Convex optimization framework for intermediate deadline assignment in soft and hard real-time distributed systems

Jinkyu Lee, Insik Shin, Arvind Easwaran

ABSTRACT

It is generally challenging to determine end-to-end delays of applications for maximizing the aggregate system utility subject to timing constraints. Many practical approaches suggest the use of intermedi-ate deadline of tasks in order to control and upper-bound their end-to-end delays. This paper proposes a unified framework for different time-sensitive, global optimization problems, and solves them in a distributed manner using Lagrangian duality. The framework uses global viewpoints to assign interme- diate deadlines, taking resource contention among tasks into consideration. For soft real-time tasks, the proposed framework effectively addresses the deadline assignment problem while maximizing the aggre- gate quality of service. For hard real-time tasks, we show that existing heuristic solutions to the deadline assignment problem can be incorporated into the proposed framework, enriching their mathematical interpretation.

Keywords: Intermediate deadline assignment, Convex optimization, Soft real-time distributed systems, Hard real-time distributed systems

1. Introduction

A distributed task is usually comprised of several subtasks, each one having processing demands at some node in order, and often requires end-to-end guarantees on quality-of-service (QoS). For example, a real-time task is subject to timing constraints typically specified by deadlines (hard real-time systems) or time-sensitive utility functions (soft real-time systems) (Wu et al., 2005). The optimization goal of such a distributed system is often defined as maximizing schedulability (*i.e.*, maximizing the number of schedu- lable tasks) in hard real-time systems, or maximizing the collective utilities of individual tasks in soft real-time systems. Achieving such delay-sensitive optimization goals is generally difficult, because it is challenging to precisely calculate maximum end-to-end delays in the presence of resource contention among tasks.

In a distributed system, tasks potentially compete with each other for computing resources whenever they go through the same node. Prioritized scheduling is an effective mechanism to provide bounded local delays to individual subtasks within a node, while they are subject to interference from other subtasks. Nonetheless, prioritized scheduling does not significantly alleviate the difficulty of computing the maximum end-to-end delay of a task. In many practical cases, it is computationally intractable to calculate the maximum local delay of a subtask (Baruah et al., 1990; Leung and Whitehead, 1982; Palencia and Harbour, 2005).

As a practical solution, many previous studies have commonly adopted an approach to approximate the maximum end-to-end delay rather than to compute it exactly (Garcia and Harbour, 1995; Saksena and Hong, 1996; Kao and Garcia-Molina, 1993, 1994; Bettati and Liu, 1992; Natale and Stankovic, 1994; Jonsson and Shin, 1997). They introduced a local deadline for each subtask in a node, and used this deadline to upper-bound the local delay of the sub- task. Then, the end-to-end delay of a task can be upper-bounded by the sum of local deadlines of all its subtasks. The problem of finding the task end-to-end delays for some optimization objectives, is then reduced to the problem of finding local subtask deadlines for the given objective. Many existing studies proposed methods to assign local subtask deadlines in order to maximize schedulability (Garcia and Harbour, 1995; Saksena and Hong, 1996; Kao and Garcia- Molina, 1993, 1994; Bettati and Liu, 1992; Natale and Stankovic, 1994; Jonsson and Shin, 1997). We believe that a perspective of considering the resource contention from other subtasks is cru- cial to determining local deadlines of subtasks. However, all these previous studies commonly lack such a perspective.

The difficulty of finding a "right" local subtask deadline for some optimization objective lies on its seemingly contradicting effects. If the local deadline of a subtask becomes larger, then it imposes more stringent timing constraints on the other subtasks that belong to the same task, given that we naturally want to

minimize its end-to-end delay. On the other hand, if the local deadline of a subtask becomes smaller, it can potentially interfere with more subtasks within the same node that belong to other tasks, threatening node schedulability. In determining a local subtask deadline, therefore, it is necessary to investigate its effect on both the other subtasks of the same task (across nodes) and the other subtasks within the same node (across tasks). It entails a global approach to the deadline assignment problem of individual tasks in the presence of optimization objectives.

We propose a framework to address different QoS-related con- vex optimization problems for both soft and hard real-time tasks. Our framework allows global viewpoints to be incorporated into the formulation of deadline assignment problems, taking resource contention among tasks into consideration. It derives local solu- tions through Lagrange duality, wherein nodes can collectively converge to a global optimum through distributed computation. There has been little work on maximizing the aggregate QoS of end-to-end real-time tasks. Our framework can be effectively used to find local deadlines that maximize the collective QoS, when each soft real-time task has a utility function that characterizes its QoS. In order to maximize schedulability of deadline-based hard real-time tasks, many previous studies proposed heuristics to the deadline assignment problem. We show that our framework can characterize these heuristics using closed-form utility functions, thereby providing a precise mathematical interpretation of the heuristics. We note that the solutions proposed by those heuris- tics do not necessarily meet end-to-end deadlines. By incorporating these heuristics in our convex optimization framework, we provide guarantees on end-to-end deadlines.

Our paper is organized as follows. Section 2 presents related work, and Section 3 describes the system model. Section 4 provides our convex optimization framework for soft real-time systems, and Section 5 extends the framework toward hard real-time systems, accommodating existing heuristics with various utility functions. Section 6 discusses how to find solutions in our framework in a distributed manner. Section 7 presents performance evaluation of our framework. Finally, Section 8 concludes this paper with future work.

2. Related work

Many studies have focused on the local subtask deadline assign- ment problem, with a view to controlling end-to-end delays (Garcia and Harbour, 1995; Saksena and Hong, 1996; Kao and Garcia- Molina, 1993, 1994; Bettati and Liu, 1992; Natale and Stankovic, 1994; Jonsson and Shin, 1997). These studies focus on how to divide the end-to-end deadline of a task into local deadlines for its subtasks, but they do not consider the resource contention in intermediate nodes between subtasks of different tasks. There are some studies on end-to-end delay analysis of distributed real- time systems (Jayachandran and Abdelzaher, 2008, 2009). These studies focus on reducing the pessimism in calculating end-to-end delays for pipelined streams of computations, but their approach cannot be used to guarantee schedulability and at the same time achieve certain optimization goals such as maximizing QoS. A study (Stavrinides and Karatza, 2010) examines some algorithms and their alternative versions for guarantees on end-to-end deadline in distributed real-time systems, but their goal is to utilize imprecise computations (Lin et al., 1987), which is different from our system model.

Convex optimization theory has been a popular tool to solve many global optimization problems for several decades. Tech- niques that find optimal solutions either in a centralized manner, or using distributed computations (Lagrangian duality), have been developed (Bertsekas and Tsitsiklis, 1997; Low, 1999). Many of these techniques have been applied to solve the problem of guar- anteeing end-to-end delays in a distributed real-time system (Chen et al., 2007; Bini and Cervin, 2008; Zhu et al., 2009; Lumezanu et al., 2008). Some of these techniques use centralized solutions to the optimization problem (Chen et al., 2007; Bini and Cervin, 2008; Zhu et al., 2009). Distributed solutions to this optimization problem have been recently considered (Lumezanu et al., 2008). This study assumes proportional share scheduling within nodes. Since such a scheduling framework is not implementable, it must be approxi- mated. However, any approximation of the scheduling framework will invalidate the proposed analysis. Distributed optimization has been applied to achieving QoS maximization for soft real-time tasks through dynamic route and rate assignments in distributed real-time systems (Shu et al., 2008), and through bandwidth allo- cation in wireless networks (Jayachandran and Abdelzaher, 2008). Our previous study (Lee et al., 2010) also provides a distributed optimization framework for QoS maximization in the presence of failure. However, this paper is differentiated from those studies in two aspects: our framework (i) aims at intermediate deadline assignment and (ii) accommodates both soft and hard real-time distributed systems.

