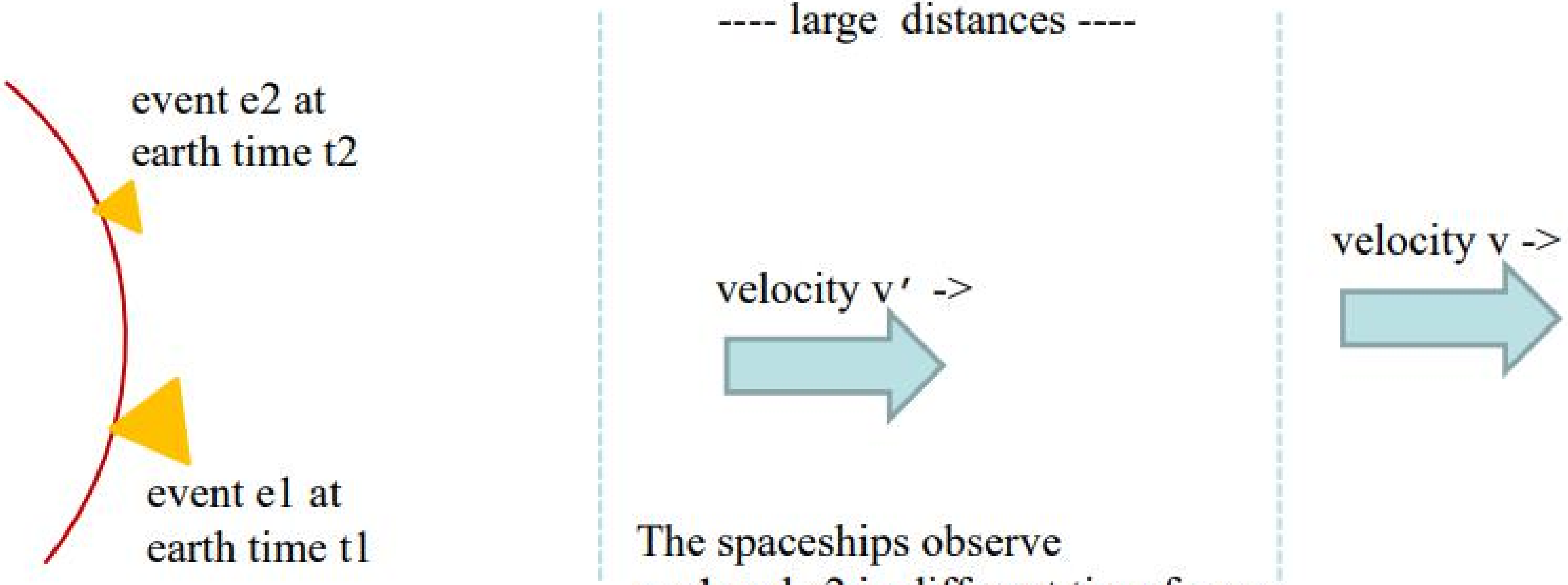
TIME IN DISTRIBUTED SYSTEM

DILIP KUMAR SHRESTHA: GCES

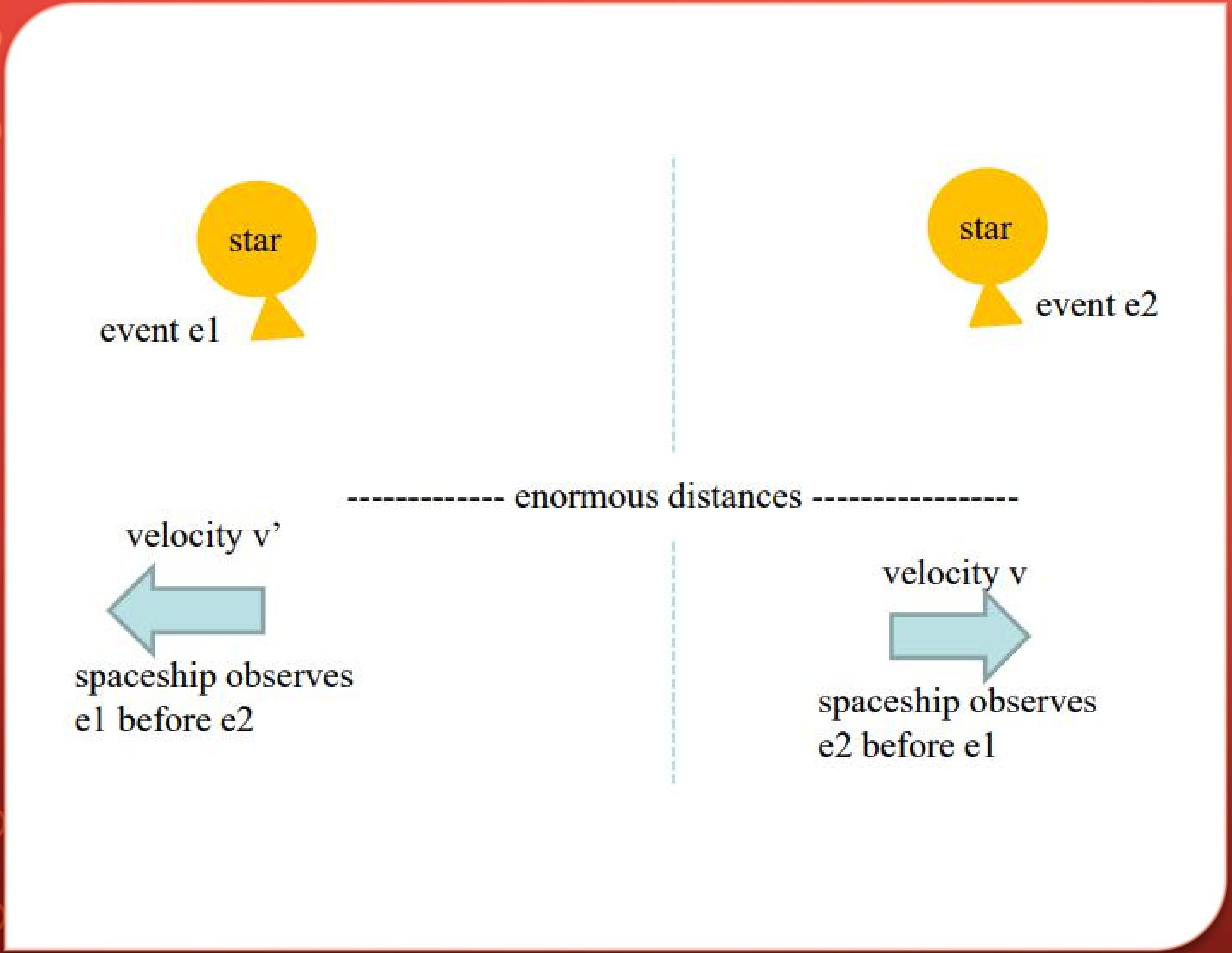
There is no common universal time (Einstein) but the speed of light is constant for all observers irrespective of their velocity



The spaceships observe

- e1 and e2 in different time frames
- different values for time elapsed between e1 and e2
- e1 and e2 (from the same source) in the same order.

TIME IN SPACE



PHYSICAL CLOCK

Based on frequency of oscillation of a quartz crystal

Each computer has a timer that interrupts periodically. Timer consists of two register

- Counter: counts the oscillation (decrement on each oscillation)
- Reset: stores the reset value of counter

Clock drift: in practice, the number of interrupts per hour varies slightly in the fabricated devices, also with temperature, so clocks may drift, typically 1/106 (1 sec in 11.6 days)

