Jitter Considerations for Worst-Case Performance Generation in Digital Controller Design

Tobias Bund, Steffen Moser, Steffen Kollmann and Frank Slomka
Institute of Embedded Systems/Real-Time Systems
Ulm University
Email: {tobias.bund | steffen.moser | steffen.kollmann | frank.slomka}@uni-ulm.de

***Abstract*—One domain of cyber-physical systems are distributed control systems. The requirements for a control system are correctness, stability and control performance. One issue when designing such a controller is that the timing behavior of the system architecture has a direct impact on the control performance. In this paper, we present an approach which allows us to consider the timing during the controller design flow. Further we propose an approach which allows us to include the maximal occurring jitter in a control loop. Based on this, we propose a new co-design method of real-time and control systems. The controller is designed based on the results of the average delay estimated by the real-time simulation to reach best performance in most cases. It is then verified with the worst- and best-case end-to-end delay bounds calculated by real-time analysis methods. In the last step, the worst control performance regarding the jitter is evaluated by constructing a bad-delay sequence. This approach combines real-time simulation and analysis for best controller design and verification under worst-case timing.**

I. Introduction

As the requirements of embedded systems are increasing, we can see more and more complex architectures consisting of many inter-networked electronic control units (ECUs). Taking the automotive industry for example, we can find up to seventy ECUs in a modern upper-class vehicle. Each ECU consists of various subcomponents like general and application-specific processing units, communication networks and memory architectures. An ECU can be a host for multiple tasks which bring the challenge of finding a suitable task schedule. All these degrees of freedom result in a large design space whereof the system engineer has to choose a solution which must fulfill all requirements that have been specified for the system.

One possibility to avoid problems when dealing with complex architectures and timing issues are to design an oversized platform. In oversized platforms, consisting of more communication and computation resources, there is in general less influence among tasks regarding their timing behavior. In most embedded systems, especially in the automotive domain, there is a trend in reducing costs and energy by resource sharing. This results in a more efficient usage of the available resources and bandwidths, but introduces a non-determinism in the response times of tasks. Additionally, there are future trends in embedded systems like multi-core architectures that lead to timing effects, caused by task migration, that need to be considered.

One commonly required task in embedded systems is controlling a physical system. This results in real-time constraints, which, if violated, can not only have a negative impact on the control performance but may also let the system fails completely. In embedded systems it is not uncommon that tasks of a controller, like gathering sensors values, calculating the actuating signal and driving the actuators are distributed over more than one ECU. This is, for example, the case if one sensor value is needed for more than one technical process. For safety-critical systems it is necessary to know that the system behaves at any time as classified. For example, in a car suspension control system a poor controller performance has direct influence on car maneuverability and therefore car stability.

A goal in the design of distributed closed-loop control systems is to achieve that the system remains stable and stays in the desired performance bounds for all possible disturbances and timing effects. Hence it is necessary to consider the underlying architecture in the controller design. So the relevant timing effects, the average and worst-case delays and the jitter of the delay have to be extracted. In the literature the jitter is mainly described statistically, but in this paper, a special attention is devoted the worst-case jitter behavior.

Finally a new controller design flow is presented that combines the classical controller design with real-time simulation and analysis, extended with worst-case jitter behavior. The main idea behind this design flow is to design the controller with the knowledge of the average delay that appears in the control loop. In the next step, the design is verified against the worst-case conditions.

The structure of the paper is as follow: In the next section we present the problem of jitter in a closed-loop control system and the work that has been done in this field of research. Section three describes our system model, containing the controller and the plant as well as the model, needed for real-time analysis. Based on the results of real-time analysis and simulation, we define the model of timing effects, that are relevant for our method. The next section discusses the influence of jitter on closed-loop control systems. In the fifth section, we present a design flow for an embedded controller by exploiting the described jitter transformation. Section six illustrates our approach in an example, followed by a conclusion.

II. Problem Formulation and Related Work

In complex cyber-physical systems it is very common to have tasks which share resources for computation as well as for communication. This creates dependencies among tasks which may lead to delays when tasks cannot be executed or messages cannot be sent while the resources are busy. These delays can occur in a constant and in a varying manner during the run-time of the system. Differences among two consecutive time-varying delays are called jitter.

