Distributed System

System that consists of several computers that communicate with each other exchanging messages over a communication network and each computer has its own memory and runs it own OS.

Local resource: the resources owned and controlled by a computer are said to be local to it.

Remote resource: the resources owned and controlled by other computers and those that can only be accessed through the network are said to be remote

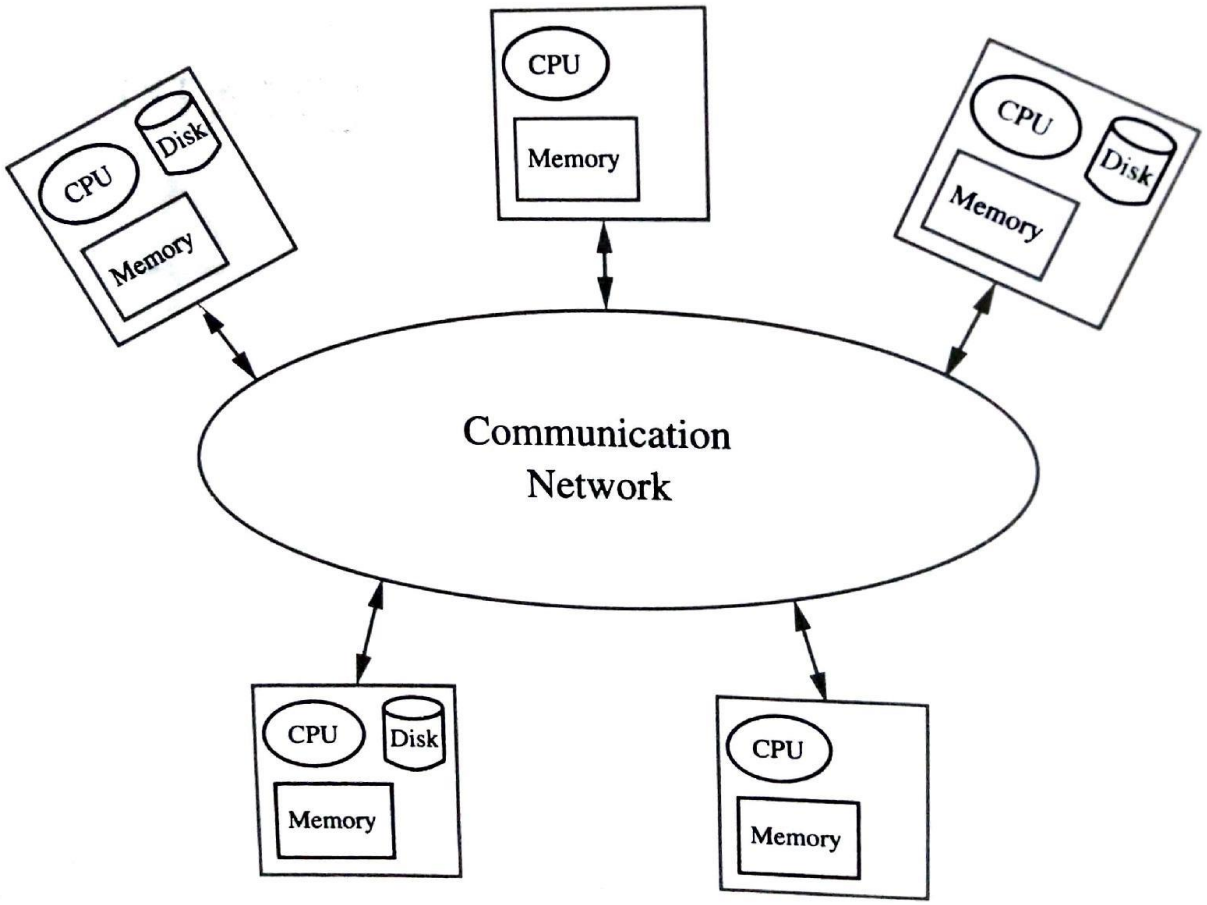


Fig. Architecture of a distributed system

Motivation

- availability of powerful microprocessor as low cost
- significant advances in communication technology

Advantages of Distributed systems over traditional time-sharing systems

- Resource sharing
- Enhanced performance
- Improved reliability and availability
- Modular expandability

System architecture types

Tanenbaum and Renesse classified distributed systems into three broad categories:

- **Minicomputer model**

-Ratio of the no. of processors to the no. of users is normally <1

- **Workstation model**

-Ratio of the no. of processors to the no. of users is normally =1

- **Processor pool model**

-Ratio of the no. of processors to the no. of users is normally >1

Distributed OS(DOS)

A DOS extends the concepts of resource management and user friendly interface for shared memory computers a step further, encompassing a distributed computing system consisting of several autonomous computers connected by a communication network.

Issues in DOS

- Global knowledge
- Naming
- Scalability
- Compatibility
- Process synchronization
- Resource management
  - -Data migration
  - -Computation migration
  - - Distributed scheduling
- Security
- Structuring
  - -The Monolithic Kernel
  - -The Collective Kernel Structure
  - -Object Oriented OS
- Client-Server Computing Model

