目录

[9.1]异步串行传输实验

课程名称	姓名	学号	专业	实验名称	实验日期
计算机网络实验			软件工程	异步串行实验	2019-10-30

实验目的

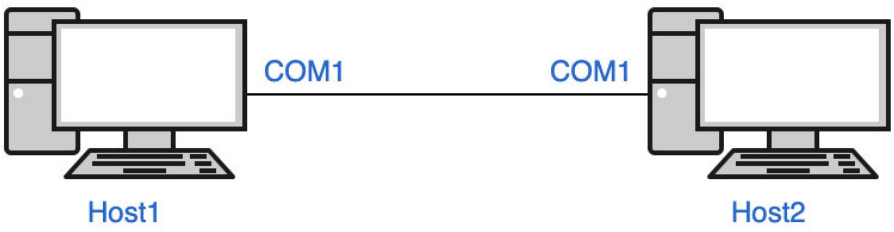
本实验是有关OSI模型物理层的实验，通过串口的字符收发，加深对通信基本原理的理解。

1. 理解通信基本原理；
2. 理解异步串行传输的基本概念；
3. 了解掌握RS－232接口标准以及RS-232帧格式；
4. 了解通信双方参数匹配是正常通信的基本保证，了解波特率等主要通信参数含义。

实验设备

1. 实验环境主要由两台带COM口的计算机，一根串行交叉线组成；
2. 将单根串行交叉线中间层组成。将单根串行交叉线将两个计算机的COM串口对接起来；
3. 两台计算机超级终端将作为路由器管理的操作平台

实验网络拓扑



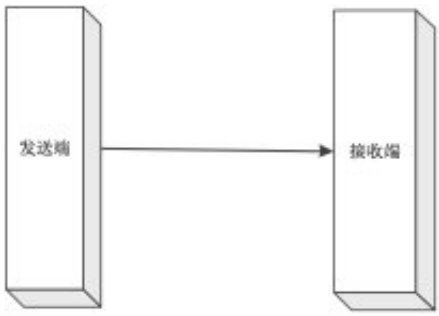
异步串行通信实验拓扑结构如上图所示，实验环境主要由两台带COM口的计算机和一根串行交叉线组成。将单根串行交叉线将两个计算机的COM串口对接起来；两台计算机超级终端将作为路由器管理的操作平台。

实验原理

> 一、异步串行通信原理

1. 串行方式：任何通信用户单独只能占用一条通信线路。

2. 串行传输：串行传输结构示意图如下图所示，每次只能传输一个数位，0或1。



3. 串行通信：在数据通信中，将传送的每个字符的二进制代码按照顺序依次发送的方式称为串行通信。串行通信方式广泛应用在远程通信中。
4. 串行传输模式在实际运用中，又分为同步传输模式和异步传输模式。
    - 同步传输模式：要求通信的收发双方在时间基准上保持一致。
    - 异步传输模式：接发两端使用不固定的时间模式，传输前也不需要协调，只要直接发送，不需要对方作为应答，但必须在传输两个数据包之间加入间隔符号，作为分隔。
5. 常用的串行通信接口标准包括RS-232、RS-449、V.24、V.35等。其中，RS-232是最常用的串行通信标准之一。

二、RS232接口

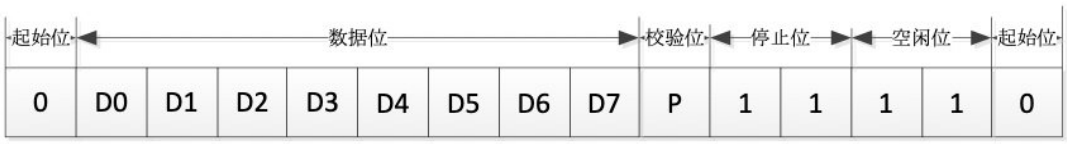
1. RS232接口是1970年由美国电子工业协会（EIA）联合贝尔系统、调制解调器厂家及计算机终端生产厂家共同制定的用于串行通讯的标准。
2. 主要内容：定义数据终端设备DTE（data terminal equipment）和数据通信设备DCE（data circuit equipment）之间的接口标准。
3. RS-232传输对象是字符，每次传输一个字符，计算机内部所有的数据都使用二进制，需要规定二进制数字表示哪个符号，称编码标准，现在我们字符使用的标准是ASCII编码。

三、ASCII码表

1. ASCII(American Standard Code for Information Interchange)，美国信息交换标准代码,是基于拉丁字母的一套电脑编码系统，主要用于显示现代英语和其他西欧语言，使用一个字（byte）来表示英语字母，比如，字母a,十进制编码值是97，二进制是0110 0001，一般用十六进制来表示，0x61，0x是16进制的前缀，ASCII编码详见附件。

四、S232数据桢格式

1. RS232在传输一个字符时，将字符转换成ASCII码，并组织成RS232数据桢进行传输
2. RS232数据桢格式：
    - 起始位为0，以一位低电平开始，表示发送端开始发送一帧数据；
    - 数据位，字符的ASCII码，一般低位在前，高位在后，数据位长度一般为8位；
    - 校验位，用于校验数据的正确性，使用奇校验或者偶校验；
    - 停止位为1，一帧信息已经发送完毕，长度一般为1~2位；
    - 空闲位：空闲位也为1，用于通知接受端等待下一个数据帧的传输；一旦开始传输下一个字符，发出新的开始位。
    - RS232数据桢格式如下图所示：



> 五、波特率

1. **波特率**：也称调制速率，指单位时间内通信信号变化的次数，它同时间模式中的时间间隔长度刚好成反比关系。
2. 波特率决定着每个数位的传输速率，显然，涉及到接发两端的两个设备配置，为保证两者在发送信号和检测信号时的时间同步，就必须设置成相同的波特率。
3. 发送设备和接收设备采用相同波特率称之为波特率匹配。如果接发双方baud rate波特率不匹配就会发生错误, 称之为帧错误。

实验内容

> 一、实验概述

使用超级终端进行两台计算机之间的串行通信，一端从键盘输入字符发送，另一端观看接收到的内容。主要实验内容：

1. 测试相同通信参数下的通信，进行相互间的字符接发送；
2. 测试不同波特率条件下的通信，其他参数保持相同，进行相互间的字符接发送；通过相互传输字符，来检测通信参数的匹配。

> 二、实验步骤

【准备过程】

1. 按照实验拓扑结构要求将两台计算机的COM口用串口反接连接线连接起来。
2. Host1：运行超级终端，建立串口通信连接test1，设置缺省通信参数。
3. Host2：运行超级终端，建立串口通信连接test2，设置缺省通信参数。

【实验一】：测试在相同通信参数下的字符传输

1. Host2的超级终端输入如下内容，此时Host2超级终端无内容显示

```
hhh2
```

2. Host1的超级终端接收到如下字符串

```
hhh2
```

【实验二】：测试采用不同波特率下的字符传输

1. Host1和Host2均断开连接，在“文件->属性->配置”中修改参数如下：

连接名	波特率	数据位	奇偶校验	停止位
test1	4800	8	奇校验	1
test2	9600	8	奇校验	1

2. 进行通信测试。首先在Host1终端从键盘键入以下字符串，Host1屏幕上并不会显示输入内容，Host2屏幕上显示乱码。

```
12 35 46 77
```

3. 在Host2终端从键盘键入以下字符串，Host2屏幕上并不会显示输入内容，Host1屏幕上显示不同乱码。

```
12 35 46 77
```

实验问题与结果分析

1. 实验1中，两台计算机的通信参数相同，波特率匹配，不会发生帧错误，因此字符传输正常。
2. 实验2中，两台计算机的波特率不同，则它们单位时间内通信信号发生变化的次数不同，相应的检测信号的间隔也不一致，因此两台计算机显示出乱码且乱码值不同。
3. 这个实验比较简单，是后面实验的基础，也让我初步掌握了一些使用超级终端的基本方法。通过动手操作串口的字符接收和发送，加深了对通信基本原理的理解。在理解通信基本原理和异步串行传输的基础上，进一步掌握RS－232接口标准以及RS-232帧格式，通过实验一和实验二的具体操作切实理解了通信双方参数匹配是正常通信的基本保证，明白了波特率等主要通信参数含义。

[9.2]VLAN 基本操作实验

课程名称	姓名	学号	专业	实验名称	实验日期
计算机网络实验			软件工程	VLAN基本操作实验	2019-10-30

实验目的

- 了解和掌握VLAN基本概念和操作
- 了解虚拟局域网的基本概念
- 掌握交换机实施虚拟局域网技术

实验设备

1. 一台 *CISCO* 交换机和两台计算机担当实验设备，使用两根双绞线网线，将两台计算机以太网网卡通交换机连接起来
2. 通过串行线将计算机 *Host 1* 串口 *COM 1* 同交换机 *CONSOLE* 口连接起来，使用超级终端作为交换机的操作平台。

实验网络拓扑

实验原理

1. VLAN是通过软件把网络按逻辑分组，不受物理上交换机端口所限制，把不同地理位置的主机分割到相同VLAN内，VLAN是在交换机上实现。
2. VLAN能够解决广播风暴问题。交换机的每个端口是一个冲突域，但不能隔离广播，而一个VLAN就是一个广播域。

实验内容

> 实验概要

使用 *CISCO* 交换机作为实验设备，将交换机划分成三个 *VLAN* ， *VLAN 1* 包含端口1 ~ 8， *VLAN 2* 包含端口9 ~ 16， *VLAN 3* 包含其余端口，连台计算机作为测试平台，设置成同一个子网 *IP* 地址。

1. 使用 *Host 1* 超级终端为交换机配置 *VLAN*
2. 为 *Host 1* 和 *Host 2* 以太网网卡配置同子网 *IP* 地址。
3. 然后用双绞线将两个网卡连接到交换机端口上，不断变换交换机端口，测试连通状况。（位于同一个 *VLAN* 时，两个主机将会连通；位于不同 *VLAN* 两个主机将无法连通）

> 一、未配置前端口通信实验

1. 配置两主机IP地址: 192.168.0.2 及 192.168.0.5

```
192.168.0.2 连接到交换机的端口:1
192.168.0.5 连接到交换机的端口:24
```

2. 测试192.168.0.2: ping 192.168.0.5，可连通192.168.0.2 和 192.168.0.5

> 二、配置VLAN2

1. 连接交换机，使用console线将计算机串口com2与路由器console口直接相连；
2. 建立HyperTerminal：开始 -> 程序 -> 附件 -> 通讯 -> 超级终端 -> 名称=**switch** -> 连接=**com2** -> (波特率)Baut Rate=**9600**,8,no parity, 1 stop bit；
3. 进入特权模式：

```
switch01> en(able)
pwd = cisco //输入密码
```

4. 查看VLAN配置，可查看到当前所有端口以及VLAN的配置

```
switch01#sh vlan
```

5. 建立VLAN2和VLAN3

```
// 进入VLAN配置模式
switch01#vlan database
// 建立VLAN2
switch01(vlan)#vlan 2 name vlan2
// 建立VLAN3
switch01(vlan)#vlan 3 name vlan3
// 退出
switch01(vlan)#exit
```

6. 为VLAN2分配端口f0/1，最终通过查看VLAN2配置观察是否成功分配，注意要先两次exit才能退出配置模式进行查看

```
// 进入配置模式
switch01#config t
// 进入f0/1端口
switch01(config)#in f0/1
// 将端口f0/1分配给vlan2
switch01(config -if)#switchport access vlan 2
// 退出，使得配置生效
switch01(config -if)#exit
switch01(config)#exit
// 查看VLAN2配置，VLAN2被成功分配了端口f0/1
switch01# sh vlan name vlan2
```

7. 测试192.168.0.2: ping 192.168.0.5 ， 不可连通
8. 为VLAN2分配新端口f0/24，最终通过查看VLAN2配置观察是否成功分配

```
// 进入配置模式
switch01#config t
// 进入f0/24端口
switch01(config)#in f0/24
// 将端口f0/24分配给vlan2
switch01(config -if)#switchport access vlan 2
// 退出，使得配置生效
switch01(config -if)#exit
switch01(config)#exit
// 查看VLAN2配置，VLAN2被成功分配了端口f0/24
switch01# sh vlan name vlan2
```

9. 测试192.168.0.2: ping 192.168.0.5 ， 连通

> 三、配置VLAN3

1. 为VLAN3分配端口f0/1，最终通过查看VLAN3配置观察是否成功分配

```
// 进入配置模式
switch01#config t
// 进入f0/1端口
switch01(config)#in f0/1
// 将端口f0/1分配给vlan3
switch01(config -if)#switchport access vlan 3
// 退出，使得配置生效
switch01(config -if)#exit
switch01(config)#exit
// 查看VLAN3配置，VLAN3被成功分配了端口f0/1
switch01# sh vlan name vlan3
```