In the following, we will mainly focus on the jitter effects on control loops. We assume that the control loop is distributed over the system and uses shared resources which lead to both, constant and time-varying delays in the communication with the controller. In controller design, two main criteria are important: stability and control performance. We will focus on the control performance in the following.

Control performance can be defined in several ways, the most common definitions are:

- as maximum overshoot of the step response
- as phase margin
- as integral of absolute error (IAE) or integral of squared error (ISE)
- as settling time in the step response

The control performance can be seen as a meter to evaluate the quality of a controller. While still being stable, a controller can show a poor performance which means, for example, too much stress for a mechanical system, too high energy consumption or even a safety risk. Both, stability and performance are influenced by the constant delay times and the jitter which occurs in the system. In many applications a poor control performance is not acceptable, so there is a strong need for a method which allows to derive the worst performance that can be expected for a given dead time and jitter.

A control loop which suffers only from a constant delay can be handled much easier than jitter. A constant delay, also called dead time in the domain of control theory, leads to a degradation of the control performance, but it can easily be considered in the design step of a controller because the system remains time-invariant. Therefore, the methods for controller design remain valid. Thus, there exist some well-founded approaches that offer possibilities to integrate the constant delay into stability and performance considerations, when developing a controller application. One approache which allow to cope with dead time is the Nyquist stability criterion [1], a simple graphical stability criterion, where the gain and phase of a frequency response of the open control loop are plotted. Out of this Nyquist plot, a phase margin can be read that provides an amount of dead time which can be added to a closed-loop control system before it becomes unstable.

Based on the Nyquist criterion and the small gain theory, Kao and Lincoln defined the jitter margin which is a simple stability criterion for systems with time-varying delays in [2]. Cervin et al. further improved this stability criterion in [3] by splitting the time-varying delay in a constant delay and jitter, resulting in a less pessimistic test. But it is, as stated above, not always sufficient to know that the system behaves stable in all occurrences of dead time and jitter.

An approach to consider the effects of jitter to the control performance is presented by Lincoln and Cervin in [4]. They developed the Matlab toolbox Jitterbug which is based on the ISE performance criterion. The closed-loop digital control system with delays is modeled as a Markov Jump Linear System. The evolution of this system with a time-grain leads to a corresponding covariance of the state, from which the costs can be calculated. Due to the stochastic manner, this approach is not sufficient for deriving the worst-case performance and therefore does not help to analyze the signal behavior in the worst-case scenario.

Alternative approaches are jitter compensation techniques as published by Lincoln in [5], where additional information in the form of time stamps has to be propagated through the system, leading to a higher utilization. Another approach is the controller parameter adaption, based on the estimated jitter. A method to avoid jitter effects in a control loop is the usage of a time-triggered communication method, as used by Goswami et al in [6], where sample and actuate only act on constant sample times with a delay of one sampling period. For systems with limited resources, where the sensor value can not be sampled at high rates, a delay of one sample time can lead to unacceptable control performances. Additionally, a clock is needed in the actuator that needs to be synchronized.

The above described approaches do not allow to derive the jitter-caused worst-case performance of the controller or they require additional effort to compensate the jitter. Our approach is to construct a delay sequence for given bounds with knowledge of the control signal sequence. This "bad-delay-sequence" causes the worst possible control performance. In the next section, we will introduce a system model which is the basis for a method that will provide a solution for this problem.

III. System Model

To analyze the performance, behavior and stability of a closed-loop control system, we need to model the system. A common platform for control modeling and simulation is Matlab/Simulink. The plant that needs to be controlled consists of continuous blocks, which represent a differential equation. Another way to model a physical system would be in the form of a transfer function in laplace domain.

The architecture surrounding the plant is modeled by blocks out of the SimEvents [7] toolbox. This is due to the event-based simulation engine SimEvents provides, following the discrete event-based behavior of networking elements and computation units. SimEvents blocks contain queuing, routing and serving elements to model delays according to the effects in a distributed architecture or in a multi-core environment. The controller itself is built out of discrete blocks, as it is implemented on a computation unit. In sum, the described model guarantees a flexible and transparent way to simulate

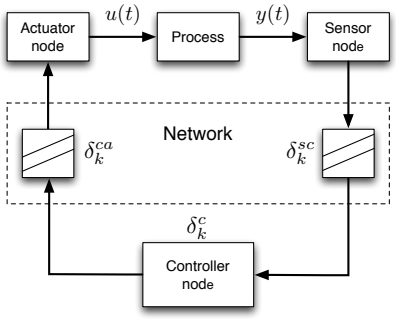

Fig. 1. Model of a distributed closed-loop control system with network delays (δ_k^{ca}) and (δ_k^{sc}) and computational delay (δ_k^c) based on [10].

the system with maximum relation to real behavior. From this model, we can derive a graph, representing the system model. A more detailed description about this approach can be found in [8].

