SPECIFICATION D'UN PROTOCOLE DE SESSION POUR UN SYSTEME DE MESSAGERIE INSTANTANEE

Rédacteurs :

Fayez DEBBABI
Fatima-Ezzahra ELAAMRAOUI

Sommaire :

1. Résumé

Ce document est une spécification d'un protocole de session pour un système de messagerie instantanée. Ce système gère des messageries privées et publiques suivant un fonctionnement bien déterminé.

2. Introduction

2.1. Définitions

Client: le logiciel du côté de l'utilisateur final qui interagit avec un serveur.

Utilisateur: La personne utilisant ce logiciel.

Serveur: Une machine qui fournit un service à différents clients. Dans notre cas le service fourni est un service de messagerie.

Message: Les données échangées entre deux unités, soit entre un client et un serveur dans notre cas ou encore entre deux clients.

Messagerie instantanée: échange instantané de messages textuels entre deux clients

Messagerie publique: échange de messages entre TOUS les utilisateurs connectés à travers le serveur et ayant un statut publique.

Messagerie centralisée: échange de messages entre un ensemble d'utilisateurs qui passe forcément par le serveur.

Messagerie décentralisée: échange de messages entre un ensemble d'utilisateurs qui ne passe pas par le serveur.

Messagerie privée: échange de messages entre un ensemble d'utilisateurs connectés soit d'une manière centralisée ou pas et ayant un statut privé.

2.2. Concept du protocole

Le protocole de session permet aux clients de se connecter et d'échanger des messages que ce soit dans une messagerie privée (centralisée ou décentralisée) ou publique. Pour le fonctionnement du protocole, le client entre dans une messagerie publique dès qu'il se connecte au serveur, une messagerie publique donc se déroulera.

Le client possède la possibilité de demander la liste des utilisateurs préalablement connectés au serveur afin d'initier une messagerie privée avec au moins un parmi eux, il peut en même temps choisir le niveau de centralisation de la messagerie (centralisée ou pas). Une fois la messagerie privée est créée dès que la liste des utilisateurs connectés au serveur ne contient qu'un seul élément, la messagerie est automatiquement supprimée et le dernier utilisateur connecté sera tout simplement notifié de la clôture de la messagerie et il sera retourné à la messagerie publique.

Le protocole en question devra fonctionner au-dessus d'UDP, tout en garantissant la fiabilité.

3. Protocole

3.1. Principe du protocole

Pour permettre l'échange de messages entre les clients et le serveur, le protocole de session est ainsi décrit :

* Les utilisateurs ou clients doivent se connecter au serveur avant toute opération. Pour la toute première connexion d'un client, ce dernier devrait envoyer au serveur un message contenant son USER_NAME qui devrait être unique dans le réseau. Le serveur lui affecte de retour un identifiant ID_USER et le lui envoie de retour. Le client garde son ID_USER pour l'utiliser dans la suite de ces messages. Une fois la connexion établie, les clients peuvent échanger des messages entre eux à travers le serveur ou pas ce qui fait de la messagerie centralisée ou décentralisée. Le serveur par contre ne peut pas de son côté se connecter aux clients.

* Dès que le client se connecte au serveur, il peut lui envoyer un message lui demandant d'initier une messagerie privée.

Pour établir une connexion le client doit envoyer vers le serveur un message particulier de type CONNEXION, et l'opération se répète régulièrement avec un intervalle de 45 secondes entre deux envois successifs pour mentionner que le client est encore connecté au serveur.

Quelques messages doivent être envoyés du client vers le serveur et pas du serveur vers le client donc pas de possibilité que le serveur initie un échange avec le client pour des messages particuliers (exemple: message de connexion),
Le serveur a donc la possibilité de répondre aux différentes demandes des clients pour répondre aux différentes requêtes des clients (demande de changement de la messagerie privée, demande d'avoir la liste des messageries privées disponibles…) et d'initier les échanges pour certains types des messages(exemple: envoi des demandes d'ajout des clients à une messagerie privée).
Ce qui est donc intuitif c'est de regrouper les différents messages échangés faits par le client en deux catégories :

*Catégorie CLIENT_GET : ce qui signifie que le client lors de son envoi d'un message au serveur attend forcément une réponse avec des données de sa part (demande de liste des clients connectés en une messagerie privée, demande de liste des messageries privées …).

*Catégorie CLIENT_SET : ce qui signifie que le client lors de son envoi d'un message au serveur attend une réponse sans données particulières (acquittement, déconnexion du client…).

Pour ces deux catégories, le serveur doit donc répondre à chaque message du client en lui envoyant de retour un autre message de catégorie SERVER_RESPONSE.

Afin d'identifier les différents messages envoyés vers le serveur de la part du client, les messages doivent contenir un champ TYPE, qui sera détaillé dans une section ultérieure, signifiant le type de message échangé avec le serveur, qui doit être connu à l'avance côté client et serveur.

Cas particuliers :

Si le serveur reçoit un message de la part du client mais dont le type n'est pas connu par le serveur, ce dernier élimine ce message en envoyant au client un message de la catégorie SERVER_RESPONSE avec un type propre aux messages d'erreur en mentionnant la raison pour laquelle la requête a échoué.

Les messages susceptibles d'être envoyés par le client au serveur sont : CONNEXION, CREER_MESSAGERIE_PRIVEE, DEMANDER_LISTE_MESSAGERIE, DEMANDE_ACCES_MESSAGERIE_PRIVEE, DEMANDER_LISTE_CLIENT, INVITER_MESSAGERIE_PRIVEE, DEMANDE_MESSAGERIE_PRIVEE, TEXTE, CHANGER_MESSAGERIE, QUITTER_MESSAGERIE_PRIVEE, DECONNEXION.

