Distributed Systems

Notes by Rohan Khurana

Date __/__/____ Page No.: _____

Distributed System:

↳ A system in which components are connected by a communication network, communicates and coordinates their actions by passing msg.

↳ Definition: A Distributed System is defined as the no. of autonomous processing elements (not necessarily homogeneous) that are interconnected by a computer Network, cooperate in processing their assigned tasks transparently.

HOSSCT

Challenges:

1) Heterogeneity (DS Must be constructed from variety of diff. OS, Comp., Hdw., lang.)
2) Openness: (DS must be Extensible, reimplemented in diff. ways)
3) Security: (CIA of Info. Resources, using Encryption)
4) Scalability: (DS remain effective when increase in Resources and Users)
5) Concurrently: (Diff. users accessing shared Resources)
6) Transparency: (Existence is to be transparent to others)

Motivation: Resources Sharing Behind DS

Examples: Internet, Intranet, Finance Trading, Online Games, Mobile & Ubiquitous Computing.

"In a gentle way, you can shake the world." –Mahatma Gandhi

Date __/__/____ Page No.: ____

Limitations of DS:

1) Absence of Global clock
2) Unprecedented Message passing
3) Non-existence of physical Shared Memory.
4) Initial Deployment cost is high

Date ___/___/_____ Page No.: _____

Logical clocks

(1) Logical clock refers to implementing a protocol on all machines within the DS, so that the machines are able to maintain consistent ordering of event within some virtual tc

(2) A logical clock is a mechanism of capturing chronological & ~~casual~~ causal relationship in DS.

Happened Before Relation (→):
$a \to b$, 'a' happended before b.
a, b ∈ same process, a sending msg, b receipient
If $a \to b$, $b \to c$
$\Rightarrow a \to c$ (transitive)

Logical clock

Let Pi is process, Ci is vector clock associated with it
for event a, $C_i(a)$ = TS of event a.

[C1] for any 2 event, a & b in Process Pi if a happend before b ($a \to b$) then.

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

[C2] If a is event of sending msg m in Pi and b is event of receiving msg m at process Pj

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

$C_j = \max(C_j, t_m + d)$

Date ___/___/_____ Page No.: _____

C_i is incremented bfr any 2 successive events in process P_i

[IR1] If $a \to b$ then

$$C_i(b) = C_i(a) + d$$

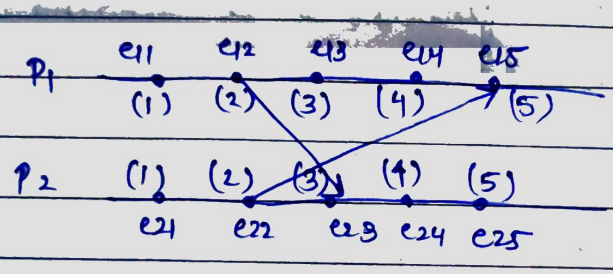
d = drift time (generally 1).

[IR2] If there are more no. of processes & event a is sending of msg m in P_i then T_s $t_m = C_i(a)$

On receiving same msg m in process P_j then

$$C_j = \max(C_j, t_m + d) \qquad d > 0$$

Example



$C_{P_i} = 0$ $d = 1$

$C_{P_2} = 0$

$C_j = \max(C_j, t_m + d)$

Causal ~~toroot~~ ordering

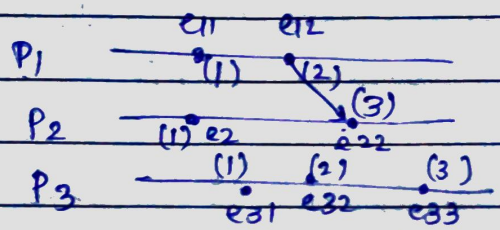
Date

Page No.: _____

Limitations of Lamport's clocks

1) Lamport's system of logical clock states, if $a \rightarrow b$ then $C(a) < C(b)$
However the reverse is not necessarily true if events have occured in diff processes

2) thus if $a \& b$ are events in different processes & $C(a) < C(b)$ then $a \rightarrow b$ is not always true.
eg.



$C(e_{11}) < C(e_{12})$
$C(e_{11}) < C(e_{32})$

e_{11} is causaly related to e_{22} but not e_{32}

3) Lamport clock system

since each clock independently advance due to occurrence of local events in process
LCS cannot distinguish b/w advancement of clocks due to local events from those due to exchange of msg. b/w processes.

∴ By using TS assign by LCS we cannot reason about the causal ordering of 2 events occuring in diff processes.

generates partial ordering
detects causality violation

Page No.:

Vector clock

An algorithm that generates partial ordering of events and detects causality violations. in DS. (causality)

Let n = no. of processes in DS
each process Pi equipped with clock Ci
Ci can be assumed to be a ~~vector~~ function that assign a vector Ci(a) to event a.
Ci(a) TS of event a. at Pi.

Ci[i] → ith entry of Ci, corresponds to Pi's own logical time.
Ci[j] → Pi's best guess of logical time at Pj

[IR1] Clock Ci is incremental b/w 2 successive events in process Pi

$$C_i[i] = C_i[i] + d$$

$C_i[i] = C_i[i] + d$
d = drift time = 1

[IR2] If a is sending msg m by process Pi then msg. m is assigned a vector ts

$$t_m = C_i(a)$$

on receiving msg m by process Pj

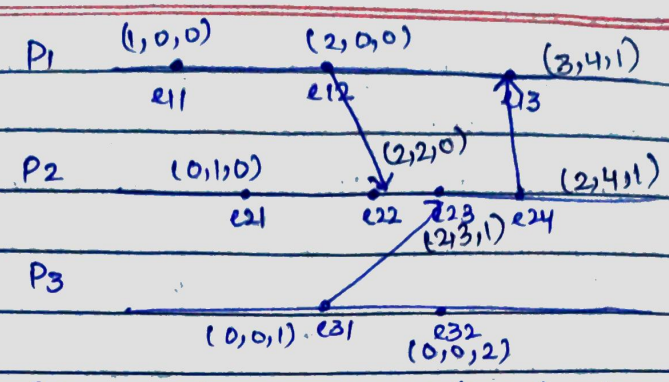
$\forall k$ $$C_j[K] = \max(C_j[K], t_m[K])$$

$\forall K$ $C_j[K] = \max(C_j[K], t_m[K])$

"Facts are facts and will not disappear

Date ___/___/_____ Page No.: _____

eg.



$e_{11} = (ci+d) = (0+1) = 1 \Rightarrow (1,0,0)$ ✓

$e_{12} = (ci+d) = (1+1) = 2 \Rightarrow (2,0,0)$ ✓

$e_{21} = (ci+d) = (0+1) = 1 \Rightarrow (0,1,0)$ ✓

$e(31) = (ci+d) = (0+1) = 1 \Rightarrow (0,0,1)$ ✓

$e(32 = ci+d = 1+1 = 2 \Rightarrow (0,0,2)$ ✓

$e_{22} = \max \begin{bmatrix} 0,2,0 \\ 2,0,0 \end{bmatrix} = (2,2,0)$ ✓

$e_{23} = (ci+d) = \max \begin{bmatrix} (2,3,0) \\ 0,0,1 \end{bmatrix} = (2,3,1)$ ✓

$e_{24} = (ci+d) = (2,4,1)$ ✓

$e_{13} = \max \begin{bmatrix} 3,0,0 \\ 2,4,1 \end{bmatrix} \Rightarrow (3,4,1)$ ✓

Advantages of vector clock over lamport's

1) Events a & b are casually related if $t^a < t^b$ or $t^b < t^a$ else they are concurrent

2) In system of vector clocks

$a \rightarrow b$ iff $t^a < t^b$

Thus, casual ordering can be determined.

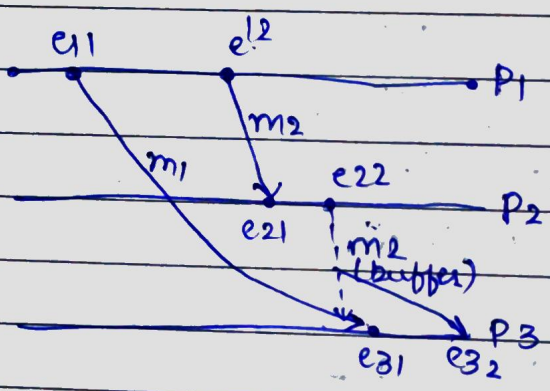
(causal)

Date __/__/____ Page No.: ____

(causal)

Casual ordering of Msg maintaining Deals with the concept of ~~casual~~ (causal) relationship among 'msg sent' event corresponding to 'msg received' event.

If 1) send(m_1)
2) send(m_2)
then every recipient of m_1 & m_2 msg. must receive m_1 before m_2.



Broadcast

Causal Causal Ordering
BSS

Date __/__/____

Page No.: _____

Birman Schiper Stephenson Protocol:

Algo

1) Each process increases its vector clock by 1 upon sending the msg.
2) Msg is delivered to the process if the process has received all the prev. msg. else buffer the msg.
3) Update the vector clock for process.

References:

Pi = process

eij = i → process no, j → event no.

Vi = V.C associated with Pi

Vi[j] = jth element.

Protocol:

1) Pi sending msg. to Pj

Pi increments Vi[i] and sets timestamp

tm = Vi[i] for m.

2) Pj receiving msg. from Pi

a) When Pj, j != i receives m with ts tm, it delays msg. delivery until both are as follows.

(1) Vj[i] = tm[i] - 1

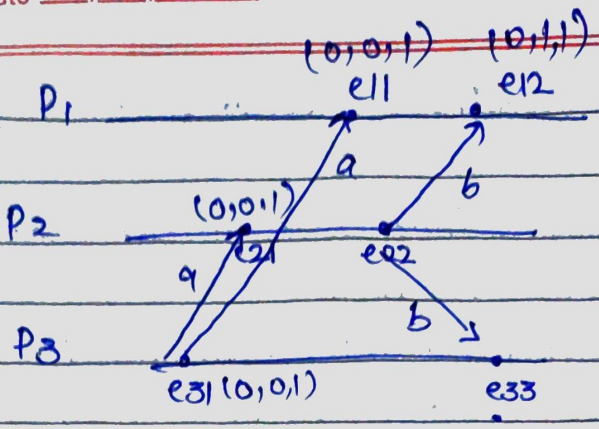
(2) ∀ K <= n && K != i Vj[K] <= tm[K]

b) When msg. is delivered to Pj, update Pj's VC.

c) Check buffer if anything can be delivered.

Date __/__/____ Page No.: ____

eg.



e31 ⇒ P3 sends msg a

$e_3 = (0,0,1)$

$t^a = (0,0,1)$

e21 ⇒ P2 receive msg a

as $c_2 = (0,0,0)$

$c_2[3] = t^a[3] - 1 = 1 - 1 = 0$

$c_2[1] = t^a[1] = 0$

$c_2[2] = t^a[2] = 0$

msg is accepted & $c_2 = (0,0,1)$

e11 P3 ~~send~~ receive msg a as,

$c_1 = (0,0,0)$

$c_1[3] = t^a[3] - 1 = 1 - 1 = 0$

$c_1[1] = t^a[1] = 0$

$c_2[2] = t^a[2] = 0$

msg is accepted & $c_1 = (0,0,1)$

e22 P2 sends msg b

$c_2 = (0,1,1)$ $t^b = (0,1,1)$

e12 P1 receive msg b as $c_1 = (0,0,1)$

$c_1[2] = t^b[2] - 1 = 1 - 1 = 0$

$c_1[1] = t^b[1] = 0$

$c_1[3] = t^b[3] = 0$

msg is accepted $c_1 = (0,1,1)$

Date ___/___/_____ Page No.: _____

e32 P_3 receives msg b as, $c_3 = (0,0,1)$

$c_3[2] = t^b[2] - 1 = 1 - 1 = 0$

$c_3[1] = t^b[1] = 0$

$c_3[2] = t^b[2] = 0$

So, msg is accepted & c_3 is set to $(0,1,1)$

A(i) **Transitless Global state**

A Global state is transitless iff

$\forall i \forall j : 1 \leq i,j \leq n :: transit(LS_i, LS_j) = \phi$

Thus all communication channels are empty in transitless Global state.

A(ii) **Strongly Consistent Global state:**

(Consistent + transitless)

(i) All channels are empty.

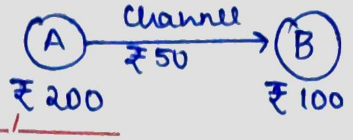
(ii) Send events of all recorded received events are recorded, receive events of all recorded send events are recorded.

(A) **Global state** It is a set of local state of all ~~local~~ individual processes involved in computation & the state of comm. channels.

Need

1) Distributed Deadlock Detection.
2) Termination Detection.
3) Checkpoint

consistent Global state



Date ___/___/______ Page No.: ______

Chandy Lamport's Global State Recording Algorithm

1) Model to capture a consistent Global state
2) It uses a control msg called Marker whose role in a FIFO system is to seperate msgs in channel
3) After a site has recorded its local state it sends marker, along all its outgoing channel before sending more msgs.
4) Marker seperates msg in the channel

Algorithm:

1) Marker sending rule process P:

a) P process records its state

b) for each outgoing channel C from P on which marker has not been already sent, P sends marker along C before sending more msgs along C.

2) Marker receiving rule process Q:

On receiving marker along channel C

if (Q hasn't recorded its state)

↳ record its state

↳ record the state of C as empty set

↳ follow marker sending rule.

else

record the state of C as the set of msg received along C after Q's state was recorded & before Q received the marker along C.

process → Active / Idle

Date ___/___/_____

problem comes when DC terminates implicitly

Page No.: _____

Termination Detection

Detecting whether an ongoing distributed computation has finished all its activities.

example of use of Coherent view of DS.

Notations:

1. B(DW): Computation msg sent as part of comp. DW is weight assigned to it.
2. C(DW): Control msg sent from processes to CA. DW is weight assigned to it.

