Fault-tolerant Control Synthesis and Verification of Distributed Embedded Systems

Matthias Kauer[1], Damoon Soudbakhsh[2], Dip Goswami[3], Samarjit Chakraborty[3], Anuradha M. Annaswamy[2]
[1] TUM CREATE, Singapore, matthias.kauer@tum-create.edu.sg
[2] Massachusetts Institute of Technology, USA, {damoon, aanna}@mit.edu
[3] Technische Universität München, Germany, {dip.goswami, samarjit}@tum.de

***Abstract*—We deal with synthesis of distributed embedded control systems closed over a faulty or severely constrained communication network. Such overloaded communication networks are common in cost-sensitive domains such as automotive. Design of such systems aims to meet all deadlines following the traditional notion of schedulability. In this work, we aim to exploit robustness of the controller and propose a novel implementation approach to achieve a tighter design. Toward this, we answer two research questions: (i) given a distributed architecture, how to characterize and formally verify the bound on deadline misses, (ii) given such a bound, how to design a controller such that desired stability and Quality of Control (QoC) requirements are met. We address question (i) by modeling a distributed embedded architecture as a network of Event Count Automata (ECA), and subsequently introducing and formally verifying a property formulation with reduced complexity. We address question (ii) by introducing a novel fault-tolerant control strategy which adjusts the control input at runtime based on the occurrence of fault or drop. We show that QoC under faulty communication improves significantly using the proposed fault-tolerant strategy.**

I. Introduction

Contemporary in-vehicle networks contain Electronic Control Units (ECUs), sensors and actuators that are connected via arbitrated, shared buses such as Ethernet, Controller Area Network (CAN) or FlexRay. Feedback control applications that are running in this context – often referred to as Cyber-physical Systems (CPS') – induce a tight coupling between computational and physical resources. This is typically coordinated by designing CPS' with a fixed pair of deadline and sampling period; it is then ensured that the worst-case sensor-to-actuator message delay on the communication hardware is smaller than this deadline. However, relying on the interface of sampling period and deadline alone is not flexible enough in situations where the communication buses are already highly utilized. This is due to the fact that worst-case delay is typically a rare occurrence triggered only by a very specific sequence of events, and it is possible to design the feedback controllers to be robust to certain amount of message drops without compromising the required control performance. In this work, we propose a design flow that first establishes a formal bound on the message drops for a certain suitable period-deadline pair. We then design a control strategy specifically with this bound in mind that guarantees exponential stability.

A. Related Work

This work is closely related with two broad areas; (i) fault-tolerant control design and (ii) analysis and formal verification of timing properties in embedded architectures.

In control theory literature faults in the feedback loop are often addressed by an appropriate Linear Matrix Inequality (LMI) formulation as in [1], [2] and [3]. In [2], robustness of a feedback control loop is quantified by a fault ratio over an *infinite* horizon. An LMI-based design approach is presented in [3]. The presented strategy assumes an upper bound on the consecutive faults and zero sensor-to-actuator delay. A more recent approach to handle dropped signals in a networked control system is the predictive outage control presented in [4]. This approach is based on predictive form and was implemented using an additional computational block next to the actuator to compute new input signals in case no new messages arrived. The input signal was computed using the previous inputs of the system. In our study, we explicitly design the controllers by considering the nominal delays of the system and the upper bounds on the consecutive missed deadlines. We use this information to design an exponentially stable controller.

On the timing analysis/verification side, [5] and [6] first showed that certain control performance requirements induce scheduling specifications that are ω-regular languages. They then describe the construction of a scheduling automaton where every path leads to a schedule for a single ECU that guarantees all the individual performance requirements. Building on their work, [7] describes how such an ω-regular specification can be used in conjunction with an Event Count Automaton (ECA) model (an automaton-based extension of Real-Time Calculus (RTC) [8]) of the communication architecture to guarantee the overall performance of a CPS. A similar angle is taken in [1] where another extension of RTC to Timed Automata (TAs) and the tool UPPAAL is used to guarantee the stability margin of a control system. Both the ECAs in [7] as well as the TAs from [1] can handle state-based scheduling in a straightforward fashion and allow the integration with RTC. This greatly helps with scalability if state-dependent scheduling is only employed in certain subnetworks. Their main difference is the trade-off between time-granularity and scalability of the modeling. The ECA framework directly incorporates the jitter into a coarser view of the architecture, leading to faster verification of typical streaming systems. Finally, a technique that relies on RTC alone – and hence trades in state-based scheduling for better scalability – has been developed to analyze worst-case failure patterns in [9]. The authors combine this with an LMI-based test for exponential stability, but no design method.

B. Contribution

Our contributions are two-fold. For a given ECA model of a distributed embedded architecture, we propose an evaluation method to *verify* an upper bound on missed deadlines. Our model-checking framework is able to verify the bound with deadlines shorter than the sampling period. Since most designs in safety-critical domains do not allow longer deadlines, the

This work was financially supported in part by the Singapore National Research Foundation under its Campus for Research Excellence And Technological Enterprise (CREATE) programme.

