Markov Chain-based Performance Evaluation of FlexRay Dynamic Segment

Jimmy Jessen Nielsen Hans-Peter Schwefel
Aalborg University, Department of Electronic Systems
Fredrik Bajers Vej 7, DK-9220, Aalborg
{jjni03, hps}@kom.aau.dk

Amen Hamdan
BMW Car IT GmbH
Petuelring 116, D-80809, München
Amen.Hamdan@bmw-carit.de

Abstract

With the launch of the new BMW X5, the FlexRay protocol for in-car communication has found its way to series-produced vehicles. The FlexRay protocol supports both deterministic and non-deterministic transmission of data frames. FlexRay data frames that are being transmitted in the non-deterministic dynamic segment might become displaced under adverse circumstances. In the design of a FlexRay network it is important to have sound understanding of any implications of certain design decisions on the performance of the overall network, here specifically the displacements in the dynamic segment.

This paper proposes a novel approach to performance analysis of the dynamic segment based on Markov chain transient analysis. The model of the dynamic segment is a two-dimensional discrete-time Markov chain, where the discrete time steps represent minislots of the dynamic segment. Model properties and assumptions are discussed and expressions for calculating performance metrics are provided. Finally, measurements obtained from a real FlexRay network are used to test model assumptions and to validate the accuracy of model output.

Keywords - flexray, in-car networks, dynamic segment, markov chain, transient analysis, performance analysis

1 Introduction

The increasing number of x-by-wire applications in cars entails that requirements on safety and hard real-time of the in-car network are becoming more strict [4]. Protocols such as Byteflight and FlexRay, where FlexRay is a recent extension of Byteflight, have been developed and used for satisfying the requirements of x-by-wire applications. The first series car to use a FlexRay in-car network is the new BMW X5, which has recently been introduced by BMW Group. In near future FlexRay is expected to connect multiple Electronic Control Units (ECUs) implementing chassis, powertrain, and driver assistance applications [6].

Besides providing strict determinism, the FlexRay protocol supports priority based media access in the dynamic segment via a Flexible Time Division Multiple Access (FTDMA) scheme. The properties of the FTDMA scheme is discussed in [2]. The flexibility provided by this media access scheme allows frames to be displaced and thus delayed in adverse situations. This dynamism combined with the increasing number of applications motivates the need for quantitative performance evaluation of the network.

The Byteflight protocol is nearly identical in functionality to the FlexRay dynamic segment as it is also based on FTDMA. The authors in [3] present an analysis of the Byteflight protocol, where they investigate the performance-related consequences of different design choices. Particularly, they identify that network designers need to make a trade-off between flexibility and performance. This issue applies similarly to the FlexRay dynamic segment, and is discussed further in Section 2. Currently available tools for performance evaluation of specific FlexRay traffic models focus on simulation. Simulation tools such as [8] and [7] may be used to make a quantitative performance analysis, by defining the traffic model using virtual traffic generators. However, the process of obtaining a sufficient level of confidence with a simulation is typically very time-consuming compared to analytical approaches.

This paper presents an approach to performance assessment of the dynamic segment in FlexRay that is based on transient analysis of a Markov Chain (MC). This performance assessment is less time-consuming than simulation and can be made prior to implementing a simulation or prototype.

Specifically, this paper contributes to the literature by 1) demonstrating how the FlexRay communication cycle can be modeled as a MC, 2) specifying how performance metrics are extracted from MC model, and 3) validating the results obtained from the model by comparing to measurements from an actual FlexRay network.

In Section 2 a brief description of the relevant FlexRay properties is given and the main motivation for the model is presented. Next, the FlexRay model is discussed and presented in Section 3. The following Section 4 specifies how performance metrics are calculated from the model. Section 5 presents a validation of the model based on measurements from an actual FlexRay network. Finally, Section 6 contains the conclusion of this paper.

2 FlexRay Dynamic Segment

The following gives a brief introduction to FlexRay and especially the dynamic segment, which is the focus of this paper. More information on FlexRay is available in [5].

In FlexRay the communication cycle is the fundamental element of the media access scheme. The communication cycle is divided into four segments as depicted in Fig. 1.

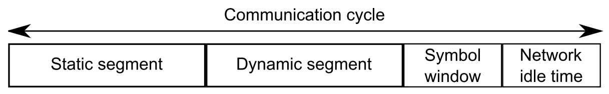


Figure 1. The FlexRay communication cycle.

Within a communication cycle, ECUs may transmit frames in the static and dynamic segments. At design-time an ECU has been assigned one or more slot IDs in which it may transmit. Each slot ID is only used by one ECU, and in this way collisions will never occur. In the *static segment*, where a Time Division Multiple Access (TDMA) scheme is used, the transmission of frames is completely deterministic. The *dynamic segment* uses a dynamic mini-slotting scheme, also generally referred to as FTDMA. This scheme is not deterministic since the offset from the beginning of a communication cycle until an ECU may transmit, varies with the number of frames already sent in the same communication cycle. Fig. 2 exemplifies the concept of this scheme.

An unused dynamic slot has the duration of one minislot, which is the common time unit in a FlexRay network, and the length of the communication cycle is a fixed number of minislots. If a dynamic slot is used for transmitting a frame, the duration of the dynamic slot is extended to several minislots, depending on the payload size of the frame. This is exemplified for dynamic slots $m + 3$ and $m + 6$ in the figure. Every ECU maintains a local counter of both the current minislot ID and the current dynamic slot ID. Both are reset at the beginning of each cycle.

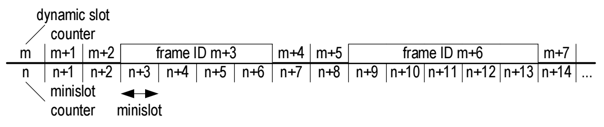


Figure 2. Dynamic minislotting scheme.

Since the payload length and the number of frames transmitted in each communication cycle varies, it may occur that a frame cannot be successfully transmitted within the available minislots. In that case the frame will not be transmitted in the current communication cycle. Instead the frame is displaced to a later communication cycle with enough available minislots. More specifically, a frame is displaced if the minislot counter exceeds the threshold *pLatestTx* that is configured in each FlexRay node. *pLatestTx* is the last minislot in which the node can successfully transmit a frame with the maximum allowed frame size.

Displacements may occur even with a low number of used dynamic slots. If for example the communication cycle is configured for a length of 250 minislots, and a frame has a frame ID of 230 and takes up 5 minislots to transmit, no more than 15 earlier minislots may be included in frame transmissions in a communication cycle, before the frame cannot be transmitted and is displaced.

In order to reduce the number of displacements, the FlexRay network designer should aim at using as low frame IDs as possible. However, this is typically inappropriate with respect to compatibility and upgradability. The designer therefore needs to determine a configuration of the network that provides an acceptable trade-off between flexibility and performance. Besides displacements, other aspects of performance could be relevant to consider. In this paper the last dynamic slot is also calculated.

The modeling approach proposed in this paper provides the network designer with a tool for making quantitative performance evaluation of different network configurations and traffic models in order to rapidly iterate over the design of the network.

3 FlexRay Model

In this section the approach to modeling the dynamic segment is described in detail. The performance metrics described in the following are the desired outputs from the model.

Last Dynamic Slot (LDS) distribution: The LDS is the value of the dynamic slot counter at the end of a communication cycle. The more minislots that have been used for transmitting frames, the lower the value of the LDS. The network designer may use the LDS as an indicator of the level of utilization and as an indication of which frames could not have been transmitted in the communication cycle. If the LDS has a lower slot ID than the ID used by a certain ECU, that ECU could have had a frame scheduled for transmission that it was not allowed to transmit in the corresponding communication cycle.

Frame displacement probability: The objective of this metric is to quantify the risk of frame displacements for a specific frame ID. The network designer may use this metric to determine the quality of service in relation to displacements that the network delivers for a given configuration.

3.1 Model Framework

The following description of the FlexRay model assumes a basic understanding of transient analysis of discrete-time MCs, see e.g. [1]. In this paper the transition probability matrix is denoted by $\mathbf{P}$ and the state probability vector for time k by $\pi(k)$. The state probabilities are calculated using the standard equation

$$\pi(k) = \pi(0)\mathbf{P}^k \tag{1}$$

The main features of the FlexRay model are described in the following.

Time step: Each time step in the MC has the duration of exactly one minislot. After k time steps, the state probability vector $\pi(k)$ describes the possible outcomes of the communication cycle via state probabilities. An example of information that could be derived from $\pi(k)$ is the probability of the frame with ID n having been transmitted after exactly k time steps.

Idle and transmission states: Each state in the MC constitutes either an idle minislot, or a minislot that is used for transmitting a frame. Typically, several consecutive minislots are involved in the transmission of a single frame. In the MC these consecutive minislots are mapped into a sequence of connected states that accurately represent the length of the frame transmission.

Frame ID priorities: The prioritization of frames via dynamic mini-slotting is included in the model by defining the transition probabilities so that states in the MC are visited in the order specified by their corresponding slot IDs.

Simplifying assumptions: The following simplifying assumptions are made for the FlexRay model:

- For each frame ID a a constant frame length l_a is used. The FlexRay specification allows a node to use a variable frame size in the dynamic segment. However, the model assumes that frames within a frame ID use the same payload length.
- For each frame ID a a constant arrival probability p_a is used. Further, frame arrival probabilities are assumed to be independent and inter-arrival times to be geometrically distributed.
- The pLatestTx check is not integrated in the model structure. This is discussed further in Section 4.

The validity of these simplifying assumptions is discussed further in Section 5.

3.2 Detailed Model Definition

The model is based on a non-homogenous MC using a two-dimensional state space (a, b). Here, a equals the ID of the dynamic slot relative to the beginning of the dynamic segment, i.e. the first dynamic slot should have $a = 1$. The value range of a is $[0; s_{max}]$ (where s_{max} is short notation for the Flexray specific parameter $gNumberOfMinislots$). b relates to the states within a dynamic slot. $b = 1$ designates an idle state and $b > 1$ designates transmission states. The value range of b is $[1; l_{max} + 1]$, where l_{max} is the maximum allowed length of a frame measured in minislots.

The initial state of the MC is $(0, 1)$.

The transition probabilities of the MC correspond to the arrival probabilities of each dynamic slot.

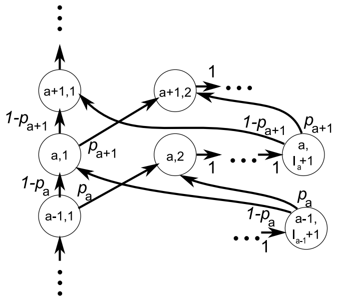


Figure 3. FlexRay MC structure

The structure of the FlexRay MC is shown in Fig. 3. For a dynamic slot a, there is an idle state with index $(a, 1)$ and l_a transmission states from $(a, 2)$ to $(a, l_a + 1)$. l_a is the transmission length in minislots of the frame in dynamic slot a. The arrival probabilities for dynamic slot a is p_a. If a dynamic slot is unused, the arrival probability becomes $p_a = 0$, i.e. there are no transmission states.

The frame length l_a that is needed to create the MC, is calculated from the payload length. However, this calculation is not completely trivial, since it depends on the low-level parameters of the concerned FlexRay network. This calculation may be found in [5, Appendix B4.14].

In order to calculate the state probability vector π, the transition probability matrix **P** needs to be generated from the traffic model. One possible approach to generating **P** is to iterate over the set of dynamic slots, and with a starting point in the associated idle state, consider the state transitions initiated in this state. For each slot s one of the following three situations apply:

a) No ECUs are assigned to slot s and only a transition to the idle state in slot $s + 1$ is possible.

b) Frame transmission is possible in slot $s+1$, but not in slot $s+2$. Transitions to idle and transmission states in slot $s+1$ and to the idle state in slot $s+2$ should be created.

c) Frame transmission is possible in slot $s+1$ and in slot $s+2$. Transitions to idle and transmission states in slot $s+1$ and to the idle and transmission states in slot $s+2$ should be created.

Having outlined the procedure for specifying the transition probability matrix $\mathbf{P}$ for the FlexRay MC, only the initial state probability vector $\pi(0)$ is left to be specified. With the initial state of $(0,1)$, the state probability vector should be set as

$$\pi(0) = [1, 0, ..., 0] \tag{2}$$

4 Calculating Performance Metrics

The calculation of the performance metrics consists of two steps. First, the state probabilities $\pi(k)$ are calculated, and secondly the performance metrics *frame displacement probability* and *LDS distribution* are computed from $\pi(k)$. Formulas for computing these metrics are presented in subsections 4.2 and 4.3. However, first the simplifying assumption that the *pLatestTx* check is not included in the MC is discussed.

4.1 pLatestTx Check

In the MC, frame transmission are being initiated even if they should not, due to the *pLatestTx* check not being performed. Thus, in the MC, the transmission states before the last transmission state may contain probability mass at the end of the communication cycle, even though this would not occur in an actual network. The start of the arrows in Fig. 4, show which states contain excess probability mass, while the arrow ends show where the probability mass should reside, had the *pLatestTx* check been carried out.

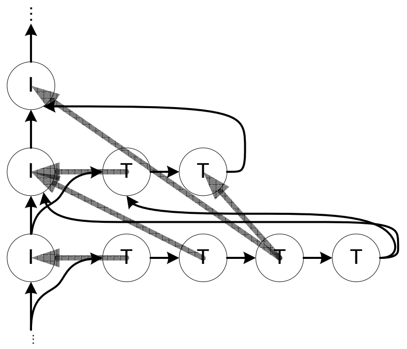


Figure 4. pLatestTx issue.

However, with relation to frame displacements, the missing *pLatestTx* check allows a more accurate calculation of the frame displacement probability, and the *pLatestTx* check should therefore purposely be left out of this calculation. On the contrary, the calculation of the LDS needs to have the MC reflect the behavior of an actual FlexRay network. Subsection 4.3 describes possible approaches for bringing the *pLatestTx* check into the calculation of the state probabilities when calculating the LDS.

Now the calculation of the performance metrics is described.

4.2 Frame Displacement Probability

In order to determine if any frames have been displaced in a communication cycle, the outcome of the MC must be calculated for the time $k = s_{max}$, i.e. when the communication cycle has finished. This is obtained using Eq. (1). The displacement probability for a dynamic slot is given by all state probabilities existing before the last transmission state of slot s. The state probabilities before slot s should be conditioned on the transmission of a frame in s, hence the sum of the state probabilities is multiplied by the arrival probability p_s. The frame displacement probability for frames transmitted in dynamic slot s can be obtained from

$$P(s \text{ is displ.}) = \sum_{b=2}^{l_s} \pi(k)_{s,b} + p_s \cdot \sum_{a=1}^{s-1} \left(\sum_{b=1}^{l_{max}+1} \pi(k)_{a,b} \right) \tag{3}$$

4.3 Last Dynamic Slot

The LDS is the value of the dynamic slot counter at the end of a communication cycle, i.e. for time $k = s_{max}$. Since the last dynamic slot in a communication cycle in an actual network may have been used either for finishing the transmission of a frame or being idle, the LDS for slot s may be calculated via

$$\mathbf{S}_s = \pi(k)_{s,1} + \pi(k)_{s,l_s+1} \tag{4}$$

However, as discussed in Subsection 4.1, the *pLatestTx* is not performed when calculating $\pi(k)$. Eq. (4) assumes that the *pLatestTx* check has been performed during the calculation of $\pi(k)$. There are several ways to include the *pLatestTx* check. In the following two exact approaches, and one approximate approach are described.

The first approach is to use multiple transition probability matrices that prevent transitions to the transmission states when the *pLatestTx* values of the concerned slots have been reached. The second approach is to perform a post-processing operation that propagates the residual probability mass according to the transition probabilities as exemplified in Fig. 4. After applying either of these two approaches, Eq. (4) may be used to calculate the LDS. The third approach is a simplified post-processing that adds the residual probability mass to the relevant idle states only. This approach does not consider cases where probability mass needs to be propagated to transmission states and is therefore only an approximation in such cases. However, it is simpler to implement than the two other solutions and may provide the exact solution if the used network configuration and traffic model does not contain such cases. The probability of slot s being the LDS is calculated via

$$\mathbf{S}_s \approx \pi(k)_{s,l_s+1} + \sum_{a=0}^{l_{max}} \pi(k)_{s-a,a+1} \cdot I(a \leq l_{s-a}) \quad (5)$$

Here, the indicator function $I(a \leq l_{s-a})$ evaluates to 0 when the expression is false and 1 when it is true. The plots of the LDS distribution that are included in Section 5, have been calculated using this approximate approach. Regardless of which approach is used to calculate the LDS probability for each of the s_{max} dynamic slots, the complete distribution of last dynamic slots is given as

$$\mathbf{S} = [\mathbf{S}_1,\ \mathbf{S}_2,\ ...,\ \mathbf{S}_{s_{max}}] \quad (6)$$

5 Model Validation

The validation of the FlexRay MC model is divided into two steps. The first step is to investigate how well the assumptions on traffic properties match the actual network traffic. The second step is a comparison of results calculated using the MC and results obtained from measurements on a real FlexRay network.

5.1 Assumptions

The first assumption is that frames with the same frame ID use the same payload length. The test data was found to comply to this assumption, which should not cause any inaccuracies in the model prediction. However, since a fixed payload length is application-dependent, this assumption may limit the applicability of the model, or at least provide inaccurate results in cases where dynamic frame sizes are used. Note however, that the Markov Chain model can also be extended to arbitrary frame size distributions via a modification of the 'upward' transitions in 4.

Further, assumptions were made that frame arrivals are independent. This covers both independence between frames with identical frame IDs, but also between frames not sharing frame IDs. These assumptions were tested using correlation analysis. In order to test the assumptions, a binary arrival sequence was created for each frame ID in the measurements. That is, the communication cycles in which a frame with the given frame ID has arrived are represented by a 1 in this sequence, while lack of arrival is a 0 in the sequence.

The autocorrelation was used to test the correlation of frames with identical frame IDs. Autocorrelation plots have been created for all frame arrival sequences. The frame arrivals were found to be highly correlated and showing a large degree of periodicity. This contradicts the assumption of independence between frame arrivals within each frame ID, and is expected to introduce some level of inaccuracy to the results of the model, compared to an actual network.

Finally, the correlation for frames with different frame IDs was investigated by computing the cross-correlation coefficient between all pairs of frame arrival sequences. The result is depicted in Fig. 5, where the darkness of each square expresses the degree of correlation between a pair of arbitrarily indexed arrival sequences. The results show that besides the auto-correlations along the diagonal, most frame arrivals were not or only weakly correlated. This result supports the assumption of independence between different frame IDs. However, it could be relevant to investigate the cross-correlation for other lags than 0.

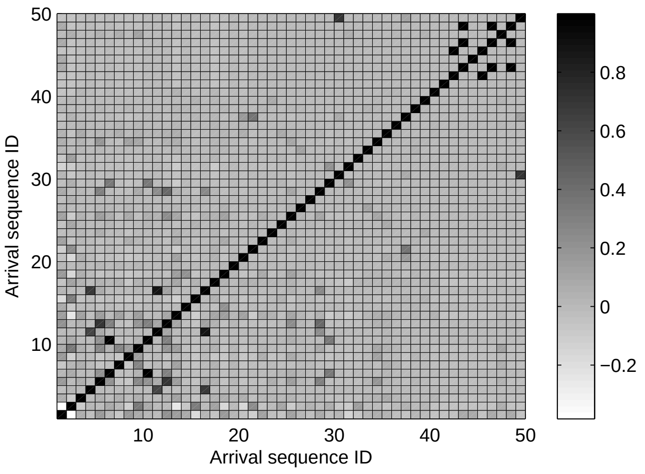


Figure 5. Cross-correlation plot of frame arrival sequences for lag 0.

5.2 Model Output Comparison

The following presents two comparison plots, Fig. 6 and Fig. 7 that have been selected from a larger result set. The plots are based on measurement and model results for two traffic models that differ in the assigned frame IDs and arrival rates. The plots in the figures show the Cumulative Distribution Function (CDF) of LDS. Here, the horizontal distance between the lines is interesting since it shows how accurately the model mimics the behavior of the actual system. The results presented here show the model predictions from the result set that have the least horizontal difference (best) and the largest difference (worst).

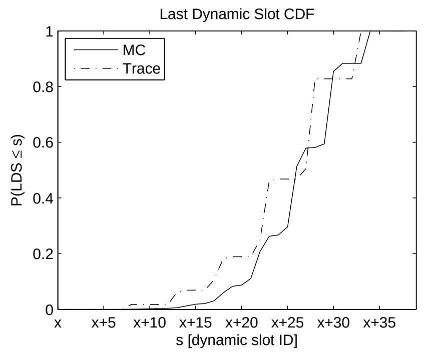


Figure 6. Best estimate of LDS distribution.

