DISTRIBUTED COMPUTING

Write a detail note on Various primtives for Distributed computing

↳ Blocking

↳ Non-blocking, Synchronous

↳ Asynchronous Primtives

↳ Message Send Communication primtives is denoted by Send() and receive Communication primtives denoted by Receiver().

↳ Message passing primtive Commands.

- SEND (msg, dest)
- Receive (src, buffer)

↳ Send primtives use two options for Sending data

- Buffered
- UnBuffered

↳ In buffered option, user data is copied in the kernel buffer. In buffereol option, the data gets copied directly from the user buffer onto the network.

↳ The communication of message between two processes implies some level of synchronization between the two processes.

↳ Three combinations are possible using blocking & non-blocking.

- Blocking send, blocking receive
- Nonblocking send, blocking receive
- Nonblocking send, Non-blocking receive
- Nonblocking send, non-blocking receive.

Blocking send Blocking Receive

↳ Both sender and receiver are blocked until the message is delivered.

Nonblocking Send, Blocking Receive

↳ Sender may Continue on, the receiver is blocked until the requested message arrives.

Nonblocking Send, Nonblocking Receive

↳ Sending process Sends the message and resumes the operation.

2) How does local state contributed to the global state in Distributed System.

local state

↳ Each process in a distributed System maintains its own local state

↳ local state includes

- Process Variable, program counter, memory values.

- messages sent /received

- Internal resources

Global State

↳ A global state is the combination of

- local state of all processes.
- States of communication channels (message in transit)

Contribution of local state

↳ Global state = union of all local state + Channel State

Ex: P1 local state - Balance = 100

P2 local state = Balance = 200

Message in transit = "Transfer 50"

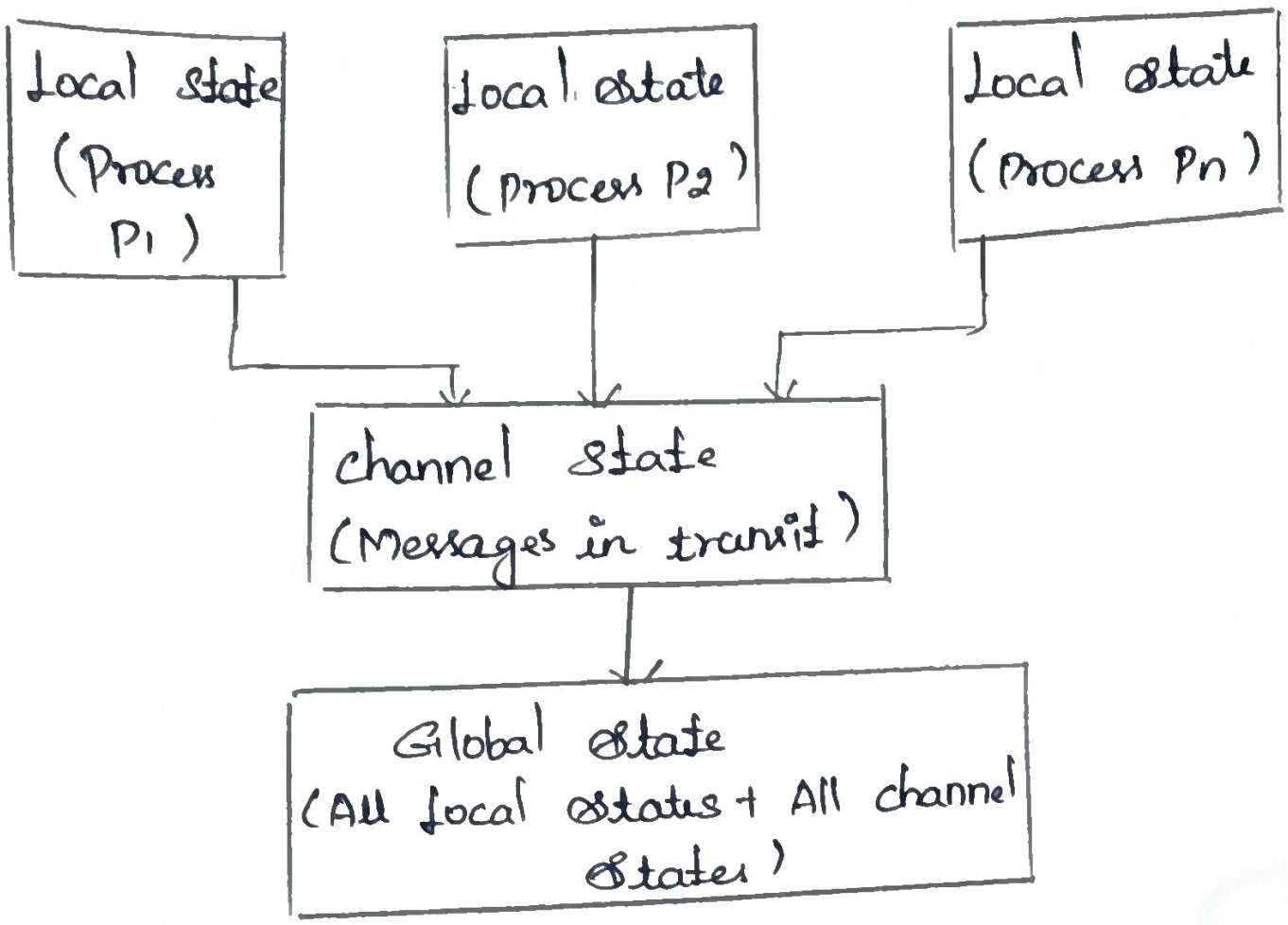
Global state = {100, 200, Transfer 50 message}

