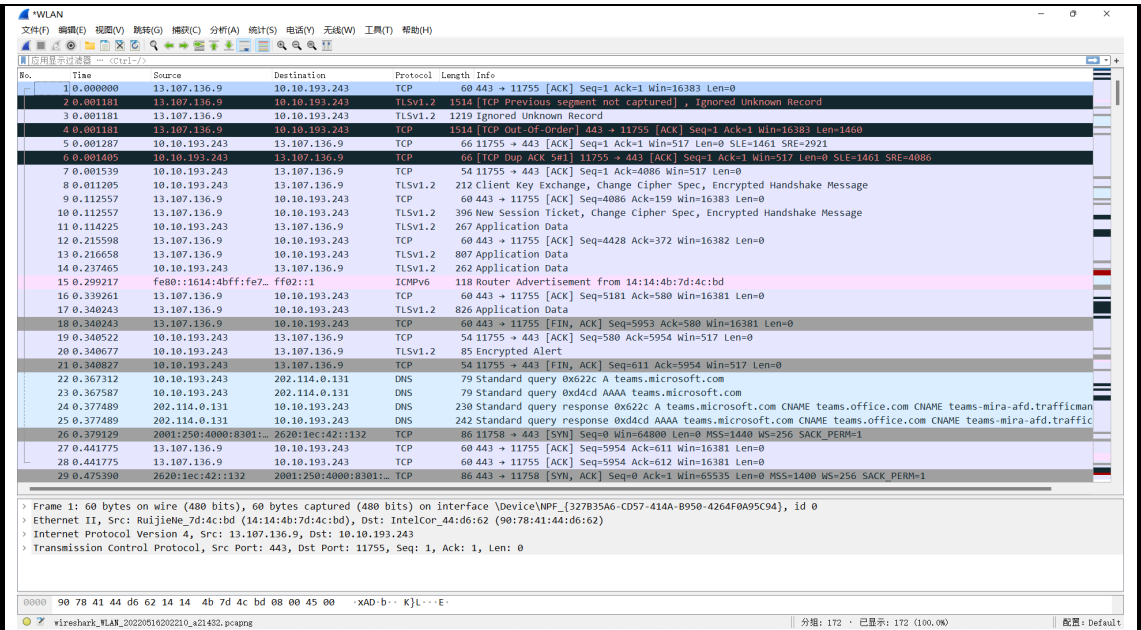
（b）捕捉任何主机发出的 DIX Ethernet V2（即 Ethernet II）格式的帧（帧的长度字段 > 1500 ，帧的长度字段实际上是类型字段），Wireshark 的 capture filter 的 filter string 设置为：ether[12:2] > 1500 。观察并分析帧结构，Ethernet II 的帧的上一层主要是哪些 PDU？是 IP、

LLC 还是其它哪种？为什么？

帧的上一层协议是 IP



5. 捕捉并分析局域网上的所有 ethernet broadcast 广播帧，Wireshark 的 capture filter 的 filter string 设置为: ether broadcast



(1). 观察并分析哪些主机在发广播帧，这些帧的高层协议是什么？

例如主机 10.10.193.243，广播帧的最高协议为 ARP 和 NBNS

(2). 1 分钟内有几个广播帧？有否发生广播风暴？

LAN 共享网段上连接了约 355 台主机，一分钟内平均 800 条广播帧，没有发生广播风暴。

6.思考问题：

(1) 本地数据存放的字节顺序和网络包中的字节顺序是否相同？

答：不一定相同的，根据处理器的不同，在计算机中存储数据的字节次序也是不同

(2) 怎样知道哪些数据包是 MAC 广播包或 IP 子网广播包？

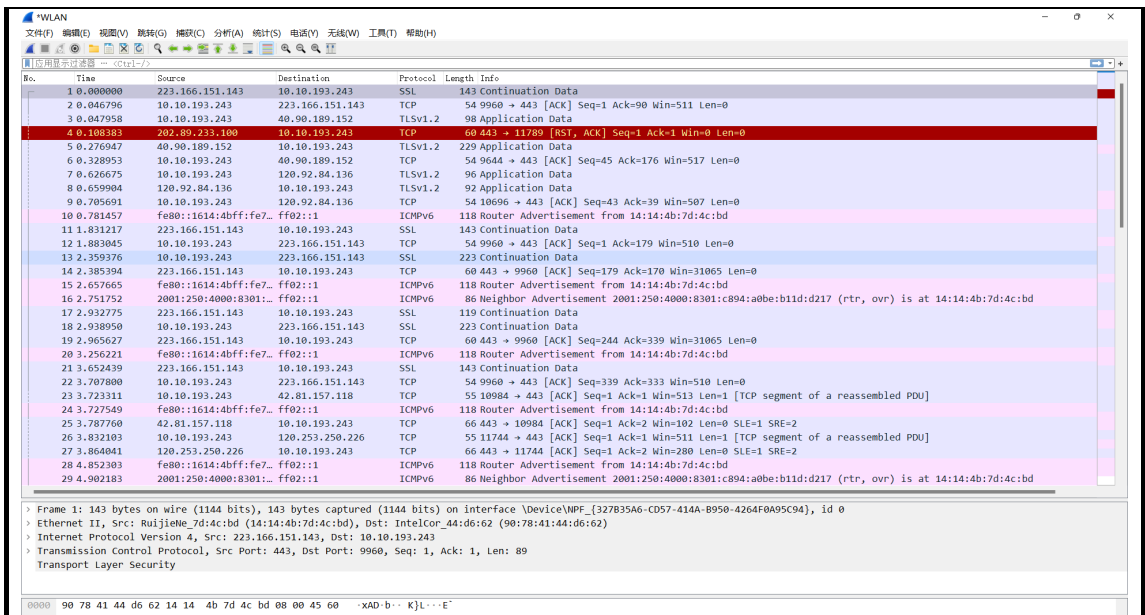
答：使用软件抓包进行分析，MAC 广播包含 IP 子网广播包不是一个层次上的概念。通过交换机后，MAC 广播的包将无法获取。

(3) 通过包捕获软件能否捕获到通过交换机连接的计算机发出的包？能够捕捉到其他计算机发出的哪些包？

答：底层抓包软件可以捕获到网卡所收到的所有包，包括目的地址不是本机的包，在同一个 HUB 下网络中能够收到局域网中所有计算机发出的包。

网络层、传输层协议分析实验

1. 捕捉局域网上本机（假设为主机 10.14.26.53）发出或接受的所有 ARP 包，Wireshark 的 capture filter 的 filter string 设置为：arp host 10.14.26.53.

