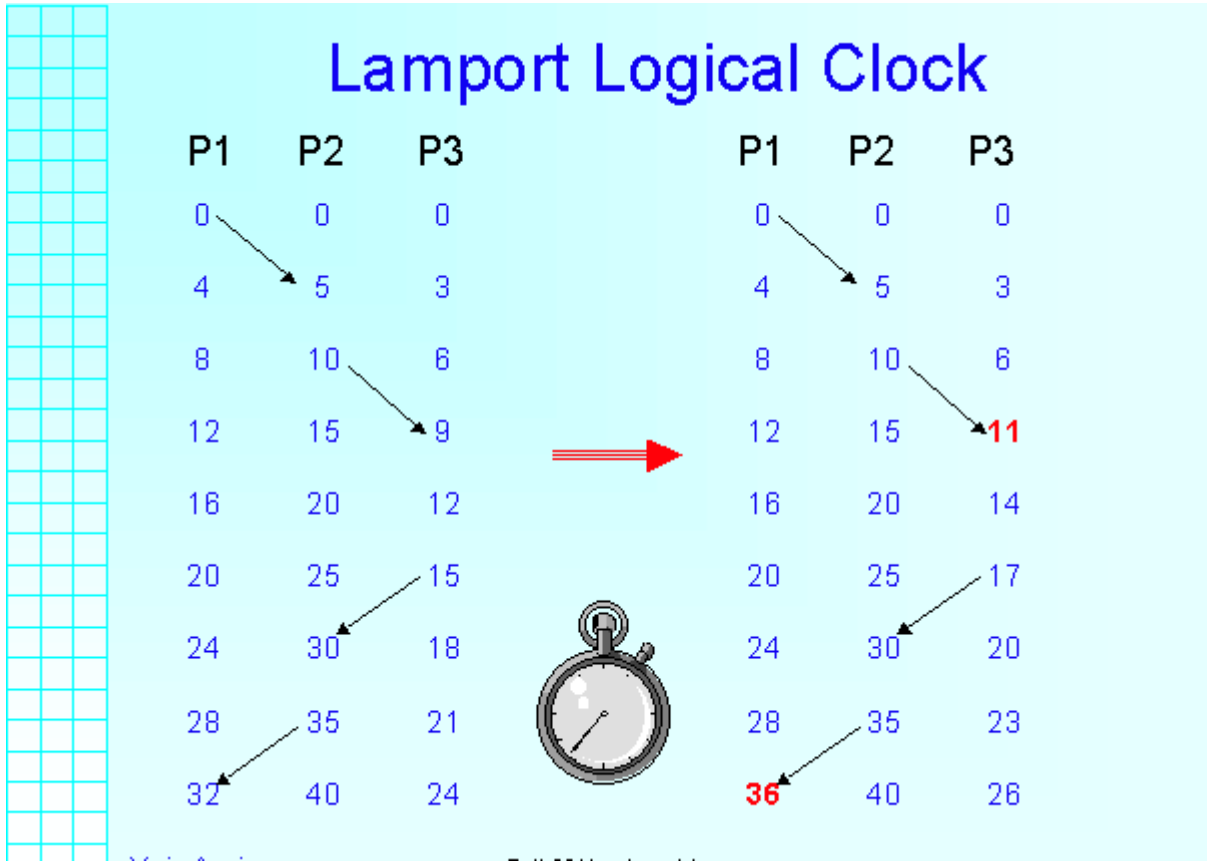
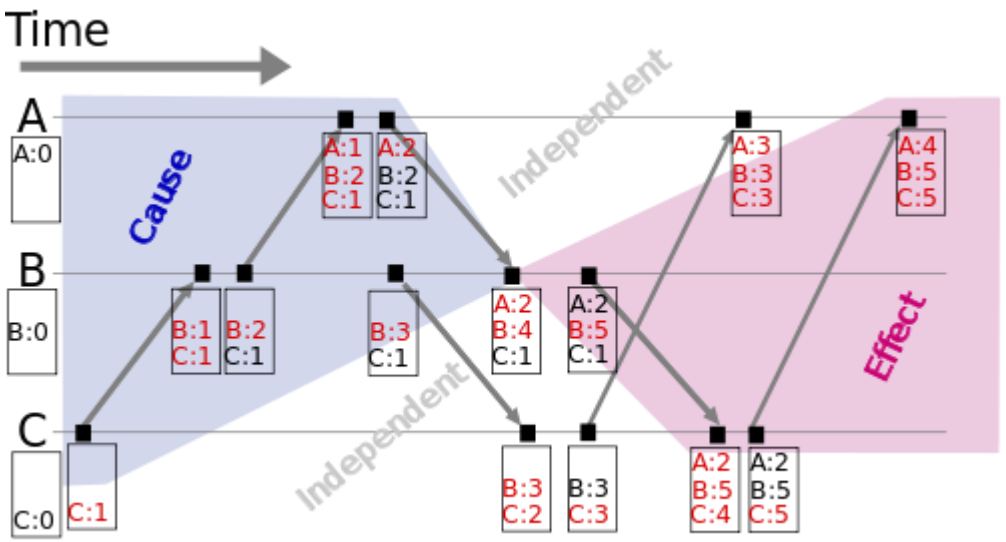
Use RMI to implement Lamport's vector clocks Using Java RMI

