**北京科技大学 计算机与通信工程学院**

**实 验 报 告**

**实验名称**： 计算机网络课程设计

**学生姓名**： 李晓坤、王若凡、王宠爱

**专 业**： 信息安全

**班 级**： 信安211

**学 号**： U202141863、U202141852、U202141853

**指导教师**： 崔晓龙

**实验成绩**：

**实验地点**： 机电楼316

**实验时间**： 2024 年 3 月

**北京科技大学实验报告**

学院：计算机与通信工程学院 专业：信息安全 班级：信安2101

姓名： 学号： 实验日期： 2024 年 3 月

李晓坤，王若凡

王宠爱

U202141863, U202141852

U202141853

**实验名称：**计算机网络课程设计

**实验目的：**

计算机网络课程设计包括计算机网络系统设计与网络编程两部分内容，其中网络系统设计可以使学生全面地掌握计算机网络的基本概念，加深对TCP/IP网络体系结构及各层的功能和工作原理的理解，培养实际的网络方案设计和组网操作的技能，达到巩固计算机网络基础理论、强化学生的实践意识等目的。

而通过网络软件编程的实践，将书本上抽象的概念与具体实现技术结合，深入理解理论课上学习到的计算机网络基本原理和重要协议，通过自己动手编程封装与发送这些协议的数据包，加深对网络协议的理解，掌握协议传输单元的结构和工作原理及其对协议栈的贡献。

**实验仪器：**

实验硬件设备：

**李晓坤：**



图1 李晓坤计算机设备信息

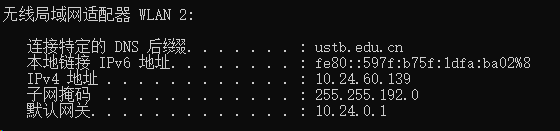


图2 李晓坤无线局域网适配器信息

**王若凡：**



图3 王若凡计算机设备信息

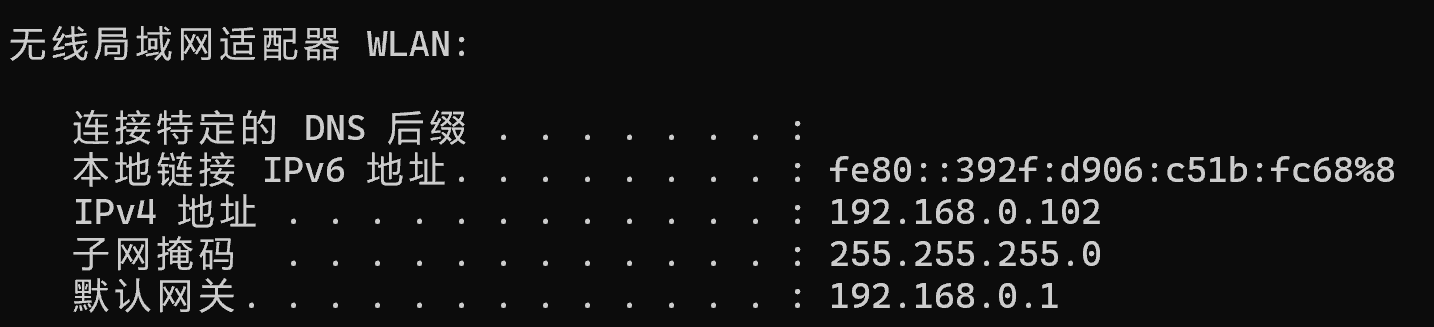


图4 王若凡无线局域网适配器信息

**王宠爱：**



图5 王宠爱计算机设备信息

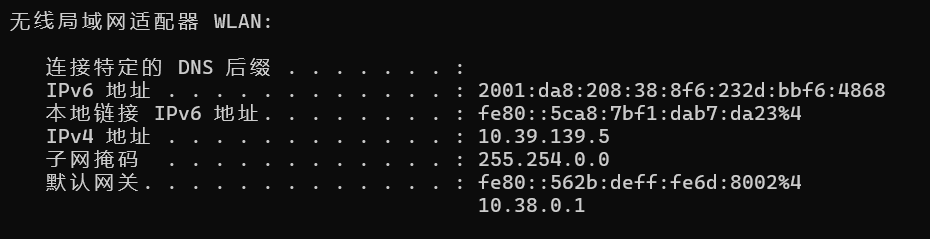


图6 王宠爱无线局域网适配器信息

实验软件要求：

表1 实验软件要求表

|  |  |
| --- | --- |
| 语言 | C++ |
| 操作系统 | Windows 10 家庭中文版，  Windows 11 家庭中文版 |
| 开发工具 | Visual Studio 2022，  Cisco Packet Tracer 8.2.1 |
| 库文件 | Winsock库 |
| 其他工具 | Notepad++文本编辑器 |

小组成员及分工：

表2 题目1小组成员及分工表

|  |  |
| --- | --- |
| 小组成员 | 分工 |
| 李晓坤 | 需求分析、拓扑结构设计、IP地址规划配置、VLAN划分及配置、项目测试、实验报告撰写 |
| 王若凡 | 需求分析、拓扑结构设计、NAT配置、防火墙及访问控制表配置、项目测试、实验报告撰写 |
| 王宠爱 | 需求分析、拓扑结构设计、链路聚合配置、OSPF动态路由协议配置、项目测试、实验报告撰写 |

表3 题目2小组成员及分工表

|  |  |
| --- | --- |
| 小组成员 | 分工 |
| 李晓坤 | 需求分析、整体设计、项目框架搭建、客户端模块实现、AES加解密模块实现、单元测试及整体测试、实验报告撰写 |
| 王若凡 | 需求分析、整体设计、用户命令行界面设计、服务端模块实现、辅助函数功能实现、单元测试及整体测试、实验报告撰写 |
| 王宠爱 | 需求分析、整体设计、端对端交互设计、服务端模块实现、Base64编解码模块、单元测试及整体测试、实验报告撰写 |

**实验原理：**

题目1：介绍涉及的理论知识与相关技术

1、 网络拓扑结构设计

分层设计: 核心层负责高速数据传输和路由，汇聚层提供策略控制和流量管理，接入层提供用户接入。

环形拓扑: 提供冗余路径，当一条路径故障时，数据可以通过另一条路径传输，增强网络的可用性。

链路聚合: 通过将多个物理链路组合成一个逻辑链路，不仅增加了带宽，还提供了故障转移能力。

负载均衡: 在聚合链路上，可以实现流量的负载均衡，优化网络性能。

2、 IP 地址规划及设备端口配置

IP地址分类: 根据网络规模和需求选择合适的IP地址类别（A类、B类、C类等）。

子网划分: 通过子网掩码将大的IP网络划分为多个小的子网，提高IP地址的利用率和管理效率。

静态IP分配: 对于需要固定IP地址的设备或服务，如服务器、主机等，进行静态分配。

3、 VLAN 划分及配置

VLAN工作原理: 在数据帧中添加VLAN标签，使交换机能够识别不同VLAN的数据帧，并进行逻辑隔离。

VLAN间通信: 需要通过三层设备（如三层交换机或路由器）来实现不同VLAN之间的通信。

802.1Q标准: 定义了VLAN数据帧的格式和标签的添加方式。

VLAN Trunking协议: 允许单个物理链路同时传输多个VLAN的数据帧。

4、 NAT配置

NAT工作原理: 将私有IP地址转换为公共IP地址，同时修改数据包中的端口号（PAT）或仅修改IP地址（静态NAT）。

NAT类型选择: 根据网络需求和公共IP地址资源的可用性选择合适的NAT类型。

ACL与NAT结合: 使用ACL定义哪些内部IP地址或IP地址范围可以进行NAT转换，实现更精细的访问控制。

5、 OSPF动态路由协议

OSPF工作原理: 使用Dijkstra算法计算最短路径，通过LSA（链路状态通告）交换网络拓扑信息。

OSPF区域划分: 将网络划分为多个区域，减少路由表的大小和路由计算的复杂性。

OSPF路由器配置: 包括设置OSPF进程ID、区域ID、网络宣告等参数。

OSPF优化技术: 如路由汇总、区域间路由过滤等，用于优化网络性能和简化管理。

6、 防火墙配置及访问控制表（ACL）

防火墙工作原理: 检查网络数据包，根据安全策略允许或拒绝数据包通过。

ACL的作用: 定义访问控制规则，限制网络流量，提高网络安全性。

扩展ACL: 基于源/目的IP地址、端口号、协议等条件定义访问控制规则，实现更精细的访问控制。

题目2：实验原理

文件传输工具是一种用于在计算机之间传输文件的软件工具。其原理基于客户端-服务器模型，客户端负责向服务器发起文件传输请求，服务器则负责接收并处理这些请求。在局域网环境中，文件传输工具通常使用TCP/IP协议进行通信，通过Socket编程实现数据传输。文件传输工具的核心功能包括文件上传和下载，其中上传功能将本地文件发送至服务器，而下载功能则从服务器获取文件到本地。为确保传输过程的安全性，文件传输工具通常会使用加密算法对传输的文件进行加密，以防止文件被非法窃取。文件传输工具的设计和实现需要考虑到网络通信、数据传输、安全加密等方面的技术，以提供高效稳定的文件传输服务。

**实验内容与步骤：**

**题目1：计算机网络系统设计**

（1）实验内容

本项目的实验内容有：设计和构建一个分层的校园网络拓扑结构，确保其具备高可用性和可扩展性，并通过环形拓扑与链路聚合提供冗余。然后，进行详细的IP地址规划和设备端口配置，为校园网络中的每个设备分配独特的IP地址，并合理配置交换机端口。进一步，将实施VLAN的划分与配置，根据校园内不同用户的需求和安全性考虑，设置不同的VLAN（VLAN10与VLAN20）以实现网络逻辑隔离。此外，将图书馆子网即内网部分进行NAT转换，使得内部私有IP能够访问外部公共网络。另外，还需要在核心交换机上配置OSPF动态路由协议，以确保校园网络的路由效率与准确性。最后，对项目工程进行防火墙的配置以及访问控制列表（ACL）的设定，以提升整个网络的安全性能。防火墙上只允许 Web 服务器和邮件服务器可以被网络上的其它计算机访问，FTP 服务器只允许内部网络访问。通过这一系列的实验内容，我们能够成功构建一个高效、安全且可扩展的校园网络环境。

（2）主要步骤

1、任务需求分析

本项目旨在构建一个安全、高效、可扩展的校园网络，满足学校日常教学、办公、科研及图书馆等场所的网络需求。项目将重点关注网络拓扑结构设计、IP地址规划及设备端口配置、VLAN划分及配置、NAT配置、OSPF动态路由协议配置、防火墙配置及访问控制表（ACL）设置，以及网络设备的配置和管理。

具体任务需求分析如下：

网络拓扑结构设计：设计一个分层的网络拓扑结构，包括核心层、汇聚层和接入层，确保网络的高可用性、可扩展性和安全性。在核心层采用环形拓扑结构，通过链路聚合提供冗余和负载均衡，确保网络的高可用性。选择高性能的三层交换机作为核心设备，支持高速数据传输和复杂的路由功能。

IP地址规划及设备端口配置：根据网络规模和子网需求，进行IP地址规划，包括选择合适的IP地址类别和进行子网划分。为每个设备分配唯一的IP地址，并根据连接需求配置交换机端口。

VLAN划分及配置：根据用户需求和安全性要求，规划不同的VLAN，实现网络的逻辑隔离。在交换机上配置VLAN，并为每个VLAN分配相应的端口。配置VLAN间路由，允许不同VLAN之间的数据传输，同时保持安全性。

NAT配置：为图书馆子网配置NAT，实现私有IP地址与公共IP地址的转换，以便访问外部网络。根据需求选择动态NAT或静态NAT，为内部网络用户提供访问外部网络的能力。

结合ACL使用NAT，控制哪些内部IP地址可以通过NAT访问外部网络，确保网络的安全性。

OSPF动态路由协议配置：在核心层交换机上配置OSPF动态路由协议，实现网络的自动路由功能。设置合适的OSPF进程ID和区域ID，确保所有参与OSPF的交换机能够正确交换路由信息。将图书馆子网宣告到OSPF进程中，以便核心层交换机能够学习到该子网的路由信息。

防火墙配置及访问控制表设置：配置防火墙，实现网络的安全防护功能，防止未经授权的访问和攻击。定义防火墙的安全策略，允许或拒绝特定的网络流量通过。结合ACL使用防火墙，根据源/目的IP地址、端口号和协议等条件实施访问控制，提高网络的安全性。

2、设计方案

2.1 网络拓扑结构设计

2.1.1 核心层

采用三台高性能的三层交换机组成环形拓扑结构，设备分别为：

Multilayer Switch0

Multilayer Switch1

Multilayer Switch2

交换机之间使用两条链路进行聚合，提高校园网络主干带宽;

为聚合后的端口分配IP：

Multilayer Switch1 - Multilayer Switch2：port-channel1 - IP地址 192.168.40.0/24

Multilayer Switch0 - Multilayer Switch1：port-channel2 - IP地址 192.168.50.0/24

Multilayer Switch2 - Multilayer Switch0：port-channel3 - IP地址 192.168.60.0/24

2.1.2 汇聚层

连接核心层和接入层，负责汇总接入层交换机的流量。设备为：Multilayer Switch3

Multilayer Switch3 - Multilayer Switch2：IP地址 192.168.100.0/24

2.1.3接入层

连接终端设备的交换机，提供接入服务。为图书馆区域的交换机分别分配一个VLAN。

Switch0：VLAN 10 - IP地址 192.168.10.0/24

Switch1：VLAN 20 - IP地址 192.168.20.0/24

Switch2

2.2 VLAN划分及配置：

根据用户需求，将图书馆用户划分为两个VLAN：

PC1、PC2- VLAN10：对应允许访问Internet的用户

PC3、PC4-VLAN20：对应仅允许访问校内网络的用户

2.3 配置NAT

2.3.1 设置路由规则和访问控制列表（ACL）

在边界路由器上设置路由规则，使用ACL 120匹配内部地址并将其转换为外部接口Serial0/1/0的IP地址，实现数据包流向控制。

针对vlan10和vlan20，设置ACL并配置动态NAT以控制其对Internet的访问，可以限制访问的源地址、目标地址、端口等信息，提高网络安全性。

2.3.2 静态NAT映射

为实现外网主机只能访问WEB服务器和邮件服务器，需要在边界路由器上设置静态NAT映射。静态NAT映射将内部服务器IP地址映射到外部地址，实现外网主机对WEB服务器和邮件服务器的访问，确保整个网络的安全性和功能性。

2.4 OSPF动态路由协议配置

2.4.1 交换机OSPF配置

对SW0、SW1、SW2、SW3四台交换机进行全面配置，将连接的所有网络添加到OSPF路由协议中。通过此配置，交换机能够实时地根据网络状况变化更新路由表，确保数据传输选择最优路径。

2.4.2 边界路由器R0的OSPF配置

考虑到网络安全因素，边界路由器R0将仅在本地接口IP地址所属的网络段（192.168.200.0/24）上参与OSPF路由信息的交换。R0将不会对外通告内部网络的详细路由信息，有效隔离和保护内部网络结构。

2.4.3 OSPF区域划分

将所有参与OSPF路由协议的交换机和路由器划分到同一个OSPF区域内。这种设计可以确保交换机和路由器之间直接交换路由信息，无需通过区域间的复杂路由汇总或过滤，简化了网络管理并提高了路由效率。

2.5 防火墙配置

2.5.1 访问控制应用与接口安全级别配置

为确保ACL 120的有效实施，需在内部和外部接口上双向应用该访问控制列表。同时，根据网络安全的最佳实践，为各接口配置相应的安全级别：

* 外部接口：配置安全级别为0，表示该接口连接的网络环境安全性较低，需实施严格的访问控制。
* 内部接口：配置安全级别为100，表示该接口连接的网络环境为内部受信任网络，安全性相对较高。但仍需通过ACL 120进行细粒度的访问控制，以确保网络安全。

2.5.2 默认路由配置

为确保网络流量的有效转发，需设置默认路由，将所有非特定路由的流量导向外部网关。

2.5.3 访问控制列表（ACL）120配置

允许内部网络和外部网络访问Web服务器和邮件服务器，确保内部用户能够正常浏览和使用该服务器提供的服务。

拒绝除内部网络外的所有其他网络访问FTP服务器，严防未经授权的外部访问，保障服务器及数据的安全。

除上述特定规则外，允许所有其他流量通过，以确保网络的正常运行和服务的连续性。

3、实施方案：

3.1 拓扑图设计及IP规划

3.1.1 拓扑图设计

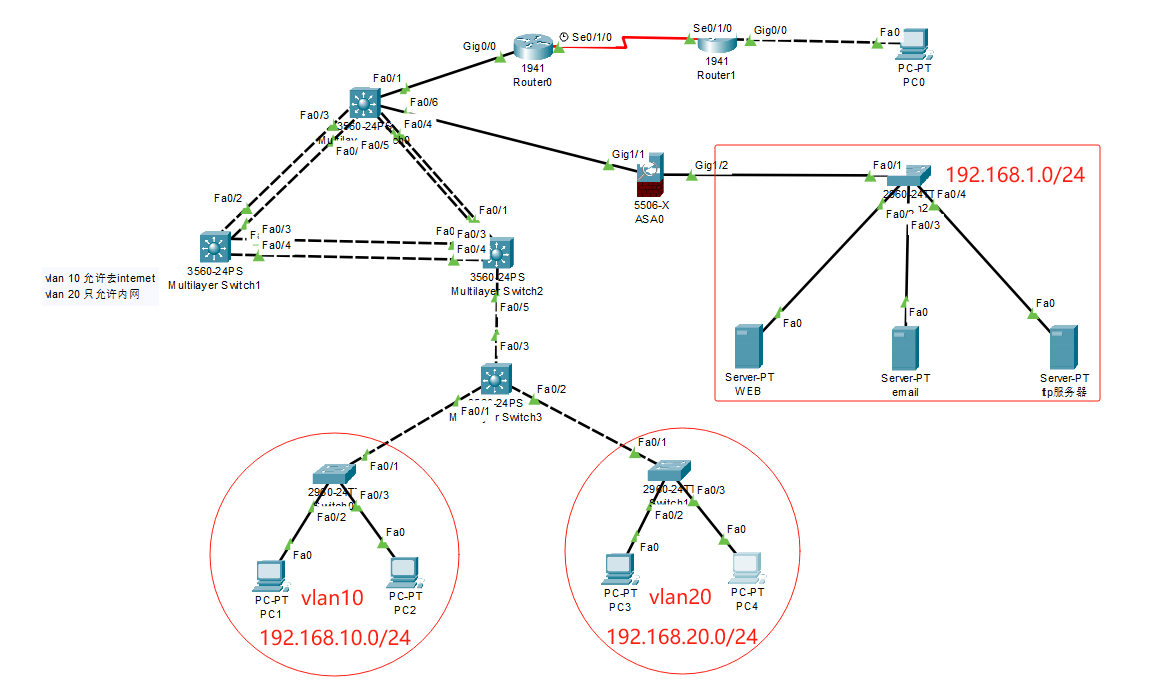


图7 网络拓扑结构图

3.1.2 IP规划表

根据网络拓扑图及实验要求，在内网中规划了2个 Vlan，其中 Vlan10内的用户允许通过 NAT 方式访问 Internet，vlan20内的用户禁止访问 Internet。

表4 网络中断地址规划表

|  |  |  |  |
| --- | --- | --- | --- |
| 设备 | IP地址 | 网关 | VLAN |
| PC0 | 10.1.1.2 | 10.1.1.1 | 无 |
| PC1 | 192.168.10.2 | 192.168.10.1 | 10 |
| PC2 | 192.168.10.3 | 192.168.10.1 | 10 |
| PC3 | 192.168.20.2 | 192.168.10.1 | 20 |
| PC4 | 192.168.20.3 | 192.168.10.1 | 20 |
| WEB服务器 | 192.168.1.11 | 192.168.1.254 | 无 |
| FTP服务器 | 192.168.1.22 | 192.168.1.254 | 无 |
| email服务器 | 192.168.1.33 | 192.168.1.254 | 无 |

表5 网络设备端口地址规划

|  |  |  |  |
| --- | --- | --- | --- |
| 设备 | 端口 | | IP地址 |
| Multilayer Switch0 | Fa0/1 | | 192.168.200.1 |
| port-channel2 | Fa0/2 | 192.168.50.2 |
| Fa0/3 |
| port-channel3 | Fa0/4 | 192.168.60.2 |
| Fa0/5 |
| Fa0/6 | | 192.168.2.2 |
| Multilayer Switch1 | port-channel2 | Fa0/1 | 192.168.40.2 |
| Fa0/2 |
| port-channel1 | Fa0/3 | 192.168.50.1 |
| Fa0/4 |
| Multilayer Switch2 | port-channel3 | Fa0/1 | 192.168.40.1 |
| Fa0/2 |
| port-channel1 | Fa0/3 | 192.168.60.1 |
| Fa0/4 |
| Fa0/5 | | 192.168.100.2 |
| Multilayer Switch3 | Fa0/1 | |  |
| Fa0/2 | |  |
| Fa0/3 | | 192.168.100.1 |
| Router0 | serial0/1/0 | | 202.204.100.1 |
| Gi0/0 | | 192.168.200.2 |
| Router1 | serial0/1/0 | | 202.204.100.2 |
| Gi0/0 | | 10.1.1.1 |
| ASA0 | Gig1/1 | | 192.168.2.1 |
| Gig1/2 | | 192.168.1.254 |

