
华中科技大学计算机学院

《计算机通信与网络》实验报告

姓名刘子煜 班级计卓 1601 学号U201612922

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	3
1.5 系统测试及结果说明	6
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
实验三 基于 CPT 的组网实验	28
3.1 环境	28
3.2 实验要求	28
3.3 基本部分实验步骤说明及结果分析	30
3.4 综合部分实验设计、实验步骤及结果分析	42
3.5 其它需要说明的问题	47
心得体会与建议	48
4.1 心得体会	48
4.2 建议	48

实验一 Socket 编程实验

1.1 环境

Intel Core i7-6700HQ CPU 2.6GHz

RAM 16GB

Windows10 1803 (实验机器的硬件配置、系统软件组件、第三方软件)

Cisco Packet Tracer Version: 7.2.0.0226

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，系统功能需求如下：

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

系统架构

本 Web 服务器系统分为两个部分：客户端和服务端。客户端主要为用户浏览器，客户端发送连接请求给服务器，服务器创建连接套接字，通过 Socket 套接字服务器与客户端进行通讯和数据交互。

本系统负责服务器端，首先要对客户端发来的连接请求作出处理，一个线程处理一个连接；然后对客户端发来的请求报文进行解析，根据请求报文的内容生成客户端所需要的响应报文，将响应报文发送给浏览器。

功能模块划分

根据功能的不同主要将服务器端分为 5 个模块，其中包括定位虚拟路径，设置监听地址和监听端口，多线程创建连接，接收并解析请求报文，生成并发送响应报文。系统功能模块图如图 1.1 所示。

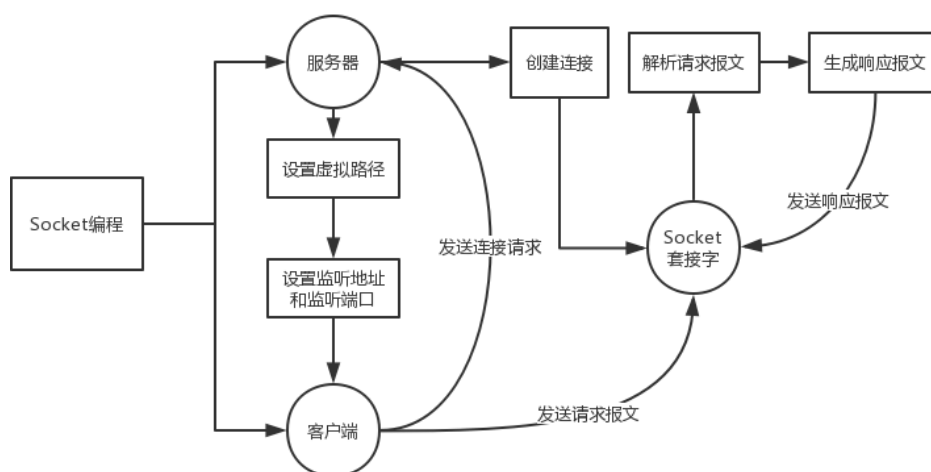


图 1.1 系统功能模块图

设置虚拟路径模块：通过命令行交互让服务器管理员设置服务器资源的虚拟路径，先给出提示信息让管理员输入资源的绝对路径。

设置监听地址和监听端口模块：通过命令行交互让服务器管理员设置服务器的监听地址和监听端口，在初始化过程中，会将绑定监听地址和监听端口。

创建连接模块：不断等待客户端的连接请求，请求到来后，为新的客户建立新的 socket 套接字，同时创建一个新的线程来负责通过这个新的 socket 套接字来实现服务器和客户端的通讯。

解析请求报文模块：收到客户端发来的请求报文后，对其内容进行解析，通过解析 URL 来定位客户端所需要的资源。

生成响应报文模块：将客户端所需要的资源装入响应报文中，发送给客户端。

1.4 系统实现

系统框架

网络应用程序是由通信进程对组成，每对互相通信的应用程序进程互相发送报文，他们之间的通信必须通过下面的网络来进行。为了将应用程序和底层的网络通信协议屏蔽开来，采用套接字（Socket）这样一个抽象概念来作为应用程序和底层网络之间的应用程序编程接口（API）。

因为网络应用程序是进程之间的通信，为了唯一的标识通信对等方的通信进程，套接字必须包含 2 种信息：

通信对等方的网络地址。

通信对等方的进程号，通常叫端口号。

就像 Unix 操作系统下有一套实现 TCP/IP 网络通信协议的开发接口：BSD Sockets 一样，在 Windows 操作系统下，也提供了一套网络通信协议的开发接口，简称 Winsock。

在 Winsock 里，用数据类型 SOCKET 作为 Windows Sockets 对象的句柄，就好像一个窗口的句柄 HWND、一个打开的文件的文件指针一样。在 Winsock API 的许多函数里，都会用到 SOCKET 类型的参数。

Socket 有 2 种类型：

流类型：流式套接字提供了一种可靠的、面向连接的数据传输方法，使用传输控制协议 TCP。

数据报类型：数据报套接字提供了一种不可靠的、非连接的数据包传输方式，使用用户数据报协议 UDP。

在本次实验中采用流类型的套接字，首先建立连接，而后才能从数据流中读出数据。其服务器端流程如图 1.2 所示。

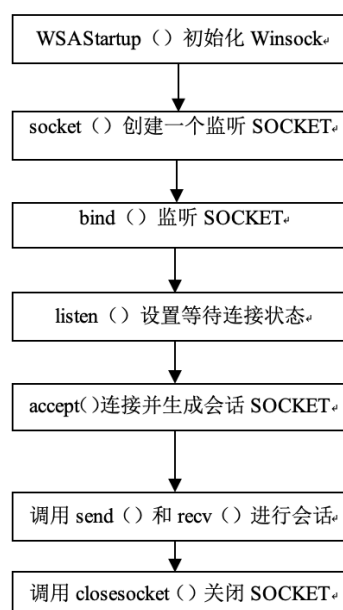


图 1.2 流类型套接字服务器端流程图

一个 SOCKET 句柄可以看成代表了一个 I/O 设备。在 Windows Sockets 里，有 2 种 I/O 模式：

阻塞式 I/O：在阻塞方式下，收发数据的函数在调用后一直要到传送完毕或者出错才能完成，在阻塞期间，除了等待网络操作的完成不能进行任何操作。阻塞式 I/O 是一个 Winsock API 函数的缺省行为。

非阻塞式 I/O：对于非阻塞方式，Winsock API 函数被调用后立即返回；当网络操作完成后，由 Winsock 给应用程序发送消息（Socket Notifications）通知操作完成，这时应用程序可以根据发送的消息中的参数对消息做出响应。Winsock 提供了 2 种异步接受数据的方法：一种方法是使用 BSD 类型的函数 `select ()`，另外一种方法是使用 Winsock 提供的专用函数 `WSAAsyncSelect ()`。

在本次实验中采用非阻塞式 I/O，通过 `select()` 函数来接收连接请求和数据。

模块设计

创建连接模块

根据监听端口和监听地址创建监听 socket，负责服务器和客户端的连接请求，通过该监听 socket 服务器等待连接请求，将该 socket 放入文件描述符中，通过 `select` 函数返回可读可写的 socket 数量来判断是否有连接请求到来，如果有请求到来，则为该用户创建新的会话 socket，创建成功后，为该会话建立新的线程，该线程负责监听该会话 socket 并根据接收到的消息作出反应。流程图如图 1.3 所示。

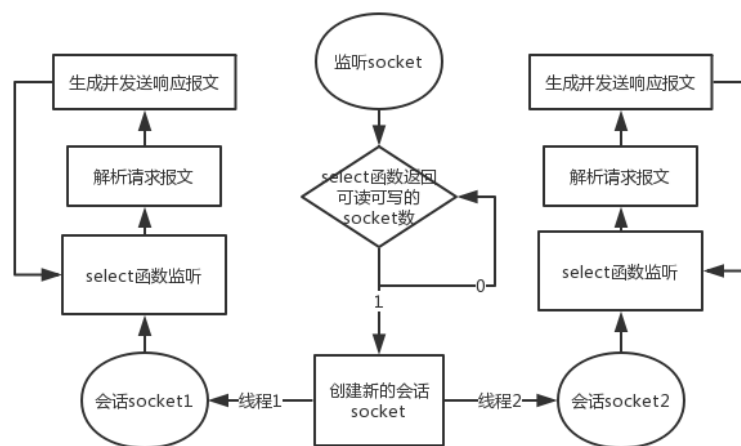


图 1.3 创建连接流程图

解析请求报文模块

通过会话 socket 接收到客户端发来的请求报文后，进行解析。Http 请求报文的具体结构如图 1-4 所示。

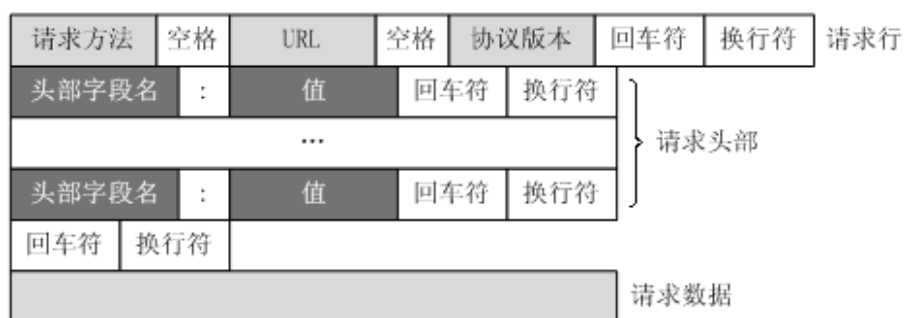


图 1-4 请求报文结构图

一个 HTTP 请求报文由请求行（request line）、请求头部（header）、空行和请求数据 4 个部分组成。

请求行由请求方法字段、URL 字段和 HTTP 协议版本字段 3 个字段组成，它们用空格分隔。

例如，GET /index.html HTTP/1.1。

GET 为最常见的一种请求方式，当客户端要从服务器中读取文档时，当点击网页上的链接或者通过在浏览器的地址栏输入网址来浏览网页的，使用的都是 GET 方式。GET 方法要求服务器将 URL 定位的资源放在响应报文的数据部分，回送给客户端。

根据请求报文的结构设计相应的解析程序，解析请求报文的请求行，得到相应的请求方式和 URL 字段。

生成请求报文模块

HTTP 响应报文由三部分组成：响应行、响应头、响应体。其结构图如图 1-5 所示



图 1.5 响应报文结构图

响应行一般由协议版本、状态码及其描述组成 比如 HTTP/1.1 200 OK

其中协议版本 HTTP/1.1 或者 HTTP/1.0, 200 就是它的状态码, OK 则为它的描述。

常见状态码:

100~199: 表示成功接收请求, 要求客户端继续提交下一次请求才能完成整个处理过程。

200~299: 表示成功接收请求并已完成整个处理过程。常用 200

300~399: 为完成请求, 客户需进一步细化请求。例如: 请求的资源已经移动一个新地址、常用 302(意味着你请求我, 我让你去找别人), 307 和 304(我不给你这个资源, 自己拿缓存)

400~499: 客户端的请求有错误, 常用 404(意味着你请求的资源在 web 服务器中没有) 403(服务器拒绝访问, 权限不够)

500~599: 服务器端出现错误, 常用 500

响应头用于描述服务器的基本信息, 以及数据的描述, 服务器通过这些数据的描述信息, 可以通知客户端如何处理等一会儿它回送的数据。

1.5 系统测试及结果说明

测试过程

设置虚拟路径, 根据提示信息, 输入虚拟路径的实际位置。如图 1-6 所示。

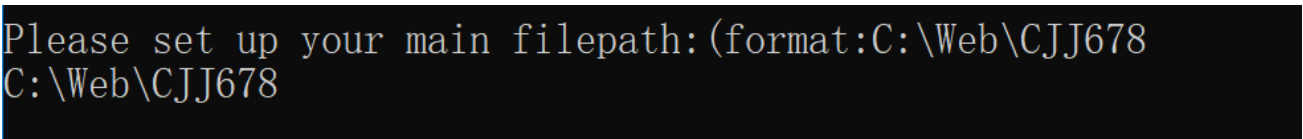


图 1.6 虚拟路径测试图

设置监听地址和监听端口，根据提示信息，输入监听地址和监听端口。如图 1-7 所示。

```
Winsock startup ok!  
Input IP and Port  
127.0.0.1 5050
```

图 1.7 监听设置测试图

根据监听地址和监听端口与 server socket 绑定并监听，负责接收客户端的连接请求。如图 1-8 所示。

```
Server socket create ok!  
Server socket bind ok!  
Server socket listen ok!  
ioctlsocket() for server socket ok!Waiting for client connection and data
```

图 1.8 开启服务器测试图

浏览器发送连接请求，服务器设置并开启后，浏览器输入指定的 URL，发送连接请求报文给服务器。如图 1-9 所示。



图 1.9 发送连接请求测试图

服务器创建连接，服务器收到连接请求后，为该用户建立新的 socket 套接字和线程，并收到请求报文服务器收到请求报文后，解析出请求报文的类型和所需资源的虚拟地址。如图 1-11 所示。

```
GET /index.html HTTP/1.1  
Host: 127.0.0.1:5050  
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/67.0.3396.99 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: zh-CN,zh;q=0.9  
IP: 127.0.0.1:14924  
Command:GET /index.html HTTP/1.1
```

