

Causally related events

- In a system with vector clocks:
 - $a \rightarrow b$ iff $t^a < t^b$
- Practical consequence: by comparing vector timestamps we can tell if two events are causally related:
 - $t^a < t^b \Rightarrow a \rightarrow b$

Process Communication Today

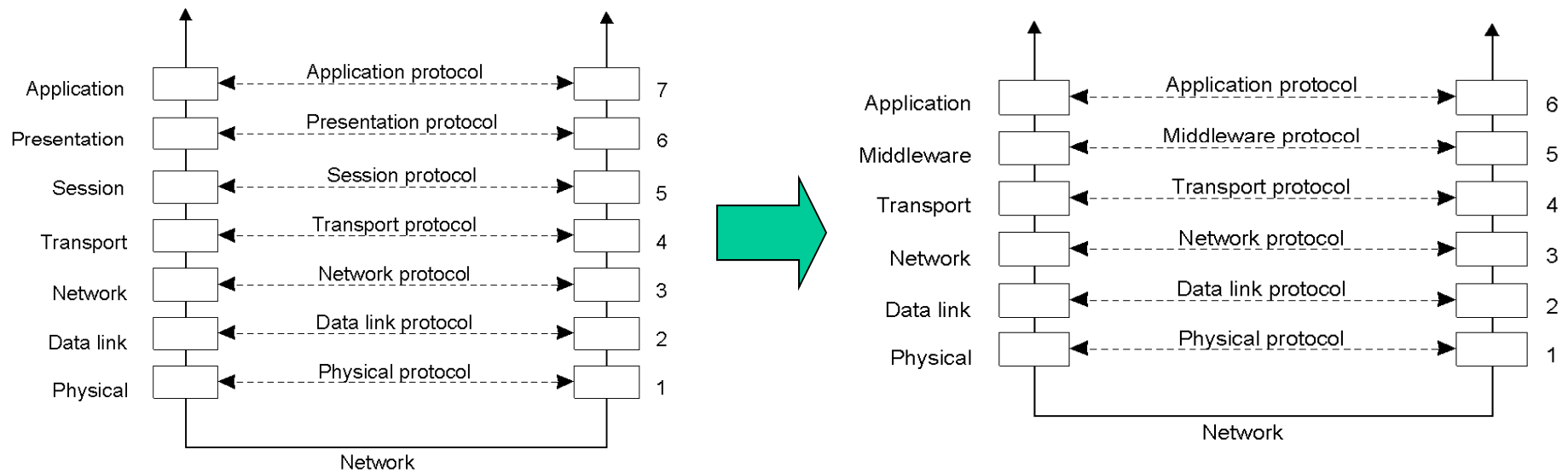
- Process communication today tends to be specialized along application domains
 - Universal communication support: TCP/IP protocol, stack
 - High performance computing: MPI
 - Distributed programming: RPC, RMI, Object brokers, web services
 - Digital Multimedia: streaming protocols
- Conceptually they can be thought of as a specialization of the early process communication concepts

Message Passing model

- Distributed systems -> processes interact via messages
 - Distributed applications on the Web
 - Parallel programs consisting of cooperating processes
- Communication is explicit
 - Programmer uses *send* and *receive* operations, involved in details of data transport and data synchronization
- Messages have double function
 - exchange data
 - synchronize their executions

Evolution of Layered Protocols

- Many new protocols developed as part of new applications – FTP, HTTP, SSH, etc, not originally foreseen by the OSI architects
- New middleware layer replaces obsolete Presentation and Session layers

