
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	3
1. 5 系统测试及结果说明	7
1. 6 其它需要说明的问题	13
1. 7 参考文献	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	25
2. 5 参考文献	25
实验三 基于 CPT 的组网实验	25
3. 1 环境	25
3. 2 实验要求	25
3. 3 基本部分实验步骤说明及结果分析	26
3. 4 综合部分实验设计、实验步骤及结果分析	35
3. 5 其它需要说明的问题	47
3. 6 参考文献	48
心得体会与建议	49
4. 1 心得体会	49
4. 2 建议	49

实验一 Socket 编程实验

1.1 环境

1.1.1 开发平台

- (1) windows 版本: Windows 10 家庭版
- (2) 处理器: Intel(R) Core(TM) i5-7300U CPU @ 2.60HZ 2.71HZ
- (3) 已安装内存: 8.00GB
- (4) 系统类型: 64 位操作系统, 基于 x64 的处理器
- (5) IDE: Qt Creator 4.5.1 (Based on Qt 5.10.1)
- (6) 编辑器: Desktop Qt 5.10.1 MinGW 32 Bit
- (7) 编程语言: C++

1.1.2 运行平台

- (1) 软件运行的机器与开发机器为同一台机器, 其硬件配置、软件组件与开发平台完全一致。
- (2) 在 Windows 下运行 Qt 编译器的执行文件的方法: 编译 Qt 的项目文件, 产生一个 release 版本的可执行文件, 我的文件叫做 Lab1-SocketProgramming.exe。
- (3) 由于我们采用的网络包是 WinSock 开发包, 所以其不支持在其它操作系统上运行(例如 MacOS 和 Linux)

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

(1) 系统模块划分

主要分为四个模块：UI 模块、建立线程模块（连接建立模块）、响应通信模块。四个模块对应的文件分别为：

1. UI 模块：mainwindow.cpp、mainwindow.h、mainwindow.ui
2. 建立线程模块：newtread.cpp、newtread.h
3. 响应通信模块：communicate.cpp、communicate.h

(2) 每个模块的具体功能

1. UI 模块（主模块）

- A. 按照 1.2 中的要求，设计用户界面。可在 mainwindow.ui 文件中进行图形化配置。用户界面主要包括以下几个部分：1) 监听地址的输入框；2) 监听端口的输入框；3) 服务器的目录路径的输入框；4) 启动服务按钮；5) 关闭服务按钮；6) 请求来源信息显示（包括但不仅限于 IP 地址、端口号、请求报文的长度、请求的结果（如 200 OK））^[1]
- B. 在 mainwindow.cpp 和 mainwindow.h 中，实现有关于上述图形化界面中的按钮的触发函数以及其它设计配置，主要包括：1) 设计来源信息显示表（IP 地址、端口号、请求报文的长度、请求的结果等等信息）；2) 编写按下“启动服务”按钮之后的响应函数 BeginServer；3) 编写按下“结束服务”后的响应函数

CloseServer; 4) 编写本地目录搜寻函数“FindDirecory”。

2. 线程建立模块（连接建立模块）

- A. 这个模块主要实现的内容是：在 UI 模块 listen(监听)成功之后，创建多线程，实现多线程通信(即 socket 面向连接通信方式中的 accept 和 connect 的过程)。
- B. 具体的，我们要实现的内容有：1) 等待新的客户的连接；2) 对于有效连接建立新的线程并且实现其与服务端的连接^[1]。

3. 响应通信模块

- A. 这个模块主要实现的内容是：在客户端建立了新的线程与服务器连接之后，服务器构造响应报文，返回文件以及在图形化界面上输出请求的必要信息的功能。
- B. 具体的，我们要实现的内容有：1) 处理请求报文提取 URL；2) 找到对应的文件，并返回，如果出错发出异常处理信息；3) 根据以上信息构造响应报文并发送。

(3) 模块之间的关系

三个模块之间的关系可以用图 1.1 表达^[1]：

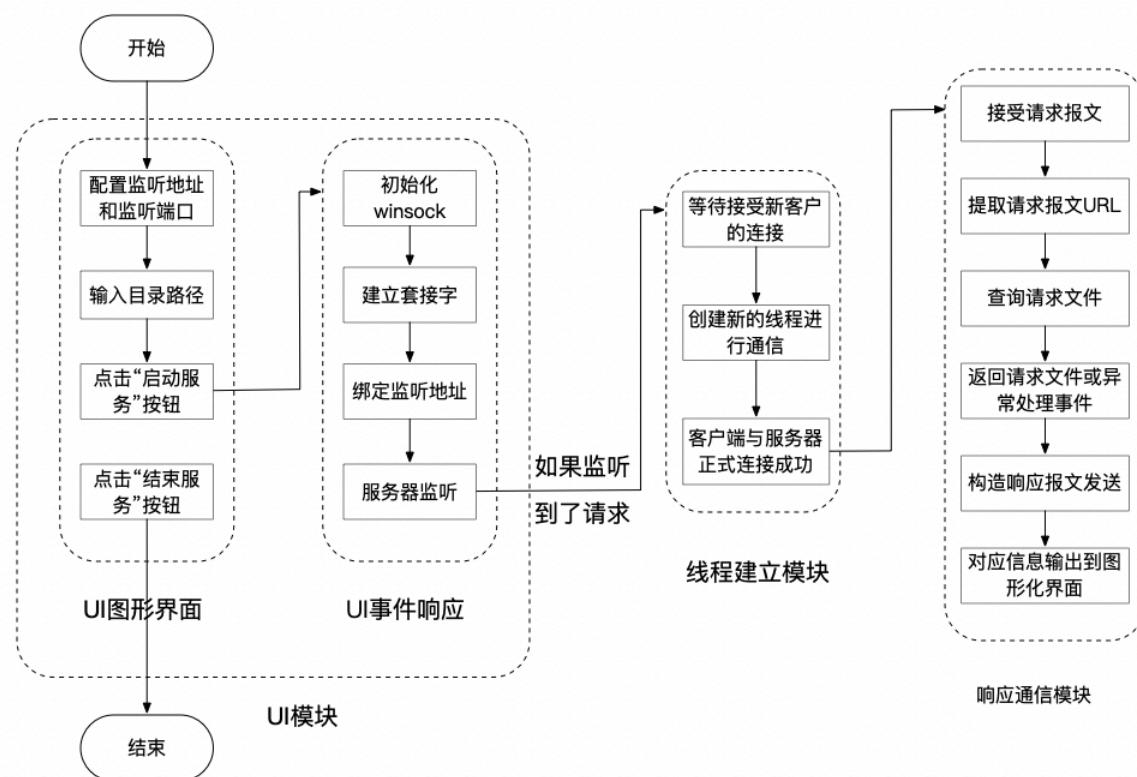


图 1.1 三个模块之间的关系

1.4 系统实现

(1) UI 模块的实现 (MainWindow 的实现)

1. 按照上述设计要求，在mainwindow.ui 中设计图形界面，如下图 1.2 所示：



图 1.2 UI 实现示意图

2. 有关 UI 的响应事件的实现

- A. 设置 UI 中响应信息栏: 我们用一个结构 “QString list” 存储表头信息, list 中存储 “IP、Port、File name、Result” 分别代表 IP 地址、端口号、请求报文的长度、请求的结果。然后我们调用 ui 函数将其配置到图形化界面^[2]。
- B. “启动服务”的响应函数: 用户点击了“启动服务”按钮之后, 主要进行以下几步: 1) 初始化 winsock; 2) 调用 socket 函数建立套接字; 3) 调用 bind 函数绑定监听地址; 4) 调用 listen 函数进行服务器监听; 5) 调用 newthread, 进入线程建立模块^{[1][2]}。
- C. “关闭服务”响应函数: 用户点击了“关闭服务”按钮之后, 进行以下几步: 1) 调用 delete 函数关闭当前线程; 2) 在图形化界面上输出有关服务关闭的信息^[2]。
- D. “寻找目录”的响应函数: 这是一个系统调用, 直接调用特定的函数, 即可以在图形化界面中, 进行目录选择, 函数名为: getExistingDirectory。

(2) 线程建立模块的实现 (newthread 的实现)

1. 等待新客户连接

- A. 只要没有超过最大连接数量, 调用 accept 函数, 接受客户端的套接字。
- B. 接收到套接字后, 看是否是无效的套接字, 如果套接字无效, 返回错误信息, 重新接收。
- C. 如果接收到正确的套接字, 则开启线程, 线程用于客户端和服务器之间的通信。

2. 新建线程之后, 建立连接

- A. 新建立一个进程后，首先调用响应通信模块 Communicate，准备进行响应通信工作^[1]。
- B. 调用 connect 函数，实现客户端与服务器之间的通信^[1]。

(3) 响应通信模块 (communicate 的实现)

1. 解析客户端发过来的请求报文，提取 URL (客户请求的文件)

- A. 在获得了用户的请求报文之后，我们首先要做的，就是提取用户所请求的文件。这里我们提取用户请求文件在服务器跟目录下的相对路径 URL 以及文件的完成路径 fullpath。

- B. 提取 URL 可以由正则表达式提取，其正则表达式为^[2]：

`regex_expression(R"(([A-Z]+) (.*)? HTTP\d\.\d")")`

- C. 求完整路径可以由目录路径 (directorypath) 加上 URL 得到，计算表达式如下面这个式子所示：

`fullpath=directorypath+URL`

- D. 构造 URL 和 fullpath 的流程图如下图 1.3 所示：

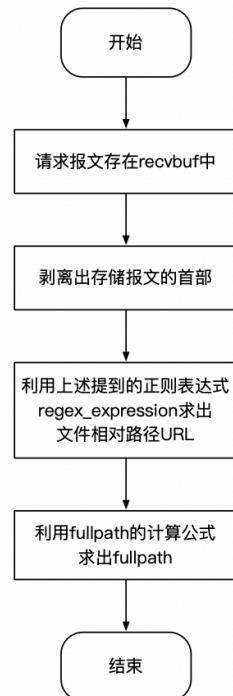


图 1.3 构造 URL 和 fullpath 的流程图

2. 构造并发送响应报文

- A. 我们构造的响应报文报文的公式如下图 1.4 所示：

```
//构造响应报文
respondhead=status_line
    +content_type
    +"Content-Length: "+to_string(flen)+"\r\n"
    +"Server: csr_http1.1\r\n"
    +"Connection: close\r\n"
    +"\r\n";
```

图 1.4 响应报文的构造结构

- B. 由图 1.4 我们知道，构造的响应报文中有三个变量，分别是：status_line、content_type 和flen。其中flen直接调用函数f tell即可求得^[2]。但是剩下两个变量要根据文件的类型进行判断。我们可以取出上述求得的 URL 的文件名的后缀，构造 status_line 和 content_type 的内容，其流程图如下图 1.4 所示：

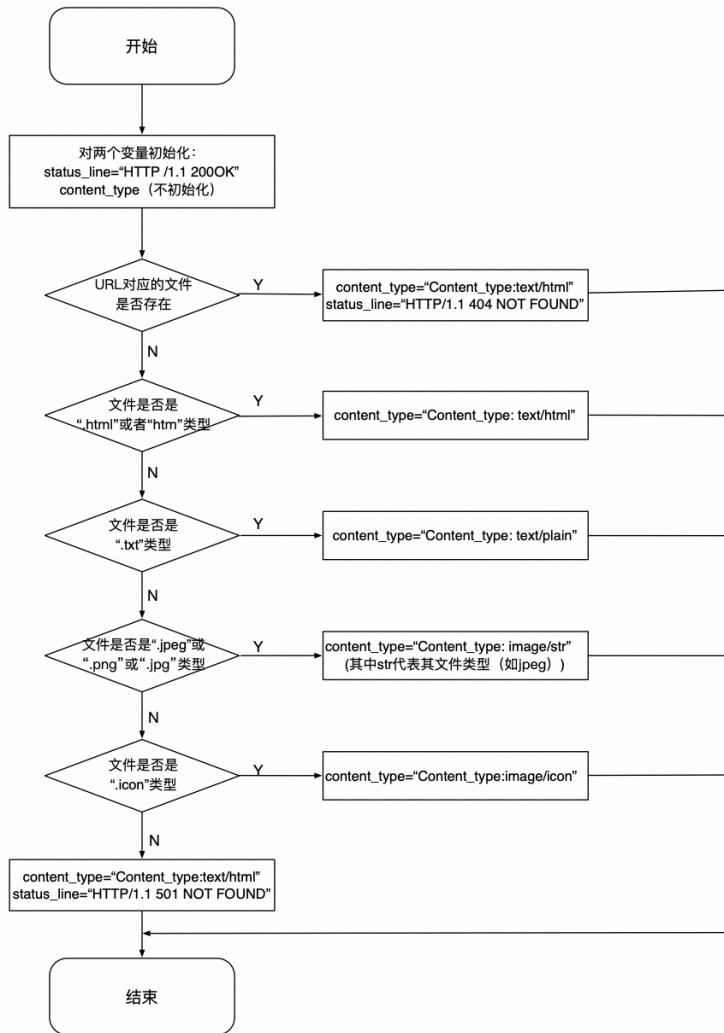


图 1.5 status_line 和 content_type 求解的流程图

- C. 依照流程图 1.5，可求出 status_line 和 content_type，顺利构造出响应报文。

3. 在 UI 中输出列表信息

- A. 我们将输出列表的信息存入一个叫 msg 的结构里面（其类型为 QStringList），在这个结构中依次存入 clientaddr（客户的地址）、clientport（客户的端口）、带有请求文件名称的 URL（file name），以及 result（里面包含报文响应结果，例如 404 NOT FOUND 等等）。
- B. 调用 ui 函数中的列表插入 insert 功能，将其插入到 ui 的信息表中。

1.5 系统测试及结果说明

1. 测试时操作系统的硬件环境

- (1) windows 版本: Windows 10 家庭版
- (2) 处理器: Intel(R) Core(TM) i5-7300U CPU @ 2.60HZ 2.71HZ
- (3) 已安装内存: 8.00GB
- (4) 系统类型: 64 位操作系统, 基于 x64 的处理器
- (5) 浏览器: Google Chrome 71.0.3578.98 64 位

2. 测试前的预备工作: 指定监听地址、监听端口和主目录

A. 监听地址: 127. 0. 0. 1

B. 监听端口: 8080

C. 主目录:

C:/Qt/ComputerNetwork-Lab/build-Lab1-SocketProgramming-Desktop_Qt_5_11_0_MinGW_32bit-Debug/resource

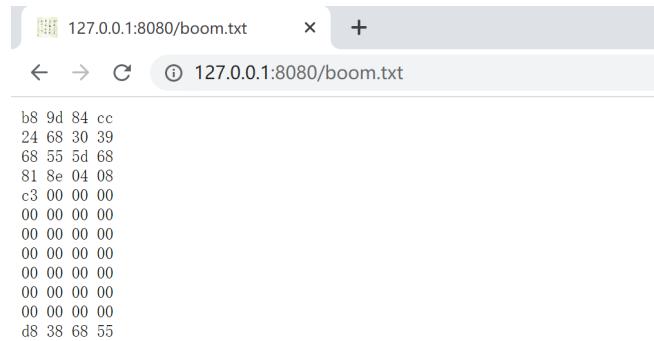
D. 在图形化界面上显示如下图 1.6 所示:



图 1.6 配置地址、端口、主目录

3. 正常功能测试 (测试文本的传输、多媒体文件的传输等等)

- 1) 先尝试传输一个纯文本, 在 google 浏览器中输入网址: 127. 0. 0. 1:8080/boom.txt 来请求这个文件, 结果如下图所示:



b8 9d 84 cc
24 68 30 39
68 55 5d 68
81 8e 04 08
c3 00 00 00
00 00 00 00
00 00 00 00
00 00 00 00
00 00 00 00
00 00 00 00
00 00 00 00
d8 38 68 55

