Taneli Hukkinen, 220628
Teemu Kekkonen, 295310
Janne Jaanila, 225733

S-38.3610 Network Programming

Software Project

Project Plan
mChat

1. Overview

The goal of the project is to implement a distributed group chat system. It consists of interconnected servers and clients. Clients will have to connect to one of the servers. Both the client and the server parts of the system will be implemented. The clients may join chat groups called rooms and discuss with the other clients in the same room. The users in a room can be directly connected to any one of the servers. Therefore, the messages have to be relayed across the network to all servers.

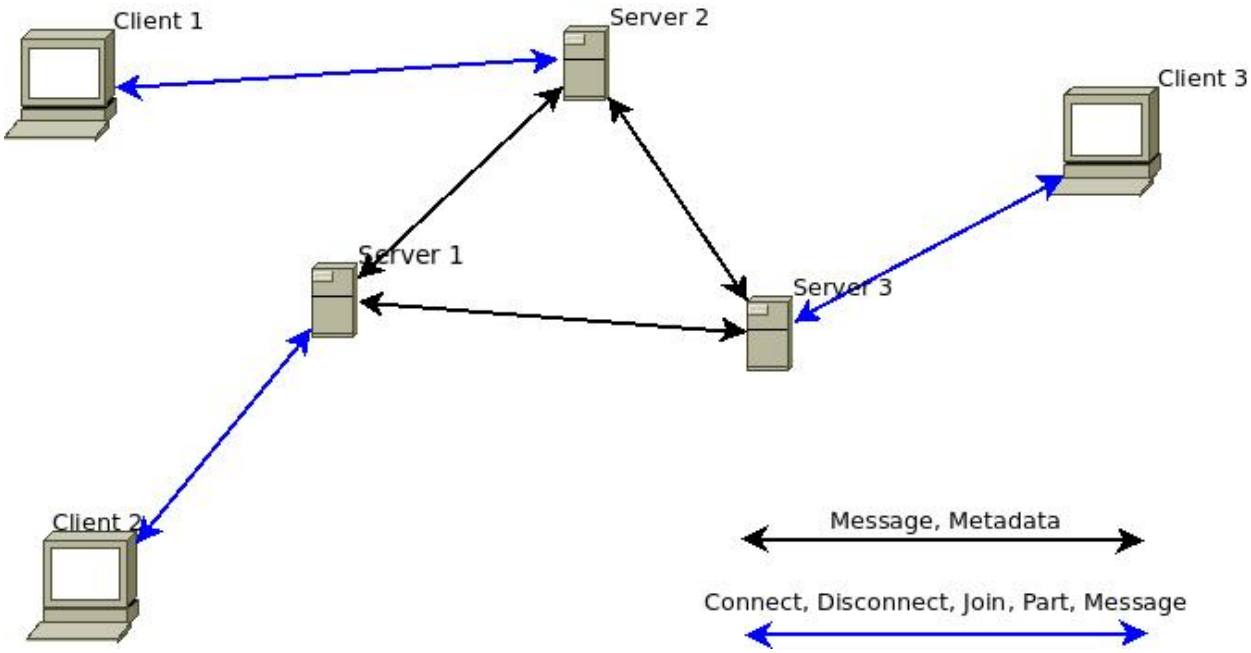

Figure 1. System overview.

2. Requirements description

The software is divided into server and client components. The server maintains connections to other servers and relays messages of clients. In our model, all servers are connected to each other, thus forming a fully connected network. The advantage of this is simple implementation and message relaying. Other servers will be preconfigured to the servers. Disadvantages include a large number of connections that is N*(N-1) where N is the number of servers. Another way could be using a spanning tree of the network like in Internet Relay Chat protocol [1], which would decrease the number of needed connections. However, we decided to use our method for simplicity. The server keeps track of which rooms the directly connected users belong to. Only the messages related to rooms occupied by that particular client are forwarded to it.

The client part consists of a chat user interface and a section that communicates with one server. The user interface will be a simple command line interface as the focus is in the network programming. The client is able to perform following actions: connect to a server, disconnect from server, send message, join room and leave room. Connect and disconnect will create a TCP connection between client and server. Join and leave room will send the server connected to the client an order to perform desired action. The server will keep track of rooms which the client belongs to. Send message will send a message to the server which will then forward it to other servers.

In addition, a tester component will be implemented for testing the robustness of the servers. The methods that will be used include testing large load, network failures, and unexpected communications such as a malicious or a buggy client.

The intended operation platform for the server software will be Linux and the implementation language will be either C or C++. Python will be used for the client part. Python should allow easier cross-platform development and as a higher level language, faster development time [2].

4. Communication protocol

The communication protocol of mChat system works on top of TCP. Thus, communication is reliable and this protocol does not need separate acknowledgement messages.

Protocol messages have multiple fields separated by spaces and end in newline. Nicknames and room names should not contain spaces or newlines. Messages can contain spaces but not newlines.

Different protocol messages are identified by their starting tag MSG, PART or JOIN. They are all followed by the name of the target room. MSG has extra fields for transmitted mChat message and user nickname. Only MSG messages are sent between servers. The maximum length of a protocol message is 1024 bytes. The server will ignore any message longer than this. The different message formats are presented below.

MSG <nick> <room> <message>\n
PART <room>\n
JOIN <room>\n

8. Distribution of work

First phase

The initial meeting was during the topic giving session. Merely the project name and the potential programming languages (C or C++, Python) were decided during that meeting.

We had another project meeting on 12th of February 2015 with all the group members. In that session, we planned the overall software functionality and drafted server network figures. Additionally, different message types and their data were decided at least tentatively.

The first draft of the project plan was written to Google Drive singly. No meeting was arranged but we have also discussed in IRC. The development repository is in GitHub: https://github.com/jjaanila/mChat

Janne wrote most of the Python client in a couple of hours in 23.2.2015 and finished it 26.2. It should now be fully operational. Teemu wrote the base of the document.

9. References

[1] Internet Relay Chat, RFC 1459 - https://tools.ietf.org/html/rfc1459
[2] Ousterhout, J.K., "Scripting: higher level programming for the 21st Century," *Computer* , vol.31, no.3, pp.23,30, Mar 1998