This work was supported in part by the NSF Grant No. ECCS-1135815 via the CPS initiative.

978-3-9815370-2-4/DATE14/©2014 EDAA

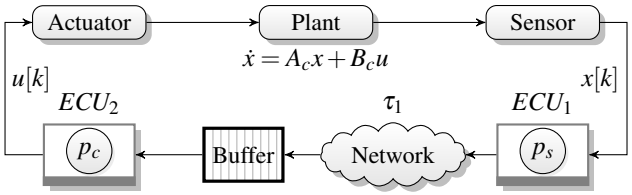


Fig. 1: Distributed control system under consideration.

above framework is of practical importance in such setups. Our proposed method reduces the size of the state space and hence improves the performance of the model checking framework compared to the previous works in this direction. Furthermore, we propose an LMI-based formulation for fault-tolerant control design that takes into account the bound guaranteed by the model checking framework. Our control strategy observes deadlines – that can be smaller than sampling period – at runtime and adjusts the gains with respect to violations. This new control design and implementation algorithm significantly improved the system's stability and reduced the required input over traditional methods such as Zero-Order Hold (ZOH). Furthermore, instead of testing and verifying the performance of a given control law as in previous attempts [7], [9], [1], our framework designs a controller for a given architecture.

II. Problem Description

We consider a continuous linear time-invariant plant closed over a highly utilized network as shown in Fig. 1. Since the communication is initially unbuffered, the feedback loop experiences a time-varying delay τ which leads to the continuous time dynamics

$$\dot{x}(t) = A_c x(t) + B_c u(t-\tau), \tag{1}$$

where $A_c \in \mathbb{R}^{n\times n}$, $B_c \in \mathbb{R}^{n\times p}$ are continuous system and input matrices; x, u denote the state variables and the control input respectively.

This control system is divided into two main tasks, sensor task p_s and controller task p_c, and implemented in a distributed fashion (see Fig. 1). On ECU_1, state $x[k]$ is read from the sensor periodically. p_s further processes the reading and sends it out over the network to ECU_2 as a message. After the message is transmitted over the network, it is stored in the buffer of ECU_2. On ECU_2, p_c is triggered periodically. It checks whether a message has arrived in its buffer, calculates input signal $u[k]$ and adjusts the actuator accordingly.

The timing of this distributed system is detailed in Fig. 2. τ_1 is the variable time interval from sensor reading until the message arrives at the buffer of ECU_2. Controller task p_c then needs time τ_2 to calculate $u[k]$ and apply the input signal via the actuator. Sensor and actuator are triggered periodically with period h and a relative offset of τ_{th}. Therefore, τ_1 must not exceed $\tau_{th}-\tau_2$ for a successful actuation. A message is considered to be *dropped* when $\tau_1 > (\tau_{th}-\tau_2)$. Thus, control system (1) with variable delay τ can be considered as a *discrete-time sampled-data system* with period h and constant sensor-to-actuator delay τ_{th}.

$$x[k+1] = Ax[k] + B_0 u[k] + B_1 u[k-1] \tag{2}$$

where A, B_0, B_1 are given by (3).

$$A = e^{A_c \cdot h},\ B_0 = \int_0^{h-\tau_{th}} e^{A\cdot v}\,\mathrm{d}v B_c,\ B_1 = \int_{h-\tau_{th}}^{h} e^{A\cdot v}\,\mathrm{d}v B_c \tag{3}$$

In this work, we aim to guarantee Quality of Control (QoC) in the form of exponential stability (Definition 2) under the presence

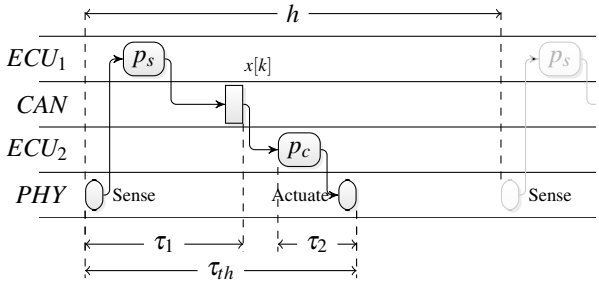


Fig. 2: Timing Diagram of the Distributed System as in Fig. 1. Controller task p_c requires constant time τ_2. A fault occurs if variable delay $\tau_1 > \tau_{th} - \tau_2$.

of some dropped messages. However, package drops degrade QoC and potentially render the system unstable. Hence, messages cannot be dropped too frequently. We quantify the permissible amount of lost signals with the notion of (m,k)-firm deadline (Definition 1). For a given $(m+1,1)$-firmness, we need to adapt our control strategy to ensure exponential stability (shown in Section IV).

Definition 1 ((m,k)-firm Deadline): A message stream meets an (m,k)-firm deadline of delay τ_{th} if over any window of m consecutive messages at least k arrive within τ_{th} time.

