IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

DESIGN AND OPTIMIZATION OF EMBEDDED CONTROL SYSTEMS

[1]Brijesh Kumar Dohare,[2]Adarsh Dhar Dubey

[1]Assistant Professor,[2]Assistant Professor

[1]Department Of Electrical Engineering, [2]Department Of Electrical Engineering,

[1]RR Institute Of Modern Technology, Lucknow, India

Abstract: Today, numerous embedded or the cyber-physical system, e.g, in comprise several control applications, automotive domain, sharing same platform. It well known that the such resources sharing leads into the complex temporal behavior that degrade quality of more importantly, control, may even risk stability in worst case, but not properly taken into the account. In this paper, we consider the embedded control or the cyber-physical system, where the several control application share same processing units. Focus is on control-scheduling, the co-design problem, where controller and the scheduling parameter are both optimized. The basic difference between the control application and the traditional embedded application motivate need for the original methodologies for design and optimization of the embedded control system. This work is one other step towards correct design and the optimization of the embedded control system. online and Offline methodologies for the embedded control system are enclosed in this paper. The significance of considering both expected manage stability and performance is discuss and control preparation co-design methodology proposed to the optimize direct performance as guarantee stability. Orthogonal to this band the width efficient stabilize control server is planned, which supports the isolation, compositionality and resource-efficiency in the design. Lastly, we extend scope of future approach to the non-periodic control scheme and the address challenges in the sharing platform by means of self-triggered controller. In addition to the offline methodologie, a narrative online scheduling policy to the stabilize control application is planned.

I. INTRODUCTION AND BACKGROUND

Today, a lot of embedded or the cyber-physical system include quite a few control applications. The majority of the control application are implement as the software responsibilities on microprocessors. These applications are charge of the controlling physical plant associated them. Often though, processing is shared in the middle of several application. This situation shown in Figure 1.1. In this form, there are inverted pendulums, two physical plant, with their controllers running on common processing unit. There might be also exists additional applications running on same platforms.

The interconnection of physical plant to cyber (processing) element introduces notion of the physical times in today's embedded system. Special care is desirable for implementation of the such applications to ensure the high presentation and assurance safety. The design of the embedded control system involves the two main steps: synthesis of controllers and implementation of control application on given execution platforms. Controller synthesis step comprise stage assignment, delay the compensation, control-law synthesis. The execution step, on other hand, is generally

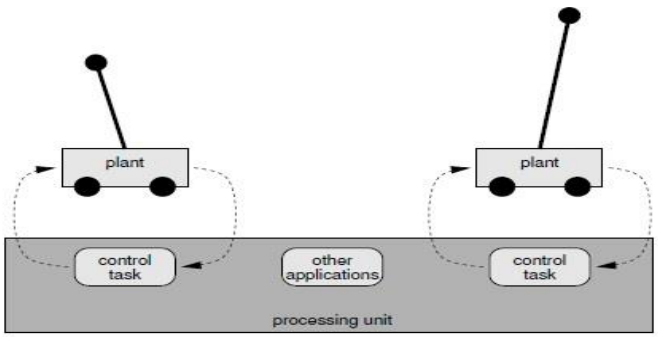


Figure 1.1: Control applications sharing a processing unit.

Concerned with the allocating computational resource to the control application (e.g., scheduling, mapping). This paper focuses on *control–scheduling* problem, and extension to the *control–scheduling co-design*. The goal of control-scheduling difficulty is to assign the processing resources to already synthesize control task such that stability of plants is definite and overall control presentation of system is optimized. The procedure of assigning processing resources to task is usually known as the task scheduling.

One additional step is control–scheduling co-design complexity, where purpose is to create regulator and schedule them in such way that the control function is sure to stay stable, as providing high act as possible. In added language, control-scheduling co-design the problem address joint optimization of control parameter and scheduling parameter. Note that interdependency between scheduling and controller

synthesis makes problem all additional demanding. That is, varying controller parameter will affect scheduling parameter that may used to the assurance high performance and stability, respectively, scheduling parameter affect controller fusion process.