Real-time systems are modeled, including tasks, holding a best-case and a worst-case execution time. These tasks are mapped to a resource, where their execution order is determined by their priority. A resource can be a processor or a network with a limited capacity. The tasks are linked to event chains according to their logical execution order and stimulated by sources, specifying an amount of events per time units as shown in [9]. It is also possible to generate a graph representation of the real-time model (a task graph) as given in [8].

In [8], also a way is described to link a functional Matlab/Simulink model to the real-time analysis by using a bijective mapping. This is sufficient in most cases, as it regards only constant delays in the control loop. As we will see in this paper, we have to expand this to simulation results and to the amount of jitter that can occur.

Figure 1 shows an example of a closed-loop control system with several sources of delays (δ_k), where (δ_k) may vary within every k-th sample. In the presented system model, the sensor samples the data ($y(t)$) from the plant with a constant period (h).

Thereby, we assume an ideal sensor with an ideal clock (no sampling jitter and no clock drift) that samples the sensor value at fix sample times ($t_k = k \cdot h$) with $k \in \mathbb{N}$ and a constant sample period ($h = t_{k+1} - t_k$). The following blocks are event activated. This means, whenever a block finishes its execution, or the time allocated to this block expires, the next block starts its execution. At the end of the chain, an actuator transforms the control sequence (u_k) in a stepwise constant continuous signal (sample-and-hold). As mentioned above, the delays (δ_k) are caused by the execution times and the blocking times of the controller task, plus the delays in the network communication.

The next section describes how to calculate valid response-time bounds based on real-time verification methods.

A. Calculating Time Delays

As mentioned, a time delay in a closed-loop control system has a direct impact on the control performance. In embedded systems these time delays are mainly induced by tasks (τ_i) which do not hold resources exclusively. Therefore, tasks or messages can be blocked or interrupted by other tasks or messages which lead then to time delays which are denoted as response times ($r(\tau_i)$).

In early design phases two possibilities are available to determine these times. The first one is to simulate the system and to extract from the trace files the response times ($r(\tau_i)$). Another possibility is to perform a real-time analysis delivering absolute bounds for the worst-case ($r^+(\tau_i)$) and best-case ($r^-(\tau_i)$) response times.

The simulation has the advantage that the average behavior of the system is considered. From this the average response and jitter times of a closed-loop control system can be derived. The great disadvantage of the technique is that corner cases are not necessarily considered, since a simulation has the typical coverage problems. Therefore analytical methods have to be used, able to calculate bounds for the worst-case and best-case response times in the system. One popular method is the real-time calculus [9]. The idea is to calculate bounds for the time behavior based on the min/plus and max/plus algebra.

For this paper the INCHRON Tool-Suite 2.3.0 [11] is used which enables the possibility to apply both techniques: simulation and validation. Consequently the condition $r^-(\tau_i) \leq r(\tau_i) \leq r^+(\tau_i)$ holds and allows it to explore the whole time behavior of a system.

As we will show in section V this condition can be used by a Matlab/Simulink model to analyze different scenarios. Since both approaches cover the possibility to calculate the end-to-end delay in a closed-loop control system, it is possible to consider the maximum, minimum and average end-to-end delay. Hence, one scenario could be to explore the closed loop with maximum or minimum delay. Another scenario is to consider a variable delay between maximum and minimum end-to-end delay with arbitrary probability distribution functions and therefore the effect of jitter.

But the question arises, how to create the worst-case performance for a given controller using the results of the real-time tools. As we will show in the following sections, the worst-case performance regarding the jitter can be created by a mixture of timing, functional behavior and knowledge of the chosen performance criteria and therefore the pure real-time tools results of a system are not sufficient.