Definition 2 (Global Exponential Stability): Equilibrium point x_0 of system (1) is said to be globally exponentially stable if there exist $C_1, \gamma > 0$ such that

$$\|x(t)\| \le C_1 \|x_0 - x(t_0)\| \cdot e^{-\gamma(t-t_0)}$$

holds for all $t \ge t_0$. Equivalently, the system is globally exponentially stable if there are suitable constants $0 < \hat{C}, 0 < \hat{\gamma} < 1$, such that

$$\|x[k]\| \le \hat{C}_1 \hat{\gamma}^k \|x_0 - x[k_0]\|$$

for $k \ge k_0$ is fulfilled for its discrete-time representation (2).

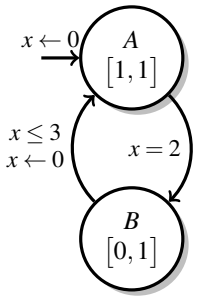
III. Event Count Automata

In this section we describe how to obtain the upper bound on the deadline misses that we need to design a suitable control strategy. We model the underlying hardware architecture of the system as a network of Event Count Automata (ECAs), similar to [7]. We briefly describe this in Section III-A. Since, we are only interested in an upper bound on the successive deadline misses, we can significantly simplify the time stamps and the corresponding Linear Temporal Logic (LTL) formulation (Section III-B). Finally, we use the model-checking tool Symbolic Analysis Laboratory (SAL) [10] to obtain the bound from the established model.

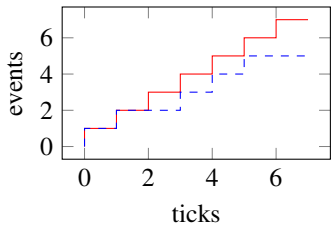
A. Networks of Event Count Automata (ECAs)

Individual ECAs. An individual ECA describes the arrival of events or the available service of a Processing Element (PE). Every *state* $s \in S$ of an ECA is characterized by an event rate $\rho(s) = [l,u]$, an interval bounding the possible occurrence of events, the *event count* c. The event count c increases the count variables X and, at the same time, it makes up the output of the ECA. Transitions with guards such as $x = 1$ or $x \le 1$ and reset actions $x \leftarrow 0$ connect the states to each other. Both the actual selection of the event count c and the transitions are taken non-deterministically – constrained by the rate function and the guards, respectively. This non-determinism in turn has every ECA produce a set of event count sequences

$$\Sigma = \{\sigma = (c_1, c_2, \dots) : \text{ECA movement could produce } \sigma\}$$



(a) Example ECA producing strings $(110+111)^{\omega}$



(b) Upper (solid) and lower (dashed) bound on the events the ECA can produce.

Fig. 3: Example ECA illustrating the framework's basic functionality.

For further illustration, consider the example ECA from Fig. 3(a). Its states A and B have associated rate intervals $\rho(A) = [1,1]$ and $\rho(B) = [0,1]$. Hence, while in A one event per time slot will occur, whereas the slots that the automaton spends in B may see zero or one event. These events affect the single count variable x. The ECA starts in the initial configuration $(s_{in}, V_{in}) = (A,[0])$ where V corresponds to the valuation of the count variable x. It cannot move out of A immediately, so it waits for two slots until $x = 2$ is fulfilled at which point the automaton moves to $(B,[2])$. All transitions are considered urgent by default and we therefore move to $(A,[0])$ immediately – whether another event occurs or not. This behavior can be seen in Fig. 3(b) when the lower bound (dashed) on the possible event count sequences diverges from the upper bound (solid) in the second tick.

Connecting Multiple ECAs. A set of ECAs can be interlinked via buffers and form an ECA network. The general idea is to consider the event count c of an ECA as available service. This service is used to process messages (or generic data units) from input to output buffer. The output buffer is another ECA's input and hence, a stream is formed. The first ECAs are without input buffer and their events model arrivals of the system that are deposited into the first buffers. These automata are also referred to as arrival ECA. Note that for verification purposes, ECAs, including their networks, can be transformed to Nondeterministic Büchi Automata (NBAs) giving access to a multitude of established tools from the digital design domain. A more complete description, with further examples and applications of ECA can be found in [8].

B. Verification of Deadline Firmness

In the following, we propose a method to verify a $(m+1,1)$-firm deadline using an ECA network model. In other words, we develop a test to see if it is guaranteed that no more than m consecutive messages fail to meet their deadline. Our approach requires the following assumptions that are fulfilled in the scenario we are concerned with:

1) Messages in the stream under discussion arrive periodically, possibly with small jitter, but without burst. This is necessary for the arrival automaton's clock to serve as clock for the evaluation automaton.
2) Messages are only overwritten by other messages from their own stream and they cannot bypass each other.
3) Messages have a deadline shorter than their arrival period.

