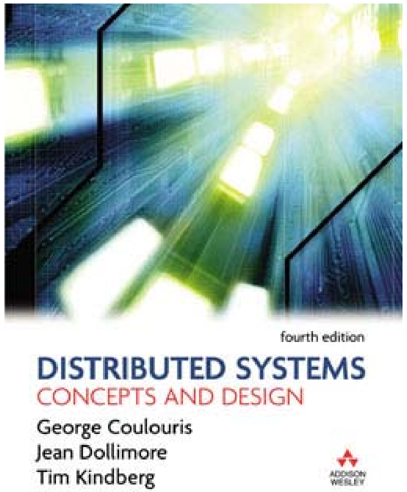
Slides for Chapter 11: Time and Global State



From Coulouris, Dollimore and Kindberg
Distributed Systems:
Concepts and Design

Figure 11.1
Skew between computer clocks in a distributed system

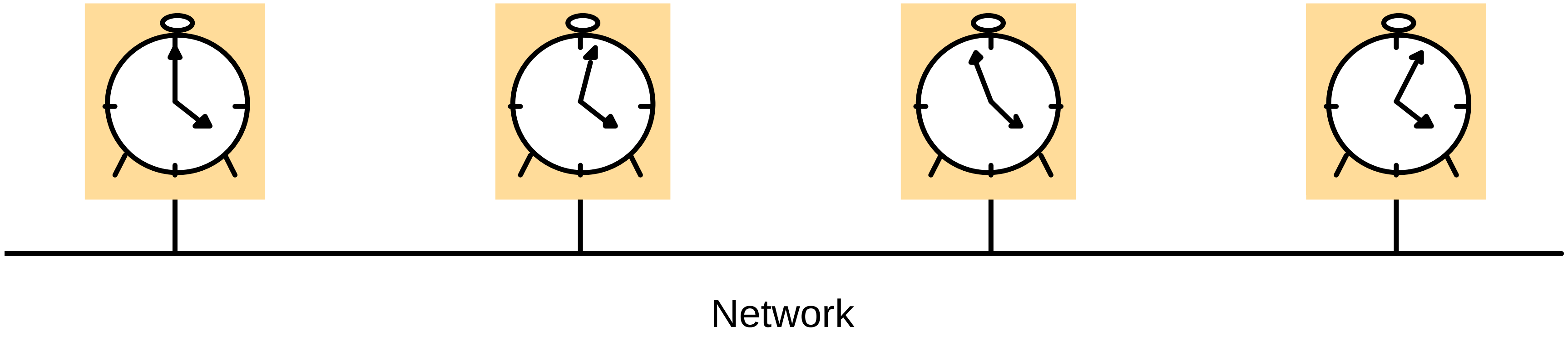


Figure 11.2
Clock synchronization using a time server

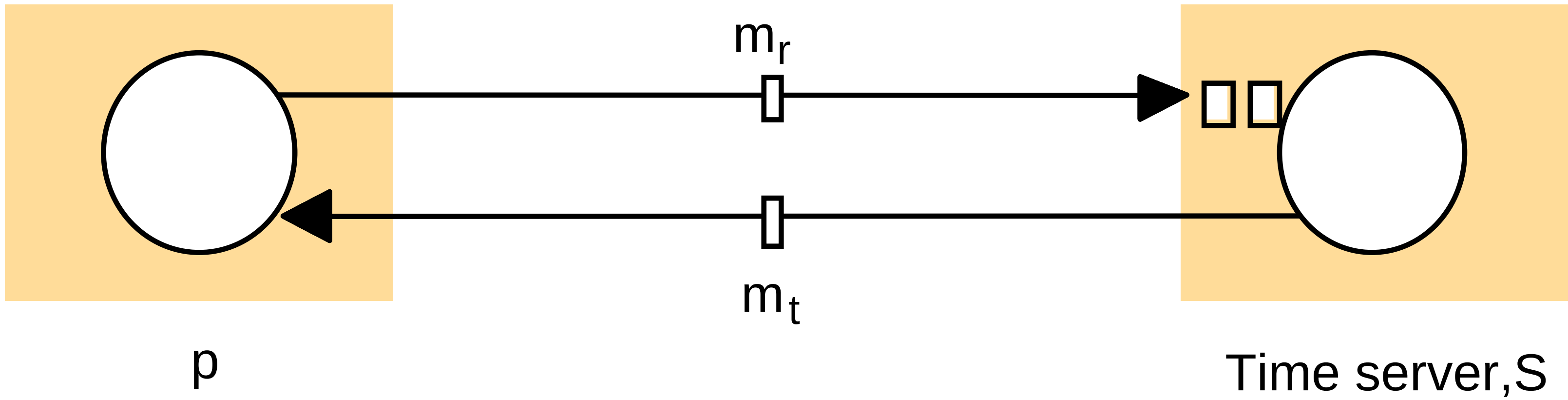
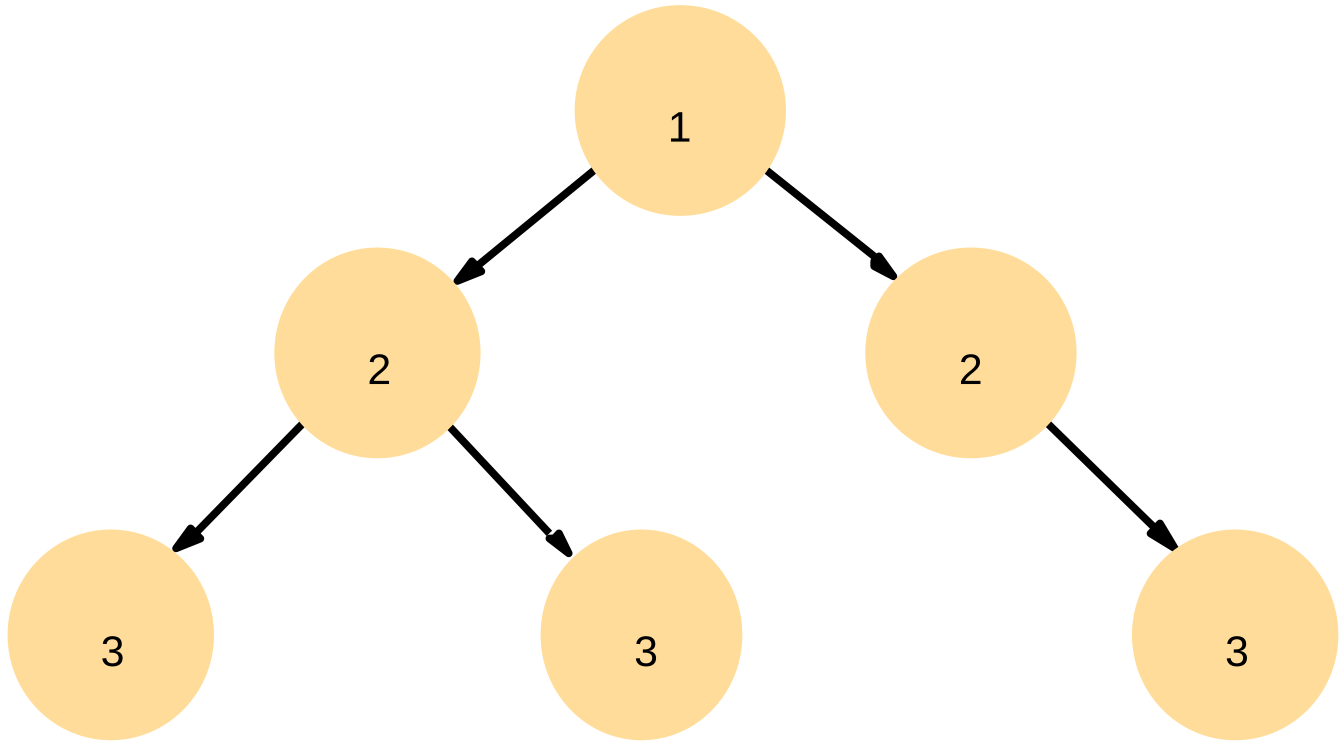


