COPENHAGEN BUSINESS ACADEMY

Network and sockets

Jens Egholm Pedersen
<jeep@cphbusiness.dk>

Learning Goals

cphbusiness

Network – Socket Programming	
Main Topics	Socket Programming Network protocols
When this lesson is over you should be able to:	• Understand ports and IP Addresses • Implement Network Clients using either TCP (or UDP) • Implement Servers using either TCP (or UDP) • Implement thread safe multi threaded servers • Implement a simple user defined application layer protocol

Ports

- An endpoint of communication in an operating system
- Like a plug; only one at the time!
- 16 bit unsigned number
  - (0 – 65535)
- Ports beneath 1024 typically needs elevated permissions

See also: Wikipedia on Socket

Known ports

20	FTP -- Data
21	FTP -- Control
22	SSH Remote Login Protocol
23	Telnet
25	Simple Mail Transfer Protocol (SMTP)
37	Time
42	Host Name Server (Nameserv)
43	WhoIs
53	Domain Name System (DNS)
80	HTTP
110	POP3
115	Simple File Transfer Protocol (SFTP)
156	SQL Server
194	Internet Relay Chat (IRC)
197	Directory Location Service (DLS)
389	Lightweight Directory Access Protocol (LDAP)
443	HTTPS
546	DHCP Client
547	DHCP Server

See also: List of known ports

IPv4 Addresses

An IPV4 address is a 32-bit address. (The future is IPV6)

Binary Notation

10000001	10100101	00001101	11100100

Decimal

129	165	13	228

Hex

81	A5	0D	E4

= 0x81A50DE4

cphbusiness

IP address ranges

- Ranges of valid addresses
- Usually expressed in subnet mask or CIDR (Classless Inter-Domain Routing)
- Subnet mask (IPv4 only)
  - Expressed in dot-decimal (like IPv4 addresses)
  - Address: 192.168.0.0, net mask: 255.255.255.0
- CIDR range (IPv4 and IPv6)
  - Expressed in remaining bits
  - 192.168.0.0/24

See also: IP subnetting, CIDR notation

Socket

- An endpoint of network communication
- Like a plug; only one at the time!

- Consists of an IP address and port number

See also: Wikipedia on Socket

Using sockets

- `java.net.Socket`
- `java.net.ServerSocket`
- `java.net.SocketAddress`

- `socket.bind()`
  - What can go wrong?
    - `IOException`

See also: Wikipedia on Socket

IO Exceptions

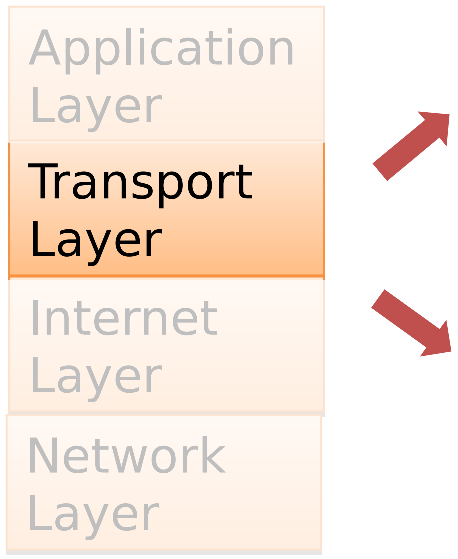
- Input/Output = stort set alt
- Known subclasses:
  - ChangedCharSetException, CharacterCodingException, CharConversionException, ClosedChannelException, EOFException, FileLockInterruptionException, FileNotFoundException, FilerException, FileSystemException, HttpRetryException, IIOException, InterruptedByTimeoutException, InterruptedIOException, InvalidPropertiesFormatException, JMXProviderException, JMXServerErrorException, MalformedURLException, ObjectStreamException, ProtocolException, RemoteException, SaslException, SocketException, SSLException, SyncFailedException, UnknownHostException, UnknownServiceException, UnsupportedDataTypeException, UnsupportedEncodingException, UserPrincipalNotFoundException, UTFDataFormatException, ZipException

