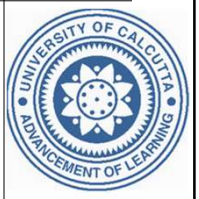
Clock Models

Clock Models for Distributed Systems



1

Event Precedence

- Suppose Marek gives a pen to Dariusz – say giving the pen is event P and receiving the pen is event K.
- Let's also assume that Marek has taken his breakfast (event A) before he gave the pen to Dariusz and Dariusz goes for a movie (event B) after he gets the pen
- Apparently, events A and B are not in any sequence.

3 May 2024

2

Event Precedence

- However, here
  - P *precedes* K, A *precedes* P and K *precedes* B

    $P \longrightarrow K,\quad A \longrightarrow P,\quad K \longrightarrow B$
  - i.e.,

    $A \longrightarrow P,\quad P \longrightarrow K,\quad K \longrightarrow B$
  - A (Marek taking his -B'fast) precedes B (Dariusz going for the movie)

Event Precedence

- Event A *precedes* B, if ∃ pair of events P and K, such that
  - P *precedes* K
  - A *precedes* P ∨ (A ‖ P)
  - K *precedes* B ∨ (K ‖ B)
- In a similar way, one may define conditions when an event A *follows* another event B

Event Precedence

- Events with causal associations have implicit precedence in the sense that
  - cause *precedes* effect
- Cause-effect association provides the base cases for event precedence

3 May 2024

5

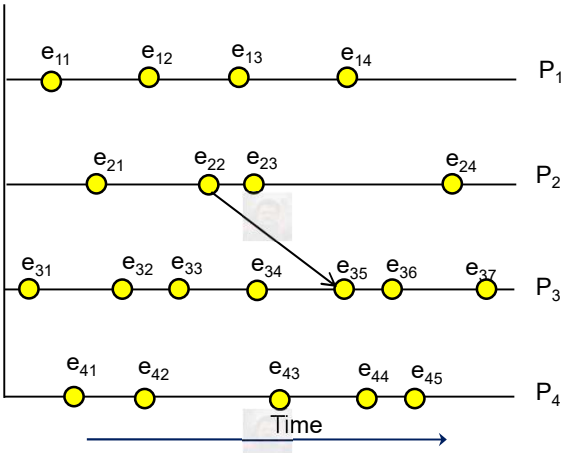
Concurrent Events

- A pair of events A and B are said to be concurrent if neither of the two events follows or precedes the other.
- Thus, two events A and B are concurrent does not necessarily imply that A and B are bring executed simultaneously.

3 May 2024

6

Event Trace Diagram



3 May 2024

7

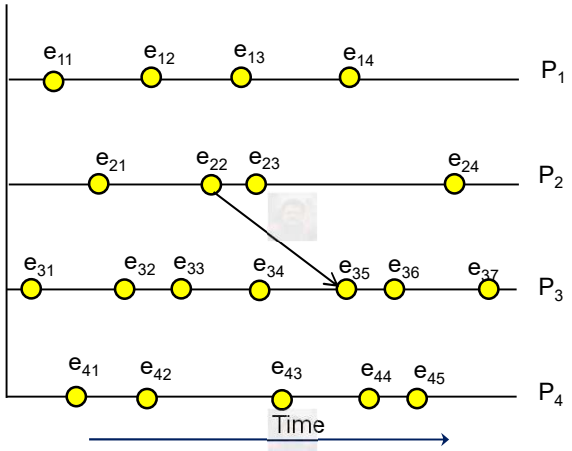
Precedence in Event Trace Diagram

- e_{22} *precedes* e_{35}
- e_{21} *precedes* e_{35}
- e_{21} *precedes* e_{36}
- e_{21} *precedes* e_{32} ??
- e_{31} *precedes* e_{24} ??

3 May 2024

8

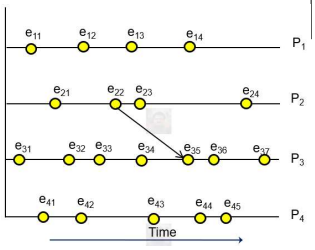
Event Trace Diagram



3 May 2024

9

Precedence in Event Trace Diagram



- e_{22} *precedes* e_{35}
- e_{21} *precedes* e_{35}
- e_{21} *precedes* e_{36}
- e_{21} and e_{32} are mutually concurrent
- e_{31} and e_{24} are mutually concurrent

3 May 2024

10

Synchronization using Cause-Effect

- Sending and receipt of message is accepted model for cause-effect relationship
- Such pair of events may in fact be used for synchronization of clocks in different nodes
- How?

