Rappels programmation réseau Java-suite

Socket programming

Two socket types for two transport services:

- UDP: unreliable datagram
- TCP: reliable, byte stream-oriented

Socket UDP

Classes

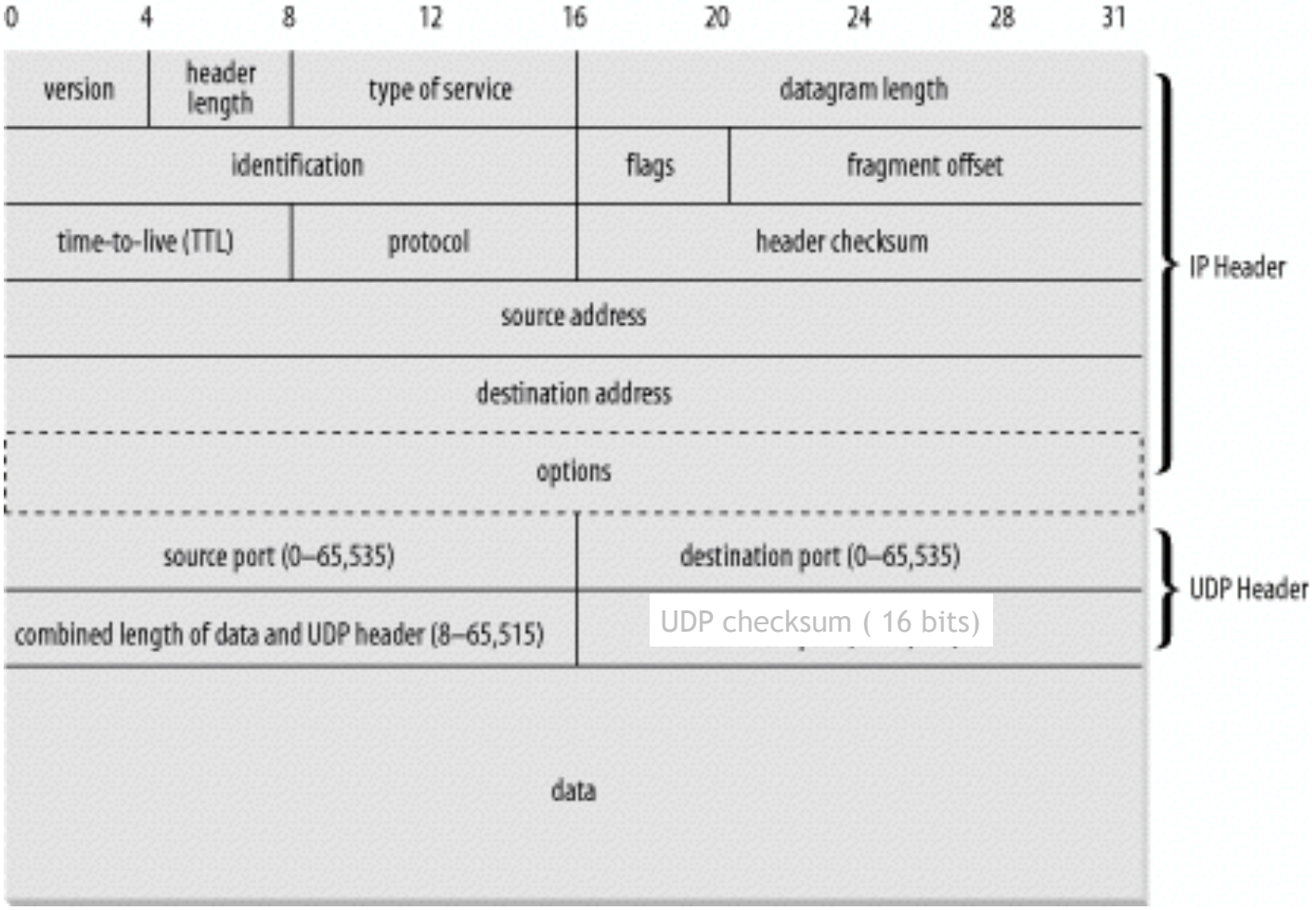
java.net.**DatagramPacket**

java.net.**DatagramSocket**

java.net.**MulticastSocket**

Socket UDP

UDP


0
4
8
12
16
20
24
28
31
version
header length
type of service
datagram length
identification
flags
fragment offset
time-to-live (TTL)
protocol
header checksum
source address
destination address
options
source port (0–65,535)
destination port (0–65,535)
combined length of data and UDP header (8–65,515)
UDP checksum (16 bits)
data
IP Header
UDP Header

DatagramPacket

- Un paquet contient au plus 65,507 bytes
- Pour construire les paquets pour recevoir

  public DatagramPacket(byte[] buffer, int length)
  public DatagramPacket(byte[] buffer, int offset, int length)

