吉 林 大 学
软件学院
实 验 报 告

实验名称	原始套接字编程				
课程名称	计算机网络课程设计				
姓名		学号		成绩	
提交日期	2024. 04. 12	座位号			

1. 实验目的

掌握原始套接字编程。

2. 实验内容

本实验要求通过原始套接字编程实现一个简单的网络数据包捕获工具，能够捕获特定目标 IP 地址的数据包，并解析显示其中的 IP 头部和 TCP 头部信息。

3. 实验分析

原始套接字编程允许应用程序直接访问数据链路层和网络层，可以自定义数据包的格式和处理过程。因此，在本实验中，需要完成以下步骤：

- 创建原始套接字并绑定到指定 IP 地址。
- 设置套接字选项，使其能够接收所有传入的数据包。
- 循环接收数据包，并解析其中的 IP 头部和 TCP 头部信息。
- 将解析得到的信息进行输出和展示。

4. 问题解答

●数据包嗅探器程序（`class PacketSniffer`）

1) 初始化数据包嗅探器（`PacketSniffer(const char* ipAddress)`）

- 初始化 Winsock。
- 创建原始套接字 sock。
- 启用 IP_HDRINCL 选项以在接收的数据包中包含 IP 头部。
- 将原始套接字绑定到指定的 IP 地址。

2) 启动数据包捕获和处理（`void Start()`）

- 循环接收数据包并调用 ProcessPacket()处理接收到的数据包。

3) 处理接收到的数据包（`void ProcessPacket(char* buffer, int bytesRecv)`）

- 解析接收到的数据包，提取 IP 头部和 TCP 头部信息。
- 打印出数据包中的源 IP、目的 IP、源端口和目的端口等信息。

4) 关闭数据包嗅探器（`~PacketSniffer()`）

- 关闭服务器套接字。等待所有子线程或进程处理完毕，释放资源。

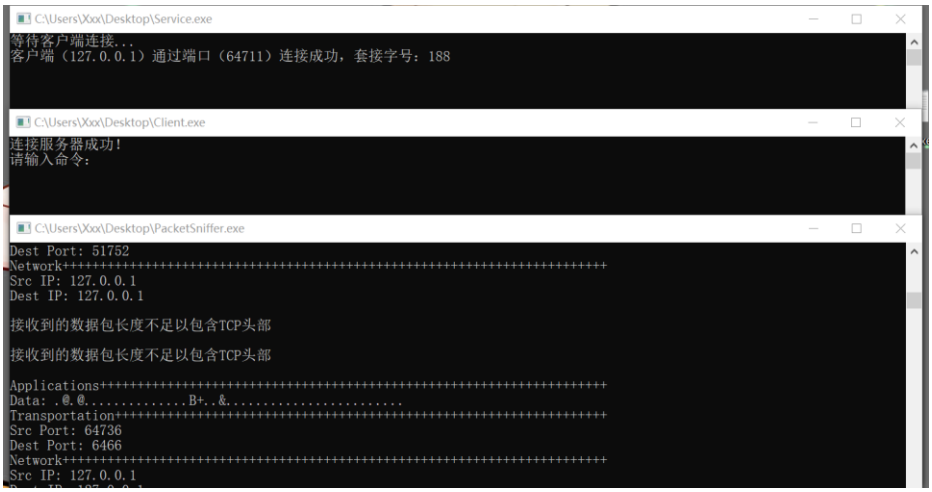
4. 3 核心代码（有必要的注释）

```
void PacketSniffer::ProcessPacket(char* buffer, int bytesRecv) {
    IPHeader* ipHeader = (IPHeader*)buffer;
    if (bytesRecv < sizeof(IPHeader)) {
        std::cerr << "接收到的数据包长度不足以包含IP头部" << std::endl;
        std::cout << std::endl;
        return;
    }

    if (ipHeader->protocol == IPPROTO_TCP) {
```

```
        // 计算IP头部长度
        int ipHeaderLen = (ipHeader->headLen & 0x0F) * 4;
        TCPHeader* tcpHeader = (TCPHeader*)(buffer + ipHeaderLen);

        if (bytesRecv < ipHeaderLen + sizeof(TCPHeader)) {
            std::cerr << "接收到的数据包长度不足以包含TCP头部" << std::endl;
            std::cout << std::endl;
            return;
        }

        // 打印应用层数据
        std::cout << "Applications++++++++++++++++++++++++++++++++++++++++++++++
                  ++++++++++++++++++++++++" << std::endl;
        std::cout << "Data: ";
        char* tcpData = buffer + 5 + 4 * ((tcpHeader->headLen & 0xf0) >> 4);
        int tcpDataSize = bytesRecv - 5 - 4 * ((tcpHeader->headLen & 0xf0)>>4);
        for (int i = 0; i < tcpDataSize; i++) {
            if (tcpData[i] >= 32 && tcpData[i] < 127) {
                std::cout << tcpData[i];
            }
            else {
                std::cout << ".";
            }
        }
        std::cout << std::endl;

        // 打印传输层信息
        std::cout << "Transportation++++++++++++++++++++++++++++++++++++++++++++
                  ++++++++++++++++++++++++" << std::endl;
        std::cout << "Src Port: " << ntohs(tcpHeader->sourcePort) << std::endl;
        std::cout << "Dest Port: " << ntohs(tcpHeader->destinPort) << std::endl;

        // 打印网络层信息
        std::cout << "Network++++++++++++++++++++++++++++++++++++++++++++++++++
                  ++++++++++++++++++++++++" << std::endl;
        std::cout << "Src IP: " << inet_ntop(AF_INET, &(ipHeader->sourceAddr),
                                  buffer, INET_ADDRSTRLEN) << std::endl;
        std::cout << "Dest IP: " << inet_ntop(AF_INET, &(ipHeader->destinAddr),
                                  buffer, INET_ADDRSTRLEN) << std::endl;
        std::cout << std::endl;
    }
}
```

4.4 测试方法、测试数据与测试结果

1) 运行程序



2) 捕获客户端发给服务器的信息

3) 捕获服务器发给客户端的信息

4.5 程序的使用说明

- **运行服务器和客户端程序**。
- **运行数据包嗅探器程序**：以管理员权限运行程序。
- **捕获网络数据包**：程序将开始捕获网络接口上的数据包。
- **解析信息**：对每个捕获到的数据包，程序会解析 IP 头部和 TCP 头部信息，并输出源 IP 地址、目的 IP 地址、源端口号和目的端口号和数据等信息到控制台。

4.6 总结

- **程序运行效果评价**
  - **功能实现**：程序成功实现了通过原始套接字捕获网络接口上的数据包。对捕获到的 TCP 数据包进行了解析，提取了 IP 头部和 TCP 头部的关键信息。输出了源 IP 地址、目的 IP 地址、源端口号和目的端口号等网络层信息，便于理解网络通信的细节。

- **遇到的问题及解决办法**
  - **权限问题：**需要以管理员权限运行程序，以便程序能够成功捕获网络数据包。这是程序运行的基本要求。
  - **注意程序的安全性：**使用原始套接字涉及到较高的权限和操作系统资源。

- **程序特色说明**
  - **原始套接字编程：**本程序利用原始套接字实现了数据包的捕获和解析，展示了网络编程中的底层实现原理。
  - **实时数据解析：**程序能够实时捕获和解析网络数据包，将关键信息输出到控制台，帮助用户实时监测和分析网络通信状态。
  - **模块化设计：**采用面向对象的模块化设计，将服务器端和客户端功能封装为类，使得代码结构清晰，易于理解和维护。