International Journal of Grid and Distributed Computing
Vol. 11, No. 2 (2018), pp.1-10
http://dx.doi.org/10.14257/ijgdc.2018.11.2.01

Comparative Analysis of Simulation Tools with Visualization based on Real-time Task Scheduling Algorithms for IoT Embedded Applications

Shabir Ahmad[1], Sehrish Malik[2] and Do-Hyeun Kim[3*]
[1,2,3]Jeju National University, Republic of Korea
[1]shabir@jejunu.ac.kr, [2]serrym29@gmail.com, [3]kimdh@jejunu.ac.kr

Abstract

In real-time tasks scheduling, intended to run on embedded devices, one of the problems which are begging to be addressed is to oversee the deadline of input tasks and their likelihood of missing the deadline and to ensure their execution within the deadline. This leads to the fact that a task can only be submitted if the system has adequate resources to execute the task without missing any deadline. In order to tackle this challenge, feasibility analysis of the real-time input tasks need to be carried out in a virtual space before applying it on a real embedded device. This paper presents a comparative analysis of existing simulators with visualization technologies which assists in the feasibility analysis of real-time input tasks for IoT embedded applications. The paper considers popular algorithms like Rate Monotonic and EDF. And we analysis on various simulators tools like ARTISTT, MAST, Cheddar, STORM, STRESS, Realtss, SimSo, GHOST, VizzScheduler and Yartiss in terms of using recorded trace, live simulation, sporadic tasks, shared resources, programming language, CPU utilization

Keywords: *Internet of Things, Real-time Scheduling, Simulation, Shared Resources, Safety Critical Systems, Visualization*

1. Introduction

Systems are referred to as real-time when their correct behavior not only depends on the logically correct output, but also on the time at which these tasks are performed [1-3].

In some situation the time constraint must be achieved [Hard Real-Time], while in some cases it can be missed marginally. For instance, in avionics control system, the software responsible for controlling the flight must finish execution within its specified deadline interval for accurately controlling the aircraft. Similarly, in automotive electronics systems similar strict timing constraints are imposed on engine management and transmission control systems related tasks that originate from the mechanical systems that they control. This heavy use of computer applications can lead to a loss of a life if the program failed to meet the deadline.

Consequently, there is a strong need to establish statistical and theoretical basis of the correct behaviors of the tasks [2]. This is sometime called Feasibility analysis of the system and it governs whether a tasks set will meet their deadlines during simulation [10]. If the simulations run as expected and all the tasks run within its specified deadline every time then these tasks are marked guaranteed and are ready to go live on a physical embedded based systems.

Conventionally, these analysis occur offline, before even a system starts executing tasks [3, 5]. In order to conduct feasibility analysis of the system efforts are made to provide virtual environments and simulators in order to facilitate the job and provide a

Received (May 25, 2017), Review Result (August 28, 2017), Accepted (September 1, 2017)
[*] Corresponding Author

virtual space. These simulators have their own pros and cons and have application specific requirements. This paper reviews these tools and conducts comparative analysis of them and identifies the main shortcomings that are still needed to be addressed for a dynamically changing multicore technology. The article also suggests the guideline for an ideal simulator for real time Internet of Things (IoT) applications.

The rest of the paper is organized as follows; Section 2 covers 3 covers simulation tools, Section 4 discusses and summarizes the tools covered in Section 3 and finally Section 5 draws conclusion and final remarks.