Les réponses du serveur faisant partie de la catégorie SERVER_RESPONSE seront de type : REPONSE_CONNEXION, REPONSE_CREER_MESSAGERIE_PRIVEE, REPONSE_DEMANDER_LISTE_MESSAGERIE, REPONSE_DEMANDE_ACCES_MESSAGERIE_PRIVEE, REPONSE_DEMANDER_LISTE_CLIENT, REPONSE_INVITER_MESSAGERIE_PRIVEE, REPONSE_DEMANDE_MESSAGERIE_PRIVEE, REPONSE_TEXTE, REPONSE_CHANGER_MESSAGERIE, REPONSE_QUITTER_MESSAGERIE_PRIVEE, REPONSE_DECONNEXION.

3.2. Fiabilité du protocole

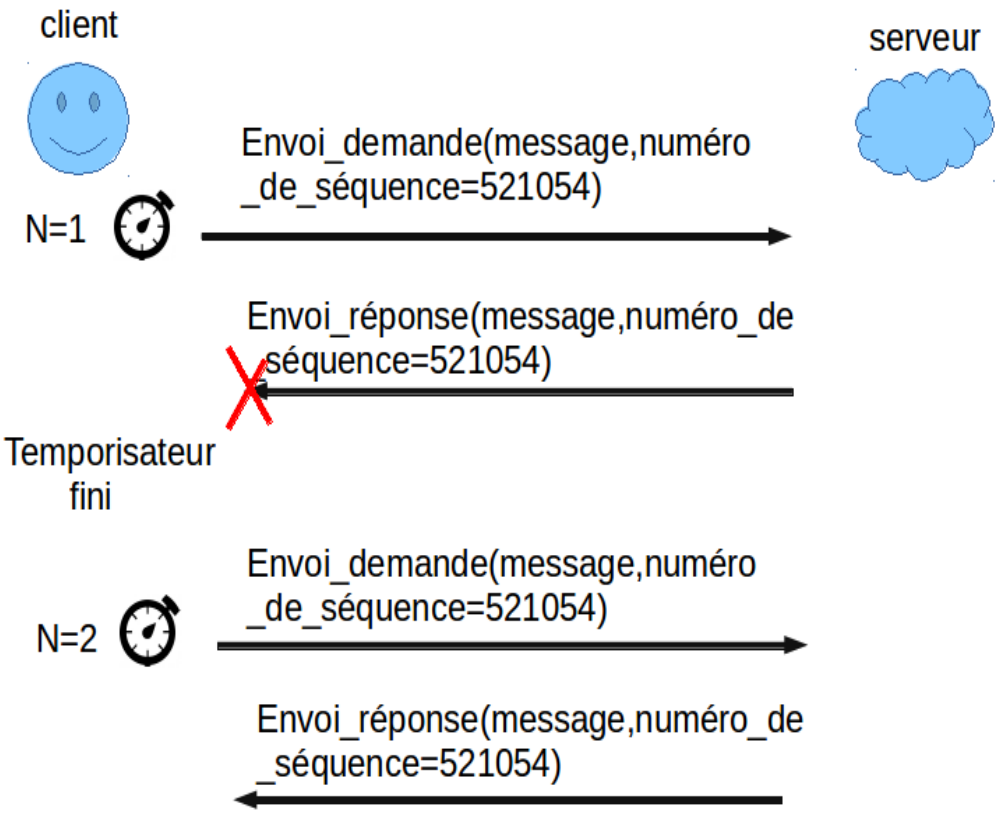
Notre protocole devra forcément assurer la fiabilité de la liaison. En effet, quand on envoie un message que ce soit du serveur au client ou du client au serveur, il va falloir à chaque fois vérifier qu'il a bien été reçu. Si le message a été perdu, il devra être retransmis jusqu'à réception de ce dernier. Il ne faut pas par contre dépasser un nombre d'essais bien fixé depuis le début.

Afin de garantir la fiabilité, une méthode similaire à celle utilisée au niveau du TCP, une technique possible de résoudre le problème de fiabilité consiste à utiliser un numéro de séquence NUM_SEQ relatif à chaque message échangé qui doit être le même pour une demande et pour la réponse qui lui correspond. Un temporisateur est déclenché à chaque envoi d'une demande, en attente d'une réponse avec le numéro de séquence correspondant, une fois une réponse de la part du destinataire est reçue avec le numéro de séquence qui correspond, l'émetteur (que ce soit client ou serveur) élimine directement la tentative d'un autre envoi du même message portant le même numéro de séquence et arrête le temporisateur. Si le temporisateur se termine tout en restant sans réponse sur le message échangé, une nouvelle tentative d'envoi se déclenche avec une demande contenant le même numéro de séquence et avec un nouveau déclenchement du temporisateur et incrémentation du nombre des essais faits N.

On choisira la période du temporisateur comme T=100 ms pour des raisons des tests.

Ce processus est limité à M tentatives successives d'envoi du même message. M correspond donc à un degré de fiabilité particulier à choisir par celui de l'émetteur (Client ou serveur).

