Bringing order to the galaxy

SYNCHRONY VS ASYNCHRONY

Synchronous systems

- Known bound on message delivery
- Known bound on processing speed
- Considered a strong assumption

Asynchronous systems

- **No bound** on message delivery
- **No bound** on processing speed
- Weak assumption = less vulnerable
- Asynchronous $\neq$ slow

This lecture: asynchronous + no process failures

Ordering events in a distributed system

What does it mean for an event to "happen before" another event?

What is a distributed system?

A collection of distinct processes that:

- are spatially separated
- communicate with one another by exchanging messages
- have non-negligible communication delay
- do not share fate
- have separate physical clocks

(imperfect, unsynchronized)

Non-distributed system

- A single clock
- Each event has a timestamp
- **Compare timestamps to order events**

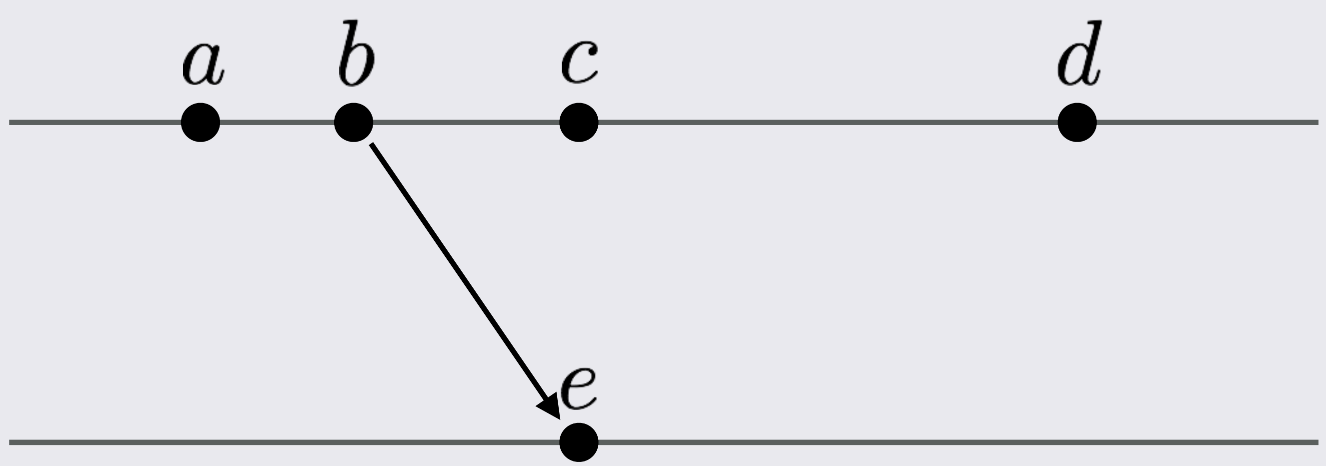
Distributed system

- Each process has its own clock
- Each clock runs at a different speed
- **Cannot directly compare clocks**

ORDERING EVENTS WITHOUT PHYSICAL CLOCKS

Modeling a process:

- A set of instantaneous events with an a priori total ordering
- Events can be local, sends, or receives.

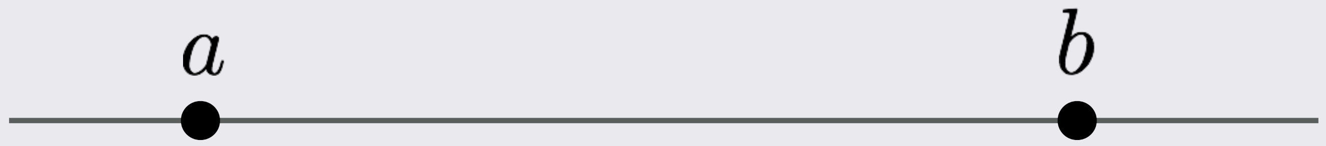


ORDERING EVENTS WITHOUT PHYSICAL CLOCKS

"Happened-before" relation, denoted: $\rightarrow$

Part 1

- If a and b are events on the same process and a comes before b, then $a \rightarrow b$

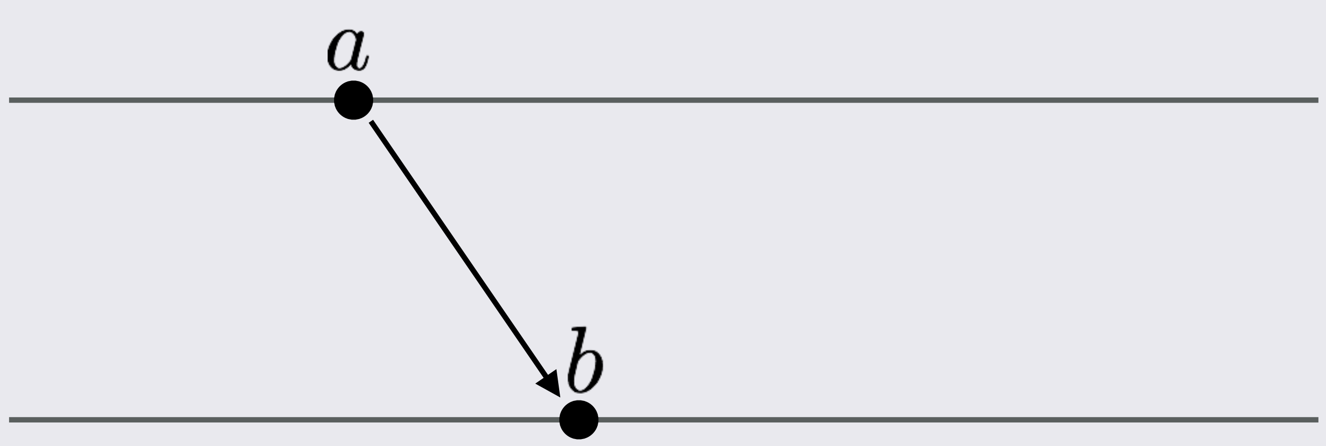


ORDERING EVENTS WITHOUT PHYSICAL CLOCKS

"Happened-before" relation, denoted: $\rightarrow$

Part 2

- If a is the sending of a message by one process and b is the receipt of the same message by another process, then $a \rightarrow b$

