CSE 489/589
Programming Assignment 1 Stage 2 Report
Text Chat Application

Notes: (IMPORTANT)

- ➔ Your submission will **NOT** be graded without submitting this report. It is required to use this report template. Do not add sections for Stage 1, as they will not be graded.
- ➔ One of your group members select <File> - <Make a copy> to make a copy of this report for your group, and share that Google Doc copy with your teammates so that they can also edit it.
- ➔ Report your work in each section (optional). Describe the method you used, the obstacles you met, how you solved them, and the results. You can take screenshots at key points. There are NO hard requirements for your description.
- ➔ For a certain command/event, if you successfully implemented it, **attach the screenshot of the result from the grader (required).** You will get full points if it can pass the corresponding test case of the automated grader.
- ➔ For a certain command/event, if you tried but failed to implement it, properly describe your work. If you can get some points (not full) from the grader, also **attach the screenshot of the result from the grader (required).** We will partially grade it based on the work you did.
- ➔ Do **NOT** claim anything you didn't implement. If you didn't try on a certain command or event, leave that section blank. We will randomly check your code, and if it does not match the work you claimed, you and your group won't get any partial grade score for this WHOLE assignment.
- ➔ This report has 5 bonus points. Bonus points grading will be based on your **description** in each section, including the sections where you got full points from the grader.
- ➔ After you finish, export this report as a PDF file and submit it on the UBLearns along with your tarball. For each group, only one member needs to make the submission.
- ➔ Maximum score for Stage 2: 58 + 5 = 63

1. Group and Contributions

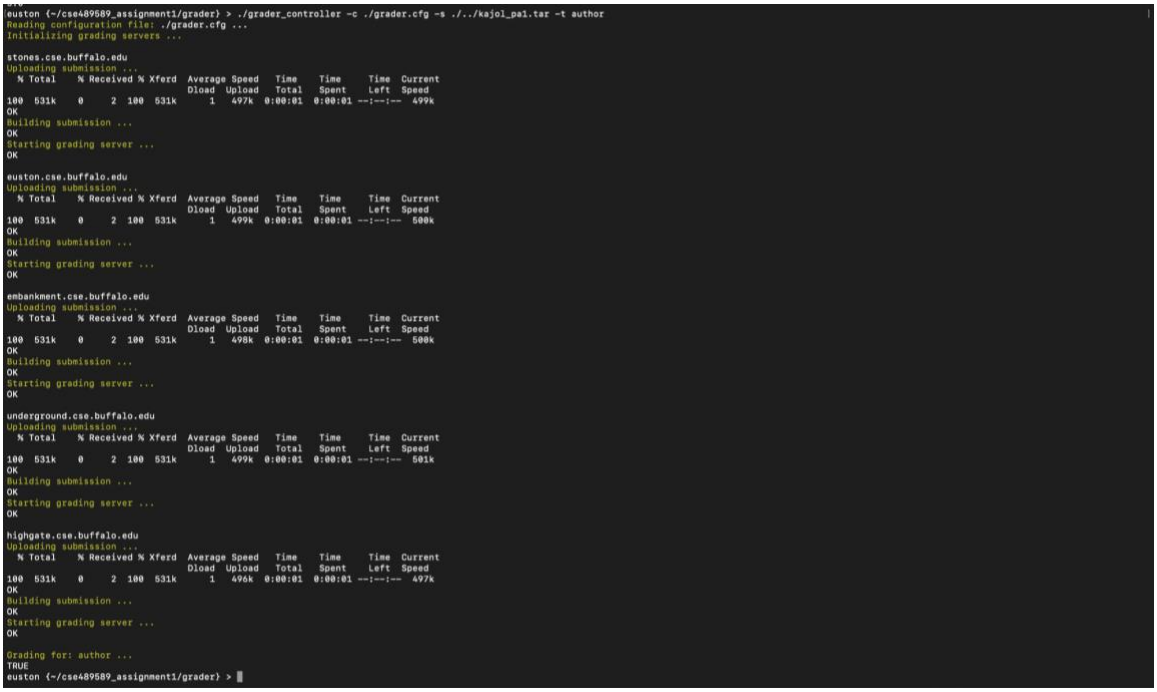
- Name of member 1:
  - UBITName: kajol (Kajol)
  - Contributions Implementation and documentation of respective functionality – Send, Statistics, Blocked, Send Exception Handling, Block Exception Handling, Blocked Exception Handling,

- Name of member 2:
  - UBITName: amehta9 (Aishwarya Mehta)
  - Contributions Implementation and documentation of respective functionality – Broadcast, Block, Unblock, Buffer, Logout, Unblocked Exception Handling

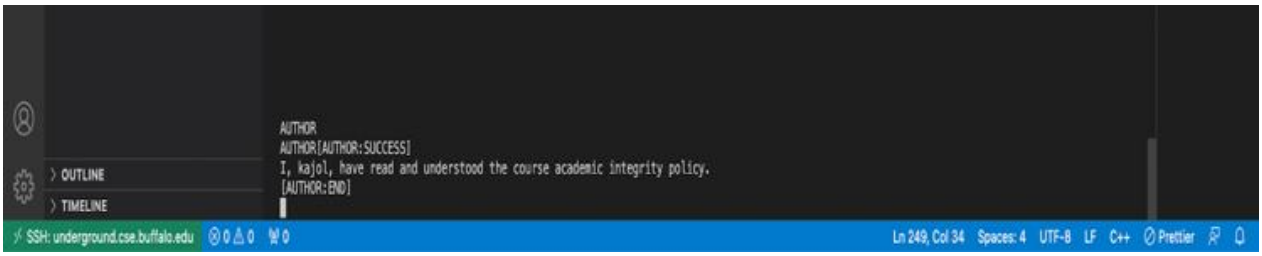
2. Test results

[0.0] AUTHOR (author)

Your submission will not be graded if the AUTHOR command fails to work.
Grader screenshot:

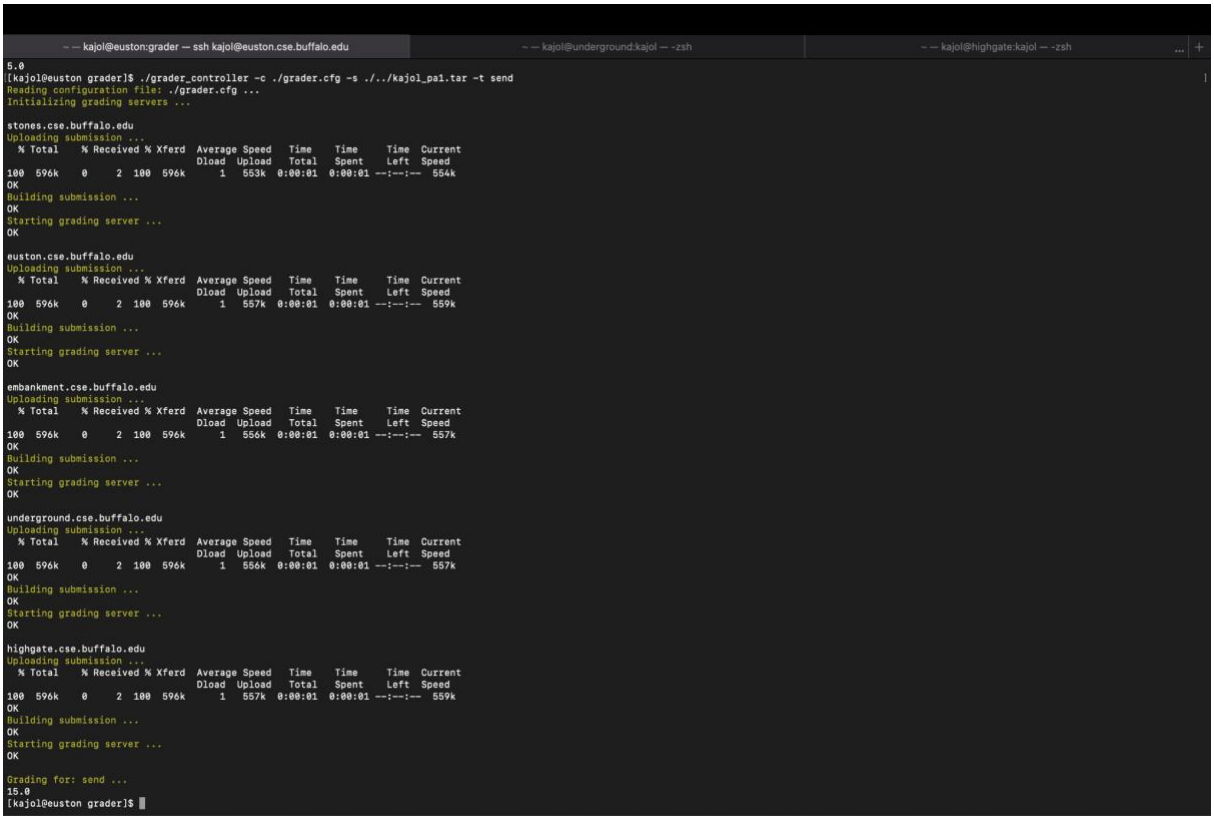


Description: Whether the application is launched as a server or a client, this sequence of commands functions.
INPUT:
AUTHOR
OUTPUT:
I, kajol, have read and understood the course academic integrity policy.



[15.0] SEND (send)

Grader screenshot:

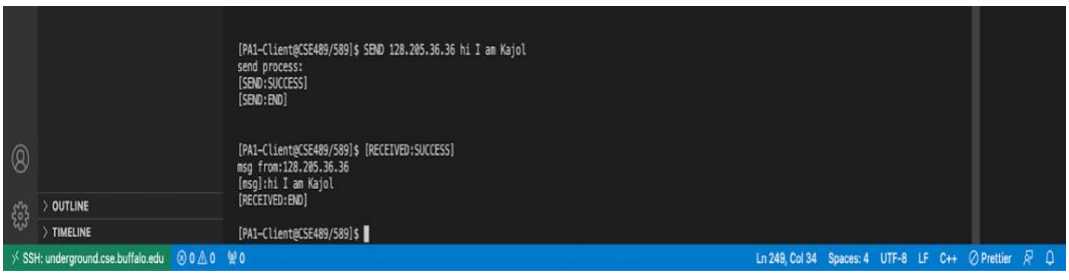


Description:

Send message:

- Send a message to the client, using client-ip as the IP address. The message can only be 256 bytes long.
- It can only contain legal ASCII characters.
- The send function must start sending a message from the specified socket to its peer. Only when the socket is connected should the function send a message.

AT CLIENT END:



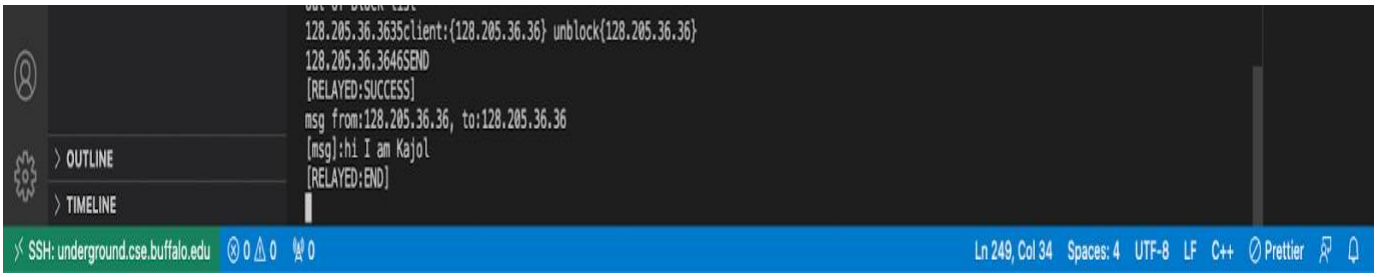
AT SERVER END:

[EVENT]: Message Relayed

Description:
The server relays all messages sent and received between clients. If a client with the IP address: <from-client-ip> sends a message to another client with the IP address: <to-client-ip>, printed the message using the format string:

("msg from:%s, to:%s\n[msg]:%s\n", from-client-ip, to-client-ip, msg)

- If a broadcast message is sent, the value for "to-client-ip" will be 255.255.255.255.
- Used command str: RELAYED for printing/logging reasons.



