# 计算机网络--类FTP实现

1713429 曾志韬 计算机科学与技术

# 实验需求

## 多用户同时上传下载

## 文件可靠传输

# 实验原理

**可视化->MFC**

**多用户同时上传下载->多线程**

**可靠传输->ACK+ Seq**

**超时重发->定时器**

**传输层->UDP 数据包**

**日志->传输过程记录txt 客户端和服务器同时备份**

# UI截图

**1 Client**

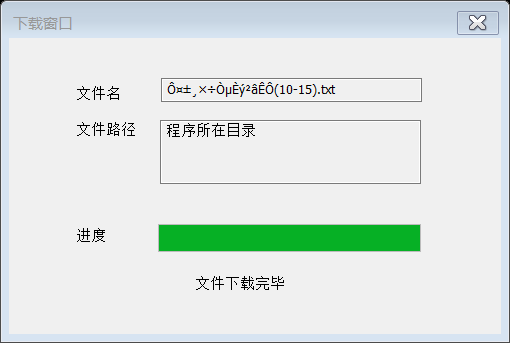
**2 Server**

**3获取文件目录**

**4下载文件 5上传文件**

****

**获取FTP 服务器文件目录**

****

**选中文件 +选择下载命令+发送请求**

****

**图一 客户端 选中上传文件+发送上传命令**

**图二 服务器 接受窗口**

# FTP\_UDP 协议

## 1 文件目录获取

**Client 请求FTP server 文件目录**

**Send 200 记录log C:200**

**Server 接受Search请求 Response 发送当前FTP选中的目录信息 Send 210 记录log C:200 S:210 +Folder文件目录信息**

**Client 收到回复的可用FTP文件目录信息，并显示**

**记录log C:200 S:210**

## 2 文件下载

**Client 请求下载指定Filename文件 Send 300**

**记录log C:300**

**Server 接受下载请求**

**Search 当前指定下载文件是否存在**

**If存在 Send 310 发送指定文件**

**记录log C:300 S:310**

**Else Send 320 回复Client指定文件不存在，并发送更新之后的FTP目录**

**记录log C:300 S:320**

**Client 接收到320，收到请求错误**

**Client 接收到310 可以下载文件, 接受到文件的基本信息**

**Length，filename....**

**Send 330+Seq序列号**

**Server 接受到330+Seq 发送ACK 340+ (Seq+1)**

**当Client判断Seq不匹配，重新发送请求330+Seq**

**Server收到的Seq不匹配，再次发送对应Seq的buffer**

**当Client 判断接受文件大小==length，文件接收成功**

**Send 350 关闭socket**

**当Server收到350 判断Client接收文件成功**

**关闭socket**

### 3 文件上传

**Client 请求上传指定文件**

**Send 400**

**记录log C:400**

**Server 接受上传请求 Send 410**

**记录log C:400 S:410**

**Client 上传 Send 420发送文件基本信息filename，size**

**Server收到420文件基本信息，Send 430+Seq 请求数据报**

**//此处与下载文件类似 ，只是对象互换**

**Client 接受到430 数据请求 Send 440+（Seq+1）**

**Server接受到440 发送ACK 430+Seq继续请求**

**或者buffersize==filesize，下载完毕**

**//此处的接受错误处理和 download一样**

# 核心代码 采用UDP的类FTP通信协议

## 主要实现方法 继承CAsyncSocket类

## 改写Receive Send

## 协议命令通信 +可靠传输 放在UDP数据包头部

1. **void** UpSocket::OnReceive(**int** nErrorCode)
2. {
3. // TODO: 在此添加专用代码和/或调用基类
5. //读输入缓冲区
6. DialogUP\* dialog = (DialogUP\*)dlg;
7. memset(command, 0, 20);
8. ReceiveFrom(command, 20, ip, port,MSG\_PEEK);
9. memset(buffer, 0, 1200);
10. **int** n = ReceiveFrom(buffer, 1200, ip, port);
11. command[3] = '\0';
12. buffer[3] = '\0';
13. buffer[n] = '\0';
15. //判断上传还是下载
16. **if** (isUP) {
17. **if** (strcmp(command, "410") == 0) {  //服务器准备好了
18. //获取需要上传的文件
19. CFileDialog filedlg(**true**);
20. filedlg.DoModal();
21. //获取文件名，文件路径，并显示到对话框
22. CString filename = filedlg.GetFileName();
23. CString filepath = filedlg.GetFolderPath();
24. filepath = filepath + "\\" + filename;
25. dialog->m\_name = filename;
26. dialog->m\_path = filepath;
27. dialog->UpdateData(**false**);
29. //打开文件并获得文件长度
30. string path = (CStringA)filepath.GetBuffer();
31. string name = (CStringA)filename.GetBuffer();
32. in.open(path, ios::in | ios::binary);
33. in.seekg(0, ios::end);
34. filelength = (**int**)in.tellg();
36. //发送数据报
37. string send = "420\n0000\n" + name;   //420\n文件大小（int）\n文件名
38. **int** \*a = (**int**\*)&send[4];
39. \*a = filelength;
40. SendTo(send.c\_str(), send.length(), port, ip);
42. //写日志
43. ofstream out1("log.txt", ios::out | ios::app);
44. out1 << "S:410\nC:420\\n"; out1 << filelength << "\\n";out1 << name.c\_str() << "\n";
45. out1.close();
46. }
47. **else** **if** (strcmp(command, "430") == 0) {  //服务器返回ACK，确定需要发送的数据报序号  430\n下一个请求的数据段号（从0开始，每段1024字节）
48. //获取数据报序号
49. **int** n = \*(**int**\*)(command + 4);  //n\*1024 --- (n+1)\*1024-1
51. //根据接收进度更新进度条（每次10%）
52. **if** (recfilesize < m\_pro && (recfilesize + n - 8) >= m\_pro) {
53. dialog->m\_ctrlpro.StepIt();
54. m\_pro += (filelength / 10);
55. }
57. //读文件，发送序号对应的数据段
58. memset(buffer, 0, 1200);    //440\n数据段号(int)数据
59. buffer[0] = '4'; buffer[1] = '4'; buffer[2] = '0'; buffer[3] = '\n';
60. \*(**int**\*)(buffer + 4) = n;
61. in.seekg(n \* 1024, ios::beg);
62. in.read(buffer + 8, 1024);
63. **if** ((n + 1) \* 1024 - 1 >= filelength)
64. SendTo(buffer, filelength - n \* 1024 + 8, port, ip);
65. **else**
66. SendTo(buffer, 1024 + 8, port, ip);