See also: Java API on IOException

Recap

- Ports
- IP addresses
- IP address ranges
- Sockets

Socket Programming



The Internet protocol stack

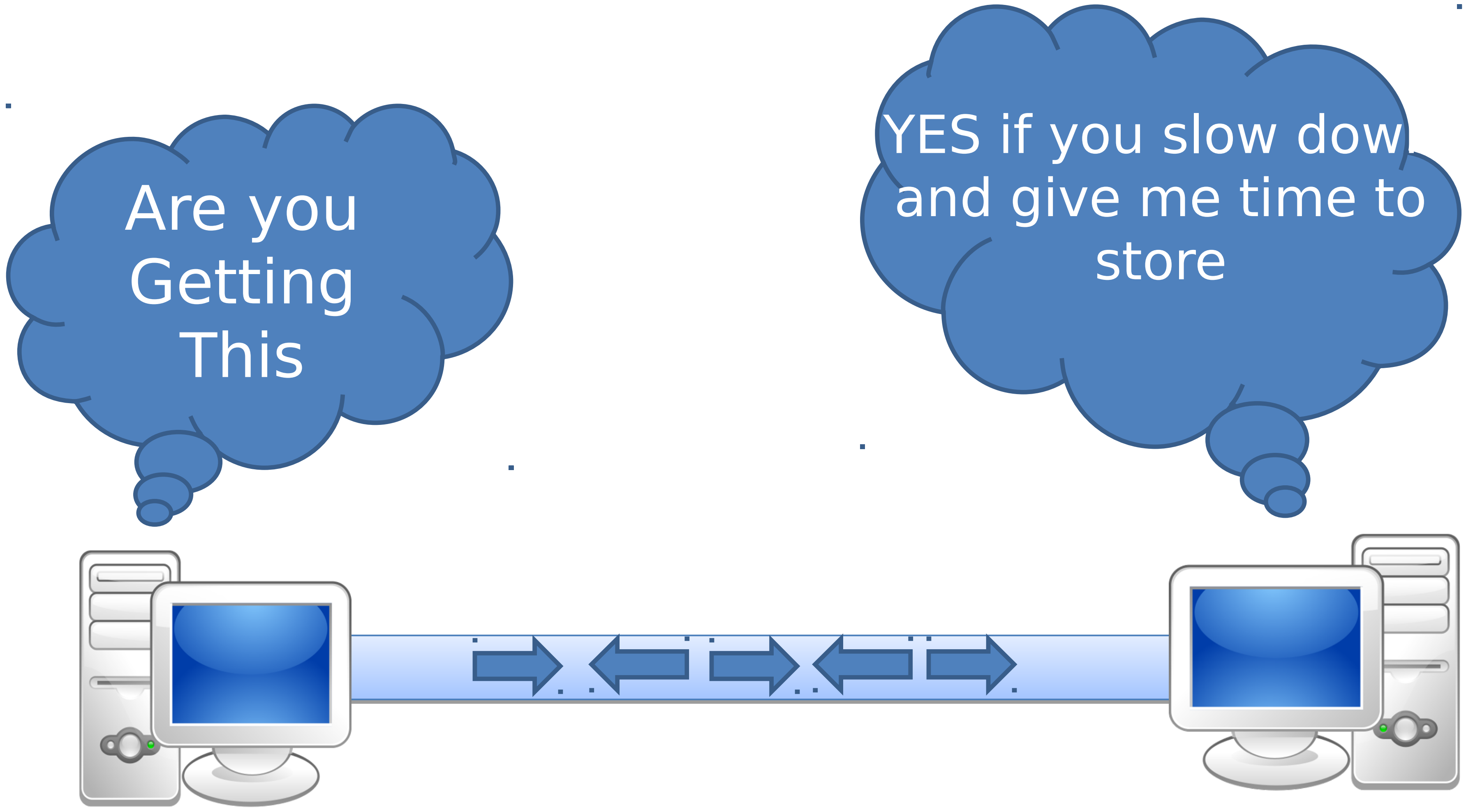
TCP (Transmission Control Protocol)

TCP provides a connection-oriented service to its applications. This service includes guaranteed delivery of application-layer messages to the destination .

