Analyzing End-to-End Delays in Automotive Systems at Various Levels of Timing Information*

Matthias Becker[1], Dakshina Dasari[2], Saad Mubeen[1], Moris Behnam[1], Thomas Nolte[1]
[1]Mälardalen University, Västerås, Sweden
{matthias.becker, saad.mubeen, moris.behnam, thomas.nolte}@mdh.se
[2]Robert Bosch GmbH, Renningen, Germany
dakshina.dasari@de.bosch.com

ABSTRACT

Software design for automotive systems is highly complex due to the presence of strict data age constraints for event chains in addition to task specific requirements. These age constraints define the maximum time for the propagation of data through an event chain consisting of independently triggered tasks. Tasks in event chains can have different periods, introducing over- and under-sampling effects, which additionally aggravates their timing analysis. Furthermore, different functionality in these systems, is developed by different suppliers before the final system integration on the ECU. The software itself is developed in a hardware agnostic manner and this uncertainty and limited information at the early design phases may not allow effective analysis of end-to-end delays during that phase. In this paper, we present a method to compute end-to-end delays given the information available in the design phases, thereby enabling timing analysis throughout the development process. The presented methods are evaluated with extensive experiments where the decreasing pessimism with increasing system information is shown.

1. INTRODUCTION

Automotive systems are getting complex with respect to traditional components like the Engine Management System (EMS) as well as modern features like assisted driving. While the increase in the EMS complexity is attributed to newer hybrid engines and stricter emission norms, assisted driving requires the perfect convergence of various technologies to provide safe, efficient and accurate guidance. This has led to software intensive cars containing several million lines of code, spread over up to hundred Electronic Control Units (ECU) [10].

Given this complexity, over the last decades, standards like AUTOSAR [5] have been formulated in order to provide a common platform for the development of automotive software. These standards allow software components provided by different suppliers to be integrated on the same ECU, since they provide for a hardware agnostic software development. Such robust interfaces enable designers to design software at early stages without knowledge of the concrete hardware platform on which it will be eventually executed. Thus, during the development it is often not known which other applications share the same execution platform.

Most of these automotive applications typically have strict real-time requirements – it is not only important that a computation result is the correct result, but also that the result is presented at the correct time. In addition to the timing requirements for each task execution (i.e. the tasks deadline), these applications often have constraints for the data propagation through a chain of tasks, so called *end-to-end timing constraints*, one of which is the age constraint. The age constraint specifies the maximum time from reading a sensor value until the corresponding output value is produced at the end of the chain. This kind of constraint is especially important for control systems, such as the EMS, where it directly influences the quality of control.

Many design decisions have direct influence on the data age. Thus, bounding the data age of a chain early during the design process can potentially avoid costly software redesigns at later development stages. The analysis gets complex as a chain may consist of tasks with different periods leading to over- and under-sampling situations. Most of the available analysis methods for such systems analyze existing schedules [11] and thus they are not applicable during early phases. In [8], a generic framework to calculate the data age of a cause-effect chain is presented, which targets single processor systems and is agnostic of the scheduling algorithm used. In this paper we show how this analysis can be extended to cater the needs of the complete development process of automotive applications. The increased system information during the design process can thus be used to obtain end-to-end latencies with decreasing pessimism at various levels of timing information.

Contributions: In this work, we extend our earlier work on analyzing end-to-end delays among multi-rate effect chains [8] to utilize the information available in different development stages. Specifically, we highlight the generic nature of the framework by showing how extensions with varied levels of information can be used to compute the maximum data age given the:

1. Knowledge of offsets: The analysis for systems without knowledge of the schedule is extended to allow for task release-offset specifications.

2. Knowledge of the scheduling algorithm (like Fixed Priority Scheduling (FPS)): Most ECUs utilize operating systems which schedule tasks based on FPS. This allows to utilize existing analysis for such systems to determine worst-case response times of the individual tasks. It is then shown how the concepts of the analysis can be adapted to account for this information.

3. Knowledge of the exact schedule: Similar to most of the existing end-to-end delay analyses, we show how the exact schedule can be used to determine the exact delays with low computational overheads.