The plots are indexed relatively from x, since the specific technical details are not of interest here. The result in Fig. 6 shows a high degree of resemblance between the results of the model and the trace. The horizontal difference seems to be limited to 5 minislots and within 2-3 minislots for the most part.

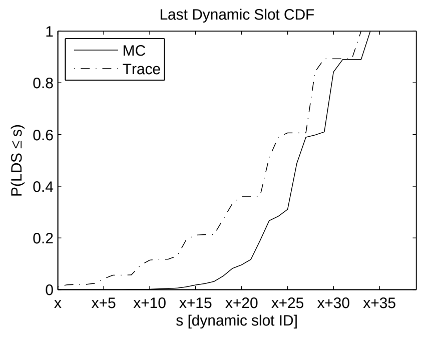


Figure 7. Worst estimate of LDS distribution.

The results in Fig. 7 show a lower degree of similarity, where the horizontal difference is as high as 10 minislots. Common for both plots is a tendency of the model towards a more optimistic result, which is caused by the simplifying assumption regarding independence of frame arrivals being inconsistent with the actual network. However, these are only minor issues that will not hinder the network designer in making qualified decision on the design of the network.

6 Conclusion and Further Work

A Markov chain based model for quantitively analyzing the performance of the FlexRay dynamic segment has been presented as a tool for the network designer to make early predictions on network behavior. Results are obtained via transient analysis of this two-dimensional Markov chain. Expressions for calculating the performance metrics *distribution of Last Dynamic Slot* and *frame displacement probability* have been discussed and presented.

The model is based on assumptions regarding mutually independent frame arrivals and constant payload lengths within each frame ID. This allows for the traffic within each frame ID to be described by just three constant parameters *frame ID*, *frame length* and *arrival probability*. A validation of model output against traces obtained from a real FlexRay network shows that the accuracy of the prediction of the distribution of LDS is within 5 minislots in one case and within 10 minislots in another. This level of accuracy appears sufficient for a network designer to make qualified decisions in the early phases of network design.

Further work could add additional performance metrics such as jitter, which is caused by displacements and variations in the number of minislots that a dynamic slot is offset from the beginning of the dynamic segment. Another topic could be to validate the model more thoroughly with a wider selection of network configurations and traffic models. Further it would be interesting to use the model for designing a network or predicting the performance of future network configurations and traffic models via extrapolation. Finally, it would be interesting to extend the model to allow the effect of errors on performance to be investigated.

The research work was in parts conducted in the context of a student project at the Department of Electronic Systems at Aalborg University, and as such supported by the Study Board at Aalborg University, see www.esn.aau.dk.

References

[1] C. Cassandras and S. Lafortune. *Introduction to discrete event systems*. Kluwer Academic Boston, 1999.

[2] G. Cena and A. Valenzano. On the properties of the flexible time division multiple access technique. *IEEE Transactions on Industrial Informatics*, 2(2):86–94, 2006.

[3] G. Cena, A. Valenzano, C. IEIIT, and I. Torino. Performance analysis of Byteflight networks. *IEEE International Workshop on Factory Communication Systems. Proceedings.*, pages 157–166, 2004.

[4] G. Cena, A. Valenzano, and S. Vitturi. Advances in automotive digital communications. *Computer Standards & Interfaces*, 27(6):665–678, 2005.

[5] F. Consortium. *FlexRay Communications System Protocol Specification Version 2.1 Revision A*. FlexRay Consortium, 2005.

[6] W. Kuffner, G. Reichart, J. Berwanger, M. Peteratzinger, and A. Schedl. FlexRay Exploitation of a Standard and Future Prospects. In *Convergence 2006, Detroit, MI, USA, Session: International Standards*. SAE International, 2006.

[7] Mirabilis Design. Visualsim. www.mirabilisdesign.com.

[8] Vector. CANoe.FlexRay. www.vector-worldwide.com.

Asynchronous Traffic Signaling over Master-Slave Switched Ethernet protocols

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

Abstract

Network protocol designers have always been divided between the adoption of centralized or distributed communication architectures. Despite exhibiting negative aspects like the existence of a single point-of-failure in the master as well as computational overhead and an inefficient handling of the asynchronous communications, Master-Slave protocols have always found their space mainly given their simplicity of operation and deployment as well as good control over the communication medium. Along the years, many protocols based in this paradigm have been proposed, with many of them being still used today. Some of the negative aspects traditionally exhibited by these protocols have also been attenuated, e.g. with master replication and master/multi-slave control, but the handling of asynchronous requests is still a limitation concerning the response time and overhead. In this paper, we address the specific case of micro-segmented switched Ethernet networks, where Master-Slave protocols are used to control the load submitted to the switch and prevent high queuing jitter and memory overflows. Particularly, we propose a novel signaling mechanism that reduces the asynchronous traffic response time and network overhead, by exploiting the full duplex channels, and analyze the integration of this mechanism in the FTT-SE and Ethernet Powerlink protocols.

1 Introduction

Ethernet became generalized as a communication protocol in many application domains, well beyond office automation, for which it was originally conceived. Particularly, Ethernet is now a de facto standard in the industrial automation domain, as well as in large embedded systems. Reasons for this dissemination include its low cost, the scalable and high bandwidth, mainly comparing to other fieldbuses, the mature technology, the existence of device drivers for many operating systems and the easy integration with Internet protocol stacks [5].

At the lower levels of automation systems on the plant floor, the applications are commonly time constrained. To cope with this requirement several Ethernet-based protocols were proposed to fill the intrinsic gap associated to the original medium access indeterminism of Ethernet, e.g., EPL (ETHERNET Powerlink) [2], EthernetIP [3], EtherCAT [1], PROFINET [4] and FTT-Ethernet (Flexible Timed Triggered) [9].

The recent evolution to Switched Ethernet brought multiple forwarding paths in a collision free domain, which allows a less stringent control in the transmission instants and guarding windows that were typically required with shared Ethernet. However, switches essentially shifted the collision problem at the communication medium to a congestion problem in the queues, which still introduces some level of timing indeterminism associated to the queuing policies and queue lengths, possibly leading to blocking effects among traffic streams. Consequently, achieving low communication jitter and latency values still depends on the use of specific RT protocols to enforce strict temporal transmission control, as it was typical with shared Ethernet, assuring queuing free transactions (e.g. EPL [2] and PROFINET [4]). For instance in the case of EPL that temporal control is such that all transactions are serialized, thus not even exploiting the availability of multiple forwarding paths made available by the Ethernet switches. Other protocols act upon the input data flows [7] to bound the maximum queuing depths and, consequently, the transmission jitter and latency, and to prevent queue overflows.

Some of the real-time (RT) communication protocols recently developed for switched Ethernet networks are based on the Master-Slave paradigm (e.g. EPL, EtherCAT and FTT-Ethernet). The use of this paradigm reduces the traffic management requirements on the slaves, reduces inconsistent communication scenarios and favors predictability by preventing nodes from transmitting out of time. However, it is also well known that this paradigm is not very efficient in handling asynchronous communications, requiring special mechanisms for that purpose. A common technique consists in having queues within the nodes for the asynchronous traffic, possibly several queues with different priorities, and having the master polling those queues with rates adequate to the degree of responsiveness required by the asynchronous traffic. That poll mechanism encompasses two different phases. The first one, called *signalling phase*, in which the master inquires the nodes about the existence of asynchronous traffic ready to be transmitted, and a *transmission phase*, in which the master polls the transmission of that traffic in adequate instants in time. These two phases can either be separate where the *transmission phase* comes as consequence of the *signalling phase*, or merged in a periodic poll mechanism that leads to unused bandwidth when no messages are to reply the poll.

Notice that whether the two phases are jointly or separately implemented, in both there is an intrinsic compromise between responsiveness and overhead; a higher responsiveness requires polling at higher rates, but polling at higher rates potentially implies an higher overhead bandwidth. In this paper it is proposed exploring the full duplex features of common Ethernet switches to implement a new signaling mechanism in Master-Slave Switched Ethernet protocols, which does not suffer from the referred responsiveness versus overhead compromise and may dramatically improve the QoS given to the asynchronous traffic.

The mechanism herein described can be potentially applied to any master-slave protocol based on full-duplex micro-segmented switched Ethernet infrastructures. For illustration and performance assessment purposes the paper describes how the mechanism can be adapted to the EPL and FTT-SE [8] protocols. The remainder of the paper is structured as follows: the next section describes possible signaling mechanisms. Then the novel signaling mechanism is presented. Afterwards the paper describes how the signaling mechanism can be adapted to the FTT-SE and EPL protocols. The following section presents a response time analysis for the RT asynchronous traffic within FTT-SE and an assessment of the throughput and jitter enhancements brought by the new mechanism. Finally the conclusions close the paper.

2 Background

One of the key aspects that must be considered when developing a hard-RT protocol over shared Ethernet networks is that collisions must be completely avoided, in face of the non-deterministic collision resolution mechanism employed by this standard. Therefore, all the message transmission instants are rigorously controlled to avoid collisions. Protocols such as EPL and FTT-Ethernet enforce this behavior with a Master-Slave architecture, centralizing all the communication control in the Master node; slave nodes can only issue message transmission after being explicitly polled by the Master. Despite the intrinsic absence of collisions, the Master-Slave paradigm is still useful in micro-segmented switched Ethernet topologies to control the switch queuing and thus enforce timeliness guarantees, particularly when very constrained values for jitter and end-to-end latency are required.

One negative aspect of the Master-Slave paradigm is the overhead. This overhead comes in two forms, one due to the need to transmit the *master poll* and another caused by the time spent by the slaves decoding and preparing the reply to the poll, which is normally called the *turnaround time*. This interval of time, which mediates between the poll reception and the beginning of the reply transmition, depends on the slave implementation technology. For standard PC-based solutions and network interface cards (NICs) can reach 200 μs, even using RT kernels and stacks. Reducing this value to a few tens of microseconds is possible with special hardware support.

The absence of transmission autonomy makes the slaves fully master dependant, which turns out to be a bottleneck when developing asynchronous messaging schemes in Master-Slave protocols. Several techniques have been developed to override this constraint:

- Using a periodic polling mechanism (e.g. FTT-Ethernet), oriented to the individual asynchronous streams in a blind way. The Master treats the asynchronous messages (AM) like the synchronous (periodic) ones, periodically scheduling them, with a period equal to their minimum inter-transmission times ($Tmit$), without knowing whether there are pending requests for its transmission or not. The only difference concerning the periodic traffic is that for the AMs the absence of an actual message transmission after a Master poll does not mean an error but instead signals the absence of pending asynchronous requests. Deterministic time guarantees can also be given for these messages but with a, normally, very large degree of pessimism associated to the inefficient use of the bandwidth. The signaling latency, i.e., the interval of time from the moment an asynchronous message becomes ready in a node and that information reaching the master, can grow up to two times $Tmit$ in the worst case.

- A bandwidth reclaiming mechanism can be applied on top of the periodic polling mechanism above described, as a feedback scheme to detect unanswered polls, re-using that bandwidth and improving the average network throughput. Unanswered polls can be either autonomously sensed by the master or explicitly reported by the respective slave with a short message. The unused time can then be re-allocated with a follow up polling message. Notice that this mechanism has no impact on the AM worst-case response time, which happens when there are pending requests in all Master pools, but may have a positive impact on the average throughput since unused bandwidth may be reallocated. This technique is easily deployed in processor systems [6] to re-use the budget left free by a task. However, this model is not always efficiently transportable to the network level. An effective adaptation is proposed by Nolte in the Server-CAN protocol [10] (Master-Slave with cyclic scheduling windows in CAN) that reclaims the bandwidth by anticipating the next cycle. In this case the Master takes advantage of the broadcast nature of the CAN network to readily infer the state of the nodes asynchronous queues. However, switched Ethernet networks support unicast and multicast traffic, and thus this mechanism in not directly applicable, and a practical implementation would require an explicit acknowledge mechanism, which would be much more complex to manage and would consume bandwidth.

- An in-band backward signaling scheme is used, e.g. EPL, where the backward information is piggybacked onto the synchronous traffic. This approach has two main problems. On one hand, additional bandwidth is used if the application does not require the node to produce any synchronous message. In that case, a synchronous message must be added just to implement the backward channel. On the other hand, the responsiveness to asynchronous requests dependends on the polling rate of the backward channel. This may also result in overhead since it may be necessary to poll synchronous message more often than necessary just to allow the polling of the asynchronous queues at rate compatible with the desired asynchronous traffic responsiveness.
- A backward signaling mechanism allowing the slaves to periodically report to the master the current status of their asynchronous queues. The master, then, only polls the asynchronous traffic when it is ready in the node queues. Provided that such backward channel is available and does not consume extra bandwidth, e.g. using out-of-band communication, this solution is optimal with respect to bandwidth utilization since all the polls correspond to an asynchronous request and no bandwidth is consumed by the signaling procedure .

In this paper we propose a signaling mechanism similar to the backward signaling mechanism above listed, i.e., using a backward channel that is, essentially, out-of-band with respect to the regular messages, exploiting the full duplex features of current Ethernet switches.

3 Out-of-band signaling scheme

In Master-Slave protocols, the traffic is scheduled and controlled by means of a Master poll message. This token may be associated with a single slave message transmission or may delimit a transmission window for multiple slave messages (Master/Multi-slave). In any case the slave nodes are synchronized by the poll message reception and, therefore, it is of utmost importance transmitting this message with no interference. This behavior is typically achieved by reserving transmission windows wide enough to ensure that all the traffic related with a transaction is completely dispatched by the switch before

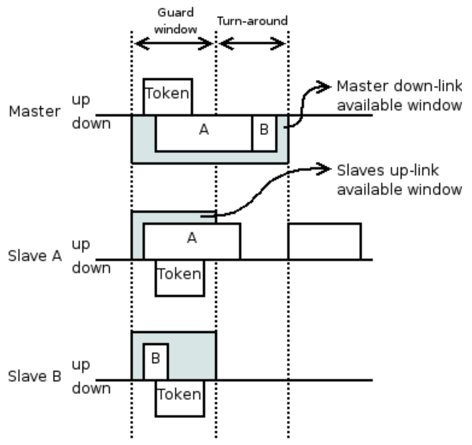


Figure 1. Token

the beginning of the subsequent transaction. In practice, this window includes the polling message and the reply transmission time, the turn-around time necessary for the slaves to decode and prepare the reply and guarding windows to compensate the jitter inherent to each one of these operations.

In a full-duplex switch, the Master download link is not included in the token switching path, i.e messages transmitted in unicast to the Master do not interfere with the actual token message transmission (see Fig. 1). Therefore, the slave nodes may safely report to the Master node the internal state of its asynchronous queues during the guarding window and the turn-around window. Provided that these messages end their transmission before the end of the turn-around window there is no impact on the regular protocol messages and the slaves turn-around time is also not affected, since the report messages are submitted to the NIC during the transmission of the poll message by the Master. Figure 1 shows one situation in which two nodes send their status messages (A and B) to the Master. Both messages are completely received within the time gap defined by the poll message transmission plus the turn-around windows, thus not interfering with the normal protocol operation.

Thus, our proposal relies on this transmission scheme to handle the backwards signaling information containing the asynchronous queuing status in the Slaves. It can be applied to any Master-Slave full-duplex Switched Ethernet protocol, as long as the Slaves are able to synchronize with the transmission instant of the following Master polls.

The following Sections describe the inclusion of this signaling mechanism in two distinct protocols, FTT-SE and EPL, which have different ways to handle the asynchronous traffic.

4 Implementation on FTT-SE

FTT-SE [9] is a RT protocol that exploits the advantages brought by the Ethernet micro-segmentation. Such advantages include a throughput enhancement by taking

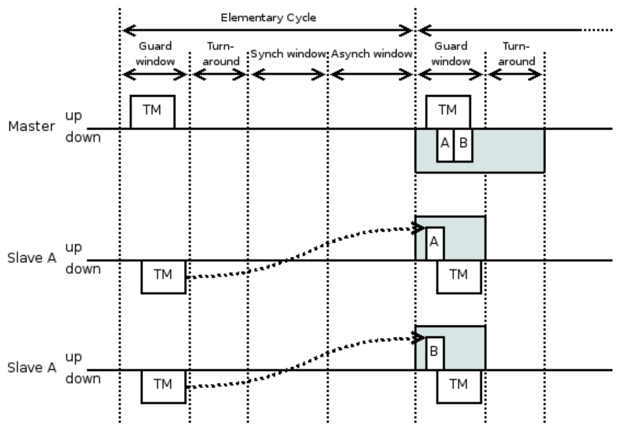


Figure 2. FTT-SE signaling proposal

advantage of the multiple forwarding paths and the full-duplex links, as well as the timing relaxation in the slaves by taking advantage of the collision free domain. In this protocol the Master poll message, known as Trigger Message (TM) is periodically transmitted, polling the Slaves to transmit messages in two consecutive and disjoint time windows, designated synchronous and asynchronous windows. The synchronous window is associated with the transmission of the synchronous (periodic) traffic, which is centrally scheduled by the Master node. The asynchronous traffic (sporadic) is transmitted in the asynchronous window. Associated with each sporadic message there is a minimum inter-transmission time ($Tmit$) that, when associated with the message size, defines the maximum bandwidth consumed by the respective message stream. The FTT-SE protocol, uses the $Tmit$ to periodically poll the asynchronous messages as if the messages had a periodic activation. However, contrarily to what happens with the periodic traffic, the poll of a sporadic message may or may not result in an actual transmission, depending on the existence or not of pending requests. Therefore, this polling mechanism consumes all the bandwidth allocated to a given sporadic message independently of the actual transmission request rate.

In the FTT-SE protocol the system granularity is stated by the *Elementary Cycle* (EC) which is headed by the TM transmission and followed by the turn-around window, Synchronous window and Asynchronous window. Figure 2 sketches the EC structure with an example of the signaling scheme herein proposed.

As described in Section 3, the signaling channel uses the reverse path taken by the periodic Master message, avoiding interfering with the normal protocol operation. Therefore, the slaves are required to synchronize themselves with the Master before being able to report its status.

Each signaling message include information regarding the asynchronous queues status of the associated node and shall not use more than the minimum payload required for an Ethernet frame. The idea is to transmit one signaling message per node in each EC. This option limits the number of nodes in the system to the ability of fitting all the messages below the TM size and the turn-around time.

The turn-around time plays a significant role in this scheme. In FTT-SE this parameter is configurable and must be properly tuned according to the node's performance. Typically must be set to a value no lower than the worst-case turn-around time among all the nodes present in the system.

The following equation allows computing the maximum number of nodes ($MaxN$) with respect to a system s with a given turn-around time, Tr:

$$MaxN(s) = \frac{Tr(s) + TM\_size(s)}{SIG\_size(s)}$$

where $TM\_size(s)$ stands for the trigger message transmission time and $SIG\_size(s)$ for the signaling message transmission time. Assuming a Fast Ethernet network (100 Mbps), $TM\_size(s)$ equals $24\mu s$ and $SIG\_size(s)$ takes the minimum Ethernet frame transmission time, which is $6.72\mu s$. Assuming also that the system nodes are based on standard PCs architectures and use a RT kernel and stack, yielding a typical turn-around time (Tr) of $200\mu s$, the maximum number of nodes ($MaxN$) is 33. Notice that for applications requiring more nodes or exhibiting lower turn-around times (specialized hardware) two approaches can be used to avoid this scalability constraint. One is to reduce the rate at which nodes issue the signaling messages. This approach implies a negative impact on the signaling latency but permits supporting an arbitrary number of nodes. Other possible approach is to extend the Master downloading window beyond the turn-around time using the synchronous window for the purpose. This would require the proper adjustment of the synchronous messages scheduler to include this extra traffic in the Master downlink but has the advantage of potentially not affecting the Slaves application since typically regular application messages are not directed to the Master.

5 Implementation on EPL

The Ethernet Powerlink protocol [2] is a Master-Slave protocol that supports both the shared and the switched Ethernet media. This protocol aims at real-time jitter-sensitive applications and enforces strict time restrictions.

