Slot Allocation Schemes for the FlexRay Static Segment

Rodrigo Lange[1,2], Francisco Vasques[1], Paulo Portugal[1], Rômulo S. de Oliveira[2]

[1]IDMEC/ISR - Faculty of Engineering, University of Porto, Portugal

{lange,vasques,pportuga}@fe.up.pt

[2]DAS - Federal University of Santa Catarina, Florianópolis, Brazil

{lange,romulo}@das.ufsc.br

Abstract

In recent years, the FlexRay communication protocol has been promoted as a *de facto* standard for in-vehicular communications. In the FlexRay protocol, the communication timeline is organized as a sequence of four segments, whereas the static segment defines a set of slots specifically designed for the transmission of synchronous messages. In this paper, we investigate the following problem: "how to allocate a minimum number of static slots to each FlexRay node, while guaranteeing that all synchronous messages will be transmitted before their deadlines". Unlike previous studies that use linear programming based techniques, we evaluate the number of allocated slots using traditional response time analysis (RTA) techniques. The use of RTA techniques allows us to consider the timing requirements imposed by the set of synchronous message streams. Unlike other approaches, the RTA-based technique proposed in this paper is able to deal with a) message stream sets where periods are not multiple of the FlexRay cycle duration and b) the generation of messages at the application layer that are not synchronized with the FlexRay cycle. The proposed allocation schemes may be integrated into an AUTOSAR environment as an RT-Middleware, allowing FlexRay real implementations to take full advantage of the proposed schemes.

I. Introduction

The automotive industry is currently one of the largest economic sectors in the world, producing every year tens of millions of vehicles and contributing significantly to the revenue of many countries. In recent years, this industry has seen an exponential growth in the number of embedded applications. This whole class of applications includes electronic navigation,

traction control, stability control, active safety, multimedia and many other computerized systems. Modern vehicles have more than 70 electronic controllers exchanging more than 2500 signals representing information as elementary as speed or engine temperature [1].

The growth in complexity of electronic architectures, together with the timing requirements related to sensors and actuators, is leading to the adoption of a distributed implementation. Within this context, networks and communication protocols are of utmost importance since they support the integration of distributed functions, reducing the cost and complexity of cabling and providing means for the implementation of advanced fault tolerance techniques.

Typically, a vehicle embedded system is sub-divided in functional domains, each one having its own characteristics and constraints. Two of these domains are specifically related to the vehicle real-time control: the power train, which is related to engine and transmission control, and the chassis, which is related to the suspension control, steering and brakes. Safety issues are fundamental in these domains and the related technical solutions must ensure the system reliability. In systems currently being used in vehicles, these functions are distributed across multiple electronic control units (ECUs) that are connected by a data communication network. The Controller Area Network (CAN) has been a *de facto* standard for communications in automotive applications, with over 400 million CAN devices produced each year [2].

According to several authors, CAN is not suitable for the future X-by-wire applications due to some serious limitations in what concerns performance and safety aspects. New protocols for automotive applications are under development [3], being the **FlexRay Communication System** the most probably future *de facto* standard for vehicle communications [3], [4].

The FlexRay Communication System is a digital serial bus for automotive applications designed to meet the demands of bandwidth, reliability, determinism and synchronization of X-by-Wire systems. Among other features it provides flexibility, bandwidth and determinism by combining static and dynamic approaches for message transmission, incorporating the advantages of synchronous and asynchronous protocols. FlexRay was developed between 2000 and 2009 by an alliance of manufacturers that included BMW, DaimlerChrysler and Bosch [5]. The FlexRay media access control (MAC) protocol is based on a fixed length cycle that periodically repeats itself. Each cycle is composed of a static segment (ST) and a dynamic segment (DN). The static segment employs a media access method that is similar to time-division multiple-access (TDMA).

The design of the FlexRay static segment plays an important role in the overall system design. It is a task that involves defining parameter values for, among other, the definition

of the number and size of the static slots, the length of the static segment and the evaluation of the number of allocated slots.

State-of-the-art approaches traditionally generate schedules where message streams are associated with unique slots and the schedule itself has an unique cycle repetition. Actually, this is a requirement of the AUTOSAR standard that, in practice, restricts the periods of message streams to be integer multiples of the FlexRay cycle. Another traditional approach is to use optimization techniques such as Integer Linear Programming (ILP) that have high computational requirements and can be unacceptable in practice due to its potentially long computation times [4]. Finally, although asynchronous scheduling approaches could be of high practical interest when considering the migration from protocols like CAN [6] to FlexRay, most of the existing works assume a strong synchronization between tasks, signals and the FlexRay cycle, imposing a system-based synchronization.

In this work, we propose a different approach to deal with the design of the static segment. Basically, we evaluate the number slots required by each FlexRay node, using traditional response time analysis (RTA) techniques. Therefore, the number of required slots will be function of the timing requirements imposed by the set of synchronous message streams. Unlike other state-of-the-art approaches, the technique proposed in this paper is able to deal with message stream sets where periods are not multiple of the FlexRay cycle duration, nor the message arrivals are synchronized with the FlexRay cycle. The proposed approach can still be integrated in AUTOSAR based systems, as it will be demonstrated in the paper with an application example.

The remainder of this paper is organized as follows. Section II summarizes the state-of-the-art works about scheduling the FlexRay static segment. Sections III and IV present a brief description of FlexRay and AUTOSAR, respectively. Section V presents the network system model assumed in this work. Section VI describes the real-time middleware that may support the proposed scheduling approach. Sections VII and VIII describe the proposed slot allocation scheme, with a complete example being illustrated in Section IX. Finally, Section X explains how the proposed schemes can be integrated in the AUTOSAR environment, and Section XI presents the conclusions of this work.

II. Previous Relevant Work

In this section, we report the most relevant works related to the design of the FlexRay static segment that can be found in the literature.

In [4], the main focus is the schedule of the FlexRay dynamic segment. This paper also presents three approaches to define the bus configuration for a FlexRay system. Each approach tries to define the most favorable cycle length, using different methods to set the length of the static segment. The first approach considers that there is an allocation of just one static slot per node in the system, and then tries to define the length of the FlexRay cycle considering the requirements of the dynamic segment. The second approach considers a greedy heuristic that explores different alternatives to the length of the static segment, considering only the restrictions imposed by the FlexRay specification over the minimum and maximum number of static slots in each FlexRay cycle. The third approach is a simulated annealing based approach that considers the length of the FlexRay cycle and segments as the parameters for an optimization problem.

The work presented in [7] deals with the problem of packing signals into frames that will be transmitted in the FlexRay static segment. This work considers the case where tasks producing signals are not synchronized with the FlexRay communication cycle. It also considers that signals are transmitted and received through an AUTOSAR architecture. The autors present two different algorithms to pack signals into frames. Both algorithms consider that the size and number of allocated slots are parameters to be minimized in order to save bandwidth. In this work, the authors assume that the length of both the FlexRay cycle and the static segment are known *a priori* and are not subject to optimization.

The work presented in [8] describes an approach that uses genetic algorithms to find feasible schedulings for the FlexRay static segment. This work considers that the period of tasks and messages in a system are integer multiples of the FlexRay cycle.

In [9] it is also addressed the scheduling of the FlexRay static segment in compliance with the AUTOSAR standard. In this work the authors propose the use of a greedy heuristic to transform the scheduling problem into a special two-dimensional bin packing problem (BPP), solving the BPP through an ILP formulation. The minimization of the number of static slots is the parameter to be optimized within the proposed approach. This work considers that the length of both the FlexRay cycle and of each segment are predefined.

The work presented in [10] addresses the problem of building feasible and efficient schedules for the FlexRay static segment, starting from the signal data to be transmitted. In this work the authors divide the scheduling problem in two subproblems. The first subproblem is packing signals into message streams of equal size, using a Non-linear Integer Programming (NIP) approach that maximizes bandwidth utilization, while computing an optimal message

stream set. The second subproblem is the definition of the message schedule at each node. To this end, it is proposed an Integer Linear Programming (ILP) formulation that a) tries to minimize the jitter for periodic messages and b) defines the allocation of static slots to the message streams. This research work assumes that all signals have to be scheduled in integer multiples of the FlexRay cycle, and signal periods and deadlines are also defined as integer multiples of the FlexRay cycle. The message schedule computation is carried out assuming an additional software architecture on top of FlexRay.

As an extension to [10], the work presented in [11] proposes an ILP approach to assign a frame identifier, a schedule repetition and a schedule offset to each periodic message stream in the system. As in [10], this work assumes that all messages must be scheduled in integer multiples of the FlexRay cycle. Signal periods and deadlines are also expressed in multiples of FlexRay cycles. The authors also consider that the cycle repetition is a free parameter that can be used in the ILP problem.