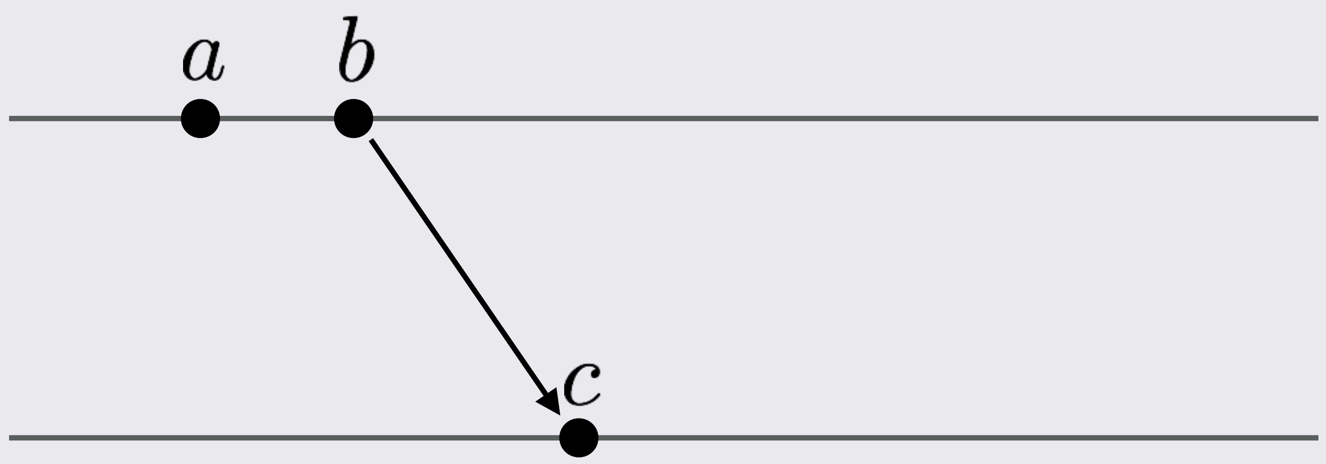


ORDERING EVENTS WITHOUT PHYSICAL CLOCKS

"Happened-before" relation, denoted: $\rightarrow$

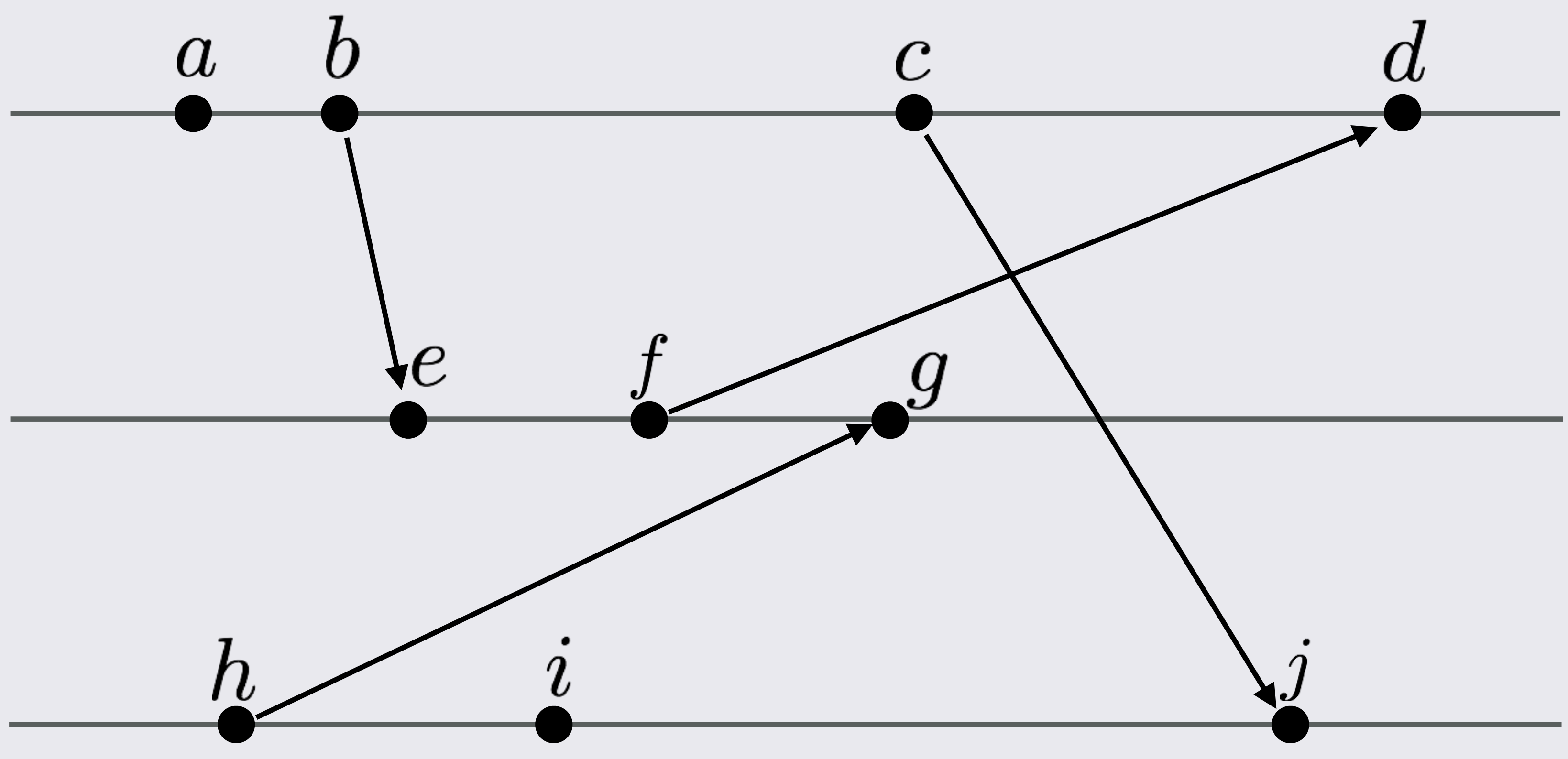
Part 3

- If $a \rightarrow b$ and $b \rightarrow c$, then $a \rightarrow c$



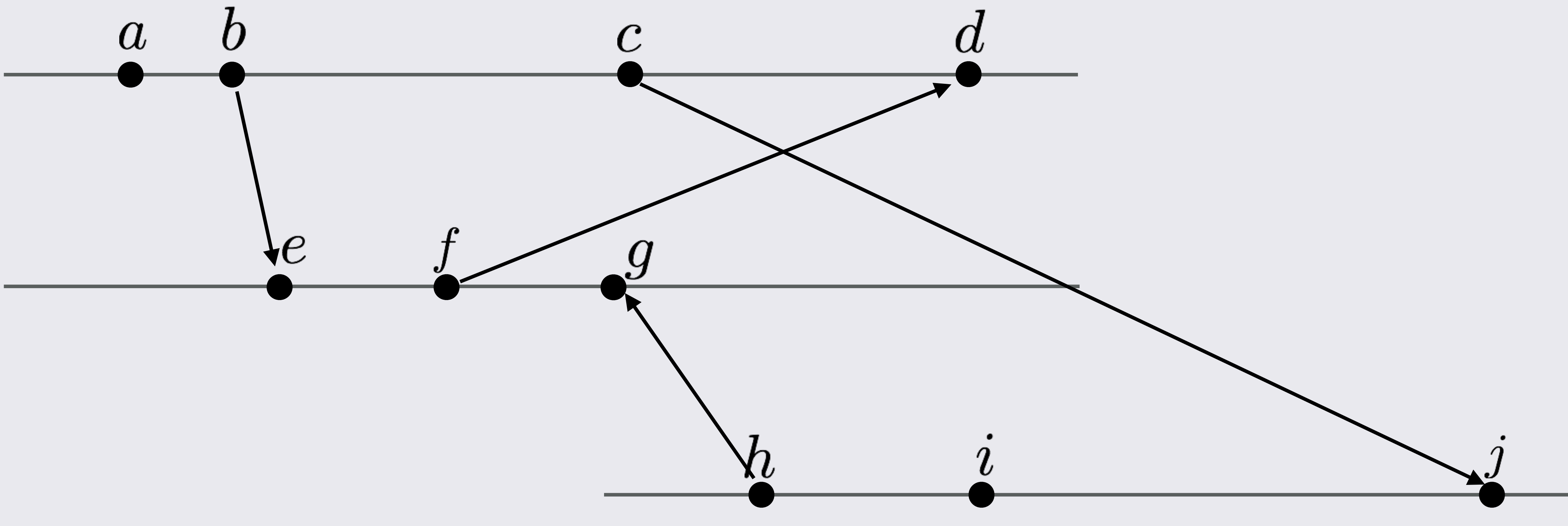
Ordering events without physical clocks

Putting it all together



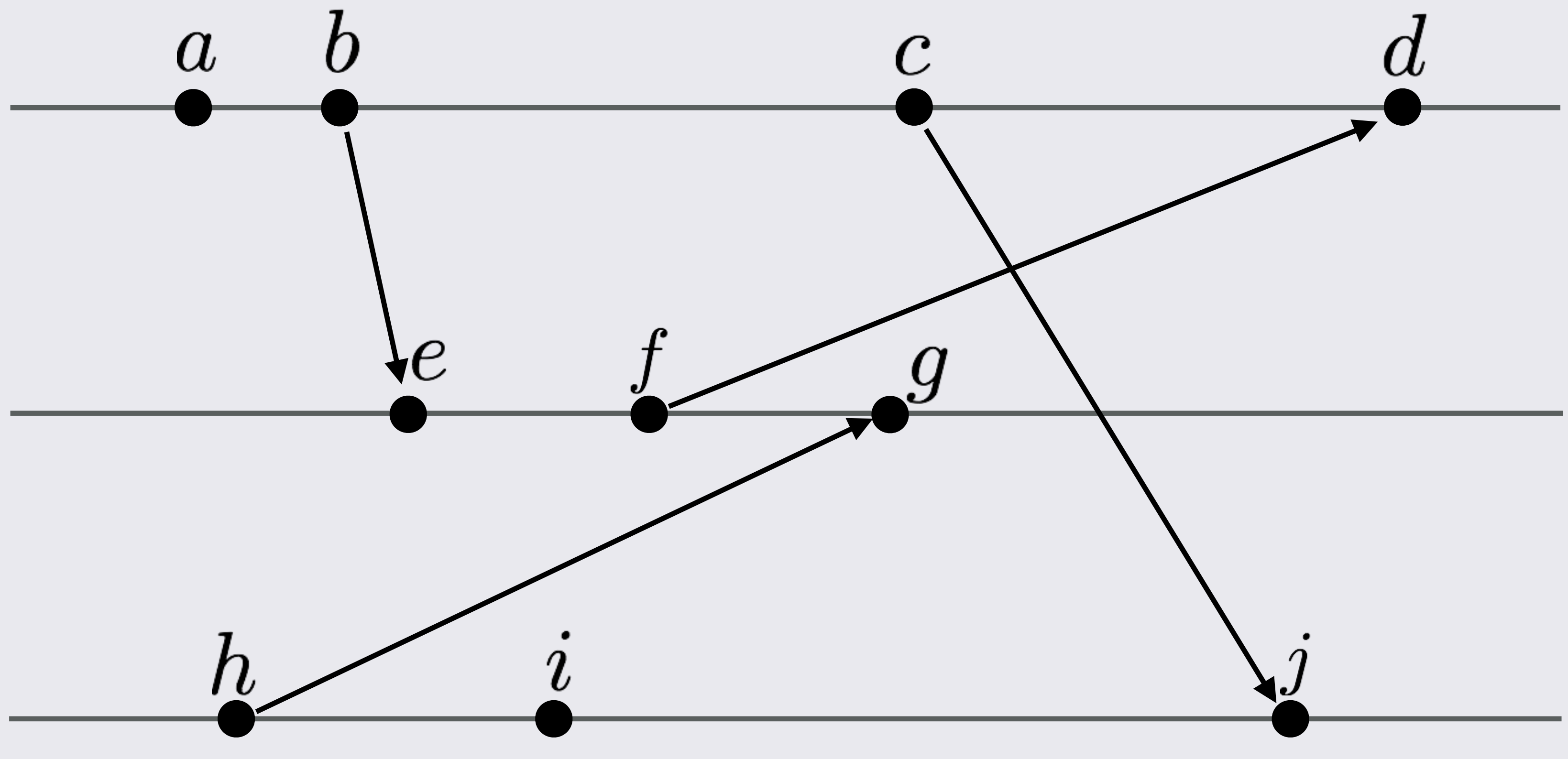
ORDERING EVENTS WITHOUT PHYSICAL CLOCKS

Can arrows go backwards?



