

Le but du TP est de relier le client au serveur et de les faire communiquer pour que le message s'écrit sur le serveur s'affiche chez le client via WireShark. Pour cette expérience il nous faut 4 étapes. L'étape suivante sera de désigner un client et un serveur, le but sera de communiquer avec notre voisin.

- 1\ Écrire le code du serveur :

```
// Librairie version 1
#include <winsock.h>
#pragma comment(lib, "wsock32.lib")

int main()
{
    // Initialisation des variables WSADATA; sock; csock; sin et csin
    WSADATA WSADATA;
    SOCKET sock;
    SOCKET csock;
    SOCKADDR_IN sin;
    SOCKADDR_IN csin;

    WSStartup(MAKEWORD(1, 0), &WSADATA); // 2 paramètres (1 pour la version 1)
    sock = socket(AF_INET, SOCK_STREAM, 0); // Création du socket
    sin.sin_addr.s_addr = INADDR_ANY; // Définit l'adresse du serveur
    sin.sin_family = AF_INET; // La "famille" du socket
    sin.sin_port = htons(1234); // Port 1234 où l'on se connecte
    bind(sock, (SOCKADDR*)&sin, sizeof(sin)); // la commande qui va attacher
    // votre socket directement au port et à l'adresse. Il y a trois paramètres. Le premier
    // est sock, le deuxième est la structure SOCKADDR_IN et le dernier est la taille de
    // cette structure : sizeof(sin).
    listen(sock, 0);
    while (1)
    {
        int sinsize = sizeof(csin);
        if ((csock = accept(sock, (SOCKADDR*)&csin, &sinsize)) != INVALID_SOCKET)
        {
            send(csock, "Hello world!\r\n", 14, 0);
            closesocket(csock);
        }
    }
    /* On devrait faire closesocket(sock); puis WSACleanup(); mais puisqu'on a entré
    une boucle infinie ... */
    return 0;
}
```

- 2\ Écrire le code client :

```
// Librairie version 1
#include <winsock.h>
#pragma comment(lib, "wsock32.lib")

int main()
{
    // Initialisation des variables WSADATA; sock et sin
    WSADATA WSADATA;
    SOCKET sock;
    SOCKADDR_IN sin;    // la struct du SOCKADDR contient les informations techniques
    du socket
    char buffer[255];
    WSASTartup(MAKEWORD(1, 0), &WSADATA);        // 2 paramètres (1 pour la version 1)
    sock = socket(AF_INET, SOCK_STREAM, 0);      // Création du socket
    sin.sin_addr.s_addr = inet_addr("127.0.0.1"); // Définit l'adresse du serveur
    sin.sin_family = AF_INET;                    // La "famille" du socket
    sin.sin_port = htons(1234);                  // Port 1234 où l'on se connecte
    connect(sock, (SOCKADDR*)&sin, sizeof(sin)); // Raccroche le socket au port
    et à l'adresse
    recv(sock, buffer, sizeof(buffer), 0);
    closesocket(sock);
    WSACleanup();    // Ferme la variable
    return 0;
}
```

Ici on aura donc le message « Hello world » écrit au client.

Avant de vérifier si le message s'est bien retranscrit nous devons faire une autre étape.

3\ Lancer les deux fichiers exécutables à l'aide de l'invité de commandes

À l'aide de deux invité de commandes, il nous faut lancer le fichier exécutable du client et le fichier exécutable du serveur comme ceci.

Ouverture du fichier exécutable pour le serveur :

```
Invite de commandes - serveur.exe

25/01/2022 09:26 <DIR> .
25/01/2022 09:26 <DIR> ..
25/01/2022 09:25      39 936 client.exe
25/01/2022 09:25      317 client.exe.recipe
25/01/2022 09:25     413 992 client.ilc
25/01/2022 09:25      19 343 client.obj
25/01/2022 09:25      1 110 016 client.pdb
25/01/2022 09:25 <DIR> client.tlog
25/01/2022 10:27      40 448 serveur.exe
25/01/2022 09:15      20 401 serveur.obj
25/01/2022 10:27      1 003 520 serveur.pdb
25/01/2022 09:04      318 Sockets.exe.recipe
25/01/2022 09:04     407 376 Sockets.ilc
25/01/2022 09:25     455 Sockets.log
25/01/2022 09:04      1 159 168 Sockets.pdb
25/01/2022 09:08 <DIR> Sockets.tlog
25/01/2022 09:17      76 Sockets.vcxproj.FileListAbsolute.txt
25/01/2022 09:04      20 392 Source.obj
25/01/2022 09:25     535 552 vc142.idb
25/01/2022 09:25     479 232 vc142.pdb
      16 fichier(s)      5 250 542 octets
      4 Rép(s)  35 838 337 024 octets libres

D:\dugelaye.SNIRW\JLMarquette\Sequence_03\Sockets\Sockets\Debug>serveur.exe
^C
D:\dugelaye.SNIRW\JLMarquette\Sequence_03\Sockets\Sockets\Debug>serveur.exe
```

