

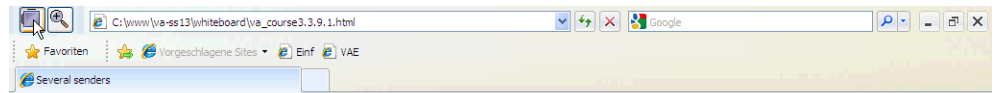
Script generated by TTT

Title: Distributed_Applications (06.05.2013)

Date: Mon May 06 09:22:44 CEST 2013

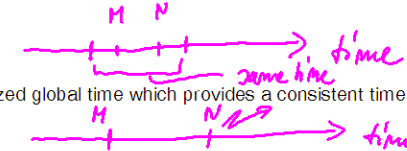
Duration: 35:13 min

Pages: 8



If several senders are involved, the following message ordering schemes may be applied:

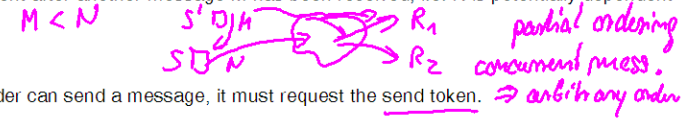
1. no serialization.
2. loosely-synchronous.
3. virtually-synchronous.



There is a loosely synchronized global time which provides a consistent time ordering.

4. totally ordered.

The message order is determined by causal interdependencies among the messages. For example, a message N has been sent after another message M has been received, i.e. N is potentially dependent on M.

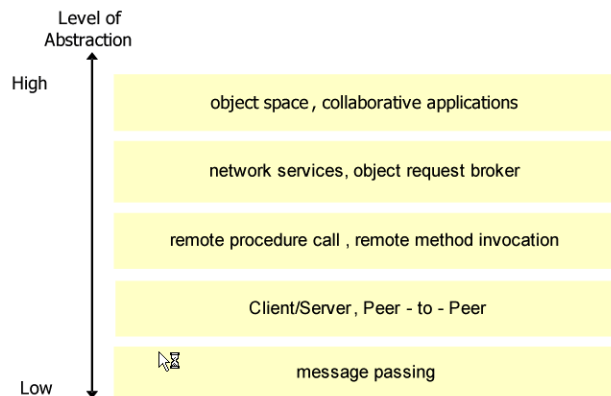


by token: before a sender can send a message, it must request the send token.

a selected component (the coordinator) determines the order of message delivery for all recipients.



Levels of Abstraction



Client-server model



The client-server model implements a sort of *handshaking principle*, i.e., a client invokes a server operation, suspends operation (in most of the implementations), and resumes work once the server has fulfilled the requested service.

[Terms and definitions](#)

[Concepts for client-server applications](#)

[Processing of service requests](#)

[File service](#)

[Time service](#)

Definition: A **time service** provides a synchronized system-wide time for all nodes in the network.

[Name service](#)

[LDAP - Lightweight Directory Access Protocol](#)

[Failure tolerant services](#)



Definitions



sender, receiver: pure message exchanging entities.

client, server: entities acting in some specialized protocol.

Client

Definition: A **client** is a process (some say, an application) that runs on a client machine and that typically initiates requests for service operations.

Potential clients are a priori unknown.

Service

Definition: A **service** is a piece of software that provides a well-defined set of service operations. This piece of software may run on one or multiple (server) machines.

Server

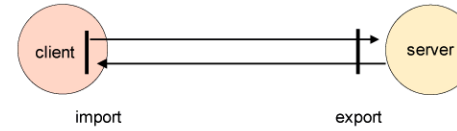
Definition: A **server** is a subsystem that provides a particular service to a set of a priori unknown clients. A server executes a (piece of) service software on a particular server machine. Obviously, a single server machine can host multiple server subsystems.

A server provides a set of operations (procedures).

Generated by Targeteam



Client-server interfaces



1. Client interface (import interface)

It represents the server within the client;

It prepares parameters and sends the request messages to the server;

It prepares the interpretation of the result that is extracted from the answer message submitted by the server.

2. Server interface (export interface)

It represents all potential clients within the server;

It accepts client requests; interprets the parameters; prepares results;

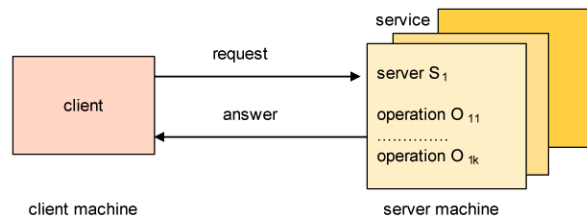
It invokes the respective service operation;

It prepares and sends the answer message containing the result of the service operation.

Generated by Targeteam



Terms and definitions



Definitions

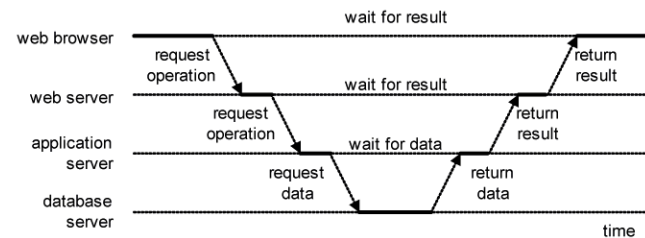
Client-server interfaces

Multitier architectures

Generated by Targeteam



Timing process



Generated by Targeteam



<i>Client</i>					
presentation execution	presentation	presentation	presentation execution	presentation execution (with local database)	presentation execution database
<i>Server</i>					
database	presentation execution database	execution database	execution database	execution (with local database)	database
Case 1	Case 2	Case 2	Case 3	Case 3	Case 4

Different cases

Case 1: remote data storage. access, for example, via Sun NFS.

Case 2: remote presentation (for example X window system).

Case 3: distributed application

cooperative processing among the individual components of an application.

Case 4: distributed data storage

The information is distributed between client and server; information replication is possible.