Synchronization

Introduction

- Sharing system resources among multiple concurrent processes may be:
  - Cooperative or competitive in nature.
- Both require adherence to certain rules of behavior:
  - For enforcing correct interaction.

This requires synchronization mechanisms.

Synchronization-related issues

1. Clock synchronization
2. Event ordering
3. Mutual exclusion
4. Deadlock
5. Election algorithms

Clock synchronization

- Requirement of a timer mechanism for every computer:
  - To keep track of current time,
  - For various accounting purposes,
  - For correct results, and
  - helps in measuring the duration of distributed activities.

Implementation of computer clocks

- Components of a computer clock:
  1. A quartz crystal
     - Oscillates at a well-defined frequency.
  2. A counter register
     - Keeps track of the oscillations of the quartz crystal.
     - If value is zero, an interrupt is generated and reinitialized to the value of constant register.
  3. A constant register
     - Stores a constant value, based on the frequency of oscillation of the quartz crystal.

Cont…

- Each interrupt is known as a *clock tick.*
- The value in the constant register is chosen so that 60 clock ticks occur in a second.
- The computer clock is synchronized with real time.

Drifting of clocks

- A clock always runs at a constant rate.
  - There may be difference in the crystals.
- For clocks based on a quartz crystals, the drift rate is approximately 10^{-6}.
  - It must be periodically resynchronized with the real-time clock.
- A clock is non-faulty if:
  - There is a bound on the amount of drift from real time for any given finite time interval.

Cont…

Example

- real time = *t*

- The time value of a clock *p* is = $C_p(t)$.
  - If all clocks in the world are perfectly synchronized, we would have $C_p(t) = t$ for all *p* and for all *t*.

- Ideal case:

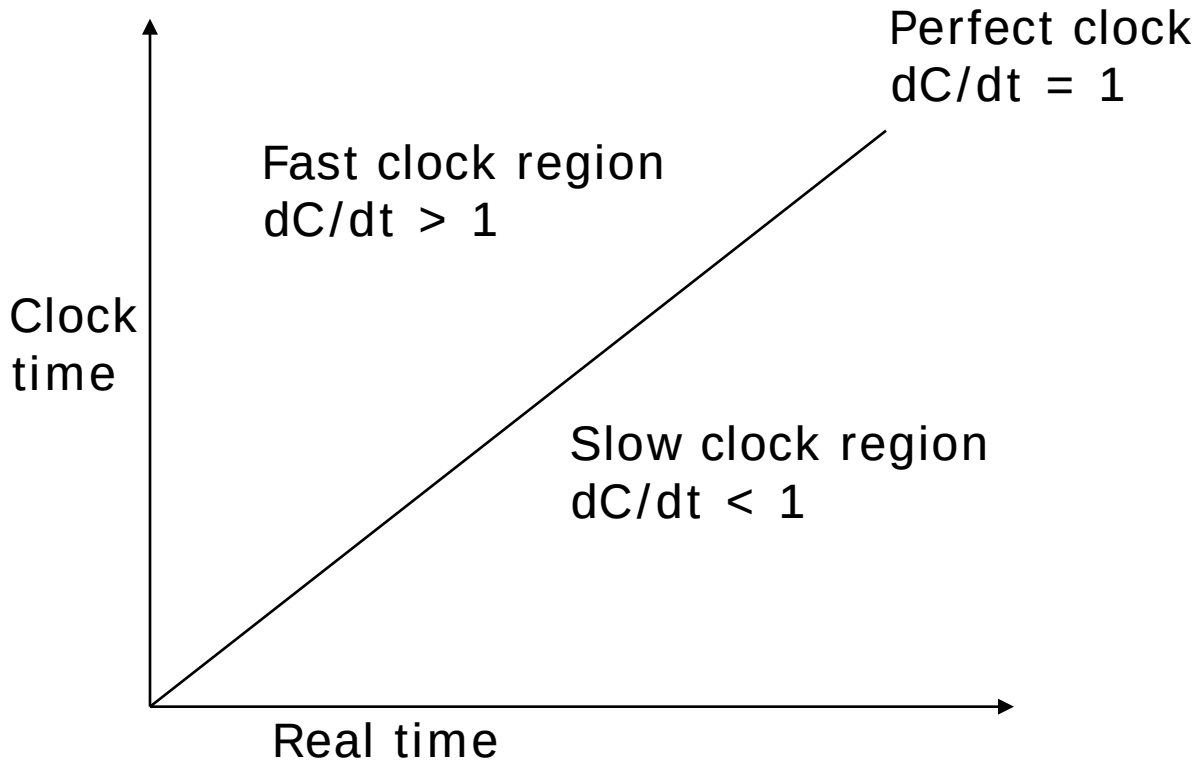
  $dC/dt = 1$

Cont…

- A clock is nonfaulty if
  - The maximum drift rate allowable is ρ.
  - Condition :

    $1 - \rho <= dC/dt <= 1 + \rho$

slow, perfect and fast clocks



Cont…

- Types of clock synchronization in DS:
  1. Synchronization of the computer clocks with real-time (or external) clocks.
     - Required for real-time applications.
     - Coordinated Universal Time (UTC), an external time source is used for synchronization.
     - Also synchronized internally.