When dealing with jitter, we have to differentiate between sampling jitter and sampling-actuating jitter. Sampling jitter is caused by different values for the sampling period, which does not occur in our system model. We only focus on the sampling-actuating jitter, that is caused by different values for the time delay. For controller applications it is adequate, to identify the response time of tasks in the event-chain of the control loop. Thus, there exist multiple jitter sources in the signal flow of a distributed controller. Assuming, the control loop is not nested or chained in any way, the jitter sources can be summed

together to one jitter in the closed loop. The maximum jitter of a task (τ_i) is defined as $j_i = r^+(\tau_i) - r^-(\tau_i)$. The summed maximum and minimum end-to-end delays can be described as $\delta_{min} = \sum_i r^-(\tau_i)$ and $\delta_{max} = \sum_i r^+(\tau_i)$ for all tasks (τ_i) in the control loop event chain. Thus the maximum round-trip jitter, that can occur is $j_{max} = \delta_{max} - \delta_{min}$.

IV. Influence of Jitter on Closed-Loop Control

In this section, we present a method capable of identifying the delay sequence that causes the worst performance degradation. We focus in our consideration on the maximum overshoot on a step response as performance criteria. This is a suitable criteria, because it is intuitive to identify in the plot of the control signal and relevant for many real-life systems like car suspension control. Thereby we assume a sufficiently small maximum end-to-end delay $(\delta_{max} < h)$, which means no sampled value gets lost.

To identify the maximum overshoot of a closed-loop control system when influenced by a reference value or a specific disturbance, we need three kind of information from our system. First of all we need the actuating signal sequence (u_k) that affects the system. This sequence can be determined out of a simulation or a measurement of the controlled system. Further we need to know the behavior of the system on an impulse on the actuating signal. The sequence of the so called impulse response is denoted as (ϕ_k). Finally the time step $(m \cdot h)$, when the overshoot occurs must be known.

As the end-to-end delay from sensor to actuator can vary between (δ_{min}) and (δ_{max}), the time when the actuator begins to affects the process lies between $(t_k + \delta_{min})$ and $(t_k + \delta_{max})$. This means, a time varying end-to-end delay modifies the duration, on which an actuating signal (u_k) affects the process. The maximum duration on the actuator is given as $(h + j_{max})$ and the minimum as $(h - j_{max})$. Figure 2 displays an example, where the actuating signal is extended to the duration of $(h + j_{max})$. As we can see, the maximum additional impulse on the system is a function of the maximum possible jitter and the step size between two consecutive actuating signals, e.g. (u_k) and (u_{k+1}).

In our approach, we take advantage of this influence on the duration of the actuating signal to manipulate the process in a manner to construct worst performance or the highest overshoot respectively. The result is of course an improbable but nevertheless possible scenario.

To reach worst performance, we modify the duration of the actuating signal with the maximum amount of absolute jitter that can occur, resulting in the most possible expansion or reduction of the actuating signal duration. When expanding the actuating signal by (j_{max}), one either can retain the duration of the next actuating signal as the sample period (h), or reduce it by (j_{max}).

The choice, which sample to delay and which to shorten by the maximum jitter depends on two factors. The first one is the difference between two consecutive actuating signals defined as $\tilde{u}_k = u_{k+1} - u_k$. The second factor is the impact of the error, made by jitter in the future, when the overshoot appears. This

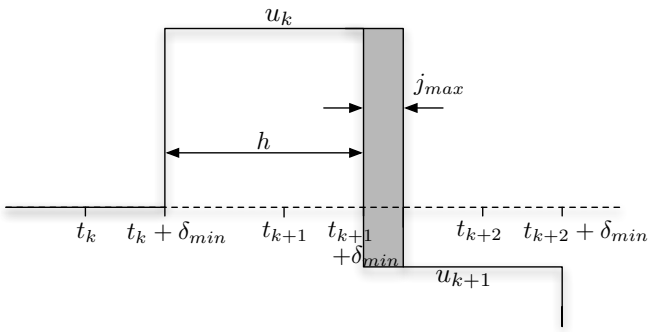

Fig. 2. Additional impulse on the system, caused by the additional delay of (j_{max}).

is done by the above described impulse response. Since the error made by the maximum jitter (grey area in figure 2) has a sufficiently short duration in relation to the sample period, it can be seen as an impulse on the system. This impulse is weighted by $(m - k)$-th element of the above described impulse response sequence (ϕ_{m-k}), to get the influence from an expansion on k-th actuating signal on the overshoot at m-th sample time.