Communication Networks

- WAN

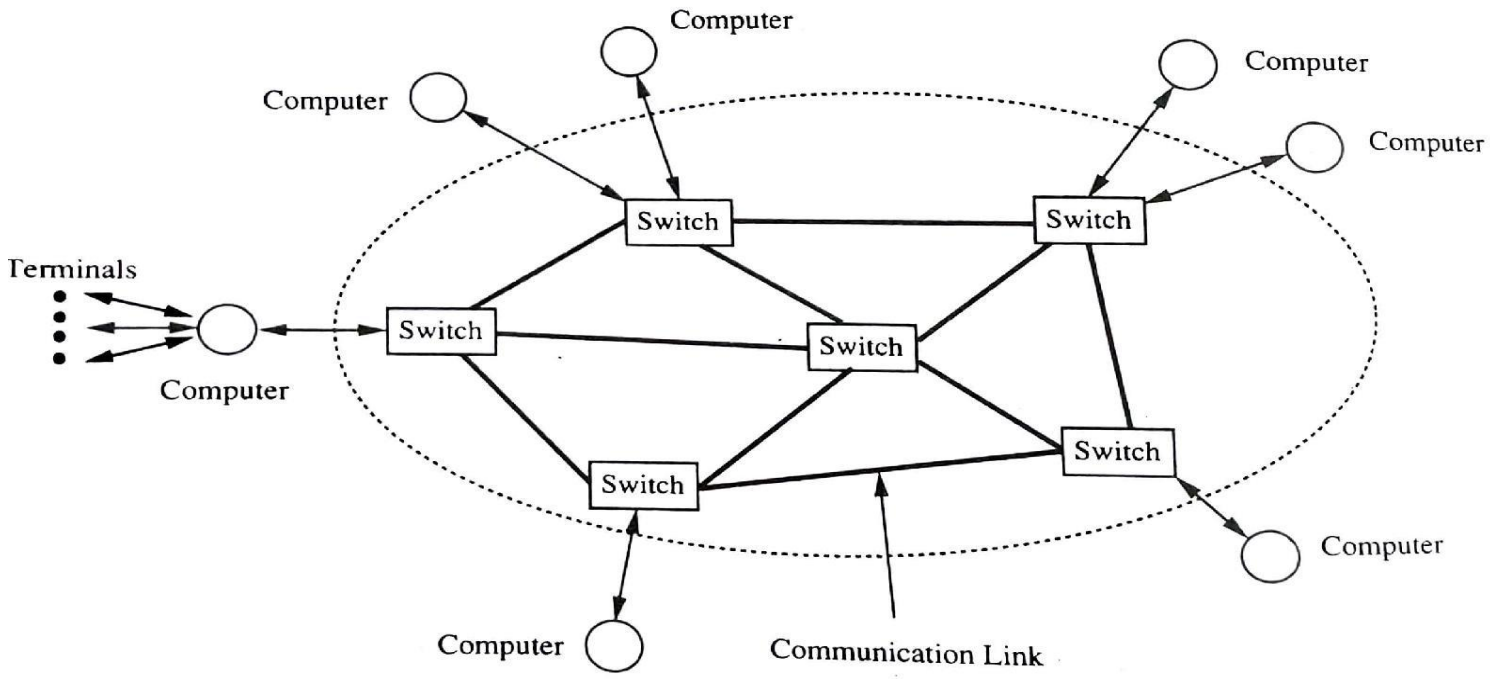
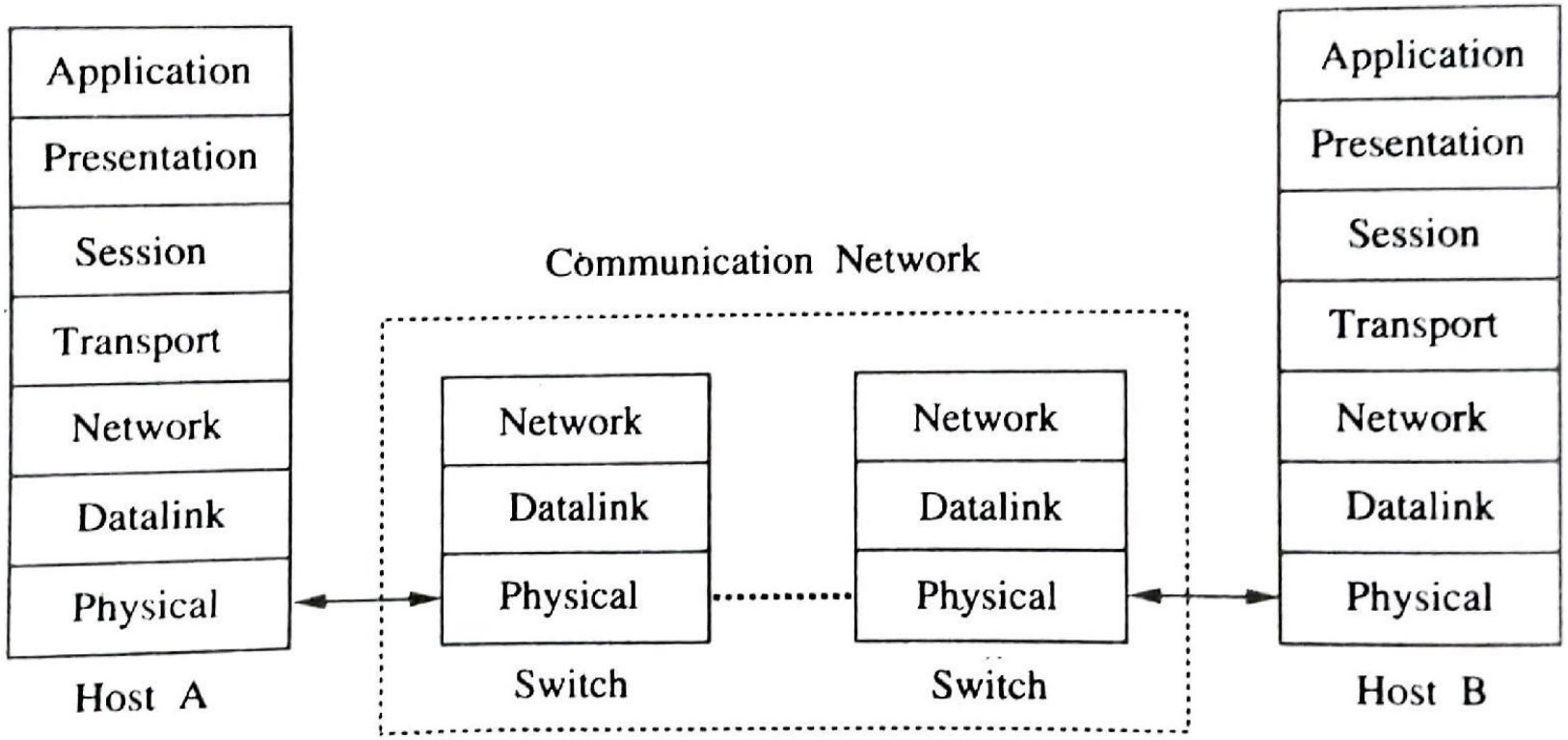


