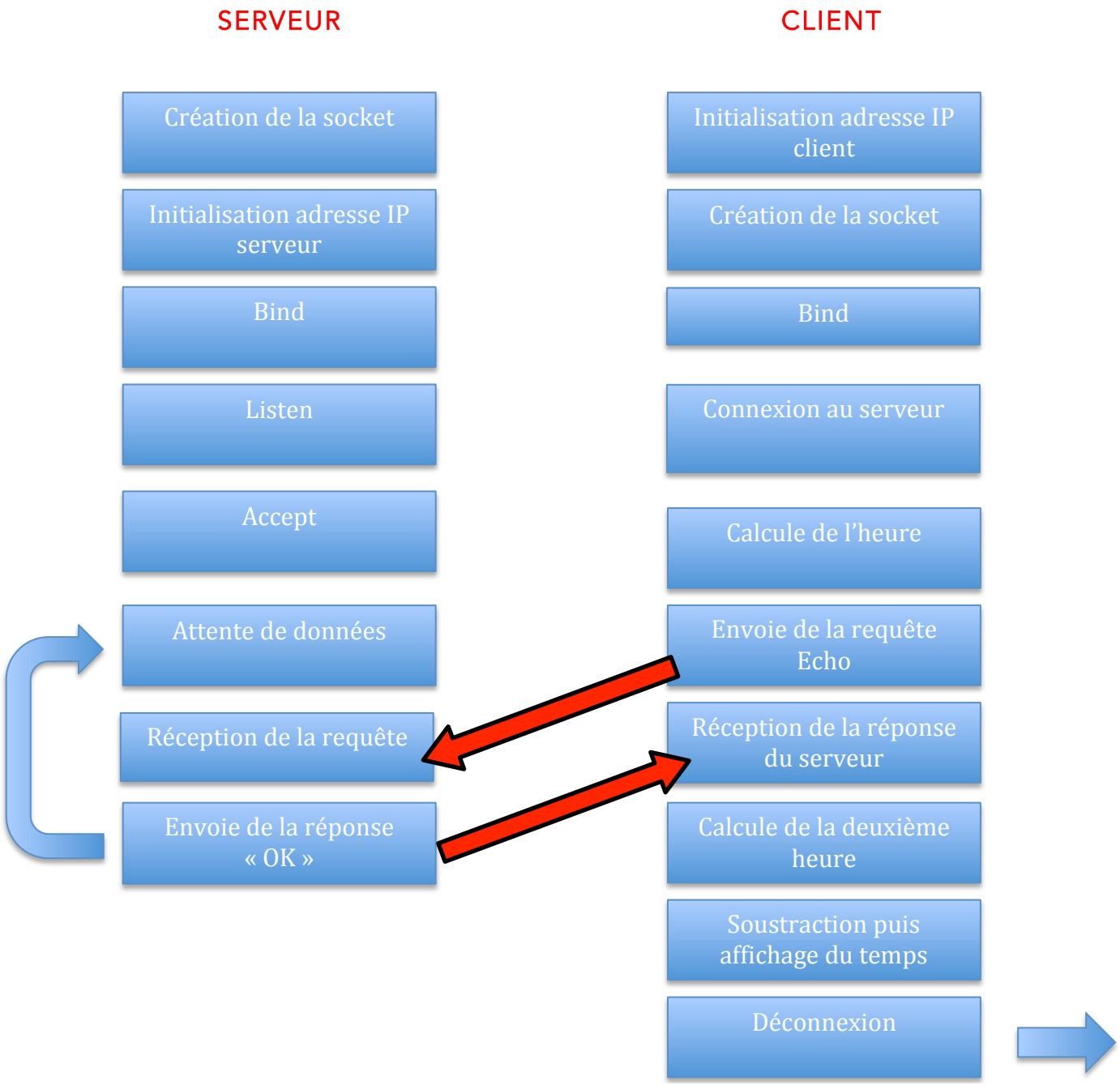
TROTTIER Arthur
SEBILLE Florian
L2 SPI

Développement d'applications - Client/Serveur (Socket TCP et UDP)

ServeurEcho :

