REMOTE PROCEDURE CALLS

Overview

- The RPC Model
- Transparency
- Implementation
- Stub Generation
- RPC Messages
- Marshaling
- Server Management
- Call Semantics

RPC

- The RPC is an accepted IPC mechanism in distributed systems.

A remote procedure call is an interprocess communication technique that is used for client-server based applications.

It is also known as a subroutine call or a function call.

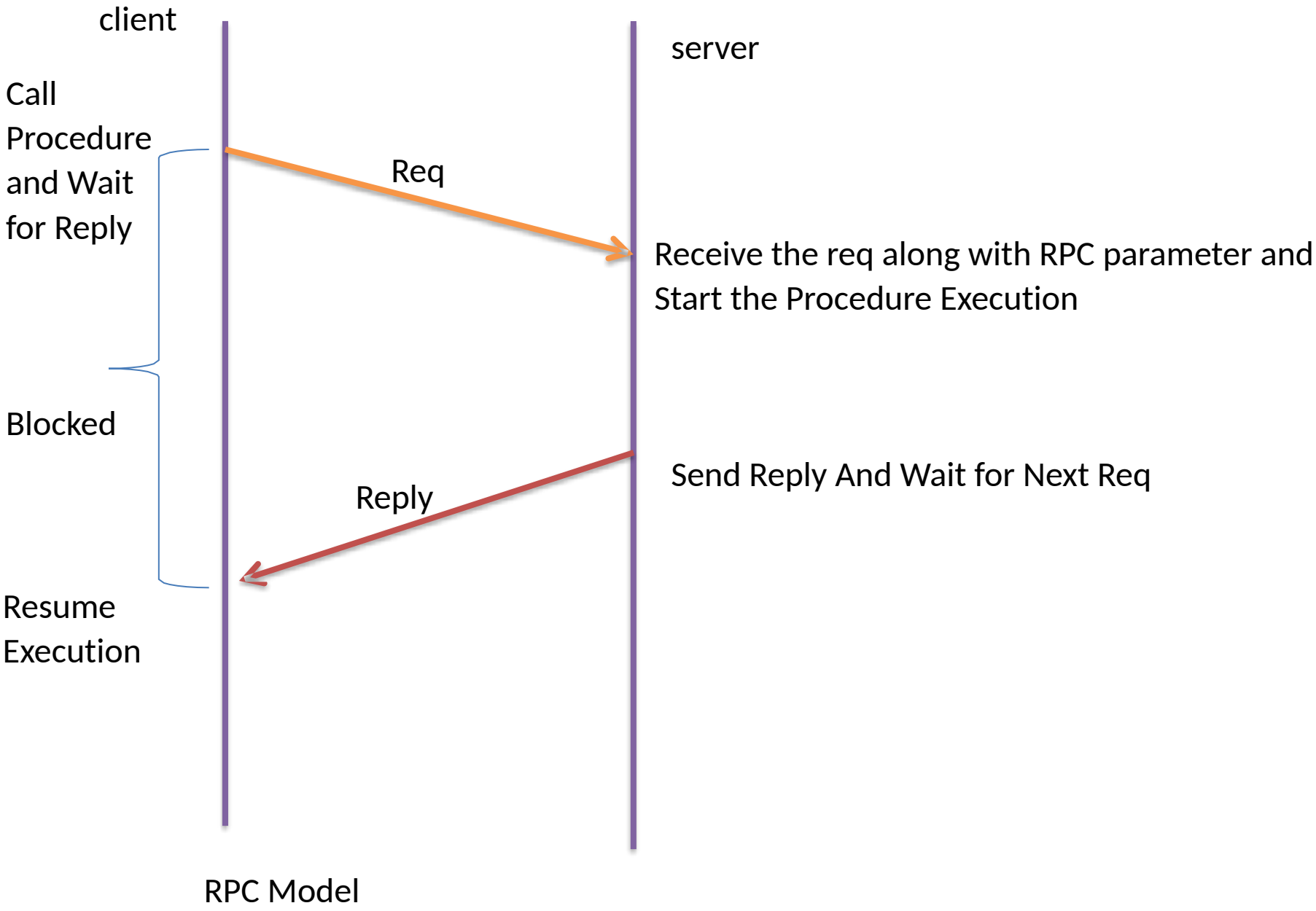
Features of RPC

- Simple call syntax.
- Familiar semantics - similar to local procedure calls
- A well-defined interface.