Cont…

2. Mutual (or internal) synchronization of the clocks of different nodes of the system.

- Used when a consistent view of time across all nodes of a DS is required &
- A measurement of the duration of distributed activities are required.

Clock Synchronization Issues

- The difference between the two clock values is called *clock skew*.
- Required to be synchronized if the clock skew is more than some specified constant δ.

Cont…

- Issues:
  1. Calculating the value of the unpredictable communication delays between two nodes to deliver a clock signal is practically impossible.
     - Depends on the amount of communication and computations,
  2. Time must never run backward.
     - It may cause problems like repetition of certain operations.

Clock synchronization algorithms

1. Centralized algorithms
2. Distributed algorithms

Centralized algorithms

- Use of time server node for referencing the correct time.
- These algorithms keep the clocks of all other nodes synchronized with the clock time of the time server node.

Cont…

- Algorithms depending on the role of time server node:

1. Passive time server Centralized Algorithm.
2. Active time server Centralized Algorithm.

Passive time server Centralized Algorithm

- Each node periodically sends a message for the current time to the time server at any time t0.
- Time server responds with the current time T at the time t1.
- The propagation time of the message :

  $(t1 – t0) /2$
- When the reply is received at the client's node, its clock is readjusted to

  $T + (t1 – t0) /2$

Proposals to improve Centralized Algorithm

1. Assuming availability of some additional Info.

   - Assumes that the approximate time taken by the time server to handle the interrupt and process the request is known.

Cont…

2. Cristian method

- Several measurements are made & the measurements for which t1-t0 exceeds some threshold value are considered to be unreliable and discarded.
- If the value of t1-t0 is below threshold, it is averaged.
  - Half of this value is used as the value to be added to T.
- Need to restrict the number of measurements for estimating the value to be added to T.

Active time server centralized algorithm

- The time server periodically broadcasts its clock time.
- The receiving nodes use the clock value of the message for correcting their own clocks.
- Nodes have a priori knowledge of the approximate time required for the propagation of the message.

Cont…

- Drawback :
  1. It is not fault tolerant.
  2. Requires broadcast facility to be supported by the network.

- Solution : Berkeley algorithm

Cont…

Berkeley Algorithm

- Used for internal synchronization of clocks of a group of computers running the Berkeley Unix.
- The time server periodically sends a message to the group.
- Receivers send back their clock value to the time server.

Cont…

- Based on the priori knowledge of propagation time, the time server readjusts the clock value of the reply messages.
- It uses a fault-tolerant average.
- The time server sends the amount of time by which each node should readjust its clock.

Cont…

- Drawbacks of centralized clock synchronized algorithms:

1. Subject to a single-point failure.
2. Not acceptable from the scalability point of view.

Distributed algorithms

- Externally synchronized clocks are internally synchronized for better accuracy.
- Algorithms for internal synchronization:
  - Global averaging distributed algorithms.
  - Localized averaging distributed algorithms.

Global averaging distributed algorithms

- Clock process of each node broadcast its local time at the beginning of every fixed-length resynchronization interval.
  - Can't happen simultaneously from all nodes.
  - Hence each node waits for some time T.
- After this, the clock process collects resync messages broadcast by other nodes.
  - Records their time,
  - Estimates the skew of its clock, &
  - Computes the fault-tolerant average.

Cont…

- Algorithms for computing fault-tolerant average of the estimated skews:
  - Takes the average of the estimated skews and use it as the correction for the local clock.
    - Use threshold for comparison.
  - Each node limits the impact of faulty clocks by first discarding the m highest and m lowest estimated skews.
    - Calculate the average of the remaining skews for correcting the local clock.

Cont…

- Drawback
  - Not scalable: due to broadcast facility and a large amount of network traffic.

Localized averaging distributed algorithms

- The nodes are logically arranged in some kind of pattern (such as ring or a grid).
- Periodically, nodes exchange their clock time with their neighbours.
  - Sets their clock time to the average.

Event Ordering

- Lamport observed that for most applications, clock synchronization is not required.
- Total order of all events is sufficient that is consistent with observed behavior.
- For partial ordering of events, following can be used:
  - Relation of Happened-before, &
  - Logical clocks.

