
华中科技大学计算机学院

《计算机通信与网络》实验报告

姓名 吕鹏泽 班级 CS1601 学号 U201614532

项目	Socket 编程 (30%)	数据可靠传输协议设计 (15%)	CPT 组网 (15%)	实验报告 (20%)	平时成绩 (20%)	总分
得分						

教师评语：

教师签名：

给分日期：

目 录

实验一 SOCKET 编程实验	1
1.1 环境	1
1.2 系统功能需求	1
1.3 系统设计	2
1.4 系统实现	5
1.5 系统测试及结果说明	11
1.6 其它需要说明的问题	13
实验二 数据可靠传输协议设计实验	14
2.1 环境	14
2.2 实验要求	14
2.3 协议的设计、验证及结果分析	14
2.4 其它需要说明的问题	29
实验三 基于 CPT 的组网实验	30
3.1 环境	30
3.2 实验要求	30
3.3 基本部分实验步骤说明及结果分析	33
3.4 综合部分实验设计、实验步骤及结果分析	41
3.5 其它需要说明的问题	46
心得体会与建议	47
4.1 心得体会	47
4.2 建议	47

实验一 Socket 编程实验

1.1 环境

1.1.1 开发平台

操作系统：Microsoft Windows 10 家庭中文版 (64 位)
处理器：Intel(R) Core(TM) i5-8250U CPU @ 1.60GHz(1800 MHz)
内存：8.00 GB (2400 MHz)
显卡：NVIDIA GeForce MX150
开发平台：Microsoft Visual Studio Community 2017 版本 15.8.8, Qt 5.8.0
编程语言：C++
第三方组件：Qt VS Tools

1.1.2 运行平台

操作系统：Microsoft Windows 10 家庭中文版 (64 位)
处理器：Intel(R) Core(TM) i5-8250U CPU @ 1.60GHz(1800 MHz)
内存：8.00 GB (2400 MHz)
显卡：NVIDIA GeForce MX150
测试平台：Chrome 版本 71.0.3578.98
第三方组件：Qt5Core.dll, Qt5Widgets.dll, Qt5Gui.dll

1.2 系统功能需求

题目按难度分为三级，其中第一级为基本级，第三级为最难级，需依次完成该题目各个级别的实验。三个级别评分分别为 30%、50%、20%。

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、监听端口和虚拟路径。
- ✧ 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

总体架构：

本次实验需要完成一个 Web 服务器，服务器要求能配置 ip 地址，端口等信息，可以使用一个类来描述配置信息。本次编写的 Web 服务器是基于 TCP 的流类型 Socket 编程，连接的创建需要经过初始化、创建监听 SOCKET、监听、绑定、等待连接等流程，因此可以将服务器封装成一个类，服务器的启动、配置和关闭通过公有类函数的接口实现，图形化界面只需将按钮和接口函数绑定即可。服务器需要接受 http 报文进行分析并回复请求的报文，可以使用专门的函数来处理 http 请求报文。综上，服务器可以分为图形模块、连接建立模块、报文处理模块。图形模块为 Web 服务器的前端，连接建立和报文处理为服务器的后端。为了方便程序的编写，我决定将前端和后端分离开来，首先完成控制台形式的 Web 服务器，在测试完基本功能正确无误后再为服务器增加图形界面，此外，为了方便增加图形界面，需要将 Web 服务器的封装成一个类。服务器的关闭可以通过关闭服务器的套接字来实现。系统执行的总体架构图如图所示：

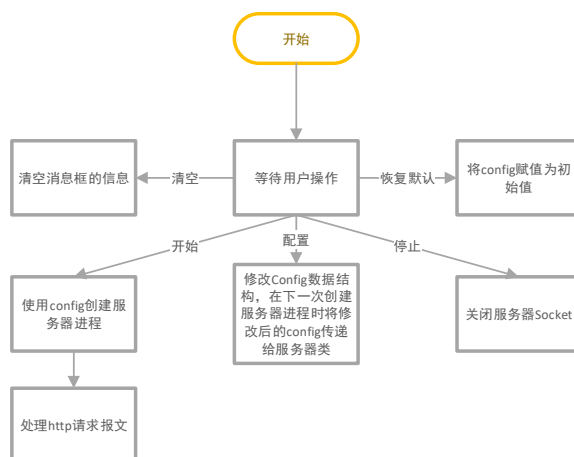


图 1-1 系统总体架构图

建立连接设计：

本实验的 Web 服务器基于 TCP 的流类型 Socket 编程，由于流式套接字使用的是基于连接的协议，所以必须首先建立连接，而后才能从数据流中读出数据，建立连接的过程可概括为 WSAStartup 初始化、bind 绑定监听、listen 监听请求、accept 获取请求、sendto 回复报文、recv 接收报文、closesocket 关闭会话等几个步骤，具体的服务器创建流程图如图所示：

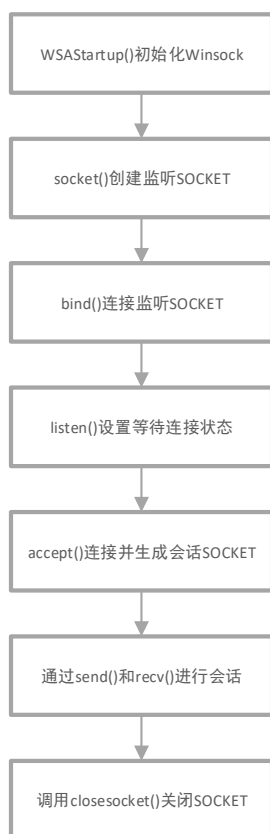


图 1-2 服务器连接建立

在具体实现时，需要增加每一步是否执行成功的判断，若执行失败则抛出异常并退出。并

且将服务器的建立和关闭封装成一个类，提供简便的接口以方便图形框架的调用。

会话处理设计：

listen()设置完毕后，服务器将进入无限循环之中，重复执行接收会话、处理会话、接收会话、处理会话...的过程，直到服务器被关闭。每当收到一个会话时，服务器进程就创建一个线程处理该会话，然后继续监听会话请求，等待下一个连接请求。会话的处理过程流程图如图 1-3 所示：

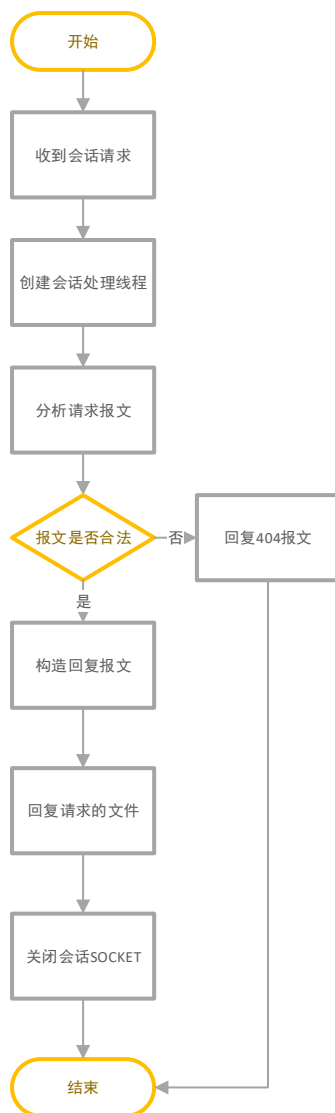


图 1-3 会话处理流程图

每当服务器线程收到一个新的会话请求时，就会建立一个线程处理该会话，处理过程首先会在消息框中输出请求报文，然后根据请求报文的内容判断请求的文件是否存在，如果存在，则构造回复报文并将文件发送过去。如果不存在就回复 404 报文。

图形界面设计：

本次实验中我使用 Qt 进行图形界面的开发。进行到这一步时 Web 服务器的后端代码已经实现，并且提供了服务器启动和关闭的接口，因此图形界面只需要将界面按钮和服务器的启动/关闭函数绑定即可。图形界面的设计我将使用 Qt Designer 进行设计，如图 1-4，通过这个软件我可以直接拖动按钮、文本框、标签进行界面设计，设计完毕后程序会自动生成组件的相应代码，然后就可以使用 `connect` 函数将组件对象的事件和对应的服务器接口函数绑定，即可完成服务器的图形框架。此外，由于 `cout` 是默认输出到控制台中的，因此需要使用经过重定向到文本框的 `Qdebuf()` 将服务器中的 `cout` 替换，以将消息输入到图形界面的文本框之中。

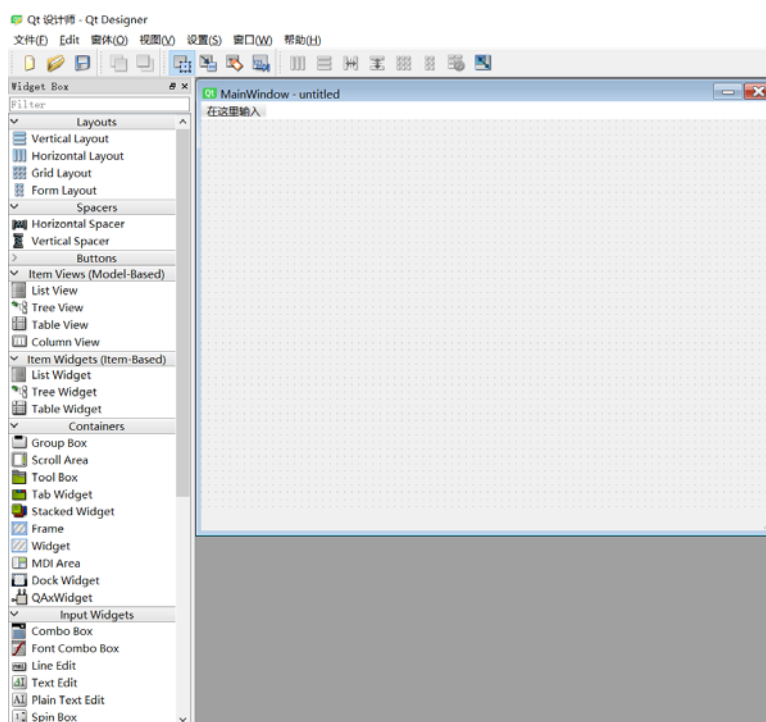


图 1-4 Qt Designer 界面

1.4 系统实现

Config 类实现:

Config 类是服务器的配置信息，当服务器创建连接时将使用 Config 对象的信息进行连接的建立。外部函数通过 `setConfig()` 来修改服务器的配置信息，在下一次服务器创建连接时服务器的配置信息将会生效。Config 类的定义如下：

```
struct Config
{
    string SERVERADDRESS;
    int PORT;
    string HOMEDIR;
```

```

int MAXCONNECTION;
Config(string srvAddr, int Port, string HDir, int MaxCon);
Config(void) {};
~Config(void) {};
};

```

Config 类中各成员的意义如下：

- ✧ SERVERADDRESS 是点分十进制表示的 IP 地址
- ✧ PORT 为应用程序端口
- ✧ HOMEDIR 为服务器工作的主目录
- ✧ MAXCONNECTION 为服务器可创建的最大会话个数

程序中声明了 3 个 Config 对象，Server.cpp 中的 Config 对象用于服务器的创建，可被外部程序通过调用 setConfig() 进行修改。QtGuiApplication1.cpp 创建了两个 Config 对象，一个被定义为 const 类型，用于恢复默认的功能。另一个用于修改配置，并调用 setConfig() 将值传递给 Server 类。

服务器类实现：

Server 类对外提供了 WinsockStartup, ServerStartup, ListenStartup, Loop, setConfig, getConfig 6 个接口函数，其中前四个成员函数用于服务器的初始化和创建，setConfig 用于服务器的配置，getConfig 用于获取服务器当前的配置。各成员函数的具体介绍及实现如下：

◆ int WinsockStartup()

