SFU SIMON FRASER UNIVERSITY
ENGAGING THE WORLD

CMPT 431 Distributed Systems

Fall 2019

Global State & Snapshot Recording

https://www.cs.sfu.ca/~keval/teaching/cmpt431/fall19/

Instructor: Keval Vora

Reading

- [DC] Chapter 4
  - Upto 4.3

Introduction

- Record the global state of a distributed system on-the-fly

- No global shared memory
- No global clock
- Unpredictable message delays

Consistent Global State

- Global state is a collection of the local states of all processes, and the states of all channels

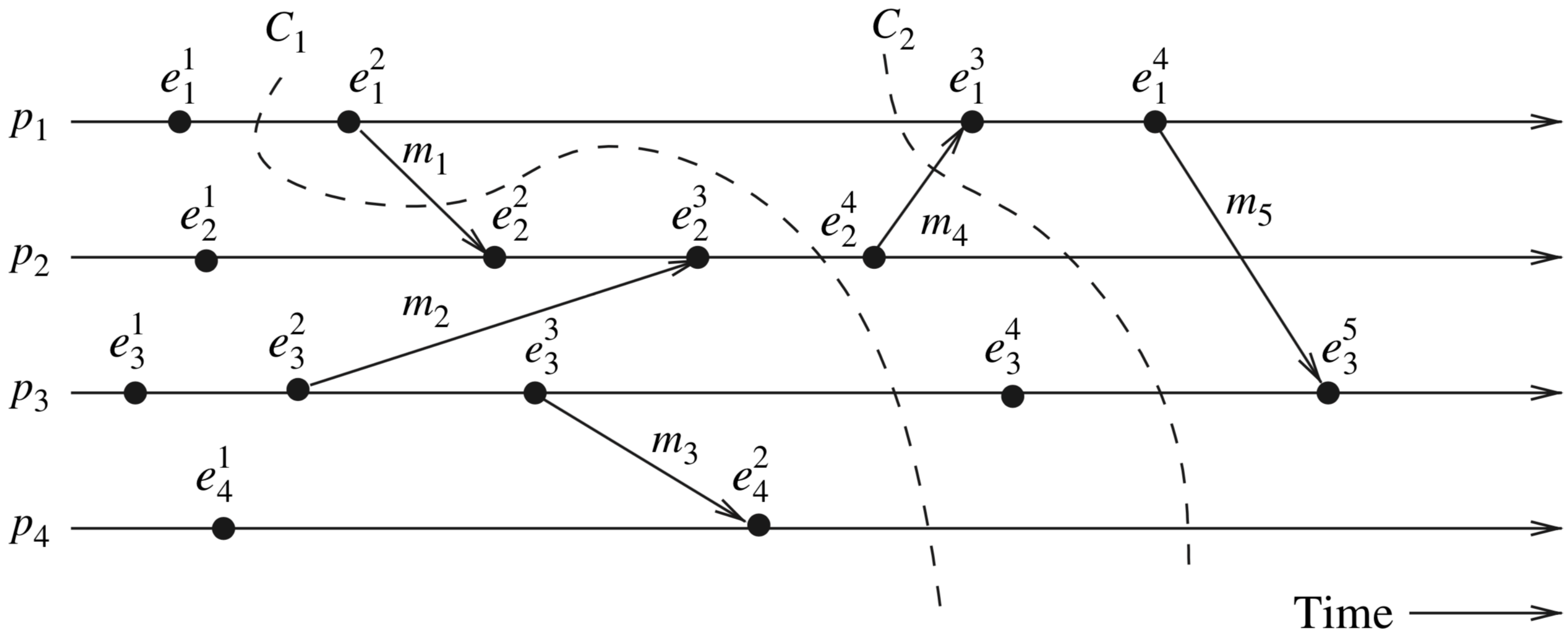
$$GS = \{ \cup_i LS_i , \cup_{i,j} SC_{ij} \}$$

- Global state GS is a consistent global state iff:

$$C1: \ send(m_{ij}) \in LS_i \Rightarrow m_{ij} \in SC_{ij} \ \oplus \ rec(m_{ij}) \in LS_j$$