ORDERING EVENTS WITHOUT PHYSICAL CLOCKS

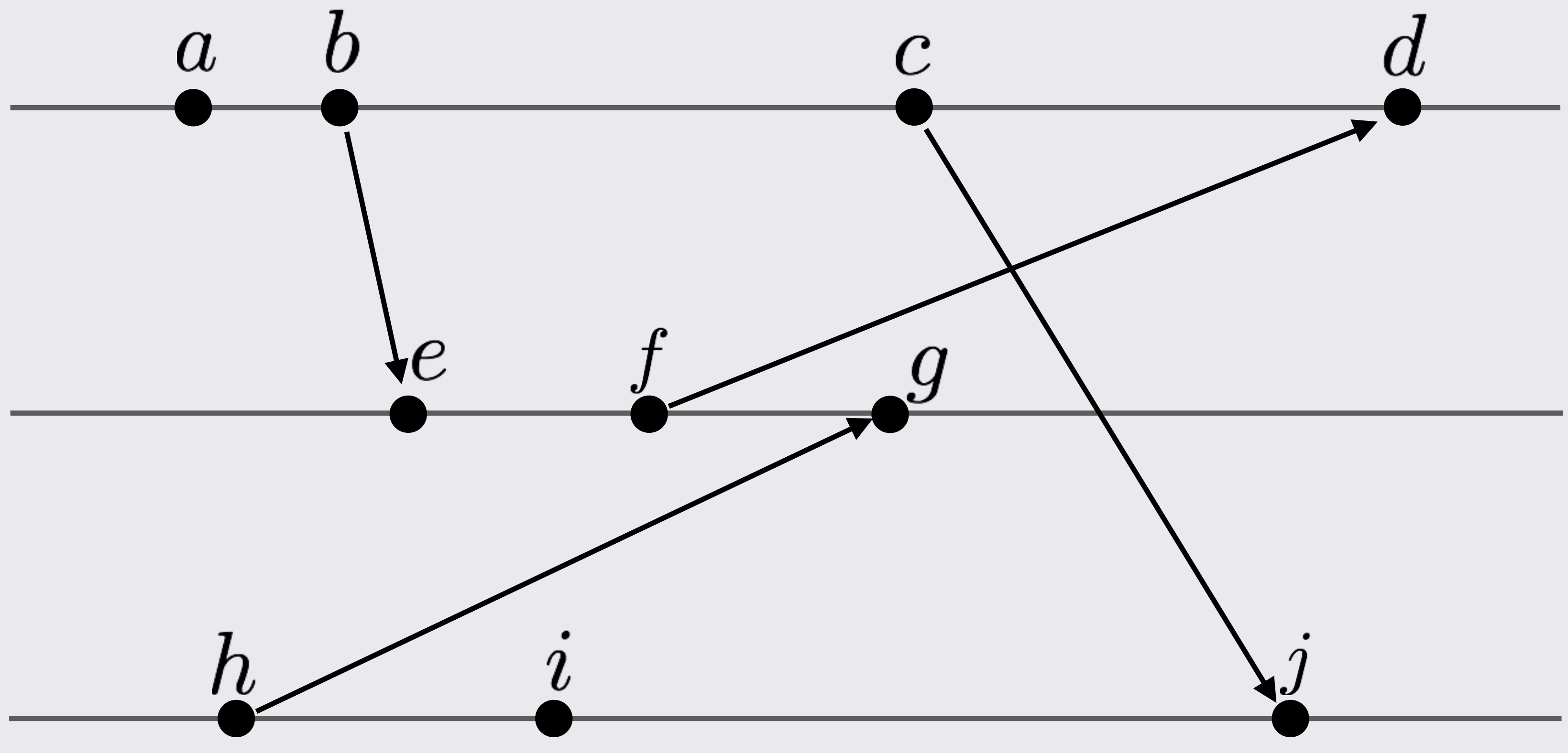
Can cycles be formed?



No, because an event would happen before itself

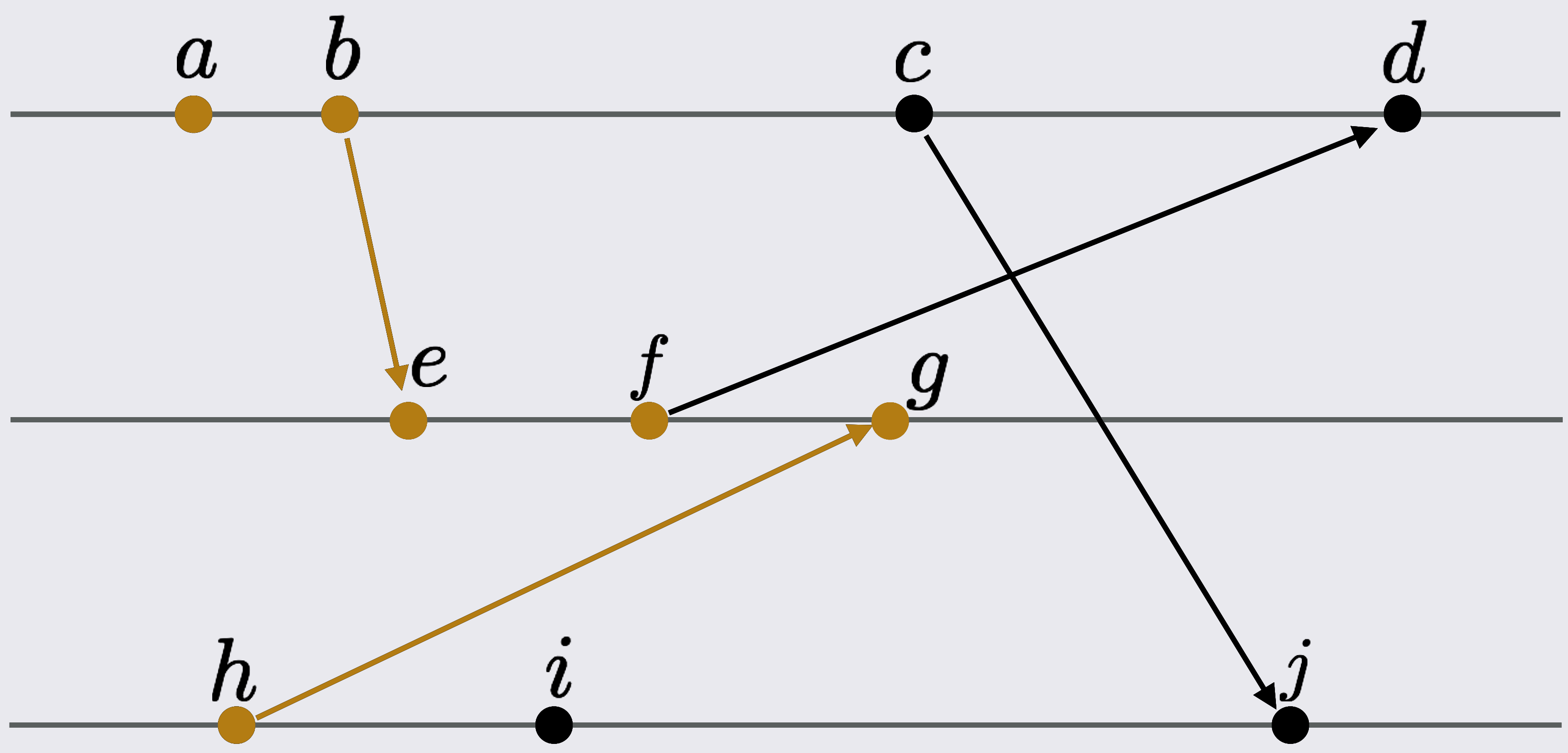
ORDERING EVENTS WITHOUT PHYSICAL CLOCKS

Are all events related by $\rightarrow$?



