上机作业三：使用 UDP 实现可靠的文件传输

1711342 李纪

2019 年 12 月 19 日

摘要

这是我的**上机作业**三的实验报告，请老师查阅，谢谢。以下把我设计的协议称为 **RUDP 协议**（**R**eliable **U**ser **D**atagram **P**rotocol）。

本程序的特点有：**支持多用户同时下载或上传文件；支持单客户端同时下载或上传文件**。

关键字：UDP、可靠传输、RUDP

目录

1 程序的基本功能要求

程序的基本功能要求：

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：能可靠下载文件，能同时下载文件。

2 用户手册

2.1 服务器

图 1: 服务器端

服务器界面介绍：

1. File list：文件列表区。展示服务器根目录里的所有文件。
2. Log：服务器日志区。显示服务器日志。
3. Port：选择服务器的监听端口号。
4. Start 按钮：服务器开启按钮。需要在选择文件夹及端口号之后再点击。
5. Open Folder 按钮：选择服务器的根目录。
6. Clear Log 按钮：清空日志区。

2.2 客户端

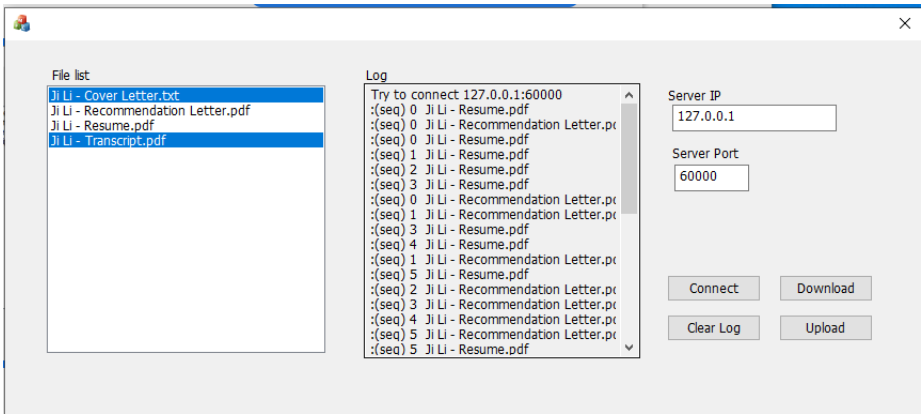

图 2: 客户端

客户端界面介绍：

1. File list：文件列表区。展示服务器根目录里的所有文件。
2. Log：客户端日志区。显示客户端日志。
3. Server IP：输入服务器的 IP 地址。
4. Server Port：输入服务器的端口号。
5. Connect 按钮：连接服务器按钮。在填入 Server IP 和 Server Port 后点击。
6. Download 按钮：下载文件按钮。下载在 File List 中已选中的文件。
7. Upload 按钮：上传文件按钮。上传指定文件到服务器根目录。
8. Clear Log 按钮：清空日志区。

3　RUDP 协议原理介绍

RUDP 属于**停等协议**。**基本思想：**

1. 传输数据时，发送方每发送一次数据后，都必须等待接收方发回的确认数据报被自己收到后，才能发出下一个数据报。
2. 如果发送的数据丢失或发回的确认数据报文丢失，发送方会重发上一次发送的数据。

协议原理图可参考图 3 。

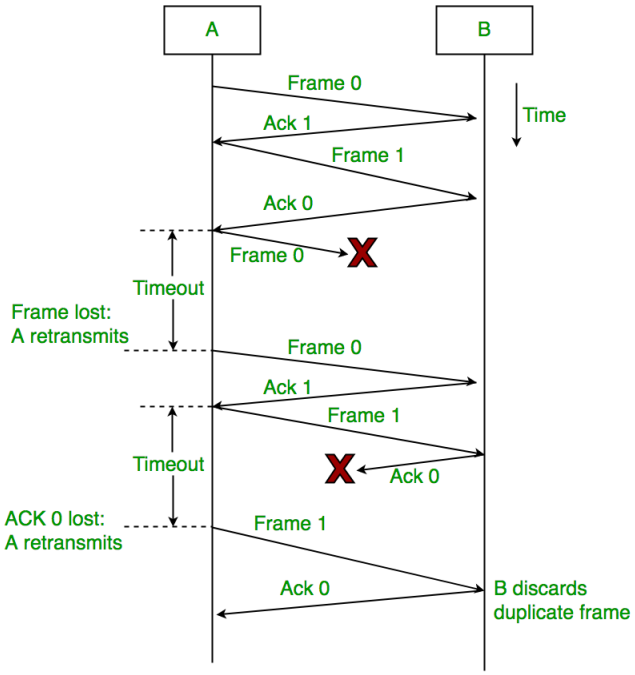

图 3: RUDP 协议原理

4 RUDP 协议实现介绍

本节通过一个下载的样例来分析在 RUDP 协议下客户端和服务器的交互过程。此过程用 Wireshark 抓包的结果如图 4 。RUDP 中各类指令的格式及含义在附录 A 中给出。

4.1 客户端和服务器的初始连接

服务器：首先我们需要通过“Open Folder”按钮选择服务器的根目录（此时把文件名装入一个 deque 容器中），然后在“port”编辑框中写入端口号，点击”start”按钮后服务器正式开启服务。**对应代码：** 5.1.2 。

客户端：输入服务器的 IP 和端口号后，单击“Connect”按钮。此时，客户端向服务器发送了一个 100 请求，请求服务器文件列表。**对应代码：** 5.2.2 。

服务器：服务器收到来自服务器的 100 请求后，发送 200 文件列表应答给客户端。**对应代码：** 5.1.3 。

客户端：客户端收到 200 应答后，使用 200 应答报文中的信息更新自己的文件列表容器，刷新 File list 列表框。客户端和服务器的初始连接过程结束。**对应代码：** 5.2.2 。

4.2 客户端请求下载文件

客户端：用户在 File list 中选中想要获取的文件（可多选），单击“Download”。此时，对于选中的每一个文件，客户端逐一地开启一个新的线程和端口向服务器的原端口发送 110 文件请求，格式为 110+ 请求文件在列表中的序号。**对应代码：** 5.2.3 。

服务器：每收到一个 110 文件请求，服务器逐一地开启一个新的线程和端口来响应。**对应代码：** 5.1.4 。

服务器：在服务器的下载线程中，线程根据 110 请求的文件序号查找文件的路径，再把文件读入文件流中。设置发送数据的报头，在数据报头之后的第一个位置读入数据本身，完成即将发送的数据的组装，发送数据，等待客户端的确认报文。**对应代码：** 5.1.4 。

客户端：客户端收到正确的报文（客户端端口号、服务器端端口号、序列号都相同的报文）后，将接收到的数据写入文件流，然后组装一个仅含有确认信息的数据报，发往服务器；如果没有收到正确的数据报文，客户端重发上一个收到的正确报文对应的确认报文。**对应代码：** 5.2.4 。

服务器：服务器收到正确的确认报文后，继续发送这次应发送数据；如果服务器在发送数据后 0.5 秒内没有收到正确的报文，服务器重发上一次发送的数据；如果服务器线程连续发送了 30 次同样的数据，就认为文件传输完毕但没有收到确认报文，或者认为客户端已死，结束传输。**对应代码：** 5.1.4 。

