RFC : PROTOCOLE CH « ChatHack »

Projet M1 Programmation Réseau

LOPES MENDES Ailton &

LAMBERT—DELAVAQUERIE Fabien

Table des matières

CHANGEMENTS

Changement 1 : Nous avons changé les Code Opération suite aux valeurs réponses fournis pas le serveur Mdp. Toutes les trames ont donc leur code modifié.

3. Trames CH

Il faut tout s'abord signaler que tous les paquets doivent être encodés en UTF-8.

CH reconnaît 8 types de paquets selon des opérations liées :

Code opération	Opération
3	Connexion
4	Message Global
5	Connexion privée
6	Message privé
7	Transfert Fichier privé
8	Déconnexion
9	ACK
10	Erreur

Changement 2 : On ajoute le pseudonyme de l'expéditeur pour les trames de messages privés et transfert de fichier.

7. Message privé

Une fois connectés, les utilisateurs C1 et C2 peuvent s'envoyer des messages via leur connexion privée.

L'utilisateur qui veut envoyer un message à l'autre doit lui envoyer un paquet de ce type :

OP_Code	INT	M octets	INT	N octets
9	M = Taille Pseudo Expéditeur	Pseudo expéditeur	N = Taille Message	Message

En cas d'erreur, le destinataire doit alors répondre une erreur avec un paquet de ce type :

OP_Code	Request_Code
ERROR = 10	6

8. Transfert de fichier

Les utilisateurs connectés en privés peuvent aussi s'échanger des fichiers. Le client qui veut envoyer le fichier doit envoyer un paquet de ce type :

OP_Code	INT	O	INT	N	INT	M
7	O = Taille Pseudo Expéditeur	Pseudo Expéditeur	N =Taille File Name	File Name	M = Taille File	File

Le destinataire doit lui répondre par un acquittement ou une erreur d'envoi :

OP_Code	Request_Code	INT	N
9	7	N = Taille Message	Fichier bien reçu
10	7		

PROTOCOLE CH « ChatHack »

Statut de ce document

Ce document spécifie un protocole standard du projet ChatHack de Programmation Réseau M1 2020.

Résumé

CH est un protocole simple de chat écrit, de messages privés et de transfert de fichiers. Son appellation vient du projet en lui-même ChatHack. Ce document décrit le protocole et ses différents types de paquets et explique aussi les raisons ayant conduit à certaines décisions.

1.Objectif

Ce protocole fonctionne comme un chat où les utilisateurs se connectent à un serveur, ont un identifiant peuvent envoyer des messages globaux à tous les utilisateurs connectés au server ainsi que créer des connexions privées avec d'autres utilisateurs pour s'envoyer des messages privés ou se transmettre des fichiers.

2. Vue d'ensemble

N'importe qui peut utiliser ce protocole en se connectant avec un pseudonyme qui n'est pas déjà utilisé par un utilisateur déjà connecté ou faisant partie de la base de données, l'utilisateur est connecté en tant qu' « invité » mais a accès aux mêmes fonctionnalités que les autres utilisateurs.

Il existe en effet une base de données pour les utilisateurs ayant enregistré un compte, leur permettant de conserver toujours le même pseudonyme, on considérera ces utilisateurs comme « premium ».

Un message d'erreur est envoyé en cas d'échec lors d'une trame. Un acquittement est seulement envoyé lors de la connexion au serveur ou d'un transfert de fichier réussi.

Le protocole CH est implémenté au-dessus du protocole TCP.

3. Trames CH

Il faut tout s'abord signaler que tous les paquets doivent être encodés en UTF-8.

CH reconnaît 8 types de paquets selon des opérations liées :

Code opération	Opération
3	Connexion
4	Message Global
5	Connexion privée
6	Message privé
7	Transfert Fichier privé
8	Déconnexion
9	ACK
10	Erreur

Les réponses du serveur débutent par le code ACK = 6 ou Erreur=7 suivi du Code opération de la requête liée à la réponse.

