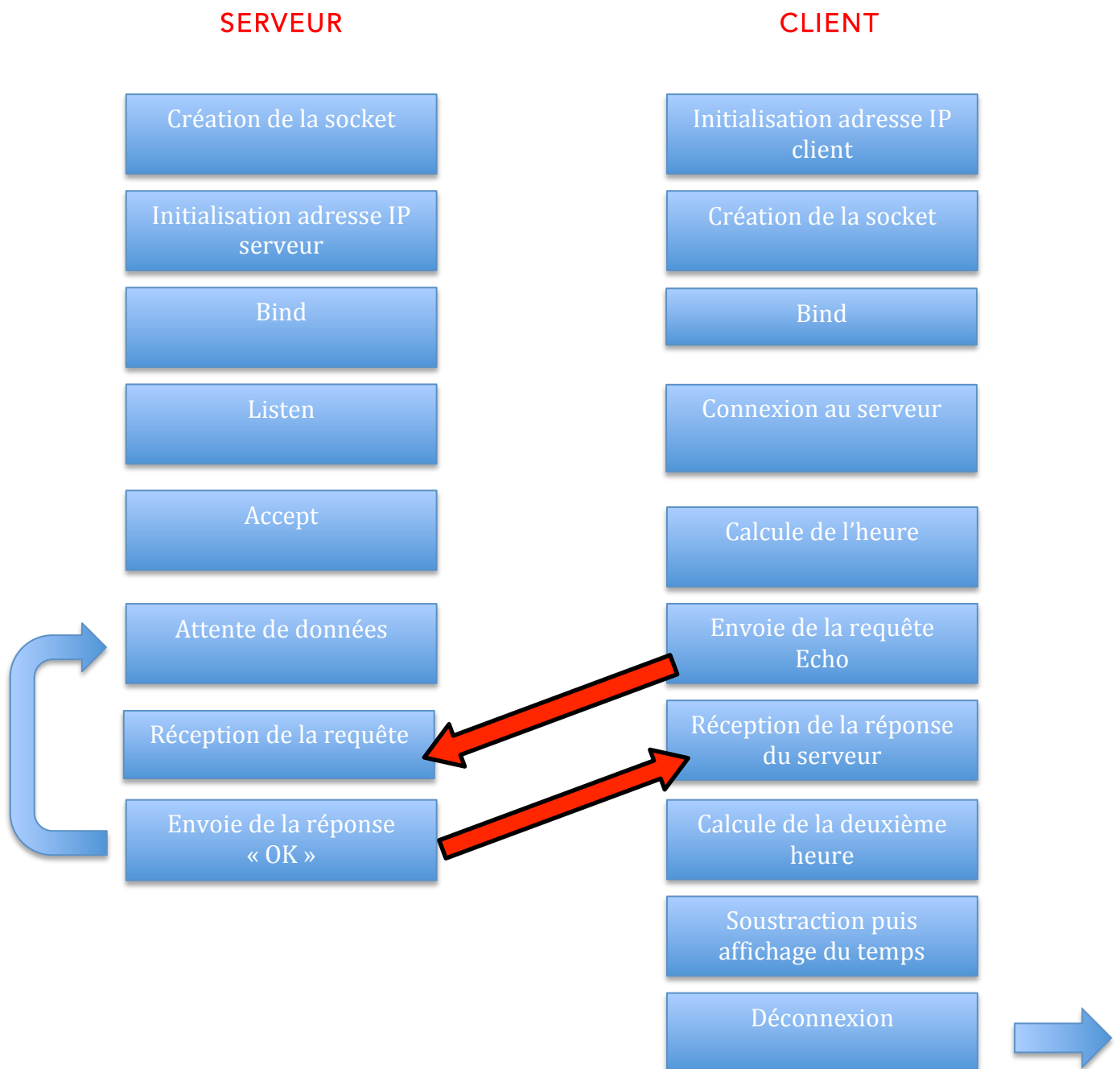


# Développement d'applications - Client/Serveur (Socket TCP et UDP)

## 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 :

tcp.port == 1500						
No.	Time	Source	Destination	Protocol	Length	Info
3	10.813237	127.0.0.1	127.0.0.1	TCP	68	52938 → 1500 [SYN, ECN, CWR] Seq=0 Win=65535 Len=0 MS
4	10.813300	127.0.0.1	127.0.0.1	TCP	68	1500 → 52938 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 M
5	10.813318	127.0.0.1	127.0.0.1	TCP	56	52938 → 1500 [ACK] Seq=1 Ack=1 Win=408288 Len=0 TSval
6	10.813336	127.0.0.1	127.0.0.1	TCP	56	[TCP Window Update] 1500 → 52938 [ACK] Seq=1 Ack=1 W
7	10.813539	127.0.0.1	127.0.0.1	TCP	58	52938 → 1500 [PSH, ACK] Seq=1 Ack=1 Win=408288 Len=2
8	10.813565	127.0.0.1	127.0.0.1	TCP	56	1500 → 52938 [ACK] Seq=1 Ack=3 Win=408288 Len=0 TSval
9	10.820827	127.0.0.1	127.0.0.1	TCP	58	1500 → 52938 [PSH, ACK] Seq=1 Ack=3 Win=408288 Len=2
10	10.820862	127.0.0.1	127.0.0.1	TCP	56	52938 → 1500 [ACK] Seq=3 Ack=3 Win=408288 Len=0 TSval
11	10.820916	127.0.0.1	127.0.0.1	TCP	56	52938 → 1500 [FIN, ACK] Seq=3 Ack=3 Win=408288 Len=0
12	10.820933	127.0.0.1	127.0.0.1	TCP	56	1500 → 52938 [ACK] Seq=3 Ack=4 Win=408288 Len=0 TSval

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 attente de donnees 1500
cliaddr.sin_family : 2
cliaddr.sin_addr.s_addr : 127.0.0.1
cliaddr.sin_port : 14019
./serveur : en attente de donnees 1500
_
```

Figure 3 : Partie du code client

```
ftime(&temps_debut); /*recuperation temps avnt l'envoi de la requete*/
rc = send(sd, "OK", sizeof(char)*2, 0);
printf("envoie demande echo %s\n", argv[1]);

