BIRLA INSTITUTE OF TECHNOLOGY & SCIENCE, PILANI
Work-Integrated Learning Programme Division
Second semester 2014-15
SS ZG526: Distributed Computing

Assignment No: 2 **Date of Submission: 26/03/2015**
Nature of component: Open book/ Take home **Weightage: 7%**

A number of individual systems wish to work together and operate as a single distributed system. These systems communicate with each other through Remote Procedure Call. There is no centralized component in this system arrangement. Essentially every system has two components.

1. **Server Handler** which can process request from any or all of the other n-1 individual systems.
2. **Client Handler** which can query or request something to any or all of the other n-1 individual systems.

Both the client and server handler can be run through a single executable file or two different executable files. It means a system can work as both client and server at the same time where each peer acts as both client and the server.

Hint: pthreads can be used to run both server and client handler. Server handler can be made to run in the background which responds to requests from another process.

Design this system in C using RPC which will act as your test bed to implement **one of the following of your choice:**

1. Agreement protocols are used to arrive at a consensus between different systems. We have studied about the effect of faulty processes on the outcome of the consensus in Chapter 14. Using the setup we developed, your aim is to test whether the system arrive at a consensus or not using any of the agreement protocol discussed in class. Implement a server and client handler according to the program. A faulty processor can be designed by implementing a server handler which sends random values after it receives a particular value to other systems.

2. Your aim is to implement a leader election algorithm of your choice. You can use bully algorithm or you can tweak Lamport Mutual Exclusion algorithm to have your leader or any other algorithm of your choice. Write your server and client handler accordingly to help you implement a leadership algorithm. When the leader is killed, the next leader should be automatically elected by the other n-1 processes. You can identify a leader is called by pinging it periodically on the port it is listening.

Submit your work by taring the entire source and executable files with your id as the tar file name (e.g.2014h103004.tar). A running sample RPC program for leader election algorithm is attached with this.

I/C
SS ZG526