Name!- Nikhil kumar Roll no !- 1806055 (CSE-1) AN
Course Title !- Distributed systems
Course code !- CS 74079
Exame date !- 21/12/21
Program code !- UG-CS

Q ② (a) process - crash failures of coordinator !-
Ans!- (i) before sending COMMIT-REQ message -

→ on timeout of COMMIT-REQ, all Pi to abort independently
→ After recovery, P' to abort independently (no additional messages).

(ii) before sending prepare / Abort message:-

→ On time of prepare / Abort message ~~indendently~~, all Pi to abort independently.

→ After recovery, P' to abort independently.

(iii) before sending commIT message !-

→ on time of commIT message, all Pi to commit.
→ After recovery, P' to commit independently.

Q ② (b) process - crash failure of any single cohort !-
Ans!- (i) before sending AGREED / DISAGREED message:-

→ On timeout of AGREED / DISAGREED message P' to eventually broadcast ABORT message to all Pi, and all ther Pi to abort on receipt of ABORT message.

→ After recovery, failed Pi to ~~on~~ abort independently.

Name!- Nikhil kumar. Rollno!- 1806055 (CSE1) AN
Course Title!- Distributed systems, program code!-UG-CS
Course code!- CS7479
Exam date !- 21/12/21

CLASSMATE
Date :
Page : 2

Q. (5)

Sol:-

(ii) before sending ACK message:-

→ On time out of ACK message, pi to eventually broadcast ABORT message to all other pi to abort on receipt of ABORT message.

→ After recovery, failed pi to abort independently.

Name!- Nikhil kumar Roll no!- 1806055
Course code!- CS7479 Program code!- UK-CS
Course Title!- Distributed systems
Exam date!- 21/12/21

Q(3)

Ans!- (1) Linearizability!-
Criteria for Linearizability
(i) Inter leaved sequence to be same for all replica massage.
(ii) client's requests in inter leaved sequence ordered by global timestamp.
iii) Each client issuing order of sequence in inter leaved sequence to follow their actual order of occurrence.
In case of facebook since backup ~~sp~~ replica message will handle safe read request, linearizability not possible.

(ii) Sequential consistency!-
Criteria for sequential consistency
(i) Inter leaved sequence of write request to be same for all replica messages
(ii) each client program order preserved in interleaved sequence.

In case of facebook, a client safe read & safe write ~~safe~~ request to be executed by different replica massege, so sequential consistency also not possible.

(iii) Casual consistency!- Criteria for casual consistency.
(i) order of casually related write requests in Interleaved sequence to be same for all ~~sp~~ replica massages.
(ii) No ~~sp~~ of requirements about order of concurrent write requests in interleaved sequence.

Name !- Nikhil Kumar Roll no!- 1806055 (CSE1)
Course Title !- Distributed Systems
Course code !- CS7479 Program code !- UG-CS
Exam date !- 21/12/21

in case of facebook, causal consistency models shall work fine for most of time since all write request in interleaved by primary server but at times due to backup server handling read request, sometimes casuality may not be ~~computed~~ captured.

<iv> FIFO Consistency !-
criteria for FIFO consistency.

<i> ~~Ord~~ order of write request by individual clients in interleaved sequence to be same for all replica ~~msg~~ messages.

<ii> no requirements about order of write requests from multiple clients in interleaved sequency

in case of facebook, since primary replica ~~mang~~ mangers will always same for all replica messages hence FIFO consistency would be supported.

Name!- Nikhil kumar   Rollno!- 1806055 (CSE 1)   AN
Course Title!- Distributed systems
Course code!- CS7479
Exam date!- 21/12/21   program code!- UG_CS

CLASSMATE
Date :
Page : 4

page no (4)

4.(a) Ans!- • Note!- as per individual data item, sequential consistency achieved, as shown below.

• p2's view of x's order as per given signature : winit (x) 0 Wp1 (x) 1 Rp2 (x~~0~~) 1.

