RTSS 2013 Organizing Committee

General Chair
Chenyang Lu, Washington University, USA

Program Chair
Michael González Harbour, Universidad de Cantabria, Spain

Ex-Officio (TC Chair)
Steve Goddard, University of Nebraska-Lincoln, USA

Finance Chair
Tei-Wei Kuo, Academia Sinica and National Taiwan University, Taiwan

Track Chairs
Cyber-Physical Systems: Alfons Crespo, Technical University of Valencia, Spain
Design and Verification: Dionisio de Niz, Software Engineering Institute, USA
Wireless Sensor Networks: Mário Alves, Politécnico do Porto, Portugal

Workshop Chair
Nathan Fisher, Wayne State University, USA

Work-in-Progress Chair
Rodolfo Pellizzoni, University of Waterloo, Canada

RTSS@Work Chair
Aniruddha Gokhale, Vanderbilt University, USA

Publications Chair
Shinpei Kato, Nagoya University, Japan

Publicity Chairs
Harini Ramaprasad, Southern Illinois University, USA
Insik Shin, Dept. of Computer Science, KAIST, Korea
Marko Bertogna, University of Modena, Italy

Local Arrangements Chair
Sathish Gopalakrishnan, University of British Columbia, Canada
Work-in-Progress Program Committee

Benny Akesson, Czech Technical University in Prague, Czech Republic
Gabriel Parmer, George Washington University, USA
Haohan Li, The MathWorks, Inc., USA
Heechul Yun, University of Kansas, USA
Jian-Jia Chen, Karlsruhe Institute of Technology, Germany
Marko Bertogna, University of Modena, Italy
Rodolfo Pellizzoni, University of Waterloo, Canada
Sibin Mohan, University of Illinois at Urbana-Champaign, USA

RTSS@Work Program Committee

Miroslav Pajic, University of Pennsylvania, USA
Dionisio de Niz, Software Engineering Institute, USA
Abhishek Dubey, Vanderbilt University, USA
Aniruddha Gokhale, Vanderbilt University, USA
Message from the Chairs

Dear Colleagues,

Welcome to Vancouver and the 34th IEEE Real-Time Systems Symposium (RTSS’13). These proceedings collect the accepted abstracts for the Work-in-Progress and the RTSS@Work demo sessions of RTSS’13.

The Work-in-Progress session is devoted to the presentation of new and on-going research in real-time systems, theory, and applications. This year the WiP track received 16 submissions, out of which 10 were selected to be presented during the conference. Each submission was reviewed by at least three members of the program committee. The WiP session will take place during the first day of the main conference (December 4). Authors will first give a short, five minutes overview of their work, followed by a poster session to allow for further discussion.

RTSS@Work is the open demo session organized as a part of RTSS. The goal of RTSS@Work is to provide a platform for researchers to present prototypes, tools, simulators, and systems, which extend the state-of-the-art in Real-Time Technologies and Techniques. This session will augment the traditional forum by enabling presenters to demonstrate working systems, thereby allowing them to directly engage with the audience, generate interest in new research topics, and encourage wider adoption of common frameworks. This year we received 8 submissions, out of which 7 were selected for demonstration. At least three reviews including comments for improving the demo were provided to the authors. RTSS@Work will take place during the day of workshops at the end of the day (December 3). The authors will first make a short presentation of 2 minutes followed by the demonstration of their techniques. Authors are also encouraged to set up a poster next to their demos.

The sessions would not occur without the outstanding effort of many individuals. We would first like to thank all the authors who submitted their work, as well as the members of the program committees who worked extremely hard to provide all reviews in a timely manner. We would also like to thank the members of the RTSS’13 Organizing Committee for their support and guidance. Finally, we would like to thank all of you for attending RTSS’13 and sharing your thoughts and feedback with the authors.

Rodolfo Pellizzoni  
University of Waterloo, Canada  
WiP Chair

Aniruddha Gokhale  
Vanderbilt University, USA  
RTSS@Work Chair
# Table of Contents

## Work-in-Progress Session

An Energy-Saving Approach for Real-Time Highway Traffic Estimation Using GPS-Enabled Smartphones
*Daxiao Liu and Albert M. K. Cheng* ................................................................. 1

Cache-aware static scheduling for hard real-time multicore systems based on communication affinities
*Lilia Zaourar, Mathieu Jan and Maurice Pitel* .................................................. 3

Capacitive Sensor Array-based Gesture Recognition System for Paralysis Patients
*Alexander Nelson, Ali Jafari, Ryan Robucci, Nilanjana Banerjee and Susan Fager* ......... 5

Model-Based Design: Anti-lock Brake System with Priority-Based Functional Reactive Programming
*Kaleb R. Christoffersen and Albert M. K. Cheng* ................................................. 7

Monitoring On-line Timing Information to Support Mixed-Critical Workloads...
*Angeliki Kritikakou, Olivier Baldellon, Claire Pagetti, Christine Rochange, Matthieu Roy and Fabian Vargas* ................................................................. 9

Probabilistic Application Interfaces for Hierarchical Scheduling
*Nima Moghaddami Khalilzad, Meng Liu, Moris Behnam and Thomas Nolte* .............. 11

QoS-aware AFDX: benefits of an efficient priority assignment for avionics flows
*Tasnim Hamza, Jean-Luc Scharbarg and Christian Fraboul* .................................. 13

Supporting Multi-Hop Communications with HaRTES Ethernet Switches
*Mohammad Ashjaei, Paulo Pedreiras, Moris Behnam, Luis Almeida, Thomas Nolte* ....... 15

WCET-aware Instruction Scheduling and Register Allocation for Clustered VLIW Processors
*Xuemeng Zhang, Hui Wu, Yuxing Peng, Haiyan Sun and Yijie Wang* ....................... 17

XDense: A Sensor Network for Extreme Dense Sensing
*João Loureiro, Vikram Gupta, Nuno Pereira, Eduardo Tovar and Raghuraman Rangarajan* ................................................................. 19
RTSS@Work

Reconfiguration-based Energy Optimization in Battery Systems: a Testbed Prototype ................................................................. 21
Liang He, Shaodong Ying and Yu Gu

Distributed Real-time Fault Tolerance on a Virtualized Multi-Core System ...... 22
Eric Missimer, Ye Li and Richard West

Demonstrator for Modeling and Development of Component-based Distributed Real-time Systems with Rubus-ICE ................................................................. 24
Alessio Bucaioni, Saad Mubeen, John Lundbäck, Kurt-Lennart Lundbäck, Jukka Mäki-Turja and Mikael Sjödin

DREMS: A Toolchain and Platform for the Rapid Application Development, Integration, and Deployment of Managed Distributed Real-time Embedded Systems ........................................................................................................... 26
William Emfinger, Pranav Kumar, Abhishek Dubey, William Otte, Aniruddha Gokhale and Gabor Karsai

Atacama: An Open Experimental Platform for Mixed-Criticality Networking on Top of Ethernet ......................................................................................................................... 31
Gonzalo Carvajal and Sebastian Fischmeister

Demonstrating Dynamic Ethernet Virtualization for Efficient Bandwidth Management .......................................................................................................................... 32
João Reis, Tiago Silva, Luis Almeida, Ricardo Marau and Paulo Pedreiras

XtratuM Hypervisor for Mixed-Criticality Systems: Configuration and Deployment ........................................................................ 33
M. Masmano, J. Coronel, P. Balbastre, A. Crespo, J. Simó and S. Peiró
An Energy-Saving Approach for Real-Time Highway Traffic Estimation Using GPS-Enabled Smartphones*

Daxiao Liu, Albert M. K. Cheng
Department of Computer Science
University of Houston, Texas, USA
aristoegoliu@gmail.com, cheng@cs.uh.edu

Abstract—This paper presents a microscopic traffic estimation algorithm for smartphones by employing their built-in probes (such as GPS and acceleration sensors) to increase the accuracy of estimating the real-time traffic condition without significantly increasing these smart phones’ energy consumption. In this approach, real-time traffic data is collected through cooperating smartphone users traveling on urban roads. A new reporting algorithm is provided on smartphone users’ side to minimize smartphone probes’ connection with the server. Based on these probes’ data, real-time traffic condition is estimated on the server side by applying the Kalman Filtering algorithm. Also the real-time traffic level of service (LOS) is estimated on urban roads. Simulation results show that the proposed algorithm requires less energy usage than existing methods without sacrificing the accuracy of the real-time traffic estimation.

Keywords—Real-Time Traffic Estimation; Energy Saving; Kalman-Filtering; GPS

I. INTRODUCTION

Real-time traffic information is essential in daily lives. Among numerous methods for collecting real-time traffic information, using smartphone probes is on the spotlight in the recent five years. Compared with traditional methods which employ fixed-location sensing infrastructures such as inductive loop detectors, radars or video cameras, using GPS-enabled smartphones has lower cost and higher scalability. Compared with other new technologies which use floating traffic probes such as LiDAR, the deployment of GPS-enabled smartphones has higher feasibility on the market and lower requirement on equipment installation and maintenance. Realizing this, traffic engineers have started to explore this area. Ganti et al [11] provided a fuel-efficient map application based on real-time traffic information but do not focus on the smartphones’ energy usage. Lovell [4] studied the accuracy of speed measurement from cell phones, Herrera [6] went a step further and found that a 2%-3% penetration of cell phones in the driver population is sufficient to provide accurate measurements of the velocity of the traffic flow. Kalman Filtering (KF) has widely been adopted as a traffic estimation model [7]. Meanwhile the UC-Berkeley wireless group released the Mobile Millennium [3] to monitor the Northern California area based on smartphone probes. Waze [5], a successful commercial mobile application, provided turn-by-turn navigation and also collected real-time traffic information.

However, when running Mobile Millennium and Waze on smartphones, a major concern is their increased energy consumption. Smartphones have limited battery and limited 4G network coverage. From the users’ perspectives, energy consumption and 4G network usage are important factors determining the practicality of the traffic estimation. Based on Kalman Filtering, this article proposes a new energy-saving algorithm to address these issues. To accurately measure the power consumption of smartphones, Zhang et al [8] described a Regression-Based Finite Power State approach. They found that given fixed channel and packet rates, the packet size did not influence power consumption. Pathak et al [9] also adopted a Finite Power State Model, but emphasized more from a system’s perspective. Cui et al [10] presented a survey on current energy efficient modeling in mobile cloud computing. They summarized that “the main idea of reducing transmission energy is to keep the WNIC in low power state (sleep state for Wi-Fi or idle state for cellular) as long as possible” and “Burst is better than scattered data flow”. Our proposed algorithm meets their recommendations by sending more burst data than scattered data. This research is part of our larger project to balance a city’s traffic among its roads and reduce travel time for each vehicle in a city-wide traffic cyber-physical system [12].

II. METHODOLOGY

A simulation-based framework emulates the smartphone-based real-time urban traffic estimation. The framework consists of three parts: traffic generator, traffic estimation evaluation, and energy consumption estimation. The traffic generator simulates traffic on the given urban road network. It provides three different traffic congestion states each of which has its pre-defined normal distribution of vehicle speed. A randomly selected 5% (penetration rate) of the “generated” vehicles are equipped with smartphones recording their locations and speeds. Each smartphone-equipped vehicle uses the following algorithm to determine the frequency of reporting its recorded information to the server. Concurrently at the server side, the traffic condition is estimated using Kalman Filtering. Average link speeds are calculated and classified while the corresponding traffic level of service (or road condition) is estimated. Finally, the user’s energy consumption is estimated at the client side. Two essential factors of the simulation is the smartphones’ penetration rate and their sampling/reporting intervals [2]. Here we apply a 5% probe penetration rate and the same sampling time interval (20 seconds) but a different reporting time interval which is calculated based on the following

*Supported in part by the United States National Science Foundation (NSF) under Awards No. 0720856 and No. 1219082.
employed to evaluate the algorithm’s performance. morning rush hour, noon rush hour, and non-rush hours are exit and highway US59. Different traffic scenarios such as Houston urban road as a case study. The road is divided into and 54 Mbps. Here, we assumed the same channel rate. R refers to uplink channel rate, typically varied as 11, 36, 48 this reporting is voluntary. Here we use a linear regression information in the smartphones is reported to the server and this reporting time interval to the server. No personal location of the vehicle, the current street’s speed limit, and the GPS, the reverse-geocoding process determines the III. RESULTS

As shown in Table I, we chose a total of 10.3 miles of Houston urban road as a case study. The road is divided into 7 links which include local street, highway entry, highway exit and highway US59. Different traffic scenarios such as morning rush hour, noon rush hour, and non-rush hours are employed to evaluate the algorithm’s performance.

Table I. Actual Link Speed vs Estimated Link Speed

<table>
<thead>
<tr>
<th>Link</th>
<th>Description</th>
<th>Actual Speed (m/s)</th>
<th>Estimated Speed (m/s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>#108</td>
<td>Renwick St.</td>
<td>33.41</td>
<td>33.41</td>
</tr>
<tr>
<td>#210</td>
<td>US59 East</td>
<td>35.84</td>
<td>35.84</td>
</tr>
<tr>
<td>#511</td>
<td>US59</td>
<td>36.54</td>
<td>36.54</td>
</tr>
<tr>
<td>#613</td>
<td>US59</td>
<td>37.24</td>
<td>37.24</td>
</tr>
<tr>
<td>#614</td>
<td>US59</td>
<td>37.64</td>
<td>37.64</td>
</tr>
<tr>
<td>#710</td>
<td>US59 Ext.</td>
<td>39.56</td>
<td>39.56</td>
</tr>
</tbody>
</table>

To achieve traffic equilibrium in different traffic conditions, different numbers of vehicles are generated with random trips. For instance, 790 vehicles are generated in the morning rush. After estimating the average link speed at the server, the traffic level of service is estimated with the actual average link speeds in a one-hour frame. In this simulation, each smartphone updates the estimated level of traffic service every five minutes. This part of the energy consumption is not included in our simulation results. The simulation results show our approach has good performance: only 3 of the 21 estimations are inaccurate and it needs only 60% of the energy consumed by traditional algorithms during non-rush hours and about 85% during rush hours.