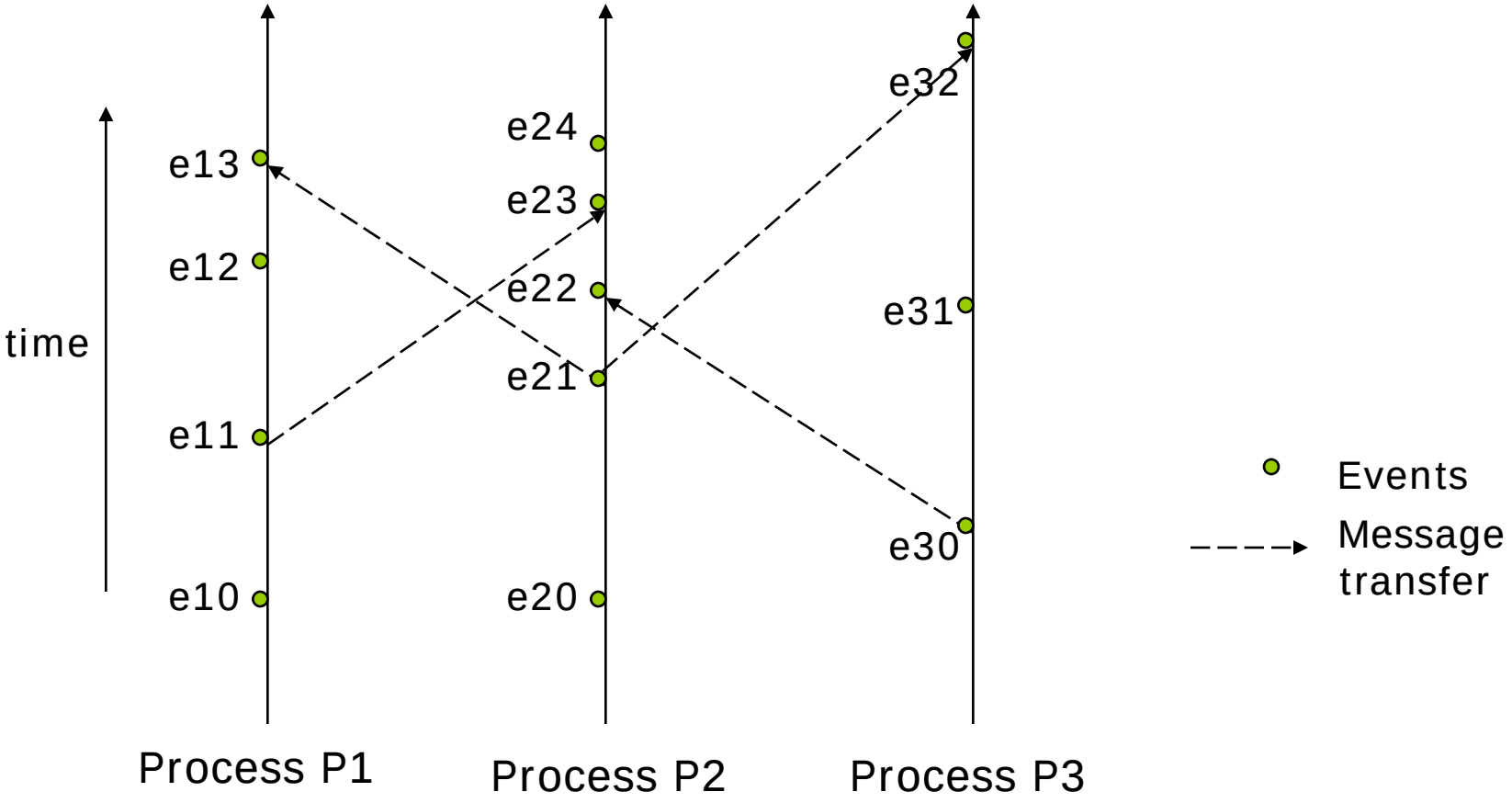
Happened-Before relation

- A relation (→) on a set of events satisfies the following conditions:
  1. If ‘a’ and ‘b’ are events in the same process and ‘a’ occurs before ‘b’, then a→ b.
  2. If ‘a’ is the event of sending a message by one process and ‘b’ is the event of the receipt of same message by another process, then a→b.
  3. If a→b and b→c, then a→c [a transitive relation].

Cont…

- Happened-before is an irreflexible partial ordering on the set of all events in the system.
  - a→a is not true since an event can't happen before itself.
- Concurrent events are not related by this relation.
  - a→b and b→a
    - That why, happened before is sometimes also known as the relation of causal ordering.

Space-time diagram for three processes



Cont…

- Some of the events that are related by the happened-before relation:

  e10→e11

  e20→e24

  e11→e23

  e21→e13

  e30→e24 (since e30→e22 and e22→e24)

  e11→e32 (since e11→e23, e23→e24 and)

Cont…

- Concurrent events:

  e12 and e20, e21 and e30,
  e10 and e30, e11 and e31,
  e12 and e32, e13 and e22

  This is because no path exists between these.

Logical clocks concept

- It is a way to associate a timestamp with each system event for correct working of happened-before relation.

- Each process P_i has a clock C_i associated with it that assigns a number $C_i(a)$ to any event ‘a’ in that process.
  - The clock of each process is known as logical clock.

Cont…

- The timestamps assigned to the events by the system of logical clocks must satisfy the clock condition:

  For any two events a and b, if a→b then C(a) < C(b).

  C is the clock function assigned to events.

Implementation of logical clocks

Clock conditions

- C1: If ‘a’ and ‘b’ are two events within the same process Pi and a→b, then Ci(a)<Ci(b)
- C2: if ‘a’ is the sending of a message ‘m’ by process P_i, and ‘b’ is the receipt of that message by process P_j, then $C_i(a)<C_j(b)$.
- C3: a clock C_i associated with a process P_i must always go forward, never backward.

