Administration Documentation and Use Guide

This is a documentation manual for administrative purposes. The following is an extensive guide on how to run, mange, and troubleshoot multiple multithreaded server and client instances one or multiple machines. For general user instructions and troubleshooting see the general user documentation.

In this package is a simple client/server chat model that allows the user to select which protocol they would like to use to communicate. The client is dependent on the servers they have access to. The executable files included each open up one instance of their respective naming. The server is formatted to be changed therefor the only way to update an IP address is to create a new server.

Those who actively use the client application will need access to the port number and server information to be able to successfully connect. It is up to an administrator of the program to forward connection to users. Any changes made to the connection information is also the responsibility of the administration to rely to users.

Contents

What's in the Zip

The following is a list of the contents in the zipped folder. If any contents are missing download the zipped file again or contact *We Want an 'A'* Industries representative to receive the correct packages.

- Client executable file
- Server executable file
- General User Documentation and Use Guide(PDF and Word Format)
- Administrative Documentation and Use Guide(PDF and Word Format)
- Java files include:
    - **MainServer.java:** Starts the chat multi-threaded UDP and TCP/IP server, handles clients, and forwarded messages between clients.
    - **Client.java**: Creates a client for sending messages over TCP/IP or UDP. Requires a server name and port for connection.
    - **ServerGUI.java**: Creates the user interface for the server instances. It displays the server IP, server Port, and a list of the clients that are connected to the server.
    - **ServerThread.java**: An extension of the thread class that creates and runs an instance of the TCP/IP or UDP server threads. Also controls the formatting of system messages.
    - **TCPServer.java:** TCP/IP protocol server that waits for messages from a client and starts a new TCP/IP thread to handle the response (forwarded to other clients).
    - **UDPServer.java:** UDP protocol server that waits to receive a message from a client and forwards the response.
    - **ClientGUI.java**: Creates the user interface for the client instances. It displays an area to enter the server information, select protocol, view chat messages, and send a custom message.

Compiling the Server and the Client

There are many ways to compile the server and client java classes. A precompiled executable jar file is given to simplify confusion. Although if editing needs to be done to the files they will have to be recompiled to run with the new changes. Below is are two short walkthrough of how to compile the client and server files using command line if changes have been made. All the walkthrough assume that the current directory contains the java files.

If all files do not need to be compiled the following can be used to compile the files via command line:

javac file1.java file2.java

An example of this with all of the client server java files would go as follows:
javac Client.java ClientGUI.java ServerGUI.java MainServer.java TCPServer.java UDPServer.java

The simplest way to compile all of the java files if needed is the following:

javac *.java

Setting up an Active Server

Before attempting to start a chat server make sure the following tools are available on the machine that the server will be executed on.

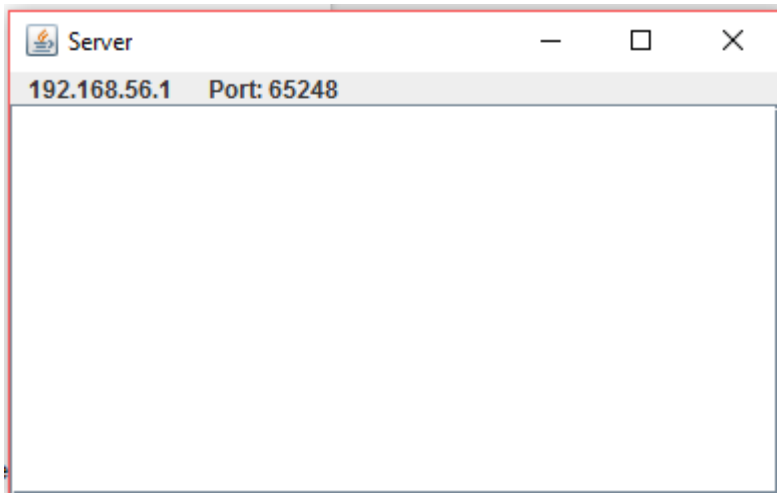
- **Java 8** – The most recent download can be found here : Download Java 8
  A quick description is provided to indicate which package of the Java Runtime Environment is needed to run a java program (ie. MainServer).
- **Server executable file** – this is provided in the zip file
- **Network Access** – The server requires for network access to properly run setup.