4. Knowledge of the communication semantic: We extend the analysis to incorporate Logical Execution Time (LET), an important paradigm guiding how and when data is exchanged between tasks of an automotive application [12].

Finally, we compare these different scenarios with extensive evaluations, considering i) the tightness of the computed bounds and ii) the computation time for the analysis.

2. RELATED WORK

The end-to-end timing constraints found in automotive multi-rate systems were first described in [19]. Here, the authors describe the different design phases and link them to EAST-ADL [4] and AUTOSAR [5]. An increased level of system knowledge during the consecutive design phases is outlined.

A method to compute the different end-to-end delays of multi-rate cause-effect chains is presented in [11]. In addition, the authors relate the reaction delay to "*button to reaction*" functionality and the maximum data age delay to "*control*" functionality. In this work the focus lies on the maximum data age and hence on control applications.

A model-checking based technique to compute the end-to-end latencies in automotive systems is proposed in [17]. The authors generate a formal model based on the system description which is then analyzed.

The end-to-end timing analysis in an industrial tool suite is discussed in [16]. Two different activation methods are discussed; trigger chains, where a predecessor task triggers the release of a successor task, and data chains, where tasks are individually triggered and hence over- and under-sampling may occur. In this work we focus on chains with the latter activations.

End-to-end delays in heterogeneous multiprocessor systems are analyzed in [18]. Ashjae et al. [7] propose a model for end-to-end resource reservation in distributed embedded systems, and also present the analysis, based on [11], for end-to-end delays under their model.

Additionally, several industrial tools implement the end-to-end delay analysis for multi-rate effect chains [20, 6, 21]. However all of the discussed works require system information which is only available in the implementation level. In [8], a scheduling agnostic end-to-end delay analysis for data age is described, where only information about the tasks of the cause-effect chains is required. In this work, *we extend the results presented in [8], and show that by augmenting information available during the different design phases, we can analyze the maximum data age with decreasing degree of pessimism.*

3. SYSTEM MODEL

This section introduces the application model, inter task communication mechanisms, and the notion of cause-effect chains as used in this work.

3.1 Application Model

We model the application as a set of periodic tasks Γ. Each task $\tau_i \in \Gamma$ can be described by the tuple $\{C_i, T_i, \Psi_i\}$, where C_i is the task's Worst Case Execution Time (WCET), and T_i is the task's period. All tasks have implicit deadlines, i.e. the deadline of τ_i is equal to T_i. A task can further have a release offset Ψ_i. For all tasks executing on a processor, the hyperperiod can be defined as the least common multiple of all periods, $HP = \text{LCM}(\forall T_i, i \in \Gamma)$. Hence, a task τ_i executes a number of jobs during one HP, where the j^{th} job is denoted as $\tau_{i,j}$.

3.2 Communication between Tasks

In this work inter-task communication is realized via shared registers. This is a common form of communication and can be found in many industrial application areas. In such a communication model, a sending task writes an output value to a shared register. Similarly, a receiving task reads the current value of this register. Hence, there is no signaling between the communicating tasks, and a receiving task always consumes the newest value (i.e. last-is-best).

In order to increase determinism, tasks operate on the *read-execute-write* model. Meaning, a task reads all its input values into local copies before the execution starts. During the execution phase only those local copies are accessed. Finally, at the end of the execution the task writes the output values to the shared registers, making them available to other tasks. In short, reading and writing of input and output values is done at deterministic points in time, at the beginning and end of the tasks execution respectively. This is a common communication form found in several industrial standards (i.e. in AUTOSAR this model is defined as the *implicit communication* [3]. Also the standard IEC 61131-3 for the automation systems defines similar communication mechanisms [1]).

3.3 Cause-Effect Chains

For many systems it is not only important that the individual tasks execute within their timing constraints but also that the data propagates through a chain of tasks within certain time bounds. One example is the Air Intake System (AIS), which is part of the Engine Management System (EMS) in a modern car. For a smooth operation, the air and fuel mixture inside the engine must be controlled and the AIS is responsible for injecting the correct amount of air. To do so, an initial sensor task periodically samples the position of the pedal, followed by a number of control tasks that process this information, and finally an actuator task actuates the throttle to regulate the amount of air inside the engine. For the control algorithm, it is important that the sensed input data is fresh in order to reach the required control quality. Hence, the time from reading the data until the actuation is subject to delay constraints in addition to the task's individual timing constraints.

