
目 录

实验一 SOCKET 编程实验	1
1.1 环境	1
1.2 系统功能需求	2
1.3 系统设计	2
1.4 系统实现	4
1.5 系统测试及结果说明	8
1.6 其它需要说明的问题	14
实验二 数据可靠传输协议设计实验	15
2.1 环境	15
2.2 实验要求	15
2.3 协议的设计、验证及结果分析	16
2.4 其它需要说明的问题	26
实验三 基于 GPT 的组网实验	27
3.1 环境	27
3.2 实验要求	27
3.3 基本部分实验步骤说明及结果分析	27
3.4 综合部分实验设计、实验步骤及结果分析	48
3.5 其它需要说明的问题	60
心得体会与建议	61
4.1 心得体会	61
4.2 建议	62

实验一 Socket 编程实验

1.1 环境

1.1.1 开发平台

硬件配置：

处理器： Intel(R)Core(TM)i5-7200U CPU @ 2.50GHz(4 CPUs), ~2.7GHz

内存： 8192MB RAM

显卡： Intel(R) HD Graphics 620

系统软件组件：

Windows 10 64 位操作系统

Windows SDK 10.0.14393.0

开发平台：

Microsoft Visual Studio 2017 Community

第三方组件：

MFC

1.1.2 运行平台

硬件配置：

处理器： Intel(R)Core(TM)i5-7200U CPU @ 2.50GHz(4 CPUs), ~2.7GHz（可能最低运行要求会更低）

内存： 8192MB RAM（可能最低运行要求会更低）

显卡： Intel(R) HD Graphics 620（可能最低运行要求会更低）

系统软件组件：

Windows 操作系统

Windows SDK 10.0.14393.0

第三方组件：

MFC

1.2 系统功能需求

题目：

编写一个支持多线程处理的 Web 服务器软件，要求如下：

第一级：

- ✧ 可配置 Web 服务器的监听地址、监听端口和主目录。
- ✧ 能够单线程处理一个请求。当一个客户（浏览器,输入 URL: `http://127.0.0.1/index.html`）连接时创建一个连接套接字；
- ✧ 从连接套接字接收 http 请求报文，并根据请求报文的确定用户请求的网页文件；
- ✧ 从服务器的文件系统获得请求的文件。 创建一个由请求的文件组成的 http 响应报文。（报文包含状态行+实体体）。
- ✧ 经 TCP 连接向请求的浏览器发送响应，浏览器可以正确显示网页的内容；
- ✧ 服务可以启动和关闭。

第二级：

- ✧ 支持多线程，能够针对每一个新的请求创建新的线程，每个客户请求启动一个线程为该客户服务；
- ✧ 在服务器端的屏幕上输出每一个请求的来源（IP 地址、端口号和 HTTP 请求命令行）
- ✧ 支持一定的异常情况处理能力。

第三级：

- ✧ 能够传输包含多媒体（如图片）的网页给客户端，并能在客户端正确显示；
- ✧ 对于无法成功定位文件的请求，根据错误原因，作相应错误提示。
- ✧ 在服务器端的屏幕上能够输出对每一个请求处理的结果。
- ✧ 具备完成所需功能的基本图形用户界面（GUI），并具友好性

1.3 系统设计

系统架构

该系统由自己编写的有 Web 服务器，完成此系统还需要的部分为客户端，此客户端为用户自己安装的浏览器，包括 Chrome、Firefox 等等。在开发者自己的电脑上使用的是 Chrome 作为客户端。

Web 服务器的功能有：可配置 Web 服务器的监听地址、监听端口和主目录，能够多线程处理用户请求，能够创建响应报文并且正确在浏览器上显示内容，支持一定情况的异常处理，具备基本图形用户界面等。

该服务器与客户端的接口为一个监听端口，在浏览器上输入服务器设置的监听端口，那么客户端的请求就能被服务器响应。

功能模块划分

本实验在老师所给的实验框架下进行开发，因此有一部分模块功能由原有代码实现，因此省略了一些必要的细节。本 Web 服务器功能模块划分为：

- 1.初始化 Winsock
- 2.初始化服务器
- 3.监听客户请求
- 4.得到客户端的 IP 和端口号
- 5.生成会话 socket
- 6.接收与转发客户端的数据

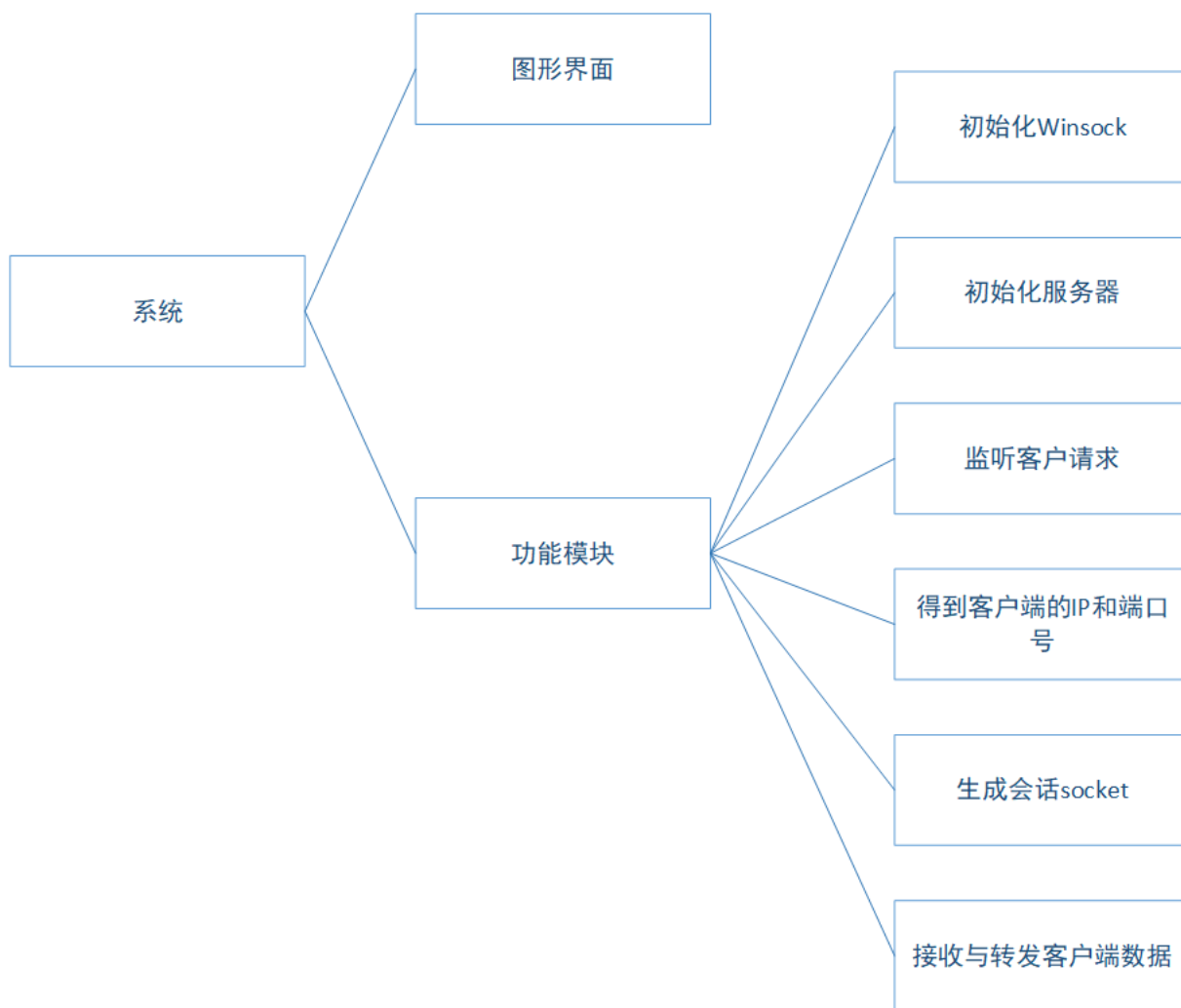


图 1-1 功能模块

在初始化服务器模块中，实现了创建 socket，设置服务器的 ip 地址和端口号，然后将它们与 socket 绑定。

监听客户请求模块实现了服务器等待连接状态。

在产生会话 socket 模块中，实现了创建响应报文，支持一定情况的异常处理，输出要求的信息等。

1.4 系统实现

1.初始化 Winsock

技术关键及解决方法：

直接调用 `WSAStartup` 初始化 Winsock，参数有两个，分别是 Winsock 版本号和用于返回 Winsock 的环境信息。如果 `WSAStartup` 函数的返回值非 0，则表明初始化失败，打印错误信息。若为 0，则判断返回的 Winsock 版本号，如果版本不对，则清除 Winsock，打印错误信息。

2.初始化服务器

技术关键及解决方法：

首先创建 socket，如果创建失败，则清除 Winsock。如果成功了，则进入下一步。然后设置服务器的 ip 地址和端口号。然后利用 `bind` 函数绑定 socket 与服务器的 ip 和端口，如果绑定不成功，则关闭 socket，清除 Winsock，输出错误信息。

流程图：

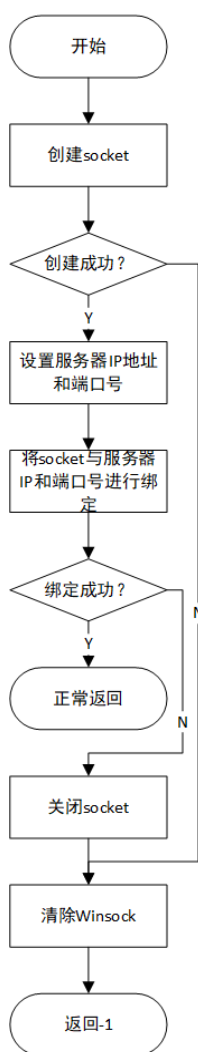


图 1-2 初始化服务器流程图

3.监听客户请求

技术关键及解决方法:

直接调用 `listen` 函数实现功能, 有两个参数, 分别为服务器 `socket` 以及最大连接数。如果创建失败, 则关闭 `socket`, 清除 `Winsock`。

4.得到客户端的 IP 和端口号

技术关键及解决方法:

直接调用 `getsockname` 函数实现功能, 有三个参数, 分别为 `socket`、客户端地址、客户端地址所占长度。如果函数返回值正常, 则输出客户端的 `ip` 地址和端口号。返回客户端地址。

5.生成会话 `socket`

技术关键及解决方法:

首先利用 `accept` 函数产生会话 `socket`, 有三个参数, 分别为服务器 `socket`, 客户端地址、客户端地址所占长度。

然后就是很重要的一部分——创建响应报文了, 首先调用 `recv` 函数接受数据, 如果接受数据错误, 则删除产生错误的会话 `socket`。然后我们要从请求报文中获取文件的目录, 这是比较关键的地方, 根据请求报文的格式, 可以看出在 `GET` /后跟的就是所请求文件的目录, 而且在 `HTTP` 前结束, 根据这个报文格式, 我们只需要截取中间一段就可以得到目录了。之后我们就要根据文件类型创建相应的响应报文了, 对于不同的文件类型, 响应报文不同, 主要是 `Content-type` 这一项不同, 在网上查阅资料后, 可以得到不同类型文件的 `Content-type`。如果找不到文件, 响应报文就为 "`HTTP/1.1 404 NotFound\r\nContent-Type: text/html\r\n\r\n`", 表示文件不存在。之后我们根据要求, 输出对应的信息即可。

流程图见图 1-3。

6.接收与转发客户端的数据

技术关键及解决方法:

首先调用 `ioctlsocket` 函数, 有三个参数, 分别为服务器 `socket`, 模式 (`FIONBIO`: 允许或禁止套接口 `s` 的非阻塞模式), 阻塞方式。

然后就是调用 `FD_ZERO` 和 `FD_SET` 函数, 将服务器 `socket` 加入到 `rfd`s, 等待用户的连接请求。用 `select` 函数判断是否处于等待用户连接请求或用户数据到来或会话 `socket` 可发送数据的状态。`select` 函数返回有可读或可写的 `socket` 的总数, 保存在 `rtn` 里。最后一个参数设定等待时间, 如为 `NULL` 则为阻塞模式。

流程图见图 1-4。

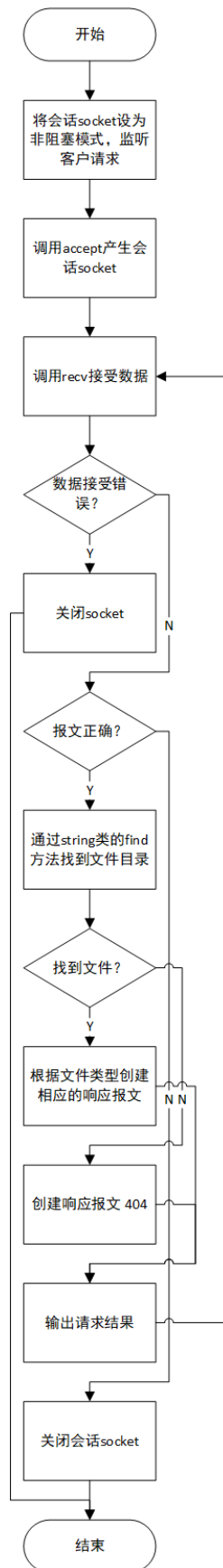


图 1-3 生成会话 socket 流程图

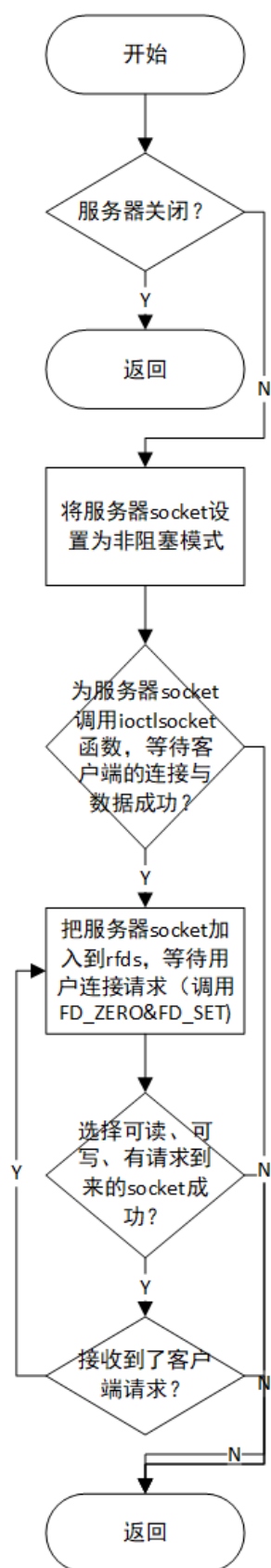


图 1-4 接收与转发客户端的数据流程图

1.5 系统测试及结果说明

硬件环境：

处理器： Intel(R)Core(TM)i5-7200U CPU @ 2.50GHz(4 CPUs), ~2.7GHz
内存： 8192MB RAM
显卡： Intel(R) HD Graphics 620

系统软件组件：

Windows 10 64 位操作系统
Windows SDK 10.0.14393.0

系统测试的结果：

第一级：

✧ 可配置 Web 服务器的监听地址、监听端口和主目录。

如下图，可以配置服务器的监听地址、监听端口和主目录，符合要求

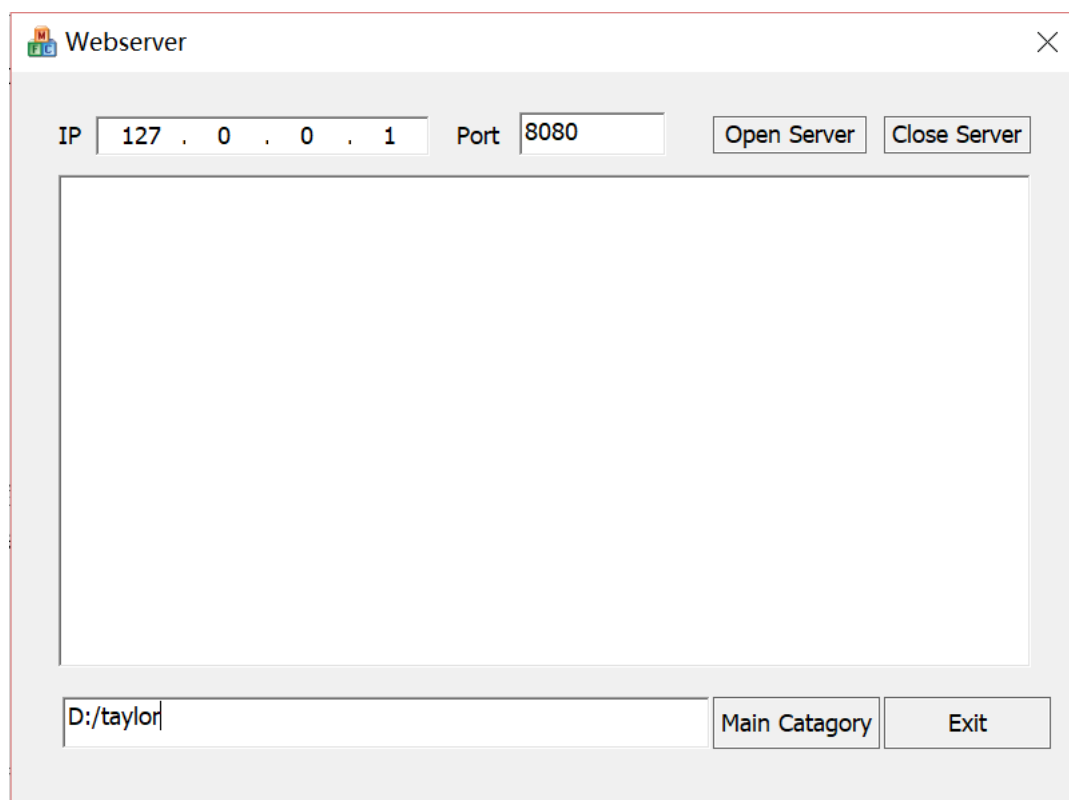


图 1-5 配置 Web 服务器的监听地址、监听端口和主目录

✧ 能够单线程处理一个请求。当一个客户（浏览器,输入 URL: <http://127.0.0.1/index.html>）连接时创建一个连接套接字，从连接套接字接收 http 请求报文，并根据请求报文的确定用户请求的网页文件，从服务器的文件系统获得请求的文件。 创建一个由请求的文

