# FONDAMENTAUX DE LA COMMUNICATION UDP EN JAVA

MUNDUS LI YUANYUAN 27/05/2017

**Table de matières**

## 1.Application de Tchat

### 1.1. UML

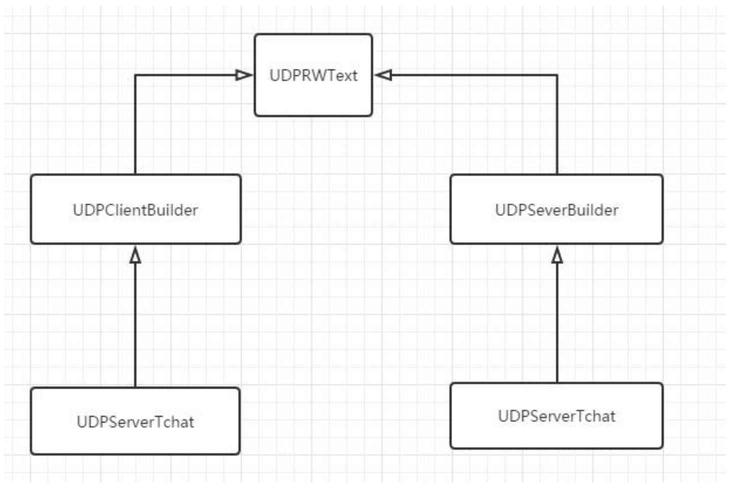


Diagramme de classe

### 1.2. Le code

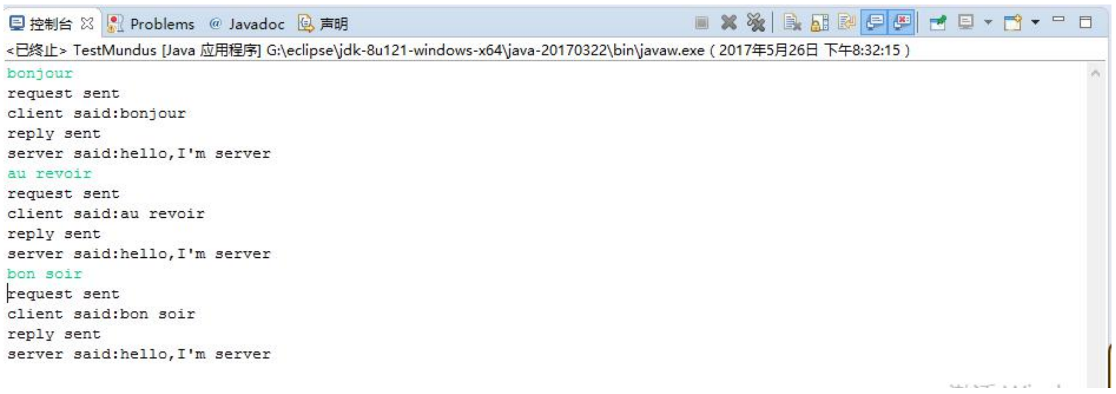
```
public class Test
{
        public static void main(String[] args)
        {
                new Thread(new UDPServerTchat()).start();
                new Thread(new UDPClientTchat()).start();
        }
}

public class UDPClientTchat extends UDPClientBuilder implements Runnable
{
        private Scanner sc;

        public void run()
        {
                try
                {
                        setConnection();
                        while(!s.isClosed())
                        {
                                /**Le client envoie le message.*/
                                sc = new Scanner(System.in);
                                String msg = sc.nextLine();
                                req = getTextSendingPacket(isA,msg,size);
                                s.send(req);
                        System.out.println("request sent");

                        /**Le client reçoit le message.*/
                        rep = getTextReceivingPacket(size);
                        s.receive(rep);
                        String receive=getMsg(rep);
                        System.out.println("server said:"+receive); //Affiche le message reçu du serveur
                    }
                }

                catch(IOException e)
                {
                    s.close();
                    System.out.println("IOException UDPClient");
                }
        }
}

public class UDPServerTchat extends UDPSeverBuilder implements Runnable
{
        public void run()
        {
                try
                {
                    setConnection();
                    String hello="hello,I'm server";
                    while(!s.isClosed())
                    {
                        /**Le serveur recoit le message du client.*/
                        req = getTextReceivingPacket(size);
                        s.receive(req);
                        String receive=getMsg(req);
                        System.out.println("client said:"+receive); //Affiche le message reçu du client

                        /**Le serveur envoie le message.*/
                        /**Si le serveur voudrait envoie l'autre message par "scanner".*/
                        //sc = new Scanner(System.in);
                        //String msg = sc.nextLine();
                        rep = getTextSendingPacket((InetSocketAddress) req.getSocketAddress(),hello,size);
                        //rep = getTextSendingPacket((InetSocketAddress) req.getSocketAddress(),msg,size);
                        s.send(rep);
                        System.out.println("reply sent");
                    }