图 1.7 文字测试传输网页显示

可以发现，正常传输并显示网页，程序编写正确。

请求完成后，UI 中显示如下图 1.8 所示：

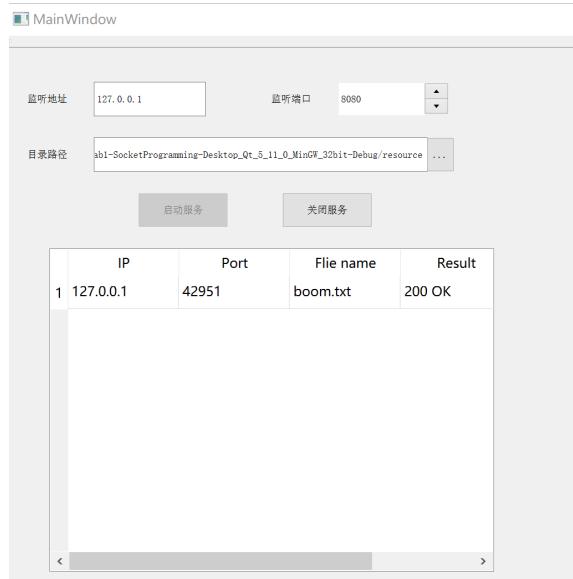


图 1.8 文字传输测试 UI 显示

UI 中正确显示了 IP、端口、文件名以及返回结果（200OK），所以 UI 设计实现正确。

- 2) 传输一个带有引用图片的 html 文件，在浏览器中输入：127.0.0.1:8080/index.html 来请求这个文件，结果如下图 1.9 所示：

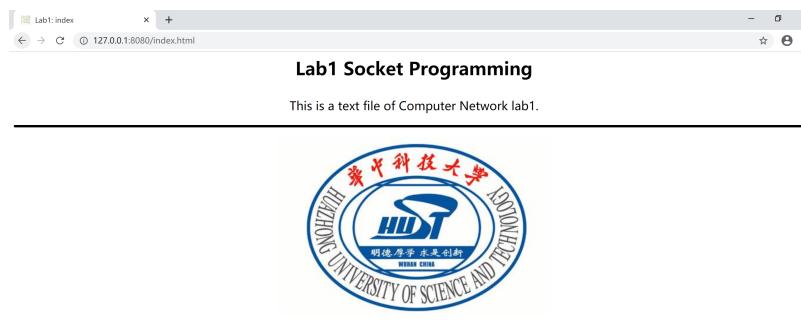


图 1.9 多媒体引用网页显示

可以看出，网页中的文字、图像以及 icon 都能正常显示，证明编写的程序支持文字和多媒体的传输，支持网页引用传输。

UI 中的信息如下图 1.10 所示：

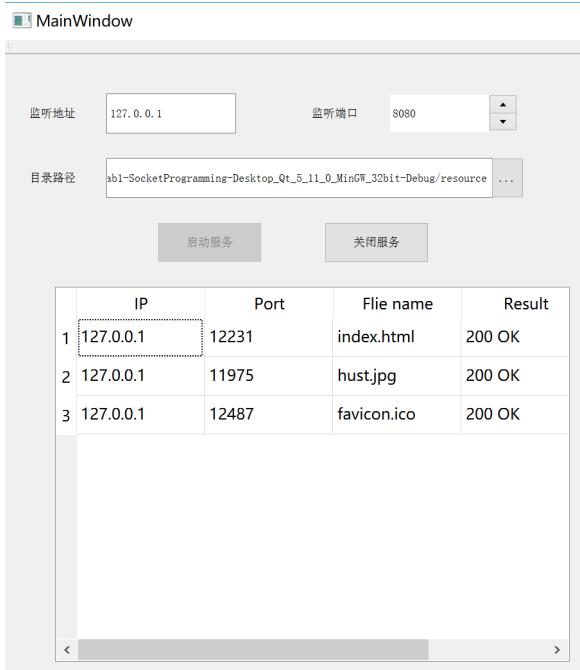


图 1.10 多媒体引用 UI 显示

由上图可以看出，UI 中正确显示了 IP、Port、文件名、传输结构这些内容，UI 设置正确。

其发送报文和接收报文的首部如下所示（这里只展示 index 的）：

```
request header:  
GET /index.html HTTP/1.1  
Host: 127.0.0.1:8080  
Connection: keep-alive  
Cache-Control: max-age=0  
Upgrade-Insecure-Requests: 1  
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/79.0.3945.117 Safari/537.36  
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9  
Sec-Fetch-Site: cross-site  
Sec-Fetch-Mode: navigate  
Accept-Encoding: gzip, deflate, br  
Accept-Language: zh-CN,zh;q=0.9
```

图 1.11 请求报文

```
respond header:  
HTTP/1.1 200 OK  
Content-Type: text/html  
Content-Length: 475  
Server: csr_http1.1  
Connection: close
```

图 1.12 响应报文

4. 压力测试（基于老师给的脚本）

- 1) 将老师给出的脚本改编如下图 1.13 所示（添加网址路径）

```

SETLOCAL
rem 压力测试，替换(1,1,100)中的100为测试数
rem url替换为http://ip:port/index.html

set url=http://127.0.0.1:8080/index.html
set cmd1= curl %url%

for /L %%i in (1,1,100) do %cmd1%
ENDLOCAL
pause

```

图 1.13 压力测试脚本

2) 执行脚本，终端中执行 100 次上述 index 文件的传输，最后 UI 中显示结果如下：

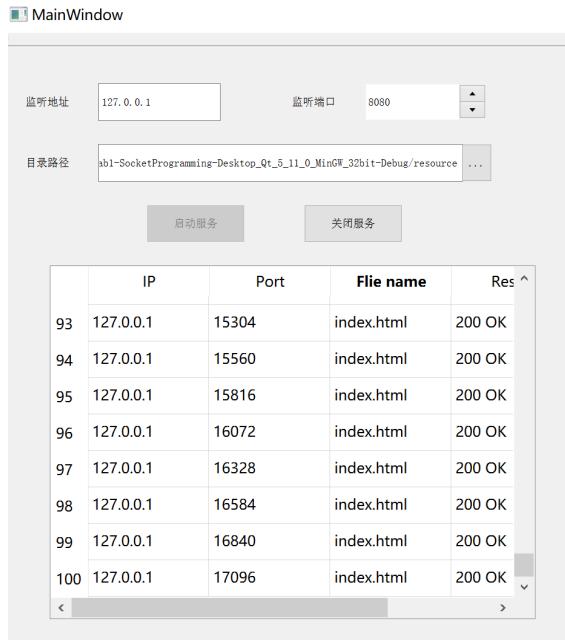


图 1.14 压力测试 UI 截图

可以看出，100 次连接传输都成功了，程序通过了压力测试

5. 多线程测试（基于了老师给的脚本）

1) 将老师给出的脚本改编如下图所示（添加网址路径）

```

SETLOCAL
rem replace the url below with your test url
rem in the format of http://xxx.xxx.xxx.xxx:xxxx/xxxx.html
rem It is a good result with the output "找不到差异"

set url=http://127.0.0.1:8080/index.html
set cmd1= curl %url%
start %cmd1% -o 1.txt
start %cmd1% -o 2.txt
start %cmd1% -o 3.txt
start %cmd1% -o 4.txt
start %cmd1% -o 5.txt
start %cmd1% -o 6.txt

ENDLOCAL
timeout /t 2 /nobreak > nul
fc 1.txt 2.txt
fc 2.txt 3.txt
fc 3.txt 4.txt
fc 4.txt 5.txt
fc 5.txt 6.txt
del 1.txt 2.txt 3.txt 4.txt 5.txt 6.txt
pause

```

图 1.15 多线程测试脚本

2) 执行脚本，执行 6 个线程同时进行请求上述 index 文件，最后终端执行结果如下：

```
C:\Windows\system32\cmd.exe
C:\Qt\ComputerNetwork-Lab\build-Lab1-SocketProgramming-Desktop_Qt_5_11_0_MinGW_32bit-Debug>fc 1.txt 2.txt
正在比较文件 1.txt 和 2.TXT
FC: 找不到差异

C:\Qt\ComputerNetwork-Lab\build-Lab1-SocketProgramming-Desktop_Qt_5_11_0_MinGW_32bit-Debug>fc 2.txt 3.txt
正在比较文件 2.txt 和 3.TXT
FC: 找不到差异

C:\Qt\ComputerNetwork-Lab\build-Lab1-SocketProgramming-Desktop_Qt_5_11_0_MinGW_32bit-Debug>fc 3.txt 4.txt
正在比较文件 3.txt 和 4.TXT
FC: 找不到差异

C:\Qt\ComputerNetwork-Lab\build-Lab1-SocketProgramming-Desktop_Qt_5_11_0_MinGW_32bit-Debug>fc 4.txt 5.txt
正在比较文件 4.txt 和 5.TXT
FC: 找不到差异

C:\Qt\ComputerNetwork-Lab\build-Lab1-SocketProgramming-Desktop_Qt_5_11_0_MinGW_32bit-Debug>fc 5.txt 6.txt
正在比较文件 5.txt 和 6.TXT
FC: 找不到差异

C:\Qt\ComputerNetwork-Lab\build-Lab1-SocketProgramming-Desktop_Qt_5_11_0_MinGW_32bit-Debug>del 1.txt 2.txt 3.txt 4.txt
.txt 6.txt
C:\Qt\ComputerNetwork-Lab\build-Lab1-SocketProgramming-Desktop_Qt_5_11_0_MinGW_32bit-Debug>pause
请按任意键继续...
```

图 1.16 多线程测试脚本运行图

可以看出，6 个文件比较都没有差异，多线程传输正确。

UI 中显示结果如下：

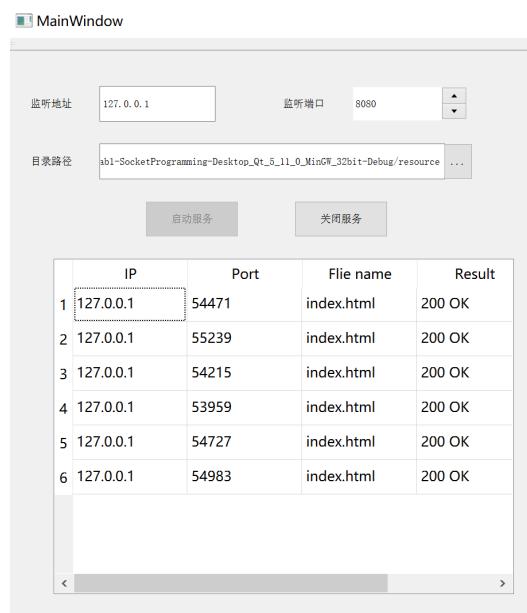


图 1.17 多线程测试 UI 显示

可以看出六个文件都成功传输，多线程测试通过。

6. 异常情况处理测试（请求不存在的文件以及类型不符合传输要求的文件）

1) 请求不符合传输类型的文件^[1]

例如我们请求一个文件 MST. exe，在浏览器输入：127.0.0.1:8080/MST. exe，浏览器返回结果如下图所示（显示的是我自制的 501 页面）：



图 1.18 请求不符合传输类型的文件的网页显示

由于网页不能够显示 exe 类型的文件，所以这里显示 501 NOT IMPLEMENTED，即不

支持这种文件。

UI 中返回结果如下图所示：



图 1.19 请求不符合传输类型的文件的 UI 显示

可以看出，UI 中返回的结果也是 501 NOT IMPLEMENTED，测试正确。

2) 请求不存在的文件^[1]

例如我们请求一个不存在的文件 not_exist.txt，此时，我们在浏览器中输入链接如：127.0.0.1:8080/not_exist.txt，浏览器返回结果如下图所示：



图 1.20 请求不存在文件的网页显示

由于 not_exist.txt 文件没有存在在我们的服务器上，所以请求这个文件会出现没有找到的情况，即 404 NOT FOUND。

UI 中结果返回如下图所示：

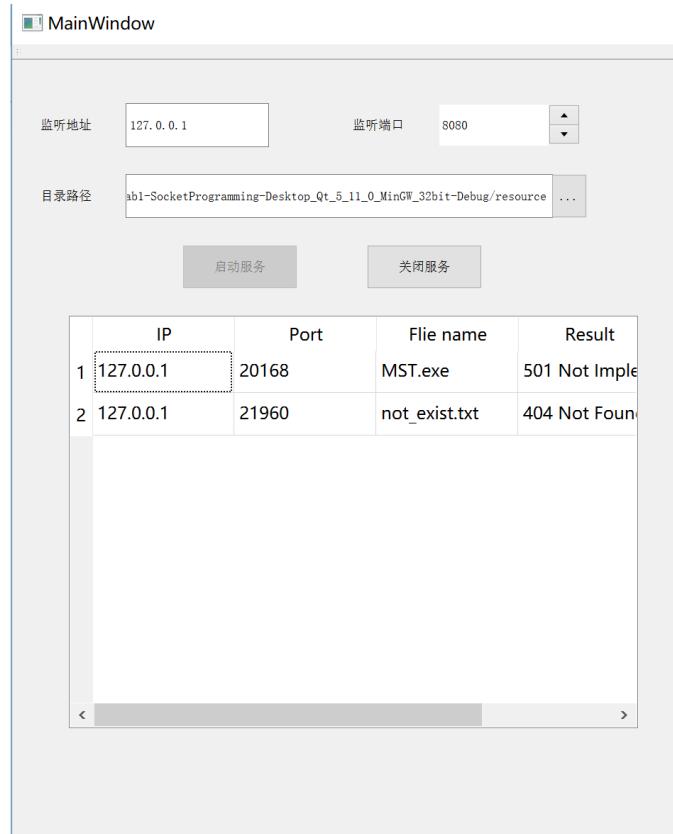


图 1.21 请求不存在的文件的 UI 显示

可以看出，UI 中也显示 not_exist.txt 这个文件是 404 NOT FOUND，测试正确。

1.6 其它需要说明的问题

- (1) QT 不支持中文，如果在编译中路径名中包含中文，可能会编译出错。我在一开始使用 QT 的时候，将其放到一个有着中文目录名的目录下，结果就连 QT 给的基础的图形界面都编译不出来。后来通过浏览大家在群里面的讨论，知道了这是中文路径名的问题，将工程的路径名改成英文的，顺利解决了问题。
- (2) QT 启动之后，到文件加载出来这一段可能需要一点时间，但这时候可以点击编译按钮。切记，这时候不能点击编译按钮，否则会出错。

1.7 参考文献

[1]James F.Kurose, Keith W.Ross, 陈鸣. 计算机网络-自顶向下方法[M]. 北京：机械工业出版社，2018: 242-330

[2]Stephen Prata, 张海龙, 袁国忠. C++ Primer Plus (第六版) 中文版. 人民邮电出版社, 2012 年 7 月 第一版: 12-298

实验二 数据可靠传输协议设计实验

2.1 环境

- (1) windows 版本: Windows 10 家庭版
- (2) 处理器: Intel(R) Core(TM) i5-7300U CPU @ 2.60HZ 2.71HZ
- (3) 已安装内存: 8.00GB
- (4) 系统类型: 64 位操作系统, 基于 x64 的处理器
- (5) IDE (编辑器): Microsoft Visual Studio Community 2017
- (6) 编程语言: C++
- (7) 外部依赖库: 网络环境库 netsimlib.lib

2.2 实验要求

(1) 实验内容

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
2. 实现基于 SR 的可靠传输协议
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。

(2) 一些细节要求

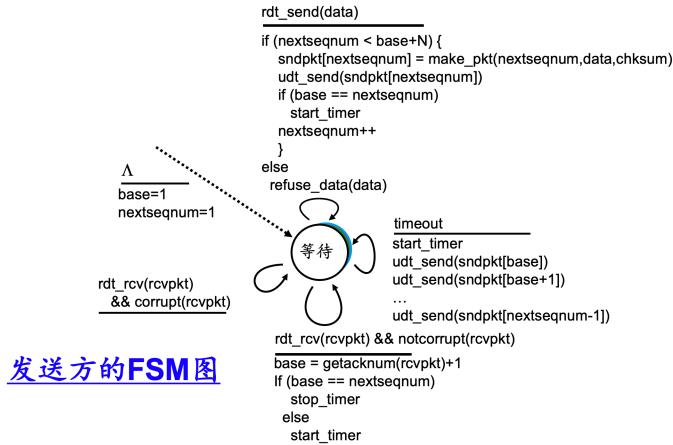
1. 可靠运输层协议实验只考虑单向传输, 即: 只有发送方发生数据报文, 接收方仅接收报文并给出确认报文。
2. 要求实现具体协议时, 指定编码报文序号的二进制位数 (例如 3 位二进制编码报文序号) 以及窗口大小 (例如大小为 4), 报文段序号必须按照指定的二进制位数进行编码。
3. 代码实现不需要基于 Socket API, 不需要利用多线程, 不需要任何 UI 界面。
4. 提交实验设计报告和源代码; 实验设计报告必须按照实验报告模板完成, 源代码必须加详细注释。

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

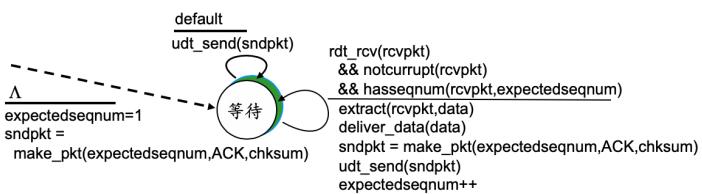
(1) 模块设计（程序的数据结构和函数结构）

根据下面图 2.1 和 2.2 的 FSM, 我们确定出发送方和接收方所需要的数据结构与函数^[1]:



发送方的FSM图

图 2.1 GBN 协议发送方的 FSM



接收方的FSM图

图 2.2 接收方的 FSM

1. 发送方

A. 数据结构

- 1) base: 窗口的基序号
- 2) nextseqnum: 窗口发送的下一个序号
- 3) N: 窗口大小
- 4) k: 序号的长度
- 5) *sndwinbuf: 发送方窗口缓冲区

B. 函数结构

- 1) insendbuf(int seqnum): 判断 seqnum 是否在发送方的滑动窗口中。
- 2) showbuf(): 展示发送方的滑动窗口
- 3) getWaitingState(): 判断是否需要等待
- 4) send(Message& message): 发送方发送报文, 移动滑动窗口
- 5) receive(Packet& ackpkt): 发送方确认接受 Ack, 被网络层调用
- 6) timeoutHandler(int seqnum): 超时重传

