RES 302
25 février 2017

Spécification du protocole

Gabriel Jaime Becerra gabriel.becerraeslava@imt-atlantique.net
Florencia Costa f.costaepiscopo@imt-atlantique.net

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

1 Résumé

Ce document explique en détail le fonctionnement d'un service de messagerie instantanée en précisant tous les messages échangés entre les différents acteurs du système et ses fonctionnalités.

Il s'agit d'un protocole de session qui permet la connexion et échanges de messages entre clients, leur participation en chat privés et sa création, ainsi que leur déconnexion. Le système est géré par un serveur dont son rôle est indispensable pour le correct fonctionnement.

2 Introduction

Étant donné que ce protocole fonctionne au-dessus d'UDP et à la fois UDP fonctionne au-dessus d'IP, on sait qu'on a une limite sur la taille du message qui est 65535 bytes. Sans considérer l'entête d'IP (60 bytes max) et UDP (8 bytes) il reste 65467 bytes disponibles pour l'implémentation du protocole.

Pour déterminer la taille maximum du message à envoyer, on doit d'abord établir l'entête du protocole. Cet entête a six champs :

ACK_SEQ		FLAG_CODE		Type de Message	NICKNAME_Source	Taille DATA	DATA
1 bit	7 bits	1 bit	7 bits	1 byte	10 bytes	2 bytes	Variable

- **ACK-SEQ :** cet octet est pour gérer la fiabilité de la liaison. Le premier bit est pour la confirmation des messages reçus et les 7 bits qui restent déterminent un numéro de séquence qui identifie le message.
- **FLAG-CODE :** le premier bit de cet octet s'agit d'un drapeau multi-usages qui dépend du Type de Message, on décrira ses usages dans la description détaillée de messages. Les 7 bits qui restent déterminent un code à interpréter pour le client est qui sera détaillé dans la section 3.B
- **Type de Message :** il identifie le message reçu et par conséquente, les actions à prendre
- **NICKNAME_Source :** il contient le nickname du client qui envoie le message
- **Taille DATA :** il indique le nombre de bytes qui contient le champ DATA.

On appellera pour simplicité FLAG au premier bit (le bit le plus significatif) du champ FLAG_CODE et ACK au premier bit (le bit le plus significatif) du champ ACK_SEQ.

Pour représenter le contenu des champs dans ce document on a utilisé les critères suivants :

Lorsque le contenu d'un champ est X, cela signifie que ne sera pas regardé pour le récepteur mais il doit quand même être rempli par zéros au niveau de l'émetteur. Le numéro de séquence sera représenté par 'seq'. La partie du champ FLAG_CODE qui fait référence au CODE du message sera représenté par 'code', le nickname du client par 'nickname_source' et le numéro qui indique la taille du message dans le champ Taille_DATA sera représenté par 'taille'.

Pour accomplir les différentes fonctions du protocole, on a défini différents types de messages qui seront identifiés dans l'entête par le champ « Type de Message » et qu'on précisera dans la description détaillée, à savoir :

Type de Message	Identificateur
Connexion	A
Message du serveur a client	B
Message chat public	C
Message chat privé	D
Demande de liste	E
Invitation à une messagerie privée	F
Accepter o refuser une invitation	G
Abandonner chat privé	H
Déconnexion	I

Le champ DATA ne fait pas partie de l'entête, il contient l'information à transmettre à la couche supérieure, soit comme un message envoyé entre clients du chat, soit comme information structuré à traiter par le client ou serveur. Ce traitement du champ DATA sera déterminé par les autres champs et sera précisé dans la description détaille du message.

2.1 Nickname

Les caractères possibles pour le nickname sont [a-z][A-Z] [0-9]. La longueur maximum est 10 caractères, ce qui nous semble une limite convenable pour avoir un nombre considérable d'clients.

Les champs destinés pour contenir les NICKAMEs dans l'entête du message ont 10 bytes chacun et seront utilisés du byte plus significatif au moins significatif. Si le NICKNAME a moins de 10 caractères, il faut remplir les bytes pas utilisés avec le caractère NULL.

Example :

Si le nickname est «user123 » on a dans le champ NICKNAME_Source

Byte	0	1	2	3	4	5	6	7	8	9
caractere	u	s	e	r	1	2	3	NULL	NULL	NULL

2.2 Taille du message et clients

On établit que pour ce type d'application, les messages échangés entre les clients ont une taille maximum de 500 caractères ce qui nous semble une limite convenable étant donné qu'il s'agit de la messagerie instantanée. Ce fait se traduit en 500 bytes dans le champ DATA des messages transmis ce que n'est pas un problème puisqu'on a 65467 bytes disponibles. Néanmoins, le contraint principal se trouve dans la quantité de clients du chat, parce que dans notre conception du protocole, dans certains cas on doit envoyer cette liste de forme structuré dans le champ DATA, comme on le précise à continuation :

<Nickname0>‘: ’<status>‘&’<Nickname1>‘: ’<status>‘&’…‘&’<NicknameN>‘: ’<status>

Ces données se trouvent dans une liste de clients qui possèdent le serveur et que lui permet de gérer le système. Cette liste contient les informations suivantes :

- Nickname de client
- Adresse IP
- Port
- Status (public ou Privé)
- Type de chat privé (**C**entralisé ou **D**écentralisé)
- Identificateur du chat
- Numéro de séquence entre 0 et 127

L'élément Type de chat privé est regardé seulement si le status est Privé.

