Indiana University Purdue University Indianapolis

Department of Computer and Information Science

CSCI 53700

Fall 2018

Assignment – 4

Devanapally Sai Krishna

2000276942

Contents

Introduction:

The aim of this assignment is to implement lamports clock scenario using remote method invocation technique's where clocks drift apart in PO's and they send their logical clock value to MO where it corrects it by averaging the logical clocks of multiple PO's including itself and send out the updated clock to its PO's where their values are updated.

Design:

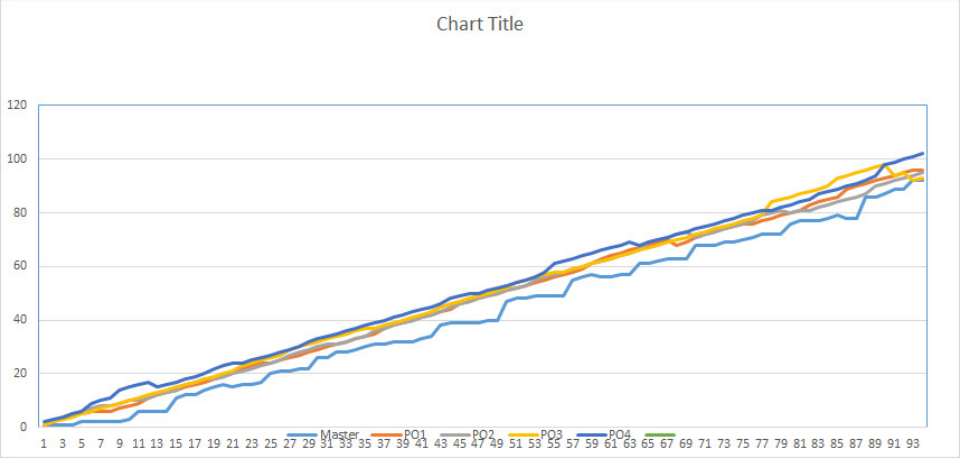
RMI is an extension of the RPC programing with object oriented. A Remote Invocation Method (RMI) runs on a skeleton stub kind of a format. Whenever a request is made on the server from the remote client, the remote object uses the skeleton of the server to send the data through the server remote interface model to invoke the method requested and the stub is the server-side proxy which acks as a gateway to the request.

Here RMI registry comes in play. Client gets the object which is registered with it by using lookup(). Server invokes the method by using the fetched object. After the completion of the usage the server rebinds the object onto the RMI registry.

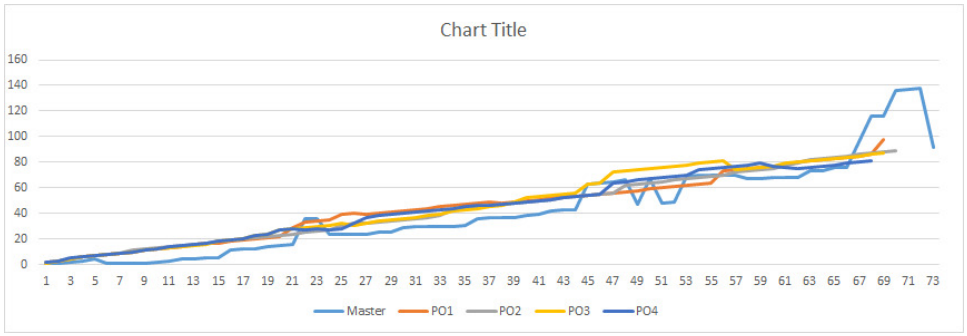
In our design we are sending logical clock values by random from multiple PO's to MO where the average is calculated, and it is sent back to all the PO's to readjust their clocks. The PO's exhibit byzantine behavior also by arbitrarily increasing the clock value.

Results:

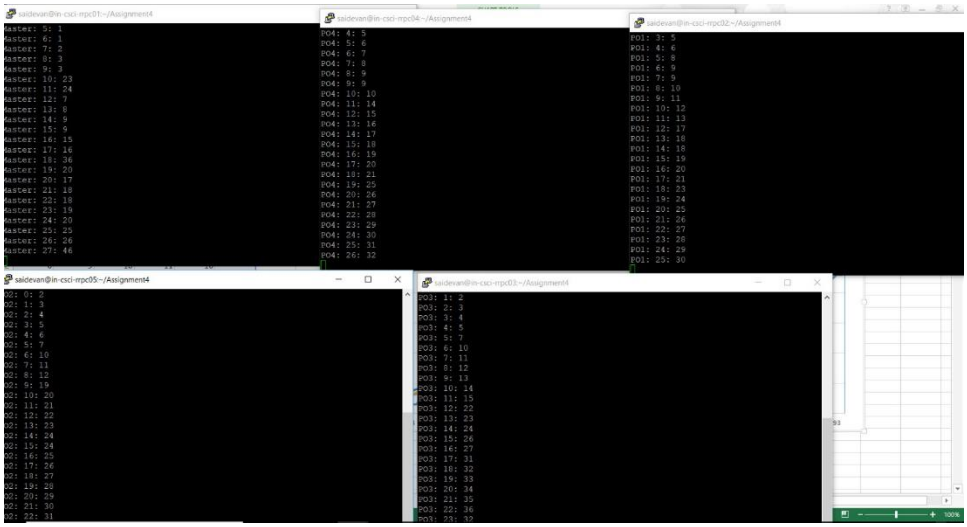
1. Without Byzantine behavior:



2. With Byzantine behavior:



Screenshots:



Comparation Between 1st assignment and RMI:

There are 2 main differences between the two assignments are

1. The threads are automatically handled in RMI but we had to manually handle the threads in first assignment
2. The use of complex data structure like blocking queue is not required here RMI register takes care of it.