**摘要**

从世界第一台计算机诞生至今只有半个多世纪，然而正因为它的方便、快捷、实用，已经被广泛应用到各个领域。随着网络的出现，信息资源的实时共享成为现实，更重要的是它提供了一种极为直接的交互方式，当然这种交互不但是指数据、信息之间的传递，也包括人与人之间的相互交流，各种聊天软件和在线聊天室正是承载这类交互的媒介。随着互联网的发展，人们通过网络可以更方便快捷地进行信息交流和学习，聊天系统可以为大家提供一个更好的交流平台，在这个平台上，人们可以通过文字、符号与表情进行实时的聊天，可以通过传输文件进行信息的交互。它是一个向整个因特网开放的地方，提供给网民一个交友与娱乐的场所，在聊天室里网民可以和各种各样的人进行交流和对话，是网民之间相互沟通、交流情感的最佳方式。通过python实现的网络聊天室，熟练使用socket

编写客户端和服务器程序，主要功能包括用户登录，在线聊天和文件传输。通过网络聊天室系统的设计，熟悉网站建设基本操作。

关键词：在线聊天室，python，socket

Abstract

Since the birth of the first computer in the world, it has been widely used in various fields because of its convenience, rapidity and practicality. With the emergence of the network, the real-time sharing of information resources has become a reality. More importantly, it provides a very direct way of interaction. Of course, this kind of interaction not only refers to the transmission of data and information, but also includes the mutual communication between people. Various chat software and online chat rooms are the media carrying this kind of interaction. With the development of the Internet, people can exchange and learn information more conveniently and quickly through the network. The chat system can provide a better communication platform for everyone. On this platform, people can chat in real time through words, symbols and expressions, and can exchange information by transferring files. It is a place open to the whole Internet, providing a place for Internet users to make friends and entertainment. In the chat room, netizens can communicate and talk with all kinds of people. It is the best way for netizens to communicate and exchange feelings with each other. The main functions include user login, online chat and file transmission. Through the design of network chat room system, familiar with the basic operation of website construction.

Keywords: online chat room, python, socket

1. **绪论**

**1.1课题概述**

人们日常生活中越来越多地使用网络为自己的工作和学习服务，一个操作简单、界面友好、运行稳定的网络聊天室对于用户可以起到很好的交流作用。所谓聊天室就是一个可以让许多用户同时互相通信的网络工具，个人用户可以通过网络聊天室进行与其他用户的通信。

聊天室是当今广泛应用的一种网络服务, 它为广大网民提供了一种方便快捷的沟通方式, 在线聊天是上网的重要活动之一。第十六次CNNIC 调查结果显示: 在网络用户经常使用的网络服务/功能调查中, 网上聊天( 聊天室、QQ、ICQ 等)占被调查人群的42.6%, 处于第四位, 作为操作简便也是出现最早的网上聊天方式, 聊天室聊天又是其中最普及的一种。而多数人认为聊天室是一种用来消遣而缺乏实际价值的东西, 许多大型网站建立了聊天室, 但基本上是被定义为聚拢人气而不是带来收益的一种服务项目。

**1.2网络聊天室研发内容和意义**

网路聊天室基于TCP通信，实现收发并行，客户端与服务器之间的通信是基于TCP协议的,服务器可以通过TCP套接字实时掌握与客户端之间的通信情况。通过文件传输主副套接字，合理开放命令通道和数据通道，实现文件传输功能。

在功能上，用户可以选择文字、符号和表情进行交流，并且可以通过下载其他用户在聊天室上传的文件或者上传文件至聊天室进行信息的交互。网络聊天室拥有即时通讯能力，即时信息可以发送到每个用户的聊天室页面。网络聊天实现了用户自主选择主题角色的自由性。一方面，这种可塑性可以使得用户的心理需求的多面性得到淋漓尽致地表达，从而以一种特殊的自我疏导机制释放了现实世界对人之原本复杂的心理需求的压抑，在这个意义上，角色的可塑性具有心理调适的积极意义。另一方面，网络聊天的角色可塑性同时也预示着网络聊天意义的复杂性。虚拟的网络聊天克服了现实聊天的空间制约，使得聊天的空间大大扩展。只要有互联网，就能使处于不用地方的人相互交流，使用户的交际圈拓宽，在交往的意义上，时间维度的拓展意味着交往的自由度进一步增大，交往的层次因为网络聊天室的存在得到了提升。

**1.3本章小结**

本章主要阐述了课题的概述和背景，网络聊天室的研发内容和研发意义。

1. **需求分析**

**2.1用户需求**

对使用网络聊天室的用户来说，网络聊天室可以即时使用文字、表情等来和其他人交流，也可以下载自己需要的文件信息，亦或将自己想分享的文件上传到聊天室供他人查看和学习。

**2.2功能需求**

**2.2.1用户登录模块**

登录界面会显示用户所在地的服务器IP地址，用户通过输入自己的用户来登录进入网络聊天室。

**2.2.2在线用户显示模块**

即时统计在线用户的人数并将在线用户显示在页面上。

**2.2.3用户发言模块**

用户可以在发言框输入文字、符号，也可以选择表情，单击“发送”按钮可以发送内容至聊天框，还能选择本地文件上传，

**2.2.4聊天内容显示模块**

主要将每个用户的聊天内容实时显示在网页上，也可以显示用户上传的文件记录。

**2.2.5文件传输模块**

用户可以在此页面上传或下载文件，也可以通过刷新来显示其他用户刚上传的文件。

**2.3性能需求**

**2.3.1性能环境**

操作系统：Windows 10

编程语言：Python 3.7.6

第三方包：pyside2 5.15.1

**2.4本章小结**

本章阐述了网络聊天室的用户需求，功能模块和系统运行的性能需求。

**第三章 系统设计**

**3.1系统模块设计**

登录聊天室后，用户可以使用的功能包括聊天功能和文件功能，其中聊天功能包括文字和表情的输入和发送，文件功能包括上传文件和下载文件。功能模块图如图1所示。

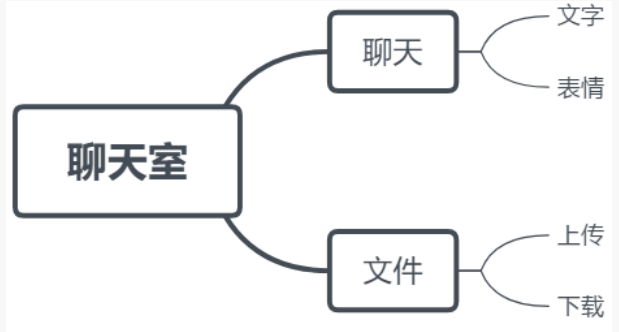


图1 功能模块

用户信息模块中，在服务器中使用了列表存储每一个用户信息，为了区分不同用户，用户名必须是唯一不可重复的。其中的套接字是服务器发挥中间站转发的关键。用户信息图如图2所示。

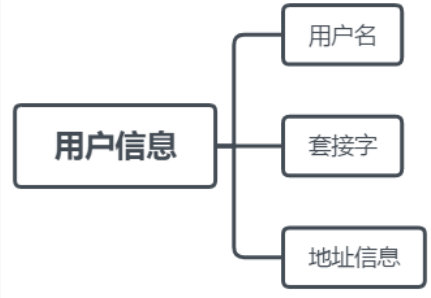


图2 用户信息

**3.2设计思路**

**3.2.1网络聊天室设计思路**

服务端开启后即时监听，主线程用以与客户端进行连接，子线程接受消息，在连接后开辟新线程处理消息，处理完毕后进行转发。

客户端开启后连接服务器，用户登录后判断是否登录成功，如果没有成功登录，便重新连接服务器，若是服务器连接成功，主线程选择阻塞，等待用户操作界面，子线程在收到消息后选择阻塞，执行接受消息操作，并建立新线程用以处理消息并反馈到用户界面。用户在操作界面结束后将内容发送至服务器并返回上一操作（线程阻塞），用户可以自行选择退出客户端结束线程。