Timers can be set from transmitted UTC.

LEAP SECOND

UTC Date	UTC Time	Difference TAI vs. UTC
30/06/1972	23:59:60	11 secs
31/12/1972	23:59:60	12 secs
31/12/1973	23:59:60	13 secs
31/12/1974	23:59:60	14 secs
31/12/1975	23:59:60	15 secs
31/12/1976	23:59:60	16 secs
31/12/1977	23:59:60	17 secs
31/12/1978	23:59:60	18 secs
31/12/1979	23:59:60	19 secs
30/06/1981	23:59:60	20 secs
30/06/1982	23:59:60	21 secs

UTC Date	UTC Time	Difference TAI vs. UTC
30/06/1983	23:59:60	22 secs
30/06/1985	23:59:60	23 secs
31/12/1987	23:59:60	24 secs
31/12/1989	23:59:60	25 secs
31/12/1990	23:59:60	26 secs
30/06/1992	23:59:60	27 secs
30/06/1993	23:59:60	28 secs
30/06/1994	23:59:60	29 secs
31/12/1995	23:59:60	30 secs
30/06/1997	23:59:60	31 secs
31/12/1998	23:59:60	32 secs
31/12/2005	23:59:60	33 secs
31/12/2008	23:59:60	34 secs
30/06/2012	23:59:60	35 secs
30/06/2015	23:59:60	36 secs
31/12/2016	23:59:60	37 secs

