Report Number: WUCSE-2004-57

2004-10-05

DEUCON: Distributed End-to-End Utilization Control for Real-Time Systems

Xiaorui Wang, Chenyang Lu, and Xenofon Koutsoukos

This paper presents the Distributed End-to-end Utiization CONtrol (DEUCON) algorithm. DEUCON can dynamically enforce desired CPU utilizations on all processors in a dis-tributed real-time system despite uncertainties in the system workload. In contrast to earlier centralized control schemes, DEUCON is a distributed control algorithm that is system-atically designed based on the Distributed Model Predictive Control theory. We decompose the global multi-processor utilization control problem into a set of localized subprob-lems, and design a peer-to-peer control structure where each local controller only needs to coordinate with a small number of neighbor processors. DEUCON can provide utilization guarantees similar to a... Read complete abstract on page 2.

DEUCON: Distributed End-to-End Utilization Control for Real-Time Systems

Xiaorui Wang, Chenyang Lu, and Xenofon Koutsoukos

Complete Abstract:

This paper presents the Distributed End-to-end Utiization CONtrol (DEUCON) algorithm. DEUCON can dynamically enforce desired CPU utilizations on all processors in a dis-tributed real-time system despite uncertainties in the system workload. In contrast to earlier centralized control schemes, DEUCON is a distributed control algorithm that is system-atically designed based on the Distributed Model Predictive Control theory. We decompose the global multi-processor utilization control problem into a set of localized subprob-lems, and design a peer-to-peer control structure where each local controller only needs to coordinate with a small number of neighbor processors. DEUCON can provide utilization guarantees similar to a centralized control algorithm, while significantly reducing the per-controller run-time overhead in terms of both computation and communication. Further-more, it can tolerate considerable network delay and indi-vidual processor failures. Consequently, DEUCON can pro-vide scalable and robust utilization control services for large distributed real-time systems that operate in unpredictable environments.

Department of Computer Science & Engineering

Washington University in St. Louis
School of Engineering & Applied Science

2004-57

DEUCON: Distributed End-to-End Utilization Control for Real-Time Systems

Authors: Wang, Xiaorui; Lu, Chenyang; Koutsoukos, Xenofon

October 5, 2004

Abstract: This paper presents the Distributed End-to-end Utiization CONtrol (DEUCON) algorithm. DEUCON can dynamically enforce desired CPU utilizations on all processors in a distributed real-time system despite uncertainties in the system workload. In contrast to earlier centralized control schemes, DEUCON is a distributed control algorithm that is systematically designed based on the Distributed Model Predictive Control theory. We decompose the global multi-processor utilization control problem into a set of localized subproblems, and design a peer-to-peer control structure where each local controller only needs to coordinate with a small number of neighbor processors. DEUCON can provide utilization guarantees similar to a centralized control algorithm, while significantly reducing the per-controller run-time overhead in terms of both computation and communication. Furthermore, it can tolerate considerable network delay and individual processor failures. Consequently, DEUCON can provide scalable and robust utilization control services for large distributed real-time systems that operate in unpredictable environments.

DEUCON: Distributed End-to-End Utilization Control for Real-Time Systems

Xiaorui Wang Chenyang Lu Xenofon Koutsoukos
Washington University in St. Louis *Vanderbilt University*
{*wang, lu*}*@cse.wustl.edu* *xenofon.koutsoukos@vanderbilt.edu*

Abstract

This paper presents the Distributed End-to-end Utiization CONtrol (DEUCON) algorithm. DEUCON can dynamically enforce desired CPU utilizations on all processors in a distributed real-time system despite uncertainties in the system workload. In contrast to earlier centralized control schemes, DEUCON is a distributed control algorithm that is systematically designed based on the Distributed Model Predictive Control theory. We decompose the global multi-processor utilization control problem into a set of localized subproblems, and design a peer-to-peer control structure where each local controller only needs to coordinate with a small number of neighbor processors. DEUCON can provide utilization guarantees similar to a centralized control algorithm, while significantly reducing the per-controller run-time overhead in terms of both computation and communication. Furthermore, it can tolerate considerable network delay and individual processor failures. Consequently, DEUCON can provide scalable and robust utilization control services for large distributed real-time systems that operate in unpredictable environments.

1 Introduction

Recent years have seen rapid growth of Distributed Real-time Embedded (DRE) applications executing in *unpredictable* environments in which workloads are unknown and vary significantly at run-time. Such systems include data-driven and open systems whose execution is heavily influenced by volatile environments. For example, task execution times in vision-based feedback control systems depend on the content of live camera images of changing environments [10]. Likewise, the supervisory control and data acquisition (SCADA) systems for power grid control may experience dramatic load increase during a cascade power failure [6]. Furthermore, as DRE systems become connected to the Internet, they are exposed to load disturbances due to variable user requests and even cyber attacks [6].

As DRE systems executing in unpredictable environments become increasingly important to our society, a new paradigm of real-time computing based on *Adaptive QoS Control (AQC)* has received significant attention. In contrast to traditional approaches to real-time systems that rely on accurate knowledge about system workload, AQC can provide robust QoS guarantees in unpredictable environments by adapting to workload variations based on online feedback. A key advantage of AQC is that it adopts a control-theoretic framework for systematically developing adaptation strategies based on formal control analysis. This rigorous design methodology is in sharp contrast to heuristic-based adaptive solutions that rely on extensive empirical evaluation and manual tuning.

As a key step toward an AQC framework for *distributed* real-time embedded systems, we are developing adaptive *utilization control* algorithms. The goal of utilization control is to enforce desired CPU utilizations on all the processors in a distributed system despite significant uncertainties in system workloads. Utilization control can be used to enforce appropriate schedulable utilization bounds on all processors to guarantee end-to-end task deadlines. It can also enhance system survivability by providing overload protection against workload fluctuation.

DRE systems introduce many new research challenges for AQC that have not been addressed in earlier work on single-processor systems. They require *multi-input-multi-output (MIMO)* control solutions to manage the system QoS on multiple processors. More importantly, the QoS of different processors are often *coupled* with each other due to complex interactions among distributed application components. In particular, many DRE systems employ the common *end-to-end task model* [14], where a task may be comprised of a chain of subtasks on different processors. In such systems, the CPU utilizations of different processors cannot be controlled independently from others. For example, changing the rate of a task will affect the load on all the processors where its subtasks are located. Therefore, the coupling among processors must be modeled and addressed in the design of QoS control algorithms.

Our earlier work in this area produced EUCON (End-to-end Utilization CONtrol) [17], the first control-theoretic utilization control algorithm designed for DRE systems with end-to-end tasks. EUCON can maintain desired CPU utilizations on multiple processors despite uncertainties in task execution times and coupling among processors. It employs a *centralized* MIMO model predictive controller to manage and coordinate the adaptation of multiple processors in a DRE system. While it is a promising starting point and suit-

able for small-scale DRE systems, this centralized control scheme has several limitations. Since its communication and computation overhead depends on the size of an *entire* DRE system, it cannot handle large-scale systems (e.g. wide-area power grid management and ubiquitous smart spaces). Furthermore, the processor executing the controller is a single point of failure since the entire system will lose the capability of QoS adaptation if it fails.