UDP (User Datagram Protocol)

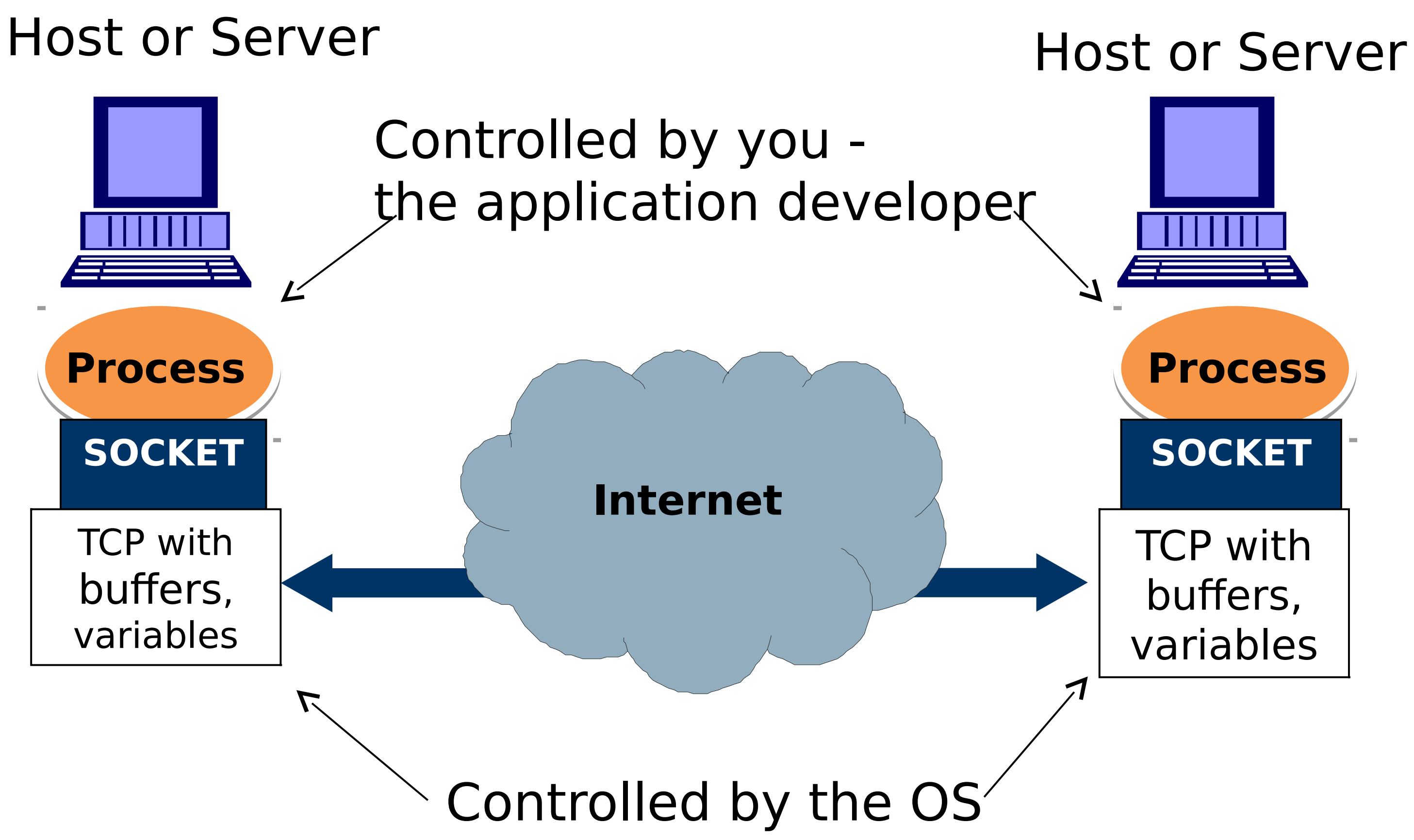
UDP provides a connectionless service with no guaranteed delivery of application-layer messages to the destination .

