Scheduling and Control Co-Design under End-to-End Response Time Constraints in Cyber-Physical Systems

Kyung-Joon Park*, Man-Ki Yoon†, Kyungtae Kang‡, and Chang-Gun Lee§
* Department of Information and Communication Engineering, DGIST
Email: kjp@dgist.ac.kr
† Department of Computer Science, University of Illinois at Urbana-Champaign
Email: mkyoon@illinois.edu
‡ Department of Computer Science and Engineering, Hanyang University
Email: ktkang@hanyang.ac.kr
§School of Computer Science and Engineering, Seoul National University
Email: cglee@snu.ac.kr

***Abstract*—In this paper, we propose an optimization approach for robust control design with end-to-end response time constraints in a multi-resource cyber-physical systems (CPS). We introduce a rigorous performance metric for robust system design from the control theoretic viewpoint. Then, we investigate the impact of end-to-end response time analysis techniques on the control performance. We show that the traditional per-job response time analysis significantly degrades the control performance when real-time tasks visit a resource multiple times. We demonstrate that we can meaningfully improve the control performance by adopting the recently-developed per-resource response time analysis. Our simulation results verify the effectiveness of the proposed co-design framework.**

I. Introduction

Recently, a cyber-physical system (CPS) has emerged as a promising research paradigm, which is a convergence of control, communication, and computation [1]. One fundamental issue in CPS is how to balance the tradeoff between control performance and real-time constraints. In general, in order to improve control performance, more processor time should be devoted to control tasks, which will obviously reduce the processor usage for meeting the deadline of real-time tasks. Consequently, it is crucial how to maximize control performance while satisfying all the deadlines of real-time tasks. An illustration of a CPS application is shown in Fig. 1, where sensors/actuators, controllers, and other nodes communicate through a network.

There have been quite extensive studies on real-time scheduling and control co-design in a single-resource system, where the utilization bound has been typically used to check the schedulability of tasks. What has not been fully investigated is how to co-design scheduling and control in a multi-resource system, where real-time transactions visit multiple resources.

In this paper, we investigate the problem of real-time scheduling and control co-design in a multi-resource CPS. Our contributions can be summarized as follows: *(i)* We formulate scheduling and control co-design in a multi-resource system as an optimization problem with an objective of maximizing a robust system performance. *(ii)* By adopting the recently-developed per-resource end-to-end response time analysis, we show that we can significantly enlarge the feasible region of the co-design optimization problem. *(iii)* By combining the control objective for robust performance and the per-resource response time analysis, we demonstrate that we can significantly improve the robustness of the overall system.

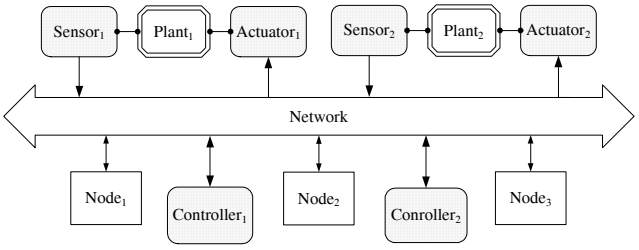


Fig. 1. An illustration of a CPS application.

The rest of the paper is organized as follows. In Section II, we formulate scheduling and control co-design as a constrained optimization problem with end-to-end response time constraints. Then, in Section III, we first investigate a metric for control performance as the objective function of the optimization problem in Section II. We introduce the per-resource analysis in order to derive a tight bound for the end-to-end response time. Our simulation results are given in Section IV. We provide a summary of related work in Section V. Finally, our conclusion follows in Section VI.

II. Problem Formulation

A. Mathematical Notation

We consider a real-time control system that consists of M resources denoted by $\mathbf{R} := \{R_1, R_2, \cdots, R_M\}$, which are either processors or communication links. Without loss of

generality, we do not distinguish the type of resources under the assumption that every resource schedules its jobs based on the fixed-priority preemptive scheduling. Note that non-preemptive tasks on communication links can be dealt with by considering one message length as a blocking factor [2].

With this M-resource real-time system, we assume N periodic control *transactions* denoted by $\{\Gamma_1, \Gamma_2, \cdots, \Gamma_N\}$, where Γ_i has a higher priority than Γ_j if $i < j$. Each transaction Γ_i is composed of $|\Gamma_i|$ *tasks*, denoted by $\{\tau_{i,1}, \tau_{i,2}, \cdots, \tau_{i,|\Gamma_i|}\}$. Each task $\tau_{i,j}$, $j = 1, \cdots, |\Gamma_i|$ of Γ_i is executed on resource $r_{i,j} \in \mathbf{R}$ with the worst-case execution time of $e_{i,j}$.