4.3　客户端上传文件

客户端：单击“Upload”按钮后，用户可以选择一个文件发往服务器的根目录。结束文件的选择之后，客户端开启一个新的线程和端口向服务器的原端口发送 120 上传请求。**对应代码：** 5.2.5 。

服务器：每收到一个 120 上传请求，服务器逐一地开启一个新的线程和端口来响应，并向客户端的对应端口发送 220 上传回复，格式为 220+ 对应生成的服务器端口，意思是回复客户端的上传请求，通知客户端自己将使用 serverPort 端口接收文件。**对应代码：** 5.1.5 。

以下部分的代码和下载部分大同小异，为了精简文档，就不贴出了。请老师打开 Visual Studio 参阅。

客户端：每收到一个 220 上传应答，对应客户端线程逐一地开始向服务器对应端口传输文件。

客户端：在客户端的上传线程中，线程根据 220 应答的服务器端口号传输文件，把文件读入文件流中。设置发送数据的报头，在数据报头之后的第一个位置读入数据本身，完成即将发送的数据的组装，发送数据，等待服务器的确认报文。

服务器：服务器收到正确的报文（客户端端口号、服务器端端口号、序列号都相同的报文）后，将接收到的数据写入文件流，然后组装一个仅含有确认信息的数据报，发往客户端；如果没有收到正确的数据报文，服务器重发上一个收到的正确报文对应的确认报文。

客户端：客户端收到正确的确认报文后，继续发送这次应发送数据；如果客户端在发送数据后 0.5 秒内没有收到正确的报文，客户端重发上一次发送的数据；如果客户端线程连续发送了 30 次同样的数据，就认为文件传输完毕但没有收到确认报文，或者认为服务器已死，结束传输。

4.4　从 Wireshark 中观察程序交互过程

这个交互过程的样例对应了 2 中的界面。一个客户端和一个服务器初始化连接后，客户端请求 4 个文件同时下载，服务器和客户端新开了 4 对全新未使用的端口提供给双方各自的下载线程。

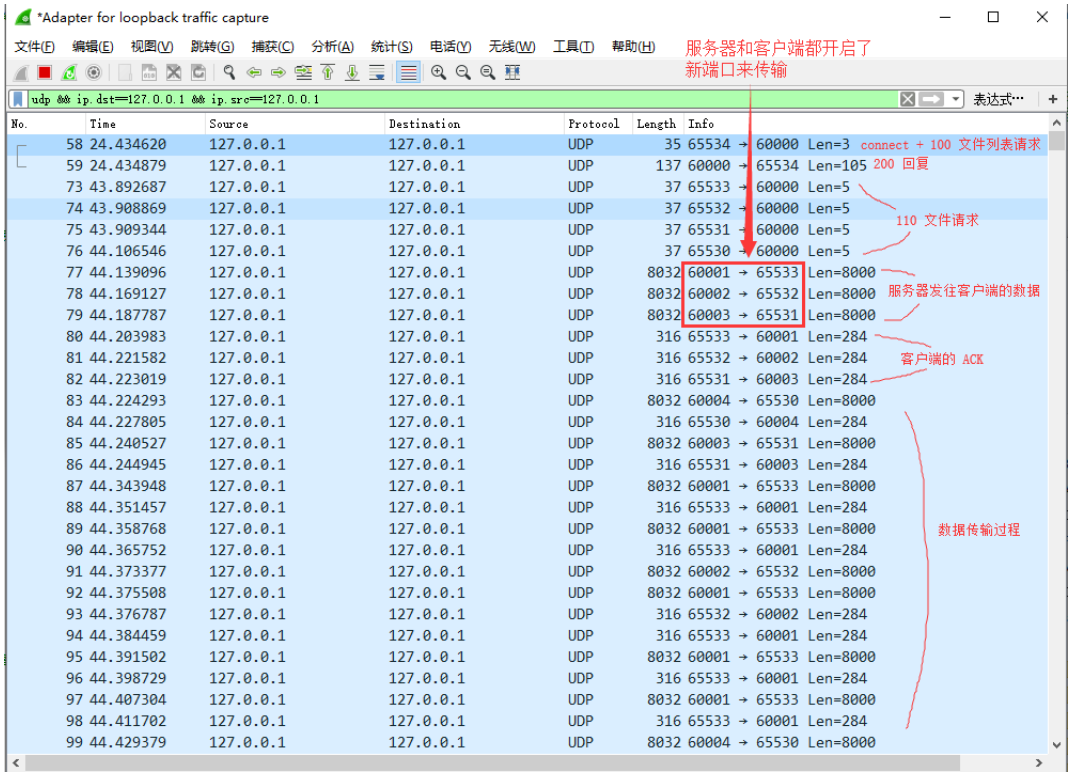

图 4: 从 Wireshark 中观察程序交互过程

5　关键部分源代码

（"..." 部分表示省略）

此部分分为 2 个大节，每个大节有 4 个小节，其中第一个小节展示函数和变量的声明，后三个小节每个小节解释一个具体功能的实现。每小节开头都有**代码摘要**，粗略地展示了功能的实现步骤。

代码解释大部分可以根据代码里的注释来解释说明，请老师查阅。为了方便解释与理解，代码之间的联系与具体实现请直接通过 Visual Studio 查看源代码。

5.1　服务器

5.1.1　头文件（函数和变量的声明）

代码摘要：函数和变量的声明。

因为很多变量都在头文件中声明，所以为了便于理解，首先贴上 Socket 类及对话框类头文件的代码。

```
class CRUDPServerDlg : public CDialogEx
{
...
public:
  void displayString(CEdit& editCtrl, CString& str);
  CString FicowGetDirectory();//选择文件夹，并返回路径
  void ShowFile();//将文件显示在列表框控件中
  void RecursiveFindFile(CString strRootPath, bool isRecursive);//查找文件夹下的所有文件
  // 打开文件夹按钮
  CButton m_folder;
  // 开始按钮
  CButton m_start;
  // 服务器日志编辑框控件
  CEdit m_log;
  // 服务器端口，默认选择60000端口
  int m_port;
  // 文件列表控件
  CListBox m_fileList;
  afx_msg void OnEnChangeEditPort();
  afx_msg void OnBnClickedButtonFolder();
  deque<CString> str_fileDeque;//文件双向队列-deque
  afx_msg void OnBnClickedButtonStart();
  int findNextPort();
  bool nextPortList[5535];//下一个将被分配的端口,从60001开始分配
              //nextPortList[i] == false表示端口60001+i没有被本程序分配
  //deque<ServerSocket>socketDeque;//存储socket的双端队列
  map<int, int>clientToServerMap;//客户端的端口和服务器端口的映射
  ServerSocket usock;// 生成 UDP Socket（用作起始连接）Port=60000
```

```
29    CString strDirectoryPath;//打开的文件目录
30    bool receiveFile(ServerSocket& skt, int clientPort);
31    map<string, int>fileSeq;//记录哪个文件到了哪个seq，用于判断重复seq
32    afx_msg void OnBnClickedButtonClear();
33  };
34
35  #define MAX_PACKET 7000
36
37  struct header {
38    char fileName[260];//Windows下完全限定文件名必须少于260个字符，目录名必须小于248个字符。
39    bool isAck;
40    int seq;
41    int clientPort;
42    int serverPort;
43    int dataLen;
44    int totalLen;
45  };
46
47  struct tempForDownAndUp {//为了开启一个线程随意创建的临时变量
48    CString strData;
49    int clientPort;
50    int initClientPort;
51  };
52
53  class ServerSocket : public CAsyncSocket
54  {
55  public:
56    ServerSocket();
57    virtual ~ServerSocket();
58    virtual void OnSend(int nErrorCode);
59    bool waitACK(ServerSocket& skt, header* hd);
60    CString initReply100();
61    void Reply110(CString str, int clientPort, int serverPort, ServerSocket& skt);
62    void Reply120(CString str, int clientPort, int serverPort, ServerSocket& skt);
63    char* CStringToPChar(CString str);
64    virtual void OnReceive(int nErrorCode);
65    virtual void OnAccept(int nErrorCode);
66    bool processFile(ServerSocket& skt, char* raw, int clientPort);
67    std::map<int, std::vector<int> > clientPortMap;
68  };
```

