

MYE017 Distributed Systems

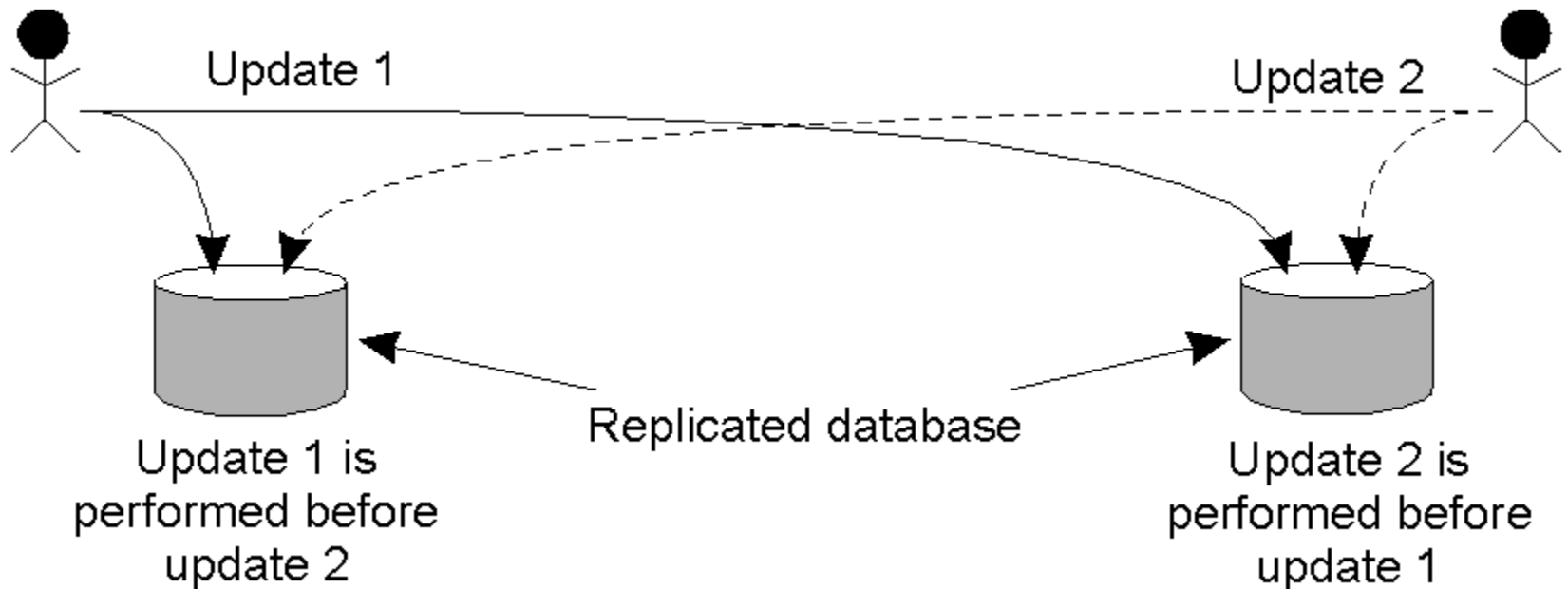
Kostas Magoutis

magoutis@cse.uoi.gr

<http://www.cse.uoi.gr/~magoutis>

Total order multicasting

Updating a replicated database may leave it in an inconsistent state.



