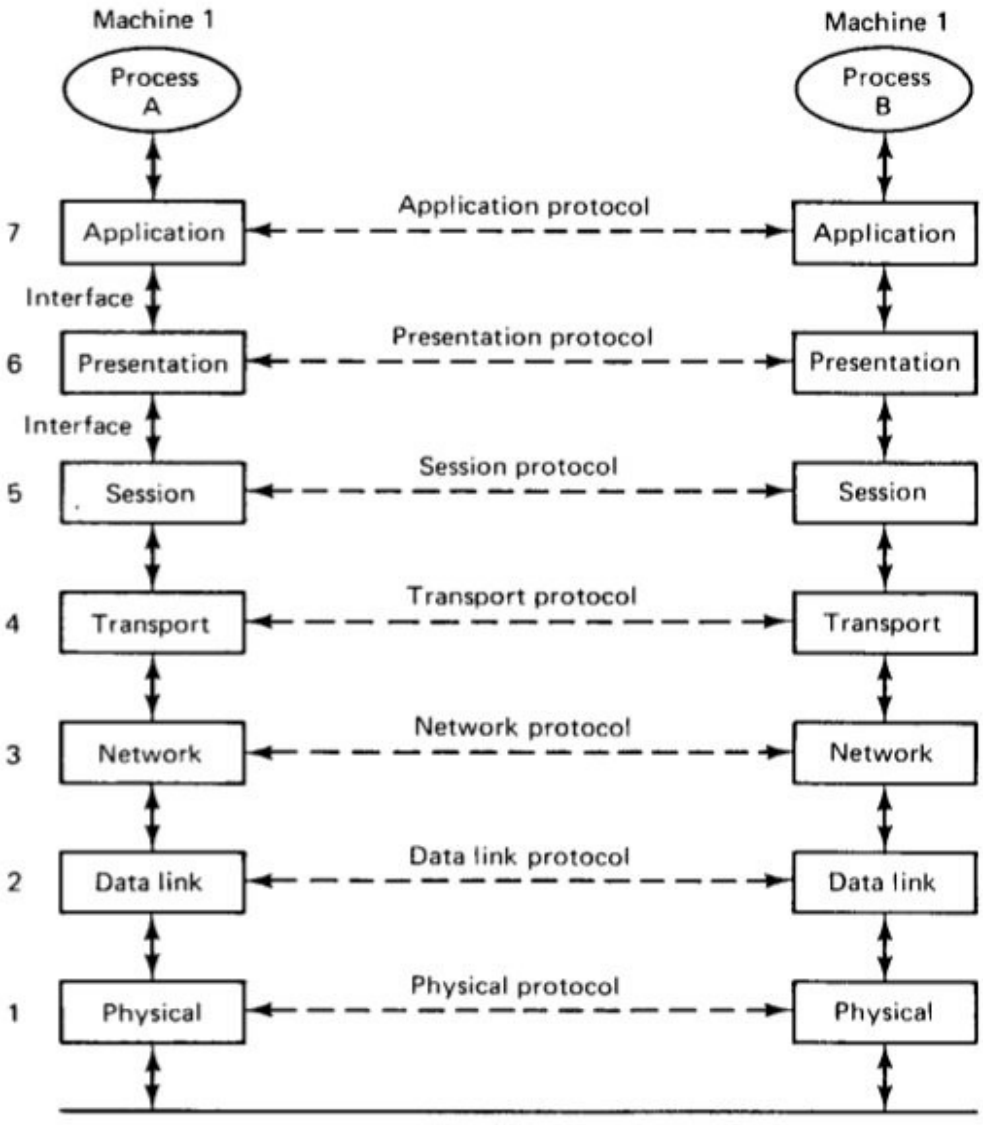
# COMMUNICATION IN DISTRIBUTED SYSTEM AND REMOTE PROCEDURE CALL

## *Communication in Distributed System:*

**Layered Communication:**

All communication in distributed system is based on message passing. When a process A wants to communicate with a process B it first builds a message in its own address space. Then it executes a system call that causes the operating system to fetch the message and send it over the network to B. To make it easier to deal with numerous levels and issues involved in communication, the international Standards Organization has developed a reference model, that clearly identifies the various level involved, gives them standard names and points out which level should do which job. This model is called OSI (Open System Interconnection Reference Model).

