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 宋君平 班级 cs1609 班 学号 U201614749

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 环境

1.1.1 开发平台

处理器： Intel® Core™ i5-7200 CPU @ 2.50GHz 2.70GHz
内存大小： 8.00 GB (7.85GB 可用)
操作系统： Windows 10 家庭中文版
编译器： VS2017（gcc version 8.1.0）
链接器： vs2017
调式器： vs2017 (gdb version 8.0.1)

1.1.2 运行平台

处理器： Intel® Core™ i5-7200 CPU @ 2.50GHz 2.70GHz
内存大小： 8.00 GB (7.85GB 可用)
操作系统： windows 10 中文家庭版
第三方组件： 无

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

第二级：

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

行）

- ✧ 支持一定的异常情况处理能力。

第三级：

- ✧ 能够传输包含多媒体（如图片）的网页给客户端，并能在客户端正确显示；
- ✧ 对于无法成功定位文件的请求，根据错误原因，作相应错误提示。
- ✧ 在服务器端的屏幕上能够输出对每一个请求处理的结果。

具备完成所需功能的基本图形用户界面（GUI），并具友好性

1.3 系统设计

设计系统分为三个模块：

- ✧ Config.h 模块，声明端口号，最大连接数，IP 地址，缓冲区最大值等变量。
- ✧ Config.cpp 模块，为 Config.h 中变量赋初始值，防止在其他模块中修改对应值导致出错。
- ✧ server.cpp 模块，声明并定义 createclient 函数，主要用来处理当客户端发出请求时，服务器端打印出的请求报文和对应的相应报文。Main 函数，原本是想把建立连接，创建套接字，bind，listen 和前面的 createclient 放在一个类中，后面就直接放在 main 函数中，不过也没有太大差别。这样设计就可以完成系统的各个阶段的要求了。

1.4 系统实现

系统实现的关键在 server.cpp 函数中，关于服务器端和客户端通信的步骤在操作说明中也有介绍，

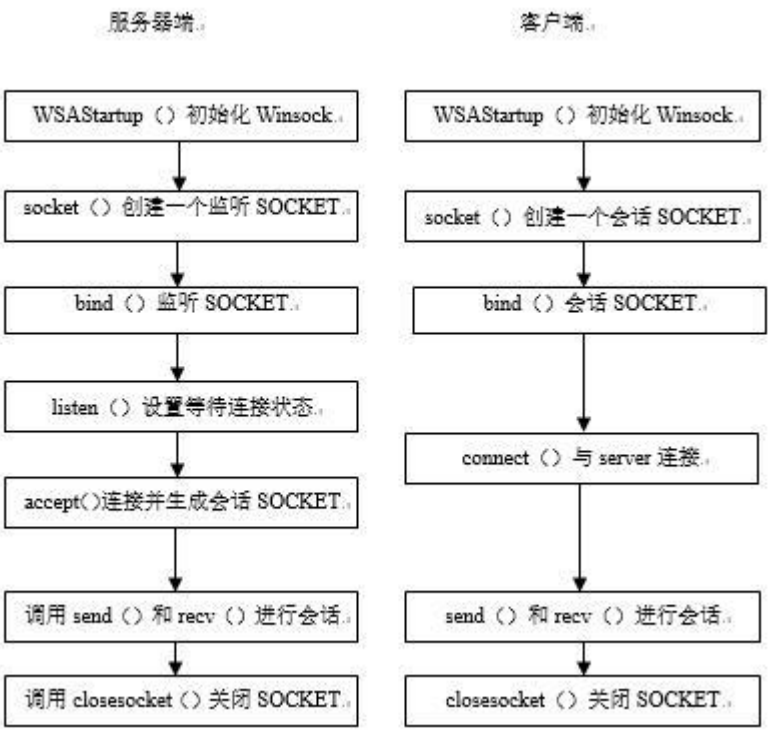

图 1.4.1 通信流程图

对应题目要求，在创建套接字，建立连接之后，需要获取客户端向服务器端发送的请求报文，解析报文后，服务器可以向客户端发送响应报文，并能将报文的回显在界面上。

- ✧ 对于简单的 html 文件，其中无图片，我们只需指出文件在服务器端的具体位置，当客户端请求时给出相对应的位置，如果匹配到的话，就将该文件 send 给客户端。
- ✧ 对于含有 jpg 或者文字和图片都有的文件，在发送时需要考虑一下大小，若果图片太大的话会导致后续发送出现问题。Send 会返回-1.并关闭 socket。所以在设置缓冲区大小时可以尽量设置大一点的值。

解析报文的方法是字符串的匹配，这个过程比较麻烦，我是用的方法是不断调用 string 库函数中的字符串匹配函数，找到自己需要的报文起点，再根据需求设置相对应的值。这个过程耗费了大量的时间。

```
strncpy(url_temp, strchr(req, '/') + 1, (strstr(req, "HTTP") - strchr(req, '/') - 2));
url_temp[strstr(req, "HTTP") - strchr(req, '/') - 2] = '\0';
if (strchr(url_temp, '.') == NULL) { //无后缀 +index.html
    strcat(url_temp, "index.html");
```

其中 url_temp 保存文件相对路径，url 保存文件在主机的绝对路径。

以下给出 createclient 的函数流程图：

Html 类型：

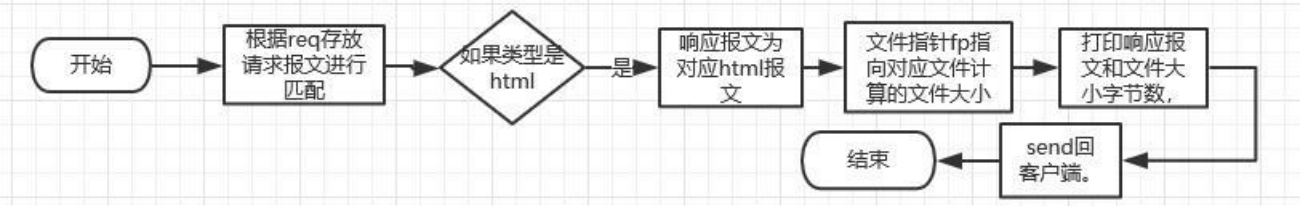

图 1.4.2 传输 html 文件

传输图片：

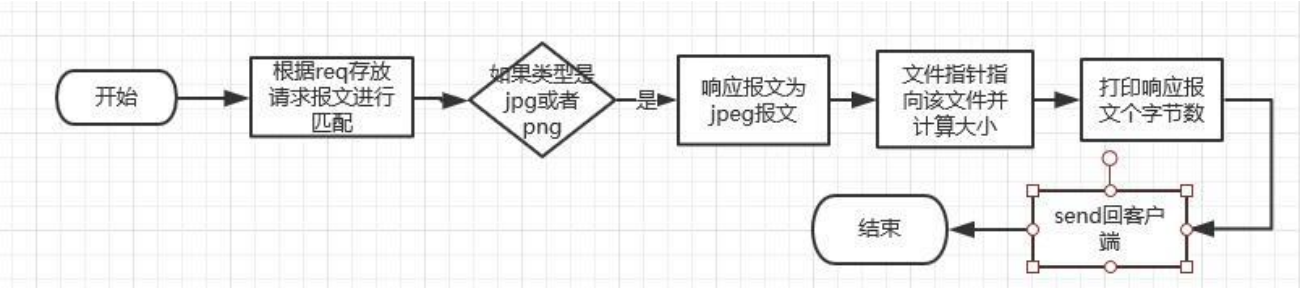

图 1.4.3 传输 jpg 文件

关于多线程的处理，采用了 h_thread = ::CreateThread(nullptr, 0, CreateClientThread, (LPVOID)sClient, 0, nullptr)创建线程，在每次循环时会创建一个线程。

1.5 系统测试及结果说明

（说明系统测试的硬件环境，说明系统测试的结果，对其进行分析，是否符合功能需求及你的设计要求）

按照要求编辑好后进行调试，所以使用测试环境为 vs2017，测试结果如下：

✧ 当建立好连接后出现如下结果，所用 ip 为本机 ip：

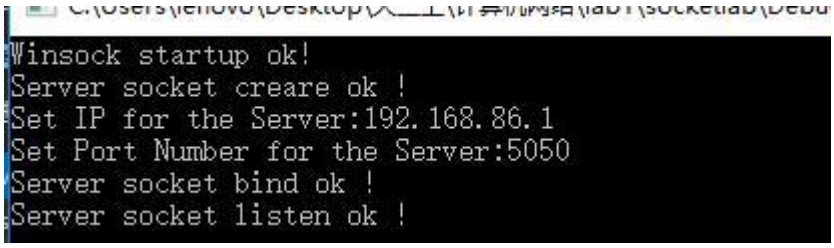

图 1.5.1 测试结果

✧ 此时在浏览器可以请求一个链接：输入 127.0.0.11：5002/index.html 得到结果如下：

图 1.5.2 测试结果

可以得到 hello world 的响应报文。同时控制台会打印出内容的大小包括首部行。

```
IP address: 2.0.218.228, port number: 56036
GET /index1.html HTTP/1.1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/6
/537.36 Edge/17.17134
Accept-Language: zh-Hans-CN,zh-Hans;q=0.8,en-US;q=0.5,en;q=0.3
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Upgrade-Insecure-Requests: 1
Accept-Encoding: gzip, deflate
Host: 127.0.0.1:5002
Connection: Keep-Alive


URL:C:/Users/lenovo/Desktop/index1.html
url_temp:index1.html
type:html
HTTP/1.1 200 OK
Content-Type: text/html;charset=UTF-8859-1
Content-Length: 293
```

图 1.5.3 测试结果

图中可以看出报文长度为 293 字节，其中包括了响应报文的首部行长度。

如果传输文件找不到：

```
URL:C:/Users/lenovo/Desktop/index.html
url_temp:index.html
type:html
HTTP/1.1 404 Not Found
Content-Type: text/html;charset=ISO-8859-1
Content-Length: 300

<html><body><h1>404 Not Found</h1><p>Cr's server: URL is wrong.</p></body></html>
```