件组成的 http 响应报文（报文包含状态行+实体体），经 TCP 连接向请求的浏览器发送响应，浏览器可以正确显示网页的内容。

如下图，服务器能够单线程处理一个请求，并且能根据请求报文确定用户请求的网页文件，并且生成响应报文。浏览器也可以正常显示网页。

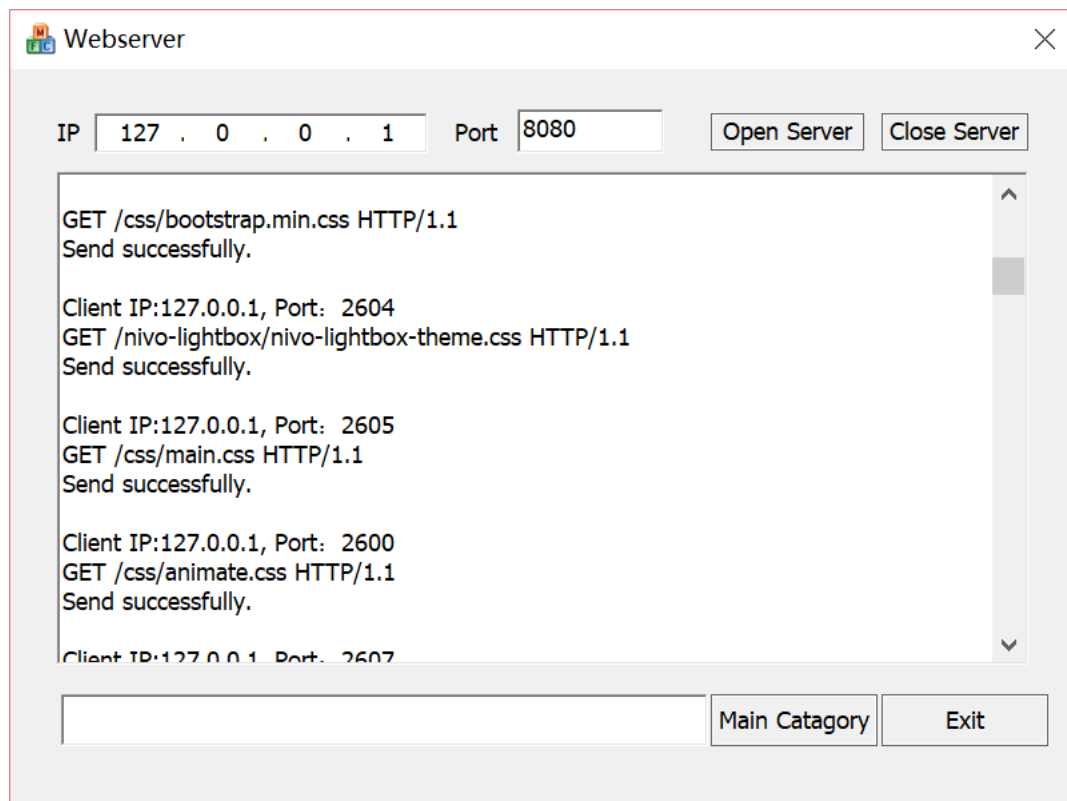


图 1-6 响应报文

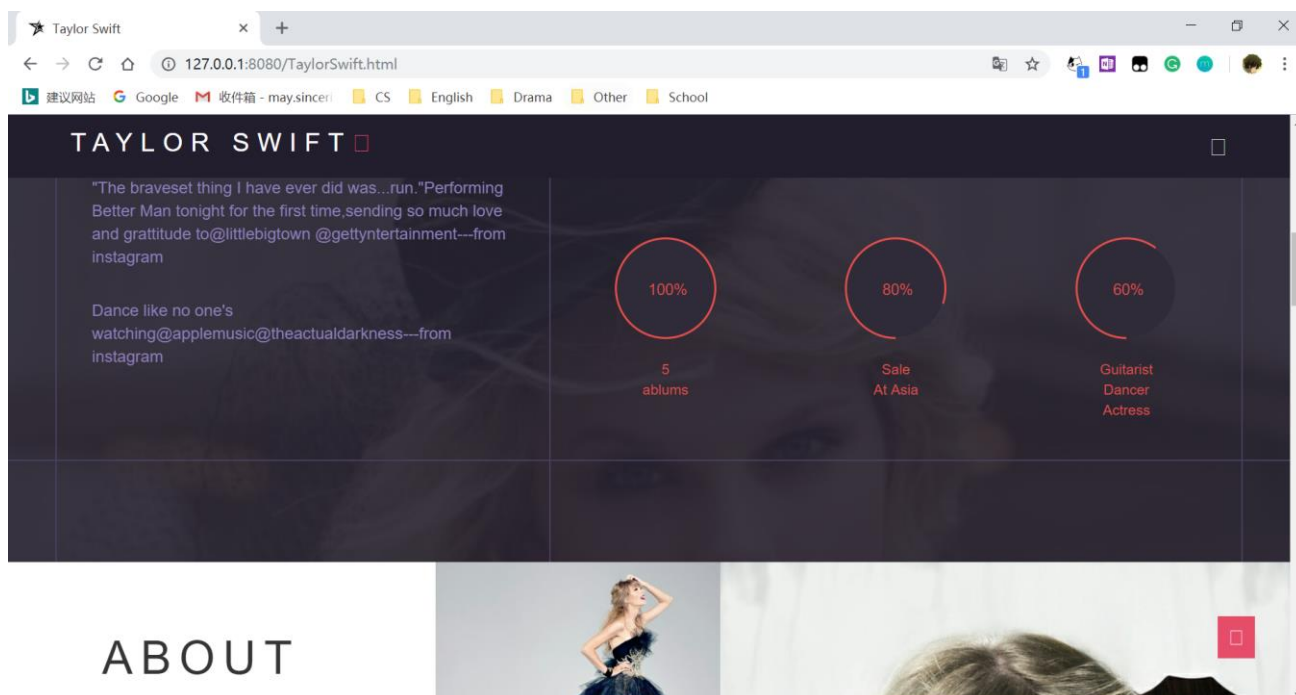


图 1-7 显示网页

✧ 服务可以启动和关闭。

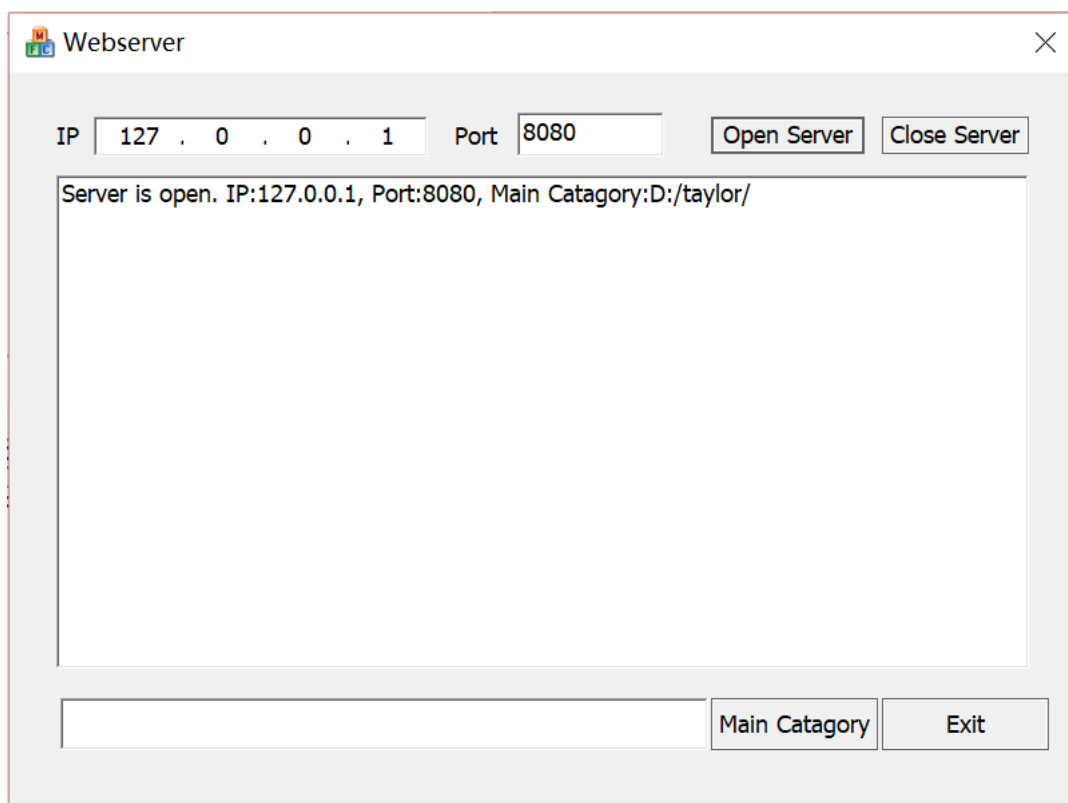


图 1-8 服务器开启

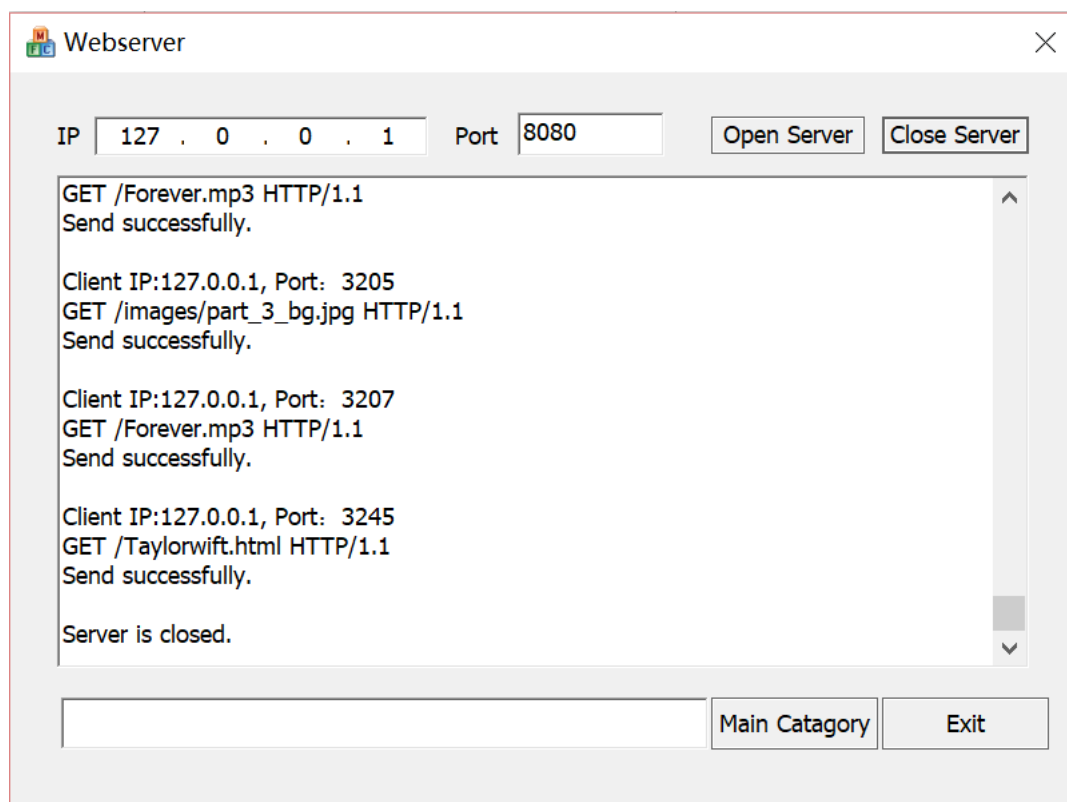


图 1-9 服务器关闭

第二级:

- ✧ 支持多线程，能够针对每一个新的请求创建新的线程，每个客户请求启动一个线程为该客户服务；

如下图，服务器支持多线程，能看到两个客户同时请求网页。

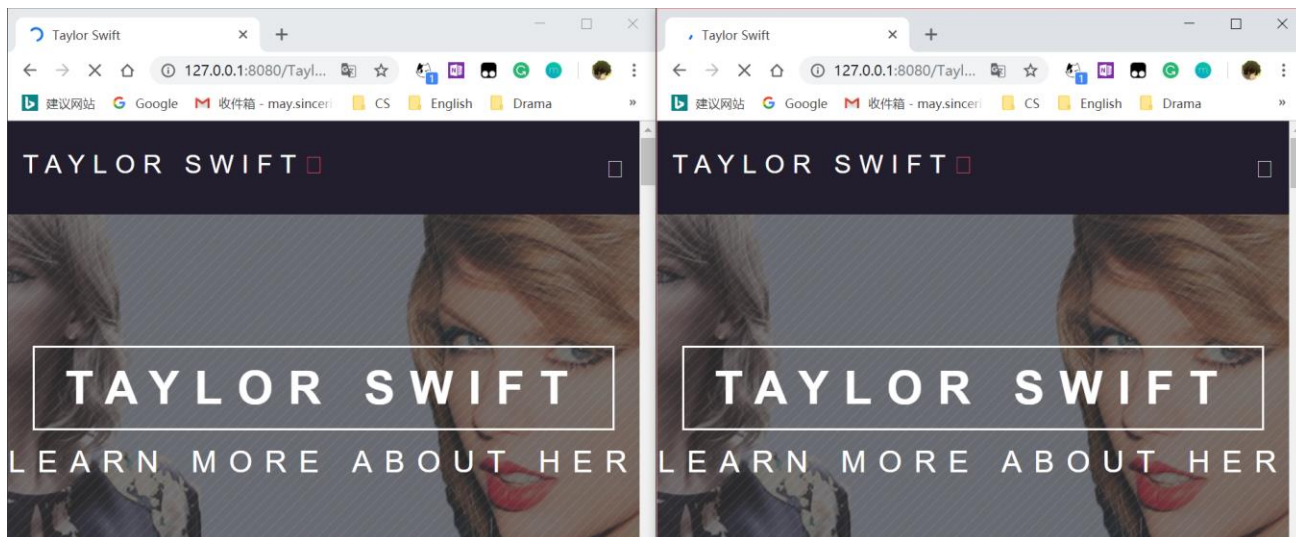


图 1-10 多线程请求

- ✧ 在服务器端的屏幕上输出每一个请求的来源（IP 地址、端口号和 HTTP 请求命令行）如下图，正确显示请求的来源。

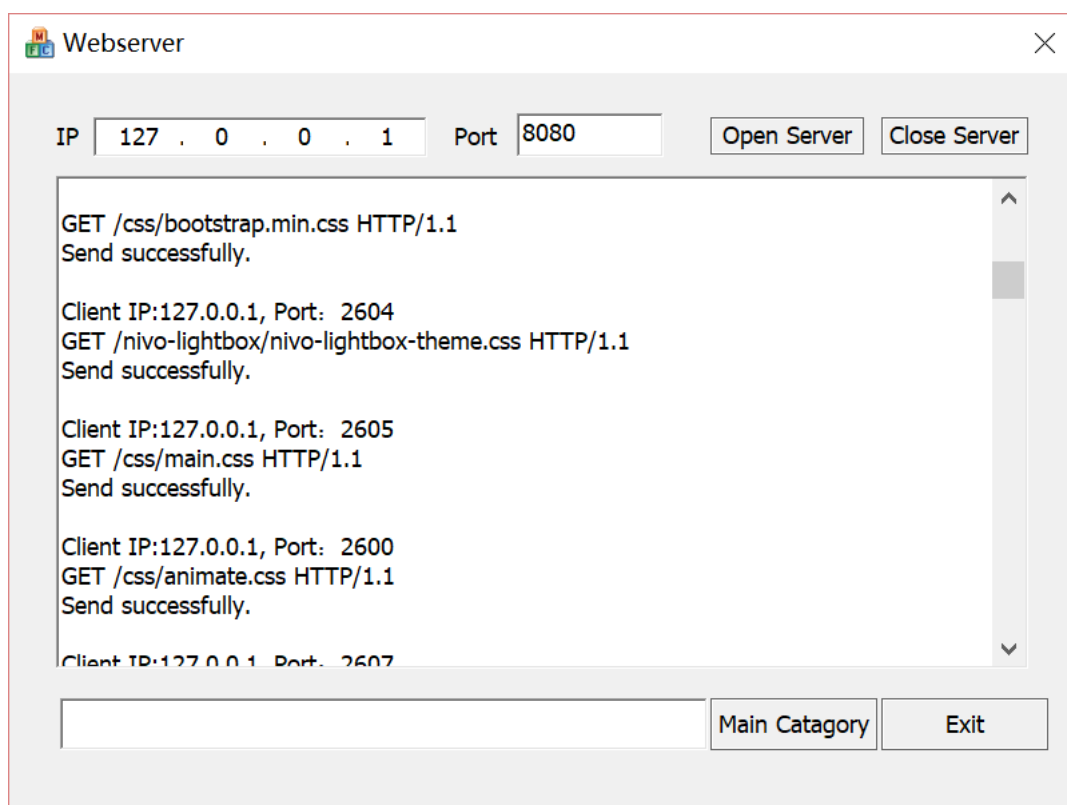


图 1-11 显示请求的来源

✧ 支持一定的异常情况处理能力。

如下图，说明服务器支持一定的异常处理能力，给出提示。

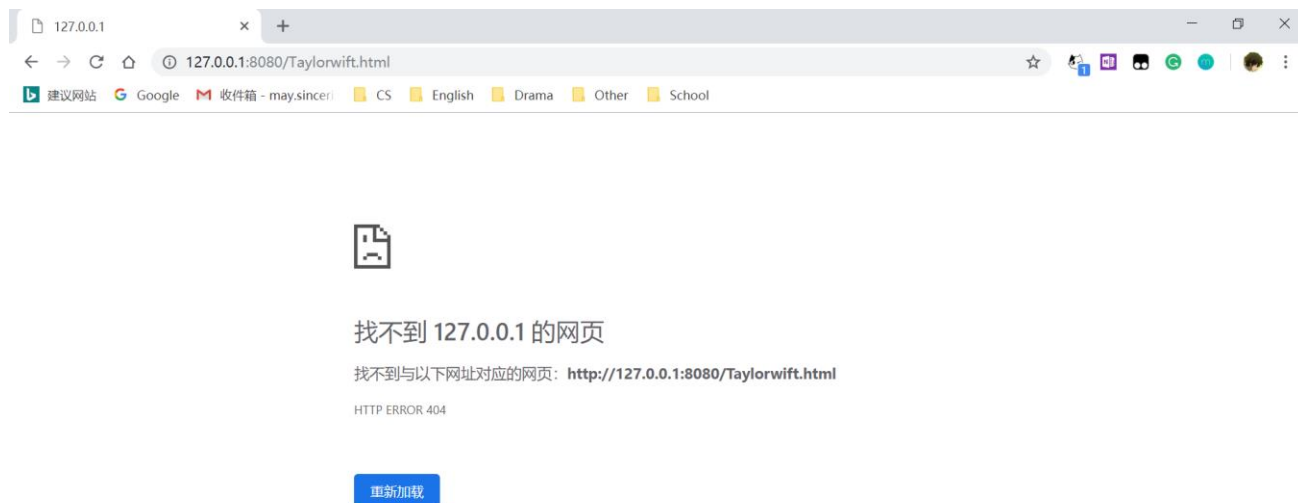


图 1-12 异常处理

第三级:

✧ 能够传输包含多媒体（如图片）的网页给客户端，并能在客户端正确显示；

如下，本服务器支持多种文件类型传输，如 jpg、gif、txt、png、ico、mp3、js、css 等。

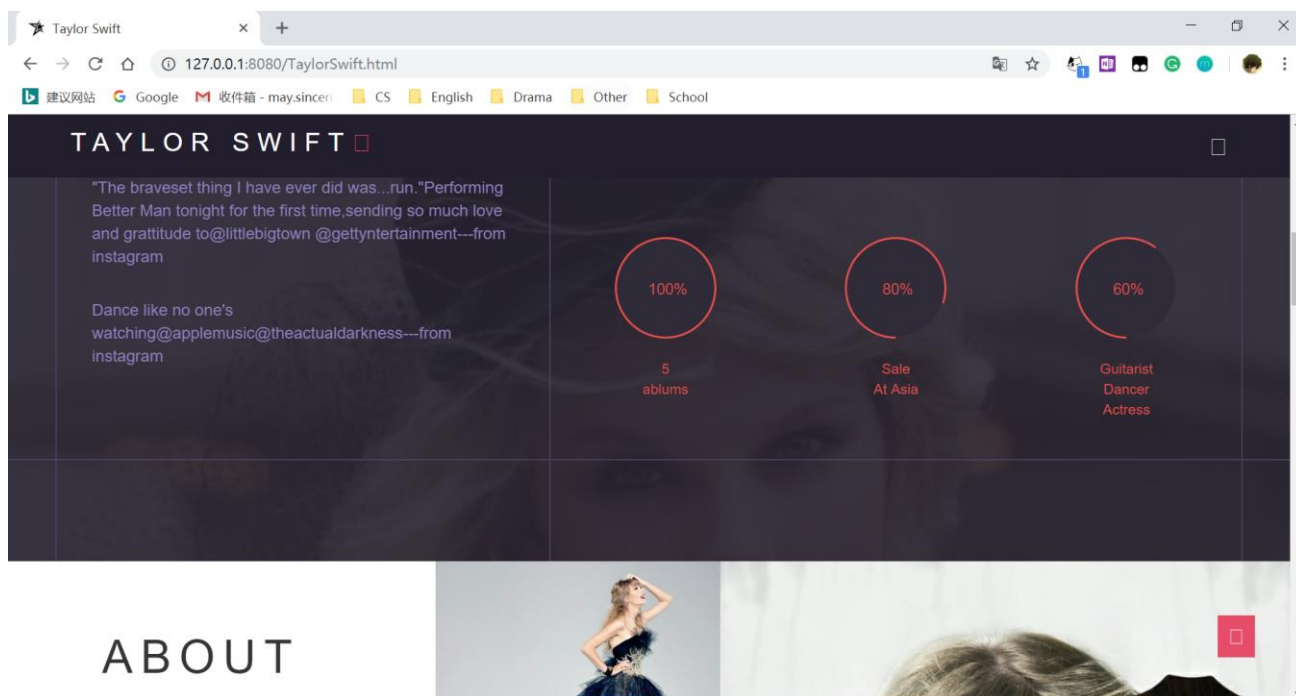


图 1-13 传输多媒体

- ✧ 对于无法成功定位文件的请求，根据错误原因，作相应错误提示。
当请求的文件不存在时，给出 404 提示。

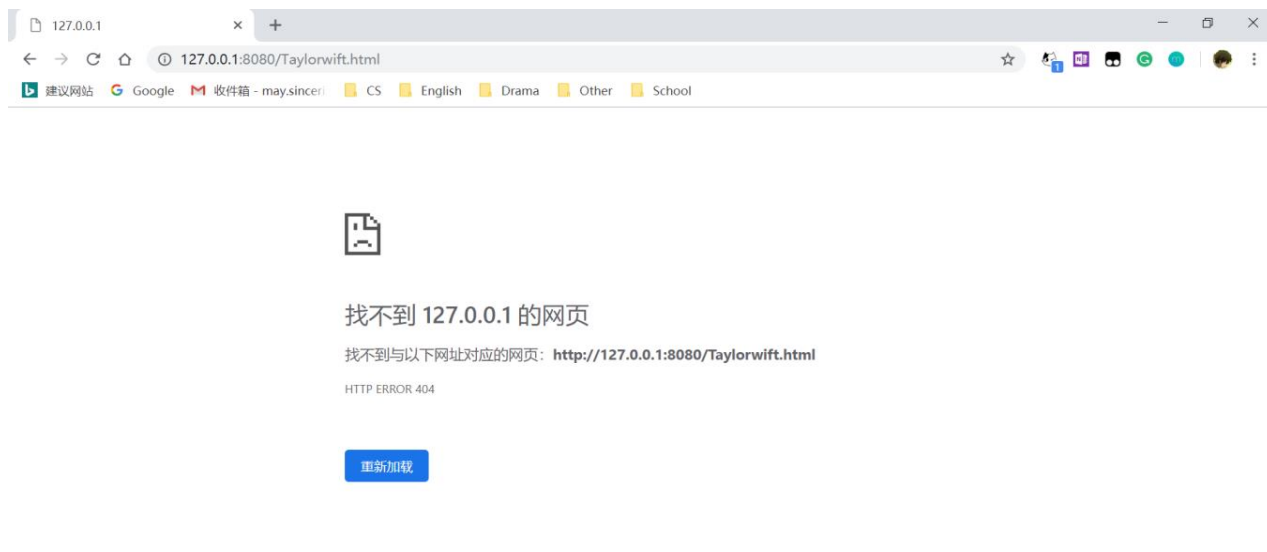


图 1-14 无法成功定位文件时

- ✧ 在服务器端的屏幕上能够输出对每一个请求处理的结果。
如下可以看到对每一个请求处理的结果。

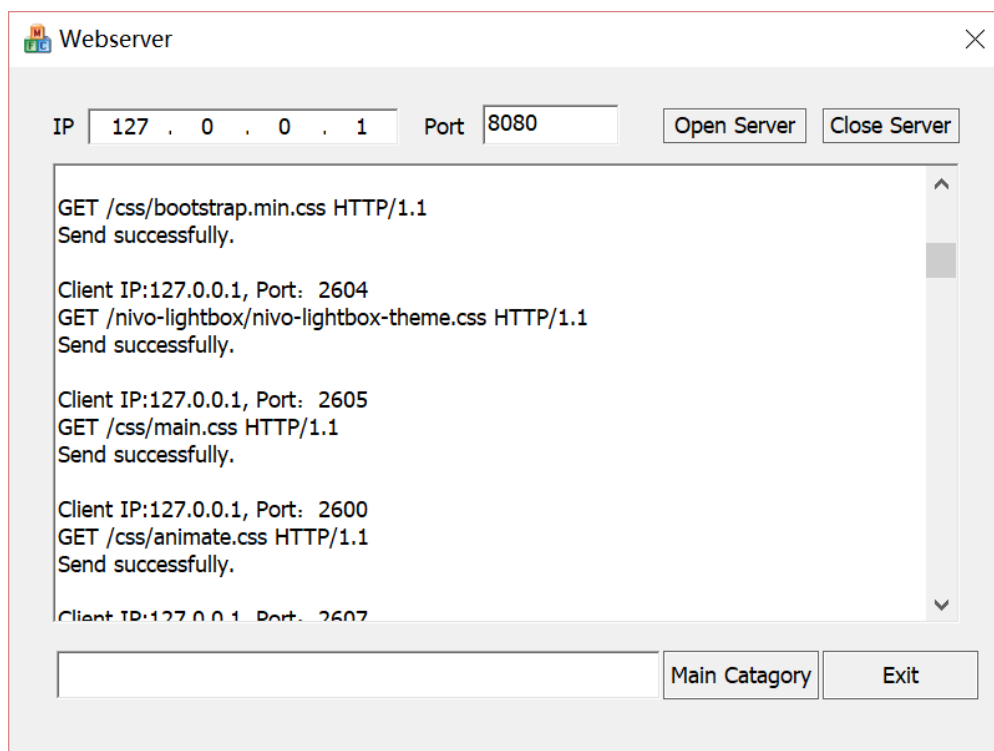


图 1-15 对每个请求的结果

✧ 具备完成所需功能的基本图形用户界面（GUI），并具友好性如下，图形界面在实验框架下修改得来。

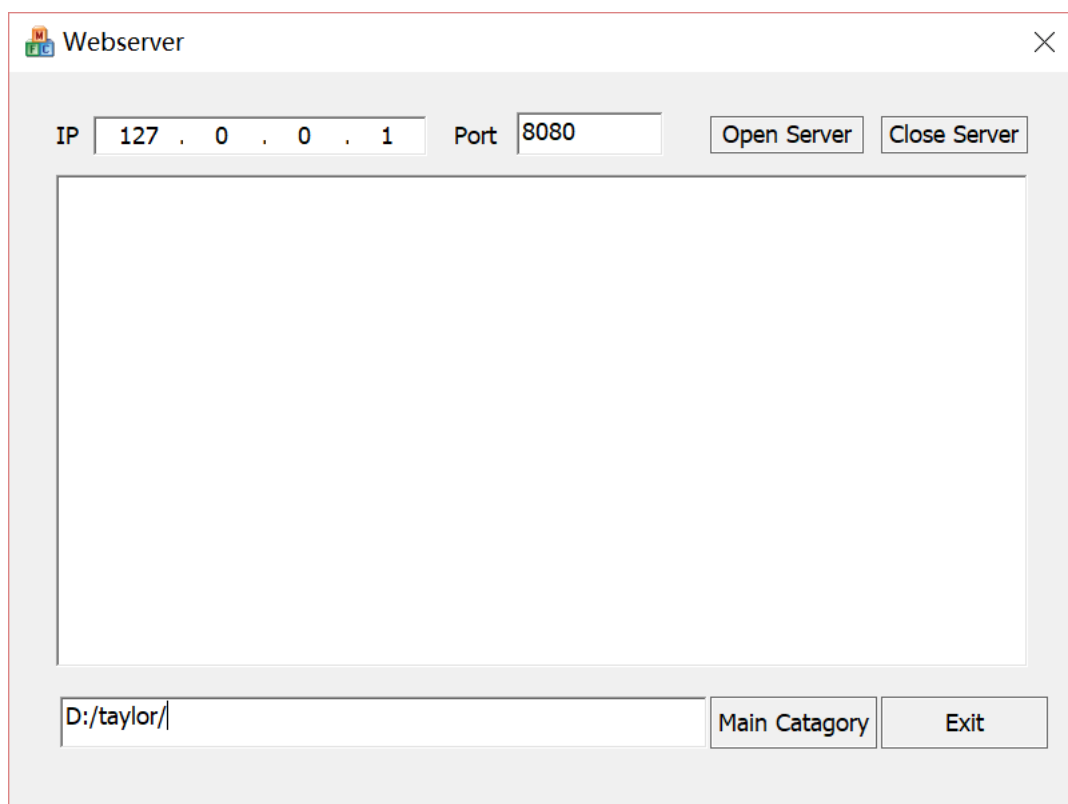


图 1-16 图形界面

1.6 其它需要说明的问题

本次实验所用的图形界面由实验框架修改而来，因此很少给出实现。并且本次实验完成时间不多，所完成的服务器能支持的功能还不是非常多，异常处理能力也不是特别强，如果时间充足应该可以做的更加完善。

实验二 数据可靠传输协议设计实验

2.1 环境

硬件配置：

处理器： Intel(R)Core(TM)i5-7200U CPU @ 2.50GHz(4 CPUs), ~2.7GHz
内存： 8192MB RAM
显卡： Intel(R) HD Graphics 620

系统软件组件：

Windows 10 64 位操作系统
Windows SDK 10.0.14393.0

第三方软件：

Microsoft Visual Studio 2017 Community

2.2 实验要求

1.可靠运输层协议实验只考虑单向传输，即：只有发送方发生数据报文，接收方仅仅接收报文并给出确认报文。

2.要求实现具体协议时，指定编码报文序号的二进制位数(例如 3 位二进制编码报文序号)以及窗口大小(例如大小为 4)，报文段序号必须按照指定的二进制位数进行编码。

3.代码实现不需要基于 Socket API，不需要利用多线程，不需要任何 UI 界面。

本实验包括三个级别的内容，具体包括：

1.实现基于 GBN 的可靠传输协议，分值为 50%。

2.实现基于 SR 的可靠传输协议，分值为 30%。

3.在实现 GBN 协议的基础上，根据 TCP 的可靠数据传输机制(包括超时后只重传最早发送且没被确认的报文、快速重传)实现一个简化版的 TCP 协议。报文段格式、报文段 序号编码方式和 GBN 协议一样保持不变，不考虑流量控制、拥塞控制，不需要估算 RTT 动态调整定时器 Timeout 参数。分值 20%。

2.3 协议的设计、验证及结果分析

2.3.1 GBN 协议的设计、验证及结果分析

设计:

按照下方所示的 GBN 发送方与接收方的状态转移图，设计 GBN 协议。

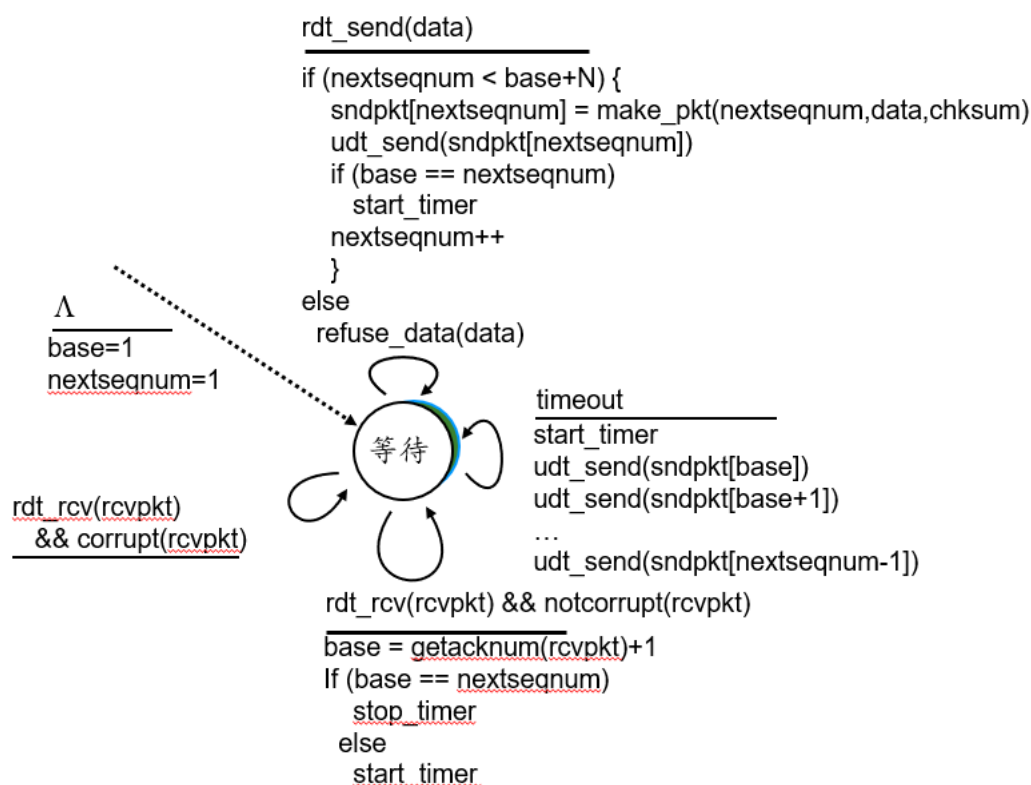


图 2-1 GBN 发送方

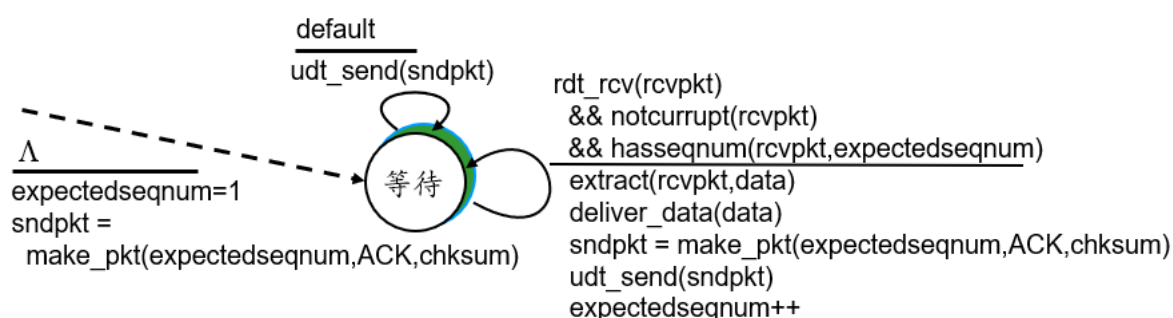


图 2-2 GBN 接受方

假设实验编码报文序号的二进制位数为 2，则发送方窗口大小为 3，接收方窗口大小为 1。

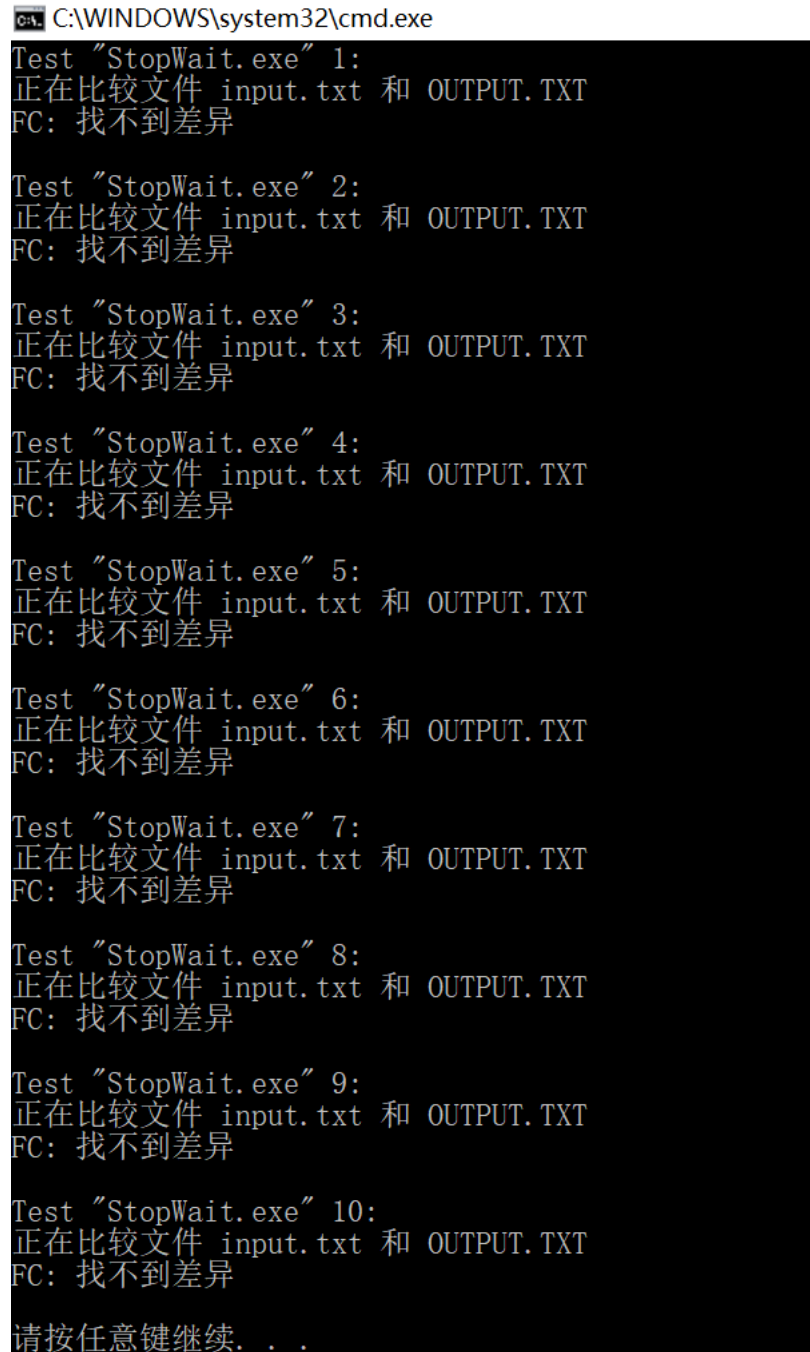
GBN 发送方缓冲区的实现方法是建立一个链表，将已发送但未确认的报文保存，等到需要重传的时候将链表中所有的报文重新传送。

当 GBN 缓冲区里报文满了的时候，进入等待状态。

值得注意的是 GBN 发送方的 receive 方法中，每次滑动窗口改变的时候，要求输出缓冲区中的报文。另外就是定时器重新开启的时候，必须先关闭后开启。

验证：

使用实验提供的脚本进行检查，如下图：



```
C:\WINDOWS\system32\cmd.exe
Test "StopWait.exe" 1:
正在比较文件 input.txt 和 OUTPUT.TXT
FC: 找不到差异

Test "StopWait.exe" 2:
正在比较文件 input.txt 和 OUTPUT.TXT
FC: 找不到差异

Test "StopWait.exe" 3:
正在比较文件 input.txt 和 OUTPUT.TXT
FC: 找不到差异

Test "StopWait.exe" 4:
正在比较文件 input.txt 和 OUTPUT.TXT
FC: 找不到差异

Test "StopWait.exe" 5:
正在比较文件 input.txt 和 OUTPUT.TXT
FC: 找不到差异

Test "StopWait.exe" 6:
正在比较文件 input.txt 和 OUTPUT.TXT
FC: 找不到差异

Test "StopWait.exe" 7:
正在比较文件 input.txt 和 OUTPUT.TXT
FC: 找不到差异

Test "StopWait.exe" 8:
正在比较文件 input.txt 和 OUTPUT.TXT
FC: 找不到差异

Test "StopWait.exe" 9:
正在比较文件 input.txt 和 OUTPUT.TXT
FC: 找不到差异

Test "StopWait.exe" 10:
正在比较文件 input.txt 和 OUTPUT.TXT
FC: 找不到差异

请按任意键继续. . .
```

图 2-3 验证 GBN 协议

```

接收方发送确认报文: seqnum = -1, acknum = 1, checksum = 12850, .....
*****模拟网络环境*****: 接收方的确认包将在97.845到达对方, 确认包为-->seqnum = -1, acknum = 1,checksum = 12850,payload = .....
此时窗口的base值为: 1
缓存区中报文: seqnum = 1, acknum = -1, checksum = 16705, FFFFFFFFFFFFFFFFFF
缓存区中报文: seqnum = 2, acknum = -1, checksum = 14134, GGGGGGGGGGGGGGGGGGGG
*****模拟网络环境*****: 关闭定时器, 当前时间 = 89.835, 定时器报文序号 = 0
*****模拟网络环境*****: 启动定时器, 当前时间 = 89.835, 定时器报文序号 = 0, 定时器Timeout时间 = 109.835
接收方正确收到发送方的报文: seqnum = 2, acknum = -1, checksum = 14134, GGGGGGGGGGGGGGGGGGGG
*****模拟网络环境*****: 向上递交给应用层数据: GGGGGGGGGGGGGGGGGGGG

接收方发送确认报文: seqnum = -1, acknum = 2, checksum = 12849, .....
*****模拟网络环境*****: 接收方的确认包将在100.575到达对方, 确认包为-->seqnum = -1, acknum = 2,checksum = 12849,payload = .....
此时窗口的base值为: 2
缓存区中报文: seqnum = 2, acknum = -1, checksum = 14134, GGGGGGGGGGGGGGGGGGGG
*****模拟网络环境*****: 关闭定时器, 当前时间 = 97.845, 定时器报文序号 = 0
*****模拟网络环境*****: 启动定时器, 当前时间 = 97.845, 定时器报文序号 = 0, 定时器Timeout时间 = 117.845
此时窗口的base值为: 3
发送方正确收到确认: seqnum = -1, acknum = 2, checksum = 12849, .....
*****模拟网络环境*****: 关闭定时器, 当前时间 = 100.575, 定时器报文序号 = 0
发送方发送报文: seqnum = 3, acknum = -1, checksum = 11563, HHHHHHHHHHHHHHHHHHHH

```

图 2-4 验证 GBN 协议

结果分析:

比较输入和输出没有差异, 可以看到窗口滑动正确, 并且缓冲区中的报文也是正确的。测试通过。

2.3.2 SR 协议的设计、验证及结果分析

设计:

下面将给出 SR 发送方 send 和 receive 方法和接收方 receive 方法的流程图。

SR 发送方超时, 先关闭计时器, 后打开计时器, 然后在缓冲区报文里找到超时报文, 进行重传就行了。

假设实验编码报文序号的二进制位数为 3, 则发送方窗口大小为 4, 接收方窗口大小为 4。

SR 发送方缓冲区的实现方法是建立一个链表, 将已发送但未确认的报文保存, 等到需要重传的时候将链表中所有的报文重新传送。接收方缓冲区的实现也是一样的。

值得注意的是 SR 发送方和接收方, 每次滑动窗口改变的时候, 要求输出缓冲区中的报文。另外就是定时器重新开启的时候, 必须先关闭后开启。

图 2-5 为发送方 send 方法流程图。

图 2-6 为发送方 receive 方法流程图。

图 2-7 为接收方 receive 方法流程图。

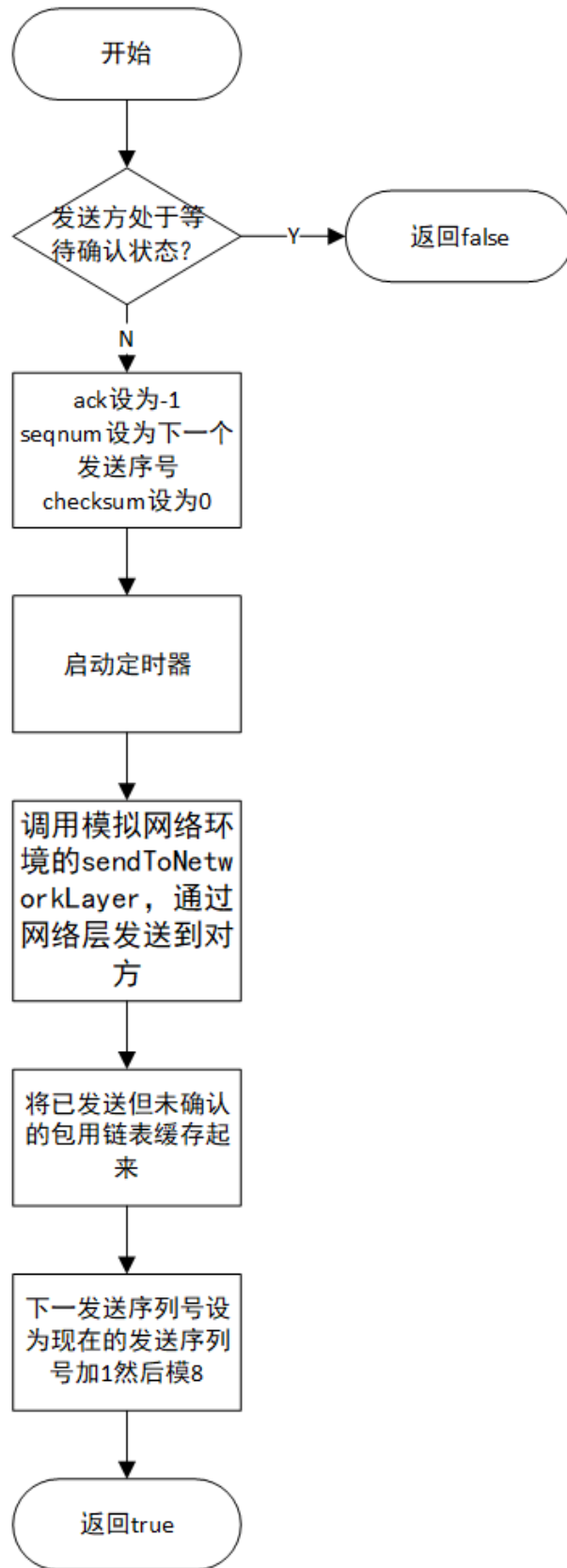


图 2-5 SR 发送方 send 方法流程图

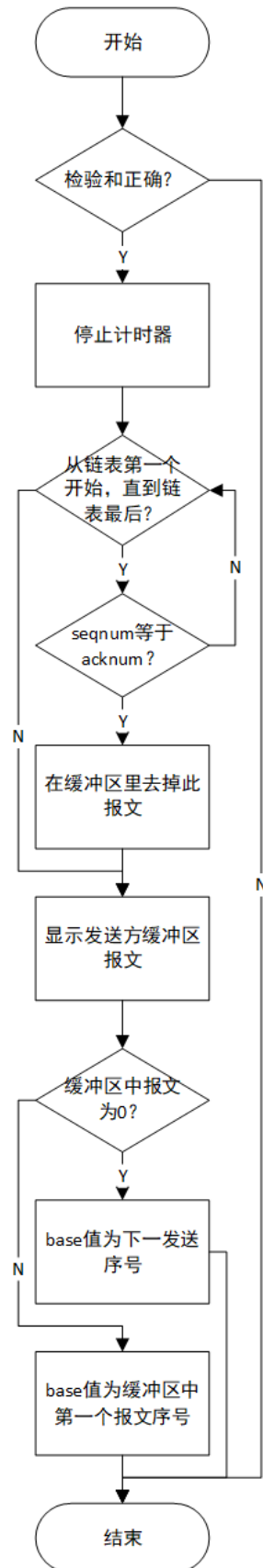


图 2-6 SR 发送方 receive 方法流程图

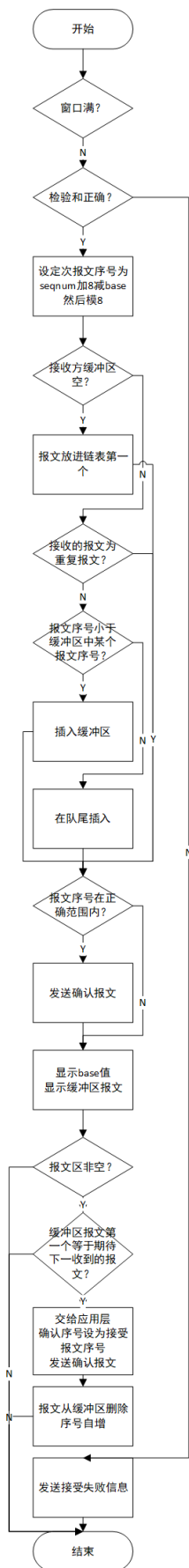
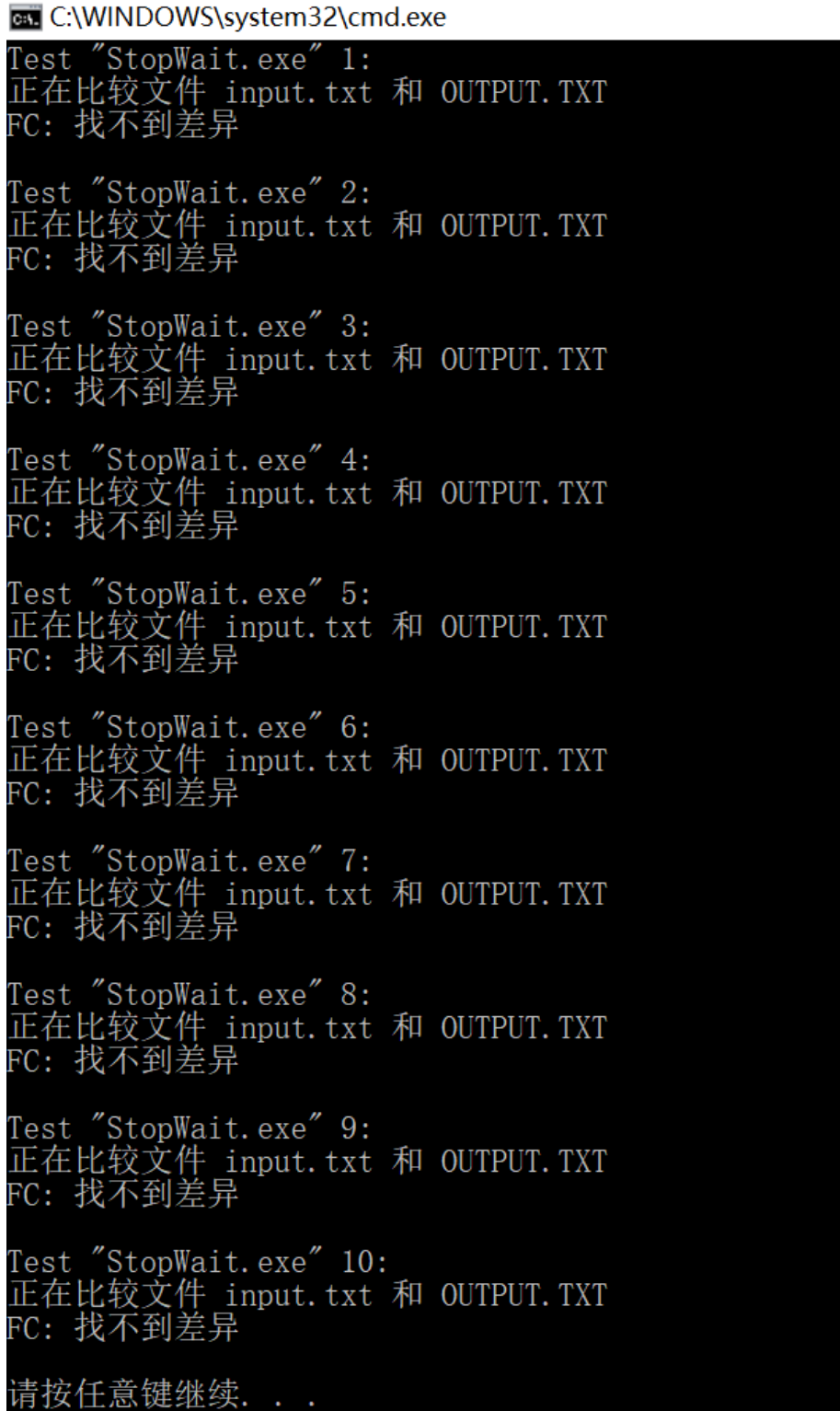


图 2-7 SR 接收方 receive 方法流程图

验证:

使用实验提供的脚本进行检查, 如下图:



```
C:\WINDOWS\system32\cmd.exe
Test "StopWait.exe" 1:
正在比较文件 input.txt 和 OUTPUT.TXT
FC: 找不到差异

Test "StopWait.exe" 2:
正在比较文件 input.txt 和 OUTPUT.TXT
FC: 找不到差异

Test "StopWait.exe" 3:
正在比较文件 input.txt 和 OUTPUT.TXT
FC: 找不到差异

Test "StopWait.exe" 4:
正在比较文件 input.txt 和 OUTPUT.TXT
FC: 找不到差异

Test "StopWait.exe" 5:
正在比较文件 input.txt 和 OUTPUT.TXT
FC: 找不到差异

Test "StopWait.exe" 6:
正在比较文件 input.txt 和 OUTPUT.TXT
FC: 找不到差异

Test "StopWait.exe" 7:
正在比较文件 input.txt 和 OUTPUT.TXT
FC: 找不到差异

Test "StopWait.exe" 8:
正在比较文件 input.txt 和 OUTPUT.TXT
FC: 找不到差异

Test "StopWait.exe" 9:
正在比较文件 input.txt 和 OUTPUT.TXT
FC: 找不到差异

Test "StopWait.exe" 10:
正在比较文件 input.txt 和 OUTPUT.TXT
FC: 找不到差异

请按任意键继续. . .
```

图 2-8 验证 SR 协议

```

接收方正确收到发送方的报文: seqnum = 1, acknum = -1, checksum = 26985, BBBBBBBBBBBBBBBBBBBB
接收方期望: 1
此时接收方窗口的base值为: 1
接收方缓存区中报文: seqnum = 1, acknum = -1, checksum = 26985, BBBBBBBBBBBBBBBBBBBB
*****模拟网络环境*****: 向上递交给应用层数据: BBBBBBBBBBBBBBBBBBBB

接收方发送确认报文: seqnum = -1, acknum = 1, checksum = 12850, .....
*****模拟网络环境*****: 接收方的确认包将在44.3到达对方, 确认包为-->seqnum = -1, acknum = 1,checksum = 12850,payload = .....
接收方正确收到发送方的报文: seqnum = 2, acknum = -1, checksum = 24414, CCCCCCCCCCCCCCCCCC
接收方期望: 2
此时接收方窗口的base值为: 2
接收方缓存区中报文: seqnum = 2, acknum = -1, checksum = 24414, CCCCCCCCCCCCCCCCCC
*****模拟网络环境*****: 向上递交给应用层数据: CCCCCCCCCCCCCCCCCC

```

图 2-9 验证 SR 协议

结果分析:

比较输入和输出没有差异, 可以看到窗口滑动正确 (上图由 1 到 2), 并且缓冲区中的报文也是正确的。测试通过。

2.3.3 简单 TCP/IP 协议的设计、验证及结果分析

设计:

TCP 发送方的 send 方法和 GBN 发送方的 send 方法实现相同, 在此不赘述。TCP 发送方的 receive 方法在 GBN 的基础上有所修改。下面给出流程图。

