Analysis of Distributed Control Systems with Shared Communication and Computation Resources

Payam Naghshtabrizi and João P. Hespanha

***Abstract*— We address the analysis and implementation of a distributed control system on a network of communicating control units, resulting in a Networked Control System (NCS). We propose an approach based on three steps: control system analysis in terms of sampling times and delays, mapping of control loops to computation/communication hardware components, and scheduling analysis. This procedure is especially important for applications that place a heavy load on the available computation and communication resources and finds direct application in FlexRay control networks.**

I. INTRODUCTION

Inexpensive computation and ubiquitous embedded sensing, actuation, and communication provide tremendous opportunities for social impacts. These systems can be found in the newest generation of engineered systems such as automobiles, high precision medical devices, aerospace systems, and power distribution systems [1]. In particular for automotive applications, such systems are essential to achieve energy efficient, low emission, and safety with high drivability performance [2]. Modern vehicles may have up to 85 ECUs (Electric Control Units) that implement sophisticated control algorithms and communicate through in-vehicle networks such as high-speed and low-speed CAN (Control Area Network) and FlexRay networks connected through gateways [3]. Moreover, software and electronics are an essential part of automotive systems and it is forecasted that they will account for 50% of the total cost of an automobile by 2020. Most of this comes from advanced and control intensive features, including active safety and hybrid powertrains [4].

To physically implement control algorithms we have to assign control tasks to ECUs sharing information through communication networks. The use of shared communication and computation resources, can lead to significant variable delays that in tern can lead to system instability, violation of specifications (loss of correctness), and lack of robustness.

The current trend in the automotive industry is to simulate, implement, and calibrate this type of systems. This is followed by thorough performance tests, which are used to identify faulty behaviors and eventually may lead to system redesign. However, this approach is becoming unpractical as the system complexity increases. The focus of this paper is on the development of a procedure that guarantees correctness and robustness *by design*. In addition, we are interested in modular design procedures that facilitate the calibration, testing, and diagnosis of these complex systems.

P. Naghshtabrizi is with Ford Motor Company, Dearborn, Michigan, U.S.A `pnaghsht@ford.com`. J. P. Hespanha is with the Department of Electrical & Computer Engineering, University of California, Santa Barbara, CA, U.S.A. `hespanha@ece.ucsb.edu`.

Our design procedure consists of three steps that operate at different levels of abstraction for the NCS.

1) The first step operates at the *control system* level of abstraction. At this level, the NCS is viewed as a collection of (possibly coupled) control loops with variable sampling intervals and delays. The details of the computation/communication hardware implementation are ignored and the focus is on determining which combinations of sampling and delays lead to stability and adequate performance.
2) The second step operates at the *physical architecture* level and consists of mapping the control algorithms to computation resources and assigning communication resources to the required information flows between processors.
3) The last step consists of a *scheduling analysis* that, based on the parameters obtained from the control system and the physical architecture, determines if given computation/communication protocols will result in a correct and robust system. A negative answer may require modifications of the physical architecture. For example, it may lead to implementing an algorithm in another ECU, dividing an algorithm between multiple ECUs, increasing or decreasing the sampling rates, or adding another communication network to the system.

This procedure is especially crucial for FlexRay network systems that support fast control algorithms. FlexRay networks are next generation, deterministic, and fault tolerant network protocols to enable high bandwidth and safety critical applications [5]. In FlexRay systems (static segment), application and communication tasks are synchronized across the network based on a global time trigger structure. This structure requires the worse time execution of the control algorithms and functions to make sure that an application task finishes before its communication task starts. Moreover, fast control algorithms, such as x-by-wire and active safety features, are sensitive to the presence of delay and currently dedicated wires are usually used to connect different elements of these systems. As these types of networked applications become common, the need for the type of design procedures proposed in this paper grows. We do not limit ourselves to FlexRay networks as this work applies to any deterministic network (i.e., any network for which the access to the network may not be predetermined but the latest delivery time of a packet can be computed) such as token-passing networks, CAN, switched networks, and FlexRay.