Fig. A point-to-point network

Circuit Switching Vs Packet Switching

Circuit Switching	Packet Switching
Physical path between source and destination	No physical path
All packets use same path	Packets travel independently
Reserve the entire bandwidth in advance	Does not reserve
Bandwidth Wastage	No Bandwidth wastage
No store and forward transmission	Supports store and forward transmission

ISO OSI Reference Model



Overview of the ISO OSI Layers

Physical Layer

- allow a device to send raw bit stream of data over the communication network
- not concerned with transmission errors, how bits are organized or what they mean
- aware and take care of the communication network implementation details like circuit/packet switching, type of network(telephone system, digital transmission, etc.)

Data Link Layer

- responsible for recovering from transmission errors and flow control (takes care of any disparity between the speeds at which the bits can be sent and received)
- makes the communication facility provided by the physical layer reliable

Network Layer

- mainly responsible for routing and congestion control
- breaks a message into packets and decide which outgoing line will carry the packets toward their destination

Transport Layer

- primary function is to hide all the details of the communication network from the layers above
- provides a network independent device-to-device (end-to-end) communication
- can provide the ability to the network to inform a host that the network has crashed or has lost certain packets

This layer can provide improved reliability if necessary.

Session Layer

- responsible for establishing and maintaining a connection known as session between two processes
- may keep track of the outstanding requests and replies from processes and order them in such a manner to simplify the design of user programs

Presentation Layer

- interface between a user program and the rest of the network
- provides data transmission utilities to take care of the differences in representing information at the source and at the destination
- may perform data compression, encryption and conversion to and from network standards for terminals and files

Application Layer

- provide a facility for the user processes to use the ISO OSI protocols

Local Area Networks(LANs)

Some of the key characteristics of LANs:

- high data transmission rates(10 – 100 Mbps)
- small geographic scope, generally confined to a single building or perhaps several buildings(such as a college campus)
- low transmission error rate

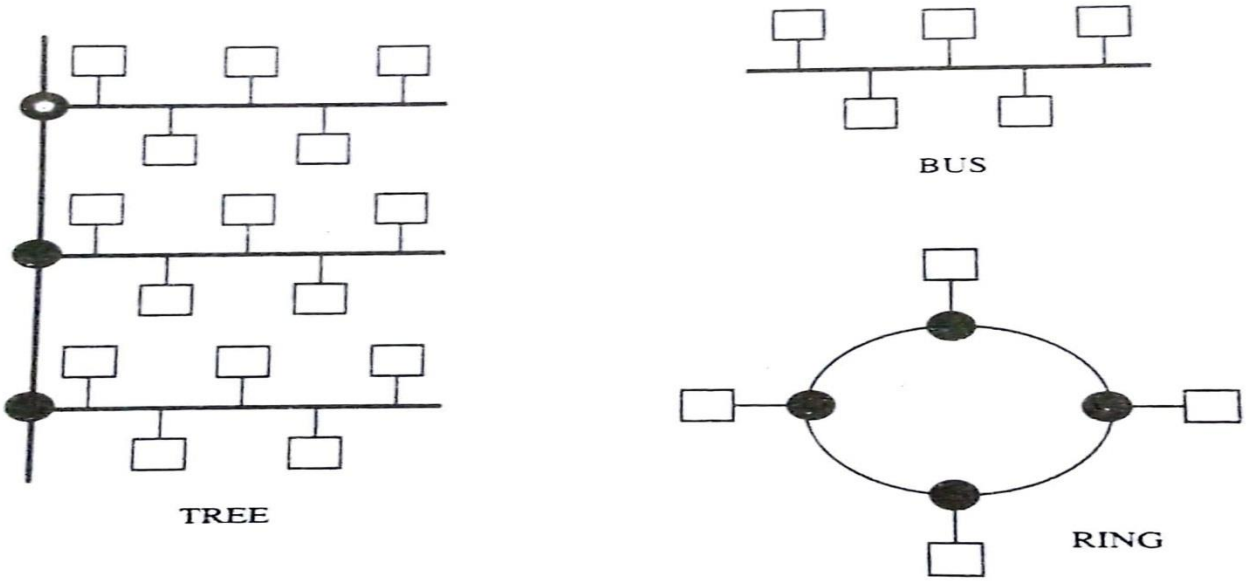


Fig. Network topologies

Widely used network topologies for LANs are bus, ring and tree.

The communication media can be coaxial cable, twisted pair wire or optical fiber.

Bus/Tree Topology

In Bus topology the communication devices transmit data in the form of packets(contains the address of the destination and message).

Tree topology LAN obtained by interconnecting many bus topology LANs to a common bus.

Bus topology LANs can be viewed as branches of a tree.

