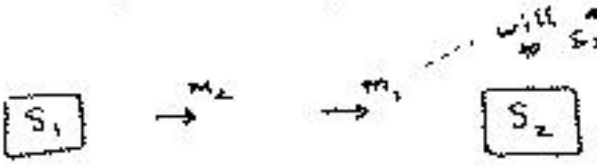
How can Invariant 1 be ensured?

we need to force each queue to be ordered by timestamp



what if two event-messages have been sent from S_1 to S_2 and timestamp(m_2) < timestamp(m_1)?

Can this happen?

First, consider a non-networked simulation

Theorem

If e is the smallest event (least timestamp) in the eventlist, then no event will be scheduled with a smaller timestamp than e.

Proof. Event e will be the first to be executed, and it can only schedule future events.

Example:

ATM queue, with event list

t	event
3	customer finish @ ATM
4	customer arrives to queue

simtime = 2 (current).

Now: simtime → 3, fire first event.

Suppose there is already a queue of 3 customers waiting at the ATM, so firing of first event schedules next in queue to finish at time 3.5.

Now, state of simulation is

t	event
3.5	customer finish @ ATM
4	customer arrives to queue

simtime = 3.

So, events <u>are</u> processed in order, but they can be <u>scheduled</u> out-of-order.

This is dangerous!

So, to ensure Invariant 1,

S, should not send an event message m until it is certain no other message will later be sent with a smaller timestamp than m has.

How can this be done?

Easy answer: "buffer" the messages rather than sending them; wait until simtime ≥ timestamp (m) before sending m.