Report Number: WUCS-2007-17

2007-04-03

Proceedings Work-In-Progress Session of the 13th Real-Time and Embedded Technology and Applications Symposium

Chenyang Lu

The Work-In-Progress session of the 13th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS'07) presents papers describing contributions both to state of the art and state of the practice in the broad field of real-time and embedded systems. The 17 accepted papers were selected from 19 submissions. This proceedings is also available as Washington University in St. Louis Technical Report WUCSE-2007-17, at http://www.cse.seas.wustl.edu/Research/FileDownload.asp?733. Special thanks go to the General Chairs – Steve Goddard and Steve Liu and Program Chairs - Scott Brandt and Frank Mueller for their support and guidance.
... **Read complete abstract on page 2.**

Proceedings Work-In-Progress Session of the 13th Real-Time and Embedded Technology and Applications Symposium

Chenyang Lu

Complete Abstract:

The Work-In-Progress session of the 13th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS'07) presents papers describing contributions both to state of the art and state of the practice in the broad field of real-time and embedded systems. The 17 accepted papers were selected from 19 submissions. This proceedings is also available as Washington University in St. Louis Technical Report WUCSE-2007-17, at http://www.cse.seas.wustl.edu/Research/FileDownload.asp?733. Special thanks go to the General Chairs – Steve Goddard and Steve Liu and Program Chairs - Scott Brandt and Frank Mueller for their support and guidance.

Proceedings
Work-In-Progress Session of the 13th Real-Time and Embedded Technology and Applications Symposium

3-6 April, 2007
Bellevue, USA

Organized by the
IEEE Technical Committee on Real-Time Systems

Edited by Chenyang Lu

Introduction

The Work-In-Progress session of the 13th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS'07) presents papers describing contributions both to state of the art and state of the practice in the broad field of real-time and embedded systems. The 17 accepted papers were selected from 19 submissions. This proceedings is also available as Washington University in St. Louis Technical Report WUCSE-2007-17, at http://www.cse.seas.wustl.edu/Research/FileDownload.asp?733. Special thanks go to the General Chairs – Steve Goddard and Steve Liu and Program Chairs - Scott Brandt and Frank Mueller for their support and guidance.

Chenyang Lu
Work-In-Progress Chair
13th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS'07)

Table of Contents

Sharp threshold result for rate monotonic scheduling

Sathish Gopalakrishnan
University of British Columbia

Marco Caccamo
University of Illinois

Abstract

Scheduling policies for real-time systems exhibit threshold behavior that is related to the utilization of the task set they schedule, and in some cases this threshold is sharp. For the rate monotonic scheduling policy, we show that periodic workload with utilization less than a threshold U^*_{RM} can be scheduled almost surely and that all workload with utilization greater than U^*_{RM} is almost surely not schedulable. We study such sharp threshold behavior in the context of uniprocessor scheduling using static task priorities. The notion of a utilization threshold provides a simple schedulability test for most real-time applications. These results improve our understanding of scheduling policies and provide an interesting characterization of the typical behavior of policies. The threshold is sharp (small deviations around the threshold cause schedulability to appear or disappear) for most policies; this is a happy consequence that can be used to address the limitations of existing utilization-based tests for schedulability.

1. Introduction

Computing systems have become larger in scale and more pervasive in their applications. The constant interaction between embedded computing systems and the physical world requires a notion of predictable behavior from the deployed computing systems. Even in large-scale computing clusters and server farms there is a growing emphasis on providing service guarantees. This need for predictable operation can often be characterized by a need for timely completion of activities. Tasks can usually be associated with deadlines; systems need to ensure that the tasks meet their deadlines.

In a sense, the convergence of computation, communication and control, which is often seen in distributed embedded systems, has led to a renewed interest in understanding the conditions for a system to meet deadlines. Additionally, most tasks are recurring: they need to be performed repeatedly because of the constant interaction with the physical environment (or because of user demand). Such problems have been at the heart of real-time scheduling since the seminal work by Liu and Layland [11] on utilization bounds for schedulability using static and dynamic priority scheduling policies.

The fundamental contribution that Liu and Layland made was to show that for a specific scheduling policy ζ – they studied the Rate Monotone policy and the Earliest Deadline First policy – there exists a utilization bound U_ζ such that any task set with utilization $U < U_\zeta$ is definitely schedulable (all deadlines will be met). This has formed the basis for much work in real-time systems.

There are, however, some obvious limitations to Liu and Layland's result. The first drawback is that the utilization bound test is pessimistic: there are many task sets that may exceed the bound but are still schedulable. Second, for models when the relative deadline does not equal the period, additional tests are needed. Lastly, obtaining the utilization bound is difficult for many policies because such derivations involve identifying the worst-case task set (the task set with low utilization that is not schedulable) and this is non-trivial for certain policies.

In contrast with prior work on schedulability and predictability, we show that the rate monotonic scheduling policy has a utilization threshold U^*_{RM} such that any task set with utilization less than U^*_{RM} is almost surely schedulable and a task set with utilization greater than U^*_{RM} is almost surely not schedulable. Establishing the sharpness of utilization thresholds provides a better understanding of scheduling policies and removes most of the pessimism that is associated with traditional utilization bounds because of the implication that task sets with utilization greater than U^* are unlikely to be schedulable. These results are independent of the relationship between task periods and task deadlines.On the other hand, it is prudent to note that these results indicate that schedulability appears and disappears *almost surely*. For hard real-time systems, which cannot afford to miss any deadlines, this suggests that the threshold can be used as an initial estimate and as an aid to optimization; schedulability needs to be verified by an exact test at some step but the *almost sure* nature of the threshold allows us to be confident. For soft real-time systems, which can tolerate some deadline misses, our results provide a simple test and a tight performance guarantee.

As an example, consider rate monotonic scheduling with the Liu and Layland task model. We would like to show that when n, the number of tasks to schedule, is large, a task set of utilization less than about 0.80 utilization is almost surely schedulable and a task sets with greater utilization is almost surely unschedulable. This shows that the average performance of the rate monotonic policy is much better than the Liu and Layland worst-case utilization of 0.69 [11]. Our results concern

the behavior of RM scheduling in expectation. Note that the worst-case results for RM scheduling are obtained from task sets that have many tasks with the same utilization. The worst-case, however, requires low utilization from each task and a specific period ratio. The sharp threshold results consider the more general case when the periods are uniformly distributed.

In this article, our emphasis is on rate monotonic scheduling on a uniprocessor although some preliminary experiments lead us to believe that these results will hold for deadline monotonic scheduling, and for multiprocessor and distributed (multistage) systems as well.

Motivation The main reason for studying sharp thresholds is to ease resource provisioning for soft real-time systems, and, in some cases, simplify the offline optimization of hard real-time systems. The existence of sharp thresholds allows us to make efficient use of computing resources. Many mainstream operating systems (especially Linux) support simple fixed-priority scheduling and being able to identify a workload limit for such systems allows for simple admission control and resource management. Many applications have tasks with deadlines but are built to tolerate a few deadline misses. Multimedia applications have been traditional examples, but many emerging pervasive computing applications are of a similar nature. Timely response leads to high quality of service but occasional delays are not catastrophic. For these systems, being able to utilize resources better can lead to substantial cost savings that will allow these applications to achieve greater market penetration.

2. System and task models

We consider Liu and Layland task model [11] for uniprocessor scheduling.

Task model Each task τ_i is periodic with period P_i. Each instance of the task has an execution time requirement c_i on the processor and a *relative deadline* P_i. If a job of τ_i is released (ready for execution) at at time t then it is expected to finish execution by time $t+P_i$. Tasks are independent of each other.

The utilization of a periodic task set is $U := \sum_i \frac{c_i}{P_i}$.

Monotone scheduling policies In this article, we will mostly be concerned with the rate monotonic scheduling policy, which is a work-conserving (non-idling) policy. It is also useful to keep in mind a more general classification of policies: the class of monotone policies. Let us suppose that a scheduling policy successfully schedules a set of tasks $\Gamma = \{\tau_i\}$. We will call the policy a *monotone scheduling policy* if and only if: a) It can schedule any set $\Delta \subset \Gamma$ successfully; b) For any task $\tau_i \in \Gamma$, the policy can schedule all tasks successfully if c_i were to be reduced; c) For any task $\tau_i \in \Gamma$, the policy can schedule all jobs successfully if P_i were increased.

3. Utilization thresholds

Let S_n represent the set of all schedulable task sets with n tasks and let $\mu(U,S_n)$ represent the probability that a task set with utilization U is schedulable using the rate monotonic policy. This can also be stated in the following manner. Suppose Γ, a task set with n tasks, is drawn at random from the space of all possible task sets of utilization U. Then, $\mu(U,S_n)$ is the probability of the event "$\Gamma \in S_n$."

Definition 1 (Threshold) *U_n^* is said to be a threshold for S_n if for any U*

$$\lim_{n\to\infty} \mu(U,S_n) = \begin{cases} 0 & \text{if } U \gg U_n^*, \\ 1 & \text{if } U \ll U_n^*. \end{cases} \tag{1}$$

Note that $f \ll g$ means $f/g \to 0$.

Definition 2 (Sharp threshold) *A threshold is said to be sharp if there exists a U_n^* such that for every $\varepsilon > 0$ and any U*

$$\lim_{n\to\infty} \mu(U,S_n) = \begin{cases} 0 & \text{if } U > (1+\varepsilon)U_n^*, \\ 1 & \text{if } U < (1-\varepsilon)U_n^*. \end{cases} \tag{2}$$

ε, typically, is inversely proportional to n. Thresholds become sharper with increasing values of n. The interval of width 2ε over which the probability of finding a valid schedule drops from 1 to 0 is called the *threshold interval*. A threshold that is not sharp is a coarse threshold. Sharp thresholds represent phase transition phenomena because we can divide the task set space into two phases: one in which the property holds almost always and one in which it almost always does not hold.

We emphasize once more that, although the results are asymptotic, in practice a reasonable number of tasks suffices for observing sharp thresholds. When we think of $n \to \infty$, we do not conjure up task sets with 1000s of tasks; we are usually dealing with many 10s of tasks.

The main result of our work is that schedulability, with the rate monotonic scheduling policy, of periodic tasks has a sharp threshold.

By proving such a result we provide a platform for the average-case analysis of real-time scheduling problems and highlight the validity of using empirical utilization thresholds for managing resource allocation.

4. Analyzing typical-case schedulability

To show that scheduling problems of the type that we are interested in have a sharp threshold we leverage some excellent work carried out in the context of random graphs. The study of phase transitions can be traced back to the work of Erdös and Rényi on random graphs [5, 6]. A random graph is a graph with a fixed set of vertices and edges between two given nodes occur with some probability, p. Erdös and Rényi showed that as the parameter controlling the edge probability varies, the random graph system experiences a swift qualitative change. This transition is similar to observations

in the physical world. Akin to water freezing abruptly as its temperature drops below zero, the random graph changes rapidly from having many small components to a graph with a giant component that contains a constant proportion of vertices.

We use results that have been obtained by Friedgut and Bourgain [7] to prove the existence of a sharp utilization threshold for schedulability by mapping the real-time scheduling problem to a problem on a complete bipartite graph. We do not discuss the entire proof owing to space limitations. We will, instead, present some empirical evidence to illustrate the theoretical results.

5. Sharpness of utilization thresholds

When examining experimental data, it behooves us to recall that sharp threshold behavior is a property of very large task sets. For moderate size task sets, one can observe a threshold but it many not be as sharp as one would expect. (We present only a limited number of graphs for space considerations. Given the immense number of graphs that can be obtained, those shown here are intended as a visual cue to the theoretical machinery we have used.)

If Γ is a task set drawn at random, the sharp threshold result suggests that $E(\Gamma)$, the expected task set, can almost surely be scheduled for $U < U^*$ and almost surely not be scheduled for $U > U^*$.[1] Utilizations were generated using this approach for n tasks. Periods were then drawn uniformly at random from $[1, 10^5]$. Task set utilization was varied in steps of 0.1 and at each level we tested 10^4 task sets. The different numbers of tasks in a task set for the experiments were $8, 16, 32$ and 64. Notice (in Figure 1(a)) that schedulability drops rapidly when utilization is in the range $[0.8, 0.9]$. The width of the threshold interval is smaller for larger task sets. Within a rather short interval, we go from almost all task sets being schedulable to almost no task set being schedulable. This transition allows us to approximate the schedulbility test by using a utilization threshold close to 0.8.

There are multiple ways to generate task sets to test schedulability. Bini and Buttazzo [3] have studied different approaches to generating random task sets and have suggested methods with almost no bias. The goal of Bini and Buttazzo's work is to generate task sets uniformly at random from the space of all possible task sets that achieve utilization U. This is a slightly different approach from the typical case task set because it allows certain tasks to dominate the overall utilization. Dominance prevents us from observing sharp threshold behavior because it does not allow for *reasonable scaling* in the number of tasks. *Even with limited dominance, we can generate task sets uniformly at random and still see a sharp threshold.* For instance, we can bound the maximum possible utilization of an individual task at $1/3$ ($\max_i u_i \leq 1/3$) and obtain a sharp threshold (Figure 1(b)). This is not a significant restriction because this is typical with most modern processors. Increasing processor speeds have led to smaller per-task utilization. For these set of experiments, we retained the same sets of parameters and altered only the procedure for generating per-task utilization. We employed the UUNISORT procedure from the article by Bini and Buttazzo [3]. The threshold when task sets are generated uniformly across all possible task sets is still close to 0.8 but the width of the interval is not as sharp as in the earlier batch of experiments. There is a small contraction in the width of the interval with increasing number of tasks. Overall, we can still approximate schedulability for task sets reasonably well using threshold behavior.

The sharp utilization threshold is remarkable because it makes no assumptions about task periods and yet provides quite a precise estimate of schedulability. The general methodology for deriving utilization bounds for any scheduling policy involves identifying a task set that achieves low utilization and is yet unschedulable. It is not always easy to isolate the worst-case task set and determine its utilization. Sharp threshold results provide us with the support for empirical determination of the threshold. When the worst case is rare (a low probability event) we are not burdened with a low utilization bound.

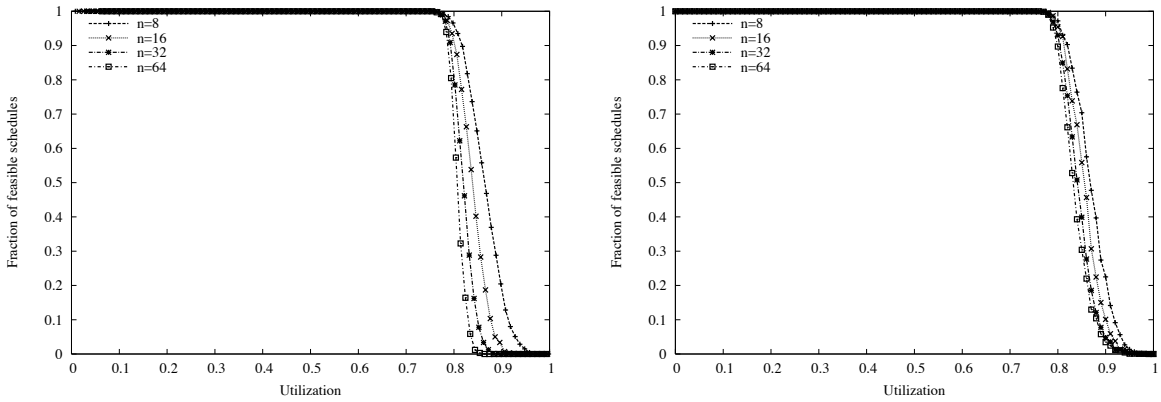
A possible concern is the asymptotic nature of the result. Sharp threshold behavior occurs when the number of tasks is large. We contend that this exactly the case for which existing real-time scheduling results are often inefficient (high complexity for analysis). As experiments reveal, a moderate number of tasks is sufficient for observing sharp thresholds. For small task sets, even exact tests may be performed very quickly. There is a dependency between the threshold and the number of tasks. It is easily possible to compute – offline – the threshold for different numbers of tasks and utilize the appropriate threshold.

The use of thresholds becomes extremely useful in the case of soft real-time systems and for performing fast exploration of design space in developing (near-)optimal systems. An example is radar dwell scheduling, explored by Gopalakrishnan et al. [9] and Ghosh et al. [8]. There are many task parameters that need to be tuned in a radar system to minimize tracking error subject to schedulability but the scheduling algorithms are hard to analyze; using thresholds for these problems simplifies the online optimization. Because performance is controlled at run-time, optimization routines cannot invoke exact tests that have high time complexity. Apart from online optimization, thresholds can be used as offline guidance measures to improve system designs.

6. Conclusions

Sharp thresholds are indicators of phase transitions. Phase transitions are common in physical systems. Freezing of ice and superconductivity are phenomena that have temperature as the critical parameter. Phase transitions have been identified in many combina-

[1] $E(\Gamma)$ is such that each task has almost the same utilization level. This is the configuration that generates the worst task sets for rate monotonic scheduling [3] and yet the thresholds are better than the Liu and Layland asymptotic results of 0.69.



(a) Expected task set method

(b) UUNISORT method ($\max_i u_i \leq \frac{1}{3}$)

Figure 1. Thresholds for rate monotonic scheduling

torial optimization problems, especially constraint satisfaction problems [4, 13, 10]. Phase transitions provide very interesting insight into the behavior of combinatorial optimization problems, of which scheduling is an instance, and mayhold the key to faster, near-optimal solutions.

The search for efficient tests for schedulability has been at the center of real-time systems research. We have generalized the use of utilization as a schedulability metric. By identifying the sharp threshold behavior of scheduling policies with respect to utilization, we provide a new test for schedulability. Schedulability tests using utilization thresholds are well-suited for soft real-time systems. For hard real-time systems these tests can be backed up by exact tests; thresholds can be used to perform initial filtering before using exact tests.

It appears that the sharp threshold results can also be extended to the case of aperiodic task systems. This will allow us to further the work of Abdelzaher, Sharma and Lu [1] for uniprocessor systems, and Abdelzaher et al. [2] for multistage systems.

Our approach to dealing with average or typical case behavior of scheduling policies makes interesting connections with results from percolation theory and random graphs. We hope to explore these links further to fully characterize the performance of scheduling policies.

References

[1] ABDELZAHER, T., SHARMA, V., AND LU, C. A utilization bound for aperiodic tasks and priority-driven scheduling. *IEEE Transactions on Computers 53*, 3 (Mar. 2004), 334–350.

[2] ABDELZAHER, T., THAKER, G., AND LARDIERI, P. A feasible region for meeting aperiodic end-to-end deadlines in resource pipelines. In *Proceedings of the IEEE International Conference on Distributed Computing Systems* (Mar. 2004).

[3] BINI, E., AND BUTTAZZO, G. C. Measuring the performance of schedulability tests. *Real-Time Systems 30*, 1-2 (May 2005), 129–154.

[4] CHEESEMAN, P., KANEFSKY, B., AND TAYLOR, W. M. Where the really hard problems are. In *Proceedings of the International Joint Conference on Artificial Intelligence* (1991), pp. 331–337.

[5] ERDÖS, P., AND RÉNYI, A. On random graphs I. *Publicationes Mathematicae Debrecen 6* (1959), 290–297.

[6] ERDÖS, P., AND RÉNYI, A. On the evolution of random graphs. *Publ. Math. Inst. Hungar. Acad. Sci. 5* (1960), 17–61.

[7] FRIEDGUT, E. Sharp thresholds for graph properties, and the *k*-SAT problem; with an appendix by Jean Bourgain. *Journal of the American Mathematical Society 12*, 4 (1999), 1017–1054.

[8] GHOSH, S., RAJKUMAR, R., HANSEN, J., AND LEHOCZKY, J. P. Integrated resource management and scheduling with multi-resource constraints. In *Proceedings of the IEEE Real-Time Systems Symposium* (Dec. 2004), pp. 12–22.

[9] GOPALAKRISHNAN, S., CACCAMO, M., SHIH, C.-S., LEE, C.-G., AND SHA, L. Finite horizon scheduling of radar dwells with online template construction. In *Proceedings of the IEEE Real-Time Systems Symposium* (Dec. 2004), pp. 23–33.

[10] KIRKPATRICK, S., AND SELMAN, B. Critical behavior in the satisfiability of random boolean expressions. *Science 264* (1994), 1297–1301.

[11] LIU, C. L., AND LAYLAND, J. W. Scheduling algorithms for multiprogramming in a hard real-time environment. *Journal of the ACM 20*, 1 (Jan. 1973), 46–61.

[12] LU, C., STANKOVIC, J. A., TAO, G., AND SON, S. H. Feedback control real-time scheduling: Framework, modeling and algorithms. *Real-Time Systems 23*, 1/2 (Jul./Sept. 2002), 85–126.

[13] MITCHELL, D., SELMAN, B., AND LEVESQUE, H. Hard and easy distributions of SAT problems. In *Proceedings of the National Conference on Artificial Intelligence (AAAI92)* (1992), pp. 459–465.

Real-Time, Dynamic Calculations of Polynomial Coefficients for Use in Telemetry

Timothy L. Allen
Lockheed Martin and University of Houston
Timothy.Allen.ctr@cisf.af.mil

Albert Mo Kim Cheng
University of Houston
cheng@cs.uh.edu

***Abstract*—This paper presents several new techniques for sending telemetry. These new techniques include dynamically altering the polynomial coefficients used to translate and compress the engineering values of the measurands in the telemetry stream. Using these new techniques, it is now possible to increase the precision of the data stream, thus reducing error. It also enables us to broaden the range of visible values, providing meaningful data streams for measurands that were previously out of range. Further, general algorithms are provided so that it is no longer necessary to dedicate an engineer's time to generate the polynomial coefficients of every measurand onboard the airborne system before deployment based on approximations and assumptions; rather these coefficients can be generated automatically in real-time, after a sufficient sample set of precise data has been obtained directly from the sensors. Additionally, telemetric systems that support 2-way telemetry can be programmed to recalculate these coefficients upon command.**

***Index Terms*—Telemetry, Polynomials, Real-Time Systems, Optimization Methods**

I. Introduction

TELEMETRIC systems are becoming ubiquitous[1]. While some may argue that Moore's law no longer applies, it is evident that components, in general, are getting better and less expensive. Not only does this provide opportunities to introduce telemetric systems where they've never been, but it also enables us to customize the telemetry in real-time.

Telemetric systems are usually comprised of two separate subsystems. In the industry, these subsystems are generally called an airborne system and a ground system. These systems are designated as such only by the roles they play in the telemetric system; the physical location of each system could be anywhere. Data that is sent from the airborne system to the ground system is called telemetry. Two-way telemetry occurs when data is also sent from the ground system to the airborne system. This data is called tele-commands, or just commands. A few examples of airborne systems are spacecrafts, missiles, RPVs, (remotely piloted vehicles), the Formula One race car, oil rigs, and heart monitors.

The two most common techniques used in transmitting telemetry are table lookups and polynomial conversions. The airborne system collects sensor data. To reduce bandwidth, the sensor data is converted using one of these two techniques, whose conversion data (the whole table and the polynomial coefficients respectively) are usually hard coded in the ROM (Read Only Memory) of the airborne system. When the transmitted data is received by the ground system, it is converted back into its original units. This paper introduces new dynamic telemetry transmission techniques. In particular, the paper describes the methodology of generating the polynomial coefficients for these conversions in real-time and transmitting these coefficients so that the airborne and ground systems are in sync, and discusses the benefits and drawbacks.

II. Telemetry Overview

Each specific attribute of an airborne system that is measured is called a measurand. Each sensor populates one measurand's value. For example, a system that includes 11 temperature sensors on a wing will have 11 different measurands, which could be called port_wing_temp_1 through port_wing_temp_11, each populating the value associated with it. These values are sent to the encoder, which contains a commutator or multiplexer, and an output formatter.

The output formatter converts the engineering unit into a counts value, which is more suited for data transmission. It does this by looking up the type of conversion required for the specific measurand, usually a polynomial conversion or a table lookup. It then takes the data represented in engineering units and performs the required conversion to package the data into the limited space for it in the output telemetry stream.

The true engineering value in a specific unit is translated into a counts value, or counts, for short. After the transmission has occurred, the ground unit must be equipped to translate the counts back into its natural engineering unit.

For a simple example, if a sensor is intended to consistently measure temperatures ranging from 118.312º Fahrenheit to 118.319º Fahrenheit, and the digital representation can only provide precision to the thousandths, much bandwidth could be saved by only transmitting 3 bits for this signal (0 corresponding to 118.312º, 1 corresponding to 118.313º, etc.).

Using a polynomial conversion, the engineering value is run through an equation to produce the counts value (the value transmitted). For this example, the following equation could be used:

$$y = (x - 118.312) * 1000 \tag{1}$$

When the ground system receives the signal and in order to obtain the engineering value, the ground system would then

[1] A few examples include vending machines that can report their inventory wirelessly, vehicles that can transmit 2 way vehicle telemetry through OnStar, and Verizon's introduction of "Chaperone" which reports the location of your child's cell phone.

* Work supported in part by Lockheed Martin and a 2007 GEAR Grant.

run the counts value it obtained (of which only 0-7 are legal values in the example) through an equation which is (approximately) the inverse of the equation used in the airborne system.

$$y = x/1000 + 118.312 \quad (2)$$

The airborne system usually has the coefficients of the polynomials built into the read only portion of the hardware. These values are painstakingly engineered a long time before deployment or launch of the ground system, usually during the design phase of the airborne system. For every sensor on every airborne system, a decision needs to be made and a set of coefficients found that best fits its projected output.

The output, however, is only a prediction until the airborne system reaches its intended environment. In making these predictions, there exists an element of chance and the possibility of erroneous calculations. Once deployed, the airborne system cannot reallocate the number of bits used in transmitting specific data or adjust the polynomial coefficients or curves.

Given the number of sensors in complicated systems, the chances of making a mistake are high, the time to engineer the coefficients is expensive, and once the airborne system has been deployed, there is usually nothing that can be done to correct an error. Additionally, a proven technique for obtaining these coefficients in an automated manner could be reused across many sensors on many types of real-time systems.

The coefficients and polynomials on the airborne system that convert engineering units to counts are called reverse coefficients and reverse polynomials, respectively. Conversely, the coefficients and polynomials on the ground system that convert counts back into engineering units are called forward coefficients and forward polynomials, respectively.

When a polynomial method is used, each measurand is associated with one or more polynomials. Sometimes, in order to achieve the desired accuracy with limited order polynomials, more than one polynomial is associated with each measurand. Each polynomial is defined and only considered valid over a specific range for each measurand. To maintain a deterministic system, the ranges should never overlap. Once this conversion has occurred, the output formatter takes the counts value and packages it into the appropriate telemetry page.

In large telemetric systems, the airborne system may use some form of lossless data compression [2]and encryption before the telemetry stream is transmitted to the ground.

III. Obtaining Dynamic Polynomial Coefficients

A. Calculating the Reverse Polynomial Coefficients on the Airborne System

It is quite possible for the airborne system to gather data from a particular sensor, and using that data, calculate new coefficients to be used in conjunction with that sensor. Of course, as the price of storage decreases, as storage capacities increase, and as CPU speeds continue to increase, this option is continually becoming more feasible. Attention would need to be given to which algorithms are used for the computations, and that they converge, among many other requirements.

After enough data is collected and the coefficients are calculated, they would need to be transmitted to the ground system. Despite the cost in bandwidth, the coefficients will be transmitted without a lossy compression. Because the transition to the new coefficients isn't as time critical as other telemetry, the coefficients could be sent in a subframe of a page controlled by an internal variable set by the ground. When all of the coefficients are obtained by the ground, the ground system must transmit an acknowledgement of reception so that it will be clear when the new coefficients are going to be used. Since large samples of data facilitate calculating more efficient polynomial coefficients and transmitting the new coefficients is costly, these updates would not be occurring too frequently. A toggle bit packaged in the page could be altered to signify that the new coefficients are being used. This is better than sending a one-time acknowledgement which might get lost in transit. If the ground system doesn't see the toggle bit updated, it would retransmit the acknowledgement. Using the new reverse coefficients, the ground system would then need to generate corresponding forward polynomials to decode the data.

B. Calculating the Coefficients on the Ground System

As data is received from the airborne system using the coefficients that were originally deployed, the ground system, which generally has significantly more resources than the airborne system and is already storing all of the transmitted data streams, could generate new polynomial coefficients. When it is determined that a new set of coefficients is to be used on the airborne system, the ground system would transmit a command to the airborne system with the coefficients to be used for the specific measurand, or group of measurands. Then the airborne system would need to transmit an acknowledgement that a transition to the new coefficients has taken place. A toggle bit packaged in the page could be altered to signify that the new coefficients are being used.

Although the ground system generally has ample resources for such tasks, one drawback of this method is that the ground system only has access to the counts values that were

[2] Although the method of creating polynomial coefficients in real-time and converting engineering units to counts using actual obtained data may seem like an adaptive compression algorithm, and this process, whether the coefficients were computed in real-time or not, can reduce the bandwidth required to transmit the data, the improvements suggested by the authors in this paper are not made with the intent of reducing bandwidth. It is true that it would be possible in some cases to get the same accuracy and shave off a bit here or there, however, telemetry frames, pages and some encryption algorithms usually require data in complete chunks of 2^n bits. In fact, it is quite common to have extra bits in a frame that are padded with data to fill 2^n bits.

transmitted. Of course, the engineering values can be computed from the counts, but the new engineering units are not 100% accurate. New coefficients created on the ground have the limitation of only being created from data that has gone through the transmission process. Because of machine error and limited digits of precision on both systems, this process produces engineering values that are generally less accurate than the original engineering values on the airborne system.

C. Calculating the Coefficients on both the Ground System and on the Airborne System

Of course, whenever the coefficients need to be transmitted, this takes away valuable bandwidth. An alternative to transmitting the values is to recalculate them on both the airborne system and on the ground system. Both systems could be independently triggered to use the new coefficients created using the exact same algorithms and inputs. Though the airborne system has access to more accurate data than the ground system, it would need to use the data sent to the ground system to be in synch. This would mean that the airborne system would only need to store counts values, which are much smaller than engineering values, but it would also require forward polynomial coefficients for each measurand.

IV. Advantages and Disadvantages

Depending on the method of obtaining the dynamic polynomial coefficients, the advantages and disadvantages can vary. Common to all of the methods mentioned is the ability to replace erroneous or less accurate coefficients with more accurate ones. This could mean a lot less time spent up front in engineering. Since the airborne system's intended environment is often difficult to predict, such calibrations of the conversion polynomials could result in obtaining information more precise than what could be obtained otherwise. Flexibility exists to narrowly focus on a specific point of interest, obtaining significant precision, possibly disregarding parts of the original range. Conversely, we could widen the function's range and, while still transmitting the same number of bits, start receiving values that would have shown as out of range before.

Of course, if caution is not taken in recalculating the coefficients, valuable resources such as power, CPU usage, memory, and bandwidth could be used without benefit. However, if this technique is done prudently, one could obtain more accurate engineering values than could ever have been obtained otherwise. Additionally, incorrect values could be fixed; ranges could be broadened or narrowed, providing the ability to zoom-in and obtain more precision. A possible extension to this could be to alter the implementation of a minor frame to dynamically adjust the bandwidth dedicated to particular sensors. If, for example, a hardware failure were to occur disabling a sensor, or whether we simply decide we don't care about certain readings as much as others, we could reallocate our available telemetry stream in real-time.

V. Results

The authors have written a Java program that implements some of the ideas mentioned in this paper. First, we generated a simulated signal which produced values based on a function. The function we used was as follows:

$$y = 200 * \sin(x/100) + 200 \tag{3}$$

We chose a sinusoid because it was simple and like the values being obtained from most sensors, it doesn't diverge to positive or negative infinity. Although our program allows real-time user input to alter the values produced by the function, one possible upgrade is to use an actual data stream from an airborne system as input. The signal is spawned in its own thread, changing its value with time.

Then we created a sensor. The sensor periodically queries the signal, and stores and sorts the most recent unique signals obtained.

Finally, we created two different transmitters. The first is an implementation of the way telemetry is normally processed. The second starts off like the first in transmitting data with a built-in polynomial. However, after a certain number of engineering values are collected, a new set of polynomial coefficients are calculated. We calculated them using our own Java implementations of the Least Squares method and Gaussian Elimination.

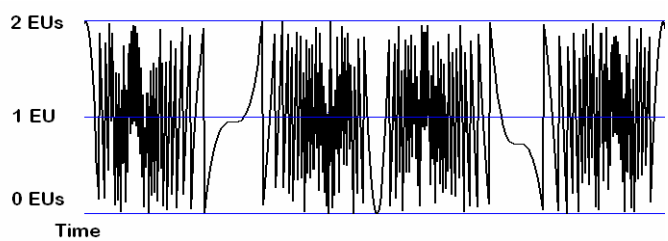


Figure 1: Error before recalculating coefficients

The graph above represents the error in engineering units (EUs) generated by the normal method of transmitting telemetry. Since the output of equation 3 ranges from 0 to 400, we translated the output to an 8 bit count value using the following simple equation:

$$y = \frac{1}{2}x \tag{4}$$

The output of the equation was truncated to an integer, transmitted, and then converted back to engineering units with using equation 4's inverse.

The above graph was obtained by taking the absolute value of the difference of the original engineering value and the engineering value obtained from the transmission. The average error over the first 255 points was 1.10376, and the average error over the first 1000 points was 1.01086 engineering units.

The second transmitter transmitted the first 255 points exactly as the first, but at that point recalculated the

polynomial coefficients. The time of recalculation is indicated with a vertical red line on the following graph.

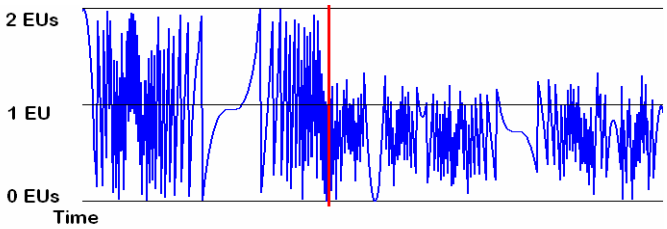


Figure 2: Error using recalculation method after 255 points

Of course, the average error for the first 255 points was identical, but the average error for the following 1000 points was only 0.5678. Additionally, we incorporated the limitation of truncating all of the values to 32 bits for both test cases in performing the calculations to create reverse polynomial coefficients, since this is a common limitation on real-time systems in the industry. In this test, we also limited this program to generating 1st order polynomials. This required 256 kB of memory and 375 ms. When the program was allowed to calculate polynomials up to the 13th degree, it still only used 355 kB of memory and finished in 578 ms, well within a 1 s deadline. The longer version of this paper will show higher order polynomials, and test cases where in order for the dynamically generated reverse polynomial's error to be within a certain error, the algorithm used multiple reverse polynomials for a single forward polynomial.

We've found that the higher order polynomials produced by the program are likely to produce even more favorable results, but that this is not always the case. When calculating forward polynomials on the ground to match the new reverse polynomials, we kept standard deviation error metrics of the errors of hundreds of with evenly distributed points. When a lower order polynomial produced less error, we used it instead.

In the example graphed, the new reverse polynomial generated was

$$y = 1.4371428x \quad - 324.12164 \tag{6}$$

And the new forward polynomial was

$$y = 0.69582504 \ \ x + 225.53197 \tag{7}$$

VI. Future Work

We imagine the telemetry of the future could be similar to viewing a movie where you have the ability to zoom in or out, and alter the resolution depending on your interest. Future work we would like to see includes using the idea of dynamically updateable polynomial coefficients in actual real-time systems that contain airborne systems (possibly satellite systems or manned space craft), that can handle the extra overhead processing and where the extra detail obtained is important.

VII. Conclusion

Dynamically altering polynomial coefficients for use in telemetry can be a powerful, yet dangerous tool. If done haphazardly, valuable resources could be wasted with little or no benefit, including CPU time, power, bandwidth, and memory. This could put a real strain on timing constraints, and affect adversely many parts of the airborne system, and our view of every part of it as seen through the telemetry stream.

However, problems with original coefficients could be fixed, and code could be reused, thus decreasing delivery time of large systems that are often behind schedule and over budget. Moreover, this ability could provide ranges of data, or data precision that simply would not have been possible otherwise. Additionally, the structure of the telemetry minor frames or pages could be altered to provide more bandwidth to measurands deemed important at any time.

