Computing Robustness of FlexRay Schedules to Uncertainties in Design Parameters

Arkadeb Ghosal, Haibo Zeng
General Motors Research
{arkadeb.ghosal,haibo.zeng}@gm.com

Marco Di Natale
Scuola Superiore SantAnna
marco@sssup.it

Yakov Ben-Haim
Technion - Israel Institute of Technology
yakov@techunix.technion.ac.il

***Abstract*—In the current environment of rapidly changing in-vehicle requirements and ever-increasing functional content for automotive EE systems, there are several sources of uncertainties in the definition of EE architecture design. This is also true for communication schedule synthesis where key decisions are taken early because of interactions with the suppliers. The possibility of change necessitates a design process that can analyze schedules for robustness to uncertainties, e.g., changes in estimated task durations or communication load. A robust design would be able to accommodate these changes incrementally without changes in the system scheduling, thus reducing validation times and increasing reusability. This paper introduces a novel approach based on the info-gap decision theory that provides a systematic scheme for analyzing robustness of schedules by computing the greatest horizon of uncertainty that still satisfies the performance requirements. The paper formulates info-gap models for potential uncertainties in schedule synthesis for a distributed automotive system communicating over a FlexRay network, and shows their application to a case study.**

I. Introduction

Rapid change in electronic control features in automotive systems introduces uncertainties in design decisions. This is especially true for the early-binding design process when critical decisions are made under the following uncertainties: (1) system requirements are captured prior to significant design and development, (2) the architecture needs to be developed sufficiently in advance to be available at the right production time frame, and (3) an architecture is projected to be reused across multiple implementations, and different features to cut down cost.

Uncertainties in the system or feature set may cause significant redesign down the life cycle of the system. To avoid such a situation, a system designer needs to estimate the robustness of a design to uncertainties; the most prevalent techniques for such analyses are max-min or sensitivity analysis.

Sensitivity studies the variation of the output due to changes in input; the goal is to identify the inputs which cause substantial change in output in contrast to those which cause minor change. *Min-max* or worst-case analysis chooses a design which minimizes the maximum loss at a specified level of uncertainty. In this paper, we discuss the use of the *info-gap methodology* [3] to evaluate design decisions based on their ability to tolerate uncertainty. Info-gap differs from the above techniques in (1) allowing unbounded horizon on uncertainty, (2) enabling exploration of structural or functional uncertainty, and (3) analyzing the design decisions across different regions of uncertainty and thus enabling the final decision to reflect the best possible robustness according to requirements.

A critical problem for any distributed system is the synthesis of communication schedules. The choice is made by considering time constraints/metrics (e.g., latency), or extensibility/uncertainty metrics (e.g., utilization).

When used for schedule selection, the info-gap technique does not necessarily choose one schedule; it provides different schedules for different ranges of uncertainty. This necessitates trend analysis to understand the zone of uncertainty, and the performance constraints of the system. The system model under study is an electronic control system implemented on a distributed architecture communicating over a FlexRay network. In this paper, we use the info-gap technique for evaluating robustness to uncertainty in the payloads of messages transmitted over the network. The technique can be extended for uncertainties in dependency (read-write relation between tasks and messages), number of tasks and messages, period (rate of task execution, or message transmission), and topology (mapping of tasks to hosts and messages to channels).

We next introduce references to work in the area of scheduling subject to uncertainty or sensitivity analysis. Section II presents an overview of the info-gap technique, and compares it with min-max and sensitivity analysis. Section III discusses a system model that uses a FlexRay network and the schedule synthesis problem. Section IV describes in detail the formulation and construction of info-gap models for uncertainty in the length of messages. Section V discusses a representative schedule synthesis problem. Section VI summarizes the key aspects and discusses the next steps.

A. Related work

The literature on FlexRay scheduling, scheduling extensibility and sensitivity analysis in real-time systems is rich. Sensitivity analysis was studied for priority-based scheduled distributed systems [11], with respect to end-to-end deadlines. In [10] a design optimization heuristics-based algorithm for mixed time-triggered and event-triggered systems was proposed. In [5], task allocation and priority assignment were defined with the purpose of optimizing the extensibility with respect to changes in task computation times; the proposed solution was based on simulated annealing. In [7], [9], [8], a generalized definition of extensibility on multiple dimensions

(including changes in the execution times of tasks, but also period speed-ups and possibly other metrics) was presented.

