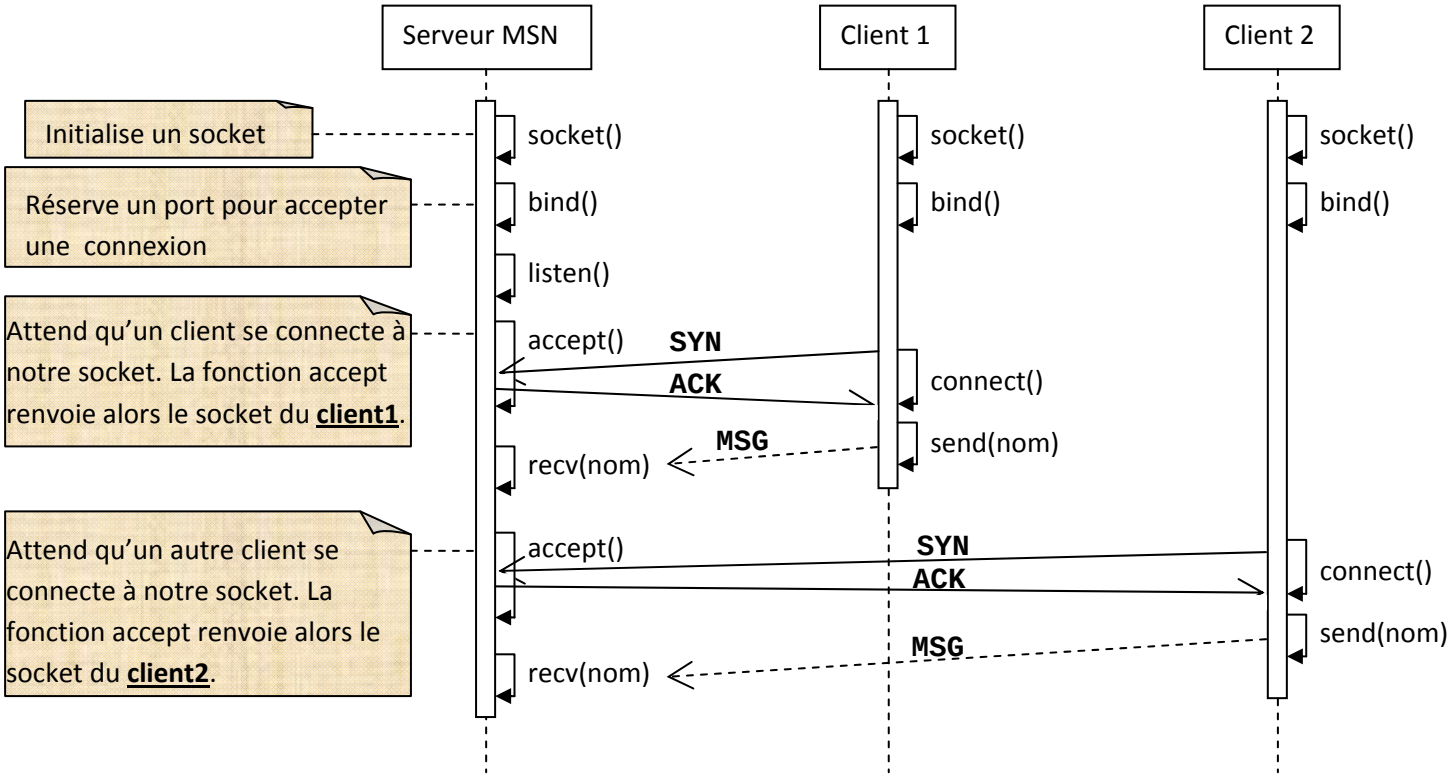
Notre MSN

Fonctionnement réel

Lors du TP1, nous avons pu voir comment faire pour s'envoyer des messages entre ordinateurs. Proche du fonctionnement de MSN, ce n'est pourtant pas comme ça que les ingénieurs de Microsoft ont créés leur logiciel. En réalité, MSN est basé sur le modèle Client-Serveur. Au lancement du logiciel, on se connecte chez Microsoft et on s'identifie. Pour des raisons de simplicité, on va supposer pour l'instant qu'il ne peut y avoir que deux clients qui parlent à tour de rôle. Le diagramme de séquence suivant présente la phase d'initialisation :