Problem Statement:-

To Implement Lamports Vector clocks using RMI (Remote method invocation).





- *It's a counter within a single process that increments from*

1. Events within a process is assigned an unique id based on the counter value. Thus all compute and messaging events within a process are assigned an unique value

2. When sending a message from process-a, we set C(a) = C(a) + 1, then pass on C(a) as part of the message.

3. On recipient of message in process-b we set C(b) = max(C(b) + 1, C(a))

Lamport Vector clock algorithm is a method to maintain the order of occurring events using their chronological order and casual relationship. The algorithm uses messages sent between the computers to ensure that the logical vector clock values remain synchronized and does partial ordering on events (Not every pair of events needed to be compared).

- Happened-before relation: a -> b

This relation is the central concept with logical vector clocks which means event ‘a’ happened before event ‘b’.

-

Logical Clock: (Ci/Cj is logical vector clock time at any timestamp i/j)

[C1]: Ci(a) < Ci(b) - If ‘a’ happened before ‘b’, then time of ‘a’ will be less than time of ’b’.

[C2]: Ci(a) < Cj(b) – Clock value of Ci(a) is less than Cj(b).

Algorithm:

• Each computer in the distributed system maintains a logical vector clock value, which is initially set to zero.

• Whenever an event occurs on a computer, the logical vector clock value is incremented by one.

• When a message is sent from one computer to another, the sender includes its current logical vector clock value in the message.

• When a computer receives a message, it compares the logical vector clock value in the message to its own logical vector clock value, received logical vector clock values and increment it by 1.

• This process is repeated for each event and message in the system, ensuring that the events maintain the order of occurrence

Concurrent Events:

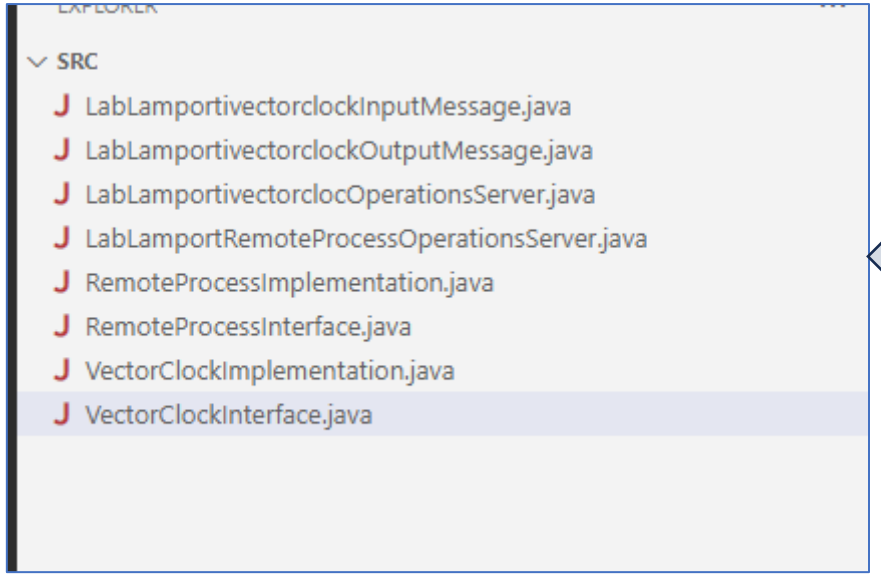
• The algorithm does a partial ordering, means it is not necessary to compare every event, so if two events can’t be compared or do not have casual path, they are called concurrent.

• According to happened before:

If event E1 , E2, then logical timestamp of E1 < E2, however we cannot exactly tell that E1 happened before E2 or E1 and E2 are concurrent.

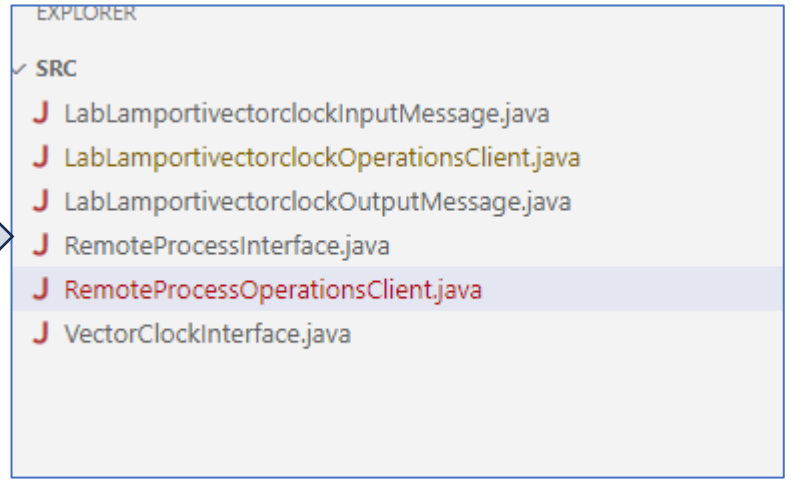
• To detect the concurrent events, concept of Vector clocks was introduced.

Server



Logics and algorithm mentioned in the code as comments

Client



Instructions:-

1. Open Visual studio Code
2. Point to the Server location
3. Open LabLamportRemoteProcessOperationServer.java
4. Run command from Visual studio code

Instructions:-

1. Open Visual studio Code
2. Point to the Client location location
3. Open RemoteProcessOperationClient.java
4. Run command from Visual studio code

```
Process: RP1    Local logical clock: (0, 0 , 0) EventOrdering.NOT_COMPARABLE
Initial connection Internal Event:Create registry for LabLamportRemoteProcess for process id RP1        Local logical clock: (0,1,0)
Initial connection Internal Event:Rebind object for process id RP1      Local logical clock: (0,2,0)
Process: RP1    Local logical clock: (0, 2 , 0) EventOrdering.NOT_COMPARABLE
LabLamport RemoteProcess Server has been started.
```

Communication starts.

