

UNIVERSITÉ NATIONALE DU VIETNAM À HANOÏ
INSTITUT FRANCOPHONE INTERNATIONAL



Option : Systèmes Intelligents et Multimédia (SIM)

Promotion : XXI

Conception et architecture des réseaux (CAR)

Projet de programmation réseau #1

Projet : Messagerie instantanée interactive sur le réseau local
Rapport final

DIALLO Azise Oumar

KAFANDO Rodrique

KOUADIO Kouamé Olivier

Encadrant :

Pr NGUYEN Quang, Enseignant-chercheur (IFI)

nguyen.hong.quang@ifi.edu.vn

Année académique 2016-2017

Table des matières

| | | |
|----------|--|-----------|
| 1 | Introduction générale | 2 |
| 2 | Cahier de charges | 2 |
| 2.1 | Présentation du projet | 2 |
| 2.2 | Les Objectifs visés par ce travail | 2 |
| 2.3 | Les sources d’inspirations | 3 |
| 2.4 | Les acteurs du projet | 3 |
| 2.5 | Les exigences de la messagerie instantanée | 3 |
| 2.5.1 | Les exigences fonctionnelles | 3 |
| 2.5.2 | Les exigences non fonctionnelles | 3 |
| 3 | Conception | 6 |
| 3.1 | Protocoles | 6 |
| 3.1.1 | Protocole de communication | 6 |
| 3.1.2 | Protocole de transport | 7 |
| 3.2 | Architecture du système | 8 |
| 3.3 | Quelques diagrammes de séquence | 8 |
| 4 | Implémentation | 9 |
| 4.1 | Environnement matériel | 9 |
| 4.2 | Environnement logiciel | 9 |
| 4.2.1 | Langage de programmation | 9 |
| 4.2.2 | Les plateformes et logiciels utilisés | 12 |
| 5 | Expérimentations et analyse des résultats | 12 |
| 5.1 | Fonctionnement du programme | 12 |
| 5.2 | Compilation | 12 |
| 5.3 | Exécution | 13 |
| 5.4 | Tests des fonctionnalités de l’application | 13 |
| 5.4.1 | Connexion au chat | 13 |
| 5.4.2 | Affichage de la liste des utilisateurs connectés | 13 |
| 5.4.3 | Chat groupé | 14 |
| 5.4.4 | Chat avec un seul utilisateur | 14 |
| 5.5 | Évaluation de performance du programme | 14 |
| 6 | Conclusion générale | 18 |

1 Introduction générale

De nos jours, l'information est la base (la clé) de tout développement que ce soit individuel ou collectif. Être bien informé, c'est avoir un coup d'avance sur les autres. Le développement des nouvelles technologies de l'information et de la communication (NTIC) a favorisé la propagation, la diffusion et le partage de l'information entre les personnes. En effet, la principale source d'information aujourd'hui à travers le monde est le Web grâce notamment à Internet. L'une des applications les plus utilisées est sans aucun doute la messagerie instantanée. Ainsi, que soit sur Internet ou dans le réseau local d'entreprise, la messagerie instantanée demeure un véritable outil de communication privilégié car permettant des échanges interactifs presque à temps réel entre les utilisateurs [1].

Le fonctionnement d'un tel système repose principalement sur les protocoles de communication mis en place. Ainsi, chaque fournisseur de messagerie instantanée utilise soit son propre protocole (propriétaires) tels que Yahoo Messenger, Windows Live Messenger, soit des protocoles ouverts tels que Google Talk, Facebook Messenger. Comment la conception et l'implémentation d'un tel protocole se font ? C'est dans cette dynamique que s'inscrit notre projet de programmation réseau du module Conception et architecture des réseaux informatiques (CAR).

La suite de ce présent rapport consistera à présenter dans un premier temps une description du projet à travers notamment un cahier de charge. Ensuite, nous faisons la conception de notre protocole de communication. Après la phase de conception, nous faisons la mise en oeuvre du protocole obtenu dans le logiciel de messagerie instantanée. Par la suite, nous présentons quelques cas d'expérimentations. Enfin, nous terminons par les difficultés rencontrées et les perspectives.

2 Cahier de charges

2.1 Présentation du projet

Le présent projet s'inscrit dans le cadre du cours de Conception et architecture des réseaux informatiques (CAR). Il s'agit d'un projet de programmation réseau qui vise à organiser des échanges interactifs de données entre les utilisateurs qui se trouvent sur le même réseau local. Pour ce faire, il a été demandé de concevoir et d'implémenter une messagerie instantanée interactive sur le réseau local.

En rappel, la messagerie instantanée ou le dialogue en ligne, permet l'échange instantané de messages textuels et/ou de fichiers entre plusieurs personnes par l'intermédiaire d'ordinateurs connectés au même réseau informatique (réseau local) et plus communément celui d'Internet. Contrairement au courrier électronique, ce moyen de communication permet de conduire un dialogue interactif et instantané (quasiment en temps réel, les contraintes temporelles n'étant pas fortes dans ces systèmes)[1].

2.2 Les Objectifs visés par ce travail

Le présent projet vise deux objectifs principalement définis comme suit :

- La conception propre d'un protocole de messagerie instantanée. C'est ce protocole qui permettra de définir les règles de communication entre les utilisateurs de la messagerie.
- L'implémentation de ce protocole conçu dans un logiciel pour la communication interactive sur un réseau local. Dans notre cas, le logiciel sera une messagerie instantanée interactive dans le réseau local.

2.3 Les sources d'inspirations

Pour la conception de notre protocole et le développement de l'application, nous nous sommes inspirés du système de messagerie instantanée de Yahoo Messenger [2].