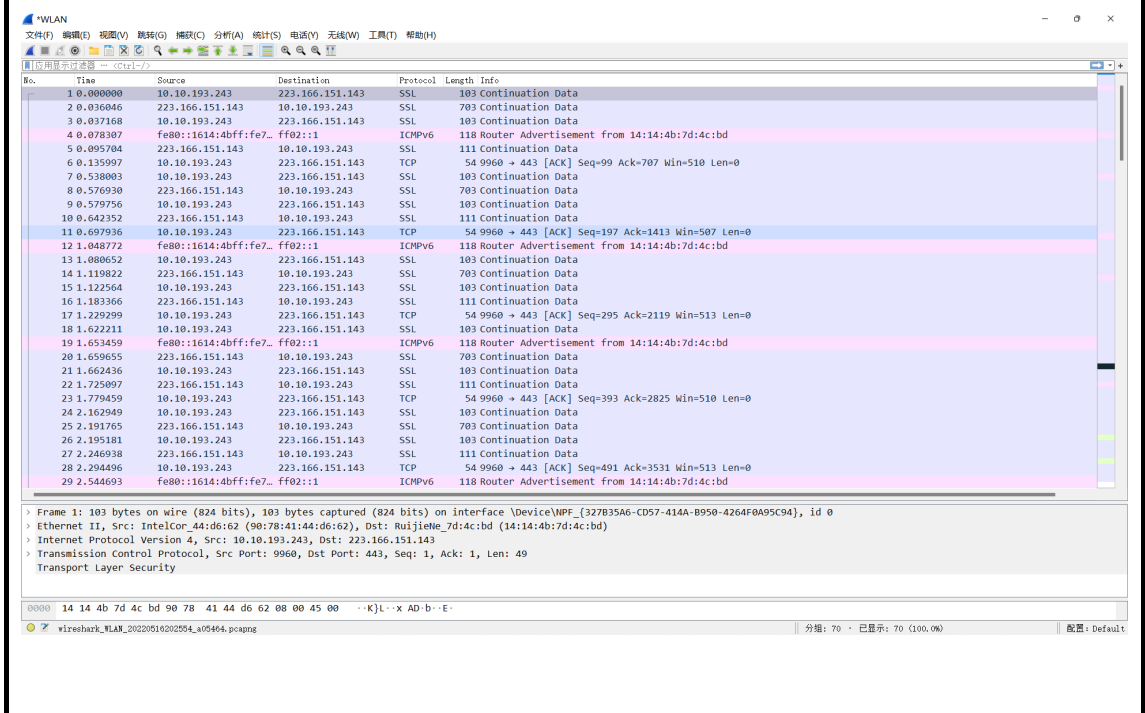


- (1).主机 10.14.26.53 上执行 ”arp -d *” 清除 arp 缓存。
- (2).观察并分析主机 10.14.26.53 发出或接受的所有 ARP 包，及 arp 包结构。
2. 捕捉局域网上的所有 IP 广播包，Wireshark 的 capture filter 的 filter string 设置为: ip broadcast。观察并分析哪些节点在发广播包，这些包的高层协议是什么？

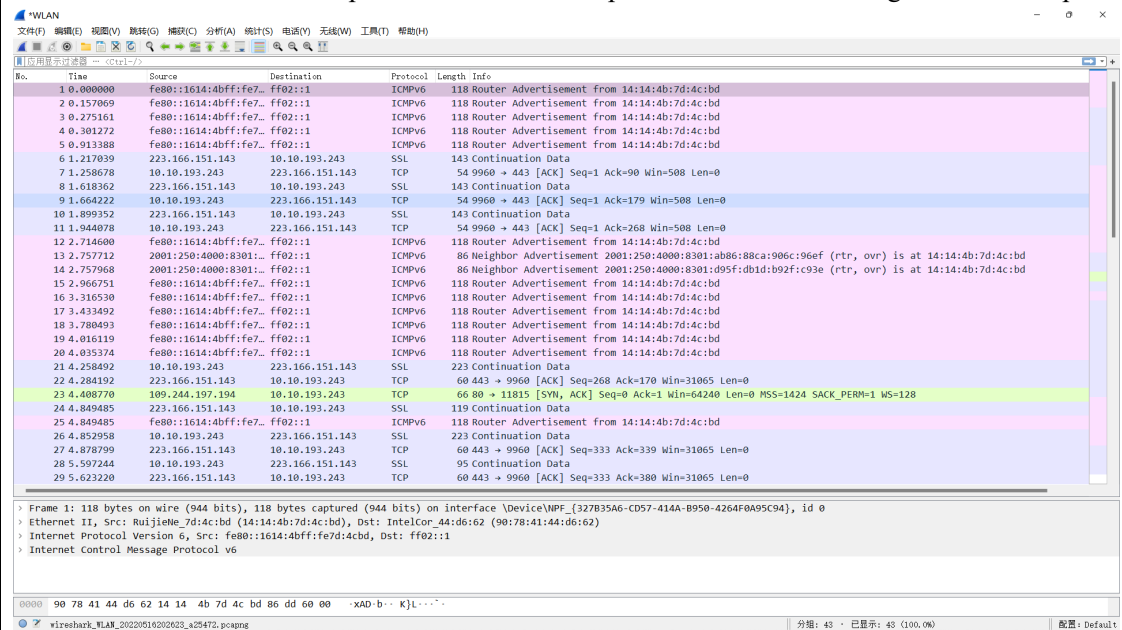
高层协议：TCP

3. 捕捉局域网上的所有 IP 组播包，Wireshark 的 capture filter 的 filter string 设置为: ip multicast。观察并分析哪些节点在发组播包，这些包的高层协议是什么？

高层协议：ARP LLC STP IP



4. 捕捉局域网上的所有 icmp 包，Wireshark 的 capture filter 的 filter string 设置为: icmp

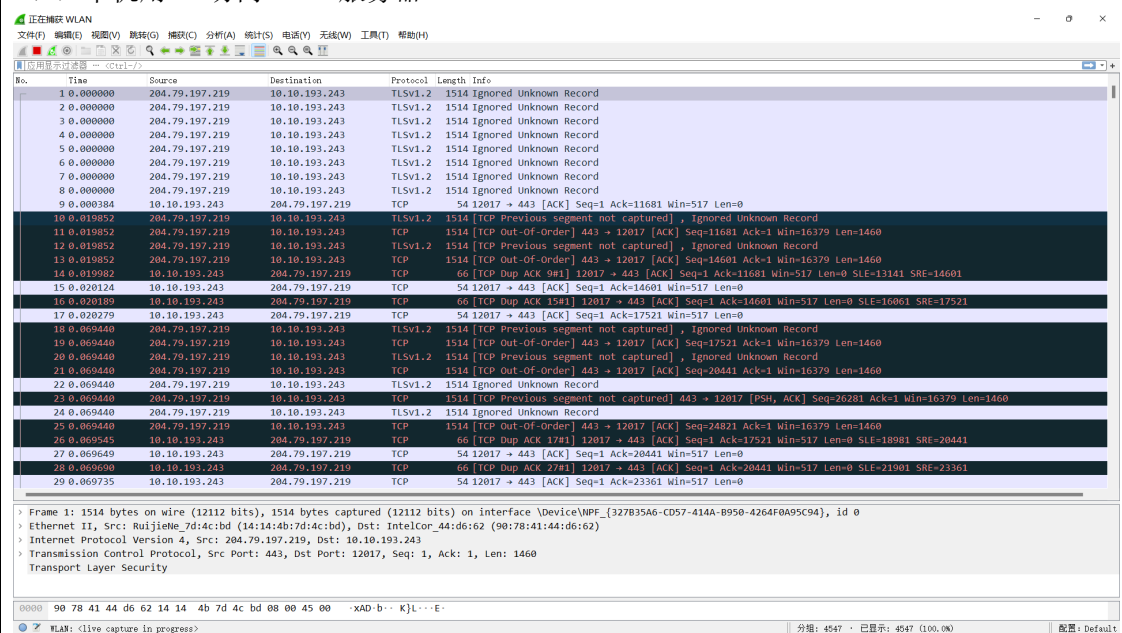


- (1). 在主机 10.14.26.53 上 ping 局域网上的另一主机（例如 10.14.26.54）。
 - (2). 观察并分析主机 10.14.26.53 发出或接受的所有 icmp 包，及 icmp 包的类型和结构。
- 5、捕捉任意一个 IP 包，分析该包中的：IP 头中各个字段作用及其数值，解释其含义。
- 6、捕捉任意一个 TCP 包，分析该包中的：TCP 头中各个字段作用及其数值，解释其含义。
- 7、捕捉任意一个 UDP 包，分析该包中的：UDP 头中各个字段作用及其数值，解释其含义。

四、应用层协议分析实验

捕捉本机和某 www 服务器（如 www.hust.edu.cn）之间的通信，Wireshark 的 capture filter 的 filter string 设置为: host 本机 ip 地址 and www.hust.edu.cn。（本机 ip 地址需要替换为本机真正的 ip 地址，如 219.219.54.111，以下同）

（1）本机用 IE 访问 www 服务器 www.hust.edu.cn。



（2）观察并分析本机和 www 服务器之间传输的 IP 数据报结构，TCP segment 结构，HTTP 报文的结构。

(3) 观察并分析本机和 www 服务器之间建立 TCP 连接时的三次握手过程。

三次握手过程:

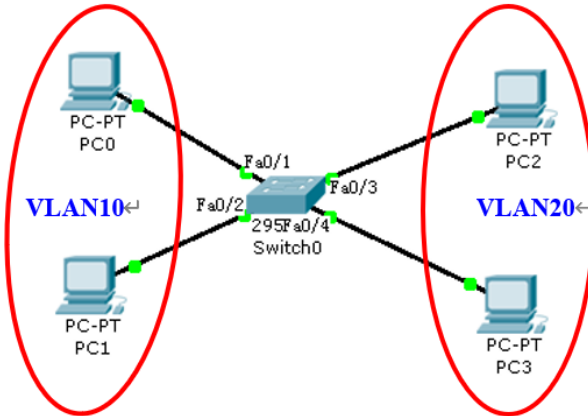
204.79.197.219 -> 10.10.193.243 SYN(Seq = 0)

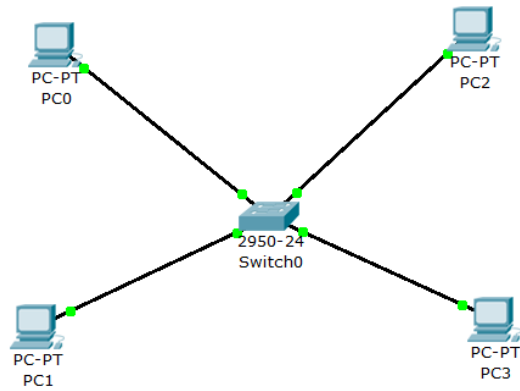
10.10.193.243 -> 204.79.197.219 SYN(Seq = 0, Ack = 1)

204.79.197.219 -> 10.10.193.243 SYN(Seq = 1, Ack = 1)

实验过程中遇到的问题如何解决的?

实验四

姓名		院系	人工智能与自动化学院	学号	
实验地点	科技楼 12 楼		实验日期	2022 年 5 月 3 日	
实验目的：					
1. 掌握交换机的工作原理、在网络中的作用及交换机设备选型。 2. 掌握路由器的工作原理、在网络中的作用及交换机设备选型。 3. 掌握交换机及路由器的基本配置方法，了解各配置命令的作用。					
实验内容：					
1. 单交换机 VLAN 配置实验					
请按下图连接好线缆，并配置好计算机的 IP 地址，所有的子网掩码均为 24 位掩码。实验原理如下：					
将 PC0 和 PC1 设为 VLAN10，PC2 和 PC3 设为 VLAN20。划分 VLAN 之前，四台计算机之间都可以相互通信，即能够 ping 通。划分 VLAN 之后，只有同一个 VLAN 中的计算机能够通信（即能 ping 通），不同 VLAN 之间的计算机不能通信（即不能 ping 通）。					
					
单交换机虚拟局域网 VLAN 配置实验组网图					
实验步骤如下：					
（a）在 Packet Tracer5 软件中，画好网络拓扑图，给四台计算机分别配置好 IP 地址。各计算机的 IP 地址配置如下表：					
VLAN	计算机	IP 地址	子网掩码		
VLAN10	PC0	192.168.1.11	255.255.255.0		
	PC1	192.168.1.12	255.255.255.0		
VLAN20	PC2	192.168.1.13	255.255.255.0		
	PC3	192.168.1.14	255.255.255.0		



(b) 在四台计算机上分别使用 ping 命令，确认它们之间全部能够相互通信。

```
PC>ping 192.168.1.12

Pinging 192.168.1.12 with 32 bytes of data:

Reply from 192.168.1.12: bytes=32 time=142ms TTL=128
Reply from 192.168.1.12: bytes=32 time=50ms TTL=128
Reply from 192.168.1.12: bytes=32 time=77ms TTL=128
Reply from 192.168.1.12: bytes=32 time=81ms TTL=128

Ping statistics for 192.168.1.12:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 50ms, Maximum = 142ms, Average = 87ms
```

对 PC0 使用 ping 命令测试 PC0 与 PC1 是否能通信

测试后 PC0, PC1, PC2, PC3 都可以互相通信

(c) 将 PC0 和 PC1 设为 VLAN10，PC2 和 PC3 设为 VLAN20。

(d) 测试同一个 VLAN 之间的计算机能否通信（即能否 ping 通），不同的 VLAN 之间的计算机能否通信（即能否 ping 通）。

答：1. 同一个 VLAN 之间的计算机能通信 2. 不同的 VLAN 之间的计算机不能通信

用 PC0 分别 ping PC1 和 PC3

```

PC>ping 192.168.1.12

Pinging 192.168.1.12 with 32 bytes of data:

Reply from 192.168.1.12: bytes=32 time=142ms TTL=128
Reply from 192.168.1.12: bytes=32 time=50ms TTL=128
Reply from 192.168.1.12: bytes=32 time=77ms TTL=128
Reply from 192.168.1.12: bytes=32 time=81ms TTL=128

Ping statistics for 192.168.1.12:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 50ms, Maximum = 142ms, Average = 87ms

PC>ping 192.168.1.13

Pinging 192.168.1.13 with 32 bytes of data:

Request timed out.
Request timed out.
Request timed out.
Request timed out.

Ping statistics for 192.168.1.13:
    Packets: Sent = 4, Received = 0, Lost = 4 (100% loss),

```

结果：PC0 和 PC1 能 ping 通，PC0 和 PC3 不能 ping 通

用 PC3 分别 ping PC1 和 PC2

```

PC>ping 192.168.1.12

Pinging 192.168.1.12 with 32 bytes of data:

Request timed out.
Request timed out.
Request timed out.
Request timed out.

Ping statistics for 192.168.1.12:
    Packets: Sent = 4, Received = 0, Lost = 4 (100% loss),

PC>ping 192.168.1.13

Pinging 192.168.1.13 with 32 bytes of data:

Reply from 192.168.1.13: bytes=32 time=117ms TTL=128
Reply from 192.168.1.13: bytes=32 time=60ms TTL=128
Reply from 192.168.1.13: bytes=32 time=67ms TTL=128
Reply from 192.168.1.13: bytes=32 time=66ms TTL=128

Ping statistics for 192.168.1.13:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 60ms, Maximum = 117ms, Average = 77ms

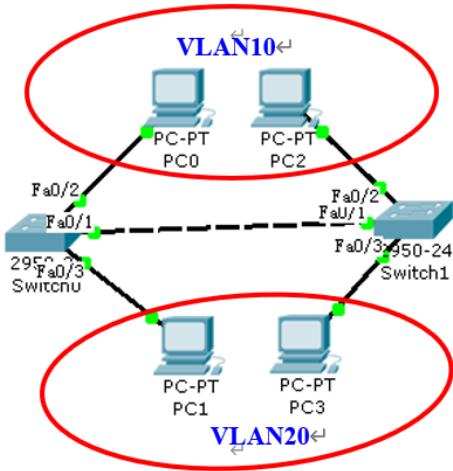
```

结果：PC3 和 PC2 能 ping 通，PC3 和 PC1 不能 ping 通

2. 跨交换机 VLAN 配置实验

请按下图连接好线缆，并配置好计算机的 IP 地址，所有的子网掩码均为 24 位掩码。实验原理如下：

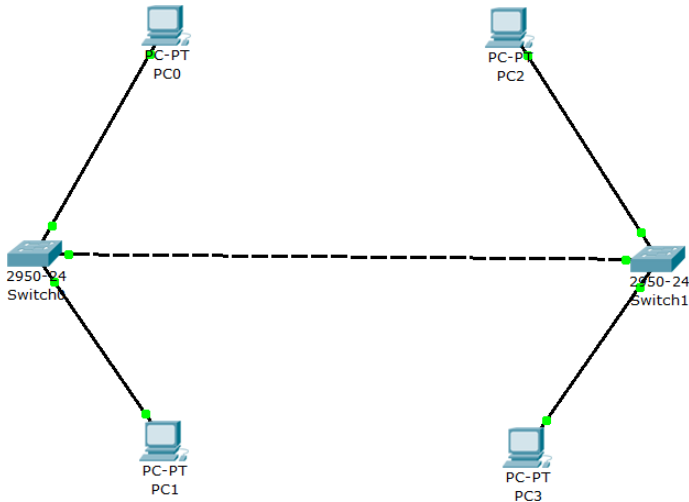
两台交换机之间使用 f0/1 端口相连，交换机 switch1 连接 PC0 和 PC1，交换机 switch2 连接 PC2 和 PC3。将 PC0 和 PC2 设为 VLAN10，PC1 和 PC3 设为 VLAN20。划分 VLAN 之前，四台计算机之间都可以相互通信，即能够 ping 通。划分 VLAN 之后，只有同一个 VLAN 中的计算机能够通信（即能 ping 通），不同 VLAN 之间的计算机不能通信（即不能 ping 通）。跨交换机划分 VLAN 的实验拓扑图如下：



