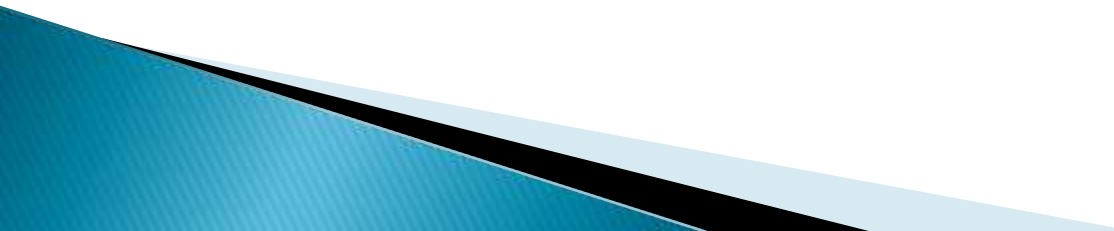


Remote Procedure Calls


Introduction:RPC

- ▶ RPC allows programs to call procedures located on other machines.
 - ▶ When a process on machine *A* *calls* a procedure on machine *B*, *the calling process on A is suspended, and execution of the called procedure takes place on B. Information can be transported from the caller to the callee in the parameters and can come back in the procedure result.*
 - ▶ No message passing at all is visible to the programmer. This method is known as **Remote Procedure Call**, or often just **RPC**.
- 

Introduction

- ▶ It can be said as the **special case of message-passing model**.
- ▶ It has become widely accepted because of the following features:
 - **Simple call syntax** and similarity to local procedure calls.
 - Its **ease of use, efficiency** and **generality**.
 - It can be used as an IPC mechanism between
 - processes on different machines and
 - also between different processes on the same machine.