Explication graphique :



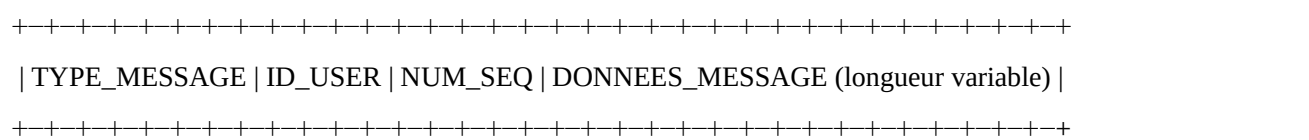
4. Description des messages

4.1. Types des messages

Code qui représente le champ TYPE_MESSAGE	MESSAGE
00001	CONNEXION
00010	REPONSE_CONNEXION
00011	CREER_MESSAGERIE_PRIVEE
00100	REPONSE_CREER_MESSAGERIE_PRIVEE
00101	DEMANDER_LISTE_MESSAGERIE
00110	REPONSE_DEMANDER_LISTE_MESSAGERIE
00111	DEMANDE_ACCES_MESSAGERIE_PRIVEE
01000	REPONSE_ACCES_MESSAGERIE_PRIVEE
01001	DEMANDER_LISTE_CLIENT
01010	REPONSE_DEMANDER_LISTE_CLIENT
01011	INVITER_MESSAGERIE_PRIVEE
01100	REPONSE_INVITER_MESSAGERIE_PRIVEE
01101	DEMANDE_MESSAGERIE_PRIVEE
01110	REPONSE_MESSAGERIE_PRIVEE
01111	TEXTE
10000	REPONSE_TEXTE
10001	CHANGER_MESSAGERIE
10010	REPONSE_CHANGER_MESSAGERIE
10011	QUITTER_MESSAGERIE_PRIVEE
10100	REPONSE_QUITTER_MESSAGERIE_PRIVEE
10101	DECONNEXION
10110	REPONSE_DECONNEXION

4.2. Format des messages

Tous les messages ont le même format comme indiqué dans la figure en dessous (Figure 2) :



TYPE_MESSAGE: comme indiqué dans le tableau ci-dessus il caractérise chaque message et sera codé sur 5 bits ID_USER: ce champ contient l'identificateur de chaque utilisateur et sera codé sur 1 octet pour utiliser 254 (256-2) clients maximum dans le logiciel, les deux possibilités restantes correspondent bien à un ID_USER pour le serveur avec lequel on va communiquer (on va choisir ici 11111111) et un ID_USER par défaut (on va choisir 00000000) qui joue le rôle d'un initiateur pour les clients afin que la procédure de l'inscription puisse avoir lieu.

NUM_SEQ : ce champ contient le numéro de séquence dont le but a déjà été précisé dans le paragraphe 3.2 Fiabilité du protocole. 2 Octets lui seront réservés afin de pouvoir atteindre 65535 à cause de l'évolution rapide du nombre des échanges.

DONNEES_MESSAGE : de longueur différente selon le type de message, il sera détaillé dans les parties suivantes.

4.3. Format des données de chaque message

Chaque message contient des données qui lui sont spécifiques et qui ont un format donné :

4.3.1 CONNEXION

S'il s'agit de la première connexion du client, le champ ID_USER sera mis à zéro comme mentionné précédemment, et les données de ce message contiennent un nom d'utilisateur USER_NAME que fournit le client et qui ne devrait pas dépasser en longueur 10 octets (maximum dix caractères autorisés qu'ils soient des chiffres, des lettres ou des symboles, un seul caractère n'est pas autorisé qui est le UN_DELIMITER (on prendra l'élément '#' dans notre protocole qui nous permettra d'indiquer la fin du USER_NAME).

+–+

| USER_NAME (1-10 octets) | UN_DELIMITER : 1 octet |

+–+

4.3.2 REPONSE_CONNEXION

Si c'est la première connexion du client, le serveur génère un identifiant pour ce client ID_USER et le lui envoie comme réponse à sa demande de CONNEXION. C'est un nouveau ID_USER mentionné dans le champ de données.

Le serveur renvoie également le champ STATUT_INSCRIPTION renseignant le client sur la validation ou l'échec de sa demande d'inscription qui se fait lors de cette première connexion.

2 éléments binaires seront réservés pour le champ STATUT_INSCRIPTION pour coder les 3 cas possibles qui sont :

* INSCRIPTION_VALIDEE : 00

* USER_NAME_INVALIDE : 01

* USER_NAME_EXISTANT : 10

+–+

| STATUT_ INSCRIPTION (2 bits) | ID_USER (1 octet) |

+–+

Si ce n'est pas la première connexion du client, le serveur ne renvoie de retour que le STATUT_CONNEXION pour lui confirmer ou pas sa connexion au sein du réseau. Ce champ sera codé sur 1 bit et 2 cas seront possibles :

* CONNEXION_REUSSIE : représentée par l'élément binaire 0.

* CONNEXION_ERREUR : représentée par l'élément binaire 1.

+−+−+−+−+−+−+−+−+−+−+−+−+−+−+

| STATUT_ CONNEXION (1bit) |

+−+−+−+−+−+−+−+−+−+−+−+−+−+−+

4.3.3 CREER_MESSAGERIE_PRIVEE

Une fois le client est connecté au serveur, il aura la possibilité de créer une messagerie privée, pour cela il doit fournir comme données de ce message les informations suivantes :

MODE_MESSAGERIE qui doit être soit 0 pour une messagerie centralisée ou 1 pour une décentralisée, donc un seul bit lui sera réservé.

NOM_MESSAGERIE qui comportera 50 octets et qui sera traité de la même manière que le USER_NAME en mettant à la fin un M_DELIMITER (on choisit également '#' pour pouvoir donc délimiter le nom de la messagerie).

