IEEE Control Systems Letters paper presented at
2023 62nd IEEE Conference on Decision and Control (CDC)
December 13-15, 2023. Marina Bay Sands, Singapore

A multi-processor implementation for networked control systems*

Alejandro I. Maass[1], Wei Wang[2], Dragan Nešić[1], Ying Tan[1], and Romain Postoyan[3]

Abstract— **We study nonlinear networked control systems (NCS) with a multi-processor emulation-based controller. We start with a stable and centralised NCS commonly considered in the literature. Then, we show how to implement the centralised controller over multiple processors inspired by parallel computing techniques, so that stability is preserved (semi-globally and practically) under sufficiently fast computations. An example illustrates the main results.**

I. Introduction

THE use of digital technology is rapidly changing the world through the introduction of new paradigms such as the Internet of Things (IoT) and industrial IoT (IIoT) [1]. In the realm of IIoT, networked control systems (NCS) research is pivotal for modeling, analysing, and designing digital control systems. These systems find use in various IIoT applications, such as vehicle-to-vehicle communication on automated highways [2]. Analysing and designing IIoT applications is challenging due to their complexity, featuring multi-scale systems with diverse communication networks and parallel/distributed computation.

Of greatest interest to this work is the parallel computation aspect that arises when using multiple processors technology in IIoT applications. The main drive for this setup is the need to utilise parallelism effectively for solving complex, large-scale problems while ensuring that communication overhead and delays remain at acceptable levels. Indeed, multiple processors are increasingly used in IIoT, see e.g., automotive industry [3], [4], internal combustion engines [5], and intelligent transportation systems [6], where several processors are used in parallel to solve various computational tasks. This new technology is widely recognised as the main driver for future performance improvements of digital systems [7]. Exploiting the multi-processor structure in NCS is crucial for maximising hardware benefits. The key question in NCS revolves around how parallel control/estimation algorithms interact with plant dynamics. Hence, translation to multi-processor technology poses an interesting question of implementing existing (single processor) control algorithms on multiple processors. This problem is significant as considerable effort has likely been invested in developing and fine-tuning the existing algorithms to satisfy stringent performance objectives (on a single processor) and their implementation on multiple processors should be done with minimum performance loss.

Most literature on (nonlinear) NCS design concentrates on a *monolithic* view of the system, by adopting single processor implementations only, see e.g., [8]–[12]. For linear NCS, most works on distributed controller computation consider architectures where the controllers are often far apart and the topology might change while the system is operating, see [13]–[15]. We emphasise that this approach, although very important, is different to parallel computing systems, where the various processors are located within a small distance of each other. In this context, the work [16] studied scheduling of computational tasks with the aim of optimising the usage of network and computing resources under end-to-end deadline constraints. An approach for decentralised implementation of centralised controllers for linear interconnected systems was proposed in [17]. Specifically, given a centralised linear time-invariant (LTI) controller and a strongly connected plant, a decentralised controller was proposed, so that the state and input of the system under the decentralised controller can become arbitrarily close to those of the system under the centralised controller. Similarly, given a centralised LTI controller, the work [18] showed that a (high-frequency) periodic decentralised implementation is internally stabilising if the centralised LTI controller is stable, minimum-phase, and satisfies some relative degree conditions. We are not aware of any work along these lines for general nonlinear systems.

In this paper, we develop an alternative decentralised design methodology inspired by wave relaxation methods [19] for general nonlinear NCS, which are very much related to Picard's successive approximation method [20]. The goal is to "emulate" a centralised controller on multiple processors that satisfy certain requirements on computational speed and accuracy. This enables us to obtain provable guarantees of stability for nonlinear systems by adapting the single processor hybrid modelling framework [9], [11], [12] to this new scenario. We consider an NCS scenario where the plant and controller communicate via a network, and the control signals are generated by multiple processors. As in parallel computing systems [19], processors are assumed to be located within a small distance of each other, and communications between them are reliable. To generate the control signal, we schedule all processing nodes to take turn in computing components of the controller state.

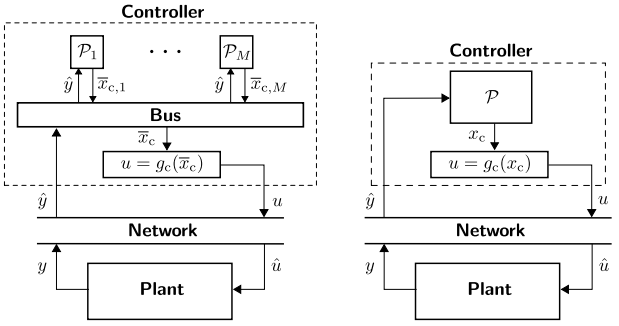
We illustrate that this underlying behaviour can be captured by fast switching flow dynamics in the processors'

*This work was supported in part by the Australian Research Council under the Discovery Project DP200101303, the France-Australia collaboration project IRP-ARS CNRS, and the grant HANDY ANR-18-CE40-0010.

[1]A.I. Maass, D. Nešić and Y. Tan are with the School of Electrical, Mechanical and Infrastructure Engineering, The University of Melbourne, Parkville, 3010, Victoria, Australia. alejandro.maass@unimelb.edu.au, dnesic@unimelb.edu.au, yingt@unimelb.edu.au

[2]W. Wang is an independent researcher. wweiqust@gmail.com

[3]R. Postoyan is with the Université de Lorraine, CNRS, CRAN, F-54000 Nancy, France. romain.postoyan@univ-lorraine.fr



(a) Multi-processor NCS. (b) Single processor NCS.

Fig. 1: Considered NCS architectures.

state. We then apply the hybrid averaging tools from [21] to obtain an averaged model for the multi-processor NCS. To study stability, we use an emulation-based approach. That is, we assume the availability of a centralised controller that stabilises the NCS in absence of the multi-processing structure. Then, we implement such controller over multiple processors, and use the tools in [21] to show that stability of the centralised NCS ensures semi-global practical stability of the multi-processor NCS, for sufficiently small *maximum allowable computation interval* (MACI). This is a newly introduced design parameter to cope with the multi-processor implementation, in addition to the commonly used *maximum allowable transmission interval* (MATI), that typically ensures stability of centralised NCS.

Our main contributions can be summarised as follows.

1) We use hybrid systems to model the interconnection between parallel computing techniques, controller design, and communication networks; leading to a multi-processor model structure often ignored in the NCS literature.

2) We show that stability of the centralised controller can be preserved (in an appropriate sense) for the multi-processor NCS. Compared to existing nonlinear and centralised NCS literature, our results are tailored for multi-processor implementations that arise in IIoT applications such as [3]–[6].

3) Most decentralisation of centralised controllers focus on linear systems only [16]–[18]. We suggest a novel approach using relaxation methods, employing hybrid averaging for controller parallelisation, yet to be explored in control literature, to achieve results for general nonlinear systems.