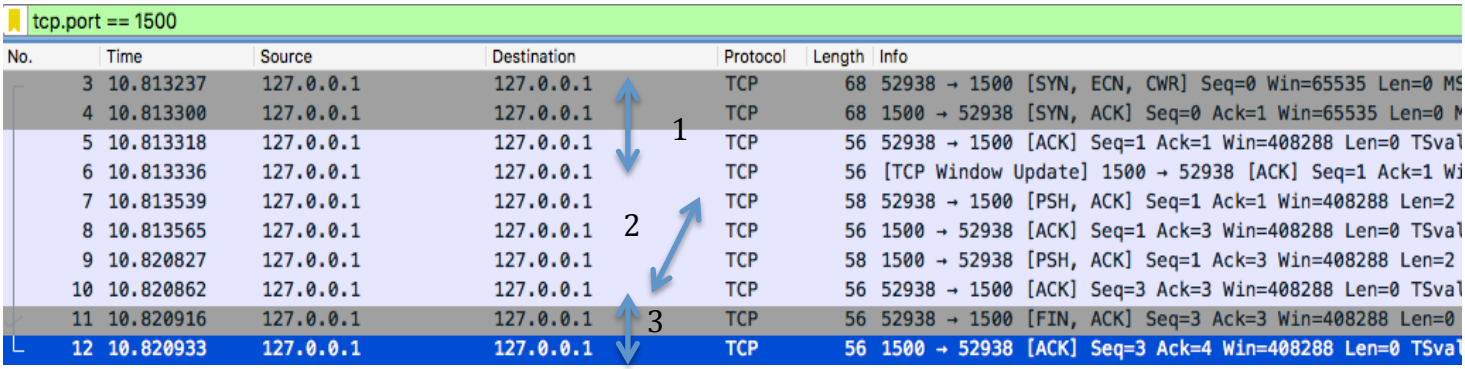
1. Principe et organisation

L'application a pour but de calculer le temps d'aller/retour entre un client et un serveur. Pour cela, l'heure sera calculée avant l'envoi de la requête puis après la réponse du serveur au client. Ensuite il suffira de faire une soustraction entre le deuxième temps et le premier. Ci-dessous l'organisation :



2. Wireshark et visualisation

Pour visualiser les échanges client/serveur ainsi que les trames, nous pouvons utiliser le logiciel Wireshark. Ainsi on obtient ceci :



On impose au logiciel un filtre qui permettra d'afficher uniquement les trames où le port TCP 1500 entre en jeu (source ou destination).

Partie 1 (N°3 à 6) : Il s'agit du protocole de connexion entre le client et le serveur, on observe que la longueur de données des trames est nul (Len = 0). Il n'y a que le n° d'ACK et de SEQ qui change et qui passe à 1.

Partie 2 (N°7 à 10) : La connexion effectuée, les échanges de données peuvent désormais commencer. Pour chaque envoi, on a l'ACK venant du receveur qui va avec qui correspond au numéro de séquence de la trame reçu + LEN + 1.
Dans cet exemple on observe 2 échanges de données (1 requête du client et une réponse du serveur).

Partie 3 (N°11 à 12) : L'échange terminé, le client demande une déconnexion au serveur qui répond avec ACK semblable au SEQ de la demande + 1 (les données étant nulles).

3. Code et visualisation du programme

Figure 1/2 : Exemple de requête Echo

```
kirozz@Florian-PC:/mnt/c/Users/Florian/Desktop/network-master/echo$ ./client 127.0.0.1
 envoie demande echo 127.0.0.1
la reception de la reponse a la requete a pris 0 seconds et 11 ms
kirozz@Florian-PC:/mnt/c/Users/Florian/Desktop/network-master/echo$
```

```
kirozz@Florian-PC:/mnt/c/Users/Florian/Desktop/network-master/echo$ ./serveur
./serveur : en attende de donnees 1500
cliaddr.sin_family : 2
cliaddr.sin_addr.s_addr : 127.0.0.1
cliaddr.sin_port : 14019
./serveur : en attende de donnees 1500
```

