计算机网络——使用 UDP 实现可靠的文件传输

1711361 刘炼

南开大学网络空间安全学院

摘要：本文主要介绍了本次实验的相关内容，本次实验的目的是实现使用 UDP 实现可靠的文件传输，设计相应的协议，可以支持多用户上传和下载，最后介绍代码可实现的功能，及相关的代码实现。

关键词：UDP，可靠文件传输，协议设计

1 作业要求

程序的基本功能要求：

要求：

（1）下层使用 UDP 协议（即使用数据报套接字完成本次程序）

（2）完成客户端和服务器端程序。

（3）实现可靠的文件传输

2 协议设计

2.1 设计要求

（1）下层使用 UDP 服务。

（2）支持多用户。

（3）多用户文件的上传和下载可以仅支持当前目录。

（4）给出协议的具体内容。

（5）给出收发双方的交互日志。

2.2 具体协议

协议的基本思想是分组传输。

1、开始传输文件时，发送方根据文件长度，以一个包 1000byte，一组 10 个包，计算出总共包数，发送给接收方。

2、接收方收到包数后，返回 ack，并申请相应数目的“检查数组”，每一组传完将相应的位置为 true（该数组的作用是等到全部传输完后检查整个“检查数组”是否全为 true，若是则传输成功，若不是则建议重传）。

3、发送端开始一组的传输。发送端发送该组的组号与该组的包数（一般为 10 个，最后一组可能不为 10）。

4、接收端收到组信息后将“包检查数组”对应数量的位置为 false，并返回 ack（该数组的作用是检查该组的包是否都收到）。

5、发送端开始发数据包，每个数据包都包含“组序号”与“包序号”。接收端收到数据包后，根据其“包序号”将其缓存进“缓存数组”，并“包检查数组”对应的位置为 true。

6、发送端将该组所有数据包发出后，发出“该组发完”指令。

7、接收端收到“该组发完”指令，查看“包检查数组”，

7.1、若全为 true，则返回“该组已全部收到”指令，将“缓存数组”中的数据全部写入文件后清空，将“包检查数组”重置；

7.2、若不全为 true，则说明有丢包，将丢包项的“包序号”发给发送端。

8.1、发送端若收到丢包项的“包序号”，根据“包序号”将相应的包重发。

8.2、发送端若收到“该组已全部接收”，则重复 3-8.2 号动作。

9、发送端所有组发送完成后发送“文件全部发送完毕”指令。

10、接收端收到“文件全部发送完毕”指令，检查“检查数组”是否全为 true，若是则传输成功，若不是则建议重传。

PS1：协议中的所有控制信号，比如“该组已全部接收”，采用超时重传，即若接收端发出“该组已全部接收”指令长时间没收到接下来的指令则重发一次。

PS2：该程序中，由服务端维护一个用户列表，由“IP 地址 + 端口号”的方式区分用户。每个用户的上传/下载事件都存在其对应的数据结构中。该协议相较于“发送端每发一个包，接收端回复一个 ack”的方式的优势在于，一是减少了 ack 的数量，也就减少了发送端等待 ack 所浪费的时间，提高了传输效率。二是一定程度上避免了传输乱序的问题，每组数据包到达的先后次序对传输没有影响。

PS3: 该程序中，当对连接进行测试的时候，会为不同的客户端分配一个单独端口进行数据的传输，根据这样一种实现方式，每个客户端都是和服务器的不同端口进行连接，可以实现同时传输。

