P R O C E E D I N G S

O F T H E

5th International Workshop on

Real Time Networks RTN'06

Dresden, Germany, July 4, 2006
in conjunction with the 18th ECRTS

ECRTS
July 5-7
Dresden, Germany
2006

Jean-Dominique Decotignie (ed.)
Swiss Center for Electronics and Microtechnology
Neuchâtel, Switzerland

PROCEEDINGS

OF THE

5th International Workshop on

Real Time Networks RTN'06

Dresden, Germany, July 4, 2006
in conjunction with the 18th ECRTS

Chairman :
Jean-Dominique Decotignie
Swiss Center for Electronics and Microtechnology
Neuchâtel, Switzerland

Chairman

Jean-Dominique Decotignie,
Swiss Center for Electronics and Microtechnology, Neuchâtel, Switzerland,
jean-dominique.decotignie@csem.ch

Program Committee

Foreword

The fifth issue of the workshop on real-time networks took place in the wonderful city of Dresden. These post-conference proceedings profit from some updates of the presented papers as well as from a flavour of the discussions held. Besides the cultural aspects, we had a very active and lively workshop with 18 participants in a small room that made the interaction even closer. Discussions were quite animated and the opportunity to get a much better insight on many topics. I hope that, as a reader, you will have as fun as we had during the workshop.

I would like to thank all the persons who made this workshop a success. First the local organisers in Dresden, in particular Hermann Härtig, who attracted us in their wonderful city and who did a great job to facilitate my task. Second, all the members of the program committee who read the contributions and made the program that you have in front of you. I also would like to thank those members of the program committee who could join us for the workshop. They were a lot in the quality of the discussions. Finally, my thanks go the authors and the rapporteurs without whom this workshop would not exist.

A new issue of the workshop is under way and I hope that reading these proceedings will be an incentive to join us in Pisa.

Jean-Dominique Decotignie
Centre Suisse d’Electronique et de Microtechnique
Neuchâtel, Switzerland.

Table of Content

Session 4 – Quality of Service (Chair: Eduardo Tovar, Rapporteur: Anis Koubâa)

Session 1 – AutomotiveApplications

Chair: Christos Koulamas

Rapporteur: Jean-Dominique Decotignie

Session Summary: Automotive Applications

Rapporteur:
Jean-Dominique Decotignie
Swiss Center For Electronics and Microtechnology (CSEM)
Neuchâtel, Switzerland
Jean-dominique.decotignie@csem.ch

1. Introduction

The session had only one paper but on a very interesting topic, the gap between researchers and practionnners in the domain of real-time networking in particular in the automotive domain. This is a very important issue as networks today become the integration point between different automation components of the car.

2. The presentation

The paper is based on practical experience with engineers developing automotive applications. It explains that, despite nearly 2 decades of research papers on the automotive networks, little of the "theoretical" results is used in practice. Reasons may be found in the difference between the "clean" models used in the papers and the actual hardware employed. The same apply to software stacks deployed on the hardware that may not implement any idea of priority thus rendering void the nice properties of networks such as CAN.

The automotive industry tries to enhance the development process by defining common architectures such as AUTOSAR. The paper shows that temporal aspects are not the main concern of such initiative. Furthermore, AUTOSAR leaves too much freedom and it seems very difficult to find a temporal model that can be applied to the system. For instance, different interaction models, client server, periodic, …, may coexist and it becomes difficult in such a context to provide a clear definition of deadlines.

3. Discussion

For the presentation and the discussions, it is clear that bridging the gap between researchers and practionners is desirable but far from easy. It is also obvious that this should be the task of the research community although there is a need to change the minds in the companies. In particular, studies should address the following aspects:

- the analysis should include all the software aspects;
- this must include a model of the temporal behaviour of the ECUs (Electronic Control Units) that interact through the communication network; this may go as far as looking at the behaviour of the operating system or kernel used in the ECU;
- using the "theoretical results" should be as easy as possible and this may be related to finding the right level of abstraction;
- the model should allow to define simple things such as deadline in a common manner;
- implementing (in software) the "theoretical" models should be easy.

4. Conclusion

This presentation opened a number of possible research venues and let us hope that the subject will be at the agenda of future RTN workshops.

Applying Real-Time Network Research in the Automotive Industry: Lessons Learned and Perspectives

Dr. Kai Richter[1], Dr. Marek Jersak[1], Prof. Dr. Rolf Ernst[2]

1: Symtavision GmbH
Frankfurter Straße 3b
38122 Braunschweig, Germany
{richter|jersak}@symtavision.com

2: Institute of Computer and Communication Network Engineering
Technical University of Braunschweig
Hans-Sommer-Straße 66
38106 Braunschweig, Germany
r.ernst@tu-bs.de

Abstract

This paper addresses the still very large gap between the research community and the industry with respect to the application of real-time networks analysis. As a university spin-off providing scheduling analysis solutions, we have made several controversial experiences with the technology transfer that we would like to discuss during the workshop. Key examples from practice and a look into the industrial process of designing –and the way of thinking– shall help structuring the discussions.

1. Introduction

The area of real-time systems research including networks is a very active field for more that 30 years. Countless publications are available such as on analyzing and optimizing the timing behavior of CAN (controller-area network) communication, a widely used standard in the automotive industry. The number of contributions concerning FlexRay, often promoted as a CAN "successor", is growing, too. With the integration of more and more networked functionality in cars, network timing and performance has become a critical bottleneck in automotive architecture design, with a direct impact on design time and cost. As optimal network design and configuration requires reliable analyses and good optimizations, one could conclude that car manufacturers should be eagerly adopting technical contributions in the field of real-time networks research. However, the willingness to do so is surprisingly low. But why is that?

The reasons are multifaceted. Over the past years, we have been continuously facing that question in a number of projects in the automotive industry [1], from car manufacturers to tier-1 and software suppliers to service providers. As a university spin-off that now develops and markets the SymTA/S scheduling analysis and optimization tool suite and services, there have been interesting technical "surprises" that might appear as a key reason. In fact, there is very often a mismatch between well-defined theoretical models and the industrial practice. Additionally, practicability concerns and political, cultural, and economical reasons add to the dilemma, as they complicate convergence of both parties. This paper outlines key experiences that we have made with respect to the issues mentioned, some of them have been presented earlier [2].

2. Model Mismatch

Researchers and designers have significantly different focus. Designers have to produce something that works within a reasonable time frame. Hence, they stick to established approaches, even if it requires an enormous effort to finish the task at hand more or less on time. Quite to the contrary, researchers use their freedom to consider a variety of conceptual options to develop a consistent and well-structured theory, and then write it down. Eventually, researchers and designers are worried about the same general topic, for instance, network integration, and start talking to each other. This often reveals a different view on "the problem"; different with respect to importance, model soundness, and analyzability. We will briefly outline two illustrative examples of such "surprises".

a. CAN Example

The first example is the use of queuing strategies in CAN networks. The medium access in CAN is based on a priority-scheme. CAN frames that compete for the bus are scheduled according to their priority, coded in the CAN Id. Waiting frames on an ECU (electronic control unit) are buffered to be sent later. To no surprise, the big majority of formal methods to analyzing such systems assume that the buffering strictly follows the priority-driven strategy of CAN, as this is (!) consistent with the protocol itself. However, CAN implementations contain several software and hardware buffers. Each uses its own access strategy, including FIFO instead of priority-ordered queuing. FIFOs undermine the CAN protocols inherent access strategy and thus the available analysis techniques. Figure 1 allows comparison between two such schedules. Priority-queuing on the left leads to schedules that can be analyzed with a static-priority analysis technique, while FIFO queuing significantly complicates timing behavior and reduces analyzability.

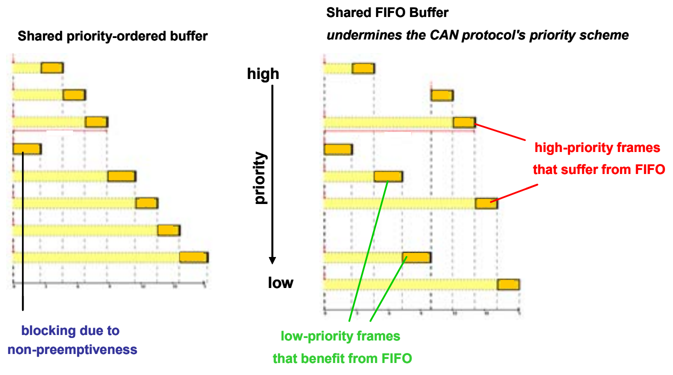


Figure 1 Effects of Priority and FIFO queuing

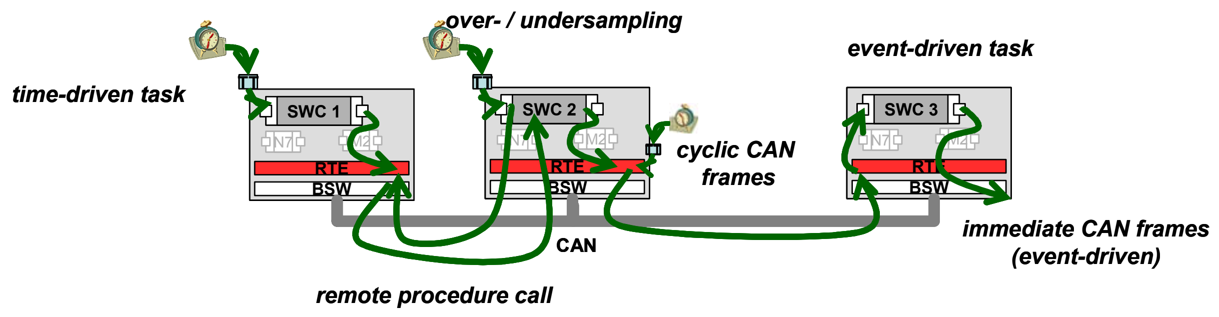


Figure 2 Causality Chains in Automotive Implementations

Other real-world mechanisms further complicate analysis. In particular, so called "overload management mechanisms" skip frames that wait in the buffers "too long", thus violating another assumption (fixed load) of typical scheduling analyses.

It is amazing to observe what designers do for lack of reliable analysis. Often significantly more messages than actually required are sent. The assumption is that allowing "N out of M" messages to get lost is a way to "guarantee" that a minimum number of messages get through. Obviously this increases bus load in the typical case and is thus counter-productive to the desire to reduce bus load to make room for more messages required for novel vehicle functions.

How should we approach this discrepancy between analysis and the real-world? The researcher might say: "This is not analyzable! Go redesign your system and come back to me!" The designer might say: "If you do not develop an analysis for exactly my problem, your kind of research is useless!" It is clear that neither party has any benefit of insisting on his/her position.

b. AUTOSAR Example

The second example illustrates another important type of model mismatch at a higher level of communication. With the increasing distribution of functions over several ECUs in a car, the importance of end-to-end timing (and deadlines) is also increasing. Industrial standardization efforts such as AUTOSAR have already defined models for capturing such "timing chains" composed of communicating "software components", illustrated in Figure 3.

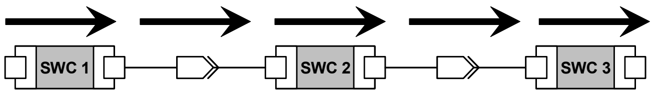


Figure 3 AUTOSAR View on "Timing Chains"

Similar models are known from data-flow theory, where clear semantics relate the execution of nodes (here: software components) with timing behavior of the stream. "Surprisingly" though AUTOSAR has not yet defined such relations. Quite to the contrary, the actual timing of software components is mostly left open. Additionally, there exist several valid communication semantics including client-server (remote procedure call), periodic sampling including under- and over-sampling, polling, and event-driven. This leads to a variety of possible "causality chains" in the actual implementation that can be subject to analysis. Figure 2 shows examples for these causality chains through the layered software defined by AUTOSAR.

What does "end-to-end timing" mean in absence of semantic definitions? Again, the lack of a "common ground" leads to a mismatch between the work of researchers and the challenges system designers face, and both proceed in isolation.

3. AUTOSAR Background

It is important to understand that the primary goal of AUTOSAR is not to solve timing problems in particular. AUTOSAR rather defines a software infrastructure for application and basic software, illustrated in Figure 4. The goal is to be able to exchange parts of the system's software without rebuilding everything. This shall enable modularity, scalability, transferability and re-usability of software among projects, variants, suppliers, customers, etc.. Hence, timing is not in the center of AUTOSAR but has later been recognized as an "important issue" that requires further consideration.

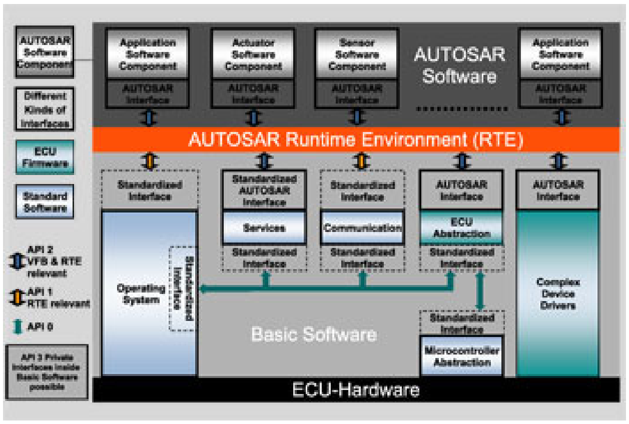

AUTOSAR Software Component
Different Kinds of Interfaces
ECU Firmware
Standard Software
API 2 VFB & RTE relevant
API 1 RTE relevant
API 0
API 3 Private Interfaces inside Basic Software possible
Application Software Component
Actuator Software Component
Sensor Software Component
AUTOSAR Software
Application Software Component
AUTOSAR Interface
AUTOSAR Runtime Environment (RTE)
Standardized Interface
Standardized AUTOSAR Interface
Services
Communication
ECU Abstraction
Operating System
Standardized Interface
Complex Device Drivers
Microcontroller Abstraction
Basic Software
ECU-Hardware

Figure 4 Standardized AUTOSAR Software

Although the AUTOSAR specification is not open to the public, the website www.autosar.org provides a quick overview and few papers [3]. However, AUTOSAR borrows many key concepts from the OSEK/VDX standard that is available through www.osek-vdx.org. These documents define a layered software-architecture with many APIs in many configurations but only few semantics. With respect to example b), especially the communication layer with its various configuration options such as "triggered vs. pending signals" that are sent through "periodic, direct, or mixed frames" [4] is a major source of network and system-level timing complexity. The standard further lets open the implementation of the lower-level drivers. We have seen in example a) that queuing strategies in particular can make the difference. Finally, also the OSEK OS [5] standard knows of several task types and activation mechanisms, complicating the analysis of schedules and timing chains (example b) further.

4. Practical Issues

In addition to a common technical ground, designers also need to be able to embed a researched technology into their everyday design-flow. Based on the feedback we have been receiving from a variety of designers, this in particular requires:

1. generating or obtaining the data needed for analysis (be it by definition, measurement, test, or simply asking the right people)
2. having a specific strategy when and how to apply the technology
3. interpreting the results and consequently taking decisions.

All this in a reasonable amount of time, after a reasonable amount of training on that technology.

If the technology appears too complex, designers will ignore it. If input data is not readily available, they can't use it, and if using the technology takes longer than finding a sub-optimal manual solution, it will be considered useless, again.

We highlight this, as researchers (rightfully) tend to do work that is "elegant" or "systematic" in itself without paying too much attention to practical issues.

5. Supply-Chain Issues

Specifically car manufacturers nowadays have to cope with an increasing number of network real-time problems that are fully new to them. Historically, car manufacturers designed mechanical parts. The electronics parts including the software were, for a long time and still, supplied externally. Hence, the OEMs became used to their suppliers solving the technical problems related to software.

Now, the OEMs still do not develop large parts of the software. However, as a result of function distribution, the network turns into the center of many integration efforts for which the OEM is responsible. As illustrated in the CAN example, the network timing depends not only on the protocol but also on driver hardware and software. And even though the OEM controls many network parameters such as topology, speed, and frame priorities, the drivers are often not in the OEM's area of responsibility.

6. Possible Solutions

The supply-chain communication between OEMs and suppliers will have to evolve. As ECU implementation possibly affects network timing, relevant data may have to be disclosed by the suppliers. From the other perspective, OEMs could impose ECU timing requirement on their suppliers that they know will satisfy assumptions on the timing of the communication infrastructure.

This leads to the idea of establishing timing contracts between OEMs and suppliers for each "module" or "component" that is designed individually but contributes to the overall system timing.

Finding a right strategy is difficult. In order to be accepted

- Responsibilities and scope must be clearly defined, and must (more or less) match the established roles of suppliers and OEMs.
- IP protection must be ensured, in particular on the supplier's side. Together with already existing standards such as AUTOSAR, this will have a dominant impact on the structure of the analytical model.
- A suitable timing analysis methodology must be in place. Based on the structure just mentioned, the analytical possibilities will to a large part define the parameters of the model, since there is no point in modeling something that cannot be analyzed.
- It must be clarified what kind of analysis results and what level of accuracy can be obtained at a particular design stage, and the required effort.
- Any analysis methodology must allow engineers to reason about their decisions systematically. 100% accuracy may not be needed if only the results are significantly better than "gut feeling".

An important step is to further standardize and "homogenize", in order to reduce complexity. Today some OEMs have defined a "standard core" with predefined OS and driver-level concepts which every supplier must implement. This ensures more predictable timing of the communication infrastructure; and better configurability. AUTOSAR has helped to define standard interfaces between components at various granularities and levels of abstraction. First reference implementations show that the interfaces work.

However, the current view is still very function- and software-centric; AUTOSAR version 1.0 does not include timing, and hence does not tackle timing-related integration issues. Furthermore, the standard does not contain clear guidelines how to use the standardized technology. Therefore, it is not clear how to establish a ready-to-use analysis approach. Guidelines along the "standard core" approach may therefore be needed in order to cover a significant number of problems with a suitable timing analysis methodology.

In several projects with OEMs, tier-1 and -2 suppliers, we have seen that each particular partner is in fact capable and willing to apply a certain amount of timing analysis, if only the scope is suitable, the analysis can be performed efficiently, and they see a real value for them.

For instance, OS and basic software suppliers can determine the latency of service routines, driver functions, and disclose key mechanisms such as queuing strategies. Function designers can use measurements or formal analysis to obtain execution times of their functions, along with amount of communicated data. Similarly, ECU suppliers can do more precise and systematic measurements to generate the data required for thorough scheduling analysis, and they can build timing interfaces to the bus, specifically with respect to dynamic driver interrupts. This is already a large step towards supplier-OEM timing contracts. Finally, OEMs have started to use their knowledge about the "standard core" to gather information about key queuing mechanisms used in their systems. From the knowledge about these mechanisms, together with the software supplier's data and the dynamic ECU-network timing interface, we have established and solved scheduling models that particularly support OEMs in comparing the performance and robustness of several configurations, without requiring any of them to understand the full picture.

7. SymTA/S Review

Our own technology, SymTA/S, has been originally developed at the Institute of Computer and Communication Network Engineering [6] and is based on the idea of compositional scheduling analysis. In contrast to the holistic approaches, SymTA/S allows direct re-use of the host of existing single-processor scheduling techniques such as RMA/DMA, EFD, TDMA, RR, etc.. Details are not individually cited here but can be found in a SymTA/S overview paper [7].

SymTA/S captures system-level dependencies through event models at the interfaces between locally analyzable components. This gives structure to the model and protects IP internal to the components, be it software components (tasks), ECUs (CPU resources) or buses. Furthermore, we have developed configurable analysis libraries tailored to the concepts defined by OSEK, AUTOSAR, CAN, and we are currently working on a compliant library for FlexRay.

By keeping the first-class citizens of the analytical model small and in line with the established industry system view, the involved parties can in fact establish timing contracts that they can oversee. And the compositional approach enables establishing a system-level analysis from such black boxes.

Furthermore, using a "tool box" of technologies from real-time systems research rather than a single approach, allows quick extension and customization of the analysis, a prerequisite for meeting key requirements mentioned in Section 6. We have successfully applied SymTA/S in several industry projects with customers [1, 8], and are constantly extending it with academic partners.

The "pure" analysis is supplemented by a set of productivity plug-ins. An exploration module [10] uses genetic algorithms to find optimized system configurations. Sensitivity analysis [11] is used to detect and avoid critical hot spots in the design. Finally, the technology has also been used in a multi-supplier risk management system [12].

8. Summary

The gap between research and industry is still large in the area of real-time networks. Two key examples have shown that technical barriers are only one reason. Practicability issues, supply-chain communications and other strategic or even political decisions are other reasons. In this paper, we have outlined a set of requirements and possible solutions. We have further seen that we could already apply some of them successfully in practice using our SymTA/S tool suite. Key to this is that all involved parties must approach each other within a bounded scope of technical problems and clear goals. We ultimately believe that, after some time, designers will themselves distinguish a technically sound (and elegant) solution from a less systematic one. They will do it to reasons of analyzability and safety rather than elegance. However, any such successful cooperation between industry and research, at best with evident benefits, helps fostering the appreciation of real-time networks research.

This step-wise approach still requires a suitable methodology, along with models, which have to be developed. This also includes re-thinking the roles of OEMs and suppliers and their communication along the supply chain, possibly leading to an engineering evolution for individual partners, and a cultural change in a new multi-supplier design process management.

References

[1] Kai Richter, Rolf Ernst. Real-Time Analysis as a Quality Feature: Automotive Use-Cases and Applications. In *Embedded World Conference*, Nürnberg, Germany, February 2006.

[2] Kai Richter, The AUTOSAR Timing Model—Status and Challenges, *ARTIST2 Workshop "Beyond AUTOSAR"*, Innsbruck, 2006

[3] AUTOSAR Partnership. AUTOSAR – Current results and preparations for exploitation. *7th EUROFORUM conference Software in the vehicle*. 3-4 May 2006, Stuttgart, Germany

[4] OSEK/VDX Communication. v.3.0.3, OSEK/VDX Consortium, July 2004

[5] OSEK/VDX Operating System. V.2.2.3, OSEK/VDX Consortium, February 2005

[6] SymTA/S Project. Institute of Computer and Communication Network Engineering, Technical University of Braunschweig, Germany, www.symta.org

[7] Rafik Henia, Arne Hamann, Marek Jersak, Razvan Racu, Kai Richter, Rolf Ernst. System Level Performance Analysis - the SymTA/S Approach. *IEE Proceedings Computers and Digital Techniques*, 2005.

[8] Kai Richter, Marek Jersak, Rolf Ernst. How OEMs and Suppliers can tackle the Network Integration Challenges. In *Proc. Embedded Real-Time Software Congress (ERTS)*, Toulouse, France, January 2006.

[9] Arne Hamann, Marek Jersak, Kai Richter, Rolf Ernst. A framework for modular analysis and exploration of heterogeneous embedded systems. *Real-Time Systems*, volume 33, pages 101-137, July 2006.

[10] R. Racu, M. Jersak, and R. Ernst. Applying sensitivity analysis in real-time distributed systems. *In 11th IEEE Real-Time Technology and Applications Symposium (RTAS)*, San Francisco, USA, 2005.

[11] J. Kruse, T. Volling, C. Thomsen, R. Ernst, and T. Spengler. Towards Flexible Systems Engineering by Using Flexible Quantity Contracts. In *Proc. Automation, Assistance and Embedded Real Time Platforms for Transportation (AAET*), Braunschweig, Germany, 2005.

Session 2 - Old Problems revisited

Chair: Luis Almeida

Rapporteur: Björn Andersson

Session Summary: Old Problems Revisited

Chair : Luis Almeida
DET/IEETA, Universidade de Aveiro
Aveiro, Portugal
lda@det.ua.pt

Rapporteur: Björn Andersson
DEI-ISEP, Polytechnic Institute of Porto
Porto, Portugal
bandersson@dei.isep.ipp.pt

1. Luis Report

This was a short session with one paper dedicated to response-time analysis issues. The paper, entitled *Message response time analysis for ideal controller area network (CAN) refuted* was presented by R. Bril and co-authored by J. Lukkien, both from TU Eindhoven and R. Davis and A. Burns from the University of York. The paper basically shows that the well known analysis to deduce the worst-case response time of messages in CAN, initially presented by Ken Tindell in 1994, is optimistic in some cases. In fact, for such cases, the worst-case response time of a message does not occur when it is released synchronously with all higher priority ones. The cause seems to be the blocking that a previous instance of a given message can cause to higher priority messages leading to higher interference on the next instance of the same message. Curiously, this effect is known for many years in the context of non-preemptive task scheduling and appropriate analysis was proposed, which is based on the fact that the worst-case response time still occurs in the synchronous busy interval.

Thus, because of the large impact that Tindell's work had on the real-time analysis developed for CAN in the past 12 years, this paper was awaited with some anxiety. A lively discussion took place after the presentation trying to understand the problem, its probability of occurrence and conditions that can lead to its occurrence. It was acknowledged that the situation indicated is relatively rare, which is also confirmed by the time that it took to find it. Also, it was acknowledged that such situation is not necessarily associated with very high utilization levels. The discussion ended considering whether Tindell's analysis could be adapted, with some non-optimal parameter, e.g. extra blocking or release jitter, to cope with the found situation but, as R. Bril indicated, it does not seem likely.

.

2. Björn Report

The paper claims that the schedulability analysis published (by Ken Tindell) on the CAN bus is not a sufficient schedulability test. None of the workshop participants disagreed on that. Figure 1 in the paper shows that the highest priority task τ_1 can cause more than C_1 interference on task τ_3. A question was brought whether it is only the highest priority task that causes more interference than the previously published CAN analysis expresses and the author gave the answer that there are task sets where the two highest priority tasks cause more interference than the previously proposed analysis. It was discussed if the previous analysis is correct for certain restricted task sets; in particular one of the workshop participants asked if the CAN analysis is incorrect for low utilization; say less than 50%. For the system model used in the paper; the workshop did not give an answer. For systems with non-zero jitter, the author claimed that there are task sets with a utilization close to 0% where the CAN analysis (by Ken Tindell) is not sufficient. It was discussed whether this analysis carry over to another scheduling problems that are non-preemptive-like, for example PCP. No clear answer was given by the author or the workshop attendees but the general intuition of the workshop attendees was that the analysis of PCP remains valid.

