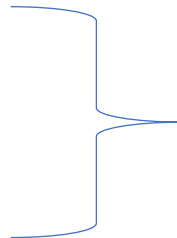


Rendu TP3

TP 3: Socket TCP/IPv4

Groupe 1 : - Zidani fahed Imed
- Riazi Ibrahim

Un serveur netcat
Client TCP Echo en Java
Client Avancé



[Voir code](#)

Serveur TCP

Fonctionnalité Echo

Le but du serveur est de faire un simple écho des lignes envoyées, précédé d'un >.

Réponse : [voir code](#)

Test Fonctionnels 1 :

Filter: **tcp.port==1234** Expression... Clear Apply Enregistrer

No.	Time	Source	Destination	Protocol	Length	Info
845	82.815997826	127.0.0.1	127.0.0.1	ILP	60	43474 → 1234 [ACK] Seq=1 Ack=1 Win=65536 Len=0 TSval=486185113 TSecr=622488392
814	91.748923103	127.0.0.1	127.0.0.1	TCP	70	43474 → 1234 [PSH, ACK] Seq=1 Ack=1 Win=65536 Len=4 TSval=486194106 TSecr=622488302
815	91.748923212	127.0.0.1	127.0.0.1	TCP	66	1234 → 43474 [ACK] Seq=1 Ack=5 Win=65536 Len=0 TSval=622497235 TSecr=486194106
1332	109.0948262	127.0.0.1	127.0.0.1	TCP	72	43474 → 1234 [PSH, ACK] Seq=5 Ack=1 Win=65536 Len=6 TSval=486211455 TSecr=622497235
1333	109.0948815	127.0.0.1	127.0.0.1	TCP	66	1234 → 43474 [ACK] Seq=1 Ack=11 Win=65536 Len=0 TSval=622514584 TSecr=486211455

Checksum: 0xff2c [unverified]
[Checksum Status: Unverified]
Urgent pointer: 0

Options: (12 bytes), No-Operation (NOP), No-Operation (NOP), Timestamps

- TCP Option - No-Operation (NOP)
- TCP Option - No-Operation (NOP)
- TCP Option - Timestamps: TSval 486194106, TSecr 622488302

Kind: Time Stamp Option (8)
Length: 10
Timestamp value: 486194106
Timestamp echo reply: 622488302

[SEQ/ACK analysis]
[Timestamps]

TCP payload (4 bytes)

Data (4 bytes)
Data: 0865780a
[Length: 4]

```

0010 00 38 d3 1a 40 00 00 06 68 a3 7f 00 00 01 7f 00 .8..@.@.h.....
0020 01 01 a9 d2 04 d2 70 4c 01 50 45 e0 96 43 80 18 .....PL..PE..C..
0030 02 00 ff 2c 00 00 01 01 08 0a 1c fa bb ba 25 1a .....%.....
0040 6a ee 08 65 79 0a j..hey.

```

Filter: **tcp.port==1234** Expression... Clear Apply Enregistrer

No.	Time	Source	Destination	Protocol	Length	Info
845	82.815997826	127.0.0.1	127.0.0.1	ILP	60	43474 → 1234 [ACK] Seq=1 Ack=1 Win=65536 Len=0 TSval=486185113 TSecr=622488392
814	91.748923103	127.0.0.1	127.0.0.1	TCP	70	43474 → 1234 [PSH, ACK] Seq=1 Ack=1 Win=65536 Len=4 TSval=486194106 TSecr=622488302
815	91.748923212	127.0.0.1	127.0.0.1	TCP	66	1234 → 43474 [ACK] Seq=1 Ack=5 Win=65536 Len=0 TSval=622497235 TSecr=486194106
1332	109.0948262	127.0.0.1	127.0.0.1	TCP	72	43474 → 1234 [PSH, ACK] Seq=5 Ack=1 Win=65536 Len=6 TSval=486211455 TSecr=622497235
1333	109.0948815	127.0.0.1	127.0.0.1	TCP	66	1234 → 43474 [ACK] Seq=1 Ack=11 Win=65536 Len=0 TSval=622514584 TSecr=486211455

