Distributed System:

↳ A system in which components are connected by a communication network, communicate and coordinate 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 transparent to others)

Motivation: Resources Sharing Behind DS

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

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 ts

(2) A logical clock is a mechanism of capturing Chronological & Casual 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.

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

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

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

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

Date ___/___/_____ Page No.: _____

C_i is incremented bfw any 2 successive events in process P_i

[IR1] If $a \rightarrow b$ then

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

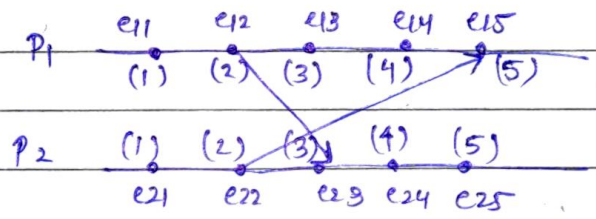
d = drift time (generally 1).

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

On receiving same msg m in process P_j then

$$C_j = \max(C_j, tm+d) \quad d > 0$$

Example



$C_{P_1} = 0$ $d = 1$

$C_{P_2} = 0$

Date

Page No.:

casual ordering

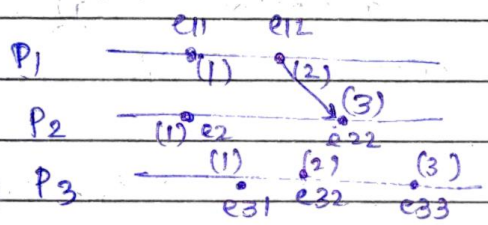
Limitations of Lamport's clocks

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.



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

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

e11 is causaly related to e22 but not e32

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 casual ording of 2 events occuring in diff processes.

generates partial ordering
detects causality violation

Date / / Page No.:

Vector clock

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

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

$C_i[i]$ → ith entry of C_i, corresponds to P_i's own logical time.
$C_i[j]$ → p_i's best guess of logical time at P_j

[IR1] Clock C_i is incremental b/w 2 successive events in process P_i

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

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

d = drift time = 1

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

$$t_m = C_i(a)$$

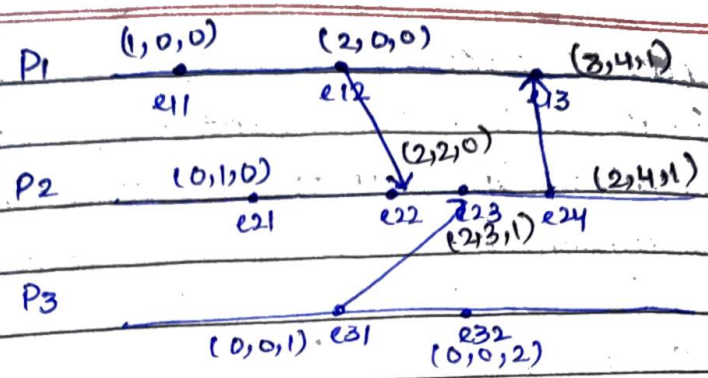
on receiving msg m by process P_j

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

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

Date ___/___/_____ Page No.: _____

eg.



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

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

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

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

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

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

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

$e_{24} = (C_i^0 + d) = \underline{(2,4,1)}$ ✓

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

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.

Date ___/___/_____ Page No.: _____

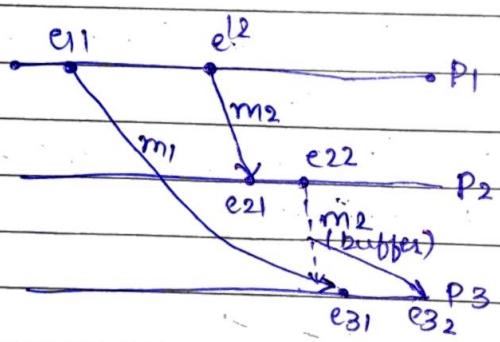
Casual Ordering of Msg maintaining

Deals with the concept of a casual relationship among 'msg sent' event corresponding to 'msg recived' 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 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. use buffer the msg.
3) Update the vector clock for process.