表6 网络地址转换 NAT 地址映射表

|  |  |  |
| --- | --- | --- |
| 设备 | 内网地址 | 公网地址 |
| PC1 | 192.168.10.2 | 202.204.100.1 |
| PC2 | 192.168.10.3 | 202.204.100.1 |
| WEB服务器 | 192.168.1.11 | 202.204.100.11 |
| email服务器 | 192.168.1.22 | 202.204.100.22 |

3.1.3 设备配置

二层交换机关键配置命令

交换机与终端设备连接的端口设置为access模式并划分vlan，与三层交换机相连的端口设置为trunk模式，保证通信正常进行。

* Switch0

vlan 10

exit

int range f 0/2-3

switchport mode access

switchport access vlan 10

exit

int f 0/1

switchport mode trunk

switchport trunk allowed vlan all

exit

Switch1的配置与Switch0类似

三层交换机端口配置

交换机与二层交换机相连的端口通过划分vlan进而配置IP地址，与网络中其它设备相连的端口在开启路由功能后可直接配置IP地址，实现更灵活、高效的网络通信

* Multilayer Switch3

vlan 10

int vlan 10

ip add 192.168.10.1 255.255.255.0

exit

vlan 20

int vlan 20

ip add 192.168.20.1 255.255.255.0

exit

int f 0/3

no switchport

ip address 192.168.100.1 255.255.255.0

no shutdown

exit

* Multilayer Switch0

int f0/1

no switchport

ip address 192.168.200.1 255.255.255.0

no shutdown

exit

int f0/6

no switchport

ip address 192.168.2.2 255.255.255.0

exit

* Multilayer Switch2

int f0/5

no switchport

ip address 192.168.100.2 255.255.255.0

exit

路由器端口配置

* 出口路由器Router0

int Gi0/0

ip add **192.168.200.2** 255.255.255.0

no shutdown

exit

int Serial0/1/0

ip add **202.204.100.1** 255.255.255.0

//发送一个时钟信号给另一端的路由器，确保数据传输的同步和正常进行

clock rate 64000

no shutdown

Router1的配置与Router0类似

3.2 NAT配置

在出口路由器 Router0 上配置 NAT ，实现vlan10内的用户可通过动态NAT 方式访问 Internet，vlan20内的用户禁止访问 Internet，同时通过静态NAT映射，实现内网中的WEB服务器和邮件服务器可以被外部网络访问

int Gi0/0

ip nat inside

int serial 0/1/0

ip nat outside

//将内部网络中的所有符合访问控制列表规则的数据包进行动态NAT转换为路由器Serial0/1/0接口的IP地址，并共享同一个外部IP地址

ip nat inside source list 120 interface Serial0/1/0 overload

//将内部网络中的WEB服务器的IP地址映射为外部网络中的地址202.204.100.11

ip nat inside source static 192.168.1.11 202.204.100.11

//将内部网络中的邮件服务器的IP地址映射为外部网络中的地址202.204.100.22

ip nat inside source static 192.168.1.22 202.204.100.22

//将所有未知目的地的数据包发送到Router1的Serial0/1/0端口

ip route 0.0.0.0 0.0.0.0 202.204.100.2

//将访问服务器区域的数据包发送到下一跳地址

ip route 192.168.1.0 255.255.255.0 192.168.200.1

//拒绝来自vlan20的访问

access-list 120 deny ip 192.168.20.0 0.0.0.255 any

//允许来自vlan10的访问

access-list 120 permit ip 192.168.0.0 0.0.255.255 any

3.3 链路聚合

* Multilayer Switch0

//创建聚合端口port-channel2

int port-channel 2

int range f0/2-3

switchport mode trunk

//将f0/2、f0/3端口聚合到port-channel2端口中

channel-group 2 mode on

int port-channel 3

int range f0/4-5

switchport mode trunk

channel-group 3 mode on

//为聚合端口分配IP地址

int port-channel 1

no switchport

ip add 192.168.40.1 255.255.255.0

exit

int port-channel 3

no switchport

ip add 192.168.60.1 255.255.255.0

Multilayer Switch1、Multilayer Switch2的配置与Multilayer Switch0类似。

3.4 OSPF动态路由协议

* Multilayer Switch0

router ospf 1

//在OSPF协议中启用路由接口，并将设备上的所有接口都加入到OSPF中

network 0.0.0.0 255.255.255.255 area 0

exit

Multilayer Switch1、Multilayer Switch2、Multilayer Switch3的配置与Multilayer Switch0类似，所有参与OSPF路由协议的交换机和路由器划分到同一个OSPF区域内。

为了保证网络中的终端设备能够正常访问服务器区域的设备，需要在Multilayer Switch0上设置一条静态路由，将目的地址为192.168.1.0/24的IP数据包发送到防火墙的Gig1/1端口：

ip route 192.168.1.0 255.255.255.0 192.168.2.1

* Router0

router ospf 1

network 192.168.200.2 0.0.0.0 area 0

exit

边界路由器R0仅在本地接口IP地址所属的网络段（192.168.200.0/24）上参与OSPF路由信息的交换，故只需将本地接口IP添加到OSPF路由协议中。同时将外网接口排除在OSPF区域之外，以确保内部网络的路由信息不会被传播到外部网络。

3.5 防火墙上只允许 Web 服务器和邮件服务器可以被网络上的其它计算机访问，FTP 服务器只允许内部网络访问；

1.定义了两个网络接口GigabitEthernet1/1和GigabitEthernet1/2，分别命名为out和in，并设置了它们的安全级别和IP地址。

2.配置了静态路由，使防火墙能够向外部网络发送数据包。

3.创建了一个扩展访问控制列表（ACL）120，用于控制网络流量。

4.将访问控制列表应用到防火墙的接口上，以实施访问控制。

* ASA0

# 进入GigabitEthernet1/1接口配置 命名接口为out 设置安全级别为0（通常外部接口安全级别较低）设置接口的IP地址和子网掩码

interface GigabitEthernet1/1

nameif out

security-level 0

ip address 192.168.2.1 255.255.255.0

# 进入GigabitEthernet1/2接口配置

interface GigabitEthernet1/2

nameif in

security-level 100

ip address 192.168.1.254 255.255.255.0

# 配置静态默认路由，指向下一跳IP地址192.168.2.2

route out 0.0.0.0 0.0.0.0 192.168.2.2 1

# 配置访问控制列表120，首先允许内部网络192.168.0.0/16访问ftp服务器192.168.1.33，然后拒绝任何其他地址访问ftp服务器，并允许其他所有IP流量

access-list 120 extended permit ip 192.168.0.0 255.255.0.0 host 192.168.1.33

access-list 120 extended deny ip any host 192.168.1.33

access-list 120 extended permit ip any any

access-list 120 extended permit tcp any any

access-list 120 extended permit icmp any any

access-list 120 extended permit udp any any

# 将访问控制列表120应用于in和out接口的入站和出站方向

access-group 120 in interface in

access-group 120 in interface out

access-group 120 out interface out

access-group 120 out interface in

**题目2：子网内文件传送**

（1）实验内容

根据实验指导书中的要求，我们需要设计并实现一个局域网内部的文件传送工具，使用TCP协议进行可靠文字传输，以命令行或图形界面运行，不同节点上文件自动同步。实验内容将围绕局域网文件传输工具展开，经需求分析后，总结提炼出以下几点：

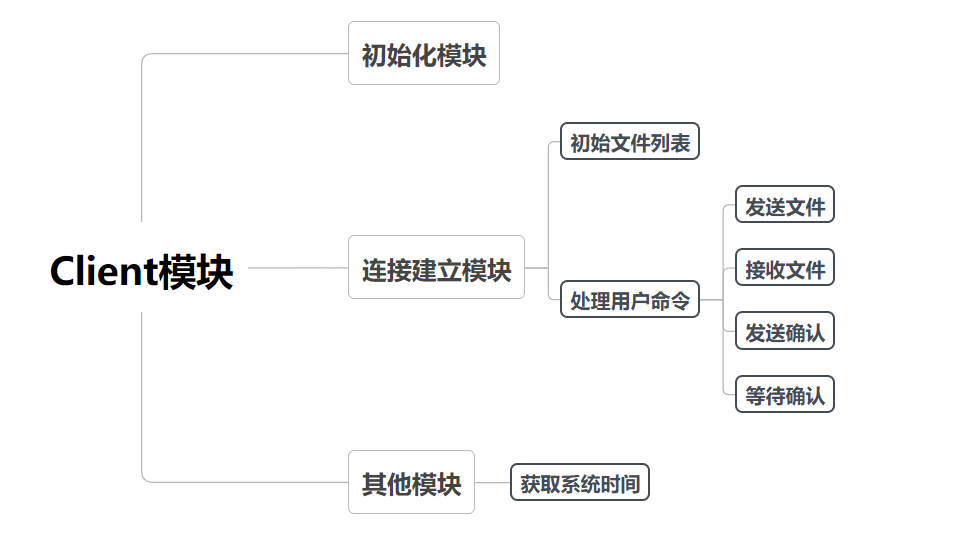
* 设计并实现一个局域网内的文件传输工具
* 完成命令行或图形界面
* 完成局域网内主机之间的文件传送
* 使用Socket API技术来实现
* 实现不同节点文件同步
* 对文件进行加密，保证安全性
* 将以往文件传输记录保存在文件中
* 保证文件传输的稳定和完整

（2）主要步骤

1 模块划分与功能设计

1.1 客户端

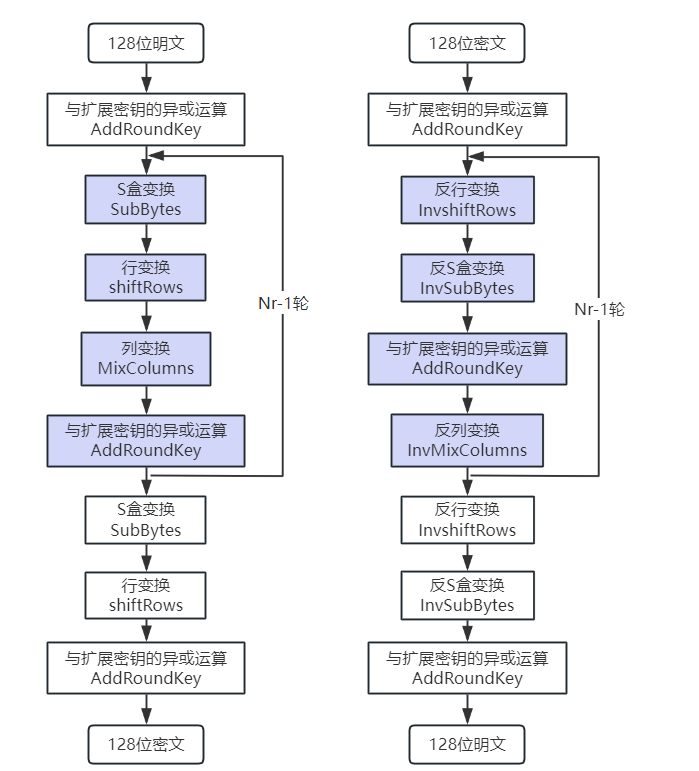
1.1.1 Client模块



图？Client模块结构图

如上图所示，这是Client模块的结构分解图，Client模块下有初始化模块、连接建立模块和其他模块。其中，初始化模块主要进行Winsock的创建和初始化以及套接字的创建；连接建立模块主要进行文件列表的初始化和用户命令的处理，即用户有关的操作和文件收发工作均由该模块完成；其他模块主要完成一些辅助功能如获取系统时间、获取当前目录文件列表等。

1.1.2 AES模块



图？ AES模块结构图

如上图所示，AES模块主要完成对文件的加解密任务，内置会话16字节密钥，方便用户对即将发送的文件进行加密和对接收到的文件进行解密。虽然可以基于Openssl进行AES加解密，但是我们仍选择比较传统的c++代码的方式进行书写。

1.1.3 Base64模块



图？Base64模块结构图

如上图所示，这是Base64模块的结构图，该模块为用户提供两个接口：Base64编码接口与Base64解码接口，主要完成对加密后的数据进行编解码，保证数据的完整性和规范性。

1.1.4 main模块



图？ main模块结构图

如上图所示，这是main模块的结构图，该模块是用户端程序的入口，主要完成Client的实例化与连接的建立，而后续的文件传输任务则是Client对象本身的功能。

1.2 服务端

1.2.1 Server模块



图？ Server模块结构图

如上图所示，这是Server模块的结构图，Server模块下有初始化模块、连接建立模块和其他模块。其中，初始化模块主要进行Winsock的创建和初始化以及套接字的创建、绑定和监听；连接建立模块主要进行文件列表的初始化和用户连接的建立，这是一个线程模块，也就是说Server会创建子线程处理用户的命令；其他模块主要完成一些辅助功能如获取系统时间、获取当前目录文件列表、维护用户文件列表等工作。

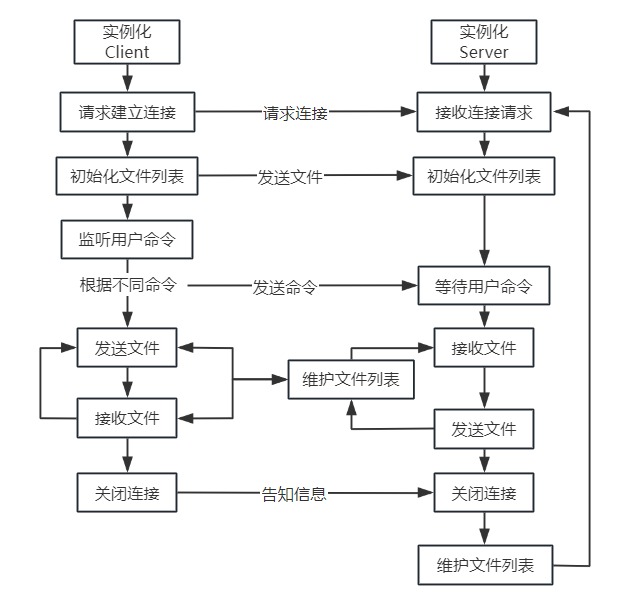
1.2.2 main模块



图？ main模块结构图

如上图所示，这是main模块的结构图，该模块是服务端程序的入口，主要完成Server的实例化与连接的建立，而后续处理用户的文件传输任务则是Server对象本身的功能。

2 模块间的联系



图？ 模块间的联系图

如上图所示，这是客户端与服务端的模块联系图，双方依靠相关命令字符进行交互。首先，对于客户端，实例化Client后会向服务端发送连接请求，成功建立连接后开始向服务端发送自己的初始文件，对于服务端，实例化Server后会接收来自用户的连接请求，成功建立连接后开始接收客户发送的初始文件，并相应更新文件列表，创建子线程处理该用户；然后，对于客户端，开始监听用户的命令，此时用户可以通过命令行界面键入命令，该命令会被发送给服务端，对于服务端，开始等待用户的命令，接收到用户的命令后，做出对应的行为；接下来，客户端和服务端根据事先约定好的命令进行相应动作，不同于普通socket通信，客户端在发送文件前会利用AES算法对文件进行加密，服务端会在接收文件后维护用户文件列表；当客户端关闭连接时，会发送相应提示信息给服务端，服务端接收后更新文件列表并结束该子线程，同时主线程接收连接请求。

3 关键问题的解决方法

在这一环节，将针对前文需求分析环节的诸多需求，提出相应的解决方法，对于部分问题，可能存在效率更高的解决方法，这会在“设计中存在的问题”这一环节中进行阐述和分析。

表？ 关键问题的解决方法展示表

|  |  |
| --- | --- |
| 需求 | 解决方法 |
| 完成命令行或图形界面 | 由于本项目结构并不复杂且需求简单，因此我们选择控制台输入输出来模拟用户界面。通过输出特定的字符和控制台命令等，实现简单的界面效果。 |
| 完成局域网内主机之间的文件传送 | 我们选择C++中的socket编程，实现客户端和服务端之间的文件传输。基于TCP协议，服务端监听指定端口，客户端连接并发送文件。 |
| 使用Socket API技术来实现 | 在学习了实验指导书和其他相关网络资料后，我们决定按照下面的步骤进行解决：  1、创建服务端：在一个主机上创建一个服务端程序，监听指定的端口，等待客户端连接。  2、创建客户端：在另一个主机上创建一个客户端程序，连接到服务端指定的IP地址和端口。  3、建立连接：客户端和服务端建立连接后，可以进行文件传输。可以使用TCP协议保证可靠性，也可以使用UDP协议。  4、传输文件：客户端将要传输的文件发送给服务端，服务端接收文件并保存到本地。  5、错误处理：处理连接错误、传输错误等异常情况，保证程序稳定可靠。 |
| 实现不同节点文件同步 | 通过查阅相关资料，我们确定同步策略为单向同步，由Client和Server维护自身的列表并完成对文件的监控，同时考虑到可能出现的错误和异常，我们需要增加相应的处理机制。 |
| 对文件进行加密，保证安全性 | 基于密码学课程学习的知识，我们选择AES算法对文件进行加解密操作。作为一种对称加密算法，其安全性取决于密钥的安全性，我们在进行密钥的配置时，采取用户事先约定的方式，关于其改进也会在后续环节中阐述。 |
| 将以往文件传送记录保存在文件中 | 我们在Client和Server中文件流操作对象，负责对文件进行操作。将每次文件传送的记录保存到一个文件中，可以选择文本文件或者其他格式的文件，每条记录占据一行，并记录相应的日期。 |
| 保证文件传输的稳定和完整 | 基于计算机网络课程的基础知识，我们采取了文件分块、数据校验和等待确认机制，这些机制能够保证文件传输过程具有一定的稳定性和有序性。 |

4 详细设计

4.1 客户端

4.1.1 数据结构

（一）Client类的声明

受限于报告篇幅，我们将类的成员变量和成员函数的含义记录在源代码中，如下表所示。

表？Client类的声明

|  |
| --- |
| class Client {  public:  Client();// 构造函数，用于初始化客户端对象  void connectToServer();// 连接服务器  void sendFile(std::string filePath);// 发送文件  void recvFile();// 接收文件  void sendOneFile(std::string filePath);// 发送单个文件  void recvOneFile();// 接收单个文件  vector<string> getFilesInCurrentDirectory();// 获取当前目录下的所有文件  void InitSendFile();// 初始化发送文件  void recvFileThread();// 接收文件线程  void sendConfirm();// 发送确认信息  void recvConfirm();// 接收确认信息  std::string getCurrentTimeAsString();// 获取当前时间的字符串表示  private:  SOCKET clientSocket;// 客户端套接字  AES aes;// AES加解密对象  vector<string> files;// 当前目录下的所有文件列表  ofstream log;// 日志文件  int sync = 0;// 同步标志  }; |

（二）AES类的声明

表？AES类的声明

|  |
| --- |
| class AES {  public:  // 密钥  \_byte key[16];  // 加密后编码  void encryptAndEncode(\_byte[4 \* 4], string fileName, string en\_fileName);  // 解码后解密  void decodeAndDecrypt(\_byte[4 \* 4], string fileName, string de\_fileName);  // 加密文件  void encryptFile(\_byte[4 \* 4], string fileName, string en\_fileName);  // 解密文件  void decryptFile(\_byte[4 \* 4], string fileName, string de\_fileName);  // 加密  void encrypt(\_byte[4 \* 4], \_byte[4 \* 4]);  // 解密  void decrypt(\_byte[4 \* 4], \_byte[4 \* 4]);  private:  \_word K[4 \* (Nr + 1)];  // 四个字节合成一个字  \_word Word(\_byte&, \_byte&, \_byte&, \_byte&);  // 对输入word中的每一个字节进行S-盒变换  \_word SubWord(\_word);  // 按字节 循环左移一位,即把[a0, a1, a2, a3]变成[a1, a2, a3, a0]  \_word RotWord(\_word);  // S盒变换 - 前4位为行号，后4位为列号  void SubBytes(\_byte[4 \* 4]);  // 行变换 - 按字节循环移位  void ShiftRows(\_byte[4 \* 4]);  // 有限域上的乘法 GF(2^8)  \_byte GFMul(\_byte a, \_byte b);  // 列变换  void MixColumns(\_byte[4 \* 4]);  // 轮密钥加变换 - 将每一列与扩展密钥进行异或  void AddRoundKey(\_byte[4 \* 4], \_word[4]);  // 密钥扩展函数 - 对128位密钥进行扩展得到 w[4 \* (Nr + 1)]  void KeyExpansion(\_byte[4 \* Nk], \_word[4 \* (Nr + 1)]);  // 逆S盒变换  void InvSubBytes(\_byte[4 \* 4]);  // 逆行变换 - 以字节为单位循环右移  void InvShiftRows(\_byte[4 \* 4]);  void InvMixColumns(\_byte[4 \* 4]);  // 将一个char字符数组转化为二进制,存到一个 byte 数组中  void charToByte(\_byte[16], const char[16]);  // 将连续的128位分成16组，存到一个 byte 数组中  void divideToByte(\_byte[16], bitset<128>&);  // 将16个 byte 合并成连续的128位  bitset<128> mergeByte(\_byte[16]);  // 轮常数，密钥扩展中用到。（AES-128只需要10轮）  \_word Rcon[10] = { 0x01000000, 0x02000000, 0x04000000, 0x08000000, 0x10000000,  0x20000000, 0x40000000, 0x80000000, 0x1b000000, 0x36000000 };  \_byte S\_Box[16][16];  \_byte Inv\_S\_Box[16][16];  \_byte coff[16];  }; |

4.1.2 核心算法

（一）客户端与服务端建立连接

客户端与服务端建立连接，通过connect()函数来建立TCP连接，后不断监听用户的命令，根据用户键入命令的不同来实现相应的操作，如“send”、“show”、“flush”等操作。

