**JITENDRA NAKRA**

**2020mt13005**

**SSZG526**

**Snapshot algorithms for Non-FIFO channels**

FIFO system ensures that all messages sent after a marker on a channel will be delivered after the marker. This ensures that condition C2 is satisfied in the recorded snapshot if LSi, LSj , and SCij are recorded as described in the Chandy–Lamport algorithm. **In a non-FIFO system**, the problem of global snapshot recording is complicated because a marker cannot be used to delineate messages into those to be recorded in the global state from those not to be recorded in the global state. In such systems, different techniques have to be used to ensure that a recorded global state satisfies.

## Overview - Lai and Yang

The non-FIFO algorithms by Lai and Yang use message piggybacking to distinguish computation messages sent after the marker from those sent before the marker. The Lai–Yang algorithm fulfills this role of a marker in a non-FIFO system

by using a coloring scheme on computation messages.

## Objective:

The objective of this simulation program is to understand working of Lai-Yang Algorithm in distributed system. Simulation automatically captures the local and global snapshot post manual entry of Trigger point for any Process.

## Platform:

* Windows operating System(Windows 10)
* Java Programming language

## Assumptions:-

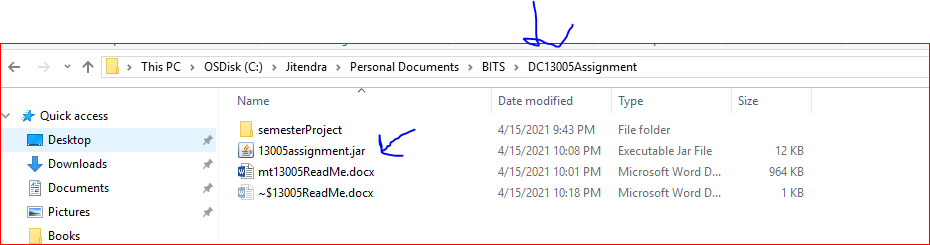
* Total number of processes – 2 ONLY
* Timestamps start from T1 onwards
* User will have to enter the values carefully one by one to generate the desired outcome.
* User shall specify the timestamp and Process number which will start the snapshot
* Reset of algorithm or re capturing of any state is out of scope.
* Valid data entry for Process ,Sender, Destination ,Timeslots - Any irregular special characters apart from Integer will lead to custom exception created in program referring as “ Custom Lai-Yang exception “



**Glossary in Output screen**

* Under Process summary : ProcessP1t1 – implies – >Process p1 at timestamp t1
* Under Channel summary: P1C12t1 implies – > Process P1 at timestamp t1 on channel 1 to 2.

**Project Code view:**



**How to run the JAR :** Execution filename = LaiYangExecution.java

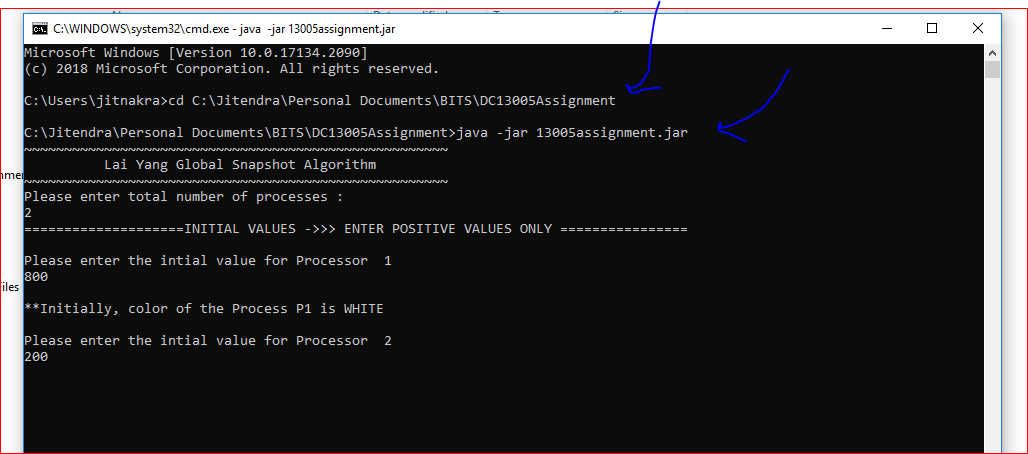
Step 1: Open Command Prompt or press windows + R