IV. FUTURE WORK

Ongoing work applies this algorithm with statistical learning leading to another prevailing traffic estimation algorithm other than the one with Kalman Filtering. Also, experiments with real vehicles and smartphones are being planned to validate our simulation result.

REFERENCES


Cache-aware static scheduling for hard real-time multicore systems based on communication affinities

Lilia Zaourar and Mathieu Jan
CEA, LIST
Embedded Real Time Systems Laboratory
F-91191 Gif-sur-Yvette, France
Email: Firstname.Lastname@cea.fr

Maurice Pitel
Schneider Electric Industries
37, quai Paul Louis Merlin
F-38050 Grenoble, France
Email: Maurice.Pitel@schneider-electric.com

Abstract—The growing need for continuous processing capabilities has led to the development of multicore systems with a complex cache hierarchy. Such multicore systems are generally designed for improving the performance in average case, while hard real-time systems must consider worst-case scenarios. An open challenge is therefore to efficiently schedule hard real-time tasks on a multicore architecture. In this work, we propose a mathematical formulation for computing a static scheduling that minimizes $L_1$ data cache misses between hard real-time tasks on a multicore architecture using communication affinities.

I. INTRODUCTION

Multicore processors have become the norm in many execution platforms in various fields. Such architectures come with a cache memory hierarchy made of several levels, shared or not between cores. Figure 1 represents the typical cache hierarchy that can be found in a multicore architecture with two levels of cache, noted $L_{1x}$ and $L_2$ where $x$ is the number of the core (in the figure $x$ ranges from 1 to 4). In such architectures, the $L_2$ cache is larger but provides slower access time than the $L_1$ cache. Generally, the $L_2$ cache is unified and shared among all cores, while at the $L_1$ level data and instruction caches are separated and private to each core. However, in this work we focus only on the data caches.

Using such multicore architectures for developing hard real-time systems is an important research area. Multicore are generally tuned for optimizing performance for the average case, while hard real-time systems must consider worst-case scenarios due to certification constraints. A major problem lies in the management of cache for mastering the impact of conflicts on the Worst-Case Execution Time (WCET) of each task. Designing cache-aware scheduling policies is becoming a popular research area. In this work, we show how *a static scheduling that minimizes $L_1$ data cache misses between hard real-time tasks on a multicore architecture can be computed.*

![Classical cache memory hierarchy of a multicore architecture.](image)

II. RELATED WORK

[2] focuses on the memory-to-$L_2$ traffic in the cache hierarchy of soft real-time systems. They propose a two steps method to discourage the co-scheduling of the tasks generating such traffic. First, the tasks that may induce significant memory-to-$L_2$ traffic are gathered into groups. Then at runtime, they use a scheduling policy that reduces concurrency within groups. [1] also proposes several global multi-core scheduling strategies for soft real-time systems to minimize the $L_2$ cache trashing. Co-scheduling of the tasks of a same group is used to optimize the efficient use of the $L_2$ shared cache. Task promotion is another example of a studied scheduling policy.

When considering hard real-time systems, to the best of our knowledge we are only aware of [4]. Cache-partitioning techniques are used to avoid interferences, at the $L_2$ cache level, between the tasks that are running simultaneously. In addition to regular temporal constraints used within a schedulability test, cache constraints due to cache-partitioning are added and steer the computation of the scheduling. They propose a linear programming formulation to solve this problem and an approximation of this formulation for larger task sets.

The closest work to ours is [5]. While proposed cache-aware scheduling strategies are evaluated using a soft-real time kernel, the results can also be used for hard real-time systems. They propose a bin packing approach to evenly distribute the Working Set Size (WSS) of the tasks on all cores in order to reduce conflicts. The resolution algorithm is based on the next fit decreasing heuristic applied on the tasks ordered by their decreasing WSS. Besides, they rely on a notion of distance between caches of non-uniform memory architectures to further optimize the solution. This is only used for the tasks that share some common memory area and are gathered into groups. However, it is unclear how the common memory area defines a group as well as how groups are reduced when the heuristic fails to allocate a group.

To summarize, most of the existing cache-aware scheduling proposal have focused on the efficient use of the $L_2$ cache.

III. TASK MODEL AND NOTATIONS

Let $\Gamma = \{\tau_1, \tau_2, ..., \tau_n\}$ be a set of $n$ independent, synchronous, preemptible and periodic tasks. $\Gamma$ is handled using the implicit deadline periodic task model. Each task $\tau_i \in \Gamma$ has the following temporal parameters $\tau_i = (P_i, C_i)$. $P_i$ is the
period of the task and \(C_i\) is the WCET. A job \(i\) represents an instance of a task with \(C_i\) its WCET. Let \(H\) be the hyper-period of task set. It equals to the least common multiple of all periods of tasks in \(\Gamma\).

As in [6], the hyper-period \(H\) is divided in intervals, an interval being delimited by two task releases. A job can be present on several intervals, and we note \(w_{j,k}\) the weight of job \(j\) on interval \(k\). We denote by \(I\) the set of intervals and \(|I_k|\) the duration of the \(K\)th interval, \(|I_k| = t_{k+1} - t_k\). The weight of each job is the amount of processor necessary to execute job \(i\) on interval \(|I_k|\) only (it is not an execution time but a fraction of it). \(J_i\) is the job set of all jobs of \(\Gamma\) scheduled during the hyper-period \(H\).

Then, temporal schedulability constraints are expressed using a linear program described in [7] to compute the optimal job weights on each interval for all tasks \(T_i \in \Gamma\). First, the sum of all job weights on an interval does not exceed the processor maximum capacity:

\[
\sum_{i \in J_k} w_{i,k} \leq M, \forall k
\]  

Then each job weight does not exceed each processor maximum capacity:

\[
0 \leq w_{i,k} \leq 1, \forall k, \forall i.
\]

Finally, jobs must be completely executed:

\[
\sum_{k \in E_i} w_{i,k} \times |I_k| = C_i, \forall i.
\]

### IV. Problem Formulation

The problem we address in this work is to reduce \(L_1\) data cache misses when scheduling hard real-time tasks on a multicore architecture. To this end, we maximize the co-scheduling on a same core of tasks that exchange data while still ensuring temporal schedulability constraints. We assume a static knowledge of data exchange between the tasks of an application. Therefore, we extend the classical periodic task model with the WSS parameter for each task and model this problem using a variant of the knapsack problem. We also assume that the system is schedulable and hence we only seek to optimize the allocation of the tasks on the \(L_1\) caches. In addition, we suppose that the size of a \(L_1\) cache enables to host several tasks simultaneously, a valid hypothesis in the case studies we consider. We leave as future work the management of cache conflicts, using techniques such as in [9].

The multicore platform is made of \(m\) cores and we note \(C_{L_1}\) the capacity of each data cache \(L_1\) (we assume the \(L_1\) caches to have an equal size). Finally, from the data of the application we can calculate \(WSS_i\) which is the WSS of job \(J_i\). This is computed by adding the size of each data section from the binary of an application. We note \(a_{ij}\) the affinity between the job \(i\) and the job \(j\). The affinity between two tasks is defined as the number of communication flows between them. Higher is the number of communication flow, higher is the affinity between two tasks. Communication flows between tasks are extracted using the software architecture of the considered hard real-time application.

Then, integrating into the previously described schedulability constraints require to introduce a decision variable representing the allocation of the tasks on cores for each intervals. Let \(x_{ijk}\) be this decision variable that is equal to 1 if the job \(i\) is assigned to cache \(j\) during time interval \(I_k, k \in \{1, \ldots, T\}\) and 0 otherwise. The sum of the WSS of the jobs allocated to a cache should not exceed its capacity:

\[
\sum_{i=1}^{n} WSS_i \times x_{ijk} \leq C_{L_1}, \forall j, \forall k.
\]

In addition, each job must be assigned to a single cache:

\[
\sum_{j=1}^{m} x_{ijk} \leq 1, \forall i, \forall k
\]

Besides, to link the temporal schedulability constraints with the aforementioned cache constraints, the following relationship can be define: if \(\sum_{i=1}^{n} x_{ijk} = 0\) then \(w_{i,k} = 0\) and if \(\sum_{i=1}^{n} x_{ijk} = 1\) then \(w_{i,k} > 0\). Finally, since our aim is to maximise affinity \(L_1\), we obtain the following objective function:

\[
Max(Z) = \sum_{i=1}^{n} \sum_{j=1}^{m} \sum_{k=1}^{T} a_{ij} \times x_{ijk} \times x_{ijk}^{'} \forall i, \forall j, \forall k.
\]

### V. Conclusion and Future Work

In this work, we show how we can extend classical (temporal) schedulability constraints to minimize \(L_1\) data cache miss between communicating hard real-time tasks on a multicore architecture. In future work, we plan to generalise the formulation to address other level of a cache memory hierarchy. As our formulation of the problem uses a quadratic knapsack, known to be \(NP\)-hard [8], another next step is therefore the linearization of the objective function. Then, we plan to implement it using the CPLEX solver to generate the static scheduling of hard real-time tasks. Finally, we plan to test our method on several hard real-time industrial applications.

### References


Capacitive Sensor Array-based Gesture Recognition System for Paralysis Patients

*University of Maryland, Baltimore County, **Madonna Rehabilitation Hospital, Lincoln, Nebraska
{alnel1, a137, robucci, nilanb}@umbc.edu

Abstract—This work presents the proposed design and implementation of a capacitive sensor array-based platform for gesture recognition in paralysis patients. These sensors are designed to be embedded into clothing, pillows and bed sheets of the patients and are composed of conductive fabric based electrodes for contactless detection of motion. This is combined with duty cycled inertial sensors in a feedback loop for greater accuracy and system flexibility. To assess the system, we have performed a set of preliminary experiments using different gestures and motions. The experimental results verify the capability and effectiveness of the proposed system for real time motion detection.

Keywords—capacitive sensing, biomedical sensor, biofeedback systems, microharvesting

I. INTRODUCTION

Paralysis, paresis, weakness and limited range of motion are common sequels resulting from severe injuries such as stroke, brain injury, spinal cord injury and many chronic (guillian barre syndrome) and degenerative (amyotrophic lateral sclerosis) conditions. While the exact movement capabilities of individuals vary with diagnosis and severity of injury or stage of illness, common movement capabilities following severe injury can include head movement, shoulder movement, eye movement, facial musculature movement, and limited arm and hand movements. Depending on the exact etiology, some participants may either be recovering movement capabilities over time (i.e., stroke recovery) or may be losing physical abilities over time (i.e., degenerative conditions such as amyotrophic lateral sclerosis). Individuals with limited movement capabilities often rely on assistive technology to communicate, control their environment and to use computers and mobile devices. Rehabilitation practitioners find movements that are reliable options (switches, head tracking, eye tracking) to that physical movement. While a wide range of diverse and often mutually incompatible assistive technology access options exist, challenges remain finding reliable options for those with severe impairments and options that require minimal set-up and maintenance, and cause minimal fatigue and intrusiveness [1].

To address the above challenges, we are pursuing an exploratory project to develop an ultra low-cost, low maintenance, unobtrusive, micro-harvester powered, capacitive sensor array [2], [3] driven system that would require minimal setup, would be usable by patients with limited motor skills to assist in environmental control and would facilitate movement recovery throughout the rehabilitation process. The proposed system uses a combination of three sensing modalities including capacitive sensing arrays on bedsheets, pillows, and wheelchair pads; inertial sensors on finger tips, caps, or head gear; and capacitive patches sewn into sleeves, wristbands, band-aids on the forehead to capture three types of gestures common in patients suffering from severe paralysis: (1) Finger twitching and lower arm rotations; (2) shoulder shrugs; (3) head movements.

The on body sensing system will leverage a micro-harvester scavenging indoor light while the bed and wheelchair sensors would utilize standalone power sources. A key innovation in the design of the system is the hierarchy of sensing modalities for accurate, ultra-low power sensing of the above gestures when the patient is seated on his bed or wheelchair. The data from the body sensors are transmitted to the collection and signal conditioning module on the bed using a novel thread-based inductive coupled communication technique. The use of sensors embedded into items of daily use make the system invisible to the patient and would work seamlessly with minimal clutter and maintenance cost. Our proposed system presents the following novel research contributions:

**Micro-harvester Powered Hierarchical Gesture Sensing:** As depicted by figure 1, the proposed system will utilize capacitive sensor arrays (CSA) on pillows and bedsheets while CSA on clothes and on body will be coupled with inertial sensors to explore tradeoffs in gesture recognition accuracy and energy consumption. The on body sensors will be powered by scavenging indoor light to minimize maintenance needed in replacing batteries. The more power-intensive inertial sensors are more accurate in capturing motion, and will be used sparingly to supplement, train, and correct for errors in the
capacitive patches. For this project, we are exploring signal processing algorithms that leverage our sensor hierarchy to minimize energy consumption while maintaining accurate gesture recognition.

**Feedback for Patient Normalized Sensing:** The proposed system will normalize for patient positioning and context. For example, the type of gestures and speed of gestures are a function of context such as the time of day and whether the patient has had drug administration. The developed system will model the correlation between context with the type of gesture using a rule association technique. The system then would use a maximum likelihood gesture inference process to determine the gesture that would follow a context, to adapt sensing based on patient usage context.

**II. APPLICATIONS**

This work would lead to developing environmental control of appliances for patients recovering from paralysis. The environmental control application could be used to control appliances like televisions, thermostats, and lights using patient gestures. We have already designed a z-wave-enabled home automation system that exposes a webservice for remote device control in the home. The automation system could be combined with the presented work to remotely control appliances. The novel computational components of our proposed technology, however, is not the home automation system, but in the sensing and signal processing schemes.

