MEDJEBER Safir
GEDIK Marc

Spécification et test de l'application Skaïpeuh

Chapitre 1 :

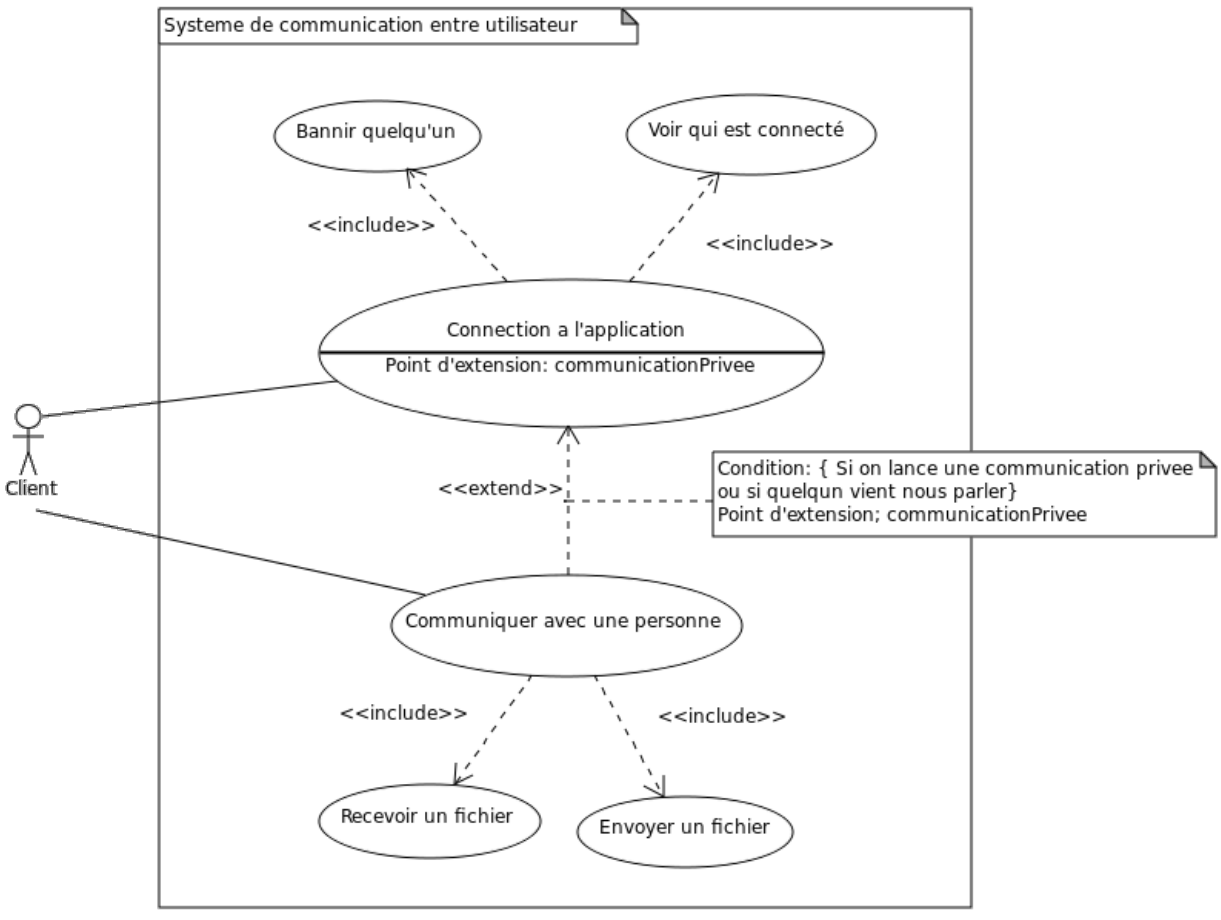
1)

L'application Skaipeuh consistera en un système de communication entre utilisateurs. Ainsi elle devra permettre à un client de se connecter à un groupe d'utilisateurs sur le réseau.

Une fois connecté, le client pourra avoir accès à une liste des personnes présentes sur ce même réseau. Cette liste sera mise à jour en temps réel et comprendra l'ensemble des personnes avec qui le client pourra échanger des messages.

