Apprenticeship Network Programming Group
Request for Comments: 9912
Category: Experimental

Y. Rameau
M. Bouazza
UGE

04/05/2021

ChatOS Protocol : Anonymized direct TCP chat service

Abstract

This note describes the ChatOs Protocol (COP), a protocol used in ChatOS Application to route messages in TCP. COP can provides private connexion between two users or a public connexion to send messages for all users.

Summary

The ChatOs protocol is a very simple protocol used to communicate with other users. This document describes the protocol and its types of packets. The document also explains the reasons behind some of the design decisions.

Table of Contents

1. Purpose

The ChatOs Protocol is intended to be used for the transmission of messages between clients. It may also be used for the local message system of a network or host.

The focus here is on the internal mechanisms to transmit messages, rather than the external interface to users. These will be both new programs designed to work with this system and old programs designed to work with earlier systems.

2. Overview of the protocol

2.1. Clients

A client is defined in the eyes of the server by a nickname and an IP address. The nickname is chosen by the user during his first connection, it must not be larger than the maximum size imposed by the server and must not be offensive or rude. Each user has his own nickname and must therefore be unique for better identification.

2.2. Client identification

Clients are subject to a level of identification. An nickname lookup (and reverse check on this) is performed for all connections made to the server in order to avoid collisions with other nicknames which can lead to identification difficulties. Users are currently not subjected to a password check. Even if these checks are possible on all connections although the password check is currently not being set by the protocol.

2.3. Message

Server and clients send each other messages wich may or may not generate a reply.

When a server receive a message, it MUST identify its source using (eventually assumed) nickname. If the nickname cannot be found in the server’s internal database, it then is registered if it is valid [See section 2.2]

ChatOs messages are always lines of characters and these messages SHALL NOT exceed 512 characters in length. There is no provision for continuation message lines.

2.4. One-to-one communication

Communication on a one-to-one basis is usually only performed by clients. To send a private message you have to choose to whom to send the message. In order to provide a secure means for clients to talk to each other, it is required that the server is able to send a message in exactly one direction in order to reach any client.

Example : A message between clients 1 and 2 is only seen by server A, which sends it straight to client 2.

With this way, this provide a secure manner of exchanging messages with another client.

2.5. One-to-all

The one-to-all type of message is better described as a broadcast message, sent to all clients. On a large network of users, a single message can result in a lot of traffic being sent over the network in an effort to reach all of the desired destinations.

2.6. Private connection

The establishment of a private connection between 2 users is achieved by creating a TCP bridge relayed by the server between 2 clients to guarantee permanent anonymity. Through this feature it is possible to exchange all types of information (messages, files, etc...).

2.7. Client commands

all the below-mentioned commands are automatically parsed at each new instance

if the command does not start with / or @, it is interpreted as a public message.

if the command is of the form @login message, the rest of the command is interpreted as a private message for the login user.

if the command is of the form /login file, the client will make a private connection request for the login client if this connection has not already been established. Once the connection is established, it will make a GET file request in the HTTP protocol. If file has the extension txt, it will be displayed and if not, it will be saved in the directory specified at the start;

At both ends of a private connection, the client behaves like an HTTP server: it responds to GET requests by serving the resources in the directory specified as an argument.

Note: The commands available to clients depend on the context they are in to avoid sending packets that are inconsistent with the client's state.

For example, a client cannot select the option to send a public message without being connected to the server first

3. Implementation details

3.1. Data representation:

Integers (Int) are 4 signed bytes and Bytes (Byte) represents bytes. Strings (String) are encoded in UTF-8 and preceded by the size of their representation in bytes on an Int in every packets which are all transmitted in BigEndian.

3.2. Introduction

To ensure the anonymity of each clients, the server redirects the messages provide by the sender with it’s own connection to the receiver.