In [6], it is presented a Mixed Integer Linear Programming (MILP) framework to define signal to frame packing, frame to slot assignment, task schedule and the synchronization of signal and task scheduling in an AUTOSAR environment. The minimization of the used static slots and the maximization of the minimum laxity are the metrics for the MILP formulation. The proposed approach assumes that tasks and signals are synchronized, and it is also assumed that the FlexRay communication cycle, the static segment length and the slot size are predetermined. This work discusses two possible synchronization patterns between tasks and signals (synchronous and asynchronous) and their relationship with the AUTOSAR standard. Similarly, the paper discusses possible options to define a FlexRay system, and the relationship between these choices and AUTOSAR.

Some common aspects in the above reported works can be highlighted. A significant part of these works consider that periods of signals or message streams are integer multiples of the FlexRay cycle [8], [10], [11]. Nevertheless, this assumption that is induced by the AUTOSAR standard (and not by the FlexRay standard itself) can restrict the available design options. Similarly, some of the reported approaches assume that lengths of the FC, DN and ST are previously known [6], [7], [9], and therefore cannot be subject to further optimization.

Additionally, a relevant number of research works assume a strong synchronization between tasks, signals/message streams and the FlexRay cycle [6], [7], [10], [11]. However, a solution addressing asynchronous scheduling could be of high practical interest when considering the remapping of existing CAN message streams into FlexRay ones [6].

On the other hand, several of the above reported works employ optimization techniques (NIP, ILP or MILP) to define optimal sets of parameters (optimal in the sense that the resulting parameters are optimal for the given message stream set) [6], [8]–[11]. However, optimization techniques are known to be computationally expensive, and can be unacceptable in practice due to their long computation times [4].

Finally, there is a common characteristic to all the above reported works: signals or message streams are statically associated with unique static slots and unique FC repetition cycles, without any further optimization. Actually, this is an requirement induced by the AUTOSAR standard (and not by the FlexRay standard itself) that, in practice, restricts the periods of the message streams to be integer multiples of FC.

III. FlexRay Communication System

FlexRay is a communication system for in-vehicular communications, that has been developed by an alliance of manufacturers including BMW, DaimlerChrysler and Bosch. Its main target is to meet future communication requirements for determinism, synchronization, bandwidth and reliability in automotive systems, such as those related to X-by-Wire. FlexRay provides static and dynamic methods for the transmission of messages, incorporating the advantages of well-known synchronous and asynchronous protocols. Among other features, it provides fault-tolerant clock synchronization, collision-free medium access, two communication channels with transmission rates up to 10Mbit/s per channel and a data payload up to 254 bytes per frame [5].

In FlexRay, the Media Access Control (MAC) is ruled by a communication cycle with a predetermined size FC_{bus}. A FlexRay cycle (FC) runs periodically from the beginning to the shutdown of the network. The cycle number is controlled by the variable $vCycleCounter$ that is incremented by one at the beginning of each communication cycle. $vCycleCounter$ ranges from 0 to $cCycleCountMax$ ($cCycleCountMax$ is a system constant whose value is 63). When $cCycleCountMax$ is reached, the cycle counter $vCycleCounter$ shall be reset to zero in the next communication cycle instead of being incremented.

Each FlexRay cycle is structured as a sequence of four segments: a *Static Segment* (ST), a *Dynamic Segment* (DN) and two control segments, the *Symbol Window* (SW) and the *Network Idle Time* (NIT) (Figure 1).

The ST is based in the Time Division Multiple Access (TDMA) technique. It has size ST_{bus} and it is composed of a number $gNumberOfStaticSlots$ of static slots that are

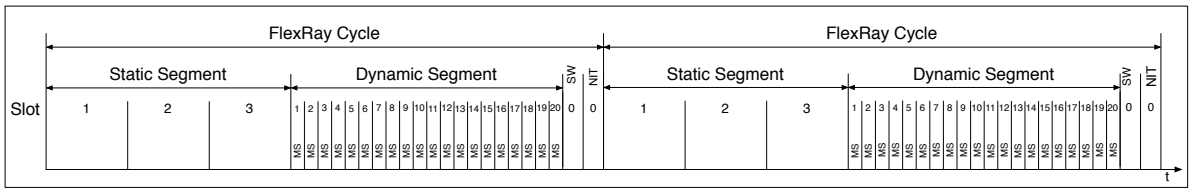

Figure 1. FlexRay Cycle

incrementally counted at each communication cycle. Each static slot is composed of the same number of macroticks whose value is set by $gdStaticSlot$. In a FlexRay cluster, $gNumberOfStaticSlots$ and $gdStaticSlot$ are global values that must be defined during the design phase. Arbitration in the ST is performed by assigning frame identifiers ($FrameIDs$) to each network node. When the $FrameID$ assigned to a specific node matches the static slot counter, that node is allowed to transmit. The ST supports a deterministic communication environment, as it is exactly defined when each frame will be transmitted in a given channel [3]. It is mandatory that each FlexRay node is associated with at least one $FrameID$ of the static segment. It is optional that a FlexRay system implements the dynamic segment.

FlexRay allows slot multiplexing, which means that the same $FrameID$ can be used for different messages in different FCs. As $vCycleCounter$ periodically counts from 0 to 63, assignments that repeat every 1, 2, 4, ..., 64 FCs can be realized. The cycle multiplexing is described using this repetition of the respective FC assignment and using the information about the first FC among the possible 64 FCs where the respective assignment starts [11].

The Dynamic Segment, the NIT and the SW will not be addressed in this work. More information about these segments can be found in [4], [5].

IV. AUTOSAR

The Automotive Open System Architecture (AUTOSAR) has been proposed by a partnership of automotive manufactures and supliers. This partnership works together to develop and estabilsh a de-facto open industry standard for automotive architectures [12].

In AUTOSAR, an application is modeled as a composition of interconnected components. A communication mechanism called "virtual functional bus" (VFB) allows the interaction between components (upper half of Figure 2). During the design phase, components are mapped on specific system resources (ECUs) and the virtual connections among them are mapped onto local connections (within a single ECU) or upon network-technology specific

communication mechanism, such as FlexRay frames (Figure 2, botton half).

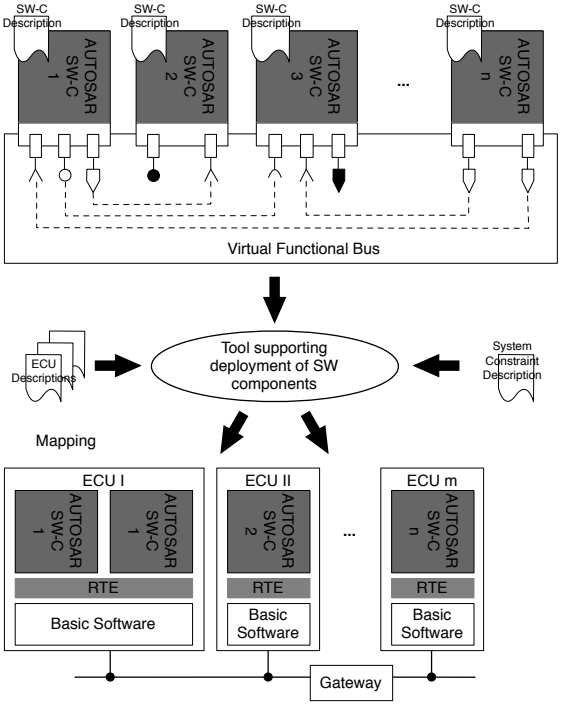

Figure 2. VFB View [12]

AUTOSAR uses a layered architecture that ensures the decoupling between the functionality and the supporting hardware or software services. For a single ECU, the AUTOSAR defines an architecture with three layers: **Application Layer**, which is composed by "Software Components" (SW-Cs) that encapsulate complete or partial automotive functionality; **Real-Time Environment (RTE)**, which is the concrete interface between the SW-Cs and the remainder of the ECU system; **Basic Software Layer**, which is standardized software that does not have any specific functionality, but offers hardware-dependent and hardware-independent services to the layer above (RTE).

The Basic Software Layer is composed by the **Services Layer**, the **ECU Abstraction Layer**, the **Complex Drivers** and the **Microcontroller Abstraction Layer**.

The **Services Layer** offers operating system functionality, network communication and management services, memory services, diagnostic services and ECU state management. The **ECU Abstraction Layer** interfaces the drivers of the Microcontroller Abstraction Layer. Its parts are the *Onboard Device Abstraction*, *Memory Hardware Abstraction*, *Communication Hardware Abstraction* and *I/O Hardware Abstration*. **Complex Drivers** implements complex

sensor and actuator control with direct access to the microcontroller using specific interrupts and/or complex μC peripherals. Finally, the **Microcontroller Abstraction Layer** is the lowest software layer in the Basic Software. It contains the internal drivers and it is composed by *Microcontroller Drivers*, *Memory Drivers*, *Communication Drivers* and *I/O Drivers*.

The communication within AUTOSAR is based in *Protocol Data Units* (PDUs). A PDU receives a prefix that varies according to the AUTOSAR layer. A message generated in a SW-C at the application layer is packed into an $ISignalIPdu$. An $ISignalIPdu$ is packed into an *Interaction Layer PDU* (I-PDU) by the AUTOSAR Com module. If an I-PDU passes through the TP module, it is packed into a *Network Layer PDU* (N-PDU). The protocol interface sends *Data Link Layer PDUs* (L-PDUs) to the protocol driver. The protocol driver sends the FlexRay frames to the bus. The full naming convention for PDUs can be found in [12].