Figure 11.3
An example synchronization subnet in an NTP implementation



Note: Arrows denote synchronization control, numbers denote strata.

Figure 11.4
Messages exchanged between a pair of NTP peers

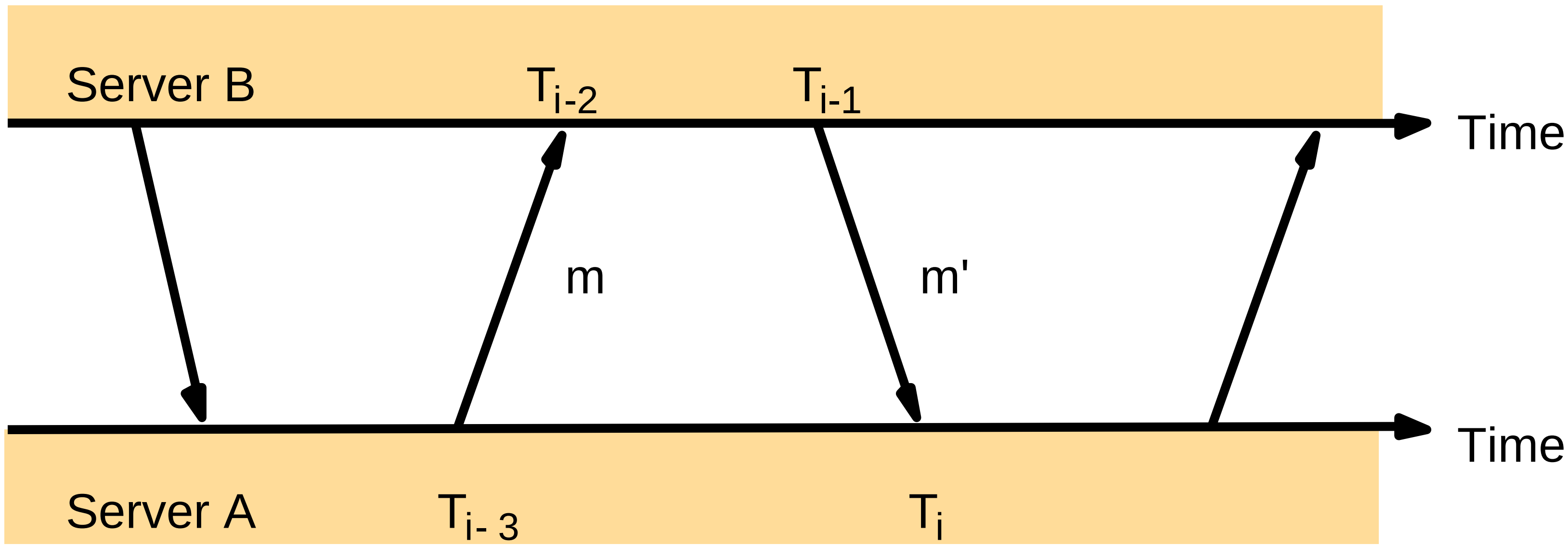


Figure 11.5
Events occurring at three processes

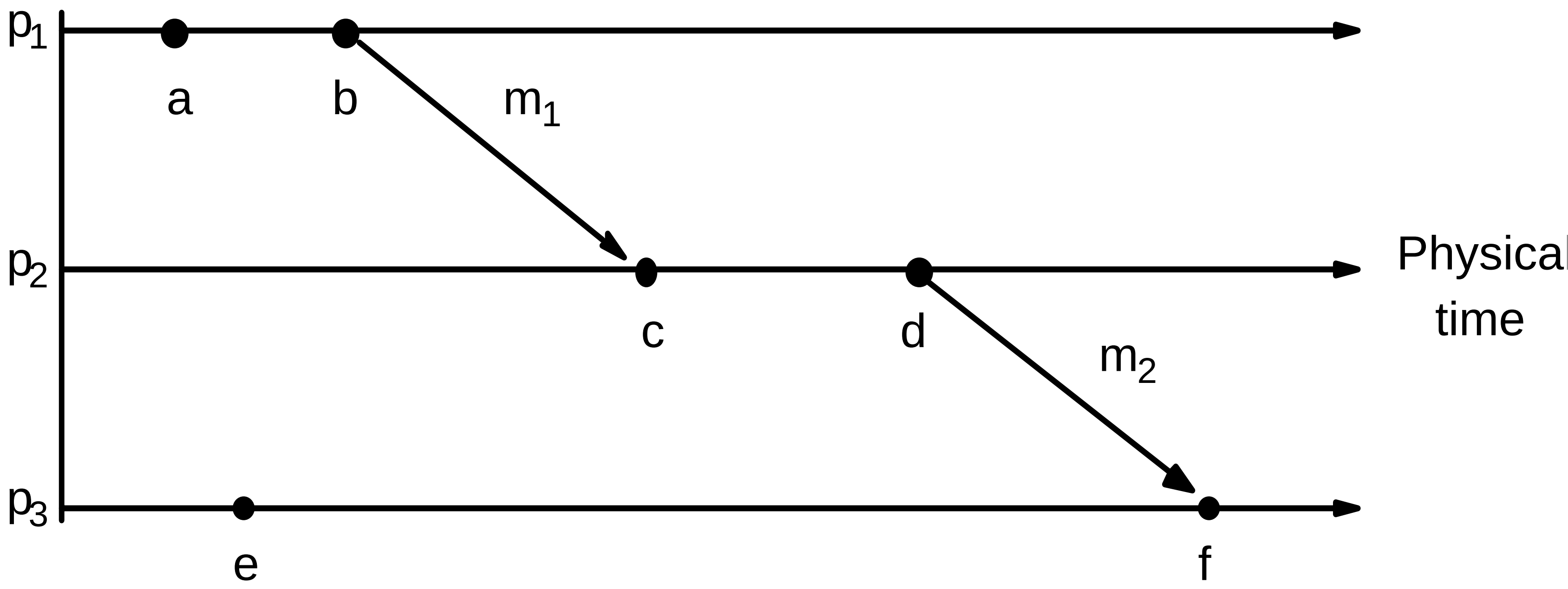