Similarly to the FTT-SE protocol, the EPL protocol employs a bi-phase cyclic communication structure, with the time divided in *Elementary Cycles* (EC) and the ECs comprising an isochronous and an asynchronous phase. Heading the EC, a Start of Cycle poll message (SoC) indicates the beginning of the isochronous phase. Thereafter, for each message (Pres) polled in the isochronous phase, a Master poll is transmitted (Preq). Each individual polling is issued in a specific time slot within the isochronous phase to reduce the message jitter. The number and size of the slots is parametrized offline. The asynchronous phase

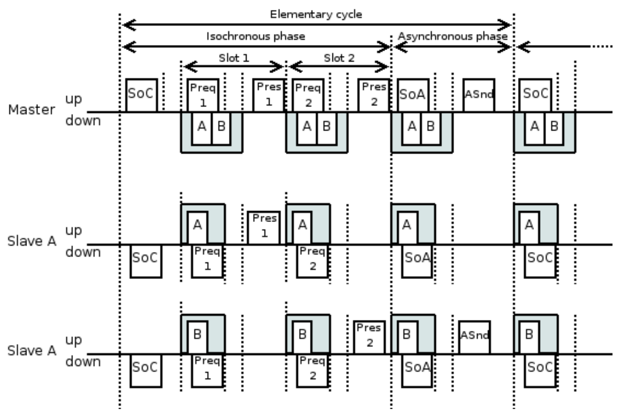


Figure 3. EPL signaling proposal

starts with a Master message (SoA) polling a single asynchronous message in each EC.

The asynchronous traffic is totally managed in the *Managing Node* (MN) after explicit transmission requests by the *Controlled Nodes* (CN). The request for transmission can be either sent piggybacked in the synchronous messages or when explicitly requested by the MN, using in this latter case a specific asynchronous message for the purpose. The asynchronous requests include also the requested message priority. The MN keeps track of all pending requests, including their owns, and in the asynchronous phase of each EC schedules the highest priority one for transmission, as explained above.

Besides the explicit AM status requests, the EPL protocol also considers the polling of the nodes identification to verify the network status and take the necessary procedures on connectivity events.

The out-of-band signaling mechanism described in Section 3 may also be used in EPL both to reduce the signaling latency and minimize the protocol overhead due to the explicit polling for the asynchronous status and nodes connectivity or due to the over-scheduling of isochronous messages for faster signaling purposes. Figure 3 sketches an EPL elementary cycle including the signaling scheme. Like in the FTT-SE approach, the Slaves are synchronized with all the Master polls so that the signaling messages can be unicasted to the Master.

6 Comparative analysis

The $Tmit$ parameter sets the maximum transmission rate that a given AM may have and, in conjunction with the message length, bounds the message maximum bandwidth utilization ($C/Tmit$). However, frequently the sporadic messages have average activation rates significantly lower than the maximum one. With the backward signaling scheme the network bandwidth resulting from the missed activations can be reclaimed. The RT traffic is typically admitted against the worst-case scenario, and thus reclaimed extra bandwidth does not result in better schedulability levels for the RT assynchronous traffic. However, the reclaimed bandwidth allows for reducing the average response time of those AMs, since the activations absent anticipates the pools for the lower priority traffic. Furthermore, the reclaimed bandwidth may also be used by the background non-RT traffic, which can see its average throughput significantly improved.

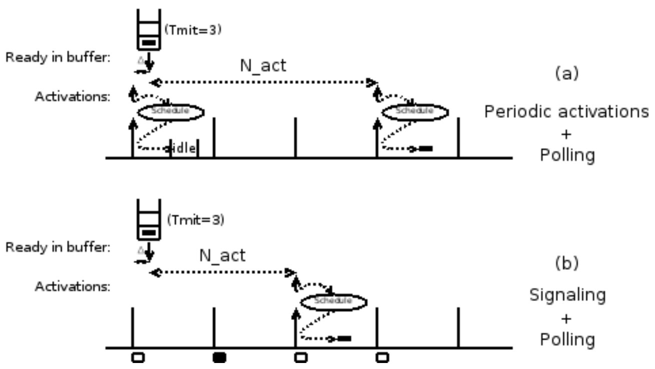


Figure 4. FTT-SE activations response time

Besides the average gains in bandwidth and response time, the backward signaling scheme may also have a positive impact on the runtime schedulability of the asynchronous RT messages due to a potentially prompter acquisition of the nodes state by the Master node. Let us define the *activation delay* (N_act) as the time that goes between the AM deployment in a Slave queue and its inclusion in the Master scheduler. The activation delay is part of the AM response time (AM_Rt), which can be computed as follows:

$$AM\_Rt = N\_act + Jsch + C$$

where C stands for the AM size and $Jsch$ for the scheduling jitter which varies with the scheduling algorithm and the applied load. Therefore, taking two similar scenarios differing only in the applied signaling mechanism and maintaining the same scheduler and load, it is possible to evaluate the impact on the AM_Rt due the N_act variation.

Taking the FTT-SE implementation case study, Fig. 4 sketches a timeline with the worst case situation for the *activation delay* (N_act) in two scenarios. In (a) it is used a pure polling mechanism where the master periodically schedules the AM in a blind way, while in (b) it is represented the backward signaling mechanism operation. In both scenarios the asynchronous message is registered with a minimum inter-transmission time of 3 ECs.

When activations are periodically triggered (a), the Slave may ultimately have an AM transmission request right after the last poll, leading to an $N\_act = Tmit - \Delta = 3 * LEC - \Delta$, where Δ is an infinitesimal and LEC in the EC length. In this scenario the activation delay N_act varies linearly with message's $Tmit$. However, when the backward signaling scheme is applied (b), once an AM transmission in requested it produces a signal that is transported in the following EC and is then included in the Master scheduler. The worst-case delay is, in this case, independent on the message's $Tmit$

and fixed to $N\_act = 2 * LEC - \Delta$. The message activation signal takes at worst case 1 EC to reach the Master. After reaching the Master the request is eligible by the scheduler and the schedule result is dispatched on the following EC.

The best case activation delay happens in (a) when the message is queued right before its poll ($N\_act = \Delta$), and in (b), when it is queued before the signaling message ($N\_act = 1 * LEC + \Delta$). Assuming that the messages are asynchronous and randomly triggered by the application we may preview a uniform distribution in the queuing events, leading to an average delay between the best and the worst cases.

Table 1. Generalized activation delays

scenario	Worst Case	Best Case	Average
(a)	$Tmit$	0	$Tmit/2$
(b)	$2 * LEC$	$1 * LEC$	$1.5 * LEC$

Table 1 outlines the activation delays for the two depicted scenarios and for the average case. In the assumed average condition, we may see that the proposed signaling scheme (b) induces a better asynchronous responsiveness for discrete values of $Tmit$ greater than 2 ECs. It is thus obvious the benefits for messages registered with long inter-transmission times.

7 Conclusions

The advent of switched Ethernet has opened new perspectives for real-time communications over Ethernet. However, a few problems subsist related with queue management policies, queue overflows and limited priority support. To overcome such difficulties several real-time protocols were proposed. Some of these are based in the Master-Slave paradigm, which, despite exhibiting many interesting properties like simplicity of operation and deployment and good control over the communication medium, is also known by its inefficient handling of the asynchronous traffic. This paper presents a novel signaling mechanism, suitable for master-slave protocols based on full-duplex micro-segmented switched Ethernet networks. It allows the slave nodes to periodically inform the Master about their status in particular time instants during which the communication path between the Slaves and the Master is idle, thus without interfering with the normal protocol operation. The inclusion of this signaling mechanism brings important advantages both in terms of the asynchronous messages responsiveness and protocol overhead. The paper illustrates how the mechanism can be applied to the FTT-SE and EPL RT protocols and shows the potential gains in terms of responsiveness and overhead reduction. The signaling of the AMs status is the most direct use for the permanent backward channel, however, as future work it is planned extending the use of this channel for other protocol operations like configuration during setup phase, supporting e.g. plug&play protocol.

References

[1] EtherCAT - Ethernet for Control Automation Technology specifications. http://www.ethercat.org/.

[2] Ethernet Powerlink protocol home page. http://www.ethernet-powerlink.org/.

[3] Ethernet/IP (Industrial Protocol) specifications. http://www.odva.org.

[4] Real-Time PROFINET IRT. http://www.profibus.com/.

[5] J. Decotignie. A perspective on Ethernet as a fieldbus. In *Proceedings of FeT'2001, 4^{th} International Conference on Fieldbus Systems and Their Applications Nancy, France*, pages 138–143, 2001.

[6] Deepu C. Thomas, Sathish Gopalakrishnan, Marco Caccamo, and Chang-Gun Lee. Spare CASH: Reclaiming Holes to Minimize Aperiodic Response Times in a Firm Real-Time Environment. In *Proc. on the 17th Euromicro Conference on Real-Time Systems (ECRTS'05).*, pages 147–156. IEEE, July 2005.

[7] J. Loeser and H. Haertig. Using Switched Ethernet for Hard Real-Time Communication. In *Proc Parallel Computing in Electrical Engineering, International Conference on (PARELEC'04)*, pages 349–353, Dresden, Germany, Sept. 2004.

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

[9] P. Pedreiras, P. Gai, L. Almeida, and G. C. Buttazzo. FTT-Ethernet: a flexible real-time communication protocol that supports dynamic QoS management on Ethernet-based systems. *IEEE Transactions on Industrial Informatics*, 1(3):162–172, Aug. 2005. ISSN: 1551-3203.

[10] Thomas Nolte, Mikael Nolin, and Hans Hansson. Server-based scheduling of the CAN bus. In *Proc of the 9th IEEE Int. Conf. on Emerging Technologies and Factory Automation (ETFA'03)*, pages 169–176 (volume1), Lisbon, Portugal, Sept. 2003. IEEE Industrial Electronics Society.

Position Paper on Dependability and Reconfigurability in Distributed Embedded Systems

Julián Proenza
SRV, Dept. de Matemàtiques i Informàtica
Universitat de les Illes Balears, Spain
julian.proenza@uib.es

Luís Almeida
IEETA, DETI
Universidade de Aveiro, Portugal
lda@det.ua.pt

Abstract

Dynamic Reconfiguration (DR) has been generating a substantial interest since it allows improving efficiency in the use of system resources, which can impact both on the maximum functionality that the system can execute, on the level of resources needed for a given functionality, on the number of instantaneous users that the system can support, or even on the capacity to adapt to changes in the environment or on the system operational architecture such as those caused by hazardous events. However, DR also requires extra mechanisms to manage the reconfiguration itself, which can increase system complexity and reduce a priori knowledge, increasing the potential for lower reliability. Therefore, DR has not been considered in safety-critical systems. In this paper we argue that adequately preventing specific error situations at the lower levels of the architecture simplifies the upper-level systemwide Fault Tolerance mechanisms, and may compensate for the extra complexity and lower a priori knowledge that DR implies, thus opening the door to the construction of highly-reliable dynamically reconfigurable systems.

1. Introduction

Reconfigurability has long been recognized as a way to improve efficiency in the use of system resources [22], for example, when a system undergoes variable load situations, when it evolves during its lifetime or even when faults affect part of its structure [25]. This means that reconfigurability, in a broad sense, may be beneficial to areas that range from Quality of Service (QoS), e.g., when the number of system users varies [20], to Dependability, e.g., through *graceful degradation* [26].

However, achieving reconfigurability may conflict with operational goals such as continued real-time and safe operation, and it becomes more difficult when the system is distributed, requiring adequate support from the network. Hence, whenever either of those two operational goals are relevant, the typical option has been to rely on a single static configuration [7][27]. In some cases, reconfigurable solutions have been devised but limited to few predefined operational modes, thus still with reduced flexibility and efficiency [27]. Conversely, for QoS purposes, reconfigurability seems to bring along clear benefits [20] and the conflicting goals referred for the case of safety critical systems do not seem to apply.

In this paper we discuss the interaction between Dynamic Reconfiguration (DR) and Fault Tolerance. We show that, despite its higher efficiency, DR introduces new dimensions in the system state space and, mainly, new mechanisms to mediate and enforce the system state changes. In other words, DR reduces the a priori knowledge concerning the exact state the system is in and introduces extra mechanisms that may lead to higher complexity and thus lower reliability. We, then, discuss ways to compensate for such negative aspects and propose using hardware-implemented mechanisms to prevent specific error situations at the lowest levels of the architecture, in order to simplify the upper-level systemwide Fault Tolerance mechanisms and improve their coverage. Several examples of recent related work will be referred. The remainder of the paper is organized as follows, Section 2 discusses the definition of DR, Section 3 discusses how DR is sometimes used to improve the system dependability making use of existing system resources, Section 4 discusses the limitations of DR in what concerns Fault Tolerance aspects and Section 5 presents the proposed solutions. Section 6 concludes the paper.

2. On the concept of DR

DR is a broad concept that spans many application domains. A common or integrated perspective of DR, including a taxonomy and boundaries of what is and what is not DR is still to be done despite recent initiatives in that direction [1]. Concerning this paper, it is important to separate the concepts of DR and Fault Tolerance. Notice that Fault

Tolerance mechanisms typically involve some kind of on-line reconfiguration, e.g., disconnecting nodes affected by faults and replacing them with spares or adding new nodes to compensate for disconnected replicas. In this sense, Fault Tolerance mechanisms are a subset of DR. However, these mechanisms normally aim at maintaining the same functionality, only, despite the occurrence of faults.

On the other hand, DR is normally taken in a broader way, considering changes in the allocation of tasks and messages to resources (e.g. nodes, links, bandwidth and energy), or even changes in the operational parameters of the system (e.g. scheduling and control parameters). The purpose of DR is typically to improve resources usage, considering the resources that are already available because they are needed for the basic functionality of the system. It is also common to consider that DR implies a high level of flexibility / adaptability in the system.

Therefore, Fault Tolerance mechanisms based on replication are not normally taken as DR. On the other hand, as discussed in the next Section, DR can still be used to improve the dependability of systems that were not designed as traditional fault-tolerant systems., e.g. by reallocating system resources that were currently available, possibly providing *graceful degradation*.

3. Improving dependability with DR

The idea of taking advantage of the already available resources relates DR to the low-cost Fault Tolerance approach of using *Unintentional Redundancy* [17]. This kind of redundancy is usually available in all systems (particularly in distributed ones). We illustrate next how Unintentional Redundancy with DR can be used to tolerate faults.

First, in some interconnection topologies, such as the one in Fig. 1, there are several paths that can be used to connect each pair of nodes. This opens doors for dynamically reconfigurable architectures that, in case of failure of one link, reestablish the communication using an alternate path. For example, in Fig. 1, in case the direct link between nodes 1 and 2 is faulty, the communication between these two nodes can be reestablished though nodes 3 and 4. A specific example of this kind of DR is described in [3].

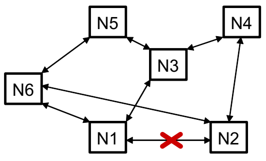


Figure 1. A network performing DR

Second, in general distributed systems, the presence of multiple nodes enables the reallocation of tasks from faulty nodes to non-faulty ones. For example, in Fig. 2, in case node 2 is known to be faulty its tasks could be assigned to another node. An example of such type of reconfiguration in an automotive system has been pointed out in [19]

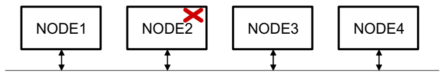


Figure 2. Dynamic reallocation

4. Limitations of DR

Obviously, the use of Unintentional Redundancy with DR has a limited capacity for Fault Tolerance. Among the reasons for this, we point out two. First, depending on the system, the available redundancy can be not enough to tolerate some faults. Therefore single points of failure may still exist. And second, in many cases the reconfiguration causes a *graceful degradation*, which means that either the level of performance has been decreased or even the system is no longer fully functional and some supposedly non-critical functions have been shutdown. Notice that graceful degradation is, normally, a positive feature in the sense that what would be a global failure is exchanged by an operating configuration that still provides a reduced level of QoS. Nevertheless, for certain critical systems that reduced level of QoS might not be sufficient to meet the minimum operational requirements.

Therefore, when using Unintentional Redundancy, DR is a suitable means to achieve a general improvement of the system dependability, but it might be not so well suited to reach the high levels of dependability that are usually pursued by fault-tolerant architectures. Moreover, dynamic reconfigurations are likely to take some time to complete, which, depending on the specific nature of the system, might also be constrained by the dynamics of the environment. This might rule out the use of DR in critical systems with fast dynamics unless special architectures that support fast reconfiguration are used, e.g. [12].

4.1. Addition of resources for DR

Despite the efficiency improvement in using already available resources being one of the typical characteristics of DR, the truth is that some resources must be added to a system for it to be able to perform DR. In particular, many systems that perform DR do it thanks to the inclusion of suitable mechanisms in their middleware. Moreover, some hardware additions can be also used, which, as will be discussed below, may provide an advantageous support for a safe reconfiguration.

However, too many additions would represent a deviation from the initial target of achieving low cost by efficiently using the available resources. Therefore, a trade-off has to be found between cost and functionality.

4.2. DR means flexibility, complexity and overhead

As referred before, DR is supposed to imply a (high) flexibility in the system. But, flexibility also means complexity. In the case of a distributed system, this complexity appears in two forms. First, nodes have to perform new actions for the system to be able to react in the face of various situations and to adapt to the ever changing reality. As indicated above, these new actions are usually implemented in the middleware. And second, the communication channel has to transport an increased number of messages, e.g. to coordinate the reconfiguration among nodes.

Therefore, DR normally introduces a computational and communication overhead that may be not acceptable for many distributed embedded systems based on low-performance microcontrollers and low-bandwidth communication technologies. Moreover, beyond these overheads, an increased complexity usually means a decreased reliability.

4.3. Facets of unreliability in DR

The unreliability caused by the increased complexity of systems performing DR is essentially provoked by the increased number of scenarios that the system has to be designed to deal with, arising from the multiple possible configurations that the system can adopt and the faults that the system must react to.

The presence of these multiple scenarios makes it much more difficult to achieve the so-called *systemwide integration of fault tolerance*. This integration is pointed out in [2] as one of the fundamental steps in the design of complex fault-tolerant systems, since those are prone to suffer failures caused by improper interactions among their non-faulty subsystems. Some examples of parts that are difficult to integrate are the Fault Tolerance mechanisms that are intended to deal with the local faults of each subsystem with those that provide Fault Tolerance for functions that are executed as a global cooperation among several subsystems. This systemwide integration also has to prevent improper interference among concurrently active recovery or reconfiguration algorithms.

Similarly the increased number of scenarios to deal with also make it difficult the qualitative evaluation [2] of the system. This kind of evaluation is intended to verify that the design of the system includes all the mechanisms which are necessary to deal with the expected classes of faults. The higher the number of error scenarios is, the higher the difficulty in verifying the correct operation of the system in all these scenarios will be. More specifically, *model checking* [8] is likely to become a standard evaluation procedure for fault-tolerant systems in the next few years, much in the same way as simulation is already a de facto requirement in the development of computing systems. Although modern model checkers such as UPPAAL [18] already exhibit an enhanced capacity to deal with large state spaces, the nature of model checking makes this technique quite vulnerable to the complexity of the systems to be modeled and verified. The amount of memory required to generate the state space of complex models makes model checking useless in practice for the verification of highly-complex systems.

For all the reasons discussed above, keeping the complexity of a system performing DR under reasonable bounds should receive the maximum attention when dependability is the main concern.

5. Reconciling DR and Fault Tolerance

A way of reducing the complexity, and thereby increasing the reliability, is to reduce the number of scenarios that the system has to deal with, in particular those scenarios created by the faults that the system has to tolerate.

The techniques to be used in order to achieve this reduction of scenarios are implemented in the form of hardware additions that prevent specific error situations as close as possible to the faults that generate them. Since part of the errors that the subsystems can suffer are resolved as close as possible to their origin, the upper layers of the system architecture, such as the middleware, are much less complex and more reliable for they do not have to deal with the aforementioned error situations or only have to deal with simplified ones.

Examples of these techniques are, first, the use of specific circuitry to restrict the *failure semantics* [9] of the nodes. If Byzantine or arbitrary failures of the nodes are not possible, the software of the other nodes does not have to deal with them. And second, the use of hardware-implemented communication protocols that provide consistent communication services. Thereby the communication channel does not cause additional complex scenarios, e.g., inconsistencies in the delivery of messages, to be resolved by the upper layers of the architecture. For instance, the *MajorCAN* protocol [23] is a slightly modified version of the *Controller Area Network* (CAN) [14] that truly provides atomic broadcast at the data-link layer.

5.1. Defining error containment boundaries

An adequate restriction of the nodes failure semantics has the additional effect of preventing a node from acting as a *babbling idiot* [16] that, by sending messages at the wrong moment, blocks the communication channel and impedes the exchange of messages among nodes.

In more general terms, the restriction of the failure of semantics is a significant help to prevent the propagation of errors from a faulty node to the rest of them. In the same manner, the use of hardware-implemented communication protocols that provide consistent communication services also helps to prevent that errors in the communication are propagated to the receiving nodes.