The path of a PDU through the VFB is defined during the design phase and cannot be changed during the system execution. This means that the relationship between a PDU of one layer with a PDU of another layer can not be modified during run-time.

In AUTOSAR, the modules that are related to the FlexRay protocol are grouped in the **FlexRay Communication Services**. The relationship between modules is shown in Figure 3. The prefixes that a PDU receives at each level of the architecture are also depicted in the Figure.

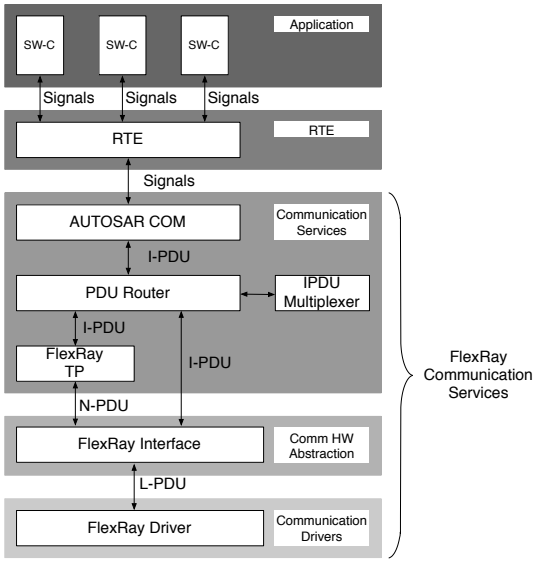

Figure 3. AUTOSAR Basic Software related to FlexRay

The **Communication Services** are a group of modules for vehicle network communication. They are interfaced with the communication drivers via the communication driver interface. AUTOSAR Specification 4.0 defines modules for FlexRay, CAN, LIN and Ethernet. Communication Services contain, among other modules, the *PDU Router*, which route the I-PDUs between different abstract communication controllers and upper layers, the *COM*, which provides routing of individual signals or groups of signals between different I-PDUs, the *FlexRay Transport Protocol* (FlexRay TP), which is responsible for the segmentation and reassembly of messages (I-PDUs) that do not fit in one of the assigned FlexRay L-PDUs (FlexRay frames), and the *IPDU Multiplexer*, which provides the possibility to add information that enables the multiplexing of I-PDUs (different contents but same IDs).

The **Communication Hardware Abstraction** (Comm HW Abstraction) is a group of modules that provide equal mechanisms to access a bus channel regardless of it's location (on-chip/on-board). The Comm HW Abstraction contains the *FlexRay Interface (FrIf)*, which provides standardized mechanisms to access a FlexRay bus channel via a FlexRay Communication Controller. *FrIf* does not make any PDU payload-dependent routing decisions. Each datum that should be transmitted or received has to be scheduled at system configuration time. This even holds true for event-driven data. The *FrIf* module takes the information on how to pack PDUs into FlexRay frames from the so-called Frame Construction Plans. The rules defining these packings are defined at configuration time.

If an I-PDU or N-PDU is intended to be transmitted over a FlexRay bus, it must be associated with exactly one FlexRay FrameID, base cycle and cycle repetition. This association is defined during the design phase and cannot change during the normal operation mode of FlexRay. Consequently, the association between a message stream and a static slot can not be changed during FlexRay's normal operation mode.

To avoid concurrent access to the transmit and receive buffers by the hardware/software, the *FrIf* module shall call all functions accessing the transmit and receive buffers only at well-defined points in time. These points in time are synchronized to the FlexRay Global Time. To provide the necessary synchronization, the *FrIf* module defines for each cluster a FlexRay Job List.

By definition, a FlexRay Job List is a list of (maybe different) Communication Jobs sorted according to their respective execution time [12]. Each Communication Job contains the following properties:

- Job start time by means of

  - FlexRay Communication Cycle. The AUTOSAR standard defines that, for FlexRay v3.0 hardware, the base cycle ranges from 0 to 63, and the cycle repetition has the value 2^R, where $R \in [0, ..., 6]$.
  - Macrotick Offset within the Communication Cycle
- A list of Communication Operations sorted according to a configurable Communication operation index.

The *FrIf* module shall process the actions determined by the Communication Operations assigned to each FlexRay Communication Job. Each Communication Operation contains an index that determines its execution order, a Communication Action which specifies the actual action to perform and a reference to a frame triggering which is associated with the Communication Action to perform.

To more informations about FlexRay Jobs and Communication Operations please refer to [13].

V. System Model

After describing the basic operation of both FlexRay Communication System and the AUTOSAR Architecture, we will present now an overview of the system under consideration, including the message and network models. Basically, these three steps will allow us to introduce the two major contributions of this paper, which are a methodology to define the minimum number of static slots that must be allocated to each FlexRay node, in order to guarantee that all synchronous messages will meet their deadlines, and the integration of such methodology within an AUTOSAR environment.

A. Message Model

We consider a distributed system that communicates by message exchanges. Such message exchanges are modeled by a set of m synchronous messages streams $S_1, S_2, ..., S_m$. The related synchronous message set $\mathbb{S}$ is:

$$\mathbb{S} = \{S_1, S_2, ..., S_m\}. \tag{1}$$

Messages generated by each message stream are transmitted through the FlexRay static segment. Each message stream $S_i \in \mathbb{S}$ can be characterized as a tuple (C_i, P_i, D_i) where:

- C_i is the maximum amount of time required to transmit a generated message. It is considered that C_i includes both the payload and the required headers and tails.

- P_i is the message interarrival time. As synchronous messages are periodically generated messages, P_i is constant and equal to the generation period.
- D_i is the relative deadline of a generated message. In this work it is considered that $D_i = P_i$.

A message generated by message stream S_i is represented by M_i, where M_i^k ($k = 1, 2, ..., +\infty$) is the $k - th$ message generated by S_i.

The utilization factor U of the synchronous message set is defined as the fraction of time spent by the network for the transmission of the synchronous messages:

$$U = \sum_{i=1}^{m} \frac{C_i}{P_i}. \tag{2}$$

B. Network Model

Similarly to other FlexRay research works, it is considered a network topology with n network nodes interconnected by one FlexRay communication channel (Figure 4). That is, the use of the second communication channel is not considered in this work.

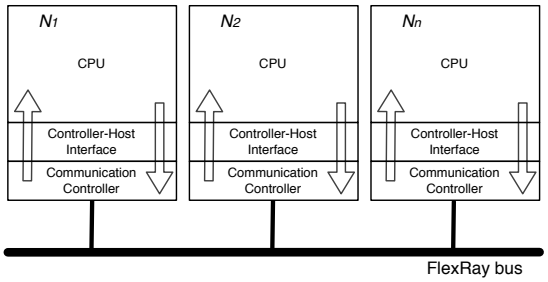

Figure 4. Network Topology

An integer number H_j of static slots is allocated to each FlexRay node N_j ($j = 1, 2, ..., n$). The Proportional Allocation Scheme proposed in Section VII will enable the definition of the number H_j of static slots to be allocated to each FlexRay node, according to its communication requirements.

Each FlexRay node is composed of a central processing unit (CPU) and a communication controller (CC). The CPU and the CC are interconnect by a controller-host interface (CHI). The CC runs independently of the CPU and implements the FlexRay communication services. The CHI follows the FlexRay specification as described in [5].

The synchronous message streams sets models the message generation at the application level. Each node N_j ($j = 1, 2, ..., n$) supports m_j message streams. The message streams

associated to a node N_j form a set $\mathbb{S}_j = \{S_{1,j}, S_{2,j}, ..., S_{m_j,j}\}$. $\mathbb{S}_j$ is a subset of $\mathbb{S}$. In each CPU a specific middleware will define, at run-time, which message will be transferred in the slots allocated to the node. That is the target of the proposed RT-Middleware described in Section VI.

As described in Section III, the FlexRay static segment is divided in slots of fixed and identical size $gdStaticSlot$, this size being defined during the design phase. As $gdStaticSlot$ has equal length for all static slots, it must be fixed to accommodate the longest message in the system:

$$gdStaticSlot = C_{max} \tag{3}$$

where C_{max} is the size of the longest message.

In this paper, without loss of generality, the message length is considered to be unitary, i.e.:

$$C_i = 1. \tag{4}$$

Consequently

$$gdStaticSlot = 1. \tag{5}$$

Through out this paper, it is assumed that all system parameters, including FC_{bus}, ST_{bus} and any parameter related to the message streams, are given in integer numbers of static slots.

Additionally, it is also assumed that the generation of messages at the application layer and the slot sequence at the FlexRay bus cycle are not synchronized.