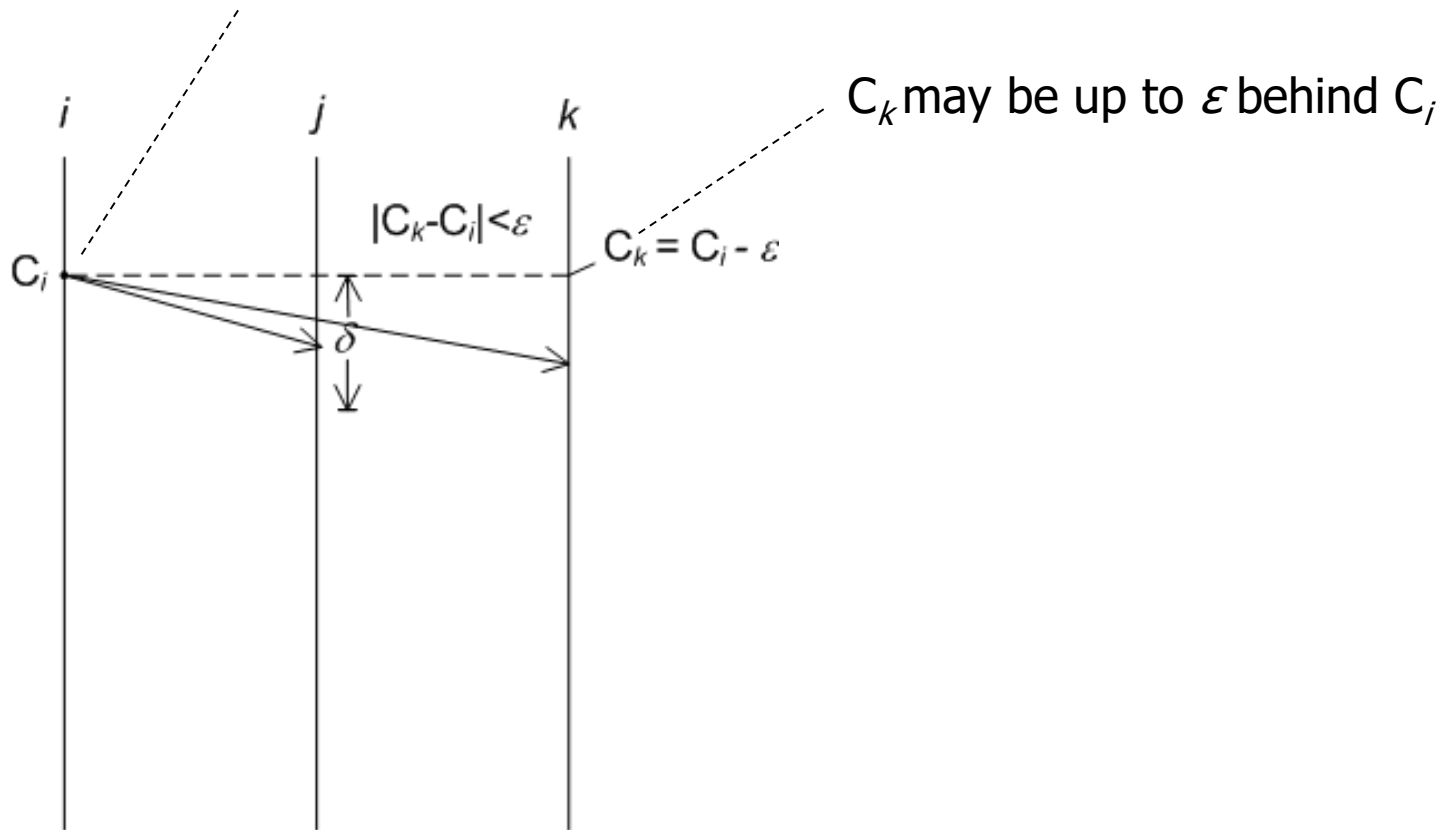
Totally ordering operations across replicas ensures consistency

Using synchronized physical clocks

- Senders timestamp message with physical timestamp
- Clocks synchronized within ε (skew)
- Communication delay at most δ
- Deliver a message m only if there is no possibility that another message with timestamp $< ts(m)$ arrives