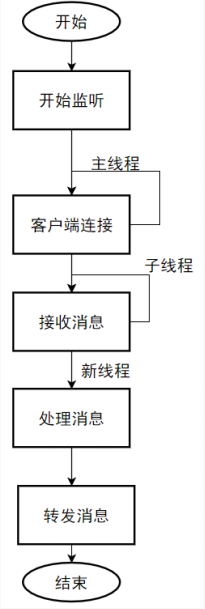


图3 服务端流程图

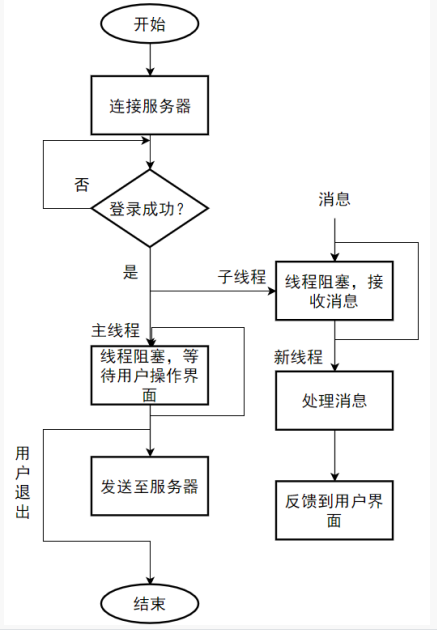


图4 客户端流程图

**3.2.2基于TCP通信**

由于TCP的可靠性，面向连接特点，在本例中，客户端与服务器之间的通信是基于TCP协议的。服务器可以通过TCP套接字实时掌握与客户端之间的通信情况。

**3.2.3收发并行**

为了客户端界面不会因为接收信息而遭到阻塞无法操作，另外开辟了子线程来接收消息。有些如IO这些比较耗时间的操作，会另外开辟新线程来处理。

**3.2.4用户信息**

在服务器中使用了列表存储每一个用户信息，为了区分不同用户，用户名必是唯一不可重复的。其中的套接字是服务器发挥中间站转发的关键。

**3.2.5消息**

通过固定格式的消息实现客户端与服务端之间的通信。

**消息格式**

在本例中，服务器主要是发挥了中间站转发作用，通过消息中的接收者来进行消息的转发。



图5 消息格式图

**消息类型**

服务器通过消息类型和消息内容来进行进一步的消息处理，是转发还是返回响应内容。

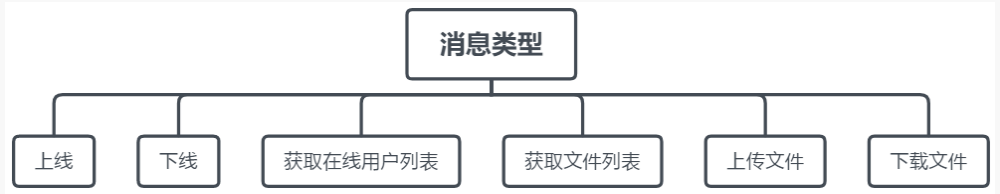


图6 消息类型图

**3.2.6粘包**

由于TCP面向字节流的特点，在本例中，可能会出现多条消息同时接收的情况。因此，可以通过特定字符作为分割标志来把多条消息分开；为了防止消息内容中出现分割标志导致消息分割混乱，该字符的选择应选取ASCII码中的不可打印字符/控制字符。

**3.2.7文件传输模块设计思路**

在文件传输中，有两个用于通信的套接字，一个是主套接字，是命令通道，用于收发各种请求消息；另一个是副套接字，是数据通道，只用于传输文件数据。

为了降低通信的复杂性，是由服务端发起连接到客户端建立的套接字，该套接字只用于传输文件数据，请求队列的最大长度为1，是一次性的。

首先是客户端建立副套接字，然后使用命令通道向服务器发送副套接字信息和件传输请求，服务器通过命令通道收到文件传输请求消息后，使用副套接字连接到客户端副套接字，从而建立数据通道，开始文件传输。

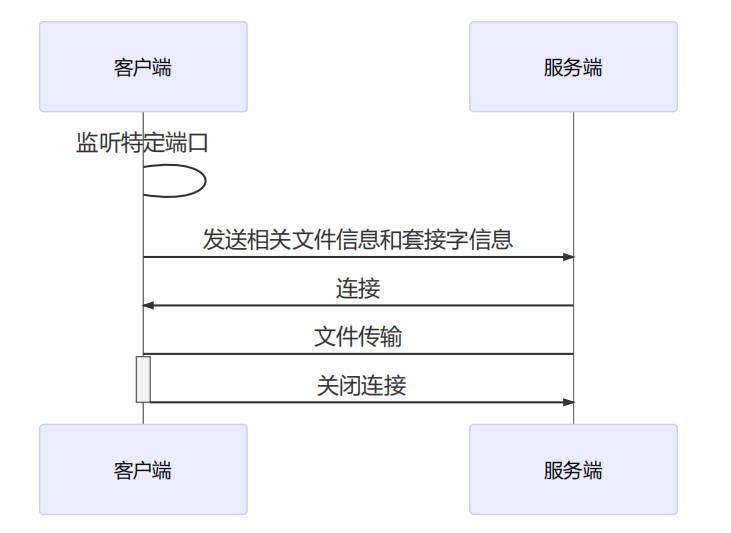


图7 文件传输流程图

**3.2.8停止文件传输基本原理**

通过关闭相应文件传输任务的套接字，导致该套接字抛出异常，停止该套接字收发数据，从而停止文件传输任务。因为每一个文件传输任务都会开启一个新的子线程，所以可以在主线程中关闭文件传输任务的套接字。

**3.2.9存储文件传输任务**

为了在主线程管理文件传输任务，在主线程中使用字典存储每一个文件传输任务，使用传输方向和文件名的生成UUID（唯一性）作为键，对应的套接字作为值。通过管理该文件传输任务字典从而达到管理所有的文件传输任务。

**3.3本章小结**

本章阐述了系统各个模块的设计理念以及聊天室主体和文件传输模块的设计思路

**第四章 系统实现**

**4.1小组分工**

组长XXX：负责网络聊天室的底层设计，整体功能实现

组员XXX：负责网络聊天室界面的UI设计

组员XXX：负责网络聊天室文件传输模块的实现

**4.2网络聊天室的实现**

**4.2.1服务端**

在tcpserver.py源文件中处理客户端会话，首先对每一个连接的客户端开启一个线程，根据客户端发送过来的用户名进行唯一性处理，然后等待接收消息

1. **def** start(self):
2. self.servSock.bind(self.ADDR)
3. self.servSock.listen(120)
4. **print**('服务器已就绪....')
5. **while** True:
6. clntSock, clntAddr = self.servSock.accept()
7. threading.Thread(target=self.tcp\_connect, args=(clntSock, clntAddr)).start()
9. **def** tcp\_connect(self, clntSock, clntAddr):
10. '''''
11. 处理每个客户端连接/通信
12. :param clntSock:客户端套接字
13. :param clntAddr:客户端地址和端口信息
14. :return:
15. '''
16. userName = None
17. **try**:
18. self.msgHandler = ServerMessageHandler(func\_errorHandle=self.errorHandle, userInfoList\_online=self.userInfoList\_online, dict\_msg=None)
19. # 首先是确认用户名(唯一性)
20. userName = clntSock.recv(self.BUFSIZE).decode()
21. userName = self.msgHandler.onLine(userName, clntSock, clntAddr)
23. # 处理信息（核心）
24. isEnd = False
25. **while** **not** isEnd:
26. rData = clntSock.recv(self.BUFSIZE).decode()
27. str\_msgs = rData.split(u'\x02\x02')  # 以不可见字符分割解决tcp粘包问题
28. **for** str\_msg **in** str\_msgs:
29. **if** (str\_msg != ''):  # 空字符
30. dict\_msg = eval(str\_msg)
31. **if**('OFFLINE' == dict\_msg['msgType']):
32. isEnd = True
33. self.msgHandler = ServerMessageHandler(func\_errorHandle=self.errorHandle, userInfoList\_online=self.userInfoList\_online, dict\_msg=dict\_msg)
34. self.msgHandler.start()
35. **except**:
36. self.errorHandle(userName + '连接强制断开', True)
37. self.msgHandler.offLine(userName)

在源文件ServerMessageHandler.py中进行消息处理

1. **def** handleMsg(self, dict\_msg):
2. '''''
3. 核心函数：处理来自客户端的信息
4. :param str\_msg:
5. :return:
6. '''
7. **if** dict\_msg['msgType'] == 'ONLINE':
8. self.broadcast(dict\_msg)
9. **elif** dict\_msg['msgType'] == 'OFFLINE':
10. self.offLine(dict\_msg['msgFrom'])
11. **elif** dict\_msg['msgType'] == 'CHATMSG':
12. self.broadcast(dict\_msg)
13. **elif** dict\_msg['msgType'] == 'LISTFILE':
14. dict\_msg['msgOther'] = self.getFilesList(self.uploadBasePath)
15. self.broadcast(dict\_msg)
16. **elif** dict\_msg['msgType'] == 'LISTUSER':
17. self.broadcast(dict\_msg)
18. **elif** dict\_msg['msgType'] == 'UPFILE':
19. self.uploadFile(dict\_msg)
20. **elif** dict\_msg['msgType'] == 'DOWNFILE':
21. self.downloadFile(dict\_msg)
22. **else**:
23. **print**('传输格式错误')