3. System model

In this paper, we consider a distributed real-time system with V_N nodes and V_T tasks. The nodes are numbered 1, . . ., V_n such that each node has a unique number, and we express a node number n as N_n. Each task $r_i \in \boldsymbol{r}$ is comprised of m_i subtasks such that each subtask executes on exactly one node. The kth subtask executing on N_n is denoted as $J_{(i,k,n)}$; whenever n is irrelevant we omit the third parameter completely. Adjacent subtasks $J_{(i,k)}$ and $J_{(i,k+1)}$ execute in sequence in a pipelined fashion; $J_{(i,k+1)}$ becomes ready for execution when $J_{(i,k)}$ completes. We let $C_{(i,k)}$ denote the worst-case (maximum) execution time of subtask $J_{(i,k)}$. Each task r_i is a sporadic task such that its first subtask $J_{(i,1)}$ is released repeatedly with a minimum gap of T_i time units.

Let $d_{(i,k)}$ denote the maximum local response time that subtask $J_{(i,k)}$ experiences in its node; it is a time duration from an instant at which it is released in the node to another instant at which it finishes its execution. Then, we denote the end-to-end response time of a task r_i by d_i, where $d_i = \sum_{k=1}^{m_i} d_{(i,k)}$.

In this paper, we assume each node consists of a uniprocessor platform, and is scheduled by Earliest Deadline First (EDF). How- ever, the technique described in this paper can be easily extended to other platforms and scheduling algorithms as long as the utilization bound (explained in Section 4.2) is provided.

4. Convex optimization framework for soft real-time systems

In this section, we develop a convex optimization framework for soft real-time systems. To do this, we first explain the characteris- tics of soft real-time systems, and then derive a node schedulability condition. Using the condition, we formulate a convex optimization problem to determine local deadlines for soft real-time systems.

4.1. Soft real-time systems and their goal

Different from hard real-time systems, soft real-time systems allows a task to miss its deadline, and there are many notions of soft real-time supports, such as QoS depending on delay (Wu et al., 2005), satisfying a given tardiness (Devi and Anderson, 2004), prob- abilistic guarantees on timing requirements (Tia et al., 1995; Atlas and Bestavros, 1998), etc. Among various notions of soft real-time supports, in this paper, we focus on maximizing QoS depending

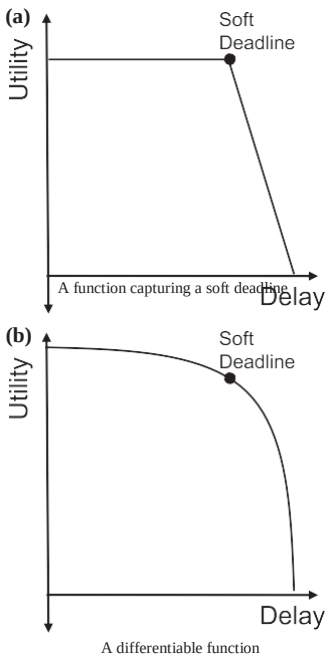


Fig. 1. Utility functions.

on delay. That is, we assume that each task has its own utility function U_i, which is a function of its end-to-end delay d_i. Util- ity functions can be viewed as characterizing different QoS levels. We consider concave and non-increasing utility functions to cap- ture that a greater QoS comes with a shorter end-to-end delay and degradation of QoS gets more severe as delay gets longer. Then, such a QoS based soft real-time support can also capture a situation where degradation of QoS is smooth before a certain point (*i.e.*, a soft deadline), but becomes rapid after this point, as shown in Fig. 1. Examples of tasks subject to such soft deadlines (*i.e.*, Fig. 1(a)) include plot correlation and track maintenance of a coastal air defense system (see Fig. 2(b) in Wu et al. (2005)). In this paper, we consider utility functions to be differentiable in order to incorporate them into the proposed optimization framework. If an original function is not differentiable as shown in Fig. 1(a), it can be approximated as the one shown in Fig. 1(b). We then define the system utility as

$$U_{sys} = \sum_{\tau_i \in \Gamma} U_i(d_i). \tag{1}$$

Given a set of QoS-sensitive tasks as above, informally, our aim is to provide QoS guarantees as much as possible. We capture it by maximizing the system utility function U_{sys}. Then our goal is to bound the delay d_i of each task $r_i \in \boldsymbol{r}$ that maximizes U_{sys}. However, computing this bound exactly in general distributed sys- tems is computationally intractable (Baruah et al., 1990; Palencia and Harbour, 2005). Hence we approximate the bound on d_i as follows. For each subtask $J_{(i,k)}$, we define a local (artificial) deadline $D_{(i,k)}$ such that $D_{(i,k)} \leq T_i$. We derive conditions which guarantee that every occurrence of each subtask finishes by its local deadline, *i.e.*, we enforce the condition $d_{(i,k)} \leq D_{(i,k)}$ for each subtask $J_{(i,k)}$. Then we can upper-bound d_i as

$$d_i \overset{def}{=} \sum_{k=1}^{m_i} d_{(i,k)} \leq \sum_{k=1}^{m_i} D_{(i,k)}. \tag{2}$$

Thus the problem of bounding d_i for each task r_i that maximizes U_{sys}, is transformed to the problem of finding $D_{(i,k)}$ for each subtask $J_{(i,k)}$ that maximizes U_{sys}. Note that it is essential to decompose the end-to-end delay into delays (and therefore artificial deadlines) for individual subtasks, because of the limitations of existing real-time scheduling theory.[2] These local deadlines enable us to optimize the global system utility while still maintaining the schedulability of individual nodes.

4.2. Node schedulability condition

We now consider a node N_n with subtasks $\{J_{(i,k,x)} | x = n\}$, and derive schedulability conditions under the preemptive EDF sched- uler, *i.e.*, conditions which guarantee $d_{(i,k)} \leq D_{(i,k)}$. We first define a density of subtask $J_{(i,k)}$ as follows:

$$\mathrm{den}_{(i,k)} = \frac{C_{(i,k)}}{D_{(i,k)}}. \tag{3}$$

Then, we guarantee that all subtasks executed in N_n finish their execution within their local deadlines, if the sum of density values of the subtasks is no larger than a given utilization bound as follows (Liu and Layland, 1973):

$$\sum_{J_{(i,k,x)}:x=n} \mathrm{den}_{(i,k)} \leq \mathrm{UB}_n, \tag{4}$$

where UB_n represents the utilization bound of the scheduling algo- rithm used by node N_n, and $\mathrm{UB}_n = 1$ when the preemptive EDF scheduler is deployed on a uniprocessor platform (Liu and Layland, 1973). Since a local deadline $D_{(i,k)}$ is no larger than its period, theabove schedulability condition holds, but it is only sufficient andnot necessary.

