Remote Procedure Calls

Remote Procedure Calls

- Procedures send and receive do not conceal communication for access transparency
- Birrell and Nelson suggested to allow 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
  - Execution of the called procedure takes place on B.
  - Information can be transported from the caller to the callee in parameters
  - Information can come back in the procedure result.
  - No message passing at all is visible to the programmer.
    - Constitutes the idea of RPC.

Remote Procedure Calls

- Conventional Procedure Call:
- count =read(fd, buf, nbytes);

- Data is passed using:
- Call by value,
- Call by reference
- Call by copy or restore.

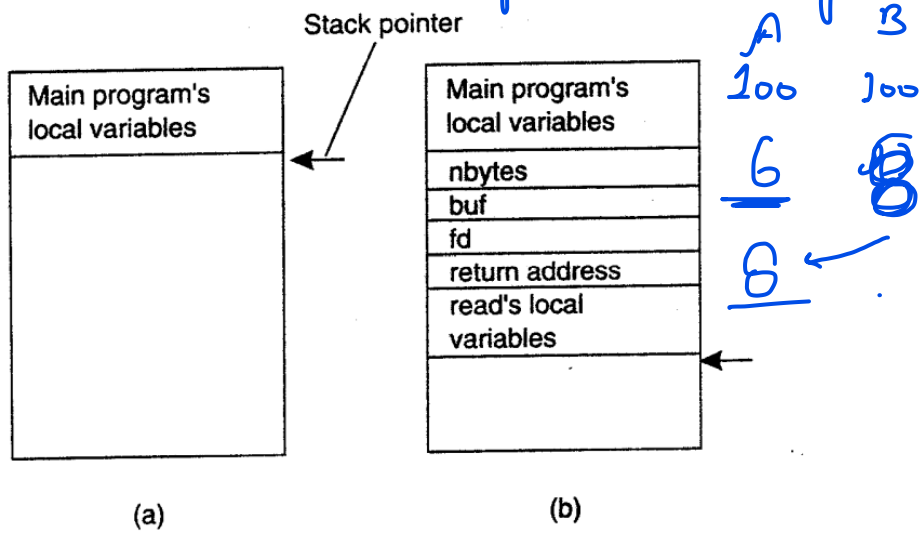


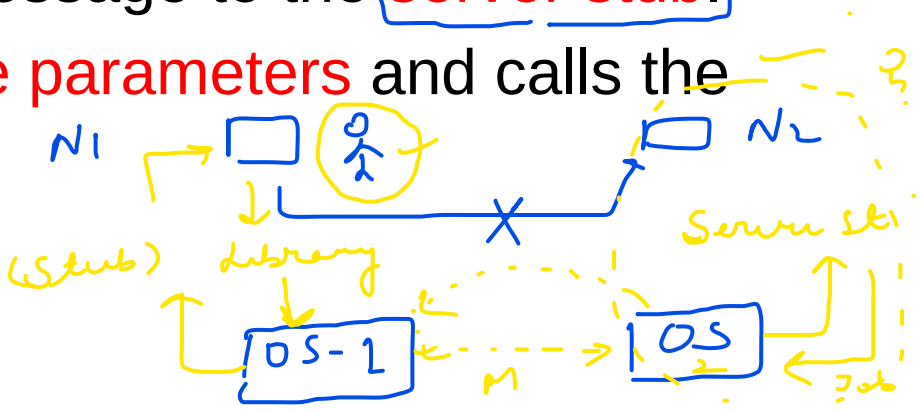
Figure 4-5. (a) Parameter passing in a local procedure call: the stack before the call to read. (b) The stack while the called procedure is active.

