Portland State University
Network Application Protocol
Request for Comments: 594MV

Mina Vu
Winter 2022
Expires: Fall 2022

Mina's Internet Relay Chat House (M.I.R.C.H) Protocol

Status of this memo

This memo defines the protocol for Mina's Internet Relay Chat House (M.I.R.C.H) that is a simple implementation of the Internet Relay Chat Protocol defined in RFC 1459 for the Internetworking Protocol class at Portland State University during the Winter 2022 quarter term. This memo is the final version for submission and will be presented as the final product.

Abstract

The Internet Relay Chat (IRC) protocol is a well-established text-based protocol in which clients can use to communicate with each other over the Internet. This RFC describes a simple implementation of the IRC for an Internetworking Protocol course identified as M.I.R.C.H to represent the student developer's final project.

Table of Contents

1. Introduction

Mina's Internet Relay Chat House (M.I.R.C.H) protocol is a simple Internet Relay Chat protocol implementation for the study of network application protocols. It allows for communication between clients by means of the client-server network architecture running over a TCP connection. In this protocol, clients send messages intending for a "room" of clients or a single other client to the server and the server "relays" that message to all clients in the room.

1.1. Server

There is only one server in the M.I.R.C.H protocol due to the small scale of the application. Working as the backbone, the server is an always-on hub continuously listening for client connection attempts and granting service by creating a connection socket over TCP. The server maintains internal data to streamline all events that occur within the M.I.R.C.H protocol and to provide a sensible service expected of a text-based teleconferencing system.

1.2. Client

There can be zero to many clients in the M.I.R.C.H protocol. A client is considered to be any host connecting to the server and using its services. A client must be identifiable by a unique user ID within the server. For clarity, this means that a client in one room cannot have the same user ID as a client in another room because many functions require clear indications of which client is performing or intended for the action.

2. Metadata

Metadata is kept by both the client and server processes to perform M.I.R.C.H functions appropriately. Specific socket connection data must be kept in sync between the client and server. The server will keep separate metadata for each client connection whereas the client will only keep metadata for its own connection.

2.1. Server

io = the M.I.R.C.H server,

port = the port number that the M.I.R.C.H server is listening on,

```
MY_DATA = {     my_username = string,
                my_rooms = array of strings,
                my_current_room = string,
                my_color = string           },

TEXT_COLORS = [ …strings… ]
```

2.2. Client

socket = the underlying TCP connection to the M.I.R.C.H server,

```
MY_DATA = {     my_username = string,
                my_rooms = array of strings,
                my_current_room = string,
                my_color = string           },

BG_COLORS = [ …strings… ]
```

3. Identifiers

Identifiers are names of clients and rooms in the M.I.R.C.H application. Users cannot pick their own identifiers in this simple implementation of the IRC protocol. Every identifier must be unique to distinguish between clients and also between rooms.

3.1. Usernames

When a client browser successfully makes a new connection with the M.I.R.C.H server, the underlying TCP socket is assigned a random and unique 20-characters identifier. Allowable characters are numbers and letters, both uppercase and lowercase. Hyphens and underscores may also be used. The server will take the first 6 characters and make them all uppercase for use as the username for the client at the new connection socket. The client username will be displayed in the greeting and welcome message as well as the announcements when the client leaves the chat or their system crashes. It will also precede all messages that the client sends to identify the sender.

3.2. Public Rooms

When a public room is created, the M.I.R.C.H server will generate a unique room name from a pool of adjective single words. The name will use formal capitalization. Users cannot choose the name for the room created. If they abhor the given room name, all users can decide to close the room by leaving and creating a new room with a new unique room name.

3.3. Private Rooms

When a private room is created, the client process will combine the usernames of the two users, the requester and requestee, to formulate the private room name. The formula is as follows:

DM__<username>__<username>

In this nomenclature, "DM__" indicates explicitly that this room is intended for direct messaging. The two usernames that follow are separated with two underscores to show the occupants of the room. Note that the usernames in actual room names will not have the greater-than and less-than signs. Usernames in this nomenclature are ordered by the ASCII values of their characters.

4. Events

Events are the operations that are initiated either by the server or the client and are performed in the M.I.R.C.H protocol. All communications between the clients and server are completed by sending events and data content through the TCP socket connections. Below is the list of all events:

4.1. Connection

A connection event is fired when a client browser attempts to connect to the M.I.R.C.H server. The connection uses underlying TCP that opens a socket which persists until either the server or the client disconnects. Each client socket connection to the server is identified with a unique string comprising of letters, numbers, hyphens, and/or underscores. This is saved in the socket.id property.