However it is important to note that the failure semantics restriction together with the use of consistent communication services is not enough in order to completely define an *error containment boundary* [2] around each node (Fig. 3) because, after all, even if we restrict the failure semantics of a node, it is possible for it to suffer a failure. In order to cope with these situations it is also very important to design the other nodes' software (e.g. middleware) in such a way that they are able to recognize these failures and to properly react to avoid their effects. This is much easier to achieve when the failure semantics is restricted, e.g. if nodes present crash failure semantics, node failures can be detected by timing out on regularly transmitted *I am alive* messages. Designing the global operation of the system to work properly in the event of node failures is an additional concern of the design of any truly fault-tolerant system.

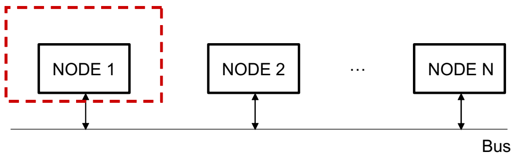


Figure 3. Error containment boundaries

5.2. Advantages of defining low-level error containment boundaries

Beyond the aspects that have been already discussed (e.g. reduction of the middleware complexity and thus increase of its reliability), using techniques at the lower levels of the system architecture to prevent error propagation exhibits a number of additional relevant advantages.

First, it reduces the overhead generated by the communication since no higher-layer protocols are required in order to ensure consistency. This has two facets, on the one hand less messages are transmitted and, on the other hand, less computation time is devoted in the nodes to the tasks related to communication.

Second, less nodes are required. Since failure semantics are restricted, the requirements on the number of nodes to be used in order to achieve agreement on a value under byzantine failure semantics [10] are relaxed.

And third, it prevents the so-called *amplification of failures* [13]. A typical high-level implemented consistent communication service requiring several rounds of message transmissions uses lower-level (and less dependable) communication primitives, such as point-to-point message *sends* and *receives* [13]. In this kind of high-level communication services, the broadcast of a message requires the execution of several instructions, and may include several sends and receives. It is well known that in this kind of complex communication schemes a failure at the low level of send and receive primitives (e.g. an omission to send a message) does not necessarily manifest at the high level as the same type of failure (e.g. an omission to broadcast a message to all receivers). In fact, it is said that this kind of broadcast algorithms are likely to amplify the importance of failures which occur at the low level [13] (e.g. messages delivered to different receivers in a non consistent order due to an omission to send a message). Therefore by substituting this kind of high-level implemented protocols by low-level services already presenting the required properties we would make it possible to eliminate this risk.

It is important to note that the set of advantages pointed out above are achieved when both failure semantics restriction and low-layer consistent communication services are used at the same time. However we do not claim that both features must always be included in the architecture. Even if only one of the features is used, significant reduction of the potential error scenarios is obtained.

5.3. Failure semantics restriction implementation

When using any mechanism to restrict the failure semantics of the nodes it is very important to achieve a high probability for the final node design to exhibit the pursued failure semantics, i.e., to have an *assumption coverage* [21] as high as possible. After all, the design of the rest of the distributed system is based on assuming said failure semantics.

To enforce a restriction in the failure semantics of the nodes several design techniques can be used. One of the most effective ones is *duplication with comparison* [15]. This technique is based on using two identical pieces of hardware actively performing the same operations in parallel, and comparing the results of said operations. In case there is a discrepancy in the results, an error signal is activated. This allows the detection of errors in the duplicated module and can be used to restrict the failure semantics by using the error signal to disconnect the node from the network (e.g. disabling the communication transceiver). This scheme is shown in Fig. 4. In this simple manner a *crash failure semantics* is enforced, meaning that the node either works properly or crashes. For a high assumption coverage to be achieved using this technique it is important to duplicate as much parts of the circuitry as possible. For example, [24] and in [12] describe CAN nodes with internal duplication and comparison that can disable their transceivers in case of discrepancy.

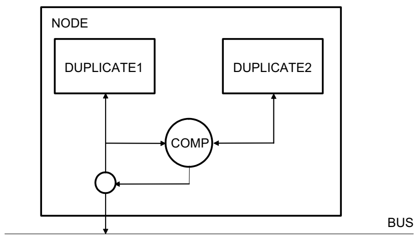


Figure 4. Duplication with comparison for failure semantics restriction

Another technique for restricting the failure semantics of a node in a bus-based architecture is the incorporation of a *bus guardian* [11]. This mechanism is devoted to prevent a node acting as a babbling idiot from keeping the bus busy with the transmission of useless messages, thereby making it more difficult for the other nodes to communicate. Several different bus guardians have been reported in the literature, either for time-triggered and event-triggered communication as well as for flexible communication requirements such as the one described in [12] for the FTT-CAN protocol.

5.4. Other error containment techniques

Having to modify the structure of each of the nodes is unacceptable in some applications due to cost reasons. In these cases a different approach for failure semantics restriction can be adopted. This new approach is based on the use of star topologies instead of buses. Such a topology allows to include mechanisms for error detection and *fault passivation* [17] (e.g. disconnecting the faulty nodes from the network) in the hub and, thereby, standard hardware can be used for the nodes. Fig. 5 illustrates this idea. By placing the error containment mechanisms into the hub and by preparing all nodes to deal with the scenarios caused by faulty nodes an equivalent definition of error containment boundaries can be achieved. Note that the hub may prevent the propagation of errors generated either in the nodes, in the links that connect each node with the hub or even in the circuits that implement the communication protocol.

Besides the capacity of an active hub to enforce a restricted failure semantics for each node of the network, the aforementioned capacity of preventing the propagation of the errors caused by faults in the circuits that implement the communication protocol (including cables) is a significant advantage of the star topology when it is compared to a bus [6]. The main drawback of the bus topology is that the structure of the network presents multiple components, which have direct electrical connections to each other without proper error containment. As a consequence, a fault in

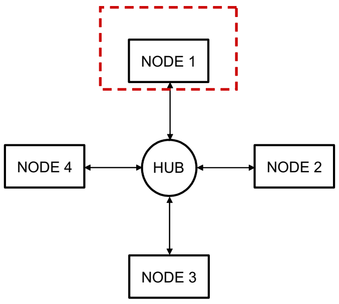


Figure 5. A star for error containment

any of them may generate errors that propagate and effectively prevent further communication to take place.

In contrast, a star topology reduces this multiplicity of potentially failure-generating components to only one, the hub. Although it is clear that a star significantly reduces the probability of having more than one node affected by a failure (most of the errors caused by a single faulty component are not allowed to affect more than one node) [5], in some applications the presence of the single point of failure that the hub represents is unacceptable. In these cases redundant star topologies can be used [4].

5.5. Bringing it together

As referred before, we consider that it is possible to reconcile Dynamic Reconfiguration with high levels of Fault Tolerance as long as nodes failure semantics restriction and low-layer consistent communication services are incorporated into the system design from the beginning. Most of the actual components needed for the particular case of CAN technology have been developed by our groups along the past 8 years. Specifically, we consider that the FTT-CAN protocol is naturally adapted to support prompt reconfiguration under continued timeliness and that the mechanisms presented in [12] can be further simplified and the overall Fault Tolerance features improved, mainly concerning their analyzability, by merging that protocol with MajorCAN [23] to achieve true atomic broadcast, and the (Re)CANcentrate hubs [4] for strong error containment.

6. Conclusions

Dynamic Reconfiguration is gaining a growing interest as a way to improve the efficiency in using system resources. Moreover, DR has also been pointed out as a way to achieve inexpensive Fault Tolerance, e.g., by means of graceful degradation. However, combining DR with high levels of FT raises several problems, mostly related with the reduced a priori knowledge of DR systems and with the reconfiguration mechanisms themselves that introduce extra complexity and overhead, thus lower reliability. In this paper we have discussed the interaction between DR and FT and we have proposed combining nodes failure semantics restriction and low-layer consistent communication services to simplify the system middleware layers and improve their analyzability. This will allow building highly reliable and resource efficient systems that are capable of adapting to the environment, to systems changes or to different load situations while tolerating the designated faults.

References

[1] Neres 2007 - artist2 workshop on networks for reconfigurable embedded systems. `http://www.artist-embedded.org/artist/-NERES-2007-.html`, April 2007.

[2] A. Avižienis. Building dependable systems: How to keep up with complexity. In *Special Issue of the IEEE 25th Int. Symp. Fault-Tolerant Computing. FTCS-25. Pasadena, CA*, pages 4–14, June 27–30 1995.

[3] D. Avresky and N. Natchev. Dynamic reconfiguration in computer clusters with irregular topologies in the presence of multiple node and link failures. *IEEE Transactions on Computers*, 54(5):603–615, May 2005.

[4] M. Barranco, L. Almeida, and J. Proenza. ReCANcentrate: A replicated star topology for CAN networks. In *Proceedings of the 2005 IEEE Conference on Emerging Technologies and Factory Automation (ETFA 2005). Catania, Italy*, 2005.

[5] M. Barranco, J. Proenza, and L. Almeida. First results of the assessment of the improvement of error containment achieved by CANcentrate. In *Proceedings of the 6th IEEE International Workshop on Factory Communication Systems (WFCS 2006). Torino, Italy*, 2006.

[6] M. Barranco, J. Proenza, G. Rodríguez-Navas, and L. Almeida. An active star topology for improving fault confinement in CAN networks. *IEEE Transactions on Industrial Informatics*, 2(2):78–85, May 2006.

[7] Belschner, R. *et al*. FlexRay Requirements Specification, version 2.0.2. *FlexRay Consortium, `http://www.flexray-group.com`*, 2002.

[8] E. Clarke, O. Grumberg, and D. Peled. *Model Checking*. The MIT Press, 1999.

[9] F. Cristian. Questions to ask when designing or attempting to understand a fault-tolerant distributed system. In *Keynote Address in Proc. 3rd Brazilian Conference on Fault-Tolerant Computing. Rio de Janeiro, Brazil*, September 1989.

[10] D. Dolev. The byzantine generals strike again. *Journal of Algorithms*, 3(1):14–30, 1982.

[11] J. Ferreira, L. Almeida, and J. Fonseca. Bus guardians for can: a taxonomy and a comparative study. In *Proc. of WDAS 2005, Workshop on Dependable Automation Systems*. Brazilian Computing Society, October 2005.

[12] J. Ferreira, L. Almeida, J. Fonseca, P. Pedreiras, E. Martins, G. Rodriguez-Navas, J. Rigo, and J. Proenza. Combining operational flexibility and dependability in FTT-CAN. *IEEE Transactions on Industrial Informatics*, 2(2):95–102, May 2006.

[13] V. Hadzilacos and S. Toueg. Fault-tolerant broadcasts and related problems. In S. J. Mullender, editor, *Distributed Systems*, ACM-Press, chapter 5, pages 97–145. Addison-Wesley, second edition, 1993.

[14] ISO. *International Standard 11898 – Road Vehicles – Interchange of Digital Information – Controller Area Network (CAN) for High-Speed Communication*. 1993.

[15] B. W. Johnson. *Design and Analysis of Fault Tolerant Digital Systems*. Addison-Wesley Publishing Company, 1989.

[16] H. Kopetz. A node as a unit of failure. In *Real-Time Systems: Design Principles for Distributed Embedded Applications*, Real-Time Systems. Engineering and Computer Science, chapter 6.3, pages 129–131. Kluwer Academic Publishers, Boston, Dordrecht, London, 1997.

[17] J.-C. Laprie, editor. *Dependability: Basic Concepts and Terminology*. Springer-Verlag Wien New York, 1992.

[18] K. G. Larsen, P. Pettersson, and W. Yi. UPPAAL in a Nutshell. *Int. Journal on Software Tools for Technology Transfer*, 1(1–2):134–152, Oct. 1997.

[19] J. Li, Y. Song, and F. Simonot-Lion. Providing real-time applications with graceful degradation of qos and fault tolerance according to (m,k)-firm model. *IEEE Transactions on Industrial Informatics*, 2(2):112–119, May 2006.

[20] R. Moghal and M. Mian. Adaptive QoS-Based Resource Allocation in Distributed Multimedia Systems. In *Proceedings of Workshop on Parallel and Distributed Real-Time Systems (WPDRTS)*, 2003.

[21] D. Powell. Failure mode assumptions and assumption coverage. In *Digest of Papers of the IEEE 22th Int. Symp. Fault-Tolerant Computing FTCS-22*, pages 386–395, Boston, Massachusetts-USA, July 1992.

[22] D. Prasad, A. Burns, and M. Atkins. The Valid Use of Utility in Adaptive Real-Time Systems. *Real-Time Systems*, 25(2-3):277–296, 2003.

[23] J. Proenza and J. Miro-Julia. MajorCAN: A modification to the Controller Area Network protocol to achieve Atomic Broadcast. In *Proceedings of the IEEE Int. Workshop on Group Communications and Computations. IWGCC. Taipei, Taiwan*, April 2000.

[24] J. Proenza, J. Pons, and J. Miro-Julia. A low-cost fail-safe circuit for fault-tolerant control systems. In *Proceedings of the 6th IEEE Int. Conf. on Electronics, Circuits and Systems . ICECS'99. Pafos, Cyprus*, September 1999.

[25] D. Schmidt, R. Schantz, M. Masters, J. Cross, D. Sharp, and L. DiPalma. Towards Adaptive and Reflective Middleware for Network-Centric Combat Systems, CrossTalk, November. 2001. `http://www.cs.wustl.edu/~schmidt/PDF/crosstalk.pdf`; accessed February 21, 2005., 2001.

[26] C. Shelton and P. Koopman. Improving system dependability with functional alternatives. In *Proceedings of the 2004 International Conference on Dependable Systems and Networks (DSN'04)*, page 295. IEEE Computer Society, 2004.

[27] TTTech. Time-Triggered Protocol TTP/C High-Level Specification Document (edition 1.0). *`http://www.ttagroup.org`*, 2002.

An approach to enhance the QoS support to real-time traffic on IEEE 802.11e networks

Salvatore Vittorio, Lucia Lo Bello
RETISNET Lab
Department of Computer Engineering and Telecommunications
University of Catania
Catania, ITALY
lucia.lobello@unict.it

***Abstract*-This paper proposes an approach to overcome some limitations of the 802.11e protocol highlighted in recent literature and improve the QoS support provided to real-time industrial traffic. The proposed solution does not change the IEEE 802.11e protocol, but introduces a technique to reduce the number of collisions and therefore to use the channel more efficiently for real-time traffic, especially when the traffic load is high and approaches saturation conditions.**

The proposed mechanism, called a Contention Window Adapter, dynamically changes the contention window size of the different Access Categories defined by the IEEE 802.11e protocol according to the wokload conditions of the wireless network. The paper describes the rationale behind the CWA mechanism, the algorithm itself and discusses the performance obtained through simulations run using ns-2.

1. Introduction and motivation

The final version of the 802.11e standard, released by the IEEE Task Group E in 2005 [1], introduces two new access mechanisms, i.e., the Enhanced Distributed Channel Access (EDCA) and the Hybrid Coordination Function (HCF) Controlled Access (HCCA). These mechanisms correspond to the ones already present in the IEEE 802.11a/b/g standards, but provide differentiated levels of Quality of Service (QoS) to the supported applications. The EDCA mode extends the IEEE 802.11 Distributed Coordination Function (DCF) [2] by differentiating traffic into four Access Categories (ACs), mapped into the priorities defined by the 802.1D standard [3] as follows:

- AC_BK (background category) for priorities 1 and 2;
- AC_BE (best-effort category) for priorities 0 and 3;
- AC_VI (video category) for priorities 4 and 5;
- AC_VO (voice category) for priorities 6 and 7.

AC_VO is the highest priority category, while AC_BK is the lowest. Each frame arriving at the MAC layer with a defined priority will be mapped into one of the ACs. A dedicated queue for each AC exists, and different service levels are provided to each queue, based on the Arbitration Inter-Frame Space (AIFS), the Contention Window size (CW) and the Transmission Opportunities (TXOP) time interval.

Each AC has its own backoff procedure. According to the values of the minimum Contention Window size (*CWmin*) and the maximum Contention Window size (*CWmax*) which, according to the standard are statically set, each AC has a different probability of accessing the channel. Such a probability is higher for the highest priority AC_VO, which features the lowest *CWmin, CWmax* values. The differentiation mechanism provided by the standard offers advantages in terms of delay, jitter and throughput for the AC_VO class: However, recent literature outlined some limitation of the 802.11e protocol when different kinds of traffic are supported on the same channel and the total offered workload is high.

Recent work [4] showed through simulations that the default parameter values of the EDCA mode are not able to guarantee industrial communication timing requirements, when the AC_VO class is used to support real-time traffic in shared medium environments, where other types of traffic are present. The paper concludes stating that new communication approaches must be devised in order to adopt IEEE 802.11e networks on the factory floor. The work in [5] showed that, even in the presence of traffic from the highest priority class only (i.e. AC_VO), according to the amount of real-time traffic and the size of packets in this class, the real-time performance can rapidly and significantly deteriorate with growing workloads. This is due to the *CWmin* and *Cwmax* settings provided by the standard for the AC_VO class, which determine a narrow range of backoff values for the packets in this class. In [5] it was shown that it is beneficial to adapt *CWmin* and *Cwmax* for the AC_VO class to allow for a larger spectrum of backoff values, thus reducing the number of collisions inside the AC_VO class. The approach was evaluated in an industrial scenario where periodic traffic was exchanged and the CWA was applied to the Contention Window size of the AC_VO class only. Simulation results in [5] showed that CWA outperforms the 802.11e standard both as far as throughput and deadline miss ratio are concerned. The reason for this is that it reduces the number of collisions in the AC_VO class.

Here, the approach in [5] is extended to address a general industrial scenario, where Voice, Video and

Background traffic generated by Workstations (WS) is exchanged on the same channel on which field nodes send small-sized periodic RT control traffic with tight deadlines. Our aim is to improve the performance in terms of RT throughput, delay, number of collisions experienced by industrial RT traffic (consisting of small sized periodic packets exchanged between sensors, PLCs, actuators) when generic workstations generate other kind of traffic (multimedia, background) on the same channel. Here we map the industrial RT traffic on the AC_VO class provided by the 802.11e standard.

The targeted extension is not trivial. First, it has to be considered that, when different ACs are to be supported, if the CW_{max} of the highest priority class is increased, the CW_{min} of the lower priority ACs has to be accordingly set, so as to enforce the different QoS support offered to the each AC according to the standard. The CWA has therefore to dynamically tune the $\{CW_{min}, Cw_{max}\}$ range of the different ACs defined in the IEEE 802.11e standard [1] in order to reduce the potential interference (in terms of collisions) that real-time traffic, here mapped into the highest priority class AC_VO, could suffer from other lower-priority ACs. Second, if there is a station which transmits only traffic with a priority other than the highest (AC_VO), the approach in [5] cannot react on the basis of the collisions affecting that type of traffic, as the adaptation mechanism in [5] is defined for transmissions in the AC_VO class only. The CWA extension proposed in this paper addresses both the above mentioned points.

Here we underline that the proposed mechanism does not change the IEEE 802.11e protocol, but introduces a technique to reduce the contention overhead and therefore to use the channel more efficiently for real-time flows, especially when the traffic load approaches saturation conditions.

The paper is organized as follows. Sect. 2 outlines the 802.11e standard, while Sect.3 addresses related work. Sect. 4 describes the CWA mechanism here proposed, while Sect. 5 discusses performance obtained through an extensive set of simulations run in different scenarios under the ns-2 tool [6]. Finally, Sect. 6 gives our conclusions.

2. Overview of IEEE 802.11e

The EDCA mode of the IEEE 802.11e protocol, in order to manage the different Access Categories, implements in each node a dedicated transmit queue and an independent backoff entity for each AC. Each queue works as an independent DCF station and uses its own parameter set, which includes the Arbitration Inter-Frame Space (AIFS), the minimum Contention Window size (*CWmin*), the maximum Contention Window size (*CWmax*), and the Transmission Opportunity limit (TXOPlimit).

Similar to a 802.11 DCF node, each AC starts a backoff timer after sensing an idle channel for a duration equal to an AIFS length. However, while in DCF all nodes have the same opportunity to access the channel, in EDCA the AIFS depends on the AC. As a result, the duration of an idle medium before initiating a transmission is shorter for the higher priority ACs, which thus have higher probabilities of accessing the channel than the lower ACs.

The backoff value is selected as a random number in [0, CW], with CW set as *CWmin* at the beginning of a backoff procedure and increased up to CWmax whenever collisions occur, according to formula (2.1):

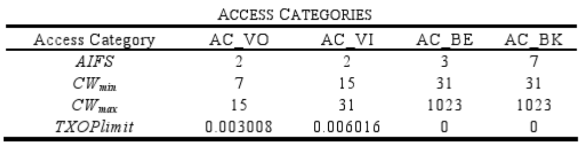
$$CW_{new}[AC]= ((CW_{current} [AC] + 1)*2) – 1 \quad (2.1)$$

In case of successful transmission, the CW value of the AC queue is reset to *CWmin*. As *CWmin* and *CWmax* determine the size of the CW, the smaller *Cwmin* and *CWmax* are, the greater the chances for a node gaining access to the medium are.

