Transmission Control Protocol

Descrizione livelli TCP/IP

- fisico: hardware e protocollo scheda di rete
- rete: protocollo IP
- trasporto: protocolli TCP e UDP
- applicazione: telnet, ssh, i vostri programmi

Protocollo TCP

- **Transmission Control Protocol**
  - affidabile (3way handshake)
  - ordinamento pacchetti
  - orientato alla connessione
  - punto punto (no broadcast/multicast)
  - full duplex

Server e Client Socket

- Socket per client:

  si connette ad un socket per server

  `Socket`

- Socket per server:

  attende la connessione di un socket per client

  `ServerSocket`

Apertura Client Socket

```
InetAddress serverRem = null;
InetAddress localH = null;

try {
    serverRem =
            InetAddress.getByName("fujih5.cli.di.unipi.it");

    localH = InetAddress.getLocalHost();

}
catch(UnknownHostException e) {
    System.err.println("Host sconosciuto");
}
```

Apertura Client Socket (2)

```
Socket s = null;
int portaRem = 22;

try {
    s = new Socket(serverRem, portaRem);
    }
catch(ConnectException e) {
    System.out.println("Porta remota " + portaRem +
                " su " + serverRem + " non disponibile");
    }
catch (NoRouteToHostException e) {
    System.out.println("Connessione a" + serverRem +
                                    " non disponibile");
    }
catch (IOException e) {
    System.out.println("Errore di IO");
    }
```

Host e porta remoti

Apertura Client Socket (3)

```
Socket s = null;
int portaRem = 22;
int portaLoc = 3333;

try {
    s = new Socket(serverRem, portaRem, localH, portaLoc);
    }
catch(ConnectException e) {
    System.out.println("Porta remota " + portaRem +
                  " su " + serverRem + " non disponibile");}

catch(BindException e) {
    System.out.println("Porta locale " + portaRem +
         " su interfaccia" + localH + " non disponibile");}

catch(NoRouteToHostException e) {
    System.out.println("Connessione a" + serverRem +
                                  " non disponibile");}
catch(IOException e) {
    System.out.println("Errore di IO");}
```

Host e porta remoti

Host e porta locali

Client Socket (cont)

```
try
    {
    s.setSoTimeout(1000); // 1 sec

    System.out.println("socket sulla porta locale " +
        s.getLocalPort() + " dell'interfaccia locale " +
        s.getLocalAddress().getHostName() +
        " connesso con la porta remota " + s.getPort() +
        " dell'host remoto " +
        s.getInetAddress().getHostName());
    }
catch (SocketException e) {
    System.out.println("Errore nell'impostare il timeout");
    }
```

Lettura su Client Socket

```
InputStream is = null;
byte[] buf = new byte[100];

try
     {
     is = s.getInputStream();

     if (!s.isInputShutdown())
     {
          int bytesLetti = is.read(buf, 0, 100);
          System.out.println(new String(buf, 0, bytesLetti));

          s.shutdownInput();
     }
}
catch (SocketTimeoutException e) {
     System.out.println("timeout scaduto");
}
catch (IOException e) {
     System.out.println("errore sulla read");
}
```

Lettura su Client Socket (2)

```
try {

    InputStreamReader r = new
                        InputStreamReader(s.getInputStream());

    int car = r.read();

    }
catch(IOException e) {
    System.out.println("Errore di IO");
    }
```

Scrittura su Client Socket

```
OutputStream os = null;
byte[] buf = new String("messaggio in uscita").getBytes();

try
    {
        os = s.getOutputStream();

        if (!s.isOutputShutdown())
        {
            os.write(buf, 0, buf.length);

            s.shutdownOutput();
        }
}
catch (IOException e) {
    System.out.println("errore sulla write");
}
```

Scrittura su Client Socket (2)

```
String str = "ciao";

try {

    OutputStreamWriter w = new
                    OutputStreamWriter(s.getOutputStream());

    w.write(str, 0, str.length());

}

catch(IOException e) {
    System.out.println("errore di IO");
}
```

Chiusura Client Socket

```
try {

    if (!s.isClosed())
        s.close();

}
catch (IOException e) {
    System.out.println("errore sulla close");
}
```

Server Socket

```
ServerSocket ss;
Socket so;
int portaLoc = 2000;

try {
    ss = new ServerSocket(portaLoc);

    so = ss.accept();

    System.out.println("accettata connessione da " +
        so.getInetAddress().getHostName() +
        " su porta locale " + so.getLocalPort());

    OutputStream os = so.getOutputStream();
}
catch(BindException e) {
    System.out.println("errore di Bind");}
catch(IOException e) {
    System.out.println("errore di IO");}
```