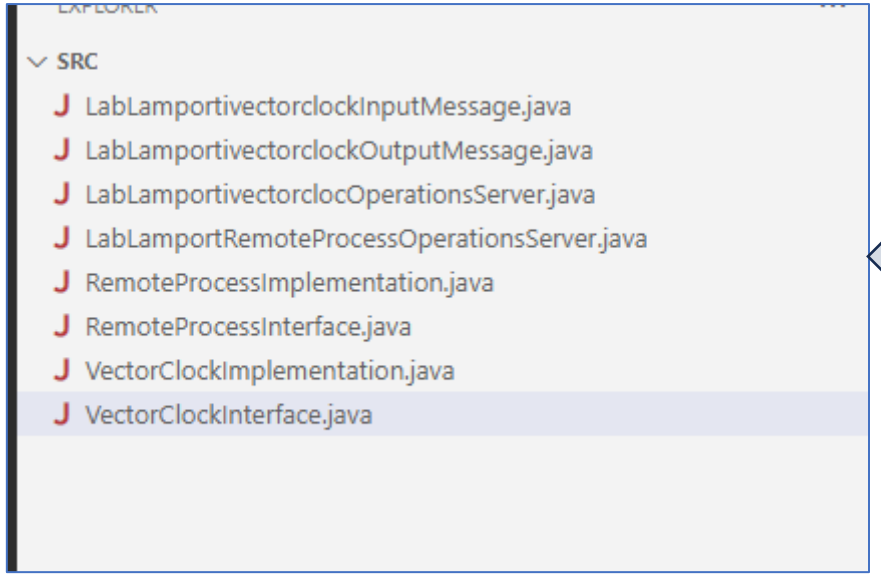
```
Current Local Process: RP0      has the Local logical vector clock: ( 0, 0, 0)
Initial connection Internal Event: Get RemotePRocessOperationsClient registry for process id RP0        Local logical clock: (1, 0, 0)
Initial connection Internal Event: LabRemote Stub creation for process id RP0    Local logical clock: (2, 0, 0)
Choose an Vector operation: 1 GetVectorClock | 2 Send Event and Data
```

Press 1 for GetVectorClock
Press 2 for send event

Process: RP1 Local logical clock: (0, 0 , 0) EventOrdering.NOT_COMPARABLE
Initial connection Internal Event:Create registry for LabLamportRemoteProcess for process id RP1 Local logical clock: (0,1,0)
Initial connection Internal Event:Rebind object for process id RP1 Local logical clock: (0,2,0)
Process: RP1 Local logical clock: (0, 2 , 0) EventOrdering.NOT_COMPARABLE
LabLamport RemoteProcess Server has been started.
Initial connection Internal Event:Receive message for process id RP1 Local logical clock: (0,3,0)
Message received from process: RP0 (logical clock:3) Local logical clock: 3 with logical vector clock value (3,3,0)
Process: P1 Local logical clock: (0, 0 , 0) EventOrdering.NOT_COMPARABLE
Initial connection Internal Event:Send message for process id RP1 Local logical clock: (0,4,0)
Message send to process: Send message: VectorClock getClock method called from Remote Process Local logical clock: 4 : EventOrdering.HAPPENS_AFTER
Initial connection Internal Event:Receive message for process id P1 Local logical clock: (0,1,0)
Message received from process: RP0 (logical clock:3) Local logical clock: 1 with logical vector clock value (3,1,0)
Initial connection Internal Event:Send message for process id P1 Local logical clock: (0,2,0)
Message send to process: RP0 Local logical clock: 2 : EventOrdering.HAPPENS_AFTER
Initial connection Internal Event:Receive message for process id RP1 Local logical clock: (0,5,0)
Message received from process: RP0 Receive message: VectorClock getClock method return from Vector Clock (logical clock:3) Local logical clock: 5 with logical vector clock value (3,5,0)
Initial connection Internal Event:Send message for process id RP1 Local logical clock: (0,6,0)
Message send to process: RP0 Local logical clock: 6 : EventOrdering.HAPPENS_AFTER
Initial connection Internal Event:Receive message for process id RP1 Local logical clock: (0,7,0)
Message received from process: RP0 (logical clock:6) Local logical clock: 7 with logical vector clock value (6,7,0)
Process: P1 Local logical clock: (0, 2 , 0) EventOrdering.NOT_COMPARABLE
Initial connection Internal Event:Send message for process id RP1 Local logical clock: (0,7,0)
Message send to process: Send message: VectorClock Update method called from Remote Process Local logical clock: 7 : EventOrdering.HAPPENS_AFTER
Initial connection Internal Event:Receive message for process id P1 Local logical clock: (0,3,0)
Message received from process: RP0 (logical clock:6) Local logical clock: 3 with logical vector clock value (6,3,0)
Initial connection Internal Event:Send message for process id P1 Local logical clock: (0,7,0)
Message send to process: RP0 Local logical clock: 7 : EventOrdering.HAPPENS_AFTER
Initial connection Internal Event:Receive message for process id RP1 Local logical clock: (0,8,0)
Message received from process: RP0 Receive message: VectorClock Update method return from Vector Clock (logical clock:6) Local logical clock: 8 with logical vector clock value (6,8,0)
Initial connection Internal Event:Send message for process id RP1 Local logical clock: (0,9,0)