In this paper, we present a new *distributed* control algorithm called Distributed End-to-end Utiization CONtrol (*DEUCON*) that significantly enhance the scalability and reliability of utilization control for DRE systems. In sharp contrast to the centralized control scheme adopted by EUCON, DEUCON features a novel peer-to-peer control structure where each processor has an efficient *local* controller that only coordinates with a small number of neighbor controllers. This feature allows DEUCON to scale well in large distributed systems and tolerate individual processor failures. To our best knowledge, DEUCON is the first *distributed* QoS control algorithm designed for DRE systems with end-to-end tasks. The primary contributions of this paper are three-fold.

- A new scheme for decomposing the global multiprocessor utilization control problem into a set of localized subproblems to facilitate the design of distributed control solutions.
- A novel peer-to-peer control structure and localized utilization control algorithm designed based on *Distributed Model Predictive Control* (DMPC) theory [5].
- Simulation results showing that DEUCON can provide robust utilization guarantees to multiple processors through task rate adaptation, while introducing only a fraction of computation and communication overhead of a centralized solution.

The rest of this paper is organized as follows. Section 2 formulates the end-to-end utilization control problem. Section 3 revisits the EUCON algorithm as a starting point for this work. Section 4 presents the design and analysis of DEUCON. Section 5 evaluates DEUCON with simulations. Section 6 reviews related work. The paper concludes with Section 7.

2 End-to-End Utilization Control

In this section we formulate the end-to-end utilization control problem for DRE systems.

2.1 Task Model

We adopt an end-to-end task model [14] implemented by many DRE applications. A system is comprised of m periodic tasks $\{T_i|1 \leq i \leq m\}$ executing on n processors $\{P_i|1 \leq i \leq n\}$. Task T_i is composed of a chain of subtasks $\{T_{ij}|1 \leq j \leq n_i\}$ located on different processors. The release of subtasks is subject to precedence constraints, i.e., subtask $T_{ij}(1 < j \leq n_i)$ cannot be released for execution until its predecessor subtask T_{ij-1} is completed. If a non-greedy synchronization protocol (e.g., release guard [26]) is used to enforce the precedence constraints, all the subtasks of a periodic task share the same rate as the first subtask. Therefore, the rate of a task (and all its subtasks) can be adjusted by changing the rate of its first subtask. In this paper, the processor hosting the first subtask of a task is called its *master processor*. Only a task's master processor can change its rate.

Our task model has two important properties. First, while each subtask T_{ij} has an *estimated* execution time c_{ij} available at design time, its *actual* execution time may be different from its estimation and vary at run time. Modeling such uncertainty is important to DRE systems operating in unpredictable environments. Second, the rate of a task T_i may be dynamically adjusted within a range $[R_{min,i}, R_{max,i}]$. This assumption is based on the fact that the task rates in many DRE applications (e.g., digital control [19][22], sensor update, and multimedia [3]) can be dynamically adjusted without causing system failure.

Each task T_i is subject to an end-to-end relative deadline related to its period. In an end-to-end scheduling approach [26], the deadline of an end-to-end task is divided into subdeadlines of its subtasks [11][20]. Hence the problem of meeting the deadline is transformed to the problem of meeting the subdeadline of each subtask. A well known approach for meeting the subdeadlines on a processor is to ensure its utilization remains below its schedulable utilization bound [12][13].

2.2 Problem Formulation

Utilization control can be formulated as a dynamic constrained optimization problem. We first introduce several notations. T_s, the sampling period, is selected so that multiple instances of each task may be released during a sampling period. $u_i(k)$ is the CPU utilization of processor P_i in the k^{th} sampling period, i.e., the fraction of time that P_i is not idle during time interval $[(k-1)T_s, kT_s]$. B_i is the desired utilization set point on P_i. $r_j(k)$ is the invocation rate of task T_j in the $(k+1)^{th}$ sampling period.

Given the utilization set point vector, $\mathbf{B} = [B_1 \dots B_n]^T$ and the rate constraints $[R_{min,j}, R_{max,j}]$ for each task T_j, the control goal at k^{th} sampling point (time kT_s) is to dynamically choose task rates $\{r_j(k)|1 \leq j \leq m\}$ to minimize the difference between B_i and $u_i(k)$ for all processors:

$$\min_{\{r_j(k)|1 \leq j \leq m\}} \sum_{i=1}^{n} (B_i - u_i(k+1))^2 \quad (1)$$

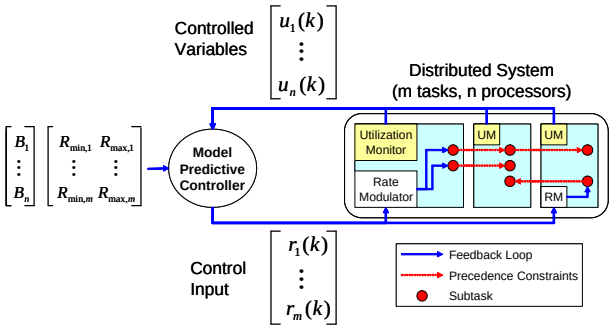


Figure 1. EUCON's MIMO feedback loop with a centralized controller

subject to constraints

$$R_{min,j} \leq r_j(k) \leq R_{max,j} \quad (1 \leq j \leq m)$$

The rate constraints ensure all tasks remain within their acceptable rate ranges. The optimization formulation maximizes task rates by making the utilization of each processor as close to its set point as allowed by the constraints. The design goal is to ensure that all processors quickly converge to their utilization set points after a workload variation, whenever it is feasible under the rate constraints. Therefore, to guarantee end-to-end deadlines, a user only needs to specify the set point of each processor to be a value below its schedulable utilization bound. Utilization control algorithms can be used to meet all the end-to-end deadlines by enforcing the set points of all the processors in a DRE system.

3 EUCON Revisited

In this section we briefly describe the EUCON algorithm developed in our earlier work [17], which provides a starting point and baseline for DEUCON.

3.1 Feedback Control Loop

As shown in Figure 1, EUCON features a MIMO feedback control loop composed of a central controller and a utilization monitor and rate modulator on each processor. EUCON is invoked periodically at each sampling point k. The controlled variables are the utilizations of all processors, $\mathbf{u(k)} = [u_1(k)...u_n(k)]^T$. The control inputs from the controller are the change in task rates $\mathbf{\Delta r(k)} = [\Delta r_1(k) \ldots r_m(k)]^T$, where $\Delta r_i(k) = r_i(k) - r_i(k-1)$ $(1 \leq i \leq m)$.

