AMARU Marie

SERO Ruben

ChatOS RFC

Groupe 2

Protocole ChatOS

Ce document décrit le protocole ChatOS. Le protocole permet à des clients de communiquer avec un serveur Chaton. Via ce serveur, les clients peuvent s'échanger des messages et demander à établir des connexions privées avec les autres clients connectés. Ces connexions privées sont relayées par le serveur de sorte qu'un client n'apprend jamais l'adresse IP des autres clients connectés. L'ensemble des communications entre clients et serveurs se feront au dessus de TCP.

Représentation des données:

Les entiers (INT) sur 4 octets signés et les longs (LONG) sur 4 octets signés sont tous transmis en BigEndian.

Les chaînes de caractères (STRING) sont encodées en UTF-8 et précédées de la taille de leur représentation en octets sur un INT.

    STRING = taille (INT) chaîne encodée en UTF-8

Commande Client<->Serveur

La communication entre les clients et le serveur et entre les clients sur leurs connexions privées se font au moyen de COMMAND. Toutes les commandes commencent par un entier signé sur un octet (OPCODE) qui donne le type de la commande.

Paquets ChatOS

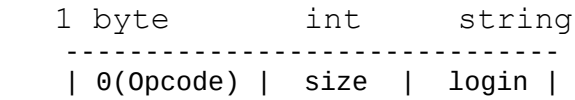
OpCode	Operation
0	Identification with login
1	Login Accepted
2	Login Refused
3	Private Message
4	Global Message
5	Private Connection Asked
6	Private Connection Accepted
7	Private Connexion Refused
8	Connect ID Given
9	Connect ID Entered
10	Private Connexion Established
11	Request Close Private Connection
12	Close Private Connection

1) Identification

La première étape du protocole dans la communication entre le client et le serveur est l'identification. Le client qui se connecte au serveur doit proposer son login. Le serveur, après avoir vérifié que ce login n'est pas utilisé par un autre client connecté, peut accepter l'identification. Dans le cas contraire, il la refuse et le client doit en proposer une nouvelle.

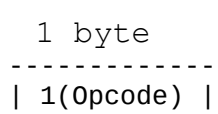
La proposition du login par le client se fait par la commande LOGIN(0) d'OPCODE 0 suivi d'une STRING contenant le login. Cette string ne peut pas occuper plus de 30 octets.

LOGIN(0) = 0 (OPCODE) login (STRING)



Si l'identification est acceptée, le serveur renvoie la commande LOGIN_ACCEPTED(1) d'OPCODE 1. Si l'identification est refusée, il renvoie la commande LOGIN_REFUSED(2) d'OPCODE 2.

LOGIN_ACCEPTED(1) = 1 (OPCODE)



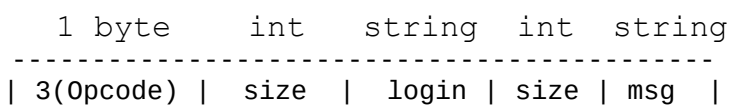
LOGIN_REFUSED(2) = 2 (OPCODE)

Seule les commandes LOGIN(0) peuvent être traitée par le serveur tant que le client qui n'est pas identifié avec succès (à l'exception des commandes LOGIN_PRIVATE(9), voir partie 4). L'identification ne peut avoir lieu qu'une seule fois avec succès.

2) Fonctionnalité de chat

Pour envoyer un message à tous les clients connectés, un client envoie une commande MESSAGE(3) d'OPCODE 3.

MESSAGE(3)=3(OPCODE) login(STRING) msg(STRING)



Le login doit être le login utilisé lors de l'identification. Le message msg ne peut pas occuper plus de 1024 octets.

Après la réception d'une commande MESSAGE(3), le serveur doit transmettre cette commande à tous les clients connectés.

Donc quand un client reçoit du serveur une commande MESSAGE(3), elle doit être comprise comme un message général envoyé par le client login.

Pour envoyer un message à un unique client de login login_target, un client de login login_sender envoie une commande MESSAGE_PRIVATE(4) d'OPCODE 4.

MESSAGE_PRIVATE(4) = 4 (OPCODE) login_sender (STRING) login_target (STRING) msg (STRING)

1 byte	int	string	int	string	int	string
4(Opcode)	size	login_sender	size	login_target	size	msg

Le login_sender doit être le login utilisé par l'émetteur lors de l'identification.
Le login_target est le login d'un autre utilisateur.
Le message msg ne peut pas occuper plus de 1024 octets.

Après la réception d'une commande MESSAGE_PRIVATE(4), le serveur doit transmettre cette commande au client login_target s'il est connecté. Si aucun client connecté n'a ce login_target, la commande est ignorée par le serveur.

Donc quand un client reçoit du serveur une commande MESSAGE_PRIVATE(4), elle doit être comprise comme un message personnel envoyé par le client login_sender.

3) Négociation d'une connexion privée

Lorsque un client requester veut établir une connexion privée avec le client target, il fait une demande qui va transiter par le serveur.

Il envoie au serveur une commande REQUEST_PRIVATE(5) d'OPCODE 5.