Message response time analysis for ideal controller area network (CAN) refuted

Reinder J. Bril and Johan J. Lukkien
Technische Universiteit Eindhoven (TU/e),
Den Dolech 2, 5600 AZ Eindhoven, The Netherlands
{r.j.bril, j.j.lukkien}@tue.nl

Robert I. Davis and Alan Burns
University of York,
York, Y01 5DD, England
{rob.davis, burns}@cs.york.ac.uk

Abstract

This paper revisits basic message response time analysis of controller area network (CAN). We show that existing message response time analysis, as presented in [17], is optimistic. Assuming discrete scheduling, the problem can be resolved by applying worst-case response time analysis for fixed-priority non-preemptive scheduling (FPNS) as described in [6].

1 Introduction

Controller Area Network (CAN) is a serial, broadcast, bus for sending and receiving short real-time control messages, consisting of between 0 and 8 bytes, and has been designed to operate at speeds of up to 1 Mbit/sec. CAN was originally developed for the automotive industry [1, 7]. Currently, it is not only a widely used vehicular network, with more than 100 million CAN nodes sold in 2000 [10], but it is also used in numerous industrial applications.

Analysis of worst-case message response times for CAN has been pioneered in [17], based on the observation that scheduling messages on a CAN bus is analogous to scheduling tasks by fixed priorities. Because CAN messages are non-preemptive, the existing worst-case response time analysis for fixed-priority preemptive scheduling (FPPS) has been updated to take account of tasks being non-preemptive, i.e. resulting in worst-case response time analysis for fixed-priority non-preemptive scheduling (FPNS). The result has subsequently been applied to CAN. The analysis is well-known and has been used widely in the academic literature and in industrial practice. The analysis presented in [15, 16] is similar to the analysis of [17].

In this paper, we show that worst-case response time analysis for FPNS with arbitrary phasing and deadlines within periods, as presented in [17], is optimistic. As a result, the worst-case message response time analysis for ideal CAN is also optimistic. The response time of a message can therefore be larger than the worst-case message response time as determined by the analysis presented in [17], and an unschedulable set of messages can therefore incorrectly be considered schedulable. Assuming discrete scheduling, the problem can be resolved by applying worst-case response time analysis for FPNS as described in [6].

This paper is organized as follows. Section 2 briefly describes a real-time scheduling model for FPNS. Response time analysis for FPNS is recapitulated in Section 3. In Section 4, we present two examples that refute the analysis in [17]. Whereas the first example is primarily meant for illustration purposes, the second example is based on realistic worst-case transmission times for CAN. Section 5 recapitulates the worst-case response time analysis for FPNS under discrete scheduling as described in [6], and presents the results of that analysis for the examples of Section 4. The paper is concluded in Section 6.

2 Real-time scheduling models

This section describes a basic scheduling model for FPPS and a refined model for FPNS. Most of the definitions and assumptions of these models originate from [12].

2.1 Basic model for FPPS

We assume a single processor and a set $\mathcal{T}$ of n periodically released, independent tasks $\tau_1, \tau_2, \ldots, \tau_n$. At any moment in time, the processor is used to execute the highest priority task that has work pending.

Each task τ_i is characterized by a (*release*) *period* $T_i \in \mathbb{R}^+$, a *computation time* $C_i \in \mathbb{R}^+$, a (*relative*) *deadline* $D_i \in \mathbb{R}^+$, where $C_i \leq \min(D_i, T_i)$, and a *phasing* $\varphi_i \in \mathbb{R}$. An *activation* (or *release*) *time* is a time at which a task τ_i becomes ready for execution. A release of a task is also termed a *job*. The job of task τ_i with release time φ_i serves as a reference activation, and is referred to as job zero. The release of job k of τ_i therefore takes place at time $a_{ik} = \varphi_i + kT_i$, $k \in \mathbb{Z}$. The deadline of job k of τ_i takes place at time $d_{ik} = a_{ik} + D_i$.

The set of phasings φ_i is termed the phasing φ of the task set $\mathcal{T}$. We assume that we do not have control over the phasing φ, for instance since the tasks are released by external events, so we assume that any arbitrary phasing may occur. This assumption is common in real-time scheduling literature [8, 9, 12].

The *response interval* of job k of τ_i is defined as the time span between the activation time of that job and its completion time c_{ik}, i.e. $[a_{ik}, c_{ik})$. The *response time* r_{ik} of job k of τ_i is defined as the length of its response interval, i.e. $r_{ik} = c_{ik} - a_{ik}$. The *worst-case response time* WR_i of a task τ_i is the largest response time of any of its jobs, i.e.

$$WR_i = \sup_{\varphi,k} r_{ik}. \tag{1}$$

A *critical instant* of a task is defined as an (hypothetical) instant that leads to the worst-case response time for that task.

As well as arbitrary phasing, we also assume other standard basic assumptions [12], i.e. tasks are ready to run at the start of each period and do not suspend themselves, tasks will be preempted instantaneously when a higher priority task becomes ready to run, a job of a task does not start before its previous job is completed, and the overhead of context switching and task scheduling is ignored. Finally, we assume that the deadlines are hard, i.e. each job of a task must be completed before its deadline. Hence, a set $\mathcal{T}$ of n periodic tasks can be scheduled if and only if

$$WR_i \leq D_i \tag{2}$$

for all $i = 1, \ldots, n$.

For notational convenience, we assume that the tasks are given in order of decreasing priority, i.e. task τ_1 has the highest priority and task τ_n has the lowest priority.

2.2 Refined model for FPNS

For FPNS, we need to refine our basic model of Section 2.1. Unlike FPPS, tasks are no longer instantaneously preempted when a higher priority task becomes ready to run, but are allowed to complete their execution. As a result, the processor need not execute the highest priority task that has work pending at a particular moment in time.

3 Recapitulation of existing analysis

In this section, we recapitulate worst-case response time analysis for FPPS and worst-case message response time analysis for ideal CAN. The latter is based on worst-case response time analysis for FPNS. Because we discuss response times under both FPPS and FPNS, we will use subscripts P and N to denote FPPS and FPNS, respectively.

3.1 Worst-case response time analysis for FPPS

To determine worst-case response times under arbitrary phasing, it suffices to consider only critical instants. For FPPS, critical instants are given by time points at which all tasks have a simultaneous release [12].

From this notion of critical instants, Joseph and Pandya [8] derived that for deadlines within periods (i.e. $D_i \leq T_i$) the worst-case response time WR_i^{P} of a task τ_i is given by the smallest $x \in \mathbb{R}^+$ that satisfies

$$x = C_i + \sum_{j<i} \left\lceil \frac{x}{T_j} \right\rceil C_j. \tag{3}$$

To calculate worst-case response times, we can use an iterative procedure based on recurrence relationships [2]. The procedure starts with a lower bound.

$$\begin{aligned} wr_i^{(0)} &= \sum_{j\leq i} C_j \\ wr_i^{(k+1)} &= C_i + \sum_{j<i} \left\lceil \frac{wr_i^{(k)}}{T_j} \right\rceil C_j \end{aligned}$$

The procedure is stopped when the same value is found for two successive iterations of k or when the deadline D_i is exceeded. In the former case, it yields the smallest solution of the recursive equation, i.e. the worst-case response time of τ_i. In the latter case the task is not schedulable. Termination of the procedure is ensured by the fact that the sequence $wr_i^{(k)}$ is bounded (from below by C_i, and from above by D_i) and non-decreasing, and that different values for successive iterations differ by at least $\min_{j<i} C_j$.

The interested reader is referred to [9, 11, 14] for techniques to derive worst-case response times for tasks with arbitrary deadlines. The main difference with deadlines within periods is that for arbitrary deadlines the worst-case response time of a task is not necessarily assumed for the first job that is released at the critical instant.

3.2 Message response time analysis for CAN

In this section, we recapitulate basic message response time analysis for ideal CAN. To this end, we first present the update of [8] given in [17] to take account of tasks being non-preemptive. Next, we recapitulate how the updated analysis can be applied to CAN as described in [17]. The analysis assumes deadlines within periods (i.e. $D_i \leq T_i$).

The non-preemptive nature of tasks may cause blocking of a task by at most one lower priority task. The maximum blocking B_i of task τ_i by a lower priority task is equal to the longest computation time of a task with a priority lower than task τ_i, i.e.

$$B_i = \max_{j>i} C_j. \tag{4}$$

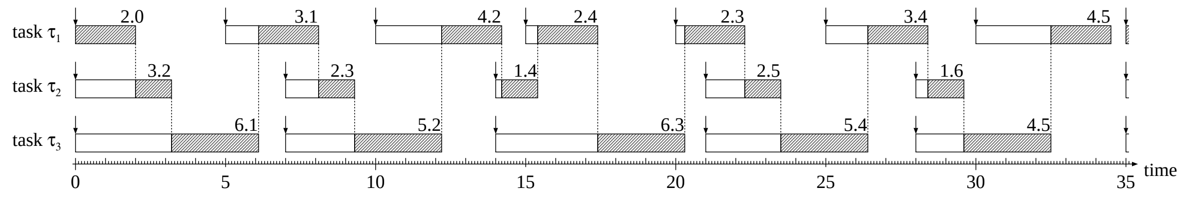


Figure 1. Timeline for $\mathcal{T}_1$ under FPNS with a simultaneous release at time zero. The numbers at the top right corner of the boxes denote the response times of the respective releases. Note that response time is counted from the moment of release up to the corresponding completion.

The worst-case response time $\widetilde{WR}_i^{\mathrm{N}}$ is given by

$$\widetilde{WR}_i^{\mathrm{N}} = w_i + C_i, \tag{5}$$

where w_i is the smallest $x \in \mathbb{R}^+$ that satisfies

$$x = B_i + \sum_{j<i} \left\lceil \frac{x + \tau_{res}}{T_j} \right\rceil C_j. \tag{6}$$

In this latter equation, τ_{res} is the resolution with which time is measured. To calculate w_i, an iterative procedure based on recurrence relationships can be used. An appropriate initial value for this procedure is $w_i^{(0)} = B_i + \sum_{j<i} C_j$.

Because scheduling messages on a CAN bus is analogous to scheduling tasks by fixed priorities, the analysis for FPNS, like the analysis given above, can be used to determine the worst-case message response time for CAN. A message μ_i has a *period* T_i, a *worst-case transmission time* C_i, and a (*relative*) *deadline* D_i. On a CAN bus, one deals with time units as multiples of the bit-time, which is denoted as τ_{bit}, i.e. $\tau_{res} = \tau_{bit}$ in Equation (6). With a 1Mbit/sec bus, τ_{bit} is equal to $1\mu s$. In this paper, we express the message characteristics T_i, C_i and D_i as multiples of τ_{bit}. Based on Version 2.0 A, standard format [1], we use for C_i

$$C_i = 47 + 8b_i + \left\lfloor \frac{34 + 8b_i - 1}{4} \right\rfloor = 55 + 10b_i \tag{7}$$

where b_i is the number of data bytes in the message (i.e. $b_i \in \{0, 1, \ldots, 8\}$), 47 is the number of control bits in a CAN frame, and 34 is the number of control bits that are subject to bit-stuffing. Bit-stuffing is required, because six consecutive bits of the same polarity (i.e. 111111 or 000000) are used for error signaling in CAN. A bit of opposite polarity is therefore inserted after five consecutive bits of the same polarity, giving rise to the floor-function and the numbers 1 and 4 in the equation.

The worst-case message response time can now be derived using Equations (4), (5), and (6). In the next section, we will show that analysis based on these equations can be optimistic.

4 Counterexamples

In this section, we give two examples that refute the existing analysis in [17]. Whereas the first example is primarily meant for illustration purposes, the second example is based on realistic worst-case transmission times for CAN.

4.1 Analysis for FPNS is optimistic

The task characteristics of our first counterexample are given in Table 1. The table includes the worst-case response times of the example as determined by means of [17] and [6]. Note that the (*processor*) *utilization factor* U of the

task	$T = D$	C	$\widetilde{WR}^{\mathrm{N}}$ ([17])	WR^{N} ([6])
τ_1	5	2	4.9	4.8
τ_2	7	1.2	6.1	6.0
τ_3	7	2.9	6.1	6.3

Table 1. Task characteristics of $\mathcal{T}_1$ and worst-case response times under FPNS.

task set $\mathcal{T}_1$ is given by $U = \frac{2}{5} + \frac{1.2}{7} + \frac{2.9}{7} \approx 0.986$.

We will now show that the worst-case response time of task τ_3 as determined by Equations (4), (5) and (6) is optimistic.

Based on Equations (6) and (4), and using $\tau_{res} = 0.1$, we derive

$$\begin{aligned} w_3^{(0)} &= B_3 + C_1 + C_2 = 0 + 2.0 + 1.2 = 3.2 \\ w_3^{(1)} &= B_3 + \sum_{j<3} \left\lceil \frac{w_3^{(0)} + \tau_{res}}{T_j} \right\rceil C_j \\ &= 0 + \left\lceil \frac{3.2 + 0.1}{5} \right\rceil \cdot 2.0 + \left\lceil \frac{3.2 + 0.1}{7.0} \right\rceil \cdot 1.2 \\ &= 3.2, \end{aligned}$$

and we find $w_3 = 3.2$. Using Equation (5), we now get $\widetilde{WR}_3^{\mathrm{N}} = 3.2 + 2.9 = 6.1$. Similarly, we find $\widetilde{WR}_1^{\mathrm{N}} = 4.9$ and $\widetilde{WR}_2^{\mathrm{N}} = 6.1$.

Figure 1 shows a timeline with the executions of the three tasks of $\mathcal{T}_1$ in an interval of length 35, i.e. equal to the *hy-*

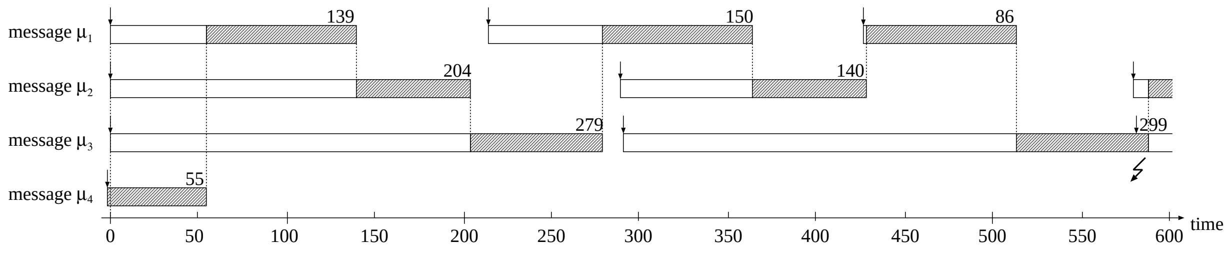


Figure 2. Timeline for $\mathcal{M}_2$ with a transmission at time 0 for μ_1, μ_2, and μ_3, and at time -1 for μ_4.

perperiod H of the tasks, which is equal to the least common multiple (lcm) of the periods. The schedule in $[0,35)$ is repeated in the intervals $[hH,(h+1)H)$ with $h \in \mathbb{Z}$, i.e. the schedule is periodic with period H. As illustrated in Figure 1, the derived value for $\widetilde{WR}_3^{\mathrm{N}}$ corresponds to the response time of the 1st job of task τ_3 upon a simultaneous release with tasks τ_1 and τ_2. However, the response time of the 3rd job of task τ_3 is equal to 6.3 in that figure, illustrating that the existing analysis is optimistic.

4.2 Existing analysis for CAN is optimistic

Table 2 presents message characteristics of a message set $\mathcal{M}_2$ with realistic worst-case transmission times for CAN, including the worst-case message response times for ideal CAN. Messages μ_1 to μ_4 contain 3, 1, 2, and 0 data bytes, respectively; see also Equation (7). Note that $\mathcal{M}_2$ has a utilization $U = \frac{85}{214} + \frac{65}{289} + \frac{75}{290} + \frac{55}{3000} \approx 0.90$.

message	$T = D$	C	$\widetilde{WR}^{\mathrm{N}}$ ([17])	WR^{N} ([6])
μ_1	214	85	160	159
μ_2	289	65	225	224
μ_3	290	75	280	299
μ_4	3000	55	590	590

Table 2. Message characteristics (as multiples of τ_{bit}) of $\mathcal{M}_2$ and worst-case message response times for ideal CAN.

We will now show that the worst-case response time of message μ_3 as determined by Equations (4), (5) and (6) is also optimistic.

Based on Equations (6) and (4), and using $\tau_{res} = \tau_{bit} = 1$, we derive

$$\begin{aligned} w_3^{(0)} &= B_3 + C_1 + C_2 = 55 + 85 + 65 = 205 \\ w_3^{(1)} &= B_3 + \sum_{j<3} \left\lceil \frac{w_3^{(0)} + \tau_{bit}}{T_j} \right\rceil C_j \\ &= 55 + \left\lceil \frac{205+1}{214} \right\rceil \cdot 85 + \left\lceil \frac{205+1}{289} \right\rceil \cdot 65 \\ &= 205, \end{aligned}$$

and we find $w_3 = 205$. Using Equation (5), we now get $\widetilde{WR}_3^{\mathrm{N}} = 205 + 75 = 280$. Similarly, we find $\widetilde{WR}_1^{\mathrm{N}} = 160$, $\widetilde{WR}_2^{\mathrm{N}} = 225$, and $\widetilde{WR}_4^{\mathrm{N}} = 590$. Hence, according to the existing analysis the set of messages is schedulable on a CAN bus.

Figure 2 shows a timeline with a transmission at time $t = 0$ for messages μ_1, μ_2, and μ_3, and at time $t = -1$ for message μ_4. As illustrated in Figure 2, the 2nd transmission of message μ_3 has a response time of 299. This value is not only larger than the derived value for $\widetilde{WR}_3^{\mathrm{N}} = 280$, but also larger than the deadline $D_3 = 290$. Hence, although the set of messages is deemed schedulable according to the existing analysis, it is actually unschedulable. The existing analysis is therefore also optimistic for the example given in Table 2.

4.3 Cause of optimism in existing analysis

Above, we have shown that even when deadlines are within periods, we cannot restrict ourselves to the response time of a single job of a task when determining the worst-case response time of that task under FPNS. The reason for this is that a job of task τ_i can defer the execution of higher priority tasks, which can potentially give rise to higher interference for subsequent jobs of task τ_i. This is illustrated in Figure 1, amongst others. The 1st job of task τ_3 experiences an interference of 3.2, corresponding to the sum of the computation times of tasks τ_1 and τ_2. The 3rd job of τ_3 experiences an additional interference of 0.2 because the 3rd job of τ_1 is deferred by the 2nd job of τ_3.

We observe that the origin of the problem is basically the same as described in [4] for the problem with existing analysis for worst-case response times for fixed-priority scheduling with deferred preemption (FPDS) with arbitrary phasing and deadlines within periods. A similar issue with work on preemption thresholds [18] was first identified and corrected by Regehr [13] in 2002.

5 CAN analysis based on discrete scheduling

In [6], worst-case response time analysis is presented for FPNS with *arbitrary* deadlines, arbitrary phasing, and

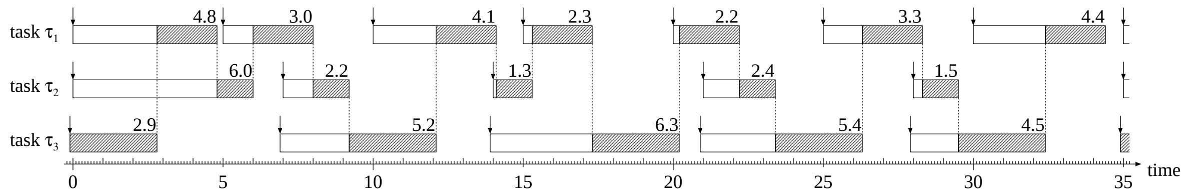


Figure 3. Timeline for $\mathcal{T}_1$ under FPNS with a release at time 0 for τ_1 and τ_2, and at time -0.1 for τ_3.

discrete (rather than continuous) scheduling [3]. For discrete scheduling, all task parameters are restricted to integers, and tasks are scheduled at integer times. Assuming discrete scheduling for CAN, the problem with the existing analysis can be resolved by applying the analysis for FPNS as described in [6]. In this section, we first recapitulate the analysis from [6]. Next, we present the results of applying the analysis to the counterexamples given in Section 4. We conclude this section with a remark about the differences between the values for $\widetilde{WR}^{\mathrm{N}}$ and WR^{N}.

5.1 Analysis for FPNS for discrete scheduling

To recapitulate the worst-case response time analysis as presented for FPNS in [6], Lemma 6 and Theorem 15 of that report are given below, with minor modifications to match our terminology and scheduling model. The lemma describes a critical instant for task τ_i.

Lemma 1 *The worst-case response time of τ_i is found in a level-i busy period by releasing all tasks τ_j with $j \le i$ simultaneously at time $t = 0$, and by releasing the longest task τ_k with $k > i$, if any, at time $t = -1$.*

Theorem 1 *Given a task set $\mathcal{T}$ consisting of n tasks $\tau_1, \ldots, \tau_n$, the worst-case response time of any task τ_i is given by*

$$WR_i^{\mathrm{N}} = \max_{q=0,\ldots,Q} \{w_{i,q} + C_i - qT_i\}, \tag{8}$$

where

$$w_{i,q} = qC_i + \sum_{j<i} \left(1 + \left\lfloor \frac{w_{i,q}}{T_j} \right\rfloor \right) C_j + \max_{k>i}\{C_k - 1\}, \tag{9}$$

and $Q = \left\lfloor \frac{L_i}{T_i} \right\rfloor$, where L_i is the length of the longest level-i busy period in non-preemptive context, which is given by the smallest positive integer l satisfying the following equation

$$l = \max_{j>i}\{C_j - 1\} + \sum_{j\le i} \left\lceil \frac{l}{T_j} \right\rceil C_j. \tag{10}$$

We observe that equation $Q = \left\lfloor \frac{L_i}{T_i} \right\rfloor$ in Theorem 1 yields a value that is one too large when the length L_i of the longest level-i busy period is an integer multiple of the period T_i. This can be easily resolved by using the equation $Q = \left\lceil \frac{L_i}{T_i} \right\rceil - 1$ instead. Although the existing equation does not give rise to problems, i.e. Equation (9) is just evaluated one extra, we prefer this more efficient formulation.

5.2 Counterexamples revisited

The worst-case response times WR^{N} of the tasks of $\mathcal{T}_1$ as determined by the analysis of [6] are also included in Table 1. In order to make the analysis applicable, we first multiplied all task parameters with 10, subsequently performed the analysis, and finally divided the resulting worst-case response times by 10. Based on Lemma 1, we conclude that the worst-case response times of tasks τ_1 and τ_2 are illustrated in Figure 3, and of task τ_3 in Figure 1.

Similarly, Table 2 includes the worst-case message response times WR^{N} of the messages of $\mathcal{M}_2$. Based on Lemma 1, we conclude that the worst-case message response time WR_3^{N} of message μ_3 is illustrated in Figure 2.

5.3 Concluding remarks

Considering Tables 1 and 2, it is remarkable that the values for $\widetilde{WR}^{\mathrm{N}}$ and WR^{N} are different for all but the lowest priority message μ_4. The optimism in $\widetilde{WR}_3^{\mathrm{N}}$ for task τ_3 in Table 1 and message μ_3 in Table 2 has already been explained in Section 4.3. This section deals with the differences in the other values.

We observe that the characteristics of the tasks and messages of both our counterexamples are integral multiples of a value $\delta \ge \tau_{res}$. As a consequence, reducing τ_{res} to an arbitrary small positive value does not change the values for $\widetilde{WR}^{\mathrm{N}}$ in either Table 1 or Table 2. Moreover, using τ_{res} and a ceiling function in Equation (6) therefore also has the same effect for our counterexamples as using a floor function and an addition term $+1$ in Equation (9). Hence, the differences are not caused by the usage of τ_{res}. Instead, the cause of the differences is found in the values used for the maximum blocking, i.e. Equation (9) includes an additional term -1 when compared to Equation (4). Note that $\widetilde{WR}_4^{\mathrm{N}} = WR_4^{\mathrm{N}}$ in Table 2 because the maximum blocking is, in both cases, equal to zero for the lowest priority message.

6 Conclusion

In this document, we revisited basic worst-case message response times for ideal controller area network (CAN). We showed by means of two examples with a high load (i.e. of $\approx$ 99% and $\approx$ 90%) that the analysis as presented in [17] is optimistic. Assuming discrete scheduling, the problem can be resolved by applying the analysis for FPNS presented in [6].

We are currently investigating how the optimism scales, i.e. whether or not the existing analysis can result in optimistic results for *any* task (or message) given an arbitrary number of tasks (or messages). We are also investigating whether or not optimistic results can occur for task(or message) sets with low utilization. Worst-case response time analysis under FPNS for continuous scheduling is a topic of future work.

Acknowledgements

We thank M.A. (Alina) Weffers-Albu and P.H.F.M. (Richard) Verhoeven from the TU/e for their comments on an earlier version of [5], a document on which this paper is based.

References

[1] *CAN specification.* Technical Report Version 2.0, Bosch, Postfach 50, D-700 Stuttgart 1, September 1991.

[2] N. Audsley, A. Burns, M. Richardson, and A. Wellings. Hard real-time scheduling: The deadline monotonic approach. In *Proc.* 8th *IEEE Workshop on Real-Time Operating Systems and Software (RTOSS)*, pp. 133–137, May 1991.

[3] S. Baruah, L. Rosier, and R. Howell. Algorithms and complexity concerning the preemptive scheduling of periodic, real-time tasks on one processor. *Real-Time Systems*, 2:301–324, 1990.

[4] R. Bril. *Existing worst-case response time analysis of real-time tasks under fixed-priority scheduling with deferred preemption is too optimistic.* CS-Report 06-05, Department of Mathematics and Computer Science, Technische Universiteit Eindhoven (TU/e), The Netherlands, February 2006.

[5] R. Bril, J. Lukkien, R. Davis, and A. Burns. *Message response time analysis for ideal controller area network (CAN) refuted.* CS-Report 06-19, Department of Mathematics and Computer Science, Technische Universiteit Eindhoven (TU/e), The Netherlands, May 2006.

[6] L. George, N. Rivierre, and M. Spuri. *Preemptive and non-preemptive real-time uni-processor scheduling.* Technical Report 2966, Institut National de Recherche et Informatique et en Automatique (INRIA), France, September 1996.

[7] ISO. *ISO 11898 road vehicles - Interchange of Digital Information - Controller Area Network (CAN) for high speed communication.* Technical report, International Standards Organization, November 1993.