2. 测试192.168.0.2: ping 192.168.0.5 ， 不可连通

> 四、删除VLAN

1. 删除VLAN3的端口f0/1和VLAN2的f0/24

```
// 进入配置模式
switch01#config t
// 进入f0/1端口
switch01(config)#in f0/1
// 将f0/1端口返还给VLAN1
switch01(config-if)#switchport access vlan 1
// 退出
switch01(config-if)#exit

// 进入f0/24端口
switch01(config)#in f0/24
// 将f0/24端口返还给VLAN1
switch01(config-if)#switchport access vlan 1
// 退出使配置生效
switch01(config-if)#exit
switch01(config)#exit
```

2. 查看VLAN配置，查看当前所有端口以及VLAN的配置。端口返还成功给VLAN1，VLAN2和VLAN3没有分配到任何端口

```
switch01#sh vlan
```

3. 删除VLAN2和VLAN3

```
// 进入VLAN配置模式
switch01#vlan database
//删除VLAN2
switch01(vlan)#no vlan 2
//删除VLAN3
switch01(vlan)#no vlan 3
//退出使配置生效
switch01(vlan)#exit
```

4. 查看VLAN配置，VLAN2和VLAN3被删除

```
switch01#sh vlan
```

实验问题与结果分析

> 一、测试192.168.0.2: ping 192.168.0.5能否连通

1. 尚未配置VLAN前，VLAN1拥有所有端口，一个VLAN即一个广播域，在同一广播域下的92.168.0.2与192.168.0.5可以连通。
2. 建立了VLAN2和VLAN3后，端口f0/1分配给VLAN2时，192.168.0.2与192.168.0.5位于不同的VLAN，不同VLAN中的两台计算机在不进行其他配置的情况下，不可连通
3. 将端口f0/24分配给VLAN2后，两台计算机又处于同一VLAN，可以连通；
4. 将端口f0/1分配给VLAN3后查看VLAN配置，一个端口只能被分配给一个VLAN(一个VLAN可以拥有多个端口，一对多关系)，两台计算机处于不同VLAN，不可连通；

二、实验注意事项

1. 分配完端口后需查看VLAN2配置，观察是否成功分配，且要退出配置模式才能进行查看；
2. 在进行实验前要注意清除两台计算机的DNS，并关闭防火墙；
3. 有时端口不可用，可切换到其他端口进行实验；
4. 注意连接交换机时，使用console线将计算机串口com2与路由器console口直接相连。
5. 要注意设备上其他同学之前配置过VLAN没有删除，否则可能一直不能连通。

三、实验心得

本次实验中接触了 *IOS* 网络操作系统，了解了实验室中的交换机与路由器，并初步掌握了该操作系统的一些常用指令，如配置端口等操作。通过对交换机进行 *VLAN* 的配置，对虚拟局域网有了更深的理解。

[10.1]静态路由配置实验

课程名称	姓名	学号	专业	实验名称	实验日期
计算机网络实验			软件工程	静态路由配置实验	2019-11-06

实验目的

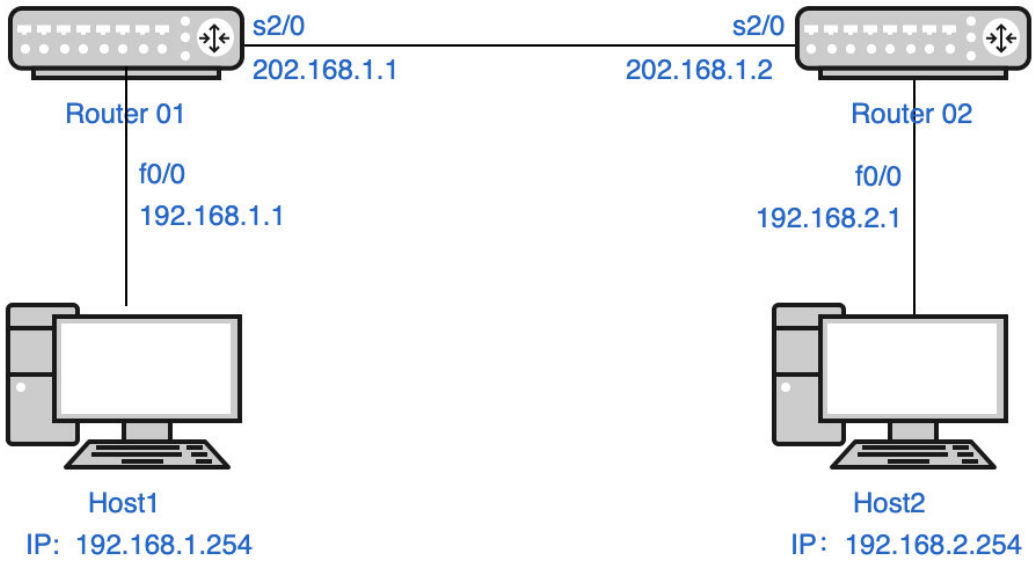
1. 了解和掌握路由器端口配置；
2. 配置静态路由，实现网际通信。

实验设备

由两台路由器、两台计算机和一台交换机组成，模拟两个远程子网互联。

- 使用单根串行交叉线将两个路由器的串口对接起来，代表路由器之间的远程网络；
- 将路由器以太网端口和两台计算机网卡都用网线直接连接到交换机，由交换机担当网络连接；
- 通过串行线将计算机串口COM同路由器CONSOLE口连接起来，两台计算机超级终端作为路由器管理的操作平台

实验网络拓扑



实验原理

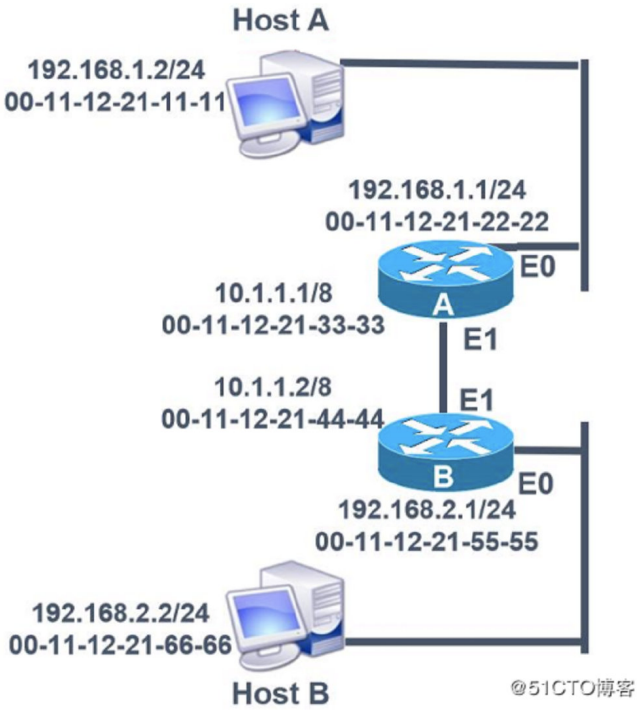
一、静态路由原理

- 路由：负责路由器控制层面的工作，决定数据包从来源端到目的端所经过的路由路径（源主机到目标主机之间的最佳传输路径）
- 路由器上的每个接口是一个广播域；交换机上的每个接口是一个冲突域*；
- 路由表：路由器中维护的路由条目的集合，路由器根据路由表做出路径选择
- 路由表的形成：直连网段和非直连网段（直接相连和间接相连）
- 非直连：静态路由和动态路由
- 静态路由由管理员手工配置，特点是单向的且缺乏灵活性（适合小范围）
- 默认路由：路由表找不到目标网络的路由条目时，将请求转发到默认路由接口（通向其他网段有且仅有一条路径）
- 优先级：当路由表中同时存在静态路由和默认路由的时候，静态路由的优先级最高，匹配上了立刻进行转发，如果没有匹配上静态路由，则按照默认路由进行转发。

二、路由器转发数据包的封装过程（三层转发原理）

- 特点：IP不变，MAC始终在变；路由器是隔离广播的

从网上找到以下资料，用于详细说明这个过程：



首先，我们的环境是所有设备都是初始状态，即对外界几乎一无所知，现在要实现主机 A和主机 B之间通信。

1. 主机A想要与主机B通信，但是目前只知道主机B的IP地址，所以要先经过网关，而此时主机A并不知道路由器A的MAC地址，所以发送ARP广播请求从而获取路由器A的MAC地址，此时四要素：

```
源IP地址：192.168.1.2/24
源MAC地址：  00-11-12-21-11-11
目的IP地址：192.168.1.1/24
目标MAC地址：FF-FF-FF-FF
```

2. 由于是ARP广播，所以路由器A的E0端口接收到这个广播，查看数据帧发现是自己的IP地址，由路由器A的E0端口给出回应，主机A收到回应进行ARP缓存，此时由主机A发送一个ping的数据包，此时四要素（主要内容是）：

```
源IP地址：192.168.1.2/24
源MAC地址：  00-11-12-21-11-11
目的IP地址：192.168.2.2/24
目标MAC地址：00-11-12-21-22-22
```

3. 路由器A的E0接口发现目标MAC地址是自己，然后拆开数据包（解开MAC头部），发现IP地址是主机B的IP地址，此时路由器A会打开路由表查询是否有该条目信息，匹配目标网段，找到路径是E1方向，此时需要重新封装MAC头部信息，但是现在无法封装，因为没有目标的MAC地址，只能将该ping包丢弃，此时路由器A的E1端口发起一个ARP广播，此时四要素是：

```
源IP地址：10.1.1.1/8
源MAC地址：00-11-12-21-33-33
目的IP地址：10.1.1.2/8
目标MAC地址：FF-FF-FF-FF
```

4. 此时路由器B的E1端口接收到此广播，发现是目标IP是自己，将给出回应，于是将（路由器B的E1端口）自己的MAC地址（00-11-12-21-44-44）告诉路由器A的E1端口，那么路由器A的E1端口缓存路由器B的E1接口的MAC地址，此时主机A的ping包可以到达路由器B的E1端口，路由器B发现找的是自己，此时开始拆包，查询路由表，找到要经过路由器B的E0端口将数据封装后发送出去，但是此时不知道目标（主机B）的MAC地址，将该ping 包丢弃，那么此时路由器B的E0端口将发起一个ARP广播，从而获取主机B的MAC地址，此时四要素是：

```
源IP地址：192.168.2.1/24
源MAC地址：00-11-12-21-55-55
目的IP地址：192.168.2.2/24
目标MAC地址：FF-FF-FF-FF
```

5. 此时主机B收到该广播，拆包发现目标IP是自己的，将对路由器B的E0接口给出回应，告诉它自己的MAC的地址，此时路由器B的E0接口缓存该回应的信息，此时主机A的ping包数据再次过来，此时四要素：

```
源IP地址：192.168.2.1/24
源MAC地址：00-11-12-21-55-55
目的IP地址：192.168.2.2/24
目标MAC地址：00-11-12-21-66-66
```

此后主机A和主机B就可以正常互相通信，从而不再需要进行ARP的广播过程。注意上述过程中的(2)和(5)，是否验证了前面所说的“IP不变，MAC始终在变”的转发特点。

实验内容

> 一、连接路由器

1. 打开路由器电源

2. 连接交换机，使用console线将计算机串口com1与路由器console口直接相连；
3. 建立HyperTerminal：开始 -> 程序 -> 附件 -> 通讯 -> 超级终端 -> 名称=**router** -> 连接=**com1** -> (波特率)Baut Rate=**9600**,8,no parity, 1 stop bit；
4. 进入特权模式：

```
router01>en(able)
pwd=cisco //输入密码
// 记录FastEthernet0/0的IP
router01#sh interface f0/0
// 记录Serial3/0的IP
router01#sh interface s2/0
```

二、配置Router的FastEthernet0/0及Serial2/0

1. router01的配置快速以太网f0/0，配置的IP地址需要与连接的Host主机的网关地址一致

```
// 进入配置模式
router01#config t
// 进入以太口
router01(config)#in f0/0
// 添加IP地址
router01(config-if)#ip address 192.168.1.1 255.255.255.0
// 开启端口功能
router01(config-if)#no shut
```

2. router01的配置串口s2/0

```
// 退到配置模式
router01(config-if)#exit
// 进入串口
router01(config)#in s2/0
// 设置IP地址
router01(config-if)#ip addr 202.168.1.1 255.255.255.0
```

3. 同理配置router02

```
`router01`
f0/0的以太网IP地址：192.168.1.1
s2/0的串口地址：202.168.1.1

`router02`
f0/0的以太网IP地址：192.168.2.1
s2/0的串口地址：202.168.1.2
```

三、静态路由配置

1. 添加对端路由

```
/******* router01路由器 *******/
router01#config t
// 添加对端路由
router01(config)#ip route 192.168.2.0 255.255.255.0 202.168.1.2
// 查看路由表
router01#sh ip route

/******* router02路由器 *******/
router02#config t
// 添加对端路由
router02(config)#ip route 192.168.1.0 255.255.255.0 202.168.1.1
// 查看路由表
router01#sh ip route
```

- route01的路由表

```
Gateway of last resort is not set

C    192.168.1.0/24 is directly connected, FastEthernet0/0
S    192.168.2.0/24 [1/0] via 202.168.1.2
C    202.168.1.0/24 is directly connected, Serial2/0
```