Let us center on inverted pendulum, wherever would similar to change cart place, while observance pendulum in upright position. In an ideal world, each moment devoted continuous-time regulator reads velocity and angular position of inverted pendulum and velocity and position of cart and according to sensed data, applies optimal control signal, in terms of the force. However, it is well recognized that to attain batter performance and suitable stability limits as capable resource-usage and plasticity, frequently, it is required to control plant frequently enough. The option is discrete-time control task, e.g., execute every so often, which can be implemented in software on microcontroller. Considering a continuous time plant and discrete time control task, following process is complete sometimes: (1) The position and velocity of pendulum and cart are sample. (2) Based on sampling information appropriate control input is calculated by software task on processing unit. (3) Finally, control signal is useful in terms of force to cart in actuation phase.

Often, controller is synthesize for given sampling period and time-delays skilled by corresponding control task, e.g., delay from sampling instant to actuation instant. Any runtime difference from assumptions on sampling period or delays during controller synthesis deteriorate control performance.

More concretely, on one hand, controller synthesis determine scheduling parameter that might be assign to control responsibilities in order to make sure imposed performance necessities. On other hand, actual schedule lead to timing property that require to be in use into thought for controller fusion to make sure act supplies.

II. PREVIOUS WORK

The Jitterbug toolbox [2] computes predictable control presentation for linear control system below various timing circumstances, e.g., in presence of delay, even lost samples. Therefore using this toolbox it is likely to study property of delay, lost samples, etc. on control presentation [3]. However, in a lot of cases, underlying assumption on independence of time delays render it not possible to provide solid guarantee about, for example, stability.

In [4], Cervin al. introduce notion of *jitter margin* and suggest an iterative control–scheduling co-design process, based on worst case manage stabilityand performance. The jitter margin is maximum quantity of jitter a control applications can be experience and still stay stable. In [7], similar to [4], authors suggest an integrated move toward for organize design and real-time preparation, based on jitter border performance metric.

In [8], Bini and Cervin expand previous results from [6], by incorporating delay into objective function. They consider objective function which is linear in sampling period and expected delay experienced by each control applications and find optimal sampling period. However, in general, control cost may be a nonlinear purpose. Therefore, iterative optimization approach used to the further get better solution obtain.

Samii et al. [9] propose an integrated the task scheduling and control approach to optimize overall manage performance. They consider static-cyclic and priority-based preparation policy on the distributed platforms. This work is extended to think embedded control system which the switch among different functional mode [9] and to case of fault-tolerant control system.

In [11], authors consider a flexible delay restraint model compare to [12], where not all control task execution meet their deadline. In this way, authors avoid scheming controller for worst-case sensor–actuator wait. While in such approach some of control computation are unnoticed, design is base on range of delay value that are likely to happen.

Naghshtabrizi and Hespanha [9] consider analysis problem of dispersed control systems with communal communication and calculation capital. While control stability consequences are developed for the variable, but surrounded, delay and sampling division interval, schedulability analysis considered is restricted to a set of episodic everyday jobs.

In [8], Koutsoukos et al. propose a passivity based manage design for cyber substantial systems. The main thought is that, by impressive passivity constraint on component dynamics, network effects can be unnoticed, hence separation of concern between control design and completion. Thus, main advantage of proposed move toward is in facilitate compositional component base plan of cyber physical system. However, price to pay for this division is worse performance compare to case where control design and completion are done in integrated manner.

III. METHODOLOGY

3.1. OPTIMAL DESIGN OF STABILIZING CONTROL SERVERS

The analysis and design of such systems, considering a server-based resource reservation mechanism, are addressed. The benefits of employing a server-based approach are manifold: providing a compositional and scalable framework, protection against other tasks' misbehaviors, and systematic control server design and controller–server co-design. We propose a methodology for designing bandwidth-optimal servers to stabilize control tasks. The pessimism involved in the proposed methodology is both discussed theoretically and evaluated experimentally.

we propose to use the resource reservation mechanism and run each controller within its own server, which then isolates each control task in the execution environment (see Figure 1.1). The usage of servers for control tasks presents the following advantages:

• it provides compositionality that is essential for systematic system design methodologies;
• the complexity of the design scales linearly with the number of applications;
• it protects each controller from possible misbehaviors, which may occur within other tasks and then possibly jeopardize the entire system;
• the bandwidth assignment, rather than the priority assignment, may constitute a more accurate instrument to allocate the available computing resources;
• the simple interface provided by the resource reservation mechanism facilitates the controller–server co-design process [5], [4];

• running the controller over a dedicated server, may reduce significantly the jitter of the controller, especially if the server period is smaller than the period of the controller. In short, this

