ANNEXURE - I

ANNA UNIVERSITY
CHENNAI - 25

College Code	0	0	0	4							
College Name	MADRAS INSTITUTE OF TECHNOLOGY										
Register Number	2	0	1	8	5	0	3	0	6	0	
Name of the Candidate	A S SIVA HARI										
Degree	BE										
Branch	CSE						Semester	6			
Question Paper Code											
Subject Code	C	S	6	3	0	3					
Subject Name	DISTRIBUTED SYSTEMS										
Date	10	08	21	Session	FN	AN					
No. of Pages used	23	In words	Twenty three								
All particulars given above by me are verified and found to be correct											
Signature of the Student with date	Ashwath 10/8/21.										

For Office Use Only

Instructions to the Candidate: Put Tick mark (✓) for the questions attended in the tick mark column against each question

PART – A			PART – B & C									
Question No.	✓	Marks	Question No.		(i) ✓	(i) Marks	(ii) ✓	(ii) Marks	(iii) ✓	(iii) Marks	Grand Total (in words)
1			11	a							
2				b							
3			12	a							
4				b							
5			13	a							
6				b							
7			14	a							
8				b							
9			15	a							
10				b							Grand Total
			16	a							
				b							
Total											

Declaration by the Examiner: Verified that all the questions attended by the student are valued and the total is found to be correct

Date	Name of the Examiner	Signature of the Examiner

PART-B

15. Lamport's algorithm.

Lamport was the first to give a distributed mutual exclusion algorithm as an illustration of his clock synchronization scheme. Let R_i be the request set of site S_i i.e the set of sites from which S_i needs permission when it want to enter CS. In Lamport's algorithm, $\forall i : 1 \leq i \leq N :: R\{S_1, S_2 \ldots S_N\}$. Every site S_i keeps a queue, request_queue i, which contains mutual exclusion requests ordered by their timestamps. This algorithm requires messages to be delivered in FIFO order between every pair of sites.

The Algorithm

Requesting the critical section

1. When a site S_i wants to enter the CS, it sends REQUEST (tsi, i) message to all the sites in its request set R_i and places the request on request_queue i (tsi is the timestamp of the request).

2. When a site S_j recieves the REQUEST (ts_i, i) from site S_i, it returns a timestamp message to S_i and places site S_i's request on request_queue$_j$.

Executing the critical section

1. site S_j enters the CS when the two following condition hold :

a) [L1 :] S_i has received a message with timestamp larger than (ts_i, i) from all other sites

b) [L2 :] S_i's request is at the top request_queue$_i$.

Releasing the critical section

1. site S_j, upon exiting the CS, removes its request from the top of its request queue and sends a timestamped RELEASE message to all the sites in its request set.

2. When a site S_j recieves a RELEASE message from site S_i, it removes S_i's request from its request queue.

When a site removes a request from its queue, its own request may come at the top of the queue, enabling it to enter CS. Thus, the algorithm executes CS requests in the increasing order of timestamps.

16 Suzuki - Kasami's broadcast algorithm.

In Suzuki - Kasami's algorithm, if a site wants to enter the CS but does not have token, it broadcasts a REQUEST message for the token to all other sites. A site that posseses the token sends it to the requesting site upon the receipt of the REQUEST message. when it is executing the CS, it sends the token only after it has completed the execution of CS.

Requesting the critical section

1. If the requesting site s_i does not have the token, then it increments its sequence number, $RN_i[i]$ and sends a REQUEST (i, sn) message to all other sites (sn is the updated value of $RN_i[i]$).

2. When a site S_j recieves this message it sets $RN_j[i]$ to $\max(RN_j[i], sn)$. If S_j has the idle token, then it sends the token to S_i if $RN_j[i] = LN[i] + 1$.

Executing the critical section

1. Site S_j executes the critical section when it has recieved the token.

Releasing the critical section.