Finally, it is also considered that $hp(i,j)$ and $lp(i,j)$ are the set of message streams with priority respectively, higher and lower than stream $S_{i,j}$ of node j:

$$hp(i,j) = \{S_{1,j}, S_{2,j}, ..., S_{i-1,j}\} \tag{6}$$

$$lp(i,j) = \{S_{i+1,j}, S_{i+2,j}, ..., S_{m_j,j}\} \tag{7}$$

VI. RT-Middleware

Traditionally, state-of-the-art research works about the FlexRay static segment assume that message streams are associated to specific static slots and cycle repetitions during the design phase, assuming also that this association does not change during the system execution [7], [10], [11].

In this work, instead of statically assigning message streams to slots, we consider that the static slots are simply allocated to nodes, as defined in the FlexRay Standard [5]. At each node, a specific RT-Middleware will arbitrate in run-time which message will be transmitted within each slot of a FlexRay cycle.

The proposed RT-Middleware is implemented at the ECU of each node. Mainly, the RT-Middleware has a buffer for each message stream in the node. When a message is generated at the application layer, it is placed in the related buffer waiting to be dispatched.

Later, in an instant of time called *freeze instant* f_j^l, being $j = 1, 2, ..., n$ and the index l denoting the $l-th$ freeze instant, the FPS dispatcher will remove H_j waiting messages from the local buffers and move them into the input buffer of the CHI. The freeze instant will be periodically repeated and is synchronized with the FlexRay bus cycle. The freeze instant is defined for each node j during the design phase, and the time interval δ_{f_j} between the node's freeze instant and the start of the first static slot allocated to the node must be large enough to accommodate all the dispatching processes, including the generation of the FlexRay frames to be transmitted. In this work, we consider that there is one freeze instant per node in each FC, always in the same moment. So, the time interval Δ_{f_j} between two consecutive freeze instants is $\Delta_{f_j} = FC_{bus}$. The freeze instant can be regarded as a generalization of the AUTOSAR FlexRay Jobs that were described in Section IV.

In this work, we define that the dispatcher selects the messages to be transmitted according to a Rate-Monotonic priority strategy scheduling (RM): smaller the period of the message stream, higher will be its priority [14].

The basics of the proposed RT-Middleware are depicted in Figure 5. In Figure 5.b is illustrated an example of an FC with the freeze instants f_j^l and $f_j^l + 1$, and the time interval Δ_{f_j} elapsed between them.

The use of this type of RT-Midleware requires the previous allocation of an adequate number (H_j) of static slots to node N_j. In the following section, we discuss an adequate methodology to define the required slot allocation.

VII. Static Slot Allocation Schemes

The static slot allocation scheme plays an important role in guaranteeing the deadlines of synchronous messages. In this section we formally present the definition of slot allocation scheme and discuss its requirements and performance metrics.

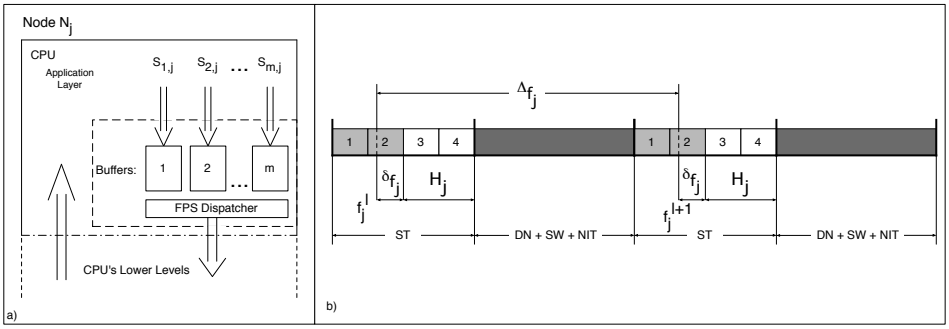

Figure 5. a) RT-Middleware. b) Freeze instants f_j^l and $f_j^l + 1$ and time interval Δ_{f_j}

We define a *static slot allocation scheme* as an algorithm that produces as output the number of static slots that must be allocated for each node in order to guarantee the deadlines of its messages. The input for such algorithm are just the system parameters and the message streams set.

In Section VIII a specific slot allocation scheme will be discussed and analyzed. In the remaining of this section, the requirements that any allocation scheme must satisfy are presented and discussed. Such requirements can be modeled as a *protocol constraint* and a *deadline constraint*, as early suggested in [15], [16] to solve the bandwidth allocation problem in the timed token protocol [17].

A. Protocol Constraint

According to the FlexRay specification, the static segment is composed of a integer number $gNumberOfStaticSlots$ of static slots. As described earlier, we assume that all system parameters are given in integer multiples of static slots and therefore

$$ST_{bus} = gNumberOfStaticSlots. \tag{8}$$

The FlexRay specification also states that at least one static slot must be assigned to each node connected to a FlexRay bus, and that the number of static slots is limited by 1023, which is the maximum $FrameID$ that can be attributed to a static slot.

The work reported in [18] sugests that there is a cyclic dependency between the size of the static and dynamic segments of FlexRay. In this paper it is considered that the size ST_{bus} of the static segment is exactly the size required to accommodate all static slots allocated to

each node, and the remainder of the FC is available for the dynamic and the control segments. Therefore the size of FlexRay static segment is given by:

$$ST_{bus} = \sum_{j=1}^{n} H_j \tag{9}$$

where H_j is the static slot allocation of node j.

Size of the FlexRay cycle: The FlexRay cycle should have a size FC_{bus}, such that all synchronous messages will be able to meet their deadlines. This means that the definition of FC_{bus} must consider the characteristics of the message stream set $\mathbb{S}$, as exemplified below.

Consider a system where $S_{i,j}$ is the message stream i that belongs to station j. $S_{i,j}$ has a period $P_{i,j}$, a deadline $D_{i,j}$ that is equal to its period $P_{i,j}$ and a size $C_{i,j}$ that is unitary, and equal to the slot length. $S_{i,j}$ is associated with the static slot with $FrameID = 1$. The arrival of the kth message generated by $S_{i,j}$ is represented by $M_{i,j}^k$ ($k = 1, 2, ..., +\infty$). Similarly, $D_{i,j}^k$ represents the relative deadline of $M_{i,j}^k$.

Consider that message $M_{i,j}^1$ arrives just after the beginning of its static slot in the first FC, as depicted in Figure 6, where FC_{bus} has been set equal to $P_{i,j}$.

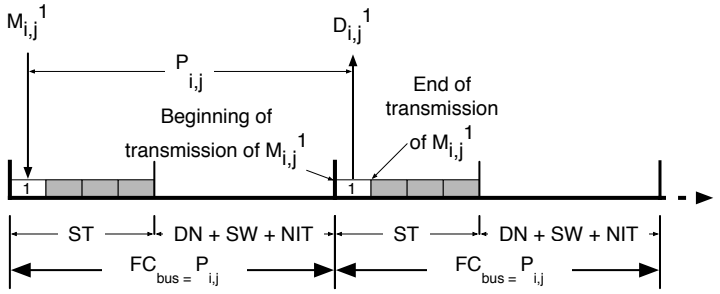

Figure 6. Size of FlexRay Cycle

According to [4], the CC decides if a message generated by $S_{i,j}$ will be send to the bus at the beginning of the slot whose identifier is associated with. If a message is generated immediately after the beginning of the slot, it must wait an entire FC until the beginning of the next slot to be transmitted. In the depicted example, $M_{i,j}^1$ will starts its transmission in the first slot of the second FlexRay cycle. As $M_{i,j}^1$ has a unitary size ($C_{i,j} = 1$), it will miss its deadline because there is no enough time to conclude its transmission before $D_{i,j}^1$.

The above example shows that the period of a message stream $S_{i,j} \in \mathbb{S}$ must be at least longer than the size of FC (in integers of $gdStaticSlot$): $P_{i,j} \geq FC_{bus} + 1$. This leads to a

restriction upon the size of the FlexRay Cycle:

$$FC_{bus} \leq P_{min} - 1 \tag{10}$$

where P_{min} is the smallest period of the message stream set $\mathbb{S}$.

Consider now the freeze instant concept introduced in Section VI. In this case, the decision about the transmission of $M^1_{i,j}$ in the current FC is made at the freeze instant. Consequently, if $M^1_{i,j}$ want to use a static slot allocated to the station j, it must arrive before the freeze instant, incurring in an additional delay that must be considered in $M^1_{i,j}$ response time. Therefore the period of a message stream $S_{i,j} \in \mathbb{S}$ must be at least longer than the size of FC (in integers of $gdStaticSlot$) plus the delay introduced by the freeze instant of its node: $P_{i,j} \geq FC_{bus} + 1 + \delta_{f_j}$.

This leads to the following mandatory restriction upon the size of the FlexRay Cycle:

$$FC_{bus} \leq P_{min} - (1 + \delta_{f_{max}}) \tag{11}$$

where $\delta_{f_{max}}$ is the greatest δ_f of the nodes in the system.

In the study presented in [10], the authors conclude that it is favorable to use the largest possible value for the size of the FC. Indeed, the use of a FC that is larger than the strictly necessary to the synchronous message streams allows the reservation of static slots for future expansions, and also allows some flexibility in the definition of the DN's length.

