

⑩ Chandy-Lamport algorithm for consistent state capture in distributed database

Ans → The Chandy-Lamport Algorithm is a distributed snapshot algorithm used to record a consistent global state of a distributed system without stopping execution.

→ It is mainly used for deadlock detection, checkpointing, recovery mechanisms, debugging distributed databases.

→ A global state consists of :-

• Local state of each process

• State of communication channels

→ A state is consistent if a message is recorded as received, then it must also be recorded as sent.

→ The algorithm assumes:-

• Process communicates via message passing

• Channels are FIFO

• No shared memory

• System continues execution during snapshot

• No message loss

The algorithm uses a special control message called a MARKER

to initiate and propagate the snapshot

It works without stopping the system.

◆ Steps of the Algorithm

• Step 1: Snapshot Initiation

One process (say P1):

→ Record its own local state

→ Sends a MARKER message on all outgoing channels.

• Step 2: When a process receives a MARKER

There are two cases :-

Case 1 : Marker received for the first time

The process

→ Records its local states

→ Records channel state as empty for the channel from which

marker arrived.

- Sends marker to all outgoing channels
- Starts recording messages from other incoming channels.

Case 2 : Marker received again (on another channel)

The process :

- stops recording messages on that channel
- All messages received b/w first & marker and this marker are saved as channel state.
- Messages that are sent before sender's snapshot and received after receiver's snapshot are considered in-transit messages and are recorded as channel state.

Example : Consider 3 process :- $P_1 \rightarrow P_2 \rightarrow P_3$

If P_1 starts snapshot

- P_1 records state and sends marker
- P_2 records state when it receives marker. P_3 does some
- Any messages sent before marker but received after snapshot is recorded as in-transit.

thus a global state is formed.

Properties

Non-blocking, no need to stop processes, produces consistent global snapshot,
works in asynchronous systems

Time complexity : $O(M+N)$, where N = number of processes
 M = number of channels.

Applications

- Global clock pointing, Recovery after failure, deadlock detection,
- Distributed transaction debugging, consistency verification

Advantages

- Simple and elegant
- No global clock needed
- works during normal execution.

Limitations

- Requires FIFO channels
- Not suitable for non-FIFO systems (without modification)
- Snapshot does not represent real time global state too.