Auteurs: Angelica Amine, Mohamed Yassine Yahyaoui
Date : Février 2021
UGE

ChatOS

Abstrait

ChatOs est un service de discussion entre plusieurs Clients utilisant le même serveur appliquant un protocol pré-défini sans partager leur adresse IP.

Statut de ce mémo

Cette requête résume le protocol pour être implémenté dans différents paquets utilisés pour la communication entre les utilisateurs et les autres clients.
Ce document explique de plus, les décisions derrière la structure des paquets.

Table des Matières

1. Introduction

Le but du protocol ChatOs est de permettre une communication multiple et privée entre des utilisateurs sans le besoin d'exposer leur adresse IP, le protocol appliqué et décrit ci-dessous est une version simplifiée d'un serveur de chat

Tous les paquets échangés dans ce protocol utilise le TCP (Transmission/Transfer Control Protocol). Ainsi, la disponibilité des paquets est géré par le système d'exploitation

ChatOs permet trois principales opérations:
- envoyer un message à tous les utilisateurs
- envoyer un message à un utilisateur spécifique
- établir une connexion privée avec un autre utilisateur

Pour être capable d'utiliser n'importe quelle opération, une connexion p**ar l'utilisateur** avec un serveur qui utilise le même protocol DOIT être établi au préalable.

1.1. Notation des exigences

Ce document utilise occasionnellement des termes qui apparaissent en lettres capitales

Quand les termes "DOIT", "DEVRAIT", "RECOMMANDE", "NE DOIT PAS", "NE DEVRAIT PAS", "POURRAIT" apparaissent, ils sont utilisés pour indiquer des exigences particulières dans cette spécification.

Une discussion de la significations de ces termes apparaissent dans [RFC2119].

1.2. Terminologie

Nous décrivons dans cette section la terminologie utilisée dans le protocol.

UTF8
: L'UTF8 est envoyé comme le jeu de caractères (charset) par défault utilisé dans ChatOS. Toutes les requêtes et paquets DOIVENT utiliser ce jeu de caractères sauf si indication contraire dans la RFC.

Requête
: Un paquet contenant des données est envoyé du le client au le serveur. Le paquet commence avec un code, la taille de l'ID du client suivi de l'ID, et enfin, la taille de la zone de données suivie de la zone de données.

Ack
: Une réponse du serveur au client après avoir reçu une requête. Elle correspond à un code représenté par un entier.

Code paquet
: Un code au début du paquet, représenté par un entier, indique le but ou l'état du paquet.

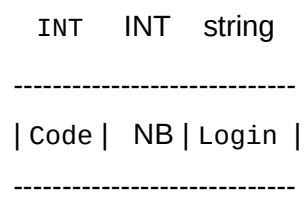
Code
: Un code est un entier signé et codé sur un octet. Il est transmis en Big Endian.

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

2. Connexion au serveur et premières requêtes

2.1. Connexion au serveur

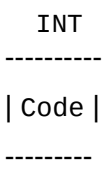
La première étape consiste à la connexion du client au serveur. Le client envoie un paquet de connexion au serveur contenant son login.
Pour cela, le client envoie un paquet connecté qui suit le format suivant :



Le code représente le type du paquet que le serveur reçoit. Dans ce cas précis, le code est **“0”** .

The Login est une chaîne de caractère qui est le pseudonyme du client limitée à 30 caractères (taille fixe). Elle est précédée par le champ NB, un entier correspondant à la taille du login.

Le serveur après avoir reçu le paquet, va vérifier si celui-ci existe déjà ou non.
En réponse à cette requête, le serveur renvoie un paquet ACK afin que le client sache si la connexion a pu être établie ou non.

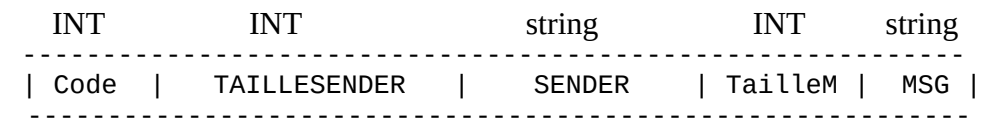


Le champ code est un entier qui vérifie si le paquet reçu du client est valide ou non.

