**CS3.401** **Distributed Systems**

**Name- Vilal Ali**

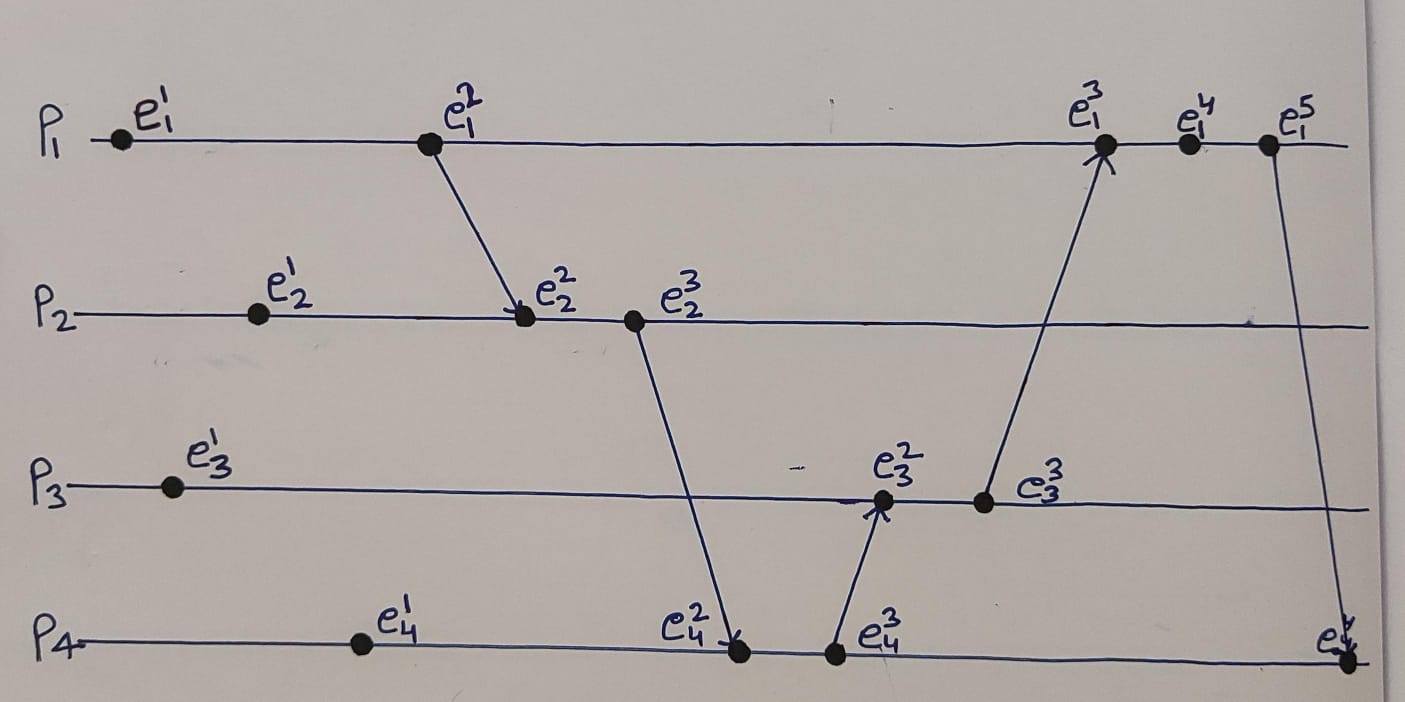
**Roll Number: 2024701027**

**2024701027\_Homework\_1**

**Question 1. Consider a system with four processors and a distributed program with about 15 events.**

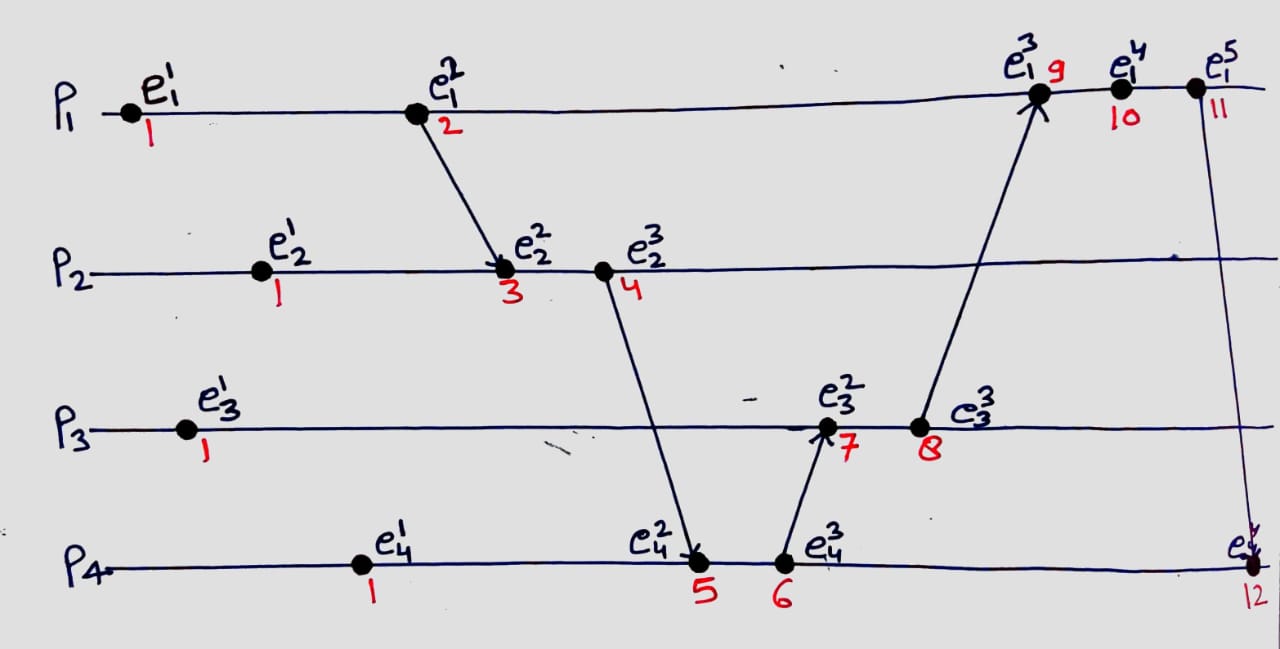
**Question 1 (a) - Draw the time diagram for such a program and mark the scalar time of the events. The set of events should include all possible events.**

**Answer:**

Let’s consider a distributed system of four processes (P1, P2, P3, P4) with 15 events. Scalar timestamps are assigned to each event using Lamport's logical clocks, where each event within a process receives a timestamp that increases in a sequential manner.  Figure 1.0 The space-time diagram of a distributed execution.

**Scalar Time:**

Below is the time space diagram created to represent this. Here we are set the increment d to 1 always.

**Figure 1.1:** The space-time diagram of a distributed execution.

**Question 1 (b) - Identify events that are logically concurrent:**

**Answer:**

* **ex i** : Represents an event x on process *Pi* .
* **ey j** : Represents an event y on process Pj.

Two events **ex i** and **ey j** are logically concurrent if there is no path (causal relationship) between them.

Logically Concurrent Events Are:

* e11 | | e12, e11 | | e13, e11 | | e14
* e12 | | e13, e12 | | e14
* e13 | | e14, e23 | | e14, e23 | | e42, e33 | | e42

