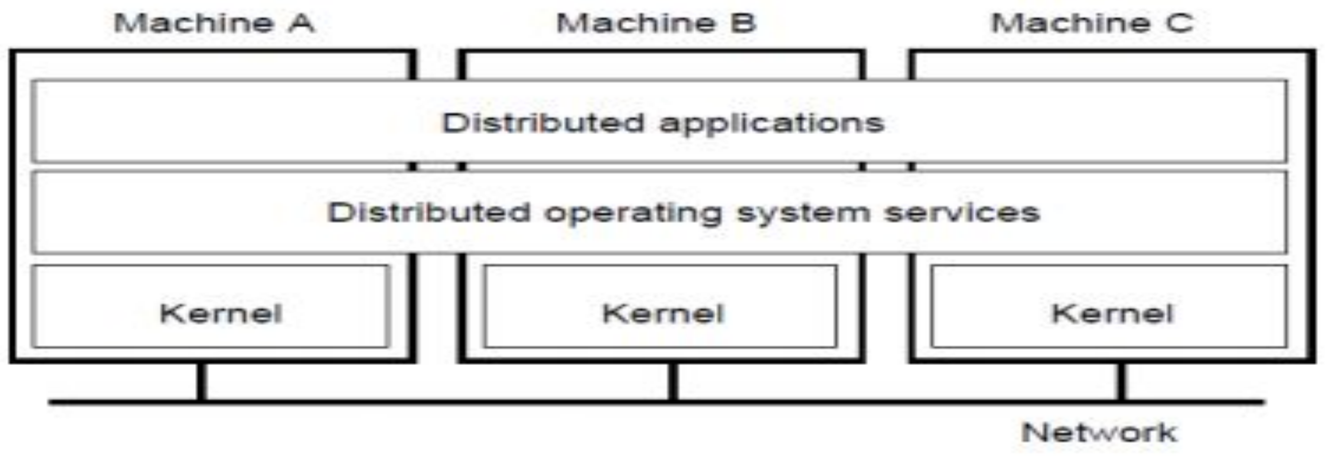
Operating System
Unit-9
Distributed operating System

Introduction

- Distributed system is a collection of autonomous computer system capable of communicating each other.
- A distributed system is a collection of loosely coupled processors interconnected by a communication network.
- Processors are also called nodes, computer, machine, hosts etc.
- Generally, one host at one site, the server, has a resource that another host at another site, the client (or user), would like to use
- In this system the processors do not share memory and clock but the processor has its own local memory.

- A DOS is a system which contains multiple components located on different machines, which coordinate and communicate actions in order to appear as a single coherent working system to the user.
- Distributed operating system is model where distributed application are running on multiple computer linked by communication.
- Example of distributed operating systems are: Windows Server 2003, Windows Server 2008, ubntu, Linux(Apache Server)etc.



Reasons for distributed systems

- **Resource sharing** - user at one site may be able to use the resources available at another
- **Computation speedup** – Distributed system allow us to distribute the computation among the various sites to run the computation concurrently.
- **Reliability** – detect and recover from site failure, function transfer, reintegrate failed site
- **Communication** – message passing i.e, processes at different sites can communicate each other to exchange information. User may initiate file transfer or communicate with another via electronic mail

Advantages of Distributed Systems over Centralized Systems:

Economics:

- Microprocessors offer a better price/performance than mainframes

Speed:

- A distributed system may have more total computing power than a mainframe

Inherent distribution:

- Some applications involve spatially separated machines

Reliability:

- If one machine crashes, the system as a whole can still survive

Incremental growth:

- Computing power can be added in small increments

Advantages of Distributed Systems over independent Computers:

Data sharing:

- Allow many users access to a common data base

Device sharing

- Allow many users to share expensive peripherals like color printers

Communication

- Make human to human communication easier, for example, by electronic mail

Flexibility

- Spread the workload over the available machines in the most cost effective way

Disadvantages of a Distributed Operating System

- Security problem due to sharing
- Some messages can be lost in the network system
- Bandwidth is another problem if there is large data then all network wires to be replaced which tends to become expensive
- Overloading is another problem in distributed operating systems
- If there is a database connected on local system and many users accessing that database through remote or distributed way then performance become slow
- The databases in network operating is difficult to administrate then single user system