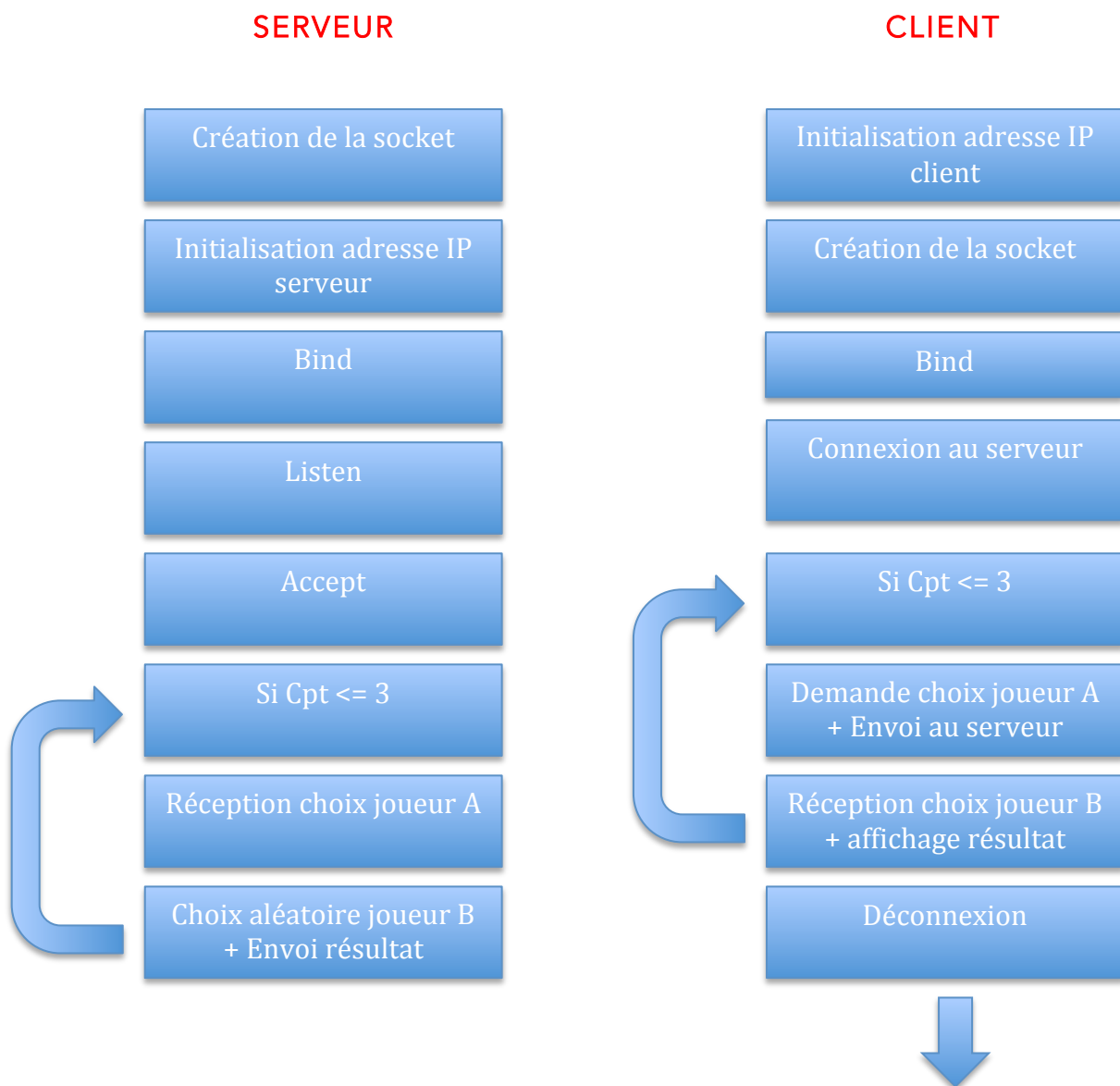
rc = recv(sd, line, sizeof(line) + 1, 0);
ftime(&temps_fin); /*recuperation temps aprées reception de la reponse*/
printf("la reception e la reponse a la requete a pris %lu seconds et %d ms\n", (temps_fin.time - temps_debut.time), (temps_fin.millitm - temps_debut.millitm));
close(sd);
return 0;
```

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 :

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	127.0.0.1	127.0.0.1	TCP	68	50172 → 1500 [SYN] Seq=0 Win=65535 Len=0 MSS=16344 WS=32
2	0.000072	127.0.0.1	127.0.0.1	TCP	68	1500 → 50172 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1
3	0.000088	127.0.0.1	127.0.0.1	TCP	56	50172 → 1500 [ACK] Seq=1 Ack=1 Win=408288 Len=0 TSval=523
4	0.000099	127.0.0.1	127.0.0.1	TCP	56	[TCP Window Update] 1500 → 50172 [ACK] Seq=1 Ack=1 Win=40
5	5.947148	127.0.0.1	127.0.0.1	TCP	68	50172 → 1500 [PSH, ACK] Seq=1 Ack=1 Win=408288 Len=12 TSv
6	5.947190	127.0.0.1	127.0.0.1	TCP	56	1500 → 50172 [ACK] Seq=1 Ack=13 Win=408288 Len=0 TSval=52
7	5.947226	127.0.0.1	127.0.0.1	TCP	156	1500 → 50172 [PSH, ACK] Seq=1 Ack=13 Win=408288 Len=100 T
8	5.947255	127.0.0.1	127.0.0.1	TCP	56	50172 → 1500 [ACK] Seq=13 Ack=101 Win=408192 Len=0 TSval=