Ouverture du fichier exécutable pour le client :

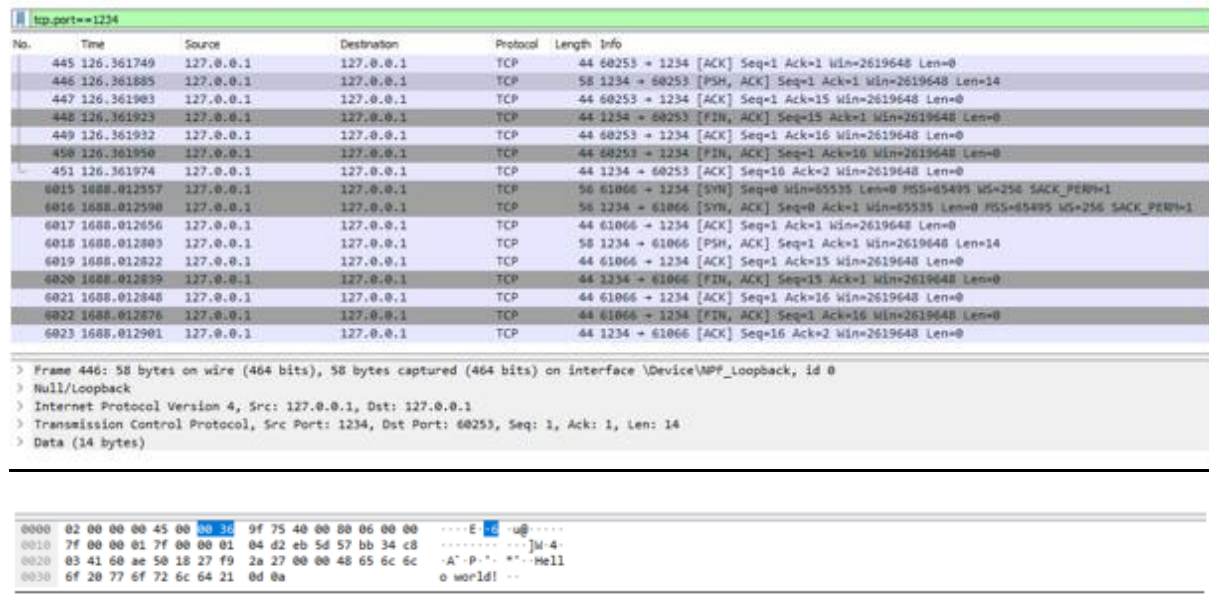
```
Invite de commandes

25/01/2022 09:26 <DIR> .
25/01/2022 09:26 <DIR> ..
25/01/2022 09:25      39 936 client.exe
25/01/2022 09:25      317 client.exe.recipe
25/01/2022 09:25     413 992 client.ilc
25/01/2022 09:25      19 343 client.obj
25/01/2022 09:25      1 110 016 client.pdb
25/01/2022 09:25 <DIR> client.tlog
25/01/2022 10:27      40 448 serveur.exe
25/01/2022 09:15      20 401 serveur.obj
25/01/2022 10:27      1 003 520 serveur.pdb
25/01/2022 09:04      318 Sockets.exe.recipe
25/01/2022 09:04     407 376 Sockets.ilc
25/01/2022 09:25     455 Sockets.log
25/01/2022 09:04      1 159 168 Sockets.pdb
25/01/2022 09:08 <DIR> Sockets.tlog
25/01/2022 09:17      76 Sockets.vcxproj.FileListAbsolute.txt
25/01/2022 09:04      20 392 Source.obj
25/01/2022 09:25     535 552 vc142.idb
25/01/2022 09:25     479 232 vc142.pdb
      16 fichier(s)      5 250 542 octets
      4 Rép(s)  35 838 337 024 octets libres

D:\dugelaye.SNIRW\JLMarquette\Sequence_03\Sockets\Sockets\Debug>client.exe
D:\dugelaye.SNIRW\JLMarquette\Sequence_03\Sockets\Sockets\Debug>client.exe
D:\dugelaye.SNIRW\JLMarquette\Sequence_03\Sockets\Sockets\Debug>client.exe
D:\dugelaye.SNIRW\JLMarquette\Sequence_03\Sockets\Sockets\Debug>
```

4\ À l'aide de wireshark on vérifie si le message a bien été transmis au client

Sur wireshark on se connecte au même port que dans le code, ici le port numéro 1234 puis on peut constater que le message « Hello worlg » du serveur a bien été transmis au client.