Cont…

Implementation Rules:

- IR1: Each process P_i increments C_i between any two successive events.
  - Ensures condition C1.

Cont…

- IR2: If event ‘a’ is the sending of a message ‘m’ by process P_i , the message ‘m’ contains a timestamp $Tm = C_i(a)$, and upon receiving the message ‘m’, a process P_j sets C_j greater than or equal to its present value but greater than T_m.

  - Ensures the condition C2.
  - Both ensures the condition C3.

Logical Clock Implementation

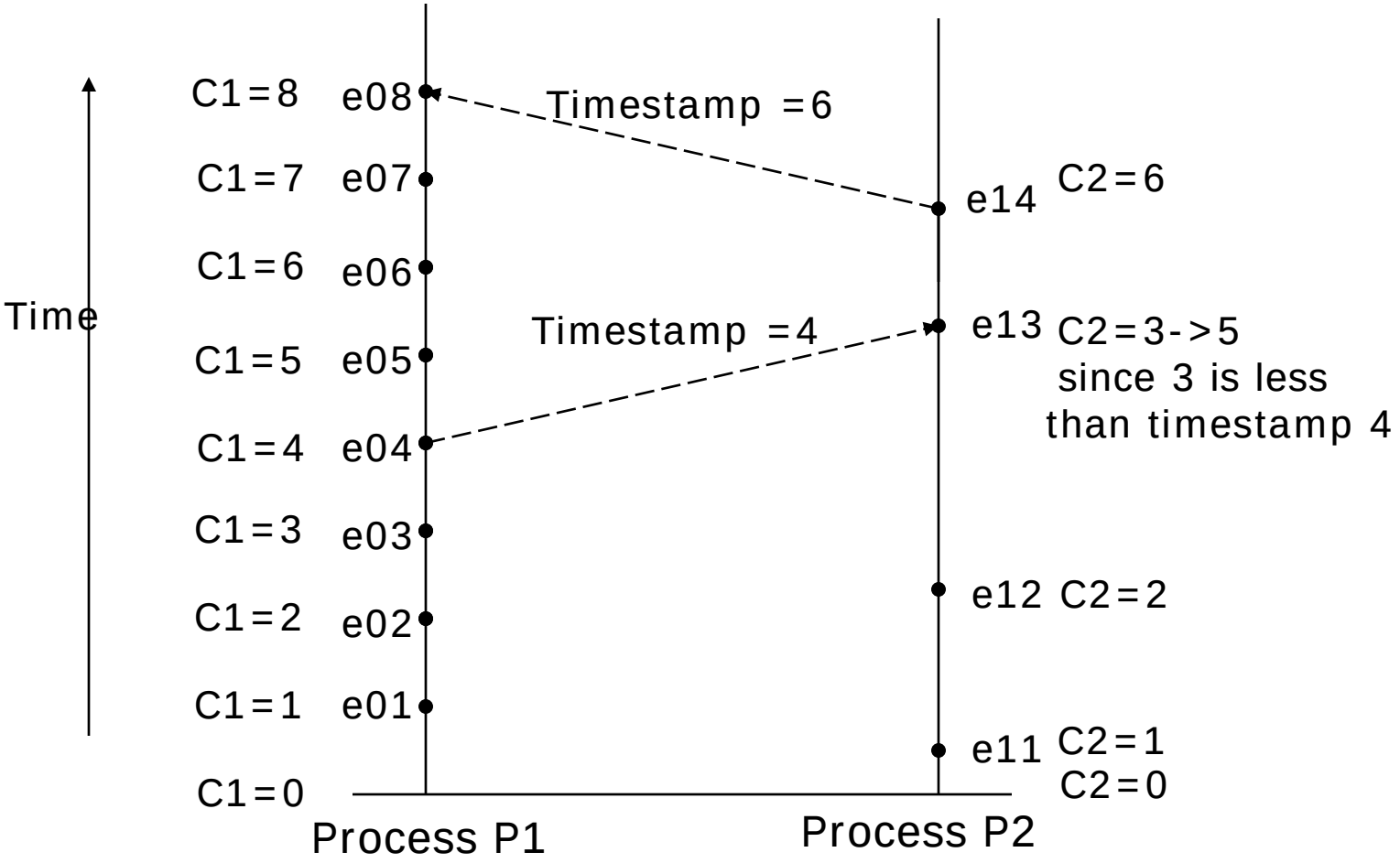
- Implementation of logical clocks by using
  1. Counters
  2. Physical clocks

Using counters

- Each process has a counter like C1 and C2 for process P1 and P2, respectively.