3.3. OP packet

OP Code	Description
0	Connection request with the server
1	Connection acceptance
2	Connection refusal
3	Connection request with a specific user
4	Public message
5	Private message
6	Unknown user

3.4. Identification

In ChatOS the first step for an user is to start a connection with the server. The user send a packet with the OPCode 0:

Byte	Int	String
0	nickname_size	nickname

If the nickname is already use by an other user or it’s does not respect the rules (too long, prohibited characters), the server does not allowed the connection and send a connection refusal packet.

Byte	Int	String
2	sender_size	sender

However, if the server accept then it send a connection acceptance packet indicating the client that he is free to come and now connected.

```
                     Byte         Int           String
                  +-------+-----------------+------------+
                  |   1   |   sender_size   |   sender   |
```

3.5. Private connection with an other client

The ChatOS specificity is to have establish a private connection between two users.

For this, the client will first ask the other client and send a private connection request packet through the server which will send it to the other client after that:

```
    Byte      Int           String         Int           String
   +---+-------------+--------------+-------------+----------------+
   | 3 | S_nick_size | sender login | R_nick_size | receiver login |
```

If it’s a negative response, the client send back a connection refusal packet to the server which is responsible to transmit it to the sender client.

```
                  Byte         Int           String
               +-------+-----------------+------------+
               |   2   |   sender_size   |   sender   |
```

However if the response is positive then the receiver client send a connection acceptance packet to the server who transmit this packet to the sender client enabling now a new transparent TCP connection between the two client passing through the server.

```
                  Byte         Int           String
               +-------+-----------------+------------+
               |   1   |   sender_size   |   sender   |
```

If the user cannot be found by the server then the user receives a packet with OpCode 6 from the server indicating that the recipient cannot be found

```
                  Byte          Int           String
               +-------+-----------------+------------+
               |   6   |   sender_size   |   sender   |
```

3.6. Send a public message

When a client is connected into ChatOs, he can send a message for all connected users.

The client send a packet with the OP code 6 and his message coded in UTF8 charset:

```
    Byte      Int          String         Int         String
   +---+-------------+--------------+--------------+---------+
   | 4 | S_nick_size | sender login | message size | message |
```

Then the server broadcast it for each connected users.

Finally the clients catch and decode it to have the message and the sender’s nickname.

3.7. Send a private message

To send a private message, an user need to be connected on the server and be aware of the other user's nickname. If it have these two conditions, the client can send a private message on the private connection with the code 4. (For more detailed description, refer to the section 2.4 One-to-One communication)

```
    Byte      Int    String     Int       String     Int     String
   +---+------------+-----+------------+----------+--------+---------+
   | 5 | exp_l_size | exp | rcv_l_size | receiver | m_size | message |
```

The server at the reception of the packet checks if the recipient's login is correct, if this is the case it sends the packet to the recipient afterwards.

If the user cannot be found by the server then the user receives a packet with OpCode 6 from the server indicating that the recipient cannot be found

```
               Byte          Int           String
            +-------+-----------------+------------+
            |   6   |   sender_size   |   sender   |
```

4. Normal Termination

The client is responsible for the termination. When a client want to end the connection with ChatOs server, it close all the connections which it is involved. That implies, the public connection and all privates connections. The others clients having a connection with him will be notified of his departure.

5. Premature Termination

A premature termination will be occurred in some cases:

- When a client send a message or a file request without authentication with the server. In this case, the server send an ERROR packet and the TCP connection is closed.

- When a client has been inactive for more than 5 minutes he is disconnected. The others clients having a connection with this client will be notified of his departure. The server send an ERROR packet and the TCP connection is closed.

6. Known concerns

6.1. identification

Since the lifetime of a pseudonym for the client depends on the duration of its connection to the server, it is easy once the connection is closed to usurp the pseudonym by connecting from a new address but with the same name.

From the server's point of view, the client will not be the same because the name/address:port association is different, but as some of this information is hidden from other clients, they will only have access to the usurper's nickname and will not be able to tell the difference.

To overcome this problem, we would have to implement an SSL/TLS handshake protocol, but the goal of the chatOs protocol is to be simple to implement, which is why we do not use this type of security protocol.