Step 2 : Go to the path where you have to kept the source code of project

Step 3: Locate the JAR with name “ 13005assignment.jar”

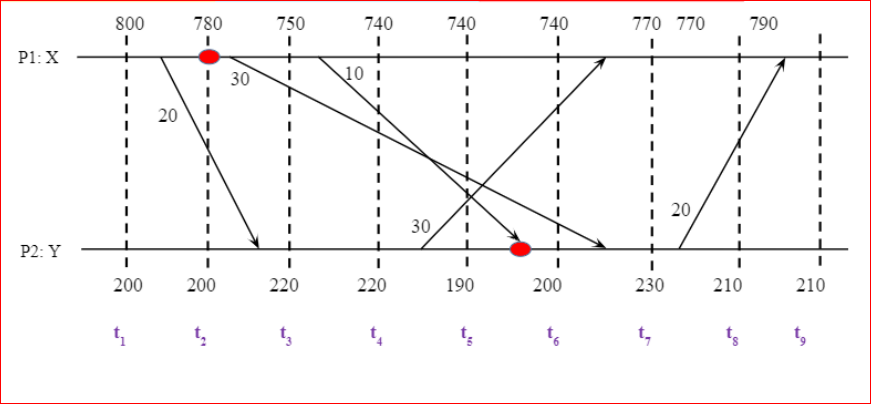
Step 4 : Type Java –jar 13005assignment.jar

Step 4 : Enter the inputs mentioned below in the document to simulate the algorithm



**Case Study 1 : Non-FIFO Implementation:**

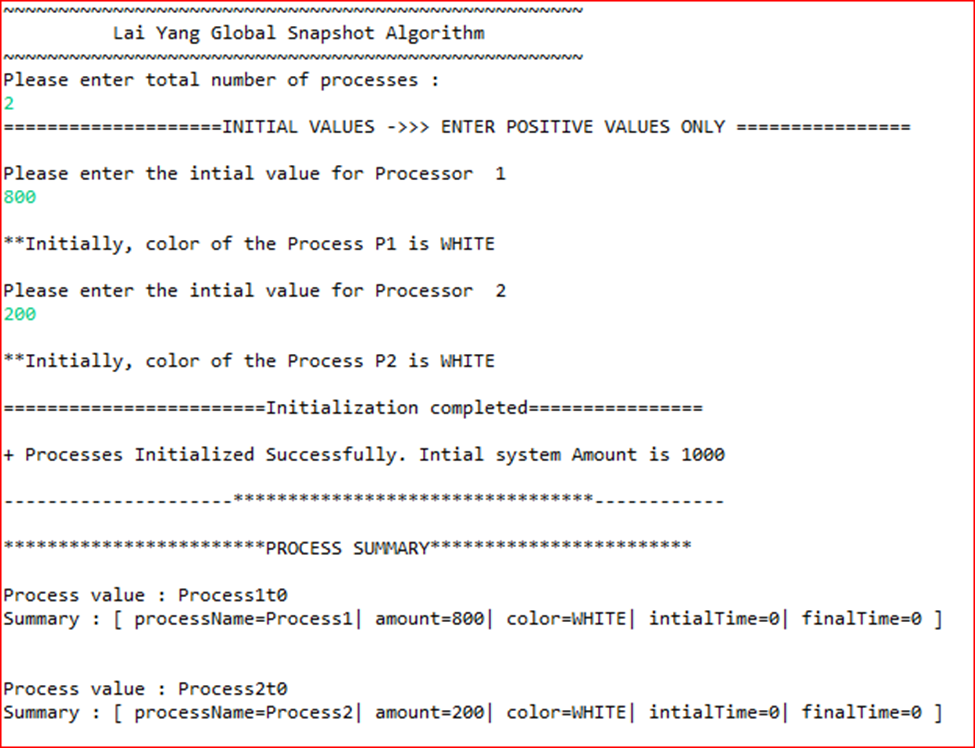
The below document will show a Step by Step guide for simulation which has been done as per the below diagram :