Huang's Termination Algo:

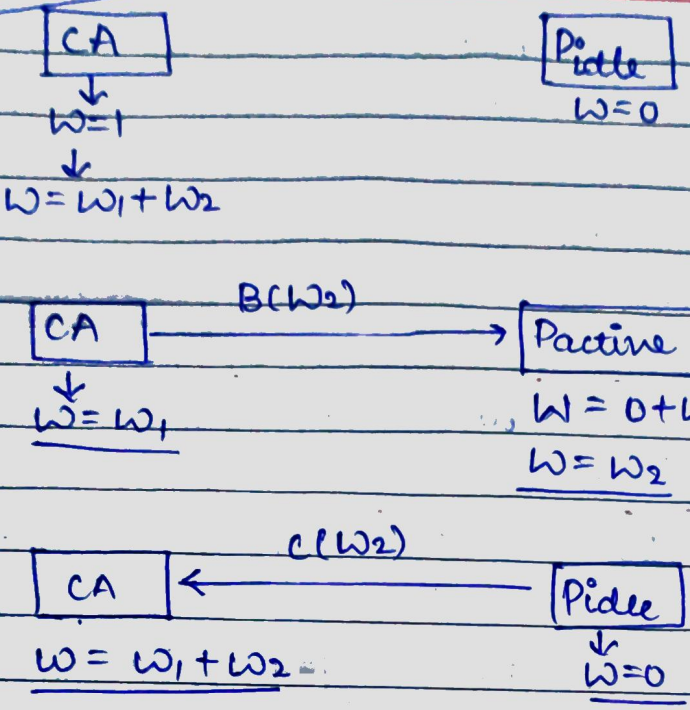
1) **Rule to send B(DW):**
   - ↳ CA having weight W may send a
   - ↳ computation msg to process P.
   - ↳ Split the weight, $W = W_1 + W_2$, $W_1 > 0$, $W_2 > 0$
   - ↳ Set weight of P, $W = W_1$
   - ↳ Send $B(W_2)$ to process Q, $DW = W_2$

2) **On receiving B(DW) by process Q:**
   - ↳ Add DW to weight of process Q, $W = W + DW$
   - ↳ If Q was idle it becomes active on receiving B(DW).

3) **Rule to send C(DW):**
   - ↳ Any active process with weight W becomes idle by sending C(W) to CA.
   - ↳ send control msg C(W), $DW = W$.
   - ↳ set weight of proces as 0. $\boxed{W = 0.}$

Date ___/___/_____ Page No.:

Basic Implementation



4) On Receiving C(DW) by CA:

↳ Add weight through control msg to

↳ wt of CA, i.e. W = W + DW

If After adding, CA weight W = 0 ⇒ process computation terminated

Date __/__/__ Page No.: _____

Mutual Exclusion

↳ Program obj. that prevents simultaneous access to a shared resource.

↳ This concept is used in concurrent programming with a critical section. (A piece of code in which process access shared resource)

Requirement

1) freedom from Deadlock → useless wait due to circular wait Relationship
2) freedom from Starvation → unbounded waiting due to order of service policy
3) Fairness → requests are not served in order they are made
4) Fault Tolerance → algo breaks if process die/ msg are lost / garbled

Token based Algo	Non-Token Based Algo
1) Token is shared among all the sites.	1) Central site will communicate with all site
2) Token contains seq. of No. of site inorder to req. ~~execute~~ CS.	2) It use TS value in order to request for CS.
3) A site having token can only enter CS.	3) A site with smaller TS will enter the CS.
4) Example ↳ Lamport's ↳ Rickart Agarwala ↳ Maekawa's	4) Example: ↳ Raymond Tree ↳ Suzuki-Kasami ↳ Singhal's Heuristic
5) req. are executed exactly in Order as they made in	5) No surety of exe order
6) scalable	6) less scalable
7) Easy Authentication	

Non token

"In a gentle way, you can shake the world." –Mahatma Gandhi

Date __/__/____ Page No.: ____

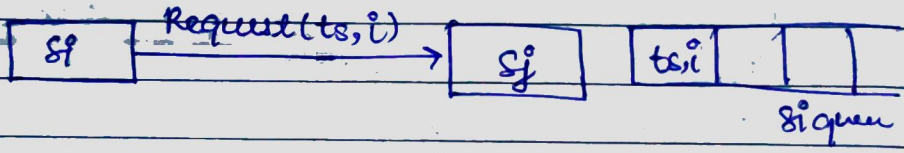
Lamport's Non-Token Algorithm

Requesting set

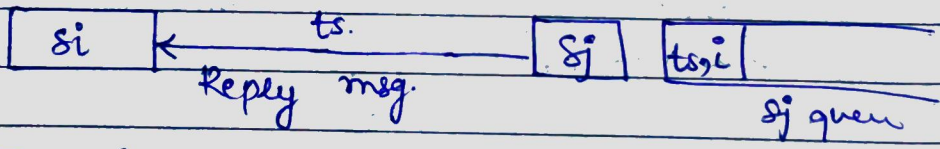
$R_i = \{S_1, S_2 \ldots S_n\}$, every site has queue.

1) **Requesting the CS:**

a) S_i wants to enter the CS, it send Request (ts, i) msg to all the sites in R_i & places request in req-queue$_i$



b) On receiving Request (ts, i) msg by S_j it returns a timestamp reply msg to S_i & places S_i's req. in req-queue$_j$.

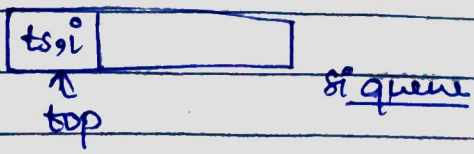


2) **Executing the CS:**

S_i enters in CS if

(i) S_i has received msg with $(ts_j) > (ts, i)$ from all sites. AND

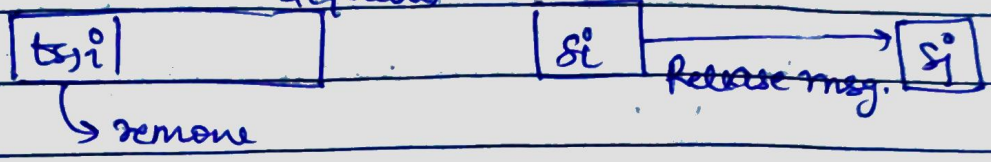
(ii) S_i's request is at top of requestqueue$_i$.



Date ___/___/_____ Page No.: _____

3) Releasing the CS:

a) On exiting Si remove its request from top of RQ & send the ts Release msg. to all sites in Si



b) On receiving the release msg Sj removes Si's request msg. from its RQ.

Performance: Require 3(N-1) msg / CS. invocation

1) (N-1) Request
(N-1) Reply
(N-1) Release

2) Synchronization Delay = T

3) Can be optimized to 2(N-1) msgs / CS. by suppressing Reply msgs in certain situation

Drawbacks:

1) Unreliable Approach:
failure of any process will halt the progress of entire system.

2) High Message Complexity: 3(N-1) / CS.

Date __/__/____ Page No.: ____

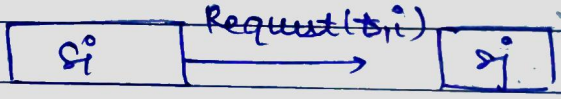
Ricart-Agawala Non-Token Algorithm

Optimization of Lamport's Algo, it dispenses with release msg by merging them with Reply msgs.

Request set $R_i = \{s_1, s_2 \dots s_n\}$
every site has queue.

1) **Requesting the CS:**

a) S_i wants to enter the CS, it send ts_i Request to all sites



b) When S_j receives request msg from S_i, it sends a reply msg to S_i,
if S_j is neither requesting nor executing the CS
or if S_j is requesting but $(ts,i) < (ts,j)$
else request is defered.

2) **Executing the CS:**
S_i enter the CS after receiving reply msg from all site in R_i.

3) **Releasing the CS:**
S_i exits the CS, it sends reply msg to all the defered requests.

Date ___/___/_____ Page No.: _____

Performance :

1) 2(N-1) msgs / CS execution
(N-1) Request msgs
(N-1) Reply msgs.

2) Synchronization Delay T.

Drawback :

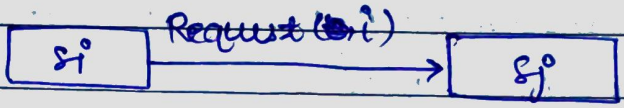
Unreliable approach.
failure of any one node in system can halt the progress of system. In this situation process will starve.

Maekawa's Algorithms

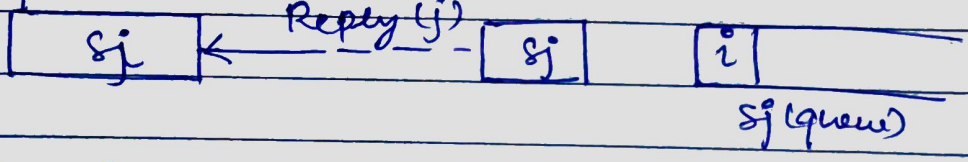
Requesting set, $R_i = \{S_1, S_2 \ldots S_n\}$

1) **Requesting the CS:**

a) S_i wants to enter the CS, it sends request msg to all the site in R_i



b) When S_j receives the request msg from S_i it will sent a reply msg to S_i only if S_j hasn't sent a reply msg to a site from the time it received the last Release msg. else it queues up the request



2) **Executing the CS:**

S_i enters the CS only after receiving reply msg from all the sites in R_i.

3) **Releasing the CS:**

a) S_i exits the CS & sends Release(i) msg to all the sites in R_i

b) On receiving Release(i), S_j send reply to next site waiting in queue & delete that entry from queue.

Date __/__/____ Page No.: _____

c) If queue is empty, Sj update its states to show that it hasn't sent any reply msg since the receipt of last Release msg.

Performance:

1) $3\sqrt{N}$ msgs per CS
$\sqrt{N}$ request msgs.
$\sqrt{N}$ reply msgs
$\sqrt{N}$ release msgs.

2) Synchronization Delay = $2T$ (msg. propagation Delay).

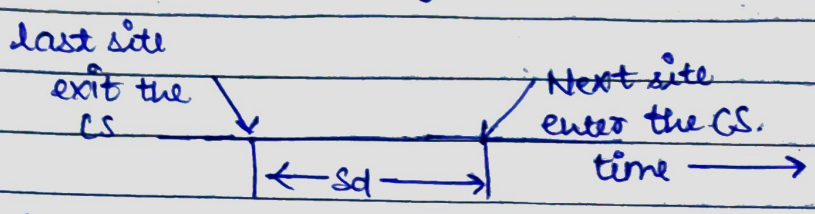
Drawback:

Deadlock prone → A site is exclusively locked by other sites & request are not prioritized by the timestamp.

Date ___/___/_____ Page No.: _____

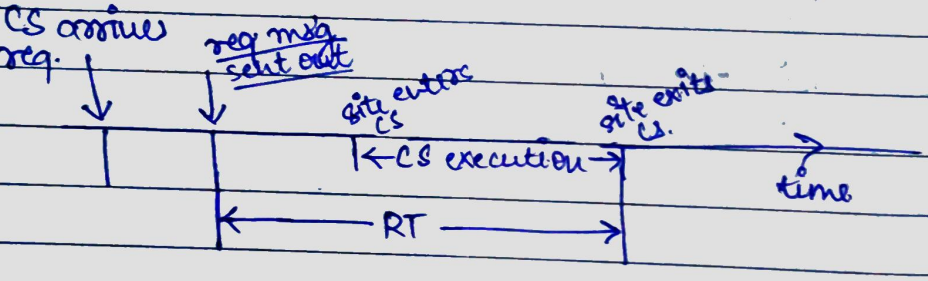
Performance Metric

1) No. of msgs sent per CS:
2) Synchronization Delay



Time between 2 consecutive CS:

3) Response Time (RT):
Time between request msg sent out and completion of Critical section



4) System throughput: Rate at which system executes for CS.

Sd = Synchronization delay
E = Avg. CS execution time

$$\text{System throughput} = \frac{1}{Sd + E}$$

ll → light load
hl → heavy load

Date / / Page No.:

	Algorithm	RT(ll)	Sd	Msg.(ll)	Msg(hl)
1)	Lamport	2T+E	T	3(N-1)	3(N-1)
2)	Ricart Agarwala	2T+E	T	2(N-1)	2(N-1)
3)	Maekawa	2T+E	2T	$3\sqrt{N}$	$5\sqrt{N}$
4)	Suzuki Kasami	2T+E	T	N	N
5)	Singhal Heuristic	2T+E	T	N/2	N
6)	Raymond Tree	TlogN+E	TlogN/2	logN	4

Remember

Chandy-Lamport's Global state (Marker)
Casual Ordering Birman schiper stephenson
Termination Detection - Haang's Termination CA
Lamport's logical clock Cj = max(Cj, tm+1) Rules
Vector clock Cj[k] = max(Cj[k], tm[k]) ∀ k

Rohan

Date 3/03/

Page No.:

Goals of DS

1) Resource sharing
2) Improved system performance
3) Improve Reliability & Availability.
4) Modular ~~complexity~~. (expandability)

Web challenges:

1) Naming
2) Security
3) Availability
4) Access control.
5) Performance management

System Models: Common properties & design choices for DS in singer descriptive Model.

(i) ~~Architectural~~ (ii) Fundamental.

Casual Ordering Algo:

1) Birman-Schiper-Stephenson.
2) Schiper-Eggli-Sandoz

Threads over processes

↳ Created & Destroyed much faster.
↳ faster to switch b/w threads.
↳ Threads share data easily.

"Facts are facts and will not disappear on account of your likes."--Jawaharlal Nehru

Date ___/___/_____ Page No.: _____

Proxy Server:

↳ A server that acts as a gateway between user's computer & Internet.

↳ It verifies and fwd incoming client req. to other servers for further communication.

↳ Enhance privacy.

↳ eg. Req. made from Proxy server may help to hide client's IP from web server.

Reason Middleware moved from distributed objects to distributed components.

1. Implicit Dependencies
2. Interaction with Middleware.
3. Lack of seperation of distributed Concerns
4. No support for deployment.

Consistent Cut:

A cut is consistent if for each event that it contains, it also include all events casually ordered before it.

Let a, b be 2 events in DS

$$(a \in \text{consistent cut } C) \wedge (B < a) \Rightarrow b \in C$$

Date ___/___/_____ Page No.: _____

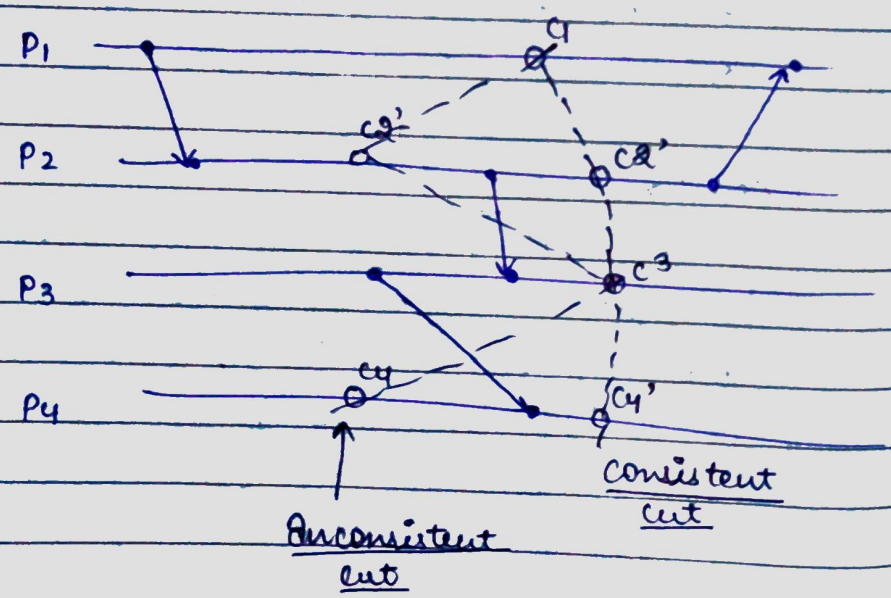
Inconsistent cut:

A cut C is inconsistent iff any of the following conditions are true.

