Outline

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

- **Prefix of Pi’s History & Global History**
  \[ h_i^k = \langle e_i^j \mid j = 1, \ldots, k \rangle, \quad H = \bigcup_{i=1}^{N} h_i \]

- **Cut & Frontier**
  \[ C = \bigcup_{i=1}^{N} h_i^{C_i}, \quad F = \{ e_i^{C_i} \mid i = 1, \ldots, N \} \]

- **Global State (Corresponding to C)**
  \[ S = \{ s_i^{C_i} \mid i = 1, \ldots, N, s_i^{C_i} \text{ is } P_i's \text{ state immediately after } e_i^{C_i} \} \]

- **Run: a Total Ordering in a Global History Consistent with Each Local History**
Consistent Cuts & Runs

C Is Consistent If the Following Holds:

\[ \forall e \in C, \quad e' \rightarrow e \Rightarrow e' \in C \]

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

\[ S = \{ s_i \mid i = 1, \ldots, N \} \] is consistent iff \( VC_i(s_i)[i] \geq VC(s_j)[i] \) for \( i, j = 1, \ldots, N \).

- \( S_{ij} \) is in Level \((i+j)\)

\( S_{ij} \) is the global state after \( i \) events at process 1 and \( j \) events at process 2.

Inconsistent Cut, Inconsistent \( S_{01} \)

Inconsistent Cut, Inconsistent \( S_{11} \)

Not Reachable

Coulouris, Dollimore and Kindberg   Distributed Systems: Concepts and Design   Edn. 4   © Pearson Education 2005
Detecting Global Properties

a. Garbage collection

b. Deadlock

c. Termination
Distributed ‘Snapshot’ Algorithm

[Chandy85]

Consistent Global-State Detection

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 $c$).

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

$\text{P}_2$ Has Already Received an Order of Five Widgets
Illustration (Cont’d)

1. Global state $S_0$

$$p_1 \xrightarrow{c_2} (\text{empty}) \xrightarrow{c_1} p_2$$

2. Global state $S_1$

$$p_1 \xrightarrow{c_2} (\text{Order 10, $100$}, M) \xrightarrow{c_1} p_2$$

3. Global state $S_2$

$$p_1 \xrightarrow{c_2} (\text{Order 10, $100$}, M) \xrightarrow{c_1} p_2$$

4. Global state $S_3$

$$p_1 \xrightarrow{c_2} (\text{Order 10, $100$}) \xrightarrow{c_1} p_2$$

$(M = \text{marker message})$

$S = p_1: <$1000, 0$>; p_2: <$50, 1995$>; c_1: <(five widgets)>; c_2: <>$

P2 Has Already Received an Order of Five Widgets
**Illustration w/ a Diagram**

- **SS**: p∅: <>; p1: <e11,e12>; p2: <e21>
- **c∅1**: <>; c∅2: <>; c1∅: <m1∅>; c12: <>
- **c2∅**: <>; 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: \( e_i \notin C, \ e_i \rightarrow e_j \Rightarrow e_j \notin C \)

- 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

"Actual" Run: \( <e_i^k | i = 1, ..., N > = <e_j^j > \)

"Permutated" Run: \(<,..., e_i^{R-1}, e_i^R, ... >\)

- 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