This paper is organized as follows. In Section II we show

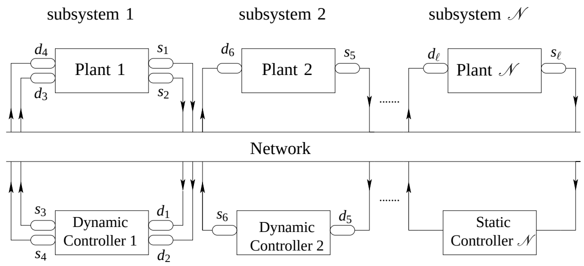

Fig. 1. Schematic description of an NCS with $\mathscr{N}$ different subsystems consisting of a plant and a controller. In this example, subsystem 1 consists of a MIMO plant and a MIMO dynamic controller, subsystem 2 consists of a SISO plant and a dynamic controller, and subsystem $\mathscr{N}$ consists of a SISO plant with a static controller. We refer to a path between a sampler-hold pair as a connection and we assume that there are ℓ connections. A subsystem with one connection, e.g. subsystem $\mathscr{N}$, is modeled by a SISO delay impulsive system, whereas a subsystem with more than one connection, e.g. subsystem 1 is modeled by a MIMO delay impulsive system. Note that the controller of subsystem 2 is a dynamic controller so subsystem 2 consists of two connections but the controller of subsystem $\mathscr{N}$ is a static controller so subsystem $\mathscr{N}$ can be regarded as having only one connection. Network in Fig. 1 can consist of several sub-networks of different types connected by gateways and also elements of the control systems may be implemented on different ECUs.

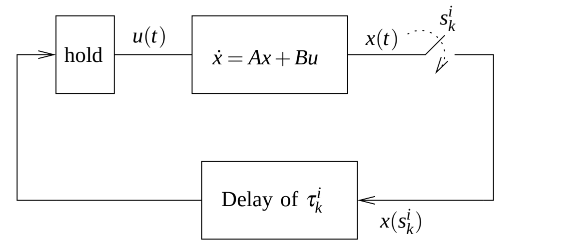

Fig. 2. A subsystem with one connection (denoted by connection i) with delay in the feedback loop where $u(t) = x(s_k^i)$, $\forall t \in [s_k^i + \tau_k^i, s_{k+1}^i + \tau_{k+1}^i)$. Variable delay τ_k^i represents the effect of sharing communication and computation resources.

how to model distributed control systems with shared communication and computation resources as delay impulsive systems and we characterize admissible sampling intervals and delays. In Section III we focus on the physical architecture level and the mapping of algorithms and network connections to specific computation and communication resources. In Section IV we consider the scheduling analysis to guarantee control algorithm correctness. We present an example in Section V and Section VI is devoted to conclusions and future work.

Notation: We denote the transpose of a matrix P by P'. We write $P > 0$ (or $P < 0$) when P is a symmetric positive (or negative) definite matrix and we write a symmetric matrix $\left[\begin{smallmatrix} A & B \\ B' & C \end{smallmatrix}\right]$ as $\left[\begin{smallmatrix} A & B \\ * & C \end{smallmatrix}\right]$. We denote the limit from below of a signal $x(t)$ by $x^-(t)$, i.e., $x^-(t) := \lim_{\tau \uparrow t} x(\tau)$.

II. Control System Level

The schematic description of the system we consider is given in Fig. 1. We consider $\mathscr{N}$ subsystems, each consisting of a plant and a controller connected through the network. The sensors, actuators, and controllers are spatially distributed and they may be implemented on different ECUs. Note that the network in Fig. 1 can consist of several subnetworks of different types connected by gateways. The overall system consists of ℓ connections, where a *connection* is a path between a sampler and its corresponding hold.

A key observation is that the unity feedback of "*traditional control*" that operates in continuous time or at the fixed sampling rate is not adequate in the advanced system in Fig. 1, where sensor data arrives from multiple sources, asynchronously, delayed, and possibly corrupted. In Section II-A we show that delay impulsive systems provide a natural framework to model systems with shared communication and computation resources in a unified way. In Section II-B we briefly review analysis tools to study delay impulsive systems, from which we can determine *maximum tolerable delays* for each connection $i \in \{1, \ldots, \ell\}$, such that all the subsystems remain (exponentially) stable given known upper bounds on the sampling intervals. For simplicity, we focus our attention on the subsystems with one connection modeled as Single-Input Single-Output (SISO) delay impulsive systems, such as subsystem $\mathscr{N}$ in Fig. 1. We refer the reader to the PhD thesis [6] for more general cases, such as subsystem 1 or 2 that can be modeled as Multi-Input Multi-Output (MIMO) delay impulsive systems.