Listing 1: 头文件代码

5.1.2 功能：服务器初始化

代码摘要：通过“Open Folder”按钮选择服务器的根目录（此时把文件名装入一个 deque 容器中），然后在 “port” 编辑框中写入端口号，点击”start” 按钮后服务器正式开启服务。这里用到了 6 个不同的函数。

使用函数介绍：

1. CString CRUDPServerDlg::FicowGetDirectory()
   - 选择文件夹，并返回路径。
2. void CRUDPServerDlg::RecursiveFindFile(CString strRootPath, bool isRecursive = false)
   - 查找文件夹下的所有文件。
3. void CRUDPServerDlg::ShowFile()
   - 在列表框中展示所有文件。
4. void CRUDPServerDlg::OnBnClickedButtonFolder()
   - 综合上述函数。
5. void CRUDPServerDlg::OnEnChangeEditPort()
   - 实时更新填入的端口号。
6. void CRUDPServerDlg::OnBnClickedButtonStart()
   - 开启服务器的初始端口进行服务。

```
//选择文件夹，并返回路径
CString CRUDPServerDlg::FicowGetDirectory()
{
  BROWSEINFO bi;
  TCHAR name[MAX_PATH];
  ZeroMemory(&bi, sizeof(BROWSEINFO));
  bi.hwndOwner = AfxGetMainWnd()->GetSafeHwnd();
  bi.pszDisplayName = name;
  bi.lpszTitle = _T("选择文件夹目录");
  bi.ulFlags = BIF_RETURNFSANCESTORS;
  LPITEMIDLIST idl = SHBrowseForFolder(&bi);
  if (idl == NULL)
    return _T("");
  CString strDirectoryPath;
  SHGetPathFromIDList(idl, strDirectoryPath.GetBuffer(MAX_PATH));
  strDirectoryPath.ReleaseBuffer();
  if (strDirectoryPath.IsEmpty())
    return _T("");
  if (strDirectoryPath.Right(1) != "\\")
    strDirectoryPath += "\\";
```

```
21
22    return strDirectoryPath;
23  }
24
25  void CRUDPServerDlg::ShowFile()
26  {
27    m_fileList.ResetContent();
28    unsigned sz = str_fileDeque.size();
29    for (unsigned i = 0; i < sz; i++)
30    {
31      m_fileList.InsertString(-1, str_fileDeque[i]);
32    }
33  }
34
35  //查找文件夹下的所有文件
36  void CRUDPServerDlg::RecursiveFindFile(CString strRootPath, bool isRecursive = false)
37  {
38    /*
39       主要是CFileFind类的使用。
40       重要方法;
41       FindFile()
42       FindNextFile()
43    */
44    // strRootPath 为目录名;
45    CFileFind finder;
46    CString FilePath;
47    if (strRootPath.Right(1) != "/")
48      strRootPath += "/";
49    strRootPath += "*.*";
50
51    BOOL res = finder.FindFile(strRootPath);    // 开始遍历root文件夹下有没有文件或文件夹;
52    while (res)       // res为1, 表示仍有nextFile;
53    {
54      res = finder.FindNextFile();
55      FilePath = finder.GetFilePath();
56
57      if (finder.IsDots())  continue;       // 如果文件为“.”或“..”, 则跳过本次循环;
58
59      if (finder.IsDirectory())     // 找到的是文件夹;
60      {
61        if(isRecursive)
62          RecursiveFindFile(FilePath);        // 递归;
63      }
64      else if (!finder.IsDirectory())       // 找到的是文件;
65      {
66        str_fileDeque.push_back((finder.GetFileName()));    // 显示文件名
67      }
68    }
69  }
70
```

```
71
72 void CRUDPServerDlg::OnEnChangeEditPort()
73 {
74   // TODO:  如果该控件是 RICHEDIT 控件，它将不
75   // 发送此通知，除非重写 CDialogEx::OnInitDialog()
76   // 函数并调用 CRichEditCtrl().SetEventMask()，
77   // 同时将 ENM_CHANGE 标志“或”运算到掩码中。
78   // TODO:  在此添加控件通知处理程序代码
79
80   UpdateData(true);
81 }
82
83
84 void CRUDPServerDlg::OnBnClickedButtonFolder()
85 {
86   // TODO: 在此添加控件通知处理程序代码
87
88   str_fileDeque.clear();//每次打开前清空deque
89   strDirectoryPath = FicowGetDirectory();//选择文件夹，并返回路径
90   RecursiveFindFile(strDirectoryPath);
91   ShowFile();
92 }
93
94
95 void CRUDPServerDlg::OnBnClickedButtonStart()
96 {
97   // TODO: 在此添加控件通知处理程序代码
98
99   BOOL createFlag = usock.Create(m_port, SOCK_DGRAM, FD_CLOSE | FD_READ | FD_ACCEPT |
       FD_CONNECT);
100   // 生成失败会报错
101   if (createFlag == 0)
102   {
103     MessageBox(_T("Socket create failed."));
104     usock.Close();
105     return;
106   }
107
108   CString t;
109   t.Format(_T("RUDP Server start at port: %d\r\n"), m_port);
110   // 更新日志
111   displayString(m_log, t);
112
113 }
```