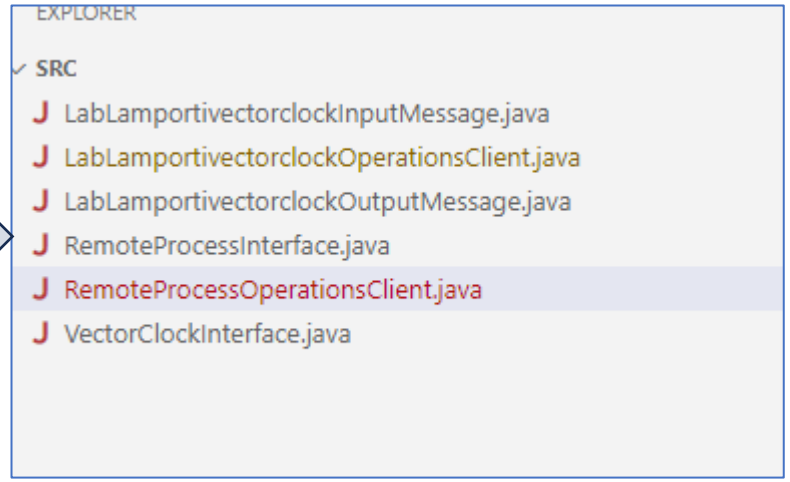
Current Local Process: RP0 has the Local logical vector clock: (0, 0, 0)
Initial connection Internal Event: Get RemotePRocessOperationsClient registry for process id RP0 Local logical clock: (1, 0, 0)
Initial connection Internal Event: LabRemote Stub creation for process id RP0 Local logical clock: (2, 0, 0)
Choose an Vector operation: 1 GetVectorClock | 2 Send Event and Data
1
Initial connection Internal Event: Choose Vector peration for Remote Process Server for process id RP0 Local logical clock: (3, 0, 0)
Initial connection Internal Event: Send message for process id RP0 Local logical clock: (4, 0, 0)
Message send to getClock method hold Process Message send to process: RP0 Local logical clock: 4 : EventOrdering.HAPPENS_BEFORE
Initial connection Internal Event: Receive message for process id RP0 Local logical clock: (5, 0, 0)
Message received from process: P1 of from getVectorClock with logical vector clock:5
Local logical vector clock: 5 with logcial vector clock value is (5, 5,0)
Local Logical Vector Clock value:RP0(5,5,0)
Logical local vector Clock value for Process id:RP0(5,5,0)
Choose an Vector operation: 1 GetVectorClock | 2 Send Event and Data
2
Initial connection Internal Event: Choose Vector peration for Remote Process Server for process id RP0 Local logical clock: (6, 0, 0)
Initial connection Internal Event: Send message for process id RP0 Local logical clock: (7, 0, 0)
Message send to sendEvent method hold Process Message send to process: RP0 Local logical clock: 7 : EventOrdering.HAPPENS_BEFORE
Initial connection Internal Event: Receive message for process id RP0 Local logical clock: (8, 0, 0)
Message received from process: P1 of from Sendevent with logical vector clock:5
Local logical vector clock: 8 with logcial vector clock value is (8, 5,0)
Logical local vector Clock value for Process id:RP0(8,5,0)
Choose an Vector operation: 1 GetVectorClock | 2 Send Event and Data

