CS594 Daniel Song

Internet Draft Portland State University

Category: IRC Project Specification December 6, 2019

**IRC Class Project**

**Abstract**

This document contains information on the IRC application, which is a client-server

system that lets users communicate via websockets. It captures the system architecture,

technologies, and extra features implemented.

**Status of This Memo**

This document is not an Internet Standards Track specification; it is published for

informational purposes. This is a contribution to the RFC Series, independently of any

other RFC stream. The RFC Editor has chosen to publish this document at its discretion

and makes no statement about its value for implementation or deployment. Documents

approved for publication by the RFC Editor are not a candidate for any level of Internet

Standard; see Section 2 of RFC 5741.

Information about the current status of this document, any errata, and how to provide

feedback on it may be obtained at http://www.rfc-editor.org/info/rfc5384.

**Copyright Notice**

Copyright (c) 2014 IETF Trust and the persons identified as the document authors. All

rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to

IETF Documents (http://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights

and restrictions with respect to this document.

**Table of Contents**

* + 1. Introduction
    2. System Architecture
       1. Server
       2. Client
    3. Communication Protocol
    4. Events emitted by the server
    5. Events emitted by the clients
    6. Extra Features
    7. Security Considerations
    8. Conclusions

**Body of the Memo**

The motivation for this RFC is to specify the IRC application developed for the

Internetworking Protocols class at Portland State University.

1. **Introduction**

Internet Relay Chat, or IRC, is an application that allows users to communicate with

each other using messages. This project implements an IRC client and server system

providing functionalities where users can create, join rooms, send messages or files to

those rooms, and list all users in rooms. In addition, users can send private messages to

any of the online users.

This document specifies description of the application, such as the system architecture,

communication protocol design, security consideration, and finally conclusions.

**2. System Architecture**

The IRC application conforms to the client-server architecture. The data-flow of this

architecture is simple: When the user sends a message to a group, or to another user,

the message is first sent to the server. Upon receiving the message, the server performs

back-end logic, and relays the message to the recipient.

The client application can be run by connecting to the URL where the cloud server is

deployed on. It provides a simple user interface, where users can control the recipients,

and view users. The client side is implemented using HTML, and relies heavily on jQuery

for HTML DOM tree manipulation.

The server side of the application is implemented in JavaScript using Node js as

backend, and Express for routing and serving the HTML client file. It is responsible for

keep tracking of all connected users and existing rooms, and relaying messages. The

server is deployed on Heroku, a cloud platform.

The WebSockets in both the client and the server are implemented using the JavaScript

library, socket.io, for realtime bidirectional communication.

3**. Communication Protocol**

The communication protocol used in the application is built on top of the WebSocket

Protocol, which enables realtime, bidirectional data flow between a client to a server

using a single TCP connection.

The communication in the app is driven by a series of events that happen through out

the usage of the app. The clients and the server will both emit certain events, and listen

for events. All data objects that get transferred by these events hold JavaScript string

values.

**4. Events emitted by the server**

4.1 message

4.1.1 Description

The Message event is the main method the server uses to relay

messages to connected users. Upon “hearing” this event, the client side

of the application will display a message that was relayed from the server.

This event is also used for announcing when a user disconnects.

This event is also emitted when the server is closing in order to shut down

gracefully. It announces to all online users that the server is closing.

4.1.2 Fields

from : The user that’s sending the message (Socket id)

to : The recipient/s of the message (Socket id/s)

message: The message being sent

4.2 newUser

4.2.1 Description

The newUser event is responsible for announcing when a new user joins

one of the rooms the client is already in. Therefore, this event is only

triggered when the server “hears” the joinRoom event, which will be

described later. Furthermore this is only announced to users already in

the room.

4.2.2 Fields

client : User name (Socket id)

room : The room that user joined

4.3 updateRooms

4.3.1 Description

The updateRooms event is fired when a user creates a new room, ie. the

server “heard” the createRoom event emitted by a client. The main

purpose of this event is to update all online users with all existing rooms.

