Request For Comments

May 2020

Revision 1 - 23/04
Revision 2 - 19/05

THE LIRC PROTOCOL (REVISION 2)

Status of this Memo

This RFC specifies a students project protocol for implementing a chat service using two modes of communication.
The first is a broadcast mode, which delivers to all connected clients a message sended by a client.
And the second one is like a unicast mode between two clients. Clients using this mode can send private messages or files to each other without requiring being relayed by the server.

Summary

LIRC stands for Legal Internet Relay Chat.

LIRC is a protocol used to facilitate communication in the form of text. The chat process works on a client/server networking model. However, unlike with IRC, this protocol allows clients to exchange files and private messages between them. In this case, the server is only here to exchange addresses between clients.

This document describes the protocol and its types of packets. The document also explains the reasons behind some of the design decisions.

Acknowledgements

The protocol was originally designed by two students, Judicaël KOFFI and Christelle NGUYEN.

1. Purpose

LIRC is a protocol to communicate in the form of text, and is implemented on top of the Transmission Control Protocol (TCP).
It allows a client to connect to a server and send messages to all connected clients on this server. LIRC also allows direct communication with another connected client, without going through the server.
A private connection between two clients is mandatory as the client asks the server to negotiate the access to target client information, before performing a direct connection with the client.

2. Overview of the protocol

Any access to chat service begins with a connection request in anonymous (only login is mandatory) or authentication mode with a login and a password from a client to the server. The login has to be unique. Any message used in this protocol is encoded with standard charset UTF-8.

When the authentication is done, the client can decide to chat either with all connected clients or, only with a specific client privately. He can also send files to a client in particular.

The client can disconnect from the chat too. To be disconnected, the client must send a disconnection request to the server.

3. Relation to other Protocols

LIRC protocol is designed to be implemented on top of Transmission Control Protocol (TCP). TCP is a connection oriented protocol, so it is assured some verifications and assurance for exchanges occurred in the network.

TCP guarantees that all bytes received will be identical and in the same order as those sent.

Please refer to this link, to learn more about TCP: https://tools.ietf.org/html/rfc793

4. Initial Connection Protocol

To establish a connection with the server, a client must send a request of connection.
There are two types of connection request:

- an anonymous connection where a password is not mandatory, only the login is
- an authenticated connection where both password and login are mandatory

Clients will receive packets from the server where the first byte contains opcode 15 followed by a byte (1 or -1) which indicates an error occurred or not during connection.
All packets with this opcode contain a string encoded in UTF-8 to give a description of the response.

Then, once the connection is established, the client can do the following things :

A. For exchanges between CLIENT and SERVER

- To send a message to all the clients connected to the server (as described above), like a broadcast message.

- To send a request to perform a private connection to another client (let us say client A to client B). The server then has to relay the request of connection to client B and if client B accepts a private connection request, the server sends the address of client B(IP and port) to client A, and client A can perform private connection to client B.

B. For exchanges between CLIENT and CLIENT:

When the connection is established, the client is able to :

- Send private message
- Send file

All exchanges packet structure will be detailed in the following part.

5. LIRC Packets

LIRC supports 14 types of packets, all of which have been mentioned above.

opcode	operation
///////// CLIENT -> SERVER ///////////	
0	Client requests to server an anonymous connection
1	Client requests to server an authenticated connection
2	Client sends a message to server to be broadcasted for all connected clients
3	Client requests server to have a private connection with another client
4	Client positive response to server for an incoming private connection request
5	Client requests for disconnection to server
-4	Client negative response to server for incoming private connection request
///////// CLIENT -> CLIENT ///////////	
8	Client private message send
9	Client sends private file
///////// SERVER -> CLIENT ///////////	
2	Server sends broadcast message to a client
3	Server notify a targeted client for incoming private connection to him
4	Server positive response to sender client for private connection request
15	Server send an message (error or information) to client
-4	Server negative response to sender client for private connection request

All the long type used in figures are on 8 bytes and in big endian.

All the int type used in figures are on 4 bytes and in big endian.