参数：无

操作：初始化 Winsock

返回值：初始化成功返回 0，初始化失败返回-1

流程图：如图 1-5 所示

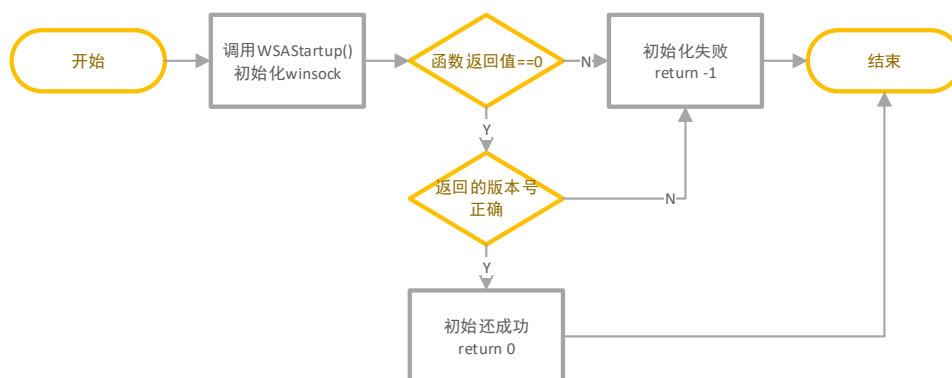


图 1-5 WinsockStartup 流程图

◆ int ServerStartup()

参数：无

操作：初始化 Server，创建 SOCKET，绑定 IP 和 PORT

返回值：执行成功返回 0，执行失败返回-1

流程图：如图 1-6 所示

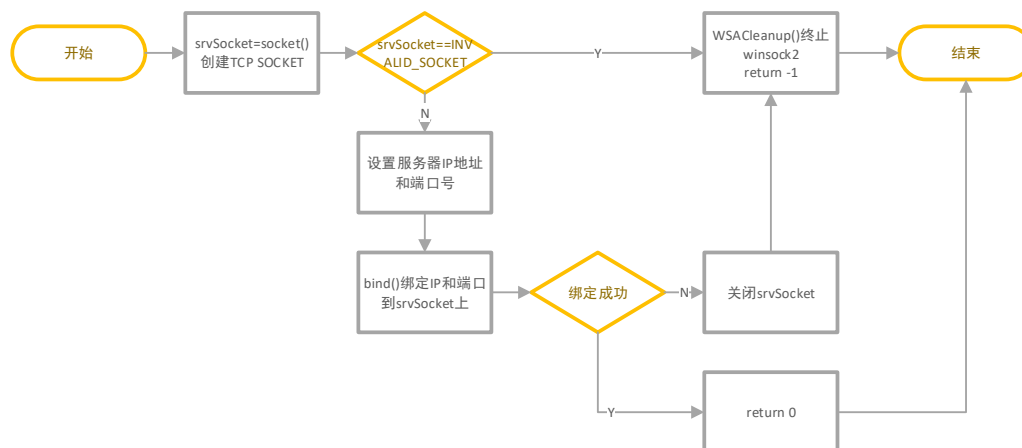


图 1-6 ServerStartup 流程图

◆ int ListenStartup()

参数：无

操作：开始监听客户端请求

返回值：执行成功返回 0，执行失败返回-1

流程图：如图 1-7 所示

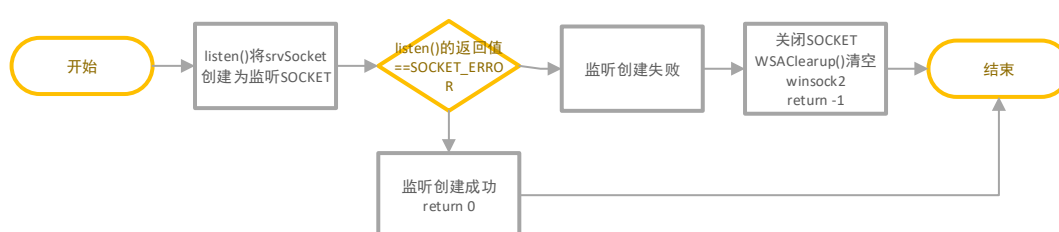


图 1-7 ListenStartup 流程图

◆ int Loop()

参数：无

操作：循环执行“等待客户端请求”->“新建线程回复报文”->“等待客户端请求”

返回值：执行成功返回 0，执行失败返回-1

流程图：如图 1-8 所示

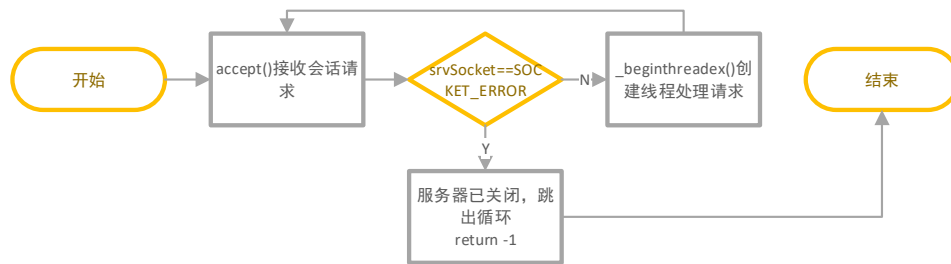


图 1-8 Loop 流程图

- ◆ **void setConfig**
参数: Config cfg
操作: 修改服务器配置类
返回值: 无
流程图: 如图 1-9 所示



图 1-9 setConfig 流程图

- ◆ **Config getConfig()**
参数: 无
操作: 返回服务器配置
返回值: config
流程图: 如图 1-10 所示

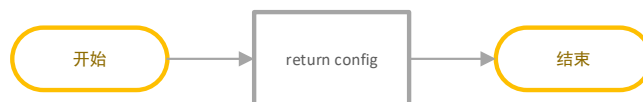


图 1-10 getConfig 流程图

会话处理实现:

由于 `_beginthreadex()` 不能将类成员函数作为其参数, 因此采取普通函数实现消息处理, 包括 `ProcessRequest`, `SendErrorMsg`, `SendData`, `ContentType`。其中 `ProcessRequest` 是创建多线程后执行的入口函数, 其实现输出报文内容、分析报文信息, 然后调用 `SendData()` 发送数据, 如果报文不合法由 `SendData()` 调用 `SendErrorMsg()` 发送 404 界面。 `ContentType()` 用于获取请求文件的拓展名。各成员函数的具体介绍及实现如下:

- ◆ **unsigned WINAPI ProcessRequest(void *arg)**
参数: 指针 `arg`, 指向一个结构体, 包含客户端的 IP、端口和会话 SOCKET
操作: 输出请求报文, 分析报文内容并回复相应信息

返回值：1

流程图：如图 1-11 所示

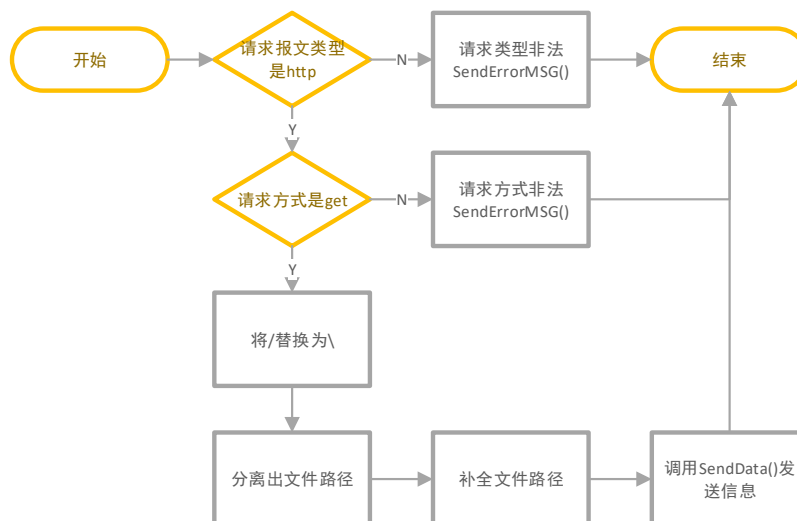


图 1-11 ProcessRequest 流程图

◆ string ContentType(string filename)

参数：文件名或文件路径 filename

操作：获取目标文件的拓展名

返回值：目标文件的拓展名

流程图：如图 1-12 所示

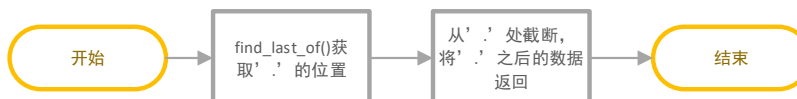


图 1-12 ContentType 流程图

◆ string SendData(SOCKET sock, const string &filename)

参数：会话 SOCKET，文件名

操作：如果文件存在，将 filename 指定的文件发送给客户端，否则调用 SendErrorMsg() 发送 404 界面

返回值：执行信息

流程图：如图 1-13 所示

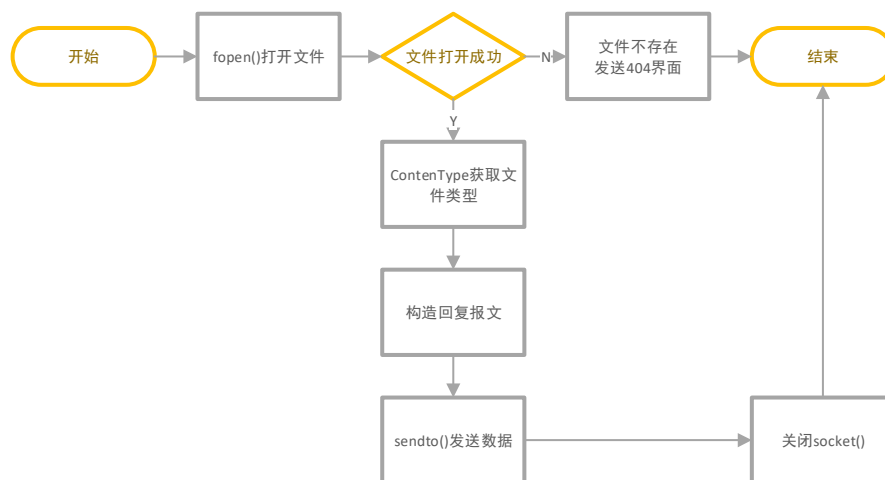


图 1-13 SendData 流程图

◆ string SendErrorMsg(SOCKET sock, string error)

参数：会话 SOCKET，错误信息

操作：回复 404 界面

返回值：执行信息

流程图：如图 1-14 所示



图 1-14 SendErrorMsg 流程图

图形界面实现：

首先使用 Qt Designer 设计出图形框架，图形界面的设计如图 1-15 所示，由按钮、文本浏览器、标签三种部件组成。图形界面主要有三个功能区构成，功能区 1 是一个文本浏览器，用于显示各种处理消息，包括服务器启动消息、报文消息、发送信息等，当消息数过多时用户可以使用 clear 按钮清空文本浏览器中的信息。功能区 2 是用于服务器的控制，start 按钮用于启动浏览器，stop 按钮用于停止浏览器，应用按钮将修改服务器的配置信息，恢复默认按钮用于恢复服务器配置。功能区 3 用于显示当前正在运行的服务器配置。

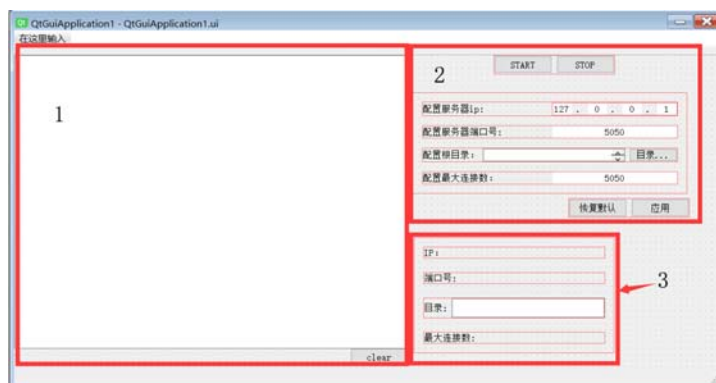


图 1-15 图形界面

1.5 系统测试及结果说明

启动服务器，按下 **START** 按钮，消息框中会输出服务器启动的相关信息，如图 1-16 所示，服务器启动成功，等待客户端的连接请求。



图 1-16 启动服务器

此时服务器客户端的 IP 地址是 127.0.0.1，端口是 5050，打开 Chrome 浏览器，在地址栏中输入 <http://127.0.0.1:5050/index.html> 访问服务器，访问的界面如图 1-17 所示