2. 接收方

A. 数据结构

- 1) expectSequenceNumberRcvd: 期待收到的下一个报文的序号
- 2) lastAckPkt: 上次发送的确认报文
- 3) k: 序号长度

B. 函数结构

- 1) receive(Packet& packet): 接受发送方报文，交给网络层调用。

(2) 函数的具体实现方式

1. 发送方

A. send(Message& message)

函数的具体实现方式如下流程图 2.3 所示^[1]:

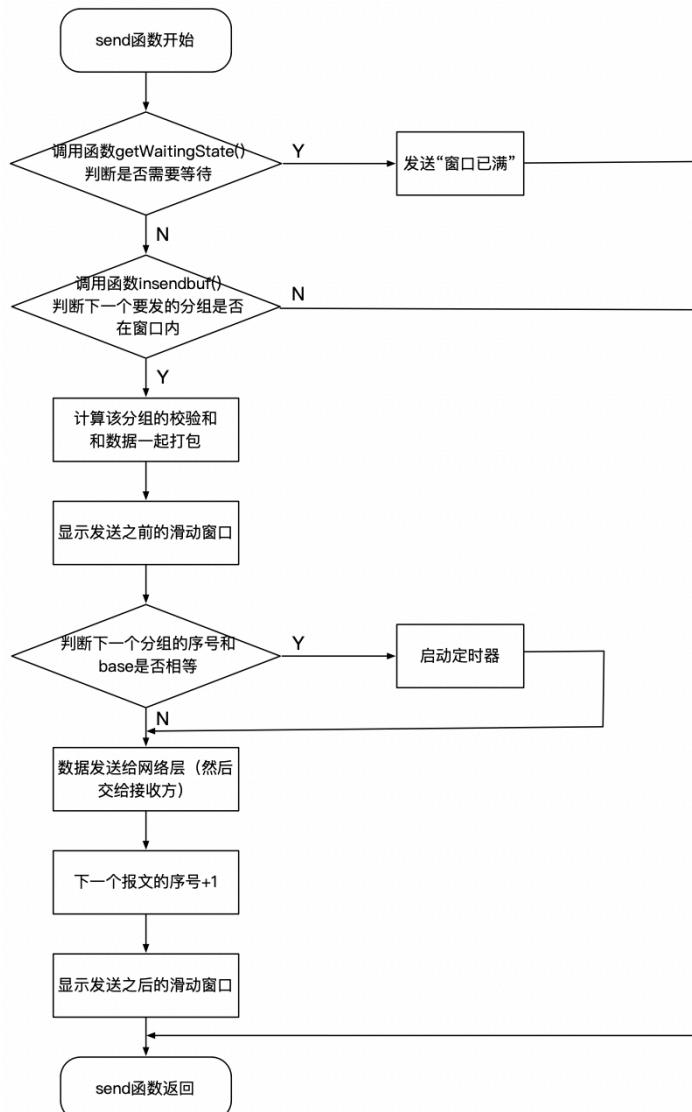


图 2.3 发送方 send 函数流程图

B. receive(Packet& packet)

函数的具体实现方式如下流程图 2.4 所示^[1]:

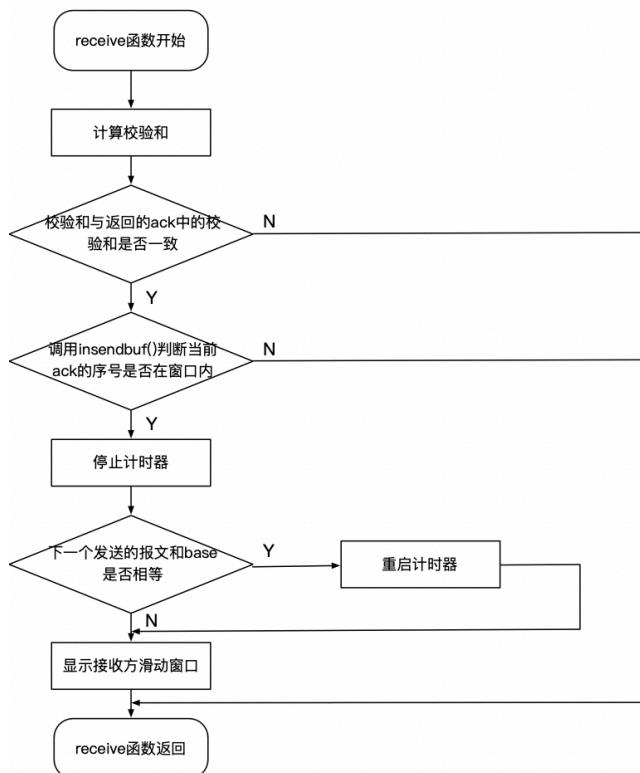


图 2.4 发送方 receive 函数流程图

C. getWaitingState()

- 1) 计算得到当前窗口容量能够到达的序号
- 2) 计算当前窗口已经占用到的序号
- 3) 比较能够到达的序号和占用序号是否一致，如果一致，证明已经占用到了最后一个可占用的位置，再来数据包就需要等待。

D. timeoutHandler(int seqnum)

- 1) 发送有分组超时的信息。
- 2) 将该分组发送给网络层（重传给接收方）
- 3) 停止计时器，然后重新启动计时器。
- 4) 发送重传完成的信息。

E. insendbuf(int seqnum)

- 1) 计算得到当前窗口序号的开始 (base) 和结束 (end)
- 2) 判断 seqnum 是否处于 base 和 end 之间，如果处于这个区间，则数据包在发送方窗口中。

F. showbuf()

- 1) 对于窗口内的每个位置 i，进行以下几个步骤的操作：

- 2) 如果遇到 $i=base$ 的情况，输出 “(”，表示滑动窗口的左边界
- 3) 如果 $i=nextseqnum$, 则代表碰到了下一个待发送的分组，输出分隔符“ | ”，分隔符前面是已经发送，分隔符后面是待发送。
- 4) 如果 $i=$ 窗口结尾，输出 “)” 表示窗口的右边界。

2. 接收方

A. receive

函数的具体实现方式如下流程图 2.5 所示^[1]:

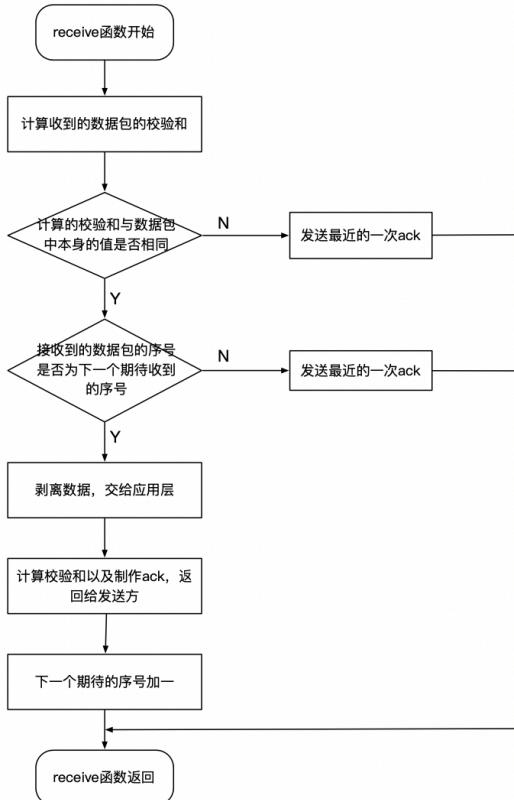


图 2.5 接收方 receive 函数流程图

(3) 结果验证与分析

1. 编译运行 gbn 的程序，得到执行文件 Lab2-RdtTrans-gbn. exe^[2]。
2. 改编测试脚本如下图 2.6 所示：

```

set appname="C:\\Qt\\ComputerNetwork-Lab\\Lab2-RdtTrans\\Lab2-RdtTrans-gbn.exe"
set inputname="C:\\Qt\\ComputerNetwork-Lab\\Lab2-RdtTrans\\input.txt"
set outputname="C:\\Qt\\ComputerNetwork-Lab\\Lab2-RdtTrans\\log\\gbn_output.txt"
set resultname="C:\\Qt\\ComputerNetwork-Lab\\Lab2-RdtTrans\\log\\gbn_out.log"

for /l %%i in (1,1,10) do (
  echo Test %appname% %%i:
  %appname% > %resultname% 2>&1
  fc /N %inputname% %outputname%
)
pause
  
```

图 2.6 GBN 测试脚本改编图

3. 执行脚本后如下图所示：

```
Test "C:\Qt\ComputerNetwork-Lab\Lab2-RdtTrans\Lab2-RdtTrans-gbn.exe" 5:  
正在比较文件 C:\Qt\COMPUTERNETWORK-LAB\LAB2-RDTTRANS\input.txt 和 C:\Qt\COMPUTERNETWORK-LAB\LAB2-RDTTRANS\LOG\GBN_OUTPUT.TXT  
FC: 找不到差异  
  
Test "C:\Qt\ComputerNetwork-Lab\Lab2-RdtTrans\Lab2-RdtTrans-gbn.exe" 6:  
正在比较文件 C:\Qt\COMPUTERNETWORK-LAB\LAB2-RDTTRANS\input.txt 和 C:\Qt\COMPUTERNETWORK-LAB\LAB2-RDTTRANS\LOG\GBN_OUTPUT.TXT  
FC: 找不到差异  
  
Test "C:\Qt\ComputerNetwork-Lab\Lab2-RdtTrans\Lab2-RdtTrans-gbn.exe" 7:  
正在比较文件 C:\Qt\COMPUTERNETWORK-LAB\LAB2-RDTTRANS\input.txt 和 C:\Qt\COMPUTERNETWORK-LAB\LAB2-RDTTRANS\LOG\GBN_OUTPUT.TXT  
FC: 找不到差异  
  
Test "C:\Qt\ComputerNetwork-Lab\Lab2-RdtTrans\Lab2-RdtTrans-gbn.exe" 8:  
正在比较文件 C:\Qt\COMPUTERNETWORK-LAB\LAB2-RDTTRANS\input.txt 和 C:\Qt\COMPUTERNETWORK-LAB\LAB2-RDTTRANS\LOG\GBN_OUTPUT.TXT  
FC: 找不到差异  
  
Test "C:\Qt\ComputerNetwork-Lab\Lab2-RdtTrans\Lab2-RdtTrans-gbn.exe" 9:  
正在比较文件 C:\Qt\COMPUTERNETWORK-LAB\LAB2-RDTTRANS\input.txt 和 C:\Qt\COMPUTERNETWORK-LAB\LAB2-RDTTRANS\LOG\GBN_OUTPUT.TXT  
FC: 找不到差异  
  
Test "C:\Qt\ComputerNetwork-Lab\Lab2-RdtTrans\Lab2-RdtTrans-gbn.exe" 10:  
正在比较文件 C:\Qt\COMPUTERNETWORK-LAB\LAB2-RDTTRANS\input.txt 和 C:\Qt\COMPUTERNETWORK-LAB\LAB2-RDTTRANS\LOG\GBN_OUTPUT.TXT  
FC: 找不到差异  
  
请按任意键继续... .
```

图 2.7 GBN 脚本测试运行图

由图中的结果可以看出，input 文件和 gbn 测试输出的 output 文件完全相同，程序运行成功。

4. 观察滑动窗口的移动截图如下所示：

```
*****模拟网络环境*****: 模拟网络环境启动...  
发送方: 发送前的滑动窗口:  
(0 1 2 3 4) 5 6  
*****模拟网络环境*****: 启动定时器, 当前时间 = 1.9675, 定时器报文序号 = 0, 定时器Timeout时间 = 21.9675  
*****模拟网络环境*****: 发送方的数据包将在4.3175到达对方, 数据包为-->seqnum = 0, acknum = -1,checksum = 29556,payload = AAAAAAAAAAAAAAAAAAAAAA  
发送方: 发送后的滑动窗口:  
(0 1 2 3 4) 5 6  
*****模拟网络环境*****: 向上述交给应答层数据: AAAAAAAAAAAAAAAA  
*****模拟网络环境*****: 接收方的确认包将在12.7575到达对方, 确认包为-->seqnum = -1, acknum = 0,checksum = 12851,payload = .....  
接收方: 目前返回的ACK序号为: 0  
发送方: 接收到ACK: 0  
*****模拟网络环境*****: 关闭定时器, 当前时间 = 12.7575, 定时器报文序号 = 0  
发送方: 接受ACK后的滑动窗口:  
(0 1 2 3 4) 5 6  
发送方: 发送前的滑动窗口:  
(0 1 2 3 4) 5 6  
*****模拟网络环境*****: 启动定时器, 当前时间 = 13.14, 定时器报文序号 = 0, 定时器Timeout时间 = 33.14  
*****模拟网络环境*****: 发送方的数据包将在15.78到达对方, 数据包为-->seqnum = 1, acknum = -1,checksum = 26985,payload = BBBBBBBBBBBBBBBBBBBBBBB  
发送方: 发送后的滑动窗口:  
(0 1 2) 3 4 5 6  
发送方: 发送前的滑动窗口:  
(0 1 2 3) 4 5 6  
*****模拟网络环境*****: 发送方的数据包将在25.62到达对方, 数据包为-->seqnum = 2, acknum = -1,checksum = 24414,payload = CCCCCCCCCCCCCCCCCCCC  
发送方: 发送后的滑动窗口:  
(0 1 2 3 4) 5 6  
发送方: 发送前的滑动窗口:  
(0 1 2 3 4) 5 6  
*****模拟网络环境*****: 发送方的数据包将在32.06到达对方, 数据包为-->seqnum = 3, acknum = -1,checksum = 21843,payload = DDDDDDDDDDDDDDDDDDDDD  
发送方: 发送后的滑动窗口:  
(0 1 2 3 4) 5 6  
发送方: 发送前的滑动窗口:  
(0 1 2 3 4) 5 6
```

图 2.8 GBN 的滑动窗口截图

图中清晰地展示了滑动窗口随着接收信息的变动过程。可以看出，每发送一个信息，“|”向后移动一次（即下一个发送单位向后移动一次）；每收到一个信息，窗口向后移动一次。综上，滑动窗口设计正确。

2.3.2 SR 协议的设计、验证及结果分析

(1) 模块设计（程序的数据结构和函数结构）

1. 发送方

A. 数据结构

- 1) base: 窗口基序号
- 2) nextseqnum: 下一个发送序号
- 3) N: 发送窗口大小
- 4) k: 序列长度
- 5) *ifrcv: 窗口中的每个位置是否已经被占据

6) *sndwinbuf: 发送方窗口缓存

B. 函数结构

- 1) insendbuf(int seqnum): 判断 seqnum 是否在发送方的滑动窗口中
- 2) showbuf(): 展示发送方的滑动窗口
- 3) getWaitingState(): 判断当前是否要等待
- 4) send(Message& message): 发送方发送报文, 移动滑动窗口
- 5) receive(Packet& ackpkt): 发送方确认接收 Ack, 交给网络层调用
- 6) timeoutHandler(int seqnum): 超时重传

2. 接收方

A. 数据结构

- 1) base: 接收窗口基序号
- 2) N: 接收窗口大小
- 3) k: 序列长度
- 4) *ifrcvbuf: 窗口中每个位置是否已经被占据
- 5) *rcvwinbuf: 接收方窗口缓存
- 6) lastAckPkt: 上次发送的确认报文

B. 函数结构

- 1) inrcvbuf(int seqnum): 判断 seqnum 是否在发送方的滑动窗口中
- 2) showbuf(): 展示发送方的滑动窗口
- 3) receive(Packet& packet): 接受报文, 交给网络层调用。

(2) 函数的具体实现方式

1. 发送方

A. send(Message& message)

- 1) 调用 getWaitingState(), 如果当前发送需要等待, 则等待。
- 2) 调用 insendbuf 函数, 判断 nextseqnum 是否在窗口内, 如果在则进行第 3 步, 不在则直接跳到第 7 步。
- 3) nextseqnum 在窗口内, 计算该数据包的 checksum, 和数据包一起打包。
- 4) 展示发送方发送前的滑动窗口
- 5) 为该分组启动其定时器, 并且发送给网络层
- 6) 显示发送后的滑动窗口
- 7) nextseqnum 不在窗口内, 则函数直接返回 false, 忽略该发送包的处理

B. receive(Packet& packet)

- 1) 计算收到的 Ack 的校验和, 和 Ack 里面的校验和进行比较, 如果不相同则发送分组损坏的消息, 不停止计时器。
- 2) 如果校验和一样, 则准备接收返回的报文, 步骤如下面几步:

-
- 3) 先停止该分组的计时器，将当前窗口的 ifrcv 数组中的值变为 true，标记为当前窗口已接收^[2]
 - 4) 循环移动窗口，将窗口移动到下一个待接收的位置（跳过所有 ifrcv 为 true 的窗口）
 - 5) 显示循环移动后的窗口
- C. getWaitingState()
- 1) 计算得到当前窗口容量能够到达的序号
 - 2) 计算当前窗口已经占用到的序号
 - 3) 比较能够到达的序号和占用序号是否一致，如果一致，证明已经占用到了最后一个可占用的位置，再来数据包就需要等待。
- D. insenbuf(int seqnum)
- 1) 计算得到当前窗口序号的开始 (base) 和结束 (end)
 - 2) 判断 seqnum 是否处于 base 和 end 之间，如果处于这个区间，则数据包在发送方窗口中。
- E. showbuf()
- 1) 对于窗口内的每个位置 i，进行以下几个步骤的操作：
 - 2) 如果当前位置 i=base，输出 “(”，表示滑动窗口的左边界
 - 3) 如果当前位置 i 在 sendbuf 中（调用 insendbuf 函数）且 i 比 nextseqnum 大，则该位置是空的，输出 “empty”。
 - 4) 如果当前位置 i 在 sendbuf 中，且该位置处于已经发送但是没收到 ack 的状态，在该位置输出 “sent”。
 - 5) 如果当前位置 i 在 sendbuf 中，且该位置处于已经收到 ack 的状态，则在该位置输出 “acked”。
 - 6) 如果当前的位置 i 处于窗口的结尾状态，输出 “)”，表示滑动窗口的右边界。
 - 7) 其余的不在窗口内的位置 i，输出 “X”。

2. 接收方

- A. receive(Packet& packet)
- 1) 接收到发送方的数据包之后，计算校验和，如果校验和不对，则输出分组损坏的错误信息，函数返回。
 - 2) 判断当前接收的包的序号是否在接收方窗口内，如果不在窗口内，先提示不是接收方窗口，然后再重新计算上一次发送的 ack 的校验码，和上一次的 ack 一起打包，然后发送给网络层（交给接收方）
 - 3) 如果当前接收的包在接收方窗口内，则在 rcvwinbuf 中存储该数据，在 ifrcv 数组中将当前序号标识为 true（已经接收状态）。然后计算校验码，

和 ack 打包，一起发送至网络层返回给发送方。

- 4) 滑动窗口，指向下一个没有接收的位置，即依次遍历所有 ifrcv 为 true 的位置，直到遇到一个为 false 的位置停止即可。

- 5) 展示滑动后的接收方的滑动窗口

B. inrcvbuf(int seq)

- 1) 计算得到当前窗口序号的开始 (base) 和结束 (end)
- 2) 判断 seqnum 是否处于 base 和 end 之间，如果处于这个区间，则数据包在接收方窗口中。

C. Showbuf()

- 1) 对于窗口内的每个位置 i，进行以下几个步骤的操作：
- 2) 如果当前位置 i=base，即输出 “(”，代表滑动窗口的左边界。
- 3) 如果位置 i 在滑动窗口中（调用 inrcvbuf 函数），而且 ifrcv 的值为真，则代表现在是已经缓存的状态，输出 “stored”。
- 4) 如果位置 i 在滑动窗口中，但是 ifrcv 的值不为真，代表这个位置还可以接收，输出 “empty”。
- 5) 遇到位置 i 为滑动窗口的结尾，输出 “)” ，代表滑动窗口的右边界。
- 6) 如果 i 不在滑动窗口中，则输出 “X”。

(3) 结果验证与分析

1. 编译运行 sr 的程序，得到执行文件 Lab2-RdtTrans-sr.exe^[2]。
2. 改编测试脚本如下图所示：

```
set appname="C:\Qt\ComputerNetwork-Lab\Lab2-RdtTrans\Lab2-RdtTrans-sr.exe"
set inputname="C:\Qt\ComputerNetwork-Lab\Lab2-RdtTrans\input.txt"
set outputname="C:\Qt\ComputerNetwork-Lab\Lab2-RdtTrans\log\sr_output.txt"
set resultname="C:\Qt\ComputerNetwork-Lab\Lab2-RdtTrans\log\sr_out.log"

for /l %%i in (1,1,10) do (
    echo Test %appname% %%i:
    %appname% > %resultname% 2>&1
    fc /N %inputname% %outputname%
)
pause
```

图 2.9 SR 运行脚本改编图

3. 执行脚本后如下图 2.10 所示：

```
Test "C:\Qt\ComputerNetwork-Lab\Lab2-RdtTrans\Lab2-RdtTrans-sr.exe" 5:
正在比较文件 C:\Qt\COMPUTERNETWORK-LAB\LAB2-RDTTRANS\input.txt 和 C:\Qt\COMPUTERNETWORK-LAB\LAB2-RDTTRANS\LOG\SR_OUTPUT.TXT
FC: 找不到差异

Test "C:\Qt\ComputerNetwork-Lab\Lab2-RdtTrans\Lab2-RdtTrans-sr.exe" 6:
正在比较文件 C:\Qt\COMPUTERNETWORK-LAB\LAB2-RDTTRANS\input.txt 和 C:\Qt\COMPUTERNETWORK-LAB\LAB2-RDTTRANS\LOG\SR_OUTPUT.TXT
FC: 找不到差异

Test "C:\Qt\ComputerNetwork-Lab\Lab2-RdtTrans\Lab2-RdtTrans-sr.exe" 7:
正在比较文件 C:\Qt\COMPUTERNETWORK-LAB\LAB2-RDTTRANS\input.txt 和 C:\Qt\COMPUTERNETWORK-LAB\LAB2-RDTTRANS\LOG\SR_OUTPUT.TXT
FC: 找不到差异

Test "C:\Qt\ComputerNetwork-Lab\Lab2-RdtTrans\Lab2-RdtTrans-sr.exe" 8:
正在比较文件 C:\Qt\COMPUTERNETWORK-LAB\LAB2-RDTTRANS\input.txt 和 C:\Qt\COMPUTERNETWORK-LAB\LAB2-RDTTRANS\LOG\SR_OUTPUT.TXT
FC: 找不到差异

Test "C:\Qt\ComputerNetwork-Lab\Lab2-RdtTrans\Lab2-RdtTrans-sr.exe" 9:
正在比较文件 C:\Qt\COMPUTERNETWORK-LAB\LAB2-RDTTRANS\input.txt 和 C:\Qt\COMPUTERNETWORK-LAB\LAB2-RDTTRANS\LOG\SR_OUTPUT.TXT
FC: 找不到差异

Test "C:\Qt\ComputerNetwork-Lab\Lab2-RdtTrans\Lab2-RdtTrans-sr.exe" 10:
正在比较文件 C:\Qt\COMPUTERNETWORK-LAB\LAB2-RDTTRANS\input.txt 和 C:\Qt\COMPUTERNETWORK-LAB\LAB2-RDTTRANS\LOG\SR_OUTPUT.TXT
FC: 找不到差异

请按任意键继续...
```

图 2.10 SR 脚本执行图

由图中的结果可以看出，input 文件和 sr 测试输出的 output 文件完全相同，程序运行成功。

4. 观察滑动窗口的移动截图如下图 2.11 所示：

```
sr_output.log - 记事本
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)
*****模拟网络环境*****: 模拟网络环境启动...
发送方: 发送方发送前的滑动窗口:
(0:empty 1:empty 2:empty 3:empty )4:X 5:X 6:X 7:X
*****模拟网络环境*****: 启动定时器, 当前时间 = 0.4225, 定时器报文序号 = 0, 定时器Timeout时间 = 20.4225
*****模拟网络环境*****: 发送方的数据包将在10.4725到达对方, 数据包为-->seqnum = 0, acknum = -1,checksum = 29556,payload = AAAAAAAAAAAAAAAA
发送方: 发送方发送后的滑动窗口:
(0:sent 1:empty 2:empty 3:empty )4:X 5:X 6:X 7:X
接收方: 收到数据包, 序号为: 0
*****模拟网络环境*****: 接收方的确认包将在20.9025到达对方, 确认包为-->seqnum = -1, acknum = 0,checksum = 12851,payload = .....
*****模拟网络环境*****: 向上递交应用层数据: AAAAAAAAAAAAAAAA
接收方: 接收端窗口滑动:
0:X 1:empty 2:empty 3:empty 4:empty )5:X 6:X 7:X
*****模拟网络环境*****: 发送方发送的数据包丢失: seqnum = 0, acknum = -1,checksum = 29556,payload = AAAAAAAAAAAAAAAA
*****模拟网络环境*****: 关闭定时器, 当前时间 = 20.4225, 定时器报文序号 = 0
*****模拟网络环境*****: 启动定时器, 当前时间 = 20.4225, 定时器报文序号 = 0, 定时器Timeout时间 = 40.4225
发送方: 接收到ACK: 0
*****模拟网络环境*****: 关闭定时器, 当前时间 = 20.9025, 定时器报文序号 = 0
发送方: 收到对方返回的ACK,滑动窗口变为:
0:X 1:empty 2:empty 3:empty 4:empty )5:X 6:X 7:X
发送方: 发送方发送前的滑动窗口:
0:X 1:empty 2:empty 3:empty 4:empty )5:X 6:X 7:X
*****模拟网络环境*****: 启动定时器, 当前时间 = 21.5725, 定时器报文序号 = 1, 定时器Timeout时间 = 41.5725
*****模拟网络环境*****: 发送方的数据包将在23.3125到达对方, 数据包为-->seqnum = 1, acknum = -1,checksum = 26985,payload = BBBBBBBBBBBBBBBBBBBB
发送方: 发送方发送后的滑动窗口:
0:X 1:sent 2:empty 3:empty 4:empty )5:X 6:X 7:X
```

图 2.11 SR 滑动窗口移动示意图

图中清晰地展示了发送方和接收方滑动窗口随着接收信息的变动过程。可以看出，发送方每发送一次信息，就会将窗口中对应位置为 sent，每收到一个信息，都会将窗口中的信息记为 acked，并滑动窗口。接收方同样也会根据接收的信息和缓存的信息，滑动窗口。可见，滑动窗口设计正确^[1]。

2.3.3 简单 TCP/IP 协议的设计、验证及结果分析

(1) 模块设计、数据结构和函数设计

1. 本模块是基于第一个实验的 GBN 的协议上改造而成的，程序的整体模块是由 GBN 协议的实现加上三次冗余 ACK 重传的机制而构成的。
2. 在 GBN 发送方模块的基础上，增加一个数据结构叫“resend”，类型为 int 型，功能是记录目前已经得到的冗余 Ack 数量，当这个变量超过 3 的时候触发重传。
3. 在 GBN 发送方模块的基础上，在 receive 函数中增加冗余 Ack 重传机制。
4. TCP 的接收方模块和 GBN 的接收方模块完全一致。

(2) 冗余 ACK 重传的具体实现

- 1) 我们在 receive 函数中处理冗余 ack 重传的问题。若当前收到的 ack 号不在发送方的窗口中，我们执行以下操作：
- 2) 如果 resend（即重传累积）已经积累到了 3，则我们重传，将当前 base 对应的数据包传给接收方，同时将 resend 置 0，程序返回。
- 3) 否则我们将 resend+1，即积累重传次数。

(3) 结果验证与分析

1. 编译运行 tcp 的程序，得到执行文件 Lab2-RdtTrans-tcp.exe^[1]。
2. 改编测试脚本如下图 2.12 所示：

```

set appname="C:\Qt\ComputerNetwork-Lab\Lab2-RdtTrans\Lab2-RdtTrans-tcp.exe"
set inputname="C:\Qt\ComputerNetwork-Lab\Lab2-RdtTrans\input.txt"
set outputname="C:\Qt\ComputerNetwork-Lab\Lab2-RdtTrans\log\tcp_output.txt"
set resultname="C:\Qt\ComputerNetwork-Lab\Lab2-RdtTrans\log\tcp_out.log"

for /l %%i in (1,1,10) do (
    echo Test %appname% %%i:
    %appname% > %resultname% 2>&1
    fc /N %inputname% %outputname%
)
pause

```

图 2.12 TCP 脚本改编图

3. 执行脚本后如下图所示：

```

Test "C:\Qt\ComputerNetwork-Lab\Lab2-RdtTrans\Lab2-RdtTrans-tcp.exe" 5:
正在比较文件 C:\Qt\COMPUTERNETWORK-LAB\LAB2-RDTTRANS\input.txt 和 C:\Qt\COMPUTERNETWORK-LAB\LAB2-RDTTRANS\LOG\tcp_output.txt
FC: 找不到差异

Test "C:\Qt\ComputerNetwork-Lab\Lab2-RdtTrans\Lab2-RdtTrans-tcp.exe" 6:
正在比较文件 C:\Qt\COMPUTERNETWORK-LAB\LAB2-RDTTRANS\input.txt 和 C:\Qt\COMPUTERNETWORK-LAB\LAB2-RDTTRANS\LOG\tcp_output.txt
FC: 找不到差异

Test "C:\Qt\ComputerNetwork-Lab\Lab2-RdtTrans\Lab2-RdtTrans-tcp.exe" 8:
正在比较文件 C:\Qt\COMPUTERNETWORK-LAB\LAB2-RDTTRANS\input.txt 和 C:\Qt\COMPUTERNETWORK-LAB\LAB2-RDTTRANS\LOG\tcp_output.txt
FC: 找不到差异

Test "C:\Qt\ComputerNetwork-Lab\Lab2-RdtTrans\Lab2-RdtTrans-tcp.exe" 9:
正在比较文件 C:\Qt\COMPUTERNETWORK-LAB\LAB2-RDTTRANS\input.txt 和 C:\Qt\COMPUTERNETWORK-LAB\LAB2-RDTTRANS\LOG\tcp_output.txt
FC: 找不到差异

Test "C:\Qt\ComputerNetwork-Lab\Lab2-RdtTrans\Lab2-RdtTrans-tcp.exe" 10:
正在比较文件 C:\Qt\COMPUTERNETWORK-LAB\LAB2-RDTTRANS\input.txt 和 C:\Qt\COMPUTERNETWORK-LAB\LAB2-RDTTRANS\LOG\tcp_output.txt
FC: 找不到差异

请按任意键继续...

```

图 2.13 TCP 脚本运行图

由图中的结果可以看出，input 文件和 tcp 测试输出的 output 文件完全相同，程序运行成功。

4. 三次 ack 重传验证

下图 2.14 展现了收到了三个冗余的 ACK 之后，重传的效果：

```

发送方: 有丢包发生时: seqnum = 1, acknum = -1, checksum = 26985, BBBBBBBBBBBBBBBBBB
发送方: 重传完成: seqnum = 1, acknum = -1, checksum = 26985, BBBB BBBB BBBB BBBB BBBB
接收方: 这个序号不是接收方所期待的 : seqnum = 3, acknum = -1, checksum = 21843, DDDDDDDDDDDDDDDDDDD
接收方: 这个序号不是接收方所期待的 : seqnum = 4, acknum = -1, checksum = 19272, EEEEEEEEEE
发送方: 收到三个冗余ACK, 进行重传: seqnum = 1, acknum = -1, checksum = 26985, BBBB BBBB BBBB BBBB

```

图 2.14 收到了三个冗余的 ACK 重传

可以看出，下图在收到了三个冗余的 ACK 之后，发送方提示进行重传，与程序设计预期相符合。

5. 观察滑动窗口的移动截图如下所示：

```

*****模拟网络环境*****: 模拟网络环境启动...
发送方: 发送方发送前的滑动窗口:
[0|1|2|3|4|5|6]
*****模拟网络环境*****: 启动定时器, 当前时间 = 0.1125, 定时器报文序号 = 0, 定时器Timeout时间 = 20.1125
*****模拟网络环境*****: 发送方的数据包将在7.2225到达对方, 数据包为-->seqnum = 0, acknum = -1,checksum = 29556,payload = AAAAAAAAAAAAAAAA
发送方: 发送方发送后的滑动窗口:
[0|1|2|3|4|5|6]
*****模拟网络环境*****: 向上递交应用层数据: AAAAAAAAAAAAAAAA
*****模拟网络环境*****: 接收方的确认包将在9.2025到达对方, 确认包为-->seqnum = -1, acknum = 0,checksum = 12851,payload = .....
目前返回的ack序号为: 0

发送方: 接收到ACK: 0
*****模拟网络环境*****: 关闭定时器, 当前时间 = 9.2025, 定时器报文序号 = 0
发送方: 发送方接收ACK后的滑动窗口:
[0|1|2|3|4|5|6]
发送方: 发送方发送前的滑动窗口:
[0|1|2|3|4|5|6]
*****模拟网络环境*****: 启动定时器, 当前时间 = 10.07, 定时器报文序号 = 0, 定时器Timeout时间 = 30.07
*****模拟网络环境*****: 发送方的数据包将在18.01到达对方, 数据包为-->seqnum = 1, acknum = -1,checksum = 26985,payload = BBBBBBBBBBBBBBBBBB
发送方: 发送方发送后的滑动窗口:
[0|1|2|3|4|5|6]
发送方: 发送方发送前的滑动窗口:
[0|1|2|3|4|5|6]

```

图 2.15 滑动窗口的示意图

图中清晰地展示了发送方滑动窗口随着接收信息的变动过程。由于 tcp 的滑动窗口设计和 gbn 的完全相同，这里就不再多叙述了。

2.4 其它需要说明的问题

在实验中，有的时候会发现，在同一个时间段内，多次频繁运行模拟网络环境有可能会出现又的时候传输进入死循环的情况，但是过一段时间之后，又能够正常运行。这是由于提供的模拟网络环境有时候多次运行导致的不稳定现象产生的，可以尝试过一段时间再运行或者反复运行一个例子。

