Connected / Disconnected Modes

• Connected mode: path chosen and packets arrive all, in correct order (e.g. Phone)

• Disconnected mode: path not chosen for all, some packets may be missing, order not guaranteed (e.g. IP)
Sockets?

- Sockets are used to have two processes, potentially distributed, interact.
- Sockets can be TCP/IP or UDP/IP.
TCP
TCP

• how to create a socket???

```c
int sockfd, portno, n;
    struct sockaddr_in serv_addr;
    struct hostent *server;

char buffer[256];
if (argc < 3) {
    fprintf(stderr,"usage %s hostname port\n", argv[0]);
    exit(0);
}
portno = atoi(argv[2]);
sockfd = socket(AF_INET, SOCK_STREAM, 0);
if (sockfd < 0)
    error("ERROR opening socket");
server = gethostbyname(argv[1]);
if (server == NULL) {
    fprintf(stderr,"ERROR, no such host\n");
    exit(0);
}
```
TCP

• how to create a socket?? (WTH?)

bzero((char *) &serv_addr, sizeof(serv_addr));
serv_addr.sin_family = AF_INET;
bcopy((char *)server-&h_addr,
    (char *)&serv_addr.sin_addr.s_addr,
    server-&h_length);
serv_addr.sin_port = htons(portno);
if (connect(sockfd,&serv_addr,sizeof(serv_addr)) < 0)
    error("ERROR connecting");
TCP

- in Java:

```java
try {
    Socket s = new Socket(args[1], Integer.parseInt(args[2]));
} catch (Exception e) {
    System.out.println(e);
    System.exit(0);
}
```
# Socket

## Constructor Summary

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Socket()</code></td>
<td>Creates an unconnected socket, with the system-default type of SocketImpl.</td>
</tr>
<tr>
<td><code>Socket(InetAddress address, int port)</code></td>
<td>Creates a stream socket and connects it to the specified port number at the specified IP address.</td>
</tr>
<tr>
<td><code>Socket(InetAddress host, int port, boolean stream)</code></td>
<td>Deprecated. Use DatagramSocket instead for UDP transport.</td>
</tr>
<tr>
<td><code>Socket(InetAddress address, int port, InetAddress localAddr, int localPort)</code></td>
<td>Creates a socket and connects it to the specified remote address on the specified remote port.</td>
</tr>
<tr>
<td><code>Socket(Proxy proxy)</code></td>
<td>Creates a connected socket, specifying the type of proxy, if any, that should be used regardless of any other settings.</td>
</tr>
<tr>
<td><code>Socket(SocketImpl impl)</code></td>
<td>Protected class. Creates an unconnected <code>Socket</code> with a user-specified <code>SocketImpl</code>.</td>
</tr>
<tr>
<td><code>Socket(String host, int port)</code></td>
<td>Creates a stream socket and connects it to the specified port number on the named host.</td>
</tr>
<tr>
<td><code>Socket(String host, int port, boolean stream)</code></td>
<td>Deprecated. Use DatagramSocket instead for UDP transport.</td>
</tr>
<tr>
<td><code>Socket(String host, int port, InetAddress localAddr, int localPort)</code></td>
<td>Creates a socket and connects it to the specified remote host on the specified remote port.</td>
</tr>
</tbody>
</table>
Methods to know

- close()
- getInputStream()
- getOutputStream()
- setSoTimeout()
- getSoTimeout()
ServerSocket

- Let’s forget the C example... ;)
- client/server.
- Binds to a port.
Example

ServerSocket servs =
    new ServerSocket(1500);
Socket s = servs.accept();
System.out.println("Connection accepted from " +
    s.getRemoteSocketAddress());
**ServerSocket**

<table>
<thead>
<tr>
<th>Constructor Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ServerSocket()</strong></td>
</tr>
<tr>
<td>Creates an unbound server socket.</td>
</tr>
<tr>
<td><strong>ServerSocket(int port)</strong></td>
</tr>
<tr>
<td>Creates a server socket, bound to the specified port.</td>
</tr>
<tr>
<td><strong>ServerSocket(int port, int backlog)</strong></td>
</tr>
<tr>
<td>Creates a server socket and binds it to the specified local port number, with the specified backlog.</td>
</tr>
<tr>
<td><strong>ServerSocket(int port, int backlog, InetAddress bindAddr)</strong></td>
</tr>
<tr>
<td>Create a server with the specified port, listen backlog, and local IP address to bind to.</td>
</tr>
</tbody>
</table>
Methods to know