Two protocols to control access to the bus:

- CSMA/CD(Carrier Sense Multiple Access with Collision Detection)
- Token bus

Ring Topology

Ring is physical, data is transmitted point-to-point. At each point the address on the packet is copied and checked to see if the packet is meant for the device connected at that point. If match the rest of the packet is copied otherwise the entire packet is retransmitted to the next device on the ring.

Access control protocol to control access to ring:

- The token ring protocol
- The slotted ring protocol

The token ring protocol:

- A token is circulated around the ring
- Token is labeled free when no device is transmitting
- When a device wishes to transmit it waits for the token to arrive, labels the token as busy on arrival and retransmits the token
- Immediately following the release of the token the device transmits data
- The transmitting device will mark the token as free when the busy token returns to the device and the device has completed its transmission

Advantage:

It is not sensitive to the load on the network, the entire bandwidth of the medium can be utilized.

Disadvantage:

- Complexity
- Token has to be maintained error-free
- If token is lost care must be taken to generate only one token
- The maintenance of the token may require a separate process to monitor it

The slotted ring protocol

- A number of fixed length slots continuously circulate around the ring. The ring is like a conveyor belt.
- A device wishing to transmit data waits for a slot marked empty to arrive, marks it full and inserts the destination's address and the data into the slot as it goes by.
- The device is not allowed to retransmit again until this slot returns to the device, at which time it is marked as empty by the device.
- After the newly emptied slot continues on, the device is again free to transmit data.
- A few bits are reserved in each slot so that the result of the transmission(accepted, busy or rejected) can be returned to the source.

Advantage:

Simplicity

Disadvantage:

Wasted bandwidth. When the ring is not heavily utilized, many empty slots will be circulating, but a particular device wishing to transmit considerable amounts of data can only transmit once per round-trip ring time.

Communication Primitives

High level constructs with which programs use the underlying communication network. They play a significant role in the effective usage of distributed systems. It influence a programmer's choice of algorithms as well as ultimate performance of the programs.

Two communication models, namely, message passing and remote procedure call, that provide communication primitives.

Message Passing Models:

This model provides two basic communication primitives

- SEND
- RECEIVE

Remote Procedure Calls(RPC):

In message passing model programmers must handle the following details:

- pairing of responses with request messages
- data representation (when computers of different architectures or programs written in different programming languages are communicating)
- knowing the address of the remote machine or the server
- taking care of communication and system failures

Contd.

- RPC mechanisms hide all the details from programmers.
- It based on the observation that procedure call is a well known and well understood mechanism for transfer of control data within a program running on a single computer.
- The RPC mechanism extends this same mechanism to transfer control and data across a communication network.
- It can be viewed as an interaction between a client and a server where the client needing a service invokes a procedure at the server.

Contd.

Basic RPC operation:

On invoking a remote procedure the calling process (client) is suspended and parameters if any are passed to the remote machine (server) where the procedure will execute.

On completion of the procedure execution the results are passed back from the server to the client and the client resumes execution as if it had called a local procedure.