In AUTOSAR such constraints are described by the *cause-effect chains* [2]. For a task set Γ, a set of cause-effect chains Π can be specified. Where Π contains the individual cause-effect chains ζ_i. A chain ζ_i is represented by a Directed Acyclic Graph (DAG) $\{\mathcal{V}, \mathcal{E}\}$. The nodes of the graph are represented by the set $\mathcal{V}$ and contain the tasks involved in the cause-effect chain. The set $\mathcal{E}$ includes all edges between the nodes. An edge from τ_i to τ_k implies that τ_i has at least one output variable which is consumed by τ_k. A cause-effect chain can have forks and joins, but the first and the last task in the chain must be the same for all possible data paths. To simplify the analysis, chains with fork/join operations are decomposed into individual sequential chains. Hence,

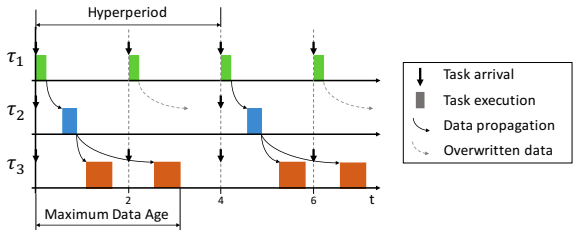


Figure 1: Data propagation between tasks of a cause-effect chain in a real-time system with maximum data age specified.

all cause-effect chains in Π are sequential.

End-to-End Timing Requirements

For each cause-effect chain, an end-to-end timing requirement can be specified. Several end-to-end timing requirements are defined for automotive systems [2, 4]. In this work the *data age*, as the most important timing requirements for control systems, is examined. A detailed discussion of end-to-end delays is provided in [11].

For the data age, the maximum time from sampling an initial input value at the beginning of the cause-effect chain, until the last time this value has influence on the produced output of the cause-effect chain is of interest. Fig. 1 depicts an example with three tasks, τ_1, τ_2, and τ_3. All tasks are part of a cause-effect chain in this order. Note that τ_1 and τ_3 are activated with a period of $T = 2$, while τ_3 is activated with a period of $T = 4$. This leads to over- and under-sampling between the different tasks. While the output value of the first instance of τ_1 is consumed by the first instance of τ_2, the data produced by the second instance of τ_1 is overwritten before τ_2 has the chance to consume it. Similarly, the data produced by the first instance of τ_2 is consumed by the first instance of τ_3. Since no new data is produced before the second instance of τ_3 is scheduled the same data is consumed. In the example, this constitutes the maximum data age, from sampling of the first instance of τ_1 until the last appearance of the data at the output of the second instance of τ_3.

4. RECAP: CALCULATION OF DATA PROPAGATION PATHS

In this section we recapitulate the calculations of data propagation paths for systems without prior knowledge of the schedule, as this is basis for the work presented in this paper. For a more in depth explanation a reader is referred to [8].

4.1 Reachability between Jobs

The main concept to decide if data can be propagated between two distinct jobs are the *read interval* and the *data interval*. For a job $\tau_{i,j}$, the read interval is defined as the interval starting from the earliest time a job can potentially read its input data ($R_{min}(\tau_{i,j})$) until the last possible time a job can do so without violating its timing constraints ($R_{max}(\tau_{i,j})$). Similarly, the data interval is defined as the interval from the earliest time the output data of a job can be available ($D_{min}(\tau_{i,j})$) up to the latest time a successor job of the same task overwrites the data ($D_{max}(\tau_{i,j})$). Hence, the read interval $RI_{i,j}$ is the interval $[R_{min}(\tau_{i,j}), R_{max}(\tau_{i,j})]$, and the data interval is $[D_{min}(\tau_{i,j}), D_{max}(\tau_{i,j}))$. These concepts are depicted in Fig. 2 for jobs of a task τ_i. For a system without any knowledge of the scheduling decisions one has to assume that a job can be scheduled anywhere, as long as it starts not before its release and finishes not after its deadline. In [8], the notations to define the intervals for systems without offset are defined as follows:

$$R_{min}(\tau_{i,j}) = (j-1)\cdot T_i \quad (1)$$
$$R_{max}(\tau_{i,j}) = R_{min}(\tau_{i,j+1}) - C_i \quad (2)$$
$$D_{min}(\tau_{i,j}) = R_{min}(\tau_{i,j}) + C_i \quad (3)$$
$$D_{max}(\tau_{i,j}) = R_{max}(\tau_{i,j+1}) + C_i \quad (4)$$

