CS3551 - Distributed Computing

Syllabus:-

UNIT - 1 Introduction:-

Introduction: Definition - Relation to Computer System Components - motivation - Message - Passing system versus shared memory systems, - Primitives for distributed communication - Synchronous Versus Asynchronous executions. - Design issues and challenges; A model of distributed computations: A distributed Program - A model of distributed executions - models of communications Network Global state of a distributed system.

Important Questions:-

1.) Discuss the design issues and challenges in distributed system from a system Perspective.

2.) Explain how a Parellel system differ from a distributed system.

3.) Why global states are essential in distributed Computing systems?

4.) Logical clock categories in distributed systems with an example.

5.) Briefly explain the ~~message~~ Message Passing system with shared memory.

6.) Explain the Primitives methods of distributed system?

Scanned with OKEN Scanner

Computation :-

→ The Process of Computation is working on a single Processor.

→ The uni Process computing can be termed as centralized computing.

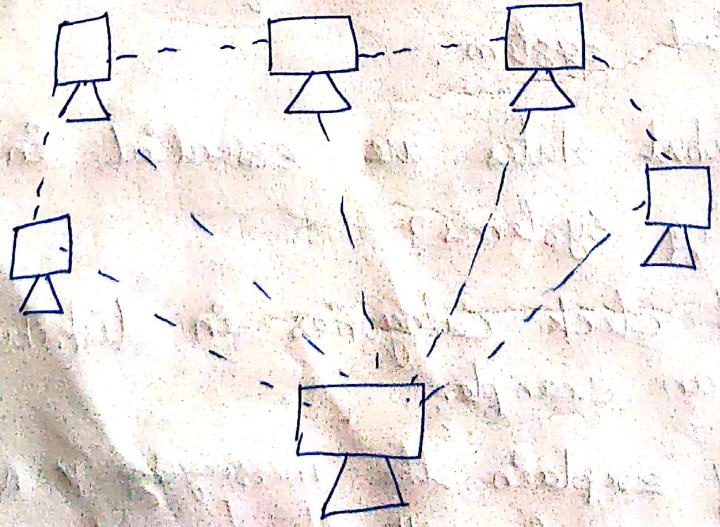
Distributed system :-

A distributed System is collection of independent computer, interconnect via network, capable of collaborating on a task.

Distributed Computing Performed in a distributed syst

Example

The multiple system connected to the network will appear as a single system to the user.



Scanned with OKEN Scanner

It deals with two aspects

1. Deals with hardware. – The machines linked in a distributed system are autonomous
2. Deals with software – It gives an impression to the user that they are dealing with a single system.

Features of distributed system :-

* NO Common Physical Clock → Introduce element of distributed system – rise inherent asynchrony amongest Processor.
* NO Shared memory → Message Passing Communication, absence Physical clk.
* Geographical seperation → The geographically wider apart the Processor, more representative is the system of dist. System
* Autonomy and heterogeneity → Processor are loosely coupled. different speed & different O.S.

Issues

1. Heterogeneity
2. Openness
3. security
4. scalability
5. Failure handling
6. concurrency
7. Transparency
8. Quality of Service.

QOS Parameter

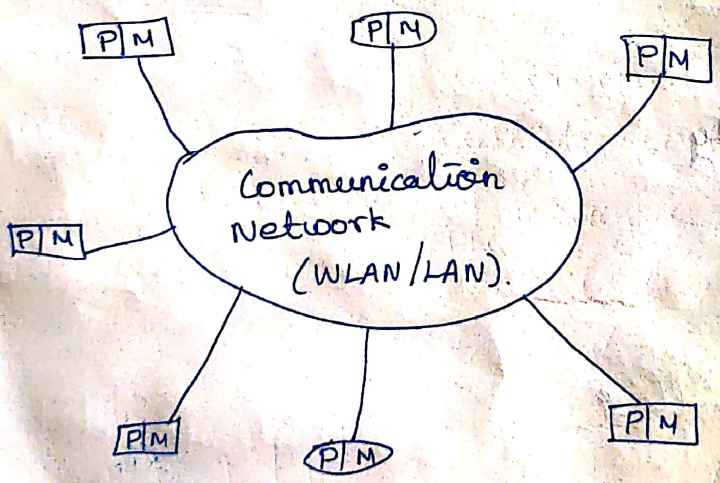
* Performance
* Reliability
* Availability
* Security.