A. Modeling of control systems with shared communication and computation resources

Consider a subsystem with one connection (e.g., subsystem $\mathscr{N}$) that consists of a linear time invariant process with state space model of the form $\dot{x}(t) = Ax(t) + B_u u(t)$, $x \in \mathbb{R}^n, u \in \mathbb{R}^m$ and a static feedback controller with constant gain K connected by sample and hold blocks through the network. The k-th sampling time and the total delay in the control loop of the connection i are denoted by s_k^i, and τ_k^i. At time s_k^i, $k \in \mathbb{N}$ the process's state, $x(s_k^i)$, is sent to the controller and the control command $Kx(s_k^i)$ is sent to the actuators to be used as soon as it arrives at time $s_k^i + \tau_k^i$, and until the next control command update at time $s_{k+1}^i + \tau_{k+1}^i$. The resulting closed-loop system is shown in Fig. 2 and can be expressed by

$$\dot{x}(t) = Ax(t) + Bx(s_k^i), \qquad t_k^i \le t < t_{k+1}^i, k \in \mathbb{N}, \qquad (1)$$

where $B := B_u K$, $t_k^i := s_k^i + \tau_k^i$. Equation (1) can represent a delay impulsive (hybrid) system, as we can define a new state $z_1(t) := x(s_k^i)$, $t_k^i \le t < t_{k+1}^i$, and rewrite the equation (1) as follows:

$$\dot{\xi}(t) = F\xi(t), \qquad t_k^i \le t < t_{k+1}^i, \qquad (2a)$$

$$\xi(t_{k+1}^i) = \begin{bmatrix} x^-(t_{k+1}^i) \\ x(s_{k+1}^i) \end{bmatrix}, \qquad k \in \mathbb{N}, \qquad (2b)$$

where $F := \left[\begin{smallmatrix} A & B \\ 0 & 0 \end{smallmatrix}\right]$, $\xi(t) := \left[\begin{smallmatrix} x(t) \\ z_1(t) \end{smallmatrix}\right]$. The equation (2a) governs the evolution of the system's state between jumps and equation (2b) determines the abrupt change of state at the jump times, which are the times that the state of hold is updated with a new control command.

In this work we do *not* assume that the sampling rate is constant, i.e., we do not require that $s_{k+1}^i - s_k^i = T, \forall k \in \mathbb{N}$.

This allows us to capture the effect of clock drift as well as packet dropout in the distributed NCSs [7].

At the *control systems* level of abstraction, we model the closed-loop subsystems in Fig. 1 as SISO or MIMO delay impulsive systems (SISO case is shown in Fig. 2) with variable delays and sampling reflecting the effect of sharing communication and computation resources. At this level, we ignore the detailed physical architecture as its effect is mostly captured by the variable delays model.

B. Analysis of control systems with variable delay and sampling

The purpose of this section is to determine conditions on the sampling-delay sequences $(\{s_k^i\},\{\tau_k^i\})$ for which one can guarantee (exponential) stability of all subsystems. Specifically, we seek to compute upper bounds $\rho_{i\max}$ and $\tau_{i\max}$ on the sampling interval and delay, respectively:

$$s_{k+1}^i - s_k^i \le \rho_{i\max}, \qquad \tau_k^i \le \tau_{i\max}, \qquad \forall k \in \mathbb{N} \tag{3}$$

for which stability is assured.

Characterizing admissible sampling-delay sequences as in (3) results in a *deterministic* delay impulsive systems, for which there are a few stability results. Here we present an analysis based on *discontinuous Lyapunov functionals*. A list of other approaches and comparison can be found in [6], [1].

For the analysis of the system described by equation (2), we employ a Lyapunov functional of the form

$$\begin{aligned} V := x'Px &+ \int_{t-\bar{\rho}_1}^{t} (\bar{\rho}_{1\max} - t + s)\dot{x}'(s)R_1\dot{x}(s)ds \\ &+ \int_{t-\bar{\rho}_2}^{t} (\bar{\rho}_{2\max} - t + s)\dot{x}'(s)R_2\dot{x}(s)ds \\ &+ (\bar{\rho}_{1\max} - \bar{\rho}_1)(x-w)'X(x-w), \end{aligned} \tag{4}$$

where P, X, R_1, R_2 are appropriately chosen positive definite matrices and

$$\begin{aligned} w(t) := x(t_k^i),\ \bar{\rho}_1(t) := t - s_k^i,\ \bar{\rho}_2(t) := t - t_k^i, \quad t_k^i \le t < t_{k+1}^i, \\ \bar{\rho}_{1\max} := \sup_{t\ge 0} \bar{\rho}_1(t), \qquad \bar{\rho}_{2\max} := \sup_{t\ge 0} \bar{\rho}_2(t). \end{aligned}$$