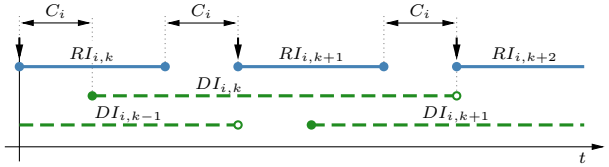


Figure 2: Read and data intervals of consecutive jobs of τ_i if no scheduling information is available.

4.1.1 Deciding Reachability Between Jobs

In order for a job $\tau_{k,l}$ to consume data of a job $\tau_{i,j}$ the data interval of $\tau_{i,j}$ must intersect with the read interval of $\tau_{k,l}$. The function Follows$(\tau_{i,j}, \tau_{k,l})$ is defined to return *true* if this is the case:

$$\text{Follows}(\tau_{i,j}, \tau_{k,l}) = \begin{cases} true, & \text{if } RI_{i,j} \cap DI_{i,j} \neq \emptyset \\ false, & \text{otherwise} \end{cases}$$

4.1.2 Adjusting the Data Interval for Long Chains

In order to capture the characteristics of data propagation in a cause-effect chain of length > 2, the data interval needs to be modified. Assume the first job of τ_i, as shown in Fig. 2 is followed by a job of a task τ_k. τ_k is released with the same period as that of τ_i, but the execution time of τ_i is shorter than the one of τ_k. Follows$(\tau_{i,1}, \tau_{k,1})$ returns true and indicates that $\tau_{k,1}$ can potentially consume the data of $\tau_{i,1}$. However, in order to decide reachability between the $\tau_{k,1}$ and a third task in the chain the data interval of $\tau_{k,1}$ must be modified. This is the case because $\tau_{k,1}$ can consume the data of $\tau_{i,j}$ earliest at time $D_{min}(\tau_{i,j})$. Consequently, this data can earliest be available as output data of $\tau_{k,l}$ at time $D_{min}(\tau_{i,j}) + C_k$. $D'_{min}(\tau_{k,l}, \tau_{i,j})$ defines the starting time of the data interval of $\tau_{k,l}$ if the data produced by $\tau_{i,j}$ shall be considered as well:

$$D'_{min}(\tau_{k,l}, \tau_{i,j}) = \max(D_{min}(\tau_{i,j}) + C_k, D_{min}(\tau_{k,l}))$$

Note that the data interval only needs to be adjusted if $D_{min}(\tau_{k,l})$ is smaller than $D_{min}(\tau_{i,j}) + C_k$. These modifications are local for the specific data path, hence, if another combination of jobs is involved then the original data interval must be used.

4.2 Calculating Data Paths

To calculate all possible data propagation paths in a system, a recursive function is used. This function constructs all possible data propagation paths from a job of the first node in a cause-effect chain up to the job of a last node of the chain. Consequently this needs to be done for all jobs of the first task of a chain, inside the hyperperiod of the chain.

The function starts at the first level of the cause-effect chain, for the initial job all jobs of the second task of the

Table 1: Description of the read and data interval for different systems with different levels of timing information.

