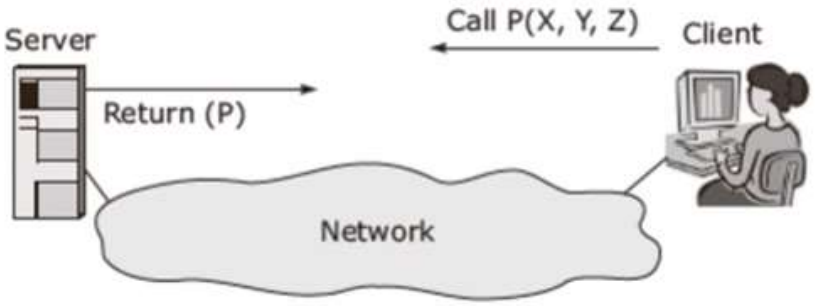
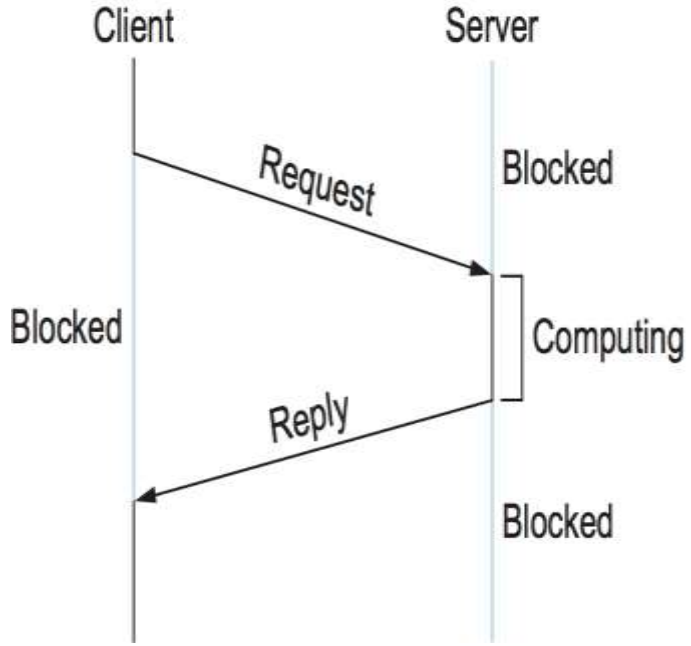
Remote Procedure Calls

Remote Procedure Call

- A common pattern of communication used by application programs structured as a *client/server* pair is the request/reply message transaction:
- A client sends a request message to a server, and the server responds with a reply message, with the client blocking (suspending execution) to wait for the reply.
- Figure illustrates the basic interaction between the client and server in such an exchange.

