Programmation Réseau B.MARIÉ
Université Gustave EIFFEL C. RUFO
Groupe 1

Le protocole ChatFusion

Statut du mémo

Cette RFC spécifie un protocole pour un service de discussions permettant de mettre en place un service de discussions et d'échanges de fichiers. Mais aussi de la fusion entre plusieurs serveurs. La diffusion de ce mémo est libre.

Sommaire

ChatFusion est un protocole simple utilisé pour le chat entre utilisateurs. Ce document décrit le protocole et les différents types de paquets utilisés. Ce document explique aussi les raisons de certaines décisions de conception.

Remerciement

Ce protocole a été réalisé dans le cadre du cours Programmation Réseau de l'Université Gustave Eiffel. Ce document ainsi que l'entièreté de ce projet doit être rendu à Pablo ROTONDO, chargé de travaux dirigé du groupe 1 dans la promotion Master 1 Informatique 2021/2022. Ce protocole doit s'inspirer des mécanismes mis en place tout au long du semestre et vu lors du cours et respecter les principes et les bonnes pratiques de la programmation réseaux.

1. But

ChatFusion est un protocole qui permet de mettre en place la fusion de plusieurs serveurs entre eux. Ce protocole sera implémenté en utilisant le Transmission Control Protocol ("protocole de contrôle de transmission" abrégé TCP).

Ce protocole permet d'envoyer des messages et des fichiers en privé, en public sur le serveur, mais aussi de faire relayer des messages d'un client vers un autre serveur qui a été fusionné avec le serveur du client initial.

Il y a dans ce protocole, deux types d'authentifications, nous reviendrons dessus plus tard dans le document :

- Sans mot de passe
- Avec mot de passe

Une fois connecté et authentifié, le client dispose de plusieurs fonctionnalités.

Deux possibilités s'offre à lui :

- Envoyer des messages publiques qui seront transmis à tous les clients connectés sur le serveur et sur les autres avec lesquels le serveur est fusionné.
- Envoyer des messages ou des fichiers qui seront destinés à un seul client identifié par son pseudonyme qui peut tout à fait être sur un serveur différent à condition qu'il soit sur un serveur qui soit fusionné avec le sien.

De plus, le serveur, quant à lui, doit être en mesure de pouvoir se "fusionner" avec un autre serveur mais rien de lui empêche de pouvoir fusionner avec plusieurs serveurs au cours de son utilisation. La fusion sera acceptée si les deux serveurs possèdent un nom différent et si l'un des deux serveurs n'est pas déjà fusionné avec un serveur de même nom.

2. Aperçu du protocole

Chaque connexion commence par une phase d'authentification. Le client aura deux façons de s'identifier, une avec un mot de passe et une autre sans.

Si il s'identifie avec mot de passe alors il doit fournir un pseudonyme et un mot de passe. Si les informations sont reconnues dans la base de données alors la connexion sera autorisée. Si les informations ne sont pas bonnes alors il recevra un message.

Si il s'identifie sans mot de passe alors il devra fournir seulement un pseudonyme. Si le pseudonyme n'est pas déjà pris ou qu'il n'est pas dans la base de données alors la connexion sera autorisée sinon il recevra un message.

Après cette phase d'authentification du client, celui-ci pourra émettre et recevoir des paquets. Chaque paquet commence par un opCode permettant de savoir le but de celui-ci. Tous les opCode et leurs utilisations seront décrit à la toute fin du document.

La plupart des erreurs ne causent pas d'arrêt de la connexion.

Dans la plupart des cas, le serveur reçoit un acquittement à ses demandes qui contiendra le déroulement de celle-ci.

Le client peut aussi décider de fermer sa connexion avec le serveur sur lequel il est connecté.

De plus, un serveur peut alors demander la fusion à un autre serveur, il va alors envoyer un paquet contenant ces informations au serveur qui va lui envoyer un paquet d’acquittement pour lui signaler si la fusion est possible ou non.

Ce protocole est assez libre quant à l'utilisation de paquet volumineux mais sera très sensible à l'ordre des paquets (ce qui sera traité par TCP).

3. Relation avec les autres protocoles

Comme mentionné plusieurs fois, ChatFusion a été conceptualisé afin d'être utilisé avec Transmission Control Protocol (TCP).

Le protocole TCP établit une connexion entre deux machines.

La lecture et l'écriture sont alors bidirectionnelles à travers deux canaux de lecture et d'écriture. On peut fermer l'un des canaux sans fermer la connexion.

