SHARTS

SHArif's Real-Time Simulator

User Guide – July 2015

Table of Contents

Contents

1. General Information

This section contains general information about real-time schedulers and system requirements for using this software.

1. 1. Real-Time Simulator

A Scheduler that contains tasks with respective deadlines, is called real-time scheduler. This software simulates a real-time scheduler with different scheduling algorithms.

1. 2. System Requirements

One of this qualifications:

- Android OS version 2.3.7 or higher
- Windows OS and Visual Studio 2012 or higher (No GUI supported)

2. Introduction

This simulator takes the definition of a task-set and try to schedule that with a user-selected algorithm. Tasks in a task-set can be in both types of periodic or aperiodic. Before starting of the scheduling process, simulator checks schedulability condition for a task-set under user-selected algorithm. Then simulator starts scheduling and shows process to the user. At the end of simulation, simulator prints some useful information about scheduling.

3. Getting Started

This section guides user to use software in both Android and Windows operating systems.

3. 1. Using in Android OS

Download the *sharts.apk* from *android_v1.0* folder and install that on your system.

In first of running software, below page is shown.

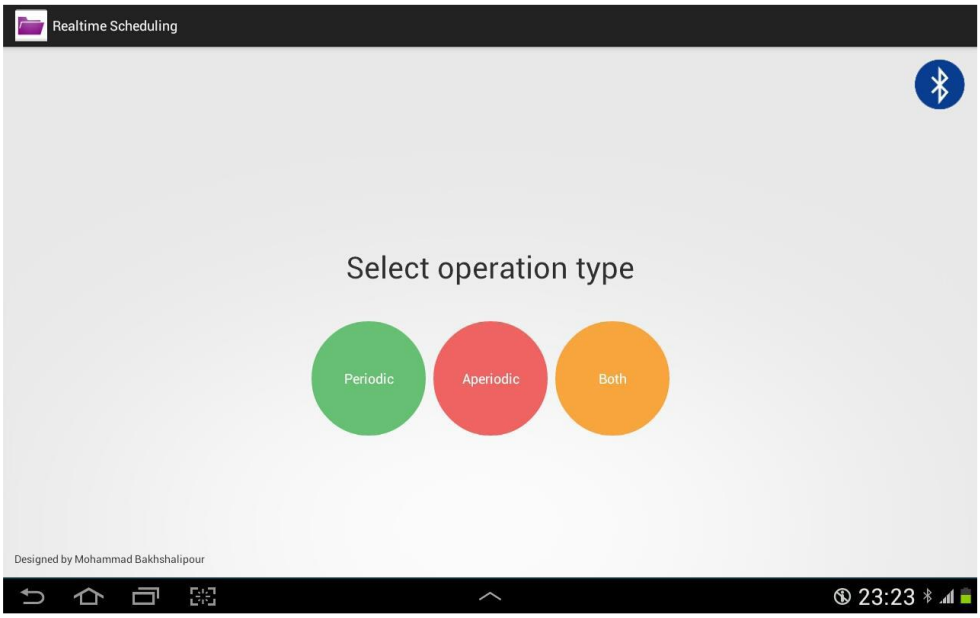

Figure 1. First Page of Software on Android OS

3. 1. 1. Sending Software Installation File

Click on the button at right, top of the page to send the software via media like Bluetooth. By clicking button this page will be shown and you can choose media and destination device and send the software installation file.

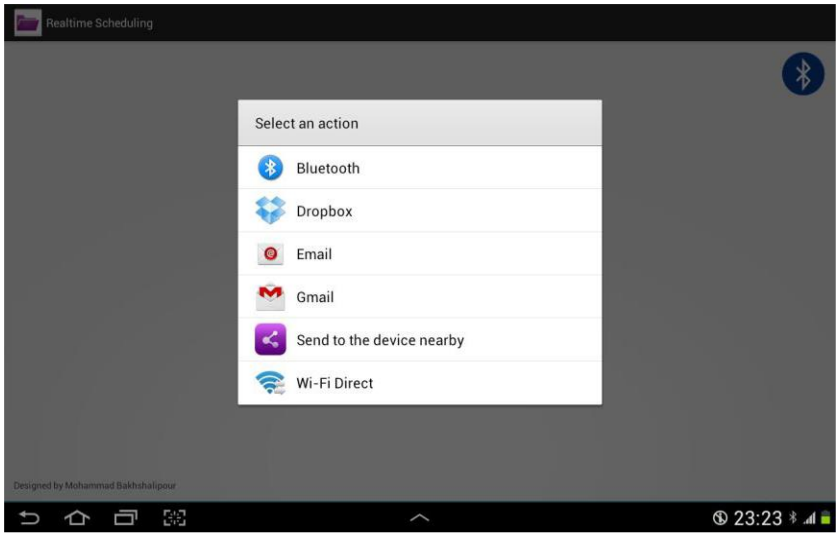

Figure 2. Sending Software Installation File

3. 1. 2. Scheduling of Periodic Task-Set

By selecting 'Periodic' button on the first page, related page will be shown. In this page, you should enter information of tasks. Enter *Name, Period, Computation Time* and *Deadline* in the respective fields like below fig.

