

ADVANCE OPERATING SYSTEM LAB (CSD-416) ASSIGNMENT 3

Simulate the functioning of Lamport's Logical Clock and Vector Clock

AKSHAT RAJ VANSI (185520)

DECEMBER 7, 2021



Contents

1	Lamport's Logical Clock	2
1.1	Code	2
1.2	Output	3
2	Vector Clock	4
2.1	Code	4
2.2	Output	5

1 Lamport's Logical Clock

1.1 Code

```

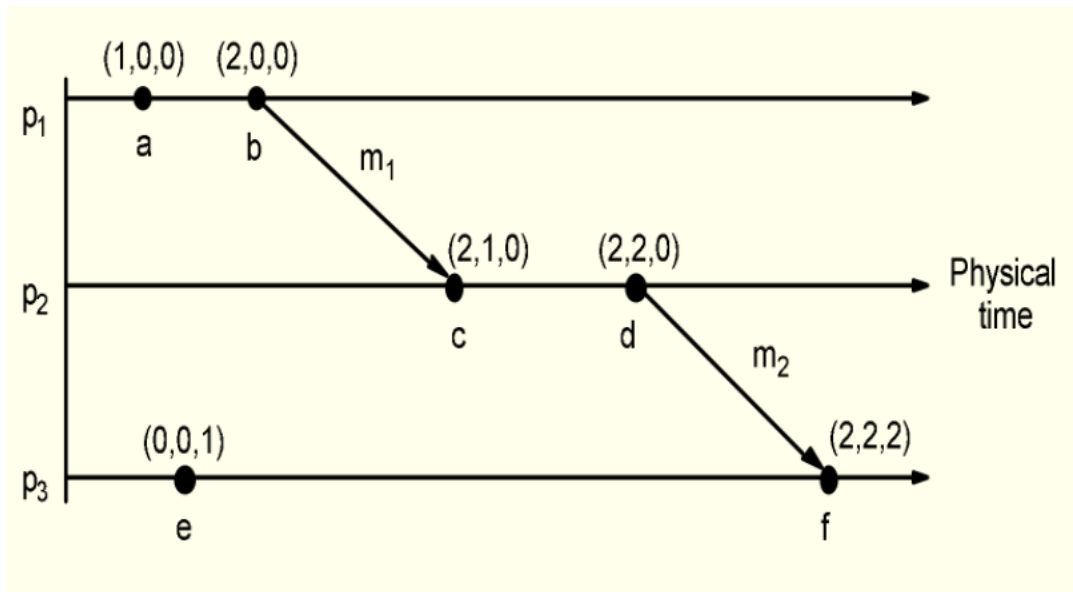
1  def logical_number(spno, seno, rpno, reno):
2      if(seno != 1):
3          P[spno][seno] = P[spno][seno-1]+1
4      if(reno != 1):
5          P[rpno][reno] = max(P[rpno][reno-1], P[spno][seno])+1
6      else:
7          P[rpno][reno] = P[spno][seno]+1
8
9
10 P = {1: {}, 2: {}, 3: {}}
11
12 inc = 0
13
14 n1 = int(input("Enter the no. of events in Process 1 : "))
15 e1 = [i for i in range(1, n1 + 1)]
16 P[1] = {key: inc + key for key in e1}
17 print(P[1])
18 print("\n")
19
20 n2 = int(input("Enter the no. of events in Process 2 : "))
21 e2 = [i for i in range(1, n2 + 1)]
22 P[2] = {key: inc + key for key in e2}
23 print(P[2])
24 print("\n")
25
26 n3 = int(input("Enter the no. of events in Process 3 : "))
27 e3 = [i for i in range(1, n3 + 1)]
28 P[3] = {key: inc + key for key in e3}
29 print(P[3])
30 print("\n")
31
32 comm = int(input("Enter the no of communication lines : "))
33 print("\n")
34
35 while inc < comm:
36     sent = int(input("Enter the sending process number : "))
37     recv = int(input("Enter the receiving process number : "))
38     sent_event_no = int(input("Enter the sending event number : "))
39     recv_event_no = int(input("Enter the receiving event number : "))
40     if sent <= 3 and recv <= 3:
41         print("P{} --> P{}".format(sent, recv))
42         logical_number(sent, sent_event_no, recv, recv_event_no)
43         print("New vector value of \"event {}\" in process P{} is : {} \n".format(
44             recv_event_no, recv, P[recv][recv_event_no]))
45     else:
46         print("Enter the sent/recv within existing process")
47     inc += 1
48
49 print("Final vectors of the 3 process are")
50 print(P[1])
51 print(P[2])
52 print(P[3])

```

1.2 Output

Lamport's Logical Clock Outputs

Example Taken



Output

```

Enter the no. of events in Process 1 : 2
{1: 1, 2: 2}

Enter the no. of events in Process 2 : 2
{1: 1, 2: 2}

Enter the no. of events in Process 3 : 2
{1: 1, 2: 2}

Enter the no of communication lines : 2

Enter the sending process number : 2
Enter the receiving process number : 3
Enter the sending event number : 2
Enter the receiving event number : 2
P2 --> P3
New vector value of "event 2" in process P3 is : 5

Final vectors of the 3 process are
{1: 1, 2: 2}
{1: 3, 2: 4}
{1: 1, 2: 5}
PS D:\Coding\Advance Operating System Lab>