Using synchronized physical clocks

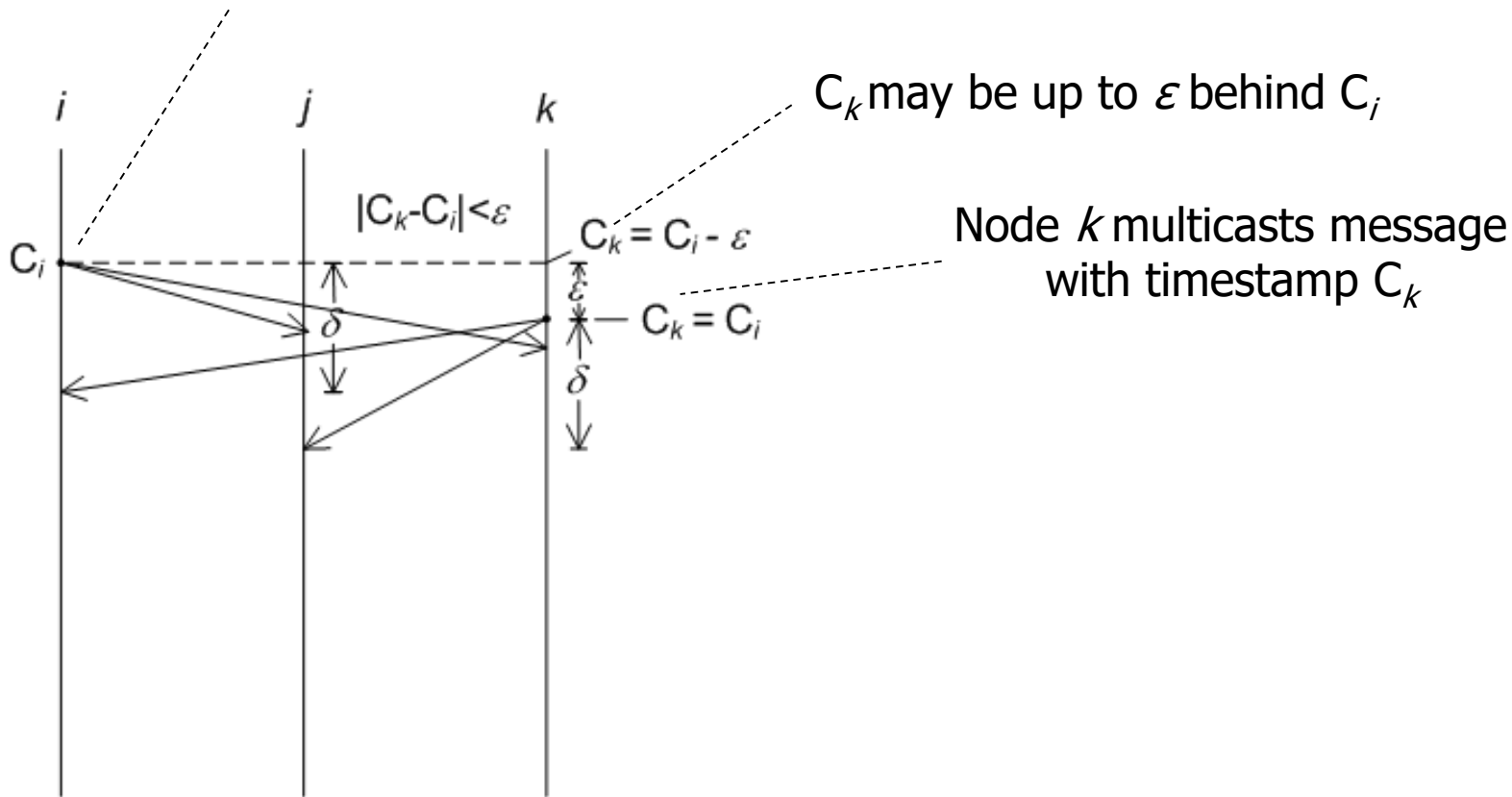
Node i multicasts message with timestamp C_i