[8] M. Joseph and P. Pandya. Finding response times in a real-time system. *The Computer Journal*, 29(5):390–395, 1986.

[9] M. Klein, T. Ralya, B. Pollak, R. Obenza, and M. González-Harbour. *A Practitioner's Handbook for Real-Time Analysis: Guide to Rate Monotonic Analysis for Real-Time Systems*. Kluwer Academic Publishers, 1993.

[10] G. Lee and D. Heffernan. Expanding automotive electronic systems. *IEEE Computer*, 35(1):88–93, January 2002.

[11] J. Lehoczky. Fixed priority scheduling of periodic task sets with arbitrary deadlines. In *Proc.* 11th *IEEE Real-Time Systems Symposium (RTSS)*, pp. 201–209, December 1990.

[12] C. Liu and J. Layland. Scheduling algorithms for multiprogramming in a real-time environment. *Journal of the ACM*, 20(1):46–61, 1973.

[13] J. Regehr. Scheduling tasks with mixed preemption relations for robustness to timing faults. In *Proc.* 23rd *IEEE Real-Time Systems Symposium (RTSS)*, pp. 315–326, December 2002.

[14] K. Tindell. *An extendible approach for analysing fixed priority hard real-time tasks.* Report YCS 189, Department of Computer Science, University of York, December 1992.

[15] K. Tindell and A. Burns. *Guaranteeing message latencies for distributed safity-critical hard real-time control networks.* Report YCS 229, Department of Computer Science, University of York, June 1994.

[16] K. Tindell, A. Burns, and A. Wellings. Calculating controller area network (CAN) message response times. *Control Engineering Practice*, 3(8):1163–1169, August 1995.

[17] K. Tindell, H. Hansson, and A. Wellings. Analysing real-time communications: Controller area network (CAN). In *Proc.* 15th *IEEE Real-Time Systems Symposium (RTSS)*, pp. 259–263, December 1994.

[18] Y. Wand and M. Saksena. Scheduling fixed-priority tasks with preemption threshold. In *Proc.* 6th *International Conference on Real-Time Computing Systems and Applications (RTCSA)*, pp. 328–335, December 1999.

Session 3 – Wireless Networks

Chair: Lucia Lo Bello

Rapporteur: Christos Koulamas

Session Summary: Wireless Networks

Rapporteur: Christos Koulamas
Industrial Systems Institute
Patras, Greece
koulamas@isi.gr

1. Introduction

The focus of the third session of the 5th international workshop on real-time networks was on the specifics of wireless networks.

Three papers were presented, covering a) an approach to dynamically calibrate the data transfer in wireless sensor networks between pushing and pulling [1], b) a usage of a prioritized MAC protocol to efficiently compute simple or complex aggregated quantities [2], and c) a proposal of practical service differentiation mechanisms in order to improve the performance of the slotted CSMA/CA operation of 802.15.4 for time-critical events [3].

Chairing this session was Lucia Lo Bello from the University of Catania in Italy.

2. Adaptive Leases in Wireless Sensor Networks

The work described in the first presentation was on the selection between a push or pull mechanism for the data transfer from a source to a sink. Each mechanism has its advantages and drawbacks for each side of the data transfer. With the objective to maximize the lifetime of the system, the authors propose a dynamic scheme, called “adaptive leases”. According to this, it is possible to negotiate the preferred mechanism dynamically at run-time, depending on the current state of the source and sink in order to save power at the critical side.

The authors define specific lease strategies for each possible major objective and they encapsulate these in a special network layer. The definition of a layer hides the internals of the “adaptive leases” scheme from the applications, which, in turn, access the data transfer functionality through a simple API without the need to know whether the actual transfer will be completed in a push or a pull way. In the simulations performed, for an 802.15.4 based network and for a specific example, the results show an increase in the system lifetime by a factor of 2.

The discussion triggered by the questions after the presentation, was mainly on the relationship of the 802.15.4 internals with the proposed scheme, especially, on how this scheme may be combined with the power management mechanisms of the 802.15.4 standard and compared with the results of the usage of a PAN coordinator. The presenter clarified that their proposal fits better on a peer-to-peer network topology (i.e. ad-hoc). After this, there was a proposal for a study of the presented mechanisms over other MAC protocols which may be more suitable in event driven architectures. Furthermore, the issue of scaling was discussed where it was made clear that the simulation studies were not focused on large scale wireless sensor networks but on the validation of the energy savings due to the “adaptive leases” scheme in smaller device communities.

3. Using a Prioritized MAC Protocol to Efficiently Compute Aggregated Quantities

The second presentation dealt with the problem of computing aggregated quantities over a number of multiple values coming from multiple individual sensor readings in a wireless sensor network.

The authors propose an innovative algorithm for the computation of simple or complex aggregated quantities, by utilizing the special characteristics of a prioritized MAC protocol for the wireless medium access. Key aspects of such a protocol are a) the availability of a very large range of priority levels, b) the guarantee that it is collision-free, if priorities are unique, and c) the assumption that it is a dominance protocol, operating in a way similar to the CAN bus, but in a wireless medium.

The proposed algorithm makes feasible to compute simple aggregated quantities, as the minimum and maximum of a set of proposed values, with a time complexity which does not depend on the number of nodes, and which only increases very slowly as the possible range of the value increases. Moreover, the computation of more complex quantities, as the median of the set, can be also achieved with an independent to the number of nodes time complexity, but trading off accuracy, since the solution is based on an estimation of the actual number of nodes in the network.

The related discussions evolved in two main axes. The first one was on what happens in the case of a network with a small number of nodes, considering the computation of a complex quantity, like the median, which is actually an estimation. The remark was that the estimation error increases when the number of nodes decreases, since the estimation is based on the intuition that if there is a large number of nodes which propose a randomly generated number, then the minimum random number will be very small.

The second axis of the discussion was triggered by practical issues related to the realization of the proposed prioritized MAC, in order for the proposed solution to bring a real advantage. The facts are that with the currently available technology, the prolonged duration of the transmission of the priority field may not always lead to better absolute time figures for the distributed computation, compared to the usage of a standard MAC and the usage of a typical algorithm which uses '*m*' messages for a network with '*m*' nodes and for typical values of '*m*'.

4. Improving the IEEE 802.15.4 Slotted CSMA/CA MAC for Time-Critical Events in Wireless Sensor Networks

The third and last paper presented in the Wireless Networks session contributes towards the provision of service differentiation mechanisms to the 802.15.4 protocol, in order to improve its performance for time-critical events.

The authors suggest the proposed differentiation mechanisms to be based on the provision of different sets of values for selected parameters of the medium access algorithm for each differentiated service class. They propose a two-priority scheme, with the low priority assigned to normal data frames and the high priority to time-critical information exchange, such as GTS allocation requests, alarms, PAN management commands etc. The realization of the two priority classes is based on different values for the contention window (CW), the minimum backoff exponent (macMinBE) and the maximum backoff exponent (aMaxBE) parameters of the 802.15.4 MAC operation.

The results extracted from the simulation of four different scenarios, in both a FIFO and a priority based queuing show the efficient differentiation on the selected performance metrics for the two service classes, in all scenarios referring to a fully connected network; an achievement which comes with no significant changes but only minor add-ons to the current version of the 802.15.4 standard.

During the discussions on the presented results, clarifications were given on the relationship and usage of the changes in the sensing and receiving sensitivity as a way to produce a partially connected network and study the effects of the hidden node problem. A case where the service differentiation is actually degraded but a case also were the selection of priority queuing combined with large values for the BE range parameters leads to a more energy efficient operation. Moreover, similarities in the approach have been noticed to exist in other research efforts, dealing with QoS provision to the WLAN standards, prior to 802.11e, which could be valuable sources for a qualitative cross evaluation of the results or for leading to more ideas applicable also to the 802.15.4.

5. Concluding Remarks

The quality of all presentations fed a number of equivalently interesting discussions on various energy and real-time performance issues in wireless networks and their applications. It is noticeable though that all three papers focused on specific aspects of wireless sensor networks and their related protocols, even if the session title did not put such a constraint; an indication of the increasing importance of the area and of the growing attention it receives from the research community.

References

[1] Robert van Herk, Willem Fontijn, "Adaptive Leases in Wireless Sensor Networks". *Proceedings of the 5th International Workshop on Real-Time Networks (RTN'06) in conjunction with the 18th Euromicro International Conference on Real-Time Systems (ECRTS'06)*, Dresden, Germany, July 2006.

[2] Bjorn Andersson, Nuno Pereira and Eduardo Tovar, "Using a Prioritized MAC Protocols to Efficiently Compute Aggregated Quantities". *Proceedings of the 5th International Workshop on Real-Time Networks (RTN'06) in conjunction with the 18th Euromicro International Conference on Real-Time Systems (ECRTS'06)*, Dresden, Germany, July 2006.

[3] Anis Koubaa, Mario Alvez, Bilel Nefzi and Ye-Qiong Song, "Improving the IEEE 802.15.4 Slotted CSMA/CA MAC for the Time-Critical Events in Wireless Sensor Networks". *Proceedings of the 5th International Workshop on Real-Time Networks (RTN'06) in conjunction with the 18th Euromicro International Conference on Real-Time Systems (ECRTS'06)*, Dresden, Germany, July 2006.

Adaptive Leases in Wireless Sensor Networks

Robert van Herk
Philips Research Eindhoven
robert.van.herk@philips.com

Willem Fontijn
Philips Research Eindhoven
willem.fontijn@philips.com

Abstract

A basic question in network communication is whether to push or pull data. Sometimes it can not be decided beforehand which mechanism to use. This article discusses an approach that makes it possible to dynamically calibrate between pushing and pulling in wireless sensor networks. This approach is called 'adaptive leases'. It allows for the sources and sinks of sensor data to decide for each communicated value at run-time whether pulling or pushing will be used. Doing so, nodes can dynamically optimize for power, memory, and bandwidth usage.

1 Introduction

We can divide the mechanisms to send data over a network into two categories: push and pull.

In a pushing network, a data *source* sends updates to its *sinks*. Usually the sources keep an administration on which sinks are interested in what data. Another solution is to broadcast all changes over the network, although in multi-hop networks the corresponding network load may be unacceptable.

In a pulling network, the burden of getting fresh data is put upon the sinks. The sink usually polls the source at a certain interval so that it is notified of changes when it is ready to receive them.

Table 1 describes some typical features. Beneficial features are marked with a plus sign, and disadvantages with a minus sign.

This article describes a solution in which, for each communicated value, the sources and sinks can negotiate and calibrate between pushing and pulling dynamically. We call this approach *adaptive leases*. The approach is most applicable in truly wireless sensor networks, i.e. networks with only battery operated nodes. One may think of e.g. wearable and in-body applications, for example in the medical domain.

Push
+ Source does not need to listen for incoming requests
- Sink must listen continuously to handle incoming updates
- Source must remember which sinks are interested in its data
+ Sink can be stateless
+ Updates are sent to the sinks immediately
- For every event there will be a message even if the sink is not interested at that moment
Pull
- Source must listen continuously to accept incoming pull requests
+ Sink does not need to listen when it is not interested in updates
+ Source can be stateless
- Sink must remember where to get which data
- Updates are sent to a sink only after it polls for fresh data
+ There will be no message when the sink is not interested at that moment.

Table 1. Push vs. pull

2 Related work

Similar approaches are already used in TCP/IP networks. For World Wide Web caches, Duvvuri et al. have described a system that uses adaptive leases [3]. A similar system is used by Microsoft to synchronize data between mobile devices and a computer [7]. It has been shown that smart mechanisms that combine pulling with pushing behavior in some general applications outperforms pure push [4].

There are algorithms designed to work with so-called *standing queries*. For example, in the APTEEN algorithm [6] the sink sends in its request to the source under what conditions new data is desired. The source then sends an update only if this condition is met. Adaptive leases can be used well together with this approach, as will be shown in Section 7.

Also, TinyDB has an standing query mechanism [5].

This is somewhat similar to our approach, where the lease time is the time window. However, in TinyDB the source has no mechanism to negotiate whether to use a lease or not. It can only refuse to execute the query, terminate it prematurely or comply. With adaptive leases the sink can indicate why it wants or does not want a lease, and will be informed about the actual window the source will honor. The application can then anticipate this. Another important difference is that in adaptive leases, a separate adaptive leases layer hides the leases mechanism from the application programmer (see Section 6.2).

3 Adaptive Leases in WWW caches

The system Duvvuri et al. describe [3] works as follows: When a WWW cache decides to store a copy of a web page, it sends a special message to the web server requesting a *lease* on that web page.

A lease can be thought of as a contract where the source (in this case a web server) has the obligation of pushing fresh data to the sink (in this case a web cache) for the duration of that contract. This duration is called the lease period. Once the contract will terminate, the sink must send a lease renewal request to the source. In the system of Duvvuri et al., the source decides on the lease periods. It may also decide that it does not want to engage in a requested lease at all, in which case the lease is denied.

This mechanism allows the web server to dynamically calibrate between pushing and pulling. For example, a web server with lots of outstanding leases may decide to use short lease times in new leases. In [3], this is called "Lease Duration Based on the State Space Overhead". In other cases, the web server may decide to grant a lease with a long lease period, so that the amount of control messages regarding the leases is minimized. This is called "Lease Duration Based on Control Message Overhead".

The shorter the lease times, the more the network will work as a pull system. When a sink gets responses with lease durations of zero, that value is effectively pulled by the sink, yielding a control message for each request. When the lease time is infinite, the value is effectively pushed by the source. But intermediate values are of course also possible; as such adaptive leases allow each sink and source to calibrate for each value they share between pushing and pulling.

4 Examples

As mentioned, we consider a topology in which resource constrained, battery operated sensor nodes are interconnected. In this topology, no mains-powered devices exist that have no power considerations and can afford to stay on constantly to relay messages.

4.1 Saving power at sink

We consider a patient temperature monitoring system. It consists of a thermometer and an actuator that shows the patient's temperature. Both nodes are carried with the patient and hence are battery operated.

In this example, the calibration between pushing and pulling is used to trade accuracy of the data at the sink against power usage at the sink. In order to show the current temperature, pushing is best used: the thermometer pushes a fresh value to the actuator immediately when the temperature changes. In order to achieve this, the thermometer grants the actuator a lease for the data. In a real-time system, this implies that the actuator must stay awake to deal with incoming updates immediately.

However, when the actuator's battery is almost depleted, a pulling system will be more appropriate, so that the device can sleep between pulls in order to save power. The actuator can then terminate the lease and pull a fresh value when the user indicates he wishes to see a fresh value, by pressing a button. The remaining time, the actuator can sleep to save power. Again, since we are dealing with a real-time system, the thermometer has to stay awake in order to deal with incoming requests immediately.

4.2 Saving power at sources

The opposite approach, i.e. granting leases, can be used to save power at the source.

Consider an application similar as the one explained above. The actuator may indicate it wishes to pull, for example because its battery is almost depleted. However, the source may have more stringent reasons *not* to allow pulling and grant a lease anyway, as its battery condition is more severe. In such a case, adaptive leases help in stretching the life-time of the source's battery. This is especially convenient in applications in which it is easier to replace or recharge the battery in the actuator (sink) than in the source.

This may be the case if the source is placed inside a patient. In such case, we want to delay the depletion of the source's battery. You may think of a slow release medicine system with an implanted sensor to measure sugar levels in blood and an externally accessible pump that can release insulin as required. The patient can replace the pump's battery by himself, but needs help from an expert to replace the sensor's battery. If the pump's battery is almost empty, which may happen while the patient is outside, it will start pulling data from the sensor and indicate to the patient that he needs to replace the battery. However, if the source's battery is almost empty too, it will always grant a lease, as replacing the sources battery requires an expert.

Another example is an application with sensors woven into clothing for monitoring a person during sports. It has a carried read-out device that is easily recharged and multiple sensors that are more difficult to recharge. During outdoor sporting activities, the system would normally push to save power at the sources, and only switch to pull if the battery in the read-out device is getting empty to ensure the applications runs for the entire sporting activity. If the batteries of the sensors are, however, also almost depleted, they will grant a lease anyway, as replacing them is more difficult.

5 Lease strategies

If we use adaptive leases in wireless sensor networks, we can dynamically calibrate between the typical attributes pull and push systems have that are given in table 1. For wireless sensor nodes, this is especially important since wireless nodes typically have limited resources. The ability to dynamically trade memory footprint, required processing power and bandwidth usage between the sources and the sinks is beneficial. The opportunity to let battery operated nodes sleep if the batteries are running low can extend their up-time.

The sources and sinks can use the following strategies to calibrate:

- For reducing the number of unnecessary updates: sinks that receive a lot of update information but only are interested in a new value seldomly, e.g. only when the user explicitly requests it, can terminate their leases and start pulling. For reducing state overhead at a source the same approach can be used: sources that want to save memory by diminishing their administration on active leases can terminate leases and let the sinks pull. On the other hand, if we want to reduce the number of messages per update: sinks that constantly need fresh data can request leases.

- To save power at sink: if a sink wants to sleep, it should terminate all leases. Doing so, it will know not to receive updates unless it pulls for them, so it can go to sleep, only to wake up once it is interested in new data again. For the corresponding sources, it is compulsory to stay awake. Note that this strategy is used in the example given in Section 4.1.

- To save power at source: if sources want to sleep, they need to grant all leases. Once all leases are granted and if the source knows no other sinks remain that are interested but did not yet subscribe, it can safely go to sleep and only wake up to send updates. It is compulsory for the sinks of such a source to stay awake. This strategy is used in the example given in Section 4.2.

6 Our approach

Our approach is to define a network layer that negotiates, establishes and calibrates the adaptive leases, and then to give that layer such an interface that this functionality is hidden from the application programmer.

6.1 Adaptive leases layer

The new layer will do the following: when a sink is interested in a value and does not have a lease yet, it sends a message to the source[1]. The message contains:

- What value the sink is interested in.
- A value indicating whether the sink would prefer a lease, e.g.:
  - NoLease: just send the value once
  - Lease: send new values from now on. Optionally, we can include a desired lease termination condition determining for how long, or for how many updates, the lease should preferably last.
- A value containing the argumentation for indicated preference. This value is used for the negation. For example, if NoLease is indicated, the argumentation, sorted from strongest to weakest, may be an indication that:
  - ”Sink is almost out of battery power so cannot afford to stay awake”
  - “Sink is only interested at this moment”

  If Lease is indicated, the argumentation may be an indication that:
  - “Sink must have the value real-time”
  - “Sink would prefer to have the value real-time”
  - “Sink expects to need fresh values often”

When the source receive such a message, it will consider the preference and argumentation of the sink and it’s own state to decide whether to engage in a lease or not. Every message regarding an update of a value the source sends to a sink, will contain:

- The value
- Lease status, e.g.:

[1]Some discovery mechanism needs to be in place to find the address of the source; broadcasting or some central look-up system could be used. However, this is not in the scope of this article.

- NoLease: sink will have to send a new pull request if it is interested in the value again. This reply is used if the sink indicated in the request that it wants 'NoLease' and the source did not have stronger arguments to do engage in a lease. Also, this reply is used if the sink did indicate a preference for a Lease, but the source has stronger arguments not to engage. Finally, this value will be sent after the lease has expired.
- Lease: sink should not send new pull requests for this data. Instead, the source will push new values. Optionally, we can include the lease termination condition defining for how long, or for how many updates, the source expects the lease to last. This reply is used if the sink indicated in the request that it wants a 'Lease' and the source did not have stronger arguments not to engage. This reply is also used if the sink did indicate a preference for NoLease, but the source has stronger arguments to do engage in a lease.

6.2 Making leases transparent

To hide the above protocol from the application programmer, the adaptive leases layer needs to be totally transparent. To achieve this, we define an interface to operate the adaptive leases layer from the application layer using the following primitives:

- `subscribe(valueID)`
- `unsubscribe(valueID)`
- `publish(valueID, value)`: used by the sources.

Using this abstraction, the application programmer can subscribe the sinks to the values of interest in a natural way. He does not notice whether a lease is used or not: the layer should be implemented such that when the strategy prevents the sink from requesting a lease (e.g. because it is currently low on power), or when the sources refuses the lease, it automatically reverts to repeated pulling.

The desired strategies for requesting and granting leases (see Section 5) are implemented in the adaptive leases layer. One possibility is to implement the layer once, and make it configurable, so that some nodes can use the strategies to save power, whereas others can save on memory, and so on. Another possibility is to make multiple implementations, one for each strategy, and install one with a suitable strategy on each node.

7 Simulation

We have implemented the adaptive leases protocol in Java to show its feasibility. The implementation simulates SAND (Small Autonomous Networked Devices) nodes of which the application layer uses the above interface to subscribe and unsubscribe to data, and of which the adaptive leases layer uses the protocol described in this article. SAND nodes are very small wireless sensor nodes. They contain a Philips CoolFlux DSP, and a ChipCon CC2420 radio, which uses IEEE 802.15.4 [1] as communication protocol. One can attach multiple sensors and actuators to this core. We assume that each SAND node has a battery that contains about 10000 Joules when fully charged.

All cases mentioned in Section 4 are suitable for simulation, but for simplicity, we chose to simulate the following: node v is a wireless thermometer and node w an actuator that shows the temperature as measured by v. Both v and w are SAND nodes. Nodes v and w can send messages to each other.

The 802.15.4 MAC allows to use network coordinators that are essentially powered nodes that relay messages. As mentioned above, adaptive leases are most suitable for a fully wireless network, meaning that we cannot depend on other nodes to relay messages. In 802.15.4 this kind of communication is called *ad hoc communication*. In this setup, once a wireless node sends out data to another node, the other node must be listening in order to receive it. Otherwise the message will be lost.

Here we will demonstrate calculations that show the energy savings adaptive leases can accomplish in wireless sensor networks. We will compare the results of pure-pushing, pure-pulling and adaptive leases.

We assume the network to be more or less dedicated for our application. Therefore, we do not take collisions into account for traffic between v and w. If we would take collisions into account, the energy needed per message sent or received will increase [2], but since in this example most energy is spent on listening this will not significantly change the result.

7.1 Power consumption

7.1.1 Processor

In the SAND node, the CoolFlux DSP uses about 2 mW when turned on. When it is sleeping for a predetermined amount of time, it uses about 0.01 mW, running a low frequent oscillator to count down the waiting time.

Most of the time we will neglect the power used by the processor. When the device is sleeping however, meaning the radio is off and the processor is using 0.01 mW, we will take these costs into account. This is because the durations of the sleeping sessions are considerable.

7.1.2 Radio

When the CC2420 radio is in transceive mode, it consumes 30 mW. In receive mode, it consumes 35 mW. When it is idle, but its oscillator is running, it consumes about 0.712 mW [2]. However, because of inefficiencies in the current implementation of the SAND node[2], these numbers raise to 56 mW, 65 mW, and 1 mW respectively.

From the 802.15.4 standard [1], we calculate that sending a message with a payload of 10 bytes takes 0.7 ms. It takes 1 ms to start up the radio. Together, this means sending a message costs $0.7 \times 0.056 + 1 \times 0.001 + 1.7 \times 0.002$, hence, about 0.04 mJ. The terms are for sending, starting up the radio, and the processor, respectively. Note that the costs of the processor and of starting the radio are indeed negligible.

To pull for a new sample, we need to send a message requesting the data, which will take about 0.7 ms. Then, we need to wait at least one round-trip time before the answer arrives, and finally receive the answer. If the round-trip time is 50 ms, this means pulling a new sample costs $0.04+(50+0.7) * 0.065$, hence 3 mJ.

7.1.3 Thermometer

For measuring the temperature, v uses an analog sensor that measures a value that is transformed to a binary number. We set it to consume 100 mW and to take 10 ms to take a sample. These components will only be powered once taking a sample, which will happen one time per minute. This means that, on average, this component consumes 0.015 mW.

We use APTEEN [6] to send only necessary updates: we calculate how much the new sample differs from the previous value sent to w and only send an update if the temperature has changed more than 1 Kelvin. We choose that this is in average the case once every three samples, so once every three minutes. This means that when new values are pushed, on average one sample per three minutes is sent, which costs $\frac{0.04}{180} = 0.0002$ mW on average.

7.1.4 Actuator

For the actuator w, we assume that an array of LEDs is used to indicate the current temperature. The user can press a button, after which one LED will light up for ten seconds, indicating the current temperature. We assume the power of the LED is 5 mW. We assume that on average the user requests one temperature indication per hour. This means that, on average, this component consumes 0.013 mW.

In a pulling scenario, the device pulls for a new sample every minute, which hence costs $\frac{3}{60} = 0.05$ mW on average.

[2]This is due to a linear voltage regulator converting from 3.3 Volts coming from the battery to 1.8 Volts used in the CC2420

7.2 Pure-push

In pure-push, v sleeps all the time, except when taking or sending a new sample. However, w is constantly awake and ready to receive.[3]

This means that in this scenario v consumes $0.01 + 0.0015 + 0.0002 = 0.0117$ mW and w consumes $0.013 + 65$ mW.

This means that in this setup, v can run about 27 years, on a single battery. However, w can only run about 42 hours, on a single battery. This means that the total uptime of the system is 42 hours; after this time the user has to replace a battery.

7.3 Pure-pull

In pure-pull, v is constantly awake and ready to receive incoming pull requests, but w sleeps all the time, except when requesting a new sample.

This means that in this scenario v consumes $0.015 + 65$ mW and w consumes $0.01 + 0.013 + 0.013$, which is about 0.036 mW.

In this setup, v can only run about 42 hours on a single battery, whereas w can run for 9 years on a single battery. Again, the total up-time of the system is 42 hours; after this time the user has to replace a battery.

7.4 Adaptive Leases

Intuitively, adaptive leases will help because it can make the nodes switch between pulling and pushing, which is useful for synchronizing the pace at which the batteries of both v and w deplete.

7.4.1 Lease negotiation

We assume the effort of replacing the the battery in the thermometer is equal to the effort of replacing the actuator's battery. Therefore, we would like both batteries to be depleted at the same time.

We chose the following negotiation conditions:

1. If the battery of v has less energy than the battery of w, v always gives w a lease
2. Otherwise, no lease is granted and pulling must be used

In the messages from w to v, w includes the amount of energy left as argument to request a lease or not. When

[3]Of course, in a pushing scenario, we cannot let w sleep and only wake up once the button for a temperature indication is pressed by the user, because v will at that moment most likely be sleeping and not send a fresh value immediately.

v receives the request, it checks whether it can agree with w's proposal according to above negotiation conditions and decides whether to grant a lease or not.

