

Building a TCP chat server using Go

As a final Project I decided to build a chat server using Go. Go enables multiple clients to connect to a single server through TCP to enable the real time sending and receiving of messages through a single connection. There are Five main packages in this project client, command, main, room, and server and let's take a closer look at each:

Client:

This package is responsible for keeping user info, the TCP connection, and parsing the client's input and then sending that to the server, the basic struct for the client function is:

```
type client struct {  
    conn    net.Conn  
    nick    string  
    room    *room  
    commands chan<- command  
}
```

Where conn is the client TCP connection, nick is the users assigned username as a string, room which is the pointer to the current chat room, and commands which contain the channel of incoming commands to be sent to the server for processing.

Command:

The command package simply contains the struct for the different commands that will be used in the chat server. The struct for this function is:

```
type command struct {  
    id      commandID  
    client  *client  
    args   []string  
}
```

Where id is the unique command type id, client is the sender of the information to the server And argos is the string to be sent from the clients message to the server

Main:

The main package contains the functionality to build a TCP server by initializing a TCP which listens for new messages, when a message is detected it sends that to the other clients connected to the server.

```
func main() {
    s := newServer()
    go s.run()

    listener, err := net.Listen("tcp", ":8888")
    if err != nil {
        log.Fatalf("unable to start server: %s", err.Error())
    }

    defer listener.Close()
    log.Printf("server started on :8888")

    for {
        conn, err := listener.Accept()
        if err != nil {
            log.Printf("failed to accept connection: %s", err.Error())
            continue
        }

        go s.newClient(conn)
    }
}
```

Room:

The room package is responsible for holding the names of rooms and the members of rooms. The struct for the room package is:

```
type room struct {
    name      string
    members   map[net.Addr]*client
}
```

Where name is the string for the room name and members is the list of members for that room using a client remove address.

Server:

Finally the Server package is responsible for executing the different functions of the program as it operates. This is achieved using a switch case to determine what to do and when:

```
func (s *server) run() {
    for cmd := range s.commands {
        switch cmd.id {
            case CMD_NICK:
                s.nick(cmd.client, cmd.args[1])
            case CMD_JOIN:
                s.join(cmd.client, cmd.args[1])
            case CMD_ROOMS:
                s.listRooms(cmd.client)
            case CMD_MSG:
                s.msg(cmd.client, cmd.args)
            case CMD_QUIT:
                s.quit(cmd.client)
        }
    }
}
```

Working Example:

Go-Chat-Server

Steps to build the chat server

In terminal 1:

```
go build .
./Go-Chat-Server
```

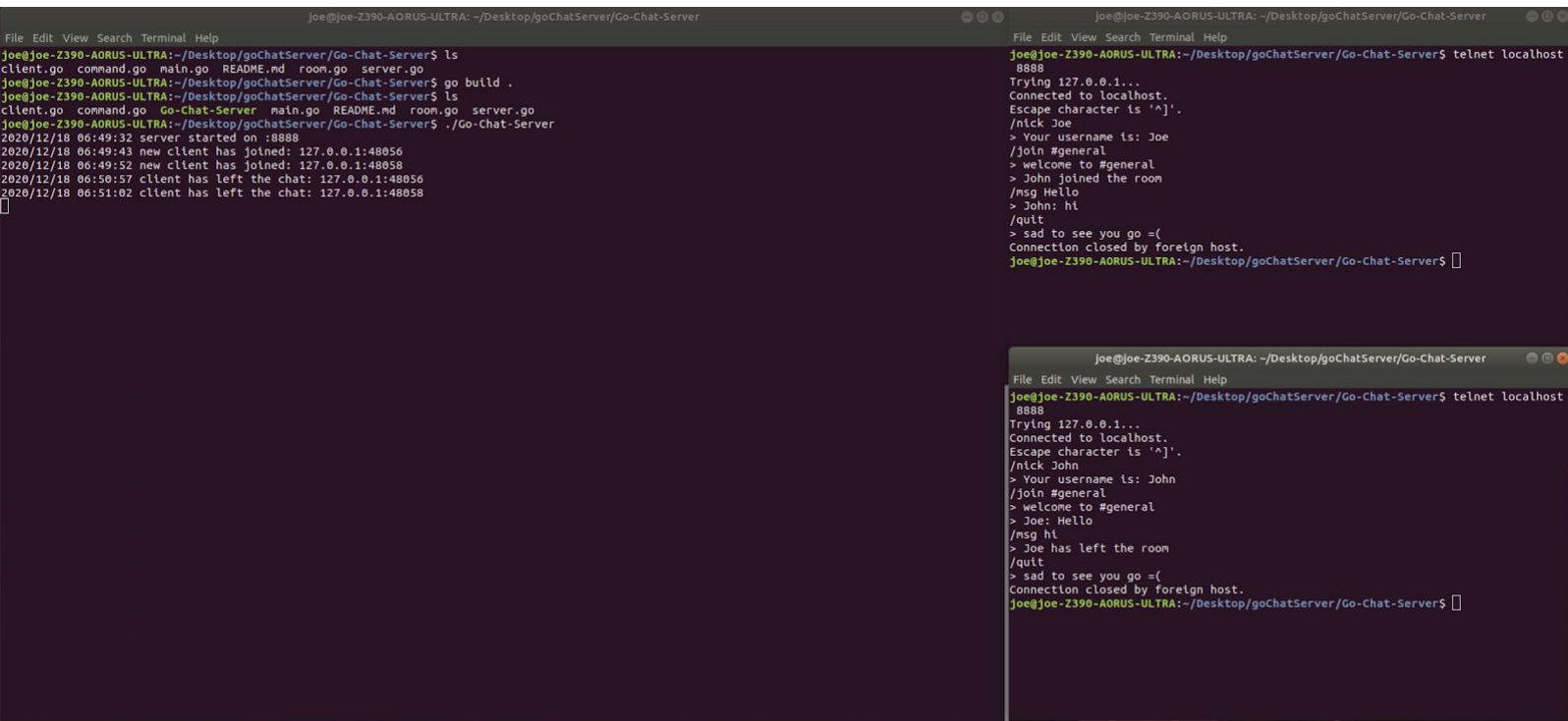
In terminal 2:

```
telnet localhost 8888
/nick <desired_name>
/join #general
/msg Hello
```