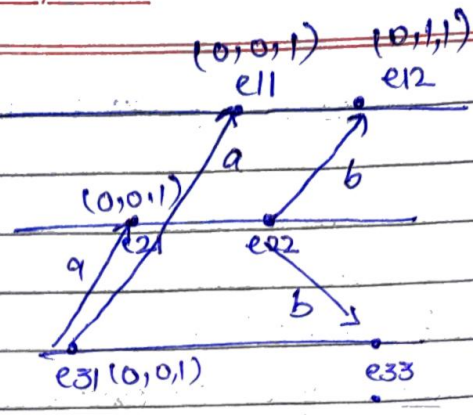
References:
P_i = process
e_{ij} = i → process no, j → event no.
v_i = v.c associated with P_i
$v_i[j]$ = jth element.

Protocol:
1) P_i sending msg. to P_j
P_i increments $v_i[i]$ and sets timestamp $t_m = v_i[i]$ for m.
2) P_j receiving msg. from P_i
a) When P_j, $j \neq i$ receives m with ts t_m, it delays msg. delivery until both are as follows.
① $v_j[i] = t_m[i] - 1$ &&
② $\forall K <= n$ && $K \neq i$ $v_j[K] <= t_m[K]$
b) When msg. is delivered to P_j, update P_j's VC.
c) Check buffer if anything can be delivered.

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

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 $c_1 = (0,0,1)$

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

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

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

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

Date __/__/____ Page No.: ____

e32 P3 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

$\forall i \forall j : 1 \le i, j \le n :: \text{transit}(LS_i, LS_j) = \phi$

Thus all communication channels are empty in transitless Global state.

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

(Consistent + transitless)

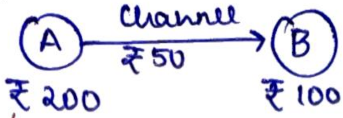
(i) All channels are empty.

(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 ↓ Idle
Active

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:

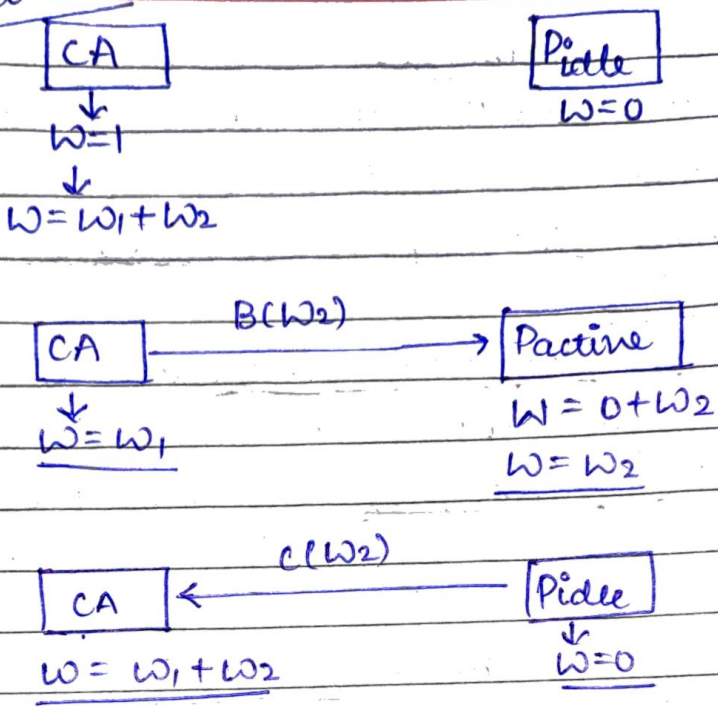
1) Rule to send B(DW):
   - ↳ CA having weight W may send a
   - ↳ computation msg to process P.
   - ↳ split the weight, $W = W_1 + W_2$, $W_1 > 0, W_2 > 0$
   - ↳ set weight of P, $W = W_1$
   - ↳ Send $B(W_2)$ to process Q. $DW = W_2$

2) On receiving B(DW) by process Q:
   - ↳ Add DW to weight of process Q, $W = W + DW$
   - ↳ If B 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 proc 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 = 10 ⇒ 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
2) freedom from Starvation
3) Fairness
4) Fault Tolerance.