**4.2.2客户端**

在源文件tcp\_login.py中建立会话，先连接到服务器，然后发送用户名（唯一），开启消息接收线程

1. **def** onLine(self, userName, serverAddr):
2. '''''
3. 客户端上线/连接服务器/开始主进程
4. :param userName: 用户名
5. :param serverAddr: 服务器密码
6. :return: 连接结果
7. '''
8. isConnected = False
9. self.mSocket = socket(AF\_INET, SOCK\_STREAM)
10. **try**:
11. self.mSocket.connect(serverAddr)
12. self.mSocket.send(userName.encode())
14. self.username = self.mSocket.recv(self.BUFSIZE).decode()
15. self.fileHandler.username = self.username
16. self.recvMsgThread = threading.Thread(target=self.tcpRecvMsg)
17. self.recvMsgThread.start()
18. isConnected = True
19. **print**('【上线】', (userName, serverAddr))
20. # self.tcpSendMsg( 'ONLINE', self.username, 'ALL', self.username + '上线了')
21. **except**:
22. self.errorHandle(''.join([serverAddr, '目标主机无法连接']))
23. **finally**:
24. **return** isConnected

在源文件tcp\_login.py中进行消息处理

1. **def** handleMsg(self, str\_Msg):
2. '''''
3. 处理接收的消息
4. :param str\_Msg: 以字符串形式的消息
5. :return:
6. '''
7. # 解构消息
8. **print**('<<:' + str\_Msg)
9. dict\_msg = eval(str\_Msg)
10. msgType = dict\_msg['msgType']
11. msgFrom = dict\_msg['msgFrom']
12. msgTo = dict\_msg['msgTo']
13. msgOther = dict\_msg['msgOther']
14. time = dict\_msg['time']
16. # 处理消息类型
17. curTime = time
18. **if** ('CHATMSG' == msgType
19. **or** 'ONLINE' == msgType
20. **or** 'OFFLINE' == msgType
21. **or** 'UPFILE' == msgType):
22. isMe = 2
23. msgBgColor = 'white'
24. strFormat = '【{0}】:\n{1}'.format(str(curTime), str(msgOther))
25. **if** (self.username == msgFrom):
26. msgBgColor = '#9eea6a'
27. isMe = 1
28. **if** (msgType != 'CHATMSG'):
29. msgBgColor = '#dadada'
30. isMe = 0
31. strFormat = '【{0}】:{1}'.format(str(curTime), str(msgOther))
32. self.ui\_mainBoard.signal\_refresh\_chatMsgShowListWidget.emit(msgFrom, strFormat, isMe, msgBgColor,
33. 'ui/ico/ico.png')
34. **elif** ('LISTFILE' == msgType):
35. filesList = msgOther
36. self.ui\_mainBoard.signal\_refresh\_filesShowTableWidget.emit(filesList)
37. **elif** ('LISTUSER' == msgType):
38. self.usr\_online = msgOther
39. self.ui\_mainBoard.refresh\_chatUserOnLineTableWidget(self.usr\_online)

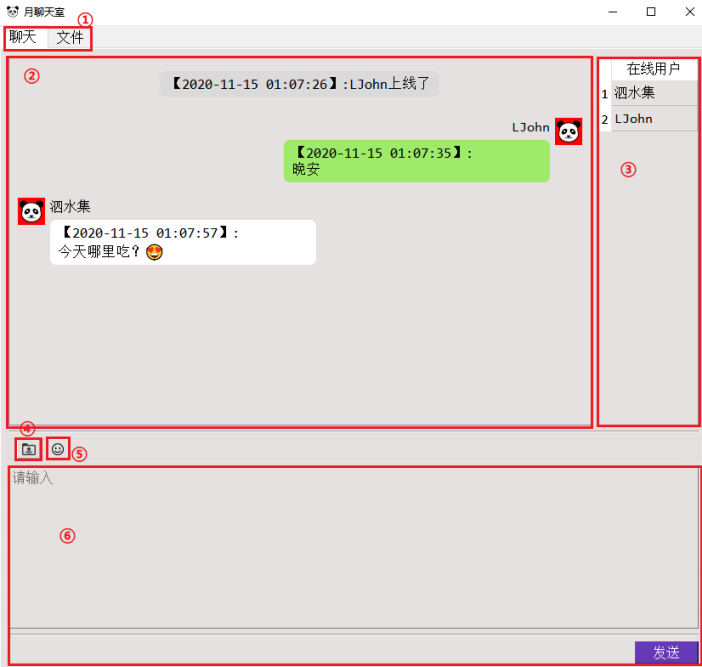
**4.3网络聊天室界面的设计**

**4.3.1核心功能界面**

登录界面



聊天窗口界面



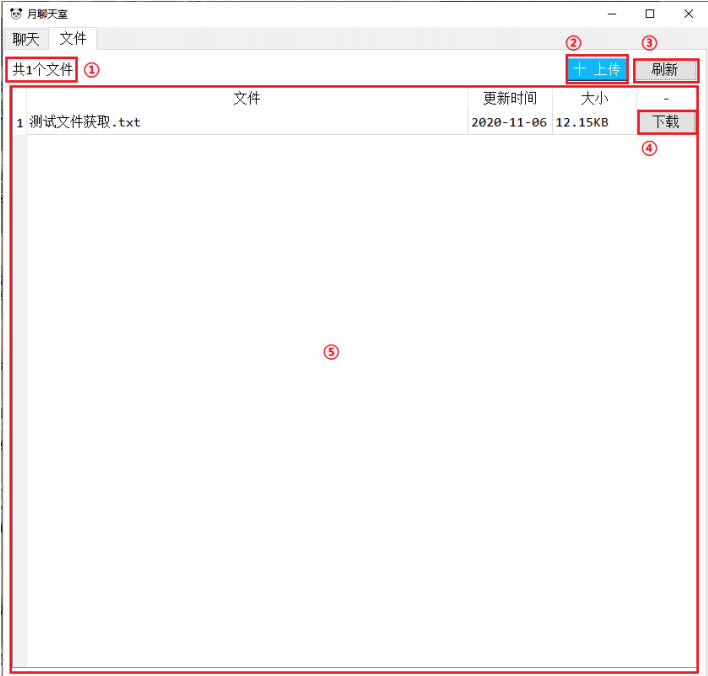


表情符号面板



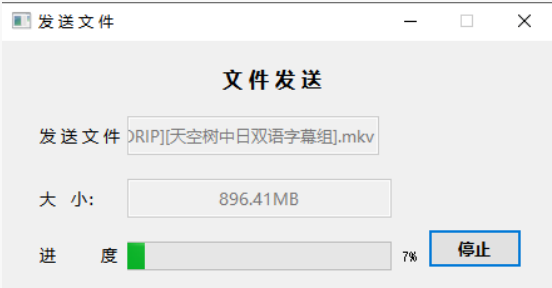
**4.3.2文件界面**

文件展示

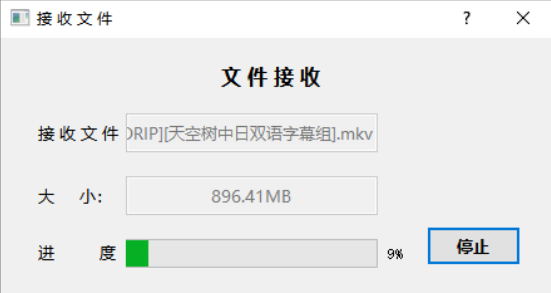




文件上传



文件下载



**4.3.3消息气泡设计**

**消息气泡类型**

消息气泡分三种类型。

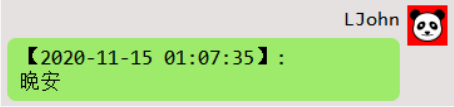
公共通知：

显示用户上下线和文件上传的消息，在聊天展示区中间位置，气泡呈现灰色背景色，如下图所示：



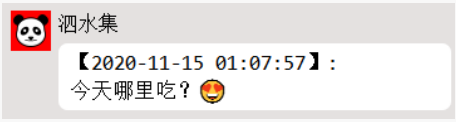
自己的消息：

显示用户自己发出的聊天消息，在聊天展示区右边位置，气泡呈现绿色背景色，如下图所示：



他人的消息：

显示其他用户发出的聊天消息，在聊天展示区左边位置，气泡呈现白色背景色，如下图所示：



**源码实现**

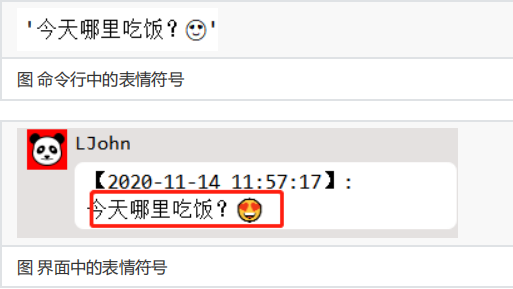
在mainboard.py源文件中，通过调用refresh\_chatMsgShowListWidget()函数根据传入的参数来生成对应的气泡