The first task $\tau_{i,1}$ of transaction Γ_i is released with a period of p_i and the subsequent tasks are released at the completion times of their immediate precedent tasks. Consequently, we can represent Γ_i as follows.

$$\Gamma_i = (p_i, \{\tau_{i,1} = (r_{i,1}, e_{i,1}), \tau_{i,2} = (r_{i,2}, e_{i,2}), \cdots, \tau_{i,|\Gamma_i|} = (r_{i,|\Gamma_i|}, e_{i,|\Gamma_i|})\}).$$

Here, we call one occurrence of the sequence $\tau_{i,1}, \tau_{i,2}, \cdots, \tau_{i,|\Gamma_i|}$ an *instance* of transaction Γ_i. Then, we assume that each instance of Γ_i should be completed in a period, i.e., the end-to-end deadline is equal to the period p_i. However, it should be noted that our analysis can also be applied in a straightforward manner to the case when the end-to-end deadline is shorter than the period [3].

B. Co-Design Problem Formulation

For a given set of N periodic transactions $\{\Gamma_1, \Gamma_2, \cdots, \Gamma_N\}$ over M resources $\{R_1, R_2, \cdots, R_M\}$, we consider the problem of how to maximize a certain control performance metric while satisfying the end-to-end schedulability constraints as follows:

$$\begin{aligned} &\text{maximize } U(\mathbf{p}) \\ &\text{subject to } e2eRspTime_i(\mathbf{p}) \leq p_i, i = 1, \cdots, N, \end{aligned} \tag{1}$$

where $\mathbf{p} = (p_1, \cdots, p_N)$, $U(\cdot)$, and $e2eRspTime_i$ denote the periods of all the transactions, a certain control performance metric, and the end-to-end response time of transaction Γ_i, respectively.

In our formulation of (1), for the schedulability constraints in multi-resource systems, we introduce the end-to-end response time instead of the utilization bound typically used for the single-resource case. Though the utilization bound condition is easy to deal with in analysis because of its simplicity, it is a sufficient condition even in a single-resource system, and may not be sufficient in multi-resource cases.

The control performance of each transaction will typically degrade as its period p_i increases. In addition, the overall objective function U in (1) is generally a certain increasing function of the control performance of the individual transactions. Hence, in order to maximize the objective function U, the periods of transactions, p_i's, should be decreased as much as possible. However, decreased p_i's will result in increase of the end-to-end response times of all lower-priority transactions $\Gamma_j, i < j$, because smaller p_i's will consume more processor time. Consequently, it is of critical importance how to balance this tradeoff between the control performance and the processor usage of control transactions, which is a fundamental issue in CPS.

III. Systematic Approach for Solving the Co-Design Optimization Problem

A. Control Problem Formulation and Performance Metric

Our first task is how to formulate the control problem with a proper performance metric. Here, we aim to design a controller that gives *robust* performance against limitations in implementation such as imprecise actuation and truncation errors. In particular, we adopt the controller design approach in [4].

For control problem formulation, consider that each control transaction Γ_i, $i = 1, \cdots, N$ controls a single input completely reachable system described by n_i linear differential equations, where n_i is called the dimension of the system. The continuous-time system dynamics with the state vector $x^{(i)} = [x_1^{(i)} \ \cdots \ x_{n_i}^{(i)}]^T$ and the control input u_i, where A^T denotes the transpose of A, can be represented in a matrix form as

$$\dot{x}^{(i)} = A_i x^{(i)} + B_i u_i, \tag{2}$$

where $A_i \in \mathbb{R}^{n_i \times n_i}, B_i \in \mathbb{R}^{n_i \times 1}$, and $i = 1, \cdots, N$. For notational simplicity, we will use the superscript (i) and subscript i only when they are strictly required.

Since there is a delay of p in each control loop, the delayed input of $u((k-1)p)$ is applied to the control system during the k-th sampling period. Hence, for the sampled discrete-time system, we introduce an additional state variable of $z(kp) = u((k-1)p)$ in order to account for the delayed input. Then, the augmented system equations sampled with the period p is given as follows from [5]:

$$\begin{bmatrix} x((k+1)p) \\ z((k+1)p) \end{bmatrix} = \Phi \begin{bmatrix} x(kp) \\ z(kp) \end{bmatrix} + \Upsilon u(kp), \tag{3}$$

where

$$\Phi = \begin{bmatrix} e^{Ap} & b\int_0^p e^{A\xi} d\xi \\ 0 & 0 \end{bmatrix}, \Upsilon = \begin{bmatrix} 0 \\ 1 \end{bmatrix}.$$