PHYSICAL CLOCK SYNCHRONIZATION

- Cristian's Algorithm
- Berkeley Algorithm
- Network time protocol

CAUSAL ORDERING

- HB1: if ∃ process pi : e → $_i$e', then e → e'
- HB2: For any message m, send(m) → receive(m)
  - Where send(m) is the event of sending the message and receive(m) is the event of receiving it
- HB3: if e, e' and e'' are events such that e → e' and e' → e'' then e → e''

LOGICAL CLOCK (LAMPORT CLOCK)

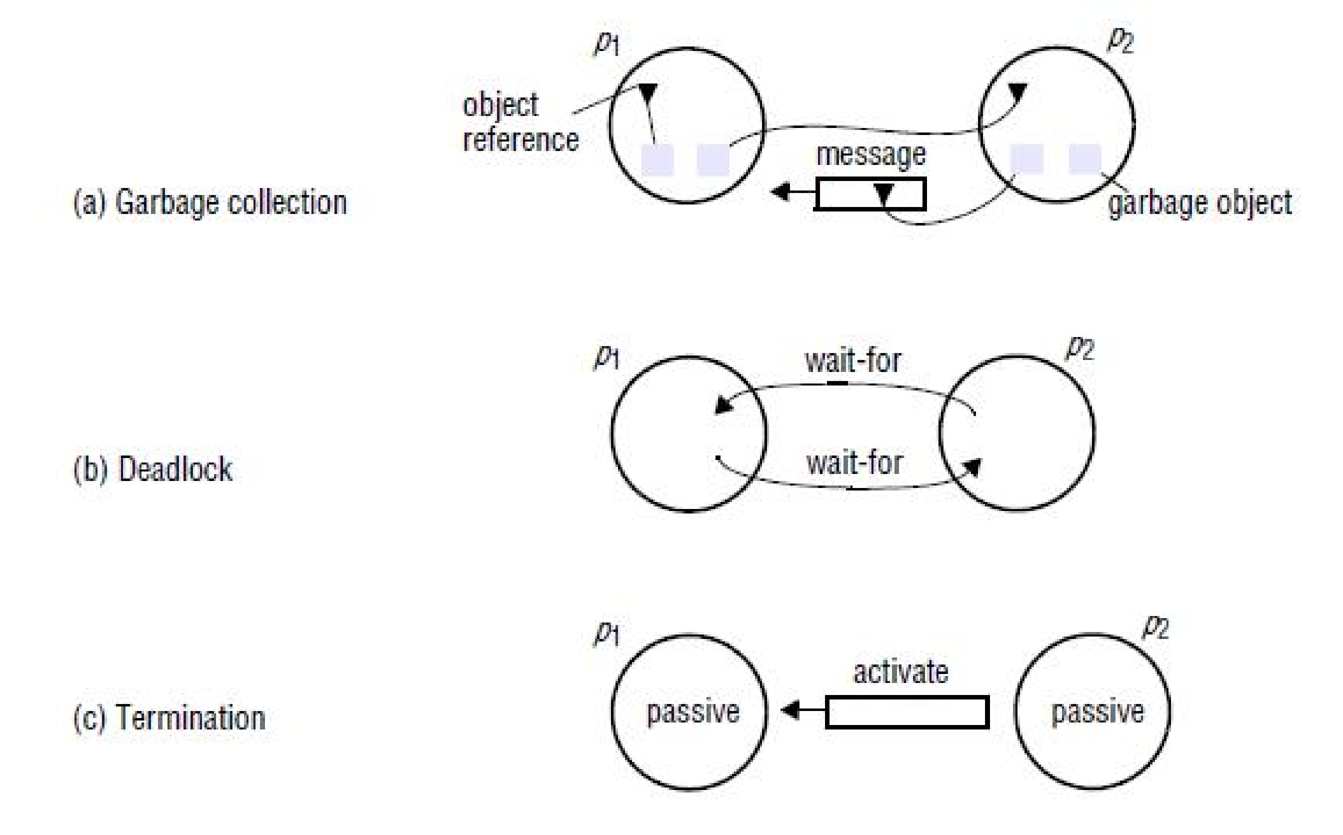
- LC1: L_i is incremented before each event is issued at process p_i: $L_i := L_i + 1$
- LC2:
  - When a process *pi* sends a message *m*, it piggybacks on *m* the value $t = L_i$.
  - On receiving (m, t), a process p_j computes $L_j := max(L_j, t)$ and then applies LC1 before timestamping the event *receive*(m).