The feedback control loop works as follows: (1) the utilization monitor on each processor P_i sends its utilization $u_i(k)$ in the last sampling period $[(k-1)T_s, kT_s)$ to the central controller; (2) the controller collects the utilization vector $\mathbf{u(k)} = [u_1(k) \ldots u_n(k)]^T$ including the utilizations of all processors, computes a new rate change vector $\mathbf{\Delta r(k)} = [\Delta r_1(k) \ldots \Delta r_m(k)]^T$, and sends the new task rates $\mathbf{r(k)} = \mathbf{r(k-1)} + \mathbf{\Delta r(k)}$ to the rate modulators on master processors (i.e., processors that master at least one task); and (3) the rate modulators on master processors change the rates of tasks according to $\mathbf{r(k)}$.

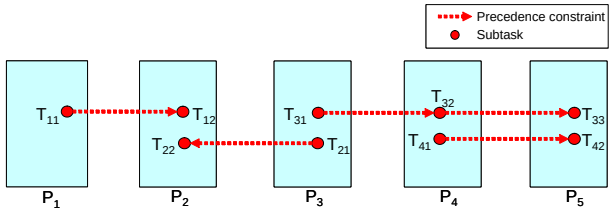


Figure 2. An example DRE system

3.2 Global System Model

Following a control-theoretical methodology, the EUCON controller is designed based on a set of difference equations that model the dynamics of the *whole* DRE system. A DRE system is described by the following *global* system model:

$$\mathbf{u(k+1)} = \mathbf{u(k)} + \mathbf{GF\Delta r(k)} \quad (2)$$

The vector $\mathbf{\Delta r(k)}$ represents the changes in task rates. The *subtask allocation matrix*, $\mathbf{F}$, is an $n \times m$-order matrix, where $f_{ij} = c_{jl}$ if a subtask T_{jl} of task T_j is allocated to processor P_i, and $f_{ij} = 0$ if no subtask of task T_j is allocated to processor P_i. $\mathbf{F}$ captures the *coupling* among processors due to end-to-end tasks. $\mathbf{G}$ is an $n \times n$ diagonal matrix in which g_{ii} represents the ratio between the change in the actual utilization and its estimation. The exact value of g_i is *unknown* due to the unpredictability in execution times. Note that $\mathbf{G}$ describes the effect of uncertainty in workload on the utilization of a DRE system. As an example, Figure 2 shows a DRE system with five processors and four end-to-end tasks. It is modeled by (2) with the following parameters:

$$\mathbf{u(k)} = \begin{bmatrix} u_1(k) \\ u_2(k) \\ u_3(k) \\ u_4(k) \\ u_5(k) \end{bmatrix}, \mathbf{G} = \begin{bmatrix} g_1 & 0 & 0 & 0 & 0 \\ 0 & g_2 & 0 & 0 & 0 \\ 0 & 0 & g_3 & 0 & 0 \\ 0 & 0 & 0 & g_4 & 0 \\ 0 & 0 & 0 & 0 & g_5 \end{bmatrix},$$

$$\mathbf{F} = \begin{bmatrix} c_{11} & 0 & 0 & 0 \\ c_{12} & c_{22} & 0 & 0 \\ 0 & c_{21} & c_{31} & 0 \\ 0 & 0 & c_{32} & c_{41} \\ 0 & 0 & c_{33} & c_{42} \end{bmatrix}, \mathbf{\Delta r(k)} = \begin{bmatrix} \Delta r_1(k) \\ \Delta r_2(k) \\ \Delta r_3(k) \\ \Delta r_4(k) \end{bmatrix}$$

The EUCON controller was designed based on the above global system model using the *Model Predictive Control (MPC)* theory [18]. The detailed controller design is described in [17].

3.3 Limitations

EUCON relies on a central controller to manage the adaptation of multiple processors in a DRE system. A centralized control scheme has several disadvantages. First, the run-time overhead depends on the size of an entire DRE system. Specifically, the worst-case computational complexity of the central MPC controller is polynomial to the total number of tasks and the total number of processors in the system. Furthermore, since every processor in the system needs to communicate with the controller in every sampling period,

the processor that runs the controller can become a communication bottleneck. Therefore, a centralized control scheme cannot scale effectively in large DRE systems. Second, the control design of EUCON assumes that communication delays between the control processor and other processors are negligible compared to the sampling period of the controller. This assumption may not hold in networks with significant delays such as the Internet and wireless sensor networks. In addition, the processor executing the controller is a single point of failure. The entire system will lose the capability to adapt to the environment if it fails.

Centralized solutions are therefore more appropriate for small-scale DRE systems (e.g., control systems in a power generation facility) than for large-scale DRE systems (e.g., wide-area power grid management). An important task is to develop distributed control algorithms to improve the scalability and reliability of adaptive utilization control in DRE systems. This is the focus of this paper.

4 Design of DEUCON

In this section, we present the design of DEUCON. In contrast to the centralized control scheme adopted by EUCON, DEUCON employs a peer-to-peer control structure with a separate local controller C_i on each master processor P_i. Each controller only coordinates with a small number of processors called its (logical) *neighbors*. As a result, the computation and communication overhead of each controller is a function of the size of the processor's neighborhood instead of the entire system.

4.1 Problem Decomposition

A key step in our control design is to decompose the global utilization control problem into a set of localized subproblems. From a controller C_i's perspective, the goal of decomposition is to partition the set of system variables into three subsets, including *local variables* on host processor P_i, *neighbor variables* on P_i's neighbors, and all other variables in the system. C_i's subproblem only includes its local and neighbor variables. A key feature of our decomposition scheme is that it balances two conflicting goals. On one hand, the number of neighbor variables should be minimized to improve system scalability. On the other hand, the neighbor variables must capture the coupling among processors so that local controllers can achieve global system stability through coordination in their neighborhoods.

We first give several definitions before presenting our decomposition scheme.

Definition Processor P_j is P_i's *direct neighbor* if P_j has a subtask belonging to an end-to-end task mastered by P_i.

Definition The *concerned tasks* of P_i are the tasks which have subtasks located on P_i or P_i's direct neighbors.

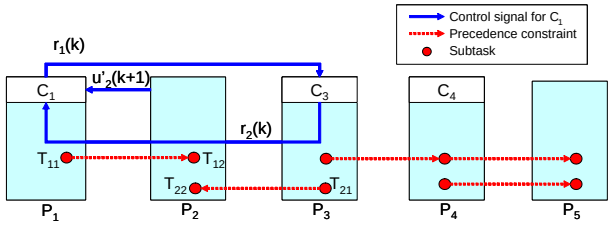


Figure 3. Data exchange between C_1 and its neighbors (other data exchanges are not shown)

Definition Processor P_j is P_i's *indirect neighbor* if (1) P_j is the master processor of any of P_i's concerned tasks and (2) P_j is not P_i's direct neighbor or P_i itself.