En considérant le pire cas, on a 10 bytes du Nickname, 1 byte pour «:», 1 byte pour « », 2 bytes pour le status (Pu ou Pr) et 1 byte pour ‘&’ qui indique où commence le client suivant. On a alors 15 bytes pour chaque client La quantité de bytes à transmettre est égal à 15*n-1, où n est le nombre de clients. Si l'on établit une limite de 2000 clients, ce qui nous semble une limite convenable par rapport à la portée du projet, on a un maximum de 30.000 bytes dans le champ DATA quand la liste complète de clients est demandé, est on reste toujours dans la limite imposé pour le protocole IP.

2.3 Quantité des chats privés

On vient de définir la quantité de clients pour le chat. Le numéro maximum de chats privés est 1000, et c'est le cas où tous les chats privés ont exactement deux participants.

2.4 Modalité de chat

Un client peut appartenir au Chat Public ou à un Chat Privé. Il existe deux possibilités du chat privé : centralisé et décentralisé. Les clients doivent savoir à tout moment à que type de chat ils appartiennent. Le serveur a aussi cette information pour tous les clients.

2.4.1 Chat public

Tous les clients, depuis sa connexion et jusqu'à accepter une invitation pour participer d'un chat privé, appartiennent par défaut au chat public.

2.4.2 Chat privé Centralisé

Dans ce cas tous les messages seront envoyés d'abord au serveur et après celui-ci le renverra aux autres clients du même chat privé.

2.4.3 Chat privé Décentralisé

Dans ce cas tous les messages seront envoyés directement d'un client à un autre sans passer par le serveur.

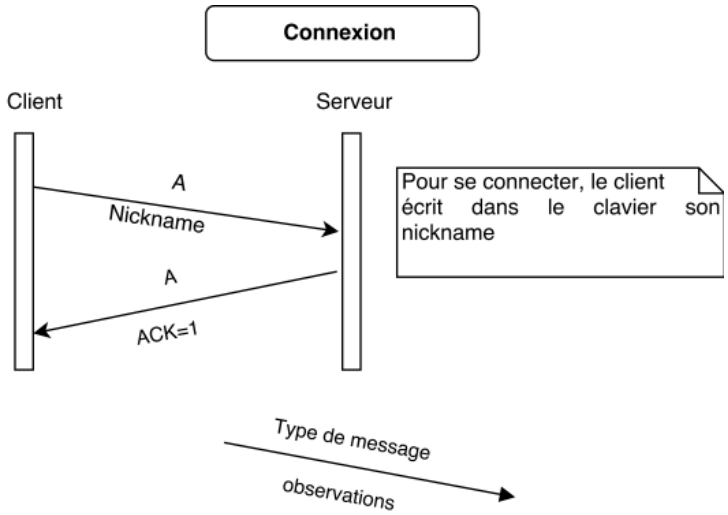
2.5 Fiabilité

Pour le bon fonctionnement de l'application, le protocole doit gérer la retransmission des messages perdus et cela est fait grâce au champ ACK-SEQ et à l'implémentation d'un TIMER. Ce TIMER est initié lorsque le message qui a besoin de confirmation est envoyé. Si le temps est dépassé sans que l'émetteur du message reçoive la confirmation, le message est retransmis. Pour ce protocole on a défini un timer de 500ms c-à-d, chaque fois qu'on attend un message de confirmation, on va l'attendre seulement pour 500ms, sinon on renvoie le message. La nombre maximum de retransmissions pour chaque message est 5.

Dans le champ ACK_SEQ on a destiné 7 bits pour le numéro de séquence. Ce numéro sert à identifier le message à fin de qui aie une concordance entre le message envoie et sa confirmation. Pour implémenter cela, chacun le client et le serveur, doivent avoir un compteur qui sera incrémenté pour chaque nouveau message émis. Le message d'acquittement devra avoir le même numéro de séquence de celui qu'il confirme. Chez le serveur, ce compteur est à s'incrémenter de façon indépendante pour chaque client, donc un champ destiné à ce compteur aura lieu dans la liste de clients qui possède le serveur. (Voir section 2.2)

3 Description détaillée des types de message

A - Connexion

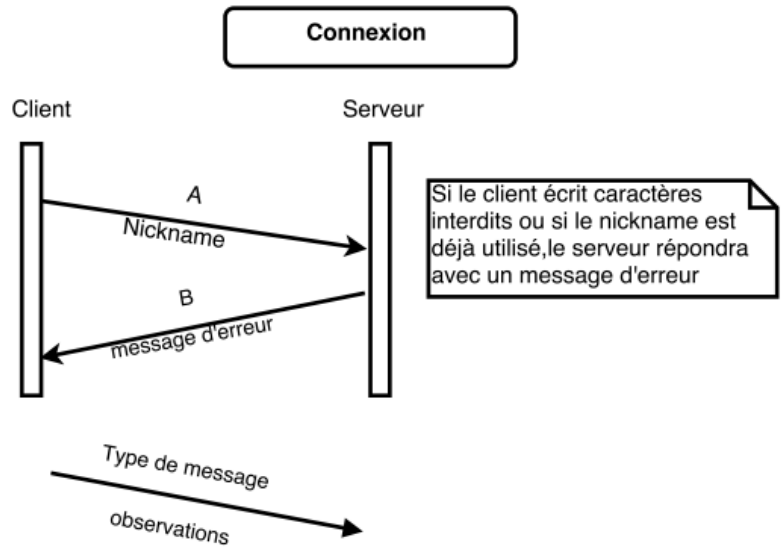


- Côté client:

  Le client envoie ce type du message au serveur quand il désire se connecter au système. Dans le champ NICKNAME_Source il envoie son nickname qui a été récupéré du clavier.

