Meeting End-to-End Deadlines Through Distributed Local Deadline Assignment

Shengyan Hong, Thidapat Chantem, Xiaobo Sharon Hu
Department of Computer Science and Engineering
University of Notre Dame
Notre Dame, IN 46556
{shong3, tchantem, shu}@nd.edu

***Abstract*—In a distributed real-time system, transactions are executed on a number of processors and must complete by their end-to-end deadlines. Without considering resource competition among different transactions on a given processor, transaction deadline requirements may be violated. We present a distributed local deadline assignment approach that allows different transactions to have different paths and where workloads on processors may be dissimilar. Preliminary results indicate that our method significantly improves upon existing approaches.**

I. Introduction

Distributed real-time systems are widely employed in many cyber-physical control applications such as vehicle control application and multimedia communication application (e.g., [6], [13], [20]). Such systems typically require that a series of jobs be executed on a chain of processors and be completed within some end-to-end deadlines. This sequence of jobs is defined as a transaction and is periodically released. Resource competition among jobs from different transactions on a shared processor could severely increase job response times, potentially resulting in end-to-end deadline misses. Therefore, it is important to assign priorities to jobs or transactions on each processor in order to guarantee the timing requirements of the transactions in a distributed real-time system.

There are some recent papers on the priority assignment of periodic transactions or jobs in a distributed real-time system. Some assign priority to jobs based on their absolute end-to-end deadlines (e.g., [8], [10]), which is ineffective if different transactions have different paths and workloads on processors are quite dissimilar. Other local deadline assignment approaches are based on the earliest deadline first (EDF) scheduling algorithm, e.g., a slicing technique based heuristic [11], a method that exploits the execution time distributions of jobs along transaction paths [2], an on-line method based on the window-constrained scheduling [19], and a distributed method based on the jobs' precedence [17]. However, none of these methods can guarantee that jobs on a shared processor are schedulable, which may lead to eventual end-to-end deadline misses. To ensure job feasibility, the work in [12] employs the necessary and sufficient condition in [1] to assign local deadlines. However, the schedulability condition employed in [12] is not only pessimistic by assuming that the transactions are synchronized, but also extremely time consuming. The authors in [18] propose minimizing transaction resource requirements but this approach only works well for a single transaction.

In addition to the study of local deadline assignments, there are published work that focus on the schedulability analysis of transactions in distributed real-time systems. The work in [15] and [16] study how to compute the time demand bound function of transactions under EDF, while [7], [9] and [14] propose methods to compute static worst case response times for systems scheduled under fixed priority and EDF scheduling, respectively. The approach in [8] and [10] transforms the distributed real-time system schedulability test into uniprocessor schedulability test. Most of the feasibility analysis based methods are extremely time consuming and not suitable for online use, while some are only for fixed priority scheduling, which may under-utilize resources compared with EDF. In addition, the schedulability test proposed in [8] and [10] does not work well if transactions have different paths and the ratio of end-to-end deadlines to periods of jobs is much larger than the number of stages.

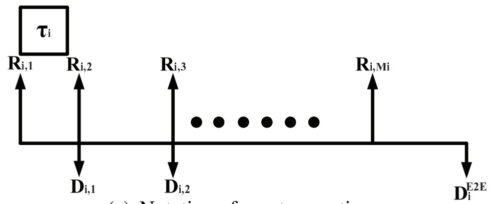
In this paper, we propose a distributed local deadline assignment approach based on the necessary and sufficient condition proposed in [4] and [5]. Our approach exploits the execution information on adjacent processors to find a feasible local deadline assignment. Our optimization approach, however, can be too time consuming for online use. We are in the process of designing an efficient heuristic. Preliminary results indicate that our approach performs much better than existing work.

II. Preliminaries

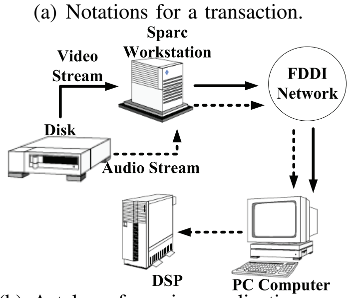
We first introduce some notations and scheduling properties. Afterwards, some motivations for our problem are presented.