The subproblem of a controller includes a set of utilizations as *controlled variables*, and a set of task rates as *manipulated variables*. In our decomposition scheme, the controlled variables of controller C_i include $u_i(k)$, the host processor P_i's utilization, and $UD_i(k)$, the set of utilizations of P_i's direct neighbors. $UD_i(k)$ are considered C_i's neighbor variables because they are affected by the rates of tasks mastered by P_i. Since each concerned task contributes to the utilizations of P_i and/or its direct neighbors, C_i's manipulated variables include the rates of all of P_i's concerned tasks. Note that a concerned task may be mastered by P_i itself, its direct neighbor, or its indirect neighbor.

Figure 3 shows the same system as in Figure 2. We consider controller C_1 as an example. P_1 has one direct neighbor (P_2) due to task T_1 mastered by P_1. Its concerned tasks include T_1 and T_2 (which has a subtask on direct neighbor P_2). Hence P_3, the master processor of T_2, is P_1's indirect neighbor. Therefore, C_1 has two controlled variables, $u_1(k)$ and $u_2(k)$, and two manipulated variables $r_1(k)$ and $r_2(k)$.

Let set $NR_i(k)$ includes the rates of all of P_i's concerned tasks, and set $NU_i(k) = \{u_i(k)\} \cup UD_i(k)$, the subproblem of C_i then becomes the following localized constrained optimization problem within its neighborhood:

$$\min_{NR_i(k)} \sum_{u_l(k) \in NU_i(k)} (B_l - u_l(k+1))^2 \qquad (3)$$

subject to

$$R_{min,j} \le r_j(k) \le R_{max,j} \quad (r_j(k) \in NR_i(k))$$

In contrast to the global model (2) used in EUCON, each controller in DEUCON has a localized model which only includes its local and neighbor variables. This local model of C_i is described as:

$$\mathbf{nu_i(k+1)} = \mathbf{nu_i(k)} + \mathbf{G_i F_i \Delta nr_i(k)} \qquad (4)$$

where $\mathbf{nu_i(k)}$ and $\mathbf{nr_i(k)}$ are vectors comprised of all elements in $NU_i(k)$ and $NR_i(k)$, respectively. $\mathbf{G_i}$ and $\mathbf{F_i}$ are defined in the same way as $\mathbf{G}$ and $\mathbf{F}$ in (2), but regarding only the processors in $NU_i(k)$ and the task rates in $NR_i(k)$.

For example, the controller C_1 shown in Figure 3 is modeled with the following parameters.

$$\mathbf{nu_1(k)} = \begin{bmatrix} u_1(k) \\ u_2(k) \end{bmatrix}, \mathbf{G_1} = \begin{bmatrix} g_1 & 0 \\ 0 & g_2 \end{bmatrix},$$
$$\mathbf{F_1} = \begin{bmatrix} c_{11} & 0 \\ c_{12} & c_{22} \end{bmatrix}, \mathbf{\Delta nr_1(k)} = \begin{bmatrix} \Delta r_1(k) \\ \Delta r_2(k) \end{bmatrix}$$

From (4), C_1's local model is

$$\begin{aligned} u_1(k+1) &= u_1(k) + g_1 c_{11} \Delta r_1(k) \\ u_2(k+1) &= u_2(k) + g_2(c_{12}\Delta r_1(k) + c_{22}\Delta r_2(k)) \end{aligned}$$

Compared to the global model (2) which includes 5 processors and 4 tasks, C_1's local model only involves 2 processors and 2 tasks. This indicates that a localized subproblem has lower complexity than the global one.

4.2 Localized Feedback Control Loop

We now present DEUCON's localized feedback control loop based on our decomposition scheme. The execution of a controller C_i at each sampling point k (time kT_s) includes three steps:

1. *Local control computation:* C_i executes a MPC algorithm to solve its local subproblem. The feedback input to the control algorithm includes (1) $u_i(k)$ from the local utilization monitor, (2) a set of *predicted utilizations* $UD_i'(k)$ of its direct neighbors, and (3) the rates of concerned tasks, $NR_i(k-1)$ in the last sampling period. The output from the controller C_i includes the new rates for concerned tasks, $NR_i(k)$. The details of the control algorithm are presented in Section 4.3.

2. *Local actuation:* The local rate modulator on P_i changes the rates of the set of tasks mastered by P_i according to the control input from C_i. The other task rates in the control input will be ignored because they are not mastered by P_i.

3. *Data exchange among neighbors:* C_i sends its *predicted utilization* at the next sampling point, $u_i'(k+1)$, to other controllers of which it serves as a direct neighbor. C_i also sends the rates of tasks mastered by P_i to those controllers which have these tasks as their concerned tasks. In addition, C_i receives new predicted utilizations from its direct neighbors, and the actual rates of the concerned tasks which are not mastered by itself, from its direct and indirect neighbors. They will be used for the local control computation at the next sampling point $(k+1)$.

An important feature of our control design is that it can tolerate considerable network delays among neighbors. Note that in step 1, the *predicted* utilizations $UD_i'(k)$ (instead of $UD_i(k)$) are provided by C_i's direct neighbors in the previous sampling period. This is because $UD_i(k)$ is not instantaneously available to C_i at time kT_s due to network delays. $UD_i'(k)$ is predicted based on $UD_i(k-1)$ as a substitute for $UD_i(k)$ to be transmitted over the network during interval $[(k-1)T_s, kT_s]$. Each element $u_j'(k) \in UD_i'(k)$ is calculated using the following reference trajectory from measured utilization $u_j(k-1)$ to its set point B_j over the following P sampling periods.

$$ref_j(k+l|k) = B_j - e^{-\frac{T_s}{T_{ref}}l}(B_j - u_j(k-1)) \quad (1 \le l \le P) \tag{5}$$

where T_{ref} is the time constant that specifies the speed of system response. P is called the *prediction horizon*. The value of $ref_j(k+1|k)$ is assigned to $u_j'(k)$. Since $UD_i'(k)$ can take the whole last sampling period to transmit, DEUCON relaxes the requirement of instant utilization transmission which is essential for EUCON.

4.3 Controller Design

In contrast to EUCON which has only *one* global controller, every *master* processor in DEUCON has a controller. Non-master processors do not need controllers because they can not change the rate of any task. For example, in Figure 3, processors P_1, P_3 and P_4 each have a controller, while P_2 and P_5 do not have a controller because they are not master processors for any tasks. This feature reduces the overhead of DEUCON.

Based on the local system model (4), we apply the DMPC theory [5] to design a local controller C_i. The goal of the controller is to optimize a cost function (7) over P sampling periods, which is the prediction horizon. The objective of optimization is to select an input trajectory that minimizes the predicted cost while satisfying the constraints. The cost is predicted based on an approximated system model (6). The input trajectory includes the control inputs in the following M sampling periods, e.g., $\Delta NR_i(k), \Delta NR_i(k+1|k), \ldots \Delta NR_i(k+M-1|k)$[1], where M is called the *control horizon*. Once the input trajectory is computed, only the first element $\Delta NR_i(k)$ is applied as the input signal to the system. At next sampling point, the prediction horizon slides one sampling period and the input is computed again as a solution to a constrained optimization problem based on the feedbacks $NU_i'(k+1)$ from its direct neighbors and itself, where $NU_i'(k+1) = UD_i'(k+1) \cup \{u_i(k+1)\}$.