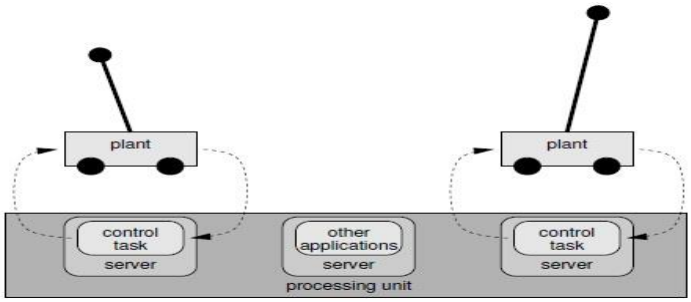


Fig.3.1 Overview of the proposed approach.

is due to the fact that the server guarantees the control task a certain resource bandwidth. This is important since it is often possible to compensate for the constant part of delay, while the process of coping with the jitter is more involved.

In addition to the analysis, we also provide analytic results that can drive the design of a server towards solutions which can guarantee the stability of the controller. The aim of such a design procedure is bandwidth minimization. Since such a solution is derived using a linear upper and lower bound of the server supply function, we also evaluate the amount of pessimism introduced by our technique, both theoretically and experimentally.

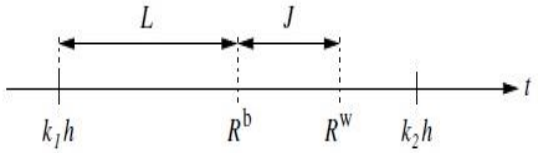


Fig. 3.2. Graphical interpretation of the latency and worst-case response-time jitter

3.2. SYSTEM MODEL

The system is composed of *n* plants. Each plant is controlled by a control task which is executing within a server. Below we describe the model of the plant and the control task. Note that since each plant is considered in isolation, we do not report the index *i* of the control application among all the control applications.

we assume that the controller is given. The plant output is sampled in a strictly periodic manner with period *h*. The control signal is computed by a control task τ . Such a control signal is updated any time the control task completes and is held constant between two consecutive updates. The instants when the input is applied to the plant do then depend on the way the task τ is scheduled. The task parameters, are the best-case execution time c^b, the worst-case execution time c^w, and sampling period *h*.

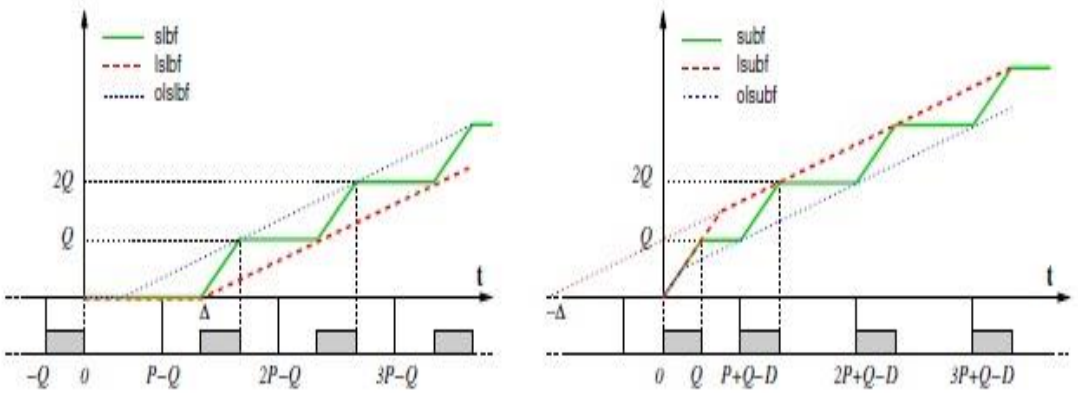
In addition, the task scheduling process determines the best-case response time *R*b, the worst-case response time *R*w, the latency $L = R^b$, and the worst-case response-time jitter (jitter) $J = R^w - R^b$.

The terminology and the notation are illustrated in Figure 3.2 (reproduced, for convenience, from Chapter 2). The latency captures the constant part of the delay, while the jitter corresponds to the variation in the delay experienced by all instances (jobs) of a task. Note that we do not consider any deadlines for control tasks.