In order to formulate the *convex* optimization problem to be developed in Section 4.3, all the constraints used in the optimiza- tion problem must be concave functions of the variables. In our framework, local subtask deadlines $D_{(i,k)}$ are the variables, and the schedulability conditions for each node (Eq. (4)) are used as con- straints. We test the concavity of the schedulability conditions, and the following statement is true for all $D_{(i_0,k_0)} > 0$:

$$\frac{\partial^2}{\partial D_{(i_0,k_0)}^2}\left[\mathrm{UB}_n - \sum_{J_{(i,k,x)}:x=n} \frac{C_{(i,k)}}{D_{(i,k)}}\right] \leq 0. \tag{5}$$

Hence each constraint is a concave function of deadline vari- ables, and it can be used in our convex optimization framework.

4.3. Primal problem

The deadline assignment problem aims to determine the local deadline ($D_{(i,k)}$) of every subtask ($J_{(i,k)}$) in order to provide a guar- anteed maximum system utility. Thus the optimization problem (primal problem) can be formulated as

(Primal problem)

$$\mathrm{Max}: U_{sys} = \sum_{\tau_i \in \Gamma} U_i(D_{(i,1)}, \ldots, D_{(i,m_i)}), \tag{6}$$

[2] Current real-time scheduling theories are mostly developed for node-level schedulability analysis, and the system-level analysis is achieved by assembling individual node-level analysis results. Therefore, it is hard to directly support a task model in which (i) a series of subtasks sequentially go through multiple nodes with one end-to-end deadline, and (ii) the sets of nodes which subtasks of different tasks use are different.

$$\text{Sub.to}: \sum_{\mathcal{J}_{(i,k,x)}:x=n} \frac{C_{(i,k)}}{D_{(i,k)}} \le \text{UB}_n, \forall n \in 1, \ldots, V_N. \quad (7)$$

By Eq. (5) and our assumption of concavity of $U_i(\cdot)$, the above primal problem is a convex optimization problem.

5. Convex optimization framework for hard real-time systems

In this section, we extend the convex optimization framework developed in the previous section toward hard real-time systems. We first explain the characteristics of hard real-time systems, and then formulate the convex optimization framework for such sys- tems. While hard real-time systems do not have any utility function by nature, we present how to decide utility functions to accommo- date existing heuristic approaches.

5.1. Hard real-time systems, their goal, and primal problem

For hard real-time systems, task r_i has its relative end-to-end deadline D_i such that the execution of task r_i should be finished within D_i time units after its release as follows:

$$d_i \overset{def}{=} \sum_{k=1}^{m_i} d_{(i,k)} \le \sum_{k=1}^{m_i} D_{(i,k)} \le D_i, \forall \tau_i \in \Gamma. \quad (8)$$

Here we assume that it holds $D_i \le T_i$ for each task $r_i \in r$.

Hard real-time tasks typically require schedulability guarantees, i.e., satisfaction of end-to-end task deadline. As long as task delays are shorter than or equal to the respective deadlines, there is no incentive to reduce those delays any further. This means there is no utility function in hard real-time systems by nature, and instead we are interested in how to distribute a deadline of a task to each subtask. Therefore, we design utility functions for hard real-time systems in order to implement certain policies of distributing dead- lines. We will present how to design utility functions for existing deadline assignment policies (Jonsson and Shin, 1997; Kao and Garcia-Molina, 1993) in Section 5.2, and now we present the primal problem for hard real-time systems for given utility functions.

The deadline assignment problem for determining local dead- lines of subtasks for hard real-time systems can be formulated by adding end-to-end deadline constraints (Eq. (11)) to the problem in Section 4.3 as follows:

(Primal problem)

$$\text{Max}: U_{sys} = \sum_{\tau_i \in \Gamma} U_i(D_{(i,1)}, \ldots, D_{(i,m_i)}), \quad (9)$$

$$\text{Sub.to}: \sum_{\mathcal{J}_{(i,k,x)}:x=n} \frac{C_{(i,k)}}{D_{(i,k)}} \le \text{UB}_n, \forall n \in 1, \ldots, V_N, \quad (10)$$

$$\sum_{k=1}^{m_i} D_{(i,k)} \le D_i, \forall \tau_i \in \Gamma. \quad (11)$$

Similar to Eq. (5), the following inequality regarding Eq. (11) is true for all $D_{(i_0,k_0)} > 0$:

$$\frac{\partial^2}{\partial D_{(i_0,k_0)}^2}\left[D_i - \sum_{k=1}^{m_i} D_{(i,k)}\right] \le 0. \quad (12)$$

Therefore by Eqs. (5) and (12), the primal problem for hard real-time systems is a convex optimization problem as long as $U_i(\cdot)$ for all $r_i \in r$ is concave.

5.2. Utility function decision

Many previous studies proposed heuristic principles to the deadline assignment problem, and such heuristics can be incorpo- rated in our framework using carefully designed utility functions. We now discuss two such existing heuristics (Jonsson and Shin, 1997; Kao and Garcia-Molina, 1993). Let *laxity* of a task denote the difference between its end-to-end deadline and the sum of its subtask execution times; laxity of task r_i is $D_i - \sum_{j:1 \le j \le m_i} C_{(i,j)}$. Heuristics in (Jonsson and Shin, 1997; Kao and Garcia-Molina, 1993) distribute this laxity between subtasks to solve the deadline assign- ment problem.

Under the *pure laxity ratio* approach (Jonsson and Shin, 1997; Kao and Garcia-Molina, 1993), a laxity of each task is assigned to subtasks as follows:

$$D_{(i,k)} = C_{(i,k)} + \frac{D_i - \sum_{j=1}^{m_i} C_{(i,j)}}{m_i}. \quad (13)$$

The principle of this approach, which is uniform laxity dis- tribution, ignores node level schedulability, and hence it can be equivalently expressed by our primal problem using the follow- ing utility function for task r_i with constraint (11) (end-to-end deadline) and without constraint (10) (node-level schedulability):

$$U_i(D_{(i,1)}, \ldots, D_{(i,m_i)}) \overset{def}{=} \sum_{k=1}^{m_i} \log(D_{(i,k)} - C_{(i,k)} + \epsilon), \quad (14)$$

where E is a small value that prevents an undesirable situation ($\lim_{x \to 0} \log(x) = -\infty$). Since $U_i(\cdot)$ is independent of $U_j(\cdot)$ ($i \ne j$), $U_i(\cdot) = \log(\prod_{k=1}^{m_i}(D_{(i,k)} - C_{(i,k)} + E)) = \sum_{k=1}^{m_i} \log(D_{(i,k)} - C_{(i,k)} + E)$ is maxi- mized when

$$D_{(i,1)} - C_{(i,1)} = \cdots = D_{(i,m_i)} - C_{(i,m_i)} = \frac{D_i - \sum_{j=1}^{m_i} C_{(i,j)}}{m_i}, \quad (15)$$

where $E \to 0$. Therefore, our framework using the utility functionof Eq. (14) with constraint (11) and without constraint (10) gives the same results as the pure laxity ratio approach. Note that this is a convex optimization problem since the utility function (14) is concave.