A. System Model

We consider a distributed real-time system, which needs to handle a set of real-time transactions. Each transaction periodically releases a job τ_i which is characterized by D_i^{E2E} and M_i, where D_i^{E2E} denotes the absolute end-to-end deadline of τ_i and M_i is the number of stages that τ_i traverses. In the system, each job τ_i is broken into a series of jobs executing at different stages, and the job running at stage k is denoted as $\tau_{i,k}$. Each job $\tau_{i,k}$ is described by its absolute release time $R_{i,k}$, absolute deadline $D_{i,k}$, and execution time $C_{i,k}$. In this



(a) Notations for a transaction.



(b) A teleconferencing application consisting of video and audio streams [3].

Fig. 1. Notations and example transactions.

paper, we assume that $R_{i,k} = D_{i,k-1}$. Figure 1(a) shows an example of a job $\tau_{i,k}$ with its execution times, release times, and local deadlines at different stages.

Similar to the transaction models in [8], [10], we assume that there is an execution order for all the processors, i.e., processor V_x should appear before processor V_y in any job's path that contains processors V_x and V_y. For example, Figure 1(b) shows a teleconferencing application presented in [3], [13], which consists of video and audio stream processing, where the video stream and the audio stream have almost the same path except that the audio stream completes its processing in a DSP component after finishing its execution on a PC computer. We refer to any processor V_x having the execution order earlier (later) than V_y as an upstream (downstream) processor of V_y. A processor V_x has a set Ω of jobs that traverse it (i.e., are executed on it). We use $\tau_{i,k(i,x)} \in \Omega(V_x)$ to signify that job $\tau_{i,k(i,x)}$ at stage $k(i,x)$ is on processor V_x. (The function $k(i,x)$ returns the integer denoting the stage number when τ_i executes on V_x.)

One way to meet transaction end-to-end deadlines is to assign local job deadlines such that all jobs on a given processor are schedulable and that the local job deadline of the last stage is less than or equal to the respective end-to-end deadline. To ensure that enough time is left for jobs at later stages, we define the time slack of $\tau_{i,k}$ as the difference between the end-to-end deadline relative to $R_{i,k+1}$ and the sum of the execution times from stage $k+1$ to stage M_i, $\sum_{m=k+1}^{M_i} C_{i,m}$, i.e.,

$$S_{i,k} = D_i^{E2E} - R_{i,k+1} - \sum_{m=k+1}^{M_i} C_{i,m}. \tag{1}$$

The time slack gives information on the longest delay that a job can endure from the current stage to the job's final stage without violating its end-to-end deadline. Maximizing the time slack of each job on any processor gives the best opportunity to satisfy the end-to-end deadline requirements.

We assume that EDF is used on each processor since it is optimal in terms of meeting the deadline requirements for a single processor. Each job should complete execution at the last stage within its end-to-end deadline. However, preemptions on processors with different workloads along the job's path may cause some jobs to have a long response time at some stages and eventually miss their end-to-end deadlines. Our problem, then, is to minimize the response times of jobs at each stage by intelligently assigning local deadlines to them. A necessary and sufficient condition for schedulability under EDF on a uniprocessor is given below.

Theorem 1. [4], [5] *The job set $\Omega(V_x)$ can be scheduled by EDF if and only if $\forall \tau_{i,k(i,x)}, \tau_{j,k(j,x)} \in \Omega(V_x)$, $R_{i,k(i,x)} \leq D_{j,k(j,x)}$,*

$$D_{j,k(j,x)} - R_{i,k(i,x)} \geq \sum_{\substack{\tau_{r,\alpha(r,x)}, \\ R_{r,\alpha(r,x)} \geq R_{i,k(i,x)}, \\ D_{r,\alpha(r,x)} \leq D_{j,k(j,x)}}} C_{r,\alpha(r,x)}. \tag{2}$$

B. Motivations

We use a simple distributed real-time system to illustrate the deficiencies of existing approaches in satisfying the real-time requirements of transactions. The example application contains 2 transactions; their computation times and end-to-end deadlines are shown from columns 2 to 5 in Table I.

We consider two representative priority assignment methods: JFP and EDP. In a job-level fixed priority based method (JFP), the job's priority is assigned and fixed according to its absolute end-to-end deadline [8], [10]. In an end-to-end deadline partition based method (EDP), both the end-to-end deadline and execution times of each transaction at different stages are considered so as to guarantee the individual transaction's end-to-end deadline [2].