1) $e \rightarrow$ sending event of msg.
   $e' \rightarrow$ receiving event of msg.
   $\boxed{e' \in C \wedge e \notin C}$

2) e and e' are on events and
   $\boxed{e \leftrightarrow e' \wedge e' \in C \wedge e \notin C}$

3) e and e' are off events and
   $\boxed{e \leftrightarrow e' \wedge e' \in C \wedge e \notin C}$



Date ___/___/_____ Page No.:

Architectural Model

in terms of computation & communication

various sys & h/w architecture exits that are usally used for distributed computing.

low level	high level
Interconnection of Multiple CPUs	Interconnection of processes running on that CPUs.

a) Client Server Model. → Dependency
b) 3-Tier Architecture → No dependency
c) Peer to Peer Model. c) tightly coupled (control)
d) Proxy server / cache.

Fundamental Model: Deals with communication that can affected by delays, failure & security attacks.

a) Interaction Model
   - ↳ performance of comm. channel.
   - ↳ sync. & async
b) Failure Model:
   - ↳ Masking, Integrity & Validity approach
   - ↳ Specification of faults.
c) Security Model:
   - ↳ Identify possible threats to proces comm.

Date __/__/____ Page No.: ____

Agreement Protocols

↳ process of sending and reaching the agreement to all sites

↳ AP are useful for error free communication among various sites.

System Model: (where agreement protocols are used)

(i) If there are n processors in DS then only m processors out of them maybe found as faulty processors.

(ii) every processor in the system is free to communicate with each other in the system due to their logical connections with each other

(iii) A receiver processor always knows the identity of sender processor of msg.

(iv) The communication medium is reliable and only processors are prone to failures.

Classification Of Agreement Problems

1) Byzantine Agreement Problem:

↳ A single value is to be agreed upon.

↳ Agreed value is initialized by an arbitrary processor and all non-faulty processors have to agree on that value.

2) Consensus Problem:

↳ every processor has its own initial value and all non faulty processor must agree on a single common value.

3) Interactive Consistency Problem:

↳ every processor has its own initial value and all non faulty processors must agree on a set of common values.

Byzantine Agreement Problem

An arbitrary chosen preprocessor (source processor) broadcasts its value to Others

A solution to BAP should meet the following objectives:

(i) Agreement: All non-faulty processors agree on the same value.

(ii) Validity: If source is nonfaulty, then the common agreed value must be the value supplied by the source processor. If source is faulty then all non-faulty processors can agree on any common value.

(iii) Termination: each non-faulty processor must eventually decide on a value.

Date ___/___/_____ Page No.: _____

Consensus Problem

Every processor broadcasts its initial value to other processors
Initial value may be different for different processors.

Protocols must meet following objectives:

(i) Agreement: All Non-faulty processors agree on the same single value.

(ii) Validity: ~~All~~ NFP
If the initial value of every NFP is v then the agreed upon common value by all NFP must be v.

(iii) Termination: Each NFP must eventually decide a value.

Date __/__/____ Page No.: ____

Iterative Consistency Problem

Every processor broadcasts its initial value to all other processors.
Initial value must be different for different processors

Protocols must meet the following objectives:

(i) Agreement: All NFP agree on the same vector $(v_1, v_2, v_3 \ldots v_n)$

(ii) Validity: If ith processor is NF and its initial value is v_i then the ith value agreed by all NFP must be v_i.

(iii) Termination: Each NFP must eventually decide on different value of vectors

Date ___/___/_____ Page No.: _____

Lamport's Shostak Pease Algorithm:

Lamport's Shostak Pease Algorithm:

One Message Algo OM(m) $m>0$ solves BAP for $3m+1$ processors in the presence of atmost m faulty processors

Let n = total no. of processors ($n \geq 3m+1$)

↳ **Algorithm OM(0):**

1) The source processor sends its value to every processor.
2) Each processor uses the value it receives from the source. If it receives no value, default value = 0

Algorithm OM(m):

1) The source processor sends its value to every processor
2) for each i, let v_i = value processor i receives from the source.
3) Processor i acts as the new source and initiates algo OM(m-1) wherein, it sends the value v_i to each of other $n-2$ processors.
4) for each i and j ($\neq i$), let v_j (value processor i received from processor j) in step (3) processor i uses the value majority($v_1, v_2 \ldots v_{n-1}$)

The majority function ($v_1, v_2 \ldots v_{n-1}$) computes the majority value if it exists else it uses default value '0'

"In a gentle way, you can shake the world." –Mahatma Gandhi

Date __/__/____ Page No.: ____

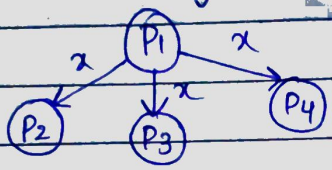
BA cannot always be reached among 4 processors if 2 processors are faulty.

Proof:

4 processors : P_1, P_2, P_3, P_4.

P_1 initiates the initial value.

P_2, P_4 are faulty.

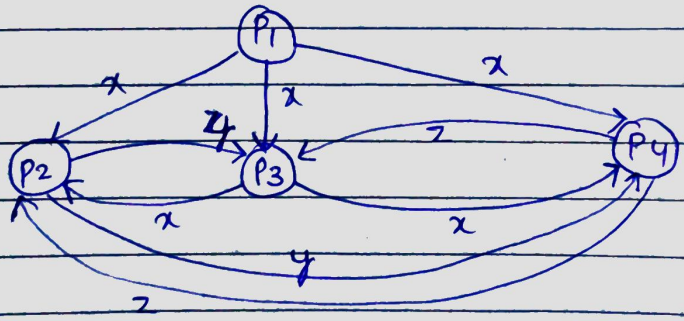


P_1 executes OM(1) and sends its value x to other processors.

After reciving the value x from P_1 P_2, P_3, P_4 executes OM(0).

P_3 is non faulty & sends x to p_2 & P_4

P_2 & P_4 are faulty sends y to P_3, P_4 and z to P_2, P_3 resp.



Majority value for Byzantine solution:

processor	Recived Majority	common
P_2	(x, x, z)	x
P_3	(x, y, z)	0
P_4	(x, x, y)	x

According to Majority value table, processor doesn't agree on single common majority value, which violates the condition of BAP.

Applications of Agreement Protocols:

1) Fault-Tolerance Clock synchronization:
   - ↳ DS require physical clocks to synch.
   - ↳ Physical clocks have drift problems.
   - ↳ AP may help to reach a common clock value.

2) Atomic Commit in DDBS:
   - ↳ DDBS sites must agree whether to commit or abort the transactions.
   - ↳ AP may help to reach a consensus.

Application of Agreement Protocol:

1) fault-Tolerance clock synchronization
2) Atomic Commit in DDBMS.

Distributed file system (DFS)

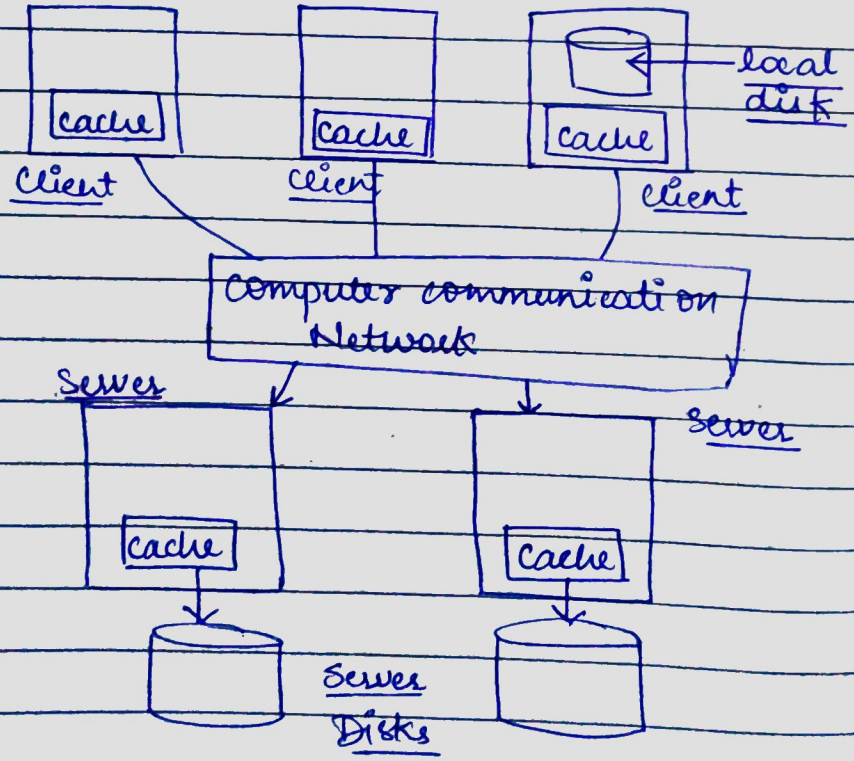
↳ DFS is a resource management component of Distributed Operating System.

↳ It implements a common file system that can be shared by all atonomous computer in system.

Goals → ① Network Transparency

→ ② High Availability

Architecture:



Date ___/___/_____ Page No.: _____

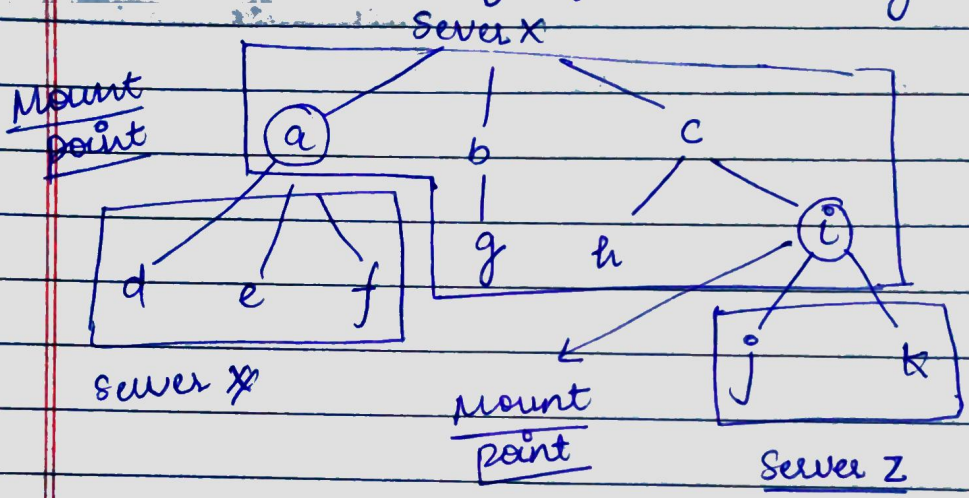
(1) Name Server:
A process that maps names specified by the clients to stored objects such as files & directories.

(2) Cache Manager: Implements file caching
The mapping occurs when process references a file/directory for the first time.

Date __/__/____ Page No.: ____

Mechanisms for Building DFS:

(1) **Mounting:** It means assigning the root directory of the new file system to a subdirectory of root directory



(2) **Caching:**

↳ Improves file system performance by exploiting the locality of reference

↳ When client references a remote file, the file is cached in Main Memory of server and at client side

↳ When multiple clients modify shared data, cache consistency becomes problem

↳ It is difficult to implement a solution that guarantees consistency

Date __/__/____ Page No.: ____

3) Hints: Timestamp + cache — Caching without cache coherence

↳ Treat cached data as hints:
ie cached data maynot be completely accurate.

↳ can be used by application that can disover that the cached data is invalid and can recover

4) Bulk Data Transfer:

↳ Overhead introduced by protocols doesn't depend on amt of data transferred in one transaction.

↳ Most files are accessed in their entirety

↳ When client req one block of data, multiple consecutive blocks are transferred

5) Encryption:

↳ Needed to provide security in DS.

↳ Entities that wed to communicate send req to authenication server.

↳ Authentication server provide key for conversation.

NCW CASS Date __/__/____ Page No.: ____

Design Issues

1) Naming and Name Resolution
A name in file system is associated with an object. Name Resolution is the process of mapping a name to an object or in case of replication to multiple objects.

2) Cache on Disk or Main Memory
Whether the data cached by a client should be in main memory at the client or on a local disk at the client.

3) Writing Policy:
It decides when a modified cache block at client side should be transferred to server.

4) Cache consistency:

5) Availability:
a) Replication b) Unit of Replications
c) Replica Management.

6) Scalability: It deals with the suitability of design of system to cater the demands of growing system.

7) Semantics: It characterises effects of accesses on files.

"Facts are facts and will not disappear on account of your likes." – Jawaharlal Nehru

Challenge → higher cost of comm^n

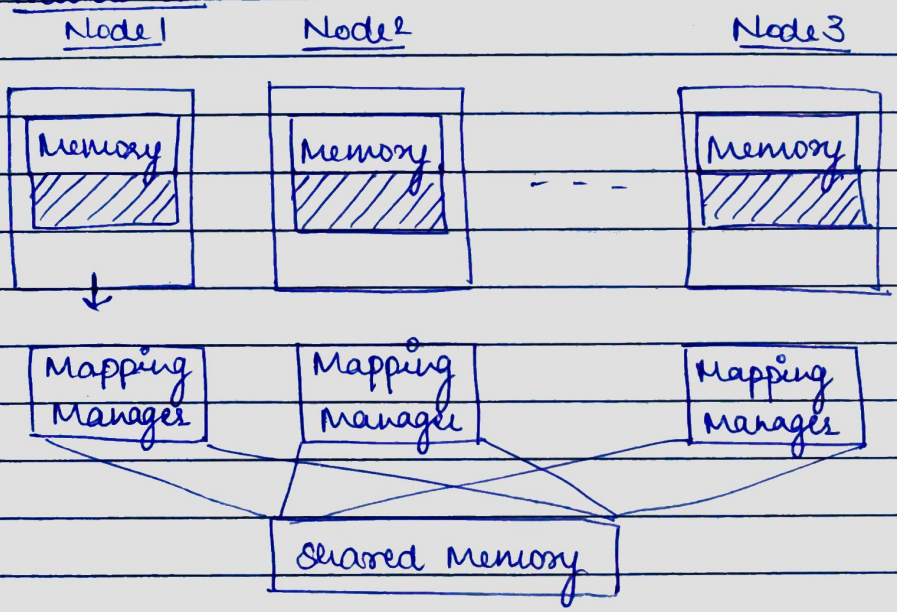
Date __/__/____ Page No.: _____

Distributed Shared Memory

↳ DSM implements shared memory model in DS, which has no physical shared memory.