A PARTIAL ORDER

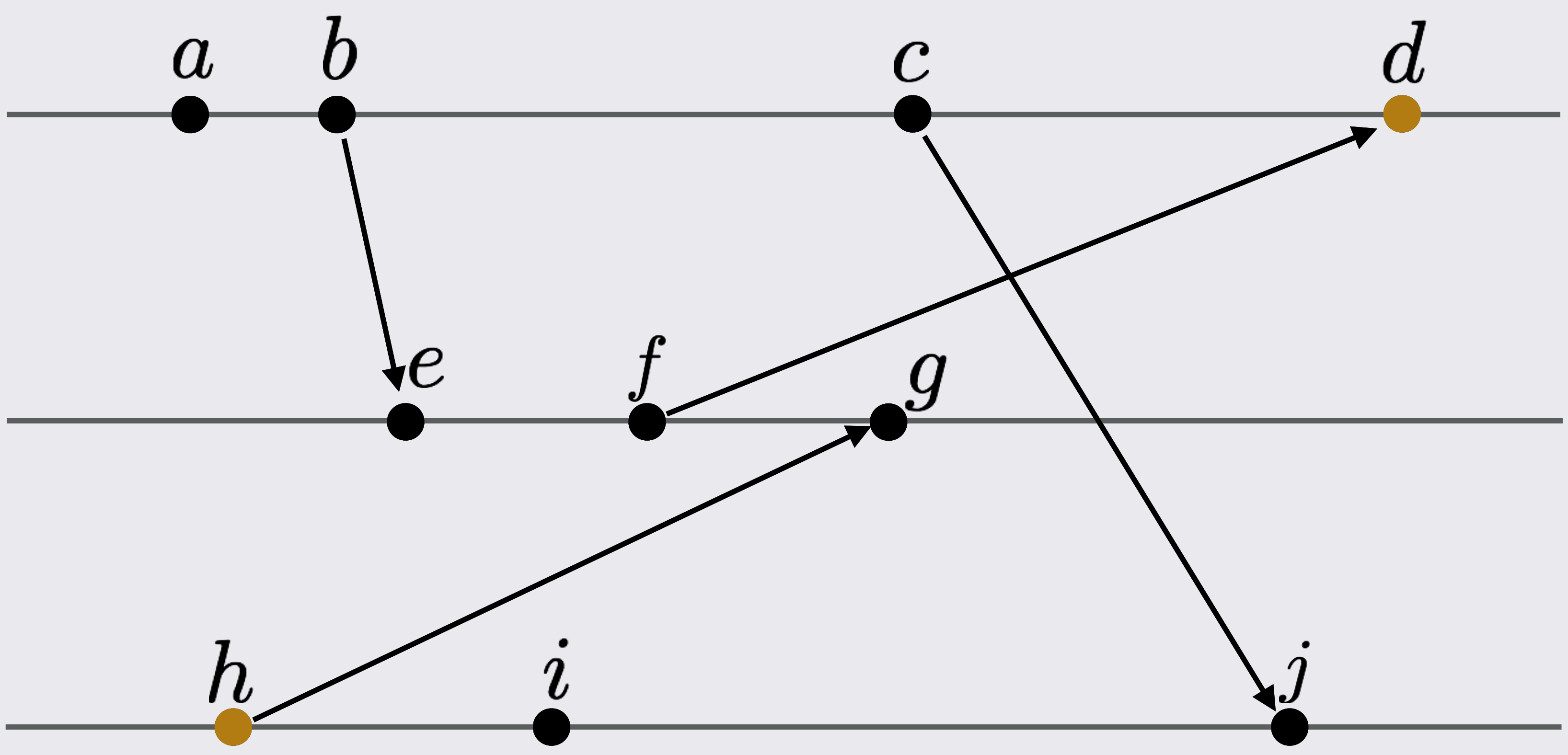
The set of events q such that $q \rightarrow p$ are the events that could have influenced p in some way



$$\{a, b, e, f, h\}$$

A ~~PARTIAL~~ CAUSAL ORDER

If two events could not have influenced each other, it doesn't matter when they happened relatively to each other



h and d are concurrent: $h \nrightarrow d$, $d \nrightarrow h$

Goal: generate a **total** order that is consistent with the happened-before partial order

Lamport clocks

Define a function **LC** such that:

$$p \to q \Rightarrow LC(p) < LC(q)$$

(the Clock condition)

LAMPORT CLOCKS

Define a function **LC** such that:

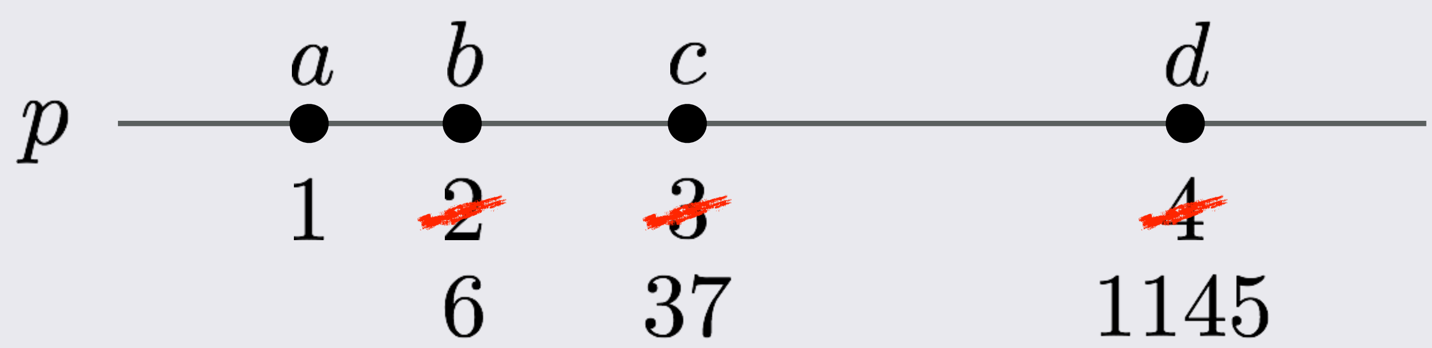
$$p \to q \Rightarrow LC(p) < LC(q)$$

(the Clock condition)