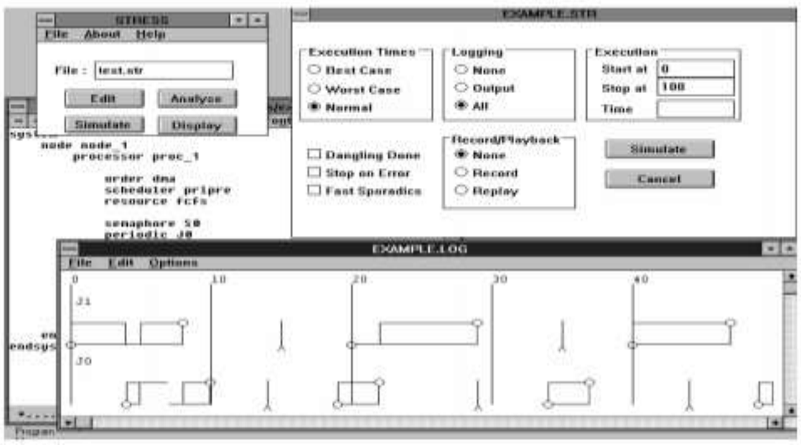
3. Comparative Analysis of Simulation Tools based on Real-time Task Scheduling Algorithms

In the previous sections it has been discussed that an effective simulator tool is of utmost importance to conduct feasibility analysis of real-time input tasks. To this end, many efforts are put up by researchers. In this section a review of some of the popular simulator and visualization tools are presented and discussed.

3.1. STRESS

The first notable real-time simulation tool [3] whose focus is solely based on the analysis and simulation of hard real-time systems' behavior which is sometime also called safety critical applications. STRESS is developed in its own programming language dedicatedly written for it called stress.

However, it is written in 1994 so the GUI is not very well-written as depicted in Figure 1. Moreover, it only support hard real time systems so as a consequence cannot be used for soft real time systems.



File : test.str

```
system
  node node_1
      processor proc_1

          order dma
          scheduler pripre
          resource fcfs

          semaphore S0
          periodic J0
...
  end
endsystem
```

Execution Times: ○ Best Case ○ Worst Case ● Normal

Logging: ○ None ○ Output ● All

Execution: Start at 0, Stop at 100, Time

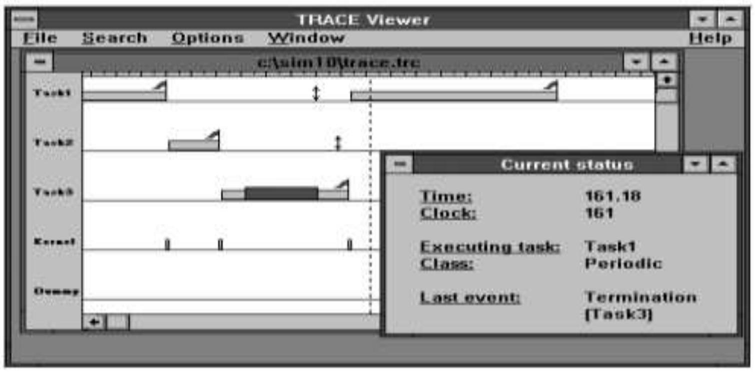
☐ Dangling Done ☐ Stop on Error ☐ Fast Sporadics

Record/Playback: ● None ○ Record ○ Replay

Figure 1. STRESS GUI

3.2. GHOST

GHOST is acronym for General Hard real-time Oriented Simulator Tool (GHOST) and is stress-inspired real-time scheduling simulator [4]. GHOST has built-in support for hard real-time systems, soft real-time systems and non-real-time system having no deadline constraints. It also works well in protocols designed for resource allocation. The structure of the GHOST has tasks as the major building block. These tasks can be grouped in the form classes and each one of the classes can be handled by its own task scheduler unit. Besides, custom scheduling policies and algorithms can also be supplied creating a new class in C programming language. The main drawback of this algorithm is that it only supports single core systems and so as this work could be extended to support multi-core systems. Figure 2 shows a sample GUI of tasks timeline and the current status of CPU.



Time: 161.18
Clock: 161

Executing task: Task1
Class: Periodic

Last event: Termination [Task3]

Figure 2. GHOST GUI

3.3. MAST

MAST [6] is comprehensive software to analyze different real-time scheduling algorithms. It leverages theoretical concepts to check whether a given input task set is schedulable or not by using some suitable heuristics for multi-core systems. UML tools aid in modeling the input tasks. It only focuses on sporadic tasks and fixed priority algorithms.