2.5 参考文献

[1]James F.Kurose, Keith W.Ross, 陈鸣. 计算机网络-自顶向下方法[M]. 北京: 机械工业出版社, 2018: 242-330

[2]Stephen Prata, 张海龙, 袁国忠, C++ Primer Plus (第六版) 中文版, 人民邮电出版社, 2012 年 7 月第一版: 12-298

实验三 基于 CPT 的组网实验

3.1 环境

- (1) windows 版本: Windows 10 家庭版
- (2) 处理器: Intel(R) Core(TM) i5-7300U CPU @ 2.60HZ 2.71HZ
- (3) 已安装内存: 8.00GB
- (4) 系统类型: 64 位操作系统, 基于 x64 的处理器
- (5) 实验软件: Cisco Packet Tracer 网络仿真工具

3.2 实验要求

- (1) 熟悉 Cisco Packet Tracer 仿真软件, 理解该工具的使用原理、方法及其局限性。
- (2) 利用 Cisco Packet Tracer 仿真软件完成设计步骤和实验内容, 内容如下:
 - 实验 1(基础部分): 给出指定的网络拓扑图, 按要求进行 IP 地址规划和 Vlan 划分。
 - 实验 2(基础部分): 给出指定的网络拓扑图, 按要求进行路由器的配置。
 - 实验 3(综合部分): 按照实验要求, 以学校为背景, 设计组网。

详细实验要求参见第 3 节。

-
- (3) 实验报告中记录详细分配步骤和设计步骤。
 - (4) 对配置好的网络进行测试，查看是否能完成预期目标。

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

(1) 实验 1 简介

- 1. 要求使用仿真软件，按照图 3-1 画出网络拓扑图
- 2. 基本内容 1：在该网络拓扑图中完成如下动态 IP 地址的规划，主要包括：
 - a. 将 PC1、PC2 设置在同一个网段，子网地址是：192.168.0.0/24。
 - b. 将 PC3~PC8 设置在同一个网段，子网地址是：192.168.1.0/24。
 - c. 配置路由器，使得两个子网的各 PC 机之间可以自由通信。
- 3. 基本内容 2：在该网络拓扑中完成如下地址划分，并划分 VLAN，步骤如下：
 - a. 将 PC1、PC2 设置在同一个网段，子网地址是：192.168.0.0/24
 - b. 将 PC3、PC5、PC7 设置在同一个网段，子网地址是：192.168.1.0/24；
 - c. 将 PC4、PC6、PC8 设置在同一个网段，子网地址是：192.168.2.0/24；
 - d. 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；
 - e. 测试各 PC 之间的连通性，并结合所学理论知识进行分析
 - f. 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。

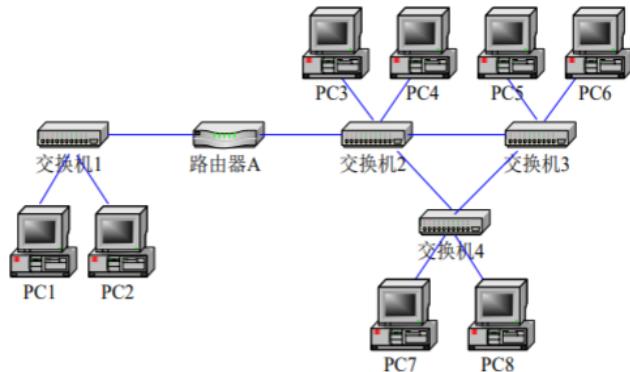


图 3.1 实验 1 网络拓扑图

(2) 网络拓扑图实现

如下图 3.2，在 Cisco Packet Tracer 中实现图 3.1 的网络拓扑：

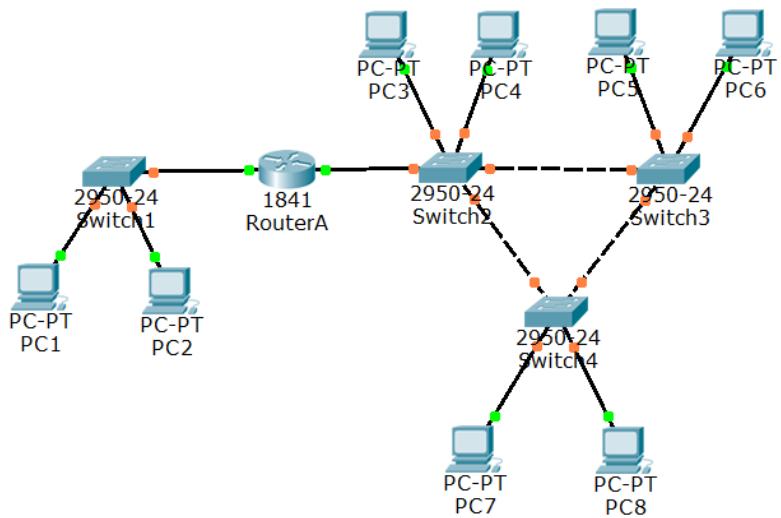


图 3.2 仿真软件上实现网络拓扑图

(3) 基本内容 1 的具体步骤与测试结果

1. 配置路由器 IP

路由器两个端口分别属于两个子网，由于 192.168.0.0 与 192.168.1.0 不可用，则我们左右两个快速以太网端口分别配置 192.168.0.1 和 192.168.1.1，子网掩码全部是 255.255.255.0，截图如下^[1]:

FastEthernet0/0	
Port Status	<input checked="" type="checkbox"/> On
Bandwidth	<input checked="" type="checkbox"/> Auto
<input type="radio"/> 10 Mbps	<input checked="" type="radio"/> 100 Mbps
Duplex	<input checked="" type="checkbox"/> Auto
<input checked="" type="radio"/> Full Duplex	<input type="radio"/> Half Duplex
MAC Address	0001.C71C.1301
IP Address	192.168.0.1
Subnet Mask	255.255.255.0
Tx Ring Limit	10

图 3.3 路由器左接口配置

FastEthernet0/1	
Port Status	<input checked="" type="checkbox"/> On
Bandwidth	<input checked="" type="checkbox"/> Auto
<input type="radio"/> 10 Mbps	<input checked="" type="radio"/> 100 Mbps
Duplex	<input checked="" type="checkbox"/> Auto
<input checked="" type="radio"/> Full Duplex	<input type="radio"/> Half Duplex
MAC Address	0001.C71C.1302
IP Address	192.168.1.1
Subnet Mask	255.255.255.0
Tx Ring Limit	10

图 3.4 路由器右接口配置

2. 配置 PC1、PC2 的 IP

这里 PC1、PC2 属于子网 192.168.0.0/24，其地址在其中任意分配，只要不与路由器左端口冲突即可，这里我们将 PC1 和 PC2 分别设置为 192.168.0.2 和 192.168.0.3，子网掩码全部自动生成为 255.255.255.0，截图如下[让 PC1 去访问其它的主机，访问成功，符合条件，截图如下：

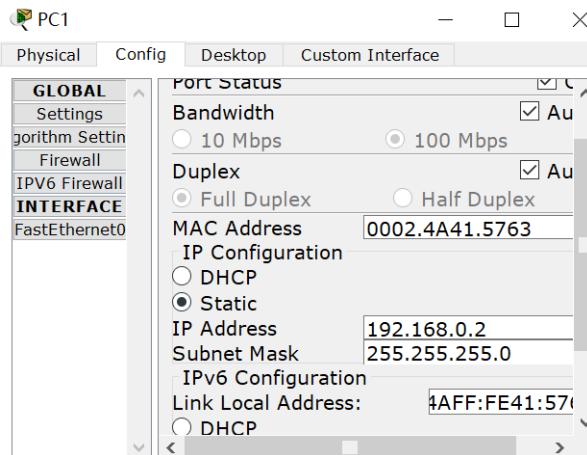


图 3.5 PC1 的 IP 配置

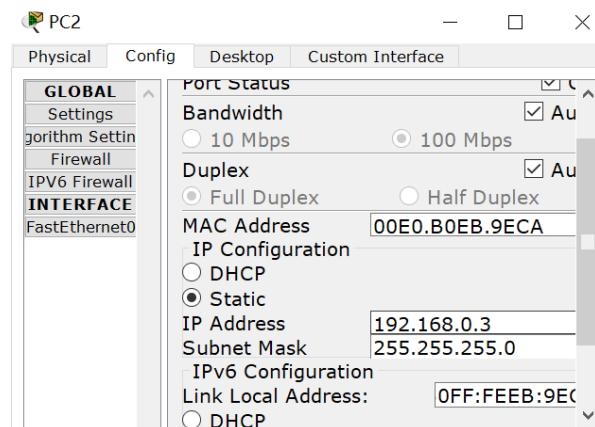


图 3.6 PC2 的 IP 配置

3. 配置 PC3~PC8 的 IP

这几个主机处于 192.168.1.0/24，与上述分配 PC1 和 PC2 的策略一致，我们给这 6 个主机分别分配地址：PC3：192.168.1.2、PC4：192.168.1.3、PC5：192.168.1.4、PC6：192.168.1.5、PC7：192.168.1.6、PC8：192.168.1.7。子网掩码全部是 255.255.255.0^[1]。

4. 主机（PC）的网关配置

PC1、PC2 的网关配置是路由器左接口为 192.168.0.1；PC3~PC8 的网关配置是路由器的右接口为 192.168.1.1。下两图展示 PC1、PC3 的网关配置作为示例^{[1][2]}：

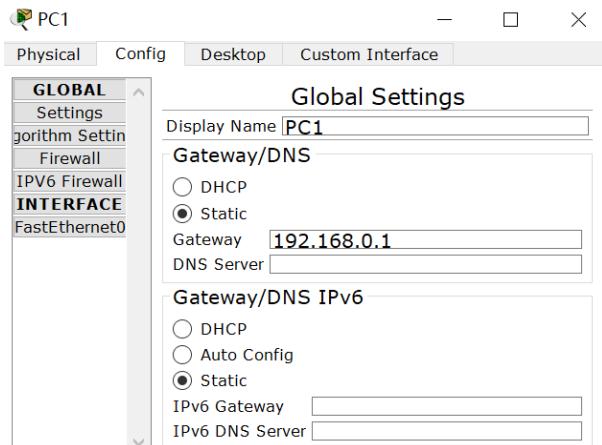


图 3.7 PC1 的网关配置

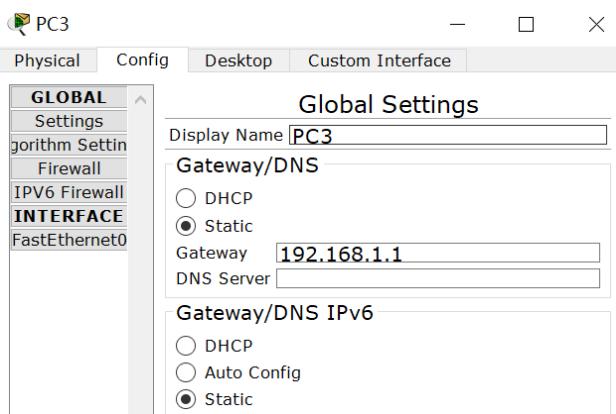


图 3.8 PC3 的网关配置

5. 网络连通测试

这里我们有两种方法测试，可以直接用图形化测试显示结果以及用终端对目的主机进行 ping 操作。例如我们在子网 1 内通信，把 P1 的数据传到 P2。

首先展示图形化操作，点击左边工具栏的第一个小信封（ADD SIMPLE PDU），将其先点一下发送方，再点一下接收方，便模拟了发送方到接收方的传输。这里我们依次点小信封、P1、P2，发现信封从 P1 发送到 P2，底部的 PDU LIST WINDOW 显示了成功信息，截图如下：

PDU List Window											
Fire	Last Status	Source	Destination	Type	Color	Time (sec)	Periodic	Num	Edit	Delete	
●	Successful	PC1	PC2	ICMP	■	0.000	N	0	(edit)	(delete)	

图 3.9 实验 1 网络联通性测试 1

可见 P1 和 P2 可连接

打开 P1 的终端，ping 一下 P2，操作截图如下^{[1][2]}：

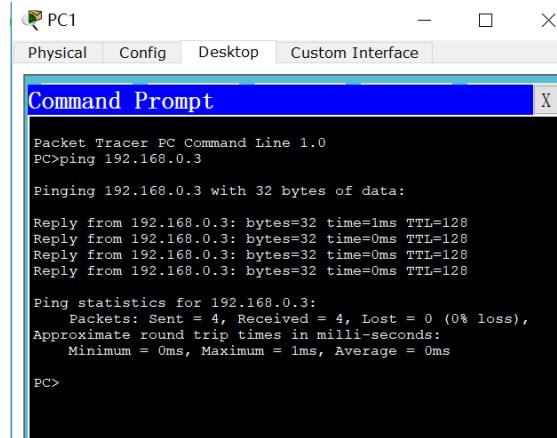


图 3.10 实验 1 网络连通性测试 2

可见，P1 和 P2 是可以连接的

由于采用图形化操作更加简单直观，我们下面的测试都采用图形化操作。

上述已经测试过了子网 1 内可以互通。

现在测试子网 2 内的联通，测试从 P3 传到 P7，截图如下：

PDU List Window										
Fire	Last Status	Source	Destination	Type	Color	Time (sec)	Periodic	Num	Edit	Delete
●	Successful	PC1	PC2	ICMP	■	0.000	N	0	(edit)	(delete)
●	Successful	PC3	PC7	ICMP	■	0.000	N	1	(edit)	(delete)

图 3.11 实验 1 网络连通性测试 2

可见 P3 到 P7 可以连接，子网 2 内联通。

测试子网 1 到子网 2 的连通性，从 P1 传到 P3，截图如下：

PDU List Window										
Fire	Last Status	Source	Destination	Type	Color	Time (sec)	Periodic	Num	Edit	Delete
●	Successful	PC1	PC2	ICMP	■	0.000	N	0	(edit)	(delete)
●	Successful	PC3	PC7	ICMP	■	0.000	N	1	(edit)	(delete)
●	Successful	PC1	PC3	ICMP	■	0.000	N	2	(edit)	(delete)

图 3.12 实验 1 网络连通性测试 4

可见 P1 能传递到 P3，子网 1 到子网 2 联通。

(4) 基本内容 2 的具体步骤与测试结果

在完成了基本内容 1 的基础上，我们完成基本内容 2，基本步骤如下：

1. 将路由器右侧划分为两个网络（192.168.2.0/24 以及 192.168.2.0/24）

将 PC4、PC6、PC8 划分为 192.168.2.0/24 子网，地址划分为 PC4—192.168.2.2、PC6—192.168.2.3、PC8—192.168.2.4，将这四个主机的网关改为 192.168.2.1（后续会分配）。下面两张图展示了 PC4 的地址划分和网关去除^{[1][2]}：

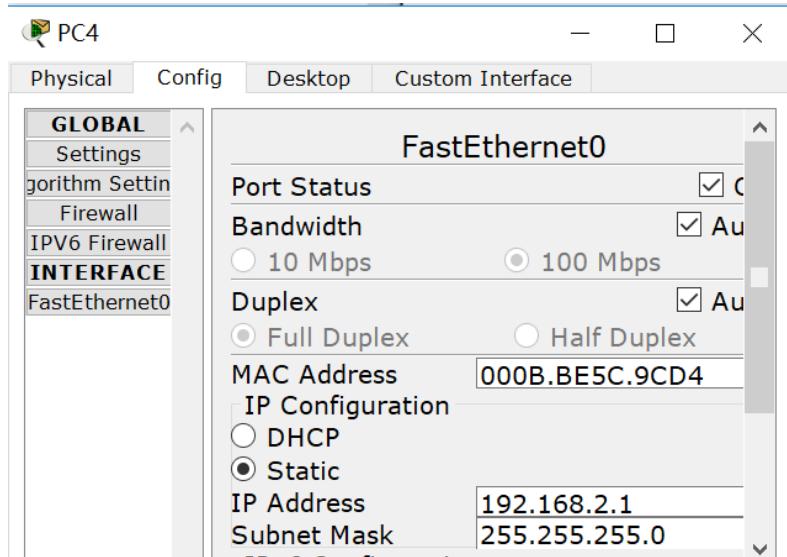


图 3.13 PC4 的设置

2. P3、P5、P7 的 Vlan 划分（划分为 Vlan3）

将右侧交换机 S2、S3、S4 都作以下配置：在 VLAN 配置里加入名字为 Vlan3、编号为 30 的 Vlan，以配置 S2 为例，如下图所示：

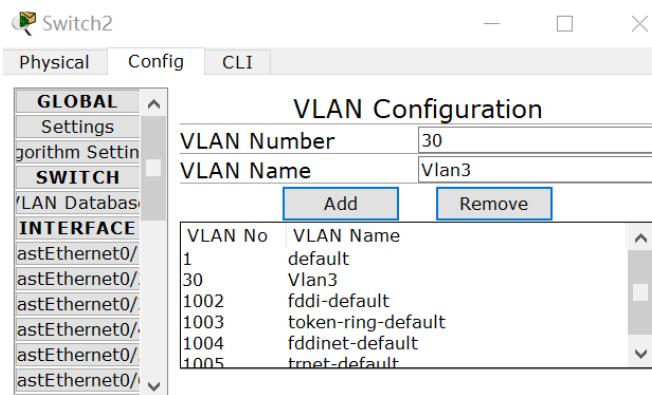


图 3.14 交换机 2 的设置 1

同理对 S3、S4 作出相同配置。

然后我们找到 S2、S3、S4 连接 P3、P5、P7 的端口号（将鼠标悬浮在端口上即可以获得）。比如我发现 S2 和 PC3 的连接端口为 Fa0/3，我们对其进行配置。我们选择类型为 Access，VLAN 号码为 30^[2]，截图如下：