Basic RPC operation





Procedure

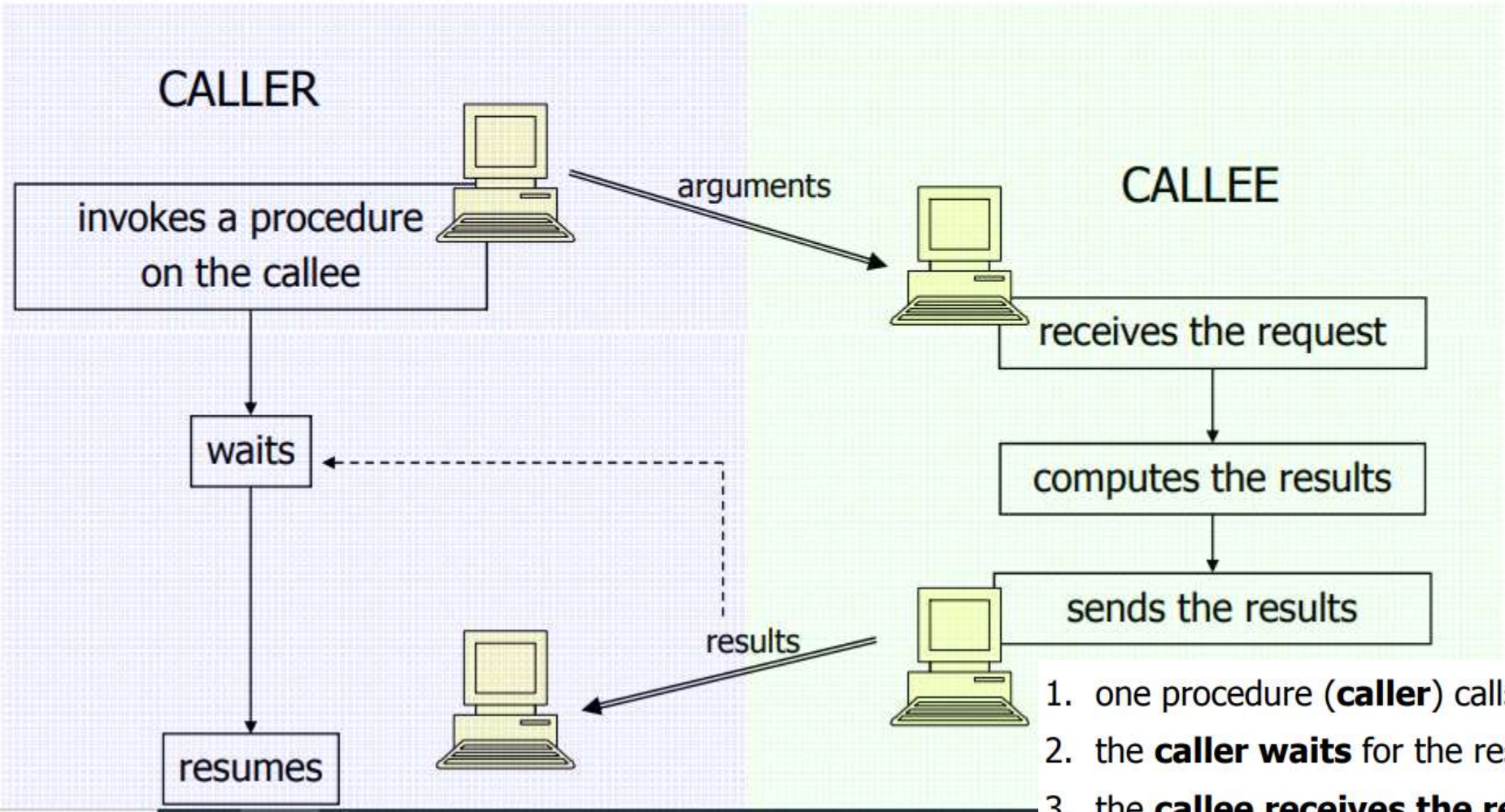
- Same as *routine*, *subroutine,* and *function*. A procedure is a section of a program that performs a specific task.

- An ordered set of tasks for performing some action.

