

Automated Configuration of TDMA-Based and Event-Triggered Vehicle-Networks with Respect to Real-Time Constraints

Stephan Brummund, Michael Bauer and Uwe Kiencke
Institut für Industrielle Informationstechnik, Universität Karlsruhe (TH)

Copyright © 2008 SAE International

ABSTRACT

Today's vehicle networks are mainly based on the event-triggered CAN-bus. In future FlexRay, which is a TDMA-based bus, will more and more be used for the implementation of safety-relevant real-time systems due to its determinism. In order to configure a CAN-based network the priorities of the messages sending via the external bus have to be defined. In this paper an approach will be presented allowing automated priority determination. Subsequently it will be shown how to adapt this method to automated cycle configuration in case of a FlexRay-based system. In order to ensure determinism not only in TDMA-based but also in event-triggered networks, a method will be presented adapting priorities of messages intending to exceed their deadline. This can be easily realized without changing the CAN protocol.

INTRODUCTION

This paper focuses on automated configuration of event-triggered as well as TDMA-based networks. As a precondition messages sent via the external bus have to be set in a preceding step realized by partitioning the software model to be allocated onto the available hardware. In [1] a partitioning method called CASA (Correlate and Soak Allocation) is presented partitioning a system with respect to a reduced bus-load without any expertises. In order to follow this paper and get knowledge about the complete tool-chain for automated network configuration, a short summary of CASA will be given in the following paragraph "Partitioning Approaches". Although it is shown that CASA leads to solutions with less bus-load than commonly used algorithms as e.g. Kernighan-Lin or Simulated-Annealing, the increasing electronically controlled functionality in cars as engine, transmission and dynamic control and the rising demand for computational

resources make it difficult to fulfill real-time constraints in order to implement safety-relevant applications satisfying quality aspects as safety, availability and reliability.

In event-triggered networks, it is hard to guarantee real-time capability. Due to this fact a method for automated priority determination based on the deadlines of the messages sent via the bus will be presented. This allows to ensure real-time behavior in a CAN network in case of freedom from error. In order to fulfill real-time constraints even in case of e.g. babbling idiots blocking the bus, it will be shown how to adapt priorities of messages intending to exceed their deadlines without changing the CAN protocol.

TDMA-based networks as FlexRay were developed to ensure a deterministic behavior of the system during runtime. In case of FlexRay functional models have to be assigned to certain time slots in the static segment or minislots in the dynamic segment forming a so-called FlexRay cycle (apart from the Symbol Window and the Network Idle Time [2]). Configuring a FlexRay cycle thus implies expertises. Our approach presented is capable to set up a FlexRay cycle in an analytical way. Using this method, demanded quality aspects as safety, availability and reliability are satisfied. Therefore FlexRay is adequate for safety-relevant applications as e.g. X-by-wire.

Finally our approach will be validated using an example system presented in the paragraph "Vehicle Model". After partitioning stage, the priorities of the messages sent via the bus will be determined. Afterwards CAN as well as FlexRay will be set up and the system configuration will be simulated on a CAN as well as a FlexRay model developed in Matlab/Simulink. By means of plotting latency times and the signals sent via the bus, real-time behavior will be verified.

PARTITIONING APPROACHES

Partitioning software models implies a standardization of hardware and software as well, allowing to allocate software onto available hardware without considering e.g. memory and bus access. Only the functional software part will be assigned whereas data management is separated and realized by so-called basic software. Such a standard also provides advantages as transferability of software modules within the network, integration of software modules of different suppliers and updates over vehicle lifetime. Therefore the AUTOSAR (AUTomotive Open System ARchitecture) standard was introduced by a partnership of automotive manufacturers and suppliers in order to develop an industry standard for automotive electronic systems. This partnership consists of three different kinds of membership: core partners, premium members and associates.

An AUTOSAR application consists of software components connected via the Virtual Functional Bus (VFB) as standardized communication service. Due to ECU (Electronic Control Unit) descriptions (e.g. resources as peripheral devices, memory and processor) the software components are mapped onto the available hardware. The resulting system configuration description contains the partitioning as well as a complete communication matrix. Software components mapped to a specific ECU are integrated in the application layer. Therefore an independence from the hardware as well as the location of the other components in the system is achieved. Hereby the Runtime-Environment (RTE) decouples applications from the ECU software, the basic software (BSW) providing the infrastructural functionalities. Moreover the RTE manages the communication between different software modules of the application layer. Due to the communication matrix obtained after mapping process the RTE enables inter- as well as intra-communication. Thus modules allocated to different ECUs can communicate as well as modules placed on the same without adapting their code.

