An application specific comparison of Real Time Scheduling Algorithms for failing task sets

Abhijit Kulkarni & Bhargava Kolanupaka

Abstract − **This paper presents an implementation of for non-ideal task sets and an approach towards the implementation and comparison of the fixed and dynamic priority real-time scheduling algorithms on a two car platooning system using two OSEK compliant NXT cars are built using the Lego Mindstorms robots. The master car (first car) is designed to follow a line using a color sensor and the servant car (second car) is designed to follow the master car using an ultrasonic sensor. A Bluetooth connection is established between the two cars and a simple track is designed to test the behaviors and the success ratio of the cars using the fixed and dynamic priority algorithms. The two cars have a set of interdependent task sets which help accomplish the concept of an automated platooning system. To demonstrate the impact of failures of such tasks on the system, a hard real-time system and non-ideal task sets containing a set of tasks or a single task which fail to execute randomly are considered. After dealing with implementation issues in NXT OSEK with real-time scheduling algorithms like RM, DM, and EDF, a FIFO based scheduling algorithm was implemented and tested with non-ideal task sets. A success rate of 42.25% was observed under the condition in which any task in the task set fails randomly.**

Key words – Non-ideal task sets, platooning system, OSEK, NXT, Bluetooth, RM, DM, and EDF.

I. INTRODUCTION

Automated platooning system is one of the most active research areas in the recent past. Automated vehicles have many benefits when compared to normal human driven cars due to the fact that they help decrease the minimum distance between two cars, thereby decreasing congestion and increasing the traffic flow rate. They reduce sudden acceleration and deceleration of the cars thereby reducing fuel usage, and increasing the mileage, eventually reducing pollution on highways. Decreasing the traffic flow leads to saving the number of man-hours spent in traffic [1], which ultimately increases the productivity. Further, it is a better solution which helps avoid the expenses and environmental damage of the only-then solution of widening the roads [2] for the increased traffic. Most importantly, automated vehicles are more reliable when compared to human driven vehicles because a majority of accidents on roads are due to human errors [3], hence automated vehicles may also decrease the fatality rate due to road accidents.

Implementation of automated vehicles requires real-time scheduling of various tasks timely to successfully increase traffic flow rate and avoid accidents. In this regard, we studied the behavior of automated vehicles and their impacts under both fixed and dynamic priority scheduling algorithms. Under the given the time frame, to accomplish our idea we simulated the concept of platooning system by building two cars of which one follows the other with the available equipment in our hands. The initial objective of the project was to build multiple cars (more than two), design complex tracks, and make the cars follow one after another in all directions. However, because of the limited time frame of a month, we planned to shift our focus from working on the already-existing algorithms which enables the preceding cars to take turns as the first one, to the implementation of real-time scheduling algorithms on OSEK platform and studying their behaviors with non-ideal task sets. We decided to implement two cars, a master & a servant car, and a simple straight line track to implement, compare and study the RM, DM, and EDF algorithms on non-ideal task sets.

The non-ideal task sets are typically a set of tasks of which one or more of the tasks may fail to execute at an instance, or goes into infinite delay randomly. The non-ideal task sets are used in the project to replicate and study the impact of a situation, where the cars on the road, sometimes, may fail to behave as expected suddenly due to a numerous factors such as a large computation time, overload of resources, sudden change in terrain, sudden breakdown of cars on highways, and so on.

To summarize our work, the following are our contributions in this article:

1) We built a two car platooning system with OSEK compliant NXT cars, a master car and a slave car, using Lego Mindstorms robots.
2) To enable communication between the two cars we established a Bluetooth connection.
3) We designed a simple straight line track and implemented a FIFO based algorithm on the platooning system.
4) We also introduced non-ideal task sets to study the impacts of unexpected behaviors of the cars on the roads.
5) We ran different combinations of test cases, i.e., implementing the failure of each tasks at once, and then failure of more than one task at a time to study the effects on the system. We studied the effects on the system and estimated the success ratios and the success rate of the algorithms.

The rest of the paper is outlined as follows. In section II, we discussed the related work. Section III describes the system model and task sets. Section IV details the implementation of the real-time scheduling algorithms, and the non-ideal task sets. Section V presents the results, and finally section VI presents conclusion and future work of the paper.

II. RELATED WORK

Platooning system is a very important concept in automated vehicle systems, which is an active research area from the past decade. [3] [4] provided a basic overview of automated vehicle system (AVS). [5] described the working of a platooning system. Claas et al in [6] described an EDF scheduler plug-in for OSEK/VDX.