TCP


Are you Getting This
YES if you slow down and give me time to store

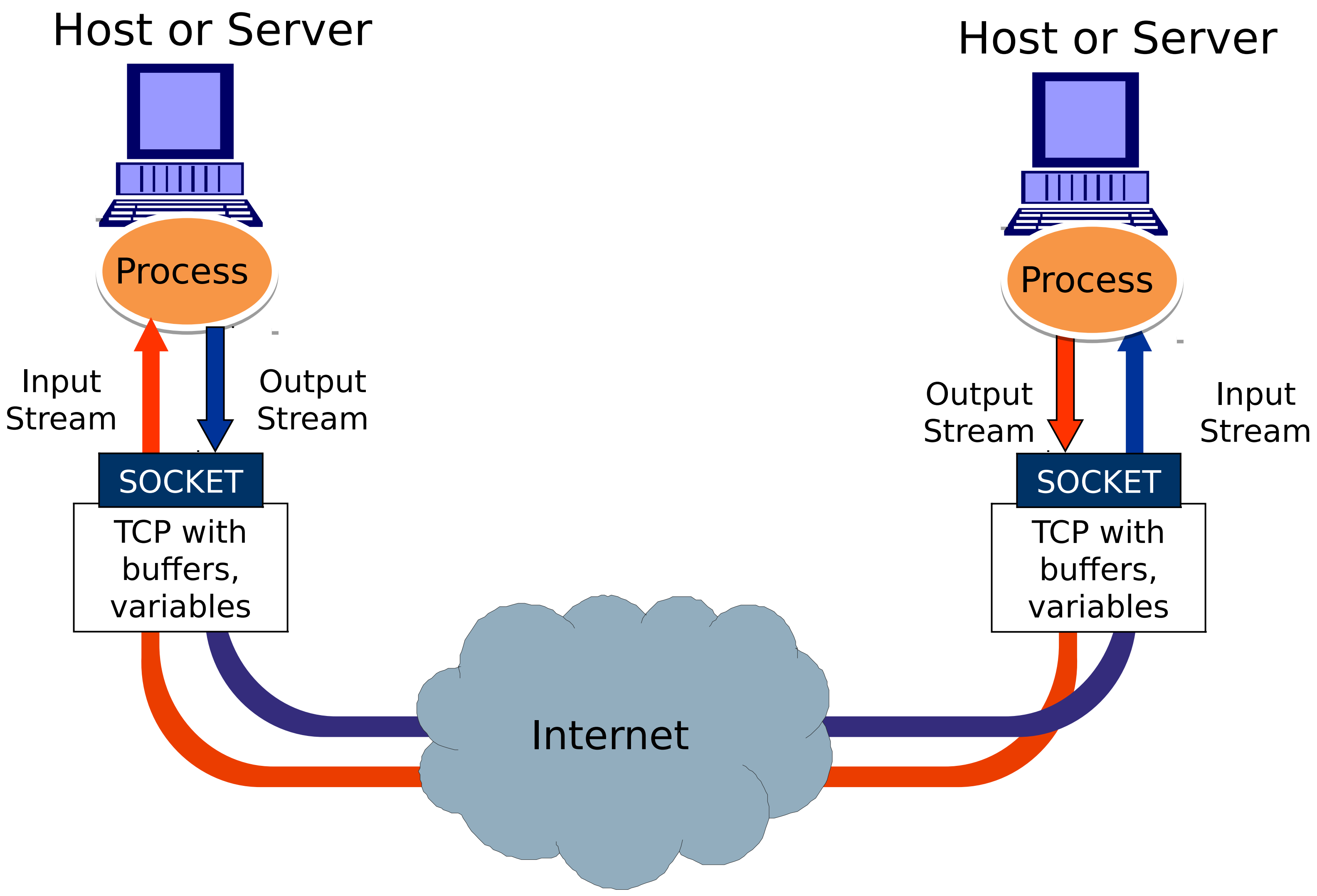
Socket Programming with TCP

cphbusiness



The application developer has the ability to fix a few TCP parameters, such as maximum buffer and maximum segment sizes.