Scanned with OKEN Scanner

(X) Difference between centralized and distributed S/S

Centralized S/S	distributed S/S.
several jobs are done on a Particular central Processing unit (CPU).	The jobs are distributed among several Processors.
They have shared memory and shared variables	The Processor are interconnected by a computer network
clocking is Present	They have no global state no shared memory and no shared variables
	No global clock.

Relation to computer system Components :-



Example of distributed system.

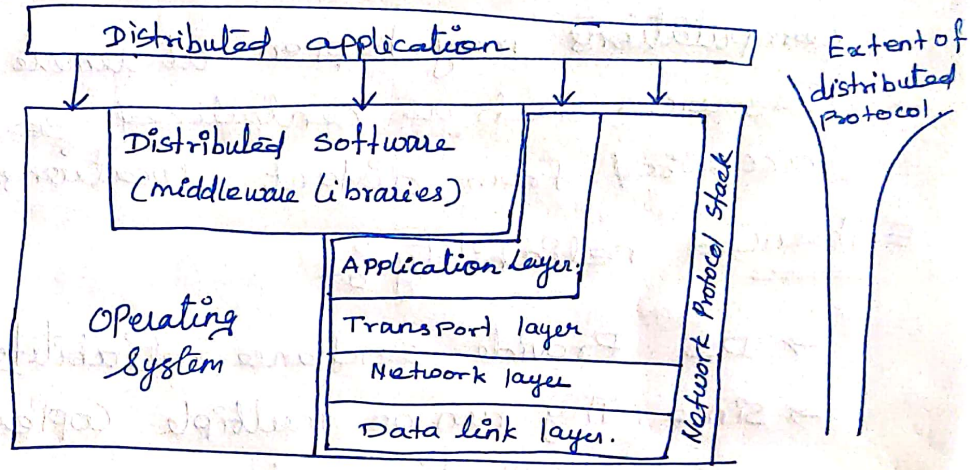
→ Each Computer has memory & Processing unit.

→ The Computed Connected by a Communication N/w

→ Each system connected to distributed N/w host distributed Software which is a middleware technology.

→ This drive the distributed system (DS) at the same time Preserve the heterogeneity of the DS.

→ The term Compulation or run in a distributed system is the execution of Process to achieve a common goal.



→ The Interation of layer of the N/w & O.S in the middleware.

→ The middleware contains important library function for faciliating the operations of DS.

→ The D.S uses a layerd architecture to break down complexity of system design.

→ The middleware is distributed software that drives the distributed System, while Providing transperency of heterogeneity at the platform level.

Example of middleware ✓ [message Passing Interface (MPI)

object Management Grap (OMG), Common object Request

Remote Procedure call (RPC)

Motivation :-

→ The keypoint to act driving force behind DS.

* Inherently distributed computations → DS can process (internally). the computations at geographically remote locations.

* Resource sharing :- The hardware, database, special libraries can be shared b/w systems without owning a dedicated copy or a replica., cost is effective & rel

* Access to geographically remote data and resource :- Computations may happen at remote locations. Resources such as centralized servers can also be accessed from distant locations.

Enhanced reliability :-

→ DS Provide enchance reliability,
→ Since, They run on multiple copies of resources.
→ The Distribution of resources at distant locatio make them less susceptible for faults.
→ [Resources such as centralized servers can also be accessed from distant locations.] *

The term Reliability comprises of

1. Availability → Resource/service by the resource should be accesible at all times.
2. Integrity → The value/state of the resource should be correct and consistent.

3. Fault - Tolerance: Ability to recover from System failure.

4. Increased Performance/cost ratio: The Resource Sharing and remote access feature of DS. naturally access increase the Performance/cost ratio.

5. Scalable:- The number of system operating in a distributed environment can be increased as the demand increase.

Scanned with OKEN Scanner

Message-Passing system Vs shared memory System

→ Communication among Processors takes place via shared data variables for synchronization among the Processor.

→ The Communications between the task in multiprocessor system takes place through main modes:

Message Passing system:-

→ This 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.