**III. PRELIMINARY EXPERIMENTS**

Our primary sensing modality are capacitive sensor arrays (CSA). While capacitive sensing has been used for applications such as multi-touch screens, the proposed approach will use an array of capacitive plates built out of conductive threads that can be sewn into items of daily use. Embedded into items of daily use, the sensors are invisible from the patients. When built out of conductive threads, these sensor arrays have distinct advantages in the domain of motion capture in paralysis patients. First, these sensors are passive, and combined with our custom designed capacitive sensing circuit consumes less than 100 microwatts when active. If placed on the human body they can be powered using energy harvested from indoor light. Second, the plates work on the principle of perturbation in electrical field lines caused by movement.

Therefore, movement on the body or in the vicinity can be captured even if the sensors are not placed at the location where the motion occurred. For instance, finger twitches can be captured using CSAs placed on the wrist, while shoulder movement can also be captured using sensor arrays placed on a shirt sleeve. Head motion, similarly, can be captured using capacitive patches in head gear. Third, a set of capacitive arrays can work in concert to determine the position and velocity of motion, which consequently can be used to determine a wide range of gestures.

To illustrate the second and third property, we present results from two experiments using capacitive plates. In the first experiment, we place capacitive plates on the wrist of a subject, and use a capacitor sensing analog circuit to measure capacitance changes as the subject twitches his finger. In the second experiment, we place three capacitor plates on a pillow and move our hand slowly over the plates. The results are shown in Figure 2 (a) and (b).

Figure 2 (a) shows that as the finger is moved, the capacitance changes, illustrating that the plate can capture this motion reliably in spite of not being placed on the finger. Figure 2 (b) shows that the capacitance changes as the hand is bought close to the plate, and the capacitance peaks at times when the hand was vertically above the plate. This initial data shows the feasibility of using capacitive plates to determine where the motion occurred and the speed of motion with the CSAs not placed at the point of movement. Figure 2 (c) shows the distinct change in capacitance when the distance between the hand and the plates are changed, illustrating the sensitivity of the capacitive sensor array. The capacitive plates cost close to 1-5 cents, and hence, they are an ultra-inexpensive mode for capturing limited motion.

**REFERENCES**


Model-Based Design: Anti-lock Brake System with Priority-Based Functional Reactive Programming*

Kaleb R. Christoffersen¹, Albert M. K. Cheng²

¹Department of Mechanical Engineering, ²Department of Computer Science
University of Houston, TX 77004, USA
¹kchristoffersen@uh.edu, ²cheng@cs.uh.edu

Keywords—anti-lock brake system, priority-based functional reactive programming, automotive control systems, embedded controller, abort-wait-restart scheduling algorithm, cyber-physical systems, model-based design

ABSTRACT

In support of ongoing research in real-time scheduling and response time analysis of cyber-physical systems (CPS), this paper describes an inter-disciplinary collaboration which seeks to use model-based design in establishment of the physical model and controller design for an automobile’s anti-lock brake system (ABS). A simplified model of vehicle dynamics is established to describe the overall system and its response. Discussion is given to control variables and system behaviors. Treatment is given to utilization of an Abort-Wait-Restart (AWR) scheduling algorithm under a priority-based functional reactive programming (P-FRP) paradigm.

A. Problem

The automotive arena is one example in which mass exploitation of networked electronics is fundamentally changing what was once a relatively simple mechanical system. Real-time processors and microcontrollers comprise the electronic control units (ECUs) of today’s modern vehicles and are responsible for performing the necessary logic on signals obtained from feedback sensors to affect the desired response. Since automobiles lack dedicated CPUs, these embedded units make up distributed networks of intelligent devices requiring deterministic control as well as safe and reliable operation. A modern vehicle contains upwards of 50 to 100 ECUs and as cars become more complex, the already voluminous code required to manage such a network—currently 100M lines compared to 6.5M for Boeing’s 787 Dreamliner avionics [1]—will soon be exacerbated by challenges of security and maintainability [2].

These problems hint at a need for new approaches in managing the competition of hardware resources [3] and requires a paradigm shift in software development. Many ECUs are developed with procedural, C-based code in which the developer provides step-by-step instructions. This linear nature of carrying out a controller’s functions is becoming antiquated in a “network” society and cannot sustain continued ballooning in lines of code as vehicles become multifaceted intelligent transportation systems (ITS). Software of the 21st century will require higher levels of abstraction and P-FRP is the basis for which the ongoing work presented here seeks to advance. Functional programming is a declarative language focusing on a desired result without explicit procedures on how to achieve it. Work in [4], [5], and [6] established a modified functional paradigm for response time analysis on single-core and multi-core architectures as well as specialized timing algorithms implementing an Abort-Wait- Restart (AWR) scheduler using a priority scheme for a controller’s tasks and messages. The AWR scheme was proven to be deterministic in [5] by reducing the impact of wrong decision making as a result of implementing executions as function evaluations grounded in mathematics. The benefit of this approach is that no function evaluates incomplete thereby mitigating code errors. This purely deterministic behavior could prove instrumental for such real-time, safety-critical systems as ABS if it can be shown that P-FRP with AWR exhibits performance characteristics on par with those of conventional ECUs implementing an imperative language. In fact, the inherent determinism and elimination of code “side effects” should demonstrate the long-term viability of this approach to future research and development of cyber-physical systems.

B. System Model Approach

The system under investigation is an anti-lock brake system consisting of wheel speed sensors, hydraulic actuators, and an electronic brake control module (EBCM). The EBCM’s primary function is to minimize wheel slippage in order to prevent wheel lockup during emergency braking while maximizing lateral stability and steerability.

\[
\begin{align*}
\text{FBD: } & \sum F_i : m \ddot{v} = -F_b \\
\sum M : & \mathbf{J} \dot{\omega} = rF_b - T_b \\
\lambda = & \frac{v - r \omega}{v} 
\end{align*}
\]

\(m\) : quarter-car mass; \(J\) : wheel moment of inertia
\(v\) : longitudinal acceleration; \(\dot{\omega}\) : angular acceleration
\(F_b\) : effective braking force; \(T_b\) : braking torque
\(r\) : wheel radius

Figure 1: System Dynamics

The controlled plant will be the standard quarter-car model in which system dynamics are equated using a single wheel. Wheel speed (angular) is measured and compared against vehicle longitudinal speed to calculate the slip ratio, \(\lambda\). These values are computed and processed by the EBCM logic which commands tasks to the hydraulic actuator to modulate brake pressure up to 15 times per second [7]. The slip ratio is determined by tire-road adhesion properties largely dependent on non-linear tire dynamics [8]. Optimal range of slip for various surfaces is 10-30% [7] with full wheel lockup occurring at 100%.

* Supported in part by the National Science Foundation under Awards No. 0720856 and No. 1219082.
Most ABS assemblies are actuated by a pair of 2-position solenoid valves: a pressure holding (primary) valve and a pressure reducing (secondary) valve. The modulator actuates each solenoid pair based on wheel behavior, i.e. slip, to dump, hold, or increase brake pressure. These events are triggered by priority-based tasks and Table 1 depicts a typical braking sequence. The armature winding of each solenoid pair allow for the duty cycle seen in Table 1 in which 0% indicates off, 50% duty induces the pressure hold solenoid, and 100% duty activates both solenoids. The pump ensures adequate pressure through various stages of the circuit.

Table 1: ABS Actuator Events & Sequencing

<table>
<thead>
<tr>
<th>Actuator Event</th>
<th>ABS</th>
<th>Slip (%)</th>
<th>Primary Solenoid</th>
<th>Secondary Solenoid</th>
<th>Armature Current (%)</th>
<th>Pump Motor</th>
</tr>
</thead>
<tbody>
<tr>
<td>Normal</td>
<td>Inactive</td>
<td>&lt; 10%</td>
<td>Off (Open)</td>
<td>Off (Closed)</td>
<td>0</td>
<td>Off</td>
</tr>
<tr>
<td>Hold</td>
<td>Active</td>
<td>10-30%</td>
<td>Off (Closed)</td>
<td>Off (Closed)</td>
<td>50</td>
<td>On</td>
</tr>
<tr>
<td>Dump</td>
<td>Active</td>
<td>&gt; 30%</td>
<td>On (Closed)</td>
<td>On (Open)</td>
<td>100</td>
<td>On</td>
</tr>
<tr>
<td>Increase</td>
<td>Active</td>
<td>&lt; 10%</td>
<td>Off (Open)</td>
<td>Off (Closed)</td>
<td>0</td>
<td>On</td>
</tr>
</tbody>
</table>

C. Motivation

Written as mathematical functions with recurrence relations, P-FRP compilers can generate resource-bounded C code that represent atomic event handlers, each responsible for a single event source, capable of interrupting the execution of other events based on priority assignment [4]. Since P-FRP involves the mapping of priorities to system behaviors [9], e.g., slip values, the 3-state events of the solenoids could benefit from the AWR model, which always incorporates an immediate switch to the highest-priority event [4]. This research seeks to analyze whether the EBCM can benefit from this algorithm since behaviors of the system can be evaluated as dynamical equations making them amenable to execution by P-FRP programs concerned with updating tasks [9]. Despite the cost of AWR shown in [4] to increase processor overhead as well as nominal reduction in priority inversion, the major advantage of this scheme is the deterministic resulting from the program always being in a stable state.

Since the dump-hold-increase events of the solenoid must occur atomically, it stands to reason that ABS response will be unaffected by such overhead; no event recurrence depends on a previous execution of the same event. In a hard braking maneuver, the objective is to maintain slip at an optimal value of \( \lambda = 0.2 \pm 0.1 \); controlling the brake torque applied by the discs or drums in a manner that achieves this is the primary concern. By applying the optimal braking torque, which achieves optimal slip, the coefficient of friction at the contact surface reaches a peak thereby maximizing the braking force: a function of normal load and friction coefficient (Figure 2).

Implicitly, “dumping” of brake pressure is the highest priority of the actuator since values beyond \( \lambda_{\text{max}} = 0.3 \) can quickly lead to lockup. The “increase” event would represent the 2nd highest priority since a value of \( \lambda_{\text{min}} = 0.1 \) is another critical behavior of the system. The “hold” task would be the lowest priority associated to any value in the feasible range since its occurrence has greater range allowing the EBCM to evaluate system response before the next event carried out.

The recurrence of these time-varying events are difficult to analyze and predict; however, with a scheme centered on priority-based function execution the desired response could be guaranteed using AWR since the state of any event aborted by another has no dependency on its previous state. We can define the event priorities from high to low as dump-increase-hold (Table 2). With “hold” mode representing an intermediary event that is the baseline state of the system, “increase” & “dump” events are higher priority tasks that will interrupt execution of “hold” and restart upon completion.

Table 2: Priority Assignment of ABS Actuator Events

<table>
<thead>
<tr>
<th>Task Priority</th>
<th>DUMP</th>
<th>INCREASE</th>
<th>HOLD</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td></td>
</tr>
</tbody>
</table>

D. Conclusion

Currently, a working model of wheels and vehicle dynamics are being developed with control design focused on hydraulic valve behavior of the ABS actuator. The priority assignment of events required for desired system response and behavior will delve into development of a P-FRP program implementing AWR for hardware-in-the-loop (HIL) testing of the controller. Similar tests will be conducted utilizing C-code and response analysis will be compared for evaluation of P-FRPs efficacy in real-time reactive systems. Ongoing work will involve several iterations of the process with increasing model complexity and validation in an attempt to push the limits of the P-FRP & AWR model on cyber-physical systems.

References

Monitoring On-line Timing Information to Support Mixed-Critical Workloads

A. Kritikakou*, O. Baldellon†, C. Pagetti*, C. Rochange†, M. Roy†, F. Vargas§
*ONERA, †LAAS/CNRS, ‡Université de Toulouse, France
§PUCRS, Brazil

I. INTRODUCTION AND MOTIVATION

Multi-/many-core architectures provide a drastic increase in computation power, enabling the simultaneous execution of several tasks on the system. Yet, in critical embedded systems, e.g. aeronautical systems, the uncertainty of the non-uniform and concurrent memory accesses prohibits the full utilization of the system potentials. To ensure safety, such systems impose static Worst Case Execution Time (WCET) estimations; memory access times are upper bounded considering a fully congested memory bus, leading to safe but almost unusable bounds and to overwhelmingly conservative schedules [1].

Existing techniques for multicore systems address the optimization of the scheduling objectives, e.g. processor utilization, when the task set is schedulable at least in the highest criticality level. In the mixed-criticality theoretical model of [2], multiple WCET values, related to different criticality levels, are considered for each task. The lower the level, the more unsafe is the WCET bound. In the highest level, the value is trustful. In [3], [4], [5], all tasks are started at the low criticality level. When a task has not finished on time, its level is increased and the less critical tasks are dropped. When the WCET of the highest level is larger than the deadline, the problem is considered as unschedulable.

Let us now consider a critical task with a WCET below the deadline, when it is the only task executed on the system, i.e. in isolation. We advocate that in this case, the safety of the critical task can be ensured, while some less critical tasks are run in parallel on other cores. Our methodology is capable of scheduling the critical task in isolation scenario. The proposed approach is schematically depicted in Fig. 1(a). However, if $T_C$ is executed in isolation, it is schedulable, as shown in Fig. 1(b). In contrast with existing approaches, our methodology is capable of scheduling the $T_C$ by considering two execution scenarios. Initially, both tasks are executed on the system. Monitoring points are used to on-line monitor the real execution time of $T_C$ and decide switching to isolation execution of $T_C$, as shown in Fig. 1(c).

In this work, we propose an approach to improve core utilization by running several tasks in parallel and guarantee the critical task safety.