As described in Section III a FlexRay cycle is composed of a static segment with size ST_{bus}, a dynamic segment with size DN_{bus}, and two control segments: the NIT and the SW. Therefore, the size FC_{bus} is given by

$$FC_{bus} = ST_{bus} + DN_{bus} + \theta \tag{12}$$

where θ represents the size of the two control segments.

According to the FlexRay specification [5], the dynamic segment is not mandatory, which means that we may have

$$DN_{bus} \geq 0. \tag{13}$$

From Equations 9, 12 and 13, it follows that:

$$\begin{aligned} FC_{bus} &= \sum_{j=1}^{n} H_j + DN_{bus} + \theta \\ FC_{bus} &\geq \sum_{j=1}^{n} H_j + \theta. \end{aligned} \tag{14}$$

Now, considering both restrictions upon the size of the FlexRay cycle (Equations 11 and 14), we may define the following equation as the **Protocol Constraint** that must be fulfilled by any static slot allocation scheme H_j:

$$\sum_{j=1}^{n} H_j + \theta \leq FC_{bus} \leq P_{min} - (1 + \delta_{f_{max}}). \tag{15}$$

As a consequence, the protocol constraint imposes both an upper and a lower bound to the length of the FlexRay cycle, in order to guarantee that there is enough time to transfer all the allocated slots and that such time is inline with the required responsiveness of the communication system.

B. Deadline Constraint

A static slot allocation scheme must ensure that the system is always schedulable. In other words, a static slot allocation scheme must ensure that all synchronous message streams will be always able to transfer its messages before their deadlines.

The fixed size of the FlexRay cycle, segments and static slots enables the evaluation of the demand generated by the set of message streams of node N_j. Therefore, it is also possible to determine the required number of static slots that need to be available during a predetermined time interval, in order to adequately transfer all the synchronous messages. A deadline constraint can therefore be expressed as a restriction that must be satisfied in order to ensure that all the generated synchronous messages will have a slot to be transferred before the expiration of their own deadlines.

Defining that $R_{i,j}$ is the worst-case response-time (wcrt) for the transfer of a message generated by $S_{i,j}$ during a time interval $t_0 \in [0, +\infty[$, a deadline constraint can be defined as follows:

$$R_{i,j} \leq D_{i,j}, \ \forall S_{i,j} \in \mathbb{S} \tag{16}$$

In this paper, it is considered that the generated messages will be dispatched to the bus at a predetermined time instant (freeze instant) according to a non-preemptive RM scheme. Thus, the wcrt $R^k_{i,j}$ of a message $M^k_{i,j}$ generated by the stream $S_{i,j}$ can be evaluated through a set of equations for fixed priority systems (FPS) based on those proposed in [19], [20].

First of all, we must recall that, as stated in Section V, all values and system parameters are expressed in integers of static slots, and that both the message length and $gdStaticSlot$ are unitary (Equations 4 and 5).

Without loss of generality, lets consider a FC with H_j static slots allocated to node N_j. The worst scenario for a message $M_{i,j}^k$ generated by message stream $S_{i,j}$ of the node N_j is to be generated immediately after a freeze instant f_j^0, since in this case it will be forced to wait an entire FC until the next freeze instant f_j^1 in order to compete for the static slots of node N_j. This initial delay is denoted by:

$$f_j^1 = f_j^0 + FC_{bus}. \tag{17}$$

In the worst case, messages generated by all streams in $hp(i,j)$ arrived at the same instant of $M_{i,j}^k$. In such case, at the freeze instant f_j^1 the messages with higher priority than $M_{i,j}^k$ have a demand Θ^1 of static slots given by

$$\Theta_{i,j}^1 = \sum_{h \in hp(i,j)} 1. \tag{18}$$

Since in each FC there is a number H_j of static slots allocated to the node N_j, the demand $\Theta_{i,j}^1$ requires a integer number η^1 of FCs given by:

$$\eta_{i,j}^1 = \lfloor \frac{\Theta_{i,j}^1}{H_j} \rfloor \tag{19}$$

and still remains a number $\iota_{i,j}^1$ of messages to be transferred given by:

$$\iota_{i,j}^1 = \Theta_{i,j}^1 - \eta_{i,j}^1 \times H_j \tag{20}$$

The transfer of $M_{i,j}^k$ can then occur from the freeze instant f_j^2 given by $f_k^2 = f_j^1 + \eta_{i,j}^1 \times FC_{bus}$. However, until the freeze instant f_j^2 there may be new messages generated by $hp(i,j)$, and the demand of these new messages must also be considered. The total demand generated by message streams in $hp(i,j)$ up to the freeze instant f_j^2 is then given by:

$$\Theta_{i,j}^2 = \sum_{h \in hp(i,j)} \lceil \frac{f_j^2 - f_j^0}{P_h} \rceil = \sum_{h \in hp(i,j)} \lceil \frac{(\eta_{i,j}^1 + 1) \times FC_{bus}}{P_h} \rceil \tag{21}$$

which requires a integer number of FCs given by

$$\eta_{i,j}^2 = \lfloor \frac{\Theta_{i,j}^2}{H_j} \rfloor \tag{22}$$

remaining a number $\iota_{i,j}^2$ of messages transferred in the last FC:

$$\iota_{i,j}^2 = \Theta_{i,j}^2 - \eta_{i,j}^2 \times H_j \tag{23}$$

It is possible to obtain iteratively the freeze instant f_j^l at which $M_{i,j}^k$ will be finally selected to be transferred through the following set of equations:

$$\Theta_{i,j}^1 = \sum_{h \in hp(i,j)} 1 \tag{24}$$

$$\eta_{i,j}^\omega = \lfloor \frac{\Theta_{i,j}^\omega}{H_j} \rfloor, \omega \geq 1 \tag{25}$$

$$\Theta_{i,j}^\omega = \sum_{h \in hp(i,j)} \lceil \frac{(\eta_{i,j}^{\omega-1} + 1) \times FC_{bus}}{P_h} \rceil, \omega > 1 \tag{26}$$

The iteration is interrupted when $\Theta_{i,j}^\omega = \Theta_{i,j}^{\omega-1}$, or when $\eta_{i,j}^\omega \times FC_{bus} > D_{i,j}^k$ (case where $M_{i,j}^k$ will lose its deadline).

Finally, the worst-case response time $R_{i,j}^k$ for the transmission of $M_{i,j}^k$ messages is given by the sum of the following terms:

- The time interval between the arrival of $M_{i,j}^k$ and the first freeze instant f^0, in the worst case: $\Delta_{f_j} = FC_{bus}$;
- The number of complete FCs until the freeze instant f_j^l in which $M_{i,j}^k$ is selected to be transferred: $\eta_{i,j}^k \times FC_{bus}$;
- The time interval between the freeze instant and the beginning of the allocation of node N_j in the last FC, which is equal to δ_{f_j};
- The number of static slots used for the transmission of higher priority messages in the last FC ($\iota_{i,j}^k = \Theta_{i,j}^k - \eta_{i,j}^k \times H_j$);
- One static slot for the transmission of the $M_{i,j}^k$ message;

which results in a worst-case response time given by:

$$R_{i,j}^k = FC_{bus} + \eta_{i,j}^k \times FC_{bus} + \delta_{f_j} + \iota_{i,j}^k + 1 \tag{27}$$

The problem of non-preemptive fixed priority scheduling of messages is a complex problem that was addressed in [19]. In the scheduling approach considered in this paper, there are however some simplifying assumptions. It is considered that messages duration is unitary and equal to one static slot. Additionally, as there is the freeze instant concept, messages with lower priority than message $M_{i,j}^k$ can not block higher priority messages of the same node and therefore can be ignored for the response time analysis.

At the freeze instant, if $M_{i,j}^k$ has already arrived, it has a priority that is higher than the lower priority messages. If $M_{i,j}^k$ has still not arrived, it will not be transmitted in the current cycle, no matter whether any lower priority message has or not arrived. This means that, in

the context of this work it is possible to completely ignore messages with lower priority than $M^k_{i,j}$, which considerably simplifies the analysis.

To highlight the use of the above deadline constraint formula (Equation 27), please consider the following example. In a given system, a node N_j has three message streams with periods $P_{1,j} = 12$, $P_{2,j} = 15$ and $P_{3,j} = 35$. The FlexRay cycle has a length $FC_{bus} = 10$, where the length of the static segment is $ST_{bus} = 4$. Two static slots are allocated for N_j, i.e., $H_j = 2$, with $FrameID = 1$ and $FrameID = 2$. As the RT-Middleware will dispatch the messages according to a RM ordering, $S_{1,j}$ has the higher priority and $S_{3,j}$ the lower. To simplify the example we define that $\delta_f = 1$.