Si la connexion a été établie avec succès, le code a alors la valeur **“10” qui signifie que l’identifiant donné n’existe pas déjà.**
Si le login est déjà utilisé par un autre client, dans ce cas la connexion échoue et le code est **“11”.**

2.2. Message publique à tous les utilisateurs

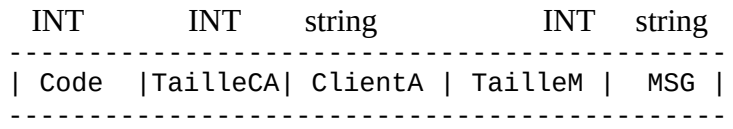
Une fois que le client est connecté au serveur, il peut alors envoyé un message à tous les utilisateurs ou à un utilisateur spécifique du serveur.
Il faut contrôler le fait que l’utilisateur ne peut pas envoyer un message avec un autre login.
Dans cette partie, nous allons décrire la procédure d’envoie d’un message publique à tous les utilisateurs.
Pour qu’un client A envoie un message à tous les autres clients, il doit envoyer un paquet au serveur avec le format suivant:



Le code est un entier de valeur 1 qui signifie que le client A envoie un message à tous les utilisateurs. Le champ suivant est la taille du login du client A.

Le SENDER (login client A) DOIT être de la même taille que TailleS, le champ suivant correspond à la taille tailleM du message MSG à envoyer. Le message MSG ne peut pas occuper plus de 1024 octets.

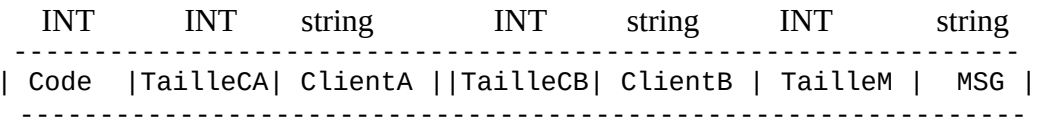
Une fois que le paquet est vérifié par le serveur, celui-ci envoie le message à tous les utilisateurs dans un nouveau paquet avec la même structure que le paquet reçu de A au serveur:



Le code envoyé du serveur à tous les autres utilisateurs est la valeur 3.

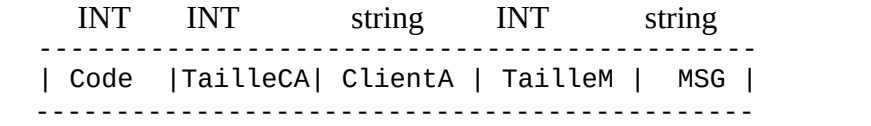
2.3. Message à un utilisateur spécifique

La procédure est la même que l'envoie du message à tous les utilisateurs. La seule différence est de donner également le login du client destinataire. Par exemple si nous envoyons un message à un client B, nous aurons le format suivant:



Dans ce cas, le code aura la valeur 2 pour dire que le client A envoie un message au client B uniquement. Si le client n'existe pas ou que le format du paquet n'est pas respecté, la requête est ignorée.

Dans le cas où le client B existe bien, un paquet du serveur vers celui-ci est envoyé. Le format est le même que le paquet du clietn A vers le serveur avec le code 4:



Lorsque le client B reçoit la requête avec le code 7, il sait qu'il reçoit un message directe du client A.

3. La connexion privée

3.1. Description

La connexion privée consiste à connecter deux clients sans qu'ils partagent leur adresse IP. Les deux clients A et B se connectent avec une nouvelle socket au serveur: le client A avec une socket SA et le client B avec une socket SB. Après une étape d'authentification, le serveur transfère tous les octets lus de SA vers SB et vice-versa. Après cette étape d'authentification, les deux clients utilisent cette connexion "sans contrôle". Tout doit passer comme si c'était une connexion TCP normal entre A et B.

Le protocol doit permettre aux clients d'effectuer des requêtes de connexions privées et accepter/refuser les requêtes de connexions privées entrantes.

3.2. Procédure avec des paquets

Nous avons deux clients: le client A et le client B. Ils sont déjà connectés au serveur et A souhaite établir une connexion privée avec le client B.
Le client A commence en se connectant au serveur avec ce paquet:

INT	INT	string	INT	string
Code	n	ClientA	m	ClientB

Si le client B n'existe pas, la requête est ignorée.

Normalement, une nouvelle connexion est établie par le paquet décrit dans la partie 2.1. mais comme les informations du client A sont déjà dans le serveur, nous pouvons utiliser ce paquet pour établir la connexion avec une nouvelle socket et informer le serveur d'une demande de connexion privée au client B.

Une fois que le serveur reçoit ce paquet, cela DOIT sauvegarder the nouvelle socket pour le client A et envoyer la requête au client B avec le format suivant:

INT	INT	string	INT	string
Code	n	ClientA	m	ClientB

Dans la conception des deux paquets ci-dessus, le code aura la valeur 4 qui correspond à une demande d'établissement d'une connexion privée.