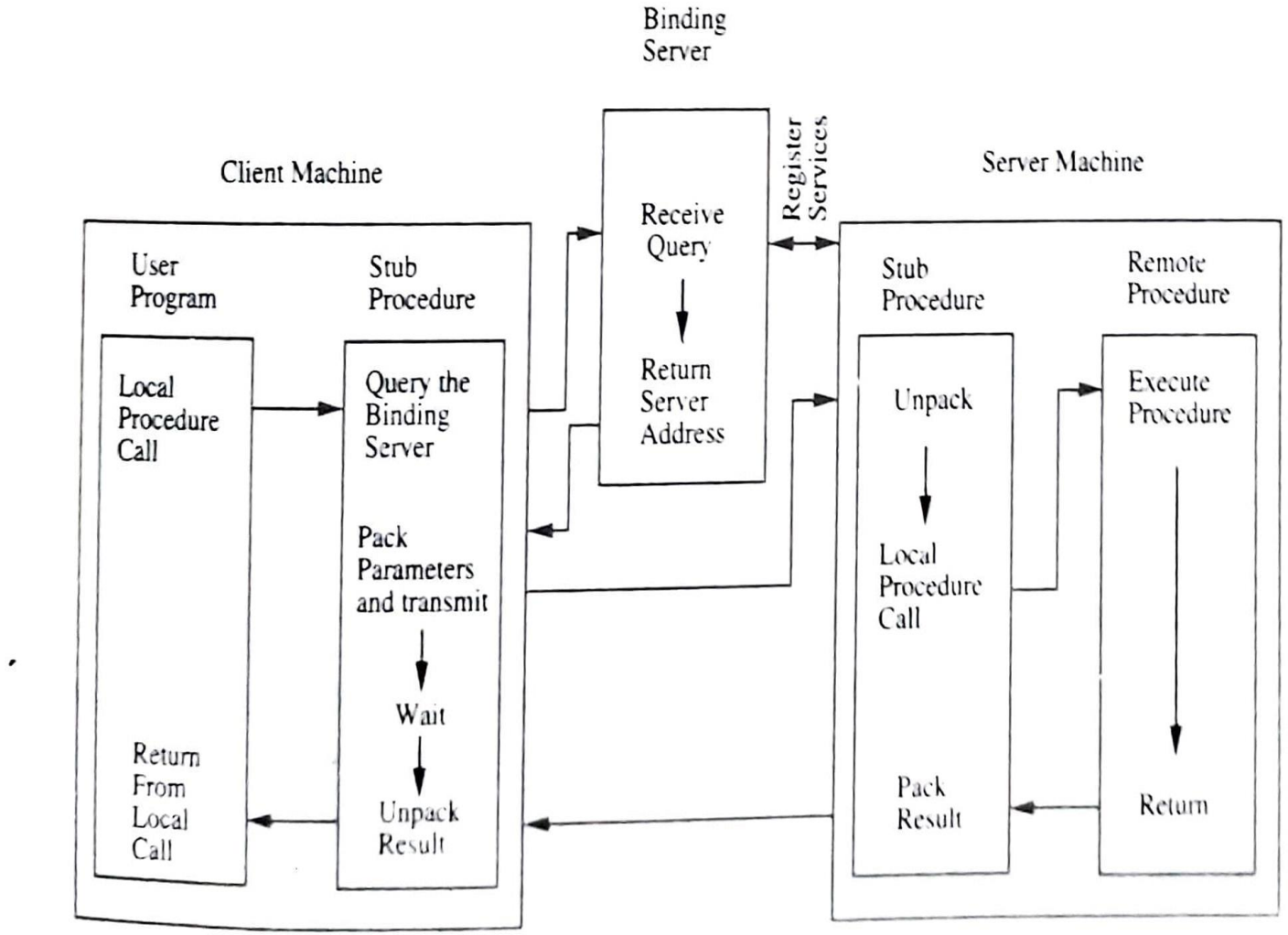


Fig. Remote procedure call

Inherent Limitations of a Distributed System

- Absence of a Global Clock
- Absence of shared memory

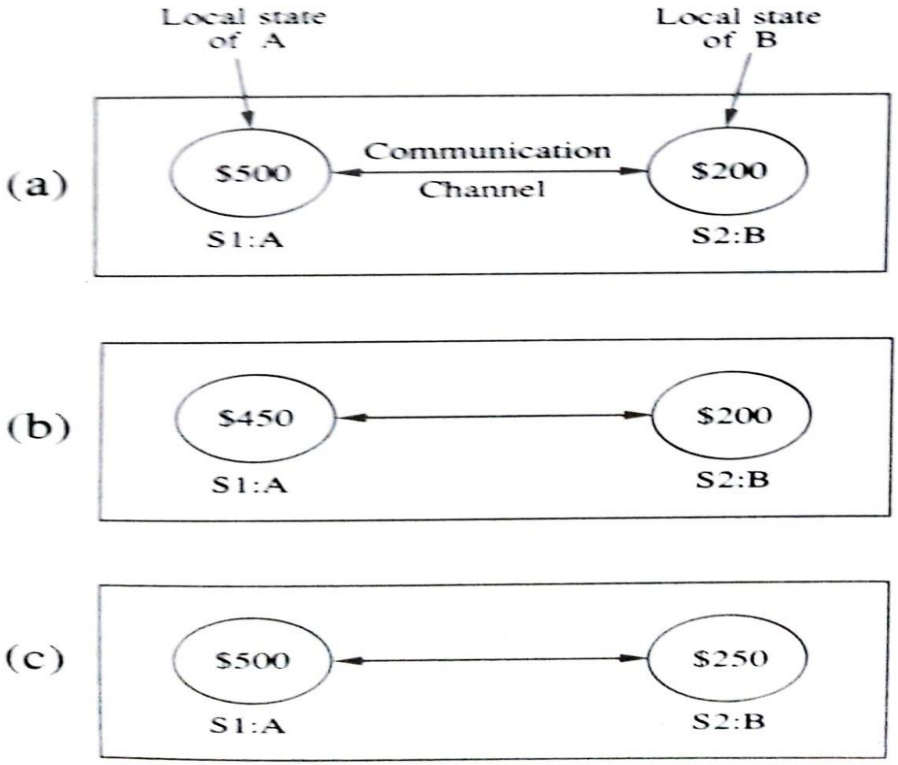


Fig. A distributed system with two sites

Lamport's Logical Clocks

Happened Before Relation(→):

The happened before relation captures the causal dependencies between events i.e. whether two events are causally related or not. The relation →is defined as follows:

- a →b, if a and b are events in the same process and a occurred before b
- a →b, if a is the event of sending a message m in a process and b is the event of receipt of the same message m by another process
- if a→b and b→c, then a→c, i.e. "→"relation is transitive

Causally Related Events

Event a causally affects event b if $a \rightarrow b$

Concurrent events: two distinct events a and b are said to be concurrent (denoted by $a||b$) if $a \not\rightarrow b$ and $b \not\rightarrow a$. In other words concurrent events do not causally affect each other.

For any two events a and b in a system either $a \rightarrow b$, $b \rightarrow a$ or $a||b$.

Contd.

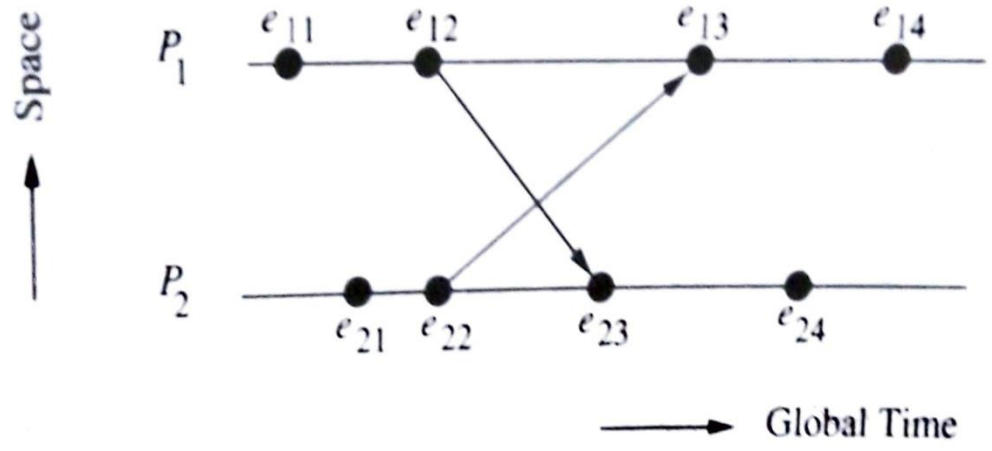


Fig. A space-time diagram

Let e_{11},e_{12},e_{13}, and e_{14} are events in process P_1 and e_{21},e_{22},e_{23} and e_{24} are events in process P_2. The arrows represent message transfers between the processes. For ex. arrow $e_{12}e_{23}$ corresponds to a message sent from process P_1 to process P_2. e_{12} is the event of sending the message at P_1 and e_{23} is the event of receiving the same message at P_2. In the above figure $e_{22} \rightarrow e_{13}$, $e_{13} \rightarrow e_{14}$ and therefore $e_{22} \rightarrow e_{14}$. In other words, event e_{22} causally affects event e_{14}.