The controller includes a least square solver, a cost function, a reference trajectory, and an approximate local system model used to predict the cost in the prediction horizon. The approximate model used by our controller is as the following.

$$\mathbf{nu_i(k+1)} = \mathbf{nu_i'(k)} + \mathbf{F_i \Delta nr_i(k)} \tag{6}$$

The above model has two differences from the actual system model (4). First, the utilizations of direct neighbors are approximated by their predicted utilizations $\mathbf{nu_i'(k)}$, where

[1]The notation $x(k+l|k)$ means that the value of variable x at time $(k+l)T_s$ depends on the conditions at time kT_s.

$\mathbf{nu'_i(k)}$ is a vector comprised of all elements in $NU'_i(k)$. As discussed in Section 4.2, this approximation allows DEUCON to tolerate network delays. Second, because the real system gains $\mathbf{G_i}$ in system model (4) are unknown in unpredicted environments, our controller assumes $\mathbf{G_i} = [1 \dots 1]^T$, i.e., the controller assumes that the estimated execution times are accurate. Although this approximate model is not an exact characterization of the real system, the closed loop system under our controller can still maintain stability and guarantee desired utilization set points as long as $\mathbf{G_i}$ are within a certain range (see results in Section 5.3). This is due to the coordination scheme and online feedbacks used in our distributed model predictive control algorithm.

At every sampling point, the controller computes the control input $\mathbf{\Delta nr_i(k)}$ that minimizes the following cost function under the rate constraints.

$$V_i(k) = \sum_{l=1}^{P} \|\mathbf{nu_i(k+l|k)} - \mathbf{ref_i(k+l|k)}\|^2 + \sum_{l=0}^{M-1} \|\mathbf{\Delta nr_i(k+l|k)} - \mathbf{\Delta nr_i(k+l-1|k)}\|^2 \quad (7)$$

where P is the prediction horizon, and M is the control horizon. The first term in the cost function represents the *tracking error*, i.e., the difference between the utilization vector $\mathbf{nu_i(k+l|k)}$, which is predicted based on (6), and a reference trajectory $\mathbf{ref_i(k+l|k)}$ defined in (5). The controller is designed to track an exponential reference trajectory that converges to the set points so that the closed-loop system behaves like a desired linear system. By minimizing the tracking error, the closed-loop system will also converge to the utilization set point. The second term in the cost function represents the *control penalty*. The control penalty term causes the controller to minimize the changes in the control input.

This constrained optimization problem can be transformed to a standard constrained least square problem. Controller C_i can then use a standard least-sqaure solver to solve this problem on-line. The detailed transformation is not shown due to space limit. The worst-case computation complexity of the solver is polynomial to the numbers of tasks and processors in the localized model (6).

4.4 Algorithm Analysis

We now analyze some theoretical and practical issues of the DEUCON algorithm.

Scalability The controller overhead includes computation overhead and communication overhead. Since a model predictive controller's worst-case computation complexity is polynomial to the number of processors and the number of tasks, EUCON's computation complexity increases as system scale increases. In contast, the local model (4) indicates that the computation overhead in DEUCON is only a function of the neighborhood size. With regard to communication overhead, EUCON needs to receive the utilizations from and send the rate changes to all processors in every sampling period. In the case of DEUCON, each controllers only receives the predicted utilizations from its direct neighbors and the actual rates of a subset of its concerned tasks from its direct and indirect neighbors. Hence, similar to computation, the communication overhead of EUCON is proportional to the entire system while DEUCON depends only on its neighborhood size.

We note that the size of a controller's neighborhood is usually much smaller than the entire DRE system, especially in large systems where scalability is most needed. Even in a DRE system comprised of many processors, an end-to-end task is typically comprised of only a small number of subtasks. In practice, very few real-time tasks have control flows over hundreds of nodes. Therefore, distributed control algorithms based on our decomposition scheme can scale effectively in many large DRE systems.

Tolerance to Network Delay An advantage of DEUCON is that it allows relatively long network delays among neighbors. Since a controller does not need to wait for the current utilization from neighbors for its control computation, the control performance is not sensitive to communication delays as long as they are upper-bounded by the controller's sampling period. This is in contrast to EUCON which requires the network delay to be much shorter than the sampling period. This feature also enables DEUCON to tolerate clock drift among neighbors as long as their differences are small compared to the sampling period.

Stability analysis DEUCON is a feedback control algorithm that at every time step selects the task rates based on the current processor utilizations. Stability analysis is important in order to derive conditions that guarantee that the utilizations will converge to the desired set points. Our previous work has successfully applied MPC stability analysis to EUCON [17]. This is a direct approach based on the eigenvalues of the closed-loop system. The approach can be extended for DEUCON by using the localized system model (4) in place of the global system model (2). Through our analysis, we can derive the range of deviations of execution times from their estimations under which the system remain stable. However, due to the coupling among controllers, our current analysis becomes very complex as the number of tasks, controllers, and prediction and control horizons increase. Stability analysis of DMPC is an active research area. Although there is no general solution, results have been derived for specific classes of systems [5][8]. We observe that the MIMO system model in DEUCON has a special structure since there is coupling in the control inputs, i.e., a change in a task rate (control input) will affect the utilization in all direct neighbors, but there is no coupling between the states, i.e., utilization of a processor does not depend on the utilization of the other processors. A more efficient method for stability analysis of such models is under investigation and out of the scope of this paper.

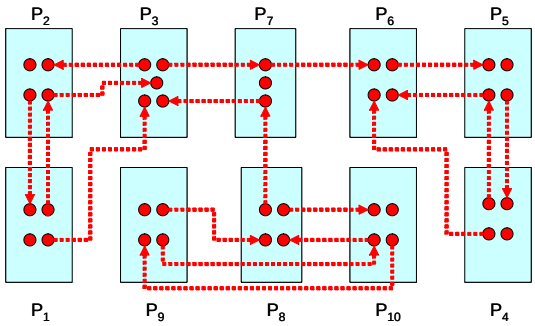


Figure 4. A medium size workload

5 Experimentation

5.1 Experimental Setup

Our simulation environment is composed of an event-driven simulator implemented in C++ and a set of controllers implemented in MATLAB (R12). The simulator implements the utilization monitors, the rate modulators and the distributed real-time system with interface to the controllers. The subtasks on each processor are scheduled by the Rate Monotonic Scheduling (RMS) algorithm [13]. The precedence constraints among subtasks are enforced by the release guard protocol [26]. The controllers are based on the *lsqlin* least square solver in MATLAB. The simulator opens a MATLAB process and initializes all the controllers at start time. In the end of each sampling period, the simulator collects the local utilization, the predicted neighborhood utilizations and the concerned task rates for each controller, and then calls the controller in MATLAB. The controllers compute the control input, $\Delta r(k)$, and return it to the simulator. The simulator then calls the rate modulators on each processor to adjust the rates of its mastered tasks.