- route02的路由表

```
Gateway of last resort is not set

S    192.168.1.0/24 [1/0] via 202.168.1.1
C    192.168.2.0/24 is directly connected, FastEthernet0/0
C    202.168.1.0/24 is directly connected, Serial2/0
```

2. 测试，用ping连通192.168.2.254，成功连通

```
C:\>ping 192.168.2.254

Pinging 192.168.2.254 with 32 bytes of data:

Reply from 192.168.2.254: bytes=32 time=3ms TTL=126
Reply from 192.168.2.254: bytes=32 time=1ms TTL=126
Reply from 192.168.2.254: bytes=32 time=1ms TTL=126
Reply from 192.168.2.254: bytes=32 time=3ms TTL=126

Ping statistics for 192.168.2.254:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 1ms, Maximum = 3ms, Average = 2ms
```

四、缺省路由

1. 删除静态路由并添加缺省路由

```
/******* router01路由器 *******/
router01#config t
// 删除静态路由
router01(config)#no ip route 192.168.2.0 255.255.255.0
// 添加缺省路由
router01(config)#ip route 0.0.0.0 0.0.0.0 202.168.1.2
router01#sh ip route

/******* router02路由器 *******/
router02#config t
// 删除静态路由
router02(config)#no ip route 192.168.1.0 255.255.255.0
```

```
// 添加缺省路由
router02(config)#ip route 0.0.0.0 0.0.0.0 202.168.1.1
router02#sh ip route
```

- route01的路由表

```
C    192.168.1.0/24 is directly connected, FastEthernet0/0
C    202.168.1.0/24 is directly connected, Serial2/0
S*   0.0.0.0/0 [1/0] via 202.168.1.2
```

- route02的路由表

```
C    192.168.2.0/24 is directly connected, FastEthernet0/0
C    202.168.1.0/24 is directly connected, Serial2/0
S*   0.0.0.0/0 [1/0] via 202.168.1.1
```

2. 测试，用ping连通192.168.2.254，成功连通

```
C:\>ping 192.168.2.254

Pinging 192.168.2.254 with 32 bytes of data:

Reply from 192.168.2.254: bytes=32 time=1ms TTL=126
Reply from 192.168.2.254: bytes=32 time=90ms TTL=126
Reply from 192.168.2.254: bytes=32 time=1ms TTL=126
Reply from 192.168.2.254: bytes=32 time=4ms TTL=126

Ping statistics for 192.168.2.254:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 1ms, Maximum = 90ms, Average = 24ms
```

3. 检查router01的运行配置，可查看到所有配置信息

```
router01#sh running config
```

```
.
interface FastEthernet0/0
 ip address 192.168.1.1 255.255.255.0
 duplex auto
 speed auto
!
interface FastEthernet1/0
 no ip address
 duplex auto
 speed auto
 shutdown
!
interface Serial2/0
 ip address 202.168.1.1 255.255.255.0
 clock rate 2000000
!
interface Serial3/0
 no ip address
 shutdown
!
interface FastEthernet4/0
 no ip address
 shutdown
!
interface FastEthernet5/0
 no ip address
 shutdown
!
ip classless
ip route 0.0.0.0 0.0.0.0 202.168.1.2
!
ip flow-export version 9
```

实验问题与结果分析

在本次实验中，两台路由器都需要进行相似的配置，需要将路由器、主机、交换机都连接起来，静态路由配置实验与之前的两个实验相比设备、连线显得更加复杂和困难一些。

通过本次静态路由配置实验，了解和掌握了路由器相关配置指令，通过不断地配置（刚开始的时候不熟练好几个地方配置错误）对静态路由相关概念有了更深刻的理解。

[11.1]帧中继配置实验

课程名称	姓名	学号	专业	实验名称	实验日期
计算机网络实验			软件工程	帧中继配置实验	2019-11-13

实验目的

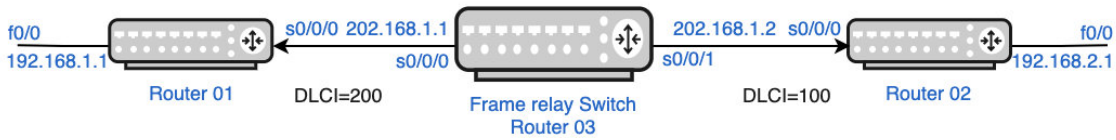
1. 了解广域网基本概念
2. 了解和掌握帧中继技术的原理，熟悉帧中继交换机永久虚电路配置步骤
3. 熟悉使用帧中继链路的路由配置

2 熟悉使用帧中继链路的路由配置。

实验设备

- 实验环境主要由三台路由器、三台计算机和一台交换机组成，模拟一个由局域网和广域网组成的网际网。
- 路由器（模拟帧中继交换机）使用单根串行交叉线将路由器A和路由器C的广域网串口连接起来，路由器B和路由器C的广域网串口连接起来（构建帧中继网络专线以连接两个远程子网），并使同 *Router C* 相连的电缆连接线类型均为 *DCE* （将两条通讯链路配置成单条PVC永久虚电路）
- 将路由器A和路由器B以太网端口和两台计算机网卡都用网线直接连接到交换机，由交换机担当网络连接
- 通过串行线将各个计算机串口COM同路由器CONSOLE口连接起来，使用各自超级终端作为路由器管理的操作平台。

实验网络拓扑



实验原理

1. 帧中继交换机

- DLCI(数据链路连接标识)：用于标识源和目的设备间的逻辑链路；
- PVC(永久虚电路)：一对进出的链路可以配置成一条虚电路；
- LMI(本地管理接口)：一种信令标志，负责管理设备之间的连接、维护，配置时需要将交换机和路

由器相连端口设置成同一种接口类型，分三种：cisco、ansi和q939a。

2. 路由器使用帧中继链路

- 帧中继映射表Router01：将IP地址同链路绑定，缺省情况下，路由器通过inverse-arp灰动态生成，也可以静态生成。

NEXT-Hop	DLCI
202.168.1.2	200

实验内容

> 一、帧中继交换机配置

1. 路由器设置成帧中继交换机

```
// 进入全局配置模式
router03#config t
// 启用作为帧中继交换机
router03(config)#frame switching
// 查看帧中继功能
router03#sh run
```

2. 设置s0/0/0端口

```
// 查看端口物理连接
router03#sh control s0/0/0
// 设置端口
router03 (config)#in s0/0/0
// 封装端口为Frame
router03 (config-if)#encap frame
// 帧封装方式
router03 (config-if)#fram intf-type dce
// 设置lmi类型
router03 (config-if)#fram lmi-type cisco
// 时钟频率
router03 (config-if)#clock rate 56000
// 配置广域路由(pvc)
router(config-if)#fram route 200 inter s0/0/1 100
// 启动端口:
router03 (config-if)#no shut
router03 (config-if)#exit
```

3. 设置s0/0/1端口

```
// 设置端口
router03 (config)#in s0/0/1
// 封装端口为Frame
router03 (config-if)#encap frame
// 配置端口类型DCE
router03 (config-if)#fram intf-type dce
// 设置lmi类型
router03 (config-if)#fram lmi-type cisco
// 时钟频率
router03 (config-if)#clock rate 56000
// 配置广域路由(pvc)
router03 (config-if)#fram route 100 inter s0/0/0 200
```

4. 检查帧中继交换机

```
// 查看配置
router03#sh run
// 查看交换机路由表，即pvc构成
router03#sh frame route
// 查看pvc使用情况
router03#sh frame pvc #端口类型DCE，DLCI用途是Switching，激活状态
```

IN_Port	IN_DLCI	OUT_Port	OUT_DLCI
S0/0/0	200	S0/1	100
S0/0/1	100	S0/0	200

> 二、配置路由器Router01和Router02

1. 配置快速以太网f0/0，相关操作与静态路由部分相似,Router01和Router02步骤相似

```
// 进入配置模式
router01#config t
// 进入以太口
router01(config)#in f0/0
// 删除旧IP地址
router01(config-if)#no ip address <ipaddress><subnet mask>
// 添加IP地址
router01(config-if)#ip address <ipaddress><subnet mask>
// 开启端口功能
router01(config-if)#no shut
```

2. 配置帧中继广域网链路

```
// 查看pvc链路号
router01#sh frame pvc #看不到
// 设置端口
router01(config)#in s0/0/0
// 封装端口为Frame
router01(config-if)#encap  frame#线路才启用;
```

```
// 查看并记录DLCI编号
router01#sh frame pvc
// 设置lmi类型
router01(config-if)#fram lmi-type cisco
// 设置IP地址
router01(config)#ip addr 202.168.2.1 255.255.255.0
// 启动端口
router01(config-if)#no shut
```

3. 地址映射管理

```
// 查看DLCI同IP地址映射状态
router01#sh frame map #已经映射就不用设立;
```

4. 静态路由配置

```
// 添加缺省路由
router01(config)#ip route 0.0.0.0 0.0.0.0 202.168.2.2 # 对端广域端口地址;
// 或添加静态路由
router01(config)#ip route 192.168.y.0 255.255.255.0 202.168.2.2
// 查看路由表: router01# sh ip route
```

三、测试

1. `ping 192.168.1.254`
2. 静态建立帧中继映射

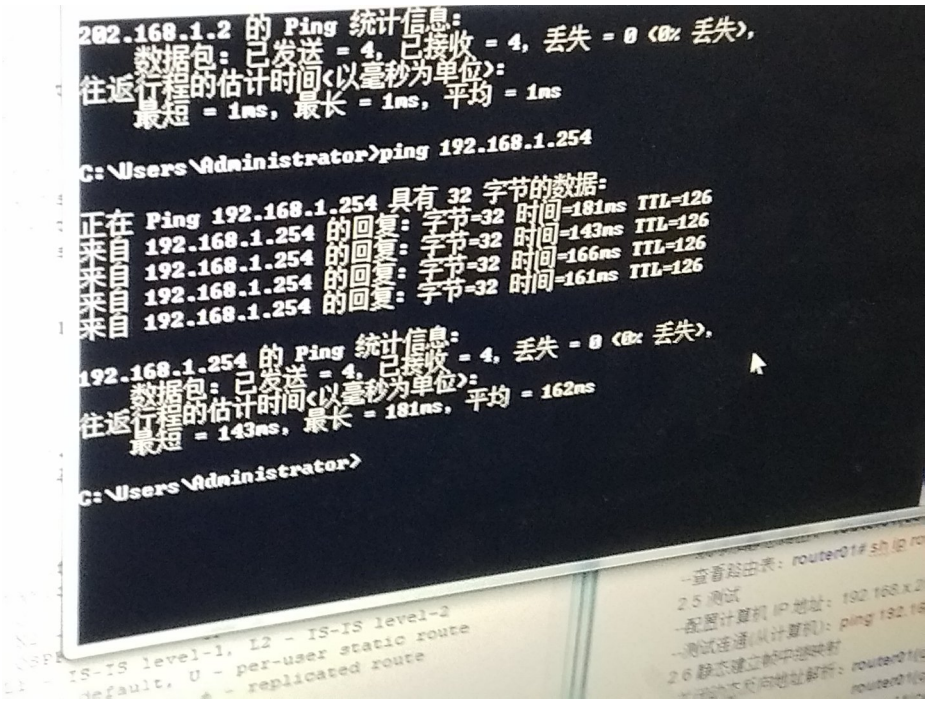
```
// 关闭动态反向地址解析
router01(config)#int s0/0/0 #指定端口
router01(config-if)#no fram inverse
router01(config-if)#shut #重启端口
router01(config-if)#no shut
// 查看DLCI同IP地址映射状态
router01#sh frame map #不存在映射
// 建立映射DLCI
router01(config-if)#fram map ip 202.168.2.2 200#对端地址
```