表？ conectToServer函数

|  |
| --- |
| void Client::connectToServer() {  sockaddr\_in serverAddr;  serverAddr.sin\_family = AF\_INET;  serverAddr.sin\_addr.s\_addr = inet\_addr(SERVER\_ADDRESS);  serverAddr.sin\_port = htons(PORT);  if (connect(clientSocket, (SOCKADDR\*)&serverAddr, sizeof(serverAddr)) == SOCKET\_ERROR) {  std::cerr << "错误：无法连接到服务器。" << std::endl;  closesocket(clientSocket);  WSACleanup();  exit(EXIT\_FAILURE);  }  std::cout << "已连接到服务器。" << std::endl;  log.open("clientLog.txt", ios::app);  log << getCurrentTimeAsString() << "：已连接到服务器。" << std::endl;  log.close();  // 初始化发送文件  cout<< "初始化发送文件..." << endl;  InitSendFile();  cout << "初始化发送文件完成。" << endl;    log.open("clientLog.txt", ios::app);  log << getCurrentTimeAsString() << "：初始化发送文件完成。" << std::endl;  log.close();  //无限循环处理用户命令  while (true)  {  std::string command;  std::cout << "请输入命令：";  std::cin >> command;  if (command == "exit") {  log.open("clientLog.txt", ios::app);  log << getCurrentTimeAsString() << "：退出客户端。" << std::endl;  log.close();  break;  }  else if (command == "send") {  // 发送"send"字段，告知服务端接下来的操作  std::string message = "send";  send(clientSocket, message.c\_str(), message.size(), 0);  // 等待确认  recvConfirm();  // 输入文件名  std::string fileName;  std::cout << "请输入文件名：";  std::cin >> fileName;  // 发送文件名  send(clientSocket, fileName.c\_str(), fileName.size(), 0);  // 等待确认  recvConfirm();  // 将该文件发送给服务端  sendOneFile(fileName);  log.open("clientLog.txt", ios::app);  log << getCurrentTimeAsString() << "：进行send操作，发送文件：" << fileName<<"。" << std::endl;  log.close();  }  else if (command == "show") {  //查看当前目录下的文件列表  std::cout << "----------------------------------当前目录----------------------------------" << std::endl;  system("dir");  std::cout << "----------------------------------------------------------------------------" << std::endl;    log.open("clientLog.txt", ios::app);  log << getCurrentTimeAsString() << "：进行show操作，查看本地文件列表。" << std::endl;  log.close();  }  else if (command == "flush")  {  // 要完成2个任务：1、根据当前目录下的文件，发送新文件给服务端；2、接收服务端的新文件  // 发送“flush”字段，告知服务端接下来的操作  std::string message = "flush";  send(clientSocket, message.c\_str(), message.size(), 0);  // 等待确认  recvConfirm();  // 更新列表并发送相应文件  InitSendFile();    // 接下来是接收有关的操作  // 收到即将接收文件的数量  int numFiles;  cout<< "准备接收文件数量..." << endl;  recv(clientSocket, (char\*)&numFiles, sizeof(int), 0);  std::cout << numFiles << endl;  // 发送确认  sendConfirm();  // 循环接收文件  for (int i = 0; i < numFiles; i++)  {  recvOneFile();  }  log.open("clientLog.txt", ios::app);  log << getCurrentTimeAsString() << "：进行flush操作，发送全部文件到服务端并更新文件列表。" << std::endl;  log.close();  }  else {  std::cerr << "错误：无效的命令。" << std::endl;  }  }  closesocket(clientSocket);  WSACleanup();  std::cout << "已断开连接。" << std::endl;  } |

（二）初始化发送文件

客户端与服务端建立连接后，初始化发送文件，在正式发送文件之前调用，用于准备正式发送文件的环境。

表？ InitSendFile函数

|  |
| --- |
| // 初始化发送文件  void Client::InitSendFile()  {  // 获取当前目录下的文件列表  files = getFilesInCurrentDirectory();  // 打印文件列表  std::cout << "------------------文件列表-----------------------------" << std::endl;  for (auto file : files)  {  std::cout << file << std::endl;  }  std::cout << "-------------------------------------------------------" << std::endl;  // 发送文件数量  int numFiles = files.size();  send(clientSocket, (char\*)&numFiles, sizeof(int), 0);  // 等待确认  recvConfirm();  cout << "目录下文件数量：" << files.size() << endl;  // 发送文件  for (const auto& file : files) {  sendOneFile(file);  }  cout << "目录下文件数量：" << files.size() << endl;  } |

（三）AES加解密

受限于报告篇幅，这里只展示AES加密算法。AES是一种对称加密算法，用于保护数据的安全性。它使用固定长度的密钥（128 位）对数据进行加密和解密。AES 算法通过多轮操作（10 轮）对数据块进行混淆和置换，从而实现高效的加密和解密过程。

表？ AES核心加密函数

|  |
| --- |
| // 加密  void AES::encrypt(\_byte in[4 \* 4], \_byte usekey[4 \* 4]) {  KeyExpansion(usekey, K);  \_word key[4];  for (int i = 0; i < 4; i++) {  key[i] = K[i];  }  AddRoundKey(in, key);  for (int i = 1; i < Nr; i++) {  SubBytes(in);  ShiftRows(in);  MixColumns(in);  for (int j = 0; j < 4; j++) {  key[j] = K[4 \* i + j];  }  AddRoundKey(in, key);  }  SubBytes(in);  ShiftRows(in);  for (int i = 0; i < 4; i++) {  key[i] = K[4 \* Nr + i];  }  AddRoundKey(in, key);  } |

4.2 服务端

4.2.1 数据结构

受限于报告篇幅，我们将类的成员变量和成员函数的含义记录在源代码中，如下表所示。

表？ Server类的声明

|  |
| --- |
| class Server {  public:  Server();// 构造函数，初始化服务端对象  void acceptConnections();// 服务端开始监听连接  vector<string> getFilesInCurrentDirectory();// 获取当前目录下的所有文件  void InitFileList(SOCKET clientSocket);// 初始化用户文件列表  void SendFile(SOCKET clientSocket, string fileName);// 发送文件  void RecvFile(SOCKET clientSocket);// 接收文件  void refreshAllFiles();// 刷新所有用户的文件列表  void sendConfirm(SOCKET clientSocket);// 发送确认信息  void recvConfirm(SOCKET clientSocket);// 接收确认信息  std::string getCurrentTimeAsString();// 获取当前时间  void txtToShowFileList();// 将文件列表写入文件  private:  SOCKET listenSocket;// 服务端监听套接字  void handleClient(SOCKET clientSocket);// 处理客户端连接  // 服务器建立连接的用户列表  std::vector <SOCKET> clientList;  // 用户文件列表  std::unordered\_map<SOCKET, std::vector<std::string>> userFilesMap;  // 所有用户的文件的并集  std::vector<string> allFiles;  ofstream log;// 日志文件  int sync = 0;// 同步标志  }; |

4.2.2 核心算法

（一）子线程处理客户信息

handClient()是一个核心函数，用于处理客户端的请求，在服务端的网络通信中，接收客户端发送的数据，并根据数据内容执行相应的操作，其核心算法主要包括以下步骤：

（1）接收数据：使用 recv 函数从客户端接收数据，通常是一个请求命令或数据包。

（2）解析请求：根据接收到的数据内容，判断客户端请求的类型，可能是文件传输、命令执行等。

（3）执行操作：根据请求类型执行相应的操作，比如如果是文件传输请求，可能会调用文件传输函数进行处理；如果是命令执行请求，可能会调用系统函数执行相应的操作。

（4）发送响应：处理完客户端的请求后，根据实际情况向客户端发送响应，通知客户端请求的处理结果。

（5）循环处理：在网络通信中，通常需要循环调用 handleClient 函数来处理连续的客户端请求，直到客户端断开连接或其他条件触发结束。

表？ handClient函数

|  |
| --- |
| void Server::handleClient(SOCKET clientSocket) {  //在无限循环中接受来自client的文件数据，并转发给其他client  while (true)  {  char buffer[1024];  int bytesReceived = recv(clientSocket, buffer, sizeof(buffer), 0);  if (bytesReceived <= 0) {  // 客户端断开连接，从客户端列表中移除并结束循环  clientList.erase(std::remove(clientList.begin(), clientList.end(), clientSocket), clientList.end());  // 客户端断开连接，从用户文件列表中移除  userFilesMap.erase(clientSocket);  break;  }  // 发送确认  sendConfirm(clientSocket);  // 将接收到的数据转换为字符串  std::string receivedMessage(buffer, bytesReceived);  // 如果收到的消息是"send"，则进行发送操作  if (receivedMessage == "send") {  // 进行相应操作  // 接收文件名  char buffer[BUFFER\_SIZE] = { 0 };  int bytesRead;  bytesRead = recv(clientSocket, buffer, BUFFER\_SIZE, 0);  if (bytesRead == SOCKET\_ERROR) {  std::cerr << "错误：无法接收文件名。" << std::endl;  return;  }  else if (bytesRead == 0) {  std::cout << "服务器已断开连接。" << std::endl;  return;  }  // 记录文件名  std::string fileName = buffer;  int temp = 0;    log.open("serverLog.txt", ios::app);  log << getCurrentTimeAsString() << "："<<clientSocket<<"进行send操作，发送文件："<<fileName <<"。" << std::endl;  log.close();  //如果文件名已经在该用户文件列表中，则temp改为1  for (auto& file : userFilesMap[clientSocket])  {  if (file == fileName)  {  temp = 1;  break;  }  }  // 发送确认  sendConfirm(clientSocket);  // 接收文件  RecvFile(clientSocket);  // 如果该文件确实是新文件，则不从文件列表中删除，否则删除最后一个文件  if (temp == 1)  {  userFilesMap[clientSocket].pop\_back();  }  //刷新所有用户文件列表  refreshAllFiles();  txtToShowFileList();  }  // 如果接收到的消息是"flush"，则刷新用户界面，更新其文件列表  else if (receivedMessage == "flush")  {  // 刷新用户文件列表  InitFileList(clientSocket);  // 更新所有用户文件列表  refreshAllFiles();  // 统计需要向当前用户发送的文件数量  int numFilesToSend = 0;  for (auto& fileName : allFiles)  {  numFilesToSend++;  }  cout<<numFilesToSend<<endl;  // 将需要向当前用户发送的文件数量发送给当前用户  send(clientSocket, (char\*)&numFilesToSend, sizeof(int), 0);  cout<<"已发送文件数量，等待确认...."<<endl;  // 等待确认  recvConfirm(clientSocket);    // 清空当前用户的文件列表  //userFilesMap[clientSocket].clear();  // 向当前用户发送allFiles  for (auto& fileName : allFiles)  {  // 将该文件添加进入当前用户的文件列表  //userFilesMap[clientSocket].push\_back(fileName);  // 发送该文件给当前用户  SendFile(clientSocket, fileName);  }  log.open("serverLog.txt", ios::app);  log << getCurrentTimeAsString() << "：" << clientSocket << "进行flush操作，发送其全部文件并更新文件列表。" << std::endl;  log.close();  // 刷新所有用户文件列表  refreshAllFiles();  txtToShowFileList();  }  // 如果是"exit",则断开连接  else if (receivedMessage == "exit")  {  txtToShowFileList();  break;  }  // 如果是"show"，不做处理  else if (receivedMessage == "show")  {  txtToShowFileList();  // 记录日志  log.open("serverLog.txt", ios::app);  log << getCurrentTimeAsString() << "：" << clientSocket << "进行show操作，显示文件列表。" << std::endl;  log.close();  continue;  }  }  // 客户端断开连接时，从客户端列表中移除  clientList.erase(std::remove(clientList.begin(), clientList.end(), clientSocket), clientList.end());  // 客户端断开连接，从用户文件列表中移除  userFilesMap.erase(clientSocket);  // 关闭套接字  closesocket(clientSocket);  log.open("serverLog.txt", ios::app);  log << getCurrentTimeAsString() << "：" << clientSocket << "与服务器断开连接。" << std::endl;  log.close();  } |

（二）初始化接收文件

该核心算法用于初始化客户端的文件列表，具体步骤如下：

（1）从客户端接收一个整数，表示客户端拥有的文件数量。

（2）发送确认消息给客户端，通知客户端可以开始发送文件信息。

（3）清空当前客户端在服务器端记录的文件列表，为接收新的文件信息做准备。

（4）循环接收客户端发送的每个文件信息，具体操作由 RecvFile 函数完成。

表？ 服务端的初始接收文件函数

|  |
| --- |
| // 第一次与客户建立连接时同步文件列表和加密后的文件  void Server::InitFileList(SOCKET clientSocket)  {  // 接收客户的文件数量  int numFiles;  recv(clientSocket, (char\*)&numFiles, sizeof(int), 0);  // 发送确认  sendConfirm(clientSocket);    // 清空用户文件列表  userFilesMap[clientSocket].clear();  // 循环接收文件  for (int i = 0; i < numFiles; i++)  {  RecvFile(clientSocket);  }  } |

5 模块测试与总体测试设计

在这个环节，将对单个模块和总体的测试方案进行设计，而测试过程和结果分析将在后续的“实验结果与分析”这一环节中展示。

5.1 客户端

（1）客户端启动

启动成功：命令行界面提示信息

启动失败：命令行界面提示信息

（2）AES文件加解密

加密成功：得到加密后的文件，无法正常查看内容

加密失败：未得到加密后的文件

解密成功：得到解密后的文件，其内容与原文件相同

解密失败：无法得到解密后的文件

（3）用户命令测试

对“send”、“show”、“flush”、“exit”命令进行测试。

测试成功：出现相应结果和日志

测试失败：报错信息提示

5.2 服务端

（1）服务端启动

启动成功：命令行界面提示信息

启动失败：命令行界面提示信息

（2）用户命令处理

对用户发出的“send”、“show”、“flush”、“exit”命令进行处理。

测试成功：出现相应结果和日志

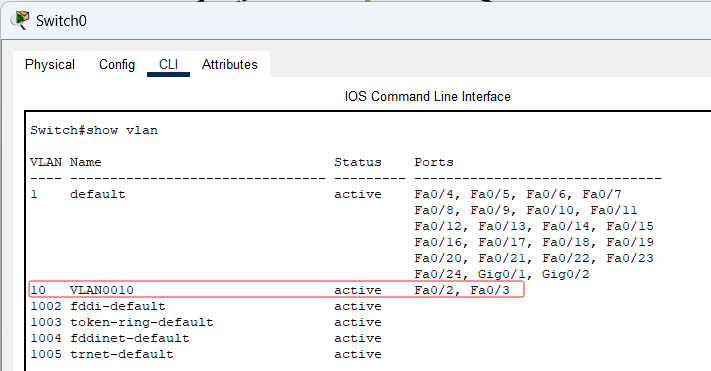
测试失败：报错信息提示

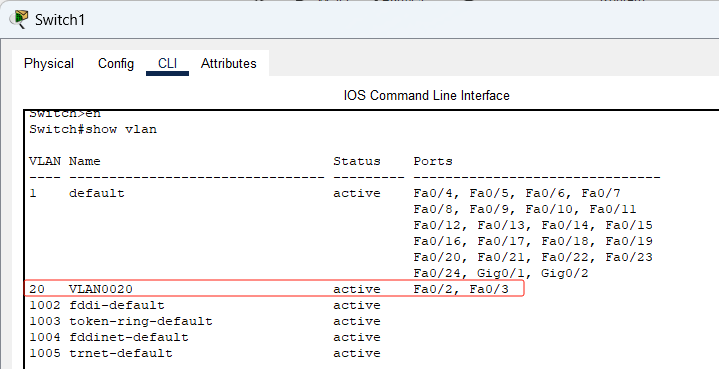
**实验结果与分析：**

**题目一：**

1. VLAN划分及配置验证

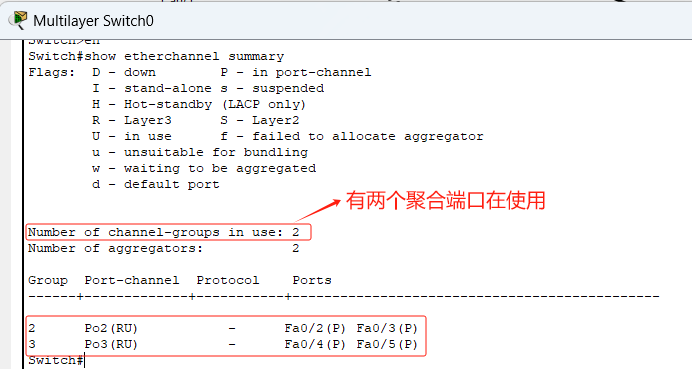
交换机连接两台终端设备的端口被划分到对应的vlan中。

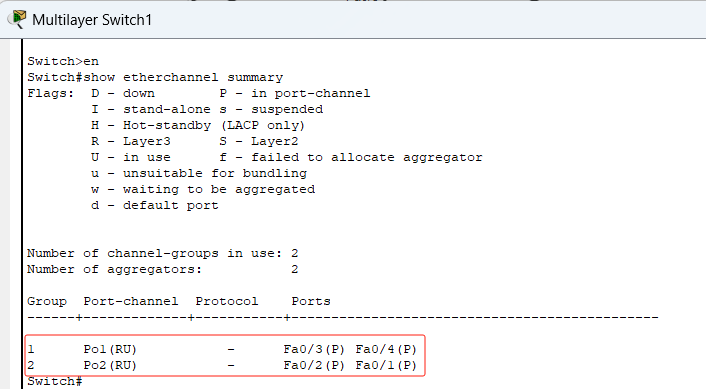


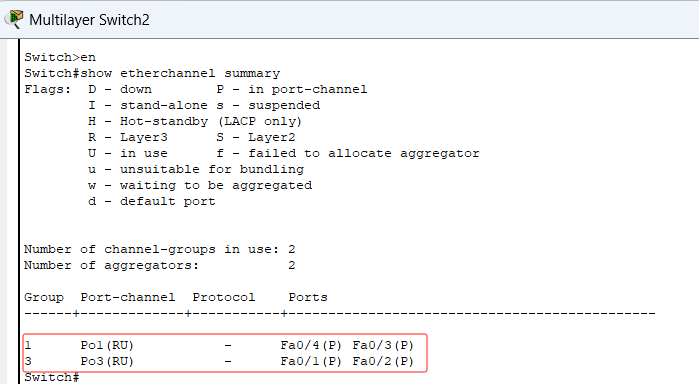


1. 骨干网区链路聚合功能验证

每台交换机下都有两个聚合端口，相连的两台交换机端口被聚合到了对应的聚合端口下。







3.OSPF动态路由协议验证

3.1 查看路由表

以Multilayer Switch0为例说明：该交换机路由表上的S 192.168.1.0/24 [1/0] via 192.168.2.1是配置的去往防火墙的静态路由。

192.168.10.0、192.168.20.0、192.168.40.0以及192.168.100.0都是它的非直连网段，是通过OSPF路由协议学习到的。

0\*E2 0.0.0.0/0 [110/1] via 192.168.200.2,00:21:38,FastEthernet0/1：表示通过OSPF协议学习到的，并且它是一个外部路由（E2表示外部路由类型）。由OSPF自治系统边界路由器（ASBR）从静态路由引入，并作为外部路由注入到OSPF中的。

具体学习过程：

（1）ASBR（Router0）从静态路由引入外部路由：

Router0配置有与其他路由协议的连接，即 ip route 0.0.0.0 0.0.0.0 202.204.100.2。当R0从这条静态协议学习到路由信息时，它会将这些路由作为外部路由保存在自己的路由表中。

（2）ASBR将外部路由注入OSPF：

R0被配置为将特定的外部路由注入到OSPF中。这通常通过特定的配置命令实现，例如使用redistribute命令来重新分发从其他协议学到的路由。

当R0执行注入操作时，它会生成一个特殊的LSA（即下面OSPF数据库中的Type-5 AS External Link states，第五类LSA），描述这些外部路由的信息，包括目标网络、下一跳地址以及路由的度量值等。

（3）OSPF路由器学习外部路由：

OSPF路由器通过正常的链路状态交换过程，包括Hello协议、数据库同步和洪泛等，接收来自R0的Type-5 LSA。

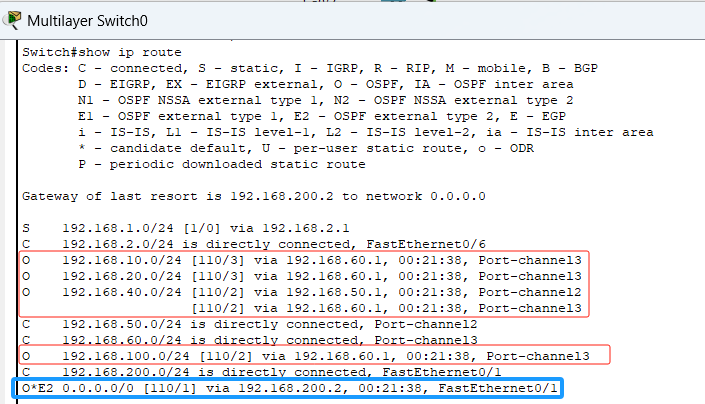
当一个OSPF路由器收到这些LSA时，它会解析其中的路由信息，并根据OSPF的算法计算到达这些外部目标网络的最佳路径。

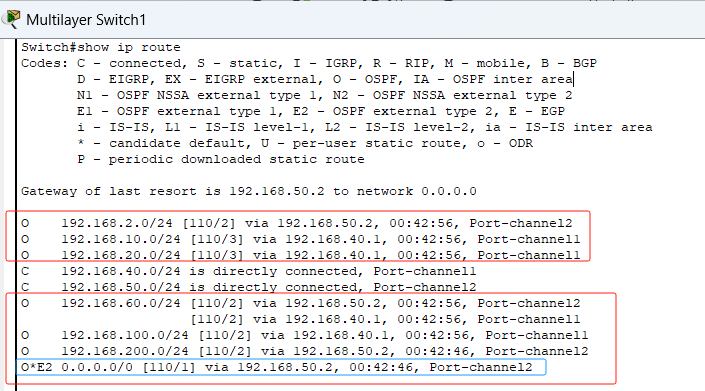
如果该外部路由是一个默认路由（如0.0.0.0/0），那么它将被添加到路由器的路由表中，作为到达没有更具体路由匹配的目标网络的默认出口。