**Question 1 (c) - Identify events where strong monotonicity of the scalar time fails.**

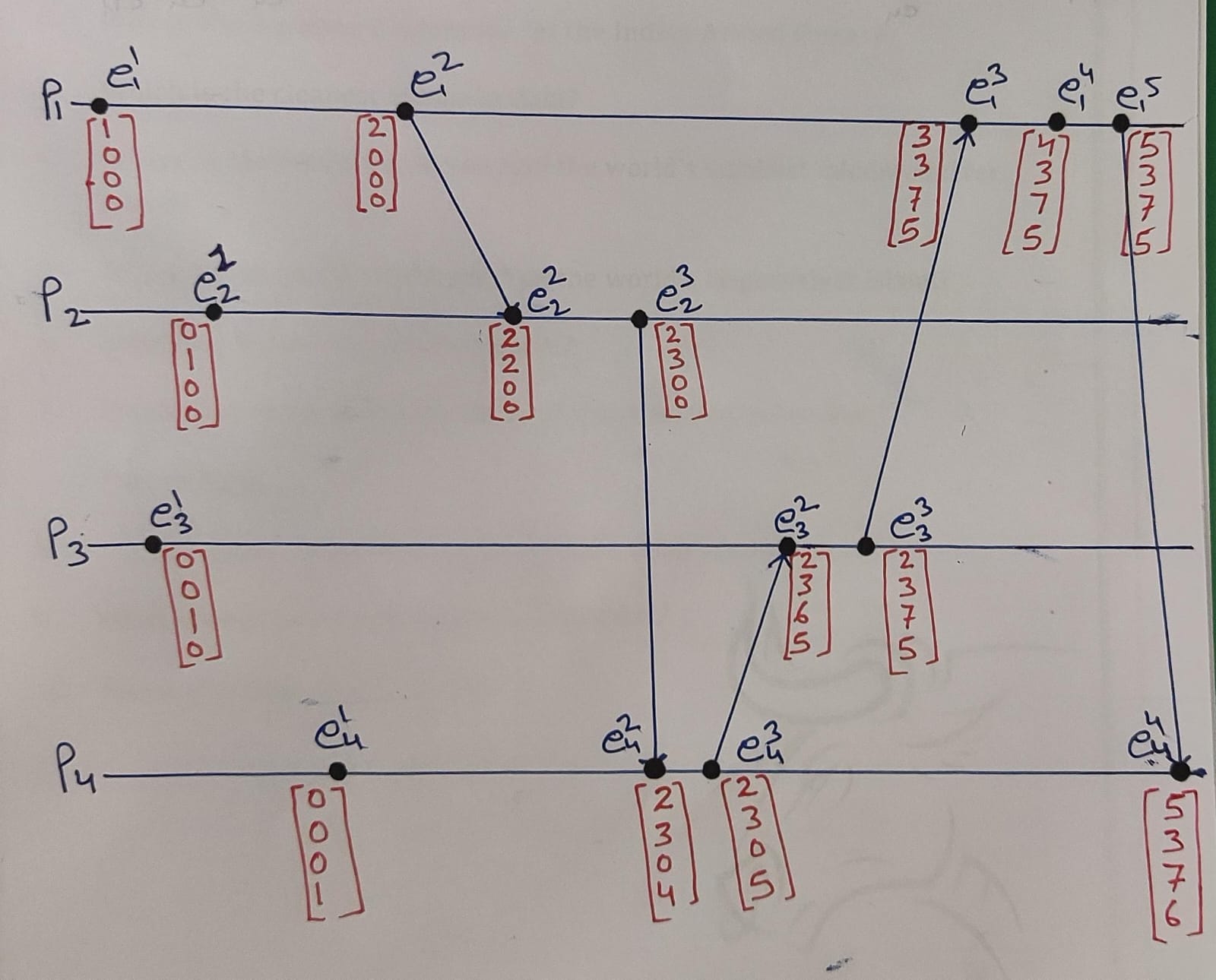
**Answer: Scalar time should preserve the "happens-before" relation (→), meaning that if ei → ej then C(ei) < C(ej).**

In the diagram, strong monotonicity of scalar time could fail if there is any sequence where an event **ex i**, occurs after another event ejy, but C(**exi**) is less than C(**eyj**). Without actual clock values provided, we can analyze the sequence of events visually:

From Figure (1.1), all the direct causality links (arrows) maintain the order of events without causing any strong monotonicity failures.

**Question 2 (a): Draw the time diagram for such a program and mark the** vector **time of the events. The set of events should include all possible events.**

**Answer:** Vector time ensures that the "happens-before" relation is preserved across all processes with vector clocks:

**Figure 2.0:** The vector space-time diagram of a distributed execution.

* **Initially, the vector clock is set to [0, 0, 0, ...., 0] and d = 1.**
* **Assign timestamps to all the events shown in the diagram:**

For each event, we have the following vector timestamps:

|  |  |  |  |
| --- | --- | --- | --- |
| **P1** | **P2** | **P3** | **P4** |
| e11 P1: [1,0,0,0] | e12 P2: [0,1,0,0] | e13 P3: [0,0,1,0] | e14 P4: [0,0,0,1] |
| e21 P1: [2,0,0,0] | e22 P2: [2,2,0,0] | e23 P3: [2,3,6,5] | e24 P4: [2,3,0,4] |
| e31 P1: [3,3,7,5] | e32 P2: [2,3,6,5] | e33 P3: [3,3,7,5] | e34 P4: [2,3,0,5] |
| e41 P1: [4,3,7,5] |  |  | e44 P4: [5,3,7,6] |
| e51 P1: [5,3,7,5] |  |  |  |

**Table: 1.0: Given Events and Vector Timestamps**

**Question 2 (b): Identify events that are logically concurrent with respect to vector time.**

Answer:

To identify logically concurrent events in the second image, we need to determine pairs of events that have no causal relationship. This means there is no direct or indirect "happens-before" relationship between them.