3 程序实现

3.1 页面初始化

3.1.1 客户端

首先实现客户端初始化端口的界面，其如图3-1所示，其中，可以设置本机的端口，从而选取不同的端口与服务器进行连接，默认端口号为 555

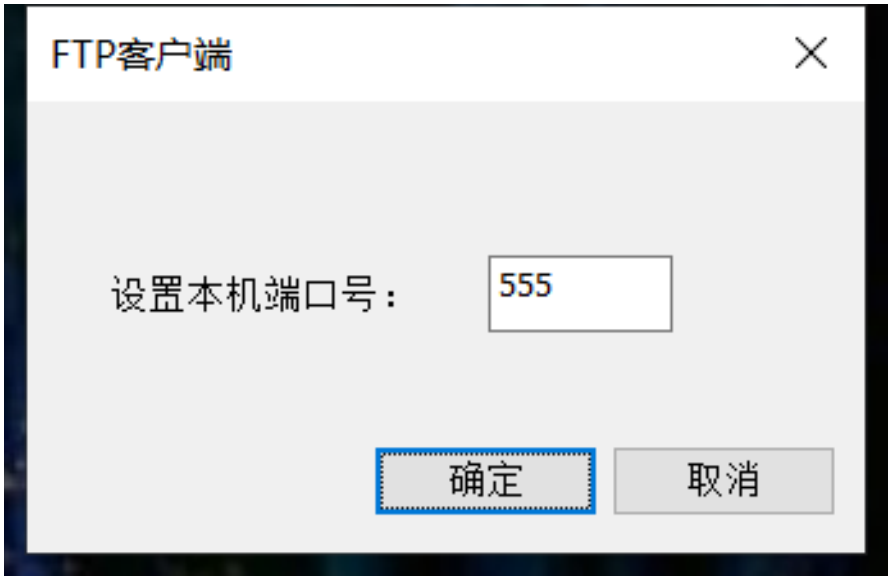

图 3-1 客户端初始化端口界面

端口初始化完成后，点击确定，会跳转到新的客户端界面，其为包括了测试连接，浏览，上传，下载和刷新功能，并且显示日志交互的新界面，如图3-2所示：

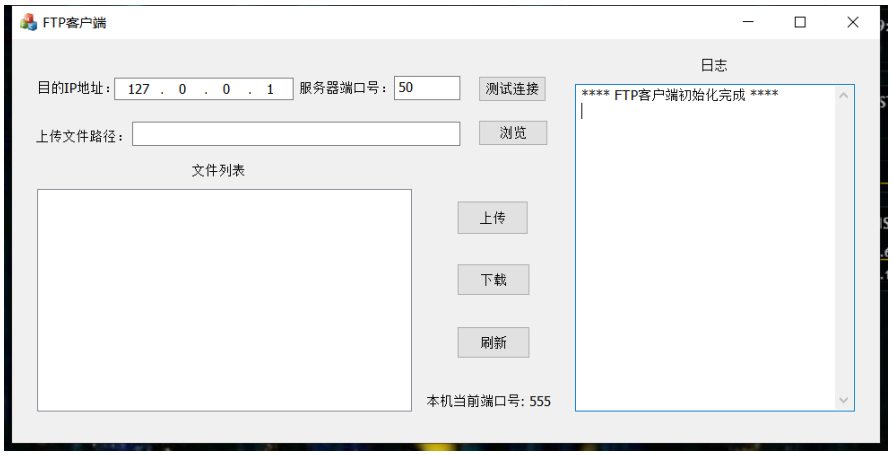

图 3-2 客户端功能界面

3.1.2 服务器

在服务器端，首先需要选择当前的工作路径和服务器登录端口，主要是根据此来选择服务器工作的当前路径，并根据此路径进行上传下载的位置选择，具体界面如图3-3所示：

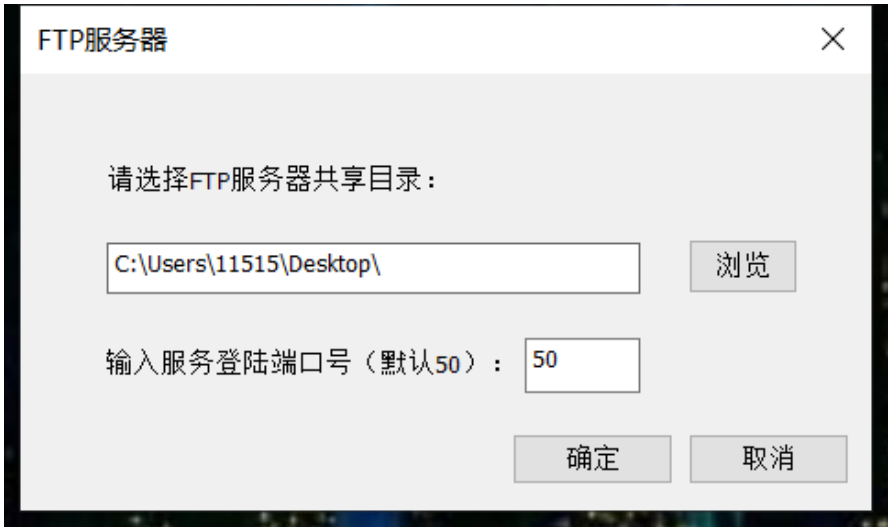

图 3-3 服务器端口路径初始化

初始化完成后，选择确定，然后服务器端会跳转到一个日志记录页面，记录服务器端和客户端的交互日志，如图3-4所示：

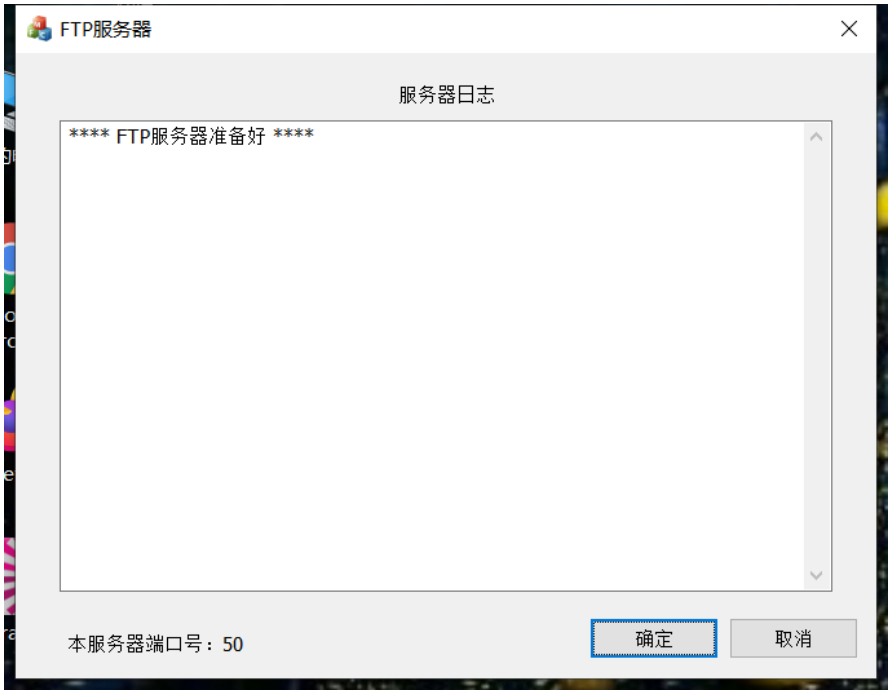

图 3-4 服务器日志交互过程

3.2 连接测试

当开启客户端和服务器相应服务之后，首先需要对连接本身进行确定，因此需要测试连接其具体过程为：首先客户端向服务器端发送连接建立的请求，然后服务器接收到请求后，则返回相应的操作码告

诉客户端连接已经建立。

3.2.1 客户端

对于客户端而言，其具体代码实现如下所示:

Listing 1 UDPFTPClientDlg.cpp

```
void CUDPFTPClientDlg::OnBnClickedButtonTestconnect()
{
  // TODO: 在此添加控件通知处理程序代码
  UpdateData(TRUE);
  MySock.sendPort = HostPort_;
  BYTE nf1, nf2, nf3, nf4;
  IPaddress.GetAddress(nf1, nf2, nf3, nf4);
  MySock.sendIP.Format(L"%u.%u.%u.%u", nf1, nf2, nf3, nf4);
  MySock.LinkCheck= true;
  MySock.AsyncSelect(FD_WRITE);
}
```