In Terminal 3:

```
telnet localhost 8888
/nick <desired_name>
/join #general
/msg Hello
```

Github Repository: <https://github.com/JPelligra/Go-Chat-Server>



The image shows two terminal windows side-by-side, both titled 'Joe@joe-Z390-AORUS-ULTRA: ~/Desktop/goChatServer/Go-Chat-Server'. The left window shows the server being built and started. The right window shows a telnet client connecting to the server and interacting with it.

```
Joe@joe-Z390-AORUS-ULTRA:~/Desktop/goChatServer/Go-Chat-Server$ ls
client.go  command.go  main.go  README.md  room.go  server.go
Joe@joe-Z390-AORUS-ULTRA:~/Desktop/goChatServer/Go-Chat-Server$ go build .
Joe@joe-Z390-AORUS-ULTRA:~/Desktop/goChatServer/Go-Chat-Server$ ls
client.go  command.go  Go-Chat-Server  main.go  README.md  room.go  server.go
Joe@joe-Z390-AORUS-ULTRA:~/Desktop/goChatServer/Go-Chat-Server$ ./Go-Chat-Server
2020/12/18 06:49:32 server started on :8888
2020/12/18 06:49:43 new client has joined: 127.0.0.1:48056
2020/12/18 06:49:52 new client has joined: 127.0.0.1:48058
2020/12/18 06:50:57 client has left the chat: 127.0.0.1:48056
2020/12/18 06:51:02 client has left the chat: 127.0.0.1:48058
```

```
Joe@joe-Z390-AORUS-ULTRA:~/Desktop/goChatServer/Go-Chat-Server$ telnet localhost 8888
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
/nick Joe
> Your username is: Joe
/join #general
> welcome to #general
> John joined the room
/msg Hello
> John: hi
/quit
> sad to see you go =(
Connection closed by foreign host.
Joe@joe-Z390-AORUS-ULTRA:~/Desktop/goChatServer/Go-Chat-Server$
```

```
Joe@joe-Z390-AORUS-ULTRA:~/Desktop/goChatServer/Go-Chat-Server$ telnet localhost 8888
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
/nick John
> Your username is: John
/join #general
> welcome to #general
> Joe: Hello
/msg hi
> Joe has left the room
/quit
> sad to see you go =(
Connection closed by foreign host.
Joe@joe-Z390-AORUS-ULTRA:~/Desktop/goChatServer/Go-Chat-Server$
```

References:

This project was made possible by a tutorial on how to use got o set up a TCP server:

<https://www.youtube.com/watch?v=Sphme0BqJiY>