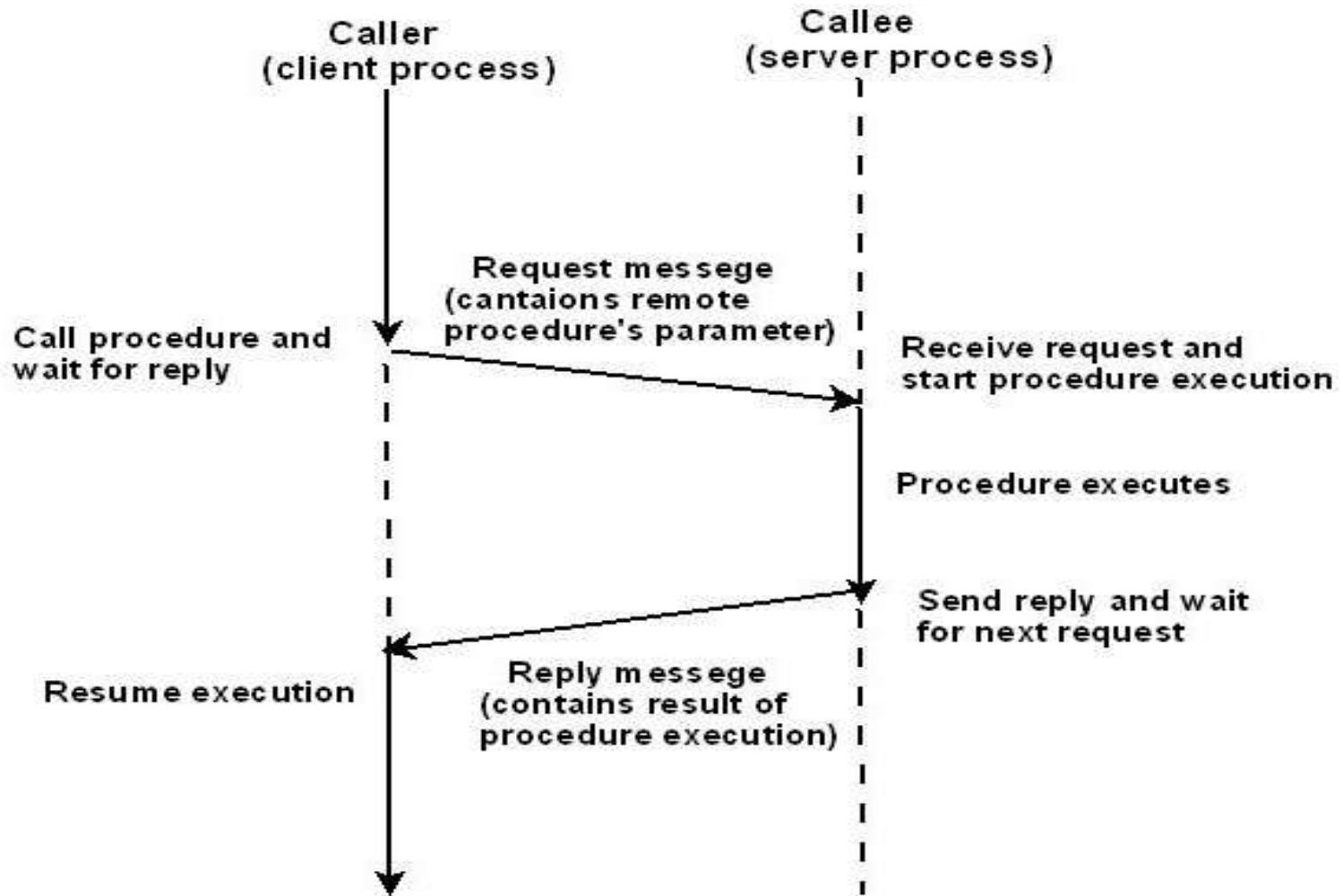
RPC Model

- ▶ It is similar to commonly used procedure call model. It works in the following manner:
 1. For making a procedure call, the **caller places arguments** to the procedure in some well specified location.
 2. **Control is then transferred** to the sequence of instructions that constitutes the body of the procedure.
 3. The **procedure body is executed** in a newly created execution environment that includes copies of the arguments given in the calling instruction.
 4. After the procedure execution is over, **control returns** to the calling point, returning a result.
- 

Cont...


- ▶ The RPC enables a call to be made to a procedure that does not reside in the address space of the calling process.
- ▶ Since the caller and the callee processes have disjoint address space, the remote procedure has no access to data and variables of the callers environment.
- ▶ RPC facility uses a message-passing scheme for information exchange between the caller and the callee processes.
- ▶ On arrival of request message, the server process
 - extracts the procedure's parameters,
 - computes the result,
 - sends a reply message, andthen awaits the next call message.

Cont...




Remote procedure call model

Cont...

- ▶ Only one of the two processes is active at any given time.
 - ▶ It is not always necessary that the caller gets blocked.
 - ▶ There can be RPC implementations depending on the parallelism of the caller and the callee's environment.
 - ▶ The RPC could be asynchronous, so that the client may do useful work while waiting for the reply from the server.
 - ▶ Server could create a thread to process an incoming request so that server is free to receive other requests.
- 

Cont...