3.2.2 服务器

对于服务器而言，其收到了相应的操作码，因此会根据操作码进行解析并返回相应内容：

Listing 2 CUDP.cpp

```
case 100://连接请求
{
  dlg->Log += head + L"客户端请求测试连接\r\n";
  OpeartionPack pack;
  pack.Operation = 200;
  pack.Port = 2055 + nowUser;
  create_port(pack.Port);
  SendTo(&pack, sizeof(OpeartionPack), user[nowUser].port, user[nowUser].IP, 0);
  break;
}
```

3.3 刷新列表

除了需要进行连接测试之外，在客户端通过服务器下载文件之前，需要首先向服务器端发送相应的浏览请求，刷新当前的服务器列表。对于客户端而言，只需要发出相应请求即可，对服务器而言，需要浏览当前路径下的所有文件信息并返回。

3.3.1 客户端

对于客户端而言，当点击相应的按钮的时候，需要对其进行相应的处理，即为：

Listing 3 UDPFTPClientDlg.cpp

```
void CUDPFTPClientDlg::OnBnClickedButtonRefresh()
{
  // TODO: 在此添加控件通知处理程序代码
  UpdateData(TRUE);
  //MySock.sendPort = HostPort_;
  BYTE nf1, nf2, nf3, nf4;
  IPaddress.GetAddress(nf1, nf2, nf3, nf4);
  MySock.sendIP.Format(L"%u.%u.%u.%u", nf1, nf2, nf3, nf4);
  MySock.FindFile = true;
  MySock.AsyncSelect(FD_WRITE);
}
```

3.3.2 服务器

对于服务器而言，首先根据相应的操作码，进行相应的接收处理，并按照相应的说明，浏览当前路径下的所有文件信息，并返回

Listing 4 CUDP.cpp

```
case 101://请求文件列表
{
  dlg->Log += head + L"客户端请求文件列表\r\n";
  dlg->UpdateData(FALSE);
  dlg->SendDlgItemMessage(IDC_EDIT_LOG, WM_VSCROLL, SB_BOTTOM, 0);
  Pack packFile;
  CString returnStr;
  packFile.Operation = 201;//返回相应操作码
  returnStr = findAllFile();//并返回所有的文件
  int length = returnStr.GetLength();
  for (int i = 5; i < length; i++)
    if (returnStr.GetAt(i) == '\\')
      packFile.Content[i - 5] = 0;
    else
      packFile.Content[i - 5] = returnStr.GetAt(i);
  packFile.Content[length - 5] = 0;
  packFile.Content[length - 4] = 0;
  for(int i=length-3;i<1000;i++)packFile.Content[i] = 0;
```

```
    SendTo(&packFile, sizeof(packFile), user[nowUser].port, user[nowUser].IP, 0);
    break;
  }

  //可以看到，上述过程中，需要使用到findAllFile()函数，故而还需要对其进行实现

  CString CUDP::findAllFile()
{
        CString fileNameList;
        fileNameList.Empty();
        bool finding = finder.FindFile(findWild);
        while (finding)
        {
                finding = finder.FindNextFile();
                fileNameList += finder.GetFileName() + L"\\";
        }
        return fileNameList;
}
```

3.4 浏览

对于浏览而言，只涉及到客户端，没有服务器相应的操作，其具体实现非常简单，只需要调用相应的控件即可：

Listing 5 UDPFTPClientDlg.cpp

```
  void CUDPFTPClientDlg::OnBnClickedButtonBrowse()
  {
    // TODO: 在此添加控件通知处理程序代码
    BOOL isOpen = TRUE;
    CString defaultDir = L"C:\\Users\\11515\\Desktop";  //默认打开的文件路径
    CString fileName = L"";
    CString filter = L"所有文件 (*.*)|*.*||";                    //文件过虑的类型
    CFileDialog fileDlg(isOpen, defaultDir, fileName, OFN_HIDEREADONLY | OFN_READONLY,
    INT_PTR result = fileDlg.DoModal();
    // 显示打开文件对话框
    if (result == IDOK)
    {
      // 得到我们的选择文件的地址，用于之后的打开文件的操作
      UpLoadName = fileDlg.GetFileName();
      UpLoadPath = fileDlg.GetPathName();
      ChoosedFile = true;
```

```
        //SetDlgItemText(IDC_EDIT_PATH, FilePath);
        UpdateData(FALSE);
    }
}
```

这里只是调用了相应的控件，浏览主机的文件目录。

3.5 上传

对于上传，其首先需要找到相应上传的文件，其已经在3.4中实现，在选定好文件之后，需要向服务器发送相应的请求，服务器接收到请求后，会返回相应的操作码，并准备收取客户端发过来的文件信息，当客户端发送了文件信息之后，服务器端利用文件信息决定接受的方式，并创建一个新的文件来进行写入。具体的内容的交互过程，请参考3.7的具体实现

3.5.1 客户端

1. 选择上传键进行上传

Listing 6 UDPFTPClientDlg.cpp

```
void CUDPFTPClientDlg::OnBnClickedButtonUpload()
{
    // TODO: 在此添加控件通知处理程序代码
    if (ChoosedFile)
    {
        UpdateData(TRUE);
        //MySock.sendPort = HostPort_;
        BYTE nf1, nf2, nf3, nf4;
        IPaddress.GetAddress(nf1, nf2, nf3, nf4);
        MySock.sendIP.Format(L"%u.%u.%u.%u", nf1, nf2, nf3, nf4);
        MySock.UpLoad = true;
        MySock.UpLoadFileName = UpLoadName;
        MySock.UpLoadFilePath = UpLoadPath;
        MySock.AsyncSelect(FD_WRITE);
    }
    else
    {
        MessageBox(L"请选择要上传的文件", L"上传文件失败！", MB_ICONERROR);
    }
}

//这里首先将UpLoad 置为true, 然后进行操作
```

```
if (UpLoad)
        {
                //InterPack pack;
                Pack pack;
                pack.Operation = 110;//请求上传文件
                pack.Port = recvPort;
                int position = 0;
                int length = UpLoadFileName.GetLength();
                for (int i = 0; i < length; i++)
                        pack.Content[position++] = UpLoadFileName.GetAt(i);
                pack.Content[position++] = 0;
                //pack.Filename = UpLoadFileName;
                dlg->Log += L"请求上传文件 " + UpLoadFileName + L" 中……\r\n";
                dlg->UpdateData(FALSE);
                dlg->SendDlgItemMessage(IDC_EDIT_Log, WM_VSCROLL, SB_BOTTOM, 0);
                SendTo(&pack, sizeof(Pack), sendPort, sendIP, 0);
                SetTimer(hHwnd, -3, 5000, CheckReceive);
                UpLoad = false;
        }
```