  - Supports compile-time type checking and automated interface generation.
- Its ease of use – Simple semantics
- Its generality
- Its efficiency.
- Facilitate to communicate between all processes

RPC MODEL

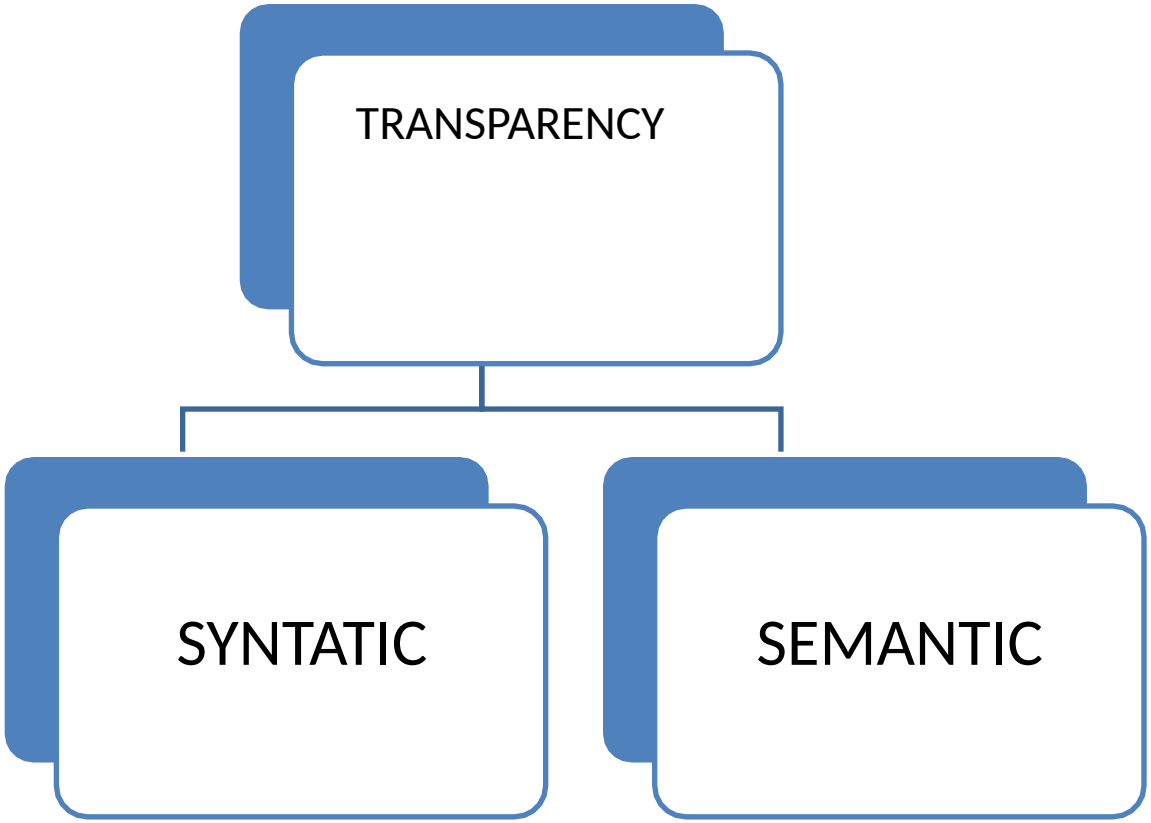
- The caller places arguments to the procedure in some well-specified location.
- Control is then transferred to the sequence of instructions that constitutes the body of the procedure.
- The procedure body is executed in a newly created execution environment
- After the execution is over, control returns to the calling point, With result.


client
server
Call
Procedure
and Wait
for Reply
Req
Receive the req along with RPC parameter and
Start the Procedure Execution
Blocked
Send Reply And Wait for Next Req
Reply
Resume
Execution

RPC Model

RPC

- Transparency of RPC
- Local procedures and remote procedures are indistinguishable to programmers.



RPC

- The calling process is suspended until the called procedure returns.
- The caller can pass arguments to the called procedure (Remote procedure).
- The called procedure (remote procedure) can return results to the caller.