ϵ : max clock skew, δ : max communication delay

Using synchronized physical clocks

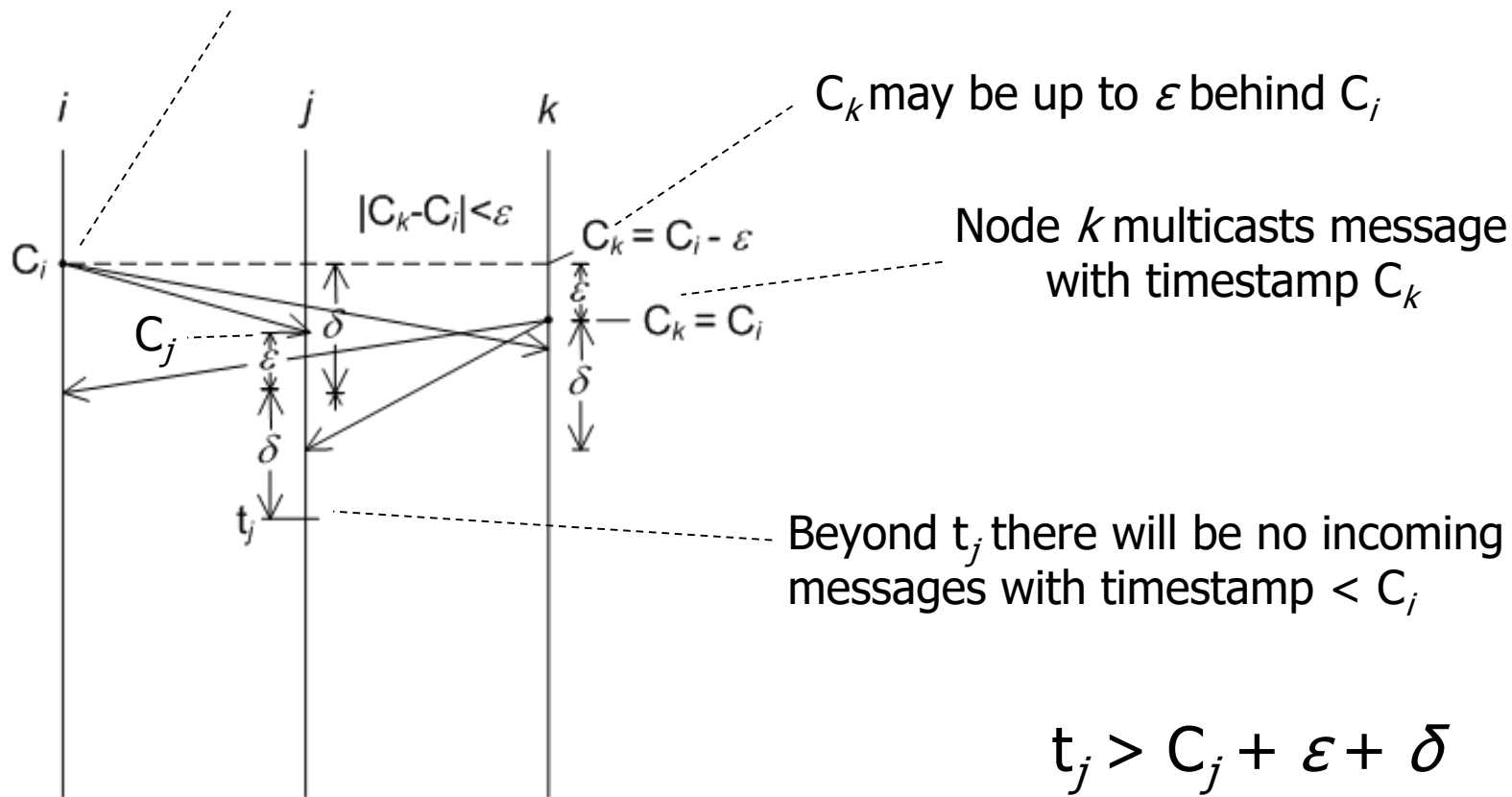
Node i multicasts message with timestamp C_i