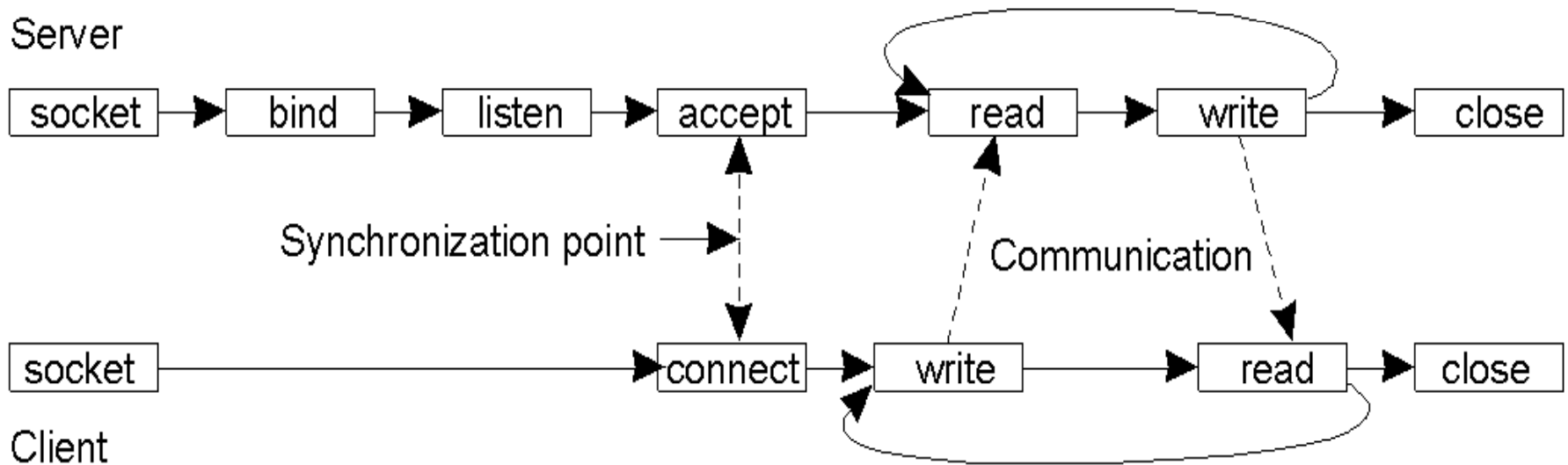


Berkeley Sockets (1)

- Socket primitives for TCP/IP.

Primitive	Meaning
Socket	Create a new communication endpoint
Bind	Attach a local address to a socket
Listen	Announce willingness to accept connections
Accept	Block caller until a connection request arrives
Connect	Actively attempt to establish a connection
Send	Send some data over the connection
Receive	Receive some data over the connection
Close	Release the connection

Berkeley Sockets (2)



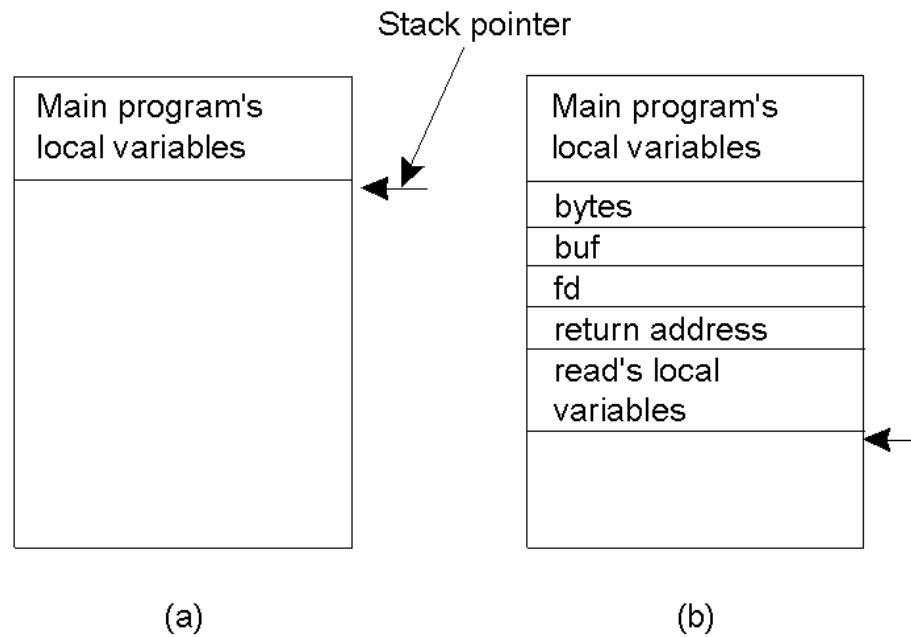
- Connection-oriented communication pattern using sockets.

The Remote Procedure Call model

- Distributed systems -> processes interact via messages
 - Remote services – access to a file
- However the communication is implicit
 - RPC looks and smells like a local procedure call
 - Conceptually simple to implement, the devil is in the details
- A RPC has multiple functions
 - Data transport
 - Synchronization of executions of processes

