Client Server Information Transfer Program

CSITP

Bogdan Dawabsheh

COIS 4310

v. 2.0

Changelog:

-- Minor bug fixes

- Added ability to encrypt Client to Client messages

Table of Contents

Introduction

Client Server Information Transfer Program is used in order to establish communication between a client and a server. The model utilizes CSITP in order to communicate at Client and Server levels and uses TCP/IP protocol in order send packets across the network. (See Appendix A for TCP/IP protocol information).

<u>Assembly and runtime:</u>

The program is written in Java, therefore a java compiler and a jdk environment is necessary to run the application. Loki can be used as an alternative platform the test and run the application

<u>Compile instructions</u>

The example uses loki and the loki java compiler:

```
javac server.java javac

client.java
```

Must be run in that order specifically. For your convenience, the distributed version contains a compiled version of the program.

<u>Run instructions</u>

On Loki:

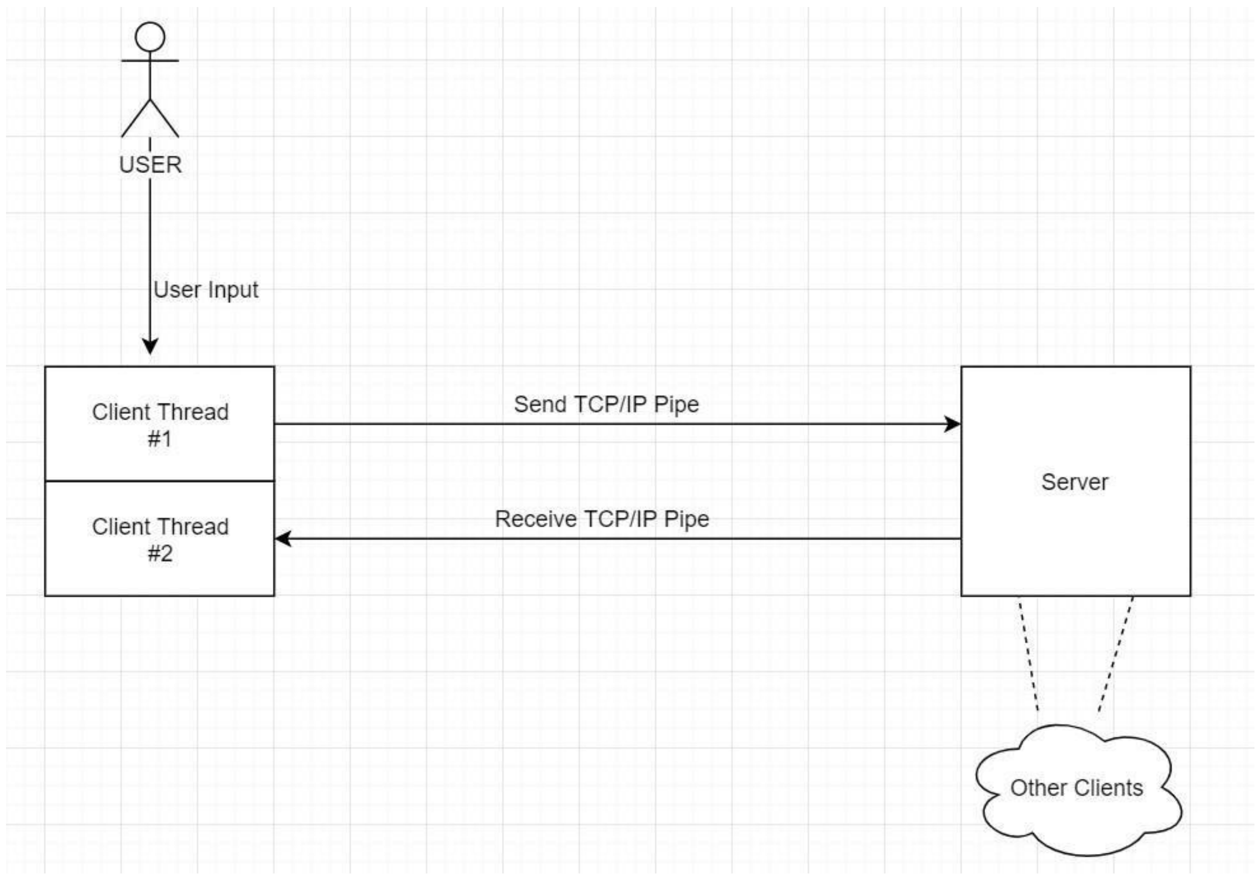
```
java Server ← ENSURE Server and Client have capitalized first letter java

Client
```