        }
        catch(IOException e)
        {
            s.close();
            System.out.println("IOException UDPServerchat");
        }
    }
}

public class UDPClientBuilder extends UDPRWText
{
    InetSocketAddress isA; // L'adresse
    DatagramSocket s; // Le socket
    DatagramPacket req, rep; // Préparer la requête et le la réponse
    final int size = 2048;

    UDPClientBuilder()
    {
        isA = null; s = null; req = rep = null;
    }
    int times=3000;

    protected void setConnection() throws IOException
    {
        s = new DatagramSocket();
        isA = new InetSocketAddress("localhost",8080);
        /** On peut ajouter l'autre configuration … */
        //s.setSoTimeout(times);
    }
}

public class UDPSeverBuilder extends UDPRWText
{
    InetSocketAddress isA; // L'adresse
    DatagramSocket s; // Le socket
    DatagramPacket req, rep; // Préparer la requête et la réponse
   final int size = 2048;

    UDPSeverBuilder()
    {
        isA = null; s = null; req = rep = null;
    }
    int times=3000;
```

```
        protected void setConnection() throws IOException
        {
                isA = new InetSocketAddress("localhost",8080);
                s = new DatagramSocket(isA.getPort());
                /** On peut ajouter l'autre configuration … */
                //s.setSoTimeout(times);
        }
}

public class UDPRWText
{
        private byte[] sB; /** "Buffer array". */

         /** Créer un socket pour envoyer le message. */
         protected DatagramPacket getTextSendingPacket(InetSocketAddress isA, String msg, int size) throws
        IOException
         {
                 sB = toBytes(msg, new byte[size]);
                 return new DatagramPacket(sB,0,sB.length,isA.getAddress(),isA.getPort());
         }

         /** Créer un socket pour recevoir le message. */
         protected DatagramPacket getTextReceivingPacket(int size) throws IOException
         { return new DatagramPacket(new byte[size],size); }

         /** Ajouter le message à un paquet. */
         protected void setMsg(DatagramPacket dP, String msg) throws IOException
         { toBytes(msg, dP.getData()); }

         private byte[] toBytes(String msg, byte[] lbuf)
         {
                 array = msg.getBytes();
                 if(array.length < lbuf.length)
                         for(int i=0;i<array.length;i++)
                                 lbuf[i] = array[i];
                 return lbuf;
         }
         private byte[] array;

         /** Pour obtenir le message d' un paquet. */
         protected String getMsg(DatagramPacket dP)
         {
                 sB = dP.getData();
                 for(int i=0;i<sB.length;i++)
```

```
            {
                  if(sB[i] == 0)
                  { p = i; i = sB.length; }
            }
      return new String(dP.getData(),0,p);
      }
      private int p;
}
```