Consistency Requirement

↳ local states must be combined in a way that respects causality

Importance

- checkpointing & Recovery
- Deadlock & Termination Detection
- Debugging & Monitoring.



3) Explain Snapshot algorithm

Definition and purposes

↳ A snapshot algorithm is a distributed algorithm used to record a consistent global state of a DS.

↳ checkpointing, rollback recovery

↳ Debugging

↳ Deadlock

↳ detecting termination

Chandy - lamport Algorithm

↳ Records a set of proces and channel states such the combination is a consistent global state.

↳ Communication channels assumed to by FIFO

Assumptions

↳ No failure, all messages arrive intact, exactly one.

↳ communication channels are unidirectional and FIFO- ordered

↳ There is a communication channel between each pair of processes

↳ Snapshot does not interface the Snapshot (send "Marker")

Algorithm

1↳ Initiator process P_0 records its state locally

2↳ Marker Sending rule for process P_i:

- After P_i has recorded its state, for each outgoing channel $chij$ P_i: Sends one message over $chij$

3↳ Marker receiving rule for process P_i:

- Process P_i on receipt of a marker over channel $chji$

Properties

- Consistency
- Non-Intruiveness
- finite time

Uses

- checkpointing
- Global predicate detection
- Monitoring /Debugging

4) How does clock synchronized physically? Explain it

Synchronize

↳ Each computer has its own hardware clock (oscillator / quartz crystal)

↳ clocks may drift over time ~~data~~ due to differences in speed or temperature

How physical clock are Synchronized

Using External Standard clock Sources

↳ The most common way is to Synchronize all nodes to a Standard reference clock.

Ex: UTC (coordinated Universal Time)

clock Synchronization Protocols

↳ Because clocks can drift, distributed systems use protocols to resynchronize

a) Cristian's Algorithm

↳ one node contacts a time server that has access to UTC.

Steps: ↳ client sends a request to Server.

↳ Server replies With its current time.

↳ client adjusts its clock = Server time + (1/2 round-trip delay).

Berkeley Algorithm

↳ No external UTC Server required.

↳ one node acts as master.

↳ Good for internal synchronization (clusters)

Network Time Protocol (NTP)

↳ Internet Standard protocol.

↳ Uses hierarchical time Servers Worldwide connected to atomic clocks & GPS

Hardware Solutions

↳ GPS receiver in each machine - directly get UTC from Satellites

5) Illustrate Causal order and total order With example.

↳ Since there is no global clock, events in Ds are ordered using logical rules. Two common types

- Causal order
- total order

Causal order

↳ Based on happened - before relation (→) introduced by leslie lamport.

↳ If event a happens before event b in the same process, or if a sends a message that is received by b, then a → b

↳ otherwise, events are Concurrent.

eg

↳ Process P_1 sends a msg m1 to P_2

↳ P_2 receives m1 and then sends m2 to P_3

↳ These events are causally related.

Total order

↳ The total order ensures that all events in the DS are ordered in the same sequence, even if they are independent.

↳ This removes concurrency.

Ex:

↳ Suppose three process send messages

- P_1 sends m_1
- P_2 sends m_2
- P_3 sends m_3

↳ In causal order, m_1 and m_2 may be concurrent.

↳ In total order, all processes must agree on one sequence.