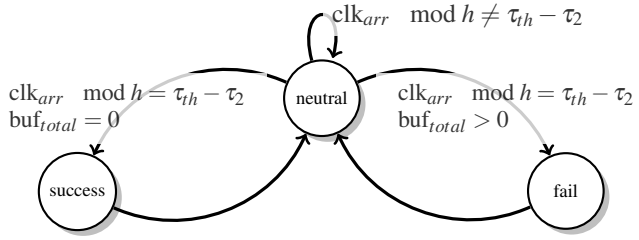


Fig. 4: Evaluation Automaton with inputs clk_{arr}, the clock of the corresponding arrival automaton, $\text{buf}_{total} = \sum_{i:i\in\text{Stream}} b_i$, the total amount of messages inside the stream of interest has inside the system.

Under these assumptions it is sufficient to test that the stream has no more messages inside the system when the clock of the arrival automaton reaches the deadline. When no messages are inside the system, the last message that was sent must have successfully arrived since – being last – it could not have been overwritten.

We implemented this check in the evaluation automaton depicted in Fig. 4. To track the maximum number of uninterrupted deadline misses, we initialize a variable $\text{failcnt} = 0$. We then update it with $(\cdot)^+$ the temporal next operator[1] depending on s, the state of the evaluation automaton.

$$\text{failcnt}^+ = \begin{cases} \text{failcnt}+1 & \text{, if } s = \text{fail} \\ 0 & \text{, if } s = \text{success} \\ \text{failcnt} & \text{, if } s = \text{neutral} \end{cases}$$

To guarantee that there are at most m message drops after every successful transmission – as we require in this work –, it then suffices to model check the simple LTL formulation

$$\mathbf{G}(\text{failcnt} < m)$$

More complicated m,k-firm deadlines as discussed in [7] can be evaluated by introducing a fail count array of size k that is updated in a similar way. We would then have to verify that none of the array's elements exceed m at any time.

IV. Fault-Tolerant Control Design with Delays under Firm Deadline Assumption

As explained in Section II, the controller's task is to stabilize the system and guarantee its required performance with nominal delay τ_{th} and packet drops. Typically, these drops are caused by delay from contending messages and subsequent deadline violation as in Section III-B. They may also occur due to sporadic hardware faults or disturbance from outside the control setup.

Fault-tolerant strategies are usually designed based on systems without delays (see for example [3]). In the following we present Drop Compensation Control (DCC), a control design algorithm using LMI and Switching Control Theory that considers both delays and drops in the system. DCC adjusts the new input signal to lessen the impact of the old input signal. Its design is more involved than for ZOH, but it achieves superior QoC and does not introduce chattering.

A. Zero-Order Hold (ZOH) Actuation Timing

The traditional implementation of a digital controller is a sample and hold block that stores the control signal until the periodically actuated controller finds a new one in its buffer. This

[1] More precisely, if $b = b[k]$ refers to the content of variable b at the end of slot k, $b^+ = b[k+1]$ refers to the content one time step later. This notation integrates nicely with model-checking tools.

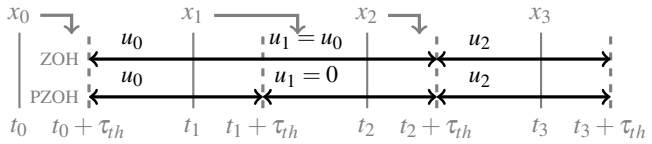


Fig. 5: Activation Sequences for ZOH and PZOH in reaction to two successful (x_0, x_2) and one failed (x_1) transmission. ZOH only changes the input after a successful transmission. PZOH sets the input to zero if a fault is detected.

is referred to as ZOH. An alternative approach that we named Periodic Zero-Order Hold (PZOH) is periodically checking the buffer for fresh state information and only actuates if the last transmission was successful.

$$u[k] = \begin{cases} Kx[k] & \text{, if } \tau_1 \leq \tau_{th} - \tau_2 \\ 0 & \text{, otherwise} \end{cases} \tag{4}$$

Fig. 5 illustrates the difference between these patterns.

B. Drop Compensation Control (DCC) Design

Any controller that is periodically checking its buffer for new information in PZOH fashion (see Fig. 5) can identify dropped messages at runtime. We therefore look for a strategy that compensates for the drops and thereby increases the robustness of the system. Fig. 6 shows a schematic implementation of the proposed drop controller design. The drop controller counts the number of consecutive drops j, i.e., the number of instants where $\tau_1 > \tau_{th} - \tau_2$, since the last successful update. In time step $k+j$, it then actuates the system based on $x[k-1]$, the last state that was transmitted, and $u[k-2]$, the input at that time.

$$u[k+j] = K_j x[k-1] + G_j u[k-2], \;\; j = 0, 1, \ldots, m-1 \tag{5}$$

This signal follows a PZOH actuation pattern, i.e., it is used until the next period recalculation (see Fig. 5). In the following, we will discuss how to select gain matrices $(K_j, G_j)_{j=0,\ldots,m}$ such that exponential stability of the system is guaranteed.