### 1.3. Le résultat

Par exemple :



控制台 ☒ Problems @ Javadoc 声明

<已终止> TestMundus [Java 应用程序] G:\eclipse\jdk-8u121-windows-x64\java-20170322\bin\javaw.exe ( 2017年5月26日 下午8:32:15 )

```
bonjour
request sent
client said:bonjour
reply sent
server said:hello,I'm server
au revoir
request sent
client said:au revoir
reply sent
server said:hello,I'm server
bon soir
request sent
client said:bon soir
reply sent
server said:hello,I'm server
```

## 2.Serveur temps

### 2.1. UML

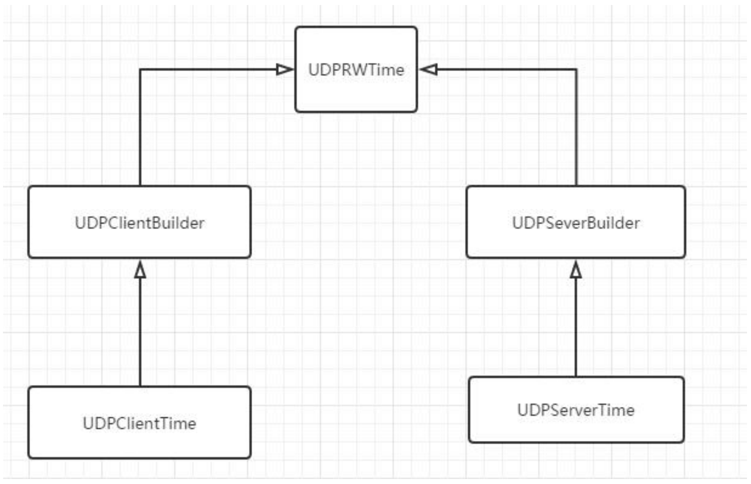


Diagramme de classe

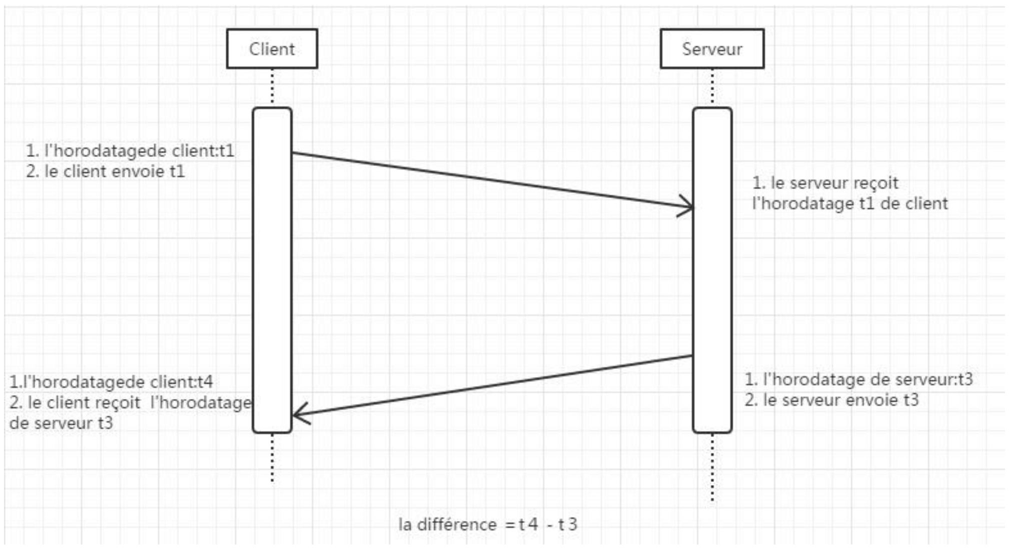

Client
Serveur
1. l'horodatagede client:t1
2. le client envoie t1
1. le serveur reçoit
l'horodatage t1 de client
1.l'horodatagede client:t4
2. le client reçoit l'horodatage
de serveur t3
1. l'horodatage de serveur:t3
2. le serveur envoie t3
la différence = t4 - t3

Diagramme de séquence

**2.2. Le code**

```
public class Test
{
        public static void main(String[] args)
        {
                new Thread(new UDPServerTime()).start();
                new Thread(new UDPClientTime()).start();
        }

}

public class UDPClientTime extends UDPClientBuilder implements Runnable
{
        private Scanner sc;
        public void run()
        {
                try
                {
                        setConnection();
                        while(!s.isClosed())
                        {
                                //Client envoie l’ horodatage
                                req = getTimeSendingPacket(isA,size);
                                s.send(req);
                                Long send=getTimeStamp();
                                System.out.println("client time sent t1:"+send);// Afficher l’ horodatage

                                //Client reçoit l’ horodatage de serveur
                                rep = getTimeReceivingPacket(size);
                                s.receive(rep);
                Long receivetime=getTimeStamp(rep);
                System.out.println("client reveive time t3:"+receivetime);

                //Calcule le decalage entre client et serveur
                Long t4=getTimeStamp();
                System.out.println("client time t4:"+t4);
                Long difference=t4-receivetime;//Le decalage entre client et serveur
                System.out.println("difference:"+difference);
                System.out.println("\n");
                Thread.sleep(3000);
            }
        }
        catch(IOException | InterruptedException e)
        {
            s.close();
            System.out.println("IOException UDPClient");
        }
    }
}

public class UDPServerTime extends UDPSeverBuilder implements Runnable
{
    private Scanner sc;
    public void run()
    {
        try
        {
            setConnection();
            while(!s.isClosed())
            {
                //Serveur reçoit l' horodatage de client
                req = getTimeReceivingPacket(size);
                s.receive(req);
                Long receivetime=getTimeStamp(req);
                System.out.println("server receive time t2:"+receivetime); //Afficher  l'horodatage de client

                //Serveur envoie l'horodatage
                rep = getTimeSendingPacket((InetSocketAddress) req.getSocketAddress(),size);
                s.send(rep);
                Long t3=getTimeStamp();
                System.out.println("server time sent t3:"+t3);// Afficher    l'horodatage de soi－même
            }
        }
        catch(IOException e)
        {
            s.close();
            System.out.println("IOException UDPServer");
        }
    }
}

public class UDPClientBuilder extends UDPRWTime
{
    InetSocketAddress isA;
    DatagramSocket s;
    DatagramPacket req, rep;
    final int size = 2048;

    UDPClientBuilder()
    { isA = null; s = null; req = rep = null; }

    int times=6000;
    protected void setConnection() throws IOException
    {
        s = new DatagramSocket();
        isA = new InetSocketAddress("localhost",8080);
        //s.setSoTimeout(times);
    }
}

public class UDPSeverBuilder extends UDPRWTime
{
    InetSocketAddress isA;
    DatagramSocket s;
    DatagramPacket req, rep;
     final int size = 2048;
    int times=3000;

    UDPSeverBuilder()
    { isA = null; s = null; req = rep = null; }

    protected void setConnection() throws IOException
    {
        isA = new InetSocketAddress("localhost",8080);
        s = new DatagramSocket(isA.getPort());
        //s.setSoTimeout(times);
    }
}
```

