Pico Christopher : chris.pico@laposte.net
Blondel Kevin : kevblond77330@gmail.com

RFC

Toutes les chaines de caractère (String) devront être encodé de la même manière que ça soit coté client ou serveur.
L'encodage utilisé actuellement est l'utf8.
Un paquet commence par un byte, qui permettra de determiner quel requête est effectué.
Il faut préciser une taille maximal pour la taille des fichiers / message à envoyer (qui sera donc la même dans les 2 modules).

Voici la liste des requetes possible du protocol :

OpCode	Operation
1	Envoie pseudo au server (E_PSEUDO)
2	Requete de connexion client à client (E_CO_CLIENT_TO_CLIENT)
3	Confirmation de la connexion au client (ACK_CO_CLIENT)
4	Message d'un client vers un autre (M_CLIENT_TO_CLIENT)
5	Fichier d'un client vers un autre (F_CLIENT_TO_CLIENT)
6	Envoie d'un packet au serveur pour signifier que le client se deconnecte (DC_PSEUDO)
7	Requete d'un client pour recuperer la liste des clients connecté (D_LIST_CLIENT_CO)
8	Reponse du serveur pour envoie liste des clients (R_LIST_CLIENT_CO)
9	Message d'un client pour tout le monde (E_M_ALL)
10	Reponse du serveur à la requete de pseudo du client (R_PSEUDO)
11	Envoie de l'adresse serveur du client au serveur (E_ADDR_SERV_CLIENT)
12	Reponse du Serveur pour de connexion client à client (R_CO_CLIENT_TO_CLIENT)

Leurs format sont décris dans la partie suivante :

- Envoie pseudo du client au serveur : E_PSEUDO :
  Byte+Integer(tailleNom)+String(Nom)

- Requete connexion client à un client : E_CO_CLIENT_TO_CLIENT :
  Byte+Integer(tailleNomDestinataire)+String(NomDestinataire)
  +Integer (tailleNomSource) + String(NomSource)

- Confirmation de la connexion au client : (renvoyé 1 si accepté, 0 sinon) ACK_CO_CLIENT :
  Byte + Integer(tailleNom) + String(nom)

- Message client vers client : M_CLIENT_CLIENT:
  Byte +Integer(tailleNomEnvoyeur) + String(NomEnvoyeur)
  +Integer (tailleMessage) + String(Message)

- Fichier d'un client vers un autre : F_CLIENT_TO_CLIENT :
  Byte + Integer(tailleNomEnvoyeur) + String(NomEnvoyeur)
  + Integer(tailleNomFichier) + String(nomFichier)
  + Integer(tailleFichier)+(bytes)fichier

- Envoie d'un packet au serveur pour signifier que le client se deconnecte : DC_PSEUDO :
  Byte

- Demande liste des autres clients connectés : D_LIST_CLIENT_CO :
  Byte

- Reponse du serveur pour envoie liste des clients : R_LIST_CLIENT_CO :
  Byte + Integer(nombre de client connecté)
  + Liste des clients -> Pour chaque client :
    Integer(taille du pseudo du client) + String(pseudo du client)
    Integer(taille adresse client) + String(adresse du client au format host:port)

- Envoie d'un message à tout le monde : E_M_ALL :
  Byte + Integer(tailleNom) + String(Nom)
  + Integer (tailleMessage)+ String(Message)

- Reponse du serveur à la requete de pseudo du client : R_PSEUDO :
  Byte + Integer (1 le pseudo demandé existe déjà)
  (0 le pseudo demandé n'existe pas et est donc valide)

- Envoie de l'adresse serveur du client au serveur : E_ADDR_SERV_CLIENT :
  Byte + Integer (taille de l'adresse que l'on envoie) + String (adresse à envoyer)

- Reponse du serveur à la requete de pseudo du client : R_CO_CLIENT_TO_CLIENT :
  Byte + Integer (1 la demande a bien été transmise)
  (0 le client demandé n'existe pas)

Afin de facilité l'écriture et la lecture , par la suite on ne précisera pas les informations évidente des paquets (par exemple avec le client qui envoie son pseudo, inutile de redire ce que vaut le Byte au début du paquet et l'entier "tailleNom" est suffisament explicite) , mais juste la description de ce que sont chacun

Plus de précision concernant les requête :

E_PSEUDO : Le client envoie un paquet contenant son pseudo au serveur. Le serveur vérifie que personne n'est connecté avec ce pseudonyme et renvoie un paquet **R_PSEUDO** qui précise s'il y a deja quelqu'un connecté avec ce pseudo.

E_CO_CLIENT_TO_CLIENT : Un client envoie un paquet contenant son nom et le nom du destinataire afin de lui demander s'il peut communiquer avec . Le serveur reçoit le paquet et envoie ce même paquet sans le nom du destinataire. Le client recoit ce paquet c'est alors qu'un message affiche la demande de connexion avec la ligne
À taper pour accepter la connexion.

Une fois cette ligne taper, le client renvoie donc un paquet contenant **ACK_CO_CLIENT** pour valider la connexion à l'autre client.

Le serveur renvoie un paquet **R_CO_CLIENT_TO_CLIENT** a l'emetteur de la demande de connexion , pour lui spécifier si la demande de connexion a bien été transmise , dans le cas contraire il indique que le client demandé n'existe pas.

M_CLIENT_CLIENT : Une fois la connexion établit entre deux clients on peux s'envoyer des messages, il faut préciser le type de message et le destinataire pour que cela se produise.
Le paquet contient son propre nom et son message ce qui permettra au client qui recoit ce paquet de directement afficher le message sous la forme "[private] from *nom* : *message*"

F_CLIENT_TO_CLIENT : Une fois la connexion établit entre deux clients on peux s'envoyer des fichier,
Le paquet contient son propre nom et le fichier voulu ce qui permettra au client qui recoit ce paquet de directement recevoir le fichier et il verra afficher "L'utilisateur *user* vous envoie le fichier *nomFichier*"

DC_PSEUDO : Un client envoie un paquet **DC_PSEUDO** contenant uniquement un byte indiquant la requete, afin d'indiquer au serveur qu'il se deconnecte et que ce dernier ferme donc la connexion avec ce client et le retire des informations stockés.

D_LIST_CLIENT_CO : Un client envoie un paquet **D_LIST_CLIENT_CO** contenant uniquement un byte indiquant la requete , afin de demander au serveur la liste des clients actuellement connecté. Le serveur répond avec la requête **R_LIST_CLIENT_CO** qui contient la liste des clients connectés afin d'actualiser cette liste pour chaque client.

E_ADDR_SERV_CLIENT : après l'envoie du pseudo au serveur, le client créer un paquet contenant son adresse pour que le serveur le repertorie dans sa liste de client.

Chaque client connait la liste des clients connecté à n'importe quel moment . Chaque client à aussi une liste d'amis avec qui il peut communiquer