聊天室开发文档

一、工作模式

每个用户通过 TCP 长连接与服务器通信。用户使用软件的过程包括连接、登录/注册、使用、登出（断开连接）。

二、数据包协议

每一个数据包由包类型、包长度与包体三部分组成。数据包结构如图 1 所示。

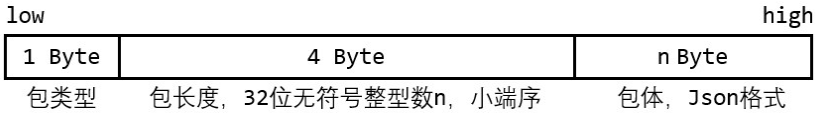

图 1 数据包结构

当前支持的数据包类型如表 1 所示。

表 1 数据包类型

包类型代码	包类型	包体字段名	字段含义	字段备注
NULL	空包	无包体		
REQ_LOGIN	登录请求	username	用户名	UTF8 字符串
		password	密码	
REQ_REGIST	注册请求	同“登录请求“		
REQ_SET_INFO	设置用户信息请求	username	用户名	UTF8 字符串
		nickname	昵称	
		age	年龄	
		gender	性别	
		email	电子邮箱	
REQ_QRY_BASIC	查询基本用户信息*	username	查询对象的用户名	UTF8 字符串
REQ_QRY_DETAIL	查询详细用户信息	同“查询基本用户信息“		
MSG_TXT	文本消息	sender	发送者用户名	UTF8 字符串
		content	消息内容	
		send_time	发送时间**	
MSG_SYS	系统消息	同“文本消息”		
MSG_IMG	图片消息	同“文本消息”		见备注***
DATA_BASIC	基本用户信息	username	用户名	UTF8 字符串
		nickname	昵称	
DATA_DETAIL	详细用户信息	同“设置用户信息请求”		
		created	用户创建时间**	UTF8 字符串

RET_DUP_USR	注册失败，用户名已存在	无包体
RET_DUP_LOGIN	登录失败，用户已登录	无包体
RET_WRONG_PWD	登录失败，密码错误	无包体
RET_WRONG_USR	登录失败，用户不存在	无包体
RET_LOGIN_SUC	登录/注册成功	无包体
RET_SET_INFO_SUC	设置用户信息成功	无包体

* 基本用户信息用于在客户端显示消息

** 发送时间/用户创建时间格式：yyyy-MM-dd hh:mm:ss

*** 包体 content 字段内容为：图片的 JPG 格式二进制流对应的 Base64 字符串

在服务端，文件 Server/chatroom/chat_packet.go 定义了数据包类型代码，并提供了收发数据包的函数 RcvPacket 与 SendPacket。

在客户端，头文件 Client/QChatClient/packet_code.h 定义了数据包类型代码，头文件 Client/QChatClient/packet.h 提供了数据包类 Packet，其中包括了收发数据包的成员函数 receive 与 send。

二、服务器端设计

服务器端的设计参考了 *The Go Programming Language* 8.10. Example: Chat Server。

服务器端主要由四个函数构成：Start、handlePublic、handleClient、writeClient。

1. 登录/注册

handlePublic 函数运行在一个 sub goroutine，其中维护着当前在线的所有用户的用户名。handlePublic 函数负责处理登录/注册，它通过两个分别名为 login 和 regist 的 channel 接收登录和注册的请求。

Start 函数运行在 main goroutine，负责监听用户的连接请求，每次接受一个连接后，新建一个 goroutine，在该 goroutine 中运行 handleClient 函数处理该连接。

handleClient 函数接收用户的登录/注册请求，并通过 channel 将请求传递给 handlePublic 函数进行处理，同时提供一个名为 ret 的 channel 接收 handlePublic 函数回传的处理结果。handleClient 收到处理结果后，将结果反馈给用户。若处理结果表示登录/注册失败，handleClient 函数会继续等待下一次登录/注册请求。

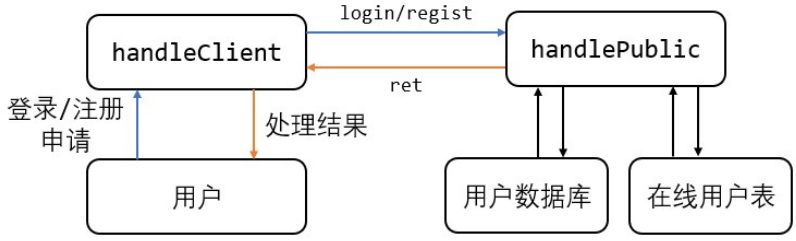

图 2 登录/注册

2. writeClient

在某个用户登录/注册成功后，他的 handleClient 函数会新建一个 goroutine，在其中运行 writeClient 函数。writeClient 函数通过一个名为 cliChan 的 channel 接收数据包，并将数据包传输给用户。