Logical clocks

In order to realize the relation → Lamport introduced system of logical clocks. There is a clock C_i at each process P_i in the system. The clock C_i can be thought of as a function that assigns a number $C_i(a)$ to any event a, called the timestamp of event a, at P_i. The numbers assigned by the system of clocks have no relation to physical time and hence the name logical clocks. The timestamp of an event is the value of the clock when it occurs.

Contd.

Conditions satisfied by the system of clocks:

For any events a and b:

If a→b, then C(a)<C(b)

The happened before relation '→' can now be realized by using the logical clocks if the following two conditions(**correctness conditions**) are met:

[C1] for any two events a and b in a process P_i, if a occurs before b, then

$$C_i(a) < C_i(b)$$

[C2] if a is the event of sending a message m in process P_i and b is the event of receiving the same message m at process P_j, then

$$C_i(a) < C_j(b)$$

Contd.

The following **implementation rules (IR)** for the clocks guarantee that the clocks satisfy the correctness conditions [C1] and [C2]:

[IR1] Clock C_i is incremented between any two successive events in process P_i:

$$C_i := C_i + d \quad (d>0)$$

If a and b are two successive events in P_i and $a \rightarrow b$, then $C_i(b) = C_i(a) + d \quad (d>0)$ [1.1]

Contd.

[IR2] If event a is the sending of message m by process P_i, then message m is assigned a timestamp $t_m = C_i(a)$ (note that the value of $C_i(a)$ is obtained after applying IR1). On receiving the same message m by process P_j, C_j is set to a value greater than or equal to its present value and greater than t_m.

$$C_j : \max((C_j, t_m + d) \quad (d>0) \qquad [1.2]$$

Contd.

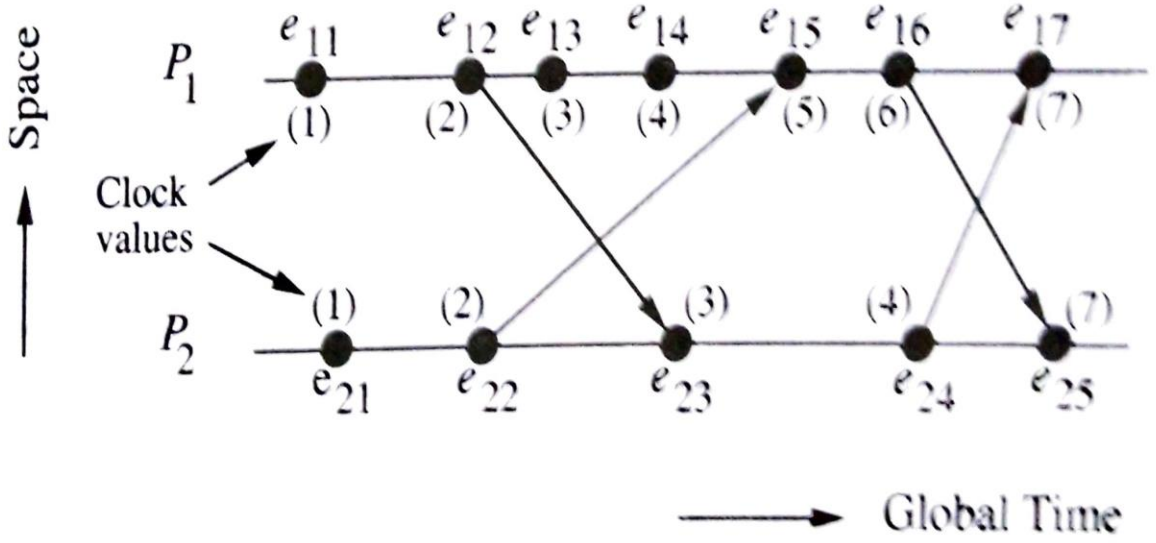


Fig. How Lamport's logical clocks advance

Let both the clock values C_{P1} and C_{P2} are assumed to be zero initially and d is assumed to be 1. e_{11} is an internal event in process P1 which causes C_{P1} to be incremented to 1 due to IR1. Similarly e_{21} and e_{22} are two events in P_2 resulting in C_{P2}=2 due to IR1. e_{16} is a message send event in P_1 which increments C_{P1} to 6 due to IR1. The message is assigned a timestamp =6 . The event e_{25} corresponding to the receive event of the above message, increment the clock C_{P2} to 7(max(4+1,6+1)) due to rules IR1 and IR2. Similarly, e_{24} is a send event in P_2. The message is assigned a timestamp=4. The event e_{17} corresponding to the receive event of the above message increments the clock C_{P1} to 7 (max(6+1, 4+1)) due to rules IR1 and IR2.

Contd.

Virtual time:

Lamport's system of logical clocks implements an approximation to global/physical time, which is referred to as virtual time. Virtual time advances along with the progression of events and is therefore discrete. If no events occur in the system, virtual time stops, unlike physical time which continuously progresses.