TCP 发送方超时处理与 GBN 相同。并且 TCP 接收方与 GBN 实现相同。在此不赘述。

假设实验编码报文序号的二进制位数为 2, 则发送方窗口大小为 3, 接收方窗口大小为 1。

值得注意的是 TCP 发送方的 receive 方法中, 每次滑动窗口改变的时候, 要求输出缓冲区中的报文。另外就是定时器重新开启的时候, 必须先关闭后开启。

图 2-10 为 TCP 发送方 receive 方法流程图。

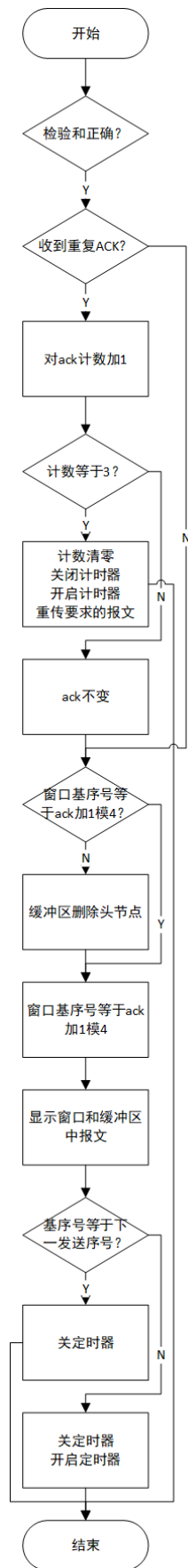
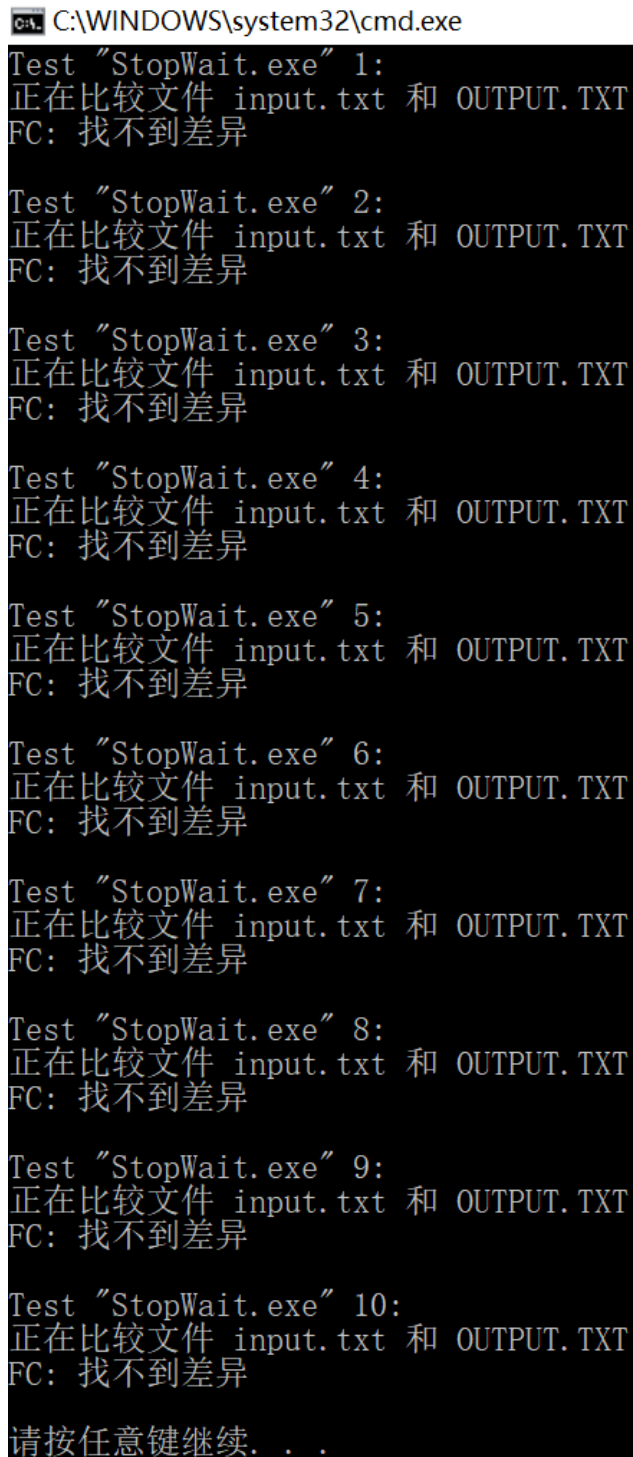


图 2-10 TCP 发送方 receive 方法流程图

验证:

使用实验提供的脚本进行检查, 如下图:



```
C:\WINDOWS\system32\cmd.exe
Test "StopWait.exe" 1:
正在比较文件 input.txt 和 OUTPUT.TXT
FC: 找不到差异

Test "StopWait.exe" 2:
正在比较文件 input.txt 和 OUTPUT.TXT
FC: 找不到差异

Test "StopWait.exe" 3:
正在比较文件 input.txt 和 OUTPUT.TXT
FC: 找不到差异

Test "StopWait.exe" 4:
正在比较文件 input.txt 和 OUTPUT.TXT
FC: 找不到差异

Test "StopWait.exe" 5:
正在比较文件 input.txt 和 OUTPUT.TXT
FC: 找不到差异

Test "StopWait.exe" 6:
正在比较文件 input.txt 和 OUTPUT.TXT
FC: 找不到差异

Test "StopWait.exe" 7:
正在比较文件 input.txt 和 OUTPUT.TXT
FC: 找不到差异

Test "StopWait.exe" 8:
正在比较文件 input.txt 和 OUTPUT.TXT
FC: 找不到差异

Test "StopWait.exe" 9:
正在比较文件 input.txt 和 OUTPUT.TXT
FC: 找不到差异

Test "StopWait.exe" 10:
正在比较文件 input.txt 和 OUTPUT.TXT
FC: 找不到差异

请按任意键继续. . .
```

图 2-11 验证 TCP 协议

```

接收方发送确认报文: seqnum = -1, acknum = 1, checksum = 12850, .....
*****模拟网络环境*****: 接收方的确认包将在40.075到达对方, 确认包为-->seqnum = -1, acknum = 1,checksum = 12850,payload = .....
此时窗口的base值为: 2
缓存区中报文: seqnum = 2, acknum = -1, checksum = 24414, CCCCCCCCCCCCCCCCCC
缓存区中报文: seqnum = 3, acknum = -1, checksum = 21843, DDDDDDDDDDDDDDDDDDDDD
*****模拟网络环境*****: 关闭定时器, 当前时间 = 40.075, 定时器报文序号 = 0
*****模拟网络环境*****: 启动定时器, 当前时间 = 40.075, 定时器报文序号 = 0, 定时器Timeout时间 = 60.075
接收方正正确收到发送方的报文: seqnum = 2, acknum = -1, checksum = 24414, CCCCCCCCCCCCCCCCCC
*****模拟网络环境*****: 向上递交给应用层数据: CCCCCCCCCCCCCCCCCC

接收方发送确认报文: seqnum = -1, acknum = 2, checksum = 12849, .....
*****模拟网络环境*****: 接收方的确认包将在43.815到达对方, 确认包为-->seqnum = -1, acknum = 2,checksum = 12849,payload = .....
此时窗口的base值为: 3
缓存区中报文: seqnum = 3, acknum = -1, checksum = 21843, DDDDDDDDDDDDDDDDDDDDD
*****模拟网络环境*****: 关闭定时器, 当前时间 = 43.815, 定时器报文序号 = 0
*****模拟网络环境*****: 启动定时器, 当前时间 = 43.815, 定时器报文序号 = 0, 定时器Timeout时间 = 63.815
接收方没有正确收到发送方的报文,数据校验错误: seqnum = 3, acknum = -1, checksum = 21843, EDDDDDDDDDDDDDDDDDDDD
接收方重新发送上次的确认报文: seqnum = -1, acknum = 2, checksum = 12849, .....

```

图 2-12 验证 TCP 协议

```

*****模拟网络环境*****: 接收方的确认包将在260.035到达对方, 确认包为-->seqnum = -1, acknum = 0,checksum = 12851,payload = .....
对于该报文进行快速重传: seqnum = 0, acknum = -1, checksum = 64251, MMMMMMMMMMMMMMMMMMMMMM
*****模拟网络环境*****: 关闭定时器, 当前时间 = 260.035, 定时器报文序号 = 0
*****模拟网络环境*****: 启动定时器, 当前时间 = 260.035, 定时器报文序号 = 0, 定时器Timeout时间 = 280.035
*****模拟网络环境*****: 发送方的数据包将在267.535到达对方, 数据包为-->seqnum = 0, acknum = -1,checksum = 64251,payload = MMMMMMMMMMMMMMMMMMM

```

图 2-13 验证 TCP 协议

结果分析:

比较输入和输出没有差异, 可以看到窗口滑动正确, 并且缓冲区中的报文也是正确的, 并且能够观察到快速重传的例子。测试通过。

2.4 其它需要说明的问题

无

实验三 基于 CPT 的组网实验

3.1 环境

硬件配置：

处理器： Intel(R)Core(TM)i5-7200U CPU @ 2.50GHz(4 CPUs), ~2.7GHz
内存： 8192MB RAM
显卡： Intel(R) HD Graphics 620

系统软件组件：

Windows 10 64 位操作系统

第三方软件：

Cisco Packet Tracer 6.0

3.2 实验要求

- 1.熟悉 Cisco Packet Tracer 仿真软件。
- 2.利用 Cisco Packet Tracer 仿真软件完成实验内容。
- 3.提交实验设计报告纸质档和电子档。
- 4.基于自己的实验设计报告，通过实验课的上机实验，演示给实验指导教师检查。

3.3 基本部分实验步骤说明及结果分析

3.3.1 IP 地址规划与 Vlan 分配实验的步骤及结果分析

基本内容 1

实验步骤：

- 1.根据实验图画出 CPT 拓扑图，如下：

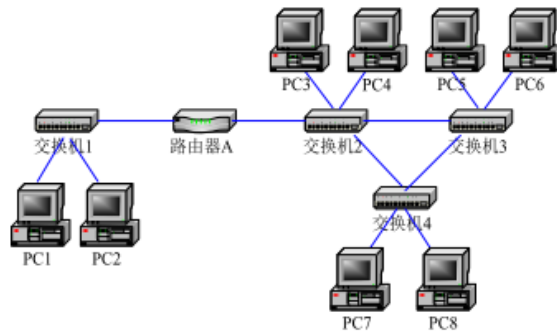


图 3.1 实验 1 拓扑图

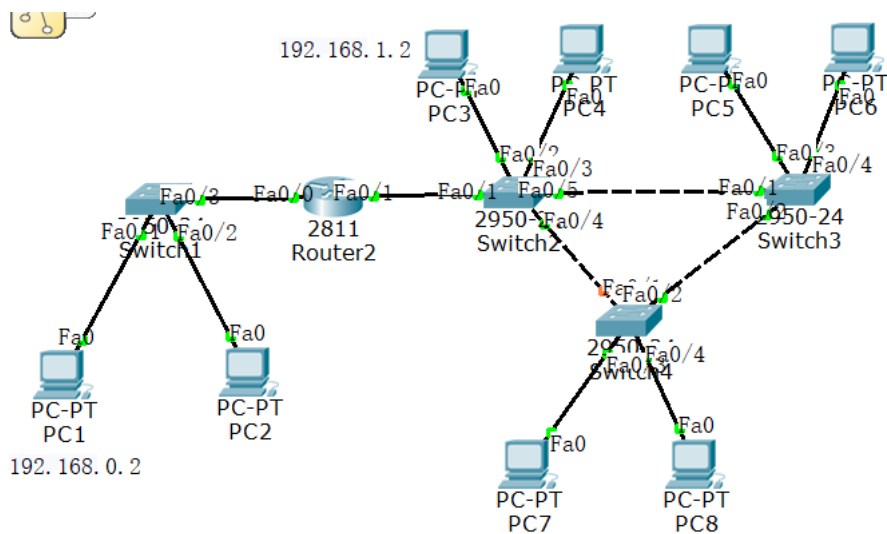


图 3.2 CPT 拓扑图

2.将 PC1、PC2 设置在同一个网段，子网地址是：192.168.0.0/24;

下图显示 PC1 的配置，PC2 配置类似。只不过将 PC2 接口的 IP 地址设为 192.168.0.3，网关相同。

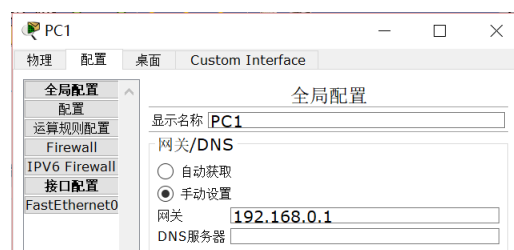


图 3.3 PC1 网关

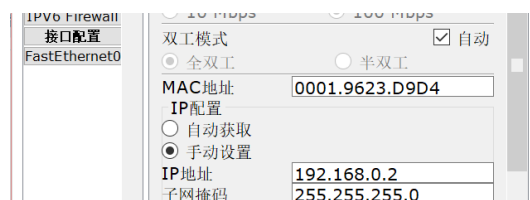


图 3.4 PC1 IP 设置

3.将 PC3~PC8 设置在同一个网段，子网地址是：192.168.1.0/24;

下图显示 PC3 的配置，PC4~PC8 配置类似。只不过将各 PC 接口的 IP 地址不同，网关相同。

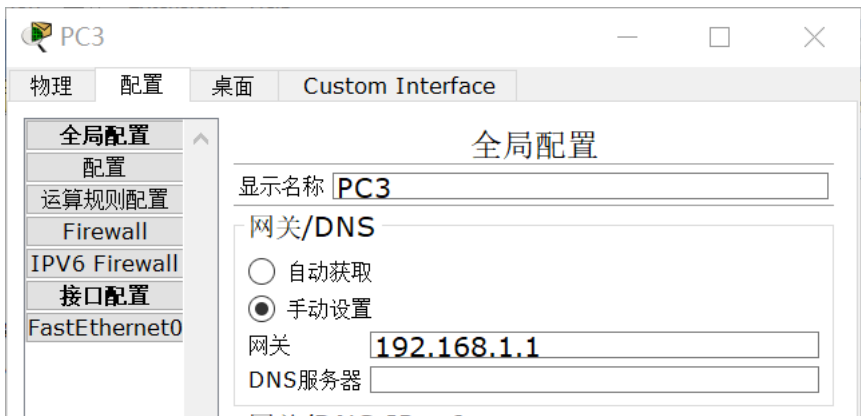


图 3.5 PC3 网关

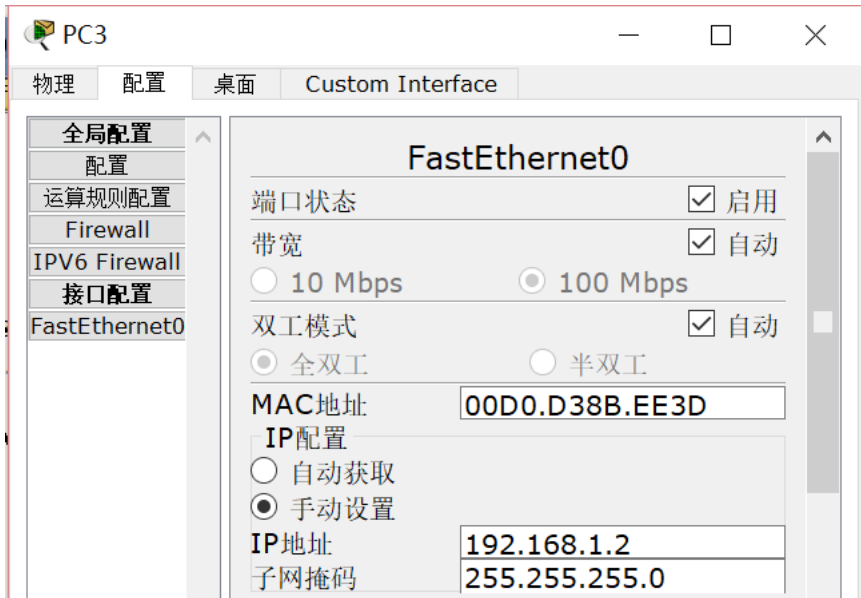


图 3.6 PC3 IP 设置

4.配置路由器，使得两个子网的各 PC 机之间可以自由通信。

只需要配置路由器 Router2 的两个接口的 IP 地址及子网掩码即可，分别作为两个子网的网关。

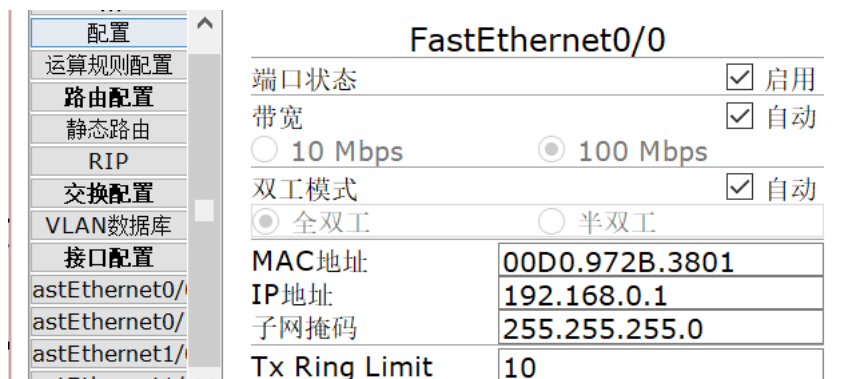


图 3.7 Router2 IP 设置

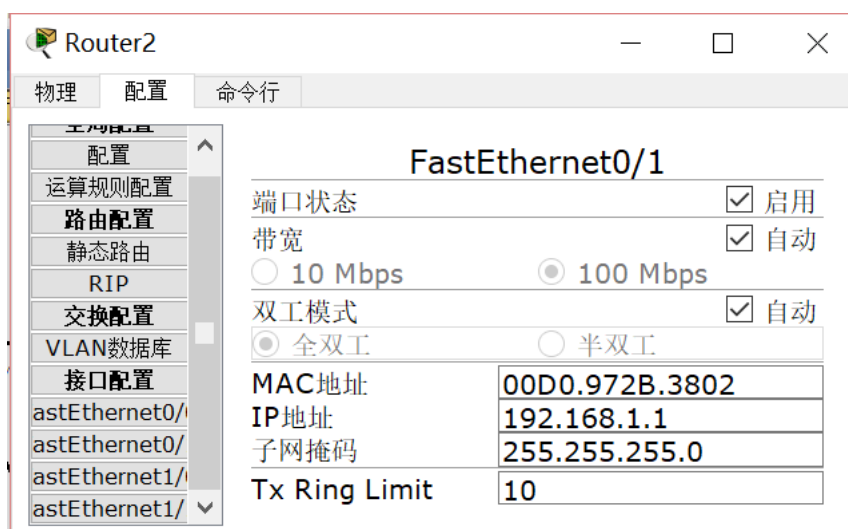


图 3.8 Router2 IP 设置

结果分析:

实验要求两个子网的各 PC 机之间可以自由通信。在此为了简洁说明，给出代表性的测试图，下图为 PC1 ping PC3，表示两个子网之间的通信是正常的。

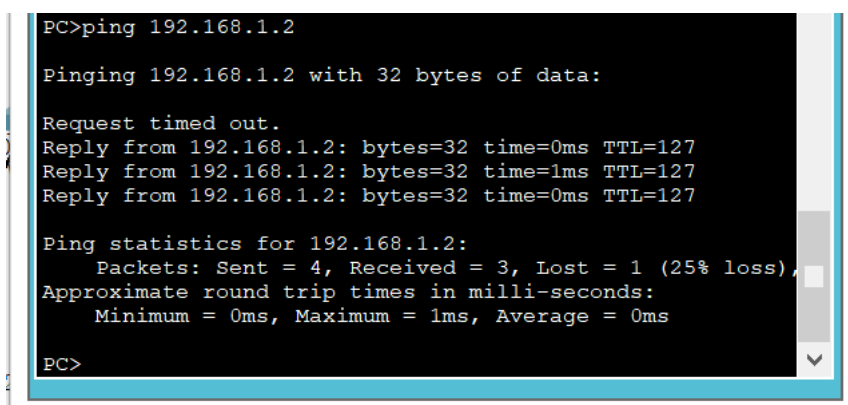


图 3.9 PC1 ping PC3

图中 192.168.1.2 即为 PC3 的 IP 地址，由上图可以看出，PC1 和 PC3 之间能够自由通信，测试通过。

基本内容 2

实验步骤:

1.根据实验图画出 CPT 拓扑图，如下：

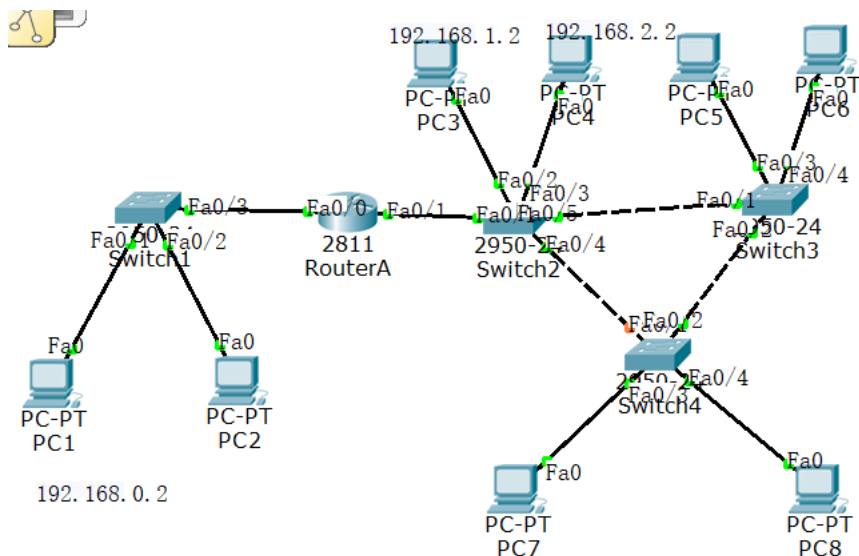


图 3.10 CPT 拓扑图

2.将 PC1、PC2 设置在同一个网段，子网地址是：192.168.0.0/24；

下图显示 PC1 的配置，PC2 配置类似。只不过将 PC2 接口的 IP 地址设为 192.168.0.3，网关相同。

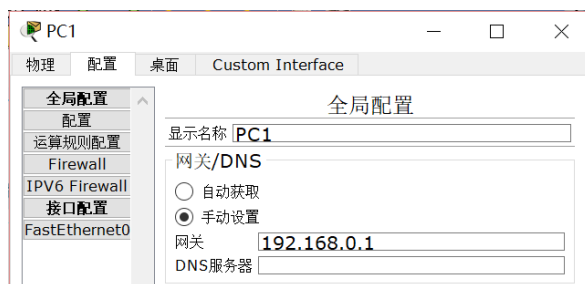


图 3.11 PC1 网关

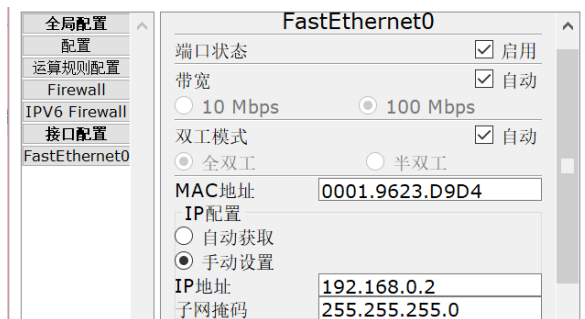


图 3.12 PC1 IP 设置

3.将 PC3、PC5、PC7 设置在同一个网段，子网地址是：192.168.1.0/24；

下图显示 PC3 的配置，PC5、PC7 配置类似。只不过将各 PC 接口的 IP 地址不同，网关相同。

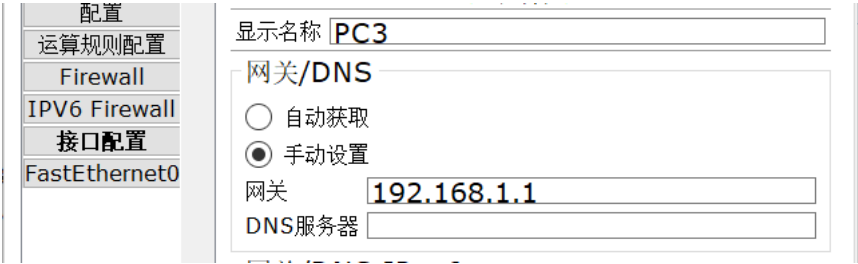


图 3.13 PC3 网关

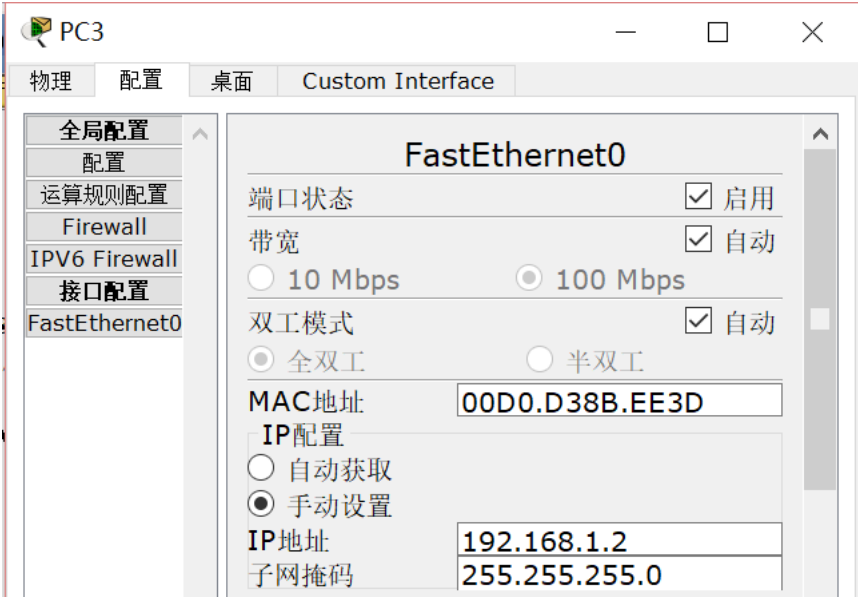


图 3.14 PC3 IP 设置

4.将 PC4、PC6、PC8 设置在同一个网段，子网地址是：192.168.2.0/24;