Le protocole TCP offrira des garanties très intéressantes pour notre protocole ChatFusion.

Le protocole TCP garantit qu'aucune donnée n'est perdue: il est fiable.

Le protocole TCP garantit que les données arrivent dans l'ordre où elles ont été envoyées et qu'elles ne sont pas modifiées: il conserve l'intégrité.

Cependant, le plus gros travail de ChatFusion sera de garantir la préservation des données puisque TCP ne s'en charge pas.

4. Protocole de la connexion initiale

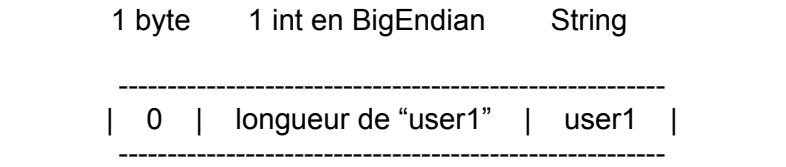
Une connexion sans mot de passe est établie avec le serveur lorsqu'une requête est envoyée avec le code 0 (un byte), la longueur du nom d'utilisateur choisi (un entier en BigEndian) et son nom d'utilisateur (les octets de la chaîne encodée en UTF-8).

Une connexion avec mot de passe est établie avec le serveur lorsqu'une requête est envoyée avec le code 1 (un byte), la longueur du nom d'utilisateur (un entier en BigEndian), son nom d'utilisateur (les octets de la chaîne encodée en UTF-8), la longueur du mot de passe (un entier en BigEndian) et le mot de passe (les octets de la chaîne encodée en UTF-8)

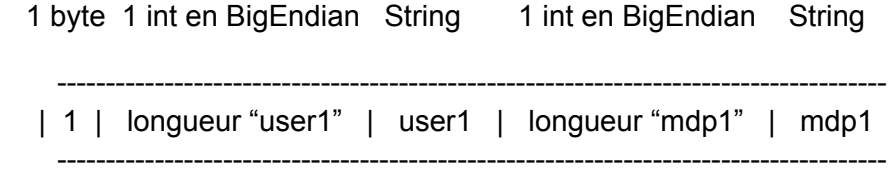
Si la connexion est bien initialisée, le serveur répond le chiffre 7 (un byte) suivi d'un id unique (un long en BigEndian), sinon ce sera juste le chiffre positif 8 (un byte) qui sera envoyé par le serveur.

L'exemple suivant montre les différentes étapes de l'initialisation d'une connexion pour chatter sur le serveur :

1. a) Le client A a sélectionné le nom d'utilisateur "user1" et se connecte sans mot de passe. Il enverra alors au serveur un paquet TCP comme suivant :

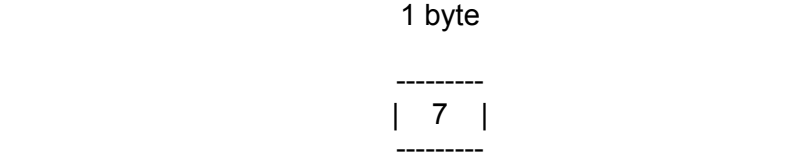


b) Le client A veut maintenant se connecter avec son nom d'utilisateur "user1" et se connecte avec le mot de passe "mdp1". Il enverra alors au serveur un paquet TCP comme suivant :

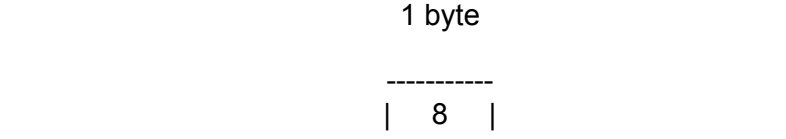


2. a) Le serveur reçoit un paquet avec le codeOpé 0 (connexion sans mot de passe) ou codeOpé 1 (connexion avec mot de passe), il le décode et se dessine alors deux scénarios :

Le serveur ne connaît aucun utilisateur connecté ou présent dans la base du nom de “user1” ou les identifiants sont corrects. Il renvoie alors un paquet avec l’opCode 7 :



Le serveur connaît déjà un utilisateur connecté du nom de “user1” ou les identifiants sont incorrects. Il renvoie alors un paquet comme suivant (indiquant une erreur de connexion) et ferme la connexion:

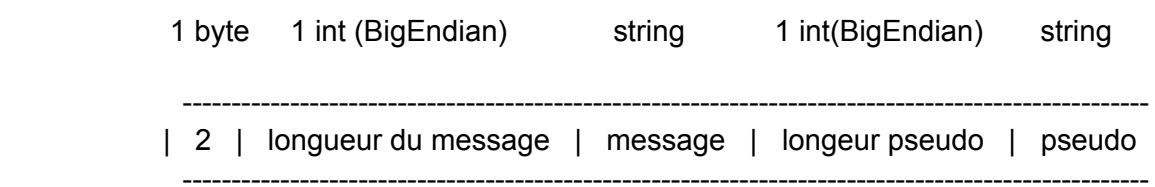


5. Protocole d’envoi de message

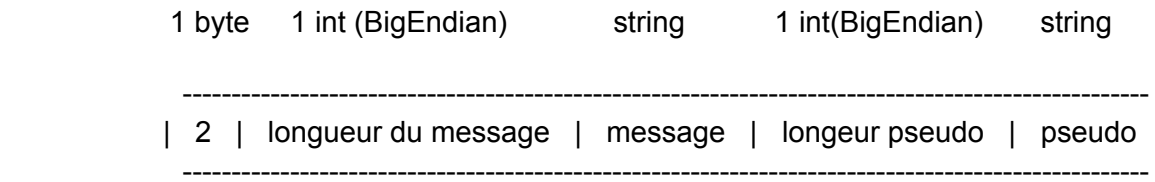
A.1. L’utilisateur, une fois connecté, va pouvoir envoyer des messages.

On utilisera l’opCode 2 pour l’envoi d’un message public à tous les utilisateurs connectés du serveur.

L’utilisateur va donc envoyer un paquet avec le code 2 (un byte), la longueur du message qu’il veut envoyer (un int en Big Endian), son message (les octets de la chaîne encodée en UTF-8), la longueur de son pseudo et son pseudo.



2. Le serveur va alors recevoir ce paquet, puis envoyer ce paquet à tous les utilisateurs.



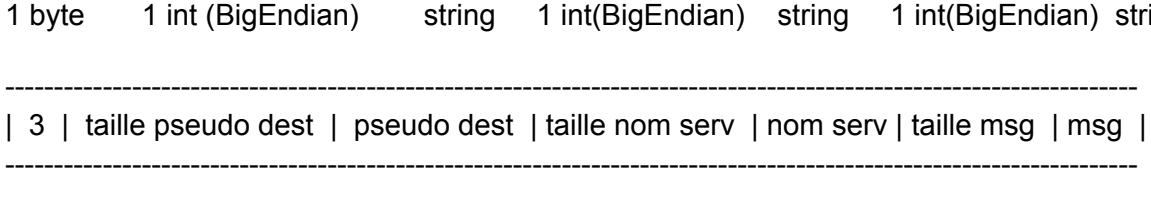
3. Le serveur va également envoyer le paquet ci-dessus à l’ensemble des serveurs avec lesquels il est fusionné. Les autres serveurs vont reconnaître le codeOp et le serveur qui émet le paquet (car déjà fusionné), ils vont uniquement retransmettre ce message à tous leurs clients.

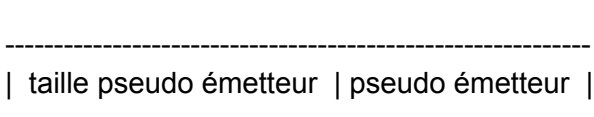
4. Les clients vont alors recevoir depuis le serveur le paquet avec le message et le pseudo de l’émetteur du message.

On utilisera l'opCode 3 pour l'envoi d'un message privé à un utilisateur connecté au serveur.

B.1. L'utilisateur va donc envoyer un paquet avec le code 3 (un byte), la taille du pseudo du destinataire(int en Big Endian), le pseudo du destinataire (les octets de la chaîne encodée en UTF-8), la taille du nom du serveur sur lequel le destinataire est connecté (un int en Big Endian), le nom du serveur sur lequel le destinataire est connecté (les octets de la chaîne encodée en UTF-8), la longueur du message qu'il veut envoyer (un int en Big Endian), son message (les octets de la chaîne encodée en UTF-8).

Il enverra alors au serveur un paquet TCP comme suivant :





2. Le serveur va alors recevoir ce paquet. Puis envoyer le paquet au client en question.

Si le serveur reconnaît que le nom du serveur dans le paquet est différent du sien alors il va envoyer le message au serveur correspondant. Sinon, le nom du serveur est le même que le sien alors, il va envoyer ce paquet au client correspondant.