In the meantime, a state feedback control law for the augmented state vector $[x(kp)^T z(kp)]^T$ is given as follows.

$$u(kp) = k_x x(kp) + k_u z(kp), \tag{4}$$

where k_x and k_u are feedback gain vectors, of which the sizes are $1 \times n_i$ and 1×1, respectively. By plugging (4) into (3), the closed-loop dynamics can be derived as

$$\begin{bmatrix} x((k+1)p) \\ z((k+1)p) \end{bmatrix} = (\Phi + \Upsilon K) \begin{bmatrix} x(kp) \\ z(kp) \end{bmatrix}, \tag{5}$$

where $K = [k_x \ k_u]$. With the discrete-time equations in (5), each control transaction has a vector of feedback gains K as control parameters.

As a performance metric for control design, same as in [4], we define the stability region for control parameters K_i of transaction Γ_i as follows: Let Λ_i denote a set such that the system in (5) is asymptotically stable if and only if $K_i \in \Lambda_i$.

Here, we call Λ_i the *stability region* of transaction Γ_i. Obviously, a small area of Λ_i requires a more accurate controller design because the control parameters K_i should remain in the region despite the imprecision in implementation. Hence, with a large area of Λ_i, the control system will become more robust to implementation errors.

The stability region Λ_i is generally a complex region in a multidimensional space. Hence, in order to quantify the stability region Λ_i with a single scalar value, we need an effective measure that properly represents the area of Λ_i. Here, we introduce the stability center θ_i and the stability radius μ_i as the Chebyshev center and the Chebyshev radius of Λ_i, respectively [4]. Briefly speaking, the Chebyshev center of a bounded set is defined as the center of the largest inscribed ball of the set, and the corresponding radius is called the Chebyshev radius. With the above definitions, the *stability radius* μ_i, which is actually the Chebyshev radius of Λ_i, is an effective measure of the stability region Λ_i.

With the stability radius μ_i, we can now define the performance metric $U(\mathbf{p}) = \min_{i=1,\cdots,N} \mu_i$, which is the smallest stability radius among those of the N stability regions. Now, the overall co-design problem in (1) becomes

$$\begin{aligned} &\text{maximize} \quad \min_{i=1,\cdots,N} \mu_i(p_i) \\ &\text{subject to } e2eRspTime_i(\mathbf{p}) \le p_i, i = 1, \cdots, N, \end{aligned} \tag{6}$$

where we explicitly show the dependencies of μ_i on p_i because the stability radius μ_i of transaction Γ_i is a function of its own period p_i.

B. Computation of the Stability Radius

In the case of first-order systems where $n_i = 1$, the stability region Λ can be analytically obtained as a triangle by using the Jury criterion [5]. First, the characteristic polynomial of the matrix $\Phi + \Upsilon K$ in (5) is given as

$$z^2 - \left(e^{\lambda_1 p} + k_u\right) z + e^{\lambda_1 p} k_u - b k_x I(p),$$

where λ_1 is the eigenvalue of the continuous-time system in (2) and $I(p) = \int_0^p e^{\lambda_1 \xi} d\xi = (e^{\lambda_1 p} - 1)/\lambda_1$. Then, the Jury criterion [5] gives the following inequalities for $K = [k_x \ k_u]$:

$$\begin{bmatrix} e^{\lambda_1 p} & -bI(p) \\ -(e^{\lambda_1 p} - 1) & bI(p) \\ -(e^{\lambda_1 p} + 1) & bI(p) \end{bmatrix} \begin{bmatrix} k_x \\ k_u \end{bmatrix} < \begin{bmatrix} 1 \\ 1 - e^{\lambda_1 p} \\ 1 + e^{\lambda_1 p} \end{bmatrix}. \tag{7}$$

Consequently, the stability region Λ for K can be obtained as a triangle, which is formed by three lines given in (7). In addition, from Proposition 1 in [4], the stability radius μ is given as

$$\mu = \begin{cases} \frac{\lambda_1}{e^{\lambda_1 p}(\lambda_1 + |B|) - |B|}, & \text{if } \lambda_1 > 0; \\ \frac{2\lambda_1}{e^{\lambda_1 p}(\lambda_1 + 2|B|) + \lambda_1 - 2|B|}, & \text{if } \lambda_1 < 0; \\ \frac{1}{1 + p|B|}, & \text{if } \lambda_1 = 0. \end{cases} \tag{8}$$

Note that it can be easily shown from (8) that the stability radius μ monotonically decreases with p.

In general, it is formidable to compute the stability radius in higher-order systems. However, it is still possible to derive the empirical probability for the system in (5) being stable by using the randomized algorithms [6].