下图显示 PC4 的配置，PC6、PC8 配置类似。只不过将各 PC 接口的 IP 地址不同，网关相同。

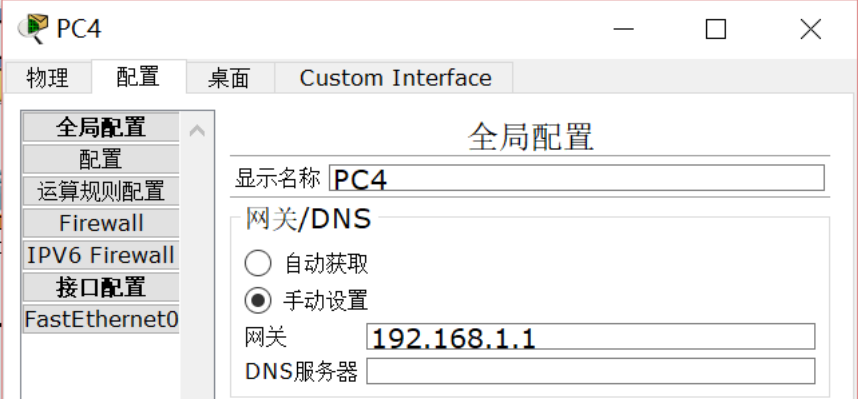


图 3.15 PC4 网关

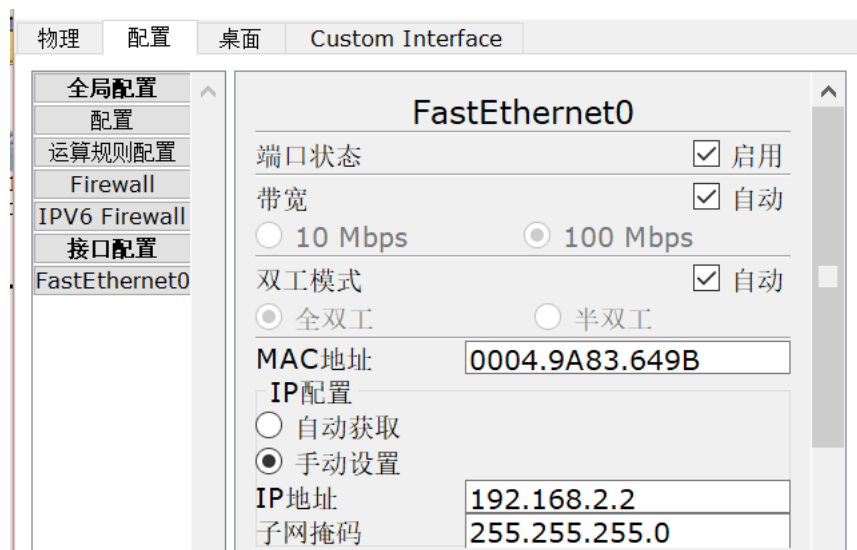


图 3.16 PC4 IP 设置

5.配置交换机 1、2、3、4，使得 PC1、PC2 属于 Vlan2，PC3、PC5、PC7 属于 Vlan3，PC4、PC6、PC8 属于 Vlan4；

下面以配置交换机 1 为例子，说明 VLAN 划分方法。

1) 在交换机上创建 VLAN2

```
Switch# config terminal //进入配置模式
Switch(config)# vlan 2 //在交换机上添加 vlan2
Switch(config-vlan)# exit //退出 vlan2，回到配置模式
```

图 3.17 创建 VLAN

2) 将 1 号端口和 2 号端口分配到 VLAN2 中

```
Switch(config)#int fa0/1
Switch(config-if)#switchport mode access
Switch(config-if)#switchport access vlan 2
Switch(config-if)#exit
```

图 3.18 给端口分配 VLAN

3) 配置两个交换机之间的 trunk 链路

```
Switch(config)# interface fa0/2
Switch(config-if)# switchport mode trunk//配置为干道模式
```

图 3.19 配置两个交换机之间的 trunk 链路

其他交换机也是进行类似配置即可。

6.测试各 PC 之间的连通性，并结合所学理论知识进行分析；

这部分在结果分析中给出。

7.配置路由器，使得拓扑图上的各 PC 机之间可以自由通信，结合所学理论对你的路由器配置过程进行详细说明。

将 f0/0 子端口 1 与 VLAN2 进行封装，并且设置 ip 192.168.0.1/24。将 f0/1 子端口 1 与 VLAN3 进行封装，并设置 ip 192.168.1.1/24。将 f0/1 子端口 2 与 VLAN4 进行封装，并设置 ip 192.168.2.1/24。

因为路由器仅通过接口 f0/1 与交换机进行连接，但是该接口实际上连接了两个 VLAN，因此为了联通两个 VLAN，需要将一个接口分为两个子接口，两个子接口对应两个 VLAN。这样不同的 VLAN 之间就可以进行相互通信了。

```
Router>en
Router#conf t
Enter configuration commands, one per line. End with CNTL
Router(config)#int f0/0.1
Router(config-subif)#encapsulation dot1q 2
Router(config-subif)#ip address 192.168.0.1 255.255.255.0
Router(config-subif)#exit
Router(config)#int f0/0
Router(config-if)#no shut
Router(config-if)#exit
Router(config)#end
Router#
%SYS-5-CONFIG_I: Configured from console by console
Router#
```

图 3.20 配置路由器

其余的接口配置也和上面类似。

结果分析：

1.在划分 VLAN 之后，测试各 PC 间的连通性。

在划分完 VLAN 后，PC3 能 ping 通 PC5、PC7，不能 ping 通其他的 PC.，结果如下图所示：

原因分析：PC3、PC5、PC7 之间能够互相通信是因为它们属于同一各 VLAN，而它们和其他的 PC 不属于同一 VLAN，因此它们不能通信。根本原因是因为同一 VLAN 中各 PC 能够互相通信，而不同 VLAN 之间不能互相通信。

```
PC>ping 192.168.1.3

Pinging 192.168.1.3 with 32 bytes of data:

Reply from 192.168.1.3: bytes=32 time=1ms TTL=128
Reply from 192.168.1.3: bytes=32 time=0ms TTL=128
Reply from 192.168.1.3: bytes=32 time=1ms TTL=128
Reply from 192.168.1.3: bytes=32 time=0ms TTL=128

Ping statistics for 192.168.1.3:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 0ms, Maximum = 1ms, Average = 0ms
```

图 3.21 测试 VLAN 连通性

```
PC>ping 192.168.2.2

Pinging 192.168.2.2 with 32 bytes of data:

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

Ping statistics for 192.168.2.2:
    Packets: Sent = 4, Received = 0, Lost = 4 (100% loss),
```

图 3.22 测试 VLAN 连通性

```
PC>ping 192.168.0.2

Pinging 192.168.0.2 with 32 bytes of data:

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

Ping statistics for 192.168.0.2:
    Packets: Sent = 4, Received = 0, Lost = 4 (100% loss),
```

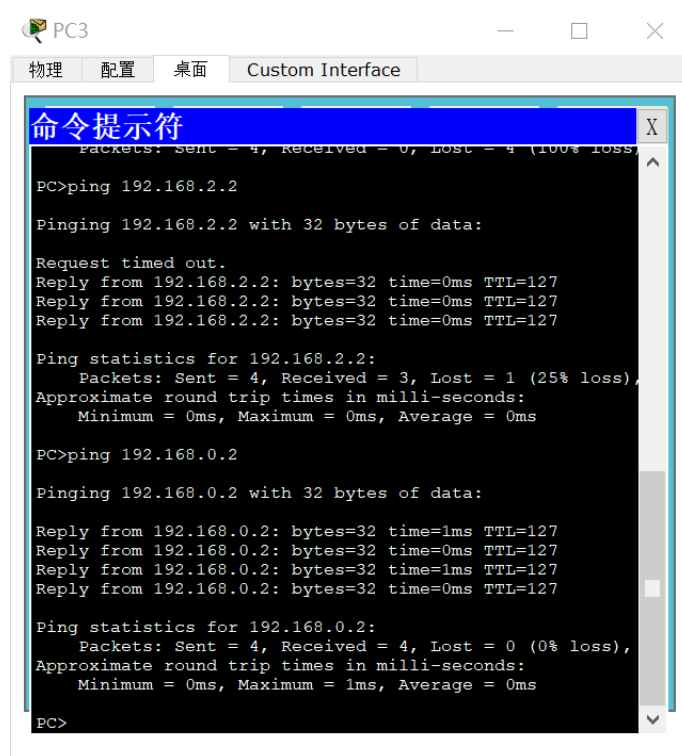
图 3.23 测试 VLAN 连通性

2.配置完路由器后，测试各 PC 机的通信情况。

测试时用 PC3 ping PC4 和 PC1，观察通信情况。

观察到 PC3 ping PC4 和 PC1 都能 ping 通，测试通过。

原因分析：路由器的接口连到了 3 个 VLAN，使得不同 VLAN 间可以通信。



```
PC3
物理 配置 桌面 Custom Interface

命令提示符
Packets: Sent = 4, Received = 0, Lost = 4 (100% loss)

PC>ping 192.168.2.2

Pinging 192.168.2.2 with 32 bytes of data:

Reply from 192.168.2.2: bytes=32 time=0ms TTL=127
Reply from 192.168.2.2: bytes=32 time=0ms TTL=127
Reply from 192.168.2.2: bytes=32 time=0ms TTL=127

Ping statistics for 192.168.2.2:
    Packets: Sent = 4, Received = 3, Lost = 1 (25% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 0ms, Maximum = 0ms, Average = 0ms

PC>ping 192.168.0.2

Pinging 192.168.0.2 with 32 bytes of data:

Reply from 192.168.0.2: bytes=32 time=1ms TTL=127
Reply from 192.168.0.2: bytes=32 time=0ms TTL=127
Reply from 192.168.0.2: bytes=32 time=1ms TTL=127
Reply from 192.168.0.2: bytes=32 time=0ms TTL=127

Ping statistics for 192.168.0.2:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 0ms, Maximum = 1ms, Average = 0ms

PC>
```

图 3.24 测试 VLAN 连通性

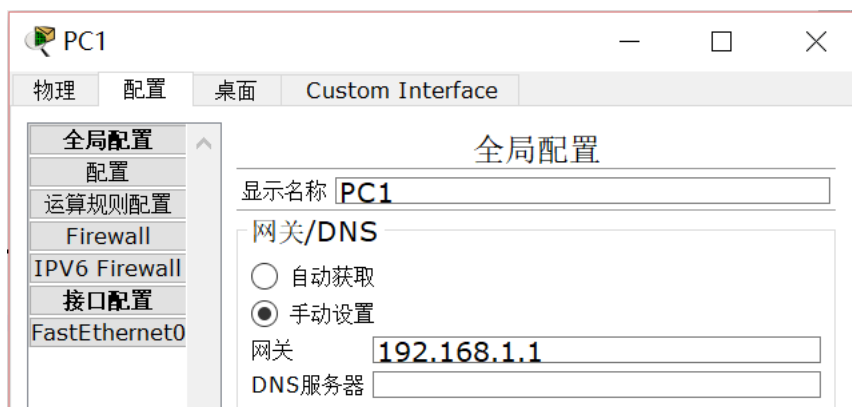


图 3.27 PC1 网关

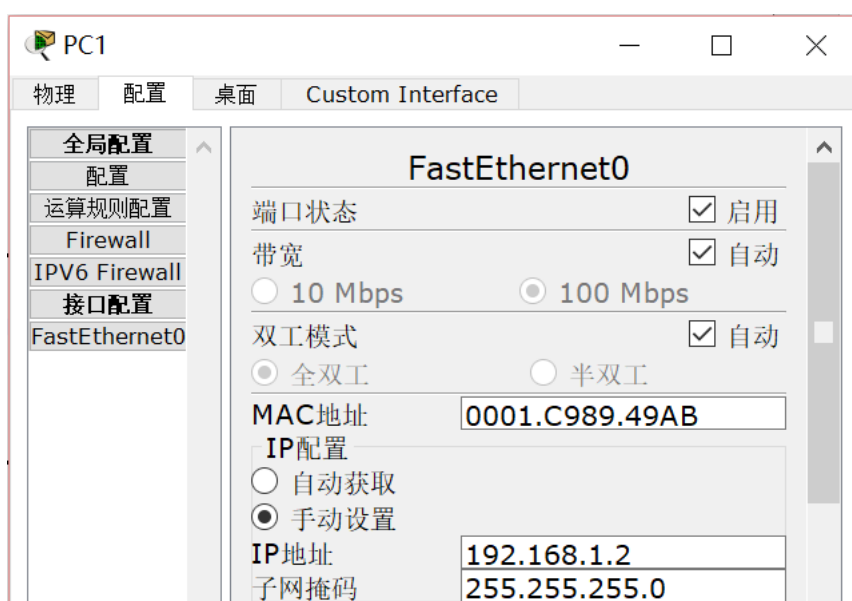


图 3.28 PC1 IP 设置

3. 设置路由器端口的 IP 地址

下图显示 RouterA 的配置。

将路由器与网段 192.168.1.0/24 连接的接口 ip 设置为 192.168.1.1/24 作为网关。将路由器 A 与其他路由器相连接的串行接口分别设为 192.168.5.1/24 和 192.168.8.1/24 用于路由器间路由。

