Distributed Information Processing

3rd Lecture

Eom, Hyeonsang (엄현상)
Department of Computer Science & Engineering
Seoul National University

Outline

- Clock and Global States
  - Global States
  - Determining Consistent Global States
- Q&A

Global States

- Prefix of Pi's History & Global History

$$h_i^k = < e_i^j \mid j = 1, \ldots, k >, \quad H = \bigcup_{i=1}^{N} h_i$$

- Cut & Frontier

$$C = \bigcup_{i=1}^{N} h_i^{C_i}, \quad F = \{e_i^{C_i} \mid i = 1, \ldots, N\}$$

Set of All Affected Values

- Global State (Corresponding to C)

$$S = \{s_i^{C_i} \mid i = 1, \ldots, N,\ s_i^{C_i} \text{ is } P_i\text{'s state immediately after } e_i^{C_i}\}$$

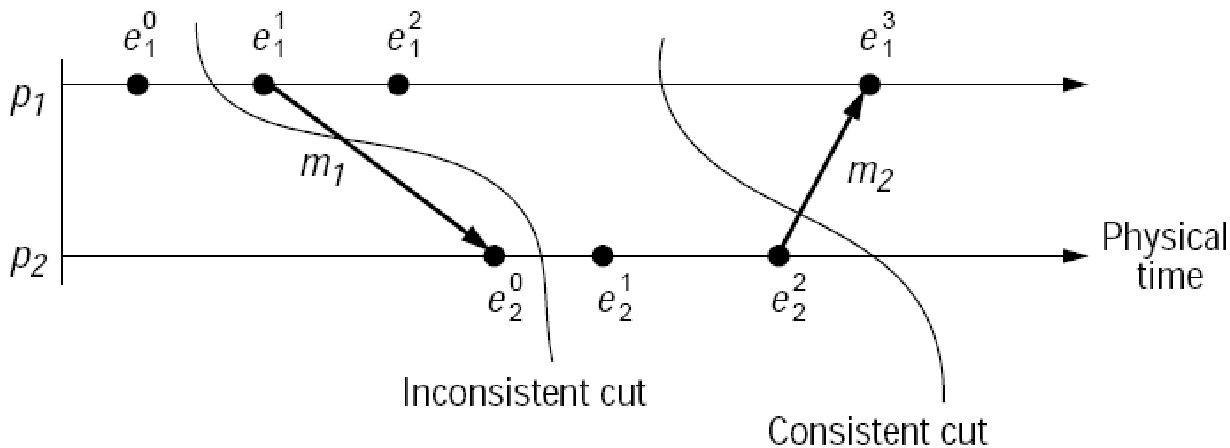
- Run: a Total Ordering in a Global History Consistent with Each Local History

Consistent Cuts & Runs

- C Is Consistent If the Following Holds:

$$\forall e \in C,\ e' \rightarrow e \ \Rightarrow\ e' \in C$$

H Is Consistent If the Corresponding C Is Consistent

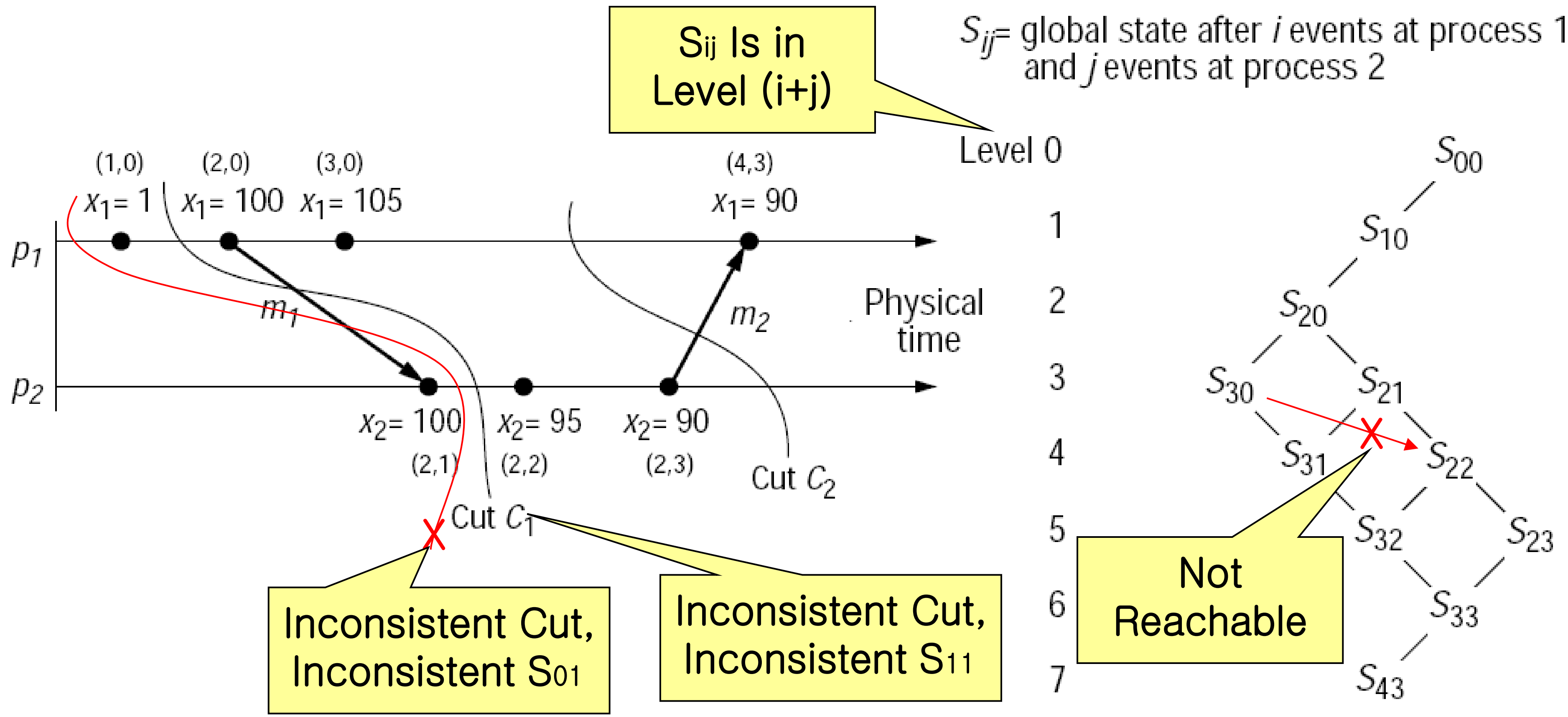


- Consistent Run: a Total Ordering in a Consistent Global History, Consistent with the Happened-Before Relation

Lattice of Global States

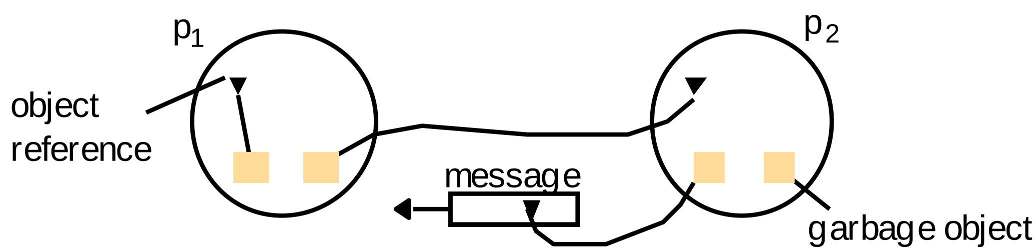
- Observing Consistent Global States

$$S = \{s_i \mid i = 1, \ldots, N\} \text{ Is Consistent iff } VC_i(s_i)[i] \geq VC(s_j)[i] \text{ for } i, j = 1, \ldots, N$$