Running a Chat Server Instance

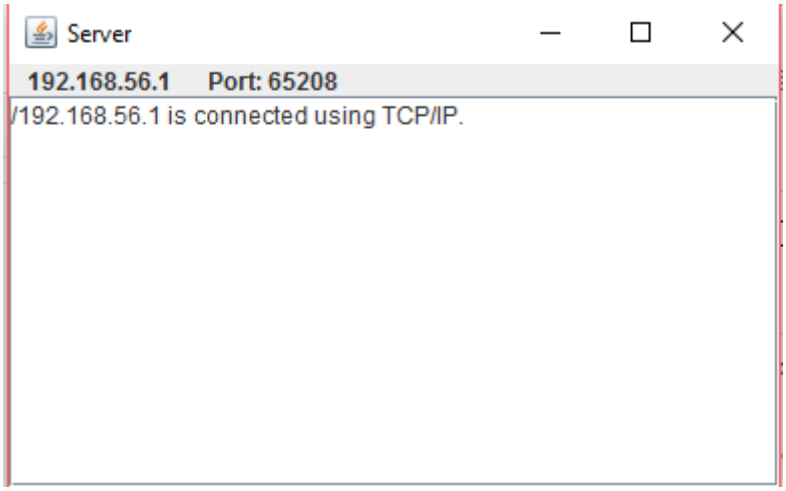
Running a chat server is extremely straightforward when you use the executable jar files. Simply open the files and the server instance should create itself. The server will assign itself the IP address of the current machine and will find an open port number to forward message to and from. If the server assigns itself the IP address 127.0.0.1, it does not have a network connection.

When connections are made the server keeps an onscreen log of the information. RIT IT lab machines are measured to be able to handle 50+ server instances at optimal performance.

1. Open the server executable file.



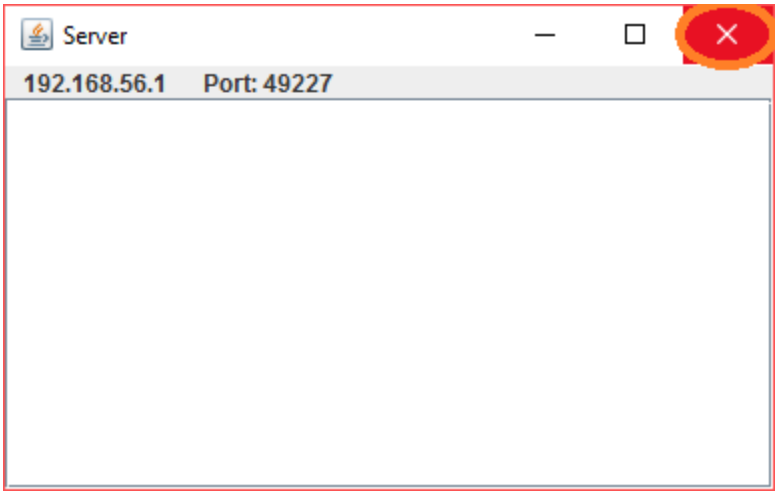
2. Wait for clients to connect



Closing a Chat Server

Closing an active chat server is clean and straightforward. Simply selection the ‘X’ button in the top right corner to exit and shutdown the server. Clients on TCP/IP connection will be notified immediately through the system. UDP clients will not. It is the administrator’s responsibility to notify users of closed server instance.

*any previous connection loggings will be deleted upon closing of a server instance. It is the admin’s responsibility to track or document loggings for long term use.



Setting up an Active Chat Client

Before attempting to start a chat client make sure the following tools are available on the machine that the client will be executed on. All chat client instances have a static connection, meaning that once a connection is made, the only way to create a new connection is to create a new instance of a client.