服务器收到上传请求之后，回复相应的消息，然后客户端开始收集文件信息，并对其进行发送

Listing 7 CUDP.cpp

```
case 210:
{
  KillTimer(hHwnd, -3);
  dlg->Log += L"服务器收到上传请求，准备上传文件信息\r\n";
  dlg->UpdateData(FALSE);
  dlg->SendDlgItemMessage(IDC_EDIT_Log, WM_VSCROLL, SB_BOTTOM, 0);
  FileInfoPack pack;
  pack.Operation = 111;//文件信息
  if (OpenReadFile(UpLoadFilePath) == 0)
  {
    dlg->Log += L"文件读取失败！请检查路径和文件名是否正确！\r\n";
    dlg->UpdateData(FALSE);
    dlg->SendDlgItemMessage(IDC_EDIT_Log, WM_VSCROLL, SB_BOTTOM, 0);
    CloseFile();
  }
  else
  {
    PktNum = (file.GetLength() - 1) / 1000 + 1;
    pack.GroupNumber = (PktNum - 1) / 10 + 1;
```

```
        GroupNum = pack.GroupNumber;
        SendTo(&pack, sizeof(FileInfoPack) ,sendPort,sendIP, 0);
        FileGroup = new bool[pack.GroupNumber];
        for (int i = 0; i < pack.GroupNumber; i++)FileGroup[i] = false;
        SetTimer(hHwnd, -210, 5000, CheckReceive);
    }
    break;
}

//文件进行分组分块传输，发送相应的信息，然后接受到之后，客户端准备发送文件

case 220:
{
    KillTimer(hHwnd, -4);
    dlg->Log += L"服务器收到请求\r\n";
    dlg->Log += L"收到文件信息，准备接收文件\r\n";
    dlg->UpdateData(FALSE);
    dlg->SendDlgItemMessage(IDC_EDIT_Log, WM_VSCROLL, SB_BOTTOM, 0);
    GroupNum = fileinfopack->GroupNumber;
    FileGroup = new bool[GroupNum];
    for (int i = 0; i < GroupNum; i++)FileGroup[i] = false;
    OpeartionPack pack;
    pack.Operation = 122;//确认文件信息
    SendTo(&pack, sizeof(OpeartionPack), sendPort, sendIP, 0);
    DownLoading = true;
    OpenWriteFile(DownLoadFilePath + DownLoadFileName);
    SetTimer(hHwnd, -221, 5000, CheckReceive);
    break;
}
```

3.5.2 服务器

服务器首先会收到相应的上传请求，请求之后，会发送相应的文件信息，然后根据文件信息，服务器跳转到接受文件的状态下：

Listing 8 CUDP.cpp

```
case 110://客户端请求上传文件
{
    int i;
    char tempStr[1000];
    for (i = 0; recvPack.Content[i] != 0; i++)
```

```
      tempStr[i] = recvPack.Content[i];
    tempStr[i] = 0;
    user[now].sendFileName.Format(L"%s", CStringW(tempStr));
    //user[now].sendFileName = interpack->Filename;
    out.Format(L"准备接收文件信息 %s %d\r\n", user[now].sendFileName, now);
    dlg->Log += head + out;
    dlg->UpdateData(FALSE);
    dlg->SendDlgItemMessage(IDC_EDIT_LOG, WM_VSCROLL, SB_BOTTOM, 0);
    OpenWriteFile(now, path + user[now].sendFileName);//当前服务器
    OpeartionPack pack;
    pack.Operation = 210;
    SendTo(&pack, sizeof(OpeartionPack), user[nowUser].port, user[nowUser].IP, 0);
    Statu = A;
    break;
  }
  //在收到上传文件请求后，发送相应的反馈，并打开文件准备进行写入
  int CUDP::OpenWriteFile(int userID, CString filePath)
{
        return user[userID].file.Open(filePath, CFile::modeCreate | CFile::modeWrite |
}

//在打开文件后，又收到相应文件的一些具体信息，包括组大小和块大小等，并根据此进行相应的
case A:
{
  if (recvPack.Operation == 111)
  {
    dlg->Log += L"收到文件信息，准备接收文件\r\n";
    dlg->UpdateData(FALSE);
    dlg->SendDlgItemMessage(IDC_EDIT_LOG, WM_VSCROLL, SB_BOTTOM, 0);
    nowUser = now;
    GroupNum = fileinfopack->GroupNumber;
    FileGroup = new bool[GroupNum];
    for (int i = 0; i < GroupNum; i++)
      FileGroup[i] = false;
    OpeartionPack pack;
    pack.Operation = 211;//确认文件信息
    SendTo(&pack, sizeof(OpeartionPack), user[nowUser].port, user[nowUser].IP, 0);
    Statu = C;
  }
  break;
}
```

3.6 下载

对于下载而言，其余上传十分相似，都首先需要将相应的信息进行交互之后，然后进行传输，但是与下载不同的是，是服务器要将相应的文件信息发送到客户端，客户端然后对其进行处理，但是发起仍然是由客户端进行的。其首先需要查看文件内容，然后选定，这里已经在3.3中实现，可以具体参看，关于具体文件内容的查看，请参见3.7。