In our example, the transactions each release a job, τ_1 and τ_2, at time 0 onto processor 1. Both jobs traverse processors 1, 2, 3 and 4 sequentially. The local deadlines at each processor assigned by EDP (and resultant job response times by applying JFP and EDP) are the first value in columns 6 and 7 (first two values in columns 8 and 9) of Table I. For example, job τ_1 has its local absolute deadline 111 and response time 170 on processor 1 by applying EDP. Using JFP, job τ_1 completes its execution at stage 4 at time 1400, which is much longer than its end-to-end deadline. EDP performs a little better than JFP in reducing the response time of job τ_1, but still causes τ_1 to miss its end-to-end deadline. Since JFP ignores the workload of each job on different processors along its path, it may assign a low priority to a job with large computation times in the remaining stages and cause the job to miss its end-to-end deadline. On the other hand, EDP ignores the resource competition among different jobs on a shared processor and causes both the local and end-to-end deadlines to be missed.

Suppose there exists an algorithm that considers both the workloads along a transaction's path and resource competition among different jobs on a shared processor, both jobs τ_1 and τ_2 can meet all the deadlines. If this algorithm produces the local deadlines of jobs τ_1 and τ_2 as shown by the second values in

TABLE I

A MOTIVATING EXAMPLE CONTAINING TWO TRANSACTIONS THAT TRAVERSE FOUR PROCESSORS

Processor Name	Job τ_1 Execution Time	Job τ_1 End-to-End Deadline	Job τ_2 Execution Time	Job τ_2 End-to-End Deadline	Local Deadline Assignment EDP / OLDA Job τ_1	Local Deadline Assignment EDP / OLDA Job τ_2	Response Time JFP / EDP / OLDA Job τ_1	Response Time JFP / EDP / OLDA Job τ_2
Processor 1	100	NA	70	NA	111 / 100	90 / 170	170 / 170 / 100	70 / 70 / 170
Processor 2	200	NA	430	NA	331 / 300	663 / 730	700 / 370 / 300	500 / 700 / 730
Processor 3	100	NA	100	NA	441 / 400	797 / 830	800 / 470 / 400	600 / 800 / 830
Processor 4	600	1100	100	930	1100 / 1100	930 / 930	1400 / 1170 / 1100	700 / 900 / 930

columns 6 and 7 and the resultant response times are as given by the third values in columns 8 and 9 of Table I, then the end-to-end deadlines can all be satisfied. For example, job τ_1 may have its local absolute deadline 100 and response time 100 on processor 1, and complete its execution at the last stage at time 1100. Our effort is to design such an algorithm.

III. OUR APPROACH

A. Problem Formulation

As shown by the motivating example in Section II-B, the probability that jobs meet their end-to-end deadlines can be greatly increased if we assign appropriate local deadlines to the jobs on different processors. We propose to accomplish this with an online approach. The general idea of our online local deadline assignment method is as follows. Every time a new job arrives on some processor V_x, we reassign local deadlines to jobs on each processor starting with V_x. (In case of ties, the first processor in the total order of execution where changes occur is the starting point.) The problem of assigning local job deadlines is formulated as a mathematical programming problem aiming to maximize the time slack as defined in (1). The mathematic programming problem is then solved efficiently using an online heuristic. The process is repeated until all the processors have been handled. In our approach, we not only tackle the resource competition among different jobs on a shared processor, but also efficiently coordinate the local deadline distribution of a job at different stages along the transaction's path.

In the time slack maximization problem, our goal is to determine the local deadline $D_{i,k}$ for job $\tau_{i,k}$ such that the end-to-end deadline of τ_i is met, and the job set $\Omega(V_a)$ on any processor V_a is schedulable. We wish to maximize the time slack of each job as given in (1) subject to the schedulability constraints as given in (2) while considering the system model given in Section II-A. Specifically, for processor V_x,

$$\text{max:} \quad \min_{\tau_{i,k(i,x)}\in\Omega(V_x)} \left\{ D_i^{E2E} - D_{i,k(i,x)} - \sum_{m=k(i,x)+1}^{M_i} C_{i,m} \right\} \tag{3}$$

$$\text{s.t.} \quad D_{i,k(i,x)-1} + C_{i,k(i,x)} \le D_{i,k(i,x)} \le D_i^{E2E} - \sum_{m=k(i,x)+1}^{Mi} C_{i,m}, \quad \forall \tau_{i,k(i,x)} \in \Omega(V_x) \tag{4}$$

$$D_{i,k(i,x)} - D_{j,k(j,x)-1} \ge \sum_{\substack{\tau_{r,k(r,x)}\in\Omega(V_x),\\ D_{r,k(r,x)-1}\ge D_{j,k(j,x)-1},\\ D_{r,k(r,x)}\le D_{i,k(i,x)}}} C_{r,k(r,x)}, \quad \forall \tau_{j,k(j,x)}, \tau_{i,k(i,x)} \in \Omega(V_x). \tag{5}$$