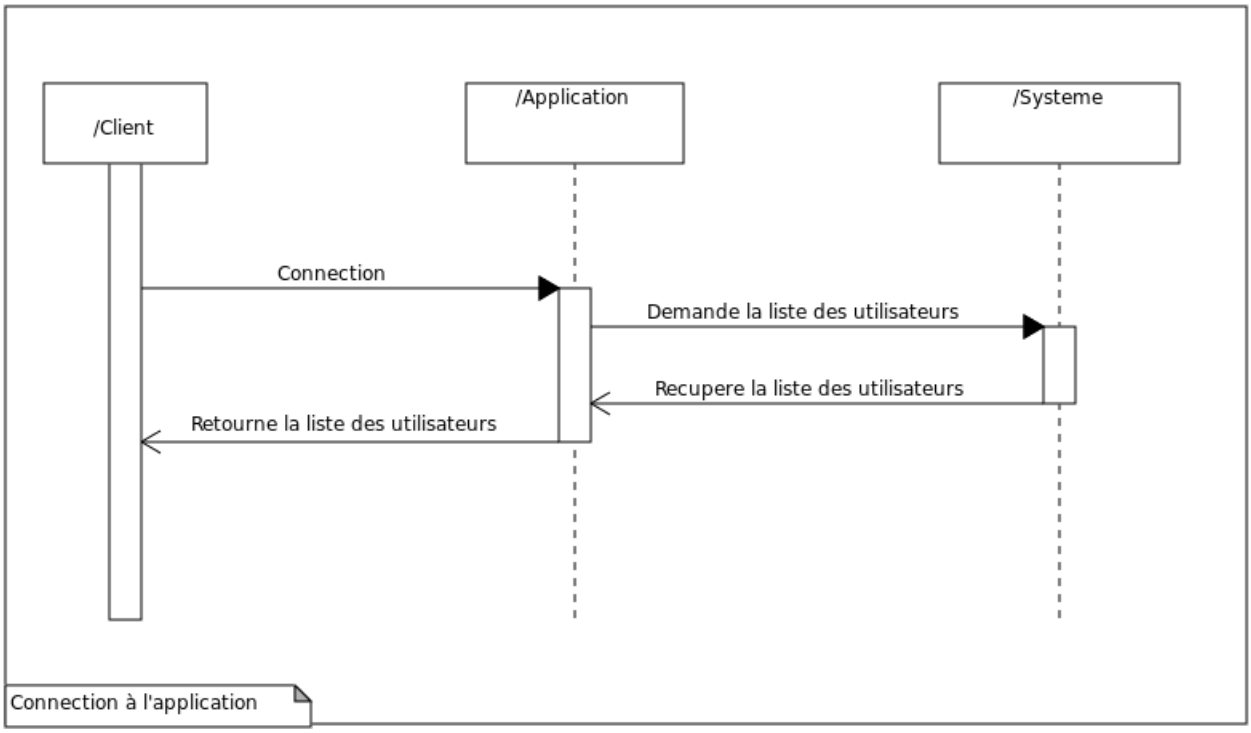
Ces échanges s'effectueront avec une personne à la fois et offriront aux deux parties la possibilité d'envoyer un fichier si le destinataire l'accepte. De plus, ils pourront chacun mettre un terme à la conversation lorsqu'ils le souhaitent et ainsi être à nouveau disponible sur le réseau pour entamer une nouvelle discussion.