3. 重新测试，`ping 192.168.2.254`

四、实验结果

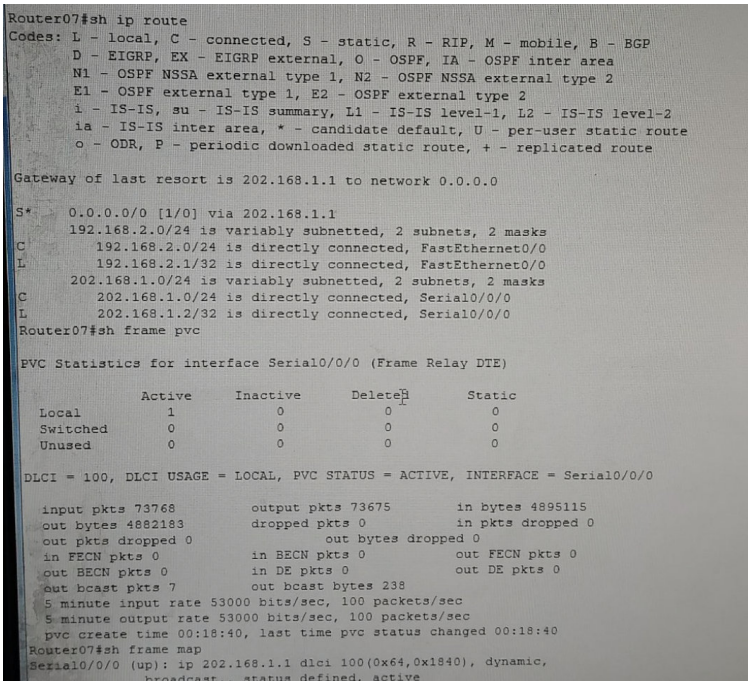
1. 测试连通

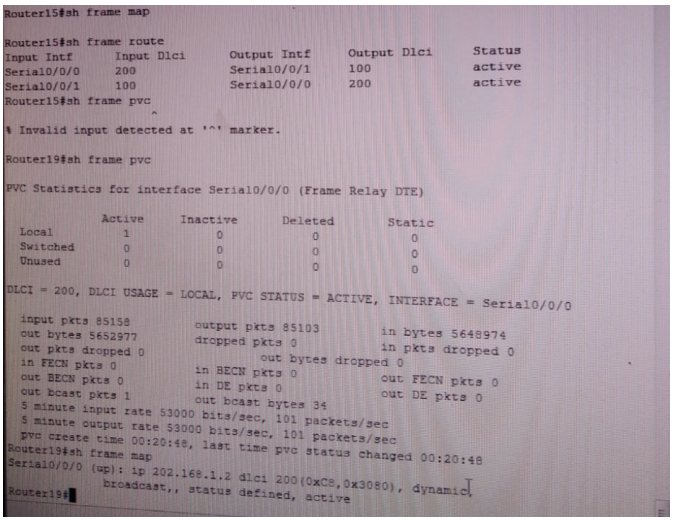
   `ping 192.168.1.254`



2. 查看路由表、查看地址映射、查看 *PVC* 使用情况

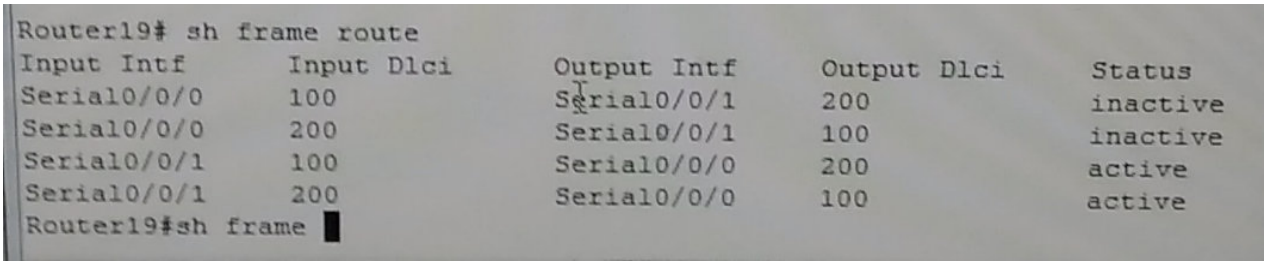
   `sh ip route` & `sh frame map` & `sh frame pvc`





3. 查看交换机路由表

   `sh frame route`



实验问题与结果分析

> 一、专业名词

1. DLCI：数据链路连接标识
2. PVC：永久虚电路

> 二、实验心得

帧中继配置实验这个实验比较复杂，完成实验也花了比较多的时间，还好之前有静态路由配置实验作为铺垫，难度衔接得还是比较有层次感的，理论课上讲到的帧中继相关理论的时候，一开始不是很理解，做了实验之后进一步认识了帧中继技术，也更加掌握了其路由表的建立的机制。

[12.1]ACL访问控制列表

课程名称	姓名	学号	专业	实验名称	实验日期
计算机网络实验			软件工程	ACL访问控制列表	2019-11-20

实验目的

1. 了解路由器包过滤基本原理
2. 了解访问控制列表实施原理
3. 利用控制列表实施网络安全
4. 掌握访问控制组配置
5. 实施包过滤，有选择地允许某些地址组进行网际通信

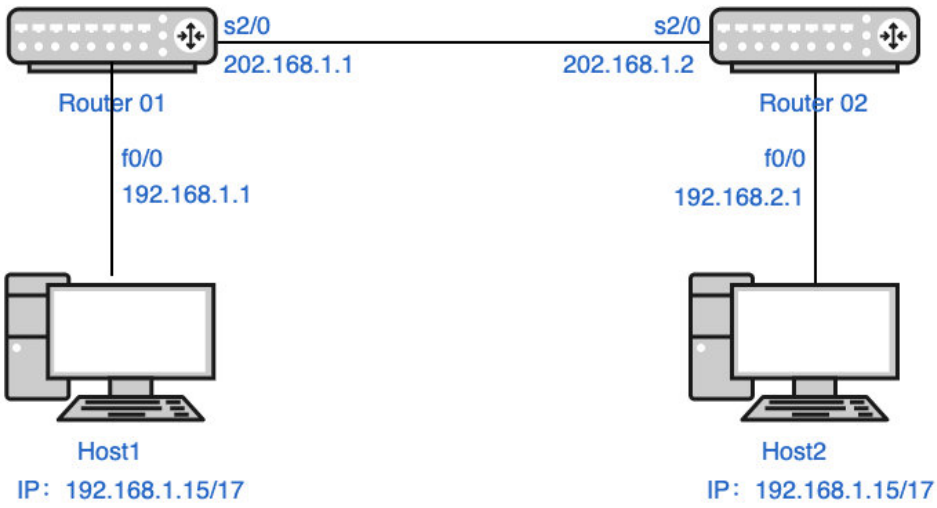
实验设备

实验环境主要由两台路由器、两台计算机和一台交换机组成。

- 使用单根串行交叉线将两个路由器的串口对接起来
- 将路由器以太网端口和两台计算机网卡都用网线直接连接交换机，由交换机担当网络连接
- 通过串行线将计算机串口 *COM* 同路由器 *CONSOLE* 连接起来，两台计算机超级终端作为路由器管理平台

路由器串口地址、端口地址，计算机网关地址、 *IP* 地址与之前实验相同

实验网络拓扑



实验原理

- 路由器对每个经过的IP包实行转发，需要对每一个IP包的头部进行分析，这就称为包过滤。正常情况下，包过滤只是转发的一个环节，但利用包过滤，可以加入其他的处理规则，比如，ACL就是添加允许是否通行的规则，起到一定的网络安全作用。
- 通配符掩码/屏蔽字，定义地址组，形式同字网掩妈相反，1表示忽略，0表示保留；150.1.1.x 0.0.0.255 150.1.1.0
- 包过滤方向，指流入包还是流出包，路由器上有两个位置都可使用存取表，最优选择应离数据流最近的地方；对于内部地址，串口应是out，以太口应是in；对于外部地址，以太口应是out，串口应是in。
- 其工程主要作用是保护内部主机，或阻止访问特定不安全的外部主机。

实验内容

> 一、连接路由器

1. 打开路由器电源
2. 连接交换机，使用console线将计算机串口**com1**与路由器console口直接相连；
3. 建立HyperTerminal：开始 -> 程序 -> 附件 -> 通讯 -> 超级终端 -> 名称=**router** -> 连接=**com1** -> (波特率)Baut Rate=**9600**,8,no parity, 1 stop bit；
4. 进入特权模式：

```
router01>en(able)
pwd=cisco //输入密码
```

> 二、静态路由配置

1. 配置Router01快速以太f0/0

```
// 进入配置模式
router01#config t
// 进入以太口:
router01(config)#in f0/0
// 添加IP地址
router01 (config-if)#ip address 192.168.1.1 255.255.255.0
// 开启端口功能
router01(config-if)#no shut
```

2. 配置Router01串口，s2/0

```
// 退到配置模式
router01(config-if)#exit
// 进入串口
router01(config)#in s2/0
// 设置地址
router01 (config-if)#ip address 202.168.1.1 255.255.255.0
// 开启端口功能
router01(config-if)#no shut
```

3. 同理配置router02

```
`router01`
f0/0的以太网IP地址：192.168.1.1
s2/0的串口地址：202.168.1.1

`router02`
f0/0的以太网IP地址：192.168.2.1
s2/0的串口地址：202.168.1.2
```

4. 静态缺省路由配置

```
/******* router01路由器 *******/
router01#config t
// 添加缺省路由
router01(config)#ip route 0.0.0.0 0.0.0.0 202.168.1.2
router01#sh ip route

/******* router02路由器 *******/
router02#config t
// 添加缺省路由
router02(config)#ip route 0.0.0.0 0.0.0.0 202.168.1.1
router02#sh ip route
```

   - route01的路由表

```
C    192.168.1.0/24 is directly connected, FastEthernet0/0
C    202.168.1.0/24 is directly connected, Serial2/0
S*   0.0.0.0/0 [1/0] via 202.168.1.2
```

   - route02的路由表

```
C    192.168.2.0/24 is directly connected, FastEthernet0/0
C    202.168.1.0/24 is directly connected, Serial2/0
S*   0.0.0.0/0 [1/0] via 202.168.1.1
```

5. host测试，用ping连通192.168.2.15，成功连通

```
C:\>ping 192.168.2.15

Pinging 192.168.2.15 with 32 bytes of data:

Request timed out.
Reply from 192.168.2.15: bytes=32 time=4ms TTL=126
Reply from 192.168.2.15: bytes=32 time=2ms TTL=126
Reply from 192.168.2.15: bytes=32 time=1ms TTL=126

Ping statistics for 192.168.2.15:
    Packets: Sent = 4, Received = 3, Lost = 1 (25% loss),
Approximate round trip times in milli-seconds:
    Minimum = 1ms, Maximum = 4ms, Average = 2ms
```

> 三、访问控制组, 只允许内部地址外出

1. 目的

   只允许内部地址外出permit：192.168.1.0/28

   ```
   192.168.1.15  可以外出；
   192.168.1.17  不可以外出；
   ```

2. 建立访问控制组

   ```
   // 进入配置模式
   router01#config t
   // 建立访问控制组1
   router01(config)# access-list 1 permit 192.168.1.0  0.0.0.15
   // 查看访问控制组
   router01#sh access-list
   ```

   router01的访问控制组：

   ```
   Router#sh access-list
   Standard IP access list 1
       10 permit 192.168.1.0 0.0.0.15
   ```

3. 实施包过滤1

   ```
   // 指定以太端口
   router01# config t
   router01(config)#in f0/0
   // 建立映射
   router01(config)#ip access-group 1 in
   // 查看包过滤
   router01#sh run
   ```

   查看包过滤：

   ```
   interface FastEthernet0/0
    ip address 192.168.1.1 255.255.255.0
    ip access-group 1 in
    duplex auto
    speed auto
   ```

4. host测试，用ping连通对端计算机

   ```
   // 测试连通(从计算机)
   ping 192.168.2.15  # 对端路由器，成功
   // 修改计算机IP地址：192.168.1.17
   // 测试连通(从计算机)
   ping 192.168.2.15   # 对端路由器，不成功
   ```

   第一次测试：

```
C:\>ping 192.168.2.15

Pinging 192.168.2.15 with 32 bytes of data:

Reply from 192.168.2.15: bytes=32 time=50ms TTL=126
Reply from 192.168.2.15: bytes=32 time=47ms TTL=126
Reply from 192.168.2.15: bytes=32 time=3ms TTL=126
Reply from 192.168.2.15: bytes=32 time=1ms TTL=126

Ping statistics for 192.168.2.15:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 1ms, Maximum = 50ms, Average = 25ms
```

修改计算机IP地址：192.168.1.17后，第二次测试：

```
C:\>ping 192.168.2.15

Pinging 192.168.2.15 with 32 bytes of data:

Reply from 192.168.1.1: Destination host unreachable.
Reply from 192.168.1.1: Destination host unreachable.
Reply from 192.168.1.1: Destination host unreachable.
Reply from 192.168.1.1: Destination host unreachable.

Ping statistics for 192.168.2.15:
    Packets: Sent = 4, Received = 0, Lost = 4 (100% loss),
```

5. 实施包过滤2

```
// 指定实施端口
router01# config t
router01(config)#in f0/0
// 删除映射
router01(config)# no ip access-group 1 in
// 查看包过滤
router01# sh run
// 指定串行端口
router01#config t
router01(config)#in s2/0
// 建立映射
router01(config)#ip access-group 1 out
```

删除映射后查看包过滤：

```
interface FastEthernet0/0
 ip address 192.168.1.1 255.255.255.0
 duplex auto
 speed auto
```

s2/0建立映射后查看包过滤：

```
interface FastEthernet0/0
 ip address 192.168.1.1 255.255.255.0
 duplex auto
 speed auto
!
interface FastEthernet1/0
 no ip address
 duplex auto
 speed auto
 shutdown
!
interface Serial2/0
 ip address 202.168.1.1 255.255.255.0
 ip access-group 1 out
 clock rate 2000000
```