3.6.1 客户端

对于客户端而言，首先需要发起相应的请求，然后根据服务器端的回应作出相应操作

Listing 9 UDPFTPClientDlg.cpp

```
void CUDPFTPClientDlg::OnBnClickedButtonDownload()
{
  // TODO: 在此添加控件通知处理程序代码
  UpdateData(TRUE);
  //MySock.sendPort = HostPort_;
  BYTE nf1, nf2, nf3, nf4;
  IPaddress.GetAddress(nf1, nf2, nf3, nf4);
  MySock.sendIP.Format(L"%u.%u.%u.%u", nf1, nf2, nf3, nf4);
  FileList.GetText(FileList.GetCurSel(), FileName);
  MySock.DownLoadFileName = FileName;
  MySock.DownLoad = true;
  MySock.AsyncSelect(FD_WRITE);
}

if (DownLoad)
      {
            Pack pack;
            pack.Operation = 120;//请求下载文件
            pack.Port = recvPort;
            int position = 0;
            int length = DownLoadFileName.GetLength();
            for (int i = 0; i < length; i++)
                  pack.Content[position++] = DownLoadFileName.GetAt(i);
            pack.Content[position++] = 0;

            //InterPack pack;
            //pack.Operation = 120;
```

```
                //pack.Filename = DownLoadFileName;
                dlg->Log += L"请求下载文件 " + DownLoadFileName + L" 中……\r\n";
                //dlg->UpdateData(FALSE);
                //dlg->SendDlgItemMessage(IDC_EDIT_Log, WM_VSCROLL, SB_BOTTOM, 0);
                SendTo(&pack, sizeof(Pack), sendPort, sendIP, 0);
                CString temp;
                temp.Format(L"the port of present process is %d ... \r\n" ,sendPort);
                dlg->Log += temp;
                dlg->UpdateData(FALSE);
                dlg->SendDlgItemMessage(IDC_EDIT_Log, WM_VSCROLL, SB_BOTTOM, 0);
                SetTimer(hHwnd, -4, 5000, CheckReceive);
                DownLoad = false;
    }
    //在请求下载文件之后，需要对其进行相应的操作，根据服务器返回的信息，接受具体内容，并

    //发来的文件信息
    case 220:
    {
      KillTimer(hHwnd, -4);
      dlg->Log += L"服务器收到请求\r\n";
      dlg->Log += L"收到文件信息，准备接收文件\r\n";
      dlg->UpdateData(FALSE);
      dlg->SendDlgItemMessage(IDC_EDIT_Log, WM_VSCROLL, SB_BOTTOM, 0);
      GroupNum = fileinfopack->GroupNumber;
      FileGroup = new bool[GroupNum];
      for (int i = 0; i < GroupNum; i++)FileGroup[i] = false;
      OpeartionPack pack;
      pack.Operation = 122;//确认文件信息
      SendTo(&pack, sizeof(OpeartionPack), sendPort, sendIP, 0);
      DownLoading = true;
      OpenWriteFile(DownLoadFilePath + DownLoadFileName);
      SetTimer(hHwnd, -221, 5000, CheckReceive);
      break;
    }
```

3.6.2 服务器

在服务器端，首先根据发送到的内容，决定打开哪个文件进行发送，然后执行发送操作（当然，如果无法打开则返回错误信息）

Listing 10 CUDP.cpp

```
  case 120://客户端请求下载文件
  {
    int i = 0;
    char tempStr[1000];
    for (i = 0; recvPack.Content[i] != 0; i++)
      tempStr[i] = recvPack.Content[i];
    tempStr[i] = 0;
    //user[now].downloadFile = interpack->Filename;
    user[now].downloadFile.Format(L"%s", CStringW(tempStr));
    //user[now].downloadFile.Format(L"%s", interpack->Filename);
    out.Format(L"客户端请求下载文件 %s\r\n", user[now].downloadFile);
    //out.Format(L"客户端请求下载文件 %S\r\n", interpack->Filename.GetBuffer(0));
    dlg->Log += head + out;
    dlg->UpdateData(FALSE);
    dlg->SendDlgItemMessage(IDC_EDIT_LOG, WM_VSCROLL, SB_BOTTOM, 0);
    FileInfoPack pack;
    pack.Operation = 220;
    nowUser = now;
    if (OpenReadFile(nowUser,path+ user[now].downloadFile) == 0)
    {
      dlg->Log += L"文件读取失败！请检查文件名是否正确！\r\n";
      dlg->UpdateData(FALSE);
      dlg->SendDlgItemMessage(IDC_EDIT_LOG, WM_VSCROLL, SB_BOTTOM, 0);
      CloseFile(nowUser);
    }
    else
    {
      PktNum = (user[now].file.GetLength() - 1) / 1000 + 1;
      pack.GroupNumber = (PktNum - 1) / 10 + 1;
      GroupNum = pack.GroupNumber;
      SendTo(&pack, sizeof(FileInfoPack), user[nowUser].port, user[nowUser].IP, 0);
      FileGroup = new bool[pack.GroupNumber];
      for (int i = 0; i < pack.GroupNumber; i++)FileGroup[i] = false;
      Statu = B;
    }
    break;
  }

  //首先收到请求之后，打开相应的文件
  int CUDP::OpenReadFile(int userID, CString filePath)
{
```

```
        return user[userID].file.Open(filePath, CFile::modeRead | CFile::typeBinary);

}

void CUDP::CloseFile(int userID)
{
        user[userID].file.Close();
}
//在发送了相应的回复之后，等待客户端的回复，然后准备发送具体的文件信息
case B:
{
  if (recvPack.Operation == 122)
  {
    dlg->Log += L"客户端收到文件信息\r\n";
    CString temp;
    temp.Format(L"正在传输第%d组文件信息...\r\n", NowGroup() + 1);
    dlg->Log += temp;
    dlg->UpdateData(FALSE);
    dlg->SendDlgItemMessage(IDC_EDIT_LOG, WM_VSCROLL, SB_BOTTOM, 0);
    GroupInfoPack pack;
    pack.Operation = 240;//发送文件组信息
    pack.GroupNumber = NowGroup();//下一个组
    if (pack.GroupNumber == GroupNum - 1)//所有分组结束发送
    {
      pack.BlockNumber = PktNum - 10 * pack.GroupNumber;
      nowPktNum = pack.BlockNumber;
      EndFile = true;
    }
    else
    {
      pack.BlockNumber = 10;
      nowPktNum = pack.BlockNumber;
    }
    SendTo(&pack, sizeof(GroupInfoPack), user[nowUser].port, user[nowUser].IP, 0);
    Statu = E;
  }
  break;
}
```