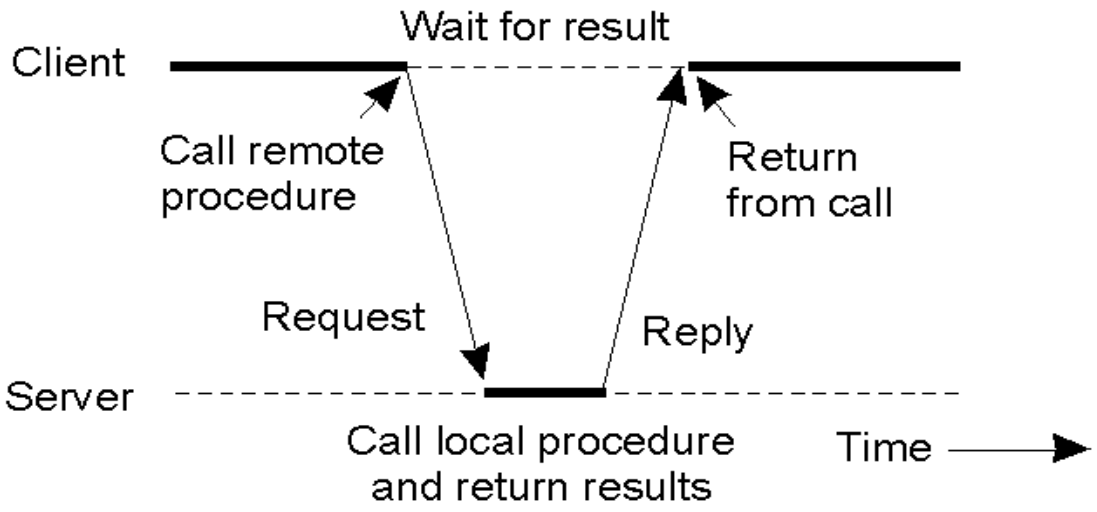


## RPC:

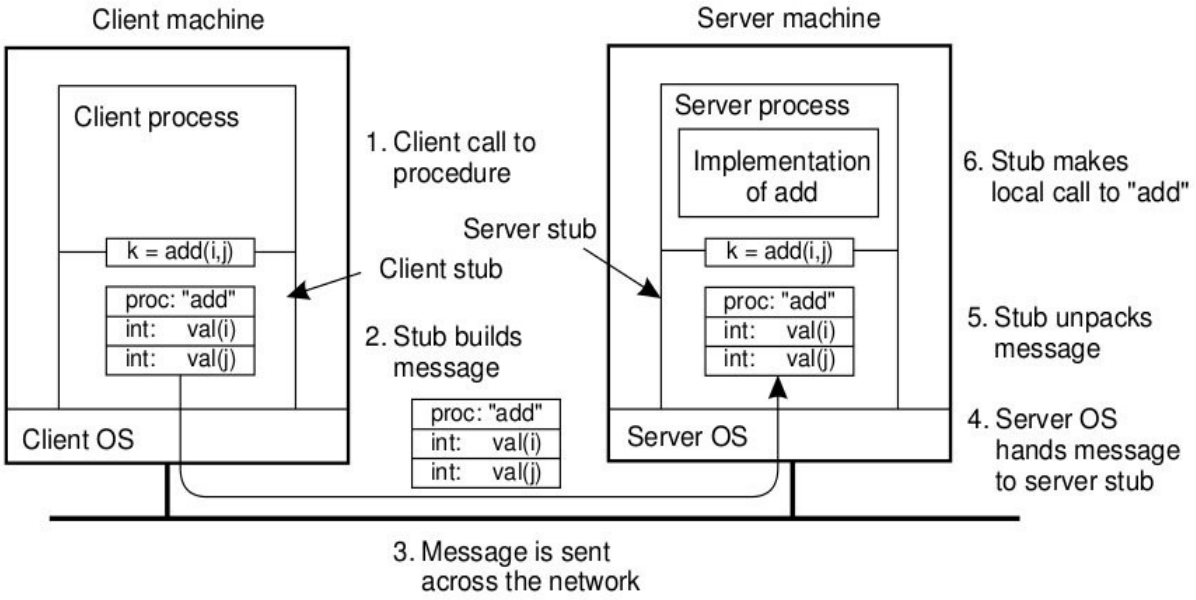
A remote procedure call (RPC) is an inter-process communication that allows a computer program to cause a subroutine or procedure to execute in another address space (commonly on another computer on a shared network) without the programmer explicitly coding the details for this remote interaction. That is, the programmer writes essentially the same code whether the subroutine is local to the executing program, or remote. When the software in question uses object-oriented principles, RPC is called remote invocation or remote method invocation.

**Principle of RPC:**

- Client makes procedure call (just like a local procedure call) to the client stub
- Server is written as a standard procedure
- Stubs take care of packaging arguments and sending messages
- Packaging parameters is called marshalling
- Stub compiler generates stub automatically from specs in an Interface Definition Language (IDL)
- Simplifies programmer task



Example of RPC:




**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 OS.
8. The server's OS sends the message to the client's OS.
9. The client's OS gives the message to the client stub.
10. The stub unpacks the result and returns to the client.

The net effect of all these steps is to convert the local call by the client procedure to the client stub, to a local call to the server procedure without either client or server being aware of the intermediate steps.

Source : http://dayaramb.files.wordpress.com/2012/02/operating-system-pu.pdf