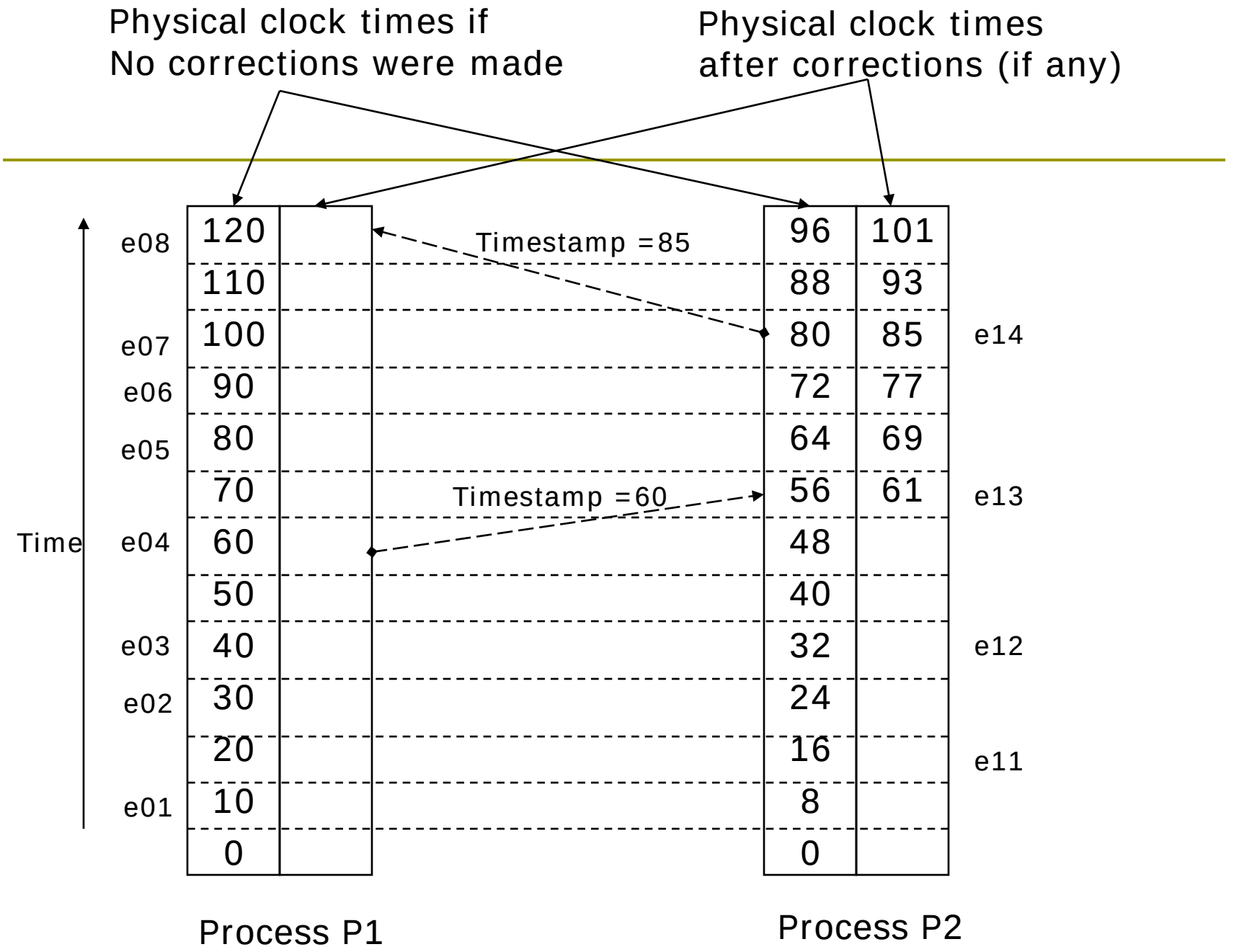
- Counters
  - Act as logical clocks.
  - Initialized to zero.
  - Increments by 1 on events of the process.

Cont…


Time
C1=8
e08
Timestamp =6
C1=7
e07
e14
C2=6
C1=6
e06
Timestamp =4
e13
C2=3->5
since 3 is less
than timestamp 4
C1=5
e05
C1=4
e04
C1=3
e03
e12
C2=2
C1=2
e02
C1=1
e01
e11
C2=1
C2=0
C1=0
Process P1
Process P2

Using physical clocks

- Each process has a physical clock associated with it.
- Each clock runs at a constant rate (it may be different for each of these process).
- Example:
  - When Process p1 has ticked 10 times, process p2 has ticked only 8 times.


Physical clock times if
No corrections were made
Physical clock times
after corrections (if any)
Timestamp =85
Timestamp =60
Time
e08
e07
e06
e05
e04
e03
e02
e01
e14
e13
e12
e11
120
110
100
90
80
70
60
50
40
30
20
10
0
96
101
88
93
80
85
72
77
64
69
56
61
48
40
32
24
16
8
0
Process P1
Process P2

Total ordering of events

- No events can occur at exactly the same time.
- Process identity numbers may be used to break ties and to create a total ordering of events.

Concurrent Access of Resources

- Requirements of an algorithm for implementing mutual exclusion:
  1. Mutual exclusion
  2. No starvation

Cont…

- Mutual exclusion:
  - Given a shared resource accessed by multiple concurrent processes, at any time only one process should access the resource.
  - Can be implemented in single-processor systems, using semaphores, monitors and similar constructs.