Having finished the execution of CS, site S_i takes the following actions:

1. It sets $LN[i]$ element of the token array equal to ~~RNi~~ $RN_i[i]$.
2. For every site S_j whose ID is not in the token queue, it appends its ID to the queue if $RN_i[j] = LN[j] + 1$.
3. If token queue is non empty after the above update, then it deletes the top site ID from the queue and sends the token to the site indicated by the ID.

Correctness

Mutual exclusion is guaranteed because there is only one token in the system and a site holds the token during the CS execution

The requesting site enters the CS in finite time.

proof: Token requests messages of a site S_i reach each other sites in finite time. Since one of these sites will have token in finite time, site S_i's request will be placed in the token queue in finite time. Since there can be at most N-1 requests in front of the this request in the token queue, site S_i will get the token and execute CS in finite time

17. In the AND model, a process request more the one resources simultaneously and request is satisfied only after all the request resources are granted to the process.

The outdegree of a node in WFG (wait-For-Graph) for AND model can be more than 1.

A presence of cycle in Wait-For-Graph indicates a dead lock in the AND model In the AND model several wait for dependencies are sent along the edges of the WFG graph. When a deadlock occurs, a resolution method sets into this iteration. Thus the process recieved starts sending release messages

The out of order sending and receiving of wait-for and release messages result in a false or phantom deadlocks. When a wait-for dependency is broken, the corresponding information must be cleaned from the system. If not cleaned in a timely manner, it may result in detection in phantom deadlocks

Therefore, In AND model false dealocks can occur due to dead lock resolution in distributed systems

18. Checkpoint Based Recovery

In the checkpoint based recovery approach, the state of the each process and the communication channel is frequently checkpointed so that upon a failure, the system can be restored to a global consistent set of checkpoints. However a checkpoint based rollback does not guarantee that pre failure execution can be deterministically regenerated after a rollback. These are three types of rollback techniques

(i) Uncoordinated checkpointing.
(ii) Coordinated checkpointing
(iii) Communication-induced checkpointing.

Uncoordinated checkpointing

In this technique, each process has autonomy in deciding when to take checkpoints. This eliminates synchronization overhead as there is no need for coordination between process and it allows processes to take checkpoints when it is most convenient or efficient. The main advantage is the lower run time during normal execution

because no coordination among processes is necessary. Autonomy in taking checkpoints also allows each process to select appropiate checkpoints position. But there are certain drawbacks.

* There is the possibility of domino effect during a recovery, which may cause the loss of a large amount of useful work.
* Recovery from a failure is slow because process need to iterate to find a consistent set of checkpoints. Since no coordination is done at the time the checkpoint is taken, checkpoints taken by a process may be useless checkpoints.
* Uncoordinated forces each process to maintain multiple checkpoints and to periodically invoke a garbage collection algorithm to reclaim the checkpoints that
* It is not suitable for applications with frequent output commits

When a failure occurs, the recovery process initates a rollback by broadcasting dependency request message, when a process recieves this message it stops its execution and replies back

The initator then calculates the recovery time based on the global dependency information and broadcasts a rollback request message containing the recovery line.

Coordinating checkpointing.

In this technique processes orchestrate their checkpointing activities so that all local checkpoints form a consistent global state. It simplifies recovery and it is not susceptible to the domino effect. The main drawback is the large latency involved in committing the output, as a global checkpoint is needed before its message is sent to the OWP.

Since perfectly synchronized clocks are not available the following approaches are used.

Blocked coordinated checkpointing

A straight ~~fow~~ forward approach to coordinated checkpointing is to block communication while the checkpointing protocal executes. After a process takes a local checkpoint, to prevent orphan messages, it remains blocked until the entire checkpointing activity is complete.

The coordinator takes a checkpoint and sends an acknowledgement message to the coordinator. After the coordinator recieves acknowledgments from all the processes, it broadcasts a commit message. a process removes the old permanent checkpoint. A problem with this approach is that computation is blocked during the checkpointing.