3.7 超时重传

由于不能保证所有的文件均能够正确传输，因此在传输过程中需要设置超时重传，首先理解本协议的文件传输过程。每次文件传输分为组和块，一个组中最多 10 个块，而每个块的大小固定

Listing 11 CUDP.cpp

```
struct Pack
{
  WORD Operation;
  UINT Port;
  BYTE Content[1006];
};
struct InterPack
{
  WORD Operation;
  CString Filename;
};
struct ReSendPack
{
  WORD Operation;
  WORD ReSend[10];
};
struct OpeartionPack
{
  WORD Operation;
  UINT Port;
};
struct FileInfoPack
{
  WORD Operation;
  WORD GroupNumber;
};
struct FilePack
{
  WORD Operation;
  WORD BlockNumber;
  WORD Length;
  BYTE Content[1000];
};
struct GroupInfoPack
{
  WORD Operation;
```

```
    WORD GroupNumber;
    WORD BlockNumber;
};

//首先需要发送组消息，然后根据组消息发送组中所有的块，当一个组发送完成之后，进行检查
//发送组消息并检查
case C:
        {
                switch (recvPack.Operation)
                {
                case 130:
                {
                        CString temp;
                        temp.Format(L"收到第%d组文件信息\r\n", groupinfopack->GroupNum
                        dlg->Log += temp;
                        dlg->UpdateData(FALSE);
                        dlg->SendDlgItemMessage(IDC_EDIT_LOG, WM_VSCROLL, SB_BOTTOM, 0
                        for (int i = 0; i < groupinfopack->BlockNumber; i++)
                                GroupCheck[i] = false;
                        PktNum = groupinfopack->BlockNumber;
                        OpeartionPack pack;
                        pack.Operation = 230;//确认文件组信息
                        SendTo(&pack, sizeof(OpeartionPack), user[nowUser].port, user[
                        Statu = D;
                        break;
                }
                case 199:
                {
                        int i;
                        for (i = 0; i < GroupNum; i++)
                                if (FileGroup[i] == false)
                                        break;
                        CString temp;
                        if (i < GroupNum)
                        {
                                temp.Format(L"文件损坏，建议重新上传\r\n");
                                dlg->Log += temp;
                                dlg->UpdateData(FALSE);
                                dlg->SendDlgItemMessage(IDC_EDIT_LOG, WM_VSCROLL, SB_BO
                        }
                        else
```

```
                        {
                                temp.Format(L"文件上传完成!\r\n");
                                dlg->Log += temp;
                                dlg->UpdateData(FALSE);
                                dlg->SendDlgItemMessage(IDC_EDIT_LOG, WM_VSCROLL, SB_BO
                                EndFile = false;
                        }
                        CloseFile(nowUser);
                        Statu = S;
                        break;
                }
                }
                break;
}

//发送块消息

case D:
        {
                switch (recvPack.Operation)
                {
                case 125:
                {
                        CString temp;
                        temp.Format(L"收到该组第%d块文件\r\n", filepack->BlockNumber +
                        dlg->Log += temp;
                        dlg->UpdateData(FALSE);
                        dlg->SendDlgItemMessage(IDC_EDIT_LOG, WM_VSCROLL, SB_BOTTOM, 0
                        for (int i = 0; i < 1000; i++)
                                temppack[filepack->BlockNumber].Content[i] = filepack-
                        temppack[filepack->BlockNumber].Length = filepack->Length;
                        GroupCheck[filepack->BlockNumber] = true;
                        break;
                }
                case 147:
                case 149:
                {
                        CString temp;
                        temp.Format(L"该组文件发送完毕\r\n");
                        dlg->Log += temp;
                        dlg->UpdateData(FALSE);
```

```
                dlg->SendDlgItemMessage(IDC_EDIT_LOG, WM_VSCROLL, SB_BOTTOM, 0

                for (int i = 0; i < PktNum; i++)
                {
                        if (GroupCheck[i] == false)
                                resend[i] = 1;
                }
                if (CheckResend())
                {
                        CString temp;
                        for (int i = 0; i < 10; i++)
                                if (resend[i] == 1)
                                        temp.Format(L"该组第%d块文件丢失，尝试
                        dlg->Log += temp;
                        dlg->UpdateData(FALSE);
                        dlg->SendDlgItemMessage(IDC_EDIT_LOG, WM_VSCROLL, SB_BO
                        ReSendPack pack;
                        pack.Operation = 245;//该组丢包项
                        memcpy(pack.ReSend, resend, 10 * sizeof(WORD));
                        SendTo(&pack, sizeof(ReSendPack), user[nowUser].port,
                        //Statu = D;
                }
                else
                {
                        CString temp;
                        temp.Format(L"该组文件全部接收\r\n");
                        dlg->Log += temp;
                        dlg->UpdateData(FALSE);
                        dlg->SendDlgItemMessage(IDC_EDIT_LOG, WM_VSCROLL, SB_BO
                        OpeartionPack pack;
                        FileGroup[NowGroup()] = true;
                        pack.Operation = 250;//文件组全部接收
                        SendTo(&pack, sizeof(OpeartionPack), user[nowUser].por
                        for (int i = 0; i < 10; i++)GroupCheck[i] = false;
                        for (int i = 0; i < PktNum; i++)
                                user[nowUser].file.Write(temppack[i].Content,
                        Statu = C;
                        break;
                }
        }
        }
```

```
                break;
        }
```