Listing 2: 服务器初始化

5.1.3 功能：服务器回应客户端的初始连接请求（发送 200 应答）

代码摘要：服务器收到来自服务器的 100 请求后，发送 200 文件列表应答给客户端。
使用函数介绍：

1. CString ServerSocket::initReply100()
   - 服务器构建 200 应答报文。
2. void ServerSocket::OnReceive(int nErrorCode)
   - 服务器回复客户端的连接请求。

```
//"200"回复的格式: 200 file/file/file
CString ServerSocket::initReply100() {
  CRUDPServerDlg* pdlg = (CRUDPServerDlg*)AfxGetApp()->GetMainWnd();
  CString s = L"200 ";//回应“100”请求

  for (size_t i = 0; i < pdlg->str_fileDeque.size(); ++i) {
    s += pdlg->str_fileDeque[i];
    s += L"/";//每个文件之间用'/'间隔
  }
  return s;
}
void ServerSocket::OnReceive(int nErrorCode)
{
  // TODO: 在此添加专用代码和/或调用基类
  //CRUDPServerDlg* pdlg = (CRUDPServerDlg*)AfxGetApp()->GetMainWnd();
  CRUDPServerDlg* pdlg = (CRUDPServerDlg*)AfxGetApp()->GetMainWnd();
  // A buffer for the incoming data.
  char m_szBuffer[4096];
  // What ReceiveFrom Returns.
  int nRead;
  CString host, strp;
  UINT port;
  // 获取接收信息
  nRead = ReceiveFrom(m_szBuffer, sizeof(m_szBuffer), host, port, 0);

  ...

    if (strTextOut.Left(3) == "100") {//收到客户端的"100"请求
      CString s = initReply100();//获取文件列表, 构建即将发送的数据
      char retchar[4096];

      int len = WideCharToMultiByte(CP_ACP, 0, s, -1, NULL, 0, NULL, NULL);
      WideCharToMultiByte(CP_ACP, 0, s, -1, retchar, len, NULL, NULL);
      clientPortMap[port];//插入空的键

```

```
36        pdlg->usock.SendTo(retchar, strlen(retchar), port, _T("127.0.0.1"), 0);
37      }
38      ...
39
40    CAsyncSocket::OnReceive(nErrorCode);
41  }
```

Listing 3: 服务器回应客户端的初始连接请求

5.1.4 功能：服务器回应客户端的下载请求

代码摘要：服务器收到来自服务器的 100 请求后，发送 200 文件列表应答给客户端。
使用函数介绍：

1. void ServerSocket::OnReceive(int nErrorCode)
   - 服务器接收客户端的 110 请求。
   - 每收到一个 110 文件请求，服务器逐一地开启一个新的线程和端口来响应。
2. void ServerSocket::Reply110(CString s, int clientPort, int serverPort, ServerSocket& skt)
   - 线程根据 110 请求的文件序号查找文件的路径，再把文件读入文件流中。设置发送数据的报头，在数据报头之后的第一个位置读入数据本身，完成即将发送的数据的组装，发送数据，等待客户端的确认报文。
3. bool ServerSocket::waitACK(ServerSocket& skt,header* hd)
   - 等待客户端的确认报文。

```
1  void ServerSocket::OnReceive(int nErrorCode){
2    // TODO: 在此添加专用代码和/或调用基类
3    //CRUDPServerDlg* pdlg = (CRUDPServerDlg*)AfxGetApp()->GetMainWnd();
4    CRUDPServerDlg* pdlg = (CRUDPServerDlg*)AfxGetApp()->GetMainWnd();
5    // A buffer for the incoming data.
6    char m_szBuffer[4096];
7    // What ReceiveFrom Returns.
8    int nRead;
9    CString host, strp;
10   UINT port;
11   // 获取接收信息
12   nRead = ReceiveFrom(m_szBuffer, sizeof(m_szBuffer), host, port, 0);
13
14   ...
15     else if (strTextOut.Left(3) == "110") {//收到客户端的"110"请求
```

```
16        tempForDownAndUp* t = new tempForDownAndUp;
17        t->clientPort = port;
18        t->strData = strTextOut;
19
20        // 调用 AfxBeginThread() 函数启动工作者线程
21        // AfxBeginThread() 函数将返回一个指向新创建线程对象的指针
22        //CRUDPClientDlg* p = (CRUDPClientDlg*)AfxGetMainWnd();
23        CWinThread* m_Downloader = AfxBeginThread(
24          (AFX_THREADPROC)Downloader, // pfnThreadProc: 指向工作者线程的控制函数, 它的值不能为
      NULL
25          t,
26          THREAD_PRIORITY_NORMAL // 用于指定线程的优先级
27        );
28        if (m_Downloader != nullptr) {
29        }
30        else {
31          AfxMessageBox(L"Download Thread Create Failed.");
32        }
33      }
34      ...
35
36    CAsyncSocket::OnReceive(nErrorCode);
37  }
38  void ServerSocket::Reply110(CString s, int clientPort, int serverPort, ServerSocket& skt) {
39    //获取客户端想要接收的文件的序号
40    int pos = s.Find(L" ");
41    CString fileNumStr = s.Mid(pos + 1);
42    int fileNum = _ttoi(fileNumStr);
43
44    //设置文件路径和文件流
45    CRUDPServerDlg* pdlg = (CRUDPServerDlg*)AfxGetApp()->GetMainWnd();
46    CString file = pdlg->str_fileDeque[fileNum];
47    CString filePath = pdlg->strDirectoryPath + file;
48    ifstream in(filePath, ifstream::ate | ifstream::binary);
49    int fileSize=in.tellg();//获取文件大小
50
51    in.seekg(0);//设置文件指针, 指向文件开头
52
53    int splice = int(fileSize / MAX_PACKET) + 1;//这个文件会被分成多少片。
54    char send[8000];//声明要发送的数组
55    header* hd = new struct header;//初始化header
56    hd->clientPort = clientPort;
57    hd->serverPort = serverPort;
58    /*int nameStart = filePath.ReverseFind(_T('\\')) + 1;
59    for (int i = nameStart, j = 0; i < filePath.GetLength(); i++,j++) {
60      hd->fileName[j] = filePath[i];
61    }
62    hd->fileName[filePath.GetLength() - nameStart] = '\0';*/
63
64    //给hd->fileName赋值, 支持多国语言文件名
```

```
65  int ns = filePath.ReverseFind(_T('\\')) + 1;
66  int nameLen = filePath.GetLength() - ns;
67  CString name = filePath.Right(nameLen);