II. NCS FORMULATION

We tackle controller design for the NCS in Figure 1(a), where a multi-processor setup is used for plant stabilisation. To solve this, we employ an emulation-based approach. Specifically, starting with a stabilising single-processor controller for the NCS in Figure 1(b), our goal is to identify conditions for the multi-processing structure that can maintain the original stability of the centralised controller for the multi-processor NCS in Figure 1(a).

A. Centralised NCS

We first describe the elements of the centralised NCS in Fig. 1(b). We consider the same (centralised) NCS model as commonly used in previous work such as e.g., [9], [11], [12]. The dynamics of the plant and controller are

$$\dot{x}_{\mathrm{p}} = f_{\mathrm{p}}(x_{\mathrm{p}}, \hat{u}), \quad y = g_{\mathrm{p}}(x_{\mathrm{p}}), \tag{1}$$

$$\dot{x}_{\mathrm{c}} = f_{\mathrm{c}}(x_{\mathrm{c}}, \hat{y}), \quad u = g_{\mathrm{c}}(x_{\mathrm{c}}), \tag{2}$$

where $x_{\mathrm{p}} \in \mathbb{R}^{n_{\mathrm{p}}}$ and $x_{\mathrm{c}} \in \mathbb{R}^{n_{\mathrm{c}}}$ denote the plant and controller states, $y \in \mathbb{R}^{n_y}$ is the plant output, $u \in \mathbb{R}^{n_u}$ is the control input, and $(\hat{y}, \hat{u}) \in \mathbb{R}^{n_y} \times \mathbb{R}^{n_u}$ are the most recently received values of (y, u) from the network. The functions $f_{\mathrm{p}}, f_{\mathrm{c}}, \frac{\partial g_{\mathrm{p}}}{\partial x_{\mathrm{p}}}$ and $\frac{\partial g_{\mathrm{c}}}{\partial x_{\mathrm{c}}}$ are assumed to be locally Lipschitz, and g_{p} and g_{c} are continuously differentiable. We note in Fig. 1(b) that a single processor $\mathcal{P}$ computes the whole state x_{c}.

Let $\{t_j\}_{j \in \mathbb{Z}_{\geq 0}}$ be a monotonically increasing sequence of *transmission instants*, where $\mathbb{Z}_{\geq 0} := \{0, 1, 2, \dots\}$. The network is composed of a set of *nodes* $\mathcal{N} := \{1, \dots, N\}$, whose access to the network is governed by an underlying protocol. A node consists of several sensors and/or actuators with their corresponding data being transmitted at the same t_j. Right after transmission, $(\hat{y}, \hat{u})$ are updated as

$$\begin{aligned} \hat{y}(t_j^+) &= y(t_j) + h_y(j, e(t_j)), \\ \hat{u}(t_j^+) &= u(t_j) + h_u(j, e(t_j)), \end{aligned} \tag{3}$$

where the functions h_y and h_u model the scheduling protocol, see e.g., [8], and e denotes the *network-induced error*, defined as $e := (e_y, e_u) \in \mathbb{R}^{n_e}$, with $e_y := \hat{y} - y$, $e_u := \hat{u} - u$, and $n_e := n_y + n_u$. As per the N network nodes, we can write $e = (e_1, \dots, e_N)$, after re-ordering (if necessary). Typically, h_y and h_u are such that $e_\ell(t_j^+) = 0$, $\ell \in \mathcal{N}$, if the ℓ–th node gets access to the network at t_j. We also have that $\dot{\hat{y}} = 0$ and $\dot{\hat{u}} = 0$ for any $t \in [t_j, t_{j+1}]$ (zero-order hold behaviour).

We assume $\tau_{\text{MIATI}} \leq t_{j+1} - t_j \leq \tau_{\text{MATI}}$, for all $j \in \mathbb{Z}_{\geq 0}$. The parameter $\tau_{\text{MATI}} \in \mathbb{R}_{>0}$ is the so-called *maximum allowable transmission interval*, as proposed in e.g., [8]; and $\tau_{\text{MIATI}} \in \mathbb{R}_{>0}$ is the *minimal allowable transmission interval*, see e.g., [22]. Due to hardware limitations, MIATI always exists. In earlier works such as [8], [9], the MIATI was always set to be (essentially) zero, and MATI was the only parameter that played a role for stability. Later on, it was shown in [22] that exploiting the knowledge on MIATI can lead to less conservative stability results. Recently, [11] and [12] illustrated why focusing solely on the MATI as a worst case bound for stability can be unnecessarily conservative. In fact, the worst case $t_{j+1} - t_j = \tau_{\text{MATI}}$ may occur only seldom, whilst the average time between successive transmissions could be significantly smaller. As a consequence, [11], [12] adopted a *reverse average dwell time* (RADT) condition to enforce that, on average, at least one jump occurs every $\tau^*_{r-dt} > \tau_{\text{MIATI}}$ time units. The RADT condition is given by

$$j - i \geq \left[(\bar{t} - t) - \tau_{\text{MATI}}\right] / \tau^*_{r-dt}, \tag{4}$$

where $j - i$ is, loosely speaking, related to the number of "jumps" of the solution of the underlying hybrid model between t and $\bar{t}$, with $(\bar{t}, j) \succeq (t, i)$. Including information about average transmission intervals helps to enlarge the values of MATI while still guaranteeing stability [12].

So far, the NCS literature has focused on single processor implementations like (2); where τ_{MATI}, τ_{MIATI} and τ^*_{r-dt} determine the stability of the NCS. In this paper, we take a step forward by implementing the controller across multiple processors. In this new scenario, not only τ_{MATI}, τ_{MIATI}, and τ^*_{r-dt} play an important role for stability of the NCS in Fig. 1(a), but also the so-called *maximum allowable computational interval* (MACI), as described further below.

B. Multi-processor implementation

We now implement the single processor NCS from the previous section over multiple processors as per Fig. 1(a). Particularly, we consider a common parallel computing scenario with *shared memory* architecture, as described in [19, Chapter 1]; where multiple *processing nodes* $\mathcal{P}_1, \ldots, \mathcal{P}_M$ are located within a small distance of each other, and the communication between them is reliable and done over a shared memory bus. Each processing node $\mathcal{P}_i$ may be a group of individual processors that execute computational tasks in parallel. The time it takes for a computation task to be completed by any processing node $\mathcal{P}_i$, $i \in \mathscr{M} := \{1, \ldots, M\}$, is denoted by[1] $\varepsilon \in \mathbb{R}_{>0}$. We call *activation instants* the times at which some $\mathcal{P}_i$ becomes active to execute such computation task. Formally, let a_k denote the k–th activation instant, $k \in \mathbb{Z}_{\geq 0}$. We note that each computation interval $[a_k, a_{k+1}]$ has length $a_{k+1} - a_k = \varepsilon$, where ε satisfies $0 < \varepsilon \leq \tau_{\text{MACI}}$. Parameter τ_{MACI} denotes the maximum allowable computational interval, and it is a design parameter to deal with the multi-processor structure.