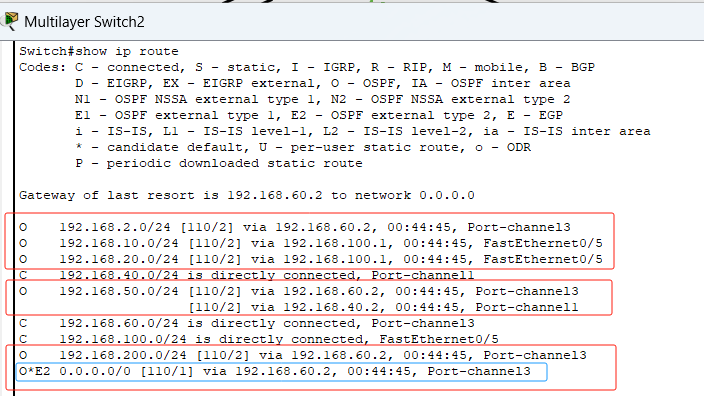
（4）路由表更新和转发决策：

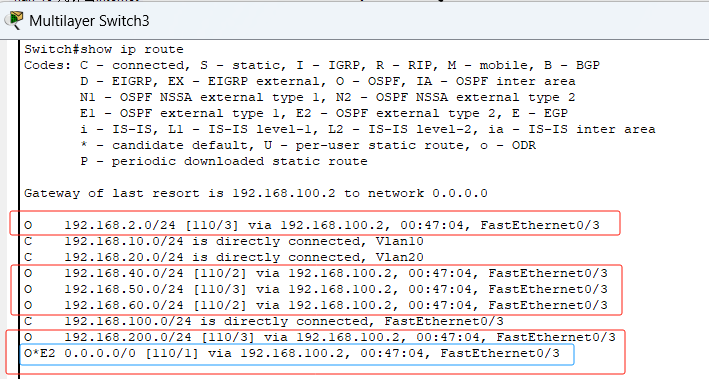
一旦外部路由被学习到并添加到路由表中，路由器就可以根据路由表的信息做出转发决策。

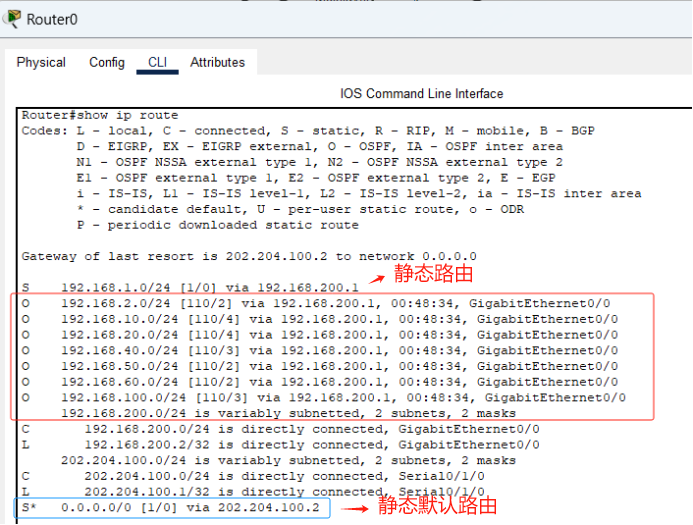
对于匹配默认路由的数据包，路由器将使用指定的下一跳地址192.168.200.2和出口接口FastEthernet0/1来转发这些数据包。







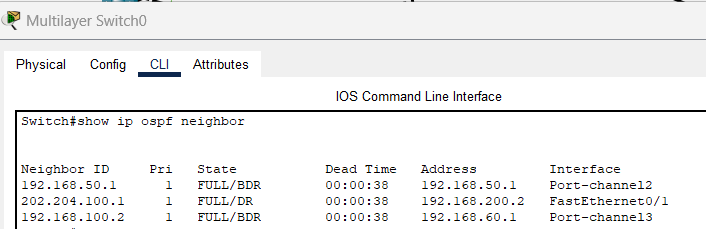


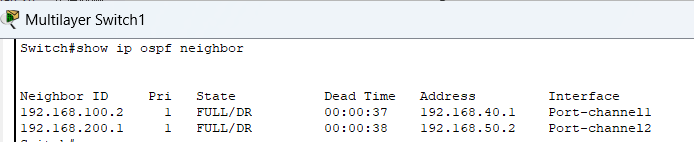


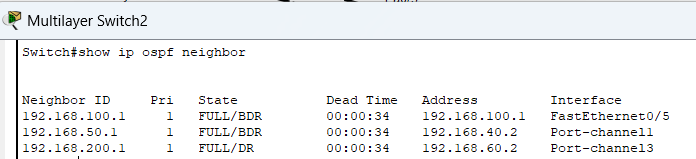
3.2 查看OSPF信息

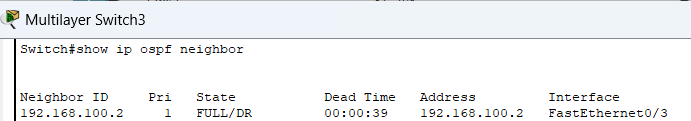
3.2.1 查看OSPF邻居

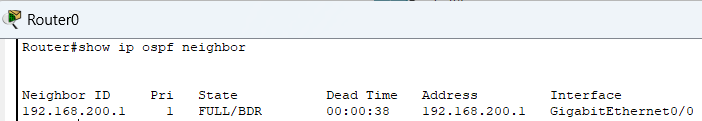
以Multilayer Switch0为例说明：在区域area0里面，该交换机有三个邻居，即Multilayer Switch1、Multilayer Switch2、Router0。邻居的状态为full，需要选举DR（指定路由器）和BDR（备份指定路由器）











3.2.2 查看OSPF数据库

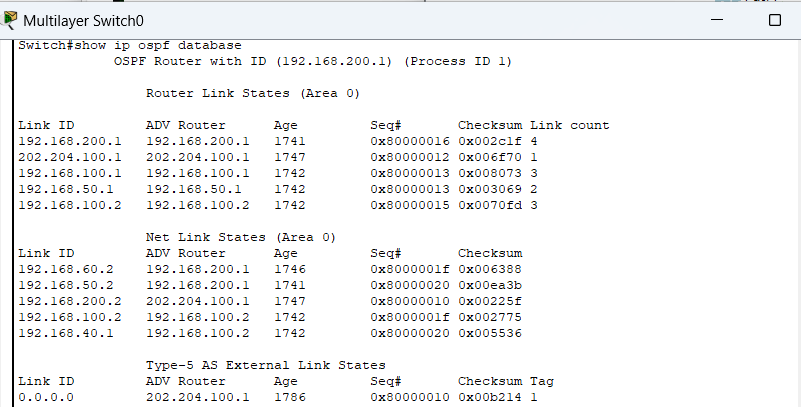
Link ID：链路上邻居接口的IP地址

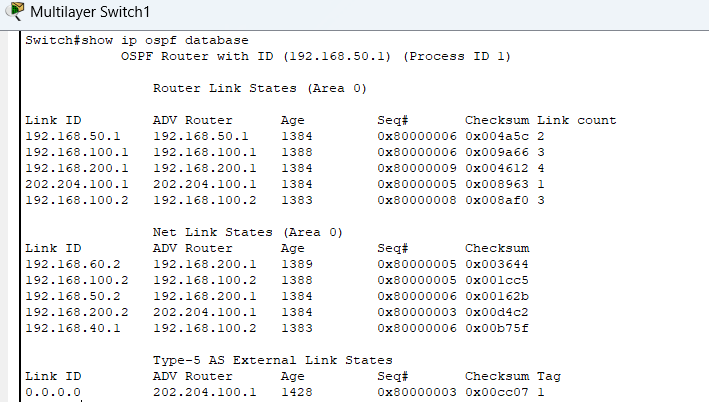
ADV Router：产生该LSA的路由器的Router-ID

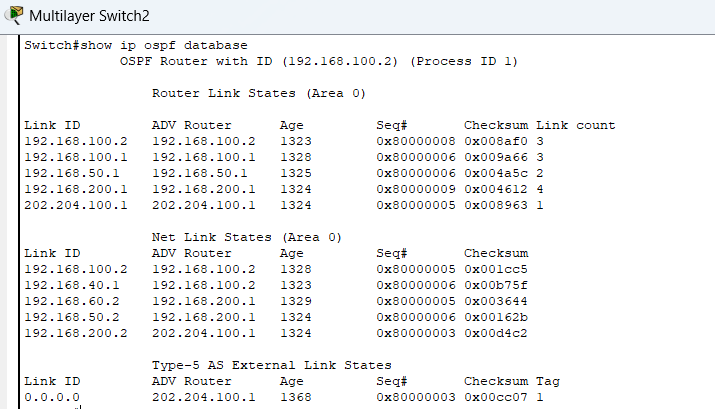
Router LSA（type1）：由该路由器产生，描述路由器的链路状态和开销，在其始发的区域内传播。

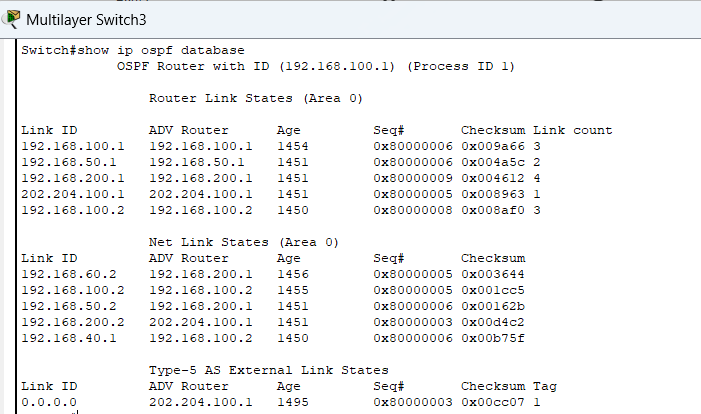
Network LSA(type2)：由 DR 产生，描述本网段所有路由器的链路状态，在其始发的区域内传播。

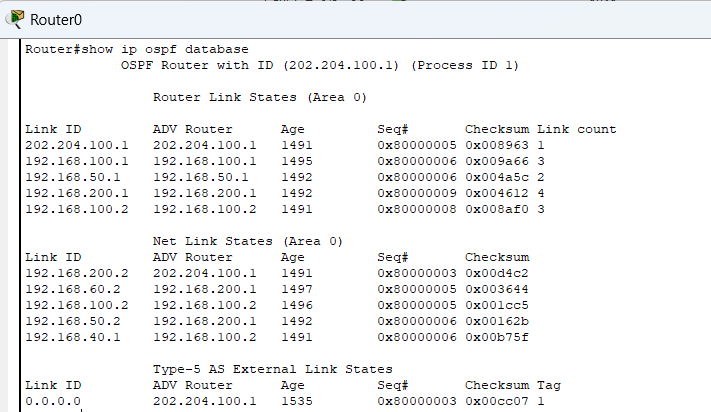
Type-5 AS External Link states ：通告到达OSPF自主系统外部的目的地或者OSPF AS外部的缺省路由的LSA。