Considering mapping stage an optimized partitioning method is needed in order to take hardware constraints as peripheral devices, memory and processor into account. Furthermore an optimized mapping must reduce inter-communication between different ECUs in order to keep the bus-load on a low level. Therefore CASA (Correlate and Soak Allocation) is introduced in [1], allocating standardized software on standardized hardware. Commonly used partitioning algorithms such as e.g. Kernighan-Lin or Simulated-Annealing are based on several bi-partitionings. The decisive disadvantage of such algorithms is the necessity to define the starting partitioning at the beginning. This requires expertises about the system. Computing bi-partitioning algorithms, in each iteration two clusters will be chosen to be optimized concerning their external costs. However, this optimization is exclusively calculated for the two actual clusters. One of the following iterations can therefore worsen previously found bi-partitionings. Thus they will

either stuck to local minima or have to be abandoned after a certain number of iterations.

CASA is a real multi-partitioning algorithm correlating the communication vectors of all functional objects. Due to the calculated correlation values and the hardware description the software modules can be assigned onto the available ECUs. A high correlation of the communication behavior of two software modules indicates that they have to be clustered. In [1] it is shown that this approach always leads to a partitioning solution having less bus-load than commonly used algorithms.

CONFIGURATION OF CAN

After partitioning stage development engineers face the problem of defining the priorities of messages sent via the external bus. Considering real-time constraints and the implementation of safety-relevant applications, configuration becomes quite difficult and presumes expertises. Thus a method will be presented realizing automated priority determination.

AUTOMATED PRIORITY DETERMINATION

The approach described in this paragraph is based on the Max-Plus Algebra ([3, 4]). The Max-Plus Algebra is an algebra defined on the amount of the real numbers including $-\infty$ analyzing task graphs. In table 1, the defined operations for scalars as well as for matrices are depicted. According to the Max-Plus Algebra, the so-called

Scalars	Matrices
$a \oplus b \hat{=} \max(a, b)$	$\underline{A} \oplus \underline{B} = a_{ij} \oplus b_{ij}$
$a \otimes b \hat{=} a + b$	$\underline{A} \otimes \underline{B} = \bigoplus_{l=1}^m (a_{il} \otimes b_{lj})$
	$\underline{A} \odot \underline{B} = a_{ij} \otimes b_{ij}$
	$\underline{A}^* = \underline{I} \oplus \underline{A} \oplus \dots \oplus \underline{A}^n$
$e \hat{=} 0$	$\varepsilon \hat{=} -\infty$

Table 1: Max-Plus Algebra

Min-Plus Algebra can be derived (table 2). In order to

Scalars	Matrices
$a \oplus' b \hat{=} \min(a, b)$	$\underline{A} \oplus' \underline{B} = a_{ij} \oplus' b_{ij}$
$a \otimes' b \hat{=} a + b$	$\underline{A} \otimes' \underline{B} = \bigoplus'_{l=1}^m (a_{il} \otimes' b_{lj})$
	$\underline{A} \odot' \underline{B} = a_{ij} \otimes' b_{ij}$
	$\underline{A}'^* = \underline{I} \oplus' \underline{A} \oplus' \dots \oplus' \underline{A}^n$
$e' \hat{=} 0$	$\top \hat{=} +\infty$

Table 2: Min-Plus Algebra

exemplify automated priority determination, the taskgraph in figure 1 will be used. Hereby the tasks $\tau_1 \dots \tau_4$ are tasks sending or receiving messages via the external bus of the already partitioned software model. Each task τ_i has a certain calculation time c_i . Moreover the transmission time of the messages is set identical to c_i .

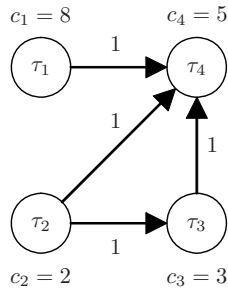


Figure 1: Example: Taskgraph

order to simplify the example. This taskgraph representing tasks with inter-communication (ECU → ECU via an external bus) has two starting nodes able to communicate immediately after initialization. Therefore the vector $\underline{s}_{s,min}$ of equation (1) representing the minimal starting times is set to

$$\underline{s}_{s,min} = (e \quad e \quad \varepsilon \quad \varepsilon)^T. \quad (1)$$

Subsequently their response times can be determined by

$$\underline{r}_{s,min} = \underline{s}_{s,min} \otimes \underline{c} = (8 \quad 2 \quad \varepsilon \quad \varepsilon)^T. \quad (2)$$