→ Message queues are quite useful for inter Process communication and are used by most operating system

Shared memory systems:-

→ The shared memory is the memory that can be simultaneously accessed by multiple Processes.

→ This is done so that the Processes can communicate each other.

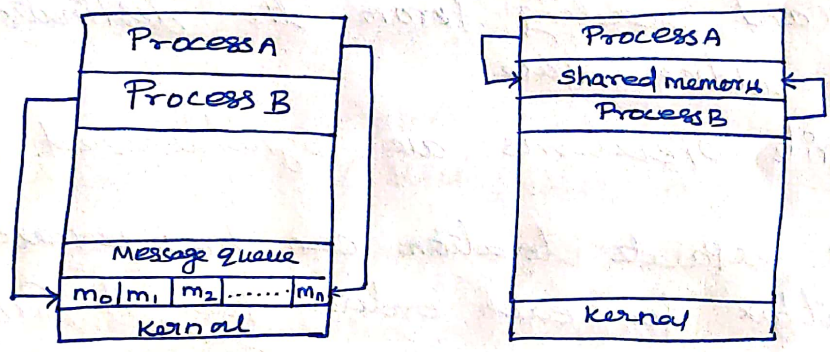
→ Message are stored on the queue until their recipient retrieves them.

→ Message queues are quite useful for inter Process communication and are used by most operating system

→ Communication among Processors takes place through shared data variables, and control variables for synchronization among the Processor.

→ Semaphores and monitor are Common synchronization mechanisms on shared memory systems.

→ When shared memory model is implemented in a distributed environment, it is termed as distributed shared memory.



Message Passing model Shared memory model.

Inter-Process communication models.

Difference b/w message Passing and shared memory models:-

Message Passing	Distributed shared memory
Servica offered: → Variable have to be marshalled from one Process, transmitted and unmarshalled from one Pro. into other variables at the receiving Process.	The Process share variables directly, so no marshalling and unmarshalling. Shared variables can be named, stored and accessed in DSM.
→ Process can communicate with other Processes. They can be Protected from one another by having Private address spaces.	Here, a Process does not have Private address space. So one Process can alter the execution of other.
→ This technique can be used in heterogeneous Computer	→ This cannot be used to heterogeneous Computers.
→ Synchronization b/w Process is through message Passing Primitives.	Synchronization is through locks & semaphores. DSM may execute with non-overlapping lifetimes.
→ Process communicating via message Passing must execute at the same time.	Process Communicate through
→ Efficiency → All remote data accesses are explicit and Programmer is always aware.	Any Particular read or update may or maynot involve Communication by the underlying

Scanned with OKEN Scanner

Emulating message-Passing on a Shared memory system (MP → SM)

→ The shared memory system can be made to act as message passing system.

→ The shared address space can be partitioned into disjoint parts, one part being assigned to each processor.

→ Send and receive operations can implemented by writing to and reading from the destination/sender processor's address space.

→ Read & write operations are synchronized.

→ Specially, a separate location can be reserved as the mailbox for each ordered pair of processes.

Emulating shared memory on a message-Passing system (SM → MP)

→ This is also implemented through read and write operations.

→ Each shared location can be modeled as a seperate process.

→ write to a shared location is emulated by sending an update message to the corresponding owner process and read operations to a shared location emulated by sending a query message to the owner process.

→ This emulation expensive as the processor has to gain access to other processor memory location.

→ The latencies involved read and write operations may be high even when using shared memory emulation. beoz the read and write operations are implemented by using Network wide communication.

Scanned with OKEN Scanner

PRIMITIVES FOR DISTRIBUTED COMMUNICATION :-

1. Blocking / Non-blocking / synchronous / Asynchronous.

→ Message send and Message receive Communication Primitives are done through Send() & Receive(). respectively.

→ Send Primitive has two Parameter
   1. The destination
   2. Bubber (holds the data to be Sent).

→ Receive Primitive also two Parameter
   1. Source (The source from which data is received)
   2. user bubber (user bubber in it is received).

There are two ways of Sending data when the Send Primitives called.

→ Bubbered :
   * The Standard option copies the data from the user bubber to the kernal bubber.
   * The data gets copied from the kernal bubber on to the network.
   * For the Receive Primitives, the bubbered is usually required bcoz. the data may already have arrived when the Primitive is invoked, needs storage place in kernal.