Token based Algo	Non-Token Based Algo
1) Token is shared among all the sites.	1) Central site will communicate with all sites
2) Token contains seq. of No. of site inorder to ~~execute~~ req. CS.	2) It uses 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

Non Token

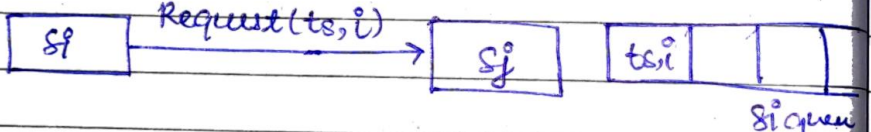
Date __/__/____ Page No.: ____

Lamport's Non-Token Algorithm

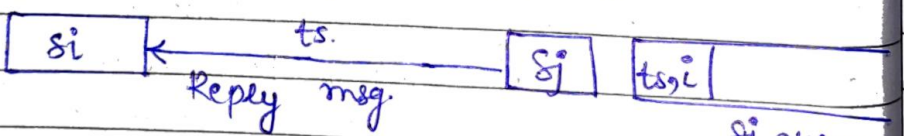
Requesting set
$R_i = \{S_1, S_2 \ldots S_n\}$, every site has queue.

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$_j$.

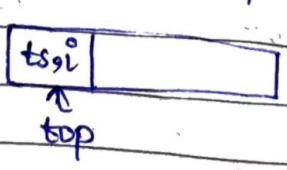


2) **Executing the CS:**

S_i enters in CS if

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

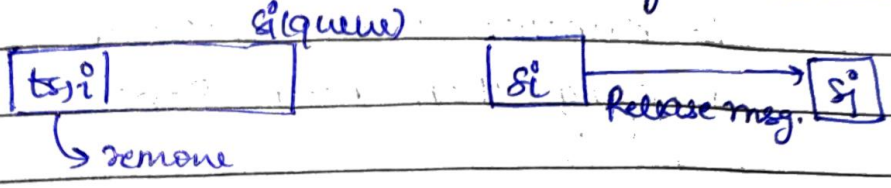
(ii) S_i's request is at top of requestqueue$_i$.



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 Ri



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.

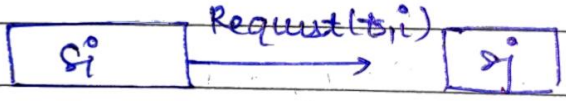
Ricart-Agarwala Non-Token Algorithm

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

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

1) **Requesting the CS:**

a) S_i wants to enter the CS, it send ts. 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 deferred.

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 deferred 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.

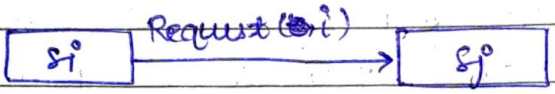
Date __/__/____ Page No.: ____

Maekawa's Algorithms

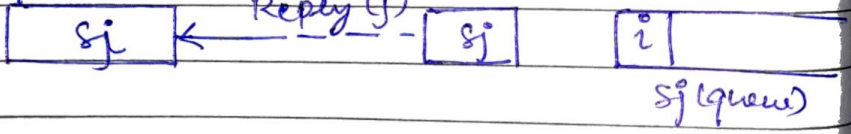
Requesting set, $R_i = \{S_1, S_2 \ldots S_n\}$

1) **Requesting the CS:**

a) S_i wants to enter the CS, it sends request msg to all the sites 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.

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

Date __/__/____ Page No.: ____

c) If queue is empty, Sj updates its status 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).