The main task of $\mathcal{P}_1, \ldots, \mathcal{P}_M$ is to implement the centralised controller (2). To that end, they adopt a parallelisation technique highly inspired by wave relaxation methods [19]. That is, each of the M nodes will be assigned to update a different group of components of f_{c} in (2). The way each component of f_{c} depends on the individual components of x_{c} determines the parallelisation strategy. For instance, if a component of f_{c} depends on every element of x_{c}, then only one component of f_{c} can be updated at a time [19]. If the dependency is sparse, then certain updates can be performed in *parallel*. To model this, we let x_{c} be decomposed as $x_{\text{c}} = (x_{\text{c},1}, \ldots, x_{\text{c},M})$, where $x_{\text{c},i} \in \mathbb{R}^{n_{\text{c},i}}$, $i \in \mathscr{M}$, $M \leq n_{\text{c}}$, and $\sum_{i=1}^{M} n_{\text{c},i} = n_{\text{c}}$. Here, $x_{\text{c},i}$ denotes the i-th block component of x_{c}, which contains all the individual components of x_{c} (after re-ordering, if necessary) that are meant to be computed in parallel by the i-th group of processors. This way, the single processor controller (2) is decomposed into M subsystems of the form $\dot{x}_{\text{c},i} = f_{\text{c},i}(x_{\text{c},1}, \ldots, x_{\text{c},M}, \hat{y})$, where $f_{\text{c},i}$ denotes the i–th (block) component of the function f_{c} in (2). There is freedom in choosing the order in which the block components $x_{\text{c},i}$ are to be updated, leading to different *scheduling* algorithms for computation [19]. We focus on a Round-Robin strategy in this work.

Let $\overline{x}_{\text{c},i}$ be the state of $\mathcal{P}_i$, and $\mu \in (0, 1)$ a design parameter. For any $t \in [a_k, a_{k+1}]$, there is an active processing node $\mathcal{P}_i$, $i \in \mathscr{M}$, and idle nodes $\mathcal{P}_n$, $n \in \mathscr{M} \backslash \{i\}$, such that

$$\dot{\overline{x}}_{\text{c},i} = (1/\mu) f_{\text{c},i}(\overline{x}_{\text{c},1}, \ldots, \overline{x}_{\text{c},M}, \hat{y}), \tag{5}$$

$$\dot{\overline{x}}_{\text{c},n} = 0. \tag{6}$$

That is, whenever a computing node is *active*, its state $\overline{x}_{\text{c},i}$ will evolve according to (5), and when it is *idle*, it will buffer its most recent value of the state according to (6). We note that μ in (5) is used to scale the vector fields in order to generate—on average—the dynamics (2) that are generated by a single processor. We design μ in Section IV. The Round-Robin schedule determines which processing node is active in every computation interval of length ε. Particularly, it assigns computing tasks[2] to the M processing nodes in a circular manner so that (5) holds whenever $i = (k-1) \bmod M + 1$, and (6) otherwise. Lastly, we note that the control input for this multi-processor scheme results in $u = g_{\text{c}}(\overline{x}_{\text{c}})$, with $\overline{x}_{\text{c}} := (\overline{x}_{\text{c},1}, \ldots, \overline{x}_{\text{c},M})$, see Fig. 1(a); as opposed to the single processor case where $u = g_{\text{c}}(x_{\text{c}})$.

III. Hybrid system model

In this section, we derive a hybrid model for the multi-processor NCS from Fig. 1(a) based on the system description in Section II. Let $\tau_s \in \mathbb{R}_{\geq 0}$ be a clock to keep track of inter-transmission times; $\tau_r \in \mathbb{R}$ be an additional clock to store the value of τ_s at the last transmission time; and $\kappa_s \in \mathbb{Z}_{\geq 0}$ be a counter for network transmissions. As per [11], [12], the RADT condition (4), which covers he cases in e.g., [9] as special cases, can be modelled by

$$\left.\begin{aligned} \dot{\tau}_s &= 1 \\ \dot{\tau}_r &= 0 \end{aligned}\right\} \tau_s \in [0, \tau_{\text{MATI}}],$$

$$\left.\begin{aligned} \tau_s^+ &= \max\{0, \tau_s - \tau^*_{r-dt}\} \\ \tau_r^+ &= \max\{0, \tau_s - \tau^*_{r-dt}\} \end{aligned}\right\} \tau_s \in [\tau_r + \tau_{\text{MIATI}}, \tau_{\text{MATI}}], \tag{7}$$

with $\tau_r(0,0) \leq \tau_{\text{MATI}} - \tau_{\text{MIATI}}$ and $\tau_r \in [0, \tau_{\text{MATI}}]$. To model the multi-processing behaviour, we introduce the rapidly varying clock $\tau \in \mathbb{R}_{\geq 0}$. Therefore, by using (1), (3), (5), (6), (7), and the definition of e, we write, for $(x_{\text{p}}, \overline{x}_{\text{c}}, e, \tau, \tau_s, \tau_r, \kappa_s) \in \mathbb{R}^{n_{\text{p}}} \times \mathbb{R}^{n_{\text{c}}} \times \mathbb{R}^{n_e} \times \mathbb{R}_{\geq 0} \times [0, \tau_{\text{MATI}}] \times [0, \tau_{\text{MATI}}] \times \mathbb{K}$ (i.e., during flows), with $\mathbb{K} \subset \mathbb{Z}_{\geq 0}$ a compact set,

$$\dot{x}_{\text{p}} = f_{\text{p}}(x_{\text{p}}, e_u + g_{\text{c}}(\overline{x}_{\text{c}})), \tag{8a}$$

$$\dot{\overline{x}}_{\text{c}} = (1/\mu)\Delta(\tau) f_{\text{c}}(\overline{x}_{\text{c}}, e_y + g_{\text{p}}(x_{\text{p}})), \tag{8b}$$

$$\dot{e} = g(x_{\text{p}}, \overline{x}_{\text{c}}, e, \tau), \tag{8c}$$

$$\dot{\tau} = 1/\varepsilon, \tag{8d}$$

$$\dot{\tau}_s = 1, \ \dot{\tau}_r = 0, \tag{8e}$$

$$\dot{\kappa}_s = 0, \tag{8f}$$

where $\Delta(\tau) := \text{diag}\{\delta_1(\tau), \ldots, \delta_M(\tau)\}$, with $\delta_i(\tau) = 1$ when $i = \lfloor \tau \rfloor \bmod M + 1$ and 0 otherwise, and $g(x_{\text{p}}, \overline{x}_{\text{c}}, e, \tau) := \big(-\frac{\partial g_{\text{p}}}{\partial x_{\text{p}}} f_{\text{p}}(x_{\text{p}}, g_{\text{c}}(\overline{x}_{\text{c}}) + e_u), -\frac{\partial g_{\text{c}}}{\partial \overline{x}_{\text{c}}} \frac{1}{\mu} \Delta(\tau) f_{\text{c}}(\overline{x}_{\text{c}}, e_y + g_{\text{p}}(x_{\text{p}})) \big)$. The matrix Δ is used to model the Round-Robin activation of processors, as per

[1] Future work will consider that computation tasks might be completed at different times for each processing node, i.e., ε_i for each $i \in \mathscr{M}$.