Moreover the adjacency matrix considering the transmission times t_i has to be calculated.

$$\begin{aligned} \underline{A} &= \underline{C} \otimes \underline{T} \\ &= \begin{pmatrix} 8 & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & 2 & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & 3 & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & 5 \end{pmatrix} \otimes \begin{pmatrix} \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & 1 & \varepsilon & \varepsilon \\ 1 & 1 & 1 & \varepsilon \end{pmatrix} \\ &= \begin{pmatrix} \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & 4 & \varepsilon & \varepsilon \\ 6 & 6 & 6 & \varepsilon \end{pmatrix} \end{aligned} \quad (3)$$

Due to the fact that task τ_4 is a task only receiving messages, its latest possible response time has to be defined in order to calculate the response times of all tasks within the network and therefore to fulfill real-time constraints of the software model. In this example it will be set to $r_{e,max,4} = 20$ leading to

$$\begin{aligned} \underline{r}_{max} &= (-\underline{A}^*)^T \otimes' \underline{r}_{e,max} \\ &= \begin{pmatrix} e' & \top & \top & -6 \\ \top & e' & -4 & -10 \\ \top & \top & e' & -6 \\ \top & \top & \top & e' \end{pmatrix} \otimes \begin{pmatrix} \top \\ \top \\ \top \\ 20 \end{pmatrix} \\ &= (14 \quad 10 \quad 14 \quad 20)^T. \end{aligned} \quad (4)$$

Equation (4) reveals the sending order of the four tasks τ_i is $\tau_2, \tau_1, \tau_3, \tau_4$ to keep the pre-defined deadline of the taskgraph. In case of an event-triggered network as CAN the priorities p_i therefore have to be defined in the following order: $p_2 > p_1 > p_3 > p_4$.

Remark: The above example is very simplified and just to exemplify priority determination. In real applications, a taskgraph representing a software model

consists also of tasks with intra-communication, i.e. tasks that communicate on the same ECU. The difference between inter- and intra-communication can be taken into account via the communication matrix $\underline{T}$ of equation (3) representing the transmission times of data. Information about messages sent on the same ECU and messages sent via the external bus respectively is given by the partitioning solution of the used partitioning algorithm. As result, vector $\underline{r}_{max}$ also specifies the deadline of tasks with intra-communication. In order to set the priorities for the CAN network, these tasks can be neglected. Only the computational order of tasks communicating via the external bus is relevant for priority determination.

CAPABILITY FOR SAFETY-RELEVANT APPLICATIONS

In the previous paragraph it was exemplified how to determine priorities of messages within a taskgraph. However the communication of each task τ_i within such a taskgraph is considered to be cyclic. In real applications stochastic signals are also existent. Moreover some tasks represent safety-relevant applications that need to be prioritized. The following two paragraphs will show how to set priorities for stochastic as well as for safety-relevant tasks.

Stochastic tasks

Stochastic tasks cannot be integrated within the cyclic taskgraph when analyzing the system for priorities using the Max-Plus Algebra as shown above. However their priorities can be derived on basis of the priorities of the cyclic tasks. In figure 2 the example system of figure 1 is expanded by the stochastic tasks $\tau_{s,1}$ and $\tau_{s,2}$. Stochastic

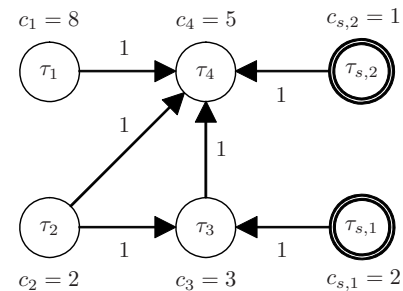


Figure 2: Example: Taskgraph with stochastic tasks $\tau_{s,j}$

tasks represent software to be calculated in case of e.g. high sensor values indicating an event. These tasks can be safety-relevant as well as non safety-relevant. The next paragraph will deal with safety-relevance. Priorities of non safety-relevant stochastic tasks will be set according to their cyclic receiving tasks. In case of the taskgraph of figure 2 the cyclic receiving task of the stochastic one $\tau_{s,1}$ is task τ_3 , that of $\tau_{s,2}$ is task τ_4 . The order of priority of these two cyclic receiving tasks was set to $p_3 > p_4$. Therefore $p_{s,1} > p_{s,2}$. The fact that stochastic tasks indicate events to be considered as fast as possible leads to