Fig. 1. Scheduling based on WCET when are considered for execution (a) both tasks, (b) only the critical task and (c) proposed hybrid approach.

II. TARGET SYSTEM

Our application domain is described by a task set with a single critical task $T_C$, which is schedulable in isolation, but non-schedulable when executed with other less critical tasks. The WCET of the critical task is computable, i.e. the whole code can be unfolded into a bounded tree. We represent binary code of the application under study as a Control Flow Graph (CFG) [6]. Each node represents a basic block and edges connect nodes that can be executed in sequence. Our target platform is a time-predictable bus-based multicore system. More precisely, we consider several processing IP-cores on an FPGA platform with hardware monitoring capabilities [7]. We assume a number of cores equal to the number of tasks, which can then be statically co-scheduled.

III. PROPOSED METHODOLOGY

Our methodology is two-fold: i) Off-line, we analyze the CFG and safely estimate the remaining WCET at several points of the $T_C$, under two scenarios: isolation, where the critical task is executed alone, and maximum load, where the maximum interference from co-running less critical tasks exists. ii) On-line, we use a hardware monitor to observe the real execution time of $T_C$ and to check whether a risk exists that the critical task misses its deadline due to system overload. If so, the less critical tasks are stopped and the execution is continued in isolation scenario. The proposed approach is schematically depicted in Fig. 2.

A. Off-line critical task analysis

As we consider critical tasks, our approach uses safe static WCET analysis [8]. From the annotated CFG, an Integer Linear Programming (ILP) formulation is written to express
the program execution time as the sum of the individual times of basic blocks (nodes) weighted by their execution counts.
This expression is maximized to find the required WCETs, with a number of constraints that reflect flow facts, e.g. loop bounds and unfeasible paths. This approach is adapted to our requirements: by modifying the ILP formulation appropriately, we are able to compute partial WCETs, i.e. WCETs between two monitoring points in the code, and the remaining WCET from one monitoring point until the end of the program. The monitoring points are set at the first instruction of each basic block of CFG, at this project stage. Ideally, for each monitoring point a, the remaining WCET until the end of the program should be estimated. However, as a point may be visited several times, e.g. when it belongs to a loop, the number of point instances to be stored is equal to the number of loop iterations. This highly increases storage requirements. Hence, we need a hybrid approach: the initial remaining WCET until the end of the program (RWCETiso) of the first point instance is computed off-line; then at runtime, the RWCETiso is updated each time the point is encountered, as described in Section III-B2. For this purpose, the information required for point a is: (i) the nested level in the CFG, levela, used to identify the reference remaining time; (ii) if a is a loop header, the WCET of the loop body (wa); (iii) the WCET from the inner loop header to point b (da). Figure 3 depicts such parameters. The timing values (ii) and (iii) are always estimated for the isolation scenario to guarantee the TC safety.

B. On-line execution

1) Hardware monitoring scheme: Our approach relies on hardware monitoring, as proposed in [9]. The scheme observes the instruction addresses from the memory. We assume that cores do not have instruction caches at this project stage. An address that has been monitored indexes a local associative memory (CAM), where levela, RWCETiso, wa and da are stored. The CAM returns the timing information of the monitoring address to be used by the on-line control mechanism. Whenever the on-line control mechanism determines that the non-critical tasks should be suspended, the hardware monitor interrupts the operating system.

2) On-line control mechanism: Our on-line low-overhead control mechanism implemented in hardware guarantees the safety of the TC execution. Initially, both the critical and less critical tasks are executed on the multicore system. At each monitoring point a, the on-line control mechanism decides whether switching from the maximum load scenario to the isolation scenario is required. This decision is taken based on the information retrieved from the CAM memory in each monitoring point a. The RWCETiso is on-line computed and updated in CAM. When the CFG is traversed in the forward direction, the RWCETiso(a) is computed as RWCETiso(levela) − da. When the CFG is traversed backward to a loop header b, the RWCETiso(b) is computed as RWCETiso(b) − wa and the new value is updated to the CAM memory. Scenario switching is triggered at monitoring point a when the condition of Equation 1 holds: ET(a) denotes the monitored real execution time of TC until point a, WCETnon is the maximum WCET between two successive monitoring points and tOver is the cost of our recovery mechanism. The latter includes the time to monitor ET(a), the time for the hardware logic to access the CAM memory and to decide, and the time for the operating system to suspend the non critical tasks. Intuitively, Equation 1 means that the critical task might miss its deadline in the current (maximum load) scenario, but by switching to the isolation scenario in point a, the deadline can still be met.

\[
ET(a) + RWCETiso(a) + WCETnon + t_{Over} > D
\]  

Our future work is the implementation of the proposed offline analysis tool and the online hardware control in the FPGA platform. We will also extend our approach to systems with several critical tasks and larger (scheduled) task sets.

References


Probabilistic Application Interfaces for Hierarchical Scheduling

Nima Moghaddami Khalilzad, Meng Liu, Moris Behnam, Thomas Nolte
MRTC/Mälardalen University, Sweden
nima.m.khalilzad@mdh.se

Abstract—The concept of hierarchical scheduling is widely used for scheduling complex real-time systems that are composed of a number of components. In the conventional hierarchical scheduling framework, targeting hard real-time systems, the size of processor capacities assigned to components is derived based on the worst case execution times of tasks. In this paper, we present our ongoing work on bringing the notion of probabilistic execution times in the context of hierarchical scheduling and deriving probabilistic component processor requirements. When dealing with soft real-time systems, this approach can eliminate the unaffordable pessimism that exists in worst-case timing analyses.

I. INTRODUCTION

Real-time embedded systems are increasingly growing in complexity. Previously separated independent systems are integrated to form new and more complex systems. To deal with such systems, compositional hierarchical scheduling techniques provide a modular approach in which the timing properties of federated real-time systems are inferred from the timing characteristics of their components [1], [2], [3]. The conventional approach in analyzing the timing properties of hierarchically scheduled systems is to derive the worst case processor demand of components (applications) based on the worst case processor demand of their inner sub-components (tasks). The Worst Case Execution Time (WCET) of tasks are the basic blocks of the analysis which are used in calculating the processor demand of applications. A processor capacity that can guarantee the worst case processor demand of the corresponding application is assigned to each application.

While deterministic timing analysis based on WCET is used in hard real-time systems, in soft real-time systems stochastic analysis based on probabilistic execution time distributions can reduce the pessimism and hence the processing capacity can be better utilized [4], [5]. In essence, the flexibility of soft real-time systems in allowing a limited amount of timing violations makes it possible to use probabilistic models of execution time instead of WCET. Hence, in the probabilistic domain the type of guarantees that the analysis provides is not deterministic anymore, i.e., instead of studying the schedulability of the system, the probability of having a deadline miss ratio below a certain level is of interest in the probabilistic analysis. In soft real-time systems, the Quality of Service (QoS) can be inferred from the probabilistic guarantees.

In our work, we bring the probabilistic analysis to the context of a hierarchical scheduling framework. This approach is especially beneficial in open real-time systems in which applications are added and removed during run-time. For instance, when the processor is overloaded, we can leverage the probabilistic model of applications’ processor requirements to decide the processor capacities assigned to the applications such that the overall timing violations are minimized. In fact, our aim is to propagate the uncertainty in task execution times to the application interfaces and to utilize this information at the integration phase.

Previous works in the area of probabilistic analysis mainly target flat systems, i.e, non-hierarchical systems. The closest work to our work is presented by Santinelli et al. which introduced a component based framework for probabilistic analysis of real-time systems [6]. Our work is different from theirs in that we assume deterministic processor provision through conventional reservation based scheduling techniques, and we use the probabilistic processor requirements of applications to derive the reservation sizes assigned to them. While in [6] the probabilistic component demands and probabilistic processor provisioning is used to analyze the probability of schedulability.

II. SYSTEM MODEL

We assume a single processor system consisting of $N$ applications. The applications are scheduled on the processor using a “global scheduler”. Applications, in turn, are responsible for scheduling their inner tasks. Although we start by investigating single processors, our ultimate goal is to extend the work and address multiprocessors.

A. System development model

We target a component based software development model in which the following two roles are defined: (i) application developer (ii) system integrator. The application developer is responsible for developing real-time tasks and selecting an appropriate scheduling policy for them. Then, the application requirement is abstracted using a number of interface parameters. The system integrator on the other hand, receives a number of applications and he/she is responsible for integrating the applications such that the requirements specified in the interface parameters are respected. The integrators’ task involves assigning sufficient processor capacities to applications and choosing an appropriate global scheduler.

B. Task model

In our work we assume a sporadic task model in which task $\tau_i$ is represented with the following parameters: minimum inter arrival time $T_i$, deadline $D_i$ and execution time $C_i$. The execution time is assumed to be a random discrete variable with a known Probability Function (PF) $f_{C_i}(\cdot)$.

C. Application interface

Each application $A_j$ is a set of $n_j$ sporadic tasks $\{\tau_{1j}, \ldots, \tau_{nj}\}$. Applications express their processor requirements using the following two parameters called interface parameters: $P_j$ and $Q_j$ where $P_j$ is the application period,
while \( Q_j \) is a random discrete variable with a known PF \( f_{Q_j}(\cdot) \) which denotes the amount of processor requirement of \( A_j \) every \( P_j \) time units. The application interface is provided by the application developer to the system integrator.

D. Server model

The processor capacity becomes available to the applications through periodic servers compliant with the periodic resource model introduced in [3]. The periodic servers are expressed using the following parameters: \( \Pi_j \) and \( \Theta_j \). The periodic servers provide \( \Theta_j \) time units of the processor time to application \( A_j \) each \( \Pi_j \) time units. In fact, the system integrator designs servers based on the application interfaces.

Although we are targeting soft real-time applications, it is still beneficial to use hard reservations such as periodic servers to (i) handle the system complexity through processor partitioning (ii) keep the timing isolation among applications (iii) reason about the system performance at design stage.

E. From a task set to application interface

In a deterministic hierarchical framework, the application interfaces often express the minimum amount of processor capacity that can guarantee the schedulability of the application’s inner tasks. Hence, the system integrator has to choose a server that provides a processor capacity at least equal to the application requirement. In fact, the server interfaces (\( \Pi \), \( \Theta \)) express both processor requirement of applications and processor supply of the servers. In contrast, the application requirement and server supply are two separate interfaces in our probabilistic hierarchical scheduling framework.

Moreover, the processor requirement of the applications is often abstracted using a demand bound function \( \text{dbf}(t) \) which denotes the maximum processor time needed by the task set inside one application in time interval \( t \). On the other hand, the processor supply through servers is often abstracted using a supply bound function \( \text{sbf}(t) \) that denotes the minimum amount of processor capacity provided to the application in time interval \( t \). The application developer has to provide an interface such that the minimum processor supply through that interface fulfills the following inequality for all time intervals:

\[
\text{sbf}(t) \geq \text{dbf}(t).
\] (1)

In the deterministic analysis, \( \text{dbf}(t) \) returns one value for each given input \( t \), and the objective is to find the minimum \( \text{sbf}(t) \) (and hence the optimal server parameters) that fulfills the above inequality. However, in our probabilistic framework, we redefine the demand bound function \( \text{dbf}^*(t) \) such that for each input \( t \) the output is a probability function with a known distribution that expresses the probability distribution of the processor demand in interval \( t \):

\[
\text{dbf}^* : t \rightarrow f_D(\cdot),
\]

where \( f_D(x) = P(\text{dbf}(t) = x) \). Besides, the objective of our probabilistic framework is to understand for a given server budget \( \Theta_j \) to what extent Inequality 1 holds.

The probabilistic interface is derived from the probabilistic execution times. We suggest the following algorithm for calculating the probabilistic budgets. In the algorithm, \( Q_t \) is the application budget that is under analysis. The goal is to derive the probability of application \( A_j \) being schedulable given that \( \Theta_j = Q_t \). We intend to analytically derive a range for \( Q_t \), \( t \) is the time interval that has to be taken into account in the analysis. Similarly, we will bound \( t \) to bound the number of iterations in the algorithm. For bounding \( t \), we will use techniques analogous to the ones used in deterministic compositional analysis which use the scheduling policy and task parameters for confining \( t \).

\[
\text{Algorithm 1: calculating } f_{Q}(.).
\]

1: for all \( Q_i \) \( \in \left[ Q_{min}, Q_{max} \right] \) do
2: for all \( t_n \) \( \in \left[ 0, t_{max} \right] \) do
3: \( f_D = \text{dbf}^*(t) \);
4: \( p(Q_i, t_n) = f_D(Q_i) \);
5: end for
6: \( f_{Q}(Q_i) = \min(p(Q_i, t_n)) \);
7: end for

In Line 3 of Algorithm 1, the probability distribution of the demand bound function for a given time interval \( t \) is calculated. In Line 4, the probability of the demand being equal to the budget under investigation (i.e., \( Q_t \)) when the time interval is equal to \( t_n \) is stored.

F. From application interfaces to periodic servers

Given a set of application interfaces, the system integrator has to derive a set of periodic servers such that the overall deadline miss ratio is minimized. When the system is overloaded, the servers can be designed according to the applications’ worst case processor requirements. However, when the system is overloaded, some applications have to receive smaller processor portions than their worst case requirement. Hence, the probabilistic budget \( Q \) provides the system integrator with valuable information to decide which application that should be sacrificed such that minimal damage is imposed to the system.

This mechanism can also be used by an online admission controller which decides whether or not a new application should enter the system and if yes how should the overall processor capacity be redistributed.