ε : max clock skew, δ : max communication delay

Using synchronized physical clocks

Node i multicasts message with timestamp C_i

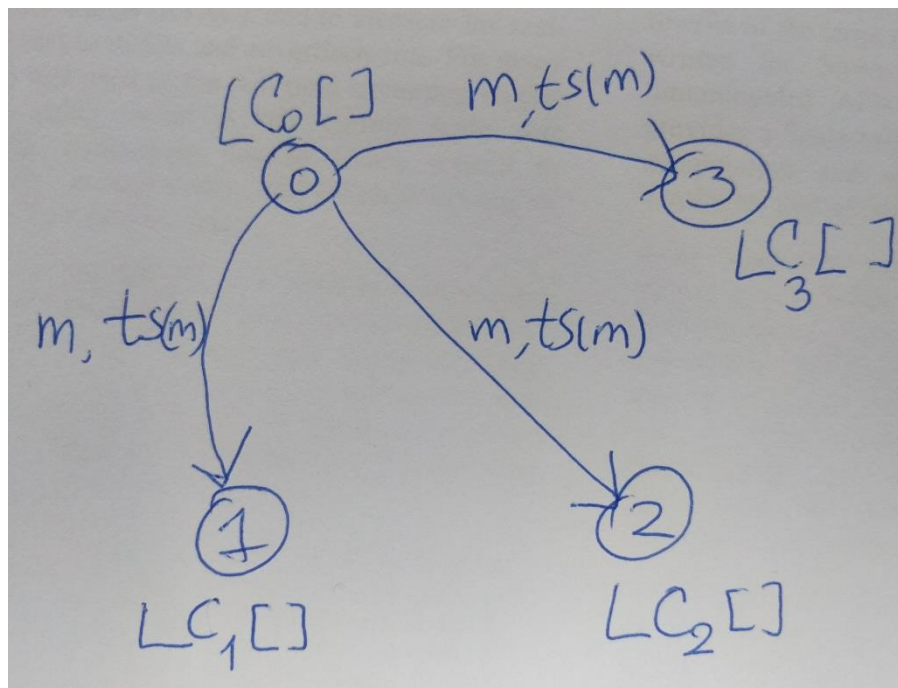


ϵ : max clock skew, δ : max communication delay

Using logical clocks

- Senders timestamp message with their Lamport clock
- Receivers order by sender LC (lowest LC first)
- But: deliver a message only if there is no possibility that another message arrives with lower timestamp
- Deliver a message m when we know that the LC of all other nodes is $> ts(m)$
- Note: Assuming FIFO channels

Total order multicasting (1)



Each node p maintains view of Lamport clocks at all nodes $LC_p []$

On arrival of $m, ts(m)$ at p :

$$LC_p[p] = \max (ts(m), LC_p[p]) + 1$$

$$LC_p[\text{sender}(m)] = ts(m)$$

$m, ts(m)$
$m', ts(m')$
$m'', ts(m'')$

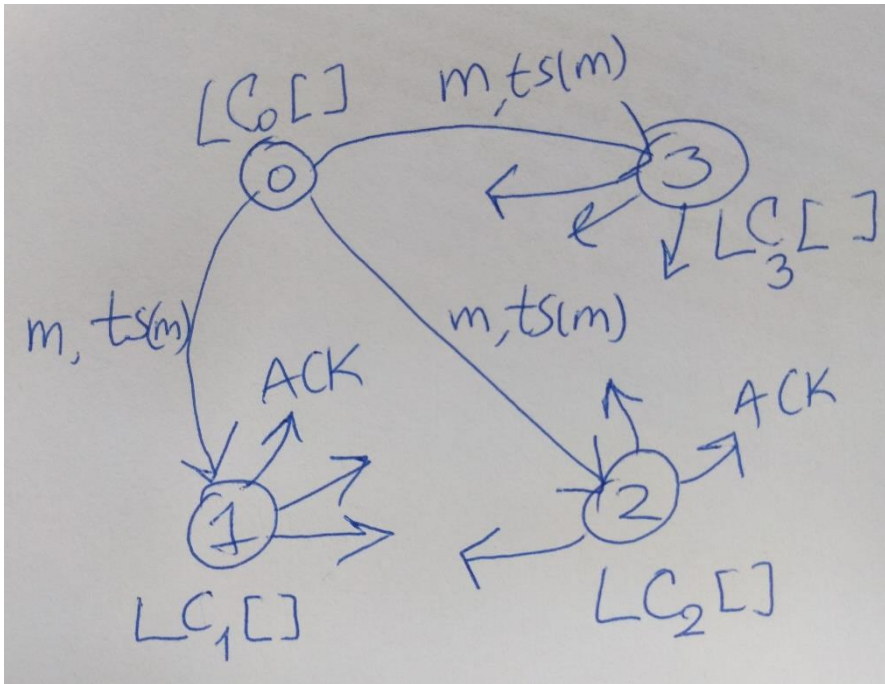
← if $(ts(m') \leq \min_{q \in P} LC_p[q])$ **m' can be delivered**

Deliver in increasing order of $ts(m)$, $\text{sender}(m)$

But, what if some processes do not send messages?