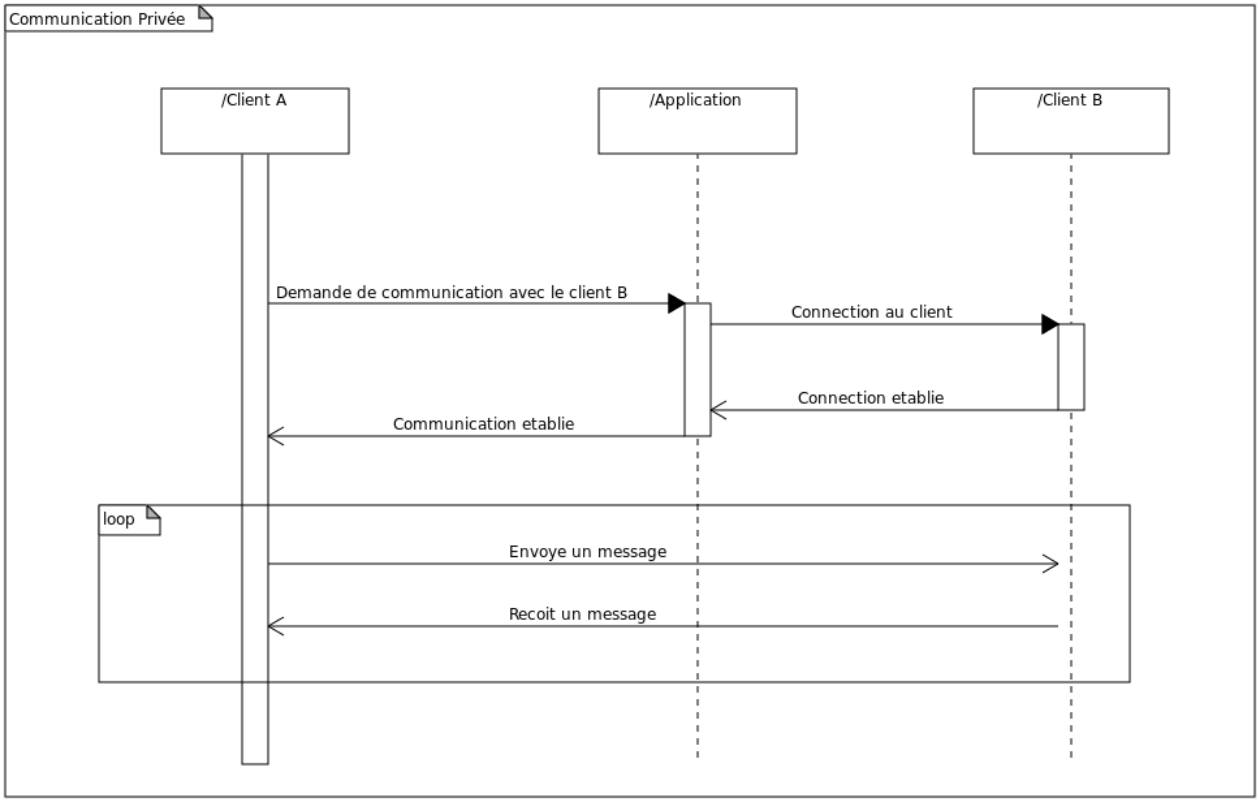
Enfin, une personne connectée pourra être bannie si deux autres membres du groupe l'ordonnent .