Socket and Streams



TCP Socket Programming with Java

cphbusiness

1: Create a ServerSocket object.

```
ServerSocket ss=new ServerSocket(8080);
```

2: Put server into waiting state

```
Socket link = ss.accept();
```

3: Set up input and output streams

```
BufferedReader in =
  new BufferedReader(
    new InputStreamReader(
      link.getInputStream()));
```

4: Send and Receive Data

5: Close Connection

```
Link.close()
```

1: Establish a connection to the server

(Three way handshake)

```
Socket link = new Socket("lmo",8080);
```

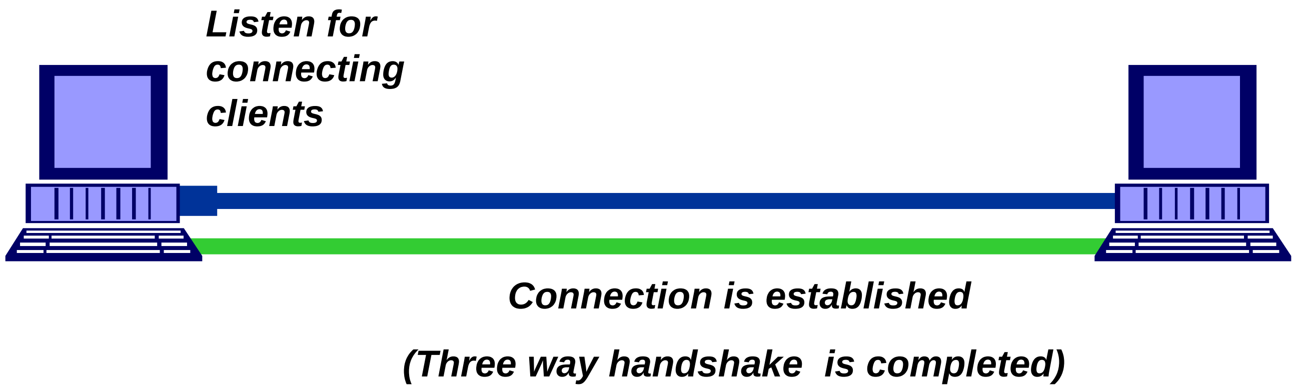
2: Set up input and output streams

```
PrintWriter out = new PrintWriter(
   link.getOutputStream(),true);
```

3: Send and Receive Data

4: Close Connection

```
Link.close()
```


Listen for connecting clients
Connection is established
(Three way handshake is completed)

Sockets for Servers and Clients

- Server
  - Server Socket (java.net.ServerSocket)
    - Handles the initial contact from a client.
  - Socket (java.net.Socket)
    - Is created at initial contact of client.
    - New socket that is dedicated to the particular client.

- Client
  - Socket (java.net.Socket)
    - Initiate a TCP connection to the server by creating a socket object. (Three-way handshake)
    - Specify the address of the server process, namely, the IP address of the server and the port number of the process.

Sockets for Servers and Clients

java.net.Socket

- Implements client sockets (also called just “sockets”).
- An endpoint for communication between two machines.
- Constructor and Methods
  - Socket(String host, int port): Creates a stream socket and connects it to the specified port number on the named host.
  - `InputStream getInputStream()`
  - `OutputStream getOutputStream()`
  - `close()`

java.net.ServerSocket

- Implements server sockets.
- Waits for requests to come in over the network.
- Performs some operation based on the request.
- Constructor and Methods
  - `ServerSocket(int port)`
  - `Socket Accept():` Listens for a connection to be made to this socket and accepts it. This method blocks until a connection is made.

The bind(..) method

In the simple examples we have done so far we only gave a port number when

we constructed a ServerSocket

What if the server had more than one network interface?

Bind your Socket to an IP + a PORT

Never hardcode IP and Port numbers
They will change when you deploy

```
public static void main(String args[]){
  if(args.length==2){
     ip = args[0];
     port = Integer.parseInt(args[1]);
  } else throw new IllegalArgumentException();
  ServerSocket serverSocket = new ServerSocket();
  serverSocket.bind(new InetSocketAddress(ip, port));
  ...
```

On a server you need to bind to a specific network interface

(IP), otherwise it won't work

Example: Chat system

Goal: Design a chat system

Three actions: Log in, log out and send message

What transport protocol to use?

- TCP

How would you log in and out?

- List of clients

How would you send messages?