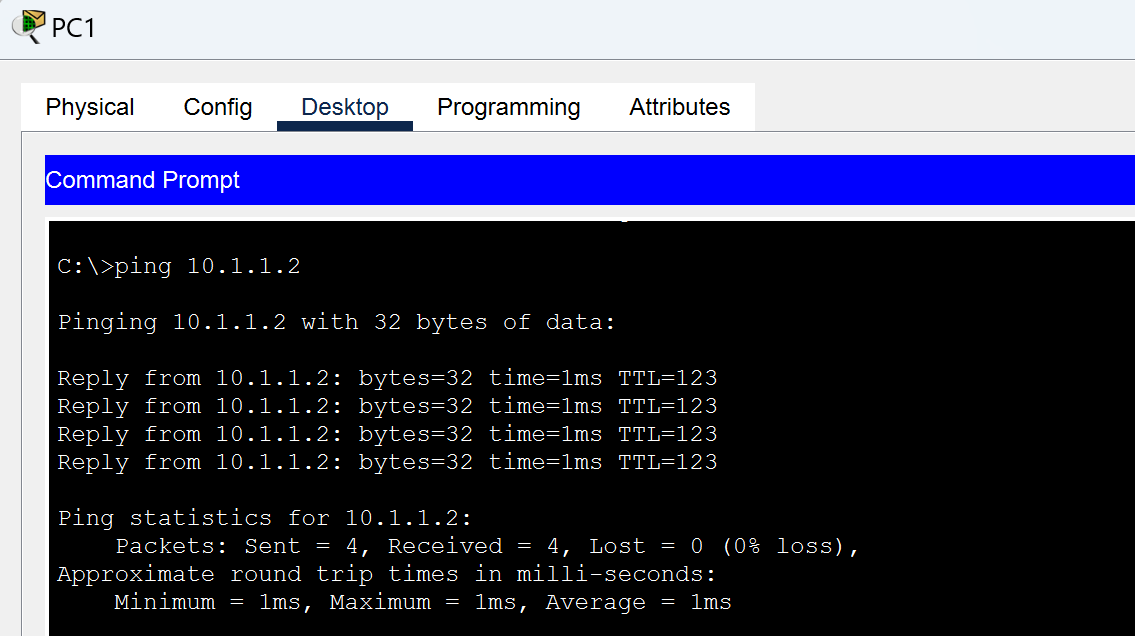


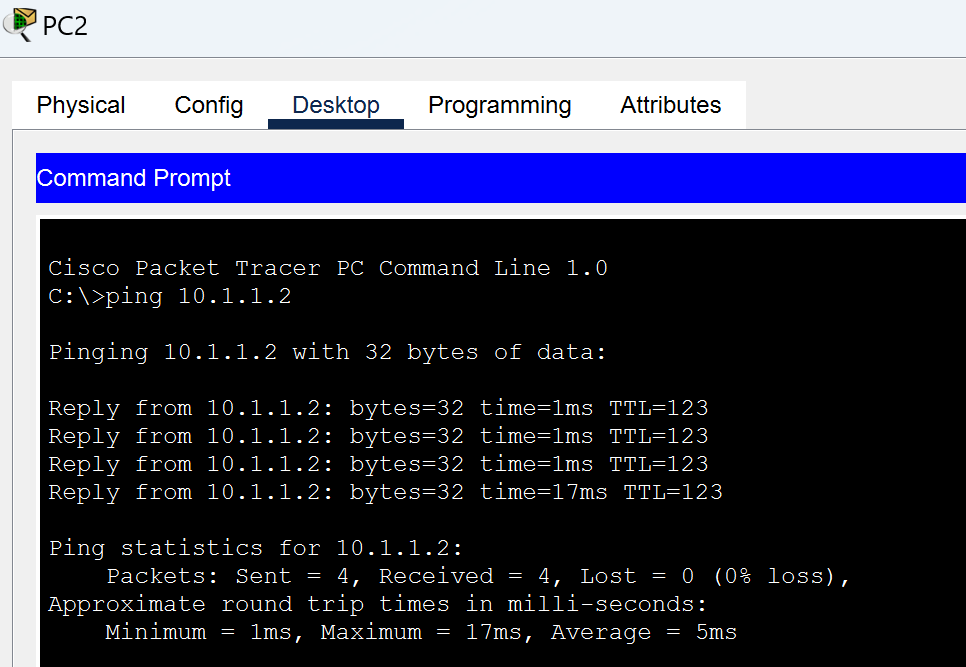




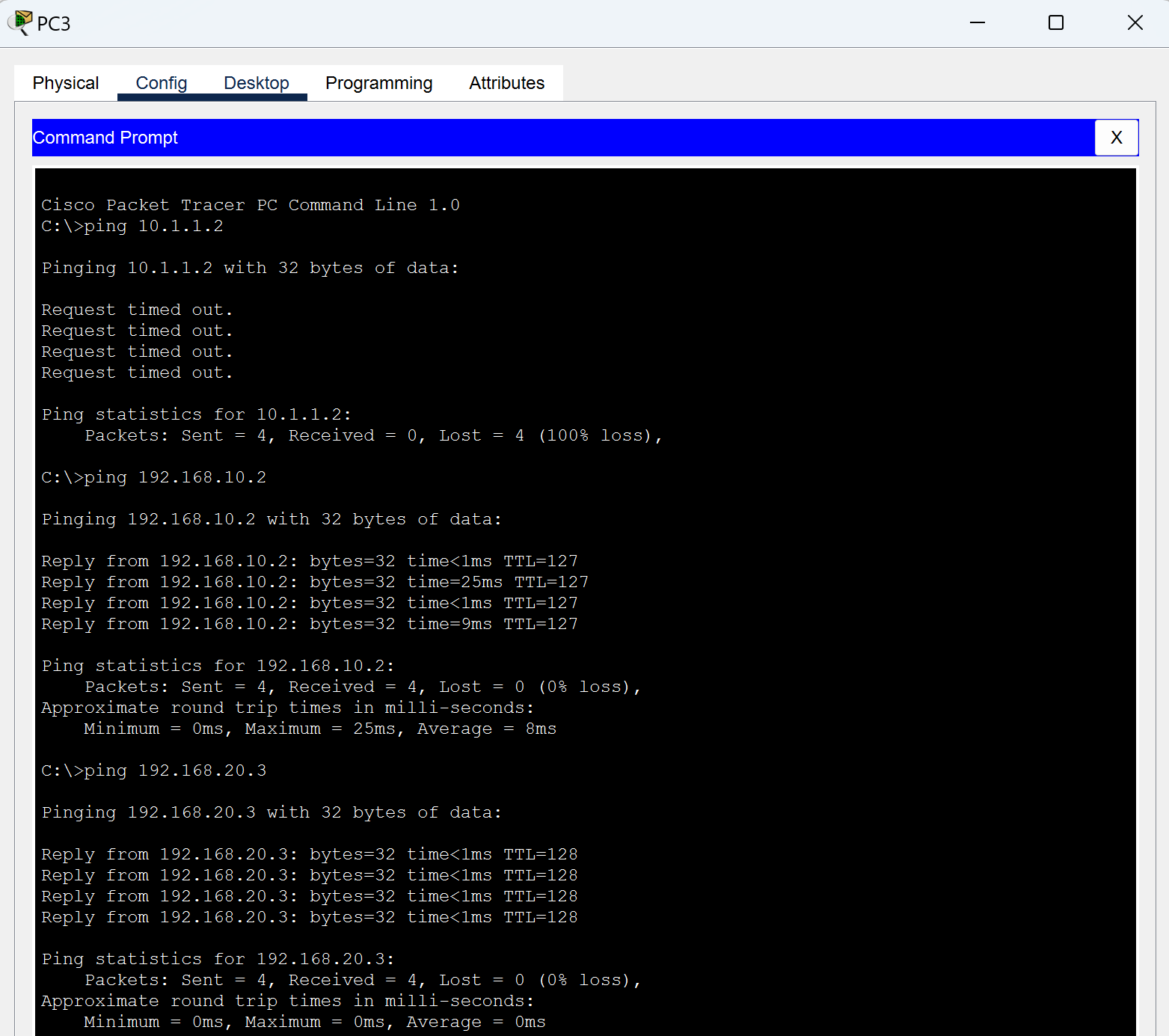
4.NAT配置与访问互联网验证

4.1 使用VLAN10下的PC1和PC2成功访问外网主机PC0

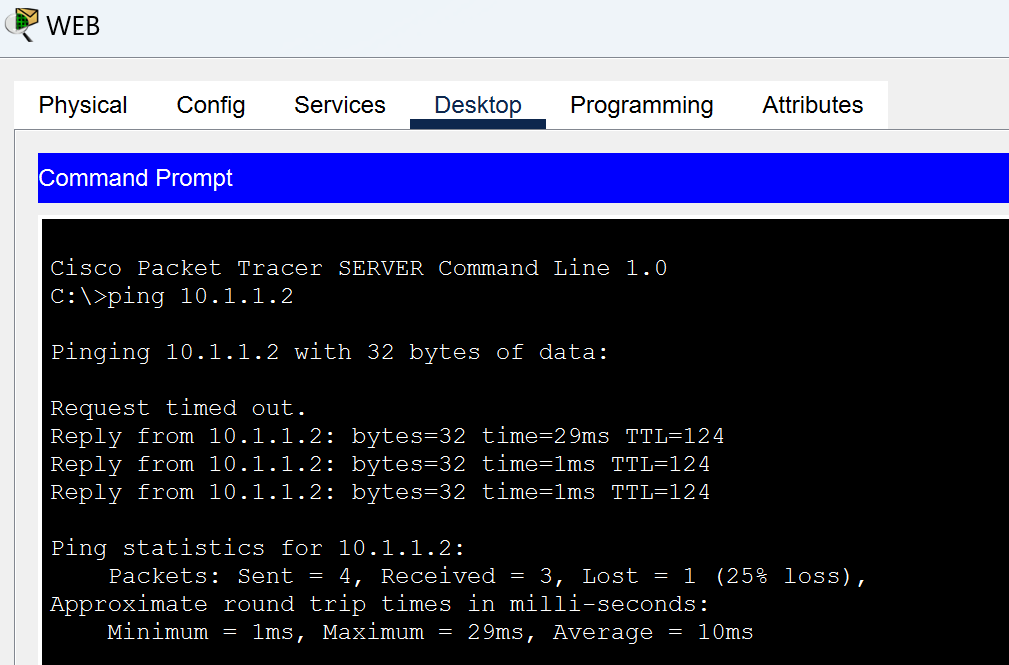


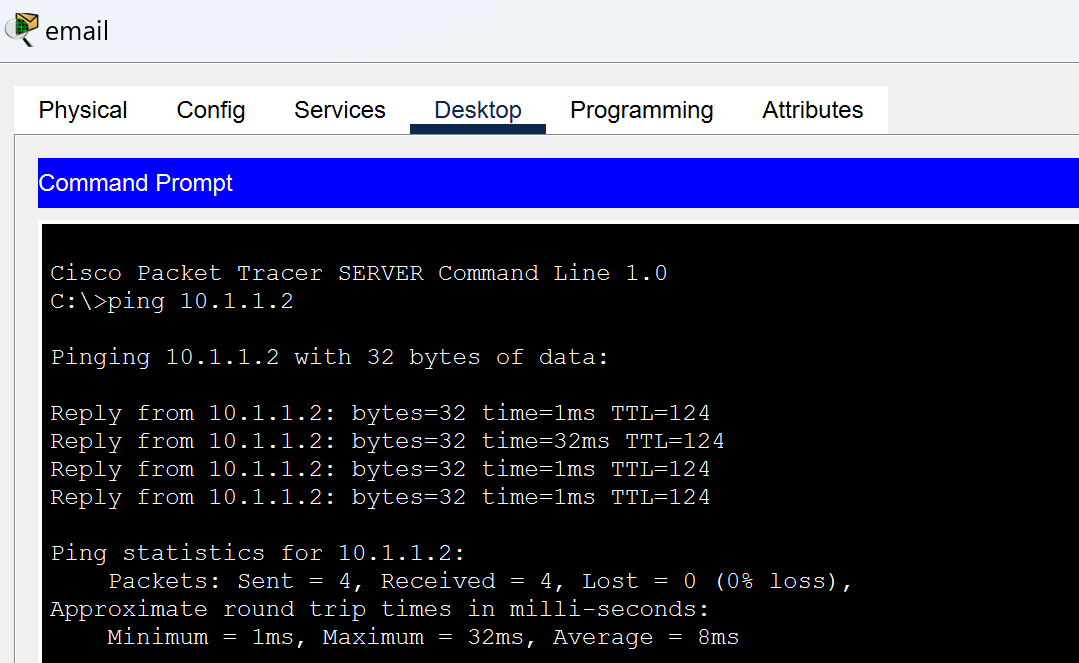


4.2 使用VLAN20下的PC3不能访问外网主机PC0，但可以访问内网主机PC1

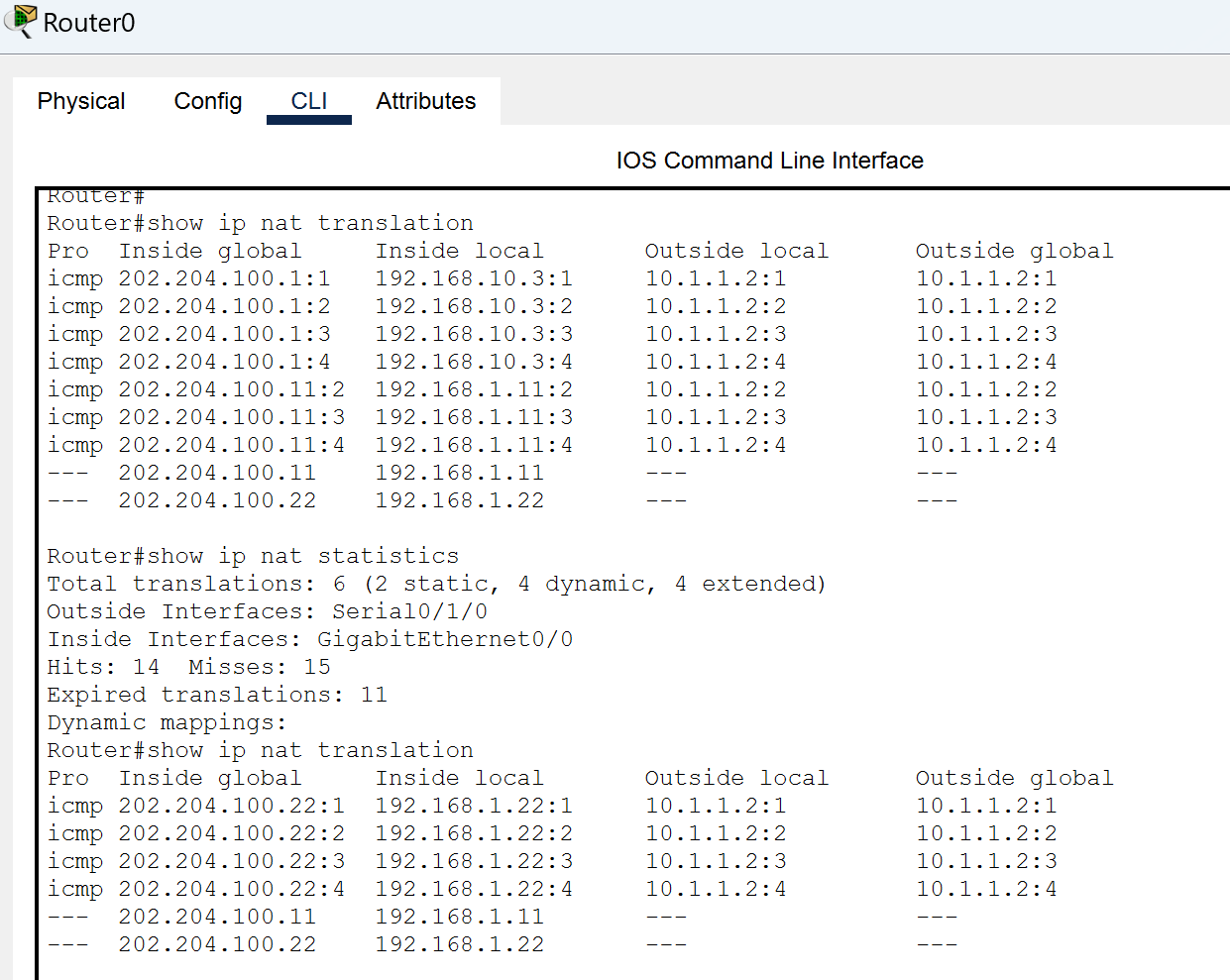


4.3 使用web服务器和邮件服务器成功访问外网主机PC0





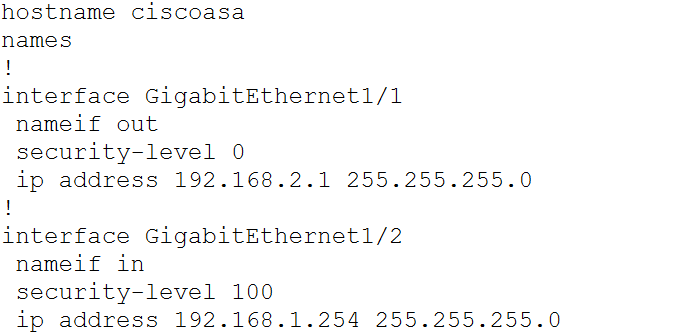
4.4 使用show ip nat translation和show ip nat statistics查看NAT转换统计情况