图 1.10 收到请求报文测试图

服务器发送响应报文服务器生成响应报文发送给客户端，浏览器正常显示内容浏览器收到所有所需的资源后正常显示内容。如图 1-11 所示。

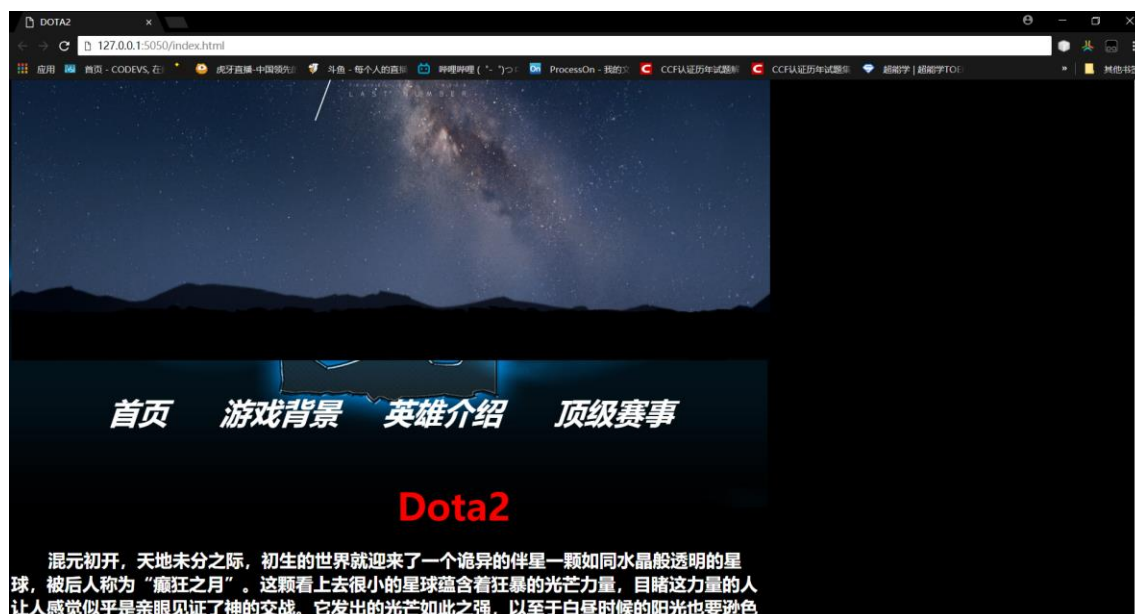


图 1.11 正常显示内容测试图

结果分析

当浏览器正常显示内容后，本次实验功能需求的第一级完整完成，第二级需要实现多线程和输出所有请求的来源，第三级需要实现传输多媒体网页和错误信息提示。根据实验结果来分析是否完成第二级和第三级的功能。

多线程实现和输出请求来源

由于浏览器跟服务器建立的不是持续性连接，所以会有多个连接，每个连接对应一个端口号，每一个连接对应着一个线程，从图中可以观察到，多个连接同时在接收请求报文和发送响应报文，所以实现多线程成功，同时也将所有请求的来源都输出成功。如图 1-12 所示。

```

Thread start
GET /index.html HTTP/1.1
Host: 127.0.0.1:5050
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/67.0.3396.99 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: zh-CN,zh;q=0.9

IP: 127.0.0.1:14975
Command:GET /index.html HTTP/1.1
-----
Thread close
Thread start
Thread start
GET /tupian/chatu.jpg HTTP/1.1
Host: 127.0.0.1:5050
Connection: keep-alive
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/67.0.3396.99 Safari/537.36
Accept: image/webp,image/apng,image/*,*/*;q=0.8
Referer: http://127.0.0.1:5050/index.html
Accept-Encoding: gzip, deflate, br
Accept-Language: zh-CN,zh;q=0.9

GET /tupian/biaoti.jpg HTTP/1.1
Host: 127.0.0.1:5050
Connection: keep-alive
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/67.0.3396.99 Safari/537.36
Accept: image/webp,image/apng,image/*,*/*;q=0.8
Referer: http://127.0.0.1:5050/index.html
Accept-Encoding: gzip, deflate, br
Accept-Language: zh-CN,zh;q=0.9

```

图 1.12 多线程结果分析图

传输多媒体网页

打开浏览器的开发者工具，观察该网页的网络连接，发现传输多个图片，实现传输多媒体网页成功。如图 1-13 所示。

Name	S...	Type	Initiator	Size	Time	Waterfall
ind...	...	document	Other	6.5 KB	59 ms	
bia...	...	jpeg	index.h...	1.3 MB	849 ms	
cha...	...	jpeg	index.h...	110 KB	146 ms	
bba...	...	jpeg	index.h...	195 KB	195 ms	
live...	...	script	livestar...	(from ...	2 ms	
favi...	(...		Other	0 B	Pending	

图 1.13 传输多媒体结果分析图

错误信息提示

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。如图所示，服务器中找不到该 URL，服务器发送类型为 404 的响应报文，客户端接收后显示“HTTP ERROR 404”的错误提示信息。如图 1-16 所示。

```
IP: 127.0.0.1:1147
Command:GET /inlex.html HTTP/1.1
-----
Cannot find the file!
HTTP/1.1 404 Not Found
```

图 1.14 错误信息提示结果分析图

至此，本次实验中所有的功能都已经实现，图形化界面未完成。

1.6 其它需要说明的问题

在对于无法定位文件时，应该发送响应类型为 404 的响应报文，而不是 200 OK。
学会有效运用 select 函数来实现监听和会话 socket 的控制响应。

实验二 数据可靠传输协议设计实验

2.1 环境

Intel Core i7-6700HQ CPU 2.6GHz

RAM 16GB

Windows10 1803 (实验机器的硬件配置、系统软件组件、第三方软件)

Cisco Packet Tracer Version: 7.2.0.0226

2.2 实验要求

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并给出确认报文。

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。

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。

实验内容

本实验包括三个级别的内容，具体包括：实现基于 GBN 的可靠传输协议，分值为 50%。实现基于 SR 的可靠传输协议，分值为 30%。在实现 GBN 协议的基础上，根据 TCP 的可靠数据传输机制（包括超时后只重传最早发送且没被确认的报文、快速重传）实现一个简化版的 TCP 协议。报文段格式、报文段序号编码方式和 GBN 协议一样保持不变，不考虑流量控制、拥塞控制，不需要估算 RTT 动态调整定时器 Timeout 参数。分值 20%。

2.3 协议的设计、验证及结果分析

2.3.1 GBN 协议的设计、验证及结果分析

GBN 协议设计结果

1) 基本参数设计

该实验中涉及了各层协议 Pay Load 数据的大小以及定时器时间等基本参数，在进行 GBN 具体设计之前需要参考《计算机通信与网络实验指导手册》选择合适的参数数值。应用层-运输层模型基本参数设置见表 2.1。

表 2.1

参数名称	数据类型	配置值	说明
PAYLOAD_SIZE	static const int	21	各层协议 PayLoad 数据大小
TIME_OUT	static const int	20	定时器时间

2) 数据结构设计

本实验中所设计的 GBN 设计为运输层可靠传输内容，为满足实验要求，需要设计运输层报文段结构；同时，为了模拟实际网络环境下的数据收发，还需设计应用层的消息结构。运输层报文段数据结构具体内容说明如表 2.2 所示。

表 2.2

参数名称	数据类型	说明
seqnum	int	首部序号
acknum	int	首部确认号
checksum	int	报文校验和
payload	char[]	PayLoad

具体到实现 GBN 重传方式时，需要设计 GBN 发送方数据结构和 GBN 接收方数据结构，上述两个 GBN 模型实体的具体参数信息的配置如表 2.3 和表 2.4 所示。

表 2.3

参数名称	数据类型	说明
base	int	最早未被确认或最初未发送的报文段序号
nextsequm	int	最早未发送报文段序号
timerseq	int	timer 的序号
N	const int	窗口的大小
pktQueue	vector<Packet>	窗口中的报文段队列

表 2.4

参数名称	数据类型	说明
expectSequenceNumberRcvd	int	期待收到的下一个报文序号
lastAckPkt	Packet	上次发送的确认报文

其中发送方的发送窗口结构示意图如图 2.1 所示



图 2.1

其中接收方的接收窗口结构示意图如图 2.2 所示

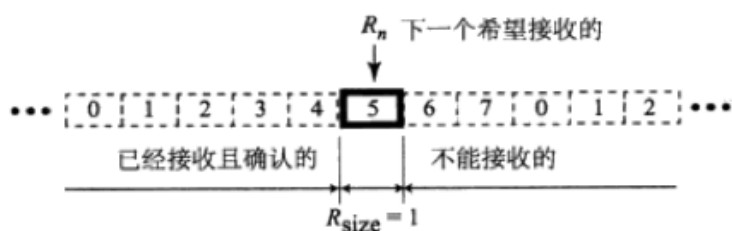


图 2.2

3) GBN 发送方逻辑设计

对于应用层发出的 RDT 调用请求，GBN 模式下发送方的响应行为可用流程图 2.3 描述。

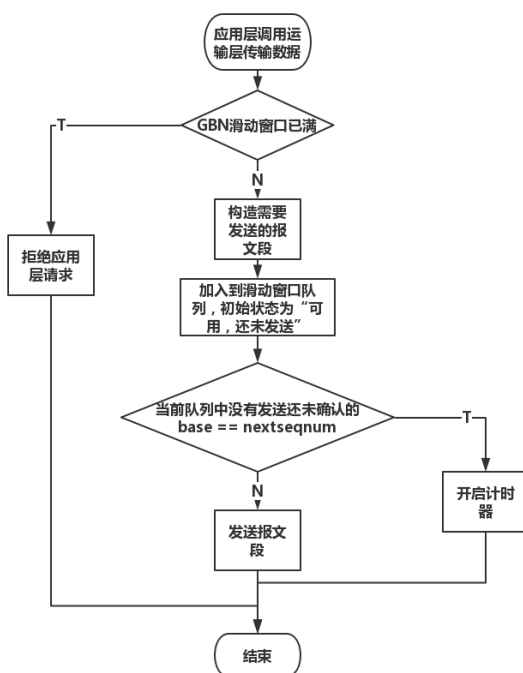


图 2.3

对于网络层发出的数据报接收请求，GBN 模式下的发送方的响应行为可用流程图 2.4 描述。

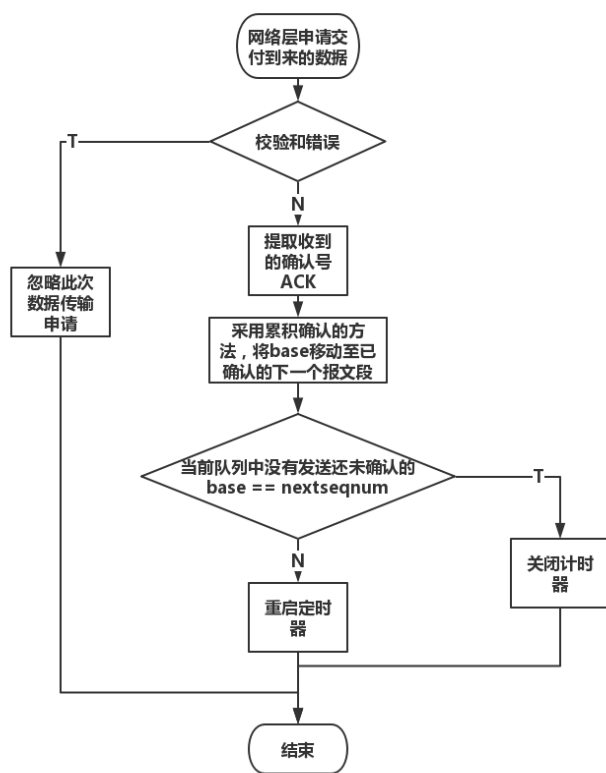


图 2.4

对于发送且未确认的报文段的超时事件。GBN 模式下的发送方的响应行为可用流程图 3 描述。

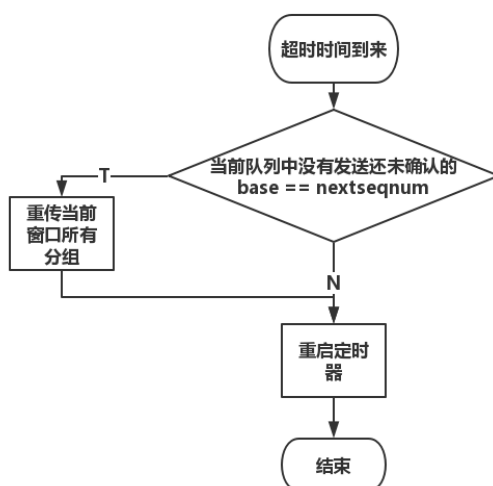


图 2.5

4) GBN 接收方逻辑设计

在本实验中，需要实现的部分是接收方状态机对于接收到低层网络层传输到来的报文段的处理。对于网络层发出的数据报接收请求，GBN 模式下的接收方的响应行为可用流程图 2.6 描述。

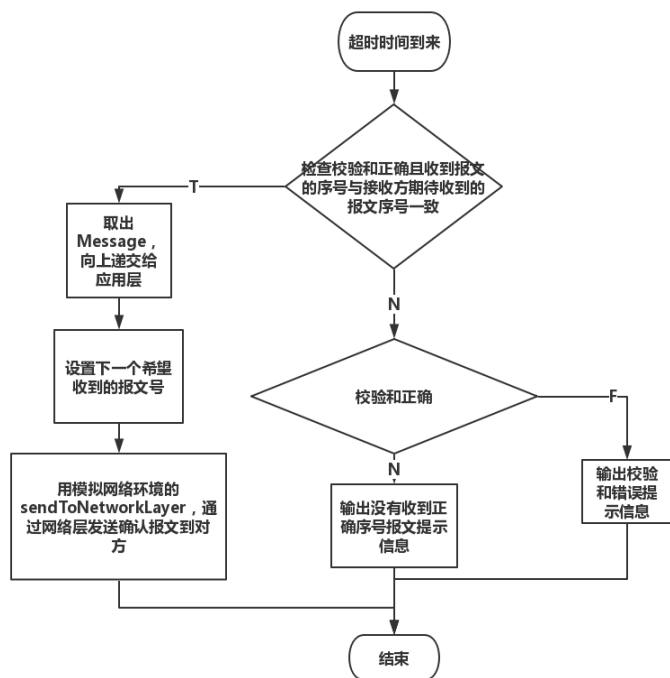


图 2.6

GBN 协议验证结果

在模拟网络环境下运行 GBN 模型下的 RDT 协议，得到验证结果如表 2.5 所示。

表 2.5

事件	接收方原滑动窗口	事件发生后接收方滑动窗口	发送方原滑动窗口	事件发生后发送方滑动窗口	额外操作
GBNRdtSender发送的报文段: seqnum = 1, acknum = -1, checksum = 29555, AAAAAAAAAAAAAAAAAAAAA	empty	empty	empty	1	无
接收方发送确认报文: seqnum = 0, acknum = 2, checksum = 12848,	2	empty	2345	2345	无