NUM_MAX : qui représentera le nombre maximal des clients qui peuvent se connecter à cette messagerie privée. Ce nombre est proposé par le client et devra être compris entre (2,254) et sera par conséquent codé sur 1 octet conformément au nombre des utilisateurs du logiciel (254 utilisateurs maximum du logiciel),

+−+

| MODE_MESSAGERIE (1bit) | NOM_MESSAGERIE | M_DELIMITER| NUM_MAX (1octet) |

+−+

4.3.4 REPONSE_CREER_MESSAGERIE_PRIVEE

En réponse au message précédent, le serveur renvoie de retour un ID_MESSAGERIE propre à la messagerie qui vient d'être créée sous la demande du client, ce champ servira à identifier la messagerie privée et sera codé sur 1 octet de (1 à 255). Ce champ par défaut sera affecté à 00000000 pour mentionner que la messagerie est publique. Le champ STATUT_CREATION servira à valider au client ou pas la création de sa messagerie, quatre cas sont envisageables, codés essentiellement sur 2 bits :

* CREATION_VALIDEE :00

* NOM_MESSAGERIE_ERREUR :01

* NOM_MESSAGERIE_EXISTANT :10

* NUM_MAX_PETIT :11

+−+

| STATUT_CREATION (2bits) | ID_MESSAGERIE |

+−+

4.3.5 DEMANDER_LISTE_MESSAGERIE

Ce message envoyé du client au serveur ne comportera pas spécialement de données, c'est au serveur de lui fournir une liste des messageries privées.

+–+–+
| |
+–+–+

4.3.6 REPONSE_DEMANDER_LISTE_MESSAGERIE

En réponse au message DEMANDER_LISTE_MESSAGERIE provenant du client, le serveur envoie un champ au début qui contient le nombre de messageries privées NUM_MESSAGERIE codé sur un 1 octet et une liste de messageries privées MESSAGERIES qui s'avère de longueur variable selon le NUM_MESSAGERIE.

+–+

| NUM_MESSAGERIE (1 octet) | MESSAGERIES (longueur variable) |

+–+

Pour chaque messagerie, le paquet envoyé aura le format suivant (ce qui veut dire que le bloc ci-dessous sera envoyé autant de fois que le nombre de messageries NUM_MESSAGERIE) :

+–

| MODE_MESSAGERIE (1 bit) | NOM_MESSAGERIE | M_DELIMITER | NUM_CLIENT |

+–

MODE_MESSAGERIE, NOM_MESSAGERIE, M_DELIMITER sont pareils que les champs déjà mentionnés dans les messages précédents.

NUM_CLIENT : précise le nombre des clients de la messagerie en question. Ce champ est de longueur 1 octet.

4.3.7 DEMANDE_ACCES_MESSAGERIE_PRIVEE

Le client envoie ce message au serveur lorsqu'il souhaite avoir accès à une messagerie privée bien précise, il n'a donc qu'à envoyer comme donnée de ce message le NOM_MESSAGERIE de la messagerie en question.

+–+–+–+–+–+–+–+–+–+–+–+–+–+–+–+–+–+

| NOM_MESSAGERIE | M_DELIMITER |

+–+–+–+–+–+–+–+–+–+–+–+–+–+–+–+–+–+

4.3.8 REPONSE_ACCES_MESSAGERIE_PRIVEE

Dans ce cas, deux cas se présentent selon le mode de centralisation de la messagerie. Dans ces deux cas qui suivent, le serveur envoie en mentionnant comme premier champ le mode de la messagerie pour que le client puisse faire le traitement particulier selon le reste des champs selon le mode qui est reçu en premier lieu. On va commencer par présenter le premier cas où la messagerie est centralisée.

Cas1 : Pour ce cas c'est une réponse à accéder à une messagerie privée en mode centralisé. Le serveur n'envoie en retour au client qu'un champ STATUT_ACCES codé sur 2 bits pour lui confirmer ou pas son accès à la messagerie privée, 3 cas sont possibles, d'où les 2 bits réservés à ce champ-là :

* ACCES_REUSSI : 00

* NOM_MESSAGERIE_INEXISTANT : 01

* MAX_CLIENT_ATTEINT : 10

+–+

| MODE_MESSAGERIE (1 bit) | STATUT_ACCES (2 bits) |

+–+

Cas2 : Pour ce cas c'est une réponse à accéder à une messagerie privée en mode décentralisée, le client doit donc avoir des informations qui sont les adresses IPs et les numéros des ports des clients dans cette messagerie afin qu'il puisse entrer en contact avec eux sans passer par le serveur. Afin de faire cela, le serveur envoie une liste des Adresses IPs, de ports et les USER_NAME des personnes dans cette messagerie.

Toutefois, le serveur garde une trace sur les différentes évolutions de la messagerie (déconnexion d'un client, ajout d'un client…).

Le champ de données inclura donc :

+–

|MODE_MESSAGERIE | STATUT_ACCES (2 bits) | NOMBRE_CLIENTS (1 octet) | CLIENTS (longueur variable) |

+–

STATUT_ACCES représente exactement les mêmes propriétés que le champ STATUT_ACCES mentionné juste avant.

Le champ CLIENTS contient autant de fois que le NOMBRE_CLIENTS le paquet suivant.