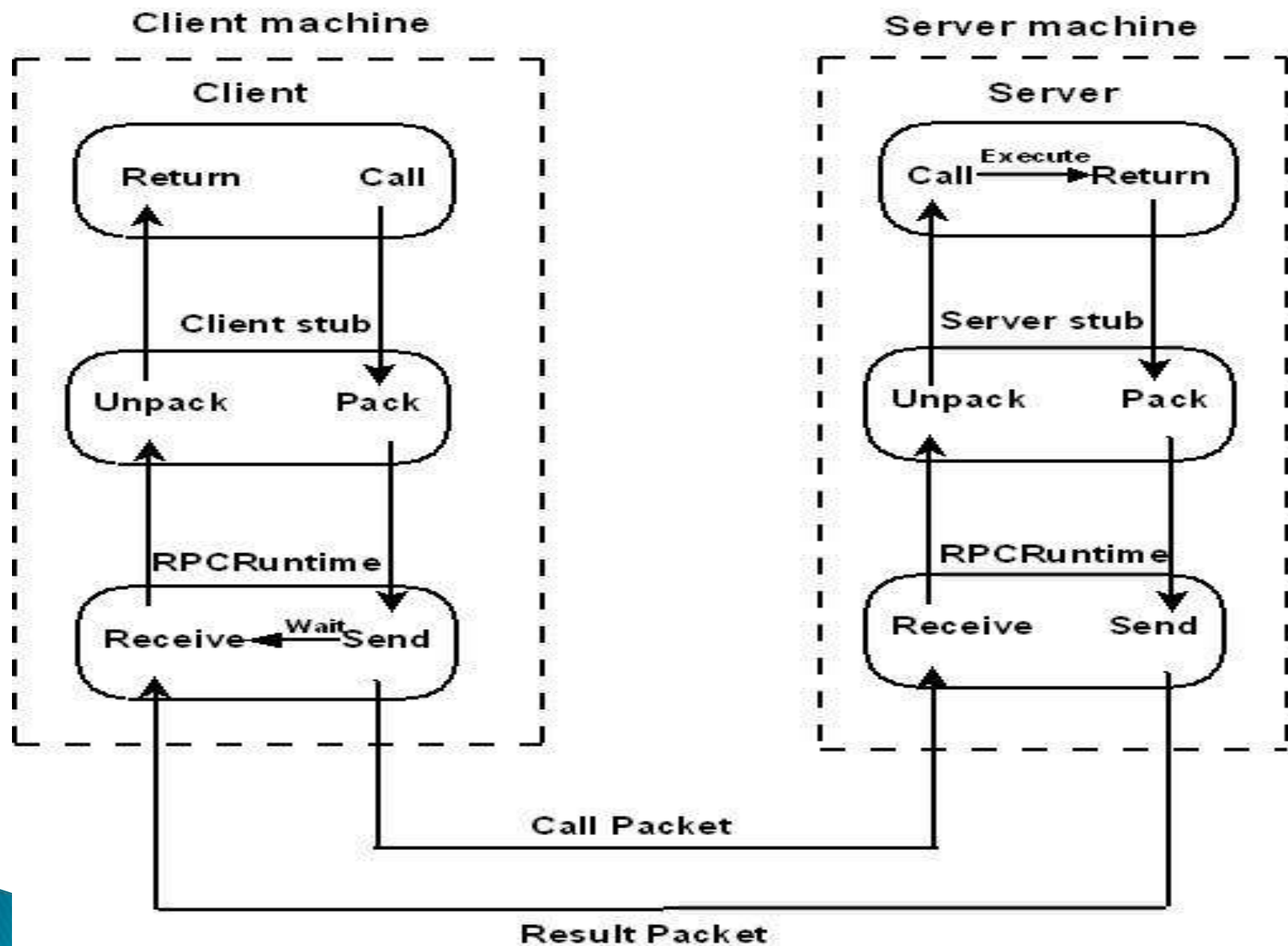
- ▶ Difference between remote procedure calls and local procedure calls:
 1. Unlike local procedure calls, with remote procedure calls,
 - Disjoint Address Space
 - Absence of shared memory.
 - Meaningless making call by reference, using addresses in arguments and pointers.
 2. RPC's are more vulnerable to failure because of:
 - Possibility of processor crashes or
 - communication problems of a network.
 3. RPC's are much more time consuming than LPC's due to the involvement of communication network.
- 

Implementing RPC Mechanism

- ▶ Implementation of RPC involves the five elements of program:
 1. Client
 2. Client Stub
 3. RPC Runtime
 4. Server stub
 5. Server
- The client, the client stub, and one instance of RPCRuntime execute on the client machine.
- The server, the server stub, and one instance of RPCRuntime execute on the server machine.
- As far as the client is concerned, remote services are accessed by the user by making ordinary LPC

▶ Stubs

- Provide a normal / local procedure call abstraction by concealing the underlying RPC mechanism.
- A separate stub procedure is associated with both the client and server processes.
- To hide the underlying communication network, RPC communication package known as **RPC Runtime** is used on both the sides.