[EVENT]: Message Received

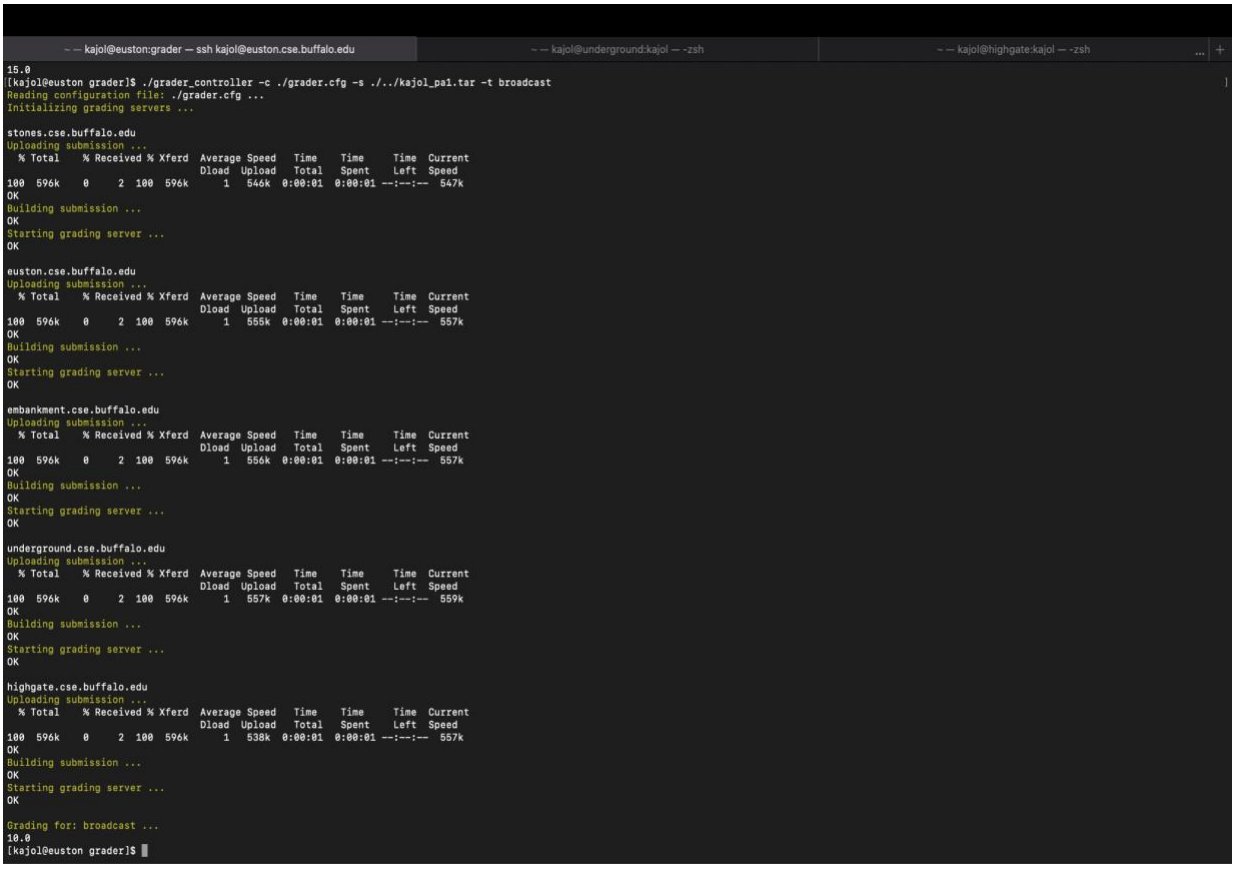
Description:
If a message with the following format string is received from a client with the IP address (client-ip):

("msg from:%s\n[msg]:%s\n", client-ip, msg)

It should be noted that "client-ip" in this case refers to the IP address of the sender, not the relaying server.
Used command str: RECEIVED for printing/logging reasons.

[10.0] BROADCAST (broadcast)

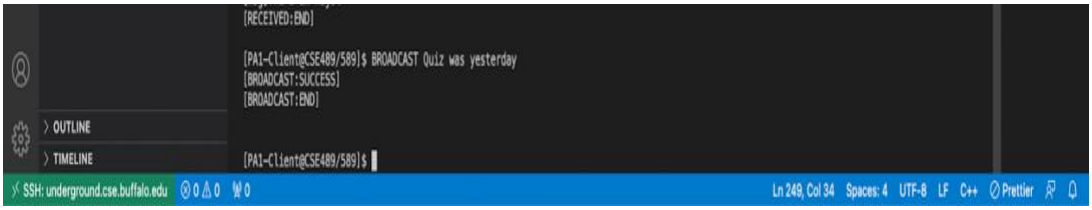
Grader screenshot:



Description:
To all clients who are logged in, message will be sent. Message can only be up to 256 bytes long and must only include legal ASCII characters.