图 1.5.3 测试结果

✧ 若传输有图片和文字的静态网页：

请求内容如下：

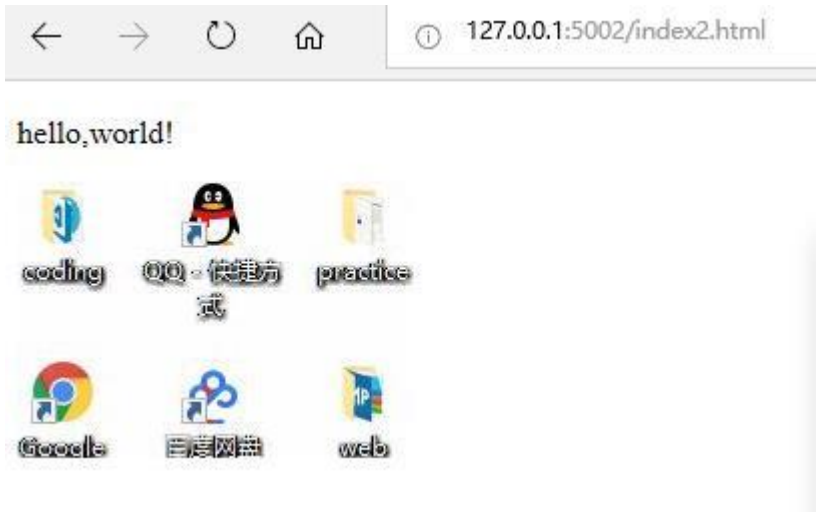

图 1.5.4 测试结果

可以看到响应内容包括了文字和图片。

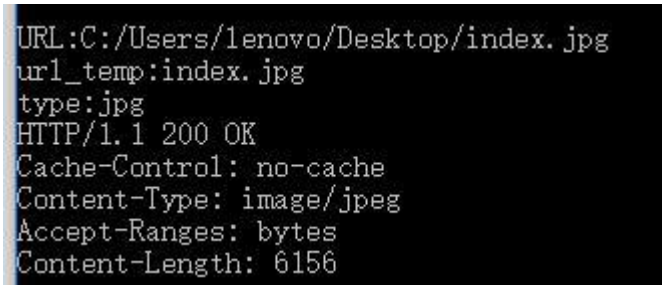

图 1.5.5 测试结果

✧ 若请求的网页不存在时会返回一个 404：

图 1.5.5 测试结果

所有测试结果均正确，符合预期。

1.6 其它需要说明的问题

✧ 在发送响应报文时要注意格式，格式错误的话是无法在浏览器中显示的。

比如我一开始的响应报文中没有加\r。虽然在控制台可以看到已经有报文发送，但是浏览器无法显示内容。

✧ 在开发过程中没有使用图形界面，因为一开不知道如何入手，花费了很长时间考虑做法，所以在界面这一块就没有太过在意。

✧ 发送时注意缓冲空间是否足够，一开始我发送的图片太过大了，就会导致发送内容为 0 的情况，发送也就失败了。

✧ 在完成之后又发现了一些可以完善的地方，比如退出的设置可以增加一个 esc 键，这也算是完成了要求的一步吧。

实验二 数据可靠传输协议设计实验

2.1 环境

2.1.1 开发平台

处理器： Intel® Core™ i5-7200 CPU @ 2.50GHz 2.70GHz
内存大小： 8.00 GB (7.85GB 可用)
操作系统： Windows 10 家庭中文版
编译器： VS2017（gcc version 8.1.0）
链接器： vs2017
调式器： vs2017 (gdb version 8.0.1)

2.1.2 运行平台

处理器： Intel® Core™ i5-7200 CPU @ 2.50GHz 2.70GHz
内存大小： 8.00 GB (7.85GB 可用)
操作系统： windows 10 中文家庭版
第三方组件： 无

2.2 实验要求

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

✧ 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。

✧ 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。

✧ 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。

本实验包括三个级别的内容，具体包括：

✧ 实现基于 GBN 的可靠传输协议。

✧ 实现基于 SR 的可靠传输协议。

✧ 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。

2.3 协议的设计、验证及结果分析

2.3.1 GBN 协议的设计、验证及结果分析

✧ 设计结果

在 GBN 协议中，允许发送方发送多个分组（当有多个分组可用时）而不需等待确认，但它也受限于在流水线中未确认的分组数不能超过某个最大允许数 N。本次实验所采用的窗口最大数为 4，因此允许的序号最大数为 8。

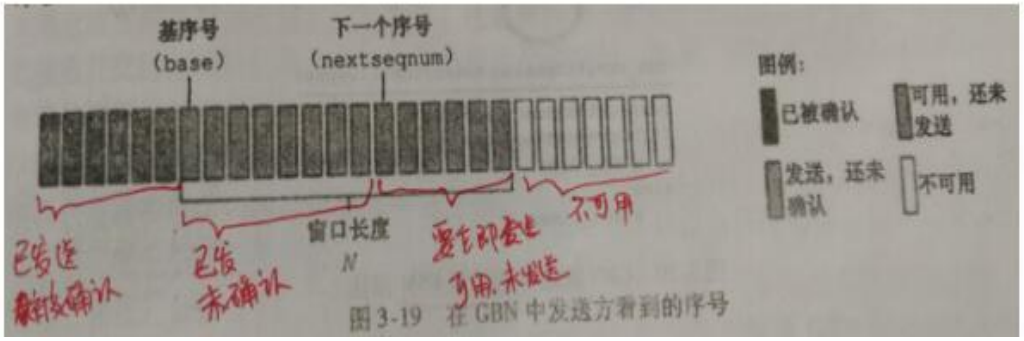

图 2.1.1GBN 协议

实现过程中首先将 base 设为基序号，他是最早未确认分组的序号，base 定义在发送方 sender 中，对应的 lastseqnum 是指从发送方收到的已确认的一个分组信息，这个分组已经通过网络层发给了发送方且发送方已经收到了确认。这个信息定义在接收方中。每次放松方通过调用网络层的发送命令将数据包发送，接收方的动作为，接收数据包，比较 cheksum 同时比较收到的序号是否为上一个数据报的下一个序号，满足的话将对应的信息保存，需要 nextseqnum 期望收到的下一个序号以及 checksum。而这里 nextseqnum 的处理比较特殊，采用了 this->expectSeq=this->expectSeq%WINDOW_SEQ;这样可以保证这个序号始终处于 0-7 之间，不会因为发送数据很多导致过大。还有一点需要注意的是，当接收方已经收到正确报文并且成功提交上层，返回发送方，由于网络层数据是不可靠的数据传输，因此发送过去的数据可能丢失，但这并不影响继续向下运行，因为此时的数据已经提交，所以这是的 sender 什么也不用做，只需滑动窗口继续向下即可。除此之外还要注意 GBN 协议中每次只需给第一个数据报定时处理，因为在这个协议中当 base 数据丢失时窗口中的所有数据都必须重新传输。

✧ 验证结果与结果分析

设计好之后可以开始测试，下面给出测试过程中的截图并进行适当分析：

首先观察最终结果：

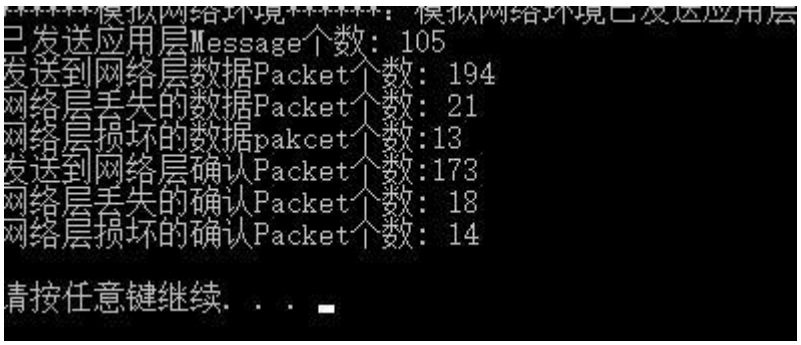

图 2.1.1GBN 测试结果

Send.txt 中数据条数为 105，所以总计发送了 105 个数据包。其余信息为网络层的不可靠传输导致的数据报丢失和重传次数灯。

首先观察由于发送方发送的数据报由于超时发送方的操作：

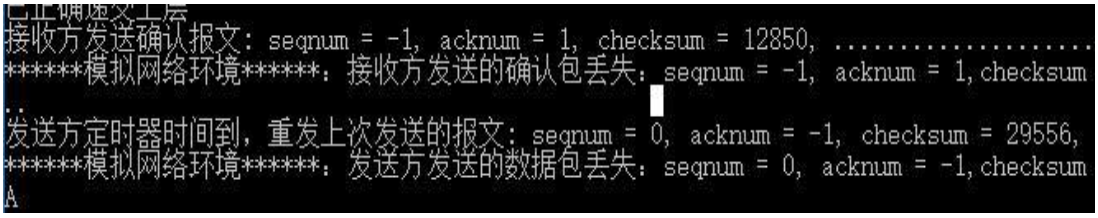

图 2.1.2GBN 测试结果

从图中可以看出由于超时，发送方会重发上次的报文，超时的原因又两种，一种是数据报并没有在规定时间到达接收方，一种是接收方已经收到数据报，但是返回的信息并没有及时到达发送方。