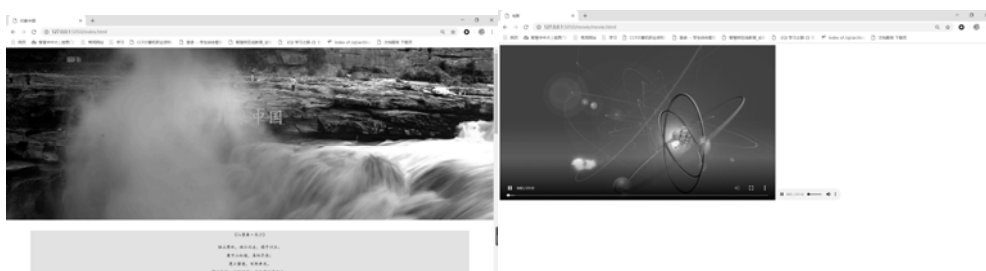


图 1-17 浏览器界面请求

请求的界面中包含图片、视频和音乐，在浏览器中均能正常显示，可知服务器正确分析了报文内容并进行了回复，如图 1-18 所示，服务器每个处理消息的第一行输出请求客户端的 ip 地址和端口号，然后输出请求报文、请求的文件以及处理结果，如果是合法的请求服务器会输

出 send success，如果是非法的请求服务器会输出 send error。

```
send success

connect request : 127.0.0.1 : 49652
GET /css/index.css HTTP/1.1
Host: 127.0.0.1:5050
Connection: keep-alive
User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/71.0.3578.98 Safari/537.36
UWT: 1
Accept: text/css,*/*;q=0.1
Referer: http://127.0.0.1:5050/index.html
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9,zh-CN;q=0.8,zh;q=0.7

request file name:D:\WebServer\css\index.css
Content-type: css

send success

connect request : 127.0.0.1 : 49654
GET /images/hh/hh.jpg HTTP/1.1
Host: 127.0.0.1:5050
Connection: keep-alive
User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/71.0.3578.98 Safari/537.36
```

图 1-18 处理消息

按下 STOP 按钮，服务器会被关闭。在 Chrome 浏览器中输入地址 <http://127.0.0.1:5050/index.html>，由于服务器已经被关闭，因此浏览器是无法获取到界面的，具体的测试结果见图 1-19，符合要求。



图 1-19 无法访问界面

修改配置信息，将 ip 地址设置为 10.14.118.49，端口号设置为 5051，重新启动服务器，然后在同校园网下的另一台主机的浏览器中输入 10.14.118.49:5051/index.html 来访问服务器，访问仍成功，测试结果如图 1-20 所示。



图 1-20 外网访问服务器

更改地址栏中的请求文件名称，将 index.html 修改为 index.h，由于此文件不存在，因此服务器会返回一个 404 的界面，同时服务器端会给出发送失败的信息，测试结果如图 1-21 所示，服务器正确识别出了错误并回复了 404 界面，消息框中输出了 send error 的提示信息。

```

connect request : 10.14.118.49 : 51772
GET /index.h HTTP/1.1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/64.0.3282.140 Safari/537.36 Edge/17.17134
Accept-Language: zh-Hans-CN, zh-Hans;q=0.8, en-US;q=0.5, en;q=0.3
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Upgrade-Insecure-Requests: 1
Accept-Encoding: gzip, deflate
Host: 10.14.118.49:5051
DNT: 1
Connection: Keep-Alive

request file name:D:\WebServer\index.h
File:D:\WebServer\index.h unfind
send error
*****

```

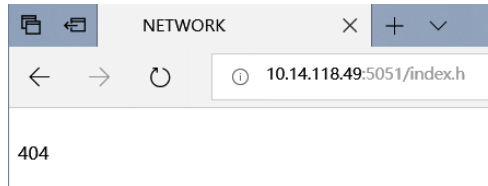


图 1-21 错误请求处理

此外，服务器还可以处理建立连接过程中的异常，将服务器的 ip 地址修改成一个错误的 ip 地址，例如修改为 222.222.222.222，这样服务器就会在 bind()阶段报错，函数返回-1，并结束服务器的运行，测试结果如图 1-22 所示，服务器将会输出绑定失败的提示信息。

```

current config:
SERVERADDRESS = 222.222.222.222
PORT = 5050
HOMEDIR = D:\WebServer
MAXCONNECTION = 5050
server config success
Effective after restarting the server

Winsock startup ok!
Server socket create ok!
Server socket bind error!

```

图 1-22 服务器绑定失败

1.6 其它需要说明的问题

编译方法：

前提条件：已安装 VS2017 和 Qt5.8.0。

- 1.直接解压源码文件夹中的工程文件，使用 Visual Studio 2017 打开工程。
- 2.将第三方组件文件夹下的 qt-vsaddin-msvc2015-2.2.2 拖入到 VisualStudio 中进行加载。
- 3.重新启动工程，选中 VS2017 工具栏中的 qt vs tools 下拉菜单，选择 qt options 选项。
- 4.在打开的 qt options 窗口中选择 qt 5.8.0 版本。
- 5.将 qt5.8.0 根目录下的 lib，include，bin 文件添加到计算机的环境变量中。
- 6.F5 编译运行工程文件。

实验二 数据可靠传输协议设计实验

2.1 环境

操作系统：Microsoft Windows 10 家庭中文版 (64 位)
处理器：Intel(R) Core(TM) i5-8250U CPU @ 1.60GHz(1800 MHz)
内存：8.00 GB (2400 MHz)
显卡：NVIDIA GeForce MX150
开发平台：Microsoft Visual Studio Community 2017 版本 15.8.8
编程语言：C++
第三方组件：netsimlib.lib

2.2 实验要求

本实验包括三个级别的内容，具体包括：

- ✧ 实现基于 GBN 的可靠传输协议，分值为 50%。
- ✧ 实现基于 SR 的可靠传输协议，分值为 30%。
- ✧ 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

GBN 协议又称回退 N 步协议，在实现 GBN 协议时要设立窗口大小、序号范围，以及一个缓冲区。GBN 协议发送方的类定义如下：

```
class GBNSender : public RdtSender
{
private:
    bool waitingState;
```



```

int base;
int nextSeqNum;
Packet *const sndPkt;
const int windowSize;
const int seqRange;
void makePkt(Packet &pck, int acknum, int nextSeqNum, Message &message);
void printWindow(void)
bool ackInWindow(int num);
public:
    bool getWaitingState();
    bool send(Message &message);
    void receive(Packet &ackPkt);
    void timeoutHandler(int seqNum);
public:
    GBNSender();
    virtual ~GBNSender();
};

```

waitingSate 用于表示窗口是否已满，base 是基序号，nextSeqNum 是下一个可用但未发送的序号，sndPkt 是窗口缓冲区，windowSize 是窗口大小，seqRange 是序号范围，makePkt() 用于生成一个分组，printWindow() 用于打印窗口，ackInWindow() 用于判断收到的 ack 的分组是否在窗口范围内。send() 用于发送分组，receive() 是接收 ack 分组后的操作，timeoutHandler() 是超时后的操作。

主要成员函数的具体设计如下：

- ♦ **bool getWaitingState()**
 参数：无
 操作：判断当前窗口是否已满
 返回值：窗口已满返回 true，否则返回 false
 流程图：如图 2-1 所示

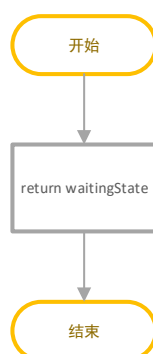


图 2-1 getWaitingState 流程图

- ◆ `bool send(Message &message)`
参数：需要发送的消息 `message`
操作：将 `message` 加入到窗口缓冲区中并发送
返回值：窗口已满返回 `false`，窗口未满返回 `true`
流程图：如图 2-2 所示

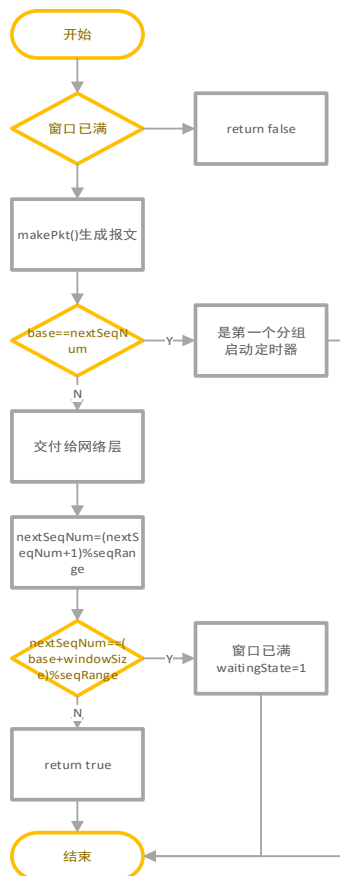


图 2-2 send 流程图

- ◆ `void receive(Packet &ackPkt)`
参数：确认分组 `ackPkt`
操作：处理收到的 `ack` 分组
返回值：无
流程图：如图 2-3 所示

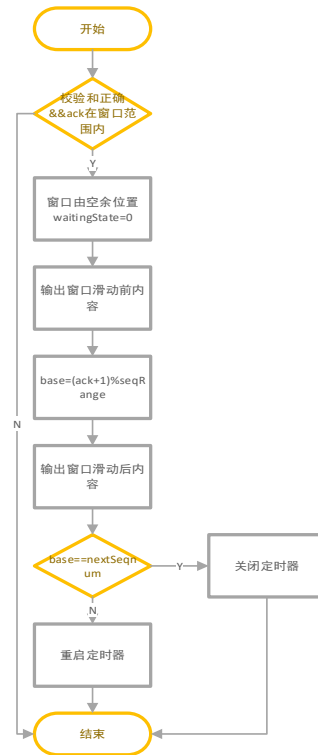


图 2-3 receive 流程图

- ◆ `void timeoutHandler(int seqNum)`
 参数：指针 `arg`，指向一个结构体，包含客户端的 IP、端口和会话 SOCKET
 操作：输出请求分组，分析分组内容并回复相应信息
 返回值：无
 流程图：如图 2-4 所示

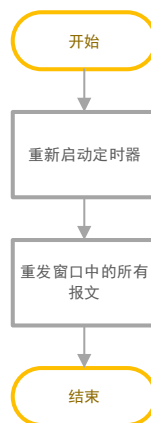


图 2-4 timeoutHandler 流程图

GBN 接收方的操作较为简单，只需要判断收到的分组是否是期待的分组即可，接收方的 `receive()` 函数设计如下：

- ◆ `void receive(Packet &packet)`
 参数：消息分组 `packet`

操作：根据收到分组的序号回复 ack 分组
返回值：无
流程图：如图 2-5 所示

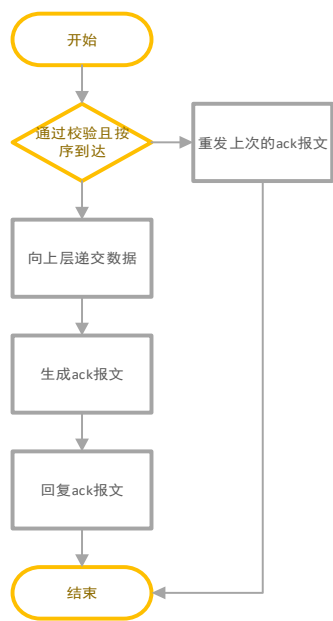


图 2-5 timeoutHandler 流程图

将老师提供的测试程序、in.txt 以及编译好的 GBN.exe 放在同一个目录下，测试程序的执行结果如图 2-6 所示，可知 GBN 协议设计正确。

```
Test "GBN.exe" 1:
正在比较文件 input.txt 和 OUTPUT.TXT
FC: 找不到差异

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

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

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

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

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

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

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

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

Test "GBN.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 2-6 GBN 协议测试结果

打开同目录下生成的 result.txt 文件，观察窗口移动过程，如图 2-7 所示，每当收到一个窗口范围内的 ack 时窗口就会进行移动，移动结果符合预期。

```

收到ack2
窗口移动前:
###窗口内容###
-----
|: seqnum = 2, acknum = -1, checksum = 24414, CCCCCCCCCCCCCCCCCC
|: seqnum = 3, acknum = -1, checksum = 21843, DDDDDDDDDDDDDDDDDDD
|: seqnum = 4, acknum = -1, checksum = 19272, EEEEEEEEEEEEEEEEEEE
|: 空缓冲区
-----

窗口移动后:
###窗口内容###
-----
|: seqnum = 3, acknum = -1, checksum = 21843, DDDDDDDDDDDDDDDDDDD
|: seqnum = 4, acknum = -1, checksum = 19272, EEEEEEEEEEEEEEEEEEE
|: 空缓冲区
-----

```

图 2-7 GBN 协议窗口移动

2.3.2 SR 协议的设计、验证及结果分析

SR 协议又称选择重传协议，与 GBN 协议不同的是，SR 协议为每一个分组都设立了一个定时器，每次只重传超时的那一个分组，因此也需要接收方缓存未按序到达的分组。发送方的类定义如下：