**Input from User :**

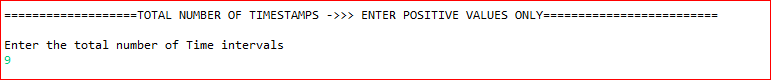
1. Process Involved : 2 [P1,P2]
2. Initiation of Process : P1 -> 800 and P2-> 200 [t0]

**Output Screen :**



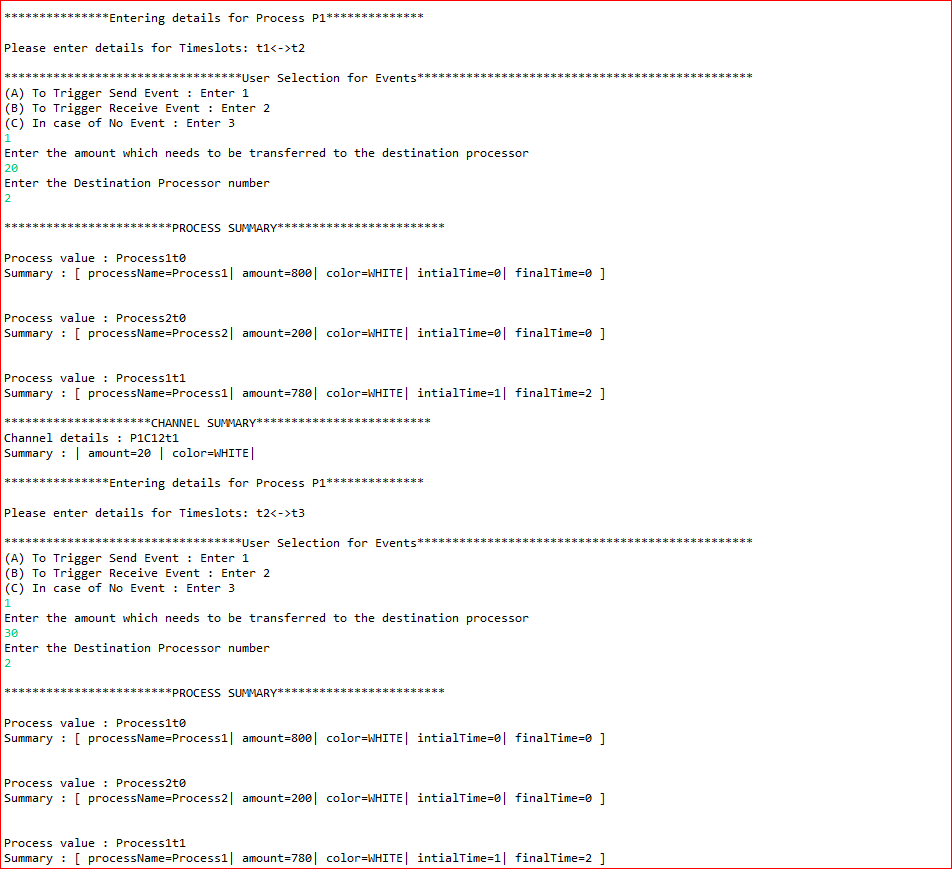
**Input from User :** Time slots : 9

**Output Screen :**



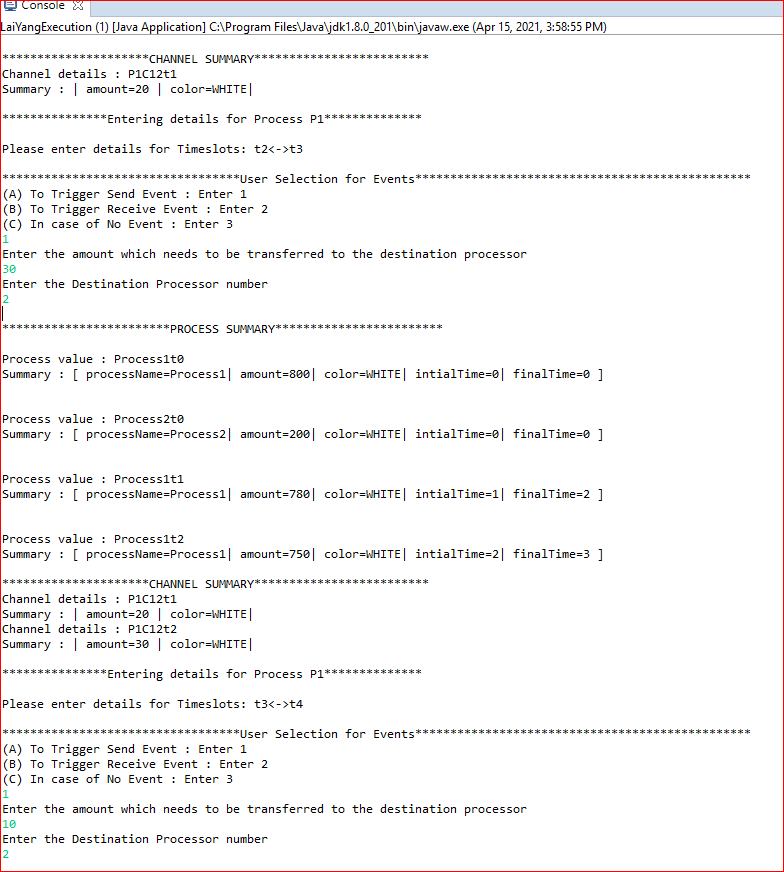
**Input from User** : As per diagram enter the following data by choosing 1/2/3

Process P1 -> 3 Send Events [20 ,30,10] || 2 Receive Events [30,20]



**Input from User** : As per diagram keep enter the data by choosing 1/2/3 till timeslots entered

**Output Screen :**



\*\*Skipping rest of the entry snapshots for Process P1 for remainder of time intervals & adding the final

Process P1 Summary view.