  - Observables

Recap

- Sockets
  - Like plugs – only one connection per plug!
- TCP and UDP
  - What is the difference?
- IP addresses
  - Unique addresses
    - … for that network

Protocols

In telecommunication, a protocol is the special set of rules that end points in a network connection use when they communicate
Protocols specify interactions between the communicating entities

What are examples of protocols for:

- Client and server on the Web
- Sending files between client and server
- Human to human

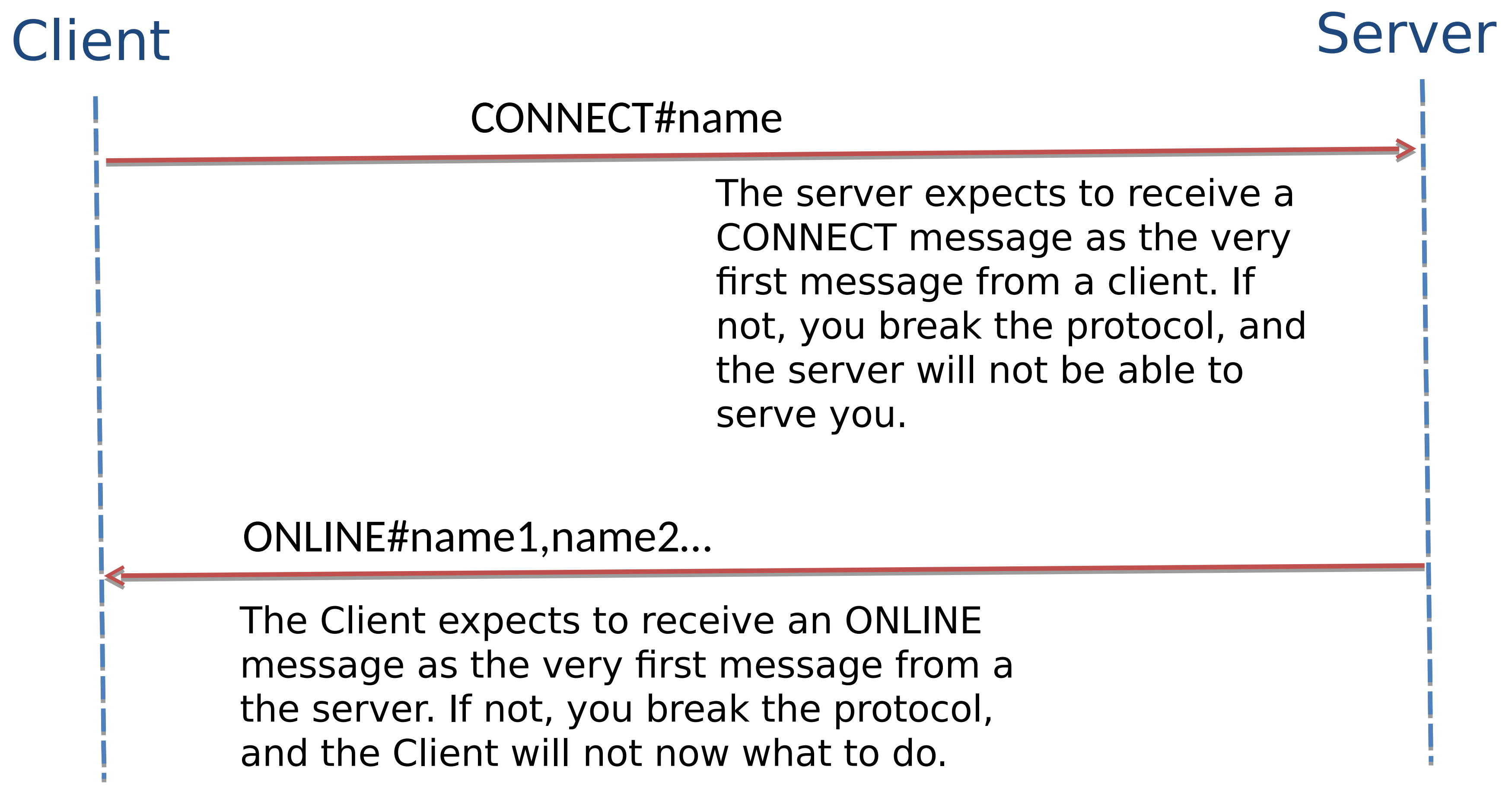
Protocols - example

http://docs.oracle.com/javase/tutorial/networking/sockets/clientServer.html

Download and test the file knockknock.zip

Protocols - example

This could be part of a protocol for a CHAT system