Contd.

Limitation of Lamport's clocks

In lamport's system of logical clocks, if a→b then C(a)<C(b). However, the reverse is not necessarily true if the events have occurred in different processes. That is if a and b are events in different processes and C(a)<C(b), then a→b is not necessarily true, events a and b may be causally related or may not be causally related. Thus Lamport's system of clocks is not powerful enough to capture such situations.

Contd.

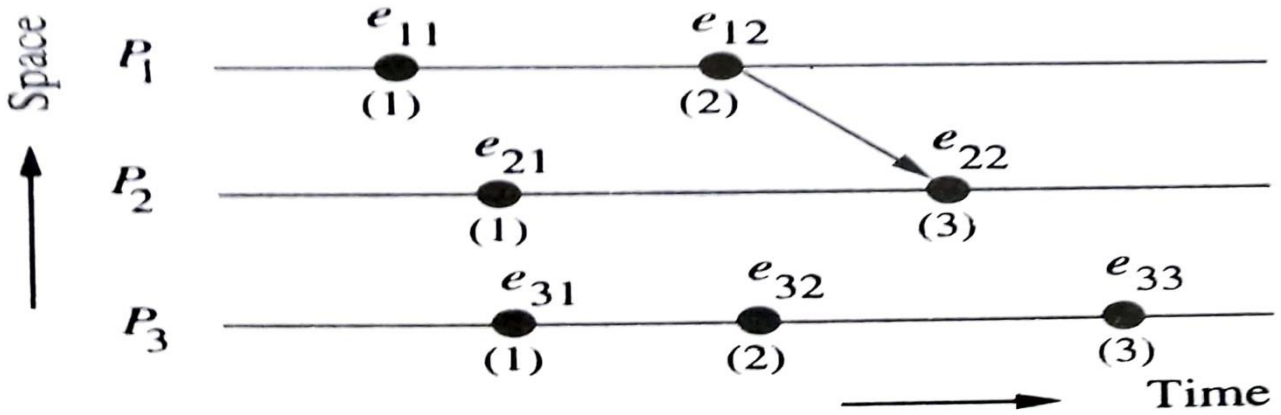


Fig. Space time diagram

From the above figure $C(e_{11}) < C(e_{22})$ and $C(e_{11}) < C(e_{32})$. Event e_{11} is causally related to event e_{22} but not to event e_{32}, since a path exists from e_{11} to e_{22} but not from e_{11} to e_{32}. Note that the initial clock values are assumed to be zero and d is assumed to equal 1. In other words in Lamport's system of clocks we can guarantee that if C(a) < C(b) then b $\not\rightarrow$ a (i.e. the future can not influence the past) however we can not say whether events a and b are causally related or not by just looking at the timestamps of the event.

Contd.

The reason for limitation is that each clock can independently advance due to the occurrence of local events in a process and the Lamport's clock system can not distinguish between the advancements of clocks due to local events from those due to the exchange of messages between processes.

Therefore using Lamport's clocks, we can not reason about the causal relationship between two events occurring in different processes by just looking at the timestamps of the events.