VECTOR CLOCK

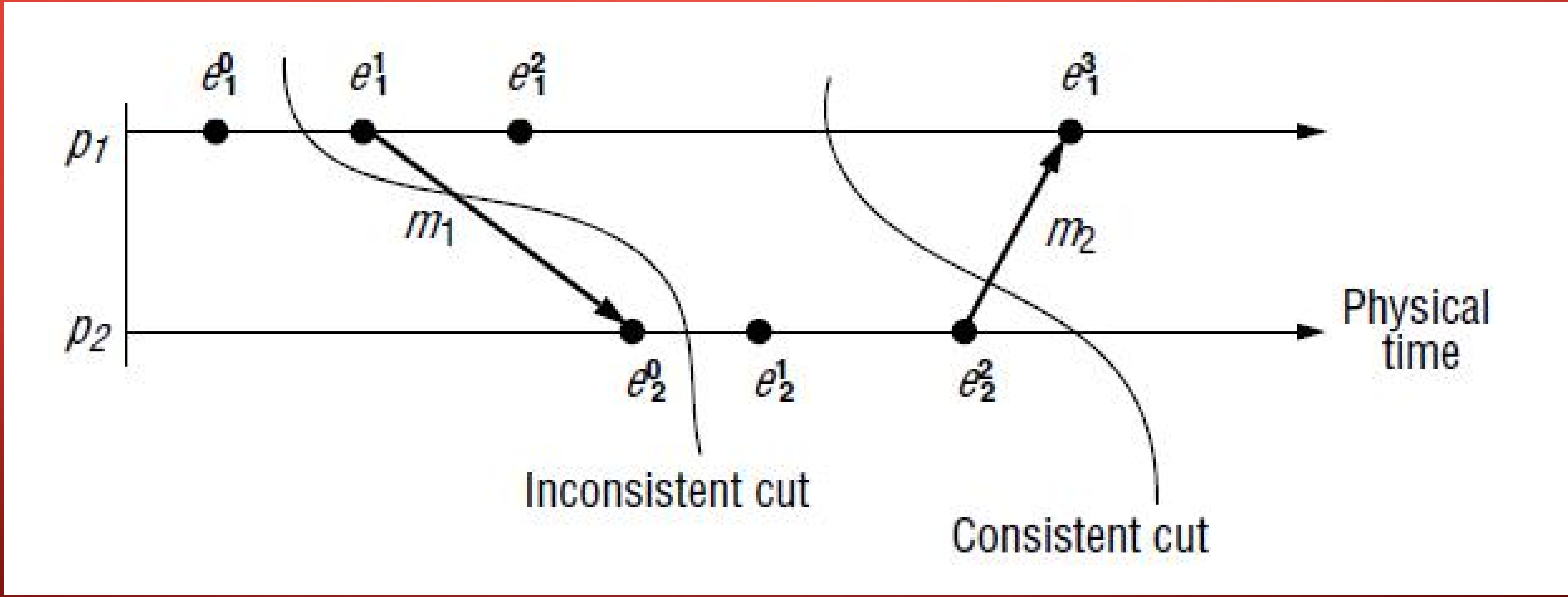
- VC1: Initially, $V_i[j] = 0$, for $i,j = 1,2\ldots,N$.
- VC2: Just before p_i timestamps an event, it sets $V_i[i] := V_i[i] + 1$.
- VC3: p_i includes the value $t = V_i$ in every message it sends.
- VC4: When p_i receives a timestamp t in a message, it sets $V_i[j] := max(V_i[j], t[j])$, for $j = 1,2\ldots,N$. Taking the component wise maximum of two vector timestamps in this way is known as a *merge operation*.

GLOBAL STATES

- Usages
  - *Distributed garbage collectio*
  - *Distributed deadlock detectio*
  - *Distributed termination detec*
  - *Distributed debugging*



CUTS



e_1^0
e_1^1
e_1^2
e_1^3
p_1
m_1
m_2
p_2
Physical time
e_2^0
e_2^1
e_2^2
Inconsistent cut
Consistent cut

‘SNAPSHOT’ ALGORITHM

By

- Chandy and Lamport

Assumptions

- Neither channels nor processes fail – communication is reliable so that every message sent is eventually received intact, exactly once.
- Channels are unidirectional and provide FIFO-ordered message delivery.
- The graph of processes and channels is strongly connected (there is a path between any two processes).
- Any process may initiate a global snapshot at any time.
- The processes may continue their execution and send and receive normal messages while the snapshot takes place.