正确收到GBNRdtSender发来的报文段: seqnum = 0, acknum = 2, checksum = 12848,	empty	empty	2345	345	无
接收方没有正确收到发送方的报文,报文序号不对: seqnum = 4, acknum = -1, checksum = 21842, DDDDDDDDDDDDDDDDDDDD	empty	empty	345	345	接收方重新发送 上次的确认报文: seqnum = 0, acknum = 2, checksum = 12848
接收方没有正确收到发送方的报文, 数据校验错误: seqnum = 3, acknum = -1, checksum = 24413, DCCCCCCCCCCCCCCCCCCC	empty	empty	345	345	接收方重新发送 上次的确认报文: seqnum = 0, acknum = 2, checksum = 12848
*****模拟网络环境*****: 超时 启动定时器, 当前时间 = 328.665, 定时器报文序号 = 4, 发送方的数据包将在336.625到达对方, 数据包为-->seqnum = 4, 发送方的数据包将在340.765到达对方, 数据包为-->seqnum = 5,	empty	empty	45	45	发送方的数据包将在336.625到达对方, 数据包为-->seqnum = 4, 发送方的数据包将在340.765到达对方, 数据包为-->seqnum = 5,

使用检测脚本检测 GBN 协议的传输正确性得到的结果如图 2.7 所示

```

FC: 找不到差异

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

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

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

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

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

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

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

```

图 2.7 GBN 运行结果

GBN 结果分析

首先验证 GBN 协议传输的正确性, 由图脚本测试结果, 该 GBN 模拟实现在测试用例上实现了可靠数据传输。

其次再验证 GBN 接收方和 GBN 发送方对各个事件的响应是否正确, 根据表 2.5, GBN 接收方和发送方对各种事件均做出正确响应。

其中需要关注的是滑动窗口的滑动形式，GBN 规定的滑动窗口滑动方式和测试用例中滑动窗口的滑动方式相同。

另一个需要关注的部分是对当发生超时时间时发送方的重新发送方式。根据 GBN 的重发规则，每次出现超时现象重发的分组都是当前滑动窗口内的所有报文段。根据测试用例中的结果，重传方式符合预期。

1.3.2 SR 协议的设计、验证及结果分析

SR 协议设计结果

数据结构设计

本实验中所设计的 SR 为运输层可靠传输内容，为满足实验要求，需要设计运输层报文段结构；同时，为了模拟实际网络环境下的数据收发，还需设计应用层的消息结构。运输层报文段数据结构具体内容说明如表 2.6 所示。

表 2.6

参数名称	数据类型	说明
seqnum	int	首部序号
acknum	int	首部确认号
checksum	int	报文校验和
payload	char[]	PayLoad

具体到实现 SR 重传方式，需要设计 SR 发送方数据结构和 SR 接收方数据结构，上述两个 SR 模型实体的具体参数信息的配置如表 2.7 和表 2.8 所示。

表 2.7

参数名称	数据类型	说明
base	int	最早未被确认或最初未发送的报文段序号
nextsequm	int	最早未发送报文段序号
timerseq	int	timer 的序号
N	const int	窗口的大小
pktQueue	vector<Packet>	窗口中的报文段队列

表 2.8

参数名称	数据类型	说明
base	int	接收窗口起始处
nextSeqNum	int	下一个等待接收的报文段编号
expectSequenceNumberRcvd	int	期待收到的下一个报文序号
lastAckPkt	Packet	上次发送的确认报文

SR 发送方逻辑设计

SR 发送方和接收方的行为描述如图 2.8 所示。

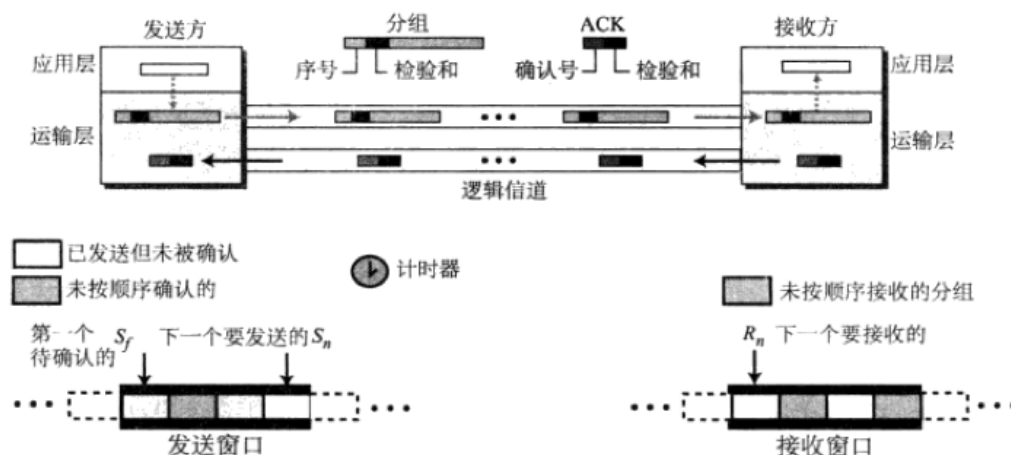


图 2.8

在本实验中，需要实现的部分是发送方状态机对于应用层请求调用运输层提供的可靠数据传输协议传输数据的响应、接收到低层网络层传输到来的报文段的处理，以及发送且未确认的报文段的超时事件响应。

对于应用层发出的 RDT 调用请求，SR 模式下发送方的响应行为可用流程图 2.9 描述。

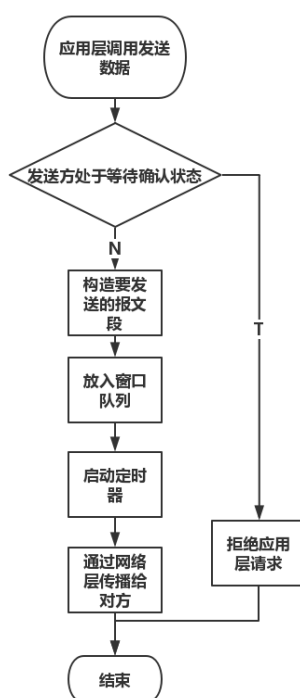


图 2.9

对于网络层发出的数据报接收请求，GBN 模式下的发送方的响应行为可用流程图 2.10 描述。

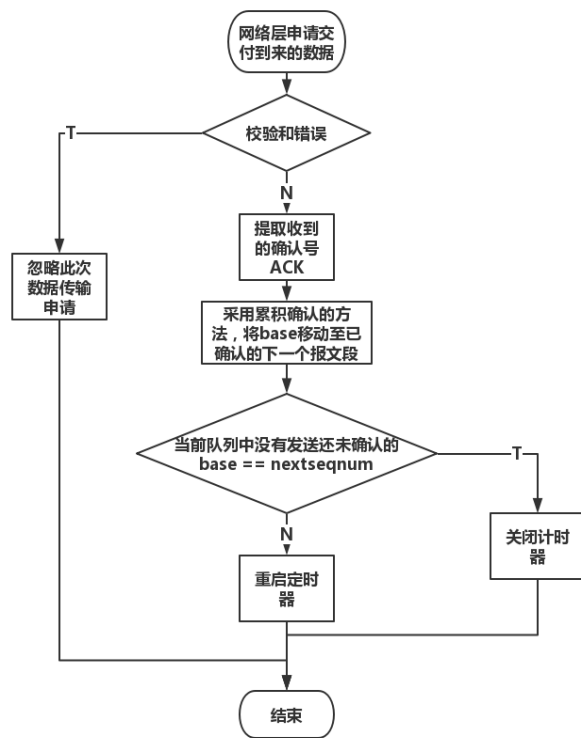


图 2.10

对于发送且未确认的报文段的超时事件。SR 模式下的发送方的响应行为可用流程图 2.11 描述。

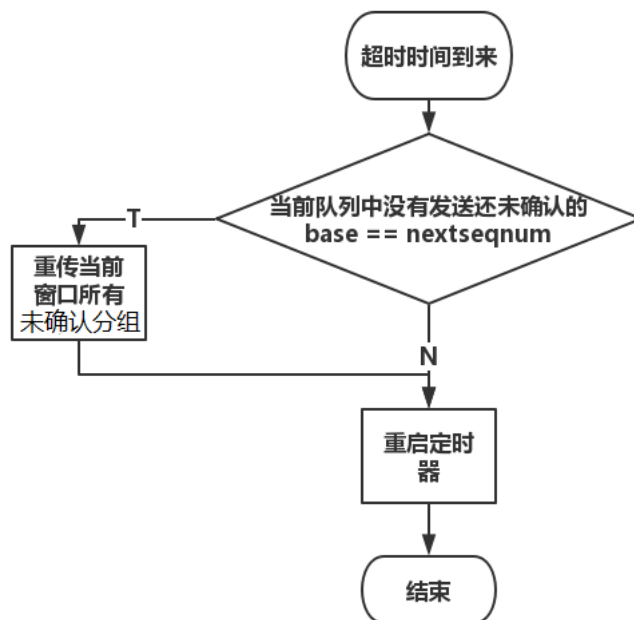


图 2.11

SR 接收方逻辑设计

在本实验中，需要实现的部分是接收方状态机对于接收到低层网络层传输到来的报文段的处理。对于网络层发出的数据报接收请求，GBN 模式下的接收方的响应行为可用流程图 2.12 描述。

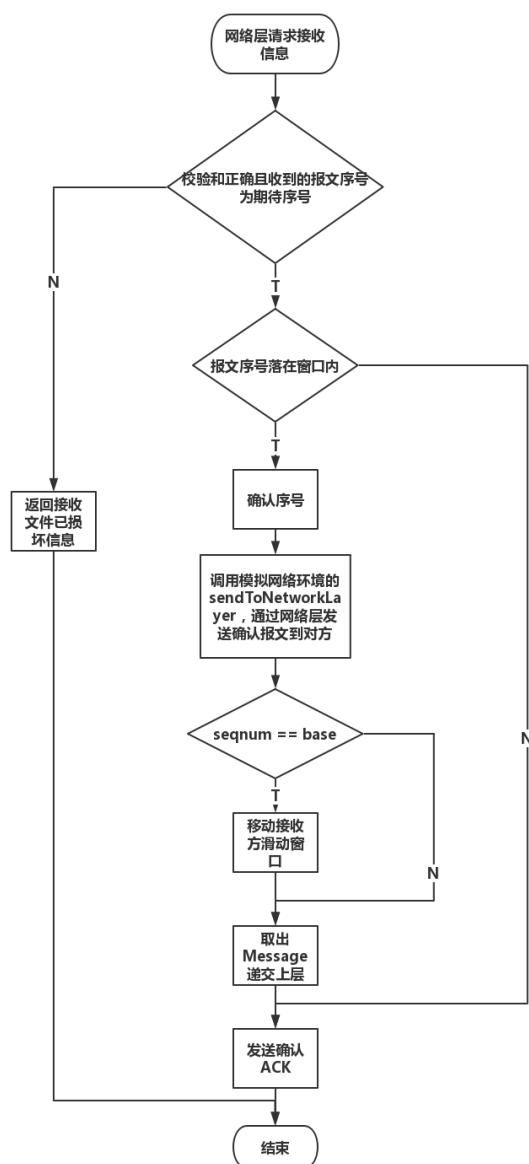


图 2.12

SR 协议验证结果

在模拟网络环境下运行 SR 模型下的 RDT 协议，得到验证结果如表 2.9 所示。

表 2.9

事件	接收方原滑动窗口	事件发生后接收方滑动窗口	发送方原滑动窗口	事件发生后发送方滑动窗口	额外操作
发送方发送报文: seqnum = 0, acknum = -1, checksum = 29556, AAAAAAAAAAAAAAAAAAAAA	0123	0123	empty	0	无
接收方发送确认报文: seqnum = 0, acknum = 2, checksum = 12848,	2345	3456	2345	2345	无
接收方没有正确收到发送方的报文,报文序号不对: seqnum = 4, acknum = -1, checksum = 21842, DDDDDDDDDDDDDDDDDDDD	3456	3456	345	345	接收方重新发送 上次的确认报文: seqnum = 0, acknum = 2, checksum = 12848
接收方没有正确收到发送方的报文, 数据校验错误: seqnum = 3, acknum = -1, checksum = 24413, DCCCCCCCCCCCCCCCCCCCC	3456	3456	345	345	接收方重新发送 上次的确认报文: seqnum = 0, acknum = 2, checksum = 12848
*****模拟网络环境*****: 超时 启动定时器, 当前时间 = 328.665, 定时器报文序号 = 4, 发送方的数据包将在336.625到达对方, 数据包为-->seqnum = 4, 发送方的数据包将在340.765到达对方, 数据包为-->seqnum = 5,	6781	6781	45	45	发送方的数据包将在336.625到达对方, 数据包为-->seqnum = 4, 发送方的数据包将在340.765到达对方, 数据包为-->seqnum = 5,

使用检测脚本检测 SR 协议的传输正确性得到的结果如图 2.13 所示

```

FC: 找不到差异
Test "Select_Repeat.exe" 4:
正在比较文件 input.txt 和 OUTPUT.TXT
FC: 找不到差异
Test "Select_Repeat.exe" 5:
正在比较文件 input.txt 和 OUTPUT.TXT
FC: 找不到差异
Test "Select_Repeat.exe" 6:
正在比较文件 input.txt 和 OUTPUT.TXT
FC: 找不到差异
Test "Select_Repeat.exe" 7:
正在比较文件 input.txt 和 OUTPUT.TXT
FC: 找不到差异
Test "Select_Repeat.exe" 8:
正在比较文件 input.txt 和 OUTPUT.TXT
FC: 找不到差异
Test "Select_Repeat.exe" 9:
正在比较文件 input.txt 和 OUTPUT.TXT
FC: 找不到差异
Test "Select_Repeat.exe" 10:
正在比较文件 input.txt 和 OUTPUT.TXT
FC: 找不到差异
请按任意键继续. . .