The timers $\bar{\rho}_1, \bar{\rho}_2$ reset at the update times, t_k^i, and essentially measure the time elapsed since the last sampling time and the last input update time, respectively. By construction, this Lyapunov functional does not increase at the update times t_k^i, at which it is discontinuous. To guarantee stability we further need to show decrease of the Lyapunov functional between these discontinuities [6]. It turns out that this decrease holds if the Linear Matrix Inequalities (LMIs) in the next theorem are satisfied. These LMIs can be solved numerically using software packages such as MATLAB.

Theorem 1: The system (2) is (exponentially) stable over the set of sampling-delay sequences defined by (3), if there exist symmetric positive definite matrices P, X, R_1, R_2 and (not necessarily symmetric) matrices N_1, N_2 that satisfy the following LMIs:

$$\begin{bmatrix} M_1 + (\rho_{i\max}+\tau_{i\max})(M_2+M_3) & \tau_{i\max}N_1 \\ * & -\tau_{i\max}R_1 \end{bmatrix} < 0, \tag{5a}$$

$$\begin{bmatrix} M_1 + (\rho_{i\max}+\tau_{i\max})M_2 & \tau_{i\max}N_1 & (\rho_{i\max}+\tau_{i\max})(N_1+N_2) \\ * & -\tau_{i\max}R_1 & 0 \\ * & * & -(\rho_{i\max}+\tau_{i\max})(R_1+R_2) \end{bmatrix} < 0, \tag{5b}$$

where

$$\begin{aligned} M_1 :=& \bar{F}' \begin{bmatrix} P & 0 & 0 \end{bmatrix} + \begin{bmatrix} P \\ 0 \\ 0 \end{bmatrix} \bar{F} - \begin{bmatrix} I \\ 0 \\ -I \end{bmatrix} X \begin{bmatrix} I \\ 0 \\ -I \end{bmatrix}' - N_1 \begin{bmatrix} I & -I & 0 \end{bmatrix} \\ & - \begin{bmatrix} I \\ -I \\ 0 \end{bmatrix} N_1' - N_2 \begin{bmatrix} I & 0 & -I \end{bmatrix} - \begin{bmatrix} I \\ 0 \\ -I \end{bmatrix} N_2' \\ M_2 :=& \bar{F}'(R_1+R_2)\bar{F}, \\ M_3 :=& \begin{bmatrix} I \\ 0 \\ -I \end{bmatrix} X\bar{F} + \bar{F}'X \begin{bmatrix} I & 0 & -I \end{bmatrix}. \end{aligned} \tag{6}$$

with $\bar{F} := \begin{bmatrix} A & B & 0 \end{bmatrix}$. □

The feasibility of the LMIs (5a) and (5b) for given pairs of $\rho_{i\max}, \tau_{i\max}$ characterizes admissible sampling-delay sequences in equation (3) for the connection i.

Theorem 1 can be extended to MIMO case to characterize sampling-delay sequences of other subsystems with more than one connection [6].

When the upper bound of the sampling intervals $\rho_{i\max}, i \in \{1,\ldots,\ell\}$ are given, one can use (5) to determine the *maximum tolerable delays*, $\tau_{i\max}$ for which (5) holds and consequently stability of all subsystems is guaranteed.

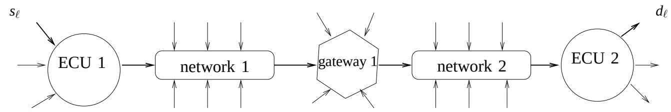

Fig. 3. Physical architecture schematic of the system in Fig. 1. The source and the destination of connection ℓ are denoted by s_ℓ and d_ℓ.The ECU 1 reads the sensor of subsystem ℓ and sends the data to the network after encoding it as a packet. The ECU 2 then receives the packet, decodes it, computes the control command and applies it to the actuator of subsystem ℓ. The network consists of two deterministic networks 1 and 2 connected through a gateway. The arrows refer to other communication and computation jobs of the ECUs, networks, and the gateway.