Let $P(\theta, \mu)$ denote the empirical probability for (5) being stable in the set of $B_\mu(\theta) = \{K \mid ||K - \theta|| \le \mu\}$. Once μ is given, we can numerically find $\theta^*(\mu)$ that maximizes $P(\theta, \mu)$. Hence, for any given tolerance of ϵ, the stability radius and the stability center can be estimated as the minimum μ and the corresponding θ^* such that $P(\theta^*(\mu), \mu) \le 1 - \epsilon$.

Here, we give a brief introduction on how to apply the randomized algorithms to the calculation of the empirical probability $P(\theta, \mu)$ for a given μ. First, draw m random samples for θ, denoted by $\theta_1, \cdots, \theta_m$. Then, for each θ_i, by drawing n samples of K in $B_\mu(\theta)$, calculate the empirical probability of $P(\theta_i, \mu)$ denoted by $P_n(\theta_i, \mu)$. Finally, we can obtain the estimate of the stability center as $\theta_{m,n} = \arg\max_{i=1,\cdots,m} P_n(\theta_i, \mu)$. Note that a detailed explanation including the selection rule for m and n with a given ϵ can be found in [6].

C. Per-Job End-to-End Response Time Analysis and Multiple Visit Problem

With the control problem formulation in the preceding sections, the remaining issue is how to derive a tight bound for the end-to-end response time. For calculation of the end-to-end response time, we may use the conventional per-job end-to-end response time analysis [7], of which a brief overview is as follows.

For task $\tau_{i,k}$ in transaction Γ_i, its per-job worst-case response time, denoted by $w_{i,k}$, is calculated by using the following recursive equation.

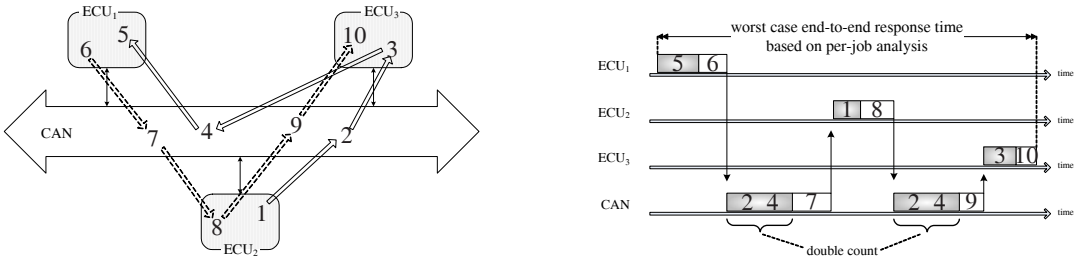
$$w_{i,k} = e_{i,k} + \sum_{\forall j < i} \sum_{\{a | r_{j,a} = r_{i,k}\}} \left\lceil \frac{J_{j,a} + w_{i,k}}{p_j} \right\rceil e_{j,a}, \tag{9}$$

where $J_{j,a}$ is the worst-case release jitter of a-th task $\tau_{j,a}$ of a higher priority transaction Γ_j. Equation (9) implies that the per-job worst-case response time of task $\tau_{i,k}$ can be calculated by adding the following two terms; *(i)* its own execution time $e_{i,k}$ and *(ii)* the largest possible delay due to higher priority jobs on the same resource. Consequently, by applying (9) to all the tasks in transaction Γ_i, the worst-case end-to-end response time, $e2eRspTime_i$, can be calculated by summing up all the per-job response times as follows.

$$e2eRspTime_i = \sum_{k=1}^{|\Gamma_i|} w_{i,k}.$$

However, this per-job analysis can severely overrate the end-to-end response time when transaction Γ_i visits the same resource multiple times, which is termed the *multiple visit problem* [3].

As an illustrative example for the multiple visit problem of the per-job analysis, we consider the case in Fig. 2(a), where three Electronic Control Units (ECUs) are connected through a Controller Area Network (CAN) bus. We assume two transactions in the system as follows: A high priority transaction consists of five tasks (1, 2, 3, 4, 5) that utilizes



(a) Two transactions on an ECU network

(b) Per-job based analysis

Fig. 2. Illustrative example of the multiple visit problem.

ECU_2, CAN, ECU_3, CAN, and ECU_1, respectively. A low priority transaction has five tasks (6 , 7, 8, 9, 10) that utilizes ECU_1, CAN, ECU_2, CAN, and ECU_3, respectively. In this system, the conventional per-job analysis is illustrated in Fig. 2(b).