Etape 1 : Création d'un serveur

En vous aidant de ce diagramme de séquence, créez un serveur MSN acceptant deux clients. Pour pouvoir dialoguer avec deux clients, il faudra qu'il ait deux sockets : un par client.

Créez donc un nouveau projet que vous appellerez Serveur, et sans faire de classes, créez une fonction **`main`** qui fait ce qui est demandé (à savoir attendre deux clients). Une fois les deux clients connectés, affichez leurs noms sur l'écran.

Etape 2 : Création des clients

Créez un autre projet où vous utiliserez la classe **`connexion`** créée lors du TP1 pour vous connecter au serveur. Une fois la connexion faite, envoyer votre nom (que l'utilisateur saisi au clavier…).

Ensuite attendez une réponse du serveur.

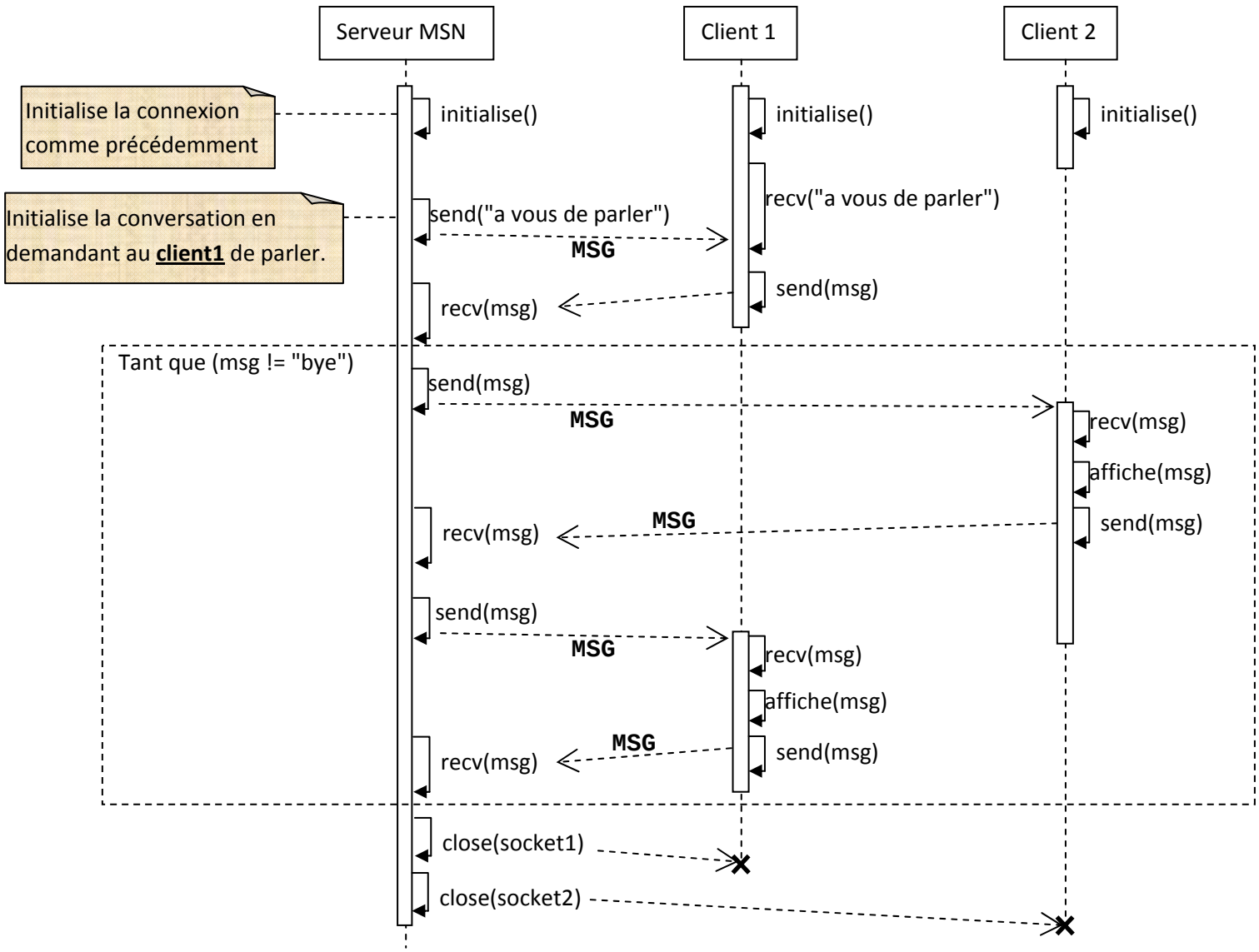
Etape 3 : Tests de fonctionnement

Pour vérifier que tout marche bien, mettez en place ce scénario :

- Vous lancez le serveur sur une machine
- Vous lancez un client sur une autre machine, et donnez le nom **client1**
- Vous lancez un client sur une autre machine, et donnez le nom **client2**
- Vérifiez que les deux noms s'affichent sur le serveur.

Etape 4 : Un véritable dialogue

Maintenant que vous arrivez à accepter deux clients sur le serveur, nous allons mettre en place un dialogue entre les deux clients. En vous basant sur le diagramme de séquence suivant, implémentez les programmes **`client`** et **`serveur`** (on suppose que l'initialisation marche) :



Etape 5 : de la couleur