Implementation of RPC mechanism

Client Stub

- ▶ It is responsible for the following two tasks:
 - On receipt of a call request from the client,
 - it packs a specifications of the target procedure and the arguments into a message and
 - asks the local RPC Runtime 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.

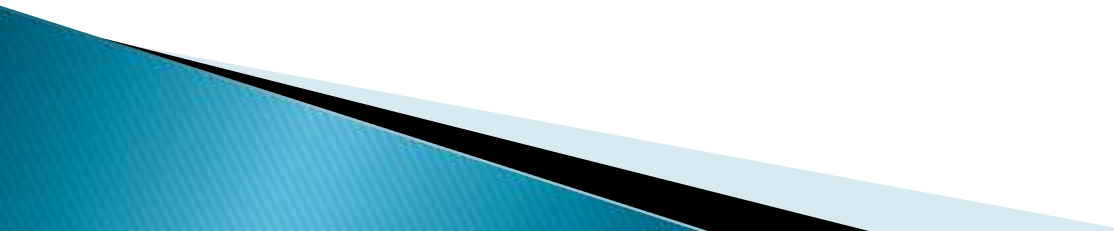
RPCRuntime

- ▶ It handles transmission of messages across the network between Client and the server machine.
- ▶ It is responsible for
 - Retransmission,
 - Acknowledgement,
 - Routing and
 - Encryption.

Server Stub

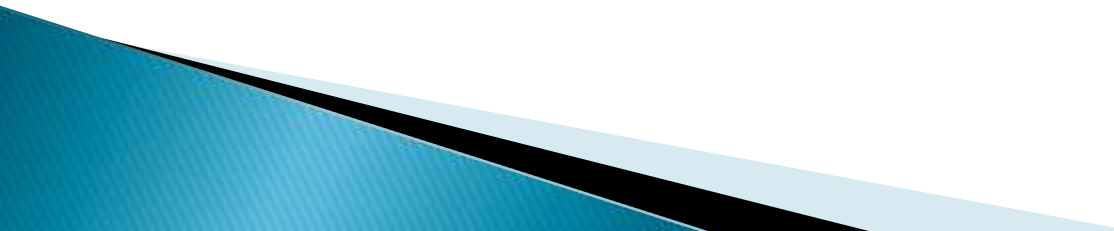
- ▶ It is responsible for the following two tasks:
 - On receipt of a call request message from the local RPCRuntime, it unpacks it and makes a perfectly normal call to invoke the appropriate procedure in the server.
 - On receipt of the result of procedure execution from the server, it unpacks the result into a message and then asks the local RPCRuntime to send it to the client stub.

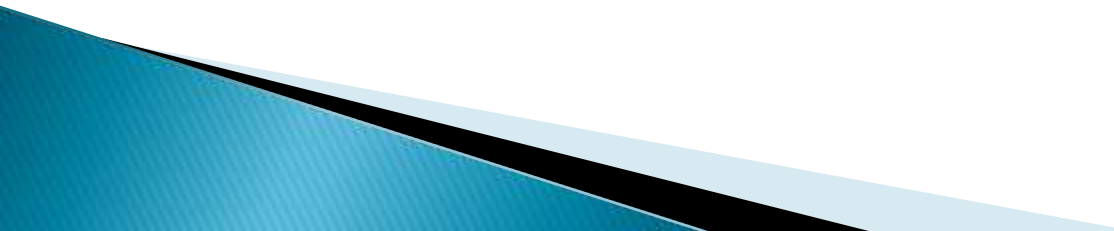
Marshalling Arguments and Results

- ▶ The arguments and results in RPC are language-level data structures, which are transferred in the form of message data.
 - ▶ Transfer of data between two computers requires encoding and decoding of the message data.
 - ▶ Encoding and decoding of messages in RPC is known as **marshaling & Unmarshaling**.
- 