Figure 11.6
Lamport timestamps for the events shown in Figure 11.5

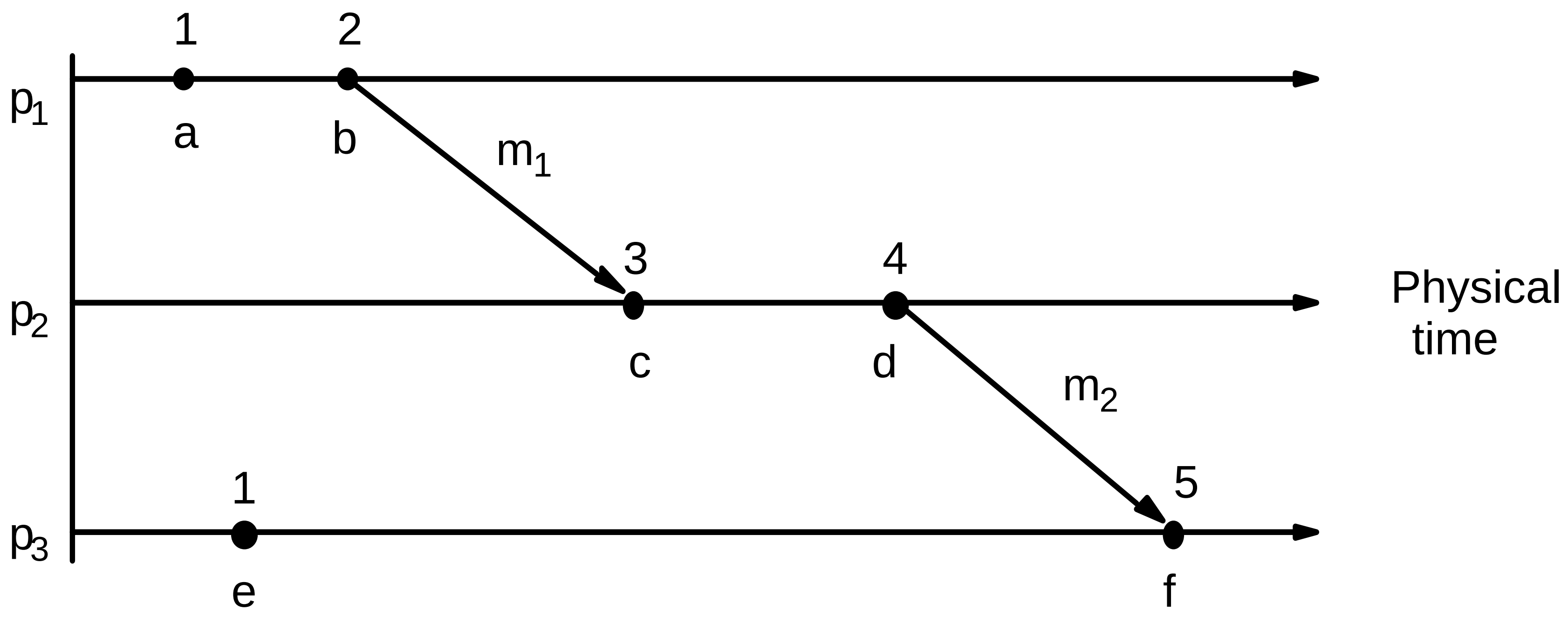


Figure 11.7
Vector timestamps for the events shown in Figure 11.5

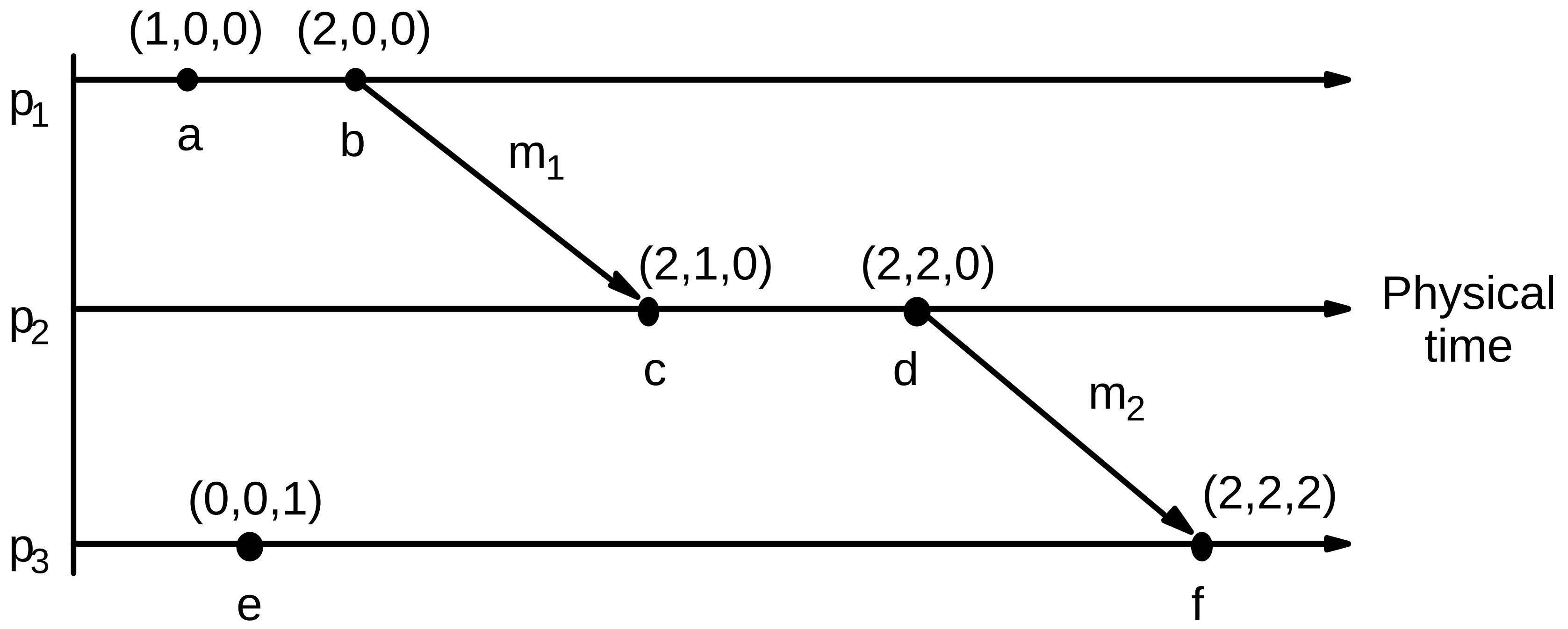
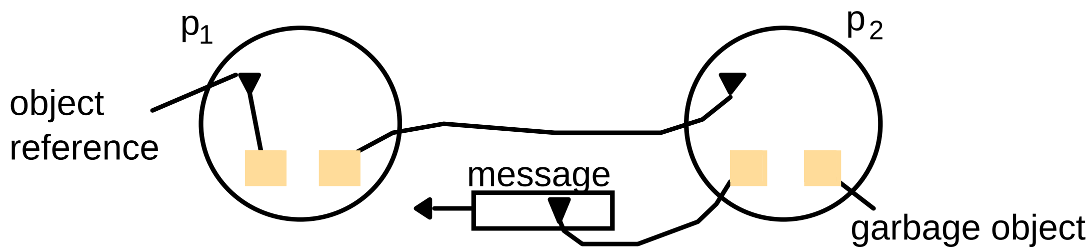
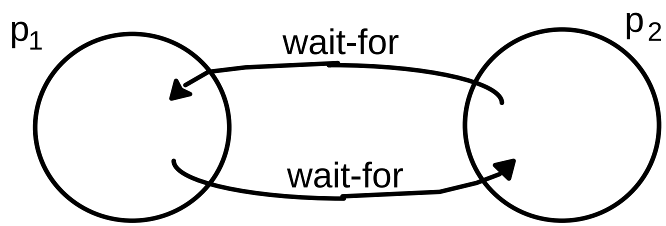