Data content of event when fired by client:

{ socket: socket metadata object, }

4.2. Disconnect

A disconnect event is fired when either the server or the client intentionally disconnects the other or when either the server or the client experience a system crash. If the event was triggered by an intentional disconnect from the client, the server will display a farewell message and shut down all functions. If the event was triggered by an intentional disconnect from the server, the server will display a goodbye message and shut down all functions. If the event was triggered by an unintentional crash of the client browser, the server will relay the event to all rooms in which the client had participated. If the event was triggered by an unintentional crash of the M.I.R.C.H server, the client will attempt to reconnect as is described in the reconnect event below.

Data content of event when fired by either server or client:

{ reason: string, }

4.3. Reconnect

A reconnect event is fired by the client when the M.I.R.C.H server crashes. Each client browser will display a notice that the server had disconnected and start attempting reconnect. A reconnect event is fired about every half second to check the status of the server. If the server remains disconnected, the client will not be able to reconnect. If the server comes back online, the reconnect event will successfully create a new socket connection to the server. Consequently, in the simple implementation of M.I.R.C.H, state is not persisted between connections therefore data from the previous connection will be lost and a new socket.id and username will associated with the new socket connection.

Data content of event when fired by client:

{ }

4.4. Welcome

A welcome event is fired upon the successful connection of a client browser to the M.I.R.C.H server. The event contains a welcome message, the connection data, and all

room names. When the client receives this event, it will update its own connection data state with data from the server. The client will also update the rooms list to display all rooms and display the welcome message the chat screen.

Data content of event when fired by server:

```
{       welcome_message:  string,
        socket_data:  MY_DATA metadata object,
        rooms:  array of all rooms as string,     }
```

4.5. Announcement

An announcement event is fired by the server in cases where an entire room of users must be notified of a participant leaving intentionally or unexpected. This event contains the room to which the announcement should be displayed and the announcement that should be displayed to the screen. When the client receives this event, it will display the announcement to the correct room.

Data content of event when fired by server:

```
{       room:  string,
        announcement:  string,          }
```

4.6. Message

A message event is fired when there is a message to be relayed. The event is first triggered by the client browser when a user sends a message to a room. The event contains the connection metadata and the message to be relayed. On the server side, when a message event is received, the server updates its connection metadata to match that from the client and fires a message event to all participants in the current room property of the metadata. The content of this event fired by the server contains the original user of the message, the room it is intended for, and the message itself. The M.I.R.C.H server also does a check of the message for the taboo word "cheese". If this word exists, the M.I.R.C.H server eliminates the socket connection effectively firing a disconnect event as described above. Internet Relay Chats may or may not check the message content that it relays; however, this special feature is included in M.I.R.C.H for the sole purpose of showcasing the capability of the M.I.R.C.H server in disconnecting an unruly client. It is not suggested for an IRC application to monitor the content of relayed messages, but censorship of profanity for the purpose of mitigating cyberbullying and observing keywords for eliminating terrorist attacks are topics of controversial debates.

Data content of event when fired by server:

```
{       user:  string,
        room:  string,
        message:  string,         }
```

Data content of event when fired by client:

```
{       socket_data:  MY_DATA metadata object,
        message:  string,         }
```

4.7. Create_room

A create_room event is fired when the client user elects to create a new room. There is no content when this event is triggered as the user cannot choose room names. The M.I.R.C.H server will create a unique room name with capitalization upon receipt of this event. It will have the underlying socket connection join this new room and update the connection metadata. Once joined, the server will emit the joined_room event as described below. The server will fire an announcement event to all connecting sockets in the new room. The server will also fire an update_users_list as described below to the new room. If this is not a private room, the server will fire an update_rooms_list event as described below to all connections.

Data content of event when fired by client:

```
{ }
```

4.8. Join_room

A join_room event is fired by the client when a room is already created and available for any user to join. To join the room, the client must send the room name to the server for processing. The server will have the underlying socket connection join the room indicated by the room name from the client. It will update its connection metadata and fire a joined_room event as described below to the client browser with the updated connection metadata. The server will also announce to all connections in the room that a new user has joined with the announcement event. If it is not a private room, the server will broadcast the current rooms list to all connections with an update_rooms_list event as described below for redundancy. Lastly, the server will fire the update_users_list event as described below to the room the client just joined.

Data content of event when fired by client:

```
{       room:  string,   }
```

4.9. Joined_room

A joined_room event is fired by the M.I.R.C.H server after a client had successfully joined a room either newly created or openly available. The event contains the connection metadata that was updated after the server processed the join request. When the client process receives the event, it updates its own connection metadata and creates a room tab and view on the graphical display of the client application. The client user's current room is automatically set as the newly joined room for any messages that the client sends.

