Réseau 2016 - 2017

1) Identification du groupe

Id	Membre	Application	Langage
#1	Thomas Perles	Serveur	Java
#2	Frédéric Lusinier	Client	Java

2) Protocole de transport

Si le message ne dépasse pas 93 caractères (utf 16, 10 pour le userName, 10 pour le roomName, 70 pour le message et un short int pour le mode) : UDP.
Sinon TCP à chaque requête pour que le client soit grandement indépendant par rapport au serveur.

La plupart des états sont stockés chez le client.
Le serveur ne stocke qu'une HashMap (roomName, moderatorName) (sous-entendu s'il y a un modérateur ou non) et répond aux requêtes.

3) Port utilisé pour communiquer

Port serveur : 12322

Port client : 12321

4) Les échanges entre le client et le serveur

4.1) Connection/déconnection

Envoyer une requête :

- RQ_CONNECT (String user, boolean connected)
  - le client stocke l'état de connexion de l'utilisateur

Deux réponses possibles :

- RP_CONNECT (boolean case)
  - true -> Connexion réussie
    - tous les boutons deviennent actifs
    - le bouton « connect » devient « disconnect »
  - false -> Déconnexion réussie
    - tous les boutons deviennent inactifs
    - le bouton « disconnect » devient « connect »
- RP_ERROR(short int request, short int case)
  - request = 0
    - request représente le discriminant de la fonction qui a causé l'erreur
    - Réponse en rapport avec la connexion
  - case = 0 -> error username already exist
    - Nom d'utilisateur déjà utilisé
  - case = 1 -> error username forbidden
    - Nom d'utilisateur interdit
  - case = 2 -> error memory space

        - espace mémoire insuffisant
    - ...
    - case = 255 -> error connection
        - Problème de connexion

4.2) Création d'une chambre

Envoyer une requête :

- RQ_CREATEROOM (String roomName, short int mode, String moderatorName)
    - mode = 0 pour free
    - mode = 1 pour moderated
    - moderatorName remplit automatiquement avec le nom du créateur

Deux réponses possibles :

- RP_CREATEROOMOK
    - chambre créée (cf structure)
    - chambre ajoutée à la liste des chambres
    - RQ_LOADROOMS
    - RQ_LOADMESSAGES
- RP_ERROR (short int request, short int case)
    - request = 1
        - Réponse en rapport avec la création de chambre

4.3) Chargement des chambres

Envoyer une requête :

- RQ_LOADROOMS ()

Deux réponses possibles :

- RP_LOADROOMSOK (short int length, struct [] room {String roomName, short int mode})
    - Mise à jour de la HashMap (roomName et mode) dans le rectangle 5
    - Mise à jour des roomName dans le rectangle 7
- RP_ERROR (short int request, short int case)
    - request = 2
        - Réponse en rapport avec le chargement des chambres

4.4) S'abonner / se désabonner (tout est traité en interne)

On utilise une HashMap (clef : String roomName, valeur : boolean subscribed), ce qui permet au client de décider de lire ou non les messages reçus en fonction de leur chambre.

4.5) Validation / invalidation du modérateur

Envoyer une requête :

- RQ_VALIDMODERATOR (String roomName, String userName, String message, short int valid)

Deux réponses possibles :

- RP_VALIDMODERATOROK (String roomName, String userName, String message, short int valid)
    - valid = 2 si le modérateur valide le message
    - valid = 3 si le modérateur invalide le message
- RP_ERROR (short int request, short int case)
    - request = 3
        - Réponse en rapport avec les décisions du modérateur

4.6) Poster un message

Envoyer une requête :

- RQ_POSTMESSAGE (String roomName, String userName, String message, short int valid)
    - valid = 0 indique que le client a envoyé un message

Deux réponses possibles :

- RP_POSTMESSAGEOK ()
    - le serveur a bien reçu le message
- RP_ERROR (short int request, short int case)
    - request = 4
        - Réponse en rapport avec l'envoi de message

4.7) clear

4.8) message

4.9) pending

5) Messages client vers serveur et organisation de ces messages

- RQ_CONNECT (String user, boolean connected)
- RQ_CREATEROOM (String roomName, short int mode, String moderatorName)
- RQ_LOADROOMS ()
- RQ_VALIDMODERATOR (String roomName, String userName, String message, short int valid)
- RQ_POSTMESSAGE (String roomName, String userName, String message, short int valid)

6) Messages serveur vers client et organisation de ces messages

- RP_ERROR(short int request, short int case)
- RP_CONNECT (boolean case)
- RP_CREATEROOMOK
- RP_LOADROOMSOK (short int length, struct [] room {String roomName, short int mode})
- RQ_VALIDATIONMODERATOR (String roomName, String userName, String message, short int valid) avec valid = 1
- RP_VALIDMODERATOROK (String roomName, String userName, String message, short int valid)
- RP_POSTMESSAGEOK ()

7) Organisation physique des types échangés

7.1) Types de base

Discriminant : entier long

Booléen : entier short (TRUE = 1111 1111 et FALSE = 0000 0000)

Caractère : UTF16 à la Java

Entier court : 8 bits en bigendian

Tableau : struct { int32 length ; byte [] content }

7.2) Types spécifiques

```
Room : struct {
        String roomName;            //UDP = 10 caractères UTF 16
        String moderatorName;       // UDP = 10 caractères UTF 16
}

Message : struct {
        String roomName;    //UDP = 10 caractères UTF 16
        String userName;    //UDP = 10 caractères UTF 16
        String message;     //UDP = 70 caractères UTF 16
        Short int valid;
}

Rooms : HashMap {
        String roomName;            //UDP = 10 caractères UTF 16
        String moderatorName;       //UDP = 10 caractères UTF 16
}
```