- Pour construire les paquets pour envoyer

  public DatagramPacket(byte[] data, int length, InetAddress destination, int port)
  public DatagramPacket(byte[] data, int offset, int length, InetAddress destination, int port)
  public DatagramPacket(byte[] data, int length, SocketAddress destination)
  public DatagramPacket(byte[] data, int offset, int length, SocketAddress destination)

Méthodes

- Adresses
  - public InetAddress getAddress( )
  - public int getPort( )
  - public SocketAddress getSocketAddress( )
  - public void setAddress(InetAddress remote)
  - public void setPort(int port)
  - public void setAddress(SocketAddress remote)

Méthodes (suite)

- Manipulation des données:
  - public byte[] getData( )
  - public int getLength( )
  - public int getOffset( )
  - public void setData(byte[] data)
  - public void setData(byte[] data, int offset, int length)
  - public void setLength(int length)

Exemple

```
import java.net.*;
public class DatagramExample {
  public static void main(String[] args) {
    String s = "Essayons.";
    byte[] data = s.getBytes( );
    try {
      InetAddress ia = InetAddress.getByName("www.liafa.univ-paris-diderot.fr");
      int port =7;
      DatagramPacket dp = new DatagramPacket(data, data.length, ia, port);
      System.out.println(" Un packet pour" + dp.getAddress( ) + " port " +
       dp.getPort( ));
      System.out.println("il y a " + dp.getLength( )  +
                                 " bytes dans le  packet");
      System.out.println(
        new String(dp.getData( ), dp.getOffset( ), dp.getLength( )));
    }
    catch (UnknownHostException e) {
      System.err.println(e);
    }
  }
}
```

DatagramSocket

- Constructeurs
  - public DatagramSocket( ) throws SocketException
  - public DatagramSocket(int port) throws SocketException
  - public DatagramSocket(int port, InetAddress interface) throws SocketException
  - public DatagramSocket(SocketAddress interface) throws SocketException
  - (protected DatagramSocket(DatagramSocketImpl impl) throws SocketException)

Exemple

```
java.net.*;
public class UDPPortScanner {
  public static void main(String[] args) {

    for (int port = 1024; port <= 65535; port++) {
      try {
        // exception si utilisé
        DatagramSocket server = new DatagramSocket(port);
        server.close( );
      }
      catch (SocketException ex) {
        System.out.println("Port occupé" + port + ".");
      } // end try
    } // end for
  }
}
```

Envoyer et recevoir

- public void send(DatagramPacket dp) throws IOException
- public void receive(DatagramPacket dp) throws IOException

Un exemple: Echo

- UDPServeur
  - UDPEchoServeur
- UDPEchoClient
  - SenderThread
  - ReceiverThread

Echo: UDPServeur

```
import java.net.*;
import java.io.*;
public abstract class UDPServeur extends Thread {
  private int bufferSize;
  protected DatagramSocket sock;
  public UDPServeur(int port, int bufferSize)
   throws SocketException {
    this.bufferSize = bufferSize;
    this.sock = new DatagramSocket(port);
  }
  public UDPServeur(int port) throws SocketException {
    this(port, 8192);
  }
  public void run() {
    byte[] buffer = new byte[bufferSize];
    while (true) {
      DatagramPacket incoming = new DatagramPacket(buffer, buffer.length);
      try {
        sock.receive(incoming);
        this.respond(incoming);
      }
      catch (IOException e) {
        System.err.println(e);
      }
    } // end while
  }
 public abstract void respond(DatagramPacket request);
}
```

UDPEchoServeur

```
public class UDPEchoServeur extends UDPServeur {
    public final static int DEFAULT_PORT = 2222;
    public UDPEchoServeur() throws SocketException {
        super(DEFAULT_PORT);
    }
    public void respond(DatagramPacket packet) {
        try {
            byte[] data = new byte[packet.getLength()];
            System.arraycopy(packet.getData(), 0, data, 0, packet.getLength());
            try {
                String s = new String(data, "8859_1");
                System.out.println(packet.getAddress() + " port "
                        + packet.getPort() + " reçu " + s);
            } catch (java.io.UnsupportedEncodingException ex) {}
            DatagramPacket outgoing = new DatagramPacket(packet.getData(),
                    packet.getLength(), packet.getAddress(), packet.getPort());
            sock.send(outgoing);
        } catch (IOException ex) {
            System.err.println(ex);
        }
    }
}
```

Client: UDPEchoClient

```
public class UDPEchoClient {
  public static void lancer(String hostname, int port) {
    try {
      InetAddress ia = InetAddress.getByName(hostname);
      SenderThread sender = new SenderThread(ia, port);
      sender.start();
      Thread receiver = new ReceiverThread(sender.getSocket());
      receiver.start();
    }
    catch (UnknownHostException ex) {
      System.err.println(ex);
    }
    catch (SocketException ex) {
      System.err.println(ex);
    }

  }  // end lancer
}
```

