ChatFusion

Groupe 2

Preanthy RAVENDRAN
Zineb SDAF

Protocole ChatFusion

Ce RFC décrit le protocole ChatFusion. Le protocole permet à des clients de communiquer avec un serveur ChatFusion. Via ce serveur, les clients peuvent s'échanger des messages textuels
et demander à établir des connexions privées avec les autres clients connectés pour échanger des messages textuels privés ainsi que des fichiers. La fusion de deux serveurs est possible par ce protocole avec des contraintes expliquées dans celui-ci.

Représentation des données

Les entiers (Int) sur 4 octets signés et les longs (Long) sur 8 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.

Paquets de ChatFusion

OPCODE	OPÉRATIONS
0	Création du serveur
1	Demande de fusion entre serveur
2	Acceptation de fusion entre serveur
3	Refus de fusion entre serveur
4	Connexion sans mot de passe
5	Connexion avec mot de passe
6	Acceptation de connexion
7	Refus de connexion
8	Déconnexion
9	Message texte public
10	Vérification d’un pseudonyme
11	Envoi d’un message texte privé par un client
12	Envoi d’un fichier par un client
13	Réception d’un message texte privé
14	Réception d’un fichier

Commande Serveur <-> Serveur

La communication entre les serveurs se fait grâce à différentes commandes.
Le protocole débute par l'assignation d'un nom qui est fixé au démarrage pour chaque serveur et ne peut pas être modifié.

NEW_SERVE(0) = 0 (Byte)

Byte
0

La fusion de deux serveurs sera possible :

- si les deux serveurs sont lancés,
- s'ils n'ont pas le même nom,
- s'ils possèdent chacun au moins un client connecté.

REQUEST_FUSION(1) composé de l'OPCODE 1 suivi d'un Int contenant la taille du nom du serveur et une String correspondant au nom du serveur.

REQUEST_FUSION(1) = 1 (Byte) nameServeSize (Int) nameServe (String)

Byte	Int	String
1	nameServeSize	nameServe

Si la fusion est acceptée, le serveur renvoi la commande FUSION_ACCEPTED(2) d'OPCODE 2 à un nouveau serveur :

FUSION_ACCEPTED(2) = 2 (Byte)

Byte
2

Sinon il renvoi la commande FUSION_REFUSED(3) d'OPCODE 3 :

FUSION_REFUSED(3) = 3 (Byte)

Byte
3

Commande Client <-> Serveur

La communication entre les clients et le serveur, entre les clients se font grâce à différentes commandes. Toutes les commandes débutent par un entier signé sur un octet qui donne le type de la commande.

Authentification

Pour une authentification, le protocole établit une communication entre le client et le serveur. Le client qui se connecte au serveur doit saisir son pseudonyme pour un accès neutre.

Le client peut se connecter de deux façon selon le type d'authentification :

- SIGN_IN : permet au client de créer un pseudonyme non existant dans la base de donnée, pour une connexion neutre sans réutilisation du compte,
- SIGN_IN_PWD : permet, cette fois ci, de se connecter avec un compte authentifié.

SIGN_IN(4) composé de l'OPCODE 4 suivi d'un Int contenant la taille du pseudonyme et d'une String contenant le pseudonyme donné par le client.

SIGN_IN(4) = 4 (Byte) pseudoSize (Int) pseudo (String)

Byte	Int	String
4	pseudoSize	pseudo

SIGN_IN_PWD(5) composé de l'OPCODE 5 suivi d'un Int contenant la taille du pseudonyme du client, d'une String contenant le pseudonyme donné par le client, un Int contenant la taille du mot de passe et une String contenant le mot de passe donné par le client.

SIGN_IN_PWD(5) = 5 (Byte) pseudoSize (Int) pseudo (String) passwordSize (Int) password (String)

Byte	Int	String	Int	String
5	pseudoSize	pseudo	passwordSize	password

Si l'authentification est acceptée, le serveur renvoi la commande AUTH_ACCEPTED(6) d'OPCODE 6.
Si le client utilise la commande SIGN_IN(4), le serveur renvoi la commande AUTH_ACCEPTED(6) d'OPCODE 6 si et seulement si le pseudonyme est unique.

AUTH_ACCEPTED(6) = 6 (Byte)

Byte
6

Dans le cas contraire, il renvoi la commande AUTH_REFUSED(7) d'OPCODE 7.

AUTH_REFUSED(7) = 7 (Byte)

Byte
7

Une fois le client connecté, il sera authentifié de manière unique et n'aura plus besoin de saisir son pseudonyme et/ou son mot de passe.

Pour se déconnecter, le client utilise la commande SIGN_OUT(8) d'OPCODE 8 :

SIGN_OUT(8) = 8 (Byte)

Byte
8

Organisation du chat

Pour envoyer un message à tous les clients connectés aux serveurs, un client envoie une commande MSG_PUBLIC(9) d'OPCODE 9.