Figure 11.8
Detecting global properties

a. Garbage collection



b. Deadlock



c. Termination

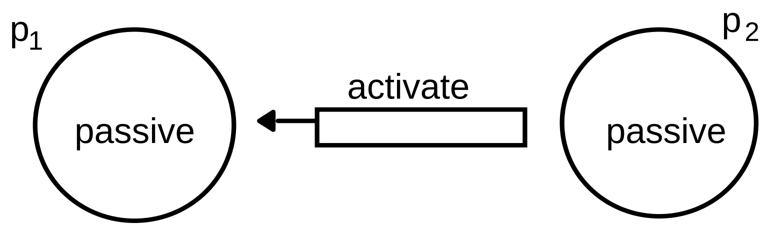


Figure 11.9
Cuts

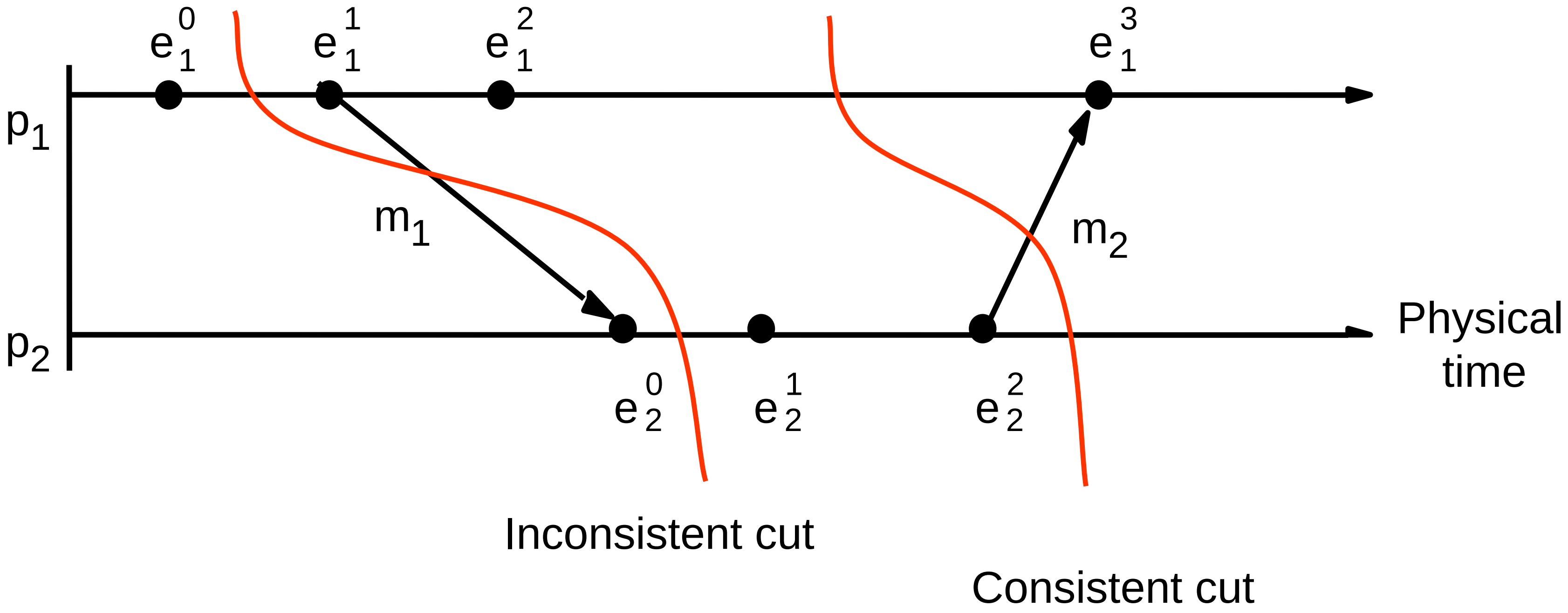


Figure 11.10
Chandy and Lamport's 'snapshot' algorithm

Marker receiving rule for process p_i

On p_i's receipt of a *marker* message 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).