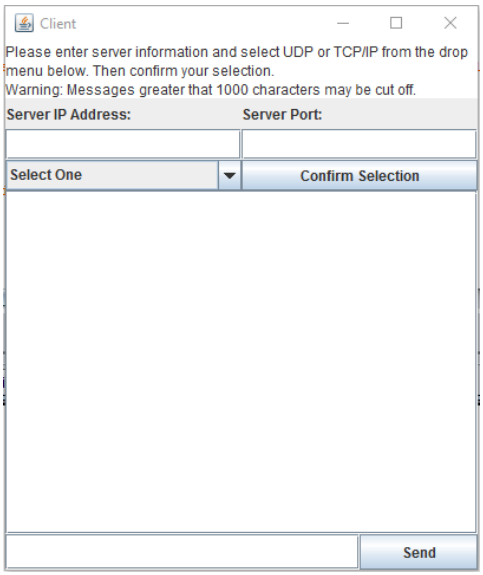
- **Java 8** – The most recent download can be found here : Download Java 8
  A quick description is provided to indicate which package of the Java Runtime Environment is needed to run a java program (ie. Chat Client).
- **Client executable file** – this is provided in the zip file
- **Network Access** – While the chat client will still run, communication between two chats will not be possible without a network connection.
- **Admin Server Information** Upon instantiation of the server the following will be at the top of the server GUI instance.
  1. Server IP address
  2. Server Port Number

Running a Chat Client

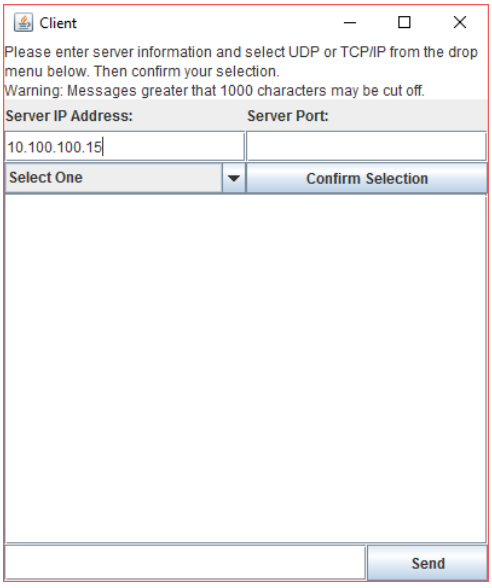
The following is a complete walkthrough of how to run and create a connection to the server given the necessary conditions from 'Setting up an Active Chat Client'. Screenshots are shown as necessary.
RIT lab machines have been measured to handle 25+ chat clients at optimal performance.

1. Open client executable file.

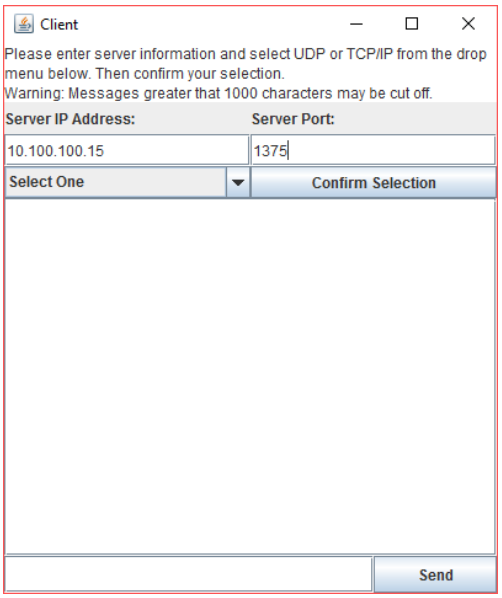
   The interface should open and allow for entering the credentials. The following application will appear:



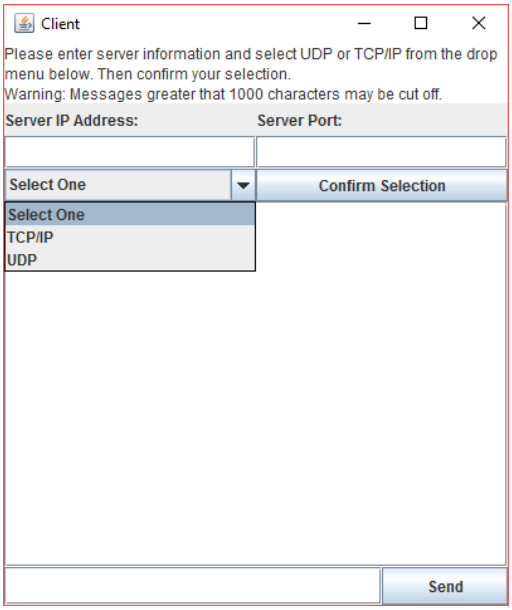
2. Enter in the server IP address.