7.4.2 Overhead of the protocol

We set the lease duration to 30 minutes, after which the lease is terminated and negotiation will take place again. Once the lease has terminated, v sends a message to w telling it that the lease has terminated and will have to wait at least one round-trip time for a lease renewal request. This costs v about 3 mJ per 30 minutes, which is 0.0016 mW. Also, w will have to send a lease renewal request once the lease has terminated, which costs 0.04 mJ per 30 minutes, which is negligible.

7.4.3 Results

It is easy to show that using adaptive leases leads to much longer up-time of the system. Both batteries will now deplete at the same pace. Running the adaptive leases protocol costs v an extra overhead of 0.0016 mW, which means its total energy consumption is now $0.0117 + 0.0016 = 0.0133$ mW when leases are granted.

To calculate up-time, we must to solve the following set of equations:

$$\begin{aligned} T_{push} + T_{pull} &= T_{total} \\ \frac{0.0133}{1000} \times T_{push} + \frac{65}{1000} \times T_{pull} &= 10000 \\ \frac{65}{1000} \times T_{push} + \frac{0.036}{1000} \times T_{pull} &= 10000 \end{aligned}$$

Hence, T_{push} and T_{pull} are both about 153000 seconds, which is about 42 hours. The total up-time is hence 84 hours.

This means that the up-time using adaptive leases is 84 hours, after which the user needs to replace both batteries. Effectively, adaptive leases doubled the up-time of the system.

8 Conclusion

Using adaptive leases is a promising mechanism for certain wireless sensor networks. It allows for nodes in the network to dynamically calibrate memory, bandwidth, processor and power usage. Contrary to standing queries, in adaptive leases the sink and the source negotiate on whether to engage in a lease, and possibly on the lease duration. Furthermore, the adaptive leases mechanism can be made totally transparent to the application programmer.

Adaptive leases seem to be most applicable in purely wireless sensor networks, such as wearable or in-body networks. Our simulation has shown that for some of these applications, adaptive leases can double the up-time of the system.

References

[1] IEEE Standard for Information technology – Telecommunication and information exchange between systems – Local and metropolitan area networks – Specific requirements. Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (LR-WPANs). IEEE Computer Society, Oct. 2003.

[2] B. Bougard, F. Catthoor, D. C. Daly, A. Chandrakasan, and W. Dehaene. Energy Efficiency of the IEEE 802.15.4 Standard in Dense Wireless Microsensor Networks: Modeling and Improvement Perspectives. In *DATE '05: Proceedings of the conference on Design, Automation and Test in Europe*, pages 196–201, Washington, DC, USA, 2005. IEEE Computer Society.

[3] V. Duvvuri, P. Shenoy, and R. Tewari. Adaptive Lease: A Strong Consistency Mechanism for the World Wide Web. Technical Report UM-CS-1999-041, 1999.

[4] M. S.-F. A. D. Leung. Pull vs. Push: A Quantitative Comparison for Data Broadcast. *"GLOBECOM -NEW YORK-"*, VOL 3:1464–1468, 2004.

[5] S. R. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong. TinyDB: An Acquisitional Query Processing System for Sensor Networks. *ACM Trans. Database Syst.*, 30(1):122–173, 2005.

[6] A. Manjeshwar and D. P. Agrawal. APTEEN: A Hybrid Protocol for Efficient Routing and Comprehensive Information Retrieval in Wireless Sensor Networks. In *IPDPS '02: Proceedings of the 16th International Parallel and Distributed Processing Symposium*, page 48, Washington, DC, USA, 2002. IEEE Computer Society.

[7] A. Yakhnin. Implementing Always-Up-To-Date Functionality in the .NET Compact Framework by Using Web Services. *http://msdn.microsoft.com/library/default.asp? url=/library/en-us/dnnetcomp/html/ autd_functionality_netcf_webservices.asp.*

Using a Prioritized MAC Protocol to Efficiently Compute Aggregated Quantities

Björn Andersson, Nuno Pereira and Eduardo Tovar
IPP Hurray Research Group
Polytechnic Institute of Porto, Portugal
{bandersson,npereira,emt}@dei.isep.ipp.pt

Abstract

Consider a distributed computer system such that every computer node can perform a wireless broadcast and when it does so, all other nodes receive this message. The computer nodes take sensor readings but individual sensor readings are not very important. It is important however to compute the aggregated quantities of these sensor readings. We show that a prioritized medium access control (MAC) protocol for wireless broadcast can compute simple aggregated quantities in a single transaction, and more complex quantities with many (but still a small number of) transactions. This leads to significant improvements in the time-complexity and as a consequence also similar reduction in energy "consumption".

1. Introduction

It has been recently discussed [1] that sensor networks often take many sensor readings of the same type (for example, temperature readings), and instead of knowing each individual reading it is important to know aggregated quantities of these sensor readings. For example, each computer node senses the temperature at the node and we want to know the maximum temperature among all nodes at a particular moment.

This can be solved with a naïve algorithm; every node broadcasts its sensor reading and hence all nodes know all sensor readings and then they can compute the aggregated quantity. This has the drawback that in a network with m nodes, it is required that m broadcasts are made. Considering that sensor networks are designed for large scale (for example thousands or millions of nodes), the naïve approach can be inefficient with respect to energy and cause a large delay.

In this paper we show that a prioritized MAC protocol for wireless broadcast can significantly improve the time-complexity for computing certain aggregated quantities. In particular we show that the minimum value can be computed with a time complexity that does not depend on the number of nodes. Also the time complexity increases very slowly as the possible range of the value increases. The same technique can be used to compute the maximum value. We also show how to compute a more complex aggregated quantitiy: the median. This computation hinges on the ability to compute the number of nodes. We propose such a technique but it only gives estimation and hence the median function is only estimated.

We consider this result to be significant because (i) the problem of computing aggregated quantities is common in wireless sensor networks which is an area of increasing importance and (ii) the techniques that we use depend on the availability of prioritized MAC protocols that support a very large range of priority levels; such protocols have recently been proposed [2], implemented and tested [3].

The remainder of this paper is structured as follows. Section 2 presents the system model and properties of the MAC protocol that we use. Section 3 shows how to compute the aggregated quantities. Section 4 shows how to estimate the number of proposed elements. Section 5 evaluates the algorithm for computing the number of elements. Section 6 discusses related work and this work. Section 7 gives conclusions.

2. System model

Consider a computer system comprised of m computing nodes that communicate over a wireless channel. Nodes do not have a shared memory; all data variables are local to each node. A computer node can make a wireless broadcast. This broadcast can be an unmodulated carrier wave or a message of data bits. We assume that all messages sent by nodes are related to computations of aggregate quantities. A node can transmit an empty message; that is, a message with no data. Every signal transmitted (unmodulated carriers or modulated data bits) is received by all computer nodes. This implies that there are no hidden stations and the network provides reliable broadcast.

Every node has an implementation of a MAC protocol. This MAC protocol is prioritized and collision-free. The fact that it is prioritized means that the MAC protocol assures that of all nodes that request to transmit at a moment, the one with the highest priority will transmit its data bits. The fact that it is collision-free implies that if priorities are unique then there is at most one node which transmits the data bits.

We assume that this MAC protocol is a dominance protocol. It operates as follows. The priority is encoded as a binary number with "0":s and "1":s. We say that a "0" is a dominant bit and a "1" is a recessive bit. We say that a low number represents a high priority. This is similar to the CAN bus [4]. Computer nodes agree on an instant when the tournament starts. Then nodes transmit the priority bits starting with the most significant bit. Priority bits are modulated using a variation of On-Off keying. A node sends an unmodulated carrier wave if it had a dominant bit and it sends nothing if it had a recessive bit. In the beginning of the tournament, all nodes have the potential to win but if it was recessive at a bit and perceived a dominant bit then it withdraws from the tournament and it cannot win. When a node has won the tournament, then it clearly knows the priority of the winner. If a node has lost the tournament then it continues to listen in order to know the priority of the winner.

The operating system exposes three system calls for interacting with other nodes. The `send` system call takes two parameters, one describing the priority of the message and one describing the data bits to be transmitted. If `send` loses the tournament then it waits until a new tournament starts. The program making this system call blocks until a message is successfully transmitted. The function `send_empty` takes only one parameters and it is a priority. Interestingly, `send_empty` does not take any parameter describing the data. The system call `send_empty` works like the function `send` but if it wins it does not send anything. In addition, when the tournament is over (regardless of whether the node wins or loses), the function `send_empty` gives the control back to the application and the function `send_empty` returns the priority of the winner. There is also a function `just_listen` which works like `send_empty` but it loses even before the first bit, so `just_listen` will only return the priority of the winner.

We assume that a computer node proposes a value. This value may be a sensor reading such as a temperature. Computer node N_i proposes the value v_i. The range of the value v_i is known; it is [*MINV*..*MAXV*], we assume $0 \le MINV$. For example it could be a 12 bit non-negative integer. Then the range is [0..4095]. All v_i have the same range for all proposed values. We assume that computer nodes do not know *m*.

We consider the problem of computing $f(v_1,v_2,\ldots,v_n)$ efficiently. We say that *f* is an *aggregated* quantity. We assume that there is one or many nodes that initiate the computation of *f*. When a node *i* has heard from one of these nodes that initiate the computation then node *i* proposes its value v_i. Every node has the potential to initiate a computation.

3. Computing aggregated quantities

We will first compute two simple quantities exactly in Section 3.1 and Section 3.2 and then, Section 3.3 shows how to compute a more complex quantity.

3.1. Computing the minimum value

Consider the case where the quantity that we want to compute $f(v_1,v_2,\ldots,v_m)$ is $min(v_1,v_2,\ldots,v_m)$. This can be performed as follows:

Algorithm 1. Calculating Min

```
When a node requests that min should be computed:
   Broadcast a message INITIATE_MIN
end
When a message INITIATE_MIN is received:
   Node i calculates value $v_i$ that it proposes.
   minv := calcmin( vi )
end
subroutine calcmin( vi )
   return send_empty( priority = vi )
end
```

3.2. Computing the maximum value

Let us consider the computation of $f(v_1,v_2,\ldots,v_m)$ is $max(v_1,v_2,\ldots,v_m)$. This can be performed as follows:

Algorithm 2. Calculating Max

```
When a node requests that max should be computed:
   Broadcast a message INITIATE_MAX
end
When a message INITIATE_MAX is received:
   Node i calculates value $v_i$ that it proposes.
   maxv := calcmax( vi )
end
subroutine calcmax( vi )
   return MAXV-send_empty( priority = MAXV - vi )
end
```

3.3. Computing the median value

We now consider the case where the function that we want to compute is the median of $v_1,v_2,\ldots,v_m$. We will find it convenient to introduce the notation $V_{less}(q)$ and $V_{greater}(q)$ as:

$$V_{less}(q) = \{v_j : v_j \le q\} \tag{1}$$

$$V_{greater}(q) = \{v_j : v_j \ge q\} \tag{2}$$

With these definitions our goal is to find *q* such that $\|V_{greater}(q)|-|V_{less}(q)\|$ is minimized. We assume the existence of the function `get_n_elements_in( LB, UB, active)`. It will be described in Section 4 and it returns the number of computer nodes that proposed a value which is greater than or equal to LB and less than or equal to UB.

Algorithm 3. Calculating median value

```
When a node requests that median should be
  computed:
   Broadcast a message INITIATE_MEDIAN
end
When a message INITIATE_MEDIAN is received:
   Node i calculates value vi that it proposes.
   median := calcmedianvalues( vi )
end
subroutine calcmedianvalue( vi )
   LB := MINV
   UB := MAXV
   for j:=1..to log2(MAXV-MINV) do
      mid := ( LB + UB ) / 2
      active   :=vi<=mid
      nVless   :=get_n_elements_in(LB,mid,active)
      active   :=vi>=mid
      nVgreater:=get_n_elements_in(mid,UB,active)
      if nVless<=nVgreater then
        LB := mid
      else
        UB := mid
      end if
   endfor
   return mid
end
```

4. Computing the number of proposed elements

Computing the number of proposed nodes is equivalent to computing the number of nodes. However, computing this is non-trivial. Consider a node i that proposes a value v_i. All nodes will receive a value R from `send_empty`. If $R = v_i$ then node i cannot know if it is the only node (and hence $m = 1$) or there are many other nodes with $v_i=R$ as well. In fact, with the use of our MAC protocol this is impossible to achieve for an algorithm where all nodes makes a single call to `send_empty` at the same time. Based on this impossibility, we will focus on algorithms that do not find the exact value of m, but try to find an estimate of m. The intuition is that each computer node generates a random number and if there is a large number of nodes then the minimum random number is very small. We repeat this k times. Hence a large value of k gives a good accuracy of the estimate whereas a low value of k has low time-complexity. We think k=5 is a reasonable compromise (which will be discussed later). Algorithm 4 describes this.

Algorithm 4. Calculating nelements

```
When a node requests that number of elements
  should be computed:
    Broadcast a message INITIATE_NELEMENTS
When a message INITIATE_NELEMENTS is received:
    nnodes :=get_n_elements_in(MINV, MAXV, TRUE)
end

subroutine get_n_elements_in( LB, UB, active)
    for q:=1 to k do
      if active then
        R[q] := send_empty(priority = random(LB,UB) )
      else
        R[q] := just_listen
      end if
    end
    return ML_estimation( R[1],…,R[k], LB, UB )
end
subroutine ML_estimation( R, LB, UB )
    for q:=1 to k do
      u[q] := (UB-R[q])/(UB-LB)
    endfor
    loginvsum := 0
    for q := 1 to k do
      loginvsum := logsinvsum + ln( 1/u[q] )
    endfor
    return k/loginvsum
end
```

In Algorithm 4, we conveniently ignore the possibility of an interval with no nodes. We can understand the function `ML_estimation` by considering the following analysis. Let A_j denote the event that there were j nodes. Let $B(R^k)$ denote the event that the minimum of the proposed values is R^l when we generated random numbers the l:th time.

Let $B(R)=B(R^1)\cap B(R^2) \cap\ldots B(R^k)$. Let A_j denote the event that there are j nodes. When we have the minimum of the proposed values (in Algorithm 4) we wish to compute P(A_j | $B(R)$) for all values of j and select the value of j that maximizes

$$P\left(A_j|B(R)\right) \tag{3}$$

We will do so now. We know from Bayes´s formula that:

$$P\left(A_j|\,B(R\,)\right)=\frac{P\left(B(R\,)\left|\,A_j\right.\right)\times P\left(A_j\right)}{\sum_{i=1}^{\infty}\left(P\left(B(R)\left|\,A_i\right.\right)\times P\left(A_i\right)\right)} \tag{4}$$

Let us assume that:

$$\forall j : P(A_1) = P(A_j) \tag{5}$$

Applying (5) in (4) gives us:

$$P\left(A_j\,|\,B(R\,)\right)=\frac{P\left(B(R\,)\left|\,A_j\right.\right)}{\sum_{i=1}^{\infty}P\left(B(R\,)\left|\,A_i\right.\right)} \tag{6}$$

Let us now compute P($B(R)|A_j$). We know that:

$$P\left(B(R\,)|\,A_i\right)=P\left(B(R^1)|\,A_i\right)\times\ldots\times P\left(B(R^k)|\,A_i\right) \tag{6b}$$

We obtain.

$$P\left(B(R^k)|\,A_i\right)=\frac{d}{dR}CDF\left(B(R^k)\left|\,A_i\right.\right) \tag{7}$$

where CDF is the probability that the minimum is less than or equal to R. We compute it as follows. The probability that a random number is greater than or equal to R is

$$P(random \quad number \geq R)=\frac{MAXV-R}{MAXV-MINV} \tag{8}$$

The probability that the minimum of the i randomly generated numbers is greater than or equal to R is

$$P(\min imum \quad number \geq R)=\left(\frac{MAXV-R}{MAXV-MINV}\right)^i \tag{9}$$

Hence, we obtain:

$$CDF\left(B(R^k)|A_i\right)=1-\left(\frac{MAXV-R}{MAXV-MINV}\right)^i \tag{10}$$

Combining (10) with (7) gives us:

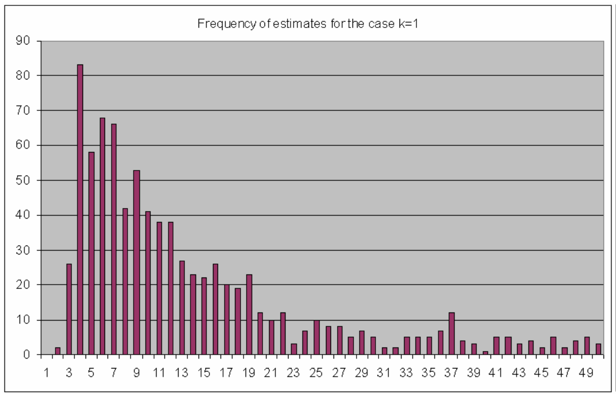
$$P\left(B(R^k)|A_i\right)=i\times\left(\frac{MAXV-R}{MAXV-MINV}\right)^{i-1} \tag{11}$$

Inserting (11) in (6) gives us:

$$P\left(A_j|B(R^k)\right)=\frac{j\times\left(\frac{MAXV-R}{MAXV-MINV}\right)^{j-1}}{\sum_{i=1}^{\infty}\left(i\times\left(\frac{MAXV-R}{MAXV-MINV}\right)^{i-1}\right)} \tag{12}$$

We wish to find the j that maximizes P(A_j | $B(R)$). We observe that this depends only on the numerator. Hence, we want to find the value of $j_{solution}$ that maximizes:

$$\prod_{q=1}^{k}\left(j_{solution}\times\left(\frac{MAXV-R^q}{MAXV-MINV}\right)^{j_{solution}-1}\right) \tag{13}$$



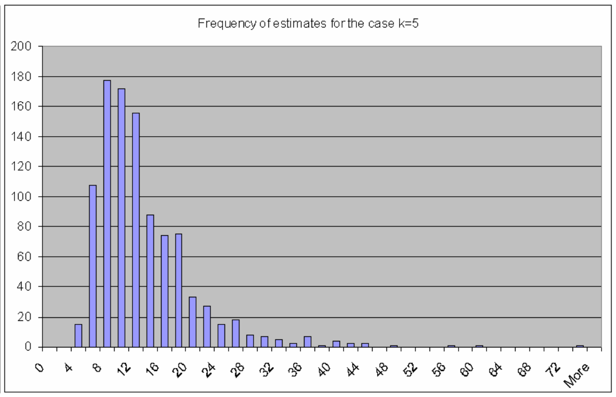


Figure 1. The frequency of the estimates for different values of *k*.

We can simplify (13) further. Let us use the notation:

$$u_q = \frac{MAXV - R^q}{MAXV - MINV} \tag{14}$$

and rewrite (13) we obtain that we want to maximize:

$$\prod_{q=1}^{k}\left(j_{solution} \times u_q^{\,j_{solution}-1}\right) \tag{15}$$

Observe that maximizing (15) is equivalent to maximizing the natural logarithm of (15). We know that the logarithm of a product is the sum of the logarithm of the factors. Hence, we want to maximize:

$$\sum_{q=1}^{k}\ln\left(j_{solution} \times u_q^{\,j_{solution}-1}\right) \tag{16}$$

We can rewrite (16) into the problem we want to maximize:

$$\sum_{q=1}^{k}\left(\ln j_{solution} + \left(j_{solution}\right)\times \ln u_q\right) \tag{17}$$

We have that the first derivative of (17) with respect to $j_{solution}$ is:

$$\sum_{q=1}^{k}\left(\frac{1}{j_{solution}} + \ln u_q\right) \tag{18}$$

And the second derivative of (17) with respect to $j_{solution}$ is:

$$-\sum_{q=1}^{k}\frac{1}{j_{solution}^{\;\;2}} \tag{19}$$

We can see from (18) and (19) that finding the $j_{solution}$ such that (18) is equal to 0 gives us the maximum likelihood estimate. Hence, we should select $j_{solution}$ such that:

$$\sum_{q=1}^{k}\left(\frac{1}{j_{solution}} + \ln u_q\right) = 0 \tag{20}$$

We can rewrite (20) to:

$$\frac{k}{j_{solution}} = -\sum_{q=1}^{k} \ln u_q \tag{21}$$

Rewriting yields:

$$\frac{1}{j_{solution}} = \frac{\sum_{q=1}^{k}\ln\frac{1}{u_q}}{k} \tag{22}$$

Further rewriting yields:

$$j_{solution} = \frac{k}{\sum_{q=1}^{k}\ln\frac{1}{u_q}} \tag{23}$$

This is a simple way to compute our estimate and we can see that `ML_estimation` in Algorithm 5 is based on this equation. We think it is simple enough to be used in a mote, although motes have very low processor speed.

5. Performance evaluation of nodes estimation

We have already mentioned that the calculation of the complex function median depends on the estimation of the number of nodes that propose a value. Hence, it is important that this estimation is accurate. For this purpose, we evaluate the accuracy using simulation experiments. Figure 1 shows the experimental results.

We ran 1000 experiments. For every experiment, 10 nodes generate random numbers and estimate the number of nodes. The estimation was made using (23). We can see that using five random numbers gives a significant improvement in the accuracy of the estimation as compared to one random number.

6. Related work and Discussion

6.1. Related work

A prioritized MAC protocol is useful to schedule real-time traffic [2, 3] and it can support data dissemination when topology is unknown [5]. In this paper we have discussed how to efficiently compute aggregated quantities using a prioritized MAC protocol.

Distributed calculations have been performed in previous research. It has been observed that nodes often [6, 7] detect an event and then needs to spread the knowledge of this event to its neighbours. This is called [6] one-to-*k* communication because only *k* neighbours need to receive the message. After that, the neighbour nodes perform local computations and reports back to the node that made the request for 1-to-*k* communication. This reporting back is called *k*-to-1 communication. Algorithms for both 1-to-*k* and *k*-to-1 communication are shown to be faster than naïve algorithm but unfortunately, the time-complexity increases as *k* increases. Our algorithms computes a function *f* and takes parameters from different nodes; this is similar to the average calculations in [8] . However our algorithms are different from [6, 7]; our algorithms have a time-complexity that does not depend on the number of nodes. We think our new algorithms are also useful building blocks for leader election and clock synchronization.

In this paper, nodes are permitted to use duplicated priorities, so any message transmitted after the tournament could collide and, for this reason, we use a send_empty primitive. However, it would be easy to code the priority in such a way that it would be unique by concatenating the node identifier to the priority. In this way, nodes could send a valid data message after winning the tournament. This is useful to because we may want to know not only the maximum value (for example the maximum temperature) but also other related values (for example the position of the node that detected the maximum temperature).

One way to use these algorithms is to encapsulate them in a query processor for database queries. Query processors for sensor networks have been studied in previous work [9, 10] but they are different in that they operate in multhop environment, do not compute aggregated quantities as efficiently as we do. They assume one single sink node and that the other nodes should report an aggregated quantity to this sink node. The sink node floods its interest in the data it wants into the network and this also makes nodes to discover the topology. When a node has new data it, broadcasts this data; other nodes hear it and it is routed and combined so that the sink node receives the aggregated. These works exploit the broadcast characteristics of the wireless medium (like we do) but they do not make any assumption on the MAC protocol (and hence they do not take advantage of the MAC protocol). One important aspect of these protocols is to create a spanning tree. It is known that computing an optimal spanning tree for the case when only a subset of nodes can generate data is equivalent to finding a Steiner-tree, a problem known to be NP-hard (the decision problem is NP-complete, see page 208 in [11]). For this reason, approximation algorithms have been proposed [12, 13]. However, in the average case, very simple randomized algorithms perform well [14]. Since a node will forward its data to the sink using a path which is not necessarily the shortest path to the sink, these protocols cause an extra delay. Hence, there is a trade-off between delay and energy-efficiency. To make this trade-off, a framework based on feedback was developed [15] for computing aggregated quantities. Techniques to aggregate data in the network such that the user at the base station can detect whether one node gives faked data has been addressed as well [16].

It has been observed that computing the median is especially difficult in multihop networks because combining two medians from different subnetworks is requires large amount of memory. Researchers in [17] observed that it is necessary for packets forwarded to be bigger and bigger the closer they get to the base station. Several algorithms for computing the exact median in O(*m*) time complexity are available (the earliest one is [18]). Our algorithm is faster; it has the time complexity O(log (MAXV-MINV)) but at the expensive of the accuracy of the result.

Computing averages has been done under the assumption that an adversary generates faults [19]. Unfortunately, it has a time-complexity which is larger than our algorithm and also larger than the algorithm proposed by [18] .

6.2. Practical issues

It is beyond the scope of this paper to describe all the details of the MAC protocol (see [2, 3] for details); it is important to observe however that the MAC protocol has the following properties. First, a priority bit has a duration adapted to time-of-flight, Rx/Tx switching time and time to detect a carrier and the duration of this bit can be quite large whereas a bit in the data packet has normal duration (for example on the CC2420 transceiver with a speed of 250kbps, a bit takes 4us). Hence, unlike CAN, in our protocol, the bit rate of the data transmission has the potential to be high even on long distances. Second, before the tournament in the protocol starts, the tournament waits for a long time of silence and synchronizes. This implies that even if nodes start the execution of the algorithms at slightly different times then the priority bits will be compared properly. This scheme only works if the different in time when message transmit messages "simultaneously" is not too big. We believe this assumption can easily be true however, by letting the algorithm start when it receives a message from a master node ordering the other nodes to start the execution of the algorithm.

So far we have assumed that all messages transmitted deal with aggregated quantities and we have assumed that there is only one type of aggregated quantity that we want to compute. This can be solved easily. We can subdivide the priority field into two subfield. The most significant bits are called service identifier and the least significant bits are called data bits. For example, we have 10 priority bits; the 4 most significant bits could be the service identifiers and the remaining 6 bits are priority bits. The MAC protocol runs the tournament base on all 10 bits. If the 4 services bits are 0000 then the following 6 bits denotes the priority of a normal message and these 6 bits number represent a unique priority and is normal payload and it is collision free. If the 4 bits are 0001 it means that the 6 remaining contains data that should be used to compute the maximum temperature. An application can make a function call `send_empty` (0001, 20) which proposes the value 20 and returns the maximum temperature.

7. Conclusions

We have shown how to use a prioritized protocol to compute aggregated quantities efficiently. The computational complexity for min and max is O(log2(MAXV-MINV)), that is they do not depend on the number of nodes. Our estimation of the median can be computed efficiently as well, its time complexity is $O(k*[\log2(MAXV-MINV)]^2)$.