The objective function in (3) maximizes the minimum time slack of all the jobs executed on V_a. Constraints (4)–(5) are used to specify schedulability requirements. Specifically, constraint (4) bounds the local deadline of jobs execution on V_a by the completion time of the job (left side of (4)) and the latest start time of the immediate next stage (right side of (4)). Note that $D_{i,k(i,a)-1}$ is the release time of $\tau_{i,k(i,a)}$. Constraint (5) is simply a restatement of (2).

The problem with the above mathematical programming formulation is that since the local deadlines are decision variables in constraint (5), the summation terms on the right hand side are variables and the given mathematical programming problem cannot be straightforwardly solved. To address this problem, we introduce Corollary 1.

Corollary 1. *The job set $\Omega(V_x)$ can be scheduled by EDF if and only if*

$$\max_{\tau_{i,k(i,x)}\in\omega(V_x)} \{D_{i,k(i,x)}\} - \min_{\tau_{i,k(i,x)}\in\omega(V_x)} \{D_{i,k(i,x)-1}\} \ge \sum_{\tau_{i,k(i,x)}\in\omega(V_x)} C_{i,k(i,x)}, \forall \omega(V_x) \subseteq \Omega(V_x). \tag{6}$$

In Corollary 1, $\omega(V_x)$ is a subset of $\Omega(V_x)$, $\min_{\tau_{i,k(i,x)}\in\omega(V_x)}\{D_{i,k(i,x)-1}\}$ is the minimum deadline of jobs in $\omega(V_x)$, and $\max_{\tau_{i,k(i,x)}\in\omega(V_x)}\{D_{i,k(i,x)}\}$ is the maximum deadline of jobs in $\omega(V_x)$. According to Theorem 1, the job set $\omega(V_x)$ is feasible if and only if the required computation demand $\sum_{\tau_{i,k(i,x)}\in\omega(V_x)} C_{i,k(i,x)}$ inside the time interval $\left[\min_{\tau_{i,k(i,x)}\in\omega(V_x)}\{D_{i,k(i,x)-1}\}, \max_{\tau_{i,k(i,x)}\in\omega(V_x)}\{D_{i,k(i,x)}\}\right]$ is less than or equal to the length of that interval, $\forall\omega(V_x) \in \Omega(V_x)$. Since the release times of jobs have been decided when computing the local deadlines, the summation terms on the right hand side in constraint (6) are constant, and constraint (6) can now be used to replace (5).

Although it is now possible to solve the time slack maximization problem using the formulation in (3), (4), and (6), a solution found may not be feasible at later stages since execution information of jobs on downstream processors are not considered and such jobs may miss their deadlines when trying to compete for resources with other jobs. In other words, because the mathematical programming formulation in (3), (4), and (6) only considers information relevant to the local node, it may greedily assign too much slack for jobs that do not require it instead of allocating such time slack to jobs that need it the most.

To tackle this problem, we consider job execution information on downstream processors when assigning deadlines of jobs on upstream processors, as stated in the following corollary.

Corollary 2. *If the job set $\Omega(V_y)$ can be scheduled by EDF, then,*

$$\max_{\tau_i \in \Phi(V_x,V_y)} \{D_i^{E2E} - \sum_{m=k(i,y)+1}^{M_i} C_{i,m}\} - \min_{\substack{\tau_i \in \\ \Phi(V_x,V_y)}} \{D_{i,k(i,x)} + \hat{C}\} \geq \sum_{\substack{\tau_i \in \\ \Phi(V_x,V_y)}} C_{i,k(i,y)},$$

where

$$\hat{C} = \begin{cases} \sum_{m=k(i,x)+1}^{k(i,y)-1} C_{i,m} & : \quad k(i,y) \geq k(i,x)+2 \\ 0 & : \quad k(i,y) = k(i,x)+1 \end{cases},$$