In the *nominal case*, there are no drops and the system is always actuated using (K_0, G_0). Using an extended state $X[k] \stackrel{\text{def}}{=} [x[k] \quad u[k-1]]^T$ we can write this as

$$X[k+1] = \begin{bmatrix} A + B_0 K_0 & B_1 + B_0 G_0 \\ K_0 & G_0 \end{bmatrix} X[k] \stackrel{\text{def}}{=} A_n X[k] \tag{6}$$

In what follows, we develop an algorithm for designing the stabilizing controller in the *drop mode*. For $j = 1$ drops, we have

$$\begin{aligned} x[k+1] =& Ax[k] + B_0 u[k] + B_1 u_{k-1} \\ =& A\big(Ax[k-1] + B_0 u[k-1] + B_1 u[k-2]\big) \\ &+ B_0\big(K_1 x[k-1] + G_1 u[k-2]\big) + B_1 u_{k-1} \end{aligned} \tag{7}$$

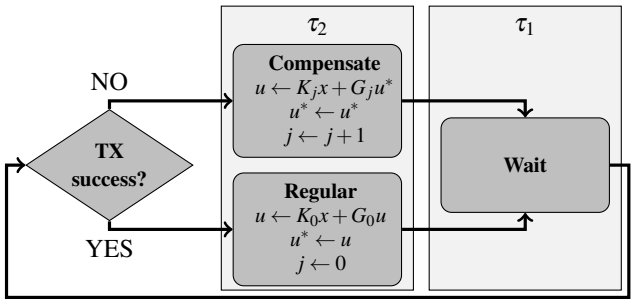


Fig. 6: Schematic representation of Drop Compensation Control (DCC). After a successful transmission (TX), the regular path is taken. If no new information is available, progressively higher compensation gains (K_j, G_j) are utilized.

and hence, using $A_B \stackrel{\text{def}}{=} AB_0 + B_1$, we obtain:

$$X[k+1] = \underbrace{\begin{bmatrix} A^2 + A_B K_0 + B_0 K_1 & AB_1 + A_B G_0 + B_0 G_1 \\ K_1 & G_1 \end{bmatrix}}_{\stackrel{\text{def}}{=} A_d^{(1)}} X[k-1]$$

For a general number of drops j, we iterate this construction, resulting in the following closed loop dynamics

$$X[k+1] = \begin{bmatrix} A_0^{(j)} & A_1^{(j)} \\ K_i & G_i \end{bmatrix} X[k-j] \stackrel{\text{def}}{=} A_d^{(j)} X[k-j] \tag{8}$$

where

$$\begin{aligned} A_0^{(j)} &\stackrel{\text{def}}{=} A^{j+1} + \sum_{\ell=1}^{j} A^{j-\ell} A_B K_{\ell-1} + B_0 K_j \\ A_1^{(j)} &\stackrel{\text{def}}{=} A^i B_1 + \sum_{\ell=1}^{j-1} A^{j-\ell} A_B G_{\ell-1} + B_0 G_j \end{aligned} \tag{9}$$

The goal is to design a controller that makes the switching system in (8) exponentially stable (Definition 2). We do this by introducing a novel LMI formulation that proves the existence of a Common Quadratic Lyapunov Function (CQLF) in Theorem 2. Exponential stability is then guaranteed by Theorem 1, taken from [11].

Theorem 1 (CQLF): A switching system

$$\mathscr{S}_{A_i}: \quad x[k+1] = A_i x[k], \quad i = 1, 2 \cdots N. \tag{10}$$

is exponentially stable with CQLF $V(x) = x^T P x$ if there exists a matrix $P \succ 0$ solving the system of LMIs

$$A_i^T P A_i - P \prec 0, \quad i \in 1, 2 \cdots N \tag{11}$$

Theorem 2: Suppose the communication of the system meets a $(m+1, 1)$-firm deadline requirement, i.e., there are at most m consecutive drops. If there exist matrices $E_j \in \mathbb{R}^{p \times n}$, $F_j \in \mathbb{R}^{p \times p}$, $Q_1 \in \mathbb{R}^{n \times n}$, $Q_2 \in \mathbb{R}^{p \times p}$ with $Q_1, Q_2 \succ 0$, and positive scalar $\gamma < 1$ such that LMI (13) and the system of LMIs (14) are satisfied. Then, utilizing $K_j \in \mathbb{R}^{p \times n}$ and $G_j \in \mathbb{R}^{p \times p}$ as given by (12) in control strategy (5) guarantees exponential stability of system (2).

$$K_j = E_j Q_1^{-1} \qquad G_j = F_j Q_2^{-1}. \tag{12}$$

$$\begin{bmatrix} -\gamma Q_1 & \mathbf{0} & * & * \\ \mathbf{0} & -\gamma Q_2 & * & * \\ AQ_1 + B_0 E_0 & B_1 Q_2 + B_0 F_0 & -Q_1 & 0 \\ E_0 & F_0 & \mathbf{0} & -Q_2 \end{bmatrix} \prec \mathbf{0}, \tag{13}$$

$$\begin{bmatrix} -Q & * \\ \mathscr{L}_j & -Q \end{bmatrix} \prec \mathbf{0}, \;\; j = 1, \cdots, m \tag{14}$$

with

$$\mathscr{L}_j \stackrel{\text{def}}{=} \begin{bmatrix} A^{j+1} Q_1 + \sum_{\ell=1}^{j} A^{j-\ell} A_B E_{\ell-1} + B_0 E_j \\ E_j \\ A^i B_1 Q_2 + \sum_{\ell=1}^{j-1} A^{j-\ell} A_B F_{\ell-1} + B_0 F_j \\ F_j \end{bmatrix}$$