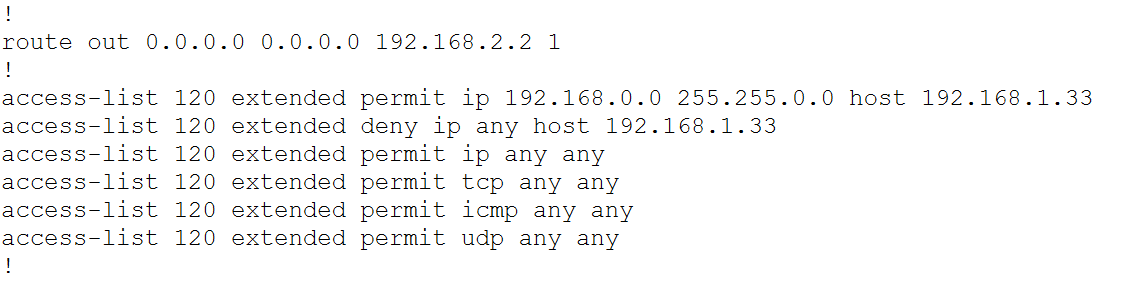
5.防火墙与服务器功能验证

5.1通过show run检验防火墙当前运行的所有配置

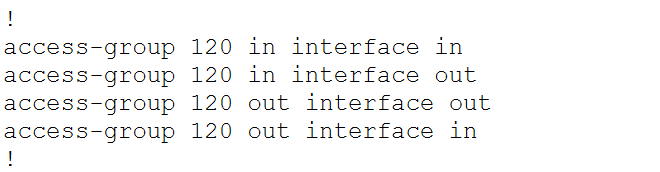
5.1.1端口安全级别与ip地址检查



5.1.2 静态默认路由与访问控制表检查

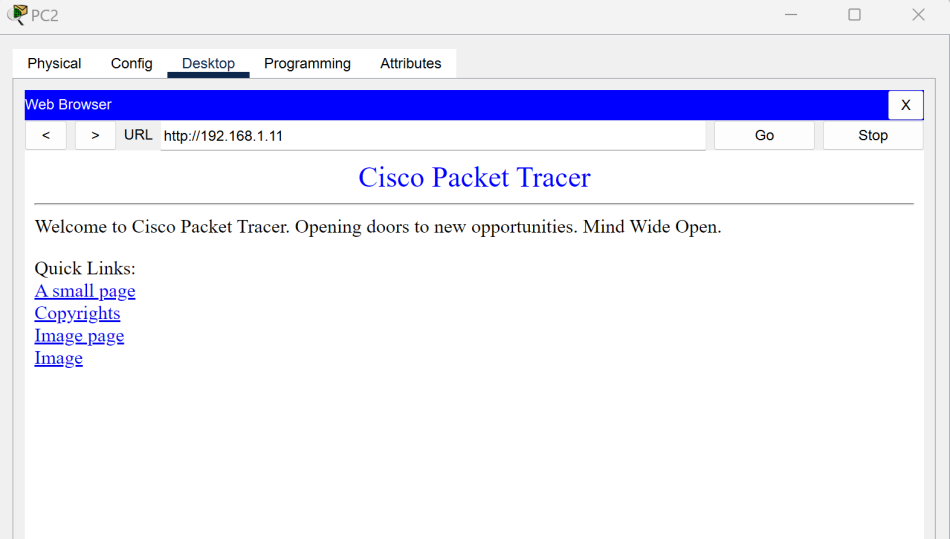


5.1.3 ACL应用于接口的入站与出站方向检查

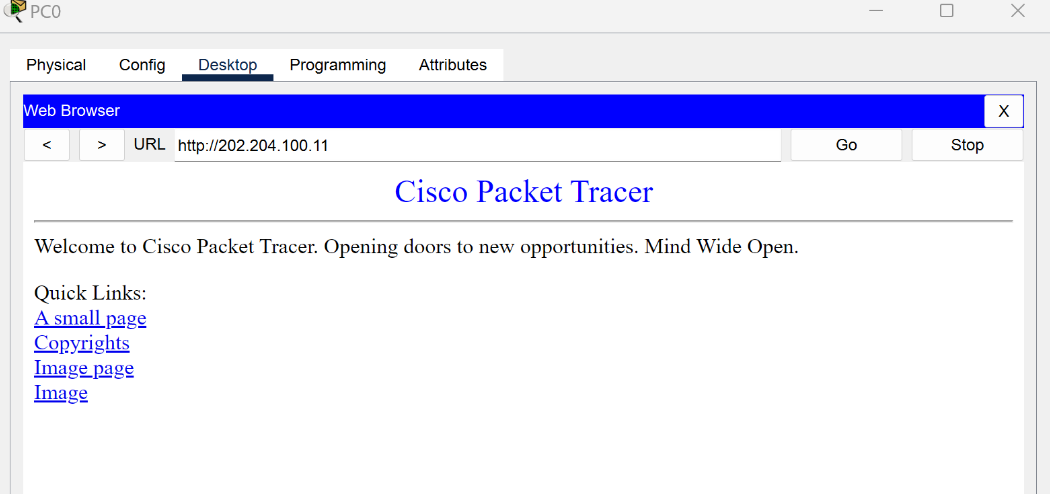


5.2 Web服务器被内网和外网访问

5.2.1 使用内网主机PC2访问Web服务器IP地址

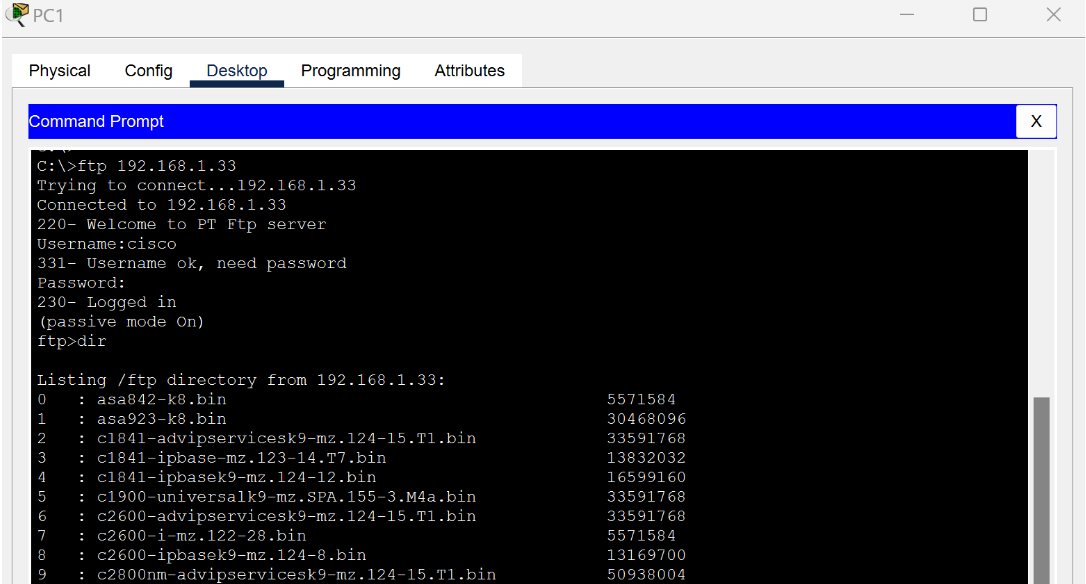


5.2.2 使用外网主机PC0访问Web服务器的静态NAT映射地址

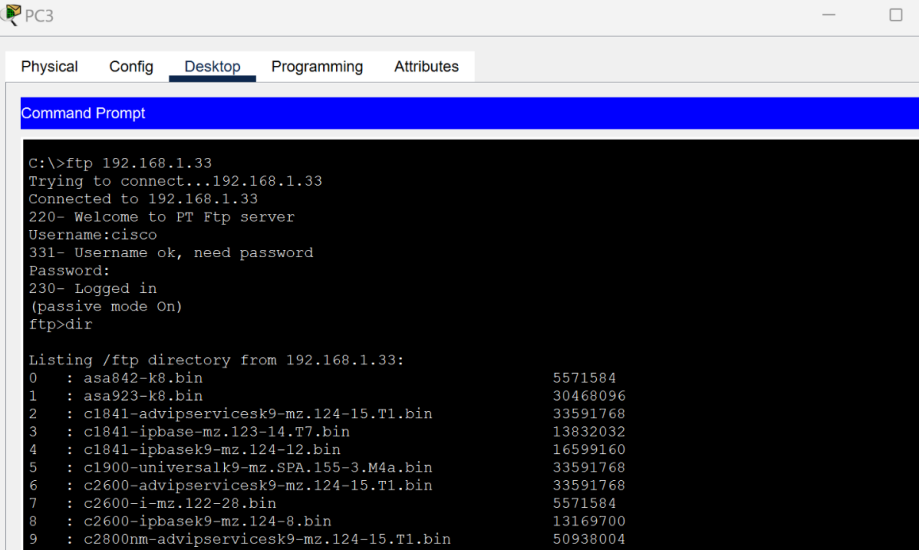


5.3 使用内网主机访问FTP服务器

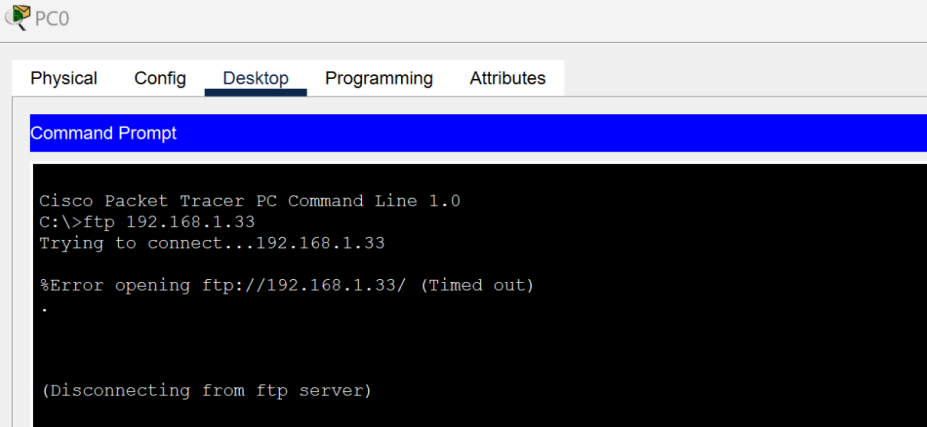
5.2.1 使用VLAN10的主机PC 1访问服务器



5.2.2 使用VLAN20的主机PC3访问FTP服务器

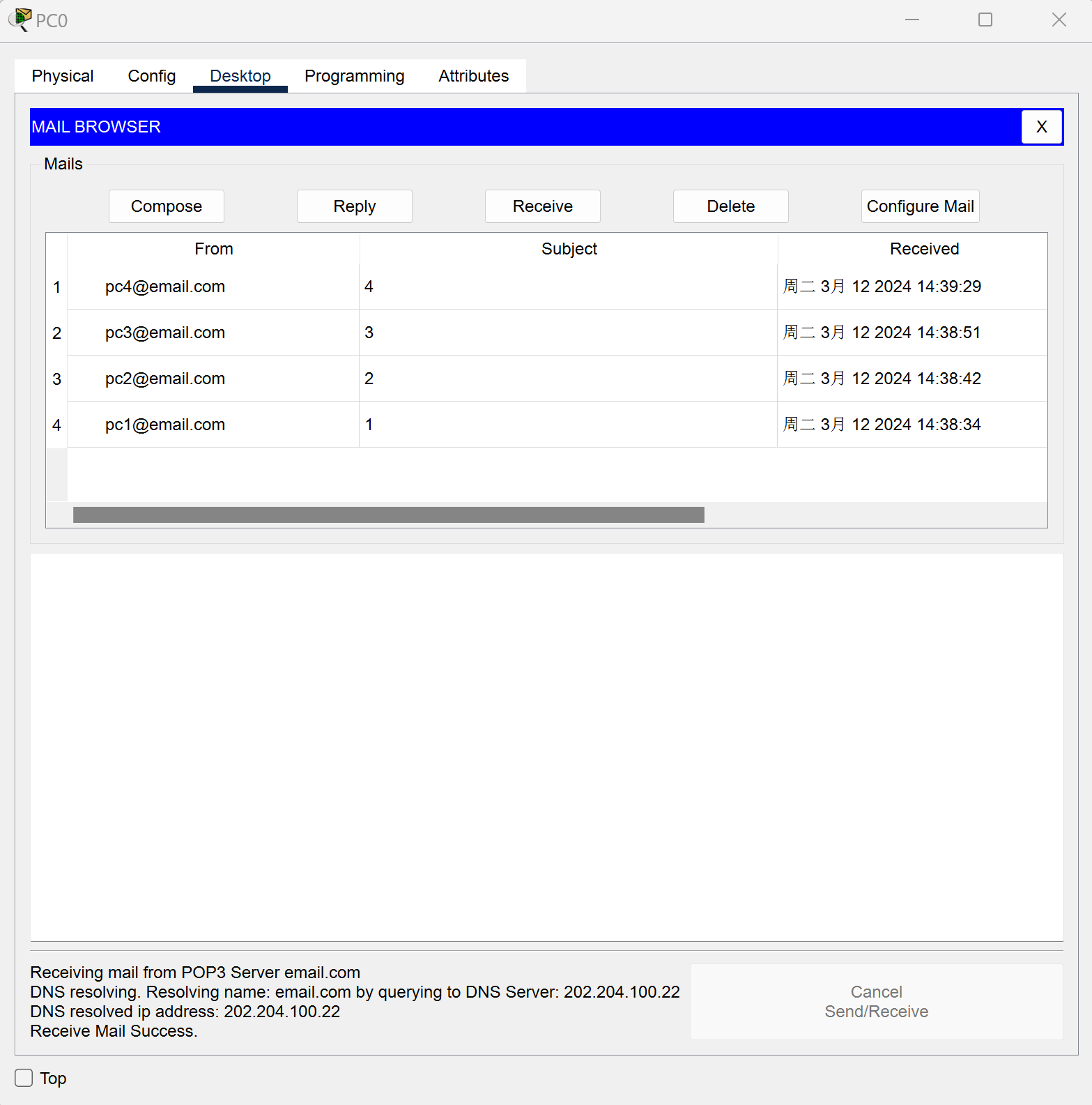


5.2.3 使用外网主机PC0无法访问FTP服务器

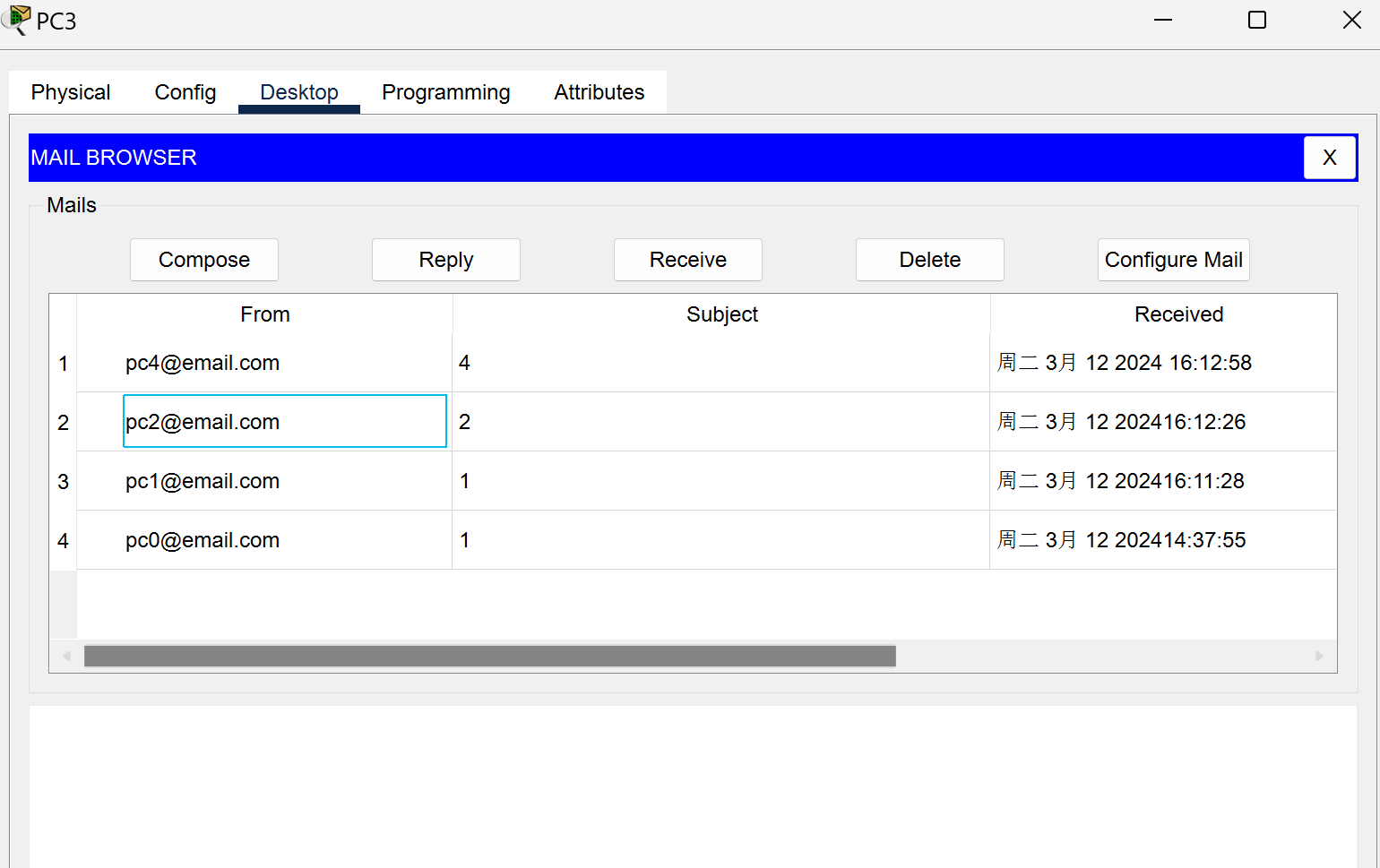


5.3 内外网使用邮件服务器互相发送邮件

5.3.1 外网主机PC0接收内网各主机发送的邮件



5.3.2 内网主机PC3接收内外网主机发送的邮件



6.总结分析

本项目以某大学网络为例，重点实现了图书馆子网与主干网的接入，并遵循了典型的三层网络结构。核心层由三台高性能的三层交换机组成环形拓扑，通过两条链路聚合提升了校园网络主干的带宽。

在项目初期，我们根据参考网络拓扑完成了详细的网络拓扑结构设计，包括设备的位置、连接方式以及各层级的划分。同时，我们进行了细致的IP地址规划，确保了每个设备、每个子网都有唯一且合理的IP地址分配，避免了地址冲突和管理混乱。

针对图书馆子网的两类用户需求，我们合理规划了VLAN划分。通过VLAN的配置，实现了对用户的逻辑隔离，确保了一类用户可以通过NAT方式访问Internet，而另一类用户则被限制在仅可访问校内网络。在NAT配置方面，我们采用了动态NAT和静态NAT相结合的方式，既满足了普通用户的上网需求，又确保了服务器等关键设备的固定公网IP地址映射。

为了提升网络的灵活性和可扩展性，我们在核心层之间运行了OSPF动态路由协议。这一配置使得网络能够自动学习并更新路由信息，提高了网络的容错能力和效率。同时，通过两条链路聚合的配置，我们成功增加了主干网的带宽，提升了网络的整体性能。

在网络安全方面，我们重点配置了防火墙的访问控制策略。通过设置访问控制表（ACL），我们确保了只有Web服务器和邮件服务器可以被网络上的其他计算机访问，而FTP服务器则仅允许内部网络访问。这一配置有效地保护了内部网络的安全，防止了未经授权的访问和潜在的网络攻击。

综上所述，本项目成功实现了图书馆子网与主干网的接入，并通过合理的网络设计和配置满足了各类用户的需求。同时，我们也注重了网络的安全性和性能优化，为校园网络的高效、稳定运行提供了有力保障。在未来的工作中，我们将继续关注网络技术的发展和用户需求的变化，不断完善和优化校园网络架构。

**题目二：**

在这个环节，将对子网文件传送这个项目进行更为详细的测试。具体针对“模块测试与总体测试设计”中的测试设计进行验证。

1 实验数据及预期结果

1.1 客户端

1.1.1 客户端启动

表？ 客户端启动功能测试表

|  |  |
| --- | --- |
| 实验数据 | 预期结果 |
| 启动客户端文件夹下的“Client.exe”程序，查看命令行界面中的提示信息和工作目录下的日志内容。 | 当启动客户端文件夹下的“Client.exe”程序后，命令行界面中出现“连接成功”的提示信息，工作目录下生成日志内容，记录着相应的日期的操作。 |

1.1.2 AES文件加解密

表？ AES文件加解密功能测试表

|  |  |
| --- | --- |
| 实验数据 | 预期结果 |
| 将Client的AES模块单独分离，进行单独测试。在工作目录下准备命名为“test.txt”的记事本文件和“test.jpg”的图片文件，利用AES模块进行加解密操作。 | 当利用分离后的AES模块进行加解密操作后，在工作目录下生成相应的加密后的文件和解密后的文件。其中，加密后的文件无法正常打开或打开后与原内容不一致；解密后的文件可以正常打开，其内容与原文件一致。 |

1.1.3 用户命令测试

表？ 用户命令功能测试表

|  |  |  |
| --- | --- | --- |
| 测试项目 | 实验数据 | 预期结果 |
| send | 当客户端与服务端建立连接后，在客户端的命令行界面中键入“send”命令，然后根据提示信息输入相应的文件名。 | 当客户端键入“send”命令后，服务端接收到该操作，然后将接收到用户所发送的加密后的文件，该文件无法正常打开或打开后与原文件不一致。客户端与服务端均有相应日志内容的记录。 |
| show | 当客户端与服务端建立连接后，在客户端的命令行界面中键入“show”命令。 | 当客户端键入“show”命令后，服务端接收到该操作。客户端的用户命令行界面中出现响应结果，客户端与服务端均有相应日志内容的记录。 |
| flush | 当客户端与服务端建立连接后，在客户端的命令行界面中键入“flush”命令。 | 当客户端键入“flush”命令后，服务端接收到该操作。客户端的用户命令行界面中出现响应结果，服务端接收到客户端的文件，这些文件无法正常打开或打开后与原文件不一致。客户端与服务端均有相应日志内容的记录。 |
| exit | 当客户端与服务端建立连接后，在客户端的命令行界面中键入“exit”命令。 | 当客户端键入“exit”命令后，服务端接收到该操作。客户端自动关闭，表示该用户退出，服务端更新相应的数据结构。客户端与服务端均有相应日志内容的记录。 |

1.2 服务端

1.2.1 服务端启动

表？ 服务端启动功能测试表

|  |  |
| --- | --- |
| 实验数据 | 预期结果 |
| 启动服务端文件夹下的“Server.exe”程序，查看命令行界面中的提示信息和工作目录下的日志内容。 | 当启动服务端文件夹下的“Server.exe”程序后，命令行界面中出现“连接成功”的提示信息，工作目录下生成日志内容，记录着相应的日期的操作。 |

1.2.2 用户命令处理

表？ 用户命令处理功能测试表

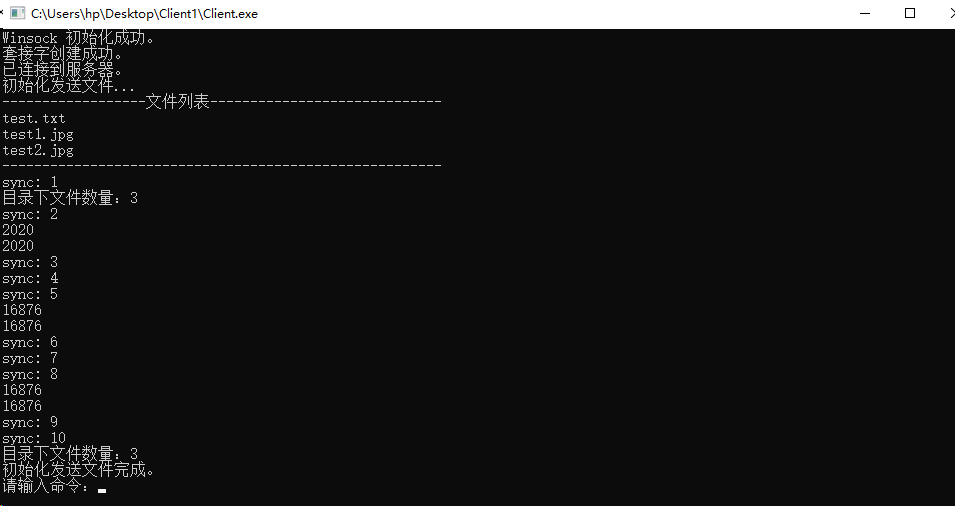
|  |  |  |
| --- | --- | --- |
| 测试项目 | 实验数据 | 预期结果 |
| send | 当客户端与服务端建立连接后，在客户端的命令行界面中键入“send”命令，然后根据提示信息输入相应的文件名。 | 当客户端键入“send”命令后，服务端接收到该操作，然后将接收到用户所发送的加密后的文件，该文件无法正常打开或打开后与原文件不一致。客户端与服务端均有相应日志内容的记录。 |
| show | 当客户端与服务端建立连接后，在客户端的命令行界面中键入“show”命令。 | 当客户端键入“show”命令后，服务端接收到该操作。客户端的用户命令行界面中出现响应结果，客户端与服务端均有相应日志内容的记录。 |
| flush | 当客户端与服务端建立连接后，在客户端的命令行界面中键入“flush”命令。 | 当客户端键入“flush”命令后，服务端接收到该操作。客户端的用户命令行界面中出现响应结果，服务端接收到客户端的文件，这些文件无法正常打开或打开后与原文件不一致。客户端与服务端均有相应日志内容的记录。 |
| exit | 当客户端与服务端建立连接后，在客户端的命令行界面中键入“exit”命令。 | 当客户端键入“exit”命令后，服务端接收到该操作。客户端自动关闭，表示该用户退出，服务端更新相应的数据结构。客户端与服务端均有相应日志内容的记录。 |

2 测试过程

2.1 客户端

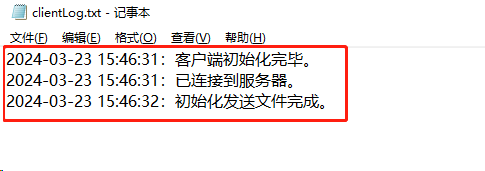
2.1.1 客户端启动

（1）启动客户端文件夹下的“Client.exe”程序，查看命令行界面中的提示信息和工作目录下的日志内容。当启动客户端文件夹下的“Client.exe”程序后，命令行界面中出现“连接成功”的提示信息。