Implement **LC** by keeping a local **LC_i** at each process i

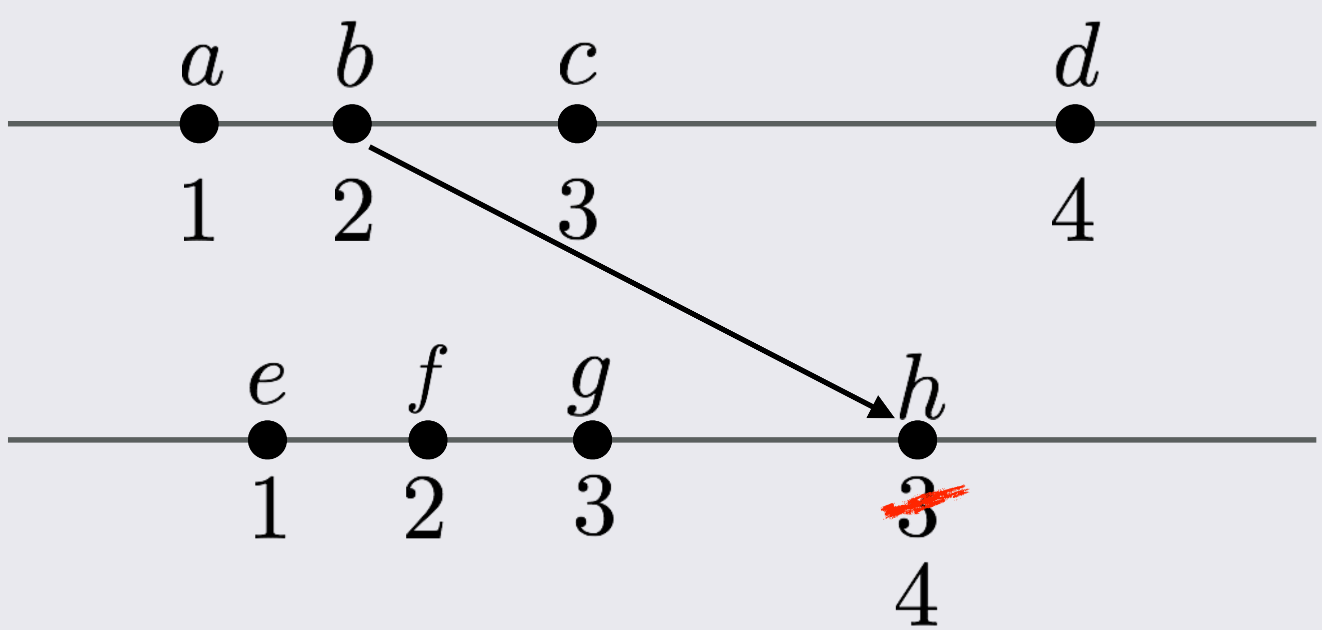
LAMPORT CLOCKS

Single process



LAMPORT CLOCKS

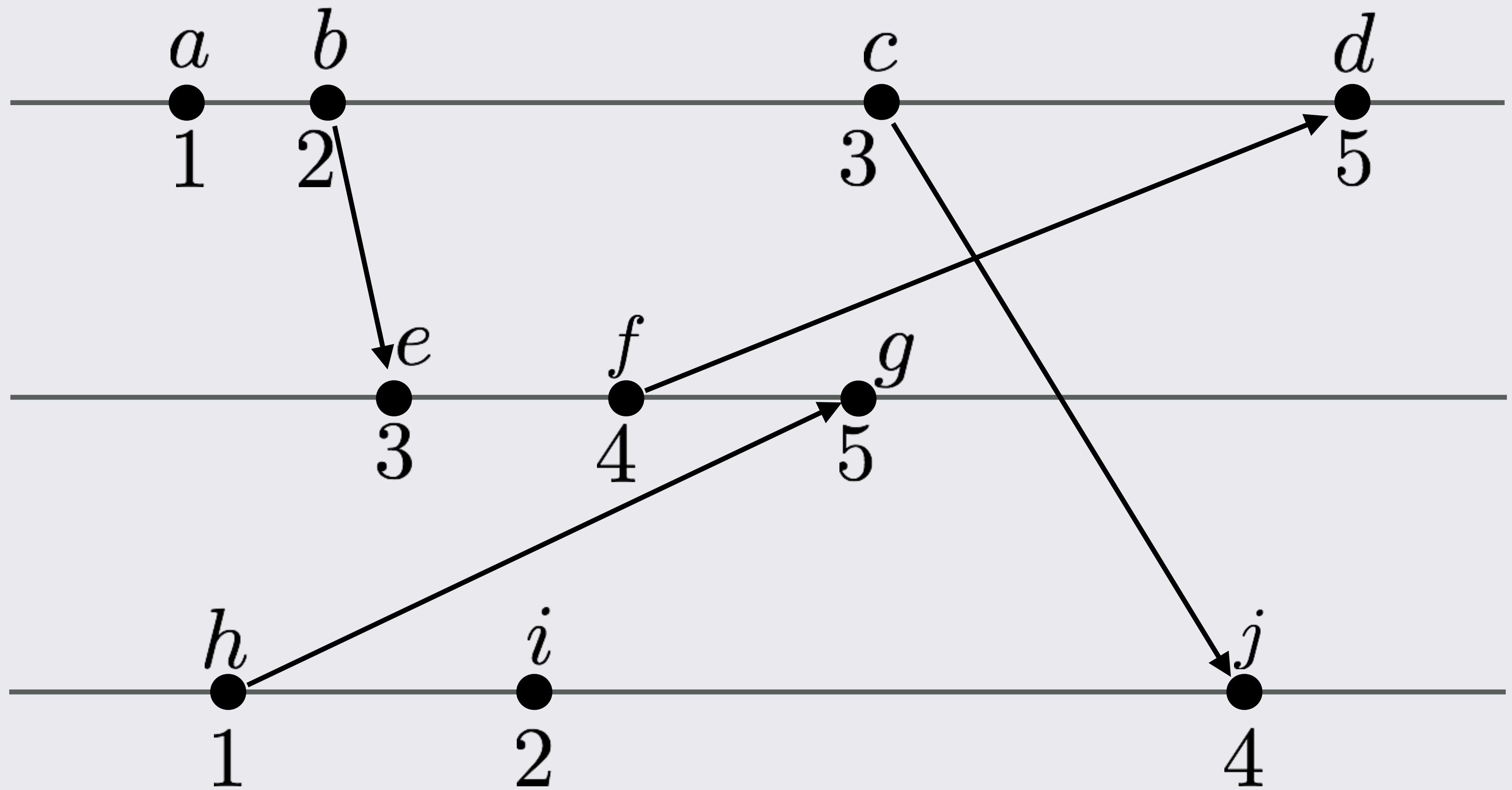
Across processes



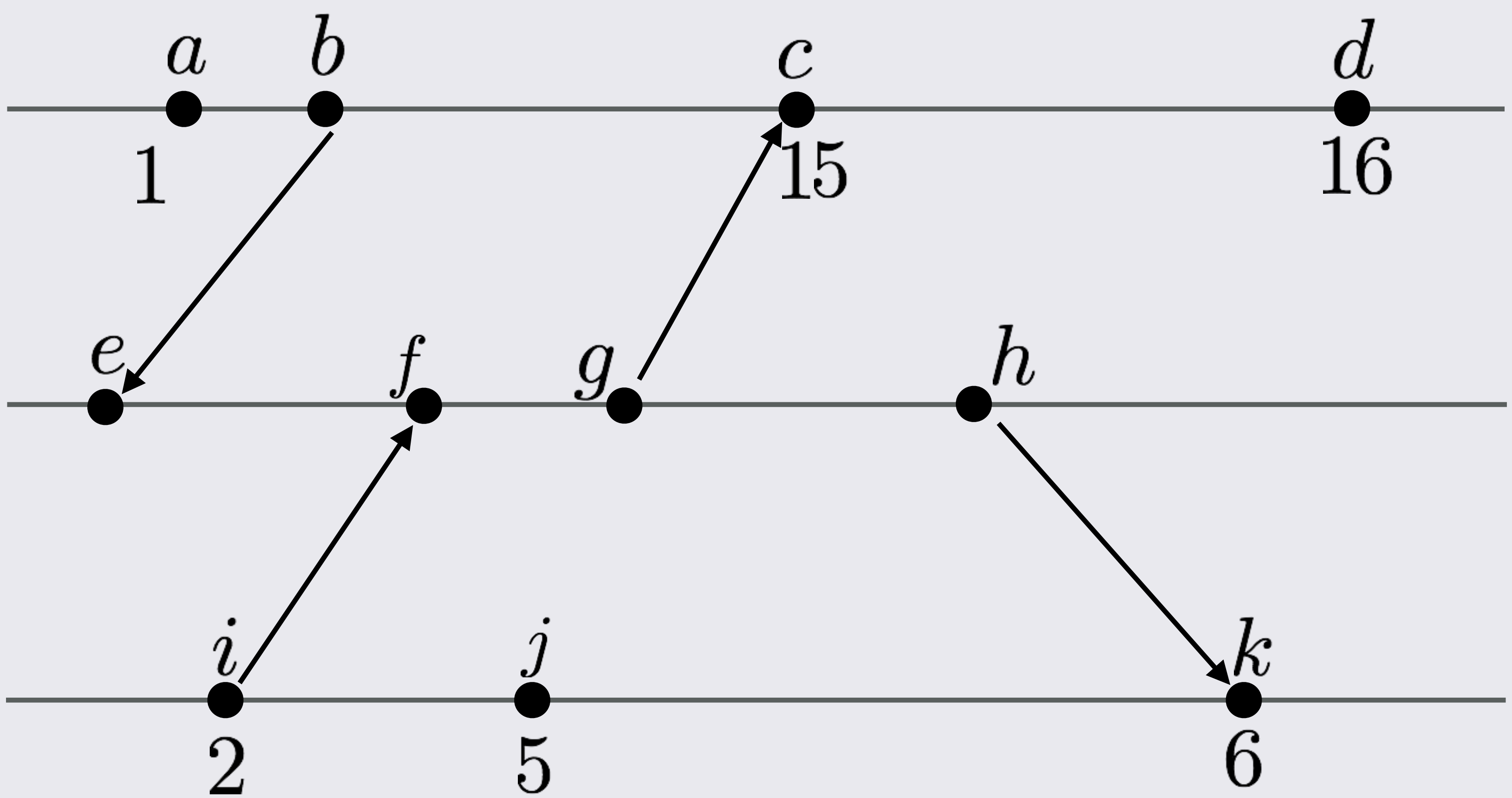
$$b \rightarrow h \Rightarrow LC(b) < LC(h)$$