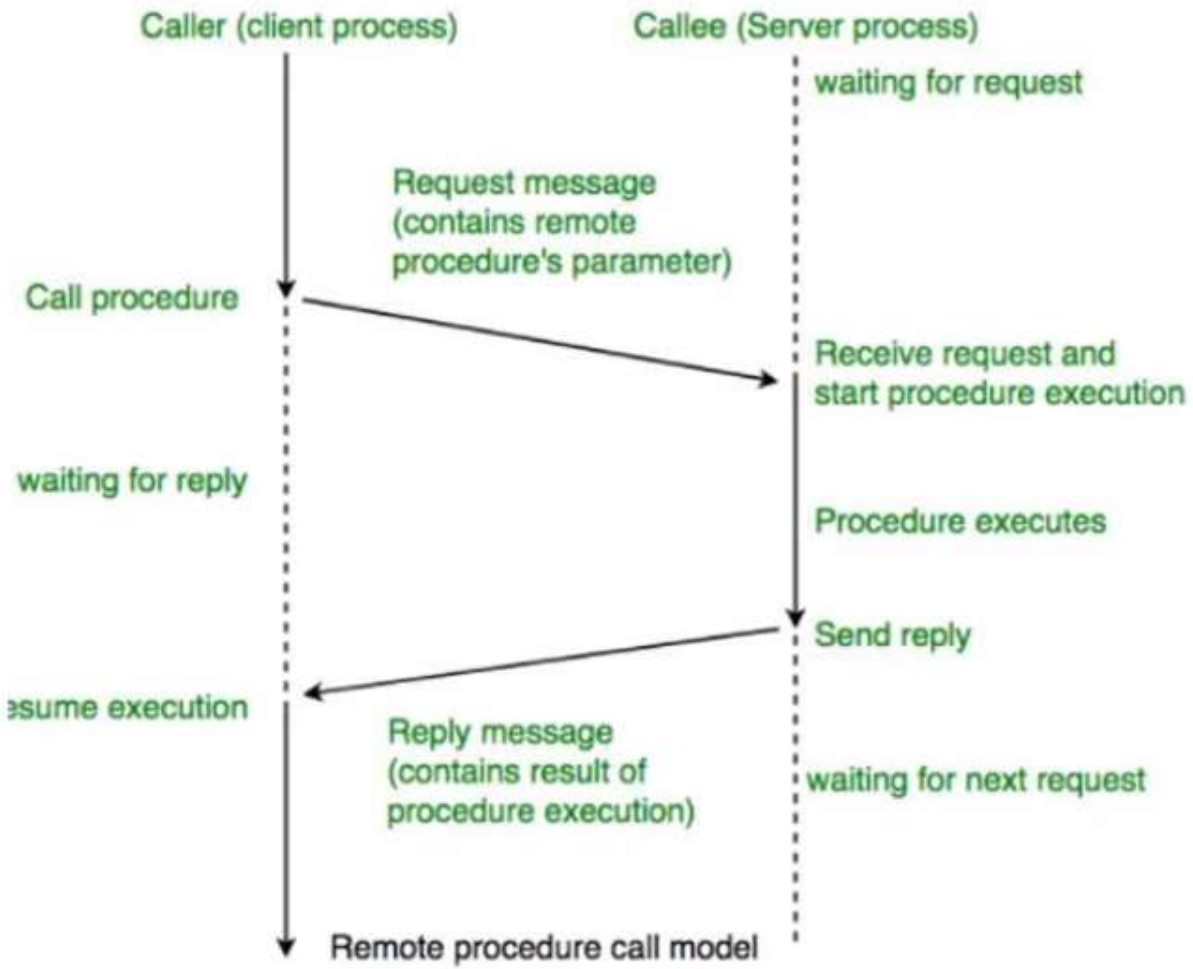
Remote Procedure Call

- Remote procedure call is a protocol that one program can use to request a service from a program located in another computer or network.
- Remote Procedure Call (RPC) is a powerful technique for constructing distributed, client-server based applications. It is also known as a subroutine call or a function call.
- A client has a request message that the RPC translates and sends to the server. This request may be a procedure or a function call to a remote server. When the server receives the request, it sends the required response back to the client. The client is blocked while the server is processing the call and only resumed execution after the server is finished.
- It is based on extending the conventional local procedure calling so that the called procedure need not exist in the same address space as the calling procedure. The two processes may be on the same system, or they may be on different systems with a network connecting them.

What are Remote Procedure Calls (RPCs)



1. one procedure (**caller**) calls another procedure (**callee**)
2. the **caller waits** for the result from the callee
3. the **callee receives the request**, computes the results, and then **send them to the caller**
4. the **caller resumes** upon receiving the results from the callee