On utilisera l'opCode 4 pour l'envoie d'un fichier privé à un utilisateur connecté au serveur.

C.1. L'utilisateur va donc envoyer un paquet avec le code 4 (un byte), la taille du pseudo du destinataire(int en Big Endian), le pseudo du destinataire (les octets de la chaîne encodée en UTF-8), la taille du nom du serveur sur lequel le destinataire est connecté (un int en Big Endian), le nom du serveur sur lequel le destinataire est connecté (les octets de la chaîne encodée en UTF-8), la taille du fichier qu'il veut envoyer (un int en Big Endian), son fichier (les octets du fichier).

Il enverra alors au serveur un paquet TCP comme suivant :

1 byte 1 int (BigEndian) string 1 int(BigEndian) string 1 int(BE) nom fichier

```
---------------------------------------------------------------------------------------------------------------------- |
4 | taille pseudo dest| pseudo dest | taille nom serv |nom serv | taille nom file | nom file |
----------------------------------------------------------------------------------------------------------------------
```

1 int(BE) octets du fichier 1 int (BE) string

```
-------------------------------------------------------------------
|  taille file  |  file  | taille nom émetteur | émetteur |
-------------------------------------------------------------------
```

2. Le serveur va alors recevoir ce paquet, le décoder. Puis envoyer le paquet au destinataire désigné.

Si le serveur reconnaît que le nom du serveur dans le paquet est différent du sien alors il va envoyer le message au serveur correspondant. Sinon, le nom du serveur est le même que le sien alors, il va envoyer ce paquet au client correspondant.

6. Protocole de fusion inter serveurs

Intéressons nous maintenant à la prise en charge de la fusion dans notre protocole. Nous allons expliquer brièvement le principe de fonctionnement car c'est un procédé un peu plus complexe que les notions expliquées précédemment dans ce document. De plus, c'est la principale spécificité du projet d'où le nom Chat**Fusion.**

Un serveur A peut demander à établir une fusion avec un autre serveur, par exemple le serveur B. On obtient alors un serveur AB sur lequel tous les clients du serveur A et tous les clients du serveur B pourrons communiquer. De plus, le serveur A peut aussi avoir envie de fusionner avec un serveur C, pour cela il va devoir faire la demande d'une fusion, si elle est acceptée alors nous aurons les serveurs A et B qui seront aussi fusionnés avec le serveur C.

On peut alors se poser légitimement la question “Comment ça marche ?”.

Pour ce faire, lors de la fusion, les serveurs A et B vont faire un transfert d'informations. En effet, chaque serveur stock les adresses des autres serveurs avec lesquelles ils sont déjà fusionnés.

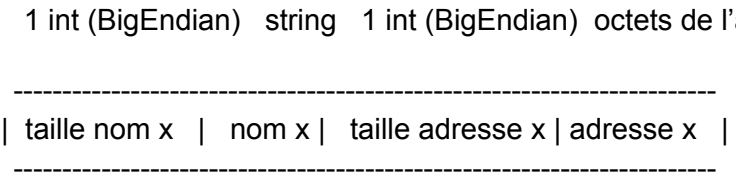
Reprenons l'exemple avec A et B.

Le serveur A va faire une demande de connexion avec le serveur B, il va donc envoyer le paquet suivant :

Dans le cas où A n'a pas déjà fusionné. Le paquet va alors posséder deux éléments l'opcode 5 et un int en BigEndian qui représente le nombre de serveurs qui sera à 0 car A n'a pas de fusion déjà faites.

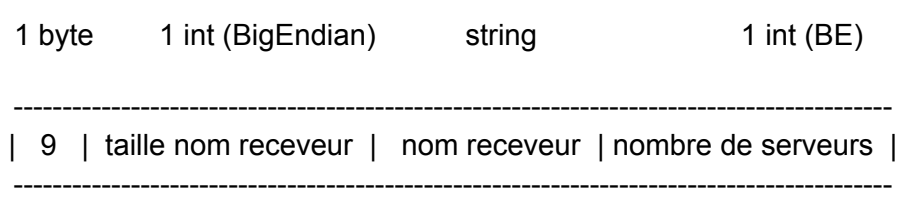
1 byte	1 int (BigEndian)	string	1 int (BigEndian)
5	taille nom Serv émetteur	nom Serv émetteur	nombre de serveurs