```

图 2.13

SR 结果分析

首先验证 SR 协议传输的正确性，由图脚本测试结果，该 SR 模拟实现在测试用例上实现了可靠数据传输。

其次再验证 SR 接收方和 SR 发送方对各个事件的响应是否正确，SR 对事件的响应情况如图 2.14 所示根据表 2.9，SR 接收方和发送方对各种事件均做出正确响应。

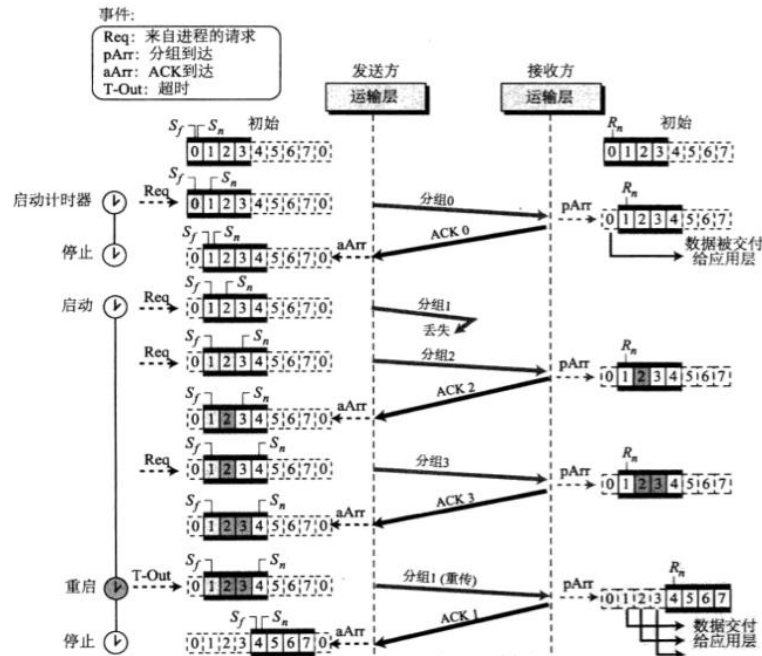


图 2.14

其中需要关注的是滑动窗口的滑动形式，SR 规定的滑动窗口滑动方式和测试用例中滑动窗口的滑动方式相同。

另一个需要关注的部分是对当发生超时时间时发送方的重新发送方式。根据 SR 的重发规则，每次出现超时现象重发的分组都是当前滑动窗口内的所有已发送但未确认报文段。根据测试用例中的结果，重传方式符合预期。

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

数据结构设计

本实验中所设计的 TCP 设计为运输层可靠传输内容，为满足实验要求，需要设计运输层报文段结构；同时，为了模拟实际网络环境下的数据收发，还需设计应用层的消息结构。运输层报文段数据结构具体内容说明如表 2.10 所示。

表 2.10

参数名称	数据类型	说明
seqnum	int	首部序号
acknum	int	首部确认号
checksum	int	报文校验和
payload	char[]	PayLoad

具体到实现 TCP 重传方式时，需要设计 TCP 发送方数据结构和 TCP 接收方数据结构，上述两个 GBN 模型实体的具体参数信息的配置如表 2.11 和表 2.12 所示。

表 2.11

参数名称	数据类型	说明
base	int	最早未被确认或最初未发送的报文段序号
nextseqnum	int	最早未发送报文段序号
timerseq	int	timer 的序号
N	const int	窗口的大小
pktQueue	vector<Packet>	窗口中的报文段队列
timeoutCount	int *	记录当前冗余

表 2.12

参数名称	数据类型	说明
expectSequenceNumberRcvd	int	期待收到的下一个报文序号
lastAckPkt	Packet	上次发送的确认报文

其中发送方的发送窗口结构示意图如图 2.15 所示



图 2.15

其中接收方的接收窗口结构示意图如图 2.16 所示

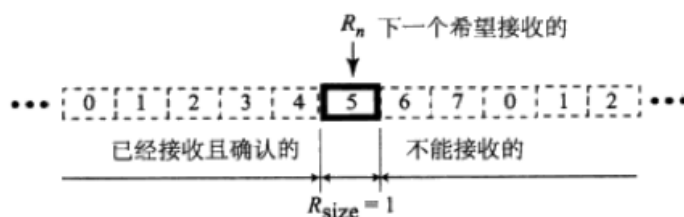


图 2.16

TCP 发送方逻辑设计

在本实验中，需要实现的部分是发送方状态机对于应用层请求调用运输层提供的可靠数据传输协议传输数据的响应、接收到低层网络层传输到来的报文段的处理，以及发送且未确认的报文段的超时事件响应。

对于应用层发出的 RDT 调用请求，TCP 协议下发送方的响应行为可用流程图 2.17 描述。

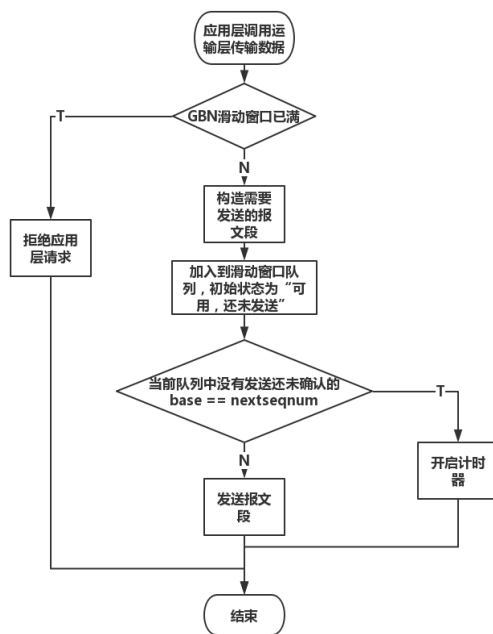


图 2.17

对于网络层发出的数据报接收请求，TCP 协议下的发送方的响应行为可用流程图 2.18 描述。

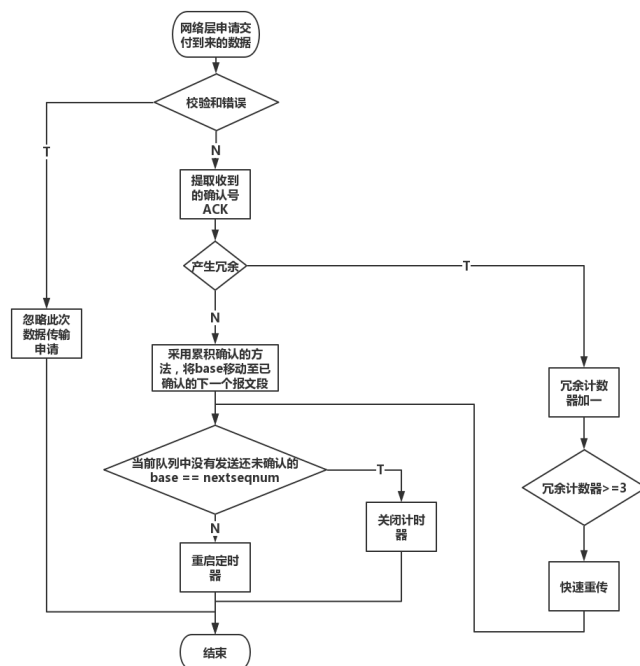


图 2.18

对于发送且未确认的报文段的超时事件。TCP 模式下的发送方的响应行为可用流程图 2.19 描述。

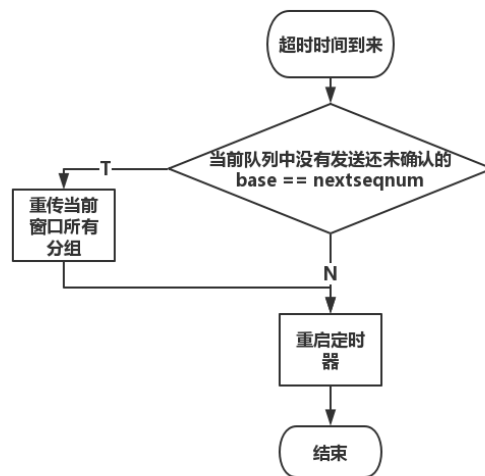


图 2.19

GBN 接收方逻辑设计

在本实验中，需要实现的部分是接收方状态机对于接收到低层网络层传输到来的报文段的处理。对于网络层发出的数据报接收请求，TCP 模式下的接收方的响应行为可用流程图 2.20 描述。这里的序号不再采用 GBN 中所使用的报文段编号，而是每个报文段的首字节编号。

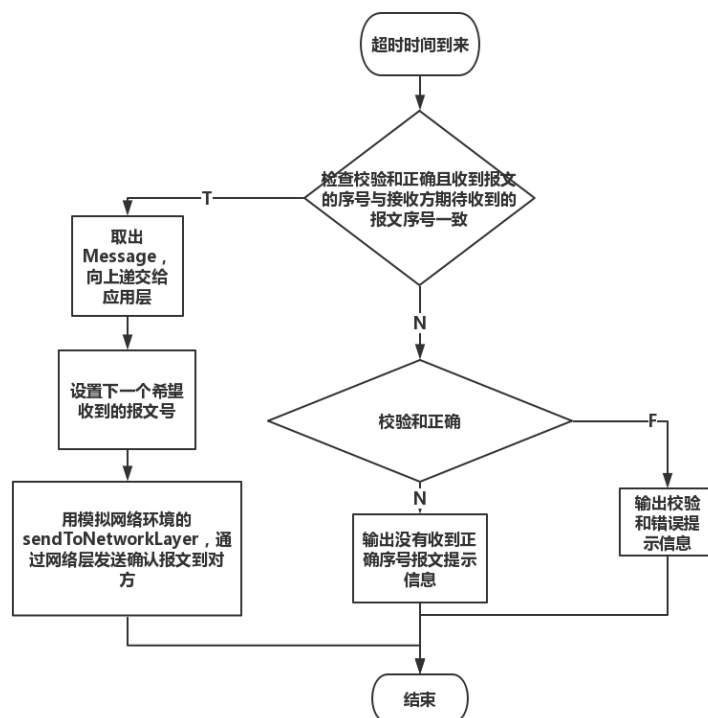


图 2.20

TCP 协议验证结果

在模拟网络环境下运行 GBN 模型下的 RDT 协议，得到验证结果如表 2.13 所示。

表 2.13

事件	接收方 原滑动 窗口	事件发 生后接 收方滑 动窗口	发送方 原滑动 窗口	事件 发生 后发 送方 滑动 窗口	额外操作
发送方发送报文: seqnum = 0, acknum = -1, checksum = 29556, AAAAAAAAAAAAAAAAAAAAA	empty	empty	empty	0	无
接收方发送确认报文: seqnum = 0, acknum = 2, checksum = 12848,	2	empty	2345	2345	无
接收方没有正确收到发送方的报文,报文序号不对: seqnum = 4, acknum = -1, checksum = 21842, DDDDDDDDDDDDDDDDDDDD	empty	empty	345	345	接收方重新发送 上次的确认报文: seqnum = 0, acknum = 2, checksum = 12848
接收方没有正确收到发送方的报文, 数据校验错误: seqnum = 3, acknum = -1, checksum = 24413, DCCCCCCCCCCCCCCCCCCC	3	3	345	345	接收方重新发送 上次的确认报文: seqnum = 0, acknum = 2, checksum = 12848
*****模拟网络环境*****: 超时 启动定时器, 当前时间 = 328.665, 定时器报文序号 = 4, 发送方的数据包将在336.625到达对方, 数据包为-->seqnum = 4, 发送方的数据包将在340.765到达对方, 数据包为-->seqnum = 5,	6	6	45	45	发送方的数据包将在336.625到达对方, 数据包为-->seqnum = 4, 发送方的数据包将在340.765到达对方, 数据包为-->seqnum = 5,
接收方的确认包将在88.1175到达对方 确认包为-->seqnum = 0, acknum = 2, checksum = 12848, payload = timeoutCount = 3, 出现3个冗余的ACK	empty	empty	4567	4567	网络层数据包损坏, 损坏的包为-->seqnum = 3, acknum = -1,checksum = 24413, payload = DCCCCCCCCCCCCCCCCCCC *****模拟网络环境*****: 发送方的数据包将在95.5375到达对方, 数据包为-->seqnum = 3, acknum = -1, checksum = 24413, payload = DCCCCCCCCCCCCCCCCCCC

使用检测脚本检测 GBN 协议的传输正确性得到的结果如图 2.21 所示

```
C:\WINDOWS\system32\cmd.exe
C: 找不到差异

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

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

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

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

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

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

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

请按任意键继续. . .
```

图 2.21

TCP 结果分析

首先验证 TCP 协议传输的正确性，由图脚本测试结果，该 TCP 模拟实现在测试用例上实现了可靠数据传输。

其次再验证 TCP 接收方和 GBN 发送方对各个事件的响应是否正确，根据表 2.13，GBN 接收方和发送方对各种事件均做出正确响应。

其中需要关注的是滑动窗口的滑动形式，TCP 规定的滑动窗口滑动方式和 GBN 相同，和测试用例中滑动窗口的滑动方式相同。

另一个需要关注的部分是对当发生超时时间时发送方的重新发送方式。在本次设计中采用 GBN 的重发规则，每次出现超时现象重发的分组都是当前滑动窗口内的所有报文段。根据测试用例中的结果，重传方式符合预期。