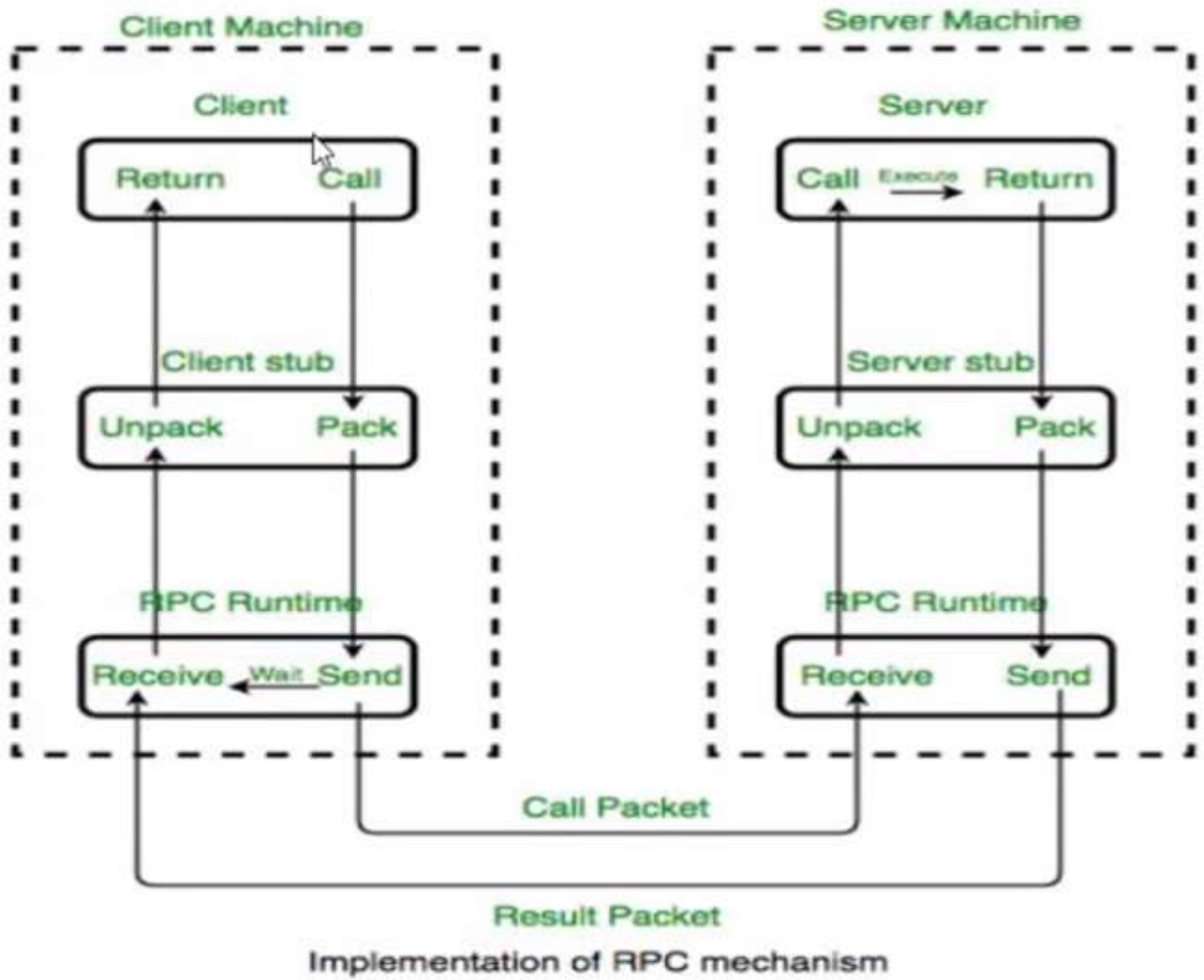

Caller (client process)
Callee (Server process)
waiting for request
Request message
(contains remote
procedure's parameter)
Call procedure
Receive request and
start procedure execution
waiting for reply
Procedure executes
Send reply
esume execution
Reply message
(contains result of
procedure execution)
waiting for next request
Remote procedure call model

The caller or client process sends a call request message to the callee or server process and then waits for a reply. The request message consists of remote procedure parameters.

On receiving the message, the server process executes the procedure and returns a result in a reply message that is sent to client process.

When the client receives the reply message, the result of the execution is extracted and the caller's execution is resumed.

Working of RPC



Implementation of RPC mechanism

steps to take during a RPC :

A client invokes a client stub procedure, passing parameters in the usual way. The client stub resides within the client's own address space.

The client stub pack the parameters into a message. packing includes converting the representation of the parameters into a standard format, and copying each parameter into the message.

The client stub passes the message to the transport layer, which sends it to the remote server machine.

On the server, the transport layer passes the message to a server stub, which unpack the parameters and calls the desired server routine using the regular procedure call mechanism.

When the server procedure completes, it returns to the server stub (e.g., via a normal procedure call return), which pack the return values into a message. The server stub then hands the message to the transport layer.

The transport layer sends the result message back to the client transport layer, which hands the message back to the client stub.

The client stub unpack the return parameters and execution returns to the caller.

- **Client:** The client process initiates RPC. The client makes a standard call, which triggers a correlated procedure in the client stub.
- **Client Stub:** The client calls the client stub. Client stub does the following tasks:
  - The first task performed by client stub is when it receives a request from a client, it packs(marshalls) the parameters and required specifications of remote/target procedure in a message.
  - The second task performed by the client stub is upon receiving the result values after execution, it unpacks (unmarshalled) those results and sends them to the Client.