↳ Provides virtual address space shared bpw all nodes

Architecture:



Design Issues:

1) Granularity:

↳ size of shared memory unit.

Large pg size, more locality, less overhead comm^n more contention more false sharing

2) Page Replacement:

Replacement algo must take into acc pg. access modes.

↳ shared, private, readonly writable.

LRO. private pg. replaced before shared ones, private pg. swap to disk Shared pg sent over network to owner, Read only may be discarded

"In a gentle way, you can shake the world." - Mahatma Gandhi

SGPWR

Date __/__/____ Page No.: ____

Memory Coherence

① Replicated shared data obj.
② concurrent access to data obj at many nodes

① Sequential Consistency: → result of exe. of opr. of all pro is same as if they are executed seq.
② General Consistency: → All copies of Mem loc. contain same data
③ processor Consistency:
→ opr. issued by processor are performed in order they are issued.
④ Weak consistency:
Memory is consistent only after sync. opr.
⑤ Release consistency:
→ Sync. opr. must be consistent.

Coherence Protocols:

① Write-invalidate Protocol.
A write to shared data causes the invalidation of all copies except one before the write

② Write update protocol.
A write to a shared data causes all the copies of that data to be updated.

Date ___/___/_____ Page No.: _____

Algo to implement DSM

Central Server Algorithms:

↳ A central server maintains all the shared data

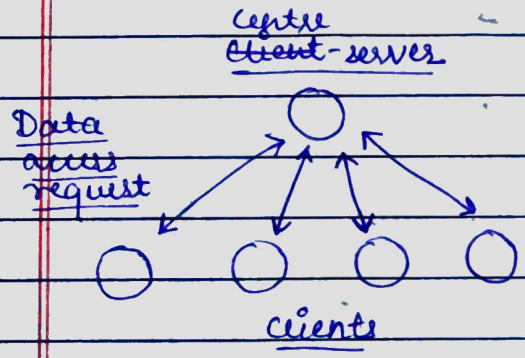
Read Request : return data item

Write Request : update data & return ack. msg.

↳ A timeout is used to resend a request if ack. fails

↳ Duplicate write req. can be detected by associated seq. no. with write requests.

↳ If an application's req. to access shared data fails repeatedly, a failure condition is sent to the application.



Clients	Central Server
① Send data request	
	① Receive Request
	② perform data access
	③ Send response
④ Receive response	

Date __/__/____ Page No.: ____

Migration Algorithm

↳ In migration algo, data is migrated to the loc of data access req, allowing subsequent accesses to data to be performed locally

↳ Allows only one node to access shared data at a time.

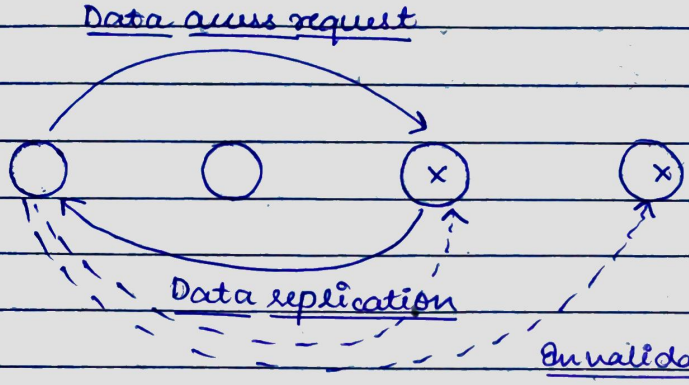
↳ Keeping track of memory location (location service, home machine for each page, broadcast)

Client	Server
① If block not local	
② Determine location	
③ Send request	
	① Receive request
	② Send location of block
① Receive location	
② Access data block.	

Date ___/___/_____ Page No.: _____

Read Replication Algorithm

1) Allows Multiple nodes to have read access and one node to have write access
2) A write to a copy causes all copies of the data to be updated or invalidated
3) All locations must be kept track of : location service / home Machine



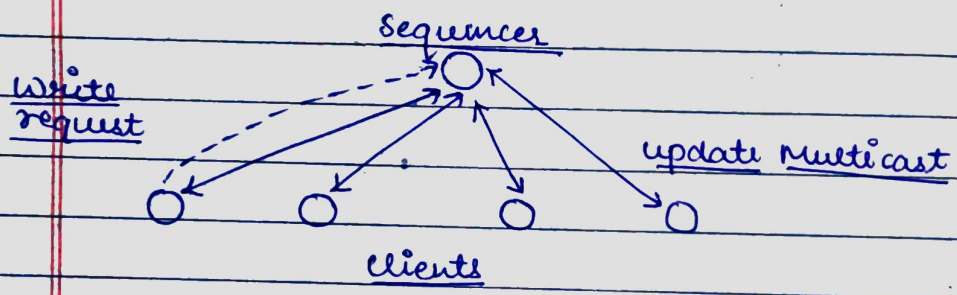
Clients	Remote Host
① If block not local	
② determine location	
③ send request. →	
	① Receive Request
	② send block
← ① Receive block	
② Multicast Invalidate →	
	① Receive Invalidate
	② Invalidate Block
① Access data	

Date ___/___/_____ Page No.: _____

Full Replication Algorithm

1) Allow multiple read/write concurrently.
2) Must control access to Shared Memory
3) Need to maintain consistency

We gap-free sequencer, all nodes wishing to modify shared data will send the modification to sequencer which Multicast the sequences.



Clients	Sequencer	Hosts
① If write send data		
	① receive data ② add Sequence No. ③ Multicast	
① receive ack ② update local memory		① receive data ② update local memory

Backward Error Recovery
↳ operation Based
↳ State - Based

Page No.

Failure Recovery

Process that involves restoring an erroneous state to an error free state

Forward error Recovery:
↳ Remove errors in process/system state if errors can be completely assessed
↳ Continue process/system forward execution

Backward

Advantages:
1) simple to implement
2) can be used as general recovery mechanism

Disadvantages:
1) No guarantee that fault doesnot occur again
2) Some components can't be recovered.
3) perform penalty

Backward Error Recovery
Restore process/system to previous error-free state and restart from there.

forward

Advantages:
1) Less Overhead

Disadvantages:
1) limited to use
2) cannot be used as general mechanism for error recovery

Date / / Page No.:

Application of Agreement protocols

Atomic Transaction
- indivisible
- instantaneous
- uninterrupted

Atomic Commit

Phase 1: Every site executes their part of distributed transactions & broadcasts their decisions (commit / Abort) to all other sites.

Phase 2: Each site decides whether to commit or abort its part of distributed transactions based on what it has received from other sites in first phase.

Advantages of DSM:

1) Cheaper than tightly coupled multiprocessor system.
2) Can form a large physical memory.
3) Easier programming
4) DSM takes advantages of Memory reference locality.

Date ___/___/_____ Page No.: _____

Token Based Algo:

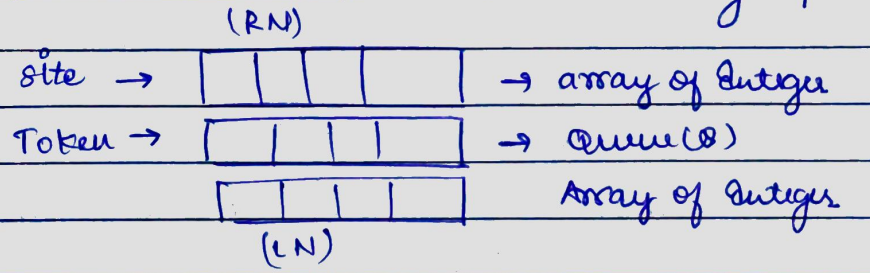
In token based algorithms, a unique token is shared among all sites. If the sites possess token then it is allowed to access the CS.

Suzuki- Kasami Algorithm: (Broadcast)

Design Issues:

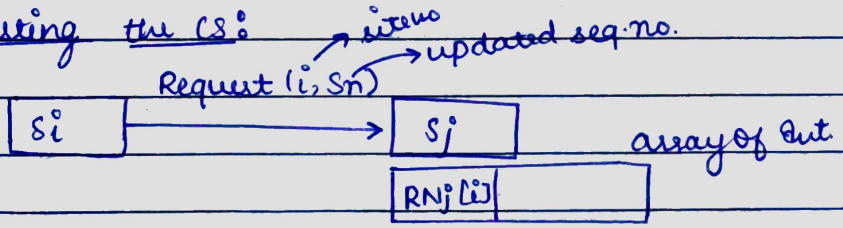
(i) Distinguish outdated Request Msg from Current Request Msg

(ii) Determine which site has outstanding req.



1) Requesting the CS:

(i)



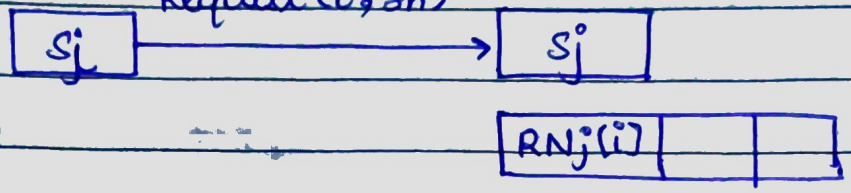
(ii) RNj[i] to max(RNj[i], Sn).
Sj send idle token if RNj[i] = LN[i]+1

2) Executing the CS:
Site Si execute the CS when it received token.

3) Releasing the CS: (i) LN[i] = RNi[i]
(ii) updated ID of site not in token queue if RNj[j] = LN[j]+1
(iii) Delete top site ID from queue

1) Requesting the CS:

(i) Request(i, Sn)



(ii) RNj[i] to max(RNj[i], Sn)

Sj send idle token if

RNj[i] = LN[i] + 1

2) Executing the CS:

Site Si executes the CS when it received token

3) Releasing the CS:

(i) Set LN[i] = RNi[i]

(ii) Updated ID of site not in token only if RN[j] = LN[j] + 1

(iii) Pop queue.

Date ___/___/_____ Page No.: _____

Singual's Heuristic Algorithm

$R = \{S_1 \dots S_n\}$

Data structure:

site → SV[1…N] (state), SN[1…N] (seq. no)

Token → TSV[1…N], TSN[1…N]

A site can be in following states:

R → req. the CS

E → execute the CS.

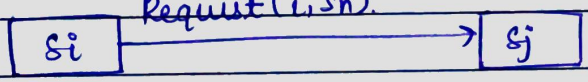
H → Hold the idle token

N → None

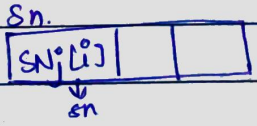
1) **Requesting the CS:**

(i) set $SV_i[i] = R$

(ii) Increment $SN_i[i] = SN_i[i] + 1$

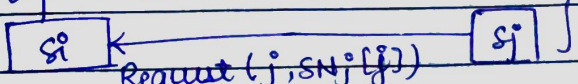


(iii) Discard if $SN_j[i] \geq sn$.



Start of site { j → N, set i → R }

{ j → R, if i ≠ R set i = R



}

{ j → E, set i → R }

{ j → H, set i → R }

Token ⇒ state → R

sequence NO → Sn.

2) **Executing the CS:**

Si execute CS and set SVi[i] = E

3) **Releasing the CS:**

Si → SVi[i] = N , TSV[i] = N

for all sj

If SNi[j] > TSN[j]

then

TSV[j] = SVi[j]

TSN[j] = SNi[j]

else

SVi[j] = TSV[j]

SNi[j] = TSN[j]

If SVi[j] = N set SVi[i] = N

Token.	RT	Sd.	Message (ll)	Message (hl)
Suzuki-Kasami	2T+E	T	N	N
Singhal-Heuristic	2T+E	T	N/2	N
Raymond	T(logN)+E	TlogN/2	log(N)	4

Date __/__/____ Page No.: ____

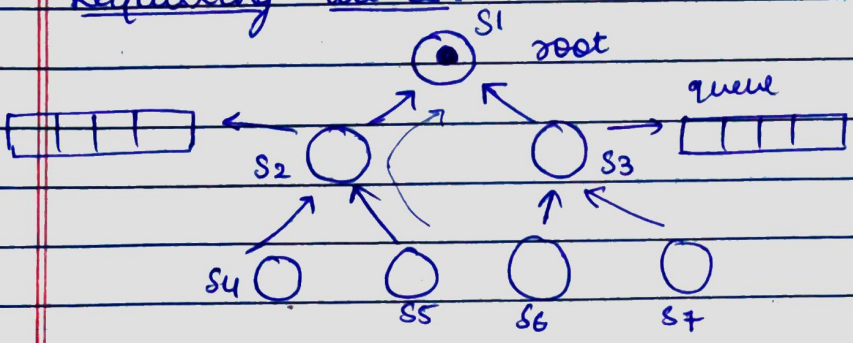
Logical variable holder

Raymond Tree Algorithm

Basic Idea:

1) The site are logically arranged as directed tree.
2) Every site has local variable holder that points to an immediate neighbour node.

1) Requesting the CS:



2) Executing the CS:

A site enter CS when it receives the token & deletes the top entry of its requesting queue.

3) Releasing the CS:

If request queue is nonempty it delete top entry & send token to that site and set holder.

Properties

1) free from deadlock, starvation
2) Average case O(logN) msg complexity O(T logN/2) sd.
3) Worst case → Balanced Binary Tree.

Deadlock

Defined as the permanent blocking of the process ie a set of process is waiting for an event need by some other process.