TCP 的另一大特点是快速重传机制，每当收到 3 个冗余的 ACK 时，重发该分组。在 TCP 协议验证阶段，可以看到这种重传机制成功实现。

实验三 基于 CPT 的组网实验

3.1 环境

Intel Core i7-6700HQ CPU 2.6GHz

RAM 16GB

Windows10 1803（实验机器的硬件配置、系统软件组件、第三方软件）

Cisco Packet Tracer Version: 7.2.0.0226

3.2 实验要求

基础部分：

分为两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1.1和1.2所示。

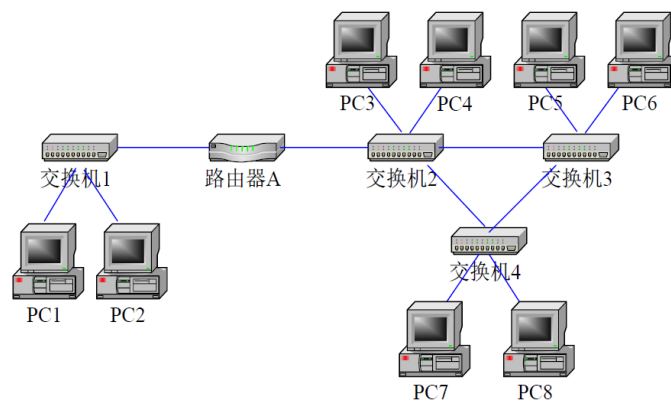


图3.1

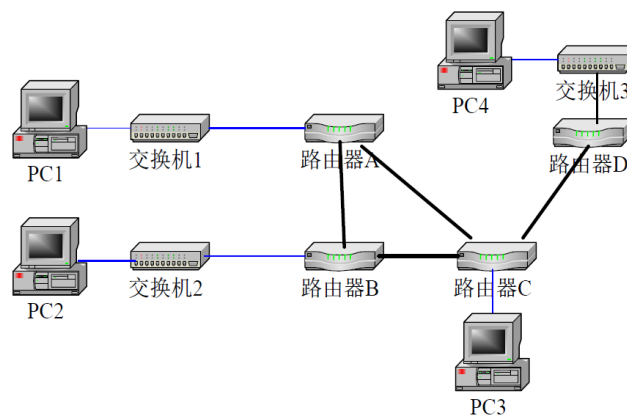


图3.2

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：

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

实验设计

IP 地址规划：对两个子网进行 IP 划分，分别设置地址 192.168.0.0/24 和 192.168.1.0/24，即对路由器的两个接口设置 IP 地址，分别为 192.168.0.254 和 192.168.1.254，并在划分的子网地址的范围内为子网内的各个主机分配地址，并将连接的路由器接口的地址作为其网关地址，主机 1 和 2 同在一个子网中，设置其 IP 地址 192.168.0.1 和 192.168.0.2，网关地址都为 192.168.0.254，主机 3 到 8 同在一个子网中，设置 IP 地址 192.168.1.3-192.168.1.8，网关地址为 192.168.1.254 同一子网的主机通过交换机连入该子网对应的路由器的接口，尝试向相同或不同子网内的主机通信，检测是否联通。

Vlan 分配：按照实验要求将各个主机分配到对应的网段中，即将主机的 IP 地址划分到相应的网段中并设置对应的网关地址，与上一步实验类似，再对交换机进行配置，对于不同的

Vlan 配置不同的编号，为 Vlan2 设置 20，Vlan3 设置 30，Vlan4 设置 40，并按要求将连接主机的端口设置到相应的 Vlan 中，例如与主机 1 连接的端口设置 Vlan 号 20，交换机之间和与路由器连接的端口设置为全 Vlan 段，测试相同和不同 Vlan 之间的通信，之后对路由器进行配置，为每个端口划分子端口，根据其对应的子网设置子端口 IP 地址并将其划分到相应的 Vlan 中，对于只连接子网 1 的端口只需要设置一个子端口 192.168.0.254 并将其划入 Vlan2 中，对于连接子网 2,3 的端口需要设置两个子端口，IP 分别为 192.168.1.254 和 192.168.2.254 分别划入 Vlan3，Vlan4 中，测试相同和不同 Vlan 之间的通信。

实验步骤

IP 地址规划：

根据要求选取相应的器件并使用适合的线连接，如图 3.3 所示

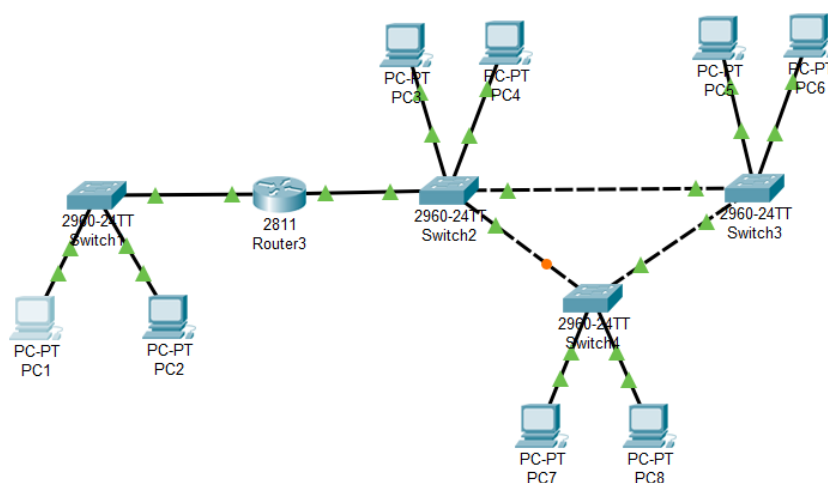
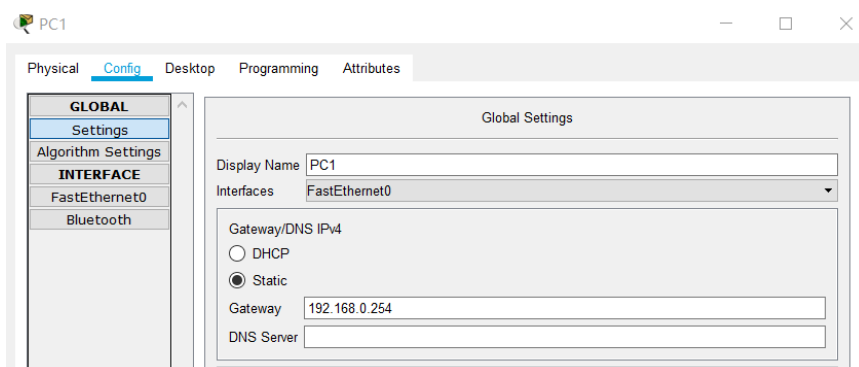


图 3.3 实验结构图

配置主机的网关和 IP 地址，如图 3.4 所示



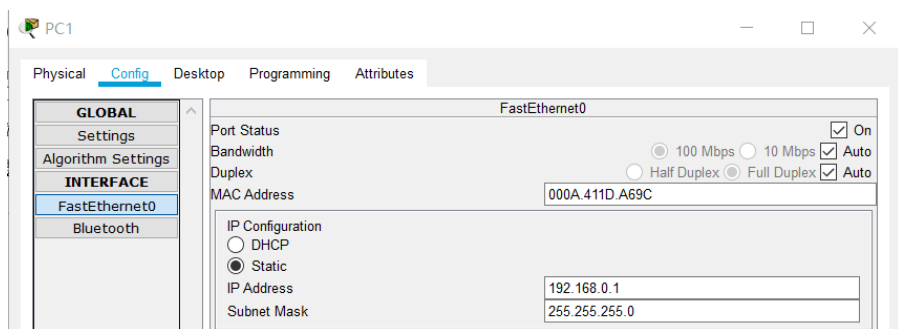


图 3.4 主机的网关设置和某一端口的 IP 设置

配置路由器的 IP，如图 3.5 所示

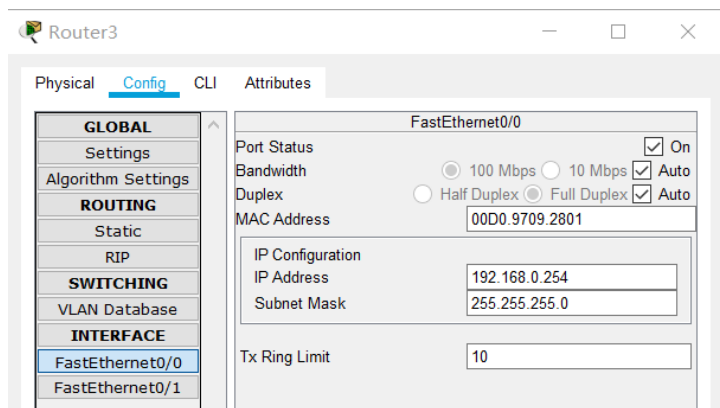


图 3.5 路由器某一端口的 IP 设置

通过主机 1 分别向主机 2 和主机 3 进行通信，通过主机上的 Command Prompt，打开后输入 ping+想要通信的主机的 IP 地址，如图 3.6 所示。

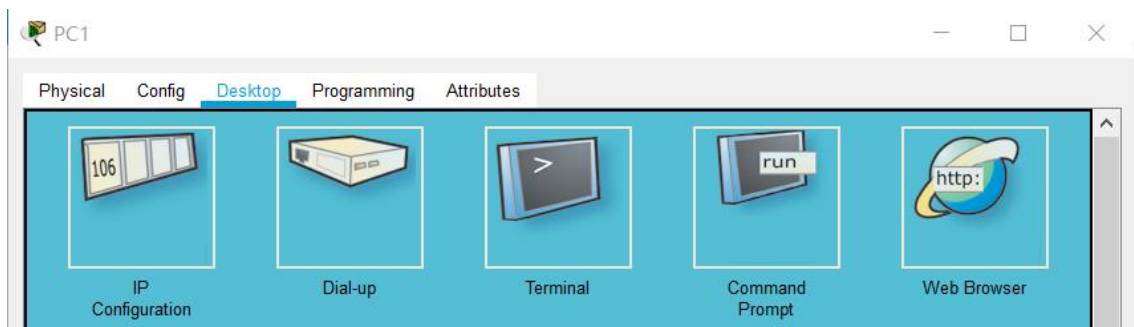


图 3.6 进行通信测试的方法

Vlan 分配:

器件及线路同 IP 地址规划实验相同，并用与之相同的方法配置每台主机的 IP 地址和网关号。

之后对交换机进行配置，添加需要的 Vlan 和其对应编号，如图 3.7 所示。

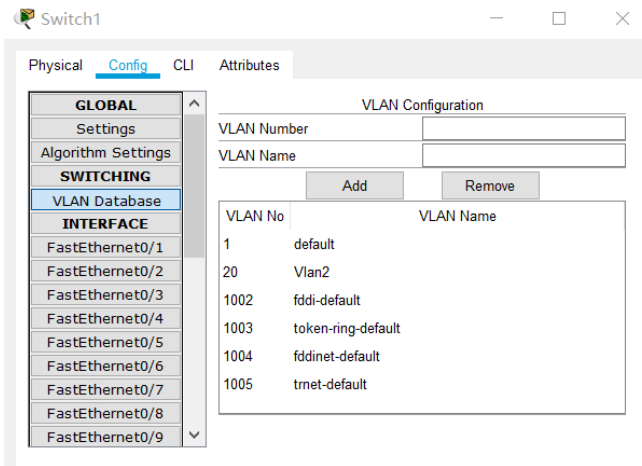


图 3.7 不同 Vlan 的添加

为与某一 Vlan 中主机相连的端口设置相应的 Vlan，如图 3.8 所示

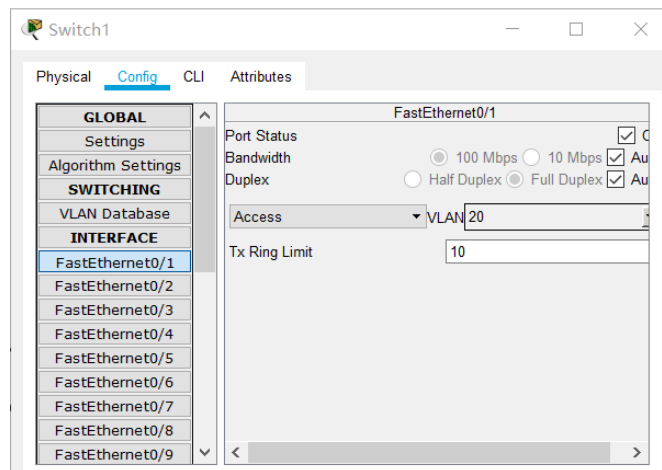


图 3.8 主机相连端口的 Vlan 设置

为与其他端口设置为 Trunk，如图 3.9 所示

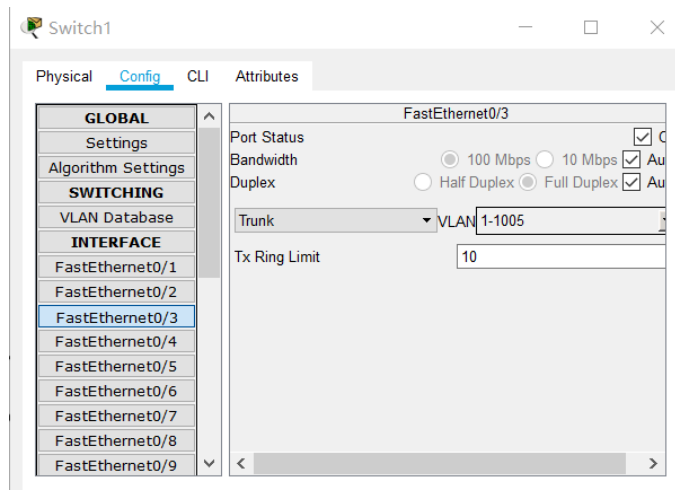


图 3.9 其他端口相连的 Vlan 设置

对相同或不同 Vlan 中的主机进行通信测试。

配置路由器，划分子端口并分配给对应的 Vlan，具体配置代码为：

```
Router>
Router>en
Router#conf t
Router(config)#int fa0/1.1 创建并进入子接口 f0/1.1
Router(config-subif)#encapsulation dot1q 20
将以太网子接口 fa0/1.1 划分到 vlan2，并且封装格式为 802.1q
Router(config-subif)#ip address 192.168.1.254 255.255.255.0
配置子接口 fa0/1.1 ip 为 192.168.1.254 255.255.255.0
Router(config-subif)#int fa0/1
Router(config-if)#no shut 激活端口
Router(config-if)#
```

另一个端口的配置与上述代码类似。

再次对相同或不同 Vlan 中的主机进行通信测试。

结果分析

IP 地址规划：

```
Packet Tracer PC Command Line 1.0
C:\>ping 192.168.0.2

Pinging 192.168.0.2 with 32 bytes of data:

Reply from 192.168.0.2: bytes=32 time=10ms TTL=128
Reply from 192.168.0.2: bytes=32 time<1ms TTL=128
Reply from 192.168.0.2: bytes=32 time<1ms TTL=128
Reply from 192.168.0.2: bytes=32 time<1ms TTL=128

Ping statistics for 192.168.0.2:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 0ms, Maximum = 10ms, Average = 2ms
```

```
C:\>ping 192.168.1.3

Pinging 192.168.1.3 with 32 bytes of data:

Reply from 192.168.1.3: bytes=32 time<1ms TTL=127
Reply from 192.168.1.3: bytes=32 time<1ms TTL=127
Reply from 192.168.1.3: bytes=32 time=1ms TTL=127
Reply from 192.168.1.3: bytes=32 time<1ms TTL=127

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

图 3.10 PC1 到 PC2, PC3 的通信测试

通过测试结果图 3.10，可以看出主机之间已经联通。

Vlan 分配：

未配置路由器：