3.4. ARTISST

Another tool known as ARTISST [5] has been widely used by many research institutes. It is a real-time tasks simulation tool designed specifically for event-driven jobs and allows modeling the inner control flow graph of input tasks. It also takes the context switch time into consideration and the result of the simulation is shown as a chronogram or by statistical values. However, like its counterparts, it does not support shared resources. The GUI is depicted in Figure 3 which shows sample CPU timeline and respective tasks. Hold_cpu is the time a task occupies cpu and added as the task executes.

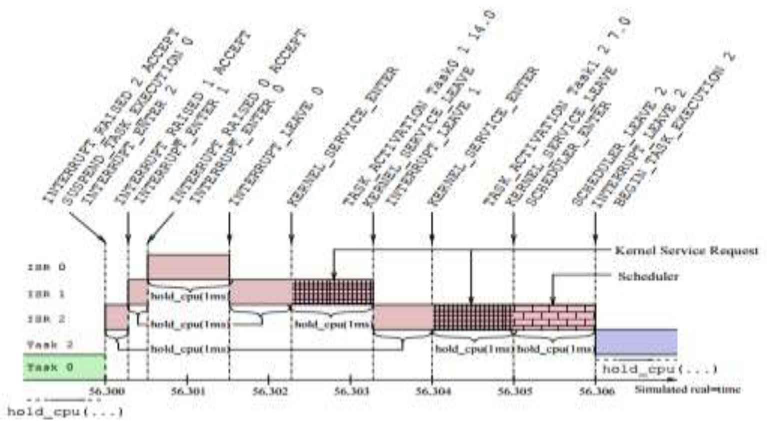
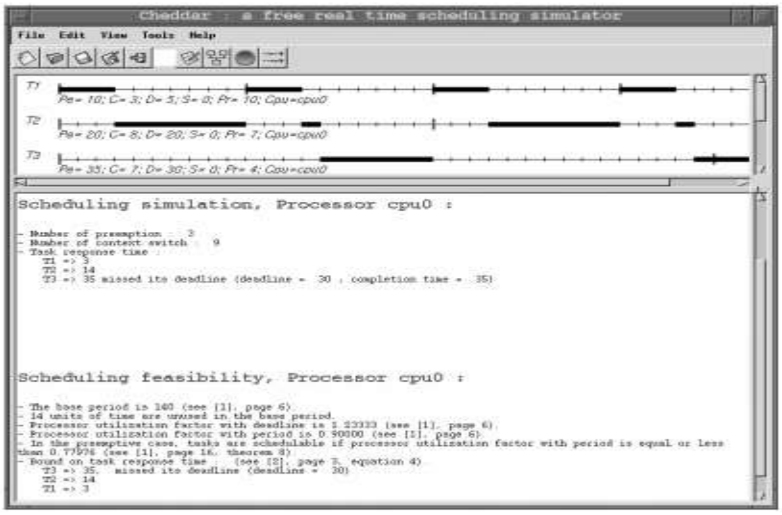


Figure 3. ARTISTT Tasks Timeline

3.5. Cheddar

Singhoff in [11] proposed Cheddar a novel tool designed for teaching purposes. It is MAST-inspired simulation tool. Cheddar has the ability to both visualize and simulate real-time tasks. It is written in ada-like language called DSL. It does support shared resources, multicore features and have a very powerful visualization interface; however, Real simulation is something which could be tried in future.