Necessary conditions

1. Mutual Exclusion
2. Hold + Wait
3. No preemption
4. Circular Wait

Communication Deadlock	Resource Deadlock
(1) Waiting is done for msgs	(1) Waiting is done for resources
(2) A process cannot proceed with the execution until it can communicate with one of the process for which it was waiting.	(2) A process cannot proceed with the execution until it receives all the resources for which it was waiting.
(3) A process can know the identity of those processes on the action of which it depends.	(3) The dependence of one transaction on actions of other transaction is not directly known.
(4) Diagram Can't be prevented by safe state	(4) can be prevented by safe state

Date ___/___/_____ Page No.: _____

Centralized Deadlock Detection

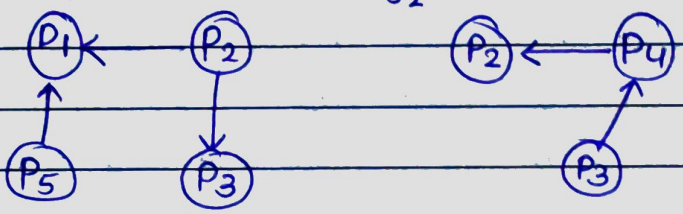
(i) **Complete centralized Algo**

$R = \{S_1, S_2 \text{ --- } S_n\}$

In this a designated site called control site maintain a WFG to check deadlock

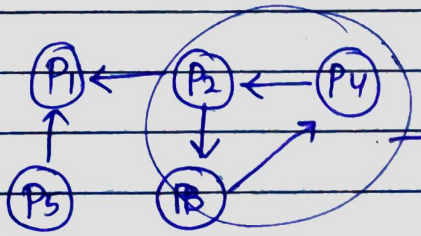
↳ Request Resource
↳ Release Resource
Info

eg. S_1 S_2



local WFG

Control site S_3.



→ cycle ⇒ Deadlock

Disadvantages

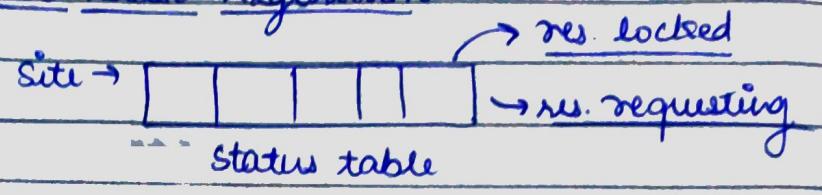
↳ Dependent only on control site.

Date __/__/____ Page No.: ____

(ii) Ho - Ramamoorthy Algorithm

A) Two Phase B) One Phase

A) Two Phase Algorithm



1) every site maintain a status table that contains the status of all process initiated at that site
2) Status includes → all resources locked
→ all resources waited upon.
3) A designated site req. the status table from all sites & constructs a WFG from the info. received & searches it for cycle.
4) If no cycle, system is free from deadlock else, designated site again requests status table from all the sites & again constructs a WFG using only those transactions which are common to both reports.
5) If same cycle is detected, it is deadlocked

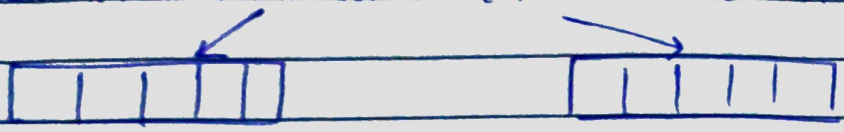
Limitations It may indeed report false deadlock.

Date ___/___/_____ Page No.: _____

B) One Phase Algorithm:

↳ Only one status report from each site

↳ each site maintains 2 status tables



Resource status Table
Keeps track of the transactions that have locked by or waited for by all the transactions at that site

Process status Table
Keeps track of the resources locked by or waited for by all the transactions at that site.

↳ A designated site requests both the tables from every site, constructs WFG using only those transactions for which entry in the resource table matches with process table & serches for the cycle.

↳ If no cycle is detected so no. deadlock

It does not detect false deadlocks b'coz it eliminates the inconsistency in state info by using only the info. that is common to both tables.

⊕ faster & few msgs, ⊖ more storage → 2 tables.

Date ___/___/_____ Page No.: _____

Distributed Deadlock Detection

All the sites cooperate to detect a cycle in the state graph that likely to be distributed.

$R = \{S_1, S_2 - \cdots - S_n\}$.

Obermark's

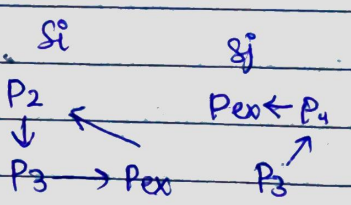
Path-Pushing Algorithm

1) In this algo, info about the waitfor dependencies is propagated in the form of paths
2) In this, each site maintain its local WFG It includes Pex.

$P_i \rightarrow Pex$

$Pex \rightarrow P_j$

eg.



Algorithms:

1) If local WFG contains cycle that doesn't involve Pex then system is in deadlock.
2) If Pex exists suppose

$Pex \rightarrow P_1 \rightarrow P_2 \rightarrow P_3 \rightarrow Pex$.

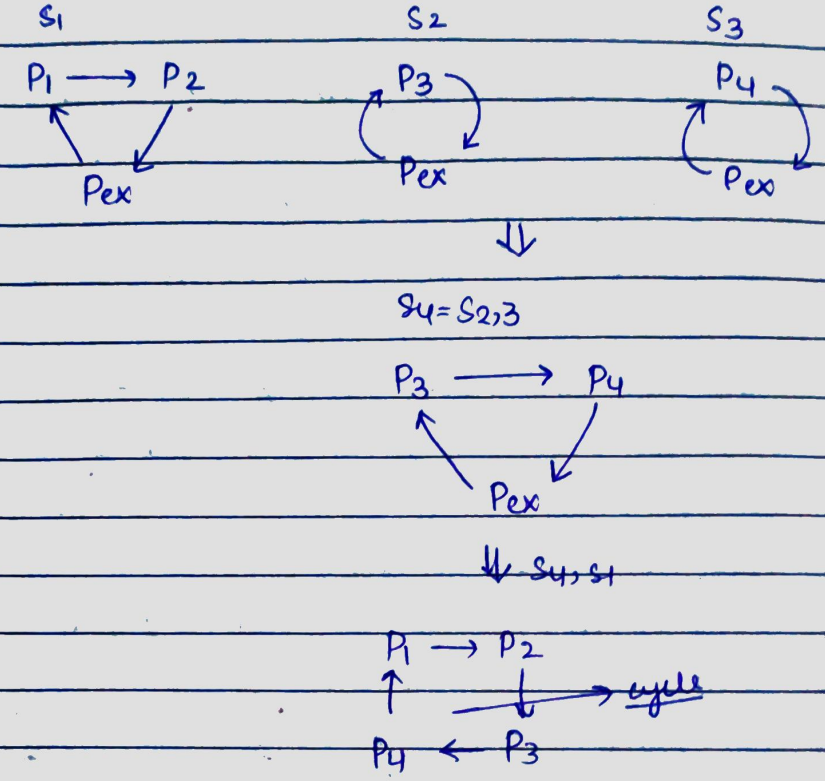
3) Si —msg update ex.→ Sj
4) Sj form WFG with new info to detect.

Problem

↳ false deadlock (due to async snapshots)

Date ____ Page No.: ____

example



Performance:

1) O(n(n+1)/2) messages
2) O(n) message size
3) O(n) delay to detect deadlock

Centralized → completely centralized → Ho-Ramanoorthy

Distributed → Edge chasing
Path Pushing

Hierarchical → Manasa-Muntz
Ho-Ramanoorthy

Date __/__/__ Page No.:

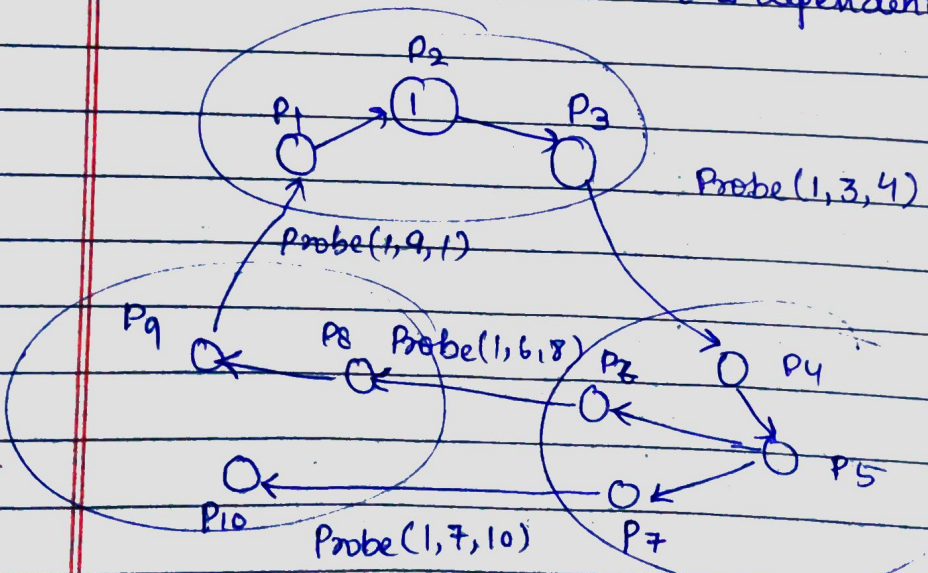
Chandy Misra Haas's Algo

Edge chasing Algorithm

1) In this, we use a special msg. called probe
A probe is a triplet (i, j, k) denoting that it belongs to a deadlock detection initiated for process P_i, P_j, P_k

2) Here we construct ~~GTWFG~~ GTWFG.
(Global Transition wait for Graph)

Data structure: Process (P_i, P_j, P_k)
↳ P_i is dependent on P_k if it exists a seq. of processes $\{P_i, P_{j1}, P_{j2}, P_{j3}, P_k\}$
↳ the system maintain a boolean array like dependent_j {if Dependent_j(i) ie P_i is dependent on P_j}



Date ___/___/_____ Page No.: _____

Edge chasing Algorithm → phantoms Deadlock Removal

3 steps

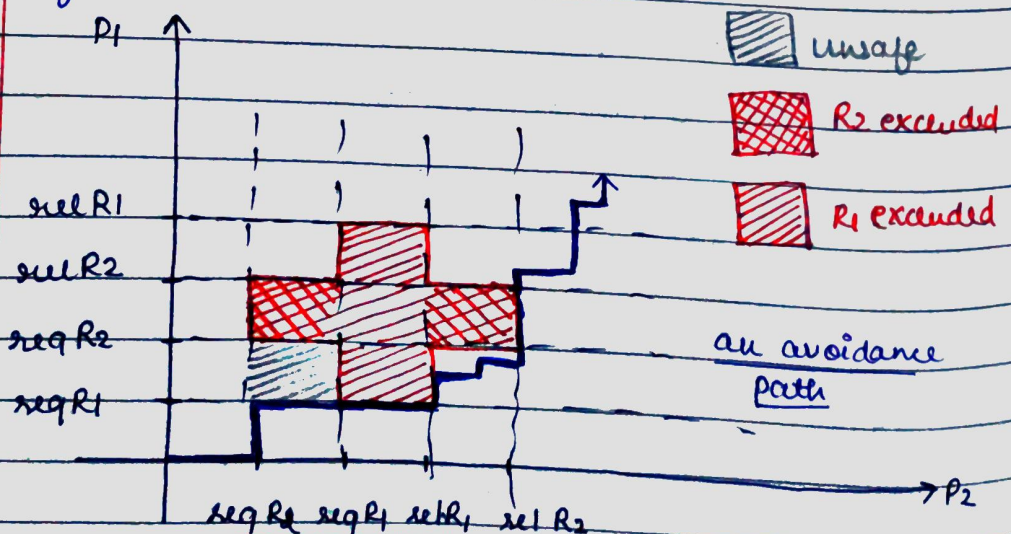
1) Initialization: Server initiates to detect deadlock
2) Detection: Detection consists of receiving probes and deciding whether deadlock has occured & forward probes.
3) Resolution: When a cycle is detected, a transaction in cycle is aborted to break deadlock

Performance

1) m(n-1)/2 exchange of msg:
   m → no. of processes
   n → no. of sites
2) Delay → O(n)
   to detect deadlock
3) 3-word message length

Date ___/___/_____ Page No.: ____

	Centralized control	Distributed Control	Hierarchical control
1)	A control site has the resp. to detect Global WFG	All site have the resp. to detect a GWFG.	Descendent site can detect GWFG.
2)	Have single point of failure	No single point of failure	No single point of failure.
3)	Easy to implement	Difficult to implement	Simple to implement
4)	Completely Centralized Algo, Ho-Ramamoorthy Algo	Path Pushing Edge-chasing Algo.	Menasce Muntz Ho-Rama-moorthy Algo



Date ___/___/_____ Page No.: _____

Deadlock Handling Strategies

A) **Prevention:**

1) Never satisfy → Mutual exclusion
   → Hold & wait
   → No preemption
   → circular wait

2) Methods 1) Collective Requests
   2) Ordered Requests
   3) Preemption.

B) **Avoidance:** Banker's Algorithm

A resource is assigned to a process if the state of global system is safe

C) **Detection:**

Maintaining WFG and searching the cycles

Deadlock prevention

(i) Ordered Allocation.