1. **def** refresh\_chatMsgShowListWidget(self, userName, msg, isMe=1, msgBgColor='#9eea6a',userHeader='ui/ico/ico.png'):
2. '''''
3. 更新聊天展示区/生成消息气泡
4. :param userName: 聊天消息发出者
5. :param msg: 消息内容
6. :param isMe: 消息气泡类型
7. :param msgBgColor: 消息气泡背景色
8. :param userHeader: 聊天消息气泡头像
9. :return:
10. '''
11. lwit = QListWidgetItem() # 列表项
12. self.chatMsgShowListWidget.addItem(lwit)
13. # 根据isMe的值生成相应的消息气泡
14. **if** 0 == isMe:
15. mif = self.newMsgItemInformFrame(msg, msgBgColor) # 生成公共通知气泡
16. **else**:
17. mif = self.newMsgItemChatFrame(userName, msg, isMe, msgBgColor, userHeader) # 生成聊天气泡
19. # 调整列表项大小 lwit.setSizeHint(QSize(400, mif.height()))
20. # 把消息气泡与列表项绑定在一起，添加到列表（聊天展示区）中
21. self.chatMsgShowListWidget.setItemWidget(lwit, mif) # 滚动条到末尾
22. self.chatMsgShowListWidget.setCurrentRow(self.chatMsgShowListWidget.count() - 1)

**4.3.4表情符号面板设计**

**基本原理**

在windows系统下，微软输入法自带的表情符号，会自动把对应字符串转换成emoji表情，表现如图：



微软输入法的表情符号的bytes编码是连续的，可以设置起始表情符号，通过对表情符号的bytes编码进行递增，从而获取一定的表情符号字符串。

**源码实现**

在tools.py源文件获取表情符号，代码如下：

1. **def** getEmoji():
2. '''''
3. 获取微软输入法的emoji
4. :return: emoji列表
5. '''
6. emoji = []
7. # 把str类型的表情符号转换成bytes类型
8. start = int.from\_bytes('.encode(), byteorder='big', signed=False)
9. end = int.from\_bytes('.encode(),byteorder='big', signed=False)
11. # 对bytes编码进行递增
12. **for** str **in** range(start, end, 1):
13. # 把bytes类型表情符号转换成str类型，并存储
14. emoji.append(struct.pack('>L', str).decode())
15. **return** emoji

在emojiboard.py源文件中实现面板表格数据更新

1. **def** refresh\_emojiTableWidget(self):
2. '''''
3. 填充emoji表格并设置窗口大小
4. :return:
5. '''
6. self.setWindowFlags(Qt.FramelessWindowHint | Qt.Popup) # 设置窗口样式为无标 题栏和弹出式顶层
7. etw = self.emojiTableWidget # 表格面板
8. etw.setRowCount(5) # 表格列
9. etw.setColumnCount(0) # 表格行
11. emojiList = getEmoji() # 获取表情符号列表
12. rows = etw.rowCount()
13. index = 0 # 表情符号列表下标
14. **while**(index < len(emojiList)):
15. colPosition = etw.columnCount() # 列位置
16. etw.insertColumn(colPosition) # 插入新列
17. # 纵向插入
18. **for** rowPosition **in** range(0, rows):
19. **if**(index < len(emojiList)): # 把表情符号插入表格面板中
20. etw.setItem(rowPosition, colPosition, QTableWidgetItem(emojiList[index])) index += 1
22. # 表格，窗体根据内容自适应大小
23. etw.setFixedSize(etw.horizontalHeader().length() +
24. etw.verticalHeader().width(), etw.height()) self.setFixedSize(etw.width(),self.height())

**4.3.5文件下载设计**

因为表格是动态生成的，每一个下载按钮也需要都需要动态地绑定点击事件响应函数，而每一个下载按钮的点击事件响应函数所需要的参数是固定值。

使用functools.partial() 函数可以把点击事件响应函数与固定参数联系起来。

**源码实现**

在mainboard.py源文件的 refresh\_filesShowTableWidget() 函数中实现动态绑定固定值。

1. fileDownloadPushButton.clicked.connect(
2. functools.partial(self.on\_fileDownloadPushButton\_click, fileName, fileInfo['fileSize']))

**4.4文件传输模块的实现**

**源码实现**

**存储文件传输任务**

为了在主线程管理文件传输任务，在主线程中使用字典存储每一个文件传输任务，使用传输方向和文件名的生成UUID（唯一性）作为键，对应的套接字作为值。通过管理该文件传输任务字典从而达到管理所有的文件传输任务。

1. # 文件传输任务字典
2. fileTransferTasks = {uuid1:套接字1, uuid2:套接字2, ...}

**获取文件传输任务的UUID**

1. **def** getFileUUID(self, fileName):
2. **return** str(uuid.uuid5(uuid.NAMESPACE\_DNS, fileName))

**防止文件传输任务重复**

1. **def** isExistFileTrasferTask(self, fileUUID):
2. **if** (fileUUID **in** self.fileTransferTasks):
3. **return** True
4. **else**:
5. **return** False

**添加文件传输任务**

1. **def** addFileTransferTask(self, fileName, fileSock):
2. fileUUID = self.getFileUUID(fileName)
3. **if** self.isExistFileTrasferTask(self.isExistFileTrasferTask(fileUUID)):
4. **return** False
5. self.fileTransferTasks[fileUUID] = fileSock
6. **print**('【文件任务】', '添加', fileName, fileUUID)
7. **return** True

**移除文件传输任务**

1. **def** removeFileTransferTask(self, fileName):
2. fileUUID = self.getFileUUID(fileName)
3. **if** **not** self.isExistFileTrasferTask(fileUUID):
4. **return** None
5. fileSock = self.fileTransferTasks[fileUUID]
6. **del** self.fileTransferTasks[fileUUID]
7. **print**('\n【文件任务】', '移除', fileName, fileUUID)
8. **return** fileSock

**停止文件传输任务**

1. **def** interruptFileTransfer(self, fileName, isUpload):
2. **print**('\n【阻止传输】', fileName)
3. **if** isUpload:
4. fileName = 'UPLOAD' + fileName
5. **else**:
6. fileName = 'DOWNLOAD' + fileName
7. fileSock = self.removeFileTransferTask(fileName)
8. **if** **not** fileSock:
9. **return** False
10. **else**:
11. fileSock.close() # 关键
12. **return** True

在本例中，上传文件时，客户端一方作为文件服务器因为网络原因被动停止或用户主动停止文件传输，导致客户端一方的文件服务器套接字关闭，服务端一方不能正常关闭通信。在客户端一方的文件服务器套接字关闭或异常时，服务端一方会一直处于接收 b'' 空字节类型死循环状态（非阻塞）。如果接收到 b'' 空字节类型，服务端一方退出死循环，并关闭通信。

1. # 服务端一方接收客户端的文件上传核心过程
2. **try**:
3. **while** **not** fileSzieRecv == fileSizeTotal:
4. **if** (fileSizeTotal - fileSzieRecv > self.BUFSIZE):
5. rData = fileSock.recv(self.BUFSIZE)
6. fileSzieRecv += len(rData)
7. # 因为Python中recv()是阻塞的(握手挥手)，只有连接断开或异常(send)时，接收 到的是b''空字节类型，因此需要判断这种情况就断开连接。
8. **if** (b'' == rData):
9. **raise** Exception('文件上传通道已关闭') # 退出死循环
10. **else**:
11. rData = fileSock.recv(fileSizeTotal - fileSzieRecv)
12. fileSzieRecv = fileSizeTotal
13. **print**('【文件上传】', '结束', fileSizeTotal, absFileName)
14. fp.write(rData)
15. **except**:
16. self.errorHandle('文件上传通道已关闭', False)
17. **finally**:
18. fp.close() fileSock.close() # 关闭通信