Each task's end-to-end deadline $d_i = n_i/r_i(k)$, where n_i is the number of subtasks in task T_i. Each end-to-end deadline is evenly divided into subdeadlines for its subtasks. The resultant subdeadline of each subtask T_{ij} equals its period, $1/r_i(k)$. Hence the schedulable utilization bound of RMS [13], $B_i = m_i(2^{1/m_i} - 1)$ is used as the utilization set point on each processor, where m_i is the number of subtasks on P_i. All (sub)tasks meet their (sub)deadlines if the utilization set point on every processor is enforced.

A medium size workload (as shown in Figure 4) has been used in our experiments. It includes 21 tasks (with a total of 40 subtasks) executing on 10 processors. There are 14 end-to-end tasks running on multiple processors and 7 local tasks. The controller parameters used for this workload include the prediction horizon as 2 and the control horizon as 1. The control period $T_s = 1000$ time units. The time constant T_{ref} used in (5) is set as 4. Specific parameters of tasks are not shown due to space limit.

To evaluate the robustness of DEUCON when execution times deviate from the estimation, the execution time of each subtask T_{ij} can be changed by tuning a parameter called the execution-time factor, $etf_{ij}(k) = c_{ij}/a_{ij}(k)$, where a_{ij} is the actual execution time of T_{ij}. The execution time factor (*etf*) represents how much the actual execution time of a subtask deviates from the estimated one. The *etf* (and hence the actual execution times) may be kept constant or changed dynamically in a run.

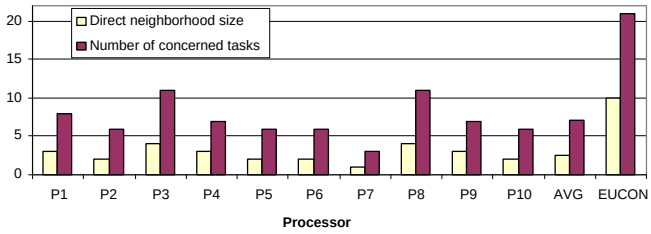


Figure 5. Entire system size vs. neighborhood size

We compare DEUCON against EUCON because EUCON is the only available utilization control algorithm for DRE systems with end-to-end tasks. Our previous results also showed that EUCON significantly outperformed a common open-loop approach that assigns fixed task rates based on estimated execution times [17]. In the following we present two sets of simulations. Experiment I examines DEUCON's overhead. Experiment II evaluates DEUCON's performance under various types of unpredictable workloads.

5.2 Experiment I: Overhead

As discussed in Section 3, a major limitation of centralized controllers is that the run-time overhead is related to the size of the entire system. In contrast, the overhead of each local controller in DEUCON is just a function of its neighborhood size. Figure 5 compares the size of the entire system with the neighborhood size of each processor for the medium size workload. The centralized EUCON controller needs to model all the 10 processors and the 21 tasks in the system while the average for DEUCON controllers is only 2.6 processors and 7.1 tasks, corresponding to a reduction by 74% and 66%, respectively.

To estimate the *average* computation overhead of the controllers, we measure the execution time of the least square solver which dominates the computation cost on a 2GHz Pentium 4 PC with 256MB RAM. In order to minimize the time delay caused by the IPC communication between the simulator and the MATLAB process in each remote command call, we use a single MATLAB command to run this least square solver for 1000 times as a subroutine. The data shown in Figure 6 is the average of those 1000 runs. The average exection time of all controllers in DEUCON is only 62% of EUCON's central controller. We note that the speedup in execution times is not strictly polynomial to the numbers of neighbors and concerned tasks as one would expect from the theoretical complexity of MPC algorithms. This is attributed to difference between the *average* execution time of MATLAB's *lsqlin* solver and the *worst-case* computational complexity in the analysis. In addition, the initialization cost in the optimization calculations is not negligible for relatively small scale problems in our workload.

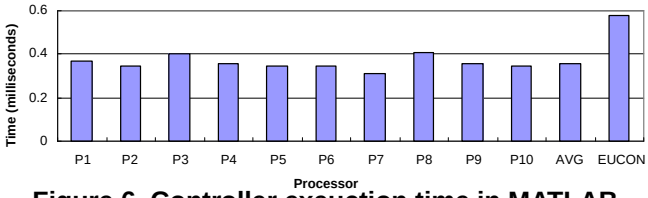


Figure 6. Controller exeuction time in MATLAB

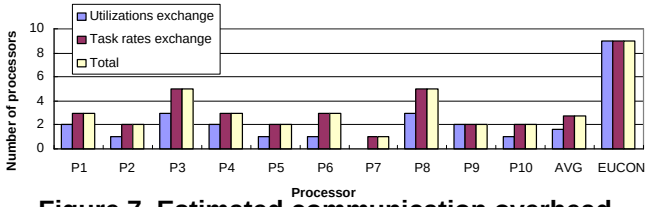


Figure 7. Estimated communication overhead

We now investigate DEUCON's communication overhead. As mentioned in Section 4, a controller's communication overhead is a function of the number of processors communicating with it. To estimate communication overhead due to utilizations exchange, we count the number of processors from which a controller receives the predicted utilizations. This is equal to the number of direct neighbors of the controller. To estimate communication overhead due to task rates exchange, we count the processors from which a controller receives the actual rate changes for one or more of its concerned tasks. The set of total processors communicating with a controller is the union of these two processor sets. From Figure 7 we can see that DEUCON's average estimated per-controller communication overhead is 33% of the EUCON controller's communication overhead.

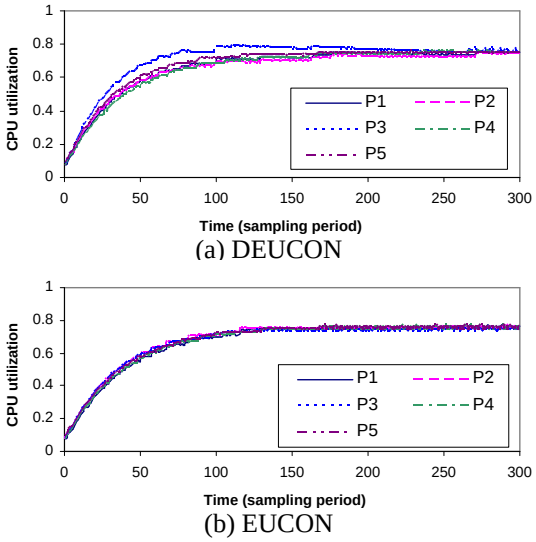
The overhead experiments demonstrate that for this medium workload, DEUCON's per-controller overhead is only 33% (communication) to 62% (computation) of EUCON's centralized controller. As discussed in Section 4.4, when system scale becomes larger, we can expect a bigger difference between DEUCON and EUCON. Although the *total* overhead of all the controllers in DEUCON is higher than EUCON's centralized controller, DEUCON improves the system scalability by distributing the overhead to different master processors in the system. Overhead evaluation of DEUCON in a real middleware environment for large scale system is part of our future work.

