ALGORITHM FOR ASYNCHRONOUS CHECKPOINTING AND RECOVERY (JUANG-VENKATESAN)

- This algorithm helps in recovery in asynchronous checkpointing.
- The following are the assumptions made:
  - communication channels are reliable
  - delivery messages in FIFO order
  - infinite buffers
  - message transmission delay is arbitrary but finite
- The underlying computation or application is event-driven: When process P is at states, receives message m, it processes the message; moves to state s' and send messages out. So the triplet (s, m, msgs_sent) represents the state of P.
- To facilitate recovery after a process failure and restore the system to a consistent state, two types of log storage are maintained:
  - **Volatile log:** It takes short time to access but lost if processor crash. Thecontents of volatile log are moved to stable log periodically.
  - **Stable log:** longer time to access but remained if crashed.

Asynchronous checkpointing

- After executing an event, a processor records a triplet (s, m, msg_sent) in its volatile storage.
  - s:state of the processor before the event
  - m: message
  - msgs_sent: set of messages that were sent by the processor during the event.
- A local checkpoint at a processor consists of the record of an event occurring at the processor and it is taken without any synchronization with other processors.
- Periodically, a processor independently saves the contents of the volatile log in the stable storage and clears the volatile log.
- This operation is equivalent to taking a local checkpoint.

Recovery Algorithm

The data structures followed in the algorithm are:

$RCVD_{i\rightarrow j}(CkPt_i)$ This represents the number of messages received by processor pi from processor pj, from the beginning of the computation until the checkpoint $CkPt_i$.

$$SENT_{i\rightarrow j}(CkPt_i)$$

This represents the number of messages sent by processor pi to processor pj, from the beginning of the computation until the checkpoint $CkPt_i$.

- The main idea of the algorithm is to find a set of consistent checkpoints, from theset of checkpoints.
- This is done based on the number of messages sent and received.
- Recovery may involve multiple iterations of roll backs by processors.
- Whenever a processor rolls back, it is necessary for all other processors to find outif any message sent by the rolled back processor has become an orphan message.
- The orphan messages are identified by comparing the number of messages sent to and received from neighboring processors.
- When a processor restarts after a failure, it broadcasts a ROLLBACK message thatit has failed.
- The recovery algorithm at a processor is initiated when it restarts after a failure or when it learns of a failure at another processor.
- Because of the broadcast of ROLLBACK messages, the recovery algorithm is initiated at all processors.

Procedure RollBack_Recovery: processor pi executes the following:STEP (a)

if processor pi is recovering after a failure **then**

$C_k\ Pt_i$:= latest event logged in the stable storage

else

$C_k\ Pt_i$:= latest event that look place in p_i {The latest event at pi can be either instable or in volatile storage}

end if

STEP(b)

for k=1 to N {N is the number of processors in the system} **do**

for each neighboring processor p_j **do**

compute $SENT_{i\rightarrow j}(C_k\ Pt_i)$

send a ROLLBACK(i, $SENT_{i\rightarrow j}(C_k\ Pt_i)$) message to p_j

end for

for every ROLLBACK(j,c) message received from a neighbor j **do**

if RCVD $_{i\rightarrow j}$ (C_k Pt_i) > c {Implies the presence of orphan message}

then

find the latest event e such that RCVD $_{i\rightarrow j}$ (e) = c {Such an event e may be in the volatile storage or stable storage}

C_k Pt_i := e

end if

end for

end for {for k}

Fig : Algorithm for Asynchronous Check pointing and Recovery (Juang- Venkatesan)

- The rollback starts at the failed processor and slowly diffuses into the entire systemthrough ROLLBACK messages.
- During the kth iteration (k != 1), a processor pi does the following:

(i) based on the state $CkPt_i$ it was rolled back in the (k − 1)th iteration, it computes $SENT_{i\rightarrow j}$ ($CkPt_i$) for each neighbor pj and sends this value in a ROLLBACK message to that neighbor

(ii) p_i waits for and processes ROLLBACK messages that it receives from its neighbors in kth iteration and determines a new recovery point $CkPt_i$ for p_i based on information in these messages.

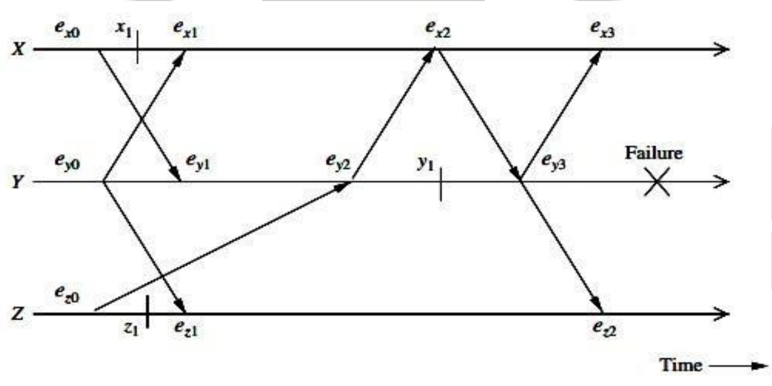

Fig : Asynchronous Checkpointing And Recovery

At the end of each iteration, at least one processor will rollback to its final recovery point, unless the current recovery points are already consistent.