2.4 Les acteurs du projet

Les principaux acteurs de ce projet sont présentés dans le tableau (TABLE 1)

TABLE 1 – Acteur du projet

| Groupe | Nom et prénom |
|--|--|
| Pilotage : Ce groupe est responsable de la conduite du projet. Il donne les directives à suivre et valide les choix d'implémentation. | Pr NGUYEN Quang, Enseignant-chercheur (IFI). |
| Projet : Ce groupe est responsable de la mise en oeuvre des choix du groupe de pilotage. Il doit faire des propositions pour la conception et l'implémentation du système au groupe de pilotage pour validation. | <ul style="list-style-type: none"> — DIALLO Azise Oumar — KAFANDO Rodrique — KOUADIO Kouamé Olivier |

2.5 Les exigences de la messagerie instantanée

Conformément aux exigences du projet, notre système de messagerie instantanée doit satisfaire les spécifications fonctionnelles et non fonctionnelles.

2.5.1 Les exigences fonctionnelles

Elles sont présentées dans le TABLEAU 2. Il représente le « Product backlog » en utilisant notamment la méthode SCRUM.

Ainsi, de l'analyse du TABLEAU 2, nous avons construit le diagramme de cas d'utilisation de notre système présenté dans la figure (FIGURE 1).

2.5.2 Les exigences non fonctionnelles

Ces sont des exigences liées notamment à la performance, la sûreté, la confidentialité, la sécurité... du système. De plus, il faut chercher des critères mesurables de l'application. Le tableau (TABLE 3) regroupe les exigences non fonctionnelles de notre application.

TABLE 2 – Les exigences fonctionnelles de l'application (Product backlog)

| ID | Nom | Priorité | Coût | Comment démontrer | Observations |
|----|---|----------|------|--|---|
| 1 | Connexion possible sur différents poste de travail | 50 | 2 | Connexion à partir d'un poste1, réessayer de se connecter à nouveau à partir d'un autre poste2 connecté au réseau | Besoin de différents postes connectés au réseau |
| 2 | Enregistrement au réseau | 50 | 2 | A partir d'un poste connecté, lancer l'application avec l'identifiant | Besoin d'un poste connecté |
| 3 | Maintien de la liste des utilisateurs connectés | 50 | 2 | Connectez-vous, afficher la liste des utilisateurs connectés | Besoin du diagramme de séquence |
| 4 | Gestion des états de l'utilisateur (connecté, absent, occupé) | 30 | 3 | Connectez-vous, choisir le profil souhaité | Besoin du diagramme de séquence |
| 5 | Chat avec un utilisateur | 50 | 3 | Connectez-vous, afficher la liste des utilisateurs connectés, envoi du message en utilisant le nom de l'utilisateur destinataire | Besoin du diagramme de séquence |
| 6 | Chat avec plusieurs utilisateurs (conférence) | 40 | 5 | Connectez-vous, afficher la liste des utilisateurs connectés, envoi du message en utilisant le nom des utilisateurs destinataires | Besoin du diagramme de séquence |
| 7 | Échange/partage de fichier | 30 | 7 | Connectez-vous, afficher la liste des utilisateurs connectés, choisir le fichier, envoi du fichier en utilisant le nom de l'utilisateur destinataire | Besoin du diagramme de séquence |

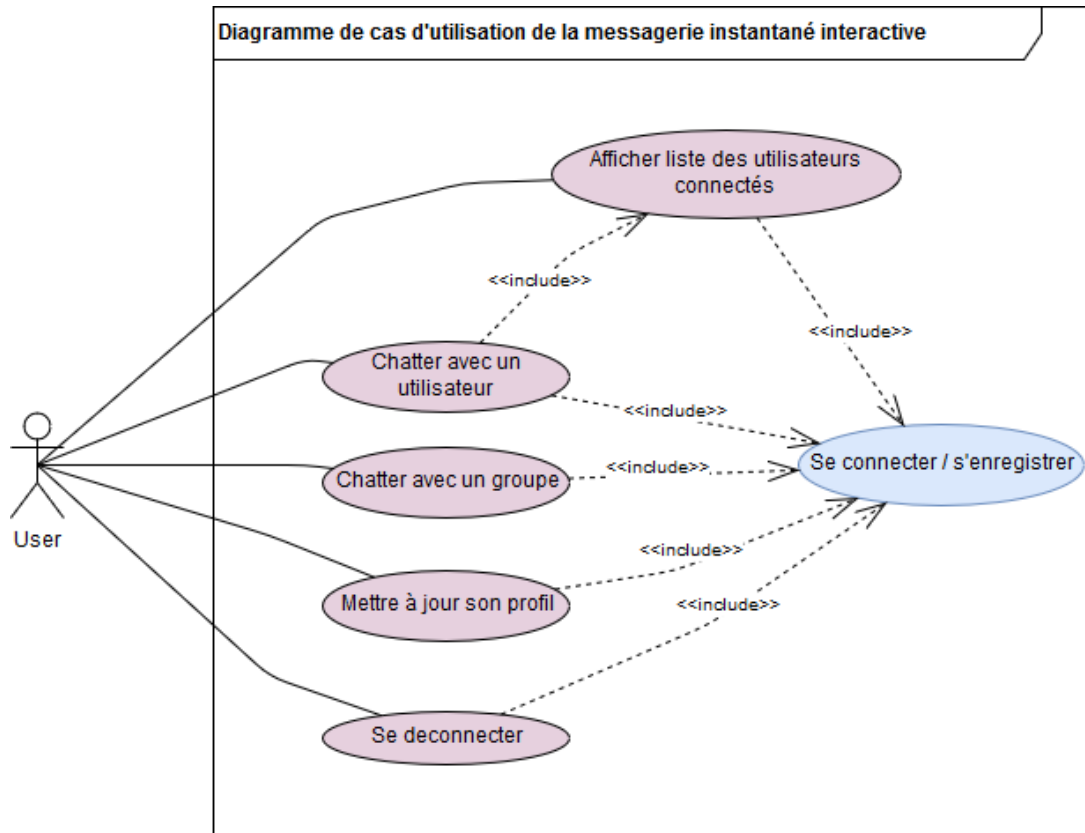


FIGURE 1 – Diagramme de cas d'utilisation de base de la messagerie

TABLE 3 – Les exigences non fonctionnelles de l'application

| Exigences | Description | Exigences fonctionnelles requises |
|---------------|---|-------------------------------------|
| Disponibilité | Le système doit être disponible durant toutes les opérations. Il doit être accessible à tout moment où les utilisateurs se trouvent sur un réseau local (ils ont le même réseau IP). | Toutes les exigences fonctionnelles |
| Intégrité | Il n'y a qu'un groupe de discussion sur le réseau. | Toutes les exigences fonctionnelles |
| Performance | L'envoi et la réception des messages doivent être inférieurs à 10ms. | Échanges des messages |
| Sécurité | La gestion du système est décentralisée, donc pas de serveur central pour gérer les utilisateurs. Les utilisateurs ne doivent pas voir les messages qui ne leur sont pas adressés sauf en cas de message de groupe | Échanges des messages |

