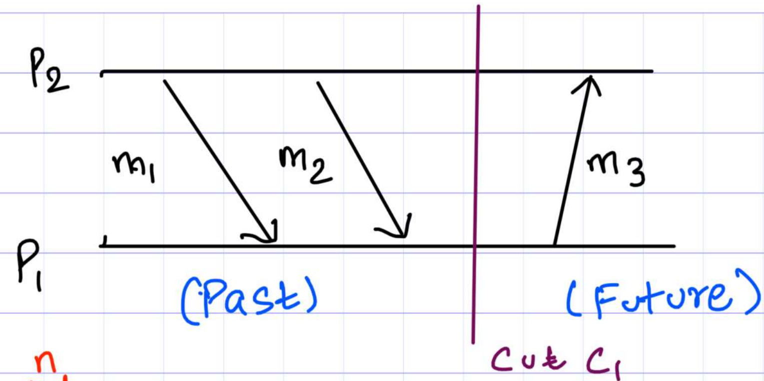
Global States



Global state = $\bigcup_{i}^{n}$ Local states$_i$

[Cut is a graphical line splitting the time-space graph into 2 parts]

Here,

$$GS = LS_1 \cup LS_2 .$$

$LS_1 = \{send(m_1) , send(m_2)\}$

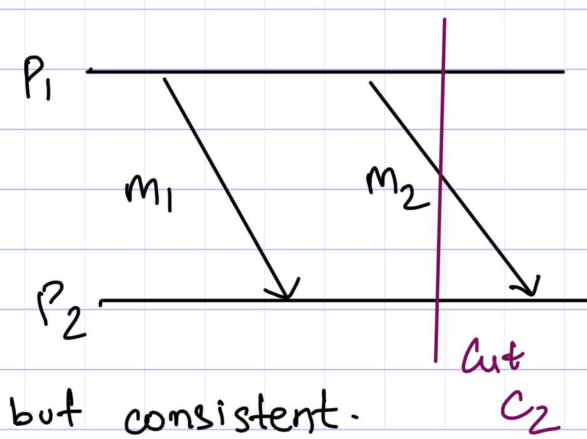
$LS_2 = \{receive(m_1) , receive(m_2)\}$

$GS = \left\{ \begin{array}{l} send(m_1), send(m_2), \\ receive(m_1) , receive(m_2) \end{array} \right\}$ <u>strongly consistent</u>

Strongly consistent global state

1) $\forall$ send event in GS, corresponding receive event will be present

2) & no(send) = no(receive)



$LS_1 = \{send(m_1), send(m_2)\}$
$LS_2 = \{recv(m_1)\}$

$GS = \{send(m_1), send(m_2), recv(m_1)\}$

<u>not strongly consistent</u>, but <u>consistent</u>.

m_2 is in transit because only send(m_2) is present.

Consistent Global state.

$\forall$ receive event in GS, corresponding send events are present.

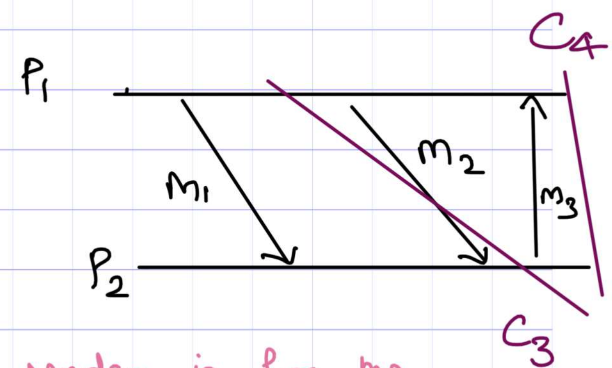
Transit would be there in consistent, but not in strongly consistent.

$LS_1 = \{send(m_1)\}$

$LS_2 = \{rec(m_1), rec(m_2)\}$

$GS_1 = LS_1 \cup LS_2$

Inconsistent global state.



P2 cannot know who the sender is for m2 from the global state.

$GS_2 = \{ send(m_2), send(m_3), rec(m_3)\}$

$GS_1 \longrightarrow GS_2$ (causal ordering)

But causality fails here because in GS_1 we have a recv(m_2) before the send(m_2) in GS_2.

But $GS_1 \rightarrow GS_2 \Rightarrow$ causal ordering is inconsistent.

Maintaining strongly consistent states is practically impossible because of performance lag.

(Need to freeze, snapshot, resume operation each time event occurs)