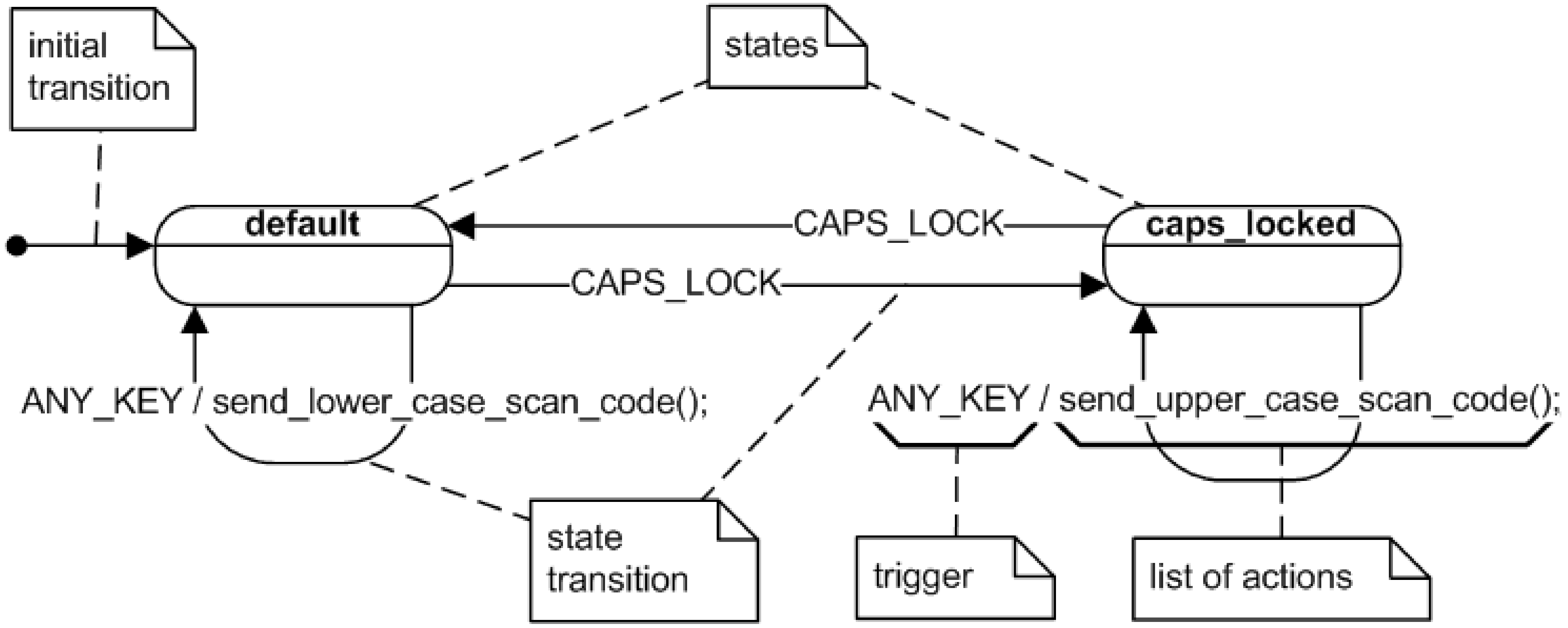


The server expects to receive a CONNECT message as the very first message from a client. If not, you break the protocol, and the server will not be able to serve you.

The Client expects to receive an ONLINE message as the very first message from a the server. If not, you break the protocol, and the Client will not now what to do.

State diagrams

Finite State Machines and State Diagrams can be used to describe protocols.



http://en.wikipedia.org/wiki/UML_state_machine

Patterns for simple protocols

- One Protocol Handler Class as in the Knock Knock example (only for simple protocols)
- The GOF **State** Pattern (http://en.wikipedia.org/wiki/State_pattern)
- The GOF **Command** Pattern (http://en.wikipedia.org/wiki/Command_pattern)
- …