3 Conception

Après la rédaction du cahier de charge dans la section 2, nous allons être à présent capable de faire la conception de notre système de messagerie. Ainsi, dans cette section nous allons présenter le protocole que nous avons utilisé dans la réalisation de l'application, l'architecture du système et le format des messages.

3.1 Protocoles

3.1.1 Protocole de communication

Dans cette partie, nous allons définir le dialogue (la liste des commandes et leurs syntaxes, la syntaxe des codes réponse et le diagramme des états) et le format des messages.

La communication entre processus applicatifs se fait grâce à des sockets. Un socket permet à un processus (le client) d'envoyer un message à un autre processus (le serveur).

Lors de la création d'un socket, il faut préciser le type d'adressage, le type de message et le protocole transport utilisés. Nous utiliseront : IPV4, les datagrammes simples et le protocole UDP.

Point de terminaison Un point de terminaison (EndPoint) est défini par une adresse IP et un numéro de port. Une communication s'établit entre deux points de terminaison.

Principe de communication Lors de l'envoi d'un message, il faut :

- créer un socket,
- créer le point de terminaison émetteur,
- lier le socket au point de terminaison émetteur (ce qui précisera le protocole transport utilisé pour l'émission),
- créer le point de terminaison récepteur,
- envoyer le message à l'aide du socket vers le point de terminaison récepteur.

Pour recevoir un message, le processus serveur doit :

- créer un socket,
- créer le point de terminaison récepteur (lui-même donc),
- lier le socket au point de terminaison récepteur (ce qui précisera le protocole transport utilisé pour la réception),
- créer le point de terminaison émetteur (sans préciser l'adresse IP ni le numéro de port puisqu'ils ne sont pas connus à ce stade)
- mettre le socket en état de réception en lui fournissant un buffer pour les données à recevoir, et une référence au point de terminaison émetteur,
- Lorsque le message est effectivement reçu, le point de terminaison émetteur est renseigné.

Le dialogue. Voici quelques commandes de base sous forme de codes :

- GROUPE_CODE : "gcd"
- ONE_USER_CODE : "onc"

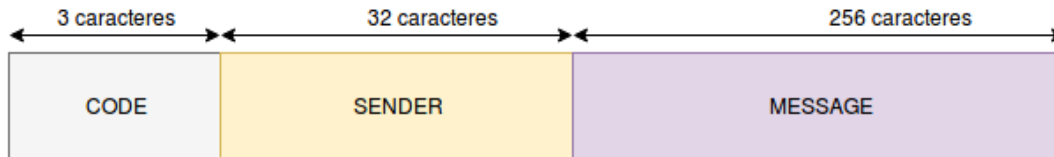


FIGURE 2 – *Format de message*

- USER_LIST_CODE : "ulc"
- KEEP_ALIVE_CODE : "kac"
- EXIT_CODE "exc"
- SEPARATEUR_MSG : ":"
- STATUS_DECONNECTE : "std"
- STATUS_CONNECTE : "stc"

Le format du message. Pour envoyer et recevoir des messages à travers le réseau un format particulier des messages doit être défini. Pour ce faire, nous avons défini un format des messages à échanger comme définit dans la FIGURE 2.

- Code : ce champ a une longueur fixée à 3 caractères maximal. Chaque type de message a un code particulier.
- Sender : ce champ contient le nom de l'utilisateur. Il a une taille de 32 caractères.
- Content : ce champ est réservé au contenu du message. Il est le plus important avec 256 caractères.

3.1.2 Protocole de transport

Comme dans le cas d'envoi de message simple, nous avons besoin d'un protocole de transport afin d'acheminer les messages vers les destinataires. Dans le cas de la messagerie simple (courriel), il faut s'assurer de la réception effective des messages peu importe le temps que cela peut prendre. Cependant, dans la messagerie instantanée interactive, le temps de réponse est capitale pour que les échanges soient vraiment interactifs. Aussi, la taille des messages est relativement plus courtes que dans un mail. En outre, on peut accepter de perdre quelques messages sans pour autant entraver à la communication. De plus, nous souhaitons faire du multipoint (multicast). Ainsi, nous avons opté pour le protocole UDP (User Datagram Protocol) pour le transport des messages.

C'est un protocole de la couche 4 (couche transport) du modèle OSI. Il n'ouvre pas de session et n'effectue pas de contrôle d'erreur (mode non connecté). De ce fait, il est peu fiable. Par contre, il permet aux applications d'accéder directement à un service de transmission de Datagrammes rapide (ce qui est notre cas) [3].

Adresse de broadcast. Étant donné que nous n'utilisons pas de serveur, nous utilisons une adresse de classe D qui est une adresse de multicast. Nous avons choisi le 225.0.0.100.