68  //CString cs("Hello");
69  // Convert a TCHAR string to a LPCSTR
70  CT2CA pszConvertedAnsiString(name);
71  // construct a std::string using the LPCSTR input
72  std::string strFileName(pszConvertedAnsiString);
73  //string s = "hello world";
74  const char* p = strFileName.c_str();
75  memcpy(&hd->fileName[0], p, strlen(p));
76  hd->fileName[strlen(p)] = '\0';
77
78  vector<char> buffer(MAX_PACKET, 0); //声明读取文件时用到的vector容器，且内容全置为0
79
80  int seq = 0;//初始化发送序列号
81  int retransNum = 0;// 重传次数，下面会用到，重传次数>=30时，认为文件传输完毕但没有收到ack
      ，或者认为客户端已死，结束传输。
82  while (!in.eof()) {
83    int restData= fileSize - seq * buffer.capacity();//记录还有多少数据等待发送
84    int curPos = in.tellg();//保存当前文件读取位置，便于重传
85
86    hd->seq = seq;//设置头部序列号
87    hd->isAck = false;//表明这个包不是ack包
88    if (restData < 0)//剩余数据为负，代表传输完成，结束传输
89      break;
90    if (restData < buffer.capacity()) {//剩余数据量小于buffer的容量，只读取和剩余数据量等同
      大小的块
91      in.read(buffer.data(), restData); // vector::data -> A pointer to the first element in
       the array used internally by the vector.
92                        //读取指定大小的文件进入vector容器，从vector容器中的第一个元素的位置开
      始写入
93      hd->dataLen = restData;//设置此数据报中数据段的长度
94      in.setf(std::ios::eofbit);
95    }
96    else {
97      in.read(buffer.data(), buffer.capacity());
98      hd->dataLen = buffer.capacity();//设置此数据报中数据段的长度
99    }
100   hd->totalLen = hd->dataLen + sizeof(struct header);//设置数据报有效部分总长度 = 数据长度
       + 报头长度
101
102   //准备要送出去的东西
103   char* cbuffer = &buffer[0];//声明一个指向buffer开始地址的指针
104   memcpy(send, hd, sizeof(struct header));//send数组的开始部分设置为数据报头部
105   char* t = &send[sizeof(struct header)];//设置t指向send数组中紧连着头部的第一个地址
106   memcpy(t, cbuffer, buffer.size());//将buffer存入send中头部之后的位置
107   int se=skt.SendTo(send, 8000, clientPort, _T("127.0.0.1"), 0);//发送send到客户端的相应通
      信端口
108
```

```
109     //ACK处理
110     clock_t begin = clock();//记录当前时间（开始等待ACK）
111     double elapsed_secs = 0;//elapsed_secs用于记录等了ACK多少时间了
112     while (1) {//进入死循环，等待ACK
113       bool crecv = waitACK(skt,hd);//传入服务器socket和头部信息
114       if (!crecv) {//如果没有收到ACK
115         crecv = waitACK(skt,hd);//继续等ACK
116         clock_t end = clock();
117         elapsed_secs = double(end - begin) / CLOCKS_PER_SEC;//过了0.2秒就重传，过了30次未收
      到ACK就终止传输
118         if (retransNum >= 30) {//重传次数>=30时，认为文件传输完毕但没有收到ack，或者认为客户
      端已死，结束传输。
119           break;
120         }
121         if (elapsed_secs >= 0.2) {
122           in.seekg(curPos);//重置文件指针至之前读取位置
123           retransNum++;//重传次数+1
124           break;
125         }
126       }
127       else {//收到ACK
128         seq++;//序列号+1
129         retransNum = 0;//重置重传次数
130         CString file(hd->fileName);//显示日志记录
131         CString out;
132         out.Format(L":(ACK) %d  \n", hd->seq);
133         out = L"\r\n" + out;
134         out = out + file;
135         pdlg->displayString(pdlg->m_log, out);
136         break;
137       }
138     }
139     if (retransNum >= 30) {//重传次数>=30时，认为文件传输完毕但没有收到ack，或者认为客户端已
      死，结束传输。
140       CString file(hd->fileName);//显示日志记录
141       CString out;
142       out.Format(L"End transfer:\n");
143       out = L"\r\n" + out;
144       out = out + file;
145       pdlg->displayString(pdlg->m_log, out);
146       break;
147     }
148   }
149 }
150 bool ServerSocket::waitACK(ServerSocket &skt,header* hd){
151   CRUDPServerDlg* pdlg = (CRUDPServerDlg*)AfxGetApp()->GetMainWnd();
152   // A buffer for the incoming data.
153   char m_szBuffer[4096];
154   // What ReceiveFrom Returns.
155   int nRead;
```

```
156   CString host, strp;
157   UINT port;
158   // 获取接收信息
159   //Sleep(15);
160   nRead = skt.ReceiveFrom(m_szBuffer, sizeof(m_szBuffer), host, port, 0);
161
162   // 按 ReceiveFrom 的返回值来判断如何继续
163   CString out; int ierr;
164   switch (nRead) {
165   case 0:
166     // this->Close();
167     break;
168   case SOCKET_ERROR:
169     ierr = GetLastError();
170     if (ierr== WSAEWOULDBLOCK) {  // currently no data available
171       Sleep(5);  // wait and try again
172       break;
173     }
174     //out.Format(L"error: %d", ierr);
175     //AfxMessageBox(out);
176     //skt.Close();
177   default:
178     header* recvhd = reinterpret_cast<header*>(m_szBuffer);
179     if (recvhd->isAck) {
180       if (hd->clientPort == recvhd->clientPort && hd->serverPort == recvhd->serverPort && hd
      ->seq == recvhd->seq) {
181         return true;//ACK的相关信息和当前头部信息对应的话，返回true。传输下一个数据报
182       }
183     }
184   }
185   return false;
186 }
```