	No Knowledge	Exact Schedule Known	WCRT Known	LET Execution Model
$R_{min}(\tau_{i,j})$	$\Psi_i + (j-1) \cdot T_i$	$start_{i,j}$	$\Psi_i + (j-1) \cdot C_i$	$(j-1) \cdot T_i$
$R_{max}(\tau_{i,j})$	$j \cdot T_i - C_i$	$R_{min}(\tau_{i,j})$	$R_{min}(\tau_{i,j}) + WCRT_i - C_i$	$R_{min}(\tau_{i,j})$
$D_{min}(\tau_{i,j})$	$R_{min}(\tau_{i,j}) + C_i$	$end_{i,j}$	$R_{min}(\tau_{i,j}) + C_i$	$j \cdot T_i$
$D_{max}(\tau_{i,j})$	$R_{max}(\tau_{i,j+1}) + C_i$	$end_{i,j+1}$	$R_{max}(\tau_{i,j+1}) + C_i$	$(j+1) \cdot T_i$

chain are found where Follows() returns *true*. To these nodes a logical data path is created. The same principle is applied from each of these nodes to the jobs of the next lower level of the cause-effect chain. Once the last level is reached all possible paths are calculated and the function returns. Interested readers are referred to [8] for a detailed explanation.

4.3 Constructing Data Propagation Paths and Maximum Data Age

For one data path the maximum end-to-end latency, and thus the data age, can be computed as follows, where τ_{start} is a job of the first task of the cause-effect chain, and τ_{stop} is a job of the last task of a cause-effect chain:

$$\text{AgeMax}(\tau_{start}, \tau_{end}) = (R_{max}(\tau_{end}) + C_{end}) - R_{min}(\tau_{start})$$

In order to compute the maximum data age for any possible path in the system, AgeMax() must be computed for all computed data paths. The maximum of these values is the maximum data age of the cause-effect chain.

5. REACHABILITY BETWEEN JOBS IN DIFFERENT SYSTEMS

The basic computation of data age latencies without prior knowledge of the schedule can result in pessimistic results. Many of the computed data propagation paths may not occur since scheduling algorithms impose a recurring order of jobs in each hyperperiod of the task set. In this section, modifications of the read and data interval are presented to reflect the behavior of systems with more elaborate knowledge on the scheduling decisions. One key observation is that the presented method to calculate the different data paths and the maximum data age is independent of the concrete system model as long as the read and data intervals are adjusted accordingly. Table 1 depicts the required changes to the read and data interval for different levels of system information. The remainder of this section discusses these required modifications in more detail.

5.1 Knowledge of Task Release Offsets

In our earlier work [8], no task release offsets were considered in the analysis. In order to account for known offsets, the read interval needs to be adjusted. Given an offset Ψ, a job of a task can now read its input data only after Ψ time units after the start of its period. The end of the read interval is unchanged at C time units before the next period starts. Since D_{min} and D_{max} are described by R_{min} and R_{max}, no direct changes are required in the formulation.

5.2 Reachability in known Schedules

Many real-time systems deploy time-triggered schedules in order to guarantee a deterministic system behavior. In such a schedule it is known at design time when the different jobs of the different tasks are executed. Thus, a complete knowledge of the system is available. On the other hand, for dynamically scheduled systems it is often possible to compute the Worst-Case Response Time (WCRT). In that case the exact execution times of a task are not known but the earliest and latest time a task can execute is known.

5.2.1 Schedule is Available

Let's assume an offline schedule is available for the system. So for each job $\tau_{i,j}$ of the task set its exact start time is known as $start_{i,j}$, and similarly its finishing time is known as $end_{i,j}$, see Fig. 3. With this additional knowledge the read and data interval can be adjusted as shown in Table 1.

Since the start of the jobs execution, and hence the time it reads its input data, is known, the read interval collapses to a point. This also leads to smaller data intervals, resulting to no overlap between consecutive jobs.

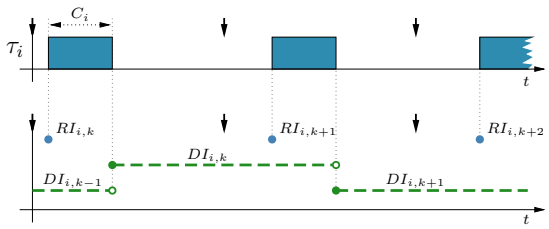


Figure 3: Read and data intervals of consecutive jobs of τ_i if the exact schedule is available.

5.2.2 Worst Case Response Time is Available

For systems where the WCRT of a task τ_i is known as $WCRT_i$, the read and data interval can be adjusted to account for this more accurate system information (see Fig. 4). The modifications of the read interval mainly reflect the possible execution of a job during its execution window (bounded by the WCRT).