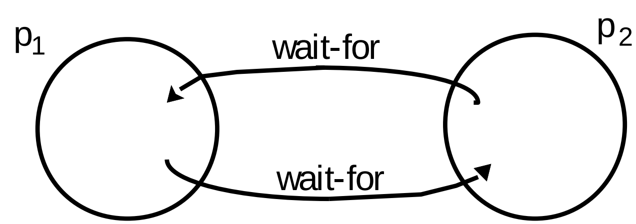


Detecting Global Properties

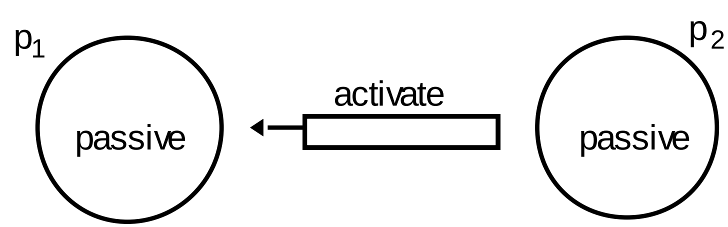
a. Garbage collection



b. Deadlock



c. Termination



Distributed 'Snapshot' Algorithm [Chandy85]

- Consistent Global-State Detection

Assumptions
- Reliable, Strongly-Connected Components
- Unidirectional Channels & In-Order Message Delivery

Marker to Record the State

Marker sending rule for process pi

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

pi sends one marker message over *c*

(before it sends any other message over *c*).

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;