**4.5本章小结**

本章阐述了小组成员的简单分工以及各模块功能的实现和设计。

**第五章 系统测试**

**5.1 登录/上线**

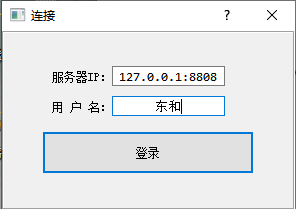


图5.1.1 我方登录界面

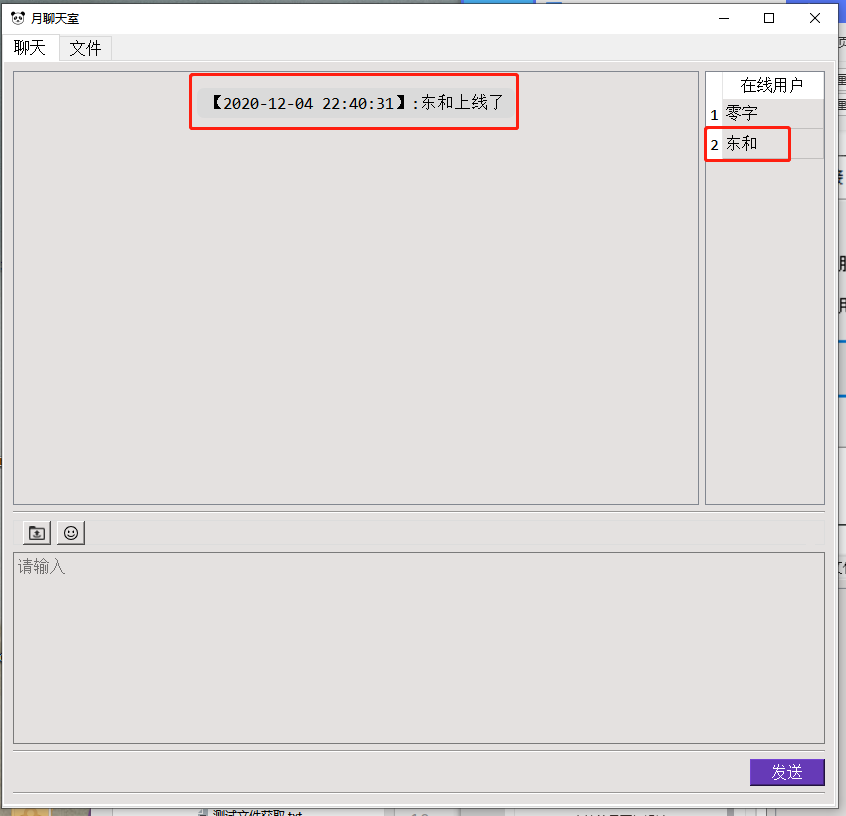


图5.1.2 我方聊天界面

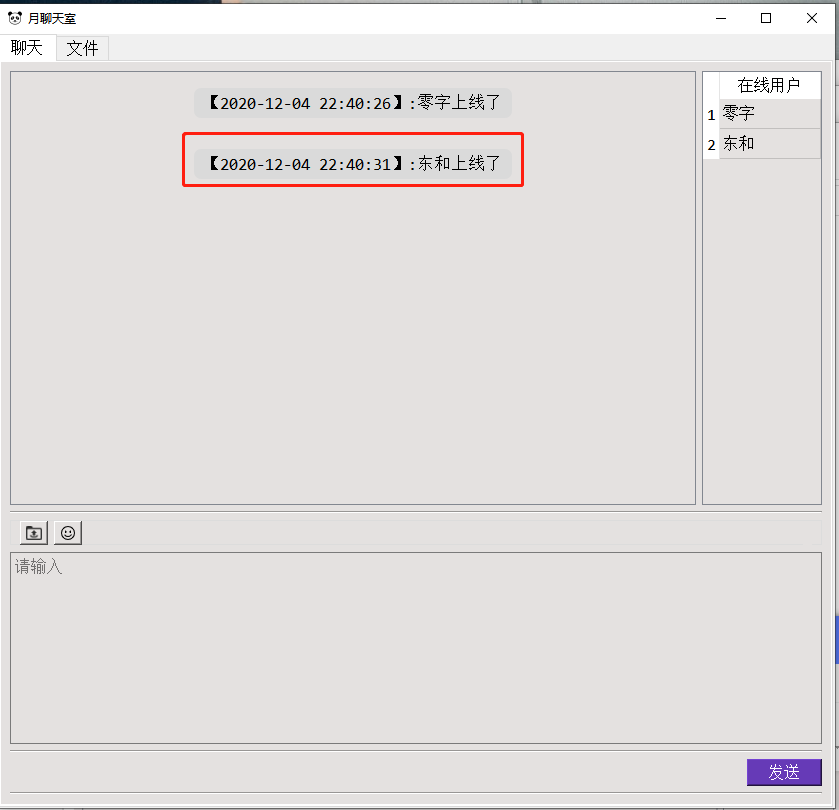


图5.1.3 对方聊天界面

**5.2 聊天**

**5.2.1 文字聊天**

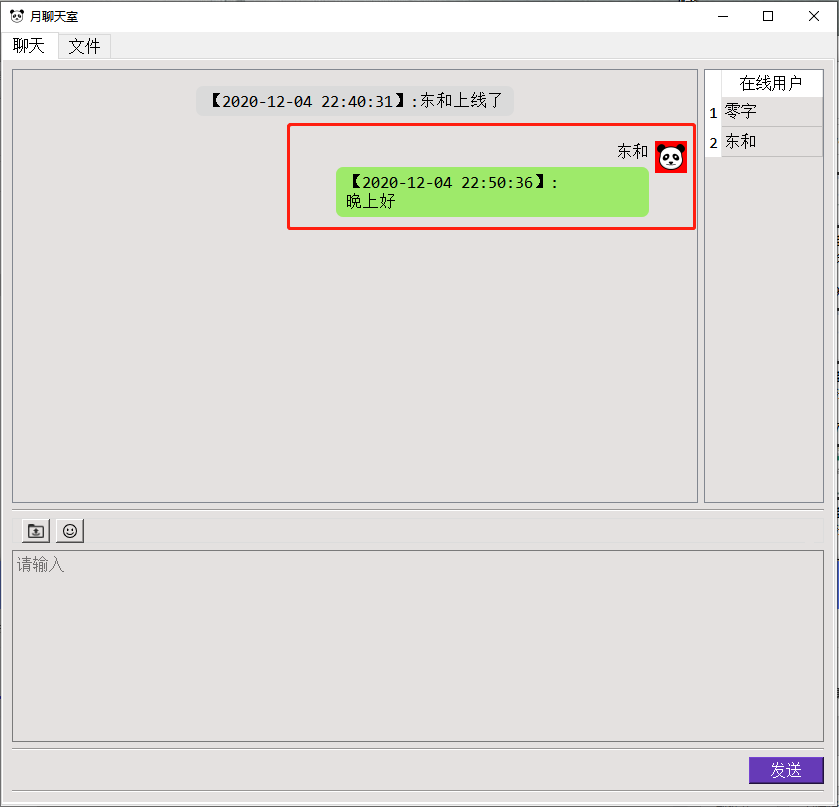


图5.2.1 我方聊天气泡显示

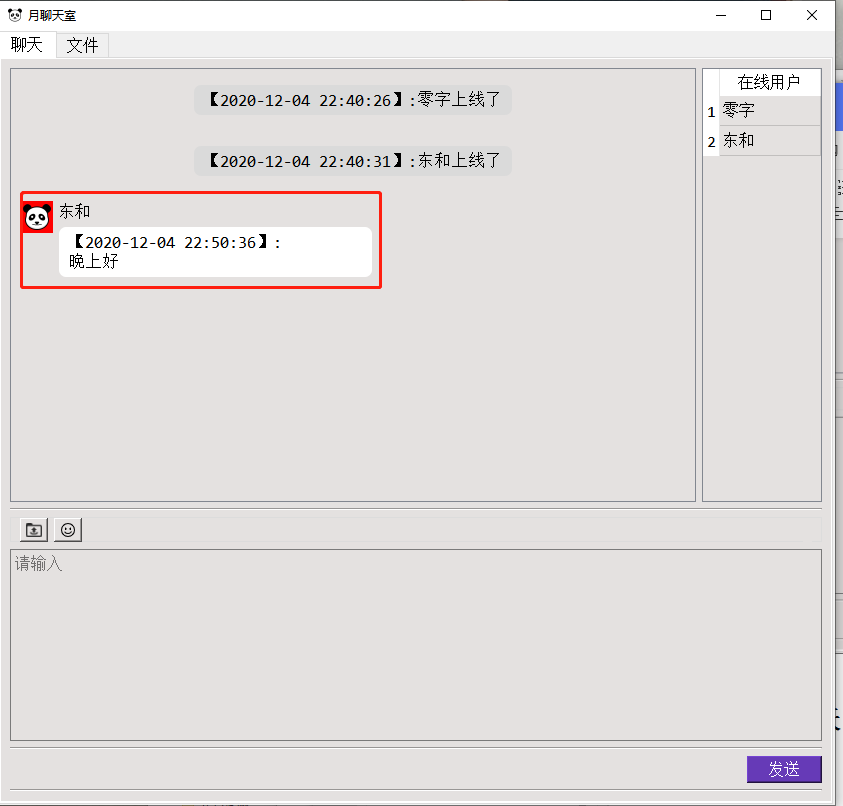


图5.2.2 对方聊天气泡显示

**5.2.2 表情聊天**

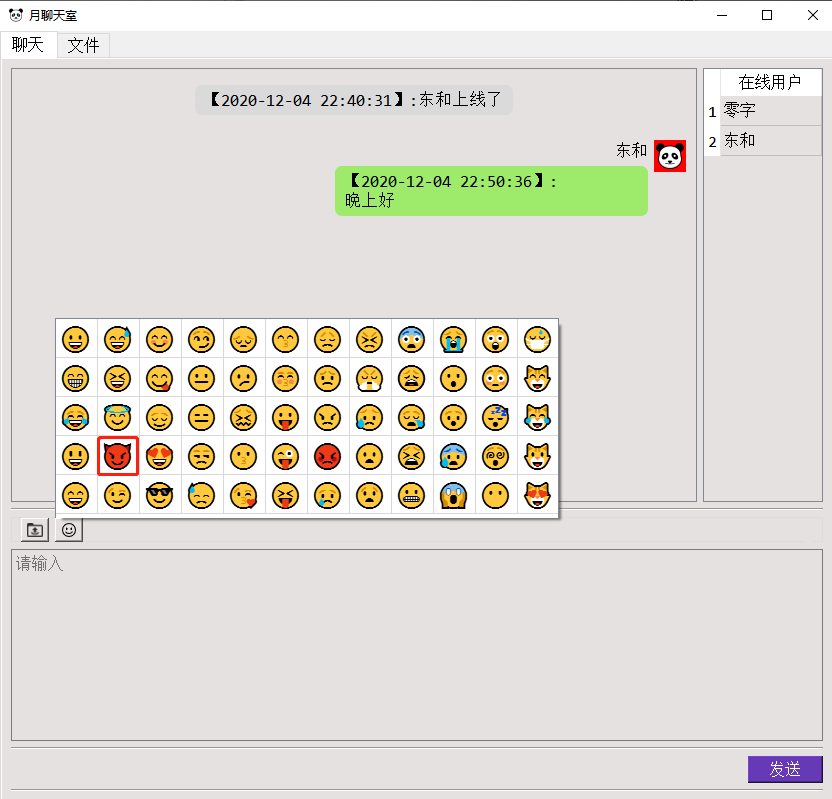


图5.2.3 聊天表情选择面板

**5.3 文件**

**5.3.1 获取文件列表**

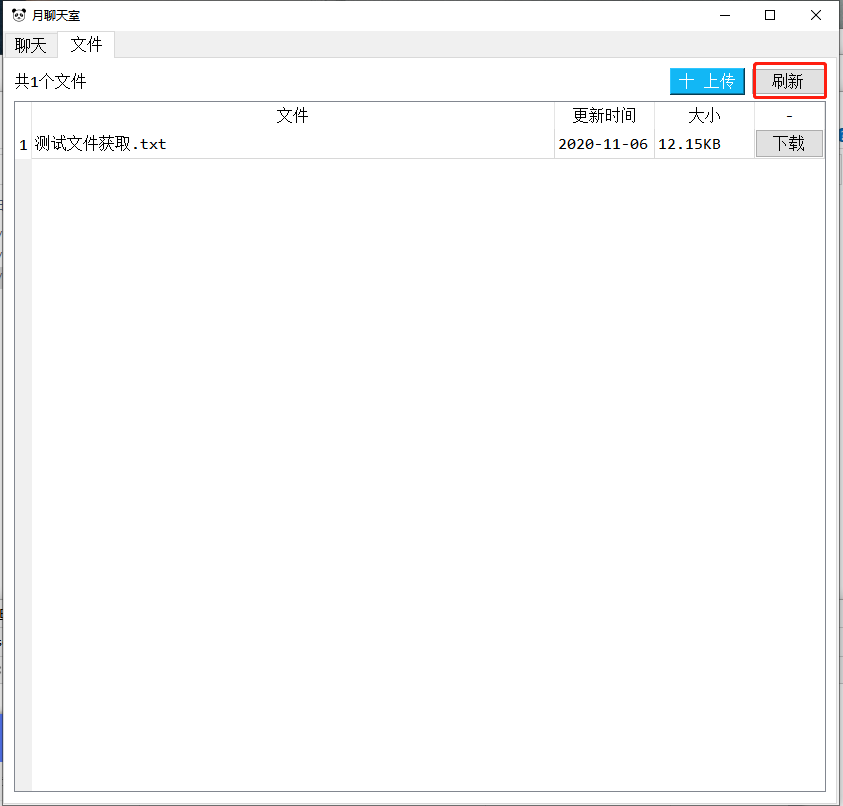