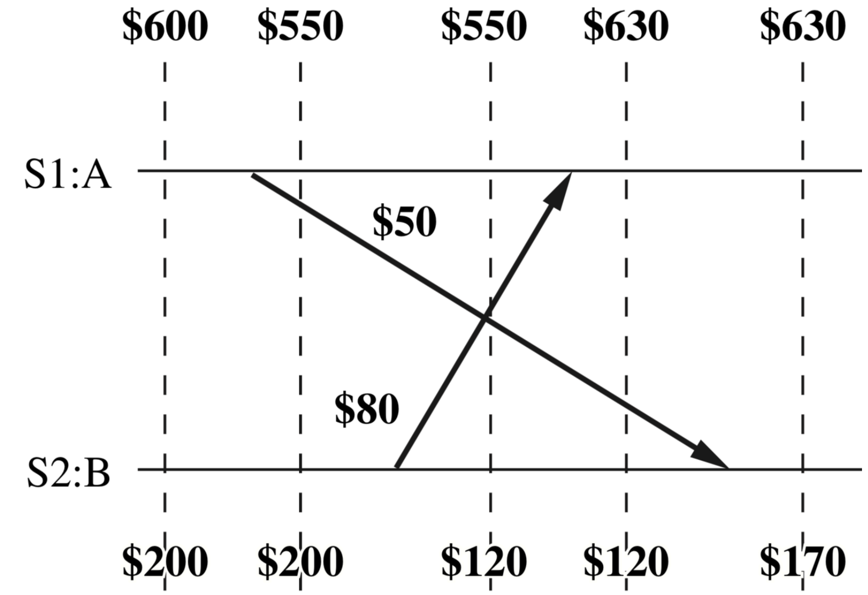
$$C2: \ send(m_{ij}) \notin LS_i \Rightarrow m_{ij} \notin SC_{ij} \wedge rec(m_{ij}) \notin LS_j$$

Consistent Global State



Consistent Global State

Sum: $800



	t_0	t_1	t_2	t_3	t_4
C_{12}	$0	$50	$50	$50	$0
C_{21}	$0	$0	$80	$0	$0

Recording a Global State

- How to distinguish between the messages to be recorded in the snapshot from those not to be recorded?
    - Any message that is sent by a process before recording its snapshot, must be recorded in the global snapshot (from C1)
    - Any message that is sent by a process after recording its snapshot, must not be recorded in the global snapshot (from C2)

- How to determine the instant when a process should take its snapshot?
    - Process p_j must record its snapshot before processing a message m_{ij} that was sent by process p_i after recording its snapshot

Chandy-Lamport Algorithm

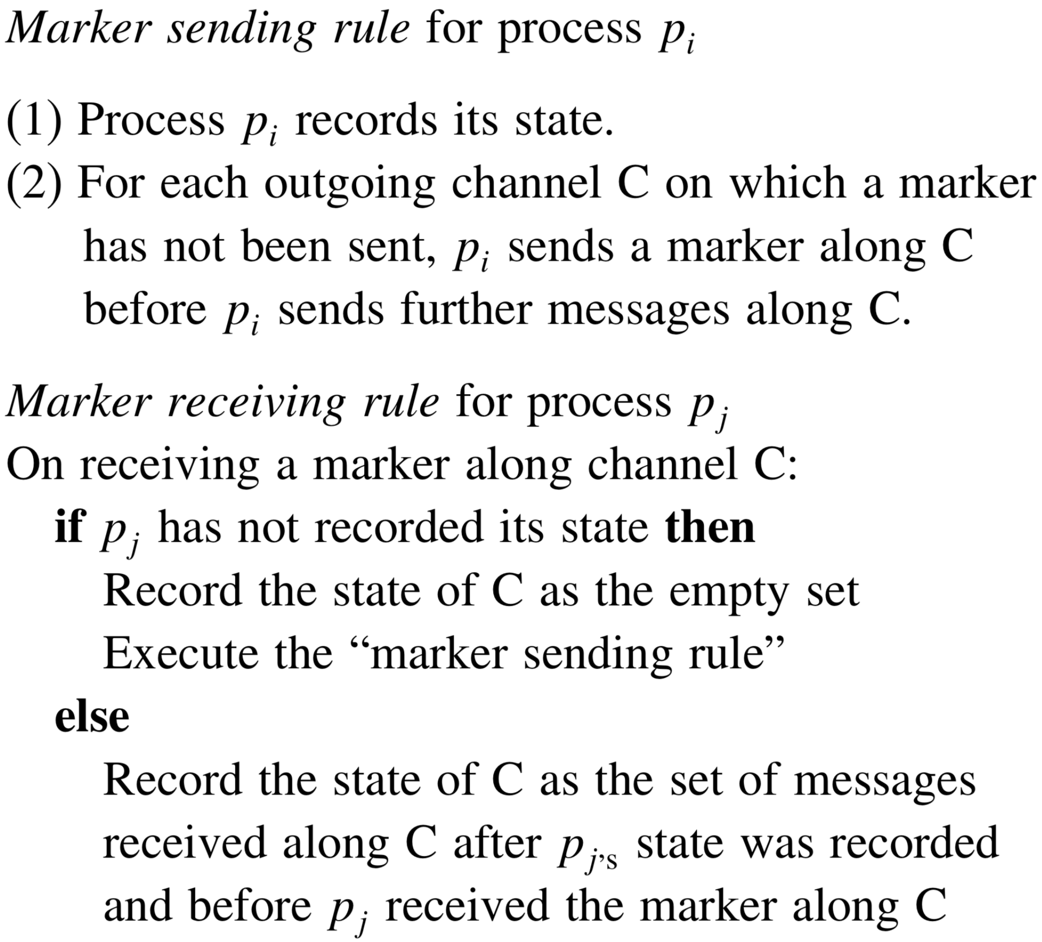
- Distributed algorithm to record global snapshot