Cheddar : a free real time scheduling simulator
File Edit View Tools Help
T1
Pe= 10; C= 3; D= 5; S= 0; Pr= 10; Cpu=cpu0
T2
Pe= 20; C= 8; D= 20; S= 0; Pr= 7; Cpu=cpu0
T3
Pe= 35; C= 7; D= 30; S= 0; Pr= 4; Cpu=cpu0

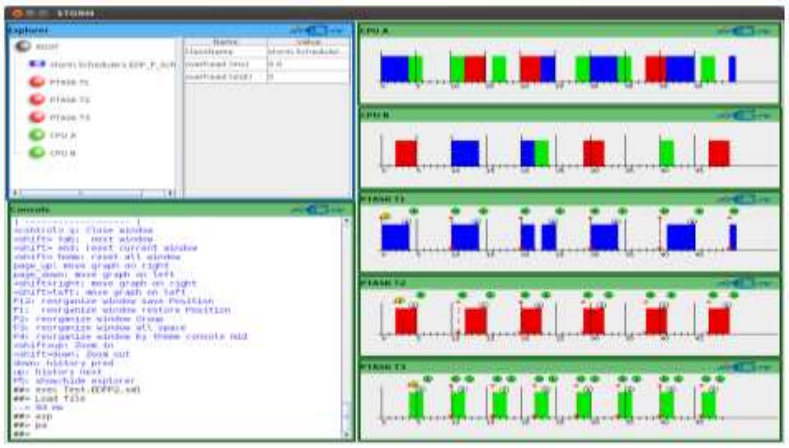
```
Scheduling simulation, Processor cpu0 :

- Number of preemption : 3
- Number of context switch : 9
- Task response time :
    T1 => 3
    T2 => 14
    T3 => 35 missed its deadline (deadline = 30 ; completion time = 35)



Scheduling feasibility, Processor cpu0 :

- The base period is 140 (see [1], page 6).
- 14 units of time are unused in the base period.
- Processor utilization factor with deadline is 1.23333 (see [1], page 6).
- Processor utilization factor with period is 0.90000 (see [1], page 6).
- In the preemptive case, tasks are schedulable if processor utilization factor with period is equal or less
than 0.77976 (see [1], page 16, theorem 8).
- Bound on task response time :  (see [2], page 3, equation 4).
    T3 => 35,  missed its deadline (deadline = 30)
    T2 => 14
    T1 => 3
```

Figure 4. Cheddar GUI

Figure 4 shows graphical user interface of Cheddar. Tasks are represented as T1, T2 and so on multiple cpu timelines. The bottom part of the figure shows the summary of a scheduling result.

3.6. STORM

Storm [8] is a yet another effort for the simulation of real-time tasks designed for multiprocessor scheduling and thus has inherent support for multicore as well as shared resources. Similar to GHOST, for every entity like tasks, CPU cores, schedulers and resources there is a class representation. So a new algorithm can be implemented by just adding a user defined class.



```
| ------------------- |
<control> q: close window
<shift> tab: next window
<shift> end: reset current window
<shift> home: reset all window
page_up: move graph on right
page_down: move graph on left
<shift>right: move graph on right
<shift>left: move graph on left
F12: reorganize window save Position
F1: reorganize window restore Position
F2: reorganize window Grow
F3: reorganize window all space
F4: reorganize window by theme console mid
<shift>up: Zoom in
<shift>down: Zoom out
down: history pred
up: history next
F5: show/hide explorer
##> exec Test_EDFP2.sml
##> Load file
... 88 ms
##> exp
##> pe
##>
```

Figure 5. STORM GUI

Figure 6 shows the GUI of STORM. Tasks are represented as red color circles while CPUs are represented as green color circles. The right portion shows the simulation the tasks and their respective tasks timeline. The major problems with this are that it is no longer open source and the source code is not available.

3.7. SimSo

Simso [7] is simulator tool designed for the comparison and the understanding of real time scheduling policies. Currently, it has more than 20 algorithms supported and it can be easily extended for other algorithms. It is open-source tool based on Python programming language. This tool has also the challenge of shared resource support and the GUI is not state of the art.