```

class SRSender :public RdtSender
{
private:
    bool waitingState;
    int sendBase;
    int nextSeqNum;
    Packet *const sndPkt;
    int *const flag;
    const int windowSize;
    const int seqRange;
    void makePkt(Packet &pck, int acknum, int nextSeqNum, Message &message);
    void printWindow(void);
    bool ackInWindow(int num);
public:
    bool getWaitingState();
    bool send(Message &message);
    void receive(Packet &ackPkt);
    void timeoutHandler(int seqNum);
public:
    SRSender();
    virtual ~SRSender();
};

```

waitingSate 用于表示窗口是否已满，sendBase 是基序号，nextSeqNum 是下一个可用但未发送的序号，sndPkt 是窗口缓冲区，windowSize 是窗口大小，flag 是窗口的平行数组，用于标记窗口中的分组是否收到了 ack，seqRange 是序号范围，makePkt()用于生成一个分组，printWindow()用于打印窗口，ackInWindow()用于判断收到的 ack 的分组是否在窗口范围内。send()用于发送分组，receive()是接收 ack 分组后的操作，timeoutHandler()是超时后的操作。

主要成员函数的具体设计如下：

- ♦ **bool getWaitingState()**
参数：无
操作：判断当前窗口是否已满
返回值：窗口已满返回 true，否则返回 false
流程图：如图 2-8 所示

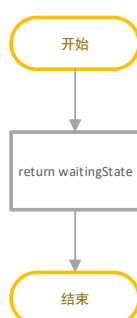


图 2-8 getWaitingState 流程图

- ♦ **bool send(Message &message)**
参数：需要发送的消息 message
操作：将 message 加入到窗口缓冲区中并发送
返回值：窗口已满返回 false，窗口未满返回 true
流程图：如图 2-9 所示

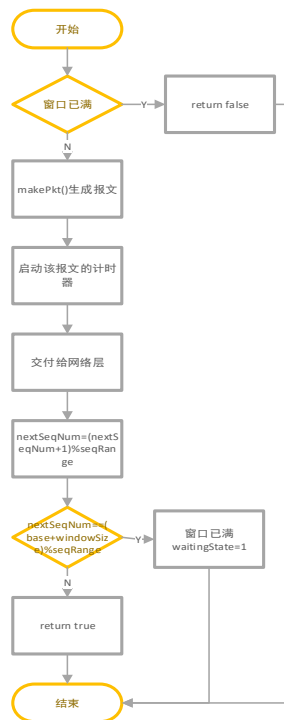


图 2-9 send 流程图

◆ void receive(Packet &ackPkt)

参数：确认分组 ackPkt

操作：处理收到的 ack 分组

返回值：无

流程图：如图 2-10 所示

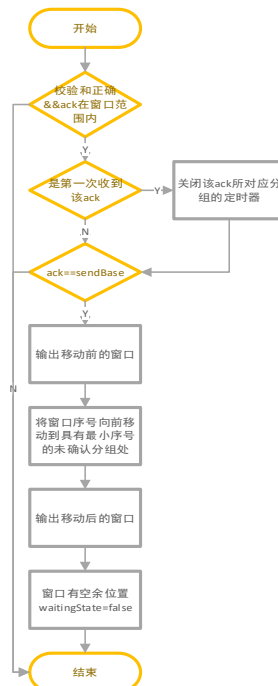


图 2-10 receive 流程图

- ◆ `void timeoutHandler(int seqNum)`

参数：指针 `arg`，指向一个结构体，包含客户端的 IP、端口和会话 SOCKET

操作：输出请求分组，分析分组内容并回复相应信息

返回值：1

流程图：如图 2-11 所示

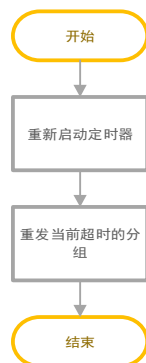


图 2-11 `timeoutHandler` 流程图

SR 接收方的操作相比于 GBN 来说较为复杂，需要设立缓冲区来保存未按序到达的分组，此外与 GBN 协议不同的是，SR 没有累计确认功能。主要函数 `receive()` 的实现如下：

- ◆ `void receive(Packet &packet)`

参数：消息分组 `packet`

操作：根据收到分组的序号回复 `ack` 分组

返回值：无

流程图：如图 2-12 所示

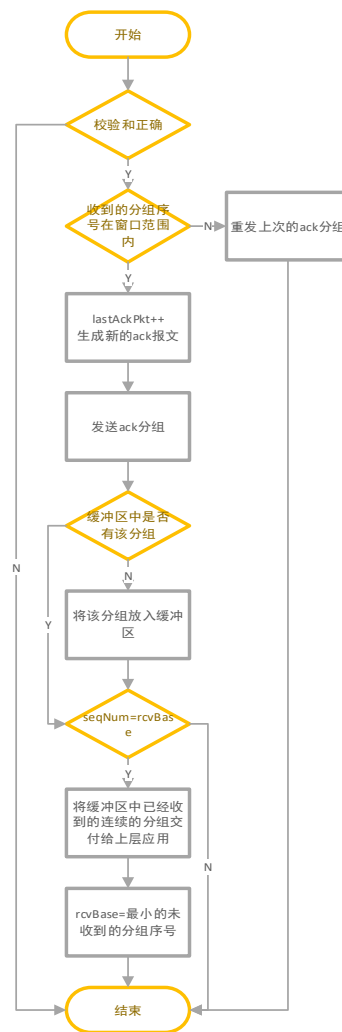


图 2-12 receive 流程图

将老师提供的测试程序、in.txt 以及编译好的 SR.exe 放置在同一个目录下，测试程序的执行结果如图 2-13 所示，可知 SR 协议设计正确。

```

Test "SR.exe" 1:
正在比较文件 input.txt 和 OUTPUT.TXT
FC: 找不到差异

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

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

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

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

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

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

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

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

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

```

图 2-13 SR 协议测试结果

打开同目录下生成的 result.txt 文件，观察窗口移动过程，如图 2-14 所示，每当收到一个窗口范围内的 ack 时窗口就会进行移动，移动结果符合预期。

```

收到ack报文: seqnum = -1, acknum = 1, checksum = 12850, .....
*****模拟网络环境*****: 关闭定时器, 当前时间 = 234.657, 定时器报文序号 = 1
窗口移动前:
###窗口内容###

| Ack已收到: seqnum = 1, acknum = -1, checksum = 51400, RRRRRRRRRRRRRRRRRRRR
| Ack未收到: seqnum = 2, acknum = -1, checksum = 48829, SSSSSSSSSSSSSSSSSSSS
| Ack未收到: seqnum = 3, acknum = -1, checksum = 46258, TTTTTTTTTTTTTTTTTTTT
| Ack未收到: seqnum = 4, acknum = -1, checksum = 43687, UUUUUUUUUUUUUUUUUUUU

窗口移动后:
###窗口内容###

| Ack未收到: seqnum = 2, acknum = -1, checksum = 48829, SSSSSSSSSSSSSSSSSSSS
| Ack未收到: seqnum = 3, acknum = -1, checksum = 46258, TTTTTTTTTTTTTTTTTTTT
| Ack未收到: seqnum = 4, acknum = -1, checksum = 43687, UUUUUUUUUUUUUUUUUUUU
|: 空缓冲区

```

图 2-14 SR 协议窗口移动

2.3.3 简单 TCP/IP 协议的设计、验证及结果分析

本实验中实现了简单的 TCP 协议，在 GBN 协议的基础上进行修改，修改内容为超时时只重传第一个分组和当收到 3 个冗余 ack 时立刻重传序号为 base 的分组，此外 TCP 协议的 ack 序号不再表示已经确认的分组，而是表示接收方希望收到的下一个分组。TCP 协议发送方的类定义如下：

```

class TCPSender :public RdtSender
{
private:
    bool waitingState;
    int base;
    int nextSeqNum;
    int redundantAckNum;
    Packet *const sndPkt;
    const int windowSize;
    const int seqRange;
    void makePkt(Packet &pck, int acknum, int nextSeqNum, Message &message);
    void printWindow(void);
    bool ackInWindow(int num);
public:
    bool getWaitingState();
    bool send(Message &message);
    void receive(Packet &ackPkt);
    void timeoutHandler(int seqNum);
public:
    TCPSender();
    virtual ~TCPSender();
};

```

waitingSate 用于表示窗口是否已满，base 是基序号，nextSeqNum 是下一个可用但未发送的序号，redundantAckNum 是连续收到的冗余 ack 个数，sndPkt 是窗口缓冲区，windowSize 是窗口大小，seqRange 是序号范围，makePkt()用于生成一个分组，printWindow()用于打印窗口，ackInWindow()用于判断收到的 ack 的分组是否在窗口范围内。send()用于发送分组，receive()是接收 ack 分组后的操作，timeoutHandler()是超时后的操作。

主要成员函数的具体设计如下：

- ◆ bool getWaitingState()
 - 参数：无
 - 操作：判断当前窗口是否已满
 - 返回值：窗口已满返回 true，否则返回 false
 - 流程图：如图 2-15 所示

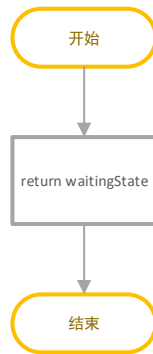


图 2-15 getWaitingState 流程图

- ◆ **bool send(Message &message)**
 参数：需要发送的消息 message
 操作：将 message 加入到窗口缓冲区中并发送
 返回值：窗口已满返回 false，窗口未满返回 true
 流程图：如图 2-16 所示

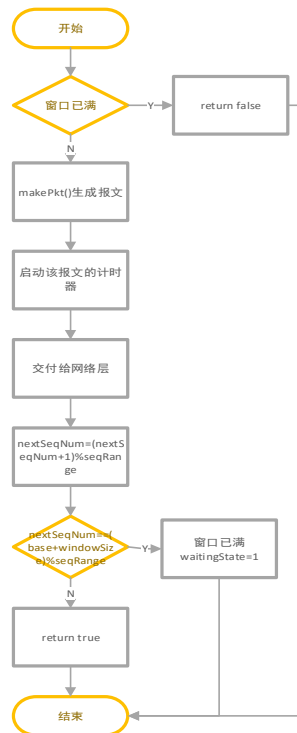


图 2-16 send 流程图

- ◆ **void receive(Packet &ackPkt)**
 参数：确认分组 ackPkt
 操作：处理收到的 ack 分组
 返回值：无
 流程图：如图 2-17 所示

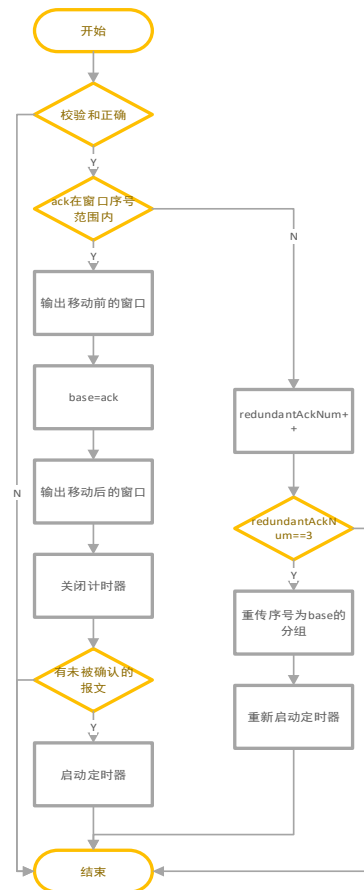


图 2-17 receive 流程图

◆ void timeoutHandler(int seqNum)

参数：指针 arg，指向一个结构体，包含客户端的 IP、端口和会话 SOCKET

操作：输出请求分组，分析分组内容并回复相应信息

返回值：1

流程图：如图 2-18 所示

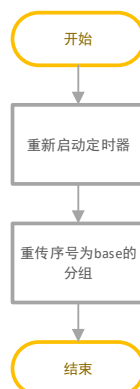


图 2-18 timeoutHandler 流程图

TCP 接收方的操作较为简单，在 GBN 的 receive()函数的基础上进行修改，将 ack 修改为期望收到的下一个分组序号即可，接收方的 receive()函数设计如下：

◆ void receive(Packet &packet)

参数：消息分组 packet

操作：根据收到分组的序号回复 ack 报文

返回值：无

流程图：如图 2-19 所示

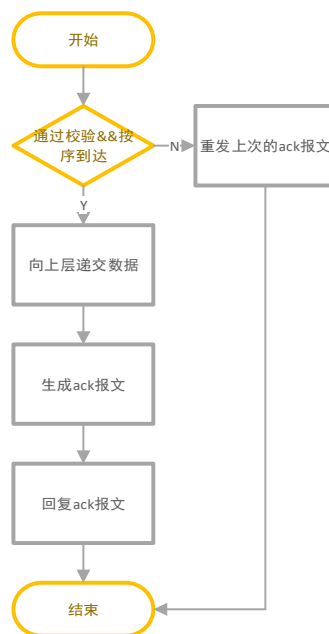


图 2-19 receive 流程图

将老师提供的测试程序、in.txt 以及编译好的 TCP.exe 放置在同一个目录下，测试程序的执行结果如图 2-20 所示，可知 TCP 协议设计正确。