[2] Here, a computational task translates into a processor updating its state according to (5), over a computation period of length ε. We neglect any integration errors coming from computing (5), as we want to concentrate on processor scheduling effects rather than numerical discretisation.

the description surrounding (5)–(6). That is, Δ is a periodic time-varying matrix such that, in each ε–long computation period, some δ_i will be equal to 1, meaning $\mathcal{P}_i$ is activated to compute $\overline{x}_{\mathrm{c},i}$, and all other processors are idle (i.e., $\dot{\overline{x}}_{\mathrm{c},n} = 0$) since $\delta_n = 0$ for all $n \in \mathscr{M}\backslash\{i\}$. We highlight that, contrary to prior work on single processor NCS such as [8]–[10], [12], the implementation of the control law via multiple processors naturally introduces rapidly changing flow dynamics when $\varepsilon > 0$ is small, see (8b) and (8d).

Similarly, whenever $(x_{\mathrm{p}}, \overline{x}_{\mathrm{c}}, e, \tau, \tau_s, \tau_r, \kappa_s) \in \mathbb{R}^{n_{\mathrm{p}}} \times \mathbb{R}^{n_{\mathrm{c}}} \times \mathbb{R}^{n_e} \times \mathbb{R}_{\geq 0} \times [\tau_r + \tau_{\text{MIATI}}, \tau_{\text{MATI}}] \times [0, \tau_{\text{MATI}}] \times \mathbb{K}$ (i.e., at jumps),

$$\begin{aligned} x_{\mathrm{p}}^+ &= x_{\mathrm{p}}, \\ \overline{x}_{\mathrm{c}}^+ &= \overline{x}_{\mathrm{c}}, \\ e^+ &= h(\kappa_s, e), \\ \tau^+ &= \tau, \\ \tau_s^+ &= \max\{0, \tau_s - \tau_{r-dt}^*\},\ \tau_r^+ = \max\{0, \tau_s - \tau_{r-dt}^*\}, \\ \kappa_s^+ &= \mathcal{G}_s(\kappa_s, \tau_s), \end{aligned} \tag{9}$$

where $h \coloneqq (h_y, h_u)$ as per (3), and $\mathcal{G}_s : \mathbb{K} \times [0, \tau_{\text{MATI}}] \to \mathbb{K}$ models the discrete dynamics of the counter κ_s. For analysis purposes, we assume κ_s takes values in the compact set $\mathbb{K}$.

We note that jumps in (9) represent network transmissions only, as processors have continuous access to a shared memory bus (i.e. *shared memory* architecture [19]) and there is no discrete communication between them. Different parallel computing architectures may lead to additional jumps. For instance, *message passing* architectures [19, Chapter 1], where there is an interconnection network between processors, may lead to extra jumps to represent this communication among processors; these are left for future work.

We now write the hybrid model (8)–(9) in a compact form that is more amenable for the forthcoming analysis. Let $q \coloneqq ((x_{\mathrm{p}}, \overline{x}_{\mathrm{c}}, e), (\tau_s, \tau_r, \kappa_s)) \in \mathbb{X} \times \mathbb{T}$, with $\mathbb{X} \coloneqq \mathbb{R}^{n_{\mathrm{p}}} \times \mathbb{R}^{n_{\mathrm{c}}} \times \mathbb{R}^{n_e}$ and $\mathbb{T} \coloneqq [0, \tau_{\text{MATI}}] \times [0, \tau_{\text{MATI}}] \times \mathbb{K}$. Then, the hybrid system (8)–(9) can be written as

$$\left.\begin{aligned} \dot{q} &= \mathcal{F}(q, \tau) \\ \dot{\tau} &= 1/\varepsilon \end{aligned}\right\} (q, \tau) \in C \times \mathbb{R}_{\geq 0}, \qquad \left.\begin{aligned} q^+ &= \mathcal{G}(q) \\ \tau^+ &= \tau \end{aligned}\right\} (q, \tau) \in D \times \mathbb{R}_{\geq 0}, \tag{10}$$

where the flow and jump sets are given by

$$\begin{aligned} C &\coloneqq \mathbb{X} \times [0, \tau_{\text{MATI}}] \times [0, \tau_{\text{MATI}}] \times \mathbb{K}, \\ D &\coloneqq \mathbb{X} \times [\tau_r + \tau_{\text{MIATI}}, \tau_{\text{MATI}}] \times [0, \tau_{\text{MATI}}] \times \mathbb{K}. \end{aligned} \tag{11}$$

The mapping $\mathcal{F}$ in (10), for $(q, \tau) \in C \times \mathbb{R}_{\geq 0}$, is defined as $\mathcal{F}(q, \tau) \coloneqq \big(f_{\mathrm{p}}(x_{\mathrm{p}}, e_u + g_{\mathrm{c}}(\overline{x}_{\mathrm{c}})), \frac{1}{\mu}\Delta(\tau) f_{\mathrm{c}}(\overline{x}_c, e_y + g_{\mathrm{p}}(x_{\mathrm{p}})), g(x_{\mathrm{p}}, \overline{x}_{\mathrm{c}}, e, \tau), 1, 0, 0\big)$. The mapping $\mathcal{G}$ in (10), for $(q, \tau) \in D \times \mathbb{R}_{\geq 0}$, is defined as $\mathcal{G}(q) \coloneqq \big(x_{\mathrm{p}}, \overline{x}_{\mathrm{c}}, h(\kappa_s, e), \max\{0, \tau_s - \tau_{r-dt}^*\}, \max\{0, \tau_s - \tau_{r-dt}^*\}, \mathcal{G}_s(\kappa_s, \tau_s)\big)$. Hereafter, and for clarity, we introduce the following definitions. Let $x \coloneqq (x_{\mathrm{p}}, \overline{x}_{\mathrm{c}}) \in \mathbb{R}^{n_x}$, $n_x \coloneqq n_{\mathrm{p}} + n_{\mathrm{c}}$, $\mathcal{F}_{\mathrm{p}}(x, e) \coloneqq f_{\mathrm{p}}(x_{\mathrm{p}}, e_u + g_{\mathrm{c}}(\overline{x}_{\mathrm{c}}))$, $\mathcal{F}_{\mathrm{c}}(x, e) \coloneqq f_{\mathrm{c}}(\overline{x}_c, e_y + g_{\mathrm{p}}(x_{\mathrm{p}}))$, $\mathcal{F}_{\mathrm{x}}(x, e) \coloneqq (\mathcal{F}_{\mathrm{p}}(x, e), \mathcal{F}_{\mathrm{c}}(x, e))$, and $\mathcal{F}_{\mathrm{e}}(x, e) \coloneqq \big(-\frac{\partial g_{\mathrm{p}}}{\partial x_{\mathrm{p}}} \mathcal{F}_{\mathrm{p}}(x, e), -\frac{\partial g_{\mathrm{c}}}{\partial \overline{x}_{\mathrm{c}}} \mathcal{F}_{\mathrm{c}}(x, e)\big)$.