Figure 11.11
Two processes and their initial states

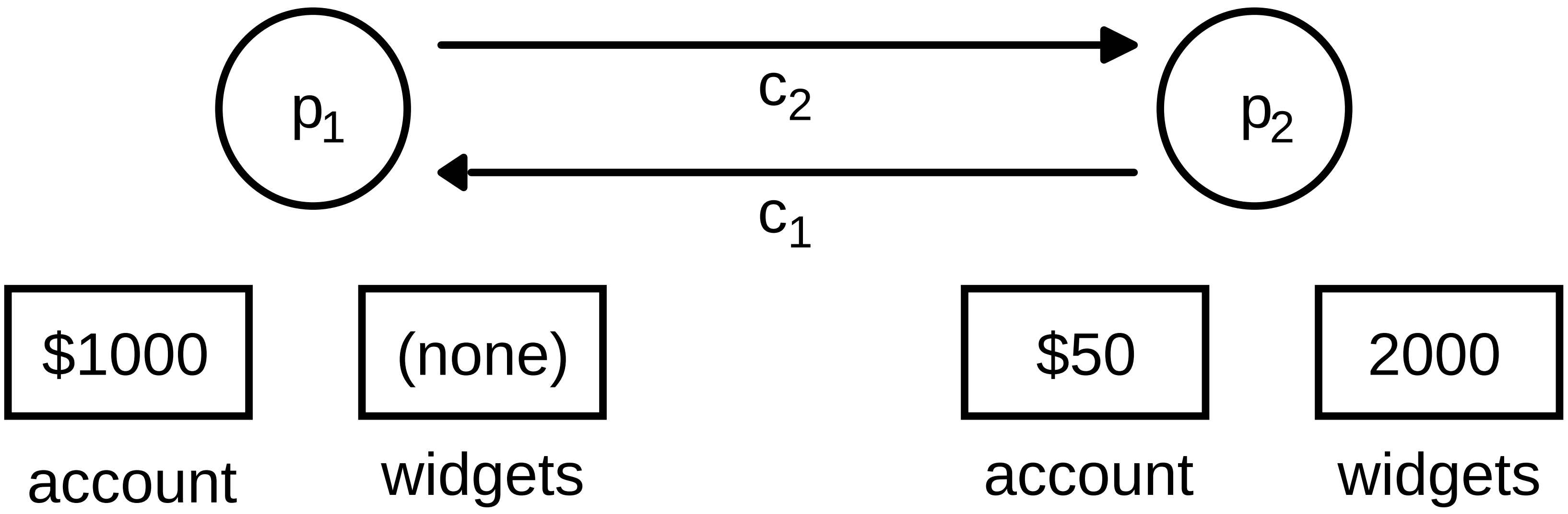


Figure 11.12
The execution of the processes in Figure 11.11