→ un Bubbered :-
   The data gets occupied or copied directly from the user bubber onto the network.

Scanned with OKEN Scanner

Blocking Primitives :-

→ The Primitives commands wait for the message to be delivered.

→ The execution of the Process is blocked.

→ The Sending Process must wait for acknowledgment is made by receiver.

→ Receiving wait for expected message from Sending Process

→ The Receipt is determined by Polling common bubber or interrupt.

→ This is a form of Synchronization or Synchronous communication.

→ Primitive is blocking, if Control returns to the invoking Process after the Processing for the Primitive Completes.

Scanned with OKEN Scanner

Non-Blocking Primitives :-

→ If send is non blocking, It returns control to the caller immediately, before the message sent.

→ Advantage — Sending Process can continue computing in parallel with the message transmission, instead of having the CPU go idle.

→ This form is asynchronous communication

→ A Primitive is non-blocking. If control returns back to the invoking Process immediately after invocation, even though the operation not completed.

→ For non-blocking send, control returns to the Process even before the data is copied out of the user buffer.

→ For non-blocking receive, control returns to the Process even before the data may have arrived from the sender.

Scanned with OKEN Scanner

Synchronous :-

→ A send or a Receive Primitive is synchronous ie both the send( ) and Receive( ) handshake with each other.

→ The Processing for the send Primitive Completes only after the invoking Processor learns.

→ other corresponding Receive Primitives has also been invoked and that the receive operation has been completed.

→ The Processing for the Receive Primitives Completes when data to be received is copied into other receiver's user buffer.

Asynchronous :-

→ A Primitive is said to be asynchronous, if control returns back to the invoking Process after the data item to be sent has been copied out of the user specified buffer.

→ It does not make sense of define asynchronous Receive Primitives.

→ Implementing non-blocking operation are tricky.

Scanned with OKEN Scanner

→ For non blocking, Primitives, a return Parameter on the Primitive call returns a system-generated handle which can be later used to check the status of the completion of the call.

→ The Process can check for the completion
  * Checking if handle has been flagged or Posted.
  * Issue a wait with a list of handles as Parameter: usually blocks untill one of the Parameter handle is Posted.

Send & Receive Primitives can be implemented in Four modes:-

* Blocking Synchronous
* Non-Blocking Synchronous
* Blocking Asynchronous
* Non-blocking Asynchronous.

Four modes for Send operation:-

1. Blocking Synchronous Send :-
   * The data copied from the user buffer to the kernal buffer, and is then sent over the N/W.
2. * After data copied → Receiver system buffer and Receiver call has been issued,
   * An ~~ack~~ acknowledgement back to the Sender cause control to the return

to the Process that invoked the Send operations and Completes the Send.

2. Non-blocking Send :-

→ control return back to the invoking Process as soon as the Copy of data from the user bubber to the Kernal bubber is initiated.

→ A Parameter in non-blocking call also gets set with handle of a location that user Process can later check for the completion of the Synchronous Send operation.

→ The location gets Posted after an acknowledge-ment returns to from the receiver.

→ The user Process can keep checking for the Completion of non blocking Synchronous Send by testing the returned handle.

→ It can invoke the blocking wait operation on the returned handle.

Blocking Asynchronous Send :-

→ The user Process that invoke the Send is blocked until the data is copied from the user's bubber to the Kernal bubber.

Scanned with OKEN Scanner

Non-blocking asynchronous send :-

→ The user Process that invokes that the Send is blocked until the transfer of the data from the user's buffer to the kernal buffer is initiated.

→ control returns to the user Process as soon as this transfer is initiated, and a Parameter in the non-blocking call also get set with the handle of a location that the user Process

→ Process can check later using the wait operation for the completion of asynchronous Send.

Modes of receive operation :-

Blocking Receive :-

The receive call blocks until the data expected arrives and is written in the Special Specified user buffer

Then Control is returned to the Process.

Non-blocking Receiver :-

The Receive call will cause the Kernal to register the call and return the handle of a location that the user Process can later check for the Completion of non-blocking Receive operation.

Scanned with OKEN Scanner

Process synchrony :-

Process synchrony indicates that all the Process execute in lock-step with their clocks synchro-ized.