References

[1] L3 Communications, Telemetry West. (2006, January). Telemetry Tutorial. [Online]. *21(3).* Available: http://www.tw.l-3com.com/tutorial/preface.html

[2] Telemetry Available: **http://en.wikipedia.org/wiki/Telemetry**

[3] Advanced Extremely High Frequency (AEHF) Available: **http://www.lockheedmartin.com/wms/findPage.do?dsp=fec&ci=11456&rsbci=0&fti=126&ti=0&sc=400**

[4] Flight Test Data Cycle Map Optimization, ITEA Instrumentation Workshop 28 March 2000, Available: **http://www.unisa.edu.au/seec/pubs/00papers/ITEA%20Workshop%20USA,%20Panton%20and%20Cook%20paper2000_24.pdf**

[5] Global Precipitation Measurement Available: **http://gpm.gsfc.nasa.gov/glossary.html**

[6] Verification of the Redundancy Management System for Space Launch Vehicle: A Case Study, Oleg Sokolsky, Mohamed Younisy, Insup Lee, Hee Hwan Kwak, and Jeff Zhouy, Available: http://www.ece.mtu.edu/faculty/rmkieckh/cla/5755/X33-rms-sok98.pdf

Detection of Malicious Nodes by Immediate Parents (DoMNiP) in Wireless Sensor Networks

Albert Mo Kim Cheng
Department of Computer Science
University of Houston
cheng@cs.uh.edu

Jawad Rasheed
Department of Computer Science
University of Houston
r4rasheed@yahoo.com

I. Abstract:

Wireless Sensor Networks (WSNs) are becoming popular day by day. These networks consist of large number of sensor devices, which are called nodes. In a WSN, the base station is considered to be the most powerful node in the network and does most of the computations on the data received from the sensor nodes. A base station can also be considered as the bridge between two or more WSNs. Sensor nodes are considered to be very weak as compared to the base station in terms of memory size and computational power. These sensor nodes operate on very low power supply; their life and the functionality depend upon the low energy consumption.

WSNs are being used in many applications for different purposes such as surveillance and safety-critical missions. These networks are also being deployed in environments where human presence is not practical; this makes these networks very crucial. In such environments where human presence is not possible, security is the main issue. In security-critical missions, these networks require a secure protocol for message broadcast. This makes it possible to keep these networks functioning properly for their sole intended purpose at all times.

II. Introduction:

When the network is under attack/intrusion, sensor nodes become compromised and then these compromised nodes along with the malicious nodes try to compromise their neighboring nodes. In terms of security, this behavior of sensor nodes is not acceptable. Many protocols and algorithms have been proposed as a solution for this problem. One of the most widely and commonly used technique is called **blind flooding**. The blind flooding technique fulfills the security requirements well but it does not satisfy real-time constraints. The other major disadvantage of this technique is that, instead of reducing the workload on the base station and the transmission load on the nodes, it does the opposite.

III. DENIAL of MESSAGE ATTACK (DoM):

When sensor nodes in the network are deprived of receiving broadcast messages by malicious nodes from the base station, the attack is called denial of message attack. In order to detect DoM in the network, base station broadcasts a special message and requests an authenticated acknowledgement message from each and every sensor node in the network. In response, all the sensor nodes send acknowledgement messages to the base station except the compromised nodes. The sending of these acknowledgement messages to the base station is called blind blooding. Since the network can be spread over several miles and the acknowledgement messages arrive at the base station at different times, the base station keeps polling for newly arrived acknowledgement messages. This way, base station wastes lots of its computational power just for processing these acknowledgement messages. On the other hand, sensor nodes cannot directly communicate with the base station, they pass the acknowledgement message to their neighboring nodes. Finally these messages arrive at the base station; as a result of this message propagation to the base station through neighboring nodes, lots of bandwidth of wireless link is wasted.

IV. DETECTION of MALICIOUS NODES by immediate PARENTS (DoMNiP):

Here we propose a new technique called "DoMNiP which can be implemented in wireless sensor networks. The main motivations behind designing DoMNiP are as follows

- ✓ To reduce the workload on the base station and network meeting all real time constraints
- ✓ To detect malicious (compromised) nodes in the network
- ✓ To conserve energy at sensor nodes in the network

The DoMNiP is a very simple technique and easy to implement. It can work with any kind of tree structured wireless network. Whenever the base station scans the network, DoMNiP reduces the number of acknowledgements sent back to the base from each node. It does this by processing acknowledgements at immediate parents. The backbone of this technique is based on the algorithm which is used to generate acknowledgement messages (format of this messages is shown later in the paper) for the sensor nodes in the wireless sensor networks. The algorithm is very simple and is controlled by the base station. Basically, the algorithm assigns a κέγ to each sensor node such that the κέγ is generated following all the constraints imposed by the base station. The same κέγ can be assigned to more than one sensor nodes in the network.

Following assumptions are made for the DoMNiP model in this paper:

- ✓ The structure of the wireless sensor network is tree like.
- ✓ There is only one base station in the network.
- ✓ Base station is not under attack.
- ✓ Each node in the WSN has 0 to n child nodes
- ✓ Each node has only one path to the base station.
- ✓ Depth of the network is directly proportional to the number of nodes in the network
- ✓ There is no packet lost in the network
- ✓ The distance between the parent node and child node is same for all the nodes

V. DoMNiP ALGORITHM:

DoMNiP algorithm is not too different from any other intrusion detection algorithms. We have assumed that

there is no packet lost in the network and every node in the WSN receives broadcast messages from the base station. The only reason when the sensor nodes do not receive broadcast messages is just because of the DoM attack.

When the base station scans the network for the detection of DoM attack by broadcasting special message, a message is sent over the network which requires every node in the network to send its acknowledgement message back to the base station. In blind flooding model, every node gets acknowledged by the base station. This model has an extreme overhead since every sensor node, except the non-leaf nodes, has to pass its acknowledgement message to the base station. In DoMNiP, sensor nodes get acknowledged by the immediate parents or by the base station while reducing the workload on the network in terms of message propagation to the base station.

The number of sensor nodes and the level at which these acknowledgement messages get acknowledged heavily depends upon the mechanism by which κέγs are assigned to the nodes (unique or duplicate).The base station keeps all the assigned κέγs in its database.

VI. DoMNiP κέγ GENERATION:

DoMNiP κέγ generation algorithm is very flexible and dynamic. In addition to its flexibility and dynamic behavior, it can be modified according to the security requirements of the network and how fast the base station needs to conclude if there is any attack.

Basically, base station controls the κέγ generation mechanism, it generates and assigns κέγ to the sensor nodes in such a manner that the parent node and the child node never gets the same κέγ. κέγ generation mechanism also assures that the two siblings never get the same κέγ. Based upon the security requirements, base station assigns the same κέγ to the sensor nodes in the path after some depth difference. This depth difference, used for assigning the same κέγ to more than one sensor nodes, is actually used as the level of security required by the WSN.

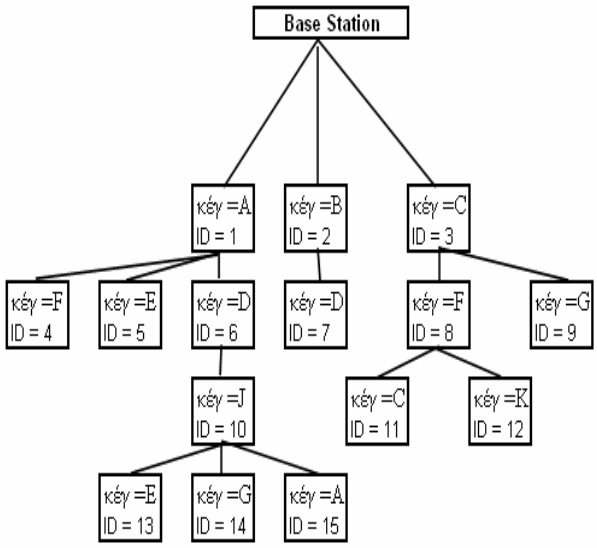


Figure 2

The κέγ generation algorithm is as follow

```
κέγ generate_new_key (κέγ  parent_node_κέγ,  κέγ
siblings_ κέγ[]  )
{
    κέγ new_ κέγ;
    bool =  FALSE

    while (bool = = FALSE)
    {
        - Get parent_node_κέγ
        - Create new _κέγ  from parent_node_κέγ
        - if (new _κέγ ! = parent_node_κέγ  & new_κέγ!=
          siblings_κέγ[i])
         {
             bool=TRUE
         }
     }
         -return  new_κέγ
}
```

VIII. ACKNOWLEDGEMENT MESSAGE FORMAT:

The following is the format of the acknowledgement message. This message is generated by the sensor node for sending it to the base station, when requested.

AM	$(\text{κέγ})_C$	Message Length	$(ID)_C$
0/1	Key of the sending node (child node)	Integer value	ID of the child node in the tree (ID)c

The following algorithm is run by every node till AM field is false.

```
Process Acknowledgement ()
{
   if (AM is false & parent node != base station)
   {
   //Compare (κέγ)C with its own (κέγ)P  where
   //(κέγ)P is the parent key
       if( (κέγ)P == (κέγ)C  )
       {
          -Set AM=1;
          -Crop & modify acknowledgement message;
          -Pass the Cropped & modified  acknowledgement
           message it to its parent;
        }
    }
    else (parent node != base station)
    {
          -Pass the received acknowledgement
           message to the parent
     }
}
```

Until the AM field is false and parent node != base station, every node in the path matches its $(\text{κέγ})_p$ with the $(\text{κέγ})_c$ in the received acknowledgement message. If $(\text{κέγ})_p$ and $(\text{κέγ})_c$ matches hundred percent, then the received acknowledgement message is modified and passed it to the parent.

The following is the format of the modified acknowledgement message.

AM	Message Length	$(ID)_C$	$(ID)_P$
1	Integer value	ID of the child node in the tree (ID)c	ID of the parent node in the tree (ID)p

This message is generated by the node whose **$(\kappa\acute{\varepsilon}\gamma)_P$** matches with **$(\kappa\acute{\varepsilon}\gamma)_C$** in the original acknowledgement message. The modified acknowledgement message is much smaller than the original acknowledgement message because it doesn't have **$(\kappa\acute{\varepsilon}\gamma)_C$**

- ✓ $(ID)_C$ is the ID of the node which generated the acknowledgment message.
- ✓ $(ID)_P$ is the ID of the node which modified the original acknowledgement message.

These are the two necessary fields in the acknowledgement message because base station uses these two fields and extract the depth of the sensor nodes in the WSN which is only known by the base station. Then the base station calculates the number of traversal steps as follows

TSteps = Extract (ID_C) - Extract (ID_P)

The Extract (ID_X) function returns the depth of the node in the network. The TSteps value provides the crucial information for detecting attacks. The TSteps value is discussed in the next section.

IX. PRE-CALCULATED TSteps VALUE:

When a new κέγ is generated and assigned to the sensor node, the base station calculates the TSteps for the new node at that time. The new sensor node generates an acknowledgement message and sends it to the base station through its parent sensor node. The parent node then matches its $(\kappa\acute{\varepsilon}\gamma)_P$ with the child node $(\kappa\acute{\varepsilon}\gamma)_C$ in the acknowledgement message using the acknowledgement() function as described in the previous section. If at some level, the $(\kappa\acute{\varepsilon}\gamma)_C$ matches with $(\kappa\acute{\varepsilon}\gamma)_P$, then the base station calculates the TSteps value for the new sensor node by first extracting the information from the acknowledgement message and then using the formula as described in section VII. iii. This calculated TSteps value is called pre-calculated TSteps value which is kept only by the base station. If the $(\kappa\acute{\varepsilon}\gamma)_C$ doesn't matches with any $(\kappa\acute{\varepsilon}\gamma)_P$, then its TSteps value is stored as 0. None of the sensor nodes know their TSteps value that is why if the attacker modifies the κέγ, its TSteps value remain unchanged at the base station.

DATABASE AT BASE STATION			
IDx	κέγ	Depth of sensor node	TSteps Value
1	A	1	0
2	B	1	0
3	C	1	0
4	F	2	0
5	E	2	0
6	D	2	0
7	D	2	0
8	F	2	0
9	G	2	0
10	J	3	0
11	***C***	***3***	***2***
12	K	3	0
13	E	4	0
14	G	4	0
15	***A***	***4***	***3***

Figure 3

X. SECURITY FEATURES OF IDx:

If the attacker modifies the κέγ of any sensor node, then it is possible that the modified (κέγ)c in the acknowledgement message may match with some $(\kappa\acute{\varepsilon}\gamma)_P$. Even if $(\kappa\acute{\varepsilon}\gamma)_C$ matches $(\kappa\acute{\varepsilon}\gamma)_P$, the base station will detect the attack on the network by calculating TSteps value which is only known by the base station.

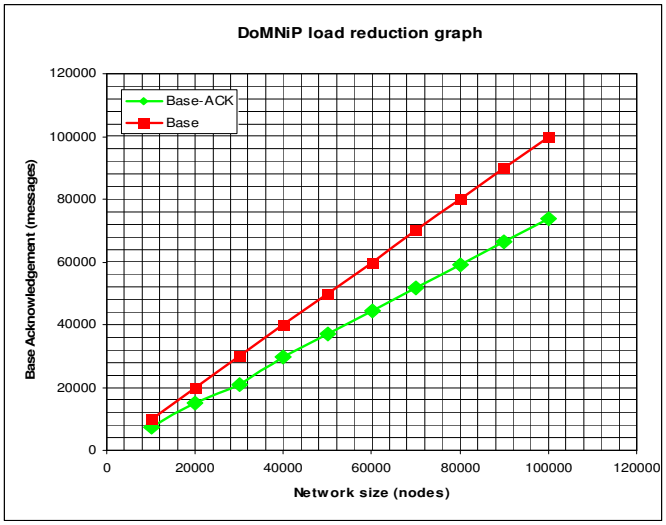
We assume that base station can never be compromised by the attacker. Whenever base station receives an acknowledgement message, it extract the (ID)c and (ID)p then calculates TSteps value. This TSteps value is compared with the pre-calculated TSteps value, if both the values are equal, then network in not under DoM attack.

XI. DoMNiP Simulation Results

For the simulation, we created different networks using different κέγs. Depth of the network was dependent on the number of nodes. Maximum numbers of nodes allowed in the network were 10^5. Readings were taken at an interval of 10^4 nodes in the network. After running several simulations using different κέγ, average values were tabulated and plotted on the graph. Simulation results are explained with the help of graph.

The red plot shows when no load reducing technique was used. All the acknowledgement messages were acknowledged by the base station. The red plot shows directly proportional results.

The green plot shows the results when DoMNiP is applied to the network. It showed that approximately 25% workload was reduced on the base station. 25% acknowledgement messages were acknowledged by the immediate parents whereas the station acknowledged only 75% when scanning the network for the DoM attack.



XII. CONCLUSION:

Simulation results show that DoMNiP reduces the workload on the base station without trading the security level required by the WSN.

From the graph, we can predict that whenever the network would be under a DoM attack, it would be detected earlier by DoMNiP as compared to blind flooding. It would be detected earlier because once the acknowledgement message is acknowledged by any parent sensor node on the path toward the base station, the length of the acknowledgement message is reduced because the (κέγ)c is dropped from the message. The smaller the message is, the faster it can propagate over the wireless link. When the message arrives at the base station, it first checks the AM field in the message. If the AM field is true then it does not have to do anything with the κέγ, it just have to calculate the TSteps value and compare it with the pre calculated TSteps value. This also shows that half of the workload on the base station is reduced.

We found three important relations between TSteps value, load on the wireless link, security of the WSN, and the time requirement.

- TSteps value has a directly proportional relation with the security.
- TSteps value has a directly proportionally relation with the load on the wireless link between the sensor nodes.
- TSteps value has a directly proportional relation with time.

A large TSteps value means that a large pool of κέγs is used meaning that κέγs are reassigned to new sensor node at greater depth difference. In this way, acknowledgement messages get acknowledgement by traversing through many parent nodes.

- This increases the security of the WSN because none the acknowledgement message will be acknowledged by the nearby neighbors (sensor nodes).
- This increases the load on the wireless link because the original acknowledgement message is bigger in size because it has the (κέγ)c field.
- This increases the time required to scan the WSN for the Dom attack.

DoMNiP is very simple and easy to implement. It is a software approach for reducing the workload on the base station and for detecting the DoM attack meeting the real-time constraint. DoMNiP is flexible and provides tunable features like pool of κέγs. Different levels of security can be achieved by simply not assigning the previously used κέγs to new sensor nodes.

Acknowledgment:

This work is supported in part by a 2006-2007 GEAR grant.

References:

[1] J.M. McCune, E.Shi, A. Perrig, M.K. Reiter. Detection of Denial-of-Message Attacks on Sensor Network Broadcasts. In Proc. 2005 IEEE Symp.on Security and Privacy.

[2] Nahla Ben Amor, Salem Benferhat, Zied Elouedi. Computer security (SEC): Naïve Bayes vs. decision trees in intrusion detection systems. In Proc. 2004 ACM Symp. on Applied computing.

[3] Paul Innella, Oba McMillan. An Introduction to Intrusion Detection Systems. 2001 Tetrad Digital Integrity, LLC. http://www.securityfocus.com/infocus/1520

[5] C. Karlof, D. Wagner. Secure Routing in Wireless Sensor Networks: Attacks and Countermeasures.

[6] Edith Cohen, Carsten Lund. Packet classification in large ISPs: design and evaluation of decision tree classifiers. In Proc. 2005 ACM SIGMETRICS international conference on Measurement and modeling of computer systems.

[7] James F. Kurose, Keith W. Ross, Computer Networking: A Top-Down Approach Featuring the Internet, 3rd ed. Pearson Education 2005

[8] J. Aslam, M. Cremonini, D. Kotz, D Rust. Using Mobile Agents for Analyzing Intrusion in Computer Networks. Department of Computer Science, Institute for Security Technology Studies, Dartmouth College , Hanover, NH 03755 http://www.ists.dartmouth.edu/library/uma0901.pdf

The Implementation of the Priority Ceiling Protocol in Ada-2005 Using a Shared Memory Programming Model

Albert Mo Kim Cheng and James Ras
Real-Time System Laboratory
Department of Computer Science
University of Houston
Houston, TX 77204, USA
cheng@cs.uh.edu, jras@cs.uh.edu

Abstract

Over the past few years, researchers have been developing the building blocks in Ada-2005 upon which it is possible to construct flexible real-time systems on parallel platforms. Now, under the semantics of Ada-2005, a comprehensive set of mechanisms are available that can deliver modern real-time scheduling theory to the system engineer. In this paper, we will show how Ada-2005 can be used to construct the Priority Ceiling Protocol in shared memory systems.

1. Introduction

In this work we turn our attention from uniprocessor to multiprocessor systems. Writing truly concurrent software is difficult. Scaling software to fully utilize hardware is one of the reasons why this is so. In a technological sense, concurrency can be defined as simultaneous execution within a computer system. Concurrency in shared-memory systems is exploited through multi-threading. Multi-threading means that more than one *thread* is running at the *same time*--one thread per processor core. In terms of hardware, this normally means more than one stream of instruction execution is taking place at the same time. Concurrency presents many challenges both in terms of creating concurrency, and utilizing concurrent systems.

One model of hardware concurrency that has recently become exceedingly popular is Symmetric Multiprocessors (SMP). It is a model in which more than one general purpose processor executes instructions. All of the processors operate in one shared memory space. Each processor shares all hardware resources, including memory, disks, and the system bus. Each processor sees the memory as directly addressable. When two or more processors access the same piece of data, we call that *sharing*. When data is shared, it has to be synchronized between all the other processors in the system. Each processor can view the results generated by another processor by simply looking at the appropriate section of memory. Sharing introduces serious overheads to write operations, especially if they are frequent and on different processors. Unfortunately, sharing is inherent in many common data structures, and more generally in many common real-time workloads.

In this paper, we consider resource synchronization issues in the context of real-time applications. To reasonably constrain the discussion, we will limit our attention to shared memory systems. Specifically, we will implement the Multiprocessor Priority Ceiling Protocol (MPCP) in Ada-2005. It should be noted that Ada is not just another programming language, but is a system for software engineering. Even if Ada is seen as just another programming language, it still reaches parts of the software development process that other languages do not attain.

2. The Problem

The problem addressed herein is motivated by the work of Chen and Tripathi [9]. They noted that the major problems associated with concurrent programming arise from task interaction. Rarely are tasks completely independent. Tasks execute essentially independently, but there are situations where it is necessary for two or more tasks to coordinate their work. Accordingly, in embedded systems design, it is important to distinguish between those activities that should be represented as tasks, and those that should be represented as protected shared resources. Additionally, it is critically important to indicate the nature of the interfaces between concurrent objects.

In priority-based pre-emptive scheduling, a task only waits for equal priority tasks that arrive earlier, or for higher priority tasks. The Priority Inheritance Protocol (PIP) [8,16] is a remedy for the uncontrolled priority inversion problem on uniprocessors. In PIP

*This work is supported in part by a 2006-2007 GEAR grant.

the need for synchronized access to shared resources can force a task to wait for lower priority tasks. This blocking is a form of priority inversion. If there were no sharing of resources, a task would not be blocked. There are potentially three types of blocking: direct blocking, push-through blocking, and ceiling blocking.

In multiple processor environments, there is an additional type of blocking. We shall refer to this new form of blocking as *remote blocking*. It is caused by remote tasks, regardless of their priorities. A task may have to wait for a task running on a remote processor to release a needed global shared resource, and for this reason, the concept of waiting needs to be generalized to include remote blocking.

3. Related work

The uniprocessor Priority Ceiling Protocol or PCP [17] is one of the best known policies for avoiding priority inversion and minimizing the blocking time in a single processor system. PCP has the following desirable properties: (**i**) a high priority task can be blocked by at most one lower priority task. Thus, there is no chained priority inversion, even if the tasks suspend themselves within critical sections. (**ii**) No deadlock is possible.

PCP is not known to be implemented on any of the popular operating systems. It is based on pre-emptive scheduling. PCP has the following rules: (**i**) A lower priority task that blocks a higher priority task *J* inherits the priority of *J*.

(**ii**) Semaphore Locking Conditions: A task *J* can only lock a semaphore *S* if: (**a**) the semaphore *S* is not locked, and (**b**) the priority of *J* is greater than the priority ceilings of all semaphores that are currently locked by tasks other than *J*.

The priority ceiling of a semaphore is defined as the highest priority of all tasks that may request to lock the semaphore at any time.

Of greatest importance to our paper is another work by Sha, Rajkumar, and Lehoczky [14] pertaining to MPCP and synchronizations in multiple processor systems. Sha *et al* noted that a high priority task must not be assigned the processor when a globally shared resource is locked by a local task. To be precise, we are going to enforce this rule only if a higher-priority remote task is waiting on the same locked global resource.

Sha *et al* assumed that tasks are statically bound to a processor. Once a task is allocated to a processor, each processor runs the same task. Thus, each task runs on its host core. No dynamic binding of tasks to processors was considered. In the work done by Chen and Tripathi [9], MPCP was extended to an EDF-based resource synchronization protocol. Unfortunately, Dhall and Liu [10] noted that the rate-monotonic algorithm, which performs well on uniprocessors, behaves poorly for multiple processor systems with dynamic binding. For example, consider the m-task set consisting of m-1 identical tasks, each with computational requirement of 2ε and period of 1, and one task with computation time of 1 and period of $1+\varepsilon$. If we schedule this set of tasks on m-1 processor system with dynamic binding, the highest priority m-1 tasks ready to run are scheduled on the m-1 processors. However, the first task with period $1+\varepsilon$ will miss its deadline. The utilization factor of this task set is $U=C/T=2\varepsilon(m-1)/(1+\varepsilon)$. As $\varepsilon\rightarrow 0$, $U\rightarrow 1$, thus a deadline can be missed with just 1/m-1 of the available processor cycles being utilized. In contrast, with static binding, the task with period $1+\varepsilon$ can be scheduled to a dedicated processor and the other task statically scheduled to a different processor. The task set becomes schedulable with just two processors.

Dynamic binding could give better performance if certain combinations of task bindings were not allowed to occur. In the example above, the worse-case scenario must be detected and avoided during run-time. Without such run-time evaluation, undesirable combinations could happen. However, note that in all of these methods, there are many context switches introducing significant overhead.

Our focus in this work is to construct MPCP under the semantics of Ada-2005. To the best of our knowledge, no prior research on MPCP and Ada-2005 has been presented in the past. Nevertheless, prior work on priority inheritance and Ada is of relevance to our work [7, 8, 9, 14, and 17].

4. Proposed approach

The goal of our approach is to maximize the overall symbiosis factor. This is a measure that indicates how well various tasks work in concert with one another. The basic objective is to discourage high-priority tasks from being co-scheduled when a globally shared resource is locked by a local task. Strictly, we are going to enforce this rule only if a higher-priority remote task is waiting on the same locked global resource. To further reduce priority inversion and missed deadlines, we will disallow global critical sections to overlap or nest with local critical sections. The objective reduces concurrency within task groups, but nonetheless, remote blocking is minimized, and a finer granularity is achieved allowing for a greater degree of concurrency overall. If we let $d_{j,i}$ denote the number of times that task J_i

locks a global semaphore. And let GLB_k be the worst-case access time of a global semaphore accessed by a lower priority task J_k. Then Task J_i can experience at most $d_{j,i} * GLB_k$ blocking delay.

We now define the *base priority ceiling* P_G to be $p(J_i) + 1$, where J_i denotes the highest priority task that may lock G_i. And note that $p(J_i) > p(J_{i+1})$. Then, using the same priority inheritance policy used in PCP, when J_h blocks on S_{Gi}, $J_{l<h}$'s gcs will execute at the priority of P_G+P_h, where P_h is J_h's priority. Then, our approach for implementing MPCP in Ada-2005 is a five-step process:

(i) When task *J* wants to access a local critical section, it uses PCP to see if it can lock the associated semaphore. Task *J* can get the lock, only if $P(J) > Ceiling(S^*_{locked})$, where S^*_{locked} denotes the semaphore with the highest priority ceiling of all local semaphores currently locked by tasks other than *J*. PCP is used to synchronize access to local resources.

(ii) If task *J* attempts to access a global critical section, it locks the associated semaphore *S* if no other task has already locked *S*, and no other global or local critical section is locked by *J*. Otherwise, it joins the priority-ordered queue associated with *S* using the original priority P(J).

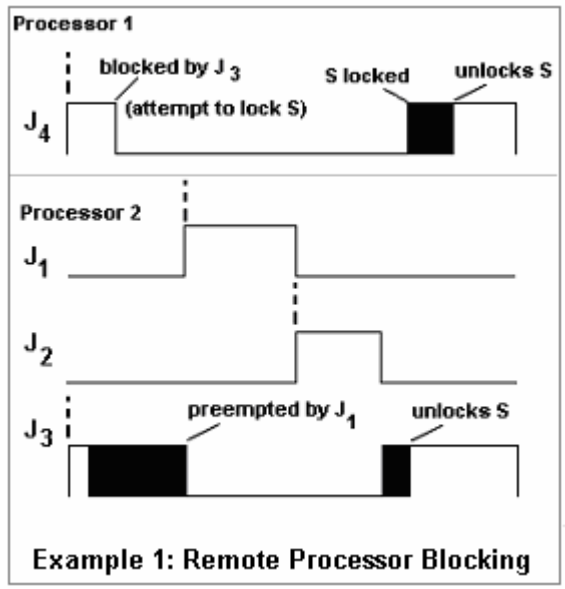
(iii) A task *J* locking a global semaphore S_g inherits the extended priority $P_{J,Sg}$, and reverts to its previous priority upon releasing S_g.

(iv) A task *J* can lock S_g and pre-empt another task J' within another global critical section guarded by S', if $P(J,S_g) > P(J',S'_g)$.

(v) Whenever a global semaphore is released, it will be given to the highest priority task waiting, if the associated queue is not empty.

It is important to state that the remote blocking duration is a function of the duration of the global critical sections of other tasks, and not a function of the duration of executing non-critical section code. In other words, it is forbidden that a task be blocked on one processor while another task executes on another processor outside a critical section. For example, suppose that task J_1, J_2, J_3 are bound to processor P_2, and that task J_0 is bound to processor P_1 (Figure 3). J_4 has the lowest priority. It is executing on processor P_1 and attempts to lock semaphore S. But *S* is currently locked by J_3 executing on processor P_2. However, J_1 now pre-empts J_3 on processor P_2, J_4 will be blocked until J_1 completes. Thus, the blocking time of J_4 will continue until arriving higher priority tasks on P_2 (such as J_1 and J_2) complete execution or suspend themselves. Since J_1 and J_2 are periodic tasks, the blocking duration of J_4 can be of indefinite length.

We should note that in the example above, even the imposition of priority inheritance does not force any changes in the event sequence, and the blocking of J_0 remains unchanged. Additionally, the direct use of PCP does not induce any changes in the execution. Unfortunately, the blocking duration of J_4 is a function of the entire execution times of tasks J_1 and J_2. This example shows that remote blocking is different from the method of uniprocessor blocking. Thus, under the semantics of Ada-2005, it is forbidden for a task to be blocked on one processor, while another task executes on another processor outside a globally shared resource.



Example 1: Remote Processor Blocking

Another important goal is to let a lower priority task wait for a higher priority task. For example, if two tasks are waiting for the release of a shared resource, the higher priority task will be allowed to access the resource first even if the lower priority task has been waiting for a longer time. Such situation happens when two tasks are executing on two different processors, and both need a shared resource currently locked by a task on a third processor. Under the earliest deadline scheduling algorithm, a lower priority task has a longer period and thus less schedulability loss. This objective is also reflected in our prioritized queues on the semaphores' requests.

5. Contribution

Our contributions to this research are three-fold. First, we will show how the Ada-2005 concurrent

programming model helps construct MPCP without the use of outdated models that could only communicate directly, via the rendezvous mechanism, thereby causing a number of task switches on each invocation. Ada now gives direct support to concurrency in the form of tasks representing threads of control, and protected objects that provide mutual exclusion and condition synchronization in a high-level, yet efficient, manner.

Second, our MPCP design will be equally applicable to multiprocessor systems as well as distributed and shared memory systems. Therefore, we do not assume the processors have synchronized or equivalent operations. Atomic test-and-set instructions are not a sufficient or necessary condition for the implementation of MPCP. The objective is not only to enforce mutual exclusion, but to also minimize blocking, especially remote blocking. Furthermore, if a task's active priority is not tracked - and dynamically raised and lowered when essential - then other tasks' deadlines are missed, and we cannot ensure schedulability.

Third, we propose a scheme for incorporating a semaphore-scheduler into the implementation of MPCP. Our choice of semaphore-scheduler is the best way to optimally grant requests of shared resources on multiprocessor systems. It may seem that the logical choice in Ada-2005 is to implement one semaphore by one synchronized protected object. However, before granting a lock on a semaphore, it is necessary to check the status of every other semaphore in the system to ensure that Locking Conditions (i-v) of MPCP (discussed in Section 4) are satisfied. Implementing one semaphore as one protected object will thus require intricate communication and synchronization between the semaphore protected objects. Accordingly, a single protected object is instead used to serve as the scheduler for granting all semaphores.

6. Conclusion and future works

The Multiprocessor Priority Ceiling Protocol is designed for use in tightly-coupled systems. It is an efficient synchronization method for preventing deadlocks and minimizing blocking time. The protocol can be supported well under the semantics of Ada-2005 in a system with multiple processors.

Our motivation of this study is to show that the performance of any method does not so much depend on the hardware but more on the implementation. A program is functionally correct if when running on any minimal execution environment it behaves according to the specification. We will report on the effectiveness of this in a future paper.

References

[1] Burns A. and Wellings A.J. Programming Execution-Time Servers in Ada-2005. RTSS 2006.

[2] Burns A. and Wellings A.J. Task attribute-based scheduling - extending Ada's support for scheduling. ACM ADA letters, 2003.

[3] Burns A. and Wellings A.J. Real-Time Systems and Programming Languages. 3rd. ed.: Addison-Wesley, 2001.

[4] Burns A. and Wellings A.J. Concurrency in Ada. Cambridge University Press (1995).

[5] Burns A. and Wellings A.J. Real-Time Systems and Programming Languages. 3rd. ed: Addison-Wesley. 2001.

[6] Cheng A. M. K. Real-Time Systems Scheduling. Analysis. and Verification. 2nd. ed.: Wiley & Sons 2002.

[7] Cheng A. M. K. and Ras J. The Implementation of the Priority Ceiling Protocol in Ada-2005. ACM Ada Letters, 4/ 2007.

[8] Aburas J. The Optimal Mutex Policy in Ada 95. ACM Ada Letters, 12/1995.

[9] Chen C.M. and Tripathi S. Multiprocessor priority ceiling based protocols. Rapport technique n CS-TR-3252. University of Maryland. 1994.

[10] Dhall S.K. and Liu C., On a Real-Time Scheduling problem, Operation Research, 1978.

[11] Leontyev & Anderson, Tardiness Bounds for EDF Scheduling on Dual-Speed Multicore Platforms. RTSS 2006.

[12] Liu C. and Layland J. Scheduling algorithms for multiprogramming in hard real-time environments. Journal of the ACM. Vol.20. No.1. 1973.

[13] Lehoczky J.P. and Ramos-Thuel S. An optimal algorithm for scheduling soft-periodic tasks fixed-priority preemptive systems. RTSS 1992.

[14] Sha L. Rajkumar R. and Lehoczky J. Time Synchronization Protocols for Multiprocessors. RTSS 1988.

[15] Sha L. Rajkumar R. and Lehoczky J. Generalized rate-monotonic scheduling theory: a framework for developing real-time systems. Proc. of the IEEE, Vol. 82, 1994.

[16] Sha. L. and Goodenough. J. Real-time scheduling theory and Ada. IEEE Trans. on Computers 4/1990.

[17] Sha L., Rajkumar R., and Lehoczky J., Priority Inheritance Protocols: An Approach to Real-Time Synchronization. IEEE Trans. on Computers, 9/1990.

INCENSE: Information-Centric Run-Time Support for Component-Based Embedded Real-Time Systems *

Andreas Hjertström, Dag Nyström, Mikael Åkerholm, and Mikael Nolin
Mälardalen Real-Time Research Centre, Västerås, Sweden
{andreas.hjertstrom, dag.nystrom, mikael.akerholm, mikael.nolin}@mdh.se

Abstract

In this paper we present a technique to allow the use of real-time database management together with component-based software development, to achieve an information centric run-time platform for the development of embedded real-time systems. The technique allows components to benefit from the advantages of a real-time database management system while still retaining desirable component properties, such as isolation and a high level of reusability. We propose that a database is integrated in the component framework, and introduce the concept of database proxies to decouple components from the database schema. The resulting system fully benefits from the advantages of component-based software development, such as reusability, all component interaction through interfaces, etc, combined with the advantages of a real-time database management system, i.e., system openness, controlled data access, and dynamic query language capabilities.

1. Introduction

Today's vehicle systems have an increasing number of computer nodes, called Electronic Control Units (ECUs), often developed by different hardware vendors, controlling engine, brakes, gearbox etc. The cost for development of electronics in high-end vehicles have increased to more than 23% of the total manufacturing cost [4]. Including subsystems a modern automotive system can contain over 70 ECUs communicating on different networks and exchanging up to 2500 signals [6]. The continuous increase of ECUs and exchanging of signals, leads to an growing amount of data that needs to be managed. A significant amount of the tasks using this data are critical hard real-time transactions, often operating at high frequencies and updated periodically. Furthermore, current trends also show an increase of tasks running non-critical, soft transactions in the system at lower frequency. These transactions, often read transactions, use the same data as hard critical tasks for logging or to present statistical information about the current state of the vehicle to the user.

To handle the increasing complexity in these systems, new approaches and design paradigms to reduce complexity are needed, since current techniques (internal data structures) are becoming increasingly insufficient. Two upcoming approaches to reduce complexity are Component-Based Software Engineering (CBSE) and DataBase Management Systems (DBMS). Real-Time Database (RTDB) [10] and RTDBMS (Real-Time Database Management System) are upcoming technologies both within research society and in industry [5] to help developers solve information management problems regarding synchronization, deadlock and persistency. This area has mainly been focused towards concurrency-control, temporal consistency, overload management and scheduling. The focus within CBSE is to create software components that are reusable entities mounted together as building blocks with a possibility to maintain and improve systems by replacing individual components [3]. Even though RTDBMS and CBSE share the same goal, their means of achieving it is unfortunately conflicting. One key philosophy for most component models is that all communication with the surrounding environment should be performed through the component's interface, eliminating all component side-effects. By introducing a real-time database management system (RTDBMS) and giving components direct access to shared data in the database introduces such side-effects. Furthermore, constructing components that query a certain database engine, with a certain database structure (schema) severely reduces the possibility of component reuse. To remove conflicts between RTDBMS and CBSE we introduce INCENSE (information-centric run-time support for component-based embedded real-time systems), a framework which combines the best of these technologies.

It would be desirable to achieve a component-based system where data is reliably managed and structured to enable

*This work is supported by the Swedish Foundation for Strategic Research within the PROGRESS Centre for Predictable Embedded Software Systems.

```
 TASK oilTemp(void){
    //Initialization part
    int temp;
1   MimerDbP dbp;
2   MimRTDBPBind(&dbp,"Select TEMP from
                       ENGINE where
                       SUBSYSTEM='oil'");
    //Control part
    while(1){
3      temp=readOilTempSensor();
4      MimRTWriteInt(dbp,temp);
       waitForNextPeriod();
    }
 }
```

Figure 1. An I/O task that uses a Mimer RT database pointer.

flexibility, a system where soft and hard real-time tasks can execute and keep isolation properties, a system that can handle critical transactions and at the same time enable openness, a system where new functionality can be added or removed without side effects to the system. To achieve this we propose to use a RTDBMS, in this case Mimer SQL Real-Time Edition [5], a commercially available[1] real-time database, together with a component technology, in this case SaveCCT, to achieve an information-centric run-time platform.