III. SYSTEM MODEL & TASK SETS

The system model, as shown in the Figure 1, has two cars built using NXT Lego Mindstorms robots. We chose NXT Lego Mindstorms robots because we wanted to implement real-time scheduling algorithms on OSEK compliant models and replicate a situation where the cars on the road may fail to behave as expected suddenly due to factors as mentioned in the Introduction. We also wanted to have the cars built easily and be available at low cost.

The first car (the one on the right in Figure 1) is the master car which has a color sensor and a touch sensor. The color sensor is programmed such that it enables the car to follow a line. The touch sensor is used to calibrate the color/light sensor. The second car (the car on the left in Figure 1) is the servant car which follows the master car using an ultrasonic sensor. The ultrasonic sensor is programmed to maintain a constant distance between the two cars by measuring the distance between them continuously. Whenever the distance between the two cars increases beyond a particular distance, the servant car sends a Bluetooth message to the master car informing it to stop immediately. Similarly, when the distance between them is reduced and is below the particular pre-set distance, the servant car stops and allows the first car to move. Hence, a certain distance is maintained between the two cars while traveling using the Bluetooth connection and the sensors. As shown in the Figure 1, the two cars travel with coordination on the straight track (black line) autonomously.

Figure 1 System Model

The non-ideal task set was implemented by generating a a random integer inside the task body. The decision to execute, abruptly terminate or infinitely delay the task was made on the basis of the randomly generated integer. Each car has a task set. The task set of the master car has four tasks as stated below:

- Bluetooth task
- Motor control task
- Light sensor task
- Initialize task

Similarly, the task set of the servant car has three dedicated tasks as stated below:

- Bluetooth task
- Motor control task
- Sonar sensing task

The two tasks are interdependent and hence the failure of one task may lead to the failure of multiple tasks or simply the failure of the entire system. For example, the failure of the Bluetooth communication task of the master car leads to the failure of communication between the two cars which also implies that the Bluetooth communication task of the servant car also failed. It might also affect the motor control task of the servant car, thus failing the enter system. The working of each of the above stated tasks are explained as follows:

i. Bluetooth task

The Bluetooth task runs continuously in the background in both the master and servant cars, and maintains the Bluetooth connection between them.

ii. Motor control task

The Motor control task is a periodic task. In the master car, it acts as a proportional controller. It calculates the error values by calculating the difference between the light sensor reading which is updated by the light sensor task and the initial reference value obtained by the calibrate sensor Task. The task then configures the car to change direction (left or right) on the basis of the error value obtained by assigning corresponding speeds to left and right motors of the car. In the master car, it also sends a message to the servant car when it starts moving forward and receives a message to stop when the distance between the master and the servant car exceeds a certain limit.

In the servant car, it assigns speeds to both the motors of the car to move forward when it receives a message from master car to move forward. It sends a message to the master car informing it to stop moving forward when the distance between the two cars exceed a pre-set distance.

iii. Light sensor task

The light sensor task is also a periodic task like the motor control task, but it is assigned only to the master car. It takes readings from the color sensor of the car and updates them continually for every 25 ms. the motor control task uses the readings obtained from the light sensor task.

iv. Sonar sensing task

The sonar sensing task is a periodic task assigned to the servant car. It is used to update the distance between the two cars continuously for every 25 ms.

v. Initialize task

The initialize task is configured to be executed at the beginning to calibrate the color sensor by obtaining the readings for the colors black and white using the color sensor in master car when the touch sensor is pressed. This is done to achieve line following.

IV. IMPLEMENTATION AND CHALLENGES

Randomly failing tasks: One of the main contributions of our work is a task set in which tasks fail randomly. This was achieved by generating a random integer in the range 0-9 and add conditional statements to alter the behavior of the task according to them. At any given instant, the task would either be terminated abruptly, be delayed or execute successfully.

Rate Monotonic (RM) and Deadline Monotonic (DM): All the tasks for the two NXT bricks were implemented in C and OIL (OSEK Implementation Language). To implement RM and DM, the tasks were configured to run as periodic tasks that repeat for periods as mentioned in the previous section.

We faced some challenges during the development process. One of them was the fact that the Bluetooth connection task had to be run continuously on both the master and the slave devices for the entire duration for which the cars were to be run. The master car's Bluetooth task would constantly call the `ecrobot_init_bt_master()` to ensure that the connection stays established. The servant car's Bluetooth task on the other hand constantly calls the

`ecrobot_init_bt_slave()` for the same purpose.

Scheduling the Bluetooth task as a periodic task affected the basic working of the NXT brick since it would get reset frequently. Hence, we had to work with the limitation of having the Bluetooth connection task running in the background.