III. PHYSICAL ARCHITECTURE LEVEL

At this level of abstraction, the physical structure of the system is considered and one maps algorithms and network connections to specific computation and communication resources. We assume that the mapping is given; however, the procedure to construct the map in order to assign functions to ECUs and to assign ECUs to networks so as to optimize and balance the loads is an important open problem. Such procedure should consider the cost of different options, module integration, delay jitter, reliability and robustness of design.

At this level, the system is regarded as a collection of several resources connected in series or parallel to service "jobs". For shared computation resources (ECUs and gateways) jobs typically correspond to the execution of control algorithms, whereas for shared communication resources, jobs typically correspond to the transmission of data in the network.

Fig. 3 shows the physical architecture schematic of the system in Fig. 1 in which we only show the details of connection ℓ. Note that Fig. 3 is more comprehensive than Fig. 1 since diagnostic algorithms, interrupts, and other computation and communication jobs were also included at this level.

IV. Scheduling analysis

The main question addressed in this section is whether the total delays in all connections are smaller than the *tolerable delays* that can be determined by the analysis in Section II or by other specifications. A negative answer to this question may lead to faulty behavior of some or all of the subsystems. The current trend in automotive industry is to simulate, implement, calibrate, and check the performance, and then identify faulty behaviors that may require system redesign. However, detecting problems and fixing them at the simulation and calibration stages are becoming increasingly harder and more expensive due to the increased complexity. This type of approach can be improved through the use of formal techniques that guarantee correctness and cover corner cases and rare events.

These formal techniques are based on results in real-time scheduling [8]. In real-time scheduling, different jobs are released periodically or aperiodically, but with given lower bounds between release times. In the most basic setting, one shared resource services different jobs and servicing a job takes a certain amount of time. Each job should be completed before a *deadline* and if all the timing requirements can be met, then the set of jobs is said to be *schedulable*.

A. Real time scheduling and priority assignment

There are two main classes of priority assignments to jobs: static and dynamic. In static priority assignments, a fixed priority is selected for each job. This form of scheduling is simple, yet it is very inflexible to changes, failures, and often it under-utilizes the shared resources [8]. When scheduling decisions are based on the current decision variables, we have dynamic scheduling. This type of priority assignment may be more difficult to implement because priorities change over time and need to be computed online; however, a dynamic priority assignment is generally more flexible and efficient. In the following, we summarize the most common scheduling policies and we refer the readers to [8] for more details.

First-Come First-Serve (FCFS) scheduling: This policy serves the oldest request first so that resource allocation is based on the order of request arrivals. This policy is generally not suitable for control application because it may serve a job with longer deadline over a job with shorter deadline. In our context, deadlines are determined by the inequalities in (3), which specify the largest admissible delays.

Round-Robin (RR) scheduling: This is a static algorithm in which a fixed time slot is dedicated to each node. This policy is simple and effective when:

- All nodes have data most of the time.
- All nodes are synchronized.
- The network structure is fixed so that no new node joins the network after the time slots are assigned to the nodes.

When a node loses its turn, no matter how close it is to its deadline, it should wait until its next allocated slot.

Deadline Monotonic (DM) scheduling: This *static* policy allocates the resource to nodes according to their deadlines. A task with the *shortest deadline*, is assigned the highest priority. In our context, static deadlines would typically be selected based on the delay upper bounds $\tau_{i\max}$. For example if $\tau_{1\max}=3$ and $\tau_{2\max}=4$ then jobs of source one will always have higher priority over jobs of source two.

Earliest Deadline First (EDF) scheduling: EDF is a dynamic algorithm that assigns priorities to jobs according to their *absolute deadlines*, which are the times remaining to miss the deadline. A job with the earliest absolute deadline, $(t_{il}+\tau_{i\max}-t)$ will have the highest priority, where t_{il} is the last sampling time of connection i and t is the current time. Again consider job one with $\tau_{1\max}=3, t_{1l}=2$ and job two with $\tau_{2\max}=4, t_{2l}=0$. If both nodes have a job ready to be serviced at time $t=3$, job two gains access to the resource because jobs 1 and 2 must be completed before times 5 and 4 respectively, so the node 2 has a closer deadline.

Each of these scheduling policies can be easily implemented on computation resources, but scheduling policies for shared communication resources depend on the specific network. FCFS is not implementable on CAN or FlexRay, whereas RR is the only scheduling policy suitable for the static segment of FlexRay. DM and EDF are both implementable on CAN and on the dynamic segment of FlexRay.

