**中国矿业大学计算机科学与技术学院**

**2023-2024(2)《计算机网络实验》课程报告(本科)**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 专业班级: | | 计算机科学与技术2022-2班 | | | | 姓名: | 杨晓琦 | | | 学号: | | 08222213 | |
|  | | | 指派的网络号： | | | 172.16.82~172.16.84 | | | | |  | | |
| **序号** | **报告题目** | **基础理论掌握程度** | | **综合知**  **识应用**  **能力** | **实验**  **内容** | **报告**  **格式** | **完成**  **状况** | **工作量** | **学习、**  **工作**  **态度** | **抄袭**  **现象** | | **其它** | **综合**  **成绩** |
|  |  |  | |  |  |  |  |  |  |  | |  |  |
|  |  |  | |  |  |  |  |  |  |  | |  |  |
|  |  |  | |  |  |  |  |  |  |  | |  |  |
|  |  |  | |  |  |  |  |  |  |  | |  |  |
|  |  |  | |  |  |  |  |  |  |  | |  |  |
|  |  |  | |  |  |  |  |  |  |  | |  |  |
|  |  |  | |  |  |  |  |  |  |  | |  |  |
|  |  |  | |  |  |  |  |  |  |  | |  |  |
| 任课教师： | | 杨东平 | | |  |  |  | |  | | |  |  |
| 批阅时间： | |  | | | |  |  | |  | | | | |

**目录**

[6 基于UDP的网络应用编程 1](#_Toc171545288)

[6.1 实验1：基于UDP的socket应用程序设计 1](#_Toc171545289)

[6.1.1 实验目的 1](#_Toc171545290)

[6.1.2 编程环境 1](#_Toc171545291)

[6.1.3 实验内容 1](#_Toc171545292)

[6.1.4 实验内容(代码) 2](#_Toc171545293)

[6.1.4 运行抓包 6](#_Toc171545294)

[6.1.5 运行结果 6](#_Toc171545295)

[6.2 实验体会 6](#_Toc171545296)

# 6 基于UDP的网络应用编程

**实验内容：**

1. 基于UDP的socket应用程序设计

**实验报告要求(必须认真阅读)：**

1. 不得使用实验指导书中的图片(截图)及图片(截图)的相关说明，否则所涉及实验内容视作无效， 并按0分计入成绩；
2. 实验中所使用的IP地址必须按照统一分配给个人的IP地址完成，否则所涉及实验内容视作无效，并按0分计入成绩；
3. 实验内容需要使用的网站、数据文件或其它软件必须作明确的说明，并配有相应的截图(或运行截图或文件目录截图)，否则所涉及实验内容视作抄袭，并按0分计入成绩；
4. 使用仿真软件的实验内容必须同时提交仿真软件所形成的工程文件，否则所涉及的实验内容视作未完成，并按0分计入成绩；
5. 编程的实验内容必须在报告中指出编程环境及其版本号，并同时提交编程环境所形成的工作空间文件或工程文件，以及源代码文件，否则所涉及的实验内容视作未完成，并按0分计入成绩；
6. 实验步骤及实验步骤所涉及的参数必须明确，否则酌情扣除一定的分数；
7. 必须在规定的时间内完成每一次的实验任务，并提交相关文件，否则该次实验内容按0分计入成绩。

说明：如果撰写规范不符合《计算机学院考查类课程报告撰写规范》要求的，整体上酌情扣除1-10分。

## 6.1 实验1：基于UDP的socket应用程序设计

### 6.1.1 实验目的

熟悉基于UDP的C/S网络通信过程，掌握 Socket的数据结构及常用的Socket函数，用C/C++语言编写一套完整的CS模式的UDP网络应用程序并进行相应的运行、测试与抓包分析。

### 6.1.2 编程环境

Visual Studio 2022

### 6.1.3 实验内容

网络编程是深度融合协议原理的实践过程，其中UDP协议扮演了关键角色，它提供了一种非确保性的数据传输服务，显著特征在于其无连接性与不可靠性。无连接特性赋予了UDP数据报传输极高的灵活性，减少了资源占用，并加速了数据处理效率。然而，这种不可靠性在网络环境不稳定时，极易导致数据包的丢失，从而对数据传输的完整性和准确性构成挑战。

鉴于UDP的这些特性，软件开发者在设计基于网络的应用程序时，必须细致考量运行环境的多变性，并采取措施应对可能的数据丢失、传输顺序错乱及数据重复等问题，以减轻这些因素对应用性能和用户体验的负面影响。

特别是在对实时性要求极高的应用场景，如音频、视频流传输中，UDP数据报套接字的优势尤为突出。它能够满足这些应用对低延迟和快速响应的需求，尽管需要开发者在系统设计时额外关注数据的完整性和可靠性保障策略，以确保即使在不良网络条件下也能提供稳定的服务。