然后观察由于数据报丢失时接收方和发送方的动作：

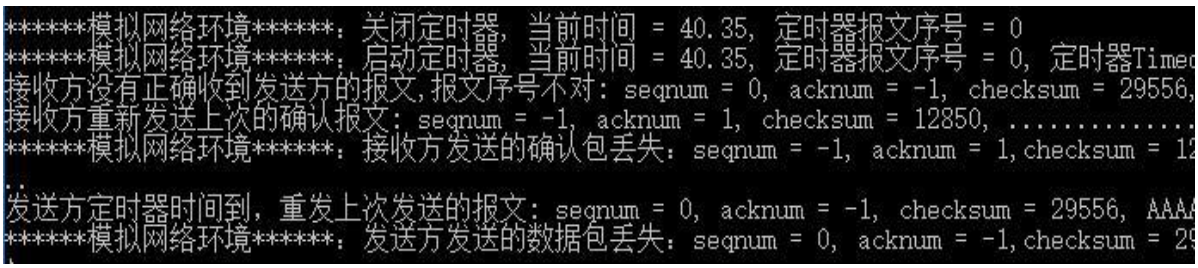

图 2.1.3GBN 测试结果

从图中可以看到此时重传的原因是接收方接收到的发送方的的报文序号不对，这样会导致从新传输。

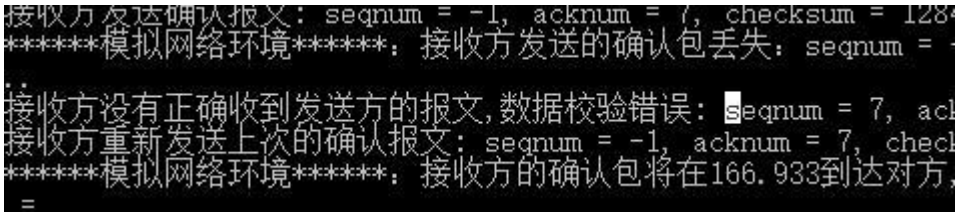

图 2.1.4GBN 测试结果

从图中看到此时重传的原因是接收方接收到数据报后计算的校验和与发送方的校验和不同。这些原因都将导致窗口不能往前滑动。

然后观察以下窗口滑动部分：

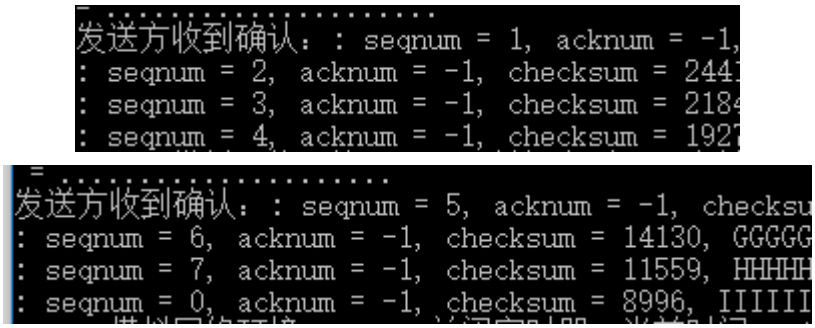

图 2.1.5GBN 协议测试

窗口的滑动限制在 0-7 之间，这是由于序号最大为 8 所导致的。滑动窗口的原因是窗口中 base 数据已经得到确认，序号可以往下挪动，由于使用的数据结构为快速队列，因此打印的四个数据报都是存放在 deque 的队列中。而打印这个窗口的前提是发送方收到来自接收方的数据报的确认信息且 cheksum 相同，此时会往后滑动窗口，并更新 deque 中的数据报内容。

接下来测试接收方收到正确数据提交上层后返回发送方但是将数据丢失的动作：

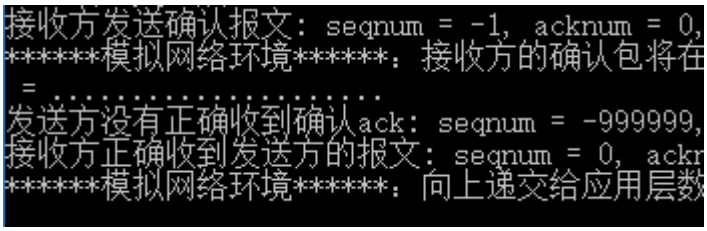

图 2.1.6GBN 协议测试

此时数据已经递交上层，所以发送方将什么都不做进行下一步操作。

经过分析，GBN 协议正确，实验符合预期。

2.3.2 SR 协议的设计、验证及结果分析

✧ SR 协议设计结果：

SR 协议在 GBN 协议的基础上进行了改进，它通过让发送方仅重传那些它怀疑在接收方出错（即丢失或受损）的分组而避免了不必要的重传。选择重传的接收窗口与发送窗口一样大. 选择重传协议允许与接收窗口一样多的分组失序到达，并保存这些失序到达的分组，直到连续的一组分组被交付给应用层. 因为发送窗口与接收窗口是相同的，所以发送出来的所有分组都可以失序到达，而且会被保留直到交付为止. 但是必须强调一点，在一个可靠的协议中，接收方永远不会把分组失序地交给应用层. 在他们被交付给应用层之前，先要等待那些更早发出来的分组到达。也正是这个原因，SR 协议中需要设计两个 deque 的数据结构，一个用来发送数据，一个用来接收方缓存数据，缓存的数据需要等到这些数据有序或者队列的第一个数据是 expcseqnum 时才会递交上层。另外需要注意的一点时，SR 协议中需要为每个分组设计一个计时器，因为不会直接重传，在 SR 中只会重传那些没有正确确认的分组. **理**论上选择重传协议

要为每个分组使用一个计时器.当某个计时器超时后,只有相应的分组被重传.换而言之,返回 N 协议将所有的分组当做一个整体对待,而选择重传协议则分别对待每一个分组.但是大多数 SR 的运输层仅使用了一个计时器. 注意只使用一个计时器而做到跟踪所有发出去的分组的情况的做法是:标记发出分组,当 ACK=Sf 时,将窗口滑过所有连续的已确认的分组,如果还有未确认的分组,则重发所有检测到的未被确认的分组并重启计时器,如果所有分组都被确认了则停止计时器。整个发送过程中其余操作与 GBN 协议中没有太大改变，每次确认分组后需要更新需要，发送方需要进行窗口的滑动。同时接收方也需要进行窗口的滑动。

✧ SR 协议结果验证与结果分析

以下按步骤对实验结果的各个部分进行简单的分析验证：

首先队最终结果进行分析：

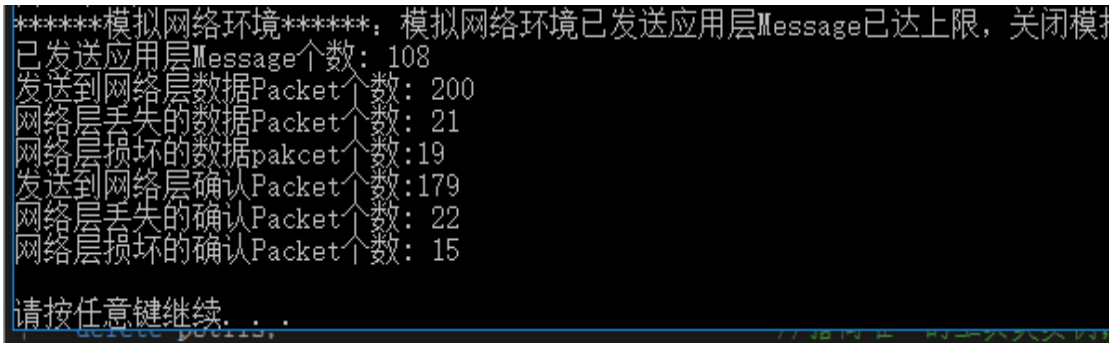

图 2.2.1SR 协议测试

整个数据报中的数据全部递交之后显示数据个数为 108，但是结果正确，这一点我觉得奇怪，后来观察代码发现应该是最初发送的问题，每次发送第一个数据报时由于之前的确认信息有过初始化，因此会导致递交上层的次数稍微有些变化，但这并不影响最终结果。其余信息是因为网络层的不可靠数据传输导致的数据报丢失的情况等内容。

然后对发送方发送数据时由于超时原因做出的动作进行分析：

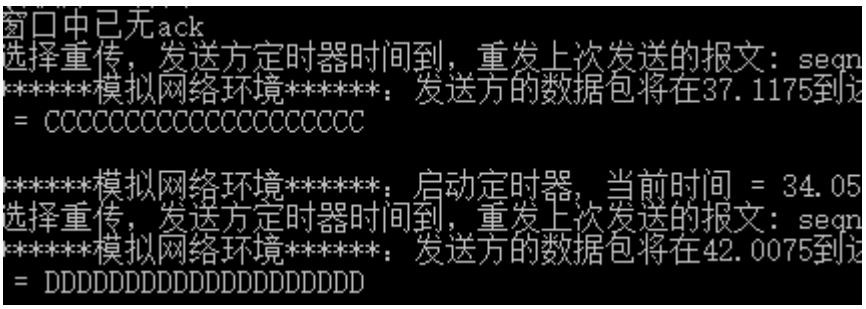

图 2.2.2SR 协议测试

数据报发送到接受方会发生超时事件，超时原因又两种，一种时数据报并没有在规定时间到达接收方，一种是接收方已经收到数据报，但是返回的信息并没有及时到达发送方。此处将两种情况归为一类处理，因为此时的超时事件导致报文需要重传，因此直接在这里将数据重新发一遍。

然后观察接收方收到数据报后的动作：

图 2.2.3SR 协议测试

此时显示数据成功递交上层，因为接收方正确收到了数据报且这个数据报为期望收到的数据报，所以会直接递交上层。同时需要更新 deque，下次到来的正确数据报放在队首，而此时由于已经递交过上层，所以就算发送方的计时器超时也会再次发送数据报，但是接收方不会再次递交。这个结果如下所示：

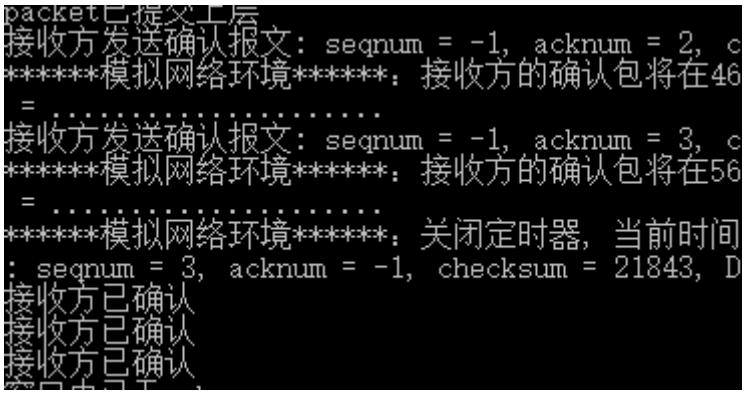

图 2.2.4SR 协议测试

这里显示接收方已经确认，说明信息已经递交上层，但是发送方还是会发送数据，不过不会再次递交。此时有一点需要注意，在这个过程中由于接收方也是一个 deque，因此还需要判断重发的分组是否在窗口内部，如果不在则会忽略。这是因为 sender 和 receiver 的窗口是互不可见的。