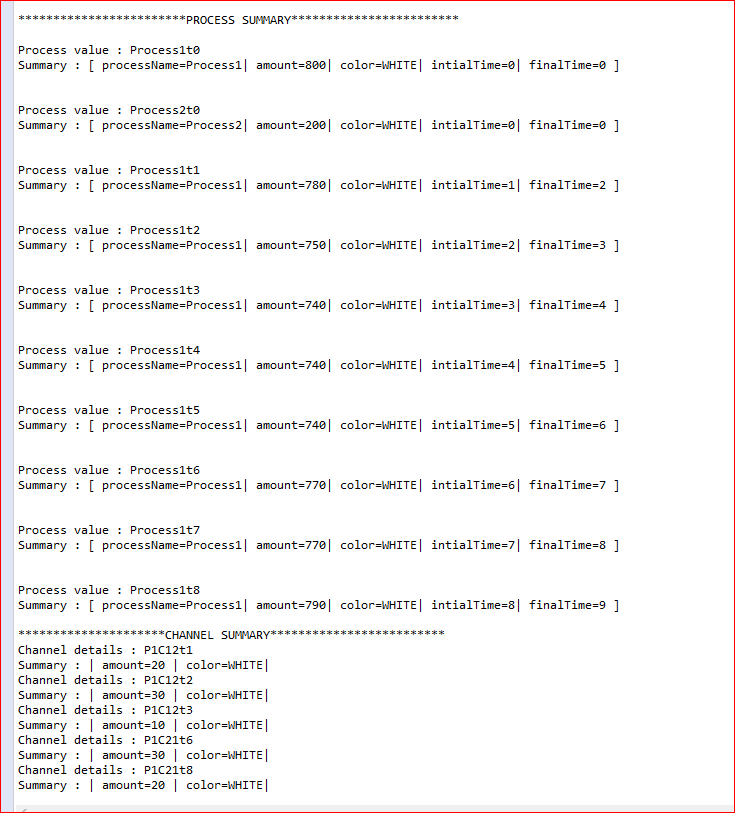


Figure 2 last Event Captured on Channel by Process p1

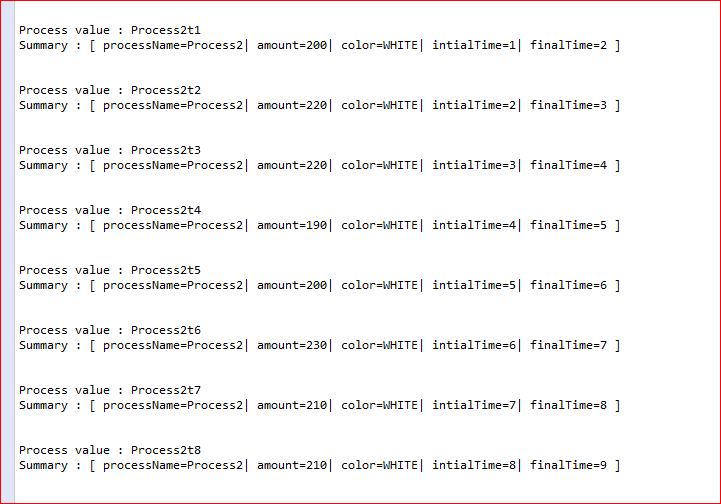
Figure 1 Last Entry details

**Input from User** : As per diagram enter the following data by choosing 1/2/3

Process P2 -> 2 Send Events [30,20] || 3 Receive Events [20,10,30]

Skipping rest of the entry snapshots for Process P1 for remainder of time intervals & adding the final

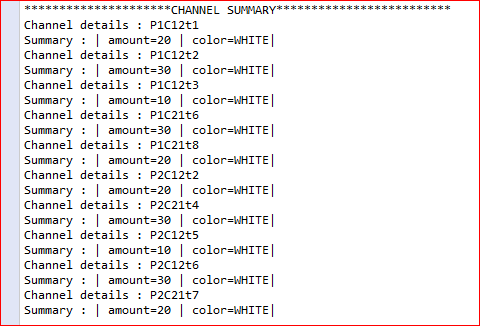
Process P2 Summary view.



In case of No event registered , Amount remains same only time intervals increases

History Maintained: Process P1 and P2 will maintain a history of all white message sent or receive along each channel.

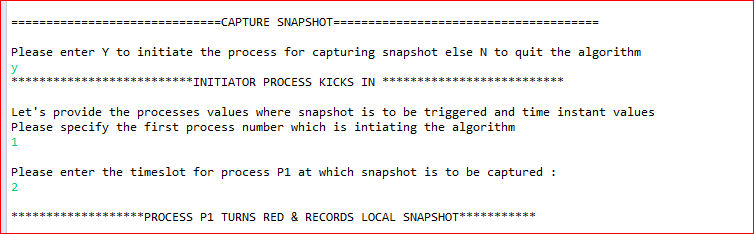
**Output Screen :**



Initiator process : Responsible for collecting the Global Snapshot.