Libraries and standards :-

1. Message Passing interface (MPI) :

   Standardized and Portable message Passing system to function on a wide Variety of Parallel computers.

2. Parallel Virtual machine (PVM) :

   It is software tool for Parallel networking of Computers.

3. Remote Procedure call (RPC) :-

   → It is common model of request repl Protocol.

4. Remote method Invocation (RMI) :-

   → A Programmer can write object oriented Programming in which objects on different Computers can interat in a distributed N/W.

5. Remote Procedure call (RPC) :-

   → Constructing distributed, client Server based applications.

6. Common object request Broker Architecture (CORBA) ... messaging mechanism by which object distributed

over a network can communicate with each other irrespective of platform and language used to develop those objects.

Synchronous Vs Asynchronous Executions :-

The execution of Process in distributed systems may be Synchronous and asynchronous.

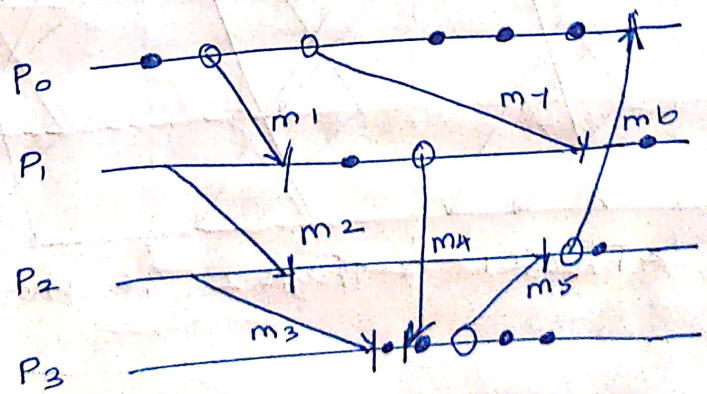
Asynchronous Execution :-

→ A Communication among Process is Considered asynchronous, when every Communicating Process can have a different observation of the order of the message being exchanged.

→ There is no Process asynchrony. there is no bound on the drift rate Processor Clocks.

→ Message delays are finite but unbounded.

→ no upper bound on the time taken by a Process.



Scanned with OKEN Scanner

Synchronous Execution :-

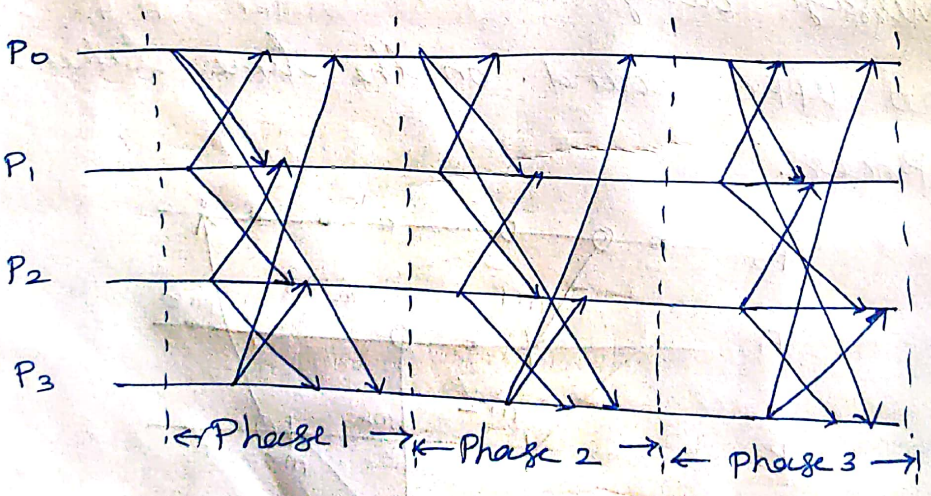
→ A Communication among Processes is considered synchronous when every Process observes the same order of message within the System.

→ Every individual Process in the System observes the same total order of all Processes which happen within it.

→ Processors are synchronized and the clock drift rate b/w two Processor is bounded.

→ Message delivery times are such that the occur in one logical step or round.

→ upper boundon the time taken by a Process to execute a step.