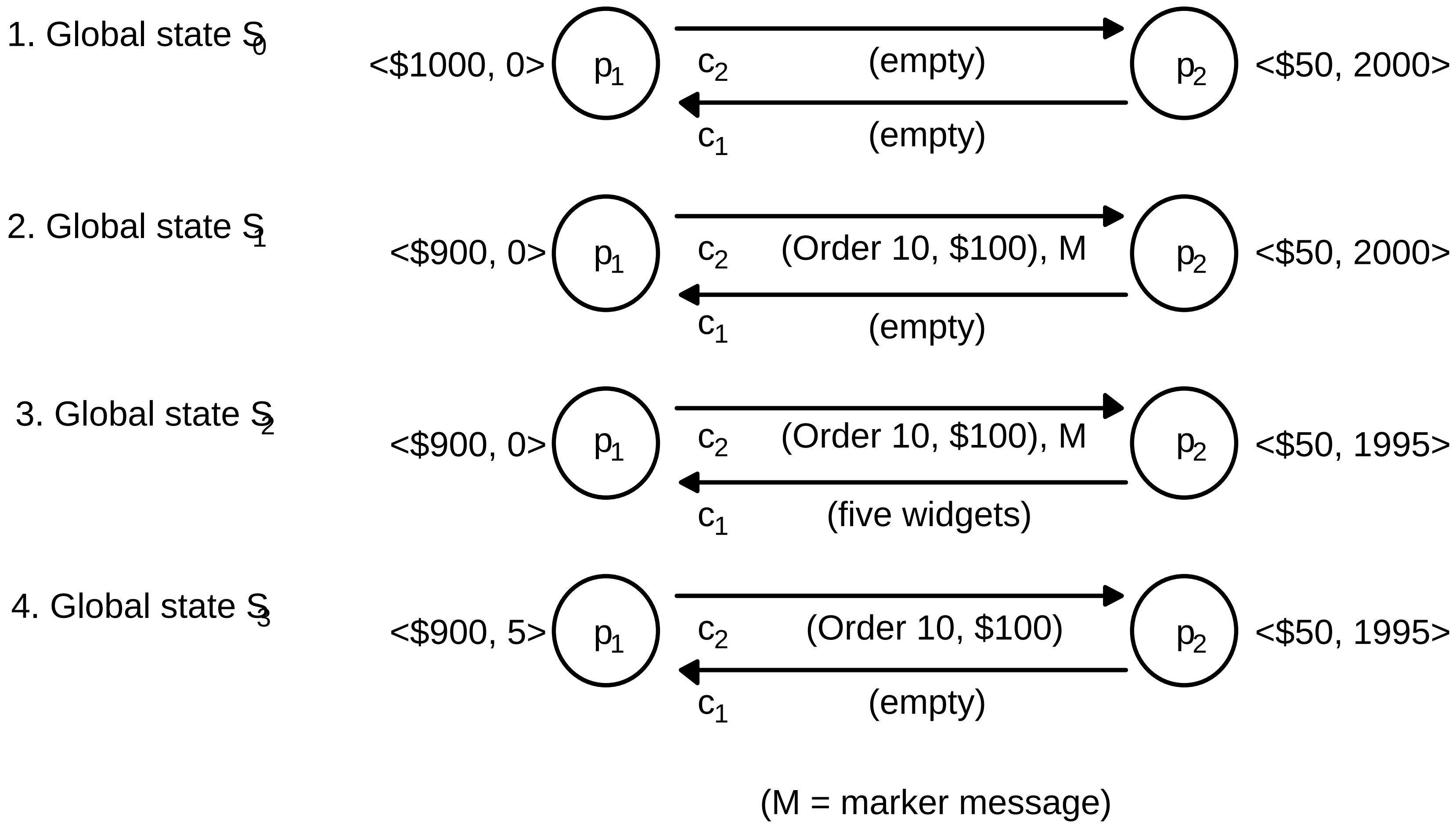


Figure 11.13
Reachability between states in the snapshot algorithm

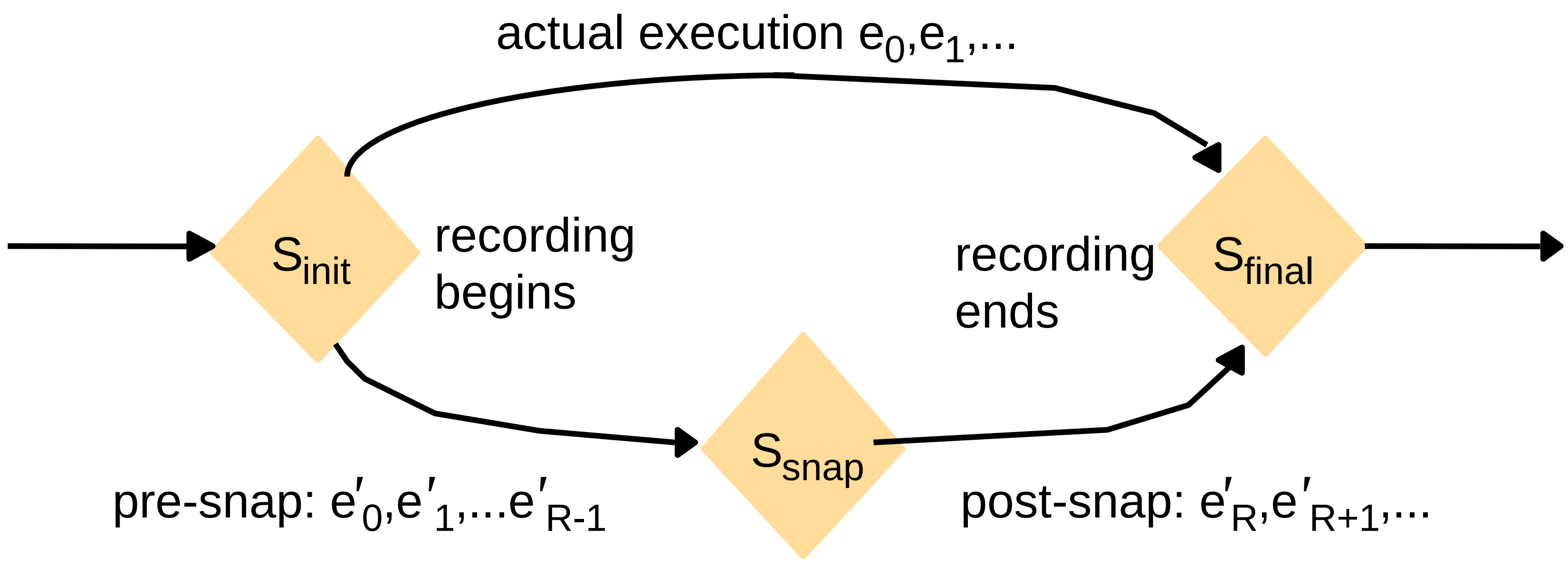


Figure 11.14