Among all the policies discussed EDF has the advantages of being a dynamic algorithm, generally leading to better network utilization.. The disadvantage of EDF is that the priority of the job is a function of time and should be updated periodically, which requires spending more computation power [6].

B. Scheduling tests and tools

The core of the scheduling analysis is a scheduling test, which determines if a particular scheduling policy can guarantee that the tasks will be serviced, even under worst-case choice of sampling-delays. When this happens, we say that the tasks are *schedulable under the policy*. Our focus here will be on EDF scheduling. The deadline to finish job $i\in\{1,\cdots,n\}$ is denoted by D_i, the lower bound between consecutive job release times is denoted by T_i and the time to service job i is denoted by C_i. If the conditions in the next theorem hold for a given set of jobs, then the set of jobs is schedulable under the EDF policy. This means that the worse delay experienced by every job i, from the time it is released to the resource until the time that it is serviced, is always smaller than its deadline D_i. This delay consists of the service time plus the waiting time to get serviced. The waiting time depends on the scheduling policy and on the priority assignment.

Theorem 2 ([9]): A set of connections $(T_i, C_i, D_i), i\in\{1,\cdots,n\}$ is schedulable over a network under the (non-preemptive) EDF scheduling policy if and only if the following conditions hold:

$$\sum_{i=1}^{n}\frac{C_i}{T_i}\le 1, \tag{7}$$

$$\sum_{i=1}^{n}\lfloor\frac{t-D_i}{T_i}\rfloor^{+}C_i+C_{\max}\le t, \qquad \forall t\in\cup_{i=1}^{n}S_i, \tag{8}$$

where $C_{\max} := \max_i C_i$

$$S_i := \left\{D_i+hT_i : h=0,1,\cdots,\lfloor\frac{d_{\max}-d_i}{T_i}\rfloor\right\},$$

$$d_{\max} := \max\left\{D_1,\cdots,D_n,\frac{\sum_{i=1}^{n}(1-D_i/T_i)C_i+C_{\max}}{1-\sum_{i=1}^{n}C_i/T_i}\right\},$$

$\lfloor . \rfloor$ is a floor function, $\lfloor x\rfloor^{+} := \lfloor x+1\rfloor$ for $x\ge 0$ and zero otherwise. □

Assuming that the only shared resource in Fig. 1 is the network and there is no computation delays. Given $\rho_{i\max}, \tau_{i\max}$ for which the LMIs (5) in Theorem 1 hold, if the sampling-delay satisfies (3) then stability of all subsystems is guaranteed. In real-time scheduling it is assumed that the lower bound between consecutive release times of all source nodes are given. In our problem we denote this lower bound by $\rho_{i\min}$, which corresponds to the smallest sampling interval ever used by the source i. We also denote by $trans_i$ the transmission time from the source to the destination of job i (time taken by the network to "service" a packet sent by source i). Based on Theorems 1 and 2 we have:

Corollary 1: If the set of "jobs" of the shared resource are schedulable based on Theorem 2 with $D_i=\tau_{i\max}, C_i=trans_i, T_i=\rho_{i\min}$ the completion of the job i is guaranteed before $\tau_{i\max}$. Hence any sampling-delay sequence is characterized by (3) and consequently stability of all subsystems connected to network is guaranteed.

This corollary formally verifies the design specification that the end to end delays must be smaller than $\tau_{i\max}$. Without scheduling analysis, one have to rely on extensive testing to find rare events that may occur and violate the specification.

For complex systems such as the one depicted in Fig. 3 in which multiple computation and communication resources are connected in series or parallel, it may be difficult to verify the scheduling test. Even when the input tasks of a resource are released periodically (for example the tasks of ECU 1 in Fig. 3) the period of the output tasks, which can be input tasks of another resource (for example network 1 in Fig. 3), becomes unknown. Moreover, deadlines of jobs for each resources are not given and must be chosen such that the summation of the deadlines are smaller than the tolerable end to end delay; however, the method to choose them is not clear. For these reasons, in such cases one may need more sophisticated, commercial simulation tools for timing verification such as SymTA/S [10] which is based on scheduling analysis, symbolic simulation, and optimization. This tool can also be used for optimizing networks and identifying bottlenecks.

The parameters determined by scheduling tests and simulations (e.g., deadlines or worse case time delays of each shared resource) may be useful not only for system verification, but also for the calibration and testing of final products as well as real-time fault detection. For example, if at the testing stage a particular network component exceeds the maximum delay levels used for design, then it should be identified faulty.