Wireshark packet capture showing a successful TCP connection and data transmission. The filter is 'tcp.port==1234'. The capture shows several packets, including a SYN exchange and a data segment (packet 446) containing the message 'Hello worlg'.

Packet 446 details:

- Frame 446: 58 bytes on wire (464 bits), 58 bytes captured (464 bits) on interface \Device\NPF_{...}, id 0
- Null/Loopback
- Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
- Transmission Control Protocol, Src Port: 1234, Dst Port: 60253, Seq: 1, Ack: 1, Len: 14
- Data (14 bytes)

Hex dump of packet 446 data:

```
0000 02 00 00 00 45 00 00 36 9f 75 40 00 80 06 00 00  ....E..u@....
0010 7f 00 00 01 7f 00 00 01 04 d2 eb 5d 57 bb 34 c8  ....]M-4-
0020 03 41 60 ae 50 18 27 f9 2a 27 00 00 48 65 6c 6c  .A.P..**..Hell
0030 6f 20 77 6f 72 6c 64 21 0d 0a                   o world! ..
```

5\ Communiquer avec son camarade

Cette étape consiste donc à communiquer avec son camarade comme fait précédemment. Pour cela nous devons changer l'adresse ip dans le code, l'adresse ip du serveur étant 172.17.50.156 nous allons changer cette ligne

```
sin.sin_addr.s_addr = inet_addr("127.0.0.1");
```

en

```
sin.sin_addr.s_addr = inet_addr("172.17.50.156");
```

en enlevant la ligne

```
sin.sin_addr.s_addr = INADDR_ANY;
```

Avant, il faut rajouter dans le code du serveur l'adresse du serveur qui se trouvait seulement dans le code du client et dans le code du client il faut rajouter le port qui se trouvait seulement dans le code du serveur. On obtient bien le message :

Capture en cours de network_name

FichierEditerVueAllerCaptureAnalyserStatistiquesTelephonieWirelessOutilsAide

tcp.port==1234

No.	Time	Source	Destination	Protocol	Length	Info
155	9.419078	172.17.50.155	172.17.50.156	TCP	66	52235 → 1234 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256
156	9.419202	172.17.50.156	172.17.50.155	TCP	66	1234 → 52235 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1
157	9.419877	172.17.50.155	172.17.50.156	TCP	60	52235 → 1234 [ACK] Seq=1 Ack=1 Win=2102272 Len=0
158	9.420210	172.17.50.156	172.17.50.155	TCP	68	1234 → 52235 [PSH, ACK] Seq=1 Ack=1 Win=2102272 Len=14
159	9.420253	172.17.50.156	172.17.50.155	TCP	54	1234 → 52235 [FIN, ACK] Seq=15 Ack=1 Win=2102272 Len=0
160	9.420455	172.17.50.155	172.17.50.156	TCP	60	52235 → 1234 [ACK] Seq=1 Ack=16 Win=2102272 Len=0
161	9.433885	172.17.50.155	172.17.50.156	TCP	60	52235 → 1234 [FIN, ACK] Seq=1 Ack=16 Win=2102272 Len=0
162	9.433966	172.17.50.156	172.17.50.155	TCP	54	1234 → 52235 [ACK] Seq=16 Ack=2 Win=2102272 Len=0

<

>

> Frame 158: 68 bytes on wire (544 bits), 68 bytes captured (544 bits) on interface \Device\NPF_{836D49D1-87FC-4F2F-A525-C94B1EC10921},

> Ethernet II, Src: HewlettP_34:86:28 (c8:d9:d2:34:86:28), Dst: HewlettP_34:d4:bd (c8:d9:d2:34:d4:bd)

> Internet Protocol Version 4, Src: 172.17.50.156, Dst: 172.17.50.155

> Transmission Control Protocol, Src Port: 1234, Dst Port: 52235, Seq: 1, Ack: 1, Len: 14

▼ Data (14 bytes)

Data: 53616c75742063686566621210d0a

[Length: 14]

<

>

0000 c8 d9 d2 34 d4 bd c8 d9 d2 34 86 28 08 00 45 00 ...4...4·(·E·

0010 00 36 23 25 40 00 00 06 00 00 ac 11 32 9c ac 11 ·6#%@...·2...·

0020 32 9b 04 d2 cc 0b e8 83 d5 46 92 0f 19 6c 50 18 2·...·F...1P·

0030 20 14 bd 82 00 00 53 61 6c 75 74 20 63 68 65 66 ·····Sa lut chef

0040 21 21 0d 0a !!..