图5.3.1 刷新文件列表

**5.3.2 文件上传**

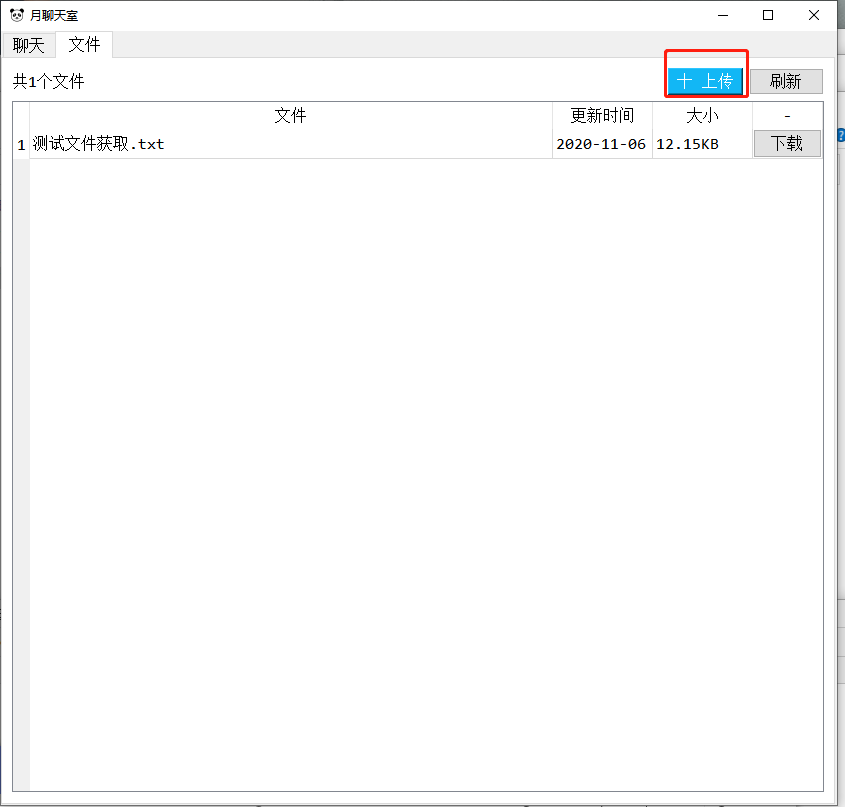


图5.3.2 上传文件按钮

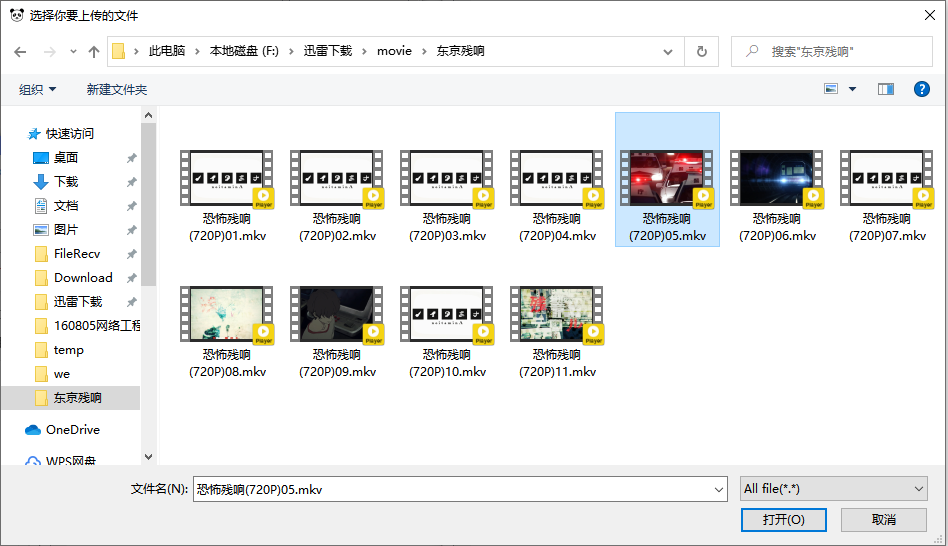


图5.3.3 选择要上传的文件

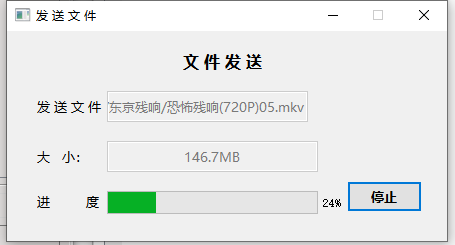


图5.3.4 文件上传进度

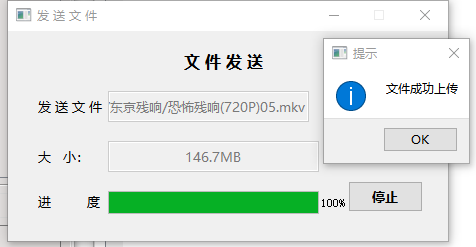


图5.3.5 文件上传结束

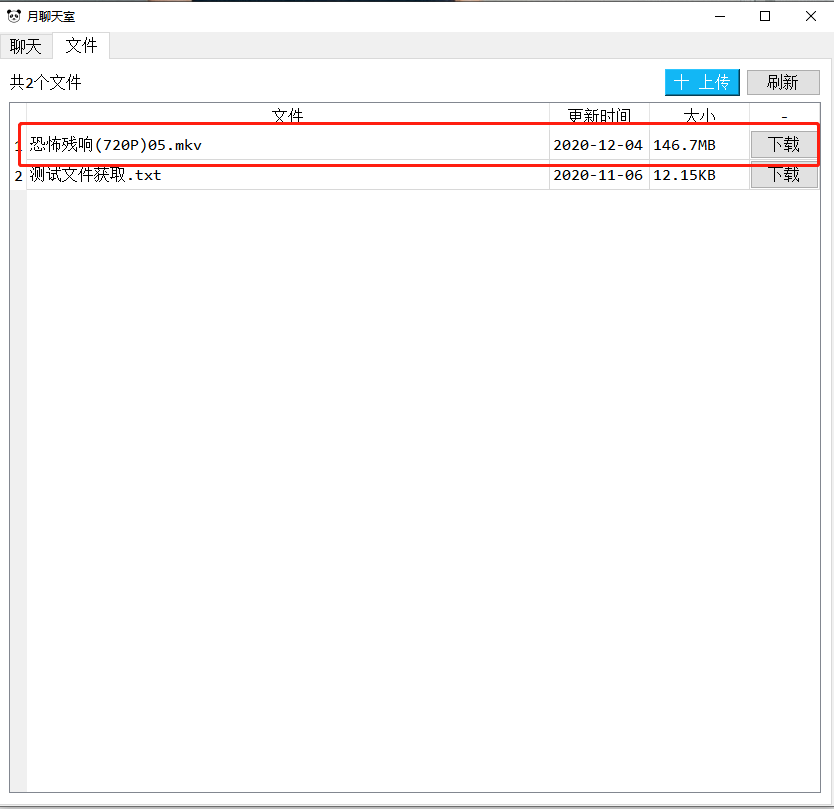


图5.3.6 文件上传后列表变化

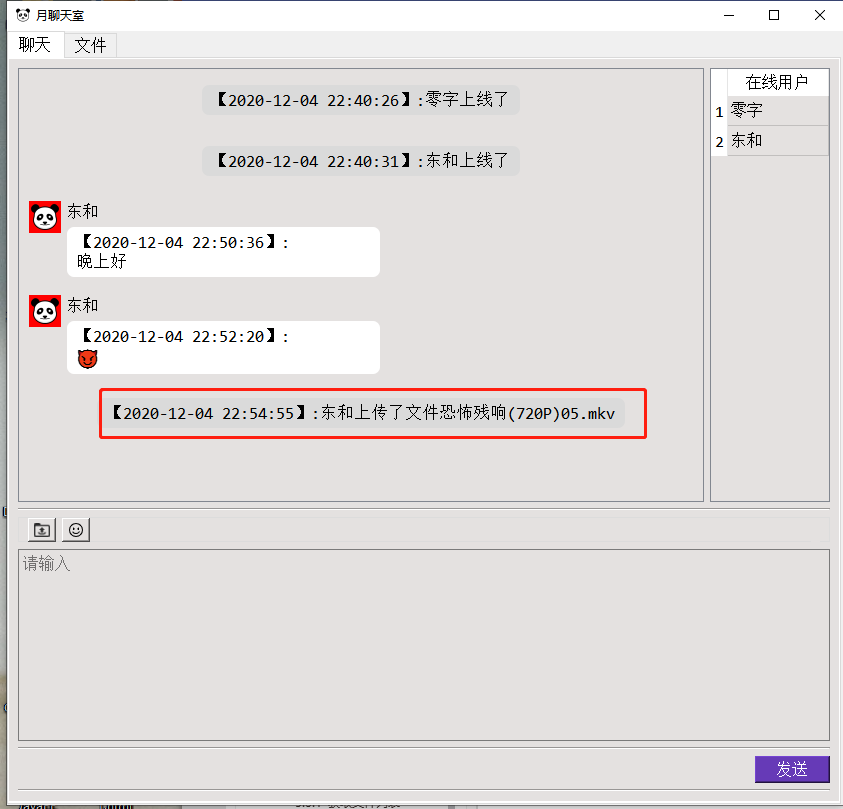


图5.3.7 文件上传通告

**5.3.3 文件下载**

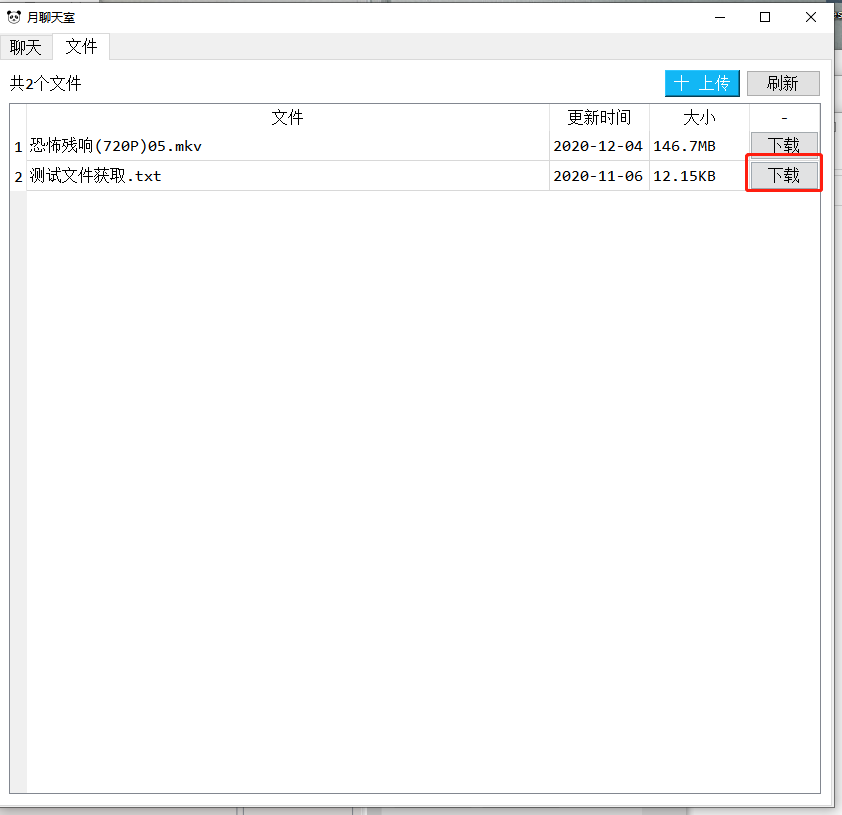


图5.3.8 选择要下载的文件

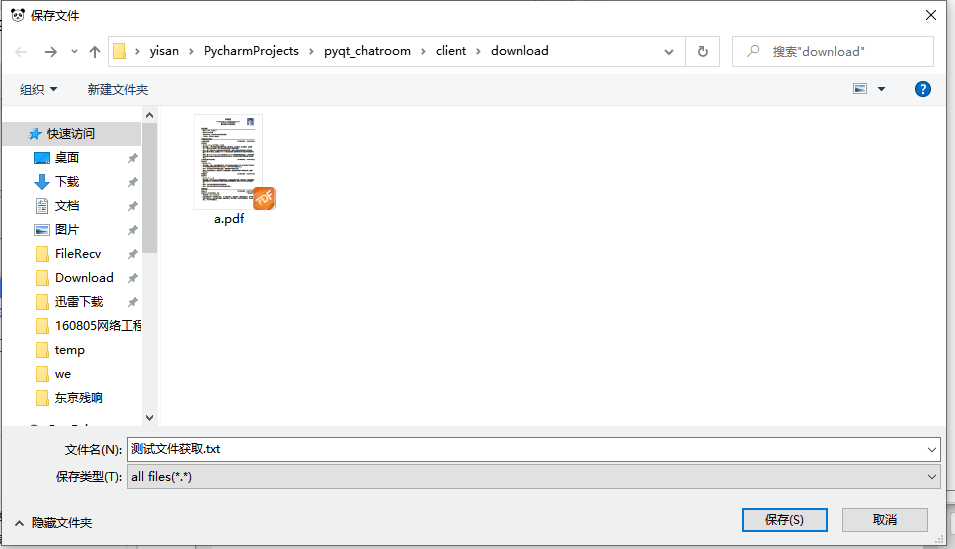