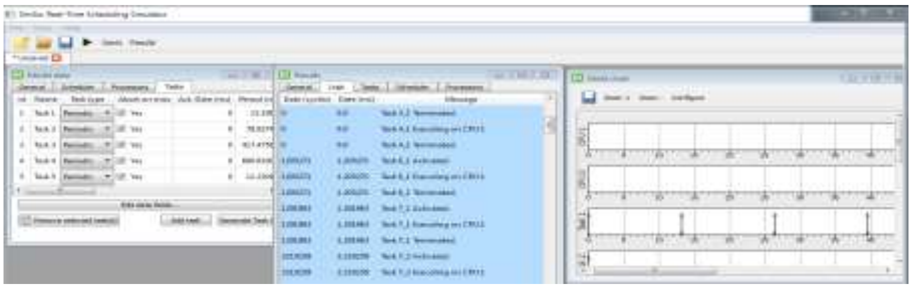


Figure 6. SimSo GUI

Figure 6 depicts the main component of SimSo simulator. It has different interfaces for tasks creation, cpu creation and scheduling algorithm selection. This is called setup phase. Once setup phase is done the result of the algorithm is visualized using Gantt chart.

3.8. RealTSS

RealTSS [9] is an open-source real time scheduling simulator. It is inspired from cheddar and it is designed as teaching and research tool. It is written in TCL but new features can also be added in C or C++.

It is platform independent and can be run on any general purpose and embedded operating system. The GUI is based on KIWI which is very powerful but on the downside it has no support for shared resources.

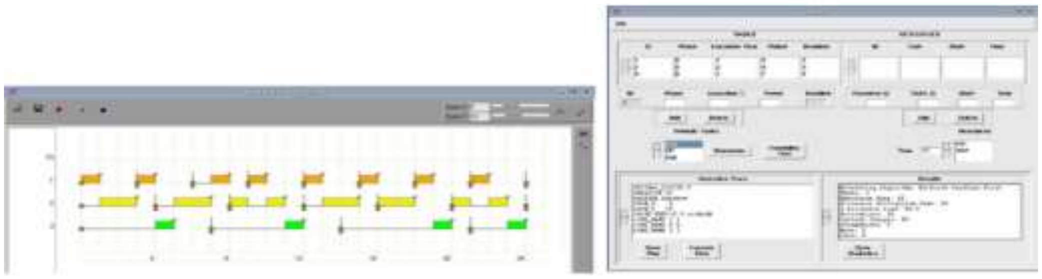


Figure 7. RealTss GUI

Figure 7 shows the simulation result visualized using KIWI based interface. The tasks are given its specific colors to distinguish it from others.

3.9. VizzScheduler

VizzScheduler [12] is a framework for the simulation and visualization of real-time tasks scheduling algorithms. It uses *LogP* cost model for parallel machine and simulate effect of various algorithm based on actual system jobs. It has different components to perform different tasks. For example, VizzEditor support rapid design of general algorithm for visualization. VizzSchedular is an instance of VizzEditor and simulate the jobs.