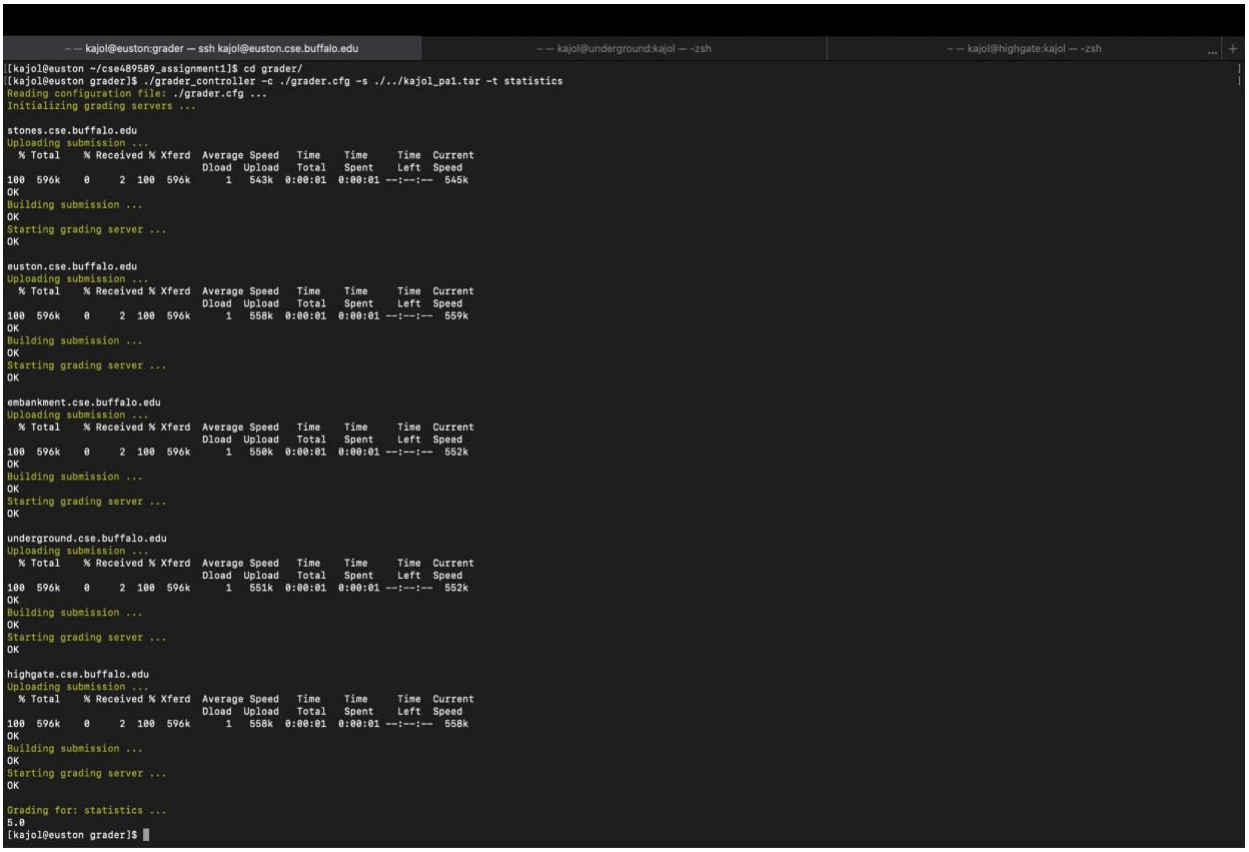
This ought to be a broadcast with server assistance. To identify it as a broadcast, the transmitting client should only send one message to the server. The server then stores/buffers the message for the remaining clients and forwards/relays it to every client who is currently signed in.

The BROADCAST client will not get the exact identical message back.



[5.0] STATISTICS (statistics)

Grader screenshot:



Description:

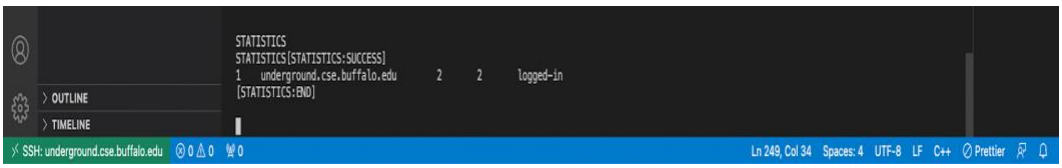
(Describe how you implement the LOGIN and LIST commands)

Show statistics for each client and a numbered list of all clients who have ever logged in to the server (but who have not used the EXIT command). Depending on whether the client is currently logged in or out, the output should show the hostname, #messages-sent, #messages-received, and the current status: logged-in/logged-out, arranged by their listening port numbers, in ascending order.

Used the format string that follows:

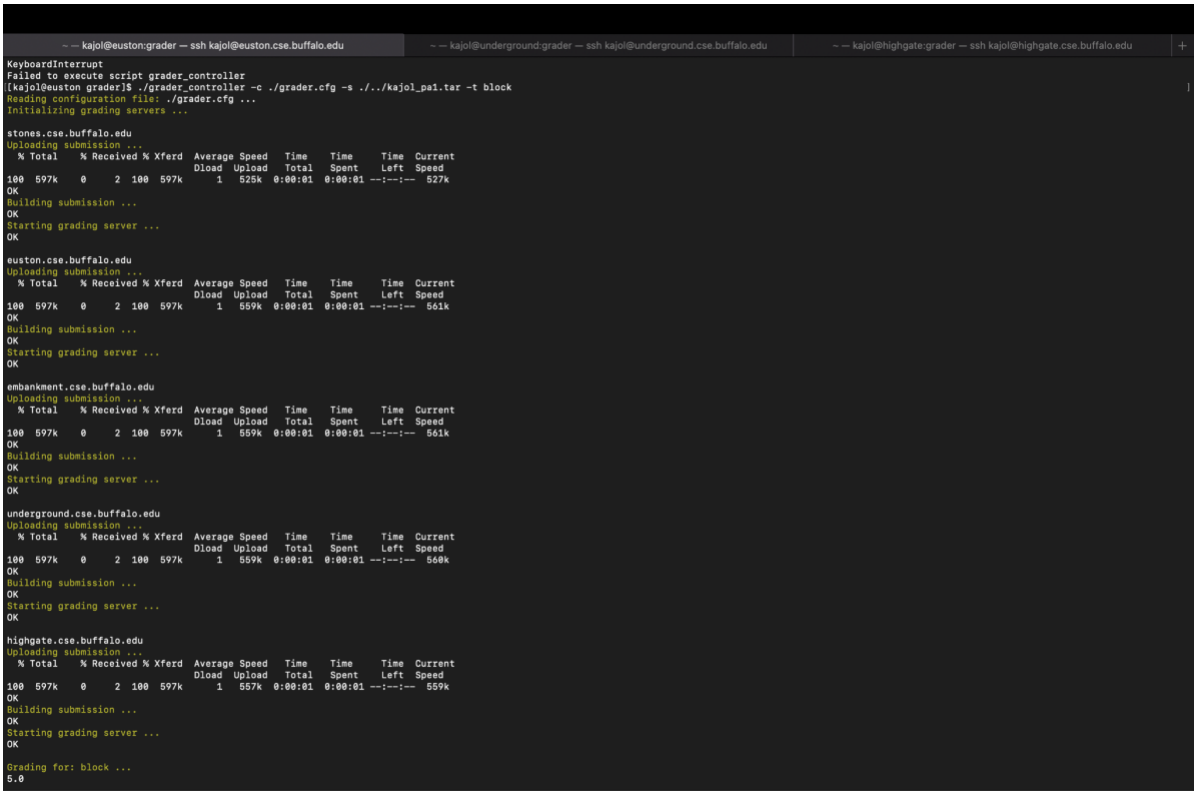
The printf that follows prints out a single host. Repetition of the printf command

- Listed every host
- list_id: an item number in an integer
- hostname: fully qualified hostname in a null-terminated char array.
- num_msg_sent: integer representation of the client's message sending count
- num_msg_rcv: integer representation of the number of messages the client has received.
- status: logged-in or logged-out null-terminated char array