Differences between remote procedure calls and local procedure calls

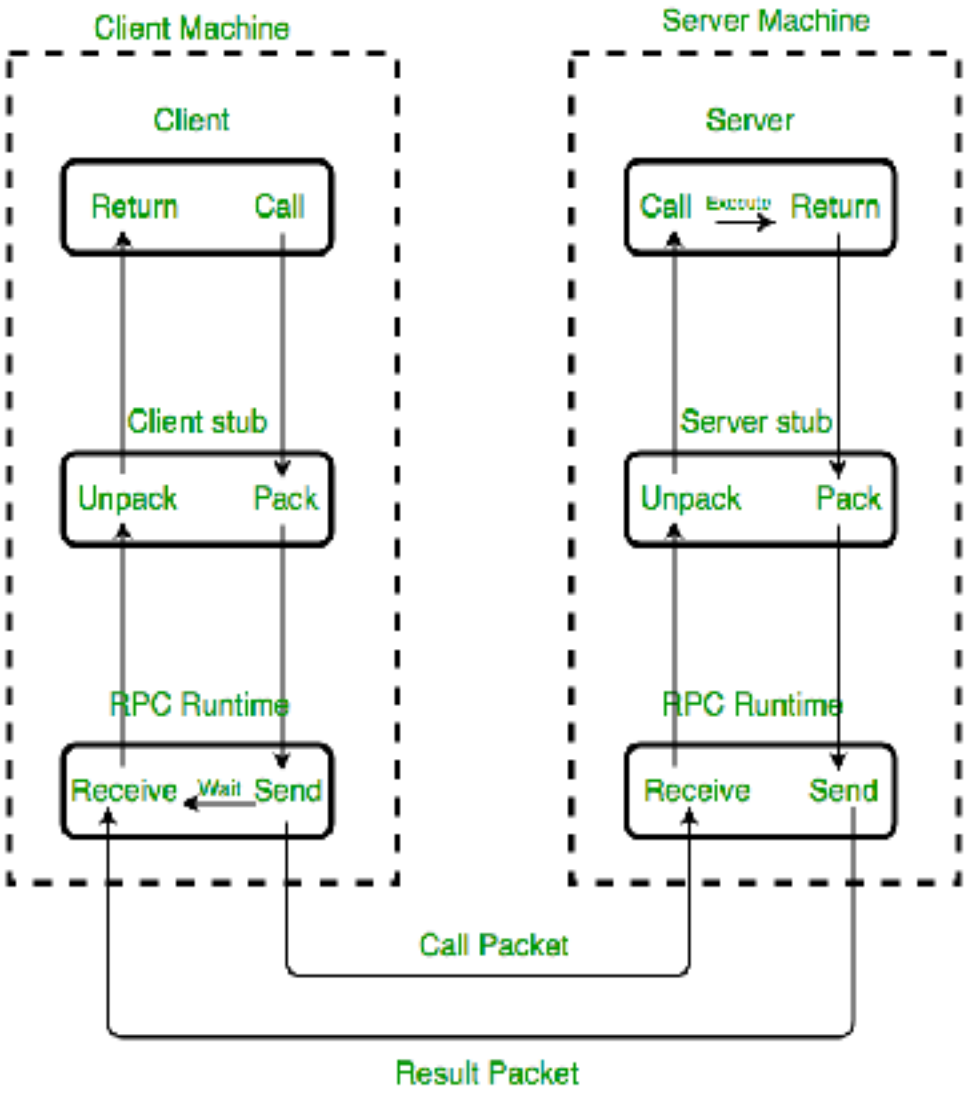
SL NO	LOCAL PROCEDURE CALLS	REMOTE PROCEDURE CALLS
1	same	Address space is disjoint from the calling program
2	Have access	called (remote) procedure cannot have access to any variables or data values in the calling program's environment.
3		it is meaningless to pass argument values containing pointer Structures
4		More vulnerable to failures
5		consume much more time (100-1000 times more) than local procedure calls due to communication n/w in RPC

IMPLEMENTING RPC MECHANISM

To achieve semantic transparency, implementation of RPC is based on the concepts of stubs, which provide a local procedure call abstraction

Five elements of program with RPC

1. The client
2. The client stub
3. The RPCRuntime
4. The server stub
5. The server



Implementation of RPC mechanism

IMPLEMENTING RPC MECHANISM

- **Client**
  - A user process that initiates a remote procedure call.
  - Makes a normal local call that invokes a procedure in the client stub.
- **Client Stub**

-A stub is a piece of code that converts parameters during a remote procedure call (RPC)

- Responsible for conversion (marshalling) of parameters and de conversion of results .

- packs a procedure and the arguments into a message and ask the local RPCRuntime to send it to the server stub.
- On receipt of the result of procedure execution, it unpacks the result and passes it to the client.

IMPLEMENTING RPC MECHANISM

- **RPC Runtime**
  - Handles transmission of messages between client and server.
  - Responsible for retransmissions, acknowledgments, packet routing, and encryption.
  - on the client machine receives the call request message from the client stub. Receives the result of procedure execution
  - on the server machine receives the message containing the result of procedure from the server stub and receives the call request.