Total order multicasting (2)

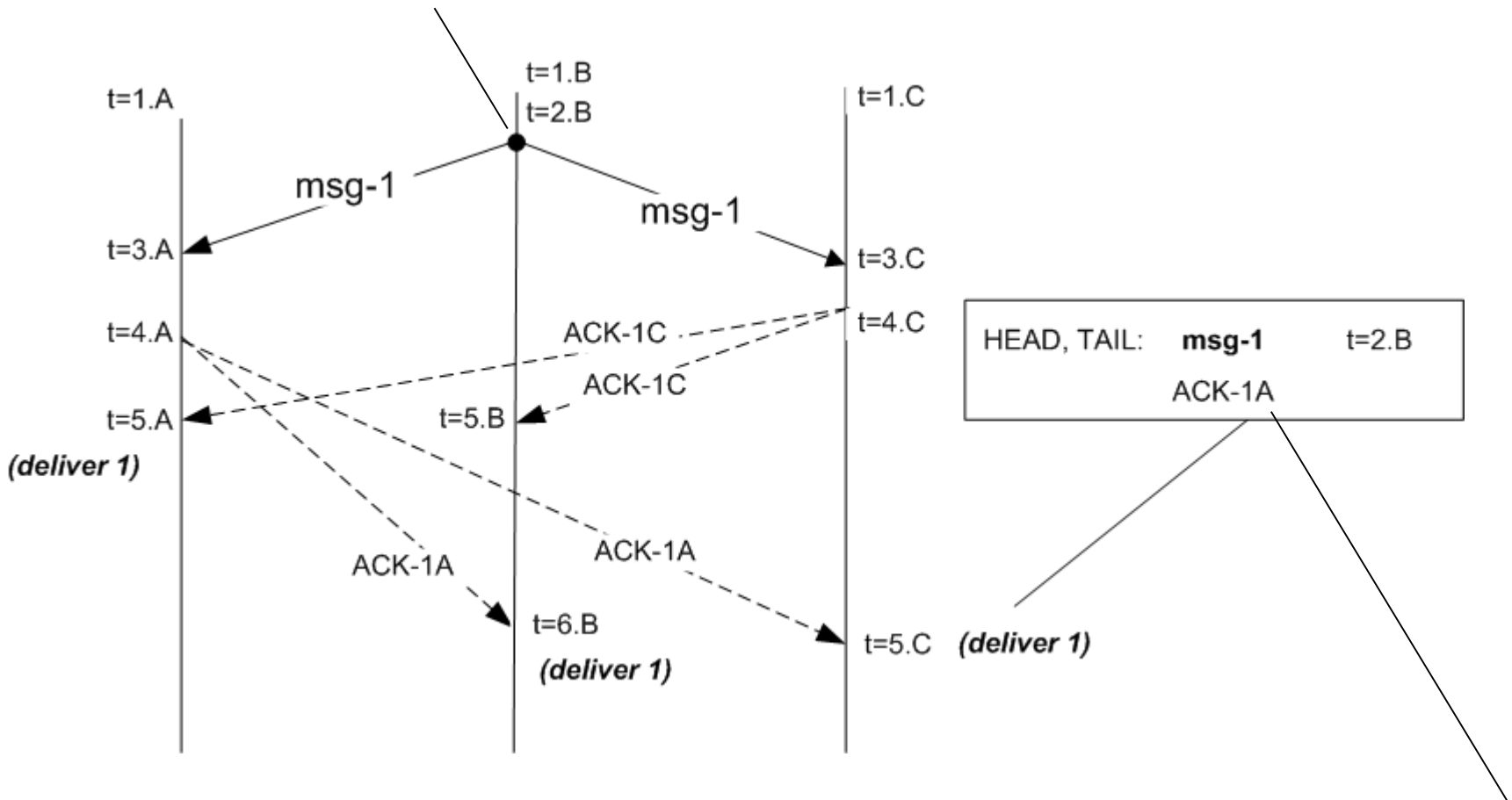
Request that nodes acknowledge (ACK to all) any message received



- ACKs carry timestamps too
- Taken into account in $LC_p[] \forall p$
- Safe to deliver m after receiving $ACK(m)$ from all nodes

Example (1)