```
Test "TCP.exe" 1:
正在比较文件 input.txt 和 OUTPUT.TXT
FC: 找不到差异

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

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

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

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

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

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

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

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

Test "TCP.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 2-20 TCP 协议测试结果

打开同目录下生成的 `result.txt` 文件，观察窗口移动过程，如图 2-21 所示，每当收到一个窗口范围内的 `ack` 时窗口就会进行移动，移动结果符合预期。如图 2-22 所示，当收到 3 个冗余 `ack` 时立即执行快速重传。

```

# 设置lock4
窗口移动前:
###窗口内容###
.....

| seqnum = 3, acknum = -1, checksum = 21843, DDDDDDDDDDDDDDDDDDDDD
| seqnum = 4, acknum = -1, checksum = 19272, EEEEEEEEEEEEEEEEEEEE
| 空缓冲区
| 空缓冲区
#####

窗口移动后:
###窗口内容###
.....

| seqnum = 4, acknum = -1, checksum = 19272, EEEEEEEEEEEEEEEEEEEE
| 空缓冲区
| 空缓冲区
| 空缓冲区

```

图 2-21 TCP 协议窗口移动

[illegible]

图 2-22 TCP 协议快速重传

2.4 其它需要说明的问题

编译方法:

- 1.解压工程文件
- 2.修改 lib 文件的绝对路径
- 3.F5 编译运行

实验三 基于 CPT 的组网实验

3.1 环境

操作系统：Microsoft Windows 10 家庭中文版 (64 位)

处理器：Intel(R) Core(TM) i5-8250U CPU @ 1.60GHz(1800 MHz)

内存：8.00 GB (2400 MHz)

显卡：NVIDIA GeForce MX150

第三方软件：Cisco Packet Tracer 6.0.0.0045

3.2 实验要求

(一) 基础部分

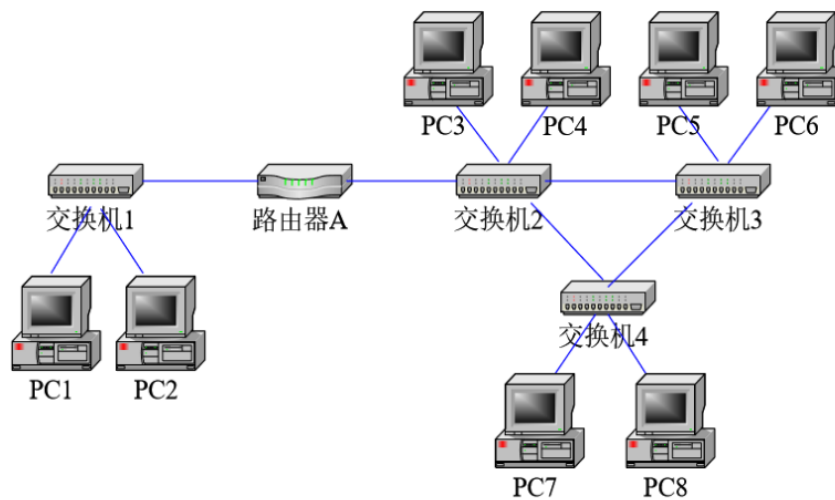


图 3-1 第一项实验拓扑图

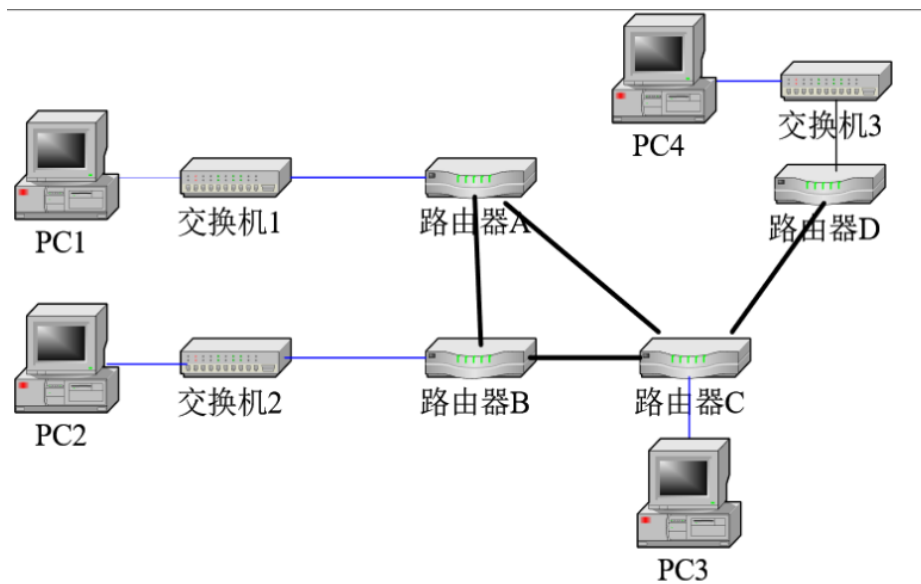


图 3-2 第二项实验拓扑图

第一项实验——IP 地址规划与 Vlan 分配实验

✧ 使用仿真软件描述网络拓扑图 3-1。

✧ 基本内容 1

- 将 PC1、PC2 设置在同一个网段，子网地址是：192.168.0.0/24；
- 将 PC3~PC8 设置在同一个网段，子网地址是：192.168.1.0/24；
- 配置路由器，使得两个子网的各 PC 机之间可以自由通信。

✧ 基本内容 2

- 将 PC1、PC2 设置在同一个网段，子网地址是：192.168.0.0/24；
- 将 PC3、PC5、PC7 设置在同一个网段，子网地址是：192.168.1.0/24；
- 将 PC4、PC6、PC8 设置在同一个网段，子网地址是：192.168.2.0/24；
- 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；
- 测试各 PC 之间的连通性，并结合所学理论知识进行分析；
- 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。

第二项实验——路由配置实验

✧ 使用仿真软件描述网络拓扑图 3-2

✧ 基本内容 1

- 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

-
- 设置路由器端口的 IP 地址
 - 在路由器上配置 RIP 协议，使各 PC 机能互相访问
 - ◇ 基本内容 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
 - 设置路由器端口的 IP 地址
 - 在路由器上配置 OSPF 协议，使各 PC 机能互相访问
 - ◇ 基本内容 3
 - 在基本内容 1 或者 2 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。
 - 在基本内容 1 或者 2 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

（二）综合部分

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

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

IP 地址规划

对于基础内容 1，首先根据网络拓扑图选取设备放置到主界面中，这里选择的计算机型好是 PC-PT，路由器型号是 1804，交换机型号是 2950-24。在连线过程中，计算机和交换机间使用直连线，交换机于交换机之间使用交叉线、交换机与路由器之间使用直连线、路由器之间使用 DCE 线。连接后的到如图 3-3 所示的网络拓扑图。对于图中的设备，为 PC1 分配的的 ip 地址为 192.168.0.1，PC2 的 ip 地址为 192.168.0.2，它们的网关是 192.168.0.100。PC3-PC8 的 ip 地址为 192.168.1.1-192.168.1.6，它们的网关是 192.168.1.100。注意，路由器的端口要勾选启用选项后才能生效。

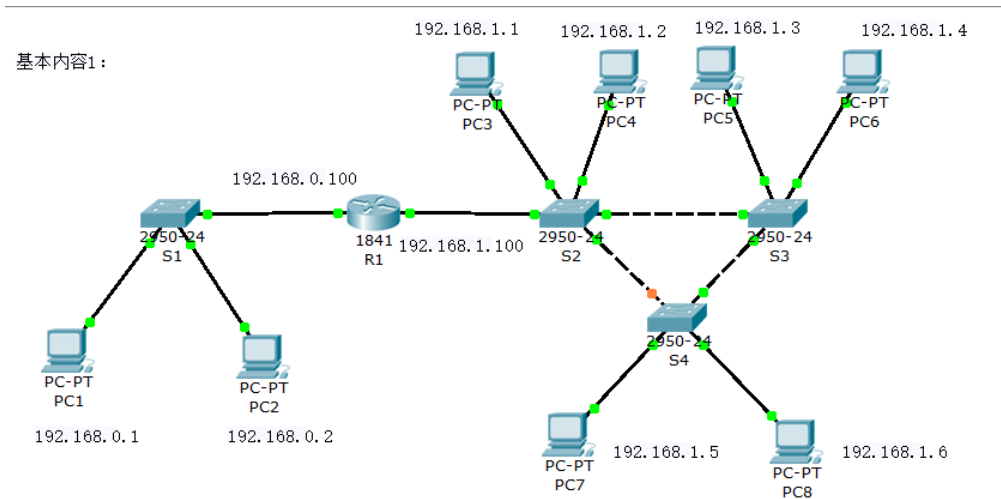


图 3-3 网络拓扑图

ip 地址和网管配置完毕后进行连通性测试，如图 3-4 所示，使用 PC1 ping PC3，PC5，PC5 ping PC1，PC2 均可 ping 通，即两个子网的各设备之间可以 ping 通，因此所连拓扑图正确，完成实验要求。

激活	最后状态	来源设备	目的设备	类型	颜色	时间(秒)	固定周期	顺序	编辑	删除
	成功	PC1	PC3	ICMP		0.000	N	0	(编辑)	(删除)
	成功	PC1	PC5	ICMP		0.000	N	1	(编辑)	(删除)
	成功	PC5	PC1	ICMP		0.000	N	2	(编辑)	(删除)
	成功	PC5	PC2	ICMP		0.000	N	3	(编辑)	(删除)

图 3-4 连通性测试

VLAN 划分

对于基本内容 2，首先为各设备分配 ip 地址，各设备的 ip 地址分配如表 3-1 所示：

表 3-1 ip 地址分配

设备名称	ip 地址		网关
PC1	192.168.0.1/24		192.168.0.100
PC2	192.168.0.2/24		192.168.0.100
PC3	192.168.1.1/24		192.168.1.100
PC4	192.168.2.1/24		192.168.2.100
PC5	192.168.1.2/24		192.168.1.100
PC6	192.168.2.2/24		192.168.2.100
PC7	192.168.1.3/24		192.168.1.100
PC8	192.168.2.3/24		192.168.2.100
R1	Fa0/0	192.168.0.100/24	\
	Fa0/1.1	192.168.1.100/24	\
	Fa0/1.2	192.168.2.100/24	\

仍使用基础内容 1 的网络拓扑图，按照表 3-1 的内容为各主机配置 ip 地址和网关，结果如图 3-5 所示。

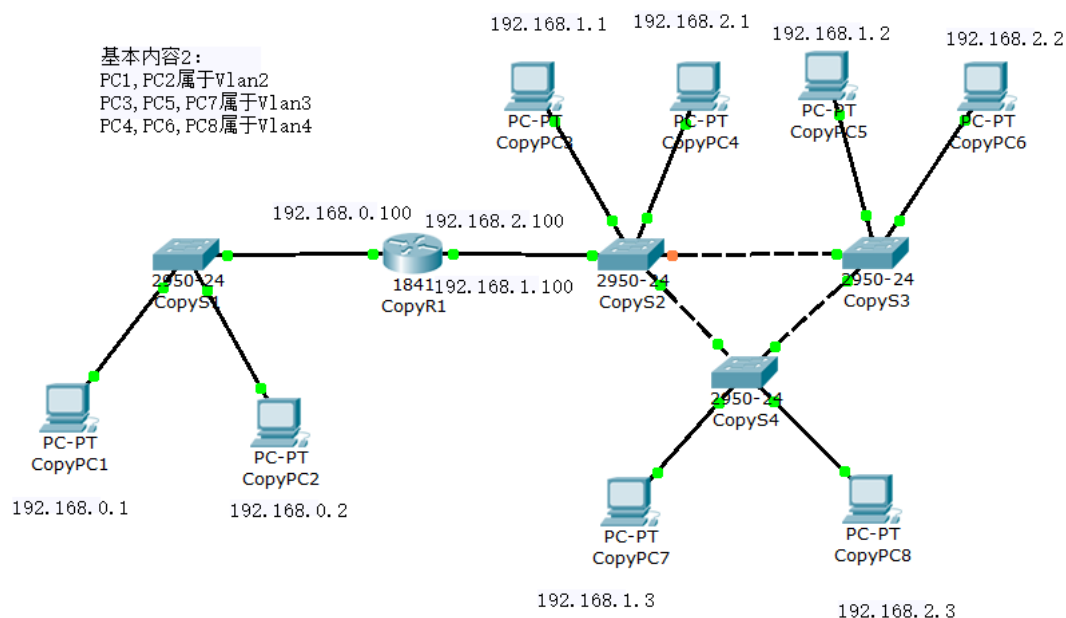


图 3-5

然后进行 vlan 划分，划分方法是双击交换机（以交换机 S2 为例），选择配置→vlan 数据库，填写 vlan 号和 vlan 名称，配置界面如图 3-6 所示。利用同样的方式，为交换机 S2、S3、