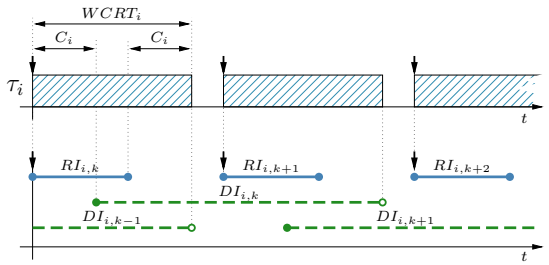


Figure 4: Read and data intervals of consecutive jobs of τ_i if WCRT of the tasks are available.

5.3 Reachability in the LET model

The LET model provides an abstraction to the system designer by temporally decoupling the communication among tasks from the tasks execution. In this model, the input values of a task are always read at the release of the task. The

output values become available once the next period starts. In Fig. 5 these points are highlighted by the thick orange line below the arrows marking the job releases. This temporal decoupling of communication and execution has significant advantages for the end-to-end delay calculations.

The periodic access to all input variables at the beginning of the period collapses the read interval to a point. The data interval is also defined, making the output data available for exactly the period after the jobs execution. These modifications are shown in Table 1. As can be seen, all descriptions are independent of the actual execution time of the job.

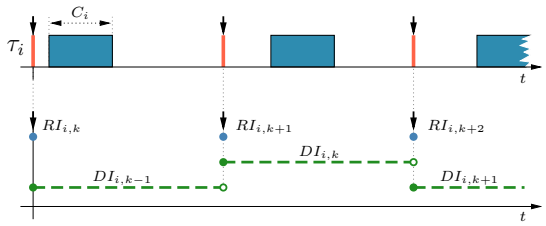


Figure 5: Read and data intervals of consecutive jobs of τ_i if the system operates based on the LET model.

5.4 Discussion

All presented modifications affect solely the read and/or data intervals of the jobs. Hence, the existing calculations for the maximum data age, as presented in [8] and recapitulated in Section 4, can be applied without modification. This fact allows the system designer to perform the calculation of maximum data age during early design phases, where only limited knowledge is present, or during the end of the system design where more complete system knowledge can be obtained.

A tradeoff between required system knowledge and accuracy of the obtained maximum data age exists. For systems with exact knowledge, and for the LET system, it can be observed, that data intervals of different jobs of the same task are never overlapping with each other. This means that it is always certain which data is consumed by a job and thus, all data paths which are computed are observed in the execution of the system.

LEMMA 1. *If it holds for all tasks in a chain, that the read intervals of a task are reduced to a point (i.e. $R_{min}(\tau_{i,j}) = R_{max}(\tau_{i,j})$), then the calculated end-to-end delays are exact. Here "exact" means that all calculated end-to-end delays are observed during the execution of the real system.*

PROOF. From the definition of the read- and data-interval in Section 5 we can see that once $R_{min}(\tau_{i,j}) = R_{max}(\tau_{i,j})$ the resulting data intervals of consecutive jobs are not overlapping. Given that data intervals are not overlapping and read intervals are reduced to points, the function Follows$(\tau_{i,j}, \tau_{k,l})$ only returns *true* for the jobs which actually consume the respective data during the execution of the real system. □

6. EVALUATION

This section presents the evaluation of the proposed approaches to analyze the end-to-end delay based on various levels of system knowledge. First the computed worst-case data-age based on various information levels is compared. Further the required computation time to perform the analysis at the presented information states are evaluated.

6.1 Experimental Setup

The analyzed cause-effect chains for the experiments are generated according to the automotive benchmarks described in [14]. The task periods are uniformly selected out of the set $\{1, 2, 5, 10, 20, 50, 100, 200, 1000\}ms$. The individual task utilization is computed by UUnifast [9]. As stated in [14], an individual cause-effect chain is comprised of either 2 or 3 different periods, where tasks of the same period appear in sequence in the chain. Note that not all period-pairs are valid predecessors in a cause-effect chain [14], which is accounted for during the random generation of the cause-effect chain. For each of the presented data points *1000 random cause-effect chains* are examined.