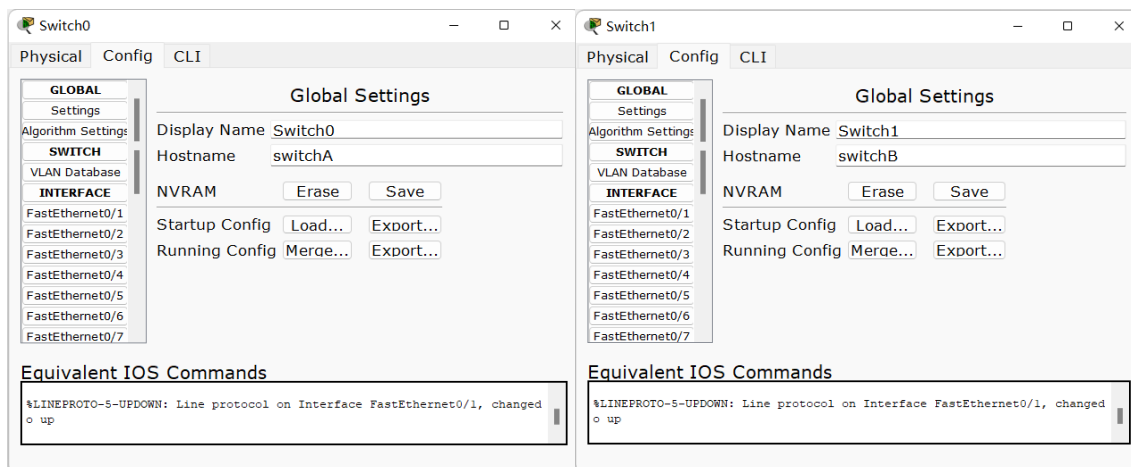
实验步骤如下：

（a）在 Packet Tracer5 软件中，画好网络拓扑图，给四台计算机分别配置好 IP 地址。各计算机的 IP 地址配置如下表：

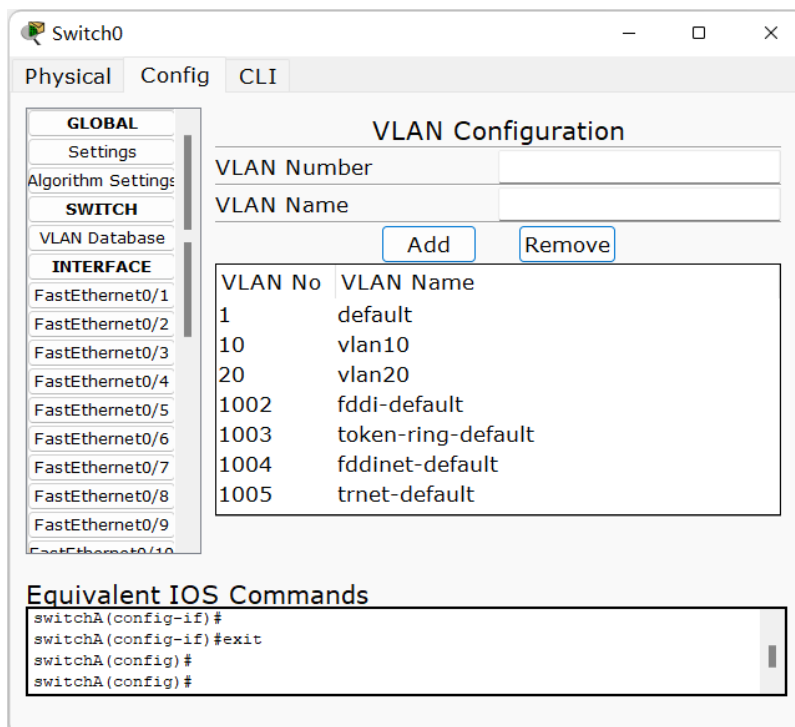
VLAN	计算机	IP 地址	子网掩码
VLAN10	PC0	192.168.1.11	255.255.255.0
	PC1	192.168.1.12	255.255.255.0
VLAN20	PC2	192.168.1.13	255.255.255.0
	PC3	192.168.1.14	255.255.255.0



- (b) 在四台计算机上分别使用 ping 命令，确认它们之间全部能够相互通信。
- (c) 将两台交换机改名称 **switchA**, **switchB**



- (d) 在交换机 **switchA** 上划分 vlan 10 和 vlan 20



- (e) 在交换机 **switchA** 上将 f0/2 端口加入到 vlan 10 中，将 f0/3 端口加入到 vlan 20 中

- (f) 在交换机 **switchB** 上划分 vlan 10 和 vlan 20

Switch1

PhysicalConfigCLI

GLOBAL

Settings

Algorithm Settings

SWITCH

VLAN Database

INTERFACE

FastEthernet0/1

FastEthernet0/2

FastEthernet0/3

FastEthernet0/4

FastEthernet0/5

FastEthernet0/6

FastEthernet0/7

VLAN Configuration

VLAN Number

VLAN Name

AddRemove

VLAN No	VLAN Name
1	default
10	vlan10
20	vlan20
1002	fddi-default
1003	token-ring-default

Equivalent IOS Commands

switchB>enable
switchB#configure terminal
Enter configuration commands, one per line. End with CNTL/Z.
switchB(config)#

(g) 在交换机 **switchB** 上将 f0/2 端口加入到 vlan 10 中，将 f0/3 端口加入到 vlan 20 中

此时，将 vlan 10 中的两台计算机相互 ping，结果 ping 不通，**请思考为什么**。将 vlan 20 中的两台计算机相互 ping，结果也是 ping 不通。

答：两台电脑不在同一个交换机中，且交换机没有配置 trunk 端口，故同一 vlan 的计算机 ping 不通。

(h) 将两台交换机之间的链路设置为 trunk 模式

(i) 测试同一个 VLAN 之间的计算机能否通信（即能否 ping 通），不同的 VLAN 之间的计算机能否通信（即能否 ping 通）。

答：同一个 VLAN 之间的计算机**能通信**，不同的 VLAN 之间的计算机**不能通信**。

```

PC>ping 192.168.1.13

Pinging 192.168.1.13 with 32 bytes of data:

Reply from 192.168.1.13: bytes=32 time=253ms TTL=128
Reply from 192.168.1.13: bytes=32 time=123ms TTL=128
Reply from 192.168.1.13: bytes=32 time=87ms TTL=128
Reply from 192.168.1.13: bytes=32 time=98ms TTL=128

Ping statistics for 192.168.1.13:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 87ms, Maximum = 253ms, Average = 140ms

PC>ping 192.168.1.12

Pinging 192.168.1.12 with 32 bytes of data:

Request timed out.
Request timed out.
Request timed out.
Request timed out.

Ping statistics for 192.168.1.12:
    Packets: Sent = 4, Received = 0, Lost = 4 (100% loss),