Finally, TXOPlimit is the time duration an EDCA function may transmit after winning access to the medium.

According to the IEEE standard [1], the above mentioned parameters are set as in Table 1. Note that the highest priority class, AC_VO has the narrowest [Cwmin, Cwmax] range.

Table 1: ACs and relevant parameters



ACCESS CATEGORIES

Access Category	AC_VO	AC_VI	AC_BE	AC_BK
AIFS	2	2	3	7
CW_{min}	7	15	31	31
CW_{max}	15	31	1023	1023
TXOPlimit	0.003008	0.006016	0	0

3. Related work

Among the works which recently addressed the impact on the performance of the IEEE 802.11e protocol of changing the various parameters of EDCA, Xiao [7], extending the Bianchi [8] model, implemented EDCA by means of 3-dimensional Markov chains and analyzed network behaviour for CWs of various sizes. Kong [9] also used 3-dimensional Markov chains to characterize the procedures of the various ACs with variations in both the CWs and the AIFS. Both papers have shown the effectiveness of changing CW depending on the network load.

Mechanisms for CW tuning are presented in [10] and [11]. The approach in [10], called AEDCF, does not implement a mechanism to vary the range of CWs, but calculates an ideal CW on the basis of the network load estimated according to the number of collisions experienced by the transmitted frames. Once the ideal CW is known, the current CW (*CWcurrent*) for the next frame is set by taking whichever is the lower between the minimum CW (*CWmin*) of the AC the frame belongs to and the ideal CW. The approach is shown outperforming EDCF, the IEEE 802.11e pre-standard distributed medium access mechanism. However, it uses the Persistence Factor, a parameter that was present in an earlier version of the IEEE 802.11e standard, but not in the final one.

The AEDCA approach proposed in [11] estimates network congestion by using the value of the current CW (i.e. that of the last frame sent). The distance between the current CW and *CWmin* is compared to the maximum distance between *CWmax* and *CWmin* for the relevant AC, deriving a parameter that is utilized to calculate the new CW for the next frame to be transmitted. The

AEDCA approach, like the AEDCF one, does not provide for changing the values of *CWmin* and *CWmax*, but simply chooses the best one in that range.

Instead of setting the CW to an optimal given value in the [CWmin, Cwmax] range defined by the standard, which proved to be inappropriate in many network load conditions, the CWA mechanism proposed in [5] adjusts the range of the current CW (i.e. the values of *CWmin* and *CWmax*) of the AC_VO class on the basis of information on the newtork workload collected during a time interval. Here we extend this approach, by varying the values of *CWmin* and *CWmax* in a cascaded way for each class. This allows to have, for each class, a CW adapted to the current network status, not limited by the bounds defined by the standard. In addition, here we evaluate the CWA performance in two different scenarios and under different workloads.

4. The Contention Window Adapter (CWA)

The CW_{min} and CW_{max} values for each AC in EDCA are static [1]. Under low workload conditions, small CW values for backoff are a convenient choice. We recall that the backoff value is selected as a random number in [0, CW], where CW is set as *CWmin* at the beginning of the backoff procedure and increases whenever collisions occur up to Cwmax, according to formula (2.1). However, when the network load increases, the probability of collisions increases too, thus enlarging the contention window size could be beneficial to reduce collisions. Unfortunately, EDCA does not implement any mechanism to dynamically change the contention window size of the different ACs according to the workload on the wireless network. The solution here proposed, the Contention Window Adapter, is a mechanism which tries to adapt the CW size of each AC to the network load.

In order to clearly explain the rationale behind CWA, let us consider the results shown in Table 2. They refer to tests run under ns-2 [6] with a 11 Mbps network (DSSS) made up of 20 stations, each generating AC_VO packets of 160 Bytes with a period of 20 ms, giving an overall workload of 1280Kbps. With such small packets and high transmission rate, the AC_VO class obtains poor performance when the setting defined by the IEEE 802.11e standard, i.e. CW_{min}[AC_VO]=7 and CW_{max}[AC_VO]=15 are used. In the different sets of simulations reported in Table 2 the CW_{min} and CW_{max} were statically set at the beginning of each experiment.

Results showed that RT performance of EDCA quickly and significantly degrade with growing workloads (in these conditions the AC_VO class is highly congested [12]) even in the presence of traffic from the highest priority class only (i.e. AC_VO). The reason is the high number of collisions, as the CWmin and CWmax settings provided by the standard provide too narrow a range of backoff values for the packets in the AC_VO class.

The results in Table 2 reveal that, although all the packets which are delivered arrive on time (i.e. they meet the 20 ms deadline assigned to them), the throughput is significantly higher for settings such as CW_{min}[AC_VO]=15, CW_{max}[AC_VO]=31 onwards than with the default settings CW_{min}[AC_VO]=7 and CW_{max}[AC_VO]=15 provided by the IEEE 802.11e protocol.

CWmin	CWmax	Aver.delay (ms)	Throughput (%)
7	15	17	62
15	31	16	89
31	63	7	99

Table 2: Effects of varying CWmin, CWmax on AC_VO performance

In addition, Table 2 shows that, with wider contention windows, the delay experienced by RT packets is reduced. This is due to the smaller number of collisions experienced by RT packets thanks to the broader range of backoff values.

From the various tests run it also emerged that a good way to reduce the collision probability is to vary CW_{min} and CW_{max} by doubling both of them. These results were also confirmed by other tests run with a greater number of stations and different workload conditions.

Similar considerations can be made as far as the other ACs are concerned. In this paper we focus on industrial environments more general than the one addressed in [5], where the only traffic is RT, periodic and mapped into the AC_VO class. Here instead we address a scenario where a number of Workstations (WSs) transmitting different types of traffic (Voice, Video and Background) share the same channel with RT nodes transmitting small size periodic process control frames with tight deadlines. The aim of CWA here is to improve the performance of RT traffic. The parameter used in the CWA to assess the network load is the ratio between the number of collisions affecting the highest priority packets (*coll*(AC_VO)) and the total number of packets sent (*pkts_sent*(AC_VO)) in that AC during a given observation interval *Δt*:

$$ratio = \frac{coll(AC\_VO)}{pkts\_sent(AC\_VO)} \qquad (4.1)$$

This parameter gives the average number of collisions per packet, and is an index of the level of congestion on the network, with particular reference to the AC_VO class. The number of collisions can be easily obtained. Here we run ns-2 simulations, but also real measurements using network boards equipped with open source drivers are possible. In order to minimize the bias against transient collisions, an Exponentially Weighted Moving Average (EWMA) estimator was used. In a generic *i* interval, the value of r^i_{avg}, to be used by the algorithm, is updated in the following way:

$$r^i_{avg} = (\lambda - 1)\cdot r^i_c + \lambda \cdot r^{i-1}_{avg} \qquad (4.2)$$

The CWA, when adapting the *CWmin* and *CWmax* of the various ACs, takes this parameter into account, in a two-step procedure. First, the CWmin and CWmax of the highest priority class, i.e., AC_VO, are set. Second, to enforce the different QoS support offered to each AC according to the 802.11e standard, suitable changes to the CWmin and CWmax values for the other ACs are made. Let us examine the first step.

- If ratio is below a given minimum threshold α, then the network is underloaded, so it is safe reducing (i.e. halving) CWmin and CWmax for AC_VO to speed up the backoff procedure;
- If ratio is in between two values α and β, which are heuristically set depending on the requirements of the supported RT application, then the current CW_{min} and CW_{max} values for AC_VO are maintained;
- If ratio is higher than β, but still below a maximum threshold γ, then the network load is high and thus, to reduce the probability of collisions, CWA increases CW_{min} and CW_{max} for AC_VO doubling their current values;
- If ratio exceed the maximum threshold, then the network is heavily congested, so in order to drastically react, CWA doubles 2 times both CW_{min} and CW_{max} for AC_VO. This is to male the system more reactive to sudden traffic peaks.

In the second step, CWA arranges the CWmin and CWmax values for the other ACs in order to maintain the differentiation between them, and thus between the various types of traffic. CWA, therefore, whenever the CWmin and CWmax values for AC_VO increase, enforces a cascaded increase in the CWmin and CWmax values for the other ACs, i.e. AC_VI, AC_BE and AC_BK.

Here the upper bound for CW_{max} in the AC_VO class has been set to 63, because higher values, although reducing the probability of collisions, would excessively penalize the performance of this class, introducing too long backoff times. Fig.1 shows the above described procedure, written in pseudo-code.

```
procedure Contention Window Adapter
    if ($n\_pkt\_sent(AC\_VO)$ != 0) then
        $r_{avg}$ := $EWMA(n\_coll[AC\_VO]/n\_pkt\_sent[AC\_VO])$
        if ($r_{avg} \leq \alpha$) then
            call procedure Decrease()
        else if (($\alpha < r_{avg}$) && ($r_{avg} \leq \beta$)) then do nothing
        else if (($\beta < r_{avg}$) && ($r_{avg} \leq \gamma$)) then
            call procedure Increase(1)
        else if ($r_{avg} > \gamma$) then
            call procedure Increase(2)
        end if
    end if
end procedure
```

Fig. 1 The CWA algorithm

The Increase procedure is used to increment CW_{min} and CW_{max} for the various ACs. It requires as parameter an integer value (equal to 1 or 2), which indicates the amount of increment to be done on the CW_{min} and CW_{max} values of all the ACs, according to the ratio value. Here we recall that here an increment means doubling the current values. Table 3 shows the possible combinations of CWmin and CWmax for the various ACs which may occur.

The Decrease procedure is run whenever ratio is lower than the α threshold. Here a decrement means that the current CW_{min} and CW_{max} values are halved. For example, in Table 3, if the current combination at time t is the one displayed on the third row, and at time *t+ Δt* ratio is less than α, the Decrease procedure will be invoked by CWA and the CW_{min} and CW_{max} values of the various delle ACs will be set as in the second row.

	CW min (VO)	*CW max (VO)*	*CW min (VI)*	*CW max (VI)*	*CW min (BE)*	*CW max (BE)*	*CW min (BK)*	*CW max (BK)*
1	7	15	15	31	31	1023	31	1023
2	15	31	31	63	63	1023	63	1023
3	31	63	63	127	127	1023	127	1023
4	31	63	127	255	255	1023	255	1023
5	31	63	255	511	511	1023	511	1023

Table 3: CW_{min} and CW_{max} combinations for CWA

There is a potential problem when using CWA in the presence of stations that do not have traffic in the AC_VO class. If a station tries to transmit only traffic with a priority other than the highest (AC_VO), the CWA cannot react on the basis of the collisions affecting that type of traffic, as ratio is defined for the AC_VO class.
To overcome this problem, the CWA shall consider the AC_VI ratio, expressed as (4.3)

$$ratio = \frac{coll(AC\_VI)}{pkts\_sent(AC\_VI)} \quad (4.3)$$

The CWA behaviour will be the same as before, with the only difference that, to be sure that CW_{max} for AC_VO in those stations which do transmit RT traffic in that class is no greater than CW_{min} for AC_VI in the stations which do not transmit traffic in the AC_VO class, the minimum possible value for CW_{min} for AC_VI will be set to 15 or 63, according to the situation.

To obtain the right value, the station will use a parameter, here called RT_NAV, which is obtained by summing the Duration Field of the AC_VO packets sent during a given observation interval (her chosen equal to 300 ms). The AC_VO packets are identified by the Flow Identification field. All the stations are able to read the header of any packets in transit on the network as, according to the 802.11e standard, the Duration Field is used to calculate the Network Allocation Vector (NAV). The NAV and its associated timer are used to regulate the access to the medium for the various stations avoiding that a transmission trial would interfere with an on-going transmission. The CWA can therefore exploit these features of the standard to enable a station, which does not have traffic in the AC_VO class to transmit, to assess whether there are RT stations sending AC_VO traffic on the shared channel. If, after an observation interval, the station has a non-null RT_NAV value, CWA sets the CWmin value for the AC_VI traffic in that station to 63, otherwise the value 15 will be chosen.

In the following Section, CWA performance obtained in different scenarios will be presented and discussed. In all the addressed scenarios, CWA is run on Workstations nodes (WSs), i.e., stations transmitting interference traffic (Voice, Video, Background), while RT stations, which transmit only periodic small-sized process control frames in the AC_VO class, will use the static CW_{min}(AC_VO) and CW_{max}(AC_VO) values defined by the standard [1], i.e. 7 and 15.

5. Performance Evaluation

Here two different scenarios have been considered to compare the performance of the EDCA with and without CWA. Simulations were run using the Network Simulator version 2.28, with the patch [13][14]. The thresholds used in the CWA procedure were $\alpha = 0.2$, $\beta = 0.6$, $\gamma = 2$, while in (4.2) $\lambda = 0.8$. As said before, two different station types are present: RT and WSs. All the RT stations transmit traffic to the same Base Station, while WS exchange data between them and with an Access Point. RT traffic is periodic, of CBR type, with a period=20ms, bit rate=18kbps and packet size= 45 byte. In industrial environments, RT stations could be sensors which transmit field variables to a PLC, while WS are generic stations which use the same wireless channel to transmit consumer traffic (Video, Audio, http, ftp etc.).
The performance parameters measured are throughput and delay for all the types of traffic, and the number of collisions for RT traffic. The Physical and MAC layer parameters in ns-2 were set as in Table 4:

PARAMETER	***VALUE***
MAC	802.11e
Physical Layer	802.11b
SIFS	10us
SlotTime	*20us*
PreambleLength	144 bits
PLCPHeaderLength	48 bits
PLCPDataRate	1Mbps
DataRate	11Mbps
BasicRate	1Mbps
ShortRetryLimit	7
LongRetryLimit	4
cfb	Disabled
ifqLen	50
Routing	DSDV

Table 4: Simulation parameters for scenario 1

5.1 Scenario 1

In this scenario, on the same wireless network there are RT stations sending in the AC_VO class periodic traffic with tight deadlines to the BS, and WSs exchanging large packets generated with high data rates in the AC_VO class too. The WSs are inside the same geographic area of the RT stations.

Simulations were run with a growing number of WSs, in the range [2, 10]. Each WS transmits packets of 1000 Byte at a 1Mbps. This kind of traffic will be henceforward called "greedy". The aim of this scenario is to highlight the RT performance in the presence of such WSs sending greedy interference traffic with the same priority of RT ones, i.e., in the AC_VO class. As said before, here CWA is run on the WSs, while the RT stations use the static setting foreseen in the EDCA protocol. Fig. 2 compares the throughput obtained with and without CWA. The CWA parameters The comparison reveals that there is a significant difference in throughput with and without CWA. For example, with 10 WSs, the throughput ratio is about 1/3, i.e., while standard EDCA is able to transmit about one frame out of three, CWA succeeds in transmitting almost all the RT frames. Even the throughput of greedy traffic improve, thanks to the CWA mechanism, which reduces the number of collision within the AC_VO class.

Fig. 3 depicts traffic delay. As shown in the figure, when CWA is run, the RT delay is quite lower than the one experienced when standard EDCA is used.

Finally, Fig.4 shows the number of collisions experienced by a RT station with and without CWA. The results prove that adapting the contention window size according to the network workload significantly reduces the number of collisions for RT traffic. This means that both the timing performance of RT traffic and the overall bandwidth exploitation improve.

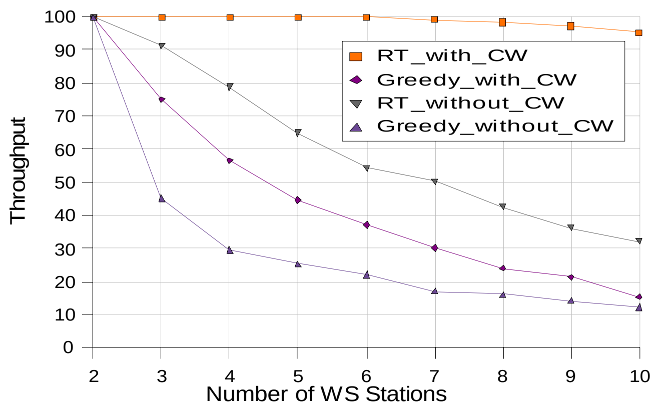


Fig.2: Throughput comparison (percentage)

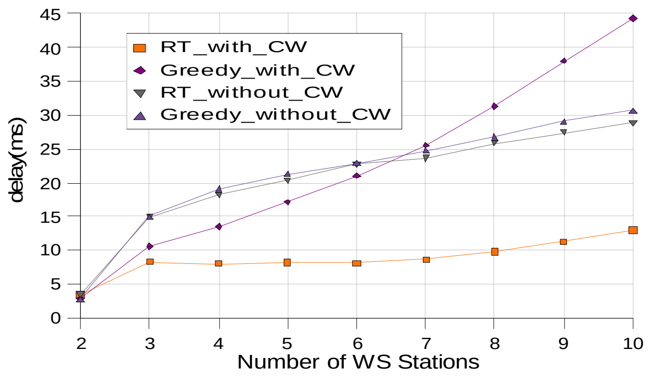


Fig.3: Delay comparison

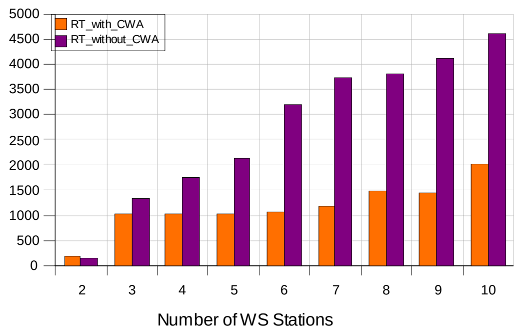


Fig.4: Number of collisions for RT traffic

5.2 Scenario 2

In this scenario, an Open Communication Environment (OCE) similar to the one in [4] is considered. Here, RT stations share the same channel with 5 WSs generating multimedia (Voice and Video) and Background traffic (BK). The aim of this scenario is to evaluate the RT performance in the presence of these interefering WSs. The RT traffic is the same used in scenario 1. For the set of WSs, the offered load, here indicated as GST, ranges from 10% to 100% of the 802.11b PHY data rate (11 Mbps). Each WS generates λ packets per second for the different types of traffic, with the same rate, but different packet size. In order to impose the requested GST overall network load, λ is obtained as in formula (5.1)

$$\lambda = \frac{G_{st}}{(\mathrm{PK}_{\mathrm{VO}} + \mathrm{PK}_{\mathrm{VI}} + \mathrm{PK}_{\mathrm{BK}})} \quad (frame/s)$$

where $\mathrm{PK}_{\mathrm{VO}}$, $\mathrm{PK}_{\mathrm{VI}}$ and $\mathrm{PK}_{\mathrm{BK}}$ represent the packet size (bits) transmitted in each AC by the WSs (Table 5).

Parameter	***RT Stations***	***Workstations***		
Traffic	RT	VO	VI	BK
CW	CWmin[VO]=7 CWmax[VO]=15	CWA	CWA	CWA
AIFSN	2	2	3	7
Packet size (byte)	45	160	1280	1600

Table 5: Traffic parameters in Scenario 2

Figs. 5-6 show the performance of RT stations, while Figs. 7 depicts the throughput obtained for WSs. For RT traffic, the benefits of CWA, in terms of throughput (fig.5) and delay (Fig.6), are evident.

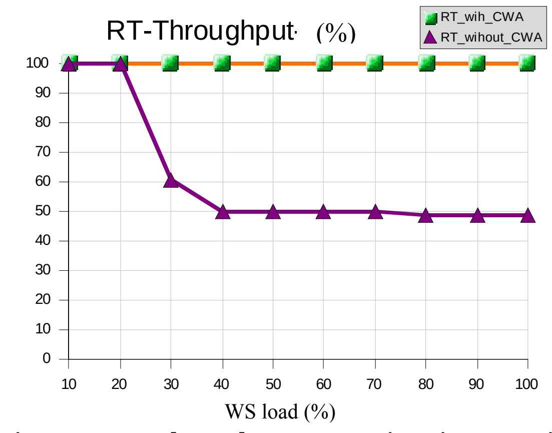


Figure 5: RT Throughput comparison in Scenario 2

There is an improvement even for traffic generated by WSs, both in throughput (Fig.7) and delay values (not shown for reasons of space).