图5.3.9 选择要下载文件的本地保存路径

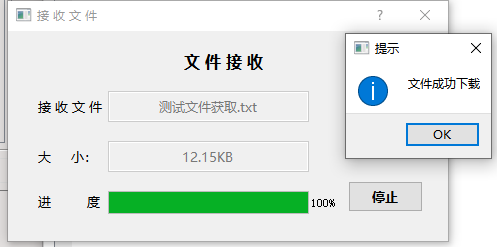


图5.3.10 文件下载完成

**5.4 下线/退出**

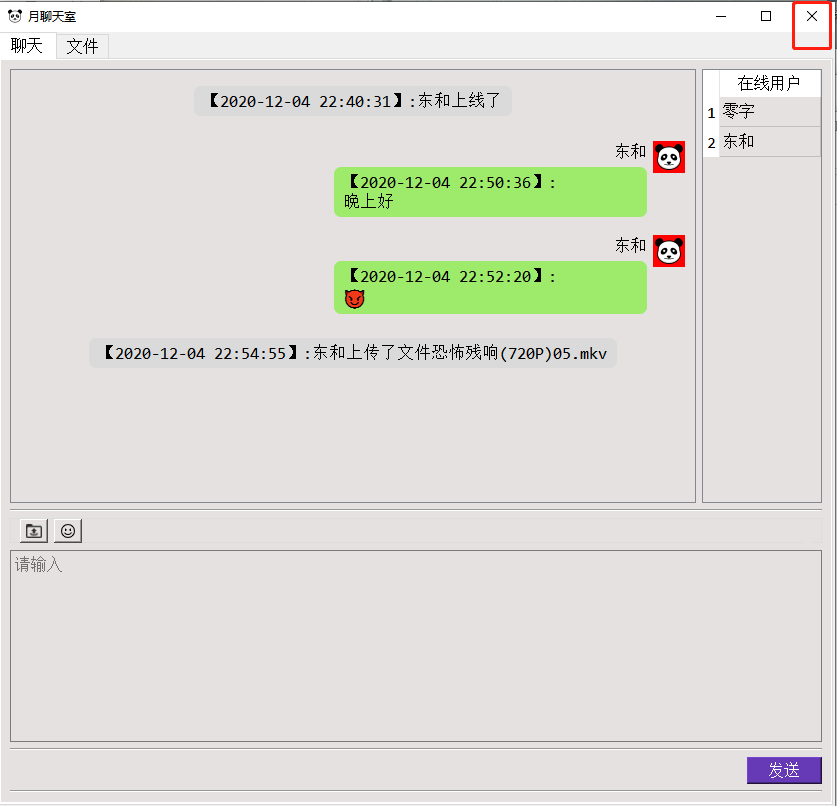


图5.4.1 选择下线

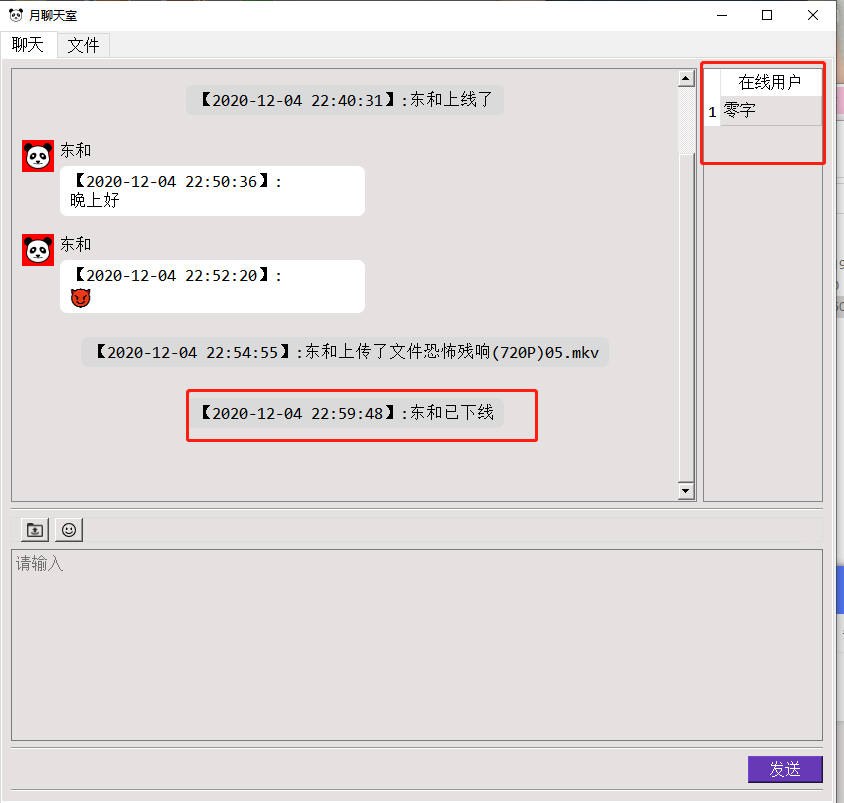


图5.4.2 通告其他人下线

**结论**

本例描述的是一个基于python语言完成的私人聊天室，并且TCP的可靠性，面向连接特点，在本例中，客户端与服务器之间的通信是基于TCP协议的。服务器可以通过TCP套接字实时掌握与客户端之间的通信情况。 而我负责的是整个聊天室的UI设计，整个聊天室UI分为登录跟聊天以及文件上传下载界面。首先设计的登录页面设计，需要用户输入服务器IP跟用户名。聊天界面又细分为聊天/文件切换标签页、聊天内容展示区 、在线用户展示列表 、文件上传、表情符号面板、聊天内容编辑区。文件上传下载界面有显示文件名，文件大小以及上传下载进度条功能，以上我是用QT designer完成的。