### 6.1.4 实验内容(代码)

**说明：**

客户端和服务端都是在本地主机上运行，联网采用手机热点，主机ip地址为192.168.1.24。

客户端代码

#include <iostream>

#include <winsock2.h>

#include <ws2tcpip.h>

#include <windows.h>

#pragma comment(lib, "ws2\_32.lib")

#define SERVER\_PORT 8412

#define BUFF\_LEN 512

#define DESTINATION\_IP "192.168.1.24"

using namespace std;

void send\_udp\_message(SOCKET socket\_fd, struct sockaddr\* dest\_addr)

{

int addr\_len = sizeof(struct sockaddr\_in);

struct sockaddr\_in server\_addr;

int bytes\_sent, bytes\_received;

char buffer[BUFF\_LEN];

while (true) {

cout << "输入消息发送到服务器端:";

cin.getline(buffer, BUFF\_LEN);

bytes\_sent = sendto(socket\_fd, buffer, strlen(buffer), 0, dest\_addr, sizeof(\*dest\_addr));

if (bytes\_sent == SOCKET\_ERROR) {

cout << "sendto 发送失败: " << WSAGetLastError() << endl;

break;

}

memset(buffer, 0, BUFF\_LEN);

bytes\_received = recvfrom(socket\_fd, buffer, BUFF\_LEN, 0, (struct sockaddr\*)&server\_addr, &addr\_len);

if (bytes\_received == SOCKET\_ERROR) {

cout << "recvfrom 接收失败: " << WSAGetLastError() << endl;

break;

}

if (bytes\_received > 0) {

cout << "服务器响应: " << buffer << endl << endl;

} else {

cout << "未收到响应. 服务器可能不在线!" << endl;

}

}

}

int main(int argc, char\* argv[])

{

WSADATA wsaData;

SOCKET client\_socket;

struct sockaddr\_in server\_address;

if (WSAStartup(MAKEWORD(2, 2), &wsaData) != 0) {

cout << "WSAStartup 初始化失败: " << WSAGetLastError() << endl;

return 1;

}

client\_socket = socket(AF\_INET, SOCK\_DGRAM, 0);

if (client\_socket == INVALID\_SOCKET) {

cout << "创建套接字失败: " << WSAGetLastError() << endl;

WSACleanup();

return 1;

}

memset(&server\_address, 0, sizeof(server\_address));

server\_address.sin\_family = AF\_INET;

server\_address.sin\_port = htons(SERVER\_PORT);

if (inet\_pton(AF\_INET, DESTINATION\_IP, &server\_address.sin\_addr) <= 0) {

cout << "无效的地址或地址不受支持" << endl;

closesocket(client\_socket);

WSACleanup();

return 1;

}

send\_udp\_message(client\_socket, (struct sockaddr\*)&server\_address);

closesocket(client\_socket);

WSACleanup();

return 0;

}

服务器端代码

#include <iostream>

#include <winsock2.h>

#include <ws2tcpip.h>

#pragma comment(lib, "ws2\_32.lib")

#define SERVER\_PORT 8412

#define BUFF\_LEN 512

using namespace std;

void receive\_udp\_messages(SOCKET socket\_fd)

{

char buffer[BUFF\_LEN];

struct sockaddr\_in client\_address;

int buffer\_len;

int bytes\_received;

int client\_address\_len = sizeof(client\_address);

while (true) {

cout << "等待客户端响应..." << endl;

memset(buffer, 0, BUFF\_LEN);

buffer\_len = client\_address\_len;

bytes\_received = recvfrom(socket\_fd, buffer, BUFF\_LEN, 0, (struct sockaddr\*)&client\_address, &buffer\_len);

if (bytes\_received == SOCKET\_ERROR) {

cout << "recvfrom 接收失败: " << WSAGetLastError() << endl;

return;

}

cout << "收到客户端消息: " << buffer << endl;

memset(buffer, 0, BUFF\_LEN);

strcpy\_s(buffer, "已收到");

cout << "返回确认消息：收到" << endl << endl;

sendto(socket\_fd, buffer, strlen(buffer), 0, (struct sockaddr\*)&client\_address, buffer\_len);

}

}

int main(int argc, char\* argv[])

{

WSADATA wsaData;

SOCKET server\_socket;

struct sockaddr\_in server\_address;

int bind\_result;

if (WSAStartup(MAKEWORD(2, 2), &wsaData) != 0) {

cout << "WSAStartup 初始化失败: " << WSAGetLastError() << endl;

return -1;