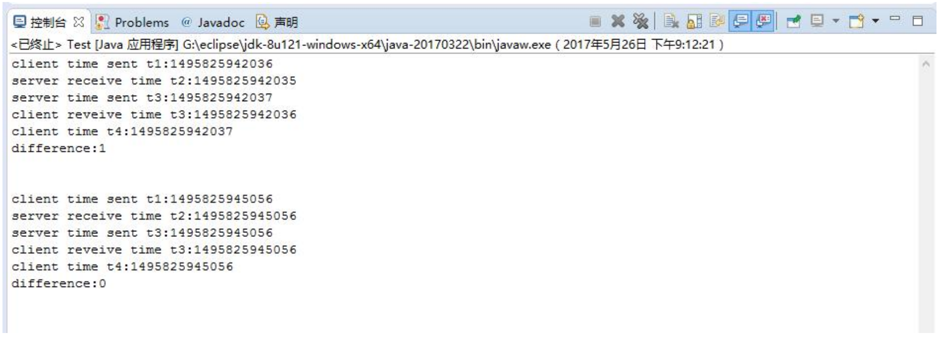
```
public class UDPRWTime
{
        private byte[] sB;
        private long tstamp; /**L' horodatage. */

        /** Obtenir le temps local. */
        protected long getLocalTime()
        {
                return System.nanoTime();
        }

        /** Bbtenir l' horodatage . */
        protected long getTimeStamp()
        {
                return System.currentTimeMillis();
        }

        /** Envoyer un socket avec l' horodatage. */
        protected DatagramPacket getTimeSendingPacket(InetSocketAddress isA, int size) throws IOException
        {
                tstamp = getTimeStamp(); sB = toBytes(tstamp, new byte[size]);
                return new DatagramPacket(sB,0,sB.length,isA.getAddress(),isA.getPort());
        }

          protected DatagramPacket getTimeReceivingPacket(int size) throws IOException
          {
                  return new DatagramPacket(new byte[size],size);
          }

        /** Configurer l' horodatage à un paquet . */
        protected void setTimeStamp(DatagramPacket dP)
        {
                tstamp = getTimeStamp(); sB = toBytes(tstamp, dP.getData());
        }

        private byte[] toBytes(long data, byte[] lbuf)
        {
                for(int i=0;i<8;i++)
                        lbuf[i] = (byte)((data >> (7-i)*8) & 0xff);
                return lbuf;
        }

        /** Obtenir l' horodatage d' un paquet . */
```

```
    protected long getTimeStamp(DatagramPacket dP)
    {
        return getLong(dP.getData());
    }

    private long getLong(byte[] by)
    {
        value = 0;
        for (int i = 0; i < 8; i++)
        {
            value = (value << 8) + (by[i] & 0xff);
        }
        return value;
    }
    private long value;
}
```

**2.3. Le résultat**

Par exemple ：

La différence＝client time t4－client receive time t3



```
<已终止> Test [Java 应用程序] G:\eclipse\jdk-8u121-windows-x64\java-20170322\bin\javaw.exe ( 2017年5月26日 下午9:12:21 )
client time sent t1:1495825942036
server receive time t2:1495825942035
server time sent t3:1495825942037
client reveive time t3:1495825942036
client time t4:1495825942037
difference:1


client time sent t1:1495825945056
server receive time t2:1495825945056
server time sent t3:1495825945056
client reveive time t3:1495825945056
client time t4:1495825945056
difference:0
```