In (10), we can see that τ changes faster compared to the rest of the state q, because of the parallel computing implementation of the controller via (5)–(6). This contrasts significantly with centralised NCS implementations [8]–[10].

IV. Stability analysis

Since the implementation of the centralised controller (2) over multiple processors leads to fast switching flow dynamics when $\varepsilon > 0$ is small; our stability analysis for system (10) will be based on averaging. We impose conditions on an average (approximated) system to conclude stability of the original system (10). To do so, we will apply the averaging tools for hybrid systems proposed in [21].

A. Average system

The first step is to derive the average system. For that purpose, we need the following two preliminary lemmas, whose proofs are given in the appendix.

Lemma 1: The following holds for $\mathcal{F}$ in (10).

(i) For each compact set $K \subset \mathbb{R}^{n_q}$, there exists $R > 0$ such that $|\mathcal{F}(q, \tau)| \leq R$ for all $(q, \tau) \in (K \cap C) \times \mathbb{R}_{\geq 0}$.

(ii) There exists $T \in \mathbb{R}_{>0}$ such that $\mathcal{F}(q, \tau + T) = \mathcal{F}(q, \tau)$ for all $(q, \tau) \in C \times \mathbb{R}_{\geq 0}$. □

Lemma 1(i) ensures the boundedness of $\mathcal{F}$ on compact sets, and (ii) states that $\mathcal{F}$ is periodic with respect to τ. Before stating the next lemma, we define for each $(q, \tau) \in C \times \mathbb{R}_{\geq 0}$,

$$\mathcal{F}_{\mathrm{av}}(q) \coloneqq \frac{1}{T} \int_0^T \mathcal{F}(q, s) \mathrm{d}s, \tag{12}$$

$$\sigma(q, \tau) \coloneqq \int_0^\tau \left[\mathcal{F}(q, s) - \mathcal{F}_{\mathrm{av}}(q)\right] \mathrm{d}s, \tag{13}$$

with T as per Lemma 1(ii). We highlight that (12) will determine the average system, and (13) helps to quantify the difference between the multi-processor and average systems.

Lemma 2: The following holds.

(i) If $\mu = 1/M$ in (8b), then $\mathcal{F}_{\mathrm{av}}$ in (12) is given by $\mathcal{F}_{\mathrm{av}}(q) = (\mathcal{F}_{\mathrm{x}}(x, e), \mathcal{F}_{\mathrm{e}}(x, e), 1, 0, 0)$, where $\mathcal{F}_{\mathrm{x}}$ and $\mathcal{F}_{\mathrm{e}}$ are defined after (11).

(ii) For each compact set $K \subset \mathbb{R}^{n_q}$, there exists $L > 0$ such that σ in (13) satisfies

$$|\sigma(q, t)| \leq L\,, \tag{14}$$

$$|\sigma(q, t) - \sigma(w, s)| \leq L(|q - w| + |t - s|)\,, \tag{15}$$

for all $(q, t), (w, s) \in (K \cap C) \times \mathbb{R}_{\geq 0}$. □

Lemma 2(i) designs μ so that the average of the rapidly time-varying function $\frac{1}{\mu}\Delta(\tau)\mathcal{F}_{\mathrm{c}}(x, e)$, coming from the multiple processor architecture, is given by the centralised controller function $\mathcal{F}_{\mathrm{c}}(x, e)$, when $\mu = 1/M$. Lemma 2(ii) shows some regularity properties for σ in (13), that are useful to state the main stability result. These conditions were adopted as assumptions in previous literature for generic rapidly-varying hybrid systems [21, Assumption 4], and Lemma 2 shows that these are verified for our class of multi-processor NCS.

We can now introduce the *average hybrid system* for the rapidly time-varying system (10), which is given by

$$\begin{aligned} \dot{q} &= \mathcal{F}_{\mathrm{av}}(q), \quad q \in C, \\ q^+ &= \mathcal{G}(q), \qquad q \in D, \end{aligned} \tag{16}$$

with $\mathcal{F}_{\text{av}}$ as per Lemma 2(i), and $C, D, \mathcal{G}$ exactly as in (10). It is important to note that the average system actually coincides with the hybrid system that models the single processor NCS in Fig. 1(b), see e.g., [11], [12]. In the sequel, we show that stability of the centralised NCS in Fig. 1(b), i.e., the average system (16), can be preserved in a semi-global and practical sense, for the multi-processor NCS (10).

B. Stability of the multi-processor NCS

We present the main result of the paper below, whose proof can be found in the appendix.

Theorem 1: Suppose the set $\mathcal{A} := \{((x,e),(\tau_s,\tau_r,\kappa_s)) \in \mathbb{X}\times\mathbb{T} : x = 0, e = 0\}$ is *uniformly globally asymptotically stable*[3] for the average system (16). Then, for the multi-processor NCS (10) with $\mu = 1/M$, the set $\mathcal{A}$ is *semi-globally practically asymptotically stable as* $\varepsilon \to 0^+$ [21]. That is, there exists $\beta \in \mathcal{KL}$ such that for any compact set $K \subset \mathbb{R}^{n_q}$ and any $\nu > 0$, there exist $\varepsilon^* > 0$ and $\delta > 0$ such that any $\varepsilon \in (0, \tau_{\text{MACI}}]$, with $\tau_{\text{MACI}} \in (0, \min\{\delta/(2L(R+1)), \varepsilon^*\})$, implies any solution to (10) initialised in K satisfies

$$|q(t,j)|_{\mathcal{A}} \leq \beta(|q(0,0)|_{\mathcal{A}}, t+j) + \nu, \tag{17}$$

for all $(t,j) \in \operatorname{dom} q$, where R and L are as per Lemmas 1 and 2, respectively. □

Theorem 1 shows that, provided set $\mathcal{A}$ is UGAS for the average system (16), then stability is preserved in a semi-global practical sense for the multi-processor NCS (10), under sufficiently fast computations. A large difference between the multi-processor and average systems, quantified by the bound L in (14), leads to faster computations for stability. Ensuring stability of the average system can be done through existing results in the literature on nonlinear NCS. Particularly, the average system (16) coincides with the centralised NCS models adopted in e.g., [9], [11], [12]. Therefore, stability of (16) can be ensured by the latter.