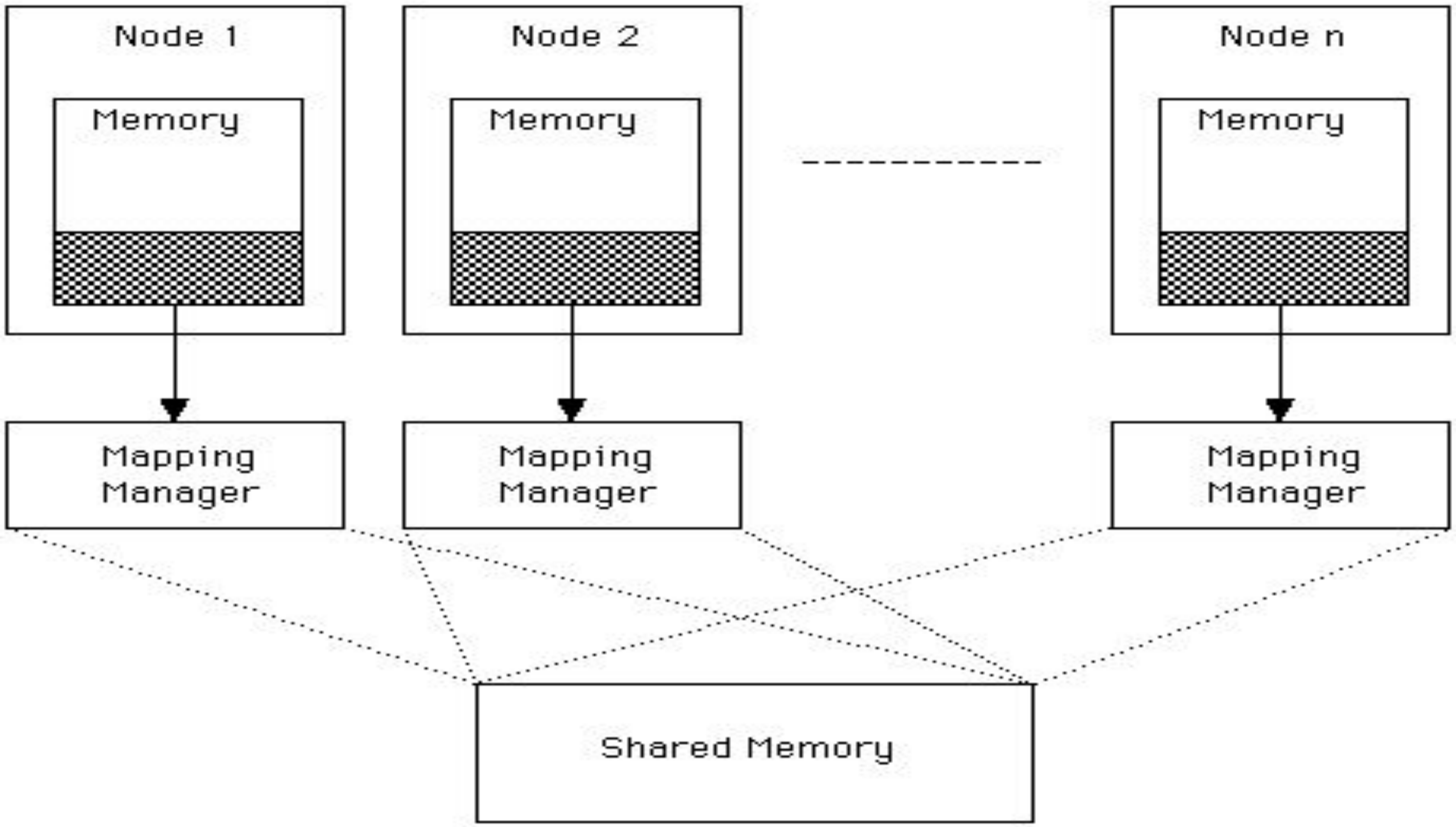
Network OS vs. Distributed OS

Network Operating System	Distributed Operating System
Network Operating System’s main objective is to provide the local services to remote client.	Distributed Operating System’s main objective is to manage the hardware resources.
In Network Operating System, Communication takes place on the basis of files.	In Distributed Operating System, Communication takes place on the basis of messages and shared memory.
Network Operating System is more scalable than Distributed Operating System.	Distributed Operating System is less scalable than Network Operating System.
In Network Operating System, fault tolerance is less.	While in Distributed Operating System, fault tolerance is high.
Rate of autonomy in Network Operating System is high.	While The rate of autonomy in Distributed Operating System is less.
Ease of implementation in Network Operating System is also high.	While in Distributed Operating System Ease of implementation is less.
In Network Operating System, All nodes can have different operating system.	While in Distributed Operating System, All nodes have same operating system.