3.3. SERVER MODEL

As discussed above, to isolate controllers from one another, each control task is bound to execute over a dedicated server. The periodic server *S* is described by:

- the *server budget Q*;



(a) Worst-case resource allocation scenario. (b) Best-case resource allocation scenario.

Figure 3.3: Worst-case and best-case resource allocation scenarios

• the server period P, and
• the server deadline D

This model was also called EDP (Explicit Deadline Periodic) model [EAL07]. Every period P the server is activated. Then, it allocates Q amount of time to the task, before the server deadline expires.

The latency and jitter experienced by a task are tightly connected to the best-case and worst-case response times. To compute these two quantities, it is then necessary to determine the worst and best case scenarios with regard to the computational resource supplied by the server.

To perform worst-case analysis for the tasks running within a server, a classic approach [2,3,4,7] is to define the supply lower bound function slbf(t), which is formally defined as follows.

3.4. EXACT CHARACTERIZATION

In this section, the exact real-time analysis for a control task is derived. To derive the worst-case response time of a task r , we must consider the minimum amount of resource time available to the task, which is described by slbf(t).

The worst-case response time Rw of the first job of the control task (released at time 0) is equal to the first instant when the server has necessarily provided at least c^w amount of time, that is

$$R^w = \min \{t : \text{slbf}(t) \geq c^w\}$$

By computing the pseudo-inverse of slbf(t), such a value can be computed explicitly and it is equal to

$$R^{\mathrm{w}} = D - Q + \left\lceil \frac{c^{\mathrm{w}}}{Q} \right\rceil (P - Q) + c^{\mathrm{w}}.$$

Unfortunately, the longest response time may occur even at the later jobs, and not necessarily at the first job. This is the case since, as mentioned before, we do not enforce any task deadline, thus, response times are allowed to be longer than the sampling periods h. Therefore, we must evaluate the response times of all jobs within the busy period, as indicated by Lehoczky [9] for the arbitrary deadline case.

The worst-case response time of the control task within a server $S = (Q, P, D)$ is obtained as follows,

$$R^{\mathrm{w}} = \sup_{q \in \mathbb{N}} \left\{ D - Q + \left\lceil \frac{qc^{\mathrm{w}}}{Q} \right\rceil (P - Q) + qc^{\mathrm{w}} - (q-1)h \right\}.$$

We remind that (for example, see the proof of Lemma 1 in [BHCNRB09]) the supremum of (4.9) has a finite solution only when

$$\alpha = \frac{Q}{P} \geq \frac{c^{\mathrm{w}}}{h}.$$

In analogy with the best-case response time Rb is defined through the subf function as follow

$$R\text{b} = \min\{t : \text{subf}(t) \geq c^b\},$$

which can also be computed explicitly, and it is equal to

$$R^{\mathrm{b}} = \max \left\{ 0, 2Q - D - P + \left\lceil \frac{c^{\mathrm{b}}}{Q} \right\rceil (P - Q) \right\} + c^{\mathrm{b}}.$$

3.5. CHARACTERIZATION WITH LINEAR BOUNDS

The main obstacle in using the exact response time for finding the optimal server is that Equations involve ceiling functions. Hence, we propose to compute an upper bound R w to the Rw and a lower bound Rb of Rb using, respectively, the lslbf and lsubf functions, rather than the exact ones, i.e., slbf and subf.

Observe that while this approximation involves pessimism, it is safe from the stability point of view.

By replacing the slbf with the lslbf we can readily compute the response time upper bound, which is

$$\overline{R}^{\mathrm{w}} = \frac{c^{\mathrm{w}}}{\alpha} + \Delta.$$

such an upper bound to the response time is finite only if the server bandwidth is not smaller than the worst-case utilization of the control task, that is

$$\alpha = \frac{Q}{P} \geq \frac{c^{\mathrm{w}}}{h}.$$

Similarly, by replacing by lsubf the lowerbound to the best-case response time is given by,

$$\underline{R}^{\mathrm{b}} = \max \left\{ c^{\mathrm{b}}, \frac{c^{\mathrm{b}}}{\alpha} - \Delta \right\}.$$

3.6. STABILITY CONSTRAINT

it has been discussed that the latency and jitter in the execution of the control applications are decisive factors in the performance and stability of the plants associated with them. To quantify the tolerable amount of latency and jitter by a control application before the instability of the plant, or to guarantee a certain degree of performance, we use the Jitter Margin toolbox [04]. It provides sufficient stability conditions for a closed-loop system with a linear continuous-time plant and a linear discrete-time controller.