4. Protocole de connexion

Il existe 2 façons de se connecter au serveur. Le serveur possède une base de données d'identifiants liés à un mot de passe, on peut donc se connecter avec ou sans mot de passe.

La connexion sans mot de passe ne fonctionne pas avec un identifiant déjà enregistré dans la base de données.

La connexion avec un identifiant déjà connecté est impossible.

Présentation des paquets de connexion :

Nb octets	1 octet	1 octet	4 octets	chaîne	4 octets	chaîne
Information	OP_code	Type Connexion	Id Size	Id	PassWord Size	PassWord
Avec Mot de Passe	3	0				
Sans Mot de Passe	3	1				

Voici les paquets de réponse envoyés par le serveur sur cette opération :

➔ Connexion réussie :

OP_Code	Request_Code	Taille Message	Message
9	3	X	Connexion établie

➔ Echec :

OP_Code	Request_Code
10	3

5. Message Global

I. Envoi Message Global :

Un utilisateur doit être connecté pour pouvoir envoyer des messages globaux aux utilisateurs connectés.

Il doit envoyer un paquet propre à cet envoi :

OP_Code (1 byte)	Etape (1 byte)	Taille Message Réponse (INT)	N
4	1	N = Taille Message	Message

En cas d'échec, le serveur répond alors à cette requête :

OP_Code	Request_Code
10	4

II. Réception Message Global :

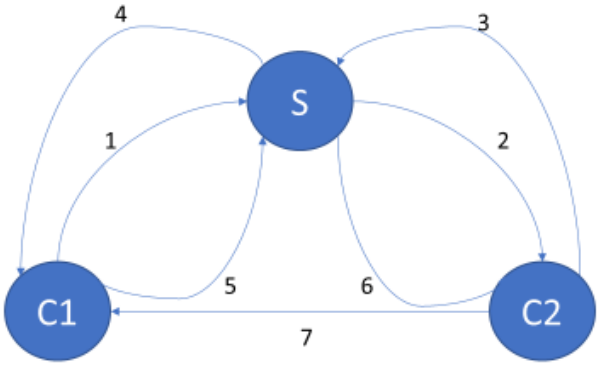
Une fois l'acquittement envoyé à l'expéditeur, le serveur envoie à tous les autres utilisateurs connectés avec un paquet de ce type :

OP_Code (1 byte)	Etape (1 byte)	Taille Pseudo Expéditeur (Int)	Pseudo Expéditeur (N octets)	Taille Message (Int)	Message (M octets)
4	2	N		M	

6. Connexion privée

Il est possible avec ce protocole d'échanger des messages privés et des fichiers via une connexion avec un autre utilisateur. Il faut pour cela créer une connexion privée entre ces deux utilisateurs.

Voici un schéma représentant les différentes étapes de cette connexion :



C1 représente le client qui demande la connexion privée.
C2 représente le client auquel C1 souhaite se connecter.
S représente le serveur de chat.

Dans cette représentation de connexion privée, C1 devient une sorte de serveur où C2 se connecte.

Différentes étapes :

1. C1 envoie au serveur le pseudo de l'utilisateur auquel il veut se connecter avec le paquet suivant :

OP_Code	Etape (1 byte)	INT	N
5	1	N =Taille Pseudo C2	Pseudo C2

2. Le serveur envoie ensuite le pseudo de C1 à C2 pour que celui-ci accepte ou non la connexion :

OP_Code	Etape	INT	N
5	2	N =Taille Pseudo C1	Pseudo C1

3. C2 répond au serveur en indiquant le pseudo de la personne à qui il répond à la demande de connexion avec byte de validation ou de refus

OP_Code	Etape	INT	N	Etat
5	3	N = Taille Pseudo C1	Pseudo C1	OK = 0
5	3	N = Taille Pseudo C1	Pseudo C1	NON = 1

4. Le serveur transmet la réponse de C2 à C1.

OP_Code	Etape	INT	N	Etat
5	4	N = Taille Pseudo C2	Pseudo C2	OK = 0
5	4	N = Taille Pseudo C2	Pseudo C2	NON = 1

5. Si C2 a refusé, l'opération s'arrête ici, sinon C1 envoie son adresse ainsi que son port sur lequel C2 pourra se connecter au serveur, ainsi qu'un token (long) qui servira de clef pour s'assurer lors de la connexion que le client qui se connecte est bien C2.