三、访问控制组, 只允许外部地址进入

1. 目的

只允许外部地址进入permit：192.168.y.0/28

```
192.168.2.15  可以进入；
192.168.2.17  不可以进入；
```

2. 建立访问控制组2

```
// 进入配置模式
router01#config t
// 建立访问控制组2
router01(config)#access-list 2 permit 192.168.2.0  0.0.0.15
// 查看访问控制组
router01#sh access-list
```

router01的访问控制组：

```
Router#sh access-list
Standard IP access list 1
    10 permit 192.168.1.0 0.0.0.15 (4 match(es))
Standard IP access list 2
    10 permit 192.168.2.0 0.0.0.15
```

3. 实施包过滤

```
// 指定以太端口
router01#config t
router01(config)#in s2/0
// 建立映射
router01(config)#ip access-group 2 in
// 查看包过滤
router01#sh run
```

查看包过滤：

```
interface Serial2/0
 ip address 202.168.1.1 255.255.255.0
 ip access-group 2 in
 ip access-group 1 out
 clock rate 2000000
```

4. host测试，用ping连通对端计算机，本机计算机IP地址：192.168.1.15

```
// 测试连通(从对端计算机)
ping 192.168.2.15# 对端路由器，成功
// 配置对端计算机IP地址：192.168.2.17
// 测试连通(从对端计算机)
ping 192.168.2.17# 对端路由器，不成功
```

第一次测试，成功：

```
C:\>ping 192.168.2.15

Pinging 192.168.2.15 with 32 bytes of data:

Reply from 192.168.2.15: bytes=32 time=50ms TTL=126
Reply from 192.168.2.15: bytes=32 time=47ms TTL=126
Reply from 192.168.2.15: bytes=32 time=3ms TTL=126
Reply from 192.168.2.15: bytes=32 time=1ms TTL=126

Ping statistics for 192.168.2.15:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 1ms, Maximum = 50ms, Average = 25ms
```

修改对端计算机IP地址：192.168.2.17后，第二次测试，不成功：

```
C:\>ping 192.168.2.17

Pinging 192.168.2.17 with 32 bytes of data:

Reply from 192.168.1.1: Destination host unreachable.
Reply from 192.168.1.1: Destination host unreachable.
Reply from 192.168.1.1: Destination host unreachable.
Reply from 192.168.1.1: Destination host unreachable.

Ping statistics for 192.168.2.17:
    Packets: Sent = 4, Received = 0, Lost = 4 (100% loss),
```

5. 查看

```
//查看运行状态
router01# show run
```

```
.
interface FastEthernet0/0
 ip address 192.168.1.1 255.255.255.0
 duplex auto
 speed auto
!
interface FastEthernet1/0
 no ip address
 duplex auto
 speed auto
 shutdown
!
interface Serial2/0
 ip address 202.168.1.1 255.255.255.0
 ip access-group 2 in
 ip access-group 1 out
 clock rate 2000000
!
interface Serial3/0
 no ip address
 shutdown
!
interface FastEthernet4/0
 no ip address
 shutdown
!
interface FastEthernet5/0
 no ip address
 shutdown
!
ip classless
ip route 0.0.0.0 0.0.0.0 202.168.1.2
!
ip flow-export version 9
!
!
access-list 1 permit 192.168.1.0 0.0.0.15
access-list 2 permit 192.168.2.0 0.0.0.15
```

```
// 查看访问控制组
router01# show access-list 1
router01# show access-list 2
```

```
Router#show access-list 1
Standard IP access list 1
    permit 192.168.1.0 0.0.0.15 (8 match(es))

Router#show access-list 2
Standard IP access list 2
    permit 192.168.2.0 0.0.0.15 (4 match(es))
```

```
// 删除
router01(config) no access-list 1
// 测试连通(从计算机)
ping 192.168.2.15# 对端计算机,成功
```

```
C:\>ping 192.168.2.15

Pinging 192.168.2.15 with 32 bytes of data:

Reply from 192.168.2.15: bytes=32 time=53ms TTL=126
Reply from 192.168.2.15: bytes=32 time=7ms TTL=126
Reply from 192.168.2.15: bytes=32 time=78ms TTL=126
Reply from 192.168.2.15: bytes=32 time=6ms TTL=126

Ping statistics for 192.168.2.15:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 6ms, Maximum = 78ms, Average = 36ms
```

实验问题与结果分析

有了前面实验的铺垫，已经对连线和路由器配置等基本操作比较熟悉，实验比较顺畅。ACL控制访问列表实验是一个基于静态路由配置的实验，其网络拓扑结构与静态路由配置实验完全相同。通过本次实验掌握了访问控制组的相关配置指令，初步认识了路由器包过滤机制，更加认识了网络安全的相关内容。

[13.1]NAT网络地址转换

课程名称	姓名	学号	专业	实验名称	实验日期
计算机网络实验			软件工程	NAT网络地址转换	2019-11-27

实验目的

1. 了解地址转换原理，理解私有网与互联网互通和互联网接入共享原理
2. 了解 *NAT* 技术
3. 了解和掌握路由器地址转换功能，作为网络安全内容

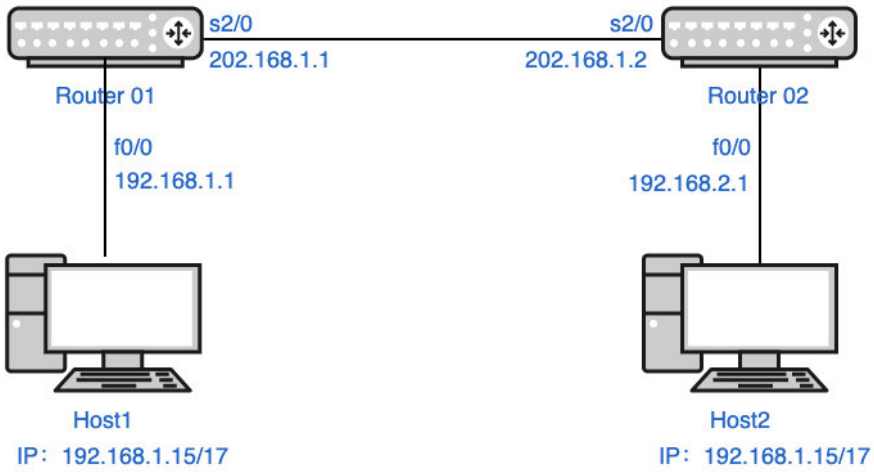
实验设备

由两台路由器、两台计算机和一台交换机组成。

- 使用单根串行交叉线将两个路由器的串口对接起来
- 将路由器以太网端口和两台计算机网卡都用网线直接连接交换机，由交换机担当网络连接
- 通过串行线将计算机串口 *COM* 同路由器 *CONSOLE* 连接起来，两台计算机超级终端作为路由器管理平台

路由器串口地址、端口地址，计算机网关地址、 *IP* 地址与之前实验相同

实验网络拓扑



实验原理

1. 网络地址转换（Network Address Translation，缩写为NAT），也叫做网络掩蔽或者IP掩蔽（IP masquerading），是一种在IP数据包通过路由器或防火墙时重写来源IP地址或目的IP地址的技术。这种技术被普遍使用在有多台主机但只通过一个公有IP地址访问因特网的私有网络中。它是一个方便且得到了广泛应用的技术。
2. NAT缺点：让主机之间的通信变得复杂，导致了通信效率的降低。
3. NAT将网络划分为内部网络(inside)和外部网络(outside)两部分。局域网主机利用NAT访问网络时，是将局域网内部的本地地址转换为全局地址(互联网合法IP地址)后转发数据包。
4. NAT有三种类型：静态NAT(StaticNAT)、动态地址NAT(PooledNAT)、网络地址端口转换NAPT（Port-LevelNAT）。
5. 网络地址端口转换NAPT（NetworkAddressPortTranslation）则是把内部地址映射到外部网络的一个IP地址的不同端口上。它可以将中小型的网络隐藏在一个合法的IP地址后面。NAPT与动态地址NAT不同，它将内部连接映射到外部网络中的一个单独的IP地址上，同时在该地址上加上一个由NAT设备选定的端口号。 NAPT是使用最普遍的一种转换方式。

实验内容

> 一、连接路由器

1. 打开路由器电源
2. 连接交换机，使用console线将计算机串口com1与路由器console口直接相连；
3. 建立HyperTerminal：开始 -> 程序 -> 附件 -> 通讯 -> 超级终端 -> 名称=**router** -> 连接=**com1** -> (波特率)Baut Rate=**9600**,8,no parity, 1 stop bit；
4. 进入特权模式：

```
router01>en(able)
pwd=cisco //输入密码
```

> 二、转换方案

1. 转换方案如下：

```
// 不转换，内网访问
192.168.1.1～15
// 转换，外网访问
192.168.1.16～31 -> 202.168.1.4～6
```

> 三、建立NAT

1. 在Router01建立访问控制组和转换地址池。

```
// 建立访问控制组
router01(config)# access-list 1 permit 192.168.1.16 0.0.0.15
// 建立转换地址池
router01(config)# ip nat pool mypool 202.168.1.4 202.168.1.6 netmask
255.255.255.0
```

2. 设置动态NAT关系

```
// 设置动态NAT关系
router05(config)# ip nat inside source list 1 pool mypool
```

四、端口设置

1. 设置Router01出口，端口 s2/0

```
// 设置端口 s0/0
router01 (config)#in s2/0
// 设置地址
router01 (config-if)#ip address 202.168.1.1 255.255.255.0
// 设置转换方向，出口
router01 (config-if)#ip nat outside
```

2. 设置Router01进口，端口 f0/0

```
// 设置端口 f0/0
router01 (config)#in f0/0
// 设置地址
router01 (config-if)#ip address 192.168.1.1 255.255.255.0
// 设置转换方向，进口
router01 (config-if)#ip nat inside
```

3. 设置单向路由

```
router01 (config)#ip route 0.0.0.0 0.0.0.0 202.168.1.2
```

五、测试

1. 检测

```
router01# debug ip nat
```

2. 配置计算机Host1的IP地址如下，此时是内网访问，不会被转成202.168.1.4~6，不能ping通

```
Host1的IP地址：192.168.1.15
ping 192.168.2.1
```

3. 配置计算机Host1的IP地址如下，此时是外网访问，会被转成202.168.1.4~6，能ping通

```
Host1的IP地址：192.168.1.17
ping 192.168.2.1
```

4. 查看转换地址表，动态转换只有才完成时才会显示。

```
router01# sh ip nat translation
```

实验问题与结果分析

> 一、实验总结

1. NAT全称是“NetworkAddressTranslation”，即“网络地址转换”。
2. 实验场景类似于微观角度的同济校园网，允许一个整体机构以一个公用IP（InternetProtocol）地址出现Internet上，是一种把内部私有网络地址（IP地址）翻译成合法网络IP地址的技术。
3. 记得设置端口方向inside和outside。

> 二、实验心得

NAT网络地址转换实验是一个基于静态路由配置的实验，需要用到上节课学到的访问控制组的相关配置指令。本次实验一开始的时候对NAT网络地址转换有较大的疑惑，尚未学习相关的理论，感觉比较抽象。后来通过跟老师的交流并结合网上一些资料，对NAT网络地址转换的原理有了更深入的理解，了解了VPN的相关原理和私网与公网的接入原理。

```
<div style="page-break-after: always"> </div>
```

[14.1]RIP动态路由配置

课程名称	姓名	学号	专业	实验名称	实验日期
计算机网络实验			软件工程	RIP动态路由配置	2019-12-04

实验目的

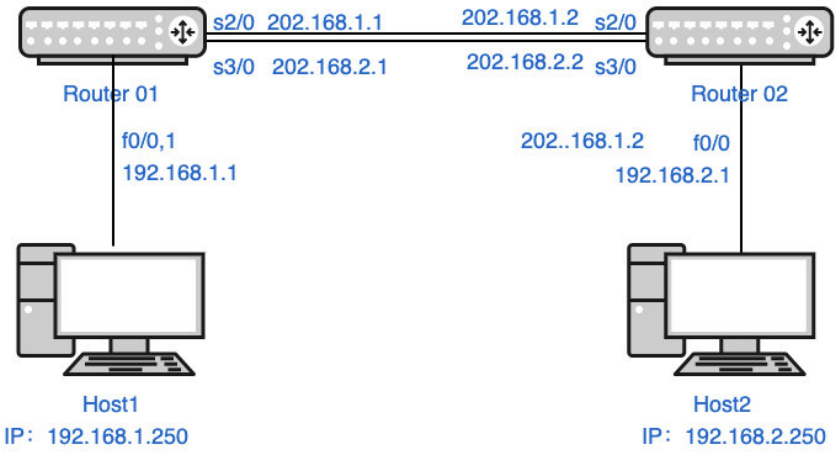
1. 了解动态路由表生成基本原理
2. 了解最短路径优先算法基本思想
3. 了解掌握开放式最短路径优先 *OSPF* 概念及其动态路由技能
4. 配置 *OSPF* 动态路由，实现网际通信