Figure 7 depicts the behavior of the system. Immediately after the freeze instant f^0_j there is a simultaneous arrival of messages $M^1_{1,j}$, $M^1_{2,j}$ and $M^1_{3,j}$. Since the freeze instant has already occurred, the static slots allocated to the node N_j in the first FC will remain empty. In the freeze instant (f^1_j) the RT-Middleware queue contains messages $M^1_{1,j}$, $M^1_{2,j}$ and $M^1_{3,j}$. Due to their relative priorities, $M^1_{1,j}$ and $M^1_{2,j}$ are selected to be transferred in the H_j static slots of the second FC. Therefore, the response times are respectively $R^1_{1,j} = 12$ and $R^1_{2,j} = 13$. Message $M^1_{3,j}$remains in the queue. Immediately after the time instant 11 $M^2_{1,j}$ arrives, and immediately after time instant 14 $M^2_{2,j}$ arrives. In f^2_j the RT-Middleware queue contains $M^2_{1,j}$, $M^2_{2,j}$ and $M^1_{3,j}$, being selected $M^2_{1,j}$ and $M^2_{2,j}$ to be transferred in the static slots allocated to N_j. Immediately after the time instant 23, $M^3_{1,j}$ arrives. As the new message from $S_{2,j}$ arrives immediately AFTER the freeze instant f^3_j, in f^3_j the queue contains only $M^3_{1,j}$ and $M^1_{3,j}$, and the message $M^1_{3,j}$ is finally selected to be transmitted with $R^1_{3,j} = 33$.

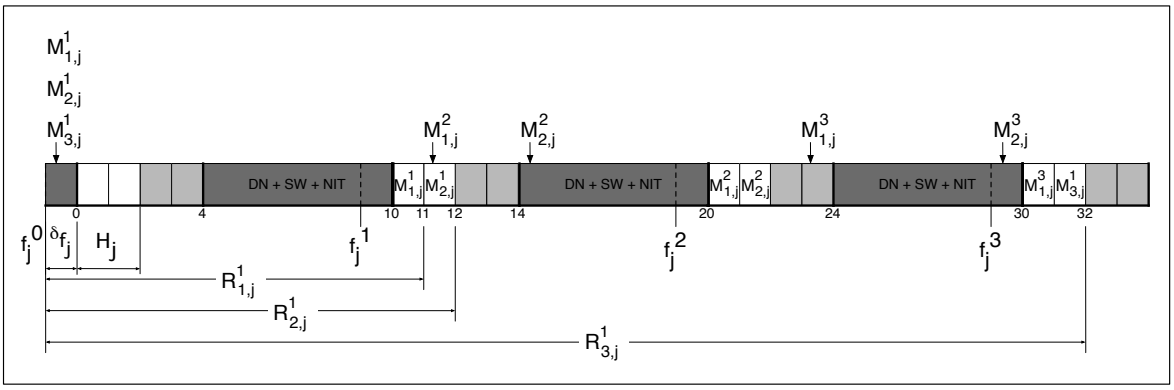

Figure 7. Example of Deadline Constraint

The values for $R^1_{1,j}$, $R^1_{2,j}$ and $R^1_{3,j}$ can be confirmed by the deadline constraint (Equation

16), as follows.

As $hp(1,j)$ is an empty set, we have for $S_{1,j}$: $\Theta^k_{1,j} = 0$, $\eta^k_{1,j} = 0$ and $\iota^k_{1,j} = 0$. From Equation 27 we have that:

$$R^k_{1,j} = 10 + 0 + 1 + 0 + 1 = 12. \quad (28)$$

As $R^k_{1,j} = 12 \leq D_{1,j}$, the deadline of $S_{1,j}$ is respected.

For $S_{2,j}$ we have:

$$\begin{aligned}
\Theta^1_{2,j} &= \sum_{h \in hp(2,j)} 1 = 1 \\
\Theta^2_{2,j} &= \lceil \frac{(\lfloor \frac{1}{2} \rfloor + 1) \times 10}{12} \rceil = 1 \\
\Theta^3_{2,j} &= \lceil \frac{(\lfloor \frac{1}{2} \rfloor + 1) \times 10}{12} \rceil = 1 \\
\Theta^k_{2,j} &= \Theta^3_{2,j} = \Theta^2_{2,j} = 1 \\
\eta^k_{2,j} &= \lfloor \frac{1}{2} \rfloor = 0 \\
\iota^k_{2,j} &= \Theta^k_{2,j} - \eta^k_{2,j} \times H_j = 1 - 0 = 1 \\
R^k_{2,j} &= 10 + 0 + 1 + 1 + 1 = 13 \quad (29)
\end{aligned}$$

Again, the deadline of $S_{2,j}$ is respected ($R^k_{2,j} = 13 \leq D_{2,j}$).

Finally, for $S_{3,j}$ we have:

$$\begin{aligned}
\Theta^1_{3,j} &= \sum_{h \in hp(3,j)} 1 = 2 \\
\Theta^2_{3,j} &= \lceil \frac{(\lfloor \frac{1}{2} \rfloor + 1) \times 10}{12} \rceil + \lceil \frac{(\lfloor \frac{1}{2} \rfloor + 1) \times 10}{15} \rceil = 2 \\
\Theta^3_{3,j} &= \lceil \frac{(\lfloor \frac{2}{2} \rfloor + 1) \times 10}{12} \rceil + \lceil \frac{(\lfloor \frac{2}{2} \rfloor + 1) \times 10}{15} \rceil = 4 \\
\Theta^4_{3,j} &= \lceil \frac{(\lfloor \frac{4}{2} \rfloor + 1) \times 10}{12} \rceil + \lceil \frac{(\lfloor \frac{4}{2} \rfloor + 1) \times 10}{15} \rceil = 5 \\
\Theta^5_{3,j} &= \lceil \frac{(\lfloor \frac{4}{2} \rfloor + 1) \times 10}{12} \rceil + \lceil \frac{(\lfloor \frac{4}{2} \rfloor + 1) \times 10}{15} \rceil = 5 \\
\Theta^k_{3,j} &= \Theta^4_{2,j} = \Theta^5_{2,j} = 5 \\
\eta^k_{3,j} &= \lfloor \frac{5}{2} \rfloor = 2 \\
\iota^k_{3,j} &= \Theta^k_{3,j} - \eta^k_{3,j} \times H_j = 5 - 4 = 1 \\
R^k_{3,j} &= 10 + (2 \times 10) + 1 + 1 + 1 = 33 \quad (30)
\end{aligned}$$

The deadline of $S_{3,j}$ is also respected, which means that the deadline constraint is respected.

This result is in-line with the values that can be observed from Figure 7.

VIII. Proportional Allocation Scheme

Finally, there is the need to define the static slot allocation H_j for each node N_j. In this section, we propose the use of an adequate heuristic: proportional allocation scheme, to derive a static slot allocation H_j that fulfills the requirements imposed by both the protocol constraint (Equation 15) and the deadline constraint (Equation 16).

In the proposed Proportional Allocation Scheme, the number H_j of static slots allocated to a node N_j is given by:

$$H_j = \lceil \sum_{i=1}^{m_j} \frac{FC_{bus}}{P_{i,j}} \rceil \tag{31}$$

The rationale behind the proposed heuristic is the following: consider that a message from stream $S_{i,j}$ is fragmented and the resulting fragments are scattered among the static slots during the period $P_{i,j}$ (one fragment per FlexRay cycle FC_{bus}). $\frac{FC_{bus}}{P_{i,j}}$ would represent the percentage of slot utilization during each FlexRay cycle. Consider now the set $M_j(i = 1..m_j)$ of message streams assigned to node N_j. The equation $\lceil \sum_{i=1}^{m_j} \frac{FC_{bus}}{P_{i,j}} \rceil$ represents the number of slots required to serve all the message streams assigned to node N_j. Therefore, it represents the number of static slots that must be allocated to node N_j in order to guarantee the fulfillment of all message deadlines.

To define the static segment of a FlexRay system we can then use Algorithm 1:

Algorithm 1: Algorithm for the Proportional Allocation Scheme

1 - Define a value for FC_{bus} with the longest possible duration (Equation 10);

2 - **for** *j:= 1 to n* **do**
| Evaluate the static slot allocation H_j (Equation 31)
end

3 - Check if the Protocol Constraint (Equation 15) is respected;

4 - **for** *j:= 1 to n* **do**
| Check if the Deadline Constraint for node N_j (Equation 16) is respected
end

IX. Example: Use of the Proportional Allocation Scheme

This section presents an example that highlights the use of the proposed static slot allocation scheme.

Consider a FlexRay system composed by 3 nodes with $\delta_{f_{max}} = 1$. Each node has a set of message streams whose periods are presented in Table I, and the size of the control segments is $\theta = 1$. It is worth noting that all values are given in integers of static slots.

Node N_1	$P_{1,1}$	$P_{2,1}$	$P_{3,1}$	$P_{4,1}$	
	12	15	29	50	
Node N_2	$P_{1,2}$	$P_{2,2}$	$P_{3,2}$		
	23	33	100		
Node N_3	$P_{1,3}$	$P_{2,3}$	$P_{3,3}$	$P_{4,3}$	$P_{5,3}$
	12	23	29	37	44

Table I

Set M of message streams

As stated in Section VIII, Algorithm 1 must be followed to define the static segment of a FlexRay system. This leads to the following steps:

1) Set the size of FC_{bus} with the longest possible duration.