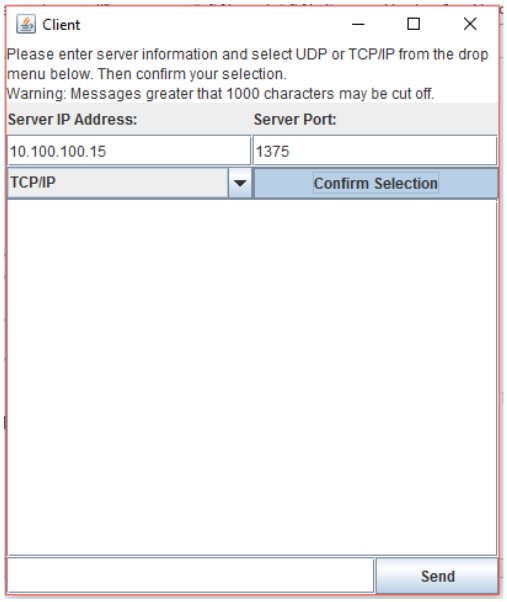
3. Enter in the server port.



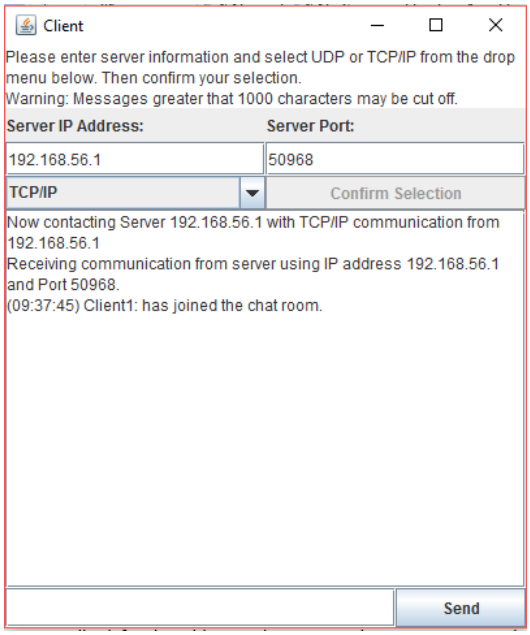
4. Select a form of connection. A brief explanation of each type of connection is given in ‘Connection Types’.



5. Select ‘Confirm Selection’ to connect to the chat server.



6. Upon connecting a success message should be displayed.
   *Once a connection is made, restarting the current or creating another client is the only way to switch to another connection

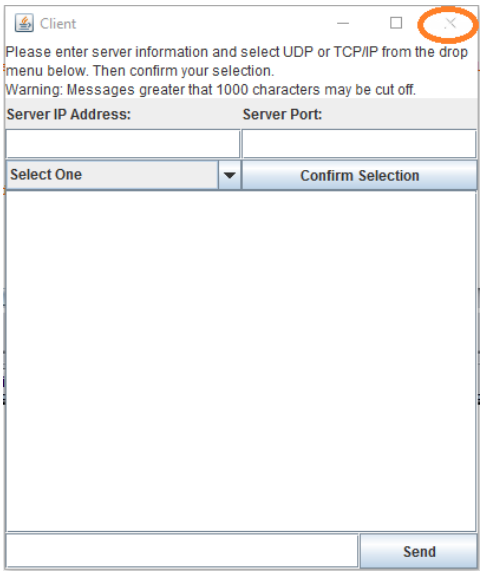


ProTip: If trying to connect and the program seems to have frozen, check the entered IP address. If the address is correct wait for the application to finish processing. If a connection cannot be made the application will ask for the address to be reentered. Processing may take longer than expected depending on the specifications of the machines hardware.