REFERENCES

QoS-aware AFDX: benefits of an efficient priority assignment for avionics flows

Tasnim Hamza, Jean-Luc Scharbarg, Christian Fraboul
Université de Toulouse - INPT/ENSEEIHT - IRIT
2, rue Camichel - 31000 Toulouse, France

Abstract—AFDX (Avionics Full Duplex Switched Ethernet) standardised as ARINC 664 is a major upgrade for avionics systems. The certification imposes to guarantee that the end-to-end delay of any frame transmitted on the network is upper-bounded and that no frame is lost due to buffer overflow. This guarantee is obtained thanks to a worst-case analysis assuming a FIFO scheduling policy of flows on each output port.

For future aircraft, it is envisioned to modify AFDX switch and to use a Fixed Priority scheduling policy of flows (QoS AFDX using IEEE 802.p mechanisms). A worst-case analysis of such a network has been proposed, based on the Trajectory approach.

But the remaining issue is to efficiently assign available priorities to the avionics flows inside the network without modifying the application knowledge. The objective is then to minimise overall the worst case end to end delay of flows and consequently to minimise needed buffer size at switch level.

The main contribution of this paper deals with the assignment of priorities to the flows using the well-know Optimal Priority Assignment algorithm (OPA) which was first defined for monoprocessor preemptive systems. The schedulability test is then based on the worst case delay analysis of each flow allocated on the AFDX QoS network computed by the trajectory approach.

The proposed mechanisms have been applied on an industrial AFDX case study using two priority levels and the overall worst-case delay could be reduced by 20%.

I. CONTEXT

Full duplex switched Ethernet eliminates the inherent indeterminism of vintage (CSMA-CD) Ethernet. Nevertheless, it shifts the indeterminism problem to the switch level where various flows can compete for output ports.

Avionics AFDX multicast flows are called Virtual Links (VLs) [1]. They are statically defined (burst and rate contract) and are statically mapped on the network of AFDX switches.

For a given VL, the end-to-end communication delay of a packet is the sum of transmission delays on links and latencies in switches. As the links are full duplex there is no packet collision on links. The transmission delay only depends on the transmission rate and on the packet length. But, the latency in switches is highly variable because of the confluence of asynchronous VLs, which compete on each switch output port (according to a scheduling policy).

Many work has been devoted to the worst case analysis of end-to-end delays on an AFDX network implementing FIFO scheduling policy. For certification reasons, a first tool, based on the Network Calculus theory and implemented by Rockwell Collins, has been proposed for the computation of an upper bound for the end-to-end delay of each VL. This approach models the traffic on the AFDX network as a set of sporadic flows with no QoS classes differentiation. A second approach based on the trajectory concept has been proposed [2]. It identifies for a given frame all the competing frames which can delay this frame in all the output ports visited.

This second approach has been generalised in order to implement a Fixed Priority scheduling policy of QoS-aware AFDX network [3]. A worst-case analysis of such a network has been proposed. It can be applied for any number of priority levels and gives tight end-to-end delay upper bounds.

The remaining issue is to assign priorities to VLs in order to demonstrate the efficiency of QoS aware switches. A relevant objective for this priority assignment consists in minimising the overall worst case end-to-end delay of VLs. Indeed, the worst-case end-to-end delay of a VL highly depends on its path. When all the VLs have the same priority, it leads to very different worst-case delays for different VLs. Assigning priorities to VLs should reduce this difference, leading to smaller higher delays. This priority assignment must be done without additional knowledge of avionics flows.

II. PRIORITY ASSIGNMENT FOR QoS-AWARE AFDX

An Optimal Priority Assignment algorithm (OPA) has been proposed in the context of monoprocessor preemptive systems [4]. For an asynchronous periodic task set, OPA generates an optimal priority ordering while using a polynomial number of schedulability tests. It first assigns the lowest priority to one task which respects its deadline with this lowest priority. It continues till the remaining unassigned set of tasks is empty. If at a step no task can be assigned the current priority, no feasible priority assignment exists. At each step, a schedulability test is applied to the task which is assigned the current priority in order to determine whether it respects its deadline or not. The schedulability test has to be OPA compatible. It means that it has to respect the following conditions:

- **Condition 1:** Schedulability of a task may, according to the test, be dependent on the set of higher priority tasks, but not on their relative priority ordering.
- **Condition 2:** Schedulability of a task may, according to the test, be dependent on the set of lower priority tasks, but not on their relative priority ordering.
- **Condition 3:** When the priorities of any two tasks of adjacent priority are swapped, the task being assigned the higher priority cannot become unschedulable according to the test, if it was previously deemed schedulable at the lower priority.
A first extension of OPA has been proposed in order to minimise the number of priority levels [4]. Indeed concrete systems support a limited number of priority levels. The minimisation is achieved by successively maximising the number of tasks assigned to priority levels from the lowest one to the highest one. OPA has also been extended to other types of systems (e.g. multiprocessor systems) [5]. In most cases the priority assignment is no more optimal since the schedulability test is not exact. However OPA-based solutions are often better than solutions considering heuristic priority assignment [5].

**Algorithm 1: Proposed approach algorithm**

```latex
for (each priority level i, lowest first) do
    for (each unassigned VL v) do
        if (v feasible with priority i assuming that all unassigned VL have higher priority) then
            assign priority i to v;
        end
    end
    if (no VL is feasible with priority i) then
        return unschedulable;
    end
    if (no unassigned VL remains) then
        break;
    end
end
return schedulable;
```

In this paper, we propose the extension of OPA for the assignment of priorities to VLs transmitted on a QoS-aware AFDX. The approach is summarised in Algorithm 1. Since the goal is to minimise the overall worst-case delay of any VL transmitted on the network, the proposed solution consists in considering that this overall worst-case delay is the deadline of all the VLs. The approach minimises priority levels: as proposed in [4], it assigns as many VLs as possible to each priority level. The remaining issue is to get an OPA-compatible schedulability test. Such a test is presented in the next paragraph.

### III. Schedulability Test

The trajectory approach computes a sure upper-bound of the end-to-end delay of a VL transmitted on an AFDX network implementing either FIFO scheduling [2] or Fixed Priority (FP) scheduling [3]. The solution proposed in this paper is based on the trajectory approach for FP. The trajectory computation includes all the delays encountered by a frame of a given VL on its trajectory, namely:

- the transmission time of the considered frame on links,
- switching latencies,
- delay due to the workload of the competing VLs with the same priority as the VL under study,
- delay due to the workload of the competing VLs with higher priority than this of the VL under study,
- delay due to the workload of the competing VLs with lower priority than this of the VL under study.

This computation is OPA-compatible.

- The workload of higher priority VLs is computed without considering their relative priority order. Thus condition 1 is respected.
- The workload of lower priority VLs is the largest frame with lower priority. Thus it does not depend on the relative priority order of these VLs and condition 2 is respected.
- Increasing the priority of a VL has two impacts. First, higher priority VLs can become same priority VLs or lower priority VLs. Second same priority VLs can become lower priority VLs. It can be shown that none of these impacts can increase the overall workload of competing VLs. Thus condition 3 is respected.

### IV. First Results

The proposed approach has been applied to a realistic AFDX configuration including 8 switches and 984 VLs (see [6] for a description of the configuration). The worst-case delays of VLs assuming FIFO scheduling are between 0.2 ms and 15.4 ms. Thus we fix the deadline for any VL to 12.3 ms (roughly 80 % of the worst-case delay with FIFO). The approach proposed in this paper is able to assign priorities such that all the VLs respect their deadlines. Only two priority levels are needed.

### V. Conclusion

First results on priority assignment of avionics flows on a realistic AFDX configuration are promising: the overall worst-case delay is reduced by 20 % with two priority levels.

Further evaluation is still needed in order to better estimate the improvement that could be obtained with of a larger number of priority levels.

Moreover the minimisation of the overall worst-case end to end delay is interesting in order to minimise the size of needed buffer of a QoS-aware switch. Then the test should concern the maximum backlog in any switch. Such a test has been proposed in [7] for two priority levels. It has to be extended to any number of priority levels.

### REFERENCES


Supporting Multi-Hop Communications with HaRTES Ethernet Switches

Mohammad Ashjaei¹, Paulo Pedreiras², Moris Behnam¹, Luis Almeida³, Thomas Nolte¹
¹ MRTC/Mälardalen University, Västerås, Sweden
² DETI/IT/University of Aveiro, Aveiro, Portugal
³ IT/DEEC/University of Porto, Portugal

Abstract—In this paper we identify the challenges of multi-hop communication when using micro-segmented switched-Ethernet with HaRTES switches. These switches provide dynamic virtual channels that can be composed hierarchically and provide bounded latency together with temporal isolation. Herein we propose two different solutions regarding the traffic forwarding in multi-switch architectures, while maintaining the unique properties of the single HaRTES switch case. In the first approach, the traffic is buffered and scheduled sequentially in each hop. In the second approach the traffic is scheduled once and forwarded immediately through multiple switches without buffering. In this paper we present a brief comparison of both approaches and we report on the on-going work towards effective support to real-time communications in dynamic and complex Cyber-Physical Systems.

I. INTRODUCTION AND MOTIVATION

Cyber-Physical Systems (CPS) are nowadays pervasive, being present in countless aspects of everyday life, such as medical devices, industrial process control and automotive systems, to name a few. The complexity of CPS has grown fast and reached the stage in which conventional technologies and development methodologies reveal limitations [1]. Focusing specifically on the data distribution infrastructure, CPS pose new requirements that are not efficiently handled by existing communication protocols. For example, many CPS incorporate traffic with diverse activation patterns (event- and time-triggered) and timeliness constraints (e.g. hard, soft and firm), that must be handled in a dynamic way, providing on-the-fly reconfiguration support without service disruption.

The inefficient support of existing communication protocols to CPS led to the development of the new Hard Real-Time Ethernet Switches (HaRTES) [2] [3]. Specifically, these switches aim at supporting: 1) heterogeneous traffic classes with temporal isolation; 2) partitioning and virtualization mechanisms; 3) hierarchical multi-level server composition; 4) dynamic adaptation and reconfiguration of message streams with temporal guarantees.

II. PROBLEM DEFINITION

We define the HaRTES architecture as a micro-segmented network using HaRTES switches. We also consider two traffic types, synchronous and asynchronous. The synchronous traffic is scheduled by the switch, following a master-slave paradigm, in a FTT fashion [4]. Asynchronous traffic is transmitted autonomously by the source-nodes and immediately forwarded by the switch through a hierarchy of servers, in order to enforce isolation and to guarantee minimum QoS levels.

The switch organizes the communication in fixed duration time slots, designated Elementary Cycles (ECs). ECs are composed by two disjoint windows, one carrying the synchronous traffic and the other one the asynchronous traffic. Each EC starts with a Trigger Message (TM), transmitted by the switch, which contains the EC-schedule, i.e., the identification of which synchronous messages shall be transmitted in that EC. Synchronous and asynchronous streams can be real-time, being confined to virtual communication channels, following a multicast-based publisher-subscriber model. Each channel has a set of real-time attributes that are enforced by the HaRTES switch. Such attributes include deadline, minimum inter-arrival time/period and priority.

Until this moment, only single switch HaRTES architectures have been considered. In this work we report on the on-going efforts to build multi-switch HaRTES architectures without jeopardizing the unique dynamic, real-time and traffic separation capabilities of the HaRTES switch. Such a problem has already been addressed in the scope of the FTT-SE protocol, which operates over COTS switches with a software layer in the end nodes that controls all traffic submitted to the network [5]. However, the extended traffic control capabilities of HaRTES open the way to more efficient solutions. Thus, herein we propose two possible architectures with multiple HaRTES switches, that allow forming complex topologies, while preserving the timeliness guarantees of the real-time traffic.

III. PROPOSED SOLUTIONS

In order to build architectures with multiple HaRTES switches, we propose connecting them in a tree topology (Fig. 1). In this architecture we define two types of messages. The messages transmitted between nodes connected to the same switch are called local, otherwise they are called global. Moreover, we define the connections between nodes and a switch as local-links, whereas the connections among the switches are defined as inter-links. Finally, the HaRTES switch on the top of the tree is called the root switch.

Fig. 1. The Multi-Hop HaRTES Network
A. Distributed Global Scheduling

In this solution, the scheduling of global traffic is carried out in a distributed fashion by all the switches involved. Basically, each switch schedules its hop without distinction of global or local messages. The global messages received by a switch are buffered and scheduled for the next hop in following ECs. The step-wise scheduling of the global messages continues until the last switch, where the destination node is connected to.

Assume a network depicted in Figure 1 and consider \( m_1 \) to be transmitted from node C to node D. H2 schedules \( m_1 \) to be transmitted to H1. H1 then schedules \( m_1 \) to be transmitted to H3 with an adequate offset so that it always occurs after the reception of \( m_1 \) from H2. Finally, H3 schedules \( m_1 \) for transmission to node D, again using an adequate offset.

The definition of consistent offsets in the transmission of synchronous messages across multiple switches requires global synchronization. The HaRTES architecture guarantees a contention-free transmission of the TM, which is broadcasted to all the nodes connected to a switch, including other switches down the tree topology. Therefore, the TM can be used as a precise time mark. In this proposal, all switches synchronize with their parent switch whenever receiving a TM. The exception is the root switch, which behaves as a time master.

The asynchronous traffic is forwarded immediately by the switches, within the asynchronous windows, only. During the synchronous windows, or when the capacity of associated servers is exhausted, such traffic is suspended and queued.

B. Synchronous Global Scheduling

In this solution, the global messages are scheduled globally by the root switch. The global scheduling in each EC is encoded in a particular message called Global Trigger Message (GTM) which is broadcasted to the other switches during the synchronous window, i.e., the GTM acts as a global synchronous message. The GTM is received by the other switches, it is decoded and the ID’s of scheduled global messages are inserted into the local TM at the source switches, only, to trigger the global messages. The switches in the route of the global messages forward them immediately in the same EC in which they are scheduled. This way, the global messages are propagated through the network in just one EC.