接下来验证窗口滑动部分：

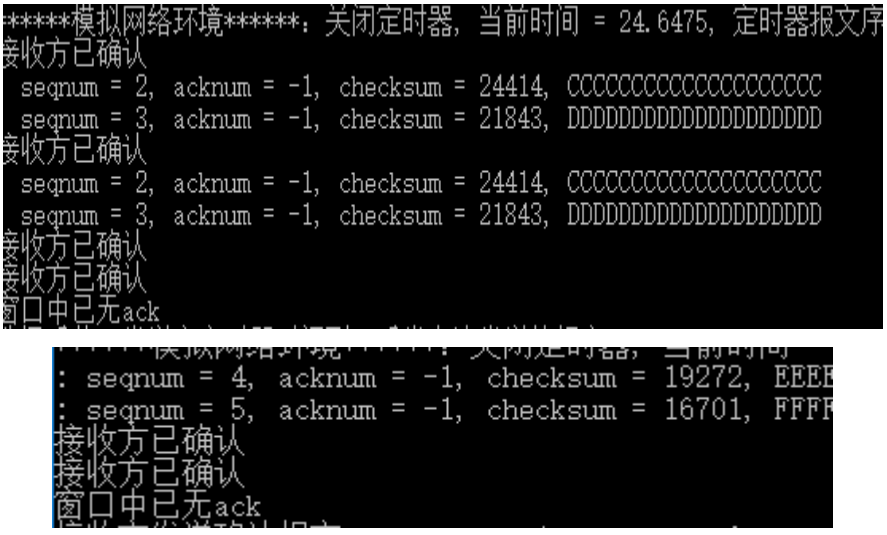

图 2.2.5SR 协议测试

从上图可以看出，这里打印了窗口大小为 4 的窗口中的数据信息，在循环过程中会更新窗

口，首先第一个数据报已经确认，第二三个数据报还未确认，因此在下一个循环中会更新队列，说明此时窗口已经向后滑动。下一个图片是下一次的队列信息。

接下来验证选择重传的部分：

选择重传出现在 1.数据发送到接收方时由于数据报出问题，队列中不是 base 的信息出错，2 由于超时，信息出错：

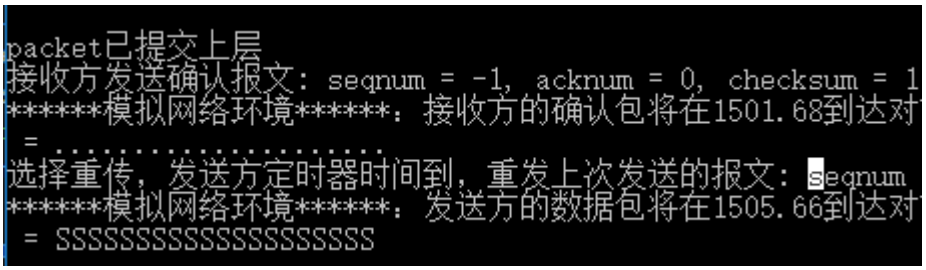

图 2.2.6SR 协议测试

从图中可以看到选择重换的发送信息，此时重传可能由上述两个原因中任意一个导致。最后还需声明一点，就是在最后一个报文发送结束后也要停止定时器，否则会导致错误。

协议验证正确，与预期一致。

2.3.3 简单 TCP/IP 协议的设计、验证及结果分析

✧ TCP 协议设计结果

快速重传是在 GBN 协议上进行增加的，当发送方接收到接收方的信息后会有一个比对过程，这个过程比对序号是否相同，由于 GBN 协议是一个累计确认的过程，因此在快速重传的判断中只需要判断基序号的下一个序号是否与接收方发送过来的数据报的序号相等，如果不想等的话，计数器就会自增 1.同时发送方会重发这个报文，最严重的情况是这部分报文连续三次都没有收到正确的确认信息，这有两个原因，一是这个数据报连续三次都在网络层发生错误，还有一个原因是由于发送方发送的数据报并不是接收方期望收到的数据报文。这是说明这之前都一个报文已经发生错误。需要进行重传，其他情况按 GBN 协议处理，比如只出现 2 次错误则不会使用快速重传。需要注意的是在进行快速重传时，确保计时器已经关闭，再重新打开计时器。当进行快速重传后计数器做归 0 处理。

✧ TCP 协议结果验证和分析

以下对快速重传协议进行简单的结果测试和分析结果的准确性。

首先是变量的更换，在构造函数中需要新增一个变量用来计数收到错误 ACK 的次数。因为 Packet 等数据封装都是 const 类型，所以这个变量只能在构造函数中初始化，在 GBN 协议基础上没有太大的变化，只是增加了这部分内容，首先观察最终结果的正确性：

图 2.3.1TCP 协议测试

从结果可以看出，总计发送 105 个数据报，最后报文为 EOF，其余数据内容是由于网络层的不可靠数据传输引起的报文丢失而进行的重传等数据内容。

然后观察由于数据报丢失时接收方和发送方的动作：

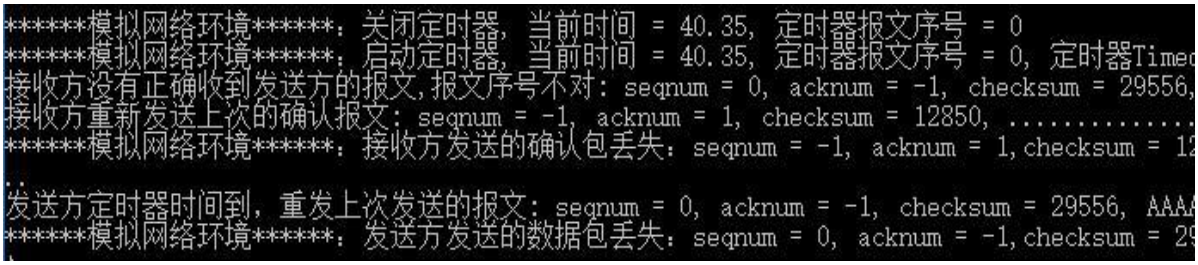

图 2.3.2TCP 测试结果

从图中可以看到此时重传的原因是接收方接收到的发送方的的报文序号不对，这样会导致从新传输。

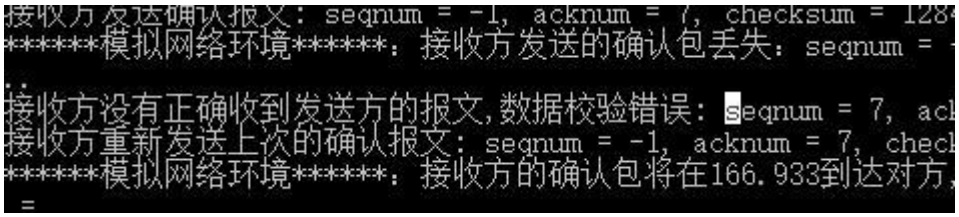

图 2.3.3TCP 测试结果

从图中看到此时重传的原因是接收方接收到数据报后计算的校验和与发送方的校验和不同。这些原因都将导致窗口不能往前滑动。

从图中看到此时重传的原因是接收方接收到数据报后计算的校验和与发送方的校验和不同。这些原因都将导致窗口不能往前滑动。

然后观察以下窗口滑动部分：

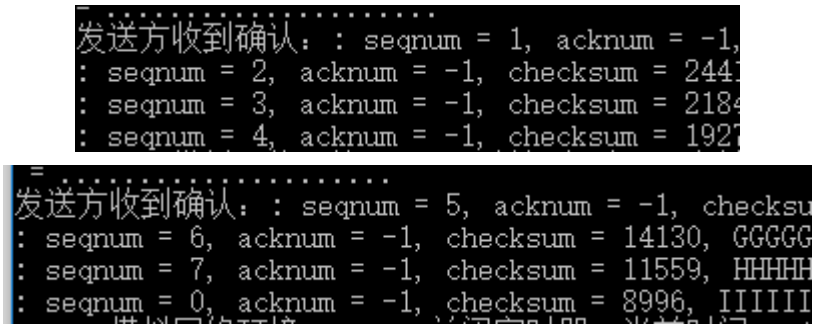

图 2.3.4TCP 协议测试

窗口的滑动限制在 0-7 之间，这是由于序号最大为 8 所导致的。滑动窗口的原因是窗口中 base 数据已经得到确认，序号可以往下挪动，由于使用的数据结构为快速队列，因此打印的四个数据报都是存放在 deque 的队列中。而打印这个窗口的前提是发送方收到来自接收方的数据报的确认信息且 cheksum 相同，此时会往后滑动窗口，并更新 deque 中的数据报内容。