Vector timestamps and variable values for the execution of Figure 11.9

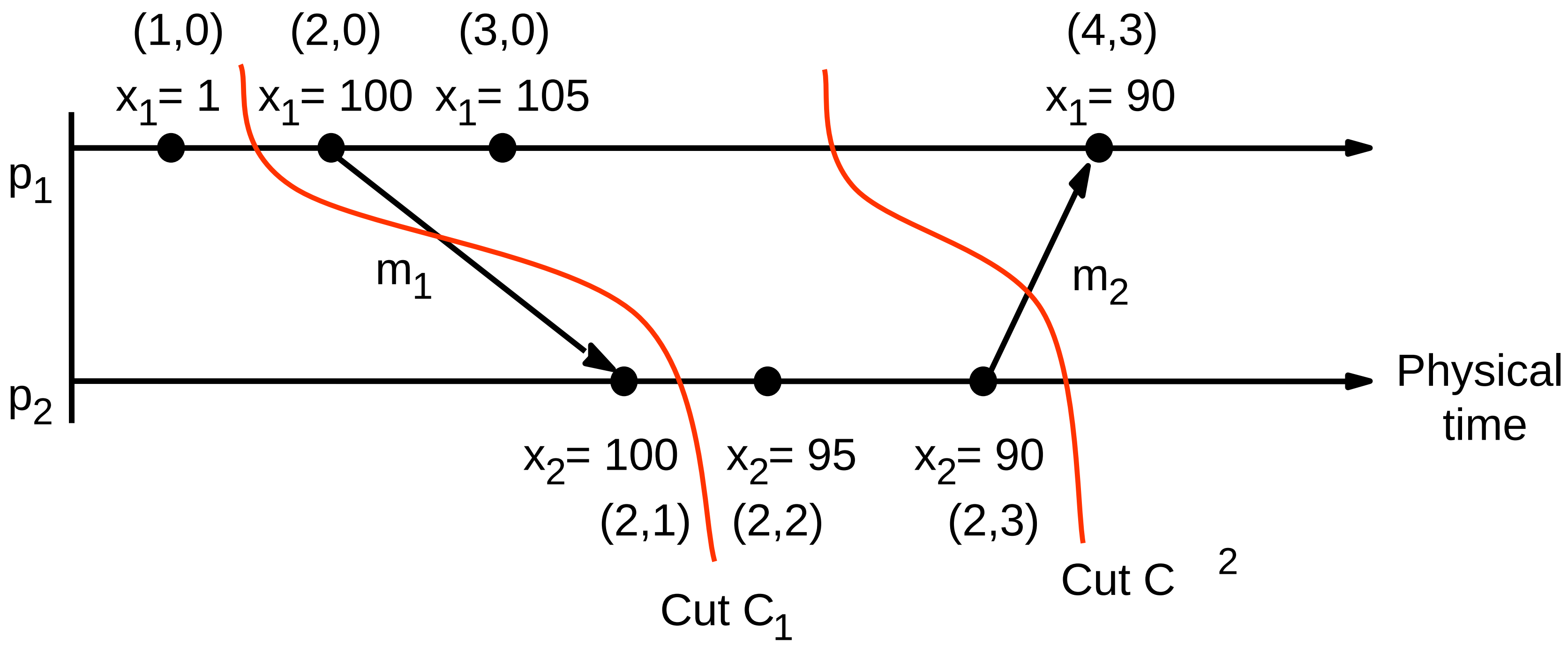
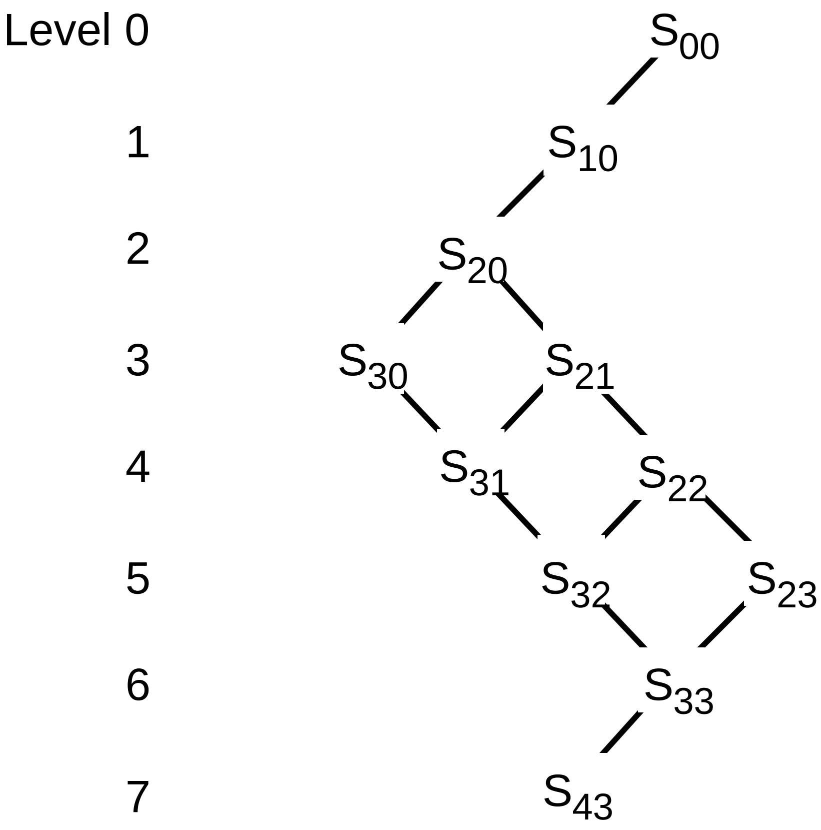