Under the *normalized laxity ratio* approach, existing heuristics assign laxity in proportion to subtask execution time as follows (Jonsson and Shin, 1997; Kao and Garcia-Molina, 1993):

$$D_{(i,k)} = C_{(i,k)}\left(1 + \frac{D_i - \sum_{j=1}^{m_i} C_{(i,j)}}{\sum_{j=1}^{m_i} C_{(i,j)}}\right). \quad (16)$$

The normalized laxity ratio distributes the laxity of each task proportional to the execution time of its subtasks, and we also express this principle by our primal problem using the following utility function for task r_i with constraint (11) and without con- straint (10):

$$U_i(D_{(i,1)}, \ldots, D_{(i,m_i)}) \stackrel{def}{=} \sum_{k=1}^{m_i} \log\left(D_{(i,k)} - C_{(i,k)}\left(1 + \frac{D_i - \sum_{j=1}^{m_i} C_{(i,j)}}{\sum_{j=1}^{m_i} C_{(i,j)}}\right) + \epsilon\right), \quad (17)$$

where E is a small value that prevents an undesirable situation ($\lim_{x\to 0} \log(x) = -\infty$). Similar to the pure laxity ratio approach, the normalized laxity ratio approach also does not consider node schedulability. Hence, we can incorporate this approach in our framework by maximizing the utility function of Eq. (17), subject to only the end-to-end deadline constraints given by Eq. (11). With a similar reasoning to that of the pure laxity ratio approach, we can show that the utility functions of Eq. (17) enforce the principle specified in Eq. (16).

We note that both the pure and normalized laxity ratio approach themselves do not consider the schedulability of subtasks within a node. This means we cannot guarantee that the local delay $d_{(i,k)}$ is less than or equal to the local deadline $D_{(i,k)}$, resulting in no guarantee on end-to-end delays. To guarantee such end-to-end delays for hard real-time systems, our framework can easily extend these approaches with node schedulability by incorporating the constraint of Eq. (10) in the optimization problem. Then, we can find the local subtask deadlines according to the principle of heuristics, subject to node schedulability. We will explain this feature in detail in Section 7.

6. Distributed solution framework

While our framework in Sections 4 and 5 support soft and hard real-time distributed systems, respectively, the framework requires a coordinator node. That is, whenever tasks are in and out or task specifications are modified, the coordinator node should receive the information about such changes, re-compute all intermediate deadlines of each subtask, and distribute the deadlines. Therefore, such a centralized approach may require a lot of message exchanges and high computing power of the coordinator node, and therefore it may not be suitable for some environments where explicit message exchanges are neither possible nor inexpensive or no node has enough computing capability. This entails the need of a distributed framework. In this section, we propose a distributed solution framework, which corresponds to the primal problems in Sections 4.3 and 5.1. Then, we discuss implementation issues.

6.1. Dual problem and distributed computation

Any optimization problem can be re-written in its dual form using Lagrange multipliers (see Chapter 5 in Boyd and Vandenberghe (2004)). This formulation is called the *Lagrange dual problem*. For the optimization problem presented in the previous sections, its Lagrange dual problem can be defined as follows:

(Dual problem)

$$\min_{\mathbf{p}\geq 0, \mathbf{q}\geq 0} \max_{\mathbf{D}\geq 0} L(\mathbf{D}, \mathbf{p}, \mathbf{q}) = \sum_{\tau_i \in \Gamma} U_i\left(D_{(i,1)}, \ldots, D_{(i,m_i)}\right) + \sum_{n=1}^{V_N} p_n \cdot \left(UB_n - \sum_{J_{(i,k,n)}: x=n} \frac{C_{(i,k)}}{D_{(i,k)}}\right) + \sum_{\tau_i \in \Gamma} q_i \cdot \left(D_i - \sum_{k=1}^{m_i} D_{(i,k)}\right), \quad (18)$$

In this formulation, *node price* p_n is the Lagrange multiplier for the schedulability constraint of node N_n. Likewise, *task price* q_i is the Lagrange multiplier for the end-to-end deadline constraint of task r_i. Suppose each utility function U_i is concave. Then, forcing node prices p_n and task prices q_i to be non-negative guarantees strong duality (see Chapter 5 in Boyd and Vandenberghe (2004)). This means that (i) there is no duality gap between the primal and Lagrange dual problems (i.e., optimal dual solution is equivalent to the optimal primal solution), and (ii) dual optimal node and taskprices exist. In this case, we can solve the dual problem using the gradient projection algorithm (Bertsekas and Tsitsiklis, 1997; Low, 1999). That is, we can find the optimal solution in an iterative manner (*i.e.*, $A(t+1) = f(A(t))$, where $A(t)$ means the value of A at the tth iteration). Node prices p_n and task prices q_i can be iterated as follows:

$$p_n(t+1) = \left[p_n(t) - \gamma_n\left(UB_n - \sum_{J_{(i,k,n)}: x=n} \frac{C_{(i,k)}}{D_{(i,k)}}\right)\right]^+, \quad (19)$$

$$q_i(t+1) = \left[q_i(t) - \delta_i\left(D_i - \sum_{k=1}^{m_i} D_{(i,k)}\right)\right]^+, \quad (20)$$

where $[x]^+$ means $\max(0, x)$.

The constants γ_n and ι_i are step sizes and determine the rate of convergence of the iteration. These constants guarantee convergence of the iteration whenever they satisfy Lipschitz continuity (Bertsekas and Tsitsiklis, 1997). We can then obtain deadline $D_{(i,k)}(t+1)$ of subtask $J_{(i,k,n)}$, by solving the differential equation given below in which $D_{(i,x)} = D_{(i,x)}(t)$ for all x:

$$\frac{\partial U_i(D_{(i,1)}, \ldots, D_{(i,k-1)}, D_{(i,k)}(t+1), D_{(i,k+1)}, \ldots, D_{(i,m_i)})}{\partial D_{(i,k)}(t+1)} + p_n(t)\frac{C_{(i,k)}}{\left[D_{(i,k)}(t+1)\right]^2} - q_i(t) = 0. \quad (21)$$

Note that the only variable in this equation is $D_{(i,k)}(t+1)$, and therefore it can be computed.

The optimal solution to the dual problem can be obtained through distributed computation. Each node N_n needs to compute its node price p_n in Eq. (19), and this requires knowledge of all the subtask deadlines in N_n. Each task r_i needs to compute its task price q_i in Eq. (20), and this requires knowledge of deadlines of r_i's subtasks. Thus, to compute q_i, information needs to be exchanged between nodes that execute r_i's subtasks. The computation of subtask deadline $D_{(i,k)}$ in Eq. (21) always requires knowledge of p_n and q_i, and may also sometimes require knowledge of deadlines of r_i's other subtasks. This means that solving Eqs. (20) and (21) will in general require cross-node communication. This information exchange can be effectively implemented with little extra communication cost. For example, many approaches to the network utility maximization problem employ efficient mechanisms to exchange implicit information (*e.g.*, congestion price marked in packets, loss rate, or some piggybacked values) with no extra packet delivery (Athuraliya et al., 2001; Athuraliya and Low, 2000).

