Part – B

1) Explain the concept of checkpoint and rollback recovery and discuss the issues in failure recovery with neat diagram.

2) List and define the different types of messages. Discuss the ways to handle the issues in checkpoint and rollback failure recovery with neat diagram.

For Question 1 and 2 refer the answers which are mailed earlier

(Brief the concepts)

3) Explain the algorithms for ordering the message using causal order with suitable diagram.

For Question 3 refer the answers which are mailed earlier (Raynal Schiper Toueg and Kshemkalyanai Singhal)

(Brief the concepts)

4) Explain the algorithms for ordering the message using Total order with suitable diagram.

For Question 4 refer the notes, dictated earlier in the class (Centralized and Three phase distributed)

Centralized Algorithm: Draw diagram and explain the concepts involved in it, step by step.

Three Phase Algorithm: Draw the diagram which is said in the class and explain the concepts in a step by step manner.

(Brief the concepts)

5) Explain the concept of global states and Snapshot algorithms for FIFO channels with suitable diagram.

Global State - Define

Consistent State - Define

Cuts - Define

Purpose of Cuts – Define and explain with diagram (PAST AND FUTURE)

Issues in recording a global state – Define (Three Conditions)

Chandy Lamport Algorithm (Explain Briefly)

Marker sending rule for process p_i

(1) Process p_i records its state.
(2) For each outgoing channel C on which a marker has not been sent, p_i sends a marker along C before p_i sends further messages along C.

Marker receiving rule for process p_j
On receiving a marker along channel C:
  if p_j has not recorded its state **then**
    Record the state of C as the empty set
    Execute the "marker sending rule"
  else
    Record the state of C as the set of messages received along C after p_j's state was recorded and before p_j received the marker along C

---

6) Elucidate the concept of Checkpoint-based recovery with suitable diagram.

Checkpoint-based recovery:

- In the checkpoint-based recovery approach, the state of each process and the communication channel is checkpointed frequently so that, upon a failure, the system can be restored to a globally consistent set of checkpoints.
- Checkpoint-based rollback-recovery techniques can be classified into three categories: uncoordinated checkpointing, coordinated checkpointing, and communication-induced checkpointing

a) Uncoordinated checkpointing:

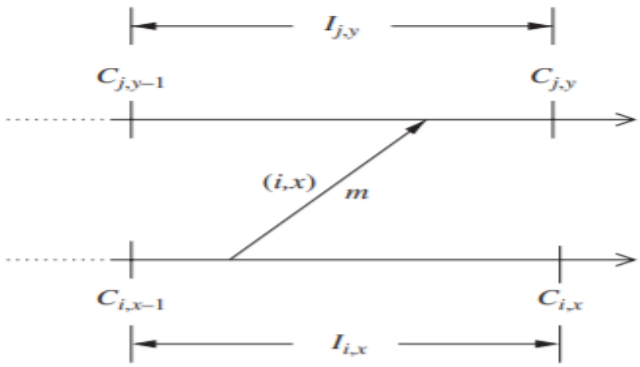
- In uncoordinated checkpointing, each process has autonomy in deciding when to take checkpoints
- This eliminates the synchronization overhead as there is no need for coordination between processes and it allows processes to take checkpoints when it is most convenient or efficient
- The main advantage is the lower runtime overhead during normal execution, because no coordination among the processes is necessary.

Disadvantage*:*

- First, there is the possibility of the domino effect during a recovery, which may cause the loss of a large amount of useful work.
- Second, recovery from a failure is slow because processes need to iterate to find a consistent set of checkpoints
- Third, uncoordinated checkpointing forces each process to maintain multiple checkpoints

When Fail:

- When a failure occurs, the recovering process initiates rollback by broadcasting a dependency request message to collect all the dependency information maintained by each process
- When a process receives this message, it stops its execution and replies with the dependency information saved on the stable storage as well as with the dependency information, if any, which is associated with its current state
- The initiator then broadcasts a rollback request message
- Upon receiving this message, a process whose current state belongs to the recovery line simply resumes execution; otherwise, it rolls back to an earlier checkpoint as indicated by the recovery line.



Let $C_{i,x}$ be the xth checkpoint of process P_i, where i is the process i.d. and x is the checkpoint index (we assume each process P_i starts its execution with an initial checkpoint $C_{i,0}$). Let $I_{i,x}$ denote the *checkpoint interval* or simply *interval* between checkpoints $C_{i,x-1}$ and $C_{i,x}$. Consider the example shown in Figure 13.5. When process P_i at interval $I_{i,x}$ sends a message m to P_j, it piggybacks the pair (i, x) on m. When P_j receives m during interval $I_{j,y}$, it records the dependency from $I_{i,x}$ to $I_{j,y}$, which is later saved onto stable storage when P_j takes checkpoint $C_{j,y}$.