Figure 11.15
The lattice of global states for the execution of Figure 11.14



Sij = global state after *i* events at process 1 and *j* events at process 2

Figure 11.16
Algorithms to evaluate *possibly* ϕ and *definitely* ϕ

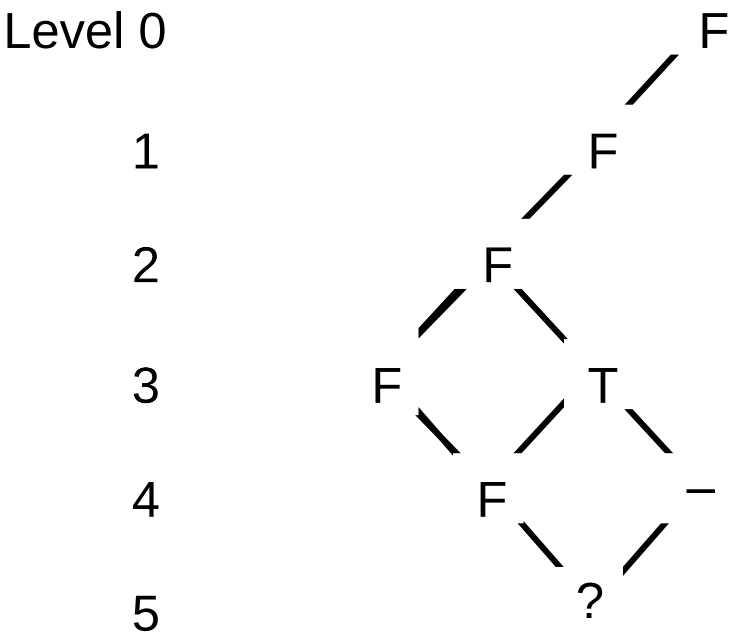
1. *Evaluating possibly* ϕ *for global history H of N processes*

$L := 0;$
$States := \{ (s_1^0, s_2^0, \ldots, s_N^0) \};$
$while\ (\phi(S) = False$ for all $S \in$ States$)$
  $L := L + 1;$
  $Reachable := \{ S' : S'$ reachable in H from some $S \in$ States $\wedge$ level$(S') =$ L$\};$
  $States := Reachable$
end while
output "*possibly* ϕ";

2. *Evaluating definitely* ϕ *for global history H of N processes*

$L := 0;$
$if\ (\phi(s_1^0, s_2^0, \ldots, s_N^0))\ then\ States := \{\}\ else\ States := \{ (s_1^0, s_2^0, \ldots, s_N^0) \};$
$while\ (States \neq \{\})$
  $L := L + 1;$
  $Reachable := \{ S' : S'$ reachable in H from some $S \in$ States $\wedge$ level$(S') =$ L$\};$
  $States := \{ S \in Reachable : \phi(S) = False \}$
end while
output "*definitely* ϕ";

Figure 11.17
Evaluating *definitely* ϕ



$F = (\phi(S) = \text{False}); T = (\phi(S) = \text{True})$