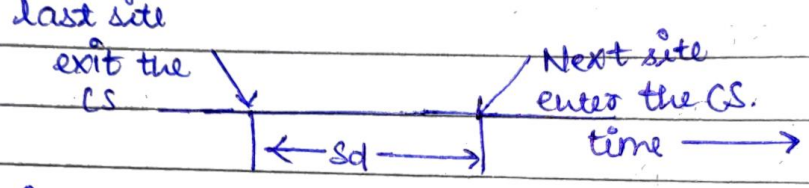
Drawbacks:

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

Date ___/___/_____ Page No.: _____

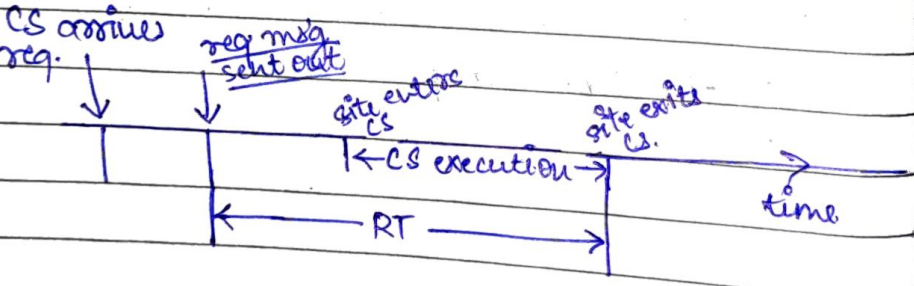
Performance Metric

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 schipes stephenson
Termination Detection - Haang's Termination CA
Lamport's logical clock Cj = max(Cj, tm+d) Rules
Vector clock Cj[k] = max(Cj[k], tm[k]) ∀ k

Date 1/09/

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) Access control.
2) Security
3) Availability
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.

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

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

Inconsistent cut:

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

1) e → sending event of msg.
   e' → receiving event of msg.
   $e' \in C \wedge e \notin C$.

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

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

Date __/__/____ Page No.

Architecural Model

→ interms of computation & communication

various spr & hfar architecture exits that are wally used for distributed computing.

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

a) Client Servers Model. → Dependency
b) 3-Tier Architecture. → No dependency
c) Peer to Peer Model. c) tightly coupled (cluster)
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.

Date __/__/____ Page No.: _____

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.

Date ___/___/_____ Page No.: _____

Byzantine Agreement Problem

An arbitrary chosen ~~pre~~processor (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.: _____

Interactive 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 __/__/__

Lamport's Shostak Pease Algorithm:

Page No.: _____

Lamport's Shostak Pease Algorithm:

One Message Algo OM(m) $m \geq 0$ 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 process
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 exits else it uses default value '0'.

Date ___/___/_____ Page No.: _____

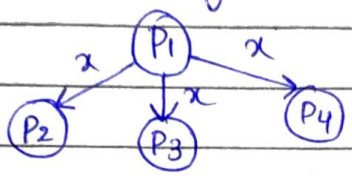
BA cannot always be reached among 4 processors if 2 processors are faulty.

Proof:

4 processors : P_1, P_2, P_3, P_4.

P_1 initiates the initial value.

P_2, P_4 are faulty.

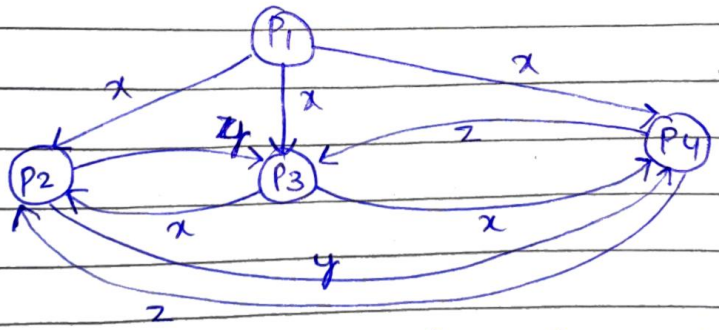


P_1 executes OM(1) and sends its value x to other processors.