In the figure, the low priority transaction visits CAN two times with task 7 and task 9. For each visit, the per-job analysis assumes that the worst-case delay by the high-priority tasks is attributed by task 2 and task 4. Hence, as shown in Fig. 2(b), the execution times of tasks 2 and 4 in the high-priority transaction may be *double-counted* in calculation of the end-to-end response time of the low-priority transaction. Obviously, this redundant counts in the per-job analysis will result in an overestimation of the end-to-end response time, which becomes more severe as the number of the multiple visit increases.

D. Per-Resource End-to-End Response Time Analysis

To resolve the multiple visit problem of the traditional per-job response time analysis explained in the previous section, we introduce the recently developed per-resource end-to-end response time analysis [3]. Figure 3 gives an illustration that compares the per-job analysis and the per-resource analysis.

In our per-resource response time analysis, the end-to-end response time of transaction Γ_i can be calculated as follows:

$$e2eRspTime_i = \sum_{\forall R_l \in \mathbf{R}} \left(\sum_{\{(i,k)|r_{i,k}=R_l\}} e_{i,k} + \sum_{j=1}^{i-1} TD_i^j(R_l) \right), \qquad (10)$$

where $e_{i,k}$ is the execution time of task $\tau_{i,k}$ and $TD_i^j(R_l)$ denotes the per-resource total delay, which is defined as the worst-case total delay that one instance of Γ_i experiences due to higher priority transactions Γ_j, $j = 1, \cdots, i-1$ at resource R_l.

In order to further derive the total delay $TD_i^j(R)$ in (10), we introduce a notion of the per-resource total window, denoted by $TW_i(R)$, which is defined as the time duration during which an instance of transaction Γ_i has unfinished tasks on resource R. Then, to find $TD_i^j(R)$, we introduce an iterative convergence approach, similarly as in the traditional recursive response time equation [8], [9].

Initially, we set $TD_i^j(R) = 0$ for all the transactions Γ_j, $j = 1, \cdots, i-1$ and for all the resources $R \in \{R_1, R_2, \cdots, R_M\}$. Then, we have the following iterative equation between $TW_i(R)$ and $TD_i^j(R)$:

$$TW_i(R) = \sum_{v_1 \le k \le v_m} e_{i,k} + \sum_{\forall R_l \in \mathbf{R}} \sum_{j=1}^{i-1} TD_i^j(R_l) X_{v_1}^{v_m}(R_l), \qquad (11)$$

where

$$X_{v_1}^{v_m}(R_l) = \begin{cases} 1, & \text{if any of } \{\tau_{i,v_1}, \cdots, \tau_{i,v_m}\} \text{ visits } R_l; \\ 0, & \text{otherwise.} \end{cases}$$

Once $TW_i(R_l)$ for resource R_l is given, $TD_i^j(R_l)$ can be obtained by

$$TD_i^j(R_l) = \sum_{\{a|r_{j,a}=R_l\}} \left(C_i^{j,a}(TW_i(R_l)) \times e_{j,a} \right), \qquad (12)$$

where $C_i^{j,a}(TW_i(R_l))$ denotes the worst-case total number of instances of $e_{j,a}$ attributing to $TD_i^j(R_l)$ in $TW_i(R_l)$. We can calculate $C_i^{j,a}(TW_i(R_l))$ by

$$C_i^{j,a}(TW_i(R_l)) = \min \left[Z_{j,a}(TW_i(R_l)), \sum_{\{k|r_{i,k}=R_l\}} I_{j,a}(i,k) \right], \qquad (13)$$

where $Z_{j,a}(TW_i(R_l)) = \lceil J_{j,a} + TW_i(R_l)/p_j \rceil$ and $I_{j,a}(i,k)$ is the largest possible number of release of task $\tau_{j,a}$ during the busy period of task $\tau_{i,k}$ which can be obtained from (9).

Consequently, by applying (11), (12), and (13) altogether in an iterative manner, we can calculate the total delay $TD_i^j(R)$, which in turn gives the end-to-end response time by (10).

IV. Numerical Study

A. Simulation Setup

We consider a multi-resource system in Fig. 1. Assume that there are four transactions, denoted by $\{\Gamma_1, \Gamma_2, \Gamma_3, \Gamma_4\}$. Our goal is to determine the periods of transactions Γ_1 and Γ_2 while Γ_3 and Γ_4 have fixed periods. In addition, we assume that the shared network in Fig. 1 is a Controller Area Network (CAN) bus, which is considered as one of the resources as