```
C:\>ping 192.168.0.2

Pinging 192.168.0.2 with 32 bytes of data:

Reply from 192.168.0.2: bytes=32 time=1ms TTL=128
Reply from 192.168.0.2: bytes=32 time<1ms TTL=128
Reply from 192.168.0.2: bytes=32 time<1ms TTL=128
Reply from 192.168.0.2: bytes=32 time<1ms TTL=128

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


C:\>ping 192.168.1.3

Pinging 192.168.1.3 with 32 bytes of data:

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

Ping statistics for 192.168.1.3:
    Packets: Sent = 4, Received = 0, Lost = 4 (100% loss),
```

图 3.11 不同 Vlan 直接的通信

由图 3.11 可知同一 Vlan 中的主机可以相连，不同的不能相连。

配置路由器后：

```
C:\>ping 192.168.0.2

Pinging 192.168.0.2 with 32 bytes of data:

Reply from 192.168.0.2: bytes=32 time=1ms TTL=128
Reply from 192.168.0.2: bytes=32 time<1ms TTL=128
Reply from 192.168.0.2: bytes=32 time<1ms TTL=128
Reply from 192.168.0.2: bytes=32 time<1ms TTL=128

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


C:\>ping 192.168.1.3

Pinging 192.168.1.3 with 32 bytes of data:

Request timed out.
Reply from 192.168.1.3: bytes=32 time<1ms TTL=127
Reply from 192.168.1.3: bytes=32 time=3ms TTL=127
Reply from 192.168.1.3: bytes=32 time<1ms TTL=127

Ping statistics for 192.168.1.3:
    Packets: Sent = 4, Received = 3, Lost = 1 (25% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 0ms, Maximum = 3ms, Average = 1ms
```

图 3.12 配置路由器后不同 Vlan 直接的通信

在配置路由器后，不同 Vlan 之间的也可以进行通信。

3.3.2 路由配置实验的步骤及结果分析

实验设计

RIP 协议：按照要求为各个主机配置相应的 IP 地址使其在相应的网段中，并设置路由器各个接口的 IP 地址以及主机的网关地址，路由器之间的连接需要通过 Serial 接口，并用相应的线进行连接，由于各个主机之间的连接由它们对应的路由器实现，所以为实现各个主机之间

的通信，需要在路由器上配置 RIP 协议，RIP 协议的实现只需在路由器中配置各个接口的对应的网段即可实现，例如在路由器 A 中设置：连接 PC1 端口的网段 192.168.1.0，连接路由器 B 和 C 的端口的网段 192.68.1.0 和 192.68.2.0。测试不同主机间的通信。

OSPF 协议：在与上述协议相同的电路中，为实现主机间的通信采用 OSPF 协议，设置路由器，将其所连的主机对应的端口号作为路由 id，将其所有端口对应的网段配置到相应网络区域中，例如路由器 A 中设置：连接 PC1 端口的 IP 地址 192.168.1.254 作为其 id 号，并将所有端口的网段，即到 PC1，路由器 B，C 的端口的网段 192.168.1.0，192.68.1.0，192.68.2.0 配置到相应的网络区域。测试不同主机间的通信。

访问控制：在 RIP 协议的基础上按照要求进行访问控制的配置，控制 PC1 无法被任何其他主机访问且无法访问其他主机，可以通过在在路由器 A 上设置拒绝 PC1 所在网段访问路由器 A 实现，而从 PC1 与 PC2 之间的通信也可以通过在路由器 A 上设置拒绝从 PC1 到 PC2 的访问实现，在设置时填入对于相关端口的设置，需填入对应网段及反子网掩码，设置完成后需添加到相应的接口上。测试不同主机间的通信。

实验步骤

RIP 协议：

选择相应器件并按题目要求进行连接，如图 3.13 所示。

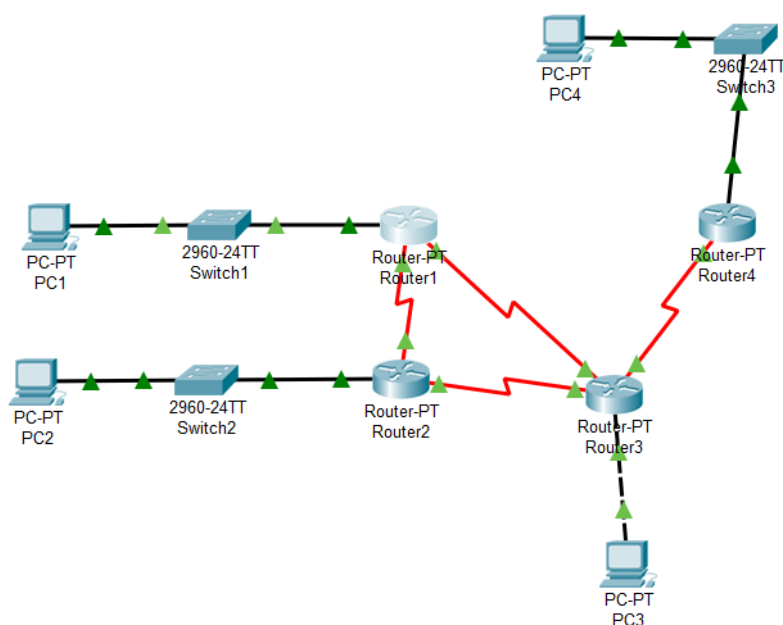


图 3.13 实验结构图

设置好个主机和路由器端口的 IP 地址后，在路由器中进行 RIP 协议的配置，如图 3.14 所示。

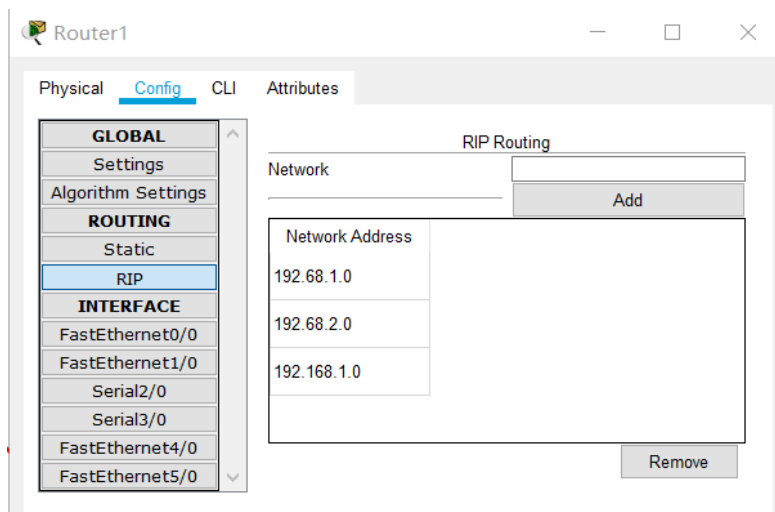


图 3.14 路由器 RIP 配置

对每个路由器按照自己端口的连接进行如上图所示的配置。
配置完成后，从主机 1 对各个主机进行通信测试。

OSPF 协议：

在进行如 RIP 协议实验中一样的连接和 IP 配置后，对每个路由器进行 OSPF 协议的配置，路由器 A 的配置代码如下所示：

```
Router(config)#router ospf 1
Router(config-router)#router-id 192.168.1.254
Router(config-router)#net 192.168.1.0 255.255.255.0 area 0
Router(config-router)#net 192.68.1.0 255.255.255.0 area 0
Router(config-router)#net 192.68.2.0 255.255.255.0 area 0
```

根据每个路由器端口的对应的 IP 地址或网段的不同进行不同的配置。
配置完成后，从主机 1 对各个主机进行通信测试。

访问控制：

在 RIP 协议实现的基础上对其中的路由器 A 进行访问控制。

PC1 无法访问其他主机，其他主机也无法访问 PC1：

对路由器 A 进行如下配置，设置访问列表：

```
Router(config)#access-list 1 deny 192.168.1.0 0.0.0.255
Router(config)#access-list 1 permit any
```

然后让相应的端口应用该配置：

```
Router(config)#int fa0/1
```

```
Router(config)#ip access-group 1 in
```

```
Router(config)#exit
```

由此便实现了对 PC1 的访问控制。

PC1 无法访问 PC2，PC2 也无法访问 PC1：

对路由器 A 进行如下配置，设置访问列表：

```
Router(config)#access-list 101 deny icmp 192.168.1.0 0.0.0.255 192.168.2.0 0.0.0.255
```

```
Router(config)#access-list 101 permit ip any any
```

然后让相应的端口应用该配置：

```
Router(config)#int fa0/1
```

```
Router(config)#ip access-group 1 in
```

```
Router(config)#exit
```

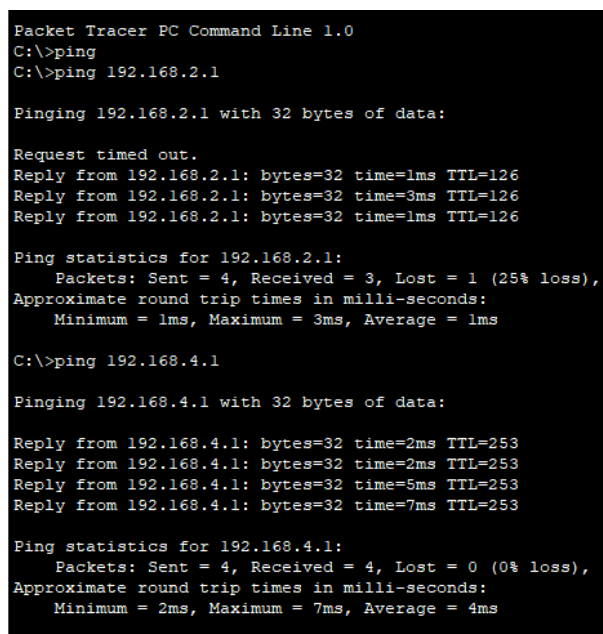
由此便实现了对 PC1 的访问控制。

配置完成后，从 PC1 对各个主机进行通信测试。

结果分析

RIP 协议：

分别尝试与 PC2 和 PC4 通信，结果如图 3.15 所示。



```
Packet Tracer PC Command Line 1.0
C:\>ping
C:\>ping 192.168.2.1

Pinging 192.168.2.1 with 32 bytes of data:

Request timed out.
Reply from 192.168.2.1: bytes=32 time=1ms TTL=126
Reply from 192.168.2.1: bytes=32 time=3ms TTL=126
Reply from 192.168.2.1: bytes=32 time=1ms TTL=126

Ping statistics for 192.168.2.1:
    Packets: Sent = 4, Received = 3, Lost = 1 (25% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 1ms, Maximum = 3ms, Average = 1ms

C:\>ping 192.168.4.1

Pinging 192.168.4.1 with 32 bytes of data:

Reply from 192.168.4.1: bytes=32 time=2ms TTL=253
Reply from 192.168.4.1: bytes=32 time=2ms TTL=253
Reply from 192.168.4.1: bytes=32 time=5ms TTL=253
Reply from 192.168.4.1: bytes=32 time=7ms TTL=253

Ping statistics for 192.168.4.1:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 2ms, Maximum = 7ms, Average = 4ms
```

图 3.15 PC1 到 PC2，PC4 的通信测试

由图 3.15 可知，在配置完 RIP 协议后，主机之间可以进行通信。

OSPF 协议：

分别尝试与 PC3 和 PC4 通信，结果如图 3.16 所示。


```

Packet Tracer PC Command Line 1.0
C:\>ping 192.168.3.1

Pinging 192.168.3.1 with 32 bytes of data:

Request timed out.
Reply from 192.168.3.1: bytes=32 time=1ms TTL=126
Reply from 192.168.3.1: bytes=32 time=2ms TTL=126
Reply from 192.168.3.1: bytes=32 time=2ms TTL=126

Ping statistics for 192.168.3.1:
    Packets: Sent = 4, Received = 3, Lost = 1 (25% loss),
Approximate round trip times in milli-seconds:
    Minimum = 1ms, Maximum = 2ms, Average = 1ms

C:\>ping 192.168.4.1

Pinging 192.168.4.1 with 32 bytes of data:

Reply from 192.168.4.1: bytes=32 time=10ms TTL=253
Reply from 192.168.4.1: bytes=32 time=2ms TTL=253
Reply from 192.168.4.1: bytes=32 time=2ms TTL=253
Reply from 192.168.4.1: bytes=32 time=2ms TTL=253

Ping statistics for 192.168.4.1:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 2ms, Maximum = 10ms, Average = 4ms

```

图 3.16 PC1 到 PC3, PC4 的通信测试

由图 3.16 可知，在配置完 OSPF 协议后，主机之间可以进行通信。

访问控制：

配置路由器使 PC1 无法访问其他 PC，且其他 PC 无法访问 PC1：

测试 PC1 能否与其他 PC 通信，结果如图 3.17 所示。

```

Packet Tracer PC Command Line 1.0
C:\>ping 192.168.2.1

Pinging 192.168.2.1 with 32 bytes of data:

Reply from 192.168.1.254: Destination host unreachable.
Reply from 192.168.1.254: Destination host unreachable.
Reply from 192.168.1.254: Destination host unreachable.
Reply from 192.168.1.254: Destination host unreachable.

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

C:\>ping 192.168.3.1

Pinging 192.168.3.1 with 32 bytes of data:

Reply from 192.168.1.254: Destination host unreachable.
Reply from 192.168.1.254: Destination host unreachable.
Reply from 192.168.1.254: Destination host unreachable.
Reply from 192.168.1.254: Destination host unreachable.

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

C:\>ping 192.168.4.1

Pinging 192.168.4.1 with 32 bytes of data:

Reply from 192.168.1.254: Destination host unreachable.
Reply from 192.168.1.254: Destination host unreachable.
Reply from 192.168.1.254: Destination host unreachable.
Reply from 192.168.1.254: Destination host unreachable.