La commande MSG_PUBLIC(9) composé de l'OPCODE 9 suivi d'un Int contenant la taille du pseudonyme du client, d'une String contenant le pseudonyme du client, d'un Int contenant la taille du nom du serveur dans lequel le client veut envoyer son message, d'une String correspondant au nom du serveur, d'un Int contenant la taille du message et d'une String contenant le message que le client envoi :

MSG_PUBLIC(9) = 9 (Byte) pseudoSize (Int) pseudo (String) nameServeSize (Int) nameServe (String) msgSize (Int) msg (String)

Byte	Int	String	Int	String	Int	String
9	pseudoSize	pseudo	nameServeSize	nameServe	msgSize	msg

Si un client veut établir un chat privé avec un autre client, on vérifie si le pseudonyme donné par le client existe et est bien dans le serveur indiqué.
Il envoie au serveur une commande CHECK_PRIVATE(10) composé de l'OPCODE 10 suivi d'un Int contenant la taille du client destinataire, d'une String contenant le client destinataire correspondant au pseudonyme du client recherché, d'un Int contenant la taille du nom du serveur dans lequel le client destinataire se trouve et d'une String contenant le nom du serveur :

CHECK_PRIVATE(10) = 10 (Byte) receiverSize (Int) receiver (String) nameServeSize (Int) nameServe (String)

Byte	Int	String	Int	String
10	receiverSize	receiver	nameServeSize	nameServe

Si la commande CHECK_PRIVATE(10) est renvoyé par le serveur à l'identique alors le client peut dialoguer avec ce destinataire.
Sinon la commande CHECK_PRIVATE(10) est ignorée.

Après que le pseudonyme du destinataire est accepté, si le client veut envoyer un message privé à un unique client destinataire, le client envoie une commande PRIVATE_MSG(11) composé de l'OPCODE 11 suivi d'un Int contenant la taille du pseudonyme du client expéditeur, d'une String contenant le pseudonyme du client expéditeur, d'un Int contenant la taille du pseudonyme du client destinataire, d'une String contenant le pseudonyme du client destinataire, d'un Int contenant la taille du nom du serveur du client destinataire, d'une String contenant le nom du serveur, d'un Int contenant la taille du message et d'une String contenant le message que le client envoi :

PRIVATE_MSG(11) = 11 (Byte) senderSize (Int) sender (String) receiverSize (Int) receiver (String) nameServeSize (Int) nameServe (String) msgSize (Int) msg (String)

Byte	Int	String	Int	String	Int	String	Int	String
11	senderSize	sender	receiverSize	receiver	nameServeSize	nameServe	msgSize	msg

Si un client veut envoyer un fichier, il envoie une commande PRIVATE_FILE(12) composé de l'OPCODE 12 suivi d'un Int contenant la taille du pseudonyme du client expéditeur, d'une String contenant le pseudonyme du client expéditeur, d'un Int contenant la taille du pseudonyme du client destinataire, d'une String contenant le pseudonyme du client destinataire, d'un Int contenant la taille du path du fichier, d'une String contenant le path du fichier, d'un Int contenant la taille des données du fichier, de Bytes contenant les données du fichier, d'un Int contenant la taille du nom du serveur dans lequel le client destinataire se trouve et d'une String contenant le nom du serveur :

PRIVATE_FILE(12) = 12 (Byte) senderSize (Int) sender (String) receiverSize (Int) receiver (String) pathSize (Int) path (String) dataSize (Int) data (Bytes) nameServeSize (Int) nameServe (String)

Byte	Int	String	Int	String	Int	String	Int	Bytes	Int	String
12	senderSize	sender	receiverSize	receiver	pathSize	path	dataSize	data	nameServeSize	nameServe

Transmission par le serveur

Après la réception d'un message (texte ou fichier), le serveur doit communiquer une commande à tous les clients connectés ou aux clients d'un chat privé.

Lors de la réception d'un message, le serveur envoie une commande MSG_OBTAIN(13) composé de l'OPCODE 13 suivi d'un Int contenant la taille du pseudonyme du client expéditeur, d'une String contenant le pseudonyme du client expéditeur, d'un Int contenant la taille du nom du serveur de l'expéditeur, d'une String contenant le nom du serveur de l'expéditeur, d'un Int contenant la taille du message et d'une String contenant le message du client :

MSG_OBTAIN(13) = 13 (Byte) senderSize (Int) sender (String) nameServeSize (Int) nameServe (String) msgSize (Int) msg (String)

Byte	Int	String	Int	String	Int	String
13	senderSize	sender	nameServeSize	nameServe	msgSize	msg

Lors de la réception d'un fichier, le serveur envoie une commande FILE_OBTAIN(14) composé de l'OPCODE 14 suivi d'un Int contenant la taille du pseudonyme du client expéditeur, d'une String contenant le pseudonyme du client expéditeur, d'un Int contenant la taille du nom du fichier, d'une String contenant le nom du fichier, d'un Int contenant la taille des données du fichier, de Bytes contenant les données du fichier, d'un Int contenant la taille du nom du serveur et d'une String contenant le nom du serveur du client qui est l'auteur de l'envoi du fichier :

FILE_OBTAIN(14) = 14 (Byte) senderSize (Int) sender (String) filenameSize (Int) filename (String) dataSize (Int) data (Bytes) nameServeSize (Int) nameServe (String)

Byte	Int	String	Int	String	Int	Bytes	Int	String
14	senderSize	sender	filenameSize	filename	dataSize	data	nameServeSize	nameServe