Data content of event when fired by server:

{ socket_data: MY_DATA metadata object, }

4.10. Leave_room

A leave_room event is fired by the client when the user elects to leave a room. This requires the room name to be sent to the server for processing. On receipt of this event in the server process, the M.I.R.C.H server emits an announcement to all participants of that room that the user left the room. The server will update its connection metadata and set the socket's current room property to the last room the client joined. It emits the left_room event as described below to the socket connection. It also fires the update_rooms_list event as described below to all connections for redundancy because the room will automatically close if the socket client was the last participant in the room, and it also fires the update_users_list event as described below to the room.

Data content of event when fired by client:

{ room: string, }

4.11. Left_room

A left_room event is fired by the server in response to the client's request to leave a room and was successfully accomplished. This event contains the updated metadata and the room the client left. When the client receives this event, it knows to update its own metadata to match that of the server and remove the graphical display of the room tab and view screen.

Data content of event when fired by server:

{ socket_data: MY_DATA metadata object,
room_left: string, }

4.12. Create_private_room

A create_private_room event is fired when the client elects to direct message another client user. When this happens, the client process creates a room name by combining the two usernames together and sending to the server process along with the username of the requestee. If the room is already open, the room tab and view simply comes into focus for the client to send messages. The client process prevents the browser from attempting to create a private room for itself. The server receives this event and immediately creates and places the requester into this new room. It then broadcasts the join_private_room_request as described below to all connections with the private room name, the requester's username, and the requestee's username.

Data content of event when fired by client:

{ private_room: string,

other_user: string, }

4.13. Join_private_room_request

A join_private_room_request event is fired by the server in response to a create_private_room request from a client socket as described above. It includes the private room name, the requester's username, and the requestee's username. Every client socket connection will receive this broadcast. Only the intended recipient will process this event. Every other connection will simply ignore it. The intended client socket will display a modal to the browser nudging the requestee to accept or decline the requester's direct message request. If the requestee accepts the join request, the client socket will fire the join_room event with the private room name. Otherwise, the client socket will fire the decline_private_room_request as is described below.

Data content of event when fired by server:

{ request_private_room: string,
request_from_user: string,
request_to_user: string, }

4.14. Decline_private_room_request

A decline_private_room_request event is fired if the client user elects not to join the private room chat with the requester. The contents of this event is the exact same as that of the join_private_room_request. Namely, it contains the private room name, the requester's username, and the requestee's username. Once the server receives this event, it will announce to the requester of the declination by the requestee with an announcement event as described above. The requester is not kicked out of the open private room and can choose to leave anytime.

Data content of event when fired by client:

{ request_private_room: string,
request_from_user: string,
request_to_user: string, }

4.15. Update_rooms_list

An update_rooms_list event is fired every time there is a perceived change to the list of available open rooms. The server gathers a list of all open rooms and sends this data to all connecting sockets. Private rooms are not included in the list. The client updates the rooms list in its graphical user interface.

Data content of event when fired by server:

{ rooms: array of all rooms as strings, }

4.16. Update_users_list

An update_users_list event is fired every time there is a change to the list of users in a particular room or when there is a new connection to the M.I.R.C.H server. When a user creates, joins, or leaves a room, this event is fired by the server to each room in question. When a user connects to the server, this event is fired to the default room. When a user is disconnected from the server, this event is fired to the default room.

Data content of event when fired by server:

```
{       room:  string,
        users:  array of all users in room as strings,      }
```

5. Rooms

Rooms signify a group of clients communicating with each other through the M.I.R.C.H application. A room is created when at least one client wants to enter the room. A room is open once it is created. A room is closed when the last client occupying the room leaves. Each room will have a unique name managed by the server. Clients cannot rename a room.

5.1. Default Room

The default room in M.I.R.C.H is the Lobby. All client connections are automatically placed into the Lobby upon successful connection to the M.I.R.C.H server. In the Lobby, clients view a welcome message that tells them to select a room to start chatting with others. Clients cannot chat in the Lobby as this function is disabled. Clients can view all users connected to M.I.R.C.H when in the Lobby and can request direct message to a user. Clients cannot choose to leave the Lobby room.

5.2. List Rooms

Clients will always see the list of all rooms. This list contains all open public rooms and indicate the room in which the client has already joined. Clients can select any room in this list to join and start chatting with the participating users. Once joined, the client will see an updated list with indication that the client is participating in the room. Any action that opens a new room or closes an existing room will update the list of all rooms for every client connection to view.