- close()
- accept()
- setSoTimeout()
- getSoTimeout()
Example

- Telnet?
SocketInteractor (1)

```java
import java.io.*;
import java.net.*;
public class SocketInteractor extends Thread{
    InputStream is;
    /**
     * Creates an instance with the input stream to
     * redirect to the keyboard
     */
    public SocketInteractor(InputStream is){
        this.is=is;
    }
    /**
     * Creates a new Thread and redirect a stream
     * on the keyboard
     */
```
public void run(){
    try{
        int a;
        // reads from the socket and prints on the terminal
        // as long as the socket is open.
        while(true){
            a=is.read();
            if (a==-1) throw new Exception("Socket closed.");
            System.out.write(a);
        }
    } catch (Exception E){
        System.out.println("socket closed.");
        System.exit(0);
    }
}
/**
 * Prints the usage and exits.
 */
public static void usage(){
    System.out.println("Usage: java SocketInteractor host port_number");
    System.out.println("connects to a socket and receive/send information through it");
    System.exit(0);
}
public static void main(String[] args) {
    OutputStream out = null;
    try{
        // checks the arguments
        if (args.length != 2)
            throw new Exception("Bad number of arguments.");
        // creates the socket
        Socket s = new Socket(args[0], Integer.parseInt(args[1]));
        out = s.getOutputStream();
        // starts the new thread
        (new SocketInteractor(s.getInputStream())).start();
    } catch (Exception e) {
        usage();
    }
    try{
        // reads on the terminal, outputs on the socket
        while (true) {
            out.write(System.in.read());
        }
    } catch (Exception e) {
        System.out.println("socket closed.");
        System.exit(0);
    }
}
}
public static void main(String[] args) {
    OutputStream out=null;
    ServerSocket servs=null;
    try{
        // checks the arguments
        if (args.length!=1)
            throw new Exception("Bad number of arguments.");

        // creates the socket
        servs=new ServerSocket(Integer.parseInt(args[0]));
        Socket s=servs.accept();
        System.out.println("Connection accepted from "+
            s.getRemoteSocketAddress());
        servs.close();
        out= s.getOutputStream();

        // starts the new thread
        (new SocketInteractor(s.getInputStream())).start();
    } catch (Exception E){
        usage();
    }

    try{
        // reads on the terminal, outputs on the socket
        while(true){
            out.write(System.in.read());
        }
    } catch (Exception E){
        System.out.println("socket closed.");
        System.exit(0);
    }
}
# DatagramSockets

## Constructor Summary

<table>
<thead>
<tr>
<th>Constructor</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>DatagramSocket()</code></td>
<td>Constructs a datagram socket and binds it to any available port on the local host machine.</td>
</tr>
<tr>
<td><code>DatagramSocket(DatagramSocketImpl impl)</code></td>
<td>Creates an unbound datagram socket with the specified DatagramSocketImpl.</td>
</tr>
<tr>
<td><code>DatagramSocket(int port)</code></td>
<td>Constructs a datagram socket and binds it to the specified port on the local host machine.</td>
</tr>
<tr>
<td><code>DatagramSocket(int port, InetAddress laddr)</code></td>
<td>Creates a datagram socket, bound to the specified local address.</td>
</tr>
<tr>
<td><code>DatagramSocket(SocketAddress bindaddr)</code></td>
<td>Creates a datagram socket, bound to the specified local socket address.</td>
</tr>
</tbody>
</table>
Methods to know

• close()
• connect(...)
• send(DatagramPacket)
• receive(DatagramPacket)
Want more?

- Multicast Sockets...
- How would you make a small Web server?
- How would you make a small FTP server?
Java Threads
Manuel Oriol - May 4th, 2006
Volatile Variables

- Reading from Manson, Pugh and Adve
Happens before model?

- a read r is allowed to observe a write w to v if r does not happen before w (and transitively)
Causality

- key to guarantee observable sequential consistency
- Intra-thread consistency
- synchronized actions are in mutual exclusion and respect causality
Thread

- a call to the method `start()` spawns a new thread (start a thread only once)
- The code executed is the one in `run()`
Runnable

- Implement the method run
- create a thread using the runnable object
ThreadGroup

• By default a thread belongs to its parent thread’s ThreadGroup

• These are a way of grouping threads and restricting accesses (protection domain)

• Hierarchical
Old Constructs????

- suspend
- resume
- destroy
- stop

Throws [NoSuchMethodError](https://docs.oracle.com/javase/8/docs/api/java/lang/NoSuchMethodError.html).

Throws:

[NoSuchMethodError](https://docs.oracle.com/javase/8/docs/api/java/lang/NoSuchMethodError.html) - always

deprecated... really????
So what?

• interrupt!

• or know what you are doing with locks...
Interrupt

interrupt

public void interrupt()

Interrupts this thread.

Unless the current thread is interrupting itself, which is always permitted, the checkAccess method of this thread is invoked, which may cause a SecurityException to be thrown.

If this thread is blocked in an invocation of the wait(), wait(long), or wait(long, int) methods of the Object class, or of the join(), join(long), join(long, int), sleep(long), or sleep(long, int), methods of this class, then its interrupt status will be cleared and it will receive an InterruptedException.

If this thread is blocked in an I/O operation upon an interruptible channel then the channel will be closed, the thread's interrupt status will be set, and the thread will receive a ClosedByInterruptException.

If this thread is blocked in a Selector then the thread's interrupt status will be set and it will return immediately from the selection operation, possibly with a non-zero value, just as if the selector's wakeup method were invoked.

If none of the previous conditions hold then this thread's interrupt status will be set.

Throws:
    SecurityException - if the current thread cannot modify this thread
Synchronizing with standard functionalities

- `join()`, `join(long)`
- `setPriority()` (higher is better)
- `yield()`
- `Thread.sleep(long)`