9	7.082858	127.0.0.1	127.0.0.1	TCP	68	50172 → 1500 [PSH, ACK] Seq=13 Ack=101 Win=408192 Len=12
10	7.082902	127.0.0.1	127.0.0.1	TCP	56	1500 → 50172 [ACK] Seq=101 Ack=25 Win=408256 Len=0 TSval=
11	7.082931	127.0.0.1	127.0.0.1	TCP	156	1500 → 50172 [PSH, ACK] Seq=101 Ack=25 Win=408256 Len=100
12	7.082953	127.0.0.1	127.0.0.1	TCP	56	50172 → 1500 [ACK] Seq=25 Ack=201 Win=408096 Len=0 TSval=
13	8.379091	127.0.0.1	127.0.0.1	TCP	68	50172 → 1500 [PSH, ACK] Seq=25 Ack=201 Win=408096 Len=12
14	8.379130	127.0.0.1	127.0.0.1	TCP	56	1500 → 50172 [ACK] Seq=201 Ack=37 Win=408256 Len=0 TSval=
15	8.379159	127.0.0.1	127.0.0.1	TCP	156	1500 → 50172 [PSH, ACK] Seq=201 Ack=37 Win=408256 Len=100
16	8.379182	127.0.0.1	127.0.0.1	TCP	56	50172 → 1500 [ACK] Seq=37 Ack=301 Win=408000 Len=0 TSval=
17	8.379198	127.0.0.1	127.0.0.1	TCP	156	1500 → 50172 [PSH, ACK] Seq=301 Ack=37 Win=408256 Len=100
18	8.379209	127.0.0.1	127.0.0.1	TCP	56	50172 → 1500 [ACK] Seq=37 Ack=401 Win=407872 Len=0 TSval=
19	8.379251	127.0.0.1	127.0.0.1	TCP	56	50172 → 1500 [FIN, ACK] Seq=37 Ack=401 Win=407872 Len=0 T
20	8.379276	127.0.0.1	127.0.0.1	TCP	56	1500 → 50172 [ACK] Seq=401 Ack=38 Win=408256 Len=0 TSval=

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)