可以看到，对于每个组消息，其中需要对每个块进行判断，如果有块没有被收到，那么需要将其相应的位置为 false，然后实现重传机制。

重传的块首先需要发送块消息，然后再进行重传，对原来的块进行替换。

同时，也需要对发送的报文是否接受到进行检测，如果没有检测到，同样需要报出相应日志信息和进行重传，其具体实现为：

Listing 12 CUDP.cpp

```
VOID CALLBACK CUDP::CheckReceive(HWND hwnd, UINT uMsg, UINT idEvent, DWORD dwTime)
{
    CUDPFTPClientDlg* dlg = ((CUDPFTPClientDlg*)theApp.GetMainWnd());
  HWND hHwnd = AfxGetMainWnd()->m_hWnd;
   switch (idEvent)
   {
   case -1:
   {
     dlg->Log += L"无法连接到服务器端，连接超时\r\n";
     dlg->UpdateData(FALSE);
     dlg->SendDlgItemMessage(IDC_EDIT_Log, WM_VSCROLL, SB_BOTTOM, 0);
     KillTimer(hHwnd, -1);
     break;
   }
   case -2:
   {
     dlg->Log += L"获取文件列表请求超时，请重新尝试\r\n";
     dlg->UpdateData(FALSE);
     dlg->SendDlgItemMessage(IDC_EDIT_Log, WM_VSCROLL, SB_BOTTOM, 0);
     KillTimer(hHwnd, -2);
     break;
   }
   case -3:
   {
     dlg->Log += L"文件上传请求超时 ,请重新尝试\r\n";
     dlg->UpdateData(FALSE);
     dlg->SendDlgItemMessage(IDC_EDIT_Log, WM_VSCROLL, SB_BOTTOM, 0);
     dlg->MySock.AsyncSelect(FD_WRITE);
     KillTimer(hHwnd, -3);
     break;
   }
   case -4:
```

```
    {
      dlg->Log += L"文件下载请求超时,请重新尝试\r\n";
      dlg->UpdateData(FALSE);
      dlg->SendDlgItemMessage(IDC_EDIT_Log, WM_VSCROLL, SB_BOTTOM, 0);
      KillTimer(hHwnd, -4);
      break;
    }
    case -221:
    {
      dlg->Log += L"确认包可能丢失\r\n";
      dlg->Log += L"重新发送确认信息\r\n";
      dlg->UpdateData(FALSE);
      dlg->SendDlgItemMessage(IDC_EDIT_Log, WM_VSCROLL, SB_BOTTOM, 0);
      OpeartionPack pack;
      pack.Operation = 122;//确认文件信息
      dlg->MySock.SendTo(&pack, sizeof(OpeartionPack), dlg->MySock.sendPort, dlg->MySo
      SetTimer(hHwnd, -221, 5000, CheckReceive);
      break;
    }
    }
}
```

3.8 多用户

对于多用户，实际上，应该为多线程的实现方式，那么应该为每个用户单独开启一个端口，为其进行相应的服务操作，这样才能够保证服务能正常进行。具体实现非常简单，只需要在进行测试连接的时候，开启一个新的端口，并告诉客户端用此端口进行连接即可。具体实现如下：

Listing 13 CUDP.cpp

```
case 100://连接请求
{
  dlg->Log += head + L"客户端请求测试连接\r\n";
  OpeartionPack pack;
  pack.Operation = 200;
  pack.Port = 2055 + nowUser;
  create_port(pack.Port);
  SendTo(&pack, sizeof(OpeartionPack), user[nowUser].port, user[nowUser].IP, 0);
  break;
}
//在这里，要重新创建端口连接
```

```
BOOL CUDP::create_port(UINT port)
{
        CUDPFTPServerDlg* dlg = ((CUDPFTPServerDlg*)theApp.GetMainWnd());
        HWND hHwnd = AfxGetMainWnd()->m_hWnd;
        CUDP* intersocket = new CUDP();
        intersocket->UserNumber = this->UserNumber;
        intersocket->path = this->path;
        intersocket->findWild = this->findWild;
        intersocket->nowUser = this->nowUser;
        intersocket->Statu = this->Statu;
        BOOL bFlag = intersocket->Create(port, SOCK_DGRAM, FD_READ);
        CString temp;
        if (!bFlag)
        {

                temp.Format(L"FTP服务器端口%d 开启失败 \r\n", port);
                dlg->Log += temp;
                dlg->UpdateData(FALSE);
                dlg->SendDlgItemMessage(IDC_EDIT_LOG, WM_VSCROLL, SB_BOTTOM, 0);
                exit(0);
        }
        else
        {
                temp.Format(L"FTP服务器端口%d 开启成功 \r\n", port);
                dlg->Log += temp;
                dlg->UpdateData(FALSE);
                dlg->SendDlgItemMessage(IDC_EDIT_LOG, WM_VSCROLL, SB_BOTTOM, 0);
        }

        intersocket->AsyncSelect(FD_READ);
        return bFlag;
}

//然后在客户端中接收到相应端口
KillTimer(hHwnd, -1);
//recvPort = pack.Port;
sendPort = pack.Port;
CString temp;
temp.Format(L"成功连接到服务器，其端口号为%d \r\n", sendPort);
//dlg->Log += L"成功连接到服务器端！ \r\n";
```

```
dlg->Log += temp;
dlg->UpdateData(FALSE);
dlg->SendDlgItemMessage(IDC_EDIT_Log, WM_VSCROLL, SB_BOTTOM, 0);
break;
}
```