$$g \rightarrow h \Rightarrow LC(g) < LC(h)$$

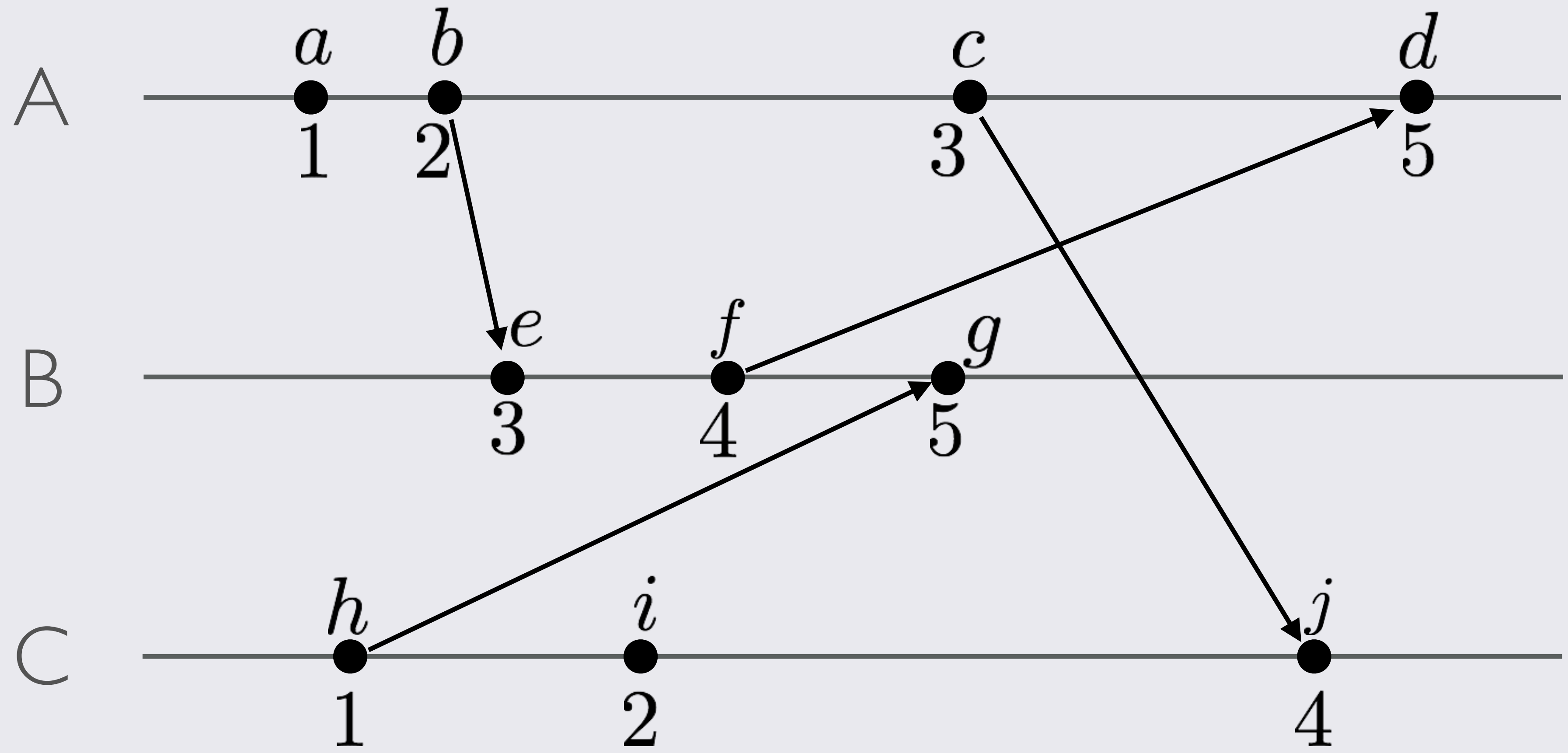
Putting in all together



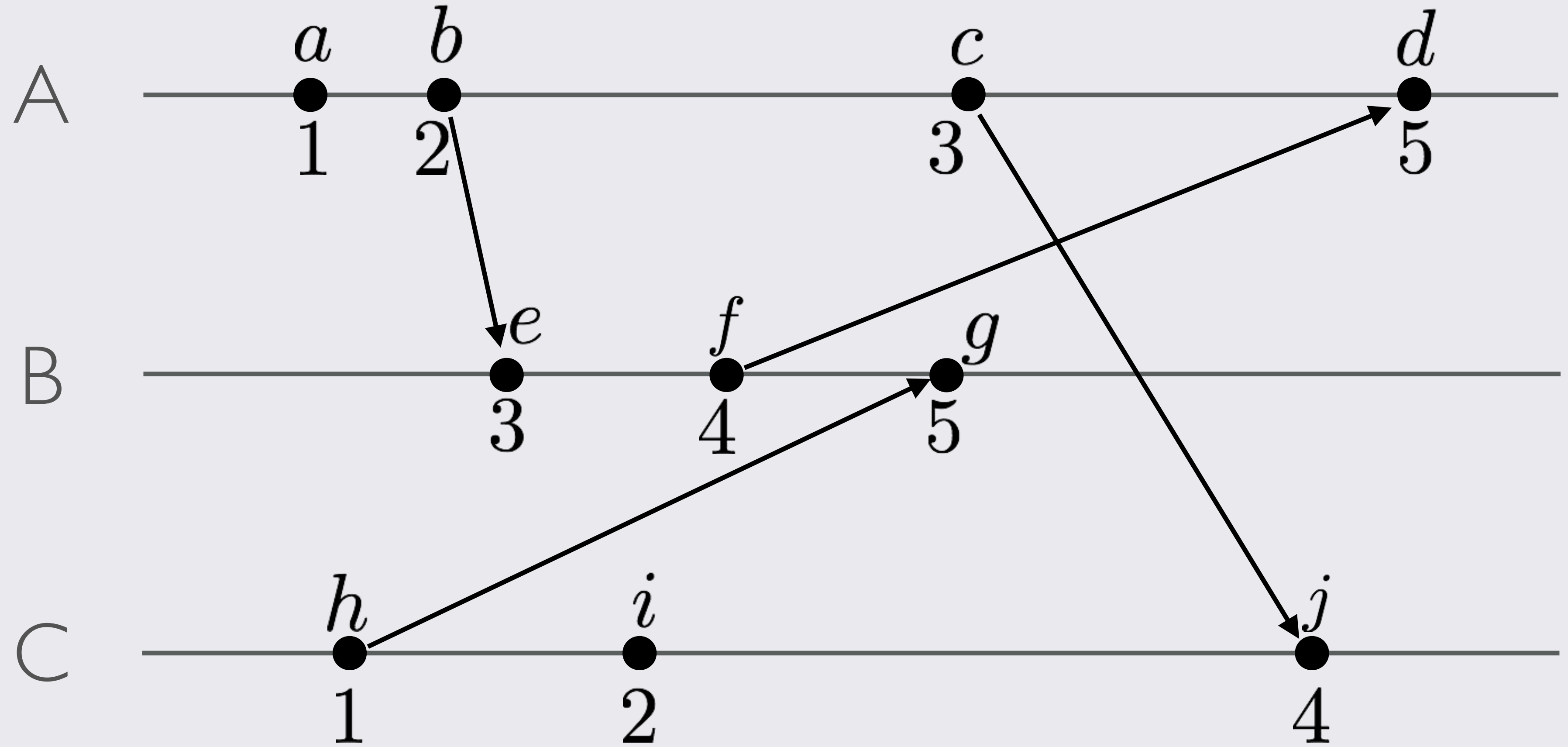
Is this correct?



Generating a total order



- Order messages by LC
- Ties are broken by unique process ID



Lamport clocks implement the Clock condition

$$p \to q \Rightarrow LC(p) < LC(q)$$

But is that all we need?

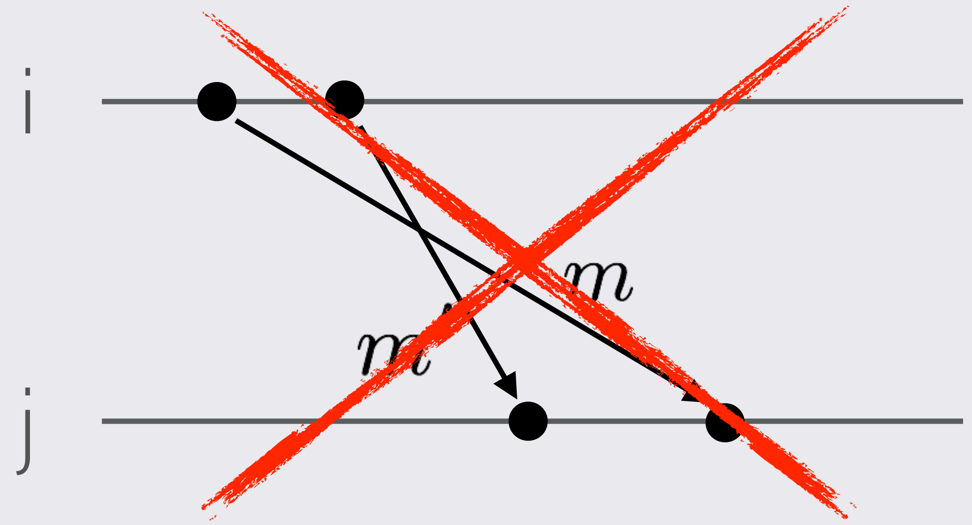
ADMINISTRIVIA

- Make sure to subscribe to our Piazza forum
  - Announcements, discussion, etc.
- At capacity, will issue more overrides as more people drop
- Remember to send me a selfie of you!

FIFO DELIVERY

FIFO delivery

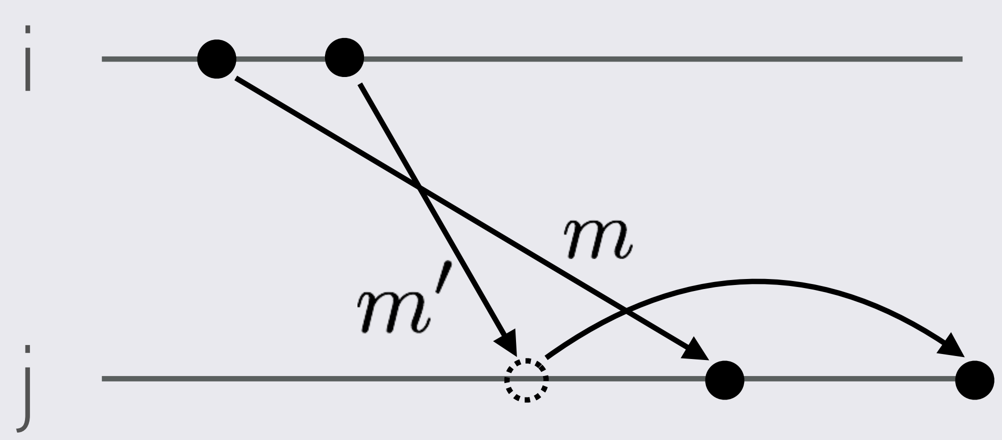
$$send_i(m) \rightarrow send_i(m') \Rightarrow deliver_j(m) \rightarrow deliver_j(m')$$