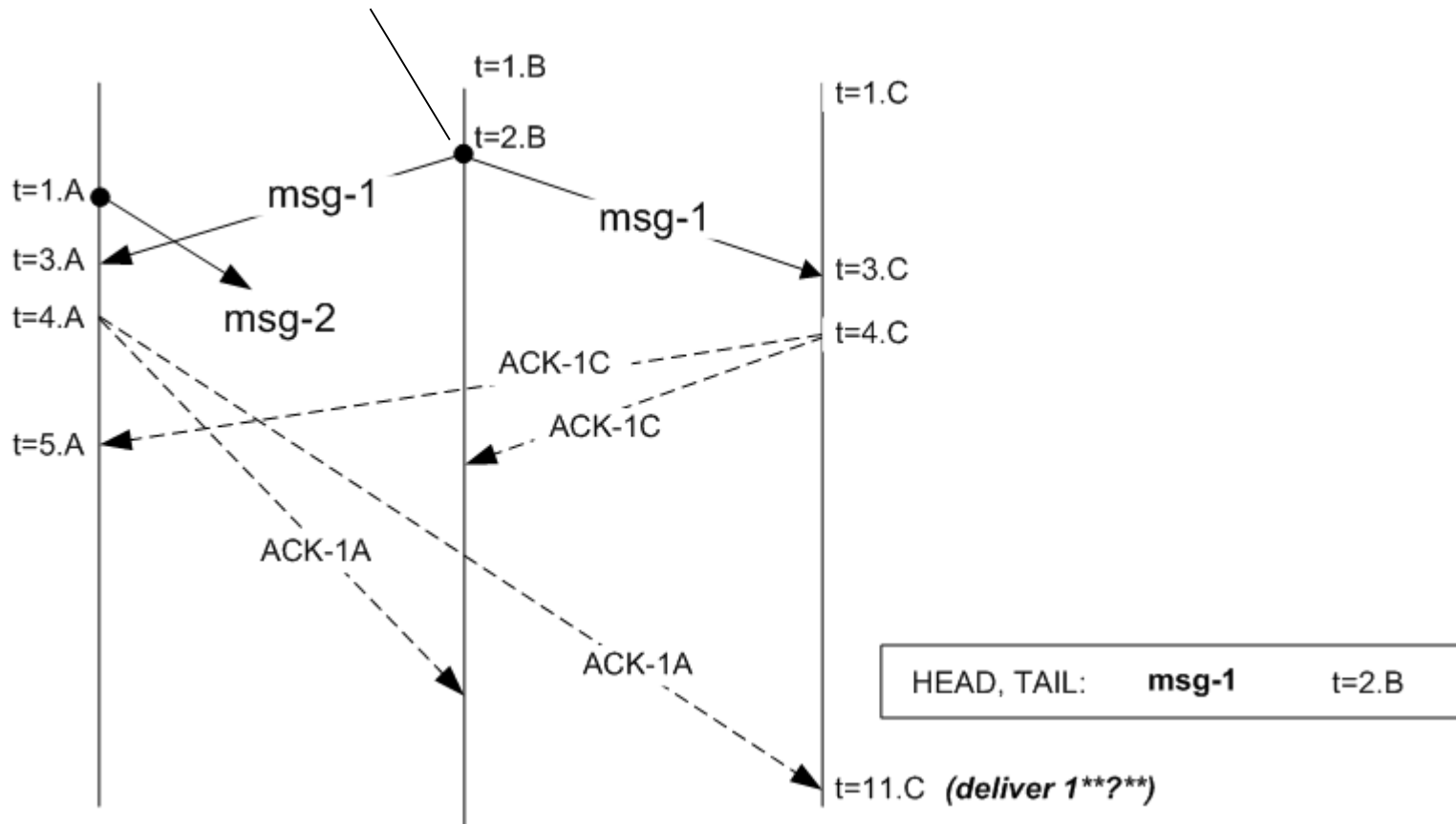
B multicasts msg-1 at $t = 2.B$ (using that timestamp over all msg-1 messages)



ACK-1A assures C that no message with timestamp $< t=2.B$ is expected from any node