P1 req(R1), req(R2), release(R2), release(R1)

P2 req(R1), req(R2), release(R2), release(R1)

(ii) Release before Request

P1 req(R1), release(R1), req(R2), release(R2)

P2 req(R2), release(R2), req(R1), release(R1)

(iii) Request all at once:

P1 request(R1,R2), release(R1,R2)

P2 request(R1,R2), release(R1,R2)

Date __/__/____ Page No.: ____

Edge chasing Algorithm

1) In this, we use a special msg called probe
Probe is a triplet (i, j, k), sent for detection initiated by Pi by site Pj to site Pk
2) Deadlock is detected when probe returns to its initiator.
3) Here, we construct a global transition wait for Graph GTWFG

Terminology & DS:

(i) Pi is dependent on Pk if there exist a sequence of processes (Pi, Pj1, Pj2, Pj3, Pk)
(ii) Boolean Array
dependentj(i) = Pi is dependent on Pj

A) Algorithm Initiation by Pi

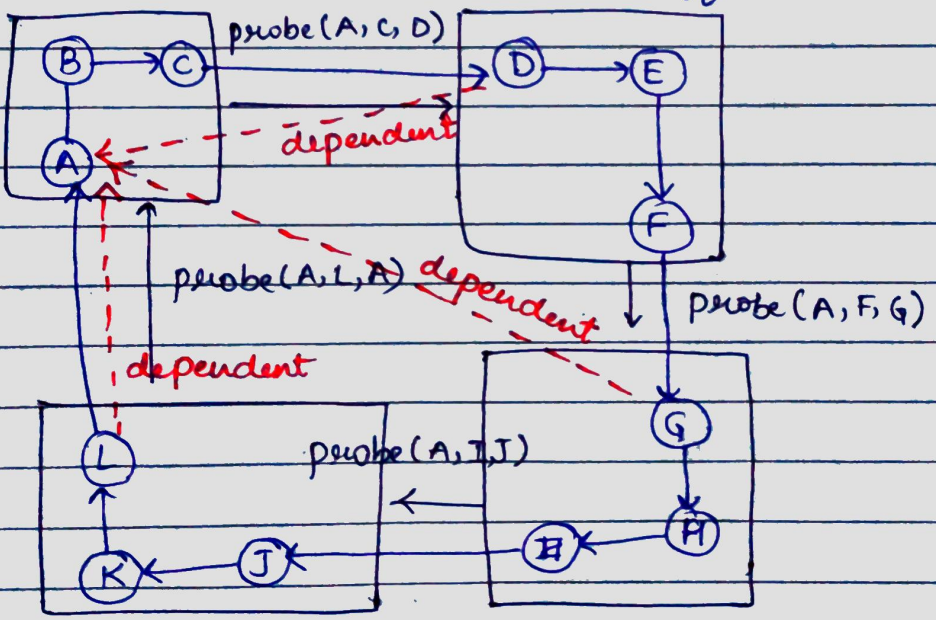
1) If Pi is locally dependent on itself then declare deadlock.
2) else send probe (i, j, k) to homesite of Pk if
   (i) Pi is locally dependent on Pj
   (ii) Pj is waiting for Pk
   (iii) Pj & Pk are on diff sites

Date __/__/____ Page No.: ____

B) Algorithm on receipt of probe (i, j, k)

1) Check the following
   (i) P_K is blocked
   (ii) $dependent_K(i)$ = false.
   (iii) P_K hasn't replied to all req. of P_j

2) If all are true.
   (i) set $dependent_K(i)$ = true
   (ii) If k=i ⇒ Pi is deadlocked
   (iii) else send probe(i, m, n) to home site of P_n for every m & n following condition hold.
      ↳ P_K is locally dependent on P_m
      ↳ P_m is waiting on P_n
      ↳ P_m & P_n are on diff. sites



Performance:
1) m(n-1)/2 messages → n = sites, m = processes
2) 3-word msg length
3) O(n) delay

"In gentle way, you can shake the world." –Mahatma Gandhi

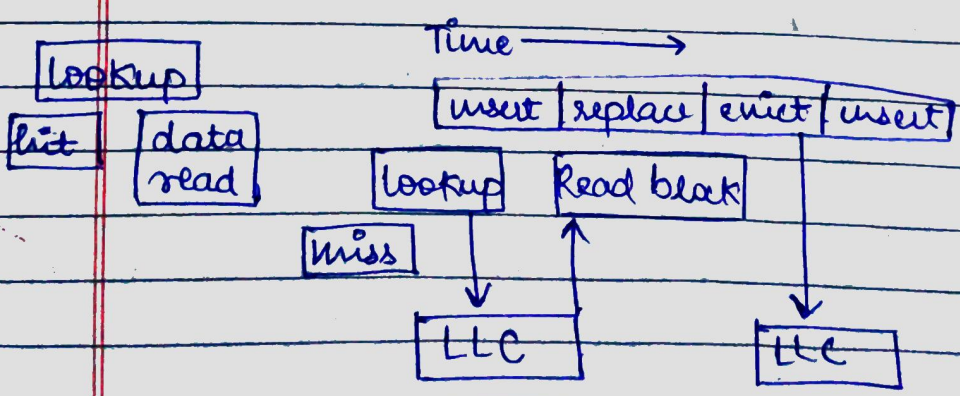
Date __/__/____ Page No.: ____

Cache

Data storage technique that provide the ability to access data or files at higher speed.

Read Operation

1) Starts with lookup operation.
2) If cache hit, then cache return value to processor or higher level cache.
3) If cache miss, then cancel read opr & send request to lower level cache.
4) Lower level cache will perform the same seq. of accesses & return entire cache block.
5) Cache controller then extract the requested data from block & send it to processor.
6) Simultaneously, cache controller invoke the insert operation to insert the block into the cache.



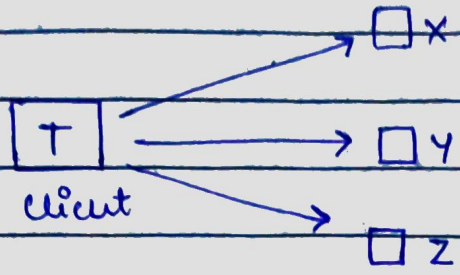
Distributed Transactions

Date ___/___/_____ Page No.: _____

Flat Transactions

↳ A flat transaction has a single initiating point and a single end point.

↳ for short activities.



Limitations:

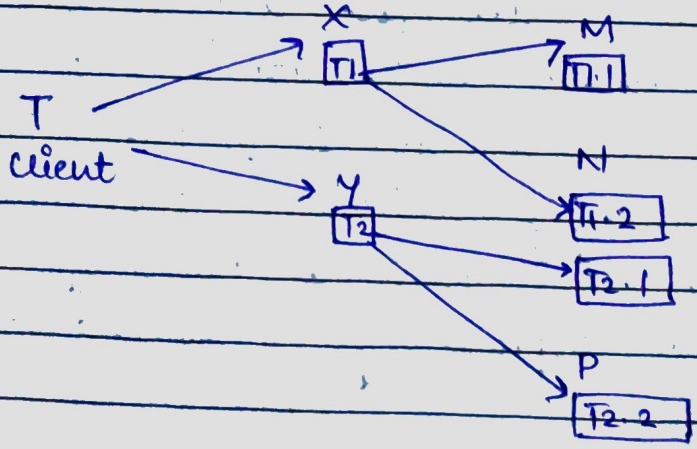
↳ All work is lost in the event of crash

↳ only one DBMS is used at a time

↳ No partial Rollback.

Nested Transactions:

↳ A transaction that includes other transactions within its initiating point & end point



Date __/__/____ Page No.: ____

Checkpointing:

↳ A mechanism that enables transactions to recover from inconsistent state using backward error recovery.

↳ In DS, all the sites save their local ~~sites~~ states which are known as local checkpointings, and the process of saving local state is called local checkpointing.

↳ All the local checkpoints, one from each site, collectively form a global checkpoint

Strongly consistent set of checkpoints

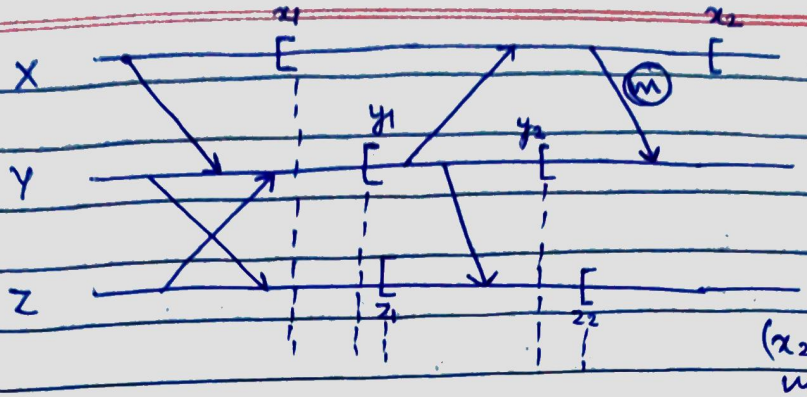
(1) Domino effect is caused by orphan msgs which themselves are due to rollbacks

(2) To overcome domino effect, a set of local checkpoints are needed such that no info. flow takes place bfw any set of processes without saving checkpoints

(3) A set of checkpoints is known as Recovery line or strongly consistent set of checkpoints

(4) A Global checkpoint is SCSoC if there is no orphan msg & no lost msg

Date ___/___/___ Page No.: ___



(x_1, y_1, z_1)
SCSOC

(x_2, y_2, z_2)
not strongly consistent set of checkpoint

Method of taking consistent set of cp:

① If every process takes a checkpoint after sending every msg, the set of most recent cp are always consistent.

② However it has high overhead, so we can resolve this, by taking cp after every K, K>1

Recovery in DDBS:

1) To enhance performance, availability, a DDBS is replicated whose copies are stored in diff sites.

2) The copies of DB at the failed site may miss some updates whereas sites which are not operational have inconsistent copies of data.

3) Proposed Approaches

- Message spooler: Save all the updates directed towards failed sites
- Copier transactions: Read up-to-date copies at all operational sites & update the copies at recovering site. It runs concurrently with user transactions.

Date __/__/____ Page No.: ____

4) Recovery algorithms should guarantee that-

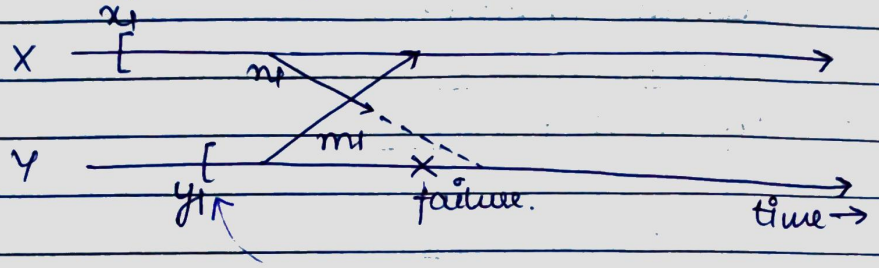
↳ The out of date replicas are not accessible to user transactions.

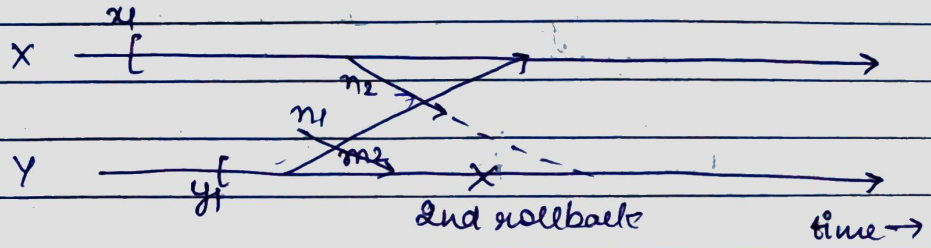
↳ Once outofdate replicas are uptodate by copies transactions, they are also updated along the other copies of user transactions

Date ___/___/_____ Page No.: _____

Livelocks:

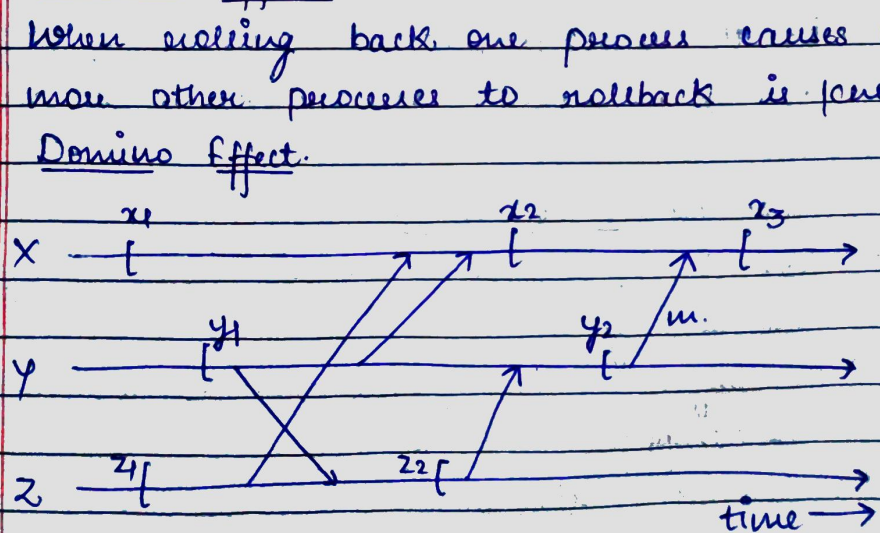
In rollback recovery, livelock is a situation in which single failure can cause an infinite no. of rollbacks preventing the system from making progress





Domino Effect:

When rolling back one process causes one or more other processes to rollback is known as Domino Effect.



Date __/__/____ Page No.: ____

Failure Resilient Processes:

↳ A process is said to be resilient, if it masks failures and guarantees progress despite a certain no. of system failure.

Approaches proposed to implement

↳ Resilient process:

a) Backup processes
b) Replicated Execution.

Backup processes:

↳ ① primary process & ⑪ backup processes.

primary process → terminates because of failure → backup process becomes active & takes over the fxn of primary process.

↳ To minimize computation that has to be redone by the backup process, the state of primary process is stored at appropriate intervals.

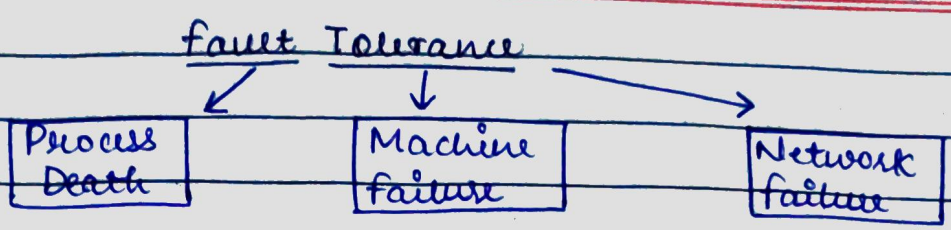
Replicated Execution:

↳ Several processes execute same program concurrently

↳ As long as one of the processes survive failures, the computation/service continues.

↳ Increases reliability & availability

Date ___/___/_____ Page No.: _____



Process Death:

↳ If process has reached a deadlocked state or had terminated abruptly, then all the resources allocated to the process must be recouped

↳ In client-server architecture, if client dies then server must be made aware about it so that all the resources can be released & vice-versa

Machine failure:

↳ In this case, all processes running in the Machine will die.

Network failure:

↳ A communication link failure can partition a Network into subnet, making it impossible for a machine to communicate with another machine in a different subnet.

Date __/__/____ Page No.: ____

Commit Protocols

- These protocols are used to ensure atomicity across sites
- A transaction which executes at multiple sites must either be committed at all sites or aborted at all the sites.

2-Phase Commit Protocol

Assumptions:

- One of the cooperating process acts as a coordinator other is refered as cohorts.
- Stable storage is available at each site
- each site uses write-ahead protocol to achieve local fault recovery & rollback.
- At the beginning of transaction, the coordinator reads start transaction msg to every cohort

A) **PHASE 1:**

- At coordinator
  a) coordinator → COMMITREQUEST → cohorts
  b) wait for reply.
- At cohorts
  a) If transaction executing at cohort is successful, it writes UNDO & REDO log on stable storage & sends AGREED msg to coordinator else send ABORT msg to the coordinator.

Date ___/___/_____ Page No.: _____

B) PHASE 2:

↳ at coordinator:

a) If all cohorts agree & coordinator agree, then coordinator writes a COMMIT record into the log & sends COMMIT msg to all the cohorts else ABORT msg.