**Input from User** Enter “Y” to Start the algorithm and provide the triggering point for process to turn red details ex- Process 1 ,2 timeslot

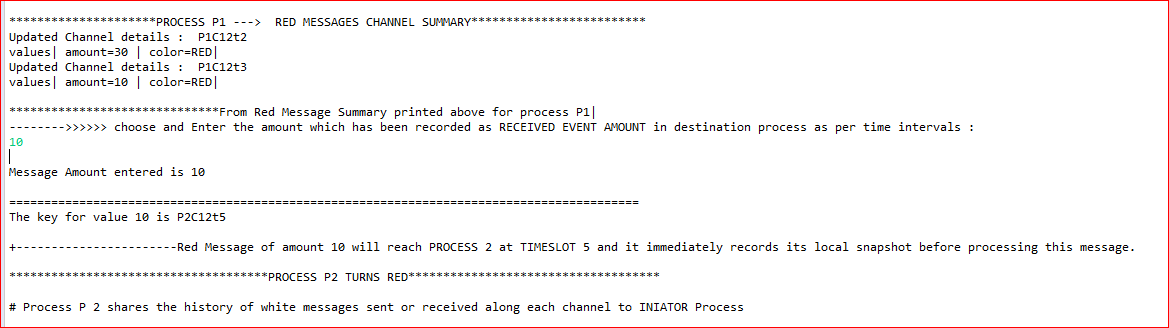
Triggering Point : P1 at timestamp t2 & turn itself into Red and captures local state and share above history to initiator process .



Proof of NON –FIFO Implementation:

**Input from User** Red Message on P1 : Choose amount as per values entered above in Process P2 Receive event entered in earlier time interval.

Ex- in this case 30 started first than 10 but being in a NON-FIFO Model 10 reaches P2 at t5 & 30 after t5 so 10 will be the RED message which will make P2 turn Red.



v

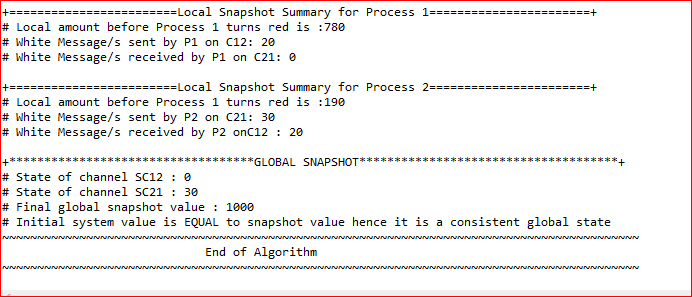


\*\*10(Red) will make the Process P2 (Initially state white) capture its local snapshot before processing the message and turn Red.

Global Snapshot :

Red Circle indicates -> User wants to take snapshot at P1 ,t2 and P2 will be auto triggered based on Red messages received from Process P1.

Summary view for Local Snapshots and Channel State & Global Snapshot calculation :



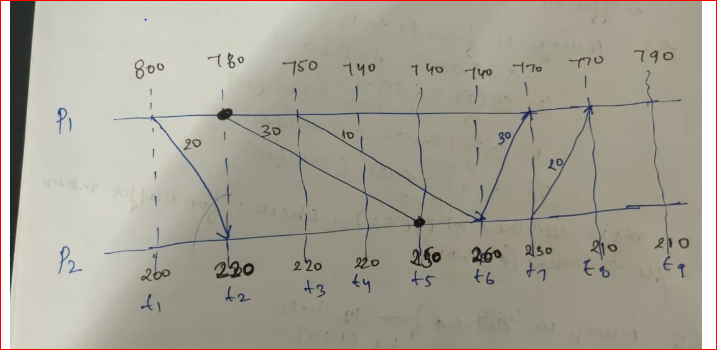
Channel State calculation : Initiator process evaluates transit(LSi, LSj) to compute the state of a channel Cij as: SCij = { messages sent on that channel since the last snapshot)−(messages received on that channel since the last snapshot)

= {mij | send(mij) ∈ LSi} − {mij | rec(mij) ∈ LSj}

|  |  |  |
| --- | --- | --- |
| Channel | Sent | Receive |
| C21 | P2 =30 | P1=Not applicable | P1 =0 | P2=Not applicable |
| C12 | P1=20 | P1=Not applicable | P2= 20 | P1=Not applicable |