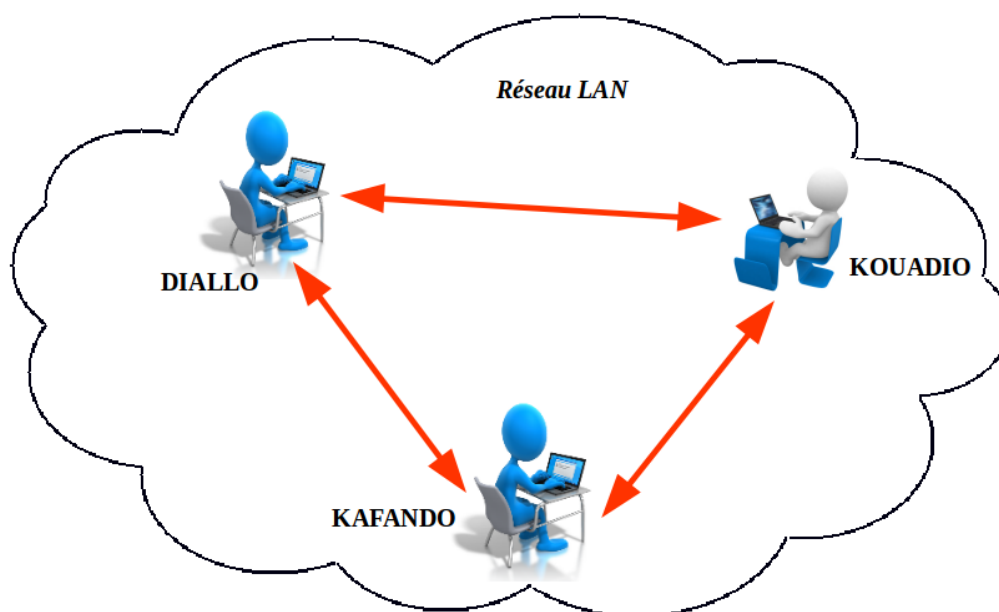


FIGURE 3 – Architecture du système avec trois postes utilisateurs

Port de communication. Comme toute application, notre application doit utiliser un port pour la communication. Le numéro de port que nous avons choisi est 10000. Ainsi, les processus applicatifs utiliseront ce port à travers les sockets pour communiquer.

3.2 Architecture du système

Dans les exigences du projet, il ne doit pas avoir de serveur central dans le système. Le système doit être distribué sur tous les postes utilisateurs. Ainsi, le système doit suivre une architecture *Peer-To-Peer (P2P)* car il n'y a pas de serveur central. Chaque poste joue à la fois le rôle de client et de serveur. De ce fait, chaque client peut communiquer directement avec un autre client. Ce type d'architecture à l'avantage d'être plus direct évitant ainsi les encombrements par l'utilisation d'un serveur. Le schéma (FIGURE 3) donne un aperçu de notre architecture avec trois postes utilisateurs.

3.3 Quelques diagrammes de séquence

Dans cette partie de notre travail nous présentons les diagrammes de séquence de la connexion au système, l'affichage de la liste des connectés et l'envoi de message à un utilisateur connecté. Ces différents diagrammes montrent les différents processus d'une fonctionnalité donnée.

Les schéma ci-dessous (FIGURE 4,5,6) présentent trois diagrammes de séquence que nous nous avons effectué.

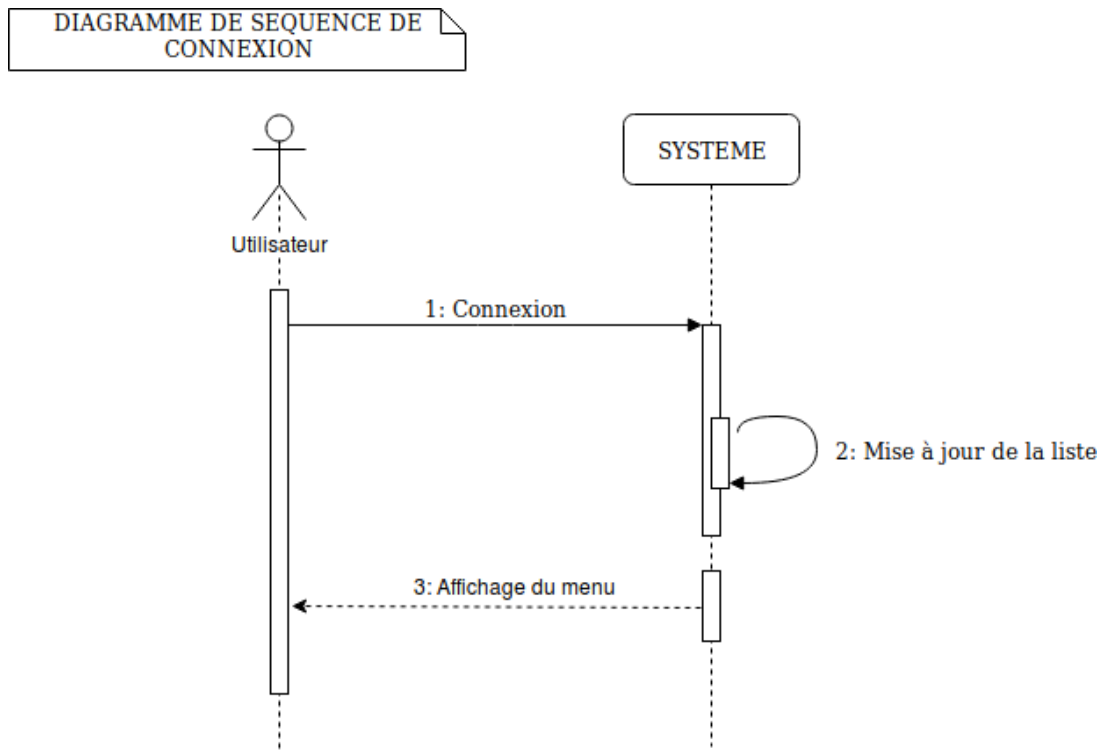


FIGURE 4 – *Diagramme de séquence de la connexion au système*

4 Implémentation

4.1 Environnement matériel

Pour le développement nous avons utilisé un ordinateur portable avec les caractéristiques suivantes :

- Processeur : Intel(R) Core™ i5-M370 @2.4 GHZ
- RAM : 8.00 Go
- OS : Ubuntu 16.04
- JDK8

Pour les expérimentations, nous avons créé un réseau local grâce à un point d'accès (AP) TP-Link disposant d'un routeur.

Les machines utilisées sont de caractéristiques diverses utilisant principalement Ubuntu 16.04 comme système d'exploitation. Nous avons utilisé au total trois (03) machines.