As discussed in Section 3.1, for a given sampling period, the stability curve can safely be approximated by a linear function of the latency and worst-case response-time jitter. In this case, the linear *stability condition* for a control application is of the form

$$L + aJ \leq b,$$

where $a \geq 1$, $b \geq 0$. The latency L identifies the constant part of the delay that the control application experiences, whereas the worstcase response-time jitter J captures the varying part of the delay (see Figure 4.2, where R^b and R^w represent the best-case and worst-case

response times, respectively). In order to apply the stability analysis discussed, the values of the latency (L) and worst-case response-time jitter (J) of the control task should be computed. The two metrics are defined based on the worst-case and best-case response times as follows,

$$L = R^b,$$
$$J = R^w - R^b,$$

where R^w and R^b denote the worst-case and best-case response times, respectively. The stability constraint, hence, can be formulated as,

$$L + aJ \le b,$$
$$R_b + a(R_w - R_b) \le b$$

For a given server, the stability condition , which is based on the exact best-case and worst-case response times, determines if the server, in the worst-case, can guarantee the stability of the control task associated with it (analysis problem).

In the context of the optimization problem, as will be discussed in Section 4.6, however, the presence of discontinuous operators (ceiling) in the exact expressions of the worst-case and best-case response times makes them unsuitable. Hence, we use the upper/lower bound of the worst/best-case response times and redefine the latency and the worst-case response-time jitter as follows,

$$\underline{L} = \underline{R}^b,$$
$$\overline{J} = \overline{R}^w - \underline{R}^b.$$

While using the linear supply bounds involves some pessimism compared to the original supply bounds, it is safe from the stability point of view. Nonetheless, the amount of introduced pessimism is discusse.

The stability constraint based on the linear bounds is given in the following,

$$\begin{aligned} b &\ge \underline{L} + a\overline{J}, \\ b &\ge \underline{R}^b + a(\overline{R}^w - \underline{R}^b), \\ b &\ge a\left(\frac{c^w}{\alpha} + \Delta\right) - (a-1)\max\left\{c^b, \frac{c^b}{\alpha} - \Delta\right\}, \\ &= a\left(\frac{c^w}{\alpha} + \Delta\right) + (a-1)\min\left\{-c^b, -\left(\frac{c^b}{\alpha} - \Delta\right)\right\}, \end{aligned}$$

which we rewrite as

$$\min\left\{\frac{a(c^w - c^b) + c^b}{\alpha} + (2a-1)\Delta - b, \quad \frac{ac^w}{\alpha} + a\Delta - (a-1)c^b - b\right\} \le 0.$$

Hence, Equation describes the constraint on the server parameters (the bandwidth α and the delay Δ), which guarantees the stability of the controller running within such a server.

3.7. OPTIMAL DESIGN OF STABILIZING SERVERS

In this section, we describe the procedure to design optimal stabilizing servers. The objective of the optimization is to minimize the utilization required in order to guarantee the stability of all control applications, that is

$$U = \sum_{i=1}^{n}\left(\alpha_i + \frac{\epsilon}{P_i}\right),$$

where ε denotes the switching overhead for the server and is considered to be strictly positive. If no overhead is considered, then the solution would be with P ->0, making this an impractical server period.

We consider the implicit deadline server design, in which all server deadlines are set equal to the periods, $Di = Pi$.

If we assume $D = P$ for all servers, we can perform the following optimization for each control application and conclude based on the obtained results,

$$\begin{aligned} \min_{\alpha,\Delta} \quad & \alpha + \frac{2\epsilon(1-\alpha)}{\Delta} \\ \text{s.t.} \quad & \min\left\{\frac{a(c^w - c^b) + c^b}{\alpha} + (2a-1)\Delta - b, \quad \frac{ac^w}{\alpha} + a\Delta - (a-1)c^b - b\right\} \le 0. \end{aligned}$$

Notice that in the above cost the period P is replaced by $\Delta/2(1-\alpha)$, as it follows from (3)–(4) for $D = P$.

The solution to the above problem is the minimum bandwidth (including the overhead) required to guarantee stability of control task r .