V. Numerical Example

We consider a single-link flexible joint robot as in [23]. The system dynamics are nonlinear and can be described as $\dot{x}_{\text{p}} = Ax_{\text{p}} + Bu + \Phi(x_{\text{p}})$ and $y = Cx_{\text{p}}$, with $B^\top = (0, 21.6, 0, 0)$, $C = (1,0,0,0)$, $\Phi(x_{\text{p}})^\top = (0,0,0,-7.93\sin(x_{\text{p},3}))$, and $A = \begin{bmatrix} 0 & 1 & 0 & 0 \\ -48.6 & -1.25 & 48.6 & 0 \\ 0 & 0 & 0 & 1 \\ 19.5 & 0 & -19.5 & 0 \end{bmatrix}$. To stabilise the plant, we consider the observer-based centralised controller from [23], which has the form $\dot{x}_{\text{c}} = Ax_{\text{c}} + Bu + \Phi(x_{\text{c}}) - L(y - Cx_{\text{c}})$ and $u = Fx_{\text{c}}$, with gains $L^\top = (6.057, 9.609, 5.918, 5.300)$ and $F = (6.528, 2.637, 0.861, 3.889)$. Only the plant output y is sent over the network (i.e., $e = \hat{y} - y$) with $\tau_{\text{MATI}} = 0.005$ and $\tau_{\text{MIATI}} = 0.001$. The norm of $x = (x_{\text{p}}, x_{\text{c}})$ for the centralised NCS described above is plotted in Fig. 2 (solid blue), with $x(0,0) = (-1,1,-1,1,0,0,0,0)$.

We now implement the centralised controller over four processors with the parallel computing method described in Section II-B. That is, processor $\mathcal{P}_1$ computes $\dot{\overline{x}}_{\text{c},1} = (1/4)(\overline{x}_{\text{c},2} + 6.057(x_{\text{p},1} - \overline{x}_{\text{c},1} + e))$, $\mathcal{P}_2$ computes $\dot{\overline{x}}_{\text{c},2} = (1/4)(-58.2\overline{x}_{\text{c},1} - 1.25\overline{x}_{\text{c},2} + 48.6\overline{x}_{\text{c},3} + 21.6u + 9.609(x_{\text{p},1} + e))$, $\mathcal{P}_3$ computes $\dot{\overline{x}}_{\text{c},3} = (1/4)(\overline{x}_{\text{c},4} + 5.918(x_{\text{p},1} - \overline{x}_{\text{c},1} + e))$, and $\mathcal{P}_4$ computes $\dot{\overline{x}}_{\text{c},4} = (1/4)(14.2\overline{x}_{\text{c},1} - 19.5\overline{x}_{\text{c},3} - 7.93\sin(\overline{x}_{\text{c},3}) + 5.3(x_{\text{p},1} + e))$ in a Round-Robin fashion. We plot $|x(\cdot,\cdot)|$ for the multi-processor NCS in Fig. 2 for three different values of computation interval ε. We can see that, as ε is reduced, the behaviour of the centralised NCS (average system) is preserved for the multi-processor implementation. Also, larger ε leads to instability.

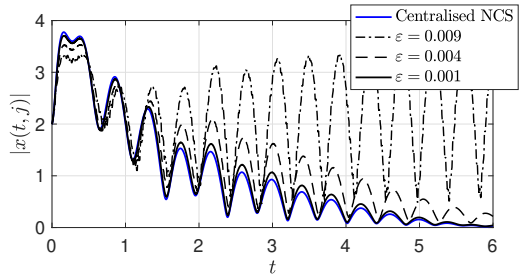


Fig. 2: Comparison between centralised NCS and multi-processor NCS for three values of computation interval ε.

VI. Conclusions and Future Work

We showed how to implement a centralised nonlinear controller in a decentralised manner that preserves stability under the context of NCS and parallel computing. We assumed the communication between processors is reliable and accessible through a shared memory bus. Future work will explore *message-passing* architectures, where each processor has its own local memory and communicates through an interconnection network. This will open the door for fully distributed control in asynchronous scenarios, even with distant processors. We are keen on establishing explicit MACI bounds, especially for linear systems. Additionally, we plan to explore optimisation-based networked control in parallel/distributed computing as part of our future research.

Appendix

Proof of Lemma 1: (i) This item follows from continuity of the maps $\mathcal{F}_{\text{p}}, \mathcal{F}_{\text{c}}, \frac{\partial g_{\text{p}}}{\partial x_{\text{p}}}, \frac{\partial g_{\text{c}}}{\partial \overline{x}_{\text{c}}}$ and the fact that $|\Delta(\tau)| \leq 1, \forall \tau \in \mathbb{R}_{\geq 0}$. (ii) The only τ-dependent component of $\mathcal{F}(q,\tau)$ in (10) corresponds to $\frac{1}{\mu}\Delta(\tau)\mathcal{F}_{\text{c}}(x,e)$. By definition, we know that $\Delta(\tau + M) = \Delta(\tau)$ for any $\tau \in \mathbb{R}_{\geq 0}$. Therefore, $\mathcal{F}(q, \tau + T) = \mathcal{F}(q,\tau)$ for all $(q,\tau) \in C \times \mathbb{R}_{\geq 0}$, with $T = M$. ■

Proof of Lemma 2: (i) Let $\mu = 1/M$ and $(q,\tau) \in C \times \mathbb{R}_{\geq 0}$, and we recall that $\mathcal{F}(q,\tau) = \big(\mathcal{F}_{\text{p}}(x,e), \frac{1}{\mu}\Delta(\tau)\mathcal{F}_{\text{c}}(x,e), g(x_{\text{p}}, \overline{x}_{\text{c}}, \tau), 1, 0, 0\big)$, where $g(x_{\text{p}}, \overline{x}_{\text{c}}, \tau) = \big(-\frac{\partial g_{\text{p}}}{\partial x_{\text{p}}}\mathcal{F}_{\text{p}}(x,e), -\frac{\partial g_{\text{c}}}{\partial \overline{x}_{\text{c}}}(1/\mu)\Delta(\tau)\mathcal{F}_{\text{c}}(x,e)\big)\big)$. Then, it suffices to compute $\frac{1}{T}\int_0^T \frac{1}{\mu}\Delta(s)\mathcal{F}_{\text{c}}(x,e)\mathrm{d}s$, as all the other terms are independent of τ. We proceed component-wise. Let $\mathcal{F}_{\text{c}} = (\mathcal{F}_{\text{c},1}, \ldots, \mathcal{F}_{\text{c},M})$, then $\frac{1}{T}\int_0^T \frac{1}{\mu}\delta_i(s)\mathcal{F}_{\text{c},i}(x,e)\mathrm{d}s = \frac{1}{\mu T}\mathcal{F}_{\text{c},i}(x,e)$, by definition of δ_i and since $T = M$. Consequently, since $\mu = 1/M$, $\mathcal{F}_{\text{av}}(q) = \big(\mathcal{F}_{\text{p}}(x,e), \mathcal{F}_{\text{c}}(x,e), -\frac{\partial g_{\text{p}}}{\partial x_{\text{p}}}\mathcal{F}_{\text{p}}(x,e), -\frac{\partial g_{\text{c}}}{\partial \overline{x}_{\text{c}}}\mathcal{F}_{\text{c}}(x,e), 1, 0, 0\big)$, completing the proof of item (i).

