*Socket*

- We use socket to establish the connection between client and server.
- Socket identifies a connection using host address and port number.

- A socket is a bi-directional communication channel

- Java differentiates client sockets from server sockets.

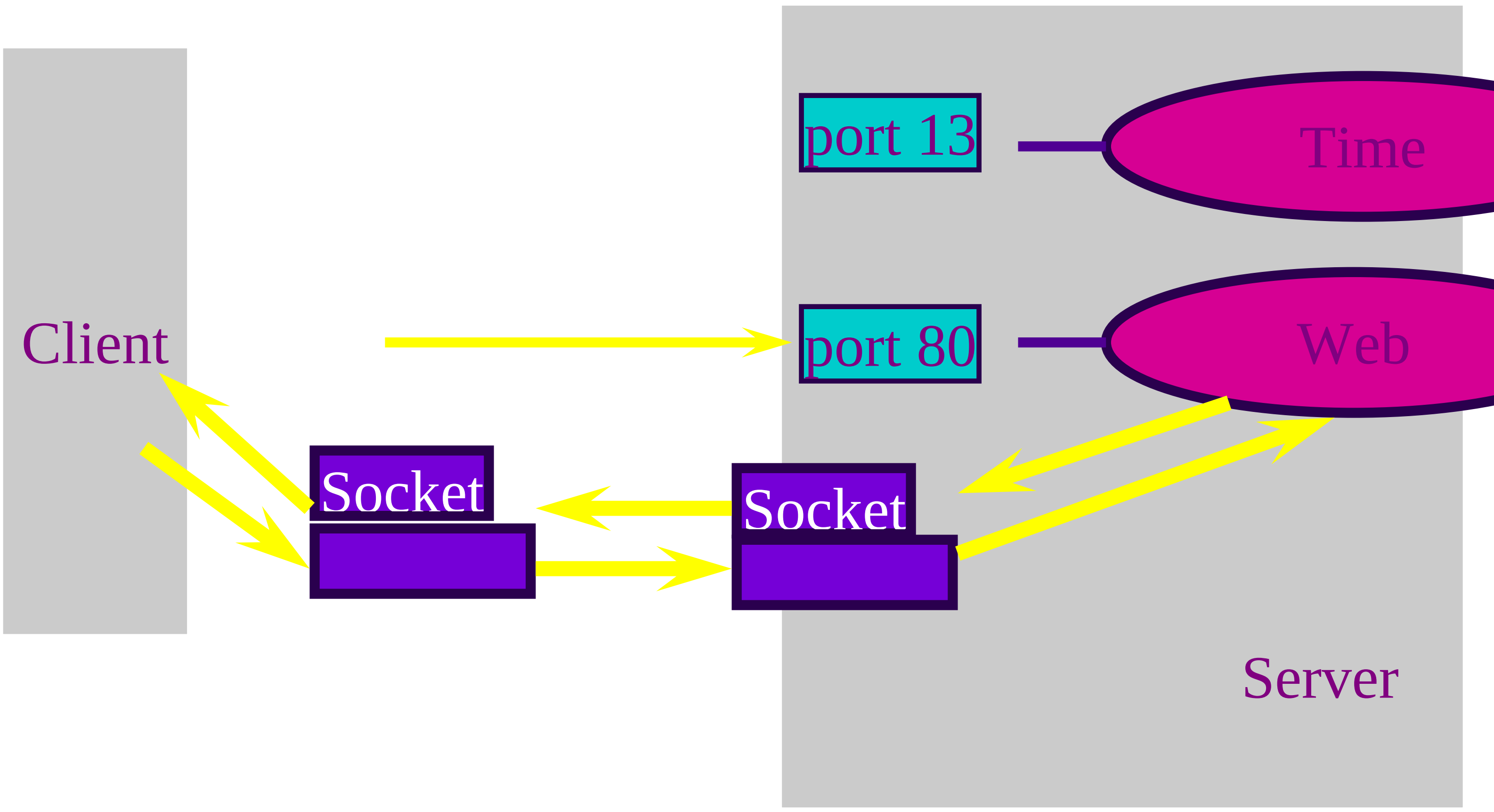
  e.g. *for client*

  Socket client=new Socket("hostname",portNumber);

  *for server*

  ServerSocket server=new SeverSocket(portNumber);

-



- Well known ports for server

| | |
|---|---|
| 80 | web server |
| 21 | Ftp server |
| 13 | Time server |
| 23 | Telnet |
| 25 | Email(SMTP) |

## *The connection between server and client using socket*

### Socket Operations at Client Side

- create a client socket:
  **Socket** (host, port)
  s = new **Socket** (“www.google.com”, 13)

- get input / output data streams out of the socket:
  in = new DataInputStream(s.**getInputStream** ());

  out = new DataOutputStream( s.**getOutputStream**());

// used if you want to store information “binary data”
    out = new PrintStream( s.**getOutputStream**());
// used if you want to display information “character data”

- read from input / write to output data streams:
  String str = in.**readLine**();
  out.**println** ( “Echo:” + str + “\r”);

- close the socket:
  s.**close**();

## Socket Operations at Server Side

A server is always waiting for being connected. It need not initiate a connection to a host. So a server socket need only specify its own port no.

- create a server socket:

  **ServerSocket** (port)
  ServerSocket s = new **ServerSocket**(8189);

- accept an incoming connection:

  Socket snew = s.**accept** ();

- get input / output data streams out of the socket for the incoming client:

  in = new **DataInputStream**(snew.getInputStream());
  out = new **PrintStream**(snew.getOutputStream());

- close the socket for the incoming client:

  snew.**close**();