Let us proceed with finding the global optimum of the problem, which is concerned with a single control task in isolation. The stability constraint in can be written as

$$\min\{gi\{\alpha,\Delta), gii(\alpha, \Delta)\} \leq 0,$$

which is equivalent to

$$(gi(\alpha, \Delta) \leq 0) \text{ V } (gii(\alpha, \Delta) \leq 0),$$

with V denoting the *logical or* between two propositions. Thus, the problem (4.22) can be solved by solving individually the following two problems

$$\begin{aligned} \min_{\alpha,\Delta} \quad & \alpha + \frac{2\epsilon(1-\alpha)}{\Delta} \\ \text{s.t.} \quad & \frac{a(c^{\mathrm{w}} - c^{\mathrm{b}}) + c^{\mathrm{b}}}{\alpha} + (2a-1)\Delta - b \leq 0, \end{aligned}$$

And,

$$\begin{aligned} \min_{\alpha,\Delta} \quad & \alpha + \frac{2\epsilon(1-\alpha)}{\Delta} \\ \text{s.t.} \quad & \frac{ac^{\mathrm{w}}}{\alpha} + a\Delta - (a-1)c^{\mathrm{b}} - b \leq 0. \end{aligned}$$

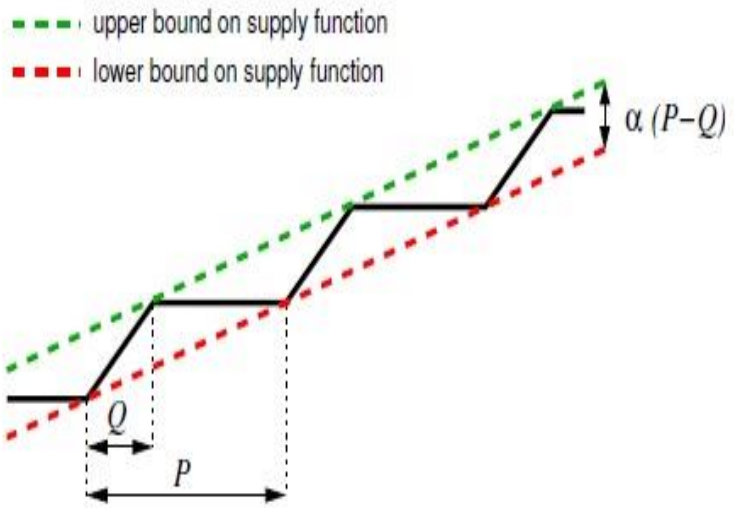


Fig.3.4.The relation between linear, optimistic, and exact supply functions

The next two subsections discuss the theoretical results on the amount of pessimism involved in our design method. We shall first restrict our attention to the stability of one single controller. Then, we focus on both stability and schedulability of the set of all servers.

.

IV. RESULTS AND DISCUSSION

The algorithm has pseudo-polynomial complexity, similar to responsetime analysis for periodic tasks under fixed-priority policy. first we partition the space into convex polytopes, as shown in Figure 4.1. Secondly, for each polytope, we shall find the maximum time the plant could run in open-loop before instability. Thirdly, from this information, we can construct the transition graph. It turns out that the transition graph is as follows,

$$G = \begin{bmatrix} +\infty & 1.1 & +\infty & +\infty \\ +\infty & 1.1 & +\infty & +\infty \\ 0.8 & 0.8 & +\infty & +\infty \\ 0.9 & 0.9 & +\infty & +\infty \end{bmatrix}.$$

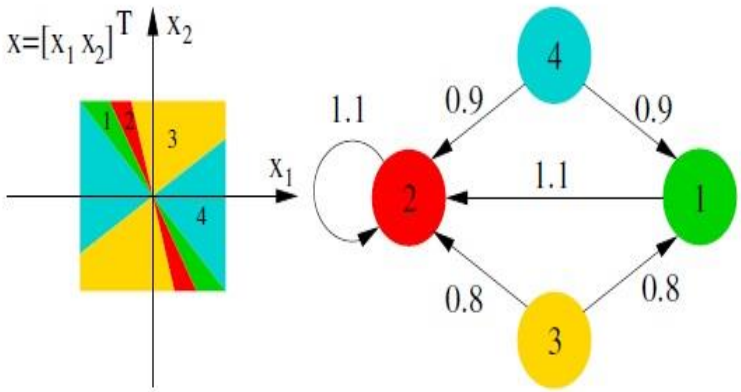


Fig. 4.1The state space partitioning (on the left) and the corresponding transition graph (on the right)

It is often assumed that the computation of the control input and the next activation instant is instantaneous. However, this is different from what occurs in practice. To account for the delay experienced by each self-triggered task, at each execution, the self-triggered task computes the plant state at mthe next triggering instant based on the dynamics of the systems and current control input. Then, based on the plant state at the next triggering instant, the control input after the next triggering instant and the amount of time the plant could run in open-loop after the next triggering instant are calculated.

