User Data Protocol

Protocollo UDP

User Data Protocol

- non affidabile (può perdere pacchetti)
- ricezione pacchetti non ordinati
- senza connesione
- punto punto o multicast
- molto piu' efficiente di TCP
- utilizzato quando la velocità è più importante dell’errore provocato dalla perdita di pacchetti
  - e.g.: streaming audio e video

UDP Datagram e Socket

La comunicazioni con protocollo UDP si volge in due fasi:

- costruzione del pacchetto di spedizione/ricezione
- invio/ricezione del pacchetto sul socket

Conversione String - byte[]

Da String a byte[]:

```
byte[] buf
String messaggio = new String("ciao a tutti");

buf = messaggio.getBytes();
```

Da byte[] a String:

```
String messaggio2;

messaggio2 = new String(buf, 0, buf.length);
```

Ricezione su socket UDP

```
byte[] buf = new byte[65507];
int portaloc = 2000;

DatagramPacket dp = new DatagramPacket(buf, 65507);

try {
    DatagramSocket so = new DatagramSocket(portaloc);

    so.receive(dp);
}
catch(BindException e) {
    System.out.println("porta locale udp occupata");}
catch(SocketException e) {
    System.out.println("errore sul socket udp");}
catch(IOException e) {
    System.out.println("errore di IO sulla receive");}
```

64Kb – intestazione pacchetto

Creazione pacchetto ricezione

Creazione socket

Porta locale

Ricezione bloccante del pacchetto dal socket

Timeout su ricezione

```
try {
     DatagramSocket so = new DatagramSocket(portaloc);

     so.setSoTimeout(1000);

     so.receive(dp);

}
catch ........

catch(SocketTimeoutException e){
     System.out.println("timeout");}

catch ........
```

Intervallo di timeout sul socket in ricezione

Ricezione pacchetto dal socket entro 1000 millisec

Messagio ricevuto

```
try {
    .....

    so.receive(dp);

    String messaggio =
      new String(dp.getData(), 0, dp.getLength());

    System.out.println("datagram ricevuto " + messaggio +
        " dalla porta remota " + dp.getPort() +
        " host remoto " + dp.getAddress().getHostName());

    System.out.println("socket porta locale " +
        so.getLocalPort() + " host locale " +
        so.getLocalAddress().getHostName() +
        " il timeout era " + so.getSoTimeout());
}
```

Spedizione su socket UDP

```
int portarem = 2000;
byte[] buf = new String("ciao a tutti!").getBytes();

try {
    InetAddress host = InetAddress.getLocalHost();

    DatagramPacket dp =
      new DatagramPacket(buf, buf.length, host, portarem);

    DatagramSocket so = new DatagramSocket();

    so.send(dp);
}
catch(UnknownHostException e){...}
catch(SocketException e) {...}
catch(IOException e) {...}
```

max 65457 bytes = 64Kb – intestazione pacchetto

Host e porta destinatario

non necessario specificare la porta locale

Spedizione su socket UDP

```
int portarem = 2000;
int portaloc = 3000;
byte[] buf = new String("ciao a tutti!").getBytes();
try {
    InetAddress host = InetAddress.getLocalHost();

    DatagramPacket dp =
      new DatagramPacket(buf, buf.length, host, portarem);

    DatagramSocket so = new DatagramSocket(portaloc);

    so.send(dp);
}
catch(UnknownHostException e){...}
catch(SocketException e) {...}
catch(IOException e) {...}
```