II. Info-Gap Theory: A Précis

Info-gap theory is a methodology for planning, design, and decision under severe uncertainty which has been applied to engineering design, project management, economics, biological conservation, medicine, and homeland security. We present some of the basic features of info-gap theory; for more detailed discussions refer to [2], [3], and *info-gap.com*.

An info-gap robustness analysis is based on three components: a system model, performance requirements, and uncertainty models. In time-triggered systems, the *system model* entails the number of task and messages with the flow topology, the durations of computation/communication, and the allocation of tasks/messages to ECUs/channels (ECU stands for Electronic Control Unit, a network node). The *performance requirements* include the latency constraints between data reads and data writes, the utilization bound of hosts and channels, and the buffer size restrictions for data transmission/reception. The *uncertainties* include execution time estimates for tasks, payloads for messages, dependencies among tasks and messages, period assignments, and the allocation of tasks and messages.

The three components of an info-gap analysis are combined in the evaluation of robustness and opportuneness. The *robustness against uncertainty* is the greatest horizon of uncertainty at which critical or minimal performance requirements are guaranteed; in other words, robustness is the degree of immunity of the system against unforeseen contingencies. The *opportuneness from uncertainty* is the lowest horizon of uncertainty at which windfall performance—much better than the critical performance—is possible (though not guaranteed). The opportuneness function expresses the propensity for better-than-anticipated outcomes. A simple example of an info-gap model is presented here. In Section IV we discuss in details the info-gap model corresponding to uncertainty in message payloads.

Consider the uncertainty in object (task or message) durations. The best estimate of the duration of the i-th object is $\widetilde{w}_i$, and the unit for error estimates is ϵ_i. There is no probability distribution of the actual duration, w_i, since the factors that cause deviation are varied and incompletely understood. Let $\widetilde{w}$ and ϵ denote the vectors of these estimates. A simple *info-gap model for uncertainty* in the object durations is the fractional-error model:

$$\mathcal{U}(\alpha) = \left\{ w : \ w_i \geq 0, \ \left| \frac{w_i - \widetilde{w}_i}{\epsilon_i} \right| \leq \alpha, \ \forall \ i \right\}, \quad \alpha \geq 0 \quad (1)$$

$\mathcal{U}(\alpha)$ represents an unbounded family of nested sets of object duration vectors w.

Robustness. Let $L(w, \delta)$ denote the model (for example, the end-to-end latency) of a system with uncertain object durations w and design parameters δ. Let $\mathcal{U}(\alpha)$ denote an info-gap model for uncertainty in w. The performance requirement is that $L(w, \delta)$ must not exceed a critical value L_{c} i.e.,

$$L(w, \delta) \leq L_{\mathrm{c}} \quad (2)$$

The robustness of the system represented by $L(w, \delta)$, with the requirement in (2) and uncertainties $\mathcal{U}(\alpha)$, is the greatest horizon of uncertainty at which the requirement is guaranteed to be satisfied:

$$\widehat{\alpha}(\delta, L_{\mathrm{c}}) = \max \left\{ \alpha : \ \left(\max_{w \in \mathcal{U}(\alpha)} L(w, \delta) \right) \leq L_{\mathrm{c}} \right\} \quad (3)$$

The true horizon of uncertainty is not known. However, the performance is guaranteed provided that the horizon of uncertainty does not exceed the robustness.

The robustness function induces a *robustness preference* ranking on the possible designs - δ, δ':

$$\delta \succ_{\mathrm{r}} \delta' \quad \text{if} \quad \widehat{\alpha}(\delta, L_{\mathrm{c}}) > \widehat{\alpha}(\delta', L_{\mathrm{c}}) \quad (4)$$

Opportuneness. Uncertainty can be propitious, and the opportuneness function evaluates the propensity for better-than-anticipated outcomes. Let L_{w} be a value of end-to-end latency which is smaller (better) than the requirement L_{c}. Latency as short as L_{w} is not required, but would be a valuable windfall should it occur. The opportuneness of the system $L(w, \delta)$, with uncertainty model $\mathcal{U}(\alpha)$, is the lowest horizon of uncertainty at which latency as short as L_{w} is possible, though not guaranteed:

$$\widehat{\beta}(\delta, L_{\mathrm{w}}) = \min \left\{ \alpha : \ \left(\min_{w \in \mathcal{U}(\alpha)} L(w, \delta) \right) \leq L_{\mathrm{w}} \right\} \quad (5)$$

A latency as short as L_{w} is possible if the horizon of uncertainty is no less than the opportuneness.