```

对 PC0 使用 ping 指令测试和 PC1, PC2 能否通信, 结论: PC0 和 PC1 不能通信, PC0 和 PC2 能通信。

3. 路由器设备使用实验

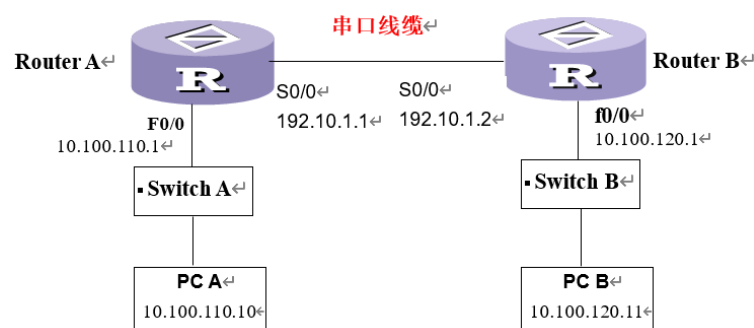
1. 实验设备及器材: Cisco1841 路由器、Windows 操作系统的 PC 机、Packet Tracer5 软件

2. 实验原理

路由器在没有配置路由时, 只能实现与它直连的网络间的通信, 为了实现在更大范围的网络间通信, 需要进行路由配置, 路由包括静态路由、默认路由和动态路由几类。

静态路由是一种特殊的路由, 它由管理员手工配置而成。网管必需了解路由器的拓扑连接, 通过手工方式指定路由路径。但这种配置问题在于: 当一个网络故障发生后, 静态路由不会自动发生改变, 必须有网管手工修改路由路径。

本次实验中, 网络拓扑结构及其相关参数如下图:

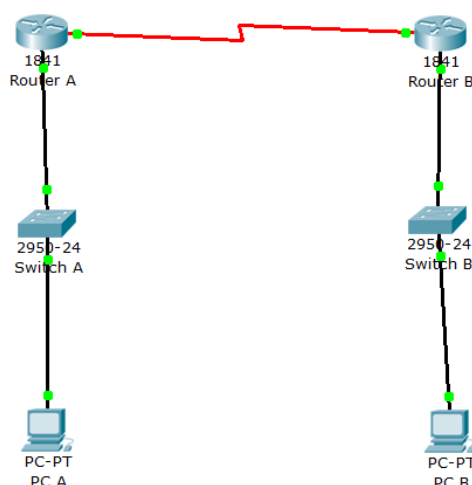


3. 实验内容

(a) 在 Packet Tracer5 软件中, 画好网络拓扑图。

配置好计算机 A 的 IP 地址 10.100.110.10、子网掩码 255.255.255.0 和默认网关 10.100.110.1, 配置

好计算机 B 的 IP 地址 10.100.120.11、子网掩码 255.255.255.0 和默认网关 10.100.120.1。



(b) 配置路由器 Router A

配置路由器 A 的接口 FastEthernet0/0 和 Serial0/1/0 的 IP 地址

配置路由器 A 的静态路由

Router A

Physical Config CLI

GLOBAL Settings

Algorithm Settings:

ROUTING

Static

RIP

SWITCHING

VLAN Database

INTERFACE

FastEthernet0/0

FastEthernet0/1

Serial0/1/0

Serial0/1/1

Port Status ☒ On

Bandwidth ☒ Auto

☐ 10 Mbps ☒ 100 Mbps

Duplex ☒ Auto

☒ Full Duplex ☐ Half Duplex

MAC Address 0002.179E.2A01

IP Address 10.100.110.1

Subnet Mask 255.255.255.0

Tx Ring Limit 10

Equivalent IOS Commands

```
RouterA(config-if)#
RouterA(config-if)#exit
RouterA(config)#interface FastEthernet0/0
RouterA(config-if)#
```

Router A

Physical Config CLI

GLOBAL Settings

Algorithm Settings:

ROUTING

Static

RIP

SWITCHING

VLAN Database

INTERFACE

FastEthernet0/0

FastEthernet0/1

Serial0/1/0

Serial0/1/1

Port Status ☒ On

Clock Rate 800000

Duplex ☒ Full Duplex

IP Address 192.10.1.1

Subnet Mask 255.255.255.0

Tx Ring Limit 10

Equivalent IOS Commands

```
RouterA(config-if)#
RouterA(config-if)#exit
RouterA(config)#interface Serial0/1/0
RouterA(config-if)#
```

在 PCA 上分别 ping RouterA 的 f0/0 接口和 s0/0 接口。若 ping 通，说明配置成功。

例如，此时在 PCA 上 ping 它的网关 10.100.110.1，结果应该显示 ping 通，如下所示：

```
Command Prompt

Packet Tracer PC Command Line 1.0
PC>ping 10.100.110.1

Pinging 10.100.110.1 with 32 bytes of data:

Reply from 10.100.110.1: bytes=32 time=117ms TTL=255
Reply from 10.100.110.1: bytes=32 time=64ms TTL=255
Reply from 10.100.110.1: bytes=32 time=48ms TTL=255
Reply from 10.100.110.1: bytes=32 time=84ms TTL=255

Ping statistics for 10.100.110.1:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 48ms, Maximum = 117ms, Average = 78ms

PC>
```

RouterA 的路由表如下所示

```
Router A
Physical Config CLI
IOS Command Line Interface

%SYS-5-CONFIG_I: Configured from console by console
RouterA#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

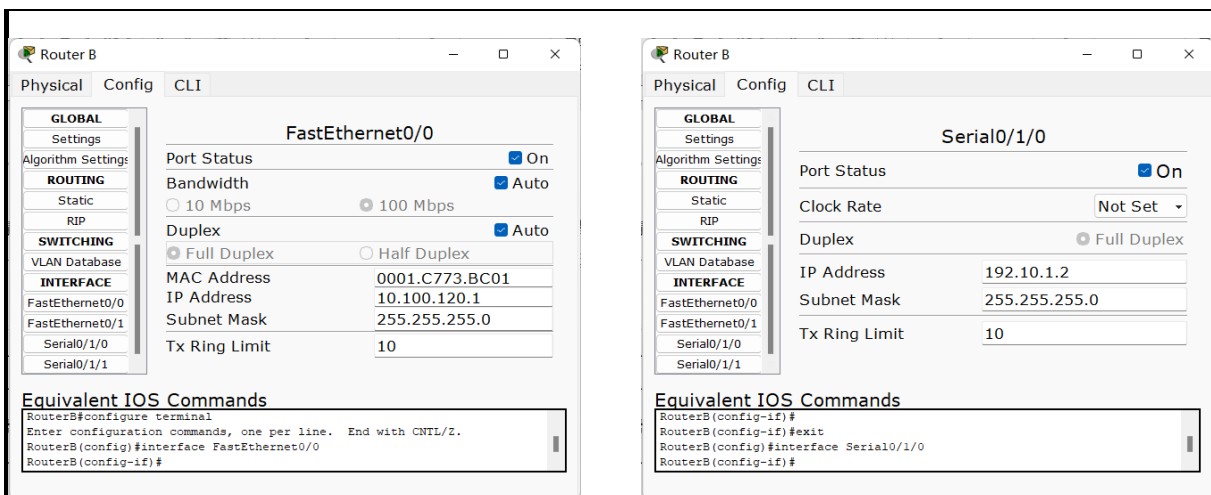
Gateway of last resort is not set

    10.0.0.0/24 is subnetted, 2 subnets
C       10.100.110.0 is directly connected, FastEthernet0/0
S       10.100.120.0 [1/0] via 192.10.1.2
C       192.10.1.0/24 is directly connected, Serial0/1/0
RouterA#
RouterA#
RouterA#
RouterA#
```