[5.0] BLOCK (block)

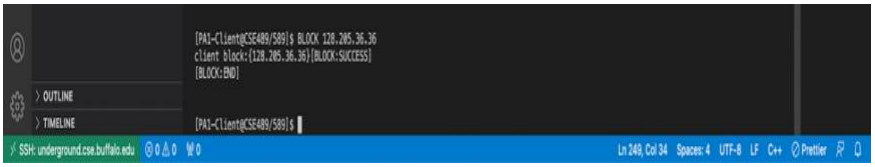
Grader screenshot:



Description:

Block any broadcast or unicast messages coming from the client with the IP address: "client-ip." This blocking should be communicated to the server by the client implementation. Any messages from a blocked sender that are intended for the blocking client shouldn't be relayed or stored/buffered by the server. However, the blocked sender won't be aware of this blocking and should successfully complete the SEND command.

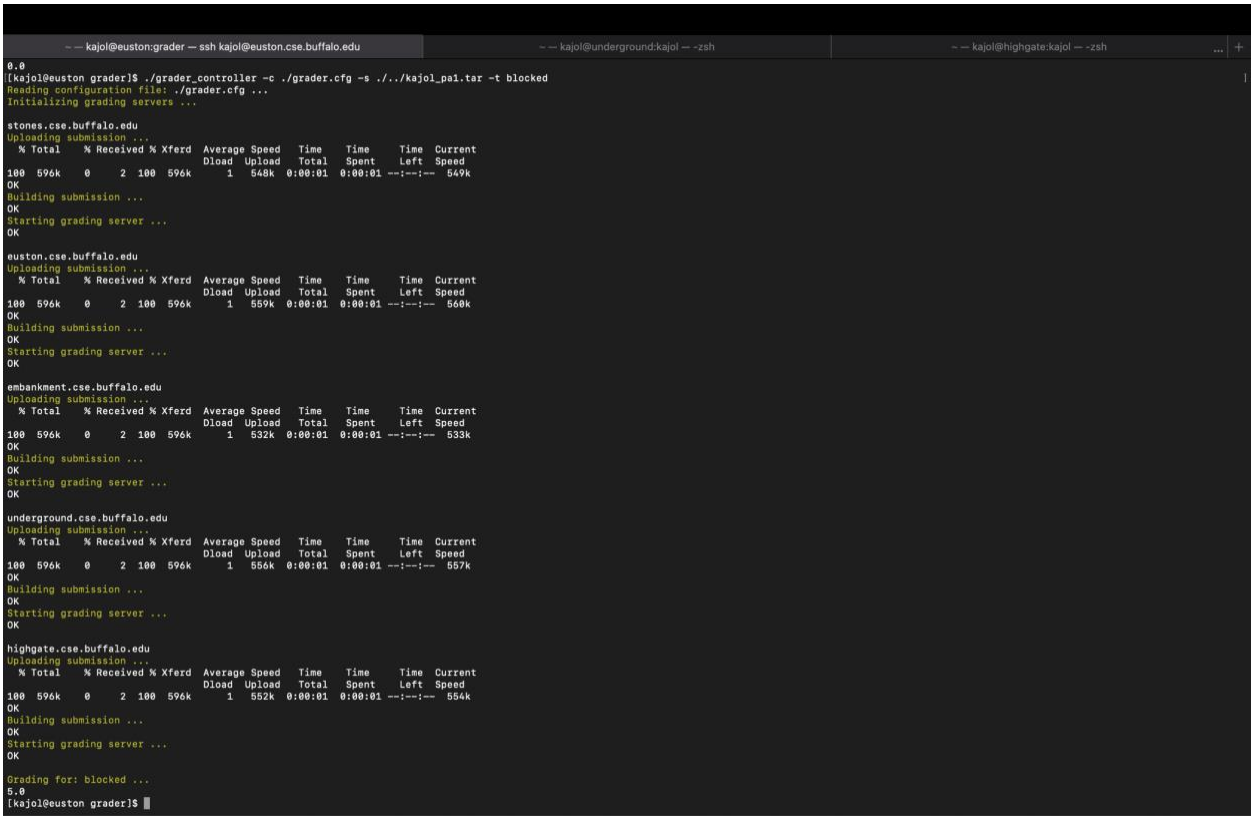
AT CLIENT



AT SERVER

[5.0] BLOCKED (blocked)

Grader screenshot:



Description:

Shown a numbered list of all the clients that the client with the IP address: <client-ip> has blocked. The output shows the hostname, IP address, and listening port numbers in increasing order, sorted by listening port numbers. The output format matches the LIST command exactly.

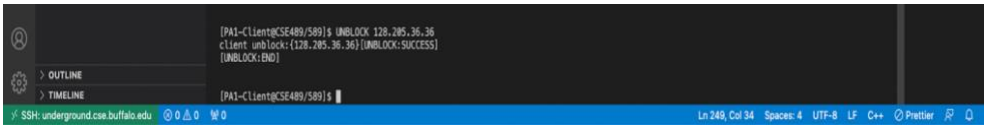
[2.5] UNBLOCK (unblock)

Grader screenshot:

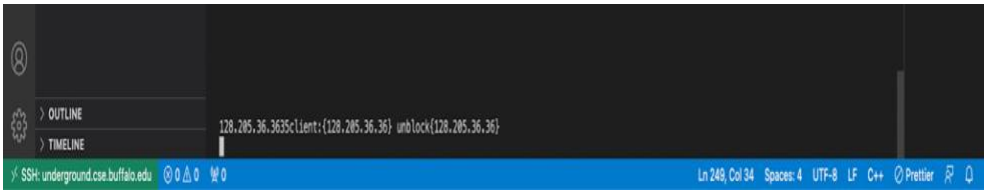
Description:
Unblock a client with the IP address “client-ip” who was previously blocked. The server should be notified of the unblocking by the client implementation.

```
Starting grading server ...
OK

euston.cse.buffalo.edu
Uploading submission ...
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100  193k    0     2  100  193k      1   187k  0:00:01  0:00:01 --:--:--  187k
OK
Building submission ...
OK
Starting grading server ...
OK

embankment.cse.buffalo.edu
Uploading submission ...
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100  193k    0     2  100  193k      1   186k  0:00:01  0:00:01 --:--:--  187k
OK
Building submission ...
OK
Starting grading server ...
OK

underground.cse.buffalo.edu
Uploading submission ...
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100  193k    0     2  100  193k      1   186k  0:00:01  0:00:01 --:--:--  186k
OK
Building submission ...
OK
Starting grading server ...
OK

highgate.cse.buffalo.edu
Uploading submission ...
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100  193k    0     2  100  193k      1   187k  0:00:01  0:00:01 --:--:--  187k
OK
Building submission ...
OK
Starting grading server ...
OK

Grading for: unblock ...
2.5
```