6. Conclusions

CWA proved to be successful in reducing the number of collisions while maintaining traffic differentiation between the different ACs in both scenarios investigated in this paper. Further work will deal with implementation of CWA on COTS network boards.

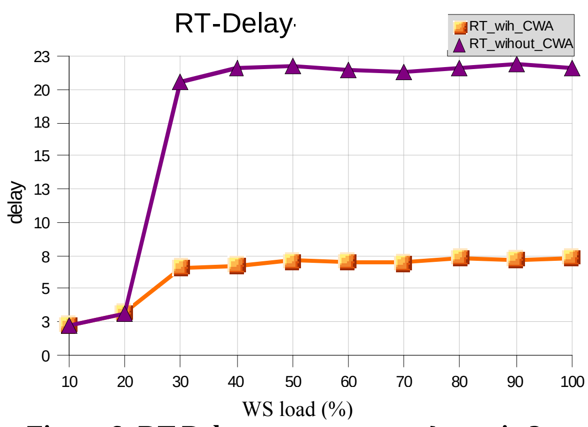


Figure 6: RT Delay comparison in Scenario 2

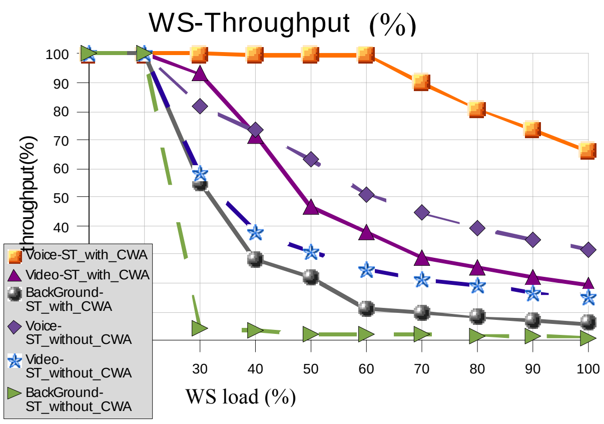


Figure 7: WS throughput comparison in Scenario 2

References

[1] "IEEE 802.11e-2005, Medium Access Control (MAC) Quality of Service Enhancements," 2005.
[2] "IEEE 802.11, Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications," 1999.
[3] "IEEE 802.1D-2004, MAC Bridges," 2004.
[4]R.Moraes, P. Portugal, F. Vasques, "Simulation Analysis of the IEEE 802.11e EDCA Protocol for an Industrially-Relevant Real-Time Communication Scenario", IEEE ETFA'06, Prague, Czech Republic, Sept. 20-22, 2006.
[5] S. Vittorio, G. Kaczynski, L. Lo Bello, "Improving the real-time capabilities of IEEE 802.11e through a Contention Window Adapter", RTAS'07_WIP, pp.64-67, Bellevue, USA, Apr.2007.
[6] The Network Simulator. http://www.isi.edu/nsnam/ns
[7]Y.Xiao, "Performance analysis of IEEE 802.11e EDCF under saturation condition," *Proc. of IEEE ICC*, 2004, pp.170-174.
[8] G. Bianchi, "Performance analysis of the IEEE 802.11 distributed coordination function," *JSAC*, 18, 2000, pp.535–547.
[9] Z.Kong, D.Tsang, B. Bensaou, D. Gao, "Performance Analysis of IEEE 802.11e Contention-Based Channel Access," *JSAC*, 22, 2004.
[10] L. Romdhani, Q. Ni, T. Turletti, "Adaptive EDCF: Enhanced Service Differentiation for IEEE 802.11 Wireless Ad-Hoc Networks," *Wireless Commun. and Mobile Comp.*, 2004.
[11] J. Sawaya, B. Ghaddar, S. Khawam, H. Safa, H. Artail, and Z. Dawy, "Adaptive Approach for QoS Support in IEEE 802.11e Wireless LAN," *Wireless Mobile*, 2005.
[12]J. Jun, P. Peddabachagari, M. Sichitiu, "Theoretical Maximum Throughput of IEEE 802.11 and its Applications," *IEEE NCA03*, 2003, pp. 249–256.
[13] S. Wiethoelther, M. Emmelmann, C. Hoene, "TKN EDCA model for NS-2," *TR TKN-06-003*. Telecom. Network Group, Tech. Univ. Berlin, 2003.
[14]TKN-802.11e.http://www.tkn.tu-berlin.de/research/802.11e_ns2

An Energy-efficient Real-Time Communication Framework for Wireless Sensor Networks

Emanuele Toscano, Orazio Mirabella, Lucia Lo Bello
RETISNET Lab
Department of Computer Engineering and Telecommunications
University of Catania
Catania, ITALY
{omirabel, lucia.lobello}@diit.unict.it

Abstract— **This paper addresses the architecture, protocol stack and routing algorithm of a framework, called RTPAW (Real-Time Power-Aware) devised to support energy-efficient real-time communication over Wireless Sensor Networks (WSNs) used in monitoring applications. The aim of RTPAW is to provide soft real-time traffic with an appropriate QoS while reducing the energy consumption of the nodes, which have to work for long periods without the possibility of replacing their batteries. The proposed framework exploits the features of an Aggregation level introduced between the MAC and Routing layers. This layer mainly deals with reducing the amount of energy dissipated, while the Routing layer is entrusted with achieving the desired QoS, in terms of delivery speed, to support the transmission of soft real-time traffic. The paper presents the RTPAW performance and discusses the way there are affected by changes in the operating parameters and network load.**

I. Introduction

A Wireless Sensor Network (WSN) for monitoring applications typically consists of nodes which process their data and exchanging it amongst themselves as well as with a base station via a Sink node. As WSN nodes are generally located in the proximity of or inside the phenomenon they are monitoring, and the environments involved are often remote or hostile to humans, they should be able to function without human intervention for as long as possible. In order to meet the long-lasting autonomy requirement, low-power consumption is the main issue to be tackled. For this reason, and to allow for deployment of WSNs at affordable production costs, low-power processors and very small memories are typically used. This, however, is not sufficient, as the amount of energy consumed by communications in WSNs is usually much greater than that used for processing. There is therefore a major need for protocols able to optimize power consumption, so as to prolong the lifetime of the nodes and thus that of the network as a whole. However, the requirement on power consumption clashes with the need for real-time support, which comes out as WSNs used for monitoring applications mostly feature periodic soft real-time traffic and thus require a way to enforce a minimum data delivery speed so as to meet delay constraints.

The communication protocols for WSNs existent in the literature aim either at minimizing power consumption (e.g., [1], [2] and [3]) or at providing soft real-time traffic with the desired QoS (e.g., [5] and [6]). This paper describes the Real-Time Power Aware Framework (RTPAW), which targets a trade-off between power consumption and delivery speed by exploiting the features of both categories of protocols. An earlier version of RTPAW was sketched in [10]. Here we give a more detailed description. In addition, a performance evaluation of RTPAW performance, obtained by ns-2 simulations, is presented. The sensitivity of RTPAW performance to changes in the operating parameters and network load is also discussed.

II. Related work and Motivation

A. Related work

In order to minimize power consumption, cluster-based routing protocols, such as LEACH [1] and MECH [3], adopt a hierarchical routing strategy. A limited number of always active nodes, called cluster heads, form a backbone, while the other nodes can remain asleep and only wake up when data is being sent. The cluster heads are elected in rotation and remain cluster heads for a certain period of time, called a round. Intra-cluster communication uses Time Division Multiple Access (TDMA). A super-frame is created, in which each node has its own time slot. Once data is acquired, the cluster heads transmit it directly to the base station. Code Division Multiple Access (CDMA) is used in order to reduce the impact of radio interference between different clusters. This approach enables energy saving, but suffers from scalability problems which make it unsuitable for large networks, as it requires clock synchronization at a network level, which is only possible for small networks. Moreover, in LEACH cluster heads communicate directly with the base station. On the other hand MECH supports hierarchical message forwarding, but does not guarantee any QoS.

Another class of routing algorithms has been developed with the aim of providing WSNs with a given QoS. Among them, SPEED [5] and MMSPEED [6]. Based on geographical routing, which is particularly efficient in networks covering a large geographical area, both approaches try to guarantee a minimum speed in data delivery. However, these algorithms were developed on 802.11 and do not target power consumption.

Conversely, the RPAR [4] protocol targets real-time applications and at the same time tries to optimize power consumption, constantly regulating the transmission power. This approach is, however, affected by anomalous behavior in heavy traffic conditions, which tends to favor network congestion. The reason for this behaviour is that, when a node is congested, due to high contention, it has to undergo a large number of retries before transmitting a packet correctly, due to high collision probability. Hence, RPAR increases the transmission power, worsening the situation. In addition to this problem, it has to be highlighted that energy saving is limited, as nodes never go to sleep.

B. Motivation

Our proposal derives from the need to find a communication technique for WSNs that is efficient as regards power consumption and able to support soft real-time traffic. Another highly desirable characteristic is the ability to use, where possible, standard protocols or established protocols that have been widely studied (e.g. in [9]). For this reason in this work we chose to use the 802.15.4 standard [7],[8] for the MAC layer, whereas for the routing layer we envisaged an adapted version of SPEED.

III. The RTPAW Framework

A. Network architecture proposed

As geographical routing is not based on the physical address of a node, but on its position, if there are several nodes geographically very close to each other, not all of them have to be active at the same time. This allows for energy saving. To achieve an alternation between activity and sleep periods, a proper network architecture has been devised.

The RTPAW architecture inherits the main features of cluster-based protocols, but introduces a set of new concepts. The nodes are grouped into clusters, which we call *Aggregated Units* (AUs). The AU structure is different from that of the clusters in the protocols currently proposed in the literature. Here, the nodes in an AU belong to three different categories:

• *Cluster Head* (CH);

• *Relay Node* (RN);

• *Cluster Node* (CN).

In each AU there is one CH, one RN and a varying number of CNs, as shown in Fig. 1. The CH has the task of collecting data from the sensor nodes belonging to the cluster (the CNs) and periodically transmitting it to the RN. The task of the latter

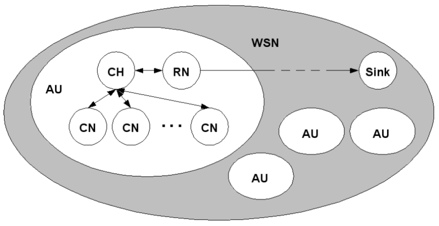


Fig. 1. RTPAW Network architecture

is to forward the data to other RNs or the Sink node. In this architecture, therefore, the CH handles transmission within the cluster, while the RN handles transmission outside the cluster.

There are three different types of traffic: communications between the CH and the CNs, the ones between the CH and the RN, and the ones between RN and RN or RN and Sink. The first and second types of traffic are periodic and, as we will explain later, mainly aim at the functioning of the AU (and are managed by the Aggregation Layer). The third type of traffic is not periodic, and is handled at a higher layer, as it is relevant to the single AUs (and is managed by the Routing Layer).

Splitting the RN and CH roles implies several benefits. Firstly, the RN is able to perform full time packet forwarding, thus we have better routing performance: if RN and CH roles were unified in the CH, packet forwarding could be performed only when there is no data from CNs. This would require network-wide clock synchronization and reduce the bandwidth utilization (CH would be a bottleneck). The parallelism between RN and CH operations achievable splitting the roles provides a better bandwidth exploitation, and reduces latencies and chances of congestion. Furthermore, having RN and CH roles, in conjunction with the use of different radio channels for nearby AUs, allows for isolation between contention-free intra-cluster communications and contention-based inter-cluster communications, to the benefit of both performance and network scalability.

B. Protocol architecture of the RTPAW Framework

The RTPAW protocol architecture proposed here features an Aggregation Layer which acts as a mediator between the MAC and Routing layers for the combined handling of energy awareness and real-time support. The Aggregation Layer deals with creating and managing the AU and transmitting the first two types of traffic described before. The Routing Layer lies above the Aggregation Layer and forwards packets between AUs, thus handling the third type of traffic. In this architecture, the MAC layer closely collaborates with the Aggregation layer to provide the Routing layer with a uniform view of the set of sensor nodes making up the AU. The basic addressable entity in the Routing layer is therefore not the single WSN node, but the single AU.

The Aggregation layer is split into two sub-layers, with the lower part (called MAC-dependent) which strongly depends on the MAC protocol used and represents an extension of the basic functions needed to implement the level above. This sub-layer has to provide primitives in order to set the radio channel, put nodes to sleep and wake them up, query the battery charge status, perform channel scans (i.e. Energy Detection scans), send and receive frames. Using this set of primitives it is possible to create the MAC- independent sub-layer. The upper layers primitives depend on the protocol used; however, a set of basic primitives should be provided for every protocol. For example, the MAC-independent part of the Aggregation Layer should always provide primitives to create the AU, set up the AU (i.e. beacon period), manage the AU (i.e. CH or RN election), send and receive data (i.e. CN to

CH or CH to RN). While the Routing layer should always provide primitives to send and receive data (i.e. RN to RN), and to send control packets whenever needed.

The main task of the Aggregation Layer is to create and handle the cluster and the aim is to reduce consumption by scheduling periods of activity and sleep periods. The Aggregation Layer may also perform some data processing if it is not single CN data, but some aggregated quantity obtained from multiple CN samples, that has to be forwarded in the WSN. As mentioned previously, the MAC level and the MAC-dependent part of the Aggregation Layer work closely, as the activity periods may coincide with certain states of the MAC Layer. For example, if TDMA is used for transmission inside a cluster, it is possible to make nodes go to sleep during time slots other than their own. Above the Aggregation Layer virtually any routing algorithm providing a certain QoS can be used. The algorithm will operate viewing the whole AU as a single node.

The expected advantages of the proposed architecture are:

- Reduced power consumption, depending on the efficiency of the Aggregation protocol used;
- Advanced QoS management, depending on the efficiency of the Routing protocol used.

Moreover, depending on the aggregation protocol used, as the routing unit is the whole AU, rather than the single node, the AU will continue to live even if several of its nodes cease to function. In addition, the distance between two aggregated units is much greater than that between the single nodes in the network. Therefore, if a geographical routing algorithm is used, the system is less sensitive to the inaccuracy of location mechanisms.

IV. The Protocols Used

A. *Physical and MAC Layer*

At PHY and MAC layers the 802.15.4 standard [7],[8] has been adopted; the non-beacon enabled mode has been chosen to guarantee greater scalability and fault tolerance. In order to avoid inter-AU interference, we create a cell-based architecture using the 16 different radio channels on 2.4GHz. In this manner it is possible to make the radio cells at the Physical layer coincide with the AUs at the Aggregation Layer. Selection of the transmission channel can be automatic during initialization of the nodes, using the Energy Detection scan (ED scan) procedure defined in [7] and [8], or set according to the position of a node. In the latter case, we can create the cellular radio architecture by manually setting transmission channels with the aim of minimizing the interferences among nodes on different AUs.

B. *Aggregation Layer*

The Aggregation Layer handles data transmission in a single AU. In every AU a super-frame structure is created, and each CN belonging to the AU sends data to its own CH, during the assigned timeslot. It should be noted that the Aggregation Layer super-frame, which is shown in Fig. 2, is not mapped on the 802.15.4 super-frame, but is created at a higher level using the 802.15.4 non-beacon enabled mode. An important difference between RTPAW and the other cluster-based protocols existent in the literature is that, whereas the latter ones usually require network-wide clock synchronization, our protocol requires synchronization at the AU level only.

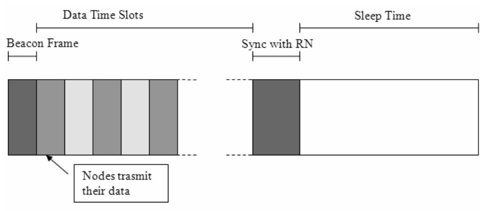


Fig. 2. Aggregation Layer super-frame.

As mentioned previously, in our approach there are not only cluster heads (CHs) and nodes belonging to a cluster (CNs), but also relay nodes (RNs). A CH and an RN are elected in each cluster. The former collects data from the other nodes (except the relay node), whereas the latter forwards packets from one cluster to another. It is necessary to provide a small period of time in which the CH and RN nodes synchronize their data. The RNs must always be active, while the CHs can go to sleep only after synchronization with the RN. All the others can go to sleep and only wake up to receive synchronization signals from the CH or to transmit their data during the assigned time slot. As CHs and especially RNs consume more power than the other nodes, they have to be elected in rotation, in such a way as to balance the power consumption.

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

1) Initialization: The main aim of the initialization phase is definition of the cellular architecture. We assume that all the nodes know their own position and that they have been randomly arranged with a relatively uniform density. It is therefore possible to create a homogeneous cellular structure, as a grid subdividing the area being monitored into a number of small uniform regions, each hosting a cell. The next step is the first election, during which any of the nodes equipped with the greatest amount of energy can be elected as the CH of his AU. Then the CH elects the RN (as described below) and sends the transmission schedule to the CNs.

2) Election: In cluster-based protocols integrating a cluster head rotation mechanism, whenever a CH is elected it is necessary to reconstruct the whole cluster. In the presence of tight deadlines, or when constant updating of the variables being monitored is needed, this may degrade the QoS. It was therefore decided to separate the distributed algorithm for the first election from the one used later on, which is centralized. In the latter case, at a certain point (after a pre-established time or because its remaining power has dropped beneath a given threshold) the CH autonomously decides which node is to be its successor and notifies the node involved. From the next transmission cycle onwards, the new CH will start operating. The decision regarding the next CH is based on the residual energy of the nodes in the cluster, signalled in the frame that nodes send during normal transmission phases.

Election of the RN is different. The CH elects the RN autonomously when it is requested. An RN whose power has dropped beneath a certain threshold notifies the CH during their synchronization phase. The CH consequently chooses as the next RN whichever of the nodes with the greatest amount of energy has the strongest signal. The former information can be directly devised by the hardware, while the latter can be obtained with a negligible overhead, inserting it in the packets that CN nodes send to the relevant CH.

3) Data transfer: Intra-AU data transfer follows a pre-established synchronized sequence which emulates a super-frame structure in the Aggregation Layer. In this way, it is possible to avoid collisions. The super-frame starts with a beacon frame from the cluster head, used for transmission synchronization in the AU. After the beacon, there are time slots during which the CNs can transmit their data to the CH, using TDMA. During all the time slots assigned to the other nodes, a CN can go to sleep. It must, however, wake up again on time to receive the next beacon frame. The last section of the super-frame is for synchronization between the CH and the RN. In the meanwhile, the RNs form a backbone of nodes that are always active in forwarding packets to the Sink node. They communicate over a single dedicated channel, so during the synchronization phase it is necessary to switch channels temporarily. When the RN acquires the updated CN data from the CH, it forwards it as defined by the Routing Layer. Only RNs can forward data, so they are the only nodes that run the routing algorithm.

C. Routing Layer

As the Routing layer is located above the Aggregation layer, packets are not addressed to single nodes, but to single AUs. So, the only task of the routing algorithm is to forward packets from a source AU to their final destination, usually the Sink node. The scenario RTPAW was devised for is one in which the WSN comprises a large number of nodes and may cover a wide area. For this reason, although the underlying Aggregation layer contributes towards increasing the scalability of the network, the algorithm used for routing between the AUs has to be able to handle a large network without any difficulty. In addition, it is advisable to use a routing algorithm that is as fault-tolerant as possible. As said before, the presence of an underlying Aggregation layer mitigates the system-wide impact of faults occurring in single nodes. Finally, the routing algorithm has to make it possible to achieve the desired QoS, which in our case is delivery speed. A routing algorithm which possesses all these features is SPEED [5]. For this reason a SPEED-inspired approach is used in RTPAW. There are a few differences between the RTPAW adapted version of SPEED and the one described in [5], which are given below.

In RTPAW, the forwarding of packets does not involve singol nodes, but whole AUs (through RNs); therefore the address used here to route data packets is not constituted by the real geographical coordinates of the current RNs, but on the *virtual coordinates* of the whole AU, which are an approximation of the AU centroid coordinates. Another difference is that hop-to-hop transmissions require Acks, and the per-hop delay is calculated according to the formula

$$delay = W_q + (T_{ack} - T_s)/2 \qquad (4.1)$$

where W_q is the time elapsed waiting in a queue, T_s is the arrival time of a packet and T_{ack} is the time when the Ack is received.

Finally, as the RNs periodically change, we need some means to keep the network in the steady state even after the election of new RNs. When a new RN is elected, the old one sends the new RN its neighbouring table. As soon as an RN becomes active, it immediately sends a broadcast beacon, so that its neighbours can update their neighbouring tables with the MAC address of the new RN. A second beacon is sent after a short time, in order to minimize the chance that any neighbours will fail to update their table. Then the node can start to send periodic beacons normally, as described in [5].

