
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 龙际全 班级 CS1603 学号 U201614577

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	2
1.5 系统测试及结果说明	5
1.6 其它需要说明的问题	10
实验二 数据可靠传输协议设计实验	11
2.1 环境	11
2.2 实验要求	11
2.3 协议的设计、验证及结果分析	11
2.4 其它需要说明的问题	17
实验三 基于 CPT 的组网实验	18
3.1 环境	18
3.2 实验要求	18
3.3 基本部分实验步骤说明及结果分析	20
3.4 综合部分实验设计、实验步骤及结果分析	36
3.5 其它需要说明的问题	40
心得体会与建议	41
4.1 心得体会	41
4.2 建议	41

实验一 Socket 编程实验

1.1 环境

1.1.1 开发平台

- (1) 操作系统: Microsoft Windows 10 专业版;
- (2) 处理器: AMD64 Family 21 Model 96;
- (3) 物理内存: 7,628 MB;
- (4) 网卡: Broadcom 802.11ac Network Adapter 和 Realtek PCIe GBE Family Controller;
- (5) IDE: Qt Creator 4.5.1 (Based on Qt 5.10.1);
- (6) 编译器: Desktop Qt 5.10.1 MinGW 32Bit。

1.1.2 运行平台

Windows 下直接运行 SocketWebServerByQt.exe 即可, 所需的动态库及其他依赖已经一并打包到软件目录, 由于使用 Winsock 开发包, 因此该软件暂不支持在其他平台编译运行。

1.2 系统功能需求

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

本系统主要由以下四个模块构成：

- （1） ui 模块 MainWindow;
- （2） 连接模块 AcceptThread;
- （3） 响应模块 RespondThread;
- （4） 参数配置模块 Config。

1.4 系统实现

1.4.1 ui 模块 MainWindow

- （1） 系统整体 ui，如图 1 - 1：



图 1 - 1 系统整体 ui

-
- (2) MainWindow 构造函数:
 - a) 加载 Config 中的程序参数;
 - b) 设置 ui 各部件初值;
 - c) 初始化 winsock 环境, 该功能本应该放在开始服务里, 但考虑到可能管理员会反复开启服务、停止服务, 反复初始化网络环境会影响性能。
 - (3) 显示标签 QLabel 设置方法:
 - a) 直接在 IDE 中将 Display Widget 中的 Label 拉到 ui 界面中, 并设置 Label 的 text 即可;
 - b) new 一个 QLabel, 调用 QLabel::setText 即可。
 - (4) 输入框 QLineEdit, 用于接收“监听地址”和“主目录”输入:
 - a) QLineEdit::setText 设置初始值;
 - b) QLineEdit::text() 获取管理员输入;
 - (5) 输入框 QSpinBox, 用于接收“监听端口”和“最大连接数”输入:
 - a) QSpinBox::setMinimum 设置最小值;
 - b) QSpinBox::setMaximum 设置最大值;
 - c) QSpinBox::setValue 设置初值;
 - d) QSpinBox::value 获得管理员输入;
 - (6) 主目录选择:
 - a) 管理员按下按钮调用 QPushButton 调用槽函数 QPushButton::on_PushBotton_clicked;
 - b) QFileDialog::getExistingDirectory 返回选择的目录路径;
 - (7) 启动服务器:
 - a) 创建服务器套接字 listenSocket;
 - b) 将服务器套接字 listenSocket 绑定到管理员输入的监听地址;
 - c) 将服务器套接字 listenSocket 绑定到管理员输入的端口;
 - d) 启动 AcceptThread 线程用于接收客户端请求, 防止阻塞 ui 主线程使程序无法响应;
 - e) 接收 AcceptThread 线程发来的相应信息, 更新 ui。
 - (8) 关闭服务器:
 - a) 析构 AcceptThread 线程;
 - b) 显示提示信息。
 - (9) 响应结果表格显示 QTableWidgetItem:
 - a) 根据实验要求, 待显示的信息为请求时间 Time、客户端 IP 地址、客户端端口号 Port、客户端请求文件方式 Method、客户端请求的文件 URL、客户端请求文件大小 File Length、响应结果状态 Respond, 调用 QTableWidgetItem::setHorizontalHeaderLabels 显示类名;
 - b) 实时更新客户端请求响应, 调用 QTableWidgetItem::insertRow 增加一行, 调用 QTableWidgetItem::setItem 增加表格元素。
 - (10) MainWindow 析构:
 - a) 关闭服务器套接字 listenSocket;
 - b) 关闭网络环境;
 - c) 释放 ui 资源。
 - (11) 管理员直接关闭程序:
 - a) 通过 MainWindow::closeEvent 获取;

-
- b) 强制打断连接线程 AcceptThread;
 - c) 调用自身析构函数释放资源。

1.4.2 连接模块 AcceptThread

- (1) 继承自 Qthread, 重写 Qthread::run() 即可, 由于实验中服务器不存在多个线程互斥同步的问题, 因此不需要复杂的信号灯或互斥锁操作, 线程中止可通过 AcceptThread 的析构函数传递信号;
- (2) 调用 accept 函数循环接收客户端请求;
- (3) 将每个客户端的请求套接字 sAccept 交由响应线程 RespondThread 处理, 避免阻塞线程使得其他客户端无法连接服务器;
- (4) 接收线程 RespondThread 发来的响应结果, 并将响应结果发到主线程更新 UI;
- (5) 循环执行 (1) ~ (3), 除非线程被打断;
- (6) AcceptThread 析构时发送打断信号等待线程中止, 不再接收客户端请求。

1.4.3 响应模块 RespondThread

- (1) 与 AcceptThread 同理通过继承 Qthread 而来;
- (2) 调用 recv 函数获取客户端发来的报文;
- (3) 找到报文头部 clientHttpRequest;
- (4) 正则表达式 regRequest(R"(([A-Z]+) (.*) HTTP/\d\.\d)") 提取头部中的关键信息: 请求方法 Method 和请求文件 URL;
- (5) 将 URL 替换为 Windows 下的路径, 即将 '/' 替换为 '\\', 并与主目录连接得到文件完整路径;
- (6) URL 若为一个目录, 则替换为目录下的主页 index.html;
- (7) 如果文件存在, 则响应报文头部中的状态码构造为 "200 OK", 如果不存在, 构造为 "404 Not Found" 并发送自定义 "404.html";
- (8) 根据 URL 文件类型构造不同的 HTTP 响应报文头部中的 "Content-Type", 让客户端知道该文件如何渲染;
- (9) 通过 fseek 函数获得 URL 文件大小, 构造响应报文头部的 "Content-Length", 让客户端知道何时停止接收文件;
- (10) 如果请求方法 Method 为 GET, 则调用 send 函数发送请求报文头部和请求文件;
- (11) 如果请求方法 Method 不为 GET 或者请求的文件类型服务器不支持, 则重新构造响应报文头部为 "501 Not Implemented", 并发送自定义 "501.html";
- (12) 将响应结果信息发送给 AcceptThread 线程, 由 AcceptThread 线程通知 ui 主线程更新表格;
- (13) 循环执行 (1) ~ (12), 除非 RespondThread 线程被打断;
- (14) RespondThread 线程析构时, 强制中止线程。

1.4.4 参数配置模块 Config

- (1) 服务器默认监听端口;
- (2) 服务器默认监听地址;
- (3) 服务器默认最大连接数;
- (4) 发送缓冲区大小;
- (5) 使用的 HTTP 协议版本号;

- (6) 使用的服务器连接方式 (Keep Alive)。

1.5 系统测试及结果说明

- (1) 操作系统: Microsoft Windows 10 64Bit Professional;
(2) 浏览器: Google Chrome 71.0.3578.98 64 位;
(3) 设置基本参数, 如图 1 - 2:



图 1 - 2 基本参数设置

- (4) 选择主目录, 如图 1 - 3:

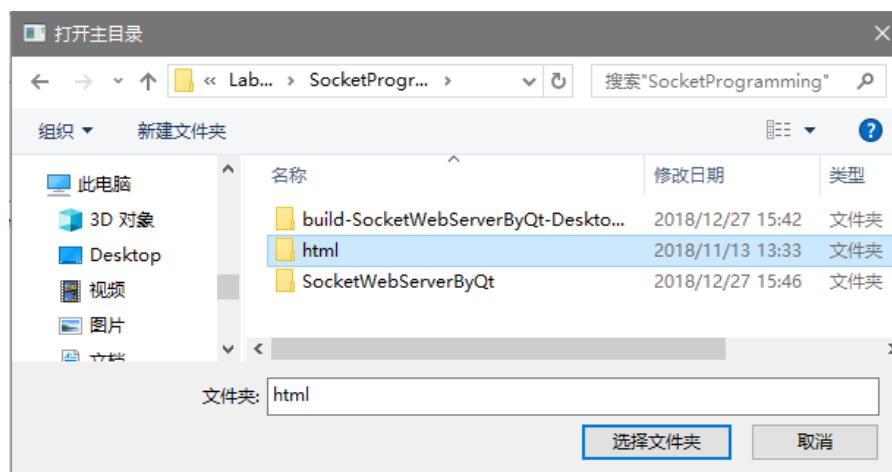


图 1 - 3 选择主目录

- (5) 访问服务器主页 <http://127.0.0.1:23456/index.html>, 如图 1 - 4:

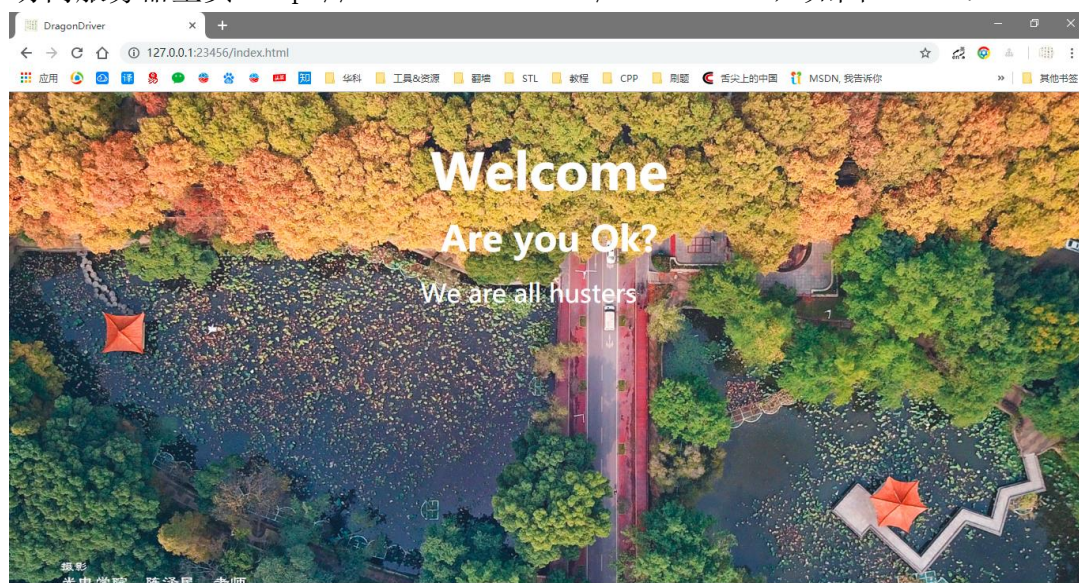


图 1 - 4 服务器主页

(6) 响应结果，如下图 1 - 5:

	Time	IP	Port	Method	Url	File Length	Respond
1	18:23:22	127.0.0.1	24134	GET	\index.html	1390	200 OK
2	18:23:23	127.0.0.1	24390	GET	\hust.jpg	833198	200 OK
3	18:23:23	127.0.0.1	24646	GET	\favicon.ico	4286	200 OK

图 1 - 5 主页响应结果

(7) 请求报文，如下图 1 - 6:

```
client http header: GET /index.html HTTP/1.1
Host: 127.0.0.1:23456
Connection: keep-alive
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/71.0.3578.98 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/
apng,*/*;q=0.8
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9,zh-CN;q=0.8,zh;q=0.7
Cookie: csrftoken=epyyes7gG8CQKhFEE6pzU3jADbxftoEF59k1Mat9hnCcTg0yeUunTKT9BoivsBkx
```

图 1 - 6 请求报文

(8) 响应报文，如下图 1 - 7:

```
respond http header: HTTP/1.1 200 OK
Content-Type: text/html
Content-Length: 1390
Server: csr_http1.1
Connection: close
```

图 1 - 7 响应报文

(9) TXT 纯文本显示，如下图 1 - 8、图 1 - 9、图 1 - 10:

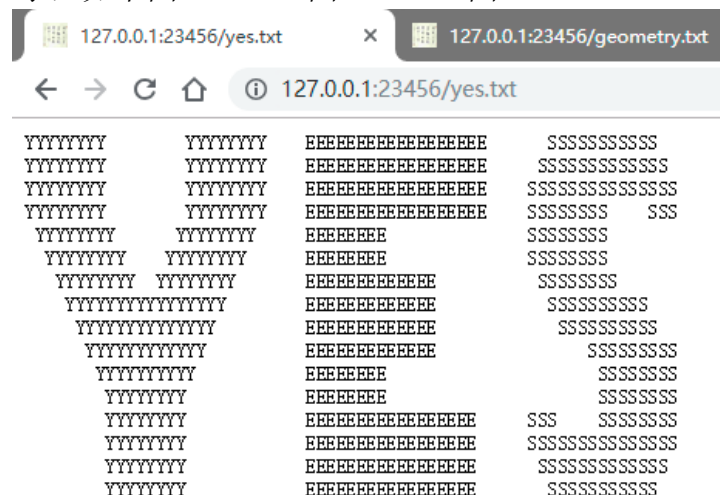


图 1 - 8 yes.txt

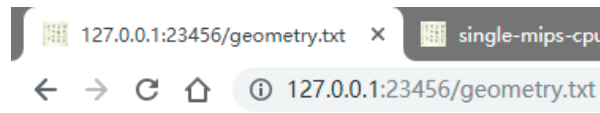


图 1 - 9 geometry.txt

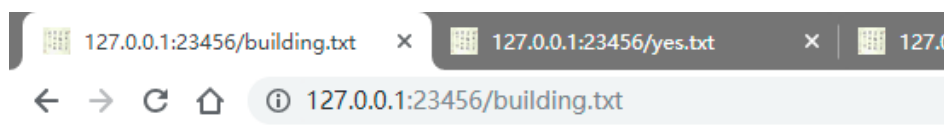


图 1 - 10 building.txt

(10) 文本响应结果，如下

21:22:34	127.0.0.1	10085	GET	\geometry.txt	1731	200 OK
21:27:27	127.0.0.1	43623	GET	\building.txt	1444	200 OK
21:28:20	127.0.0.1	360	GET	\MonaLisa	1293	501 Not Impl...
21:28:26	127.0.0.1	616	GET	\MonaLisa.txt	3694	200 OK
21:31:28	127.0.0.1	64105	GET	\yes.txt	1041	200 OK

图 1 - 11:

21:22:34	127.0.0.1	10085	GET	\geometry.txt	1731	200 OK
21:27:27	127.0.0.1	43623	GET	\building.txt	1444	200 OK
21:28:20	127.0.0.1	360	GET	\MonaLisa	1293	501 Not Impl...
21:28:26	127.0.0.1	616	GET	\MonaLisa.txt	3694	200 OK
21:31:28	127.0.0.1	64105	GET	\yes.txt	1041	200 OK

图 1 - 11 文本响应结果

(11) 图片显示，如下图 1 - 12:

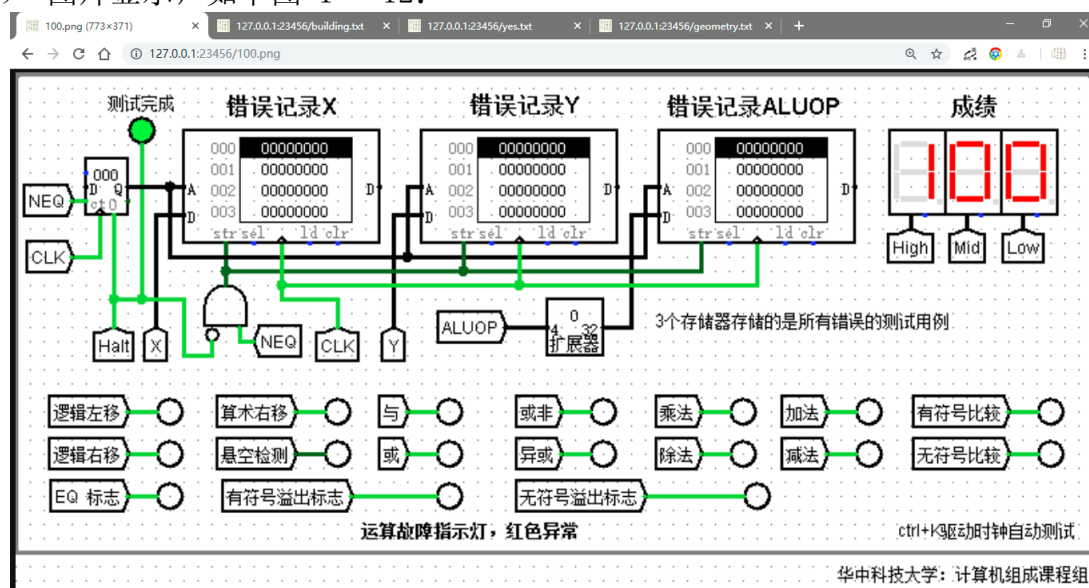


图 1 - 12 100.png

(12) 图片响应结果，如下图 1 - 13:

21:52:27	127.0.0.1	28789	GET	\100.png	12300	200 OK
21:55:10	127.0.0.1	12407	GET	\single-mips-cpu.png	28054	200 OK

图 1 - 13 图片响应结果

(13) 其他文件，返回未实现 501.html，如下图 1 - 14:



图 1 - 14 501.html

(14) 其他文件响应结果，如下图 1 - 15：

21:57:17	127.0.0.1	16760	GET	\geometry.exe	1293	501 Not Implemented
21:57:40	127.0.0.1	31352	GET	\hust.docx	1293	501 Not Implemented

图 1 - 15 501 未实现响应结果

(15) 文件不存在，返回 404.html，如下图 1 - 16：

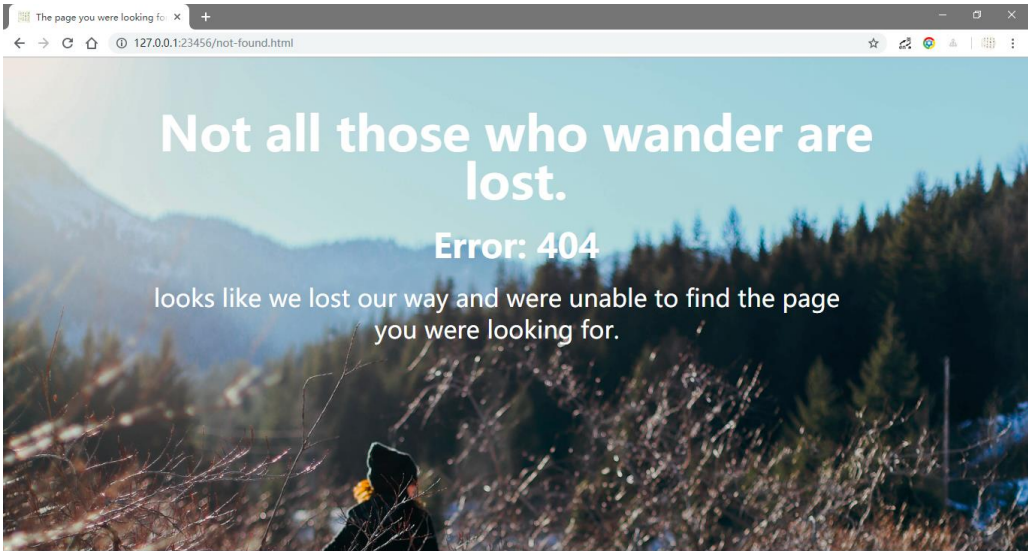


图 1 - 16 404.html

(16) 文件不存在响应结果，如下：

22:00:54	127.0.0.1	20602	GET	\not-found.html	1516	404 Not Found
22:00:54	127.0.0.1	20858	GET	\wander.jpeg	549463	200 OK
22:01:39	127.0.0.1	43386	GET	\aha.txt	1516	404 Not Found
22:01:40	127.0.0.1	43642	GET	\wander.jpeg	549463	200 OK

图 1 - 17 文件不存在 404 响应

(17) 局域网访问（手机访问），如下图 1 - 18

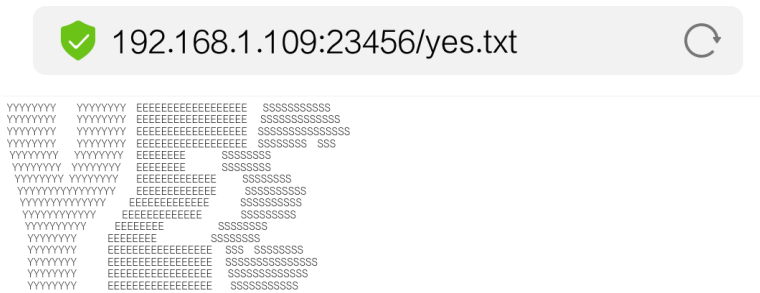


图 1 - 18 手机访问 yes.txt

(18) 局域网请求结果，如下图 1 - 19：

22:05:21	192.168.1.102	63383	GET	\yes.txt	1041	200 OK
22:05:21	192.168.1.102	63895	GET	\favicon.ico	4286	200 OK
22:06:24	192.168.1.102	64407	GET	\index.html	1390	200 OK
22:06:24	192.168.1.102	65175	GET	\hust.jpg	833198	200 OK
22:06:25	192.168.1.102	152	GET	\favicon.ico	4286	200 OK

图 1 - 19 局域网响应结果

(19) 多客户端并发，可从响应时间看，如下图 1 - 20：

22:14:41	127.0.0.1	59521	GET	\geometry.txt	1731	200 OK
22:14:41	192.168.1.102	4248	GET	\geometry.txt	1731	200 OK
22:14:42	192.168.1.102	4760	GET	\favicon.ico	4286	200 OK
22:14:42	127.0.0.1	59777	GET	\favicon.ico	4286	200 OK

图 1 - 20 多客户端并发响应结果

1.6 其它需要说明的问题

(1) 键入的 URL 最好不要有中文，系统没有对中文编码进行识别，如下：

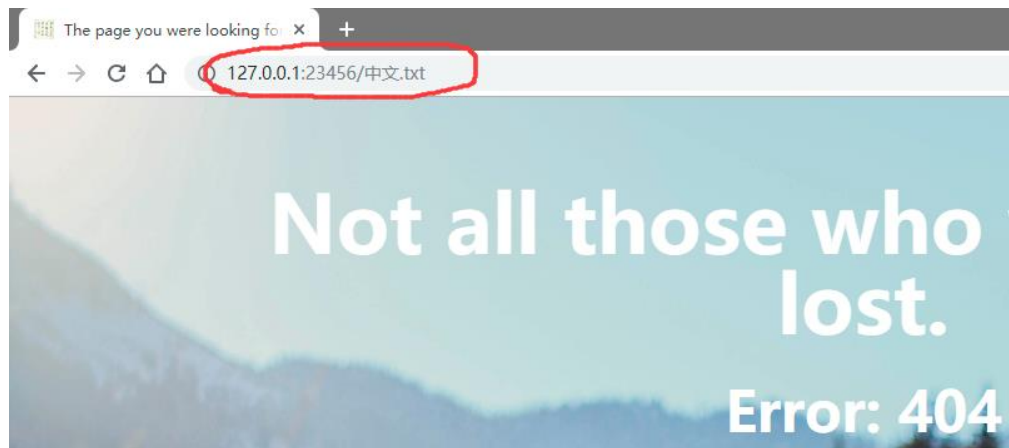


图 1 - 21 中文 URL 返回 404

22:18:24	127.0.0.1	33667	GET	\%E4%B8%A...	1516	404 Not Found
22:18:24	127.0.0.1	33923	GET	\wander.jpeg	549463	200 OK

图 1 - 22 中文 URL 响应结果

- (2) Qt 的 UI 系统不稳定，所以刚打开程序时，稍安勿躁，UI 稳定后再开启服务。

实验二 数据可靠传输协议设计实验

2.1 环境

- (1) 操作系统: Microsoft Windows 10 专业版;
- (2) 处理器: AMD64 Family 21 Model 96;
- (3) 物理内存: 7,628 MB;
- (4) IDE: Microsoft Visual Studio Community 2017;
- (5) 依赖库: 老师所发模拟网络环境 netsimlib.lib。

2.2 实验要求

本实验包括三个级别的内容, 具体包括:

- (1) 实现基于 GBN 的可靠传输协议, 分值为 50%。
- (2) 实现基于 SR 的可靠传输协议, 分值为 30%。
- (3) 在实现 GBN 协议的基础上, 根据 TCP 的可靠数据传输机制 (包括超时后只重传最早发送且没被确认的报文、快速重传) 实现一个简化版的 TCP 协议。报文段格式、报文段序号编码方式和 GBN 协议一样保持不变, 不考虑流量控制、拥塞控制, 不需要估算 RTT 动态调整定时器 Timeout 参数。分值 20%。

2.3 协议的设计、验证及结果分析

2.3.1 GBN 协议的设计、验证及结果分析

(1) GBNRdtSender

a) 数据结构

```
int base;           //基序号, 最早的未确认分组的序号
int nextSeqnum;     //下一个待发分组的序号
const int wndsize;   //滑动窗口大小, 实验建议为 4
const int seqsize;   //序号大小, 实验建议位数为 3 位, 即 0~7
Packet *const sendBuf; //发送缓冲区, 保存发送的报文, 用于重传
```

b) bool GBNRdtSender::send(Message & message)

- ① 将应用层 message 封装为传输层数据包 Packet, Packet 中除包含 message 所有数据外, 还包含当前发送的滑动窗口序号、数据包校验和;
- ② 由于 GBN 中采取累计确认机制, 因此只需要一个计时器, 因此当前滑动窗口 base 序号和 nextSeqnum 序号相等时, 表示所有接收方接收到了所有的数据包, 因此再次发送数据包时应重新启动定时器;