69. //写日志
70. ofstream out0("log.txt", ios::out | ios::app);
71. out0 << "S:430\\n"; out0 << n << "\t"; out0 << "C:440\t";
72. out0.close();
73. }
74. **else** **if** (strcmp(command, "450") == 0) {  //服务器接收文件完毕
75. //关闭文件
76. in.close();
78. //更改对话框显示状态
79. dialog->m\_finish.ShowWindow(SW\_SHOW);
80. dialog->m\_ctrlpro.SetPos(10);
82. //写日志
83. ofstream out0("log.txt", ios::out | ios::app);
84. out0 << "\nS:450\n";
85. out0.close();
87. //关闭套接字
88. **this**->Close();
89. }
90. **else** {
91. dialog->MessageBox(L"error 2");
92. }
93. }
94. **else** {
95. **if** (strcmp(buffer, "340") == 0) {
96. //关闭计时器
97. dialog->KillTimer(1);
98. //写日志
99. ofstream out0("log.txt", ios::out | ios::app);
100. out0 << "S:340\t";
101. out0.close();
102. //判断是否是需要的下一个数据段
103. **if** (\*(**int**\*)(buffer + 4) == nextnumb) {
104. //根据接收文件大小更新进度条
105. **if** (recfilesize < m\_pro && (recfilesize + n - 8) >= m\_pro) {
106. dialog->m\_ctrlpro.StepIt();
107. m\_pro += (filelength / 10);
108. }
109. //写文件
110. nextnumb++;
111. out.write(buffer + 8, n - 8);
112. recfilesize += n - 8;
113. //根据接收文件大小判断是否接收到整个文件
114. **if** (recfilesize >= filelength) {
115. //关闭写文件流
116. out.close();
117. //发送数据报
118. //350  服务器收取文件完毕
119. buffer[0] = '3'; buffer[1] = '5';
120. buffer[2] = '0'; buffer[3] = '\n';
121. SendTo(buffer, 4, port, ip);
122. //写日志
123. ofstream out0("log.txt", ios::out | ios::app);
124. out0 << "\nC:350\n";
125. out0.close();
126. //更改对话框显示状态
127. dialog->m\_down.ShowWindow(SW\_SHOW);
128. dialog->m\_ctrlpro.SetPos(10);
129. //关闭套接字
130. **this**->Close();
131. }
132. **else** {//文件收取没有结束
133. //继续请求下一个数据段
134. //330\n下一个请求的数据段号（从0开始，每段1024字节）
135. buffer[0] = '3'; buffer[1] = '3';
136. buffer[2] = '0'; buffer[3] = '\n';
137. \*(**int**\*)(buffer + 4) = nextnumb;
138. SendTo(buffer, 8, port, ip);
139. //设置计时器
140. dialog->SetTimer(1, 50, NULL);
141. //写日志
142. ofstream out0("log.txt", ios::out | ios::app);
143. out0 << "C:330\\n"; out0 << nextnumb << "\t";
144. out0.close();
145. }
146. }
147. **else** {//没有收到预期数据段
148. //继续请求当前数据段
149. //430\n下一个请求的数据段号（从0开始，每段1024字节）
150. buffer[0] = '3'; buffer[1] = '3';
151. buffer[2] = '0'; buffer[3] = '\n';
152. \*(**int**\*)(buffer + 4) = nextnumb;
153. SendTo(buffer, 8, port, ip);
154. //设置计时器
155. dialog->SetTimer(1, 50, NULL);
156. //写日志
157. ofstream out0("log.txt", ios::out | ios::app);
158. out0 << "C:330\\n"; out0 << nextnumb << "\t";
159. out0.close();
160. }
161. }
162. **else** {
163. dialog->MessageBox(L"error 1");
164. }
165. }
166. CAsyncSocket::OnReceive(nErrorCode);
167. }

170. **void** UpSocket::MySend()
171. {
172. //对话框控件赋值并更新
173. DialogUP\* dialog = (DialogUP\*)dlg;
174. dialog->m\_name.Format(L"%S", filename.c\_str());
175. dialog->UpdateData(**false**);
177. //新建将要接收的文件，发送数据报
178. out.open(filename, ios::out | ios::binary | ios::trunc);
179. string send = "330\n0000";
180. \*(**int**\*)&(send[4]) = nextnumb;
181. SendTo(send.c\_str(), send.length(), port, ip);
183. //设置计时器
184. dialog->SetTimer(1, 50, NULL);
186. //写日志
187. ofstream out0("log.txt", ios::out | ios::app);
188. out0 << "C:330\\n"; out0 << nextnumb << "\t";
189. out0.close();
190. }