S4 增加 vlan 条目 vlan3，vlan4，为交换机 S1 增加 vlan 条目 vlan2。

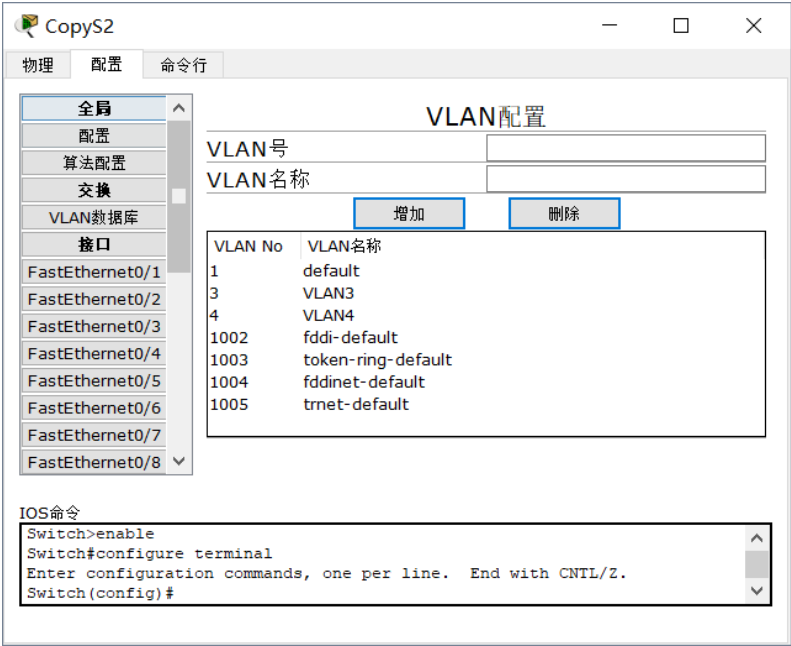


图 3-6 vlan 配置

按照 PC1、PC2 属于 vlan2，PC3、PC5、PC7 属于 vlan3，PC2、PC4、PC6 属于 vlan4 的原则，将交换机和主机相连的接口设置为 access 模式，并选则对应的 vlan 编号，将交换机和交换机以及交换机和路由器相连的接口设置为 trunk 模式。配置完毕后，同一 vlan 下的主机之间可以 ping 通，但是不同 vlan 下的主机不能 ping 通，因为它们不属于同一个子网，要想使不同 vlan 下的主机相互 ping 通，需要对路由器进行配置。路由器左侧端口的配置同基础内容 1，而路由器右端端口由于连接了两个 vlan，需要为其分配两个子端口 Fa0/1.1 和 Fa0/1.2，分别作为 vlan3 和 vlan4 的网关。最后，需要在路由器的 vlan 数据库中增加 3 个 vlan 条目：vlan2、vlan3、vlan4。路由器配置后的结果如图 3-7 所示：

端口	链路	VLAN	IP地址
FastEthernet0/0	Up	--	192.168.0.100/24
FastEthernet0/1	Up	--	<not set>
FastEthernet0/1.1	Up	--	192.168.1.100/24
FastEthernet0/1.2	Up	--	192.168.2.100/24
Vlan1	Down	1	<not set>

主机名称:Router

物理位置: 城市 城市类型 公司市/从室 布线室 机型

图 3-7 路由器配置

配置完路由器后，各 PC 之间均能相互 ping 通，测试结果如图 3-8 所示：

激活	最后状态	来源设备	目的设备	类型	颜色	时间(秒)	固定周期	顺序	编辑	删除
	成功	CopyPC1	CopyPC2	ICMP		0.000	N	0	(编辑)	(删除)
	成功	CopyPC1	CopyPC3	ICMP		0.000	N	1	(编辑)	(删除)
	成功	CopyPC1	CopyPC4	ICMP		0.000	N	2	(编辑)	(删除)
	成功	CopyPC1	CopyPC5	ICMP		0.000	N	3	(编辑)	(删除)
	成功	CopyPC1	CopyPC6	ICMP		0.000	N	4	(编辑)	(删除)
	成功	CopyPC1	CopyPC7	ICMP		0.000	N	5	(编辑)	(删除)
	成功	CopyPC1	CopyPC8	ICMP		0.000	N	6	(编辑)	(删除)
	成功	CopyPC3	CopyPC1	ICMP		0.000	N	7	(编辑)	(删除)
	成功	CopyPC4	CopyPC1	ICMP		0.000	N	8	(编辑)	(删除)
	成功	CopyPC8	CopyPC1	ICMP		0.000	N	9	(编辑)	(删除)

图 3-8 路由器配置完毕后连通性测试

3.3.2 路由配置实验的步骤及结果分析

RIP 协议

首先选择设备并连线，构建网络拓扑图。需要注意的是路由器和路由器之间使用 DCE 线，要接到 Serial 接口上。路由器和 PC 之间使用交叉线，要接到以太网接口上。连接的网络拓扑图如图 3-9 所示，其中各设备的 ip 地址和网关配置如表 3-2 所示。

表 3-2 ip 地址分配表

设备名称	ip 地址		网关
PC1	192.168.1.1/24		192.168.1.100
PC2	192.168.2.1/24		192.168.2.100
PC3	192.168.3.124		192.168.3.100
PC4	192.168.4.124		192.168.4.100
A	Fa0/0	192.168.1.100/24	\
	Se0/0	192.168.6.2/24	\
	Se0/1	192.168.2.100/24	\
B	Fa0/0	192.168.0.100/24	\
	Se0/0	192.168.8.2/24	\
	Se0/1	192.168.7.2/24	\
C	Fa0/0	192.168.0.100/24	\
	Se0/0	192.168.8.1/24	\
	Se0/1	192.168.6.1/24	\
	Se0/2	192.168.5.1/24	
D	Fa0/0	192.168.4.100/24	\
	Se0/0	192.168.5.2/24	\

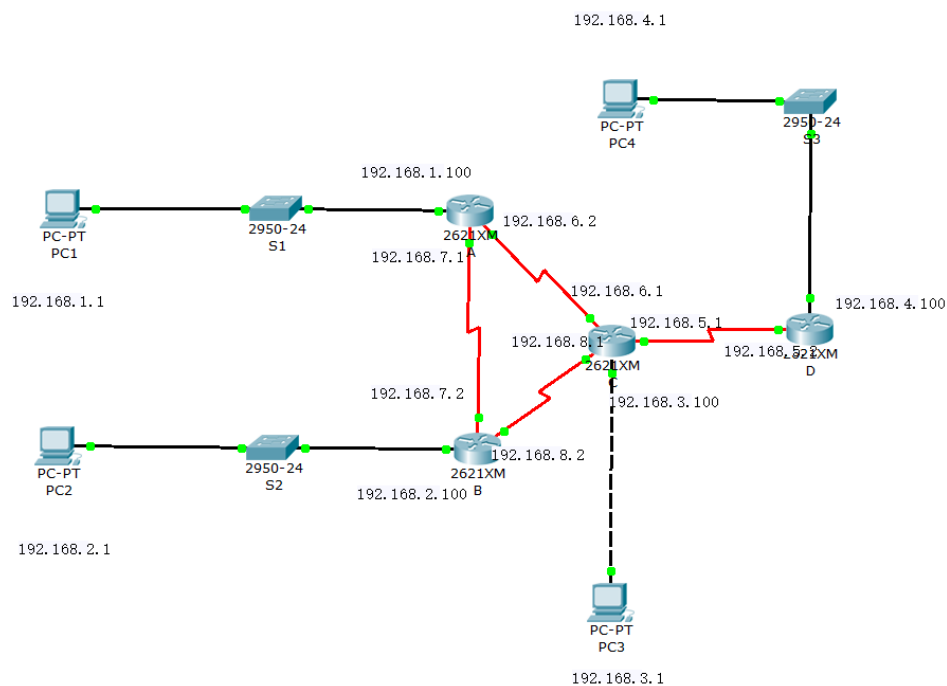


图 3-9 网络拓扑图

接下来为路由器设置时钟频率，将各个路由器之间的时钟频率均设置为 2000000。最后配置 RIP 协议，配置的方法是双击路由器→配置→RIP，将和各个路由器直接相连的子网地址输入即可。配置界面（以路由器 A 为例）如图 3-10 所示。



图 3-10 RIP 协议配置界面

对各主机之间的通信状况进行测试，测试结果如图 3-11 所示，可知各台主机之间均可相互通信。

激活	最后状态	来源设备	目的设备	类型	颜色	时间(秒)	固定周期	顺序	编辑	删除
	成功	PC1	PC2	ICMP		0.000	N	0	(编辑)	(删除)
	成功	PC1	PC3	ICMP		0.000	N	1	(编辑)	(删除)
	成功	PC4	PC2	ICMP		0.000	N	10	(编辑)	(删除)
	成功	PC4	PC3	ICMP		0.000	N	11	(编辑)	(删除)
	成功	PC1	PC4	ICMP		0.000	N	2	(编辑)	(删除)
	成功	PC2	PC1	ICMP		0.000	N	3	(编辑)	(删除)
	成功	PC2	PC3	ICMP		0.000	N	4	(编辑)	(删除)
	成功	PC2	PC4	ICMP		0.000	N	5	(编辑)	(删除)
	成功	PC3	PC1	ICMP		0.000	N	6	(编辑)	(删除)
	成功	PC3	PC2	ICMP		0.000	N	7	(编辑)	(删除)
	成功	PC3	PC4	ICMP		0.000	N	8	(编辑)	(删除)
	成功	PC4	PC1	ICMP		0.000	N	9	(编辑)	(删除)

图 3-11 RIP 通信测试

OSPF 协议

OSPF 协议使用和基础内容 1 相同的网络拓扑图和 ip 地址，唯一的区别在于路由器的配置。OSPF 的协议只能在命令行中进行配置，这里以路由器 A 为例说明一下配置过程，其他路由器的配置过程同理。

首先，双击路由器 A，进入 A 的命令行配置界面，输入如下命令：

```
Router#en
Router#conf t
Router(config)#router ospf 1
Router(config-router)# network 192.168.1.0 0.0.0.255 area 0
Router(config-router)# network 192.168.6.0 0.0.0.255 area 0
Router(config-router)# network 192.168.7.0 0.0.0.255 area 0
Router(config-router)#end
```

即可完成 OSPF 协议的配置。使用同样的方法为路由器 B、C、D 进行配置。为了确认 OSPF 配置成功，使用 show ip router 查看 A 的路由表，如图 3-12 所示，C 表示和路由器 A 直接相连的子网，O 表示路由器 A 通过 OSPF 协议学到的路由条目，可知 OSPF 协议正常运行。

```
Gateway of last resort is not set

C    192.168.1.0/24 is directly connected, FastEthernet0/0
O    192.168.2.0/24 [110/65] via 192.168.7.2, 00:07:01, Serial0/0
O    192.168.3.0/24 [110/65] via 192.168.6.1, 00:07:11, Serial0/1
O    192.168.4.0/24 [110/129] via 192.168.6.1, 00:07:01, Serial0/1
O    192.168.5.0/24 [110/128] via 192.168.6.1, 00:07:11, Serial0/1
C    192.168.6.0/24 is directly connected, Serial0/1
C    192.168.7.0/24 is directly connected, Serial0/0
O    192.168.8.0/24 [110/128] via 192.168.7.2, 00:07:01, Serial0/0
    [110/128] via 192.168.6.1, 00:07:01, Serial0/1

Router#
```