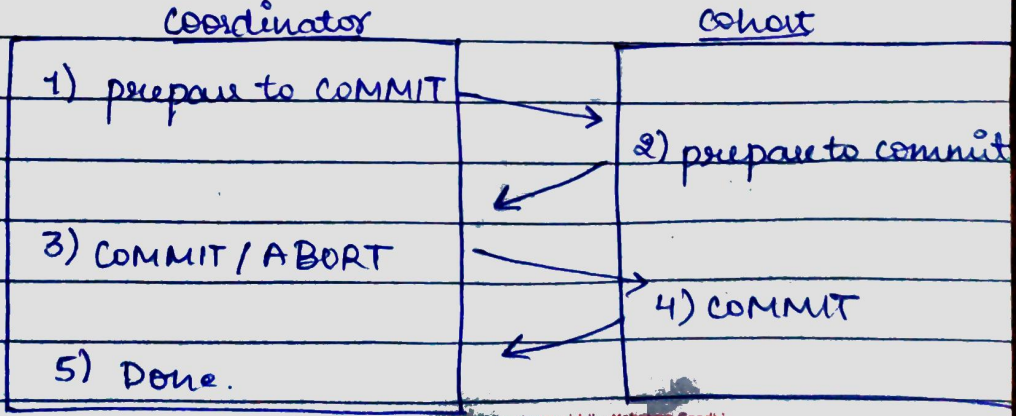
b) Waits for ACK from each cohort, If ACK is not received from any cohort within timeout, coordinator resends the COMMIT/ABORT msg to that cohort.

c) If all ACK's are received then coordinator writes a COMPLETE record to the log.

↳ at cohorts:

a) On receiving COMMIT msg, a cohort releases all the resources & locks held by it for executing transactions & send an ACK.

b) On receiving ABORT msg, cohort undoes the transaction using UNDO log record, releases all the resources & locks held & sends an ACK.



Date ___/___/_____ Page No.: _____

Voting Algorithm

↳ More fault Tolerant

↳ Allow data access under ① Network partition
② Site failure
③ Msg Losses

↳ Every replica is assigned certain no. of votes
This info is stored in Stable Storage.

↳ R/W ops is permission when a certain no of votes are collected by the req. process.

Assumptions / Basic Idea

① each file access require that an appropriate lock is obtained (R/W lock)
② each Site has lock Manager
③ each file has a version no. = no. of time it has been updated.
④ each replica has certain no. of votes.
⑤ vote allocation → stable storage.
⑥ R/W ops requires a quorum

Stable Storage

Lock Manager

4 steps

Date ___/___/_____ Page No.: _____

Static Voting Algo

Requesting set, $R = \{S_1, S_2 \ldots S_n\}$

1) S_i send LOCK REQ to its local lock manager
2) If request granted, S_i sends VOTE REQ to all sites in R.

3) S_j send lock REQ, if granted then it return Version No. & No. of votes assigned to replica.

4) S_i decides whether it has quorum or not.
   - ↳ If S_i doesnot have quorum, issue RELEASE LOCK
   - ↳ If S_i is successful, check if its copy of file is current
   - ↳ If request is read, read local copy
   - ↳ If request is write, update copy & Version No. and send RELEASE LOCK request

Dynamic Voting Protocols:

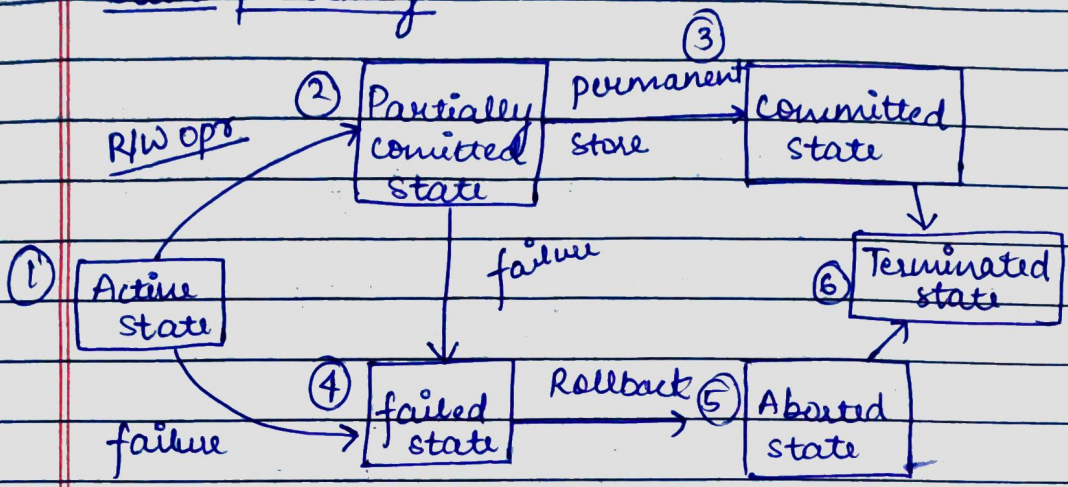
- ↳ change set of sites that can form majority
- ↳ change distribution of votes

Date __/__/____ Page No.: _____

Static Route Protocol	Dynamic route protocol
1) Non-adaptive in nature	1) Adaptive.
2) can cause comm. failure	2) prevents comm. failure
3) may / may not ensure availability	3) ensure availability.
4) Selection time of replica doesn't depend on system state.	4) selection time of replica depends on system state.

Date ___/___/_____ Page No.: _____

Transaction : A program including collection of DB opr, executed as a logical unit of Data processing



Properties of Transactions (ACID)

1) **Atomicity** :
   - ↳ either transaction executes completely or it doesn't occurs at all.
   - ↳ Responsibility of Transaction Control Manager

2) **Consistency** :
   - ↳ ensures integrity constraints are maintained, system remains consistent before & after the transaction.
   - ↳ Responsibility of DBMS & Application programmer

3) **Isolation** :
   - ↳ ensure multiple transactions can occur simultaneously without causing inconsistency, each transaction exe. is isolated
   - ↳ responsibity of Concurrency Control Manager

Date ___/___/_____ Page No.: _____

4) Durability:

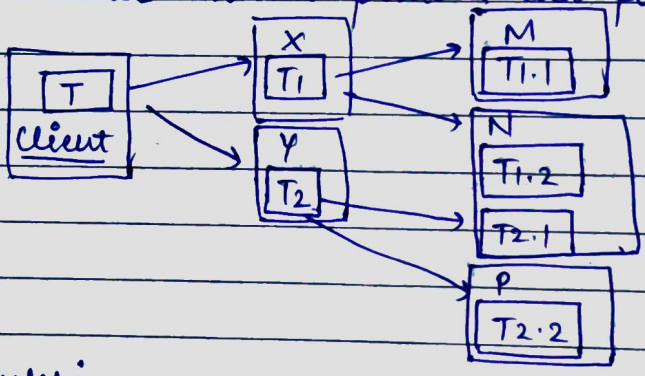
↳ ensures all the changes made by transaction after successful execution are written successfully in disk.

↳ Changes never lost in case of failure.

↳ Responsibility of Recovery Manager

Nested Transactions

↳ A transaction that includes other transaction within its initial point & end point



4/ Rules:

(1) When a parent transaction commits then all the sub transactions that have provisionally commit can commit.

(2) When a parent aborts, all of its subtransactions are aborted.

(3) When a child completes, it makes an idependent decision either to commit psorisionally or abort.

(4) When a child abort, parent can decide whether to abort or not.

Date ___/___/_____ Page No.: _____

Locks : Variable associated with shared resource that ensures serializability, by providing access to a data item in mutually exclusive manner

Types of locks:

1) Binary locks
   - ↳ locked (1)
   - ↳ unlocked. (0)

2) Shared / Exclusive lock

(i) Shared: If Ti locks X in shared mode then before Ti unlock X, no other transaction Tj can write into X, only Tj can read.

(ii) Exclusive: If Ti lock X in exclusive mode then before Ti unlocks X, no other transaction Tj can read or write into X.

Strict → 2PL → No shrinking phase
→ Release all the locks when whole transaction commit

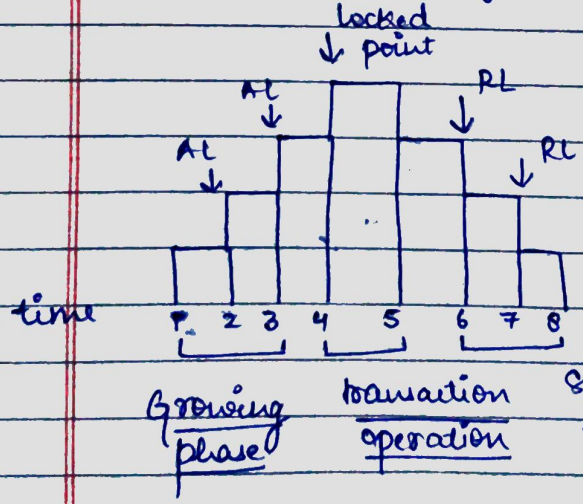
Date __/__/____ Page No.: ____

Two Phase locking Protocol

2PL is a method of concurrency control in DS that ensures serializability by applying lock to transaction data which blocks other trans. to acces same data simultaneously

1) **Growing Phase:**
   - ↳ When transaction begins to execute, it requires permission of locks on DI, it needs.
   - ↳ In this phase, transaction may obtain any locks but may not release any lock

2) **Shrinking phase:**
   - ↳ After executing transaction opr, it releases all the aquired locks on DI
   - ↳ In this phase, transaction can only release locks but not obtain any locks



AL → aquire lock
RL → release lock

Date __/__/____ Page No.: ____

Timestamp Ordering Protocol

TS(Ti) Timestamp : unique identifier created by DS to identify a transaction

for two transaction, Ti & Tj

If TS(Ti) < TS(Tj)

⇒ Ti started before Tj

With every shared data item, 2 timestamps are associated.

(i) R_TS(X) → largest TS value that execute successful read operation Read(X)

(ii) W_TS(X) → largest Ts value that execute successful Write operation Write(X)

Conditions :

1) Ti issues Read(X) opr.

↳ If W_TS(X) > TS(Ti) then opr is rejected

↳ If W_TS(X) ≤ TS(Ti) then opr is executed.

↳ timestamps of all data items updated.

2) Ti issues Write(X) opr

↳ If TS(Ti) < R_TS(X) then opr is rejected

↳ If TS(Ti) < W_TS(X) then opr is rejected & Ti is rolled back.

↳ else operation is executed.

Date __/__/____ Page No.: _____

Validation-Based Protocol
(Optimistic Concurrency Control Technique)

In VBP, transaction is executed in 3 phases

1) Read Phase:
↳ Transaction T is read & executed
↳ It is used to read the value of various data items and store them in temp variable. It can perform all write operations on temp variable without update of actual database.

Buffer

2) Validation Phase:
In this phase, temp variable is validated against the actual data to see if it violates the serializability

3) Write Phase:
If validation of transaction is validated, then the temporary results are written in db/system otherwise transaction is Rolled back.

Each Phase has the following Timestamps:
(i) Start(T_i): time when T_i starts its execution
(ii) Validation(T_i): time when T_i finishes its read phase and starts validation phase.
(iii) finish(T_i): time when T_i finishes its write phase.

Date ___/___/_____ Page No. _____

Comparison of Methods of Concurrency Control

Locking	Timestamp	OCC
The locking schemes are used to restrict the availability of DI for other transactions	It works on checking of TS TS is the start time of transaction generated by logical clock	All transactions are allowed to proceed but some are aborted when they attempt to commit

Timestamp ordering	Two phase lock Protocol
1) Decide serialization order Statistically	1) decide serialization order dynamically
2) Better than locking for Read dominated trans.	2) Better than TS ordering for update-dominated trans.

Both are pessimistic Methods

TM → user & DBS

Date ___/___/_____ Page No.: _____

Concurrency Control in Distributed Transaction

DBS contains 3 Modules

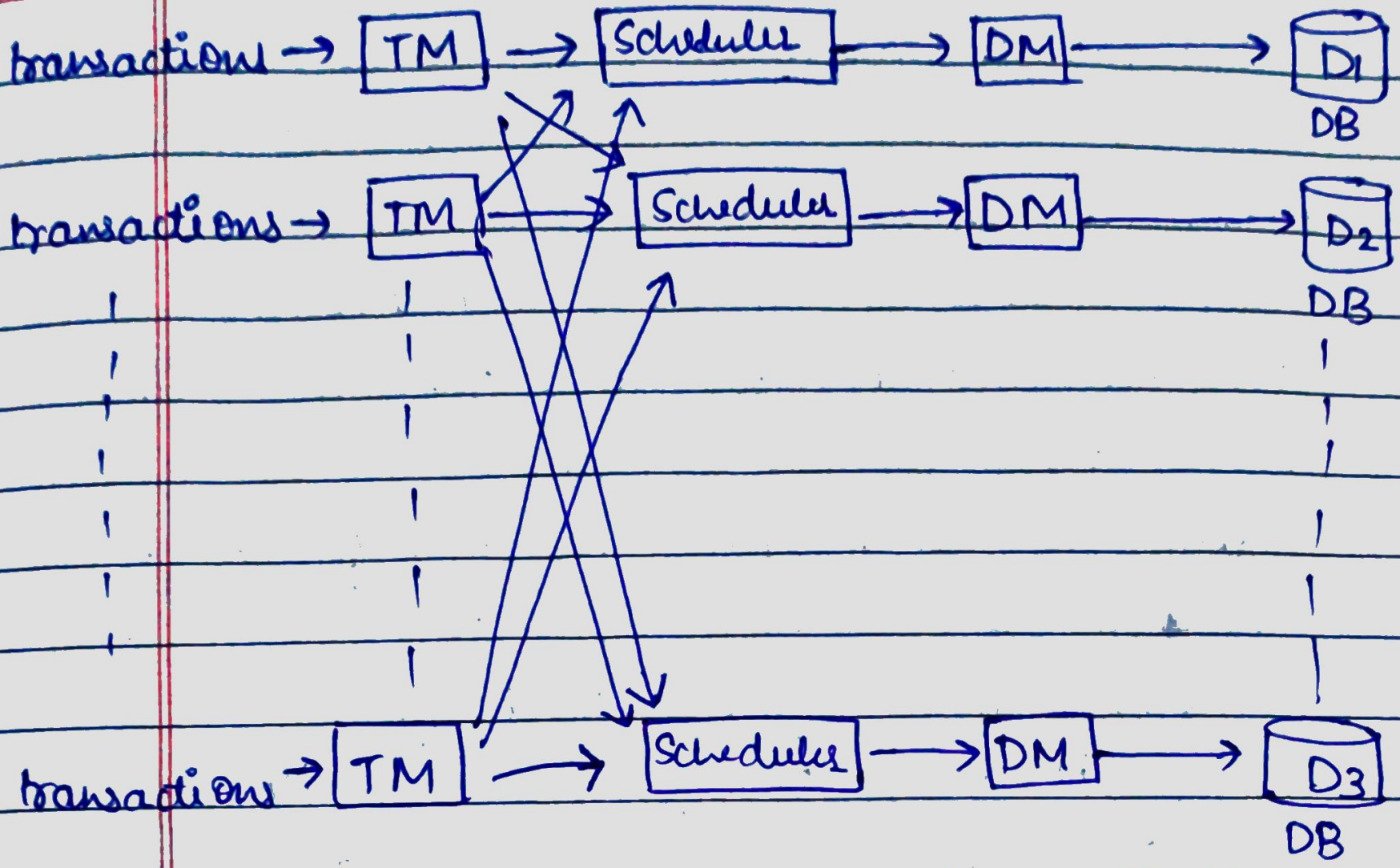
① Transaction Manager ② Data Manager ③ Scheduler

1) Transaction Manager:
- supervise the execution of transaction
- Interact with DM to carry out the eot.
- It assign a TS to a transaction or issue request to lock / unlock data obj. on behalf of user.
- Acts as interface b/w user & DBS

2) Scheduler:
- used to enforce concurrency control.
- It grant / release locks on data objects as required by a transaction.