Scanned with OKEN Scanner

Emulating an asynchronous system by a Synchronous system (A→S) :-

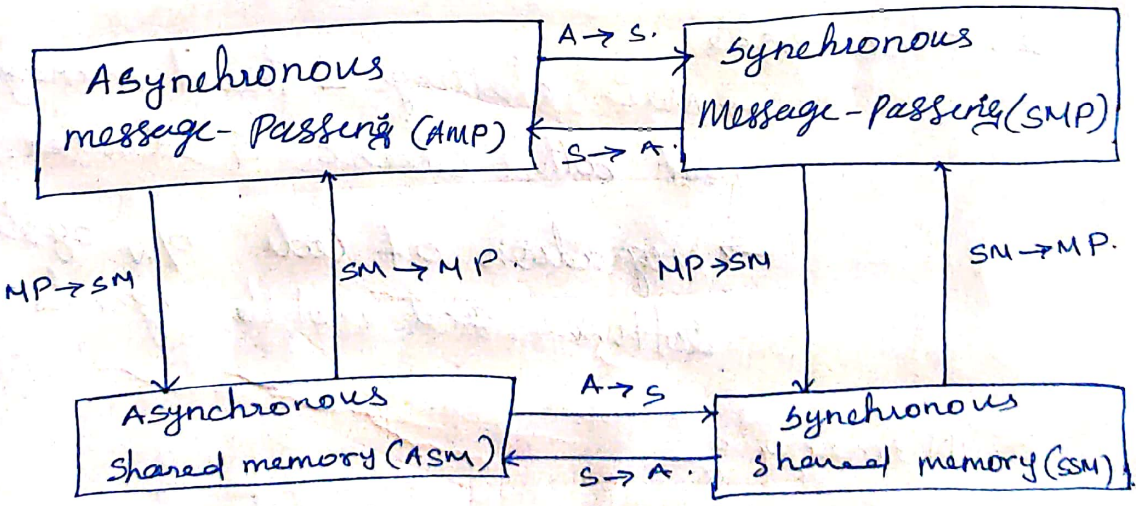
→ An asynchronous Program can be emulated on a Synchronous system fairly trivially as the Synchronous system is special case of an asynchronous system.

→ All Communication finishes within the same round in which its is initated.

Emulating a Synchronous system by an asynchronous system (S→A) :-

→ A Synchronous Program can be emulated on an asynchronous system using a tool called synchronizer.

Emulation for a fault free system :-



→ If system A can be emulated by System B, denoted A/B, and if a Problem is not solvable in B., then it is also not solvable in A.

→ If Problem is solvable in A, it is also solvable in B,

→ In sense, all four classes are equivalent it terms of computability in failure-free.

Scanned with OKEN Scanner

Design Issues and challenges in distributed systems

1. Issues related to system and operating system design.
2. Issues related to algorithm design
3. Issues arising due to emerging technologies

Issues related to System and operating system design

Common challenges to be addressed in designing a distributed system.

1. Communication :-
   → The task involves designing suitable communication mechanism among the various Processes in the N/w.
   Ex : RPC, RMI

2. Processes :
   → Main challenge – Thread management at both client & server.
   → Migration of code b/w system design software and mobile agents.

3. Naming :-
   The location Identifier, remote and highly varied geographical location make this task.

Scanned with OKEN Scanner

4. Synchronization
5. Data storage and access schemes.
6. Consistency and replication
7. Fault tolerance.
8. Security
9. Application Programming interface (API) and transparency
10. Access transperency
11. Location transperency
12. Migration transperency
13. Replication transperency
14. Concurrency transperency
15. Failure transperency
16. Scalability and modularity

Algorithmic challanges in distributed Computing :-

1. Designing useful execution models and frameworks.
2. Dynamic distributed graphic algorithms and distributed routing algorithm.
3. Time and global state in a distributed system.
4. Synchronization / co-ordination mechanism.
5. Physical clock Synchronization
6. Leader election
7. Mutual ~~execution~~ exclusion

Scanned with OKEN Scanner

8. Deadlock detection and resolution.
9. Termination detection.
10. Garbage collection
11. Group communication, multicast and ordered message delivery.
12. Monitoring distributed events and Predicates.
13. Distributed Program design and verification tools.
14. Debugging distributed Program
15. Data replication, consistency models and caching
16.) World wide web design - caching, Searching, Scheduling.
17. Distributed shared memory abstraction
18. Wait free algorithm
19. Mutual exclusion
20. Register construction
21. Reliable and fault tolerance distributed System
22. Consensus algorithm
23. Replication and replica management
24. Voting and quorum System
25. Distributed database and distributed Commit.
26. Checking Pointing and recovery algorithms
27. Failure detectors
28. Load balancing
29. Data migration
30. Computation migration
31.