1.1. Mimer SQL Real-Time Edition

The Mimer SQL Real-Time Edition (Mimer RT) [5] is a real-time database management system intended for applications with a mix of hard and soft real-time requirements. The hard real-time algorithms are based upon the work performed within the COMET research project [8]. Mimer RT uses the concept of database pointers [9] to access individual data elements in an efficient and deterministic manner. For soft real-time database management, standard SQL [2] queries are used. To achieve database consistency without jeopardizing the real-time requirements the 2V-DBP concurrency control algorithm [7] is used. 2V-DBP allows hard and soft transactions to share data independent of each other.

In Figure 1 an I/O task that reads a sensor and propagates it into the database is shown. The task consists of two parts, an initializing part, and a control part. In the initialization part, the database pointer is created (line 1) and associated with a data element (line 2). The `MimRTDBPBind` function executes the query and a direct link to the data element is established. In the control part, the sensor is scanned (line 3) and its value is written to the database (line 4). The `MimRTWriteInt` call uses the direct link and performs the write in constant time.

[1]Mimer SQL Real-Time edition will be available during Q2 2007.

1.2. SaveCCT Real-Time Component Technology

The SaveComp Component Technology (SaveCCT) [1] is described by distinguishing manual design, automated activities, and execution. The entry point for a developer is the Integrated Development Environment (IDE), a tool supporting graphical composition of components, where the application is created. Developers can utilize a number of available analysis tools with automated connectivity to the design tool. SaveCCT is based on a textual XML syntax which allows components and applications to be specified. Automated synthesis activities generate code used to glue components together and allocate them to tasks. Resource usage and timing are resolved statically during the synthesis instead of using costly run-time algorithms. SaveCCT is, as Mimer RT, intended for applications with both hard and soft real-time requirements.

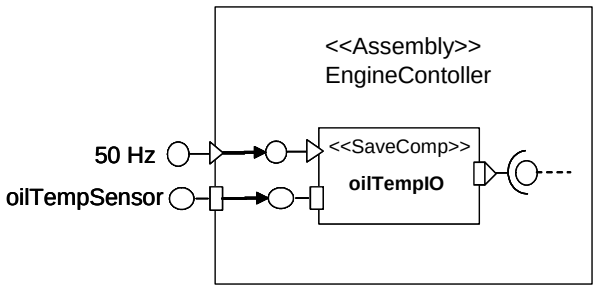


Figure 2. Save graphical application design

In SaveCCT applications are built by connecting components input and output ports using well defined interfaces, see Figure 2. Components are then executed using trigger based strict "read-execute-write" semantics. A component is always inactive until triggered. Once triggered it starts to execute by reading data on input ports to perform its computations. Data is then written to its output ports and outgoing triggering ports are activated.

Figure 2 shows how the XML code in Figure 3 is graphically represented. There are two inports into the Engine Controller application, data and trigger port. Data is read by the oilTempIO component from its inport oilTempSensor once triggered every 50Hz. Computations are done and results propagated onto the output port. In this case the output port is a combined trigger and output port. The XML code also includes ATTRIBUTE which describes the different component properties. In this example we have chosen to exclude all attributes except Worst Case Execution Time (WCET), which is analyzed and entered to the system.

```
<APPLICATION id="EngineController">
   <IODEF>
      <INPORT mode="trig" type="void"
              id="trigFiftyHz" value="20"/>
      <INPORT mode="data" type="int"
              id="oilTempSensor" />
   </IODEF>
   <TYPEDEFS>
      <COMPONENTDESC id="oilTempIO">
         <INPORT mode="trig" type="void"
                 id="trigOilTemp" />
         <INPORT mode="data" type="int"
                  id="oilTemp" />
         <OUTPORT mode="combined" type="int"
                  id="newOilTemp" />
         <ATTRIBUTE id="WCET" type="ms"
                    value="5" />
      </COMPONENTDESC>
      ...
   </TYPEDEFS>
   <CONNECTIONLIST>
      <CONNECTION>
         <FROM id="EngineController"
               port="trigFiftyHz" />
         <TO id="oilTempIO"
             port="trigOilTemp" />
      </CONNECTION>
      <CONNECTION>
         <FROM id="EngineController"
               port="oilTempSensor" />
         <TO id="OilTempIO" port="oilTemp" />
      </CONNECTION>
      ...
   </CONNECTIONLIST>
</APPLICATION>
```

(Figure simplified for readability)

Figure 3. SaveCCT XML description file

2. The Information-centric Component Framework

To efficiently integrate real-time database management and component-based software engineering in order to gain the potential benefits of both approaches, we propose that the RTDBMS is made part of the component framework. Figure 4 shows the architecture of the framework in which it acts as a proxy between the application components and the RTDBMS. This allows components to be *database unaware*. We define a database unaware component as a component which does not have knowledge of the database schema, i.e., the structure of the data in the database. This database decoupling is made possible due to *database proxies*, see Figure 4, which creates a database view that is consistent with the interface of the component.

If components are database aware, i.e., calls to a database is made from within the component-code, a number of unwanted properties emerge, such as (i) decreased component reusability, since the component can only be used in systems with a certain database schema, and (ii) undesirable component side-effects since interaction with the environment is made from outside the component interface.

In this paper, we distinguish between two types of database proxies, namely *hard real-time database proxies* (hard proxies) and *soft real-time database proxies* (soft proxies).

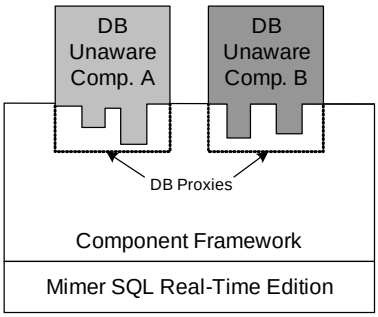


Figure 4. The Incense component framework

2.1. Hard real-time database proxies

Hard proxies are intended for hard real-time components, which need efficient and deterministic access to individual data elements. In Figure 5, a hard proxy is declared as a `DBHARDPROXY`. The declaration contains all information to set up a database pointer, which will be constructed in the component framework as glue code between component calls. Since 2V-DBP provides constant response-time for database pointers, an attribute for worst-case execution time is included in the declaration.

Hard proxies are connected to a component's in- or outport, and the data element in the database is either provided to the component or written back to the database after completion of the component's execution. As a hard proxy can provide a data element of any type, they can be used with any existing components since the database is fully transparent to the component.

2.2. Soft real-time database proxies

Soft proxies are intended for soft real-time components, which might need more complex data-structures. Consider a component monitoring the overall status of a subsystem, e.g., all the temperatures in an engine, or logging of errors etc.

In order for a component to be able to use a soft proxy, it must have a *relational interface*, which means that it must be able to take a relational table as a parameter (or return value). Therefore, the SaveCCT component-model is extended to include `TABLEDESC`'s as parameters, see Fig-

```
<TYPEDEFS>
   <TABLEDESC id="temperatureTable">
      <ATTRIBUTES>
         <ATTRIBUTE type="CHAR(20)"
                    id="subsystem"
                    key="primary"  />
         <ATTRIBUTE type="int"
                    id="temperature"
                    key="false"    />
     </ATTRIBUTES>
   </TABLEDESC>
</TYPEDEFS>

<DBPROXIES>
   <DBHARDPROXY type="int" id="oilTemperature"
                bind="SELECT temp FROM engine
                      WHERE subsystem='oil'"
                <ATTRIBUTE id="WCET" type="us"
                           value="5" />
   />
   <DBSOFTPROXY type="temperatureTable"
                id="engineTemperatures"
                bind="SELECT temp from engine"
                <ATTRIBUTE id="WCET" type="ms"
                           value="3" />
   />
</DBPROXIES>
```

Figure 5. SaveCCT proxy representations

ure 5. A TABLEDESC table descriptor is a relational table containing the information needed by the component. It is worth noting that the structure of this descriptor is completely decoupled from the database schema.

Soft proxies are, as hard proxies, connected to a component's in- or out-ports. At run-time, the hard proxy converts the database schema into the format of the table descriptor. This glue-code, i.e., the database query associated with the proxy, is embedded into the component framework.

This approach implies that the component is aware that an RTDBMS is present, but it is still generic with respect to the schema of the database, i.e., component reusability is maintained.

3. Conclusions and Future Work

In this paper we present a technique to integrate a real-time database management system into a component-based system, and thereby gaining the advantages of high level data management while retaining important properties, such as component isolation and reusability, of component-based software engineering. We introduce the concept of a *database proxy* to enable components to be database unaware, i.e., components do not need to be tailored to fit a certain database engine or database schema. Instead all database interactions are performed from the information-centric component framework used in this technique.

Key benefits of this approach include, system openness due to standardized query language languages, the possibility of creating on-the-fly dynamic database queries, total decoupling of data and components. Furthermore, when using a hard real-time database engine such as Mimer SQL Real-Time edition, hard real-time guarantees on data access is provided and synchronization of shared data is transparently managed. By separating data access and components, component isolation is retained and managed by application specific database proxies connected to component interfaces.

In our future work we intend to extend the information-centric view with high level tools and design paradigms to manage and organize data in a logical view rather than a physical. During design, developers should have full control of each data item involved, who are the producers/consumers, timing requirements etc. The overall aim of our work is to create an information-centric design paradigm for real-time systems, where data management is treated as its own design entity.

References

[1] M. Åkerholm, J. Carlson, J. Fredriksson, H. Hansson, J. Håkansson, A. Möller, P. Pettersson, and M. Tivoli. The Save Approach to Component-Based Development of Vehicular Systems. *Journal of Systems and Software*, 2006.

[2] S. Cannan and G. Otten. *SQL - The Standard Handbook*. MacGraw-Hill International, 1993.

[3] I. Crnkovic. Component-Based Software Engineering - New Challenges in Software Development. *Software Focus*, December 2001.

[4] L. Gabriel and H. Donal. Expanding Automotive Electronic Systems. *Computer*, 35(1):88–93, Jan 2002.

[5] Mimer SQL Real-Time Edition, Mimer Information Technology. Uppsala, Sweden. http://www.mimer.se.

[6] N. Navet. Trends in Automotive Communication Systems. In *Proceedings of the IEEE*.

[7] D. Nyström, M. Nolin, A. Tešanović, C. Norström, and J. Hansson. Pessimistic Concurrency Control and Versioning to Support Database Pointers in Real-Time Databases. In *Proceedings of the 16th Euromicro Conference on Real-Time Systems*, 2004.

[8] D. Nyström, A. Tešanović, M. Nolin, C. Norström, and J. Hansson. COMET: A Component-Based Real-Time Database for Automotive Systems. In *Proceedings of the Workshop on Software Engineering for Automotive Systems*, pages 1–8. The IEE, June 2004.

[9] D. Nyström, A. Tešanović, C. Norström, and J. Hansson. Database Pointers: a Predictable Way of Manipulating Hot Data in Hard Real-Time Systems. In *Proceedings of the 9th International Conference on Real-Time and Embedded Computing Systems and Applications*, pages 623–634, February 2003.

[10] K. Ramamritham, S. H. Son, and L. C. Dipippo. Real-Time Databases and Data Services. *Journal of Real-Time Systems*, 28(2/3):179–215, November/December 2004.

FPGA Design and Performance Analysis of a Mobile Video Processing System

Andréa Karsko, *Member, IEEE* and John Hodapp, *Member, IEEE*

***Abstract*—We describe an architecture design and implementation approach for a mobile real-time multi-camera video capture and processing system used to detect changes in scenery that have occurred between successive runs down the same path. The current system is implemented using traditional processor hardware and uses multiple cameras operating in different wavelengths to capture imagery in real-time. Captured imagery is analyzed by various image processing techniques in near real-time. We propose that these image processing algorithms be ported to a FPGA design for increased speed and flexibility with the goal of achieving hard real-time performance. Analysis and implementation are currently underway.**

***Index Terms*—FPGA, VHDL, mobile video processing**

I. Introduction

IN late 2005 a need for a vehicle-based video acquisition testbed was identified. This acquisition testbed requires a flexible and modular design to support rapid prototyping of signal and image processing algorithms. To achieve this requirement, a small multidisciplinary team of engineers were established and directed to design, develop, and use a vehicle-based testbed. Real-time acquisition and near real-time processing of georegistered data acquired simultaneously from multiple sensors was an initial requirement. The use of multiple sensors and sensor types in combination with GPS technology would permit the system to capture a wealth of data for algorithm research and development in areas such as object detection, scene change detection, and surveillance.

During development, various operational concepts (CONOPs) were explored to identify how such a testbed system could be configured to address specific requirements. One CONOP that resulted from the effort involves the use of two operators working together as a team to detect and identify objects in live streaming imagery. Here, one operator (OP1) screens the data acquired from wide field-of-view (WFOV) streaming sensors, taking advantage of any automated cues provided by the processing algorithms and overlaid onto the imagery. With the availability of archived imagery from previous runs on the same path, OP1 has sufficient information to perform visual change detection. When areas of interest are detected by OP1, image chips centered around these areas are sent to the second operator (OP2). OP2 has access to this imagery, as well as imagery taken from an additional pair of narrow field-of-view (NFOV) high resolution sensors that take detailed snap shots of these objects for identification purposes. This is the configuration of the system discussed here. The processing system associated with OP2 is not addressed here as the computational load of its software is minimal in comparison with that of OP1. Its overall system architecture, however, is similar. As such, any performance improvements resulting from architectural enhancements discussed here would apply to both operating systems.

Initial operational capability of the testbed was achieved in early 2006 with field trials. Currently, the system operates with measurable latency and does not provide hard real-time performance. Planned sensor enhancements, such as higher resolution cameras, increased frame rates, and new processing algorithms will add to the processor loading. In order to support an increased processor load while satisfying hard real-time requirements, Field Programmable Gate Array (FPGA) technology is being designed into the next generation system to support the additional workload.

II. System Design

To provide the greatest algorithm research and development capability in a variety of test environments, the system design must satisfy the following requirements, as shown in Figure 1.

- Simultaneous multi-sensor real-time acquisition
- Georegistration of imagery and time synchronization
- Real-time display and playback
- Near real-time algorithm execution
- Support vehicular operation

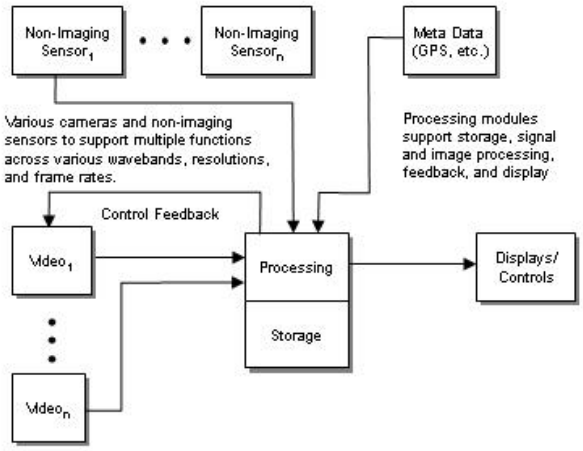


Fig. 1. Functional Vehicle Testbed System Requirements

The cameras selected for this application are all digital in nature. They each provide uncompressed raw digi-

US Army RDECOM CERDEC NVESD
Ft Belvoir, VA 22060, USA
US Army RDECOM CERDEC NVESD
Ft Belvoir, VA 22060, USA

tal data directly from an imaging CCD or focal plane array (FPA). The visible camera used for the fixed-forward WFOV context view is a bayer-encoded color camera. This camera produces imagery that consists of 1600×1200 frames. In the current testbed system, this sensor is triggered via a time-synched external source and maintains a 30 Hz input rate. Paired with this camera providing the WFOV infrared imagery is an uncooled microbolometer-based Foward Looking Infrared (FLIR) camera with a resolution of 640×512 at 30 Hz. This camera is not externally triggered and operates in free-run mode. The data rate resulting from these two cameras is approximately 134 MB/s.

Each acquired frame of input data is tagged with geolocation and heading information as well as time information from a microsecond-accurate clock. A positioning system comprised of a wide area augmentation system (WAAS) enabled GPS, coupled with an inertial measurement unit (IMU), is connected to the testbed via a serial connection. The data from this system are generated at a 100 Hz rate and provide sub-meter outputs once the IMU is initialized. This tagged data is placed in archived data sets and stored at a one frame per meter rate. A RAID subsystem combined with the high precision and high rate GPS/IMU measurements, provides real-time storage and playback of the georeferenced imagery.

Two Intel CPUs currently provide the computation resources available for the image and signal processing modules. These processors, combined with the various slots and interfaces needed to instantiate the acquisition and storage subsystems, are available on several standard workstation-class motherboards from commercial vendors. The current system utilizes the Windows XP/Pro operating system (OS). This OS was chosen because it provided the greatest base of compatible third-party products from which to select the cameras, interfaces, and support tools and libraries.

The software development tools are self-hosted and the design allows testing of various user-interface (GUI) configurations. The software development tools are flexible and compatible with each other at the application programming interface (API) level. Initial analysis of requirements, heavily influenced by the short development time, resulted in the use of many commercial off the shelf (COTS) products.

In order to refine the design, a market survey of the relevant technologies was conducted. In some cases, representative samples of the technologies were obtained, evaluated, and benchmarked. One result of the market survey, is the selection of the Camera Link standard for the digital camera interface which permits for a wide selection of camera technology. To support research and development using sensors that do not adhere to the Camera Link standard, flexible off-the-shelf translators that convert RS422 and low-voltage differential signaling (LVDS) to Camera Link are employed.

III. Current Architecture, Performance, and Limitations

The hardware architecture of the testbed is based upon an Intel server system and consists of dual Intel Xeon (Nocoma) processors in a Symmetric Multiprocessor (SMP) configuration, each executing at 3.60 GHz. A general outline of the hardware architecture is shown in Figure 2. Note there are only two frame grabbers in the system, CL

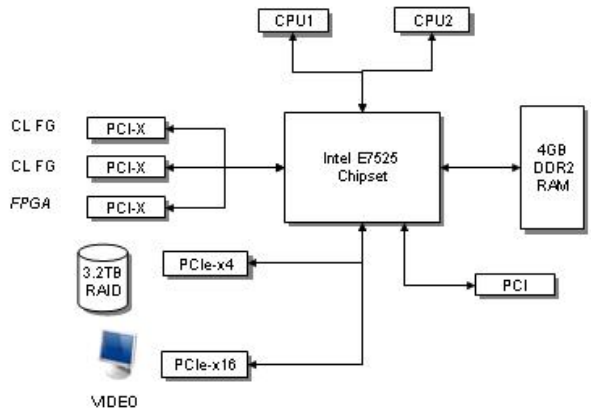


Fig. 2. Current Hardware Architecture

FG on the left side. However, each of the frame grabbers permit acquisition from two Camera Link sensors simultaneously. Therefore, as configured, the system can thus support four of these cameras. A high resolution ruggedized touchscreen serves as the primary operator interface to the system. Although not pictured in Figure 2, lower bandwidth interfaces including RS232 and gigabit ethernet (GigE) interface are also provided.

The design of the software architecture is modular in nature and a description is shown in Figure 3. Each of the main functional areas are implemented as software threads. These threads consist of the application, data management, and display threads, to name a few, and are implemented in a similar fashion. Each of the major threads contain code and buffers to permit synchronization and exchange of data between the various software subsystems in a controlled manner. Once started, the main application does the customary initializations typically found in a complex application with multiple software subsystems. Upon completion of setup and initialization, application threads are launched. These threads then remain active for the duration of the application. The threading approach employed spawns all user-level threads during system initialization and destroys those threads at system shutdown. No new threads are spawned to service incoming data. After the system is fully initialized, video data arrives into buffers residing in the frame grabbers. After each frame grabber transfers the data into local system memory, a callback function is invoked to notify the appropriate thread that data is available. These callback functions are key and their implementation requires real-time operation (execution within 33 ms).

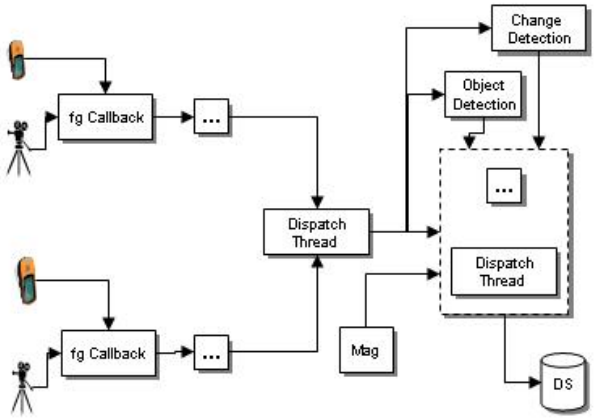


Fig. 3. Current Software Architecture of the Vehicle Testbed

A key role of the callback function is to acquire a set of meta-data, such as GPS location and time, to accompany each frame of acquired data. The callback functions retrieve the meta-data from a continuously updated ring of GPS data that is managed independently by a GPS thread. The digital video data, as well as data from the non-imaging sensors, are dequeued by a separate thread and forwarded to the individual application threads. These include the algorithm threads such as video change detection. Data is also forwarded to the data storage subsystem for archiving.

The performance of the current system is such that live video acquisition and storage from the two high resolution WFOV streaming cameras is sustained at 30Hz. The display rate of the WFOV visible camera is supported at 11 Hz and the FLIR at 30Hz, in parallel with display of the each georeferenced archived data stream, respectively. Although acceptably responsive, the system is heavily loaded with high CPU utilization when all algorithm and display modules are in operation. This can be seen from Table I. Steps have been taken to optimize the implementation of the code modules that are CPU and I/O intensive. Optimized math libraries are also currently employed and detailed code analysis of critical sections have been done, resulting in improved data flow and reduced execution times. However, since the system is dynamic, methods must be investigated to continue increasing performance.

Average Display Rate	11.554 Hz
Object Detection Alg1	0.0308 sec/frame
Object Detection Alg2	1.041 sec/frame
Bayer Conversion	0.0565 sec/frame
IR Image Display Prep	0.0173 sec/frame

TABLE I
CURRENT PERFORMANCE CHARACTERISTICS

It is believed that the current performance limitations of the system can be eliminated by a well-defined application utilizing FPGA technology. Functions such as Bayer decoding, display preprocessing, IR histogram equalization, video stabilization and key portions of various algorithms, to name a few, are promising candidates for FPGA execution.

IV. Next Generation Architecture

As stated, the current architecture of the system is designed to be modular. This permits the redesign of this system from a complete software implementation to a combination of software and hardware implementation. A PCI based FPGA card will be inserted into an unused backplane slot. The FPGA contained on the development card is a Xilinx Virtex-II Pro 30. This FPGA has two PowerPC processor blocks, 644 user defined I/O, and approximately 31,000 logic cells. The design therefore should not be limited by the size of the FPGA.

There are two distinct approaches currently being considered for the next generation system. Both approaches utilize different aspects and strengths of the FPGA. The first approach does not utilize the FPGA as much as the second, but may offer reduced complexity and a shorter implementation time frame. This design keeps a majority of the pre-processing duties on the host computer. These pre-processing duties consist of preparing the data to be analyzed by the FPGA modules and selecting the appropriate FPGA modules to be executed. With this approach the necessary data will be transfered over the PCI bus to memory located on the FPGA board. The PowerPC is notified with a data ready flag then the VHDL modules on the FPGA will perform the necessary operations and notify the host computer when finished. In this approach, the slowest and most portable portions of the system are moved to the FPGA and the system does not take full advantage of the built in PowerPC.

In contrast, the second approach utilizes the PowerPC of the FPGA. This design moves the pre-processing duties into the PowerPC so that the communication between the pre-processing and the VHDL hardware modules are closer together. This will allow for increased speed between the PowerPC and the VHDL modules since communication is contained on only the FPGA board, as opposed to communication occurring between the host computer and the development board over the PCI bus. Here, the PowerPC is directly involved in the entire system design by controlling the communications between the host computer and the FPGA development card. At this point, it is not known which approach is optimum, although it is hypothesized that the second approach (taking advantage of the PowerPC cores) may simplify host-FPGA control.

V. Research Tasks

Although the next generation design of the vehicle testbed is still in the development stages, an implementation approach as well as possible challenges must also be considered. A comprehensive approach that permits an evaluation of the two candidate FPGA solutions suggested in the previous section shall be performed. First, a thorough examination of the current working system and

subsystems will be completed. Each part of the system will be analyzed to identify functions, algorithms, or entire subsystems that can benefit from the strengths of a FPGA implementation. Using this analysis, these selected functions, algorithms, or subsystems will be implemented and tested on a FPGA. Finally, the current system will be evaluated against the FPGA-augmented system, with performance differences measured and documented.

Section III identifies several bottlenecks in the system. These include image analysis functionality, such as the change detection, video stabilization, and display latency. For the next generation system, the design will remove the software implementations of these modules and replace them with VHDL cores. These VHDL modules will be designed with portability in mind. Therefore, once they are designed, implemented, and tested, the modules can be added to both next generation design approaches so that both can be fully evaluated and tested for performance differences.

Algorithms such as change detection and video stabilization fit well into a parallel processing system. Typically, image processing algorithms are computationally intense and therefore are more efficient in the FPGA fabric. When implemented in the FPGA fabric, their only limiting factor is the speed required to pull data into the FPGA and to push it out of the FPGA.

While the theory of implementing a PowerPC system with custom VHDL modules is straight forward, the actual implementation can be challenging. Some of these challenges includes efficient VHDL design and determining an optimal degree of FPGA/CPU coupling to preserve quick implementations and hard real time requirements. Significant improvement in processing capability due to the use of a FPGA makes an increase in implementation complexity worthwhile given future algorithm plans. Depending on the FPGA design size this may pose to be difficult. Other challenges exist during the testing phase of development. Testing the VHDL modules, is relatively straight forward with simulators using waveform test benches to simulate valid data. But when testing the PowerPC system utilizing the VHDL modules more variables are introduced resulting in a longer debug and testing cycle. However, by anticipating these issues, it is hopeful that proper planning can help eliminate these challenges with more ease.

To test the next generation system for performance improvements, subjective and quantitative aspects will be considered. Pre-recorded digital data serving as controlled test data will be injected into each system. The data will be processed by both implementations (FPGA and non-FPGA) and timed for system efficiency, display latency, and accuracy of results (from the various algorithm modules.) The qualitative results are measured via timing results on each perspective system. With respect to the FPGA design, simulations combined with actual timing results will be used to establish the system processing time. In the case of the software system, timers will be used in the software to calculate the time needed to perform the same calculations. Using the qualitative in conjunction with the subjective comparisons, adequate knowledge of the FPGA performance increase will be obtained.

VI. Next Steps

Upon review of the system, software modules that are best suited for a VHDL implementation must be identified. Next, coding and development of test benches for the VHDL cores shall be performed to establish performance baselines. Test data shall be executed in the VHDL cores and compared against outputs from the software modules in the current system, documenting differences in order to understand and eliminate them as much as possible. VHDL cores and associated control software shall be integrated into the overall system and tested. Finally, the performance of the implementations shall be measured and evaluated.

Acknowledgment

The authors wish to acknowledge Dr. Christopher Marshall (US Army RDECOM CERDEC NVESD), Mr. Sean Jellish (US Army RDECOM CERDEC NVESD), and Mr. Ken Ramsey (EOIR Technologies) for their significant contributions to this effort.

Use of Helper Threads for OS Support in the Multithreaded Embedded TriCore 2 Processor

Florian Kluge, Jörg Mische, Sascha Uhrig, Theo Ungerer
HiPEAC European Network of Excellence
University of Augsburg, Augsburg, Germany
{kluge|uhrig|ungerer}@informatik.uni-augsburg.de

Rafael Zalman
HiPEAC European Network of Excellence
Infineon Technologies AG, Munich, Germany
rafael.zalman@infineon.com

***Abstract*— Infineon equipped their TriCore 2 microcontroller with multithreading capabilities. As memory protection techniques are getting more important, it also implements a range-based memory protection system. Based on the multithreading capability a helper thread can run in a thread slot in separation from the real-time application thread to support embedded operating systems like OSEK or AUTOSAR OS used in automotive systems. We show that our concept can save more than 70% of task switching time by pre-loading the memory protection registers for the application that is predicted to be scheduled next. Also, we propose modifications to the TriCore 2 architecture that would support our concept.**

I. Introduction

A multithreaded processor [14] is characterized by the ability to execute instructions of different threads within the processor pipeline simultaneously. The contexts of two or more threads of control are stored in separate on-chip hardware thread slots each including its own register set, instruction pointer, and processor status registers. Multithreading within one processor can be used to hide memory latencies (e.g. from instruction fetching or data loading) of one thread while executing another thread. Typically, application threads are loaded into the hardware thread slots. Another application domain for this kind of threads are helper threads that run separated from an application and support the application or a running operating system. Such helper threads are proposed for tasks like branch prediction [2], prediction of accessed memory addresses [3], [9], [16], exception handling [8], [15] and accelerated execution of loops [10]. In the embedded Java microcontroller Komodo helper threads are also used for the garbage collection of hard real-time threads [13] and dynamic preloading of software upgrades of running hard real-time threads [12].

In section II we present characteristics of the TriCore 2 architecture relevant for our work. In section III we develop a helper thread concept to utilize the TriCore 2's second thread and propose some changes to the TriCore 2 architecture. Section IV concludes this paper.

II. Infineon TriCore 2 architecture

The Infineon TriCore architecture defines a 32-bit microcontroller, which is mostly used for automotive applications. It combines a RISC load/store architecture with a DSP-like Harvard memory architecture.

The TriCore 2 is binary compatible to its predecessor, but provides a second hardware thread slot, which can be used to bridge long instruction fetch latencies [11]. Currently, the TriCore 2 architecture is licenced as an IP Core.

A. TriCore 2 Multithreading

The Infineon TriCore 2 features two hardware threads, T0 and T1 [5]. Generally, a program is executed in T0. When the pipeline impends to stall due to long instruction fetch latencies, execution can be transferred to the second thread T1. This thread usually is executed from fast on-chip scratchpad RAM. It is also possible to have T0 and T1 running alternately by setting for each thread a number of clock cycles it should run. Although, there are some restrictions to T1. Interrupt service requests will always be served by T0. Furthermore, T1 is only allowed to run with interrupts enabled. Disabling interrupts automatically transfers execution to T0.

As can be seen from these restrictions, the two threads of the TriCore 2 are not fully equal. Applications running in T1 are restricted to threads that need not to disable interrupts. Examples are the "untrusted" applications in AUTOSAR [1]. Another use for T1 would be as helper thread that supports a running application or operating system, as proposed in the next section. The TriCore 2 scheduler ensures that the real-time behaviour of the application thread in T0 is not disturbed by the execution of a helper thread in T1.

B. Memory Protection

Current automotive control units usually run several applications. There need to be ways to ensure that an application cannot be harmed by other applications, i.e. to prevent other applications from manipulating their data or even code. This can happen due to programming errors, for example. Current automotive software specifications, like AUTOSAR or Protected OSEK [4] pick up this problem by claiming the existence of some kind of hardware-based Memory Protection System (MPS).

Usually one of two kinds of MPS are implemented in current microcontrollers. The *page-based* approach allocates memory in the form of equal-sized pages, e.g. 1kB. It is possible, to have as many pages as desired for an applications. The management of these pages is usually done by the Memory Management Unit (MMU) or a special Memory Protection Unit (MPU).

The other technique is the *range-based* approach. Here, the CPU or MMU has some special registers, where memory ranges are described by lower and upper bounds. There are usually separate sets for data and code memory, differing in the kind of access privileges (Read/Write/eXecute).

The Infineon TriCore family offers a range-based memory protection system (MPS) with two to four Memory Protection Register (MPR) sets each for data and code memory [7] (see Figure 1).

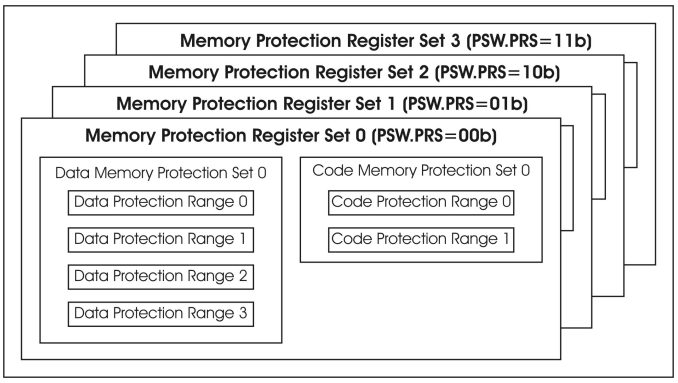


Fig. 1. Memory Protection Register Sets of the TriCore architecture [7]

The real number of MPR sets depends on the implementation of the processor. Each register set is made up of several range registers. Here again, the number of range registers is implementation dependent. Figure 1 shows the range register sets of the TriCore architecture, however, a TC1130 [6] TriCore processor implements only two of the possible four MPR sets. Most other TriCore implementations share this configuration with four data memory protection ranges and two code memory protection ranges (as far as they implement a MPS at all). For the TriCore 2 architecture the afore said also applies [5]. From the available MPR sets, at each time exactly one can be active, while the others are not considered. The active set is referenced in the `PSW.PRS` bits, as shown in Figure 1. If within one set there are overlapping ranges, the least restrictive access privileges are applied to the memory access.

III. A Helper Thread for the TriCore 2 Microcontroller

A. Design

Within an AUTOSAR OS implementation, each application is assigned its own Memory Protection Register Set. During scheduling, the MPR set must be changed along with the application's other context data. On a TC1130, the whole process of switching from one application to another (determining the next application and switching the context data) takes about 1400 clock cycles.

As the software on such a node usually is statically configured, it is simple to determine the next application at each point of time. The only exception to this rule is the occurence of interrupts. Here the regular flow of execution may be disturbed, but as it is induced from outside the processor, we cannot do anything about this case.

Now, ideally we have a helper thread running that predicts the next application and loads all context data in advance into the processor. Thus, at the point of scheduling the processor would only have to switch from one context set to another. This is not possible, because the processor contains only two sets of context data that both are in use already for the application thread and for the helper thread. So this technique would require a third set of context data.

However, we can speculatively determine the next task to be scheduled and pre-load its Memory Protection Registers. Here we assume a minimum of two MPR sets from which only one can be active at a time. So the other one could be used for the pre-loading of memory protection registers.

All calls to operating system functions will be done using the `syscall` trap that transfers the execution into a privileged mode. The operating system itself will then have full access to all memory areas. Thereby we assume the OS is correctly implemented. Thus, we would get by with the two available register sets.

Although, there is one drawback to our concept. As mentioned above, the currently selected MPR set of a running task is referenced in the `PSW` register of the CPU (`PSW.PRS`, see figure 1). This register is saved at each `call` instruction into the context save area and restored at the corresponding `return`. Now, if a task is assigned another MPR set than it had before its last preemption, all these values in the task's context save area need to be adjusted. The complexity of this operation depends linearly on the depth of the task's current call stack and would nullify our gained speed-up. Therefore, we could not yet evaluate our concept on a real TriCore 2 processor.

B. Proposed Architectural Changes

To overcome these problems and make good use of the second hardware thread, we propose the following changes to the TriCore 2 architecture:

- Implementation of all four memory protection register sets, and
- Split `PSW.PRS` into a local bit (`PRSL`) that is saved with each context, and a global bit (`PRSG`) that is not put into the context save area.

Thus, we would have two sets of memory protection register sets, where the globally active one is referenced by the `PSW.PRSG` bit, and therein the actually active one is referenced by the `PSW.PRSL` bit.

With the additional MPR sets, we would also be able to protect the operating system. Thus, risks through programming bugs would be reduced.

Figure 2 shows how we intend to use the four MPR sets. Both sets with `PSW.PRSL`=0 will be used for the operating system, i.e. they contain the same values. Thus they could be mapped onto the same hardware registers. The other two sets are used for the application. Now, the helper thread can preload the register values in the currently inactive set, and

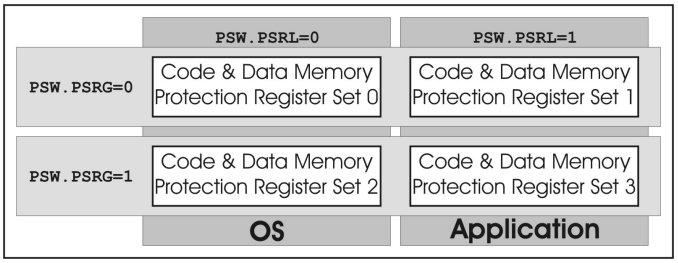


Fig. 2. Proposed organization of the Memory Protection Register Sets of the TriCore architecture

at an application switch, only the `PSW.PRSG` bit needs to be flipped, and the processor would automatically run with the correct MPR set.

C. Evaluation

We did some measurements using a TC1130 to find out what improvement would be possible. Here the complete process of scheduling takes about 1400 clock cycles. Thereof, determination of the task to be scheduled next amounts up to about 900 clock cycles. Nearly 300 cycles are needed for the swapping of OS management data. Loading of the memory protection registers amounts to 200 cycles. So if a helper thread predicts the next task correctly and already loads its memory protection registers, 1100 cycles (78%) of the total context switch time can be saved. For the remaining 300 cycles, we see no way for further improvements, as the data processed here directly depends on the program flow.

