

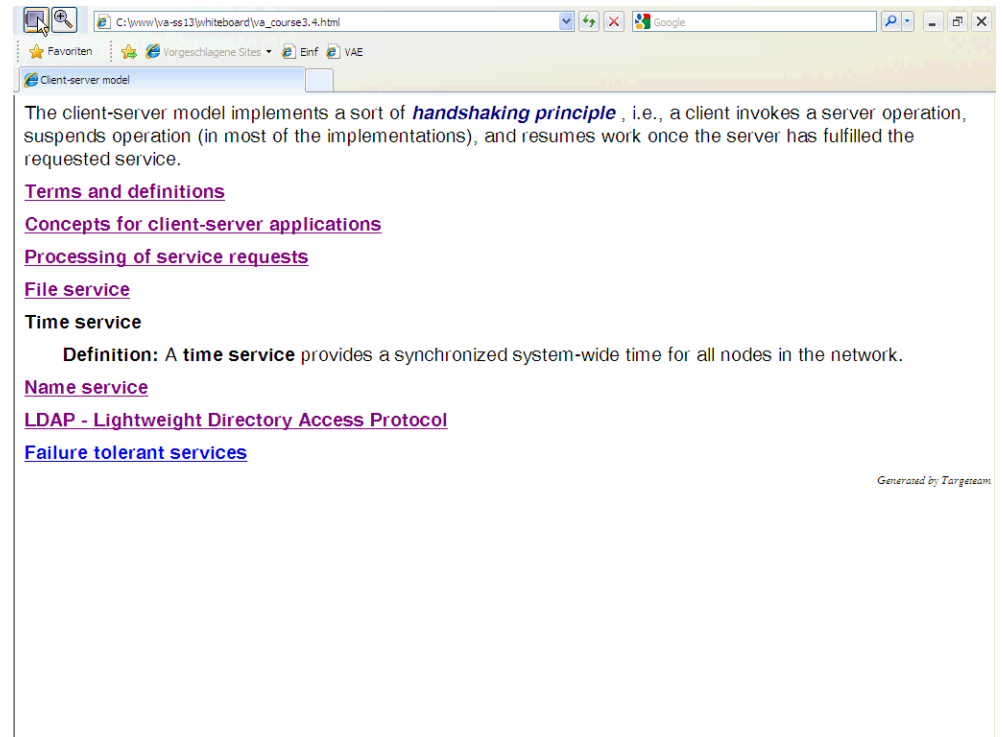
Script generated by TTT

Title: Distributed_Applications (13.05.2013)

Date: Mon May 13 09:10:41 CEST 2013

Duration: 44:51 min

Pages: 10



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](#)

Generated by Targeteam



Failure tolerant services



There may exist multiple redundant services; server copies and client copies are grouped together into server and client groups.

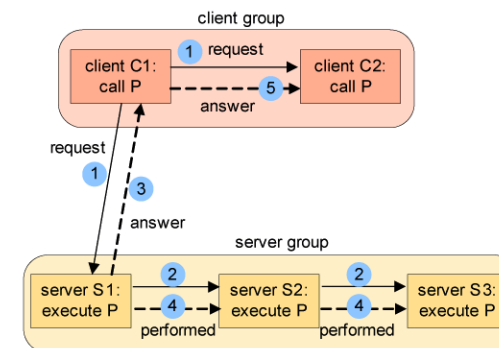
[Modular redundancy](#)

[Primary-standby-approach](#)

Generated by Targeteam



Primary-standby-approach



At any specific time, there is only one replica acting as master (primary replica); RPC requests are always propagated to the primary replica; at checkpoints the current state is propagated to the secondary replicas.

in case of an error the master is replaced by a backup replica.

distinction between hot and cold standby.

Generated by Targeteam



There may exist multiple redundant services; server copies and client copies are grouped together into server and client groups.