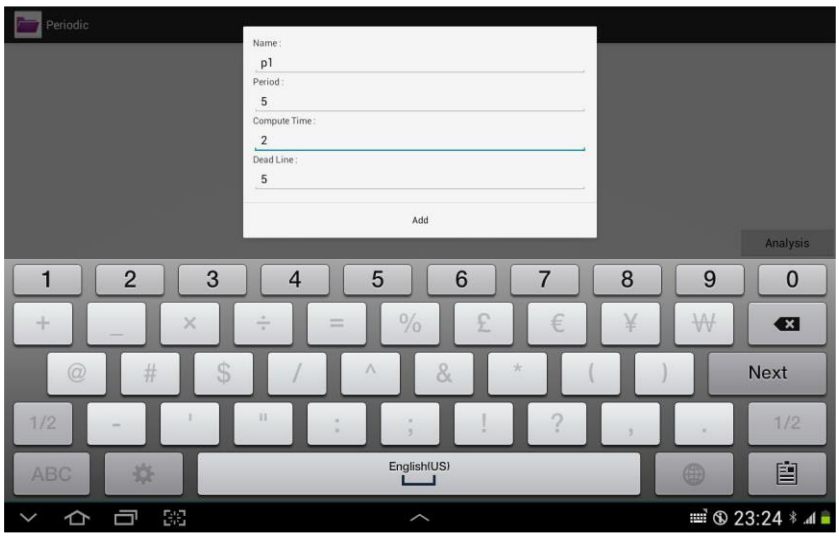

Figure 3. Scheduling of Periodic Task-Set

After adding tasks, information of tasks will be shown like below fig.

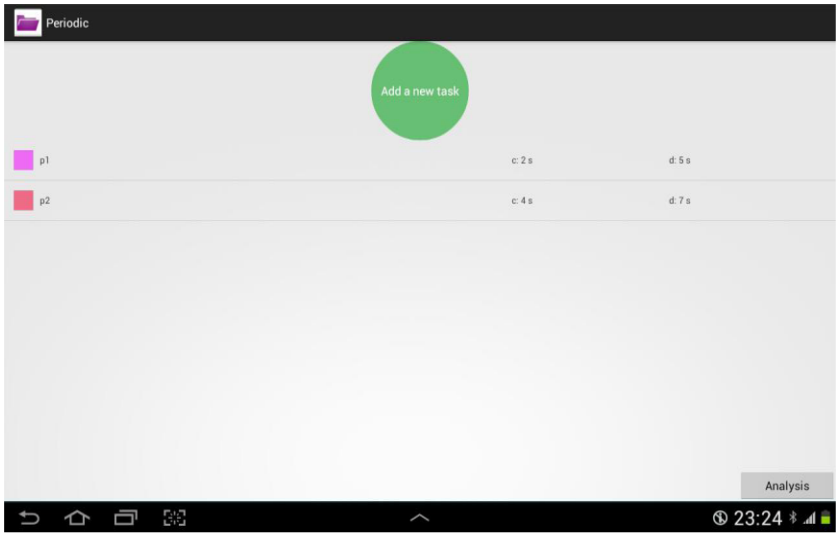

Figure 4. Information of Tasks

By clicking the Analysis button, you can choose a scheduling algorithm like below fig.

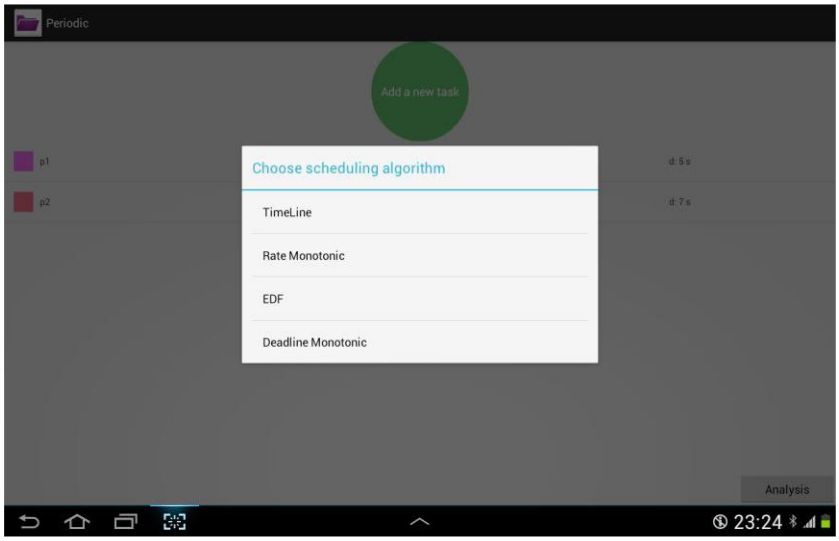

Figure 5. Choosing Scheduling Algorithm

After selecting algorithm, the schedulability condition of task-set under selected algorithm will be checked like below fig.