IP_ADRESS représente l'adresse IP du client, PORT_NUMBER représente le numéro de port du client en écoute, USER_NAME son nom d'utilisateur :

+–+

| IP_ADRESS (4 octets) | PORT_NUMBER (2 octets) | USER_NAME (longueur variable) | M_DELIMITER |

+–+

4.3.9 DEMANDER_LISTE_CLIENT

En ce qui concerne ce message, le client devrait préciser le NOM_MESSAGERIE de la messagerie dont il souhaite obtenir la liste des clients

+–+–+–+–+–+–+–+–+–+–+–+–+–+–+–+–+

| NOM_MESSAGERIE | M_DELIMITER |

+–+–+–+–+–+–+–+–+–+–+–+–+–+–+–+–+

4.3.10 REPONSE_DEMANDER_LISTE_CLIENT

En analogie avec le message REPONSE_DEMANDER_LISTE_MESSAGERIE le serveur renvoie en tant que données de ce message un champ NUM_CLIENT au début qui contient le nombre de clients de la messagerie privée déjà spécifiée dans le message précédent par le client (1 octet) et une liste de clients CLIENTS qui est de longueur variable.

+–+

| NUM_CLIENT (1 octet) | CLIENTS (longueur variable) |

+–+

Pour chaque client, le paquet envoyé aura le format suivant (ce qui veut dire que le bloc ci-dessous sera envoyé autant de fois que le nombre de clients NUM_CLIENT) :

+–+–+–+–+–+–+–+–+–+–+–+–+–+–+

| USER_NAME |UN_DELIMITER |

+–+–+–+–+–+–+–+–+–+–+–+–+–+–+

USER_NAME, UN_DELIMITER sont pareils que les champs déjà mentionnés dans les messages précédents.

4.3.11 INVITER_MESSAGERIE_PRIVEE

Dans ce message, le client qui invite indiquera le USER_NAME de la personne qu'il aimerait inviter à la messagerie dont il fait partie même s'il n'est pas le créateur de la messagerie en question.

+–+–+–+–+–+–+–+

| USER_NAME |

+–+–+–+–+–+–+–+

4.3.12 REPONSE_INVITER_MESSAGERIE_PRIVEE

Ce message ne comportera spécialement aucune donnée, c'est juste un acquittement pour confirmer au client la réception de son message.

+–+–+–+

| |

+–+–+–+

4.3.13 DEMANDE_MESSAGERIE_PRIVEE

Suite à la demande d'un premier client dans son message INVITER_MESSAGERIE_PRIVEE où il invite un second client, le serveur envoie à ce dernier le message DEMANDE_MESSAGERIE_PRIVEE pour lui demander de participer à la même messagerie privée, en lui communiquant le USER_NAME du premier client qui l'a invité, un UN_DELIMITER, le MODE_MESSAGERIE (centralisée ou décentralisée) et le NOM_MESSAGERIE avec nécessairement un M_DELIMITER comme déjà expliqué.

+–+

| USER_NAME | UN_DELIMITER | MODE_MESSAGERIE (1 bit) | NOM_MESSAGERIE | M_DELIMITER |

+–+

4.3.14 REPONSE_MESSAGERIE_PRIVEE

Ce message ne comportera qu'une seule donnée REPONSE codée sur 1 bit :

* accepter l'invitation et rejoindre la messagerie privée : 0

* refuser de participer à la messagerie privée : 1

+−+−+−+−+−+−+−+−+

| REPONSE (1 bit) |

+−+−+−+−+−+−+−+−+

A partir du fait que le client accepte ou pas, le serveur s'occupera de faire migrer le client vers la nouvelle messagerie.

4.3.15 TEXTE

Cas 1 : Pour ce type de message, que ce soit dans la messagerie publique ou privée (centralisée) le client envoie CHAT en tant que champ de données dont la longueur est CHAT_LONGUEUR envoyé en premier lieu. CHAT comporte le texte brut de la discussion à échanger entre les clients de la même messagerie.

+−+

| CHAT_LONGUEUR (2 octets) | CHAT (longueur variable) |

+−+

CHAT_LONGUEUR : devrait être codé sur 2 octets.

CHAT : de longueur variable selon le CHAT_LONGUEUR.

Cas 2 : Ce cas correspond à un échange des textes dans une messagerie privée décentralisée, c'est pour cela qu'à partir du message DEMANDE_ACCES_MESSAGERIE_PRIVEE, le client va récupérer la liste des adresses IPs qui appartiennent aussi à la même messagerie privée que lui. Il va donc envoyer son texte à tous les clients un par un en utilisant leurs adresses IPs.

+−+

| CHAT_LONGUEUR (2 octets) | CHAT (longueur variable)
|+−+

4.3.16 REPONSE TEXTE

Ce message ne comportera spécialement aucune donnée, c'est juste un acquittement pour confirmer au client la réception de son message, qui est envoyé par chaque entité qui reçoit le texte, que ce soit serveur pour le mode centralisé ou clients pour le mode décentralisé.

+−+−+−+

| |

+−+−+−+

4.3.17 CHANGER_MESSAGERIE

Afin de changer la messagerie actuelle dont fait partie le client pour migrer vers une autre, ce dernier envoie ce message en précisant le nom de la nouvelle messagerie NOM_MESSAGERIE toujours associé avec un M_DELIMITER.