Communication in Distributed System

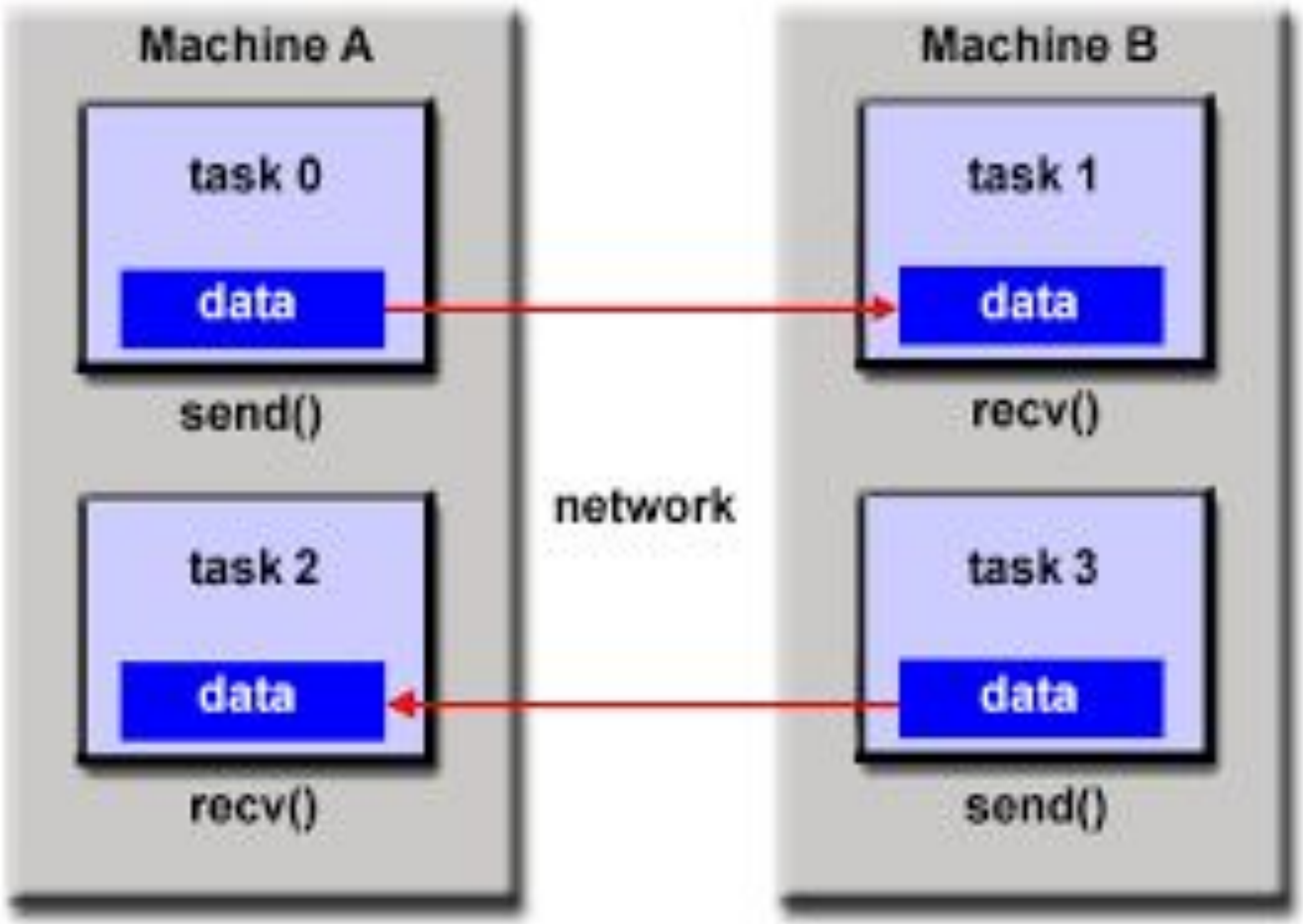
1. Shared memory

- Distributed shared memory (DSM) is form of memory architecture where physically separated memories can be addressed as one logical address space.
- Hera the term shared does not mean that there is single centralized memory but that the address space is shared.
- A distributed shared memory system implements the shared memory model on physically distributed memory system.
- The syntax used for DSM is the same as that of normal centralized memory multiprocessor systems.
- **Read(address)**
- **Write(data, address)**
- Shared memory gives the system illusion of physically shared memory.


Node 1
Memory
Node 2
Memory
Node n
Memory
Mapping Manager
Mapping Manager
Mapping Manager
Shared Memory

2. Message Passing

- Message passing model allows multiple processes to read and write data to the message queue without being connected to each other.
- Message are stored on the queue until their recipient retrieves them.
- Massage passing is the basis of most inter-process communication in distributed system.
- Communication in the message passing paradigm is performed using the **send()** and **receive()** primitives.
- The syntax is generally of the form:
  - **Send(receiver, message)**
  - **Receive(sender, message)**
- The **send()** primitives requires the name of destination process and message data as parameters.
- The **receive()** primitives requires the name of the anticipated sender and should provide the storage buffer for message.


Machine A
task 0
data
send()
task 2
data
recv()
network
Machine B
task 1
data
recv()
task 3
data
send()

Remote Procedure Call

- RPC is an interaction between a client and a server
- Client invokes procedure on sever
- Server executes the procedure and pass the result back to client
- Calling process is suspended (blocked) and proceeds only after getting the result from server