- Marker: Control message to separate messages that should be included in the snapshot
- After a process records its snapshot, it sends a marker to all outgoing channels before sending other messages

- A process must record its snapshot no later than when it receives a marker on any of its incoming channels

Chandy-Lamport Algorithm

- Initiated by any process by running ‘Marker Sending Rule’



```
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 $p_{j\text{'s}}$ state was recorded
      and before $p_j$ received the marker along C
```

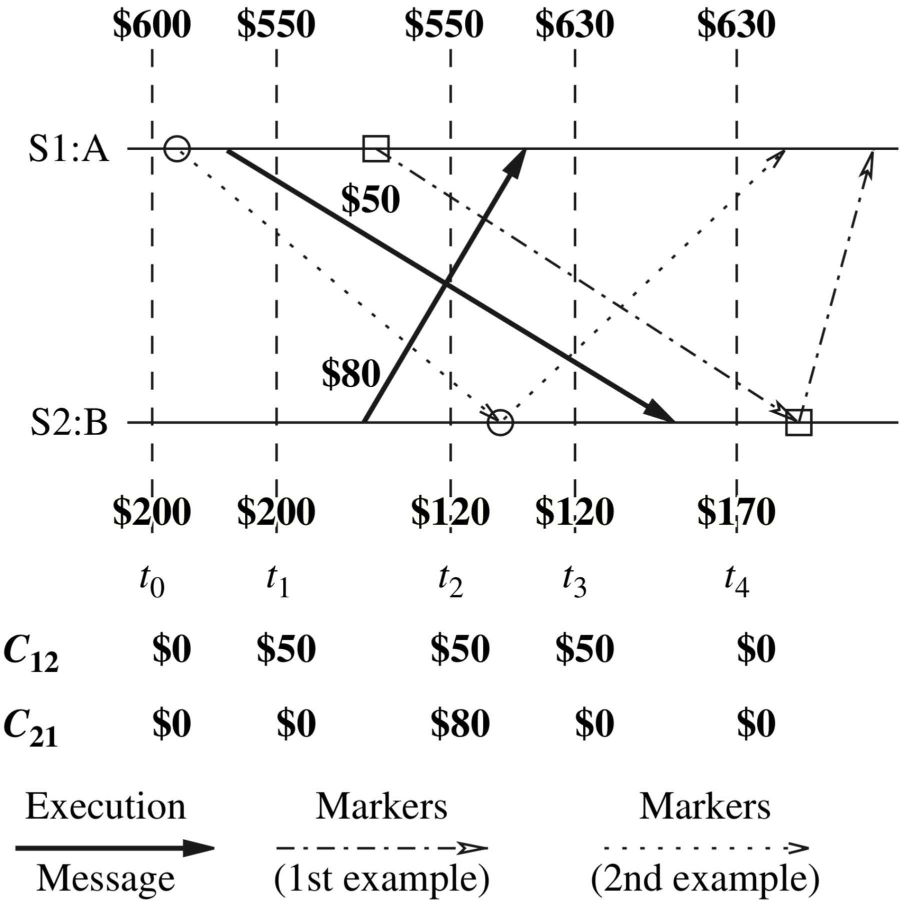
Chandy-Lamport Algorithm: Correctness

- How to reason about correctness?
  - Does the algorithm satisfy C1 and C2?
- No message sent after the marker on that channel is recorded in the channel state (assumption: FIFO channel)
  - C2 is satisfied
- When a process p_j receives message m_{ij} that precedes the marker on channel C_{ij} :
  - If p_j has not taken its snapshot yet, it includes m_{ij} in its recorded snapshot
  - Otherwise, it records m_{ij} in the state of the channel C_{ij}
  - C1 is satisfied

Chandy-Lamport Algorithm

- 1st example:
  - A = $550, B = $170
  - C_{12} = $0, C_{21} = $80
- 2nd example:
  - A = $600, B = $120
  - C12 = $0, C_{21} = $80

Any issues?



Chandy-Lamport Algorithm

- The recorded global state may not correspond to any of the global states that occurred during computation
- Reason: process can change its state asynchronously before the markers it sent are received by other sites and the other sites record their states

- The recorded state is a valid state that could happen
  - It retains "stable" properties (e.g., sum = $800, no deadlock, etc.)
- Useful in detecting stable properties