193. **void** UpSocket::sendagain()
194. {
195. // TODO: 在此处添加实现代码.
196. DialogUP\* dialog = (DialogUP\*)dlg;
197. //再次发送需要的下一个数据段序号
198. string send = "330\n0000";
199. \*(**int**\*)&(send[4]) = nextnumb;
200. SendTo(send.c\_str(), send.length(), port, ip);
201. //设置计时器
202. dialog->SetTimer(1, 50, NULL);
203. }
204. **void** ReceiveSocket::OnReceive(**int** nErrorCode)
205. {
206. // TODO: 在此添加专用代码和/或调用基类
207. ReceiveDlg\* recdlg = (ReceiveDlg\*)dlg;
208. memset(buffer, 0, 1200);
209. **int** n = ReceiveFrom(buffer, 1200, ip, port);
210. buffer[3] = '\0';
211. buffer[n] = '\0';
212. **if** (isReceive) {
213. **if** (strcmp(buffer, "420") == 0) {  //420\n文件大小\n文件名
214. //获取文件名，文件大小
215. filesize = \*(**int**\*)(buffer + 4);
216. **char**\* filename = buffer + 9;
217. CString name(filename);
218. //更新窗口
219. recdlg->m\_ip = ip;
220. recdlg->m\_port = port;
221. recdlg->m\_filename = name;
222. recdlg->UpdateData(**false**);
223. //创建文件输出流
224. out.open(filename, ios::out | ios::binary | ios::trunc); //以二进制写入方式打开文件，若文件存在则先删除文件
225. //发送
226. memset(buffer, 0, 1200);
227. buffer[0] = '4'; buffer[1] = '3'; buffer[2] = '0'; buffer[3] = '\n';  //430\n下一个请求的数据段号（从0开始，每段1024字节）
228. \*(**int**\*)(buffer + 4) = 0;
229. nextnumb = 0;
230. SendTo(buffer, 8, port, ip);
231. //设置计时器
232. recdlg->SetTimer(1, 50, NULL);
233. //写日志
234. ofstream out1("log.txt", ios::out | ios::app);
235. out1 << "C(" << ((string)(CStringA)ip.GetBuffer()).c\_str() << ":" << port << "):420\\n" << filesize << "\\n" << filename << "\n";
236. out1 << "S:430\\n" << nextnumb << "\n";
237. out1.close();
238. }
239. **else** **if** (strcmp(buffer, "440") == 0) {
240. //销毁计时器
241. recdlg->KillTimer(1);
242. //写日志
243. ofstream out0("log.txt", ios::out | ios::app);
244. out0 << "C(" << ((string)(CStringA)ip.GetBuffer()).c\_str() << ":" << port << "):420\n";
245. out0.close();
246. //判断是否是需要的数据段
247. **if** (\*(**int**\*)(buffer + 4) == nextnumb) {//是则写入文件
248. nextnumb++;
249. out.write(buffer + 8, n - 8);
250. recfilesize += n - 8;
251. //判断文件是否接收完毕
252. **if** (recfilesize >= filesize) {
253. //发送
254. out.close();
255. buffer[0] = '4'; buffer[1] = '5'; buffer[2] = '0'; buffer[3] = '\n';  //450  服务器收取文件完毕
256. SendTo(buffer, 4, port, ip);