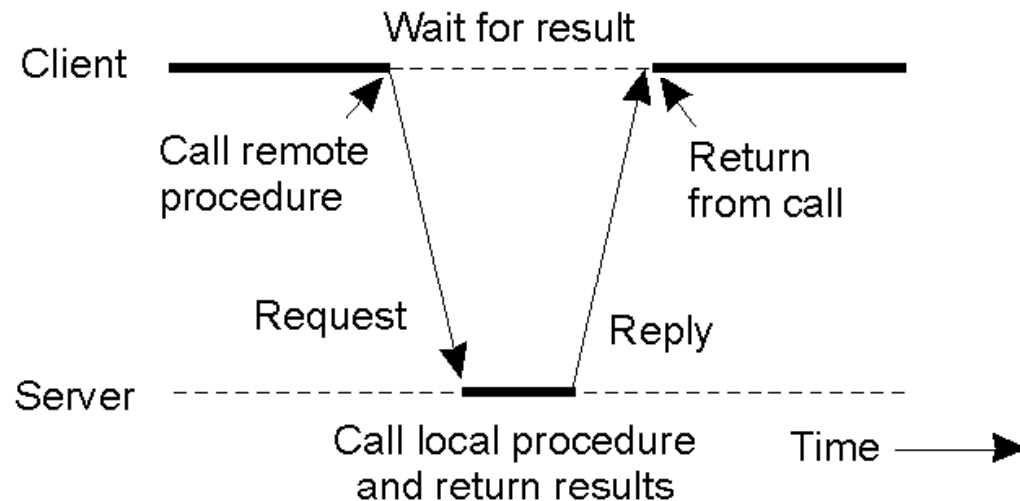
Conventional Procedure Call

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



Client and Server Stubs

- Principle of RPC between a client and server program.
- Client and server stubs (intermediate auxiliary functions) used to build the illusion of a local procedure call
- Main task of stub: packing and unpacking of parameters and results in messages