REQUEST_PRIVATE(5) = 5 (OPCODE) login_requester (STRING) login_target (STRING)

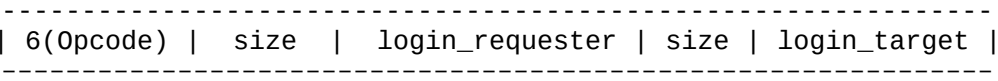
1 byte	int	string	int	string
5(Opcode)	size	login_requester	size	login_target

Le champ login_requester doit être le login utilisé à l'identification par le client requester qui émet la requête. Le champ login_target est le login du client target sollicité.

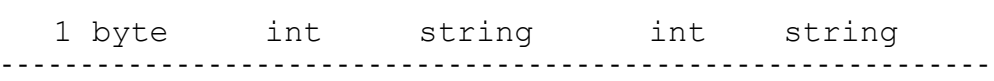
Le serveur transfère cette commande exclusivement au client target. Si aucun client connecté n'a cet identifiant, la commande est ignorée.

Si un client target reçoit une commande REQUEST_PRIVATE(5) du serveur, il peut soit accepter d'établir une connexion privée avec la commande OK_PRIVATE(6), soit la refuser avec la commande KO_PRIVATE(7). Ces commandes sont envoyées par le client target au serveur.

OK_PRIVATE(6) = 6 (OPCODE) login_requester (STRING) login_target (STRING)



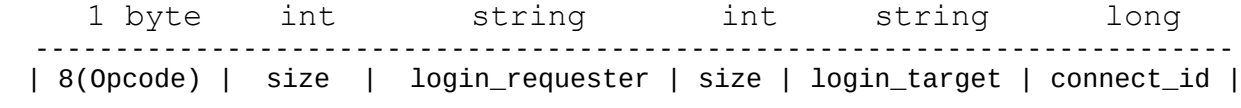
KO_PRIVATE(7) = 7 (OPCODE) login_requester (STRING) login_target (STRING)



Si la commande est KO_PRIVATE(7), celle-ci est transmise par le serveur au client requester.

Dans le cas d'une commande OK_PRIVATE(6), le serveur renvoie aux deux clients, requester et target, une commande ID_PRIVATE(8) leur permettant de connaître le connect_id de cette connexion privée négociée.

ID_PRIVATE(8) = 8 (OPCODE) login_requester (STRING) login_target (STRING) connect_id (LONG)

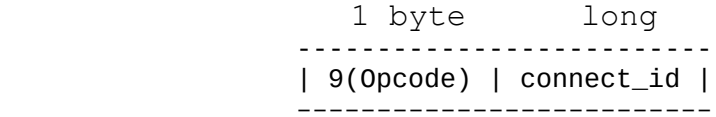


Le LONG connect_id est un identifiant unique, propre à cette connexion privée négociée, qui sera utilisé par les deux clients quand ils vont ensuite "établir" la connexion privée.

4) Établissement de la connexion privée

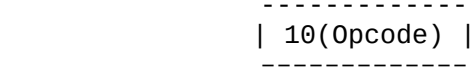
Deux clients peuvent établir une connexion privée en se connectant au serveur avec chacun une nouvelle connexion TCP sur laquelle ils commencent par envoyer la commande LOGIN_PRIVATE(9) contenant le connect_id obtenu au moment de la négociation de la connexion privée.

LOGIN_PRIVATE(9) = 9 (OPCODE) connect_id (LONG)



Quand le serveur a accepté deux connexions TCP qui présentent le même connect_id (et que ce connect_id a été distribué par le serveur), il envoie une commande ESTABLISHED(10) sur ces deux connexions à chacun des clients.

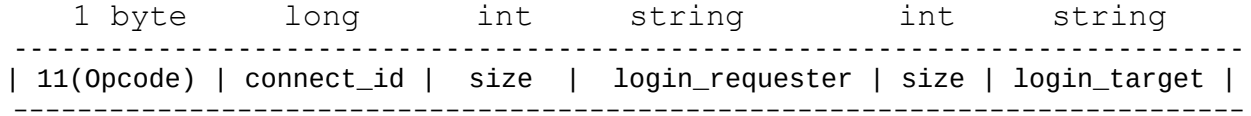
ESTABLISHED(10) = 10 (OPCODE)



A partir de ce moment, la connexion privée est réputée établie: tous les octets écrits par un client sur l'une des connexions sont relayés par le serveur vers l'autre connexion. Lorsqu'un client ferme sa connexion en écriture, le serveur fait de même sur la connexion correspondante.

Afin de demander à fermer une connexion privée entre deux clients, un client envoie la commande CLOSE_CONNECTION.

CLOSE_CONNECTION(11) = 11 (OPCODE) connect_id (LONG) login_requester (STRING) login_target (STRING)



Lorsque le serveur reçoit cette commande, il envoie la commande ACCEPT_CLOSE_CONNECTION au client concerné afin de fermer la connexion de son côté.

ACCEPT_CLOSE_CONNECTION(12) = 12 (OPCODE) login_target (STRING)