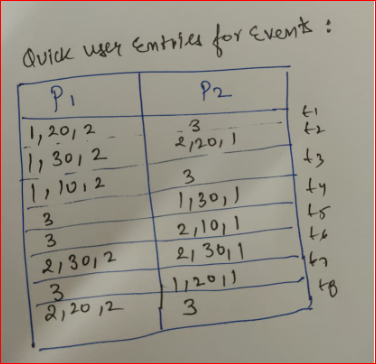
Consistent state : Initial Amount captured and final amount captured are compared if conservation of both processes are equal it is referred as Consistent else Inconsistent state.

**Case Study 2 : FIFO Implementation:**



**Input from User :**

**Follow the steps as above to enter the values for each Processes or refer to Quick User Guide :**

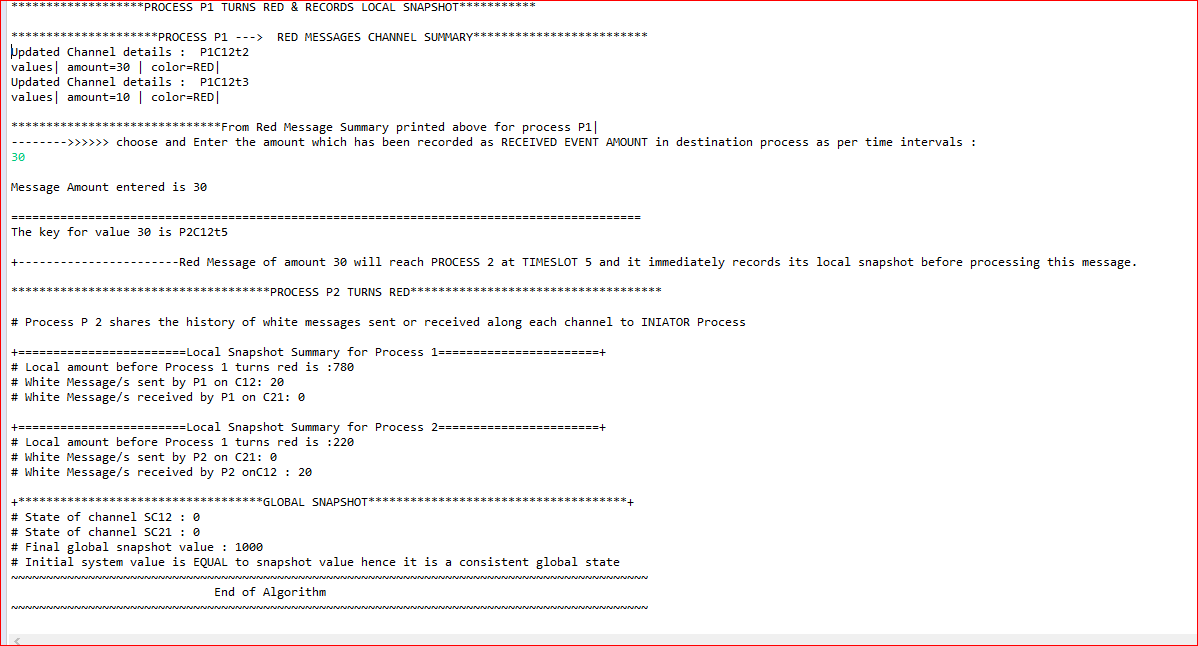


Difference Point – FIFO Let’s say Trigger happens at t2 –P1 for amount 30 ,since it happens First before 10 Whatever time it will reach to P2 , P2 will record its state accordingly.

**Attaching the difference Output Screen :**

Summary view for Local Snapshots and Channel State & Global Snapshot calculation :

v



-------------------------------------------------XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX-------------------------------------------------------------