接下来测试接收方收到正确数据提交上层后返回发送方但是将数据丢失的动作：

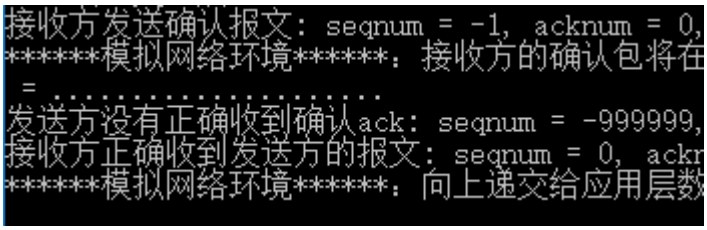

图 2.3.5TCP 协议测试

此时数据已经递交上层，所以发送方将什么都不做进行下一步操作。

最后观察快速重传的正确性：

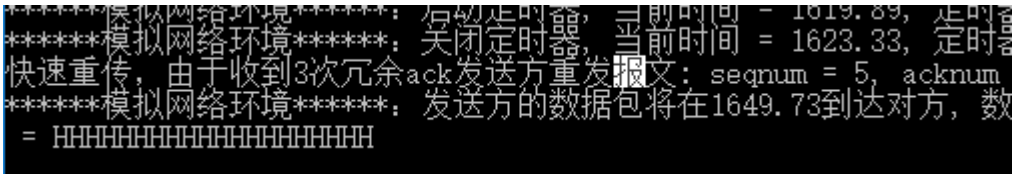

图 2.3.6TCP 协议测试

在这里可以看出由于计数器计数到 3 时会进行快速冲床，首先会关闭定时器，为了防止定时器没关带来的影响，然后再次开启定时器，发送数据到网络层。检查前面是否有三次连续发送错误：

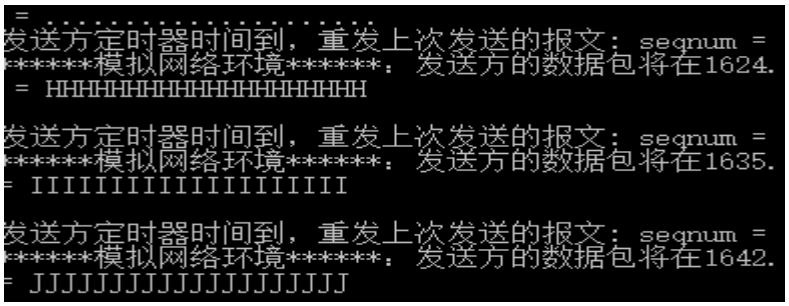

图 2.3.7TCP 协议测试

最后使用检查脚本检查两个文件的差异：

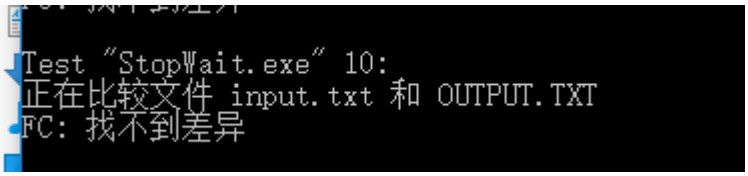

图 2.3./TCP 协议测试

从图中可以看到由于连续三次发生错误才会使用快速重传，其余与 GBN 协议一致。

综上所述，TCP 协议正确，实验结果与预期一致。

2.4 其它需要说明的问题

✧ 首先明确所使用的可靠数据传输协议，它们大同小异，但是在细微的处理上必须注意严谨处理，特别是搞清楚确认号和序号的区别，发送方发送数据的函数 sender 和接收数据的函数 receiver 的区别。

✧ 在对接受方的处理时序号的移动也可以在发送方中处理，在接收方收到正确信息后可以先不增加 1，直接返回发送方，发送方在下一次传输前在增加 1 之后才进行传输。

✧ SR 协议的实现需要 GBN 协议作为基础，而且 SR 协议中需要对每个数据报进行计时控制。还需要在接收方增加一个队列用来保存收到的正确但不是需要的数据报。

✧ 最后 TCP 协议的实现虽然做的比较简单，但是如果想要做的复杂的话就比较难了，需要增加慢启动步骤。不过庆幸的是实验过程中不需要计算时延来更新定时器的 timeout 时间。

实验三 基于 CPT 的组网实验

3.1 环境

3.1.1 开发平台

处理器： Intel® Core™ i5-7200 CPU @ 2.50GHz 2.70GHz

内存大小： 8.00 GB (7.85GB 可用)

操作系统： Windows 10 家庭中文版

3.1.2 运行平台

处理器： Intel® Core™ i5-7200 CPU @ 2.50GHz 2.70GHz

内存大小： 8.00 GB (7.85GB 可用)

操作系统： windows 10 中文家庭版

第三方组件： Cisco packet tracer 6.0 汉化版（增加中文语言包）

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。
- ✧ 提交实验设计报告纸质档和电子档。
- ✧ 基于自己的实验设计报告，通过实验课的上机实验，演示给实验指导教师检查。
- ✧ 实验内容

  本部分实验为基础部分的实验，分为两项内容，每项实验内容在最终的评价中占比 30%，本部分实验将使用两张拓扑结构图配合完成实验，如图 1.1 和 1.2 所示。

  路由器 A 交换机 1 交换机 2 交换机 3 交换机 4 PC1 PC2 PC3 PC4 PC5 PC7 PC8 PC6

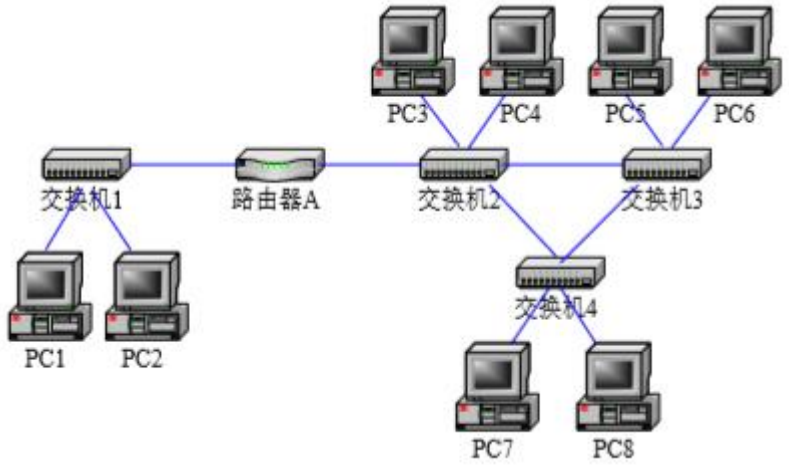

图 3.1

路由器 A 交 换机 1 交换机 2 交换机 3 路由器 B 路由器 D 路由器 C PC1 PC2 PC4 PC3

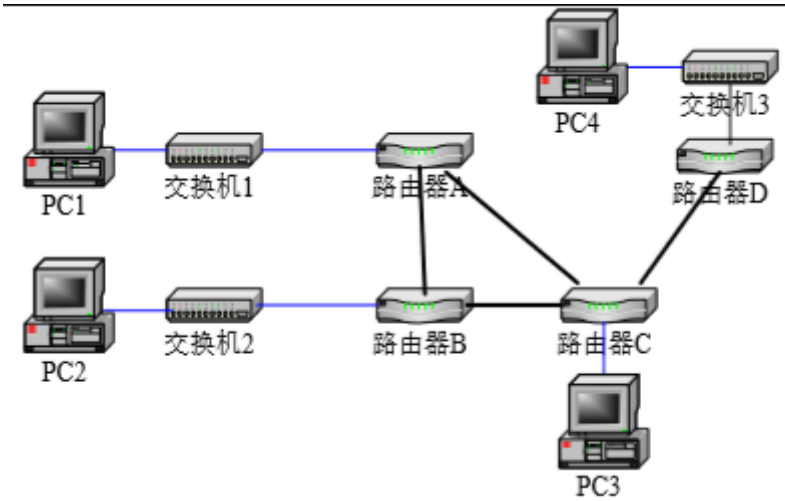

图 3.2

第一项实验——IP 地址规划与 VLan 分配实验：

✧ 使用仿真软件描述网络拓扑图 1.1。

✧ 基本内容 1

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

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

✧ 配置路由器，使得两个子网的各 PC 机之间可以自由通信。

✧ 基本内容 2

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

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

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

✧ 配置交换机 1、2、3、4，使得 PC1、PC2 属于 Vlan2，PC3、PC5、PC7 属于 Vlan3，PC4、PC6、

PC8 属于 Vlan4；

✧ 测试各 PC 之间的连通性，并结合所学理论知识进行分析；

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

详细说明。

第二项实验——路由配置实验

- ✧ 使用仿真软件描述网络拓扑图 1.2
- ✧ 基本内容 1
- ✧ 将 PC1 设置在 192.168.1.0/24 网段；
- ✧ 将 PC2 设置在 192.168.2.0/24 网段；
- ✧ 将 PC3 设置在 192.168.3.0/24 网段；
- ✧ 将 PC4 设置在 192.168.4.0/24 网段
- ✧ 设置路由器端口的 IP 地址
- ✧ 在路由器上配置 RIP 协议，使各 PC 机能互相访问
- ✧ 基本内容 2
- ✧ 将 PC1 设置在 192.168.1.0/24 网段；
- ✧ 将 PC2 设置在 192.168.2.0/24 网段；
- ✧ 将 PC3 设置在 192.168.3.0/24 网段；
- ✧ 将 PC4 设置在 192.168.4.0/24 网段
- ✧ 设置路由器端口的 IP 地址
- ✧ 在路由器上配置 OSPF 协议，使各 PC 机能互相访问
- ✧ 基本内容 3
- ✧ 在基本内容 1 或者 2 的基础上，对路由器 1 进行访问控制配置，使得 PC1 无法访问其它 PC，也不能被其它 PC 机访问。
- ✧ 在基本内容 1 或者 2 的基础上，对路由器 1 进行访问控制配置，使得 PC1 不能访问 PC2，但能访问其它 PC 机

（综合部分）

本部分实验为综合部分的实验，在最终的评价中占比 40%。

实验背景：

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

组网需求：

- ✧ 图书馆能够无线上网
- ✧ 学院之间可以相互访问
- ✧ 学生宿舍之间可以相互访问
- ✧ 学院和学生宿舍之间不能相互访问
- ✧ 学院和学生宿舍皆可访问图书馆。

实验任务要求：

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

的问题)

- ✧ 根据理论课的内容，对全网的 IP 地址进行合理的分配
- ✧ 在绘制的网络拓扑结构图上对各类设备进行配置，并测试是否满足组网需求，如有无法满足之处，请结合理论给出解释和说明