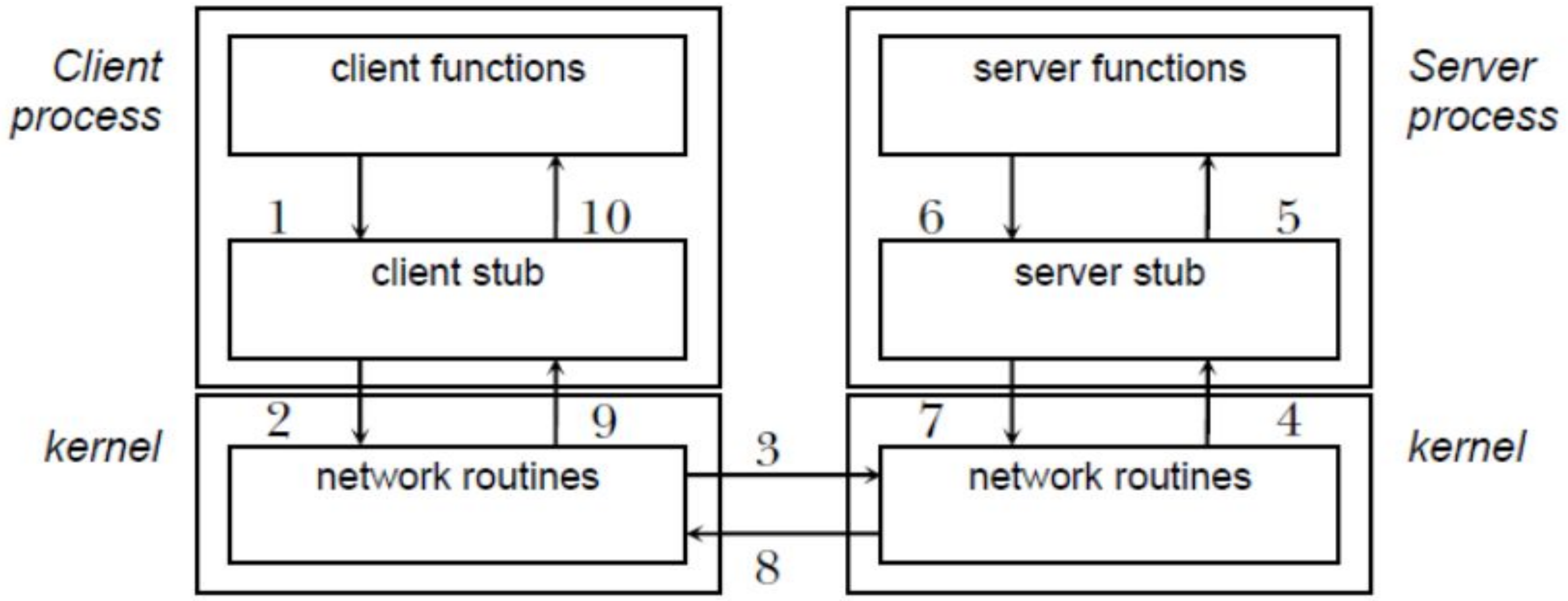


Figure Functional steps in a remote procedure call

1. The client procedure calls the client stub in the normal way.
2. The client stub builds a message including parameters, name or number of procedure to be called etc and calls the local operating system. The packaging of arguments into a network message is called marshaling.
3. The client's as sends the message to the remote OS via a system call to the local kernel. To transfer the message some protocol (either connectionless or connection-oriented) are used.
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 waiting client procedure.

Clock Synchronization

- In distributed system the internal clock of several computer may differ with each other.
- Even when initially set accurately, real clock will differ after some time due to clock drift, cause by clock counting time at slightly different rates.
- Clock synchronization is mechanism to synchronize the all computer in distributed system.
- It is the process of setting all the cooperating systems of distributed network to the same logical or physical clock.

Physical clock

- Physical clock is an electronic device that counts oscillation in crystal at particular frequency.
- Physical clock can be used time stamp an event on that computer.
- clocks whose values must not deviate from the real time by more than a certain amount.
- The time difference between two computer is known as drift.
- Several methods are used to attempt the synchronization of physical clock in distributed system.
  - **Coordinated Universal time (UTC)**
  - **Christian algorithm**
  - **Berkeley Algorithm**