Must be run in that order

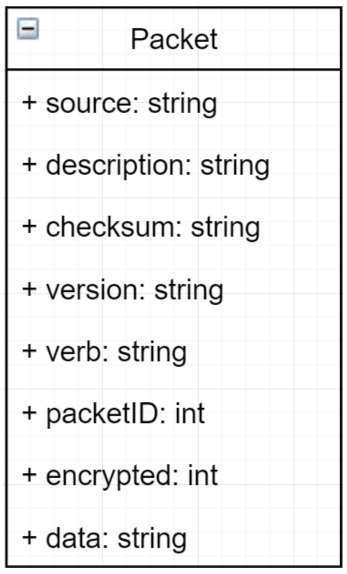
Client Server Information Transfer Program Model

The CSITP is designed with the following result: One or more clients establishes connection to the server and registers its name in the Server structure. Client then separates into two threads, one is responsible for monitoring server sent packets and display them to the user, while the other thread is responsible for user interaction: command access, and upon user input, it assembles packets with server instructions and send them over the TCP/IP protocol to the server. The server has as many threads as there are clients, each responsible for communicating with that specific client. Upon receiving a packet from the end user, the server performs the assigned duty as outlined by the packet -> display active users, forward packet to all users, forward packet to specific user, safely disconnect, etc.



Packet Structure

The packets are structured as follows:



Before communicated on TCP/IP layer, they are converted into a ASCII byte array.

Source identifies the sender of the packet

Destination identifies the recipient of the client or "ALL" if ALL active clients are impacted. See Specifications

Checksum contains packet verification information

Version contains program version number

Verb contains SERVER specific instructions ("Disconnect", "WHO"). See Specifications.

Encrypted contains information whether data is encrypted

Data contains message data to be transferred to other clients.

Client Server Information Transfer Protocol Procedures

Server Forwarding

Upon receiving a packet from the client, if the recipient is not the server, the server will check the connected list of clients and if the recipient exists, it will forward the message to the recipient.

Similarly, if the recipient is ALL, as in all active users, the message will be transmitted to all users that are connected to the server.

Server Replies

The server doesn't send replies to clients upon a successful packet receiving. The only transmission from server to client occurs when forwarding is necessary or a resend is necessary due to network error.

Client Recognition

Upon the first connection to the server, the client sends a request with the verb: "FIRST" indicating the first connection to the server. The server saves the client socket information as well as the source of the packet (in our case, the username the client chose) and uses it when forwarding information is required.

Sending and Receiving

The Sending and received is performed by both the client and the server through the means of the TCP/IP protocol. The clients connect to the server on the local host through the port specified: 59404

The packet is created with required information and is converted into a string the likes of which are transported through the network. When received, the string is converted back into a packet and analyzed.

Encryption

When the client choses, he is able to toggle on/off the encryption of his messages between him and specific user or all users. Server will be unable to read the encrypted messages and the messages will travel through the network encrypted.

Encryption is based on ROT13, where the characters are moved 13 spaces forward or backwards,

Opening and Closing

When the client sends a DISCONNECT request to the server, the client information is removed from the active user list. The client safely disconnects from the server and shuts down both of its threads.

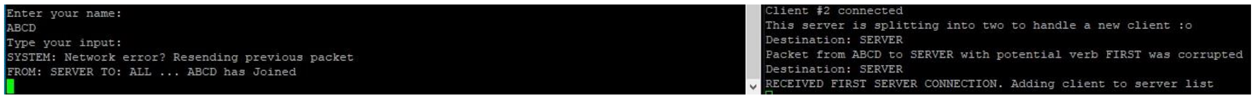
Checksums

Checksums in this application involves encoding the message into an MD5 hash, attaching it to the packet, sending the packet over the wire, and comparing the MD5 hashes on both sides.

For example, the client creates a hash "12345" using the method found in code using a MessageDigest java library (see appendix for more info) based on message "HELLO". The client would attach the hash to the packet and send it over to the server. The server would take the message found in the packet, create its own hash, and if they match, the checksum verification has been done successfully.

If checksum verification failed, the server or the client would send a RESEND request and request a packet again.