IMPLEMENTING RPC MECHANISM

- **Server Stub**
  - On receipt of the call request message from the local RPCRuntime, the server stub unpacks it and makes a perfectly normal call to invoke the appropriate procedure in the server.
  - On receipt of the result of procedure execution from the server, the server stub packs the result into a message and then asks the local RPCRuntime to send it to the client stub.

IMPLEMENTING RPC MECHANISM

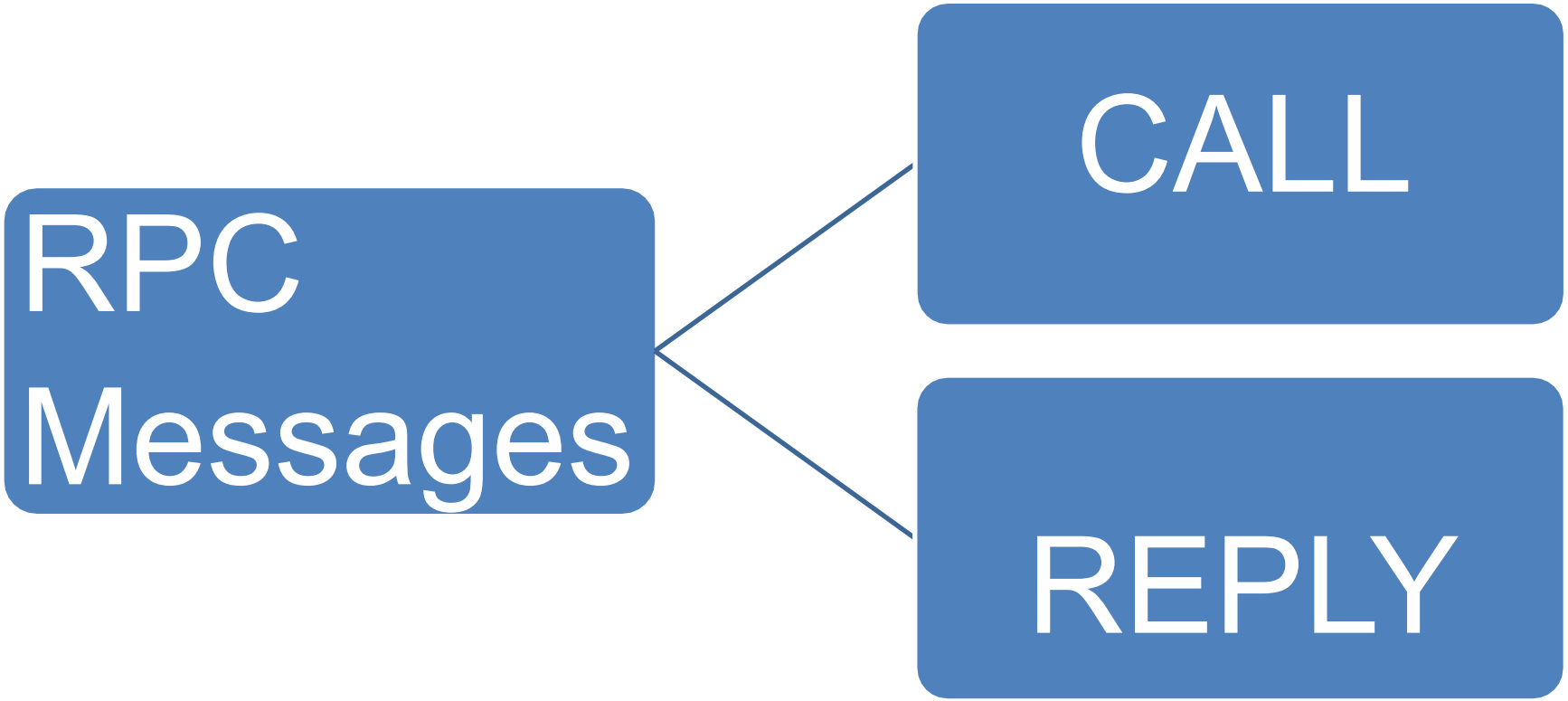
- **Server**
  - On receiving a call request from the server stub,
    - Executes the appropriate procedure and
    - Returns the result of procedure execution to the server stub

IMPLEMENTING RPC MECHANISM

STUB GENERATION

- *Manually.*
  - RPC implement or provides a set of translation functions
  - User can construct his or her own stubs.
  - Simple to implement and can handle very complex parameter types.
- *Automatically.*
  - Commonly used method for stub generation.
  - Uses *Interface Definition Language (IDL) for defining* the interface between a client and a server.