```
► Transmission Control Protocol, Src Port: 51594, Dst Port: 1500, Seq: 1, Ack: 1, Len: 12
0000  02 00 00 00 45 00 00 40 bd 5a 40 00 40 06 00 00  ....E..@ .Z@.@...
0010  7f 00 00 01 7f 00 00 01 c9 8a 05 dc f7 b1 e9 ad  ....C...1. .4.....
0020  d9 e4 63 87 80 18 31 d7 fe 34 00 00 01 01 08 0a  ..C...1. .4.....
0030  03 fe 00 c9 03 fd f4 41 01 00 00 00 01 00 00 00  .....A .....
0040  00 00 00 00                                     ....
```

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

```
► Transmission Control Protocol, Src Port: 1500, Dst Port: 51594, Seq: 1, Ack: 13, Len: 100
0000  02 00 00 00 45 02 00 98 d1 67 40 00 40 06 00 00  ....E... .g@.@...
0010  7f 00 00 01 7f 00 00 01 05 dc c9 8a d9 e4 63 87  ....C...C...
0020  f7 b1 e9 b9 80 18 31 d7 fe 8c 00 00 01 01 08 0a  ....1. ....
0030  03 fe 00 c9 03 fe 00 c9 4d 41 4e 43 48 45 20 6e  .... MANCHE n
0040  c2 b0 31 20 3a 20 63 6c 69 65 6e 74 20 22 66 65  ..1 : cl ient "fe
0050  75 69 6c 6c 65 22 20 56 53 20 20 73 65 72 76 65  uille" V S serve
0060  75 72 20 22 63 69 73 65 61 75 22 20 2d 2d 3e 20  ur "cise au" -->
0070  52 65 73 75 6c 74 61 74 20 73 65 72 76 65 75 72  Resultat  serveur
0080  20 67 61 67 6e 65 0a 00 00 00 00 00 00 00 00 00  gagne.. ....
0090  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ....
```