Porta su cui attende le connessioni

Server Socket

```
ServerSocket ss;
Socket so;
int portaLoc = 2000;
int lCoda = 10;

try {
    ss = new ServerSocket(portaLoc, lCoda);

    so = ss.accept();

    System.out.println("accettata connessione da " +
        so.getInetAddress().getHostName() +
        " su porta locale " + so.getLocalPort());

    OutputStream os = so.getOutputStream();
}
catch(BindException e) {
    System.out.println("errore di Bind");}
catch(IOException e) {
    System.out.println("errore di IO");}
```

Lunghezza coda connesioni in attesa

Server Socket

```
ServerSocket ss;
Socket so;
int portaLoc = 2000;
int lCoda = 10;

try {
    ss = new ServerSocket(portaLoc, lCoda, localH);

    so = ss.accept();

    System.out.println("accettata connessione da " +
        so.getInetAddress().getHostName() +
        " su porta locale " + so.getLocalPort());

    OutputStream os = so.getOutputStream();
}
catch(BindException e) {
    System.out.println("errore di Bind");}
catch(IOException e) {
    System.out.println("errore di IO");}
```

Interfaccia di rete su cui aprire il socket

Chiusura Server Socket

```
try {

    if (!ss.isClosed())
        ss.close();

}
catch (IOException e) {
    System.out.println("errore sulla close");
}
```

Oggetti Serializzabili

```
import java.io.*;

public class pippo implements Serializable {

   int intero;
   double doppio;

   // costruttore e metodi vari

}
```

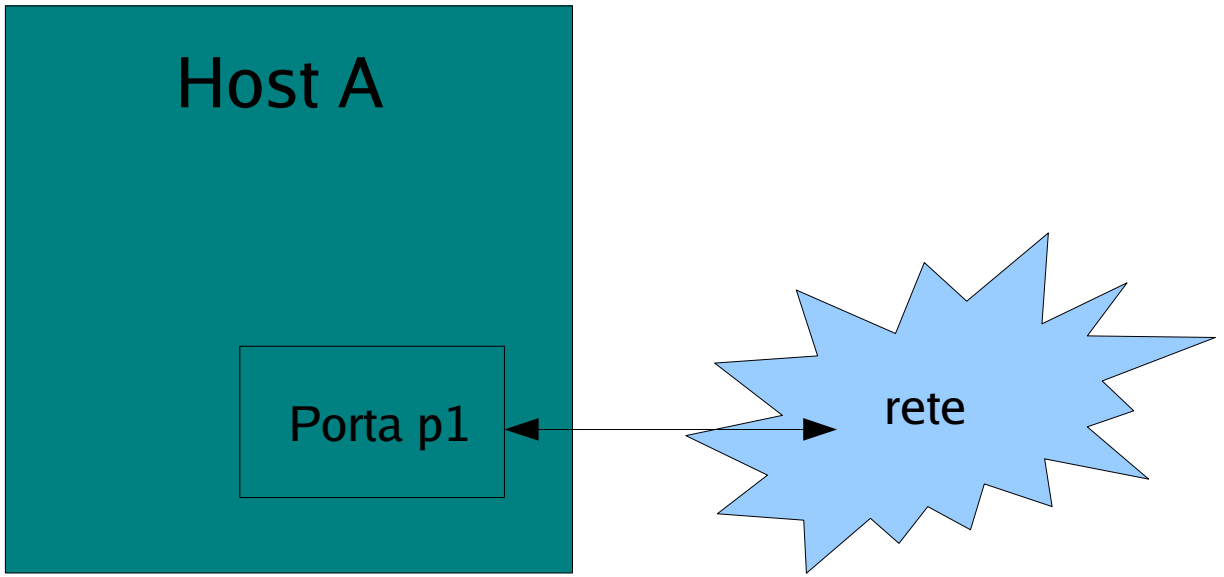
Spedizione Oggetti

```
try {
    ......

    ObjectOutputStream oos =
          new ObjectOutputStream(so.getOutputStream());

    pippo oggettoDaSpedire = new pippo();

    oggettoDaSpedire.intero = 33;
    oggettoDaSpedire.double = 33.3d;

    oos.writeObject(oggettoDaSpedire);

    oos.close();
    ......
}
catch(IOException e) {
    System.out.println("errore");}
```