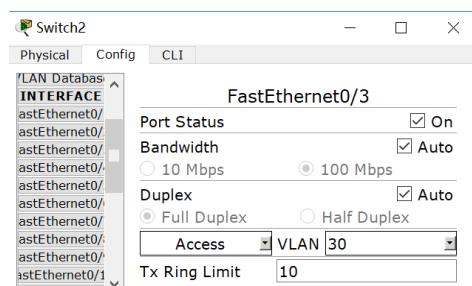


图 3.15 交换机 2 的设置 2

同理对 S3 连接 PC5、S4 连接 PC7 进行相应配置。

3. P4、P6、P8 的 Vlan 划分（划分为 Vlan4）

这里我们选择 Vlan 名字为 Vlan4，编号为 40。分别对 S2、S3、S4 进行相应表项的添加，找到 S2 和 PC4 连接、S3 和 PC6 连接、S4 和 PC8 连接的端口，进行相应配置，具体步骤和配置 P3、P5、P7 一致。

4. S2、S3、S4 的链路连接以及 S2 和路由器的连接

分别找到 S2 和 S3、S3 和 S4、S4 和 S2、S2 和路由器的连接端口两端，全部设置为 trunk 型的端口，vlan 号使用默认的即可（因为其处于主干电路）。例如我们对 S2 中，S2 和 S3 连接的端口 Fa0/2 的配置如下^{[1][2]}：

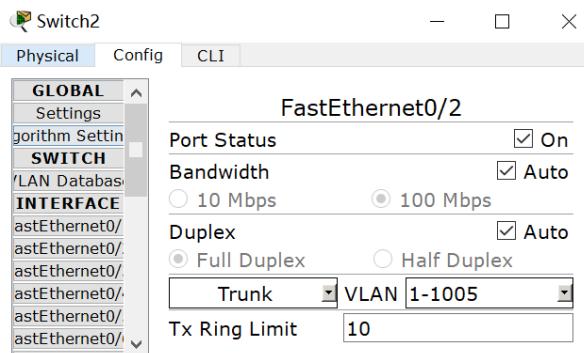


图 3.16 交换机 2 的配置 3

5. PC1、PC2 设置（划分为 Vlan2）和 S1 的设置（和路由器连接）^[1]

题目要求我们将此区域也划分成 Vlan，按照以上步骤，我们要做的有：

- 在 Vlan 表项中添加名字为 Vlan2，编号为 20。
- 找到交换机 S1 和 PC1、PC2 连接的端口，设置为端口的 Vlan 号码为 20。
- 将交换机和路由器相连的端口设置为 trunk 型。

配置过程与前面步骤基本一致，这里不再详细叙述。

6. 路由器的 IP 设置

我们发现路由器右边的端口只支持一个子网的网关，这里我们给以太网接口创建两个子接口 fa0/1.1 和 fa0/1.2，用来支持两个网关。下面给出配置 fa0/1.1 的过程如下^{[1][2]}：

```
Router(config)#int fa0/1.2
Router(config-subif)#encap dot1q 40
Router(config-subif)#ip addr 192.168.2.1 255.255.255.0
Router(config-subif)#exit
Router(config)#int fa0/1.1
Router(config-subif)#encap dot1q 30
Router(config-subif)#ip addr 192.168.1.1 255.255.255.0
Router(config-subif)#exit
Router(config)#int fa0/1.2
Router(config-subif)#encap dot1q 40
Router(config-subif)#ip addr 192.168.2.1 255.255.255.0
Router(config-subif)#exit
```

图 3.17 创建子接口截图

这之后，我们将 PC1、PC2 的网关设置为 192.168.0.1，PC3、PC5、PC7 的网关设置为 192.168.1.1，PC2、PC4、PC6 的网关设置为 192.168.2.1，整个网络即可以正常访问了。

7. 测试结果及分析

首先我们测试 vlan2、vlan3、vlan4 三个 vlan 的内部是否能够互相通信。

我们采用图形化界面进行测试，信息分别从 P1 发给 P2、P3 发给 P5、P6 发给 P8，测试结果如下：

PDU List Window											
Fire	Last Status	Source	Destination	Type	Color	Time (sec)	Periodic	Num	Edit	Delete	
●	Successful	PC1	PC2	ICMP	orange	0.000	N	0	(edit)	(delete)	
●	Successful	PC3	PC5	ICMP	grey	0.000	N	1	(edit)	(delete)	
●	Successful	PC6	PC8	ICMP	yellow	0.000	N	2	(edit)	(delete)	

图 3.18 实验 1 网络连通性测试 5

可以发现信息全部发送成功，测试通过。

我们测试 vlan 之间是否能够互相通信，我们测试以下三组数据：

从 PC3 发送到 PC4（从 vlan3 发送到 vlan4，不跨路由器）、PC8 发送到 PC5（从 vlan4 发送到 vlan3，不跨路由器）、PC1 发送到 PC3（从 vlan2 发送到 vlan3，跨路由器），测试结果如下：

PDU List Window											
Fire	Last Status	Source	Destination	Type	Color	Time (sec)	Periodic	Num	Edit	Delete	
●	Successful	PC3	PC4	ICMP	green	0.000	N	0	(edit)	(delete)	
●	Successful	PC8	PC5	ICMP	magenta	0.000	N	1	(edit)	(delete)	
●	Successful	PC1	PC3	ICMP	magenta	0.000	N	2	(edit)	(delete)	

图 3.19 实验 1 网络连通性测试 6

可以发现信息全部发送成功，测试通过。

这个实验中我们将路由器右侧的一个子网划分成为两个 vlan，成为两个逻辑上不受子网和路由器限制的主体，我们通过配置交换机在子网内划分出了这样两个逻辑主体，配置路由器的端口使得两个逻辑主体都能够配置网关，这样使得右侧两个 vlan 和左侧一个 vlan 能够互联互通。

3.3.2 路由配置实验的步骤及结果分析

(1) 实验 2 简介

1. 要求使用仿真软件，按照图 3.20 画出网络拓扑图：

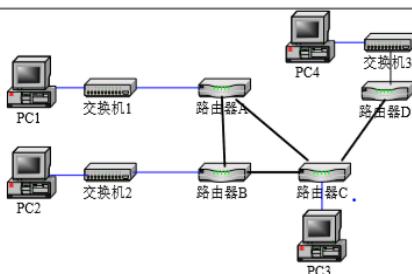


图 3.20 实验 2 的网络拓扑结构

2. 基本内容 1：配置 RIP 协议

- a. 将 PC1 设置在 192.168.1.0/24 网段。
- b. 将 PC2 设置在 192.168.2.0/24 网段。
- c. 将 PC3 设置在 192.168.3.0/24 网段。
- d. 将 PC4 设置在 192.168.4.0/24 网段。
- e. 设置路由器端口。
- f. 在路由器上配置 RIP 协议，使得上述拓扑图中的 PC 机能够互相访问。

3. 基本内容 2：配置 OSPF 协议

- a. PC1、PC2、PC3、PC4 的配置和基本内容 1 一致，路由器端口设置和基本内 1 相同
- b. 再录预期上配置 OSPF 协议，使得上述拓扑图中的 PC 机能够互相访问。

4. 基本内容 3：指定机器无法互访

- a. 在内容 1 或 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，也无法被其它 PC 访问。
- b. 在基本内容 1 或 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，但是可以访问其它 PC 机。

(2) 网络拓扑图的实现

如下图 3.2， 在 Cisco Packet Tracer 中实现图 3.21 的网络拓扑：

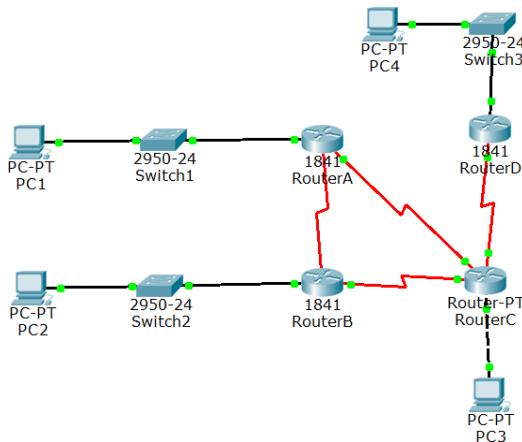


图 3.21 Cisco Packet Tracer 上实现的网络拓扑图

(3) 基本内容 1 的具体步骤和测试结果

1. 增加路由器的接口以便路由器和其它路由器互联

在 Cisco Packet Tracer 中，路由器都默认有两个以太网接口（即 FastEthernet），但是根据实验要求来看，有的路由器不止连了两个设备，这个时候我们需要添加路由器的接口实现互联^[2]。

最简单的一种方式就是添加串口 (serial)，步骤如下：

- 1) 在路由器的 physical 模块中找到 HWIC-2T 模块
- 2) 关闭路由器电源
- 3) 将该模块拖入到右侧小黑框中
- 4) 打开电源，启动路由器所有的端口

示意图如下图所示：

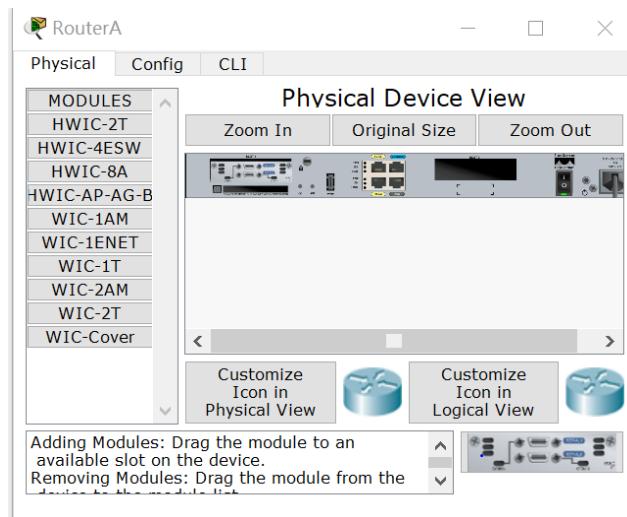


图 3.22 增加路由器的接口示意图

完成了这个步骤之后，我们才能画出像上述图 3.21 所示的网络拓扑图。

2. 配置主机以及路由器的 IP、主机的网关以及子网掩码

这个步骤与上一个实验十分类似，这里不再截图解释，按照题目要求的子网划分，给出划分结果如下^{[1][2]}：

- 1) PC1：其 IP 为 192.168.1.2；其网关为 192.168.1.1；子网掩码 255.255.255.0
- 2) PC2：其 IP 为 192.168.2.2；其网关为 192.168.2.1；子网掩码 255.255.255.0
- 3) PC3：其 IP 为 192.168.3.2；其网关为 192.168.3.1；子网掩码 255.255.255.0
- 4) PC4：其 IP 为 192.168.4.2；其网关为 192.168.4.1；子网掩码 255.255.255.0
- 5) RouterA：与 PC1 连接的 IP：192.168.1.1；与 RouterB 连接的 IP：192.168.5.1；与 RouterC 连接的 IP：192.168.7.1；子网掩码全部都是 255.255.255.0
- 6) RouterB：与 PC2 连接的 IP：192.168.2.1；与 RouterA 连接的 IP：192.168.5.2；与 RouterC 连接的 IP：192.168.6.1；子网掩码全部都是 255.255.255.0
- 7) RouterC：与 PC3 连接的 IP：192.168.3.1；与 RouterB 连接的 IP：192.168.6.2；与 RouterA 连接的 IP：192.168.7.2，与 RouterD 连接的 IP：192.168.8.2；子网掩码全部都是 255.255.255.0
- 8) RouterD：与 PC4 连接的 IP：192.168.4.1；与 RouterC 连接的 IP：192.168.8.1；子网掩码全部都是 255.255.255.0

3.RIP 协议的配置

这一步是这个实验的关键，我们要在路由器中配置 RIP 协议，告诉路由器可以访问的网段是什么。

我们以 RouterA 为例，RouterA 连接的 PC 或者路由器有 PC1、RouterB 的一个端口、RouterC 的一个端口，所以我们要在 RIP 协议中添加这些 PC 或者路由器端口的网段。而 PC1、RouterB、RouterC 对应的端口的网段分别为：192.168.1.0、192.168.5.0、192.168.7.0，所以添加这些网段如下图所示：

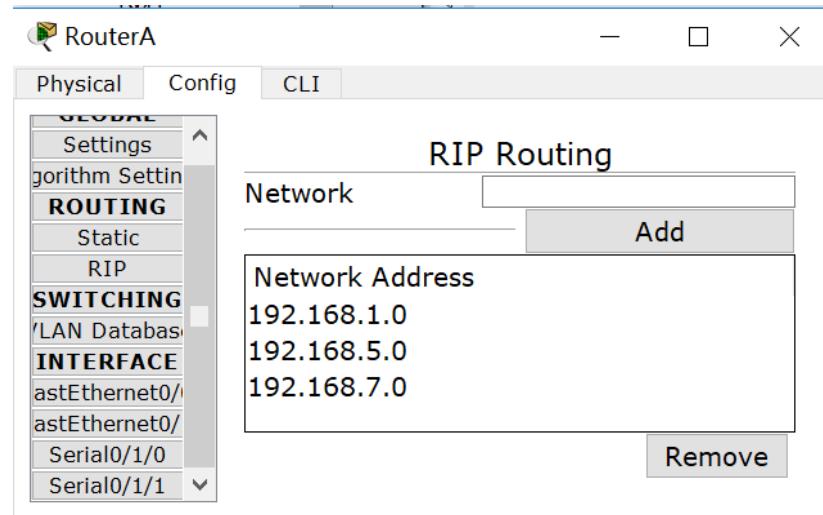


图 3.23 RIP 协议配置示意图

同理，现在我们还要添加 RouterB、RouterC 和 RouterD 对应的网段，对应如下^[1]：

- 1) RouterB: 192.168.2.0; 192.168.5.0; 192.168.6.0
- 2) RouterC: 192.168.3.0; 192.168.6.0; 192.168.7.0; 192.168.8.0
- 3) RouterD: 192.168.4.0; 192.168.8.0

4.连通性测试

我们要保证任意两台主机都能通信，我们设置了下面几次通信方式：PC1 到 PC2；PC2 到 PC1；PC3 到 PC4；PC2 到 PC4

通信结果如下图所示：

PDU List Window											
Fire	Last Status	Source	Destination	Type	Color	Time (sec)	Periodic	Num	Edit	Delete	
●	Failed	PC1	PC2	ICMP	■	0.000	N	0	(edit)	(delete)	
●	Successful	PC1	PC2	ICMP	■	0.000	N	1	(edit)	(delete)	
●	Successful	PC2	PC1	ICMP	■	0.000.	N	2	(edit)	(delete)	
●	Failed	PC3	PC4	ICMP	■	0.000	N	3	(edit)	(delete)	
●	Successful	PC3	PC4	ICMP	■	0.000	N	4	(edit)	(delete)	
●	Successful	PC2	PC4	ICMP	■	0.000	N	5	(edit)	(delete)	

图 3.24 实验 2 连通性测试 1

由上图可知，我们的主机之间的通信全部成功。

这里有一个很有趣的现象，就是每台机器在第一次 ping 的时候，会失败，但是后

来再 ping 的时候就成功了，一开始我以为是软件的 bug，后来经过和同学讨论知道，其实这是因为，第一次通信的时候，其链路上的交换机之前还没有学习到其 IP 地址，通过一次传输，会自学习到其 IP 地址，所以下一个便能正常通信。

(4) 基本内容 2 的具体步骤和测试结果

1.各网络原件的 IP、网关与子网掩码配置

这个配置过程与基本内容 1 一模一样，详细请参考上述基本内容 1。

2.OSPF 协议的配置

其思路和 RIP 协议的配置一致，都是要在目标路由器上指明其可以到达的网络，具体配置过程中有所不同，OSPF 必须用终端代码进行配置（而 RIP 可以直接用图形化界面进行配置）。对于 4 个路由器需要配置的网段，列出如下^{[1][2]}：

- 1) RouterA: 192.168.1.0、192.168.5.0、192.168.7.0
- 2) RouterB: 192.168.2.0; 192.168.5.0; 192.168.6.0
- 3) RouterC: 192.168.3.0; 192.168.6.0; 192.168.7.0; 192.168.8.0
- 4) RouterD: 192.168.4.0; 192.168.8.0