See image for live demonstration: (LEFT = CLIENT. RIGHT = SERVER)



```
Enter your name:
ABCD
Type your input:
SYSTEM: Network error? Resending previous packet
FROM: SERVER TO: ALL ... ABCD has Joined
```

```
Client #2 connected
This server is splitting into two to handle a new client :o
Destination: SERVER
Packet from ABCD to SERVER with potential verb FIRST was corrupted
Destination: SERVER
RECEIVED FIRST SERVER CONNECTION. Adding client to server list
```

Delimiters

The conversion from a string packet to a packet format involves using a delimiter, in our case "___" when creating packets.

In order to avoid any issues with user input, all user input has been scrubbed as well as limited only to AZ, a-z, 0-9 as well as whitespace. Meaning you can only write words you can't add special characters.

The delimiters create the packet in a format as follows:

< SOURCE >___<DESTINATION>____<VERB>____<PACKETID>.... And so on.

Specifications

Client Commands

WHO

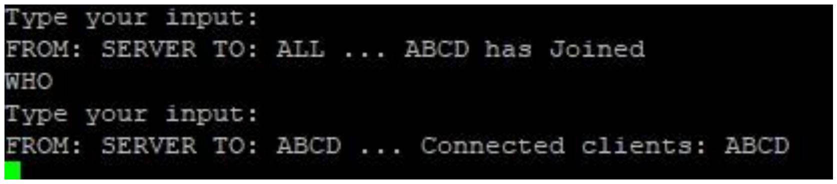
Command to display the list of active users connected to the server

Client sends a packet to the server, to which the server replies with a list of active users in a string format

Usage: WHO <

ENTER >

Example:



```
Type your input:
FROM: SERVER TO: ALL ... ABCD has Joined
WHO
Type your input:
FROM: SERVER TO: ABCD ... Connected clients: ABCD
```

QQQ

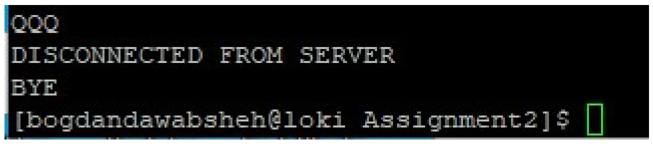
Command to disconnect from server and shut down the program.

Client sends a disconnect request from the server, the server removes the client from the list of active users and the client shuts down connection and threads uses.

Usage: QQQ <

ENTER >

Example:



```
QQQ
DISCONNECTED FROM SERVER
BYE
[bogdandawabsheh@loki Assignment2]$
```

ENC

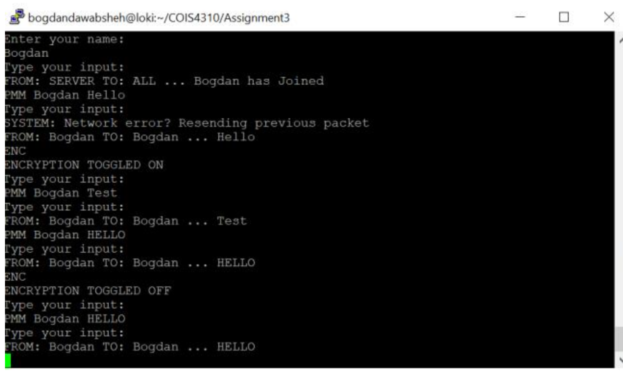
Toggles encryption on client side of application.

When encryption is off, server is able to read client data

When encryption is on, server is unable to read data.

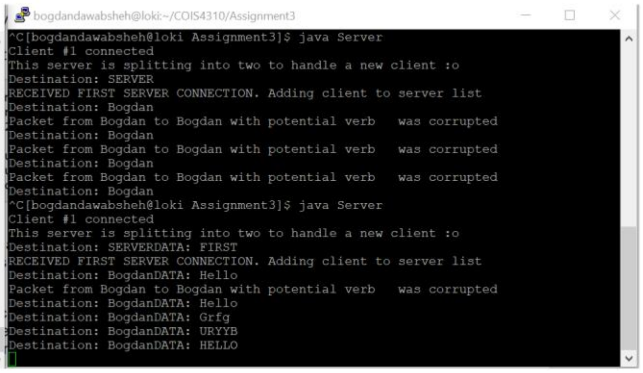
All of the messages between Client -> Server and Server -> Client remain unencrypted