IV. Conclusion

We have presented a possible application for the Tricore 2's second hardware thread. As shown, the use of a helper thread for the pre-loading of memory protection registers would speed up application switching over 70%, if the next application is predicted correctly. Also, we have shown that only a small change in the processor architecture would be necessary to implement our proposed helper thread concept. As the TriCore 2 is traded as an IP Core, licensees would be able to easily adopt our proposed enhancements into their processors.

In the future, we intend to apply the proposed changes to a TriCore 2 simulator and evaluate the actual possible speed-up. The outcome of this evaluation is particularly dependent on the ratio of a correct task prediction.

V. Acknowledgements

This work was performed during a PhD internship of Florian Kluge at Infineon Technologies AG. The internship was supported by the EC Network of Excellence HiPEAC. Florian Kluge would like to thank Erik Norden for his mentoring during the internship.

References

[1] AUTOSAR AUTomotive Open System ARchitecture. http://www.autosar.org.

[2] Chappell, R. S., Stark, J., Kim, S. P., Reinhardt, S. K., and Patt, Y. N. Simultaneous subordinate microthreading (ssmt). In *ISCA* (1999), pp. 186–195.

[3] Collins, J. D., Wang, H., Tullsen, D. M., Hughes, C. J., Lee, Y.-F., Lavery, D. M., and Shen, J. P. Speculative precomputation: long-range prefetching of delinquent loads. In *ISCA* (2001), pp. 14–25.

[4] HIS Standard Software Working Group. *OSEK OS Extensions for Protected Applications*, 1.0 ed., July 2003.

[5] Infineon Technologies AG. *Tricore^{TM} 2 Architecture Manual*, 1.1 ed., June 2003.

[6] Infineon Technologies AG. *TC1130 User's Manual*, 1.3 ed., November 2004.

[7] Infineon Technologies AG. *Tricore^{TM} 1 Core Architecture*, 1.3 ed., February 2005.

[8] Keckler, S. W., Chang, A., Lee, W. S., Chatterjee, S., and Dally, W. J. Concurrent event handling through multithreading. *IEEE Trans. Computers 48*, 9 (1999), 903–916.

[9] Luk, C.-K. Tolerating memory latency through software-controlled pre-execution in simultaneous multithreading processors. In *ISCA* (2001), pp. 40–51.

[10] Marcuello, P., González, A., and Tubella, J. Speculative multithreaded processors. In *International Conference on Supercomputing* (1998), pp. 77–84.

[11] Norden, E. Keynote: Multithreading for low-cost, low-power applications. In *ARCS* (2004), C. Müller-Schloer, T. Ungerer, and B. Bauer, Eds., vol. 2981 of *Lecture Notes in Computer Science*, Springer, pp. 4–5.

[12] Pfeffer, M., and Ungerer, T. Dynamic real-time reconfiguration on a multithreaded java-microcontroller. In *ISORC* (2004), IEEE Computer Society, pp. 86–92.

[13] Pfeffer, M., Ungerer, T., Fuhrmann, S., Kreuzinger, J., and Brinkschulte, U. Real-time garbage collection for a multithreaded java microcontroller. *Real-Time Systems 26*, 1 (2004), 89–106.

[14] Ungerer, T., Robic, B., and Silc, J. A survey of processors with explicit multithreading. *ACM Comput. Surv. 35*, 1 (2003), 29–63.

[15] Zilles, C. B., Emer, J. S., and Sohi, G. S. The use of multithreading for exception handling. In *MICRO* (1999), pp. 219–229.

[16] Zilles, C. B., and Sohi, G. S. Execution-based prediction using speculative slices. In *ISCA* (2001), pp. 2–13.

Adaptive Mixed Query Scheduling in Real-Time Data Streams

Xin Li, Li Ma, Li Kun, Kun Wang, Hongan Wang, *Member, IEEE*

Abstract

This paper focuses the problems of real-time query scheduling and load management in a data stream management system. A mixed query model is introduced, which characterizes periodic and continuous real-time queries on data streams. Based on the strategy of bandwidth preserving, an integrated algorithm called Adaptive Mixed Query Scheduling (AMQS) is proposed. The objective of the scheduling algorithm is to guarantee the deadlines of periodic queries and minimize the number of deadline violations for continuous queries, while at the same time maximize the overall query quality according to load management strategy.

1. Introduction

A growing number of information processing applications, including traffic engineering, stock trading, network monitoring and sensor network, have to manipulate high volume stream data in a timely manner[1]. These applications have sparked researchers' interest in the area of data stream management system (DSMS) in both the database and real-time computing communities. A growing body of research rages from synopsis and algorithms for stream processing to prototype systems such as Aurora[3], STREAM[4]. Run-time resource allocation and optimization is one of the most important components in DSMS. In this paper we focus on CPU time allocation and load management in real-time circumstance.

In a real-time DSMS, continuous queries have to be completed by certain deadlines for the results to be of full value. For real-time applications, approximate query answers in time are more preferred to accurate results that miss deadlines. To provide predictable query responses, RTStream [2] proposed a periodic real-time query model for data streams, which eliminates the drawback that the triggered time and frequency of query instances can not be controlled in a continuous query model. In this paper, we consider both periodic and continuous queries with timing constraints and propose a mixed query model for real-time applications.

Several algorithms based on bandwidth preserving strategy are proposed to joint schedule soft aperiodic tasks and hard periodic tasks in real-time system, such as DDS[5], DSS[5], TBS[6,7], DPE[6] and EDL[6]. Among these algorithms, the Total Bandwidth Server (TBS or TB Server) showed the best performance/cost ratio [7]. The CPU utilization factors for a given periodic and aperiodic tasks are fixed in TB Server. However, the workloads for periodic queries often vary for the unpredictable arrival time and content of stream data. So the existing algorithms can not be using directly for the mixed queries over data streams.

To process both periodic and continuous queries, we give an adaptive mixed query scheduling (AMQS), in which EDF is used to schedule periodic queries and TB Server for continuous queries. The objective of the scheduling is to guarantee the deadlines of periodic queries and minimize the number of deadline violations for continuous queries. In order to guarantee the schedulability of periodic queries, some continuous query must be rejected in overload conditions. To achieve fair scheduling for mixed queries, QoS-aware load management and adaptive adjustment for CPU utilization factors using feedback mechanism are adopted according to system resource configuration and workloads. In our new mixed query model, the strategy introduces graceful degradation and effectiveness under any load condition.

2. System Overview

Now we are developing a prototype named RT-DSMS based on the STREAM [4] system running on a Linux PC to experimentally evaluate our mixed query model and real-time scheduling strategy. We have implemented the detailed design of the system.

Figure 1 illustrates the system architecture of our proposed prototype system. The system consists of five components: data source manager, query processing engine, real-time scheduler, QoS Monitor and Load Shedder. Data source manager receives data from multiple streams and inserts data into corresponding input queues of query plans as well as monitors input characteristics of streams (e.g. stream rates). The query processing engine compiles user-defined queries into query plans, i.e. operator network, and optimizes them dynamically. According to scheduling strategy, the real-time scheduler chooses one operator in operator network to execute during running time. To meet QoS requirements of queries, the QoS monitor periodically collects system performance, evaluates workload and employs various delivery mechanisms(such as load shedding and admission control) to guarantee the QoS of various queries. The load shedder reduces workloads by dropping a portion of tuples. The statistical characteristics of each stream and operator provide

useful information for query optimization, cost estimation and load shedding. In this paper, we concentrate on query model, real-time scheduler and load shedder.

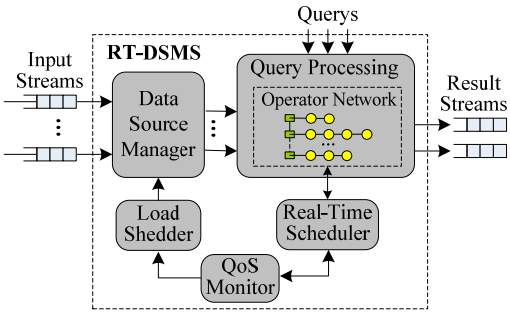


Figure 1. Architecture of RT-DSMS

In RT-DSMS, a long-running query (periodic or aperiodic) is pre-registered and decomposed into an execution plan tree composed of a set of basic pipelined operators (such as project, select, join, aggregate) and rooted by an output operator. Multiple query execution plans over streams can be modeled as a directed acyclic graph, termed operator network, in which a node represents an operator that process tuples and a directed edge between two adjacent nodes represents the queue connecting those two operators. In query plans, the output of a former (or child) operator is buffered in a queue which acts as the input queue to the later (or parent) operator. The path between a source node and an output node excluding the two nodes forms an *Operator Path* (OP).

3. Query Model

3.1 Assumptions and Notations

A data stream is defined as a real-time, continuous, ordered sequence of data items [1]. It is impossible to control the arrival rates and contents of the data streams. Typically, queries are constrained to process data inside a sliding window, which is a recent segment of data stream. Although the memory constraint is an important yet complicated research problem, in this paper, we only consider the CPU resource constraints and assume that there is always enough main memory space for query processing.

A Data stream management system is a query processing system on dynamic data streams, in which queries are known as long-running and persistent queries. Any kind of query Q_i consists of a sequence of instances (or jobs) q_i (j).

In this paper, we consider a firm real-time DSMS, i.e. a query instance becomes worthless if it fails to complete by its deadline.

In addition, we make the following assumptions:

1. The arrival time of each tuple and stream rates r_i are unknown.

2. The selectivity s_i and the execution cost c_i of any operator are known as they can be derived by running-time statistics of operators.

3. The deadline of a periodic query is equal to its period for simplicity, i.e., D_i=T_i.

4. The transient system overload does not persist for a long period of time.

3.2 Mixed Query Model

We propose the *Mixed Query Model* (MQM) for real-time applications that need QoS-aware query processing. In this model, all queries fall into two categories: Continuous Query (CQ) and Periodic Query (PQ). The former is trigged to execute by aperiodic data while the latter runs periodically.

For continuous queries, the query instances are triggered by unpredictable streaming data. These queries run repeatedly at an interval of unfixed time after they are registered. They are data-initiative, i.e., the execution instance of continuous queries are released by data arrival. Continuous query processing is suitable to use in an environment where input rate is low and gentle.

For the periodic queries, they are registered in advance and run repeatedly over a fixed period of time. Their instances are initiated periodically by the system. The life time of a periodic query can be divided into two parts: execution stage and dormancy stage. During execution stage, a query instance takes the data in a jumping window [1] over streams as input tuples and processes them in batches according to query plan. Consequently, the results are produced and outputted to external applications periodically. The data in a window do not change during dormancy stage even when new data arrive in its input queue.

Each query (periodic or continuous) is decomposed into a query plan when it is registered in the system. After the query plans are generated, the operators are sent to the scheduler to execute. The scheduler creates periodic query instances at intervals of their periods and adds them into ready queue of the scheduler. For a continuous query, the query instance is added to ready queue when a tuple arrives at its input queue. After the scheduler chooses a query instance to execute, the operators in corresponding query plan are scheduled one by one to process input tuples. Finally, those tuples are pushed through the operator paths and outputted to applications or they are consumed by intermediate operators.

3.3 Performance Metrics

- **Deadline Miss Ratio**

In a firm real-time DSMS, query execution time is constrained by deadline and a query result is useful for applications only if it is produced within its deadline. Therefore, the primary metric is deadline miss ratio (DMR), which denotes the percentage of query instances that do not complete before their deadlines.

$$DMR=\sum_{i=1}^{m} N_i^C \Big/ \sum_{i=1}^{m} N_i \quad (1)$$

In the above formula, N_i and N_i^C are the count of all instances of the i-th query and those finished in their deadlines, respectively.

- **Data Sample Ratio**

During high overload stage, a feasible solution is to gracefully drop a portion of unprocessed or partially processed tuples. To depict the problem, the date sample ratio is used to measure the percentage of input tuples processed to produce query results. RT-DSMS allows applications to specify the relations between output quality and sample ratio by a two-dimensional QoS graph.

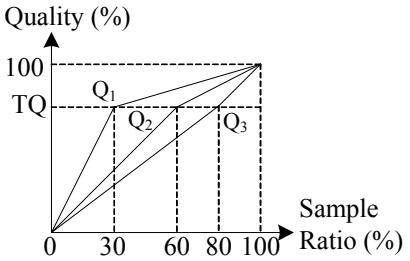


Figure 2. QoS Graph based on Sample Ratio

For simplicity, the QoS graph is shown as a piece-wise linear function of sample ratio with one critical point, at which a query reaches its tolerable quality (TQ) with the least sample ratio. In figure 2, Q_1 can tolerate dropping more tuples than Q_2 and Q_3.The sample ration at the critical point is called the Least Tolerable Sample Ratio (LTSR). The LTSRs are equal to 30%, 60% and 80% for Q_1, Q_2 and Q_3 separately. Based on the QoS graph assigned by applications, RT-DSMS can supply satisfying QoS for as many queries as possible, meanwhile minimize the system workloads.

4. Real-time Query Scheduling

For a mixed query set, an available approach that does jeopardize the schedulability of periodic queries is to introduce a special purpose query called *server* [6], whose computation time is used to process continuous queries. In order to improve the response time for continuous queries, the server is usually scheduled by a specific algorithm. In this paper, the Total Bandwidth Server [6] is used as the aperiodic server algorithm.

As illustrated in figure 3, the adaptive mixed query scheduling is composed by two levels: the Earliest Deadline First (EDF) scheduler is used to schedule periodic queries in ready queue and a TB server to choose one of instances in CQ instance queue into ready queue. The TB server organizes the CQ instance queue according the EDF strategy. At a time, only one continuous query instance can enter into the ready queue and be scheduled together with periodic queries by the EDF scheduler.

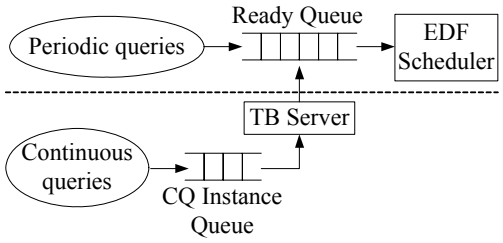


Figure 3. Mixed Query Scheduling Model

The objective of this scheduling is to guarantee the deadlines of periodic queries and minimize the number of deadline violations for continuous queries, while at the same time maximize the overall query quality according to load management strategy.

4.1 Total Bandwidth Server Algorithm

The key idea of the TB Server algorithm is to assign each CQ instance a deadline as early as possible, only if the schedulability of PQ instances is not affected. Suppose that U_P is the CPU utilization factor for PQ, and U_C is for CQ instances. It has been proven in [7] that, the schedulability of periodic tasks in the presence of TB server can simply be tested by verifying the following condition: $U_P+U_C \leq 1$.

TB server assigns a suitable pseudo-deadline to the query instance and to schedule it according to the EDF algorithm together with the periodic queries in the system. On one hand, the deadline is the shortest possible to improve the aperiodic responsiveness. On the other hand it must not jeopardize the schedule of periodic queries. The definition of the server is the following. When the k-th continuous query instance arrives at time $t=a_k$, it receives a deadline $d_k=\max(a_k, d_{k-1})+C_k/U_C$, where C_k is the remaining time of expected execution cost of the instance and U_C is the CQ server utilization factor. By definition d_0=0. The CQ instance is then inserted into the ready queue of the system and scheduled by EDF scheduler, as any PQ instance.

Different PQs maybe have different periods, so we introduce the term *super-period* to analysis the schedulability of multiple PQs. The super-period is the least common multiple of all periods of PQs. Given a set of PQs, the super-period is fixed before query execution. Since the starting time and count of the periodic query instances in a super-period are easier to estimate at any given time, it is available to check the schedulability of the set of PQs.

4.2 Feedback Mechanism

The performance of the mixed scheduling strategy is affected by the two parameter U_P and U_C. Statically setting the parameters does not work well because the query execution time determined by the cost and selectivity of operators and stream characteristics changes

over time. Therefore, we use feedback mechanism to adjust the parameters dynamically.

Proportional-Integral-Derivative (PID) controller is not suit to DSMS because the workloads vary dramatically from one sampling period to another [2]. We use a Proportional-Integral (PI) controller to control the parameter U_P.

$$\begin{cases} U_P(k) = \max(U_P(k-1) + \Delta U_P(k), EL_P(k)) \\ \Delta U_P(k) = \alpha(MR_\delta - MR_C^S(k-1)) + \beta \times (MR_\delta - MR_C^L(k-1)) \end{cases} \quad (2)$$

$EL_P(k)$ is the average workload of periodic queries in one unit and it is estimated through LTSRs.

$$EL_P = \sum_{i=1}^{m} r_i \times \overline{C_i} \times LTSR_i \quad (3)$$

where m is the count of period queries and $\overline{C_i}$ is the cost that the i-th query plan takes to process one input tuple. r_i and $LTSR_i$ is the stream rate and the least tolerable sample ratio corresponding to the i-th query plan.

MR_C^S and MR_C^L are the short-term and long-term miss ratio of continuous queries measured periodically by QoS monitor, respectively. MR_δ is the maximum miss ratio for continuous queries assigned by the administrator. α and β are two controller parameters which control the weights for the EL_P and short-term query miss ratio. In this paper, α and β are set by the administrator to give the balanced allocation between periodic and continuous queries to achieve the scheduling objective.

5. Load Management

The RT-DSMS must be able to predict the workload in supper-period and distribute CPU time over the query instances before execution, or some of the instances might miss their deadlines. Due to the unpredicted stream rates and time-varying contents, the system can be overloaded for a short time. To solve the problem, the accuracy of query result may often be traded off for timely response. Our load management is driven by the quality of query service.

The load management is divided into two levels: global management and local management.

In global load management, CPU time is pre-distributed fairly over all active query instances in a super-period and global load shedding is adopted under overload to guarantee maximal overall query quality. Every query instance is assigned a pseudo-deadline, based on the estimated execution time.

The local load management is adopted to distribute the allocated time over different operators during the execution of one query instance to maximize its quality. The sample ratio of the operators is calculated to finish the instance within the pseudo-deadline.

6. Future Work

For future work, we plan to test the mixed query model and scheduling algorithm with different workloads and configurations. Based on the mixed model, a scheduling strategy involving the requirements of CPU and memory usage will be considered in the future. Another work is to explore ways to shorten the adjustment time of workload evaluation and improve the accuracy of workload estimation. The QoS management will be able to take account of all queries and allocate CPU and memory resources fairly among queries. In addition, we would like to make several enhancements to scheduling and load management using feedback control theory.

References

[1] L. Golab, M.T. Ozsu. Issues in Data Stream Management. SIGMOD Record, 2003, 32(2):5-14.

[2] Y. Wei, V. Prasad, S.H. Son, J. Stankovic. Prediction-based QoS Management for Real-time Data Stream. IEEE Real-Time Systems Symposium (RTSS'06), Dec. 2006

[3] J.A. Daniel, D. Carney, U. Cetintemel, M. Cherniack, C. Convey, S. Lee, M. Stonebraker, N. Tatbul, S. Zdonik. Aurora: A new model and architecture for data stream management. Journal on VLDB, 2003,12(2):120-139.

[4] A. Arasu, B. Babcock, S. Babu, J. Cieslewicz, M. Datar, K. Ito, R. Motwani, U. Srivastava, J. Widom. STREAM: The Stanford Stream Data Manager, IEEE Data Engineering Bulletin,2003,26(1). http://dbpubs.stanford.edu: 8090/pub/2003-21.

[5] T. M. Ghazalie, T. P. Baker. Aperiodic servers in a deadline scheduling environment. In Journal of Real-Time Systems, 1995, 9(1): 31-67

[6] M. Spuri, G. Buttazzo. Efficient Aperiodic Service under Earliest Deadline Scheduling. In Proc. Of Real-time Systems Symposium,1994:2-11

[7] M. Spuri, G. Buttazzo. Scheduling Aperiodic Tasks in Dynamic Priority Systems. Journal of Real-Time Systems, 1996, 10(2): 179-210

Application of Feedback Control Real-Time Scheduling to Synthetic Aperture Radar

Hongan Wang, Yong Fu, Ying Qiao
Intelligence Engineering Lab, Institute of Software, Chinese Academy of Sciences
Beijing, 100080, China

***Abstract*-SAR (Synthetic Aperture Radar) is a radar system that uses remote sensing techniques to obtain high-resolution radar images of the earth's surface. SAR imaging system is a real-time heterogeneous system in which the real-time scheduling is a vital issue. In this paper, we present a feedback control scheduling algorithm for SAR imaging system (FC-SAR) to simultaneously maintain deadline guarantee of the hard real-time tasks in SAR system and improve CPU utilization of soft real-time tasks such as image display. Furthermore, the proposed algorithm is evaluated by experiments simulating the SAR imaging system.**

I. INTRODUCTION

SAR (Synthetic Aperture Radar) is a radar system that produces high-resolution, all-weather images of the observed terrain using signal processing techniques [1,2]. It is used to identify man-made objects on the ground or in the air [3]. The radar is placed on moving object, such as airplane or satellite [4] to transmit pluses and receives the echoed signals [5]. SAR system has been widely used in many fields such as homeland defense, the exploration and flood monitoring/evaluation etc. Plane-based SAR imaging system receives the raw data via the radar [6] mounted on the plane, and then processes these data to form images. The system also records and displays these processed data to support the online monitoring and off-line analysis. Furthermore, all above procedures are performed in a real-time manner. Thus, how to guarantee deadlines of all tasks in the system is a challengeable issue. The real-time scheduling provides an elementary way to ensure real-time properties of a planed-based SAR imaging system.

However, few works have been done on developing dedicated scheduling algorithm for SAR imaging system. Moreover, traditional scheduling algorithm in real-time system can not handle fluctuation of workload in the SAR system, for example, when displaying terrain with rich features, the image display tasks consume more CPU utilization than the case of plain terrain. Therefore, more flexible real-time scheduling is required to meet the needs of the SAR system.

In this paper, we present a Feedback Control scheduling algorithm on SAR (FC-SAR) which features a feedback controller designed by control theory. FC-SAR schedules hard and soft real-time tasks together in the system through adaptive resource reservation. CBS [8] is chosen for resource reservation of soft real-time tasks while the controller monitors the soft real-time tasks in the SAR system and adjusts these tasks to accommodate the utilization allocation. This algorithm not only maintains temporal guarantee of the hard real-time tasks but also improves CPU utilization of soft real-time tasks. Unlike previous work of adaptive resource reservation [11], both utilization and fairness control between different real-time tasks are considered in FC-SAR.

The rest of the paper is organized as follows: section 2 addresses the model of a SAR imaging system; section 3 presents the FC-SAR algorithm and controller design; Section 4 evaluates the proposed scheduling algorithm via the experiments simulating the SAR imaging system; Conclusions and future works are stated in section 5.

II. MODELING SAR IMAGING SYSTEM

The SAR imaging system is composed of a main system and several sub-systems including input board, range dimension FDC board, CTM (Corner Turn Module) board, azimuth dimension FDC board and ICTM (Inverse Corner Turn Module) board. The main system includes the following functional modules, i.e., device monitoring, real-time data receiving, real-time data storage, real-time data visualizing, GPS information processing, geographical information processing, real-time diagnose and record, system parameter management and real-time imaging simulation [6]. We denote these tasks as $\{T_i | 1<i<10\}$. A SAR imaging system is in fact a real-time heterogeneous system composed of a set of multiprocessors with different speed. There are 6 processors in the system. They are denoted as P1, P2, P3, P4, P5, P6. Each of first five processors (P1, P2, P3, P4 and P5) has one task (T1, T2, T3, T4 and T5) while P6 has other five tasks (T5, T6, T7, T8, T9 and T10). The independencies between tasks are as follows: (Here, $T_x \prec T_y$ means that task T_x must precede task T_y.)

$$T_1 \prec T_2 \prec T_3 \prec T_4 \prec T_5 \prec T_6 \prec T_7, \quad T_6 \prec T_9, \quad and \quad T_6 \prec T_{10} \tag{1}$$

All these tasks are either hard real-time or soft real-time. Each task T in the SAR imaging system is described as a tuple $(a_T, r_T, P_T, D_T, v_T, E_T)$. Here, a_T is T's arrival time and r_T is T's ready time. P_T is the interval between two instances of task T. D_T denotes T's deadline. v_T is the number of T's different logic versions. E_T represents T's maximum execution time. For hard real-time tasks, E_T is a vector denoted by $(e_T^1, e_T^2, ..., e_T^m)$, where e_T^j $(j = 1,...,m)$ is the maximum execution time of task when it executes on processor p_j. For soft real-time tasks, E_T is a matrix denoted by $\{e_T^{ij}\}$ $(i = 1,...,v_T; j = 1,...,m)$, where e_T^{ij} is

the maximum execution time of T's logic version i when it executes on processor p_j. Furthermore, for each $j(j=1,...,m)$, $e_T^{1j} \le e_T^{2j} \le ... \le e_T^{vrj}$.

Thus, we develop two different models corresponding to the behavior of the SAR system. First, we model the dynamics of utilization. Since only soft real-time tasks can be reallocated with the utilization, the model of utilization involves the utilization of soft real-time tasks T9 and T10. The utilization model can be formulated as:

$$u(k+1) = u(k) + g_u r_u(k), \tag{2}$$

where $u(k)$ is the sum of utilization of T9 and T10 at sampling time k, g_u is the utilization gain and $r(k)$ is estimation of utilization change. Secondly, the fairness model is considered. Let the utilization of display task, T9, be $u_d(k)$ and geography information processing task, and T10, be $u_g(k)$. We denote the utilization ratio as $c_f(k) = u_d(k)/u_g(k)$. When the utilization of each soft real-time task changes the ratio between them also varied accordingly. The fairness control needs to maintain such ratio as a constant. However the ratio is subject to the constraint $u(k) = u_d(k) + u_g(k)$. Like utilization modeling, we can also model the ratio of the utilization as:

$$c_f(k+1) = c_f(k) + g_f r_f(k), \tag{3}$$

where g_r is the gain of ratio change and $r_f(k)$ is the controller output. It is noted that the utilization gain g_u and g_r may change in the execution. For robustness of the closed-loop system, the maximum values are chosen as gains. For simplicity, in our approach we assign them as unity, that is, $g_u = g_r = 1$.

III. Feedback Control Scheduling algorithm

In this section based on the model aforementioned we present the feedback control scheduling algorithm for SAR imaging system (FC-SAR). Essentially, our algorithm ensures the SAR system to avoid being overloaded to crash and maintaining the utilization of the processors to improve system's efficiency.

According to the system task model addressed in previous section, the scheduling for SAR system can be formulated as a mixed real-time scheduling problem, i.e., the hard real-time tasks and soft real-time tasks are scheduled together. For hard real-time tasks, T1, T2, T3, T4 and T5, there is no need to consider the scheduling of them since they are executed in dedicated processors (P1, P2, P3, P4 and P5). Thus, for SAR system, we only consider the real-time scheduling for tasks in processor P6. When scheduling mixed real-time tasks in processor P6, three aspects should be considered: schedulibility, overload protection and fairness. First, the schedulibility of hard real-time tasks, such as data storage, must be guaranteed to avoid missing their deadlines. Secondly the overall utilization of the processor, including hard and soft real-time tasks utilization, should keep close to the predefined set point so that the processor could prevent crash caused by transient workload. Finally, the two soft real-time tasks, display task and geography information processing task, need to run fairly. It is not preferred that one task, for example display task, consumes all processor resources to achieve high service quality while another task, geography information processing task, can not acquire any portion of computation time to run. Based on these three requisitions, we can formulate the scheduling problem in SAR system as: Given a set of hard real-time tasks {Thi | 1<i<n} and a set of soft real-time tasks {Tsi | 1<i<n} running on a processor P, design an adaptive scheduling algorithm to maintain schedulibility of the hard real-time tasks Thi while keeping the utilization of the processor and fairness among these soft real-time tasks Tsi close to their set points.

We design a feedback scheduling algorithm with the structure illustrated in Figure 1 to address the adaptive mixed real-time scheduling problem. Two control loops are involved in this scheduling algorithm. First control loop adjusts utilization of the processor. Specifically, in this loop, we control the portion of the processor's utilization consumed by the soft real-time tasks, display and geography information processing tasks in SAR system. Thereby the controller of the first loop is called utilization controller. The controller of the second control loop is a fairness controller which makes the display and geography information processing tasks run fairly. We use a ratio of tasks' utilization as the set point of fairness control. In addition, fairness control depends on the output of the utilization control. Fairness control always changes the utilization ratio between display and geography information processing tasks in CBS server [8] based on utilization controller output. The architecture of overall scheduling algorithm is shown in Figure 1.

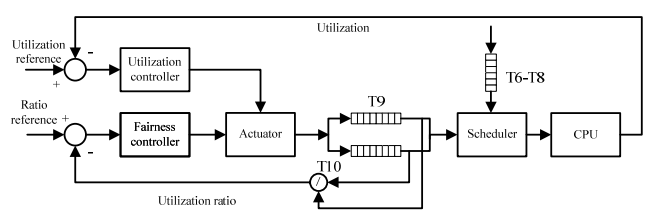


Figure 1. Architecture of FC-SAR

To design utilization and fairness controller, we first transform equation (2) and (3) from the time domain model to Z domain in which the model can be accommodated to be analyzed by control theory [9]. Based on derived dynamic models, we can design PI controller to control utilization and fairness of SAR tasks which includes Proportional and Integral term.

$$u_p(k) = u_p(k-1) + k_p e(k) - k_p e(k-1) - k_i T_s e(k), \tag{4}$$

where $u_p(k)$ is the output of the PID controller, k_p and k_i are coefficient of proportional and integral terms respectively and Ts is sampling interval. The general procedure to determine k_p and k_i is firstly to set the performance index, such as settling time, overshoot and so on, and then to solve the coefficients according to these

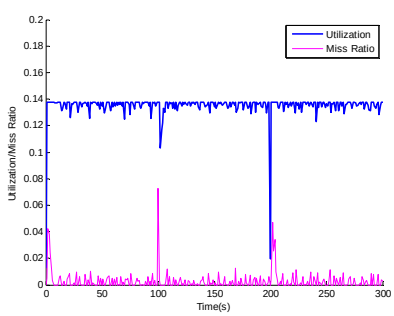
performance index through root locus method. Details of design of PI controller can be found in [9].

IV. EXPERIMENTS

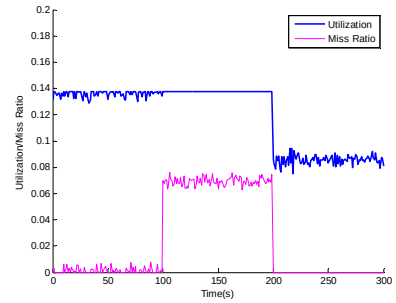
We employ RTSim [10], a real-time simulation library written in C++, and Matlab Control Toolbox [12] to build the simulation environment of Feedback Control Scheduling Algorithm for SAR system with Linux 2.6 and GCC 3.2.4. We implement the CBS scheduler, workload generator and utilization monitor through RTSim. The utilization controller and fairness controller are developed through Matlab Control Toolbox. In the experiment running, the controller and other components communicate through shared memory. For the SAR system, there are totally 10 real-time tasks but we only consider T6-T10 as described in SAR system model. T6, T7 and T8 are hard real-time tasks that totally use 71.2% CPU utilization. T9 and T10 are soft real-time tasks that are allocated with 13.8% CPU utilization. Note that there is 15% CPU utilization unused for overload protection. The utilization ratio of task T9, image displaying, and task T10, geography information processing, is 2:1, that is, normally T9 consumes 9.2% CPU utilization and T10 consumes 4.6% CPU utilization. We denote etf as estimated factor of execution time of the task. The actual execution time of the task is the product of estimated execution time and etf. The utilization of the real-time tasks can be adjusted by changing etf.

FC-SAR employs two PI controllers that control utilization and fairness of the system respectively. The controller of utilization has parameters $k_p = 1.35$ and $k_i = 0.05$ while fairness controller has $k_p = 1.1$ and $k_i = 4.4$. In addition, we also develop a baseline algorithm which is denoted as OPEN-SAR. In the baseline algorithm, only CBS scheduler is adopted and the utilization as well as its ratio can not be adjusted in running but is assigned initially.

We first compare the performance of FC-SAR and OPEN-SAR under utilization variation. In this experiment both algorithms are stress tested under varied workload. The experiment runs 300s. In the first 100s, the utilization of T9 and T10 is 13.8%，which is the normal utilization. At 100s, the utilization increases abruptly by changing etf from 1 to 1.5. Thus the utilization at this time is 1.5 times of the normal case. At 200s, we decrease the utilization by changing etf to 0.8, which means that the utilization declines 20% than the normal utilization.



(a) FC-SAR

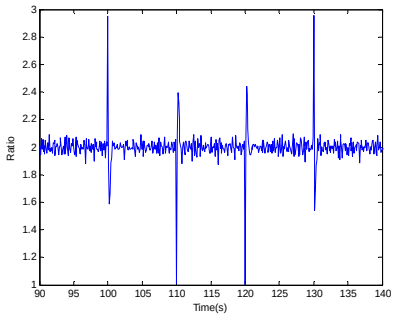


(b) OPEN-SAR

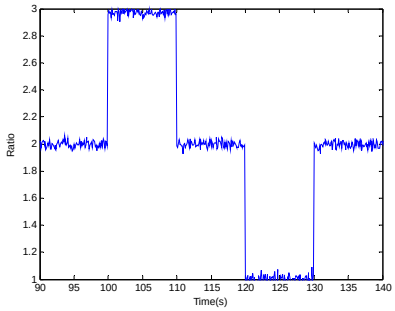
Figure 2. Utilization of T9 and T10

Figure 2(a) shows the performance of FC-SAR under utilization variation. At 100s, although the utilization increases significantly, the utilization maintains at the initial value 13.8% after 10s adjustment. At 200s, the utilization drops greatly caused by decreasing etf. In this case, the utilization also experiences a decline, but after 7s adjustment the utilization restores the initial value and maintains it in the rest time. Moreover, in the whole running, miss ratio keeps small except some points where the adjustment is invoked. From this result, we can conclude the utilization controller can maintain the utilization allocation even under the significant variation. As shown in Figure 2(b), OPEN-SAR, as a sharp comparison, can only keep the utilization when there is no disturbance at the first 100s. At 200s the miss ratio increases greatly and at 300s the utilization drops below the normal level which leaves some spare CPU utilization.

Meanwhile we also test the performance of FC-SAR under the case of the utilization and its variation together. Thus, the test profile includes two portions. First, we use the same utilization variation profile like the experiment of utilization variation. Then for utilization ratio, we change it by adjusting T9's etf at each time utilization varies. For example, at 100s in running, the utilization ratio increases to 3.0 and recovers. Then at 120s the ratio decreases to 1.0 for 10s. In other points of utilization variations, the ratios change in the same way.



(a) FC-SAR



(b) OPEN-SAR

Figure 3 Utilization ratio of T9 and T10

Figure 3(a) shows FC-SAR fairness control under utilization ratio varies from 90s to 140s. At other times there are the same results. In whole running the utilization ratio keeps constant in FC-SAR except some points. At 100s, the ratio increases to 3.0 but the controller adjusts ratio by allocating more reservation to T10 while the reservation of T9 is decreased. This procedure is inversed at 110s where the reservation assigned to T9 is greater than T10. At other times controller can also keep the utilization ratio constant. In contrast, OPEN-SAR, illustrated in Figure 3(b), varies its utilization ratio according to the test profile, that is, it can not handle the utilization ratio variation.

V. CONCLUSION

In this paper, we present a feedback control scheduling algorithm for SAR system, called FC-SAR. The tasks in SAR system are analyzed. Based on the analysis, the task model is developed. Next the FC-SAR architecture is described and the outline of controllers design is presented. Finally we evaluate the proposed algorithm by experiments. The results show that our approach not only maintains the utilization under workload variation but also the ratio between soft real-time tasks.

ACKNOWLEDGMENT

Authors are supported in part by National Natural Science Foundation of China (Grant No. 60373055, 60542005 and 60374058).

REFERENCES

[1] D. R. Wehner, *High resolution radar*, Second Edition, Artech House, 1995.

[2] J. C. Curlander and R. N. McDonough, *Synthetic Aperture Radar-Systems and Signal Processing*, Wiley, 1991.

[3] B.Zuerndorfer, G.A. Shaw, "SAR Processing for RASSP Application", in *Proceedings of 1st Annual RASSP Conference*, Arlington, VA.,1994.

[4] B.Dawidowica, et al, "Space synthetic aperture radar system analysis," available at: http://www.stec2005.space.aau.dk/getpdf.php?id=53K.

[5] J. Suh, et al, "Parallel implementation of synthetic aperture radar on high performance computing platforms," in Proceedings of 3rd International Conference on Algorithms and Architectures for Parallel Processing. 1997.

[6] K. Wang, et al, 2003. "Study of Real-time imaging systems," *Journal of Computer Research and Development*, Vol.40, No.1, pp.26-32.

[7] C. Lu, et al, "Feedback control real-time scheduling: Framework, modeling, and algorithms." *Real-Time Systems*, Vol. 23. No (1/2), pp 85-126, 2002.