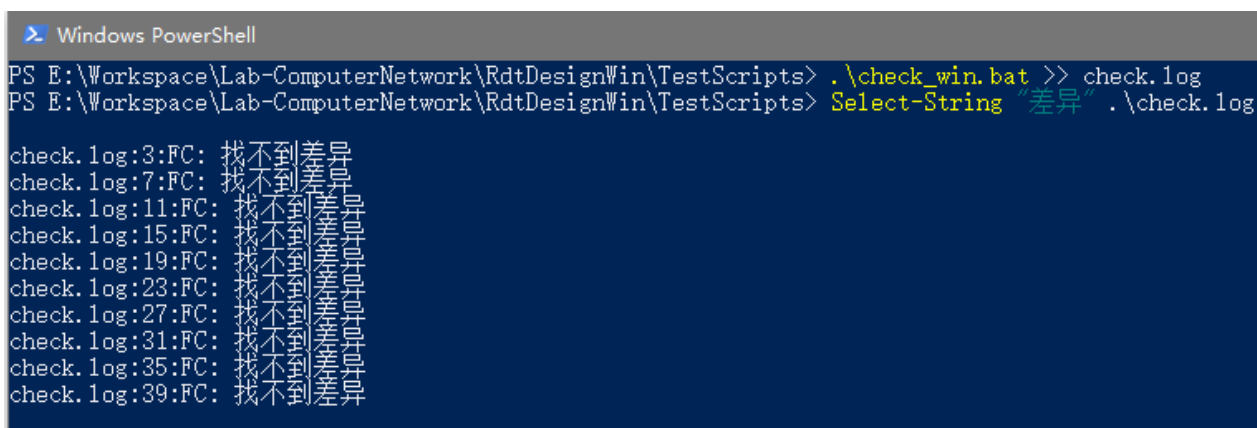
- ③ 调用 `sendToNetworkLayer` 发送数据宝;
 - ④ 发送完毕, 更新滑动窗口状态, 即 `nextSeqnum` 序号循环右移。
 - c) `void GBNRdtSender::receive(Packet &ackPkt)`
 - ① 通过数据包校验和检查发来的 `ack` 是否损坏, 如果损坏, 不做处理;
 - ② 更新滑动窗口状态, 由于采取累计确认机制, 接收到 `ack` 表明 `ack` 之前序号的所有数据包接收方已经全部接收完毕, 因此 `base` 序号直接循环右移到 `ack+1` 处, 同样, 如果此时 `base` 序号和 `nextSeqnum` 序号相同, 应停止计时器计时, 表明这一轮所有数据包接收方已全部接收完毕;
 - ③ 打印滑动窗口变化。
 - d) `void GBNRdtSender::timeoutHandler(int seqNum)`
 - ① 由于采取累计确认机制, 接收方没有缓存不会保留顺序, 因此必须重发所有已经发送且未确认的分组;
 - ② 重新启动定时器;
 - ③ 打印超时重传调试信息。
 - e) `bool GBNRdtSender::getWaitingState()`
 - ① 该函数返回 `true` 时, 表明发送方滑动窗口已满, 上层应用不应该再把 `message` 交由传输层处理;
 - ② 因此 `return (base + wndsize) % seqsize == (nextSeqnum) % seqsize;`即可。
- (2) `GBNRdtReceiver`
- a) 数据结构


```
int expectedSeqNum;      //下一个待接收的数据包
const int seqsize;      //窗口大小, 实验建议为 8
Packet lastAckPkt;      //保存上一次成功接收对应 ack 包
```
 - b) `void GBNRdtReceiver::receive(Packet & packet)`
 - ① 查看数据包 `packet` 校验和是否正确, 数据包损坏则发送上一次 `ack` 数据包;
 - ② 查看数据包 `packet` 序号是否是自己期待接收的包, 如果不是也即发送上一次 `ack` 数据包;
 - ③ 将传输层数据包 `packet` 解包成应用层 `message`, 并调用 `delivertoAppLayer` 递交给上层应用;
 - ④ 构造 `ack` 数据包, `ack` 序号为 `packet` 中序号 `seqnum`, 通知发送方;
 - ⑤ 更新 `expectedSeqnum`, 由于接收方没有缓存机制, 因此 `expectedSeqnum` 加一即可。
- (3) GBN 协议验证
- a) 执行结果, 如下图 2-1:

```
*****模拟网络环境*****: 模拟网络I
已发送应用层Message个数: 105
发送到网络层数据Packet个数: 162
网络层丢失的数据Packet个数: 10
网络层损坏的数据pakcet个数: 16
发送到网络层确认Packet个数: 152
网络层丢失的确认Packet个数: 15
网络层损坏的确认Packet个数: 6
```

图 2-1 GBN 执行结果

b) 检查脚本检查结果如下图 2-2:



```
Windows PowerShell
PS E:\Workspace\Lab-ComputerNetwork\RdtDesignWin\TestScripts> .\check_win.bat >> check.log
PS E:\Workspace\Lab-ComputerNetwork\RdtDesignWin\TestScripts> Select-String "差异" .\check.log

check.log:3:FC: 找不到差异
check.log:7:FC: 找不到差异
check.log:11:FC: 找不到差异
check.log:15:FC: 找不到差异
check.log:19:FC: 找不到差异
check.log:23:FC: 找不到差异
check.log:27:FC: 找不到差异
check.log:31:FC: 找不到差异
check.log:35:FC: 找不到差异
check.log:39:FC: 找不到差异
```

图 2-2 GBN 检查脚本检查结果

(4) GBN 结果分析

a) 滑动窗口变化, 如下图 2-3:

```
[SENDER]发送前窗口: [0* 1 2 3] 4 5 6 7
*****模拟网络环境*****: 启动定时器, 当前时间 = 2.475, 定时器报文序号 = 0, 分
*****模拟网络环境*****: 发送方的数据包将在12.905到达对方, 数据包为-->seqnum

[SENDER]发送后窗口: [0 1* 2 3] 4 5 6 7

*****模拟网络环境*****: 向上递交给应用层数据: AAAAAAAAAAAAAAAAAAAAAA
*****模拟网络环境*****: 接收方的确认包将在17.585到达对方, 确认包为-->seqnum

[RECEIVER]确认号ack: 0

*****模拟网络环境*****: 关闭定时器, 当前时间 = 17.585, 定时器报文序号 = 0

[SENDER]收到ack, 滑动窗口移动: 0 [1* 2 3 4] 5 6 7
```

图 2-3 滑动窗口变化

b) 超时重传, 如下图 2-4:

```
[Debug]发送超时

*****模拟网络环境*****: 启动定时器, 当前时间 = 204.638, 定时器报文序号 = 0, 定时器
*****模拟网络环境*****: 发送方的数据包将在212.178到达对方, 数据包为-->seqnum = 2,
*****模拟网络环境*****: 发送方的数据包将在220.698到达对方, 数据包为-->seqnum = 3,
*****模拟网络环境*****: 发送方的数据包将在230.718到达对方, 数据包为-->seqnum = 4,

[Debug]重发数据包完毕
```

图 2-4 超时重传情况

c) 综上, GBN 设计正确。

2.3.2 SR 协议的设计、验证及结果分析

(1) SRRdtSender

a) 数据结构

```
const int seqsize;      //发送序号大小
const int wndsize;      //发送窗口大小
Packet *const sendBuf;  //发送缓冲区, 避免反复构造析构
bool *const bufStatus;  //数据包确认状态
int base, nextSeqnum;   //窗口标志序号
```

b) bool SRRdtSender::send(Message & message)

- ① 将应用层 message 封装为传输层数据包 Packet, Packet 中除包含 message 所有数据外, 还包含当前发送的滑动窗口序号、数据包校验和;
- ② 由于 SR 中采取选择重传机制, 因此需要为每一个发送的数据包设置一个计时器;
- ③ 发送完毕, 更新滑动窗口状态、

c) void SRRdtSender::receive(Packet & ackPkt)

- ① 通过数据包校验和检查发来的 ack 是否损坏, 如果损坏, 不做处理;
- ② 更新滑动窗口状态, 由于采取选择重传机制, 因此可以将 base 序号循环右移到最后一个未确认的数据包;
- ③ 打印滑动窗口变化。

d) void SRRdtSender::timeoutHandler(int seqnum)

由于采取选择重传机制, 接收方设置有缓存, 因此只需将超时的数据包 seqnum 重传即可。

e) bool SRRdtSender::getWaitingState()

- ① 该函数返回 true 时, 表明发送方滑动窗口已满, 上层应用不应该再把 message 交由传输层处理;
- ② 因此 return (base + wndsize) % seqsize == (nextSeqnum) % seqsize;即可。

(2) SRRdtReceiver

a) 数据结构

```
const int seqsize;      //发送序号大小
const int wndsize;      //发送窗口大小
Packet lastAckPkt;      //上一个接收成功的 ack 包
Packet *const recvBuf;  //接收缓存
bool *const bufStatus;  //数据包接受状态
int base;
```

b) void SRRdtReceiver::receive(Packet & packet)

- ① 查看数据包 packet 校验和是否正确, 数据包损坏则发送上一次 ack 数据包;
- ② 查看数据包 packet 序号是否是自己期待接收的包, 如果不是也即发送上一次 ack 数据包, 判断是否为期待接收的数据包即判断数据包序号是否在接收方滑动窗口内;
- ③ 将接收缓存 recvBuf 里可用的数据包全部解包为应用层 message, 并调用 deliverToAppLayer 递交给上层应用;
- ④ 构造 ack 数据包, ack 序号为 packet 中序号 seqnum, 通知发送方;

(3) SR 协议验证

a) 执行情况, 如下图 2-5:

```

*****模拟网络环境*****: 模拟网络
已发送应用层Message个数: 210
发送到网络层数据包个数: 546
网络层丢失的数据Packet个数: 57
网络层损坏的数据pakcet个数: 44
发送到网络层确认Packet个数: 458
网络层丢失的确认Packet个数: 40
网络层损坏的确认Packet个数: 35

```

图 2 - 5 SR 协议验证

b) 检查脚本检查情况，如下图 2 - 6:

```

Windows PowerShell
PS E:\Workspace\Lab-ComputerNetwork\RdtDesignWin\TestScripts> rm .\check.log
PS E:\Workspace\Lab-ComputerNetwork\RdtDesignWin\TestScripts> .\check_win.bat >> check.log
PS E:\Workspace\Lab-ComputerNetwork\RdtDesignWin\TestScripts> Select-String "差异" .\check.log

check.log:3:FC: 找不到差异
check.log:7:FC: 找不到差异
check.log:11:FC: 找不到差异
check.log:15:FC: 找不到差异
check.log:19:FC: 找不到差异
check.log:23:FC: 找不到差异
check.log:27:FC: 找不到差异
check.log:31:FC: 找不到差异
check.log:35:FC: 找不到差异
check.log:39:FC: 找不到差异

```

图 2 - 6 检查脚本检查 SR 情况

(4) SR 协议结果分析

a) 滑动窗口变化，如下图 2 - 7:

```

[SENDER]发送前窗口: [可用 可用 可用 可用 ]不可用 不可用 不可用 不可用
*****模拟网络环境*****: 发送方的数据包将在1840.62到达对方, 数据包为-->seqnum = 0, acknum = -1,
*****模拟网络环境*****: 启动定时器, 当前时间 = 1838.8, 定时器报文序号 = 0, 定时器Timeout时间 =
[SENDER]发送后窗口: [已发送 可用 可用 可用 ]不可用 不可用 不可用 不可用
*****模拟网络环境*****: 接收方的确认包将在1844.48到达对方, 确认包为-->seqnum = 0, acknum = 0, c
*****模拟网络环境*****: 向上递交给应用层数据: AAAAAAAAAAAAAAAAAAAAAA

[RECEIVER]收到数据包, 窗口移动: 不可用 [期待 期待 期待 期待 ]不可用 不可用 不可用

```

图 2 - 7 SR 滑动窗口变化

b) 超时重传如下（只重传超时的包）图 2 - 8:

```

[Debug]数据包超时: 2
*****模拟网络环境*****: 发送方的数据包将在1900.63到达对方, 数据包为-->seqnum = 2,
*****模拟网络环境*****: 启动定时器, 当前时间 = 1888.19, 定时器报文序号 = 2, 定时器
[Debug]重发数据包完毕:2

```

图 2 - 8 SR 超时重传情况

c) 综上，SR 协议正确。

2.3.3 简单 TCP/IP 协议的设计、验证及结果分析

(1) TcpRdtSender (基于 GBNRdtSender)

- a) 数据结构, 相比 GBN, 增加 dupAckNum 检测冗余情况, 如下:

```
int base;           //基序号, 最早的未确认分组的序号
int nextSeqnum;     //下一个待发分组的序号
const int wndsize;  //滑动窗口大小, 实验建议为 4
const int seqsize;  //序号大小, 实验建议位数为 3 位, 即 0~7
Packet *const sendBuf; //发送缓冲区, 保存发送的报文, 用于重传, 大小应该是 seqsize
int dupAckNum;      //收到 3 个冗余 ack 快速重传
```

- b) bool TcpRdtSender::send(Message & message)

和 GBN 发送方法完全相同, 不再赘述。

- c) void TcpRdtSender::receive(Packet & ackPkt)

和 GBN 稍有不同的是, TCP 接收到窗口外 ack 即冗余 ack 时, 判断冗余 ack 是否连续到了三个, 若是如此, 表明当前网络环境不好, 数据包传输过程中会超时, 这个时候可以手动重传未确认的包而不需要等到数据包超时再重传。

- d) void TcpRdtSender::timeoutHandler(int seqNum)