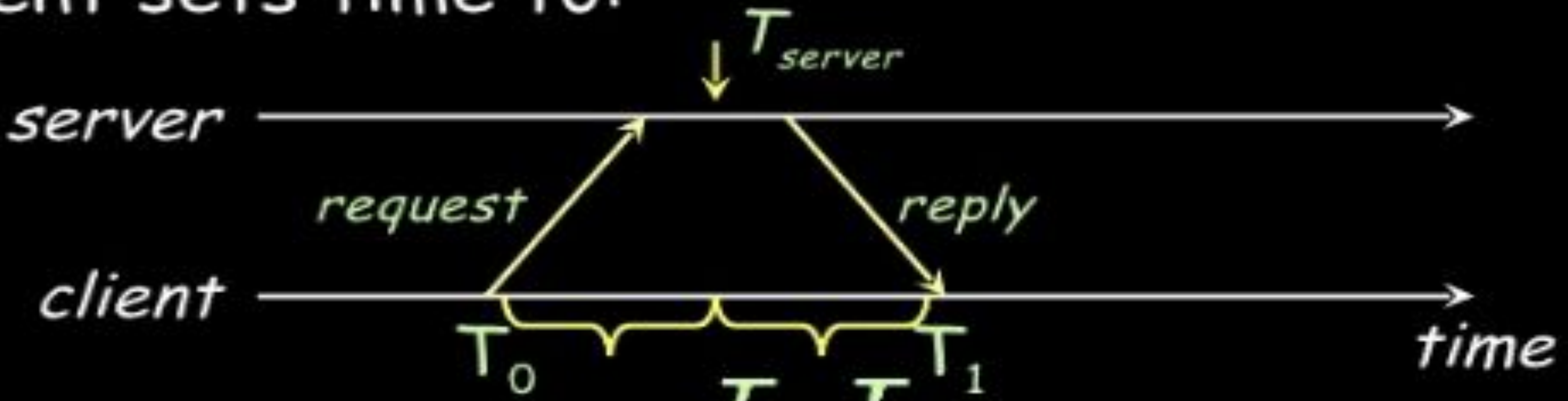
Coordinated Universal Time

- All computers are generally synchronized to a standard time called coordinated Universal Time (UTC).
- UTC is primarily the standard by which the world regulates clock and time.
- It is available via radio signal , telephone lines and satellite (GPS).
- UTC is broadcast via satellites.
- Computer servers and online services with UTC receiver can be synchronized by satellite broadcast.
- Use UTC as reference time to synchronize clock of computers.

Cristian Algorithm

- The simplest algorithm for setting time , it issues a Remote Procedure call to time server and obtain the time.
- A machine send a request time server in d/2 seconds where d=max. difference between a clock and UTC.