• P3's view of x's order as per given signature! winit (x) 0 Rp3 (x) 0 wp1 (x) 1. [P2's and p3's view for x not contradicted, complying rule 1]

• P1's view of y's order as per given signature ! w init (y) 0 Rp1 (y) 0 wp2 (y) 1.

• P3's view of y's order as per given signature ! winit (y) 0 wp2 (y) 1 Rp3 (y) 1. [P1's and p3's view of y's ~~and~~ not contradicted complying rule1].

• P1's view of z's order as per given signature! w init (z) 0 Rp1 (z) 0 Wp3 (z) 1.
• P2's view of z's order as per given signature ! winit (z) 0 Rp2 (z) Wp3 (z) 1 [p1's and p2's view for z matched, complying rule 1]

• Individual process's program order irrelevant here as each process performs only one operation on each data item. [no inconsistency, complying rule2]

Scanned with CamScanner

Name :- Nikhil Kumar Roll no :- 1806055
Course title :- Distributed systems programcode :- UG-CS
Course code :- CS7479
Exam date :- 21/12/21

4.(b) Ans :- Eventual consistency :- is a consistency model used in distributed computing to achieve high availability that informally guarantees that, if no new updates are made to given data item, eventually all access of to that item will return the last updated value. [1] Eventual consistency, also called optimistic replication, [2] is widely deploy in distributed systems, and has origins in early mobil computing projects. [3] A system that has achieved eventual consistency is often said to have converged, or achieved replica convergence. [4] Eventual consistency is weak guarantee - most stronger models, like linearizability are trivially eventually consistent. Eventually - consistent services are often classified as providing - consistent services are often classified as providing BASE semantics (basically - available soft ~~pro~~ state, ~~one~~ eventual consistency), in contrast to traditional, ACID (atomaticity, consistency, isolation, durability). [5] [6] in chemistry, a base is the oppasite of an acid, which helps in remembering the acronym. [7]