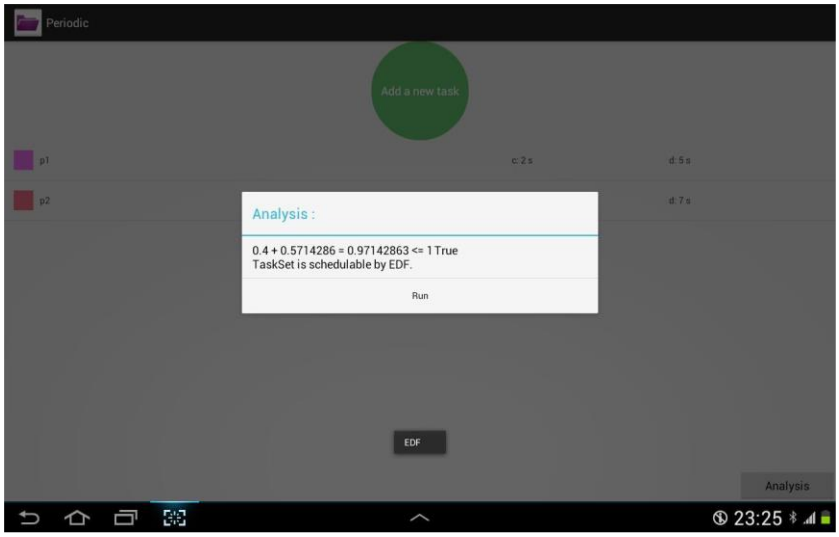

Figure 6. Checking Schedulability Condition

After seeing that, you can simulate scheduling by clicking on 'Run' button or go to back step by 'back' button.

By clicking on 'Run' button, the simulation will be started and at the end, simulation process and results will be shown like below fig.

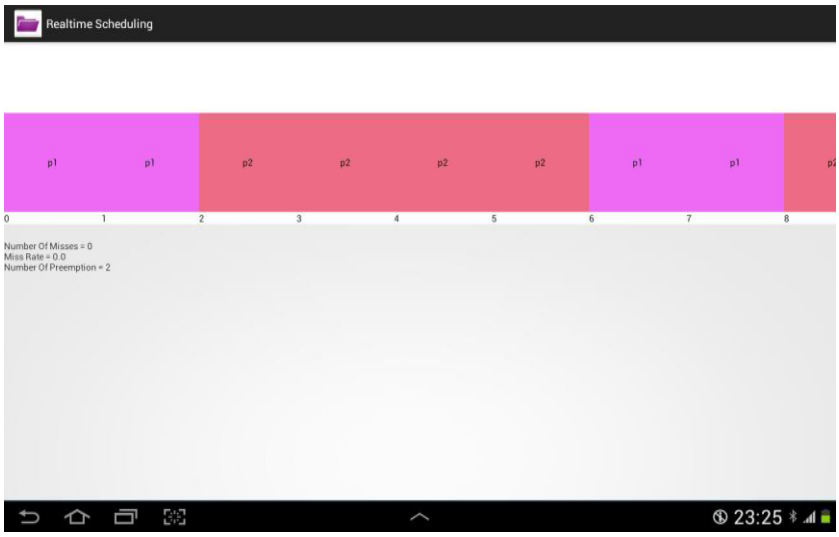

Figure 7. Simulation Process and Results

3. 1. 3. Scheduling of Aperiodic Task-Set

For scheduling aperiodic tasks, you should choose 'Aperiodic' button on the first page. By this, below page will be shown as a result. You should enter characteristics of task, such as *Name, Computation Time, Deadline* and *Release Time* in respective fields.