- RPC relies on same idea.
- Caller must not know that called procedure lies in separate machine

Remote Procedure Calls

A remote procedure call occurs in the following steps:

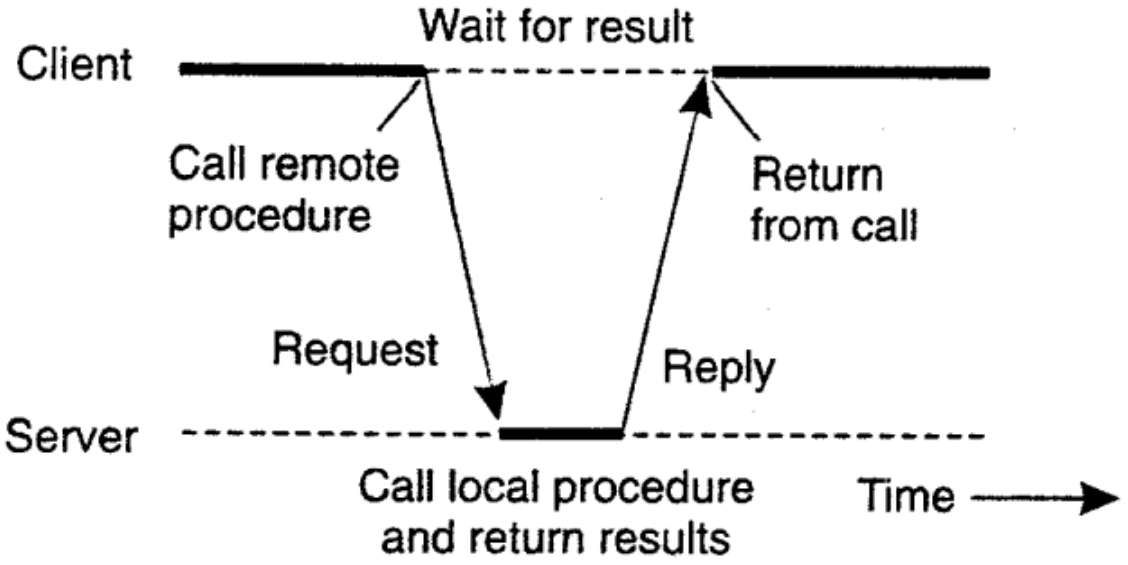
1. Read is a remote procedure.
2. The client procedure calls the client stub (in library) in the normal way.
3. The client stub builds a message and calls the local operating system.
4. The client's OS sends the message to the remote OS.
5. The remote OS gives the message to the server stub.
6. The server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



Remote Procedure Calls (2)

A remote procedure call occurs in the following steps:

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



Passing Value Parameters (1)