Relation to Sensitivity Analysis. The info-gap robustness function, $\widehat{\alpha}(\delta, L_{\mathrm{c}})$ can be thought of as sensitivity to uncertainty. There are, however, two differences between info-gap robustness and conventional what is usually referred to as sensitivity analysis. First, the info-gap robustness is not differential i.e., info-gap analysis does not consider only small variations. Rather, since the horizon of uncertainty is unknown, we ask what is the greatest error in our models, data and estimates which can be tolerated. Second, an info-gap model is well suited to represent uncertainty in functional relationships, as well as uncertainty in parameters or vectors.

Relation to the Min-Max Strategy. The min-max strategy selects the design that minimizes the maximal loss. The info-gap robustness function has a formal relation to the min-max strategy. However, there are two important differences. First, implementation of a min-max strategy requires knowledge of a worst case. In contrast, an info-gap model of uncertainty is explicitly designed to represent situations in which we do not know how wrong the best estimate can be. Second, even if we reliably know the worst that can occur, we may not want to design for that contingency. The clearest case is when the outcome anticipated from the min-max design is unacceptable because it violates the performance requirements.

III. FlexRay Standard and Models

FlexRay [1] is an automotive standard for high-speed and reliable communication that is being widely deployed for next-generation cars. FlexRay is a time-triggered protocol, which provides a global notion of time shared across all nodes, and is used to schedule communication and computational elements. In FlexRay, the global time base is comprised of an application cycle, defined as the least common multiple of all periods of tasks and messages, broken into an integer number of equal duration communication cycles. Each cycle contains up to four segments: static, dynamic, symbol, and network idle time (*nit*), as shown in Figure 1. Clock synchronization is embedded into the standard, and realized using part of the *nit* segment. Of the communication segments, the static part allows transmission of time-critical messages according to a periodic cycle in which the system allocates time slots to nodes for the transmission of their outgoing messages. The number of static slots is fixed at design time, and all slots have the same length. A static slot is assigned to a node for all cycles or is not assigned at all. Application data transmitted by a node may be less than the available data content of a slot.

The transmission time on the dynamic segment is partitioned in mini-slots with increasing identifiers. Each message has an associated frame ID. Messages (of different length) are transmitted when their ID matches the minislot identifier. In case of transmission, the duration of the minislot is extended to accomodate the message transmission time. The assignment of frame IDs to dynamic messages is done (at design time) based on priority, deadlines, or other considerations.

The scheduling problem is essentially the assignment of messages to the slots in accordance with the protocol rules and in such a way that the precedence (from sender task to message, and from message to receiver task) and timing constraints are met. The application software tasks are executed based on a dispatch table that defines the individual task start times.

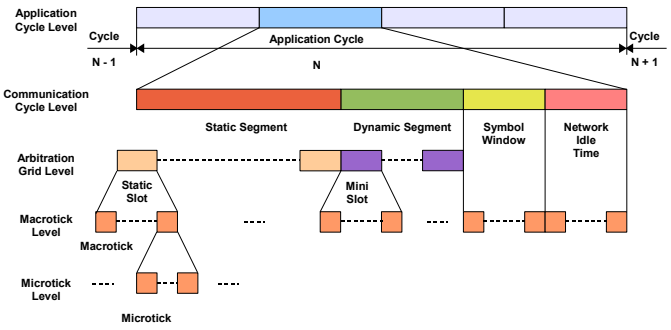

Fig. 1. FlexRay Timing Hierarchy

The FlexRay model used here assumes that the size of the application cycle, communication cycle, dynamic segment, symbol window and network idle time are expressed as a number of slots. A *slot* is assumed to be consistent and constant globally.

The system communications consist of a set of frames $\mathcal{F} = \{f_1, \ldots, f_n\}$ transmitted over an architecture $\mathcal{A}$ in which a set of hosts (ECUs) $\mathcal{H}$ are connected by a FlexRay bus. A host is a processor that performs computation and transmits/receives frames over the communication channel. A frame f_i is a collection of signals and is associated with a tuple $(h_i, o_i, d_i, \tilde{w}_i, bc_i, cr_i)$ where $h_i \in \mathcal{H}$ denotes the processor transmitting the frame, $o_i \in \mathbb{N}^+$ denotes the earliest, and $d_i \in \mathbb{N}^+$ the latest (among the static slots of a communication cycle) slots that can be assigned to f_i. The pair (o_i, d_i) is derived from the execution of the tasks transmitting and receiving the frame. In other words, the interval $[o_i, d_i]$ provides a window of transmission for the frame. The nominal size $\tilde{w} \in \mathbb{N}^+$ denotes the payload size of the frame (the sum of the size of its signals). The base cycle $bc \in \mathbb{N}^+$ denotes the offset, and the cycle repetition $cr \in \mathbb{N}^+$ denotes the periodicity of a frame with respect to an application cycle. The set of frames transmitted by a host h is denoted by $\mathcal{F}_h$.