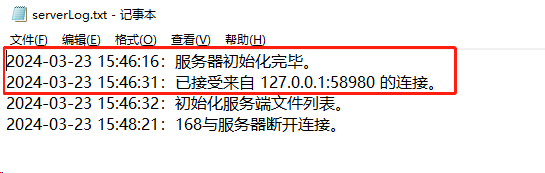
图？ 客户端成功连接服务端图示

（2）客户端生成相应的日志记录。



图？ 客户端日志内容

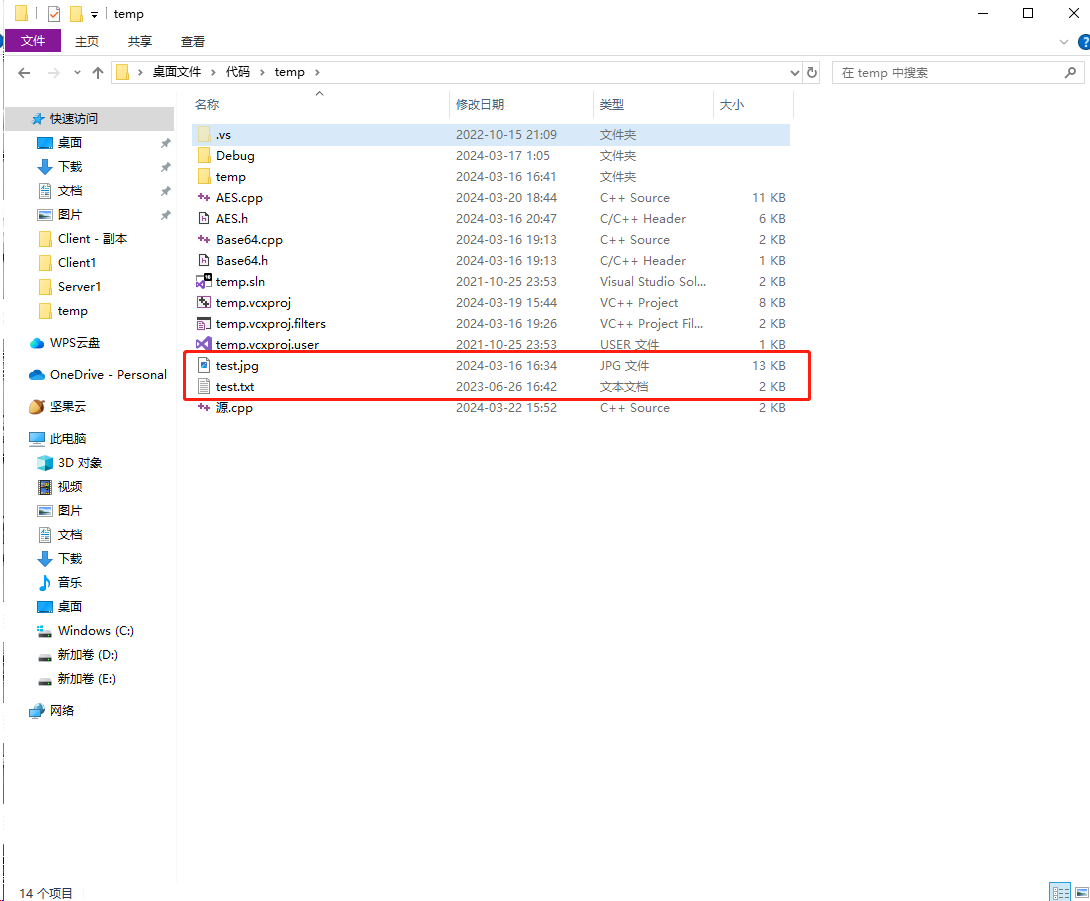
（3）服务端生成相应的日志记录。



图？ 服务端日志内容

2.1.2 AES文件加解密

（1）将Client的AES模块单独分离，进行单独测试。在工作目录下准备命名为“test.txt”的记事本文件和“test.jpg”的图片文件。

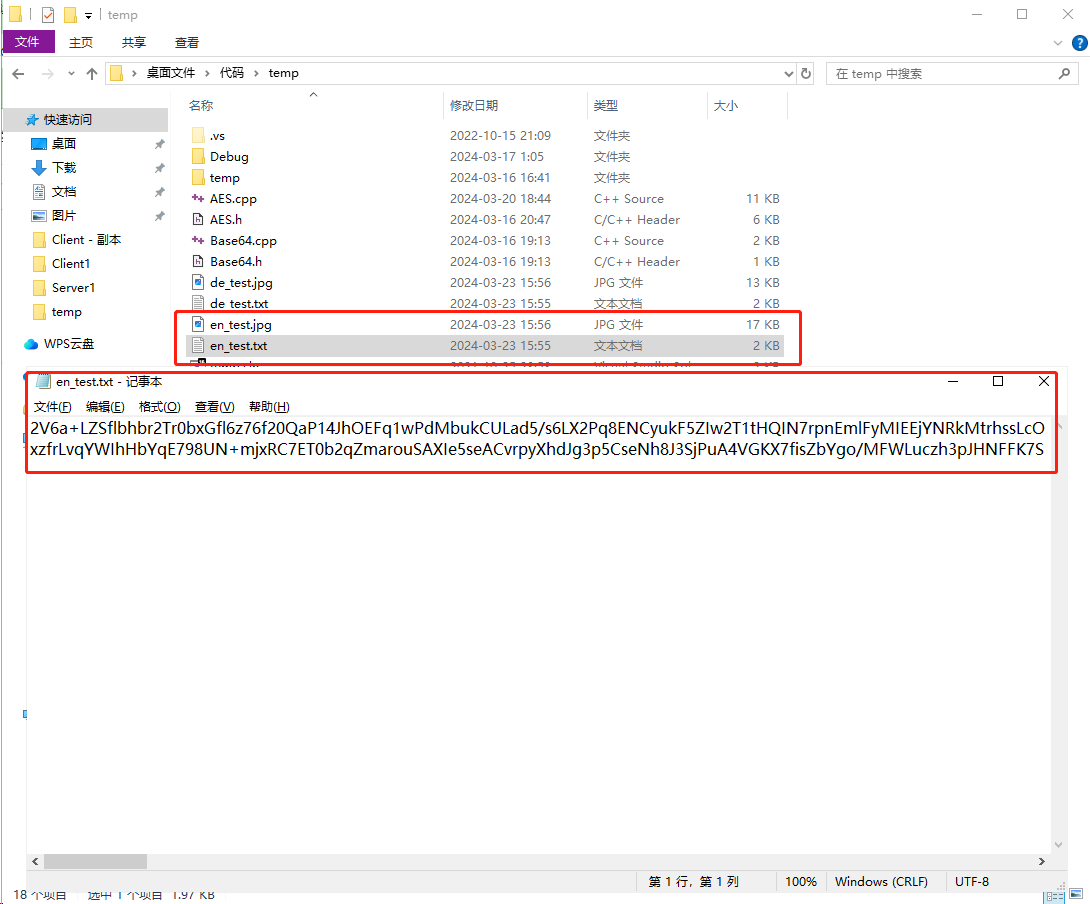


图？ 准备测试文件

（2）当利用分离后的AES模块进行加解密操作后，在工作目录下生成相应的加密后的文件和解密后的文件。其中，加密后的文件无法正常打开或打开后与原内容不一致。

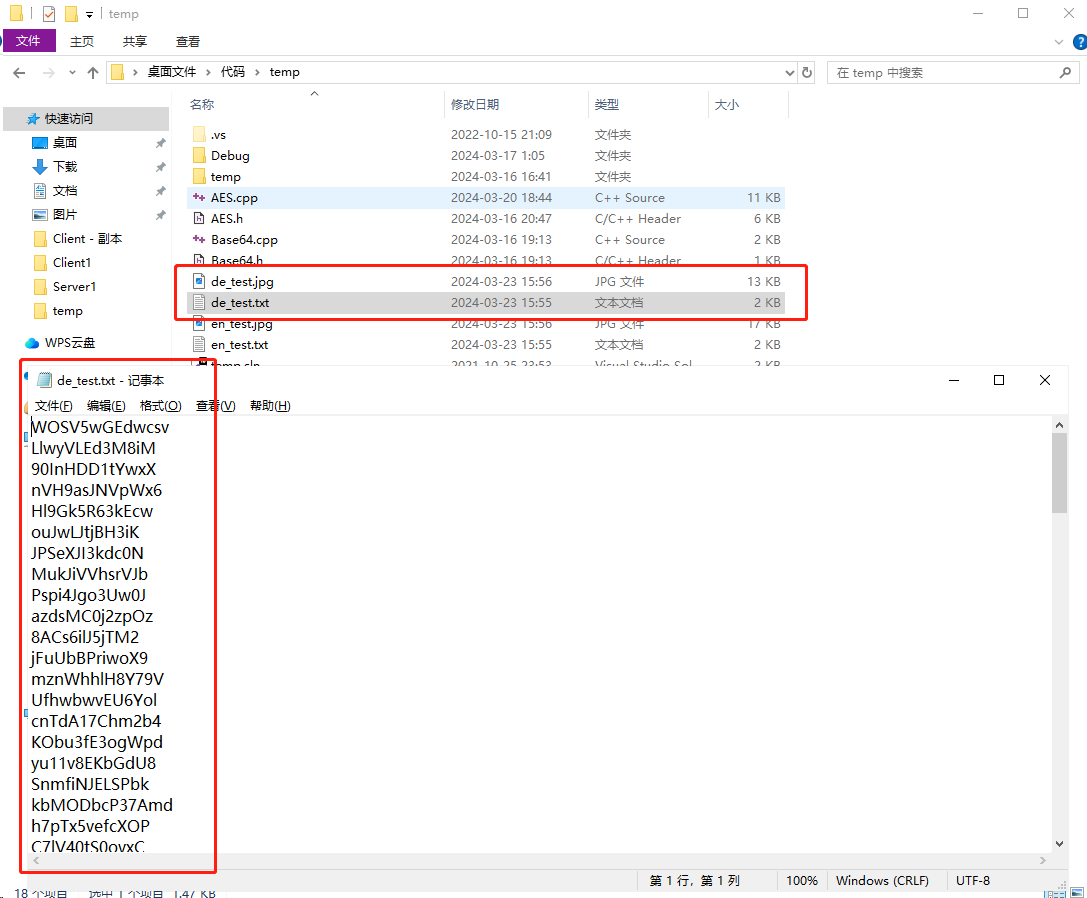
表？ 测试代码

|  |
| --- |
|  |



图？ 加密后的文件

（3）解密后的文件可以正常打开，其内容与原文件一致。

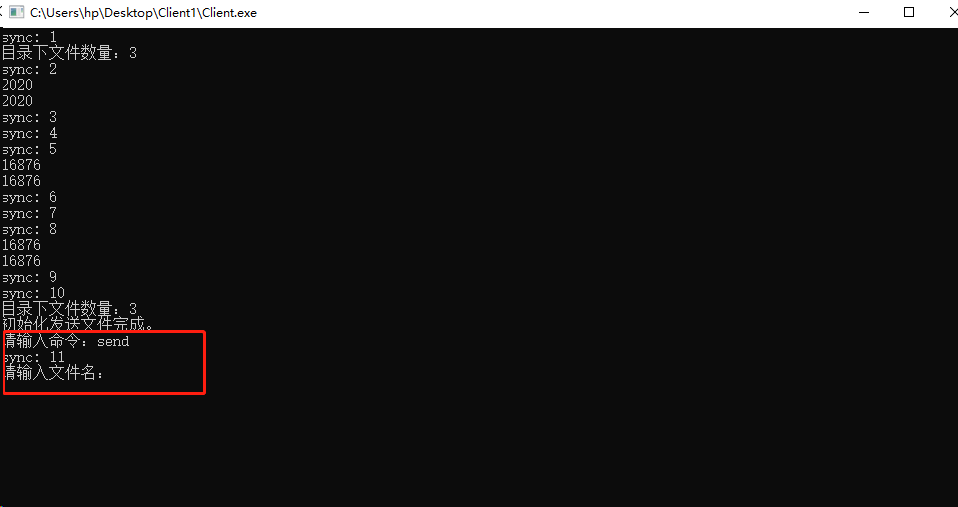


图？ 解密后的文件

2.1.3 用户命令测试

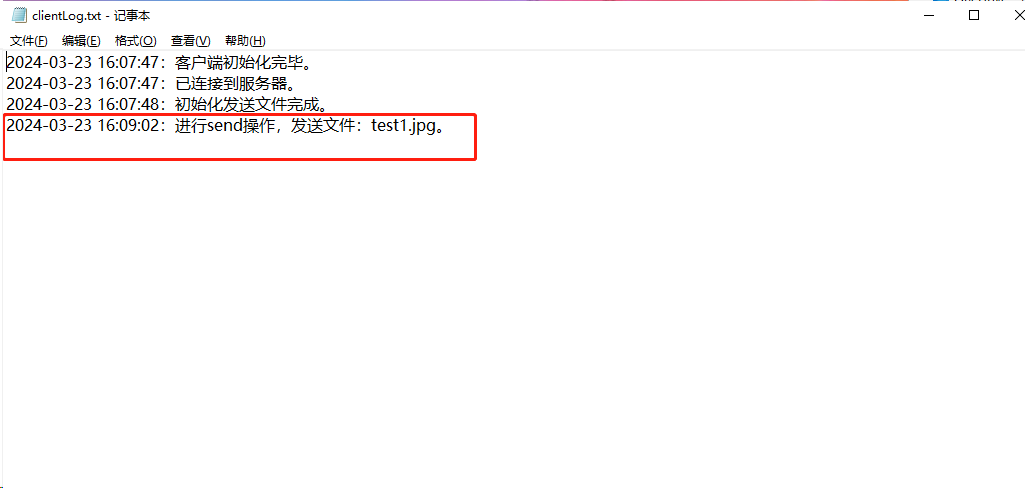
（一）send

（1）当客户端与服务端建立连接后，在客户端的命令行界面中键入“send”命令，然后根据提示信息输入相应的文件名。



图？ 客户端发送指定文件

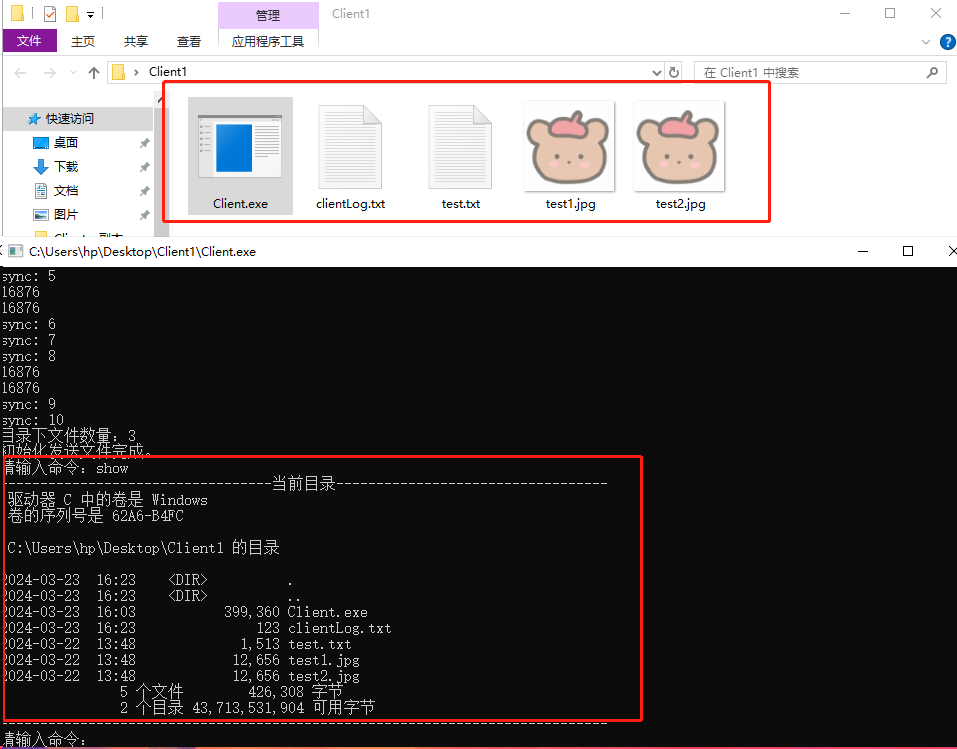
（2）服务端将接收到用户所发送的加密后的文件，该文件无法正常打开或打开后与原文件不一致。客户端有相应日志内容的记录。