If this procedure is done for all possible sample times up to $(m \cdot h)$, the result is a sequence that identifies the impact of (j_{max}) on the overshoot for all sample times. By expanding the actuating signal duration on the local maximums of this sequence by the maximum jitter and shorten the duration for the local minimums, the bad delay sequence is defined.

V. Controller Design Flow

In this section, we propose a design flow for an embedded controller by considering time-varying delays based on the idea presented in section IV.

According to figure 3, the system engineer defines the platform design of the distributed system at the design entrance. On the other side, the task architecture, that defines the mapping between functionality and software tasks is derived from the controller design and other functions in the system. By mapping the task architecture to the hardware platform, the system architecture is specified. In this step, the scheduling is assigned to the tasks.

In the next step, suitable information about the occurring end-to-end delays of the messages that are exchanged by the components of the distributed controller has to be acquired. These delays are variable during the run-time of the system within an interval of best-case and worst-case end-to-end-delay.

To get the best control performance over the run-time of the system, the designer has to optimize the controller and to parametrize its settings in a way that holds best performance at the average end-to-end delay that occurs in the system. Therefore, the average end-to-end delay has to be known in advance of the controller design. It can be obtained from a simulation of the system's real-time behavior which has been described in section III-A.

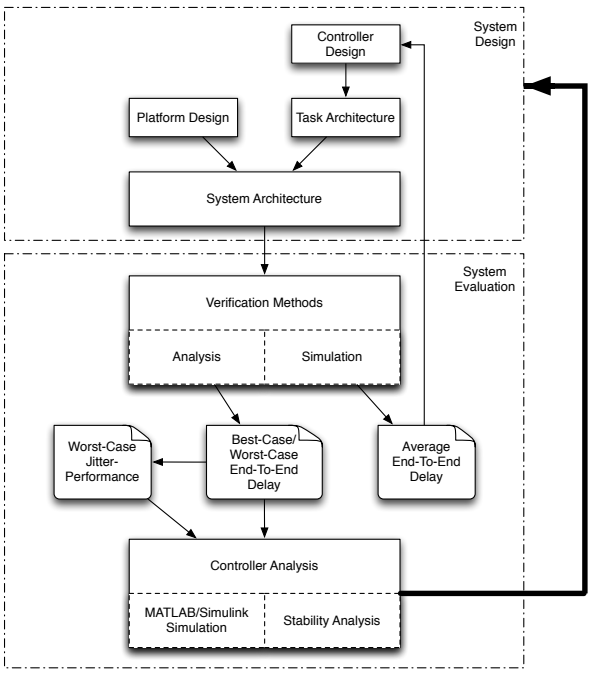

Fig. 3. Proposed new design flow for an embedded controller to consider the effect of time-varying delays to control performance.

While this leads to a controller that delivers its optimum performance in the average case, we must consider both, stability and performance for delay times higher or lower than the average. But this is not enough: We have also to consider the influence of the end-to-end delay jitter.

At the first step, we focus on the minimum and maximum end-to-end delays which can occur in the system. The real-time simulation we used in the previous step to gather the average end-to-end delay shows the coverage problem which is quite common for simulations, which means that simulations do not necessarily find the best and worst case end-to-end delays. This makes clear that we cannot rely on the simulation only. Therefore we make use of deterministic real-time analysis to get the absolutely best- and the worst-case end-to-end delays which can occur in the system. By using this information, we simulate our controller to check if it holds the stability and performance requirements even for the absolute minimum and maximum end-to-end delays.

As previously discussed, not only constant end-to-end delays affect the stability and performance of a controller, they rather suffer much stronger from time-varying end-to-end delays jittering between the maximum and minimum value. Based on the method we present in section IV, we identify the performance degradation due to jitter.

If the controller analysis does not hold the requirements, the system design, including the controller, platform and task architecture and even the scheduling have to be adjusted, which leads to a re-design phase. If system specifications are violated, the system design needs to be redesigned.

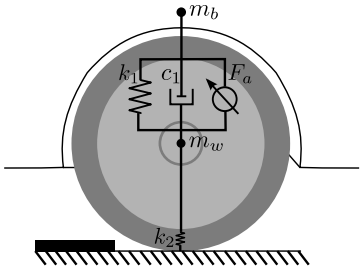

Fig. 4. Model of active car suspension based on [12].