ACK_SEQ		FLAG_CODE		Type de Message	NICKNAME_Source	Taille DATA	DATA
0	seq	X	X	A	nickname_source	0	

Le client reste à l'attente d'une réponse du serveur, soit une acceptation de connexion, soit un message d'erreur.



- Côté serveur

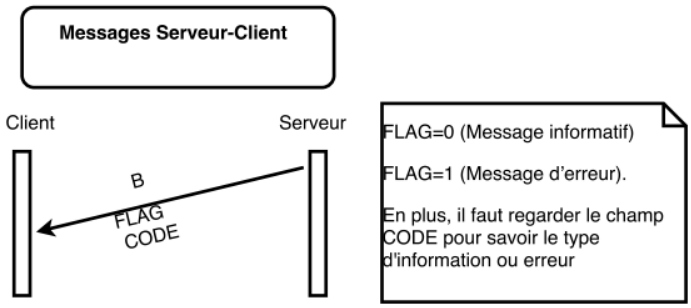
  Lorsqu'un message de ce type provient du client, le serveur seulement s'intéresse par les champs Type de Message et NICKNAME_Source. Si le NICKNAME_Source n'est pas encore utilisé, il répond avec un message de confirmation, avec ACK=1.

ACK_SEQ		FLAG_CODE		Type de Message	NICKNAME_Source	Taille DATA	DATA
1	seq	X	X	A	X	0	

De plus, il crée une nouvelle entrée dans la liste de clients avec le nouveau client qui vient de se connecter.
Si le NICKNAME_Source existe déjà dans la liste des clients, le serveur envoie un message d'erreur qu'on précisera à continuation.

B - Message du serveur au client

Pour ce type de message on a deux possibilités : FLAG=0 (Message informatif) et FLAG=1 (Message d'erreur).



F=0 : Message informatif

CODE	Description
0000001	Chat privé refusé
0000010	Chat privé accepté
0000100	Client a abandonné le chat privé
0001000	Chat privé fermé pour manque de clients

Chat privé refusé : un message informatif de ce type indique au client créateur du chat privé qu'un client a refusé son invitation. Dans ce cas le nickname du client qui refuse l'invitation ira dans le champ NICKNAME_Source.

Chat privé accepté : un message informatif de ce type indique au client créateur du chat privé qu'un client a accepté son invitation. Dans ce cas le nickname du client qui accepte l'invitation ira dans le champ NICKNAME_Source.

Client a abandonné le chat privé : un message informatif de ce type indique aux clients qui font partie du chat privé qu'un client l'a abandonné. Dans ce cas le nickname du client qui abandonne le chat ira dans le champ NICKNAME_Source.

Chat privé fermé pour manque de clients : un message informatif de ce type indique au client du chat privé que le chat sera fermé parce qu'il ne reste que lui dans le chat.

F=1 : Message d'erreur

CODE	Description
0000001	Nickname déjà utilisé
0000010	Caractères interdits
0000100	Client inexistant
0001000	Client déjà dans un chat privé

Nickname déjà utilisé : un message d'erreur de ce type indique au client qui vient de solliciter la connexion que le nickname est déjà utilisé. Alors, s'il veut se connecter il devra utiliser un nickname diffèrent.

Caractères interdits : un message d'erreur de ce type indique au client qui vient de solliciter la connexion que le nickname écrit a au moins un caractère interdit

Client inexistent : un message d'erreur de ce type indique au client qui vient d'envoyer une invitation pour un chat privé que ce nickname n'existe pas

Client déjà dans un chat privé : Un message d'erreur de ce type indique au client qui vient d'envoyer une invitation pour un chat privé que cet client déjà appartient à un autre chat privé.