4.2 Environnement logiciel

4.2.1 Langage de programmation

Dans le cadre du présent projet, nous avons utilisé le langage de programmation C pour développer notre application de messagerie instantanée. Le choix de ce langage s'explique principalement par le fait que le langage C est un langage de bas niveau. De ce fait, il est donc approprié (indiqué) pour implémenter notre protocole et la gestion des ressources, telles que le

DIAGRAMME DE SEQUENCE DE
AFFICHAGE LISTE DES UTILISATEURS
CONNECTES

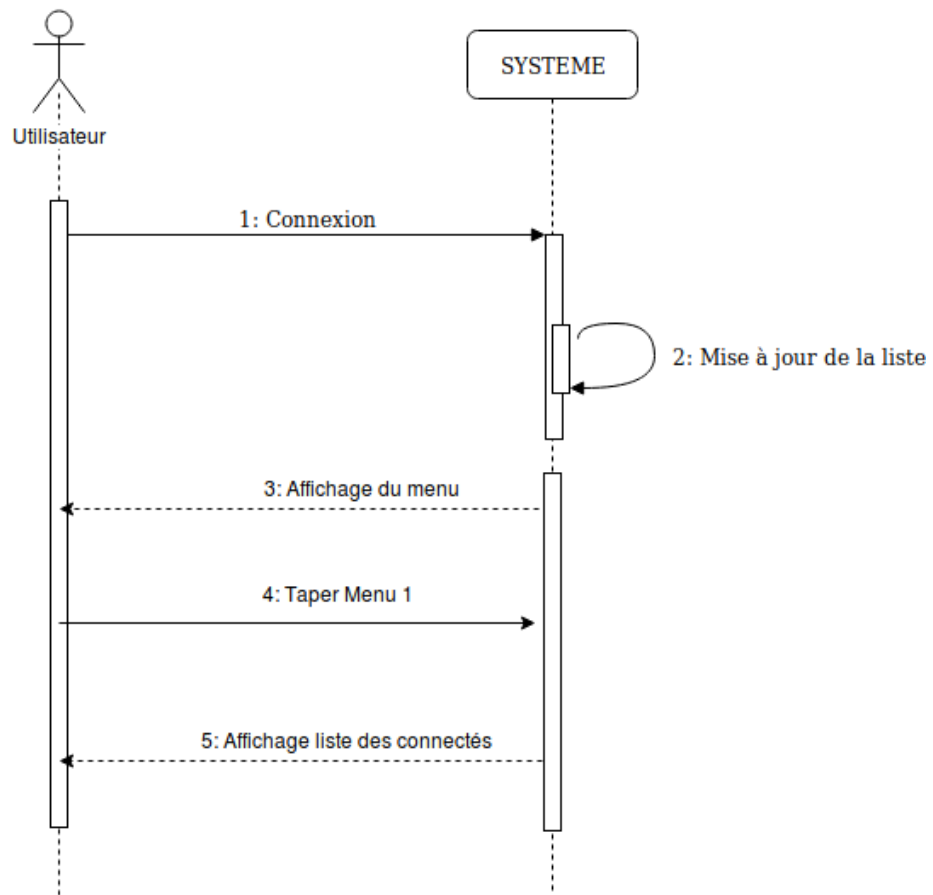


FIGURE 5 – *Diagramme de séquence de l’affichage de la liste des utilisateurs connectés*

DIAGRAMME DE SEQUENCE ENVOI DE MESSAGE

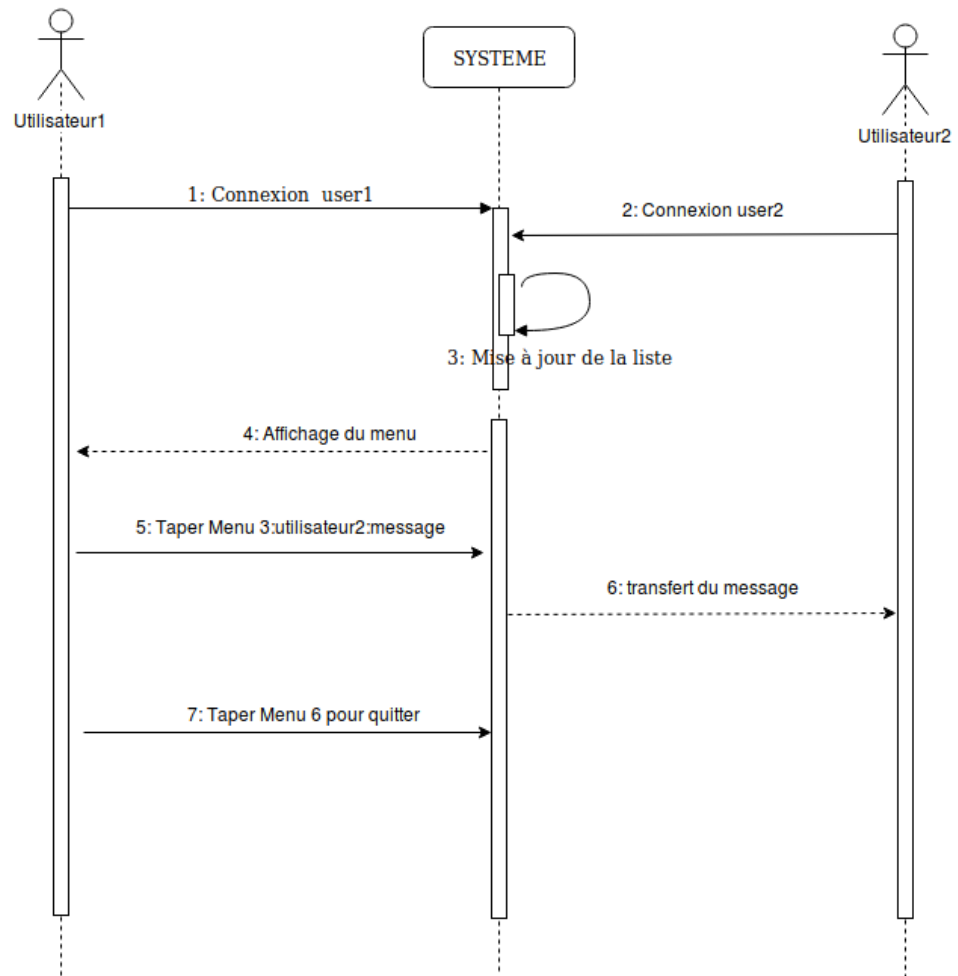


FIGURE 6 – Diagramme de séquence d'envoi de message à un utilisateurs connectés

temps et l'espace. En outre, il est recommandé pour la programmation réseau (voir notamment [4]).