Non blocking checkpoint coordination

In this approach the process need not stop their execution while taking checkpoints. A fundamental problem is coordinated checkpointing is to prevent a process from receiving messages that could make the checkpoint inconsistent. The checkpoint indices are piggbacked to avoid blocking.

---

12 i) No, the run is not a linearization of events. One of the reason is that the event e_2^4 (effect) is before e_3^3 event (cause) which violated the happen before relation. Linearization is a sequence of events that is consistent with happen before relation. $e_3^3 \rightarrow e_2^4$ is in a happen before relation and so if it were a linearization run e_2^4 would have been before e_3^3.

(ii) No, For a cut to be consistent, if e_i is in the cut and e_j happens before e_i, then ~~e_{nj}~~ e_j must be in the cut. In this example e_2^4 (recieve event) is in the cut, but the corresponding send event (e_3^2) that happens before it is not the cut.

(iii). Events that happed before e_2^4

e_1^1, e_2^1, e_2^2, e_2^3, e_3^1, e_3^2, e_3^3.

---

13 i) a synchronous execution

a) All the processors are synchronized and the clock drift rate between any two processes is bounded.

b) message delivery (transmission + delivery) times are such that they occur in one logical step or round

c) there is known upperbound on the time taken by a process to execute a step

(ii) Asynchronous execution

(ii) There is no proper synchrony as in synchronous execution

b) There is bound on the drift rate of the processor clocks

c) Message delays (transmission + ~~delivery~~ propagation time) are finite but unbounded

d) There is no upper bound on the time taken by the process to execute a step.

(iii) an asynchronous execution that uses synchronous communication.

a) When all the communication between the pairs of process is by using synchronous send and recieve primitives, the resulting order is synchronous order. The send and recieve events of message appears instantaneous.

b) say asynchronous execution:

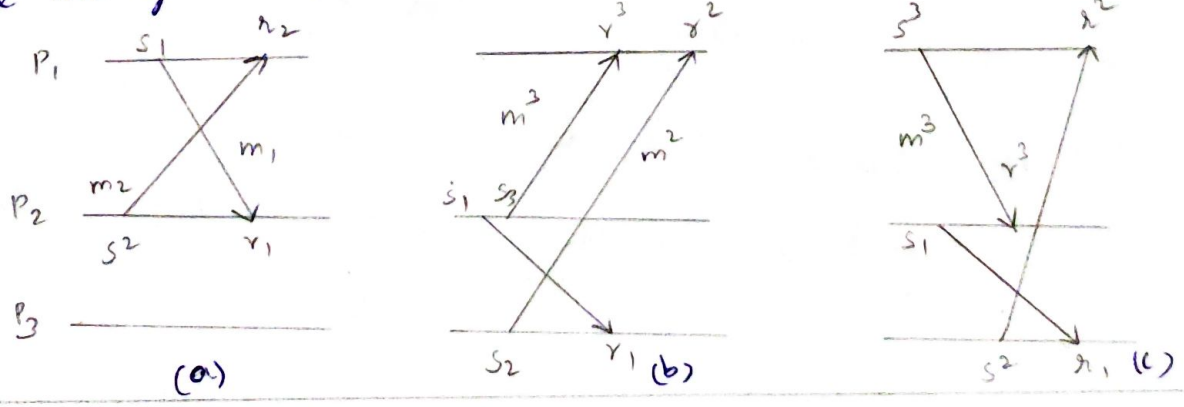
Process i

```
...
send (j)
Recieve (j)
...
```

Process j

```
...
send (i)
Recieve (i)
...
```

c) The timing diagram.



a) asynchronous execution with crown ~~size~~

a) size2 b) size 2 c) of size 3.

(iv) A synchronous system is a system that satisfies the following properties.

Synchronous system

* There is known upper bound on the message communication delay

* There is known bounded drift rate for the local clock of each processor with respect to real time. The drift rate between two clocks is defined as the rate at which their values diverges.