31. Distributed Scheduling
32. Real time Scheduling
33. Performance
34. Metrics
35. Measurement methods / tools ;-

Applications of distributed Computing and newer challenges

1. Mobile Systems.
   → shared broadcast medium.
   Such as battery power consumption, interfacing with wired internet, signal processing.

2. Base - station approach (cellular approach) :-
   → Geographical region is divided into hexagonal Physical location called cells.

3. Ad-hoc network approach ;-
   → This is an infrastructure - less approach which do not have any base station to transmit signals.

4. Sensor networks.
   → Electro mechanical interface that is capable of sensing Physical Parameter.

5. ubiquitous or Pervasive Computing
   → embedded in the environment to Performable application function
   Ex - Intelligent devices, Smart home etc.

6. Peer to Peer Computing
   → All interation among the Processor are at a same level.

7. Publish - Subscribe, content distribution and multimedia
   → Publish
   → subscribe.

8. Distributed agents

9. Distributed data mining

10. Grid Computing

11. Security in distributed system.

A model of Distributed computations: Distributed Program :-

→ A distributed Program is composed of set of asynchronous Process.

→ Global memory and communicate by passing messages. These Processor do not share a global clock.

→ Global state of distributed computation is compose of the states of the Processes and the communicatio channels.

Model of distributed executions :-

→ The execution of a Process consist of sequential execution of its actions.

→ Internal event changes the state of the Process that send the message the channel on which the message is sent.

Scanned with OKEN Scanner

→ The execution of Process P_i Produces a sequence of event $e_1, e_2 \ldots$ and it's denoted by H_i;

$H_i = (h_i \rightarrow_i)$

P_i and $\rightarrow$ casual dependencies among events P_i.

→ msg → dependency exist due to message passing b/w two events.

→ send (m) → msg rec (m).

Models of Communication N/w :-

Three types of Communication N/w.

1. FIFO.
2. non-FIFO.

casual ordering (CO) :-

→ Relation b/w three models given by

$$CO \subset FIFO \subset N\text{-}FIFO$$

CO : For any two messages m_{ij} and m_{kj}

If send $(m_{ij}) \rightarrow$ send (m_{kj}),

then rec $(m_{ij}) \rightarrow$ rec (m_{kj}).

Scanned with OKEN Scanner

Global state :-

Distributed Snapshot represent a state in which the distributed system might have been in. A Snapshot of the system is a single configuration of the system.

→ The global state of distributed system is collection of local states of its components namely, the Processes and the Communication channels.

→ The state of a channel is dibbicult to state

→ The state of a channel is given by the set of message in transit in channel.

→ The state of channel is dibbicult to state formally because a channel is distributed entity

→ Its state depends upon the state of the states of the Process, it connects.

→ $SC_{i,j}^{x,y}$ → State of a channel C_{ij} defined

$$SC_{ij}^{x,y} = [m_{ij} | send(m_{ij}) \leq_L s_i^x \wedge rec(m_{ij}) \not\leq_L s_j^y]$$

→ If we have recorded that a Process P has received a received a message from a Process Q, then we should have also recorded that Process Q had actually send that message.

Scanned with OKEN Scanner

→ Snapshot will contain the recording of messages that have been received but never sent.

→ The reverse condition ((Q) has sent message that P has not received) is allowed.

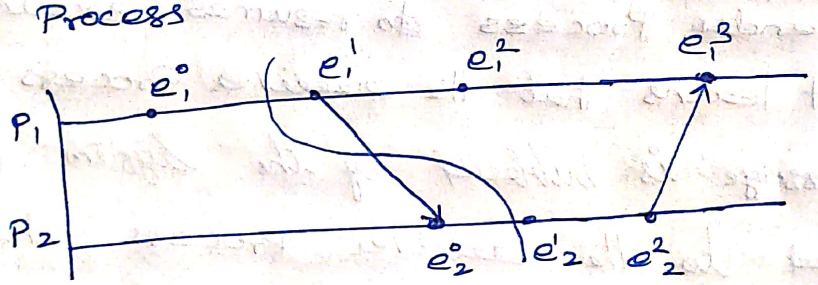
→ The notion of a global state can be graphically represented by what is called a cut.