References

[1] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless sensor networks: a survey," *Computer Networks*, vol. 38, pp. 393-422, 2002.

[2] B. Andersson and E. Tovar, "Static-Priority Scheduling of Sporadic Messages on a Wireless Channel," presented at International Conference on Principles of Distributed Systems (OPODIS´05), Pisa, Italy, 2005.

[3] N. Pereira, B. Andersson, and E. Tovar, "Implementation of a Dominance Protocol for Wireless Medium Access," presented at IEEE International Conference on Embedded and Real-Time Computing Systems and Applications, Sydney, Australia, 2006.

[4] Bosch, "CAN Specification, ver. 2.0, Robert Bosch GmbH, Stuttgart," 1991.

[5] B. Andersson, N. Pereira, and E. Tovar, "Disseminating Data Using Broadcast when Topology is Unknown," presented at Proceedings of the 26th IEEE Real-Time Systems Symposium, Work-in-Progress Session, Miami Beach, Florida, 2005.

[6] R. Zheng and L. Sha, "MAC Layer Support for Group Communication in Wireless Sensor Networks," Department of Computer Science, University of Houston UH-CS-05-14, July 21 2005.

[7] K. Jamieson, H. Balakrishnan, and Y. C. Tay, "Sift: A MAC Protocol for Event-Driven Wireless Sensor Networks," presented at Third European Workshop on Sensor Networks, Zurich, Switzerland, 2006.

[8] D. S. Scherber and H. C. Papadopoulos, "Distributed computation of averages over ad-hoc networks," *IEEE J. Select. Areas Commun*, vol. 23, pp. 776-787, 2005.

[9] Y. Yao and J. E. Gehrke, "Query Processing in Sensor Networks," presented at Proceedings of the First Biennial Conference on Innovative Data Systems Research (CIDR 2003), Asilomar, California, 2003.

[10] S. Madden, M. Franklin, J. Hellerstein, and W. Hong, "TAG: a Tiny Aggregation Service for Ad-Hoc Sensor Networks," presented at Proceedings of OSDI, Boston, MA., 2002.

[11] M. R. Garey and D. S. Johnson, *Computers and Intractability A guide to the Theory of NP-Completeness* New York: W. H. Freeman and Company, 1979.

[12] B. Krishnamachari, D. Estrin, and S. B. Wicker, "The Impact of Data Aggregation in Wireless Sensor Networks," presented at 22nd International Conference on Distributed Computing Systems, 2002.

[13] C. Intanagonwiwat, D. Estrin, R. Govindan, and J. Heidemann, "Impact of Network Density on Data Aggregation in Wireless Sensor Networks," presented at Proceedings of the 22nd International Conference on Distributed Computing Systems, Vienna, Austria, 2002.

[14] M. Enachescu, A. Goel, R. Govindan, and R. Motwani, "Scale Free Aggregation in Sensor Networks," presented at First International Workshop on Algorithmic Aspects of Wireless Sensor Networks, 2004.

[15] T. Abdelzaher, T. He, and J. Stankovic, "Feedback Control of Data Aggregation in Sensor Networks," presented at 43rd IEEE Conference on Decision and Control, Paradise Island, Bahamas, 2004.

[16] B. Przydatek, D. Song, and A. Perrig, "SIA: Secure Information Aggregation in Sensor Networks," presented at ACM SenSys (Conference on Embedded Networked Sensor Systems)], 2003.

[17] N. Shrivastava, C. Buragohain, D. Agrawal, and S. Suri, "Medians and beyond: new aggregation techniques for sensor networks. SenSys 2004: 239-249," presented at SenSys, Baltimore, Maryland, USA, 2004.

[18] M. Blum, R. W. Floyd, V. Pratt, R. Rivest, and R. Tarjan, "Time bounds for selection," *J. Cornput. System Sci*, vol. 7, pp. 448-461., 1973.

[19] M. Kutylwski and D. Letkiewicz, "Computing Average Value in ad hoc Networks," presented at MFCS 2003 28th International Symposium on Mathematical Foundations of Computer Science, Bratislava, Slovak Republic, Europe, 2003.

Improving the IEEE 802.15.4 Slotted CSMA/CA MAC for Time-Critical Events in Wireless Sensor Networks

Anis KOUBAA[1], Mário ALVES[1], Bilel NEFZI[2], Ye-Qiong SONG[2]

[1] IPP-HURRAY! Research Group, Polytechnic Institute of Porto
Rua Dr. Antonio Bernardino de Almeida, 431, 4200-072 Porto, PORTUGAL

[2] LORIA-TRIO, 615 rue du jardin botanique 54600 Villers Les Nancy FRANCE

akoubaa@dei.isep.ipp.pt, mjf@isep.ipp.pt, bilel.nefzi@loria.fr, song@loria.fr

Abstract

In beacon-enabled mode, IEEE 802.15.4 is ruled by the slotted CSMA/CA Medium Access Control (MAC) protocol. The standard slotted CSMA/CA mechanism does not provide any means of differentiated services to improve the quality of service for time-critical events (such as alarms, time slot reservation, PAN management messages etc.). In this paper, we present and discuss practical service differentiation mechanisms to improve the performance of slotted CSMA/CA for time-critical events, with only minor add-ons to the protocol. The contribution of our proposal is more practical than theoretical since our initial requirement is to leave the original algorithm of the slotted CSMA/CA unchanged, but rather tuning its parameters adequately according to the criticality of the messages. We present a simulation study based on an accurate model of the IEEE 802.15.4 MAC protocol, to evaluate the differentiated service strategies. Four scenarios with different settings of the slotted CSMA/CA parameters are defined. Each scenario is evaluated for FIFO and Priority Queuing. The impact of the hidden-node problem is also analyzed, and a solution to mitigate it is proposed.

1. Introduction

Providing Quality of Service (QoS) support in Wireless Sensor Networks (WSNs) for improving their timing and reliability performance under severe energy constraints has attracted recent research works [1-3]. The standardization efforts of the IEEE Task Group 15.4 have contributed to solve this problem by the definition of the IEEE 802.15.4 protocol for *Low-Rate, Low-Power* Wireless Personal Area Networks (WPANs) [4]. In fact, this protocol shows great potential for flexibly fitting different requirements of WSN applications by adequately setting its parameters (low duty cycles, guaranteed time slots (GTS)). In beacon-enabled mode, the IEEE 802.15.4 protocol uses slotted CSMA/CA as a Medium Access Protocol (MAC). Even though the IEEE 802.15.4 protocol provides the GTS allocation mechanism for real-time flows, the allocation must be preceded by an allocation request message. However, with its original specification, the slotted CSMA/CA does not provide any QoS support for such time-sensitive events, including GTS allocation requests, alarms, PAN management commands, etc., which may result in unfairness and degradation of the network performance, particularly in high load conditions.

Related work. The improvement of CSMA/CA MAC mechanisms has drawn many research efforts. Particularly for the case of the IEEE 802.15.4 protocol, some recent research works have contributed to enhance the slotted CSMA/CA mechanism for achieving reduced (soft) delay guarantees and better reliability of time-critical events, as described next.

In [5], the authors modified the slotted CSMA/CA algorithm to enable fast delivery of high priority frames in emergency situations, using a priority toning strategy. Nodes that have high priority frames to be transmitted must send a tone signal just before the beacon transmission. If the tone signal is detected by the PAN Coordinator, an emergency notification is conveyed in the beacon frame, which alerts other nodes with no urgent messages to defer their transmissions by some amount of time, in order to privilege high priority frame transmissions at the beginning of the contention access period. In [6], the authors extend the previous schemes by allowing high priority frames to perform only one *Clear Channel Assessment* (CCA) operation instead of two, using a frame tailoring strategy, which aims to avoid collisions between data frames and acknowledgment frames when only one CCA is performed. These solutions seem to improve the responsiveness of high priority frames in IEEE 802.15.4 slotted CSMA/CA, but require a non-negligible change to the IEEE 802.15.4 MAC protocol to support the priority toning and frame tailoring strategies, thus turning them non-compatible with the standard.

In this paper, we investigate other alternatives for improving slotted CSMA/CA without forcing fundamental changes to the MAC protocol. We particularly aim to assess different parameter settings of the protocol with some basic queuing strategies (FIFO and Priority Queuing) for each traffic priority. Note that in [5, 6], the toning mechanism imposes some changes to the hardware (using a tone signal transmitter) and also to the protocol itself, due to the frame tailoring strategy.

The motivation for proposing differentiated QoS support with only minor add-ons to the slotted CSMA/CA mechanism is to ensure backward compatibility with the standard. Also, we would like to assess if such a simple approach is sufficient to satisfy the requirements of time-critical messages. This proposal can be easily adopted in the IEEE 802.15.4b extension [7] of the standard.

The rest of the paper is organized as follows. Section 2 highlights the IEEE 802.15.4 features and its slotted CSMA/CA mechanism. Section 3 presents the proposed differentiation service strategies. Section 4 presents the simulation study and performance evaluation results. Section 5 concludes the paper.

2. IEEE 802.15.4 Slotted CSMA/CA MAC

In beacon-enabled mode, beacon frames are periodically sent by a central device, referred to as *PAN coordinator*, to identify its PAN and synchronize nodes that are associated with it. The PAN coordinator defines a superframe structure characterized by a *Beacon Interval* (*BI*) specifying the time between two consecutive beacons, and a *Superframe Duration* (SD) corresponding to the active period, defined as:

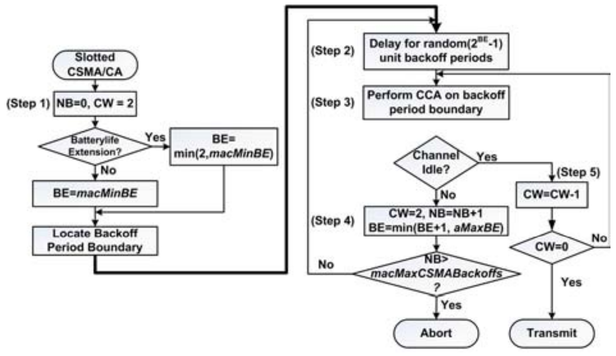
$$\begin{aligned} BI &= aBaseSuperframeDuration \cdot 2^{BO} \\ SD &= aBaseSuperframeDuration \cdot 2^{SO} \\ &for \;\; 0 \le SO \le BO \le 14 \end{aligned} \qquad (\mathbf{1})$$

BO and *SO* are called *Beacon Order* and *Superframe Order*, respectively. The Beacon Interval may optionally include an inactive period (for *SO* < *BO*), in which all nodes may enter into a sleep mode, thus saving energy. More details can be found in [4].

By default, nodes compete for medium access using slotted CSMA/CA during the *Contention Access Period* (CAP). The

IEEE 802.15.4 protocol also provides a *Contention-Free Period* (CFP) within the superframe, in which a node may request the PAN coordinator to allocate Guaranteed Time Slots (GTS). In this paper, we consider the physical layer operating in the 2.4 GHz frequency band and with a 250 kbps data rate.

The slotted CSMA/CA algorithm is based on a basic time unit called *Backoff Period* (BP), which is equal to $aUnitBackoffPeriod = 80$ bits (0.32 ms). The slotted CSMA/CA backoff algorithm mainly depends on three variables: **(1)** the *Backoff Exponent* (*BE*) enables the computation of the backoff delay, **(2)** the *Contention Window* (*CW*) represents the number of BPs during which the channel must be sensed idle before channel access, **(3)** the *Number of Backoffs* (*NB*) represents the number of times the CSMA/CA algorithm was required to backoff while attempting to access the channel. Fig. 1 presents the slotted CSMA/CA algorithm [4].



Slotted CSMA/CA
(Step 1) NB=0, CW = 2
Batterylife Extension?
Yes
No
BE= min(2,macMinBE)
BE=macMinBE
Locate Backoff Period Boundary
(Step 2) Delay for random(2^BE-1) unit backoff periods
(Step 3) Perform CCA on backoff period boundary
Channel Idle?
(Step 5) CW=CW-1
(Step 4) CW=2, NB=NB+1 BE=min(BE+1, aMaxBE)
CW=0
NB> macMaxCSMABackoffs ?
Abort
Transmit

Fig. 1. The slotted CSMA/CA algorithm

First, the number of backoffs and the contention window are initialized ($NB = 0$ and $CW = CW_{init} = 2$) (Step 1). The backoff exponent is also initialized to $BE = 2$ or $BE = \min(2, macMinBE)$, depending on the value of the *Battery Life Extension* MAC attribute. *macMinBE* is a constant, which is by default equal to 3. Then, the algorithm starts counting down a random number of BPs uniformly generated within $[0, 2^{BE}-1]$ (Step 2). The count down must start at the boundary of a BP. When the timer expires, the algorithm then performs one CCA operation at the BP boundary to assess channel activity (Step 3). If the channel is busy (Step 4), *CW* is re-initialized to $CW_{init} = 2$, *NB* and *BE* are incremented. *BE* must not exceed *aMaxBE* (default value fixed to 5). Incrementing *BE* increases the probability for having greater backoff delays. If the maximum number of backoffs ($NB = macMaxCSMABackoffs = 5$) is reached, the algorithm reports a failure to the higher layer; otherwise, it goes back to (Step 2) and the backoff operation is restarted. The protocol allows $aMaxFrameRetries = 3$ after each failure. If the channel is sensed as idle, *CW* is decremented (Step 5). The CCA is repeated if $CW \neq 0$. This ensures performing two CCA operations to prevent potential collisions of acknowledgement frames. If the channel is again sensed as idle, the node attempts to transmit, provided that the remaining BPs in the current CAP are sufficient to transmit the frame and the subsequent acknowledgement. If not, the CCAs and the frame transmission are both deferred to the next superframe. This is referred to as *CCA deference*.

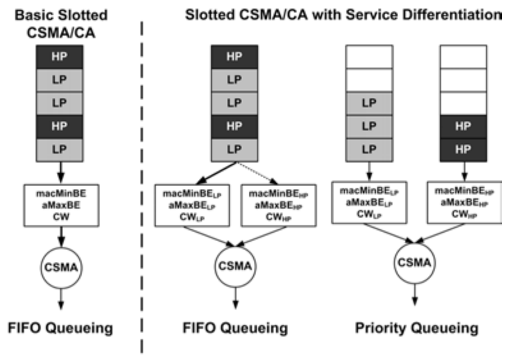
3. Service Differentiation Strategies for Slotted CSMA/CA

Observe that the behavior of slotted CSMA/CA is affected by four *initialization* parameters, which are: (1) the minimum backoff exponent (*macMinBE*), (2) the maximum backoff exponent (*aMaxBE*), (3) the initial value of the *CW* (CW_{init}) and (4) the maximum number of backoffs (*macMaxCSMABackoffs*).

Changing the value of any of these parameters will have an impact on the performance. For instance, a performance evaluation study in [8] has shown that the average delay of broadcast frames increases with *macMinBE*, whereas the probability of success remains independent of *macMinBE* in large-scale WSNs. However, the probability of success increases for high *macMinBE* values, in small-scale WSNs. Based on those observations, we propose to offer differentiated services for time-critical messages. In this paper, our service differentiation mechanisms are particularly based on the *macMinBE*, *aMaxBE* and CW_{init} parameters.

Note that IEEE 802.15.4 defines two frame types: (1) *data traffic*, which typically represents sensory data broadcasted to the network (without using acknowledgments), (2) and *command traffic*, which comprises critical messages (such as alarm reports, PAN management messages and GTS requests) sent by sensor nodes to the PAN Coordinator. Due to their importance, command frames are sent using acknowledged transmissions to ensure the reliability of their transfer, and require a particular QoS support to be delivered to their destination in a bounded time interval. In this paper, we consider command frames as the *high priority service class* and data frames as the *low priority service class*.

The differentiated service strategies are presented in Fig. 2.



Basic Slotted CSMA/CA
Slotted CSMA/CA with Service Differentiation
HP
LP
macMinBE aMaxBE CW
macMinBE_LP aMaxBE_LP CW_LP
macMinBE_HP aMaxBE_HP CW_HP
CSMA
FIFO Queueing
FIFO Queueing
Priority Queueing

Fig. 2. Differentiated service strategies

The idea is simple. Instead of having the same CSMA/CA parameters for both traffic types, we assign each class its own attributes. We denote [$macMinBE_{HP}$, $aMaxBE_{HP}$] and CW_{HP} the backoff interval and the contention window initial values for high priority traffic related to command frames, and [$macMinBE_{LP}$, $aMaxBE_{LP}$] and CW_{LP} the initial values for low priority traffic related to data frames. While, the slotted CSMA/CA described in Section 2 remain unchanged, the adequate initial parameters that correspond to each service class must be applied.

In addition to the specification of different CSMA/CA parameters, Priority Queuing can be applied to reduce queuing delays of high priority traffic (Fig. 2). In this case, slotted CSMA/CA uses priority scheduling to select frames from queues, and then applies the adequate parameters corresponding to each service class. Note that if a low priority frame is selected, i.e. the high priority queue is empty, then the backoff process corresponding to this frame will not be preempted, if a high priority frame arrives during that service time, until this frame is sent, or rejected if the maximum number of backoff is reached.

The heuristics for adequately setting the CSMA/CA parameters are the following. Intuitively, a first differentiation consists in setting CW_{HP} smaller than CW_{LP}. It results that low priority traffic has to assess the channel to be idle for a longer time before transmission. A second differentiation is related to the backoff interval. Providing lower backoff delay values for high priority traffic by setting $macMinBE_{HP}$ lower than $macMinBE_{LP}$ would improve its responsiveness without degrading its throughput, as it has been observed in [8]. These intuitive heuristics are evaluated in the next section.

4. Performance Evaluation

4.1 Simulation Workload and scenarios

We present a simulation study based on an accurate model of IEEE 802.15.4 using OPNET simulator [9], to assess the impact of differentiated services in slotted CSMA/CA. We consider a WSN in a surface of 100 m x 100 m with one PAN coordinator, *BO* = *SO* = 3 and 100 identical nodes (randomly spread) generating *low priority* (*data*) *traffic* with Poisson distributed arrivals with the same mean arrival rate. The data frame size is fixed to 404 bits (300 bits of data payload + 104 bits of MAC header). These nodes also generate *high priority* (*command*) *traffic* with an inter-arrival time exponentially distributed with a mean value equal to 1 second. The command frame size is fixed to 304 bits (200 bits of data payload + 104 bits of MAC header). Frame size values are chosen as illustrative examples of short frame sizes. Command frames are sent from nodes to the PAN Coordinator using acknowledged transmissions. Data frames are simply broadcasted to the network. The maximum number of backoffs *macMaxCSMABackoffs* is equal to 4 and the maximum number of retries *aMaxFrameRetries* is by default equal to 3. The transmission power is equal to 0.1 mW.

The simulation study consists in varying the intensity of data traffic, while the command frames remain exponentially generated with the average of 1 frame/second in each node, and analyzing the performance of command frames in terms of average delay (*D*), probability of success (*S/Gapp*) and power efficiency. *S* denotes the throughput of command frames and *Gapp* denotes the offered load of command frames generated by the application layers of 100 nodes. In this study, *Gapp* is approximately equal in average to 31.5 kbps (= 100 * 304 bits per second), which represents 12.5% of the overall network capacity (250 kbps).

Note that there is a difference between *Gapp* and *Gmac*. The latter is defined as the offered load generated by the MAC layers due to the transmissions of original command frames and the retransmissions of their copies in case of non successful delivery. Hence, the power efficiency is reflected by the *Gmac* performance metric, i.e. fewer retransmissions (lower *Gmac*) results in a better power efficiency.

In this paper, the performance of data frames is also analyzed in terms of average delay and probability of success $\left(S_{data}/G_{mac}^{data}\right)$, which reflects the degree of reliability achieved by the network for successful transmissions of data frames. In case of data traffic, the probability of success is measured by the throughput of data frames S_{data} divided by the offered load of data frames generated by the MAC layers (G_{mac}^{data}). Since there is no retransmissions in case of a transmission failure of a data frame (unacknowledged transmissions), G_{mac}^{data} is at most equal to G_{app}^{data}, which is the data traffic generated by the application layer. This is because, at a given time, it may happen that some data frames are still waiting for service in the queue. Note that in our scenario with 100 nodes, we have verified that $G_{mac}^{data} = G_{app}^{data}$, for all network loads (*G*) considered in this simulation study (no buffer overflow for data frames). The network load (*G*) represents all command and data frames generated by the MAC layers of 100 nodes.

We consider four different scenarios, presented in Table 1. Each scenario is simulated with FIFO and Priority Queuing scheduling policies (refer to Fig. 2).

Table. 1. Simulation scenarios

Scenario	[macMinBE$_{HP}$, aMaxBE$_{HP}$]	[macMinBE$_{LP}$,aMaxBE$_{LP}$]	CW$_{HP}$	CW$_{LP}$
Sc1	[2,5]	[2,5]	2	2
Sc2	[2,5]	[2,5]	2	3
Sc3	[0,5]	[2,5]	2	2
Sc4	[0,5]	[2,5]	2	3

4.2 Case of a fully connected network (no hidden-node problem)

First, we consider a fully connected network, where all nodes hear each other.

Fig. 3 clearly shows the impact of the first differentiation scheme related to the initial contention window size on the success probability. As it was intuitively expected, setting CW_{LP} greater than CW_{HP} notably results in higher throughputs for high priority command frames, either for FIFO or Priority Queuing. The success probability remains satisfactory even in high load conditions for Sc2 and Sc4 (up to 80%). However, the effects of *macMinBE* and scheduling policies are negligible on *S*/*Gapp* since Sc1 and Sc3 have the same throughput (similarly to Sc2 and Sc4) for different *macMinBE* values. This confirms the result in [8].

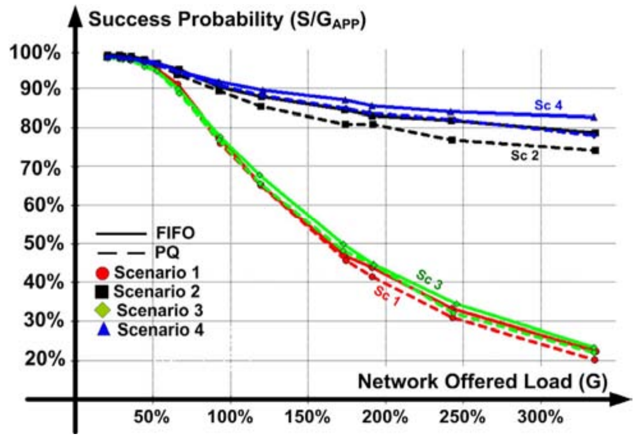


Fig. 3. Success probability of command frames without hidden nodes

Fig. 4 shows the average delays for all scenarios. Sc1 is only comparable to Sc3, whereas Sc2 is comparable to Sc4, due to the success probability results in Fig. 3 (it is not logical to compare delays for scenarios with different success probabilities).

Observe that lower *macMinBE* for high priority frame leads to lower average delays, since the backoff delays are reduced. The beauty of this result is that lower *macMinBE* does not degrade the throughput, as shown in Fig. 3. The advantage of using Priority Queuing in reducing average delays is also observable in Fig. 4.

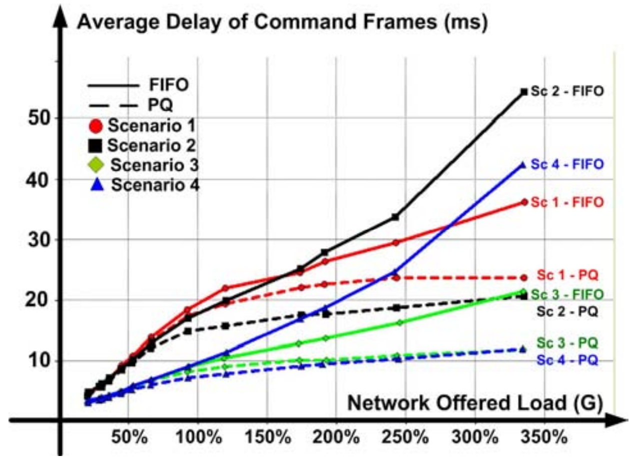


Fig. 4. Average delay of command frames (ms) without hidden nodes

As for power efficiency (Fig. 5), setting CW_{LP} greater than CW_{HP} clearly results in lower energy consumption, since fewer retransmissions are needed in Sc2 and Sc4. Priority Queuing seems also to be advantageous for improving energy efficiency.

The impact of *macMinBE* on *Gmac* depends on the values of CW_{LP} and CW_{HP}. If both are equal (Sc1 and Sc3), higher *macMinBEs* are more energy efficient. However, if $CW_{LP} < CW_{HP}$ (Sc2 and Sc4) lower *macMinBEs* are more energy efficient. This is because retransmissions are mostly due to collisions with data frames. Since Sc4 provides more differentiation to high priority frames than the other scenarios, it presents the best performance for all metrics.

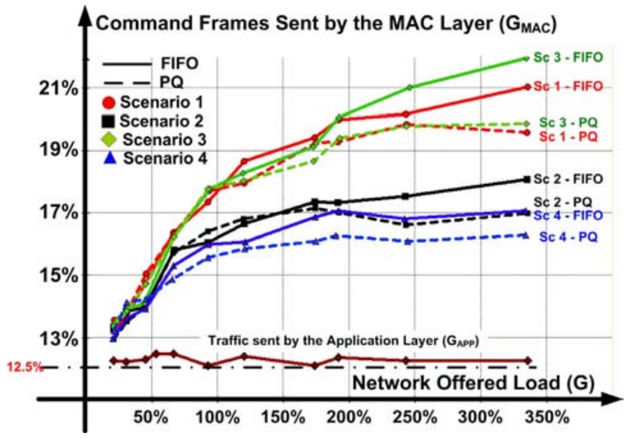


Fig. 5. Command traffic sent by the MAC layer without hidden nodes

As for the performance of low priority data frames, Figs. 6 and 7 present the success probability and the average delay, respectively. In Fig. 6, it is shown that setting CW_{LP} greater than CW_{HP} (Sc2 and Sc 4) results in relatively lower throughputs for low priority data frames, due to the privileges given to the high priority frames, as it can be observed in Fig. 3. However, the improvement of this differentiation scheme to the throughput of high priority command frames is more significant than the degradation of the throughout of low priority data frames, which further demonstrates the efficiency of this differentiation mechanism.