- Côte client

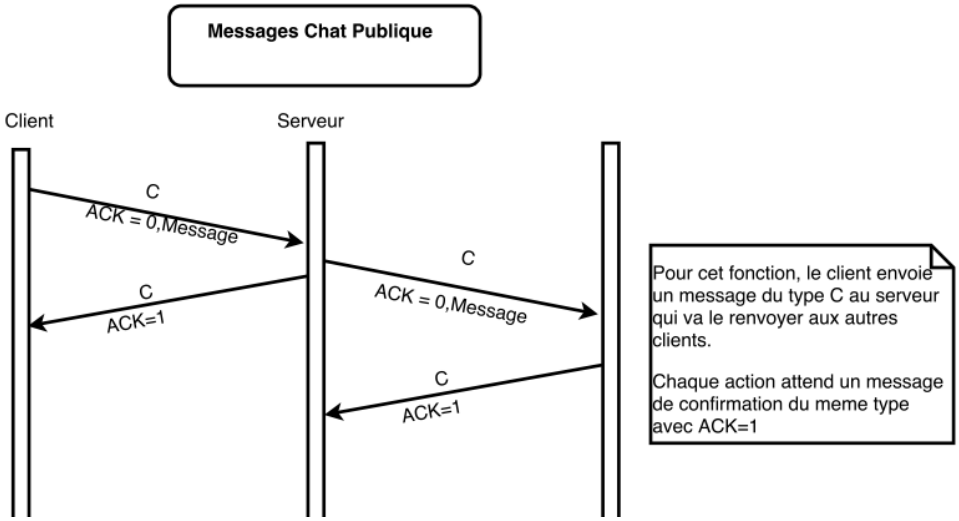
  Lorsque le client reçoit un message de ce type, il doit regarder le FLAG pour savoir s'il s'agit d'un message d'erreur ou d'un message informatif. Ensuite, en regardant le champ CODE il sait lequel message afficher.

- Côté serveur

  Les messages qu'envoie le serveur ont le format suivant :

ACK_SEQ		FLAG_CODE		Type de Message	NICKNAME_Source	Taille DATA	DATA
0	seq	0/1	code	B	X	0	

C - Message chat public



- Côte client émetteur

  Le client envoie un message au serveur, avec l'information qu'il désire transmettre au chat dans le champ DATA. Cette information est récupérée du clavier et sera affiché dans le chat dans lequel le client fait partie.

ACK_SEQ		FLAG_CODE		Type de Message	NICKNAME_Source	Taille DATA	DATA
0	seq	X	X	C	nickname_source	taille	message

Le client attend une confirmation du message du côté de serveur pour assurer qu'il l'a reçu.

- Côté client récepteur

  Lorsque un client reçoit un message de ce type, il affiche le contenu du champ DATA dans le champ et envoie une confirmation au serveur :

ACK_SEQ		FLAG_CODE		Type de Message	NICKNAME_Source	Taille DATA	DATA
1	seq	X	X	C	nickname_source	0	

- Côté serveur

  Lorsque le serveur reçoit un message de ce type, il regarde le champ ACK.

  Si ACK=0 il sait qu'il s'agit d'un message à envoyer au reste des clients du chat du client émetteur. Il l'envoie donc un message de confirmation et à continuation distribue le message aux autres clients.

  Message de confirmation :

ACK_SEQ		FLAG_CODE		Type de Message	NICKNAME_Source	Taille DATA	DATA
1	seq	X	X	C	X	0	

  Message distribué:

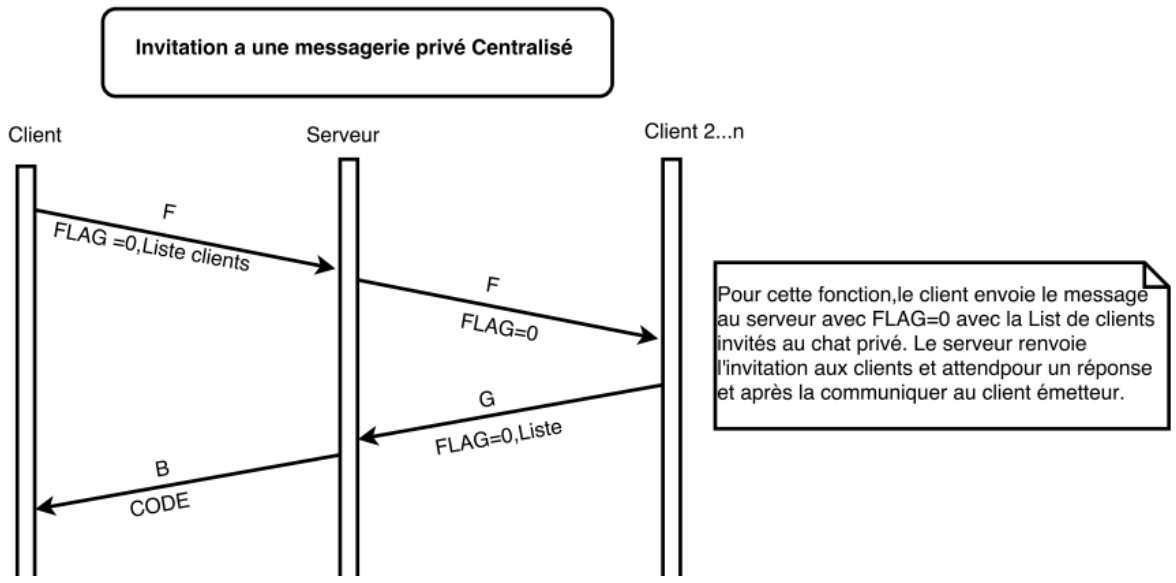
ACK_SEQ		FLAG_CODE		Type de Message	NICKNAME_Source	Taille DATA	DATA
0	seq	X	X	C	nickname_source	taille	message

  Si par contre, le message reçu par le serveur a ACK=1 il sait qu'il s'agit d'un message de confirmation.

D - Message chat privé

Pour ce type de message on a deux possibilités : FLAG=0 (centralisé) et FLAG=1 (décentralisé).