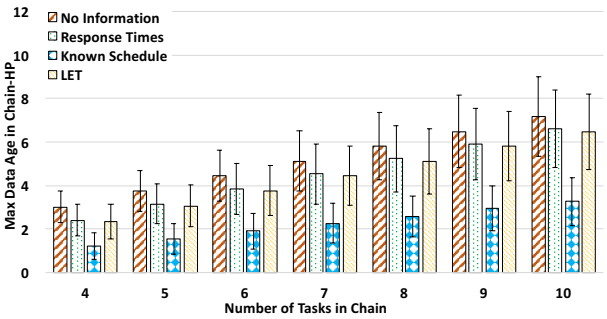
Fixed priority scheduling is used as scheduling algorithm in order to compute the response times and the information for the exact schedule of the tasks. Priorities are assigned based on the Rate Monotonic [15] policy, where priorities between tasks of the same period are assigned in arbitrary order. For the evaluation of the systems with required response times or known schedule, the response times are calculated based on the well-known analysis for task scheduling presented in [13], and the schedule is generated by simulation of the tasks execution.

6.2 Pessimism during the Individual Design Phases

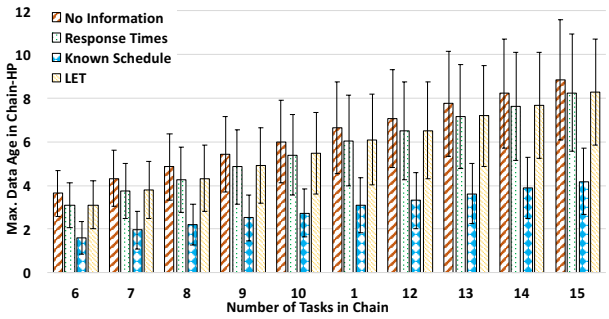
The first experiment examines systems under all four presented information states, *no information*, *response times*, *known schedule*, and *LET model*. The same cause-effect chain is analyzed with increased available system information. The system contains 30 tasks while the cause-effect chain is comprised of 4 to 10 tasks in the case of two activation patterns (i.e. periods), and 6 to 15 tasks in the case of three activation patterns and the system utilization is set to 80%. The results are presented in Fig. 6. The calculated end-to-end latencies are normalized in respect to the chains hyperperiod and shown on the y-axis. The decreased pessimism in the analysis with increased system knowledge is visible for all experiments. The computed worst-case data age of the same scenario includes lesser pessimism from systems with no prior information to systems with known response times up to systems where the schedule is available. Additionally, we present the maximum data age under the LET model, which behaves close to the computed results based on response times in our setting. The difference of the execution semantic becomes visible when comparing with the results for known schedules. Both results are exact results under the respective execution semantic but the observed maximum data age for the LET model is two times as large as the value for the known schedule.

6.3 Analysis of the Computation Time

One of the main improvements behind the presented approaches is to only modify the input set while the analysis framework is unchanged. In this experiment we evaluate the required computation time for the analysis under the different levels of system knowledge. The system contains 30 tasks while the cause-effect chain under analysis has a length of 4 to 10 tasks with two involved activation pattern. All experiments were performed on a system containing an Intel i7 CPU (4 cores at 2,8GHz), and 16GB of RAM. The results are shown in Fig. 7. The two scenarios with exact knowledge (i.e. the known schedule and the LET model) have very



(a) Cause-effect chains with 2 involved periods.



(b) Cause-effect chains with 3 involved periods.

Figure 6: Comparison of the max. data age (normalized to the chains HP) for chains with 2 and 3 activation patterns under various levels of system knowledge.

low analysis times with almost no increase with increasing length of the chain under analysis. On the other hand, the scenarios with less system information experience an exponential increase in analysis time. This can be explained by the increased uncertainty due to overlapping data-intervals, which leads to multiple possible successors which all need to be checked by the algorithm.

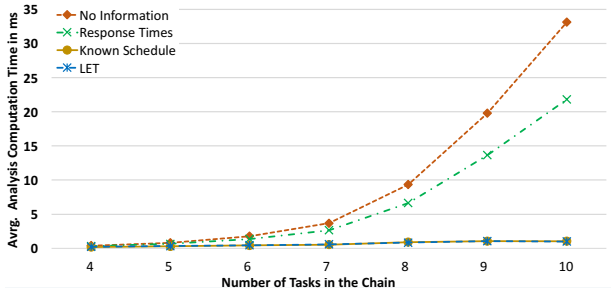


Figure 7: Average analysis time for cause-effect chains with 2 involved periods.

