HAN_UNIVERSITY
OF APPLIED SCIENCES

Individual assignment

Distributed Systems
Capita Selecta

Optimal deterministic schedule strategies for Real Time Embedded System applications with Time Triggered Architecture

Semester 2 Term 2

Diego Martín López 659292

19th May 2021

Abstract

Hard Real Time systems require certain processes to finish within some time bounds in order to ensure deterministic behaviour and so reliability and safety are procured. Time Triggered schedules build its fundamentals on these ideas to find an ordered task set that guarantees the requirements of these particular problems. However, these schedules are specific for the problem characteristics they have to solve so there is not a general approach. Many industrial applications decide not to use them because of their rigidity for system updates due to a lack on automation. In attempt for searching some automated algorithms to schedule task sets, three different articles are explored in this report which feature systems uni-core and multi-core, present different criticality jobs or employ a Real Time Operating System. Each of them need different approaches rooted within the same idea: try to find the most general algorithm given the requirements of the system design so the algorithm search can be automatized with results as optimal as possible. One of them presents deployment documentation with the major challenges that the authors encountered while implementing the system in real life.

1 Introduction

As computing processing becomes faster the number of Real-Time applications grows exponentially because program load in embedded systems becomes a minor challenge to overcome. However, engineers find other problems when the size of the systems scale with the ambition of the tasks they try to accomplish. For instance, distributing the application between several nodes to achieve redundancy and improve reliability of the system increases the challenge of keeping communication time constraints. Furthermore, corroborate that a system is indeed safe and fulfills the safety regulations of regulatory institutions is also a difficult endeavour if there is not enough care on the system design. Time Triggered (TT) architecture helps solving these problems, as it guarantees the temporal constraints of a hard Real Time (RT) system.

Even though TT scheduling algorithms are in some cases necessary for hard time RT systems due to the hard time requirement restraints, there are some challenges this strategy has to overcome in order to become popular in more general industrial applications, as (Gendy, 2009) suggests. One of the drawbacks of choosing a TT scheduler over an Event Triggered (ET) one is that the former are fragile at design time, which means that small changes in the timing of some task when extending or uploading the design might require substantial changes in the schedule, sometimes even redesign of the whole schedule from scratch. TT schedules are also NP-Hard problems, meaning that there are times in which some task sets and their constraints may even not be able to be scheduled in this way. Lastly, TT schedules require knowing in advance some of the task set properties, such as their worst case execution time (WCET). This is not always an easy measure to have, and when estimated wrongly could result on catastrophic domino effects in the system during run-time. Different efforts have been taken to reduce the impact of these inconveniences when choosing a TT schedule. For instance, the different algorithm proposals selected in this report try to find an optimal algorithm for systems with different specifications in an automated way, so the scheduling effort is minimized to a simple knowledge of the constraints and specifications of the task set.

Because TT schedule algorithms rigidity require very specific approaches depending on the system specifications, it seems interesting to see if there is any common ground among them. This report tries to look into some of these strategies and their implementations when any, as it is well known that deployment of optimal software in the theory usually becomes a challenge on the real life scenario. These are the research questions that are going to be solved in the following pages:

1. What different deterministic schedule strategies have been proposed?
2. What are the challenges of implementing deterministic schedules?

This report follows with a classification of different schedulers in section (2) to introduce some key ideas used later by the main articles. Three main sources have been chosen to be reviewed for this report, and will be described in this order: Section (3) Time-Triggered scheduling analysis for Real-Time applications on multi-core applications (Freier & Chen, 2014), section (4) An energy-efficient Time Triggered scheduling algorithm for mixed-criticality systems (Behera & Bhaduri, 2020) and section (5) Optimal static scheduling of Real-Time tasks on distributed Time Triggered network systems (Serna Oliver, Craciunas & Ecker, 2014). Then in section (6) some conclusions are extracted from the sources in order to answer the research questions and finally in section (7) a discussion is held about the work achieved in the project.

