Florian DURAND
Benjamin JEDROCHA

ChatOS
RFC COSP

Le COSP protocole

Résumé

COSP (ChatOS Protocol) est un protocole mettant en place un service de discussions permettant la mise en relation TCP directe des utilisateurs sans divulguer leur adresse IP.

But

COSP est un protocole TCP permettant à des clients de discuter entre eux sans divulguer leur adresse IP. Chaque client connecté au serveur sera identifié par un pseudonyme et un identifiant distincts.
Trois types de requêtes seront possibles, la première étant l'envoi d'un message à tous les clients connectés. La seconde étant l'envoi d'un message à un client spécifique identifié par son pseudonyme. Et la troisième étant l'établissement d'une connexion TCP privée avec un client spécifique. L'établissement d'une connexion TCP privée obligera les deux clients à créer chacun un nouveau socket pour se connecter au serveur.

Présentation du protocole

Tout d'abord le client établit la connexion avec le serveur, une fois connecté, il envoie une demande pour enregistrer son pseudonyme. Son pseudonyme ne doit pas être déjà existant dans la liste du serveur. Une fois connecté, le client peut envoyer tout type de requête. Il peut également recevoir des messages, des mises à jour des données et des demandes de connexion TCP privée à tout moment. Lorsqu'un client reçoit une demande de connexion TCP il peut décider de l'accepter ou de la refuser en signalant son choix au client qui a effectué la demande de connexion.

La plupart des erreurs viennent de l'absence d'un client lors d'une déconnexion ou encore du refus de l'établissement d'une connexion TCP privée. Mais seule la fermeture de la connexion du serveur pourra mettre fin au service.

Spécifications

Les messages et les pseudonymes sont encodés en UTF-8. Lorsque la taille d'un message, ou d'un pseudonyme, est évoquée, il s'agit de la taille après encodage en octet.
Chaque pseudonyme est rattaché à un identifiant unique représenté par un SHORT.

Request code*	Description
0	Envoyer un message à tout le monde
1	Envoyer un message à un client spécifique
2	Demande de connexion TCP privée à un client spécifique
3	Accepte la connexion TCP privée
-1	ERREUR : Refuse la connexion TCP privée

*Request code = Code envoyé par un client au serveur.

Response code*	Description
0	Connexion d'un utilisateur au serveur : Mise à jour de la liste des participants envoyée à tous les clients connectés
1	Déconnexion d'un utilisateur : Mise à jour de la liste des participants envoyée à tous les clients connectés
2	Message envoyé à tous les clients connectés
3	Message envoyé à un client précis
4	Demande de connexion TCP privée
5	Demande de connexion acceptée à l'émetteur
6	Demande de connexion acceptée au receveur
-1	ERREUR : Client absent
-2	ERREUR : Demande de connexion refusée

*Response code = Code envoyé par le serveur à un client.

Connexion code*	Description
0	Pseudonyme accepté, renvoie de la liste des pseudonymes et identifiants
-1	ERREUR : Pseudonyme déjà utilisé

*Connexion code = Code spécifique à la connexion et à l'envoie du pseudonyme du client. Valable tant que l'utilisateur n'est pas enregistré sur le serveur.

Établissement de la connexion

Lors de la première connexion, le client devra envoyer son pseudonyme. Le serveur lui répond pour indiquer si ce pseudonyme est disponible :

Client -> Serveur

short	n bytes
Taille pseudonyme (n)	Pseudo

Cas nominal

Dans le cas nominal, le serveur enregistre l'utilisateur et renvoie la liste des identifiants et pseudonymes des autres clients connectés :

Serveur -> Client

byte	short	short	short	n bytes	...
Connexion code = 0	Nombre de clients connectés (m)	Identifiant du premier client	Taille du pseudonyme du premier client (n)	Pseudo	Répétition des colonnes 3, 4 et 5 pour tous les m -1 autres clients connectés

Erreur pseudonyme déjà utilisé

Si le pseudonyme est déjà utilisé par un autre client, le serveur renvoie une erreur au client. Dans ce cas, le client devra réitérer sa demande s'il veut se connecter :

Serveur -> Client

byte
Connexion code = -1

Mise à jour de la liste des pseudonymes

Chaque client devra conserver la liste des pseudonymes à jour. Pour cela le serveur envoie à chaque nouvelle connexion ou déconnexion un paquet TCP pour signaler aux clients de mettre à jour leur liste.

Dans le cas d'un nouveau client sur le serveur

Serveur -> Clients

byte	short	short	n bytes
Response code = 0	Identifiant du nouveau client	Taille pseudonyme (n)	Pseudo

Déconnexion d'un client au serveur

Serveur -> Clients

byte	short
Response code = 1	Identifiant du client déconnecté

Client envoie un message à tout le monde

Un client décide d'envoyer un nouveau message à tous les clients connectés :

Client -> Serveur

byte	short	n bytes
Request code = 0	Taille message (n)	Message

Le serveur fait suivre le message à tous les autres clients en précisant l'identifiant de l'expéditeur :

Serveur -> Autres Clients

byte	short	short	m bytes
Response code = 2	Identifiant de l'expéditeur	Taille message (m)	Message

Client envoie un message à un client spécifique

Un client A décide d'envoyer un message privé à un client B. Le client A devra indiquer l'identifiant du client B et le contenu de ce message :

Client A -> Serveur

byte	short	short	m bytes
Request code = 1	Identifiant du client B	Taille message (m)	Message

Réponse du serveur dans le cas nominal

Le serveur envoie au client B le message avec l'identifiant du client A :

Serveur -> Client destinataire

byte	short	short	m bytes
Response code = 3	Identifiant du client A	Taille message (m)	Message

Cas d'erreur

Si le client B s'est déconnecté pendant que le paquet était encore sur le réseau, le serveur renverra alors au client A que ce dernier n'est plus connecté :

Serveur -> Client A

byte
Response code = -1

Un client établit une connexion TCP privée avec un client spécifique

Un client A décide d'établir une connexion TCP privée avec un autre client B. Le client A envoie sa demande en spécifiant le port sur lequel il ouvrira une socket de connexion au serveur :

Client A -> Serveur

byte	int	short
Request code = 2	port	Identifiant Client B

Client B n'est pas connecté

Le client B peut se déconnecter pendant que les données transitent encore sur le réseau, alors le serveur envoie une erreur au client A :

Serveur -> Client A

byte
Response code = -1

Sinon le serveur notifie le client B qu'une demande de connexion TCP lui a été envoyé :

Serveur -> Client B

byte	short
Response code = 4	Identifiant Client A

Client B accepte la connexion

Si le client B accepte la demande de connexion, il envoie sa réponse avec le port sur lequel il acceptera les paquets TCP :

Client B -> Serveur

byte	int	short
Request code = 3	port	Client A

Ainsi le serveur notifie le client A que la connexion a été établi :

Serveur -> Client A

byte	short
Response code = 5	Client B

Ainsi le serveur notifie le client B qu'il a bien reçu la validation de connexion :

Serveur -> Client B

byte	short
Response code = 6	Client A

Client B refuse la connexion

Si le client B refuse la demande de connexion, il envoie une erreur au serveur :

Client B -> Serveur

byte	short
Request code = -1	Id A

Ainsi le serveur notifie le client A que la connexion a été refusé :

Serveur -> Client A

byte	short
Response code = -2	Id B