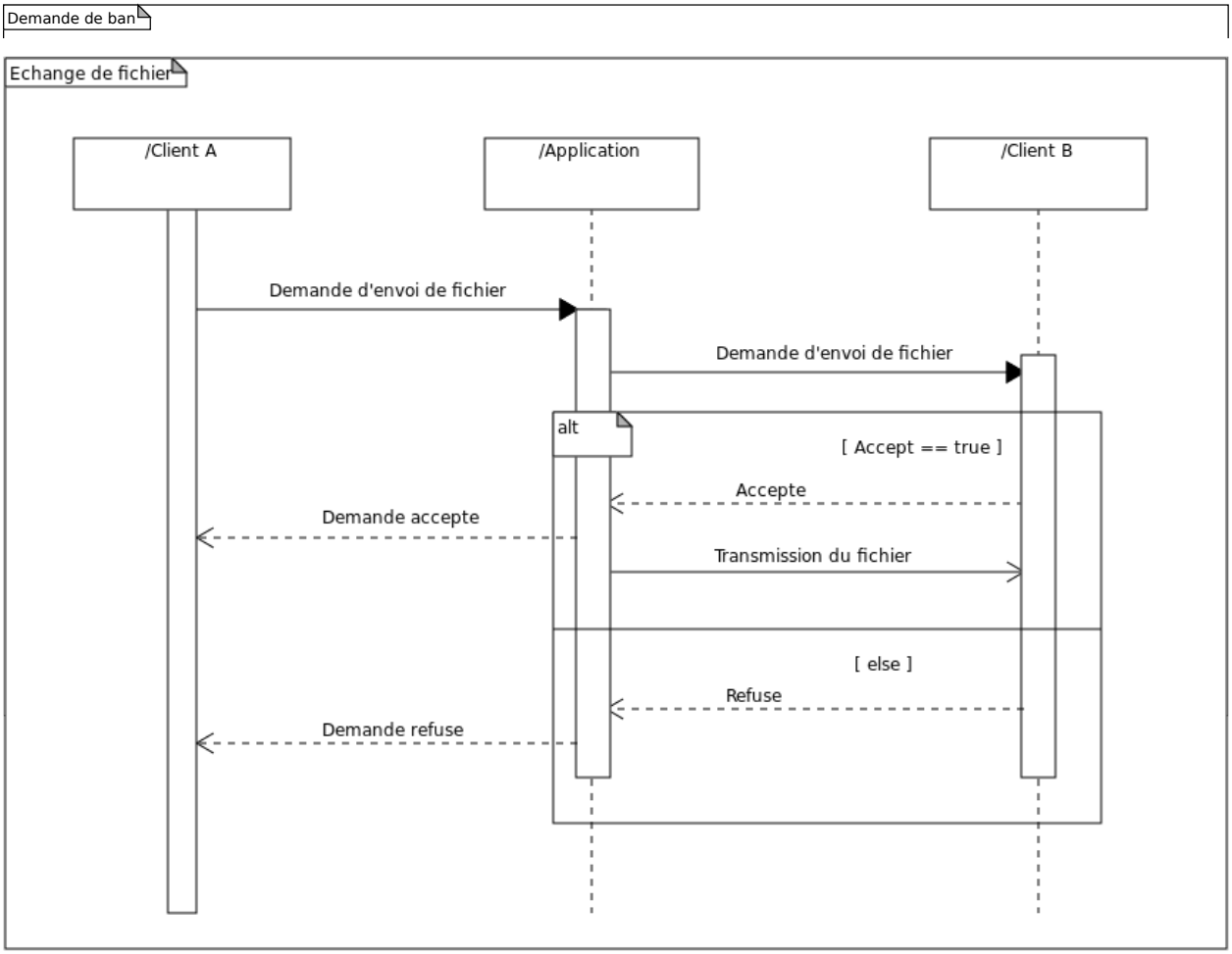


2.a)