我们还是用 RouterA 的配置举例子，讲述用终端配置 OSPF 协议的方法，截图如下：

```
Router>enable
Router#configure terminal
Enter configuration commands, one per line. End with CNTL/Z.
Router(config)#router ospf 1
Router(config-router)#network 192.168.1.0 0.0.0.255 area 0
Router(config-router)#network 192.168.5.0 0.0.0.255 area 0
Router(config-router)#network 192.168.6.0 0.0.0.255 area 0
Router(config-router)#end
Router#
%SYS-5-CONFIG_I: Configured from console by console

Router#copy run startup
Destination filename [startup-config]?
Building configuration...
[OK]
Router#
```

图 3.25 路由器 A 的 OSPF 的配置

对于上述代码解释如下：

- 1) 首先指定一个 ospf 的进程号，这里指定为 1（注意同一时刻进程号不可重复）
- 2) 用 network 命令，输入可通信的网段号和子网掩码的反码
- 3) 运行这个配置

用同样的方法，按照上述给出的可通信网段，可以配置 RouterB、RouterC、RouterD。

3. 连通性测试

我们要保证任意两台主机都能通信，我们设置了下面几次通信方式：PC3 到 P C4；PC4 到 PC3；PC1 到 PC4；PC2 到 PC3 的通信结果如下图所示：

PDU List Window											
Fire	Last Status	Source	Destination	Type	Color	Time (sec)	Periodic	Num	Edit	Delete	
●	Failed	PC3	PC4	ICMP	■	0.000	N	0	(edit)	(delete)	
●	Successful	PC3	PC4	ICMP	■	0.000	N	1	(edit)	(delete)	
●	Successful	PC4	PC3	ICMP	■	0.000	N	2	(edit)	(delete)	
●	Successful	PC1	PC4	ICMP	■	0.000	N	3	(edit)	(delete)	
●	Successful	PC2	PC3	ICMP	■	0.000	N	4	(edit)	(delete)	

图 3.36 实验 2 连通性测试 2

由上图可以看出，四组通信全部可以成功。（关于第一组失败的问题，是因为第一次交换机会自学习，在基本内容 1 中已经有论述，不再展开）

(5) 基本内容 3 的具体步骤和测试结果

1.第一次配置 RouterA

要求使得 PC1 既不能访问其它主机，也不可以被其它主机访问。

要做到这一点，我们要屏蔽 PC1 的通信（即屏蔽 192.168.0.1 网段），我们想到了可以用 deny 指令完成这一点，完整的代码如下图所示：

```
Router(config)#access-list 10 deny 192.168.1.0 0.0.0.255
Router(config)#access-list 10 permit any
Router(config)#int fa0/1
Router(config-if)#ip access-group 10 in
Router(config-if)#exit
Router(config) #
```

图 3.37 路由器 A 的配置

可以看出步骤如下：

- 1) 首先指定一个数值，代表这一组命令，这里取 10。
- 2) 用 deny 命令屏蔽掉 PC1 的网段，然后输入子网掩码的反码。
- 3) 用 permit 命令允许其他网段的通信^[2]。
- 4) 打开 PC 网段的网关接口，用 in 命令将权限制令写入接口。

2.连通性测试（对于 PC1 不能访问其它 PC 也不能被访问的情况）

我们首先用 PC1 尝试给其他终端进行通信，结果是失败的，如下图所示：

PDU List Window											
Fire	Last Status	Source	Destination	Type	Color	Time (sec)	Periodic	Num	Edit	Delete	
●	Failed	PC1	PC2	ICMP	■	0.000	N	0	(edit)	(delete)	
●	Failed	PC1	PC2	ICMP	■	0.000	N	1	(edit)	(delete)	
●	Failed	PC1	PC4	ICMP	■	0.000	N	2	(edit)	(delete)	
●	Failed	PC1	PC3	ICMP	■	0.000	N	3	(edit)	(delete)	
●	Failed	PC1	PC3	ICMP	■	0.000	N	4	(edit)	(delete)	
●	Failed	PC1	PC4	ICMP	■	0.000	N	5	(edit)	(delete)	

图 3.38 实验 2 连通性测试 3

可以看出没有一次通信成功，符合题目要求。

然后我们用其他终端对 PC1 进行通信，结果仍然是失败的，如下图所示：

PDU List Window											
Fire	Last Status	Source	Destination	Type	Color	Time (sec)	Periodic	Num	Edit	Delete	
●	Failed	PC4	PC1	ICMP	■	0.000	N	0	(edit)	(delete)	
●	Failed	PC4	PC1	ICMP	■	0.000	N	1	(edit)	(delete)	
●	Failed	PC2	PC1	ICMP	■	0.000	N	2	(edit)	(delete)	
●	Failed	PC2	PC1	ICMP	■	0.000	N	3	(edit)	(delete)	
●	Failed	PC3	PC1	ICMP	■	0.000	N	4	(edit)	(delete)	
●	Failed	PC3	PC1	ICMP	■	0.000	N	5	(edit)	(delete)	

图 3.39 实验 2 连通性测试 4

可以看出没有一次通信成功，符合题目要求。

然后我们再在除了 PC1 之外的任何主机之间通信，每个主机间都可以互相访问，如下图所示：

PDU List Window											
Fire	Last Status	Source	Destination	Type	Color	Time (sec)	Periodic	Num	Edit	Delete	
●	Successful	PC3	PC2	ICMP	■	0.000	N	0	(edit)	(delete)	
●	Successful	PC3	PC4	ICMP	■	0.000	N	1	(edit)	(delete)	
●	Successful	PC2	PC3	ICMP	■	0.000	N	2	(edit)	(delete)	
●	Successful	PC2	PC4	ICMP	■	0.000	N	3	(edit)	(delete)	
●	Successful	PC4	PC2	ICMP	■	0.000	N	4	(edit)	(delete)	
●	Successful	PC4	PC3	ICMP	■	0.000	N	5	(edit)	(delete)	

图 3.40 实验 2 连通性测试 5

可以看出通信均取得成功，符合题目要求。

综上所述，该网络符合设计条件，配置正确。

3. 第二次配置 RouterA

使得 PC1 不能访问 PC2，但是可以访问其它 PC 机

这个就是在上述题目改动一点即可，我们将 deny 的网段换成 PC2 的网段（192.168.2.0），然后再用 out 指令输入接口即可^[2]，配置的代码如下图所示：

```
Router(config)#access-list 20 deny 192.168.2.0 0.0.0.255
Router(config)#int fa0/1
Router(config-if)#ip access-group 20 in
Router(config-if)#exit
Router(config) #
```

图 3.41 配置路由器 A

4. 连通性测试 2（对于 PC1 只不能访问 PC2 的情况）

首先让 PC1 多访问几次 PC2，发现都不能访问，符合条件，截图如下：

PDU List Window											
Fire	Last Status	Source	Destination	Type	Color	Time (sec)	Periodic	Num	Edit	Delete	
●	Failed	PC1	PC2	ICMP	■	0.000	N	0	(edit)	(delete)	
●	Failed	PC1	PC2	ICMP	■	0.000	N	1	(edit)	(delete)	
●	Failed	PC1	PC2	ICMP	■	0.000	N	2	(edit)	(delete)	
●	Failed	PC1	PC2	ICMP	■	0.000	N	3	(edit)	(delete)	

图 3.42 实验 2 连通性测试 6

让 PC1 去访问其它的主机，访问成功，符合条件，截图如下

PDU List Window											
Fire	Last Status	Source	Destination	Type	Color	Time (sec)	Periodic	Num	Edit	Delete	
●	Successful	PC1	PC3	ICMP	■	0.000	N	0	(edit)	(delete)	
●	Successful	PC1	PC3	ICMP	■	0.000	N	1	(edit)	(delete)	
●	Successful	PC1	PC4	ICMP	■	0.000	N	2	(edit)	(delete)	
●	Successful	PC1	PC4	ICMP	■	0.000	N	3	(edit)	(delete)	

图 3.43 实验 2 连通性测试 7

其它主机之间互相访问，都能成功，符合条件，截图如下：

PDU List Window											
Fire	Last Status	Source	Destination	Type	Color	Time (sec)	Periodic	Num	Edit	Delete	
●	Successful	PC2	PC3	ICMP	■	0.000	N	0	(edit)	(delete)	
●	Successful	PC2	PC4	ICMP	■	0.000	N	1	(edit)	(delete)	
●	Successful	PC3	PC2	ICMP	■	0.000	N	2	(edit)	(delete)	
●	Successful	PC3	PC4	ICMP	■	0.000	N	3	(edit)	(delete)	
●	Successful	PC4	PC2	ICMP	■	0.000	N	4	(edit)	(delete)	
●	Successful	PC4	PC3	ICMP	■	0.000	N	5	(edit)	(delete)	

图 3.44 实验 2 连通性测试 8

综上所述，该网络符合设计条件，配置正确。

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

(1) 任务书中的设计要求

1. 设计背景

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。

2. 组网需求

- a. 图书馆能够无线上网（即需要无线路由器或接入节点 AP）
- b. 学院之间、学生宿舍之间可以互相访问
- c. 学院和学生宿舍之间不能互相访问
- d. 学院和学生宿舍都可以访问图书馆

(2) 设计过程

1. 拓扑结构的设计

- 1) 整个网络中存在三个路由器：宿舍路由器(R_Dormitory)、图书馆路由器(R_Library)、学院路由器 (R_School)。

-
- 2) 宿舍子网中存在三台交换机，互相连接形成一个三角形。三角形的一个顶点连接宿舍路由器。三台交换机的三个顶点再分别连接主机，配置成三个 Vlan，代表三个宿舍区。（使用 Vlan 可以使得以后要更改设计要求，特别是更改宿舍间的设计要求，更加简便。本题可以不设置 Vlan，但是设置 Vlan 可以使得整个结构更加灵活和易于更改，所以这里还是设计了 Vlan）
 - 3) 图书馆子网中存在一台交换机和一个无线路由器 Wireless Router。其中，交换机连接图书馆路由器、所有有线主机以及无线路由器，而无线路由器连接无线设备。
 - 4) 学院子网中有四台交换机，其中一台交换机为主要交换机，连接学院路由器和其他三个路由器。四个路由器分别再连接主机，配置成四个 Vlan，代表四个学院。（使用 Vlan 的好处和第 2) 点一致）
 - 5) 拓扑结构图见下一节实验步骤中的图 3.45。

2. 网络地址的划分

已知我们现在可以分配使用的网络块为 211.69.4.0/22。每个学院 20 台主机，给其分配 32 大小的地址块即可，一共 4 个学院，分配 4 个大小为 32 的地址块。图书馆有 100 台主机，分配一个大小为 128 的地址块即可。每个学生宿舍有 200 台主机，分配大小为 256 的地址块即可，有三个宿舍，即分配 3 个 256 大小的地址块。这样一共分配了 3 个 256 大小的地址块，一个 128 大小的地址块，4 个大小为 32 的地址块，刚好把 211.64.4.0/22 全部分配出去。具体的我们有以下策略^{[1][2]}:

- 1) 宿舍 1: 分配地址块 211.69.4.0/24，网关地址 211.69.4.1
- 2) 宿舍 2: 分配地址块 211.69.5.0/24，网关地址 211.69.5.1
- 3) 宿舍 3: 分配地址块 211.69.6.0/24，网关地址 211.69.6.1
- 4) 图书馆: 分配地址块 211.69.7.0/25，网关地址 211.69.7.1
- 5) 学院 1: 分配地址块 211.69.7.128/27，网关地址 211.69.7.129
- 6) 学院 2: 分配地址块 211.69.7.160/27，网关地址 211.69.7.161
- 7) 学院 3: 分配地址块 211.69.7.192/27，网关地址 211.69.7.193
- 8) 学院 4: 分配地址块 211.69.7.224/27，网关地址 211.69.7.225

3. 学院、宿舍划分 Vlan 的设计

- 1) 宿舍的 Vlan 号：三个宿舍的 Vlan 号分别为：110、120、130，名字分别为 dor1、dor2、dor3^[2]。
- 2) 学院的 Vlan 号：四个学院的 Vlan 号分别为 10、20、30、40，名字分别为 school1、school2、school3、shcool4^[2]。

4. 路由器的地址设计

我们要实现路由器之间的互联，要设置路由器端口地址。而根据我们现在对网络地址的划分，所有地址已经划分完毕给每一个模块了。经过多方协商结果，我们采用192.168.xxx.xxx的私有网络格式配置路由器的端口地址。

我们将宿舍路由器（R_Dormitory）和图书馆路由器（R_library）之间划分子网192.168.1.0/24；宿舍路由器（R_Dormitory）和学校路由器（R_School）之间划分子网192.168.3.0/24；图书馆路由器（R_library）和学校路由器（R_School）之间划分子网192.168.2.0/24。具体端口如下所示：

- 1) 宿舍路由器(和宿舍连接)的端口地址: 分别为3个Vlan的网关地址——211.69.4.1、211.69.5.1、211.69.6.1。
 - 2) 宿舍路由器(和图书馆路由器连接)的端口地址: 192.168.1.2
 - 3) 宿舍路由器(和学院路由器连接)的端口地址: 192.168.3.1
 - 4) 学院路由器(和学院连接)的端口地址: 分别为4个Vlan的网关地址: 211.69.7.129、211.69.7.161、211.69.7.193、211.69.7.225
 - 5) 学院路由器(和宿舍路由器相连)的端口地址: 192.168.3.2
 - 6) 学院路由器(和图书馆路由器相连)的端口地址: 192.168.2.2
 - 7) 图书馆路由器(和图书馆相连)的端口地址: 211.69.7.129
 - 8) 图书馆路由器(和宿舍路由器相连)的端口地址: 192.168.1.1
 - 9) 图书馆路由器(和学院路由器相连)的端口地址: 192.168.2.1
5. 访问控制的设计
- 1) 用OSPF协议配置每一个路由器，使得整个网络可以互相访问。
 - 2) 配置宿舍路由器，使得其屏蔽来自学院网段发来的消息。
 - 3) 配置学院路由器，使得其屏蔽来自宿舍网段发来的消息

3.4.2 实验步骤

(1) 根据上述设计，画出网络拓扑图

设计的网络拓扑图如下：

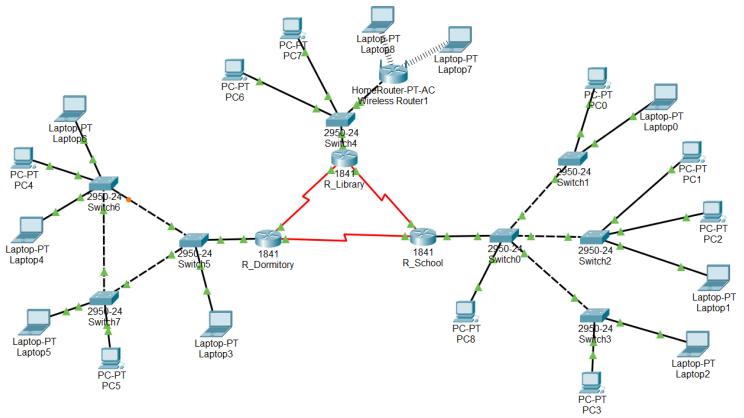


图 3.45 自主设计的网络拓扑图

(2) 配置路由器 IP 和各个主机的 IP

根据上一节实验设计中“网络地址设计”、“路由器的地址设计”两个部分的内容，给主机和路由器的每个端口配置IP、网关以及子网掩码，相关配置方式在实验1中已经有详细叙述，这里不再一一说明。

要注意的是，每个主机的网络的地址应该从网关地址的后一个地址开始配置，切不可配置成网关地址。

(3) 路由器协议配置

这里我们选择了 OSPF 协议，具体的代码实现在实验 2 中有详细介绍，这里不再过多叙述，这里列出每个路由器要选择连接哪几个网络^{[1][2]}：

1. R_Dormitory : 211.69.4.0/24 , 211.69.5.0/24 , 211.69.6.0/24 , 192.168.1.0/24 , 192.168.3.0/24
 2. R_Library: 211.69.7.0/25 , 192.168.1.0/24 , 192.168.2.0/24
 3. R_School: 211.69.7.128/27 , 211.69.7.160/27 , 211.69.7.192/27 , 211.69.7.224/27 , 192.168.2.0/24 , 192.168.3.0/24.