实验设备

由两台路由器、两台计算机和一台交换机组成。

- 使用两根串行交叉线将两个路由器的串口对接起来，创建两个远程传输子网，便于动态路由选择。
- 将路由器以太网端口和两台计算机网卡都用网线直接连接到交换机，由交换机担当网络连接。
- 通过串行线将两台计算机串口 *COM* 和路由器 *CONSOLE* 口连接起来，两台计算机超级终端作为路由器管理的操作平台

实验网络拓扑



实验原理

> 一、动态路由协议概述

路由协议是运行在路由器上的软件进程，与其他路由器上相同路由协议之间交换路由信息，学习非直连网络的路由信息，加入路由表。并且在网络拓扑结构变化时自动调整，维护正确的路由信息。

动态路由协议通过路由信息的交换生成并维护转发引擎需要的路由 表。网络拓扑结构改变时自动更新路由表，并负责决定数据传输最佳路径。

动态路由协议的**优点**是可以自动适应网络状态的变化，自动维护路由信息而不用网络管理员的参与。其**缺点**为由于需要相互交换路由信息，需要占用网络带宽，并且要占用系统资源。另外安全性也不如使用静态路由。在有冗余连接的复杂网络环境中，适合采用动态路由协议。目的网络是否可达取决于网络状态

动态路由协议分类

- 按路由算法划分：
  - 距离-矢量路由协议(如RIP)： 定期广播整个路由信息，易形成路由环路，收敛慢
  - 链路状态路由协议（如OSPF）： 收集网络拓扑信息，运行协议算法 计算最佳路由根本解决路由环路问题，收敛快

> 二、RIP协议概述

RIP协议采用距离向量算法，在实际使用中已经较少适用。在默认情况下，RIP使用一种非常简单的度量制度：距离就是通往目的站点所需经过的链路数，取值为0~16，数值16表示路径无限长。RIP进程使用UDP的520端口来发送和接收RIP分组。RIP分组每隔30s以广播的形式发送一次，为了防止出现“广播风暴”，其后续的分组将做随机延时后发送。在RIP中，如果一个路由在180s内未被刷，则相应的距离就被设定成无穷大，并从路由表中删除该表项。RIP分组分为两种：请求分组和响应分组。

RIP-1被提出较早，其中有许多缺陷。为了改善RIP-1的不足，在RFC1388中提出了改进的RIP-2，并在RFC1723和RFC2453中进行了修订。RIP-2定义了一套有效的改进方案，新的RIP-2支持子网路由选择，支持CIDR，支持组播，并提供了验证机制。

随着OSPF和IS-IS的出现，许多人认为RIP已经过时了。但事实上RIP也有它自己的优点。对于小型网络，RIP就所占带宽而言开销小，易于配置、管理和实现，并且RIP还在大量使用中。但RIP也有明显的不足，即当有多个网络时会出现环路问题。为了解决环路问题，IETF提出了分割范围方法，即路由器不可以通过它得知路由的接口去宣告路由。分割范围解决了两个路由器之间的路由环路问题，但不能防止3个或多个路由器形成路由环路。触发更新是解决环路问题的另一方法，它要求路由器在链路发生变化时立即传输它的路由表。这加速了网络的聚合，但容易产生广播泛滥。总之，环路问题的解决需要消耗一定的时间和带宽。若采用RIP协议，其网络内部所经过的链路数不能超过15，这使得RIP协议不适于大型网络。

实验内容

> 一、连接路由器

1. 打开路由器电源
2. 连接交换机，使用console线将计算机串口com1与路由器console口直接相连；
3. 建立HyperTerminal：开始 -> 程序 -> 附件 -> 通讯 -> 超级终端 -> 名称=**router** -> 连接=**com1** -> (波特率)Baut Rate=**9600**,8,no parity, 1 stop bit；
4. 进入特权模式：

```
router01>en(able)
pwd=cisco //输入密码
```

> 二、配置Router的IP地址和动态路由

1. router01的配置快速以太网f0/0

```
// 进入配置模式
router01#config t
// 进入以太口
router01(config)#in f0/0
// 添加IP地址
router01(config-if)#ip address 192.168.1.1 255.255.255.0
// 开启端口功能
router01(config-if)#no shut
```

2. router01的配置串口s2/0

```
// 退到配置模式
router01(config-if)#exit
// 进入串口
router01(config)#in s2/0
// 设置IP地址
router01(config-if)#ip addr 202.168.1.1 255.255.255.0
```

3. router01的配置串口s3/0

```
// 退到配置模式
router01(config-if)#exit
// 进入串口
router01(config)#in s3/0
// 设置IP地址
router01(config-if)#ip addr 202.168.2.1 255.255.255.0
// 设置带宽
router01(config-if)#band 256
```

4. router01配置RIP动态路由，如果路由功能关闭，rip必须重新配置

```
// 添加RIP
router01(config)#router rip
// 指定邻居网络
router01(config-router)# network 192.168.1.0
router01(config-router)# network 202.168.1.0
router01(config-router)# network 202.168.2.0
// 查看RIP路由表
router01# sh ip route rip
```

结果如下：

```
Router#sh ip route rip
R    192.168.2.0/24 [120/1] via 202.168.1.2, 00:00:02, Serial2/0
                    [120/1] via 202.168.2.2, 00:00:02, Serial3/0
```

5. 同理配置router02，以下为两个路由器配置的IP地址和动态路由

```
`router01`
f0/0的以太网IP地址：192.168.1.1
s2/0的串口地址：202.168.1.1
s3/0的串口地址：202.168.2.1，其中带宽设置为256
RIP路由表：  192.168.2.0 202.168.1.0 202.168.2.0

`router02`
f0/0的以太网IP地址：192.168.2.1
s2/0的串口地址：202.168.1.1
s3/0的串口地址：202.168.2.1，其中带宽设置为256
RIP路由表：  192.168.2.0 202.168.1.0 202.168.2.0
```

三、测试

1. 配置计算机IP地址

```
Host1: 192.168.1.250
Host2: 192.168.2.250
```

2. 输入如下命令：

```
router01(config)#no ip domain-lookup
router01(config)#exit
router01#trace ip 192.168.2.250
```

结果如下：

```
Router#trace ip 192.168.2.250
Type escape sequence to abort.
Tracing the route to 192.168.2.250

  1   202.168.2.2     36 msec   1 msec    3 msec
  2   192.168.2.250   1 msec    1 msec    0 msec
```

3. 跟踪调试

```
// 查看信息发送端口
router01#debug ip rip
```

结果如下：

```
Router#debug ip rip
RIP protocol debugging is on
Router#RIP: sending  v1 update to 255.255.255.255 via FastEthernet0/0 (192.168.1.1)
RIP: build update entries
      network 192.168.2.0 metric 2
      network 202.168.1.0 metric 1
      network 202.168.2.0 metric 1
RIP: sending  v1 update to 255.255.255.255 via Serial3/0 (202.168.2.1)
RIP: build update entries
      network 192.168.1.0 metric 1
      network 202.168.1.0 metric 1
RIP: sending  v1 update to 255.255.255.255 via Serial2/0 (202.168.1.1)
RIP: build update entries
      network 192.168.1.0 metric 1
      network 202.168.2.0 metric 1
```

4. 被动接口设置

```
// 进入RIP设置
router01(config)#router rip
// 以太网端口配置成被动模式
router01(config-router)#passive-interface f0/0
```

5. 查看调试，以太口不再发送：

```
RIP: sending  v1 update to 255.255.255.255 via Serial2/0 (202.168.1.1)
RIP: build update entries
      network 192.168.1.0 metric 1
      network 202.168.2.0 metric 1
RIP: received v1 update from 202.168.1.2 on Serial2/0
      192.168.2.0 in 1 hops
      202.168.2.0 in 1 hops
RIP: received v1 update from 202.168.2.2 on Serial3/0
      192.168.2.0 in 1 hops
      202.168.1.0 in 1 hops
RIP: sending  v1 update to 255.255.255.255 via Serial3/0 (202.168.2.1)
RIP: build update entries
      network 192.168.1.0 metric 1
      network 202.168.1.0 metric 1
RIP: sending  v1 update to 255.255.255.255 via Serial2/0 (202.168.1.1)
RIP: build update entries
      network 192.168.1.0 metric 1
      network 202.168.2.0 metric 1
RIP: received v1 update from 202.168.1.2 on Serial2/0
      192.168.2.0 in 1 hops
      202.168.2.0 in 1 hops
RIP: received v1 update from 202.168.2.2 on Serial3/0
      192.168.2.0 in 1 hops
      202.168.1.0 in 1 hops
RIP: sending  v1 update to 255.255.255.255 via Serial3/0 (202.168.2.1)
RIP: build update entries
      network 192.168.1.0 metric 1
      network 202.168.1.0 metric 1
RIP: sending  v1 update to 255.255.255.255 via Serial2/0 (202.168.1.1)
RIP: build update entries
      network 192.168.1.0 metric 1
      network 202.168.2.0 metric 1
```

实验问题与结果分析

> 一、注意事项

1. 一开始的时候将router02的两个串口地址配置成了202.168.1.1和202.168.1.2，两边路由串口设置成了一样的，后来在查看rip路由表的时候发现了这个问题

> 二、实验心得

通过本次实验，我掌握了RIP协议的概念和原理，初步了解了RIP和OSPF的异同点。本实验跟后续的OSPF动态路由配置实验拓扑结构上相同，而且配置指令也相差无几。

[14.2]OSPF动态路由配置

课程名称	姓名	学号	专业	实验名称	实验日期
计算机网络实验			软件工程	OSPF动态路由配置	2019-12-04

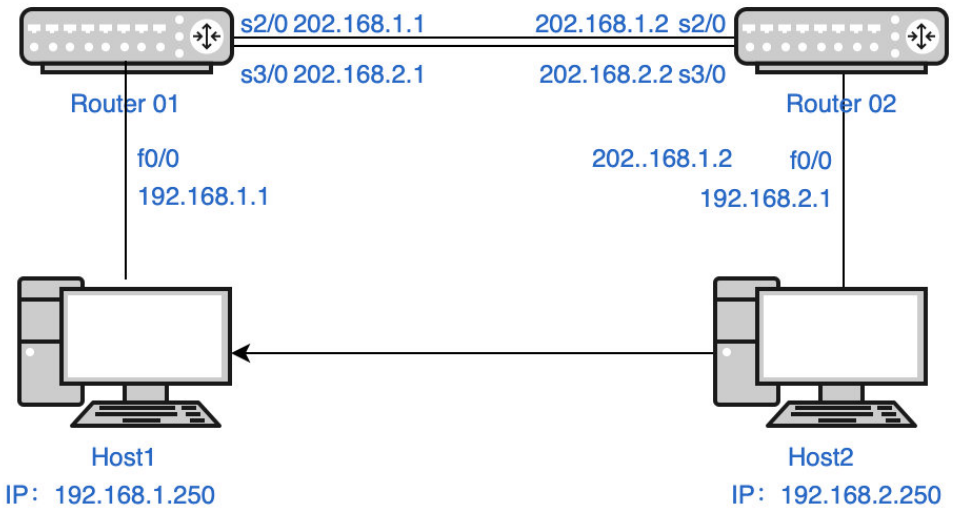
实验目的

1. 了解和掌握开放式最短路径优先OSPF概念；
2. 配置OSPF动态路由，实现网际通信。

实验设备

1. 两台路由器，使用串行线将两个0串口对接；
2. 两台计算机作为操作平台；
3. 一台交换机担当网络连接。

实验网络拓扑



实验原理

> 一、OSPF协议概述

OSPF路由协议是用于[网际协议（IP）网络的链路状态路由协议。该协议使用链路状态路由算法的内部网关协议（IGP），在单一自治系统（AS）内部工作。适用于IPv4的OSPFv2协议定义于RFC 2328，RFC 5340定义了适用于IPv6的OSPFv3。

开放式最短路径优先（Open Shortest Path First，OSPF）是目前广泛使用的一种动态路由协议，它属于链路状态路由协议，具有路由变化收敛速度快、无路由环路、支持变长子网掩码（VLSM）和汇总、层次区域划分等优点。在网络中使用OSPF协议后，大部分路由将由OSPF协议自行计算和生成，无须网络管理员人工配置，当网络拓扑发生变化时，协议可以自动计算、更正路由，极大地方便了网络管理。但如果使用时不结合具体网络应用环境，不做好细致的规划，OSPF协议的使用效果会大打折扣，甚至引发故障。[1]

OSPF协议是一种链路状态协议。每个路由器负责发现、维护与邻居的关系，并将已知的邻居列表和链路费用LSU(Link State Update)报文描述，通过可靠的泛洪与自治系统AS(Autonomous System)内的其他路由器周期性交互，学习到整个自治系统的网络拓扑结构;并通过自治系统边界的路由器注入其他AS的路由信息，从而得到整个Internet的路由信息。每隔一个特定时间或当链路状态发生变化时，重新生成LSA，路由器通过泛洪机制将新LSA通告出去，以便实现路由的实时更新。