Le client B a deux possibilités:

1- Accepter la demande :

Le client B informe le serveur qu'il accepte la demande de connexion privée avec le client lui envoyant le code 12 :

INT	INT	string	INT	string
Code	m	REQUESTER	n	TARGET

Une fois que le serveur reçoit ce code de la part du client B, celui-ci va générer un identifiant de connexion de type long propre à cette connexion privée et l'envoyer aux deux clients A et B avec le code 5 signifiant que l'identifiant de connexion a été créée et qu'une connexion entre les deux clients A et B est prête à être établie:

INT	INT	string	INT	string	LONG
Code	m	REQUESTER	n	TARGET	connect_id

2- Refuser la demande :

Le client B décline la demande de connexion privée en envoyant au serveur le paquet suivant avec le code 13 qui signifie que le client B a refusé la connexion :

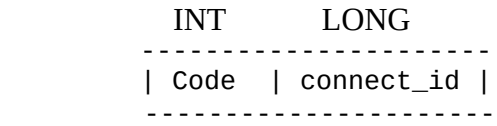
INT	INT	string	INT	string
Code	m	REQUESTER	n	TARGET

Pour éviter toutes connexions pendantes, le client B envoie la réponse au serveur sans lancer une nouvelle connexion. Quand le serveur reçoit la réponse, il alors le client A que sa demande de connexion privée avec le client B a été refusé. Après cela, le serveur ferme proprement les deux connexions.

Quelque soit la demande, afin d'éviter toutes demandes multiples (spam) d'un client A à un client B, un système d'historique est mise en place pour un client afin de vérifier si une demande est déjà en cours. Ainsi, un client A ne peut pas demander une connexion privée plusieurs fois à un client B tant que la première demande est en cours.

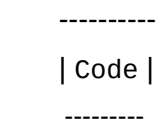
4. Etablissement de la connexion privée

Après que le client B ait accepté la demande de connexion privée du client A et que le serveur a envoyé l'identifiant unique aux deux clients, les deux peuvent établir une connexion privée. Les deux clients se connectent chacun avec une nouvelle connexion TCP sur laquelle ils envoient une requête de la forme suivante:



Le code est un entier ayant la valeur 8 qui correspond à l'établissement de la connexion privée. Le champ connect_id correspond quant à lui à l'identifiant unique du serveur attribué aux client A et client B. Lors de l'envoie de ce paquet,le connect_id doit non seulement être celui donné par le serveur mais aussi être identique pour A et B lorsqu'il envoie chacun leur requête.

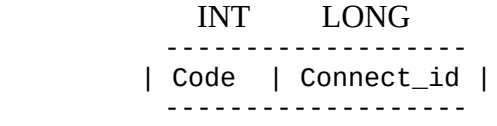
Une fois que la connexion est établie, le serveur envoie à A et à B le code 6 qui signifie que la connexion privée est établie.



En conséquence, le client A peut envoyer des données sur sa connexion sans suivre un protocole spécifique. Lorsqu'un client ferme sa connexion en écriture, le serveur fait de même sur la connexion correspondante.

5. Déconnecter une connexion privée

Si un client A souhaite couper sa connexion privée avec un client B, il doit envoyer une requête au serveur sur la connexion publique de la forme suivante avec le code 9 :



Après avoir reçu ce paquet, le serveur se chargera de fermer la connexion du côté du client A ainsi que du côté du client B.
Remarque: Si un client ferme totalement sa connexion avec le serveur, celui-ci se chargera de fermer toutes les connexions privées avec ce client. Egalement, si un client souhaite fermer une connexion privée inexistante avec un autre client, le paquet ci-dessus envoyé au serveur sera ignoré.

Codes

Valeur	Signification

0	Connexion au serveur (partie 2.1)
1	Envoie d’un message d’un client à tous les autres utilisateurs
2	Envoie d’un message d’un client à un autre client
3	Réception d’un message publique d’un client
4	Demande d’établissement d’une connexion privée
5	Génération de l’identifiant de connexion
6	Connexion privée établie
7	Réception d’un message privé d’un client
8	Etablissement de la connexion privée
9	Déconnexion d’une connexion privée entre deux clients

ACK Codes

Valeur	Signification
10	Connexion au serveur établie
11	Connexion au serveur non établie
12	Connexion privée acceptée
13	Connexion privée refusée

Adresses des auteurs

Angelica Amine

email: aamine@etud.u-pem.fr

Mohamed Yassine Yahyaoui

email: myahyao01@etud.u-pem.fr