5.3. Create Room

The very first client connection to the M.I.R.C.H server must create a new room to starting chatting because the only room already open is the Lobby room where no users can chat. In creating a new room, the client will be automatically placed into the room. In other words, the client creating the room will be the first client to join the room. All subsequent client connections to M.I.R.C.H will be able to create new rooms. At any time, a client can create a new room and join for relayed chatting.

5.4. Join Room

When there are public rooms open, clients can elect to join a room for chatting. Public rooms are open to all M.I.R.C.H users without restriction. Clients can join as many rooms as they please. Chat messages directed to a joined room will only be viewable by the existing members of the room. Newly joined participants will not be able to view past messages. If a client system crashes and the client reconnects to M.I.R.C.H and the room, they will not see old messages. When a client joins a room, the server will announce this action to all users in the room.

5.5. Leave Room

Besides the default Lobby room, a client can choose to leave any room in which they are participating. The action will remove the client from the room and announce that the client has left to all remaining users in the room. If the client is the last member in the room, choosing to leave the room will ensure the closure of the room permanently. The room name can subsequently be reused for a new room.

5.6. Private Room

Private rooms are opened when a client attempts to direct message another client user. Upon opening, the requester of the direct message will immediately join the private room and wait for the requestee to accept or decline the request. The server will announce to this private room when the requestee respond to the request by either stating that the client has joined the room or has declined the request. Private rooms are not open to the public and therefore cannot be seen in the list of all rooms. Only up to two client users are allowed in a private room. One user is in the room in the case where the request to join the private room was declined by the requestee.

6. Graphical User Interface

The M.I.R.C.H application has a graphical user interface component to allow easy and enjoyable access for clients. It is not a requirement for this final project in the Internetworking Protocol course at Portland State University. However, it neatly presents the different components of M.I.R.C.H features for a better user experience.

6.1. M.I.R.C.H Title Bar

The M.I.R.C.H title bar is seen at the top of the application with a welcome message to M.I.R.C.H. When the client chooses to exit the application, the title bar will extend the entire viewport and display a farewell message. When the server chooses to disconnect the client from the application, the title bar will extend the entire viewport and display a goodbye message.

6.2. All Rooms View Window

To the left of the screen below the M.I.R.C.H title bar is the window that displays all open public rooms in M.I.R.C.H. The very first room listed is the default Lobby room in which all clients are automatically joined upon successful connection to the server. All other public rooms are displayed after the Lobby. After all public rooms is the create new room option that clients can select to create and join a new room. Below that is the exit M.I.R.C.H option for clients to leave the application altogether. Finally, there is a warning message telling clients that the server will disconnect the client connection if a buzz word is typed in the chat rooms.

6.3. Joined Rooms Tabs Section

The tabs section is seen immediately to the right of the all rooms view window. There is a tab for every room the client is in beginning with the Lobby tab. The tab for the currently active viewing room will be colored to draw focus. Other tabs, in any exists, will be shaded in gray to draw attention to the active tab. Room tabs are appended to this section by the order in which the rooms are joined. When clients leave a room, the room tab will also disappear from this section. Clients can click on any shaded room tab in this section to switch their currently active chat room view window.

6.4. Chat Room View Windows

Right of the tabs section is the chat room view window. Every room that the client joins will have its own chat room view window color-coded the same as the associated room tab in the tabs section. Only one view window is seen at a time, and the room associated with the viewed window is the currently active room. This allows for a larger view window to show more chat messages in a room. The individual chat room view window is comprised of three sections: the room name at the top section; the chat screen in the middle section; and the chat box and send button at the bottom section. New messages from users and announcements from the server are appended to the chat screen. The chat screen automatically scrolls to view the new messages when there is overflow. Automatic scroll is paused when the client scrolls up in the chat screen and resumed when the client scrolls down to the end of the chat screen. Clients type their messages into the chat box and press send to relay that message to other users in the same room.

6.5. Users View Windows

The users view window is located right-most of the viewport below the M.I.R.C.H title bar. The number of users view windows matches the number of rooms the client is joined. Users view window is associated with each room and displays all users participating in that room. The users view window for the Lobby will show all client users connected to M.I.R.C.H. Only one users view window is presented at a time, and that is the window belonging to the currently active room. All users view windows will be updated automatically when there is a change in the users list notified by the server. This happens when a client connects to or disconnects from the M.I.R.C.H server or when a user joins or leaves a room

6.6. Buttons

Existing buttons in the M.I.R.C.H application allows the user to select a function with a quick mouse click. Buttons with different functions are also color-coded differently to draw different focus from the user.