实验内容

> 一、连接路由器

1. 打开路由器电源
2. 连接交换机，使用console线将计算机串口com1与路由器console口直接相连；
3. 建立HyperTerminal：开始 -> 程序 -> 附件 -> 通讯 -> 超级终端 -> 名称=**router** -> 连接=**com1** -> (波特率)Baut Rate=**9600**,8,no parity, 1 stop bit；
4. 进入特权模式：

```
router01>en(able)
pwd=cisco //输入密码
```

> 二、配置Router的IP地址和动态路由

1. router01的配置快速以太网f0/0

```
// 进入配置模式
router01#config t
// 进入以太口
router01(config)#in f0/0
// 添加IP地址
router01(config-if)#ip address 192.168.1.1 255.255.255.0
// 开启端口功能
router01(config-if)#no shut
```

2. router01的配置串口s2/0

```
// 退到配置模式
router01(config-if)#exit
// 进入串口
router01(config)#in s2/0
// 设置IP地址
router01(config-if)#ip addr 202.168.1.1 255.255.255.0
```

3. router01的配置串口s3/0

```
// 退到配置模式
router01(config-if)#exit
// 进入串口
router01(config)#in s3/0
// 设置IP地址
router01(config-if)#ip addr 202.168.2.1 255.255.255.0
// 设置带宽
router01(config-if)#band 256
```

4. router01配置OSPF动态路由，如果路由功能关闭，rip必须重新配置，同理配置router02。

```
// 添加OSPF
router01(config)#router ospf 100
// 指定邻居网络
router01(config-router)# network 192.168.1.0 0.0.0.255 area 0
router01(config-router)# network 202.168.1.0 0.0.0.255 area 0
router01(config-router)# network 202.168.2.0 0.0.0.255 area 0
// 查看OSPF ID
router01# sh ip ospf int
// 查看OSPF路由表
router01# sh ip route ospf
// 查看OSPF邻居
router01# sh ip ospf nei
```

结果如下：

OSPF路由表：

```
Router#sh ip route ospf
O    192.168.2.0 [110/65] via 202.168.1.2, 00:01:11, Serial2/0
```

OSPF邻居：

```
Router# sh ip ospf nei


Neighbor ID     Pri   State           Dead Time   Address         Interface
202.168.2.2       0   FULL/  -        00:00:34    202.168.1.2     Serial2/0
202.168.2.2       0   FULL/  -        00:00:39    202.168.2.2     Serial3/0
```

三、测试

1. 配置计算机IP地址

```
Host1: 192.168.1.250
Host2: 192.168.2.250
```

2. 输入如下命令，测试连通：`ping 192.168.2.1`，测试连通，表示网关、远程连接子网和动态路由均发挥作用。

```
router01(config)#no ip domain-lookup
router01(config)#exit
router01#trace ip 192.168.2.250
```

结果如下，跟踪传输路径：`trace ip 192.168.2.250`，经过了网关、远程连接子网 `202.168.1.0/24` 的 `202.168.1.2` 端口，没有经过远程连接子网 `202.168.2.0/24`

```
Router#trace ip 192.168.2.250
Type escape sequence to abort.
Tracing the route to 192.168.2.250

  1   202.168.1.2     37 msec    0 msec    3 msec
  2   192.168.2.250   0 msec     0 msec    4 msec
```

实验问题与结果分析

本实验无需手动配置路由表，通过本次实验，加深了对 *Dijkstra* 算法的应用的理解，了解了动态路由的便利性，掌握了 RIP与OSPF 这两个不同的动态路由协议之间在算法、距离度量上的差异。

[15.1]期末自选实验ns3

姓名	学号	专业	分工
		软件工程	实验一代码
		软件工程	实验一整体设计+报告
		软件工程	实验二代码
		软件工程	实验二整体设计+报告

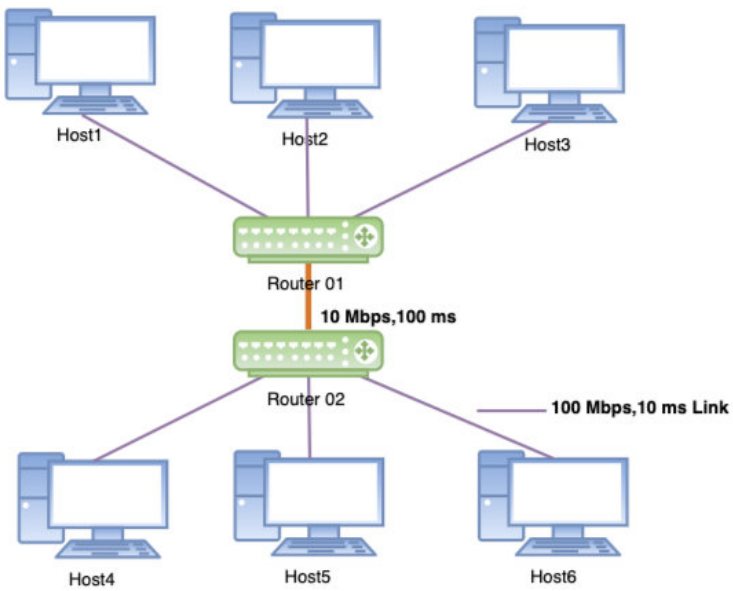
实验目的

1. 观察TCP、UDP共享瓶颈链路时如何相互影响的探究。
2. 探究缓冲区大小对TCP和UDP流的影响；

实验设备

仿真工具NS3

网络拓扑结构



实验原理

一、实验设计

1. **拓扑结构构建：**

   构建一个哑铃型网络拓扑结构，其中路由器R1和R2由 `10 Mbps, 100 ms` 的链接相连接。每个路由器都连接3个主机，即R1连接H1，H2，H3，R2连接H4，H5，H6。所有主机都通过 `100 Mbps, 10 ms` 的链接连接到路由器。

   两个路由器（即R1和R2）都使用 drop-tail 队列，并根据带宽延迟乘积设置相等的队列大小。选择1.5 KB的数据包大小。

2. **实验一**

   开启4个TCP New Reno流，过一会儿再开启2个基于UDP的CBR流，每个流为20 Mbps。这些流量随机分布在H1，H2和H3之间。将一个UDP流的速率提高到100 Mbps，并观察其对TCP流和另一个UDP流的吞吐量的影响。

3. **实验二**

   在10个数据包到800个数据包的范围内改变缓冲区大小，并重复上述实验以找出缓冲区大小对系统公平份额的影响。

二、实验原理

1. 计算机网络速率，带宽，吞吐量概念

   - 速率（DataRate）

     速率是指计算机网络中的主机在数字信道上，单位时间内从一端传送到另一端的数据量，即数据传输 率，也称数据率或比特率。比特(bit)是数据量的最小单位，s(秒)是时间的最小单位。所以速率单位为 bit/s或bps(bit persecond)

   - 带宽（Bandwidth）

   计算机网络中的主机在数字信道上，单位时间内从一端传送到另一端的最大数据量，即最大速率。

   - 吞吐量（Throughout）

     吞吐量是指对网络、设备、端口或其他设施在单位时间内成功地传送数据的数量（以比特、字节等测量 单位），也就是说吞吐量是指在没有帧丢失的情况下，设备能够接收并转发的最大数据速率。

2. TCP/UDP的接受缓冲区和发送缓冲区

   每个TCP socket在内核中都有一个发送缓冲区和一个接收缓冲区，TCP的全双工的工作模式以及TCP的

   流量(拥塞)控制便是依赖于这两个独立的缓冲区以及缓冲区的填充状态。

   - TCP接收缓冲区

     接收缓冲区把数据缓存入内核，应用进程一直没有调用recv()进行读取的话，此数据会一直缓存在相应 socket的接收缓冲区内。不管进程是否调用recv()读取socket，对端发来的数据都会经由内核接收并且 缓存到socket的内核接收缓冲区之中。

     > recv()，就是把内核缓冲区中的数据拷贝贝到应用层用户的buffer里面，并返回。

- TCP发送缓冲区

  进程调用send()发送的数据的时候，一般情况下，将数据拷贝贝进入socket的内核发送缓冲区之中，然后 send便会在上层返回。换句话说，send（）返回之时，数据不一定会发送到对端去（和write写文件有 点类似）。

  > send()，仅仅是把应用层buffer的数据拷贝贝进socket的内核发送buffer中，发送是TCP的事情，和 send其实没有太大关系。

  一般来说，接收缓冲区被TCP和UDP用来缓存网络上来的数据，一直被保存到应用进程读取为止。对于 TCP来说，如果应用进程一直没有对缓冲区进行读取，那么接收缓冲区满了以后，接收端会通知发送 端，接收窗口关闭，保证了TCP套接口接收缓冲区不会溢出，从而保证了TCP是可靠传输。

  发出TCP包的数量及每个包承载的数据的规模除了受到自身服务器配置和环境带宽影响，对端的接收状 态也会影响本端的发送状况。

- UDP接收缓冲区

  每个UDP socket都有一个接收缓冲区，没有发送缓冲区，字面上说就是UDP有数据就发，不管对方是否 可以正确接收。

  UDP当接收缓冲区满时，新来的数据无法进入缓冲区，此数据包被丢弃。UDP是没有流量控制的，快的 发送者可以很容易的就填满满的接收者，导致接收方的的UDP丢弃数据包

  UDP、TCP在缓冲区上的差异，亦是TCP可靠、UDP不可靠的表现。

3. 公平性度量（Fair Measure）

   公平性度量或量度用于网络工程中，以确定用户或应用程序是否获得了系统资源的公平共享。公平有几 个数学和概念上的定义。

   新的网络传输协议或相应应用程序的拥塞机制（Congestion control mechanism）必须与TCP良好交

   互。TCP公平性要求新协议接收的网络份额不得超过可比较的TCP流。这一点十分重要，因为TCP是因特 网上的主要传输协议，如果新协议获得不公平的容量，它们会引起诸如网络堵塞崩溃的问题。

   Jain平滑指数：公平性度量或量度用于网络工程中，以确定用户或应用程序是否获得了系统资源的公平共享。公平有几个数学和概念上的定义。

   使用**Jain平滑指数**分析流量变化，指示系统资源是否公平分配。

$$FairnessIndex = (sumFlow * sumFlow)/(6 * sumFlowSqr)$$

4. 为何使用吞吐量作为参考指标

   一般使用的参考指标：丢包率、抖动（jitter）和吞吐量。

   - 丢包率与数据包长度以及包发送频率相关。
   - 抖动是 QOS 里面常用的一个概念，其意思是指分组延迟的变化程度。
   - 吞吐量通常是对一个系统和它的部件处理传输数据请求能力的总体评价。

   为了更好地从整体观察实验结果，因此使用 吞吐量 作为参考指标来评估整个实验系统和它的部件处理传输数据请求能力。

5. ns3的PCAP Tracing

```
  AsciiTraceHelper ascii; //创建一个ASCII trace对象
  p2p.EnableAsciiAll(ascii.CreateFileStream ("script_buffersize.tr"));
 //script_buffersize.tr
  p2p.EnalbePcapAll("script_buffersize"); //script_buffersize.pcap
```

