CS431: Assignment 6

1. Submission Procedure: Email your JAVA source codes as attachment to < asahu AT iitg.ernet.in> and name of the attached file should be #RollNo.Assign6.CS431.tgz/zip, **Copy case Lead to F grade.**
2. **Deadline: 11.55PM 16 November 2013:** Compute server is an important and scarce resource for this assignment. Please try to finish your work as early as possible
3. **Write an 2 page report about your implemented simulator and its performance..**

Assignment Statement: Implement Discrete Events Simulator to simulate one Million (10^6) Agents in Java and test the performance on Compute server 202.141.80.43. No need to implement CDS but use the available ones.

Compare performances of the following Concurrent Priority Queues (CPQ)

(a) Coarse Grain Concurrent Priority Queue (Modify SequentialHeap.java)
(b) Fine Grain Concurrent Priority Queue (Use FineGrainedHeap.java)
(c) Skip List Based CPQ (Use SkipQueue.java along with PrioritySkipList.java)

Java source codes for these PQueues are available at Chapter 15
http://www.elsevierdirect.com/v2/companion.jsp?ISBN=9780123705914/

Description and Pseudo code:

An agent has two states SEND and THINK, in both SEND and WAITING state agent lies inside PQ. Initially all the agents are in THINK state. After end state of an agent, it transit to another state by generating event and waits in the priority queue.

Type of Event	Meaning	State	Prob	Time to execute Event
SEND	Send info to another Agent, Send() need to lock both own lock and receiver lock in order. ***If*** *(i<j) lock(i); lock(j)* ***else*** *lock(j);lock(i)*	SEND	p	Ts=10
THINK	Thinks for random amount of time	THINK	1-p	Rand()%1000

At time CurrTime, all events to be executed will be extracted from PQ, and distributed among T thread to execute. T0 signal other processor/Thread to start the work, other thread waits for signal from T0 and T0 always increment the current time.

// A typical Simulation Pseudo code (in C++/Java like) is given….

```
Struct/Classs Agent{
     Lock L; List <int> MessageBox;
     Think(){Messagebox.clear();}
     Send(Agent A, int Data){ A.AddMessage(Data);}
     AddMessage(int Data){MessageBox.push_back();}
};
struct Event {
     int AgentNum, TypeOfEvent, Receiver, Data, FiringTime;
};
Initialize(){
     for(i=0;i<TotalAgent;i++) AddEvent(i, time0);
     }
```

```
void AddEvent(int i, time T){
     if ( (rand()%100)< (p*100.0) ){
          EventObj=new Event(i,SENT,rand()%TotalAgent,
                    rand()%100, T+Ts);
          PQ.add(EventObj, T+Ts)
       } else {ThinkTime=T+rand()%1000;
               EventObj =new Event(i, THINK, ThinkTime);
               PQ.add(EventObj,  ThinkTime);
          }
}
void Execute Event(Event E){
     if (E.TyepOfEvnt==THINK) {
     E.AgentNum.Lock();E.AgentNum.Think();E.AgentNum.Unlock();
     }else{ // Event is type SENT
          if(E.AgentNum<E.Receiver){//Lock in order
               E.AgentNum.Lock();E.Receiver.Lock();
               E.AgentNum.Send(E.Receiver, E.Data);
               E.Receiver.unlock();E.AgentNum.unLock();
          } else {
               E.Receiver.Lock();E.AgentNum.Lock();
               E.AgentNum.Send(E.Receiver, E.Data);
               E.AgentNum.unLock();E.Receiver.unlock();
               }
     }
}
void Work(){
     while(1){
     if ( CurrTime > MaxSimultionTime )exit(1);
     E = PQ.ExtractMin(); //Assume lots of events are in PQ with
                          // fire time CurrTime+1
     if(Threadid==0){
        if(E.FiringTime<=CurrTime) {
               ExecuteEvent(E);AddEvent(E.AgentNum, CurrTime);
               }
          else { CurrTime++; signalAllThread();}
     } else {
          if(E.FiringTime<=CurrTime){
               ExecuteEvent(E);AddEvent(E.AgentNum, CurrTime);
               }
          else wait();
       }
     }//end while
}
void main(){
     Initialize();
     CreateNThreads(Work);
}
```