+–+–+–+–+–+–+–+–+–+–+–+–+–+–+–+–+

| NOM_MESSAGERIE | M_DELIMITER |

+–+–+–+–+–+–+–+–+–+–+–+–+–+–+–+–+

4.3.18 REPONSE_CHANGER_MESSAGERIE

Dans ce type de message, deux cas se présentent selon le mode de centralisation de la messagerie demandée, le serveur envoie en mentionnant comme premier champ le mode de la messagerie pour que le client puisse faire le traitement particulier selon le reste des champs selon le mode qui est reçu en premier lieu. On va commencer par présenter le premier cas où la messagerie est centralisée

Cas1: Pour ce cas c'est une réponse à migrer vers une messagerie privé en mode centralisé. Le serveur n'envoie en retour au client qu'un champ STATUT_ACCES codé sur 2 bits pour lui confirmer ou pas son accès à la messagerie privée voulue, 3 cas sont possibles, d'où les 2 bits réservés à ce champ-là :

* ACCES_REUSSI : 00

* NOM_MESSAGERIE_INEXISTANT : 01

* MAX_CLIENT_ATTEINT : 10

+–+

| MODE_MESSAGERIE (1 bit) | STATUT_ACCES (2 bits) |

+–+

Cas2 : Pour ce cas c'est une réponse à migrer vers une messagerie privée en mode décentralisée, le client doit donc avoir des informations qui sont les adresses IPs et numéro des ports des clients dans cette messagerie afin qu'il puisse entrer en contact avec eux sans passer par le serveur. Afin de faire cela, le serveur envoie une liste des Adresses IPs, de ports et les USER_NAME des personnes dans cette messagerie.

Au contraire, le serveur garde une trace sur les différentes évolutions de la messagerie (déconnexion d'un client, ajout d'un client…).

Le champ de données inclura donc :

+–+

|MODE_MESSAGERIE|STATUT_ACCES (2 bits)|NOMBRE_CLIENTS(1 octet)|CLIENTS (longueur variable)|

+–+

STATUT_ACCES représente exactement les mêmes propriétés que le champ STATUT_ACCES mentionné juste avant, le champ CLIENTS contient autant de fois que le NOMBRE_CLIENTS le paquet suivant dans IP_ADRESS représente l'adresse IP du client, PORT_NUMBER représente le numéro de port du client en écoute, USER_NAME son nom d'utilisateur :

+–+

| IP_ADRESS (4 octets) | PORT_NUMBER (2 octets) | USER_NAME (longueur variable) | M_DELIMITER |

+–+

Remarque :

Après chaque changement passé avec succès, le serveur vu qu'il garde une trace sur chaque messagerie il va changer au niveau de sa base de données que le nombre des clients connectés à la messagerie ancienne décrémente de 1 afin de faire une synchronisation et garder des données correctes sur le nombre des clients connectés sur chaque messagerie. Cette gestion de faire éliminer l'utilisateur de sa messagerie précédente n'est faite que lorsqu'on obtient un succès de l'opération!! Si une erreur s'est produite, rien ne se passe pour que toutes les informations restent cohérentes au niveau du serveur.

4.3.19 QUITTER_MESSAGERIE_PRIVEE

Le client envoie ce message quand il souhaite quitter la messagerie privée dont il fait partie migrer vers la messagerie publique. Aucune donnée ne sera envoyée avec ce message.

+−+−+−+

| |

+−+−+−+

4.3.20 REPONSE_QUITTER_MESSAGERIE_PRIVEE

Le serveur n'envoie en retour au client qu'un champ STATUT_EXIT codé sur 1 bits pour lui confirmer ou pas sa demande, 2 cas sont possibles, d'où le 1 bit réservé à ce champ-là :

* DEMANDE_REUSSIE : 0
* CLIENT_INEXISTANT : 1

+−+−+−+−+−+−+−+−+−+

| STATUT_EXIT (1 bit) |

+−+−+−+−+−+−+−+−+−+

Si le STATUT_EXIT est mis à 0 l'opération s'est déroulée avec succès et donc il faut que le serveur gère l'élimination du client de sa précédente messagerie afin de garder une cohérence de données.

Si STATUT_EXIT est mise à 1 rien ne se passe et le client reste dans messagerie précédente.

4.3.21 DECONNEXION

Aucune donnée ne sera spécialement envoyée du client au serveur.

+−+−+−+

| |

+−+−+−+

4.3.22 REPONSE_DECONNEXION

Le serveur n'envoie en retour au client qu'un champ STATUT_DECONNEXION codé sur 1 bits pour lui confirmer ou pas sa demande, 2 cas sont possibles, d'où le 1 bit réservé à ce champ-là :

* DEMANDE_SUCCESS : 0
* ERREUR_DECONNEXION : 1

+–+–+–+–+–+–+–+–+–+–+–+–+–+–+–+–+

| STATUT_ DECONNEXION (1 bit) |

+–+–+–+–+–+–+–+–+–+–+–+–+–+–+–+–+

Si le STATUT_ DECONNEXION est mis à 0 l'opération s'est déroulée avec succès et donc il faut que le serveur gère l'élimination du client de sa précédente messagerie afin de garder une cohérence de données.

Si STATUT_ DECONNEXION est mise à 1 rien ne se passe et le client reste dans messagerie précédente.

5. Exemples de messages d'erreur

5.1 CONNEXION

5.1.1 USER_NAME_INVALIDE

Cette erreur se produit lorsque le client rentre un nom d'utilisateur USER_NAME erroné, c'est-à-dire qui ne respecte pas les règles déjà mentionnées (qui ne contient pas le caractère de délimitation # , et qui ne dépasse pas 10 octets en longueur).

5.1.2 USER_NAME_EXISTANT

Cette erreur a lieu lorsque le client rentre un nom d'utilisateur USER_NAME qui existe déjà au niveau du serveur.