4 实现结果

最终的实现结果可以展示如下

1. 测试端口连接，并分配相应端口，对于客户端而言，可以参见图4-5, 而服务器同样也展示了相应的日志记录，见图4-6

图 4-5 客户端测试连接日志

图 4-6 服务器测试连接并分配端口

2. 刷新列表，根据相应的列表进行查看，对于客户端而言，现在可以看到服务器中所具有的所有内容，见图4-7，而服务器端也会记录相应的访问记录4-8

图 4-7 客户端文件显示内容

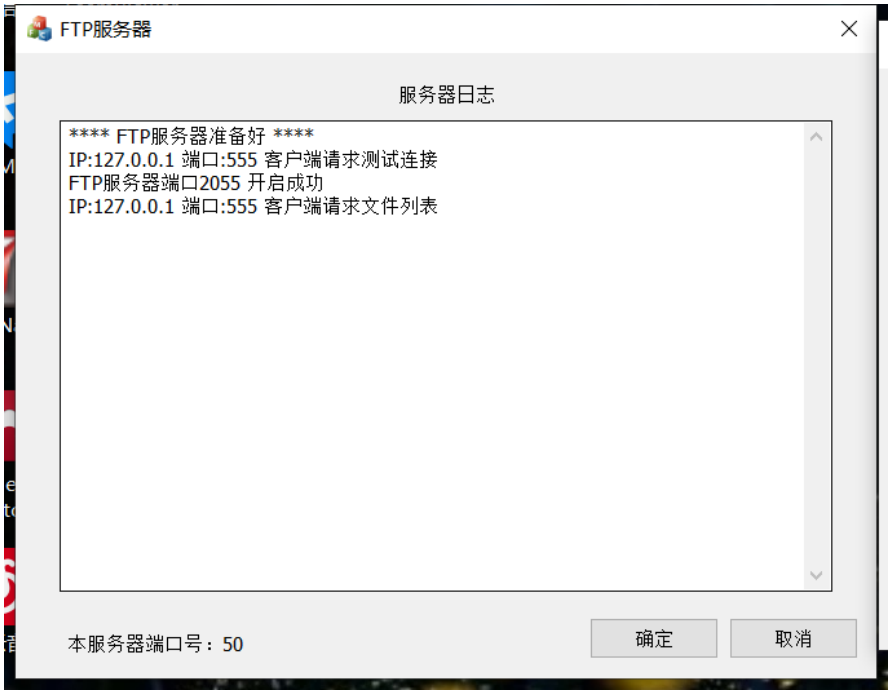

图 4-8 服务器日志记录

3. 浏览文件，在客户端中进行浏览即可，具体可见4-9

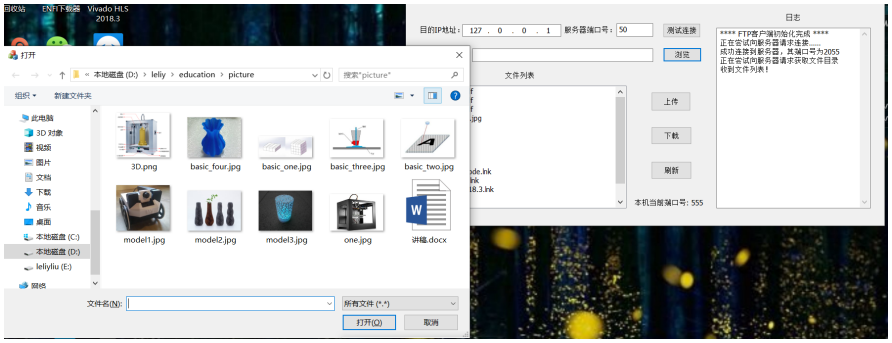

图 4-9 客户端浏览

4. 同时上传文件选择文件进行上传，可以查看日志记录，看到同时上传的内容，如图4-10所示

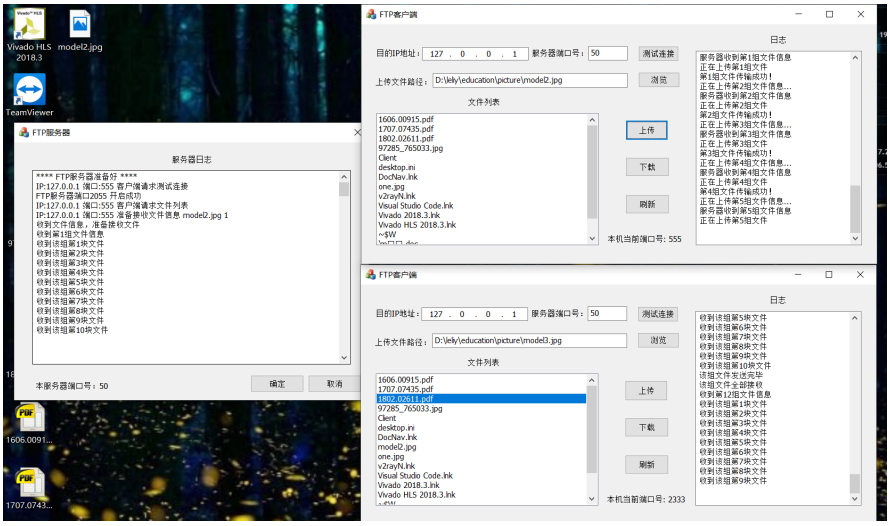

图 4-10 同时上传文件

5. 同时下载文件与同时上传文件类似，还可以同时下载文件，同样的，其实现结果如图4-11

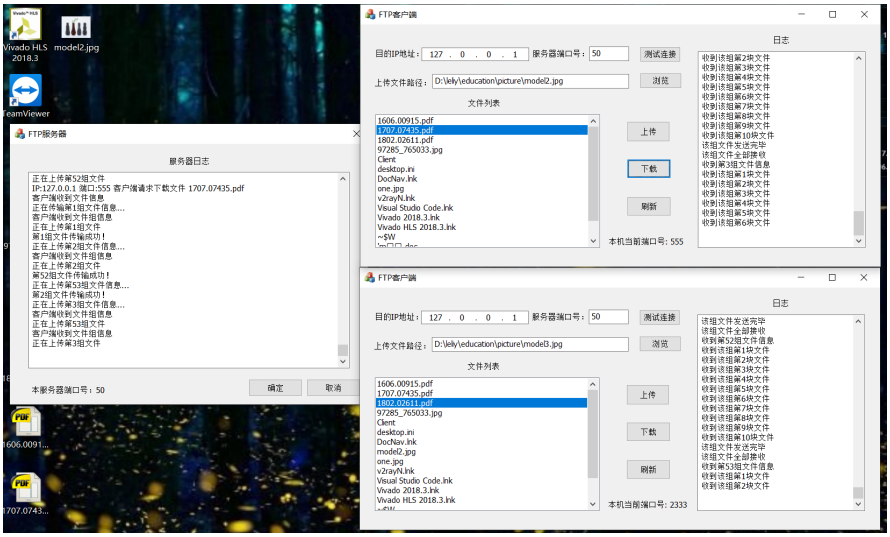

图 4-11 同时下载文件