[3]The set $\mathcal{A}$ is UGAS for system (16) if there exists $\beta \in \mathcal{KL}$ such that any solution ξ to (16) satisfies $|\xi(t,j)|_{\mathcal{A}} \leq \beta(|\xi(0,0)|, t+j), \forall (t,j) \in \operatorname{dom}\xi$.

(ii) Let $K \subset \mathbb{R}^{n_q}$ be a compact set and $(q,t),(w,s) \in (K \cap C) \times \mathbb{R}_{\geq 0}$. We first prove (14). By item (i) above, we can write, for any $(q,\tau) \in C \times \mathbb{R}_{\geq 0}$,

$$\mathcal{F}(q,\tau) - \mathcal{F}_{\text{av}}(q) = \big(0, M\Delta(\tau)\mathcal{F}_{\text{c}}(x,e) - \mathcal{F}_{\text{c}}(x,e), 0, \\ -\left(\partial g_{\text{c}}/\partial \overline{x}_{\text{c}}\right)\left[M\Delta(\tau)\mathcal{F}_{\text{c}}(x,e) - \mathcal{F}_{\text{c}}(x,e)\right] 0, 0, 0\big). \quad (18)$$

Define $\tilde{\sigma}(q,t) \coloneqq \int_0^t \left[M\Delta(n)\mathcal{F}_{\text{c}}(x,e) - \mathcal{F}_{\text{c}}(x,e)\right] \mathrm{d}n$, for any $(q,t) \in C \times \mathbb{R}_{\geq 0}$. Given $t \in \mathbb{R}_{\geq 0}$, let $\kappa \in \mathbb{Z}_{\geq 0}$ and $\tilde{T} \in [0,T)$ satisfying $t = \kappa T + \tilde{T}$. We proceed element-wise. For any $i \in \mathscr{M}$, $\tilde{\sigma}_i(q,t) = \int_0^t \mathcal{J}_i(n)\mathrm{d}n = \int_0^T \mathcal{J}_i(n)\mathrm{d}n + \int_T^{2T} \mathcal{J}_i(n)\mathrm{d}n + \cdots + \int_{\kappa T}^{\kappa T+\tilde{T}} \mathcal{J}_i(n)\mathrm{d}n$, where $\mathcal{J}_i(n) \coloneqq M\delta_i(n)\mathcal{F}_{\text{c},i}(x,e) - \mathcal{F}_{\text{c},i}(x,e)$. Note that $\tilde{\sigma}_i(q,\kappa M) = 0$ for each $\kappa \in \mathbb{Z}_{\geq 0}$. Then, $|\tilde{\sigma}_i(q,t)| = \left|\int_{\kappa T}^{\kappa T+\tilde{T}} \left[M\delta_i(n)\mathcal{F}_{\text{c},i}(x,e) - \mathcal{F}_{\text{c},i}(x,e)\right] \mathrm{d}n\right| \leq (M - \tilde{T})c_{K,i}$, where $c_{K,i} \coloneqq \max_{q\in K} |\mathcal{F}_{\text{c},i}(x,e)|$; noting that $\tilde{T} < M$. From (18) we thus get $|\sigma(q,t)| \leq L_A$, with $L_A \coloneqq (1+b_K)Mc_K$, $b_K \coloneqq \max_{q\in K} \left|\frac{\partial g_{\text{c}}}{\partial \overline{x}_{\text{c}}}(\overline{x}_{\text{c}})\right|$ and $c_K \coloneqq \sum_{i=1}^M c_{K,i}$.

We now focus on (15). Similarly to above, we first consider $\tilde{\sigma}$, and assume $s \leq t$ without loss of generality. To save space, we will also use the slight abuse of notation $\mathcal{F}_{\text{c}}(q) = \mathcal{F}_{\text{c}}(x,e)$. Like before, we let $s = \kappa T + \tilde{T}_s$, with $\tilde{T}_s \in [0,T)$. By definition of $\tilde{\sigma}$, we can write $\tilde{\sigma}_i(q,t) - \tilde{\sigma}_i(w,s) = \int_0^t [M\delta_i(m) - 1]\mathcal{F}_{\text{c},i}(q)\mathrm{d}m - \int_0^s [M\delta_i(m) - 1]\mathcal{F}_{\text{c},i}(w)\mathrm{d}m = (\mathcal{F}_{\text{c},i}(q) - \mathcal{F}_{\text{c},i}(w)) \int_0^s [M\delta_i(m) - 1]\mathrm{d}m + \mathcal{F}_{\text{c},i}(q) \int_s^t [M\delta_i(m) - 1]\mathrm{d}m \leq |\mathcal{F}_{\text{c},i}(q) - \mathcal{F}_{\text{c},i}(w)|\,|M - \tilde{T}_s| + |\mathcal{F}_{\text{c},i}(q)||M - 1||t - s|$, where, in the last term, we used the fact that $\delta_i(m) \leq 1$ for any $m \in \mathbb{R}_{\geq 0}$. Thus, $|\tilde{\sigma}_i(q,t) - \tilde{\sigma}_i(w,s)| \leq ML_{\mathcal{F}_{\text{c},i}}|q-w| + c_{K,i}|M-1||t-s| \leq \tilde{L}_i(|q-w| + |t-s|)$, where $\tilde{L}_i \coloneqq \max\{ML_{\mathcal{F}_{\text{c},i}}, c_{K,i}|M-1|\}$ and $L_{\mathcal{F}_{\text{c},i}}$ denotes the Lipschitz constant of $\mathcal{F}_{\text{c},i}$. Naturally, $|\tilde{\sigma}(q,t) - \tilde{\sigma}(w,s)| \leq \tilde{L}(|q-w| + |t-s|)$, with $\tilde{L} \coloneqq \sum_{i=1}^M \tilde{L}_i$. From (18), we can see that it remains to find $\left|\frac{\partial g_{\text{c}}}{\partial \overline{x}_{\text{c}}}(w)\tilde{\sigma}(w,s) - \frac{\partial g_{\text{c}}}{\partial \overline{x}_{\text{c}}}(q)\tilde{\sigma}(q,t)\right| = \left|\frac{\partial g_{\text{c}}}{\partial \overline{x}_{\text{c}}}(q)\left[\tilde{\sigma}(w,s) - \tilde{\sigma}(q,t)\right] + \left[\frac{\partial g_{\text{c}}}{\partial \overline{x}_{\text{c}}}(w) - \frac{\partial g_{\text{c}}}{\partial \overline{x}_{\text{c}}}(q)\right]\tilde{\sigma}(w,s)\right| \leq b_K\tilde{L}\left(|w-q| + |t-s|\right) + L_g|w-q|Mc_K \leq \hat{L}\left(|w-q| + |t-s|\right)$, where $\hat{L} \coloneqq \max\{b_K\tilde{L} + L_gMc_K, b_K\tilde{L}\}$ and L_g denotes the Lipschitz constant for $(\partial g_{\text{c}}/\partial \overline{x}_{\text{c}})$ for the given compact set K. Then, $|\sigma(q,t) - \sigma(w,s)| \leq (\tilde{L} + \hat{L})(|q-w| + |t-s|)$, and thus (14) and (15) are satisfied with $L = \max\{L_A, \tilde{L} + \hat{L}\}$. ■