TCP 采用快速重传, 配合 receiver 方法, 数据包超时时只重传最早未确认的包 (即 base) 即可。

- e) bool TcpRdtSender::getWaitingState()

和 GBN 完全相同, 不再赘述。

(2) TCPRdtReceiver

和 GBN 完全相同, 不再赘述。

(3) TCP 验证

- a) 运行情况, 如下图 2-9:

```
已发送应用层Message个数: 315
发送到网络层数据Packet个数: 744
网络层丢失的数据Packet个数: 71
网络层损坏的数据pakcet个数: 76
发送到网络层确认Packet个数: 631
网络层丢失的确认Packet个数: 55
网络层损坏的确认Packet个数: 57
```

图 2-9 TCP 执行情况

- b) 检查脚本检查如下图 2-10:

```
PS E:\Workspace\Lab-ComputerNetwork\RdtDesignWin\TestScripts> rm .\check.log
PS E:\Workspace\Lab-ComputerNetwork\RdtDesignWin\TestScripts> .\check_win.bat > check.log
PS E:\Workspace\Lab-ComputerNetwork\RdtDesignWin\TestScripts> Select-String "差异" .\check.log

check.log:3:FC: 找不到差异
check.log:7:FC: 找不到差异
check.log:11:FC: 找不到差异
check.log:15:FC: 找不到差异
check.log:19:FC: 找不到差异
check.log:23:FC: 找不到差异
check.log:27:FC: 找不到差异
check.log:31:FC: 找不到差异
check.log:35:FC: 找不到差异
check.log:39:FC: 找不到差异
```

图 2-10 检查脚本检查 TCP 情况

(4) TCP 结果分析

滑动窗口变化和 GBN 相同，因此只给出超时重传以及冗余重传截图。

a) 超时重传，如下（只重传最早未确认的包）图 2-11：

[Debug]发送超时

*****模拟网络环境*****：发送方的数据包将在4447.28到达对方，数据包为-->seqnum = 6,

*****模拟网络环境*****：关闭定时器，当前时间 = 4437.19, 定时器报文序号 = 0

*****模拟网络环境*****：启动定时器，当前时间 = 4437.19, 定时器报文序号 = 0, 定时器

[Debug]重发数据包完毕

图 2-11 TCP 超时重传情况

b) 冗余重传，如下（收到连续三个冗余 ack）图 2-12：

[SENDER]发送后窗口：0] 1* 2 3 4 [5 6 7

*****模拟网络环境*****：接收方的确认包将在4781.32到达对方，确认包为-->seqnum = -1, acknum = 4,

*****模拟网络环境*****：接收方的确认包将在4789.31到达对方，确认包为-->seqnum = -1, acknum = 4,

*****模拟网络环境*****：接收方的确认包将在4793.14到达对方，确认包为-->seqnum = -1, acknum = 4,

*****模拟网络环境*****：发送方发送的数据包丢失：seqnum = 5, acknum = -1, checksum = 41116, paylo

[SENDER]收到连续三个冗余ack:4, 快速重传

图 2-12 TCP 冗余超时情况

c) 综上，TCP 协议正确。

2.4 其它需要说明的问题

无。

实验三 基于 CPT 的组网实验

3.1 环境

- (1) 操作系统: Microsoft Windows 10 专业版;
- (2) 处理器: AMD64 Family 21 Model 96;
- (3) 物理内存: 7,628 MB;
- (4) IDE: Microsoft Visual Studio Community 2017;
- (5) 依赖库: 老师所发模拟网络环境 netsimlib.lib。

3.2 实验要求

熟悉 Cisco Packet Tracer 仿真软件, 利用 Cisco Packet Tracer 仿真软件完成实验内容。

3.2.1 基本部分

本部分实验为基础部分的实验, 分为四项内容。本部分实验将使用两张拓扑结构图配合完成实验, 如图 3-1 和图 3-2 所示。

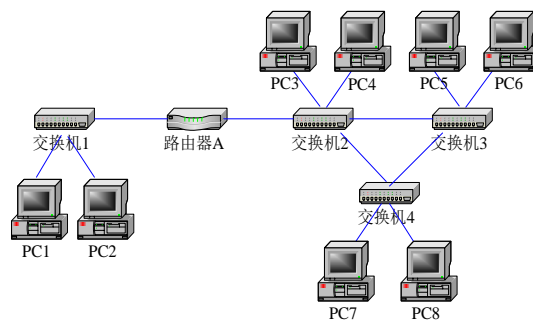


图 3-1 拓扑图 1

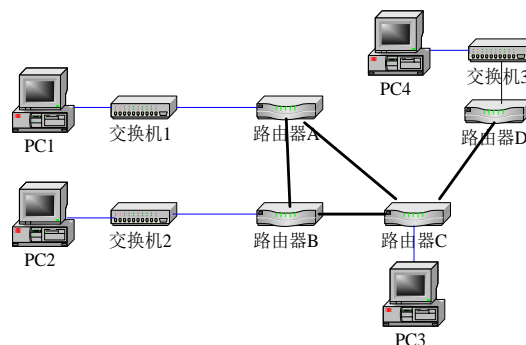


图 3-2 拓扑图 2

第一项实验——IP 地址规划与 Vlan 分配实验：

- (1) 使用仿真软件描述网络拓扑图 3.1。
- (2) 基本内容 1：
 - 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) 为路由器配置端口地址，使得两个子网内部的各 PC 机之间可以自由通信。
- (3) 基本内容 2：
 - 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。

第二项实验——路由配置实验：

- (1) 使用仿真软件描述网络拓扑图 3.2。
- (2) 基本内容 1：
 - 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) 设置路由器端口的 IP 地址
 - f) 在路由器上配置 RIP 协议，使各 PC 机能互相访问
- (3) 基本内容 2：
 - 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) 设置路由器端口的 IP 地址
 - f) 在路由器上配置 OSPF 协议，使各 PC 机能互相访问
- (4) 基本内容 3：
 - a) 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。
 - b) 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。

3.2.2 综合部分

(1) 实验背景：

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

(2) 组网需求：

- a) 图书馆能够无线上网
 - b) 学院之间可以相互访问
 - c) 学生宿舍之间可以相互访问
 - d) 学院和学生宿舍之间不能相互访问
 - e) 学院和学生宿舍皆可访问图书馆。
- (3) 实验任务要求:
- a) 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);
 - b) 根据理论课的内容,对全网的 IP 地址进行合理的分配;
 - c) 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

(1) 绘制网络拓扑图

根据实验要求给定的拓扑图,在 packet tracer 中绘制等效的拓扑图,如图 3 - 3。图中有 8 台 pc 机,四台交换机和一台路由器。

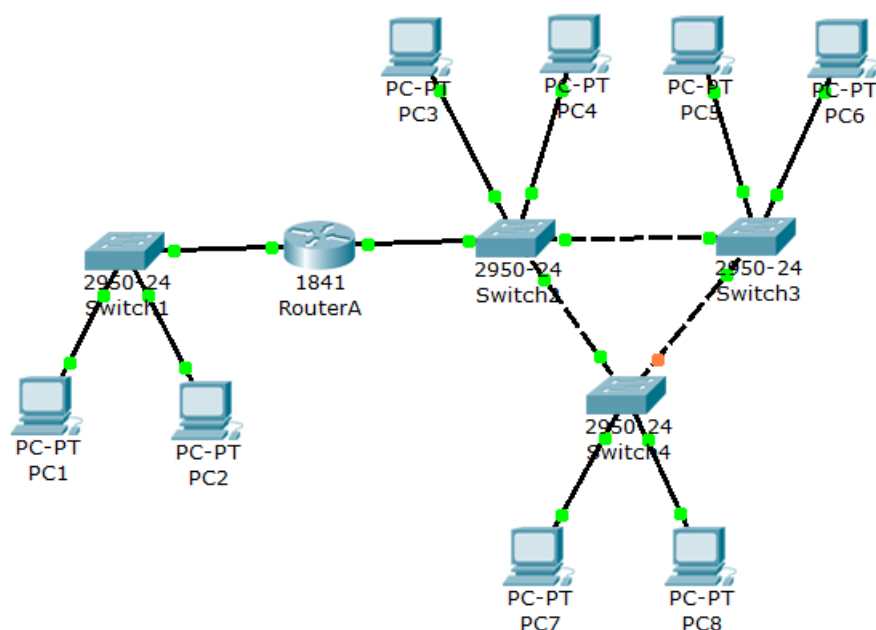


图 3 - 3 IP 地址规划拓扑图

(2) 基本内容 1:

- a) 单击路由器,弹出其配置信息,选择接口配置中的以太网接口 FastEthernet0/0,这是路由器与交换机 1 相连的接口,掌管的是 pc1 和 pc2,该子网地址为 192.168.0.0,

所以路由器端口也应使用该子网的地址。由于 192.168.0.0 地址不可用，所以将该端口地址设为 192.168.0.1，子网掩码自动生成，为 255.255.255.0。配置结果如图 3-4：

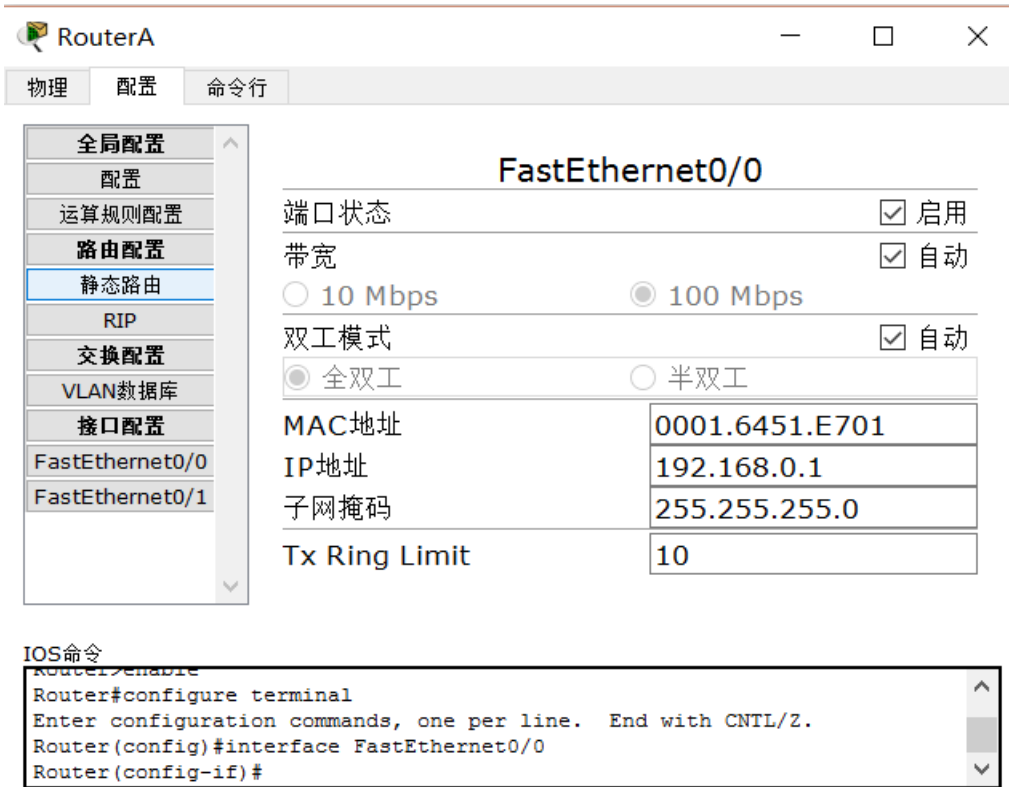


图 3-4 路由器配置

在图 3.4 的下方可以看到，每次配置以后，都会自动生成相应的命令行。因此我们可以想到，这些配置信息也可以用命令行实现，只是该软件提供了相当一部分功能的图形界面配置方法，所以我们要充分利用起来，尽量使用图形界面。

- b) 配置完接口 0/0，还需要配置接口 0/1，选择快速以太网端口 0/1，该端口控制右侧的 6 台 pc 机，子网地址为 192.168.1.0，所以该端口地址为 192.168.1.1，配置方法和结果都与端口 0/0 类似，不再赘述。
- c) 配置 PC 机 PC1 IP 地址，单击 PC1，弹出其配置窗口，选择快速以太网接口 0，配置 IP 地址为 192.168.0.2，当然也可以选择 192.168.0.0 网段的其他地址，只要不与路由器端口 1 的 IP 地址 192.168.0.1 重复即可；子网掩码可以直接生成，为 255.255.255.0，配置结果如图 3-5：