Closing a Chat Client

Closing an active chat client is clean and straightforward. Simply selection the ‘X’ button in the top right corner to exit the chat and disconnect from the server.

*previous communication of a closed chat, regardless of if the same server is used, will not be kept and cannot be back tracked or retained. It is the user’s responsibility to track any conversations that need to be saved.



Connection Types

The client chat server is able to use UDP and TCP/IP connections. The following is a very brief description and documentation of the differences between each type.

UDP

UDP stands for User Datagram Protocol and is a basic communication language or protocol of the Internet. UDP uses connectionless transmission to maintain a minimum protocol. This means that in the client chat, once a message is sent, there is no way of tracking if the entire message is received on the other end. UDP is faster, but less reliable.

TCP/IP

TCP/IP stands for Transmission Control Protocol/Internet Protocol and is the major protocol through which communication is conducted over the Internet. TCP/IP communication is primarily point-to-point, meaning each communication is from one point (or host computer) in the network to another point or host computer. This protocol is more reliable.

UDP Information and Restrictions

If using a UDP connection there are certain features that must be kept track of. The following are a list of the major differences in a UDP connection that may not occur in a TCP/IP connection.

- Message Sending

  Due to UDP being connectionless, there is no way to track if one chat client is actually relaying information to another client.

- UDP Timeouts

  Due to the possibility of server error, if a message is not received within an hour, the client will disconnect and instruct the user to restart the service and try to reconnect. This is the only way to ensure that a connection is still active on a UDP based chat client.

- Server Disconnections

  When a server is disconnected or crashes, UDP clients cannot receive an error message to inform the user. For this reason timeouts are incorporated to monitor activity and reduce the chance of an unnoticed disconnect.

ProTip: If a message is sent and it does not appear on the console it may indicate that the server is down or there is a problem in the application. Restart the chat client with a new connection or contact your administrator with questions.

TCP/IP Information and Restrictions

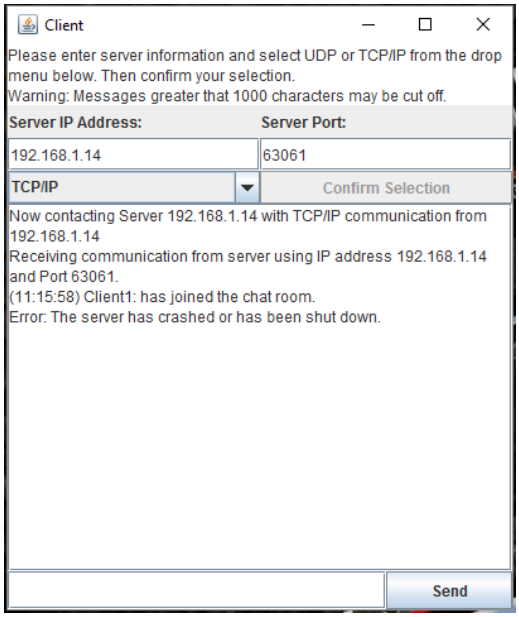
TCP/IP will most likely be the protocol of choice for users that wish to keep live connections to a server. Below is a list of the difference in features that TCP/IP has in comparison to UDP.

- Message Sending

  Messages are more secure over TCP/IP in that they are always connection based. This means that confirmations are sent if a message is well received and vice versa. Users will be alerted if there is an error in any of the processes of sending a message.

- Server Disconnections

  When a server disconnects from a client, a message will appear to the client (image shown below). Restart the client and try to reconnect. If connection fails, restart the server instance.