* There is no upper bound on the time taken by a process to execute a logical step in the execution.

Content addressible Networks (CAN)

21

A content-addressible network (CAN) is essentially an indexing mechanism that maps objects to their location in the network

A Distributed reassignment.

A failure of the node in the network can

in a neighbour holding more than one region if its region cannot be merged with that of the departed or failed node. To prevent the resulting fragmentation and restore 1→1 node to region assignment, there is a background reassignment algorithm that runs periodically.

~~It is a~~ Coceptually it is a binary tree whose roots represents the entire space. An internal node represents a region that existed earlier but is now split into regions represented by its children Nodes. A leaf represents existing node.

When a leaf node fails or departs

a) If its sibling node y is also a leaf, then the regions of x and y are merged and assigned to y. The region corresponding to the parent of x and y becomes a leaf and its assigned to y.

b) If the sibling node y is not a leaf, run a depth-first search in the subtree rooted at y until a pair of sibling leaves (say, z_1 and z_2) is found. Merge the regions of z_1 and z_2 making their parent z a leaf node, assign the merged regions to node z_2 and the

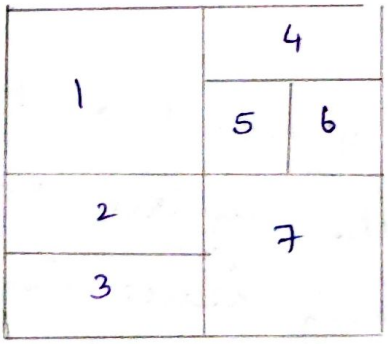
region of x is assigned to zode z1. ~~If node 2 fails~~ Fig: 1 illustrates this A distributed version of this depth-first centralized tree traversal can be done.

a) Identify the highest ordered dimension dim a that has the shortest ~~distance~~ coordinate $[i^{dim\,a}_{min}, i^{dim\,a}_{max}]$

b) Identify neighbour j such that j is assigned the region that was split off from i's region in the last partition along dimension dim a

c) j's region vol = i's region vol, two nodes are sibling and regions are combined.

d) Otherwise j's region vol < i's region vol. Node i forwards a recursive depth-first search request to j.



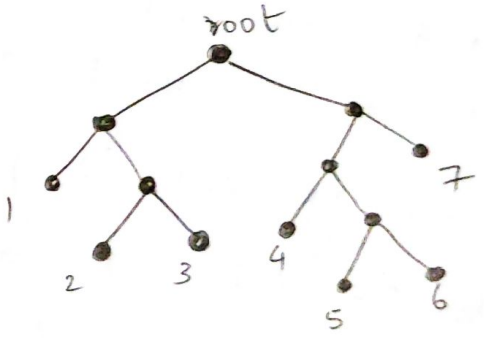


fig 1: region assignment in CAN.

Time Complexity

Overhead of new joiner is O(d) for updating the new neighbours. O(d/4. log(n)) for routing to the appropriate location in the coordinate space The time overhead and overhead in terms of the number of the messages for a node departure is $O(d^2)$ because TAKEOVER protocol uses a message exchange between each pair of neighbours of the departed node.

22. Distributed shared memory consistency models.

A consistency model is essentially a contract between the software and the memory. It says that if the software agrees to obey certain rules, memory promises to work correctly. The models are

strict consistency:

The most stringent consistency model is called strict consistency. It is defined by the following condition "Any read to a memory location x returns the value by the most recent read write operation to x".

sequential consistency.

It is the most intuitive model greatly restricts the use of many performance optimizations commonly used by uniprocessor hardware and compile designers, thereby reducing the benefit of using a multiprocesor. There are two aspects (i) Maintaing program order among operation (ii) Mainting a single sequential order among operations from all processors.

Casual consistency.

The casual consistency model represents a weakening of sequential consistency in that it makes a distinction between events that are potentially casually related and those that are not.