All the String type are encoded in UTF-8.

All the targets described here are the ones who will receive the packet.

5.a From CLIENT to SERVER

Opcode 0 : Client requests to server an anonymous connection

```
1 byte    long          String

------------------------------------------
| 0 |  login length |   login (UTF8) |
------------------------------------------
```

Figure 5-1: anonymous connection request packet

- login length : a long number used to represent the client login’s length
- login : a string used to identify a client in the chat

Opcode 1 : Client requests to server an authenticated connection

```
1 byte   long            String           long            String

------------------------------------------------------------------------------
| 1 |  login length |   login (UTF8)  | pass length |   pass(UTF8) |
------------------------------------------------------------------------------
```

Figure 5-2: authenticated connection request packet

- login length : a long number used to represent the client login’s length
- login : the client’s login, in String, who wants to be authenticated to access to the chat
- pass length : a long number used to represent the client password’s length
- pass: it is the client’s password in String

Opcode 2 : Client sends a message to server to be broadcasted for all connected clients

```
1 byte   long                    string                  long              string
---------------------------------------------------------------------------------------------
| 2 | from login length |  from login | message length | message (UTF-8) |
---------------------------------------------------------------------------------------------
```

Figure 5-3 : message to be broadcasted from client to server packet

- from login length : it is the length in long, of the sender client who initially wanted to broadcast his message
- from login : it the client's login in String who initially wanted to broadcast his message
- message length: the message's length sent by the client in long
- message : it is the message itself in String

Opcode 3: Client requests server to have a private connection with another client

```
1 byte  long                         String            long               String
--------------------------------------------------------------------------------------
| 3 | target login length | target login | from login length |  from  login|
--------------------------------------------------------------------------------------
```

Figure 5-4: request private connection to the server packet

- target login length : it represents the length of the target's login in long
- target client login : it represents the target client's login in String
- from login length : it represent the login's length of the client sending the request to the server in long
- from client login : it is the client's login sending the request to make a private connection in String

Opcode 4: Client positive response to server for an incoming private connection request

```
 1 byte      long            String       long   long   long   long                     String

--------------------------------------------------------------------------------------------------------
| 4 | target login length | target login | ip | port | token | from login length | from login  |
--------------------------------------------------------------------------------------------------------
```

Figure 5-5: client positive response packet

- target login length: target's login size of private connection
- target login: login of private connection target (a string encoded in UTF8)

- ip: a long in big endian representing ip address
- port: a long representing a port number
- token: a long representing a token to certify connection between clients
- from login length : it represents the login's length in long of the client who originally sent the request to make a private connection with the current client (long)
- from client login : it is the client's login who originally sent the request to make a private connection with the current client in String

Opcode 5 : Client requests for disconnection to server

```
1 byte     long                  String

   -------------------------------------------
   | 5 |  from login length | from login |
   -------------------------------------------
```

Figure 5-6 : client sending disconnection packet

- from login length : it represents the login's length in long of the client who wants to disconnect from the chat
- from client login : it is the client's login who wants to disconnect from the chat in String

Opcode -4 : Client negative response to server for incoming private connection request

```
  1 byte       long                    string                      long               string

-----------------------------------------------------------------------------------------------------
| -4   |  target login length | target login length | from login length | from login length|
-----------------------------------------------------------------------------------------------------
```

Figure 5-7: client negative response packet

- target login length: target's login size of private connection
- target login: login of private connection target (a string encoded in UTF8)
- from login length : it represents the login's length in long of the client who originally sent the request to make a private connection with the current client
- from client login : it is the client's login who originally sent the request to make a private connection with the current client in String

5.b From CLIENT to CLIENT

Opcode 8 : client sends private message

```
 1 byte          long                string              long                string

--------------------------------------------------------------------------------------------------
| 8 |  target login length | target login | message length | message (UTF-8) |
--------------------------------------------------------------------------------------------------
```

Figure 5-8 : client sending private message packet

- target login length: target’s login size of private message
- target login: login of private message target (a string encoded in UTF8)
- message length : it is the size of the whole message sent. It is represented by a long type
- message: the message corresponds to the content of the message itself in String