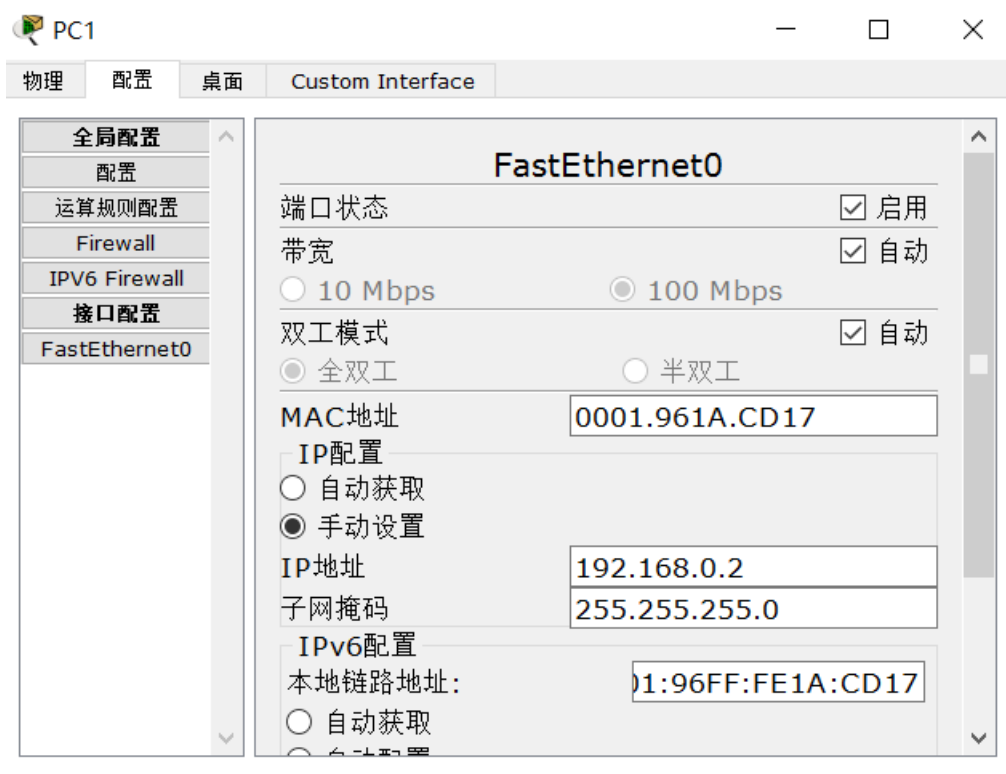


图 3 - 5 PC 配置 IP 地址和子网掩码

- d) 配置 PC 机 PC2 子网掩码，配置完 PC1 的 IP 地址以后，还要配置子网掩码，也就是掌管该 PC 机的路由器端口的 IP 地址，即路由器端口 0/0 的地址。选择 PC1 的“配置”，在“网关”一栏填写路由器端口 0/0 的 IP 地址 192.168.0.1，如图 3 - 6：

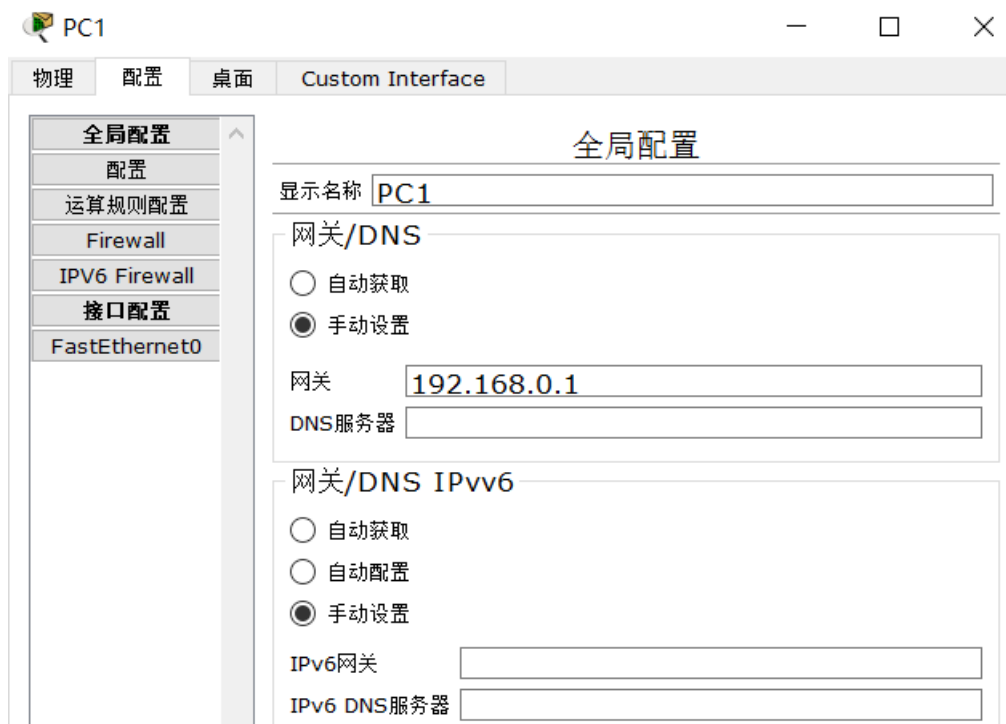


图 3 - 6 PC 配置网关

- e) PC2 和 PC1 处于同一网段，网关和子网掩码相同，因此只需给 PC2 分配同一网段内的不同 IP 地址 192.168.0.3 即可；
- f) PC3-PC8 位于 192.168.1.0 网段，所以可以在 192.168.1.0 网段中选择 8 个 IP 地址分别对其进行配置，该网段由路由器的 fa0/1 端口进行管理，所以 6 台 PC 机的网关地址都应填写 fa0/1 的 IP 地址 192.168.1.1。
- g) 网段 1 连通性测试：

配置完成以后，应测试各个 PC 机之间的连通性是否符合实验要求，这里先测试网段 192.168.0.0。

- ① 单击 pc1，选择：桌面->命令提示符，进入命令行界面，用“ping”指令与其他 pc 机进行通信，输入命令：ping 192.168.0.3，表示对 pc2（192.168.0.3）进行回话请求，向 pc2 发送四个报文段，均成功收到回复，如图 3-7：

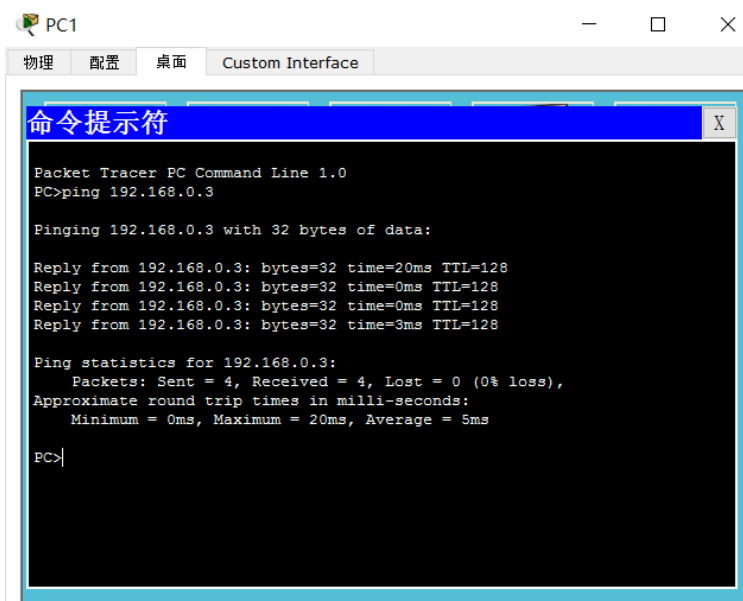


图 3-7 PC1 与 PC2 连通测试

- ② 单击 pc2，同样选择命令提示符，向 pc1 发起会话，也成功收到回复，如图 3-8：

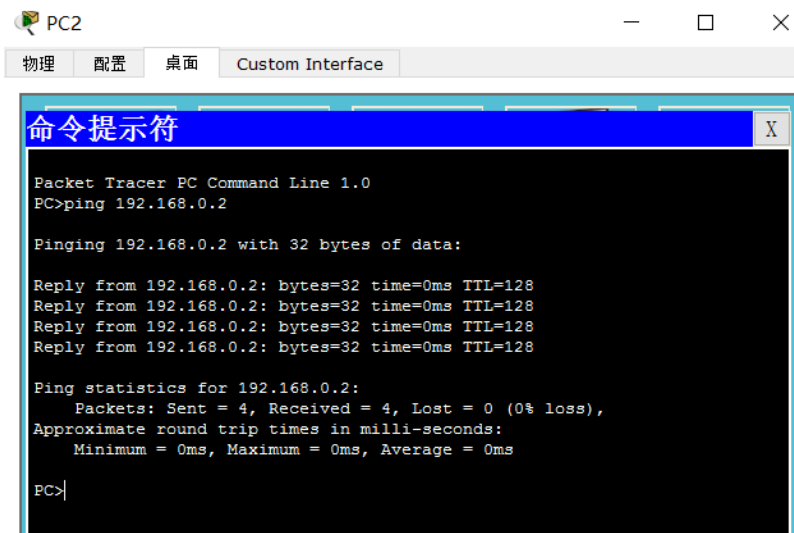


图 3-8 PC2 与 PC1 连通测试

- ③ 除了可以使用命令提示符进行连通性测试以外，还可以使用图形界面进行测试，其效果与命令提示符相同，在今后的测试中，都将使用图形界面。在 packet tracer 界面的右侧单击 图标，在需要通信的两台 pc 机上分别点击一次，就代表将信息从 pc1 发向 pc2，通信是否成功可以在界面下方的 PDU 列表窗口查看。如图 3-9，pc1 向 pc2 发送消息，然后 pc2 向 pc1 发送消息，两次连接的状态均为“成功”。

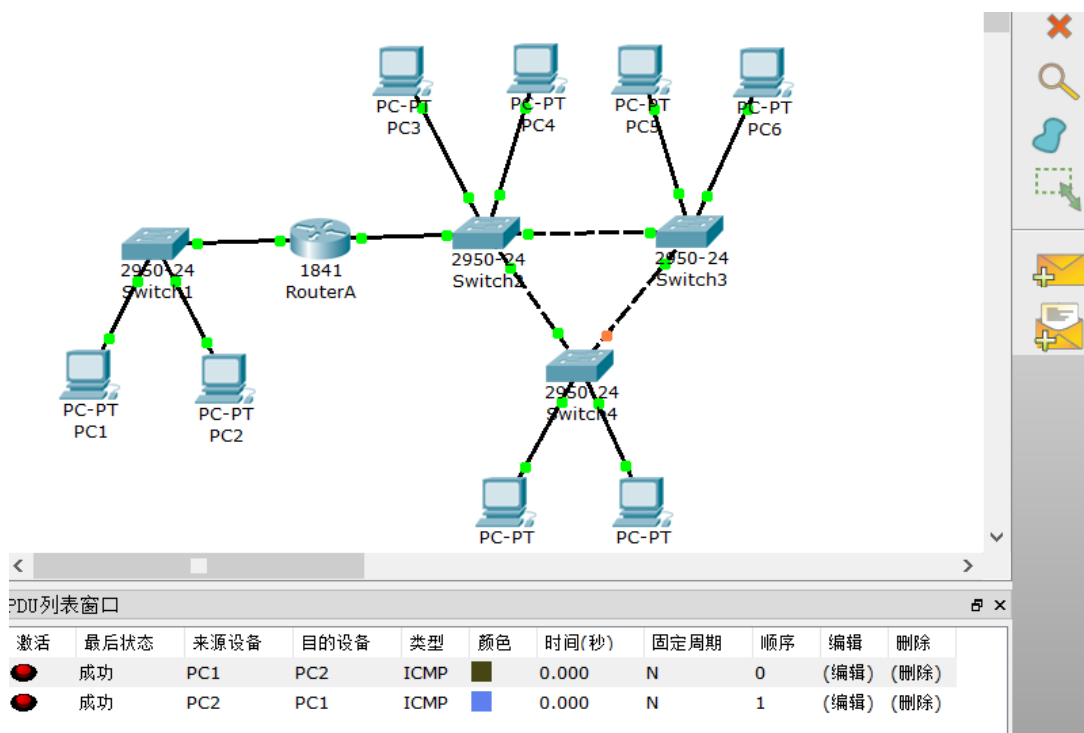


图 3-9 pdu 窗口

h) 网段 2 连通性测试：

在 pc3-pc8 中，任意两台 pc 之间进行通信测试，都会通信成功，测试结果如图 3-10 的 pdu 窗口所示。

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	PC3	PC4	ICMP		0.000	N	0	(编辑)	(删除)
	成功	PC3	PC7	ICMP		0.000	N	1	(编辑)	(删除)
	成功	PC6	PC8	ICMP		0.000	N	2	(编辑)	(删除)
	成功	PC7	PC5	ICMP		0.000	N	3	(编辑)	(删除)
	成功	PC8	PC4	ICMP		0.000	N	4	(编辑)	(删除)

图 3-10 网段 2 连通测试

i) 结果分析：

配置 IP 后，两个网段内部都可以进行通信，第二次配置 IP 以后，三个网段内部的内部也可以分别进行通信，符合实验要求。网段每个网段内部的 pc 机之间都是通过交换机连接起来的，内部通信不需要经过路由器就可以直接发送至对方，所以不管有没有给路由器端口分配 IP 地址、有没有给 PC 机配置网关，都可以进行内部通信。

(3) 基本内容 2:

完成基本内容 1 以后，要求将 pc4、pc6、pc8 从子网 192.168.1.0 网段中分离出去，编入子网 192.168.2.0，所以需要对这三台 pc 重新配置 IP。

a) 选择 pc4，将 IP 改为 192.168.2.1，如图 3-11:

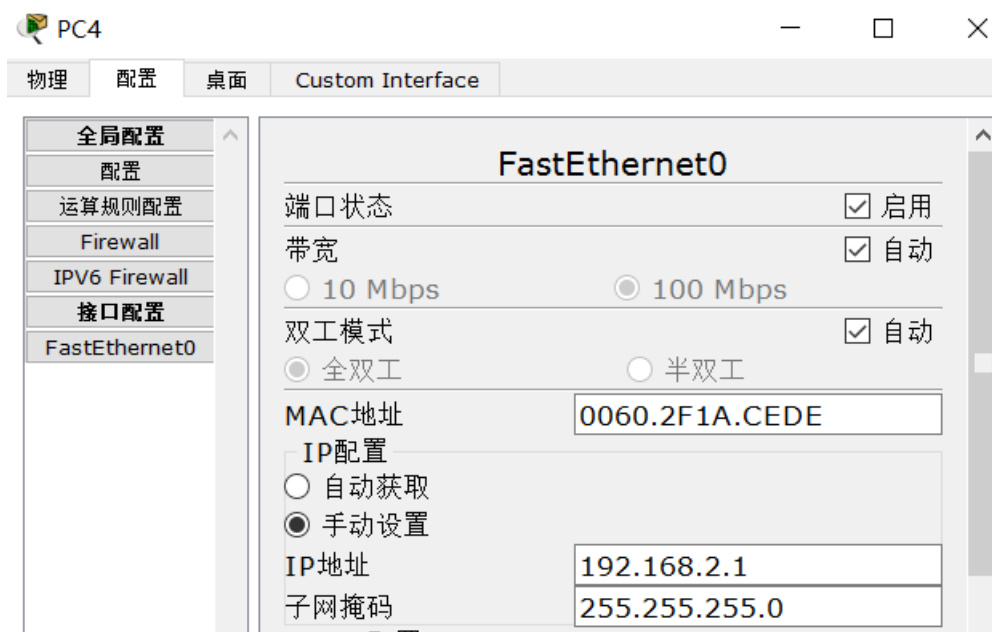


图 3-11 修改 IP

将 pc4 原来配置的网关 192.168.1.1 删除，如图图 3-12，不再需要网关。这是因为，路由器的右端接口 fa0/1 只能管理一个网段，这里没有分配子端口，所以只能从 192.168.1.0 和 192.168.2.0 中任选一个网段进行管理，这里选择了第一个网段，所以第二个网段没有网关，它只能内部通信，不能跨路由器通信。



图 3-12 删除网关

b) 连通性测试

- ① 为验证新的配置没有影响原有配置，需要对网段 192.168.0.0 和 192.168.1.0 的 pc 重新进行连通性测试，如图 3-13，这两个网段内部仍然可以通信。

PDU列表窗口										
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	PC3	PC5	ICMP		0.000	N	1	(编辑)	(删除)

图 3 - 13 网段 192.168.0.0 连通测试

- ② 测试新网段的 pc 机，pc4、pc6 和 pc8 之间的连通性，如图 3 - 14，任两台 pc 间都能正常通信。

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	PC4	PC6	ICMP		0.000	N	0	(编辑)	(删除)
	成功	PC4	PC8	ICMP		0.000	N	1	(编辑)	(删除)
	成功	PC8	PC4	ICMP		0.000	N	3	(编辑)	(删除)
	成功	PC8	PC6	ICMP		0.000	N	2	(编辑)	(删除)

图 3 - 14 新网段连通测试

- ③ 测试三个网段之间的连通性。如图 3.15，pc1 与 pc3，pc5 与 pc2 之间可以通信，说明 192.168.0.0 和 192.168.1.0 网段之间可以进行通信。Pc1 与 pc4，pc3 与 pc6……等都无法通信，也就是说 192.168.2.0 网段无法与其他两个网段进行通信，如图 3 - 15：

PDU列表窗口										
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	PC5	PC2	ICMP		0.000	N	1	(编辑)	(删除)
	失败	PC1	PC4	ICMP		0.000	N	2	(编辑)	(删除)
	失败	PC1	PC4	ICMP		0.000	N	3	(编辑)	(删除)
	失败	PC3	PC6	ICMP		0.000	N	4	(编辑)	(删除)
	失败	PC3	PC6	ICMP		0.000	N	5	(编辑)	(删除)
	失败	PC6	PC7	ICMP		0.000	N	6	(编辑)	(删除)

图 3 - 15 三个网段无法互通

c) 配置交换机 vlan；

- ① 交换机的 vlan 可用图形界面配置。题目要求将右侧的 6 台 pc 机划分到两个 vlan 中，所以需要在所有的路由器上都添加两个 vlan，vlan 号可以任意选择，只要不与路由器本身的 vlan 号重复就可以。这里选择 20 和 30 作为 vlan 号，两个 vlan 名分别为 class1 和 class2。以交换机 2 为例，选择在：配置->交换配置->vlan 数据库，添加两个 vlan。配置结果如图 3 - 16：



图 3 - 16 添加 vlan

- ② 交换机的 vlan 添加完成后，还需要对它的端口进行 vlan 划分。对于交换机与交换机相连，或者交换机与路由器相连的链路，都称为主干链路，选中该链路对应的端口，选择链路类型为“trunk”，trunk 链路的 vlan 默认包含了所有的 vlan，不需要修改，如图 3 - 17：

FastEthernet0/1

端口状态	<input checked="" type="checkbox"/> 启用
带宽	<input checked="" type="checkbox"/> 自动
<input type="radio"/> 10 Mbps <input checked="" type="radio"/> 100 Mbps	
双工模式	<input checked="" type="checkbox"/> 自动
<input checked="" type="radio"/> 全双工 <input type="radio"/> 半双工	
<div style="border: 1px solid #ccc; padding: 2px; display: flex; justify-content: space-between;"> Trunk VLAN 1-1005 </div>	
Tx Ring Limit	10

图 3 - 17 配置交换机主干线路

对于交换机与 pc 相连的链路，链路类型应该选择为 access 类型，access 链路的默认 vlan 为 vlan1，我们应该把它改成我们添加的 vlan，如果该端口连接了 pc3、pc5 或 pc7，就选择 vlan 10，如图 3. 29；如果端口连接了 pc4、pc6 或 pc8，则选择 vlan 20，如图 3 - 18 和图 3 - 19：

FastEthernet0/4

端口状态	<input checked="" type="checkbox"/> 启用
带宽	<input checked="" type="checkbox"/> 自动
<input type="radio"/> 10 Mbps <input checked="" type="radio"/> 100 Mbps	
双工模式	<input checked="" type="checkbox"/> 自动
<input checked="" type="radio"/> 全双工 <input type="radio"/> 半双工	
<div style="border: 1px solid #ccc; padding: 2px; display: flex; justify-content: space-between;"> Access VLAN 10 </div>	

图 3 - 18 配置交换机 PC 线路

FastEthernet0/5

端口状态	<input checked="" type="checkbox"/> 启用
带宽	<input checked="" type="checkbox"/> 自动
<input type="radio"/> 10 Mbps <input checked="" type="radio"/> 100 Mbps	
双工模式	<input checked="" type="checkbox"/> 自动
<input checked="" type="radio"/> 全双工 <input type="radio"/> 半双工	
Access	VLAN 20

图 3 - 19 配置交换机 PC 线路

d) 访问测试

- ① 先分别对两个 vlan 内部的主机进行访问测试，如图 3 - 20，对 pc3、pc5 和 pc7 进行互访，都能成功；对 pc4、pc6 和 pc8 进行互访，也可以成功。

PDU列表窗口										
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	PC3	PC5	ICMP		0.000	N	0	(编辑)	(删除)
	成功	PC3	PC7	ICMP		0.000	N	1	(编辑)	(删除)
	成功	PC7	PC5	ICMP		0.000	N	2	(编辑)	(删除)
	成功	PC8	PC6	ICMP		0.000	N	3	(编辑)	(删除)
	成功	PC4	PC6	ICMP		0.000	N	4	(编辑)	(删除)
	成功	PC6	PC4	ICMP		0.000	N	5	(编辑)	(删除)

图 3 - 20 同 vlan 互访

- ② 尝试让 vlan 10 的主机访问 vlan 20，不能成功，反之亦然，结果如图 3 - 21：

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	PC3	PC4	ICMP		0.000	N	0	(编辑)	(删除)
	失败	PC5	PC6	ICMP		0.000	N	1	(编辑)	(删除)
	失败	PC7	PC8	ICMP		0.000	N	2	(编辑)	(删除)
	失败	PC8	PC3	ICMP		0.000	N	3	(编辑)	(删除)
	失败	PC6	PC7	ICMP		0.000	N	4	(编辑)	(删除)

图 3 - 21 不同 vlan 互访

e) 路由器 vlan 配置

- ① 由于路由器只有两个端口，fa0/1 端口所管理的网络中有两个网段的 pc 机，所以要为该端口分配子端口，并为子端口分配 vlan，才能使两个 vlan 的主机可以通信。
- ② 对路由器 A 的快速以太网接口 0/1，创建子接口 fa0/1.1，并将其划分到 vlan 10 中，ip 地址设置为 192.168.1.1。这些都只能用命令行配置，如图 3 - 22：

```

Router(config)#int fa0/1.1
Router(config-subif)#
%LINK-5-CHANGED: Interface FastEthernet0/1.1, changed state to up

%LINEPROTO-5-UPDOWN: Line protocol on Interface FastEthernet0/1.1, change
to up

Router(config-subif)#encap dot1q 10
Router(config-subif)#ip addr 192.168.1.1 255.255.255.0
Router(config-subif)#exit

```

图 3-22 路由器增加子接口

- ③ 再用同样的方法建立子接口 fa0/1.2，将其划入 vlan 20，ip 地址设置为 192.168.2.1。
- f) PC1 和 PC2 的 vlan 划分
- ① 首先需要在 pc1 和 pc2 相连的交换机 1 上添加 vlan 30，与其他三台交换机的添加方法相同，但是不需要添加 vlan 20 和 vlan 10。
 - ② 将交换机与路由器相连的链路设置为 trunk 链路，vlan 使用默认参数，即全选；交换机与 pc 相连的链路设置为 access 链路，vlan 都选择 30 号，这样就将两台 pc 都划分到了 vlan 30 中。
 - ③ 为路由器的 fa0/0 接口创建一个子接口 fa0/0.1，划分到 vlan 30，ip 设置为 192.168.0.1，至此，pc1 和 pc2 的 vlan 划分完成。
- g) 访问测试

- ① 如图 3-23，pc1、pc2 与 pc3、pc5、pc7 都可以互相访问；

PDU列表窗口										
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	PC2	PC7	ICMP		0.000	N	1	(编辑)	(删除)
	成功	PC5	PC2	ICMP		0.000	N	2	(编辑)	(删除)
	成功	PC5	PC1	ICMP		0.000	N	3	(编辑)	(删除)

图 3-23 不同网段互访

- ② 尝试让两个不同 vlan 的主机互相访问，结果如图 3.35，vlan 10、vlan 20 和 vlan 30 间的任意两台 pc 都能互访成功。

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	PC2	PC4	ICMP		0.000	N	0	(编辑)	(删除)
	成功	PC6	PC5	ICMP		0.000	N	1	(编辑)	(删除)
	成功	PC8	PC2	ICMP		0.000	N	2	(编辑)	(删除)
	成功	PC3	PC6	ICMP		0.000	N	3	(编辑)	(删除)

- h) 结果分析：

- ① 网段 192.168.0.0 与 192.168.1.0 之间可以进行通信，这是因为路由器的端口都分配了 IP，而且在两个网段的 pc 机上将路由器端口 ip 设为了网关，所以由第一个网段发往第二个网段的消息在经过路由器时，路由器可以进行转发，故两个网段可以跨过路由器进行通信。
- ② 网段 192.168.2.0 与其他两个网段都不能进行通信，这是因为该网段没有配置网关，路由器只有两个端口，都已经被其他两个网段占用了，该网段将没有网关可用，所以它无法与其他子网进行通信，发送出的消息将被路由器忽略。

- ③ 在交换机上划分 vlan 以后，各个主机被分入不同的 vlan，一个 vlan 是一个逻辑上的整体，他们不受空间和子网的限制，所以 pc1、pc2 可以跟 pc3 处于同一个 vlan 中。若不在路由器上进行 vlan 配置，一个 vlan 内部的主机可以进行通信，但是他们不能和其他 vlan 的主机通信，相当于这个 vlan 是与世隔绝的。只有在路由器上进行 vlan 配置，一个 vlan 的主机向其他 vlan 主机发送的消息才能被路由器识别并转发，否则，路由器不清楚消息该发向何处，因为路由器本地没有相关 vlan 可以选择。