Proof: Using the Schur complement, LMI (13) can be rewritten to

$$\mathscr{L}_0^T Q^{-1} \mathscr{L}_0 - \gamma Q \prec 0$$

where matrices $Q, \mathscr{L}_0$ are defined as

$$Q = \begin{bmatrix} Q_1 & \mathbf{0} \\ \mathbf{0} & Q_2 \end{bmatrix} \qquad \mathscr{L}_0 = \begin{bmatrix} AQ_1 + B_0E_0 & B_1Q_2 + B_0F_0 \\ E_0 & F_0 \end{bmatrix}$$

Substituting (12) and keeping A_n from (6) in mind, we obtain that

$$\mathscr{L}_0 = \begin{bmatrix} AQ_1 + B_0K_0Q_1 & B_1Q_2 + B_0G_0Q_2 \\ K_0Q_1 & G_0Q_2 \end{bmatrix} = A_n \cdot Q \tag{15}$$

and hence

$$QA_n^TQ^{-1}A_nQ - \gamma Q \prec \mathbf{0}. \tag{16}$$

Multiplying (16) from left and right by Q^{-1}, and defining a new positive definite matrix $P \stackrel{\text{def}}{=} Q^{-1}$, we arrive at the following inequality

$$A_n^TPA_n - \gamma P \prec \mathbf{0}, \tag{17}$$

similarly, we can show that inequality (14) implies that

$$A_d^{(j)^T}PA_d^{(j)} - P \prec \mathbf{0}, \quad j = 1, \cdots, m \tag{18}$$

These inequalities imply that a quadratic Lyapunov function $V(X) = X^TPX$ exists for systems (13) and (14), proving Theorem 2. ■

Remark 3: One can view DCC as a special case of state estimation scheme.

V. Case Study

A. Fault-Tolerant Control Evaluation

In order to compare DCC to the established ZOH design, we selected the following highly unstable system matrix A_c and two variations of input matrices $B_c^{(1)}$, $B_c^{(2)}$ as in (19).

$$A_c = \begin{bmatrix} 0 & 1 \\ 6.31 & -15.48 \end{bmatrix} \quad B_c^{(1)} = \begin{bmatrix} 0 \\ 10 \end{bmatrix} \quad B_c^{(2)} = \begin{bmatrix} 0 \\ 1000 \end{bmatrix} \tag{19}$$

For this system, a sampling time of $h = 10ms$ and a delay threshold of $\tau_{th} = 4ms$ was chosen and A, B_0, B_1 were calculated via (3). The system is required to maintain exponential stability tolerating a maximum of $m = 2$ consecutive drops and we designed control gains K_i and G_i using Theorem 2 accordingly. Figure 7 shows the behavior of the plant under the various control strategies subject to random drops within the specifications. All the traces start with an initial disturbance of $x_0 = [0.25 \quad 0]^T$. On the left, we see that PZOH is clearly not suited for the bigger input amplitude as it introduces considerable chatter in u and converges much slower towards 0 in x_1 than the other approaches. In case of a smaller input amplitude, as with $B_c^{(2)}$ on the right, it is a more reasonable choice than ZOH however. Since this system is more sensitive, ZOH can easily lead to over-actuating and hence instability there. The proposed DCC strategy does not suffer from either of these drawbacks and performs well for both input matrices.

B. Formal Verification

We modeled an exemplary network of 5 streams with periods P and jitter j as follows:

$$\begin{aligned} P &= [4\,\text{ms} \quad 5\,\text{ms} \quad 7.5\,\text{ms} \quad 7.5\,\text{ms} \quad 10\,\text{ms}] \\ j &= [1\,\text{ms} \quad 1.5\,\text{ms} \quad 1.0\,\text{ms} \quad 1.5\,\text{ms} \quad 0.5\,\text{ms}] \end{aligned} \tag{20}$$

The streams are traveling over a network with two service automatons. The first is a Time Division Multiple Access (TDMA) automaton that rotates 10 slots, each $0.5ms$ long. In its second slot, it processes one item of stream # 5, the stream of interest. The sixth slot is reserved for maintenance tasks and the remaining slots each process higher priority messages.