Write only the points highlighted in yellow

b) Coordinated Checkpoint

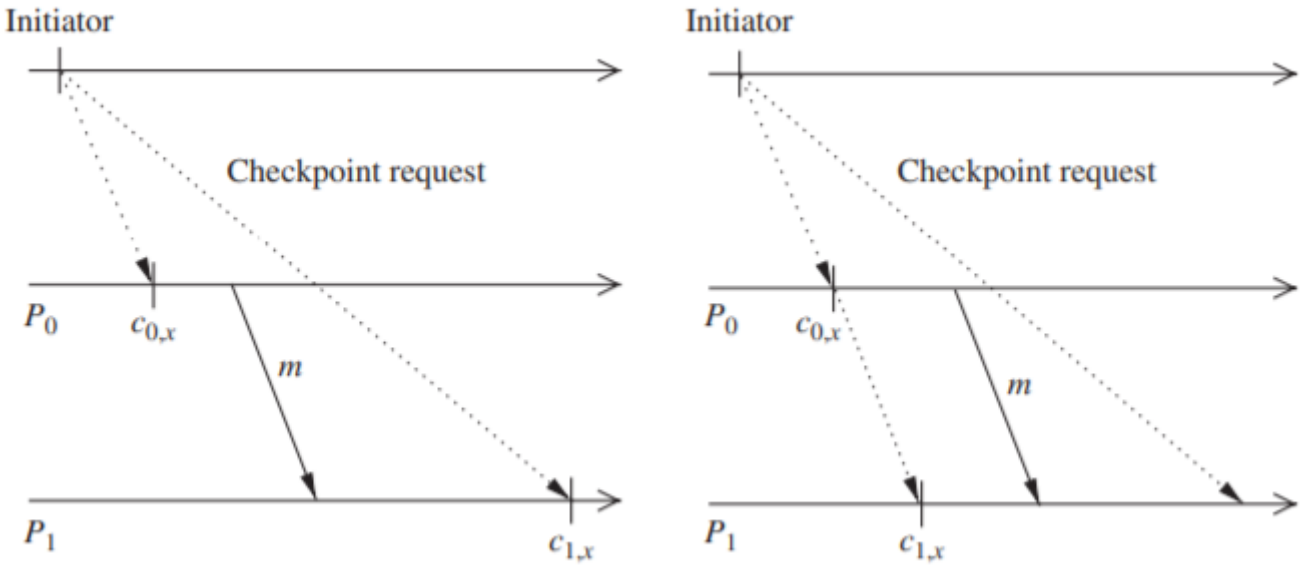
Blocking coordinated checkpointing

- After a process takes a local checkpoint, to prevent orphan messages, it remains blocked until the entire checkpointing activity is complete.
- The coordinator takes a checkpoint and broadcasts a request message to all processes, asking them to take a checkpoint
- When a process receives this message, it stops its execution, flushes all the communication channels, takes a tentative checkpoint, and sends an acknowledgment message back to the coordinator
- After the coordinator receives acknowledgments from all processes, it broadcasts a commit message that completes the two-phase checkpointing protocol
- After the receving the commit message all the slave processes makes the tentative checkpoint permanent and then resumes its execution and exchange of messages with other processes

Non-blocking checkpoint coordination

In this approach the processes need not stop their execution while taking checkpoints

In this algorithm, the initiator takes a checkpoint and sends a marker (a checkpoint request) on all outgoing channels



FIFO Channel:

Each process takes a checkpoint upon receiving the first marker and sends the marker on all outgoing channels before sending any application message

NON – FIFO Channel:

- first, the marker can be piggybacked on every post-checkpoint message. When a process receives an application message with a marker, it treats it as if it has received a marker message, followed by the application message
- Alternatively, checkpoint indices can serve the same role as markers, where a checkpoint is triggered when the receiver's local checkpoint index is lower than the piggybacked checkpoint index

c) Communication-induced checkpointing

- Communication-induced checkpointing is another way to avoid the domino effect, while allowing processes to take some of their checkpoints independently
- Processes may be forced to take additional checkpoints (over and above their autonomous checkpoints), and thus process independence is constrained to guarantee the eventual progress of the recovery line.
- In communication-induced checkpointing, processes take two types of checkpoints, namely, autonomous and forced checkpoints. The checkpoints that a process takes independently are called local checkpoints, while those that a process is forced to take are called forced checkpoints.

There are two types of communication-induced checkpointing: modelbased checkpointing and index-based checkpointing

Model-based checkpointing

- The MRS (mark, receive and send) model avoids the domino effect by ensuring that within every checkpoint interval all message receiving events precede all message-sending events
- This model can be maintained by taking an additional checkpoint before every message-receiving event that is not separated from its previous message-sending event by a checkpoint
- Another way to prevent the domino effect by avoiding rollback propagation completely is by taking a checkpoint immediately after every message-sending event

Index-based checkpointing

- Index-based communication-induced checkpointing assigns monotonically increasing indexes to checkpoints, such that the checkpoints having the same index at different processes form a consistent state
- Inconsistency between checkpoints of the same index can be avoided in a lazy fashion if indexes are piggybacked on application messages to help receivers decide when they should take a forced a checkpoint