AT CLIENT



AT SERVER



[5.0] (buffer)

Grader screenshot:

Description:

The server replies when a registration is successful with:

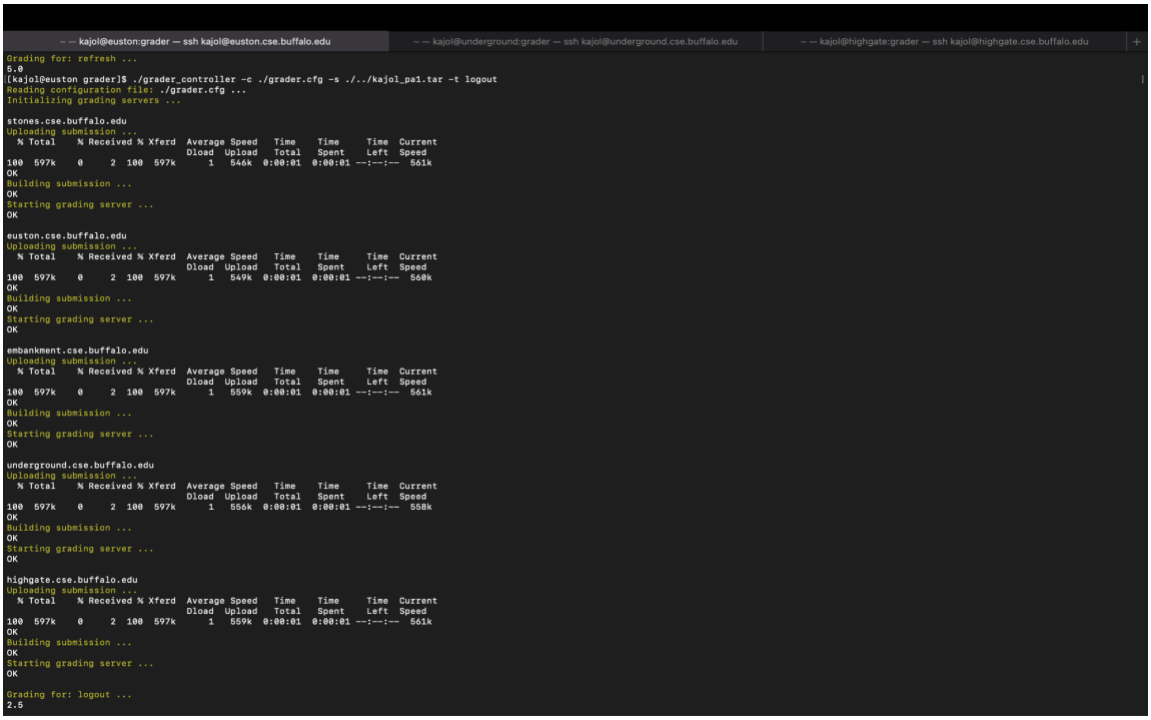
1. A list of all clients who are logged in right now. This list would be saved by the client for subsequent use and display.

2. All the buffered/stored messages for this client in the sequence in which the server received them. These messages will each result in a [EVENT]: Message Received In the event that a message (msg) with the following format string is received from a client with the IP address (client-ip):

("msg from:%s\n[msg]:%s\n", client-ip, msg)

[2.5] LOGOUT (logout)

Grader screenshot:



Description:

Will close the server session. However, application will not shut down and will keep accepting instructions like LOGIN, EXIT, IP, PORT, and AUTHOR. On LOGOUT, all client-related state is typically kept on both the client and the server.

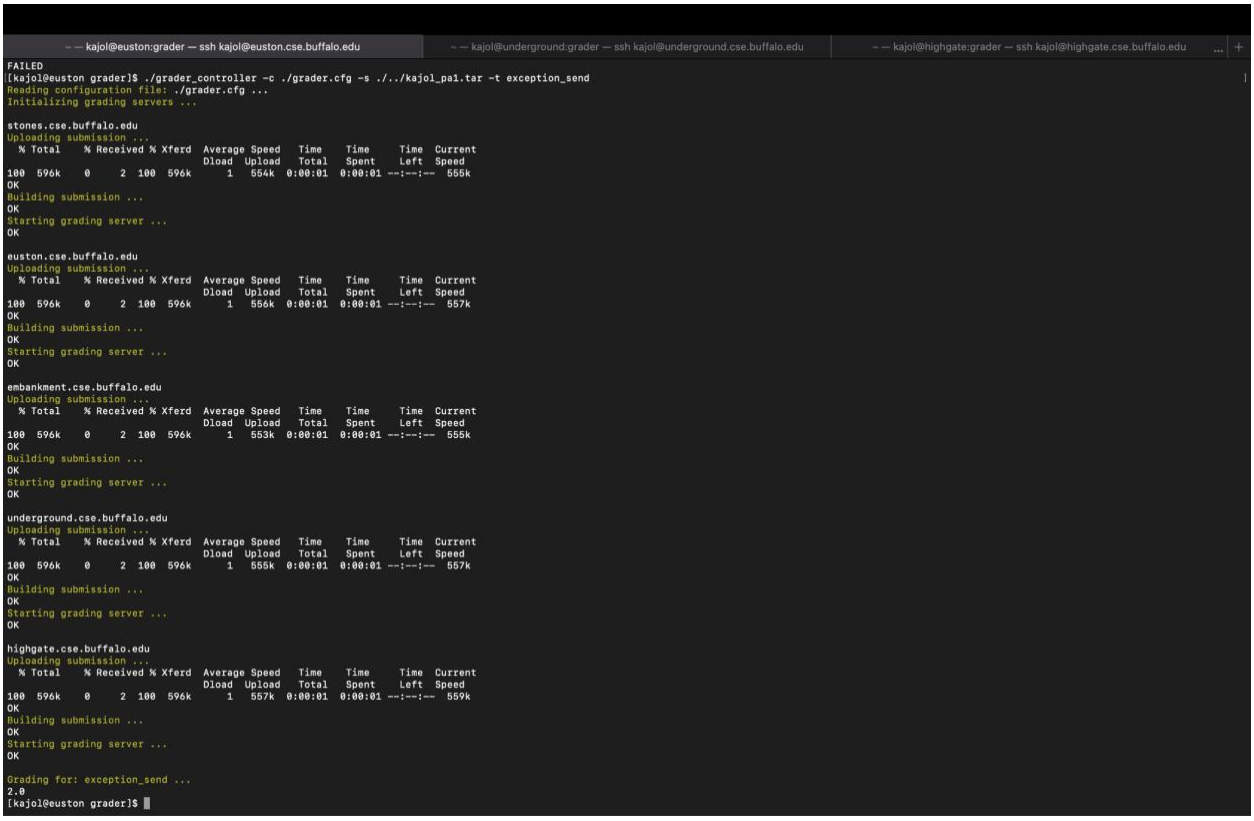
- The statistical counter reset is not affixed by LOGOUT
- No clients blocked by this client are unblocked after LOGOUT.
- The blocked/unblocked state of this client on the server is not changed by LOGOUT.