Figure 3 : Partie du code client

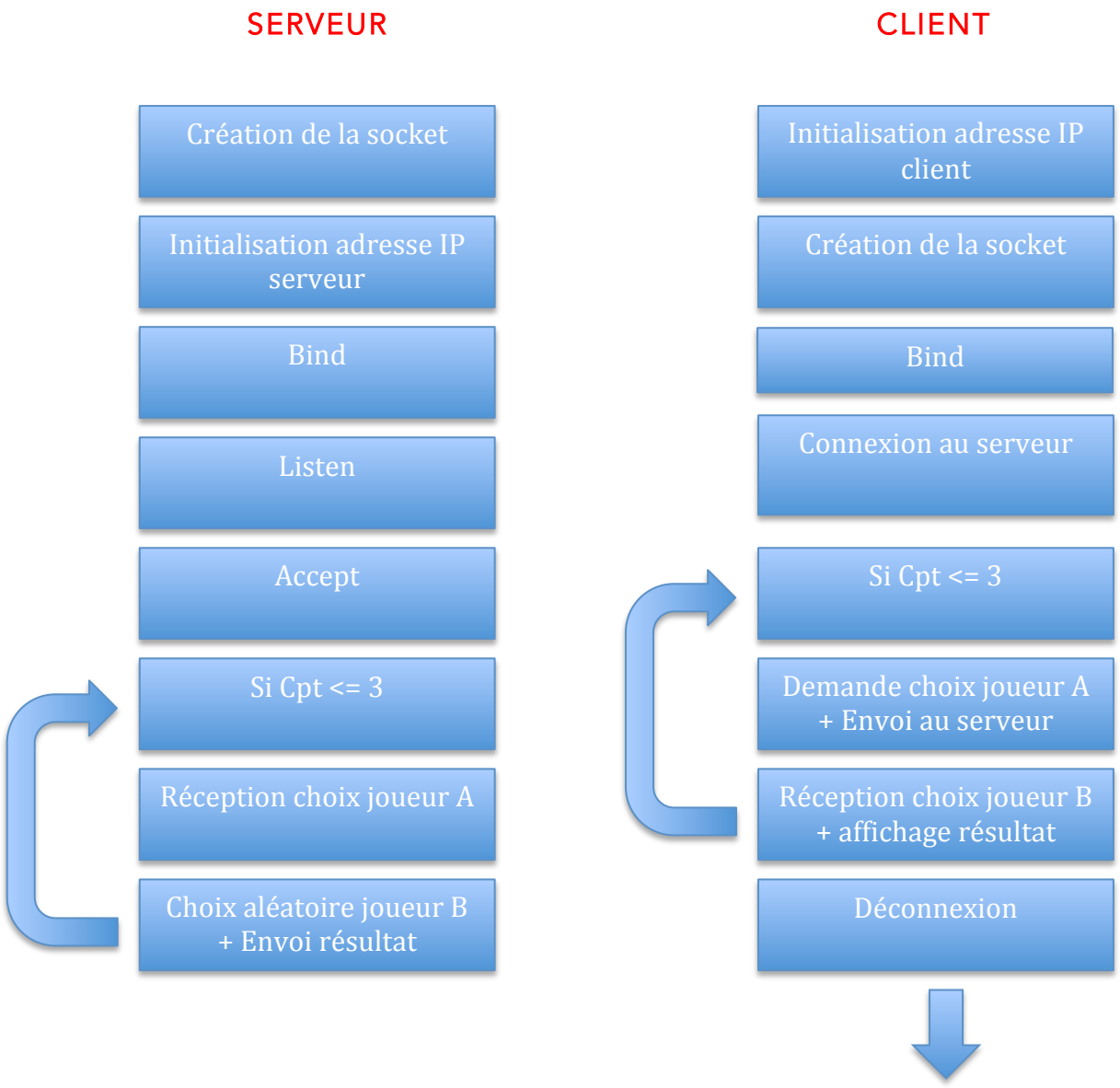
```
ftime(&temps_debut);    /*recuperation temps avnt l'envoie de la requete*/

rc = send(sd, "OK", sizeof(char)*2, 0);
printf("envoie demande echo %s\n", argv[1]);

rc = recv(sd, line, sizeof(line) + 1, 0);
ftime(&temps_fin);    /*recuperation temps aprèes reception de la reponse*/
printf("la reception de la reponse a la requete a pris %lu seconds et %d ms\n", (temps_fin.time - temps_debut.time), (temps_fin.millitm - temps_debut.millitm));

close(sd);
return 0;
```

La première fonction ftime récupère l'heure avant l'envoi de la requête du client au serveur. Une fois la requête émise et la réponse renvoyée, on récupère une nouvelle fois l'heure. Ensuite pour l'affichage nous faisons la soustraction entre la deuxième heure et la première.

Pierre/Feuille/Ciseaux :