FLAG=0: Chat privé centralisé



- Côte client émetteur
  Le client envoie un message au serveur, avec l'information qu'il désire transmettre au chat privé dans le champ DATA. Cette information est récupérée du clavier et sera affiché dans le chat dans lequel le client fait partie. Le message envoyé a la structure suivante :

ACK_SEQ		FLAG_CODE		Type de Message	NICKNAME_Source	Taille DATA	DATA
0	seq	0	X	D	nickname_source	taille	message

Le client attend une confirmation du message du côté de serveur pour assurer qu'il l'a reçu.

- Côte client récepteur

  Lorsque un client reçoit un message de ce type, il vérifie d'abord que le message est adressé à lui et ensuite il affiche le contenu du champ DATA dans le chat et envoie une confirmation au serveur :

ACK_SEQ		FLAG_CODE		Type de Message	NICKNAME_Source	Taille DATA	DATA
1	seq	0	X	D	nickname_source	0	

Si le message reçu a un Nickname_destin différent au client récepteur (ce qui peut arriver si deux clients ont le même port et IP) il ne fait rien avec le message.

- Côté serveur

  Lorsque le serveur reçoit un message de ce type, il regarde le champ ACK.

  Si ACK=0 il sait qu'il s'agit d'un message à envoyer au reste des clients du chat du client émetteur. Il l'envoie donc un message de confirmation et à continuation distribue le message aux autres clients.

  Message de confirmation :

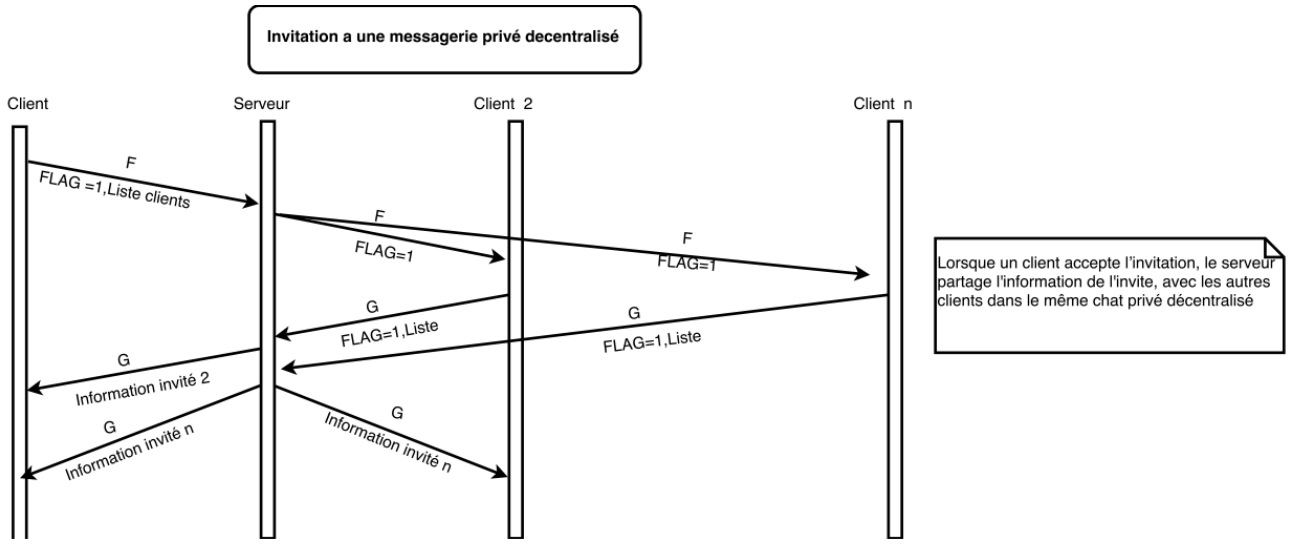
ACK_SEQ		FLAG_CODE		Type de Message	NICKNAME_Source	Taille DATA	DATA
1	seq	0	X	D	X	0	

Message distribué :

ACK_SEQ		FLAG_CODE		Type de Message	NICKNAME_Source	Taille DATA	DATA
0	seq	0	X	D	nickname_emmeteur	taille	message

Si par contre, le message reçu par le serveur a ACK=1 il sait qu'il s'agit d'un message de confirmation.

FLAG=1 : Chat privé décentralisé



- Côte client émetteur

  Lorsque le client désire envoyer un message dans un chat privé décentralisé, il l'adresse lui-même à chaque client du chat privé, dont ses informations aura disponible dans une liste dés moment qu'il intègre le chat. Chaque message envoyé a la structure suivante :

ACK_SEQ		FLAG_CODE		Type de Message	NICKNAME_Source	Taille DATA	DATA
0	seq	1	X	D	nickname_source	taille	message

  Le client attend une confirmation du message du côté de chaque client pour s'assurer qu'il l'a reçu.