The synchronous window is divided into two sub-windows, local and global. This partitioning is carried out just for the local-links. The inter-links are not divided since there is no local traffic transmitted through them. The EC windows allocation is depicted in Figure 2 for different links. Moreover, the assigned global sub-window in each link can vary depending on the global load, which can increase the efficiency in using the links bandwidth.

Since the global messages take longer to propagate through the network due to additional forwarding delays, to guarantee their transmission within the synchronous window we enforce the global traffic to be transmitted by the nodes earlier than the local traffic. This can be done by assigning higher priorities to the global messages or by sequencing the IDs in the TM according to the desired order and forcing end-nodes to follow this sequence.

Time synchronization among the switches is also required for this solution. The synchronization can be carried out similarly to the previous solution using the TM.

This approach bears some resemblances with the solution for FTT-SE presented in [5]. However, contrarily to the FTT-SE scenario, in this case it is possible to have differentiated windows allocation, i.e., each link may have an individual global window, with a duration computed according to the traffic that effectively crosses that link. This feature is expected to allow a significant performance improvement. In fact, note that the local scheduling can be carried out after the reception of the GTM. Thus, the local scheduler can deduce the effective time taken by the already scheduled global messages (remember they are scheduled 1 EC ahead) and use the remainder of the synchronous window for local traffic.

IV. CONCLUSION AND FUTURE WORK

In this paper we identified the challenges that arise from connecting multiple HaRTES switches together. This will allow benefiting from the enhanced dynamic, timing and isolation features of the HaRTES switches, to cater for the requirements of complex CPS. We proposed two solutions to handle traffic forwarding, one is based on buffering the traffic in each hop and the other is by transmitting the global messages in a synchronized manner. The former requires several ECs to transmit the global traffic hop-by-hop, whereas in the latter we scheduled and transmit the global traffic in one EC. Also, in the former solution we need to dedicate a smaller window for the message transmission, while we need a bigger global synchronous window in the second solution to be able to send the traffic through different switches. The on-going work aims at developing the timing analysis for both solutions.

ACKNOWLEDGMENT

This work is supported by the Swedish Foundation for Strategic Research via MRTC at Mälardalen University. Also, it is partially supported by the Portuguese Government through FCT grant Serv-CPS PTDC/EEA-AUT/122362/2010, FCT grants SENODS CMU-PT/SAI/0045/2009 and FEDER (COMPETE program).

REFERENCES

Abstract—The WCET (worst case execution time) of a program is crucial to guarantee that the timing constraints of real-time systems are met. Instruction scheduling and register allocation play an important role for reducing the WCET of a program. We study the problem of performing instruction re-scheduling and register re-allocation for basic blocks of a program executed on a clustered VLIW processor such that the WCET of the program is minimise, and propose a new unified approach.

Index Terms—worst case execution time; register allocation; instruction scheduling; clustered VLIW processor

I. INTRODUCTION

In real-time systems, it is crucial to guarantee that all the timing constraints are met at design stage. The WCET (Worst-Case Execution Time) of each task has a significant impact on finding a feasible schedule for a set of real-time tasks. If we can reduce the WCET of each task, it’s more likely to find a feasible schedule. Therefore, it is an important problem to minimize the WCET of each task.

Instruction scheduling and register allocation are two important components of an optimising compiler. Traditionally, register allocation is performed separately from instruction scheduling, causing the phase ordering problem. Therefore, it is desirable to integrate register allocation and instruction scheduling into a single phase. Clustering is a well-known technique for improving the scalability and energy consumption of VLIW processors. However, clustered VLIW processors make instruction scheduling and register allocation more challenging. Firstly, new live intervals may be created dynamically when values are transferred to different clusters, and more than one registers are needed to keep the copies of values on different clusters. Secondly, the exact live range of a variable depends on when the operations of its first definition and last use are scheduled, which cannot be determined by traditional liveness analysis for static code. Thirdly, bad cluster assignment may cause unnecessary inter-cluster communications, thus increasing the schedule length of a basic block.

In this paper, we propose a unified approach for instruction re-scheduling and register re-allocation which aims at minimizing the WCET of a program. Our approach consists of two phases. In the first phase, it uses our unified approach [1] to perform instruction scheduling and register allocation to all the basic blocks in the reverse postorder. In the second phase, it performs instruction re-scheduling and register re-allocation for all the basic blocks having register spills, sorted by the lengths of the longest paths they belong to and their worst case execution frequencies. Our approach differs from the state-of-the-art approach [2] in the following major aspects. Firstly, our approach applies not only instruction scheduling but also register allocation before performing instruction re-scheduling and register re-allocation while the previous approach only performs instruction scheduling. Hence, the longest path selected by our approach is more reliable. Secondly, during instruction re-scheduling and register re-allocation, our approach selects a basic block with the maximum worst-case execution frequency on the longest path while the previous approach selects a basic block with the maximum ILP (Instruction Level Parallelism) on the longest path. Since a basic block with the highest worst-case execution frequency on the longest path has the most impact on reducing the length of the longest path, it should be selected first. Thirdly, we assign a dynamic priority for each operation of a basic block to reduce the degree of ILP when register pressure is high, resulting in fewer register spills, while the previous approach does not handle high degree of ILP, which may result in a lot of register spills.

II. RELATED WORK

There have been lots of approaches to compiler optimisations to reduce the ACET (average case execution time) of a program. However, optimisations for minimising ACET may lead to spills along the longest execution path, thus increasing WCET considerably. Compared to the works on reducing ACET, there have been fewer works on compiler optimizations to reduce WCET. Zhao et al. [3] describe their approach to reduce WCET by adapting optimizations designed for frequent paths to the worst-case paths. Zhao et al. [4] propose a WCET code-positioning algorithm that searches for the best layout of the code in the memory for WCET. Falk [5] presents the first WCET-aware register allocator such that it uses precise WCET models to avoid spill code generation along the critical path defining WCET. Falk et al. [6] also present an integer-linear programming based register allocator using precise WCET models. Huang et al. [2] propose a WCET-aware rescheduling and register allocation approach for real-time embedded system with clustered VLIW architecture. Their approach consists of two phases. The first one performs an initial aggressive instruction scheduling without considering
register pressure. The second one reschedules operations and allocate registers to reduce the number of register spills.

III. WCET-AWARE INSTRUCTION SCHEDULING AND REGISTER ALLOCATION

A program $P$ is represented by a weighted CFG (Control Flow Graph) $G = (V, E, W)$, where $V = \{B_1, B_2, \ldots, B_n\}$: $B_i$ is a basic block of the program}, $E = \{(B_i, B_j) : B_j$ is directly control dependent on $B_i\}$, $W = \{w_i'^{l} : w_i^{l}$ is the execution time of $B_i\}$. We convert $G$ into a DAG (Directed Acyclic Graph) as follows. $\text{DAG}(G) = (V', E', W')$, where $V' = V$ is the set of basic blocks of $P$, $E' = E - \{(B_i, B_j) : (B_i, B_j)$ is a back edge\}, is the set of edges of $\text{DAG}(G)$, and $W' = \{w_i'^{l} : w_i'^{l}$ is the node weight of $B_i$, and $w_i'^{l} = w_i \cdot N(B_i)\}$, where $w_i$ is the execution time of $B_i$, and $N(B_i)$ is the worst-case execution frequency of $B_i$.

Our approach consists of the following steps:

1) Construct the weighted CFG $G$ of the program $P$, where $W_i$ is undefined.
2) Perform instruction scheduling and register allocation to each basic block $B_i$ in the reverse postorder by using our unified approach [1].
3) Update $W_i$ for each basic block $B_i$ in $G$.
4) Construct $\text{DAG}(G)$.
5) Repeat the following until the longest path of $\text{DAG}(G)$ cannot be reduced anymore.
   a) Compute the longest path of $\text{DAG}(G)$.
   b) Find the basic block $B_k$ ($B_k \in V$) which has the maximum worst-case execution frequency among all the basic blocks having register spills on the longest path.
   c) Perform instruction re-scheduling and register re-allocation for $B_k$.
   d) Update the node weight of $B_k$ in $\text{DAG}(G)$.

During the first phase, our unified approach [1] integrates an incremental register allocator into a priority based instruction scheduler [7]. The instruction scheduler schedules all the basic blocks in the reverse postorder and all the operations in each basic block by their priorities. The priority of each operation considers the inter-operation latencies and the processor resource constraints. During the scheduling, the operation priorities are dynamically updated to reduce the register pressure. For a ready operation with the highest priority, the instruction scheduler assigns the operation to a functional unit on a cluster and uses the incremental register allocator to allocate physical registers to the virtual registers of the operation. The cluster assignment considers both the start time of the operations and the register pressure on each cluster.

In the second phase, we aim at minimising the total number of register spills on the longest paths by using register re-allocation and instruction re-scheduling. At each time, we select a basic block $B_i$ with the maximum worst-case execution frequency on the longest path of $\text{DAG}(G)$, and for each live range $R_j$ spilled to the memory in $B_i$, perform the following steps to reduce the spills.

1) Find a live range $R_k$ that has the least impact on the longest path of $\text{DAG}(G)$ such that $R_k$ contains $R_j$ and allocate the register holding $R_k$ to $R_j$.
2) Add spill code of $R_k$ and re-schedule all the instructions affected.
3) Re-calculate the execution time for each basic block affected by $R_k$.

In order to find such a live range $R_k$, we introduce a new graph, namely, $\text{DAG}(k)$ as follows. $\text{DAG}(k)$ is a subgraph of $\text{DAG}(G)$ such that the length of any path in $\text{DAG}(k)$ is no more than $k + l_{min}$, where $l_{min}$ is the length of the shortest path of $\text{DAG}(G)$. We set $k$ to $(l_{max} - l_{min})/2$, where $l_{max}$ is the length of the longest path of $\text{DAG}(G)$. $\text{DAG}(G, k)$ can be constructed by using breadth-first search in $O(e)$ time [8], where $e$ is the number of edges in $\text{DAG}(G)$.

After constructing $\text{DAG}(G, (l_{max} - l_{min})/2)$, compute the rank of each live range $R_s$ as follows. rank($R_s$) = $n1(R_s)/n(R_s)$, where $n1(R_s)$ is the total number of references to $R_s$ in the basic blocks of $\text{DAG}(G, (l_{max} - l_{min})/2)$, and $n(R_s)$ is the total number of references to $R_s$ in all the basic blocks. The live range $R_k$ on the longest path selected is the one with the maximum rank.

IV. CONCLUSION

We propose a WCET-aware approach to unified instruction re-scheduling and register re-allocation for clustered VLIW processors. We will evaluate our approach by using a set of benchmarks and compare it with the state-of-the-art approach proposed in [2].

V. ACKNOWLEDGEMENT

This work is sponsored by the National Natural Science Foundation of China under Grant No.61303072.

REFERENCES

XDense: A Sensor Network for Extreme Dense Sensing†

João Loureiro‡, Vikram Gupta, Nuno Pereira, Eduardo Tovar, Raghuraman Rangarajan
CISTER/INESC-TEC, ISEP, Polytechnic Institute of Porto, Porto, Portugal
Email: {joflo, vigup, nap, emt, raghu}@isep.ipp.pt

Abstract—We introduce XDense, a grid topology based wired sensor network architecture tailored for high sampling rate applications and high density of sensor node deployments. The purpose of such an architecture is to monitor, detect and report features of observed dynamic phenomena, in a distributed fashion. We present preliminary simulations validating our concept.

I. INTRODUCTION

Recent advances with respect to performance, cost and size of Micro-Electro Mechanical Systems (MEMS) have revolutionized the design of sensors, actuators, and other microsystems. This has made possible new solutions for many applications ranging from fundamental scientific research to industrial process control. For example, the measurement of shear-stress to determine flow conditions over time is of high importance in highly dynamic application scenarios like monitoring the air-flow over an aircraft wing [1].

Such applications require sensors to be smaller than the spatial granularity of the observed phenomena (of 100 μm or less) and to have high sampling rates (in excess of 10 kHz) [2]. For such scenarios, there are some limitations in the current technologies. Battery-powered wireless sensors can suffer from size limitations and transmission latency due to concurrency issues. Wired sensor networks with shared buses limit the scalability of the system and are susceptible to noise.

Our work moves away from traditional wireless and wired sensor network approaches and resembles Network-on-Chip (NoC) architectures closely, regarding some aspects like the network architecture topology, routing schemes and timing analysis [3]. On the other hand, we believe that the key differentiating features are: (a) The network is not on a single chip, but on a larger surface, which should physically interfere as less as possible with the observed phenomena, and (b) the node count is likely to be much greater than the one usually found in NoC applications.

We propose a grid network architecture for very dense sensor deployments, able to distributively detect events without the need of centralized data acquisition and processing. Moreover, it reduces number of transmissions between the nodes and the sink in order to update the sink about the overall scenario. We now describe the architecture of the proposed system and provide preliminary simulation analysis to support our proposition.

†This work was partially supported by National Funds through FCT (Portuguese Foundation for Science and Technology) and by ERDF (European Regional Development Fund) through COMPETE (Operational Programme Thematic Factors of Competitiveness), within projects ref. FCOMP-01-0124-FEDER-037281 (CISTER), ref. FCOMP-01-0124-FEDER-020312 (SMARTSKIN), ref. FCOMP-01-0124-FEDER-012988 (SENODS) and ref FCOMP-01-0124-FEDER-028990 (PATTERN).
‡João Loureiro is supported by the government of Brazil through CNPq, Conselho Nacional de Desenvolvimento Científico e Tecnológico.
by establishing a connection path to the sink. This is done using the handshaking pin for the control signal. The adjacent node, on the path to the closest sink, gets the request in its handshaking input, and if it is not busy, it propagates the signal in the direction of the sink. Intermediate nodes do the same until the sink is reached. The sink responds with a handshake in the reverse direction granting the connection. Through this signalling process, a simple through-line circuit is established between the node and the sink. The node now sends the data to the sink over the established circuit.