图 3-12 A 的路由表

使用 PDU 测试各主机之间的连通性，测试结果如图 3-13 所示，可知各台主机之间均可相互通信。

激活	最后状态	来源设备	目的设备	类型	颜色	时间(秒)	固定周期	顺序	编辑	删除
	成功	CopyPC1	CopyPC2	ICMP		0.000	N	0	(编辑)	(删除)
	成功	CopyPC1	CopyPC3	ICMP		0.000	N	1	(编辑)	(删除)
	成功	CopyPC1	CopyPC4	ICMP		0.000	N	2	(编辑)	(删除)
	成功	CopyPC2	CopyPC1	ICMP		0.000	N	3	(编辑)	(删除)
	成功	CopyPC2	CopyPC3	ICMP		0.000	N	4	(编辑)	(删除)
	成功	CopyPC2	CopyPC4	ICMP		0.000	N	5	(编辑)	(删除)
	成功	CopyPC3	CopyPC1	ICMP		0.000	N	6	(编辑)	(删除)
	成功	CopyPC3	CopyPC2	ICMP		0.000	N	7	(编辑)	(删除)
	成功	CopyPC3	CopyPC4	ICMP		0.000	N	8	(编辑)	(删除)
	成功	CopyPC4	CopyPC1	ICMP		0.000	N	9	(编辑)	(删除)
	成功	CopyPC4	CopyPC2	ICMP		0.000	N	10	(编辑)	(删除)
	成功	CopyPC4	CopyPC3	ICMP		0.000	N	11	(编辑)	(删除)

图 3-13 OSPF 通信测试

ACL 配置

基础内容 1 要求配置路由器 A 使得主机 1 不能访问外部机器，也不能被外部机器访问，已知拓展 ACL 表可以根据源 IP 和目的 IP 进行访问控制，因此只需增加两个控制规则即可。配置方法为双击路由器 A→进入到命令行界面，输入：

```
Router(config)#access-list 100 deny ip host 192.168.1.1 any
```

```
Router(config)#access-list 100 deny ip any host 192.168.1.1
```

```
Router(config)#access-list 100 permit ip any any
```

第一条规则表示拒绝 PC1 访问其他 PC，第二条规则表示拒绝其他 PC 访问 PC1，第三条规则表示允许其他 PC 之间的通信。然后将该控制表应用到路由器的 Fa0/0 端口的输入规则和输出规则上，即在控制台中输入如下命令：

```
Router(config)#interface Fa0/0
```

```
Router(config-if)#ip access-group 100 in
```

```
Router(config-if)#ip access-group 100 out
```

接下来对主机 A 的连通性进行测试，首先使用 PC1 ping PC2、PC3、PC4，均显示 unreachable，结果如图 3-14 所示。然后分别使用 PC2，PC3，PC4 ping PC1，也是 unreachable，结果如图 3-15 所示。

```
Router Tracer PC Command Line 1.0
PC>ping 192.168.2.1

Pinging 192.168.2.1 with 32 bytes of data:

Reply from 192.168.2.100: Destination host unreachable.
Reply from 192.168.2.100: Destination host unreachable.
Reply from 192.168.2.100: Destination host unreachable.
Reply from 192.168.2.100: Destination host unreachable.

Ping statistics for 192.168.2.1:
    Packets: Sent = 4, Received = 0, Lost = 4 (100% loss),

PC>ping 192.168.3.1

Pinging 192.168.3.1 with 32 bytes of data:

Reply from 192.168.3.100: Destination host unreachable.
Reply from 192.168.3.100: Destination host unreachable.
Reply from 192.168.3.100: Destination host unreachable.
Reply from 192.168.3.100: Destination host unreachable.

Ping statistics for 192.168.3.1:
    Packets: Sent = 4, Received = 0, Lost = 4 (100% loss),

PC>ping 192.168.4.1

Pinging 192.168.4.1 with 32 bytes of data:

Reply from 192.168.4.100: Destination host unreachable.
Reply from 192.168.4.100: Destination host unreachable.
Reply from 192.168.4.100: Destination host unreachable.
Reply from 192.168.4.100: Destination host unreachable.

Ping statistics for 192.168.4.1:
    Packets: Sent = 4, Received = 0, Lost = 4 (100% loss),
```

图 3-14 ACL 测试 1

<pre>PC>ping 192.168.1.1 Pinging 192.168.1.1 with 32 bytes of data: Reply from 192.168.7.1: Destination host unreachable. Reply from 192.168.7.1: Destination host unreachable. Reply from 192.168.7.1: Destination host unreachable. Reply from 192.168.7.1: Destination host unreachable. Ping statistics for 192.168.1.1: Packets: Sent = 4, Received = 0, Lost = 4 (100% loss),</pre>	<pre>PC>ping 192.168.1.1 Pinging 192.168.1.1 with 32 bytes of data: Reply from 192.168.6.2: Destination host unreachable. Reply from 192.168.6.2: Destination host unreachable. Reply from 192.168.6.2: Destination host unreachable. Reply from 192.168.6.2: Destination host unreachable. Ping statistics for 192.168.1.1: Packets: Sent = 4, Received = 0, Lost = 4 (100% loss),</pre>	<pre>PC>ping 192.168.1.1 Pinging 192.168.1.1 with 32 bytes of data: Reply from 192.168.6.2: Destination host unreachable. Reply from 192.168.6.2: Destination host unreachable. Reply from 192.168.6.2: Destination host unreachable. Reply from 192.168.6.2: Destination host unreachable. Ping statistics for 192.168.1.1: Packets: Sent = 4, Received = 0, Lost = 4 (100% loss),</pre>
--	--	--

图 3-15 ACL 测试 2

基础内容 2 要求 PC1 和 PC2 不能相互访问，但可以访问其他主机，根据要求，在路由器 A 的命令行中输入如下命令：

```
Router(config)#access-list 100 deny ip host 192.168.1.1 host 192.168.2.1
```

```
Router(config)#access-list 100 permit ip any any
```

第一条规则表示 PC1 不能访问 PC2，第二条规则表示其余的主机之间可以相互访问。同样的，将该控制表应用到路由器 A 的 Fa0/0 端口上，命令同上。最后对连通性进行测试，使用 PC1 ping PC2，结果为 unreachable，如图 3-16 所示。这是因为 PC1 发送给 PC2 的报文要经过 PC1 的网关（即路由器 A），符合第一条规则，被路由器 A 拒绝转发，然后路由器 A 向 PC1 回复一个 PC2 不可达的消息。而 PC2 ping PC1 的结果为 time out，这是因为 PC2 发送给 PC1 的报文被 PC1 接收了，但是 PC1 发送给 PC2 的回复报文却无法到达，因此 PC2 报 time out，结果如图 3-17 所示。PC1 和 PC4 之间可以相互 ping 通，结果如图 3-18 和图 3-19 所示。

```
PC>ping 192.168.2.1
Pinging 192.168.2.1 with 32 bytes of data:
Reply from 192.168.1.100: Destination host unreachable.
Reply from 192.168.1.100: Destination host unreachable.
Reply from 192.168.1.100: Destination host unreachable.
Reply from 192.168.1.100: Destination host unreachable.
Ping statistics for 192.168.2.1:
    Packets: Sent = 4, Received = 0, Lost = 4 (100% loss),
```

图 3-16 PC1 ping PC2

```
PC>ping 192.168.1.1
Pinging 192.168.1.1 with 32 bytes of data:
Request timed out.
Request timed out.
Request timed out.
Request timed out.
Ping statistics for 192.168.1.1:
    Packets: Sent = 4, Received = 0, Lost = 4 (100% loss),
```

图 3-17 PC2 ping PC1

```
PC>ping 192.168.4.1
Pinging 192.168.4.1 with 32 bytes of data:
Reply from 192.168.4.1: bytes=32 time=11ms TTL=125
Reply from 192.168.4.1: bytes=32 time=3ms TTL=125
Reply from 192.168.4.1: bytes=32 time=7ms TTL=125
Reply from 192.168.4.1: bytes=32 time=2ms TTL=125
Ping statistics for 192.168.4.1:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 2ms, Maximum = 11ms, Average = 5ms
```

图 3-18 PC1 ping PC4

```
PC>ping 192.168.1.1

Pinging 192.168.1.1 with 32 bytes of data:

Reply from 192.168.1.1: bytes=32 time=3ms TTL=125
Reply from 192.168.1.1: bytes=32 time=8ms TTL=125
Reply from 192.168.1.1: bytes=32 time=9ms TTL=125
Reply from 192.168.1.1: bytes=32 time=2ms TTL=125

Ping statistics for 192.168.1.1:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 2ms, Maximum = 9ms, Average = 5ms
```

图 3-19 PC4 ping PC1

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

首先根据各区域的主机数量对子网进行划分，宿舍区拥有 3 个子网，图书馆拥有 1 个子网，学院拥有 4 个子网，划分结果如表 3-3 所示。

表 3-3 子网划分

区域	子网	可用ip数量	分配ip数量	分配地址	子网掩码	网关
宿舍1	211.69.4.0/24	254	200	211.69.4.1-211.69.4.200	255.255.255.0	211.69.4.254/24
宿舍2	211.69.5.0/24	254	200	211.69.5.1-211.69.4.200	255.255.255.0	211.69.5.254/24
宿舍3	211.69.6.0/24	254	200	211.69.6.1-211.69.4.200	255.255.255.0	211.69.6.254/24
图书馆	211.69.7.0/25	126	100	211.69.7.1-211.69.7.100	255.255.255.128	211.69.7.126/25
学院1	211.69.7.128/27	30	20	211.69.7.129-211.69.7.148	255.255.255.224	211.69.7.158/27
学院2	211.69.7.160/27	30	20	211.69.7.161-211.69.7.180	255.255.255.224	211.69.7.190/27
学院3	211.68.7.192/27	30	20	211.69.7.193-211.69.7.212	255.255.255.224	211.69.7.222/27
学院4	211.68.7.224/27	30	20	211.69.7.225-211.69.7.244	255.255.255.224	211.69.7.254/27

整个校园网只使用一台核心路由器，考虑到宿舍区的主机台数较多，流量大，因此将 3 个宿舍区的主机经各自交换机汇总后分别接到路由器的 3 个接口上；而学院主机数少，使用一台交换机汇总后接到路由器上，并使用 VLAN 对学院进行虚拟局域网划分；图书馆经一台交换机汇总后接到路由器上。此外，考虑到交换机的接口有限，因此将每 20 台主机分为一组连接到一台交换机上，再把这些交换机连接到另一台交换机上进行汇总，构建出总体网络拓扑图如图 3-20 所示。