(c) 配置路由器 Router B

配置路由器 B 的接口 FastEthernet0/0 和 Serial0/1/0 的 IP 地址

配置路由器 B 的静态路由



在 PCB 上分别 ping RouterB 的 f0/0 接口和 s0/0 接口。若 ping 通，说明配置成功。

例如，此时在 PCB 上 ping 它的网关 10.100.120.1，结果应该显示 ping 通，如下所示：

```

Command Prompt
Packet Tracer PC Command Line 1.0
PC>ping 10.100.120.1

Pinging 10.100.120.1 with 32 bytes of data:

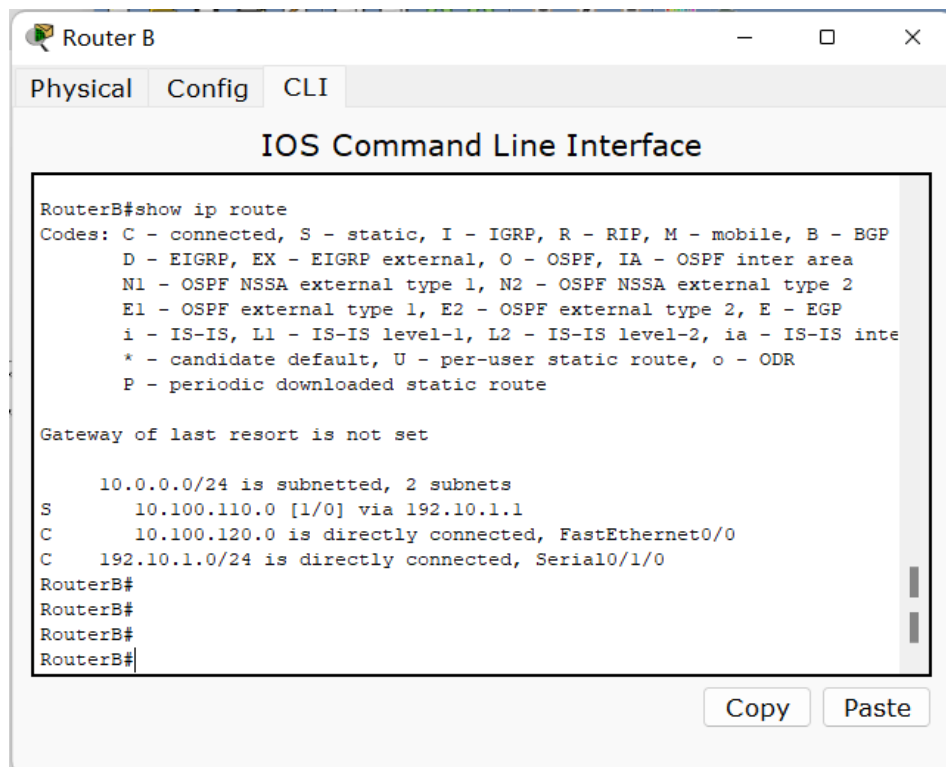
Reply from 10.100.120.1: bytes=32 time=160ms TTL=255
Reply from 10.100.120.1: bytes=32 time=60ms TTL=255
Reply from 10.100.120.1: bytes=32 time=63ms TTL=255
Reply from 10.100.120.1: bytes=32 time=51ms TTL=255

Ping statistics for 10.100.120.1:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 51ms, Maximum = 160ms, Average = 83ms

PC>

```

RouterA 的路由表如下所示



The screenshot shows a window titled "Router B" with tabs for "Physical", "Config", and "CLI". The "CLI" tab is active, displaying the "IOS Command Line Interface". The command "RouterB#show ip route" has been entered, and the output is displayed. The output includes a legend for route codes, a message about the gateway of last resort, and a list of routes. The routes shown are:

```
RouterB#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 inte
       * - candidate default, U - per-user static route, o - ODR
       P - periodic downloaded static route

Gateway of last resort is not set

10.0.0.0/24 is subnetted, 2 subnets
S    10.100.110.0 [1/0] via 192.10.1.1
C    10.100.120.0 is directly connected, FastEthernet0/0
C    192.10.1.0/24 is directly connected, Serial0/1/0
RouterB#
RouterB#
RouterB#
RouterB#
```

At the bottom right of the CLI window, there are "Copy" and "Paste" buttons.

(d) 在计算机 A 和计算机 B 上分别测试网络的连通性：

在计算机 A 上 ping 计算机 B，在计算机 B 上 ping 计算机 A，检查静态路由配置是否成功。若不成功，请说明可能出现的原因与解决方法。

小提示：

测试网络连通性时可以采用分段测试法，这样方便查找问题所在。例如，在计算机 A 上 ping 计算机 B 时，可以通过以下步骤来测试：

- 1) 在 PCA 上 ping 它的网关 10.100.110.1，若 ping 通，继续；
- 2) 在 PCA 上 ping RouterA 的外网端口 192.10.1.1，若 ping 通，继续；
- 3) 在 PCA 上 ping RouterB 的外网端口 192.10.1.2，若 ping 通，继续；
- 4) 在 PCA 上 ping RouterB 的内网端口 10.100.120.1，若 ping 通，继续；
- 5) 在 PCA 上 ping PCB 的 ip 地址 10.100.120.11，若 ping 通，说明静态路由配置成功。

Command Prompt

```
PC>ping 10.100.110.1

Pinging 10.100.110.1 with 32 bytes of data:

Reply from 10.100.110.1: bytes=32 time=82ms TTL=255
Reply from 10.100.110.1: bytes=32 time=81ms TTL=255
Reply from 10.100.110.1: bytes=32 time=43ms TTL=255
Reply from 10.100.110.1: bytes=32 time=79ms TTL=255

Ping statistics for 10.100.110.1:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 43ms, Maximum = 82ms, Average = 71ms

PC>ping 192.10.1.1

Pinging 192.10.1.1 with 32 bytes of data:

Reply from 192.10.1.1: bytes=32 time=67ms TTL=255
Reply from 192.10.1.1: bytes=32 time=76ms TTL=255
Reply from 192.10.1.1: bytes=32 time=63ms TTL=255
Reply from 192.10.1.1: bytes=32 time=51ms TTL=255

Ping statistics for 192.10.1.1:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 51ms, Maximum = 76ms, Average = 64ms

PC>ping 10.100.120.1

Pinging 10.100.120.1 with 32 bytes of data:

Reply from 10.100.120.1: bytes=32 time=112ms TTL=254
Reply from 10.100.120.1: bytes=32 time=58ms TTL=254
Reply from 10.100.120.1: bytes=32 time=107ms TTL=254
Reply from 10.100.120.1: bytes=32 time=106ms TTL=254

Ping statistics for 10.100.120.1:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 58ms, Maximum = 112ms, Average = 95ms

PC>ping 10.100.120.11

Pinging 10.100.120.11 with 32 bytes of data:

Reply from 10.100.120.11: bytes=32 time=171ms TTL=126
Reply from 10.100.120.11: bytes=32 time=112ms TTL=126
Reply from 10.100.120.11: bytes=32 time=145ms TTL=126
Reply from 10.100.120.11: bytes=32 time=155ms TTL=126

Ping statistics for 10.100.120.11:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 112ms, Maximum = 171ms, Average = 145ms
```

计算机 A 的静态路由配置成功。

Command Prompt

```
PC>ping 192.10.1.2

Pinging 192.10.1.2 with 32 bytes of data:

Reply from 192.10.1.2: bytes=32 time=55ms TTL=255
Reply from 192.10.1.2: bytes=32 time=72ms TTL=255
Reply from 192.10.1.2: bytes=32 time=55ms TTL=255
Reply from 192.10.1.2: bytes=32 time=62ms TTL=255

Ping statistics for 192.10.1.2:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 55ms, Maximum = 72ms, Average = 61ms

PC>ping 192.10.1.1

Pinging 192.10.1.1 with 32 bytes of data:

Reply from 192.10.1.1: bytes=32 time=86ms TTL=254
Reply from 192.10.1.1: bytes=32 time=126ms TTL=254
Reply from 192.10.1.1: bytes=32 time=62ms TTL=254
Reply from 192.10.1.1: bytes=32 time=81ms TTL=254

Ping statistics for 192.10.1.1:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 62ms, Maximum = 126ms, Average = 88ms

PC>ping 10.100.110.1

Pinging 10.100.110.1 with 32 bytes of data:

Reply from 10.100.110.1: bytes=32 time=77ms TTL=254
Reply from 10.100.110.1: bytes=32 time=99ms TTL=254
Reply from 10.100.110.1: bytes=32 time=111ms TTL=254
Reply from 10.100.110.1: bytes=32 time=100ms TTL=254

Ping statistics for 10.100.110.1:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 77ms, Maximum = 111ms, Average = 96ms

PC>ping 10.100.110.10

Pinging 10.100.110.10 with 32 bytes of data:

Reply from 10.100.110.10: bytes=32 time=139ms TTL=126
Reply from 10.100.110.10: bytes=32 time=136ms TTL=126
Reply from 10.100.110.10: bytes=32 time=184ms TTL=126
Reply from 10.100.110.10: bytes=32 time=142ms TTL=126

Ping statistics for 10.100.110.10:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 136ms, Maximum = 184ms, Average = 150ms
```