6.6.1. In

The In button is color-coded with a green outline. Users see this button in the all rooms view window. Each room that the user has joined will have an In button beside the room name in the all rooms view window. Clicking on an In button will make the associated room into the currently active room. This also brings the associated tab, view window, and users window into focus.

6.6.2. Join

The Join button is color-coded with a blue outline. Users see this button in the all rooms view window next to every open public room of which the user is not a participant. Clicking on the Join button next to a room name will tell the server to add the client connection to that named room so that the user can start relay chatting with other participants. After this action, a new tab, view window, and users window are created and associated with the room name. They are immediately brought into focus for the user to start chatting.

6.6.3. Create

The Create button is color-coded with a solid red. Users see this button in the all rooms view window. There is always only one Create button next to its 'New Room' label in each client browser. Clicking on the Create button will tell the M.I.R.C.H server to generate a new unique room name. Once that is done, the client is automatically joined in the new room and the associated tab, view window, and users window are created and brought into focus.

6.6.4. Exit

The Exit button is color-coded with a solid black. Users see this button in the all rooms view window, and there is only one Exit button in each client browser. Its label is M.I.R.C.H indicated that a button click on Exit will exit the user and disconnect the client connection to the M.I.R.C.H server. Users can click this button when they want to exit the entire application.

6.6.5. Leave

The Leave button is color-coded with a black outline. Users see this button in the chat room view windows next to the room title in the top section. Clicking this button will tell the server that the client wants to leave the room and therefore no longer see any subsequent messages sent to the room. Once the user click to Leave a room, the

associated tab, view window, and users window will be removed from the client browser. The last room that the client had joined will come into focus.

6.6.6. Send

The Send button is color-coded with a solid green. Users see this button in the chat room view windows at the bottom section next to the chat box. Clicking this button will send the message in the chat box to the M.I.R.C.H server to relay to all participants in the same room. Once clicked, the chat box will be cleared and ready for the next message. Every chat room view window has a Send button associated with the room name for proper message relaying. The default Lobby room also has a Send button but it is disabled because clients cannot send messages to the Lobby room.

6.6.7. Self

The Self button is color-coded with a gray outline. Users see this button in the users view window next to their own username. It is a disabled button and therefore has no function and cannot be clicked. The sole purpose of Self is to show the user their own username inside every users view window.

6.6.8. DM

The DM button is color-coded with a blue outline. Users see this button in the users view window next to every username displayed. Clicking this button will tell the server the client wants to send a direct message request to the username that is associated with the clicked DM button. A private room will be created and the client is joined immediately to wait for the other user. The server will relay the request to the other user. If the other user accepts the request, they will join the private room and can start chatting. If the other user declines the request, the server will notify the requester waiting in the private room.

6.6.9. Tabs

Each tab in the joined rooms tabs section is essentially a button that is color-coded with the same color as the room name heading of its associated chat room view window. Users will see the color when it is of the currently active room otherwise the tab will be shaded gray. Clicking on a tab will bring the tab, associated view window, and associated users window into focus as the currently active room.

7. Error Handling

7.1. Server Crash

In the event that the M.I.R.C.H server crashes, a disconnect event is implicitly sent to the clients. Because this is an unintentional disconnect by the server, the client processes will attempt to reconnect with timely pings. If a reconnect event is successful, the client browser

will reload the application. Reconnect attempts will continue until the user exits the client browser.

7.2. Client Crash

In the event that the client system crashes, a disconnect event is implicitly sent to the server. The server sends an announcement event to all rooms that the client was in before the crash using its metadata that is sync with the client connection data. The server will send an update_users_list to each room.

8. Extra Feature

Private messaging is supported in M.I.R.C.H where two users can chat with each other in a separate room that no other users can see. One user must first request to direct message another and enter the private room. The other user will be notified of the request and can either accept or decline. Accept means the client sends a join_room event to the server, and the server adds them to the private room. Decline means the client sends a decline_private_room_request event to the server, and the server will inform the first user of the declined request.

9. Conclusion

This RFC describes the network protocol of Mina's Internet Relay Chat House (M.I.R.C.H) application. It is a simple implementation of the well-established Internet Relay Chat protocol as described in RFC 1459. Features of M.I.R.C.H include creating, joining, and leaving chat rooms as well as the added extra feature of private messaging between two users. The application uses TCP for reliable data transport. M.I.R.C.H does not provide a secure connection between client and server therefore sensitive communication must be weary of this trait.

10. IANA Consideration

None

11. Acknowledgements

RFC 1459 Internet Relay Chat Protocol by J. Oikarinen and D. Reed, May 1993

CS594SampleRFC Internet Relay Chat Class Project, December 2015