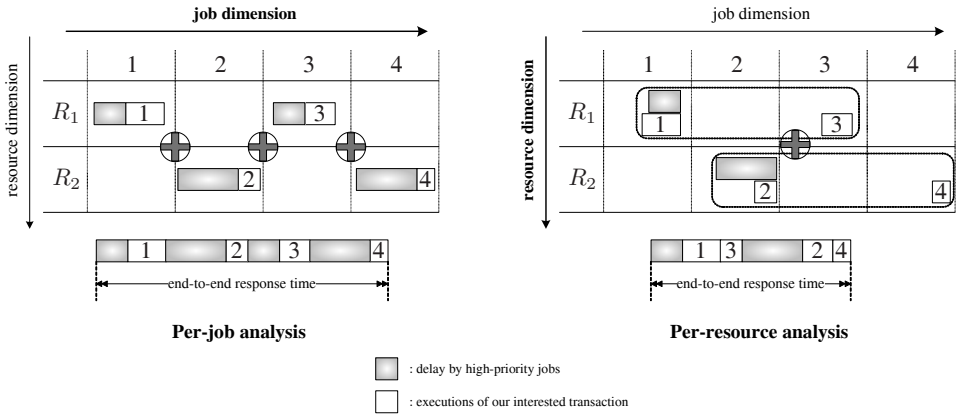


Fig. 3. Conceptual comparison of per-job analysis and per-resource analysis.

	Period	Execution time	
		S_i, C_i, A_i, Node$_i$	CAN
Γ_1	p_1	10	50
Γ_2	p_2	20	60
Γ_3	1500	30	70
Γ_4	3000	40	80

TABLE I
THE PERIODS OF TRANSACTIONS AND THE EXECUTION TIMES OF RESOURCES IN MS.

	Per-job analysis	Per-resource analysis
U_{prim}	$\mathbf{p}^* = (340, \underline{780})$ $\mu(\mathbf{p}^*) = (0.5525, \underline{0.0740})$	$\mathbf{p}^* = (340, \underline{400})$ $\mu(\mathbf{p}^*) = (0.5525, \underline{0.2442})$
U_{ours}	$\mathbf{p}^* = (700, \underline{580})$ $\mu(\mathbf{p}^*) = (0.3303, \underline{0.1377})$	$\mathbf{p}^* = (650, \underline{290})$ $\mu(\mathbf{p}^*) = (0.3532, \underline{0.3510})$

TABLE II
THE OPTIMAL SOLUTION $\mathbf{p}^*$ TO THE RESPECTIVE FORMULATION AND THE CORRESPONDING STABILITY RADIUS.

already explained in Section II-A. The visit sequences of transactions are given as follows:

$$\Gamma 1 : \{S_1, \text{CAN}, C_1, \text{CAN}, A_1\},$$
$$\Gamma 2 : \{S_2, \text{CAN}, C_2, \text{CAN}, A_2\},$$
$$\Gamma 3 : \{N_1, \text{CAN}, C_1, \text{CAN}, N_2, \text{CAN}, C_1, \text{CAN}, N_1\},$$
$$\Gamma 4 : \{N_3, \text{CAN}, C_2, \text{CAN}, N_2, \text{CAN}, C_2, \text{CAN}, N_3\},$$

where S_i, C_i, A_i, $i = 1, 2$ and N_i, $i = 1, 2, 3$ denote Sensor$_i$, Controller$_i$, Actuator$_i$, and Node$_i$ in Fig. 1, respectively. The periods of each transaction and the execution times at each resource are summarized in Table I. The priority of a transaction is given with the rate-monotonic priority assignment. For the dynamics of Plant$_1$ and Plant$_2$, we use $\lambda_1 = 1$ and $\lambda_2 = 3$, respectively, and $B = 1$ for both plants.

B. Comparison of the Stability Regions

Due to the page limit, here we only show the numerical result of the robustness performance of the overall system. In particular, in order to see the effect of both the control metric and the response time analysis, we introduce the following objective in (1) for comparison with our objective function: $U_{primitive}(\mathbf{p}) = -\sum_{i=1}^{N} p_i$.

With the introduction of $U_{primitive}$, we have the following four combinations for solving the optimization problem of (1): ($U_{primitive}$, per-job analysis), ($U_{primitive}$, per-resource analysis), (U_{ours}, per-job analysis), and (U_{ours}, per-resource analysis). Table II shows the optimal solution $\mathbf{p}^* = (p_1^*, p_2^*)$ to each combination and the corresponding stability radius $\mu(\mathbf{p}^*) = (\mu_i(p_1^*), \mu_i(p_2^*))$. Note that the underlined values are the period that gives the smallest stability radius and the corresponding stability radius between p_1^* and p_2^*.