RPC MESSAGES



RPC MESSAGES

1. *Call Messages* that are sent by the client to the server for requesting execution of a particular remote procedure
2. *Reply Messages* that are sent by the server to the client for returning the result of Remote Procedure Execution

RPC MESSAGES

- The FIVE basic components necessary in a call message are as follows:
  - The identification information of the remote procedure to be executed
  - The **arguments** necessary for the execution of the procedure
  - A **message identification** field that consists of a sequence number
    identifying lost/duplicate messages
    For properly matching reply messages to outstanding call messages
- A **message type field** that is used to distinguish call messages from reply messages.
  - A **client identification field** for executing the concerned procedure - To allow the server to identify the client to send the reply message and to authenticate the client

RPC MESSAGES

- CALL MESSAGE

MESSGE IDENTIFIER	MESSAGE TYPE	CLIENT IDENTIFIER	REMOT PROCEDURE IDENTIFIER			Arguments
			PROGRAM NUMBER	VERSION NUMBER	PROCEDURE NUMBER	

RPC MESSAGES

- REPLY MESSAGES

1. Server finds that the call message is not intelligible to it. Happens when a call message violates the RPC protocol. The server rejects it.

2. Server scans the clients identification field and detects client is not authorized to use the service. The server will return an unsuccessful reply

3. The server finds that the remote program, version, or procedure number in the remote procedure identifier of the call message not available with it. Will return an unsuccessful reply

4. The remote procedure is not able to decode the supplied arguments.

5. An exception condition (such as division by zero)

RPC MESSAGES

SUCCESSFUL

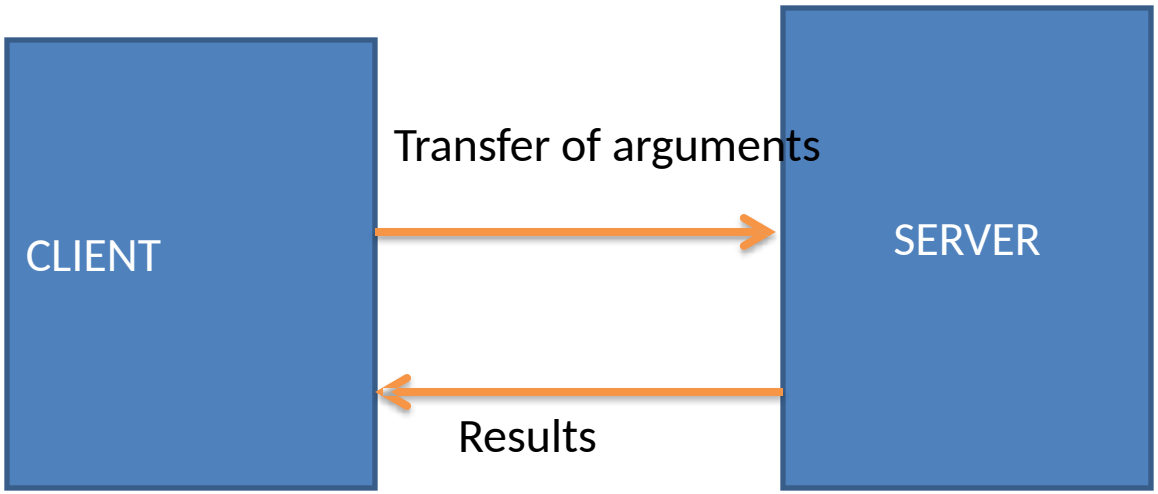
Meaage Identifier	Message Type	Reply Status (Successful)	Result

UNSUCCESFUL REPLY

Message Identifer	Message Type	Reply Status (UnSuccessful)	Reason for failure

Marshalling Arguments

- Implementation of remote procedure calls



Marshalling Arguments

- Transfer of message data requires encoding and decoding of the message data.
- For RPCs this operation is known as *Marshaling* and involves the following Actions:

1. Taking the arguments of a client process or the result of a server
1. Encoding the message data of step 1 above on the sender's computer.This encoding process involves the conversion of program objects into a stream form that is suitable for transmission and Placing them into a message buffer.
2. Decoding of the message data on the receiver's Computer.The reconstruction of program objects from the message data that was received in stream form.

Marshalling Arguments

- Marshalling may be
  1. Provided as a part of the RPC software- Marshalling procedures for scalar data types and compound types build from the scalar ones
  2. Those that are defined by the users of the RPC system- Marshalling procedures for user defined data types and dat types that include pointers
- A good RPC system

– generate in-line marshaling code for every remote call

– it is difficult to achieve this goal because of the large amounts of code