Ricezione Oggetti

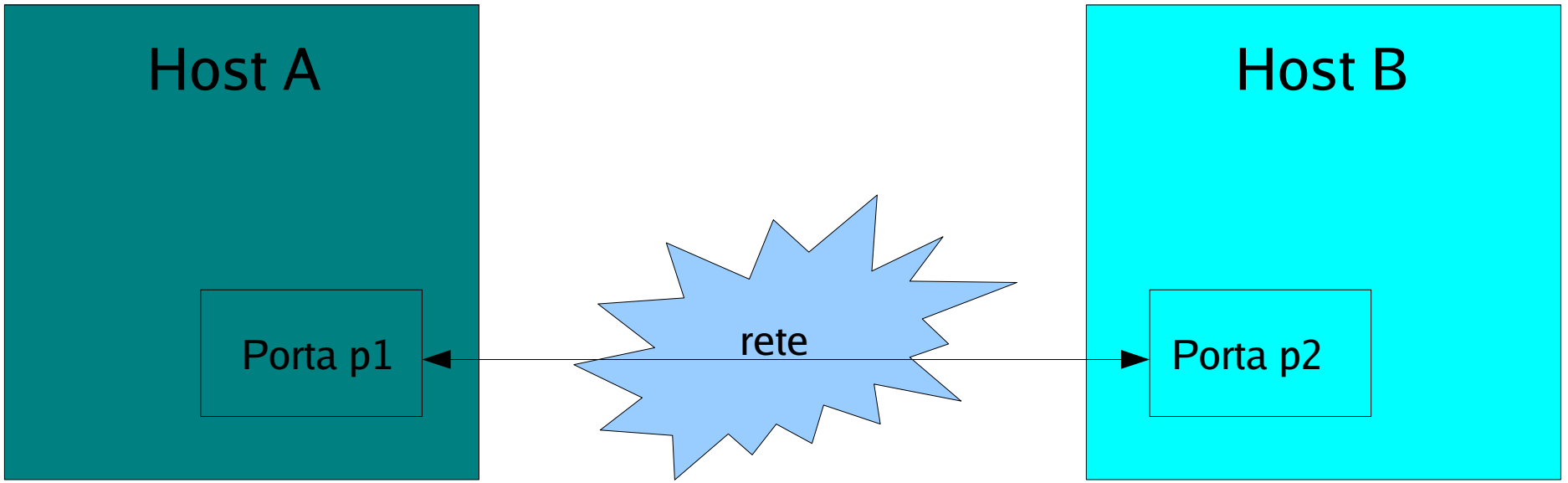
```
try {
    ......

    ObjectInputStream ois =
            new ObjectInputStream(so.getInputStream());

    pippo oggettoRicevuto = (pippo) ois.readObject();

    ......

    ois.close();

    ......
}
catch(IOException e) {
    System.out.println("errore");
}
```

Porte e Indirizzi nelle connessioni TCP




```
ss = new ServerSocket(p1);

so = ss.accept();
```

Porte e Indirizzi nelle connessioni TCP

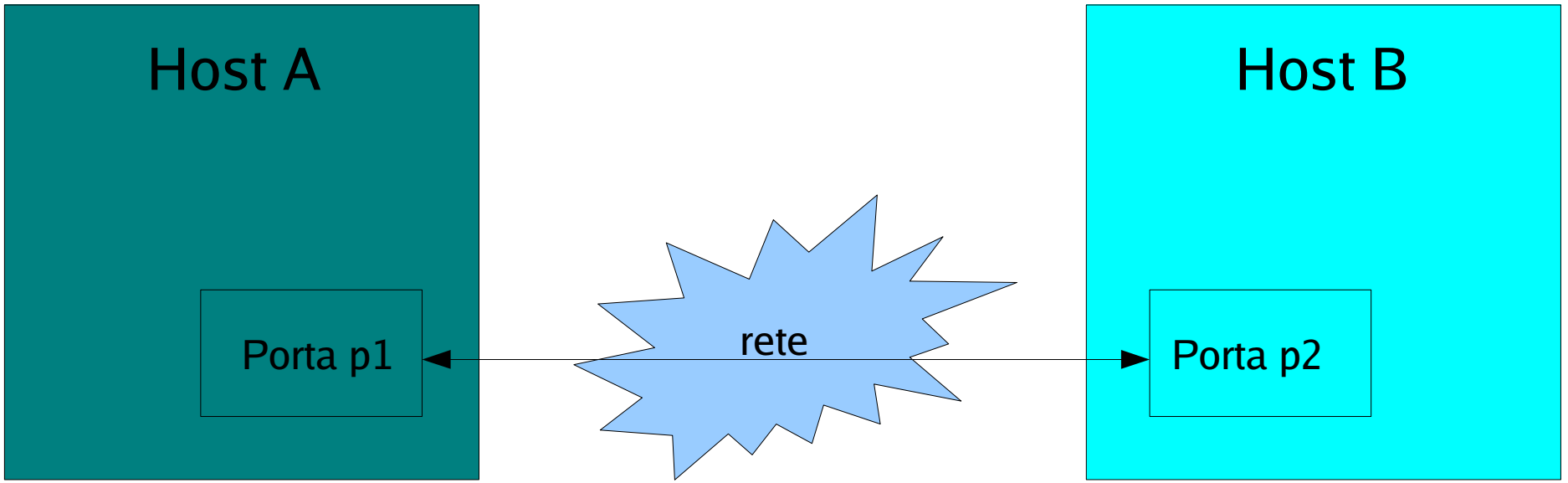



```
ss = new ServerSocket(p1);
```

```
s = new Socket(A, p1);
```

```
so = ss.accept();
```