VI. Example

In our example, we demonstrate our approach on an active car suspension control. An active car suspension control is implemented for comfort and for car stability reasons. To affect the chassis dynamic, a linear electric motor is placed between wheel and chassis. In [12] a simplified model of a suspension control is proposed and the mathematic equations of motion are derived. The model consists of a spring-mass-damper system and a linear motor, modeled as a force (F_a) between chassis and wheel (see figure 4). The wheel itself is modelled as a spring to the ground. As the model covers only a quarter of the car, the mass (m_b) denotes a quarter of the car's mass and (m_w) the mass of one wheel. The dynamics can be described in the form of a linear state-space model

$$\dot{x}(t) = Ax(t) + Bu(t) \tag{1}$$

with parameters

$$A = \begin{pmatrix} 0 & 0 & 1 & -1 \\ 0 & 0 & 0 & 1 \\ -\frac{k_1}{m_b} & 0 & -\frac{c_1}{m_b} & \frac{c_1}{m_b} \\ \frac{k_1}{m_w} & -\frac{k_2}{m_w} & \frac{c_1}{m_w} & -\frac{c_1}{m_w} \end{pmatrix},$$

$$B = \begin{pmatrix} 0 & 0 & \frac{1}{m_b} & -\frac{1}{m_w} \end{pmatrix}^T.$$

Thereby the system input is denoted as (u) and the state vector as (x). A detailed description of the states can be found in [12].

The controller is designed as a digital control. Therefore equation 1 is discretized by the zero-order hold model, analogous to a digital-to-analog converter. The control system is then in the form of the following difference equation

$$x(k+1) = Ax(k) + Bu(k) \tag{2}$$

with state vector $x(k)$ including n state variables. The according state feedback

$$u(k) = -Kx(k) \tag{3}$$

is calculated by pole placement.

Our control objective is to minimize the difference between road and wheel ($z_w - z_r$) after a disturbance on the road in form of a stepwise displacement, as displayed in figure 4. The smaller the difference ($z_w - z_r$), the better the car handling in the case of a disturbance. Hence, car suspension control is a safety-critical system in the automotive domain.

$\overline{\delta}$	δ_{min}	δ_{max}	j_{max}
$0.636\,mm$	$0.757\,mm$	$3.790\,mm$	$4.974\,mm$

TABLE I
OVERSHOOT CAUSED BY DELAY

Based on the controller design with a sample time of $10\,ms$ and the surrounding functionality, a task architecture is derived. The platform architecture consists of three ECUs, interconnected by a CAN bus. In our case, the messages assigned to the control loop have the lowest priority on the CAN bus. Hence, a system architecture is defined with a bus speed of $500\,kbit/s$ resulting in an overall utilization of $25.88\,\%$.

The described real-time system was simulated for a duration of $3600\,s$, leveling off a distribution of the end-to-end delay. The average end-to-end delay ($\overline{\delta}$) can be calculated out of the histogram as an arithmetic mean value.

Based on the average end-to-end delay $\overline{\delta} = 1786\,\mu s$, an improved controller with optimized parameter setting is designed. This is done by an improved model of the system, including the average end-to-end delay ($\overline{\delta}$). A consequence, when considering $\overline{\delta} < h$ in the controller design is that the state vector $(x(k+1))$ does not only depend on the actual actuator signal $(u(k))$, but also on the previous one $(u(k-1))$. This is done by expanding the state space vector with the additional state variable

$$x_{n+1}(k) = u(k-1). \tag{4}$$

We obtain the new differential equation

$$\begin{pmatrix} x(k+1) \\ x_{n+1}(k+1) \end{pmatrix} = \begin{pmatrix} A & b_{-1} \\ 0' & 0 \end{pmatrix} \begin{pmatrix} x(k) \\ x_{n+1}(k) \end{pmatrix} + \begin{pmatrix} b_0 \\ 1 \end{pmatrix} u_k \tag{5}$$

where the coefficients (b_{-1}) and (b_0) are calculated, as described in [13] and depend on the amount of delay (δ).

For the ascertained average end-to-end delay and a road-displacement of $10\,mm$, the maximum overshoot of $(z_w - z_r)$ is $0.636\,mm$.