}

server\_socket = socket(AF\_INET, SOCK\_DGRAM, 0);

if (server\_socket == INVALID\_SOCKET) {

cout << "创建套接字失败: " << WSAGetLastError() << endl;

WSACleanup();

return -1;

}

memset(&server\_address, 0, sizeof(server\_address));

server\_address.sin\_family = AF\_INET;

server\_address.sin\_addr.s\_addr = htonl(INADDR\_ANY);

server\_address.sin\_port = htons(SERVER\_PORT);

bind\_result=bind(server\_socket, (struct sockaddr\*)&server\_address, sizeof(server\_address));

if (bind\_result == SOCKET\_ERROR) {

cout << "绑定套接字失败: " << WSAGetLastError() << endl;

closesocket(server\_socket);

WSACleanup();

return -1;

}

receive\_udp\_messages(server\_socket);

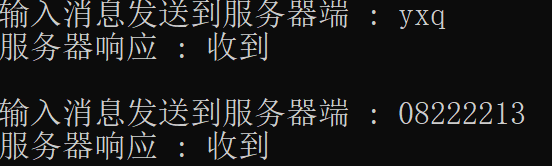
closesocket(server\_socket);

WSACleanup();

return 0;

}

### 6.1.4 运行结果



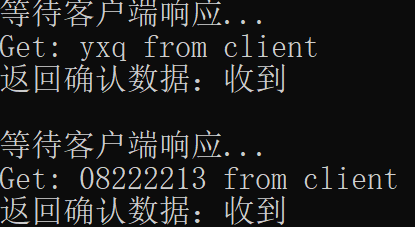
图6-1 客户端运行结果

图6-2 服务端运行结果

### 6.1.5 运行抓包

图6-3 抓包过滤

## 6.2 实验体会

在完成了基于UDP的socket应用程序设计的实验后，我深感网络编程的复杂性和挑战性，同时也体会到了UDP协议在特定场景下的优势。这次实验不仅加深了我对套接字（Socket）编程的理解，也让我对C/S模式下的UDP通信有了更为直观的认识。

首先，实验要求我们分别设计基于UDP的client和server程序，并在两台不同的主机上运行测试。这一过程中，我深刻体会到了网络通信的复杂性。在编写程序时，需要考虑到各种可能的网络环境和异常情况，如网络延迟、数据包丢失等。因此，我在编写程序时特别注意了错误处理和异常捕获，以确保程序的健壮性和稳定性。

其次，通过实际运行和测试程序，我更加深入地理解了UDP协议的特点。UDP协议是一种无连接、不可靠的传输协议，它不提供数据包的排序、错误检查、重传等机制。这意味着在网络质量不佳的环境下，UDP通信可能会出现数据包丢失、乱序等问题。然而，这也使得UDP协议具有资源消耗小、处理速度快的优点，特别适合于音频、视频的实时传输等场合。在实验中，我通过发送和接收数据，亲身体验了UDP协议的这种特点。

此外，实验还要求我们在两台主机上运行抓包软件，抓取并分析程序运行过程中发送和接收的数据。这一步骤让我对网络通信的底层细节有了更为深入的了解。通过分析数据包的内容、大小、时间戳等信息，我能够更加清晰地看到数据在网络中的传输过程，以及UDP协议的工作原理。

最后，通过这次实验，我深刻体会到了网络编程的实践性和挑战性。网络编程不仅要求程序员具备扎实的编程基础，还需要对网络协议、操作系统等底层知识有深入的了解。同时，网络编程也充满了各种可能的问题和挑战，需要程序员具备灵活的思维和解决问题的能力。

总之，这次基于UDP的socket应用程序设计的实验让我受益匪浅。我不仅掌握了套接字编程的基本知识和技巧，还深入了解了UDP协议的特点和工作原理。我相信这次实验的经验和体会将对我未来的学习和工作产生积极的影响。

----------------------------------以下表格用于评阅，不得擅自修改、删除--------------------------------------------

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **序号** | **报告内容** | **基础理论掌握程度** | **综合知**  **识应用**  **能力** | **报告**  **内容** | **报告**  **格式** | **完成**  **状况** | **工作量** | **学习、**  **工作**  **态度** | **抄袭**  **现象** | **其它** | **综合**  **成绩** |
| **8** | **应用层** | 请选择 | 请选择 | 请选择 | 请选择 | 请选择 | 请选择 | 请选择 | 学号:  姓名: | 目录、章节编目、错别字、乱码、截图、程序功能、程序源代码、源代码注释、体会、文献 |  |
| 批阅时间 | | 2024年5月22日 | | | | | | | | | |

----------------------------------以上表格用于评阅，不得擅自修改、删除--------------------------------------------