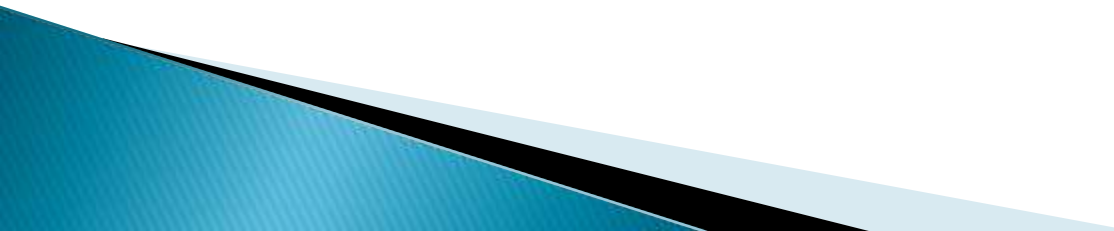
Working

To summarize, a remote procedure call occurs in the following steps:

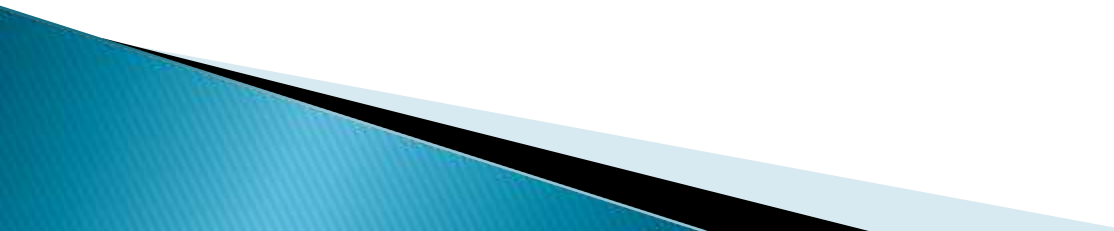
1. The client procedure calls the client stub in the normal way.
 2. The client stub builds a message and calls the local operating system.
 3. The client's OS sends the message to the remote OS.
 4. The remote OS gives the message to the server stub.
 5. The server stub unpacks the parameters and calls the server.
- 

6. The server does the work and returns the result to the stub.
 7. The server stub packs it in a message and calls its local as.
 8. The server's as sends the message to the client's as.
 9. The client's as gives the message to the client stub.
 10. The stub unpacks the result and returns to the client.
- 

Call-by-value

- ▶ In the call-by-value method, all parameters are copied into a message that is transmitted from the client to the server through the network.
 - ▶ It is time consuming for passing large data types such as trees, arrays, etc.
- 

Call-by-reference

- ▶ Most RPC mechanisms does not use the call-by-reference semantics for parameter passing when client and the server exist in different address space.
 - ▶ Distributed systems having distributed shared-memory mechanisms can allow passing of parameters by reference.
- 