2 Scheduler classifications

As presented in (Kopetz, 2011), "*A hard real-time system must execute a set of concurrent real-time tasks in such a way that all time-critical tasks meet their specified deadlines*". In order to ensure that every task has enough computational resources to accomplish its goal and that the precedence relations are met, a schedule must be proposed to satisfy the time requirements of the system. The minimum amount of tasks containing all the information of the schedule is called hyperperiod, which repeats in time while the system is working.

There are different features that define different aspects of a Real-Time schedule, such as when is the order of the tasks decided or if the tasks in the schedule can be interrupted by other tasks with higher priority. Some classifications are presented by (Buttazzo, 2011) or (Kopetz, 2011). Online schedules are computed on run-time depending on the system current requirements and available resources, in contrast with offline schedules, that must be predefined prior execution of the application. Whereas online scheduling is more flexible offline scheduling is the only that can guarantee a deterministic and predictable behaviour. Regarding task interruption, a task that can be interrupted is called preemptive and is usually found on schedules determined by priorities, while non preemptive schedules present an uninterrupted task execution. In this work, because we focus on TT applications, the different algorithms presented are offline schedules which, depending on the context, are constituted by either preemptive or non preemptive tasks.

Scheduling is not limited to single core embedded systems applications. RT systems can be found in multi-core applications too, which brings a series of advantages and drawbacks regarding the search for an optimal schedule. Different approaches depending if the system studied is uni-core or multi-core are presented in (Voss, 2010). When discussing uni-processor strategies two main algorithms are proposed when the schedule presents preemptive tasks. If the priorities are fixed a Rate Monotonic (RM) algorithm could be employed, while if the priorities are chosen to be dynamic Earliest Deadline First (EDF) is a very common algorithm choice. Multi-processor schedules, can be designed to be global scheduled, where every processor knows the task that must be executed and chooses the highest priority task from the table queue, or partitioned, where each processor always receives the same set of tasks. A problem with this last strategy is that optimal allocation on processors is unknown.

3 Time-Triggered scheduling analysis for Real-Time applications on multi-core applications

This first article (Freier & Chen, 2014) presents a schedule for multi-core systems. As presented in section (2), communication between cores makes the schedule choices more difficult, as tasks and messages may require specific order (precedence) and time constraints. The strategy followed is a partitioned schedule, where it is already decided prior run-time which task is going to run at which moment in each processor during the hyperperiod.

Instead of modelling all the task dependencies at once, the authors make two task sets, one for communication and other for computations. They use the communication task set to model the dependencies among tasks. For the computation task set a Time Triggered Constant Phase (TTCP) schedule is chosen, which is defined by non preemptive tasks characterized by a particular constant time, called phase, prior the computation time from the moment the task is scheduled. One of the main issues when evaluating the safety and reliability of an embedded system is validating that the system is able to finish all of it's tasks in the Real-Time constraints required. As this article proposes, "*A TTCP scheduler on each core ensures time-predictable communication impacts, which severely reduce the worst-case analytical effort in the communication analysis.*"

The main problem of a TTCP scheduler is determining the phase times for each task. The authors developed and describe a heuristic algorithm, named LPF-LBF, which can calculate the time-slots in a fast and efficient way, as their results suggest when comparing it with other algorithms. The article does not present a real life deployment of the algorithm in an embedded system application, but their experiments, based on typical industrial applications, show that the algorithm reaches a core utilization above 90%. As in section (2) was already mentioned, no optimal allocation on multi-core processors has been proven, but the authors propose this result seems good when compared to other multi-core applications.

4 An energy-efficient Time Triggered scheduling algorithm for mixed-criticality systems

In article (Behera & Bhaduri, 2020), energy, even when being a non functional element, is optimized with a TT dynamic voltage and frequency scaling (DFVS) algorithm for uni-core mixed-criticality systems. During the first two decades of the twenty first century, as more TT schedules have been proposed for different applications, researches have made efforts to optimize this schedules for some non functional features in a system, such as energy. This is an important feat, as this means the research has gone further enough to think about how to improve certain non-essential aspects of the system while guaranteeing the temporal constraints of the requirements.