在做这个聊天室的过程当中，我感受到了自己的很多不足——编程的基础不够扎实，很多都需要查资料。在debug过程中，也产生了很多错误。有些通过查资料一起解决了，再不会的就请教同班的同学。经过这一次的课程设计，我学习到了许多技术和学习方法，进步很大。我深刻的认识到自己的知识面和技术熟练度的不足，同时也确立了后续的学习目标。在设计过程中，我也感受到了做事不能过于急躁，确定好方案一步一步地向前推进就好，面对困难也要保持冷静，太过着急只会扰乱自己的步伐。本次设计当然也存在着不足，没能达到最好的效果，不过相信在以后的学习中，我会学习到更多知识，希望有机会将我们的聊天室完善得更好。

总的来说，完成这次课程设计对我意味很大。今后我也会不断尝试用自己的能力解决问题，写出有用的系统。只有不断实践，不断在实践中吸收经验教训，自己的能力才会得到提升。最后，谢谢老师负责任的教导！！！

**参考文献**

1. 陆文周.Qt5开发及实例[M].北京:电子工业出版社,2019-04.
2. 王硕,孙洋洋.PyQt 5快速开发与实战[M].北京:电子工业出版社,2017-10.
3. (美)Wesley Chun.Python核心编程[M].北京:人民邮电出版社,2016-06