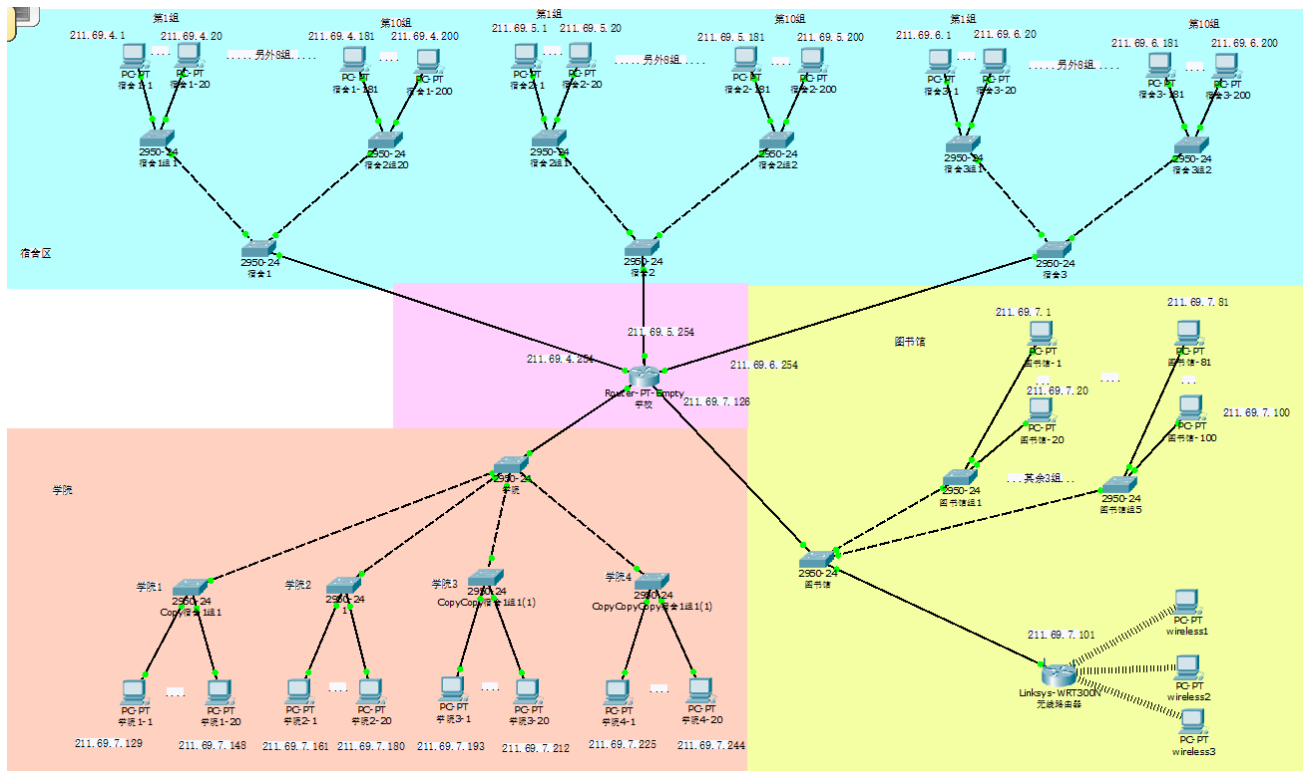


图 3-20 总体网络拓扑图

其中，上方的蓝色区域表示宿舍区，左下角的橙色区域表示学院，右下角的黄色区域表示图书馆，中间的粉色区域表示核心路由器。

3.4.2 实验步骤

首先按照总计网络拓扑图进行连线，然后根据表 3-3 对各主机的 ip 地址和网关进行配置。对于每一组的主机只画出了两台主机，分别为该组的第一台主机和最后一台主机，为其分配该组的第一个 ip 地址和最后一个 ip 地址。然后对学院进行 VLAN 划分，将学院 1 划分为 VLAN11，学院 2 划分为 VLAN12，学院 3 划分为 VLAN13，学院 4 划分为 VLAN14，VLAN 的划分方式同 3.3.1 小节，接下来对路由器进行配置，路由器的各端口配置及连接的子网如表 3-4 所示

表 3-4 路由器端口配置表

路由器端口		连接区域		ip 地址
Gig1/0		宿舍区 1		211. 69. 4. 254/24
Gig2/0		宿舍区 2		211. 69. 5. 254/24
Gig3/0		宿舍区 3		211. 69. 6. 254/24
Gig4/0		图书馆		211. 69. 7. 126/25
Gig5/0	Gig5/0.1	学院 1	VLAN11	211. 69. 7. 158/27
	Gig5/0.2	学院 2	VLAN12	211. 69. 7. 190/27
	Gig5/0.3	学院 3	VLAN13	211. 69. 7. 222/27
	Gig5/0.4	学院 4	VLAN14	211. 69. 7. 254/27

其中，路由器配置 VLAN 的命令如下：

```
int Gig5/0.1
encapsulation dot1q 11
ip address 211.69.7.158 255.255.255.224
exit
int Gig5/0.2
encapsulation dot1q 12
ip address 211.69.7.190 255.255.255.224
exit
int Gig5/0.3
encapsulation dot1q 13
ip address 211.69.7.222 255.255.255.224
exit
int Gig5/0.4
encapsulation dot1q 14
ip address 211.69.7.254 255.255.255.224
exit
```

最终配置完毕后路由器的配置信息如图 3-21 所示。

端口	链路	IP地址	IPv6地址
Serial0/0	Down	<not set>	<not set>
GigabitEthernet1/0	Up	211.69.4.254/24	<not set>
GigabitEthernet2/0	Up	211.69.5.254/24	<not set>
GigabitEthernet3/0	Up	211.69.6.254/24	<not set>
GigabitEthernet4/0	Up	211.69.7.126/25	<not set>
GigabitEthernet5/0	Up	<not set>	<not set>
GigabitEthernet5/0.1	Up	211.69.7.158/27	<not set>
GigabitEthernet5/0.2	Up	211.69.7.190/27	<not set>
GigabitEthernet5/0.3	Up	211.69.7.222/27	<not set>
GigabitEthernet5/0.4	Up	211.69.7.254/27	<not set>
GigabitEthernet6/0	Down	<not set>	<not set>
GigabitEthernet7/0	Down	<not set>	<not set>
GigabitEthernet8/0	Down	<not set>	<not set>
GigabitEthernet9/0	Down	<not set>	<not set>

主机名称:Router

图 3-21 路由器配置信息

为了实现学院和宿舍区的访问限制，需要配置 ACL 表，ACL 表的配置命令如下：

```
ip access-list extended 100
```

```
access-list 100 deny ip 211.69.4.0 0.0.0.255 211.69.7.128 0.0.0.127
```

```
access-list 100 deny ip 211.69.7.128 0.0.0.127 211.69.4.0 0.0.0.255
```

```
access-list 100 permit ip any any
```

```
interface Gig1/0
```

```
ip access-group 100 in
```

```
ip access-group 100 out
```

```
exit
```

```
ip access-list extended 101
```

```
access-list 101 deny ip 211.69.5.0 0.0.0.255 211.69.7.128 0.0.0.127
```

```
access-list 101 deny ip 211.69.7.128 0.0.0.127 211.69.5.0 0.0.0.255
```

```
access-list 101 permit ip any any
```

```
interface Gig2/0
```

```
ip access-group 101 in
```

```
ip access-group 101 out
```

```
exit
```

```
ip access-list extended 102
```

```
access-list 102 deny ip 211.69.6.0 0.0.0.255 211.69.7.128 0.0.0.127
```

```
access-list 102 deny ip 211.69.7.128 0.0.0.127 211.69.6.0 0.0.0.255
```

```
access-list 102 permit ip any any
```

```
interface Gig3/0
```










```
ip access-group 102 in
```

ip access-group 102 out

配置完毕后对各区域之间的连通性进行测试，连通性测试结果如表 3-5 所示。

表 3-5 连通性测试结果

序号	测试区域	测试结果									
1	宿舍 → 宿舍	激活	最后状态	来源设备	目的设备	类型	颜色	时间(秒)	固定周期	顺序	编辑 删除
			成功	宿舍1-1	宿舍1-181	ICMP		0.000	N	0	(编辑) (删除)
			成功	宿舍1-1	宿舍2-20	ICMP		0.000	N	1	(编辑) (删除)
			成功	宿舍1-1	宿舍3-20	ICMP		0.000	N	2	(编辑) (删除)
			成功	宿舍1-181	宿舍1-1	ICMP		0.000	N	3	(编辑) (删除)
			成功	宿舍2-20	宿舍1-1	ICMP		0.000	N	4	(编辑) (删除)
2	宿舍 → 图书馆	激活	最后状态	来源设备	目的设备	类型	颜色	时间(秒)	固定周期	顺序	编辑 删除
			成功	宿舍1-1	图书馆-1	ICMP		0.000	N	0	(编辑) (删除)
			成功	宿舍1-1	图书馆-100	ICMP		0.000	N	1	(编辑) (删除)
			成功	宿舍2-1	图书馆-1	ICMP		0.000	N	2	(编辑) (删除)
			成功	宿舍2-1	图书馆-100	ICMP		0.000	N	3	(编辑) (删除)
			成功	宿舍3-1	图书馆-1	ICMP		0.000	N	4	(编辑) (删除)
3	宿舍 → 学院	激活	最后状态	来源设备	目的设备	类型	颜色	时间(秒)	固定周期	顺序	编辑 删除
			失败	宿舍1-1	学院1-1	ICMP		0.000	N	0	(编辑) (删除)
			失败	宿舍1-1	学院2-1	ICMP		0.000	N	1	(编辑) (删除)
			失败	宿舍1-1	学院3-1	ICMP		0.000	N	2	(编辑) (删除)
			失败	宿舍1-1	学院4-1	ICMP		0.000	N	3	(编辑) (删除)
			失败	宿舍2-1	学院1-1	ICMP		0.000	N	4	(编辑) (删除)
4	学院 → 学院	激活	最后状态	来源设备	目的设备	类型	颜色	时间(秒)	固定周期	顺序	编辑 删除
			成功	学院1-1	学院1-20	ICMP		0.000	N	0	(编辑) (删除)
			失败	学院1-1	学院2-1	ICMP		0.000	N	1	(编辑) (删除)
			成功	学院1-1	学院3-1	ICMP		0.000	N	2	(编辑) (删除)
			失败	学院1-1	学院4-1	ICMP		0.000	N	3	(编辑) (删除)
			成功	学院2-1	学院1-20	ICMP		0.000	N	4	(编辑) (删除)
5	学院 → 宿舍	激活	最后状态	来源设备	目的设备	类型	颜色	时间(秒)	固定周期	顺序	编辑 删除
			失败	学院1-1	宿舍1-1	ICMP		0.000	N	0	(编辑) (删除)
			失败	学院2-1	宿舍2-1	ICMP		0.000	N	1	(编辑) (删除)
			失败	学院3-1	宿舍3-1	ICMP		0.000	N	2	(编辑) (删除)
			失败	学院4-1	宿舍3-181	ICMP		0.000	N	3	(编辑) (删除)
6	学院 → 图书馆	激活	最后状态	来源设备	目的设备	类型	颜色	时间(秒)	固定周期	顺序	编辑 删除
			成功	学院1-1	图书馆-1	ICMP		0.000	N	0	(编辑) (删除)
			成功	学院1-1	图书馆-100	ICMP		0.000	N	1	(编辑) (删除)
			成功	学院2-1	图书馆-1	ICMP		0.000	N	2	(编辑) (删除)
			成功	学院2-1	图书馆-100	ICMP		0.000	N	3	(编辑) (删除)
			成功	学院3-1	图书馆-1	ICMP		0.000	N	4	(编辑) (删除)

7	图书馆 → 图书馆		成功	图书馆-1	图书馆-20	ICMP		0.000	N	0	(编辑) (删除)
			成功	图书馆-1	图书馆-81	ICMP		0.000	N	1	(编辑) (删除)
			成功	图书馆-1	图书馆-100	ICMP		0.000	N	2	(编辑) (删除)
8	图书馆无线上网	激活	最后状态	来源设备	目的设备	类型	颜色	时间(秒)	固定周期	顺序	编辑 删除
			成功	wireless1	宿舍2-181	ICMP		0.000	N	0	(编辑) (删除)
			成功	wireless1	宿舍3-20	ICMP		0.000	N	1	(编辑) (删除)

3.4.3 结果分析

由表 3-5 的测试结果可知，图书馆能够进行无线上网，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。在进行 ACL 配置时，使用了拓展 ACL 控制表，可以根据源 IP 和目的 IP 进行访问控制，将其配置在和学院连接的端口即可生效。

3.5 其它需要说明的问题

图书馆的无线终端可以访问学校的其他任意一台主机，但是其他的主机不能访问图书馆的无线终端，这是因为无线路由器的 NAT 将无线终端隔离了。在两台主机第一次进行通信时第一个包会超时，这是因为网关路由器不知道目的 IP 的 MAC 地址导致的。

心得体会与建议

4.1 心得体会

这三次实验分别涉及了应用层、传输层以及网络、链路层三方面的内容，基本涵盖了课上所学知识，通过这次实验，不仅使我巩固了课上知识，更使我掌握了许多课外知识，使我对计算机网络顶层到底层有了更深入的理解。**Socket** 编程实验让我对网络应用的编写和设计有了深刻理解，使我掌握了图形化界面的设计方法。传输协议设计实验使我更深刻的理解了传输层协议的设计与实现，也提升了我的编程能力。**CPT** 组网实验使我加深了对于网络构建和配置的理解。通过 **CPT** 的 **PDU** 模拟传输过程，我可以清晰的看到每个报文的传输路径，使我对网络层、链路层、物理层之间的相互关系以及 **IP** 和 **MAC** 地址之间的关系有了更深一步的了解。

4.2 建议

希望在课堂上提前介绍一下 **socket** 编程的内容，在做实验时可以更快的上手，能有更多的时间完善 **Web** 服务器。