Although in general solving the dual problem requires communication between nodes, we characterize a domain in which no information needs to be exchanged. Consider a soft real-

time system such that the utility function for each task can be decomposed into functions of its subtask deadlines, *i.e.*, $U_i(D_{(i,1)}, \ldots, D_{(i,m_i)}) = \sum_{1 \le k \le m_i} U_{(i,k)}(D_{(i,k)})$. Since it is a soft real-time system, as shown in Section 4.5, the end-to-end task deadline constraint $\sum_{1 \le k \le m_i} D_{(i,k)} \le D_i$ can be ignored. Therefore, in this case the subtask deadlines are independent of each other, because (i) task

where $\mathbf{D} = \{D_{(i,k)}\}, \forall (i,k) \in \{(s,w) | r_s \in r, w = 1, \ldots, m_s\}$, $\mathbf{p} = \{p_n\}$, $\forall n = 1, \ldots, V_N$ and $\mathbf{q} = \{q_t\}, \forall t = 1, \ldots, V_T$.

Note that for soft real-time systems, the last term in Eq. (18) and related values (e.g., $\mathbf{q}$) are removed.

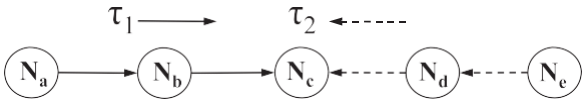


Fig. 2. Topology of a toy example.

price q_i is not used, and (ii) $\frac{\partial}{\partial D_{(i,k)}} U_i$ is independent of all subtask deadlines except $D_{(i,k)}$.

6.2. Implementation issues

For many practical environments, exchange of control messages can also fail, and one may wonder the effect of such failures on our distributed computations. Fortunately, our optimization frame-works can converge to an optimal solution, even in the presence of such control message losses. For example, when a control message is lost at some iteration step, the frameworks can use the con- trol message from the previous step. This asynchronous iteration reduces the rate of convergence, but still guarantees convergence (Bertsekas and Tsitsiklis, 1997). A key idea of the proof (Bertsekas and Tsitsiklis, 1997) is to set a worst-case period by which the con- trol messages become outdated, and the rest of the proof is similar to the case of synchronous iterations.

Another implementation issue is how to determine when the iterative computation of Eq. (21) converges. We define our convergence criteria if the following condition holds for all $D_{(i,k)}$:

$$|D_{(i,k)}(t+1) - D_{(i,k)}(t)| < E_D, \tag{22}$$

where E_D is a sufficiently small positive real number; this generates a trade-off between accuracy and rate of convergence. Many gradi- ent algorithms employ this kind of convergence criteria (Bertsekas and Tsitsiklis, 1997).

7. Performance evaluation

In this section, we evaluate the performance of our framework. To do this, we first present an intuitive example, which shows how our utility function decisions presented in Section 5.2 accom-modate and improve the existing heuristics (Jonsson and Shin, 1997; Kao and Garcia-Molina, 1993). Then, we present quantitative results through simulations, and discuss them.

7.1. A toy example

We devise a simple topology that consists of two tasks and five nodes, as shown in Fig. 2. Only the node N_c has multiple subtasks, and other node has one subtask. For the two tasks, Table 1 lists the execution times of their subtasks and their end-to-end deadlines.

For the same topology in Fig. 2, we simulate three approaches: PLR, PO, and POS. Here PLR stands for the Pure Laxity Ratio approach, which equally divides the laxity as shown in Eq. (13). Here PO represents our framework in Section 5.1 using the utility functions of Eq. (14) and the constraints of Eq. (11), but without the node schedulability constraints (Eq. (10)). POS also represents our framework using the utility functions of Eq. (14) and the constraints of Eq. (11) along with the schedulability constraints of Eq. (10). Table 2 lists the deadline assignment for the three approaches.

Table 1
Subtask execution times and task end-to-end deadline for the example shown in Fig. 2.

	N_a	N_b	N_c	N_d	N_e	Deadline (D_i)
r_1	1.0	2.0	2.0	–	–	17.0
r_2	–	–	1.0	2.0	2.0	6.0

Table 2
Local deadlines and density for the pure laxity ratio approach.

	N_a	N_b	N_c	N_d	N_e
PLR					
r_1	5.000	6.000	6.000	–	–
r_2	–	–	1.333	2.333	2.333
Density	0.200	0.333	**1.083**	0.857	0.857
PO					
r_1	5.000	6.000	6.000	–	–
r_2	–	–	1.333	2.333	2.333
Density	0.200	0.333	**1.083**	0.857	0.857
POS					
r_1	4.551	5.551	6.898	–	–
r_2	–	–	1.408	2.296	2.296
Density	0.219	0.360	**1.000**	0.871	0.871

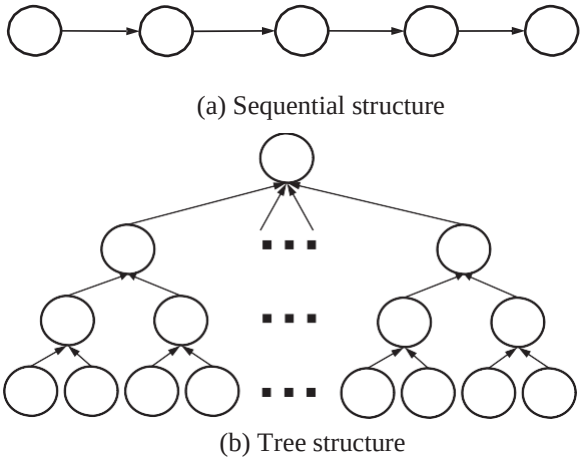
As expected, the deadline distribution of PO is exactly the same as that of PLR. However, in PLR and PO, the density $(\sum_j C_{(i,j)}/D_{(i,j)})$ of node N_c exceeds 1.0, which means subtasks in N_c may not be schedulable. If we add the schedulability constraints of Eq. (10) to our formulation, then the resulting deadlines guarantee the schedulability of node N_c, as shown in Table 2 under POS. We note that the increase to local deadlines in N_c comes from the laxity of other subtasks, in particular, taking the laxity equally out of those subtasks. For instance, in POS, the local deadline $D_{(1,3,c)}$ is 6.898, which is an increase from 6.000 in PLR and PO. Such an increase by 0.898 comes equally from the decrease of $D_{(1,1,a)}$ and $D_{(1,2,b)}$ by 0.449, respectively. Thus, we can see that our approach (POS) is able to find a schedulable solution while being able to follow the principle of the pure laxity ratio as much as possible.

For the same example shown in Fig. 2, we simulate another three approaches: NLR, NO, and NOS. NLR (the Normalized Laxity Ratio approach) corresponds to PLR, but it divides the laxity proportion-ally to the subtask's execution time as shown in Eq. (16). Here NO and NOS correspond to PO and POS, but they use the utility func- tion of Eq. (17). Table 3 compares the deadline assignment of the three approaches. It shows that NO produces the same result as that of NLR. Similar to POS, NOS guarantees the node's schedulability by increasing the deadlines of subtasks in N_c and by decreasing thosein N_a, N_b, N_d and N_e.