**3.NTP**

**3.1.UML**

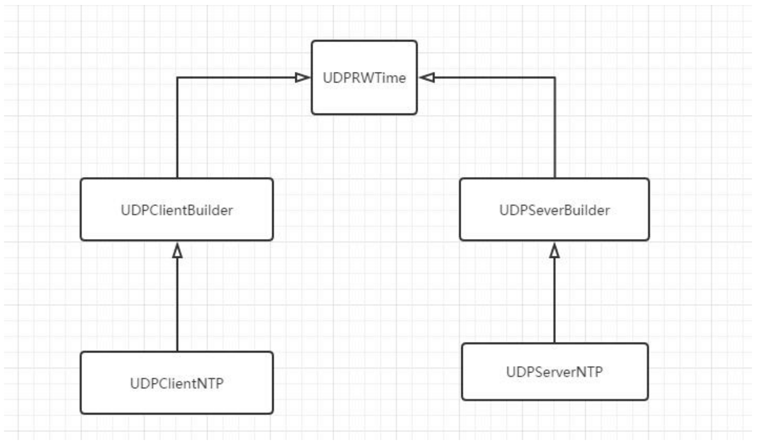


Diagramme de classe

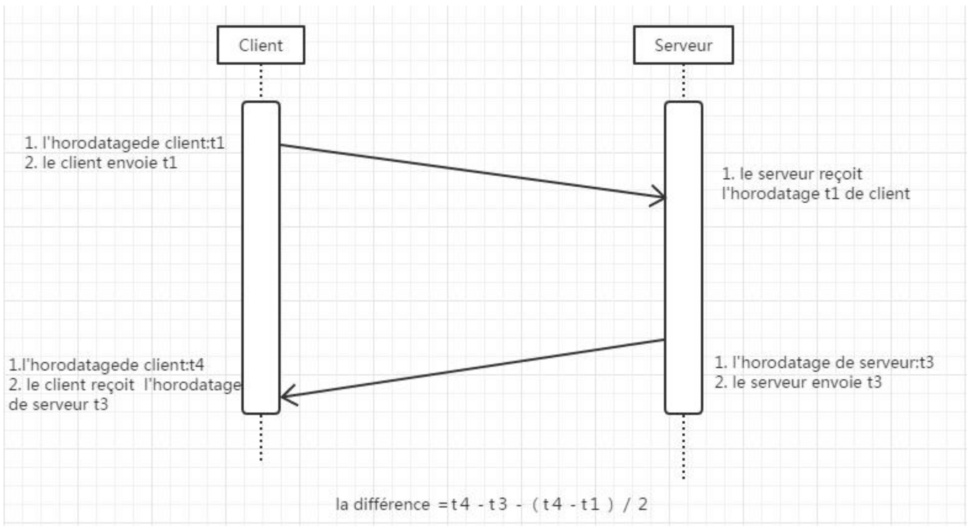
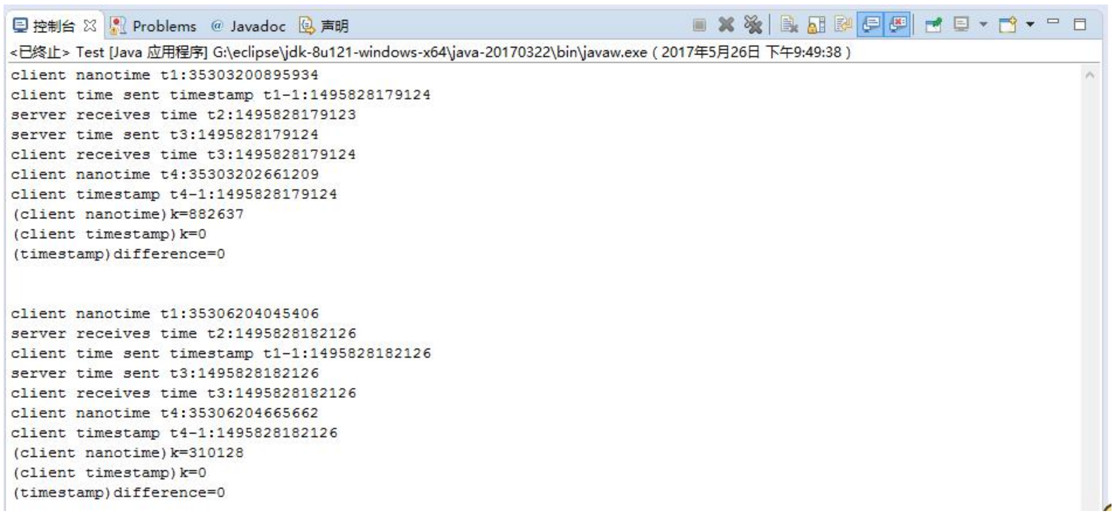

Client
Serveur
1. l'horodatagede client:t1
2. le client envoie t1
1. le serveur reçoit l'horodatage t1 de client
1.l'horodatagede client:t4
2. le client reçoit l'horodatage de serveur t3
1. l'horodatage de serveur:t3
2. le serveur envoie t3
la différence =t4 - t3 - (t4 - t1) / 2

Diagramme de séquence

**3.2. Le code**

```
public class UDPClientNTP extends UDPClientBuilder implements Runnable
{
        private Scanner sc;
        public void run()
        {
                try
                {
                        setConnection();
                        while(!s.isClosed())
                        {
                                //Client envoie l' horodatage vers serveur
                                Long t1=getLocalTime();
                                System.out.println("client nanotime t1:"+t1);
                                req = getTimeSendingPacket(isA,size);
                                s.send(req);
                                Long t11=getTimeStamp();
                                System.out.println("client time sent timestamp t1-1:"+t11);

                                //Client reçoit l' horodatage de serveur
                                rep = getTimeReceivingPacket(size);
                                s.receive(rep);
                                Long t3=getTimeStamp(rep);
                                System.out.println("client receives time t3:"+t3);

                                Long t4= getLocalTime();
                                System.out.println("client nanotime t4:"+t4);
                                Long t41=getTimeStamp();
                                System.out.println("client timestamp t4-1:"+t41);
                Long k=(t4-t1)/2;
                System.out.println("(client nanotime)k="+k);
                Long k1=(t41-t11)/2;//Le temps de transport
                System.out.println("(client timestamp)k="+k1);
                Long difference=t41-t3-k1;//Le decalage entre client et serveur
                System.out.println("(timestamp)difference="+difference);
                System.out.println("\n");
                Thread.sleep(3000);
            }
        }
        catch(IOException | InterruptedException e)
        {
            s.close();
            System.out.println("IOException UDPClient");
        }
    }
}

public class UDPServerNTP extends UDPSeverBuilder implements Runnable
{
    private Scanner sc;
    public void run()
    {
        try
        {
            setConnection();
            while(!s.isClosed())
            {
                //Serveur reçoit l' horodatage de client
                req = getTimeReceivingPacket(size);
                s.receive(req);
                Long receivetime=getTimeStamp(req);
                System.out.println("server receives time t2:"+receivetime);
                //Serveur envoie l' horodatage vers client
                rep = getTimeSendingPacket((InetSocketAddress) req.getSocketAddress(),size);
                s.send(rep);
                Long t3= getTimeStamp();
                System.out.println("server time sent t3:"+t3);
            }
        }
        catch(IOException e)
        {
            s.close();
            System.out.println("IOException UDPServer");
        }
    }
}
```