Pram consistency and processor consistency.

It is defined by the following condition. Writes done by a single process are recieved by all other processes in the order in which they are not issued, but writes from different processes may be seen in a different order by different processes. PRAM is easy to implement. It says that there are no guarantee about the order in which different processes see writes

Weak consistency

i) Access to synchronization variables are sequentially consistent

ii) No access to a synchronization variable is allowed to be performed until all previous writes have been completed everywhere.

iii) No data access (read or write) is allowed to be performed until all previous access to synchronization variables have been performed.

PART-C

23.

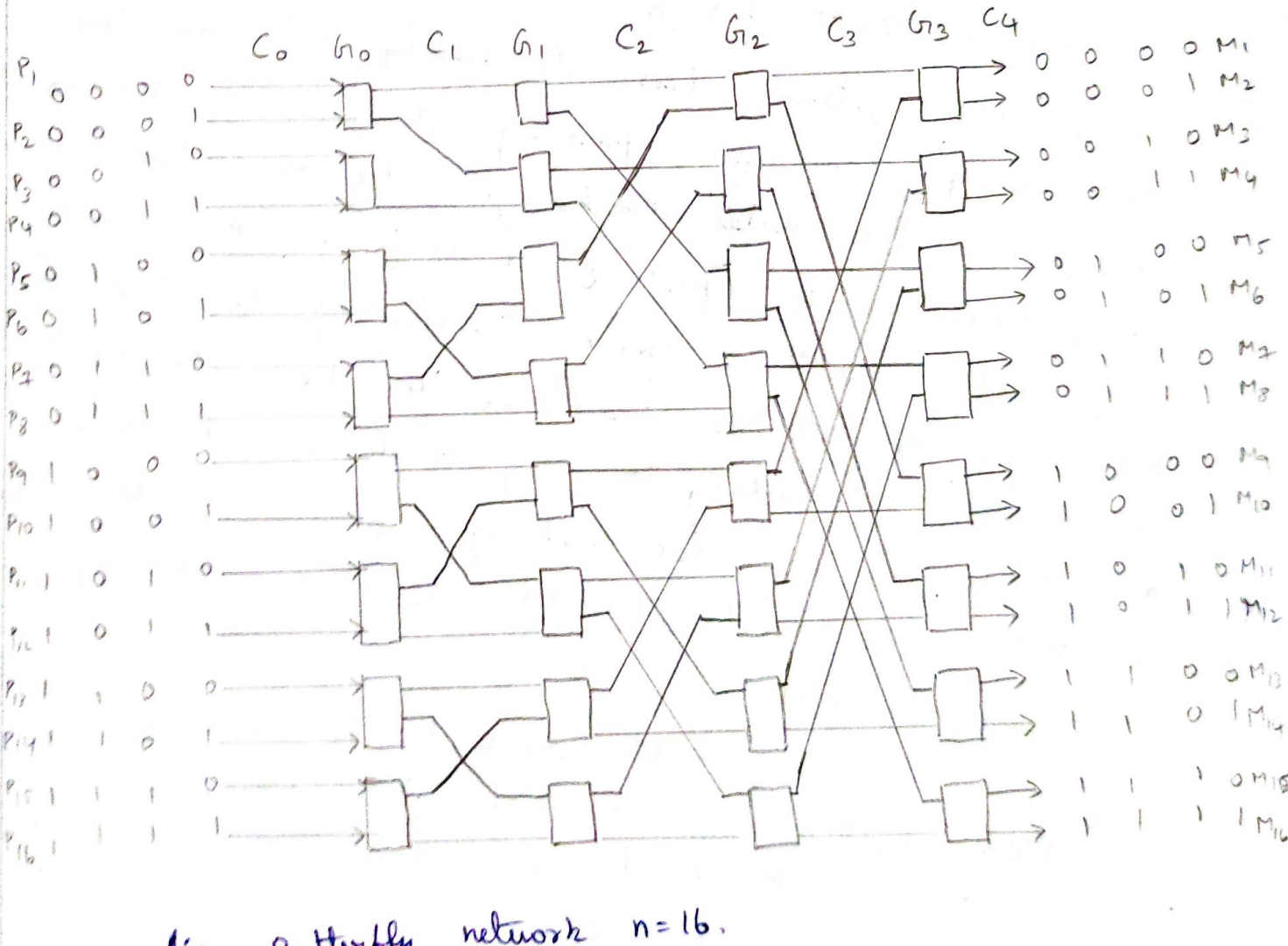


fig Butterfly network n=16.

24. Distributed shared memory (DSM) has become a very popular paradigm in distributed systems Implementation of DSM can be categorized as page based, shared-variable based or object based

We try to design an object based DSM that runs in the user space. The choice of objects as units of granularity over a page or shared variables is because of modularity and flexibility offered by objects.

Issues in DSM design.

Any DSM system has to ~~reply~~ rely on the underlying message passing technique across the network for data exchange between two computers. The DSM has to present a uniform global view of the entire address space (consisting of physical memories of all the machines in the network) to a program executing on any machine. A design of a DSM would involve making following choices

A design of a DSM would involve making following choices

1) Where with respect to the virtual Memory Manager does the DSM operate.
2) What kind of consistency model should the system provide.

3) What should be the granularity of the shared data.
4) What kind of naming scheme has to be used to access remote data.

Granularity.
When a non local memory word is referenced, a chunck of memory containing the word is fetched from its current location and put on the machine making the reference.

structured of shared memory space.
It is the layout of the shared data in memory.
It is normally dependent on the type of application that the DSM system intended to support.

Data location and access