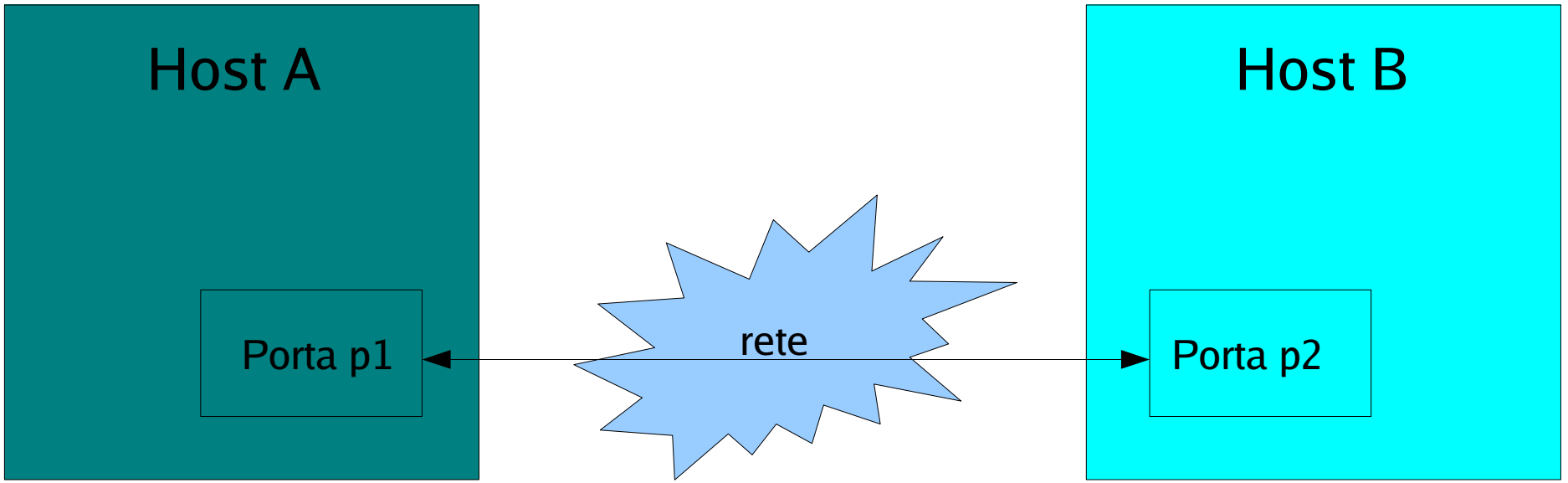
Porte e Indirizzi nelle connessioni TCP




```
ss = new ServerSocket(p1);

so = ss.accept();

so.getInetAddress()  = B
so.getPort()         = p2
so.getLocalPort()    = p1
```

```
s = new Socket(A, p1);

so.getInetAddress()  = A
so.getPort()         = p1
so.getLocalPort()    = p2
```

Porte e Indirizzi nelle connessioni TCP




```
ss = new ServerSocket(p1);

so = ss.accept();

so.getInetAddress()  = B
so.getPort()         = p2
so.getLocalPort()    = p1
```

```
s = new Socket(A, p1, B, p2);

so.getInetAddress()  = A
so.getPort()         = p1
so.getLocalPort()    = p2
```