Script generated by TTT

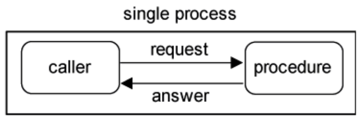
Title:	Distributed_Applications (12.05.2014)
Date:	Mon May 12 09:17:15 CEST 2014
Duration:	45:47 min
Pages:	15

C:\www\va-ss14\whiteboard\va_course4.2.html

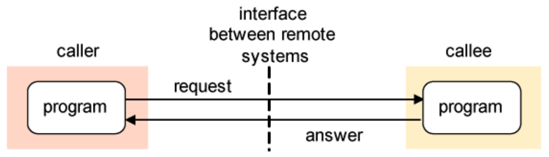
Favoriten | Vorgeschlagene Sites | Einf | VAE

Introduction

Local vs. remote procedure call



RPC is an extension of the same type of communication to programs running on different computers; single thread of execution and transfer of data.



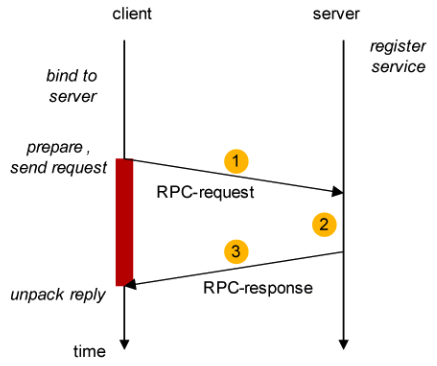
Definition

RPC properties

RPC properties

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

Differences between RPC and local procedure call

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.

Basic RPC characteristics

An RPC can be characterized as follows

1. uniform call semantics.
2. "type-checking" of parameters and results.
3. parameter functionality.
4. Optimize response times rather than throughput.
5. new error cases

   bind operation failed; request timed out; arguments are too large

   goal is some transparency concerning exception handling and communication failures (relevant for the programmer).

RPC and OSI

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.

RPC vs message exchange

RPC	message exchange
synchronous (generally)	asynchronous
1 primitive operation (RPC call)	2 primitive operation (send, receive)
messages are configured by RPC system	message specification by programmer
one open RPC	several parallel messages possible

The RPC protocol defines only the structure of the request/answer messages; it does not supply a mechanism for secure data transfer.

RPC exchange protocols

RPC exchange protocols

There are different types of RPC exchange protocols

the request (R) protocol

the request-reply (RR) protocol

the request-reply-acknowledge (RRA) protocol.

Remote Invocation (RPC/RMI)

Stubs

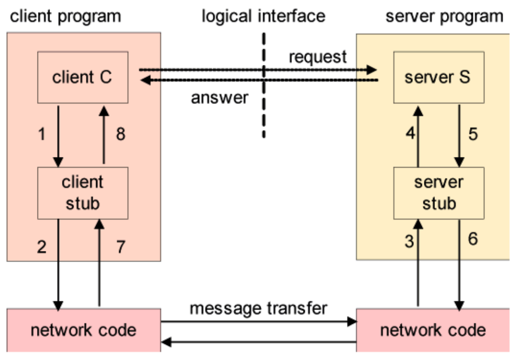
Integration of software handling the communication between components of a distributed application.

Stubs encapsulate the distribution specific aspects.

Stubs represent interfaces.

Client Stub : contains the proxy definition of the remote procedure P.

Server Stub : contains the proxy call for the procedure P.



Stub functionality

Client and server stubs have the following tasks during client - server interaction.

1. **Client stub**

   specification of the remote service operation; assigning the call to the correct server; representation of the parameters in the transmission format.

   decoding the results and propagating them to the client application.

   unblocking of the client application.

2. **Server stub**

   decoding the parameter values; determining the address of the service operation (e.g. a table lookup).

   invoking the service operation.

   prepare the result values in the transmission format and propagate them to the client.

Distributed applications based on RPC

How to implement distributed applications based on remote procedure calls?

Distributed application

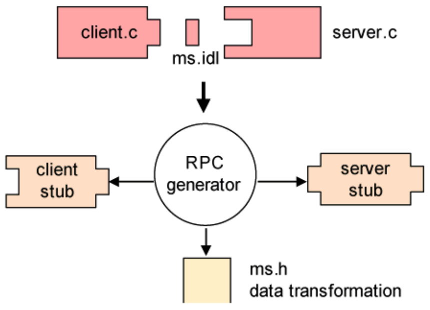
In order to isolate the communication idiosyncrasy of RPCs and to make the network interfaces transparent to the application programmer, so-called ***stubs*** are introduced.

RPC generator

An **RPC** generator

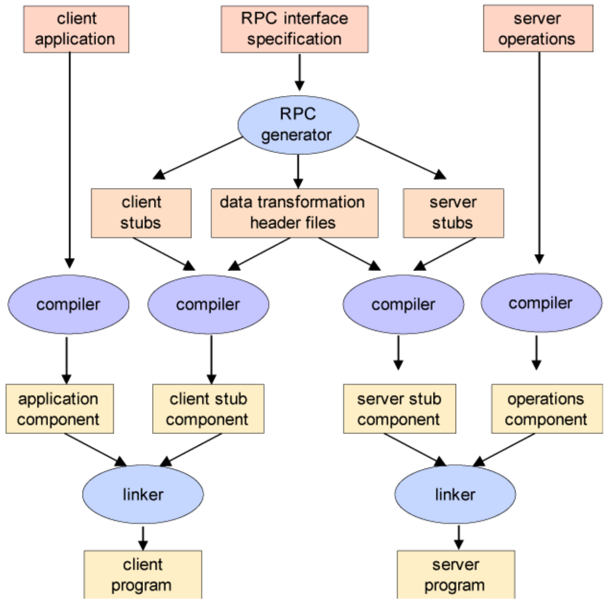
reduces the time necessary for implementation and management of the components of a distributed application.

a declarative interface description is easier to modify and therefore less error-prone.



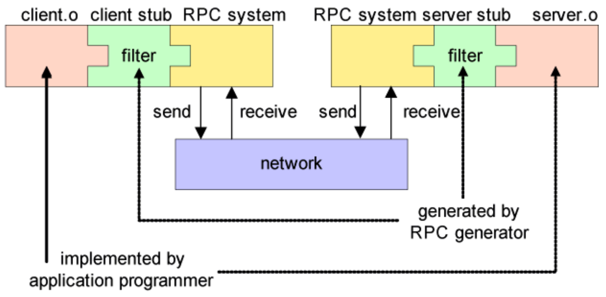
Applying the RPC generator

The individual steps for generating a distributed application are illustrated in the following figure.



Structure of a distributed application

The internal structure of a distributed application created using an **RPC** generator is as follows:



Implementing a distributed application

Manual implementation of stubs is error-prone ⇒ use of a ***RPC generator*** to generate stubs from a declarative specification.

[RPC generator]

[Applying the RPC generator]

[Structure of a distributed application]