使用同样的方法用计算机 B ping 计算机 A，得到结果为全部连通，故计算机 B 的静态路由配置成功。

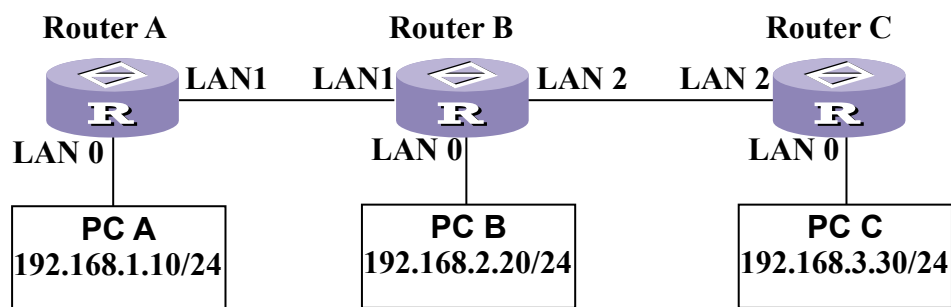
4. 综合实验

请根据以下网络拓扑图和相关参数，对相应路由器进行静态路由的配置，使得网络连通。

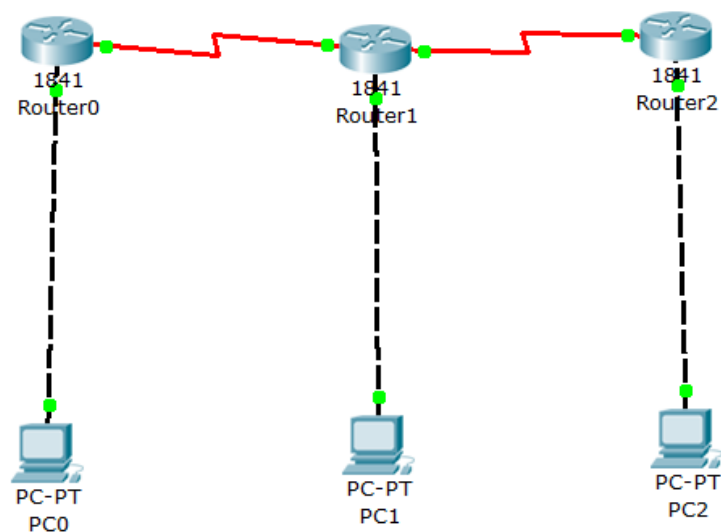
Router A LAN0: 192.168.1.1/24 LAN1: 10.10.10.1/24

Router B LAN0: 192.168.2.1/24 LAN1: 10.10.10.2/24 LAN2: 10.10.20.1/24

Router C LAN0: 192.168.3.1/24 LAN1: 10.10.20.2/24



(a) 在 Packet Tracer5 软件中，画好网络拓扑图。



配置好计算机 A 的 IP 地址 192.168.1.10、子网掩码 255.255.255.0 和默认网关 192.168.1.1，配置好计算机 B 的 IP 地址 192.168.2.20、子网掩码 255.255.255.0 和默认网关 192.168.2.1，配置好计算机 C 的 IP 地址 192.168.3.30、子网掩码 255.255.255.0 和默认网关 192.168.3.1，

(b) 配置路由器 Router A

配置路由器 A 的接口 FastEthernet0/0 和 Serial0/1/0 的 IP 地址

配置路由器 A 的静态路由

Router0

Physical Config CLI

GLOBAL

Settings

Algorithm Settings:

ROUTING

Static

RIP

SWITCHING

VLAN Database

INTERFACE

FastEthernet0/0

FastEthernet0/1

Serial0/1/0

FastEthernet0/0

Port Status ☒ On

Bandwidth ☒ Auto

☐ 10 Mbps ☒ 100 Mbps

Duplex ☒ Auto

☒ Full Duplex ☐ Half Duplex

MAC Address 0090.0C28.1201

IP Address 192.168.1.1

Subnet Mask 255.255.255.0

Tx Ring Limit 10

Equivalent IOS Commands

Router#configure terminal

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

Router(config)#interface FastEthernet0/0

Router(config-if)#

Router0

Physical Config CLI

GLOBAL

Settings

Algorithm Settings:

ROUTING

Static

RIP

SWITCHING

VLAN Database

INTERFACE

FastEthernet0/0

FastEthernet0/1

Serial0/1/0

Serial0/1/0

Port Status ☒ On

Clock Rate 800000

Duplex ☒ Full Duplex

IP Address 10.10.10.1

Subnet Mask 255.255.255.0

Tx Ring Limit 10

Equivalent IOS Commands

Router(config-if)#

Router(config-if)#exit

Router(config)#interface Serial0/1/0

Router(config-if)#

Router0

Physical Config CLI

GLOBAL

Settings

Algorithm Settings:

ROUTING

Static

RIP

SWITCHING

VLAN Database

INTERFACE

FastEthernet0/0

FastEthernet0/1

Serial0/1/0

Static Routes

Network

Mask

Next Hop

Add

Network Address

192.168.2.0/24 via 10.10.10.2

192.168.3.0/24 via 10.10.10.2

10.10.20.0/24 via 10.10.10.2

Remove

Equivalent IOS Commands

Router(config)#interface Serial0/1/0

Router(config-if)#

Router(config-if)#exit

Router(config)#

RouterA 的路由表设置为如上图所示，next hop 为 RouterB 的 LAN1 地址，目的 IP 为 RouterB 和 RouterC 的子网 IP

(c) 配置路由器 RouterB

配置路由器 B 的接口 FastEthernet0/0 和 Serial0/1/0 和 Serial0/0/0 的 IP 地址

配置路由器 B 的静态路由

Router1

Physical Config CLI

GLOBAL Settings

Algorithm Settings

ROUTING

Static

RIP

SWITCHING

VLAN Database

INTERFACE

FastEthernet0/0

FastEthernet0/1

Serial0/0/0

Serial0/1/0

FastEthernet0/0

Port Status ☒ On

Bandwidth ☐ 10 Mbps ☒ 100 Mbps

Duplex ☒ Full Duplex ☐ Half Duplex

MAC Address 0002.1785.D001

IP Address 192.168.2.1

Subnet Mask 255.255.255.0

Tx Ring Limit 10

Equivalent IOS Commands

```
Router#configure terminal
Enter configuration commands, one per line. End with CNTL/Z.
Router(config)#interface FastEthernet0/0
Router(config-if)#
```

Router1

Physical Config CLI

GLOBAL Settings

Algorithm Settings

ROUTING

Static

RIP

SWITCHING

VLAN Database

INTERFACE

FastEthernet0/0

FastEthernet0/1

Serial0/0/0

Serial0/1/0

Serial0/0/0

Port Status ☒ On

Clock Rate 800000

Duplex ☒ Full Duplex

IP Address 10.10.20.1

Subnet Mask 255.255.255.0

Tx Ring Limit 10

Equivalent IOS Commands

```
Router(config-if)#
Router(config-if)#exit
Router(config)#interface Serial0/0/0
Router(config-if)#
```