A. Scheduling the Static Segment

Schedule synthesis solves the problem of mapping frames to static slots such that frames are transmitted within their respective windows. A static slot is fully identified by a pair: a slot number s, and communication cycle number cc. A schedule σ for static segment is a set of three mappings. The ***frame mapping*** β provides a mapping for each frame to a set of static slots; $\beta(f)$ is the set of slots assigned to frame f. The ***host mapping*** α denotes the slots assigned to each host; $\alpha(h)$ is the set of slots assigned to host h. The ***static slot mapping*** γ denotes the frame assigned to a particular static segment slot in a communication cycle; given a slot s, $\gamma(cc, s)$ denotes the frame assigned to the slot s in the communication cycle cc.

A schedule σ is *protocol-safe* if all of the following hold:

- A slot on FlexRay is not allocated to two different hosts, i.e. $\forall h_1, h_2 \in \mathcal{H}$: $\alpha(h_1) \cap \alpha(h_2) = \emptyset$.
- Two frames f_1, f_2 cannot be mapped to identical slots i.e., $\beta(f_1) \cap \beta(f_2) = \emptyset$ (each slot in a communication cycle is either empty or mapped to a unique frame).
- The size of a frame must be less than the slot size.
- The allocation of slots to frames must be consistent with the allocation of slots to hosts. If a slot is allocated to a frame, then it must also be allocated to the source host for the frame, i.e., $f_i = \gamma(cc, s) \Rightarrow (cc, s) \in \alpha(h_i)$.
- A slot in a communication cycle may be empty and still occupied by a host.

A schedule σ is *execution-safe* if for each frame, it assigns enough slots to accommodate the payload, and each frame is assigned transmission slots within the feasible time window. A schedule is *valid* if the schedule is *protocol-safe* and *execution-safe*. FlexRay scheduling, however, is not the objective of this paper. Please refer to [12] for the definition of the scheduling problem, the protocol restrictions, and the solutions.

B. Scheduling Policies and Performance Requirements

Schedules can be differentiated based on the strategy used to define them, or the performance objective of the system:

- *Strategy of Schedule Synthesis.* A schedule can be generated based on deadline, priority, or other user-defined constraints. *Deadlines* or *priorities* may determine the scheduling order of two frames if both are ready to be transmitted at identical time instants. Other than the above constraints, the user may provide *limitations* e.g. a frame can only be scheduled in specific slots.
- *Performance of Synthesized Schedule.* Several metrics can be used to evaluate the quality of a schedule. Possible options are slack, latency, utilization and buffer size. *Slack* is the minimum idle time available within a given time window; the intent is to provide space for extensibility. *Latency* is the delay between two events like reading of input, and writing of output. *Utilization* is a measure of the resources used, and is normally expressed as a fraction of processor time or network bandwidth. *Buffer size* denotes the storage required by hosts at the interface of channels. The storage is used for frames that are ready to be transmitted, or the frames that are received from the channels. Of those, latency and buffer sizes are typically also associated with (possibly) hard constraints.

In our analysis, we tried two different performance metric functions: the latency on a critical path, and the utilization of the communication cycles. The buffer requirements are assumed to be handled by the assignment of transmission windows to frames, and the mapping of frames to hosts. The application of info-gap is of course not limited to these performance functions. The end-to-end latency $L_{i,j}$ associated with a path $P_{i,j}$ is defined as the largest possible time interval that is required for the change of the input at one end of the path to be propagated to the last task at the other end of the path. The static segment utilization Γ_{cc} for a communication cycle cc is defined as the ratio of used slots (of the static segment) to the total number of slots (in the static segment) in the communication cycle. The static utilization of the system Γ is the maximum utilization among all communication cycles, i.e., $\Gamma = \max_{cc} \Gamma_{cc}$.

IV. INFOGAP MODEL

The model assumes that the size of the frames may be uncertain, i.e., there is a possible error in the estimation of the frame payload. Given a frame f, the actual size w_f is $\tilde{w}_f + \alpha\epsilon_f$ where $\tilde{w}_f$ is the nominal size, ϵ_f is the unit considered for the estimation error, and α is the unknown horizon of uncertainty. Let $\widetilde{w}$ denote the vector of estimated size of frames, ϵ denote the vector of error units and w denote the vector of unknown actual durations. The fractional-error info-gap model in (1) is used for the info-gap quantification of uncertainty in frame sizes.