Listing 4: 服务器回应客户端的 110 请求

5.1.5 功能：服务器回应客户端的上传请求（发送 220 应答）

代码摘要：每收到一个 120 上传请求，服务器逐一地开启一个新的线程和端口来响应，并向客户端的对应端口发送 220 上传回复，格式为 220+ 对应生成的服务器端口，意思是回复客户端的上传请求，通知客户端自己将使用 serverPort 端口接收文件。

使用函数介绍：

1. void ServerSocket::OnReceive(int nErrorCode)

   - 服务器接收客户端的 120 请求。
   - 每收到一个 120 文件请求，服务器逐一地开启一个新的线程和端口来响应。

2. UINT Uploader(tempForDownAndUp* t)
   - 线程根据 120 请求的服务器端口来创建自己的端口，并将自己新创建的端口通知客户端。
3. void ServerSocket::Reply120(CString str, int clientPort, int serverPort, ServerSocket& skt)
   - 接收上传报文，并在上传结束时更新文件列表，向客户端发送新的文件列表。

```
void ServerSocket::OnReceive(int nErrorCode)
{
  // TODO: 在此添加专用代码和/或调用基类
  //CRUDPServerDlg* pdlg = (CRUDPServerDlg*)AfxGetApp()->GetMainWnd();
  CRUDPServerDlg* pdlg = (CRUDPServerDlg*)AfxGetApp()->GetMainWnd();
  // A buffer for the incoming data.
  char m_szBuffer[4096];
  // What ReceiveFrom Returns.
  int nRead;
  CString host, strp;
  UINT port;
  // 获取接收信息
  nRead = ReceiveFrom(m_szBuffer, sizeof(m_szBuffer), host, port, 0);

  ...
    else if (strTextOut.Left(3) == "120") {
      tempForDownAndUp* t = new tempForDownAndUp;
      t->clientPort = port;
      t->strData = strTextOut;
      t->initClientPort = port;
      CWinThread* m_Uploader = AfxBeginThread(
        (AFX_THREADPROC)Uploader, // pfnThreadProc: 指向工作者线程的控制函数，它的值不能为
    NULL
        t,
        THREAD_PRIORITY_NORMAL // 用于指定线程的优先级
      );
      if (m_Uploader != nullptr) {
      }
      else {
        AfxMessageBox(L"Upload Thread Create Failed.");
      }
    }
  }

  CAsyncSocket::OnReceive(nErrorCode);
}
UINT Uploader(tempForDownAndUp* t) // 上传线程的控制函数
{
```

```
38    CRUDPServerDlg* pdlg = (CRUDPServerDlg*)AfxGetApp()->GetMainWnd();
39    int serverPort = pdlg->findNextPort() + 60001;//生成新的socket，占用一个新的端口，用于回应
        客户端的上传指令
40    ServerSocket skt;
41    BOOL Flag = skt.Create(serverPort, SOCK_DGRAM, FD_CLOSE | FD_READ | FD_ACCEPT | FD_CONNECT
        );
42    while (!Flag) {//端口被占用的话就遍历端口数组
43      serverPort = pdlg->findNextPort() + 60001;
44      Flag = skt.Create(serverPort, SOCK_DGRAM, FD_CLOSE | FD_READ | FD_ACCEPT | FD_CONNECT);
45    }
46    skt.clientPortMap[t->initClientPort].push_back(t->clientPort);
47    skt.Reply120(t->strData, t->clientPort, serverPort, skt);
48    return 0;
49  }
50  void ServerSocket::Reply120(CString str, int clientPort, int serverPort, ServerSocket& skt)
51  {
52    CRUDPServerDlg* pdlg = (CRUDPServerDlg*)AfxGetApp()->GetMainWnd();
53    char retchar[4096];
54    CString s;
55    s.Format(L"220 %d\n", serverPort);
56    int len = WideCharToMultiByte(CP_ACP, 0, s, -1, NULL, 0, NULL, NULL);
57    WideCharToMultiByte(CP_ACP, 0, s, -1, retchar, len, NULL, NULL);
58    skt.SendTo(retchar, strlen(retchar), clientPort, _T("127.0.0.1"), 0);//通知客户端，服务器
        已经为上传指令开启了新的端口
59
60    while (1) {
61      skt.Listen();
62      bool done = pdlg->receiveFile(skt, serverPort);
63      if (done) {//重新构建文件列表，发送给客户端。
64        pdlg->str_fileDeque.clear();
65        pdlg->RecursiveFindFile(pdlg->strDirectoryPath, 0);
66        pdlg->ShowFile();
67        CString s = skt.initReply100();
68        char retchar[4096];
69
70        int len = WideCharToMultiByte(CP_ACP, 0, s, -1, NULL, 0, NULL, NULL);
71        WideCharToMultiByte(CP_ACP, 0, s, -1, retchar, len, NULL, NULL);
72
73        map<int, vector<int>>::iterator it = skt.clientPortMap.begin();
74        bool find = false;
75        for (it = skt.clientPortMap.begin(); it != skt.clientPortMap.end();) {
76          for (int i = 0; i<it->second.size(); i++) {
77            if (it->second[i] == clientPort) {
78              find = true;
79              break;
80            }
81            it++;
82          }
83          if (find)
84            break;
```

```
85        }
86          //;//std::cout << it->first << " => " << it->second << '\n';
87        pdlg->usock.SendTo(retchar, strlen(retchar), it->first, _T("127.0.0.1"), 0);
88        break;
89      }
90    }
91  }
```

Listing 5: 服务器回应客户端的上传请求

5.2 客户端

5.2.1 头文件（函数和变量的声明）

代码摘要：函数和变量的声明。

因为很多变量都在头文件中声明，所以为了便于理解，首先贴上 Socket 类及对话框类头文件的代码。

```
1  // CRUDPClientDlg 对话框
2  class CRUDPClientDlg : public CDialogEx
3  {
4  ...
5  public:
6    void displayString(CEdit& editCtrl, CString& str);
7    void ShowFile();
8    // 连接按钮
9    CButton m_connect;
10   // 下载按钮
11   CButton m_download;
12   // 上传按钮
13   CButton m_upload;
14   // 服务器的IP地址 默认为127.0.0.1
15   CString m_ip;
16   // 客户端日志
17   CEdit m_log;
18   // 服务器端口号 默认为60000
19   int m_port;
20   // 文件列表控件
21   CListBox m_fileList;
22   deque<CString> str_fileDeque;//文件双向队列-deque
23   bool nextPortList[5535];//下一个将被分配的端口,从65535开始分配
24               //nextPortList[i] == false表示端口60001+i没有被本程序分配
25   afx_msg void OnBnClickedButtonConnect();
26   bool receiveFile(ClientSocket& skt, int clientPort);
27   afx_msg void OnBnClickedButtonDownload();
28   afx_msg void OnBnClickedButtonUpload();
29   int findNextPort();//找到下一个将被分配的端口
```

```
30   ClientSocket usock;// 生成 UDP Socket (用作起始连接) Port=65535
31   bool curFile[10];//表示当前选中了第几个文件
32   int fileWait;//有多少个文件还没有开始下载
33   map<string, int>fileSeq;//记录哪个文件到了哪个seq, 用于判断重复seq
34   //deque<CWinThread*>threadDeque;//线程地址队列
35
36   afx_msg void OnSelchangeListFile();
37   afx_msg void OnBnClickedButtonClear();
38 };
39
40 #define MAX_PACKET 7000
41 // ClientSocket 命令目标
42 struct header {
43   char fileName[260];//Windows下完全限定文件名必须少于260个字符, 目录名必须小于248个字符。
44   bool isAck;
45   int seq;
46   int clientPort;
47   int serverPort;
48   int dataLen;
49   int totalLen;
50 };
51
52 struct tempUpload {
53   CString filePath;
54   //int initClientPort;
55 };
56
57 class ClientSocket : public CAsyncSocket
58 {
59 public:
60   ClientSocket();
61   virtual ~ClientSocket();
62   void initReceive200(CString s);
63   int waitFor220(ClientSocket& skt, int clientPort);
64   bool uploadFile(CString filePath, ClientSocket& skt, int clientPort, int serverPort);
65   bool waitACK(ClientSocket& skt, header* hd);
66   virtual void OnReceive(int nErrorCode);
67   bool processFile(ClientSocket& skt, char* raw,int clientPort);
68 };
```

Listing 6: 头文件代码

5.2.2 功能：客户端连接服务器（发送 100 请求）

代码摘要：输入服务器的 IP 和端口号后，单击 “Connect” 按钮。此时，客户端向服务器发送了一个 100 请求，请求服务器文件列表。这里主要用到了 2 个函数。

使用函数介绍：

1. void CRUDPClientDlg::OnBnClickedButtonConnect()

- 客户端开启自己的初始端口。
- 向服务器发送 100 文件列表请求。

2. void ClientSocket::initReceive200(CString s)