Pour bien différencier les deux clients, MSN propose de donner de la couleur au texte. Vous allez faire une fonction similaire. Pour cela, vous utiliserez la fonction `SetConsoleTextAttribute` de la librairie `windows.h`. Donc pendant la phase d'initialisation, le serveur devra demander en plus aux clients la couleur de leur texte. Trouvez un moyen pour faire passer cette information du serveur aux clients.

Etape 6 : envoyer des fichiers

Enfin, pour que notre MSN soit parfaitement opérationnel, il faut pouvoir envoyer des fichiers. Commençons par essayer d'envoyer un fichier au serveur. Par exemple, le client1 possède l'image `img1.jpg`. Pour l'envoyer à travers le réseau, il va falloir découper ce fichier en paquets, les envoyer à travers le réseau et réassembler les différents paquets sur le serveur.

Etape 6.1 : découper les paquets

Pour découper un fichier, il faut commencer par l'ouvrir, en utilisant la classe `ifstream`.

```
#include <fstream.h>
...
char buffer[250];
ifstream myFile ("img1.jpg", ios::in | ios::binary);
```

Ensuite, on va lire 250 octets par 250 octets, et on va les envoyer au serveur :

```
myFile.read (buffer, 250);
send(monSocket, buffer,250,0);
while(myFile.read (buffer, 250)) {
   send(monSocket, buffer,250,0);
}
send(monSocket, buffer, myFile.gcount(),0);
myFile.close();
```

L'avant dernière ligne permet d'envoyer les derniers octets du fichier.

Etape 6.2 : assembler les paquets

Pour assembler les paquets reçus, on va créer un fichier avec la classe `ofstream`.

```
#include <fstream.h>
...
char buffer[512];
ofstream myFile ("img1.jpg", ios::out | ios::binary);
```

Ensuite, on va écrire les octets que l'on reçoit du serveur :

```
int nbOctet=0;
while((nbOctet=recv(monSocket, buffer, 512,0))>0){
      myFile.write (buffer, nbOctet);
};
myFile.close();
```