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

```

图 3.17 PC1 到其他主机的通信测试

根据图 3.17，PC1 无法与其他 PC 通信。

测试其他 PC 能否与 PC1 通信，结果如图 3.18 所示。

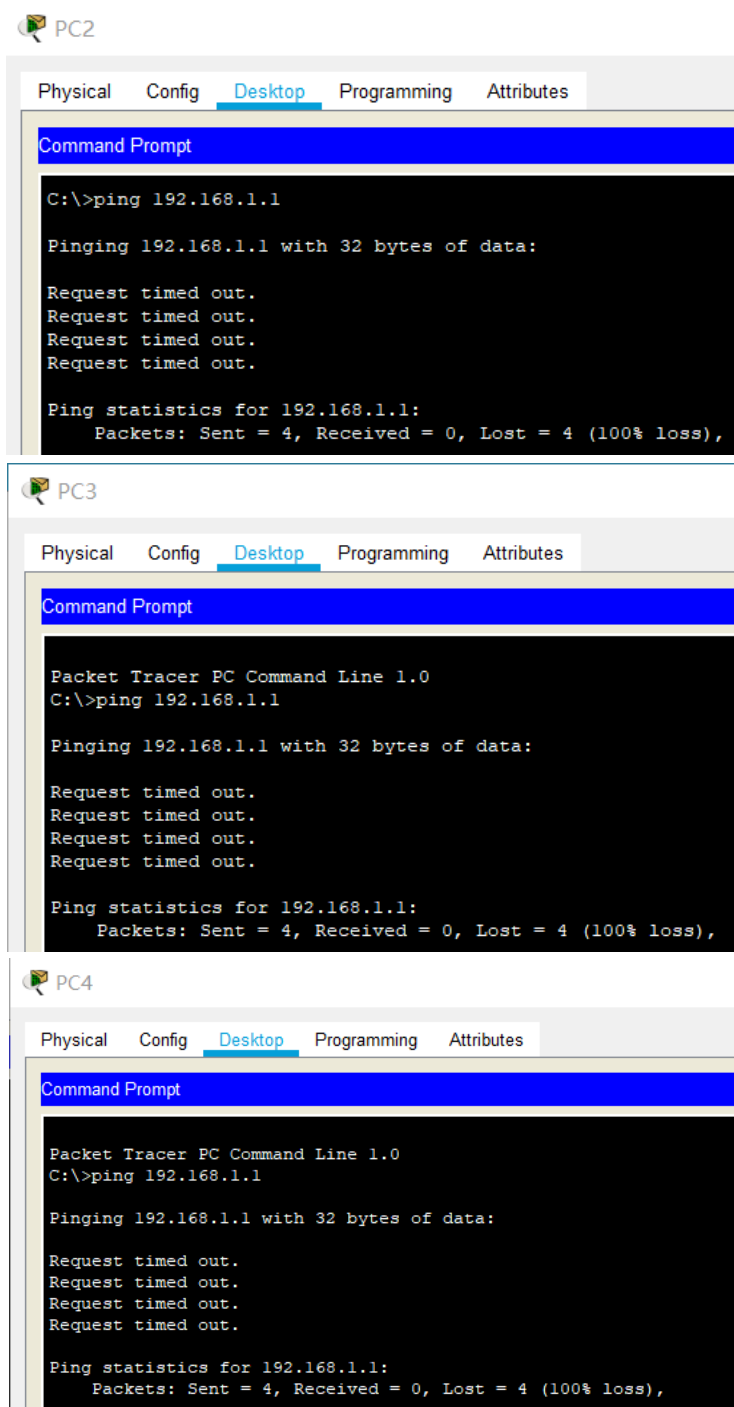


图 3.18 其他主机到 PC1 的通信测试

根据图 3.18 可知，其他 PC 也无法与 PC 通信。

控制访问实现。

配置路由器使 PC1 无法访问其他 PC2，且 PC2 无法访问 PC1：

测试 PC1 与各个 PC 通信，结果如图 3.19 所示。

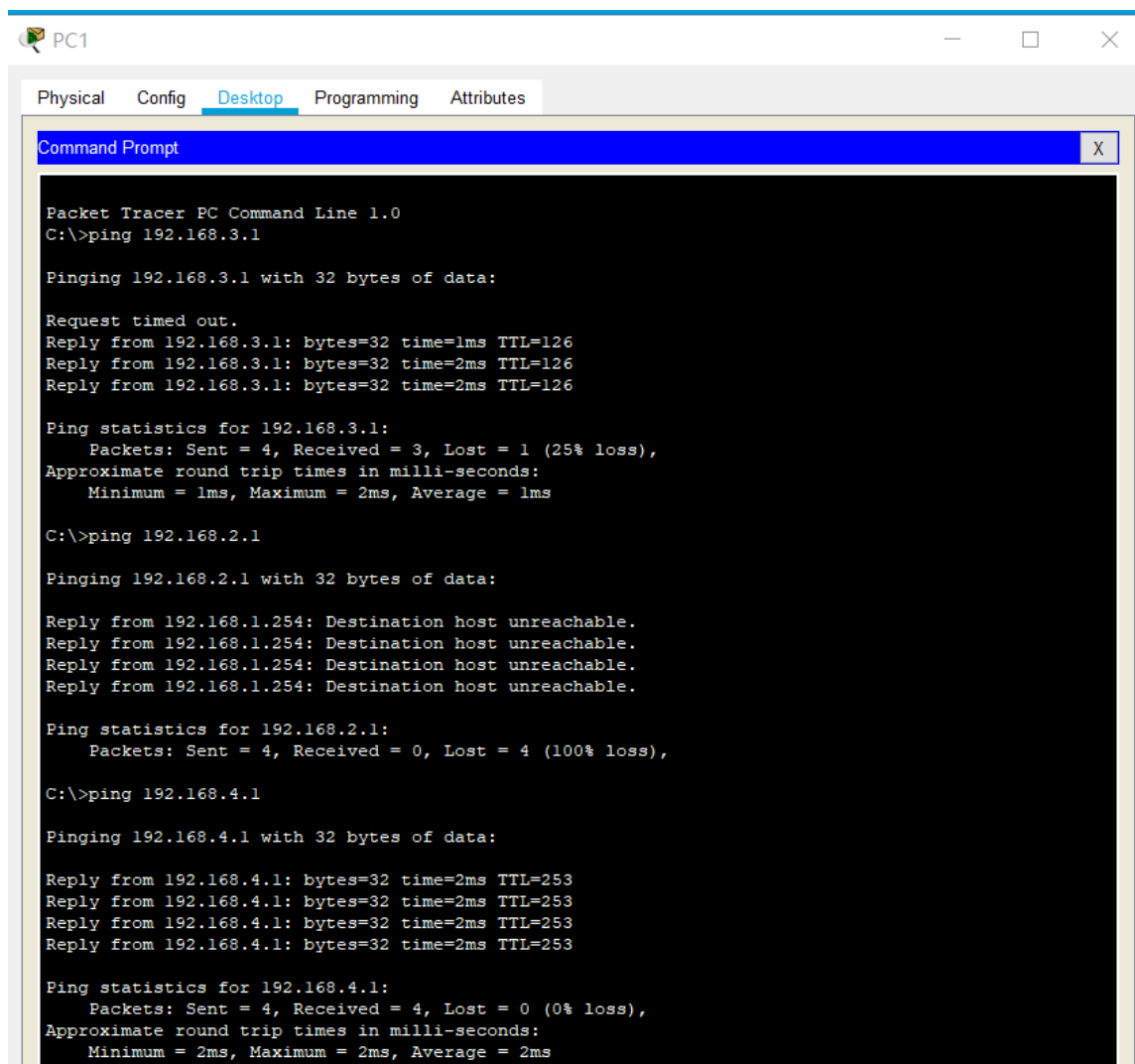
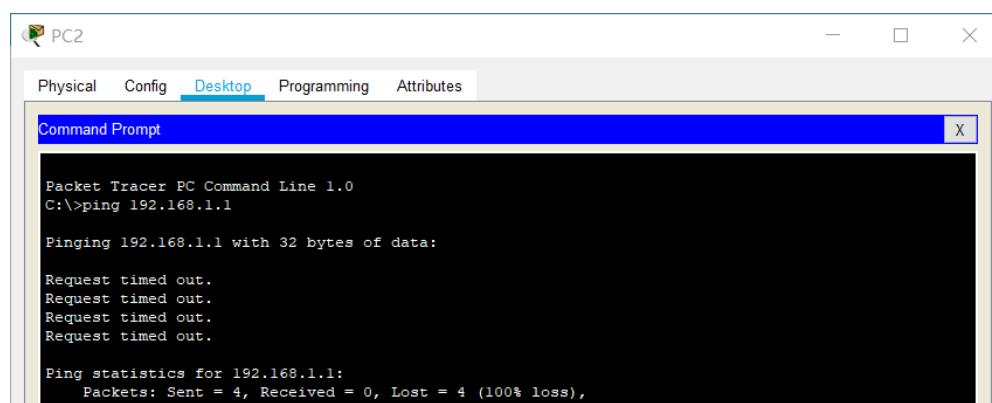


图 3.19 PC1 到其他主机的通信测试

根据图 3.19 可知，PC1 无法访问 PC2，而可以访问 PC3 和 PC4。

测试各个 PC 与 PC1 之间的通信，结果如图 3.20 所示。



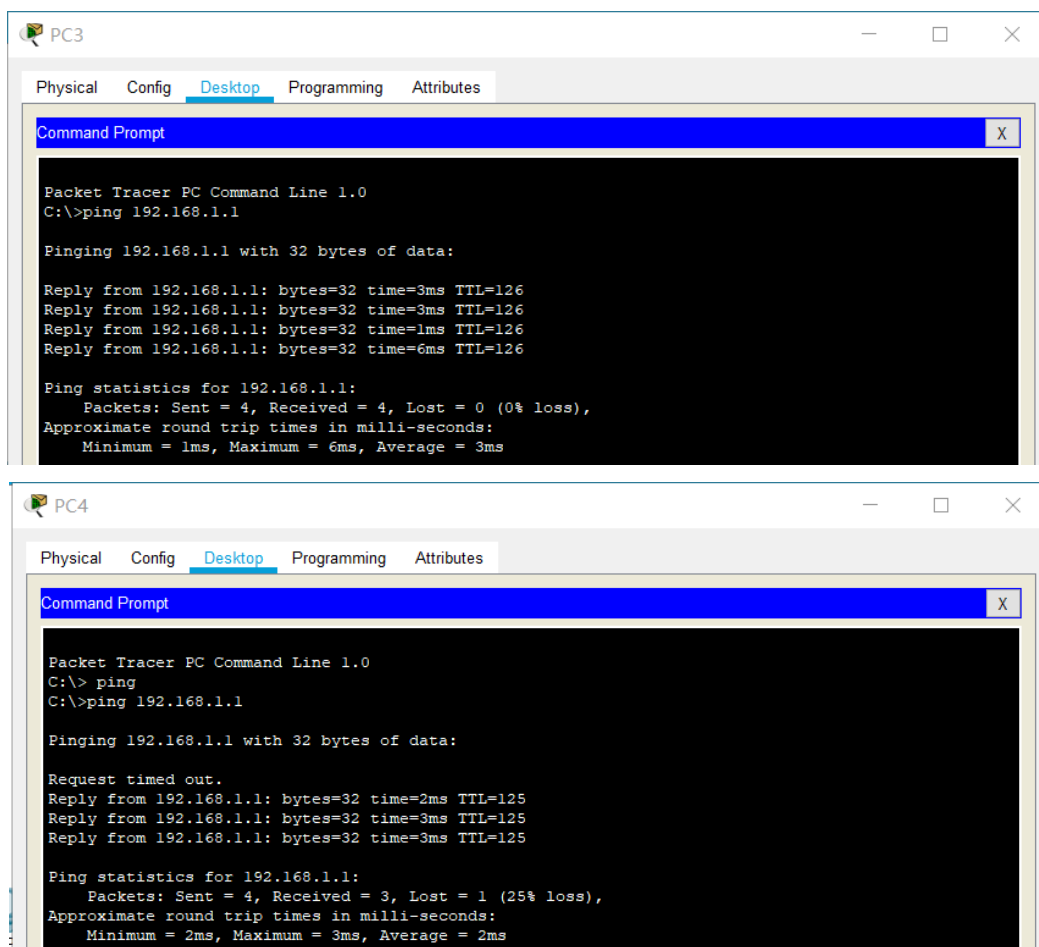


图 3.20 其他主机到 PC1 的通信测试

根据图 3.20 可知，PC2 无法访问 PC1，而 PC3 和 PC4 可以访问 PC1。
控制访问实现。

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

在该部分中，实现对于不同网段的划分和通信，可以仅通过一个路由器来完成，然后考虑对于不同网段的划分，由于共三个学生宿舍，每个学生宿舍有 200 台主机，所以为满足其要求，为每个宿舍划分 256 个 IP 地址，其网段分别为 211.69.4.0/24，211.69.5.0/24，211.69.6.0/24，然后考虑图书馆要求的 100 个主机，为其划分 128 个 IP 地址，其网段为 211.69.7.0/25，最后考虑各个学院，每个学院有 20 台主机，所以为每个学院划分 32 个 IP 地址，其网段分别为 211.69.7.128/27，211.69.7.160.27，211.69.7.192/27，211.69.7.224/27，至此网段划分完成，由于只使用一个路由器所以为每个宿舍，学院，图书馆设置单独的 Vlan 连接到路由器相应端口的子端口，并配置路由器 Vlan 信息以完成不同 Vlan 之间的通信，然后通过对路由器的访问控制设置实现宿舍和学院之间的无法互相访问，最后在图书馆设置无线网，并放入可连接无线网的

笔记本电脑，路由器为其向普通主机一样分配 IP 地址，其网关也为路由器相应端口的 IP 地址。配置全部完成后测试各个位置主机相互之间的通信。

3.4.2 实验步骤

每个学院，宿舍，图书馆有一个交换机连接所有 PC，所有学院，宿舍还要设置单独的交换机连接单独学院和宿舍的交换机，最后学院，宿舍，图书馆共三个交换机连入路由器，并在图书馆中设置无线网络和笔记本电脑。最后连接如图 3.21 所示（每个区域用一台 PC 用作测试）

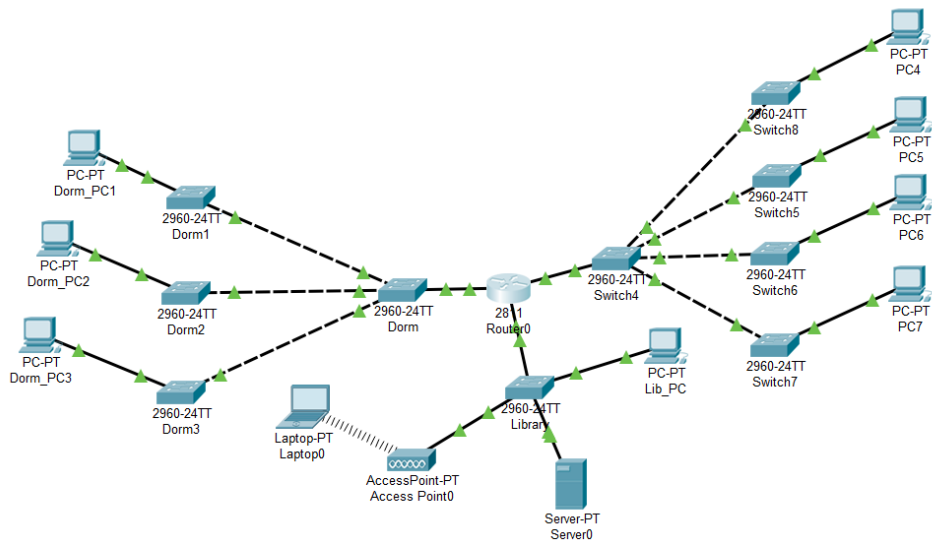


图 3.21 学校内部通信结构图

根据为 PC 和路由器配置设计好的网段设置其端口的 IP 地址，路由器配置如图 3.22 所示。

Port	Link	VLAN	IP Address	IPv6 Address	MAC Address
FastEthernet0/0	Up	--	<not set>	<not set>	0060.3E61.3801
FastEthernet0/0.1	Up	--	211.69.4.254/24	<not set>	0060.3E61.3801
FastEthernet0/0.2	Up	--	211.69.5.254/24	<not set>	0060.3E61.3801
FastEthernet0/0.3	Up	--	211.69.6.254/24	<not set>	0060.3E61.3801
FastEthernet0/1	Up	--	211.69.7.125/25	<not set>	0060.3E61.3802
FastEthernet1/0	Up	--	<not set>	<not set>	0010.1167.7B01
FastEthernet1/0.1	Up	--	211.69.7.150/27	<not set>	0010.1167.7B01
FastEthernet1/0.2	Up	--	211.69.7.182/27	<not set>	0010.1167.7B01
FastEthernet1/0.3	Up	--	211.69.7.214/27	<not set>	0010.1167.7B01
FastEthernet1/0.4	Up	--	211.69.7.246/27	<not set>	0010.1167.7B01
FastEthernet1/1	Up	--	<not set>	<not set>	0010.1167.7B02
Vlan1	Down	1	<not set>	<not set>	000A.F319.E9B1
Hostname: Router					
Physical Location: Intercity, Home City, Corporate Office, Main Wiring Closet					

图 3.22 子端口 IP 划分设置

在上述为子端口配置 IP 地址的过程中，同时对相应的端口设置 Vlan 划分，为不同的网段设置对应的 Vlan 序号，在各个交换机中也对根据不同网段对 Vlan 进行设置，如图 3.23 所示是为学院设置的 Vlan。（注意交换机与路由器连接的端口要设置为 Trunk，交换机之间或与主机的端口设置为相应网段的 Vlan 号）

Port	Link	VLAN	IP Address	MAC Address
FastEthernet0/1	Up	4	--	0002.173A.B701
FastEthernet0/2	Up	5	--	0002.173A.B702
FastEthernet0/3	Up	6	--	0002.173A.B703
FastEthernet0/4	Up	7	--	0002.173A.B704
FastEthernet0/5	Up	--	--	0002.173A.B705

图 3.23 学院间 Vlan 划分

最后在路由器中设置从学院到宿舍的访问控制，设置限制访问的代码如下：
设置访问控制表：