5.3 Experiment II: System Peformance

In this subsection we present two sets of simulation experiments. The first one evaluates DEUCON's system performance when task execution times deviate from the estimation. The second experiment tests DEUCON's ability to provide robust performance guarantees when task execution times vary dynamically at run-time.

5.3.1 Steady Execution Times

In this experiment, all subtasks share a fixed execution-time factor (*etf*) in each run. Figure 8(a) and (b) show the utilizations of processors P_1 to P_5 when execution times of tasks



(a) DEUCON

(b) EUCON

Figure 8. CPU utilization of P_1 to P_5 (etf=8)

are *eight times* their estimations. In this case, we can observe a noticable difference in the transient state between DEUCON and EUCON. While the utilizations of EUCON follow the same trajectory, utilizations of DEUCON diverge in the middle of the run and then converge to their set points in the end. The reason for this divergence is that each controller in DEUCON only utilizes local information and makes local decision. Despite this slight difference in transient state, all utilizations converge to their set points within similar settling times. Both DEUCON and EUCON achieve desired utilization guarantees in steady states.

To examine DEUCON's perfromance under different execution time factors, we plot the mean and standard deviation of utilization on P_1 during each run in Figure 9. Every data point is based on the measured utilization $u(k)$ from time $200T_s$ to $300T_s$ to exclude the transient response in the beginning of each run. The system performance is satisfactory if the average utilization is close to the utilization set point, and the standard deviation is small. Both EUCON and DEUCON provide good utilization guarantees for all tested execution-time factors within the etf range $[0.5, 10]$. In this range, the average utilizations under EUCON and DEUCON remain within ± 0.012 of the utilization set points and the standard deviations remain below 0.025. However, when $etf = 8$, DEUCON's performance is slightly worse than that of EUCON, as its average utilization is 0.012 lower than its set point. In addition, EUCON has a high deviation when $etf = 9$, because P_1 has a longer settling time under EUCON. As a result, the system is still in its transient state for part of the interval $[200T_s, 300T_s]$. We also observe that both DEUCON and EUCON suffer a deviation of ± 0.025 when $etf = 0.5$. In general, as discussed in [17], MPC/DMPC algorithms cause oscillation when the execution times are underestimated ($etf < 1$). However, as a key benefit, both DEUCON and EUCON can achieve desired utilizations even when execution times are severely overestimated.

To further investigate the CPU utilizations on other pro-

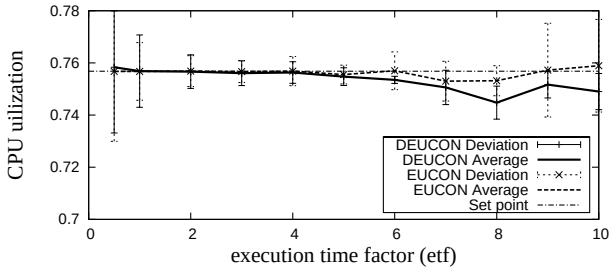


Figure 9. CPU utilization average and deviation on P_1 under different execution-time factors

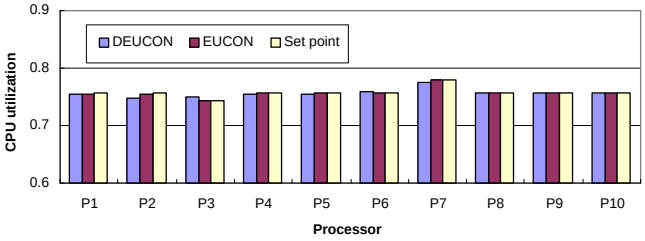


Figure 10. Average CPU utilization (etf=5)

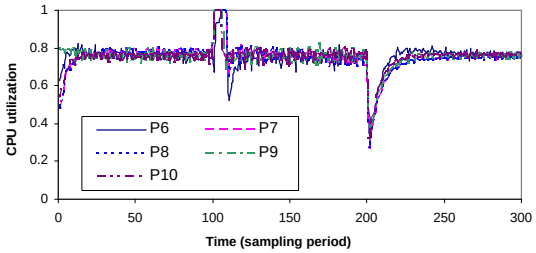
cessors, Figure 10 plots the average utilizations for all processors when *etf* is 5. The deviations of all utilizations are less than 0.008. We observe that on P_2 to P_7, the difference between the utilizations and the set points for DEUCON are slightly larger than that of EUCON. However, all the differences are within the ± 0.009 range. In practice, such small steady-state errors can be handled by setting the set points to slightly lower than the schedulable utilization bounds.

In summary, the simulation results demonstrate that DEUCON can achieve almost the same performance as EUCON, for a wide range of *etf* ($[0.5, 10]$ in our experiments).

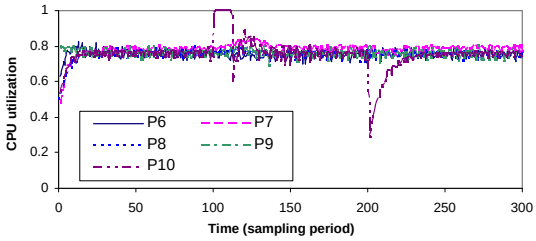
5.3.2 Varying Execution Times

In this experiment, execution times vary *dynamically* at runtime. To investigate the robustness of DEUCON we tested two scenarios of workload fluctuation. In the first set of runs, the average execution times on all processors change uniformly. In the second set of runs, only the execution times on P_{10} change dynamically, while those on the other processors remain unchanged. The first scenario represents *global* load fluctuation, while the second scenario represents *local* fluctuation on a part of the system.

Figure 11(a) shows a typical run with global workload fluctuation. The *etf* is initially 1.0. At time $100T_s$, it is decreased to 0.56, which corresponds to an 80% increase in the execution times of all subtasks such that all processors are suddenly overloaded. DEUCON responds to the overload by decreasing task rates which causes the utilizations on all processors to re-converge to their set points within $20T_s$. At time $200T_s$, the *etf* is increased to 1.67 corresponding to a 67% decrease in execution times. The utilizations on all processors drop sharply, causing DEUCON to dramatically increase task rates until the utilizations re-converge to their set points. The system maintains stability and avoids any significant oscillation throughout the run, despite the variations in execution times.



(a) Global fluctuation



(b) Local fluctuation on P_{10}

Figure 11. CPU utilization of P_6 to P_{10} when execution times fluctuate at run-time

In each run with local workload fluctuation, the *etf* on P_{10} follows the same variation as the global fluctuation, while all the other processors have a fixed *etf* of 1.0. As shown in Figure 11(b), the utilization of P_{10} converges to its set point after the significant variation of execution times at $120T_s$ and $250T_s$, respectively. We also observe that the other processors experience only slight utilization fluctuation after the execution times change on P_{10}. This result demonstrates that DEUCON effectively handles the coupling among processors during rate adaptation. The performance results of DEUCON in this experiment are very close to the results reported in EUCON [17].