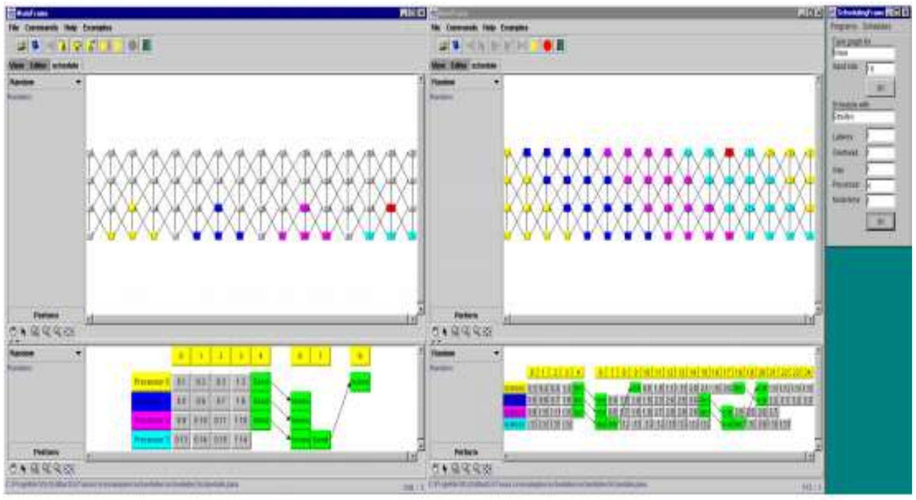


Figure 8. VizzSchedular Visualization

Figure 8 shows algorithm simulation for a one dimensional wave. The underneath portion shows the task graphs and CPU time.

3.10. Yartiss

Yartiss [13] is multi-processor based simulation and visualization tool designed for energy -based real-time tasks. This tool follows a modular approach and all the modules are loosely coupled so new modules for new algorithms can be added very easily. Yartiss leverages graph theory concepts to design and visualize tasks. For instance, tasks are represented as nodes and assignments of tasks to CPU are represented with edges. Figure 9 shows the various interfaces of task visualization when the rate monotonic algorithm runs.

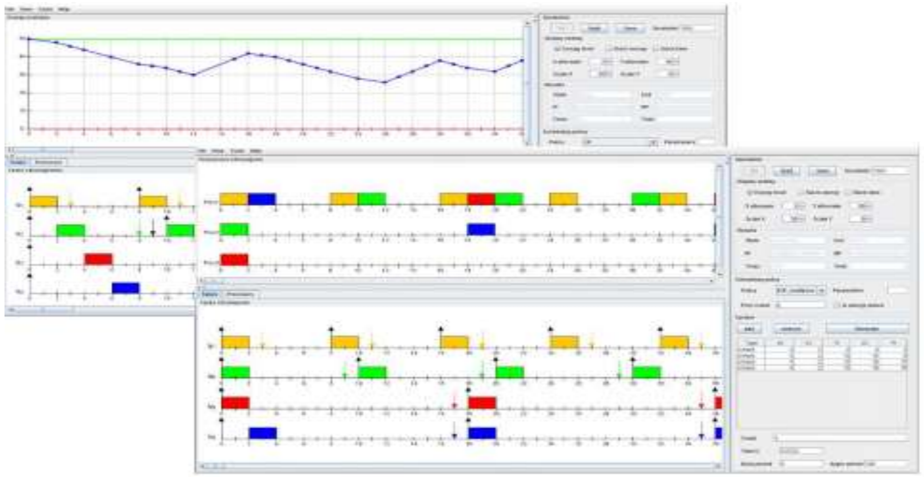


Figure 9. Yartiss GUI Interfaces

4. Comparative Review Results and Discussions

The criteria of effectiveness of a simulation tool are characterized as the utilization of resources and the amount of tasks achieved their desired behaviors. Table 1 describes the desired characteristics of the real-time scheduling algorithms.

In the subsequent chapters a comparative analysis has been carried out based on the architecture diagram mentioned in Figure 1 and the effectiveness of the tools are assessed with respect to the parameters outlined in Table 1.

Table 1. Output Performance Factors

#	Name	Remarks
1	Open Source	This means that source code of the simulator tool is under public GNU license and is available for the community use.
2	Multiple Cores	Whether the tool support multiple core or not.
3	Live Simulation	Support Simulation of actual tasks
4	Sporadic Tasks	Support for non-periodic sporadic tasks
5	Shared Resources	The support to get access to share resources.
6	Programming language	The language the simulator used for the development
7	CPU Utilization	The utilization factor the algorithm

Simulation tools like Cheddar [11], STORM [8], STRESS [3], Realtss [9], SIMSO [7], and GHOST [4] have been considered for the review and have been assessed thoroughly against performance factors identified in Table 1. The result of the review is summarized in Table 2.