[8] Luca Abeni, et al, "Analysis of a Reservation-Based Feedback Scheduler," in *Proceedings of the Real Time Systems Symposium (RTSS 2002)*, Austin, Texas, 2002.

[9] Gene F. Franklin, et al, 1998. *Digital Control of Dynamic Systems*, Third Ed., Addison Wesley Longman.

[10] Cesare Bartolini and Giuseppe Lipari, *RTsim*, available at: http://rtsim.sssup.it/

[11] L. Abeni, G.Buttazzo, "Adaptive Bandwidth Reservation for Multimedia Computing", in *Sixth International Conference on Real-Time Computing Systems and Applications (RTCSA'99)*, pp.70, 1999

[12]Mathworks, *Matlab Control Toolbox*, available at: http://www.mathworks.com

Integration of a flexible network in a resource contracting framework*

R. Marau, L. Almeida, P. Pedreiras
DETI/IEETA
Universidade de Aveiro, 3810-193 Aveiro, Portugal
{marau,lda,pedreiras}@det.ua.pt

M. González Harbour, D. Sangorrín
Universidad de Cantabria
39005 - Santander, Spain
{mgh,daniel.sangorrin}@unican.es

J. L. Medina
CEA LIST, Boîte 94
F-91191, Gif-sur-Yvette, France
julio.medina@cea.fr

***Abstract*— In this paper we overview the integration of a framework that generically manages the system resources in the form of contracts, namely the FRESCOR framework, with a flexible network resource. We describe how a network resource, namely FTT-SE, supports the FRESCOR framework services and, likewise, how the network services are made available to the application through the contracting framework.**

I. Introduction

Networked Embedded Systems (NES) were originally associated with industrial supervision and control applications, which employed simple sensors, actuators and controllers. However, a steep evolution in this application domain is being experienced, pushed by the growing number of sensors and overall complexity present at the plant level. As an example, the use of imaging sensors, both for supervision and control purposes, is spreading widely in classes of applications such as mobile robotics, traffic control and assembly lines inspection. Consequently, the sensors become inherently more complex, so as the flows of information exchanged at the cell and plant levels, integrating periodic and aperiodic flows of short and large data, some of multimedia nature, with considerable variability during run-time.

The new demands and increased complexity posed by these applications pushed the development on new techniques and design methodologies. Two key aspects in this regard are the complexity management and the resource management. Complexity management is being addressed by the adoption of adequate middleware layers in NES (e.g. CORBA and Java RMI, DCOM, etc) [1], which facilitate distribution, aiming at transparent interaction mechanisms between objects, components or applications. Regarding the resource management (e.g. CPU, memory, network, energy, etc) several approaches have been proposed recently, aiming at fulfilling the needs of those emerging applications in aspects like dynamic configuration and QoS management, support for new and more efficient scheduling techniques, etc.

The FIRST Scheduling Framework (FSF) [2] provides a high-level abstraction for real time resource schedulers while maintaining predictability and performance efficiency. It provides a homogeneous interface so it can be used in different platform architectures. This framework was initially designed to cope with the application needs for processor and network management, although with some limitations in the latter. The Framework for Real-time Embedded Systems based on COntRACTS (FRESCOR) aims at extending the FSF framework for multi-resource reservation, comprising various classes of resources commonly found in NES applications.

Regarding the communication subsystem, the recently proposed Flexible Time triggered communication over Switched Ethernet (FTT-SE) [3] provides flexible and deterministic real-time communication services combined with dynamic Quality-of-Service (QoS) management. This protocol has been developed specifically to address the requirements presented by the emerging applications referred above, combining realtime requirements with a high degree of adaptability. It looks, thus, a natural network candidate for inclusion in a contracting framework, to efficiently exploit and enrich the high level of flexibility that it already offers.

This paper analyzes the integration of the FTT-SE protocol in the scope of the contract model framework and describes how this integration can be performed. Using the former FSF framework, a network resource implementation exists for the Real-Time Ethernet Protocol (RT-EP) [4]. However, for FTT-SE, a different architecture must be used due to the different data link layer features of the two network protocols.

The remainder of the paper presents an overview of FRESCOR and FTT-SE background in sections II and III, discusses the integration of FTT-SE under FRESCOR in section IV, details the contracting procedure in section V and shows conclusions and on-going work in section VI.

II. FRESCOR background

The FRESCOR framework is based on the notion of contracts between the application and the system resources manager. These contracts are created, managed and enforced by a *Contract Layer*, which assures that sufficient resources capacity is available. The framework is divided in modules that allow abstracting away the specificities of the resources typically found in NES. Of particular interest to this work are the *Core* module, which contains the basic contract information that must exist in all contracted resources, the *Spare capacity*

* This work has been funded in part by the Plan Nacional de I+D+I of the Spanish Government under grant TIC2005-08665-C03 (THREAD project), and by the European Union's Sixth Framework Programme under contracts FP6/2005/IST/5-034026 (FRESCOR project) and IST-004527 (ARTIST2 NoE). This work reflects only the author's views; the EU is not liable for any use that may be made of the information contained herein.

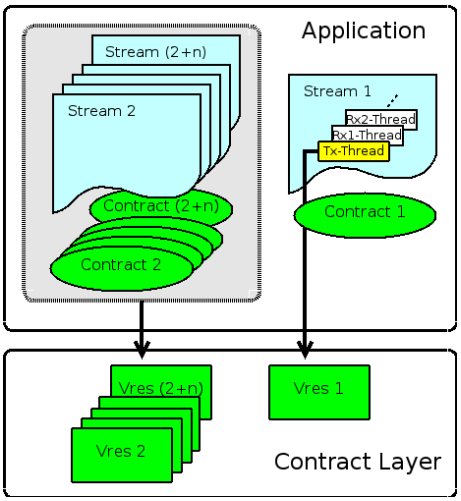


Fig. 1. Contract layer

module, which defines how the application may take advantage of currently unused resource capacity, and the *Distribution* module that deals with issues of distributed applications.

The contract parameters associated to these modules are referred in table I. The *Contract id* is a unique identifier inside one resource (here a network resource) that distinguishes the contracts globally, the *Resource type* and the *Resource id* inform about the kind of, and which resource the contract refers to; the *Minimum budget* and *Maximum period* define the minimum resource capacity required by the application, *Importance* and *Weight* allow prioritizing the contracts associated to one resource when distributing spare bandwidth.

TABLE I
CONTRACT PARAMETERS

Core	Contract id Resource type Resource id Minimum budget Maximum period Deadline
Spare capacity	Granularity Maximum budget Minimum period Utilization set Importance & Weight Stability
Distribution	Protocol dependent information

A. FRESCOR application model

Within FRESCOR, the application is a global entity enclosing several *Threads* that access the system resources by means of *Virtual resources (Vres)* residing in the *Contract Layer*. Each *Vres* holds one associated contract. In distributed applications, this layer is also distributed and comprises the management of both, the processors and the network resources. Several network contracts may be atomically negotiated as a group, ensuring that they are either all accepted, or all rejected as a whole. The contract negotiation for a given stream is initiated at the sending node. If successful, the associated *Vres* may be created in that node, or in some other node that may be in charge of scheduling the network traffic.

In a distributed application, the FRESCOR framework considers the network as just another resource that is managed by contracts. Each of these contracts refers to one Stream through which application threads exchange messages. Each stream has a unique identifier mapped on the *Contract id* and used as a stream descriptor by the application when accessing the *Contract Layer*. Figure 1 shows the FRESCOR distributed application model with a network resource highlighting just the network contracts under two possible situations: when a contract *Virtual resource* is created in a transmitter node (Stream 1), or in a contract group situation (Streams 2 .. (2+n)).

III. FTT-SE BASICS

The FTT-SE protocol was designed to support hard real time applications using Switched Ethernet networks in a flexible but predictable manner. It supports both time-triggered and event-triggered communication semantics in two well defined and temporally isolated communication subsystems, namely the Synchronous and the Asynchronous Messaging Systems, SMS and AMS, respectively [3].

The protocol is based on a master-slave paradigm, in which a master controls the access to the network by the remaining nodes in the system (slaves). This allows managing the communication load submitted to the switch at each instant, thus preventing overloads and maintaining a predictable behavior.

However, the master-slave control is carried out on a cyclic basis, i.e., the master sends out one control message per cycle, only, indicating which messages must be transmitted therein. The cycle is called Elementary Cycle (EC) and it is triggered by the master control message called Trigger Message (TM), which is broadcast to all nodes.

The master holds the System Requirements DataBase (SRDB) that contains, among other information, the current communication requirements, and builds the traffic schedules for each EC on-line. Changes to these requirements can be carried out at run-time and are subject to an admission control that guarantees continued timely communication.

The requirements tables for synchronous and asynchronous message streams hold the following parameters, respectively:

$$SM_i\left(C_i, D_i, T_i, O_i, S_i, \{R_i^1..R_i^{k_i}\}\right) \quad , i = 1..N$$
$$AM_i\left(C_i, D_i, mit_i, S_i, \{R_i^1..R_i^{k_i}\}\right) \quad , i = 1..N$$

C_i is message i transmission time, D_i is its deadline, T_i the period, mit_i is the minimmum inter-transmission time and O_i the offset. Both D_i, T_i/mit_i and O_i are expressed as integer numbers of ECs. S_i is the sender node and $\{R_i^1..R_i^{k_i}\}$ is the set of k_i receivers for this message stream.

Finally, FTT-SE also provides mechanisms to synchronize the application threads with their periodic communications. This synchronization plus the use of offsets is the basis for the

so-called network-centric approach to the design of distributed systems, which facilitates the synchronization of threads in different nodes and the reduction of end-to-end delays.

IV. Integration of FTT-SE under FRESCOR

One important goal of the integration was to keep the performance level of the FTT-SE real-time communication services throughout the abstraction process associated with the creation of a middleware. The FRESCOR framework was selected as middleware because it facilitates achieving this goal given its simple and generic application interface and real time concerns. Moreover, its modular flexibility extends the resource management to an holistic application perspective which reduces the project design complexity.

This section describes how the FTT-SE protocol can be integrated as a FRESCOR pluggable resource. This integration allows abstracting away the network access from the application perspective and it defines two sets of services, the negotiation procedure and the communication access primitives. The former handles the contract (re-)negotiations requested by the application to change the Stream properties provided by the network resource. Once a contract is accepted the application may start using the respective communication Stream through the services provided by the latter.

As referred before, the FRESCOR modules used when integrating FTT-SE are the *Core*, *Spare capacity* and *Distribution* modules. Each of these takes its role in the contract negotiation with the parameters described in table I. The *Distribution* module needs specific attention since it contains network protocol dependent information. For the RT-EP distributed resource no special parameters were required, thus the *Core* parameters *Minimum period* and *Maximum Budget* were enough to carry out the network management. However, FTT-SE includes several features that require appropriate configuration and management, which must thus be included in this module:

- Two communication triggering models are provided by the network, namely time-driven and event-driven, which must be defined in the contract specification;
- To take advantage of the multiple forwarding paths in the network switch and still provide real-time guarantees, the contract must include the specific switching path used by each channel, i.e., the identification of the producer and consumers involved and the switch ports they are connected to;
- To exploit the explicit synchronization between time-driven channels supported by the network, the contracts, or contract groups, must include two additional parameters, one describing the *Contract id* of the channel to synchronize with, and another specifying the desired synchronization offset. If no synchronization is specified the channel is considered as float and the network will arbitrarily allocate relative offsets to the contract;

The integrated FRESCOR / FTT-SE architecture is sketched in Fig. 2. The network contract negotiation procedure is centralized in the FTT master node and it is handled by the *Master Contract Layer*. This is a natural choice since the FTT master centralizes all the real time requirements of current communication channels and controls the network access. The contracts are then reflected on the involved slave nodes. The *Slave Contract Layer* handles network contract requests, holds local contract copies and makes them available to the application threads. This centralized approach is substantially different from the one taken in the RT-EP implementation, where the negotiation procedure is fully distributed requiring every node to keep a consistent replica of all running contracts.

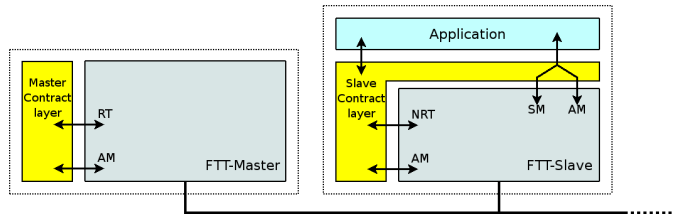


Fig. 2. Architecture overview

A. The FRESCOR / FTT-SE interface

The communication between the *Master Contract Layer* and the *Slaves Contract Layer*, both for conveying negotiation requests and publishing the contract copies, uses permanent bi-directional channels between the master and each slave node in the network, implemented with FTT-SE asynchronous messages (AM).

The network contracts in the *Master Contract Layer* are reflected in the FTT Master, in its Requirements Table (RT). On the other side, the *Slave Contract Layer* keeps the copies of its contracts, reflecting them in the respective FTT Slave, in the Node Requirements Table (NRT). This layer also provides interfaces to the application, to negotiate contracts and to access the communication services, both synchronous (SM) and asynchronous (AM).

B. Supporting the application interface

Figure 3 shows the FRESCOR common resource interface and the objects involved in network contracts. The *Negotiation* service allows the system to establish the required resource reservations and, in this case, involves communication with

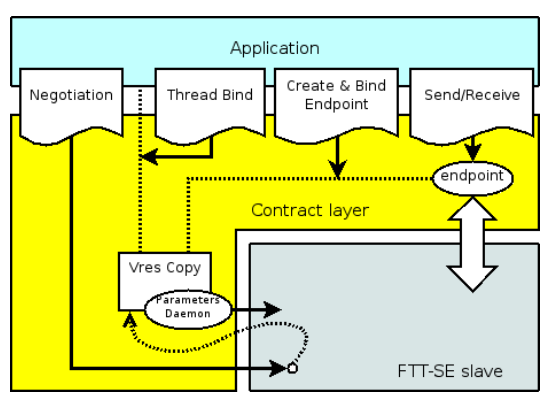


Fig. 3. FRESCOR interface for network contracts

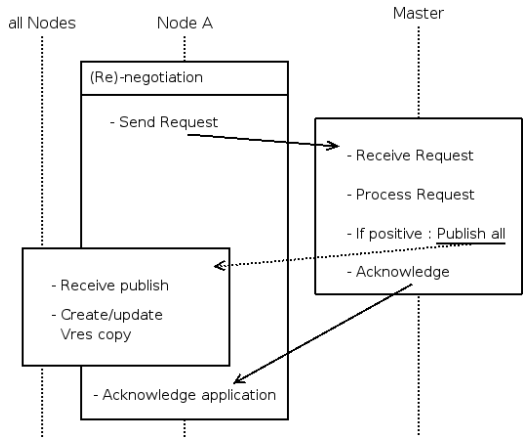


Fig. 4. Negotiation steps

the *Master Contract Layer*. Upon a negotiation success, the respective contract *Virtual resource(s)* is created/updated in the *Master Contract Layer* and the *Virtual resource copies* are created/updated in the *Slaves Contract Layer*. The *Thread Bind* allows associating an application thread with a contracted resource and provides access to the respective *Vres copy*. The access to the contracted resource, a Stream in this case, is made through an *endpoint*, which is created and bound to the *Vres* by the *Create & Bind Endpoint* services. Finally, the *Send/Receive* services allow the communication through the respective endpoint.

The network *Virtual resources* in the contract layer must be consistent with the communication parameters within FTT-SE so that the protocol actually enforces the contracted communication parameters with its control mechanisms. Therefore, a *parameters daemon* is used to keep such consistency.

V. Internals of the Contracting Procedure

The establishment of network contracts with FTT-SE, as referred before, requires an interaction between the *Slave Contract Layer* of the involved nodes and the *Master Contract Layer*. The process is triggered by the thread that manages the contract or the contract group and its sequence diagram is depicted in Fig. 4.

The request is enqueued in the *Master Contract Layer* until it can be processed (Fig. 5). At that point, it is removed from the requests queue and submitted for admission process, which involves the admission control in the FTT-SE master. If the contract is accepted, which may result in changes to other contracts, the master updates the FTT-SE internal structures and publishes all *Vres* that were updated. The respective slaves receive this information and update/create the respective *Vres copies*. Then, the master acknowledges the negotiation result.

VI. Concluding Remarks and On-going Work

The FRESCOR framework has been proposed recently to cope with the growing application complexity and interoperability requirements in embedded systems. The approach followed by FRESCOR allows abstracting the management

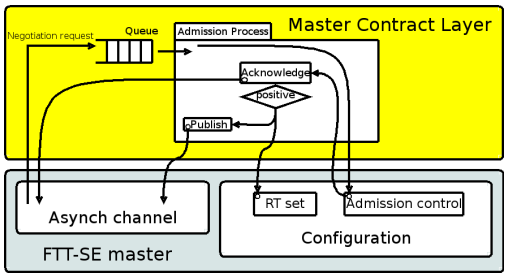


Fig. 5. Master contract negotiation procedure

of the application resources, which are accessed through a common interface based on contracts.

In this paper we discussed the integration of the FTT-SE real-time communication protocol within the FRESCOR contract layer framework. This framework efficiently exploits the FTT-SE natural ability for dynamically adapting the network resource usage while maintaining predictability. Another positive aspect in this symbiosis is that the interface given by the framework to the application can be commonly applied together with other shared resources of the system.

Previously, only the RT-EP network protocol had been integrated within the FSF/FRESCOR framework. Such protocol works over a shared medium with a priority based event-triggered messaging paradigm. The integration of FTT-SE brings in the features of a different network paradigm and topology, namely time-triggered and event-triggered communication over Switched Ethernet. We believe that the dynamism of FTT-SE will impact positively on the efficiency of network management in the FRESCOR framework. On the other hand, the abstraction provided by FRESCOR will benefit the FTT-SE protocol in terms of its usability and applications development.

Currently we are carrying out the temporal analysis of the negotiation process. In the near future we plan to apply the implementation herein described to an application that allows illustrating its flexibility and negotiation capabilities.

References

[1] N. Wang, D. Schmidt, K. Parameswaran, and M. Kircher, "Towards a Reflective Middleware Framework for QoS-enabled CORBA Component Model Applications," IEEE Distributed Systems Online special issue on Reflective Middleware (Vol. 2, No. 5), May 2001.

[2] M. Aldea *et al.*, "FSF: A Real-Time Scheduling Architecture Framework," in *12th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS'06)*. San Jose (CA, USA): IEEE, Apr. 2006, pp. 113–124.

[3] R. Marau, L. Almeida, and P. Pedreiras, "Enhancing real-time communication over COTS Ethernet switches," in *WFCS'06: IEEE International Workshop on Factory Communication Systems*, 27 June 2006, pp. 295–302.

[4] J. M. Martínez and M. G. Harbour, "RT-EP: A Fixed-Priority Real Time Communication Protocol over Standard Ethernet," in *10th International Conference on Reliable Software Technologies, Ada-Europe*. Springer, June 2005, pp. 180–195.

CheckerMode : A hybrid scheme for timing analysis of modern processor pipelines involving hardware/software interactions

Sibin Mohan and Frank Mueller
Dept. of Computer Science, Center for Embedded Systems Research,
North Carolina State University, Raleigh, NC 27695-7534, mueller@cs.ncsu.edu

Abstract

Real-time systems often require determinism to ensure that task deadlines are met. Schedulability analysis provides a firm basis to ensure that tasks deadlines are met, and for this, knowledge of bounds on worst-case execution times (WCET) of tasks is a critical piece of information. Static timing analysis derives these bounds on WCETs. A limiting factor for real-time systems design is the class of processors that may be used. Contemporary processors with their advanced architectural features, such as out-of-order execution, branch prediction, speculation, and prefetching, cannot be statically analyzed to obtain WCETs for tasks because these features introduce non-determinism to task execution, which can only be resolved at run-time. We introduce a new paradigm which proposes minor enhancements to modern processor architectures, which, on interaction with software modules, is able to obtain tight, accurate timing analysis results for modern processors. To the best of our knowledge, this method of hardware/software interactions to calculate WCET results for out-of-order processors is the first of its kind.

1. Introduction

Embedded systems are increasingly deployed in safety-critical applications and environments, such as avionics, power plants, automobiles, *etc.* The software used, in general, must be validated . This traditionally amounts to checking the correctness of the input/output relationship. Many such systems also impose timing constraints, which, if violated, may result in fallouts that are dangerous to the environment. Such systems are typically referred to as *real-time systems*. They impose timing constraints (”deadlines”) on the computation to ensure that necessary results are provided on time. The worst-case execution time (WCET) of each task is one critical piece of information required by real-time systems designers to verify that tasks meet their deadlines.

Static timing analysis [3–7, 9] provides bounds on the WCET. The *tighter* that these bounds are relative to the actual worst-case times, the better the value of the analysis. Of course, any tight bound has to be *safe* in that it must *never* underestimate the true WCET; it may only match or exceed it.

A serious handicap in performing static timing analysis is the complexity of modern processors and their functional units. Out of order (OOO) processing [8], branch prediction [10] *etc.* introduce non-determinism to task execution that cannot be resolved at compile time. Hence, designers of real-time systems are often forced to use older, less complicated and inherently less powerful processors. In this paper, we attempt to bridge this gap by the use of the *CheckerMode* infrastructure.

We propose minor enhancements to the micro-architecture of future processors that will aid the processes of obtaining tight WCET bounds. A ”checker mode” is added to processors that will, on demand, capture varying details of the processor state (called “snapshots”). This information is communicated to a software module that stores the snapshots and also drives the execution of the processors along statically determined paths to capture accurate timing information for each of them. The snapshots are used to track back along the various execution paths and to restart along a different path if necessary. The execution times obtained for each of the paths is analyzed and combined by the software driver to calculate an accurate WCET for the entire module/program. Decisions on where to obtain snapshots, the details required for a snapshot, *etc.*, are made by the software ”driver”.

The CheckerMode concept (implemented on an enhanced SimpleScalar processor simulator [2]), widens the scope of processors that may be used in a real-time system. Contemporary processors with state-of-the-art functionality and performance may subsequently be used in a real-time system. We believe that this also changes the landscape for timing analysis as more accurate results can be obtained on modern pipelines without loss of functionality. To the best of our knowledge, this method is the first of its kind in using a hardware/software co-design technique to obtain accurate WCETs for modern, out-of-order processors.

This paper is organized as follows. Section 2 discusses the CheckerMode idea, while section 3 talks about the experimental setup and preliminary results. Section 4 summarizes the work.

2. CheckerMode

We use hardware/software interactions to perform WCET analysis of contemporary processors. We propose enhancements to embedded processors that, in addition to executing software normally (in “deployment” mode), are capable of executing in a novel *CheckerMode* that supports timing analysis. The CheckerMode provides cycle-accurate bounds on the WCET by assessing alternate execution paths in a program. In deployment mode, a processor executes along just one path following a conditional branch depending on input data. In CheckerMode a processor executes all alternate paths, one at a time, following each conditional branch in order to find the path with the largest execution time. Before the execution of each alternate path, the original execution context, named “snapshot” (caches, branch history tables etc.), is restored to correctly simulate the effect of alternations in isolation from one another. The timing information as well

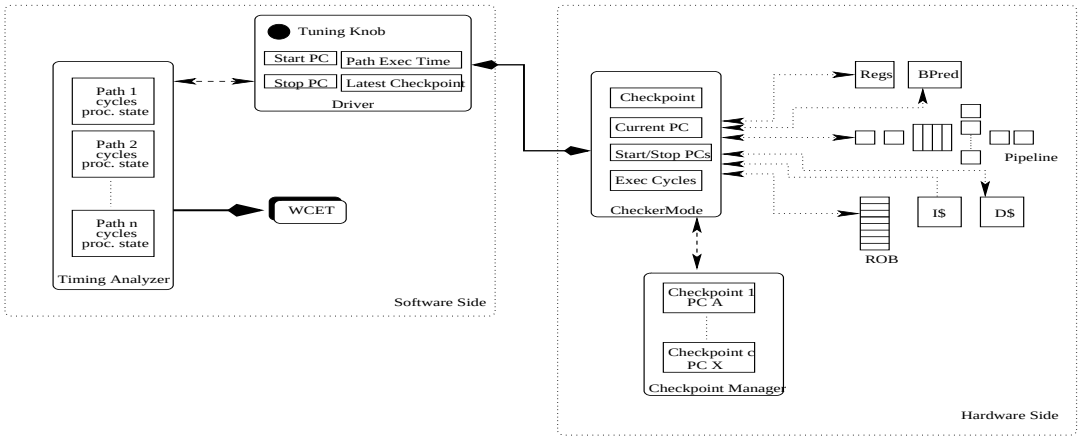


Figure 1. CheckerMode Design for High-Confidence WCET Analysis

as the "state" of the processor are combined when alternate paths join. The combination ("merge") is performed such that the state that results from the combination must not underestimate the execution time of the alternate paths, or even the future execution of the task. These low-level WCET results are propagated inter-procedurally in a bottom-up fashion until the WCET for an entire task has been computed.

We will represent input-dependent register values as "NaN" (not-a-number) values. Operations on unknown values are straightforward: if any input is unknown then the output is also unknown, even for condition codes at the bit level. A branch condition based on an unknown value then indicates a need to consider alternate paths. Conversely, concrete (known) values are evaluated as always, and input-invariant branches will result in timing of only the taken execution path. We will alter the semantics of execution (for instructions that depend on input-dependent or memory-loaded operations) in CheckerMode to include this NaN value. *E.g.*, addition will now be rewritten as:

$$r_{result} = \begin{cases} \mathbf{NaN} & if\ r_a = \mathbf{NaN} \bigvee r_b = \mathbf{NaN} \\ r_a + r_b & otherwise \end{cases}$$

Hence, any operation with NaN as one of the operands will result in NaN (unless the result is independent of that particular operand, for *e.g.*, multiplication with 0 will always result in 0).

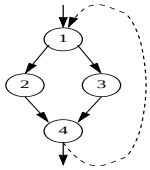
2.1. Overview of the framework

The hardware-supported CheckerMode is complemented by software analysis to govern execution (Figure 1). The **analysis controller (or driver)** steers execution along distinct execution paths, *i.e.*, it indicates which direction a branch along the path should take till all paths have been traversed. The timing information and the states of the processor obtained for each possible path are then used by a "timing analyzer" to obtain the WCET for the entire task (or even certain code sections).

Processor enhancements: The embedded hardware is also enhanced to support access to the unit-level context of hardware resources, which can be saved and restored. The analysis phase restores a context prior to examining a path and then saves the newly composed context at the end of a path, together with the timing of the path. The novel **CheckerMode** unit of the processor supports the following functions (right-hand side of Figure 1): *(a)* Capture *snapshots* of the processor state and communicate them to the software controller. Snapshots capture the current state of the pipeline, functional units, caches, ROB, etc. *(b)* Reset the processor to a previously saved state. The state of the pipeline, caches, functional units, *etc.*, is overwritten with information from the stored checkpoint. *(c)* Start and stop execution between arbitrarily provided program counter (PC) values. This includes support to calculate the number of cycles elapsed between the execution of the given start and stop PCs. The CheckerMode tracks the execution time for a given path (delineated by start and stop PCs) and is controlled by the driver on the software side.

Software controller: The left-hand side of Figure 1 illustrates the various components that make up the software side of the design. It consists of the following components: *(a)* **Timing Analyzer (TA):** breaks down the task code into a control-flow graph (CFG) and then extracts path information from it. It is able to determine the start of alternate execution flows – points where snapshots must be obtained. It also provides the start and stop PCs to the driver and obtains the WCET and processor state for that particular path from the driver.*(b)* **Checkpoint Manager (CM):** maintains various snapshots that have been captured as well as the PCs at which they were obtained. CM abstractions can be integrated into the processor as depicted in Fig. 1, or, alternatively, into the driver within the software controller.*(c)* **Driver:** controls the hardware side of the system. It instructs the hardware on when to start and stop execution, when snapshots must be captured, and when the state of a processor must be reset to a given snapshot.

The input to the TA is the executable of a task, which is then converted to internal representations. Start and stop PCs provided by the TA encapsulate a single path. The TA, the driver, and the CM interact to decide which checkpoint corresponds to which path, which PC, *etc.*, and thereby control program execution. The TA is responsible for obtaining the final WCET for the entire program as well for various program segments (functions/scopes). It "combines" the information from various paths (execution time/pipeline



(a) CFG

Path	SimIO	delta	SupIO	delta	OOO	delta
BB1	82	BB1-BB0=56	66	BB1-BB 0=4	47	BB1-BB0=1
BB1,2	114	BB1,2-BB1=32	94	BB1,2-BB1=28	59	BB1,2-BB1=12
BB1,3	241	BB1,3-BB1=159	131	BB1,3-BB1=65	92	BB1,3-BB1=45
BB1,2,4	151	BB1,2,4-BB1,2=37	97	BB1,2,4-BB1,2=3	61	BB1,2,4-BB2=2
BB1,3,4	278	BB1,3,4-BB1,3=37	134	BB1,3,4-BB1,3=3	94	BB1,3,4-BB1,3=2

(b)Measured Cycles for Aggregate Technique

Figure 2. Control Flow Graph and Measured Cycles for Aggregate Technique

state/*etc.*) for this purpose.

Driver / analysis controller and tuning: The driver is responsible for controlling processor operations. Besides directing the execution of the code on the pipeline, it relays instructions from the TA, such as when to capture/restore checkpoints. The driver represents the interface between the hardware and software components and provides reconfigurability in terms of the amount of information to capture for the pipeline state and the state of associated functional units. We propose to provide a **virtual "knob"** that will allow real-time systems designers to tune the analysis, thereby trading off accuracy with overhead. We intend to explore the full design space of tuning options to assess which processor state information is more vital for WCET accuracy (and analysis performance) than some others. The more information is checkpointed, the tighter and more accurate WCET values will be. Conversely, less information will lead to a looser and more conservative WCET bound. Of course, greater demands on the amount of information being captured will lead to a slower WCET analysis whereas less information speeds up the analysis.

Reducing analysis overheads: We can reduce the complexity of determining WCETs by **partial execution of loops** such that the analysis overhead is independent of the number of loop iterations. Using our prior approach of a fixpoint algorithm to determine a stable execution time for the loop body [1], we can steer loop executions such that paths of a loop body are repeatedly executed until a stable value is reached. The controller records the decaying execution times for each iteration up to the fixpoint using the hardware CheckerMode. When reaching the fixpoint, the WCET of the remainder of loop iterations up to the loop bound is calculated by a closed formula based on the fixpoint value. Typically, loops reach a fixpoint after only 2–4 iterations, which implies that this partial execution can reduce the overhead of WCET analysis significantly. Thus, the complexity of WCET analysis is **independent of the number of iterations**, *i.e.*, it does not depend on the actual execution time of analyzed code.

3. Experimental setup and Results

We have prototyped some of the key components of our design in the SimpleScalar processor simulator [2]. This cycle-accurate simulator can be configured for the various processor and branch prediction schemes mentioned in the previous section. Current enhancements include path-level timing capabilities and snapshot/restore of selected state information within the processor.

We used SimpleScalar in three configurations:*(a) Simple-IO (SimIO)* simulates a simple, in-order (IO) processor pipeline with pipeline width 1, instruction issue in program order); *(b) Superscalar-IO(SupIO)* with a pipeline width (from fetch to retire) of 16 and in-order instruction execution; *(c) Out-of-order (OOO)* execution with the same pipeline width as in Superscalar-IO.

Notice that instructions are retired in order, even for OOO. Execution time for paths is measured using four different techniques, extending a basic block (BB) to paths (sequences of consecutive BBs): *(a) Short* measures the execution time for a singular BB, starting from the time that **any** instruction in the BB moves into the *execute* stage of the pipeline and finishing when the last of instruction of the BB exits from the *retire* stage; *(b) Path-Short* captures the execution time for paths (concatenated BBs) using the "short" technique so that timing starts at the first BB and ends with the last BB in the path; *(c) Program-Aggregate* includes the time from the start of the execution (main function) to the end of a BB in the path being timed, starting when the first instruction in the main function is *fetched* and finishing when the last of the path exits from the *retire* stage; *(d) Path-Aggregate* captures the time for concatenated paths using the aggregate technique so that timing starts at the first BB and ends with the last BB of path.

The results obtained for the "short" and "path-short" techniques (numerical details omitted due to space) show that timings for the processor modes SimIO and SupIO accurately reflect the actual WCET bounds, both for single BBs and paths. However, the OOO results exceed those of SupIO, due to early out-of-order execution of some instructions in parallel to other instructions from prior BBs in the path. Even timing multiple BBs of a path in sequence does not alleviate this problem. In contrast, the "aggregate" technique (Figure 2(b)) reflects the time from instruction fetch (instead of execute), which addresses the above problem of early execution by some instructions. It shows a strict ordering of $SimIO \geq SupIO \geq OOO$, as expected by the amount of instruction parallelism, since time is measured from the first fetch of an instruction. The differences between paths ("delta") provide a bound on the number of cycles for the tail BB in the path excluding any pipeline overlap with prior BBs. Hence, these delta values can be used to assess the amount of cycles attributed to specific BBs. They also adhere to the same strict ordering. In general, such timings are only valid in the same execution context / path, *i.e.*, different BB sequences of one path may influence a subsequent BB in the control flow.

Our objective is to leverage path timings under the "path-aggregate" technique as a refinement to the "aggregate" technique discussed so far. Consider the construct depicted in

Path	SimIO			SupIO			OOO		
	$+$	o	δ	$+$	o	δ	$+$	o	δ
LLL	453	443	10	291	193	98	183	123	60
LLR	580	570	10	328	230	98	216	156	60
LRL	580	570	10	328	230	98	216	156	60
LRR	707	697	10	365	267	98	249	189	60
RLL	580	570	10	328	230	98	216	156	60
RLR	707	697	10	365	267	98	249	189	60
RRL	707	697	10	365	267	98	216	189	60
RRR	834	824	10	402	304	98	282	222	60

Table 1. Program-Aggregate Cycles (3 Iterations)

Figure 2(a) embedded within a loop (dashed vertex) such that consecutive executions of paths can be assessed. *E.g.*, within one iteration, the L-left (BB 1,2,4) and R-right (BB 1,3,4) paths are timed; within two iterations, concatenations of all permutations for these paths are timed (L-L/L-R/R-L/R-R); and so on for three and four iterations. Since this search space grows exponentially with the number of alternate paths and loop iterations, we propose to devise a bounded technique to limit the path space in depth and breadth.

Table 1 depicts the results for 3 iterations of this loop around the left (L) or right (R) paths for the 3 processor models. It also distinguishes path composition without overlap ($+$) and with overlap (o), where the former is equivalent to draining the pipeline while the latter captures continuous execution. The difference between the compositions is depicted as δ and indicates constant δ values for all processor models regardless of the paths executed. (D-caches are disabled here.) More significantly, early results within our experimentation environment indicate that 2-4 iterations generally suffice to reach a fix point. After that point, concatenation of another iteration results in a constant increase in cycles for this path that does not change for the remainder of the loop. For instance, a 2-path experiment (omitted here) resulted in exactly half the δ values of the 3-path experiment, which reinforces the claim about reaching a fix point.

4. Conclusion

We have outlined a "hybrid" mechanism for performing timing analysis that utilizes interactions between hardware and software. This "CheckerMode" concept provides the foundation to make contemporary processors predictable and analyzable. These higher-end microprocessors can safely be used in real-time systems. Current trends in microprocessor features indicate that our proposed hardware modifications are realistic [11]. Once fully implemented within the SimpleScalar simulator, the CheckerMode unit will have the ability to drive execution along given program paths and also capture and writeback processor state to/from snapshots. It will also be able to accurately gauge the execution time for a given program path. We believe this work will enhance the choices available to real-time systems designers. The CheckerMode concept will provide them with the ability to use current and future microprocessors in their systems and utilize a hybrid of static and dynamic timing techniques to validate WCETs.

References

[1] R. Arnold, F. Mueller, D. B. Whalley, and M. Harmon. Bounding worst-case instruction cache performance. In *IEEE Real-Time Systems Symposium*, pages 172–181, Dec. 1994.

[2] D. Burger, T. M. Austin, and S. Bennett. Evaluating future microprocessors: The simplescalar tool set. Technical Report CS-TR-1996-1308, University of Wisconsin, Madison, July 1996.

[3] C. A. Healy, R. D. Arnold, F. Mueller, D. Whalley, and M. G. Harmon. Bounding pipeline and instruction cache performance. *IEEE Transactions on Computers*, 48(1):53–70, Jan. 1999.

[4] S. Malik, M. Martonosi, and Y.-T. S. Li. Static timing analysis of embedded software. In *Proceedings of the 34th Conference on Design Automation (DAC-97)*, pages 147–152, NY, June 1997. ACM Press.

[5] S. Mohan, F. Mueller, W. Hawkins, M. Root, C. Healy, and D. Whalley. Parascale: Expoliting parametric timing analysis for real-time schedulers and dynamic voltage scaling. In *IEEE Real-Time Systems Symposium*, pages 233–242, Dec. 2005.