- Côte client récepteur

  Lorsque un client reçoit un message de ce type, il vérifie d'abord que le message est adressé à lui et ensuite il affiche le contenu du champ DATA dans le chat et envoie une confirmation au serveur :

ACK_SEQ		FLAG_CODE		Type de Message	NICKNAME_Source	Taille DATA	DATA
1	seq	1	X	D	nickname_source	0	

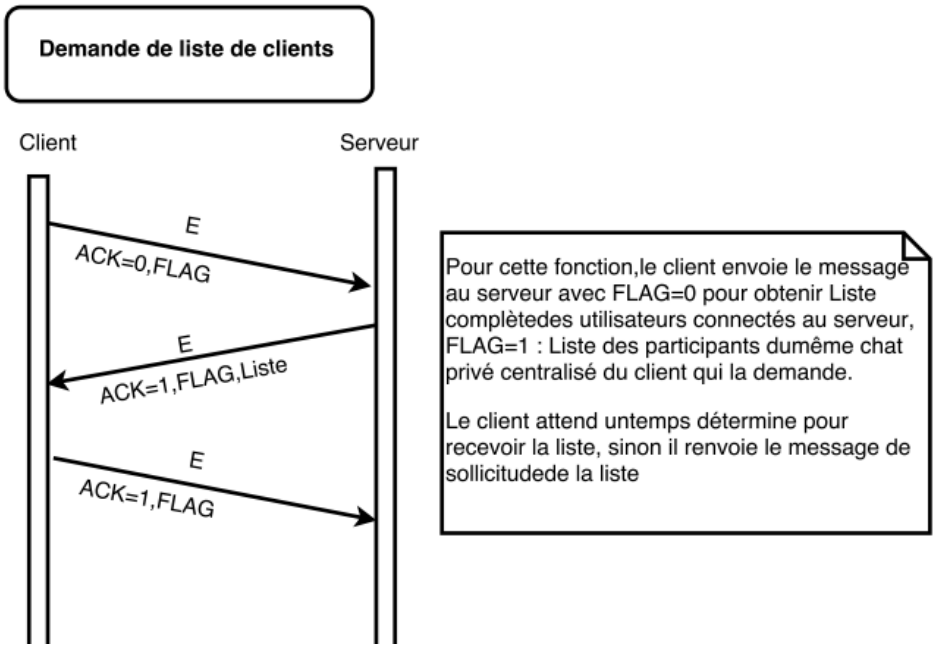
- Côté serveur

  Le serveur reste inactif pendant l'échange des messages dans un chat privé décentralisé.

E - Demande de liste

Dans ce cas, le champ FLAG de l'entête indique si la demande est de la liste complète des clients ou s'il s'agit seulement de la liste des clients du chat privé dont le client fait partie.

FLAG=0 : Liste complète des clients connectés au serveur.
FLAG=1 : Liste des clients du même chat privé du client qui la demande.



- Côté client

  Le client envoie ce type du message au serveur pour demander une liste de clients.

  Message de demande :

ACK_SEQ		FLAG_CODE		Type de Message	NICKNAME_Source	Taille DATA	DATA
0	seq	0/1	X	E	nickname_source	0	

  Le client reste à l'attente d'un message du serveur où il espère trouver la liste demandée dans le champ DATA. Une fois ce message est reçu il doit envoyer un message de confirmation. Au contrario, il doit renvoyer la demande de liste.

  Message de confirmation :

ACK_SEQ		FLAG_CODE		Type de Message	NICKNAME_Source	Taille DATA	DATA
1	seq	X	X	E	nickname_source	0	

- Côté serveur

  Lorsque le serveur reçoit un message de type E, il répond avec une liste dans le champ DATA qui a la structure suivante :

  <Nickname0>‘: ’<status>‘&’<Nickname1>‘: ’<status>‘&’…‘&’<NicknameN>‘: ’<status>

  De cette façon le client peut interpréter la DATA et savoir grâce au caractère « & » où commence un nouvel client dans la liste et l’afficher correctement.

  Le message envoyé par le serveur a la structure suivante :

ACK_SEQ		FLAG_CODE		Type de Message	NICKNAME_Source	Taille DATA	DATA
1	seq	0/1	X	E	X	taille	liste

F - Invitation à une messagerie privée

On a deux possibilités pour le champ FLAG : FLAG=0 (centralisé) et FLAG=1 (décentralisé).

- Côté client émetteur

  Lorsqu’un client veut commencer un chat privé, il envoie un message de type F au serveur avec la liste des clients qu’il désire inviter dans le champ DATA, et en précisant dans le champ FLAG le type de chat privé qu’il veut commencer.

ACK_SEQ		FLAG_CODE		Type de Message	NICKNAME_Source	Taille DATA	DATA
0	seq	0/1	X	F	nickname_source	taille	liste

Pour ce cas, la liste à envoyer avec les nicknames aura le format suivante :

<Nickname0> ‘&’<Nickname1> ‘&’…‘&’<NicknameN>

Le chat commence lorsque le premier client accepte l’invitation.

- Côté client récepteur

  Lorsqu’un client reçoit un message de ce type, il peut savoir s’il s’agit d’une invitation à un chat privé centralisé ou décentralisé en regardant le champ FLAG. Il peut aussi savoir qui l’a invité au chat en regardant le champ NICKNAME_Source. Il répondra au serveur avec un message de type G pour indiquer s’il accepte ou refuse l’invitation.

- Côté serveur

  Lorsque le serveur reçoit ce type de message il identifie les clients dans le champ DATA par ses nicknames et le type de chat privé dans le champ FLAG. Il envoie ensuite un message d'invitation du type F aux clients en précisant le créateur par son nickname dans le champ NICKNAME_Source et le type de chat privé dans le champ FLAG.