- No starvation:
  - If every process that is granted the resource eventually releases it, every request must be eventually granted.

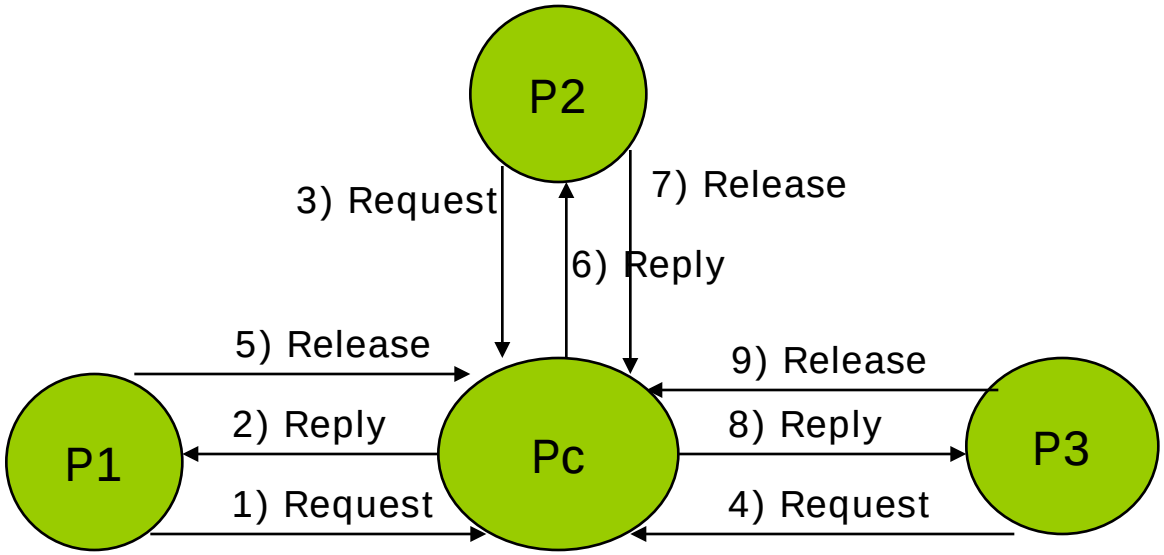
Cont…

- Approaches:
  1. Centralized approach
  2. Distributed approach
  3. Token passing approach

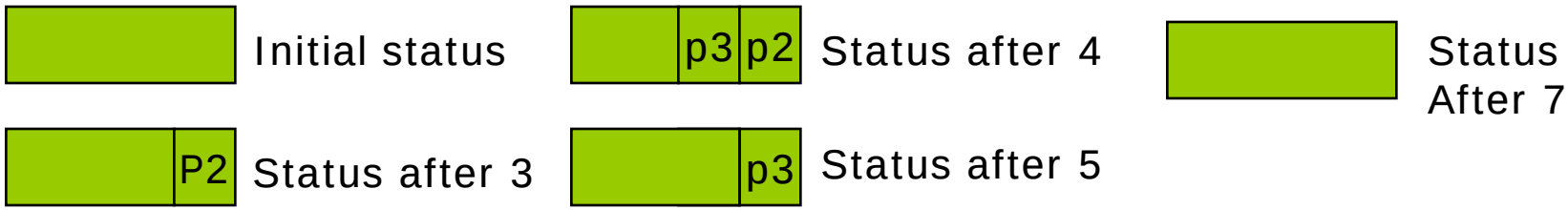
Centralized approach

- One of the processes in the system is elected as the coordinator.
- The coordinator coordinates the entry to the critical sections.
- Every process takes permission from the coordinator before entering critical sections.
- Request queue is used to grant requests.
  - First-come, first-serve

Cont…



Status of request queue:



Cont…

- Mutual exclusion by the coordinator:
  - Only one process is allowed to enter a critical section.
- No starvation:
  - Due to use of first-come, first-served scheduling policy.

Cont…

- Advantages of algorithm:
  - Simple to implement
  - Requires only three messages- a request, a reply and a release.
- Drawbacks:
  - Coordinator is the single point of failure.

Distributed approach

- The decision making for mutual exclusion is distributed across the entire system.
  - All processes that want to enter the same critical section cooperate with each other before reaching a decision on which process will enter the critical section next.