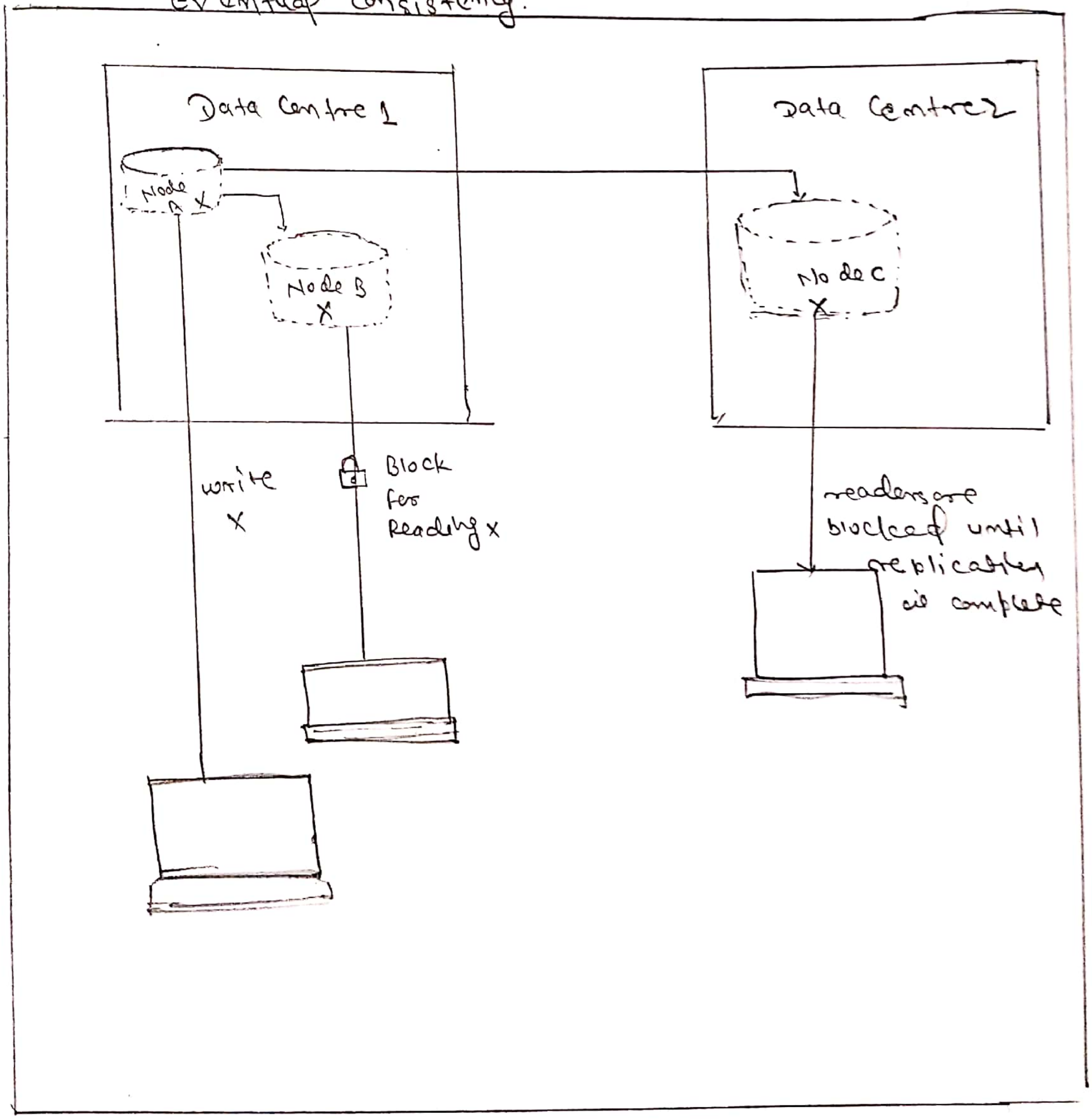
→ Basically available :- reading and writing operation are available as much as possible (using all nodes of database cluster).

→ Soft - state :- without consistency guarantee, after some amount of time, we only have some probability of knowing the state, since it may not yet have converged.

→ Eventually consistent :- if we execute some writes and they the system functions long enough.

Name!- Nikhil kumar | Branch!- CSE 1
Roll no!- 1806055 | Examdate!- 21/12/21 (AN)
Course Title!- Distributed systems
Course code!- CS7479 | Programcode!- UR-CS | pag no: 6

Eventual consistency.



Name !- Nikhil Kumar Roll no !- 1806055
Course Title !- Distributed systems program code !- UG-CS
Course code !- CS7479
Exam date !- 21/12/21

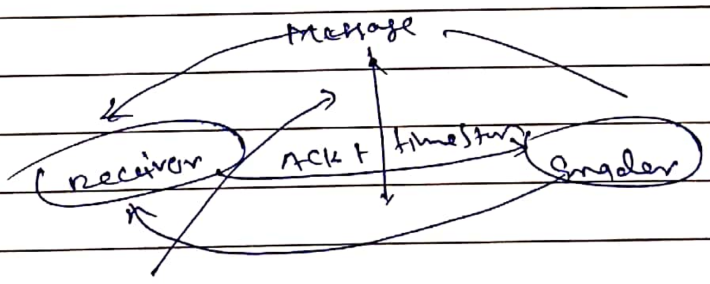
CLASSMATE
Date :
Page : 8

Q(1)
Ans !- (b)

Logging !-

when making progress - It is always important to capture progress. this way, it is ever interrupted, all is not lost. this can be done, for example, by periodic checking points by means "logging" or continuous replication (via, for eg RPC messaging).

Maintaining geographically diverse replicas provides a way to secure against whole system failures, as well as physical replicas provide, as to that may affect one location such as natural phenomena (earth quakes, fires, etc) and those created by humans.



Receiver can't message 3 Ack-Ack.