FIFO DELIVERY

FIFO delivery

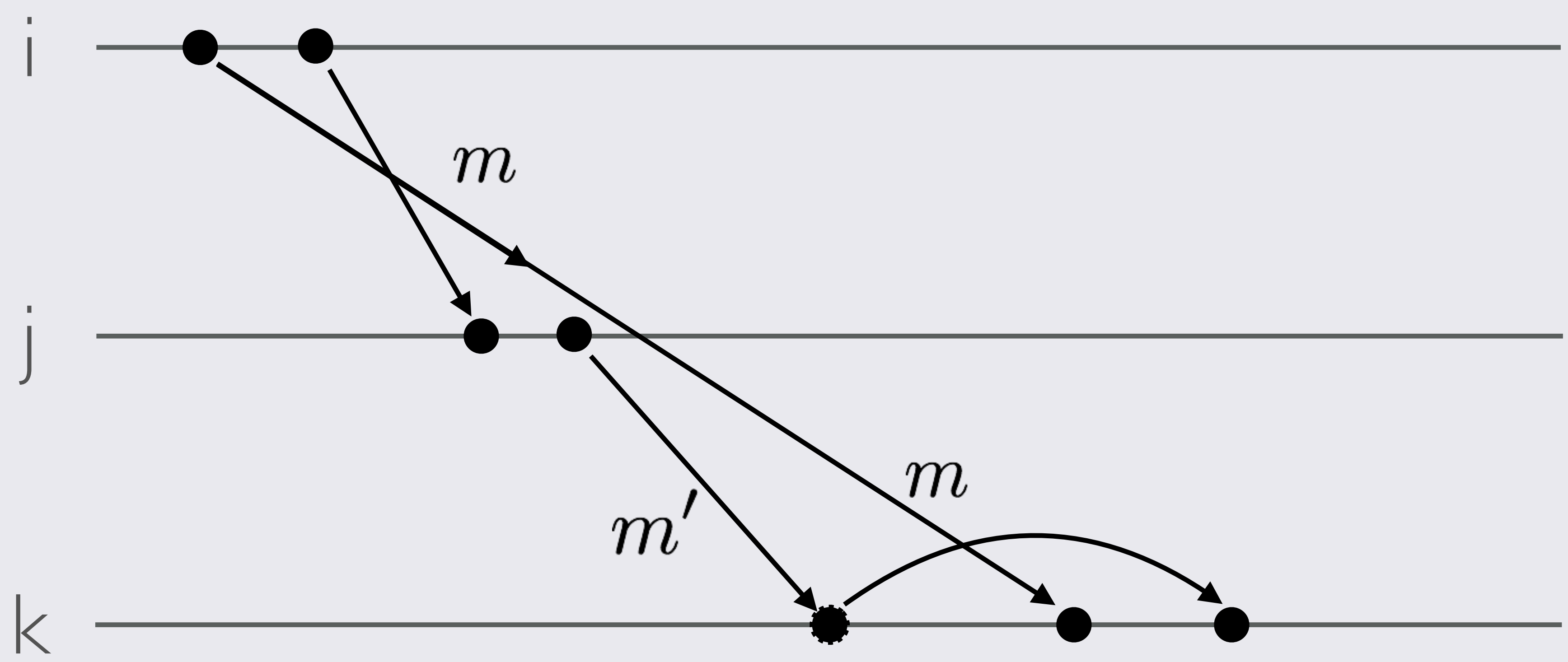
$$send_i(m) \rightarrow send_i(m') \Rightarrow deliver_j(m) \rightarrow deliver_j(m')$$



Causal delivery

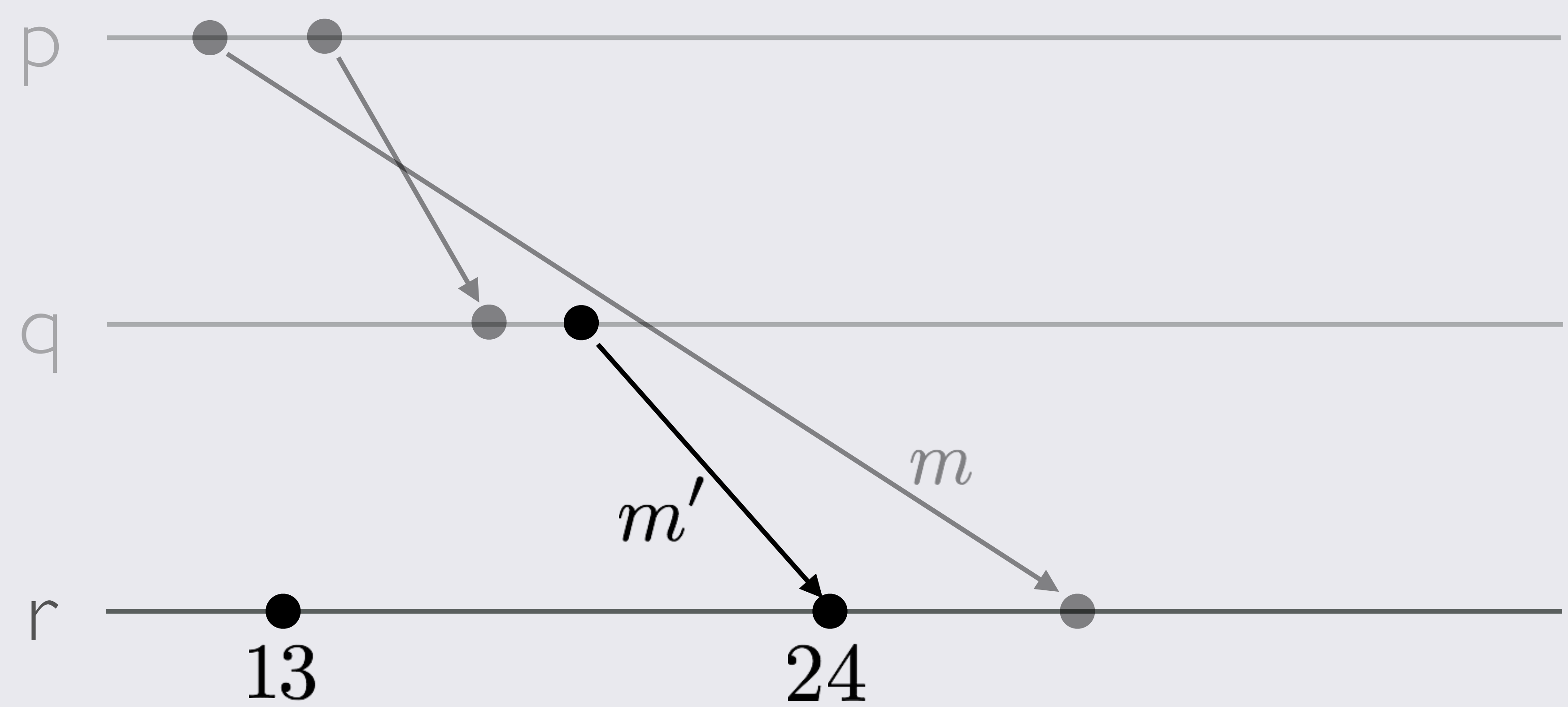
When more processes are involved, causal delivery is needed:

$$send_i(m) \rightarrow send_j(m') \Rightarrow deliver_k(m) \rightarrow deliver_k(m')$$



GAP DETECTION

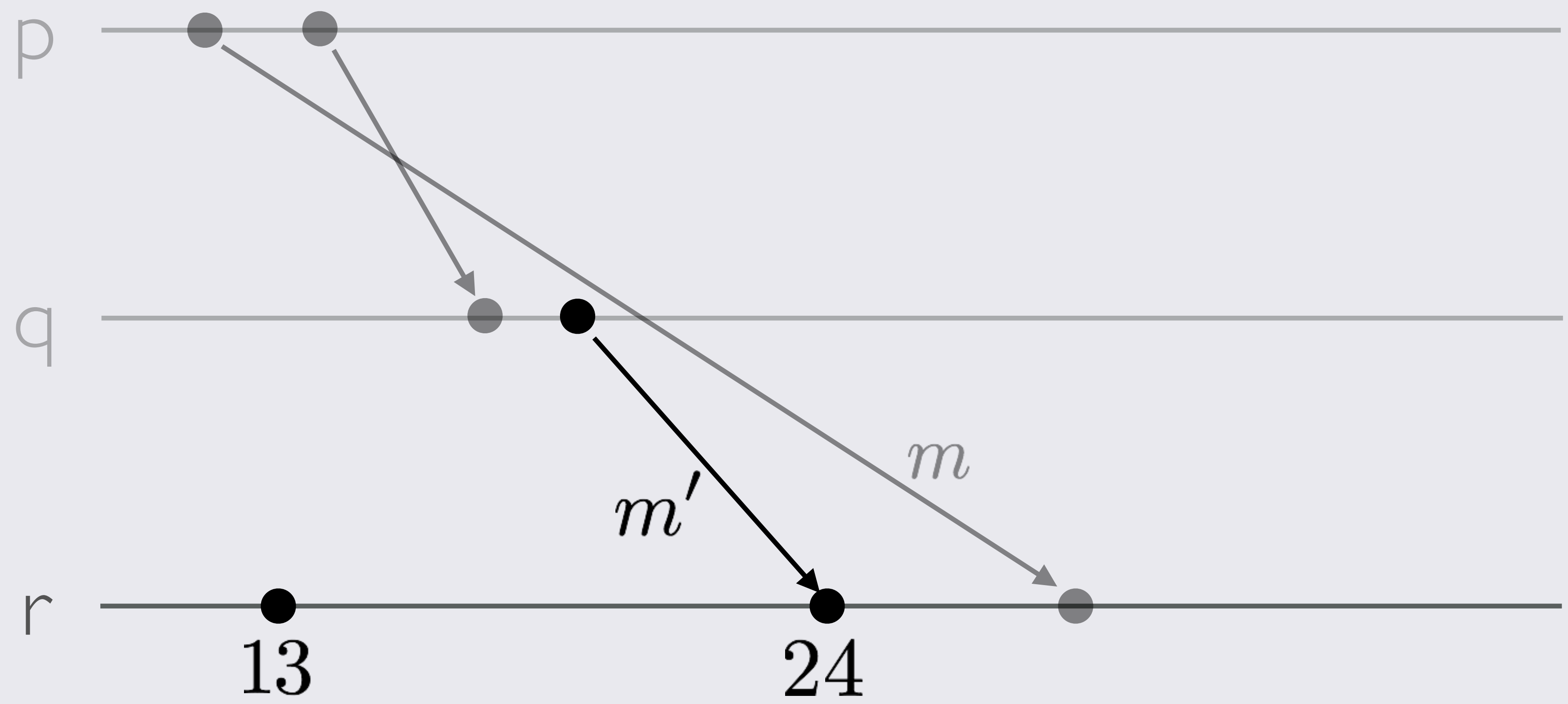
Should r deliver m'?