### 3.3. Le résultat



控制台 ☒ Problems @ Javadoc 声明

<已终止> Test [Java 应用程序] G:\eclipse\jdk-8u121-windows-x64\java-20170322\bin\javaw.exe ( 2017年5月26日 下午9:49:38 )

```
client nanotime t1:35303200895934
client time sent timestamp t1-1:1495828179124
server receives time t2:1495828179123
server time sent t3:1495828179124
client receives time t3:1495828179124
client nanotime t4:35303202661209
client timestamp t4-1:1495828179124
(client nanotime)k=882637
(client timestamp)k=0
(timestamp)difference=0


client nanotime t1:35306204045406
server receives time t2:1495828182126
client time sent timestamp t1-1:1495828182126
server time sent t3:1495828182126
client receives time t3:1495828182126
client nanotime t4:35306204665662
client timestamp t4-1:1495828182126
(client nanotime)k=310128
(client timestamp)k=0
(timestamp)difference=0
```

## 4.Communication UDP avancée

### 4.1 UML

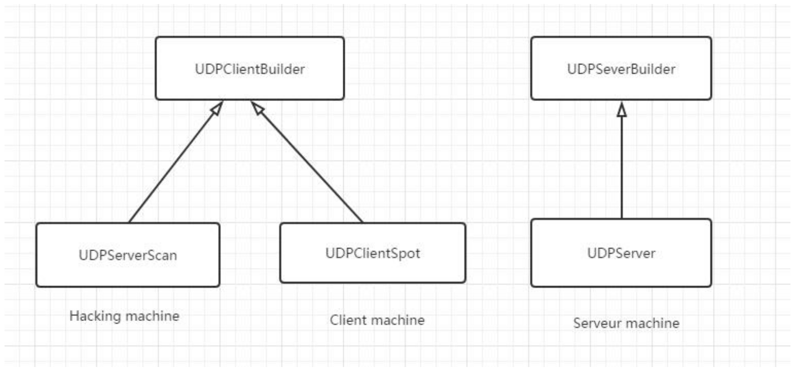


Diagramme de classe

### 4.2. Le code

```
public class Test
{
    public static void main(String[] args)
    {
        new Thread(new UDPServer()).start();
        new Thread(new ThreadPoolServer()).start();
        new Thread(new ThreadPoolClient()).start();
    }
}
```