The lack of efficient schedulability analysis of real-time systems in the presence of self-triggered controllers has been an obstacle in implementing such applications alongside hard real-time applications on shared platforms. In this paper, we have proposed an approach for response-time analysis in the presence of self-triggered control tasks, under the fixed-priority scheduling policy. The proposed approach can be extended to other scheduling policies (e.g., earliestdeadline- first) and task models (e.g., digraph or arbitrary deadlines)

REFERENCES

[1] Karl Johan Åström and Bo Bernhardsson. Comparison of periodic and event-based sampling for firstorder stochastic systems. In Preprints of the 14th World Congress of IFAC, 1999.

[2] Amir Aminifar, Enrico Bini, Petru Eles, and Zebo Peng. Designing bandwidth-efficient stabilizing control servers. In Proceedings of the 34th IEEE Real- Time Systems Symposium, 2013.

[3] Amir Aminifar, Enrico Bini, Petru Eles, and Zebo Peng. Bandwidth-efficient controller–server co-design with stability guarantees. In Proceedings of the 17th Conference for Design, Automation and Test in Europe (DATE), 2014.

[4] Amir Aminifar, Enrico Bini, Petru Eles, and Zebo Peng. Analysis and desing of real-time servers for control applications. IEEE Transactions on Computers, 2015.

[5] Karl Erik Årzén and Anton Cervin. Control and embedded computing: Survey of research directions. In Proceedings of the 16th IFAC World Congress, 2005. [ÅCES00] Karl Erik Årzén, Anton Cervin, Johan Eker, and Lui Sha. An introduction to control and scheduling co- 140

[6] Amir Aminifar, Petru Eles, and Zebo Peng. Jfair: A scheduling algorithm to stabilize control applications. In Proceedings of the 21st IEEE Real-Time and Embedded Technology and Applications Symposium, 2015.

[6] Amir Aminifar, Petru Eles, Zebo Peng, and Anton Cervin. Control-quality driven design of cyberphysical systems with robustness guarantees. In Proceedings of the 16th Conference for Design, Automation and Test in Europe (DATE), 2013.

[7] Amir Aminifar, Petru Eles, Zebo Peng, and Anton Cervin. Stability-aware analysis and design of embedded control systems. In Proceedings of the International Conference on Embedded Software (EMSOFT), pages 1–10, 2013.

[8] Ebru Aydin Gol, Xuchu Ding, Mircea Lazar, and Calin Belta. Finite bisimulations for switched linear systems. IEEE Transactions on Automatic Control, 59(12):3122–3134, 2014.

[9] D. Antunes and W.P.M.H. Heemels. Rollout eventtriggered control: Beyond periodic control performance. IEEE Transactions on Automatic Control, 59(12):3296–3311, Dec 2014.

[10] Luis Almeida and Paulo Pedreiras. Scheduling within temporal partitions: response-time analysis and server design. In Proceedings of the 4th ACM international, conference on Embedded software, pages 95–103, 2004.

[11] Karl Erik Årzén. A simple event-based pid controller. In Preprints of the 14th World Congress of IFAC, 1999. [ASE+12] Amir Aminifar, Soheil Samii, Petru Eles, Zebo Peng, and Anton Cervin. Designing high-quality embedded 141

[12] Amir Aminifar, Soheil Samii, Petru Eles, and Zebo Peng. Control-quality driven task mapping for distributed embedded control systems. In Proceedings of the 17th IEEE Embedded and Real-Time Computing Systems and Applications (RTCSA) Conference, pages 133–142, 2011.

[13] Adolfo Anta and Paulo Tabuada. On the benefits of relaxing the periodicity assumption for networked control systems over CAN. In Proceedings of the 30th IEEE Real-Time Systems Symposium, pages 3– 12, 2009.