```
Router(config)#access-list 101 deny icmp 211.69.4.0 0.0.0.255 211.69.7.128 0.0.0.127
Router(config)#access-list 101 permit ip any any
Router(config)#access-list 102 deny icmp 211.69.5.0 0.0.0.255 211.69.7.128 0.0.0.127
Router(config)#access-list 102 permit ip any any
Router(config)#access-list 103 deny icmp 211.69.6.0 0.0.0.255 211.69.7.128 0.0.0.127
Router(config)#access-list 103 permit ip any any
```

然后让相应的端口应用该配置：

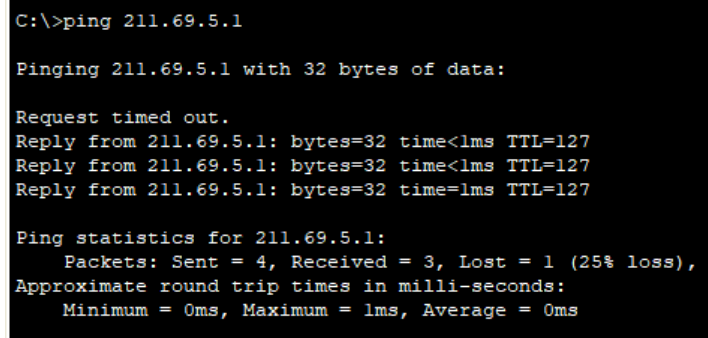
```
Router(config)#int fa0/0.1
Router(config)#ip access-group 101 in
Router(config)#exit
Router(config)#int fa0/0.2
Router(config)#ip access-group 102 in
Router(config)#exit
Router(config)#int fa0/0.3
Router(config)#ip access-group 103 in
Router(config)#exit
```

配置完成后测试各区域 PC 之间的通信。

3.4.3 结果分析

使用宿舍 PC：

测试宿舍 PC 之间相互通信，结果如图 3.24 所示：



```
C:\>ping 211.69.5.1

Pinging 211.69.5.1 with 32 bytes of data:

Request timed out.
Reply from 211.69.5.1: bytes=32 time<1ms TTL=127
Reply from 211.69.5.1: bytes=32 time<1ms TTL=127
Reply from 211.69.5.1: bytes=32 time=1ms TTL=127

Ping statistics for 211.69.5.1:
    Packets: Sent = 4, Received = 3, Lost = 1 (25% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 0ms, Maximum = 1ms, Average = 0ms
```

图 3.24 宿舍间通信测试

由图 3.24 可知，宿舍间通信已实现。

测试宿舍 PC 与图书馆 PC 相互通信，结果如图 3.25 所示：

```
C:\>ping 211.69.7.1

Pinging 211.69.7.1 with 32 bytes of data:

Request timed out.
Reply from 211.69.7.1: bytes=32 time<1ms TTL=127
Reply from 211.69.7.1: bytes=32 time<1ms TTL=127
Reply from 211.69.7.1: bytes=32 time=1ms TTL=127

Ping statistics for 211.69.7.1:
    Packets: Sent = 4, Received = 3, Lost = 1 (25% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 0ms, Maximum = 1ms, Average = 0ms
```

图 3.25 宿舍到图书馆通信测试

由图 3.25 可知，宿舍与图书馆间通信已实现。

测试宿舍 PC 与学院 PC 相互通信，结果如图 3.26 所示：

```
C:\>ping 211.69.7.130

Pinging 211.69.7.130 with 32 bytes of data:

Reply from 211.69.4.254: Destination host unreachable.
Reply from 211.69.4.254: Destination host unreachable.
Reply from 211.69.4.254: Destination host unreachable.
Reply from 211.69.4.254: Destination host unreachable.

Ping statistics for 211.69.7.130:
    Packets: Sent = 4, Received = 0, Lost = 4 (100% loss),
```

图 3.26 宿舍到学院通信测试

由图 3.26 可知，宿舍到学院的访问控制已实现。

使用学院 PC：

测试学院之间 PC 相互通信，结果如图 3.27 所示：

```
C:\>ping 211.69.7.172

Pinging 211.69.7.172 with 32 bytes of data:

Request timed out.
Reply from 211.69.7.172: bytes=32 time<1ms TTL=127
Reply from 211.69.7.172: bytes=32 time=1ms TTL=127
Reply from 211.69.7.172: bytes=32 time<1ms TTL=127

Ping statistics for 211.69.7.172:
    Packets: Sent = 4, Received = 3, Lost = 1 (25% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 0ms, Maximum = 1ms, Average = 0ms
```

图 3.27 学院间通信测试

由图 3.27 可知，学院之间的通信已实现。

测试学院 PC 与图书馆 PC 相互通信，结果如图 3.28 所示：

```
C:\>ping 211.69.7.1

Pinging 211.69.7.1 with 32 bytes of data:

Reply from 211.69.7.1: bytes=32 time=1ms TTL=127
Reply from 211.69.7.1: bytes=32 time<1ms TTL=127
Reply from 211.69.7.1: bytes=32 time<1ms TTL=127
Reply from 211.69.7.1: bytes=32 time=1ms TTL=127

Ping statistics for 211.69.7.1:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 0ms, Maximum = 1ms, Average = 0ms
```

图 3.28 学院到图书馆通信测试

由图 3.28 可知，学院与图书馆的通信已实现。

测试学院 PC 与宿舍 PC 相互通信，结果如图 3.29 所示：

```
C:\>ping 211.69.4.1

Pinging 211.69.4.1 with 32 bytes of data:

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

Ping statistics for 211.69.4.1:
    Packets: Sent = 4, Received = 0, Lost = 4 (100% loss),
```

图 3.29 学院到宿舍通信测试

由图 3.29 可知，学院到宿舍的访问控制已实现。

使用图书馆 PC：

测试图书馆 PC 到学院 PC 和宿舍 PC 的通信，结果如图 3.30 所示：

```
Packet Tracer PC Command Line 1.0
C:\>ping 211.69.4.1

Pinging 211.69.4.1 with 32 bytes of data:

Reply from 211.69.4.1: bytes=32 time<1ms TTL=127
Reply from 211.69.4.1: bytes=32 time=2ms TTL=127
Reply from 211.69.4.1: bytes=32 time<1ms TTL=127
Reply from 211.69.4.1: bytes=32 time<1ms TTL=127

Ping statistics for 211.69.4.1:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 0ms, Maximum = 2ms, Average = 0ms

C:\>ping 211.69.7.130

Pinging 211.69.7.130 with 32 bytes of data:

Reply from 211.69.7.130: bytes=32 time<1ms TTL=127
Reply from 211.69.7.130: bytes=32 time=1ms TTL=127
Reply from 211.69.7.130: bytes=32 time<1ms TTL=127
Reply from 211.69.7.130: bytes=32 time<1ms TTL=127

Ping statistics for 211.69.7.130:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 0ms, Maximum = 1ms, Average = 0ms
```

图 3.30 图书馆通信测试

由图 3.30 可知，图书馆到学院，宿舍的通信已实现。

使用图书馆无线网：

测试连接图书馆无线网的笔记本到学院 PC 和宿舍 PC 的通信，结果如图 3.31 所示：

```
C:\>ping 211.69.4.1

Pinging 211.69.4.1 with 32 bytes of data:

Reply from 211.69.4.1: bytes=32 time=16ms TTL=127
Reply from 211.69.4.1: bytes=32 time=18ms TTL=127
Reply from 211.69.4.1: bytes=32 time=14ms TTL=127
Reply from 211.69.4.1: bytes=32 time=15ms TTL=127

Ping statistics for 211.69.4.1:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 14ms, Maximum = 18ms, Average = 15ms

C:\>ping 211.69.7.130

Pinging 211.69.7.130 with 32 bytes of data:

Reply from 211.69.7.130: bytes=32 time=26ms TTL=127
Reply from 211.69.7.130: bytes=32 time=12ms TTL=127
Reply from 211.69.7.130: bytes=32 time=12ms TTL=127
Reply from 211.69.7.130: bytes=32 time=6ms TTL=127

Ping statistics for 211.69.7.130:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 6ms, Maximum = 26ms, Average = 14ms
```

图 3.31 图书馆无线网通信测试

由图 3.31 可知，图书馆的无线网络已实现。

至此要求中的全部要求已全部完成。

3.5 其它需要说明的问题

在连接不同的功能器件时，要注意其接口的类型，和使用的线的类型，有些接口和线不符合规范但也可以实现相应的功能，如：路由器之间使用 Fa 端口，且使用实线。

在进行通信测试的时候，第一个数据传输往往会超时，而之后的向同一网段的数据传输便不会超时，且同一网段的其它主机向这一网段发送信息都不会超时，而这一网段向接受过消息的网段通信时也不会超时。

在进行访问控制时，从被控网段与控制网段通信时提示不可达，而从控制网段向被控网段通信时会超时。

无线网络中的设备同有线设备一样设置在网段内的 IP 地址及路由器对应的网关。

实验中的主机和路由器的地址都为静态设置，也可以通过 DHCP 进行动态设置。

心得体会与建议

4.1 心得体会

通过一系列的网络实验，我对在网络课上学习的知识的理解更加深刻了，对于网络各层所实现的功能也有了更加清晰的理解，同时实验也让我亲身接触到了网络各个协议的设计和实现，从设计者的角度看网络，同时在实验过程中，我对各个软件及其功能也更加熟悉和了解，也掌握了一些以前从来没有接触到的软件和以前没有使用过的编程功能和技巧，总而言之，网络实验使我学会了很多新的知识，可以说是受益匪浅。

4.2 建议

希望老师提供或讲解更多与实验相关的知识，一些从未接触过得软件或功能在初次使用时会遇到各种各样的问题，影响完成实验的效率，一些知识仅仅是通过我们自学可能会花费大量的时间，如果老师能对实验过程中的一些大家不太熟悉的知识进行一定的讲解和引导，同学们完成实验的效率和质量都会有所提高。