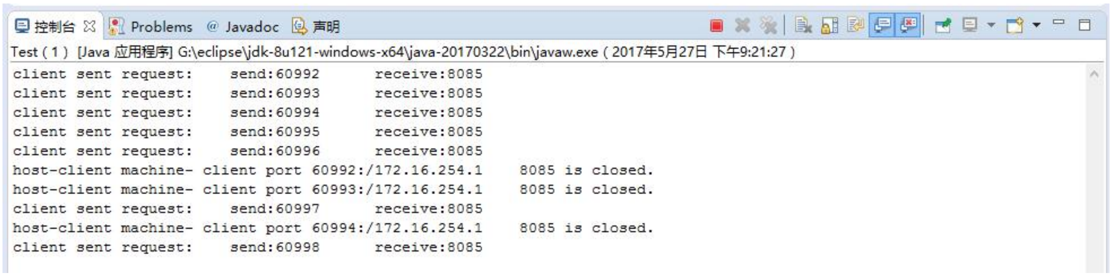
```
public class UDPServer extends UDPSeverBuilder implements Runnable // Serveur machine
{
        public void run()
        {
                try
                {
                        setConnection();
                        while(!s.isClosed())
                        {
                                // Reçoit le message
                                rep =    new DatagramPacket(new byte[size],size);
                                s.receive(rep);
                                /**System.out.println("serveur receive:"+"\t"+"send:"+rep.getPort()+"\t"+"receive:"
                                                        +s.getLocalPort());*/

                                // Envoie le message
                                req = new DatagramPacket(new byte[size],0,size,rep.getSocketAddress());
                                s.send(req);
                                /**System.out.println("server sent:"+"\t"+"send:"+s.getLocalPort() +"\t"+"receive:"
                                +req.getPort());*/
                        }
                        s.close();
                }
                catch (IOException e)
                {
                        s.close();
                        e.printStackTrace();
                }
        }
}

public class ThreadPoolClient implements Runnable    // Client machine
{
        public void run() {
                try {
                        while(true) {
                                new Thread(new UDPClientSpot()).start();
                                Thread.sleep(2000);
                        }
                }
                catch(Exception e)
                {
                        e.getStackTrace();
        }
    }
}

public class ThreadPoolServer implements Runnable // Hacking machine
{
    public void run() {
        try {
            while(true) {
                new Thread(new UDPServerScan()).start();
                Thread.sleep(2000);
            }
        }
        catch(Exception e)
        {
            e.getStackTrace();
        }
    }
}

public class UDPClientSpot extends UDPClientBuilder implements Runnable // Les clients envoient les requêtes
{
    public void run()
    {
        try
        {
            setClientConnection();
            req = new DatagramPacket(new byte[size],0,size,isA.getAddress(),isA.getPort());
            sClient.send(req);
            /**System.out.println("client sent request:"+"\t"+"send:"+sClient.getLocalPort()+"\t"+"receive:"
                            +req.getPort());*/
            Thread.sleep(5000); // Attend le serveur et hacking machine

            rep = new DatagramPacket(new byte[size],size);
            sClient.receive(rep);
            System.out.println("client receive reply:"+"\t"+"send:"+rep.getPort()+"\t"+"receive:"
                            +sClient.getLocalPort() );
            sClient.close();
        }
        catch(IOException | InterruptedException    e)
        {
            System.out.println("host-client machine- "+"client port "+sClient.getLocalPort()+":"
                            +req.getAddress()+"\t"+req.getPort()+" is closed.");
            sClient.close();
```

```
                e.getStackTrace();
            }
        }
}

public class UDPServerScan extends UDPClientBuilder implements Runnable
{
        public void run()
        {
            try
            {
                setServerConnection();
                req = new DatagramPacket(new byte[size],0,size,isA.getAddress(),isA.getPort());

                sClient.send(req); //Hacking machine envoie la requête vers le serveur
                //System.out.println("hacking client sent:"+"\t"+"send:"+sClient.getLocalPort()+"\t"
                                        +"receive:"+req.getPort());

                rep = new DatagramPacket(new byte[size],size);
                sClient.receive(rep); //Hacking machine reçoit la réponse du serveur
                /**System.out.println("hancking client receive:"+"\t"+"send:"+rep.getPort()+"\t"
                                        +"receive:"+sClient.getLocalPort()); */

                Random random2 = new Random();
                //La quatrième paramètre est l'adresse d'un des clients
                //La cinquième paramètre est le port d'un des clients
                InetSocketAddress isB=new InetSocketAddress("localhost",random2.nextInt(65535));
                req = new DatagramPacket(new byte[size],0,size,isB.getAddress(),isB.getPort());
                sServer.send(req);
                System.out.println("hacking server sent:"+"\t"+"send:"+sServer.getLocalPort()+"\t"
                                        +"receive:"+req.getPort());

                rep = new DatagramPacket(new byte[size],size);
                //Si le port du client est ouvert，le serveur de hacking machine peut reçoit la réponse du client
                sServer.receive(rep);
                System.out.println("hancking server receive:"+"\t"+"send:"+rep.getPort()+"\t"
                                        +"receive:"+sServer.getLocalPort());

                sServer.close();
                sClient.close();
                Thread.sleep(5000);
            }
            catch(IOException | InterruptedException e)
            {
                /**Si il y a non-retour de client ( le port n’est pas ouvert ), il y a une exception. Donc hacking
                                                machine peut sait quel port de client est ouvert . */
                System.out.println("host-hacking machine:"+"client port "+req.getPort()+" is closed.");
                sServer.close();
                sClient.close();
                e.getStackTrace();
            }
        }
}

public class UDPClientBuilder
{
        InetSocketAddress isA;
        DatagramSocket sClient,sServer;
        DatagramPacket req, rep;
        final int size = 2048;
        Random random = new Random();
        int port=random.nextInt(65535);

        UDPClientBuilder()
        { isA = null; sClient = null; req = rep = null; sServer=null;random=null;}

        int times=3000;
        protected void setClientConnection() throws IOException
        {
            // L’adresse et le port du serveur avec qui le client va connecter
            isA = new InetSocketAddress("172.16.254.1",8085);
            sClient = new DatagramSocket();
            sClient.setSoTimeout(times);
        }

        protected void setServerConnection()throws IOException
        {
            isA = new InetSocketAddress("172.16.254.1",8085); // L’adresse et le port du serveur
            sClient = new DatagramSocket(); // Hacking machine a un comportement client
            /** Les serveurs de hacking machine ouvrent sur des ports non spécifiés. */
            sServer = new DatagramSocket(port);
            sServer.setSoTimeout(times);
        }
}

public class UDPSeverBuilder
{
        InetSocketAddress isA;
```

```
    DatagramSocket s;
    DatagramPacket req, rep;
   final int size = 2048;

    UDPSeverBuilder()
    { isA = null; s = null; req = rep = null; }

    private int times=3000;
    protected void setConnection() throws IOException
    {
           isA = new InetSocketAddress("localhost",8085); // L'adresse et le port du serveur
           s = new DatagramSocket(isA.getPort());
           //s.setSoTimeout(times);
    }
}
```