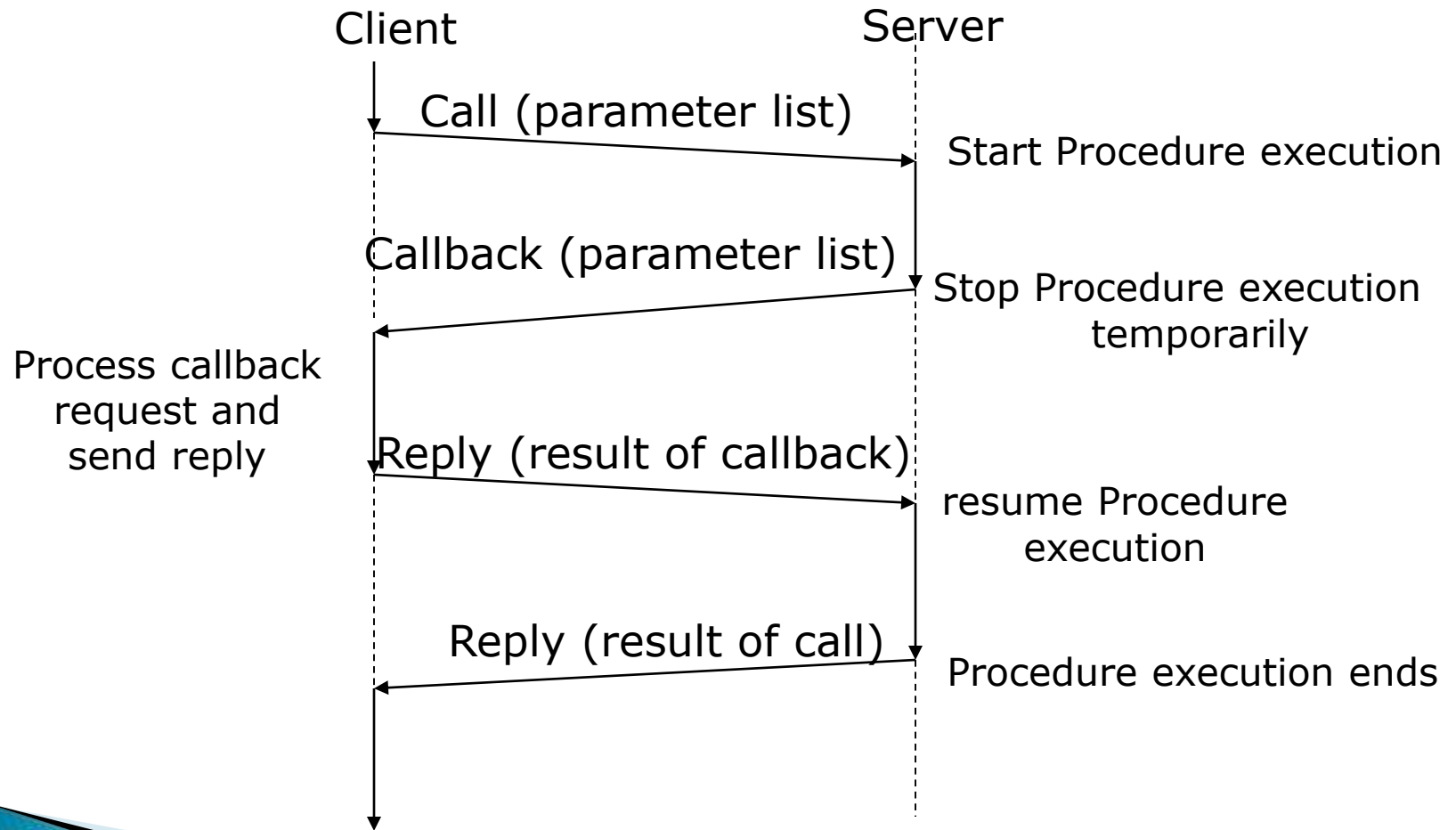
Some special types of RPCs

- ▶ Callback RPC
- ▶ Broadcast RPC
- ▶ Batch-mode RPC

Callback RPC

- ▶ It facilitates a peer-to-Peer paradigm among participating processes.
- ▶ It allows a process to be both a client and a server.
- ▶ Remotely processed interactive Application
- ▶ Issues
 - Providing server with clients handle
 - Making the client process wait for the callback RPC
 - Handling callback deadlocks

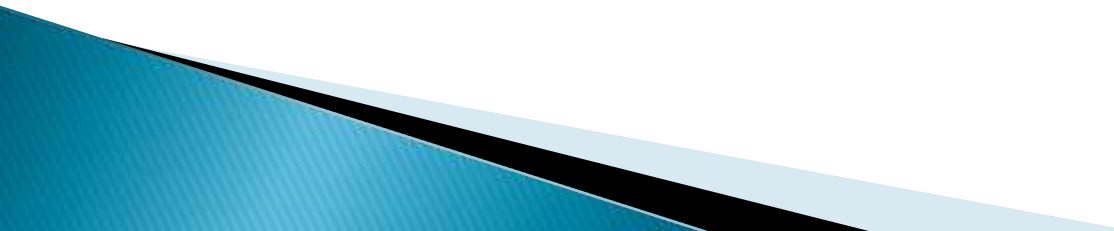
Cont...



