Script **generated by TTT**

Title: Distributed_Applications (30.04.2012)

Date: Mon Apr 30 09:16:44 CEST 2012

Duration: 31:10 min

Pages: 9

Names are used to uniquely identify entities and refer to locations. An important issue is name resolution.

Names

A ***name*** is a string of characters that is used to refer to an entity (e.g. host, printer, file).

entities have access points to invoke operations on them ⇒ ***address*** is the name of the access point.

an identifier is a name which uniquely identifies an entity.

Name space

Names in distributed systems are organized into a name space.

Name spaces are organized hierarchically.

Representation as a labeled directed graph.

Path along graph edges specifies the entity name, e.g. documents/projects/lecture2003/concept.tex; absolute vs relative path names.

Name resolution : a name lookup returns the identifier or the address of an entity, e.g. LDAP Name Service.

Information Sharing
Message exchange
Naming entities
Bidirectional communication
Producer-consumer interaction
Client-server model
Peer-to-peer model
Group model
Taxonomy of communication
Message serialization
Levels of Abstraction

Generated by Targeteam

constructors of java.net.socket

Socket(): Creates an unconnected socket, with the system-default type of SocketImpl.

Socket(InetAddress address, int port): Creates a stream socket and connects it to the specified port number at the specified IP address.

Socket(Proxy proxy) Creates an unconnected socket, specifying the type of proxy, if any, that should be used regardless of any other settings.

Socket(String host, int port) Creates a stream socket and connects it to the specified port number on the named host.

methods of java.net.socket

void bind(SocketAddress bindpoint): Binds the socket to a local address.

void close(): Closes this socket.

void connect(SocketAddress endpoint): Connects this socket to the server.

void connect(SocketAddress endpoint, int timeout): Connects this socket to the server with a specified timeout value.

Example

Example from the client perspective

```
import java.io.*
import java.net.*

public class EchoClient {
public static void main(String[] args) throws IOException {
Socket echoSocket = null;
Printwriter out = null;
BufferedReader in = null;

try {
echoSocket = new Socket("www.in.tum.de", 7); //create Socket
// create Writer, Reader
out = new PrintWriter(echoSocket.getOutputStream(), true);
in = new BufferedReader(
    new InputStreamReader(echoSocket.getInputStream()) );
}
catch (UnknownHostException e) {
    System.err.println("unkown host in.tum.de");
    System.exit(1);
}
catch (IOException e) {
```

Example

client Server

```
    System.err.println("unkown host in.tum.de");
    System.exit(1);
}
catch (IOException e) {
    System.err.println("No I/O from in.tum.de");
    System.exit(1);
}

//read streams
BufferedReader stdIn = new BufferedReader (
    new InputStreamReader(System.in));
String userInput;
while ((userInput = stdIn.readLine()) != null) {
    out.println(userInput);
    System.out.println("echo: " + in.readLine());
}

// close streams and sockets
out.close();
in.close();
stdIn.close();
echoSocket.close();
```

Bidirectional communication

Usage of the request-answer scheme for message exchange.

Sockets

Call semantics

Communication between sender and receiver is influenced by the following situations

loss of request messages.

loss of answer messages.

sender crashes and is restarted.

receiver crashes and is restarted.

Different types of call semantics

Any communication between a sender and a receiver is subject to communication failures. Therefore, we distinguish between different call semantics.

[at-least-once semantics]

[exactly-once semantics]

last semantics

Under a last semantics, the requested service operation is processed once or several times, however, only the last processing produces a result and, potentially, some side-effects.

at-most-once semantics

Under an at-most-once semantics, the requested service operation is processed once or not at all.

Example for providing at-most-once semantics

After timeput at the sending site the request is not retransmitted.

The request is transmitted in the context of a transaction.

Under an at-least-once semantics, the requested service operation is processed once or several times.