As stated in Section VII-A, it is favorable to use the largest possible value for the length of the FC, as long as FC_{bus} respect the restriction imposed by Equation 10:

$$\begin{aligned} FC_{bus} &\leq P_{min} - 1 \\ P_{min} &= 11 \\ FC_{bus} &= 10 \end{aligned} \tag{32}$$

2) For all nodes in the system, calculate the static slot allocation H_j

The second step of the algorithm uses the Proportional Allocation Scheme (Equation 31) to calculate the allocation of each node. The calculations are shown below.

For the node N_1, the allocation H_1 is:

$$H_1 = \lceil \frac{10}{12} + \frac{10}{15} + \frac{10}{29} + \frac{10}{50} \rceil = 3 \tag{33}$$

For the node N_2, the allocation H_2 is:

$$H_2 = \lceil \frac{10}{23} + \frac{10}{33} + \frac{10}{100} \rceil = 1 \tag{34}$$

Finally, for the node N_3, the allocation H_3 is:

$$H_3 \quad = \quad \lceil \frac{10}{12} + \frac{10}{23} + \frac{10}{29} + \frac{10}{37} + \frac{10}{44} \rceil = 3 \tag{35}$$

Table II summarizes the allocation for each node.

H_1	3
H_2	1
H_3	3

Table II
NODE ALLOCATION

3) To check if the Protocol Constraint is respected.

As explained in Section VII, any allocation scheme must satisfy the Protocol Constraint. As now we have the allocation H_j of each node N_j, the Protocol Constraint can be verified with the use of Equation 15:

$$\begin{aligned} \sum_{j=1}^{n} H_j + \theta \quad & \leq FC_{bus} \quad \leq P_{min} - (1 + \delta_{f_{max}}) \\ 3 + 1 + 3 + 1 \quad & \leq 10 \quad \leq 12 - 1 \\ 8 \quad & \leq 10 \quad \leq 11 \end{aligned} \tag{36}$$

And therefore the Protocol Constraint is respected for the FlexRay system.

4) For all nodes in the system, check if the Deadline Constraint is respected.

We use the information of the allocation to verify if the Deadline Constraint is respected. A sketch of the calculations is presented bellow.

For node N_1:

$$\begin{aligned} R^k_{1,1} \quad &= \quad 12 \leq D_{1,1} \\ R^k_{2,1} \quad &= \quad 13 \leq D_{2,1} \\ R^k_{3,1} \quad &= \quad 14 \leq D_{3,1} \\ R^k_{4,1} \quad &= \quad 24 \leq D_{4,1} \end{aligned} \tag{37}$$

Since the deadlines are respected for all message streams, the Deadline Constraint is respected for node N_1.

For node N_2:

$$\begin{aligned} R^k_{1,2} &= 12 \leq D_{1,2} \\ R^k_{2,2} &= 22 \leq D_{2,2} \\ R^k_{3,2} &= 62 \leq D_{3,2} \end{aligned} \tag{38}$$

Since the deadlines are respected for all message streams, the Deadline Constraint is respected for node N_2.

Finally, for node N_2:

$$\begin{aligned} R^k_{1,3} &= 12 \leq D_{1,3} \\ R^k_{2,3} &= 13 \leq D_{2,3} \\ R^k_{3,3} &= 14 \leq D_{3,3} \\ R^k_{4,3} &= 23 \leq D_{4,3} \\ R^k_{5,3} &= 24 \leq D_{5,3} \end{aligned} \tag{39}$$

Since the deadlines are respected for all message streams, the Deadline Constraint is also respected for node N_3.

Table III summarizes the worst-case response time for each message stream in the system, the deadline of each message stream and if the Deadline Constraint is respected.

As both the Protocol and Deadline Constraints are respected, the Proportional Allocation Scheme has generated a schedulable allocation of static slots for the above example.

This example clearly highlights the use of the proposed static slot allocation scheme, together with the protocol and the deadline constraints, to generate a schedulable solution for the FlexRay system.

Node N_1	$R_{i,1}$	$D_{i,1}$	$R_{i,1} \leq D_{i,1}$
	$R_{1,1} = 12$	$R_{1,1} = 12$	$R_{1,1} \leq D_{1,1}$ ok
	$R_{2,1} = 13$	$R_{2,1} = 15$	$R_{2,1} \leq D_{2,1}$ ok
	$R_{3,1} = 14$	$R_{3,1} = 29$	$R_{3,1} \leq D_{3,1}$ ok
	$R_{4,1} = 24$	$R_{4,1} = 50$	$R_{4,1} \leq D_{4,1}$ ok
Node N_2	$R_{i,2}$	$D_{i,2}$	$R_{i,2} \leq D_{i,2}$
	$R_{1,2} = 12$	$R_{1,2} = 23$	$R_{1,2} \leq D_{1,2}$ ok
	$R_{2,2} = 22$	$R_{2,2} = 23$	$R_{2,2} \leq D_{2,2}$ ok
	$R_{3,2} = 62$	$R_{3,2} = 100$	$R_{3,2} \leq D_{3,2}$ ok
Node N_2	$R_{i,3}$	$D_{i,3}$	$R_{i,3} \leq D_{i,3}$
	$R_{1,3} = 12$	$R_{1,3} = 23$	$R_{1,3} \leq D_{1,3}$ ok
	$R_{2,3} = 22$	$R_{2,3} = 23$	$R_{2,3} \leq D_{2,3}$ ok
	$R_{3,3} = 62$	$R_{3,3} = 100$	$R_{3,3} \leq D_{3,3}$ ok
	$R_{4,2} = 22$	$R_{4,3} = 23$	$R_{4,3} \leq D_{4,3}$ ok
	$R_{5,3} = 62$	$R_{5,3} = 100$	$R_{5,3} \leq D_{5,3}$ ok

Table III

SUMMARY OF DEADLINE CONSTRAINT IN EXAMPLE 1

X. INTEGRATION OF THE PROPOSED METHODOLOGY WITH AUTOSAR

As discussed in Section IV, AUTOSAR imposes several restrictions to the design of a FlexRay system, forcing the association between message streams, static slots and cycle repetition that can not be changed during run-time. The use of the techniques proposed in this paper within the AUTOSAR environment imposes the flexibility of considering systems where there may be no synchronization between tasks, signals and the FlexRay cycle, or where the periods of the message streams may not be integer multiples of the FC.

The RT-Middleware described in this paper defines in run-time which synchronous message will be transmitted within each static slot associated to a node N_j. However, as described in Section IV, in the AUTOSAR the association between a message stream and a static slot is defined during the design phase and cannot be changed during the execution of the system. This constraint impairs the implementation of the RT-Middleware below the AUTOSAR application layer. However, it can be easily implemented as an SW-C in the AUTOSAR application layer, as described in this section.

The real-time middleware that will be integrated within AUTOSAR (called here RTM-A) is the RT-Middleware presented in Section VI. In the RT-Middlware, messages associated

with the static segment that are generated at node N_j are placed in buffers waiting to be dispatched. In a freeze instant that is synchronized with the an AUTOSAR FlexRay Job, a dispatcher moves the messages from the buffers to a lower layer according to a predefined fixed-priority scheduling (FPS) policy.

Due to the constraints imposed by AUTOSAR related to the static association between PDUs and FlexRay frames, the sender node has to pack a message into an special $ISignalIPDU$ before send it to the RTE layer. This pack is built by a software component called $ISignalIPDU$ Builder. The special $ISignalIPDU$ will be later unpacked in the receiver node and send to the target application by another software component called $ISignalIPDU$ Filter. The proposed RTM-A architecture is depicted in Figure 8.

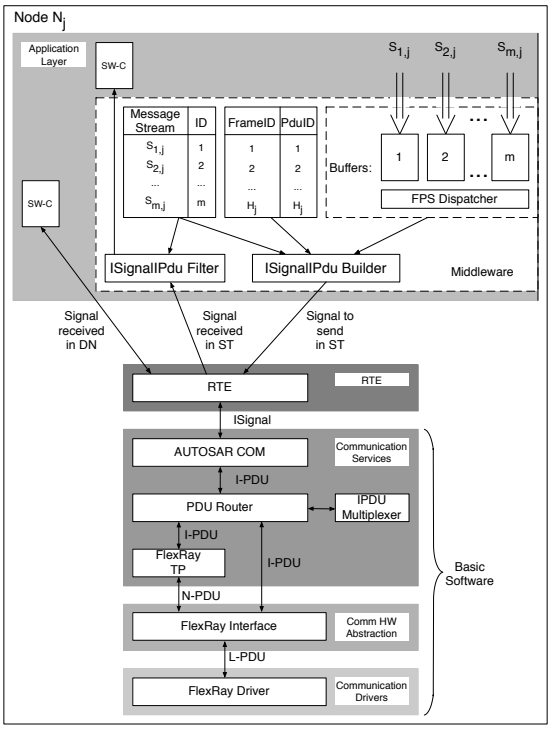

Figure 8. RTM-A architecture

The PCI of the $ISignalIPDU$ in which a message will be send contains the required information to route the $ISignalIPDU$ to the desired static slot. In order to assemble the $ISignalIPDU$ with the correct PCI, the $ISignalIPDU$ Builder consults an internal table which indicates the current static slot, the allocation of the node and the PCI which represents the desired static slot.