Ricart and Agrawala Algorithm

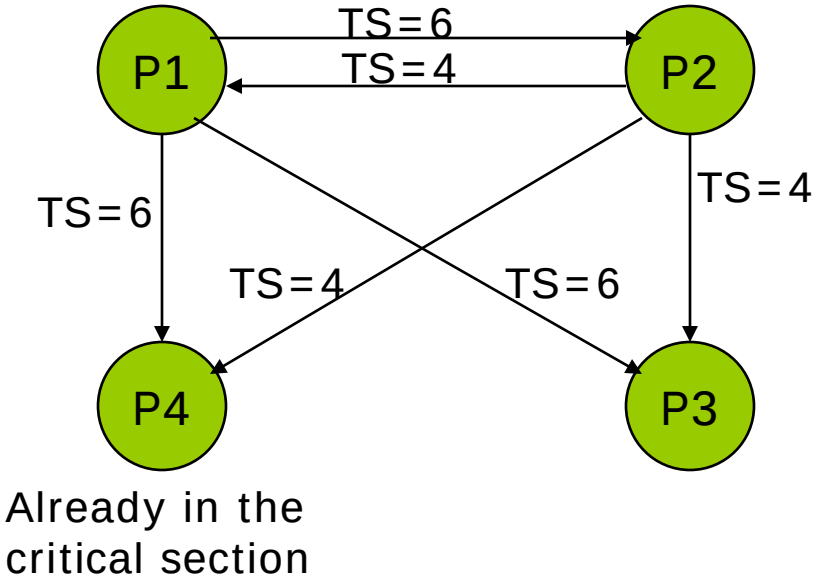
- Use of event-ordering scheme to generate a unique timestamp for each event in the system.
- When a process wants to enter a critical section, it sends a request message to all other processes.

Cont…

- Message:
  - Process id
  - Name of the critical section
  - A unique timestamp generated by the sender process.

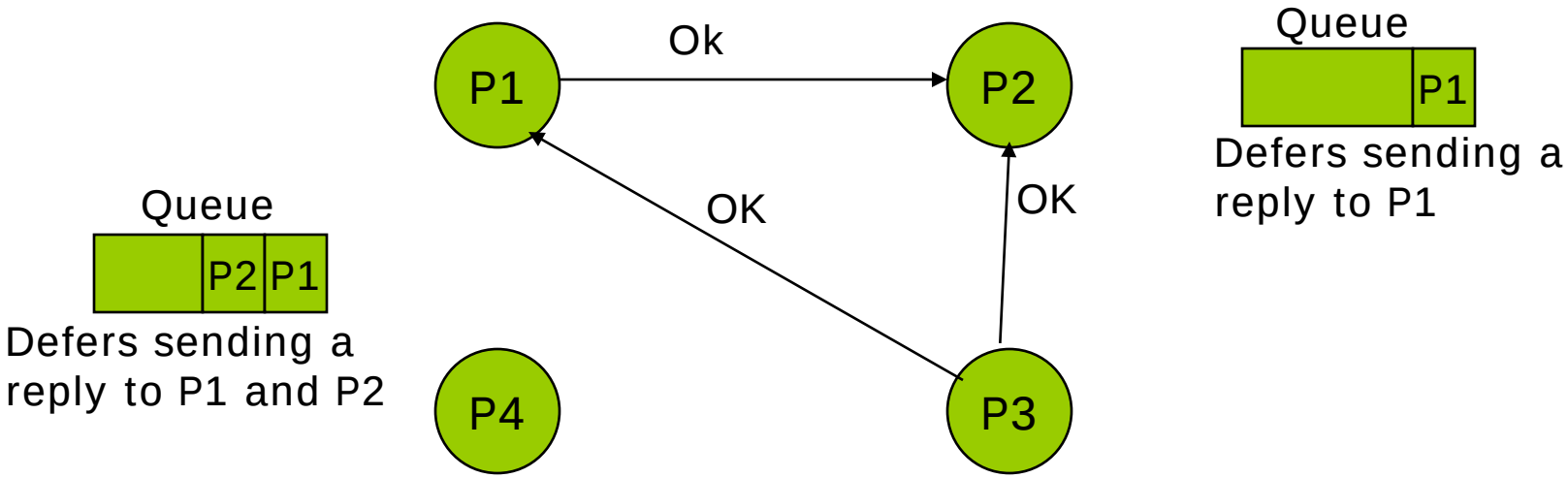
- A process either immediately sends back a reply message to the sender or defers sending a reply.