3.3.2 路由配置实验的步骤及结果分析

(1) 绘制网络拓扑图

根据图 3-2 给定的拓扑图，在 cpt 中绘制等效拓扑图，如。图中有 4 台 pc 机，3 台交换机和 4 台路由器。

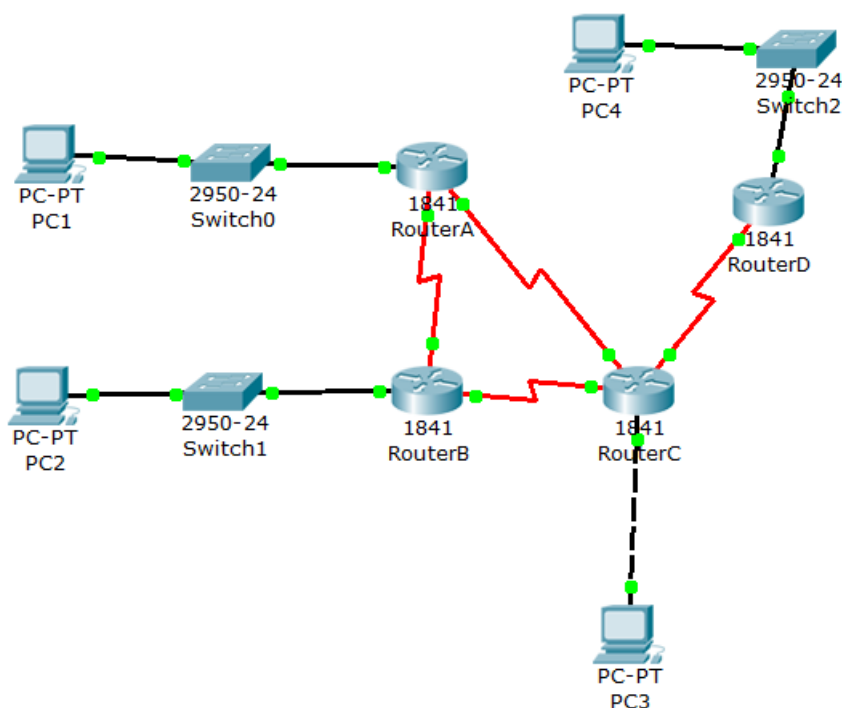


图 3 - 24 第二项实验拓扑图

(2) 基本内容 1:

a) 增加路由器接口

- ① 路由器与交换机连接可以使用快速以太网端口 FastEthernet，但是该端口不能用于路由器之间的互联，所以需要为路由器添加串口。以路由器 A 为例，如图 3 - 25，选择“物理”选项，在右端的开关处关闭路由器电源，在左侧的模块列表中，选中第一个模块“HWIC-2T”，拖到左侧的黑框中，再打开路由器电源，重新启动，并重新启用所有端口。



图 3 - 25 增加路由器接口

- ② 进入“配置”选项，可以看到路由器在原来的两个 fa 端口基础上，增加了两个 serial 串口，如图 3 - 26，这两个串口可以用于路由器之间的连接，图 3 - 24 中的红色线就是用这一串口进行连接的，用其他串口连接会提示错误。

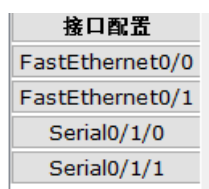


图 3 - 26 路由器 serial 串口

b) 配置路由器端口 IP

- ① 图 3.16 中有四个路由器，每个路由器管理一个子网，四个子网分别处于 192.168.1.0、192.168.2.0、192.168.3.0、192.168.4.0 网段，所以路由器与 pc 机或者交换机连接的端口 IP 应该配置为对应网段的 IP。路由器 A 的左端口 IP 为 192.168.1.1，路由器 B 的左端口 IP 为 192.168.2.1，路由器 C 的下方端口 IP 为 192.168.3.1，路由器 D 的上方端口 IP 为 192.168.4.1，图 3 - 27 所示为路由器 A 的 fa 端口 IP 配置，其他三个路由器与之类似。



图 3 - 27 路由器 fa 端口配置

- ② 第一步只配置了路由器 fa 端口的 IP,在路由器之间的 serial 端口 IP 也需要配置,这些端口的 IP 选择比较自由,只要不与 PC 机所处网段的 IP 重叠即可,我们不妨选择 192.168.5.0、192.168.6.0、192.168.7.0、192.168.8.0 四个子网为路由器之间的四条链路进行 IP 分配,链路两端的 IP 分别设为 xxx.xxx.xxx.1 和 xxx.xxx.xxx.2,比如路由器 A 的 serial0/1/0 端口 IP 为 192.168.5.1,与之相连的路由器 B 的 serial0/1/0 端口 IP 为 192.168.5.2,其他端口 IP 可以类推得到。
- c) 路由器 DCE、DTE 端口配置

路由器之间连接时,如果选用自动类型的连线,软件会报错,无法成功连接。只能选用 DCE 串口线,这种线是有方向的,先连的一端为 DCE,后连的一端为 DTE,在 DCE 一端需要设置时钟频率,DTE 一端则不需要修改,自动设置为默认值。以路由器 A 为例,其 serial0/1/0 串口为 DCE 端,选择“配置”选项,将该端口的时钟频率选为 64000,如图 3-28:

Serial0/1/0	
端口状态	<input checked="" type="checkbox"/> 启用
时钟速率	64000
双工模式	<input checked="" type="radio"/> 全双工

图 3-28 serial 串口时钟频率设置

- d) 路由器 RIP 协议配置

RIP 协议是用于自治系统内的动态路由协议,它只和与自己相连的路由器交换信息,所以每个路由器配置该协议时,只需要把自己每个端口的 IP 地址所在网段填上即可。以路由器 A 为例,它的 fa 端口 IP 为 192.168.1.1,两个 serial 端口 IP 分别为 192.168.5.1 和 192.168.6.1,所以应该配置这三个网段。选择:配置->路由配置->RIP,在方框中填入三个 IP 地址对应的网段,点击添加,如图 3-29,成功添加了三个网段,该路由器 RIP 协议配置成功。



RouterA

物理 配置 命令行

全局配置

配置

运算规则配置

路由配置

静态路由

RIP

交换配置

VLAN数据库

接口配置

RIP路由协议

网络 192.168.5.0

增加

网络地址

192.168.1.0

192.168.5.0

192.168.6.0

图 3-29 RIP 协议配置 IP 网段

- 其他三个路由器的配置与之相似,只需要改变网段的 IP 即可。
- e) 配置 PC 机 IP 地址和网关
- Pc1-pc4 分别处于 192.168.1.0、192.168.2.0、192.168.3.0、192.168.4.0 网段,可将其 IP 地址分别设置为 192.168.1.2、192.168.2.2、192.168.3.2、192.168.4.2。

每台 pc 的网关都应该与它距离最近的路由器的 IP 端口的地址，设为 192.168.1.1、192.168.2.1、192.168.3.1、192.168.4.1，这分别是路由器 A、B、C、D 的 fa 端口地址。关于 pc 机的 IP 地址和网关配置已经在前面介绍，在此不再详细阐述和截图，只大致叙述思路。

f) 访问测试

任选两台 PC 机相互通信，如图 3 - 30，pc1 与 pc2，pc2 与 pc3，pc3 与 pc4，pc4 与 pc1，均可进行访问（第一次访问请求可能失败，只要第二次访问成功，就说明两台 pc 可以进行访问），符合实验预期。

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	PC1	PC2	ICMP		0.000	N	0	(编辑)	(删除)
	成功	PC1	PC2	ICMP		0.000	N	1	(编辑)	(删除)
	失败	PC2	PC3	ICMP		0.000	N	2	(编辑)	(删除)
	成功	PC2	PC3	ICMP		0.000	N	3	(编辑)	(删除)
	成功	PC3	PC4	ICMP		0.000	N	4	(编辑)	(删除)
	成功	PC4	PC1	ICMP		0.000	N	5	(编辑)	(删除)

图 3 - 30 任意两台 PC 访问测试

(3) 基本内容 2

- 除配置路由 OSPF 协议与 RIP 协议不同外，其他配置均与 RIP 协议相同，因此这里只给出配置 OSPF 协议的截图。
- OSPF 协议是区别于 RIP 协议的另一种选路协议，同样可以用于以太网的通信，我们采用命令行来为路由器配置 OSPF 协议。以路由器 A 为例，如图 3 - 31，任选一个数字作为进程号，为路由器配置 OSPF，network 开头的命令将路由器端口的 IP 地址和子网掩码绑定到路由器上。该路由器有三个端口，所以使用了三条这样的语句。配置完成后，用 copy run startup 语句建立配置。

```

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 - 31 命令行配置 OSPF 协议

- 其他三个路由器的配置与之相似，只需要改变进程号和端口 IP。
- 访问测试

为了验证前面配置的 OSPF 协议是否有效，需要测试各个 PC 间的连通性。如果配置成功，那么 PC 之间应该和 RIP 协议配置完成以后一样可以任意通信。如图 3 - 32，pc1 对 pc2，pc2 对 pc3，pc4 对 pc2，pc3 对 pc1 都可以进行访问，所以我们可以认为，网络中的任意两台 pc 都可以互相访问，OSPF 协议配置成功。

PDU列表窗口										
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	PC1	PC2	ICMP		0.000	N	0	(编辑)	(删除)
	成功	PC1	PC2	ICMP		0.000	N	1	(编辑)	(删除)
	失败	PC2	PC3	ICMP		0.000	N	2	(编辑)	(删除)
	成功	PC2	PC3	ICMP		0.000	N	3	(编辑)	(删除)
	成功	PC4	PC2	ICMP		0.000	N	4	(编辑)	(删除)
	成功	PC3	PC1	ICMP		0.000	N	5	(编辑)	(删除)

图 3 - 32 OSPF 协议访问控制

(4) 基本内容 3

a) 路由器访问控制列表配置

用命令行对路由器 A 进行配置。要使得 pc1 无法访问其他网段，而且不能被其他网段访问，应该在路由器 A 与交换机 0 相连的端口进行配置。如图 3 - 33，先用 access-list 命令创建访问控制列表，可选用 100 以内的数字作为 acl 编号，deny 表示屏蔽某网段的消息，permit 表示接受某网段的消息，这里使用了 deny，因为我们要屏蔽 pc1 的通信。命令中的 ip 地址是要屏蔽或接受的网段，最后一个参数是子网掩码的反码。创建 acl 完成，打开端口 fa0/0，用 access-group 命令把 acl 绑定到路由器上，acl 就配置完成了。

```
Router(config)#access-list 35 deny 192.168.1.0 0.0.0.255
Router(config)#int fa0/0
Router(config-if)#ip access-group 35 in
Router(config-if)#exit
Router(config)#
```

图 3 - 33 访问控制列表 1

b) 访问测试

① 如图 3 - 34 所示，pc1 向其他三个 pc 的访问请求都发送失败，其他三个 pc 对 pc1 的访问也发送失败。

PDU列表窗口										
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	PC1	PC3	ICMP		0.000	N	0	(编辑)	(删除)
	失败	PC1	PC2	ICMP		0.000	N	1	(编辑)	(删除)
	失败	PC1	PC4	ICMP		0.000	N	2	(编辑)	(删除)
	失败	PC3	PC1	ICMP		0.000	N	3	(编辑)	(删除)
	失败	PC2	PC1	ICMP		0.000	N	4	(编辑)	(删除)
	失败	PC4	PC1	ICMP		0.000	N	5	(编辑)	(删除)

图 3 - 34 访问控制测试 1

② 测试 pc 以外的其他三个 pc 之间的连通性，如图 3 - 35，三个 pc 之间可以正常通信。

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	PC2	PC3	ICMP		0.000	N	0	(编辑)	(删除)
	成功	PC4	PC3	ICMP		0.000	N	1	(编辑)	(删除)
	成功	PC4	PC2	ICMP		0.000	N	2	(编辑)	(删除)

图 3 - 35 访问控制测试 2

c) 重新配置 ACL

- ① pc1 不能访问 pc2，但能访问其他 pc。第一次配置 acl 使只使用了 deny 指令，所以只能屏蔽作用，这里既需要屏蔽一部分消息，又需要允许一部分消息通过，所以还要使用 permit 指令。
- ② 仍然对路由器 A 进行配置，如图 3-36，使用 36 作为 acl 号，当然也可以使用其他数字。首先 deny 来自 pc2，也就是 192.168.2.0 网段的信息，然后 permit 其他任何信息，acl 列表就创建好了，然后绑定到端口 fa0/0 上。