[Modular redundancy](#)

[Primary-standby-approach](#)

Generated by Targeteam



- Prof. J. Schlichter
 - Lehrstuhl für Angewandte Informatik / Kooperative Systeme, Fakultät für Informatik, TU München
 - Boltzmannstr. 3, 85748 Garching

Email: schlichter@in.tum.de

Tel.: 089-289 18654

URL: <http://www11.in.tum.de/>

[Overview](#)

[Introduction](#)

[Architecture of distributed systems](#)

[Remote Invocation \(RPC/RMI\)](#)

[Basic mechanisms for distributed applications](#)

[Web Services](#)

[Design of distributed applications](#)

[Distributed file service](#)

[Distributed Shared Memory](#)

[Object-based Distributed Systems](#)

[Summary](#)

Generated by Targeteam



Definition: Birrell and Nelson (1982) define an **RPC** as a synchronous flow of control and data passing scheme achieved through procedure calls between processes running in separate address spaces where the needed communication is via small channels (with respect to bandwidth and duration time).

synchronous : The calling process (client) is blocked until it receives the answer of the called procedure (server); the answer contains the results of the processed request.

procedure calls : the format of an RPC call is defined by the signature of the called procedure.

different address spaces : it is necessary to handle pointers during parameter passing different from local procedure calls.

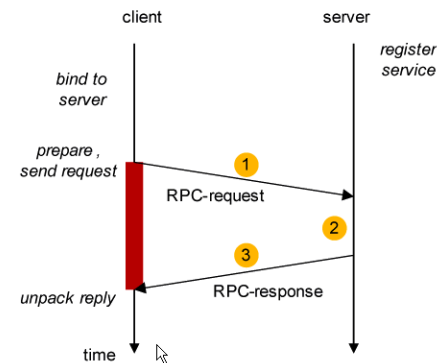
small channel : reduced bandwidth for communication between involved computers.

Generated by Targeteam



Neither the client nor the server assume that the procedure call is performed over a network.

Control flow for RPC calls



[Differences between RPC and local procedure call](#)

[Basic RPC characteristics](#)

[RPC and OSI](#)

[RPC vs message exchange](#)

Generated by Targeteam



- For an **RPC**, the caller and the callee run in different processes.
- both processes (caller and callee) have
 - no shared address space.
 - no common runtime environment.
 - different life span of [client and server](#) .

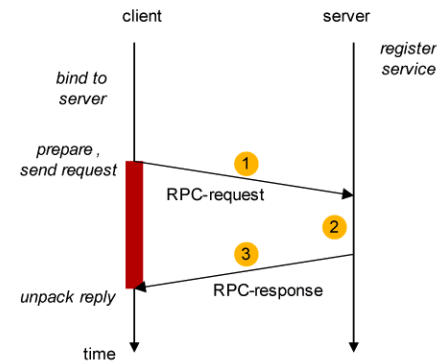
Handle errors occurring during a **RPC** call, e.g. caused by machine crashes or communication failures
RPC-based applications must take communication failures into consideration.

Generated by Targeteam



Neither the client nor the server assume that the procedure call is performed over a network.

Control flow for RPC calls



[Differences between RPC and local procedure call](#)

[Basic RPC characteristics](#)

[RPC and OSI](#)

[RPC vs message exchange](#)

Generated by Targeteam



Integration of the RPC into ISO/OSI protocol stack

layer 7 application layer	client-server model	
layer 6 presentation layer	RPC	hides communication details
layer 5 session layer	message exchange, e.g. request-response protocol	Operating system interface to underlying communication protocols
layer 4 transport layer	transport protocols e.g. TCP/UDP or OSI TP4	transfer of data packets

transport protocols: UDP (User Datagram Protocol) transports data packets without guarantees; TCP (Transmission Control Protocol) verifies correct delivery of data streams.

message exchange: socket interface to the underlying communication protocols.

RPC: hides communication details behind a procedure call and helps bridge heterogeneous platforms.

Generated by Targeteam