[6] S. Mohan, F. Mueller, D. Whalley, and C. Healy. Timing analysis for sensor network nodes of the atmega processor family. In *IEEE Real-Time Embedded Technology and Applications Symposium*, pages 405–414, Mar. 2005.

[7] F. Mueller. Timing analysis for instruction caches. *Real-Time Systems*, 18(2/3):209–239, May 2000.

[8] S. Palacharla, N. P. Jouppi, and J. E. Smith. Complexity-effective superscalar processors. In *ISCA*, pages 206–218, 1997.

[9] P. Puschner and C. Koza. Calculating the maximum execution time of real-time programs. *Real-Time Systems*, 1(2):159–176, Sept. 1989.

[10] Smith, J. E. A study of branch prediction strategies. In *Proc. 8th International Symposium on Computer Architecture*, pages 135–148, Minneapolis, 1981.

[11] B. Sprunt. Pentium 4 performance monitoring features. 2002.

Applying Colored Petri Nets to Develop Real Time Routines and Procedures

Fabricio Kenjy Morita, Osamu Saotome, Denis Silva Loubach and Giovani Dias
Electronics & Computer Engineering Dept.
Brazilian Aeronautical Institute of Technology
São José dos Campos, São Paulo, Brasil
e-mail: {fkmorita, osaotome}@ita.br, dloubach@ieee.org, giovani.d@uol.com.br

***Abstract*— On engineering world one can see many systems, softwares, and tools, but altogether it does not have all necessary resources to develop and debug a good real time embedded systems - RTES without too many errors.**

Petri Nets comes demonstrating to be an alternative and a good tool to help understanding complex systems. Among them the critical real time embedded systems are of our main concern.

This paper shows an academic case study of the Brazilian Aeronautical Institute of Technology (*Instituto Tecnológico de Aeronáutica - ITA*) graduate students. That work tackles both Petri Nets and RTES. Recently, the National Space Agency (*Agência Espacial Brasileira - AEB*) and the National Institute for Space Research (*Instituto Nacional de Pesquisas Espaciais - INPE*) had launched an student satellite program called ITASAT. The main program goal is to build up an experimental scientific small satellite.

ITASAT students are using Petri Nets tool to aid development and debugging some satellite subsystems like on-board data handling - OBDH and attitude and control determination - ACD.

This program aims to adopt a new software and hardware architecture. Real-Time Executive for Multiprocessor Systems - RTEMS was adopted as real time operating system and Blackfin processor was chosen as on-board computer test platform. RTEMS is still being tailored to be embedded in Blackfin. Then, Petri Nets were adopted as testing tool of those systems.

I. Introduction

The National Space Agency (*Agência Espacial Brasileira - AEB*) and Brazilian Institute for Space Research (*Instituto Nacional de Pesquisas Espaciais - INPE*) had launched an student satellite program so-called ITASAT [1]. That program involves some brazilian universities in order to make an experimental small satellite. Among the universities is found the Brazilian Aeronautical Institute of Technology (*Instituto Tecnológico de Aeronáutica - ITA*). Some of the main goals proposed to ITASAT Program, is the merging of the attitude control on-board computer, and the data handling computer in just one on-board computer. The use of new hardware and software technologies were comprised too.

So, it was defined starting with one essencial functionality to a satellite, the attitude control & data handling - ACDH.

The ACDH is a satellite subsystem which was programmed to verify and control satellite orbit. Thus it was necessary to study some motors, sensors, controls and programming languages to accomplish this subsystem and program system control.

To help developing the software Petri Nets were used. It is a graphical and mathematical modelling tool which can be applied to model a variety of systems types and become possible system verification [2], [3]. Embedded systems are a special kind of computer system which is scalable on both hardware and software. It must satisfy strict requirements of functionality, reliability, cost, volume, and power consumption of a particular application.

It is known that real time embedded system development has some critical constraints. Then developing and debugging it without tools are quite impossible. These constraints can be better understood by using graphical languages which shows all dynamic process and events occurring during system execution. Colored Petri Nets - CPN has been chosen because it is considered the complete one. Some high level languages commands can be inserted on it. Using CPN a new RTEMS port is still being developed based on Blackfin platform.

In this paper RTEMS, Blackfin structures, and ACDH subsystem are shown.

II. Petri Nets

On next two subsections Colored Petri Net is presented.

A. Colored Petri Nets

Colored Petri Nets - CPN are considered the most complete type of Petri Nets. CPN has colors which does not means just colors or standards. Actually they represent complex data types, using color nomenclature to refer distinction possibilities among tokens. CPN has different notations for places, arcs, and transitions, related to ordinary Petri Nets. For each one of those, it has objects configurations, data types, and conditional codes which can be associated with. Furthermore in the places data types, tokens quantity, and colors can also be associated. That helps understand and discern how processes works and how data pass through. On arcs can be defined tokens quantity, data types, restrictions (high level languages code) which are passing through them. Finally, transitions timers and data types can be associated too.

B. Colored Petri Net Tools

To create and simulate diagrams system tools are necessary. To perform that, CPN tools were used. It is a tool to simulation and analysis. Its Graphical User Interface - GUI is based on

advanced interaction techniques, such as toolglasses, marking menus, and bi-manual interactions [4].

Some feedback facilities provide contextual error messages and indicate dependency relationships between net elements. Tool features increment syntax and logical checking to added code which take place while a net is being constructed. A fast simulator efficiently handles both un-timed and timed nets. Both full and partial state spaces can be generated and analyzed, and a standard state space report contains information such as bounded properties and liveness properties [5].

Functionality of the simulation engine and state space facilities are similar to the corresponding components in Design/CPN, which is a widespread tool for CPN [4].

III. Real Time Systems

On next section RTEMS is presented with a little much details. After, some Blackfin main features is shown.

A. Real-Time Executive for Multiprocessor Systems - RTEMS

Real-time embedded systems are found in practically every facet of our everyday lives. Today there are systems ranging from the common telephone, automobile control systems, and kitchen appliances, to complex air traffic control systems, military weapon systems, and production line control including robotics and automation [6]. However, in the current environment of rapidly changing technology, it is difficult to reach consensus about real-time embedded system definition.

Hardware costs are continuing to rapidly decline while at the same time it is increasing in power and functionality. As a result, embedded systems which were not considered viable two years ago, are suddenly a cost effective solution. In this domain, it is not uncommon for a single hardware configuration to employ a variety of architectures and technologies. Therefore, it is possible to define an embedded system as any computer system which is built into a large system consisting of multiple technologies such as digital and analog electronics, mechanical devices, and sensors.

Even though hardware platforms becomes powerful, most of all embedded systems are critically dependent on the real-time software embedded in it. Regardless of how efficiently the hardware operates, the performance of the embedded real-time software determines the success or not of a system. As the complexity of the embedded hardware platform grows, so does the size and complexity of the embedded software [6].

There are RTEMS ports for many different processor platforms like ERC32, ARM, m68K, among others, but not to Blackfin yet. Then, tailoring RTEMS to make a port to embed on Blackfin processor was first task. This port has many configurations to use on ITASAT project. It was necessary to create and develop a lot of procedures [7].

Error discovery process along RTEMS is a highly important task. Tailoring is directly on to the fact of being a critical system of high risk and has as premise management of concurrent and parallel processes.

It is needed to make one system diagram and insert all necessary parameters, constraints and data types. One may has different approach and ideas about how to solve the problem or errors.

B. Blackfin processor structure

Blackfin processor is a powerful embedded processor. It is of easy access considering the benefit/cost rate, and there are a lot of available material over the Internet [8]. Blackfin is a new breed of 16-32-bit embedded processor designed specifically to meet the computational demands and power constraints of nowadays embedded audio, video and communications applications. Based on the micro signal architecture - MSA jointly developed with Intel Corporation. These processors combine a 32-bit reduced instruction set computer - RISC. The instruction set of Blackfin processor family is comprised by small and simple instruction set which is executed directly by processor without interpreter (microcode) interventions.

This instruction set makes use of dual 16-bit multiply accumulate - MAC signal processing functionality, along with the ease-of-use attributes found in general-purpose micro-controllers. All Blackfin processors offers fundamental benefits to the system designer which includes:

- High-performance signal processing and efficient control processing capability enabling a variety of new markets and applications; and
- Dynamic power management - DPM enabling the system designer to specifically tailor the device power consumption profile to the end system requirements, and an easy to use mixed 16-/32-bit instruction set architecture and development tool suite ensuring that product development time is minimized.

It also offers a variety of benefits most often found on RISC control processors. These features include a powerful and flexible hierarchical memory architecture, superior code density, and a variety of micro-controller-style peripherals including items such as 10/100 ethernet MAC, universal asynchronous receiver/transmitter - UARTS support, watchdog timer, real-time clock, and a glueless synchronous and asynchronous memory controller. All of these features provides the system designer with a great deal of design flexibility while minimizing end system costs.

IV. Case Study

Inside the aerospace context, attitude control means control of angular position and rotation of the satellite, either relative to the object that it is orbiting, or relative to the Earth Planet [9].

On flight dynamics, the orientation is often described using three angles called yaw, pitch, and roll. These angles are controlled by satellite attitude control system, according preprogrammed coordinates. That is for where the satellite will have to be pointing. Thus to analyze these coordinates other resources are required, as for example sensors, actuators, etc.

To work with sensors, it is needed to choose and analyze among diverse models, types and results. The sensor named

3DM-GX1, manufactured by MicroStrain, denominated by Gyro Enhanced Orientation Sensor are being used in this work. Thus it is used on the first stage of satellite system prototype [10].

On next steps, solar sensors will be used on integrated housekeeping unit - IHU to help determine position of the spacecraft. This is not only for normal operation in the three-axis-stabilized mode, but also to orient the satellite for rocket firings. Some accuracy is required.

The sun sensor uses a special-purpose photocell from Hamamatsu Photonics Co. called position-sensitive detector - PSD. That sensor housing acts as a pinhole camera, so the sun forms a small spot of light on the surface of the PSD. It has four electrical contacts, two of that determines the spot position along the 'X' axis and two for the 'Y' axis. Electronics assembly converts the currents from the four contacts into a pair of voltages that represents the sun position in the 'X' and 'Y' directions. These signals are sent to an analog-to-digital converter - ADC on IHU for processing [11].

Parallel of it there are actuators to correct functioning of satellite attitude. They are responsible to move the satellite to correct path. Reaction wheels is being developed with a brushless DC-servomotor with its respective servo amplifier control board. This motor has a nominal voltage of 24 volts (V) and no-load speed of 18000 rotation per minute - RPM. It has also hall effect sensors integrated which are directly connected over the servo amplifier control board for a better control of the output voltage. The servo amplifier control board has an analog control and its input voltage varies from -5V to +5V. It has to control motors with nominal voltage up to 56V and no-load speed up to 40000 RPM. To connect BlackFin to servo amplifier control board it will be required other devices like a digital to analog converters - DAC or a digital potentiometer. It will be used Blackfin programmable flags - PFs to accomplish the connection with this device. A motor-speed sign of monitor is available on servo amplifier control board. It will be used on attitude control development.

RTEMS is the real time operating system embedded on Blackfin and both are responsible for control the whole subsystem surrounding the small satellite. Figure 1 illustrates the system behavior of satellite attitude control, modelled with Colored Petri Nets.

Using CPN for development system charts, it is possible to visualize the whole processes and functionalities which surround satellite system. Thus, it is also possible to develop systems through oriented objects and abstraction layers.

Figure 1 shows the satellite and its processes and subsystems. Thus it is possible to divide the system in small subsystems or blocks and design its diagrams on CPN tools before making development processes. Searching for errors not predictable, for example logical errors, invalid conditional errors, validation and integration, are activities strongly executed through Colored Petri Nets concepts.

Figure 2 shows a prototype scheme comprising Blackfin, RTEMS, and ACDH application and some of its processes. In this case the real time operating system RTEMS is embedded

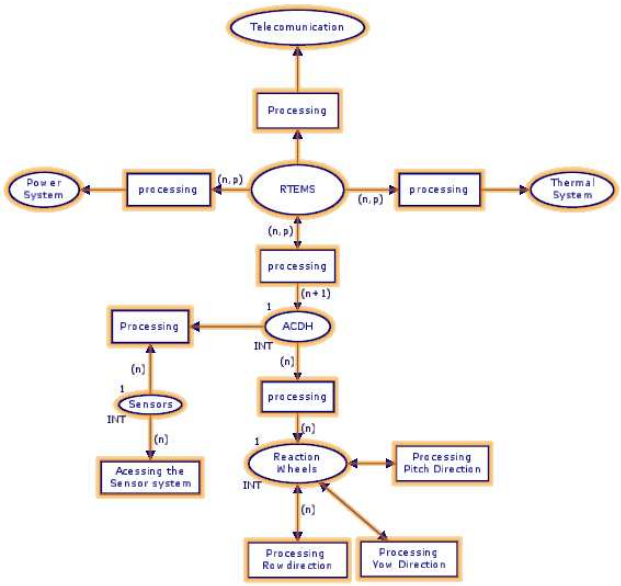


Fig. 1. Illustration of RTEMS and its subsystem controls, modelled with Colored Petri Nets

in a Blackfin processor. The attitude control and data handling is running on RTEMS accessing some system functions. ACDH control sends a message asking for sensor data, and after processes this data. The processed data is returned to ACDH control. If this data is different than pre-programmed data, commands are sent to actuators and the orbit is corrected in on each one of the angles (row, pitch, and yaw).

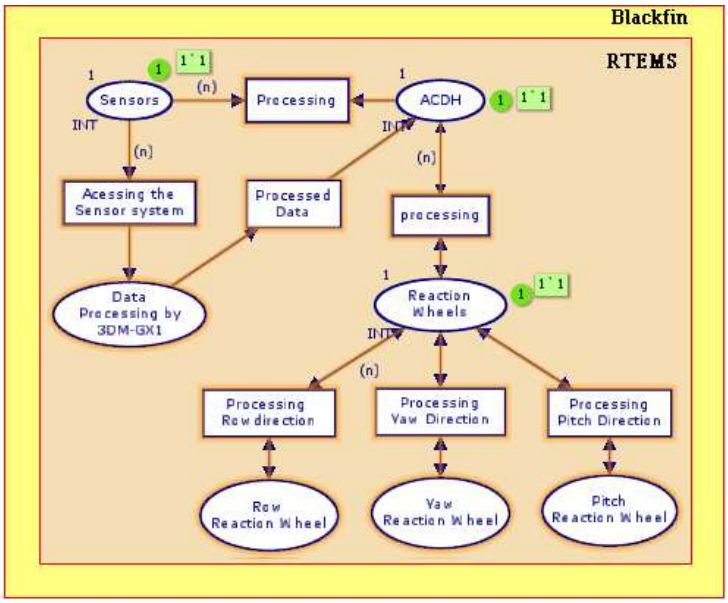


Fig. 2. Snapshot of the application

V. Conclusion

This work is still being developed at the same time of studies about real time embedded systems, RTEMS and Blackfin

platform structure.

The attitude control and data handling - ACDH development process is being done with the help of Petri Net tools.

Creating subsystem diagrams on Colored Petri Nets - CPN makes the integration process less complex, once subsystems has interdependent functionalities over the real time operating system and hardware platform structures. System routines are dependent of RTEMS and Blackfin due to management mechanisms which controls the hardware part through software.

By using CPN modeling, dependencies will be graphically visible. This strongly helps development tasks and error detection activities.

This work is helping each team member to understand functionalities of the whole satellite. Without this approach adopted here, students vision would be limited because they have worked on only one or two subsystems. Thus, through the use of subsystems diagrams with CPN, system behavior can be understood much easily.

ACKNOWLEDGMENTS

The authors of this paper would like to thank the following institutions for their support to this work: the Brazilian Aeronautical Institute of Technology (Instituto Tecnológico de Aeronáutica - ITA); the Brazilian Space Agency (Agência Espacial Brasileira - AEB); and the Brazilian National Institute for Space Research (Instituto Nacional de Pesquisas Espaciais - INPE).

REFERENCES

[1] I. Group, "Itasat student satellite program," World-Wide Web document, 2007. [Online]. Available: http://www.itasat.ita.br/

[2] C. A. Petri, *Communication with Automata*. New York: Griffiss Air Force Base, Tech. Rep.RADC-TR-65-377, vol. 1, Suppl. 1, 1966.

[3] ——, *Kommunikation mit Automaten*. Bonn: Institut fr Instrumentelle Mathematik, Schriften des IIM Nr. 3, 1962.

[4] C. Group, "What is cpn tools," World-Wide Web document, 2006. [Online]. Available: "http://wiki.daimi.au.dk/cpntools/cpntools.wiki"

[5] T. Murata, *Petri Nets: Properties, Analysis and Applications*. Campinas, SP - Brasil: Proceedings of the IEEE, Vol. 77, pp. 541-580., 1989.

[6] RTEMS, "Real time operating system for multiprocessor systems," World-Wide Web document, 2006.

[7] R. Group, "Blackfin configuration files," World-Wide Web document, 2006. [Online]. Available: http://www.rtems.com/ftp/pub/rtems/testing2/4.8/redhat/7.3/SRPMS/

[8] A. D. Group, "Blackfin processor architecture overview," World-Wide Web document, 2006. [Online]. Available: http://www.analog.com/processors/blackfin/overview/archOverview.html

[9] W. Group, "Attitude control," World-Wide Web document, 2006. [Online]. Available: "http://en.wikipedia.org/wiki/Attitude_control"

[10] MicroStrain, "3dm-gx1 detailed specifications," World-Wide Web document, 2007. [Online]. Available: http://www.microstrain.com/

[11] R. Zimmermann, "Sun sensor," World-Wide Web document, 2006. [Online]. Available: "http://www.amsat.org/amsat/sats/phase3d/sun_sensor.html"

[12] M. M. A. C. G. CHIOLA, G., "Generalized stochastic petri nets: A definition at the net level and its implications." 1993.

[13] R. N. COOLAHAN, J.E., *Timing Requirements for Time-Driven Systems Using Augmented Petri Nets.* IEEE Transactions on Software Engineering, 1983., 1983.

[14] H. P. S. R. M. JENSEN, K., *Hierarchies in Coloured Petri Nets, Lectures Notes in Computer Science.* H, Vol.483, 313-341, Springer-Verlag., 1990.

[15] K. JENSEN, *Coloured Petri Nets: Basic Concepts, Analisys Methods and Pratical Use.* New York, v. 1, Springer-Verlag., 1992.

[16] P. N. World, "Petri nets world," World-Wide Web document, 2006. [Online]. Available: http://www.daimi.au.dk/PetriNets/

[17] J. R. Norris, *Markov Chains.* Cambridge: Cambridge University Press, 1998.

[18] P. R. Maciel, *Introducão s Redes de Petri e Aplicacões*. Campinas, SP - Brasil: X Escola de Computacão, Campinas SP, 1996.

[19] R. Jain, *The Art of Computer Systems Performance Analyses*. New York, NY, April 1991: Wiley - Interscience, 1991.

[20] P. D. A. Hennessy, J. L., *Computer Architecture: A Quantitative Approach*. 3rd Edition, Morgan Kaufmann Publishers, 2003.

[21] C. Girault and R. Valk, *Petri Nets for Systems Engineering: A Guide for Modeling, Verification and Application*. Verlag: Springer, 2003.

Testing Issues in Empirical Reliability Evaluation of Embedded Real-Time Systems

Falk Salewski, Stefan Kowalewski

***Abstract*— Embedded systems containing software are nowadays used more often in safety-critical real-time applications. Thus, the reliability of their software components becomes an important issue. Established results about software reliability, especially for the different combinations of software and hardware components used in embedded systems, are rare. Accordingly, we see a great need for empirical evaluation in this context and conducted several experiments at our institute. In this paper, the particularities of empirical reliability evaluation in the domain of embedded real-time systems are presented. Testing requirements play a major role in this context and should be considered not only for the experiment design but also for system design in general.**

I. Introduction

EMBEDDED systems are taking over more and more functions in industry and daily life. Failures in their software can now directly lead to inconvenience or even serious accidents if these systems do not meet their requirements. In order to prevent hazards these systems have to be evaluated according to their reliability. In case of major design decisions and changes (e.g. software or hardware architecture, design processes) the influence of these measures on the reliability of the system's software and the overall system is of great interest. In the case of hardware (mechanics and electronics) the reliability of a system can be determined from the system's architecture and the reliability of the components used. However, not many established results on influences regarding software reliability are available [7], not to speak of the specialties of software found in embedded systems as real-time requirements and efforts caused by hardware/software interdependences. For this reason, empirical reliability evaluation of software in these systems becomes an important issue. Due to specific properties of embedded systems, this evaluation is often challenging. These specialties of empirical evaluation in embedded systems will be discussed in this paper.

All authors are with the Embedded Software Laboratory - Chair of Computer Science XI, RWTH Aachen University, 52074 Aachen, Germany, (surname)@informatik.rwth-aachen.de

II. The Need for Empirical Evaluation

As mentioned before, not many well-founded results exist for software reliability in the realm of embedded systems. Systems are designed according to best practice approaches. In some cases this may be critical, as some approaches are based on wrong assumptions. One example is the improvement of reliability by using redundant structures. This structure will only improve the reliability if these redundant units fail independently. When transferred to software design one method to improve this independence recommended by the IEC61508 [3] is the so called N-version programming. In this approach software for redundant units is developed by different teams. However, 20 years ago Knight and Leveson have already shown with an empirical experiment that even two independent teams of software engineers tend to make the same errors [4]. First steps in handling this dependency structure are based either on analyzing the fundamental domain of development or on prediction of failure probabilities as described in [1]. One of the former methods is "forced diversity" introduced by [5] with an empirical evaluation in [6]. An experiment applying the idea of forced diversity by using different hardware platforms (microcontroller and FPGA) to achieve diversity in the different versions has been conducted at our institute. As a result of this experiment, dependencies in failure behavior could be found, even between systems developed on different hardware platforms (see [10] for details).

Beside the fact that even improved methods of N-version programming do not automatically lead to modules with independent failure behavior, a redundant structure created this way will probably still have an improved reliability in comparison to a single module. However, it can be questioned if this improvement justifies the effort. Alternatively, the second team could be used for testing and/or review only or techniques as pair programming could be applied. In order to compare the efficiency of different methods of reliability improvement quantitative results are needed.

The evaluation of N-version programming is an example for the importance of empirical evaluations. Other aspects could imply the impact of design processes, programming paradigms, programming languages, operating systems, and software and hardware architecture on the reliability of the corresponding software.

III. Empirical Evaluation - Where and How?

One type of empirical evaluation is the formal experiment [2]. These experiments have to be planned carefully and a few challenges have to be met. This includes a task being complex enough to be representative but easy enough to take only reasonable time to complete. Furthermore, a sufficient number of representative participants is needed. These two aspects usually do not allow to perform experiments in industrial settings, although they probably offer a more realistic environment according to programmers experience, tools available and the task itself.

Another option are experiments in lab courses and projects at universities. According to a relatively high number of participants, lab courses are a good basis for experiments. We believe that combining lab courses with experiments whenever possible is a good option to achieve useful results in the domain of reliability evaluation. A high number of experiments performed at teaching institutions around the world could form a pool of results valuable for many analyses with respect not only to reliability, but also to other non-functional attributes.

When experiments are applied to evaluate the consequences of design decisions to software reliability, the different versions generated by the participants must be analyzed with respect to their specified functionality. Apart from reviews/inspections or formal verification, testing is the most important means for doing that [3], [8]. Although testing can never guarantee the absence of failures, the number of failures found by extensive testing can still be used as an indicator for the system's reliability [2], [8]. Analyzing the results of an experiment includes the testing of all the different outcomes created with equal test cases. Therefore, black box testing should be applied since this technique is independent from the individual subject tested. Since the significance of the results gained by black box testing is increasing with the amount of test cases a high number of different test inputs is desirable.

In section 5 we will report on particular testing-related experiences from experiments which we performed in our lab courses. Before, some specific requirements for experiments in the embedded systems domain will be presented.

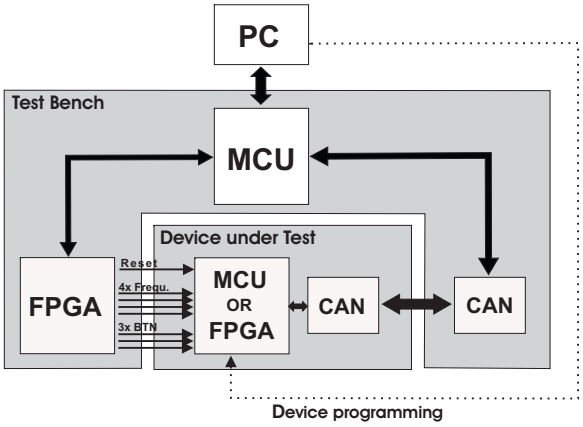


Fig. 1. Automatic test environment

IV. Particularly Requirements for Empirical Evaluation in Embedded Systems

Embedded systems realize functions in interaction with their environment (e.g. an ABS control unit in a car). In contrast to mere software functions, these functions have to be analyzed via dedicated hardware/software interfaces. In the case of an ABS control unit these are, among others, the inputs from the wheel sensors and the outputs to the actuators.

Additionally, embedded systems often have to fulfill real-time requirements. According to this requirements, it is usually of great importance to analyze the system's real-time behavior. In the case of an ABS control unit it would have to be checked if the response on inputs generates outputs according to the specified timing behavior. In order to analyze real-time behavior specific measurement equipment is necessary.

The aspects presented above make an automated test environment necessary. For the majority of embedded applications these test environments have to be designed for the individual application. The testing for correct real-time behavior demands real-time properties of the corresponding parts in the test environment, which complicates the design and verification of the test environment.

V. Experiences in Empirical Evaluation of Embedded Real-time Systems

Three experiments concerning reliability in embedded systems have been taken place at our institute, another is currently running. The first three experiments investigated the impact of different hardware platforms used in

embedded systems on the reliability of the corresponding software. One question investigated was whether N-version programming based on software developed for different hardware platforms significantly decreases the dependence between the redundant software modules. The first experiment [10] took place in a lab course described in [9] with 26 students. During the course, the students had to realize a quadruple speed measurement on different hardware platforms and sent their measurement data to a CAN bus. In order to mask out the skills of individual participants a crossed design was used (all 12 teams had to realize the same task on both hardware platforms in random order). Then, we analyzed the 22 versions produced in this experiment with an automatic test bench designed for this purpose (see Fig. 1). The majority of the following experiments used the same basic task expanded by additional task to investigate different aspects. During the evaluation of the data, recorded in the different experiments, several testing issues presented in figure 2 had to be faced which will be discussed in the following.

The first issue was the test data generation itself. Black box testing was used since test data independent from the different versions tested was needed. For the different experiments, a combination of random test data, data representing stress tests and test data representing different scenarios had been used for evaluation.

On basis of the test data generated, the corresponding physical signal had to be generated. Challenges resulted from the frequency and the complexity of the signals which have to be fed into the device under test (DUT) during test. For the first experiment, four signals of independent frequency (0Hz-12kHz) and phase had to be generated. This task had to be realized on an FPGA since available MCUs were not able to generate these signals. The number of signals which had to be generated simultaneously was no specific challenge in this case, but higher numbers (especially numbers $>$ bit width of MCU ports) could be an additional challenge.

A third aspect was the recording of the corresponding response data from the DUT. The task chosen for the experiments required measurement intervals up to 2 seconds, depending on the input (low speed required a long measurement interval). This led to particular long test runs (e.g. $> 2h$ for 20000 lines of test data). Another challenge were very short response times of the DUT. Some versions of the first experiment sent their CAN messages faster than expected so that the test environment had to be adapted in order to cope with this speed and amount of data. In order to reduce the amount of data produced with every test run we specified a minimum response time in the following experiments. In the majority of our experiments, the output format of the DUT was a CAN message which allowed a comparatively easy evaluation of the results. Outputs based on binary values seem easier to evaluate on the first view, but only if the bit width is low. Accordingly, the complexity of the output data is limited if binary outputs are used. Additionally, the CAN communication benefits from built in error detection and correction codes.

A major challenge resulted from the evaluation of the recorded results. Beside every input combination the timing of all input combinations had to be considered during evaluation. This could be handled for the evaluation of the first experiment but the complexity of the evaluation was increased by adding additional inputs in the second experiment. The timing of the DUT outputs was checked by time-stamping every incoming CAN message received by the test environment.

The following two aspects are closely related since they consider verification of the test process and the repeatability of the test result which is part of the verification process. The real-time properties of the DUT led to problems concerning the repeatability of the measurements. The repeatability has been improved by performing a defined reset of the DUT by the test environment before each test run, but in some versions little variations remained. In order to consider these variations several test runs were performed for each DUT and are included in the evaluations. The verification has been eased by a strict separation between the test run itself and the evaluation of the test results. Additionally, all major intermediate results present during evaluation have been added into the output file allowing manual examination.

Finally, the modifiability of the test environment is another aspect which turned out to be very important. In the test environment, used for our evaluations, signal generation has been realized with an FPGA which allows great flexibility since new signals can be added to the test environment without influencing already existent signals (parallel structure of the FPGA). This concept has shown to be very useful when new signals, needed for the evaluation of the second and further experiments had to be added.

The aspects discussed above in this chapter revealed that testing plays an important role in the context of reliability evaluation in embedded real-time systems. The necessary effort for the testing of DUTs and the verification of the test environment itself depends very

Testing issue	Sub issue	Challenge
1. Test data generation	test coverage	achieve sufficient test coverage
2. Test signal generation	frequency/complexity of input signals	creation of signals in real-time
	number of input signals	creation of signals in real-time
3. Recording of response data (DUT output)	long response time	results in long test runs
	short response time	recording and preprocessing of signals in real-time
	number/complexity of output signals	recording and preprocessing of signals in real-time
4. Evaluation of response data	relationship between input and output	considering all possible relations between inputs and outputs
5. Repeatability of results	test environment	in each test run with the same test data, the test environment should deliver the same results
	device under test (DUT)	in each test run with the same test data, the DUT should deliver the same results
6. Verification of test results	data collection during test run	rule out faults in the test run (recording and preprocessing)
	evaluation of test data	rule out faults in the evaluation of the test data
7. Modifiability of test environment	scaleability	handle increasing numbers of DUT inputs and outputs
	adaptability	handle new types of input and output signals

Fig. 2. Testing issues and their challenges

much on the design of the experiment. Therefore, these issues should be considered during experiment design.

VI. Conclusions and Future Works

A. Conclusions

This paper started by recalling that empirical evaluation is needed to verify techniques and measures used in software design. There is also a great need for quantitative results of different approaches in order to pick the most effective ones for the individual application.

Empirical evaluation of reliability in embedded systems requires special methods. Testing is is an integral part of it. However, testing is usually only possible at hardware interfaces and very often real-time requirements have to be checked. We presented our experience with the evaluation of own experiments. These experiences include different testing issues and their challenges according to the domain of embedded real-time systems.

As a result, testability of the functions present in the experiment and verifiability of the corresponding test environment should be considered during the design of the experiment (design for testability). These measures could significantly simplify the evaluation process, not only in empirical evaluations, but also in system design in general. Furthermore, the modifiability of the test environment should be considered during its design.

B. Future Works

As mentioned above, the task used for an experiment does significantly influence the effort needed for evaluation. Thus, the correlation between applications and testing issues will be further investigated to allow sufficient designs of experiments in the domain of embedded real-time systems. Furthermore, it will be investigated how well the architecture of our test environment is suited for the evaluation of future experiment data.

The results of the second experiment mentioned in this paper are submitted for publication. The evaluation of the other two experiments is currently running.

References

[1] X. Cai and M. R. Lyu. An empirical study on reliability modeling for diverse software systems. In *Proceedings of the 15th International Symposium on Software Reliability Engineering (ISSRE04)*, 2004.

[2] N. E. Fenton and S. L. Pfleeger. *Software Metrics - A Rigorous & Practical Approach*. PWS Publishing Company, 1997.

[3] IEC61508. *IEC61508: Functional safety for electrical / electronic / programmable electronic safety-related systems*. International Electrotechnical Comission, 1998.

[4] J. C. Knight and N. G. Leveson. An experimental evaluation of the assumption of independence in multiversion programming. *IEEE Trans. Softw. Eng.*, 12:96–109, 1986.

[5] B. Littlewood and D. R. Miller. Conceptual modeling of coincident failures in multiversion software. *IEEE Trans. Softw. Eng.*, 15(12):1596–1614, 1989.

[6] M. Lyu and Y. He. Improving the n-version programming process through the evolution of a design paradigm. *IEEE Transactions on Reliability*, 42(2):179–189, 1993.

[7] T. J. Ostrand and E. J. Weyuker. The distribution of faults in a large industrial software system. In *Proceedings of the International Symposium on Software Testing and Analysis (ISSTA)*, pages 55–64, 2002.

[8] D. L. Parnas, J. van Schouwen, and S. P. Kwan. Evaluation of safety-critical software. *Communications of the ACM*, 33:636–648, 1990.

[9] F. Salewski, D. Wilking, and S. Kowalewski. Diverse hardware platforms in embedded systems lab courses: A way to teach the differences. In *First Workshop on Embedded System Education (WESE)*, volume 2. SIGBED Review, 2005.

[10] F. Salewski, D. Wilking, and S. Kowalewski. The effect of diverse hardware platforms on n-version programming in embedded systems - an empirical evaluation. In *3rd International Workshop on Dependable Embedded Sytems (WDES'06)*, 2006.

Indoor Passive Localization System Performance Issues

Robert Sprick*, Steve Goddard*, Lance C. Pérez†, Chen Xia†
Department of Computer Science and Engineering*
Department of Electrical Engineering†
University of Nebraska–Lincoln
{rsprick, goddard}@cse.unl.edu, perez@unl.edu, chenxia@mariner.unl.edu

Abstract

Inaccurate ranging measurements and inadequate sampling rates negatively impact the performance of real-time, two-dimensional (2-D) indoor passive location systems. Methods to address these issues are presented, followed by a comparison of localization and tracking performance metrics. It is argued that a video average distance error provides a more accurate measure of a system's performance than the more common perpendicular distance error.

1 Introduction

The performance of passive localization systems often suffer from inaccurate ranging measurements and inadequate sampling rates. Inaccurate ranging data is an old problem for localization systems and methods to correct it have been implemented, e.g., [7]. However, these approaches typically try to correct the effects of inaccurate ranging data after localization instead of fixing the data directly. Current methods of handling inaccurate ranging data are discussed and ideas are proposed for future research. Inadequate sampling rates are harder to quantify because they are dependent on the application. Methods to improve the sampling rate of localization systems and some of their limitations in the indoor environment are discussed.

When studying inaccurate ranging measurements, the proper metric for measuring the performance of a passive localization system is called into question. Improper evaluation can overestimate a system's performance while hiding underlying flaws. To illustrate this point, a passive system is first evaluated using two different error metrics that ignore the latency created by tracking algorithms, and then an error metric is presented that quantifies the impact of this latency.

Section 2 defines a passive localization system and describes current methods to reduce the impact of inaccurate ranging measurements and inadequate sampling rates. Section 3 investigates multiple approaches for measuring the localization and tracking accuracy of moving objects. Finally, Section 4 summarizes the paper and presents a method under investigation to combat inaccurate ranging measurements.

2 Passive System Challenges

The motivation for this work is to improve tracking performance by developing methods to handle the asynchronous ranging data that occurs within a passive mobile node architecture. The passive mobile node retrieves only one distance or range value for each time step in a triple data set format: [t, p, d], where t is the current time, p is the known position of the transmitting beacon, and d is the distance between the mobile node and the beacon [7]. Typically the last N triples from different beacons are combined to perform a localization calculation using linear least-squares (LSQ) minimization, where N signifies the number of valid past triples within a history window. The accuracy of this calculation depends on the time difference between the ranging measurements, the movement characteristics of the tracked object, and the accuracy of the individual range measurements. The sampling rate of the system is defined as the average frequency with which a range value is received. At high speeds or with low sampling rates, the ranging data becomes stale very quickly [9]. This causes an error in the localization of the object that is similar to when bad range measurements are used.

The Cricket Location-Support System (CLS) [6] is an excellent example of a system that exhibits these traits. Thus, it is used as a test bed to evaluate our approach to providing real-time tracking in a noisy and under-sampled indoor environment. The main sources of error in our environment are inaccurate ranging data and range data staleness due to low sampling rates.

The CLS makes range measurements using the time difference of arrival (TDoA) of radio frequency (RF) and ultrasonic (US) signals. The beacon (RF) transmission schedule is a CSMA/CA [1] protocol with random back-off for RF collisons. The transmission time between two consecutive range signals (RF-US pair) varies with an individual programmable average of 1 Hz per beacon. The resulting sampling rate is dependant on the beacon density within the RF signal coverage.

2.1 Inaccurate Ranging Data

Inaccurate ranging data arises when there is a timing mismatch of incoming signals or when outside sources interfere with the measurement process. In both passive and active systems, careful handling of inaccurate ranging data is required for a robust localization system.

History Window and Mode

One method to reduce inaccurate ranging data is to store previously received ranging measurements in a history window for each beacon. The size of the window can be set to the number of data points stored or to a time limit after which the data points are no longer valid.

