Assignment 2

Simulation Program Design

Team Members :
Ajinkya (213050034), Arnav (213059002)

Outline

1. System Assumptions
2. Constants , user Inputs and Enums
3. Classes
4. Basic Logic (FCFS)
5. Basic Logic (Round Robin)

System Assumptions

System Type - Closed system

Each User Issues a Fixed number of Requests

Number of cores - 4

Number of Maximum Threads per core - 10

Request Buffer Size - 100

Constants User inputs and Enums

Constants-
- Max_buffer_Size
- Max_thread_count
- Conext_switch_time
- Max_Request_Generated

Users Inputs –
- Mean_interarrival,
- Mean_service
- Number_of_users

Scheduling Policy ***(Enum)***
- FCFS (1)
- Round Robin (2)

Server Status ***(Enum)***
- Idle (1)
- Busy (2)

Event Types ***(Enum)***
- Arrival (1)
- Departure (2)
- Context_Switch_In (3)

Distribution Type ***(Enum)***
- Exponential (1)
- Uniform (2)
- Constant (3)

Classes

Service_time

- *Attributes:*
  - typeOfDistribution(Enum Distribution type)
- *Methods:*
  - getServiceTime() /*Generation function{describes the distribution}*/

Timeout

- *Attributes:*
  - constantTime (double)
  - typeOfDistribution(Enum Distribution type)
- *Methods:*
  - getTimeoutTime() /*Generation function{describes the distribution}*/

Classes

Event

- Attributes:
  - arrival_time (double)
  - timeout(double)
  - serviceTime
  - core (int)
  - thread(int)
  - response_count
- Methods:
  - getRandomThinkTime()/*random value chosen in range [4,10]*/
  - getRemainingServiceTime()

Classes

Core

- *Attributes:*
  - threads [Max Thread Count] (Event Object List)
  - status (int) {Server Status}
  - thread_busy_count (int)
- *Methods*:
  - GetCoreStatus()
  - setCoreStatus()
  - addToThread()
  - removeFromThread()
  - getBusyThreadCount()
  - setBusyThreadCount()

Classes

Scheduler

- *Attributes:*
  - Type (int) {Scheduling policy}
  - Context_switch_time (double)
- *Methods:*
  - switching the threads ()

Server

- *Attributes:*
  - Core Object [4];
  - service_time Object;
  - Scheduler Object;
  - {Waiting Buffer} Event Obj queue [Max buffer size] (shared among all cores)
- *Methods:*
  - getNextEventFromBuffer()
  - getServerStatus()
  - setServerStatus()
  - getCoreObj()

Classes

Event Handler

- *Attributes*:
  - Server Obj
  - timing_next_event[Max_event_count] (a priority queue of tuples <event_time, event obj> prioritized on event_time)
  - Timeout Obj
- *Methods*:
  - getNextEvent()
  - manageEvent()
  - Arrive()
  - Depart()
  - getServerObj()
  - setEvent()

Base Logic (FCFS)

Main Function ()

```
{
    read_input() // take user input
    Initialize() //Initialize the simulation

    Event_handler_Obj = new Event_Handler();

    While (No requests < max Request count){
        Event_handler_obj.Next event();
        Event_handler_obj.manageEvent();
    }
}
```

Base Logic (FCFS)

```
Function Arrival(Event X)
{
    if(no of user < max user){
        Create new EVENT obj and assign arrival time
    }
    if(server busy){
        Put Event X in Event queue;
    }
    Else{
        Set the departure time of the event X.
    }
}
```

Base Logic (FCFS)

Function Departure (Event X)

```
{
     if(Event queue empty){
          Set status idle  of the core belonging to the
          event X that called up the departure
     }
     Else{
          Remove event A from the waiting buffer
          Set the departure time of the event A.
     }
     If Event X -> departure time <  Event X -> timeout{
          Event X -> Response_count++;
     }
     /*Get the Event Object Ready of next Request*/
     Set think-time of event X  randomly
     Set the arrival time of event X based on the think-time
}
```

Base Logic (Round Robin)

```
Main Function ()
{
    read_input() // take user input
    Initialize() //Initalize the simulation

    Event_handler_Obj = new Event_Handler();

    While (No requests < max Request count){
        Event_handler_obj.Next event();
        Event_handler_obj.manageEvent();
    }
}
```

Base Logic (Round Robin)

```
Function Arrival(Event X)
{
     if(no of user < max user){
          Create new EVENT obj and assign arrival time
     }
     if(server busy){
          Put Event X in Event queue;
     }
     Else{
          If Scheduler.timeQuantum > event X.remainingServiceTime()
               Set contextinTime for Event X.
          else
               Set departureTime for Event X.
     }
}
```

Base Logic (Round Robin)

Function Departure (Event X)

```
{
     if(Event queue empty){
          Set status idle  of the core belonging to the
          event X that called up the departure
     }
     Else{
          Remove event A from the waiting buffer
          If Scheduler.timeQuantum > event A.remainingServiceTime()
               Set contextinTime for Event A.
          else
               Set departureTime for Event A.
     }
     If Event X -> departure time <  Event X -> timeout{
          Event X -> Response_count++;
     }
     /*Get the Event Object Ready of next Request*/
     Set think-time of event X  randomly
     Set the arrival time of event X based on the think-time
}
```