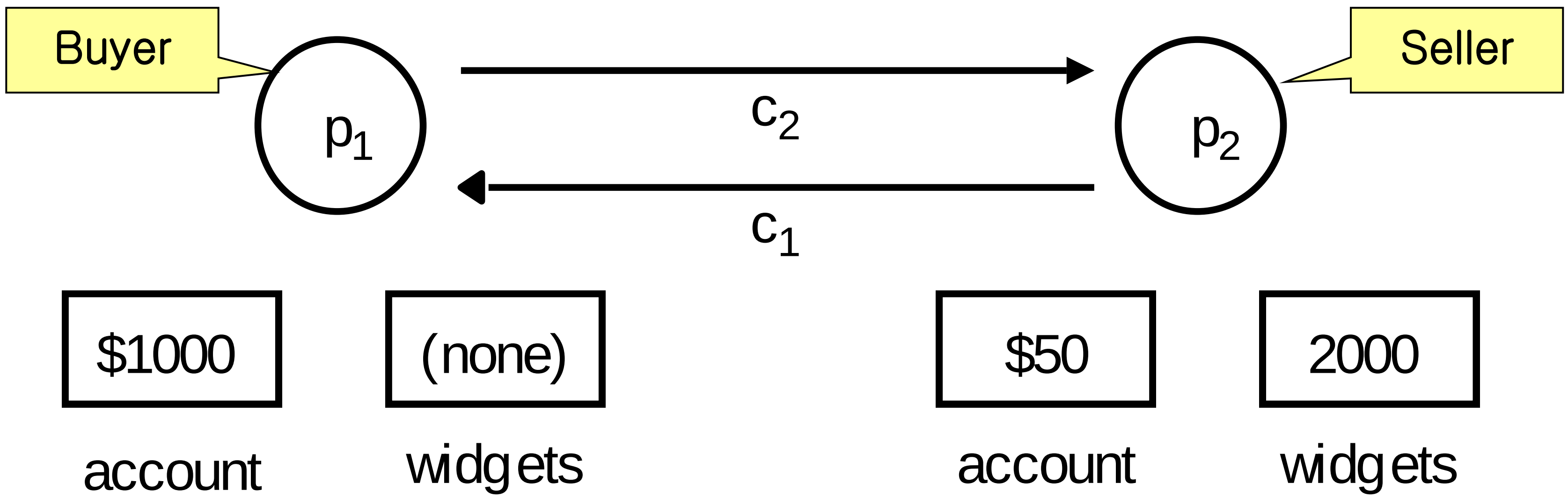
else

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

end if

Illustration: How the Alg. Works

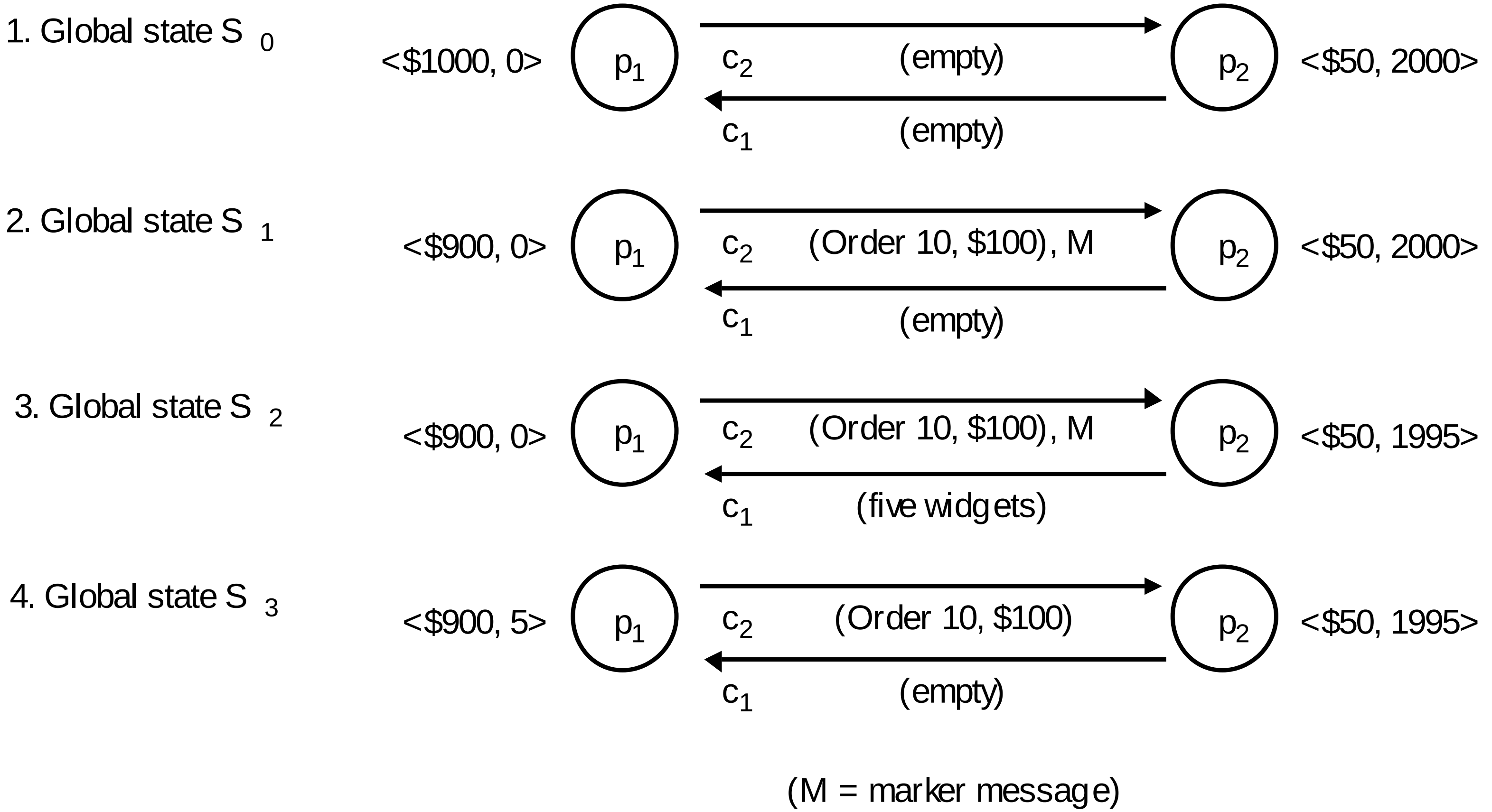
- Initial States of the Components



P2 Has Already Received an Order of Five Widgets

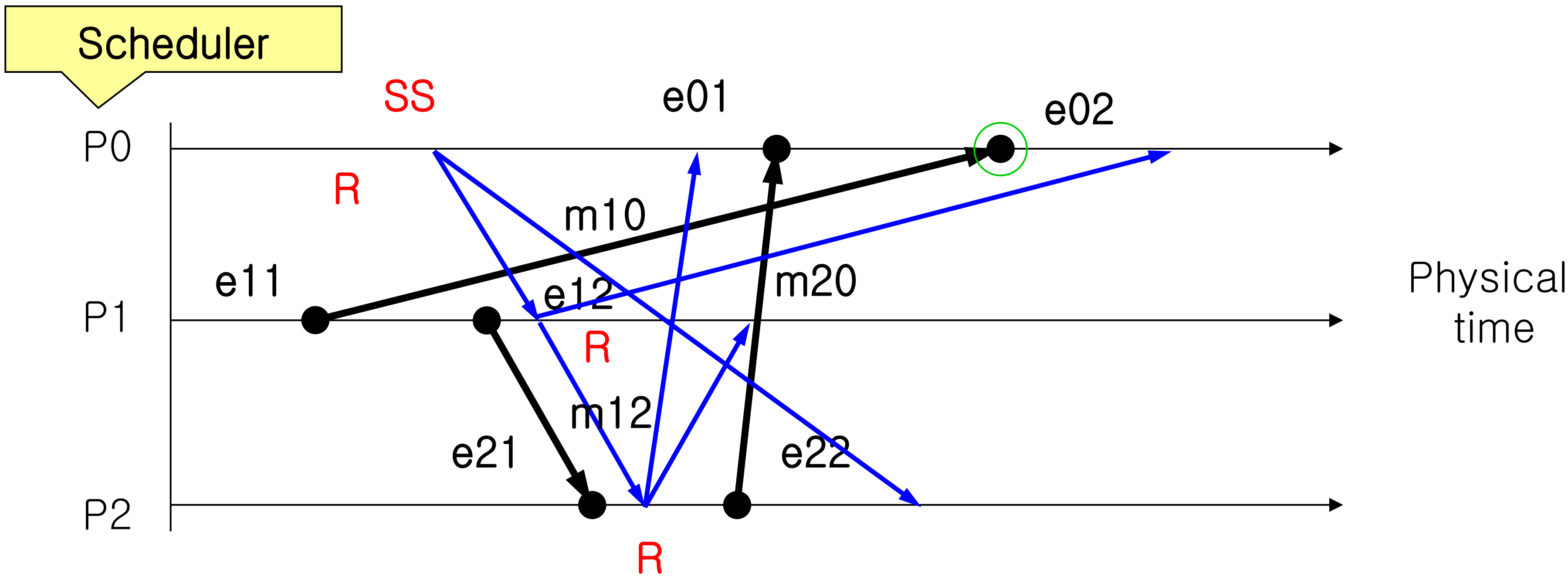
Illustration (Cont'd)

P2 Has Already Received an Order of Five Widgets



- S = p1: <$1000,0>; p2:<$50,1995>; c1: <(five widgets)>; c2:<>

Illustration w/ a Diagram



- SS: p0: <>; p1: <e11,e12>; p2:<e21>
  c01:<>; c02 <>; c10<m10>; c12 <>
  c20 <>; c21<>

Consistency Proof

- States Recorded by the Alg. Are Consistent:

$$\forall e_j \in C,\ e_i \rightarrow e_j \ \Rightarrow \ e_i \in C$$

$$Show:\ e_i \notin C,\ e_i \rightarrow e_j \ \Rightarrow \ e_j \notin C$$

i≠j

  - Assume That P_i Recorded Its State before e_i
  - Marker Would Have Reached P_j before the Message for e_j
  - P_j Would Have Recorded Its State before e_j

Characteristics of Snapshots

- Derivation of "Observed" Run from "Actual" Run

$$\text{"Actual" Run} : < e_i^k \mid i = 1, \ldots, N > = < e^j >$$

$$\text{"Permuted" Run} : < \ldots, e^{R-1}, e^R, \ldots >$$

Recodring

Observed (Consistent) Run

  - A Non-Observed Event May Occur before an Observed Event in the "Actual" Run
  - If a Non-Observed Event Precedes an Observed Event (Next to it) in the "Actual" Run, the Events Can Be Swapped Preserving Consistency

Global State Predicates

- Functions That Map Global States to True or False
  - Stable: Once True, It Remains True
    - E.g., deadlock or termination
  - Unstable: Not Stable
    - Possibly True: True At Some Point
      - E.g., snapshot by the 'Snapshot' Algorithm
    - Definitely True: True in All Cases