Router1

Physical Config CLI

GLOBAL Settings

Algorithm Settings

ROUTING

Static

RIP

SWITCHING

VLAN Database

INTERFACE

FastEthernet0/0

FastEthernet0/1

Serial0/0/0

Serial0/1/0

Serial0/1/0

Port Status ☒ On

Clock Rate 800000

Duplex ☒ Full Duplex

IP Address 10.10.10.2

Subnet Mask 255.255.255.0

Tx Ring Limit 10

Equivalent IOS Commands

```
Router(config-if)#
Router(config-if)#exit
Router(config)#interface Serial0/1/0
Router(config-if)#
```

Router1

Physical Config CLI

GLOBAL Settings

Algorithm Settings

ROUTING

Static

RIP

SWITCHING

VLAN Database

INTERFACE

FastEthernet0/0

FastEthernet0/1

Serial0/0/0

Serial0/1/0

Static Routes

Network

Mask

Next Hop

Add

Network Address

192.168.1.0/24 via 10.10.10.1

192.168.3.0/24 via 10.10.20.2

Remove

Equivalent IOS Commands

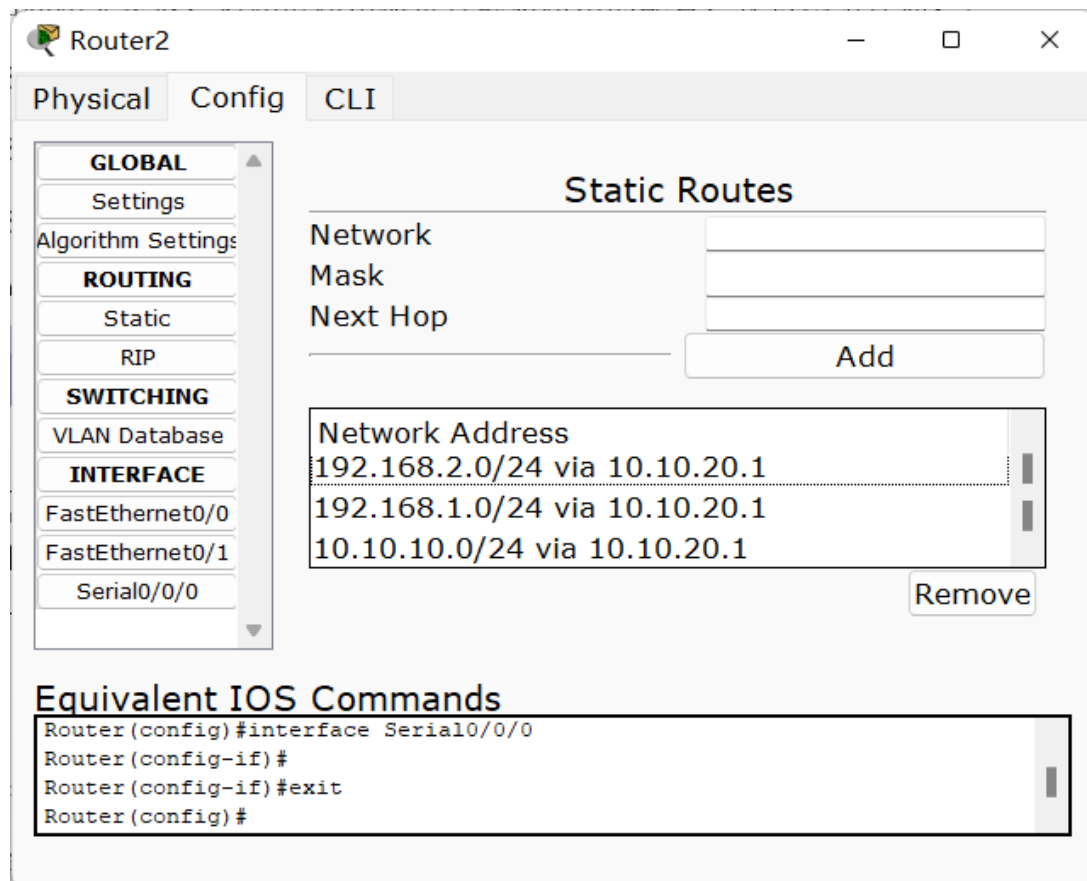
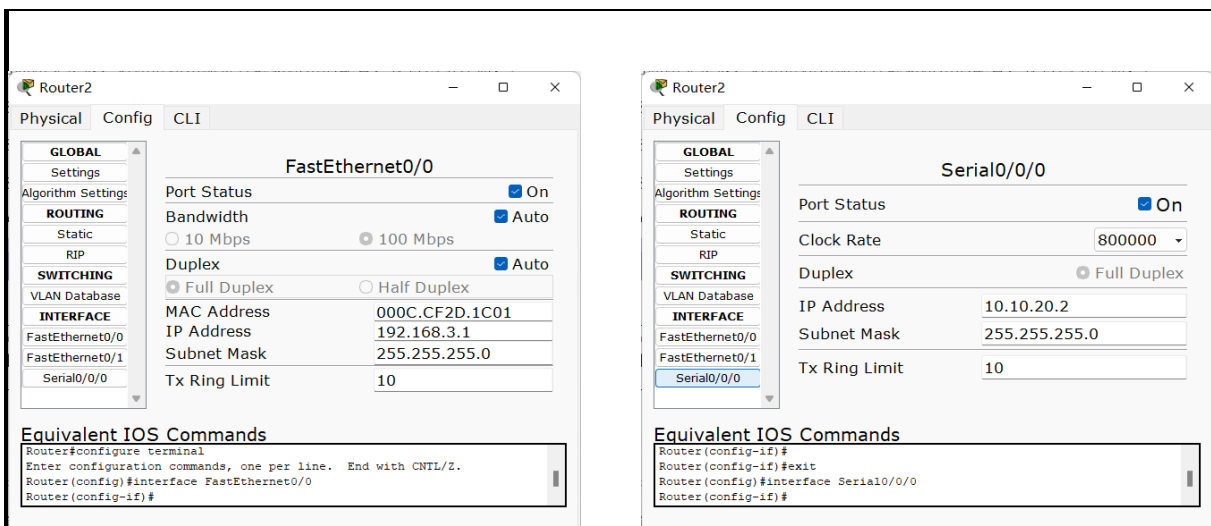
```
Router(config)#interface Serial0/1/0
Router(config-if)#
Router(config-if)#exit
Router(config)#
```

Router B 的路由表设置为如下图所示，next hop 为 RouterA 和 RouterC 的 LAN1 地址，目的 IP 为 RouterA 和 RouterC 的子网 IP

(d) 配置路由器 Router C

配置路由器 C 的接口 FastEthernet0/0 和 Serial0/1/0 的 IP 地址

配置路由器 C 的静态路由



Router C 的路由表设置为如上图所示, next hop 为 Router C 的 LAN1 地址, 目的 IP 为 Router A 和 Router B 的子网 IP

(d) 在计算机 A 和计算机 B 和计算机 C 上分别测试网络的连通性

经测试, 在计算机 A 和计算机 B 和计算机 C 分别 ping 不同路由器的 LAN0, LAN1 和其他计算机均能连通, 说明网络配置成功。

实验过程中遇到的问题如何解决的？	
<p>问题 1：路由器之间连线不知道用哪一种 Connections。</p> <p>解决：使用了 automatically choose connection type 自动选择连接类型，实验证明网络可以连通。</p> <p>问题 2：在路由器配置 Config 界面和 CLI 界面输入路由表及网络配置是否有相同的功能？</p> <p>解决：经实验发现两者实现的功能相同，且在 Config 界面更方便</p> <p>问题 3：综合实验在设置 Router B 时怎么同时配置两个端口？</p> <p>解决：在设置路由器时选择两个 Serial 串口并分别进行配置。</p>	

实验的体会（结论）	
<p>通过本次实验对计算机网络的知识有了更深的了解，包括基本的查看更改本机网络配置的指令，数据传输过程中的纠错差错方法，网络数据包的类型以及如何配置路由器，交换机等网络设备。对计算机网络的整体架构有了新的了解。</p>	