ACK_SEQ		FLAG_CODE		Type de Message	NICKNAME_Source	Taille DATA	DATA
0	seq	0/1	X	F	nickname_source	0	

Le serveur attend une réponse du type G de côté des clients, qui indique s'ils acceptent ou refusent l'invitation. Il prendra différentes actions selon le contenu du champ FLAG. Le chat privé est créé par le serveur quand au moins une des invitations est acceptée. Dans ce moment, le serveur change l'information du client créateur en assignant un identificateur de chat et en modifiant son status. De plus, le serveur modifie l'information du client invité dans sa liste, en changeant son status et son identificateur de chat.

FLAG=0 (Invitation au chat privé centralisé) :

Lorsque le serveur reçoit la réponse des clients invités, il change ou pas leur status ainsi que leur identificateur du chat dans sa liste d'clients, pour ensuite transmettre un message du type B (avec le CODE correspondant) pour informer au client créateur du chat privé la réponse à son invitation.

FLAG=1 (Invitation au chat privé décentralisé) :

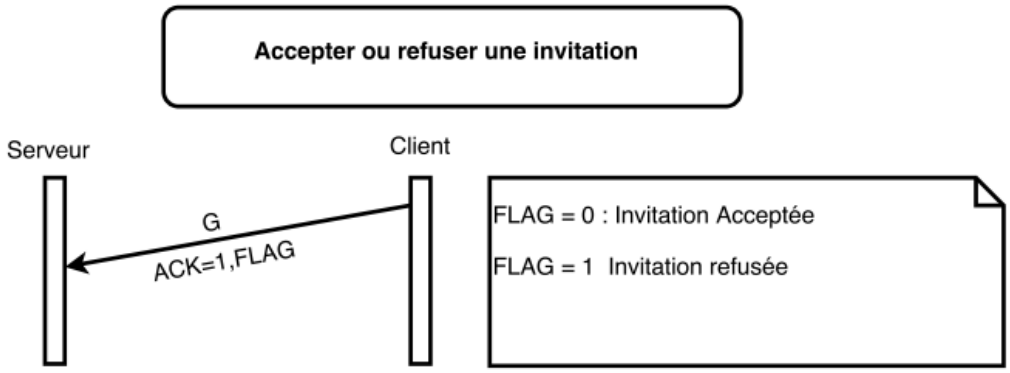
Lorsque le premier client accepte l'invitation, le serveur transmet un message du type G au client créateur du chat privé. Ce message a dans le champ DATA toute l'information du client invité pour qu'il soit possible la communication décentralisée, c'est-à-dire : nickname, port, IP. Si le client refuse l'invitation, le serveur envoie un message du type B au client créateur pour qu'il le sache. (Voir tableau 2).

Lorsqu'un nouveau client accepte l'invitation, le serveur transmet son information aux autres clients qui font partie du même chat privé et en même temps, il reçoit du côté du serveur l'information des autres membres du chat et de cette façon tous les membres peuvent se communiquer entre eux sans avoir besoin du serveur.

G - Accepter o refuser une invitation

Le champ FLAG est utilisé pour accepter ou refuser l'invitation :

FLAG=0 Invitation acceptée
FLAG=1 Invitation refusée



- Côté client

  Lorsqu'un client reçoit un message de type F d'invitation au chat privé, il répondra au serveur avec un message de type G, qui a la structure suivante :

ACK_SEQ		FLAG_CODE		Type de Message	NICKNAME_Source	Taille DATA	DATA
1	seq	0/1	X	G	nickname_source	0	

- Côté serveur

  Si le serveur se trouve dans l'attente d'une réponse à une invitation au chat privé centralisé, lorsque le message arrive, il regarde le champ FLAG. Il envoie ensuite un message du type B informatif au client créateur. La mise à jour du status des clients (changement de public à privé) aussi comme l'identificateur du chat privé dans la liste sont faites dans ce moment.

  Si le serveur se trouve dans l'attente d'une réponse à une invitation au chat privé décentralisé, lorsque le message arrive, il regarde le champ FLAG. Si FLAG=0, l'invitation a été acceptée et alors il envoie un message du type G au client créateur qui a toute l'information du client invité pour qu'il soit possible la communication décentralisée, c'est-à-dire : nickname, port, IP.

  Le champ DATA aura le format suivante :

  <Port>'&'<IP>

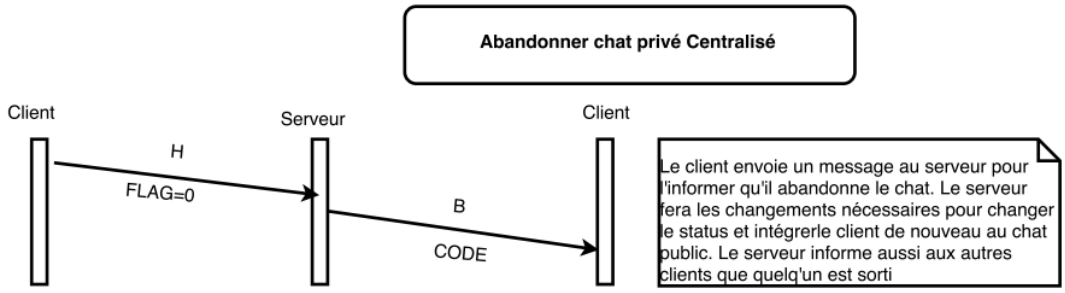
ACK_SEQ		FLAG_CODE		Type de Message	NICKNAME_Source	Taille DATA	DATA
0	seq	0	X	G	nickname_source	taille	port, IP