7. CONCLUSION AND OUTLOOK

In this work we have shown how to utilize the different levels of system information available during the design of automotive systems in order to compute the maximum data age of a cause-effect chain. This is done by extending the analysis method presented in [8] by adjusting the read- and data-intervals, which are used as input values of the analysis, to reflect the increase in system knowledge. A clear trade-off can be observed between the required information for the analysis and the pessimism in the obtained results. Future work focuses on the analysis of maximum data age over a cause-effect chain which is distributed over multiple nodes, connected by a network.

Acknowledgment

The work presented in this paper is supported by the Swedish Knowledge Foundation (KKS) through the projects PREMISE, DPAC, and PreView.

8. REFERENCES

[1] IEC 61131-3, 2003.
[2] AUTOSAR - Spec. of Timing Extensions, 2014.
[3] AUTOSAR - Specification of RTE, 2014.
[4] EAST-ADL - Domain Model Specification, 2014.
[5] *AUTOSAR*, last access October 2016. Available at www.autosar.org.
[6] Arcticus Systems. Rubus ICE, [Online] https://www.arcticus-systems.com/products/, last visited 25.10.2016.
[7] M. Ashjaei, S. Mubeen, M. Behnam, L. Almeida, and T. Nolte. End-to-end resource reservations in distributed embedded systems. In *the 22th RTCSA*, pages 1 – 11, 2016.
[8] M. Becker, D. Dasari, S. Mubeen, M. Behnam, and T. Nolte. Synthesizing job-level dependencies for automotive multi-rate effect chains. In *the 22th RTCSA*, pages 159–169, 2016.
[9] E. Bini and G. Buttazzo. Measuring the performance of schedulability tests. *Real-Time Systems Journal*, 30(1-2):129–154, 2005.
[10] M. Broy. Challenges in automotive software engineering. In *the 28th ICSE*, pages 33–42, 2006.
[11] N. Feiertag, K. Richter, J. Norlander, and J. Jonsson. A compositional framework for end-to-end path delay calculation of automotive systems under different path semantics. In *the 1st CRTS*, 2008.
[12] J. Hennig, H. Hasseln, H. Mohammad, S. Resmerita, S. Lukesch, and A. Naderlinger. Towards parallelizing legacy embedded control software using the LET programming paradigm. In *RTAS WiP*, 2016.
[13] M. Joseph and P. Pandya. Finding response times in a real-time system. *The Computer Journal*, 1986.
[14] S. Kramer, D. Ziegenbein, and A. Hamann. Real world automotive benchmarks for free. In *the 6th WATERS*, 2015.
[15] C. L. Liu and J. W. Layland. Scheduling algorithms for multiprogramming in a hard-real-time environment. *Journal of the ACM*, 1973.
[16] S. Mubeen, J. Mäki-Turja, and M. Sjödin. Support for end-to-end response-time and delay analysis in the industrial tool suite: Issues, experiences and a case study. *Computer Science and Information Systems*, 10(1), 2013.
[17] A. C. Rajeev, S. Mohalik, M. G. Dixit, D. B. Chokshi, and S. Ramesh. Schedulability and end-to-end latency in distributed ecu networks: Formal modeling and precise estimation. In *the 10th EMSOFT*, pages 129–138, 2010.
[18] S. Schliecker and R. Ernst. A recursive approach to end-to-end path latency computation in heterogeneous multiprocessor systems. In *the 7th CODES*, pages 433–442, 2009.
[19] F. Stappert, J. Jonsson, J. Mottok, and R. Johansson. A design framework for end-to-end timing constrained automotive applications. In *the 2nd ERTS*.
[20] Symtavision GmbH. SymTA/S and TraceAnalyzer for ECUs, [Online] https://www.symtavision.com/products/ecu-timing/, last visited 25.10.2016.
[21] Timing Architects. Timing Architects Inspector, [Online] https://www.timing-architects.com/ta-tool-suite/inspector/, last visited 25.10.2016.