Distributed Computing (CS - 610)
National Institute of Technology, Hamirpur

DISTRIBUTED COMPUTING (CS-610) ASSIGNMENT

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

AKSHAT RAJ VANSH (185520)

April 11, 2022

Computer Science Department
National Institute of Technology, Hamirpur

Contents

1 Lamport's Logical Clock

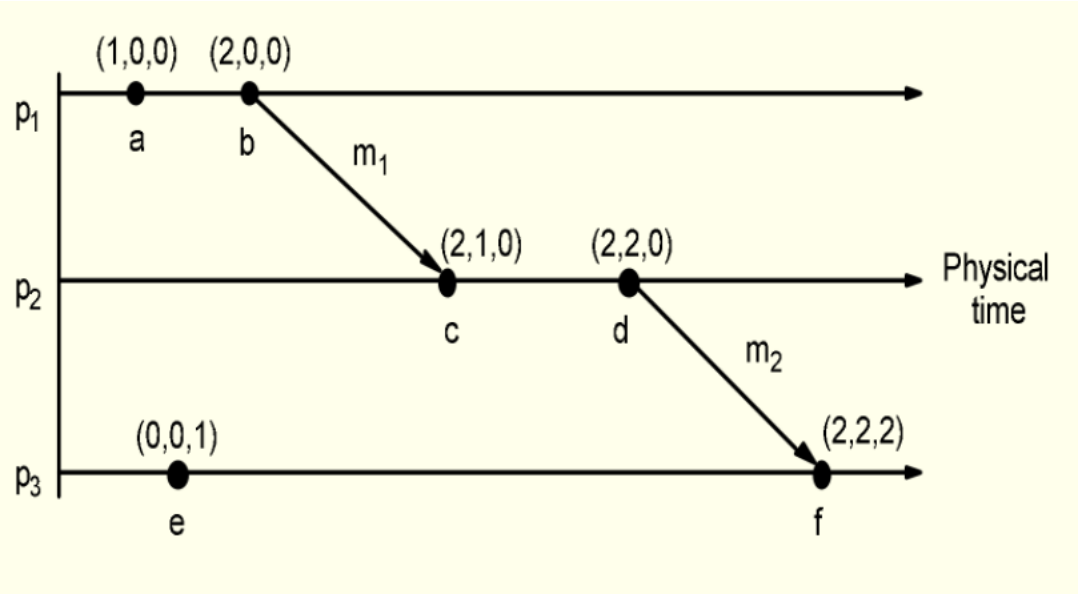
1.1 Code

```
def logical_number(spno, seno, rpno, reno):
    if(seno != 1):
        P[spno][seno] = P[spno][seno-1]+1
    if(reno != 1):
        P[rpno][reno] = max(P[rpno][reno-1], P[spno][seno])+1
    else:
        P[rpno][reno] = P[spno][seno]+1


P = {1: {}, 2: {}, 3: {}}

inc = 0

n1 = int(input("Enter the no. of events in Process 1 : "))
e1 = [i for i in range(1, n1 + 1)]
P[1] = {key: inc + key for key in e1}
print(P[1])
print("\n")

n2 = int(input("Enter the no. of events in Process 2 : "))
e2 = [i for i in range(1, n2 + 1)]
P[2] = {key: inc + key for key in e2}
print(P[2])
print("\n")

n3 = int(input("Enter the no. of events in Process 3 : "))
e3 = [i for i in range(1, n3 + 1)]
P[3] = {key: inc + key for key in e3}
print(P[3])
print("\n")

comm = int(input("Enter the no of communication lines : "))
print("\n")

while inc < comm:
    sent = int(input("Enter the sending process number : "))
    recv = int(input("Enter the receiving process number : "))
    sent_event_no = int(input("Enter the sending event number : "))
    recv_event_no = int(input("Enter the receiving event number : "))
    if sent <= 3 and recv <= 3:
        print("P{} --> P{}".format(sent, recv))
        logical_number(sent, sent_event_no, recv, recv_event_no)
        print("New vector value of \"event {}\"  in process P{} is : {} \n".format(
            recv_event_no, recv, P[recv][recv_event_no]))
    else:
        print("Enter the sent/recv within existing process")
    inc += 1

print("Final vectors of the 3 process are")
print(P[1])
print(P[2])
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

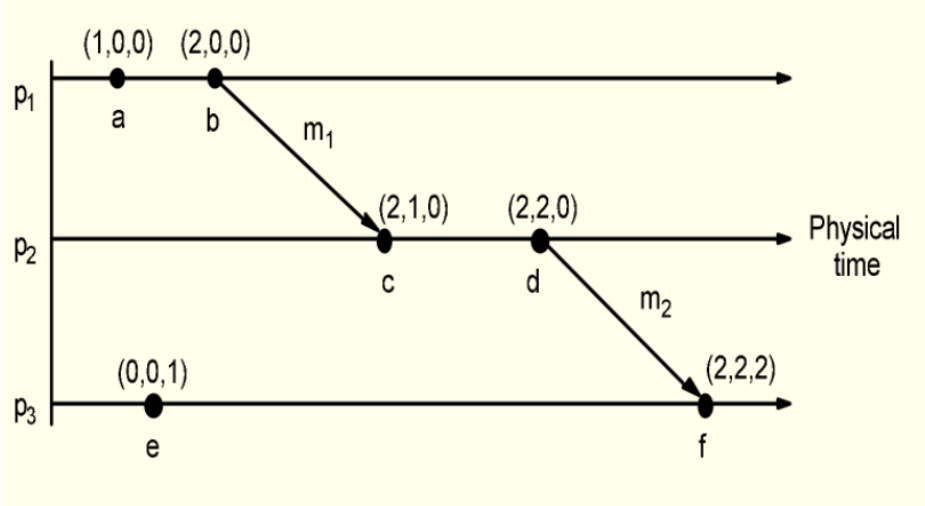
2.1 Code