Client view



```
bogdandawabsheh@loki:~/COIS4310/Assignment3
Enter your name:
Bogdan
Type your input:
FROM: SERVER TO: ALL ... Bogdan has Joined
PMM Bogdan Hello
Type your input:
SYSTEM: Network error? Resending previous packet
FROM: Bogdan TO: Bogdan ... Hello
ENC
ENCRYPTION TOGGLED ON
Type your input:
PMM Bogdan Test
Type your input:
FROM: Bogdan TO: Bogdan ... Test
PMM Bogdan HELLO
Type your input:
FROM: Bogdan TO: Bogdan ... HELLO
ENC
ENCRYPTION TOGGLED OFF
Type your input:
PMM Bogdan HELLO
Type your input:
FROM: Bogdan TO: Bogdan ... HELLO
```

Server view



```
bogdandawabsheh@loki:~/COIS4310/Assignment3
^C[bogdandawabsheh@loki Assignment3]$ java Server
Client #1 connected
This server is splitting into two to handle a new client :o
Destination: SERVER
RECEIVED FIRST SERVER CONNECTION. Adding client to server list
Destination: Bogdan
Packet from Bogdan to Bogdan with potential verb   was corrupted
Destination: Bogdan
Packet from Bogdan to Bogdan with potential verb   was corrupted
Destination: Bogdan
Packet from Bogdan to Bogdan with potential verb   was corrupted
Destination: Bogdan
^C[bogdandawabsheh@loki Assignment3]$ java Server
Client #1 connected
This server is splitting into two to handle a new client :o
Destination: SERVERDATA: FIRST
RECEIVED FIRST SERVER CONNECTION. Adding client to server list
Destination: BogdanDATA: Hello
Packet from Bogdan to Bogdan with potential verb   was corrupted
Destination: BogdanDATA: Hello
Destination: BogdanDATA: Grfg
Destination: BogdanDATA: URYYB
Destination: BogdanDATA: HELLO
```

PMM

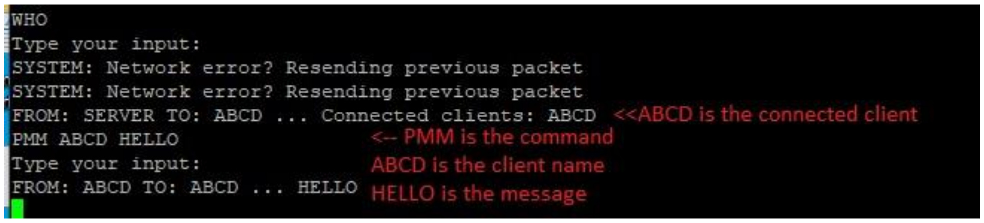
Private message to an active user.

Client sends a PMM packet to server, server forwards that packet to destination client and displays the message on their screen.

Usage:

PMM “<destination>” “<message>” < ENTER >

Example:



```
WHO
Type your input:
SYSTEM: Network error? Resending previous packet
SYSTEM: Network error? Resending previous packet
FROM: SERVER TO: ABCD ... Connected clients: ABCD
PMM ABCD HELLO
Type your input:
FROM: ABCD TO: ABCD ... HELLO
```

ALL

Public message to all active users.

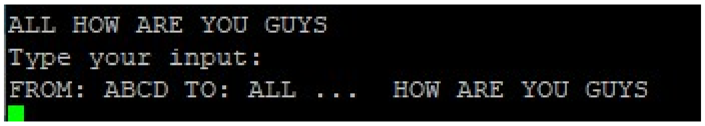
Client sends an ALL packet to server, server forwards the packet to all active users, including the client.

The client knows the message is delivered if it repeats.