SenderThread

```
public class SenderThread extends Thread {
    private InetAddress server;
    private DatagramSocket socket;
    private boolean stopped = false;
    private int port;
    public SenderThread(InetAddress address, int port)
    throws SocketException {
        this.server = address;
        this.port = port;
        this.socket = new DatagramSocket();
        this.socket.connect(server, port);
    }
    public void halt() {
        this.stopped = true;
    }
//…
```

SenderThread

```
//…
    public DatagramSocket getSocket() {
        return this.socket;
    }
    public void run() {

        try {
            BufferedReader userInput = new BufferedReader(new     InputStreamReader(System.in));
            while (true) {
                if (stopped) return;
                String theLine = userInput.readLine();
                if (theLine.equals(".")) break;
                byte[] data = theLine.getBytes();
                DatagramPacket output
                        = new DatagramPacket(data, data.length, server, port);
                socket.send(output);
                Thread.yield();
            }
        }  // end try
        catch (IOException ex) {System.err.println(ex); }
    }  // end run
}
```

ReceiverThread

```
class ReceiverThread extends Thread {
    DatagramSocket socket;
    private boolean stopped = false;
    public ReceiverThread(DatagramSocket ds) throws SocketException {
        this.socket = ds;
    }
    public void halt() {
        this.stopped = true;
    }
    public DatagramSocket getSocket(){
        return socket;
    }
    public void run() {
        byte[] buffer = new byte[65507];
        while (true) {
            if (stopped) return;
            DatagramPacket dp = new DatagramPacket(buffer, buffer.length);
            try {
                socket.receive(dp);
                String s = new String(dp.getData(), 0, dp.getLength());
                System.out.println(s);
                Thread.yield();
            } catch (IOException ex) {System.err.println(ex); }
        }
    }
}
```

Autres méthodes

- public void close( )
- public int getLocalPort( )
- public InetAddress getLocalAddress( )
- public SocketAddress getLocalSocketAddress( )
- public void connect(InetAddress host, int port)
- public void disconnect( )
- public int getPort( )
- public InetAddress getInetAddress( )
- public InetAddress getRemoteSocketAddress( )

Options

- SO_TIMEOUT
  - public synchronized void setSoTimeout(int timeout) throws SocketException
  - public synchronized int getSoTimeout( ) throws IOException
- SO_RCVBUF
  - public void setReceiveBufferSize(int size) throws SocketException
  - public int getReceiveBufferSize( ) throws SocketException
- SO_SNDBUF
  - public void setSendBufferSize(int size) throws SocketException
  - int getSendBufferSize( ) throws SocketException
- SO_REUSEADDR (plusieurs sockets sur la même adresse)
  - public void setReuseAddress(boolean on) throws SocketException
  - boolean getReuseAddress( ) throws SocketException
- SO_BROADCAST
  - public void setBroadcast(boolean on) throws SocketException
  - public boolean getBroadcast( ) throws SocketException

Multicast socket (UDP)

- public class **MulticastSocket** extends DatagramSocket
- Constructeur:
  - MulticastSocket()
  - MulticastSocket(int port)

- Groupe formé sur une adresse IP de classe D
- Classe D: entre 224.0.0.0 et 255.255.255.255)
- Adresse 224.0.0.0 réservée
- Méthodes gestion groupe
  - void joinGroup(InetAddress mcastaddr)
  - void leaveGroup(InetAddress mcastaddr)

Exemple

```
InetAddress multicastAddress ; // Une adresse IP speciale
MulticastSocket socket ;

        /* creation: */
        socket = new MulticastSocket (port) ;

        /* Adresse IP multicast pour envoyer dans le reseau
local : */
        multicastAddress = InetAddress.getByName
("230.1.1.66") ;

        /* Indiquer qu'on veut recevoir les paquets  a
destination de cette adresse de groupe : */
        socket.joinGroup (multicastAddress) ;
```

Exemple (suite)

```
    ByteBuffer b = ByteBuffer.allocate(1400) ;
    String msg = "envoi" ; b.put (msg.getBytes()) ;
    b.flip () ; /* limit devient la position courante et  position est mis a 0 */

    /* Le paquet : Une adresse IP, un port et des octets... */
    DatagramPacket datagram = new DatagramPacket (b.array(), b.limit()) ;
    SocketAddress dest = new InetSocketAddress (multicastAddress, port) ;
    datagram.setSocketAddress (dest) ;
    try {
        socket.send (datagram) ;
   } catch (IOException e) { System.err.println (e) ; }
}
```