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

✧ 对于组网实验的第一个子实验，首先按照网络拓扑图选择设备并将设备拖 拽到主界面中，此处选择的终端设备为台式计算机 PC-PT，路由器型号为 1941， 交换机型号为 Switch-PT。本着不同设备之间用直通线，同种设备之间用交叉 线的原则，计算机与交换机之间选用直通线，交换机与交换机之间选择交叉线， 交换机与路由器之间选用直通线按照拓扑图进行连接。 在连接完成后双击各个计算机并按照要求分别为其分配静态 ip 地址。对于图 3.3 中的设备，PC1 与 PC2 分配的 ip 地址为 192.168.0.101，PC3~PC8 分配 的 ip 地址为 192.168.1.100 到 192.168.1.105 的连续地址（PC3 分配 192.168.1.100PC4 分配 192.168.1.101，以此类推）。并将路由器左边的 PC 的 Gateway 配置为 192.168.0.1，路由器右边 PC 的 Gateway 配置为 192.168.1.1。

连接完成后测试是否成功：

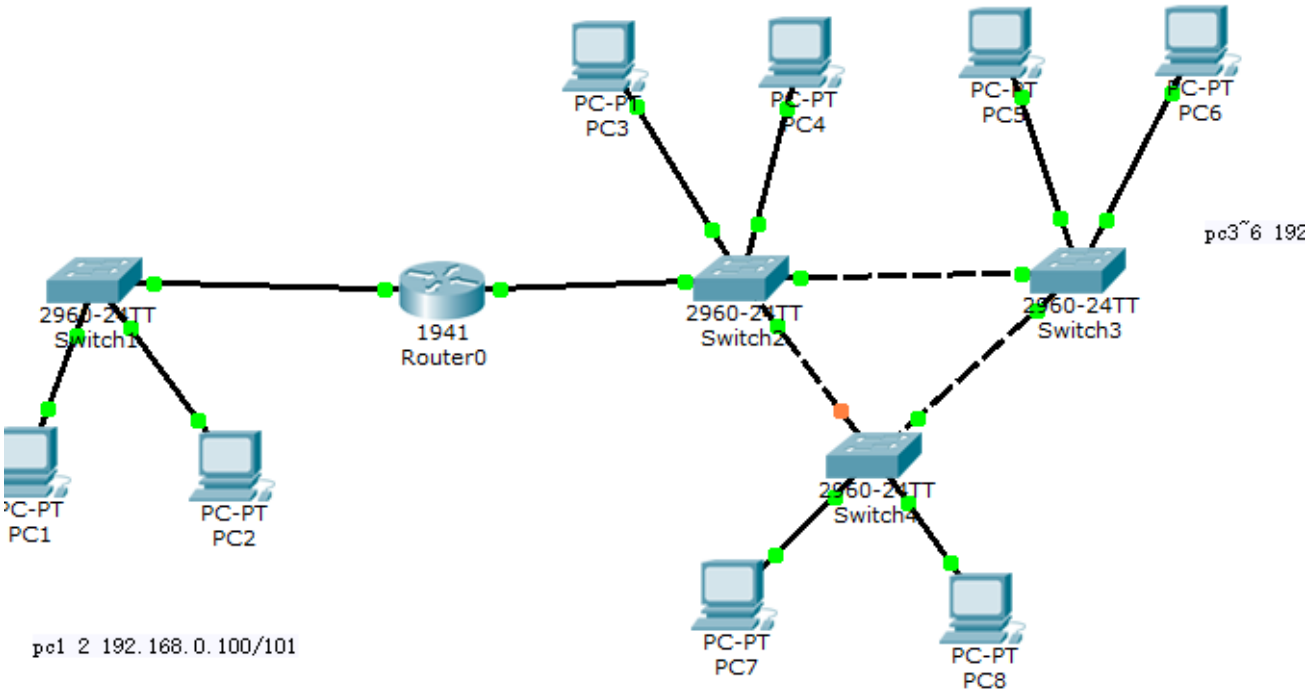

图 3.1.1 连接结果

观察路由器的配置情况：

```
端口                    链路     VLAN    IP地址                IPv6地址
GigabitEthernet0/0      Up       --      192.168.0.1/24        <not set>
GigabitEthernet0/1      Up       --      192.168.1.1/24        <not set>
Vlan1                   Down     1       <not set>             <not set>
主机名称:Router

物理位置:城际，城市家园，公司办公室，配线橱，机架
```

图 3.1.2 连接结果

路由器包含两个端口，两个端口连接了 2 个网段。

使用添加简单的 PDU 测试是否联通：

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	PC7	ICMP		0.000	N	0	(编辑)	(删除)
	成功	PC2	PC8	ICMP		0.000	N	1	(编辑)	(删除)
	成功	PC7	PC3	ICMP		0.000	N	2	(编辑)	(删除)
	成功	PC8	PC4	ICMP		0.000	N	3	(编辑)	(删除)
	成功	PC6	PC8	ICMP		0.000	N	4	(编辑)	(删除)
	成功	PC8	PC1	ICMP		0.000	N	5	(编辑)	(删除)
	成功	PC3	PC6	ICMP		0.000	N	6	(编辑)	(删除)

图 3.1.3 测试结果

从图中可以看出结果正确。所有主机均可 ping 通。

✧ Vlan 分配的第二个子实验中，将 PC3，PC5，PC7 划分在一个 vlan，PC2，PC6，PC8 划分在一个 vlan，将 PC1，PC2 划分在一个 vlan 在配置路由器之前，由于他们属于不同的 vlan，因此他们之间是不能互通的，此时在路由器上 分配端口，因为路由器右边包括了 2 个 vlan，而之连接了 1 个端口，所以将端口划分：

```
端口                  链路    VLAN    IP地址              IPv6地址
FastEthernet0/0       Up      --      192.168.0.1/24      <not set>
FastEthernet0/1       Up      --      <not set>           <not set>
FastEthernet0/1.1     Up      --      192.168.1.1/24      <not set>
FastEthernet0/1.2     Up      --      192.168.2.1/24      <not set>
Vlan1                 Down    1       <not set>           <not set>
主机名称:Router1
```

图 3.1.4 端口划分

划分之后还需要注意的是由于左边已经和右边相连接，所以左边要想通信则需要将端口划分为 access。此时它们之间可以画像通信，使用添加简单的 PDU 测试是否联通：

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	PC7	ICMP		0.000	N	0	(编辑)	(删除)
	成功	PC1	PC7	ICMP		0.000	N	1	(编辑)	(删除)
	成功	PC2	PC8	ICMP		0.000	N	2	(编辑)	(删除)
	成功	PC3	PC8	ICMP		0.000	N	3	(编辑)	(删除)
	成功	PC2	PC5	ICMP		0.000	N	4	(编辑)	(删除)
	成功	PC2	PC6	ICMP		0.000	N	5	(编辑)	(删除)
	成功	PC6	PC7	ICMP		0.000	N	6	(编辑)	(删除)

图 3.1.5 测试结果

从图中可以看出从任何主机都可以 ping 通其他主机，测试结果正确。与预期一致。

3.3.2 路由配置实验的步骤及结果分析

✧ 对于路由配置实验的第一个子实验，连接与实验 1 类似，只是注意一下，路由器之间的连线采用 DEC 串口线，配置端口时设置时钟频率为 64000。

图 3.1.6 时钟设置

将 PC1,PC2,PC3,PC4 划分在不同的网段，为了让他们能够互相通信，需要配置 rip 协议，配置方法可以直接打开路由器输入 rip 的 ip 进行配置，下面以 router0 为例简单讲一下 rip 协议的配置：

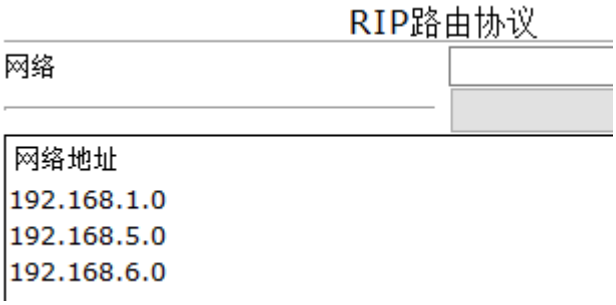

图 3.1.7rip 协议配置

因为 router0 连接了 2 个路由器和自身的一个子网，所以 rip 中网络地址分为 3 部分，一本分路由到 router1，一部分路由到 router2.还有一部分时在子网中路由。连接完成之后进行简单的 ping 测试：

