Started: 11/4/21
Last updated: 11/11/21

Port 4200

This is the port that you will connect to when implementing your server.

Message Requirements

Requirements

A message sent will consist of the following parameters in this order:

1. Message control type/ status code (byte)
2. Quantity (of items returned) (Incrementing for people)
3. User id (int)
4. Payload/message length (int)
5. "Payload/message" UTF8 (byte[])

Control Types

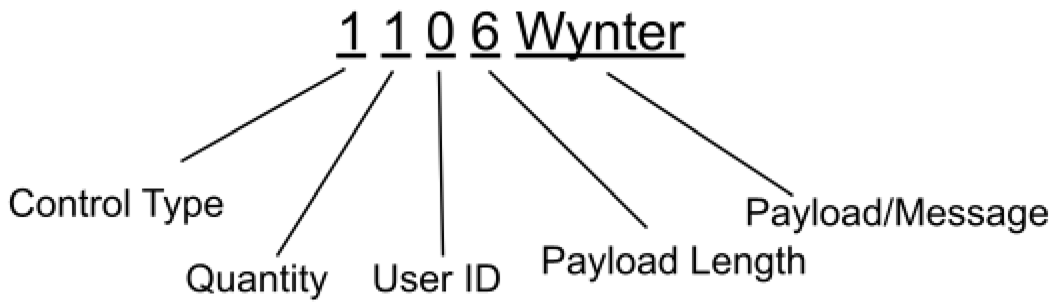
A message sent by the client can have one of the following control types:

- 0 = user list
- 1 = join (payload is user's name)
- 2 = leave
- 254 = direct message
- 255 = broadcast message

Message Example

Below is an example of what a message a client would send. Below are also examples of applicable status codes that a client can use.

****Double click image to edit!****



Status Code Examples

- 0 - user list [0 n(number of user's) 0 {user id} x "user's id with length n"]
- 1 - join [1 1 0 0 3 "Tim"] - the server responds with a
- 2 - leave [2 1 0 {user_id} 0 ""]
- 255 - broadcast message [255 1 0 n "message"]

Giving User IDs

User IDs must be unique from the currently online and created users' ids. The id's reset when the server resets. They are represented by integers and incremented by 1. The id's must begin at 2 as 0 is reserved for broadcasting and 1 is used for identifying the user sending the request.

User IDs → start at 2 (arbitrary could start at any integer > 1 and < 2147483647) and increment by one for each new user that joins.

Joining a Chat Room:

Broadcasting New Users

The server will also broadcast a newly connected user to existing connected users using a *join* response. This can be done for each individual user or for multiple users at once.

Control byte: 0 (*new user*)
Quantity: The quantity of the total users included in the response.
User Number: The user number.
Length: The length, in bytes, of the UTF-8 encoded username string.
Payload: The UTF-8 encoded username string.

Broadcasting New Users Examples:

This is an example of the broadcast when John joins with an id of 5.

1. 0 1 5 4 John

Client's Join Request to Server

The client's *join* request to the server must follow this format:

Control byte: 1 (*join*)
Quantity: 1 (single username)
User Number: Set this field to 0
Length: The length, in bytes, of the UTF-8 encoded username string.
Payload: The UTF-8 encoded username string.

Client's Join Request Examples:

This is an example of the client's join request when 3 clients join.

1. 1 1 0 4 Eden
2. 1 1 0 6 Wynter
3. 1 1 0 2 CK

Server's Response to Client

The server will respond with a *user list* response. The response *must* include the connecting user. The first user in the list is the connecting user. Subsequent users' order does not matter.

Control byte: 0 (*user list*)
Quantity: The quantity of the total users included in the response.
For each user:
User Number: The user number.
Length: The length, in bytes, of the UTF-8 encoded username string.
Payload: The UTF-8 encoded username string.

Server's Response Examples:

This is an example of the server's response when 3 clients join.

1. 0 1 2 4 Eden
2. 0 2 3 6 Wynter [24 Eden]
3. 0 3 4 2 CK [24Eden36Wynter]

Leaving a Chat Room

Client's Leave Request to Server

The client's *leave* request to the server must follow this format.