Checksum: 0xff2e [unverified]
[Checksum Status: Unverified]
Urgent pointer: 0

Options: (12 bytes), No-Operation (NOP), No-Operation (NOP), Timestamps

- TCP Option - No-Operation (NOP)
- TCP Option - No-Operation (NOP)
- TCP Option - Timestamps: TSval 486211455, TSecr 622497235

Kind: Time Stamp Option (8)
Length: 10
Timestamp value: 486211455
Timestamp echo reply: 622497235

[SEQ/ACK analysis]
[Timestamps]

TCP payload (6 bytes)

Data (6 bytes)
Data: 60616865640a
[Length: 6]

```

0010 00 3a d3 1b 40 00 00 06 68 a0 7f 00 00 01 7f 00 ...@.@.h.....
0020 01 01 a9 d2 04 d2 70 4c 01 54 45 e0 96 43 80 18 .....PL..TE..C..
0030 02 00 ff 2e 00 00 01 01 08 0a 1c fa ff 7f 25 1a .....%.....
0040 8d d3 60 61 68 65 64 0a ..fahed.

```

serverTcp.java ClientTcp.java clientSecondQuestion.java

```

// serverTcp.java
import java.io.*;
import java.net.*;

public class serverTcp {

    public ServerSocket serverSocket;

    public serverTcp(int port) throws IOException {
        this.serverSocket = new ServerSocket(port);
    }

    private void listen() throws Exception {
        String data = null;
        Socket client = this.serverSocket.accept();
        String clientAddress = client.getInetAddress().getHostAddress();
        System.out.println("New connection from " + clientAddress);

        BufferedReader br = new BufferedReader(
            new InputStreamReader(client.getInputStream()));
        while ( (data = br.readLine()) != null ) {
            System.out.println("Message from " + clientAddress + " : " + data);
        }

        InetAddress getClientAddress() {return this.serverSocket.getInetAddress();}

        public int getPort() {return this.serverSocket.getPort();}

        public static void main(String[] args) throws Exception {
            serverTcp app = new serverTcp(1234);
            System.out.println("Listening Server");
            "hey" = app.getClientAddress().getHostAddress();
        }
    }
}

```

Running: serverTcp.java ClientTcp.java

```

C:\Users\user>java -cp . serverTcp
Connected to Server: 192.168.1.10:1234

```

Running: clientSecondQuestion.java

```

C:\Users\user>java -cp . clientSecondQuestion
Connected to Server: 192.168.1.10:1234

```

Running: clientTcp.java

```

C:\Users\user>java -cp . clientTcp
Connected to Server: 192.168.1.10:1234

```

Tester avec le client

Utiliser votre client pour tester votre serveur.

Que se passe-t-il si vous utilisez deux clients en même temps.

Expliquez.

Réponse :

Il est possible pour un client de demander un port TCP spécifique pour se connecter à l'aide de l'appel système `bind()` ; cependant, si deux clients demandent le même port, seule la première demande aboutira.

Autres Clients Echo

python

Écrire un deuxième client dans un autre langage de programmation, comme python.

Réponse : [voir code](#)

Test

Tester ce client avec le serveur java. En quoi ce test illustre-t-il le paradigme client-serveur?

Réponse :

- On lit essentiellement des données à partir d'un port ou on écrit des données sur un port. Notre application est agnostique à ce que l'autre côté est écrit.
- Ce test illustre que le seul couplage entre un client et un serveur est le protocole par lequel ils communiquent, pas le langage dans lequel ils sont écrits. Si on contrôle à la fois le client et le serveur, on peut dicter ce protocole.

Compatibilité UDP et TCP

Tester vos clients TCP avec le serveur UDP. Tester vos clients UDP avec le serveur TCP.

Réponse : on ne peut pas se connecter directement à un serveur tcp avec un client udp. Parce que les protocoles doivent correspondre