Given a schedule σ, the robustness is the greatest horizon of uncertainty up to which the static segment utilization of the system (resp. latency of the critical path) under a (possibly modified) schedule is less than the critical static segment utilization (resp. critical latency of the path). A schedule σ under uncertainty α is denoted as $\sigma(\alpha)$. $\sigma(\alpha = 0)$ is the schedule obtained assuming the nominal size of frames. At any uncertainty value $\alpha > 0$, the additional payload to be transmitted is $\lceil \alpha \times \epsilon_f \rceil$, and the frame size is $\tilde{w}_f + \lceil \alpha \times \epsilon_f \rceil$.

The uncertainty is modeled as $\mathcal{U}(\alpha, \tilde{w})$ where each frame size ranges in between $\tilde{w}_f - \alpha\epsilon_f$ and $\tilde{w}_f + \alpha\epsilon_f$. The robustness is computed as follows. The utilization $\Gamma_{cc}(w)$ of a communication cycle cc is computed based on the actual frame sizes w at a given uncertainty. The robustness of the schedule is the greatest horizon of uncertainty up to which system utilization is within a critical value, Γ_C. Formally, the robustness is

$$\hat{\alpha} = \max\{\alpha : \max_{cc} \max_{w \in \mathcal{U}(\alpha, \tilde{w})} \Gamma_{cc}(w) \leq \Gamma_C\} \tag{6}$$

Intuitively, the net robustness $\hat{\alpha}$ is the robustness of the least robust communication cycle. When latency of the critical path is considered, the robustness of the schedule is the greatest horizon of uncertainty up to which the latency L_P of the critical path P is within a critical value L_C. Formally, the robustness is

$$\hat{\alpha} = \max\{\alpha : \max_{w \in \mathcal{U}(\alpha, \tilde{w})} L_P(w) \leq L_C\} \tag{7}$$

Consider a simple system (Figure 2) in which a computation pipeline with three tasks and two frames ($T1 \rightarrow f1 \rightarrow T2 \rightarrow f2 \rightarrow T3$) is scheduled on two ECUs and one Network. Time (both for computation and communication) is allocated in slots. Schedule A (top part of the figure) allows lower end-to-end latency than schedule B (bottom part of the figure) when there is no uncertainty in the frame sizes. However, there may be an error in estimating the size of the frames, which may include future uncertainties, and/or overlooked communication. In the info-gap model for this example, the uncertainty of one slot is 1 step of uncertainty (or 100% change). For the robustness computation (when the frame size is increased), if the current slot has no space, then the next available slot is used. At a given uncertainty value, the system is checked to ensure that there is a feasible schedule for increase in all frame payloads. For the above system, both schedules can accommodate the extension. However, for the schedule A, the latency increases (from 5 to 21 time units) while the latency remains the same for the second schedule (7 time units). If the critical latency of the path is 10, schedule B is a better choice than schedule A, as schedule B can tolerate changes in the frame payload (and still be within the desired latency) while schedule A cannot. However if the uncertainty is 200% or the payload changes by 2 slots, then the latency for the extended schedule jumps to 21 (for schedule A) and 23 (for schedule B). Depending on the horizon of uncertainty and the initial latency, schedule B may be more robust than schedule A. Note that the robustness computation depends on the extension policy, and the uncertainty modeling (e.g., what is the implication of one step of uncertainty). If the extension policy for the schedules was different, then the robustness may have been different. Similarly, if the model of uncertainty had been different (e.g., 1 step of uncertainty denoting a 2-unit change instead of 1 unit), then the robustness computation would have been different.

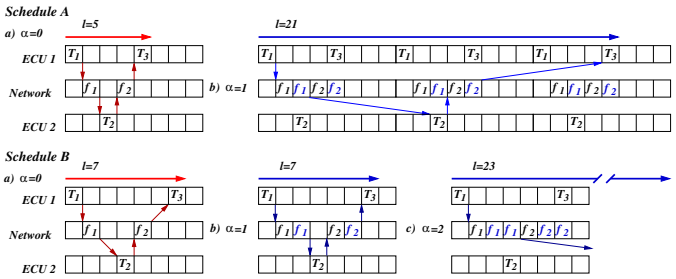

Fig. 2. Comparing Robustness

V. Case Study

Table I describes the tasks of an X-by-Wire application from an automotive OEM. The application has 10 ECUs interconnected by a FlexRay bus. The 47 application tasks exchange 132 signals with periods of $1ms$, $4ms$ and $8ms$. The FlexRay bus is configured with a communication cycle of $1000\mu s$ with 22 static slots. Each slot is $35\mu s$ and can accommodate 200 bits of signal payload. The application cycle of the case study is $8000\mu s$.