Control byte: Set this field to 2 (*leave*)
Quantity: Set this field to 1 (single request)
User Number: Set this field to 0
Length: Set this field to the length of the user_id
Payload: The user_id
Control byte: 2 (*leave*)
Quantity: set this field to 1
User Number: set this field to the User ID of the user
Length: The length, in bytes, of the UTF-8 encoded username and termination message.
Payload: The UTF-8 encoded username and termination message.

Client to Server Example:

Where 0 represents the broadcast. 2 represents the characters in the username. The payload represents the username.

2 1 0 2 CK

Broadcasting to Current Users About Leaving User

The server will also broadcast the leaving user to other connected users using a *leave* response. This can be done for each individual user and for multiple users at once.

Control byte: 2 (*leave*)
Quantity: The quantity of the total users included in the response.
User Number: The user number.
Length: The length, in bytes, of the UTF-8 encoded username string.
Payload: The UTF-8 encoded username string.

Server to All Connected Clients Example:

Where 4 represents the leaving user_id. 2 represents the characters in the username. The payload represents the username. This is sent to all connected clients.

2 1 4 2 CK

Direct & Broadcast Messaging

This section does not include broadcasting new and leaving users since their connection and disconnection will be broadcasted once they have joined or left the chatroom.

Broadcasting a Message

Control byte: 255 (*broadcast message*)
Quantity: 1 (*single request*)
User Number: user_id of the person sending the message
Length: The length, in bytes, of the message.
Payload: The message.

Client to Server Example:

Where 0 is the destination, and is broadcasted to all users.

255 1 0 5 Hello

Server to Client Example:

Where 4 represents the sender. In previous examples, it is CK.

255 1 4 5 Hello

Sending a Direct Message (Client to Server)

The client sends a message visible to one other user or to the server.

Control byte: 254 (*broadcast message*)
Quantity: 1 (*single request*)
User Number: the user's id ex: 10 (***the destination***)
Length: The length, in bytes, of the message.
Payload: The UTF-8 encoded message string.

Client to Server Example:

Where 5 is the destination. In previous examples, this corresponds to John.

255 1 5 5 Hello

Receiving a Direct Message (Server to Client)

The client receives a message from the server only visible to them. This could include the list of online users

Control byte: 254 (*message*)
Quantity: 1 (*single request*)
User Number: The user's id ex: 10 (***the sender***)
Length: The length, in bytes, of the message.
Payload: The UTF-8 encoded message string

Server to Client Example:

Where 4 represents the sender. In previous examples, it is CK.

255 1 4 5 Hello

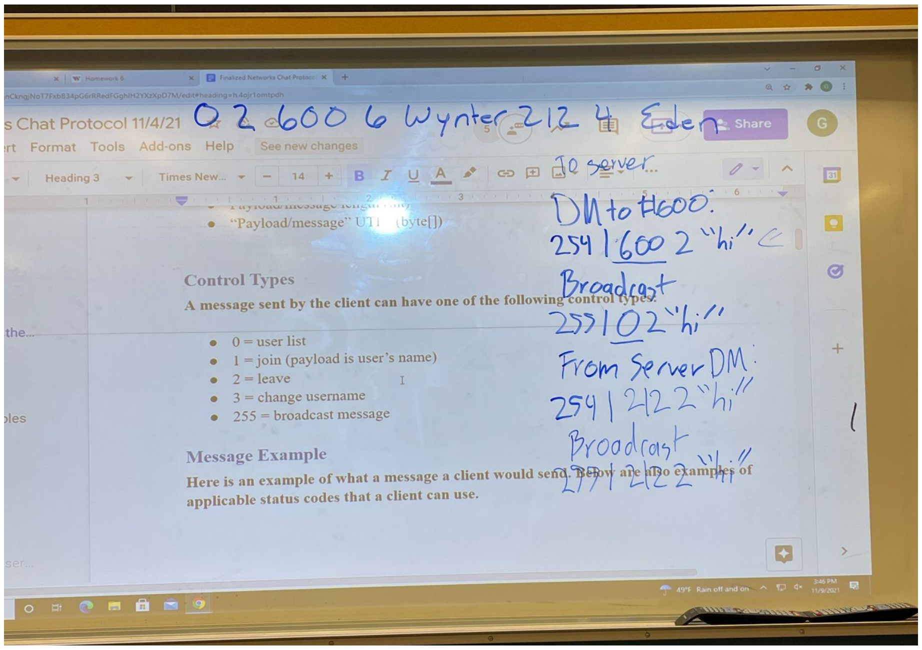
Receiving a Broadcast Message from Server

All of the clients receive a message from the server only visible to them.