Global State

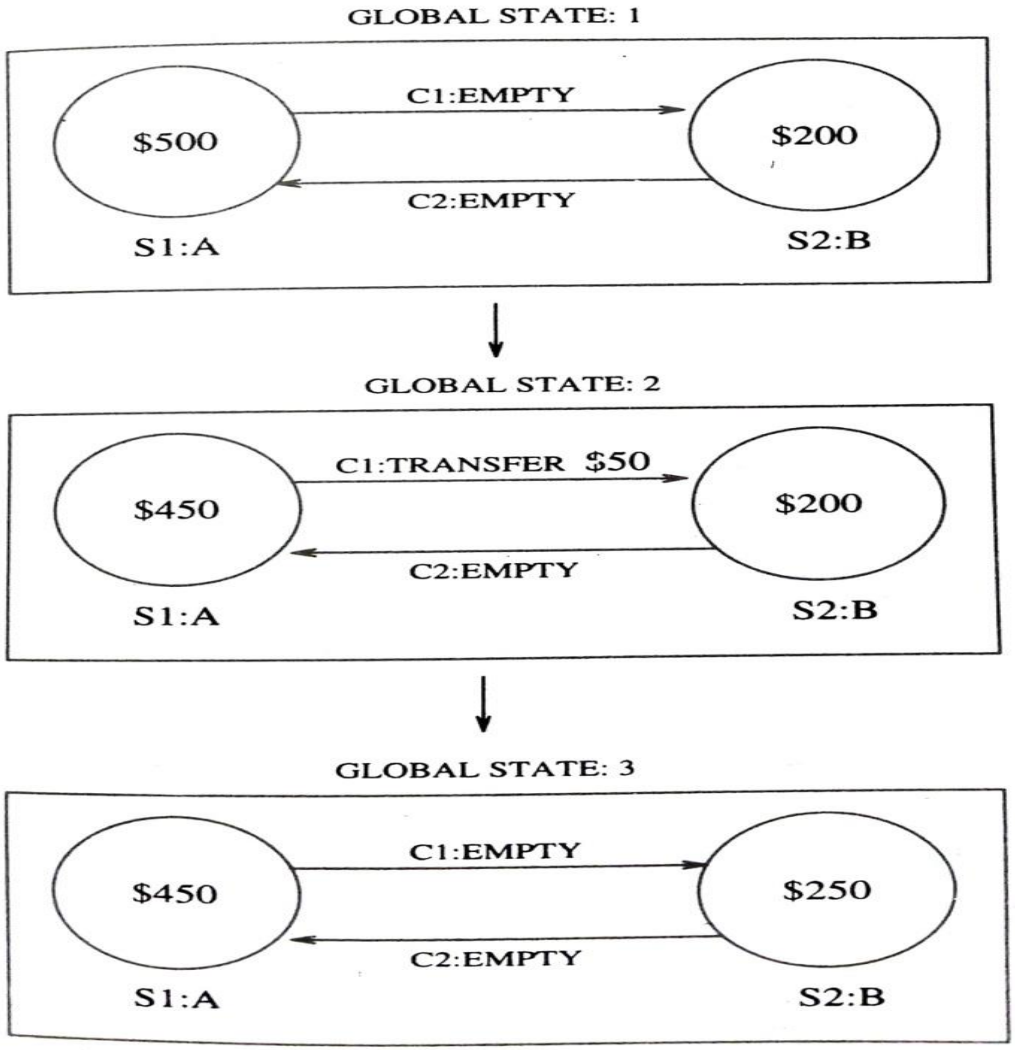


Fig. Global-states and their transitions in the bank a/c example

Contd.

The previous fig. shows the stages of a computation when $50 is transferred from a/c A to a/c B. The communication channels C1 & C2 are assumed to be FIFO.

Suppose the state of a/c A at site S1 was recorded when the global state was 1. Now assume that the global state changes to 2, and the state of communication channels C1 & C2 and of a/c B are recorded when the global state is 2. Then the composite of all the states recorded would show a/c A's balance as $500, a/c B's balance as $200, and a message in transit to transfer $50. In other words an extra amount of $50 would appear in the global state. The reason for this inconsistency is that A's state was recorded before the message was sent and the channel C1's state was recorded after the message was sent.

Contd.

Therefore a recorded global state may be inconsistent if n<n’ where n is the no. of messages sent by A along the channel before A’s state was recorded and n’ is the no. messages sent by A along the channel before the channel’s state was recorded.

Suppose channels state were recorded when the global state was 1 and A and B’s state were recorded when the global state was 2. Then composite of A, B and the channels state will show a deficit of $50. This means that the recorded global state may be inconsistent if n>n’.

Contd.

Hence a consistent global state requires

$$n=n' \quad (1.1)$$

One can show that a consistent global state requires

$$m=m' \quad (1.2)$$

where m'=no. of messages received along the channel before a/c B's state was recorded and m=no. of messages received along the channel by B before the channel's state was recorded.

Since in no system the no. of messages sent along the channel can be less than the no. of messages received along that channel, we have

$$n'>=m \quad (1.3)$$

From eq^n (1.1) and (1.3) we can get

$$n>=m \quad (1.4)$$

Therefore a consistent global state must satisfy eq^n (1.4)

Contd.

The state of a communication channel in a consistent global state should be the sequence of messages sent along that channel before the sender's state was recorded, excluding the sequence of messages received along that channel before the receiver's state was recorded.

Contd.

Some definitions

Local State: Let LS_i denote the local state of S_i at any time. Let $send(m_{ij})$ denote the send event of a message m_{ij} by S_i to S_j and $rec(m_{ij})$ denote the receive event of message m_{ij} by site S_j. Let $time(x)$ denote the time at which state x was recorded and $time(send(m))$ denote the time at which event $send(m)$ occurred.

For a message m_{ij} send by S_i to S_j, we say that

- $send(m_{ij}) \in LS_i$ iff $time(send(m_{ij})) < time(LS_i)$
- $rec(m_{ij}) \in LS_j$ iff $time(rec(m_{ij})) < time(LS_j)$

Contd.

Transit: $transit(LS_i, LS_j)=\{m_{ij} \mid send(m_{ij}) \in LS_i \wedge rec(m_{ij}) \notin LS_j\}$

Inconsistent: $inconsistent(LS_i,LS_j)=\{ m_{ij} \mid send(m_{ij}) \notin LS_i \wedge rec(m_{ij}) \in LS_j\}$

Global State: A global state ,GS, of a system is a collection of the local states of its sites, i.e. GS=$\{LS_1,LS_2,\ldots\ldots,LS_n\}$ where n is the number of sites in the system.

Consistent Global State: A global state GS=$\{LS_1,LS_2,\ldots\ldots,LS_n\}$

is *consistent* iff

for all i and j : 1<=i, j <= n :: $inconsistent(LS_i,LS_j) = \phi$

Thus in consistent global state for every received message a corresponding send event is recorded in the global state. In an inconsistent global state there is at least one message whose receive event is recorded but its send event is not recorded in the global state.

Contd.

Transitless global state:

A global state is transitless iff

for all i, j : 1< = i, j< = n :: $transit(LS_i, LS_j)=\phi$

Thus all communication channels are empty in a transitless global state.

Strongly consistent global state:

A global state is strongly consistent global state if it is consistent and transitless. In a strongly consistent state, not only the send events of all the recorded received events are recorded, but the receive events of all the recorded send events are also recorded. Thus, a strongly consistent state corresponds to a consistent global state in which all channels are empty.

Chandy-Lamport's Global State Recording Algorithm

Chandy and Lamport were the first to propose a distributed algo to capture a consistent global state. The algo uses a marker (a special message) to initiate the algo and the marker has no effect on the underlying computation. The communication channels are assumed to be FIFO. The recorded global state is also referred to as a snapshot of the system state.

Contd.

Marker Sending Rule for a process P

- P records its state
- For each outgoing channel C from P on which a marker has not been already sent, P sends a marker along C before P sends further messages along C.

Contd.

Marker Receiving Rule for a Process Q. On receipt of a marker along a channel C:

If Q has not recorded its state

then

begin

record the state of C as an empty sequence.

follow the “Marker Sending Rule.”

end

else

record the state of C as the sequence of messages received along C after Q’s state was recorded and before Q received the marker along C.