τ_i	e_i	$T_i(\mu s)$	$C_i(\mu s)$	τ_i	e_i	$T_i(\mu s)$	$C_i(\mu s)$
τ_8	e_9	8000	810	τ_{20}	e_{10}	8000	230
τ_9	e_9	8000	550	τ_{21}	e_5	1000	25
τ_{11}	e_9	8000	100	$\tau_{22}/\tau_{26}/\tau_{30}/\tau_{34}$	$e_5/e_6/e_7/e_8$	1000	60
τ_{12}	e_9	8000	770	$\tau_{25}/\tau_{29}/\tau_{33}$	$e_6/e_7/e_8$	1000	30
τ_{13}	e_9	8000	200	$\tau_{24}/\tau_{28}/\tau_{32}/\tau_{36}$	$e_5/e_6/e_7/e_8$	1000	20
τ_{14}	e_9	8000	110	$\tau_{23}/\tau_{27}/\tau_{31}/\tau_{35}$	$e_5/e_6/e_7/e_8$	1000	40
τ_{15}	e_9	8000	550	$\tau_{37}/\tau_{42}/\tau_{47}/\tau_{52}$	$e_1/e_2/e_3/e_4$	8000	1000
τ_{16}	e_{10}	8000	780	$\tau_{38}/\tau_{43}/\tau_{48}/\tau_{53}$	$e_1/e_2/e_3/e_4$	8000	500
τ_{10}	e_{10}	8000	510	$\tau_{41}/\tau_{46}/\tau_{51}/\tau_{56}$	$e_1/e_2/e_3/e_4$	8000	350
τ_{17}	e_{10}	8000	190	$\tau_{39}/\tau_{44}/\tau_{49}/\tau_{54}$	$e_1/e_2/e_3/e_4$	8000	1500
τ_{18}	e_{10}	8000	260	$\tau_{40}/\tau_{45}/\tau_{50}/\tau_{55}$	$e_1/e_2/e_3/e_4$	4000	1300
τ_{19}	e_{10}	8000	100				

τ : task, e : transmitting ecu, T : period, C : wcet

TABLE I
Tasks of the X-by-Wire example

A manually generated schedule for the design is shown in Table II. It uses 18 slots with 22 frames. The table shows the size of each frame and its position with respect to communication cycles (numbered 0 to 7 in the columns), and static segment slots (numbered 1 to 22 in the rows). The frames sent in slots 5, 6, 7, 8, 13, 14, 16, and 17 are transmitted with periodicity $1ms$ (base cycle = 0); all other frames are sent with periodicity $8ms$ (the base cycle is the same as the communication cycle in which they are transmitted). A different schedule is generated automatically based on the method in [12], which uses an MILP (Mixed Integer Linear Programming) optimization framework to find the schedule with minimum number of used slots. Table III shows the result of the automatically generated schedule which uses 13 slots and has 31 frames. The frames sent in slots 2, 5, 6, 7, 8, 9, and 19 are transmitted with periodicity $1ms$ (base cycle = 0); all other frames are sent with periodicity $8ms$ (the base cycle is the same as the communication cycle of their transmission).

Robustness Analysis. The schedules are studied for robustness against uncertainties in the payload of the frames. At each level of uncertainty, a possible schedule is generated by finding additional slots for each extended frame. When more payload is added, a frame may use space in the slots already allocated to it, or possibly be extended to open slots. Each frame is assumed to have an uncertainty unit ϵ of 1 byte of payload; thus for $0 < \alpha \leq 1$, the extra payload is $\lceil \alpha \times 1 \rceil = 1$ byte. At each uncertainty, frames are checked in order whether there is enough space for extension. In our analysis, the space for the frames is allocated in order of increasing transmission window deadlines and, as a secondary metric, in order of decreasing payload. The results for the latency-oriented analysis are (Figure 3):

slot	sender	0	1	2	3	4	5	6	7
1	e_1			32				160	
2	e_2			32				160	
3	e_3			32				160	
4	e_4			32				160	
5	e_5	160	160	160	160	160	160	160	160
6	e_6	160	160	160	160	160	160	160	160
7	e_7	192	160	160	160	160	160	160	160
8	e_8	192	192	192	192	192	192	192	192
9	e_9					192			
10	e_9					192			
13	e_7	1	1	1	1	1	1	1	1
14	e_8	64	64	64	64	64	64	64	64
15	e_9				120				
16	e_5	112	112	112	112	112	112	112	112
17	e_6	112	112	112	112	112	112	112	112
19	e_9								128
20	e_{10}								128
21	e_9		1						

(Slots 11, 12, 18, 22 are unassigned.)