OP_Code	Etape	INT	N	INT	INT	N	Long
5	5	N = Taille Pseudo C2	Pseudo C2	Port	N = taille Adresse	Adresse C1	Token clef

6. Le serveur transmet ces informations au client C2.

OP_Code	Etape	INT	N	INT	INT	N	Long
5	6	N = Taille Pseudo C1	Pseudo C1	Port	N = taille Adresse	Adresse C1	Token clef

7. A cette dernière étape C2 se connecte directement à C1 en indiquant le token clef pour que C1 puisse vérifier qu'il s'agisse bien de la même adresse transmise par le serveur précédemment.

OP_Code	Etape	INT	N	Long
5	7	N = Taille pseudo C2	Pseudo C2	Token clef

En cas d'erreur durant le processus des étapes 1 à 6, le serveur envoie au client ayant envoyé le paquet erroné un paquet suivant avec l'étape X de l'erreur :

OP_Code	Request_Code	Etape	INT	N
ERROR = 10	5	X	N= Taille Message	Erreur processus connexion privée, recommencez l'étape

Le client pourra donc reprendre à l'étape où l'erreur a été détectée.

7. Message privé

Une fois connectés, les utilisateurs C1 et C2 peuvent s'envoyer des messages via leur connexion privée.

L'utilisateur qui veut envoyer un message à l'autre doit lui envoyer un paquet de ce type :

OP_Code	INT	N octets
6	N = Taille Message	Message

En cas d'erreur, le destinataire doit alors répondre une erreur avec un paquet de ce type :

OP_Code	Request_Code
ERROR = 10	6

8. Transfert de fichier

Les utilisateurs connectés en privés peuvent aussi s'échanger des fichiers. Le client qui veut envoyer le fichier doit envoyer un paquet de ce type :

OP_Code	INT	N	INT	M
7	N =Taille File Name	File Name	M = Taille File	File

Le destinataire doit lui répondre par un acquittement ou une erreur d'envoi :

OP_Code	Request_Code	INT	N
9	7	N = Taille Message	Fichier bien reçu
10	7		

9. Déconnexion

Pour se déconnecter du serveur ou mettre fin à une connexion privée, l'utilisateur envoie un même type de paquet soit au serveur soit à l'utilisateur auquel il est connecté.

On considère qu'un utilisateur qui se déconnecte du serveur ne se déconnecte pas de ses connexions privées. Il peut donc toujours partager des fichiers et des messages privés aux utilisateurs auxquels il est connecté.

OP_Code
Déconnexion = 8

En réponse le serveur ou l'autre utilisateur envoie un acquittement ou une erreur avant de le déconnecter.

OP_Code	Request_Code	INT	N
9	8	N = Taille Message	Déconnexion effectué
10	8	N = Taille Message	Erreur : Déconnexion non effectué

10. Améliorations possibles :

Voici une liste d'idées que nous aimerions ajouter dans le futur à notre protocole :

Actuellement, la liste des utilisateurs de la base de données est fixée. Il serait donc intéressant d'ajouter une opération pour se créer un compte « premium » et s'enregistrer dans la base de données.

Dans le cas où un utilisateur veut envoyer un fichier ou un message à plusieurs autres utilisateurs, il serait aussi envisageable de proposer des groupes privés plutôt que de simples connexions entre 2 utilisateurs.