V. Simulations and Evaluation

In order to evaluate the effectiveness and performance of the RTPAW framework, we simulated the network architecture and the protocol stack of the framework with the *ns-2* [11] tool. For the physical parameters of the simulated nodes, the datasheet of COTS devices, i.e., the MaxStream XBee modules [12], were taken into account. We performed several simulations, with different network loads and number of nodes. In the following subsections, the results obtained in terms of energy consumption, e2e delay and delivery speed are discussed.

A. Energy efficiency of RTPAW

The energy efficiency evaluation was initially performed upon a small-sized network, to avoid excessively long simulation run-times. In this evaluation it is important to have a long simulated time, as our aim is to estimate the average node consumption in the long term. For this reason, we considered a scenario made up of 135 sensor nodes grouped in 9 AUs, each with 15 nodes. The monitoring area is set to 900 m^2 (a square with 30 *m* sides), while the area covered by a single AU is 100 m^2 (a square with 10 *m* sides). Each sensor node sends its data every 10 seconds towards the Sink node. The payload of a CN data packet is only a 4bytes integer, but considering the overhead due to the 802.15.4 and RTPAW headers, the frame length for a CN data packet is 15bytes. The *setpoint speed* [5] (that is, the minimum forwarding speed) is set to 1 *km/s*. Twelve hours of network functioning were simulated.

The efficiency index adopted here is the mean power consumption of a node inside an AU. We evaluated this

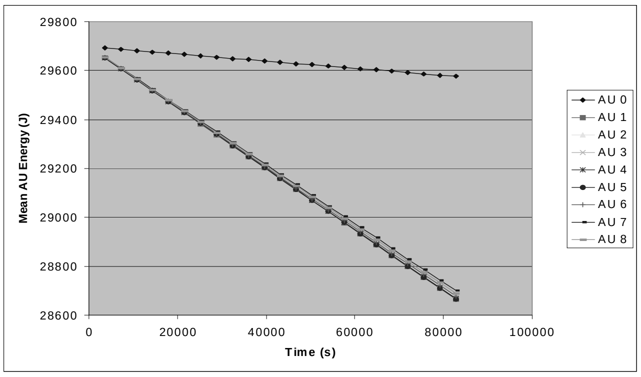


Fig. 3. Mean AU residual energy.

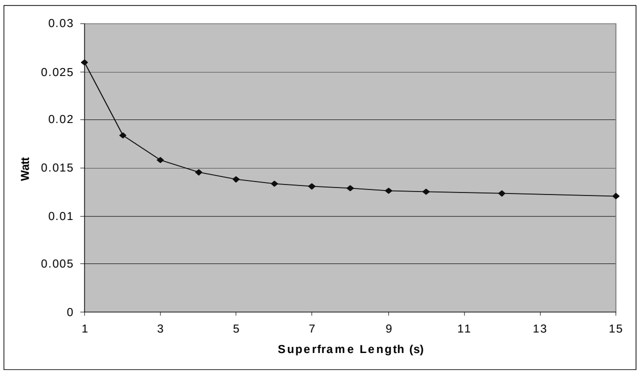


Fig. 5. Mean AU power consumption vs. varying super-frame length.

parameter measuring the residual energy power in each sensor node at regular time intervals (every hour) and then calculating the arithmetical mean of the residual energy obtained from each node belonging to the same AU.

The mean residual energy of each AU as a function of time is shown in Fig. 3. Looking at the figure, we notice that the mean consumption for AU_0 is the lowest. This is because in AU_0 nodes communicate directly with the Sink node. Here the Sink node replace the RN, and no other RN is needed in this AU. The energy consumption of the Sink node is not taken into account in the figure, because we assumed that the Sink node is directly connected to a power source. Fig. 3 highlights two important features of the RTPAW framework. The first is the energy consumption balance among different AUs, which all, except for AU_0, maintain very close energy values along the time axis. The second is the linearity of the AU mean residual energy curves.

Both features are obtained thanks to the Aggregation Layer, which schedules transmission and sleep times in a constant and fair way among the CNs. The only not-constant (and not-linear) part of the AU energy consumption is due to the RN. However, as the RN never goes to the sleep state and for sensor nodes the difference in the energy consumption of transmit and receive states is small, this part could also be approximated to a constant value.

Thanks to the linearity of the residual energy curves, it is possible to estimate the AU mean energy consumption per time unit. As a result, energy consumption over an arbitrary time interval or an approximation of the overall network duration can be calculated (but, in this case, the battery capacity as well as the mean energy consumption have to be considered). The computation of the mean AU energy consumption is equivalent to the computation of the angular coefficient of the line representing the mean AU energy. Making use of the linearity of the AU mean energy consumption, we can approximately estimate the power consumption in very large networks, by simulating only a few minutes (e.g., some dozen) of network functioning. Fig. 4 shows a graph obtained from a simulation of 25 minutes of network functioning. In this scenario we have 1500 nodes grouped into 100 different AUs. The monitoring area is 10000 m^2 (a square with 100 *m* sides), while the area covered by a single AU remains set to 100 m^2. The setpoint speed also remains set to 1 *km/s* too. Fig. 4 shows that the mean consumption of a node in any AU (other than AU_0) in this simulation is just above 12 *mW*. Without RTPAW, assuming that nodes are in the receive state for the 90% of their time and the remaining 10% are transmitting data, the mean consumption is about 163 *mW*. Thus, by lowering the duty cycle of the nodes through the Aggregation Layer, RTPAW reduces the power consumption by an order of magnitude.

What mostly affects power consumption is the length of the super-frame. In fact, with a longer super-frame, nodes can stay asleep for a longer time. As the super-frame becomes smaller, the CH and CN duty cycles increases, so we necessarily have an increase in power consumption, as shown in Fig. 5. The energy consumption obtained in our simulations is, however, much lower than the estimated 163 *mW* without node duty

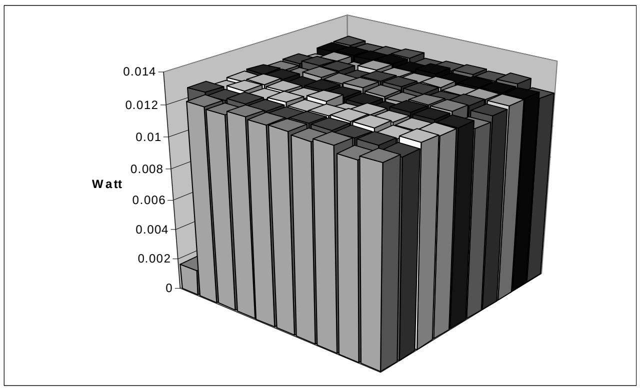


Fig. 4. Mean AU power consumption per time unit (1500 nodes grouped in 100 Aus; each node sends data every 10 *s*).

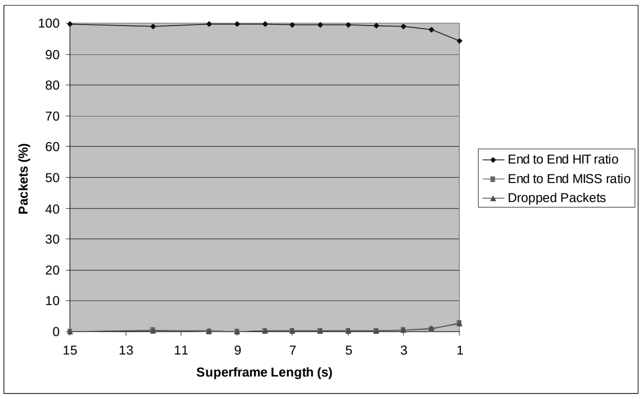


Fig. 6. End-to-end deadline hit ratio, miss ratio and dropped packets vs. varying super-frame length.

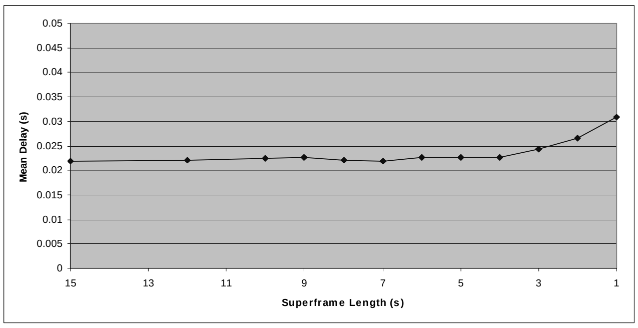


Fig. 7. End-to-end mean packet delay vs. varying super-frame length.

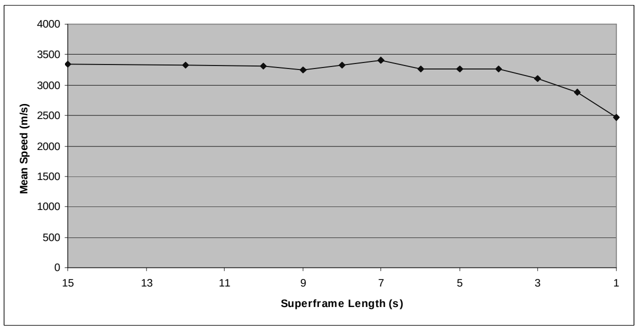


Fig. 8. End-to-end mean packet delivery speed vs. varying super-frame length.

cycles (with the assumptions explained before). Finally, we notice that the plot in Fig. 5 shows an asymptote slightly lower than 12 *mW*. This is due to the RN, which always stays awake, while all the other nodes reducing their power consumption by lowering their duty cycles. So, when the super-frame length increases greatly, the mean power consumption of a node inside an AU converges to the sum of the power consumption of the RN plus the power consumption of the other AU nodes in the sleep state, divided by the number of AU nodes.

B. QoS offered by RTPAW

To assess the QoS support offered by RTPAW the second scenario used in the previous paragraph was adopted. Here, the packet generation period ranges from a minimum of 10 seconds (with an overall network injection rate of 150 packets per second) to a maximum of 1 second (with an overall network injection rate of 1500 packets per second). The payload of the CN data packet is a 4bytes integer, which, considering the overhead due to the 802.15.4 and RTPAW headers, results in a frame length for the CN data packet of 15bytes. The final destination of every packet is the Sink node. The simulated time for this scenario was set to 25 minutes.

The graph in Fig. 6 summarizes the QoS performance in terms of end-to-end (e2e) speed hit ratio, miss ratio and dropped packets obtained by RTPAW. Here we highlight that there is a speed hit every time a packet reaches its final destination with a delivery speed greater or equal to the setpoint speed, and that end-to-end refers to the path from the source RN to the Sink node. In the opposite case, there is a speed miss. Dropped packets are due to network congestion. Referring to Fig. 6, the hit ratio always remains very close to 100% with almost any super-frame length, and decreases to about 95% with a 1-second super-frame (with an overall packet injection rate of 1500 packets per second). The percentage of late and dropped packets are both negligible in almost every simulation, and they increase equally at about 3% in the case of a 1-second super-frame.

The results in Fig. 7 and Fig. 8 represent the mean e2e delay and the mean e2e packet delivery speed, respectively, with a varying super-frame length (and therefore with a varying network load). Both delay and speed values remain almost unchanged until a 3-seconds super-frame (corresponding to a packet injection rate of 500 packets per second) is reached. When the network load increases, QoS slightly worsen, however both delay and speed values remain (considering the setpoint speed set to 1*km/s*) satisfactory.

VI. Conclusions and future work

Performance results obtained through ns-2 simulation showed the good behaviour, in terms of both QoS support and energy consumption, of the RTPAW framework. Future work will address implementation on COTS ZigBee modules and the development of novel routing protocols for RTPAW.

References

[1] W. Heinzelman, A. Chandrakasan, H. Balakrishnan, "Energy-Efficient Communication Protocols for Wireless Microsensor Networks (LEACH)". Proc. of the 33rd Hawaii International Conference on Systems Science-Volume 8, 2000, pp. 3005-3014.
[2] A. Manjeshwar, D. Agrawal, "TEEN: a Routing Protocol for Enhanced Efficient in Wireless Sensor Networks", Proc. of the 15th Internat Parallel and Distributed Processing Symp, 2001, pp. 2009-2015.
[3] Ruay-Shiung Chang and Chia-Jou Kuo, "An Energy Efficient Routing Mechanism for Wireless Sensor Networks", Proc. of the 20th Int. Conf. on Advanced Information Networking and Applications, 2006.
[4] O. Chipara, Z. He, G. Xing, Q. Chen, X. Wang, C. Lu, J. Stankovic, T. Abdelzaher, “Real-time Power-Aware Routing in Sensor Networks”, 14th IEEE Internat. Workshop on Quality of Service, 2006, pp. 83-92.
[5] T. He, J. Stankovic, C. Lu, T. Abdelzaher, “SPEED: A Stateless Protocol for Real-Time Communication in Sensor Networks”, Proc. IEEE Int'l Conf. Distributed Computing Systems, 2003, pp. 46-55.
[6] E. Felemban, C.G. Lee, E. Ekici, “MMSPEED: multipath Multi-SPEED protocol for QoS guarantee of reliability and. Timeliness in wireless sensor networks”, IEEE Trans. on Mobile Computing, 5,6, June 2006.
[7] IEEE 802.15.4 Standard Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) specifications for Low-Rate Wireless Personal Area Networks (LR-WPANs), IEEE Standard for Information Technology, IEEE-SA Standards Board, 2003.
[8] IEEE 802.15.4-2006 Standard Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) specifications for Low-Rate Wireless Personal Area Networks (LR-WPANs), IEEE Standard for Information Technology, IEEE-SA Standards Board, 2006.
[9] B. Bougard, F. Catthoor, D.C. Daly, A. Chandrakasan, W. Dehaene, “Energy efficiency of IEEE 802.15.4 standard in dense wireless microsensor networks: modeling and improvement perspectives”, Proc. of Design, Automation and Test in Europe. IEEE, 2005,V1, pp.196-201.
[10] E. Toscano, G. Kaczynski, L. Lo Bello, “RTPAW: a Real-Time Power Aware Framework for Wireless Sensor Networks”, WIP Proc. of the 13th IEEE Real-Time and Embedded Technology and Applications Symposium, Bellevue, USA, Apr. 2007, pp. 60-63, available at http://cse.seas.wustl.edu/Research/FileDownload.asp?733.
[11] The Network Simulator – http://www.isi.edu/nsnam/ns.
[12] Maxstream XBee datasheet – http://www.maxstream.net.

Preliminary Discussion on Globally Prioritized Medium Access for Multi-Channel Wireless Systems

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

Abstract

We discuss the development of a simple globally prioritized multi-channel medium access control (MAC) protocol for wireless networks. This protocol provides "hard" pre-run-time real-time guarantees to sporadic message streams, exploits a very large fraction of the capacity of all channels for "hard" real-time traffic and also makes it possible to fully utilize the channels with non real-time traffic when hard real-time messages do not request to be transmitted.

The potential of such protocols for real-time applications is discussed and a schedulability analysis is also presented.

1. Introduction

We consider the problem of designing a multi-channel medium access control (MAC) protocol for wireless networks. That is, our MAC protocol is designed for nodes equipped with radio transceivers that can receive in multiple channels at the same time. Such design aims at providing pre-run-time guarantees for real-time traffic, and thus we design a globally prioritized MAC protocol.

The fact that it is globally prioritized means that the transmission of a message is only delayed if *all* channels are being used to transmit higher-priority messages. Such a protocol would give application developers the ability of achieving "hard" pre-run-time real-time guarantees to sporadic message streams and exploit a very large fraction of the capacity of all channels for "hard" real-time traffic. It would also make it possible to fully utilize the channels with non real-time traffic when hard real-time messages do not request to be transmitted. Unfortunately, the current research literature does not (as far as we know) offer such a protocol [1].

We give a preliminary discussion on such a protocol and show how it can be designed. We also propose a schedulability analysis for it. Initially, we assume that each computer node is equipped with one transmitter module that can transmit to any selected channel and the computer node is also equipped with *CH* receiver modules where each receiver module is assigned to a specific channel (*CH* denotes the number of channels). Since contemporary hardware does not have this capability, we will later on in this paper discuss how our new protocol can be adapted to contemporary hardware.

The remainder of this paper is structured as follows. Section 2 presents the system model and the assumptions we make. Section 3 presents the MAC protocol whereas Section 4 presents its schedulability analysis. Section 5 discusses practical aspects and previous works on multi-channel MAC protocols. Finally, Section 6 gives conclusions and future work.

2. System model

Consider *m* computer nodes in a single broadcast domain, that is, every computer node can hear every other transmission. We assume $1 \le m \le UBm$, where *UBm* is an upper bound on *m*. It is assumed that computer nodes do not know *m* but they know *UBm*.

Each computer node is equipped with one transmitter module that can transmit to any channel and the computer node is also equipped with *CH* receiver modules where each receiver module is assigned to a specific channel. We say that channel 1 is the control channel, meaning that it will be used for arbitration. It is assumed that the transmission on one channel does not interfere with a transmission on another channel.

If an arbitrary computer node broadcasts an unmodulated carrier wave for *TFCS* time units, then any other node will reliably detect the existence of that carrier wave. Let *CMAX* denote the maximum packet size. We assume that the computer node can call a function `carrierOn` which causes the transmitted module to immediately start transmitting a carrier. There is also a function `carrierOff` which causes the transmitted module to immediately stop transmitting a carrier. This is close to a realistic transceiver; typically they are also able to start and stop the transmission of an unmodulated carrier within just one microsecond. Each message is assigned a priority. It is assumed that priorities are assigned such that any set

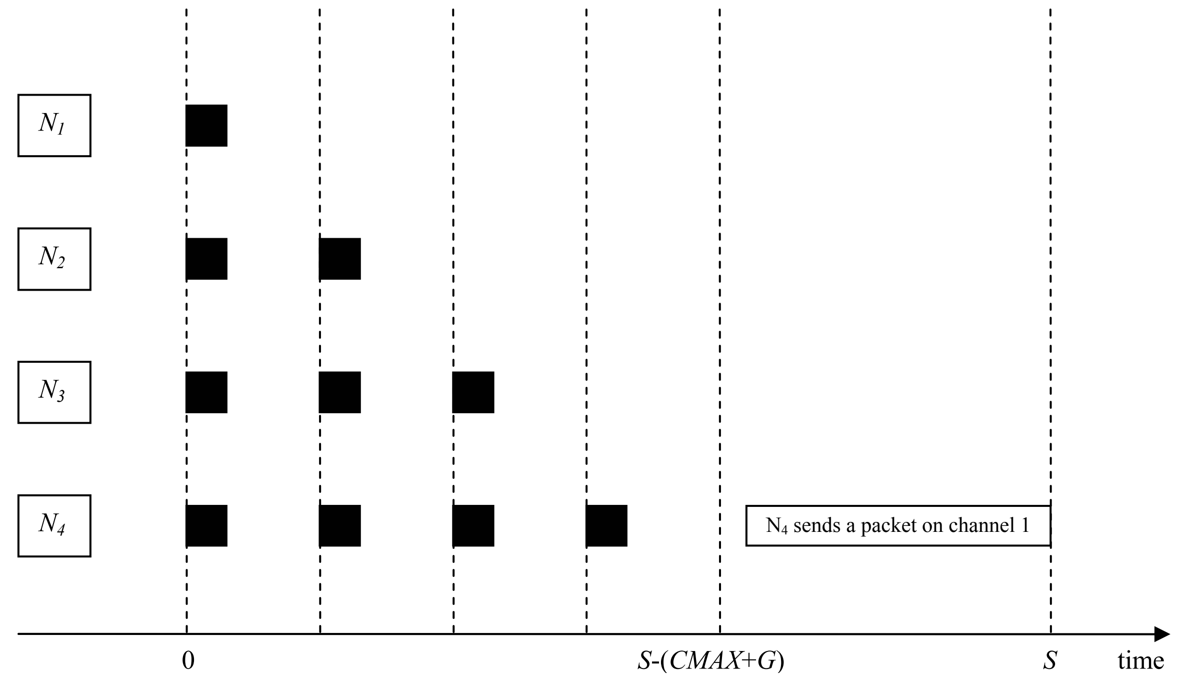


Figure 1. The illustrated MAC protocol is prioritized, but it does not allow parallel transmissions. The black filled rectangles indicate that the computer node sends an unmodulated carrier.

of messages that are contending with each other has unique priorities. One way to achieve this is to use the sporadic model (see Section 4) and assign unique priorities to message streams. It is assumed that a high number means high priority. Occasionally, we speak about the priority of a computer node and then it means the priority of the highest-priority message of that computer node.

3. Multi-Channel MAC

We will present three multi-channel MAC protocols. First, Section 3.1 presents a multi-channel MAC protocol assuming a slotted system; that is, an external device notifies all computer nodes that a slot starts. We let S denote the slot size. We do not bother about identifiers of slots; we only care about ensuring that all computer nodes know the start time of a slot.