H - Abandonner chat privé

Le champ FLAG peut prendre la valeur 0 (chat privé centralisé) ou 1 (chat privé décentralisé)

- Côté client

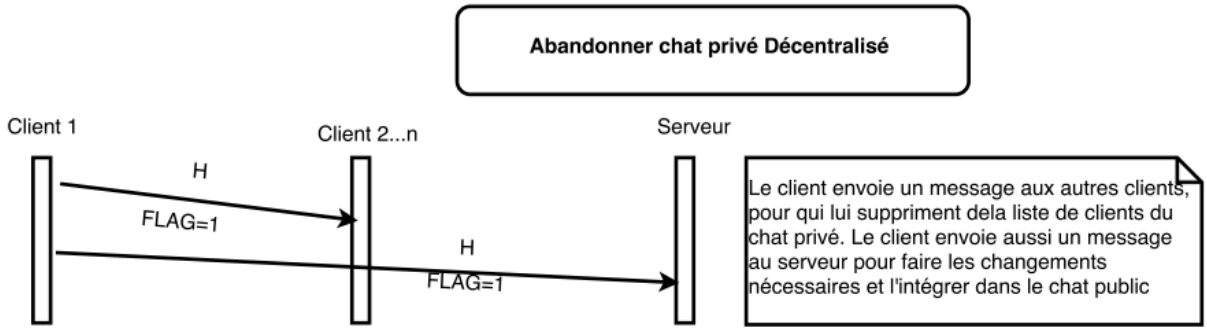
  Cas centralisé



  Lorsqu'un client désire abandonner le chat privé, il envoie un message de ce type au serveur qui a la structure suivante :

ACK_SEQ		FLAG_CODE		Type de Message	NICKNAME_Source	Taille DATA	DATA
0	seq	0	X	H	nickname_source	0	

  Cas décentralisé



  Lorsqu'un client désire abandonner un chat privé, il envoie un message de ce type au reste des clients en précisant son nickname dans le champ NICKNAME_Source et le nickname de chacun des autres clients dans le champ NICKNAME_Destin. Il envoie aussi un message au serveur. Ces messages ont la structure suivante :

ACK_SEQ		FLAG_CODE		Type de Message	NICKNAME_Source	Taille DATA	DATA
0	seq	1	X	H	nickname_source	0	

  Lorsque les clients reçoivent le message, ils doivent supprimer le client qui abandonne le chat de la liste de clients.

- Côté serveur

  Cas centralisé :

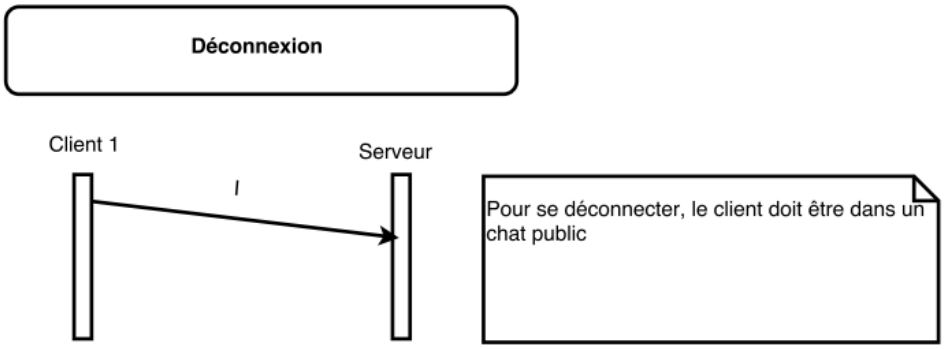
  Lorsque le serveur reçoit un message de ce type, il doit changer l'information du client qui abandonne le chat en changeant son status et son identificateur du chat et il envoie un message informatif du type B pour mettre au courant le reste des clients du même chat.

  Cas décentralisé :

  Lorsque le serveur reçoit un message de ce type, il doit seulement changer l'information du client dans sa liste de clients, et vérifier qu'il reste encore au moins deux clients du chat ` privé.

  Le serveur doit dans le deux cas, vérifier qu'il reste encore au moins deux clients dans le chat privé. Si celui n'est pas le cas, le serveur doit supprimer le chat privé et changer l'information du dernier client, qui retournera au chat public après avoir été informé pour un message de type B du serveur.

I - Déconnexion



- Côté client

  Pour pouvoir se déconnecter du système, un client doit toujours être dans le chat public. Lorsqu'un client désire se déconnecter du système, il envoie un message qui a la structure suivante :

ACK_SEQ		FLAG_CODE		Type de Message	NICKNAME_Source	Taille DATA	DATA
0	seq	X	X	I	nickname_source	0	

- Côté serveur

  Lorsqu'un message de ce type est reçu par le serveur il doit supprimer le client de la liste des clients.