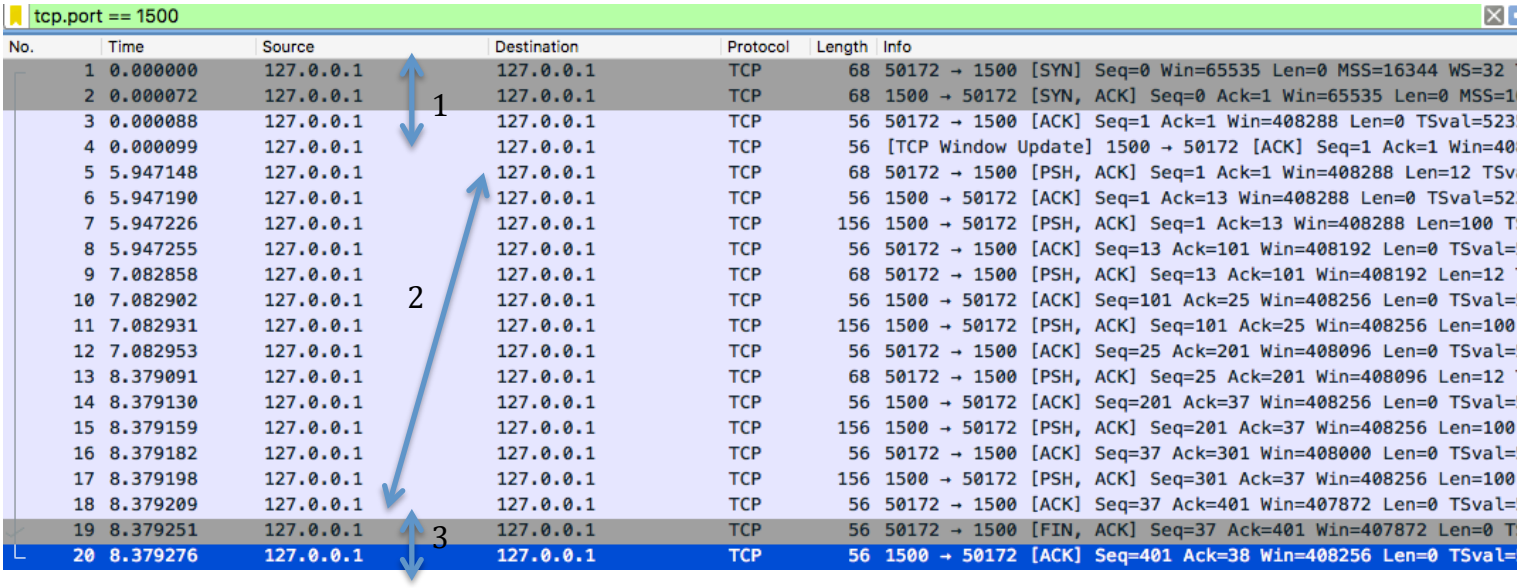
1. Principe et organisation

L'application a pour but de simuler une partie de pierre/feuille/ciseaux entre un client (Joueur A) et un serveur (Joueur B). Pour cela, après chaque tour du joueur A, le serveur renvoi le résultat aléatoire de la manche. La partie se joue en 3 manches, le joueur qui en remporte le plus gagne. Ci-dessous l'organisation :



2. Wireshark et visualisation

Pour visualiser les échanges client/serveur ainsi que les trames, nous pouvons utiliser le logiciel Wireshark. Ainsi on obtient ceci :



On impose au logiciel un filtre qui permettra d'afficher uniquement les trames où le port TCP 1500 entre en jeu (source ou destination).

Partie 1 (N°1 à 4) : Il s'agit du protocole de connexion entre le client et le serveur, on observe que la longueur de données des trames est nul (Len = 0). Il n'y a que le n° d'ACK et de SEQ qui change et qui passe à 1.

Partie 2 (N°5 à 18) : La connexion effectuée, les échanges de données peuvent désormais commencer. Pour chaque envoi, on a l'ACK venant du receveur qui va avec qui correspond au numéro de séquence de la trame reçu + LEN + 1.
Dans cet exemple on observe 7 échanges de données (3 envois du choix de A, 3 envois du choix de B + Envoi du résultat final).

Partie 3 (N°19 à 20) : L'échange terminé, le client demande une déconnexion au serveur qui répond avec ACK semblable au SEQ de la demande + 1 (les données étant nulles).

3. Exemples de trames et visualisation du programme

Figure 1 : Exemple d'une partie (Affichage coté client)

```
MANCHE n°1 : Tapez votre choix --> PIERRE: 0, FEUILLE: 1, CISEAU: 2
Choix: 2
MANCHE n°1 : client "ciseau" VS  serveur "feuille" --> Resultat client gagne

MANCHE n°2 : Tapez votre choix --> PIERRE: 0, FEUILLE: 1, CISEAU: 2
Choix: 0
MANCHE n°2 : client "pierre" VS  serveur "feuille" --> Resultat serveur gagne

MANCHE n°3 : Tapez votre choix --> PIERRE: 0, FEUILLE: 1, CISEAU: 2
Choix: 1
MANCHE n°3 : client "feuille" VS  serveur "ciseau" --> Resultat serveur gagne

serveur est le vainqueur
```

Figure 2 : Exemple d'une trame de données (Client vers Serveur)

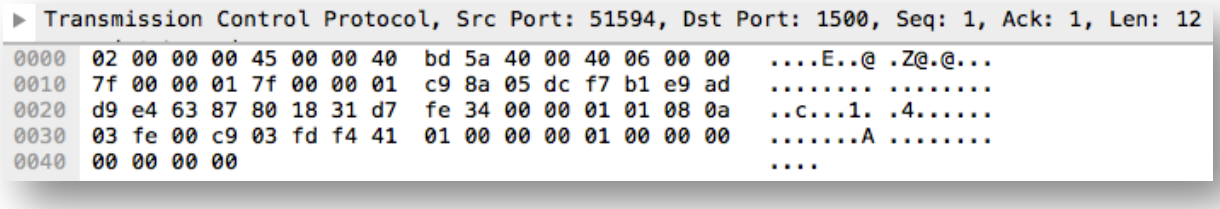


Figure 3 : Exemple d'une trame de données (Serveur vers Client)