257. //写日志
258. ofstream out0("log.txt", ios::out | ios::app);
259. out0 << "S:450\n";
260. out0.close();
261. //关闭套接字，关闭对话框（对话框关闭后，子线程关闭）
262. **this**->Close();
263. recdlg-> EndDialog(0);
264. }
265. **else** {
266. //发送
267. buffer[0] = '4'; buffer[1] = '3'; buffer[2] = '0'; buffer[3] = '\n';  //430\n下一个请求的数据段号（从0开始，每段1024字节）
268. \*(**int**\*)(buffer + 4) = nextnumb;
269. SendTo(buffer, 8, port, ip);
270. //设置计时器
271. recdlg->SetTimer(1, 50, NULL);
272. //写日志
273. ofstream out0("log.txt", ios::out | ios::app);
274. out0 << "S:430\\n" << nextnumb << "\n";
275. out0.close();
276. }
277. }
278. **else** {//不是则丢弃数据，重新发送ACK
279. //发送
280. buffer[0] = '4'; buffer[1] = '3'; buffer[2] = '0'; buffer[3] = '\n';  //430\n下一个请求的数据段号（从0开始，每段1024字节）
281. \*(**int**\*)(buffer + 4) = nextnumb;
282. SendTo(buffer, 8, port, ip);
283. //设置计时器
284. recdlg->SetTimer(1, 50, NULL);
285. //写日志
286. ofstream out0("log.txt", ios::out | ios::app);
287. out0 << "S:430\\n" << nextnumb << "\n";
288. out0.close();
289. }
290. }
291. }
292. **else** {
293. **if** (strcmp(buffer, "330") == 0) {
294. //客户端返回ACK，确定需要发送的数据报序号
295. //330\n下一个请求的数据段号（从0开始，每段1024字节）
296. **int** n = \*(**int**\*)(buffer + 4);  //n\*1024 --- (n+1)\*1024-1
297. //读取相应数据段数据并发送
298. memset(buffer, 0, 1200);    //340\n数据段号(int)数据
299. buffer[0] = '3'; buffer[1] = '4'; buffer[2] = '0'; buffer[3] = '\n';
300. \*(**int**\*)(buffer + 4) = n;
301. in.seekg(n \* 1024, ios::beg);
302. in.read(buffer + 8, 1024);
303. **if** ((n + 1) \* 1024 - 1 >= filesize) {
304. SendTo(buffer, filesize - n \* 1024 + 8, port, ip);
305. }
306. **else** {
307. SendTo(buffer, 1024 + 8, port, ip);
308. }
309. //写日志
310. ofstream out1("log.txt", ios::out | ios::app);
311. out1 << "C(" << ((string)(CStringA)ip.GetBuffer()).c\_str() << ":" << port << "):330\\n" << n << "\n";
312. out1 << "S:340\n";
313. out1.close();
314. }
315. **else** **if** (strcmp(buffer, "350") == 0) {  //服务器接收文件完毕
316. //写日志
317. ofstream out0("log.txt", ios::out | ios::app);
318. out0 << "C(" << ((string)(CStringA)ip.GetBuffer()).c\_str() << ":" << port << "):350\n";
319. out0.close();
320. //保存文件，关闭套接字，关闭对话框
321. in.close();
322. **this**->Close();
323. recdlg->EndDialog(0);
324. }
325. }
326. }