Trouble Shooting

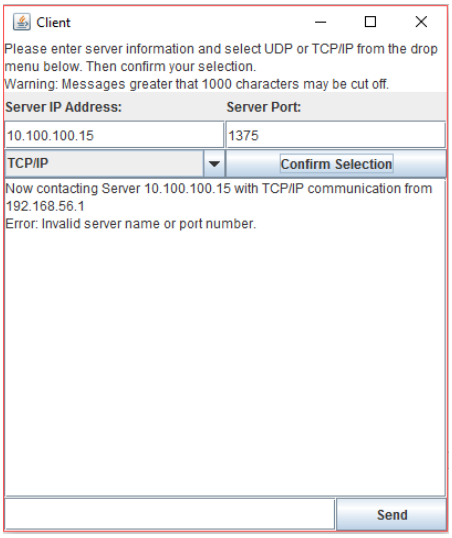
All though the client chat and chat server applications has been strenuously tested, errors do occur. The following are some of the most common errors and how to handle them if the applications malfunction.

The server application is static meaning that nothing needs to be altered. This was chosen to keep error possibilities as close to zero as possible. Due to such most errors will relate to client chat instances.

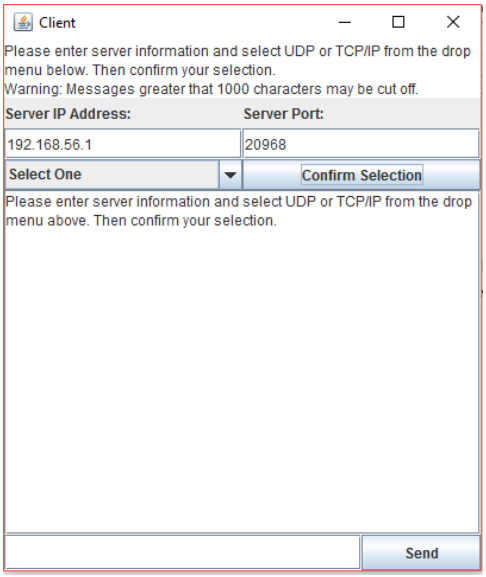
Failed Connections

A connection may fail for many reasons. If there is ever a failure, the application will warn the user and suggest the best course of action to a solution. The following are examples of some of the possible failures.

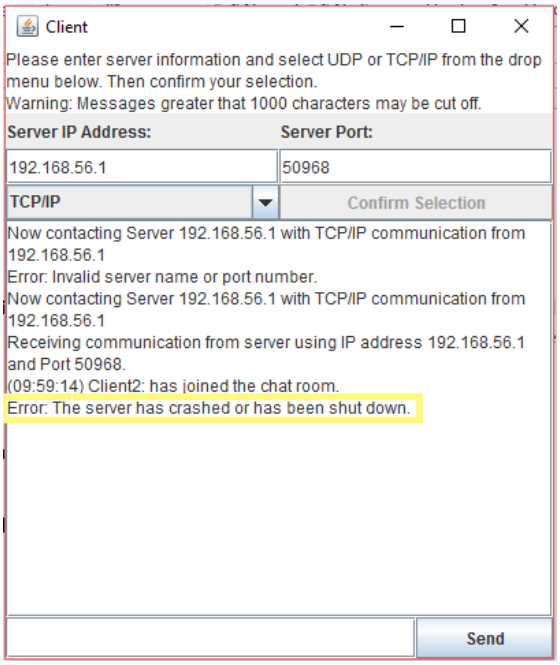
- Invalid IP address or port number



- Unselected protocol



- Disconnected Server (TCP/IP Only)



- No IP Address or port number assigned:

  This error is **extremely** unlikely. The chance of this happening is so low that the *We Want an 'A'* team could not even provide a screen shot of the issue. Since there is a still chance of this happening a solution is still given.

  If this error occurs reduce the amount of server instances. If the error still occurs restart the computer. Once restarted preform an *ipconfig* on the machine to make sure an IP address is assigned. Then go through the steps of starting a server instance.

Crashed/Unresponsive Server in a Client Instance

In the instance of an unresponsive or crashed server, restart the client chat and try to reconnect. If a connection is not possible restart the server instance.
TCP/IP will notify the user of server issues but UDP will not. If the server seems to be failing or messages seem to not be sending during UDP communication, follow the same directions.

Message Length and Loss

The maximum amount of character that can be sent in one message is limited to around 1000. If more than 1000 are sent, there is no guarantee that it will be fully received. Although the message will be completely shown to the user sending it (right), it may not be complete for the receiver (left) as shown below.