3. 广播数据包

handlePublic 函数通过一个名为 broadcast 的 channel 接收需要广播的数据包。

hanlePublic 函数中维护的在线用户表事实上是一个 map，键为用户名，值为该用户对应的 cliChan。handleClient 函数在提交登录/注册申请的同时，会将用户对应的 cliChan 也传递给 handlePublic 函数。

broadcast 中被写入数据包后，handlePublic 函数将数据包写入所有在线用户的 cliChan。

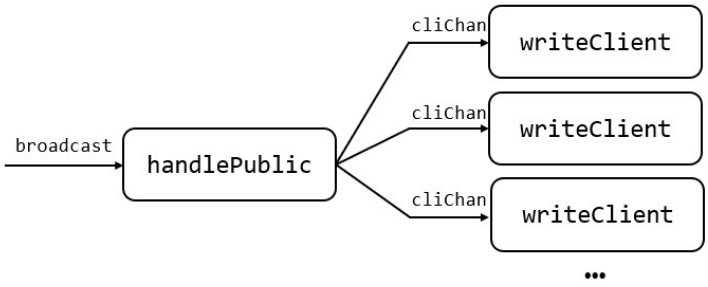

图 3 广播数据包

cliChan 被设置为 buffered channel，一定程度上防止了某些 writeClient 工作缓慢引起的 handlePublic 长时间阻塞。

4. handleClient

当某个用户登录/注册成功后，他的 handleClient 函数负责接收用户发来的数据包。对于需要广播的数据包，将其写入 broadcast。对于查询信息的请求，直接在 handleClient 函数中处理，并将查询结果写入用户的 cliChan。

5．登出

用户断开连接视作登出。登出操作需要修改在线用户表，handlePublic 函数通过一个名为 logout 的 channel 接收登出信号。当某个用户的 handleClient 函数发现连接断开后，向 logout 写入该用户的用户名。

当服务器收到 n 个 TCP 连接时，至多有 2n+2 个 goroutine 在同时运作，其中 Start 占用一个，handlePublic 占用一个，每个 TCP 连接的 handleClient 各占用一个，每个登录/注册成功用户的 writeClient 各占用一个。

所有用户的广播需求被集中到 handlePublic 函数，在线用户表仅可在 handlePublic 函数所在 goroutine 中被调用，避免了 mutex 的使用。登录/注册请求需要访问在线用户表，所以也被集中到 handlePublic 函数。

服务器端的具体实现参见文件 Server/chatroom/chat_server.go，文件 Server/chatroom/chat_userdb.go 提供了用户数据库操作所需的函数。

三、客户端的设计

1. 数据包类 Packet

Packet 类提供了收发数据包所需的函数 receive 和 send。数据包收发采用阻塞的方式，receive 函数在接收到一个完整的包后才退出，send 函数在发送一个完整的包后才退

出。

2. 登录界面 LoginWindow

登录界面是用户打开软件后首先显示的界面。在 LoginWindow 类构造函数中，创建了 QTcpSocket 对象。当用户点击菜单栏 Server-Connect 后，QTcpSocket 对象连接到服务器。连接之后，用户可以进行登录/注册操作。连接服务器与登录/注册所涉及的网络操作会阻塞 UI。

3. 聊天界面 ChatWindow

登录成功后，转至聊天界面 ChatWindow。主界面中的消息显示框为 HTML 浏览器，利用 HTML 实现了彩色文字、图像与超链接的显示。

在用户使用过程中，聊天界面可以产生两种子界面 PersonalInfoWindow 与 UserInfoWindow,分别用于本用户信息的查阅/修改与其他用户信息的查阅。

4. 后台服务 SocketHandler

在 ChatWindow 类的构造函数中，一个 SocketHandler 对象被创建，并获得了此前创建的 QTcpSocket 对象的指针。此后，所有的数据包收发将由 SocketHandler 对象负责，通过 QThread::moveToThread 函数，该对象被移至一个新的线程，以避免网络操作阻塞 UI。

QTcpSocket 对象发出 readyRead 信号后，SocketHandler 对象的 receivePacket 槽被激活，接收数据包直至 QTcpSocket:: bytesAvailable 返回 0，分析收到的数据包的类型，并通过信号将数据包转发给聊天界面或其子界面对应的槽函数，执行数据包的处理。

当用户需要向服务器发送数据包时，聊天界面或其子界面会将数据包通过 packetToSend 信号传递给 SocketHandler 对象，之后 SocketHandler 对象会将数据包发送给服务器。