```

Router(config)#access-list 36 deny 192.168.2.0 0.0.0.255
Router(config)#access-list 36 permit any
Router(config)#int fa0/0
Router(config-if)#ip access-group 36 in
Router(config-if)#exit

```

图 3-36 访问控制列表 2

d) 访问测试

- ① 尝试让 pc1 与 pc2 互相通信，如图 3-37，通信全都失败。

	失败	PC1	PC2	ICMP		0.000	N	0	(编辑)	(删除)
	失败	PC1	PC2	ICMP		0.000	N	1	(编辑)	(删除)
	失败	PC2	PC1	ICMP		0.000	N	2	(编辑)	(删除)
	失败	PC2	PC1	ICMP		0.000	N	3	(编辑)	(删除)

图 3-37 访问控制测试 3

- ② pc1 访问 pc2 以外的主机，pc2 访问 pc1 以外的主机，都能访问成功，如图 3-38。

	成功	PC1	PC4	ICMP		0.000	N	4	(编辑)	(删除)
	成功	PC3	PC1	ICMP		0.000	N	5	(编辑)	(删除)
	成功	PC2	PC3	ICMP		0.000	N	6	(编辑)	(删除)
	成功	PC4	PC2	ICMP		0.000	N	7	(编辑)	(删除)

图 3-38 访问控制测试 4

(5) 结果分析

- a) 从上述实验结果可以看出，我们用两种路由选择协议实现了网络通信，虽然配置的方法不同，但是达到了相同的目的。由于我们实验中只能模拟小型网络，所以看不出两者的区别，其实 RIP 和 OSPF 是有很大差异的。
- b) RIP 协议是一种传统的路由协议，适合比较小型的网络，但是网络的迅速发展和急剧膨胀使 RIP 协议无法适应今天的网络。OSPF 协议则是在网络急剧膨胀的时候制定出来的，它克服了 RIP 协议的许多缺陷。RIP 是距离矢量路由协议；OSPF 是链路状态路由协议。RIP 会定时广播路由表，而 OSPF 只有在路由状态发生变化时才广播路由表……两者的差异远不止这些，在实验中只做初步了解，还需要以后继续深入学习才能理解透彻。
- c) Acl 是访问控制列表 Access Control List 的英文缩写，用于控制路由器和交换机进出端口的数据包。在路由器 A 的端口上 deny 掉 pc1 所在网段的消息，路由器就会过滤掉与 pc1 有关的所有消息，使之无法与外界进行通信；若既要过滤部分消息，又要选通部分消息，则需要 deny 和 permit 指令的结合，就像进阶部分所做的一样。

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) 分配子网

先要把子网划分清楚，才能进行其他工作，原则是先分配子网主机多的，再分配子网主机较少的网段。

- a) 宿舍需要的主机最多，一个宿舍需要 200 台主机，这要划出 8 位用于子网编码，可容纳 255 台主机，多出的 50 多台主机，可以浪费这些编码，也可以等待后续使用。由于学校申请到的 IP 地址块为 211.69.4/22，低 10 位可供我们支配，所以可以将第 8、9 位限定，低 8 位用于编码。学校共有三个宿舍，所以分别使用 211.69.4/24、211.69.5/24、211.69.6/24 作为三个宿舍的 IP 地址，剩下的 211.69.7/24 可继续细分，用于学校其他部门的 IP 划分。
- b) 图书馆需要 100 台主机，这是除宿舍以外需要主机最多的部门。在分完宿舍的 IP 以后，还剩余 211.69.7/24 地址块，有 8 位可用。我们需要从中拿出 100 个 IP 地址，这至少需要 7 位，所以限定第 7 位，低 7 为用于编址，不妨设图书馆使用 211.69.7.0/25 网段，剩下 211.69.7.128/25 网段未使用，有 7 位可用。
- c) 最后剩下三个学院没有分配 IP，每个学院有 20 台主机，所以每个学院需要 5 位用于 IP 编址，前面剩余的网段还有 7 位可用。因此，低 5 位用于每个学院的 IP 编址，第 5、6 位用于学院的编码，分别为 00、01 和 10，还剩余一个 11 未使用，所以全校的 IP 未用完。三个学院的 IP 地址块分别为 211.69.7.128/27、211.69.7.160/27、211.69.7.192/27。

(2) 网络拓扑设计

- a) 从理论上来说，只需要一个路由器就可以解决全校的通信问题，所有的交换机之间也不需要相连，直接往路由器上连接即可。但是我们要充分结合实际，以我校的通信系统为例，所有宿舍楼、学院都应有自己的路由器，即便没有，也会与临近的宿舍或学院共享一台路由器，绝不会全校使用一台，因为这样更便于管理，更方便维修。
- b) 网络拓扑图的大致结构是：图书馆一台路由器、三个学院共用一台路由器、三个宿舍共用一台路由器。每 23 台主机共用一台交换机（因为一台交换机最多只有 24 个接口），一个学院内的所有交换机之间有一定的连接，保证每台交换机都是可达的，最终通过一台交换机与路由器相连。图书馆要求无线上网，所以既有无线上网的笔记本，也有有线上网的 pc 机，此外还有一台无线路由器。全校的三台路由器构成三角形结构，两两相连。
- c) 由于绘图空间不够用，所以只是象征性地画了几台主机，这不影响系统功能的实现，只要子网划分合理，主机的数量无关紧要。最终绘制的网络拓扑图如图 3-39 所示。

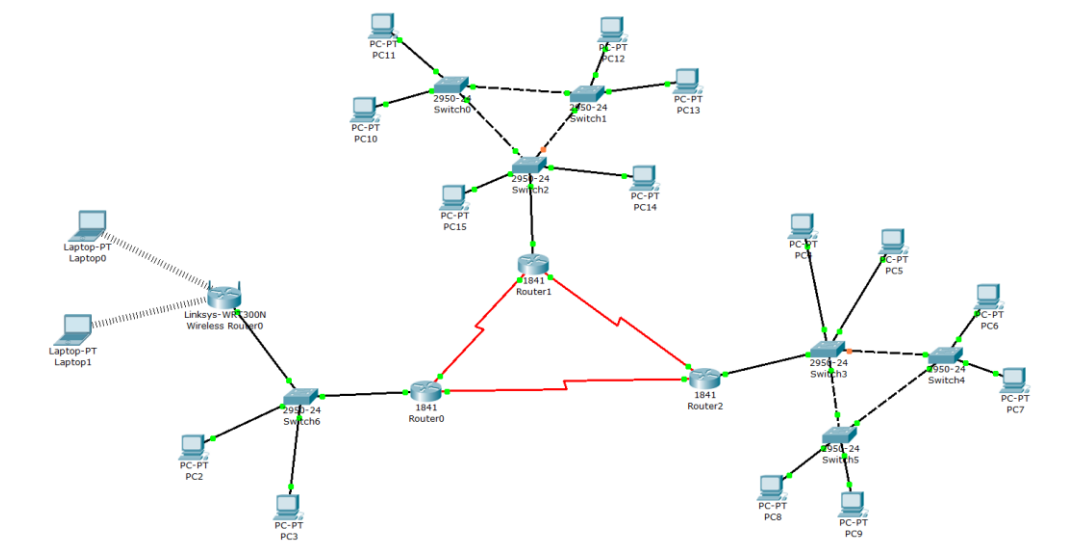


图 3 - 39 学校组网拓扑图

(3) 功能设计

- 三个学院是相对独立的子网，三个宿舍也是相对独立的子网，然而他们连接在路由器的同一个端口上，所以需要对路由器分配子端口。各学院之间要能够通信，所以还需要划分成功三个 vlan，同理三个宿舍也要划分 vlan。
- 学院和宿舍要跨过路由器访问图书馆，需要在路由器上配置 RIP 协议或者 OSPF 协议进行选路。
- 要禁止学院和宿舍互相访问，需要进行 ACL 配置。

3.4.2 实验步骤

(1) 交换机配置

学院的每台交换机都增加三个 vlan，分别编号为 2、3、4，对应三个学院。宿舍的每台交换机也增加三个 vlan，编号 5、6、7，对应三个宿舍。将交换机之间、交换机与路由器之间的链路设为 trunk 链路，交换机与 pc 机之间的链路设为 access 链路，vlan 设为该 pc 机对应的宿舍楼或者学院的 vlan，这样，就把三个宿舍、三个学院分成了 6 个 vlan。图书馆不需要划分 vlan。

(2) 路由器接口配置

- 给每个路由器添加两个 serial 接口，用于路由器之间的连接。路由器之间相连的接口 IP 使用 192.168.xxx.xxx 系列，不能使用 211.69.xx.xx 系列，否则会与学校的 pc 机 IP 相冲突。
- 学院和宿舍的路由器与交换机相连的接口，都分出三个子接口，子接口的 IP 从三个学院、三个宿舍的 IP 地址块中选出一个进行分配。图书馆的有线路由器不需要子接口，其 fa 接口 IP 分配为图书馆的 IP 地址块中的任一个。
- 学院和宿舍的路由器分出的子接口都需要分别划分到对应学院和宿舍的 vlan 中，六个子接口分别对应 6 个 vlan。
- 路由器之间的 DCE 端需要设置时钟频率为 64000。

(3) 无线路由器配置

- 选择：图形用户界面->setup, 如图 3 - 40，将无线路由器的 IP 地址设置为图书馆所在网段中的一个地址，如 211.69.7.1，子网掩码为 255.255.255.128（图中显示不全）。

- b) 接下来设置无线路由器的用户分配 IP 的范围，也就是图书馆的 IP 地址的范围，从 211.69.7.2 开始，最多 100 个用户。



图 3 - 40 无线路由器配置

(4) 笔记本改装

图书馆的无线上网功能可以用 pc 机或者笔记本实现，无论哪种电脑，都是默认安装有线网卡，所以我们先要对其改装。如图 3 - 41，将笔记本断电，把有线网卡拖下，从左侧的模块列表中选择第一个“Linksys-WPC300N”，拖到笔记本上，然后再开电，无线网卡就装上了，它会自动连接到无线路由器，并且路由器会给它分配 IP。



图 3 - 41 笔记本加无线网卡

(5) PC 机 IP 地址配置

对于除了笔记本以外的其他电脑，都将 IP 地址设置为其所在网段的任意值，只要不重复即可。网关设置为距离其最近的路由器端口的 ip 地址。

(6) 路由器 RIP 协议配置

在路由器上用图形界面配置 rip 协议，添加路由器所有端口所在的网段 IP 即可，图 3 - 42 所示为路由器 1 的 rip 协议配置。



图 3 - 42 学校组网配置 RIP 协议

(7) 路由器 ACL 配置

实验要求学院和宿舍不能互访，但是他们都可以访问图书馆，所以需要在学院和宿舍的路由器之间屏蔽掉对方的信号，以宿舍的路由器 1 为例，用如图 3 - 43 的三条命令表示过滤掉来自三个学院的信息。

```
Router(config)#access-list 35 deny 211.69.7.128 0.0.0.32
Router(config)#access-list 35 deny 211.69.7.160 0.0.0.32
Router(config)#access-list 35 deny 211.69.7.192 0.0.0.32
```

图 3 - 43 学校路由器访问控制配置

3.4.3 结果分析

(1) 如图 3 - 44 所示，测试学校各个部门内部的访问权限。三个学院内部、三个宿舍内部和图书馆内部的主机都能互相访问。

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	PC2	Laptop1	ICMP		0.000	N	0	(编辑)	(删除)
	成功	PC3	PC2	ICMP		0.000	N	1	(编辑)	(删除)
	成功	PC15	PC10	ICMP		0.000	N	2	(编辑)	(删除)
	成功	PC12	PC14	ICMP		0.000	N	3	(编辑)	(删除)
	成功	PC4	PC6	ICMP		0.000	N	4	(编辑)	(删除)
	成功	PC6	PC8	ICMP		0.000	N	5	(编辑)	(删除)

图 3 - 44 图书馆访问测试

- (2) 如图 3-45，学院和宿舍的主机 pc10 和 pc15 都可访问图书馆，而且图书馆的有限设备和无线设备都能被访问。

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	PC15	PC2	ICMP		0.000	N	0	(编辑)	(删除)
	成功	PC15	PC2	ICMP		0.000	N	1	(编辑)	(删除)
	失败	PC10	Laptop1	ICMP		0.000	N	2	(编辑)	(删除)
	成功	PC10	Laptop0	ICMP		0.000	N	3	(编辑)	(删除)

图 3-45 学校访问控制测试 1

- (3) 如图 3-46，学院和宿舍之间互访失败。

	失败	PC15	PC4	ICMP		0.000	N	4	(编辑)	(删除)
	失败	PC8	PC10	ICMP		0.000	N	5	(编辑)	(删除)
	失败	PC13	PC6	ICMP		0.000	N	6	(编辑)	(删除)
	失败	PC7	PC11	ICMP		0.000	N	7	(编辑)	(删除)

图 3-46 学校访问控制测试 2

- (4) 根据以上三点测试可知，系统实现的功能达到实验要求的功能，即各部门内部可以互访，全校都可以访问图书馆，而且学院和宿舍不可以互相访问。整个系统综合考察了因特网通信的各个部分，首先要进行合理而且不重叠的子网划分，然后要进行网络拓扑设计。通过选路协议的配置来实现网络的联通，其中 OSPF 比 RIP 更强大一些。通过 ACL 的配置来实现网络的选通，ACL 是保障网络安全的重要工具，它防止外界随意访问某些私密的信息，不然整个因特网都是“透明”的。

3.5 其它需要说明的问题

无。

心得体会与建议

4.1 心得体会

- (1) 第一次实验的收获最大，使用 Qt 和 Winsock 独立完成了一个比较完善的服务器框架，有并发，有 UI。以前学 Python 等脚本语言搭建个人网站时，不懂原理，即使照着别人的教程把网站搭好了，脱离了教程自己也什么都不会写，经过这次实验后，不光对网络的应用层协议有了更深的理解，也锻炼了自己解决问题的能力。
- (2) 第二次可靠数据传输实验相对简单一些，但也让我对网络的传输层有了一个更深入的了解，知道传输层应该做些什么，为网络哪一层服务，发送方接收方应该遵守什么样的约定。
- (3) 第三次更加简单，操作却略显繁琐，但是我其实觉得第三次实验是和整个网络课程贴合最紧密的，老师上课花时间花的最多的也是网络层的内容，经过这次实验，我感觉都可以回家去电信营业厅干活了，哈哈。

4.2 建议

- (1) 个人认为第一次 socket 编程实验虽然对同学们锻炼比较大，但是有点脱离实际，真正的服务器是不应该有 UI 的，而且老师也要求我们在 Windows 平台下用 winsock，出现问题网上的解决方案就少，我是希望以后的实验能让学生自由发挥，让学生们有更多的选择余地。
- (2) 另外我其实觉得第一次实验和第二次实验可以结合起来，毕竟两个实验都是编程实验，一个是应用层协议解析，一个是传输层协议设计，可以让以后的学弟学妹们做一个 C/S 模式的应用，规定两个应用通信必须使用不可靠的 UDP，然后让他们在 UDP 的基础上设计可靠传输协议，如果把 UI 这一块去掉，他们的整体实验难度应该高过我们这一届太多。