The above limitation had a major impact during the implementation of RM and DM. We had to configure the Bluetooth task to run in the background and the rest of tasks to run periodically. Priorities were assigned both according to the period of the task and the deadline of the task and tests were run. It was found during these tests that only the Bluetooth task and the task with the highest priority would get executed and hence the rest of the tasks would not get executed. This would lead to the non-functioning of the system.

Earliest Deadline First: OSEK supports only Static Priority based Scheduling and hence implementing EDF was another major challenge that this effort involved. Claas et al described an EDF scheduler plug-in for OSEK/VDX. We decided to implement the EDF plugin for our experiment.

After dealing with some initial implementation issues, we found out that the EDF plugin would not work because of the fact that NXT OSEK does not fully conform to the OSEK VDX specification.

We changed our approach towards implementing EDF. The new implementation involved creating a task which would calculate the absolute deadlines of a task and at any given instant, depending on the difference between the current time stamp and the absolute deadline of all tasks and run the task with the nearest deadline by invoking an event that would be serviced by it. We could not conduct experiments using the above method since there were implementation issues which were being dealt with at the time of writing.

It has to be taken into consideration that the issues with Bluetooth connection task remained and is suspected to have an impact on the EDF implementation as well.

Working Model: With the issues discussed above, we set out to ensure that the basic functionality of the system is achieved. This was done by configuring all the tasks to run periodically with equal priority with the Bluetooth connection task running in the background. First In First Out (FIFO) scheduling is used for tasks with equal priority in OSEK.

V. RESULTS

As a part of the experiment, we ran the system several times under failure of different combinations of tasks. The main metric for this experiment was the number of times the system behaved as expected (in percentage).

Table 1 shows the success percentages of the system when individual tasks such as the Bluetooth communication task, the light sensor task and the ultrasonic sensor task fail for individual cars and when different combinations of these tasks fail. C1 implies car 1 and C2 implies car 2. The most important result from the table is the overall failure which accounts for the case in which any task in either cars fail at a given instant.

Table 1 Results

Task Set	Success (%)
Bluetooth Communication (C1)	50
Light Sensor (C1)	80
Bluetooth & Light Sensor (C1)	40
Bluetooth & Ultra Sonic Sensor (C2)	60
Overall (Random failure)	42.25

VI. CONCLUSION & FUTURE WORK

This paper describes an implementation of non-ideal task sets on the NXT-OSEK platform. It also described a few approaches towards the implementation of fixed and dynamic priority real-time scheduling algorithms on a two-car system built using OSEK compliant NXT Lego Mindstorms robots.

Due to constraints and limitations discussed in section IV, RM and DM implementations were not successful. A FIFO based scheduling algorithm was eventually implemented and experiments were conducted with the same.

With the implementation of non-ideal task set, we observed that failure of critical tasks such as Bluetooth communication tasks & light sensor tasks together,

and Bluetooth task alone led to failure of the complete system in most of the cases.

EDF implementation issues were still being investigated at the time of writing.

We believe that this work can be extended by implementing algorithms for the servant car to follow the master car by taking turns which provides a better platooning system to study with non-ideal task sets.

Fault tolerance mechanisms as mentioned in [7] can be applied to the above system to reduce the effect of failure of tasks as shown in the system considered.

REFERENCES

[1] B. E. a. T. L. D. Schrank, "TTI's 2012 urban mobility report," *Texas A&M Transportation Institute. The Texas A&M University System, Tech. Rep.,* Jun. 2012.

[2] G. D. a. M. Turner, "The fundamental law of road congestion: evidence from US cities," *American Economic Review,* Vols. vol. 101, no. 6, p. pp. 2616–2652, Oct. 2011.

[3] D. F. a. K. Kockelman, "Preparing a nation for autonomous vehicles: opportunities, barriers and policy recommendations," *Transportation Research Part A: Policy and Practice,* vol. vol. 77, p. pp. 167–181, Jul. 2015.

[4] S. Tsugawa, "Automated driving systems: Common ground of automobiles," *International Journal of Humanoid Robotics,* vol. 8, pp. 1–12,, 2011.

[5] T. C. a. K. H. D. Desiraju, "Minimizing the disruption of traffic flow of automated vehicles during lane changes," *IEEE Transactions on Intelligent Transportation Systems,* Vols. vol. 16, no. 3, p. 1249–1258, Jun. 2015.

[6] U. M. F. S. G. W. Claas Diederichs, "An application-based EDF scheduler for OSEK/VDX," in *Design, Automation and Test in Europe*, Munich, 2008.

[7] K. G. S. J. W. Ching-Chih Han, "A Fault-Tolerant Scheduling Algorithm for Real Time Periodic Tasks with Possible Software Faults," *IEEE Transactions on Computers,* vol. 52, no. 3, pp. 362 - 372, 2003.