TABLE II
Schedule Generated Manually

slot	sender	0	1	2	3	4	5	6	7
1	e_2		32			160			
2	e_5	160	160	160	160	160	160	160	160
3	e_{10}			96				32	
4	e_9	104	65	80	192	16			176
5	e_7	193	193	193	193	193	193	193	193
6	e_8	112	112	112	112	112	112	112	112
7	e_6	200	200	200	200	200	200	200	200
8	e_6	72	72	72	72	72	72	72	72
9	e_5	112	112	112	112	112	112	112	112
10	e_1	16	64				32	80	
11	e_4	96	16	16	16	32	16		
12	e_3	16	48		80	48			
19	e_8	144	144	144	144	144	144	144	144

(Slots 13, 14, 15, 16, 17, 18, 20, 21, 22 are unassigned.)

TABLE III
Schedule Generated Automatically

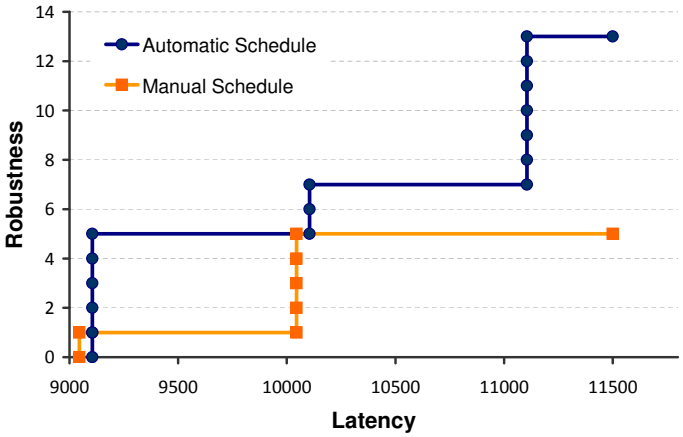

Fig. 3. Analysis With Latency Performance.

- At zero uncertainty, the latency of the critical path for the manual design ($9045\mu s$) is less than that of the periodic schedule ($9105\mu s$). The manual schedule would be preferrable if there is no uncertainty in the design. The manual design also tolerates uncertainty of 1 (i.e., each frame can have an additional load of 8 bits) at latency $9045\mu s$.
- If the allowed latency on the critical path is less than $10045\mu s$, then the automatic schedule is the better choice as it has a robustness of 5. If the allowed latency on the critical path is $100045\mu s$, then both schedules are comparable (both tolerate an uncertainty level of 5).
- If the allowed latency on the critical path is greater than $10045\mu s$, then the automatic schedule is better. For any latency greater than $10045\mu s$, the manual design tolerates an uncertainty level of 5 (i.e., each frame can have an additional load of 40 bits). For latencies between $10045\mu s$ and $11105\mu s$, the automatic schedule tolerates an uncertainty of 7 units; for latencies greater than $11105\mu s$, the schedule tolerates an uncertainty of 13 units (i.e., each frame can have an additional load of 104 bits).

The results for the utilization-based metric are (Figure 4):

- At zero uncertainty ($\alpha = 0$), the manual design ($\Gamma = .55$) is comparable to the automatic schedule ($\Gamma = .55$).
- If the uncertainty is less than 1 (i.e., each frame can increase in size by at most 8 bits), then the manual design is a better choice as the manual design has a robustness of 1 at $\Gamma = .55$,
- For utilization values $\Gamma <= .64$, both designs are comparable - each has a robustness of 5 units (i.e., each frame can accommodate an additional load of 40 bits).
- If $\Gamma > .68$ is allowed, then the automatic design is a better choice. While the manual design can tolerate a maximum robustness of 5, the automatically generated schedule can accomodate 7 additional units at $\Gamma = .68$, 11 units at $\Gamma = .73$, and 13 additional units of extension at $\Gamma = .82$.
- If the utilization is bound to the maximum limit of 100%, then the automatic schedule is a better choice over the manual design as the former has a higher maximum robustness.