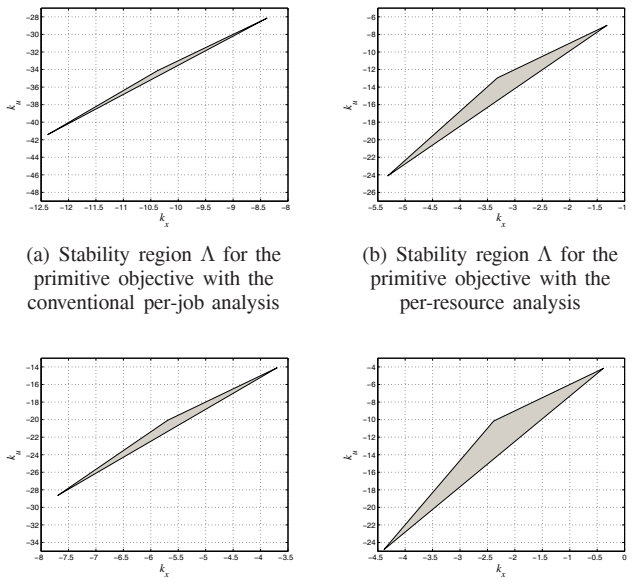
In Table II, if we compare the results in each column, we can notice that $U_{primitive}$ gives a smaller aggregate of the periods than U_{ours}. In fact, we can easily expect this result from the objective of each formulation. However, the smallest stability radius (underlined in Table II) is smaller with $U_{primitive}$. This fact indicates that $U_{primitive}$ improves the control performance of one transaction at the expense of the other one, which results in a severe unbalance between the stability radius of two transactions.

The stability region of the transaction with the smallest stability radius in Table II are shown in Fig. 4. As we can expect from the analysis, the stability region increases either with the per-resource analysis or with the proposed objective. By comparing Fig. 4(a) and Fig. 4(d), we can conclude that our approach significantly increases the stability region.

V. RELATED WORK

An early work on integration of real-time scheduling and control design was carried out by Seto *et al.* [10], where an optimal sampling period selection algorithm was proposed under the assumption that control performance monotonically increases as the periods decrease. In [11], RMA schedulability are formulated as an integer programming to obtain all the feasible periods of a task set, and then the optimal periods are derived by evaluating a given cost function. Overviews on scheduling and control co-design can be found in [12] and [13].

Palopoli *et al.* [4] presented a rigorous optimization approach for scheduling and control co-design in a single-



(a) Stability region Λ for the primitive objective with the conventional per-job analysis

(b) Stability region Λ for the primitive objective with the per-resource analysis

(c) Stability region Λ for our robustness objective with the conventional per-job analysis

(d) Stability region Λ for our robustness objective with the per-resource analysis

Fig. 4. Stability region Λ of the transaction with the smallest stability radius.

resource system under the utilization bound constraint. Our control design follows the approach in [4] by adopting the notion of the stability radius. In the meantime, it should be noted that our co-design formulation differs from previous studies in that we study the scheduling and control co-design in *multi-resource systems* with the *end-to-end response time constraints*.

A scheduling problem in multi-resource systems has been investigated in an optimization framework [14], where the multi-resource scheduling has been formulated as minimization of the aggregate response times of transactions under the traditional end-to-end response time constraints. A period assignment problem has been also tackled in [15], where an optimization approach has been proposed by taking into account the control delay.

From the perspective of real-time schedulability theory, the classic work of Joseph *et al.* [9] presented the worst-case response time analysis for multiple tasks on a single processor fixed-priority scheduling system. This analysis was extended by Tindell *et al.* for arbitrary deadlines [8] and distributed systems with multiple resources [7]. These studies have been further extended in many ways; reducing or eliminating the jitters [16], [17], or considering precedence and timing relations among jobs [18]–[20]. However, all these studies are based on Tindell's per-job analysis in [7] and hence have a common fundamental issue of the multiple visit problem.

The delay composition theorem of [21] and [22] respectively considered the overlapped executions in pipelined distributed systems and in distributed acyclic systems to reduce the overestimation of the per-job end-to-end delay analysis. However, these approaches are not applicable to our cases where transactions visit resources multiple times in arbitrary manners.

Our end-to-end response time analysis in this work mainly relies on that in [3], of which the per-resource response time analysis significantly reduces the overestimation of the per-job analysis caused by the multiple visit problem.

VI. Conclusion

In this paper, we have investigated the problem of real-time scheduling and control co-design in a multi-resource CPS from the perspective of the robustness. We expect that our framework will provide an effective framework for design of a robust CPS.

References

[1] J. A. Stankovic, I. Lee, A. Mok, and R. Rajkumar, "Opportunities and obligations for physical computing systems," *IEEE Computer*, vol. 38, no. 11, pp. 23–31, November 2005.