ALGORITHM

Marker receiving rule for process p_i

On receipt of a *marker* message at p_i over channel c:

if (p_i has not yet recorded its state) it

records its process state now;

records the state of c as the empty set;

turns on recording of messages arriving over other incoming channels;

else

p_i records the state of c as the set of messages it has received over c since it saved its state.

end if

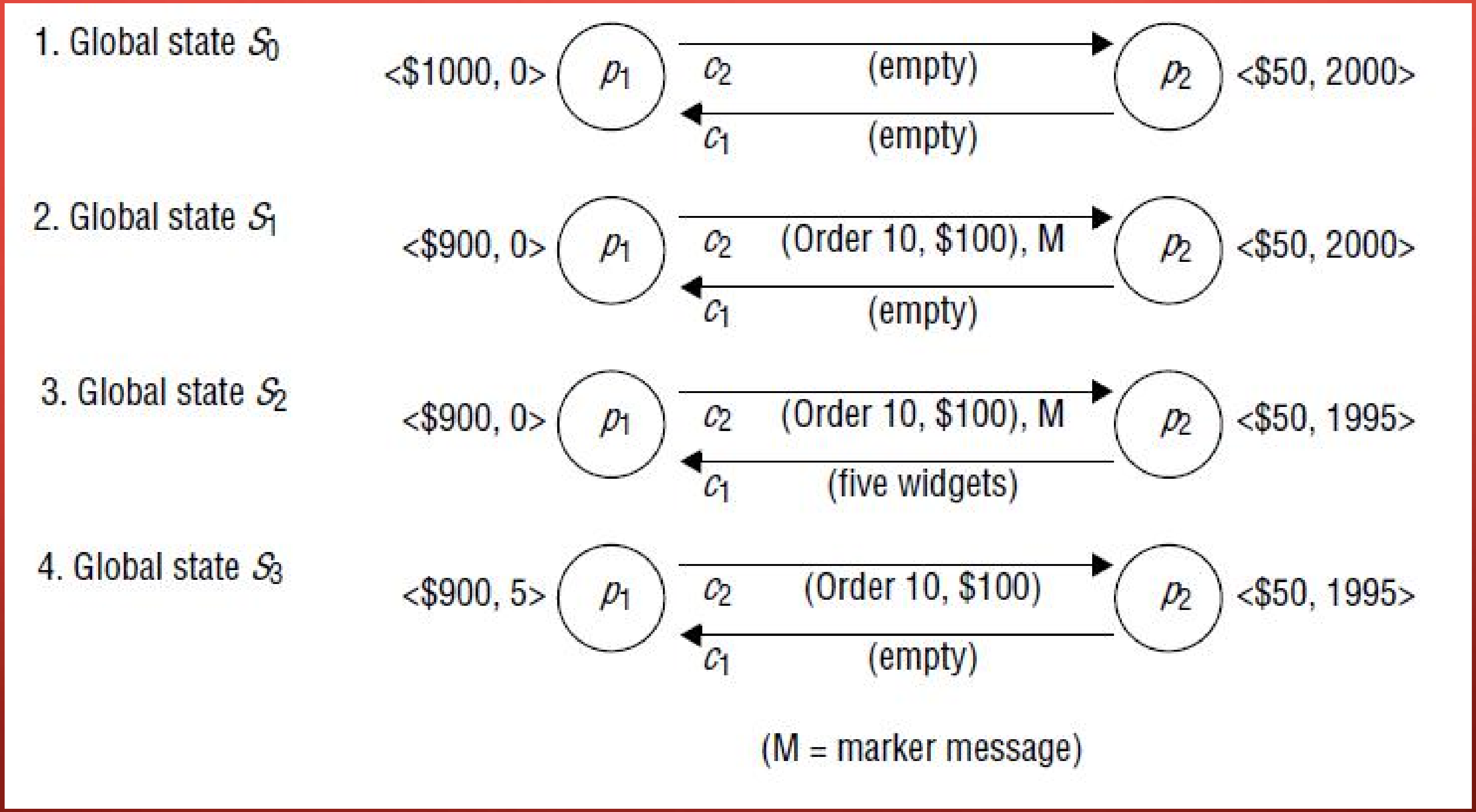
Marker sending rule for process p_i

After p_i has recorded its state, for each outgoing channel c:

p_i sends one marker message over c

(before it sends any other message over c).

EXAMPLE


1. Global state S_0
<$1000, 0>
p_1
c_2
(empty)
c_1
(empty)
p_2
<$50, 2000>
2. Global state S_1
<$900, 0>
p_1
c_2
(Order 10, $100), M
c_1
(empty)
p_2
<$50, 2000>
3. Global state S_2
<$900, 0>
p_1
c_2
(Order 10, $100), M
c_1
(five widgets)
p_2
<$50, 1995>
4. Global state S_3
<$900, 5>
p_1
c_2
(Order 10, $100)
c_1
(empty)
p_2
<$50, 1995>
(M = marker message)

THANK YOU

DILIP KUMAR SHRESTHA: GCES