When designing an application it is possible to specify which parts of the system are indispensable during critical operation and which other subsystems can fail or stop without any serious consequences on the safety requirements specified (Kopetz, 2011). For example, in a research mission using an aerial vehicle, the subsystems focused in the research part of the system play a non critical part of the overall safety of the system, as an aerial vehicle must ensure some other critical measures during flight operation not to fall and provoke damages. That is why tasks involving flight operations would have a higher priority criticalitywise compared to research tasks.

The authors in this article propose a new TT algorithm (TT-Merge) with better results than EDF - Virtual Deadlines (EDF-VD), which is a widely used algorithm for mixed criticality systems. TT-Merge counts with two different schedules, low (LO) criticality (containing all of the tasks from the task set) and high (HI) criticality (with only the HI criticality tasks). The system stays at LO criticality mode until a task LO WCET is not met. Then the scenario toggles to HI criticality so the tasks starts following this other schedule. Energy efficiency is obtained by assigning the lowest frequency possible to each job in LO criticality scenario and reducing the idle time of the processor as much as possible. The authors of this article put his efforts on minimizing energy consumption only for LO criticality tasks, and is only valid for periodic tasks. The tasks are preemptive, but are scheduled off-line.

There are proposed task precedence and relations and discrete set of frequencies extensions, but the solution is not assured to be optimal for the latter. This report is interested in distributed systems with communication among nodes but, even if there is no network communication among nodes taken into account for this algorithm, it could probably be possible to include communication tasks and their dependencies with the computational tasks using the tasks interdependencies extension of the TT-Merge schedule. No real life deployment schedule is presented, just computer simulations to show their proposal works better than other alternatives for energy optimization in mixed-criticality systems, such as EDF-VD.

5 Optimal static scheduling of Real-Time tasks on distributed Time Triggered network systems

Article (Serna Oliver, Craciunas & Ecker, 2014) focuses on industrial applications with TTEthernet communication and proposes an optimal schedule algorithm for software to minimize latency. It supposes that communication schedule is already deployed in the industrial application due to tight temporal constraints of messages. Jitter in the system depends on communication transmission with TTEthernet and tasks computations and the article puts its effort on the optimization of the latest.

The authors use TT-RTS (a Real-Time operating system) and divide the set of tasks on TT tasks (the ones to be scheduled) and preemptive best effort (BE) tasks, which run in the background only when there is no TT task running. TT tasks are scheduled non preemptively and BE tasks are preempted by the TT tasks when necessary, while TT-RTS guarantees temporal isolation between BE and TT tasks. TT Network and TT-RTS operate on different time domains so they must be synchronized. When this is achieved, tasks consuming or producing messages are synchronized with the network schedule, which is necessary to guarantee temporal constraints. Because the algorithm proposed focuses on the software schedule and not the communication, even though the article proposes TTEthernet communication for their algorithm development and examples, it is probably extendable to other communication protocols such as CAN.

In this article a deployment experiment of the algorithm is presented with the documentation of two major implementation problems:

1. TT-RTS Overhead: context change of tasks by the Operating System (OS). The authors propose increasing the WCET of tasks when making the schedule, taking OS overhead into account. This overhead is also influenced by hardware optimization choices. Different choices are shown to compare the overhead in each case.

2. TT-RTS synchronization to TTEthernet. Because of communication jitter, consumer tasks may experience receiving a message late and producer tasks shall finish earlier to account for this jitter. When setting the constraints for the scheduler this must be taken into account, delaying the consumers message receiving window and reducing the deadlines of the tasks producing messages.