Proof of Theorem 1: The proof relies on Lemmas 1 and 2, along with [21, Theorem 2]. Our multi-processor NCS in (10) fits the class of rapidly varying hybrid systems in [21]. Additionally, the conditions in [21, Theorem 2] (Assumptions 2-4) hold for (10), given Lemmas 1 and 2, since $\mathcal{A}$ is compact. Thus, the proof is complete. ■

REFERENCES

[1] H. Xu, W. Yu, D. Griffith, and N. Golmie, "A survey on industrial Internet of Things: A cyber-physical systems perspective," *IEEE Access*, vol. 6, pp. 78 238–78 259, 2018.

[2] S. Hegde, D. Plöger, R. Shrivastava, O. Blume, and A. Timm-Giel, "High-density platooning in cellular vehicle-to-everything systems: On the importance of communication-aware networked control design," *IEEE Vehicular Technology Magazine*, vol. 16, no. 3, pp. 66–74, 2021.

[3] G. Macher, A. Höller, E. Armengaud, and C. Kreiner, "Automotive embedded software: Migration challenges to multi-core computing platforms," in *IEEE International Conference on Industrial Informatics*, 2015, pp. 1386–1393.

[4] S. Grubmüller, G. Stettinger, D. Nešić, and D. Watzenig, "Concepts for improved availability and computational power in automated driving," *e & i Elektrotechnik und Informationstechnik*, vol. 135, no. 4, pp. 316–321, 2018.

[5] A. B. Khaled, M. B. Gaid, N. Pernet, and D. Simon, "Fast multi-core co-simulation of cyber-physical systems: Application to internal combustion engines," *Sim. Model. Practice and Theory*, vol. 47, pp. 79–91, 2014.

[6] S. E. Li, Z. Wang, Y. Zheng, Q. Sun, J. Gao, F. Ma, and K. Li, "Synchronous and asynchronous parallel computation for large-scale optimal control of connected vehicles," *Transportation Research Part C: Emerging Technologies*, vol. 121, p. 102842, 2020.

[7] J. Svennebring, J. Logan, J. Engblom, and P. Strömblad, "Embedded multicore: An introduction," *Technical Report, Freescale*, 2009.

[8] D. Nešić and A. Teel, "Input-output stability properties of networked control systems," *IEEE Transactions on Automatic Control*, vol. 49, no. 10, pp. 1650–1667, 2004.

[9] D. Carnevale, A. Teel, and D. Nešić, "A Lyapunov proof of an improved maximum allowable transfer interval for networked control systems," *IEEE Trans. on Autom. Control*, vol. 52, no. 5, p. 892, 2007.

[10] W. Heemels, A. Teel, N. Van De Wouw, and D. Nešić, "Networked control systems with communication constraints: Tradeoffs between transmission intervals, delays and performance," *IEEE Transactions on Automatic Control*, vol. 55, no. 8, pp. 1781–1796, 2010.

[11] S. H. Heijmans, R. Postoyan, D. Nešić, and W. Heemels, "An average allowable transmission interval condition for the stability of networked control systems," *IEEE Transactions on Automatic Control*, vol. 66, no. 6, pp. 2526–2541, 2020.

[12] M. Hertneck and F. Allgöwer, "Reverse average dwell time constraintsenable arbitrary maximum allowabletransmission intervals," in *IFAC Symposium on Nonlinear Control Systems*, 2023, pp. 379–384.

[13] X. Ge, F. Yang, and Q.-L. Han, "Distributed networked control systems: A brief overview," *Information Sciences*, vol. 380, pp. 117–131, 2017.

[14] M. Pajic, S. Sundaram, G. J. Pappas, and R. Mangharam, "The wireless control network: A new approach for control over networks," *IEEE Trans. on Automatic Control*, vol. 56, no. 10, pp. 2305–2318, 2011.

[15] D. Zhang, S. K. Nguang, and L. Yu, "Distributed control of large-scale networked control systems with communication constraints and topology switching," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 47, no. 7, pp. 1746–1757, 2017.

[16] P. Wu, C. Fu, T. Wang, M. Li, Y. Zhao, C. J. Xue, and S. Han, "Composite resource scheduling for networked control systems," in *IEEE Real-Time Systems Symposium*, 2021, pp. 162–175.

[17] J. Lavaei, "Decentralized implementation of centralized controllers for interconnected systems," *IEEE Transactions on Automatic Control*, vol. 57, no. 7, pp. 1860–1865, 2011.

[18] A. Deshmukh and A. Ghosh, "Decentralized implementation of a class of centralized LTI controllers for two-channel systems using periodic control," *IEEE Transactions on Automatic Control*, vol. 67, no. 6, pp. 3180–3187, 2021.

[19] D. Bertsekas and J. Tsitsiklis, *Parallel and Distributed Computation: Numerical Methods*. Athena Scientific, 1997.

[20] É. Picard, "Sur l'application des méthodes d'approximations successives à l'étude de certaines équations différentielles ordinaires," *Journal de Mathématiques Pures et Appliquées*, vol. 9, pp. 217–272, 1893.

[21] A. R. Teel and D. Nešić, "Averaging for a class of hybrid systems," *Dynamics of Continuous, Discrete and Impulsive Systems*, vol. 17, no. 6, pp. 829–851, 2010.

[22] S. H. Heijmans, R. Postoyan, D. Nešić, and W. M. H. Heemels, "Computing minimal and maximal allowable transmission intervals for networked control systems using the hybrid systems approach," *IEEE Control Systems Letters*, vol. 1, no. 1, pp. 56–61, 2017.

[23] M. Ekramian, "Observer-based controller for lipschitz nonlinear systems," *International Journal of Systems Science*, vol. 48, no. 16, pp. 3411–3418, 2017.