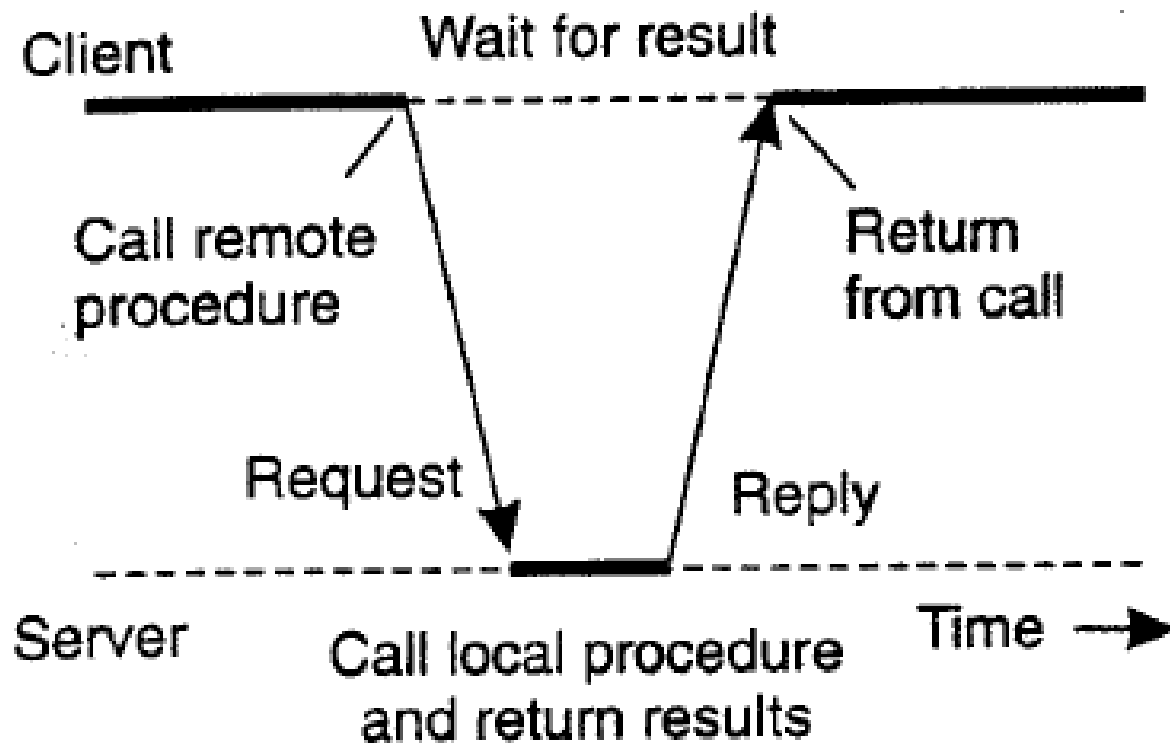
Broadcast RPC

- ▶ A client's request is broadcast on the network and is processed by all servers that have the procedure for processing that request.
- ▶ Methods for broadcasting a client's request:
 1. Use of broadcast primitive to indicate that the client's request message has to be broadcasted.
 2. Declare broadcast ports.
- Back-off algorithm can be used to increase the time between retransmissions.
 - It helps in reducing the load on the physical network and computers involved.

Batch-mode RPC

- ▶ Batch-mode RPC is used to queue separate RPC requests in a transmission buffer on the client side and then send them over the network in one batch to the server.
 - It reduces the overhead involved in sending requests and waiting for a response for each request.
 - It is efficient for applications requiring lower call rates and client doesn't need a reply.
 - It requires reliable transmission protocol.
- 

Synchronous RPC



Asynchronous RPC