In summary, our approach not only accommodates the previ- ous heuristics precisely, but it also improves them by guaranteeing the schedulability of subtasks within nodes. Since node schedu- lability ensures that each subtask can meet its local deadline, we can guarantee that the proposed solutions will meet end-to-end deadlines.

Table 3
Local deadlines and density for the normalized laxity ratio approach.

	N_a	N_b	N_c	N_d	N_e
NLR					
r_1	3.400	6.800	6.800	–	–
r_2	–	–	1.200	2.400	2.400
Density	0.294	0.294	**1.127**	0.833	0.833
NO					
r_1	3.400	6.800	6.800	–	–
r_2	–	–	1.200	2.400	2.400
Density	0.294	0.294	**1.127**	0.833	0.833
NOS					
r_1	3.391	6.791	6.817	–	–
r_2	–	–	1.415	2.292	2.292
Density	0.294	0.294	**1.000**	0.872	0.872



(a) Sequential structure

(b) Tree structure

Fig. 3. Topology of simulations.

7.2. Simulation results and discussion

This subsection aims at showing how our framework in Section 5.2 improves the existing heuristics (Jonsson and Shin, 1997; Kao and Garcia-Molina, 1993) in different environments. To do this, we choose two general topologies with various synthetic task sets. The first topology is a sequential structure in Fig. 3(a), and some control systems have such a topology. The topology contains five process- ing nodes, and each task is executed from the leftmost node to the rightmost node in a sequential manner, and therefore it has five subtasks. The second topology is a tree structure in Fig. 3(b), which is shown in sensor networks that collect sensor data at leaf nodes and relays/processes the data through intermediate nodes to the root node. The topology has 29 nodes forming a tree structure including 16 leaf nodes. Each task is executed from one of the leaf nodes to the root node in a sequential manner, and thus it has four subtasks.

We generate synthetic task sets for the two topologies, with one input parameter: the number of tasks in each task set. For the sequential-structure topology in Fig. 3(a), the number of tasks is set to 2, 3, 4, 5, 6, 7, 8, 9 and 10, and for the tree-structure topology in Fig. 3(b), the number is set to 2, 4, 6, 8, 10, 12, 14 and 16. And, each task is randomly generated as follows: D_i is uniformly chosen in [100, 10,000), and the execution time of its subtasks $\{C_{(i,k)}\}_{k=1}^{m_i}$ is chosen according to the exponential distribution of $C_{(i,k)}/D_i$ where the probability density function is $A \cdot \exp(-A \cdot x)$ with $A = 30$. Note that we remove and re-generate a task with $\sum_{k=1}^{m_i} C_{(i,k)} > D_i$ since it is impossible to meet the end-to-end deadline of the task.

For each combination of the type of topologies and the number of tasks, we generate 1000 task sets. For each task set, we assign intermediate deadlines according to our framework (in Section 5.1) using the utility function of Eqs. (14) and (17) (POS and NOS, respectively) and corresponding existing heuristic approaches PLR and NLR. To solve the convex optimization problem for POS and NOS, we use the optimization tools in MATLAB.

Fig. 4(a) and (b) plots the number of schedulable task sets when intermediate deadlines are determined by POS, NOS, PLR and NLR. Here we deem a task set schedulable, if Eqs. (10) and (11) are satisfied, i.e., the density of each node is no larger than its utilization bound, and each end-to-end timing requirement is satisfied. We observe that the number of schedulable task sets by each approach becomes smaller as the number of tasks gets larger, which is intuitive. We also observe that POS and NOS schedule more task sets than both PLR and NLR. Actually POS and NOS dominate both PLR and NLR, which means there is no task set which is schedulable by PLR or NLR, but unschedulable by POS or NOS. This is because, while PLR and NLR employ heuristic approaches to assign intermediate deadlines without considering the node schedulability, POS and NOS solve an optimization prob- lem so that they find a proper intermediate deadline assignment that makes the task set schedulable as long as such an assignment is feasible.

Another observation is that the scheduling performance gap between POS and NOS, and PLR and NLR varies with the type of topologies. In Fig. 4(a), the scheduling performance of PLR and NLR is not poor, and in particular, the difference between the number of schedulable sets by NLR and that of POS and NOS is upper-bounded by 270 in any case. However, when the number of tasks is 10 in Fig. 4(b), PLR and NLR result in only a limited number of schedu- lable task sets, while almost all task sets are schedulable by POS and NOS. This is because, for some topologies where all tasks are executed in the same nodes such as Fig. 3(a), PLR and NLR can be effective since they evenly or proportionally distribute inter- mediate deadlines. However, the same cannot be said for another type of topologies such as Fig. 3(b) since the heuristics, which do not consider which node has larger demands, do not assign longer deadlines for subtasks executed in the root node. This results in violating the root node schedulability.

One more observation is that NLR is better than PLR when all tasks are executed in the same nodes in Fig. 3(a) since it can equally distribute the contribution of each subtask to the node schedulabil- ity (i.e., $C_{(i,k)}/D_{(i,k)}$ is the same for all $1 \le k \le m_i$.). However, as shown in Fig. 3(b), this cannot be generalized because NLR cannot consider each node's demand from other tasks.

Fig. 5(a) and (b) plot the average standard deviation between POS and PLR, and NOS and NLR for task sets which are schedulable

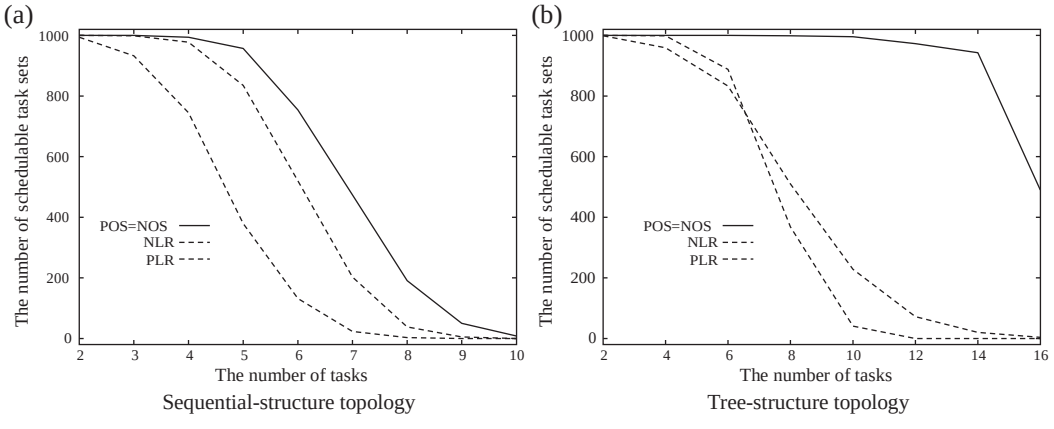


Fig. 4. Schedulability.