Table 1. Review of Various Simulation Tools

Tool Name	Live Simulation	Multicore Support	Open source	Visualization	Sporadic Tasks	Shared Resources	Programming Language
ARTISTT	No	No	Yes	Yes	Yes	No	C++
MAST	No	Yes	Yes	No	Yes	Yes	Ada
Cheddar	No	Yes	Yes	Yes	Yes	Yes	DSL Ada-Inspired
STORM	No	Yes	Yes	Yes	Yes	Yes	Java
STRESS	No	Yes	No	Yes	Yes	Yes	C
RealTSS	No	No	Yes	Yes	Yes	Yes	TCL
SimSo	No	Yes	Yes	Yes	Yes	No	Python
GHOST	No	No	No	Yes	Yes	No	C
Yartiss	No	Yes	Yes	Yes	Yes	No	Java
VizzScheduler	Yes	Yes	No	Yes	No	No	Java

In this section the main features outline in above-mentioned simulation tools are summarized and discussed. Table 2 outlines the algorithm and its main characteristics in terms of the performance parameters devised in Table 2.

It can be seen from the figure that STORM and Cheddar may be used in every real time application simulation, if they need not to have a requirement live simulations. Similarly, other algorithms also perform well in respect to the requirement for which they have developed. For example, GHOST is focus on solely on visualization and hence all other parameters are not considered.

That being said, the need to have a tool which can be used generically for all the cases and satisfies all the performance parameters outlined in this text is getting more demanding overtime.

Web-based technologies are also getting more attention and many popular frameworks use these technologies for interfacing and visualization. So to utilize the majesty of these state of the art technologies it would be worth looking to develop and visualize the virtual simulator which will be platform independent and can be accessed everywhere.

5. Conclusion and Future Work

Many real-time simulator tools are exist and so many efforts are made to design a tool that is very generic and efficient and can possibly fit in all types of tasks. In this paper a survey of various simulation tools and visualization tools conducted based on embedded devices. Rate Monotonic (RM) and Earliest Deadline First (EDF) algorithms are considered for the study. It has been found that live streaming and shared resources are some of the attributes if the simulator that is desired to have in a tool but most of the tools studied lacks these attributes. Moreover, the power of web-based technologies is not yet utilized effectively for visualization of these tools which can be considered worth looking as a future work.

Acknowledgment

This work was supported by Institute for Information & communications Technology Promotion (IITP) grant funded by the Korea government (MSIT) (No.2017-0-00756, Development of interoperability and management technology of IoT system with heterogeneous ID mechanism), and this research was supported by the MSIT (Ministry of Science and ICT), Korea, under the ITRC (Information Technology Research Center) support program (IITP-2017-2014-0-00743) supervised by the IITP (Institute for Information & communications Technology Promotion), Any correspondence related to this paper should be addressed to DoHyeun Kim.

References

[1] K. Ramamritham and J. A. Stankovic, "Scheduling algorithms and operating systems support for real-time systems", Proceedings of the IEEE, vol. 82, no. 1, **(1994)** January, pp. 55-67.

[2] A. Gamatié and T. Gautier, "Synchronous modeling of avionics applications using the Signal language", In Real-Time and Embedded Technology and Applications Symposium, 2003, Proceedings of the 9th IEEE, **(2003)**, pp. 144-151.

[3] N. C. Audsley, A. Burns, M. F. Richardson and A. J. Wellings, "STRESS: A simulator for hard real time systems", Software: Practice and Experience, vol. 24, no. 6, **(1994)**, pp. 543-564.

[4] F. Sensini, G. Buttazzo and P. Ancilotti, "Ghost: A tool for simulation and analysis of real-time scheduling algorithms", In Proceedings of the IEEE Real-Time Educational Workshop, **(1997)**, pp. 42-49.

[5] D. Decotigny and I. Puaut, "ARTISST: an extensible and modular simulation tool for real-time systems", In Object-Oriented Real-Time Distributed Computing, 2002.(ISORC 2002), Proceedings. Fifth IEEE International Symposium on IEEE, **(2002)**, pp. 365-372.