AT CLIENT

AT SERVER

[2.0] SEND Exception Handling (exception_send)

Grader screenshot:



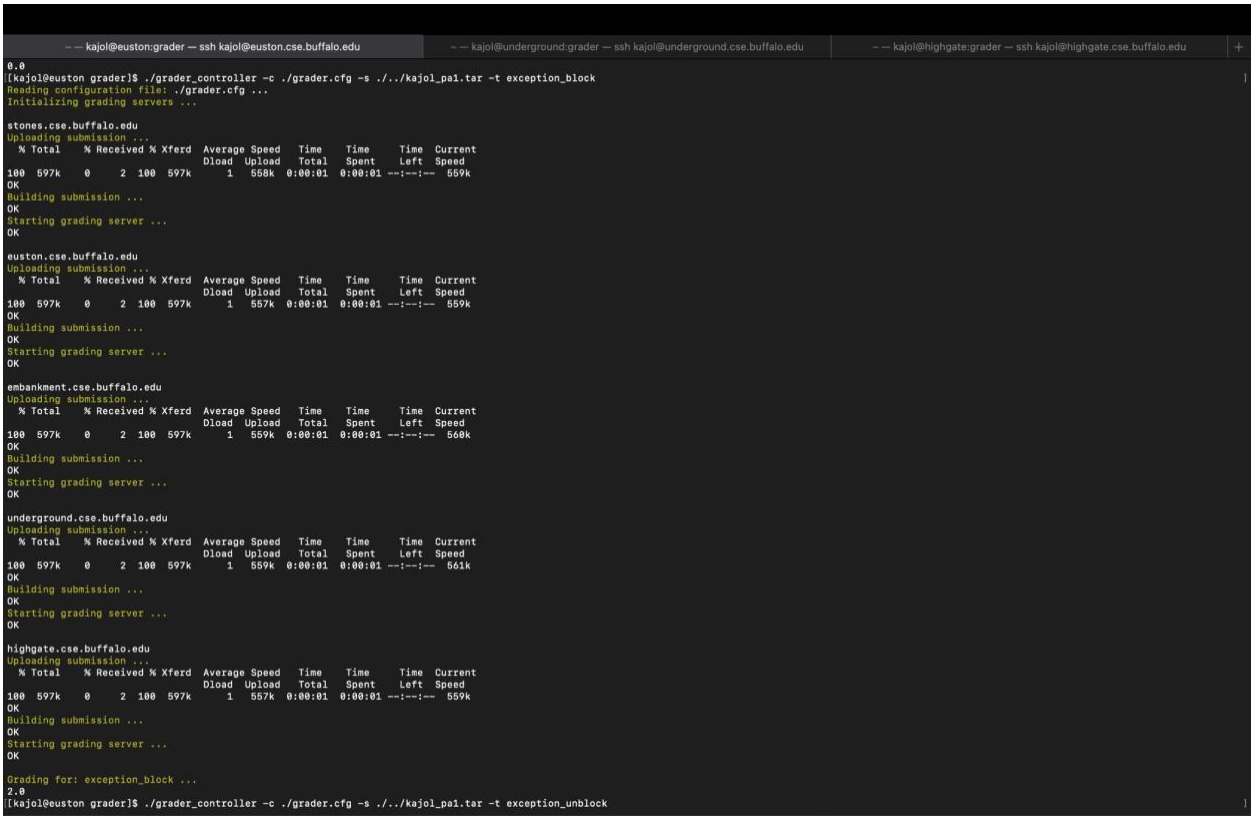
Description:

Handling of exceptions:

- Incorrect IP address
- Valid IP address that is absent from the local copy of the list of clients logged in

[2.0] BLOCK Exception Handling (exception_block)

Grader screenshot:



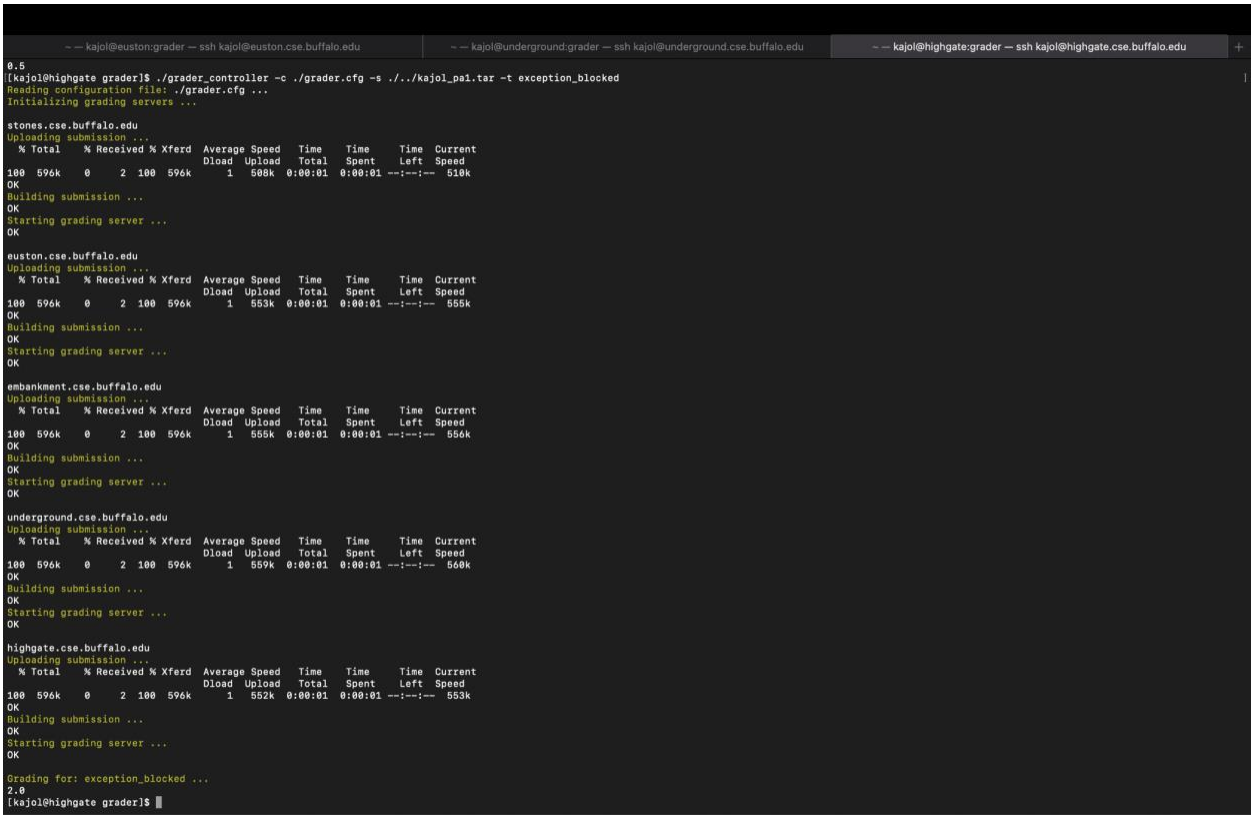
Description:

Handling of exceptions:

- Incorrect IP address
- Valid IP address that isn't on the local copy of the list of customers who are logged in
- Client with the following IP address has previously been denied access.

[2.0] BLOCKED Exception Handling (exception_blocked)

Grader screenshot:



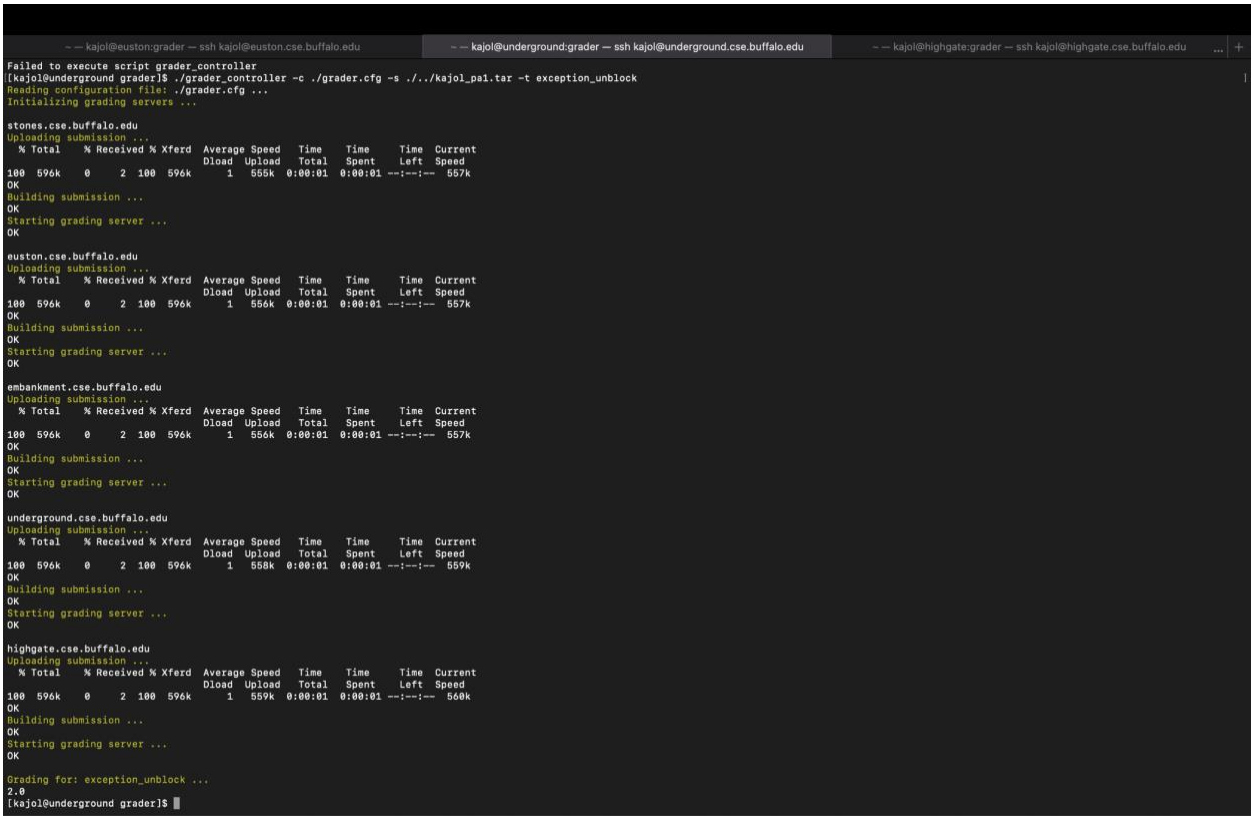
Description:

Handling of exceptions:

- Improper IP address
- Valid IP address that is ineffective or nonexistent

[2.0] UNBLOCK Exception Handling (exception_unblock)

Grader screenshot:



Description:

Handling of exceptions:

- Incorrect IP address
- Valid IP address that isn't on the local copy of the list of customers who are logged in
- Not blocked is the client with the IP address "client-ip".

REFERENCES

- **struct Socket_Object**

  https://github.com/jessefjxm/CSE589-Text-Chat-Application/blob/master/src/hwang58_assignment1.cpp

- **int_server_section**
  https://github.com/tingting0711/CSE589_network_programming/blob/master/cse489589_assignment1/twang49/src/twang49_assignment1.cpp

- **PA1 Template**