In this setup, the most important metric to be analysed is the latency of the event announcement to the sink by the sensor node. This is because, latency in the network determines the speed of detection, that is, the reaction of the sensor network to the changes in the physical phenomenon. In order to study the latency metric, we simulate an example setup described in the following setup.

III. SIMULATION AND PRELIMINARY RESULTS

To simulate our sensor node, we based the behaviour of our model on a low cost and low power microcontroller. To the best of our knowledge, this is the best micro-controller with the necessary resources for our network design. The example scenario is a grid sensor network monitoring the pressure distribution due to airflow over a surface. We take a sample static snapshot, of pressure over a surface, to analyse the detection and transmission latency of the sensor network. The network setup is as follows: a \((21 \times 21)\) grid network of 440 sensor nodes and one sink. The sink is in the center of the grid. In this example, the feature of interest is any gradient with slope greater than a given threshold. Figure 2(a) shows the grid network superimposed on the static air flow snapshot (the sink is the node in the center).

At the time instant \(t = 0\), all the nodes start the network discovery state. Once the network is discovered, nodes enter the event-monitoring and detection state. Now, sensor data is captured, sent to and received from the neighbourhood by all the sensor nodes in the network in a concurrent manner. After processing the collected data, the nodes that are positioned at high-gradient slopes trigger the \textit{event-announcement state}. The nodes then set up a connection to the sink to send this information. The end-to-end transmission latency from all the sensor nodes, is shown in Figure 2(b) (the colormap represents the transmission latency). Farther a node is from the sink, the greater is the latency due to the number of intermediate hops. Also, the number of simultaneous detections, and hence transmissions, also determine the contention in the network.

Figure 2(c) shows the number of packets forwarded by each node in the grid. A darker node signifies congestion in the path through it. As expected, nodes close to sink forward a larger number of packets than nodes away from it. The transmission paths are also shown in the figure.

IV. CONCLUSIONS AND FUTURE WORK

Low latency is an important goal in applications where the number and frequency of readings are crucial. The XDense architecture, with simple hardware and software functionalities, allows for such deployments. The proposed architecture shows some promising results for scenarios of dense deployments of sensors. We presented three concepts: Simple grid network architecture, sharing neighbourhood information allowing quick detection of events and low-latency setup of network paths to the sinks. Further work is required to examine the significance and efficacy of this approach. We intend to explore many of its aspects like routing, flow control and distributed data processing and aggregation.

REFERENCES

Reconfiguration-based Energy Optimization in Battery Systems: a Testbed Prototype

Liang He, Shaodong Ying, Yu Gu
Singapore University of Technology and Design, Singapore

Large-scale battery packs with hundreds/thousands of battery cells are commonly adopted in many emerging cyber-physical systems such as electric vehicles and smart microgrids. For many applications, the load requirements on the battery systems are dynamic and could significantly change over time. How to resolve the discrepancies between the output power supplied by the battery system and the input power required by the loads is key to the development of large-scale battery systems. Traditionally, voltage regulators are often adopted to convert the voltage outputs to match loads required input power. Unfortunately, the efficiency of utilizing such voltage regulators degrades significantly when the difference between supplied and required voltages becomes large or the load becomes light.

In our recent work [1], we have proposed to address this problem via an adaptive reconfiguration framework for the battery system. Our design is guided by two rule-of-thumb principles: first, the supplied voltage from the battery system should match the load required voltages, and second, the discharge current of individual battery cells should be minimized while the overall output current from the battery system satisfying the load requirement. Based on these design principles and by abstracting the battery system into a graph representation, we developed two adaptive reconfiguration algorithms to identify the desired system configurations dynamically in accordance with real-time load requirements, focusing on the scenario of a single and multiple loads changes respectively.

We extensively evaluated our design with empirical experiments on a prototype battery system, electric vehicle driving trace-based emulation, and battery discharge trace-based simulations. The evaluation results demonstrated that, depending on the system states, our proposed adaptive reconfiguration algorithms can significantly prolong the system operation time. As an example, Figure 1 shows one set of our empirical evaluation results obtained by running the system with the proposed algorithm (i.e., Adaptive) and the non-reconfigurable baseline (i.e., 4S2P) respectively. We can see that compared with the non-reconfigurable battery systems, the system operation time can be significantly prolonged by adaptively adjusting its configurations.

To more clearly demonstrate the feasibility and effectiveness of the proposed system reconfiguration algorithm, after the investigation presented in [1], we have built a 4-battery reconfigurable testbed implemented based on the circuit design proposed in [2], as shown in Fig. 2. We will demonstrate how the board supplied voltage can be adjusted by controlling the cell configurations during the demo. The demo testbed consists of 4 circuit units, each for one cell-level unit. One cell-level unit consists of 1 AA battery cell and 6 RTD34005 relays. These 4 cell-level units can be configured to connect in series or in parallel. As a result, based on different configuration, the reconfigurable board can achieve 5 levels of output voltage (0 V, 1.5 V, 3 V, 4.5 V and 6 V) and 24 different battery discharge profiles. Each battery cell's health status will be monitored by the 8-channel I/O modules through connector P5, which will be the indicator for our adaptive reconfiguration algorithm to determine the desired configuration for the real-time loads. This 4-cell testbed can also served as a unit module for battery systems with larger scales.

The 4-cell testbed not only assist us to verify the feasibility of reconfigurable battery systems, but also offers a promising empirical evaluation platform for the future investigations on them.

REFERENCES


Distributed Real-Time Fault Tolerance on a Virtualized Multi-Core System

Eric Missimer, Ye Li and Richard West

Computer Science Department
Boston University
Boston, MA 02215

Email: {missimer,liye,richwest}@cs.bu.edu

Chip-level multiprocessors (CMPs) are increasingly being used in real-time and embedded systems, due in part to their power, performance and cost benefits. New opportunities exist to build fault-tolerant, safety-critical systems that leverage the redundancy of separate cores on these processing platforms. In this work, we describe a new chip-level distributed real-time system, called Quest-V.

Quest-V uses hardware virtualization to partition machine resources (processor cores, memory, and I/O devices) amongst separate sandboxes. Here, a sandbox is similar to a traditional (guest) virtual machine but, for the most part, it operates independently of an underlying hypervisor (a.k.a., virtual machine monitor, or VMM). In Quest-V, each sandbox is bootstrapped by a corresponding monitor that grants access to specific physical resources. For the most part, resources are statically partitioned amongst sandboxes, so a monitor is not needed to schedule the execution of separate guests on available cores, as is done in traditional hypervisors. A trusted monitor is only needed when shared memory communication channels are established between sandboxes, or fault recovery procedures need to be performed.

Quest-V is intended for use in systems where faults can occur either because of software errors, or partial hardware failures caused by factors such as radiation. If a fault occurs in a safety-critical system, it is essential that all critical services remain operational, or available. Quest-V attempts to maintain high availability by performing real-time, on-line fault detection and recovery. It does this using a consensus protocol to capture the states of replicated processes running in separate sandboxes. Any processes not part of the consensus are deemed to have malfunctioned and are restored to the states of other functional replicas. Recovered processes are rolled forward to the most recent checkpointed state of replica processes that are deemed to be working correctly.

Applications requiring high availability use a special sync system call, to periodically checkpoint their state and invoke the consensus protocol. When a sync call occurs, the user space memory of each application instance is hashed on a per-page basis and the resulting hashes are placed into memory shared between the local sandbox and an arbitrator sandbox. A hash is only collected if the page is modified since the last sync call. The arbitrator collects the hashes and performs the consensus. It then sends responses to each sandbox, indicating whether or not the corresponding replica process can proceed past its checkpoint. If a replica passes the checkpoint it can free any copied pages it was keeping for remote recovery. If it fails at a checkpoint the arbitrator signals that it needs to recover.

Part of the recovery process involves making a virtual machine call to enter into a sandbox monitor. While in the monitor, pages from a correct process instance in a remote sandbox are copied to the recovering replica. Upon return from the monitor, the virtual machine state is changed so that it is in the sync system call, which in turn restores control back to the recovered process.

As long as each replicated process does not receive input that would vary across instances of execution, such as calling a read time-stamp counter function, the memory state across all well-behaving instances should be the same. This approach relies on copying and hashing memory pages but a faulty operating system cannot bring down the entire system, as only monitor code can modify memory in another sandbox.

The Quest-V fault tolerance approach is intended for safety-critical applications in areas such as avionics, automotive systems, robotics, healthcare and manufacturing. We are currently developing an autonomous vehicle system, called RacerX, which uses real-time sensor data inputs to influence path planning and motion control. Other applications include the safe, predictable, and fault-tolerant management of unmanned aerial vehicles (UAVs) and robots deployed in space.

We will demonstrate Quest-V using live examples of autonomous vehicle control, for both simulated and real-world scenarios. Demonstrations will include using Quest-V to manage the AI engine of a TORCS simulated vehicle system. We will show that a vehicle avoids collisions with its surroundings even when faults are injected into the AI engine.

1TORCS is “The Open Racing Car Simulator”.

---

22
Fig. 1: Example Quest-V Architecture Overview

Fig. 2: Quest-V with Linux Front-End
Demonstrator for modeling and development of component-based distributed real-time systems with Rubus-ICE

Alessio Bucaioni*, Saad Mubeen†*, John Lundbäck†, Kurt-Lennart Lundbäck†, Jukka Mäki-Turja†* and Mikael Sjödin*

* Mälardalen Real-Time Research Centre (MRTC), Mälardalen University, Västerås, Sweden
† Arcticus Systems AB, Järfalla, Sweden

⇤{alessio.bucaioni, saad.mubeen, jukka.maki-turja, mikael.sjodin}@mdh.se
† {saad.mubeen, john.lundback, kurt.lundback}@arcticus-systems.com

Abstract—We present a demonstrator for modeling and development of component-based vehicular distributed real-time systems using the industrial model Rubus Component Model (RCM) and its development environment Rubus-ICE (Integrated Component development Environment). It demonstrates various stages during the development process of these systems such as modeling of software architecture, performing timing analysis, automatic synthesis of code from the software architecture, simulation, testing, and deployment.

I. BACKGROUND – THE RUBUS CONCEPT

Development strategies for real-time embedded systems in the automotive and other vehicular applications domain are to an extent based on model- and component-based development approach. This approach uses models to describe functions, structures and other design artifacts; and supports the development of large software systems by integration of software components. It raises the level of abstraction for software development and aims to reuse software components and their architectures. Rubus [1], [2] is a collection of methods, theories and tools for model- and component-based development of predictable, timing analyzable and synthesizeable control functions in resource-constrained embedded systems. Rubus is developed by Arcticus Systems in close collaboration with Mälardalen University and is used by several international companies. The Rubus concept is based around RCM and Rubus-ICE which includes the following.

- Designer: A graphical tool for modeling a system based on RCM. It creates a set of XML-files containing the design including deployment information related to selected Run-Time Environment (RTE) and target.
- Analyzer: A graphical off-line and on-line analysis tool. The off-line analysis consists of response-time analysis of tasks and network messages, shared stack analysis, and end-to-end distributed response-time and delay analysis [3]. Whereas, the on-line analysis reads execution trace from the target via a communication channel. The Rubus Analyzer gives a possibility to feed back information from the target.
- Inspector: A graphical component test tool for software as well as hardware in the loop tests.
- Simulator: It builds a simulated environment around the application to allow the control of its execution from a high-level tool such as LabView or Matlab/Simulink.
- Build tools: compiler, linker, and plug-ins launcher.
- Synthesizer: A tool to generate the execution framework for a specific RTE-platform.

An example of software architecture modeled in RCM is shown in Figure 1. The organization and screen shots of Rubus-ICE are shown on the next page.

II. DEMONSTRATION OF DEVELOPMENT PROCESS

We demonstrate the methodology and usage of Rubus tools by modeling and developing a distributed real-time application which is the simplified Intelligent Parking Assist System. It consists of two nodes that run the Rubus operating system and are connected via Controller Area Network. We demonstrate the following steps during the development.

1) Modeling: Developing component-based software architecture of the application with Rubus modeling language.
2) Analysis: Performing different types of analysis available in Rubus-ICE such as the end-to-end response-time and delay analysis and stack-memory analysis.
3) Synthesis: Automatically generating the code for the run-time infrastructure (execution framework).
4) Simulation and Testing: Executing the modeled application in a simulated environment from Simulink and testing at various hierarchical levels.
5) Deployment: Downloading the synthesized software on hardware platform and demonstrating the functionality.

REFERENCES

Screen Shots of the Tools

Organization of Rubus-ICE

Rubus Designer

Rubus Inspector
DREMS: A Toolchain and Platform for the Rapid Application Development, Integration, and Deployment of Managed Distributed Real-time Embedded Systems

William Emfinger, Pranav Kumar, Abhishek Dubey, William Otte, Aniruddha Gokhale and Gabor Karsai
ISIS, Dept of EECS, Vanderbilt University, Nashville, TN 37235, USA

The DREMS\textsuperscript{1} toolsuite is a software infrastructure for designing, implementing, configuring, deploying, operating, and managing distributed real-time embedded systems. It consists of two major subsystems: (1) a design-time environment for modeling, analysis, synthesis, implementation, debugging, testing, and maintenance of application software built from reusable components, and (2) a run-time software platform for deploying, managing, and operating application software on a network of computing nodes. The platform is tailored towards managing a network of computers and distributed software applications running on that network: such as a cluster of networked nodes such as fractionated satellites or a group of smartphones deployed in a coordinated fashion to provide ad-hoc distributed services that can be used in disaster relief.