```

2 Vector Clock

2.1 Code

```

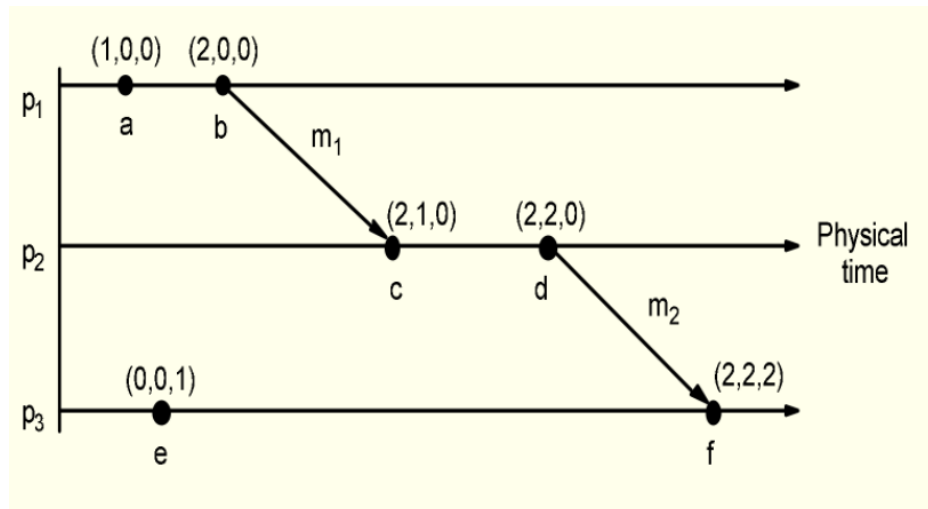
1  def vector_compare(vector1, vector2):
2      vector = [max(value) for value in zip(vector1, vector2)]
3      return vector
4
5  P = {1: {}, 2: {}, 3: {}}
6  inc = 0
7
8  n1 = int(input("Enter the no. of events in Process 1 : "))
9  e1 = [i for i in range(1, n1 + 1)]
10 P[1] = {key: [inc + key, 0, 0] for key in e1}
11 print(P[1])
12 print("\n")
13
14 n2 = int(input("Enter the no. of events in Process 2 : "))
15 e2 = [i for i in range(1, n2 + 1)]
16 P[2] = {key: [0, inc + key, 0] for key in e2}
17 print(P[2])
18 print("\n")
19
20 n3 = int(input("Enter the no. of events in Process 3 : "))
21 e3 = [i for i in range(1, n3 + 1)]
22 P[3] = {key: [0, 0, inc + key] for key in e3}
23 print(P[3])
24 print("\n")
25
26 comm = int(input("Enter the no of communication lines : "))
27 print("\n")
28
29 while inc < comm:
30     sent = int(input("Enter the sending process number : "))
31     recv = int(input("Enter the receiving process number : "))
32     sent_event_no = int(input("Enter the sending event number : "))
33     recv_event_no = int(input("Enter the receiving event number : "))
34     if sent <= 3 and recv <= 3:
35         print("P{} --> P{}".format(sent, recv))
36         new_vector = vector_compare(
37             P[sent][sent_event_no], P[recv][recv_event_no])
38         P[recv][recv_event_no] = new_vector
39         print("New vector value of \"event {}\" in process P{} is : {} \n".format(
40             recv_event_no, recv, P[recv][recv_event_no]))
41
42         if (recv_event_no + 1) in P[recv]:
43             for i in range(recv_event_no + 1, len(P[recv]) + 1):
44                 P[recv][i] = vector_compare(P[recv][i-1], P[recv][i])
45         else:
46             print("Enter the sent/recv within existing process")
47         inc += 1
48
49 print("Final vectors of the 3 process are")
50 print(P[1])
51 print(P[2])
52 print(P[3])

```

2.2 Output

Vector Clock Outputs

Example Taken



Output

```

arveus@arveus-omen:~/Advance Operating System Lab/3-Clocks/Code$ python3 vector-clock.py
Enter the no. of events in Process 1 : 2
{1: [1, 0, 0], 2: [2, 0, 0]}

Enter the no. of events in Process 2 : 2
{1: [0, 1, 0], 2: [0, 2, 0]}

Enter the no. of events in Process 3 : 2
{1: [0, 0, 1], 2: [0, 0, 2]}

Enter the no of communication lines : 2

Enter the sending process number : 1
Enter the receiving process number : 2
Enter the sending event number : 2
Enter the receiving event number : 1
P1 --> P2
New vector value of "event 1" in process P2 is : [2, 1, 0]

Enter the sending process number : 2
Enter the receiving process number : 3
Enter the sending event number : 2
Enter the receiving event number : 2
P2 --> P3
New vector value of "event 2" in process P3 is : [2, 2, 2]

Final vectors of the 3 process are
{1: [1, 0, 0], 2: [2, 0, 0]}
{1: [2, 1, 0], 2: [2, 2, 0]}
{1: [0, 0, 1], 2: [2, 2, 2]}
arveus@arveus-omen:~/Advance Operating System Lab/3-Clocks/Code$ 

```