(4) 宿舍、学院 Vlan 的划分

Vlan 的划分方式和实验 1 中的 Vlan 的划分方法完全一样。步骤如下：

1. 选中要配置的交换机，在 Vlan Database 中添加我们要加入的 Vlan 号和名称。
 2. 所有交换机和主机相连的端口全部设置为 access 型线路，Vlan 号为其要划分的 Vlan 号码。
 3. 所有交换机之间相连的端口全部设置为 trunk 型线路。

按照以上方式，参照上一节设计的 Vlan 号以及其划分设计（详见上一节“学院、宿舍划分 Vlan 的设计”），遵循以上步骤，将 Vlan 配置成功。

(5) 设置图书馆无线网络

选用 HomeRouter-PT-AC Wireless Router1 这个型号的无线路由器，将其类型选择为 Wireless AP，如下图所示：

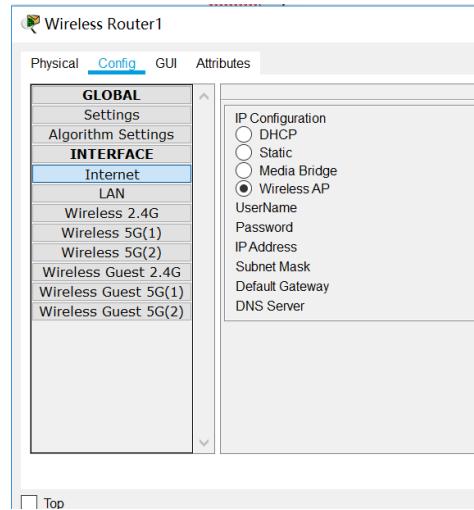


图 3.46 无线网的设置 1

同时我们要给无线主机装上网卡，即选择“Linksys-WPC300N”，截图如下图所示：

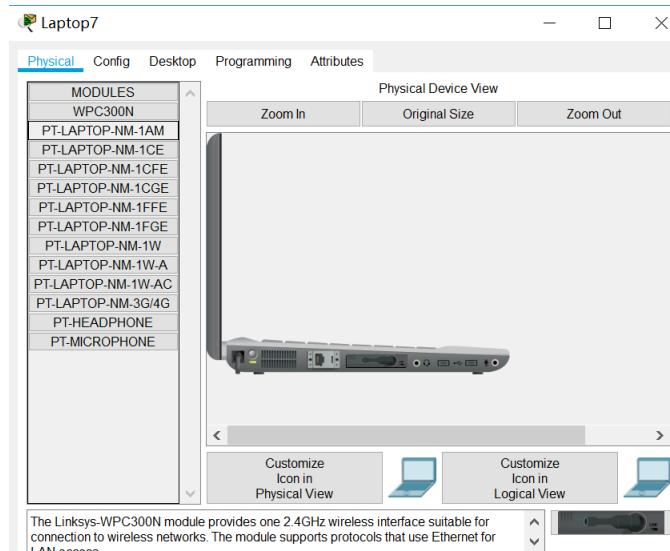


图 3.47 无线网的设置 2

这样我们便配置好了无线网络。

(6) 设置控制协议

1. 对于宿舍路由器 R_Dormitory：屏蔽网段 211.69.7.128/27、211.69.7.160/27、211.69.7.192/27、211.69.7.224/27^[2]。
2. 对于学院路由器 R_School：屏蔽网段 211.69.4.0/24、211.69.5.0/24、211.69.6.0/24^[2]。

3.4.3 结果分析

(1) 学院、图书馆、宿舍内部均随意访问测试

1. 学院内部随意访问测试

我们在学院模块的几个 Vlan 内部和 Vlan 之间进行测试。按照上述的划分，PC8 属于一个

Vlan、PC3 和 Laptop2 属于一个 Vlan、PC2, PC1 和 Laptop1 属于一个 Vlan、PC0 和 Laptop0 属于一个 Vlan, 这四个 Vlan 代表四个学院。如下图, 我们随意在这几个 Vlan 内部和 Vlan 之间进行通信:

PDU List Window											
Fire	Last Status	Source	Destination	Type	Color	Time(sec)	Periodic	Num	Edit	Delete	
Successful	PC8	PC3	ICMP	■	0.000	N	0	(edit)		(delete)	
Successful	PC8	PC1	ICMP	■	0.000	N	1	(edit)		(delete)	
Successful	PC8	PC0	ICMP	■	0.000	N	2	(edit)		(delete)	
Successful	PC1	PC3	ICMP	■	0.000	N	3	(edit)		(delete)	
Successful	PC2	Laptop2	ICMP	■	0.000	N	4	(edit)		(delete)	
Successful	Laptop2	PC3	ICMP	■	0.000	N	5	(edit)		(delete)	
Successful	Laptop2	PC3	ICMP	■	0.000	N	6	(edit)		(delete)	
Successful	Laptop2	PC1	ICMP	■	0.000	N	7	(edit)		(delete)	
Successful	PC0	PC8	ICMP	■	0.000	N	8	(edit)		(delete)	
Successful	PC2	PC8	ICMP	■	0.000	N	9	(edit)		(delete)	
Successful	PC2	PC3	ICMP	■	0.000	N	10	(edit)		(delete)	

、图 3.47 实验 3 联通性测试 1

根据图中的结果显示, 学院模块任意两台主机都可以正常通信, 符合要求, 网络设计以及实现正确。

2.图书馆内部随意访问测试

我们在图书馆模块的内部随意选择两台主机测试, 包括无线网之间的通信、有线网之间的通信、有线网和无线网之间的通信, 截图如下所示:

PDU List Window											
Fire	Last Status	Source	Destination	Type	Color	Time(sec)	Periodic	Num	Edit	Delete	
Successful	PC6	PC7	ICMP	■	0.000	N	0	(edit)		(delete)	
Successful	Laptop7	Laptop7	ICMP	■	0.000	N	1	(edit)		(delete)	
Successful	PC6	Laptop7	ICMP	■	0.000	N	2	(edit)		(delete)	
Successful	Laptop7	PC7	ICMP	■	0.000	N	3	(edit)		(delete)	

图 3.48 实验 3 联通性测试 2

根据截图中的显示, 图书馆模块任意两台主机都可以正常通信, 符合要求, 网络设计以及实现正确。

3.宿舍内部随意访问测试

我们在宿舍模块的几个 Vlan 内部和 Vlan 之间进行测试。按照上述划分, Laptop3 属于一个 Vlan、PC5 和 Laptop5 属于一个 Vlan、Laptop6, Laptop4 和 PC4 属于一个 Vlan。如下图, 我们随意在这几个 Vlan 内部和 Vlan 间进行通信。

PDU List Window											
Fire	Last Status	Source	Destination	Type	Color	Time(sec)	Periodic	Num	Edit	Delete	
Successful	Laptop5	Laptop5	ICMP	■	0.000	N	0	(edit)		(delete)	
Successful	Laptop5	PC5	ICMP	■	0.000	N	1	(edit)		(delete)	
Successful	Laptop5	Laptop3	ICMP	■	0.000	N	2	(edit)		(delete)	
Successful	PC4	Laptop5	ICMP	■	0.000	N	3	(edit)		(delete)	
Successful	Laptop5	PC5	ICMP	■	0.000	N	4	(edit)		(delete)	
Successful	PC4	PC5	ICMP	■	0.000	N	5	(edit)		(delete)	
Successful	PC4	Laptop3	ICMP	■	0.000	N	6	(edit)		(delete)	

图 3.49 试验 3 联通性测试 3

根据图中的结果显示, 宿舍间任意两台主机都可以正常通信, 符合要求, 网络设计以及实现正确。

(2) 学院、图书馆、宿舍之间互相访问

1. 图书馆和宿舍之间互相访问

根据题目要求，图书馆和学生宿舍之间是可以互相访问没有限制的。

首先，我们测试宿舍访问图书馆，我们随意在宿舍和图书馆之间各选择一个主机，从宿舍端发送到图书馆端，进行多组这样的测试，截图如下：

PDU List Window												
Fire	Last Status	Source	Destination	Type	Color	Time(sec)	Periodic	Num	Edit	Delete		
●	Successful	PC4	PC6	ICMP	■	0.000	N	0	(edit)	(delete)		
●	Successful	PC5	PC7	ICMP	■	0.000	N	1	(edit)	(delete)		
●	Successful	Laptop...	PC6	ICMP	■	0.000	N	2	(edit)	(delete)		
●	Successful	PC5	Laptop8	ICMP	■	0.000	N	3	(edit)	(delete)		
●	Successful	Laptop...	Laptop8	ICMP	■	0.000	N	4	(edit)	(delete)		

图 3.41 实验 3 联通性测试 4

从图中可以看出，每一个报文都能正常发送，符合要求，网络设计以及实现正确。

然后，我们测试图书馆能够访问宿舍，我们随意在图书馆和宿舍间各自选择一个主机，从图书馆端发送到宿舍端，进行多组这样的测试，截图如下：

PDU List Window												
Fire	Last Status	Source	Destination	Type	Color	Time(sec)	Periodic	Num	Edit	Delete		
●	Successful	PC6	PC4	ICMP	■	0.000	N	0	(edit)	(delete)		
●	Successful	PC7	Laptop5	ICMP	■	0.000	N	1	(edit)	(delete)		
●	Successful	Laptop...	Laptop5	ICMP	■	0.000	N	2	(edit)	(delete)		
●	Successful	PC6	Laptop3	ICMP	■	0.000	N	3	(edit)	(delete)		
●	Successful	Laptop...	Laptop6	ICMP	■	0.000	N	4	(edit)	(delete)		

图 3.42 实验 3 联通性测试 5

从图中可以看出，每一个报文都能正常发送，符合要求，网络设计以及实现正确。

综上，图书馆和宿舍间能够互相访问。

2. 图书馆和学院之间互相访问

根据题目要求，图书馆和学院之间是可以互相访问没有限制的。

首先，我们测试学院访问图书馆，我们随意在学院和图书馆之间各选择一个主机，从学院端发送到图书馆端，进行多组这样的测试，截图如下：

PDU List Window												
Fire	Last Status	Source	Destination	Type	Color	Time(sec)	Periodic	Num	Edit	Delete		
●	Successful	PC8	PC7	ICMP	■	0.000	N	0	(edit)	(delete)		
●	Successful	PC2	Laptop7	ICMP	■	0.000	N	1	(edit)	(delete)		
●	Successful	PC1	Laptop8	ICMP	■	0.000	N	2	(edit)	(delete)		
●	Successful	Laptop...	PC7	ICMP	■	0.000	N	3	(edit)	(delete)		
●	Successful	PC3	PC6	ICMP	■	0.000	N	4	(edit)	(delete)		

图 3.43 实验 3 联通性测试 6

从图中可以看出，每一个报文都能正常发送，符合要求，网络设计以及实现正确。

然后，我们测试图书馆能够访问学院，我们随意在图书馆和学院间各自选择一个主机，从图书馆端发送到学院端，进行多组这样的测试，截图如下：

PDU List Window											
Fire	Last Status	Source	Destination	Type	Color	Time(sec)	Periodic	Num	Edit	Delete	
●	Successful	PC7	PC1	ICMP	■	0.000	N	0	(edit)	(delete)	
●	Successful	Laptop...	PC3	ICMP	■	0.000	N	1	(edit)	(delete)	
●	Successful	Laptop...	PC0	ICMP	■	0.000	N	2	(edit)	(delete)	
●	Successful	PC7	Laptop2	ICMP	■	0.000	N	3	(edit)	(delete)	
●	Successful	PC6	PC8	ICMP	■	0.000	N	4	(edit)	(delete)	

图 3.44 实验三联通性测试 7

从图中可以看出，每一个报文都能正常发送，符合要求，网络设计以及实现正确。

综上，图书馆和学院间能够互相访问。

3.宿舍和学院之间互相访问

根据题目要求，宿舍和学院之间是不能互相访问的

首先，测试宿舍访问学院，从宿舍端和学院端随机各自选择一个主机，从宿舍端发送到学院端，进行多组测试，每组测试进行多次，截图如下：

PDU List Window											
Fire	Last Status	Source	Destination	Type	Color	Time(sec)	Periodic	Num	Edit	Delete	
●	Failed	Laptop0	Laptop0	ICMP	■	0.000	N	0	(edit)	(delete)	
●	Failed	Laptop0	Laptop0	ICMP	■	0.000	N	1	(edit)	(delete)	
●	Failed	Laptop0	PC0	ICMP	■	0.000	N	2	(edit)	(delete)	
●	Failed	Laptop0	PC0	ICMP	■	0.000	N	3	(edit)	(delete)	
●	Failed	PC5	PC2	ICMP	■	0.000	N	4	(edit)	(delete)	
●	Failed	PC5	PC2	ICMP	■	0.000	N	5	(edit)	(delete)	
●	Failed	PC4	Laptop2	ICMP	■	0.000	N	6	(edit)	(delete)	
●	Failed	PC4	Laptop2	ICMP	■	0.000	N	7	(edit)	(delete)	

图 3.45 实验三联通性测试 8

由图，宿舍不能访问学院，结果正确。

然后，测试学院访问宿舍，从学院端和宿舍端随机各自选择一个主机，从学院端发送到宿舍端，进行多组测试，每组测试进行多次，截图如下：

PDU List Window											
Fire	Last Status	Source	Destination	Type	Color	Time(sec)	Periodic	Num	Edit	Delete	
●	Failed	PC1	Laptop6	ICMP	■	0.000	N	0	(edit)	(delete)	
●	Failed	PC1	Laptop6	ICMP	■	0.000	N	1	(edit)	(delete)	
●	Failed	PC3	Laptop5	ICMP	■	0.000	N	2	(edit)	(delete)	
●	Failed	PC3	Laptop5	ICMP	■	0.000	N	3	(edit)	(delete)	
●	Failed	Laptop0	PC4	ICMP	■	0.000	N	4	(edit)	(delete)	
●	Failed	Laptop0	PC4	ICMP	■	0.000	N	5	(edit)	(delete)	
●	Failed	Laptop0	Laptop5	ICMP	■	0.000	N	6	(edit)	(delete)	
●	Failed	Laptop0	Laptop5	ICMP	■	0.000	N	7	(edit)	(delete)	

图 3.46 实验三联通性测试 9

由图，学院不能访问宿舍，结果正确。

综上，学院和宿舍之间不可以互相访问，符合要求，网络设计和具体实现过程正确。

3.5 其它需要说明的问题

通过本次实验，我熟悉了 Cisco Packet Tracer 这个仿真工具，结合了理论课所学的知识，完成了子网划分、IP 配置、路由协议、访问控制等内容。并且，在此基础上，利用仿真工具和给定的条件，独立完成了一个网络的设计、配置和仿真检验的过程，收获十分丰富。

还需要说明的问题是，Cisco Packet Tracer 这个软件不支持最长前缀匹配，这个是我在做组网设计（就是最后一个实验）的时候发现的。因为按照一般设计，给定的地址范围是无法再支持路由器之间的子网的。但是可以从宿舍网络（或者图书馆网络）中剥离出前缀较长的一些网络块（也就是地址较少的网络块），用来分配给路由器之间的子网。在这个过程中，我发现最长前缀匹配是无效的，因为配置后的网络不连通。最后在多方协商下，放弃了上述分配想法，将路由器之间的子网划成私网地址 192.168.x.x

3.6 参考文献

- [1]James F.Kurose, Keith W.Ross, 陈鸣. 计算机网络-自顶向下方法[M]. 北京：机械工业出版社, 2018: 242-330
- [2]Kevia R. Fall, W. Richard Stevens, 吴英(译), 张玉(译), 许簪玮(译), 吴功宜(审). TCP/IP 详解 卷 1 (原书第二版) : 协议. 北京: 机械工业出版社, 2016 年 6 月 1 日

心得体会与建议

4.1 心得体会

(1) 本次计网实验中，我主要完成的工作有：

- 1) 完成了多线程的 Socket 编程，实现了客户端和服务器的通信，同时我还自主设计了异常处理界面 404 NOT FOUND 和 501 NOT IMPLEMENTED，用于找不到页面或者请求未实现的类型的文件。
- 2) 完成了可靠性传输编程的实验，用 VS 设计了 GBN、SR 和 TCP 协议，并且用清晰地防止展现了它们传输过程中滑动窗口的移动。
- 3) 利用了 CPT 软件，完成了组网的实验，并且自己设计了一个学校网络。通过实验，我对与划分地址、配置路由器和路由算法、构建 Vlan 到配置控制访问协议等等内容有了更加深入的了解。

(2) 通过本次实验，我主要的感受有：

- 1) 本次实验让我收获了很多的新技能，从图形化界面工具 Qt 的使用，到 C++ 模块化编程，u 对于 C++ 类和函数的封装有了更加深入的了解。同时，通过做组网的实验，对 Cisco Packet Tracer 这个软件的用法以及处理技巧有了更加深入的理解。
- 2) 通过实验，对理论课上的一些内容有了更加深入的了解。例如，通过 Socket 编程实验，我对于网络传输中，建立连接的过程有了更深入的了解。通过可靠性编程的实验，我理解透彻了 GBN 的发送方和接收方的 FSM 的流程，了解到了冗余 ACK 重传的意义。对于 GBN、SR 和 TCP 的传输机制理解更加深入。最后一个实验，让我对于子网划分和 vlan 的知识有了更深入的了解。

4.2 建议

整体来说，计算机网络的这三个实验都是工程性十分强的实验，做完每个实验之后，都有十分大的收获以及成就感。对于整个计算机网络的实验，我想提出以下几条建议：

- 1) 希望老师能在课堂上多讲的实验的内容，将实验和理论课堂融为一体。例如 Socket 编程的内容，我花了很长时间参考指导手册才明白整个 socket 编程的套路，又花了很长时间才弄明白 Qt 软件的一些使用方法，所以整个 socket 实验花了不少时间，
- 2) 最后组网的实验，如果我们采用最新版本的 Cisco Packet Tracer，会有更多更好用的无线路由器设备，例如 HomeRouter-PC-AP 等等，可以极大简化实验中一些不必要的配置步骤（尤其是最后一个实验），希望以后的本课程可以采用最新版的

Cisco Packet Tracer。

- 3) 希望实验可以稍微提前一点点开始，最后几次实验和考试周有一定的冲突，导致最后的两次实验有一定的紧张性。