**4.3. Le résultat**

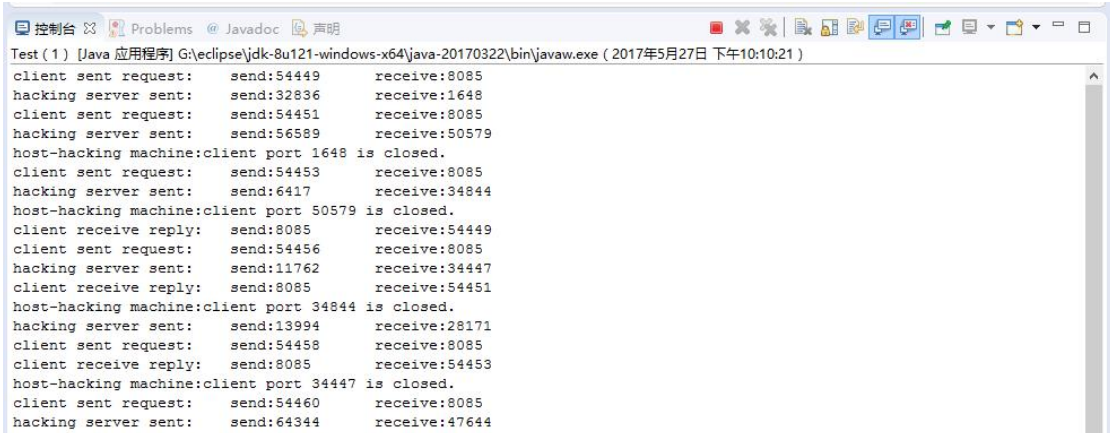
1) Juste le thread de client machine ouvre, l'addresse du serveur-172.16.254.1, le port du serveur-8085 :



```
Test ( 1 )  [Java 应用程序] G:\eclipse\jdk-8u121-windows-x64\java-20170322\bin\javaw.exe ( 2017年5月27日 下午9:21:27 )
client sent request:    send:60992      receive:8085
client sent request:    send:60993      receive:8085
client sent request:    send:60994      receive:8085
client sent request:    send:60995      receive:8085
client sent request:    send:60996      receive:8085
host-client machine- client port 60992:/172.16.254.1    8085 is closed.
host-client machine- client port 60993:/172.16.254.1    8085 is closed.
client sent request:    send:60997      receive:8085
host-client machine- client port 60994:/172.16.254.1    8085 is closed.
client sent request:    send:60998      receive:8085
```

2) Les trois threads ouvrent, donc hacking machine peut scanner les ports de clients, si le port du client est ouvert, hacking machine peut recevoir la réponse du client.

Par exemple: client machine, serveur machine, hacking machine sont sur un ordinateur. Donc l'adresse de ces trois sont "localhost".



```
Test ( 1 )  [Java 应用程序] G:\eclipse\jdk-8u121-windows-x64\java-20170322\bin\javaw.exe ( 2017年5月27日 下午10:10:21 )
client sent request:    send:54449      receive:8085
hacking server sent:    send:32836      receive:1648
client sent request:    send:54451      receive:8085
hacking server sent:    send:56589      receive:50579
host-hacking machine:client port 1648 is closed.
client sent request:    send:54453      receive:8085
hacking server sent:    send:6417       receive:34844
host-hacking machine:client port 50579 is closed.
client receive reply:   send:8085       receive:54449
client sent request:    send:54456      receive:8085
hacking server sent:    send:11762      receive:34447
client receive reply:   send:8085       receive:54451
host-hacking machine:client port 34844 is closed.
hacking server sent:    send:13994      receive:28171
client sent request:    send:54458      receive:8085
client receive reply:   send:8085       receive:54453
host-hacking machine:client port 34447 is closed.
client sent request:    send:54460      receive:8085
hacking server sent:    send:64344      receive:47644
```

3) Si on va mettre hacking machine sur une machine tierce.

Par exemple:

L'adresse de client machine: 172.168.1.2          L'adresse de serveur machine: 172.168.1.3     port:8090

L'adresse de hacking machine:172.168.1.4

Dans la classe UDPSeverBuilder:

```
protected void setConnection() throws IOException
{
        isA = new InetSocketAddress("localhost",8090);
        s = new DatagramSocket(isA.getPort());
        //s.setSoTimeout(times);
}
```

Dans la classe UDPClientBuilder:

```
protected void setClientConnection() throws IOException
{
        isA = new InetSocketAddress("172.168.1.3",8090);
        sClient = new DatagramSocket();
        sClient.setSoTimeout(times);
}

protected void setServerConnection()throws IOException
{
        isA = new InetSocketAddress("172.168.1.3",8090);
        sClient = new DatagramSocket();
        sServer = new DatagramSocket(port);
        sServer.setSoTimeout(times);
        //sServer.setSoTimeout(times);
}
```

Dans la classe UDPServerScan, pour la deuxième “req”:

```
InetSocketAddress isB=new InetSocketAddress("172.168.1.2",random2.nextInt(65535));
req = new DatagramPacket(new byte[size],0,size,isB.getAddress(),isB.getPort());
```