Control byte: 255 (*message*)

Quantity: 1 (*single request*)
User Number: The user's id ex: 10 (***the sender***)
Length: The length, in bytes, of the message.
Payload: The UTF-8 encoded message string

Examples in Class



s Chat Protocol 11/4/21

0 2 600 6 Wynter 212 4 Eden

To server...

DM to #600:

254 1 600 2 "hi"

Broadcast

255 1 0 2 "hi"

From Server DM:

254 1 212 2 "hi"

Broadcast

255 1 212 2 "hi"

- "Payload/message" UTF (byte[])

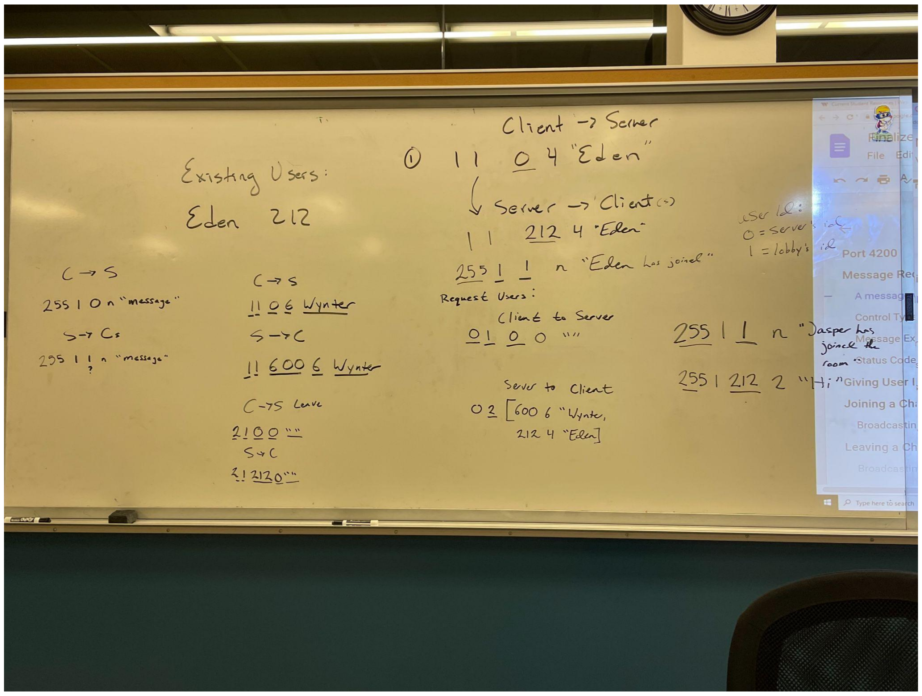
Control Types

A message sent by the client can have one of the following control types:

- 0 = user list
- 1 = join (payload is user's name)
- 2 = leave
- 3 = change username
- 255 = broadcast message

Message Example

Here is an example of what a message a client would send. Below are also examples of applicable status codes that a client can use.



Existing Users:

Eden 212

C → S

255 1 0 n "message"

S → Cs

255 1 1 n "message"

C → S

11 0 6 Wynter

S → C

11 600 6 Wynter

C → S Leave

2 1 0 0 ""

S → C

2 1 212 0 ""

Client → Server

① 1 1 0 4 "Eden"

Server → Client(s)

1 1 212 4 "Eden"

255 1 1 n "Eden has joined"

Request Users:

Client to Server

0 1 0 0 ""

Server to Client

0 2 [600 6 "Wynter,
212 4 "Eden]

user id:
0 = server's id
1 = lobby's id

255 1 1 n "Jasper has joined the room"

255 1 212 2 "Hi"