According to that, the controller design is verified against the highest possible end-to-end delay $\delta_{max} = 8790\,\mu s$ that possibly can occur. This delay is calculated from real-time analysis methods, described in section III-A, resulting in a step response with an overshoot of $3.79\,mm$. The real-time analysis also identifies an minimum end-to-end delay that causes an overshoot of $0.757\,mm$.

Finally an even worse performance, referring to the maximum overshoot, can be constructed by assuming a jitter alternating between minimum and maximum end-to-end delay. This is done out of the methods described in section IV. The overshoot of the system, when it is delayed with the bad-delay sequence, is $4.974\,mm$.

When the overshoot exceeds our specification, we need to redesign our system by adapting our controller, modifying our platform or assigning a new scheduling.

VII. CONCLUSION

In this paper we presented an approach for a digital controller design by using results from a real-time simulation to determine the average end-to-end delay to design a controller with best performance in most cases. Afterwards, the controller design is verified against the best-case and worst-case end-to-end delays, calculated by real-time analysis methods, to check if the control performances in these points of operation are satisfying. We could also construct an even worse performance by combining real-time analysis results with system characteristics.

To improve the results and push the worst-case performance to a more optimistic and valid direction, the maximum step between two consecutive delays has to be regarded. Or, in other words, is it possible, that a maximum end-to-end delay (δ_{max}) can follow the minimum end-to-end delay (δ_{min}) or vice versa?

One further challenge would be to develop an analysis method that allows to check whether a specific system is more sensitive to a constant delay or rather to jitter. One possibility to reduce the jitter sensitivity would be to increase the sample rate of the controller. This would cause in general smaller differences between two consecutive actuating signals and therefore would reduce the influence of jitter. However, a higher sample rate would possibly result in higher response times. This tradeoff needs to be studied in further research.

REFERENCES

[1] H. Nyquist, "Regeneration Theory," *Bell System Technical Journal*, vol. 11, no. 1, pp. 126–147, 1932.

[2] C.-Y. Kao and B. Lincoln, "Simple Stability Criteria for Systems with Time-Varying Delays," *Automatica*, vol. 40, 2004.

[3] A. Cervin, B. Lincoln, J. Eker, K.-E. Årzén, and G. Buttazzo, "The Jitter Margin and Its Application in the Design of Real-Time Control Systems," in *10th International Conference on Real-Time and Embedded Computing Systems and Applications (RTCSA)*, Göteborg, Sweden, Aug. 2004.

[4] B. Lincoln and A. Cervin, "Jitterbug: A Tool for Analysis of Real-Time Control Performance," in *Proceedings of the 41st IEEE Conference on Decision and Control*, Las Vegas, NV, Dec. 2002.

[5] B. Lincoln, "Jitter Compensation in Digital Control Systems," in *American Control Conference, 2002*, Anchorage, AK, USA, May 2002.

[6] D. Goswami, R. Schneider, and S. Chakraborty, "Co-design of Cyber-Physical Systems via Controllers with Flexible Delay Constraints," in *16th Asia and South Pacific Design Automation Conference (ASP-DAC)*, Yokohama, Japan, 2011.

[7] "SimEvents," http://www.mathworks.com/products/simevents/.

[8] T. Bund, S. Moser, S. Kollmann, and F. Slomka, "Guaranteed Bounds for the Control Performance Evaluation in Distributed System Architectures," in *Proceedings of the International Conference on Real-Time and Embedded Systems (RTES 2010)*, Singapore, Sep. 2010.

[9] E. Wandeler, "Modular Performance Analysis and Interface-Based Design for Embedded Real-Time Systems," Ph.D. dissertation, ETH Zurich, September 2006.

[10] J. Nilsson, "Real-Time Control Systems with Delays," 1998.

[11] "Inchron Tool-Suite 2.3.0," http://www.inchron.com.

[12] K. Hyniova, A. Stribrsky, J. Honcu, and A. Kruczek, "Active Suspension System with Linear Electric Motor," *WSEAS TRANSACTIONS on SYSTEMS*, vol. 8, pp. 278–287, 2009.

[13] M. S. Branicky, S. M. Phillips, and W. Zhang, "Stability of Networked Control Systems: Explicit Analysis of Delay," in *Proceedings of the American Control Conference*, Chicago, Illinois, Jun. 2000.