Server



Logics and algorithm mentioned in the code as comments

Client



Instructions:-

1. Open Visual studio Code
2. Point to the Server location
3. Open LabLamportiVectorClockOperationServer.java
4. Run command from Visual studio code

Instructions:-

1. Open Visual studio Code
2. Point to the Client location location
3. Open LabLAmportivectorClockOperationClient.java
4. Run command from Visual studio code

```
Process: P1     Local logical clock: (0, 0 , 0) EventOrdering.NOT_COMPARABLE
Initial connection Internal Event:Create registry for LabLamportivectorclock for process id P1   Local logical clock: (0,1,0)
Initial connection Internal Event:Rebind object for LabLamportivectorclock for process id P1     Local logical clock: (0,2,0)
Process: P1     Local logical clock: (0, 2 , 0) EventOrdering.NOT_COMPARABLE
LabLamport ivectorclock Server has been started.
```

Communication starts.

```
Current Local Process: P0       has the Local logical vector clock: ( 0, 0, 0)
Initial connection Internal Event: Get LabLamport vectorclock registry for process id P0        Local logical clock: (1, 0, 0)
Initial connection Internal Event: LabLamport vectorclock Stub creation for process id P0       Local logical clock: (2, 0, 0)
Local  Process Id : P0 with intennal event and logical vector clock (2, 0, 0 )
Choose an Vector operation: 1 ClockArray | 2 Increment Vector Clock| 3 Update Vector Clock | 4 MergeVector clock
```

Press 1 for ClockArray
Press 2 for Increment vector clock
Press 3: for Update Vector clock
Press 4: for Merge Clock

Process: P1 Local logical clock: (0, 0 , 0) EventOrdering.NOT_COMPARABLE
Initial connection Internal Event:Create registry for LabLamportivectorclock for process id P1 Local logical clock: (0,1,0)
Initial connection Internal Event:Rebind object for LabLamportivectorclock for process id P1 Local logical clock: (0,2,0)
Process: P1 Local logical clock: (0, 2 , 0) EventOrdering.NOT_COMPARABLE