330. **void** ReceiveSocket::MySend()
331. {
332. // TODO: 在此添加专用代码和/或调用基类
333. //创建文件输出流
334. in.open(filename, ios::in | ios::binary);
335. in.seekg(0, ios::end);
336. filesize = (**int**)in.tellg();   //获取文件大小
337. //更新窗口
338. ReceiveDlg\* recdlg = (ReceiveDlg\*)dlg;
339. recdlg->m\_ip = ip;
340. recdlg->m\_port = port;
341. recdlg->m\_filename.Format(L"%S",filename.c\_str());
342. recdlg->UpdateData(**false**);
344. //310\n文件大小（int）\n文件名
345. string send = "310\n0000\n" + filename;
346. **int** \*a = (**int**\*)&send[4];
347. \*a = filesize;
348. //发送
349. SendTo(send.c\_str(), send.length(), port, ip);
350. //写日志
351. ofstream out0("log.txt", ios::out | ios::app);
352. out0 << "S:310\\n" << filesize << "\\n" << filename.c\_str() << ")\n";
353. out0.close();
354. }

357. **void** ReceiveSocket::sendagain()
358. {
359. // TODO: 在此处添加实现代码.
360. //发送ACK
361. buffer[0] = '4'; buffer[1] = '3'; buffer[2] = '0'; buffer[3] = '\n';  //430\n下一个请求的数据段号（从0开始，每段1024字节）
362. \*(**int**\*)(buffer + 4) = nextnumb;
363. SendTo(buffer, 8, port, ip);
364. //设置计时器
365. ReceiveDlg\* recdlg = (ReceiveDlg\*)dlg;
366. recdlg->SetTimer(1, 50, NULL);
367. //写日志
368. ofstream out0("log.txt", ios::out | ios::app);
369. out0 << "S:430\\n" << nextnumb << "\n";
370. out0.close();
371. }