max 65457 bytes = 64Kb – intestazione pacchetto

si puo' specificare la porta locale

Spedizione su socket UDP

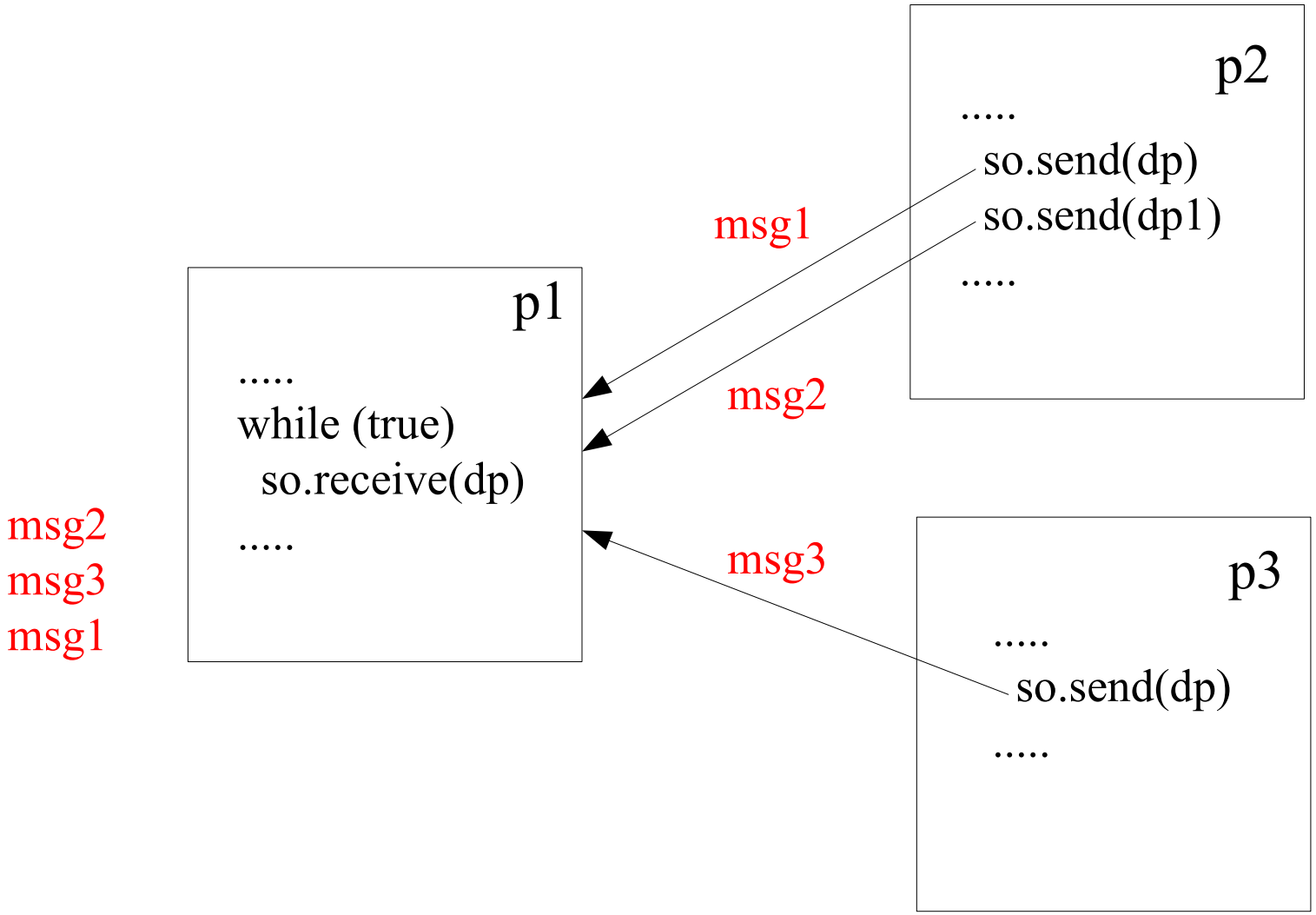
```
byte[] buf;
byte[] buf2;

buf = (new String("ciao a tutti")).getBytes();
buf2 = (new String("ciao a nessuno")).getBytes();

DatagramPacket dp = new DatagramPacket(buf, buf.length);

dp.setData(buf2);
dp.setLength(buf2.length);
dp.setAddress(host);
dp.setPort(porta);
```

Messaggio inviato

```
String msg = new String(dp.getData(),0,dp.getLength());

System.out.println("datagram spedito " + msg +
    " alla porta remota " + dp.getPort() +
    " dell'host remoto " + dp.getAddress().getHostName());


System.out.println("porta locale " + so.getLocalPort()+
    " dell'host locale " +
    so.getLocalAddress().getHostName());
```

Chiusura Socket

```
.......

so.close();


System.out.println("il socket e' chiuso? " +
    so.isClosed());
```


p2
.....
so.send(dp)
so.send(dp1)
.....
msg1
msg2
p1
.....
while (true)
so.receive(dp)
.....
msg2
msg3
msg1
msg3
p3
.....
so.send(dp)
.....