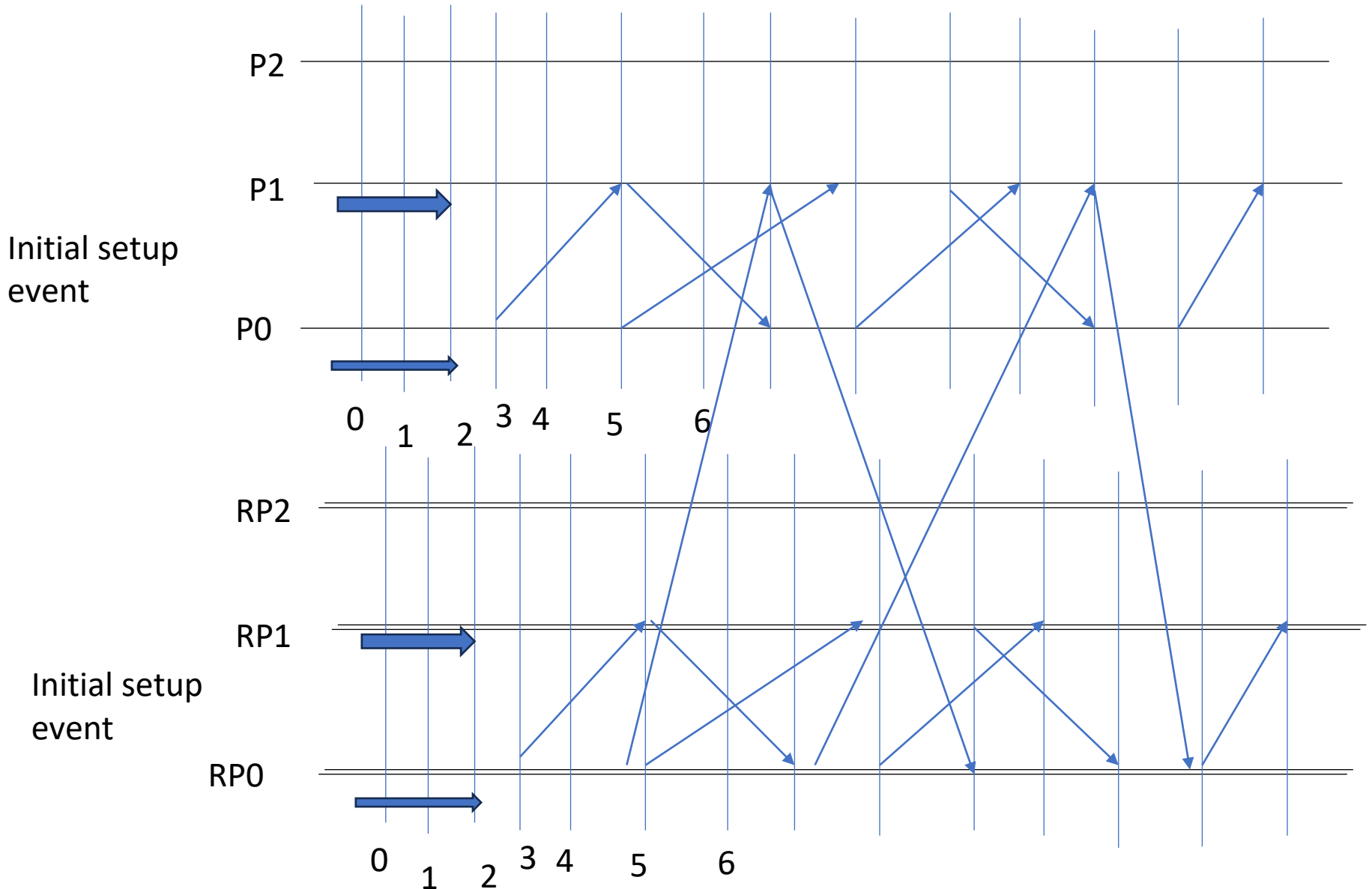
LabLamport ivectorclock Server has been started.

Initial connection Internal Event:Receive message for process id P1 Local logical clock: (0,3,0)
Message received from process: P0 (logical clock:3) Local logical clock: 3 with logical vector clock value (3,3,0)
Initial connection Internal Event:Send message for process id P1 Local logical clock: (0,4,0)
Message send to process: P0 Local logical clock: 4 : EventOrdering.HAPPENS_AFTER
Initial connection Internal Event:Receive message for process id P1 Local logical clock: (0,5,0)
Message received from process: P0 (logical clock:6) Local logical clock: 5 with logical vector clock value (6,5,0)
Initial connection Internal Event:Send message for process id P1 Local logical clock: (0,12,0)
Message send to process: P0 Local logical clock: 12 : EventOrdering.HAPPENS_AFTER
Initial connection Internal Event:Receive message for process id P1 Local logical clock: (0,13,0)
Message received from process: P0 (logical clock:9) Local logical clock: 13 with logical vector clock value (9,13,0)
Initial connection Internal Event:Send message for process id P1 Local logical clock: (0,14,0)
Message send to process: P0 Local logical clock: 14 : EventOrdering.HAPPENS_AFTER
Initial connection Internal Event:Receive message for process id P1 Local logical clock: (0,15,0)
Message received from process: P0 (logical clock:12) Local logical clock: 15 with logical vector clock value (12,15,0)
Initial connection Internal Event:Receive message for process id P1 Local logical clock: (0,16,0)
Message received from process: P2 (logical clock:0) Local logical clock: 16 with logical vector clock value (0,16,0)
Initial connection Internal Event:Concurrency Test for two clock for process id P1 Local logical clock: (0,17,0)
Initial connection Internal Event:Send message for process id P1 Local logical clock: (0,18,0)
Message send to process: P0 Local logical clock: 18 : EventOrdering.HAPPENS_AFTER