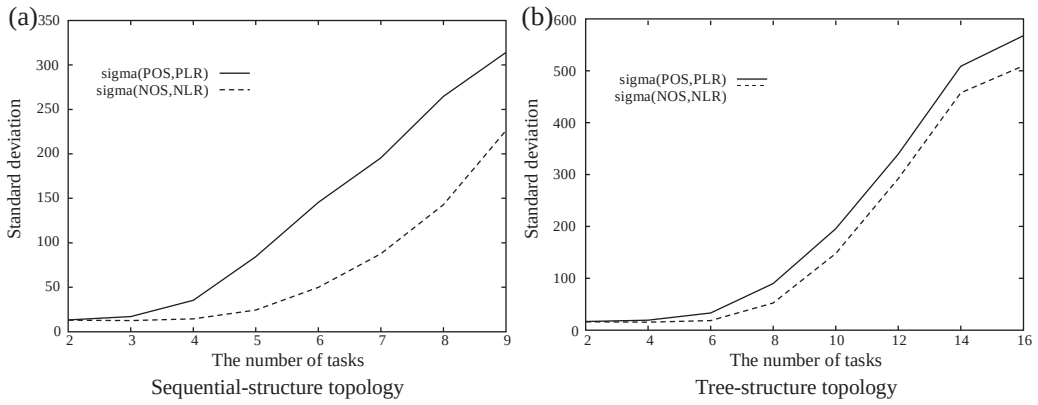


Fig. 5. Standard deviation.

by PLR and NLR.[3] These figures represent how the intermediate deadline assignment by POS and NOS is deviated from that by PLR and NLR, respectively. As shown in the figures, when the number of tasks is small, the standard deviation is almost zero, which means the assignment of POS and NOS is almost the same as PLR and NLR, respectively. This demonstrates that our choice of utility functions properly accommodate the corresponding heuristics. However, as the number of task sets gets larger, the standard deviation gets increased. This is because, at the expense of large deviation, POS and NOS make an unschedulable intermediate deadline assignment schedulable.

In summary, POS and NOS significantly improve the schedul- ing performance of PLR and NLR while they respectively emulate the intermediate deadline assignment of PLR and NLR as much as possible.

While there is significant improvement in terms of schedul- ing performance, the limitation of our framework is related to its time-complexity. A convex optimization problem, to which our framework belongs, is known to be solvable and has less time- complexity than that of a general non-linear optimization problem, but its time-complexity depends on the algorithm that is employed by the optimization problem solver (Boyd and Vandenberghe, 2004). For example, the gradient method needs $O(1/E)$ iterations to find a feasible solution with an error E (i.e., a solution X which satisfies $|X - X^*| \leq E$, where X^* is the optimal solution.) (Boyd and Vandenberghe, 2004). Therefore, for some environments where time-complexity really matters, we can adjust the tradeoff between accuracy (e.g., how accurately our framework follows heuristics) and time-complexity (e.g., the number of iterations).

8. Conclusion

This paper presented a convex optimization framework for both soft and hard distributed real-time systems, particularly, to effec- tively address the deadline assignment problem. The presented framework is suitable for finding the maximum delay-sensitive system utility for soft real-time tasks. It also provides mathe- matical foundation to existing heuristic solutions to the deadline assignment problem in hard real-time systems, fostering the understanding of these solutions and facilitating their improve- ment toward schedulability guarantees.

Several aspects of the framework are directions for further research. Our framework mainly accepts convex constraints. How- ever, most efficient schedulability conditions do not satisfy this property. Hence, one direction is to develop a new tight, con- vex schedulability condition. Our framework involves distributed computation to find an optimal solution, requiring information exchange across nodes. While such information exchange can be efficiently implemented with little extra communication cost (Athuraliya et al., 2001; Athuraliya and Low, 2000) in most cases, it could nevertheless incur some communication cost in some cases. Therefore, another interesting direction is to develop a way to obtain solutions with reasonable performance with only a little or even no information exchange, and we plan to consider a game- theoretic approach to address this problem. Another direction for future work is to extend our optimization framework toward dif- ferent wireless network environments, such as sensor networks (Raazi and Lee, 2010; Akyildiz et al., 2002) and mobile ad hoc net- works (Hieu and Hong, 2010). For example, since such networks are often bandwidth-limited, it is generally necessary to employ some resource-efficient techniques, such as selective packet drop (Lee and Shin, 2007), and we plan to explore incorporating such techniques into our optimization framework for adaptive QoS man- agement (Kim et al., 2010) and sustainable real-time guarantees (Burns and Baruah, 2008).

[3] We do not present the case of 10 tasks in Fig. 5(a), since the number of sample task sets which are scheudlable by PLR and NLR is too small.

Acknowledgements

This work was supported in part by Basic Research Lab- oratory Program (BRL, 2009-0086964), Basic Science Research Program (2010-0006650), P^3 DigiCar Research Center (NCRC, 2012-0000980), IT/SW Creative research program(NIPA-2010- C1810-1102-0003), SW Computing R&D Program of KEIT(2011- 10041313), and Global Collaborative R&D program of KIAT (M002300089) funded by the Korea Government (MEST/MKE), andKAIST-Microsoft Research Collaboration Center.

References

Akyildiz, I.F., Su, W., Sankarasubramaniam, Y., Cayirci, E., 2002. A survey on sensor networks. IEEE Communications Magazine 40 (8), 102–114.

Athuraliya, S., Low, S.H. 2000. Optimization flow control II: implementation, Tech. rep., Caltech.

Athuraliya, S., Li, V.H., Low, S.H., Yin, Q., 2001. REM: active queue management. IEEE Network, 48–53.

Atlas, A., Bestavros, A., 1998. Statistical rate monotonic scheduling. In: Proceedings of IEEE Real-Time Systems Symposium (RTSS), pp. 123–132.

Baruah, S., Howell, R., Rosier, L., 1990. Algorithms and complexity concerning the preemptive scheduling of periodic, real-time tasks on one processor. Real-Time Systems 2 (4), 301–324.

Bertsekas, D.P., Tsitsiklis, J.N., 1997. Parallel and Distributed Computation: Numer- ical Methods. Athena Scientific.

Bettati, R., Liu, J., 1992. End-to-end scheduling to meet deadlines in distributed sys- tems. In: Proceedings of International Conference on Distributed Computing Systems, pp. 452–459.