In the context of vector clocks, two events are considered logically concurrent if their vector timestamps are not comparable, meaning that neither event's timestamp is a component-wise greater than or equal to the other event's timestamp.

**Identifying Logically Concurrent Events:**

**Process P1:**

* e11 is logically concurrent with:
  + e12 [0, 1, 0, 0] (P2)
  + e13 [0, 0, 1, 0] (P3)
  + e14 [0, 0, 0, 1] (P4)
* e21 is logically concurrent with:
  + e12 [0, 1, 0, 0] (P2)
  + e13 [0, 0, 1, 0] (P3)
  + e14 [0, 0, 0, 1] (P4)
* e31 is logically concurrent with:
  + e22 [2, 2, 0, 0] (P2)
  + e24 [2, 3, 0, 4] (P4)

**Process P2:**

* e12 is logically concurrent with:
  + e11 [1, 0, 0, 0] (P1)
  + e13 [0, 0, 1, 0] (P3)
  + e14 [0, 0, 0, 1] (P4)
* e22 is logically concurrent with:
  + e31 [3, 0, 0, 0] (P1)
  + e23 [2, 3, 5, 0] (P3)
  + e24 [2, 3, 0, 4] (P4)

**Process P3:**

* e13 is logically concurrent with:
  + e11 [1, 0, 0, 0] (P1)
  + e12 [0, 1, 0, 0] (P2)
  + e14 [0, 0, 0, 1] (P4)
* e23 is logically concurrent with:
  + e24 [2, 3, 0, 4] (P4)
* e33 is logically concurrent with:
  + e41 [3, 3, 7, 5] (P1)
  + e34 [2, 3, 0, 5] (P4)

**Process P4:**

* e14 is logically concurrent with:
  + e11 [1, 0, 0, 0] (P1)
  + e12 [0, 1, 0, 0] (P2)
  + e13 [0, 0, 1, 0] (P3)
* e24 is logically concurrent with:
  + e31 [3, 0, 0, 0] (P1)
  + e22 [2, 2, 0, 0] (P2)
  + e23 [2, 3, 5, 0] (P3)

Summary of Logically Concurrent Events:

* e11 with e12, e13, e14
* e21 with e12, e13, e14
* e31 with e22, e24
* e12 with e11, e13, e14
* e22 with e31, e23, e24
* e13 with e11, e12, e14
* e23 with e24
* e33 with e41, e34
* e14 with e11, e12, e13
* e24 with e31, e22, e23

These events are logically concurrent because they have no causal dependency on each other, as indicated by their vector timestamps.

**Question 3: Show that the timestamps assigned by vector time are strongly consistent.**

Answer: To demonstrate that the timestamps assigned by vector time are strongly consistent, we need to show that for any two events ei and ej in a distributed system:

If ei→ej (i.e., event ei happens-before event ej ), then the vector timestamp of ei is less than or equal to the vector timestamp of ej in every component.

**Note:** This is the condition of strong consistency in the context of vector clocks.

**Analysis of Event Causality: Using the Table 1.0**

1. **Causality within a Process:**

* **Within any single process Pi, events are ordered in time by their index. For example:**
* **e11→e21→e31→e41→e51**
* **This implies that the vector timestamps must be monotonically non-decreasing:**
  + **[1, 0, 0, 0] ≤ [2, 0, 0, 0] ≤ [3, 0, 0, 0] ≤ [3, 3, 7, 5] ≤ [5, 3, 7, 5]**
* **We see that this holds true for each process.**

1. **Causality across Processes:**

* **Consider the causal relationships between events across different processes, as indicated by the arrows in the diagram. For example:**
  + **e21→e22**
    - **Vector timestamps: [2, 0, 0, 0] ≤ [2, 2, 0, 0]**
  + **e22→e23**
    - **Vector timestamps: [2, 2, 0, 0] ≤ [2, 3, 5, 0]**
  + **e32→e33**
    - **Vector timestamps: [2, 3, 6, 5] ≤ [2, 3, 7, 5]**
  + **e24→e34**
    - **Vector timestamps: [2, 3, 0, 4] ≤ [2, 3, 0, 5]**

**These relations hold true according to the vector timestamps. This guarantees that vector time is strongly consistent, meaning it correctly captures the causal relationships between events in a distributed system.**