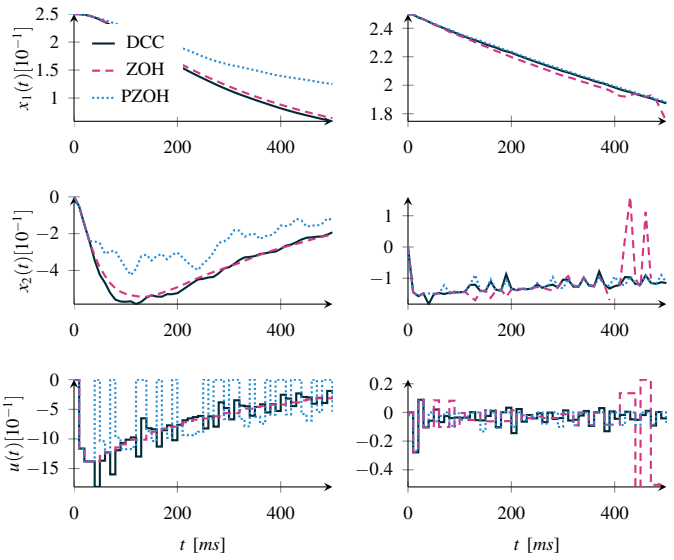


Fig. 7: Comparing x_1, x_2, u (top to bottom) for system (19) with matrix $B_c^{(1)}$ (left) and $B_c^{(2)}$ (right). ZOH (dashed) performs well for large inputs as required by $B_c(1)$; PZOH (dotted) is well-suited for small inputs as induced by $B_c(2)$. DCC (solid) remains stable and more steady in both situations.

After the TDMA ECU, messages are further processed by a bus with speed-stepping that schedules according to a synchronized Fixed Priority Non-Preemptive Scheduling (FPNS) scheme. It starts out processing 1 message every second slot. If there are more than 2 messages waiting to be processed in total, it increases its speed to one message per slot. If there are no more messages waiting, it returns to processing 1 message every second slot. Messages are arbitrated at the beginning of the slot, i.e. the ECA processes the highest priority message that was in the buffer at the end of the last slot.

Verification Result and Combined Design. If we run the system purely on the starting speed, stream # 5 will only succeed occasionally. If the system always runs on the high speed, there are no violations for a deadline of $\tau_{th} = 3\,\text{ms}$. Under the described speed stepping scheme, our model checking implementation can find two design points. We can guarantee that there will be at most five consecutive misses for a deadline $\tau_{th} = 3\,\text{ms}$ and at most two for a deadline $\tau_{th} = 4\,\text{ms}$. We designed controllers for A_c, $B_c^{(2)}$ from (19) at both points and evaluated the performance using a typical quadratic cost-term punishing both non-augmented state and input. To be precise, we used

$$J = \sum x_k'Qx_k + u_k'Ru_k \tag{21}$$

where $Q = I_n, R = I_p$ were chosen to be identity matrices of appropriate size. We ran $N = 300$ randomly initialized Monte-Carlo simulations of length $T_{\max} = 5400\,\text{ms}$. The comparison is shown in Fig. 8. Here, J_{wc} refers to the worst case, i.e., the maximum number of drops always occur and J_{bc} refers to the best case, i.e., no drops occur. Additionally, we simulated the communication using a SystemC model and generated a trace that we fed into the MATLAB simulation of the control system. This way, we obtained the cost J_{real}. The longer delay with fewer losses clearly outperforms the system with more drops and stricter deadline. Note however, that increasing the delay to the point

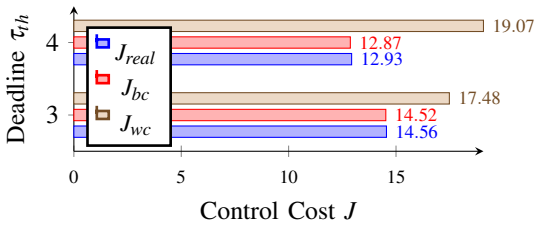


Fig. 8: Comparing two DCC design points found by the ECA network based verification. Designing for higher delay, but less drops leads to a lower cost in the best (J_b) and the simulated case (J_{real}), only worst case J_{wc} is inferior.

where no more drops occur rendered the corresponding LMI system infeasible for our solver.

Runtime Comparison. We compared the proposed evaluation implementation to the one from [7] by looking at the runtime for a deadlock-check. This was performed with the architecture with coarser granularity described there. The deadlock-check required 45 s with the old evaluation technique and less than 2 s with the proposed method. All these measurements were performed on a workstation with an Intel i7-3370 CPU @ 3.4 GHz and 16 GB RAM.

C. Comparing to an Optimistic Lossless Design

In certain cases, depending on the design flow, the designer may have more flexibility when choosing sampling period h. With this in mind, we were interested how the proposed design would fare when comparing it to a design with longer sampling period, but no losses. As shown in Fig. 9, if we forgo the transmission of the two failing messages, we arrive at lossless pattern with sampling period $h_{ll} = (1+m)\cdot h$ and unchanged delay $\tau_{th,ll} = \tau_{th}$. This is *optimistic* and only serves to demonstrate the value of our control design because the drop pattern is typically irregular and unknown a priori.