- The time server send a reply with current UTC when receive the request.
- The machine measures time delay between time server send the message and machine receiving it.
- Then it uses measures to adjust the clock.

Client sets time to:

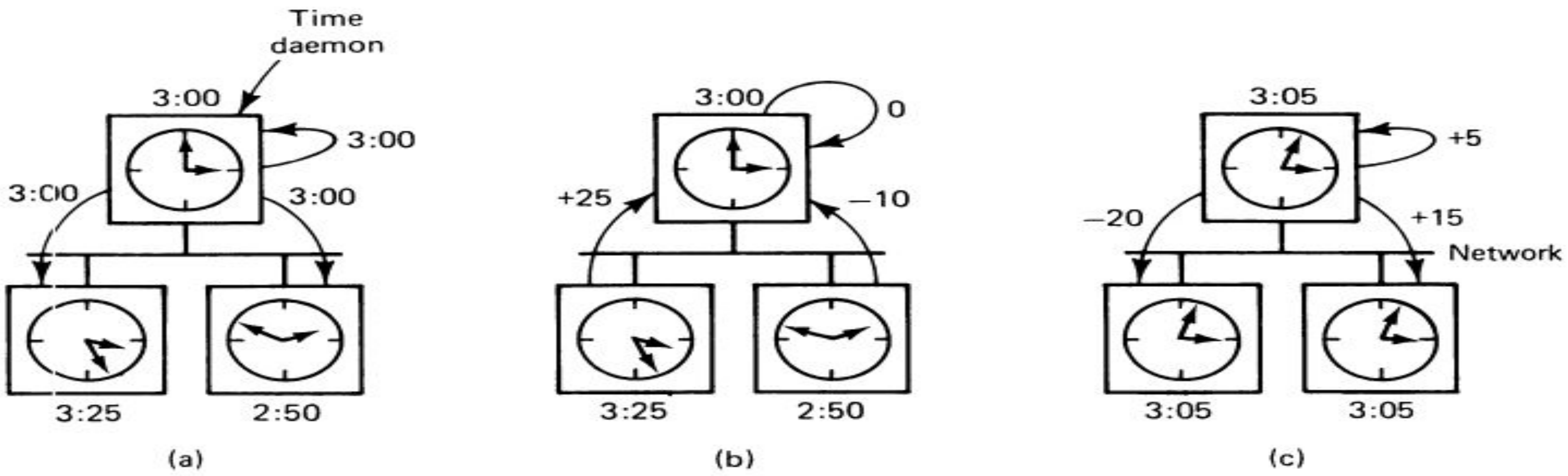


$\frac{T_1 - T_0}{2}$ = estimated overhead in each direction

$$\boxed{T_{new} = T_{server} + \frac{T_1 - T_0}{2}}$$

Berkeley algorithm

- The server polls each machine periodically, asking it for the time.
- A time server will periodically fetch the time from all the time clients, average the results, and then report back to the clients the adjustment that needs be made to their local clocks to achieve the average.
- The Time daemon tells everyone how to adjust their clock.