After receiving 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 values for Byzantine solution:

processors	Received Majority	common
P_2	(x, x, z)	x
P_3	(x, y, z)	0
P_4	(x, x, y)	x

Date __/__/____ Page No.: ____

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.

Resource Management Component

Date __/__/____ Page No.: ____

Distributed file system (DFS)

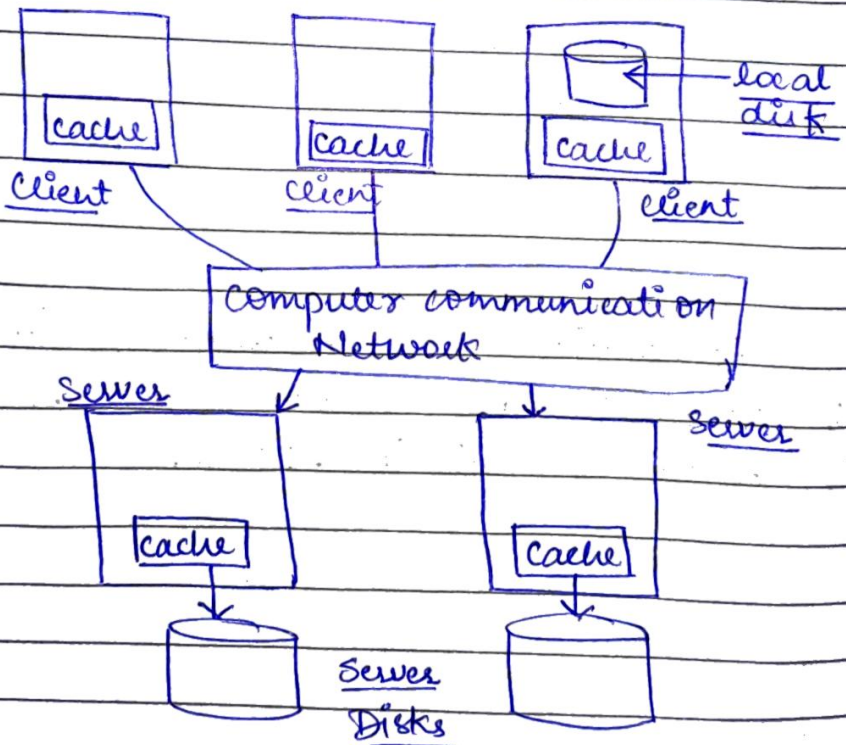
↳ DFS is a resource management component of Distributed Operating System.

↳ 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 :

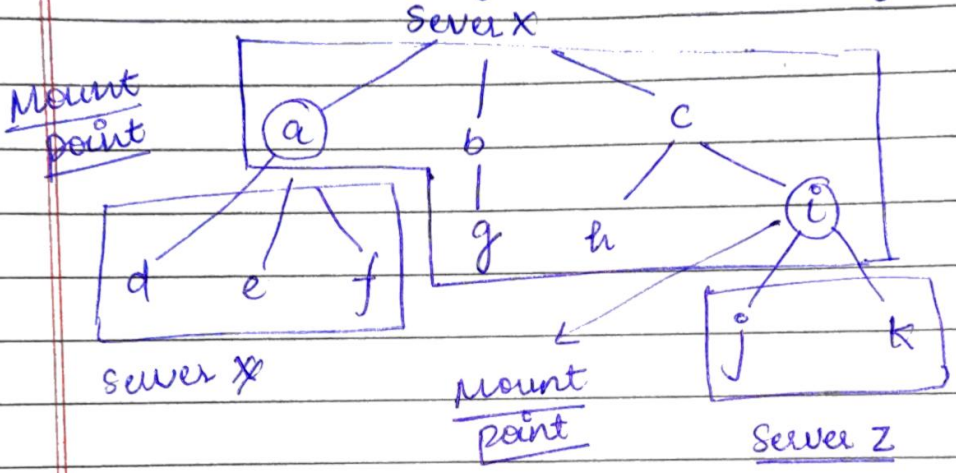
A process that maps names specified by the clients to stored objects such as files & directories.