6 Related Work

Traditional approaches for handling end-to-end tasks are based on open-loop approaches such as end-to-end scheduling [26] or distributed priority ceiling [21]. These approaches rely on schedulability analysis, which requires *a priori* knowledge about worst-case execution times. When task execution times are highly unpredictable, such open-loop approaches may severely under-utilize the system. An approach for dealing with unpredictable task execution times is resource reclaiming [4][23]. A drawback of existing resource reclaiming techniques is that they often require modifications to specific scheduling algorithms in operating systems. In contrast, the feedback control approach adopted in this paper can be easily implemented at the middleware layer on top of COTS platforms [16].

Control theoretic approaches have been applied to a number of computing and networking systems. A survey of feedback performance control in computing systems is presented in [1]. Recent research that applied control theory

to real-time scheduling and utilization control is directly related to this paper. For example, Steere et al. and Goel et al., developed feedback schedulers [9] [25] that guarantee desired progress rates for real-time applications. Abeni et al., presented control analysis of a reservation-based feedback scheduler [2]. Our earlier work has also resulted in a set of feedback control real-time scheduling algorithms [15]. These algorithms have been implemented as a middleware service (FCS/nORB) [16]. Other related work on feedback scheduling includes [7] [27]. All the aforementioned projects focused on controlling the performance of a *single* processor. Their control designs are based on single-input-single-output linear control techniques, which cannot be easily extended to end-to-end utilization control due to the coupling among multiple processors in DRE systems.

In addition to EUCON, DFCS is another feedback control real-time scheduling algorithm designed for distributed systems [24]. However, DFCS assumes, in contrast with the work presented in this paper, that tasks on different processors are independent from each other. The coupling among processors due to end-to-end tasks has not been modeled or addressed by DFCS.

7 Conclusions

DEUCON has been designed to control the utilization of distributed systems in unpredictable environments. Compared to centralized control algorithms, DEUCON features a peer-to-peer control stucture to handle the coupling among multiple processors and constraints. Our simulation results demonstrate that DEUCON can significantly improve the system scalability by distributing the controller computation and communication from one central processor to the whole system. Furthermore, DEUCON can tolerate considerable network delay which is crucial for many distributed systems. In addition, DEUCON achieves almost the same utilization control performance as the centralized control algorithm, even when task execution times deviate from the estimation or changes dynamically at run-time.

References

[1] T. Abdelzaher, J. Stankovic, C. Lu, R. Zhang, and Y. Lu. Feedback performance control in sofware services. *IEEE Control Systems*, 23(3), June 2003.

[2] L. Abeni, L. Palopoli, G. Lipari, and J. Walpole. Analysis of a reservation-based feedback scheduler. In *IEEE RTSS*, Dec. 2002.

[3] S. Brandt, G. Nutt, T. Berk, and J. Mankovich. A dynamic quality of service middleware agent for mediating application resource usage. In *IEEE RTSS*, Dec. 1998.

[4] M. Caccamo, G. Buttazzo, and L. Sha. Handling execution overruns in hard real-time control systems. *IEEE Trans. Comput.*, 51(7):835–849, 2002.

[5] E. Camponogara, D. Jia, B. Krogh, and S. Talukdar. Distributed model predictive control. *Control Systems Magazine*, 22(1):44–52, Feb. 2002.

[6] R. Carlson. Sandia SCADA program high-security SCADA LDRD final report. *SANDIA Report SAND2002-0729*, 2002.

[7] A. Cervin, J. Eker, B. Bernhardsson, and K.-E. Arzen. Feedback-feedforward scheduling of control tasks. *Real-Time Systems*, 23(1):25–53, July 2002.

[8] W. B. Dunbar and R. M. Murray. Distributed receding horizon control with applications to multi-vehicle formation stabilization. Technical Report CIT-CDS 04-003, Jan. 2004.

[9] A. Goel, J. Walpole, and M. Shor. Real-rate scheduling. In *IEEE RTAS*, 2004.

[10] D. Henriksson and T. Olsson. Maximizing the use of computational resources in multi-camera feedback control. In *IEEE RTAS*, May 2004.

[11] B. Kao and H. Garcia-Molina. Deadline assignment in a distributed soft real-time system. *IEEE Transactions on Parallel and Distributed Systems*, 8(12):1268–1274, 1997.

[12] J. P. Lehoczky. Fixed priority scheduling of periodic task sets with arbitrary deadline. In *IEEE RTSS*, 1990.

[13] C. Liu and J. Layland. Scheduling Algorithms for Multiprogramming in a Hard-Real-Time Environment. *Journal of ACM, Vol. 20, No.1, pp. 46-61*, Jan. 1973.

[14] J. W. S. Liu. *Real-Time Systems*. Prentice Hall, 2000.

[15] C. Lu, J. Stankovic, G. Tao, and S. Son. Feedback Control Real-Time Scheduling: Framework, Modeling, and Algorithms. *Real-Time Systems*, 23(1/2):85–126, July 2002.

[16] C. Lu, X. Wang, and C. Gill. Feedback Control Real-Time Scheduling in ORB Middleware. In *IEEE RTAS03*, Washington, DC, May 2003.

[17] C. Lu, X. Wang, and X. Koutsoukos. End-to-end utilization control in distributed real-time systems. In *ICDCS 2004*, Tokyo, Japan, Mar. 2004.

[18] J. Maciejowski. *Predictive Control with Constraints*. Prentice Hall, 2002.

[19] P. Marti, G. Fohler, P. Fuertes, and K. Ramamritham. Improving quality-of-control using flexible timing constraints: metric and scheduling. In *IEEE RTSS*, 2002.

[20] M. D. Natale and J. Stankovic. Dynamic end-to-end guarantees in distributed real-time systems. In *IEEE RTSS*, 1994.

[21] R. Rajkumar, L. Sha, and J. P. Lehoczky. Real-Time Synchronization Protocols for Multiprocessors. In *IEEE RTAS*, Dec. 1988.

[22] D. Seto, J. P. Lehoczky, L. Sha, and K. G. Shin. On task schedulability in real-time control system. In *IEEE RTSS*, Dec. 1996.

[23] C. Shen, K. Ramamritham, and J. A. Stankovic. Resource reclaiming in multiprocessor real-time systems. *IEEE Trans. Parallel Distrib. Syst.*, 4(4):382–397, 1993.

[24] J. A. Stankovic, T. He, T. Abdelzaher, M. Marley, G. Tao, S. Son, and C. Lu. Feedback control scheduling in distributed real-time systems. In *IEEE RTSS*, 2001.

[25] D. Steere and et al. A feedback-driven proportion allocator for real-rate scheduling. In *Operating Systems Design and Implementation*, pages 145–158, 1999.

[26] J. Sun and J. W.-S. Liu. Synchronization protocols in distributed real-time systems. In *ICDCS*, 1996.

[27] Y. Zhu and F. Mueller. Feedback edf scheduling exploiting dynamic voltage scaling. In *IEEE RTAS*, 2004.