Logical clock

- Often, it is not necessary for computer to know the exact time, only relative time. This is known as logical time.
- Logical time is not based on timing but on the ordering of events.
- Logical clocks can only advance forward, not in reverse.
- Non-interacting processes cannot share logical clock.
- Computers generally obtain logical time using interrupts to update a software clock. The more interrupts, higher the overhead.
- The most common logical clock synchronization algorithm for distributed systems is **Lamport‟s Algorithm**.
- It is used in situations where ordering is important but global time is not required.

Lamport's Algorithm.

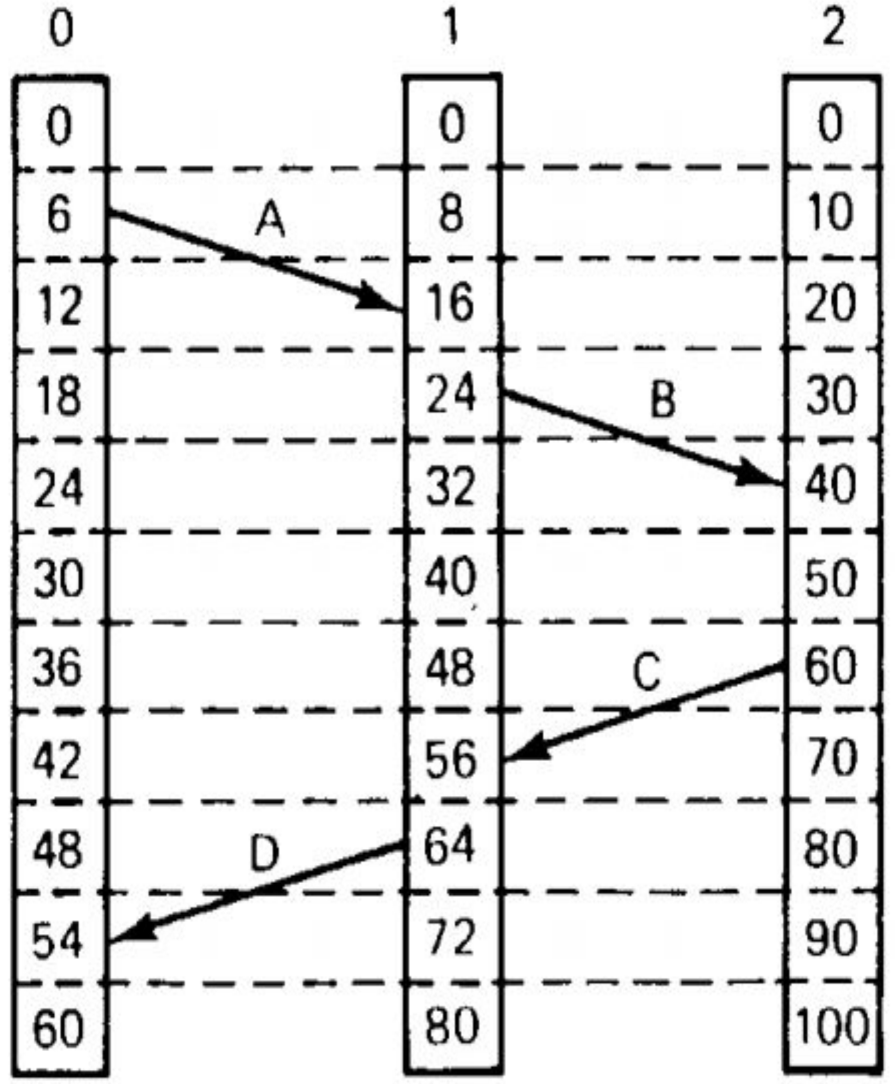
- Lamport invented a simple mechanism by which the happened-before ordering can be captured numerically.(i.e, a□b means 'a' happens before 'b')
- A Lamport logical clock is a incrementing software counter maintained in each process.
- **Algorithm follows:**

  1. *A process increments its counter before each event in that process;*
  2. *When a process sends a message, it includes its counter value with the message;*
  3. *On receiving a message, the receiver process sets its counter to be greater than the maximum of its own value and the received value before it considers the message received.*

- Conceptually, this logical clock can be thought of as a clock that only has meaning in relation to messages moving between processes. When a process receives a message, it resynchronizes its logical clock with that sender.