→ A cut represent the last event that has been recorded for each process.

→ history of each process.

$$\text{history}(P_i) = h_i = \langle e_i^0, e_i^1, \ldots e_i^n \rangle$$

→ Each event either is an internal action of the process



Types of cuts :-

→ Consistent states - state not violate causality.

→ In Consistent states - They are not meaningful in the sense that a distributed system can never be in an inconsistent state.

Scanned with OKEN Scanner

Models of Process Communications:

→ Synchronous - * The Sender Process blocks until the message has been received by receiver Process

* The Sender Process resumes execution after it learns that the receiver Process has accepted the message.

Asynchronous:

It is non-blocking communication where the sender and the receiver do not synchronize to exchange a message.

The Sender Process do resumes execution only after it learns that the receiver Process has accepted The message is buffered by the System and is delivered to the receiver Process when it is ready to accept the message.

A buffer overflow may occur if a Process sends a large number of message in a burst to another Process that causing message burst.

Scanned with OKEN Scanner

Logical Time :-

Logical clocks are based on capturing chronological and casual relationship of Process and ordering events based on these relationships.

Three types of logical clock.

* Scalar clock
* Veebor clock
* Matrix clock.

Difference b/w Physical and logical clock :-

Physical clk	logical clk.
A Physical clock is a Physical Procedure combined with a strategy for measuring that Procedure to record the Progression of time	A logical clock is a Component for catching sequential and casual connections in a dispersed framework.
The Physical clocks are based on cyclic Procedure such as a celestial rotation	A logical clock allows global ordering on events from different Process.

Frame work for a system of logical clocks :-

A frame work, system of logical clocks C is a function that maps an event e in a distributed System to an element in the time domain T denoted as C(e)

$C : H \rightarrow T$ such that

For any two events e_i and e_j.

$$e_i \rightarrow e_j \Rightarrow C(e_i) < C(e_j)$$

Scanned with OKEN Scanner

This monotonicity Property is called the clock Consistency Condition.

when T and C satisfy the following condition

$$e_i \rightarrow e_j \Leftrightarrow C(e_i) < C(e_j)$$

Physical clock synchronization : Network Time Protocol (NTP):-

→ Centralized system do not need clock synchronization as they work under a common clock.

→ The time in distributed system is measured in the following contexts.

* The time of the day at which an event happend on a specific machine in the network.
* The time interval between two events that happened on a ~~specific~~ different machine in the network
* The time the relative ordering of events that happened on different machines in the network.

→ "Clock Synchronization is the Process of ensuring that Physically distributed Processors have a common notation of time".

Scanned with OKEN Scanner

Basic Terminologies :-

→ Time :

→ Frequency :

→ offset :

→ skew :

→ Drift (rate) :

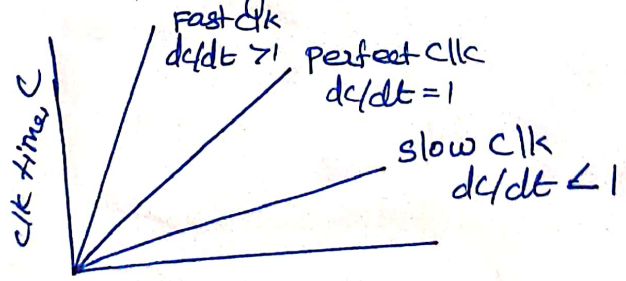
Clocking Inaccuracies :-

Physical clocks are synchronized to an accurate real-time standard like UTC (universal co-ordinated Time).

Due to inaccuracy.

a timer (clock) $1 - \rho \leq \frac{dc}{dt} \leq 1 + \rho$

1. offset delay estimation
2. Clock offset and delay estimation



Behaviour of clocks.

→ offset and delay estimation between Processes from same server.

→ offset and delay estimation between Processes from different servers.

Scanned with OKEN Scanner