Formal Notions (1)

Modeling:

Distributed systems are modeled as a set of *N processes p* residing on *M sites* (processors).

Evolution of the system is modeled by a succession of *events* e^i_p, also called a history.

t(e) denotes the real time instant when e happens.

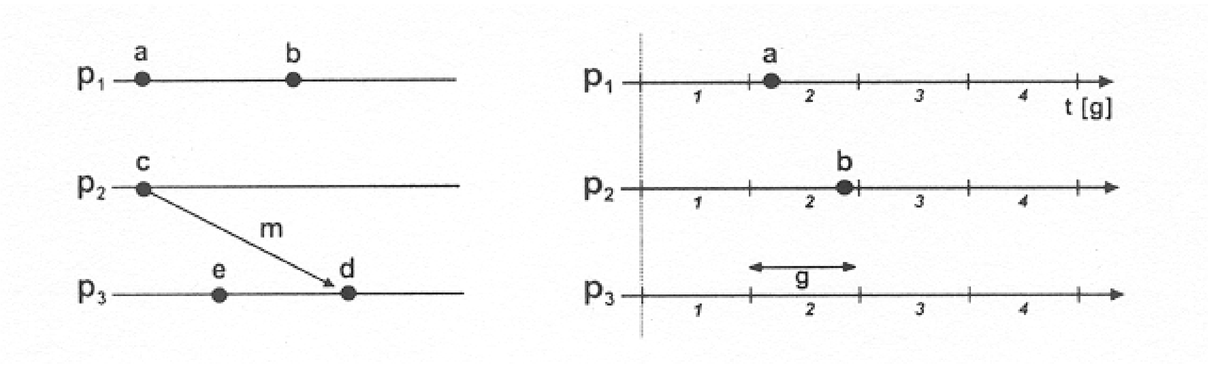
State S_i of a process i is modified by each occurring event in i.

History H of a process is modeled as an ordered set of tuples composed of the momentary state and the event.

Events can be *execution, send, receive events*, resp.

Delivery (in contrast to reception) of a message denotes its transfer to the upper (application) layer

Space-Time and Lattice Diagrams:



Timestamp T(e): = c(t(e))

Formal Notions (2)

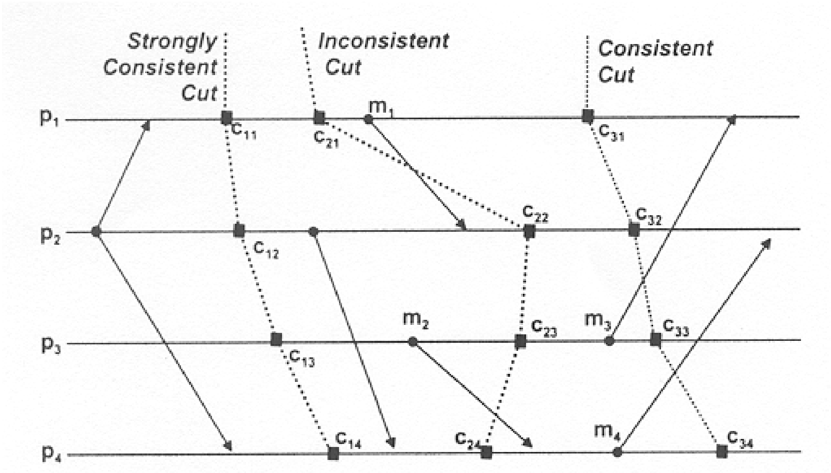
Global State S of a distributed system at a given point t in real time:

S = (S_1, …., S_N), where S_i is the state of process i at time t.

Cut in the space-time diagram:

A vertical line intersecting the (horizontal) timelines of all processes.

Example:



Safety property: Specification that a given predicate *P* is always true.

Liveness property: Specification that a given predicate *P* will eventually be true.

Timeliness property: Specification that a given predicate *P* will be true at a given instant of real time.

Distributed System Paradigms (1)

1. Naming and Addressing

names:

associated with entities, objects, resources, in order to refer to and to communicate with them. The act of associating a name with an object is called *binding.*

pure: the name is just a pattern; no information about the object can be extracted from the name alone.

impure: structure and format of the name yields additional information.

unique: names can be used for clear identification

addresses:

attributes of names that can be used to interact with the entity the name refers to.