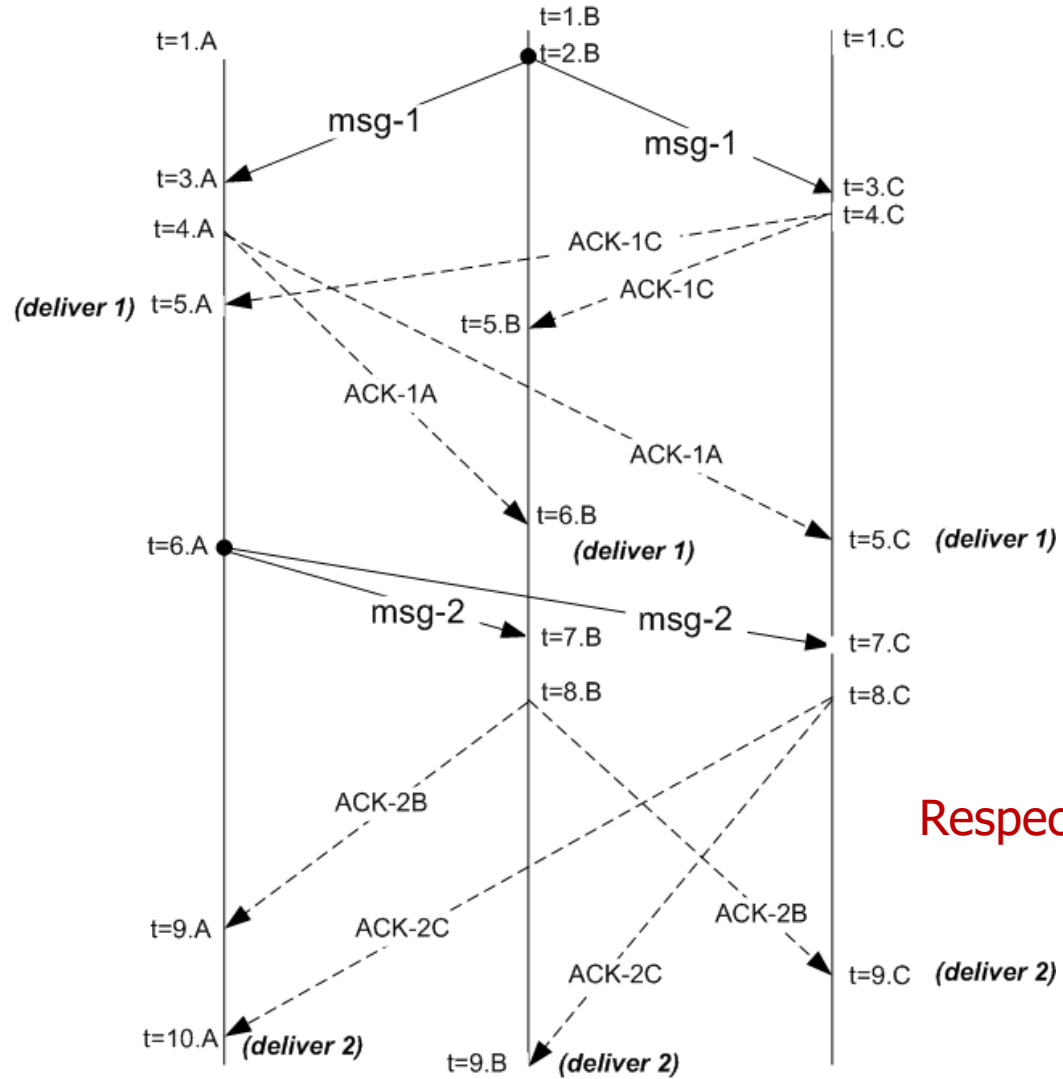
Example (2)

B multicasts msg-1 at $t = 2.B$, A multicasts msg-2 at $t = 1.A$: *delivery order?*



Cannot happen due to FIFO channels assumption!

Example (3)



Respects causal order

Total order multicasting

- Group of N processes
 - Must be aware of each other
 - Each message from a process is multicast to the group
 - FIFO and loss-less communication channels
- Each process
 - Each message carries its Lamport clock
 - Build an ordered queue of messages based on timestamp
 - Acknowledge each message to the group (multicast ACK)
 - Deliver a message only when
 - Message has been acknowledged by all processes in group
 - Message is at the top of the queue