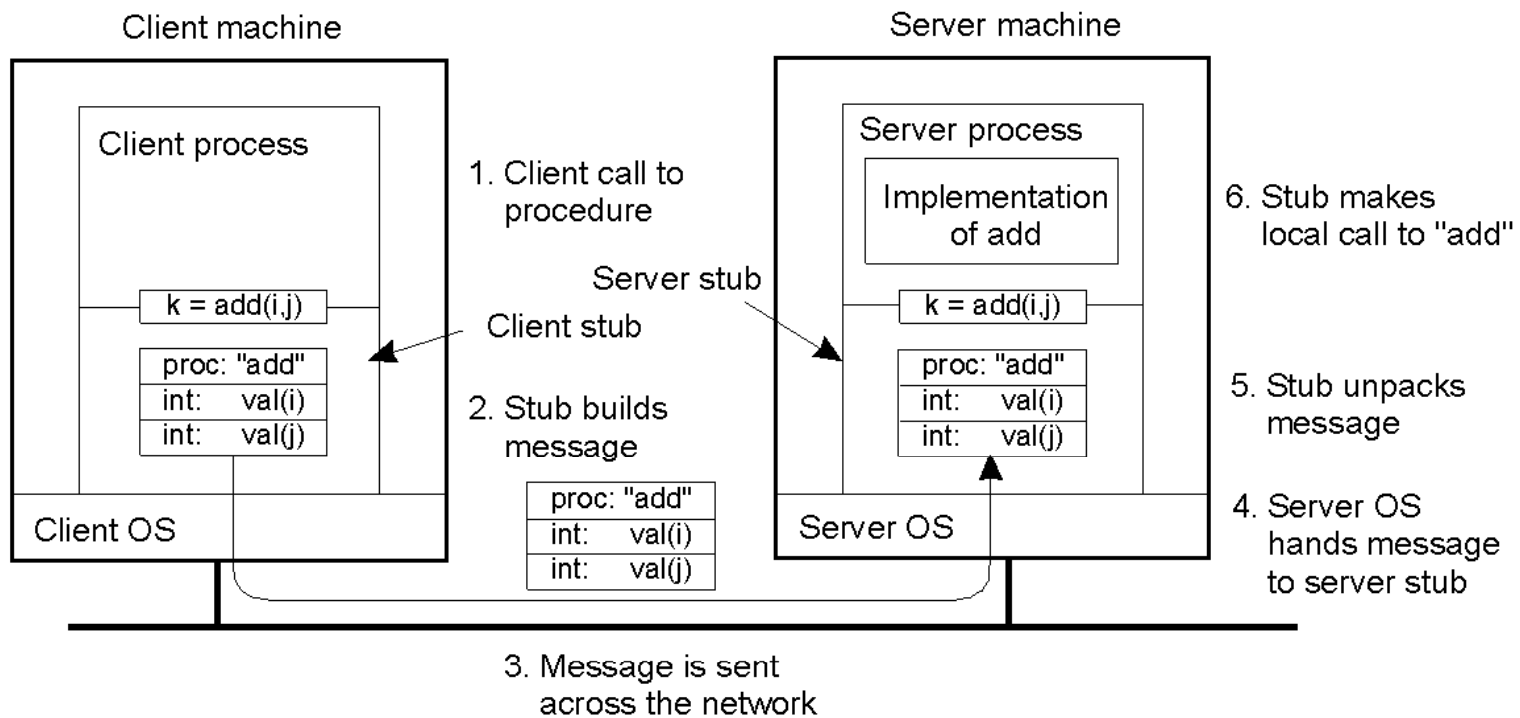


Steps of a Remote Procedure Call

1. Client procedure calls client stub in normal way
2. Client stub builds message, calls local OS
3. Client's OS sends message to remote OS
4. Remote OS gives message to server stub
5. Server stub unpacks parameters, calls server
6. Server does work, returns result to the stub
7. Server stub packs it in message, calls local OS
8. Server's OS sends message to client's OS
9. Client's OS gives message to client stub
10. Stub unpacks result, returns to client

Passing Value Parameters (1)

- Steps involved in doing remote computation through RPC



Passing Value Parameters (2)

A major complication is the different byte ordering adopted in different processor architectures

- Original message on the Pentium
- The message after receipt on the SPARC
- The message after being inverted. The little numbers in boxes indicate the address of each byte

3	2	1	0
0	0	0	5
7	6	5	4
L	L	I	J

(a)

0	1	2	3
5	0	0	0
4	5	6	7
J	I	L	L

(b)

0	1	2	3
0	0	0	5
4	5	6	7
L	L	I	J

(c)

Parameter Specification and Stub Generation

- a) A procedure
- b) The corresponding message.

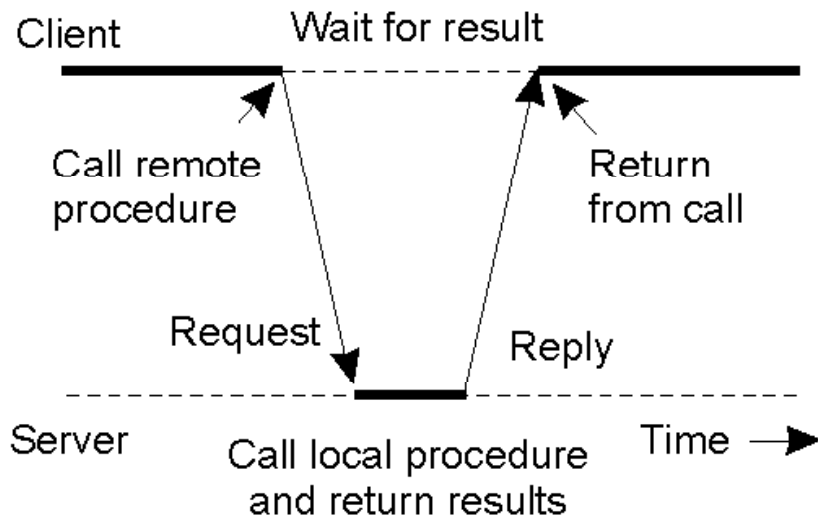
```
foobar( char x; float y; int z[5] )  
{  
  ....  
}
```

(a)

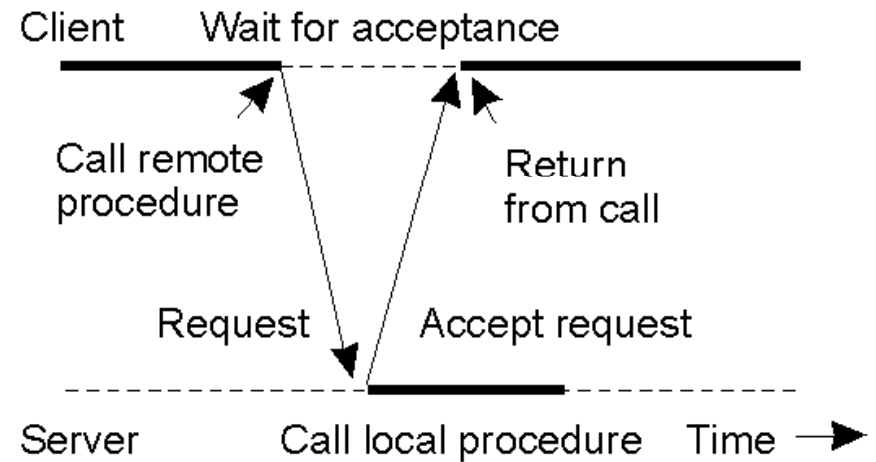
foobar's local variables	
	x
y	
5	
z[0]	
z[1]	
z[2]	
z[3]	
z[4]	

(b)

Asynchronous RPC (1)



(a)



(b)

- a) The interconnection between client and server in a traditional RPC
- b) The interaction using asynchronous RPC

Asynchronous RPC (2)

- A client and server interacting through two asynchronous RPCs