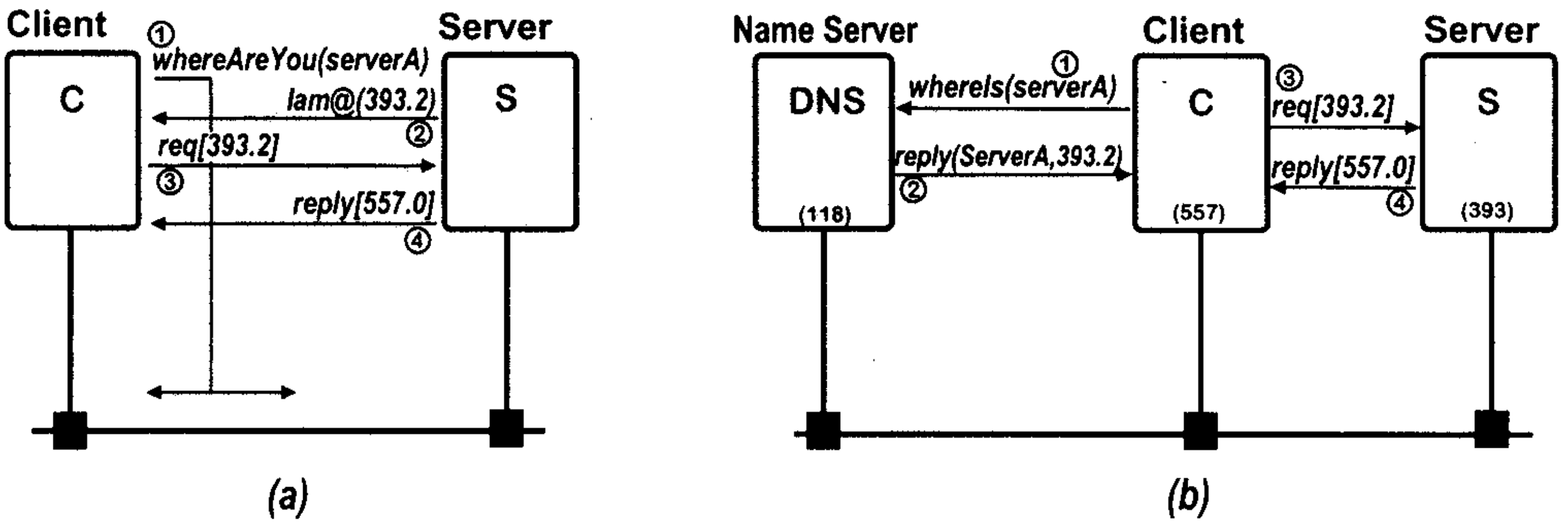
advantages of referring to objects using names instead of addresses:

- easier to remember
- more convenient than using all addresses required by the respective network protocols
- *location transparent*

name resolution: mechanism that dynamically generates an address given the name

Distributed System Paradigms (2)

Examples for name resolution: (a) Broadcast; (b) Name Server



distributed name server approach:

scalable approach to implement a name service by using a set of cooperating name servers

name service agent:

hiding the interaction with the name servers from the application

caching:

making name service efficient (analogy to accessing memory)
– by copying recent name-to-address resolutions both at the name server and the agent

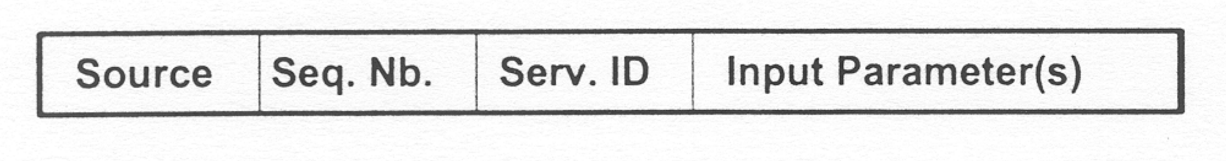
Distributed System Paradigms (3)