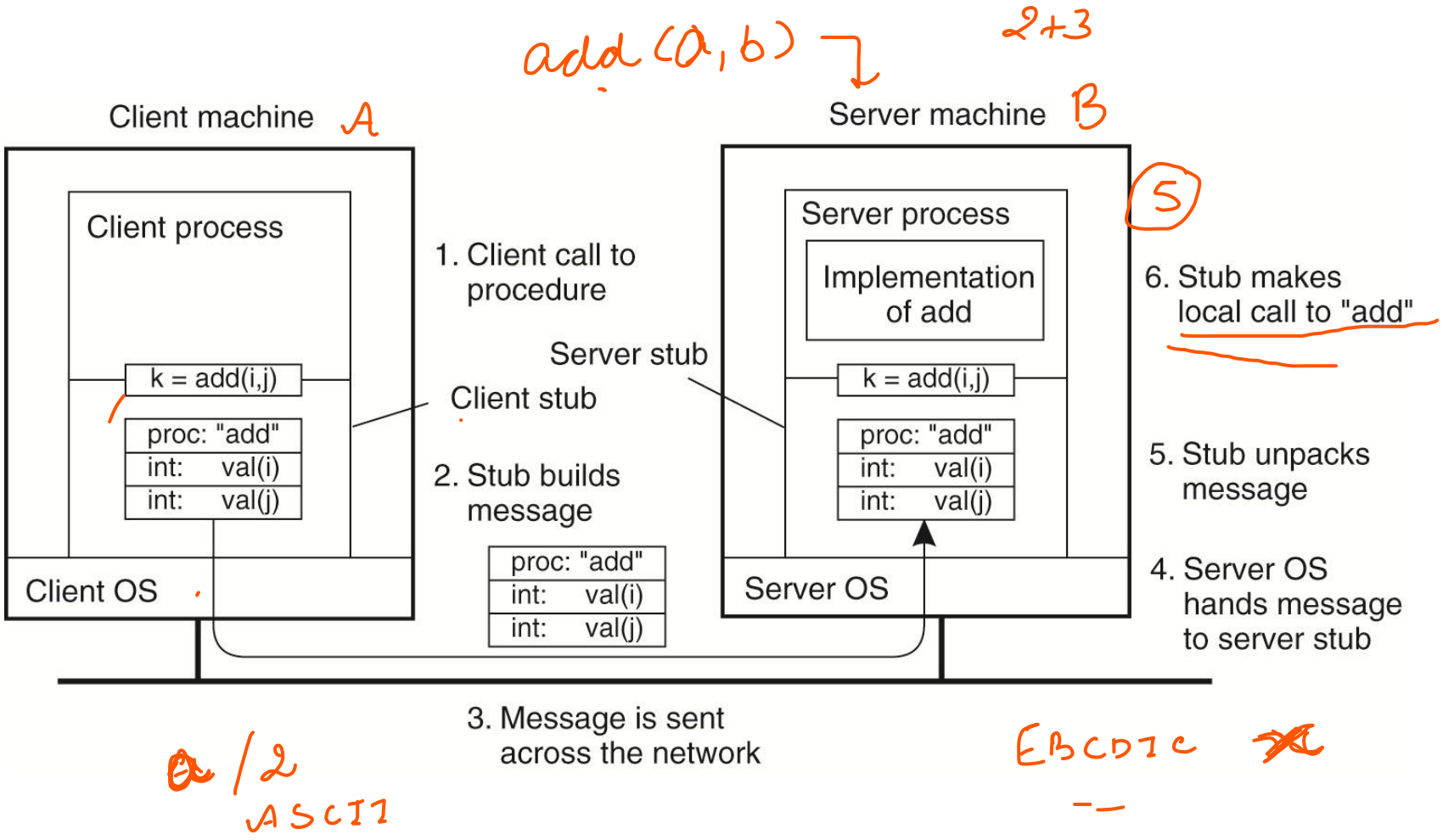


Figure 4-7. The steps involved in a doing a remote computation through RPC.

Passing Value Parameters (2)

What if the client and server machines are of different architectures, say, INTEL and SPARC