Choose an Vector operation: 1 ClockArray | 2 Increment Vector Clock| 3 Update Vector Clock | 4 MergeVector clock

1
Initial connection Internal Event: Choose Vector Operation on the Server Process for process id P0 Local logical clock: (3, 0, 0)
Initial connection Internal Event: Send message for process id P0 Local logical clock: (4, 0, 0)
Message send to getClock method hold Process Message send to process: P0 Local logical clock: 4 : EventOrdering.HAPPENS_BEFORE
Initial connection Internal Event: Receive message for process id P0 Local logical clock: (5, 0, 0)
Message received from process: P1 of from getClock with logical vector clock:4 Local logical vector clock: 5 with logcial vector clock value is (5, 4,0)
Logical Clock array are:P0(5,4,0)
Logical Clock array are:P1(5,4,0)
Logical Clock array are:P2(5,4,0)
Logical local vector Clock value for Process id:P0(5,4,0)
Choose an Vector operation: 1 ClockArray | 2 Increment Vector Clock| 3 Update Vector Clock | 4 MergeVector clock
2
Initial connection Internal Event: Choose Vector Operation on the Server Process for process id P0 Local logical clock: (6, 0, 0)
Initial connection Internal Event: Send message for process id P0 Local logical clock: (7, 0, 0)
Message send to Increment method hold Process Message send to process: P0 Local logical clock: 7 : EventOrdering.HAPPENS_BEFORE
Initial connection Internal Event: Receive message for process id P0 Local logical clock: (8, 0, 0)
Message received from process: P1 of from Increment with logical vector clock:12 Local logical vector clock: 8 with logcial vector clock value is (8, 12,0)
Logical local vector Clock value for Process id:P0(8,12,0)
Choose an Vector operation: 1 ClockArray | 2 Increment Vector Clock| 3 Update Vector Clock | 4 MergeVector clock
3
Initial connection Internal Event: Choose Vector Operation on the Server Process for process id P0 Local logical clock: (9, 0, 0)
Initial connection Internal Event: Send message for process id P0 Local logical clock: (10, 0, 0)
Message send to Update method hold Process Message send to process: P0 Local logical clock: 10 : EventOrdering.HAPPENS_BEFORE
Initial connection Internal Event: Receive message for process id P0 Local logical clock: (11, 0, 0)
Message received from process: P1 of from Update with logical vector clock:14 Local logical vector clock: 11 with logcial vector clock value is (11, 14,0)
Logical local vector Clock value for Process id:P0(11,14,0)
Choose an Vector operation: 1 ClockArray | 2 Increment Vector Clock| 3 Update Vector Clock | 4 MergeVector clock
4
Initial connection Internal Event: Choose Vector Operation on the Server Process for process id P0 Local logical clock: (12, 0, 0)
Initial connection Internal Event: Send message for process id P0 Local logical clock: (13, 0, 0)
Message send to Merge method hold Process Message send to process: P0 Local logical clock: 13 : EventOrdering.HAPPENS_BEFORE
Initial connection Internal Event: Receive message for process id P0 Local logical clock: (14, 0, 0)
Message received from process: P1 of from Merge with logical vector clock:16 Local logical vector clock: 14 with logcial vector clock value is (14, 16,0)
Logical local vector Clock value for Process id:P0(14,16,0)
Choose an Vector operation: 1 ClockArray | 2 Increment Vector Clock| 3 Update Vector Clock | 4 MergeVector clock

Send message, receive message, events are showing for client server architecture with P0,P1 process however P2 is sitting idle.



Thank you

Questions

???