4.2.2 Les plateformes et logiciels utilisés

Code : :Blocks. Dans le cadre du présent projet, nous avons utilisé l'environnement de développement intégrée *Code : :Blocks*.

Code : :Blocks est un IDE gratuit C, C++ et Fortran conçu pour répondre aux besoins les plus exigeants de ses utilisateurs. Il est conçu pour être très extensible et entièrement configurable (voir notamment [5]).

Dans le cadre du présent projet, nous avons utilisé la *version 16.01*.

Par ailleurs, plusieurs plateformes ont été utilisées durant le présent projet à savoir :

- Github : Pour l'hébergement et la gestion de développement de du système [6].
- draw.io : Pour la conception des diagrammes UML [7].
- ShareLatex (<https://fr.sharelatex.com/>) : Pour la rédaction du rapport [8].

5 Expérimentations et analyse des résultats

Dans ce qui suit, nous présentons des cas de tests afin de faire ressortir les fonctionnalités de base de notre messagerie instantanée.

5.1 Fonctionnement du programme

Le programme reçoit en entrée l'identifiant de l'utilisateur. En sortie, il affiche le menu correspondant aux commandes représentant les fonctionnalités de l'application. Pour chaque commande choisie, l'application effectue une tâche bien déterminée.

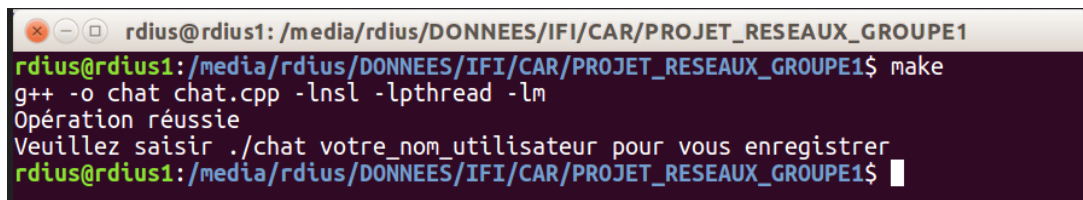
Le menu se compose comme suit :

- 1 : Pour afficher la liste des membres dans le groupe.
- 2 : Pour envoyer un message à tous les membres du groupe.
- 3 : `user_name :message_à_envoyer` : Pour envoyer un message à une seule personne à partir de son `user_name`.
- 4 : Pour mettre à jour la liste des utilisateurs.
- 5 : Pour signaler sa présence dans le groupe.
- 6 : Pour quitter l'application.

5.2 Compilation

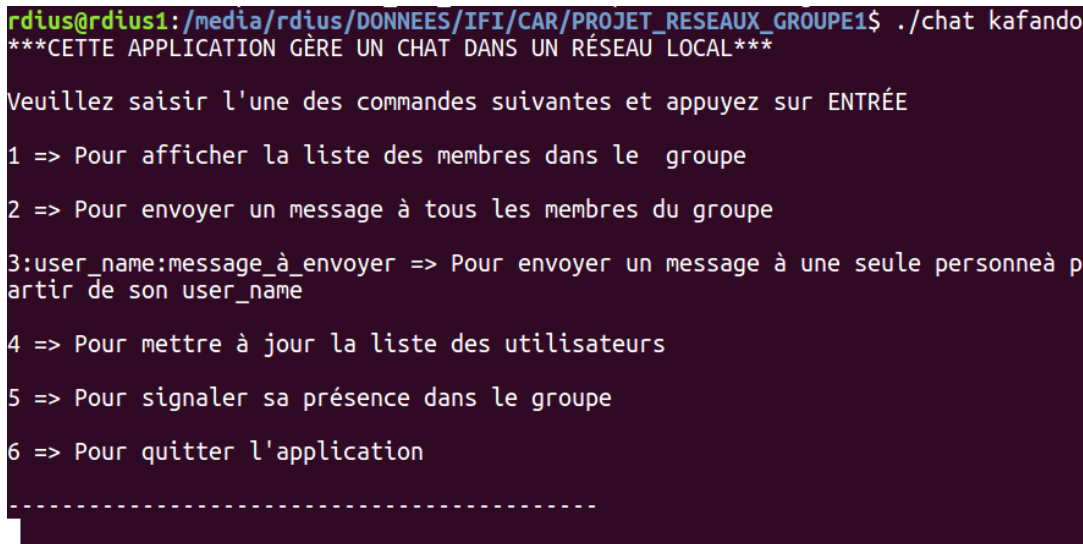
Pour la compilation du programme, nous avons créé un fichier *makefile*. Ainsi, pour compiler le programme, il suffit d'accéder en ligne de commandes (avec le terminal) au dossier contenant le fichier (exemple `~/MessagerieInstantanee`) les codes sources et taper la commande *make*.