3) Data Manager:
- Manages the DB
- It carries out R/W request issued by TM on behalf of a transaction by operating them on DB
- DM is interface between schedules & DB.
- Execution of transaction at TM results in execution of its actions at the DM.

Date ___/___/_____ Page No.: _____

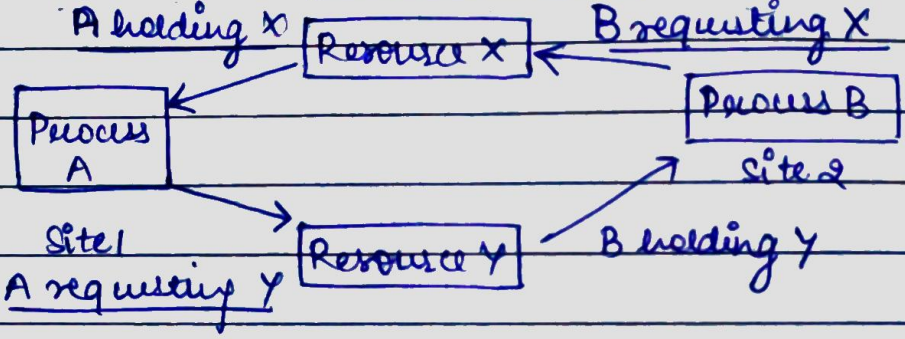

transactions
TM
Scheduler
DM
D1
DB
transactions
TM
Scheduler
DM
D2
DB
transactions
TM
Scheduler
DM
D3
DB

Date ___/___/_____ Page No.: _____

Distributed Deadlock

DS consists of a no. of sites connected by a network each site with globally unique identifier, makes resource req. to a controller
The controller at each site could maintain WFG on a process request.
Each site WFG could be cycle free & yet DS could be in Deadlock.
This is called Global Deadlock.

eg.



Transaction Recovery : REPLICATION

↳ Replication is the maintenance of copies of data multiple sites.

↳ Enhance performce, Higer Availability Fault Tolerance, Increase Reliability less Access time, Easy load Balancing

Problem : Data Inconsistency

Date ___/___/_____

Page No.: _____

Group Communication

↳ It occurs when source process sends a msg to a group of processes (Multicast comm^n).

↳ Groups are useful for managing Replicated data and in other system where processes cooperate towards a common goal by receiving & processing the same set of multicast msgs.

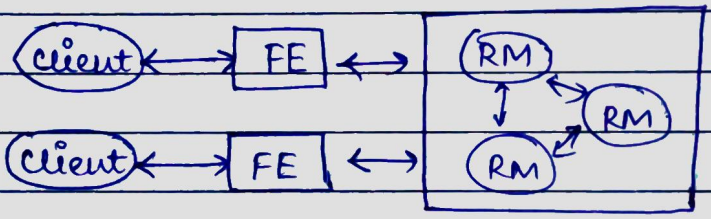
Date __/__/____ Page No.: ____

Highly Available Services:

Replication technique is used to make service highly available
ie. client access the service with reasonable response time.

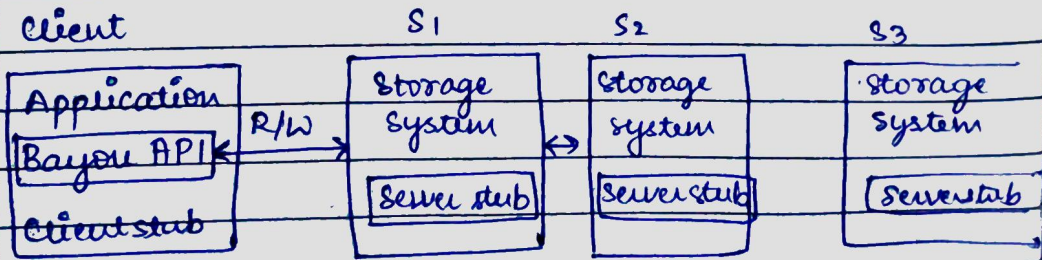
(i) Gossip Architecture:

↳ Client request service operation which are processed initially by front-end.



↳ Replica Manager update one another by exchanging gossip msg which contain the most recent update.

(ii) Bayou's Basic System Model:



↳ It is a replicated, weakly consistent storage designed for Mobile computing envt.

Date __/__/____ Page No.: ____

(iii) Coda file system:

↳ Purpose of CFS is enabling disconnected operations.

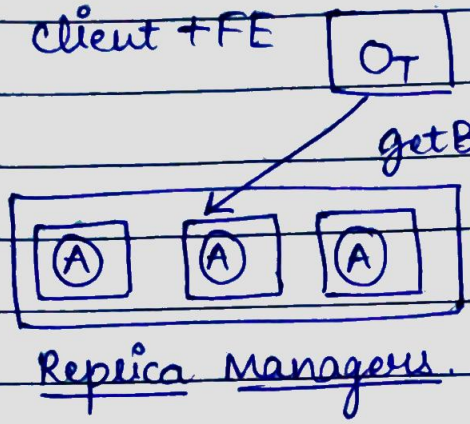
↳ Client communicate over high bandwidth network with server.

↳ Coda use optimistic replica control for high availability.

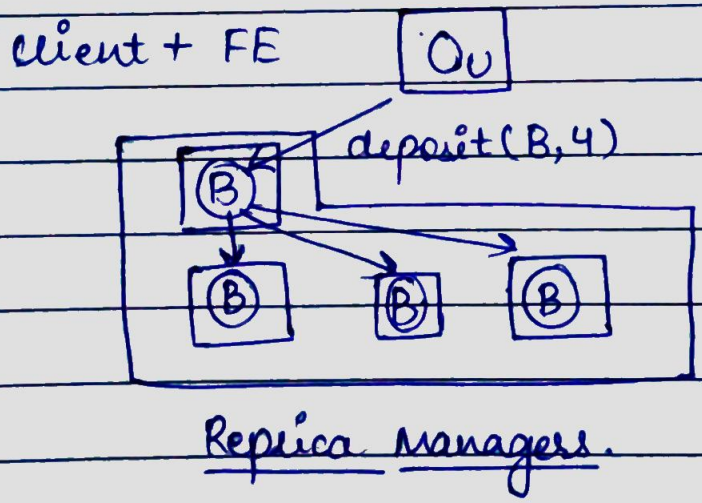
Date ___/___/_____ Page No.: _____

Transaction with Replicated Servers

(i) **Read operations**:



(ii) **Write Operation**:



Date ___/___/_____ Page No.: _____

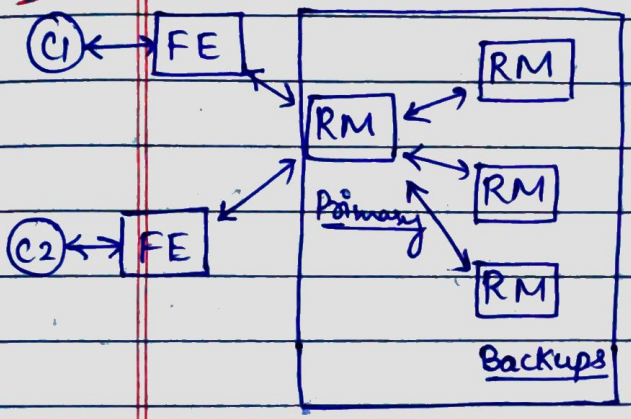
Fault Tolerant services

↳ FTS are obtained by using replication.

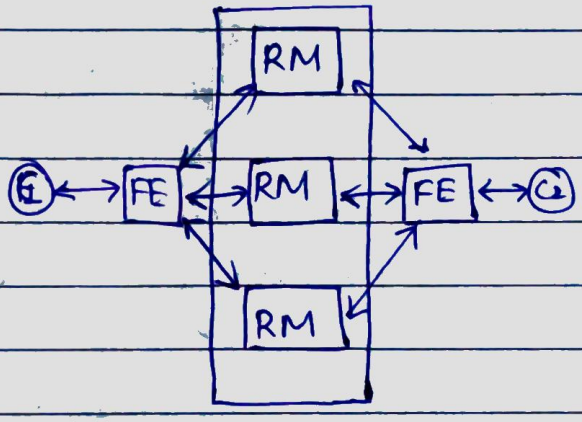
↳ By using Multiple Independent Server replica Managers each managing replicated data.

Primary Backup Replication

Passive Replication



Active Replication



RM acts as state Machines

Sequence of Events in Both Replications

unique identifier

1. Request
2. Coordination
3. Execution
4. Agreement
5. Response

Date __/__/____ Page No.: ____

Passive Replication :

↳ single primary replica Manager, one or more secondary RM ie slaves or Backups'

↳ Front end communicates with primary RM to obtain the services

↳ Primary RM executes the operations & sends copies of updated data to backups

↳ If primary RM fails, one of the backup is promoted.

1) Request: FE issues request, containing Unique Identifier to primary RM.
2) Coordination: PRM takes each request atomically. It checks the UI in case it has already executed the request.
3) Execution: PRM executes request & stores the response.
4) Agreement: If request is an update then primary send the updated state, response and UI to all the backups. The backup send an ~~agreement~~ acknowledgement
5) Response: PRM respond to FE which sends response back to client.

Date ___/___/_____ Page No.: _____

Active Replication:

↳ RM are state machines and play equivalent roles and are organized as group.

↳ FE multicast their requests to Group of RM and all RM process the req. independently but identically & reply.

1) **Request:** FE attach UI to request & multicast to Group of RM
2) **Coordination:** The Group Communication system delivers request to every RM in same order.
3) **Execution:** Every RM executes the request
4) **Agreement:** Not needed
5) **Response:** Each RM sends its response to FE

guarantees serializability

Date ___/___/_____ Page No.: _____

Multiversion Timestamp Ordering Protocol

- Multiversion schemes keep old version of data item to increase concurrency.
- each successful write results in creation of new version of data item written.

Each data item Q has seq. of versions $<Q_1, Q_2, \ldots Q_K>$ Each version Q_K contain 3 data fields:

① content — Value of version Q_K

② W-TS(Q_K) — TS of transaction that created Q_K

③ R-TS(Q_K) — largest TS of trans that successfully read Q_K.

↳ When T_i creates new version Q_K of Q Q_K's W-TS and R-TS are initialized to TS(T_i)

↳ Suppose T_i issues read(Q) or write(Q)

1) If T_i issues read(Q) then value returned is content of version Q_K.

2) If T_i issues write(Q).

(i) if TS(T_i) < R-TS(Q_K), T_i is rolled back.

(ii) if TS(T_i) = W-TS(Q_K), content of Q_K overwritten

(iii) else new version of Q created.

Page No.:

Advantages:

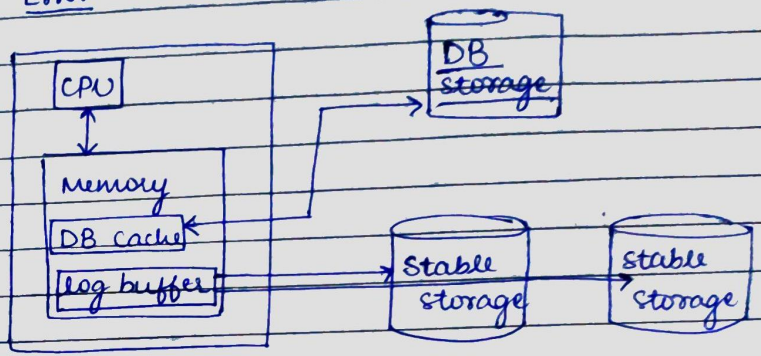
1) It allows more concurrency in DS
2) Read request never fail & never made to wait
3) Improved system responsiveness by providing Multiple Versions
4) Reduces the portability of conflict transactions

Disadvantages:

1) Conflict b/w transactions are resolved through rollbacks rather than waits.
2) Require huge amount of storage for storing Multiple versions of data objects.
3) Doesnot ensure recoverability & cascadelessness

Date

Backward Recovery Error



↳ doesnot lose info in case of sys failure
↳ Keep logs & Recovery points

Operation Based

↳ Record a log (audit trail) of operations performed.
↳ Restore previous state by reversing steps

State based

↳ Record a snapshot of state (checkpoint)
↳ Restore state by reloading snapshot (rollback)

Date __/__/____ Page No.: ____

Forward Error Recovery:

1) Remove errors in the process or system state if errors can be easily and completely assessed
2) Continue process or system forward execution.

Advantages:

1) less overhead

Disadvantages:

1) limited to use
2) cannot be used as general mechanism for error recovery.

Backward error Recovery:

1) Restore process or system to previous error-free state and restart from there
2) Two approaches to implement this
   (i) Operation Based
   (ii) State Based

Advantages

1) Simple to implement
2) Can be used as a general mechanism for error recovery

Disadvantages:

1) No guarantee that fault doesn't occur again
2) Some components can't be recovered
3) performs penalty

Date ___/___/_____ Page No.: _____

Fault is anything that deviates from what we expect (when something unexpected happens in system.

Errors are manifestation of faults in system and when propagate through system can lead to failure.

Types of faults

1. Transient occur once & goes away
2. Intermittent occurs, disappear occurs again
3. Permanent occurs once continues until fixed.

Types of failures

1. Method failure (process failure)
2. System failure
3. Disk failure (secondary storage failure)
4. Media/ Network failure