Then we will (in Section 3.2) extend this protocol to a slotted system but without any external reference signal. Finally, we will (in Section 3.3) show how the protocol (in Section 3.2) can be designed to be used for existing transceivers. It will be designed for computer nodes equipped with only a transceiver and this transceiver can at a moment only transmit to one channel (which may be changed at run-time) or receive from one selected channel (which can be changed at run-time); that is, it cannot listen to all channels at the same time.

3.1. Multi-Channel MAC for Slotted Systems

A prioritized MAC protocol should select the CH highest priority messages among all messages that are contending at an instant. In order to build up the intuition to understand the design of the new protocol, consider Example 1.

Example 1. Consider Figure 1. It shows four computer nodes N_1, N_2, N_3 and N_4 and there are two channels available (CH=2). Each computer node requests to transmit a message at time 0. The message requested to be transmitted by node N_i has priority i.

We use a scheme similar to black-bursts [5] but we will modify it slightly. Every computer node waits for the beginning of a time slot. A time slot starts at time 0; time S, time $2S$, time $3S$, etc. We will now consider the time interval $[0,S)$. This time interval consists of the interval $[0,S-(CMAX+G))$ and $[S-(CMAX+G),S)$. (G is a parameter that will be discussed later on). The former interval is used for arbitration for the medium and the latter is for transmitting the data payload. The time interval $[0,S-(CMAX+G))$ is split into UBm subintervals. Let us index these subintervals 1,2,3,…,UBm. In Figure 1, it is assumed that UBm=m=4. A node sends a pulse of an unmodulated carrier wave in the beginning of a subinterval.

A node with priority i does this for the subintervals with index 1..i. For example, node 1 sends a pulse in the

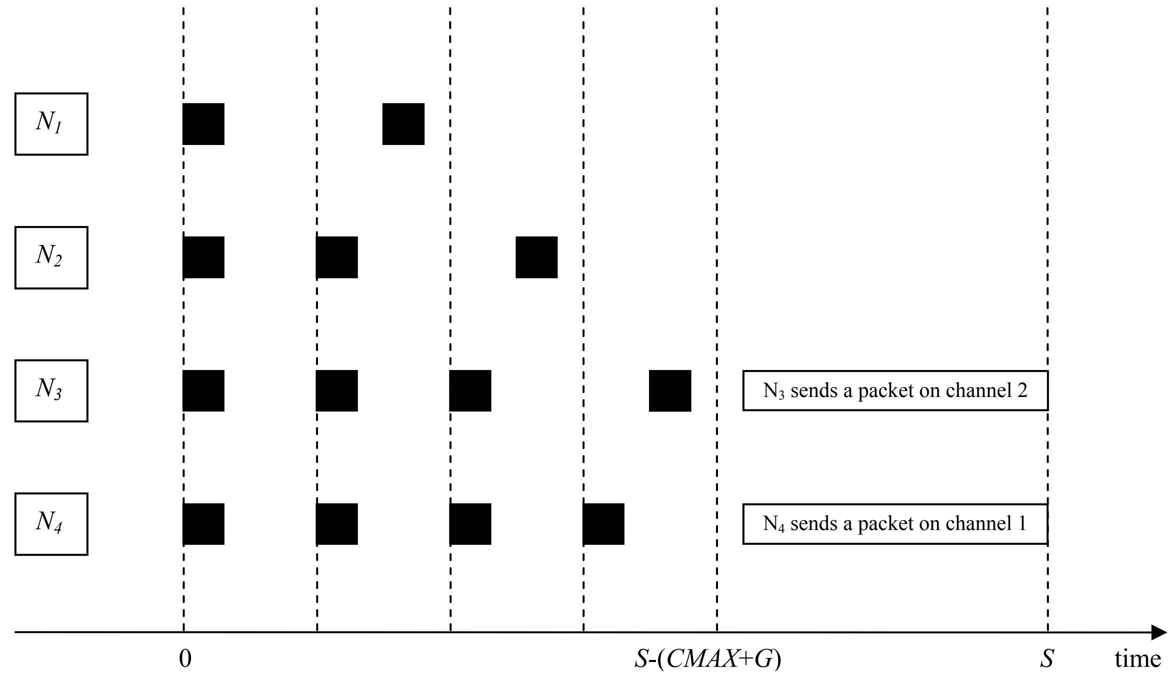

N_1
N_2
N_3
N_4
N_3 sends a packet on channel 2
N_4 sends a packet on channel 1
0
S-($CMAX$+G)
S
time

Figure 2. This MAC protocol is prioritized and allows parallel transmissions. The filled black rectangles indicate that the computer node sends an unmodulated carrier.

beginning of subinterval 1. Node 2 sends a pulse in the beginning of subinterval 1 and in the beginning of subinterval 2. For those subintervals, for which a node does not send an unmodulated carrier wave, the node performs carrier sensing in the beginning of the subinterval. For example, node 1 performs carrier sensing in the beginning of subinterval 2.

If a computer node detects a carrier then it declares itself as a loser. For example, node N_1 declares itself as a loser in subinterval 2. A computer node declares itself as a winner if it did never detect a carrier wave and then it sends its packet during [S-($CMAX$+G),S). Unfortunately, there is only one winner. □

It can be seen that the MAC protocol illustrated in Example 1 achieves prioritization. But only one computer node sends so it does not exploit the opportunity for parallel transmission on different channels. Example 2 illustrates how the behavior should be changed to allow parallel transmissions.

Example 2. Consider Figure 2. It shows four computer nodes N_1, N_2, N_3 and N_4 and there are two channels available (CH=2). Each computer node requests to transmit a message at time 0. The message requested to be transmitted by node N_i has priority i.

The computer nodes send unmodulated carriers as in Example 1. But in addition to that, a computer node sends a carrier in the later part of a subinterval if it lost in this time interval. For example, node 1 lost in the second subinterval and hence it sends an unmodulated carrier in the later part of the second subinterval. In each subinterval, there can be at most one node that lost (assuming that priorities are unique). For this reason, all computer nodes will not only know the priority of the winner, but they will also know for each priority level if this priority level lost. And in this way, all nodes will know the priority of the node with the highest priority, the priority of the node with the second highest priority, and so on. This allows us to design a MAC protocol that achieves global prioritization. □

Having seen the main idea on how to design a globally prioritized MAC protocol we are now in position to formally state the new protocol.

To simplify the presentation of the protocol, we do it using timed-automata like notation. States are represented as vertices and transitions are represented as edges. An edge is described by its guard (a condition which has to be true in order for the protocol to make the transition) and an update (an action that occurs when the transition is made). We let "/" separate the guards and the updates; the guards are before "/" and the update is after. Let "=" denote test for equality and let ":=" denote assignment to a variable. For those transitions with an update having many lines of code, it is assumed that the lines are executed sequentially.

Figure 3 shows the automaton. It describes the behaviour that every computer node does. First a

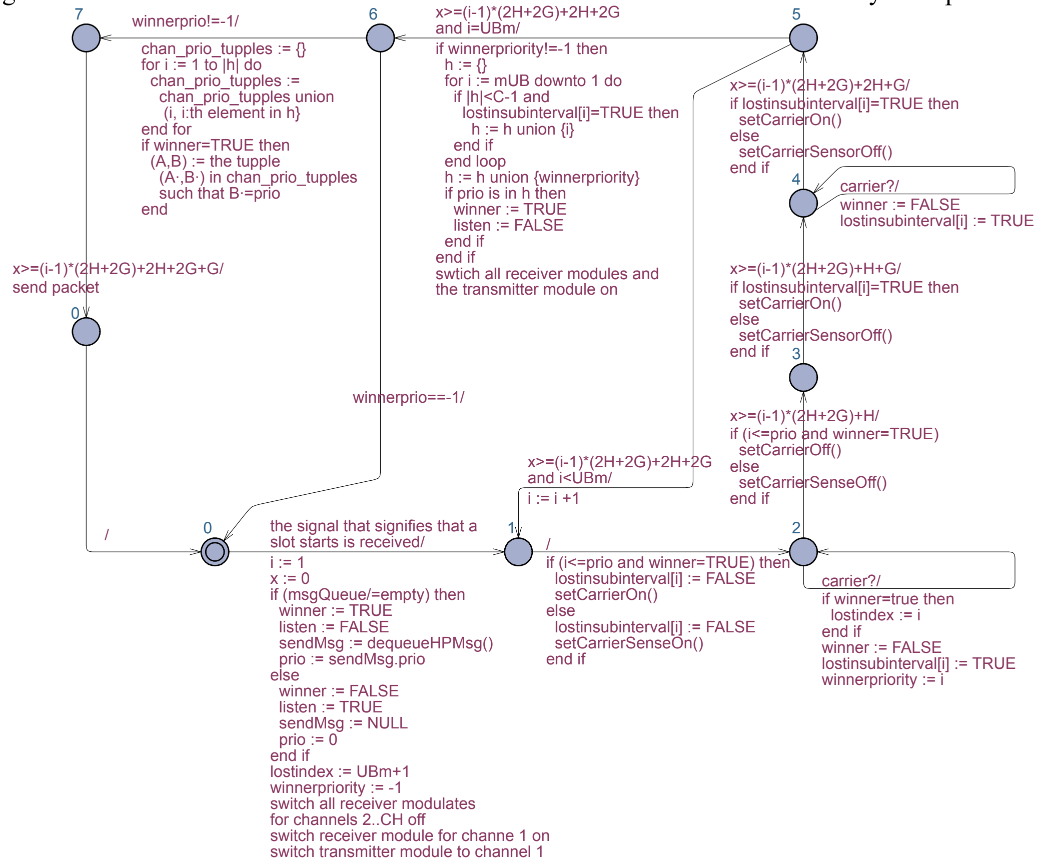


Figure 3. A timed automaton description of the proposed multi-channel MAC protocol.

computer node waits until it receives a signal that signifies that a time slot has begun. When that happens it makes a transition from state 0 to state 1 (state 0 is the
initial state). Then the protocol iterates through all subintervals (in states 1-5). A computer node makes the transition to state 6 when all subintervals have been executed. When this transition takes place, the computer node computes a set h which contains all the *CH* highest priorities that contended. A node may have lost the tournament but it had still high enough priority so for this reason, it may be declared as a winner (one of the computer nodes that will send). Computer nodes make the transition to state 7 where a mapping from priorities to channels is computed. The computer nodes that are winners send their packets on the channel given by this mapping. Computer nodes that are not winners do nothing; they already listen to all channels.

The protocol depends on timeout parameters *G* and *H*. They should be selected to satisfy the following constraints:

$$2H + 2G + G + CMAX \le S \tag{1}$$

and

$$TFCS \le H \tag{2}$$

and

$$\begin{array}{c}\textit{difference in time that computer node receive the}\\ \textit{signal that signifies that a time slot starts} \le G\end{array} \tag{3}$$

It can be seen that *H* is duration of a pulse and *G* is a guard band.

3.2. Multi-Channel MAC for Slotted Systems without External Synchronization

The protocol described in Section 3.1 can be easily extended to slotted systems without external synchronization. We simply let computer nodes wait for a long period of silence and then send an unmodulated synchronization pulse (as was done in WiDom [6], a prioritized MAC protocol for single-channel wireless systems).

3.3. Using Contemporary Transceivers

Typical contemporary transceivers are only able to either receive from a single (selectable at run-time) channel or transmit to a single (selectable at run-time) channel. For such transceivers, the MAC protocol from Section 3.2 can be used with the only restrictions that (i) if a computer node is a winner then it cannot hear any transmitted packet and (ii) if a computer node is not a winner then it can only hear one transmitted packet. Assuming that transmissions are unicast then we can require that the intended receiver sends an ACK (this can be considered to be part of *CMAX*) to the sender on the channel the sender used. If the sender receives an ACK then it knows that the receiver listened on that channel and hence the packet is successfully transmitted. If no ACK was received then the sender retries in the next "slot".

4. Schedulability analysis

In order to perform schedulability analysis, it is necessary to describe a model of the traffic. We consider the sporadic model. It is as follows. A computer node is assigned zero, one or many message streams. A message stream is assigned to exactly one computer node.

A message stream τ_i is characterized by D_i and T_i, where D_i is the relative deadline and T_i is the minimum inter-arrival time. A message stream τ_i performs (a possible infinite) sequence of message transmission requests. The time between two consecutive message transmission requests in a message stream is at least T_i. The time to transmit a message in message stream τ_i is at most *CMAX*. For this reason, we can assume (from the perspective of schedulability analysis) that the length of a message is S and this includes the time for arbitration. We also assume the constrained deadline case, that is, $\forall i$: $D_i \leq T_i$. We assume that priorities are assigned according to deadline-monotonic (DM) [7]; that is, the priority of a message stream is inversely proportionate to its deadline.

Inspired by results in static-priority scheduling on multiprocessors [2] and combining this way of thinking with results from the CAN analysis [3] gives that we can calculate an upper bound on the response time when the MAC protocol in Section 3.1 is used. The upper bound is obtained as follows. Find the minimum value of RUB_i that satisfies both (4) and (5):

$$RUB_i \leq S + S + \left\lceil \frac{1}{CH} \times \sum_{j \in hp(i)} \left\lceil \frac{RUB_i + S}{T_j} \right\rceil \right\rceil \times S \qquad (4)$$

$$RUB_i \leq S + S + \sum_{\substack{j \in hp(i) \wedge j \text{ is a message stream} \\ \text{on the same computer node as } i}} \left\lceil \frac{RUB_i + S}{T_j} \right\rceil \times S \qquad (5)$$

where $hp(i)$ denotes the set of message streams in the entire network that have higher priority than message stream τ_i. Observe that messages in $hp(i)$ may be assigned to other nodes than message stream τ_i. The first term in (4) is due to blocking; the second term is due to transmission and the third term is due to interference. Observe that we add S to the window that is used to compute interference. This is because the blocking due to lower-priority messages can increase the window of interference. The reasoning for deriving (5) is similar to that of deriving (4). The inequality (5) accounts for the fact that even if many channels are available, a single computer node can send at most one packet per time slot.

It can be seen that a large number of channels help to reduce the interference and hence it reduces the response time. Observe that the real response-time may be smaller than RUB_i. This is due to (i) pessimism in the analysis of non-preemptive static-priority scheduling and (ii) pessimism in the "multi-channel" aspect. We know however that if $\forall i$: $RUB_i \leq D_i$ then all deadlines are met.

The response-time calculations from (4) and (5) is valid for the MAC protocol in Section 3.1 But it is not valid for the MAC protocol in Section 3.2 and Section 3.3 because those protocol use a specific type of synchronization and it depends in a technique called "delayed dequeing" which complicates the analysis. See [6] for details.

One may ask whether DM is optimal for the system that we assume. We guess the answer is no. One may also ask whether Dhall´s effect [4], a scenario that occurs on multiprocessor scheduling that can cause deadlines misses although the multiprocessor/multiple channels are almost idle all the time but still a deadline is missed, can occur. Our guess is that if all messages have similar length then this is not a big problem.

5. Practical Aspects and Previous work

5.1. Practical Aspects

We assumed that a radio channel offers reliable broadcast. Whether this is reasonable in practice is debatable and it depends on the exact location of computer nodes, radio conditions, transmission power and detection techniques. Nonetheless, we have in previous work shown that there are environments where such assumption is reasonable [6].

We assumed it is possible to detect a carrier wave if this wave is transmitted for a duration of *TFCS* time units. The exact value of *TFCS* depends on the hardware being used. In CC2420 (a transceiver for 802.15.4), this time is 128μs. It has been reported that other radios can has *TFCS* of 5μs [10] and 20μs [5].

We assumed initially that each computer node is equipped with multiple receiver modules and one transmitter module. Such a computer node is more costly than computer nodes with normal transceivers and we are not aware of any such computer node on the market today. A computer node with a separate transmitted module and receiver module has been built (by others [8] in a collaborative project with us) and it would be possible to add more receiver modules. Consequently, the construction of a computer node with multiple receiver modulate is at least technically possible.

We assumed that when a computer node transmits to a channel then it causes no interference to transmission son other channels. Many current standards (such as IEEE 802.11 and IEEE 802.15.4) support many channels and they occupy different frequency band. The standard IEEE 802.11b has 14 channels (5 MHz each) but in order to for transmissions to be totally non-overlapping it is necessary that channels are separated by 30 MHz. Hence channel 1,6,11 in the IEEE 802.11 standard can transmit in parallel. It would be possible to use our protocol by letting channel 1 in IEEE 802.11 be our channel 1, letting channel 6 in IEEE 802.11 be our channel 2 and 11 in IEEE 802.11 be our channel 3. Then we would have *CH*=3.

5.2. Previous work

The scientific advances in multi-channel MAC protocol originates from two time periods: (i) before the IEEE 802.11 standard and (ii) after IEEE 802.11.

Before the IEEE 802.11 standard was proposed, significant research was performed on ground packet radio network, particularly in the U.S. One of the earliest multi-channel MAC protocols was proposed *receiver-directed transmission* [9]. Here it was assumed that each computer node is assigned a channel and it listens only to that channel. A computer node knows, for every of its neighbors, which channel this neighbour listens to. When a computer node wishes to send, it switches to the channel that the receiver listens to.

After IEEE 802.11 several multi-channel MAC protocol were proposed and they were more flexible. They can be categorized [1] as (i) dedicated control channel, (ii) common hopping, (iii) split phase and (iv) multiple rendezvous. They all have in common that a node that wishes to send a packet first sends a request-to-send (RTS) packet and if the receiving computer node receives this RTS packet, it responds with a Clear-to-Send (CTS) packet. They differ in when and on which channel this RTS/CTS exchange is performed and on which channel the subsequent data transmission is performed.

6. Conclusions and Future work

We have presented a globally prioritized multi-channel MAC protocol for wireless systems and a schedulability analysis for it.

We left four problems open and we consider them as future work. First, this protocol was based on the black-burst scheme [5] which offers few priority levels. So a natural question is: Can the prioritized MAC protocol WiDom [6] be extended to multi-channel wireless systems. If so a large number of priorities can be supported even with a small overhead. It is clear that one can run the normal WiDom *CH* times but we would like to design a protocol with an overhead lower than that. Second, we would like to make the schedulability analysis tighter and explore alternative priority-assignment schemes. Third, we would like to design a schedulability analysis technique that takes into account that fact that a message may be transmitted but a receiver does not listen to it and hence the message must be retransmitted. Fourth, we would like to implement the new protocol in contemporary transceivers.

References

[1] J. Mo, H. Sheung, W. So and J. Walrand, ”Comparison of multi-channel MAC protocols”, International Workshop on Modeling Analysis and Simulation of Wireless and Mobile Systems, pages 209 – 218, 2005, Montréal, Quebec, Canada.

[2] B. Andersson and J. Jonsson, "Some Insights on Fixed-Priority Preemptive Non-Partitioned Multiprocessor Scheduling", in Proc. of the IEEE Real-Time Systems Symposium, Work-in-Progress Session, November 27-30, 2000, Orlando, Florida.

[3] R. I. Davis, A. Burns, R. J. Bril and J. J. Lukkien, ”Controller Area Network (CAN) schedulability analysis: Refuted, revisited and revised”. Journal of Real-Time Systems, pages 239-272, vol 35, issue 3, 2007.

[4] S. Dhall and C. Liu, "On a real time scheduling problem", Operations Research, 26(1), 1978, pages 127-141.

[5] J. L. Sobrinho and A. S. Krishnakumar, "Real-time traffic over the IEEE 802.11 medium access control layer," Bell Labs Tech. J., vol. 1, no. 2, pp. 172-187, 1996.

[6] N. Pereira, B. Andersson and E. Tovar, “WiDom: A Dominance Protocol for Wireless Medium Access”, IEEE Transactions on Industrial Informatics, Vol 3, No 2, May 2007.

[7] J. Y. Leung and J. Whitehead, "On the complexity of fixed-priority scheduling of periodic, real-time tasks," Performance Evaluation, 2(4):237-250, December 1982.

[8] http://www.nanork.org:8000/nano-RK/wiki/wings

[9] N. Shacham and P. King, “Architectures and Performance of Multichannel Multihop Packet Radio Networks”, IEEE Journal on Selected Areas in Communications, 5(6), July 1987, pages 1013- 1025.

[10] F. Tobagi and L. Kleinrock, "Packet Switching in Radio Channels: Part II--the Hidden Terminal Problem in Carrier Sense Multiple-Access and the Busy-Tone Solution", IEEE Transactions on Communications, Vol. COM-23, No. 12, pp. 1417-1433, December 1975.