Table 1. Average 2-D localization error for stationary object

Method	4 beacons	8 beacons	15 beacons
Cluster	2.569 cm	4.670 cm	8.755 cm
LSQ	2.258 cm	185.712 cm	769.344 cm

The history window is analyzed to obtain statistical information about previously received ranging data. This information can either be used directly or as a basis for determining the validity of future range measurements. In CLS, history windows are used to calculate the standard deviation and mode of the previous range measurements for each beacon. The mode of the range measurements is the most frequently occurring measurement. If there are several values with the same frequency, the measurements are averaged. In CLS, the mode is used as the representative measurement for the beacon in the LSQ algorithm. A large standard deviation for a beacon indicates a lower confidence in the mode and may exclude the beacon from inclusion in the LSQ algorithm.

The history window method is biased toward the localization of a stationary object because it uses the mode. This creates a problem for tracking an object during its transition from stationary to moving. Since the mode calculation is biased toward repeated measurements, it may not use the newest range measurement for a moving object. This introduces a short latency in location updates during the transition period.

Position Clustering

Another method to reduce inaccurate ranging is to use k-means [4] clustering during the LSQ calculation. The LSQ algorithm requires 3 range measurements to calculate a position. With dense beacon coverage, more than 3 range measurements are contained in the input range set to the LSQ algorithm. To generate a set of positions for the clustering algorithm, different subsets of the input range set are iteratively applied to LSQ. The positions are analyzed using k-means [4] clustering with $k = 1$ to determine the final position.

With 4 range measurements, four position estimates are calculated using four unique range subsets of size 3. When a highly inaccurate range measurement is in the input set, the three closest positions to the center of the cluster result from the 3 subsets containing the inaccurate range measurement. Therefore, the outlier of the cluster is chosen as the final position estimate because its input range subset did not contain the inaccurate range measurement.

Table 1 shows the average 2-D localization error for a stationary object with position clustering (Cluster) and without (LSQ) in CLS with sets of 4, 8, or 15 beacons. The average 2-D localization error is calculated for the N experiment points (x_i, y_i) using the 2-D Euclidean distance to the actual position $(x^{\text{act}}, y^{\text{act}})$

$$\frac{\sum_{i=1}^{N} \sqrt{(x^{\text{act}} - x_i)^2 + (y^{\text{act}} - y_i)^2}}{N}. \quad (1)$$

Unlike previous experiments on a stationary object [3] with 15 beacons, not all of the beacons are deployed within the US transmission range. However, the RF signal is received at the listener from all 15 beacons. Ranging data is recorded for 15 minutes using the history window provided with CLS [7]. The window uses the mode calculation for the range measurement to LSQ, but only a single range measurement from a beacon is required to be included in the input range set.

Clustering is able to identify and remove the impact of inaccurate range measurements in the final result when there are greater than four beacons within RF-US transmission range. However, clustering is also biased toward objects that are stationary or have constant velocity.

2.2 Inadequate Sampling Rate

The appropriate sampling rate for a localization system is dependent on the maximum speed of the object, the allowed error level of the application and the scheduling scheme used to report the ranges. The appropriate sampling rate is difficult to quantify for tracking human movement.

The effective sampling rate for CLS is the average number of successful RF-US pairs that arrive at the listener. Table 2 shows how the effective sampling rate changes for a stationary object as beacons are added within the RF transmission limit but outside the US coverage. The effective sampling rate decreases as more beacons are added for full coverage of the space. Similar results are reported in [3] and [5].

Because of the random scheduling in the CSMA/CA protocol, the probability exists that a range measurement from a beacon will not be updated for several seconds. The lack of a range update is equivalent to the accuracy of the measurement decreasing with time. This phenomenon is labeled range staleness and is more pronounced for high speed movements [9].

There is a tradeoff between coverage density and effective sampling rate that can be mitigated by reducing the RF transmission power level. However, this increases the probability of RF interference when nearby beacons cannot schedule around each other, which is similar to the hidden node problem [2]. Another option is to switch from a CSMA/CA to a TDMA [1] transmission scheduling scheme to have a uniform and controllable sampling rate.

A TDMA scheme was implemented for a passive architecture in [5] using CLS. The implemented scheme uses a color-coding, static pipelined scheduling scheme to minimize the chance of RF and US interference at the listener. A transmission protocol was also simulated that used a pipelined scheme with localization result feedback. The location-aware scheme reduces the scheduling restrictions around the collision of two beacons. These methods report a higher sampling rate than the scheduling scheme in CLS, but their performance degrades when there are multiple listeners. In contrast, a more static TDMA protocol will support multiple listeners, but still suffers from range staleness at high speed even though range updates are structured and expected.

Table 2. Sampling rate statistics for a stationary object with 4, 8, and 15 Cricket beacons

Parameter	4 Beacons	8 Beacons	15 Beacons
Individual Beacon Range Sampling Rate	0.855 Hz	0.654 Hz	0.355 Hz
Received Rate of RF packets at Listener	3.425 Hz	5.236 Hz	5.556 Hz
Effective Range Sampling Rate	3.425 Hz	5.208 Hz	2.899 Hz

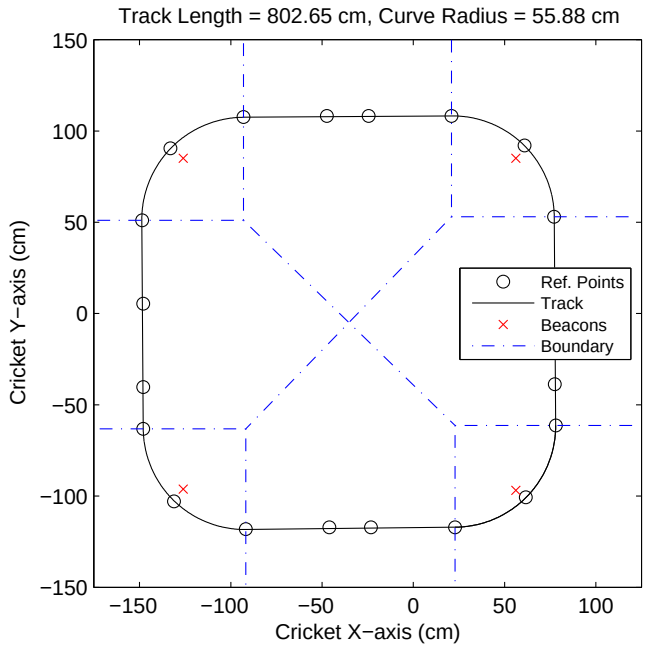


Figure 1. Train Layout for Experiment

3 Evaluating Localization Error

The performance evaluation of a passive localization system is an interesting problem. Should the estimated position be evaluated only on its 2-D distance away from the known path or should the time of the estimate be matched to the actual position of the object? How much latency is introduced by the position computation and filtering?

To study these questions and other aspects of a passive localization system, experiments are conducted with 4 Cricket beacons mounted on the ceiling and a Cricket listener mounted on a model train set for repeatable results at 3 different speeds. The train track layout is shown in Figure 1, where one loop around the track, measured with a string, is 802.64 cm in distance. The curves on the corners have 55.88 cm radius. The 3 speed levels of the train are 15.44, 29.29, and 51.09 cm/s.

In the experiments, the train is initially positioned at a preset start location. The LSQ algorithm evaluates the range measurements from the history window for localization. For performance comparison, two tracking algorithms process the range measurements and LSQ results. The first, a simple velocity model Kalman filter (SKF) [8], uses the LSQ results as input to track the object. The second tracking algorithm is the Extended Kalman Filter (EKF) found in CLS, described in [7]. The EKF accepts a single range measurement as an argument for each call to compute a position estimate. Once the localization algorithm identifies the train location, a video recorder is started and the train accelerates up to a preset speed. The train performs one loop around the track and is stopped after reaching the starting point. The video recording is stopped and ranging data, LSQ, SKF, and EKF results are recorded for each experiment.

The video recordings are used to retrieve data on the actual position of the train with respect to time. The track is physically measured to obtain 20 reference points (Figure 1) within the coordinate system used by CLS in the room. The reference points are used to generate a calculated path of the train for visual and mathematical comparison to the LSQ and tracking results of the experiment.

Two error metrics are used for evaluation of the experimental results. The first metric is the average perpendicular distance error from the known path of the train. The second metric is the average distance error of the position compared to the actual position of the train extracted from the analysis of the video.

3.1 Perpendicular Distance Error

Finding the perpendicular distance (PerDis) error from the layout shown in Figure 1 is harder than it initially appears. Each experimental position point $(x^{\text{exp}}, y^{\text{exp}})$ is naïvely related to the closest section of the path. The layout is divided into 8 sections whose boundaries are shown in Figure 1. For the straight track sections, the equation of the line is calculated as $ax+by+c=0$, where a, b, and c are constants describing the line formed by a set of (x, y) points. The average perpendicular distance error for that section is calculated as

$$\text{PerDis} = \frac{\sum_{i=1}^{N} \sqrt{\frac{(ax_i^{\text{exp}}+by_i^{\text{exp}}+c)^2}{a^2+b^2}}}{N}. \tag{2}$$

The distance calculation for a curved section of track is performed differently. Because the radius of the track pieces is known, the center point of each quarter circle is found. The experimental position is transformed into polar coordinates with the center point at the origin. The absolute value of the radial coordinate minus 55.88 cm is the perpendicular distance error.

3.2 Video Average Distance Error

The method used to analyze the video to obtain the actual position identifies when the train starts to move by examining the pixel difference between 2 consecutive frames of the video. This analysis also provides an accurate position of the train with a time stamp after the train has moved. The initial stationary data is related to the starting position of the train while all results after the train has stopped are related to the final position.

The video average distance error (VidDis) uses the 2-D distance equation between the LSQ result $(x^{\text{exp}}, y^{\text{exp}})$ and the

Table 3. Comparison of error metrics for LSQ results

Error Metric	15.44 cm/s	29.29 cm/s	51.09 cm/s
PerDis	4.447 cm	7.098 cm	19.992 cm
VidDis	7.790 cm	11.445 cm	35.121 cm

Table 4. Comparison of latency VidDis Error in SKF and EKF results for 3 speed levels

Error Metric	15.44 cm/s	29.29 cm/s	51.09 cm/s
LSQ VidDis	7.790 cm	11.445 cm	35.121 cm
SKF VidDis	8.475 cm	14.759 cm	52.995 cm
SKF tVidDis	13.247 cm	23.691 cm	171.971 cm
EKF VidDis	10.580 cm	20.822 cm	40.622 cm
EKF tVidDis	10.580 cm	22.982 cm	44.849 cm

video position $(x^{\mathrm{act}}, y^{\mathrm{act}})$ from the closest frame in time. The average video distance error is

$$\mathrm{VidDis} = \frac{\sum_{i=1}^{N} \sqrt{(x_i^{\mathrm{act}} - x_i^{\mathrm{exp}})^2 + (y_i^{\mathrm{act}} - y_i^{\mathrm{exp}})^2}}{N} \tag{3}$$

The problem is accurately relating the time step of the experimental data to when the train actually begins to move. Since we were unable to synchronize the video analysis and the localization system, the LSQ results are matched to the video results by finding the minimum error between them. The starting point of the video data is shifted by one video frame interval along the first 20 seconds of LSQ position results. At each step the VidDis is calculated and the estimated starting time of the train is recorded for the minimum error value.

Table 3 shows a comparison of the error metrics for the LSQ results of one loop around the track. The VidDis error metric has a larger error because it uses the time of the calculation to relate the location estimate to a specific point on the track. While most literature appears to report perpendicular distance error, we believe video average distance error is a more accurate measure.

3.3 Latency

The time component of the VidDis metric also enables the study of additional errors caused by the latency in the processing of the ranges and LSQ results by different tracking algorithms. The latency error of tracking above pure localization is an important factor in real-time applications. The VidDis for each tracking algorithm is calculated for each set of position results. The tracking latency VidDis (tVidDis) error is found by using the starting time stamp from the LSQ results to align the video data and calculate the error value. LSQ is chosen as the time baseline because the minimum error level occurred first in the experimental time frame before the VidDis time stamp of the tracking algorithms. Table 4 shows the VidDis and tVidDis errors for SKF and EKF filters in comparison to LSQ VidDis error levels. The tVidDis errors are larger than their counterparts because they take into account the extra processing time.

4 Conclusions and Future Work

Inaccurate ranging measurements and inadequate sampling rates negatively impact the performance of passive localization systems. Clustering during localization can reduce the impact of inaccurate ranging measurements of stationary objects, but more research is required to show improvement for moving objects. Inadequate sampling rates can be difficult to counteract using a CSMA/CA protocol. TDMA protocols provide increased sampling rates, but often at the expense of flexibility to support multiple listeners. The evaluation of a passive localization system is an important step in showing its improvement over previous systems. Detailed evaluation metrics are presented that show average video distance error more accurately measures performance than the more common perpendicular error metric. With real-time applications, the time of the localization and the latency involved in delivery of the results may have a profound impact on the systems overall performance, which is often ignored in the literature.

Future work includes a predictor algorithm that analyzes each new range measurement as it arrives to determine if the measurement is valid based on a history window and predefined limits around predicted range measurements. If the range measurement does not meet the criteria, it could be discarded or replaced with one of the predicted ranges before being passed to the LSQ algorithm.

References

[1] *American National Standard T1.523-2001, Telecom Glossary 2000*. http://www.atis.org/tg2k/

[2] A. Bachir, D. Barthel, M. Heusse, and A. Duda, "Hidden nodes avoidance in wireless sensor networks," *2005 International Conference on Wireless Networks, Communications and Mobile Computing*, vol. 1, pp. 612-617, June 2005.

[3] C. P. Gleason, L. C. Pérez and S. Goddard. "On the ranging connectivity in the cricket localization system," *Proceedings of 2006 IEEE International Conference on Electro/information Technology*, pp. 619-624, May 2006.

[4] J. MacQueen, "Some methods for classification and analysis of multivariate observations," *Proc. 5th Berkeley Symp. on Mathematical Statistics and Probability*, vol. 1, pp. 281-297, 1967.

[5] M. Nam, M. Al-Sabbagh, and C. Lee, "Combined scheduling of sensing and communication for real-time indoor tracking in assisted living," *RTSS '06. 27th IEEE International Real-Time Systems Symposium, 2006*, December 2006, pp. 281-290.

[6] N. B. Priyantha, *The Cricket Indoor Location System*. PhD thesis, Massachusetts Institute of Technology, 2005.

[7] A. Smith, H. Balakrishnan, M. Goraczko, and N. Priyantha, "Tracking moving devices with the cricket location system," in *Proc. MobiSYS'04*, pp. 190-202, June 2004.

[8] G. Welch and G. Bishop. "An introduction to the kalman filter," Department of Computer Science, University of North Carolina at Chapel Hill, Chapel Hill, NC, April 2004.

[9] C. Xia, L. C. Pérez, S. Goddard, and R. Sprick, "The effect of stale ranging data on indoor 2-D passive localization: Technical Report #TR-CSE-EE-UNL-2007-1," University of Nebraska-Lincoln, Jan 2007, http://cse.unl.edu/~goddard/Papers/TechReports/TR-CSE-EE-UNL-2007-1.pdf

VPE: Virtual Periodic Execution for Embedded System

Midori Sugaya [1], Yuki Kinebuchi [1], Shuichi Oikawa [2], Tatsuo Nakajima [1]
[1] Department of Computer Science, Waseda University
3-4-1 Okubo, Shinjuku-ku, Tokyo 169-8555, Japan
[2] Deaprtment of Computer Science, University of Tsukuba
1-1-1 Tennodai,Tsukuba

***Abstract*— General purpose operating systems are widely used for advanced embedded system, since they can provide varieties of software components for the applications. They increase the efficiency of development for embedded system. However, they have not yet provided the enough function to completely meet the requirement of real-time and non real-time applications. To solve the problem, we propose an approach named Virtual Periodic Execution (VPE) that could be used to schedule both of real-time and non real-time application with satisfying their practical requirements. In this paper, we present the ideas and the architectures of the VPE system, and its development is in progress.**

I. Introduction

In last several years, information appliances have gained more computing capability than ever, in order to retrieve data from sensors, to process the data, and to control devices. These advanced information appliances require more advanced features like networking and GUI. Those features dramatically complicate the applications and increase their code sizes. For example, popular type cell phone needs 600 million lines for one model. Moreover, the competitive market forces developers to produce these codes in short development time. To develop advanced embedded systems efficiently, software reusability is known to be important in the embedded system area. To increase the software reusability, operating system is expected to provide general API and software components for their applications such as protocol stacks, drivers and GUI libraries. So it is necessary to expand the general purpose operating system like Linux in the embedded system these days. Conventionally, embedded system develper has used tiny real-time operating systems (RTOS), however, the developers who consider the reusability of applications try to switch conventional real-time operating system to general purpose operating system as a platform for developing information appliances.

However, there are several problems to switch operating systems. The first is absorbing the differences of the task models. Conventional RTOS usually use thread model shared address with them. In contrast, general purpose operating system adopts a process model which provides the exclusive virtual address for each process. The second is how to map the domain specific APIs of conventional operating system's to general interfaces like POSIX API. The last one is how to schedule real-time applications which has been running on RTOS to the general purpose operating system with satisfying their real-time requirements.

The first two problems are actually similar and almost solved. There have already been developed libraries and wrappers to absorb the differences of task models or APIs. They use translation techniques that mapped the functions which have the same operations. However, the real-time scheduling problem has not solved because of the practical reason.

In this paper, we focus on the real-time scheduling problem. It should be considerable not only real-time requirement of the RTOS applications, but also the requirement of the non real-time applications that are running on the general purpose operating system. Moreover, we have to satisfy the practical requirements. We propose Virtual Periodic Execution (VPE). The remainder of the paper is structured as follows.

Section II describes the problem and constrained condition which are defined by the requirement of embedded system. The section III presents the VPE (Virtual Periodic Execution) model, section IV shows the example of it. We show the architecture in section V and discuss the matter of concurrency and overhead in section VI. Finally, in section VII, we conclude the paper.

II. The problems and requirements of embedded system scheduling

As we mentioned in previous section, the adjustment of scheduling for applications is significantly important to provide embedded system at the time of switching operating systems. However, there are few practical approaches. In this section, we describe the requirements and conditions to be satisfied to build practical embedded system.

First, we have to consider the requirements of the real-time applications. In advanced information appliances, multimedia applications such as video streaming programs, audio mixers are essential for the products. These applications are defined as the soft real-time applications, which are characterized by temporal constraints that must be satisfied their timing requests. These applications are configured as periodic task by the system.

The second is the responsibility of non-real-time applications. Many of advanced embedded system will provide services with GUI applications. These applications usually wait events from the user through keyboard or other input devices. They are expected to response as quick as possible with minimum latencies. They are scheduled as non-real-time applications, because of no timing constraints and deadlines, and configured as aperiodic task by the system.

To satisfy both of the requirements, developers usually assign static priorities to the transported real-time applications. In general purpose operating system, static priorities are usually assigned to the higher priorities than the dynamic priorities. And most of the applications scheduled with dynamic priorities[1]. If the utilization of periodic tasks is high or no appropriate design, the periodic tasks easily occupied the CPU. As the result, aperiodic tasks which have been scheduled with lower priorities will not have requested responsibility. Actually, the problem is widely recognized and has been tried to be solved in several past years. There are a lot of theories, however, few practical theories could give satisfying result.

In pracitical development of embedded system, in order to avoid latencies of the aperiodic tasks, embedded developers tend to modify the software with ad hoc solutions. For example, if there are latencies of the aperiodic tasks, they will try to adjust the execution time of the periodic task to yield the CPU for the aperiodic task at the time of interrupt event coming from hardwares. They find out these timing problems by simulating the applications several times and correct it. These ad hoc modifications will be the cause of the bugs. Currently, this problem tends to be considered more serious according to the rapid expansion of code sizes of the advanced embedded system. As in the case of application, the scheduler should not be modified. Moreover, scheduler controls all of the applications that running on the system, modifications have more effects for all of the applications. Considering about these demerits, we should avoid the modification both of application and scheduler. Then, our prerequisites constrained conditions should be following.

1) Satisfy the real-time constrain for real-time applications
2) Satisfy the responsibility for non-real-time applications
3) No change of the application and scheduler

III. Virtual Periodic Execution

A. Conventional algorithms

To solve the problem under the constraint three conditions, we propose Virtual Periodic Execution (VPE) system. VPE system can satisfy the prerequisites which we have discussed in previous section.

Before present our algorithm, we introduce the conventional algorithms. As the first, we refer to the rate monotonic algorithm[2]. The algorithm already has been proven the optimized scheduling that maximizes the utilization of a processor capacity with simple prioritizing that shorter period should have higher priority. However rate monotonic algorithm has not any prerequisite to satisfy the requirement of the responsibility of aperiodic tasks. Therefore in practical use, aperiodic tasks will be assigned to use idle time of periodic tasks.

On the contrary, server algorithms such as sporadic server [4] and deferrable server [5] that are based on the rate monotonic algorithm have a distinguish purpose of improving the responsibility of aperiodic task as prerequisite. These server algorithms can satisfy our first two constraint requirement.

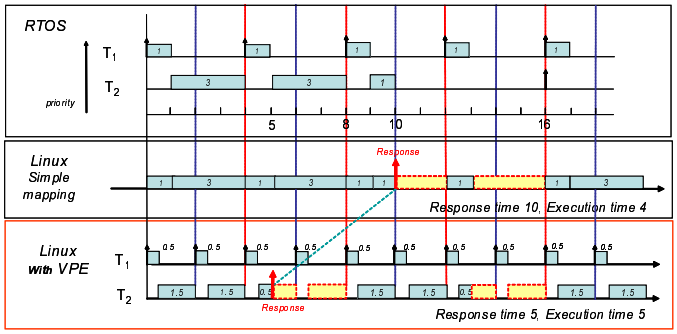


Fig. 1. Example of VPE scheduling

However, from the point of practical development, there are still some problems. If we construct a system which keeps satisfying the timing requirements of periodic tasks and aperiodic tasks, we have to extend a general prioritized scheduler. For example, the server algorithm needs the special facilities for scheduler or middleware to control the aperiodic tasks, such as creating server, and bind non-real-time priority tasks to them. It should impose the development efforts to the developers and risks to change the scheduler. Therefore, they could not satisfy our constrained condition 3.

Compared with the server algorithms, the rate monotonic can satisfy our third constrained condition, since it has no requirements to modify the scheduler that provide simple prioritized scheduling. However, rate monotonic do not refer to the problem of responsibility of aperiodic task. Compared with the demerit of changing application and scheduler, we decide to use rate monotonic algorithm and try to improve the responsibility of aperiodic tasks.

As the result, we propose the VPE system, which can provide the function to improve the responsibility of the aperiodic task still satisfy periodic requirement.

B. VPE system proposal

VPE system provides the function to give a virtual period for periodic tasks. The tasks which are given a virtual period will be scheduled with fine grained periodic time. The guarantee for the periodic tasks depends on the rate monotonic algorithm. Therefore, the periodic tasks are guaranteed and the worst case responsibility of aperiodic tasks is improved. It can satisfy the requirement of periodic and aperiodic responsibility without changing both application and scheduler.

VPE algorithm based on the idea that transforming the period and execution time of tasks. And the VPE system can provide the facility to control the tasks with the VPE algorithm with forcibly divide the period. In past years, Liu showed that transforming period and execution time of the tasks can still satisfy the condition of the rate monotonic algorithm [6]. The definition is used for the purpose of increases the priority of important task, however, we use it to increase the responsibility of aperiodic tasks without overhead of modification of application and scheduler.

In the VPE algorithm, we have to divide the period and

execution time by dividing constant value. We define the dividing constant is k $(0 \leq k)$.

For example, the general theorem of rate monotonic algorithm is shown in (1). The (2) is the theorem that shows the formula of the period (T) and computation time (C) dividing by k . In second theorem, if k is delaminated, the result is the same as the (1). It means, even if we divide both of period and execution time of tasks, the utilization of the system will not decrease.

$$\text{U} = \sum_{i=1}^{m} C_i/T_i \leq m\left(2^{1/m}-1\right) (1)$$
$$U = \sum_{i=1}^{m} (C_i/k)/(T_i/k) \leq m\left(2^{1/m}-1\right) (2)$$

In the real-time research, workload is a metric to measure the worst case utilization of periodic tasks. If worst case workload of periodic tasks will miss the maximum utilization, all periodic tasks can not scheduled without missing deadlines. Usually, the metric is used to ensure the feasibility of the periodic task, we use it to measure the worst case responsibility of the aperiodic tasks in our system. Because they show the result of worst case execution time of periodic task, it is equal to the response time of the aperiodic tasks. The worst case responsibility of aperiodic tasks can be reduced by 1/k, the result is shown in the (3).

$$\text{Workload}_m\,(t) : \sum_{i=1}^{m} \lceil t/(T_i/k) \rceil C_i/k = t (3)$$

In that case, t is multiplied by the $1/k$. That means the worst case response time of aperiodic task will be decreased. We will show the example in the next section.

IV. Example:

Consider an example scenario of two periodic tasks $\tau_1, 2$, where $U_j = C_i/T_i$. Where C is computation time, T is period time and U is utilization. In our example, the deadline of the tasks is the same as the period. It comes from the prerequisite of rate monotonic algorithm.

- $Example1$:
- Task $\tau_1 : C_1 = 1; T_1 = 4; U_1 = 0.25$
- Task $\tau_2 : C_2 = 7; T_2 = 16; U_2 = 0.43$
- $Workload_m\,(10) : \sum_{i=1}^{m} \lceil 10/4 \rceil \times 1 + \lceil 10/16 \rceil \times 7 = 10.$

The total utilization of two tasks is 0.68, which is below the rate monotonic theorem bound for two tasks: $2(2^1/2-1) = 0.828$. Hence, these two tasks are schedulable, that is, they will meet their deadlines if τ_1 is given the highest priority, τ_2 the higher next. At that time, the worst case response time of aperiodic task, which is defined as the workload of the periodic tasks is 10.

- $Example2$:
- Task $\tau_1 : C_1 = 0.5; T_1 = 2; U_1 = 0.25$
- Task $\tau_2 : C_2 = 3.5; T_2 = 8; U_2 = 0.43$
- $Workload_m\,(5) : \sum_{i=1}^{m} \lceil 5/2 \rceil \times 0.5 + \lceil 5/8 \rceil \times 3.5 = 5.$

In $Example2$, we give the virtual period and execution time for the two tasks with dividing constant $k = 2$. The result shows even if the scheduling parameters are divided by k, the utilization are equal to that of the $Example1$'s. However, the worst case response time of aperiodic task is improved to 5. Compared to the $Example1$'s, total 1/2 time is decreased. In our system, non-real-time tasks use the idle time of the real-time tasks, therefore, it means that the responsibility of aperiodic tasks is improved as much as k times.

A. Overhead

The overhead which increase according to the growth of dividing constant k is a problem to consider for our system. In our system, as we increase the k, the cost of the overhead which will be come up with the context switches or interrupts will also increase.

We assume a simple model to calculate the overhead. The computation time C is broken up the two elements. The one is the real executing time defined as C_r of a task, and the other is the additional overhead defined as Co. The relation will be simply modeled as $C = C_o + C_r$.

The time of C_o should be constant and increased with the increasing the number of k. On the contrary, the C_r will not be affected by the number of k. It means the formula will be described as the $C = kC_o + C_r$. It should be taken into the account of theorem (1). The result is derived from (4).

$$\text{U} = \sum_{i=1}^{m} (\text{k}C_o + C_r i)/T_i \leq m\left(2^{1/m}-1\right) (4)$$

When we set a k, we calculate the schedulability with the model which contains real overhead of the task switches. Before calculating the formula, we assumed the rate of the overhead (C_o) and real computation time (C_r) , 1:100 in the Fig.2, 1:1000 in the Fig.3. The y axis shows the utilization of the processor, and x axis showed the number of k. The simulation results showed that, if the difference of (C_o) and (C_r) numbers are large, the growth rate of the utilization is low. And if the initial rate of the C_r/T is low, that also the growth rate of k is low. Now we are trying to collect the data and evaluate the system overheads.

V. Architecture

Based on the presented VPE model, we try to give the architecture of the system. The system will be composed with three layers.

1) Policy layer
2) Simulation layer
3) Enforcement and Control layer

The Fig.4 depicts the whole architecture which contains the three layers. Policy layer accepts the request from the user. It can be selected from supported scheduling policies. The simulation layer provides function to evaluate the parameter, then, choose most appropriate parameter to control the system, and pass it to the next enforcement layer. The enforcement layer provides the function to support the execution and period time. We use the facilities for the enforcement layer Accounting System [7] which provide robust functions to control the task with fine grained time.

VI. Discussion

We still have some topics to discuss at constructing VPE system. First, we have to think about the divided lock problem. In previous sections, we discussed the scheduling of

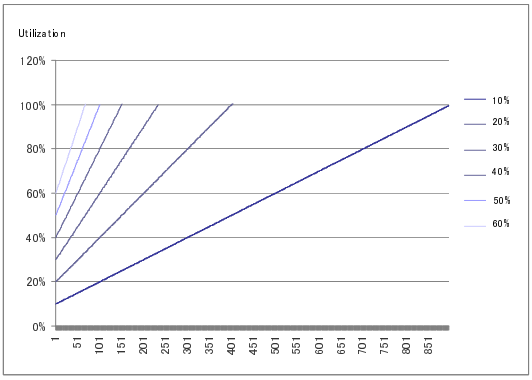


Fig. 2. The case of $C_o : C_r(1 : 100)$

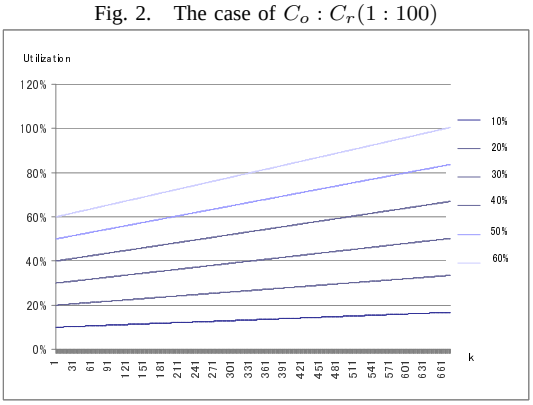


Fig. 3. The case of $C_o : C_r(1 : 1000)$

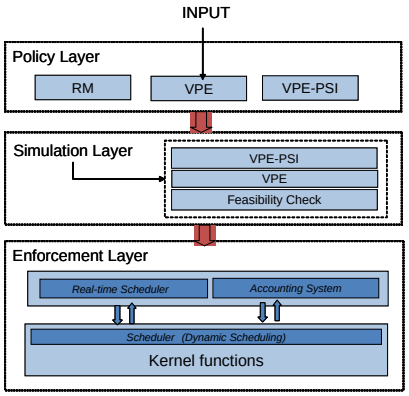


Fig. 4. Architecture of VPE System

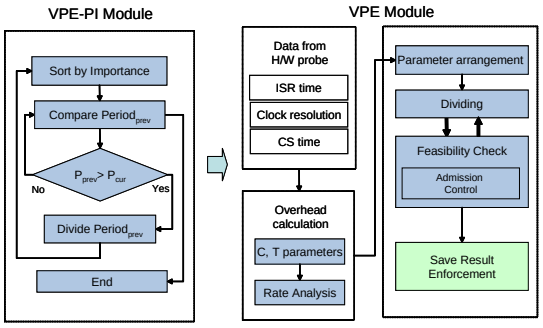


Fig. 5. Module Design in VPE

independent tasks. In an actual system, tasks are always interact with each other. Therefore, the effect of interactions especially under our system should be considered. The VPE system provides function to block a task forcibly for the purpose of dividing the computation time. When a task get into critical section and is blocked by itself, if the forcible blocking is executed at that time, the self blocked task could not wakeup until the forcible blocking is released. We named the problem divided lock problem. We have a prerequisite that the critical section of tasks are very short, therefore, non-blocking or restartable critical section is effective for our implementation. However, there are still problems how to guarantee the prerequisite that the shortness of the critical section. It is not only needed to evaluate our system but also the base operating system should be evaluated. This part of research is in progress.

The second problem is how to decide the most appropriate dividing constant k. As we mention in section IV, the k should be decided using the model which include the overhead of the dividing. However, the practical number of k will also depend on the required responsibility that non-real-time application take good responsibility even under the high utilization of real-time applications.

VII. Conclusion

In this paper, we have proposed Virtual Periodic Execution system. This system provides the generic scheduling system to satisfy the requirements of periodic and aperiodic tasks. Not only to satisfy the scheduling problem, but also try to satisfy the requirements for considering about the efficiency of the development. Our system increase the development efficiency and easy to implement. In embedded area, such efficient scheduling system is on great demand. Our research should have the great impact for them.

References

[1] William Stallings, Operating Systems:(International Edition) 5th, Prentice Hall. July 2004.

[2] C. L. Liu, James W. Layland, Scheduling Algorithms for Multiprogramming in a Hard-Real-Time Envirnment,Journal of the Association for Computing Machinery, Vol. 20, No. 1, January 1973.

[3] Realtime Systems, International Series in Computer Science, Prentice Hall; 1st edition (September 1997).

[4] B. Sprunt, Aperiodic Task Scheduling for Real-Time Systems, Ph.D. thesis, Dep. of Electrical and Computer Engineering, Carnegie Mellon University (1990).

[5] Strosnider, J. K., Lehoczky, J. P., and Sha, L. The deferrable server algorithm for enhanced aperiodic responsiveness in hard real-time environments. IEEE Transactions on Computers 44, 1 (January 1995).

[6] Lui Sha, John Lehoczky, Rangunathan Rajkumar. Solutions for some practical problems in prioritized preemptive scheduling., In Proc. 7th IEEE Real-Time Systems Symposium. IEEE Computer Society Press, 1986.

[7] Midori Sugaya, Shuichi Oikawa, Tatsuo Nakajima: Accounting System: A Fine-Grained CPU Resource Protection Mechanism for Embedded System. ISORC 2006: 72-84.

RTPAW: a Real-Time Power Aware Framework for Wireless Sensor Networks

Emanuele Toscano, Giordano A. Kaczyński, Lucia Lo Bello
RETISNET Lab
Department of Computer Engineering and Telecommunications
University of Catania
Catania, ITALY
{gakaczy, lucia.lobello}@diit.unict.it

***Abstract*— In the Wireless Sensor Networks (WSNs) used in monitoring applications, the need to provide soft real-time traffic with an appropriate QoS clashes with the energy consumption constraints of the nodes, which have to work for long periods without the possibility of replacing their batteries. The paper presents the Real-Time Power Aware Framework (RTPAW), which aims at achieving both objectives. The most significant aspects of the RTPAW are the network architecture, which adopts a cluster-based concept, and the implementation at different levels of the protocol stack of the functions needed to meet the low-power and real-time requirements. In the approach here adopted, a new layer, called the Aggregation Layer, is introduced between the MAC and Routing layers. This layer mainly deals with reducing the amount of energy dissipated, while the Routing layer is entrusted with achieving the desired QoS, in terms of delivery speed, to allow the transmission of soft real-time traffic.**

I. Introduction

A wireless sensor network (WSN) typically comprises a large number of nodes capable of monitoring a certain phenomenon (e.g. temperature, luminosity, etc.), processing the relative data and exchanging it amongst themselves as well as with a base station via a Sink node. The nodes in a WSN are generally located in the proximity of or inside the phenomenon they are monitoring. The environments involved are often remote or hostile to humans and in some cases the nodes are placed in their environment in ways that are far from being ordered and predictable. A WSN therefore has to be autonomous, and able to configure itself automatically and to function without human intervention for as long as possible. The main requirements of a network of this kind are therefore:

- Scalability;
- Low production costs;
- Fault-tolerance;
- Long-lasting autonomy;
- Ability to meet soft real-time constraints.

In order to provide nodes with a long period of autonomy (i.e. low power consumption) and affordable production costs, low-power processors and very small memories are used, but this is not enough, as the amount of energy consumed by communications in WSNs is usually much greater than that used for processing. It is therefore necessary to use protocols that aim at optimizing power consumption, so as to prolong the lifetime of the nodes and therefore that of the system as a whole. However, WSNs are generally used for monitoring applications, which mostly features periodic soft real-time traffic and require enforcing a minimum data delivery speed so as to meet delay constraints. The requirements on power consumption and data delivery speed often clash with each other, and therefore most of the communication protocols developed for WSNs fall into two categories: protocols aiming at minimizing power consumption (e.g. [1], [2] and [3]) and protocols aiming at providing the desired QoS to soft real-time traffic (such as [4] and [5]). This paper proposes the Real-Time Power Aware Framework (RTPAW), which aims at achieving a tradeoff between power consumption and delivery speed exploiting the features of both categories of protocols.

II. Related work and motivation

A. Related work

Some routing protocols, such as LEACH [1] and MECH [3], aim at minimizing power consumption, adopting a hierarchical routing strategy in which a limited number of always active and periodically re-elected nodes, called cluster heads, form a backbone, while the other nodes can remain asleep and only wake up when data is being sent. The cluster heads are elected in rotation following a probabilistic scheme and remain cluster heads for a certain period of time, called a round. Communication between cluster heads and the other nodes in the cluster takes place by means of Time Division Multiple Access (TDMA): a super-frame is created, in which each node has its own time slot. Once data is acquired, the cluster heads transmit it directly to the base station. To reduce the impact of radio interference between different clusters Code Division Multiple Access (CDMA) is implemented, each cluster using a different code.