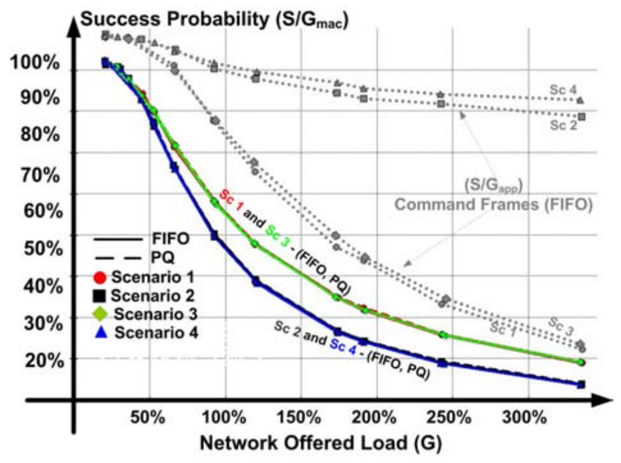


Fig. 6. Success probability of data frames without hidden nodes

In Fig. 7, observe that setting CW_{LP} greater than CW_{HP} results in greater average delays for data frames. This is because low priority data frames have a smaller probability to access the medium than high priority command frames when CW_{LP} increases, leading to return more often to the backoff process. This results in additional queuing and backoff delays (BE increases each time the channel is sensed busy) for data frames.

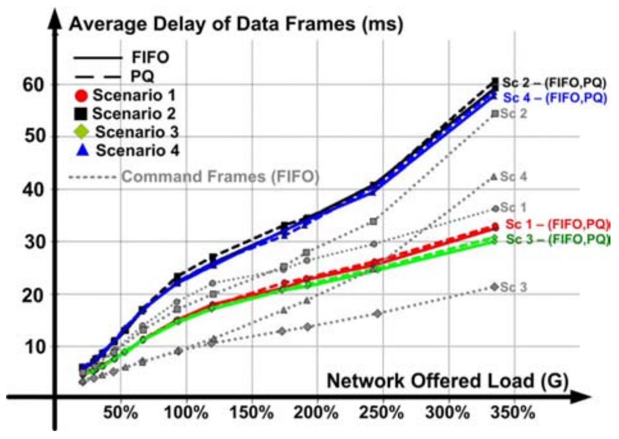


Fig. 7. Success probability of data frames without hidden nodes

Note that the Priority Queuing scheduling mechanism does not degrade the average delays of data frames even though they receive a low priority service. This is due to the fact that, in these simulation scenarios, command frames only use 12.5% (31.5 kbps) of the network capacity. The degradation would be more significant if command frames were generated at a higher rate. This behavior is typical for many WSNs, since command frames are likely to be generated with lower rate than data frames.

Another interesting observation is that the average delays of command frames are lower than those of data frames in all scenarios, except in Sc1 which does not provide any kind of differentiation. As a result, it is clearly shown that using one or both differentiation strategies (*CW* and/or *macMinBE*) always results in an improved performance for high priority frames.

4.3 Case of partially connected network (hidden-node problem)

We consider a partially connected network (we adjust the sensing sensitivity of the nodes to limit their communication range), to evaluate the impact of the hidden-node problem on the performance of slotted CSMA/CA with differentiated services. The sensing and receiving sensitivities are set such that the transmission range of each sensor node is limited to 32 m (command and data frames are sent with a transmission power equal to 0.1 mW). Beacon frames are sent by the PAN Coordinator at a transmission power equal to 1 mW, which is sufficient to reach all the nodes in the WSN. No routing protocol is used. Frames are simply broadcasted to the network (1) since most WSNs rely on broadcast transmissions and (2) we would like to provide results independent from any routing protocol.

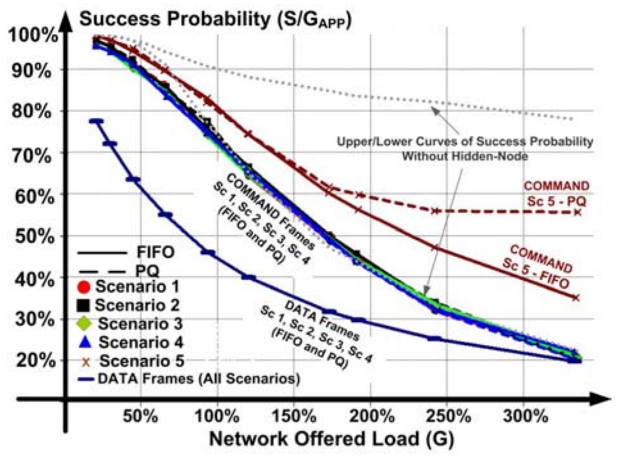


Fig. 8. Success probability of command/data frames with hidden nodes

It can be observed in Figs. 8, 9, 10 and 11 that the differentiated service strategies of the four scenarios defined in Table 1 have practically no impact on the performance metrics for both command and data frames, with an exception for the average delays. As shown in Fig. 9, lower *macMinBEs* slightly reduce the average delays of command frames. On the other hand, observe in Fig. 10 that greater CW_{LP} only results in a non significant increase of the average delays of low priority frames (difference around 1 ms). The success probabilities of command frames, as well as of data frames, remain closely insensitive to the differentiation service strategies in the four scenarios. In addition, The Priority Queuing scheduling policy has no impact on the improvement of the performance of high priority command frames.

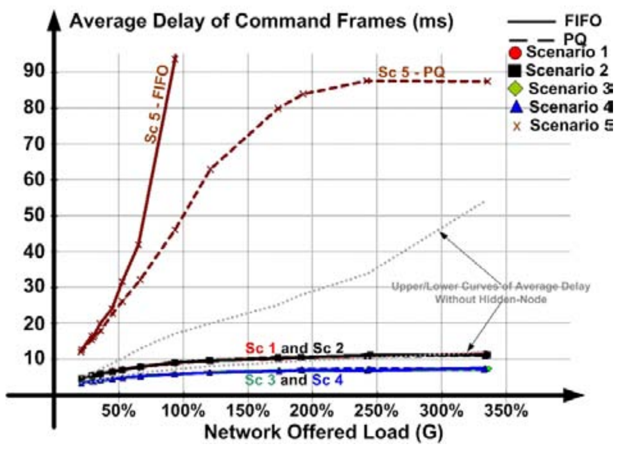


Fig. 9. Average delay (ms) of command frames with hidden nodes

These results clearly infer the severe impact of the hidden-node problem on the degradation of the performance of slotted CSMA/CA. Since nodes cannot hear each other, multiple hidden-node collisions occur independently of the differentiation schemes.

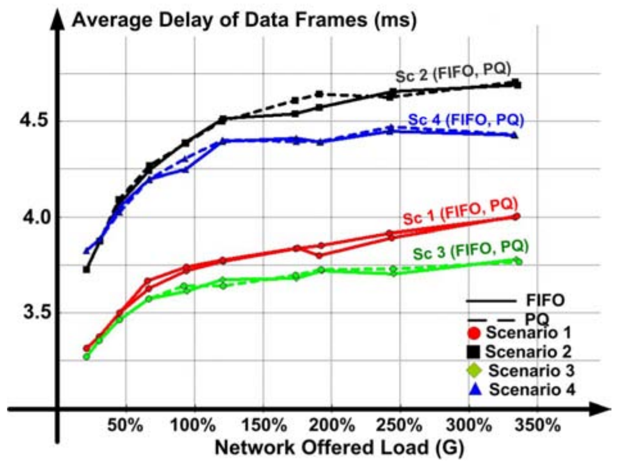


Fig. 10. Average delay (ms) of data frames with hidden nodes

The hidden-node impact is mainly a result of the small backoff interval duration. Note that with *aMaxBE* value equal to 5, the maximum backoff delay is equal to 31 BPs, which is not sufficient to avoid hidden-node collisions. One option to mitigate the hidden-node problem is to increase the backoff delay, such that competing nodes wait longer. Hence, other nodes would have more chance to successfully transmit their frames without facing hidden-node collisions. To illustrate this intuition, we propose the following additional scenario Sc5.

Table. 2. Hidden-node avoidance scenario

Scenario	[macMinBE$_{HP}$, aMaxBE$_{HP}$]	[macMinBE$_{LP}$,aMaxBE$_{LP}$]	CW$_{HP}$	CW$_{LP}$
Sc5	[4,6]	[7,8]	2	10

By increasing *macMinBE* and *aMaxBE* for both high priority and low priority traffics, the backoff delay will clearly increase for both traffic classes. Observe also that CW_{LP} is set to 10 and CW_{HP} is set to 2, to give more privileges to high priority frames.

It can be observed in Fig. 8 that the configuration of Sc5 noticeably improves the throughput of command frames, by reducing hidden-node collisions. With Priority Queuing in Sc5, the success probability reaches more that 55% even in high load conditions. However, reporting to Fig. 9, the average delays can be very large with FIFO scheduling, but are more steady using Priority Queuing (less than 90 ms).

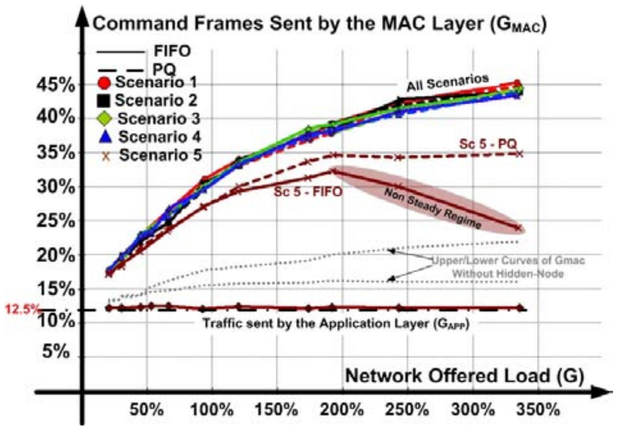


Fig. 11. Command traffic sent by the MAC layer with hidden nodes

Note that in Sc5 with FIFO, the network operates in a non steady regime (Fig. 11) in high load conditions, due to overloaded queues, which explains the expansion of average delays. The same behavior occurs for low priority data frames, both with FIFO and Priority Queuing. This is due to the blocking of high priority command frames by low priority data frames, which must wait for 10 CCA before transmission. However, with Priority Queuing, Sc5 is more energy efficient since fewer retransmissions than is other scenarios are performed.

5. Discussions

We have proposed a simple differentiated service scheme for slotted CSMA/CA in IEEE 802.15.4 to improve the performance of time-sensitive messages. It has been shown that tuning adequately the parameters of slotted CSMA/CA may result in an improved QoS for time-critical messages. This practical proposal can be easily adopted in the IEEE 802.15.4b extension of the standard since it only requires minor add-ons and ensures backward compatibility with the existing standard.

We have run the same simulation scenarios [10] using the implementation of the IEEE 802.15.4 protocol in the NS-2 simulator [11], for (1) comparative purposes, (2) the validation of our simulation results. The results obtained using NS-2 show a similar behavior to the results presented in this paper, thus confirming the validity of the approach. However, the values of the average delays observed in NS-2 results are greater than those obtained with our OPNET model. Also, NS-2 produces lower throughputs than those obtained with OPNET. To our understanding, this is mainly due to the amount of additional overheads introduced by the NS-2 simulator, since it imposes the use of a UDP (User Datagram Protocol) agent in each node for generating data, and also the generation of ARP (Address Resolution Protocol) frames. This is mainly because NS-2 was originally developed for IP (Internet Protocol) networks and then extended for IEEE 802.11 wireless networks. According to our personal experience, we strongly believe that the current version of the NS-2 simulator is not accurate for the simulation of wireless sensor networks, even though existing modules can be reused in this context. Our OPNET model implements more accurately the IEEE 802.15.4 protocol without these unnecessary overheads, turning its results more reliable than those obtained with NS-2.

References

[1] T. F. Abdelzaher, S. Prabh, and R. Kiran, "On Real-Time Capacity Limits of Multihop Wireless Sensor Networks," in Proceedings of the IEEE International Real-Time Systems Symposium, Lisbon, Portugal, 2004.

[2] J. A. Stankovic, T. Abdelzaher, C. Lu, L. Sha, and J. Hou, "Real-Time Communication and Coordination in Embedded Sensor Networks," *Proceedings of the IEEE*, vol. 91, pp. 1002-1022, 2003.

[3] A. Koubâa, M. Alves, and E. Tovar, "IEEE 802.15.4: a Federating Communication Protocol for Time-Sensitive Wireless Sensor Networks," in *Technical Report TR-060201, to appear in Sensor Networks and Configurations: Fundamentals, Techniques, Platforms, and Experiments*, N. P. Mahalik, Ed. Germany: Springer-Verlag, 2006.

[4] IEEE-TG15.4, "Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (LR-WPANs)," *IEEE standard for Information Technology*, 2003.

[5] T. Kim, D. Lee, J. Ahn, and S. Choi, "Priority toning strategy for fast emergency notification in IEEE 802.15.4 LR-WPAN," in Proceedings of the 15th Joint Conference on Communications & Information (JCCI), April, 2005.

[6] T. Kim and S. Choi, "Priority-based delay mitigation for event-monitoring IEEE 802.15.4 LR-WPANs," *IEEE Communications Letters*, Nov. 2005.

[7] IEEE-TG15.4b, "http://grouper.ieee.org/groups/802/15/pub/TG4b.html."

[8] A. Koubâa, M. Alves, and E. Tovar, "A Comprehensive Simulation Study of Slotted CSMA/CA for IEEE 802.15.4 Wireless Sensor Networks," in Proceedings of the 6th IEEE International Workshop on Factory Communication Systems (WFCS 2006), Torino (ITALY), 2006.

[9] OPNET, "OPNET Simulator, v 11, http://www.opnet.com."

[10] B. Nefzi, "Performance Evaluation and Improvement of the Slotted CSMA/CA MAC of the IEEE 802.15.4 Protocol ": Master Project, (LORIA, IPP-HURRAY and Sup'Com), 2006.

[11] NS-2, "NS-2 Simulator, http://www.isi.edu/nsnam/ns/."

Session 4 – Quality of Service

Chair: Eduardo Tovar

Rapporteur: Anis Koubâa

Session Summary: Quality of Service

Rapporteur: Anis Koubâa
IPP-HURRAY Research Group, Polytechnic Institute of Porto
Porto, Portugal
akoubaa@dei.isep.ipp.pt

1. Introduction

Quality of Service (QoS) is one of the major issues in the dimensioning of industrial networks and embedded systems, particularly when timing constraints need to be met. Roughly, the provision of QoS in Real-Time Networks (RTNs) means the specification and the set-up of the set of mechanisms necessary to meet the QoS constraints. The main QoS constraint that must be satisfied in RTNs is the message delay (or response time).

In that direction, The QoS session of the Real-Time Networks Workshop has focused on proposing new mechanisms for supporting and analyzing real-time QoS in industrial and home networks.

2. Talks

Motivated by the inefficiency of COTS Ethernet switches to guarantee real-time communication, the first paper of this session dealt with the proposal of a synchronized approach for ensuring deterministic real-time guarantees in industrial switched Ethernet based on Flexible Time Triggered (FTT) paradigm. The architecture of the FTT-SE (FTT- Switched Ethernet) is based on the master-slave model according to which the master polls its slaves periodically and the communication occurs during a time unit referred to as Elementary Cycle. It has been shown that the FTT-SE approach enables a noticeable improvement of timing constraints in switched Ethernet, but at the cost of a higher implementation complexity.

The second paper of this session was focused on the ability of Universal Plug and Play (UPnP) protocols to provide real-time guarantees in IP-based home networks. This paper proposed an abstract model that enables an efficient QoS management without having to know all underlying details in lower layers.
The last paper of this session proposed a priority based approach that facilitates the integration of several CAN-based subsystems. The basic idea of the paper was to decouple CAN identifiers from their priorities to avoid any kind of conflicts when interconnecting several subsystems together. A comparative time-predictability performance analysis of different decoupling protocols (FTT-CAN, TT-CAN and Server-CAN) has also been discussed.

Enhanced Ethernet Switching for Flexible Hard Real-Time Communication

Ricardo Marau, Paulo Pedreiras, Luís Almeida
DET/IEETA, Universidade de Aveiro, Portugal
{marau,pedreiras,lda}@det.ua.pt

Abstract

Switched Ethernet arose in the last decade as a means to increase global throughput with parallel switching paths, segment the network and create isolated collision domains, thus reducing the non-determinism of the original shared Ethernet. However the services provided by COTS Ethernet switches are not enough to guarantee real-time communication, which lead to the development of several switch Ethernet-based protocols, among which the recently proposed FTT-SE. This paper proposes moving the FTT traffic management into the Ethernet switch and discusses how this architectural change enhances the performance of the transmission control and service differentiation mechanisms as well as how error confinement mechanisms can be efficiently deployed. Preliminary experimental results from a prototype implementation validate the services provided by the enhanced Ethernet switch framework.

1 Introduction

Distributed Embedded Systems (DES) integrating intelligent cooperative nodes are found in a wide range of applications, from automotive to aerospace, passing through the lower layers of both process control and manufacturing industries [1]. In these environments, applications range from embedded command and control systems to image processing, monitoring, human-machine interfacing, etc.

Since its creation, Ethernet has been considered has a potential solution for use in DES due to its large bandwidth, cheap silicon, high availability, easy integration with Internet and clear path for future expandability [2]. Furthermore, using Ethernet also at the lower control level facilitates the vertical integration and may bring along several advantages in maintenance effort.

Ethernet, however, is a general purpose data network and was not originally designed to satisfy the requirements of DES. For this reason several modifications have been proposed, including restrictions to the traffic pattern generated by each node, modifications to the arbitration mechanism and addition of transmission control layers [9].

Since the early 90's the interest in switched Ethernet has been growing steadily, having practically replaced shared-Ethernet (single segment, hub-based). Despite avoiding message collisions and having built-in traffic scheduling capabilities (Figure 1), thus improving the predictability of the network with regard to shared Ethernet, in general switched Ethernet networks are not capable of delivering the real-time communication services needed by DES. Therefore, as for shared Ethernet, several techniques were proposed to overcome its limitations, from shaping the traffic submitted to the switch to limiting that traffic by application design, providing more efficient scheduling policies and admission control or adding transmission control features [9].

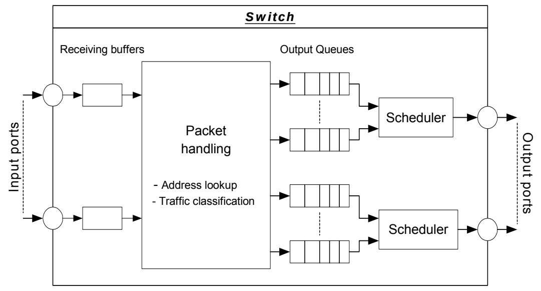


Figure 1: Internal switch architecture

The recently proposed FTT-SE [3] belongs to this latter category of protocols, and exhibits, as main features, global traffic coordination in a common timeline, the possibility for fast and atomic on-line updates to the set of streams, the possibility to support wide ranges of streams periods and the possibility to enforce any traffic scheduling policy.

While using non-standard hardware conflicts with some of the key arguments supporting the use of Ethernet in real-time applications (e.g. cost, availability, compatibility with general purpose LANs), custom switch implementations with enhanced traffic control and scheduling capabilities allows important performance and service breakthroughs, and so a number of approaches of this class have also been proposed in the recent years (e.g. [4], [6] [8]).

This paper proposes integrating the traffic management and transmission control mechanisms of the FTT-SE in an Ethernet switch. The resulting framework allows obtaining important performance gains in the following key aspects:

- A noticeable reduction in the switching latency jitter found in common Ethernet switches;
- An important performance boost of the asynchronous traffic, which in this case is autonomously triggered by the nodes instead of being pooled by the master node;
- An increase in the system integrity since unauthorized transmissions can be readily blocked at the switch input ports, thus not interfering with the rest of the system;
- Seamless integration of standard non FTT compliant nodes without jeopardizing the real-time services.

In the next section the FTT-SE protocol is briefly reviewed. Section 3 presents the architecture of the enhanced

The material in this paper is the subject of a current patent filing.

Ethernet switch. Section 4 describes a prototype implementation and presents some preliminary experimental results. Section 5 concludes the paper.

2 FTT-SE brief overview

FTT-SE is a recently proposed COTS-based real-time protocol [3] for micro-segmented switched Ethernet networks. The FTT-SE protocol is based on the Flexible Time-Triggered (FTT) paradigm and supports arbitrary traffic scheduling policies, periodic and sporadic traffic with temporal isolation, priority levels beyond the eight levels specified in IEEE 802.3D, on-line admission control and bandwidth management and, finally, completely avoids memory overflows inside the switch due to the global traffic scheduling mechanism.

2.1 FTT-SE Medium Access Control layer

The FTT-SE employs a technique called master/multi-slave, according to which the master addresses several slaves with a single poll, considerably alleviating the protocol overhead with regard to the conventional master-slave techniques. The communication occurs in fixed duration slots called Elementary Cycles (ECs), with one master message per cycle called Trigger Message (TM), which contains the periodic schedule for that EC. The periodic messages are referred to as synchronous since their transmission is synchronized with the periodic traffic scheduler. The protocol also supports aperiodic traffic, called asynchronous, which is managed in the background, in the time left within the EC, after the periodic traffic (Figure 2).

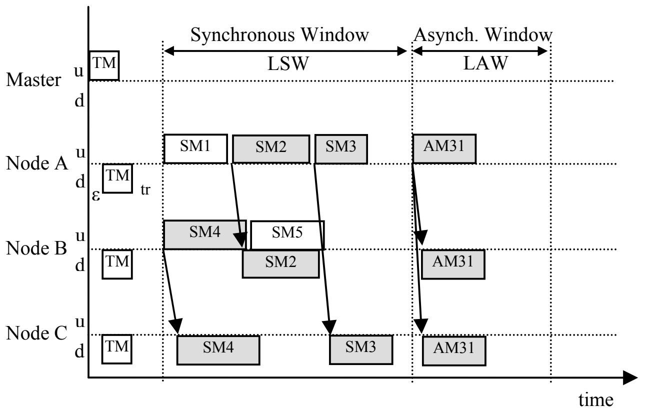


Figure 2: Traffic scheduling in FTT-SE

2.2 Synchronous traffic scheduling

The synchronous traffic scheduling activity is carried out on-line and centrally in the master and the periodic traffic schedules are disseminated by means of the TM (Figure 2). Since the traffic scheduling is local to one node, it is easy to enforce any kind of scheduling policy, as well as perform atomic changes in the communication requirements. This last feature allows for on-line stream admission and removal under guaranteed timeliness as well as on-line bandwidth management. Nodes decode the TM and transmit immediately the scheduled messages with the switch taking care of the serialization. All messages scheduled to one EC fit in that EC and so message queues have a limited and pre-known size and cannot build up from EC to EC.

The FTT master holds information about the nature of the data exchanges regarding the type of addressing (unicast, multicast and broadcast) and which end nodes are involved. With this information the master computes which messages follow disjoint paths (i.e., non overlapping source and destination nodes) and thus build schedules that exploit this parallelism, increasing the aggregated throughput. For non-multicast switches only unicast and broadcast streams can be considered. For true multicast switches the standard Internet Group Multicast Protocol (IGMP, RFC 2236) is used to setup up multicast groups.

2.3 Asynchronous traffic handling

Unconstrained aperiodic communication may generate bursts that fill in output queues, leading to long priority inversions in typical FIFO queues and possibly to queue overflow and consequent packet losses. Using switches with two (or more) priority levels and assigning to the asynchronous traffic a lower priority level than to the synchronous one does alleviate the problem. Nevertheless, due to the non-preemptive nature of packet transmission asynchronous messages can still block the synchronous messages or the TM. The blocking effect is, however, bounded to one packet. Adequate mechanisms are still required to constrain the asynchronous load and burstiness to prevent buffer overflows and consequent interference with the high priority periodic traffic [5][7]. Therefore, the use of traffic shaping or smoothing schemes is required.

Alternatively, polling can be used, being more robust and timely but less efficient. In this case, the transmission instants are adequately planned by the global scheduler but synchronization delays will increase the response times. In this case the asynchronous traffic is treated essentially as the synchronous one, except that slaves may or may not transmit a pooled message, depending on its readiness status.

FTT-SE can use any of the mechanisms above, depending on the requirements of each application. The polling approach is more adequate for situations requiring precise timeliness. When the non-preemption blocking is tolerable, the dual-priority approach seems better suited.

3 FTT enabled Ethernet switch architecture

Figure 3 depicts the FTT enabled Ethernet switch integrating the traffic management services provided by the Master node in FTT-SE systems. The System Requirements Database (SRDB) is the central repository for all the information related to the traffic management, namely the message attributes for both synchronous and asynchronous traffic (e.g. period/minimum inter-arrival time, length, priority, deadline), information about the resources allocated to each traffic class (e.g. phase durations, maximum amount of buffer memory) and global configuration information (e.g. elementary cycle duration, data rate). Change requests to the message set are submitted to an admission control (plus optional QoS manager), ensuring continued real-time traffic timeliness. The SRDB is periodically scanned by a scheduler, which builds a list of synchronous messages (EC-Schedule) that should be produced in the following EC. A dispatcher task periodically sends the EC-schedule to the switch ports having attached FTT nodes.

For FTT-SE/FTT-Ethernet systems the master role is confined to the functionalities defined above. However, the integration of the switching services with the traffic scheduling permits a tight control of the packet flow and resource utilization inside the switch. At the beginning of each EC the global dispatcher directly accesses the port dispatcher, which sends the trigger message and keeps temporal information about each of the phases within the EC. Each output port has 3 queues, one for each traffic class (synchronous, asynchronous and non real-time messages (NRT)). During the EC the port dispatcher transmits messages submitted to each of these queues, according to the EC phase. This mechanism confines the different traffic classes to the respective phases. If e.g. a malfunction node sends a synchronous message outside of the synchronous phase, the message is discarded and does not interfere with the asynchronous or non real-time phases. On the other hand asynchronous messages (either real-time or non real-time) do not need to be pooled, contrarily to what happened for FTT-SE. The port dispatcher only transmits messages from the asynchronous or NRT queues if the time left within the respective window is enough.