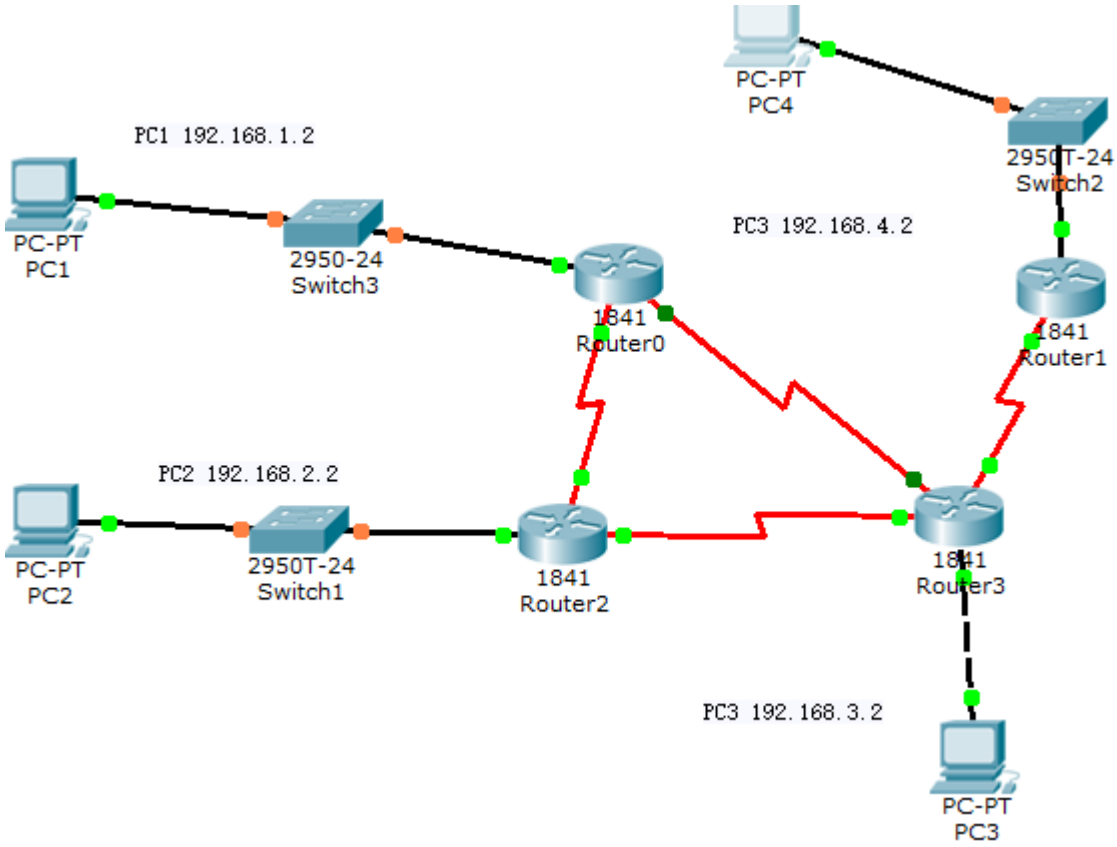

图 3.1.8 实验连接

从图中可以很清楚看出 ip 的分配情况，各个 PC 的 ip 分别为 192.168.1/2/3/4.2.测试结果如

下：

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

图 3.1.9 测试结果

从图中看出，无论从哪个主机取 ping 其他主机都是可以 ping 通的，由此也间接说明 rip 协议配置正确

✧ 在路由地址配置实验的第二个子实验中，需要我们配置 ospf 协议使得各个主机之间可以相互通信，各个部件之间的连接变化不大，只需在原来基础上去掉 rip 协议，添加 ospf 协议，而 ospf 协议的添加需要在路由器上采用命令行添加，在此以 router0 为例，在命令行采用 show 命令展示 router0 上配置的 ospf 协议：

```
!
router ospf 1
 log-adjacency-changes
 network 192.168.1.0 0.0.0.255 area 0
 network 192.168.5.0 0.0.0.255 area 0
 network 192.168.6.0 0.0.0.255 area 0
!
```

图 3.1.10ospf 协议

从图中可以看出由于 router0 配置了 ospf 协议，所以与之有关的路由器上配置的信息也有所展示。此处使用的命令是在 enable 下使用 show running-config。这个实验中还需要在另外 3 个路由器上配置 ospf 协议，所采用的方法相同，在此处不再赘述。

其他设计与前一个子实验相同，然后可以使用 ping 命令进行测试，测试结果如下：

用主机 1ping 主机 4：

```
PC>ping 192.168.4.2

Pinging 192.168.4.2 with 32 bytes of data:

Reply from 192.168.4.2: bytes=32 time=12ms TTL=125
Reply from 192.168.4.2: bytes=32 time=2ms TTL=125
Reply from 192.168.4.2: bytes=32 time=2ms TTL=125
Reply from 192.168.4.2: bytes=32 time=2ms TTL=125

Ping statistics for 192.168.4.2:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 2ms, Maximum = 12ms, Average = 4ms
```

他 3.1.11 实验测试

从图中看出结果正确，可以 ping 通，同样在测试主机二 ping 主机 4：

```
PC>ping 192.168.3.2

Pinging 192.168.3.2 with 32 bytes of data:

Reply from 192.168.3.2: bytes=32 time=1ms TTL=126
Reply from 192.168.3.2: bytes=32 time=2ms TTL=126
Reply from 192.168.3.2: bytes=32 time=2ms TTL=126
Reply from 192.168.3.2: bytes=32 time=1ms TTL=126

Ping statistics for 192.168.3.2:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 1ms, Maximum = 2ms, Average = 1ms
```

图 3.1.12 结果测试

从图中可以看出，可以 ping 通，还可以直接采用简单 PDU 测试：

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	PC4	PC1	ICMP		0.000	N	1	(编辑)	(删除)
	成功	PC3	PC4	ICMP		0.000	N	2	(编辑)	(删除)
	成功	PC2	PC3	ICMP		0.000	N	3	(编辑)	(删除)

图 3.1.13 结果测试

结果显示所有主机都可以 ping 通，与预期一致。

✧ 在路由地址配置实验的第 3 个子实验中，要求我们配置 ACL 指令实现主机 1 与其他主机无法互通，而其他主机之间可以通信，实验所需要的器件与前一个实验一致，在 ospf 实验的基础上进行配置，配置 ACL 需要我们采用命令行配置连接号图之后打开 router0 进行配置：配置结果如下：

在命令行中使用 show 命令显示 acl 如下：

```
Router#show access-lists
Standard IP access list 10
    deny 192.168.1.0 0.0.0.255
    permit any
Router#
00:00:10: %OSPF-5-ADJCHG: Process 1, Nbr 192.168.7.1 on Serial0/0/0 from LOADING
 to FULL, Loading Done

00:00:10: %OSPF-5-ADJCHG: Process 1, Nbr 192.168.8.2 on Serial0/0/1 from LOADING
 to FULL, Loading Done
```

图 3.1.14ACL

从图中可以看出配置的 acl 阻止了主机 1 与其他主机之间的通信，所以使用简单 PDU 测试如下：

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)
	失败	PC1	PC4	ICMP		0.000	N	1	(编辑)	(删除)
	失败	PC1	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	PC3	ICMP		0.000	N	4	(编辑)	(删除)

图 3.1.15 测试结果

从图中可以直接看出主机 1 与其他主机通信都将失败，而其他主机之间的互通却可以成功。结果正确与预期一致。

✧ 在路由地址配置实验的第 4 个子实验中，我们可以直接配置 router1 是主机 1 与主机 2 不能互通，而主机 1 与其他主机之间可以直接互通，其他连接与前面实验一致，这里不再赘述。这个实验中需要配置扩展 ACL 指令，具体指令经过百度查找之后得出。

在 router0 输入 show 指令显示配置的扩展 acl

```
    10 deny 192.168.2.0 0.0.0.255
Extended IP access list 101
    deny ip 192.168.1.0 0.0.0.255 host 192.168.2.2
    permit ip any any
    deny ip 192.168.1.0 0.0.0.255 192.168.2.0 0.0.0.255
Extended IP access list 103
    deny icmp 192.168.1.0 0.0.0.255 host 192.168.2.0 echo
    permit ip any any
Router#
```

图 3.1.16 扩展 acl

从图中可以看出已经配置了 acl，连接完成之后可以直接开始测试：

DU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	PC3	ICMP		0.000	N	1	(编辑)	(删除)
	成功	PC1	PC4	ICMP		0.000	N	2	(编辑)	(删除)
	成功	PC3	PC4	ICMP		0.000	N	3	(编辑)	(删除)

图 3.1.17 测试结果

从图中看出，配置了 acl 之后可以 ping 通 pc3 和 pc4，但 hi 是不与 PC2 通，说明实验结果正确，与预期一致。

3.4 综合部分实验设计、实验步骤及结果分析

3.4.1 实验设计

✧ 首先考虑 ip 地址的分配和子网的划分。经过一番考虑，最终划分的子网如下图所示：

IP 211.69.7.129-229
mask 225.255.255.128
valn4

IP 211.69.7.1-80
mask 255.255.255.128
vlan 2

学院　　图书馆

IP 211.69.4/5/6.1
mask 255.255.255.0
vlan 3

宿舍

图 3.1.17IP 地址划分

按照上述构建，
构建好各部分信息后整体电路如下：

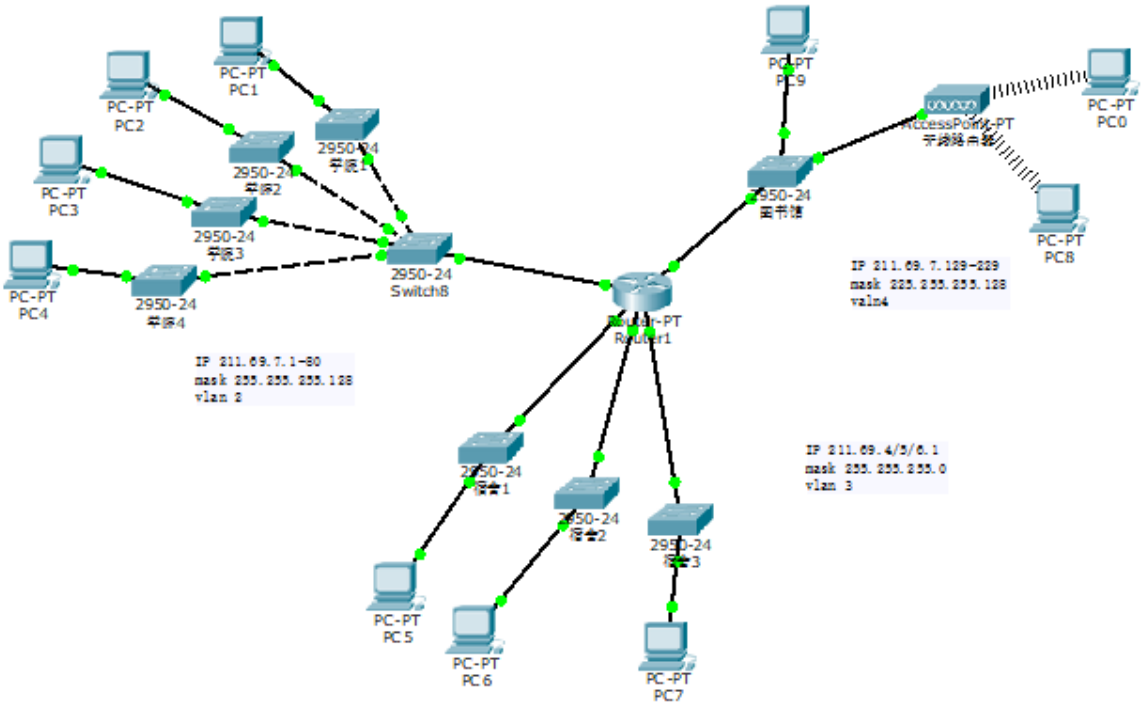

图 3.1.21 整体构造

3.4.2 实验步骤

✧ 这是按照各个部门的人数来分配的，给学院和图书馆个设置一个 vlan 进行管理，同时因为这两处

的主机数总共不超过 255 态，因此划分在同一个网段，在配置协议时需要注意这一点。其次宿舍区总共需要 600 台主机，因此进行分配时可以考虑分配为 3 个网段，也可以分配 3 个 vlan，但是此处我只分配了 1 个 vlan，这样操作起来比较简单，而且也直接实现了实验的整个要求。各个交换机上的信息如下所示：