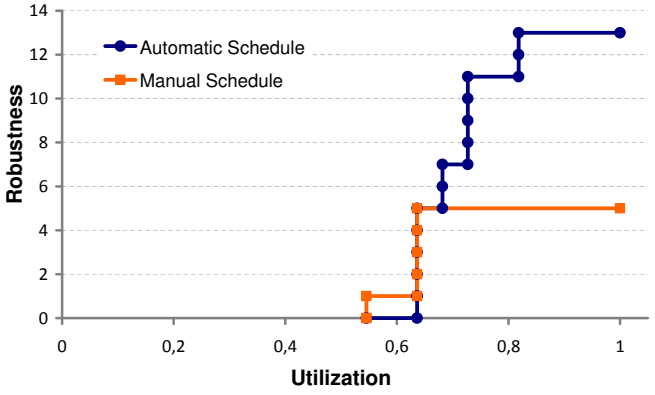

Fig. 4. Analysis With Utilization Performance.

For almost all values in our selected ranges for the two performance metrics, the schedule produced with the optimization method in [12] allows for higher robustness. This confirms in quantitative terms the intuitive perception that a higher number of free slots in a FlexRay schedule allows to better accomodate future loads and therefore provides for more extensibility against uncertainty.

VI. Conclusion

This paper presented a methodology for the analysis of FlexRay schedules subject to uncertainty in frame payloads. The *uncertainties* we considered arise from unknown future loads and/or overlooked communication. Probability distributions for the uncertain quantities are unknown and we used non-probabilistic info-gap models of uncertainty. We studied and developed info-gap models for a wide range of uncertainties: an example on the data content of messages frames is presented and discussed in this paper with realistic case studies. The *robust-satisficing* strategy presented here proposes *maximizing the robustness* to uncertainty and *satisfying latency and utilization requirements*. The practical implication of the robust-satisficing strategy is obvious when the robustness curves for two different schedules cross, as illustrated in our examples. Our plans for future work include exploration of different types of extensibility, such as a variable additional load for each frame (possibly proportional to the frame length), but also additional computation load, including additional execution time for the system tasks. Finally, we plan to include the possible addition of new tasks and messages.

References

[1] FlexRay communications system specifications 2.1. 2005.

[2] Y. Ben-Haim. *Engineering Design Reliability Handbook*, vol 3, chapter Info-gap Decision Theory For Engineering Design. Or: Why 'Good' is Preferable to 'Best'. CRC Press, 2005.

[3] Y. Ben-Haim. *Info-gap Decision Theory: Decisions Under Severe Uncertainty (2nd edition)*.

[4] Yakov Ben-Haim, Clifford C. Dacso, Jonathon Carrasco and Nithin Rajan, Heterogeneous Uncertainties in Cholesterol Management International Journal of Approximate Reasoning, 50: 1046–1065.

[5] I. Bate and P. Emberson. Incorporating scenarios and heuristics to improve flexibility in real-time embedded systems. In *12th IEEE RTAS Conference*, pages 221–230, April 2006.

[6] Enrico Bini, Marco Di Natale, and Giorgio Buttazzo. Sensitivity analysis for fixed-priority real-time systems. In *Euromicro Conference on Real-Time Systems*, Dresden, Germany, June 2006.

[7] A. Hamann, R. Racu, and R Ernst. A formal approach to robustness maximization of complex heterogeneous embedded systems. In *Proc. of the CODES/ISSS Conference*, October 2006.

[8] A. Hamann, R. Racu, and R Ernst. Methods for multi-dimensional robustness optimization in complex embedded systems. In *Proc. of the ACM EMSOFT Conference*, September 2007.

[9] A. Hamann, R. Racu, and R Ernst. Multi-dimensional robustness optimization in heterogeneous distributed embedded systems. In *Proc. of the 13th IEEE RTAS Conference*, April 2007.

[10] T. Pop, P. Eles, and Z. Peng. Design optimization of mixed time/event-triggered distributed embedded systems. In *Proc. of the CODES+ISSS Conference*, New York, NY, USA, 2003. ACM Press.

[11] R. Racu, M. Jersak, and R. Ernst. Applying sensitivity analysis in real-time distributed systems. In *Proc. of the RTAS Conference*, San Francisco (CA), U.S.A., March 2005.

[12] H. Zeng, W. Zheng, M. Di Natale, A. Ghosal, P. Giusto, and A. Sangiovanni-Vincentelli. Scheduling the FlexRay Bus Using Optimization Techniques. In *Proceedings of the 46th IEEE ACM Design Automation Conference*, 2009.