Both FTT and non FTT-compliant nodes can be seamlessly attached to the FTT enabled switch. Thus, on the ingress side the first operation carried out is the packet classification, which consists only in inspecting the Ethernet type field. When the message is identified as an FTT message it is subject to a verification process and, if judged valid, is appended to the synchronous or asynchronous message queues, according to its nature. Conversely, if the message is non-FTT it is simply appended to the NRT queue. The segmentation of the global memory pool, keeping the messages of each class in independent subdivisions allows avoiding memory exhaustion for the real-time messages, a situation that standard switches do not guarantee [5]. The real-time traffic is subject to an explicit registration. During the registration process the producers must state the message properties, in particular the length and periodicity (for periodic messages; minimum inter-arrival time for sporadic ones). With this data it is possible to compute and pre-allocate the amount of memory that each traffic class requires and thus guarantee that the resources are enough for all admitted messages. These elements, however, are not available for the NRT traffic. Thus it is not possible to predict the amount of memory necessary and, consequently, the NRT queue may become full, leading to drops of NRT packets. However, the higher layers protocols (e.g. TCP) are tolerant to occasional message drops, only with a negative impact in the performance. This situation is not critical since this traffic is granted with best effort guarantees, only.

The validation process gathers data both from the EC-schedule and from the RTDB. Regarding synchronous messages, the analysis of the EC-schedule allows detecting failures in the time domain, namely the transmission of unscheduled messages or the late transmission of scheduled messages resulting from mal-function nodes. An equivalent set of tests (e.g. minimum inter-arrival time, burstiness) may also be performed for asynchronous messages with those that fail the validation process being trashed. The policing and enforcement of the traffic attributes in the time domain guarantees the timeliness of the real-time traffic even in the presence of malfunctioning nodes.

Whenever a message is placed in the global memory pool, a packet forwarding process is executed. Control messages, targeted to the master are submitted to the Admission control/QoS manager module and possibly result in changes on the SRDB. Data messages should be forwarded to the target nodes. The forwarding mechanism of FTT messages is based on a producer-consumer model, and does not depend on MAC addresses. Whenever an FTT message arrives the Packet Forwarding module inquires the SRDB to determine the set of ports having consumers attached, and updates the output queue (synchronous or asynchronous, depending on the message nature) of each one of these ports. Non-FTT messages are forwarded according to the normal procedures of standard Ethernet switches, based on the MAC address.

4 Experimental results

A prototype implementation, based on the RT-Linux real-time operating system with the Ethernet layer provided by the LNet network stack, was carried out to validate the extended services provided by the FTT-enabled switch. This prototype switch is based on a Pentium III PC at 550MHz with four 3Com 3C905B PCI network interface cards

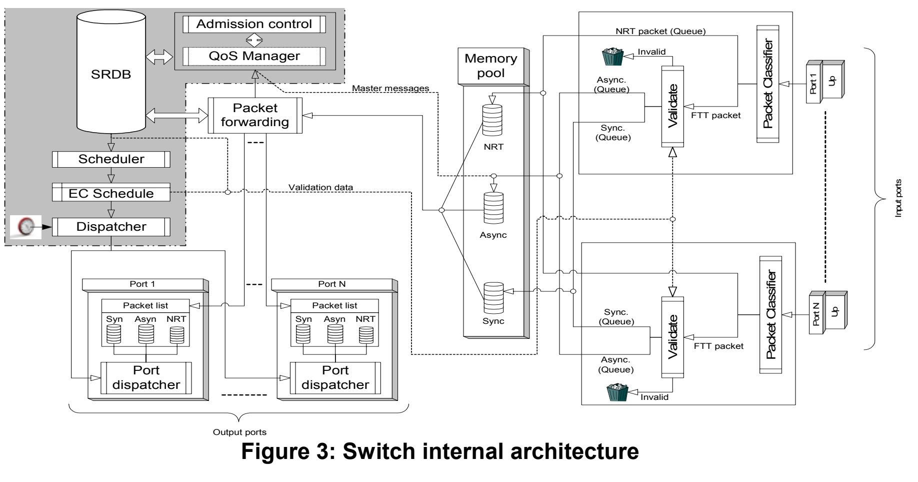


Figure 3: Switch internal architecture

The first experiment consists in the implementation of a policing service for the synchronous traffic. The ID of incoming synchronous messages is matched against the EC-schedule and discarded if a positive match is not found. This way only scheduled messages are disseminated, guaranteeing that the synchronous window is not overrun. To verify the correct behavior of the policing service we configured a setup with 1 synchronous message with period Ti=3ECs while the respective producer slave was tampered to send that message every EC. With the setup we observed that the consumer node only received the scheduled messages, one every 3ECs, and the extra messages were discarded.

The second experiment consists in the verification of the enforcement of the traffic temporal isolation. The experimental setup is configured with an EC of 40ms, with the last 3ms of the EC dedicated to the NRT traffic. The NRT test load consists in UDP packets carrying 1400 data bytes, periodically sent every 5ms. The load is generated with PackEth (http://packeth.sourceforge.net/) running on a plain Linux distribution (RedHat 9.0). Figure 4 depicts the histogram of the time differences between consecutive NRT messages in the uplink.

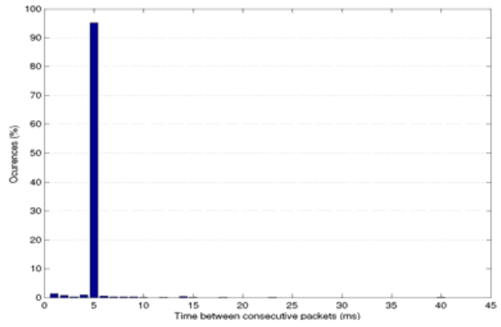


Figure 4: Histogram of the differences between consecutive NRT messages (uplink)

While NRT messages can be submitted at any time instant, the FTT-enabled switch only forwards them to the output port(s) in the NRT window, which in this setup is configured to use the last 3ms of the EC. This confinement mechanism significantly changes the message transmission pattern between the uplink and the downlink. Figure 5 shows the time difference between the beginning of the EC and the reception of the NRT messages being clear the confinement of these to the NRT window (37 to 40ms after the EC start). Therefore the NRT traffic does not interfere with the synchronous or asynchronous real-time traffic, despite being generated at arbitrary time instants by a standard node not implementing the FTT protocol.

5 Conclusions and future work

The advent of switched Ethernet has opened new perspectives for real-time communication over Ethernet. However, a few problems subsist related with queue management policies, queue overflows and limited priority support. While several techniques were proposed to overcome such difficulties, the use of standard Ethernet switches constraints the level of performance that may be achieved. In this paper we proposed an enhanced Ethernet switch, implementing FTT-class services. The resulting architecture inherits the FTT features, namely flexible communication with high level of control to guarantee timeliness, while permits a noticeable reduction in the switching latency jitter found in common Ethernet switches, an important performance increase of the asynchronous traffic, seamless integration of standard Ethernet nodes and a substantial increase in the system integrity as unauthorized transmissions from the nodes can be readily blocked at the switch input ports. On-going work addresses the FPGA implementation of the switch.

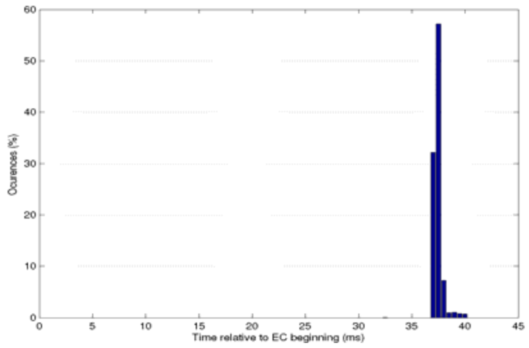


Figure 5: Histogram of the differences between the beginning of the EC and NRT messages (downlink)

6 References

[1] Thomesse, J-P. "Fieldbus and Interoperability". Control Engeneering Practice, 7(1), pp81-94. 1999.

[2] Decotignie, J-D. A perspective on Ethernet as a Fieldbus. Proc of the FeT'2001 – 4th Int. Conf. on Fieldbus Systems and their Applications, pp138-143. Nancy, France. November 2001.

[3] Almeida, L., Pedreiras, P. "Hard Real-Time Communication over COTS Ethernet Switches". " Work in Progress Session of the 26th IEEE International Real-Time Systems Symposium.

[4] Hoang, H. Jonsson, M., Hagstrom, U., Kallerdahl, A. "Switched Real-Time Ethernet with Earliest Deadline First Scheduling - Protocols and Traffic Handling". Proc of WPDRTS 2002, the 10th Intl. Workshop on Parallel and Distributed Real-Time Systems. Fort Lauderdale, Florida, USA. April 2002.

[5] Pedreiras, P., R. Leite, L. Almeida. Characterizing the Real-Time Behavior of Prioritized Switched-Ethernet. RTLIA'03, 2nd Workshop on Real-Time LANs in the Internet Age, (satellite of ECRTS'03), Porto, Portugal, July 2003.

[6] Varadarajan, S., Chiueh, T. "EtheReal: A Host-Transparent Real-Time Fast Ethernet Switch". Proc of the 6th Int Conference on Network Protocols, pp. 12-21. Austin, USA. Oct 1998.

[7] Loeser, J., H. Haertig. "Using Switched Ethernet for Hard Real-Time Communication". Proc Parallel Computing in Electrical Engineering, International Conference on (PARELEC'04), pp. 349-353, September 07 - 10, 2004, Dresden, Germany.

[8] Real-Time PROFINET IRT. http://us.profibus.com/profinet/07

[9] P. Pedreiras, L. Almeida. Approaches to Enforce Real-Time Behavior in Ethernet. in *The Industrial Communication Systems Handbook*, R. Zurawski (ed). CRC Press, ISBN:0-8493-3077-7, 2005.

The network capability model of UPnP™-QoS v3, an interoperable QoS framework for admission control and scheduled access

Michael van Hartskamp[1]

Philips Research, High Tech Campus 34, 5656 AE Eindhoven, The Netherlands

***Abstract* — Home networks are increasing in popularity. For commercial content offerings, a high guaranteed Quality of Service is needed. The heterogeneous nature of the IP-based home network complicates delivering real-time guarantees. In this paper we describe the network capability model of a proposed interoperable middleware for QoS control on the basis of UPnP [1]. The model describes QoS capabilities of underlying Layer 2 QoS technologies. It enables QoS Managers to manage the network and understand the impact of their management actions, without explicit understanding of the details of every technology.**

***Key words* — Home Networks, Interoperability, Quality of Service, UPnP, UPnP-QoS.**

I. Introduction

Home networks are increasing in popularity. More and more households have (wireless) Ethernet-based home networks connecting their PC(s) or laptop(s) to the Internet. Currently applications such as IPTV and VoIP are entering the home, yet they are often terminated at the door step and do not take the final step through the home network. For most service providers, the lack of real-time or Quality of Service guarantees inhibit mass-market deployment.

While for commercial content offerings, a high guaranteed Quality of Service is needed, the currently deployed technologies, such as the IP-protocol suite, Ethernet and Wi-Fi / Wireless Ethernet, do not go beyond 1) best effort delivery and 2) an attempt at fairness to all network users. In other words they do not provide the required real-time guarantees. Future Layer 2 technologies, which are currently or were recently standardized, such as IEEE 802.11e, WiNET, and HomePlug AV do provide mechanisms for admission control and scheduled delivery of packets, bringing support for real-time. What is still lacking for an actual deployment is a standardized middleware that provides applications with a uniform interface to use those features of the different underlying technologies in heterogeneous networks.

In this paper we contribute a network capability model. The network capability model indicates for heterogeneous networks, the devices of which have certain QoS capabilities. This model is proposed as part of an interoperable middleware for QoS control on the basis of UPnP [1]. UPnP is the de facto standard for discovery and description of home networking devices. The UPnP forum has also defined device control protocols for various devices such as AV media servers and AV media renderers. UPnP is also the backbone for the standards and guidelines of DLNA [2].

Our approach is based on the assumption that admission control is essential to guard a sufficient minimum level of Quality of Service by preventing structural overloading of the network. It is understood, that admission control is a necessary but not sufficient condition. In wireless or power line networks, there are no hard guarantees and hence point-to-point solutions have to be applied to maintain an acceptable quality in the presence of varying resource availability. These techniques are not discussed in this paper.

The remainder of this paper is structured as follows. In Section II we provide a short overview of the home network and in particular UPnP. The following section describes some of the relevant details and workings of IEEE 802.11e in more detail.

Section V presents an overview of our solution. The solution is based on a network capability model. The model is proposed for inclusion in UPnP-QoS version 3. It allows the discovery of layer 2 capabilities of certain devices. This enables the end-to-end QoS setup.

The following section details the behavior of the involved devices by describing the interaction and the relation with Layer 2 setup.

Section VII shortly discusses how the proposed solution enables decomposition of end-to-end requirements such as delay and loss requirements. With our solution it is possible to avoid local optimizations.

Finally we draw our conclusions. The network capability model is a suitable way to describe and manage the QoS capabilities present in current and future home networks. The submission to UPnP-QoS enables applications to manage QoS in heterogeneous networks in a standardized way.

II. Home Networking

A. Home Networking Architectures

The DLNA guidelines provide a model for logically describing Home Networking AV devices (see [3]). The DLNA devices are assumed to be used for applications such as AV streaming of various quality, media uploads and downloads, etc.

The DLNA model is based on, what are at least traditionally, standards from the IT-world: Ethernet, IP,

[1] Parts of the work reported here were performed within the ongoing European Research Program BETSY- IST-004042.

HTTP; protocols that are not traditionally known for their real-time capabilities. But most of them so well established that their unmodified adoption is essential and their limitations need to be circumvented.

For discovery, description, and control, the DLNA guidelines use the UPnP protocol.

B. UPnP

In UPnP, a Control Point invokes actions on a UPnP service which is running on a UPnP device. The UPnP device advertises one or more services with certain (often standardized) actions. The Control Point, whose behavior is not standardized, determines capabilities of a device (service) and then decides what the device (service) will do. The UPnP forum has already standardized various services and devices for applications ranging from home automation, via gateways to AV-applications.

Since 2005, services enabling priority-based QoS are defined. Through UPnP an application interfaces with Layer 2 (or 3) technologies for QoS, by-passing the IP-layer [4].

Currently in UPnP, the QoS working committee is extending these services to support admission control and scheduled access, i.e. parameterized QoS. In this paper we describe our network capability model which is proposed as a part of these extensions.

III. IEEE 802.11E

For the heterogeneous home network, it is crucial to work on the basis of different existing layer 2 technologies. In this section we provide background on a popular Layer 2 technology: Wi-Fi WMM and the underlying IEEE 802.11e [5] to motivate our network capability model.

WMM is a certification program of the Wi-Fi alliance on the basis of IEEE 802.11e, prioritized QoS. It also offers a simple admission control functionality. The complete IEEE 802.11e also specifies scheduled access and it is expected that this will also become part of a Wi-Fi certification program. In the remainder we consider the scheduled access function “HCCA” of IEEE 802.11e.

In IEEE 802.11e, a wireless QoS enabled station (QSTA) connects to the QoS-enabled Access Point (QAP). Two stations connected to the same AP communicate via the AP. With HCCA the QAP polls a QSTA after which the QSTA may transmit a packet.

Requests for QoS are always initiated from a QSTA, whether the station will be sending or receiving. The requests present a traffic specification consisting of among many others mean data rate, peak data rate, delay bound, and minimum PHY rate to the QAP which subsequently decides on the viability. Typically the QAP has an overview of all admitted streams, but individual QSTA(s) do not.

To save bandwidth, a direct link protocol enables direct transmission between two QSTA, by-passing the QAP. In this case, the sending station is responsible for the QoS request.

IV. MAJOR DESIGN ALTERNATIVES

In this paper, we follow a centralized approach for QoS management. The basic idea behind the centralized approach is that a QoS request is forwarded to a central entity. This entity decomposes end-to-end requirements and subsequently appropriately instructs the individual devices. Another example of a centralized approach is in [8] and it shows how real-time requirements can be met in a heterogeneous environment.

We believe this centralized solution is possible given the small size of a typical home network. When meeting end-to-end requirements we do not have to suffer from local optimizations. But to enable the central controller to make such “wise” decisions, some information has to flow from the individual devices to the controller in order to support its decisions.

This is different from an RSVP-style approach (see [6]) where the receiver sends a QoS request to the sender. On its way to the sender it passes devices. Every device determines whether it can support the request. If not the request is denied and returned to the receiver. When the request reaches the source and is accepted, a positive acknowledgment is returned. Some provisions have been foreseen for shared media [7] but these are not really used. End-to-end requirements are decomposed at every intermediate device by tightening the requirements for the others upstream.

There are two prime reasons for us to choose a centralized approach. First, the central approach is in line with the UPnP device architecture where a Control Point instructs a service on a device to do something. Secondly, a centralized approach, at least theoretically, allows better decompositions of end-to-end (real-time) requirements which a per-hop approach does not bring.

V. THE NETWORK CAPABILITY MODEL

In this section we describe our network capability model. Our approach is based on UPnP and we follow the design principles of UPnP by employing discovery and description in this case to discovery and description of QoS capabilities. In this paper we describe the network capability model which we have proposed to version 3 of the QosDevice service. For space reasons we can only describe one mapping of the model to a layer 2 technology, which is on IEEE 802.11e.

The goal of the model is enabling end-to-end requirements decomposition into, commonly called, “per-hop requirements” and appropriately configuring the network such that those “per-hop requirements” can be met. For this the network capability model needs to define those “hops” and indicate which devices have the capabilities to manage “which hop”.

In our approach the “per-hop” concept is formalized by the concept of QoS segment to accommodate Layer 2 technologies such as AV Bridges [9] that come with their own QoS management spanning multiple “hops”.

The basic QoS capabilities are admission of a stream and the release of resources. Another capability is to list the admitted streams to get an impression of the occupied resources. Since the model was derived with various actual implementations and technologies in mind, the capabilities to admit and release are often capabilities to perform Layer 2 signaling that leads to admission or release.

One of the crucial steps to take is to identify which device has the QoS capability to admit a certain stream in a certain QoS segment. The QoS capabilities are often topologically qualified, i.e., certain technologies or implementations offer in a certain device only admission for a specific link or even just for the outgoing direction, etc.

In the next section we go into the details of the model, provide and provide an example.

A. Segmentation

First consider the network as a graph consisting of vertices and edges. Since network connections are not necessarily bidirectional, edges are directed. The example of the wireless network where traffic can flow through the AP as well as through a Direct Link indicates that even with the Layer 2 spanning tree protocol the graph is not necessarily loop-free.

The goal of the segmentation is to capture the extent to which middleware actions induce Layer 2 signaling. Those middleware actions are the UPnP-actions invoked on a UPnP device by a UPnP Control Point called QoS Management Entity. For the middleware management to work some independence between the actions is needed. I.e., the middleware actions must not have side effects and preferably not unnecessary restrict the order of invocation.

We now define a QoS segment in an abstract way as a unit of independent management.

The collection of all QoS segments is closed under finite union and finite intersection. It is a cover of the graph. It is easy to see that there is a subcover such that every edge is contained in exactly one element of this subcover. The minimal QoS segments containing at least one edge typically coincide with the Layer 2 domains. In every actual deployment, such a subcover is determined through technology specific rules followed by the individual devices.

As a general principle, the smaller the QoS segments in the home network the more management steps are performed by a QoS Management Entity.

B. QoS capabilities

The three primary QoS capabilities are: Admit, Release, and List. When a device has the capability to admit it is capable to perform an admission control function as well as to reserve the resources (on networks governed by scheduled access).

There are different ways in which a device can support the capability to admit. A straight forward method is for a device such as a wireless QAP or another device with the Layer 2 scheduler. Such a device could (theoretically) easily perform algorithms for admission control and calculate appropriate schedulers. However, in many actual implementations this capability cannot be used directly. A method which is supported with (nearly) every technology is to rely on Layer 2 signaling. A device implements the capability to admit whenever it has the ability to use Layer 2 signaling to perform admission control. In this case the Layer 2 signaling protocol is limiting. The request may support only a limited set of parameters (e.g. only peak and not average bandwidth), return a limited answer (yes, no, but not "no, but you can admit x bits per second") and most frequently only for a limited number of streams, e.g., only for streams that flow through the device. In some Layer 2 technologies the latter limitation is not there and similarly general results as with admission at the scheduler can be obtained.

The capability to Release is similar and frees the assigned network resources. The capability to List enables the listing of streams that were assigned resources. This capability makes most sense when available at the device with the scheduler. It allows a QoS Management Entity to determine the viability of an admission before actually making it through UPnP-QoS and induced Layer 2 signaling.

C. Topological Qualifiers

As indicated by the examples most QoS capabilities are not general but qualified. A device can expose a limited capability, e.g., due to layer 2 signaling limitations. The following topological qualifiers are identified in the model.

QoS segment for an Interface: this device offers the QoS capability for every stream on the entire QoS segment for the given interface. A possible device to expose this capability is on the QAP for the QoS segment of the Access point and its associated stations. But see the practical limitations above.

Link: this device offers the QoS capability for the specified link. An example can again be found in IEEE 802.11e. A station (QSTA) identifies a link to the AP and the station has the capability to set up QoS for that link through the Layer 2 signaling. The Layer 2 signaling does not enable a station to set up QoS both for the link to the QAP and onwards to a next station. As an example, a QoS Management Entity will read the model and derive that this station can set up QoS from the station to the QAP but that it needs another action to set up QoS from the QAP to the next station. The fact that both links share the same underlying network medium is (weakly) expressed through the fact that both links are in the same QoS segment.

Direction: this device offers the QoS capability only for the specified direction of the identified link. Consider the Direct Link example. Here the device exposes the capability to admit only for the outgoing direction of the direct link.

D. Example of reported QoS capabilities

In this section we provide a simple example. Consider Figure 1 below representing a home with 6 wireless stations in 2 wireless networks connected through a wired backbone. There are three "minimal" QoS segments S, T, and U.

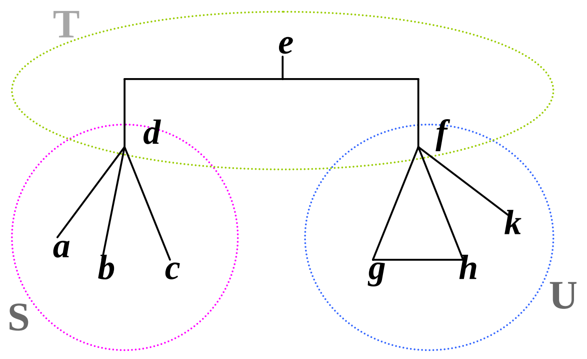


Figure 1 Example home network with QOS capabilities

We will first describe the reported QoS capabilities and then in the following section how to make use of these capabilities.

In segment S, device *d* offers the QoS capabilities to admit, release and list for the entire segment. Device *a* offers the admit and release capabilities for the link $a \leftrightarrow d$, *b* for $b \leftrightarrow d$, and *c* for $c \leftrightarrow d$, where $x \leftrightarrow y$ denotes the link (in both directions) between *x* and *y*.

In segment T, no QoS capabilities are available, e.g. with plain Ethernet. There can be no QoS management.

In segment U, device *f* offers the QoS capability to list. *g* offers the admit and release capabilities for links $g \leftrightarrow f$ and $g \rightarrow h$ (i.e. only in the direction from *g* to *h*). Similarly device *h* offers it for the links $h \leftrightarrow f$ and $h \rightarrow g$. But *k* only offers these capabilities for $k \leftrightarrow f$.

VI. USE OF THE MODEL

A device interested in participating in QoS setup hosts the UPnP QosDevice service. For every interface, the device determines the QoS segment in which it participates. The QoS segment is identified through an agreed Layer 2 specific algorithm: typically the identity of an elected or pre-determined leader is used, e.g., in wireless the MAC address of the AP or in IEEE 1394 or AV bridges the "root".

Next this QosDevice service reports its QoS capabilities. The topological qualifications are expressed via a hierarchically ordered structure (QoS segment, link, direction(s)) to avoid unnecessary repetitions.

Let us now follow the steps of performing an end-to-end admission for a stream flowing from *a* to *h* in the figure.

A QoS Management Entity is a UPnP control point for the (standardized) QosDevice service. The UPnP functionality is used to identify the available services. Through actions on the QosDevice service and comparison of the QoS segment IDs the QoS Management Entity determines the QoS segmentation of the network. This is also illustrated in the example above where the QoS Management Entity has now identified segments S, T, and U and may safely conclude that setup actions in segment S will not impact segment T or U (and similarly for segment T and U)

Also the path of the stream for which QoS is requested is determined. The QosDevice (since version 1) provides information to assist in the path determination.

Now the QoS Management Entity proceeds to setup QoS for every segment. For a given segment, the following steps are performed.

First, identify whether a QosDevice in the segment advertises the QoS capability to admit for the entire segment. If so (for example at device *d* in segment S), it invokes the admit action on this QosDevice and the process is completed. If not (for example in segment U), for every link in the QoS segment through which the stream passes, it identifies whether there is a QosDevice service which advertises the QoS capability to admit for that link (and in the desired direction) and admit at this device. In our example the link $f \leftrightarrow h$ is a relevant link and *h* offers the capability to admit.

After actually performing the admission request on the individual devices, device *d* probably performs only some internal calculations towards a new schedule. Device *h* on the other hand needs to rely on layer 2 signaling to *f* to effectuate its capability to admit. Device *f* will execute such signaling and then report the result to the QoS Management Entity.

Observe that if the stream passes through multiple links in the QoS segment, every link has to be individually set up through an admission action (cf. the example of IEEE 802.11e).

Finally, a QosDevice that receives the admit action registers the request and either performs the admission itself and updates its scheduler's polling tables, or performs a Layer 2 specific request which ensures schedules are setup and/or admission is evaluated.

VII. SOME REMARKS ON END-2-END REQUIREMENTS

The network capability model as described here indicates *whether* devices have the QoS capability to admit. This model does not describe how the admission request has to be formulated if such a QoS capability exists. For the latter a standard traffic specification needs to be used which lists several parameters which are commonly used in current Layer 2 technologies: such as mean bandwidth and peak bandwidth, but also other parameters relating to loss and delay requirements.

We believe the approach we have taken, i.e. enabling a QoS Management Entity to manage the end-to-end QoS setup allows avoiding local optimizations. It is expected that through additional interaction with the QosDevice service, local preferences on the decomposition are brought to the attention of the QoS Management Entity.