端口	链路	VLAN	IP地址
FastEthernet0/1	Up	--	--
FastEthernet0/2	Up	--	--
FastEthernet0/3	Up	--	--
FastEthernet0/4	Up	--	--
FastEthernet0/5	Up	--	--

图 3.1.18 学院交换机信息

端口	链路	VLAN	IP地址
FastEthernet0/1	Up	--	--
FastEthernet0/2	Up	3	--

图 3.1.19 宿舍交换机信息

24

端口	链路	VLAN	IP地址
FastEthernet0/1	Up	--	--
FastEthernet0/2	Up	4	--
FastEthernet0/3	Up	4	--

图 3.1.20 图书馆交换机信息

✧ 画出了部分主机的结构图，由于一个叫交换机有 20 个以上的端口，所以完全连接满足题目要求是

可以实现的。连接之后由于只有一个路由器，因此不需要配置 rip 协议或者 ospf 协议，观察路由器的端口分配情况，由于宿舍区有三个网段，因此需要分端口：

端口	链路	IP地址	IPV6地址
FastEthernet0/0	Up	<not set>	<not set>
FastEthernet0/0.1	Up	211.69.7.1/25	<not set>
FastEthernet1/0	Up	<not set>	<not set>
FastEthernet1/0.1	Up	211.69.4.1/24	<not set>
FastEthernet2/0	Up	<not set>	<not set>
FastEthernet2/0.1	Up	211.69.5.1/24	<not set>
FastEthernet3/0	Up	<not set>	<not set>
FastEthernet3/0.1	Up	211.69.6.1/24	<not set>
FastEthernet4/0	Up	<not set>	<not set>
FastEthernet4/0.1	Up	211.69.7.129/25	<not set>

✧

✧ 图 3.1.22 端口分配

具体的配置信息如下：

```
interface FastEthernet0/0
 no ip address
 ip access-group 101 in
 duplex auto
 speed auto
!
interface FastEthernet0/0.1
 encapsulation dot1Q 2
 ip address 211.69.7.1 255.255.255.128
 ip access-group 101 in
!
interface FastEthernet1/0
 no ip address
 duplex auto
 speed auto
```

```
!
interface FastEthernet1/0.1
 encapsulation dot1Q 3
 ip address 211.69.4.1 255.255.255.0
 ip access-group 102 in
!
interface FastEthernet2/0
 no ip address
 duplex auto
 speed auto
!
interface FastEthernet2/0.1
 encapsulation dot1Q 3
 ip address 211.69.5.1 255.255.255.0
 ip access-group 103 in
!
interface FastEthernet3/0
 no ip address
 duplex auto
 speed auto
!
interface FastEthernet3/0.1
 encapsulation dot1Q 3
 ip address 211.69.6.1 255.255.255.0
 ip access-group 104 in
!
interface FastEthernet4/0
 no ip address
 duplex auto
 speed auto
!
interface FastEthernet4/0.1
 encapsulation dot1Q 4
 ip address 211.69.7.129 255.255.255.128
!
router rip
!
ip classless
!
!
access-list 102 deny ip 211.69.4.0 0.0.0.255 211.69.7.0 0.0.0.127
access-list 102 permit ip any any
access-list 101 deny ip 211.69.7.0 0.0.0.127 211.69.4.0 0.0.0.255
access-list 101 deny ip 211.69.7.0 0.0.0.127 211.69.5.0 0.0.0.255
```

access-list 101 deny ip 211.69.7.0 0.0.0.127 211.69.6.0 0.0.0.255
access-list 101 permit ip any any
access-list 103 deny ip 211.69.5.0 0.0.0.255 211.69.7.0 0.0.0.127
access-list 103 permit ip any any
access-list 104 deny ip 211.69.6.0 0.0.0.255 211.69.7.0 0.0.0.127
access-list 104 permit ip any any

3.4.3 结果分析

配置好上面各部分内容之后可以进行 ping 测试，首先测试宿舍和学院是否互通：

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	PC7	PC4	ICMP		0.000	N	0	(编辑)	(删除)
	失败	PC7	PC3	ICMP		0.000	N	1	(编辑)	(删除)
	失败	PC2	PC6	ICMP		0.000	N	2	(编辑)	(删除)
	失败	PC1	PC6	ICMP		0.000	N	3	(编辑)	(删除)

图 3.1.23 结果测试

如图所示，因为配置了 acl，学院和宿舍是不能互通的满足实验要求。
然后测试学院和图书馆，宿舍和图书馆能否互通：

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	PC0	PC5	ICMP		0.000	N	4	(编辑)	(删除)
	成功	PC0	PC7	ICMP		0.000	N	1	(编辑)	(删除)
	成功	PC6	PC0	ICMP		0.000	N	5	(编辑)	(删除)
	成功	PC6	PC9	ICMP		0.000	N	3	(编辑)	(删除)
	成功	PC7	PC0	ICMP		0.000	N	2	(编辑)	(删除)
	成功	PC9	PC7	ICMP		0.000	N	0	(编辑)	(删除)

图 3.1.24 测试结果

结果显示图书馆和宿舍时可以相互通信的。满足实验要求
继续测试学院与图书馆：

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	PC0	PC2	ICMP		0.000	N	1	(编辑)	(删除)
	成功	PC1	PC9	ICMP		0.000	N	0	(编辑)	(删除)
	成功	PC8	PC3	ICMP		0.000	N	2	(编辑)	(删除)
	成功	PC8	PC4	ICMP		0.000	N	3	(编辑)	(删除)
	成功	PC8	PC4	ICMP		0.000	N	4	(编辑)	(删除)
	成功	PC9	PC3	ICMP		0.000	N	5	(编辑)	(删除)

图 3.1.25 测试结果

结果显示图书馆和学院之间也可以相互通信，结果正确。
综合以上结果，表示实验结果正确，符合预期。

3.5 其它需要说明的问题

- ✧ 首先在配置第一项实验的第二个子实验时需要注意将路由器左边的宽口划分为 access，否则将导

致 PC1 和 PC2 无法与右边的主机进行通信。

- ✧ 然后是注意实验 2 的第二个子实验在进行时，需要先去掉之前实验配置的 rip 协议，否则就算配置

成功了 ospf 协议也会导致最后结果正确，无法验证 ospf 协议。

- ✧ 还需要注意的是第二个子实验的最后一个实验使用了扩展 acl 指令，我有思考过使用 acl 指令，

但是一直没有结果。之后可以尝试一下。

- ✧ 还有一点就是综合实验中可以适当添加路由器，更改为 3 个路由器，我这种做法虽然结果正确，

但是感觉整个学校只是用一个路由器有点过分了。之所以这样做，是实验要求尽量节省资源。

- ✧ 最后整个实验的难点在于要学习 cisco 的使用，实验本身并不复杂，只是学习一个软件的使用需要一个过程，所以感到困难。

心得体会与建议

4.1 心得体会

- ✧ 通过套接字实验我体会到了套接字在网络数据传输中的作用和整个流程，明白了传输过程的各个步骤的意义和用途，也明白了请求报文和响应报文的格式，读取文件时需要注意的问题，除此之外和初步了解了多线程的使用方法，只是对于图形界面这一块还没有太多的理解，希望在以后的实验中可以更加有所收获。
- ✧ 通过可靠数据传输实验我体会到了运输层进行可靠数据传输的各个协议的重要性，也从这个过程中学会了运用可靠数据传输进行数据的传送。领悟到了 GBN 和 SR 协议之间的不同之处和相同点。实验二相比实验一相对简单，我认为主要原因在于实验一是我们之前从未接触过网络实验，需要一个适应的过程。其次，通过第二次实验我也更加明白了运输层协议在整个网络协议中的重要地位。
- ✧ 通过第三次实验我初步学会了使用 Cisco 这个软件，对于其中命令行的使用也有了一些理解。建立主机间通信是一件比较有趣的事，其次是这次实验过程真正体会到了 rip 协议和 ospf 协议的作用，以及在路由器见进行相互访问的作用。整个实验过程中最难的地方在于学会使用 Cisco 这款软件，学会之后就可以很简单的完成这次实验，这次实验给了一个模拟环境，让我收获颇丰。

4.2 建议

- ✧ 首先对于套接字编程，我觉得实验教程讲解的不够准确，还有就是建立套接字这个过程确实需要查阅很多资料才能完全明白，除此之外我觉得助教检查的时候应该多问一些问题，而不是由我们区讲，助教问了之后我们才会明白哪些地方还没有考虑到，因此大家第一次接触到这种类型的实验，还是很复杂的。
- ✧ 其次，我认为实验二是非常好的一次实验，但是他没有过多的加入实验一的内容在里面，如果要完成还是需要非很大的劲的。还有就是我觉得这个实验比较好的一点在于首先给出了停等协议的实现，让我们有了一个可以观察的样例，这一点我觉得比实验一好很多，实验一中那个样例似乎和实验关系不大。
- ✧ 第三次实验也是比较好的一个实验，只是第三次实验的时间比较尴尬，完成之后写报告需要花费很多时间，如果可以的话，我觉得可以将第三次实验的时间稍微往前提一点。完成之后已经接近期末了，感觉时间上比较紧迫。如果可以的话希望把时间稍微往前提一周。

以上是我对于这门实验课程的一些建议，也希望计网这门课程越来越好。