Example



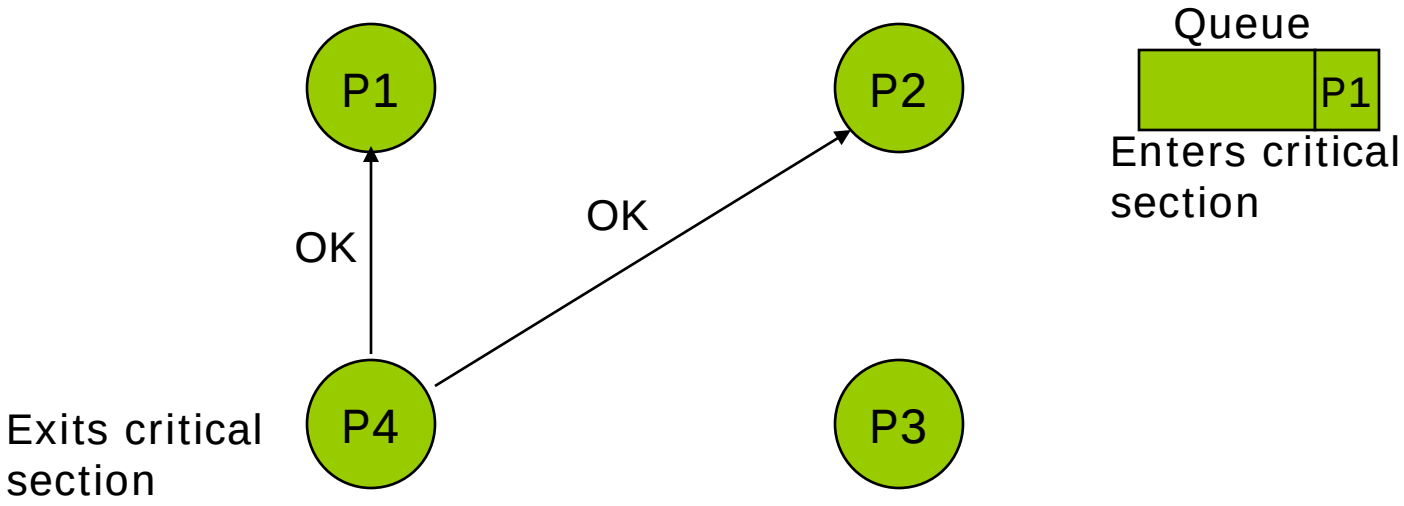
(a) Status when processes P1 and P2 send request messages to other processes while process p4 is already in the critical section.

Cont…



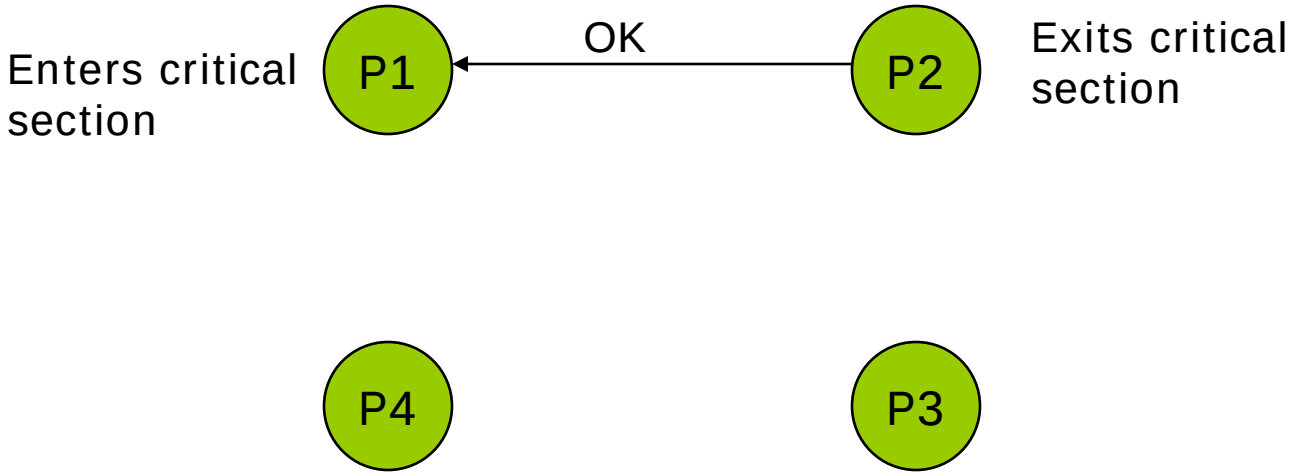
(b) Status when process P4 is still in the critical section.

Cont…



(c) Status after process P4 exits critical section

Cont…



(d) Status after process P2 exits critical section

Cont…

- Algorithm is free from deadlock.

- For n processes, the algorithm requires
  - n-1 request messages
  - n-1 reply messages,
  - Thus, 2(n-1) messages per critical section entry.

Cont…

- Drawbacks:
  1. In a system has n processes, the algorithm is liable to n points of failure.
  2. Requirement that each process knows the identity of all the processes participating in the mutual-exclusion algorithm.
  3. A process willing to enter a critical section can do so only after communicating with all other processes and getting permission from them.

Token passing approach

- Use of a single token to achieve mutual exclusion.
- Token is circulated among the processes in the system in a ring structure.
- Token: a special type of message that entitles its holder to enter a critical section.

Cont…

- If token holder wants to enter a critical section, it keeps token & enters CS.
- Otherwise, it just passes it along the ring to its neighbor process.

Cont…

- Mutual exclusion is guaranteed.
- The waiting time may be 0 to n-1.
  - Depends on the position of token.

Cont…

- Types of failures:
  1. Process failure
  2. Lost token

Process failure

- Process receiving token from neighbour always sends an ACK
- Each node maintains current ring configuration
- If process fails, dynamic reconfiguration of ring is carried out.
- When process turns alive, it simply informs to others.

Lost Token

- To regenerate lost token – one process in the ring acts as a Monitor process.
- Monitor periodically circulates “who has the token message”
- Process containing token inserts its id in the special field
- If no id found implies token lost.
- Problems:
  - Monitor process may itself fail
  - Who has the token message may be lost