6 Conclusion

Different schedule strategies have been explored depending on the system specifications for each of the articles. For instance, in section (3), because the system was multi-core, it required a completely different strategy than the ones proposed for uni-core systems in sections (4) and (5). It is also interesting to see how different combinations of TT schedule properties such as choosing preemptive or non preemptive tasks are valid depending on the scenario. Moreover, depending on the designer priorities and requirement analysis different aspects of the system can be optimized when programming the schedule, such as core utilization, energy consumption or communication jitter. However, all of them share the effort of developing a, to some extent, general algorithm to minimize the effort of searching for optimal answers to the scheduling problem.

Most of the articles found do not present a real life deployment of its algorithm proposal, just computer simulations to verify their validity. The exception is (Serna Oliver, Craciunas & Ecker, 2014), with an industrial embedded application for software scheduling, where the communication schedule was already fixed using TTEthernet protocol. Two main problems when implementing the algorithm where reported: the overhead caused by the OS context switch when preempting BE tasks and the synchronization jitter between communication and computation schedule. Measures were taken for both cases resulting in a successful RT implementation.

7 Discussion

This report shows several TT schedule strategies for systems with various properties and requirements. Part of the focus of this work was finding specific deployments of this kind of strategies in real case applications but only one was found documenting this challenge. The work space of articles documenting TT schedules for RT applications seems to be as broad as the field it tries to explain, as each different application has its own characteristics requiring its own solution, so probably the first research question of this article has a scope too broad for the dimensions of this report. However, it has been seen that the collective effort is focused on trying to automatize the scheduling process in the most general way possible, searching for possible schedules, usually optimal with respect to some feature, given a certain task set defined by the system designer.

More work has still to be done in order to find articles documenting real life application deployments, but this first approach apparently signals towards the original idea that industrial applications usually prefer an ET schedule due to its flexibility, and so only a small number of TT scheduled applications seems to be documented in the industry sector.

Bibliography

Behera, L. & Bhaduri, P., (2020), An energy-efficient time-triggered scheduling algorithm for mixed-criticality systems, *Design Automation for Embedded Systems*, *24*, https://doi.org/10.1007/s10617-019-09232-3

Buttazzo, G.C., (2011), *Hard real-time computing systems: Predictable scheduling algorithms and applications* (3rd, Vol. 24), Springer Publishing Company, Incorporated.

Freier, M. & Chen, J.-J., (2014), Time triggered scheduling analysis for real-time applications on multicore platforms.

Gendy, A.K., (2009), Techniques for scheduling time-triggered resource-constrained embedded systems.

Kopetz, H., (2011), *Real-time systems: Design principles for distributed embedded applications* (2nd), Springer Publishing Company, Incorporated.

Serna Oliver, R., Craciunas, S. & Ecker, V., (2014), Optimal static scheduling of real-time tasks on distributed time-triggered networked systems, *19th IEEE International Conference on Emerging Technologies and Factory Automation, ETFA 2014*, https://doi.org/10.1109/ETFA.2014.7005128

Voss, S., (2010), Integrated task and message scheduling in time-triggered aeronautic systems.

First peer review: Jasper Verhoef, jg.verhoef@student.han.nl
Second peer review: Jordy Koole, jd.koole@student.han.nl

A Peer Review: Jasper Verhoef

Apart from some grammar mistakes noted, emphasis was done on splitting the introduction into two sections as it was too long and some of the things stated did not seem to belong there. Section (2) was previously part of the introduction.

An extra research question was also removed as it was redundant: *Are there deterministic schedule implementations in real life applications?*

Some concepts were advised to be explained in a little more detail, such as *hyperperiod* or *TTCP schedule*. The first is explained in section (2) and the other when it is introduced in section (3).

Furthermore, the abstract lacked the essence of the article, some opinions of the author of this report were not clear to be so and the discussion was not critical enough with the work done. Efforts have been made to improve the report with respect to these observations.

B Peer Review: Jordy Koole

The report version processed after Jasper's comment was further reviewed by Jordy Koole, who did not have any major improvement suggestions.