Bini, E., Cervin, A., 2008. Delay-aware period assignment in control systems. In: Proceedings of IEEE Real-Time Systems Symposium (RTSS), pp. 291–300.
Boyd, S., Vandenberghe, L., 2004. Convex Optimization. Cambridge University Press. Burns, A., Baruah, S., 2008. Sustainability in real-time scheduling. Journal of Computing Science and Engineering 2 (1), 74–97.
Chen, Y., Lu, C., Koutsoukos, X., 2007. Optimal discrete rate adaptation for distributed real-time systems. In: Proceedings of IEEE Real-Time Systems Symposium (RTSS), pp. 181–192.
Devi, U.C., Anderson, J.H., 2004. Fair integrated scheduling of soft real-time tardiness classes on multiprocessors. In: Proceedings of IEEE Real-Time Technology and Applications Symposium (RTAS), pp. 554–561.
Garcia, J., Harbour, M., 1995. Optimized priority assignment for tasks and messages in distributed hard real-time systems. In: Proceedings of IEEE Workshop on Parallel and Distributed Real-Time Systems, pp. 124–132.
Hieu, C.T., Hong, C.S., 2010. A connection entropy-based multi-rate routing protocol for mobile ad hoc networks. Journal of Computing Science and Engineering 4 (3), 225–239.
Jayachandran, P., Abdelzaher, T., 2008. Bandwidth allocation for elastic real-time flows in multihop wireless networks based on network utility maximization. In: Proceedings of International Conference on Distributed Computing Systems, pp. 752–759.
Jayachandran, P., Abdelzaher, T., 2008. Delay composition algebra: a reduction- based schedulability algebra for distributed real-time systems. In: Proceedings of IEEE Real-Time Systems Symposium (RTSS), pp. 259–269.
Jayachandran, P., Abdelzaher, T., 2009. End-to-end delay analysis of distributed sys- tems with cycles in the task graph. In: Proceedings of Euromicro Conference on Real-Time Systems, pp. 13–22.
Jonsson, J., Shin, K., 1997. Deadline assignment in distributed hard real-time systems with relaxed locality constraints. In: Proceedings of International Conference on Distributed Computing Systems, pp. 432–440.
Kao, B., Garcia-Molina, H., 1993. Deadline assignment in a distributed soft real-time system. In: Proceedings of International Conference on Distributed Computing Systems, pp. 428–437.
Kao, B., Garcia-Molina, H., 1994. Subtask deadline assignment for complex distributed soft real-time tasks. In: Proceedings of International Conference on Distributed Computing Systems, pp. 172–181.
Kim, K., Uno, S., Kim, M., 2010. Adaptive qos mechanism for wireless mobile network. Journal of Computing Science and Engineering 4 (2), 153–172.
Lee, J.K., Shin, K.G., 2007. NetDraino: saving network resources via selective packet drops. Journal of Computing Science and Engineering 1 (1), 31–55.
Lee, J., Shin, I., Easwaran, A., 2010. Online robust optimization framework for qos guarantees in distributed soft real-time systems. In: Proceedings of the 10th ACM International Conference on Embedded Software (EMSOFT), pp. 89–98.
Leung, J., Whitehead, J., 1982. On the complexity of fixed-priority scheduling of periodic real-time tasks. Performance Evaluation 2, 237–250.
Lin, K.-J., Natarajan, S., Liu, J.W.-S., 1987. Imprecise results: utilizing partial computations in real-time systems. In: Proceedings of IEEE Real-Time Systems Symposium (RTSS), pp. 210–217.
Liu, C., Layland, J., 1973. Scheduling algorithms for multi-programming in a hard- real-time environment. Journal of the ACM 20 (1), 46–61.
Low, S.H., 1999. Optimization flow control, I: basic algorithm and convergence. IEEE/ACM Transactions on Networking, 861–874.
Lumezanu, C., Bhola, S., Astley, M., 2008. Online optimization for latency assignment in distributed real-time systems. In: Proceedings of International Conference on Distributed Computing Systems, pp. 752–759.
MATLAB: The Language of Technical Computing, http://www.mathworks.com/products/matlab/.
Natale, M.D., Stankovic, J., 1994. Dynamic end-to-end guarantees in distributed real- time systems. In: Proceedings of IEEE Real-Time Systems Symposium (RTSS), pp. 216–227.
Palencia, J., Harbour, M., 2005. Response time analysis of EDF distributed Real-time Systems. Journal of Embedded Computing 1 (2), 225–237.
Raazi, S.M.K., Lee, S., 2010. A survey on key management strategies for different applications of wireless sensor networks. Journal of Computing Science and Engineering 4 (1), 23–51.
Saksena, M., Hong, S., 1996. An engineering approach to decomposing end-to-end delays on a distributed real-time system. In: Proceedings of IEEE Interna- tional Workshop on Parallel and Distributed Real-Time Systems, pp. 244– 251.
Shu, W., Liu, X., Gu, Z., Gopalakrishnan, S., 2008. Optimal sampling rate assign- ment with dynamic route selection for real-time wireless sensor networks. In: Proceedings of IEEE Real-Time Systems Symposium (RTSS), pp. 431– 441.
Stavrinides, G.L., Karatza, H.D., 2010. Scheduling multiple task graphs with end-to-end deadlines in distributed real-time systems utilizing imprecise com- putations. Journal of Systems and Software 83 (6), 1004–1014.
Tia, T.-S., Deng, Z., Shankar, M., Storch, M., Sun, J., Wu, L.-C., Liu, J.W.-S., 1995. Probabilistic performance guarantee for real-time tasks with varying computation times. In: Proceedings of IEEE Real-Time Technology and Applications Sympo- sium (RTAS), pp. 164–173.
Wu, H., Ravindran, B., Jensen, E.D., Li, P., 2005. Time/utility function decomposi- tion techniques for utility accrual scheduling algorithms in real-time distributed systems. IEEE Transactions on Computers 54 (9), 1138–1153.
Zhu, Q., Yang, Y., Scholte, E., Natale, M.D., Sangiovanni-Vincentelli, A., 2009. Optimizing extensibility in hard real-time distributed systems. In: Proceed- ings of IEEE Real-Time Technology and Applications Symposium (RTAS), pp. 275–284.

Jinkyu Lee received B.S., M.S. and Ph.D. degrees in Computer Science in 2004, 2006 and 2011, respectively, from KAIST (Korea Advanced Institute of Science and Tech- nology), South Korea. Since October 2011, he is a visiting scholar in Department of Electrical Engineering and Computer Science, University of Michigan, USA. His research interests include reliability, power management and timing guarantees in real-time embedded systems and cyber-physical systems. He won the best stu- dent paper award from the 17th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS) in 2011.

Insik Shin is currently an associate professor in Dept. of Computer Science at KAIST, South Korea, where he joined in 2008. He received a B.S. from Korea University, an M.S. from Stanford University, and a Ph.D. from University of Pennsylvania all in Computer Science in 1994, 1998, and 2006, respectively. He has been a post- doctoral research fellow at Malardalen University, Sweden, and a visiting scholar at University of Illinois, Urbana-Champaign until 2008. His research interests lie in cyber-physical systems and real-time embedded systems. He is currently a member of Editorial Boards of Journal of Computing Science and Engineering. He has been co-chairs of various workshops including satellite workshops of RTSS, CPSWeek and RTCSA and has served various program committees in real-time embedded sys- tems, including RTSS, RTAS, ECRTS, and EMSOFT. He received best paper awards, including the Best Paper award from RTSS in 2003 and the Best Student Paper Award from RTAS in 2011, and Best Paper runner-ups at ECRTS and RTSS in 2008.

Arvind Easwaran received a Ph.D. from the University of Pennsylvania, USA, in 2008 on Advances in Hierarchical Real-Time Systems: Incrementality, Optimality, and Multiprocessor Clustering. From January 2009 to October 2010, he was a Invited Sci- entist in CISTER lab, at the Polytechnic Institute of Porto, Portugal. Since November 2010, he is working as a R&D Scientist in Honeywell Aerospace, Advanced Tech- nology. He was nominated for the best paper award in Euromicro Conference on Real-Time Systems (ECRTS) in 2008 and won the best student paper award in IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS) 2011. His research interests lie in real-time embedded systems.