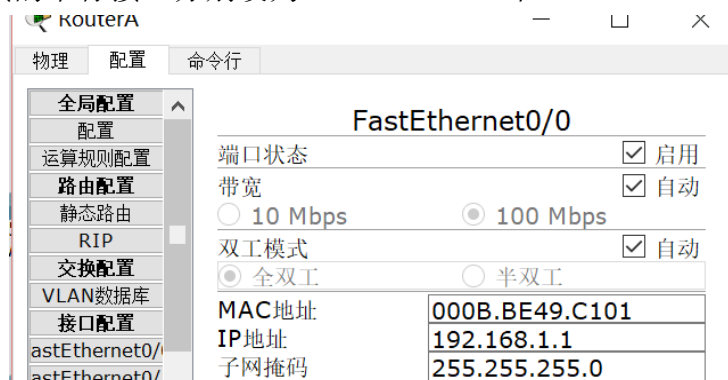


图 3.29 RouterA IP 设置

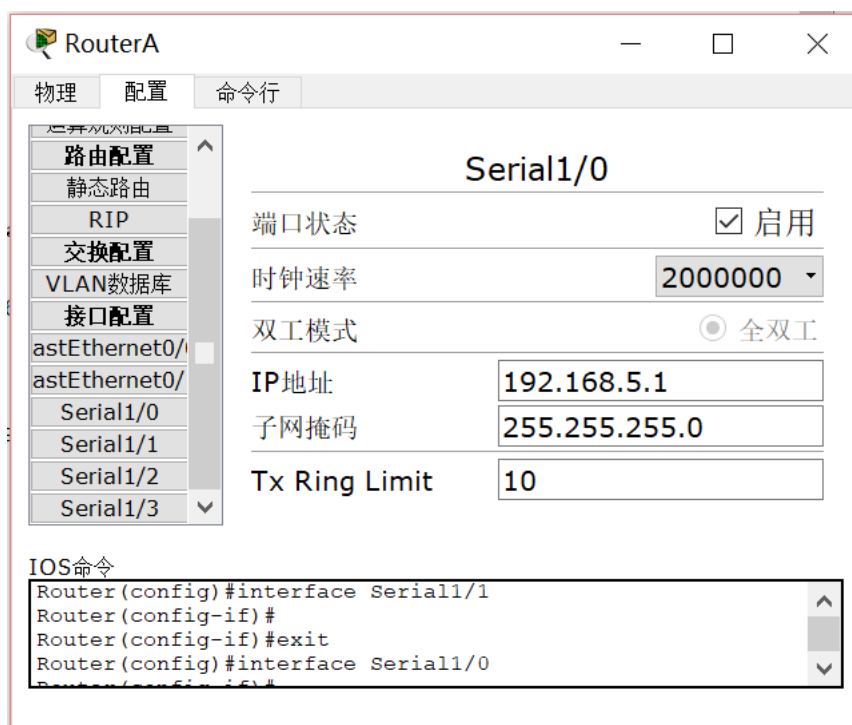


图 3.30 RouterA IP 设置

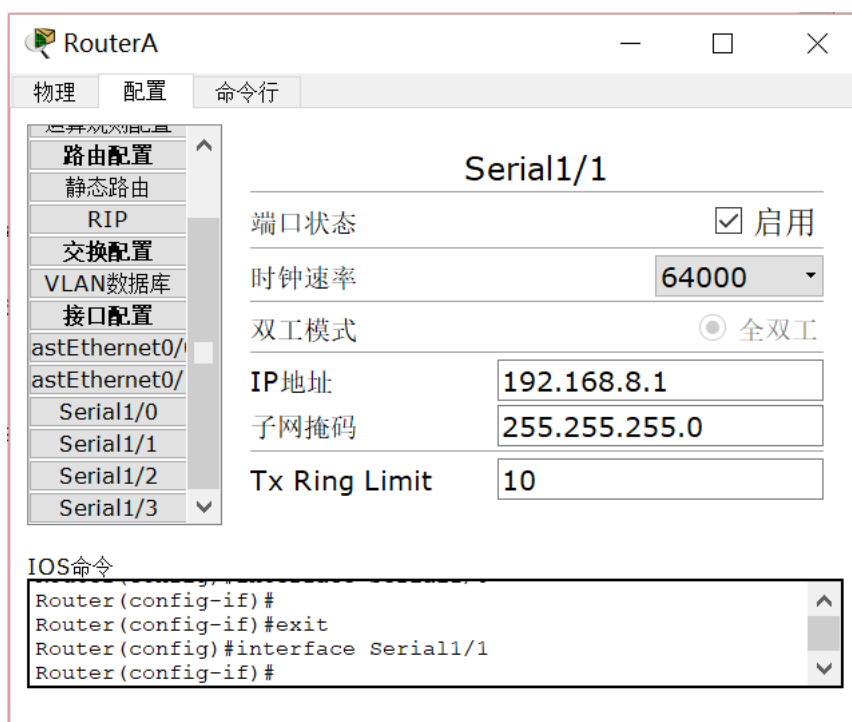


图 3.31 RouterA IP 设置

其他路由器配置与 A 类似，在此不再过多说明。

4.在路由器上配置 RIP 协议，使各 PC 机能互相访问。

只需要配置与路由器直接相连的子网即可，如下图为 RouterA 的 RIP 协议：

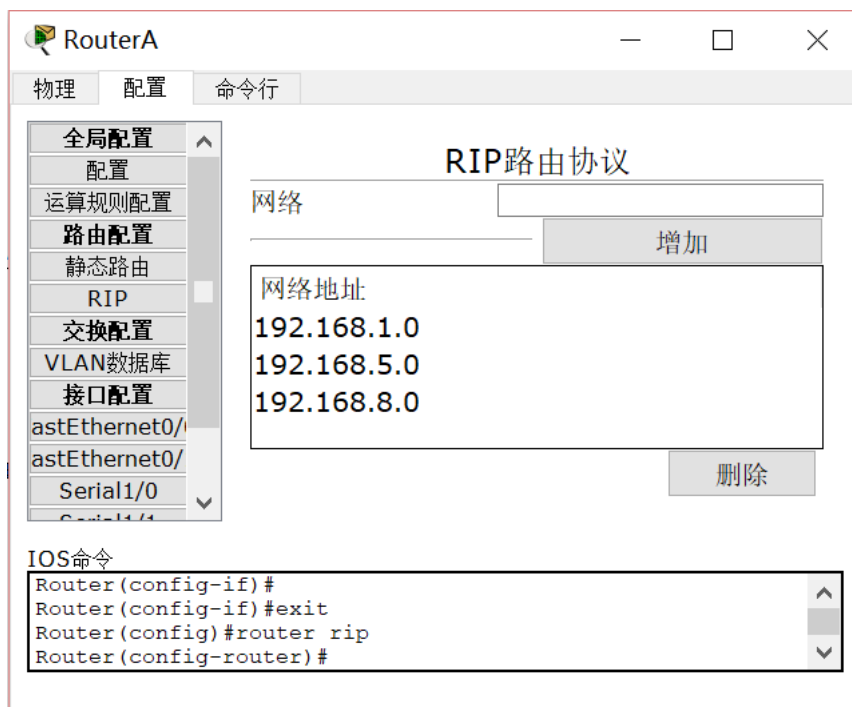


图 3.32 RouterA RIP 协议

其他路由器配置与 A 类似，在此不再过多说明。

结果分析：

实验要求各PC机间能够互相通信。在此为了表现充分性，给出三张测试图，分别为PC1 ping PC2、PC1 ping PC3、PC1 ping PC4。

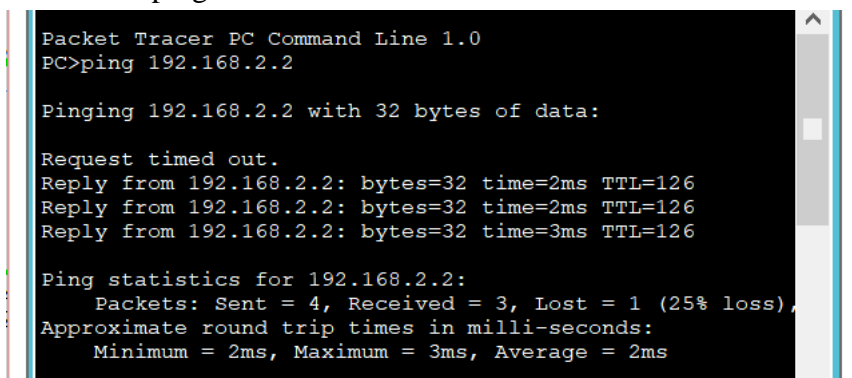


图 3.33 测试 RIP 协议

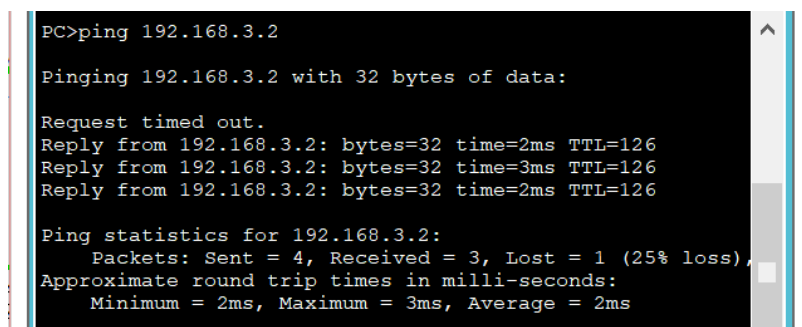


图 3.34 测试 RIP 协议

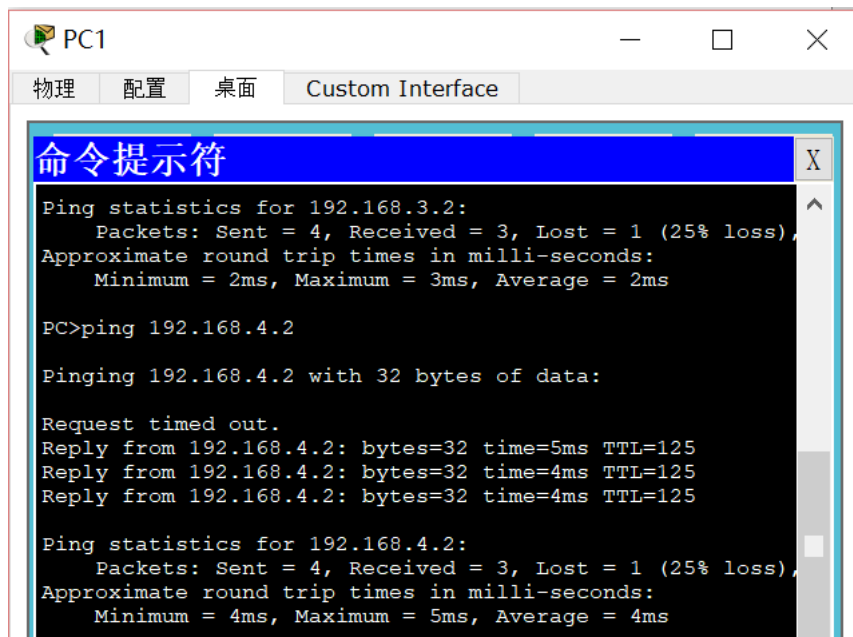


图 3.35 测试 RIP 协议

由上面三幅图可以看出，各 PC 机间能够互相通信，测试通过。

基本内容 2

实验步骤：

1.根据实验图画出 CPT 拓扑图，如下：

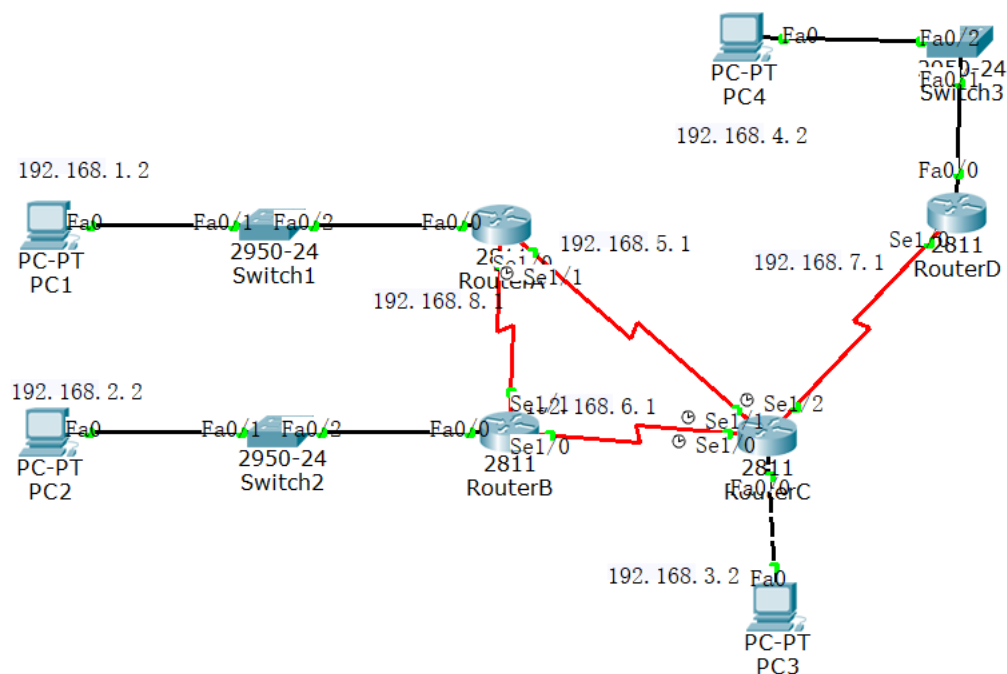


图 3.36 CPT 拓扑图

2.将 PC1 设置在 192.168.1.0/24 网段；将 PC2 设置在 192.168.2.0/24 网段；将 PC3 设置

在 192.168.3.0/24 网段；将 PC4 设置在 192.168.4.0/24 网段

下图显示 PC1 的配置。PC2、PC3、PC4 的配置类似，网关与 ip 地址不同，在此不展示。

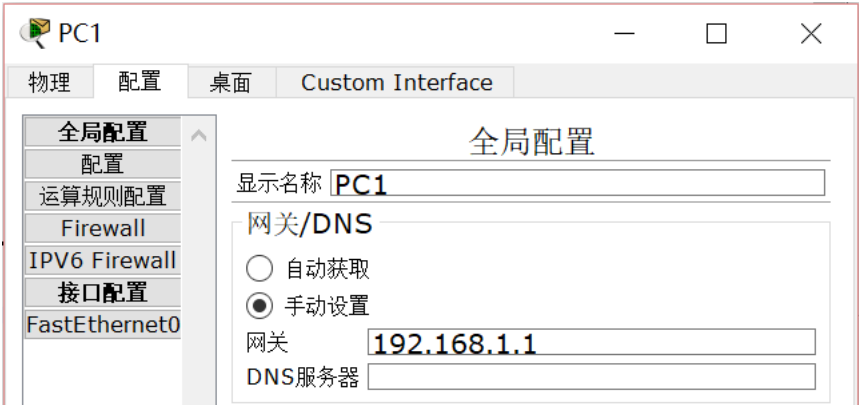


图 3.37 PC1 网关

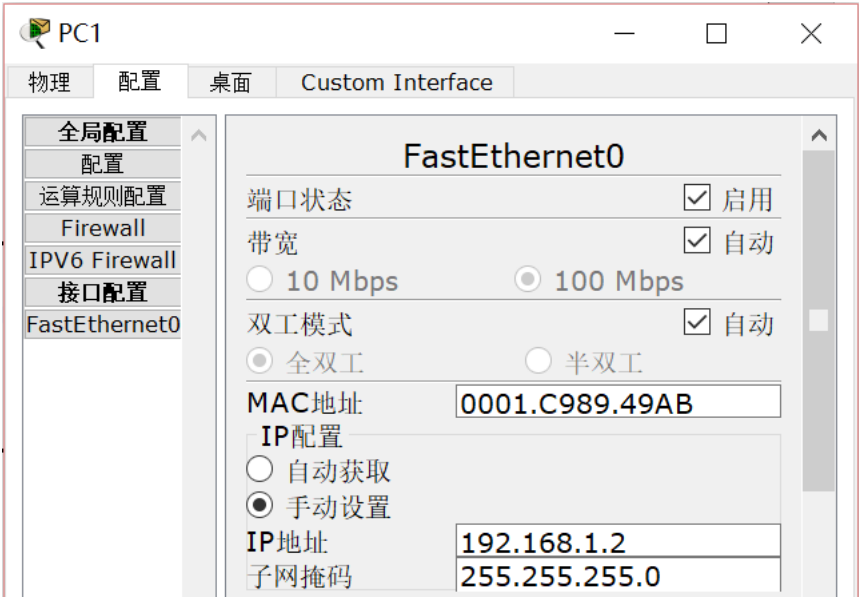


图 3.38 PC1 IP 设置

3.设置路由器端口的 IP 地址

下图显示 RouterA 的配置。

将路由器与网段 192.168.1.0/24 连接的接口 ip 设置为 192.168.1.1/24 作为网关。将路由器 A 与其他路由器相连接的串行接口分别设为 192.168.5.1/24 和 192.168.8.1/24 用于路由器间路由。

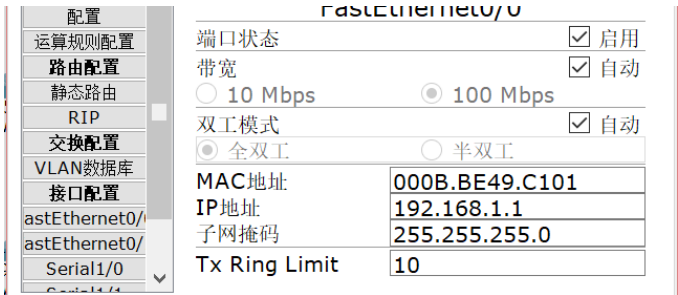


图 3.39 RouterA IP 设置

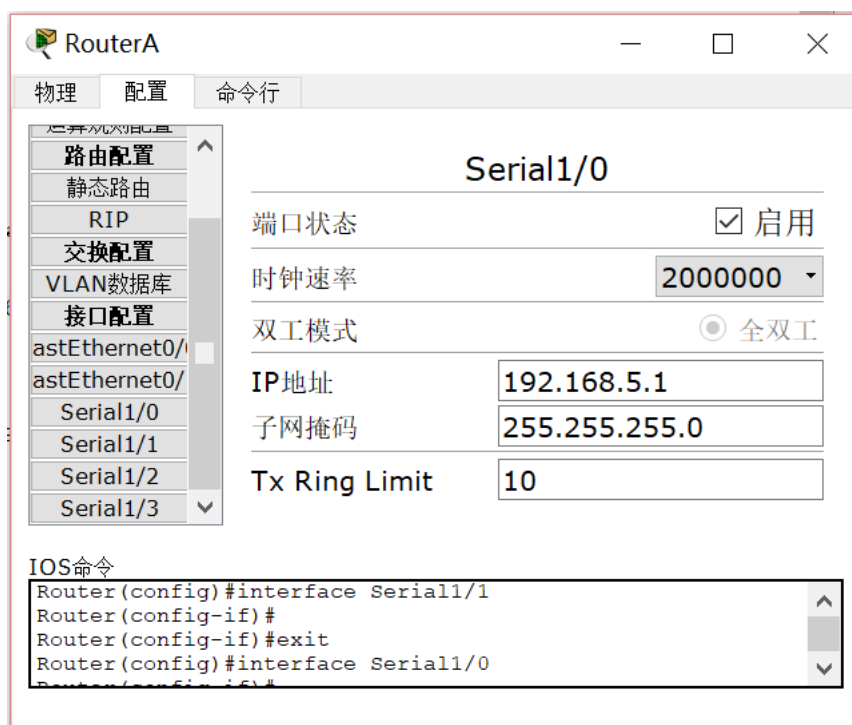


图 3.40 RouterA IP 设置

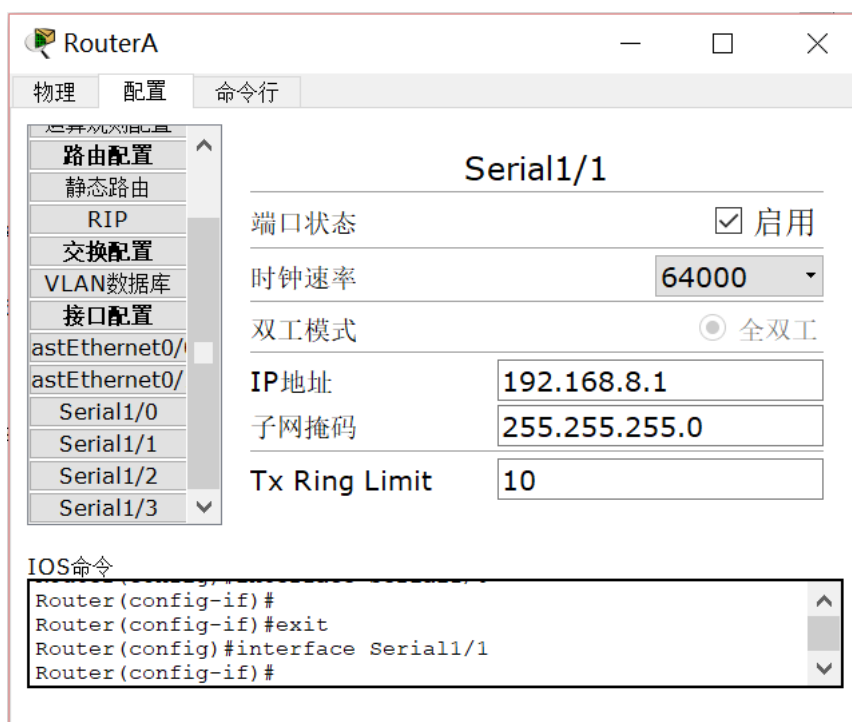


图 3.41 RouterA IP 设置

其他路由器配置与 A 类似，在此不再过多说明。

4.在路由器上配置 OSPF 协议，使各 PC 机能互相访问。

只需要配置与路由器直接相连的子网即可，配置 OSPF 协议的方法如下图所示：

```

Router#conf t           //进入全局配置模式
Router(config)#router ospf 1   //选择 ospf 协议
Router(config-router)#network 192.168.4.0 0.0.0.255 area 0
//0.0.0.255 是翻转掩码, 0 和 1 设置刚好和子网掩码相反
Router(config-router)#network 192.168.1.0 0.0.0.255 area 0
Router(config-router)#end
Router#copy run startup

```

图 3.42 OSPF 协议

如下图为 RouterA 的跳转表，图中的 O 就表示通过 OSPF 协议。其中 RouterA 自己需要配置 192.168.1.0/24、192.168.5.0/24、192.168.8.0/24。



图 3.43 RouterA OSPF 协议

其他路由器配置与 A 类似，在此不再过多说明。

结果分析：

实验要求各 PC 机间能够互相通信。在此为了表现充分性，给出三张测试图，分别为 PC1 ping PC2、PC1 ping PC3、PC1 ping PC4。

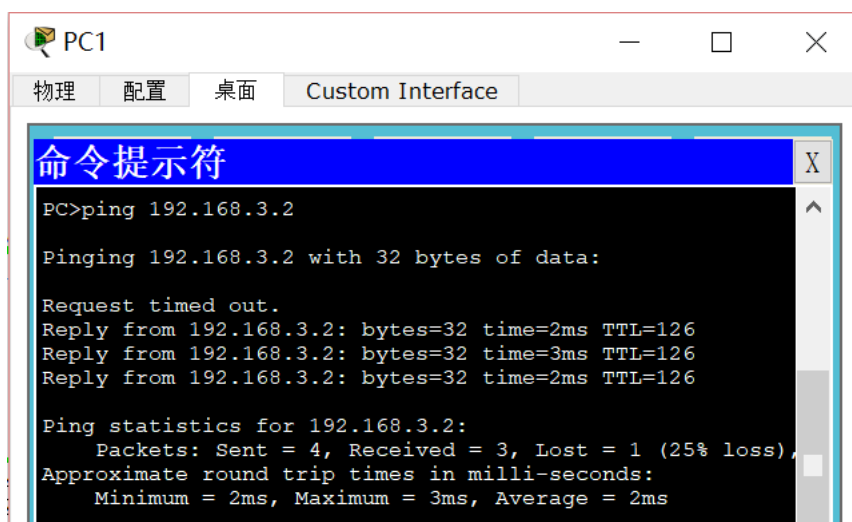
```
PC>ping 192.168.2.2

Pinging 192.168.2.2 with 32 bytes of data:

Request timed out.
Reply from 192.168.2.2: bytes=32 time=2ms TTL=126
Reply from 192.168.2.2: bytes=32 time=2ms TTL=126
Reply from 192.168.2.2: bytes=32 time=3ms TTL=126

Ping statistics for 192.168.2.2:
    Packets: Sent = 4, Received = 3, Lost = 1 (25% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 2ms, Maximum = 3ms, Average = 2ms
```

图 3.44 测试 OSPF 协议



The screenshot shows a window titled 'PC1' with tabs for '物理' (Physical), '配置' (Configuration), '桌面' (Desktop), and 'Custom Interface'. The 'Custom Interface' tab is active, displaying a command prompt window titled '命令提示符'. The command prompt shows the execution of 'PC>ping 192.168.3.2'. The output indicates a 25% loss (1 packet lost) with round trip times ranging from 2ms to 3ms.

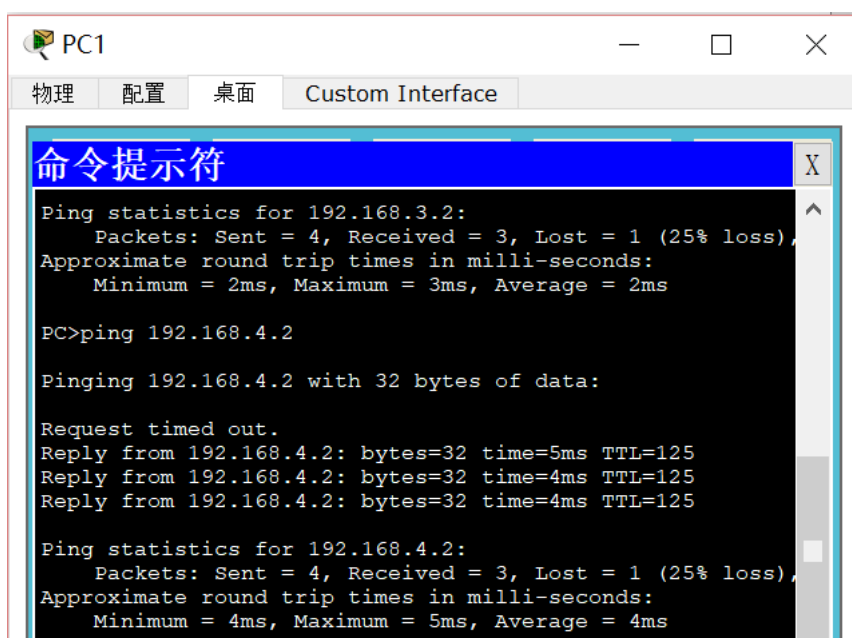
```
PC1
物理 配置 桌面 Custom Interface
命令提示符
PC>ping 192.168.3.2

Pinging 192.168.3.2 with 32 bytes of data:

Request timed out.
Reply from 192.168.3.2: bytes=32 time=2ms TTL=126
Reply from 192.168.3.2: bytes=32 time=3ms TTL=126
Reply from 192.168.3.2: bytes=32 time=2ms TTL=126

Ping statistics for 192.168.3.2:
    Packets: Sent = 4, Received = 3, Lost = 1 (25% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 2ms, Maximum = 3ms, Average = 2ms
```

图 3.45 测试 OSPF 协议



This screenshot is similar to the previous one, showing the 'PC1' window with the 'Custom Interface' tab. The command prompt now shows the execution of 'PC>ping 192.168.4.2'. The output shows a 25% loss (1 packet lost) with round trip times ranging from 4ms to 5ms.

```
PC1
物理 配置 桌面 Custom Interface
命令提示符
Ping statistics for 192.168.3.2:
    Packets: Sent = 4, Received = 3, Lost = 1 (25% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 2ms, Maximum = 3ms, Average = 2ms

PC>ping 192.168.4.2

Pinging 192.168.4.2 with 32 bytes of data:

Request timed out.
Reply from 192.168.4.2: bytes=32 time=5ms TTL=125
Reply from 192.168.4.2: bytes=32 time=4ms TTL=125
Reply from 192.168.4.2: bytes=32 time=4ms TTL=125

Ping statistics for 192.168.4.2:
    Packets: Sent = 4, Received = 3, Lost = 1 (25% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 4ms, Maximum = 5ms, Average = 4ms
```

图 3.46 测试 OSPF 协议

由上面三幅图可以看出，各 PC 机间能够互相通信，测试通过。

基本内容 3

实验步骤:

1.在基本内容 1 的基础上,对路由器 1 进行访问控制配置,使得 PC1 无法访问其它 PC,也不能 被其它 PC 机访问。

由实验指导手册可以得知配置访问控制列表的方法,下图为 RouterA 的访问控制列表。

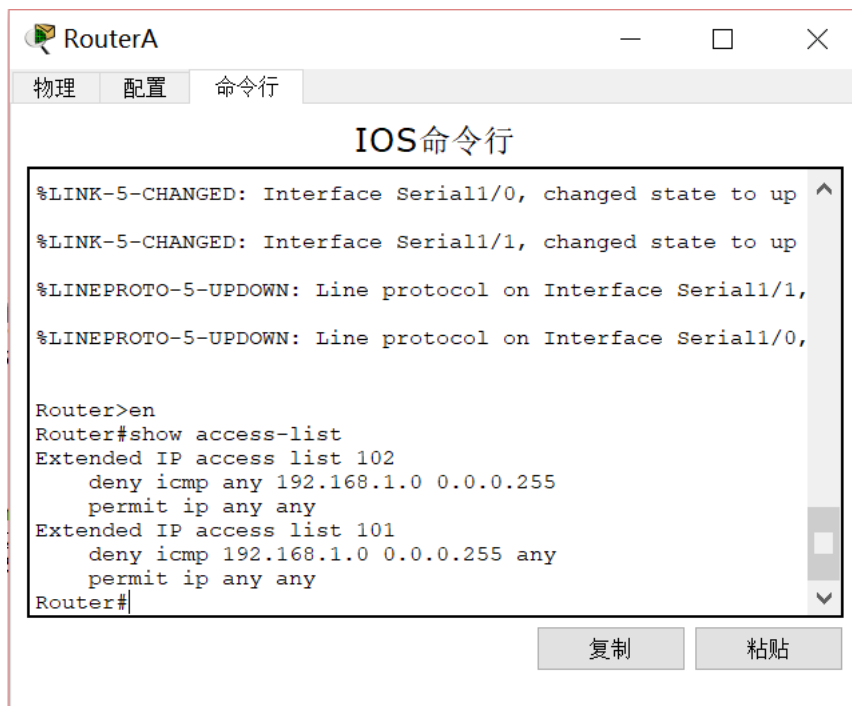


图 3.47 RouterA 访问控制列表

扩展控制列表 102 表示拒绝其他任何网段向 192.168.1.0/24 网段 (PC1 所在网段) 进行通信, 然后允许其他网段间相互通信。

扩展控制列表 101 表示拒绝 192.168.1.0/24 网段 (PC1 所在网段) 向其他任何网段进行通信, 然后允许其他网段间相互通信。

2.在基本内容 1 的基础上,对路由器 1 进行访问控制配置,使得 PC1 不能访问 PC2, 但能访问其 它 PC 机

由实验指导手册可以得知配置访问控制列表的方法, 下图为 RouterA 的访问控制列表。

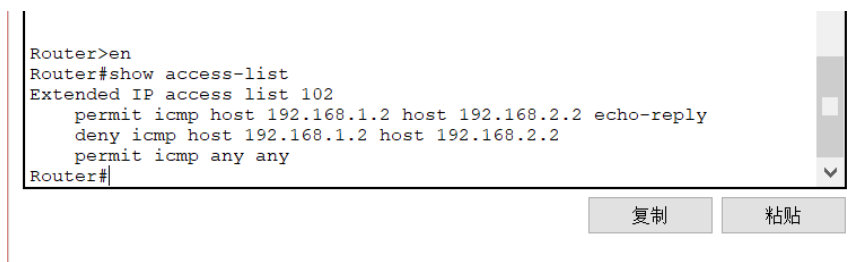


图 3.48 RouterA 访问控制列表

扩展控制列表 102 表示允许主机 PC1 对主机 PC2 的响应包通过, 然后拒绝主机 PC1 向 PC2

通信，然后允许其他网段之间相互通信。

结果分析：

1.实验要求 PC1 无法访问其他 PC，也不能被其他 PC 访问。

为了简洁，只给出两幅测试图满足必要性。如下图为 PC1 ping PC2 和 PC2 ping PC1 的情况，双方均无法 ping 通，测试通过。

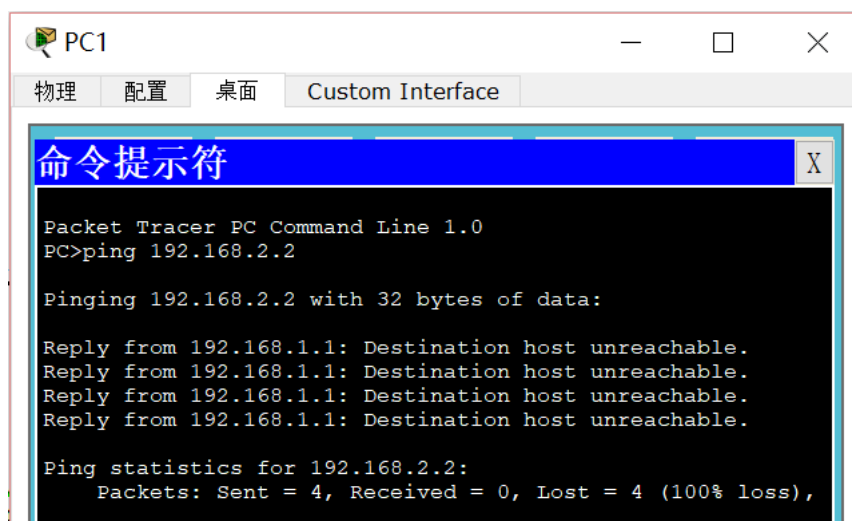


图 3.49 测试访问控制列表

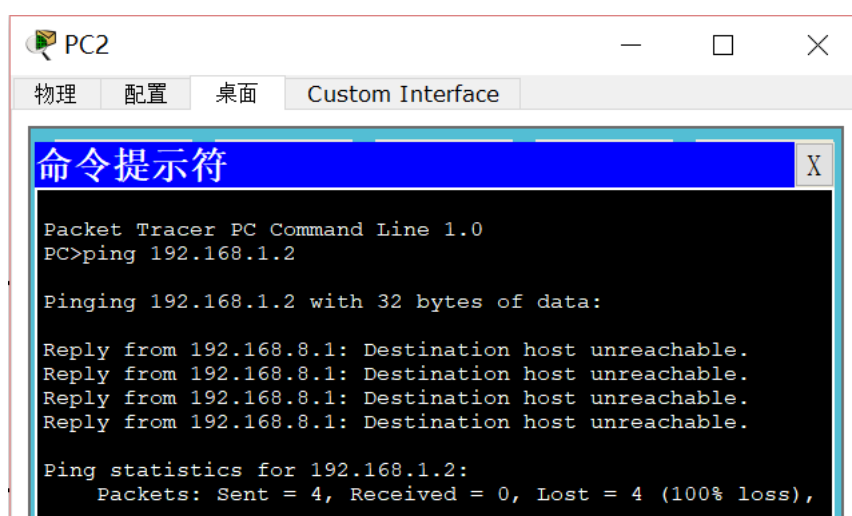


图 3.50 测试访问控制列表

2.实验要求 PC1 不能访问 PC2，但能访问其它 PC 机。

使用四个测试用例。分别为 PC1 ping PC2、PC1 ping PC3、PC2 ping PC1、PC3 ping PC1。下面几幅图分别表示测试情况：

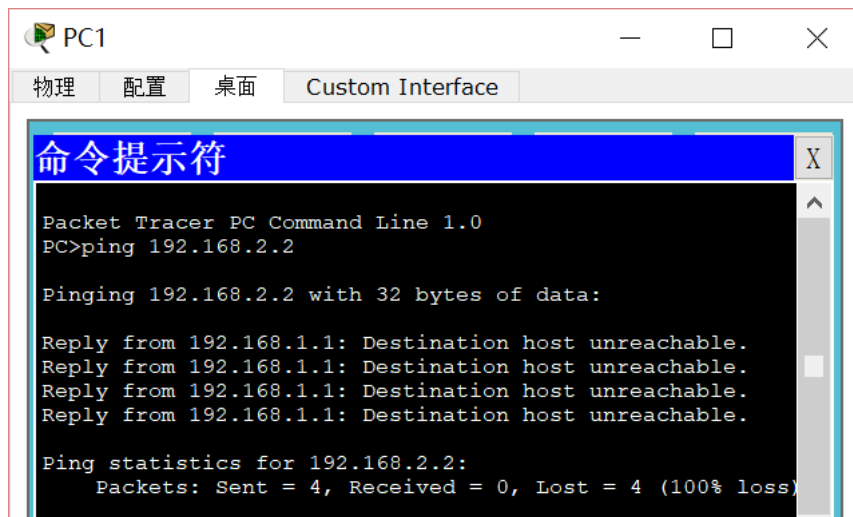


图 3.51 测试访问控制列表

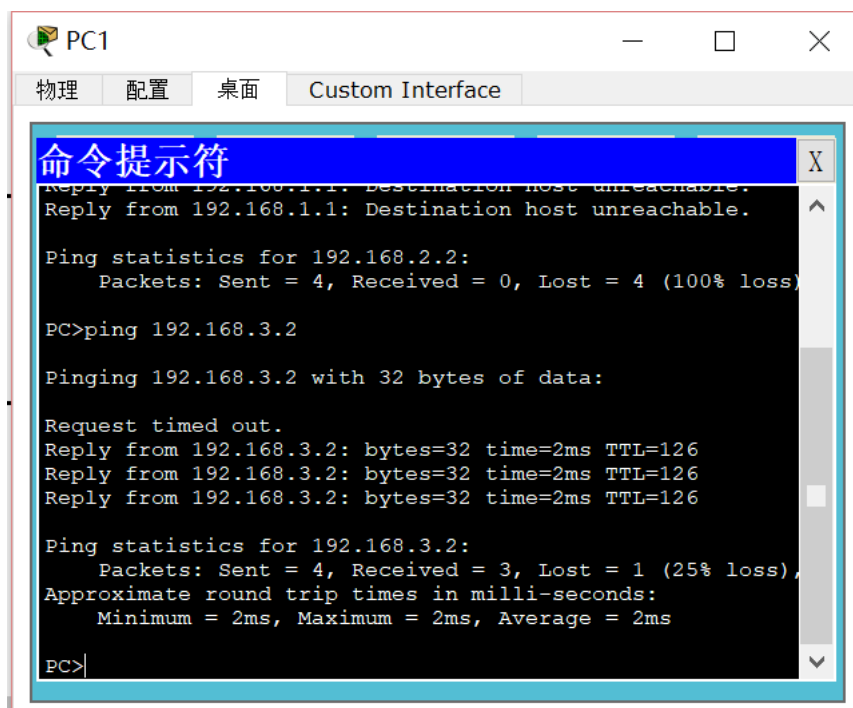


图 3.52 测试访问控制列表

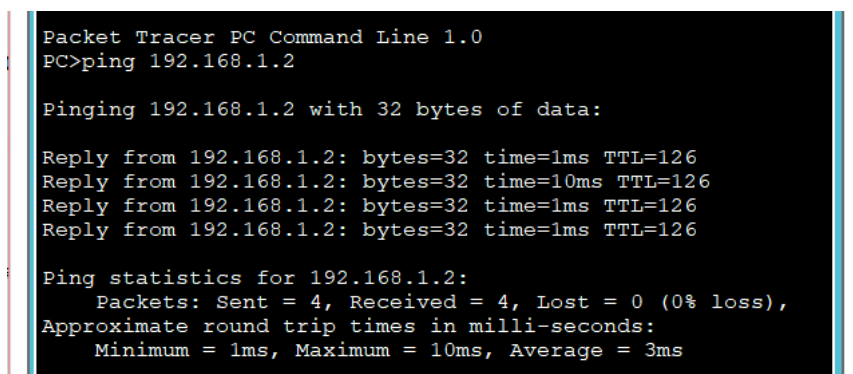


图 3.53 测试访问控制列表

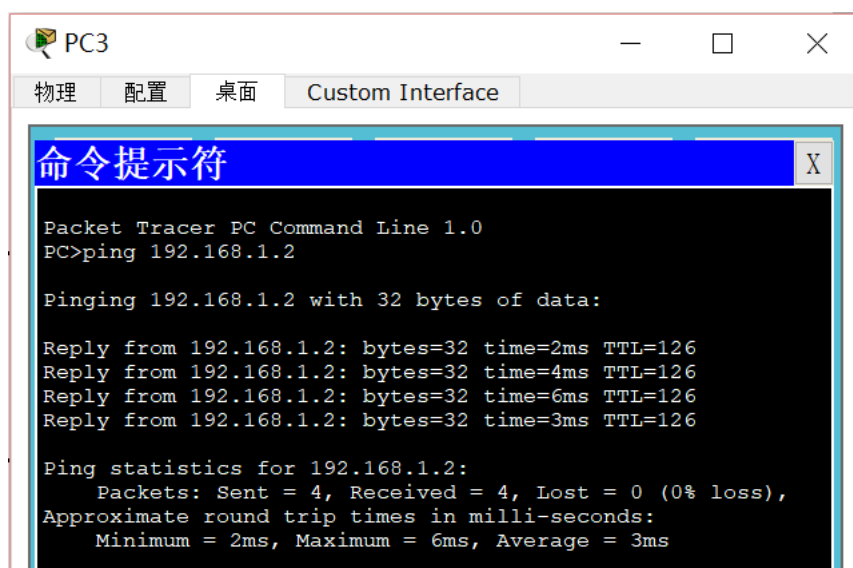


图 3.54 测试访问控制列表

由上面四幅图可知，PC1 不能 ping 通 PC2，但能 ping 通 PC3；PC2、PC3 均能 ping 通 PC1。测试通过。

3.4 综合部分实验设计、实验步骤及结果分析

3.4.1 实验设计

实验背景：

某学校申请了一个前缀为 211.69.4.0/22 的地址块，准备将整个学校连入网络。该学校有 4 个学院，1 个图书馆，3 个学生宿舍。每个学院有 20 台主机，图书馆有 100 台主机，每个学生宿舍拥有 200 台主机。

组网需求：

图书馆能够无线上网

学院之间可以相互访问

学生宿舍之间可以相互访问

学院和学生宿舍之间不能相互访问

学院和学生宿舍皆可访问图书馆。

实验任务要求：

完成网络拓扑结构的设计并在仿真软件上进行绘制(要求具有足够但最少的设备，不需要考虑设备冗余备份的问题)

根据理论课的内容，对全网的 IP 地址进行合理的分配

在绘制的网络拓扑结构图上对各类设备进行配置，并测试是否满足组网需求，如有无法满足之处，请结合 理论给出解释和说明

实验设计:

1.根据所学 IP 地址知识，对全网的 IP 地址进行如下划分，具体如下图所示：

	子网	网关
Apartment 1	211.69.4.0/24	211.69.4.1
Apartment 2	211.69.5.0/24	211.69.5.1
Apartment 3	211.69.6.0/24	211.69.6.1
Lib	211.69.7.0/25	211.69.7.1
Department 1	211.69.7.128/27	211.69.7.129
Department 2	211.69.7.160/27	211.69.7.129
Department 3	211.69.7.192/27	211.69.7.129
Department 4	211.69.7.224/27	211.69.7.129

图 3.55 IP 规划

经过检验，上图所示的 IP 划分完全可以满足组网需求。

2.完成网络拓扑结构的设计并在仿真软件上进行绘制

考虑到最少设备需求，每个学生宿舍有 200 台主机。假设每个交换机有 24 个接口，需要 10 台交换机（用一台交换机汇总其余 9 台交换机）。然而在实际绘制拓扑图的过程中，由于一张图无法容纳如此多的交换机，因此省略了一部分。

图书馆要求无线上网，因此配置了一台无线路由器。配置一台服务器，存放 CPT 软件公司自置的网页，用来检验上网需求是否得到满足。

最终的拓扑图如下图所示：

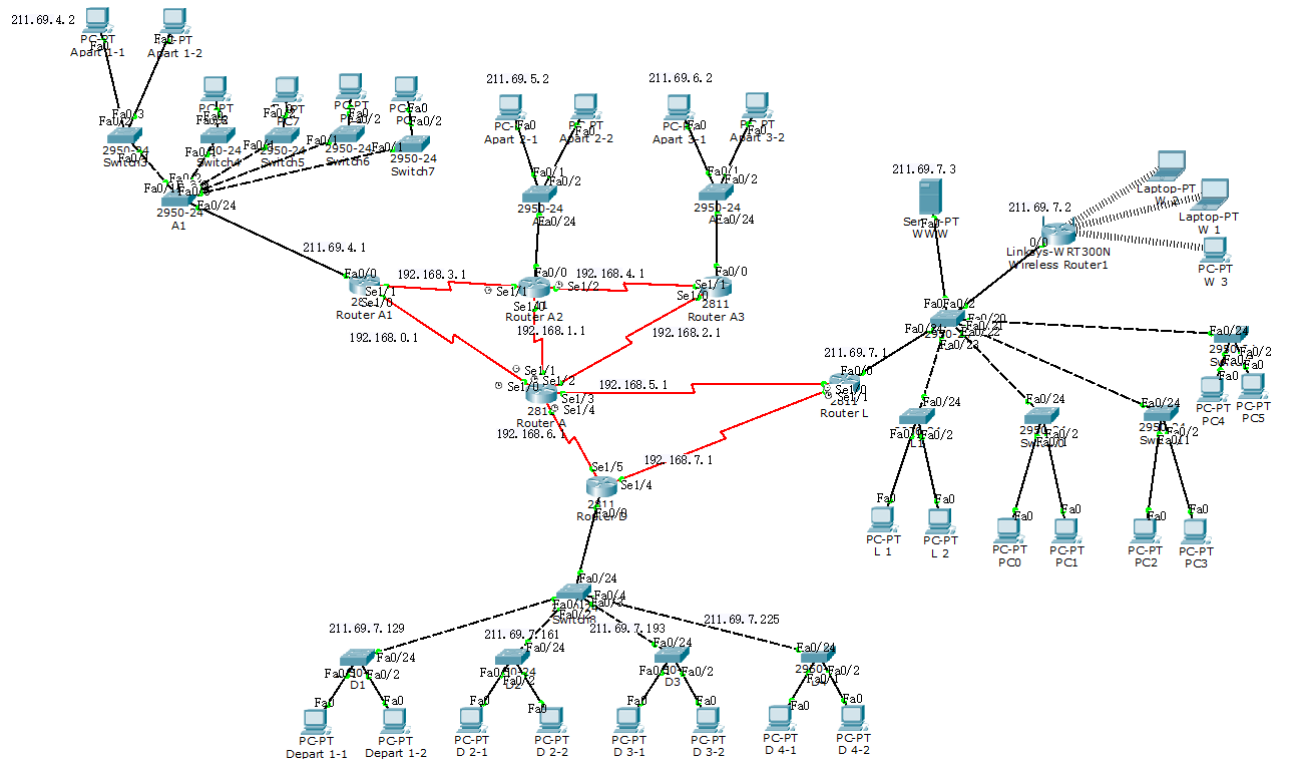


图 3.56 CPT 拓扑图

3.4.2 实验步骤

1.图书馆能够无线上网

图书馆被分配到了 211.69.7.0/25 这个网段。给图书馆路由器的以太网接口分配 IP 地址 211.69.7.1/25，作为网关。与宿舍和学院的路由器连接的串行接口分配 IP 地址 192.168.5.2/24 和 192.168.7.2/24。

接下来就是配置路由器的路由协议，在这里采用 OSPF 协议，原因是因为发现软件上的 RIP 协议无法分配后面八位的子网，这样在寻找学院的主机的过程中会产生一些麻烦。

OSPF 的配置方法在上面的实验中已经给出，在这里不给出详细实现过程。

下图为图书馆的路由器的路由表。

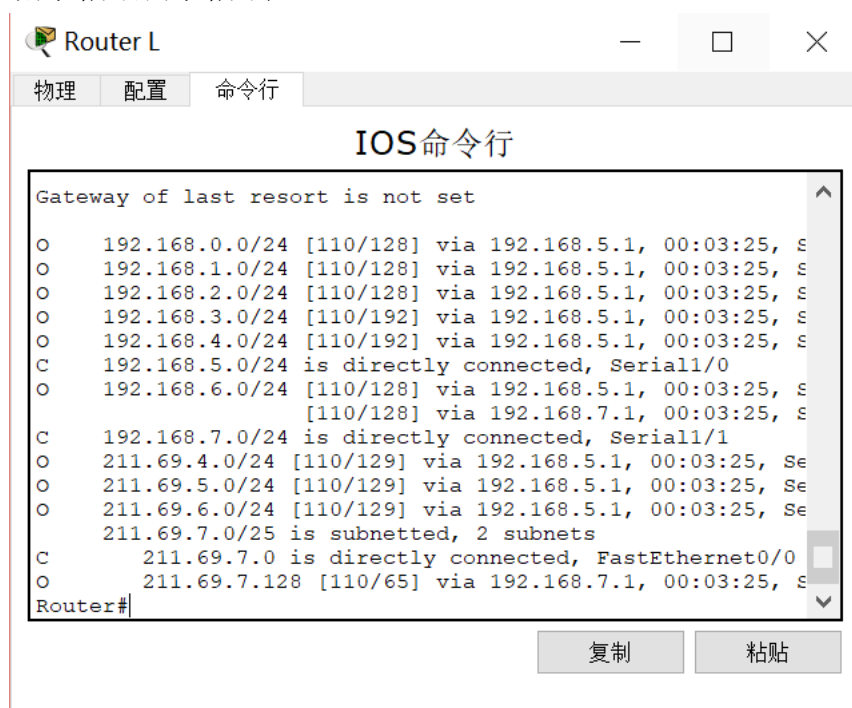


图 3.57 图书馆的路由器的路由表

接下来就是配置无线路由器以及服务器了。

将服务器的 ip 地址以及网关分别设为 211.69.7.3/25 以及 211.69.7.1。然后 HTTP 服务默认打开，如下图：

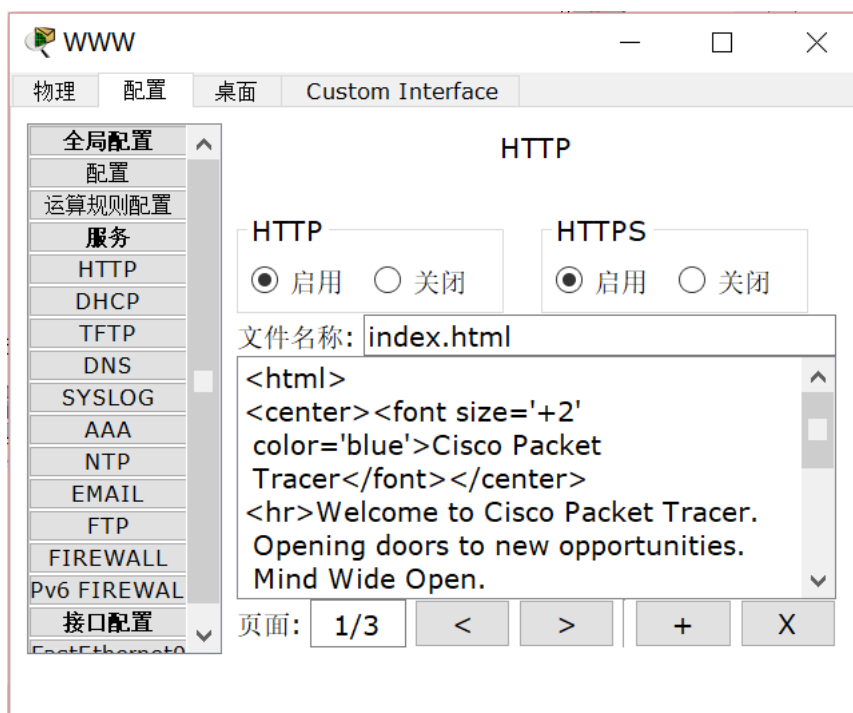


图 3.58 服务器

下面就是无线路由器的配置了，无线路由器的因特网 IP 地址分配为 211.69.7.2/25，默认网关为 211.69.7.1/25。它与其他 PC 分配 IP 的方式为 DHCP，起始 IP 地址为 192.168.0.100，最大用户数为 50，详细配置见下图：

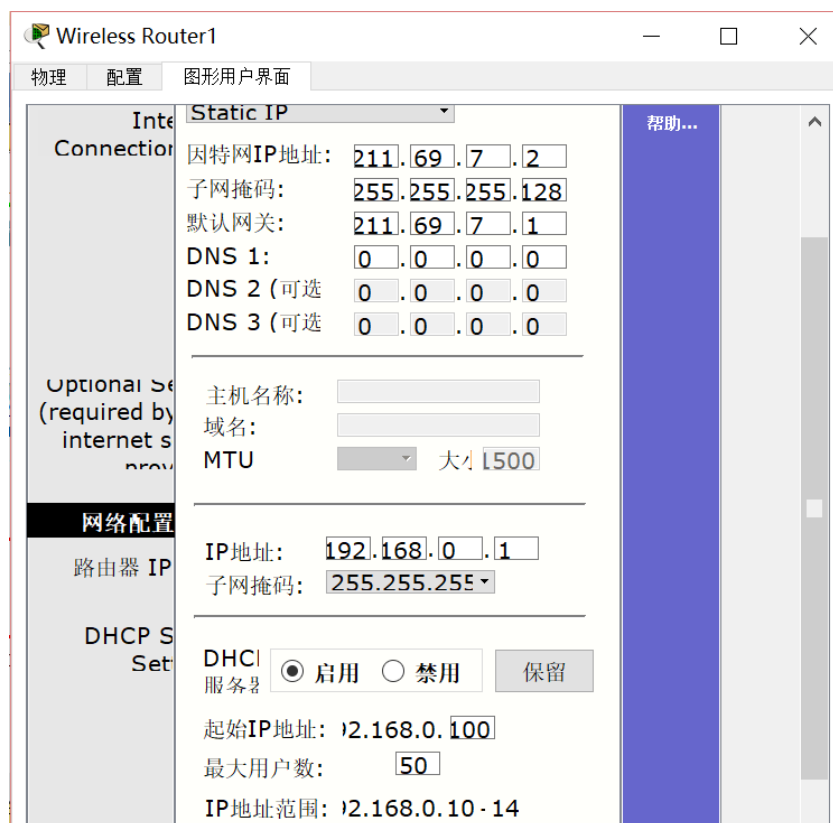


图 3.59 无线路由器配置

其余 PC 和交换机的配置在基础部分已经给出了详细的配置方法，在此不赘述。只是在配置能够无线上网的机器时，要给它换上无线网卡，如下图：



图 3.60 无线网卡配置

2.学院之间可以相互访问

给 4 个学院分配一个路由器，各个学院内部用交换机连接。

因此，所有学院的机器的网关为 211.69.7.129。对于学院里的机器，只是 ip 地址和子网掩码有些许不同而已，其余配置相同，下图为 Depart 1-1 的配置，其余 PC 类似。