导出.cc 生成的抓包文件.pcap，使用 Wireshark 可视化

`<prefix>-<node>-<device>`，为模拟中的每个点对点设备创建一个tracing文件，但是数据繁杂，不易分析

6. gnuplot可视化

   用C++的标准输入输出FairnessIndex和吞吐量，并用gnuplot可视化结果并进行分析

实验内容

一、构建网络拓扑结构

```
// 创建一个节点容器以容纳8个节点
  NodeContainer nodes;
  nodes.Create (8);

// 为每个链接创建节点容器
  NodeContainer n0n3 = NodeContainer (nodes.Get (0), nodes.Get (3));
// h1r1
  NodeContainer n1n3 = NodeContainer (nodes.Get (1), nodes.Get (3));
// h2r1
  NodeContainer n2n3 = NodeContainer (nodes.Get (2), nodes.Get (3));
// h3r1
  NodeContainer n3n4 = NodeContainer (nodes.Get (3), nodes.Get (4));
// r1r2
  NodeContainer n4n5 = NodeContainer (nodes.Get (4), nodes.Get (5));
// r2h4
  NodeContainer n4n6 = NodeContainer (nodes.Get (4), nodes.Get (6));
// r2h5
  NodeContainer n4n7 = NodeContainer (nodes.Get (4), nodes.Get (7));
// r2h6

// 在节点上安装Internet协议栈（协议）
  InternetStackHelper internet;
  internet.Install (nodes);

// 在节点之间创建点对点通道
  PointToPointHelper p2p;
// 主机到路由器的链接
  p2p.SetDeviceAttribute ("DataRate", StringValue ("100Mbps"));
  p2p.SetChannelAttribute ("Delay", StringValue ("10ms"));
  NetDeviceContainer d0d3 = p2p.Install (n0n3);
  NetDeviceContainer d1d3 = p2p.Install (n1n3);
  NetDeviceContainer d2d3 = p2p.Install (n2n3);
  NetDeviceContainer d4d5 = p2p.Install (n4n5);
```

```
  NetDeviceContainer d4d6 = p2p.Install (n4n6);
  NetDeviceContainer d4d7 = p2p.Install (n4n7);
// //路由器到路由器的链接
  p2p.SetQueue ("ns3::DropTailQueue", "MaxPackets",
UintegerValue(100000*10/packetSize));
  p2p.SetDeviceAttribute ("DataRate", StringValue ("10Mbps"));
  p2p.SetChannelAttribute ("Delay", StringValue ("100ms"));
  NetDeviceContainer d3d4 = p2p.Install (n3n4);

// 为每个接口分配IP地址
  Ipv4AddressHelper ipv4;
  ipv4.SetBase ("10.1.1.0", "255.255.255.0");
  Ipv4InterfaceContainer i0i3 = ipv4.Assign (d0d3);

  ipv4.SetBase ("10.1.2.0", "255.255.255.0");
  Ipv4InterfaceContainer i1i3 = ipv4.Assign (d1d3);

  ipv4.SetBase ("10.1.3.0", "255.255.255.0");
  Ipv4InterfaceContainer i2i3 = ipv4.Assign (d2d3);

  ipv4.SetBase ("10.1.4.0", "255.255.255.0");
  Ipv4InterfaceContainer i3i4 = ipv4.Assign (d3d4);

  ipv4.SetBase ("10.1.5.0", "255.255.255.0");
  Ipv4InterfaceContainer i4i5 = ipv4.Assign (d4d5);

  ipv4.SetBase ("10.1.6.0", "255.255.255.0");
  Ipv4InterfaceContainer i4i6 = ipv4.Assign (d4d6);

  ipv4.SetBase ("10.1.7.0", "255.255.255.0");
  Ipv4InterfaceContainer i4i7 = ipv4.Assign (d4d7);

  Ipv4GlobalRoutingHelper::PopulateRoutingTables ();
```

二、变量和统计指标

1. 设置数据传输速率：

```
void IncRate (Ptr<MyApp> app, DataRate rate){
  app->ChangeRate(rate);
    return;
}
```

2. 每秒计算吞吐量和计算Jain平滑指数

```
void Throughput (FlowMonitorHelper *fmhelper, Ptr<FlowMonitor> flowMon){
  Ptr<Ipv4FlowClassifier> classifier = DynamicCast<Ipv4FlowClassifier> (fmhelper-
>GetClassifier ());
  std::map<FlowId, FlowMonitor::FlowStats> stats = flowMon->GetFlowStats ();
  double sumFlow = 0, sumFlowSqr = 0, sumFlowTCP = 0;
  for (std::map<FlowId, FlowMonitor::FlowStats>::const_iterator i = stats.begin (); i
!= stats.end (); ++i){
```

```
    Ipv4FlowClassifier::FiveTuple t = classifier->FindFlow (i->first);
    // 计算吞吐量
    double TPut = i->second.rxBytes * 8.0 / (i->second.timeLastRxPacket.GetSeconds() -
i->second.timeFirstTxPacket.GetSeconds())/1000000;
    sumFlow += TPut;
    sumFlowSqr += TPut * TPut ;
    if(t.sourceAddress == "10.1.1.1" && t.destinationAddress == "10.1.7.2"){
      sumFlowTCP += TPut;
    }
    else if(t.sourceAddress == "10.1.1.1" && t.destinationAddress == "10.1.2.1"){
      sumFlowTCP += TPut;
    }
    else if(t.sourceAddress == "10.1.6.2" && t.destinationAddress == "10.1.7.2"){
      sumFlowTCP += TPut;
    }
    else if(t.sourceAddress == "10.1.5.2" && t.destinationAddress == "10.1.3.1"){
      sumFlowTCP += TPut;
    }
  }
  // 计算Jain平滑指数
  double FairnessIndex = (sumFlow * sumFlow)/ (6 * sumFlowSqr) ;
  dataset.Add ((double)Simulator::Now().GetSeconds(), FairnessIndex);
  if(sumFlowTCP!=0)
    datasetTcp.Add ((double)Simulator::Now().GetSeconds(), sumFlowTCP);
  if(sumFlow-sumFlowTCP!=0)
    datasetUdp.Add ((double)Simulator::Now().GetSeconds(), sumFlow - sumFlowTCP);
  std :: cout << " FairnessIndex: " << FairnessIndex << std :: endl;
  Simulator::Schedule(Seconds(1),&Throughput, fmhelper, flowMon);
}
```

三、实验一

1. 开启4个TCP New Reno流，每个流为20 Mbps

   4个TCP流：h1-h6、h4-h3、h1-h2、h5-h6

   以h1-h6为例：

```
// TCP流，h1 - h6
  uint16_t port1 = 8081;

// 创建接收器sink
  Address sinkAdd1 (InetSocketAddress (i4i7.GetAddress (1), port1));
  PacketSinkHelper packetSink1 ("ns3::TcpSocketFactory", InetSocketAddress
(Ipv4Address::GetAny (), port1));
// 使用 PacketSinkHelper 的接收器应用程序，该应用程序使用port1的所有数据包
  ApplicationContainer sinkApps1 = packetSink1.Install (nodes.Get (7));
  sinkApps1.Start (Seconds (start_time_sink));
  sinkApps1.Stop (Seconds (run_time));

// 为发送方应用程序创建一个TCP Socket
  Ptr<Socket> tcpSocket1 = Socket::CreateSocket (nodes.Get (0),
TcpSocketFactory::GetTypeId ());
```

```
  tcpSocket1->SetAttribute("SndBufSize",  ns3::UintegerValue(bufferSize));
  tcpSocket1->SetAttribute("RcvBufSize",  ns3::UintegerValue(bufferSize));

// 创建发送者应用程序
  Ptr<MyApp> tcp_Agent1 = CreateObject<MyApp> ();
  tcp_Agent1->Setup (tcpSocket1, sinkAdd1, packetSize, packetsToSend, DataRate
("20Mbps"));
  nodes.Get (0)->AddApplication (tcp_Agent1);
  tcp_Agent1->SetStartTime (Seconds (start_time_apps));
  tcp_Agent1->SetStopTime (Seconds (run_time));
```

2. 开启2个基于UDP的CBR流，每个流为20 Mbps

   2个UDP流：h2-h3、h4-h5

   以h2-h3为例

```
//UDP流，h2-h3
  uint16_t port5 = 8085;
// 创建接收器sink
  Address sinkAdd5 (InetSocketAddress (i2i3.GetAddress (0), port5));
  PacketSinkHelper packetSink5 ("ns3::UdpSocketFactory", InetSocketAddress
(Ipv4Address::GetAny (), port5));
// 创建接收器应用程序，使用来自上述端口的所有数据包
  ApplicationContainer sinkApps5 = packetSink5.Install (nodes.Get (2));
  sinkApps5.Start (Seconds (start_time_sink));
  sinkApps5.Stop (Seconds (run_time));

// 创建UDP Socket
  Ptr<Socket> ns3UdpSocket5 = Socket::CreateSocket (nodes.Get (1),
UdpSocketFactory::GetTypeId ());
  ns3UdpSocket5->SetAttribute("RcvBufSize",  ns3::UintegerValue(bufferSize));

// 创建发送者应用程序
  Ptr<MyApp> udp_Agent1 = CreateObject<MyApp> ();
  udp_Agent1->Setup (ns3UdpSocket5, sinkAdd5, packetSize, packetsToSend, DataRate
("20Mbps"));
  nodes.Get (1)->AddApplication (udp_Agent1);
  udp_Agent1->SetStartTime (Seconds (start_time_apps));
  udp_Agent1->SetStopTime (Seconds (run_time));
```

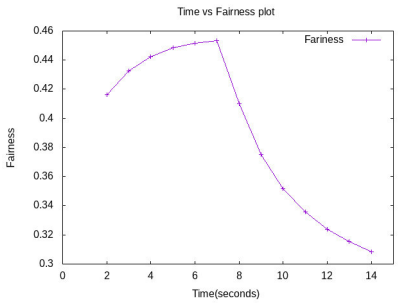
3. 在第7秒时将其中一个UDP流（h2-h3）的速率从 20 Mbps 提高到 100 Mbps

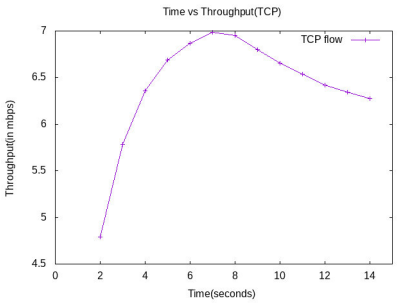
```
 Simulator::Schedule (Seconds(7.0), &IncRate, udp_Agent1, DataRate("100Mbps"));
```

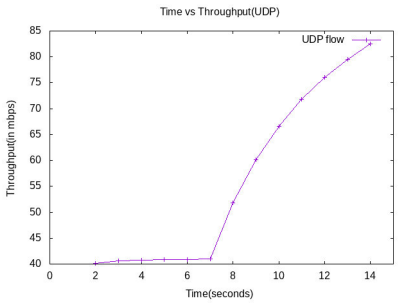
4. 实验结果

   实验表明：第7秒时，其中一个UDP流的速率从 20 Mbps 提高到 100 Mbps，用于统计系统带宽分配份额的公平性的Jain平滑指数急剧下 降，TCP New Reno流的吞吐量小幅度下降，UDP流急剧上升。

   共享瓶颈链路时，UDP流速率变化在影响自身吞吐量的同时，也会影响同一TCP吞吐量，且对整个网络的系统资源公平共享分配造成影响。







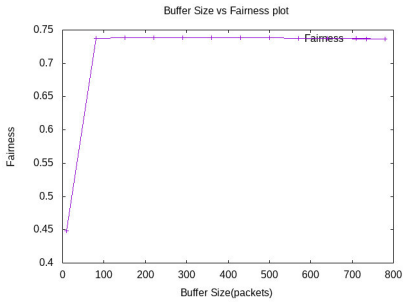
四、实验二

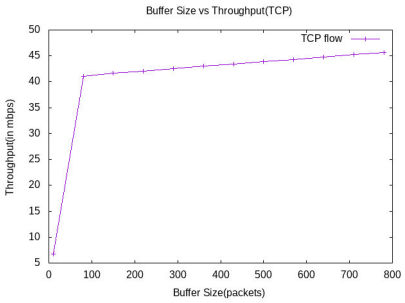
1. 在10个数据包到800个数据包的范围内改变缓冲区大小，重复实验一步骤

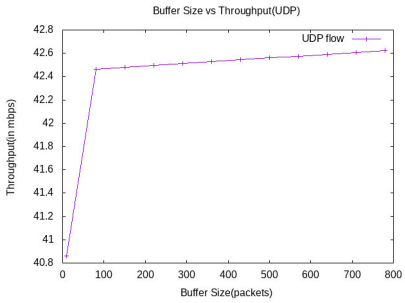
```
for(bufferSize=10*packetSize;bufferSize<=800*packetSize;){
  ...
}
```

2. 实验结果

   实验表明：在10个数据包到800个数据包的范围内改变缓冲区大小，缓冲区较小时对系统带宽分配公平性、TCP、UDP的影响较大，随着缓冲区增大而分配公平性趋于稳定值，TCP、UDP的吞吐量缓慢增加。







实验总结

做实验的时候一开始小组内各个成员ns3下载的版本差别有点大，操作系统也各不相同，所以一开始有些成员写的东西在另外一个成员电脑里试测的时候运行失败，版本问题一定要多注意，尤其是小组项目，运行环境之类的要尽量保持一致。

两个实验分别表明在此网络拓扑结构下，共享瓶颈链路时，UDP流速率变化在影响自身吞吐量的同时，也会影响同一TCP吞吐量，且对整个网络的系统资源公平共享分配造成影响。同时，改变缓冲区大小可以观测出，缓冲区较小时对系统带宽分配公平性、TCP、UDP的影响较大，随着缓冲区增大而分配公平性趋于稳定值，TCP、UDP的吞吐量缓慢增加。

实验设计中要注意一些基本的设计规范，比如控制变量，比如可映射到真实环境中，具有一定的实验价值。如TCP流和UDP流在现实的网络中是经常同时出现并相互影响的，所以实验模拟两者相互影响的现象，而现实生活中缓冲区大小也是对两种流吞吐量和带宽份额分配公平性都会产生一定影响。

一学期的计算机网络实验至此就结束了，回顾整个学期，从一开始学习一些基本操作，再到通过实验熟悉各种路由器配置并模拟一些重要的网络拓扑结构和算法等，最后再结合计算机网络理论课和实验课学到的知识设计一个有意义有价值的实验，层层递进，步步深入，在整个过程中收获良多。不论是个人实验还是小组合作，都要扎实掌握理论知识，而后运用实践，遇到难题要善于思考，敏而好学，从而学到更多东西。