Script generated by TTT

Title: Distributed_Applications (29.04.2013)

Date: Mon Apr 29 09:11:32 CEST 2013

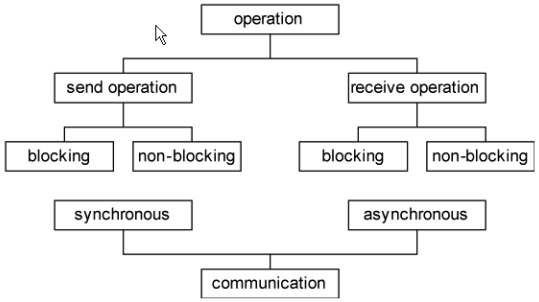
Duration: 44:02 min

Pages: 11

Paradigms for distributed applications

Message exchange

Interprocess communication (IPC): message exchange between sender and receiver.



Background

Message exchange takes place between a sending and a receiving process.

Basic functionality

```
send(E: receiver, N: message);

receive(S: sender, B: buffer);
```

Communication perspectives

We can distinguish between different perspectives with respect to the communication among the involved processes:

the sender's view, and

the receiver's view

Assumption: Sender S has invoked the operation `send(E, N)`; receiver E performs the operation `receive(S, B)`.

Categories of Message Exchange

[Asynchronous message exchange (nonblocking)]
[Synchronous message exchange (blocking)]

Remote-invocation send

Sender S suspends execution until the receiver has received and processed the submitted request that was delivered as part of the message.

Asynchronous message exchange (nonblocking)

Sender S can resume its processing immediately after the message N is put forward into the message queue NP (NP is also called message buffer).

S will ***not*** wait until the receiver E has received the message N.

A `receive` operation indicates that the receiver is interested in receiving a message.

Example
Advantages of asynchronous message exchange

Advantages of asynchronous message exchange

Advantages

useful for real-time applications, especially if the sending process should not be blocked.

supports parallel execution threads at the sender's and the receiver's sites.

it can be used for event signaling purposes.

Disadvantages

management of message buffers, handling of buffer overflow, access control problems, and of process crashes (receiver).

notification of S in case of failures may be a problem, since mostly S has already continued with its regular processing.

design of a correct system is difficult. The failure behavior depends heavily on buffer sizes, buffer contents, and the time behavior of the exchanged messages.

Asynchronous message exchange (nonblocking)

Sender S can resume its processing immediately after the message N is put forward into the message queue NP (NP is also called message buffer).

S will ***not*** wait until the receiver E has received the message N.

A `receive` operation indicates that the receiver is interested in receiving a message.

Example
Advantages of asynchronous message exchange

Generated by Targeteam

Categories of Message Exchange

Asynchronous message exchange (nonblocking)

Synchronous message exchange (blocking)

Remote-invocation send

Sender S suspends execution until the receiver has received and processed the submitted request that was delivered as part of the message.

Naming entities

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.

Paradigms for distributed applications

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

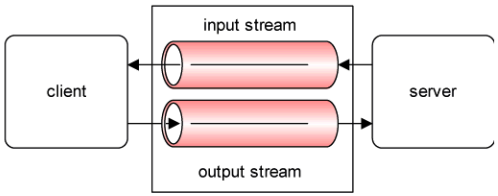
Levels of Abstraction

Sockets

Sockets provide a low level abstraction for programming bidirectional communication.

A socket is an application created, OS-controlled interface into which application can both send and receive messages to/from another application.

unique identification: IP-address and port number.



Socket connection

Sockets in Java

Java package java.net

Socket constructors - methods