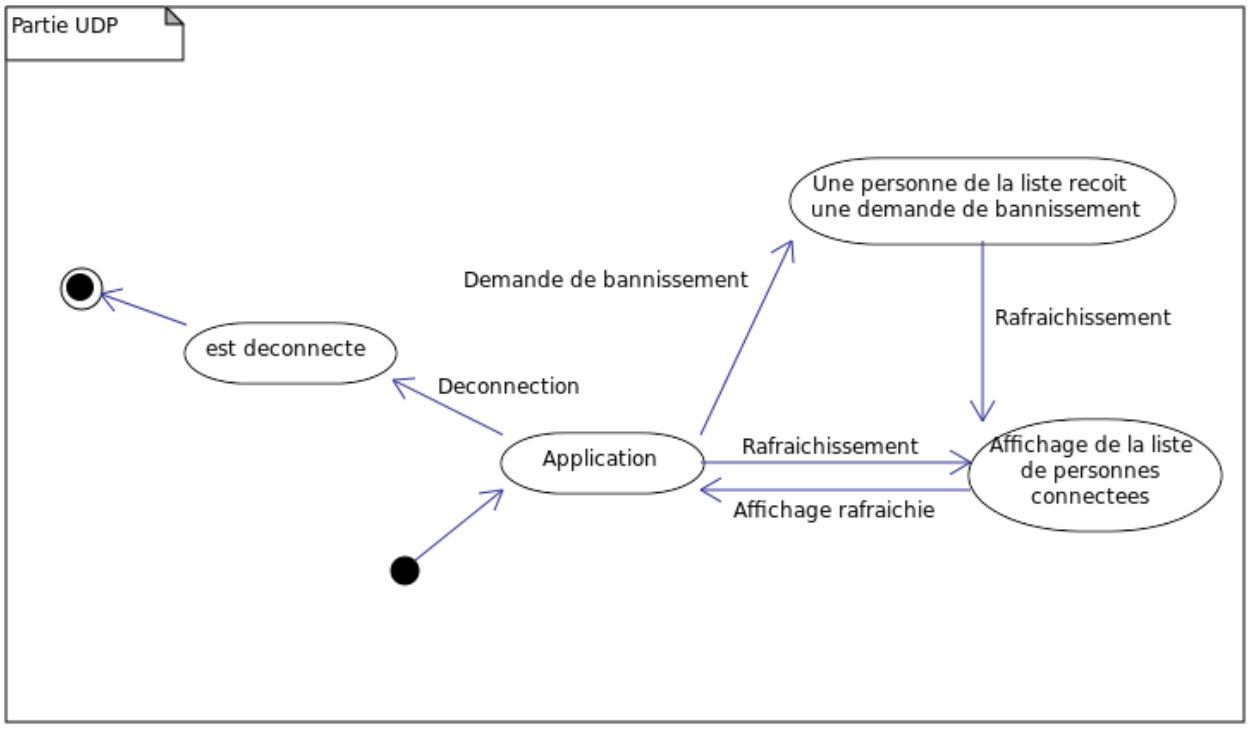
Diagrammes de séquences

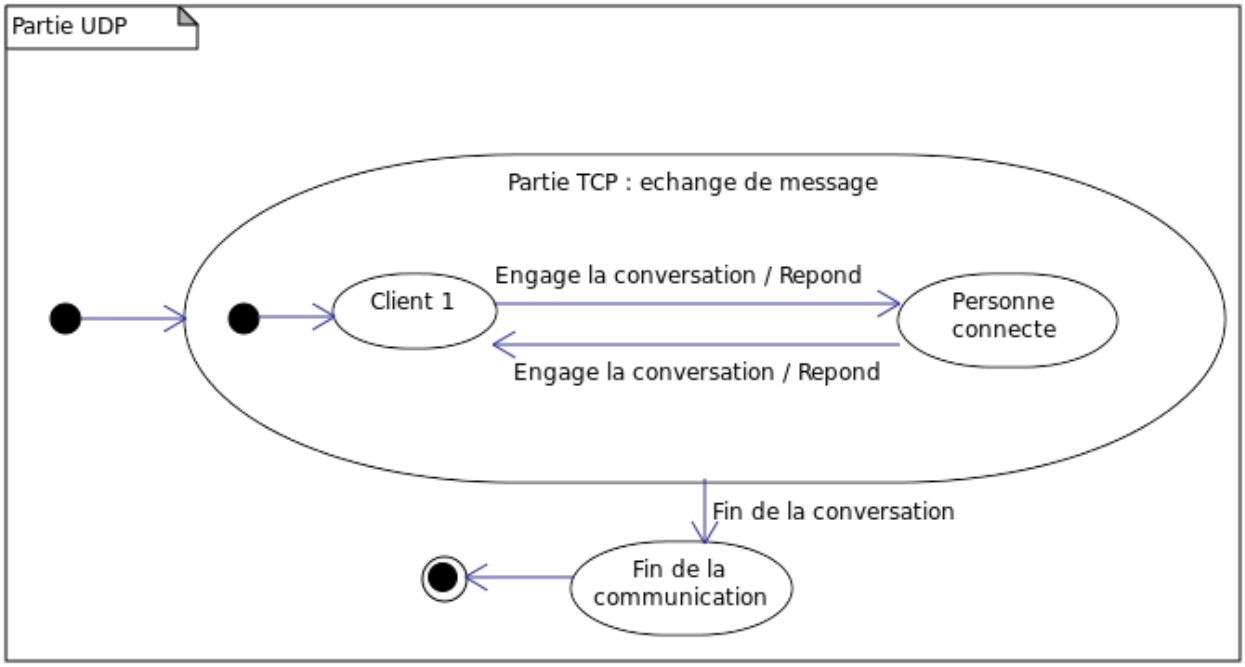


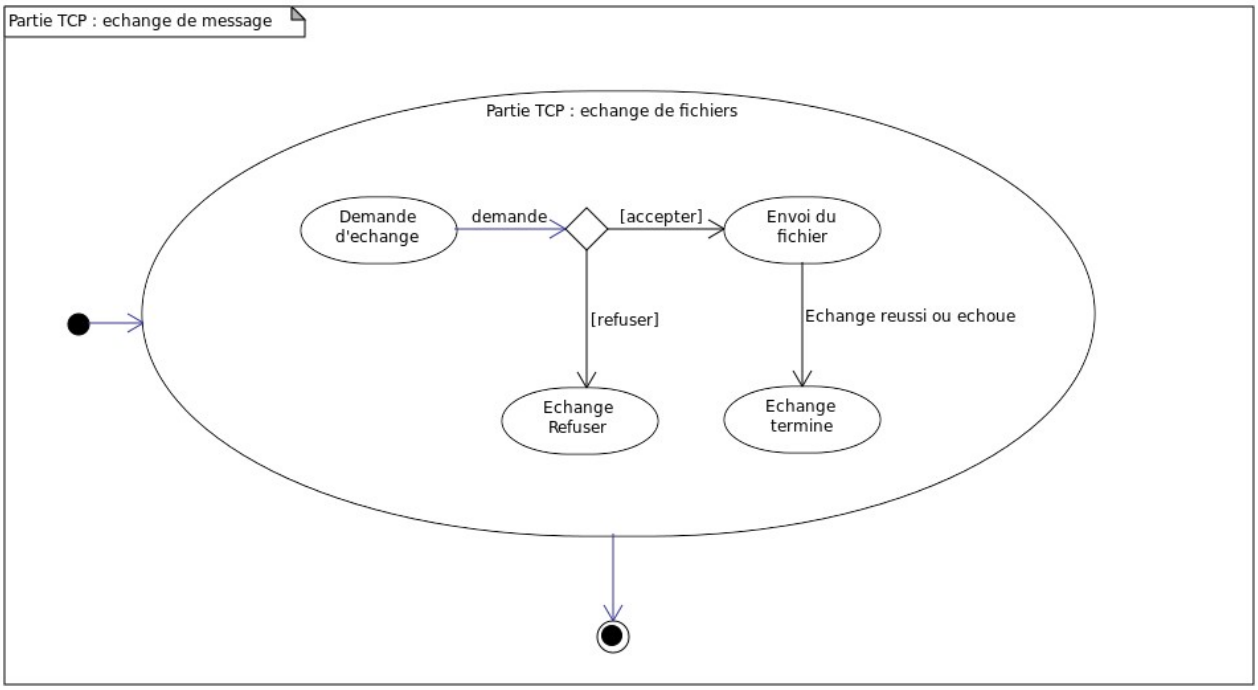

Communication Privée
/Client A
/Application
/Client B
Demande de communication avec le client B
Connection au client
Connection etablie
Communication etablie
loop
Envoye un message
Recoit un message


Demande de ban
Echange de fichier
/Client A
/Application
/Client B
Demande d'envoi de fichier
Demande d'envoi de fichier
alt
[Accept == true]
Accepte
Demande accepte
Transmission du fichier
[else]
Refuse
Demande refuse

Diagrammes états transitions







b)

Afin de pouvoir utiliser l'application, le client est contraint de choisir un pseudo dont la longueur est comprise entre **1 et 8 caractères.** Il doit également affecter à son serveur un numéro de port libre (**entre 49152 et 65535**) sur lequel il pourra être joint.

Un client ne peut à lui seul bannir un autre utilisateur. En outre, il ne peut communiquer qu'avec une personne à la fois. Enfin , il ne peut pas refuser une demande de communication privée.

Pour se déconnecter, le client doit impérativement envoyer un message du type **BYE**

c)

L'application sera implémentée en langage C ou JAVA et devra permettre l'interopérabilité. Elle se divisera en **deux parties** :

1. **1.** la phase d'entrée dans le système qui utilisera le protocole UDP
2. **2.** les communications privées qui utiliseront le protocole TCP

Protocole partie 1

L'entrée dans le système consistera à envoyer un message de multidiffusion au groupe d'utilisateurs dont l'IP est **224.5.6.7** sur le port **9876**. Ce message sera de la forme « **HLO user machine port** »* et permettra d'avertir les membres du groupe de son arrivée.

À réception d'un message de type **HLO**, chaque application présente dans le système répond par un message « **IAM user machine port »* .** Ainsi chaque client sera prévenu de l'existence d'applications déjà présentes.

Lorsqu'un utilisateur souhaite se déconnecter du système, son application devra envoyer un message multi-diffusé de la forme « **BYE user** »*

Pour mettre à jour sa liste de personnes connectées, une application peut à tout moment via un message multi-diffusé du type **RFH** demander aux autres applications de renvoyer un message du type **IAM.**

Enfin , un message multi-diffusé du type **« BAN user »*** forcera l'application concernée à retourner un message du type **BYE** et donc signaler sa déconnection .