Usage:

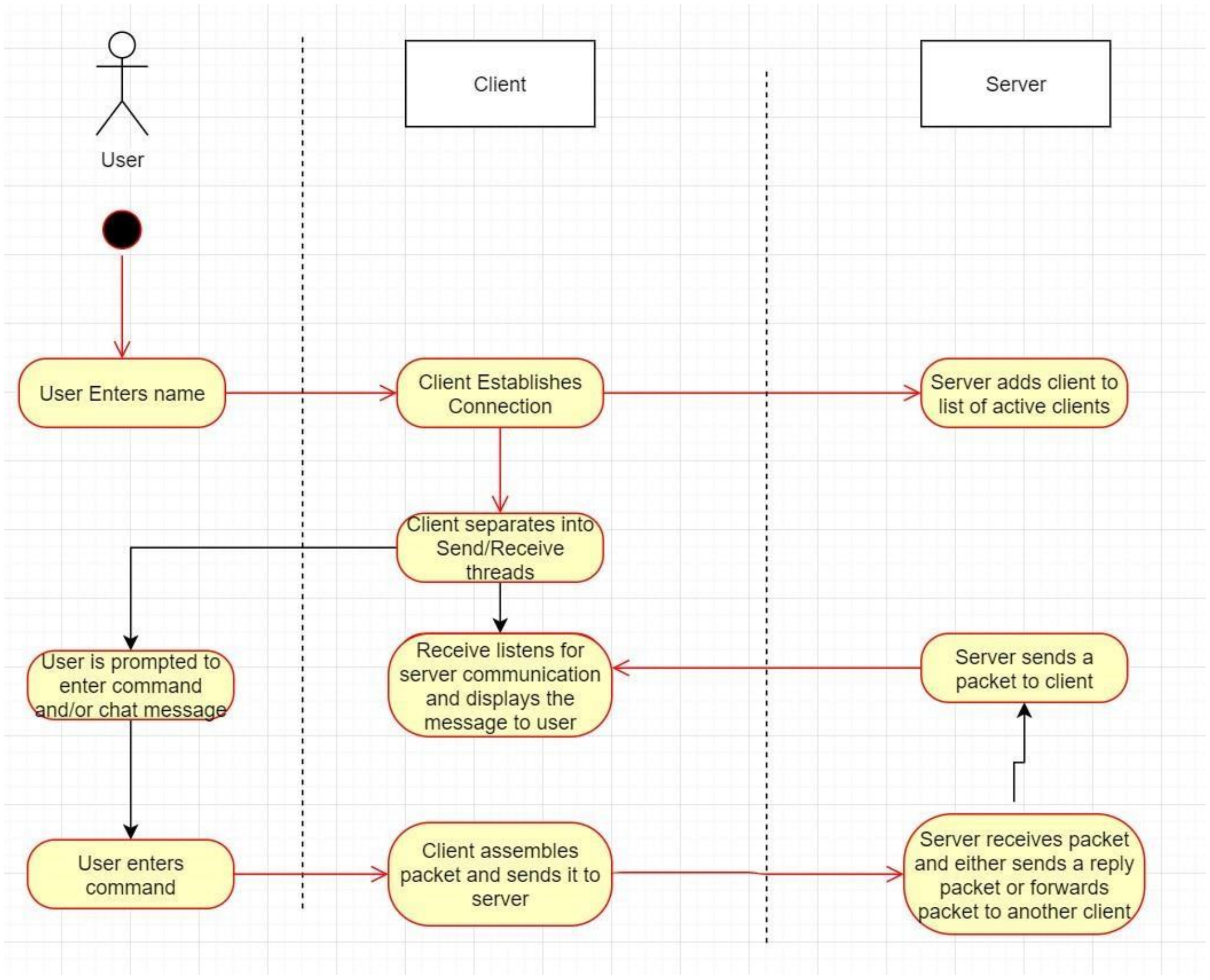
ALL “<message>” <ENTER>

Example



```
ALL HOW ARE YOU GUYS
Type your input:
FROM: ABCD TO: ALL ...  HOW ARE YOU GUYS
```

Interaction Flowchart


User
Client
Server
User Enters name
Client Establishes Connection
Server adds client to list of active clients
Client separates into Send/Receive threads
User is prompted to enter command and/or chat message
Receive listens for server communication and displays the message to user
Server sends a packet to client
User enters command
Client assembles packet and sends it to server
Server receives packet and either sends a reply packet or forwards packet to another client

Appendix A

TCP/IP

See https://tools.ietf.org/html/rfc793 and https://tools.ietf.org/html/rfc791 regarding the breakdown of TCP/IP protocol

MessageDigest

See https://docs.oracle.com/en/java/javase/12/docs/api/java.base/java/security/MessageDigest.html for more information regarding the library used to create MD5 hashes

Rot13

Symmetric Encryption Algorithm is based on:
https://introcs.cs.princeton.edu/java/31datatype/Rot13.java.html