Gap detection: Given two events e and e', where $LC(e) < LC(e')$, determine whether some other event e'' exists such that

$$LC(e) < LC(e'') < LC(e')$$

Gap detection



Gap detection: Given two events e and e', where $LC(e) < LC(e')$, determine whether some other event e'' exists such that

$$LC(e) < LC(e'') < LC(e')$$

Lamport clocks don't provide gap detection!

How to implement causal delivery?

(in other words, when is it safe to deliver m'?)

a) Wait to receive a message with higher LC from each channel

b) Implement better clocks!

From clocks to strong clocks

$$p \to q \Rightarrow LC(p) < LC(q)$$

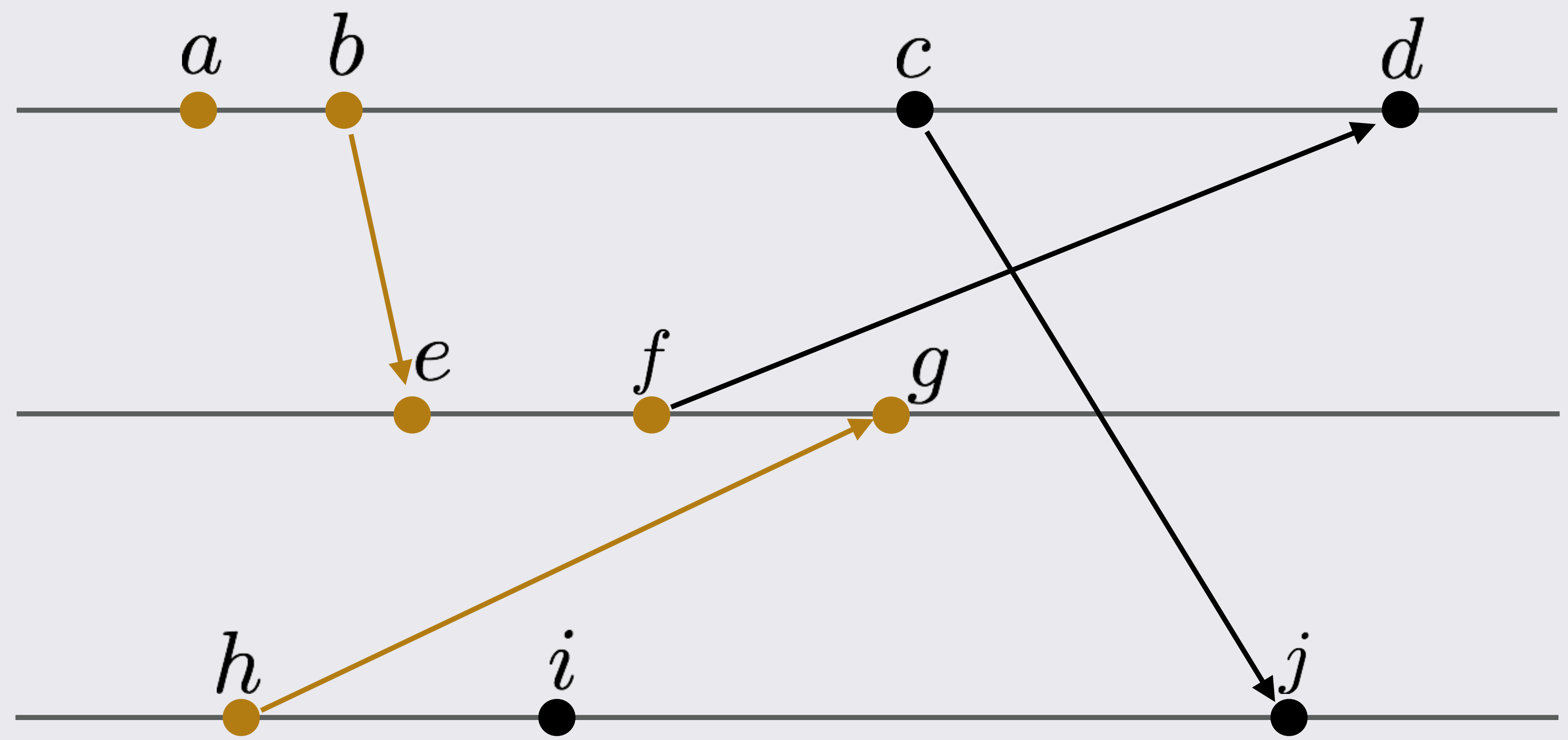
Clock condition

$$p \to q \Leftrightarrow LC(p) < LC(q)$$

Strong clock condition

Causal histories

The set of events q such that $q \rightarrow p$ are the events that could have influenced p in some way



$$\theta(g) = \{a, b, e, f, h, g\}$$

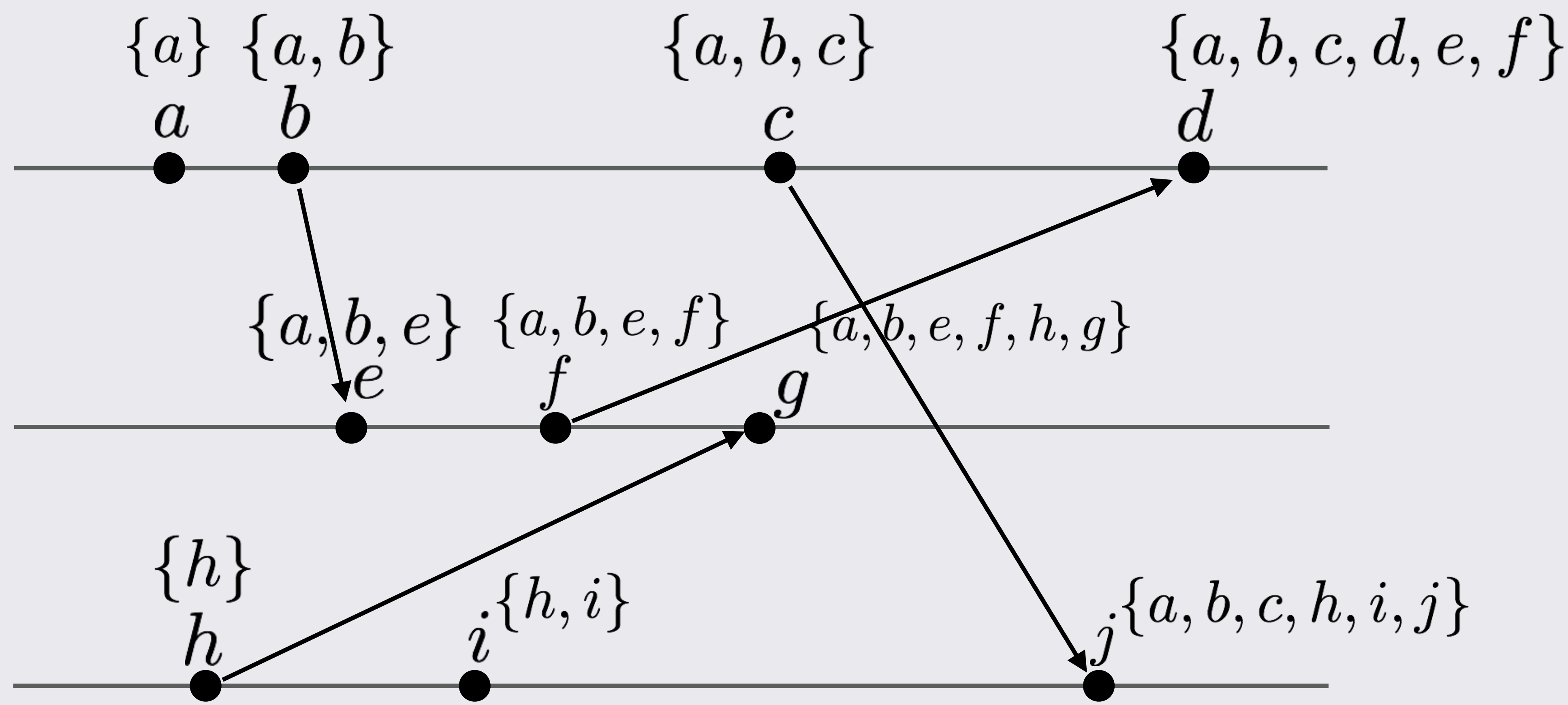
IMPLEMENTING STRONG CLOCKS

(the hard way)

- Initialize $\theta := \emptyset$
- For send and local events e, $\theta(e) := \theta \cup \{e\}$
- For receive events $e = recv(m)$, $\theta(e) := \theta \cup \{e\} \cup \theta(m)$

Implementing strong clocks

(the hard way)



Strong clock condition: $p \rightarrow q \Leftrightarrow \theta(p) \subset \theta(q)$