To share data in a DSM system, it should be possible to locate and retrieve the data accessed by a user process.

Replacement strategy
Data block of the local memory must be replaced by a new data block.

Thrashing
Data block migrate between nodes on demand.

PART-A

1. Examples of problems requiring synchronization

i) Physical clock synchronization : Physical clocks diverge in their values due to hardware limitation

ii) leader election - as all process need to agree on which process will be the leader process.

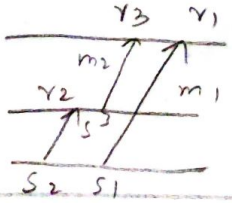
iii) Mutual exclusion : This is clearly a synchroniz -ation problem.

2. A cut is a set of cut events, one per node each of which captures the state of the node on which it occurs.

It is an inconsistent cut since message passes from future to past.

3. A CO execution is an A - execution in which for all (s, r) and $(s', r') \in T$, $(r \sim r'$ and $s \not< s') \Rightarrow r \prec r'$

If two send events s and s' are related by causality ordering, then a causally ordered execution requires that their corresponding recieve events r and r' occur in same order.



e.g CO execution

4. Distributed snap shot use cases
   i) Fault detection - Token Ring Network
   ii) Recovery from incident

5. A site send a REPLY message to requesting site to give its permissions to enter the critical section

6. In distributed systems, the state of the system can be modeled by directed called wait-for-graph (WFG). In a WFG nodes are processes and there is a directed edge from node P_1 to node P_2 if P_1 is blocked and is waiting for P_2 to release some resource.

7. Because, messages induce inter process dependencies during failure free operation. Upon failure of one or more process, these dependencies may force some of the processes that did not fail to roll back, creating a roll back propagation.

8. Formally, the difference the agreement problem and the consensus problem is that, the agreement problem a single process has the initial value, whereas the consensus problem, all processes have an initial value.

9. a) Scalability
b) one way data flow
c) decentralized resources
d) centralized user accounts
e) resource sharing without a dedicated server.

10. a) Scales well with a large number of nodes
b) Message passing is hidden
c) can handle complex and large databases without replication or sending the data to processes
d) provides large virtual space.