La figure ci-dessous (FIGURE 7) donne le résultat de la compilation.



```
rdius@rdius1: /media/rdius/DONNEES/IFI/CAR/PROJET_RESEAUX_GROUPE1
rdius@rdius1:/media/rdius/DONNEES/IFI/CAR/PROJET_RESEAUX_GROUPE1$ make
g++ -o chat chat.cpp -lnsl -lpthread -lm
Opération réussie
Veuillez saisir ./chat votre_nom_utilisateur pour vous enregistrer
rdius@rdius1:/media/rdius/DONNEES/IFI/CAR/PROJET_RESEAUX_GROUPE1$
```

FIGURE 7 – *Compilation du programme*



```
rdius@rdius1:/media/rdius/DONNEES/IFI/CAR/PROJET_RESEAUX_GROUPE1$ ./chat kafando
***CETTE APPLICATION GÈRE UN CHAT DANS UN RÉSEAU LOCAL***

Veuillez saisir l'une des commandes suivantes et appuyez sur ENTRÉE

1 => Pour afficher la liste des membres dans le groupe
2 => Pour envoyer un message à tous les membres du groupe
3: user_name: message_à_envoyer => Pour envoyer un message à une seule personne à
partir de son user_name
4 => Pour mettre à jour la liste des utilisateurs
5 => Pour signaler sa présence dans le groupe
6 => Pour quitter l'application

-----

```

FIGURE 8 – *Exécution (lancement) du programme*

5.3 Exécution

Comme pour la compilation, on se place dans le répertoire contenant le code source et on tape `./chat username`. Ou « username » correspondant à l'identifiant d'un utilisateur (voir FIGURE 8).

5.4 Tests des fonctionnalités de l'application

Nous pouvons passer maintenant aux tests des fonctionnalités de l'application conformément aux exigences fonctionnelles.

5.4.1 Connexion au chat

Comme nous l'avons déjà vu dans la section 5.3, pour se connecter on tape la commande `./chat username`.

5.4.2 Affichage de la liste des utilisateurs connectés

Notre application étant sans serveur central, il est important d'afficher d'abord la liste des utilisateurs pour voir ceux qui sont connectés pour pouvoir chatter (voir FIGURE 9).

```

rdius@rdius1:/media/rdius/DONNEES/IFI/CAR/PROJET_RESEAUX_GROUPE1$ ./chat KAFANDO
***CETTE APPLICATION GÈRE UN CHAT DANS UN RÉSEAU LOCAL***

Veuillez saisir l'une des commandes suivantes et appuyez sur ENTRÉE

1 => Pour afficher la liste des membres dans le groupe
2 => Pour envoyer un message à tous les membres du groupe
3:user_name:message_à_envoyer => Pour envoyer un message à une seule personne à partir de son user_name
4 => Pour mettre à jour la liste des utilisateurs
5 => Pour signaler sa présence dans le groupe
6 => Pour quitter l'application

-----
1
Liste des utilisateurs connectés:
Vous êtes le seul dans le groupe pour le moment...
1
Liste des utilisateurs connectés:
1:Diallo
1
Liste des utilisateurs connectés:
1:Kouadio
2:Diallo

```

FIGURE 9 – Liste des utilisateurs connectés

Il faut noter que la liste des utilisateurs est mise à jour chaque 30 secondes. Ce temps a été arrêté après plusieurs expérimentations. Cependant, l'utilisateur peut faire une mise à jour manuelle de la liste quand il souhaite. En effet, comme on peut le voir sur la FIGURE 9, après sa première connexion, l'utilisateur est seul connecté (KAFANDO). Ensuite, il y a deux autres utilisateurs qui se sont connectés (Diallo d'abord, ensuite Kouadio).

5.4.3 Chat groupé

Le chat en groupe est possible. Pour ce faire il faut choisir « 2 suivi du message » dans le menu. Les FIGURES 10,11,12 donnent un aperçu des échanges du groupe « Projet » de ce projet de Réseau.

5.4.4 Chat avec un seul utilisateur

Pour chatter avec un seul utilisateur il faut choisir « 3 :user_name :message_à_envoyer » dans le menu. Les FIGURES 13 et 14 donne un aperçu d'un chat entre les utilisateurs KAFANDO et Kouadio.

5.5 Évaluation de performance du programme

L'analyse des différents tests d'expérimentation que nous avons effectués sur notre application, montre qu'elle satisfait les exigences fonctionnelles et non fonctionnelles de base du cahier des charges. En effet, on peut noter les points suivants :

- l'application sans serveur principal. L'application est distribuée sur tous les postes utilisateurs. De plus l'utilisateur peut se connecter (s'enregistrer) à l'application à partir de

```
diallitoz@diallitoz-HP-EliteBook-8470p: /media/diallitoz/01D22EB6BA0808E0/IFI/Cours/
diallitoz@diallitoz-HP-EliteBook-8470p:/media/diallitoz/01D22EB6BA0808E0/IFI/Cours/M1/CAR/Projet/PROJE
T_RESEAUX_GROUPE1/PROJET_RESEAUX_GROUPE1$ ./chat Diallo
***CETTE APPLICATION GÈRE UN CHAT DANS UN RÉSEAU LOCAL***

Veuillez saisir l'une des commandes suivantes et appuyez sur ENTRÉE

1 => Pour afficher la liste des membres dans le groupe
2 => Pour envoyer un message à tous les membres du groupe
3:user_name:message_à_envoyer => Pour envoyer un message à une seule personne à partir de son user_name
4 => Pour mettre à jour la liste des utilisateurs
5 => Pour signaler sa présence dans le groupe
6 => Pour quitter l'application

-----
1
Liste des utilisateurs connectés:
1:KAFANDO
2:Kouadio
2 Chao em...
Group:Diallo: Chao em...

Group:KAFANDO:Chao anh, ban khoe khong?
Group:Kouadio:Chao hông Diallo ban khoe khong?
2 ok cool cool, je vais bien. Merci
Group:Diallo: ok cool cool, je vais bien. Merci

2 Comment avance le projet de CAR?
Group:Diallo: Comment avance le projet de CAR?

Group:Kouadio:Oui nous avons fini le projet , il reste seulement à envoyer au professeur
2 OK, formidableGroup:KAFANDO:Effectivement, comme le dit Kouadio, on a fini

Group:Diallo: OK, formidab
2 c'est du bon boulot, Monsieur sera content...
Group:Diallo: c'est du bon boulot, Monsieur sera content...
█
```

FIGURE 10 – Chat en groupe vu par DIALLO


```

rdius@rdius1: /media/rdius/DONNEES/IFI/CAR/PROJET_RESEAUX_GROUPE1
-----
1
Liste des utilisateurs connectés:
1:Kouadio
2:Diallo
Group:Diallo: Chao em...

Chao anh, ban khoe khong?
Group:KAFANDO:Chao anh, ban khoe khong?

Group:Kouadio:Chao hông Diallo ban khoe khong?

Group:Diallo: Comment avance le projet de CAR?

Group:Kouadio:Oui nous avons fini le projet , il reste seulement à envoyer au pro
fesseur

Effectivement, comme le dit Kouadio, on a fini
Group:KAFANDO:Effectivement, comme le dit Kouadio, on a fini

Group:Diallo: OK, formidab

Group:Diallo: c'est du bon boulot, Monsieur sera content...