3 May 2024

11

Lamport's Conditions for the Clock

- [C1] For any two events A and B in a process P_i, if A occurs before B then

$$C_i(A) < C_i(B)$$

- [C2] If A is the event of sending message m in process P_i and B is the event of receiving the same message m at process P_k, then

$$C_i(A) < C_k(B)$$

3 May 2024

12

Synchronization using Cause-Effect

- Record the time-stamp T_S at source for the message sending event
- Send T_S along with the message
- Record the time-stamp T_D on receipt of the message at the destination
- If $T_D > T_S$, clocks at source versus destination nodes are already synchronized

3 May 2024

13

Synchronization using Cause-Effect

- If $T_D \leq T_S$, then
  - Synchronization problem between source and destination
  - Clocks are relatively incorrect
- Reset the clock at the destination with the value $T_S + \partial$, where ∂ is some arbitrary value representing average transmission delay
- This keeps things **consistent** at the two ends

3 May 2024

14

Lamport's Implementation Rules

- [IR1] Clock C_i is incremented between any two successive events in a process P_i

  $C_i := C_i + \partial$

  If A and B are two successive events in P_i, then $C_i(B) = C_i(A) + \partial$

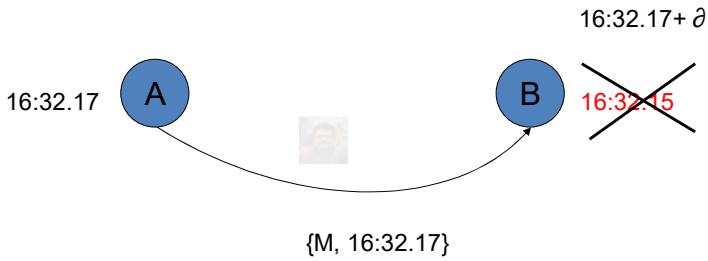
- [IR2] If A is the event of sending a message m in process P_i and B is the event of receiving the same message m at process P_k, then

  $C_k := \max(C_k, C_i(A) + \partial)$

3 May 2024

15

Synchronization using Cause-Effect



3 May 2024

16

Synchronization using Cause-Effect

- What if the clock at the target was in fact correct?
- May be clock in the source is running fast!!
- How do you know that?
- Can we adjust the clock at source to keep things logically **consistent**?

3 May 2024

17

Clock or Counter?

- One of the major aspects of synchronization following cause-effect association between a pair of events is that, you are just trying to keep things **consistent**
- The real-time is not preserved anymore
- The clocks are being adjusted with arbitrary values of ∂ for the sake of relative correctness
- If it's all about playing with numbers, lets play with integers!!!

3 May 2024

18

Clock or Counter?

- At the beginning of a new session, in each participating node P_K in the system, the local clock C_K is initialized with 0.
- Each event that occurs in a node increases the clock by 1
  - $C_K = C_K + 1$, for every event in P_K
- Events are not of equal granularities
- This clock model is referred as logical clock

3 May 2024

19

Logical Clocks

- Key ideas:
  - Clock synchronization need not be absolute
  - If two machines do not interact, no need to synchronize them
  - More importantly, processes need to agree on the ***order*** in which events occur rather than the ***time*** at which they occur

3 May 2024

20

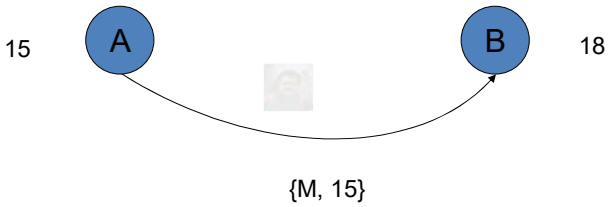
Synchronization for Logical Clock

- Record the time-stamp T_S at source
- Send T_S along with the message M
- Record the time-stamp T_D at the destination
- If $T_D > T_S$, clocks do not require to be synchronized
- If $T_D \leq T_S$, then reset the clock at destination with $T_S + 1$ after the receipt of the message
- Keeps things **consistent** at the two ends

3 May 2024

21

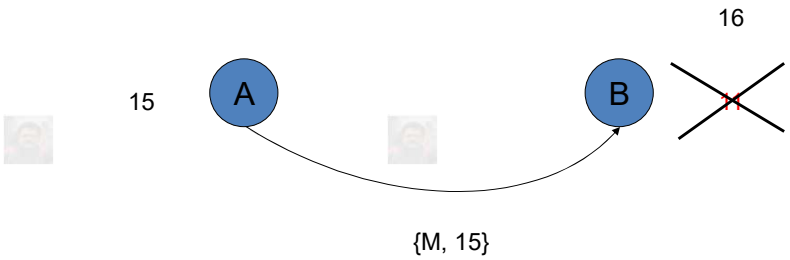
Synchronization using Cause-Effect



3 May 2024

22

Synchronization for Logical Clock



3 May 2024

23

Logical Clock and Complete Ordering

- It may be quite realistic to assume that each participating node would have a unique site Id.
- A logical clock would effectively be a doublet
  - <local clock, site id>
- When two different time stamps are to be compared, the one with lower local clock value is to be treated as smaller
- When the two local clock values are identical, the one with smaller Id is treated as smaller

3 May 2024

24

Logical Clock and Complete Ordering

- <3,8> is smaller than <5,4>
- <3,6> is smaller than <3,7>
- The purpose is to break the tie
- It is considered that the OS must remain consistent even in being unfair!
- When the two local clock values are identical, the one with smaller Id is treated as smaller
- Both the components cannot be identical for any two time-stamps in the system

25

Limitation of Lamport's Clock

- In Lamport's logical clocks,
  - if A → B, then C(A) < C(B)
  - However, the reverse is not necessarily true
  - Lamport's clock is not powerful enough to reason about the HB relationship between two events of different sites by just looking at the timestamps of the two events
- Lamport's clock cannot distinguish between the advancement of clocks due to local events from those due to the exchange of message events

26

Thanks for your kind attention

Questions??