An approach of this kind is very efficient from the point of view of power consumption, but suffers from scalability problems which make it unsuitable for large networks. In addition, both LEACH and MECH require clock synchronization at a network level, which is only possible for small networks. LEACH assumes that the cluster heads can communicate directly with the base station. MECH does support message forwarding between cluster heads, but does not guarantee any QoS.

Following a different philosophy, other routing algorithms have been developed with the aim of offering WSNs a

certain QoS; examples of these algorithms are SPEED [4] and MMSPEED [5]. Based of geographical routing, which is particularly efficient in networks covering a large geographical extension, they try to guarantee a minimum speed in data delivery. These algorithms were developed on 802.11 and do not target power consumption.

Conversely, the RPAR [6] protocol targets real-time applications and at the same time tries to optimize power consumption, by minimizing the number of deadline misses while keeping power consumption low. The power efficiency of RPAR is achieved by constantly regulating the transmission power. This approach is, however, affected by anomalous behavior in heavy traffic conditions, which tends to favor network congestion.

B. Motivation

Our proposal derives from the need to find a communication technique for WSNs that is as close as possible to LEACH as regards power efficiency and to SPEED as regards QoS.

Another highly desirable characteristic is the ability to use, where possible, standard protocols or established protocols that have been widely studied [7]. For this reason in this work we chose to use the 802.15.4 standard ([8],[9]) for the MAC layer, whereas for the routing layer we are currently envisaging an adapted version of SPEED.

III. The RTPAW framework

A. The basic idea

As a WSN may comprise thousands of nodes distributed over a wide geographical area, it is advisable to use advanced routing techniques that can offer a good QoS in terms of both (soft) real-time constraints meeting and fault-tolerance. It is therefore useful to exploit routing techniques such as SPEED [4] or MMSPEED [5]. Moreover, considering that the phenomenon to be monitored may last weeks or months, it is also necessary to have routing protocols that are efficient from the point of view of power consumption as well. In order to save power it is necessary to allow nodes to go to sleep periodically.

One way to improve the routing protocols mentioned above, which can be applied to a network with a large number of nodes, is to send certain nodes to sleep periodically, as in LEACH [1] and subsequent versions. The geographical routing is not based on the physical address of a node but on its position: if there are other nodes geographically very close to each other, not all of them have to be active at the same time. It is therefore possible to envisage an alternation between periods of activity and sleep periods that could follow a number of strategies. The simplest option is to establish, on the basis of the density of the nodes in the network, a duty cycle for the nodes (without the need for a global time synchronization), so as to guarantee with a certain probability that there will always be an active node to perform routing operations.

Another strategy would be to allow neighboring nodes to agree on the periods of activity via signaling messages. In that case it would be possible to guarantee a fairer distribution of the periods of activity, with a view to minimizing the power consumed by the nodes and at the same time to enhance the QoS on the network. This is the strategy adopted in our framework to achieve the desired objectives. Details will be given in the following.

B. Network architecture proposed

The RTPAW architecture inherits the main features of [1], but with substantial differences.

Similarly to [1] the nodes are grouped into clusters, which we call Aggregated Units (AUs) here. However, the AU structure here is different from that of the clusters in [1]. Here, the nodes in an AU belong to three different categories:

- Cluster Head (CH);
- Relay Node (RN);
- Cluster Node (CN).

In each AU there is one CH, one RN and a varying number of CNs, as shown in Fig. 1. The CH has the task of collecting data from the sensor nodes belonging to the cluster (the CNs) and periodically transmit it to the RN. The task of the latter is to forward the data to other RNs or the Sink node. In this architecture, therefore, the CH handles transmission within the cluster, while the RN handles transmission outside the cluster.

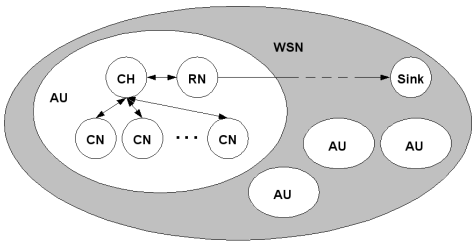


Fig. 1. Network architecture.

There are three different types of traffic:

1) Communications between CH and the CNs;
2) Communications between CH and RN;
3) Communications between RN and RN or RN and Sink.

C. Protocol architecture of the RTPAW framework

The RTPAW protocol architecture here proposed has an Aggregation Layer which acts as a mediator between the MAC and Routing layers for the combined handling of energy awareness and real-time support. The Aggregation Layer deals with creating and managing the AU and transmitting the first two types of traffic described in subsection B, i.e. 1 and 2. The Routing Layer lies above the Aggregation Layer and forwards packets between AUs, thus handling the third type of traffic (i.e. 3).

In this architecture, the MAC layer collaborates closely with the Aggregation layer to provide the Routing layer with a uniform view of the set of sensor nodes making up the AU. The basic addressable entity in the Routing layer is therefore not the single WSN node but the single AU.

The Aggregation layer is split into two sub-layers, with the lower part (called MAC-dependent) which strongly depends on the MAC protocol used and represents an extension of

the basic functions needed to implement the level above. On the other hand, the MAC-independent part implements a pre-established set of primitives operating as a shared interface between the MAC-dependent and MAC-independent layers. The task of the Aggregation Layer is to create and handle the cluster and the aim is to reduce consumption by scheduling periods of activity and sleep periods. As mentioned previously, the MAC level and the MAC-dependent part of the Aggregation Layer work closely, as the activity periods may coincide with certain states of the MAC Layer. For example, if TDMA is used for transmission inside a cluster, it is possible to make nodes go to sleep during time slots other than their own.

Above the Aggregation Layer virtually any routing algorithm providing a certain QoS can be used. The algorithm will operate viewing the whole AU as a single node.

The advantages of the proposed architecture are:

- Reduced power consumption, dependent on the efficiency of the aggregation protocol used;
- Advanced QoS management, depending on the efficiency of the routing protocol used;
- Fault-tolerance, also depending on the aggregation protocol used: as the routing unit is the whole AU rather than the single node, the AU will continue to live even if several of its nodes cease to function.

IV. The protocols used

A. Physical and MAC Layer

The Physical and MAC layers adopted here are the 802.15.4 standard ones [8],[9]; to guarantee greater scalability and fault-tolerance, the non-beacon enabled mode has been chosen. This transmission mode is generally offered by the widely available ZigBee modules. To avoid interference between neighboring nodes operating in different clusters, it is possible to implement a cell-based architecture exploiting the 16 different channels offered by the standard in the 2.4 GHz bandwidth. The idea is to make the radio cells at the Physical level coincide with the clusters of the Aggregation layer.

B. Aggregation layer

The Aggregation layer handles data transmission in a single cluster. The communication protocol for this layer follows the philosophy of LEACH [1], but with considerable differences given by both the need to use a different MAC layer and the desire to improve some of its aspects.

As in LEACH, a superframe is created, in our case at the Aggregation level, and the nodes belonging to each cluster send their data to their CH in their time slots. It should be noted that the Aggregation Layer superframe is not mapped on the 802.15.4 superframe, but is created at a higher level using the 802.15.4 non-beacon enabled mode. As mentioned previously, in our approach there are not only cluster heads (CH) and nodes belonging to a cluster (CN), but also relay nodes (RN). A CH and an RN are elected in each cluster; the former collects data from the other nodes (except the relay node) whereas the latter forwards packets from one cluster to another. It is necessary to provide a period of time, very short if compared to the duration of the whole superframe, in which the CH and RN nodes synchronize their data. The CH and RN must always be active while all the others can go to sleep and only wake up to receive synchronization signals from the CH or to transmit their data during the assigned time slot. As RN and CH nodes consume much more power than the other nodes, it is necessary for them to be elected in rotation in such a way as to balance the power consumption over the network. The CH and the RN could coincide only in the exceptional case in which the AU comprises a single node. In this case the CH acts as an RN and periodically forwards its own data.

The transmission power of the nodes varies according to whether they are RNs or not. RNs have to transmit at a power high enough to reach either the Sink node in a single hop or the RN of the next hop. The transmission power of the other nodes has to be gauged according to the density of the nodes and the number of clusters. Transmitting at too low a power could lead to premature node isolation, whereas using too high a power would not only mean a useless waste of energy, but would also limit the signal quality due to greater interference between nodes belonging to different clusters.

Unlike LEACH, we do not use CDMA to prevent interference between adjacent clusters, but Frequency Division Multiple Access (FDMA): each cluster for intra-cluster communications transmits on a different channel from that of the neighboring clusters. Selection of the transmission channel can be automatic during initialization of the nodes, using the Energy Detection scan (ED_scan) procedure defined in [8] and [9], or set according to the position of a node. Another important difference between RTPAW and LEACH is that whereas the latter required network-wide clock synchronization, our protocol requires synchronization at the AU level only.

The normal functioning of the protocol is divided into three different phases, i.e., initialization, election and data transfer. The initialization phase is executed when a node is first activated, whereas election and data transfer alternate, not necessarily at regular intervals. In the following a brief description of the three phases is given.

1) Initialization: The main aim of the initialization phase is the definition of the cellular architecture. In our scenario we assume that all the nodes know their own position and that they have been randomly arranged with a relatively uniform density. It is therefore possible in quite a simple and efficient manner to create a homogeneous cellular structure, like a grid subdividing the area being monitored into a number of small uniform regions, each hosting a cell. Channel selection can also be based on the position of a node (and the cell it belongs to) or can be determined following an ED_scan in such a way as to minimize radio interference.

The next step is the first election, during which the CH is elected. If all the nodes are the same, the CH can be any of the nodes equipped with the greatest amount of energy. Then the CH elects the RN (as described below) and sends the transmission schedule to the nodes belonging to the cluster.

2) Election: In cluster-based protocols integrating a cluster head rotation mechanism, whenever a CH is elected it is generally necessary to reconstruct the whole cluster. This gives the network flexibility and adaptability to changes in

environmental conditions. However, in the presence of tight deadlines, or when constant updating of the variables being monitored is needed, this may lead to excessive degradation in the QoS. It was therefore decided to separate the distributed algorithm for the first election from the one used later on, which is centralized. In the latter case, the CH at a certain point (after a pre-established time or because its remaining power has dropped beneath a certain threshold) autonomously decides which node is to be its successor and notifies the node involved. From the next transmission cycle the new CH will start operating. The decision regarding the next CH is based on the residual energy of the nodes in the cluster, as signaled in the frame that nodes send during normal transmission phases.

Election of the RN is different. It should be as independent as possible from the election of the CH, because the RN battery could run out more rapidly. However, an independent election would require complex management algorithms which would cancel out all the benefits. RTPAW therefore proposes a hybrid solution. The CH normally elects the RN autonomously. An RN whose power has dropped beneath a certain threshold notifies the CH during their synchronization phase. The CH consequently chooses as the next RN whichever of the nodes with the greatest amount of energy has the strongest signal (it is advisable for CH and RN to be close to each other, as the synergy between them gives a good QoS). The first information can be obtained with a negligible ovehead, inserting it in the packets that CN nodes send to their relevant CH, while the second one can be directly devised by the hardware.

3) Data transfer: Data transfer in a cluster follows a pre-established synchronized sequence which emulates a superframe structure in the Aggregation Layer. In this way, although simple CSMA/CA at the MAC level is used, it is possible to avoid collisions.

The transmission sequence making up the superframe starts with a beacon frame from the cluster head, used to synchronize transmissions in the cluster. It is important to point out that the beacon frame is generated by the Aggregation Layer and so is not a 802.15.4 beacon frame. It is used for synchronization with the CNs, so all the CNs have to receive it.

Following this there are time slots during which the CNs can transmit their data to the CH, using TDMA. Each CN is assigned a time slot during which it can communicate with the CH with no collisions. During all the time slots assigned to the other nodes, a CN can go to sleep. It must, however, wake up again in time to receive the next beacon frame.

The last section of the superframe is for synchronization between the CH and the RN.

In the meanwhile the RNs form a backbone of nodes that are always active in forwarding packets to the Sink node. They communicate over a single dedicated channel, so during the synchronization phase it is necessary to switch channels temporarily. When the RN acquires the updated CN data from the CH, it forwards it as defined by the Routing Layer. Only RNs can forward data so they are the only nodes to execute the routing algorithm.

C. Routing layer

As the Routing layer is located above the Aggregation layer, packets are not addressed to single nodes, but to single AUs. So, the only task of the routing algorithm is to forward packets from a source AU to their final destination, usually the Sink node. The scenario RTPAW was devised for is one in which the WSN comprises a large number of nodes and may cover a wide area. For this reason, although the underlying Aggregation layer contributes towards increasing the scalability of the network, the algorithm used for routing between the AUs has to be able to handle large network without any difficulty. In addition, it is advisable to use a routing algorithm that is as much fault-tolerant as possible. As said before, the presence of an underlying Aggregation layer considerably reduces the impact of faults occurring in single nodes. Finally, the routing algorithm has to make it possible to achieve the desired QoS, which in our case is delivery speed. A routing algorithm which possesses all these features is SPEED [4]. For this reason, in RTPAW a SPEED-inspired approach is used.

V. Conclusions and On-going work

The RTPAW framework has been devised to be highly scalable, flexible and modular and to allow various existing routing protocols to be adopted. On-going work deals with simulation of the network architecture and protocol stack of the RTPAW framework using the well-known NS-2 [10] tool. The aim is assessing the RTPAW performance, also in comparison with other approaches, e.g. RPAR [6]. A second activity targets the implementation of RTPAW on ZigBee COTS modules.

References

[1] W. Heinzelman, A. Chandrakasan, and H. Balakrishnan, "Energy-Efficient Communication Protocols for Wireless Microsensor Networks (LEACH)," in *Proc. of the 33rd Hawaii International Conference on Systems Science*, vol. 8, 2000, pp. 3005–3014.

[2] A. Manjeshwar and D. Agrawal, "TEEN: a Routing Protocol for Enhanced Efficient in Wireless Sensor Networks," in *Proc. of the 15th Internat Parallel and Distributed Processing Symp*, 2001, pp. 2009–2015.

[3] R.-S. Chang and C.-J. Kuo, "An Energy Efficient Routing Mechanism for Wireless Sensor Networks," in *Proc. of the 20th Internat. Conf. on Advanced Information Networking and Applications*, 2006.

[4] T. He, J. Stankovic, C. Lu, and T. Abdelzaher, "SPEED: A Stateless Protocol for Real-Time Communication in Sensor Networks," in *Proc. IEEE Int'l Conf. Distributed Computing Systems*, 2003, pp. 46–55.

[5] E. Felemban, C.-G. Lee, and E. Ekici, "MMSPEED: multipath Multi-SPEED protocol for QoS guarantee of reliability and. Timeliness in wireless sensor networks," in *IEEE Transactions on Mobile Computing*, vol. 5, no. 6, June 2006, pp. 738–754.

[6] O. Chipara, Z. He, G. Xing, Q. Chen, X. Wang, C. Lu, J. Stankovic, and T. Abdelzaher, "Real-time Power-Aware Routing in Sensor Networks," in *14th IEEE Internat. Workshop on Quality of Service*, 2006, pp. 83–92.

[7] B. Bougard, F. Catthoor, D. Daly, A. Chandrakasan, and W. Dehaene, "Energy efficiency of IEEE 802.15.4 standard in dense wireless microsensor networks: modeling and improvement perspectives," in *Proc. of Design, Automation and Test in Europe*, vol. 1, 2005, pp. 196–201.

[8] "IEEE 802.15.4 Standard Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) specifications for Low-Rate Wireless Personal Area Networks (LR-WPANs), IEEE Standard for Information Technology, IEEE-SA Standards Board," 2003.

[9] "IEEE 802.15.4-2006 Standard Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) specifications for Low-Rate Wireless Personal Area Networks (LR-WPANs), IEEE Standard for Information Technology, IEEE-SA Standards Board," 2006.

[10] "The Network Simulator. http://www.isi.edu/nsnam/ns."

Improving the real-time capabilities of IEEE 802.11e through a Contention Window Adapter

Salvatore Vittorio, Giordano A. Kaczyński, Lucia Lo Bello
RETISNET Lab
Department of Computer Engineering and Telecommunications
University of Catania
Catania, ITALY
{gakaczy, lucia.lobello}@diit.unict.it

***Abstract*— The paper reports on-going work on enhancing the support provided to soft real-time traffic by the IEEE 802.11e protocol. The proposed solution is based on a mechanism, called a Contention Window Adapter (CWA), which dynamically changes the contention window size of the different Access Categories defined by the IEEE 802.11e standard according to the workload conditions of the wireless network. The aim is improving throughput and deadline miss ratio for soft real-time traffic flows under heavy traffic conditions, i.e. when the offered workload approaches the available bandwidth. Preliminary results are presented, which show the potential of the proposed approach.**

I. Introduction

To provide support for QoS in 802.11 networks, the IEEE Task Group E was set up and in 2005 the final version of the 802.11e standard was finally published [1]. The IEEE 802.11e defines two new access mechanisms, i.e the Enhanced Distributed Channel Access (EDCA) and the Hybrid Coordination Function (HCF) Controlled Access (HCCA). This paper addresses IEEE 802.11e based networks working according to the EDCA mode, which extends the IEEE 802.11 Distributed Coordination Function (DCF) [2].

The IEEE 802.11 DCF operating mode, which is widely accepted and used in commercial products, is based on the Carrier-Sense Multiple Access with Collision Avoidance (CSMA/CA) protocol. Before starting transmission, a node listens to the channel for a time called a Distributed Inter-Frame Spacing (DIFS), to assess whether the channel is idle or not. If the channel is idle, in order to reduce the probability of collisions with other nodes trying to access it at the same time, each node generates a random backoff interval. Each node decreases its backoff counter as long as the wireless channel is sensed to be idle during a DIFS. If the counter has not reached zero and the channel becomes busy again, the backoff counter is frozen and reloaded as soon as the channel becomes idle again for a DIFS. When the backoff interval is over, if the channel is still idle, transmission starts. The random backoff interval, expressed as a number of time slots, is generated in the set {0,CW - 1}, where CW denotes the size of the contention window. The initial value of the contention window is CW_{min}.

In the event of an unsuccessful transmission (due to packet collisions or losses) CW is doubled up to a maximum value CW_{max}. After experiencing the maximum number of collisions, a packet is dropped. In the event of a successful transmission, the CW value is reset to CW_{min} before the random backoff interval is selected. In DCF, all nodes have the same opportunity to access the channel.

The IEEE 802.11e EDCA is similar to the DCF. However, the EDCA differentiates traffic into four Access Categories (ACs), mapped in the priorities defined by the 802.1D standard [3] as follows:

- AC_BK (background category) for priorities 1 and 2;
- AC_BE (best-effort category) for priorities 0 and 3;
- AC_VI (video category) for priorities 4 and 5;
- AC_VO (voice category) for priorities 6 and 7.

AC_VO is the highest priority category, while AC_BK is the lowest. Each AC has its own queue and parameter set, which includes the minimum Contention Window size (CW_{min}), the maximum Contention Window size (CW_{max}), the Arbitration Inter-Frame Space (AIFS), and the Transmission Opportunity limit (TXOPlimit). CW_{min} and CW_{max} determine the size of the CW, with CW set as CW_{min} at the beginning of a backoff procedure.

When deferring, a station needs to wait for an AIFS. The smaller CW_{min}, CW_{max}, and AIFS are, the greater the chances for a node gaining access to the medium are. Finally, TXOPlimit defines the minimum time interval that a node has to wait when trying to transmit multiple frames belonging to the same AC (this is called Contention Free Burst, or CFB).

In this paper, we propose the Contention Window Adapter (CWA), a mechanism to enhance IEEE 802.11e EDCA performance by adapting the Contention Window (CW) of the different ACs to the current offered load over the wireless network. One interesting feature which will be shown is that, even in the presence of traffic from the highest priority class only (i.e. AC_VO), according to the amount of real-time traffic and the size of packets in this class, the real-time performance can rapidly and significantly deteriorate with growing workloads. This is due to the CW_{min} and CW_{max} settings provided by the standard, i.e. 7 and 15, respectively, which determine a narrow range of backoff values for the packets in the AC_VO AC. Here in this paper we will shown that it is beneficial to adapt CW_{min} and CW_{max} to allow for

a larger spectrum of backoff values, thus reducing the number of collisions inside the AC_VO AC. The proposed mechanism does not change the IEEE 802.11e protocol, but introduces a technique to reduce the contention overhead and therefore to use the channel more efficiently for real-time flows, especially when the traffic load approaches saturation conditions.

However, it has to be considered that when different ACs are to be supported, if the CW_{max} of the highest priority class is increased, the CW_{min} of the lower priority ACs has to be accordingly set, so as to enforce the different QoS support offered to the each AC according to the standard. The CWA therefore dynamically tunes the $\{CW_{min}, CW_{max}\}$ range of the different ACs defined in the IEEE 802.11e standard in order to reduce the potential interference (in terms of collisions) that real-time traffic, here mapped into the highest priority class AC_VO, could suffer from other lower-priority ACs.

The paper is organized as follows. In Sect. 2 related work is addressed together with the motivation for our paper. Sect. 3 describes the CWA mechanism. In Sect. 4 preliminary simulation results obtained using the Network Simulator version 2 are shown. Finally, in Sect. 5, conclusions and on-going work are outlined.

II. Related work

With the swift development of 802.11 WLANs, together with the increased need to provide QoS support in such networks, in the last years many research work addressed the new IEEE 802.11e protocol.

A number of studies have targeted analytical performance models in order to evaluate the impact of changing the various parameters of EDCA. Xiao [4], for example, extends the Bianchi [5] model, implementing EDCA by means of 3-dimensional Markov chains and analyzing network behaviour for CWs of various sizes, but not including the effects of variations in the AIFS of the various ACs. Kong [6] also uses 3-dimensional Markov chains to characterize the procedures of the various ACs with variations in both the CWs and the AIFS. These works have shown the effectiveness of changing CW depending on the network load.

Recent work describing mechanisms for CW tuning include [7] and [8]. The work [7] introduces an approach called AEDCF, that does not implement a mechanism to vary the range of CWs, but calculates an ideal CW on the basis of the network load estimated according to the number of collisions experienced by the transmitted frames. Once the ideal CW is known, the current CW ($CW_{current}$) for the next frame is set by taking whichever is the lower between the minimum CW (CW_{min}) of the AC the frame belongs to and the ideal CW. The approach is shown outperforming EDCF, the IEEE 802.11e pre-standard distributed medium access mechanism. However, it uses a parameter named Persistent Factor, that was present in an earlier version of the IEEE 802.11e standard, but does not appear in the final version.

The approach proposed in [8], AEDCA, estimates network congestion by using the value of the current CW (i.e. that of the last frame sent). The distance between the current CW and the CW_{min} is compared to the maximum distance between CW_{max} and CW_{min} for the relevant AC, deriving a parameter that is utilized to calculate the new CW for the next frame to be transmitted. The AEDCA approach, like the AEDCF one, does not provide for changing the values of CW_{min} and CW_{max}, but simply chooses the best one in that range.

Instead of setting the CW to an optimal given value in the $\{CW_{min}, CW_{max}\}$ range defined by the standard, the CWA mechanism here proposed adjusts the range of the current CW (i.e. the values of CW_{min} and CW_{max}) on the basis of information on the newtork workload collected during a time interval. This allows to have a CW adapted to the current network status, not limited by the bounds defined by the standard, which proved to be inappropriate in many network load conditions, as will be pointed out in the next Section.

III. The Contention Window Adapter

The aim of the CWA is enhancing real-time performance of the highest priority traffic, AC_VO, while maintaining a better QoS than the other ACs. As stated before in Sect. 1, CWA adapts CW_{min} and CW_{max} in order to reduce the number of collisions inside the AC_VO AC.

In order to clearly explain the basic principle of CWA, let us consider the results shown in Table I. They refer to tests run under ns-2 [9] with a 11 Mbps network made up of 20 stations, each generating AC_VO packets of 160 Bytes with a period of 20 ms, giving an overall workload of 1280 Kbps. With such small packets and high transmission rate, the AC_VO class obtains poor performance when the setting defined by the IEEE 802.11e standard, i.e. $CW_{min}[AC\_VO]$ = 7 and $CW_{max}[AC\_VO]$ = 15 are used. This is because in these conditions the AC_VO class is highly congested [10].

In the 6 different sets of simulations reported in Table I the CW_{min} and CW_{max} were statically set at the beginning of each experiment.

TABLE I

Effects of varying CW_{min} and CW_{max} on AC_VO performance.

CW_{min}	CW_{max}	Average delay (s)	Average jitter (s)	Throughput (%)
7	15	0.01699	0.01249	62.7
7	31	0.01811	0.01760	80.9
15	31	0.01701	0.01596	89.7
15	63	0.00970	0.00724	98.2
31	63	0.00750	0.00490	99.7
31	127	0.00790	0.00574	100

The results show that, although all the packets which are delivered arrive on time (i.e. they meet the 20 ms deadline), the throughput is significantly higher for settings such as $CW_{min}[AC\_VO]$ = 15, $CW_{max}[AC\_VO]$ = 31 onwards than with the default settings $CW_{min}[AC\_VO]$ = 7 and $CW_{max}[AC\_VO]$ = 15 provided by the IEEE 802.11e protocol. In addition, the results show that, with wider contention windows, the delay and jitter experienced by RT packets

is reduced. This is due to the smaller number of collisions experienced by RT packets thanks to the broader range of backoff values.

From the various tests run it also emerged that a good way to reduce the collision probability is to vary CW_{min} and CW_{max} by doubling both of them. These results were also confirmed by other tests run with a greater number of stations and different workload conditions. For this reason we will henceforward adopt settings such as $CW_{min}[AC\_VO]$ = 7, $CW_{max}[AC\_VO]$ = 15 or $CW_{min}[AC\_VO]$ = 15, $CW_{max}[AC\_VO]$ = 31, etc.

The above considerations led to the development of the CWA, which is based on the idea of dynamically varying CW_{min} and CW_{max} according to the network load. The parameter used in the CWA to assess the network load is the ratio (r_c) between the number of collisions (n_coll) affecting the highest priority packets AC_VO and the total number of packets sent (n_pkt_sent) in the AC_VO AC during a given observation interval Δt:

$$r_c = \frac{n\_coll(AC\_VO)}{n\_pkt\_sent(AC\_VO)} \tag{1}$$

This parameter is an index of the level of congestion on the network seen by the AC_VO class. In order to minimize the bias against transient collisions, an Exponentially Weighted Moving Average (EWMA) estimator was used. In a generic i interval, the value of r^i_{avg}, to be used by the algorithm, is updated in the following way:

$$r^i_{avg} = (\lambda - 1) \times r^i_c + \lambda \times r^{i-1}_{avg} \tag{2}$$

The CWA takes this parameter into account when adapting the CW_{min} and CW_{max} of the various classes, every Δt, as follows:

```
procedure Contention Window Adapter
    if (n_pkt_sent(AC_VO) != 0) then
        r_avg := EWMA(n_coll[AC_VO]/n_pkt_sent[AC_VO])
        if (r_avg ≤ α) then
            call procedure Decrease()
        else if ((α < r_avg) && (r_avg ≤ β)) then do nothing
        else if ((β < r_avg) && (r_avg ≤ γ)) then
            call procedure Increase(1)
        else if (r_avg > γ) then
            call procedure Increase(2)
        end if
    end if
end procedure
```

In Sect. IV, the results obtained with α = 0.2, β = 0.6, γ = 2 and Δt = 0.3 s are presented. These boundary values have been heuristically selected through a broad set of simulations run with varying parameter sets under ns-2 [9].

To maintain the differentiation between the different traffic classes, in the CWA an increase in the CW_{min} and CW_{max} window for the AC_VO class has to correspond to a cascaded increase in the CW_{min} and CW_{max} for the other ACs, i.e. AC_VI, AC_BE and AC_BK. This cascade mechanism used to update the CW of the various ACs is shown in the two pseudo-code procedures below.

```
1. procedure Increase(n)
2.     repeat n times
3.         if (CWmax[AC_VO] != 63) then
4.             CWmax[AC_VO] := ((CWmax[AC_VO] + 1) × 2) − 1
5.             CWmin[AC_VO] := ((CWmin[AC_VO] + 1) × 2) − 1
6.             CWmax[AC_VI] := ((CWmax[AC_VO] + 1) × 2) − 1
7.             CWmin[AC_VI] := CWmax[AC_VO]
8.             CWmin[AC_BE] := CWmax[AC_VI]
9.             CWmin[AC_BK] := CWmax[AC_VI]
10.        else if (CWmax[AC_VI] < 511) then
11.            if (CWmax[AC_VO] != 63)
11.                CWmax[AC_VO] := ((CWmax[AC_VO]+1)×2)−1
12.                CWmin[AC_VO] := ((CWmin[AC_VO]+1)×2)−1
13.            end if
14.            CWmax[AC_VI] := ((CWmax[AC_VI] + 1) × 2) − 1
15.            CWmin[AC_BE] := CWmax[AC_VI]
16.            CWmin[AC_BK] := CWmax[AC_VI]
17.        end if
18.    end repeat
19. end procedure
```

To give an example on how the increment is done, let us consider a $CW_{max}[AC\_VO]$ equal to 15. Formula on line 4 of the Increase procedure will give a value of 31. In turn, a value of 31, would give 63. Thus, the CWA shifts the range of CW moving through the values defined in the IEEE 802.11e standard [1]. Here the value 63 has been chosen as an upper bound for $CW_{max}[AC\_VO]$ in order to not compromise the performance of this class with excessively long backoff times.

```
procedure Decrease
    if ((CWmin[AC_VI] < 127) and
        (CWmax[AC_VO] > 15)) then
        CWmax[AC_VO] := ((CWmax[AC_VO] + 1)/2) − 1
        CWmin[AC_VO] := ((CWmin[AC_VO] + 1)/2) − 1
        CWmax[AC_VI] := ((CWmax[AC_VO] + 1)/2) − 1
        CWmin[AC_VI] := ((CWmin[AC_VO] + 1)/2) − 1
        CWmin[AC_BE] := CWmax[AC_VI]
        CWmin[AC_BK] := CWmax[AC_VI]
    end if
    else if (CWmin[AC_VI] > 63)
        CWmax[AC_VI] := ((CWmax[AC_VO] + 1)/2) − 1
        CWmin[AC_VI] := ((CWmin[AC_VO] + 1)/2) − 1
        CWmin[AC_BE] := CWmax[AC_VI]
        CWmin[AC_BK] := CWmax[AC_VI]
    end if
end procedure
```

The Decrease procedure makes the opposite of the Increase one, shifting backwards the values of CW_{min} and CW_{max} of AC_VO and of the rest of the ACs.

IV. Preliminary results

Here preliminary results are shown which outline the benefits of CWA. Extensive simulations using ns-2 [9] with the TKN 802.11e patch ([11],[12]) were performed.

In the scenario addressed here, the nodes communicate with an Access Point at 11 Mbps. Simulations showed that if the CW size is appropriately set, RT traffic performance is considerably improved using CWA. More specifically, the probability of a packet of a given AC colliding with another packet belonging to the same AC is reduced. Here we show this with particular reference to the AC_VO class.

We considered a scenario in which a number of nodes transmit RT traffic towards a Base Station (BS). In a factory automation scenario the BS could be, for example, a PLC receiving field variables from a number of sensors. The type of traffic here considered is periodic, with a period of 20 ms and a frame size of 45 bytes, which means 18 Kbps per station. Simulations were performed with a growing number of RT stations, from 1 to 25. With 20 stations or more and such small packets, the network reaches saturation around 500 Kbps [10]. All the simulations were run for 50 s.

Fig. 1(a) compares the throughput obtained with and without CWA, while Fig. 1(b) compares the deadline hit ratio in the two cases (a 20 ms deadline was chosen). The improvement obtained with the CWA is due to the considerable reduction in the number of collisions involving the RT frames.

The greatest benefit concerns the throughput obtained, which, using CWA drops only in the scenario with 23 or more stations, as at that point the network reaches saturation. However, although the deadline hit degrades, the throughput is still much higher than the one observed in the same conditions when the CWA is not used.

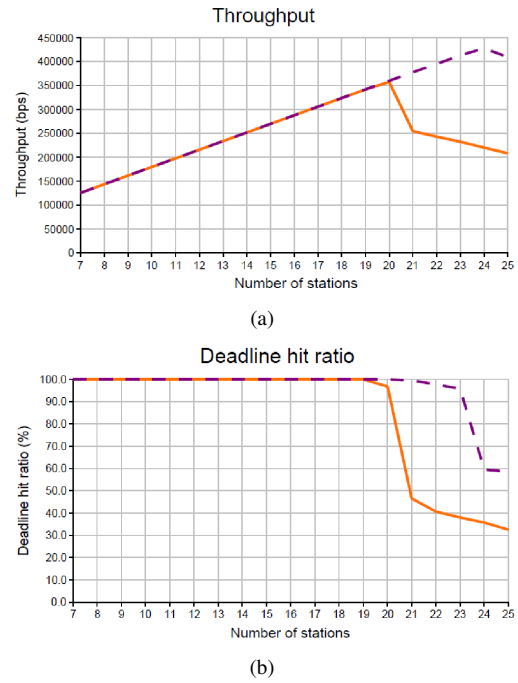


Fig. 1. (a) Total throughput with varying number of stations. (b) Deadline hit ratio, with a 20 ms deadline. Dashed lines refers to results with CWA, while solid refers to results without CWA.

V. Conclusion and on-going work

This paper has addressed an approach to enhance the soft real-time performance of the IEEE 802.11e protocol. The proposed solution, called CWA, is a mechanism which tunes the Contention Windows of the different Access Categories defined by the IEEE 802.11e standard, adapting the values of CW_{min} and CW_{max}. Preliminary results have shown the effectiveness of using CWA for soft real-time traffic, achieving higher throughput and consequently a much lower number of frame loss as compared with the standard behavior of IEEE 802.11e. A lower deadline miss ratio is also obtained.

On-going work is addressing several aspects. Firstly, the problem of using CWA in the presence of stations that do not have soft real-time traffic: if a station tries to transmit only traffic with a priority other than the highest (AC_VO), the CWA cannot react on the basis of the collisions affecting that type of traffic, as ratio is defined for the AC_VO class. On-going work is addressing this problem and possible enhancements of the algorithm proposed. Secondly, we are analyzing the effect of using CWA for lower-priority ACs, and for bigger frame sizes. Both aspects are in development and the results obtained so far are encouraging. We also plan to make comparisons with other techniques in the literature. In parallel, we are also investigating the implementation of CWA on COTS network boards, in particular on the Atheros chipset 5212 equipped with the MADWiFi open source driver [13].

References

[1] "IEEE 802.11e-2005, Medium Access Control (MAC) Quality of Service Enhancements," 2005.

[2] "IEEE 802.11, Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications," 1999.

[3] "IEEE 802.1D-2004, Media access control (MAC) Bridges," 2004.

[4] Y. Xiao, "Performance analysis of IEEE 802.11e EDCF under saturation condition," in *Proc. of IEEE ICC*, 2004, pp. 170–174.

[5] G. Bianchi, "Performance analysis of the IEEE 802.11 distributed coordination function," in *IEEE Journal on Selected Areas in Communications*, vol. 18, 2000, pp. 535–547.

[6] Z. N. Kong, D. H. K. Tsang, B. Bensaou, and D. Gao, "Performance Analysis of IEEE 802.11e Contention-Based Channel Access," in *IEEE Journal on Selected Areas in Communications*, vol. 22, 2004.

[7] L. Romdhani, Q. Ni, and T. Turletti, "Adaptive EDCF: Enhanced Service Differentiation for IEEE 802.11 Wireless Ad-Hoc Networks," in *Wireless Communications and Mobile Computing*, 2004.

[8] J. Sawaya, B. Ghaddar, S. Khawam, H. Safa, H. Artail, and Z. Dawy, "Adaptive Approach for QoS Support in IEEE 802.11e Wireless LAN," in *Wireless Mobile*, 2005.

[9] "The Network Simulator. http://www.isi.edu/nsnam/ns."

[10] J. Jun, P. Peddabachagari, and M. Sichitiu, "Theoretical Maximum Throughput of IEEE 802.11 and its Applications," in *Proc. of the 2nd IEEE Int. Symp. on Net. Comp. and Applications (NCA03)*, 2003, pp. 249–256.

[11] S. Wiethoelther and C. Hoene, "Design and Verification of an IEEE 802.11e EDCF Simulation Model in ns-2.26," in *TR TKN-03-019*. Telecommunication Network Group, Technische Universitat Berlin, 2005.

[12] S. Wiethoelther, M. Emmelmann, and C. Hoene, "TKN EDCA model for NS-2," in *TR TKN-06-003*. Telecommunication Network Group, Technische Universitat Berlin, 2003.

[13] "MADWiFi - Atheros Multiband Driver for WiFi. http://madwifi.org."