**Assignment 4- Report**

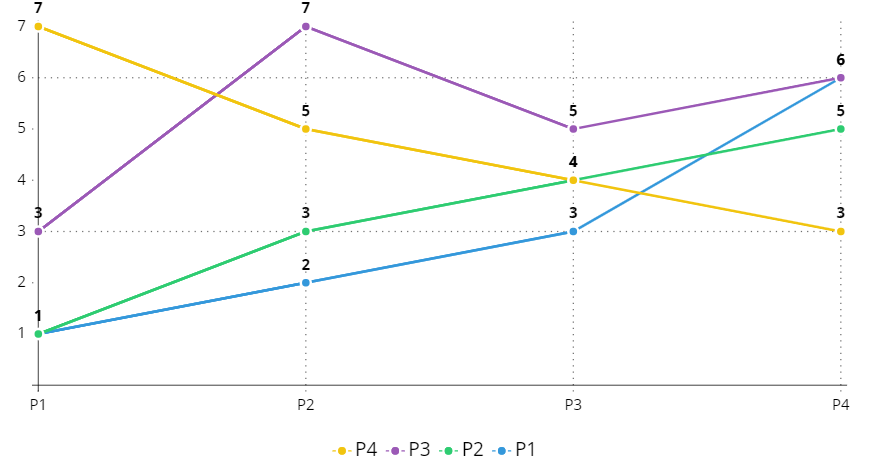
**Introduction:**

RMI stands for Remote Method Invocation. It is based on a client-server architecture model. Here we use a method to send data from one machine to another method in another machine. This is a small experiment to show the principles of remote method invocation using Java RMI. The four processes in our model have their own ID, Message, and Counter. Here the ID determines the process, the message carries a sample test message and the counter is the timestamp that we increment every time any event occurs. Each event in the system is associated with a timestamp which is incremented whenever an event occurs. The event can be internal or sent or receive. This model failed to break or stop the process from its infinite loop.

**Comparison:**

Whenever any event is generated, it checks whether the event is internal or sent to another process. The logical clock is incremented by 1 whenever any message is sent to another process. An acknowledgment is sent to the sender from the receiver and hence the time counter is incremented by a timestamp again. This shows which event happened before using the logical clocks.

Previous code observations for the first 5 loop executions:

****

The graph above is an observation of message passing among processes limited to 5 iterations. Initially, all the process counters are initialized to zero. Pi will randomly decide if it wants to send any message to the internal process or another process and the logical clock is incremented. Pj receives the message and sends back the acknowledgment to Pi and the logical clock is incremented +1 accordingly. Hence, determines that processes agree on the order of events.

**While using RMI:** We use the same implementation as above but there is a **communication** class that handles the sends and receives messages between processes. I found the execution similar between both methods. Multithreading was done on the client side rather than as a request-and-response method.

**Pros:**

* RMI is more robust and flexible to work with.
* Object-oriented message passing in distributed computing.
* It supports Multithreading, hence achieving concurrent execution between client and server.

**Cons:**

* Socket programming is more simple and easy than RMI
* Was not able to run multiple Processes at a time.
* RMI is a Java-only distributed object model.