V. EXAMPLE

We consider the example of a motion control system for sheet control in a printer paper path from [11], [12]. The system consists of several pinches or rollers, driven by motors, to move papers through the printer. Motor controllers are implemented on a shared ECU. The position and velocity measurements are sent to the ECU through a CAN network. However, the motors are directly connected to the ECU.

Each subsystem (a single motor-roller pair) can be modeled as

$$\ddot{x}_s=\frac{nr_P}{J_M+n^2J_P}u,$$

where $J_M = 1.95\times10^5$ kg/m^2 is the inertia of the motor, $J_P = 6.5\times10^5$ kg/m^2 is the inertia of the pinch, $r_P = 14\times10^{-3}$ m is the radius of the pinch, $n = 0.2$ is the transmission ratio between motor and pinch, x_s is the sheet of paper position and u is the motor torque. Each subsystem can be presented with the state-space of the form $\dot{x} = Ax + B_u u$ with

$$x=\begin{bmatrix}x_s\\ \dot{x}_s\end{bmatrix}, \quad A=\begin{bmatrix}0&1\\0&0\end{bmatrix}, \quad B_u=\begin{bmatrix}0\\ \frac{nr_P}{J_M+n^2J_P}\end{bmatrix}. \tag{9}$$

We use the state feedback control $K=-\begin{bmatrix}50 & 1.18\end{bmatrix}$ to control the motors. We assume that for each subsystem ECU needs 0.1 ms to read a measurement packet from its buffer, decode the data, calculate the control command, and apply it to the motor. Moreover, we assume that it takes 1 ms to transfer a packet (8 bytes of data on a network with speed 64 kbit/s) from a sensor to the ECU.

Usually in the control algorithm development process, the effect of delay caused by the shared network and ECU is ignored and a sampling time several times faster than what is required for maintaining stability is chosen. By checking the condition

$$eig(\begin{bmatrix}I&0\\I&0\end{bmatrix}e^{\begin{bmatrix}A&B\\0&0\end{bmatrix}h})<1, \quad B:=B_u\times K, \tag{10}$$

on a tight grid of sampling time h, we can show that the closed-loop system remains stable for any *constant* sampling interval smaller than 48 ms, and becomes unstable for larger constant sampling intervals. So a designer who follows traditional design guidelines (and does not consider the effect of shared communication and computation resources) may choose the sampling interval equal 12 ms. The main question at this point is: how many motors can be controlled given this architecture. The answer to this question depends on the designer experience and judgment. A conservative designer would choose $n = 6$ to guarantee the bus load equal to 50% and an aggressive designer would choose up to $n = 11$ so that the bus load remains under 91.7%.

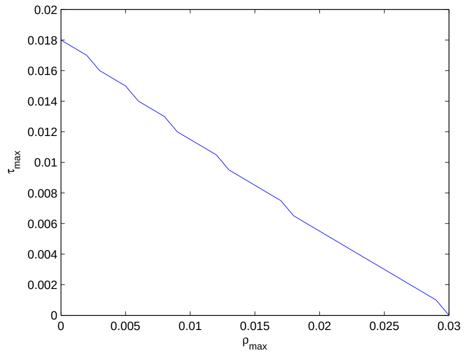

Fig. 4. Admissible set of variable sampling-delay sequences for a single motor-pinch subsystem. The closed-loop remains stable for any sampling interval and delay sequence that respects the upper bounds in (3) for points below the graph.

In the following, we illustrate the application of the design process proposed in this paper to this problem. The closed-loop subsystems can be modeled as (1) with A, B defined in (9) and (10). Based on the results presented in Section II-B, we find the admissible set of sampling-delay sequences shown in Fig. 4. For consistency with the first approach, we choose the sampling times constant and equal to 12 ms. Based on the analysis results depicted in Fig. 4, for this choice, stability of the subsystems are guaranteed for any delay sequences smaller than 10 ms. At the scheduling level, we determine how many subsystems can share the network and ECU such that the total delay in each loop remains smaller than 10 ms. To do so, we test the conditions in Theorem 2 with $T_i = 12$ ms, $C_i = 1$ ms, $D_i = 10 - 0.1 = 9.9$ ms for different values of n. It turns out that the conditions are satisfied for up to $n = 9$. This result indicates that 9 pinch-motor subsystems can share the given architecture while the stability of all subsystems is guaranteed. Note that for $n = 10, 11$ the delay can be larger than 10 ms for some corner cases that may not be easily captured by simulation and testing (in this system the worse case delay occurs when all the sensors send data at the same time). By following the proposed design procedure, we can avoid very conservative choices (e.g., n=6) or choices that lead to unsafe behavior of the subsystems by following a formal method.