2. Message Passing (point-to-point)

In order to exchange messages, the two involved components must

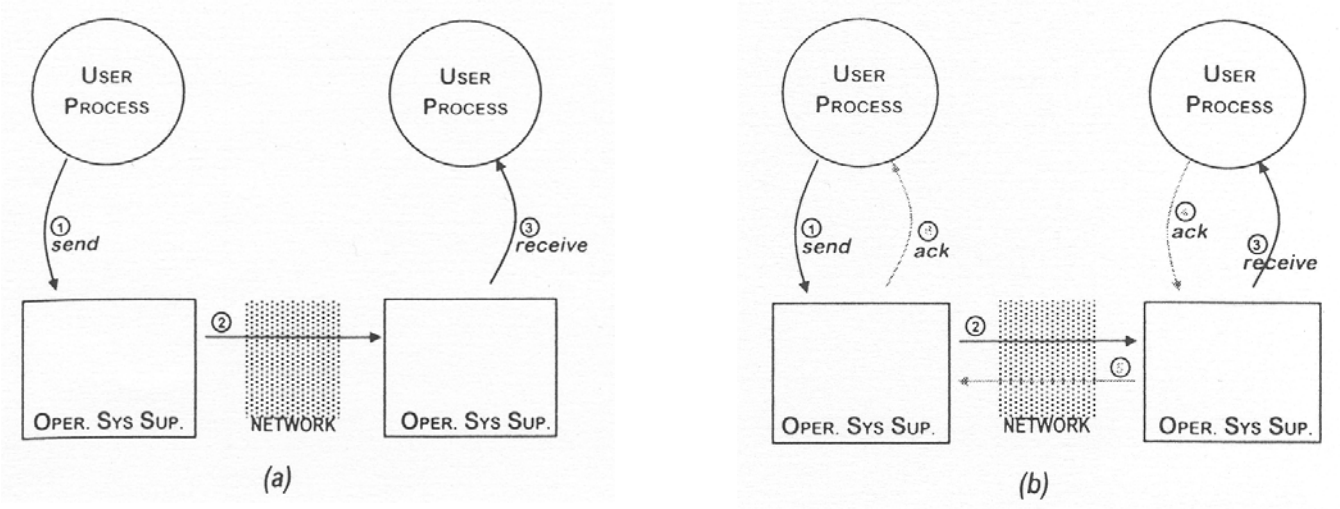
- select a protocol and obtain the address of each other
- agree on the format of the messages exchanged

Example of a message format



Source	Seq. Nb.	Serv. ID	Input Parameter(s)

Example of message passing protocols:(a) Send-Receive (b) Acknowledges-Send



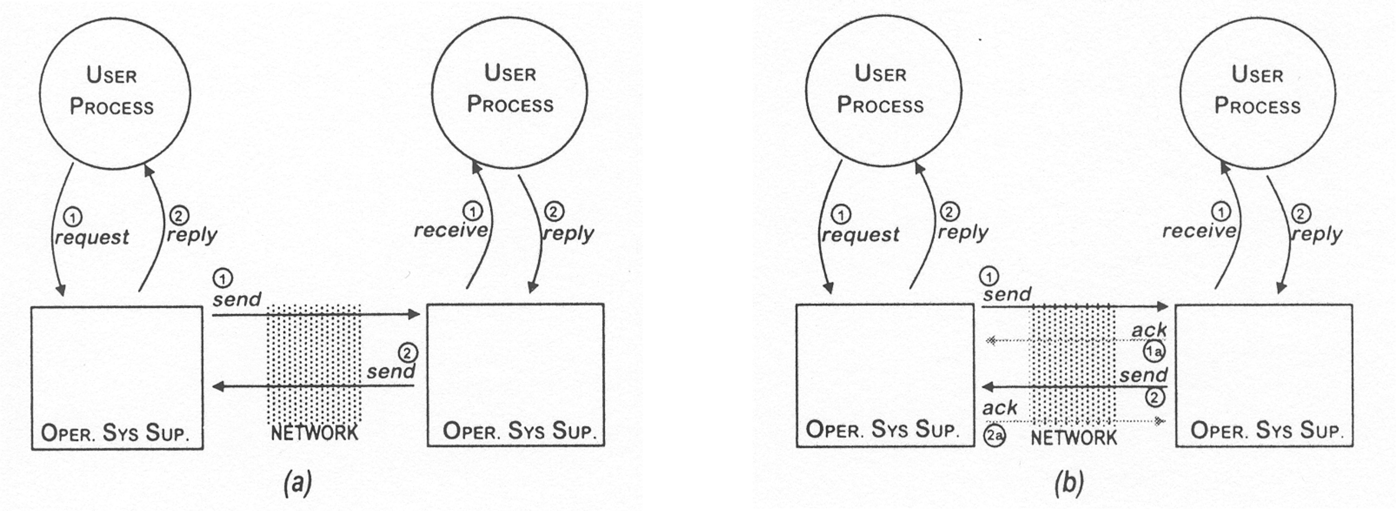
Distributed System Paradigms (4)

Open questions:

- should the provided message primitives be of blocking nature?
- how long should the sender process be blocked?

notification: messages which expect no response, e.g. notifying occurrence of events

Examples of remote operation protocols: (a) Request-Reply (b) Acknowledged


USER PROCESS
USER PROCESS
1 request
2 reply
1 receive
2 reply
1 send
2 send
OPER. SYS SUP.
NETWORK
OPER. SYS SUP.
(a)
USER PROCESS
USER PROCESS
1 request
2 reply
1 receive
2 reply
1 send
ack
1a
send
2
ack
2a
OPER. SYS SUP.
NETWORK
OPER. SYS SUP.
(b)