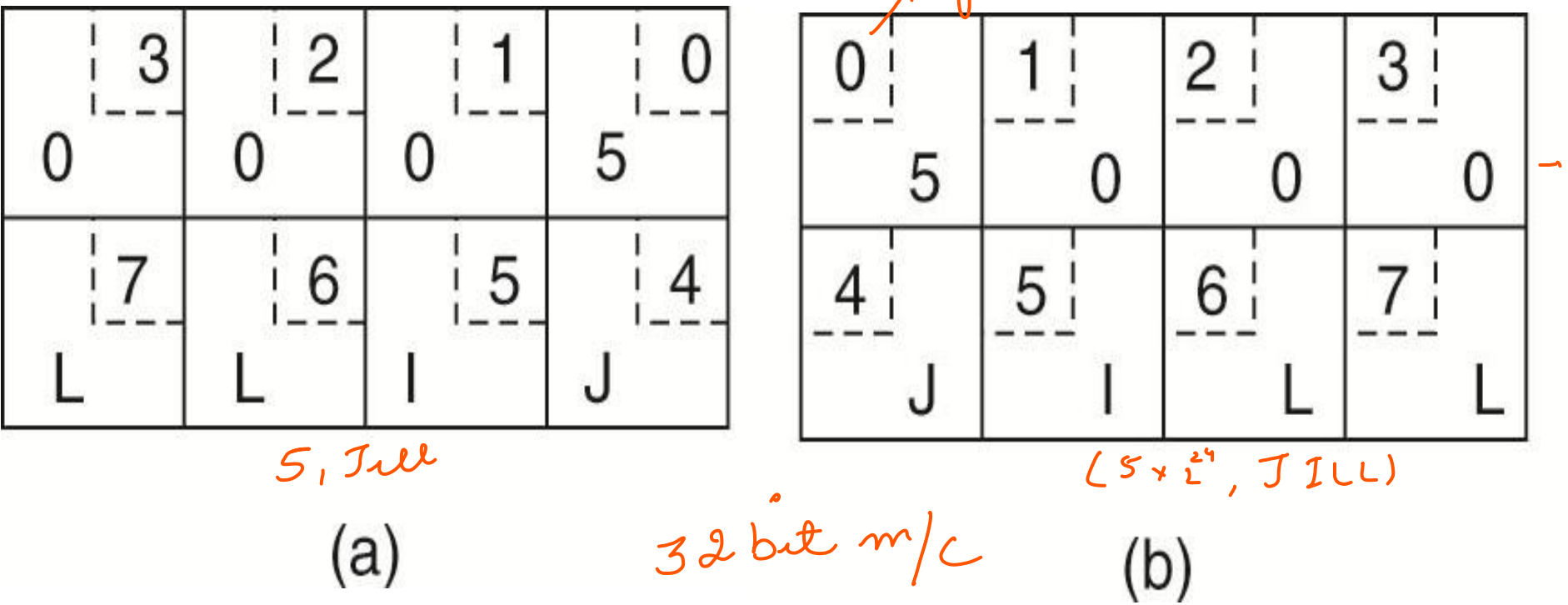
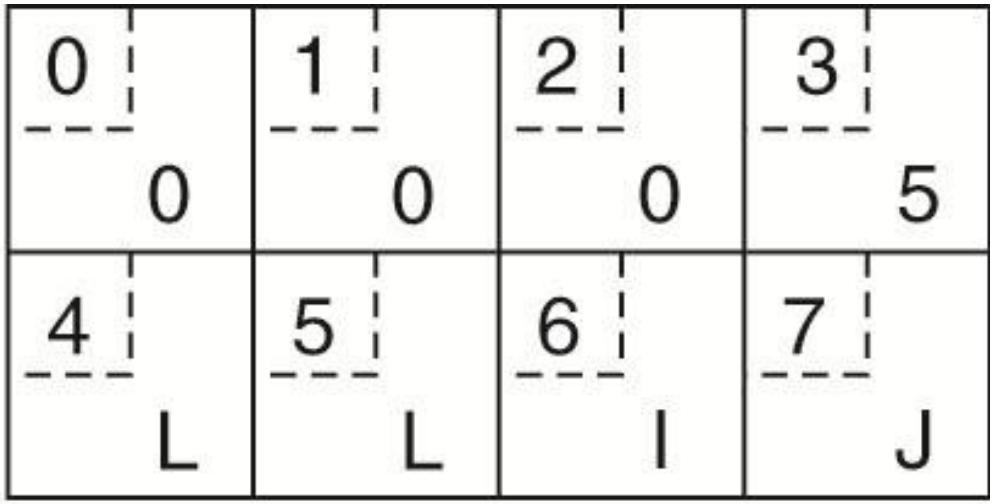


Figure 4-8. (a) The original message on the Pentium. (b) Message after receipt at SPARC

Passing Value Parameters (3)

Try reversing the bytes



(c)

Figure 4-8. (c) The message after being inverted. The little numbers in boxes indicate the address of each byte.

Parameter Specification and Stub Generation

Separate Address spaces?????

call by ref.

```
foobar( char x; float y; int z[5] )
{
   ....
}
```

Array.

N1

Base Add = 1000

1, 2, 3, 4

(a)

N2

→ 1000

offs

N1

N2

Copy by value + restore

foobar's local variables	
	x
y	
5	
z[0]	
z[1]	
z[2]	
z[3]	
z[4]	

(b)

Figure 4-9. (a) A procedure. (b) The corresponding message.