For our comparison, we used the system described in Section V-A with $B_c^{(2)}$, $\tau_{th} = 2ms$, $m = 2$, and evaluate the performance using the quadratic cost-term from (21). Simulation length and cost per step have been adjusted to account for different sampling periods across the various examples and the longer slots in the lossless cases. Note that these results are therefore not comparable with Fig. 8. Fig. 10 summarizes the result of this comparison for sampling periods $h_1 = 5ms$, $h_2 = 10ms$, $h_3 = 15ms$. The values were obtained from Monte-Carlo simulation with 500 random initializations of x_0 and 150 time steps afterwards. For the smallest sampling period the system is oversampled since reducing the sampling frequency does not reduce QoC even when compared to J_{bc} the lossless case for the smaller period h (not shown). The larger sampling periods are more sensitive however. Here, even the worst case pattern for the faster sampling achieves 10% better QoC; the real case, as obtained by simulating, is about 25% superior.

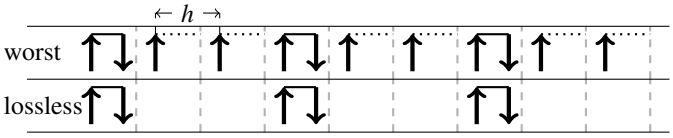


Fig. 9: Assuming the message drops follow the regular worst-case pattern under sampling period h, we could also regulate the system using the larger period $h_{ll} = 3\cdot h$. Note that this pattern is introduced to illustrate the value of our control design and that in real systems, it is actually not guaranteed to be lossless.

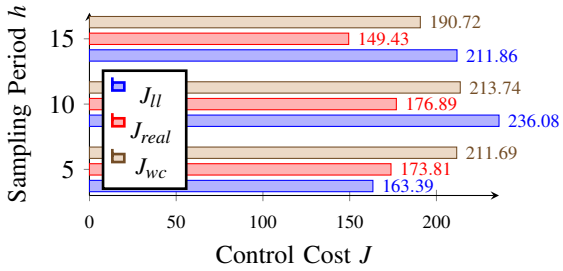


Fig. 10: Comparing the fault-tolerant controller (J_{real} and J_{wc}) to the lossless design (J_{ll})with $h_{ll} = 3\cdot h$ from Fig. 9. For $h = 5$, J_{real} and J_{ll} are on par, indicating oversampling. For larger h, J_{real} is significantly smaller than J_{ll} showing the value of the combination design.

VI. Conclusion

The presented work has two key components. The verification framework provides a formal bound on the consecutive message drops a given architecture can experience. Additionally, we proposed a fault-tolerant control strategy that compensates for a certain amount of faults in the feedback control loop. For its parameterization, one can vary deadline τ_{th}, subsequently obtain bound m for a $(m+1,1)$-firm deadline guarantee and select the most suitable controller from these design points. Together, the verification and the fault-tolerant controller form a co-design framework for distributed control loops implemented over highly utilized networks. Our case study shows a clear improvement in terms of QoC compared to both traditional Sample & Hold and lossless designs with higher sampling periods.

References

[1] P. Kumar, D. Goswami, S. Chakraborty, A. Annaswamy, K. Lampka, and L. Thiele, "A Hybrid Approach to Cyber-Physical Systems Verification," in *DAC*, 2012.

[2] W. Zhang, M. Branicky, and S. Phillips, "Stability of Networked Control Systems," *IEEE Control Systems*, vol. 21, no. 1, pp. 84–99, 2001.

[3] M. Yu, L. Wang, G. Xie, and T. Chu, "Stabilization of Networked Control Systems with Data Packet Dropout via Switched System Approach," in *CACSD*, 2004.

[4] E. Henriksson, H. Sandberg, and K. H. Johansson, "Predictive Compensation for Communication Outages in Networked Control Systems," in *CDC*, 2008.

[5] G. Weiss and R. Alur, "Automata Based Interfaces for Control and Scheduling," *HSCC*, 2007.

[6] R. Alur and G. Weiss, "Regular Specifications of Resource Requirements for Embedded Control Software," in *RTAS*, 2008.

[7] M. Kauer, S. Steinhorst, D. Goswami, R. Schneider, M. Lukasiewycz, and S. Chakraborty, "Formal Verification of Distributed Controllers using Time-Stamped Event Count Automata," in *ASP-DAC*, 2013.

[8] S. Chakraborty, L. Phan, and P. Thiagarajan, "Event Count Automata: a State-Based Model for Stream Processing Systems," in *RTSS*, 2005.

[9] D. Soudbakhsh, L. T. X. Phan, O. Sokolsky, I. Lee, and A. M. Annaswamy, "Co-design of Control and Platform with Dropped Signals," in *ICCPS*, 2013.

[10] L. de Moura, S. Owre, H. Rue, J. Rushby, N. Shankar, M. Sorea, and A. Tiwari, "SAL 2," in *Computer Aided Verification*, ser. Lecture Notes in Computer Science. Springer Berlin / Heidelberg, 2004, vol. 3114, pp. 251–254.

[11] O. Mason and R. Shorten, "On Common Quadratic Lyapunov Functions for Stable Discrete-time LTI Systems," *IMA Journal of Applied Mathematics*, vol. 69, no. 3, pp. 271–283, 2004.