VIII. Conclusions

This paper describes a network capability model for discovery and description of Quality of Service capabilities of devices. The model is rich enough to capture common state-of-the art link-layer QoS technologies such as IEEE 802.11e (others such as HomePlug AV, MoCA and AVB were verified outside this paper), yet sufficiently abstract to enable efficient QoS management without awareness of all details of every underlying Layer 2 QoS technology. With knowledge of the model a QoS management entity can perform end-to-end admission control on the home network where needed by relying on layer 2 signaling and understanding interdependencies between setup of parts of the network.

The model also facilitates the decomposition of end-to-end requirements in a centralized manner, avoiding sub-optimal decisions as are possible in e.g. RSVP.

The model is part of UPnP-QoS. In this way applications are offered an interoperable mechanism for their QoS management.

Acknowledgement

The author wishes to acknowledge the members of the UPnP-QoS working committee and in particular Daryl Hlasny (Sharp), Ally Yu-kyoung Song (LGE), and Puneet Sharma (HP) for many interesting and fruitful discussions that lead to the described model.

The author also wishes to thank Gerhard Fohler for his stimulating knowledge on real-time and IP.

Finally the author wishes to thank Peter van der Stok for a long-standing cooperation on QoS in general and his remarks and help to improve this paper in particular.

References

[1] UPnP, UPnP Device Architecture 1.0, available on www.upnp.org
[2] Digital Living Network Alliance, www.dlna.org
[3] DLNA Press presentation march 2006, www.dlna.org
[4] UPnP-QoS QosDevice service definition, 2005, available on www.upnp.org
[5] IEEE, “IEEE 802.11e,” *IEEE Amendment*, 2005.
[6] R. Braden, ed., RFC 2205, *Resource ReSerVation Protocol (RSVP)*, 1997
[7] R. Yavatkar, et al., RFC 2814, *SBM (Subnet Bandwidth Manager): A Protocol for RSVP-based Admission Control over IEEE 802-style networks*, 2000
[8] L. Rizvanovic and G. Fohler, *The MATRIX: A QoS Framework for Streaming in Heterogeneous Systems*, International Workshop on Real-Time for Multimedia (RTMM), 2004
[9] M. Johas Teener, AV Bridge summary, http://www.ieee802.org /1/files/public/docs2006/resb-mjt-bridge-summary-060111.pdf

This page left intentionally blank

Facilitating subsystem integration by decoupling priority and identifier in CAN messages

Thomas Nolte†‡, Lucia Lo Bello‡
‡RETISNET Lab, Dept. of Computer Engineering and Telecommunications
University of Catania
Catania, ITALY

Hans A. Hansson †
†Mälardalen Real-Time Research Centre
Dept. of Computer Science and Electronics
Mälardalen University
Västerås, SWEDEN

Abstract

When integrating subsystems on a common shared communication infrastructure these subsystems are likely to suffer from, and introduce interference among, each other. For CAN-based systems, the CAN message identifier is especially important, as it not only does identify the message, but it also determines the message's priority. Hence, special care needs to be taken when assigning identifiers to messages. This paper outlines how CAN-based systems are engineered today, and indicates the potential and benefits of decoupling the message priority from the message identifier. Three solutions to this are existing today: TT-CAN, FTT-CAN and Server-CAN. In this paper their strengths and weaknesses in an integration context are discussed. Also, the flexibility offered by the solutions is compared.

1 Introduction

The Controller Area Network (CAN) [8] is one of the major network technologies used in many application domains requiring embedded communications. It is particularly important in the automotive domain. A typical CAN application is any type of embedded system with real-time requirements and cycle times of $5 - 50ms$. However, CAN is used for many non real-time applications as well.

Traditionally in many application domains, subsystems have been developed incrementally as new functionalities have been added to the system. Looking at the automotive domain, value- and safety-adding functions, such as ABS and VDC, have been introduced over the years. Initially, they could be integrated as (mostly) independent subsystems having their own dedicated hardware in terms of Electronic Control Units (ECUs) and communications network. However, as the number of subsystems increases, there are strong trends towards integration of the subsystems on a common distributed architecture, rather than using a separate architecture for each subsystem. Hence, a crucial issue to solve is the migration from federated systems to integrated systems [10].

Looking at CAN-based embedded systems, it is a natural consequence that subsystems affect each others temporal performance once they are integrated on the same CAN network. This is due to the characteristics of the CAN message identifier, which defines both message priority (CAN PRIO) and message identity (CAN ID). Hence, decoupling the CAN ID from the CAN PRIO has the potential to simplify the integration process of CAN-based systems, allowing for flexible usage of CAN IDs. In this paper, three techniques that decouple the CAN ID from the CAN PRIO are presented, and their strengths and weaknesses are discussed from an integration point of view. Also, their provided offline (e.g., at design time) and online (e.g., allowing for dynamic addition and removal of subsystems) flexibility is compared.

Note that, in a CAN-based system, IDs are required to be unique for two reasons (1) due to the CAN arbitration mechanism (i.e., message collision resolution) and (2) to allow for message filtering (i.e., message identification). This implies that not any two subsystems are allowed to send a message with the same CAN ID at the same time. If this would happen as a consequence when integrating subsystems, one of them must change its conflicting IDs.

However, by the usage of recent schedulers, such as TT-CAN [9], FTT-CAN [1] and Server-CAN [15, 16], that decouple CAN ID and CAN PRIO, this change of IDs would not affect the temporal performance of the subsystems, thus allowing for an easier integration.

The outline of this paper is as follows: In Section 2 the technical properties of CAN message frame transmission are explained, showing the role of the CAN ID. In Section 3 CAN ID assignment in practice is shown, followed by Section 4 presenting three CAN schedulers that decouple the

CAN ID from the CAN PRIO. Section 5 discusses the three schedulers in the context of integration. Finally, the paper is concluded in Section 6.

2 CAN technical properties

CAN is a broadcast bus, which uses deterministic collision resolution to control access to the bus (so-called Carrier Sense Multiple Access / Collision Resolution, CSMA/CR). CAN transmits messages in an event-triggered fashion using frames containing 0 to 8 bytes of payload data. These frames can be transmitted at speeds of 10 Kbps up to 1 Mbps.

2.1 Frame arbitration

The CAN ID is required to be unique, in the sense that two simultaneously active frames originating from different sources must have distinct CAN IDs. Depending on the CAN standard used, the CAN ID can be either 11 bit (standard format) or 29 bit (extended format). Besides identifying the frame, the CAN ID serves two purposes: (1) assigning a priority to the frame, and (2) enabling receivers to filter frames (identifying the contents of the frame).

The basis for the access mechanism is the electrical characteristics of a CAN bus. During arbitration, competing communication adapters simultaneously put their CAN IDs, one bit at the time, on the bus. Bit value “0” is the dominant value. Hence, if two or more communication adapters are transmitting bits at the same time, and if at least one communications adapter transmits a “0”, then the value of the bus will be “0”. By monitoring the resulting bus value, a communications adapter detects if there is a competing higher priority frame (i.e., a frame with a numerically lower CAN ID), and in such a case it stops transmission. Because CAN IDs are unique within the system, a communications adapter transmitting the last bit of the CAN ID without detecting a higher priority frame must be transmitting the highest priority active frame, and can start transmitting the body of the frame, i.e., following the CSMA/CR rule. CAN therefore behaves as a global priority-based queue[1], i.e., a fixed priority non pre-emptive system, since at all communication adapters (nodes) the message chosen during arbitration is always the active message with the highest priority. Globally, the message with the highest priority will always be selected for message transmission.

3 Assigning CAN IDs in practice

The CAN ID is often used for scheduling purposes to fulfil temporal requirements. Here, the CAN ID can be set manually (static) or according to some online algorithm (dynamic). Also, the CAN ID can be assigned to message frames by the usage of tools, allowing for a design of the CAN-based system on a higher level than on per-message basis. Finally, as specified by several standards, the CAN ID can be solely used for message identification purposes.

3.1 Static CAN ID used for scheduling

From a scheduling point of view, the most natural CAN ID assignment method is found in the context of priority-driven scheduling, where priorities are assigned to messages according to the Rate Monotonic policy [11]. This since Fixed Priority Scheduling (FPS) is the scheduling policy implemented by the CAN arbitration mechanism. Real-time analysis techniques have been presented to determine the schedulability of CAN message frames [21], and revisited in [2].

3.2 Dynamic CAN ID used for scheduling

Several methods for Dynamic Priority Scheduling (DPS) have been proposed for CAN. By manipulating the CAN ID online, and therefore dynamically changing the message priority, several approaches to mimic Earliest Deadline First (EDF) type of scheduling have been presented [6, 12, 22].

For example, the usage of a Mixed Traffic Scheduler (MTS) [22] attempts to achieve a high utilisation (like EDF). Using the MTS, the CAN IDs are manipulated online in order to reflect the current deadline of each message. Hence, since the CAN ID is dynamically changing, only a part of it can be used for actual identification of the CAN frame.

Looking at non real-time messages, a common way to send them on a CAN network is to allocate them message identifiers with lower priority than all real-time messages. In this way blocking of a real-time message by non real-time messages can be restricted to at most the duration of the transmission of one message. However, unwise message identifier assignment to non real-time messages could cause some of them to suffer from starvation. To provide Quality of Service (QoS) for non real-time messages several approaches have been presented [4, 13]. These approaches dynamically change message identifiers in a way preventing systematic penalisation of some specific messages.

In general, by dynamically manipulating the IDs of the CAN frames, these solutions all reduce the number of possible CAN IDs to be used by the system designers. This could be problematic, since it interferes with other design activities, and is even sometimes in conflict with adopted standards and recommendations [5, 19].

[1]This is true if “FullCAN” communication controllers are used, e.g., Intel 82527 and Microchip MCP2510.

3.3 Static CAN ID assigned by tools

The growing complexity of automotive networked systems has resulted in tools to assist the system designer in the CAN ID allocation process. To understand this complexity, consider [15] where the network architectures for three cars are given. Here, the Volvo XC90 is said to contain around 40 ECUs, the BMW 7 series has around 65, and the VW Passat has around 45. Other car models are known to have up to 70 ECUs. These ECUs are part of the automotive subsystems to be integrated on a shared automotive architecture. The complexity of such an integration process is apparent, as the CAN ID represents the message priority.

As a step to overcome this, one example is found in the case of the XC90, where Volvo is using the Volcano concept [3, 18]. The Volcano concept provides tools for packaging data (signals) into network frames, both for CAN and other networks. The result is that several signals are allocated into message frames with an appropriate static CAN ID to fulfil these signals' temporal requirements. This simplifies the design, development and maintenance of an automotive system. Using the Volcano tools it is also possible to perform a timing analysis of the system, needed at the design stage to schedule the transmissions of real-time variables in such a way that their timing constraints are met.

3.4 Static CAN ID specified by standards

Several standards specify the usage of CAN IDs for various application domains. For example, the CANopen protocol [5] was released in 1995, designed for motion-oriented machine control networks, and can be found in various application domains today, e.g., medical equipment, off-road vehicles, maritime electronics, public transportation, building automation etc. The CANopen standards cover application layer and communication profile, a framework for programmable devices, and recommendations for cables and connectors. The application layer and the CAN-based profiles are implemented in software.

Another example is SAE J1939 [20], published by SAE in 1998, which specifies how messages are defined for engine, transmission and brake systems in truck and trailer applications. Nowadays, SAE J1939 is widely used in these applications, and standardised as ISO 11992.

Looking at other application domains, for tractors and machinery for agriculture and forestry, a SAE J1939-based ISO standard is used: ISO 11783 [7], and NMEA 2000® [14] defines a SAE J1939/ISO 11783 based protocol for marine usage.

These standards all restrict the use of message identifiers by prescribing which identifiers to use for messages carrying specific data.

4 Decoupling CAN ID from CAN PRIO

In this section we describe three CAN schedulers that decouple the message priority from the message identifier: TT-CAN, FTT-CAN and Server-CAN.

4.1 TT-CAN

Time-triggered communication on CAN is specified as TT-CAN, the ISO 11898-4 standard, an extension to original CAN. In TT-CAN, the exchange of messages is controlled by the progression of time, and all nodes are following a pre-defined static schedule. One node, the master node, is periodically (or on the occurrence of a specific event) transmitting a specific message, the Reference Message (RM), which acts as a reference in time. All nodes in the system are synchronising with this message, which gives a reference point in the temporal domain for the static schedule of the message transactions. This schedule is based on a time division scheme, where message exchanges may only occur during specific time slots or in time windows (so called Time Division Multiple Access, TDMA). Hence, the master's view of time is referred to as the network's global time.

4.2 FTT-CAN

Flexible Time-Triggered CAN (FTT-CAN) supports priority-driven scheduling in combination with time-driven scheduling. In FTT-CAN, time is partitioned into fixed size Elementary Cycles (ECs) that are initiated by a special message, the Trigger Message (TM). This message contains the schedule for the synchronous traffic (time-triggered traffic) that shall be sent within this EC. The schedule is calculated and sent by a specific node called the master node. FTT-CAN supports both periodic and aperiodic traffic by dividing the EC in two parts. In the first part, the asynchronous window, a (possibly empty) set of aperiodic messages are sent using CAN's native arbitration mechanism. In the second part, the synchronous window, traffic is sent according to the schedule delivered in the TM. The synchronous window can be scheduled according to an arbitrary scheduling policy. Experimental results have been shown for EDF in [17].

4.3 Server-CAN

Using Server-CAN, as with FTT-CAN, the network is scheduled by a specialised master node (called M-Server), partitioning time into ECs. Also, these ECs are initiated by a specific message, the TM, which is constructed and sent by the master node. By having a centralised scheduler, various (also EDF-based) share-driven scheduling policies can

	Static CAN ID used for scheduling	**Dynamic CAN ID used for scheduling**	**Static CAN ID assigned by tools**	**Static CAN ID specified by standards**
Objective	To provide predictable real-time message transmissions on the CSMA/CR MAC implemented by CAN.	Same as "Static CAN ID used for scheduling" although providing a higher schedulability bound or increased fairness of non real-time messages.	To achieve a system optimised with respect to, e.g., schedulability and/or message response-times.	To allow for interoperability between subsystems, usually application domain specific.
Usage	Academically scrutinised and nowadays often found in embedded systems.	Academic.	A common application is found in the automotive domain, where cars are characterised by high manufacturing volumes and stringent product cost pressures.	Examples are trucks, trailers, tractors, heavy vehicles etc. that usually are more "open" for configurability than cars, which in turn require the usage of standards.
Overhead on nodes	None, as no processing is required.	Some, as the CAN IDs are manipulated during runtime.	Small, due to encoding and decoding of signals into message frames.	None, as no processing is required.
Overhead on network	None, as no overhead is introduced in the messages and no protocol specific messages are sent.	None, as no overhead is introduced in the messages and no protocol specific messages are sent.	None, as no overhead is introduced in the messages and no protocol specific messages are sent.	None, as no overhead is introduced in the messages and no protocol specific messages are sent.
Offline flexibility	Yes, as the system designer has full control over the CAN IDs (although it might be complex without assisting tools).	Less compared with "Static CAN ID used for scheduling", due to a lower number of available CAN IDs.	Yes, more or less depending on the tool used.	None, as the standards have to be followed.
Online flexibility	None, as nothing is done online.	None, although the online manipulation of CAN IDs has the potential to provide a dynamic behaviour.	None, as nothing is done online.	None, as nothing is done online.
Facilitating subsystem integration	No, as the objective here is predictability on a message basis rather than simplifying subsystem integration.	No, and even less compared with "Static CAN ID used for scheduling", due to the lower number of CAN IDs available to use.	Yes, provided that all system details are given to the tool so that a proper optimisation can be made.	Yes, given that all subsystems comply with the standard.

Table 1. Comparative assessments of CAN IDs in practice.

be implemented. The difference between Server-CAN and FTT-CAN is that the latter has fixed size ECs whereas when using Server-CAN the length of an EC is upper-bounded, but can be shortened depending on the actual messages sent. In this way, efficient usage of network slack (e.g., when messages are not sent, or shorter than initially intended) is provided. Also, Server-CAN does not have windows inside an EC. Instead, all traffic is scheduled using network access servers denoted N-Servers.

5 Discussion on subsystem integration

Looking at how CAN IDs are assigned in practice, Table 1 outlines the differences among the four approaches presented in Section 3. Properties evaluated are overhead, flexibility and support of subsystem integration.

Static and dynamic CAN IDs used for scheduling purposes provide valuable scheduling performance but no inherent mechanisms supporting subsystem integration. Using static CAN IDs there is a strong dependency between the subsystem and its corresponding set of CAN IDs. Once several subsystems are integrated, there is a risk of temporal conflicts caused by the originally assigned priorities of the messages belonging to the different subsystems. Using dynamic CAN IDs the situation can even be worse, in the sense that the number of available CAN IDs is lower as (commonly) part of the CAN ID is used to dynamically adjust the message's priority.

The usage of a tool to optimise CAN ID assignments is good from an integration point of view, however, the runtime flexibility is limited as nothing extra (compared with

	Decoupling using TT-CAN	Decoupling using FTT-CAN	Decoupling using Server-CAN
Objective	To provide a time-triggered session layer to standard CAN.	To provide flexibility to CAN, allowing for a mix of statically and dynamically scheduled messages. Time is partitioned into fixed size ECs that in turn are partitioned into time- and event-triggered windows .	To provide a uniform way of handling message streams on CAN. Time is partitioned into dynamic size ECs, where time- and event-triggered traffic are jointly scheduled using server-based techniques.
Usage	Although a SAE standard, not widely used. Found in, e.g., some automotive concept applications.	Academic and educational use (robotics).	Academic only (so far).
Overhead on nodes	Some, due to the TT-CAN session layer (although hardware implementations exist). Higher overhead on the master node compared with the other nodes.	Yes, due to the online scheduling and TM encoding performed by the master node. Also, some overhead on every node due to the decoding of the TM.	Yes, due to the online scheduling and TM encoding performed by the M-Server, and on every node due to the TM decoding at the N-Servers.
Overhead on network	Yes, one protocol message: The cyclic transmission of RM. Also, bandwidth is lost due to the enforcement of TDMA time slots.	Yes, one protocol message: The cyclic transmission of TM. Also here bandwidth is lost due to the fixed size ECs. Moreover, the length of the EC affects the periodicity of the EC and in turn the overhead caused by FTT-CAN on the network.	Yes, two protocol messages: The cyclic transmission of TM and STOP. Also here, as with FTT-CAN, the length of the EC affects the overhead.
Offline flexibility	High, as message transmissions can be optimised into time slots.	High. Some messages can be considered static, for which pre run-time guarantees can be given.	High, as all message streams can be encapsulated into N-Servers providing bandwidth isolation.
Online flexibility	No.	Yes, FTT-CAN provides admission control for dynamic adding and removing of message transmissions in the static window.	Yes, Server-CAN provides full control over message transmissions using an admission control for dynamic adding, changing and removing of N-Servers. Also, the online scheduling allows for implementation of advanced bandwidth sharing algorithms for an adaptive behaviour.
Facilitating subsystem integration	Yes, as TDMA provides temporal partitioning, hence separating subsystems in time avoiding interference.	Yes, although the static window of FTT-CAN offers a lower temporal resolution compared with TT-CAN, due to the scheduling of messages in ECs.	Yes, due to the same reasons as for FTT-CAN.

Table 2. Comparative assessments of approaches decoupling the CAN ID from the CAN PRIO.

native CAN) is done with regards to the transmission of messages online. Also, the result of the optimisation can be degraded if some system does not allow for re-assignment of identifiers (due to, e.g., the usage of legacy and/or proprietary systems).

When conforming to one of the standards presented in Section 3.4 the situation is better from an integration point of view. Of course, this requires that all subsystems follow the same standard, and it might not always be possible to enforce such a requirement.

Instead, we argue for decoupling the message identifier from the message priority, since it removes a great obstacle in the context of subsystem integration: A change in an identifier to make it system-unique would not affect the scheduling. Such decoupling is partially or totally achieved using TT-CAN, FTT-CAN or Server-CAN. The differences among these techniques are outlined in Table 2.

TT-CAN is a pure TDMA approach, allowing for subsystem integration, although limiting the online flexibility of the system. FTT-CAN performs the scheduling online. However, the decoupling of message identifier and message priority is only valid for the synchronous window of an EC. It is possible to configure FTT-CAN such to only have a synchronous window, although there is always some bandwidth potentially lost due to the fixed size ECs and the lack of a bandwidth reclamation mechanism. The usage of Server-CAN, on the other hand, completely decouples the message identifier from the message priority, by scheduling all messages according to EDF using server-based scheduling techniques. Also, bandwidth is efficiently reclaimed thanks to the dynamic length ECs. Hence, greatest flexibility is provided using Server-CAN. However, all three approaches come at a cost in terms of protocol overhead: The cost of using Server-CAN is evaluated in [15]. Also, note that TT-CAN and the synchronous usage of FTT-CAN are optimised for periodic traffic whereas Server-CAN allows for both periodic and aperiodic traffic. Here, the two former provides less jitter for periodic traffic compared with Server-CAN, but do not have the same support for aperiodics.

6 Summary

This paper argues for the importance of decoupling CAN IDs from CAN PRIOs in the context of subsystem integration. Three CAN schedulers that allow for this feature are presented, where we particularly argue for Server-CAN as a strong candidate due to its combination of flexibility and full decoupling of CAN ID from CAN PRIO.

Acknowledgements

This work was supported by the ARTIST2 Network of Excellence on Embedded Systems, IST-004527, under the Adaptive Real-Time Cluster, and by the Swedish Foundation for Strategic Research (SSF) via the research programme PROGRESS.

References

[1] L. Almeida, P. Pedreiras, and J. A. Fonseca. The FTT-CAN protocol: Why and how. *IEEE Transaction on Industrial Electronics*, 49(6):1189–1201, December 2002.

[2] R. J. Bril, J. J. Lukkien, R. I. Davis, and A. Burns. Message response time analysis for ideal Controller Area Network (CAN) refuted. In J.-D. Decotignie, editor, *Proceedings of the 5^{th} International Workshop on Real-Time Networks (RTN'06) in conjunction with the 18^{th} Euromicro International Conference on Real-Time Systems (ECRTS'06)*, Dresden, Germany, July 2006.

[3] L. Casparsson, A. Rajnák, K. W. Tindell, and P. Malmberg. Volcano - a revolution in on-board communication. *Volvo Technology Report 98-12-10*, 1998.

[4] G. Cena and A. Valenzano. An improved CAN fieldbus for industrial applications. *IEEE Transaction on Industrial Electronics*, 44(4):553–564, August 1997.

[5] CiA. CANopen communication profile for industrial systems, based on CAL. CiA Draft Standard 301, rev 3.0, October, 1996. http://www.canopen.org.

[6] M. Di Natale. Scheduling the CAN bus with earliest deadline techniques. In *Proceedings of the 21^{st} IEEE International Real-Time Systems Symposium (RTSS'00)*, pages 259–268, Orlando, FL, USA, November 2000. IEEE Computer Society.

[7] ISO 11783-1. Tractors and machinery for agriculture and forestry – serial control and communications data network – part 1: General standard for mobile data communication. *International Standards Organisation (ISO)*, ISO Standard-11783-1, 2005.

[8] ISO 11898. Road vehicles – interchange of digital information – Controller Area Network (CAN) for high-speed communication. *International Standards Organisation (ISO)*, ISO Standard-11898, November 1993.

[9] ISO 11898-4. Road vehicles – Controller Area Network (CAN) – part 4: Time-triggered communication. *International Standards Organisation (ISO)*, ISO Standard-11898-4, December 2000.

[10] H. Kopetz, R. Obermaisser, P. Peti, and N. Suri. From a federated to an integrated architecture for dependable embedded real-time systems. Technical Report 22, Technische Universität at Wien, Institut für Technische Informatik, Treitlstr. 1-3/182-1, 1040 Vienna, Austria, 2004.

[11] C. L. Liu and J. W. Layland. Scheduling algorithms for multiprogramming in a hard real-time environment. *Journal of the ACM*, 20(1):40–61, January 1973.

[12] M. A. Livani and J. Kaiser. EDF consensus on CAN bus access for dynamic real-time applications. In J. Rolim, editor, *Proceedings of the IPPS/SPDP Workshops*, volume Lecture Notes in Computer Science (LNCS-1388), pages 1088–1097, Orlando, Florida, USA, March 1998. Springer-Verlag.

[13] L. Lo Bello and O. Mirabella. Randomization-based approaches for dynamic priority scheduling of aperiodic messages on a CAN network. In *LCTES '00: Proceedings of the ACM SIGPLAN Workshop on Languages, Compilers, and Tools for Embedded Systems*, pages 1–18, London, UK, 2001. Springer-Verlag.

[14] NMEA 2000® Standard. The National Marine Electronics Association. http://www.nmea.org.

[15] T. Nolte. *Share-Driven Scheduling of Embedded Networks*. PhD thesis, Department of Computer and Science and Electronics, Mälardalen University, Sweden, May 2006.

[16] T. Nolte, M. Nolin, and H. Hansson. Real-time server-based communication for CAN. *IEEE Transactions on Industrial Informatics*, 1(3):192–201, August 2005.

[17] P. Pedreiras and L. Almeida. A practical approach to EDF scheduling on Controller Area Network. In *Proceedings of the IEEE/IEE Real-Time Embedded Systems Workshop (RTES'01) at the 22^{nd} IEEE Real-Time Systems Symposium (RTSS'01)*, London, England, December 2001. Technical report, Department of computer science, University of York, England.

[18] A. Rajnák. Volcano: Enabling correctness by design. In R. Zurawski, editor, *The Industrial Communication Technology Handbook*, pages 32–1 to 32–18. CRC Press, Taylor & Francis Group, 2005.

[19] SAE J1938 Standard. Design/Process Checklist for Vehicle Electronic Systems. *SAE Standards*, May 1998, SAE.

[20] SAE J1939 Standard - The Society of Automotive Engineers (SAE) Truck and Bus Control and Communications Subcommittee. SAE J1939 Standards Collection, 2004, SAE.

[21] K. W. Tindell, H. Hansson, and A. J. Wellings. Analysing real-time communications: Controller Area Network (CAN). In *Proceedings of 15^{th} IEEE International Real-Time Systems Symposium (RTSS'94)*, pages 259–263, San Juan, Puerto Rico, December 1994. IEEE Computer Society.

[22] K. M. Zuberi and K. G. Shin. Non-preemptive scheduling of messages on Controller Area Network for real-time control applications. In *Proceedings of the 1^{st} IEEE Real-Time Technology and Applications Symposium (RTAS'95)*, pages 240–249, Chicago, IL, USA, May 1995. IEEE Computer Society.