Protocole partie 2

Les communications privées s'effectueront via le protocole TCP sous forme de messages de la forme « **MSG length seqchars ».** L'échange débutera dés lors où la connexion entre le client de l'application et le serveur du destinataire est établie.

L'échange prend fin lorsque l'une des deux parties envoie un message **CLO** sur le canal de communication.

Durant la communication l'une des deux parties peut souhaiter envoyer un fichier à l'autre via un message de la forme « **FIL filelength filename port »*** **.** Si le receveur accepte l'échange, il renvoie un simple message **ACK** sinon **NAK.**

La réception du fichier doit s'effectuer **en parallèle de la discussion engagée**. Le fichier reçu doit être enregistrer dans le répertoire **dowloads** du répertoire privé de l'utilisateur.

*confère annexe à la fin du document

Chapitre 2 :

Diagramme de classe

Chapitre 3 :

Dans le cadre de notre projet, nous avons implémenté la totalité des fonctionnalités de l'application en langage C ainsi qu'en JAVA. En voici donc la liste :

- Interopérabilité C et JAVA
- Connexion au groupe de multidiffusion
- Mise à jour de la liste des personnes connectées
- Possibilité d'engager une communication avec les personnes de cette liste
- Possibilité de recevoir des demandes de communication
- Possibilité de bannir une personne connectée ou d'être banni
- Possibilité d'envoyer des fichiers si le destinataire est d'accord
- Possibilité d'accepter ou refuser la réception de fichier
- Quitter une conversation « proprement » et pouvoir en commencer une nouvelle directement
- Quitter l'application « proprement »

Chapitre 4 :

Nous avons effectué des test unitaire sur quelqu’unes des fonctions en JAVA :

Dans la class Bourrage
- Le bourrage du numéro de port, le bourrage de l'adresse ainsi que le bourrage d'un mot par un motif quelconque, retourne le résultat attendu.

Dans la class TabUser
- L'ajout et la suppression d'un utilisateur dans le tableau s'effectue correctement.

Dans la class ReceveFile
- La fonction de renommage d'un fichier existant, fonctionne de la manière que nous avons souhaité.

Annexe :

user : une suite de 8 caractères ascii. Si le nom de l'utilisateur est de longueur inférieure à 8, le caractère ascii 32 sera employé comme caractère de bourrage. Ainsi pour l'utilisateur « yunes » la suite yunes␣␣␣ devra être employée.

machine : représente le codage de l'adresse IPv4 d'une machine. Ce codage est sur 15 caractères : 4 groupes de trois chiffres décimaux séparés par le caractère . (point). Le caractère 0 est employé comme caractère de bourrage. Ainsi l'adresse IPv4 192.168.1.0 sera représentée par la suite de caractères 192.168.001.000.

port : représente le codage d'un numéro d'un numéro de port. Ce codage est sur 5 caractères décimaux avec 0 comme caractère de bourrage. Ainsi le numéro de port 345 sera représenté par la suite de caractères 00345.

length : représente une longueur comprise entre 1 et 500. La valeur est codée sur trois chiffres en utilisant le caractère 0 comme bourrage*. Ainsi la longueur 23 sera codée par la suite de caractères 023.

seqchars : est une séquence de caractères. Celle-ci n'a pas de longueur en tant que telle, si une longueur doit être précisée le protocole fournit cette information par ailleurs. Les caractères sont de simples caractères ascii 8-bits de l'encodage latin-1. Attention : il n'y a pas de caractère de fin ou quelque chose de cet ordre.

filelength : représente une longueur (éventuellement nulle). La valeur est codée sur autant de chiffres décimaux que voulu. La longueur 234876 pourra être codée par la suite de caractères 234876, mais pourra très bien l'être aussi par la suite 0000000234876.

filename : représentera un nom de fichier encodé par une suite de 40 caractères ; suite uniquement constituée de caractères alphabétiques non accentuées (majuscules ou minuscules), chiffres et . (point) et rien d'autre (pas d'espaces, souligné, tiret, etc). Le caractère ␣ devra être employé comme caractère de bourrage. Ainsi le fichier «superfichier.txt» sera encodé sous la forme

superfichier.txt␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣.

*confère annexe à la fin du document