[6] M. González Harbour, J. J. Gutiérrez García, J. C. Palencia Gutiérrez and J. M. Drake Moyano, "Mast: Modeling and analysis suite for real time applications", In Real-Time Systems, 13th Euromicro Conference on IEEE, **(2001)**, pp. 125-134.

[7] M. Chéramy, P.-E. Hladik and A.-M. Déplanche, “SimSo: A simulation tool to evaluate real-time multiprocessor scheduling algorithms”, In 5th International Workshop on Analysis Tools and Methodologies for Embedded and Real-time Systems (WATERS), **(2014)**, pp. 6-p.
[8] R. Urunuela, A.-M. Déplanche and Y. Trinquet, “Storm a simulation tool for real-time multiprocessor scheduling evaluation”, In Emerging Technologies and Factory Automation (ETFA), 2010 IEEE Conference on IEEE, **(2010)**, pp. 1-8.
[9] A. Diaz, R. Batista and O. Castro. “Realtss: a real-time scheduling simulator”, In Electrical and Electronics Engineering, 2007. ICEEE 2007. 4th International Conference on IEEE, **(2007)**, pp. 165-168.
[10] B. Nikolic, M. Ali Awan and S. M. Petters, “SPARTS: Simulator for power aware and real-time systems”, In Trust, Security and Privacy in Computing and Communications (TrustCom), 2011 IEEE 10th International Conference on IEEE, **(2011)**, pp. 999-1004.
[11] F. Singhoff, J. Legrand, L. Nana and L. Marcé, “Cheddar: a flexible real time scheduling framework”, In ACM SIGAda Ada Letters, ACM, vol. 24, no. 4, **(2004)**, pp. 1-8.
[12] W. Löwe and A. Liebrich, “Vizzscheduler-a framework for the visualization of scheduling algorithms”, Euro-Par 2001 Parallel Processing, **(2001)**, pp. 62-66.
[13] Y. Chandarli, F. Fauberteau, D. Masson, S. Midonnet and M. Qamhieh, “Yartiss: A tool to visualize, test, compare and evaluate real-time scheduling algorithms”, In WATERS 2012, UPE LIGM ESIEE, **(2012)**, pp. 21-26.

Authors

Shabir Ahmad is currently pursuing Ph.D. in the Department of Computer Engineering in Jeju National University, Republic of Korea. He received his MS in Computer Software Engineering from National University of Science and Technology, Islamabad, Pakistan in 2013. He did his BS in Computer System Engineering from University of Engineering and Technology, Peshawar, Pakistan and is now serving in the same university as a faculty member of Software Engineering Department.

Sehrish Malik, she received her B.S. in Computer Science from National University of Computer and Emerging Sciences (NUCES-FAST), Pakistan. In September 2014, she moved to Republic of Korea for M.S. studies and started working in the Protocol Engineering Laboratory (PEL), Sangmyung University (Cheonan-si Campus). After completing her M.S. in 2016, she moved to Jeju-do in September 2016, and started working as a Ph.D. research fellow in the Mobile Computing Laboratory (MCL), Jeju National University. Research interests include IoT, sensor networks, actuator networks, privacy, trust, and communication systems.

Do-Hyeun Kim, He received the B.S. degree in electronics engineering from the Kyungpook National University, Korea, in 1988, and the M.S. and Ph.D. degrees in information telecommunication the Kyungpook National University, Korea, in 1990 and 2000, respectively. He joined the Agency of Defense Development (ADD), from Match 1990 to April 1995. Since 2004, he has been with the Jeju National University, Korea, where he is currently a Professor of Department of Computer Engineering. From 2008 to 2009, he has been at the Queensland University of Technology, Australia, as a visiting researcher. His research interests include sensor networks, M2M/IOT, energy optimization and prediction, intelligent service and mobile computing.