Dans les cas où il y a un ou plusieurs serveurs déjà fusionnés, alors le serveur émetteur va mettre à la fin du paquet tous ses serveurs à la suite. Sous la forme :

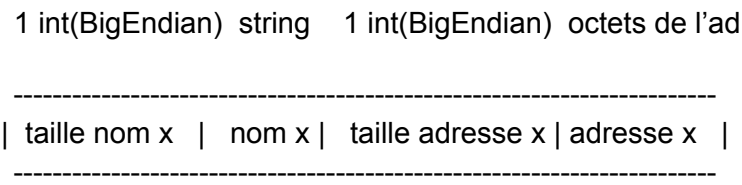


Voyons maintenant les réponses possibles du serveur recevant la demande. Si la demande est validée et acceptée. Alors le serveur va envoyer le paquet avec l'opcode 9 suivi de toutes les adresses qu'il possède qui représente les serveurs avec lesquelles il est déjà en fusion. C'est le même procédé que pour le paquet de demande.

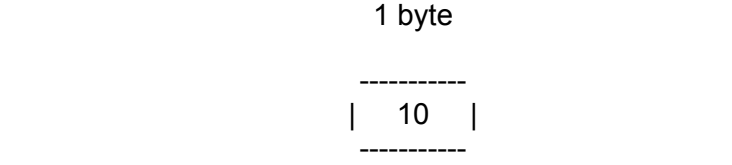
*ici "nom receveur" correspond au nom du serveur qui vient de revoir la demande



Dans les cas où il y a un ou plusieurs serveurs déjà fusionnés, alors le serveur émetteur va mettre à la fin du paquet tous ses serveurs à la suite. Sous la forme :

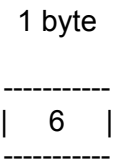


Si la demande n'est pas validée car un des serveurs déjà fusionné avec A ou B possède le même nom, le serveur va envoyer le paquet avec l'opcode 10 afin de notifier que la fusion est refusée.



7. Signal de fin de fusion

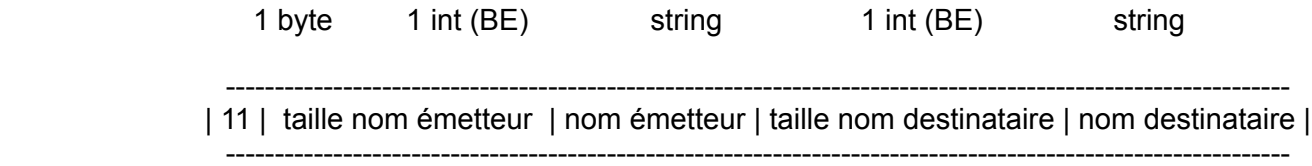
Lorsqu'un serveur souhaite s'arrêter, il va envoyer à tous les serveurs avec lesquels il est en fusion, un paquet avec le codeOp 6. De cette manière, tous les serveurs vont pouvoir cesser l'envoie de paquet avec le serveur en question.



8. Signal utilisateur introuvable

Lors d'un envoi de message privé il se peut que le serveur ne trouve pas le pseudo du destinataire. Ainsi pour avertir l'utilisateur, un paquet avec le codeOp 11 va être envoyé du serveur jusqu'à l'émetteur.

Le paquet TCP sera sous cette forme :



Ainsi, si le destinataire ne se situe pas dans le même serveur que l'émetteur, le serveur du destinataire va envoyer ce paquet au serveur de l'émetteur et le serveur de l'émetteur va pouvoir transmettre ce paquet à l'émetteur.

Le client à l'origine du message va donc recevoir ce paquet il pourrait ainsi être informé qu'il y avait une erreur dans la demande de message privé.

9. Les Paquets ChatFusion

ChatFusion possède plusieurs types de paquets différents, ceux-ci sont listés ci-dessous :

opcode (octet)	operation
0	Demande de connexion au serveur sans mot de passe
1	Demande de connexion au serveur avec mot de passe
2	Envoi à tous les utilisateurs connectés du serveur / Message global
3	Envoi d'un message à un utilisateur en particulier du serveur
4	Envoi d'un fichier à un utilisateur en particulier du serveur
5	Demande de fusion avec un serveur
6	Signal de fin de fusion (lorsqu'un des serveurs s'arrête)
7	Connexion au serveur acceptée
8	Connexion au serveur refusée (nom d'utilisateur déjà existant ou logs incorrects)
9	Fusion de serveur acceptée
10	Fusion de serveur refusée
11	Utilisateur introuvable