It is a complete, end-to-end solution for software development: from modeling tools to code to deployed applications. Open and extensible, it relies on open industry (OMG) standards, well-tested functionality, and high-performance tools. It supports a model-based paradigm of software development for distributed, real-time, embedded systems where modeling tools and generators automate the tedious parts of software development and also provide a design-time framework for the analysis of the system. The run-time software platform reduces complexity and increases reliability of software applications by providing reusable technological building blocks: an operating system, middleware, and application management services.

DREMS applications platform are built from software components that interact via only well-defined interaction patterns using security-labeled messages that support Multi-Level Security [1], and are allowed to use a specific set of low-level services provided by the operating system. Low-level services include messaging and thread synchronization primitives, but components do not use these directly, only via the middleware-provided framework abstractions. Specialized services distributed across the platform are used to control the lifecycle and update applications on demand.

The middleware libraries implement the high-level communication abstractions (synchronous and asynchronous interactions) using low-level services provided by the underlying distributed hardware platform. The DREMS Operating System, a set extension to the Linux kernel, implements all the critical low-level services to support resource sharing (incl. spatial and temporal partitioning), actor\textsuperscript{2} management, secure (labeled and managed) information flows, and fault tolerance. The OS also provides strict capability checks for the services an application can use. Three different task levels can exist on the platform: Critical (run as fast as possible), Application (run in a periodic temporal schedule), and Best Effort (run whenever possible).

Configuring the middleware and writing code that takes advantage of the component framework is a highly non-trivial and tedious task. To mitigate this problem and to enable programmer productivity a model-driven development environment is provided that simplifies the tasks of the application developers and system integrators.

**Demonstration:** We cover a complete application development cycle from design in the modeling tools to execution on a set of fanless computing nodes used to emulate a cluster of three satellites. These nodes contain a 1.6 GHz Atom N270 processor and 1 GB of RAM and communicate on a private gigabit subnet. To this subnet are also connected a physics simulation node running the Orbiter spacecraft simulation tool (http://orbit.medphys.ucl.ac.uk/) and a development node running Dummynet[2] to control the subnet's bandwidth, latency, and packet loss on a per-link basis, similarly to Emulab.

Each satellite in the emulated cluster will run two applications of different criticality levels: a cluster management application and a CPU-intensive image processing application. The cluster management application controls the (simulated) satellite hardware (satellite state, propulsion system, etc.) to maintain orbit and ensures safe cluster operation, therefore it is run as a critical application. The image processing application is not as critical and therefore runs in temporal partitions.

This application will demonstrate utility of the platform. We will also show the initial research results from our work on design time verification of properties such as network quality of service (QoS) and component performance characteristics for the applications developed and deployed on the platform.

**Acknowledgments:** This work was supported by the DARPA System F6 Program under contract NNA11AC08C.

\textsuperscript{1}Distributed REaltime Managed Systems
\textsuperscript{2}Actors are processes with persistent identifiers

REFERENCES


Software toolchain for modeling, synthesis, analysis, and verification

Software platform with support for components, resource sharing, security, and fault tolerance

The software toolchain consists of the modeling language, the constraint checking (i.e., information flow checks at configuration time, network admittance checks, and partition schedulability checks), the code generation, and the verification tools. The software platform consists of the operating system services and the middleware infrastructure code which provides all allowable services securely to the components. The management of the applications is also handled by dedicated platform applications which are secure and maintain the lifecycle of the applications.
Fig. 2. Development system and DREMS cluster: The bottom right of the image shows the 3 computing nodes used for this application deployment; the left screen shows their three corresponding satellites simulated in orbiter (which is communicating with the three nodes); and the right screen shows the application development using the modeling tools.
Only satellite 1, which is the cluster leader, communicates with the ground network. When satellite 1 receives a command, e.g. scatter, from the ground network, it relays it to the other satellites so that all satellites can perform the maneuver. All satellites communicate with Orbiter which simulates each satellite’s orbital mechanics.
Fig. 4. Application activity log: Actors 1051, 1052, 1053, and 1054 belong to the CPU-intensive image processing application, which tries to consume as much CPU as possible, but runs in temporal partitions at a lower priority than the cluster management application. Actors 1025, 1026, and 1027 belong to the critical cluster management application which is not constrained by temporal partitioning and runs at a higher priority than the image processing application. The partition schedule of the four image processing application actors is shown at the bottom for reference: Actors 1051 and 1052 belong to partition 3 and actors 1053 and 1054 belong to partition 2. The cluster management application activity is annotated in the activity log.
Atacama: An Open Experimental Platform for Mixed-Criticality Networking on Top of Ethernet

(Extended Abstract)
Gonzalo Carvajal and Sebastian Fischmeister
University of Waterloo
Waterloo, ON, Canada
Email: {gcarvaja, sfischme}@uwaterloo.ca

I. MOTIVATION

Modern real-time distributed embedded systems found in automobiles, airplanes, factories, and medical devices, are pushing the boundaries of traditional fieldbuses due to the ever increasing amount and complexity of the information that is exchanged. For example, applications such as camera-assisted parking and Advanced Driver Assistance Systems (ADAS) in cars will require the integration of multiple subsystems (cameras, braking, steering, etc.), which currently operate in isolation using highly-specialized components with incompatible electronics and protocols. System integration is becoming common across multiple application domains, and is pushing towards the definition of a versatile backbone network capable of providing both the required bandwidth and real-time properties to accommodate everything from safety-critical data to multimedia streams using a common infrastructure.

In this context, Ethernet is gaining acceptance as the networking technology for next-generation distributed systems given its high-speed, low-cost, scalability, and general maturity. However, standard Ethernet components require specific modifications and hardware support for coordinated communication to provide strict latency guarantees for hard real-time applications. During the last decade, most of the research on Real-Time Ethernet (RTE) has been focused on formal analysis to derive latency bounds and proposals to amend the standard protocol, but the practical challenges related to implement and evaluate the required hardware remains overlooked and mostly unexplored. This becomes evident from the few proposed solutions reporting prototypes and experimental validation, which hinders the acceptance and consolidation of Ethernet as an effective replacement of current fieldbuses.

II. PROPOSED DEMONSTRATION

This demonstration is based on results from our on-going work on Atacama, the first fully-implemented open-source framework that relies on reconfigurable hardware to encourage and accelerate the development, validation, extension, and adoption of RTE technology in real-world applications. Fig. 1 shows an example setup and summarizes the achieved timing properties using the developed components on a 1[Gbit/s] network. Enhanced network interfaces use custom programmable modules to execute predefined TDMA schedules for conflict-free communication of real-time data. In addition, custom modules in the switches implement a logical bus-like path with cut-through forwarding dedicated to scheduled frames tagged

Fig. 1. Example setup and observed latency as Real-Time (RT) type. The modules seamlessly integrate to and co-exist with COTS devices operating with standard Best-Effort (BE) traffic, enabling mixed-criticality communication.

The demonstration will showcase an experimental setup for a buffer-less video streaming application through a multi-hop network with additional stations emitting interfering best-effort traffic. To stress the timing requirements, the video source sends scheduled frames encapsulating raw video lines at a period that matches the required period for proper displaying of the video at the other end, and the receiver can only buffer one video line. This configuration enables easy visualization of the achieved timing properties, since any variation in the propagation latency of frames containing video lines will degrade the displayed video. An overview of the application and related documentation is available in [1].

The results are directly extensible to different application domains requiring hard real-time communication. We expect to motivate attendees to provide feedback, comments, and criticism about the framework, and specially encourage them to explore the available technology in their own applications.

RELATED BIBLIOGRAPHY

Demonstrating Dynamic Ethernet Virtualization for Efficient Bandwidth Management

João Reis¹, Tiago Silva¹, Luís Almeida¹, Ricardo Marau², Paulo Pedreiras²
¹DEEC – Faculdade de Engenharia, Universidade do Porto, 4200-465 Porto, Portugal
²DETI – Universidade de Aveiro, 3810-193 Aveiro, Portugal
{ee05100,ee07209,lda}@fe.up.pt, {marau,ppedreiras}@ua.pt

I. Motivation

In the realm of the so-called Cyber-Physical Systems (CPS), particularly larger ones such as industrial sites, intelligent buildings, energy sub-stations, datacenters, etc., the typical systems architecture is distributed and, frequently, over Ethernet. In these systems, multiple applications coexist, frequently involving a large number of data streams with heterogeneous requirements in latency and bandwidth. Moreover, many applications execute for a certain time and then leave or idle for a substantial period.

Supporting these heterogeneous and dynamic requirements efficiently, requires expensive communications equipment, namely switches and routers with high forwarding capacity together with fine grained filtering and strong Quality of Service (QoS) features, e.g. layer 3 or 2.5 carrier-grade switches with RSVP-TE or MPLS, or AVBridges [1]. Another alternative that is generating substantial interest today is the use of Software-Defined Networks (SDNs) [2], which, using OpenFlow [3], allow defining specific traffic forwarding rules and fine grained filters that are then applied to configure the networking equipment accordingly.

However, these solutions are expensive and, in some cases, they are still limited in the number of channels they support and in the strength of the traffic isolation they can provide. This opens the way to simpler solutions that are based on COTS ordinary switches but relying on an adequate virtualization software layer that enforces the network channels. The FTT-SE protocol [4] falls within this category and caters for the needs of such systems, providing dynamic channels that can be adapted on-line and which enforce strong traffic isolation and strictly bounded latency.

In this demonstration, we show the potential of such virtualization layer on Ethernet with a distributed surveillance system that adapts the bandwidth dynamically among the competing channels. This bandwidth management considers dynamic changes in the streams’ QoS requested by an operator together with their current compression levels, resulting in an efficient use of the bandwidth that maximizes the offered QoS on top of and ordinary COTS Ethernet switch. The bandwidth management scheme is similar to that proposed in [5] but here we focus mainly on capacity adaptation per channel as opposed to rate adaptation therein. In the near future we will apply game theory to assess potential benefits in robustness and efficiency.

II. Demonstration Setup

The setup consists of two cameras implemented on Raspberry Pi nodes, an inexpensive low end 100Mbit/s Ethernet switch, an embedded FTT-SE master connected to one of the switch ports, and one display node serving as operator console (Figure 1). The cameras exhibit image resolutions of 640x480 at 15fps, encoded with JPEG for low latency. Both cameras can be connected / disconnected online and their scenarios can also change dynamically between high and low structural complexity. The display channel is configured with a low bandwidth of 2Mbit/s which is not enough for both streams with low compression.

We will then show the following situations:

- Both cameras are alive with the same QoS while the manager adjusts dynamically the streams bandwidth to level their compressions.
- The operator requests a QoS increase in one camera which will cause the manager to assign it more bandwidth allowing lower compression while automatically compensating in the other camera.
- One camera is switched off, which allows the manager to assign the whole display channel to the other camera consequently increasing its bandwidth and decreasing its compression.

![Figure 1. Dynamic QoS management demonstration layout.](image)

REFERENCES


This work was partially supported by FEDER (COMPETE program), and by the Portuguese Government through FCT grants SENODS CMU-PT/SAI/0045/2009, Serv-CPS PTDC/EEA-AUT/122362/2010 and CodeStream PTDC/EEI-TEL/3006/2012.
**XtratuM hypervisor for mixed-criticality systems. Configuration and deployment**

M. Masmano, J. Coronel  
FentISS  
Valencia, Spain  
{mmasmano, jcoronel}@fentiss.com

P. Balbastre, A. Crespo, J. Simó, S. Peiró  
Universitat Politècnica de València  
Valencia, Spain  
{pbalbastre, acrespo, jsimo, speiro}@ai2.upv.es

**Abstract**—This demonstration presents the configuration, compilation and deployment processes of applications based on XtratuM hypervisor.

**Keywords**—hypervisor, partitioned systems, real-time embedded systems

I. ABSTRACT

Embedded Real-time hypervisors has recently emerged as technology to achieve the most robust, secure and reliable real-time systems with limited overheads. Partitioned systems have been successfully used in avionics [1] and now, it has been extended to other fields of application as space, automotive and control systems.

XtratuM [2][3][4] is an open source hypervisor specifically designed for embedded real time systems. XtratuM hypervisor design is based on the principles stated in the Integrated Modular Avionics IMA approach. It was initially designed for mono-core and redesigned to support multicore. XtratuM hypervisor is used in different European project [5][6] to support mixed-criticality systems. Partitions in XtratuM are an execution environment including the application and the operating system. Several partitions with different guestOSs can be used in an application.

In this demonstration, the XtratuM approach is presented jointly with the set of tools to deploy embedded real-time systems. The demonstration includes:

1. Configuration and compilation of XtratuM

2. Configuration and development of partitions with different operation systems: bare partitions, real-time POSIX operating system and Linux.

3. Configuration of the system: A configuration file describes the scheduling of partitions, the allocation of resources to partitions and the communication channels to exchange information between partitions

4. System deployment tools to build the final system

5. Execution of the system in a platform

XtratuM implements a cyclic scheduling policy (based on ARINC-653 partitioning kernel policy). Based on this policy, XtratuM permits the execution of more critical applications (partitions) with a static allocation of temporal windows and offers the capability to handle non critical applications in specific temporal windows labeled as spare temporal windows. The schedule of these temporal windows is performed at partition level by a system partition that collects the application needs and builds dynamically the plan for them. Some services at hypervisor level permit to set and modify the internal allocation in spare temporal windows.

The demonstrator will show an application with several partitions (Linux based, real-time OS based, bare) (see figure 1) and its execution in the platform.

**Figure 1. System architecture**

XtratuM based systems are statically configured through a configuration file (XMCF) as a contract between the System Integrator and the Partition Developers. Each Partition Developer must provide to the System Integrator all resources required to run the partition successfully. From these requirements, the System Integrator elaborates the configuration file.

The demonstration will show the process of EPS configuration assisted by the configuration and scheduling tool. Figure 2. shows the development process.

**Figure 2. EPS development process**
REFERENCES


