Protocole :

Sur la trame TCP, chaque paquet est envoyé de cette manière :

Les 4 premiers octets correspondent à un magic code qui vérifie l'authenticité du paquet,

Les 4 suivants correspondent à un code qui définit le type de requête,

Les 4 d'après définissent la taille X de la données à lire qui suit (entre 0 et 2048 octets)

Enfin X octets de données sous la forme de char[2048].

Type de requête :

On identifie 2 types de requête TCP :

- Le requête du client au serveur
- La réponse du serveur au client après ladite requête
- Les requêtes évènements émises par le serveur vers le client

Détails des requêtes :

- LOGIN :
  - Code = 000
  - Donnée = <username>\n<password>\n
  - Réponses possibles :
    - LOGIN_SUCCESS :
      - Code = 100
      - Donnée = <username>\n
    - LOGIN_FAILED :
      - Code = 200
      - Donnée = failed\n
- REGISTER :
  - Code = 001
  - Donnée = <name>\n<password>\n
  - Réponses possibles :
    - REGISTER_SUCCESS :
      - Code = 101
      - Donnée = <username>\n (username = name#XXX où XXX est un tag généré à l'inscription)
    - LOGIN_FAILED :
      - Code = 201
      - Donnée = failed\n
- ADD_CONTACT :
  - Code = 002
  - Donnée = <target_username>\n
  - Réponses possibles :
    - CONTACT_EXIST :
      - Code = 102
      - Donnée = <target_username>\n
    - CONTACT_ADD_FAILED :
      - Code = 202
      - Donnée = failed\n
- CALL :
  - Code = 003
  - Donnée = <target_username>\n<ip>\n<port>\n
  - Réponses possibles :
    - USER_NOT_FOUND :
      - Code = 102
      - Donnée = <target_username>\n
    - CONTACT_ADD_FAILED :
      - Code = 202
      - Donnée = failed\n

- ASK_CONTACT_LIST :
    - Code = 004
    - Donnée = <username>\n
    - Réponses possibles :
        - CONTACT_LIST :
            - Code = 004
            - Donnée = <contact1_username>\n<contact2_username>\n….
- LOGOUT :
    - Code = 010
    - Donnée = <username>\n

- CALL_WAS_REFUSE :
    - Code = 203
    - Donnée = <target_username>\n

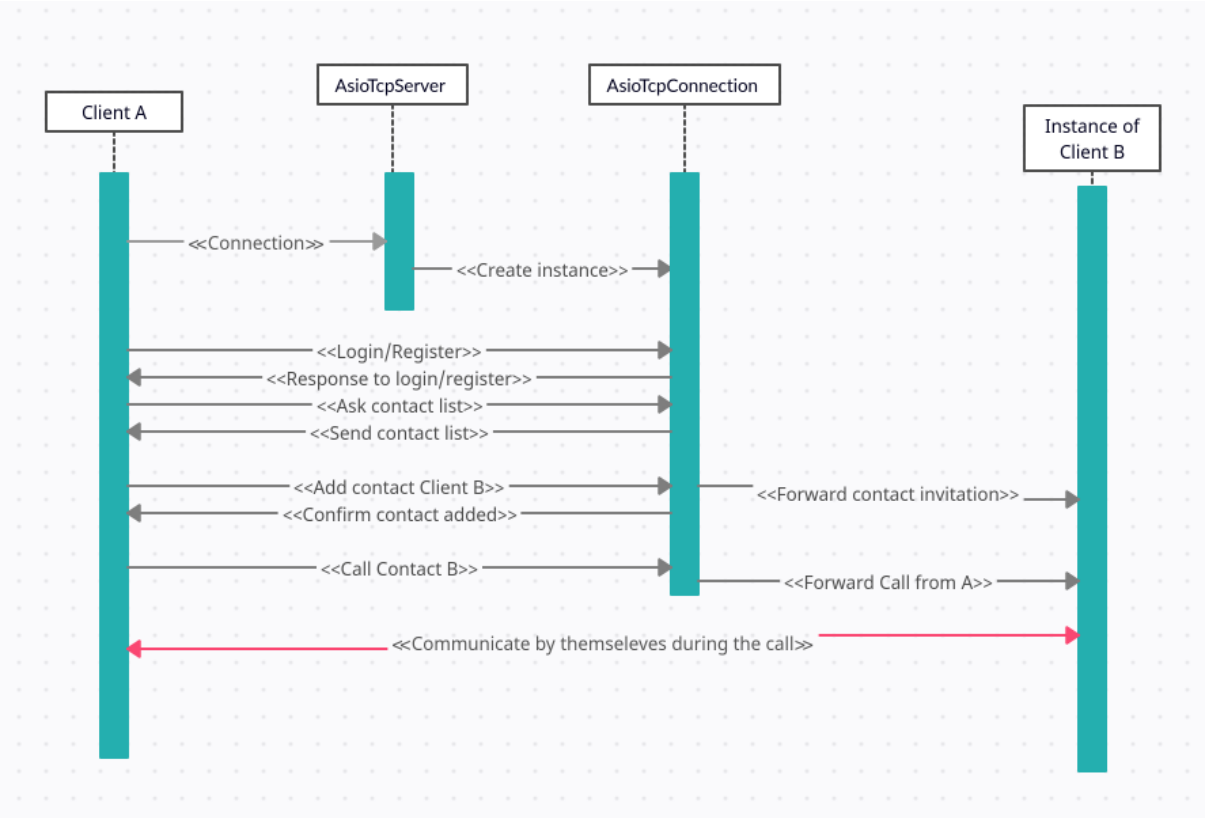
REQUÊTES ÉVÉNEMENTS :

- INCOMING_CALL :
    - Code = 303
    - Donnée = <username>\n<ip>\n<port>\n

- INVITATION_RECEIVE :
    - Code = 012
    - Donnée = <username>\n

- CALL_WAS_REFUSED :
    - Code = 203
    - Donnée = <username>\n

Diagramme de séquence :



Interface : (QT)

L'interface utilise en grande majorité un système de communication via des "QObject::connect()" qui permettent de communiquer via des signaux.

on créer un SIGNAL() qui prend ou pas des paramètres et qui les transmet à un SLOT() dans laquelle se trouve la fonction qui va être appelée à chaque fois que ce signal est envoyé avec "emit".

```
emit changePage(LOGIN);

QObject::connect(LoginPage, SIGNAL(changePage(pageNames)),

                 this, SLOT(changeCurrentPage(pageNames)));

void MainWindow::changeCurrentPage(pageNames name)

{  _pages->setCurrentPage(name); }
```

Cette fonctionnalité permet de faire communiquer les pages entre elles et de faire le lien entre les éléments de l'interface et les actions qui en découlent; servant entre autres à communiquer avec le serveur.

L'interface comprend une Page de référence "APage"

avec quelques fonctions et variables communes à toutes les pages :

```
        virtual void initConnections() = 0;
        virtual void loadPage() = 0;
        virtual void layoutLoader() = 0;
        virtual void onPage() = 0;
```

```
    protected:
        std::unique_ptr<QGridLayout> _layout;
        ClientInfos_t _infos;
```

ainsi que deux SIGNAUX qui permettent de changer de pages et de communiquer avec le serveur :

```
signals:
    void changePage(pageNames, ClientInfos_t);
    void checkCommand(ClientInfos_t, signal_e);
```

Le layout de la APage permet d'agencer les éléments “QTWidgets” dans chaque page et simplement load puis unload le layout des pages à chaque changement de celles-ci.