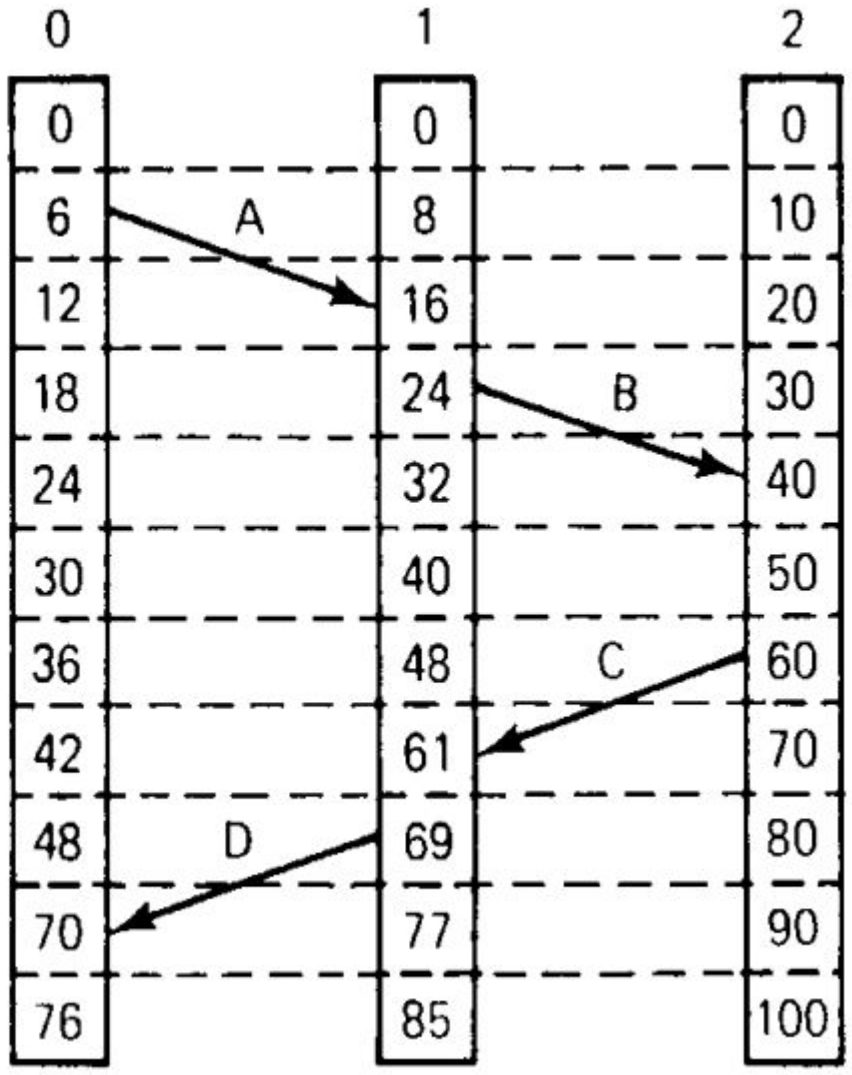
PROCESS EACH WITH ITS OWN CLOCK

- At time 6 , Process 0 sends message A to Process 1
- It arrives to Process 1 at 16(It took 10 ticks to make journey)
- Message B from 1 to 2 takes 16 ticks
- Message C from 2 to 1 leaves at 60 and arrives at 56 -**Not Possible**
- Message D from 1 to 0 leaves at 64 and arrives at 54 -**Not Possible**



LAMPORT'S ALGORITHM CORRECTS THE CLOCKS

- Use "happened-before" relation
- Each message carries the sending time (as per sender's clock)
- When arrives, receiver fast forwards its clock to be one more than the sending time. (between every two events, the clock must tick at least once


0
1
2
0
6
12
18
24
30
36
42
48
70
76
0
8
16
24
32
40
48
61
69
77
85
0
10
20
30
40
50
60
70
80
90
100
A
B
C
D