4.3.2 Fields

rooms : An array containing all existing rooms

4.4 clients

4.4.1 Description

Similar to the updateRooms event, this event is responsible for keeping

users updated with online users. The clients can use this information to

send private messages to other online users.

4.4.1 Fields

clients : An array containing all online users

4.5 userLeft

4.4.1 Description

The userLeft event is responsible for announcing to members of a room,

that a user has left the room. This is triggered after the server “hears” the

leaveRoom event emitted by the client.

4.4.2 Fields

client : User name (Socket id)

room : Name of the room the user left from

4.6 roomMessage

4.6.1 Description

This event is emitted when a client wants to send a message to all users

in a group. It is very similar to the message event, except instead of

sending to a single user, it broadcasts to all users in a room.

4.6.2 Fields

from : User name (Socket id)

to : Room name

message : Message to be sent

4.7 userList

4.7.1 Description

This event is fired when a client wants to view all the users in a room. It

will send a list of users ONLY to the client that requested for it.

4.6.2 Fields

clients : List of clients in a specified room

room : Specified room

4.8 sendFile

4.8.1 Description

This event is emitted when the user wants to send a file to a room. When

the user uploads a file using the input button, it will get encoded as base 64 data URL, and get sent to the server. The server will then relay to the

room the user specified.

4.8.2 Fields

from : User name (Socket id)

room : Room name

file : Content of the file as base 64 encoded data URL

**5. Events emitted by the client**

5.1 message

5.1.1 Description

This event is emitted when a user wants to send a private message to

another user. It doesn’t need to send the “from” data, as the server can

tell which user it is emitting the event.

5.1.2 Fields

to : User name

message : Message to be sent

5.2 roomMessage

5.2.1 Description

This event is emitted by the client and lets the server know the user wants to send a message to a room.

5.2.2 Fields

room : User name (socket id)  
 message : Message to be sent

5.3 listUsers

5.3.1 Description

This event is emitted when the user wants to list all users in a room.

5.3.2 Fields

room: Specified room selected from the dropdown menu

5.4 joinRoom

5.3.2 Description

This event is emitted when a user wants to join a specified room. Users

automatically join the Main room when they first connect to the server.

5.3.3 Fields

room name: Name of the room the user wants to join.

5.4 createRoom

5.4.1 Description

This event is emitted to the server to inform that a new room should be

created. The user will specify the name of the new room, and will

automatically join it once created.

5.4.2 Fields

room name: Name of the new room the user wants to create.

5.5 leaveRoom

5.5.1 Description

This event is emitted when the user wants to leave a room and no longer

wants to receive messages from that room. Causes server to announce

that the user left to the rest of the users still in the room.

5.5.2 Fields

room name: Name of the room the user wants to leave.

5.6 sendFile

5.6.1 Description

This event is emitted when the user wants to send a file to a room. The

file is uploaded using the input button, and is converted to base 64

encoded data URL before it gets sent to the server. Only images are

tested and work as of now.

5.6.2 Fields

file: The content of the file as base 64 encoded data URL

room: Specified room the user wants to send the file to

**6. Extra Features**

Extra features in this IRC application includes: Private messaging, file transferring, and

cloud hosted server.

**7. Security Considerations**

This IRC application provides no protection against, sniffing, man in the middle attack,

and other cyber attacks. The messages are no encrypted, and they all go through the

server. Users should refrain from sending sensitive information.

In addition, there is still work to be done for error checking. For example, it is still unclear

how long the messages can be, how many people can join a room, etc.

**8. Conclusion and Future Work**

The IRC application provides a framework for a basic messaging protocol using

WebSockets and a client-server architecture. It enables users to send group messages,

private messages, create/join rooms, and view who is online.

Some future work includes, including error checking for message transfers, and

integrating load testing to see how many clients the server can handle.

Furthermore, with more implementation, it can easily provide functionalities of secure

messaging and file transfers. It would be interesting to see the application use a

database for persisting user data such as messages, and friends list.