- **RPC Runtime:** The RPC runtime is in charge of message transmission between client and server via the network.
- Retransmission, acknowledgement, routing, and encryption are all tasks performed by it.
- On the client-side, it receives the result values in a message from the server-side, and then it further sends it to the client stub whereas, on the server-side, RPC Runtime got the same message from the server stub when then it forwards to the client machine. It also accepts and forwards client machine call request messages to the server stub.
- **Server Stub:** Server stub does the following tasks:
  - The first task performed by server stub is that it unpacks(unmarshalled) the call request message which is received from the local RPC Runtime and makes a regular call to invoke the required procedure in the server.
  - The second task performed by server stub is that when it receives the server's procedure execution result, it packs it into a message and asks the local RPC Runtime to transmit it to the client stub where it is unpacked.

- **Server:** After receiving a call request from the client machine, the server stub passes it to the server. The execution of the required procedure is made by the server and finally, it returns the result to the server stub so that it can be passed to the client machine using the local RPC Runtime.

RPC ISSUES

Asynchronous communication: The most outstanding restriction of the RPC protocol is the lack of support for asynchronous communication .This restriction forces the client process that invokes the RPC to a wait-state until the server has completed the results. This in turn limits the degree of communication concurrency in any distributed environment.

Difference in data Representation: Some system used the high memory address for storing most significant number and some systems used the low memory address for storing the high significant number.

Server Selections and Failure Recovery: The ability to handle abnormal situations is a fundamental requirement for any distributed system. The abnormalities could be due to either software or hardware components and may occur at any system level. In all cases, the behaviour of a distributed transaction may be placed in one of the following categories:

a)Normal: The transaction has terminated as expected. In this category, data may or may not necessarily be returned by the server even though the operation is considered normal. As an example, if an invalid request is received by a server, the server may issue an error message questioning the validity of the request which is a correct response in this situation.

Retry/Time-out: The transaction has timed out due to slower than expected response time of one or more of the system components. The option of retrying the transaction or voluntarily terminating the transaction must be offered to the client process.

c)Reconnection: The transaction has abnormally terminated due to unavailability or failure of one or more of the system components such as network connections, servers, or databases. The system should automatically attempt to complete the transaction via alternative routes

d) Hard Fail: The system has failed to complete the distributed transaction and will not allow any further attempts, unlike the reconnect case, through alternative routes. This is usually the case when the system detects that a crucial component of the final destination is not available regardless of the traversed route.

The RPC protocol does not support any of the above cases with the exception of the normal case which it fully supports. For other cases above, RPC informs the client of the status. The client process may choose to invoke a new RPC to retry the same transaction over or select an alternative route manually.

MESSAGE QUEUING:

In distributed systems, concurrent delivery of data and messages is essential regardless of the underlying communication protocol. Consider the messaging mechanism of a client/server environment that uses RPC as communication protocol. An RPC is invoked by the client process which triggers a sequence of activities that involve multiple computing platforms and network layers. The activities begin at the client platform by packaging and sending remote requests across Local Area Network (LAN) and/or Wide Area Network mediums.

PERFORMANCE: client has to wait till he didn't get the response form server

Binding: How does the client know who to call, and where the service resides?
The most flexible solution is to use dynamic binding and find the server at run time when the RPC is first made. The first time the client stub is invoked, it contacts a name server to determine the transport address at which the server resides.

Binding consists of two parts:

Naming

Locating

The call semantics associated with RPC :
It is mainly classified into following choices-

Retry request message –
Whether to retry sending a request message when a server has failed or the receiver didn't receive the message.

Duplicate filtering –
Remove the duplicate server requests.

Retransmission of results –
To resend lost messages without re-executing the operations at the server side.

Advantages of Remote Procedure Call

- RPC method helps clients to communicate with servers by the conventional use of procedure calls in high-level languages.
- Remote procedure calls support process oriented and thread oriented models.
- The internal message passing mechanism of RPC is hidden from the user.
- With RPC code re-writing / re-developing effort is minimized.

- Remote procedure calls can be used in distributed environment as well as the local environment.
- Process-oriented and thread oriented models supported by RPC.
- RPC method is modelled on the local procedure call, but the called procedure is most likely to be executed in a different process and usually a different computer.

Disadvantages of Remote Procedure Call

- The remote procedure call is a concept that can be implemented in different ways. It is not a standard.
- There is no flexibility in RPC for hardware architecture. It is only interaction based.
- There is an increase in costs because of remote procedure call.
- Remote procedure calling (and return) time (i.e., overheads) can be significantly lower than that for a local procedure.
- This mechanism is highly vulnerable to failure as it involves a communication system, another machine, and another process.