VI. Conclusions and Future Work

We proposed a method for the analysis of distributed control systems in which communication and computation resources are shared. Our method provides correctness and robustness of the low level control algorithms, models different elements of the system in a unified way, and provides easy calibration and testing. To achieve these objectives despite the complexity of these systems, we proposed a three-step systematic approach.

In the future we would like to integrate our approach with well-established control design processes in the industry and the V diagram. This is a crucial step to define control design processes for FlexRay distributed systems.

In this paper we focused on the analysis of distributed systems but there are other important open problems related to the design of such systems. Some of them are as follows:

1) Determining the optimal sampling intervals is not trivial. For stability, faster sampling is desirable. However, faster sampling means higher traffic in the network and more load on the ECUs. This may lead to larger end-to-end delays to the point that delays in connections become larger than their allowable upper bound.
2) Providing systematic procedures to map functions to ECUs and to assign ECUs to networks so as to optimize and balance the loads. Such procedure should consider the cost of the different options, module integration, delay jitter, reliability and robustness of design.

Acknowledgment

The first author would like to thank Chaitnaraine Phagoo and Jim Lawlis at Ford Motor Company for helpful discussions. Also the first author would like to acknowledge the support of John Blankenship, Ming Kuang, and Anthony Phillips at Ford Motor Company.

The research of the second author was supported by a grant from the National Science Foundation.

References

[1] J. P. Hespanha, P. Naghshtabrizi, and Y. Xu, "Survey of recent results in networked control systems," *Proc. of IEEE*, vol. 95, no. 1, pp. 138–162, Jan. 2007.

[2] G. Leen, D. Heffernan, and A. Dunne, "Digital networks in the automotive vehicle," *Computing & Control Engineering Journal*, vol. 10, no. 6, pp. 257–266, Dec. 1999.

[3] N. Navet, Y. Song, F. Simonot-Lion, and C. Wilwert, "Trends in automotive communication systems," *Proceedings of the IEEE*, vol. 93, no. 6, pp. 1204–1223, June 2005.

[4] A. Phillips, M. Jankovic, and K. Bailey, "Vehicle system controller design for a hybrid electric vehicle," in *Proceedings of the 2000 IEEE International Conference on Control Applications.*, June 2000, pp. 297–302.

[5] R. Makowitz and C. Temple, "Flexray - a communication network for automotive control systems," in *2006 IEEE International Workshop on Factory Communication Systems*, June 2006, pp. 207– 212.

[6] P. Naghshtabrizi, "Delay impulsive systems: A framework for modeling networked control systems," Ph.D. dissertation, University of California at Santa Barbara, Sep. 2007.

[7] P. Naghshtabrizi, J. P. Hespanha, and A. R. Teel, "On the robust stability and stabilization of sampled-data systems: A hybrid system approach," in *Proc. of the 45th Conf. on Decision and Contr.*, 2006, pp. 4873–4878.

[8] F. Cottet, J. Delacroix, C. Kaiser, and Z. Mammeri, *Scheduling in real-time systems*. Willy, 2002.

[9] Q. Zheng and K. G. Shin, "On the ability of establishing real-time channels in point-to-point packet-switched networks," *IEEE Trans. on Communications*, vol. 42, no. 2/3/4, pp. 1096–1105, Feb. 1994.

[10] R. Henia, A. Hamann, and M. Jersak, "System level performance analysis - the SymTA/S approach," in *IEE Proceedings Computer and Digital Techniques*, vol. 2, no. 152, March 2005, pp. 148–166.

[11] B. Bukkems, R. van de Molengraft, M. Heemels, N. van de Wouw, and M. Steinbuch, "A piecewise linear approach towards sheet control in a printer paper path," in *Proc. of the 2006 Amer. Contr. Conf.*, vol. 1, June 2006, pp. 1315–1320.

[12] M. Cloosterman, N. van de Wouw, W. Heemels, and H. Nijmeijer, "Robust stability of networked control systems with time-varying network-induced delays," in *Proc. of the 45th Conf. on Decision and Contr.*, Dec. 2006.