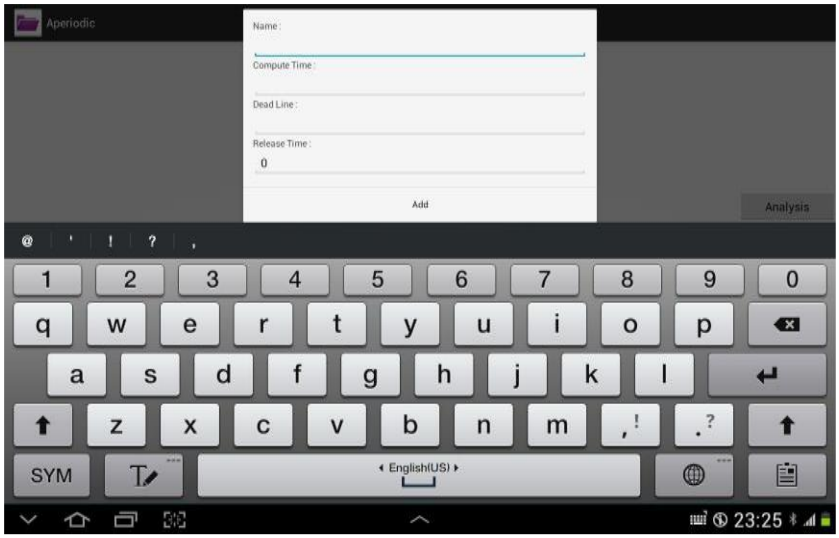

Figure 8. Scheduling of Aperiodic Task-Set

The other steps are like 3. 1. 2. Scheduling of Periodic Task

3. 1. 4. Scheduling of Mixed Periodic-Aperiodic Task-Set

For scheduling mixed task-set, click on 'both' button on the first page. By this a page like below fig will be shown.

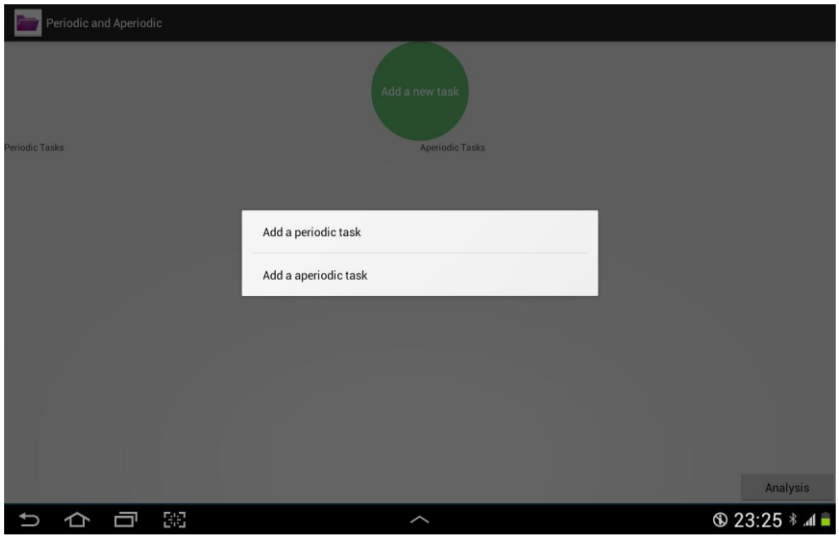

Figure 9.Scheduling of Mixed Task-Set

For defining periodic and aperiodic tasks, choose respective button and act like 3. 1. 2. Scheduling of

Periodic Task-Set and 3. 1. 3. Scheduling of Aperiodic Task-Set.

3. 2. Using in Windows OS

Download the whole folder *windows_v1.0* and open the *RealTime.sln* file with Visual Studio. In the file named *Program.cs*:

- Define a periodic task by using 'PeriodicTask' class that takes *string name, int computionTime, int period* and *int relativeDeadline* as arguments of constructor in order.
- Define an Aperiodic task by using 'AperiodicTask' class that takes *string name, int releaseTime, int computationTime* and *int absoluteDeadline* as arguments of constructor in order.
- Define a Periodic task-set by using 'PeriodicTaskSet' class that takes *List<PeriodicTask> tasks, int numberOfTasks* and *int algorithm* as arguments of constructor in order. *algorithm==0* choose TimeLine, *algorithm==1* choose Rate Monotonic, *algorithm==2* choose EDF and *algorithm==3* choose Deadline Monotonic algorithm for scheduling.
- Define an Aperiodic task-set by using 'AperiodicTaskSet' class that takes *List<AperiodicTask> taskSet, int numberOfTasks* and *int algorithm* as arguments of constructor in order. *algorithm==0* choose Jackson and *algorithm==1* choose Horn algorithm for scheduling.
- Define a Mixed task-set by using ' MixedTaskSet' class that takes *List<PeriodicTask> periodicTasks, List<AperiodicTask> aperiodicTasks* and *int algorithm* as arguments of constructor in order. *int algorithm* variable defined because of future implementation of Mixed task-set scheduling and now cannot be used.

At the end of program, you can use *PrintTaskSetInfo()* to show the results of scheduling. At the end, compile whole the project and run the scheduling.