```

FIGURE 11 – Chat en groupe vu par KAFANDO

```

olivier@olivier-kouadio:~/Bureau/PROJET_RESEAUX_GROUPE1$ ./chat Kouadio
***CETTE APPLICATION GÈRE UN CHAT DANS UN RÉSEAU LOCAL***

Veuillez saisir l'une des commandes suivantes et appuyez sur ENTRÉE

1 => Pour afficher la liste des membres dans le groupe
2 => Pour envoyer un message à tous les membres du groupe
3:user_name:message_à_envoyer => Pour envoyer un message à une seule personne à partir de son user_name
4 => Pour mettre à jour la liste des utilisateurs
5 => Pour signaler sa présence dans le groupe
6 => Pour quitter l'application

-----
1
Liste des utilisateurs connectés:
1:Diallo
2:KAFANDO
Group:Diallo: Chao em...
Group:KAFANDO:Chao anh, ban khoe khong?
Chao hông Diallo ban khoe khong?
Group:Kouadio:Chao hông Diallo ban khoe khong?
Group:Diallo: ok cool cool, je vais bien. Merci
Group:Diallo: Comment avance le projet de CAR?
Oui nous avons fini le projet , il reste seulement à envoyer au professeur
Group:Kouadio:Oui nous avons fini le projet , il reste seulement à envoyer au professeur
Group:KAFANDO:Effectivement, comme le dit Kouadio, on a fini
Group:Diallo: OK, formidab
Group:Diallo: c'est du bon boulot, Monsieur sera content...

```

FIGURE 12 – Chat en groupe vu par Kouadio

- n'importe quel poste connecte au réseau ;
- le temps d'envoi et de réception d'un message est inférieur à 10ms ;
- la mise à jour de la liste des utilisateurs chaque 30 seconde ;
- l'utilisateur n'affiche pas un message s'il n'est pas récepteur ;
- seuls les utilisateurs d'un groupe peuvent recevoir et envoyer un message sur leur groupe.

Il reste cependant quelques fonctionnalités supplémentaires qui ne sont pris en charge. Celles-ci constituent les perspectives de notre projet (voir les « Perspectives » dans la section 6).

6 Conclusion générale

Notre projet de Conception et architecture des réseaux informatiques (CAR) a consisté à concevoir et implémenter un protocole de communication dans un logiciel pour la communication interactive sur un réseau local. Dans le cadre de ce projet, le logiciel est une messagerie instantanée interactive. Pour ce faire, nous nous sommes inspirés de Yahoo Messenger et son modèle de protocole (voir notamment [2]).

Ainsi, après la conception du protocole, nous avons développé l'application en se basant sur le cahier des charges que nous avons nous même rédigé sur la base des fonctionnalités du système.

Les résultats obtenus sont satisfaisants. En effet, nous avons pu implémenté les fonctionnalités de base de la messagerie instantanée.

A l'issu de ce travail, nous pouvons dire que ce projet a répondu largement à nos attentes. En effet, nous avons beaucoup gagné en compétences en ce qui concerne la conception les protocoles applicatifs de communication réseau et la programmation réseau.

Difficultés rencontrées. Les difficultés majeures que nous avons rencontrées durant ce projet sont en autre :

- La définition claire et la conception de notre protocole de communication. notre protocole est très simpliste. En réalité, la définition d'un tel protocole est plus complexe.
- La non maîtrise de la programmation réseau. Il s'agit en effet de notre première expérience.

Perspectives. Plusieurs points s'avèrent intéressants à explorer en vue d'améliorer notre protocole et par ricochet notre application de messagerie instantanée. Parmi ces points, on peut citer :

- L'intégration d'un serveur central pour un meilleure gestion des utilisateurs ainsi que de leurs sessions.
- L'intégration d'une interface graphique afin d'améliorer l'interface Homme-machine.
- La possibilité d'envoyer des fichiers dans les échanges.
- L'intégration de la voix ou de la vidéo dans la communication.
- L'archivage des messages envoyés et reçus.

Références

- [1] Wikipedia. Messagerie instantanée — wikipédia. https://fr.wikipedia.org/wiki/Messagerie_instantanée, 29 novembre 2017 à 16 :00, consultée le 01 décembre 2017.
- [2] Yahoo Messenger. ymsg-9. <http://libyahoo2.sourceforge.net/ymsg-9.txt>, , consultée le 01 novembre 2017.
- [3] James F Kurose and Keith W Ross. *Analyse structurée des réseaux : des applications de l'internet aux infrastructures de télécommunication*. Pearson Education, 2003.
- [4] Wikipedia. Programmation système — wikipédia. https://fr.wikipedia.org/wiki/Programmation_système, 11 décembre 2016 à 19 :45, consultée le 01 novembre 2017.
- [5] Code : :Blocks IDE. Code blocks. <http://www.codeblocks.org/>, 30 août 2015 08 :16, consultée le 01 novembre 2017.
- [6] Wikipedia. Github — wikipédia. <https://fr.wikipedia.org/wiki/GitHub>, 15 novembre 2017 à 14 :43., consultée le 30 septembre 2017.
- [7] Draw.io. Draw.io. <https://www.draw.io/>, , consultée le 01 octobre 2017.
- [8] Wikipedia. Sharelatex — wikipédia. <https://fr.wikipedia.org/wiki/ShareLaTeX>, 29 mai 2017 à 17 :11., consultée le 01 octobre 2017.