5.2 REPONSE_CREER_MESSAGERIE_PRIVEE

5.2.1 NOM_MESSAGERIE_ERREUR

Cette erreur se produit lorsque le client essaie de créer une messagerie dont le nom est non conforme aux règles déjà mentionnées (le client doit donner un nom de messagerie NOM_MESSAGERIE qui ne contient pas le caractère de délimitation '#', qui ne dépasse pas 50 octets en longueur et qui n'est pas vide)

5.2.2 NOM_MESSAGERIE_EXISTANT

Cette erreur se produit lorsque le client donne un nom de messagerie NOM_MESSAGERIE qui existe déjà (qui a déjà été utilisé pour nommer une messagerie privée)

5.2.3 NUM_MAX_PETIT

Lors de la création d'une messagerie privée, le client devra donner un nombre maximal d'utilisateurs de cette messagerie NUM_MAX. Si ce nombre est inférieur strictement à 2, l'erreur NUM_MAX_PETIT est produite.

5.3 REPONSE_ACCES_MESSAGERIE_PRIVEE

5.3.1 NOM_MESSAGERIE_INEXISTANT

Ce problème a lieu lorsque le client demande l'accès à une messagerie dont le nom n'existe pas au niveau du serveur.

5.3.2 MAX_CLIENT_ATTEINT

Si un client demande au serveur d'accéder à une messagerie alors que le nombre maximal des clients pour cette messagerie est déjà atteint, le serveur renvoie cette erreur.

5.4 REPONSE_CHANGER_MESSAGERIE

5.4.1 NOM_MESSAGERIE_INEXISTANT

Cette erreur a lieu quand le client demande de migrer vers une autre messagerie dont le nom n'existe pas du côté de serveur.

5.4.2 MAX_CLIENT_ATTEINT

Si un client demande au serveur de migrer vers une autre messagerie alors que le nombre maximal des clients pour cette messagerie est déjà atteint, le serveur lui signale cette erreur.

5.5 REPONSE_QUITTER_MESSAGERIE_PRIVEE

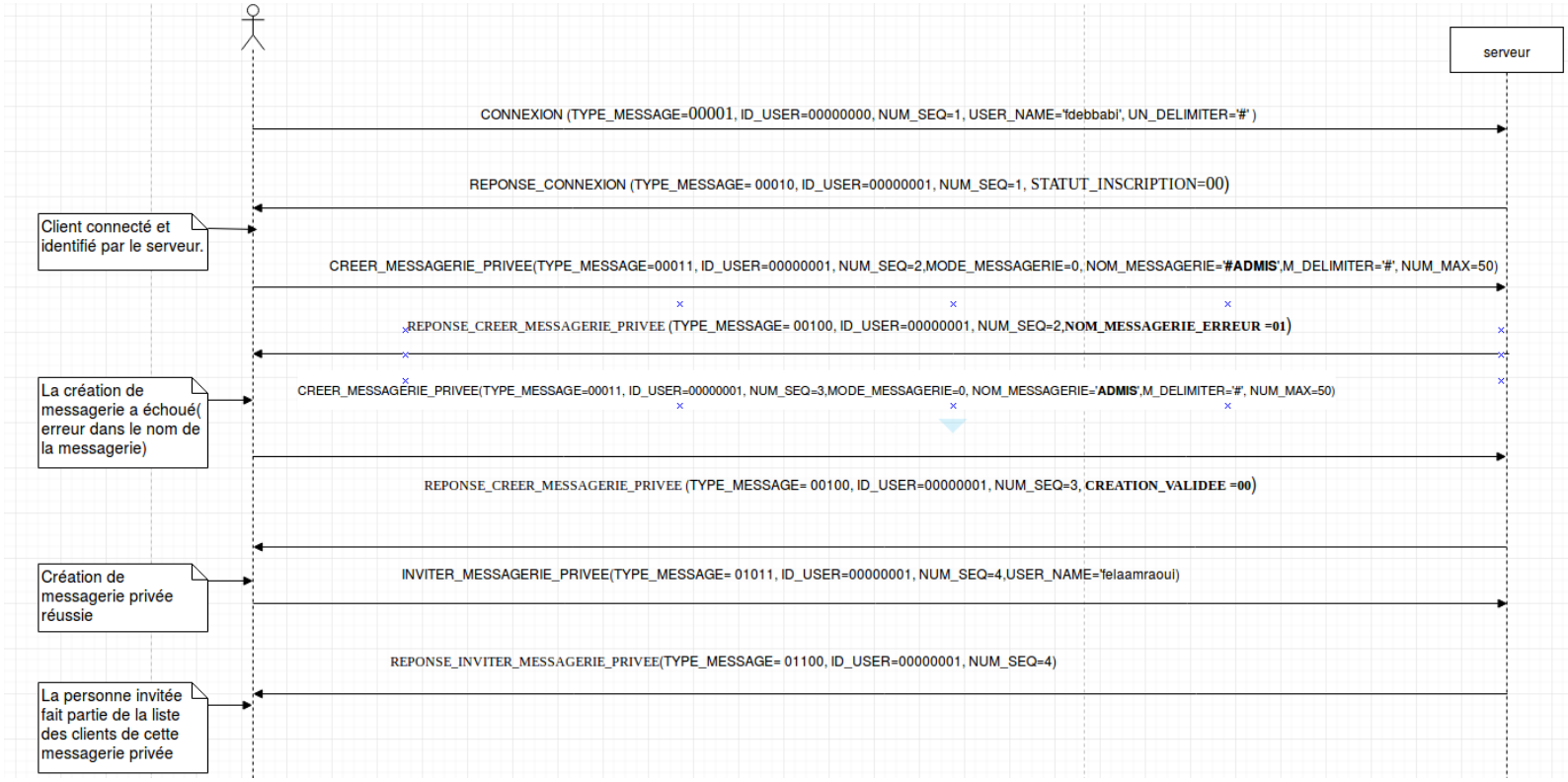
5.5.1 CLIENT_INEXISTANT

Si le client demande de quitter une messagerie privée alors qu'il n'en fait même pas partie, il n'appartient qu'à la messagerie publique, le serveur lui retourne cette erreur puisqu'il ne le trouve dans aucune messagerie privée.