[2] R. I. Davis, A. Burns, R. J. Bril, and J. J. Lukkien, "Controller Area Network (CAN) schedulability analysis: Refuted, revisited and revised," *Real-Time Systems*, vol. 35, no. 3, pp. 239–272, April 2007.

[3] M.-K. Yoon, C.-G. Lee, and J. Han, "Migrating from per-job analysis to per-resource analysis for tighter bounds of end-to-end response times," *IEEE Transactions on Computers*, vol. 59, no. 7, pp. 933–942, July 2010.

[4] L. Palopoli, C. Pinello, A. Bicchi, and A. L. Sangiovanni-Vincentelli, "Maximizing the stability radius of a set of systems under real-time scheduling constraints," *IEEE Transactions on Automatic Control*, vol. 50, no. 11, pp. 1790–1795, 2005.

[5] K. J. Åström and B. Wittenmark, *Computer-Controlled Systems*. Prentice Hall, 1997.

[6] R. Tempo, G. Calafiore, and F. Dabbene, *Randomized Algorithms for Analysis and Control of Uncertain Systems*. Springer, 2007.

[7] K. Tindell and J. Clark, "Holistic schedulability analysis for distributed hard real-time systems," *Microprocessing and Microprogramming - Euromicro Journal*, vol. 40, no. 2-3, pp. 117–134, April 1994.

[8] K. Tindell, A. Burns, and A. Wellings, "An Extendible Approach for Analyzing Fixed Priority Hard Real-Time Tasks," *J. Real-Time Systems*, vol. 6, no. 2, pp. 133–151, March 1994.

[9] M. Joseph and P. Pandya, "Finding Response Times in a Real-Time System," *BCS Computer J.*, vol. 29, no. 5, pp. 390–395, Oct. 1986.

[10] D. Seto, J. P. Lehoczky, L. Sha, and K. G. Shin, "On task schedulability in real-time control systems," in *Proceedings of the 17th Real-Time Systems Symposium*, December 1996.

[11] D. Seto, J. P. Lehoczky, and L. Sha, "Task Period Selection and Schedulability in Real-Time Systems," in *Proceedings of the 19th Real-Time Systems Symposium*, December 1998.

[12] K.-E. Årzén, A. Cervin, J. Eker, and L. Sha, "An Introduction to Control and Scheduling Co-Design," in *Proc. of the 39th IEEE Conference on Decision and Control*, December 2000.

[13] D. Simon, P. Hokayem, J. Lygeros, and E. Camacho, "State of the art in control/computing co-design," The FeedNetback Project (http://www.feednetback.eu/), Tech. Rep., March 2009.

[14] A. Davare, Q. Zhu, M. D. Natale, C. Pinello, S. Kanajan, and A. L. Sangiovanni-Vincentelli, "Period optimization for hard real-time distributed automotive systems," in *Proc. of the 44th Design Automation Conference*, June 2007.

[15] E. Bini and A. Cervin, "Delay-aware period assignment in control systems," in *Proc. of the 29th Real-Time Systems Symposium*, December 2008.

[16] J. Sun and J. Liu, "Bounding the end-to-end response times of tasks in a distributed real-time system using the direct synchronization protocol," Department of Computer Science, University of Illinois at Urbana-Champaign, Tech. Rep. UIUCDCS-R-96-1949, June 1996.

[17] J. C. Palencia, J. J. G. García, and M. G. Harbour, "Best-case analysis for improving the worst-case schedulability test for distributed hard real-time systems," in *Proc. of the 10th Euromicro Workshop on Real-Time Systems*, 1998, pp. 35–44.

[18] J. C. Palencia and M. G. Harbour, "Schedulability analysis for tasks with static and dynamic offsets," in *Proc. of the 19th IEEE Real-Time Systems Symp.*, Dec. 1998, pp. 26–37.

[19] ——, "Exploiting precedence relations in the schedulability analysis of distributed real-time systems," in *Proc. IEEE 20th Real-Time Systems Symp.*, Dec. 1999, pp. 328–339.

[20] R. Henia and R. Ernst, "Improved offset-analysis using multiple timing-references," in *Proc. of the conference on Design, Automation and Test in Europe*, March 2006, pp. 450–455.

[21] P. Jayachandran and T. Abdelzaher, "A Delay Composition Theorem for Real-Time Pipelines," in *Proc. of the 19th Euromicro Conference on Real-Time Systems*, July 2007.

[22] ——, "Transforming Distributed Acyclic Systems into Equivalent Uniprocessors Under Preemptive and Non-Preemptive Scheduling," in *Proc. of the 20th Euromicro Conference on Real-Time Systems*, July 2008.