The SDU of the $ISignalIPDU$ is composed by a) an ID segment containg an identifier

that allows the $ISignalIPDU$ Filter to send the message to the target application and b) the message itself. The $ISignalIPDU$ generated by the $ISignalIPDU$ Builder is depicted in Figure 9.

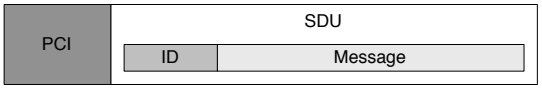

Figure 9. $ISignalIPDU$ generated by the $ISignalIPDU$ Builder

Using this RTM-A approach, it becomes clear how the proposed static slot allocation scheme can be considered within an AUTOSAR implementation.

XI. Conclusions

The FlexRay Communications System is a protocol for data communication developed by an alliance of manufacturers including BMW, DaimlerChrysler and Bosch. It is being considered the future *de facto* standard for automotive applications that require performance and reliability, as future X-by-Wire systems. Through a combination of static and dynamic methods FlexRay provides, among other things, flexibility, bandwidth and determinism.

State-of-the-art research works addressing the design of the FlexRay static segment are usually based on the use of high computational requirements techniques, as ILP or NIP. It is usually also assumed that message streams are statically associated with exact and unique static slots, despite the fact that the FlexRay specification just defines that a static slot must be associated with an unique node in the network.

In this work we asses the feasibility of defining the slot allocation for each node using traditional response time analysis (RTA) techniques, and thus considering the timing requirements imposed by the set of message streams allocated to each node. The technique proposed in this paper is able to deal with message stream sets where periods are not multiple of the FlexRay cycle duration, nor the messages arrival is synchronized with the FlexRay cycle. This work shows how the proposed technique can be integrated as a RT-Middleware in each node, to take full advantage of the proposed allocation scheme. We also show how the proposed allocation scheme can be integrated as an RT-Middleware within an AUTOSAR environment, allowing real FlexRay implementations to take full advantage of the proposed approach.

As future work we intend to derive the maximum bus utilization that can be achieved when using the proportional allocation scheme. We also intend to improve the proposed scheme in order to obtain higher bus utilizations.

ACKNOWLEDGES

This work was partially funded by a CAPES/FCT grant (Project CAPES/FCT 285/2010), by Faculdade de Engenharia da Universidade do Porto (FEUP) and by Department of Automation and Systems Engineering of the Federal University of Santa Catarina (DAS/UFSC).

LIST OF SYMBOLS, PARAMETERS AND VARIABLES

The list of symbols, parameters and variables are summarized in Table IV. When appropriated, domains are listed in the table.

REFERENCES

[1] N. Navet and F. Simonot-Lion, *Automotive Embedded Systems Handbook*. CRC, 2008.

[2] R. Davis, A. Burns, R. Bril, and J. Lukkien, "Controller area network (can) schedulability analysis: Refuted, revisited and revised," *Real-Time Systems*, vol. 35, no. 3, pp. 239–272, 2007.

[3] R. Zurawski, *The Industrial Communication Technology Handbook*. CRC, 2005.

[4] T. Pop, P. Pop, P. Eles, Z. Peng, and A. Andrei, "Timing Analysis of the FlexRay Communication Protocol," *Real-Time Systems*, vol. 39, no. 1, pp. 205–235, 2008.

[5] FlexRay, "FlexRay Communications System Protocol Specification Version 2.1," 2005.

[6] H. Zeng, M. Di Natale, A. Ghosal, and A. Sangiovanni-Vincentelli, "Schedule Optimization of Time-Triggered Systems Communicating Over the FlexRay Static Segment," *Industrial Informatics, IEEE Transactions on*, no. 99, pp. 1–1, 2011.

[7] M. Grenier, L. Havet, and N. Navet, "Configuring the Communication on FlexRay: The Case of the Static Segment," *Proceedings of ERTS*, 2008.

[8] S. Ding, H. Tomiyama, and H. Takada, "An Effective GA-Based Scheduling Algorithm for FlexRay Systems," *IEICE Transactions on Information and Systems*, vol. 91, no. 8, pp. 2115–2123, 2008.

[9] M. Lukasiewycz, M. Glaß, J. Teich, and P. Milbredt, "Flexray Schedule Optimization of the Static Segment." CODES+ISSS, 2009.

[10] K. Schmidt and E. Schmidt, "Message Scheduling for the FlexRay Protocol: The Static Segment," *Vehicular Technology, IEEE Transactions on*, vol. 58, no. 5, pp. 2170–2179, 2009.

[11] ——, "Optimal Message Scheduling for the Static Segment of FlexRay," *IEEE Vehicular Technology Conference*, 2010.

[12] AUTOSAR, GbR, "AUTOSAR Specification V. 4.0.0," 2010, available in http://www.autosar.org. Last access in 25/11/2011.

[13] ——, "Specification of FlexRay Interface V3.2.0 R4.0 Rev 2," 2010, available in http://www.autosar.org. Last access in 25/11/2011.

[14] A. Burns and A. Wellings, *Real-Time Systems and Programming Languages: Ada, Real-Time Java and C/Real-Time POSIX*. Addison-Wesley Educational Publishers Inc, 2009.

[15] G. Agrawal, B. Chen, W. Zhao, and S. Davari, "Guaranteeing Synchronous Message Deadlines With the Timed Token Medium Access Control Protocol," *IEEE Transactions on Computers*, 1994.

[16] N. Malcolm and W. Zhao, "The Timed-Token Protocol for Real-Time Communications," *COMPUTER,*, pp. 35–41, 1994.

[17] Grow, R.M., "Timed Token Protocol for Local Area Networks," in *Proc. Electro/82, Token Access Protocols*, 1984.

[18] E. Schmidt, M. Alkan, K. Schmidt, E. Yuruklu, and U. Karakaya, "Performance Evaluation of FlexRay/CAN Networks Interconnected by a Gateway," in *Industrial Embedded Systems (SIES), 2010 International Symposium on*. IEEE, 2010, pp. 209–212.

[19] B. Andersson and E. Tovar, "The utilization bound of non-preemptive rate-monotonic scheduling in controller area networks is 25%," in *Industrial Embedded Systems, 2009. SIES'09. IEEE International Symposium on*. IEEE, 2009, pp. 11–18.

[20] C. L. Liu and J. W. Layland, "Scheduling Algorithms for Multiprogramming in a Hard-Real-Time Environment," *J. ACM*, vol. 20, pp. 46–61, January 1973.

Parameter	Description
δ_{f_j}	Time interval between the beginning of the freeze instant first slot allocated to node N_j
Δ_{f_j}	Time interval between two consecutive freeze instants of node N_j
$\eta_{i,j}$	Integer number of FCs for the demand Θ_{f_j}
$\iota_{i,j}$	Number of static slots required by the demand Θ_{f_j} in the last FC
ω	Index for iteration
Θ	Demand of static slots during a time interval
DN	FlexRay Dynamic Segment
DN_{bus}	Length of DN
FC	FlexRay Cycle
FC_{bus}	Length of FC
$gNumberOfStaticSlots$	Number of Static Slots
$gdStaticSlot$	Length of the Static Slots
H_j	Number of Static Slots allocated to node N_j
$hp(i,j)$	Set of message streams with priority higher than $S_{i_j,j}$
ILP	Integer Linear Programming
h	Index for $hp(i,j)$
i	Index of a message stream $S \in M$, $i = [1, 2, ..., m]$
j	Index of a node N, $j = [1, 2, ..., n]$
k	Index for the $k-th$ message generated by a message stream
$M_{i,j}$	Message generated by $S_{i,j}$
MAC	Media Access Control
MILP	Mixed Integer Linear Programming
m	Number of synchronous message streams in the system
m_j	Number of synchronous message streams in a node N_j
n	Number of nodes in the system
N_j	Node $N$with index j
NIP	Non-linear Integer Programming
NIT	Network Idle Time
NIT_{bus}	Length do NIT
P_{min}	Smallest period among the periods of messages in $\mathbb{S}$
RTA	Real-Time Analysis
SW	Symbolic Window
SW_{bus}	Length of SW
S_i	Message stream with index i
$S_{i,j}$	Message Stream i of node N_j
ST	FlexRay Static Segment
ST_{bus}	Length of ST
$\mathbb{S}$	Set of synchronous message streams
$\mathbb{S}_j$	Set of synchronous message streams in node N_j
TDMA	Time-Division Multiple Access

Table IV

LIST OF SYMBOLS, PARAMETERS AND VARIABLES