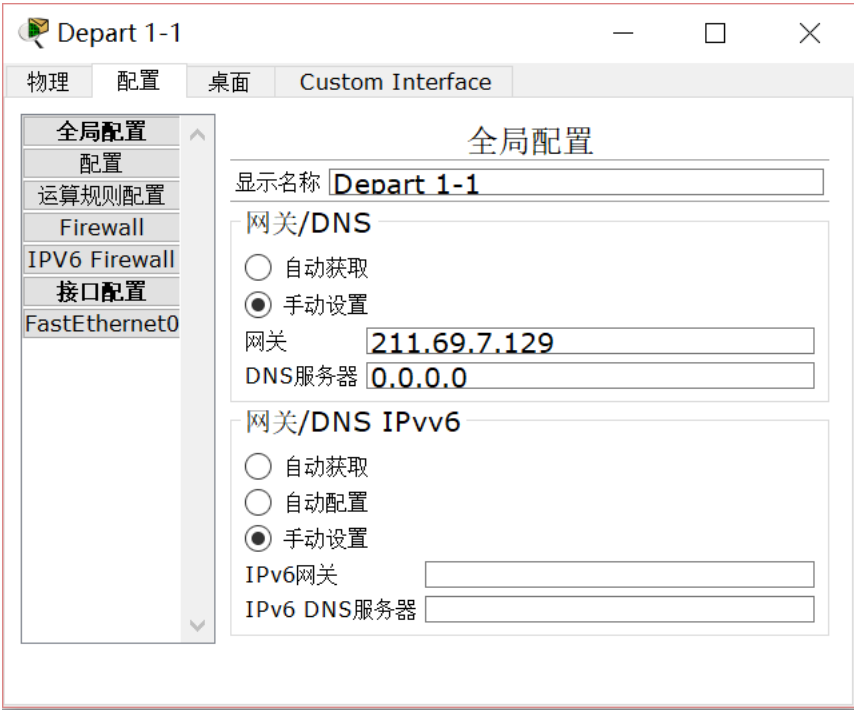


图 3.61 Depart 1-1 网关

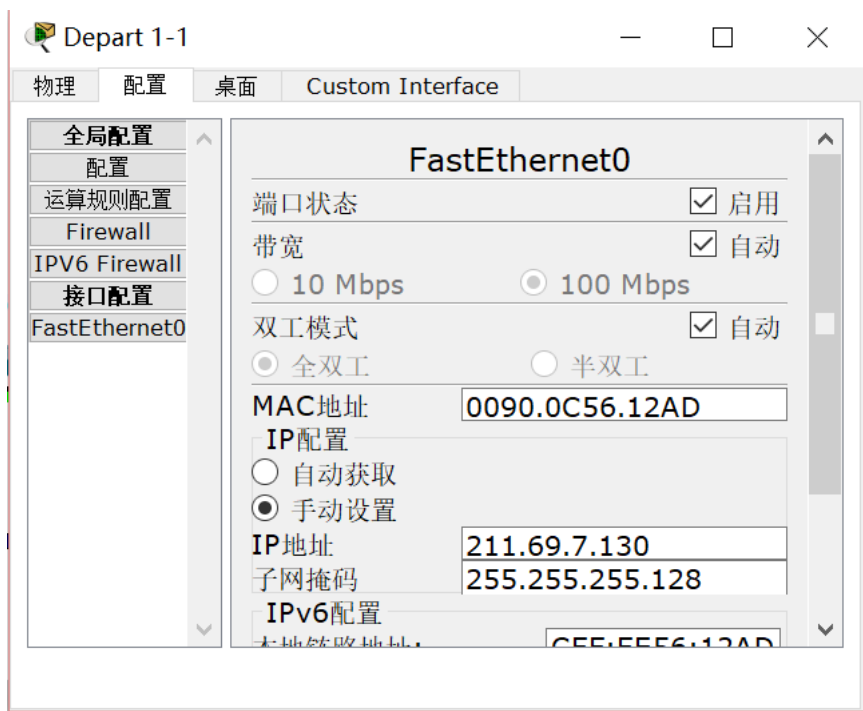


图 3.62 Depart 1-1 IP 设置

学院的路由器设置也与图书馆的路由器设置没什么区别，在此不赘述，下图为学院路由器跳转表。

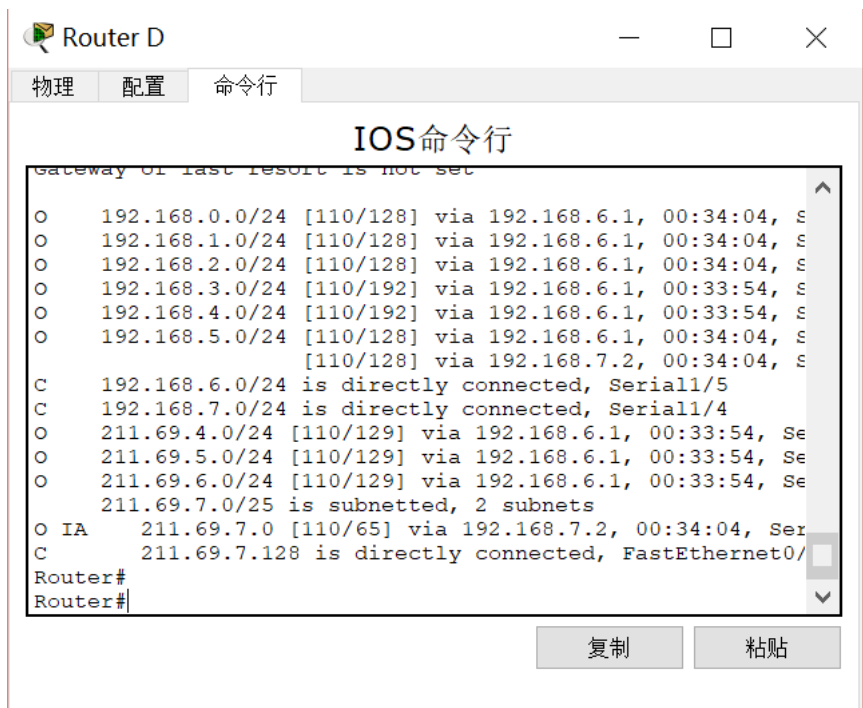


图 3.63 学院路由器跳转表

3.学生宿舍之间可以相互访问

给每个宿舍分配一台路由器，ip 地址分别为 211.69.4.1/24, 211.69.4.2/24, 211.69.4.3/24, 作为网关。

因此，不同宿舍的机器的网关不同，ip 地址与子网掩码也不一样，但是十分类似。下图为 Apart 1-1 的配置，其余 PC 类似。

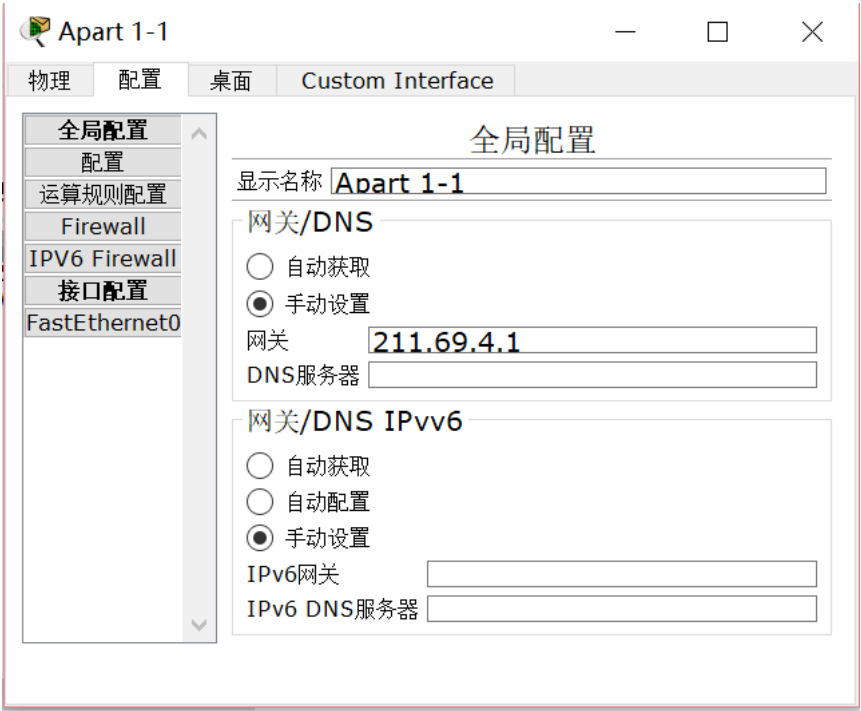


图 3.64 Apart 1-1 网关

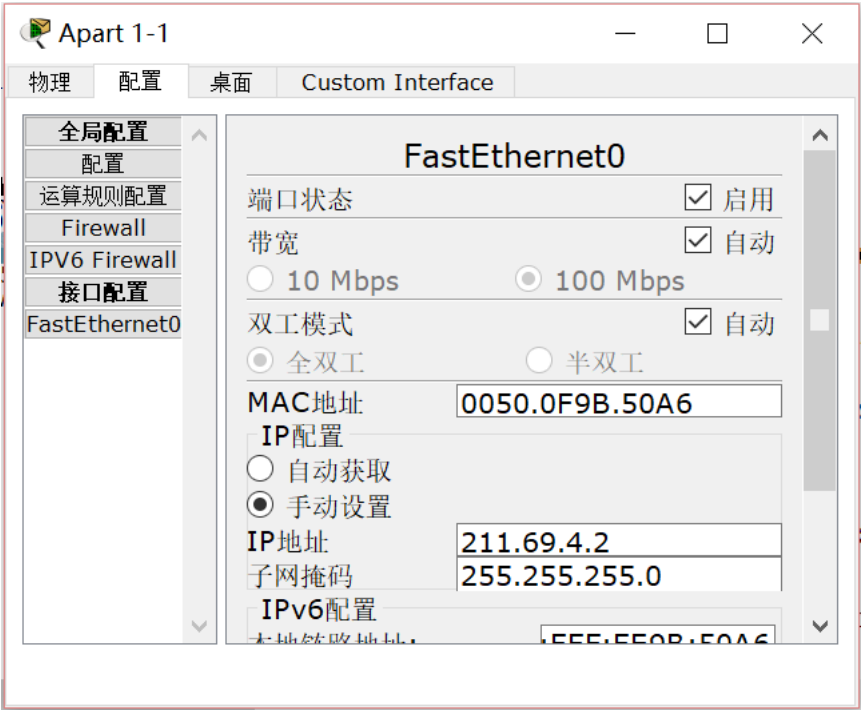


图 3.65 Apart 1-1 IP 设置

各宿舍的路由器设置也与图书馆的路由器设置没什么区别，在此不赘述，下图为宿舍 1 路由器跳转表。



图 3.66 宿舍 1 路由器跳转表

然后还用了一个路由器来汇总所有宿舍的路由器，在配置方面和其他路由器相同，下图为此路由器的跳转表。



图 3.67 宿舍路由器跳转表

4.学院和学生宿舍之间不能相互访问

根据基础部分学到的访问控制列表的配置方法，在学院的路由器上设置访问控制列表，如下图所示：



图 3.68 学院的路由器上设置访问控制列表

标准访问控制列表 1 表示，拒绝网段 211.69.4.0/24 的通信请求，然后拒绝网段 211.69.5.0/24 的通信请求，然后拒绝网段 211.69.6.0/24 的通信请求，然后允许其他网段的通信请求。

5.学院和学生宿舍皆可访问图书馆。

经过上述配置，学院和学生宿舍已经能访问图书馆，具体看结果分析。

3.4.3 结果分析

1.图书馆能够无线上网

让连接上了图书馆 WiFi 的机器去 ping 宿舍的机器，看能否 ping 通。

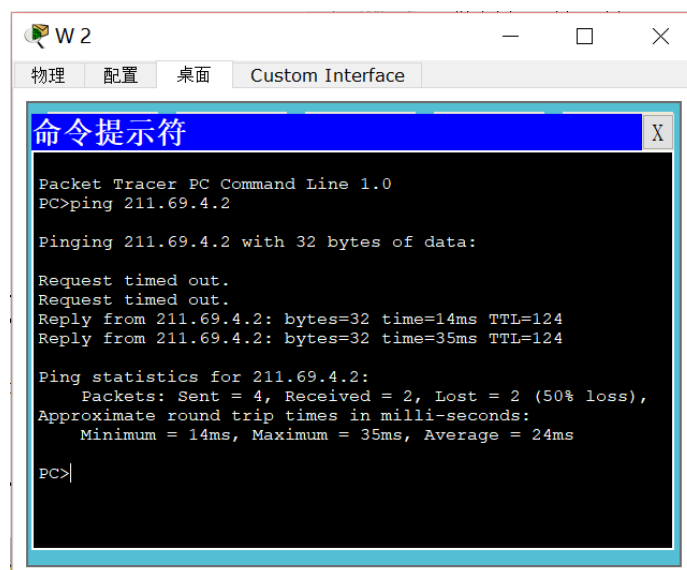


图 3.69 测试无线上网

由上图所示，W2 能 ping 通宿舍里的机器，证明其能无线上网，测试通过。

2.学院之间可以相互访问

让所属网段 211.69.7.129/25 的机器互 ping 所属网段 211.69.7.161/25 的机器，即 Depart 1-1 互 ping D 2-1，观察其通信情况。

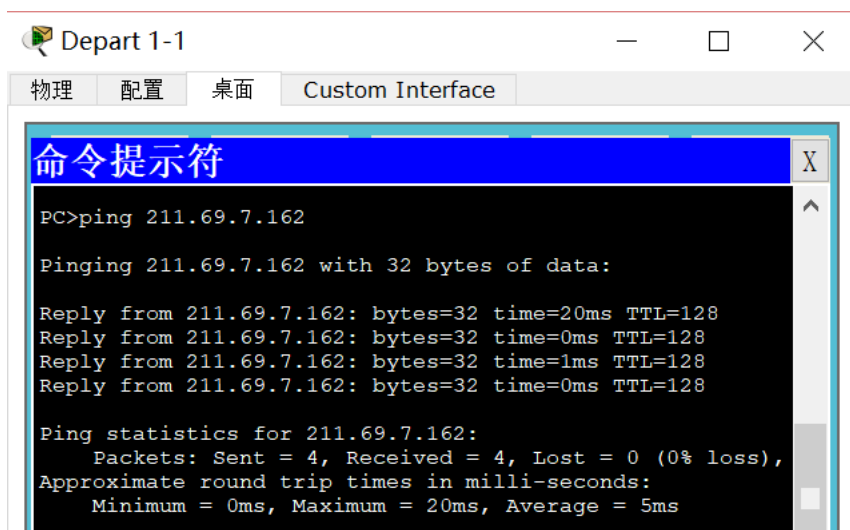


图 3.70 测试学院之间是否可以相互访问

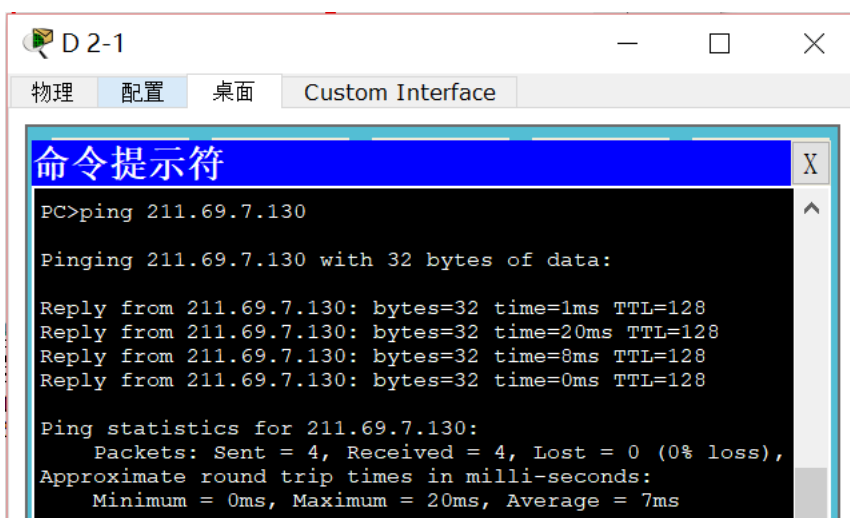


图 3.71 测试学院之间是否可以相互访问

可以看到通信正常，说明不同学院之间可以相互访问，测试通过。

3.学生宿舍之间可以相互访问

让所属网段 211.69.4.0/24 的机器互 ping 所属网段 211.69.5.0/24 的机器，即 Apart 1-1 互 ping Apart 2-1，观察其通信情况。

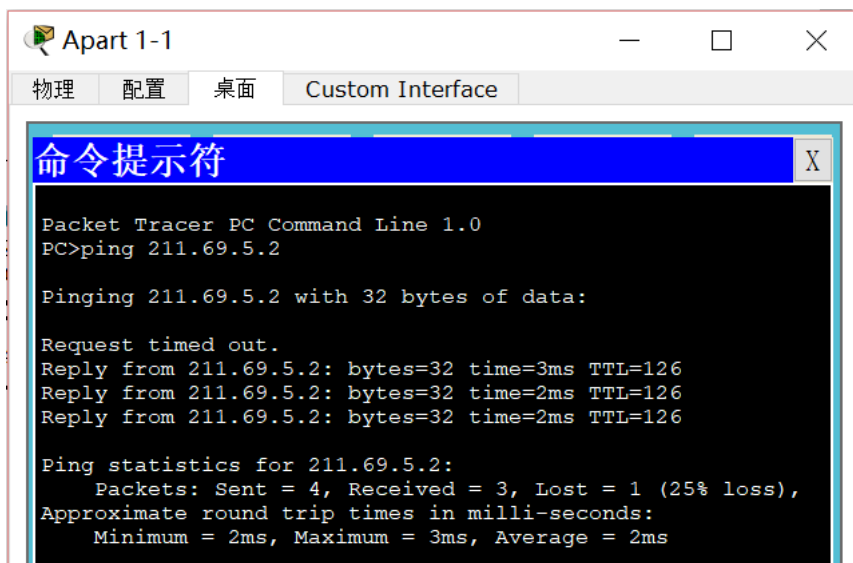


图 3.72 测试宿舍之间是否可以相互访问

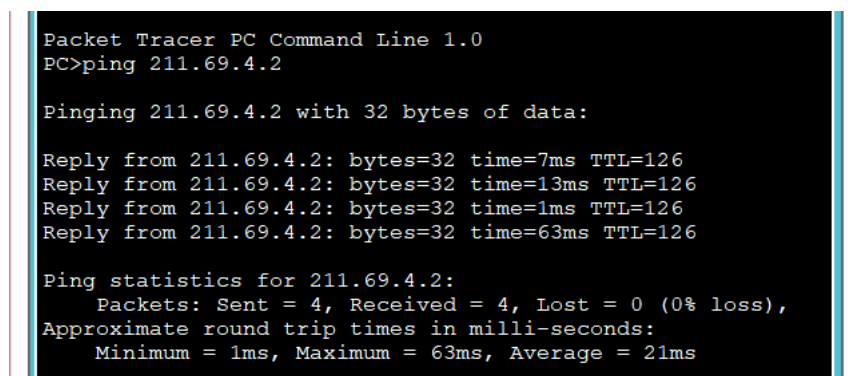


图 3.73 测试宿舍之间是否可以相互访问

可以看到通信正常，说明不同宿舍之间可以相互访问，测试通过。

4.学院和学生宿舍之间不能相互访问

让学院的机器 Depart 1-1 和宿舍的机器 Apart 1-1 互 ping，观察通信情况。

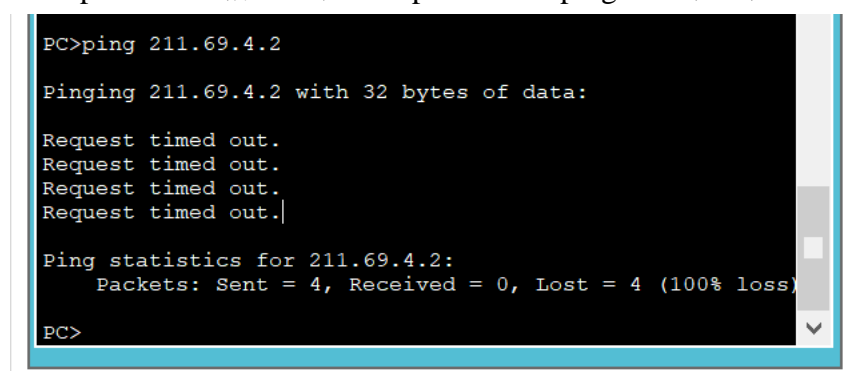


图 3.74 测试学院和学生宿舍之间不能相互访问

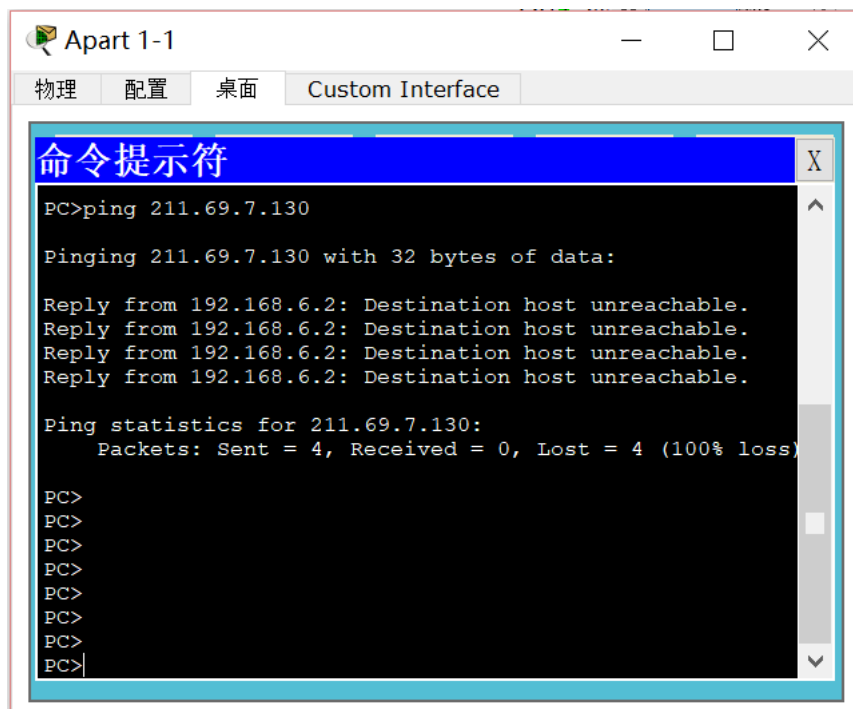


图 3.75 测试学院和学生宿舍之间不能相互访问

可以看出，宿舍 ping 学院显示不可达，而学院 ping 宿舍全部超时（回应包被学院路由器丢掉），说明学院和学生宿舍之间不能相互访问，测试通过。

5.学院和学生宿舍皆可访问图书馆

让宿舍和学院的机器去访问图书馆的服务器（CPT 内置网页），观察访问情况。

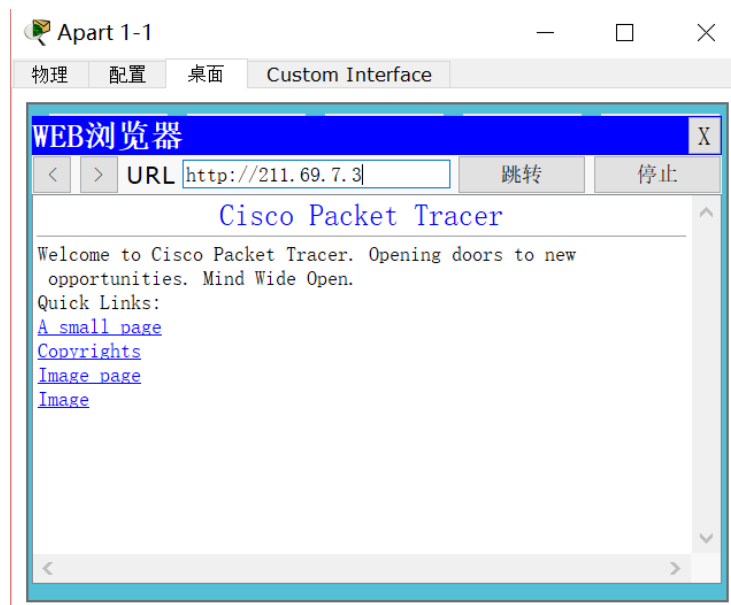


图 3.76 测试学院和学生宿舍是否皆可访问图书馆

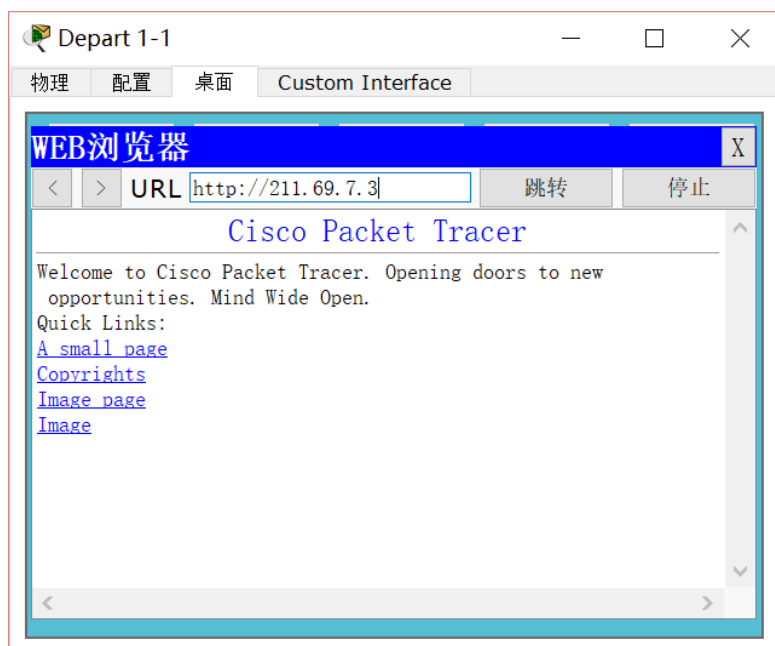


图 3.77 测试学院和学生宿舍是否皆可访问图书馆

发现学院和宿舍的机器均能正常访问，测试通过。

3.5 其它需要说明的问题

在实际绘制拓扑图的过程中，由于一张图无法容纳如此多的交换机，因此省略了一部分。能够用图上的机器测试所要求。

在图书馆仅仅配置了一台服务器，其中只包含了内置的网页。目的是测试学院和宿舍能不能访问图书馆。（ping 其实也能测试）

心得体会与建议

4.1 心得体会

在实验一中,在课堂上只是了解了 socket 的基本知识,然后在实验课上就要自己基于 socket 实现一个 web 服务器,尽管有那张服务器与客户端的 socket 图,想要真正实现还是很有难度的。在看懂了实验框架之后,才开始真正的编写程序,一开始并不知道怎么下手,感觉到无法入手,就上网查了很多相关资料。发现要做的关键的事情就是在请求报文中解析目录名,这个一开始想了一会儿,后来仔细分析了请求报文的格式后,就明白了怎么去确定这一段目录。然后另一关键就是能够显示多媒体文件,在查阅了资料之后,知道了 content-type 这一项,于是按照资料所写,发现自己编写的程序能够显示多媒体文件了,就很有成就感。然后就是多线程的处理,这个一开始认为很难,但是后来发现其实也不是想象中的那样,明白了如何使用已有的库函数之后就能解决了。总的来说,本次实验有一定的难度,在初次尝试时经常不知道该干什么,主要就是由于各种函数的作用不清楚,要调用很多 socket 函数,在之后熟悉了一些函数之后,情况也就有所好转了。在编写完自己的服务器后,还了解到性能相关的问题,知道并不是能够显示正确就行了,还需要具备很强的异常处理能力以及快速响应的优点。

在实验二中,实验要求我们能够根据课堂教学内容,自己动手实现 GBN,SR,TCP 协议的设计。一开始觉得这个实验比较简单,因为在课堂上我们都熟悉过了 GBN 的状态转移图,觉得自己已经对这个协议有了比较清晰的认识,只需要按照状态转移图来设计就会顺利实现了。但是在实际的编程过程中,并不是这么顺利的。首先我们要解决的问题是,熟悉各种调用函数,有些函数我们在状态图中见不到,因为它的实现被省略了,状态图只是一个思想,与实际的设计有所不同。在熟悉了调用函数后,我们才能进行协议的设计,在这里有一个地方就是只有一个定时器,而不是对所有的包都设置一个定时器,而且这个定时器要重新开启的话必须先关闭再开启,否则就会出现错误。在完成了 GBN 的设计后,就要开始进入 SR 的设计了,SR 的设计我认为在这个实验中算是比较复杂的,因为它的缓冲区比较复杂,选择重传并不像 GBN 那样,如果超时了,重传缓冲区中的所有报文就可以了,它更加复杂。我们需要对它在接收方的缓冲区进行规划,到来报文应该放在缓冲区的哪个位置才能不造成冲突,我们要考虑这些细节。最后就是 TCP 的设计了,因为它是基于 GBN 协议进行设计的,所以大部分设计都不需要重新做。需要做的是加入了快速重传机制,这只需要对重复 ACK 计数即可,到达 3 次后,我们就可以立即重新发送要求的报文。经过这次实验,我对计网的一些协议有了更深入的了解,有了更深刻的印象。

在实验三中,我熟悉了 Cisco Packet Tracer 仿真软件。这个软件功能强大,能够模拟网络环境,在我们学习网络的过程中能起到一个很好的作用。实验要求通过课堂上学到的 IP 地址

规划与路由器的相关知识，结合软件进行实验。在进行第一项实验——IP 地址规划与 VLAN 分配实验时，初次在 CPT 上配置了一台 PC 机的 ip 地址与网关，然后通过配置路由器的接口的 ip 地址，实现了两个不同子网内的机器的通信，发现理论是正确的时候，很有成就感。然后在划分 VLAN 的时候，由于第一次不知道如何实际划分 VLAN，出了一点小问题，在按照实验指导手册上的步骤一一实现的时候，一开始并不清楚这些命令行是什么意思，并不是很理解它背后的含义，在查阅了相关资料后，发现 VLAN 划分其实也是很简单的。要进行这个实验的目的是研究不同 VLAN 之间是否可以通信，学习了课堂上的理论知识，我们知道这是行不通的，通过实践也确实验证了这个理论。然后下一步就是要配置路由器，使得不同 VLAN 之间可以通信，这个实验是层层递进的，在给路由器的接口分配 ip 地址的时候，遇到了一点问题，因为有一个接口要连接两个 VLAN，而一个接口只有一个 ip，很显然这样是不行的，然后实验指导手册里又给出了子接口的方法，一个接口可以分成很多子接口，可以将一个子接口与一个 VLAN 封装在一起，这样就可以实现一个借口连接多个 VLAN 了。在路由器配置实验中，在看了指导手册上的内容后，知道了怎么配置 RIP 和 OSPF 协议，也能在配置了协议后，实现不同网段间的通信。有点遗憾的就是，理论课上给出了两种协议的不同之处，这一点在实验中没有体现出来，而仅仅是配置了相关协议。然后在这个实验里有点创新的就是加入了访问控制列表，这在实际网络是非常常见的，有时处于某种意愿要进行访问控制。然后通过查阅资料，实现了单向通信（即一方可以访问另外一方，而另外一方不能访问这方）。在进行综合实验的时候，与课程结合最紧密的就是要自己进行 IP 地址的规划，这也是考核要求。在进行了 IP 地址规划后，还要考虑设备的选择，了解到了路由器和交换机的实际使用场合，也知道路由器是很昂贵的，和我们平时在家里使用的不同，不能随意使用。实验中还要求图书馆能够无线上网，这也和课程里 DHCP 相关内容有一定的结合。其余的要求，只要完成了基本部分，就应该能顺利完成。

4.2 建议

实验一可以考虑多给出一些参考资料，便于我们能够快速了解要完成的工作，能够快速上手。也可以给出详细的说明，比如在异常处理中应该实现哪些功能。

实验三路由器配置实验中配置 RIP 协议和 OSPF 协议可以在内容中加入两种协议的对比。在最终实验中，交换机需要的台数较多，可以考虑不必画出所有的交换机，而且配置起来也较费时间。