- 客户端收到 200 应答后，使用 200 应答报文中的信息更新自己的文件列表容器，刷新 File list 列表框。客户端和服务器的初始连接结束。

```
void CRUDPClientDlg::OnBnClickedButtonConnect()
{
  // TODO: 在此添加控件通知处理程序代码

  //int port = findNextPort();

  int port = findNextPort() + 60000;
  //ClientSocket skt;
  BOOL Flag = usock.Create(port, SOCK_DGRAM, FD_READ | FD_WRITE | FD_OOB | FD_ACCEPT |
     FD_CONNECT | FD_CLOSE);
  while (!Flag) {//端口被占用的话就遍历端口数组
    port = findNextPort() + 60000;
    Flag = usock.Create(port, SOCK_DGRAM, FD_READ | FD_WRITE | FD_OOB | FD_ACCEPT |
     FD_CONNECT | FD_CLOSE);
  }

  Flag = usock.Connect(m_ip, m_port);
  CString t;
  t.Format(_T("Try to connect %s:%d"), m_ip, m_port);

  // 更新日志
  displayString(m_log, t);

  //请求文件列表
  char retchar[4096];
  CString s = L"100";//"100"请求的格式: 100

  int len = WideCharToMultiByte(CP_ACP, 0, s, -1, NULL, 0, NULL, NULL);
  WideCharToMultiByte(CP_ACP, 0, s, -1, retchar, len, NULL, NULL);

  usock.SendTo(retchar, strlen(retchar), 60000, _T("127.0.0.1"), 0);
}

//处理初始化数据报 //"200"回复的格式: 200 file/file/file
void ClientSocket::initReceive200(CString s) {
  int curPos;
  int startPos = s.Find(' ');//找到文件列表开始的位置。
  curPos = startPos;
  CRUDPClientDlg* pdlg = (CRUDPClientDlg*)AfxGetApp()->GetMainWnd();
```

```
38
39   while (curPos < s.GetLength()) {
40     int nextFilePos = s.Find(_T("/"), curPos+1);
41     if (nextFilePos < 0)
42       break;
43     int nCount = nextFilePos - curPos - 1;
44     CString temp = s.Mid(curPos + 1, nCount);//截取文件名
45     pdlg->str_fileDeque.push_back(temp);//将得到的数据填充进文件列表双端队列str_fileDeque
46     curPos = nextFilePos;
47   }
48
49   pdlg->ShowFile();//在列表框控件m_fileList中展示
50 }
```

Listing 7: 客户端连接服务器

5.2.3 功能：客户端请求下载文件（发送 110 请求）

代码摘要：用户在 File list 中选中想要获取的文件（可多选），单击 “Download”。此时，对于选中的每一个文件，客户端逐一地开启一个新的线程和端口向服务器的原端口发送 110 文件请求，格式为 110+ 请求文件在列表中的序号。

使用函数介绍：

1. void CRUDPClientDlg::OnSelchangeListFile()
   - 客户端生成请求文件列表。
2. void CRUDPClientDlg::OnBnClickedButtonDownload()
   - 客户端开启下载线程。
3. UINT Downloader(PVOID hWnd)
   - 客户端下载线程。正式向服务器发送 110 请求报文。

```
1  void CRUDPClientDlg::OnSelchangeListFile()
2  {
3    // TODO: 在此添加控件通知处理程序代码
4
5    int temp = m_fileList.GetCurSel(); // 获取listbox被选中的行的数目
6    for (int i = 0; i < 10; ++i) {
7      if (i == temp && curFile[i] == false) {//该文件之前未被选中，将文件序号添加进选择文件列
       表
8        curFile[i] = true;
9        fileWait++;
10       break;
```

```
    }
    else if (i == temp && curFile[i] == true) {//该文件之前被选中，将文件序号从选择文件列表中删除
      curFile[i] = false;
      fileWait--;
      break;
    }
  }
}
void CRUDPClientDlg::OnBnClickedButtonDownload()
{
  // TODO: 在此添加控件通知处理程序代码
  m_fileList.SetSel(-1, FALSE);//取消选中所有条目。deselect all items.

  while (fileWait != 0) {
    // 调用 AfxBeginThread() 函数启动工作者线程
    // AfxBeginThread() 函数将返回一个指向新创建线程对象的指针
    //CRUDPClientDlg* p = (CRUDPClientDlg*)AfxGetMainWnd();
    CWinThread* m_Uploader = AfxBeginThread(
      (AFX_THREADPROC)Downloader, // pfnThreadProc：指向工作者线程的控制函数，它的值不能为NULL
      NULL, //
      THREAD_PRIORITY_NORMAL // 用于指定线程的优先级
    );
    if (m_Uploader != nullptr) {
      //threadDeque.push_back(m_Downloader);
      fileWait--;
    }
    else {
      AfxMessageBox(L"Download Thread Create Failed.");
    }
  }
}
UINT Downloader(PVOID hWnd) // 下载线程的控制函数
{
  CRUDPClientDlg* pdlg = (CRUDPClientDlg*)AfxGetApp()->GetMainWnd();
  if (pdlg == nullptr)
  {
    AfxMessageBox(L"sad");
  }
  char retchar[4096];
  CString s;
  s.Format(L"110");// "110"指令的格式：110 filenum///////filenum filenum
  for (int i = 0; i < 10; ++i) {
    if (pdlg->curFile[i]) {
      CString t;
      t.Format(L" %d", i);
      s.Append(t);
      pdlg->curFile[i] = false;//这个文件有线程对应了，所以重置它在选择文件列表中的值
      break;
```

```
    }
  }

  int len = WideCharToMultiByte(CP_ACP, 0, s, -1, NULL, 0, NULL, NULL);
  WideCharToMultiByte(CP_ACP, 0, s, -1, retchar, len, NULL, NULL);

  int port = pdlg->findNextPort() + 60001;
  ClientSocket skt;
  BOOL Flag = skt.Create(port, SOCK_DGRAM, FD_READ | FD_WRITE | FD_OOB | FD_ACCEPT |
    FD_CONNECT | FD_CLOSE);
  while (!Flag) {//端口被占用的话就遍历端口数组
    port = pdlg->findNextPort() + 60001;
    Flag = skt.Create(port, SOCK_DGRAM, FD_READ | FD_WRITE | FD_OOB | FD_ACCEPT | FD_CONNECT
      | FD_CLOSE);
  }

  skt.SendTo(retchar, strlen(retchar), 60000, _T("127.0.0.1"), 0);

  while (1) {
    skt.Listen();
    bool done = pdlg->receiveFile(skt, port);
    if (done)
      break;
  }
  return 0;
}
```

Listing 8: 客户端请求下载文件

5.2.4 功能：客户端接收下载的数据

代码摘要：客户端收到正确的报文（客户端端口号、服务器端端口号、序列号都相同的报文）后，将接收到的数据写入文件流，然后组装一个仅含有确认信息的数据报，发往服务器；如果没有收到正确的数据报文，客户端重发上一个收到的正确报文对应的确认报文。

