Casual order:

* casual order in distributed systems ensures that events that are causally related (ie one event depends on another) are executed in same order across all processes.

* multiple senders send messages to multiple receivers.

* The ordered message delivery ensures that all msg are delivered to all receivers in an acceptable order.

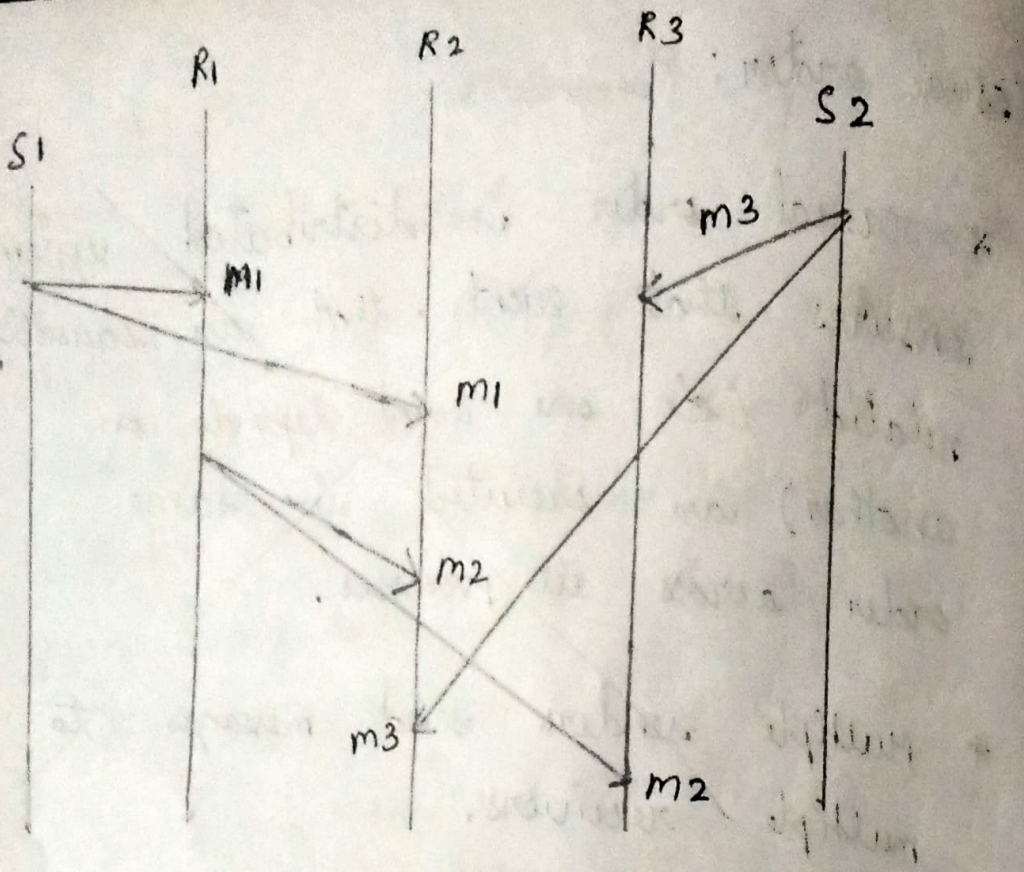
* It describes the causal relationship between a message send event and a message receive event.

* ~~Eg:~~ If ~~send~~

Eg:- Rules:

i) If e_i = send (m) & e_j = receive (m), then $e_i \rightarrow e_j$.

ii) If $e_i \rightarrow e_j \rightarrow e_k$, then $e_i \rightarrow e_k$



> If send (m1) happens befor send (m2), then every receiver of both the msg must receive m1 before m2.

> If two send events are not causally related, then the msg can be delivered in any order.

Adv :

* Less ovrhead, making it Fastis
* Easis to scale with more nodes
* Simples to implement and manage.

Di-Adv:

* can lead to inconsistencies across nodes
* Not suitable for applications needing strict order
* Harder to debug issues related to order.

Total order:

* Total order in distributed systems ensures that all events, process in the system agree on the exact same order of events.
* It is also known as consistent ordering.
* It ensures that all messages are delivered in the same order to all the receivers.
* Msgs will ~~too~~ be received in the same order, regardless of their timestamps.

Algorithm:

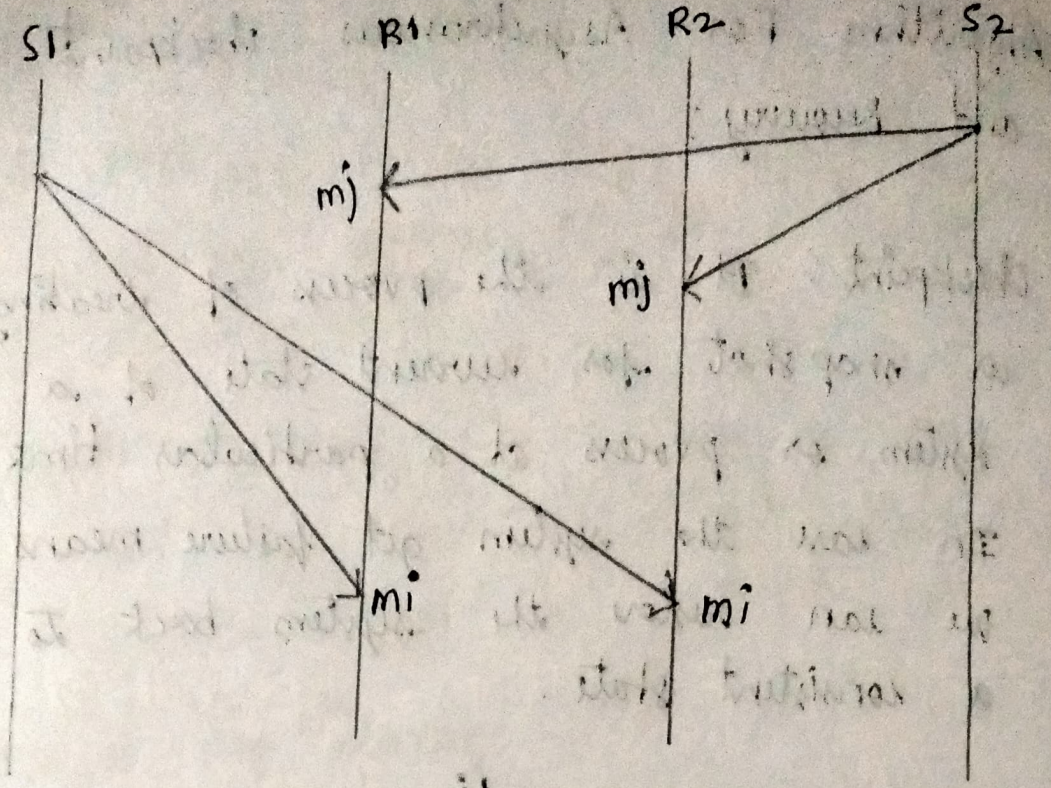
Sender: sends

1. process multicast the msg M with an unique tag and a time stamp to the group members.
2. The group members respond with a temporary proposal for a revised timestamp for that msg M.
3. The process multicasts the final timestamp to the group.

Receiver:

1. Receiver receives the msg with a tintative timestamp.
2. Receives sends the revised timestamp to the server.
3. Receiver receives the final timestamp.

Eg:



Adv:

* Ensures all nodes have same order of events.
* Suitable for application needing strict consistency.
* Easier to trace & resolve issues.

Dis-Adv:

* leading to slower performance
* Harder to scale with many nodes.
* More difficult to set up & maintain.