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 guarantee consistency

Date __/__/____ Page No.: ____

3) Hints: Timestamp + cache

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

↳ can be used by application that can discover 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 need to communicate send req. to authentication server.

↳ Authentication server provide key for conversation.

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 Replication
c) Replica Management.

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

7) **Semantics:** It characterizes effects of accesses on files.

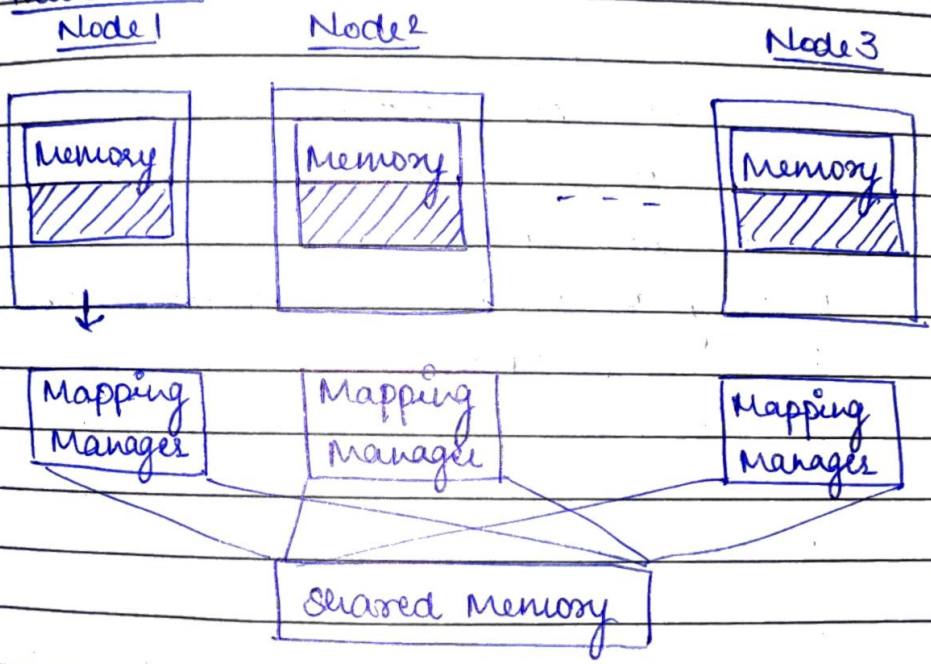
"Facts are facts and will not disappear on account of your likes." —Jawaharlal Nehru

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.

2) Page Replacement:
Replacement algo must take into acc pg. access modes.
↳ shared, private, readonly writable.
LRU private pg. replaced before shared ones, private pg. swap to disk, Shared pg sent over network to owner, Read only may be discarded.

Date ___/___/_____ Page No.:

Memory Coherence

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

① Sequential Consistency:
② General consistency:
③ process 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.
② Write update protocol.

Date __/__/____ Page No.: ____

Algo to implement DSM

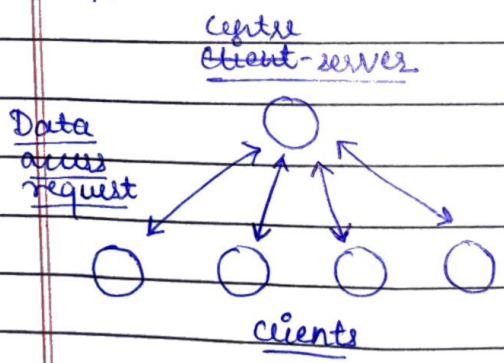
Central Server Algorithms:

↳ 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
Send data request →	Receive Request
	perform data access
	Send response
Receive response ←	

Date __/__/____ Page No.: ____

Migration Algorithm :

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

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

Date __/__/____ Page No.: _____

Read Replication Algorithm

Date ___/___/_____ Page No.: ___

Full Replication Algorithm