使用函数介绍：

1. bool CRUDPClientDlg::receiveFile(ClientSocket& skt, int clientPort)
   - 客户端接收下载数据报。
2. bool ClientSocket::processFile(ClientSocket& skt, char* raw, int clientPort)
   - 客户端处理下载数据报，并视情况发送确认报文。

```
//用于接收收到的文件分片
bool CRUDPClientDlg::receiveFile(ClientSocket& skt, int clientPort) {
```

```
  CRUDPClientDlg* pdlg = (CRUDPClientDlg*)AfxGetApp()->GetMainWnd();
  // A buffer for the incoming data.
  char m_szBuffer[8000];
  // What ReceiveFrom Returns.
  int nRead;
  CString host, strp;
  UINT port;
  // 获取接收信息
  //Sleep(5);
  nRead = skt.ReceiveFrom(m_szBuffer, sizeof(m_szBuffer), host, port, 0);
  // 按 ReceiveFrom 的返回值来判断如何继续
  CString out; int ierr;
  switch (nRead) {
  case 0:
    break;
  case SOCKET_ERROR:
    ierr = GetLastError();
    if (ierr == WSAEWOULDBLOCK) {  // currently no data available
      Sleep(5);  // wait and try again
      break;
    }
    out.Format(L"error: %d", ierr);
    AfxMessageBox(out);
    skt.Close();
  default:
    CString strTextOut = (CString)m_szBuffer;

    if (strTextOut.Left(3) == "200") {
      skt.initReceive200(strTextOut);
    }
    else {//收到文件报文的话
      header* recvhd = reinterpret_cast<header*>(m_szBuffer);
      string str(recvhd->fileName);
      if (fileSeq.find(str) == fileSeq.end()) {//如果文件名不在map容器里，说明是新的文件，把它插入容器中
        fileSeq[str] = int(-1);
      }
      bool done = skt.processFile(skt, m_szBuffer, clientPort);//处理报文
      return done;
    }
  }
  return false;
}
//处理文件分片
bool ClientSocket::processFile(ClientSocket& skt, char* raw, int clientPort) {
  //CRUDPClientDlg* pdlg = (CRUDPClientDlg*)AfxGetApp()->GetMainWnd();
  CRUDPClientDlg* pdlg = (CRUDPClientDlg*)AfxGetApp()->GetMainWnd();
  header* recvhd = reinterpret_cast<header*>(raw);

  string str(recvhd->fileName);
```

```
52  map<string, int>::iterator it = pdlg->fileSeq.find(str);//在map容器中找到文件
53
54
55  char* temp = &raw[sizeof(struct header)];//创建一个指向数据段起始地址的指针
56  ofstream myfile;
57  myfile.open(recvhd->fileName, ios::out | ios::app | ios::binary);
58
59  if (recvhd->seq == it->second + 1) {//如果收到的是期待收到的报文，写入文件。
60                  //it->second指示的是文件中已经有了第几号报文，期待的下一个报文就是it->
      second+1
61    myfile.write(temp, recvhd->dataLen);
62    it->second++;
63    CString file(recvhd->fileName);//更新日志
64    CString out;
65    out.Format(L":(seq) %d  ", recvhd->seq);
66    out = L"\r\n" + out;
67    out = out + file;
68    pdlg->displayString(pdlg->m_log, out);
69
70    header* hd = new struct header;//生成ACK
71    hd->clientPort = clientPort;
72    hd->serverPort = recvhd->serverPort;
73    hd->isAck = true;
74    memcpy(hd->fileName, recvhd->fileName, 260);
75    hd->dataLen = 0;
76    hd->totalLen = hd->dataLen + sizeof(struct header);
77    hd->seq = recvhd->seq;//确认这个报文被接收了
78
79    skt.SendTo(hd, hd->totalLen, recvhd->serverPort, _T("127.0.0.1"), 0);
80  }
81  else {//resend previous ACK //如果收到的不是期待收到的报文，重发ACK，确认信息是文件目前收
      到的最后一个报文的序列号。
82    CString file(recvhd->fileName);//更新日志
83    CString out;
84    out.Format(L":(seq) %d  ", recvhd->seq);
85    out = L"\r\n" + out;
86    out = out + file;
87    pdlg->displayString(pdlg->m_log, out);
88
89    header* hd = new struct header;
90    hd->clientPort = clientPort;
91    hd->serverPort = recvhd->serverPort;
92    hd->isAck = true;
93    memcpy(hd->fileName, recvhd->fileName, 260);
94    hd->dataLen = 0;
95    hd->totalLen = hd->dataLen + sizeof(struct header);
96    hd->seq = it->second;//如果收到的不是期待收到的报文，重发ACK，确认信息是文件目前收到的最
      后一个报文的序列号it->second。
97
98    skt.SendTo(hd, hd->totalLen, recvhd->serverPort, _T("127.0.0.1"), 0);
```

```
99    }
100
101
102   if (recvhd->dataLen < MAX_PACKET) {//当报文中数据段的长度小于报文最大长度的时候，说明接收
        完毕，关闭socket
103     return 1;
104   }
105   else
106     return 0;
107 }
```

Listing 9: 客户端接收下载的数据

5.2.5 功能：客户端请求上传（发送 120 请求）

代码摘要：单击 “Upload” 按钮后，用户可以选择一个文件发往服务器的根目录。结束文件的选择之后，客户端开启一个新的线程和端口向服务器的原端口发送 120 上传请求。

使用函数介绍：

1. void CRUDPClientDlg::OnBnClickedButtonUpload()
   - 用户选择想要上传的文件。
   - 客户端开启新的上传线程。
2. UINT Uploader(tempUpload* t)
   - 客户端上传线程。在这里开启新的端口，发送 120 请求。

```
1  void CRUDPClientDlg::OnBnClickedButtonUpload()
2  {
3    // TODO: 在此添加控件通知处理程序代码
4
5    CFileDialog dlg(TRUE);
6    CString filePath;
7    if (dlg.DoModal() == IDOK){
8      filePath = dlg.GetPathName(); // return full path and filename
9    }
10
11   tempUpload* t = new tempUpload;
12   t->filePath = filePath;
13
14   CWinThread* m_Uploader = AfxBeginThread(
15     (AFX_THREADPROC)Uploader, // pfnThreadProc: 指向工作者线程的控制函数，它的值不能为NULL
16     t, //
17     THREAD_PRIORITY_NORMAL // 用于指定线程的优先级
18   );
```

```
  if (m_Uploader != nullptr) {
  }
  else {
    AfxMessageBox(L"Upload Thread Create Failed.");
  }
}
UINT Uploader(tempUpload* t) // 上传线程的控制函数
{
  CRUDPClientDlg* pdlg = (CRUDPClientDlg*)AfxGetApp()->GetMainWnd();
  char retchar[4096];
  CString s;
  s.Format(L"120");

  int len = WideCharToMultiByte(CP_ACP, 0, s, -1, NULL, 0, NULL, NULL);
  WideCharToMultiByte(CP_ACP, 0, s, -1, retchar, len, NULL, NULL);

  int port = pdlg->findNextPort() + 60001;
  ClientSocket skt;
  BOOL Flag = skt.Create(port, SOCK_DGRAM, FD_READ | FD_WRITE | FD_OOB | FD_ACCEPT |
    FD_CONNECT | FD_CLOSE);
  while (!Flag) {//端口被占用的话就遍历端口数组
    port = pdlg->findNextPort() + 60001;
    Flag = skt.Create(port, SOCK_DGRAM, FD_READ | FD_WRITE | FD_OOB | FD_ACCEPT | FD_CONNECT
      | FD_CLOSE);
  }

  skt.SendTo(retchar, strlen(retchar), 60000, _T("127.0.0.1"), 0);

  int serverPort = -1;
  while (1) {
    skt.Listen();
    serverPort = skt.waitFor220(skt, port);
    if (serverPort == -1)
      continue;
    else
      break;
  }
  bool done = false;
  while (1) {
    skt.Listen();
    done = skt.uploadFile(t->filePath, skt, port, serverPort);
    if (done) {
      pdlg->str_fileDeque.clear();
      break;
    }
  }
  return 0;
}
```

Listing 10: 客户端请求上传

A RUDP 协议中的命令

客户端指令	指令格式	含义
100	100	请求服务器文件列表
110	110 fileSeq	请求指定文件
120	120	上传文件

服务器指令	指令格式	含义
200	200 filename/filename	发送文件列表，文件之间以'/' 分割
220	220 serverPort	回复客户端的上传请求，通知客户端自己将使用 serverPort 端口接收文件