6. Scénarios d'usage

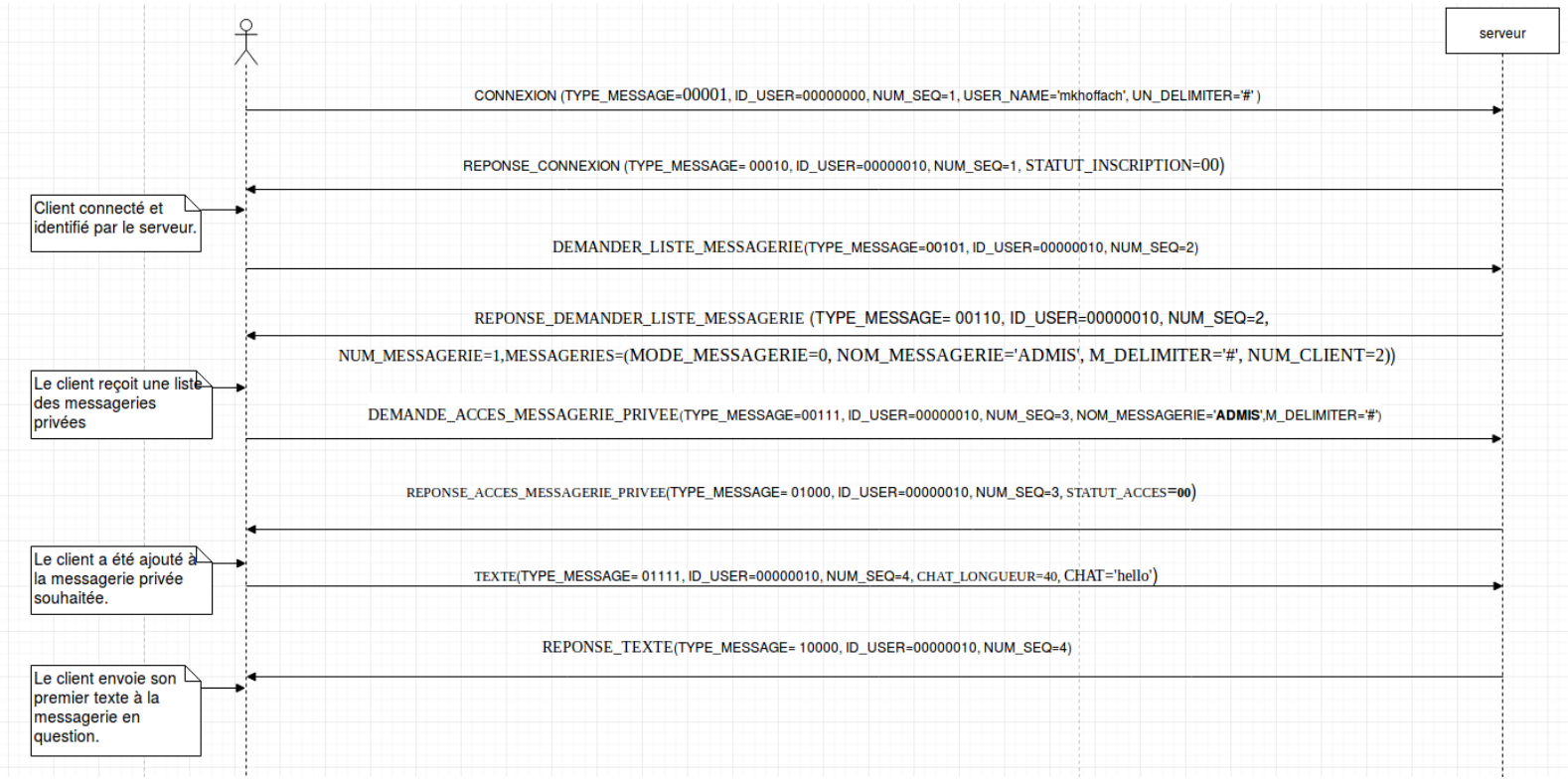
6.2 Scénario 1

Dans ce premier scénario, le client se connecte puis crée sa messagerie à laquelle il invite un autre client.



6.2 Scénario 2

Dans ce deuxième scénario, le client se connecte, demande au serveur de lui fournir la liste des messageries privées, puis demande l'accès à l'une de ces messageries et envoie à la fin un texte aux utilisateurs de cette messagerie.



8. Références

https://fr.wikipedia.org/wiki/User_Datagram_Protocol

https://www.ietf.org/rfc/rfc793.txt

https://www.ietf.org/rfc/rfc768.txt

http://abcdrfc.free.fr/rfc-vf/rfc0768.html

https://tools.ietf.org/html/rfc3828

https://tools.ietf.org/html/rfc768

https://tools.ietf.org/html/rfc7011