Opcode 9 : client sends private file

```
byte     long                String              long     byte
---------------------------------------------------------------------
| 9 | filename length | filename (UTF8) | data length| data |
---------------------------------------------------------------------
```

Figure 5-9 : client sending file packet

- filename length :After opcode, there is a long value in big endian which represents the filename’s size (encoded in UTF-8)
- filename: it is a string representing the filename
- Number of chuck is in big endian, long, to indicate how many times the file content was subdivided
- chunk1 … chunkN are file content in String

5.c From SERVER to CLIENT

Opcode 2 : Server sends broadcast message to a client

```
 byte  long                 string                      long              string
  -----------------------------------------------------------------------------------------------------------
 | 2 | from login length | from login (UTF8)   | message length | message (UTF-8)   |
  -----------------------------------------------------------------------------------------------------------
```

Figure 5-10 : message broadcasted from server to clients packet

- from login length : it is the length in long, of the sender’s client who initially wanted to broadcast his message
- from login : it the client’s login in String who initially wanted to broadcast his message
- message length : it is the message’s length in long
- message : it represents the message itself in String

Opcode 3 : Server notify a targeted client for incoming private connection to him

```
   byte      long                    String             long              String
   -----------------------------------------------------------------------------------------------
  | 3 | target login length | target login | from login length |  from login |
   -----------------------------------------------------------------------------------------------
```

Figure 5-11: request incoming private connection to target client packet

- target login length: login size of target of private message
- target login: login of private message target (a string encoded in UTF8)
- from login length: it is the client’s IP address’s length who requested a private connection (in long).
- from login: it represents the client’s login who requested a private connection (String)

Opcode 4: Server positive response to sender client for private connection request

```
 1 byte      long             String        long   long   long        long                  String

-------------------------------------------------------------------------------------------------------------------
| 4 | target login length | target login | ip | port | token | from login length | from login  |
-------------------------------------------------------------------------------------------------------------------
```

Figure 5-12 : server positive response to request private connection packet

- target login length: target’s login size of private connection

- target login: login of private connection target (a String encoded in UTF8)
- ip: a long in big endian representing an ip address
- port: a long representing a port number
- token: a long representing token to certify connection between client
- from login length : it represents the login's length in long of the client who accepted the request to make a private connection (in long)
- from login : it is the client's login who originally accepted the request to make a private connection (in String)

Opcode 15 : server response packet

In case of connection error, a server sends a packet, where the opcode is 15, with the following structure (figure 5-14) to indicate a type of error :

```
1 byte   byte              long               String

----------------------------------------------------------------------
| 15 | errorOrNot | message length  | message (UTF8)  |
----------------------------------------------------------------------
```

Figure 5-13: server error packet

- errorOrNot : it is a byte which can represents -1 or 1
- message length : it is a long value in big endian representing the length of a message which describes the error encoded with UTF-8 charset.
- message : the packet contains a string representing error messages with UTF-8 encoding.

For example, when a client requests a connection to a server with a login which is already attributed, the server responds with error packet and message like “Login unavailable”.

Opcode -4: Server negative response to sender client for private connection request

```
  1 byte      long                    String                        long              String

-------------------------------------------------------------------------------------------------
| -4   |  target login length | target login length | from login length | from login|
-------------------------------------------------------------------------------------------------
```

Figure 5-14: server negative response to private connection packet

- target login length: login size of target of private connection
- target login: login of private connection target (a string encoded in UTF8)

- from login length : it represent the login’s length in long of the client who rejected the request to make a private connection (in long)
- from login : it is the client’s login who originally rejected the request to make a private connection with the current client (in String)

6. Normal termination

The end of a connection between a client and the server or between two clients is marked by a disconnection request sent by a client who wants to be disconnected. To see the structure of this packet, please see figure 5-6.

Author's email address

Christelle NGUYEN

Judicaël KOFFI

Email:

- christelle.nguyen97@gmail.com
- judkoffi@gmail.com