```
def vector_compare(vector1, vector2):
    vector = [max(value) for value in zip(vector1, vector2)]
    return vector

P = {1: {}, 2: {}, 3: {}}
inc = 0

n1 = int(input("Enter the no. of events in Process 1 : "))
e1 = [i for i in range(1, n1 + 1)]
P[1] = {key: [inc + key, 0, 0] for key in e1}
print(P[1])
print("\n")

n2 = int(input("Enter the no. of events in Process 2 : "))
e2 = [i for i in range(1, n2 + 1)]
P[2] = {key: [0, inc + key, 0] for key in e2}
print(P[2])
print("\n")

n3 = int(input("Enter the no. of events in Process 3 : "))
e3 = [i for i in range(1, n3 + 1)]
P[3] = {key: [0, 0, inc + key] for key in e3}
print(P[3])
print("\n")

comm = int(input("Enter the no of communication lines : "))
print("\n")

while inc < comm:
    sent = int(input("Enter the sending process number : "))
    recv = int(input("Enter the receiving process number : "))
    sent_event_no = int(input("Enter the sending event number : "))
    recv_event_no = int(input("Enter the receiving event number : "))
    if sent <= 3 and recv <= 3:
        print("P{} --> P{}".format(sent, recv))
        new_vector = vector_compare(
            P[sent][sent_event_no], P[recv][recv_event_no])
        P[recv][recv_event_no] = new_vector
        print("New vector value of \"event {}\"  in process P{} is : {} \n".format(
            recv_event_no, recv, P[recv][recv_event_no]))

        if (recv_event_no + 1) in P[recv]:
            for i in range(recv_event_no + 1, len(P[recv]) + 1):
                P[recv][i] = vector_compare(P[recv][i-1], P[recv][i])
    else:
        print("Enter the sent/recv within existing process")
    inc += 1

print("Final vectors of the 3 process are")
print(P[1])
print(P[2])
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$
```