图？ 客户端相应的日志内容

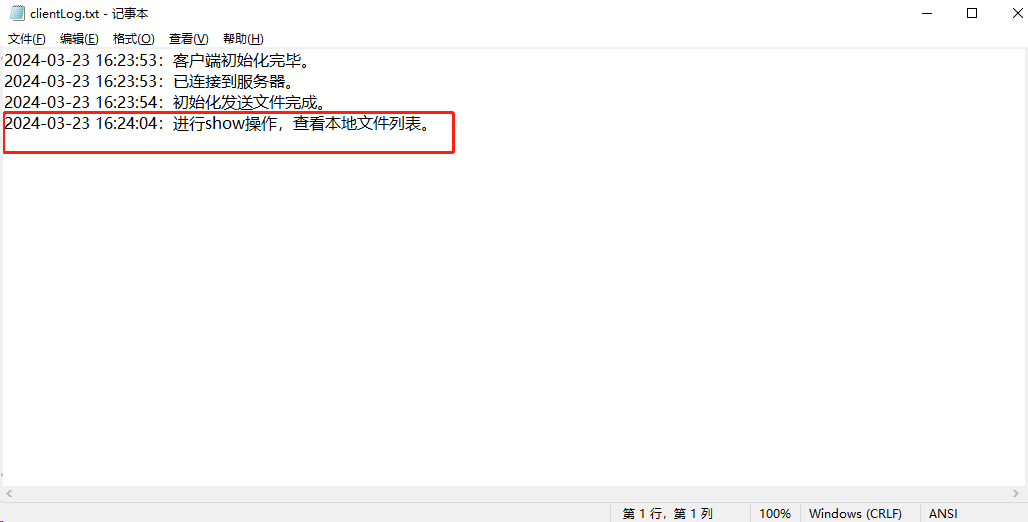
（二）show

（1）当客户端键入“show”命令后，客户端的用户命令行界面中出现响应结果。



图？ 客户端显示目录文件列表

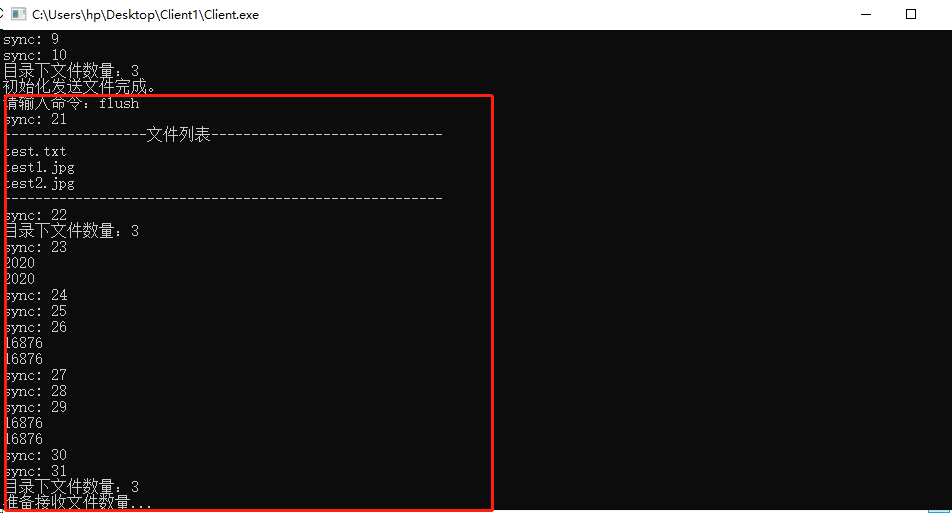
（2）客户端有相应日志内容的记录。



图？ 客户端日志内容

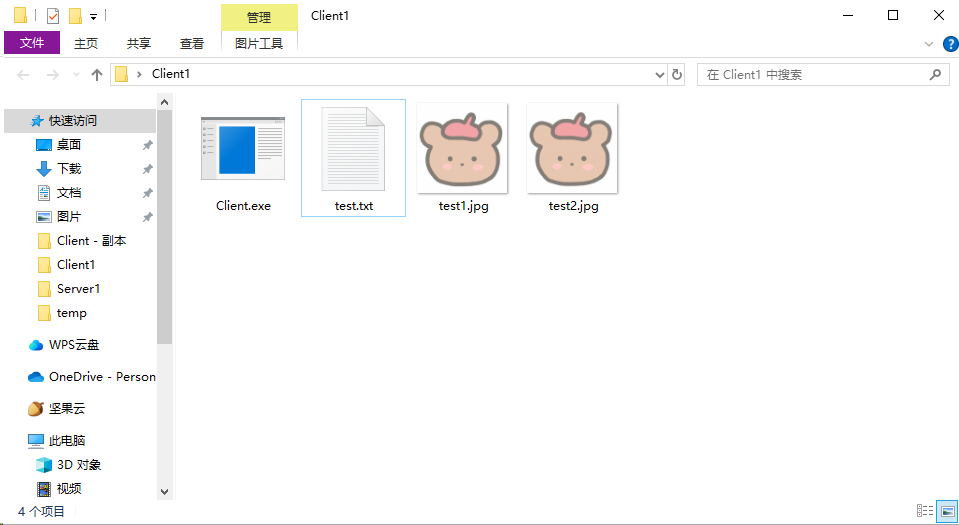
（三）flush

（1）当客户端与服务端建立连接后，在客户端的命令行界面中键入“flush”命令。客户端的用户命令行界面中出现响应结果。

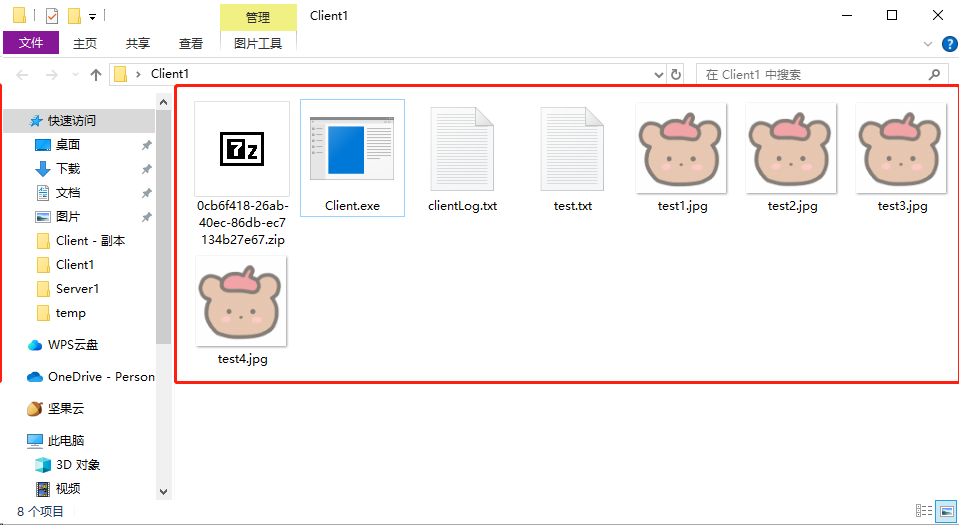


图？ 客户端命令行界面响应结果

（2）客户端与其他节点进行文件同步

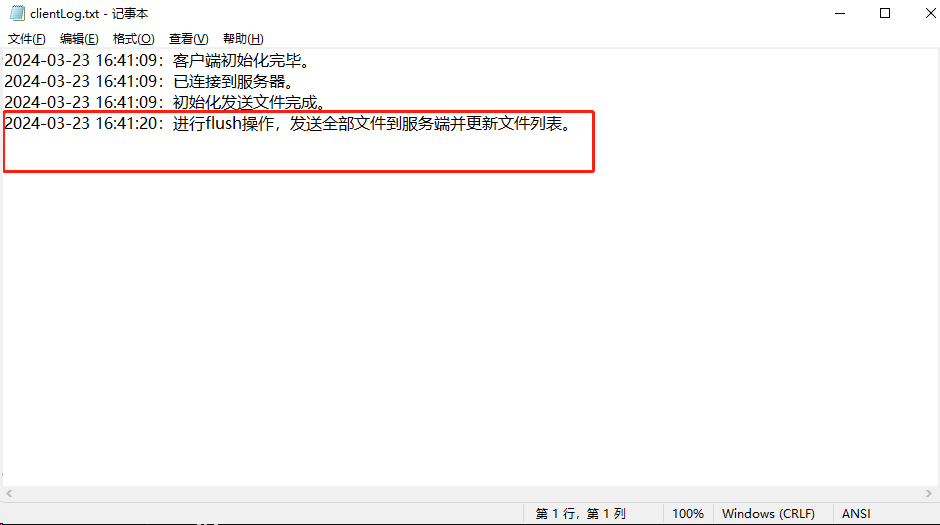


图？ 同步前的文件列表



图？ 同步后的文件列表

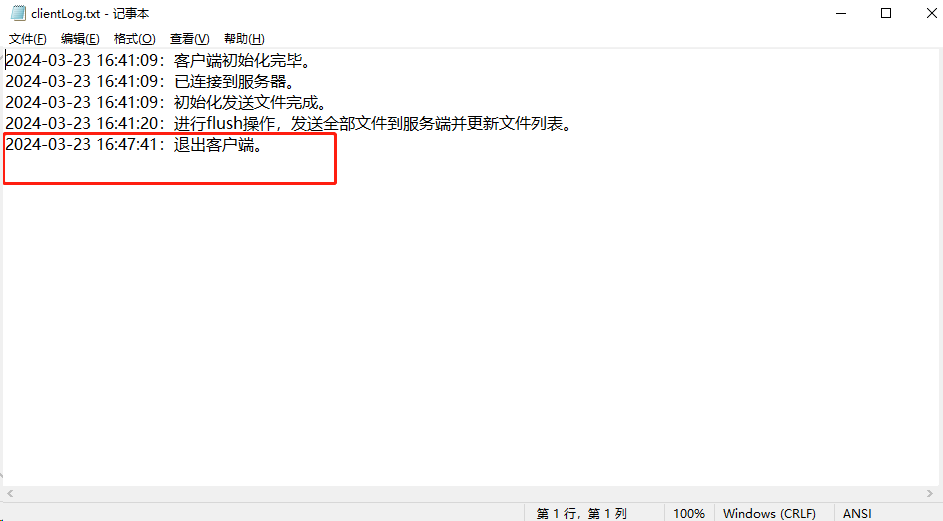
（3）客户端有相应日志内容的记录。



图？ 客户端日志内容

（四）exit

当客户端键入“exit”命令后，客户端自动关闭，表示该用户退出。客户端有相应日志内容的记录。

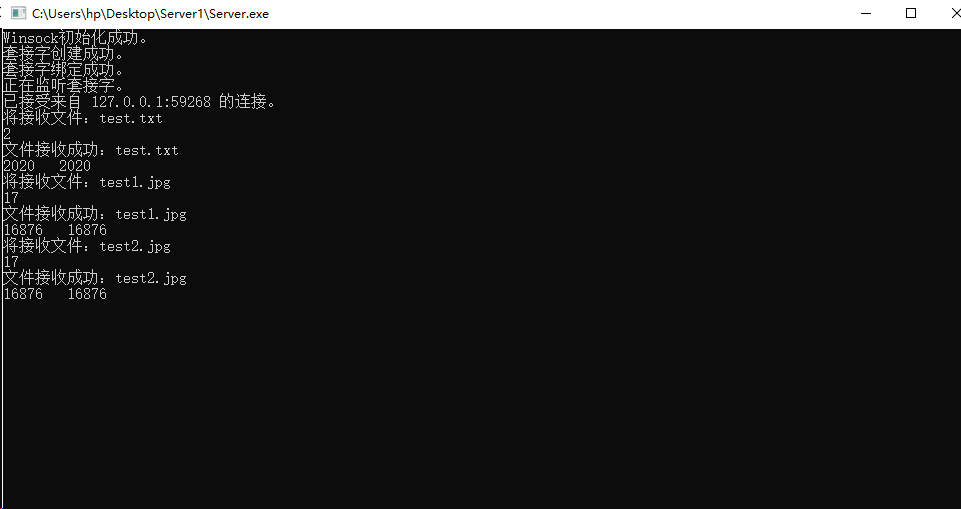


图？ 客户端日志内容

2.2 服务端

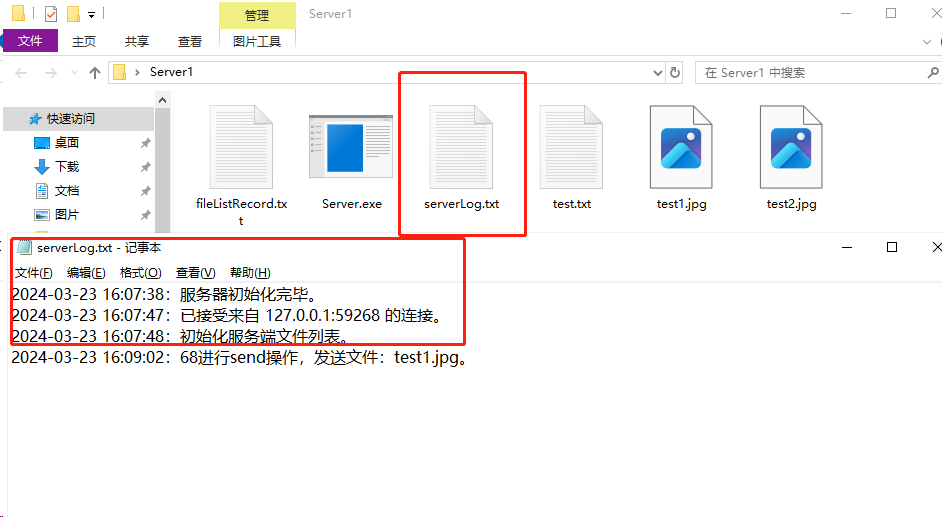
2.2.1 服务端启动

（1）启动服务端文件夹下的“Server.exe”程序，查看命令行界面中的提示信息和工作目录下的日志内容。



图？ 服务端命令行界面提示信息

（2）工作目录下生成日志内容，记录着相应的日期的操作。

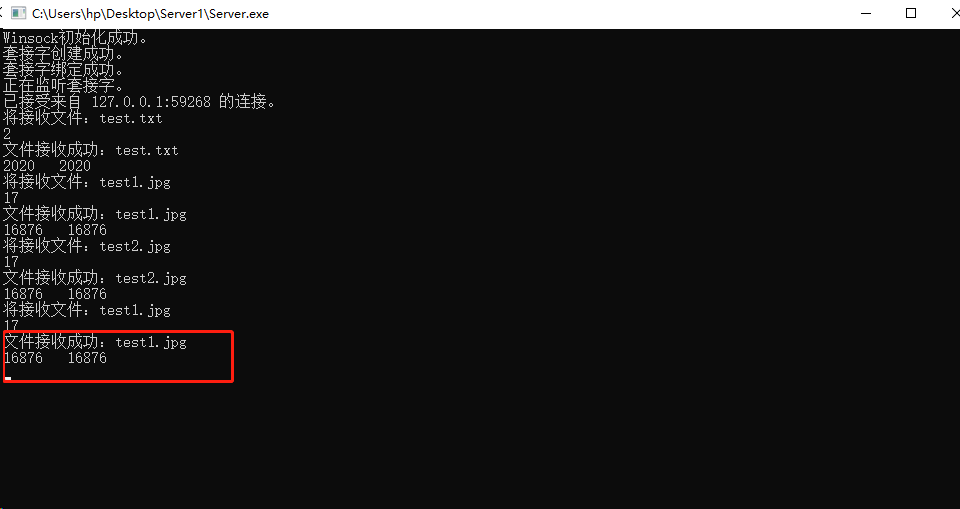


图？ 服务端生成日志

2.2.2 用户命令处理

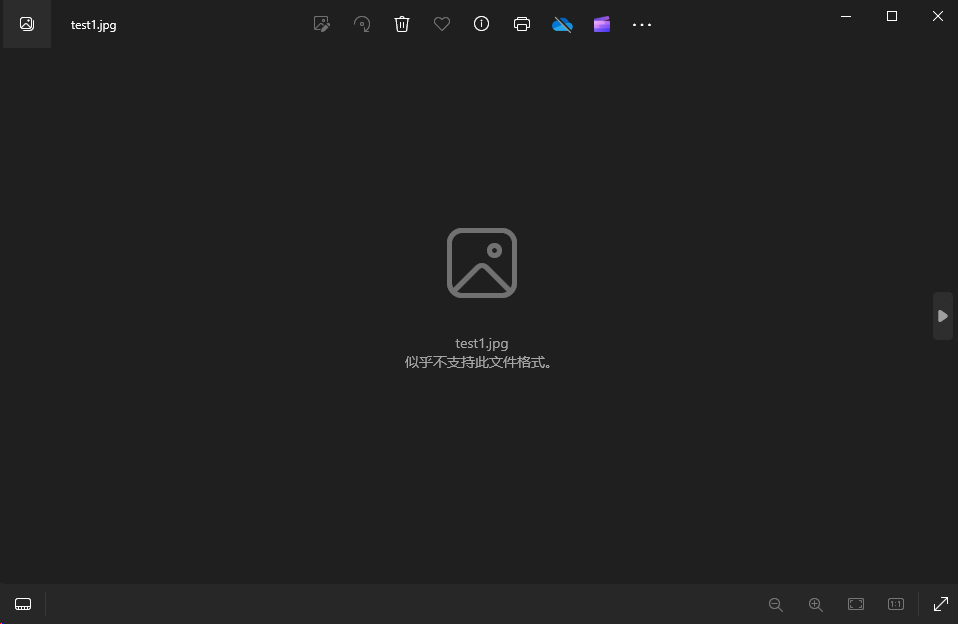
（一）send

（1）接收到用户所发送的加密后的文件。

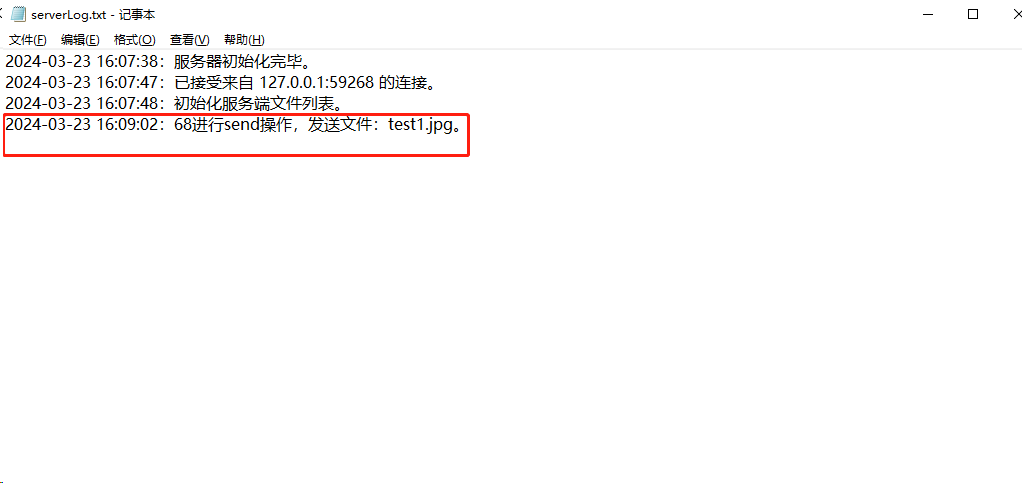


图？服务端接收用户文件

（2）该文件无法正常打开或打开后与原文件不一致。客户端与服务端均有相应日志内容的记录。



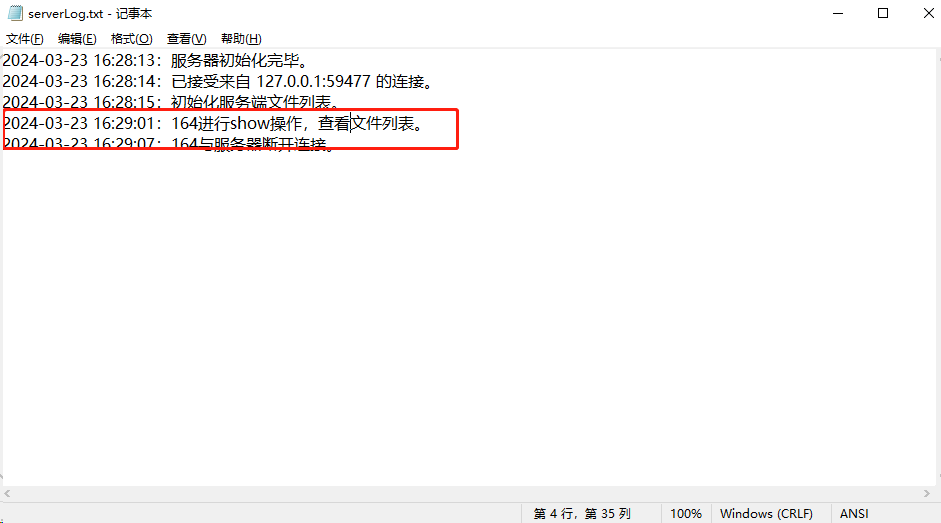
图？文件无法正常打开



图？ 服务端日志内容

（二）show

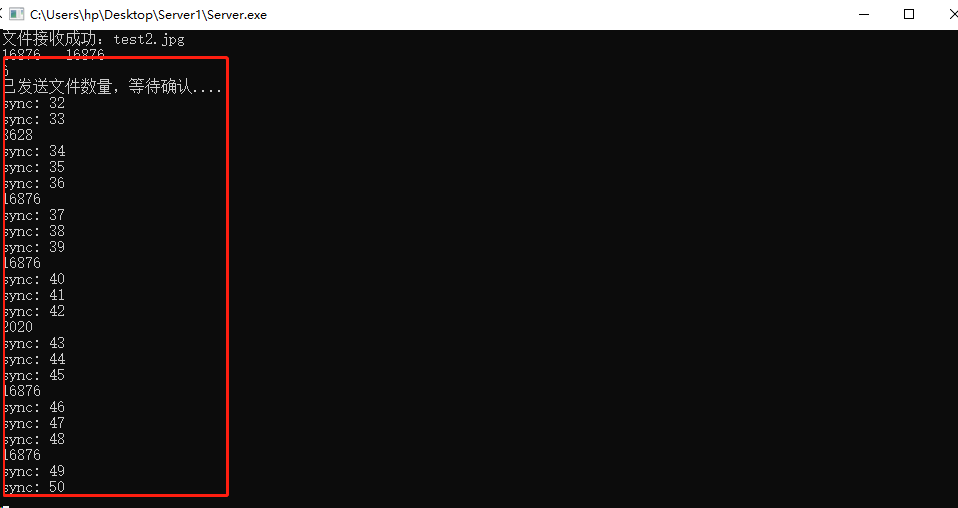
当客户端键入“show”命令后，服务端接收到该操作。服务端有相应日志内容的记录。



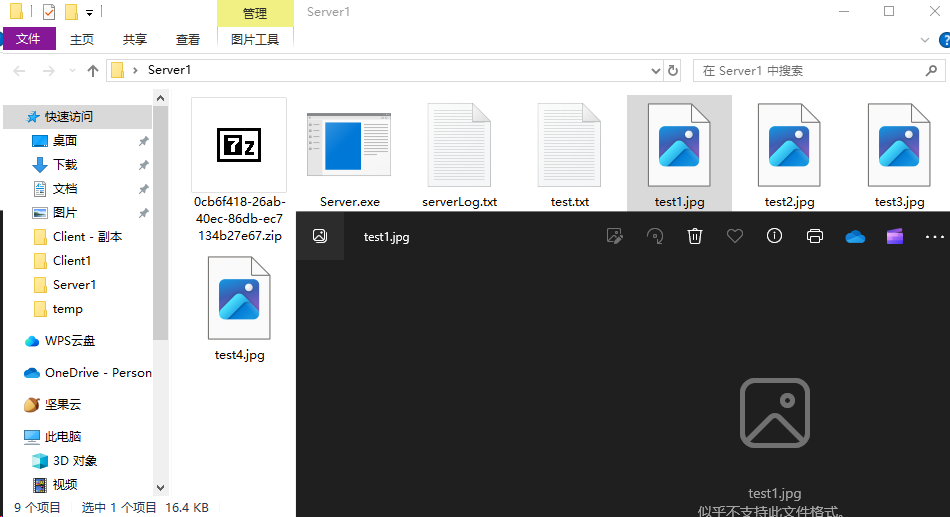
图？ 服务端日志内容

（三）flush

（1）当客户端键入“flush”命令后，服务端接收到该操作。服务端的用户命令行界面中出现响应结果，服务端接收到客户端的文件，这些文件无法正常打开或打开后与原文件不一致。

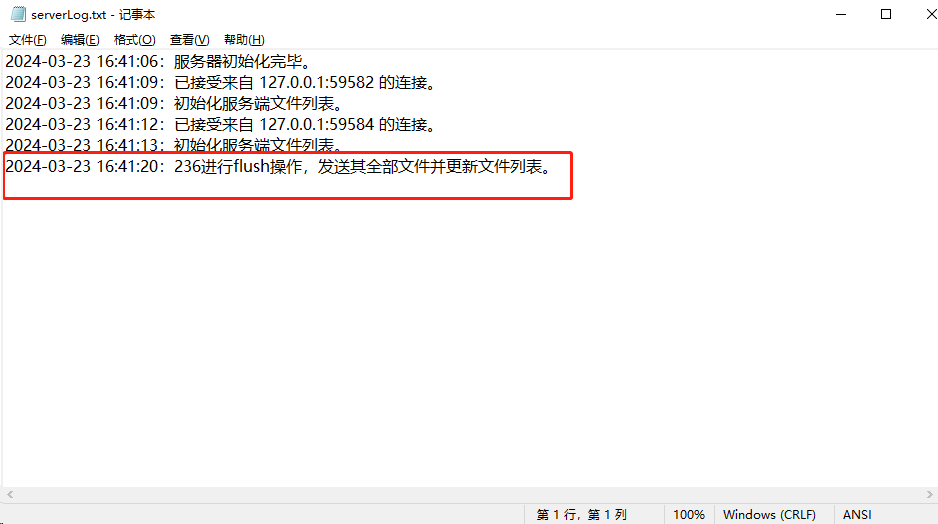


图？服务端命令行界面



图？文件无法正常打开

（2）服务端有相应日志内容的记录。



图？ 服务端日志内容

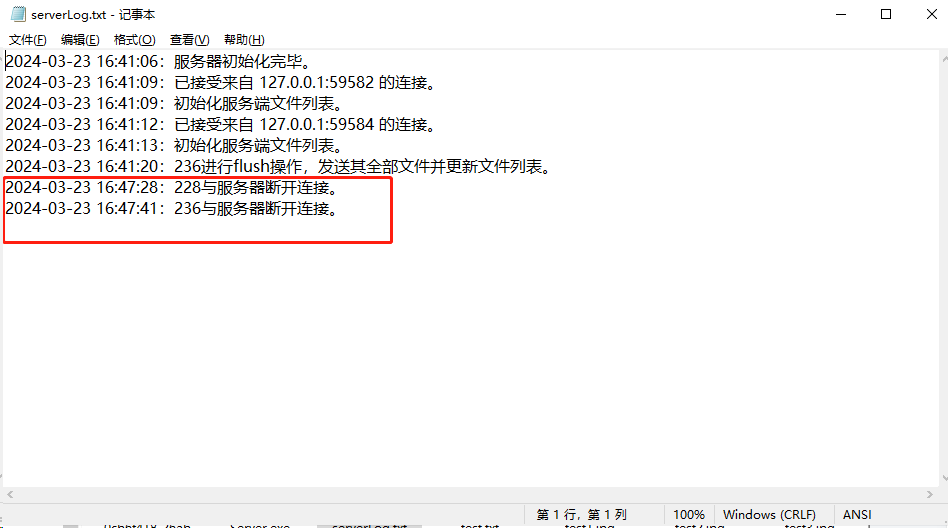
（四）exit

（1）当客户端键入“exit”命令后，服务端接收到该操作。客户端自动关闭，表示该用户退出，服务端更新相应的数据结构。



图？ 更新用户文件列表

（2）服务端均有相应日志内容的记录。



图？服务端日志内容

3 实验结果分析

系统经过连接建立、文件收发、步调有序和文件加解密等方面的测试，并且测试结果符合预期，完成了既定的目标。以下是对实验结果的简要分析：

（1）连接建立：系统成功建立了服务器和客户端之间的连接，表明网络通信部分实现正常。

（2）文件收发：系统能够正确地发送和接收文件，文件传输功能正常。

（3）步调有序：系统能够通过发送确认和等待确认的机制，保证文件传输的有序和高效。

（4）文件加解密：系统能够对文件进行加密和解密操作，加解密功能正常。

综上所述，系统在连接建立、文件收发和文件加解密等方面均表现良好，符合预期。这些结果表明系统设计和实现的正确性和稳定性较高，能够满足需求。

**总结与讨论：**

**1 课程设计学习体会**

**李晓坤：**

完成这两个课程设计对我的网络编程和网络设计能力都是很好的锻炼。在"利用思科模拟器进行校园网设计"中，我学习了如何规划和设计一个大型网络，包括划分子网、配置路由器和交换机等内容。这些技能对于理解复杂网络架构和解决实际网络问题都非常重要。

而在"利用winsock实现局域网文件传输"中，我深入了解了网络编程的基础知识和技术，学会了使用Winsock库来实现局域网内文件传输的功能。这对于理解网络通信原理、掌握Socket编程技术都是非常有益的。

通过这两个课程设计，我不仅提升了对网络设计和编程的理解和技能，还锻炼了问题解决和实践能力。这些经验将对我今后的学习和工作都会有所帮助。

**王若凡：**

通过这次计算机网络课程设计的学习，我不仅在网络设计和网络编程方面有了能力的提升，也让我更加熟悉了从需求分析、设计方案、实施方案、功能测试的完整的项目流程，还让我体会到团队协作的重要性，收获颇丰。

具体而言，在完成题目一网络设计的过程中，我们遇到了许多困难与挑战。比如链路聚合和防火墙配置都是我们之前未曾接触过的事物，需要查找大量资料来学习和理解。而网上的资料很多都不太完整或者网络结构非常简单，需要我们真正理解每条指令的具体含义才能使用，而不能直接照搬。我们遇到的最严峻的问题是防火墙配置问题，我们去图书馆借阅的几本实验指导书都没有介绍防火墙，网上资料也十分不完整，因此走过许多弯路。最后我们咨询了一位网络工程的研究生学长，得知思科模拟器中不能给防火墙配置ospf，会导致整个网络无法正常运行，只能配置静态默认路由。以及要给防火墙ACL应用到接口的入站出站方向等等。最终在小组成员的共同努力下，我们成功完成了题目一。

在完成题目二的过程中，我们同样学习了非常多的资料，从最开始的文件无法正常传输，不断调整接受方式，到能够使用手动命令实现文件文件传送，再到最后实现子网内不同结点自动同步文件，让我提升了编程能力与Socket技术的理解。在完成基本内容的同时，我们还添加了AES加密与Base64编码，大大提升了文件传输的安全性，也是将平时所学的信息安全知识付诸实践。同时，这个项目也提升了我们小组成员的团队协作能力和规划能力，让我们每个人都有许多收获。

最后，感谢老师的指导与小组成员的共同努力，这次课程让我们很有收获，相信对以后的工作和学习也会大有帮助。

**王宠爱：**

计算机网络课程设计的学习过程，于我而言，是一次充满挑战与收获的过程。在这个过程中，我感受到了自己在网络设计和网络编程方面的显著成长。这次学习不仅增强了我的专业技能，更提升了我的团队协作能力。

在完成网络系统设计的过程中，配置链路聚合和OSPF动态路由协议时，不仅需要让网络中的设备能正常通信，而且需要考虑网络安全性及配置后续整体网络的管理是否合理且方便。这就要求对网络整体有深入的规划，并熟练掌握配置命令，但是由于相关理论知识和配置方法尚不熟悉，遇到了不小的困难。通过不断地查阅资料和尝试，逐步掌握了其中的原理和方法，

在完成网络编程的过程中，从最初的简单文件传输，到后来的自动同步和加密传输，我们不断地调整和优化代码，提升了文件传输的效率和安全性。在这个过程中我深入了解了网络通信的原理和机制，学到了Socket编程的基本技术和方法，提升了编程能力。

回顾这次课程设计的学习过程，我深感收获颇丰。我不仅提升了自己的专业技能和实践能力，更深刻地体会到了团队协作的重要性。这些经验和技能将对我今后的学习和工作产生深远的影响。

**2 设计中存在的问题**

**题目1：**

（1）手动分配主机IP：目前主机IP是手动分配的，没有采用DHCP（动态主机配置协议）。这种方式在小型网络中或许可行，但在大型网络如大学网络中，手动分配IP不仅效率低下，而且容易出错。当需要添加新设备时，网络管理员必须手动配置每个设备的IP地址，这增加了管理负担，并限制了网络的扩展性。

（2）链路聚合缺乏冗余：虽然核心层之间采用了两条链路聚合来增加带宽，但没有添加额外的冗余链路。这意味着如果聚合链路中的任何一条链路或交换机出现故障，可能会导致网络连通性的降低或中断。为了提高网络的可用性，应该考虑添加更多的冗余链路和交换机。

（4）地址空间不足：目前网络中只用了IPv4地址，但IPv4的地址空间是有限的，其地址总数为2^32个。随着网络设备的不断增加，特别是在一个大型校园网环境中，IPv4地址可能会很快耗尽。这将导致新的设备无法获得有效的IP地址，从而无法接入网络。

**题目2：**

（1）数据结构冗余

在数据结构设计上存在冗余，导致资源浪费。如服务端维护用户信息时，由于用户信息较多，因此可以整合成为新的数据结构。

（2）自动刷新实现

在文件传输过程中，需要实现自动刷新的功能以保持文件同步或实时更新，而不是手动“flush”操作，即使该操作模拟的是“鼠标右键刷新页面”的操作。

（3）AES加解密效率

AES算法作为一种较为复杂的加密算法，存在多处可以优化加密速度。

（4）密钥安全性

由于使用了AES算法，算法的安全性取决于密钥的安全性，如何保证密钥的安全是一个问题。

（5）服务端文件管理效率

服务端对客户的文件进行统一处理而不是分用户处理，这使得服务端的文件管理效率降低。

**3 可能改进的方向**

**题目1：**

（1）部署DHCP服务器：在大学网络中部署DHCP服务器，实现IP地址的自动分配和管理。DHCP服务器可以根据预设的规则和策略，自动为接入网络的设备分配IP地址、子网掩码、默认网关等网络参数。

（2）增加冗余链路：在核心层之间增加额外的冗余链路，确保在主链路故障时，备用链路能够迅速接管数据传输，维持网络的连通性。配置负载均衡策略，使得在正常情况下，数据可以在多条链路上均衡传输；同时，实施故障切换机制，当检测到某条链路故障时，自动将数据传输切换到其他可用的链路上。

（4）部署IPv6：在网络中部署IPv6是长期解决校园网地址空间不足的最根本方法。可以逐步从IPv4过渡到IPv6，通过双栈技术，使设备能够同时支持IPv4和IPv6的传输，这样，校园网可以逐步过渡到IPv6，同时保持与IPv4的兼容性；或者采用隧道技术，当IPv6数据包需要在IPv4网络中进行传输时，使用隧道技术进行封装和解封装。从而实现IPv6数据包在IPv4网络中的安全传输。

**题目2：**

（1）数据结构冗余：优化数据结构设计，去除冗余部分，简化操作逻辑，提高效率。如服务端维护用户信息时，由于用户信息较多，因此可以整合成为新的数据结构。

（2）自动刷新实现：在客户端和服务端之间建立一种通信机制，当文件发生变化时，自动触发刷新操作。

（3）AES加解密效率：优化AES算法的实现，使用硬件加速或并行计算等技术提高加解密速度。

（4）密钥安全性：使用更复杂的密钥生成算法，或者由可信第三方生成相应的密钥。定期更换密钥，以及加强密钥的存储和传输安全措施。

（5）服务端文件管理效率：为每个用户建立相应的文件目录而不是整合，使用合适的数据结构和算法来管理文件，减少文件操作次数和资源消耗，提高服务端文件管理效率。