$$\Phi(V_x,V_y) = \Omega(V_x) \cap \Omega(V_y). \tag{7}$$

The corollary complements Corollary 1 in that it considers job schedulability of downstream processors in order to make local decisions based on a global view. In constraint (7), the job set $\Phi(V_x,V_y)$ contains all the jobs that are executed not only on processor V_x, but also on processor V_y, which is a downstream processor of V_x. The term $\min_{\substack{\tau_i \in \\ \Phi(V_x,V_y)}} \{D_{i,k(i,x)} + \hat{C}\}$ is the lower bound on the minimum deadline of the jobs in $\Phi(V_x,V_y)$, while $\max_{\tau_i \in \Phi(V_x,V_y)} \{D_i^{E2E} - \sum_{m=k(i,y)+1}^{M_i} C_{i,m}\}$ is the upper bound on the maximum deadline of the jobs in $\Phi(V_x,V_y)$. According to Theorem 1, if the job set $\Phi(V_x,V_y)$ is schedulable on processor V_y, constraint (7) should be satisfied. Constraint (7) can be used in conjunction with constraints (4) and (6) to exploit execution information of downstream processors.

Although it is now possible to solve this optimization problem, constraint (6) must be checked for all $2^{|\Omega(V_x)|}$ subsets of $\Omega(V_x)$, which makes searching for the solution complex and unsuitable for online use. We are working on devising a heuristic to solve the problem. Our heuristic aims to assign local absolute deadlines to jobs on any processor with time complexity $O(|\Omega(V_a)|^2)$, which would be efficient enough for online use. In addition, we intend to fully exploit execution information of downstream processors to guarantee the end-to-end deadlines of all the jobs.

IV. Summary and Future Work

We have presented a novel local deadline assignment approach to guarantee end-to-end deadlines of transactions in a distributed real-time system. The approach formulates the local deadline assignment problem as an optimization problem, which is effective even when different transactions have different paths and the workloads on different processors are dissimilar. Based on several key observations, we are designing a heuristic to solve the problem efficiently. The efficiency of the heuristic is key in a distributed on-line framework where local deadlines are assigned to newly arrived jobs or those predicted to arrive shortly. In addition, we would generalize our heuristic to handle situations where transactions have random paths in the system without any limitations. Finally, we plan on implementing our approach in a real-time operating system, and compare it with existing methods.

References

[1] S. K. Baruah, L. E. Rosier, and R. R. Howell, “Algorithms and complexity concerning the preemptive scheduling of periodic, real-time tasks on one processor,” *Real-Time Syst.*, vol. 2, no. 4, pp. 301–324, 1990.

[2] G. Buttazzo, E. Bini, and Y. Wu, “Partitioning parallel applications on multiprocessor reservations,” in *Proceedings of the 2010 22nd Euromicro Conference on Real-Time Systems*, ser. ECRTS ’10, Washington, DC, USA, 2010, pp. 24–33.

[3] S. Chatterjee and J. Strosnider, “Distributed pipeline scheduling: A framework for distributed, heterogeneous real-time system design,” 1995.

[4] H. Chetto, M. Silly, and T. Bouchentouf, “Dynamic scheduling of real-time tasks under precedence constraints,” *Real-Time Systems*, vol. 2, pp. 181–194, 1990.

[5] H. Chetto and M. Chetto, “Scheduling periodic and sporadic tasks in a real-time system,” *Information Processing Letters*, vol. 30, pp. 177–184, 1989.

[6] A. Davare, Q. Zhu, M. Di Natale, C. Pinello, S. Kanajan, and A. Sangiovanni-Vincentelli, “Period optimization for hard real-time distributed automotive systems,” in *Proceedings of the 44th annual Design Automation Conference*, ser. DAC ’07, 2007, pp. 278–283.

[7] W. Hawkins and T. Abdelzaher, “Towards feasible region calculus: An end-to-end schedulability analysis of real-time multistage execution,” in *Proceedings of the 26th IEEE International Real-Time Systems Symposium*, 2005, pp. 75–86.

[8] P. Jayachandran and T. Abdelzaher, “Transforming distributed acyclic systems into equivalent uniprocessors under preemptive and non-preemptive scheduling,” in *Real-Time Systems, 2008. ECRTS ’08. Euromicro Conference on*, 2008, pp. 233 –242.

[9] ——, “End-to-end delay analysis of distributed systems with cycles in the task graph,” in *Real-Time Systems, 2009. ECRTS ’09. 21st Euromicro Conference on*, 2009, pp. 13 –22.

[10] ——, “Delay composition in preemptive and non-preemptive real-time pipelines,” *Real-Time Systems*, vol. 40, pp. 290–320, 2008.

[11] J. Jonsson and K. G. Shin, “Robust adaptive metrics for deadline assignment in distributed hard real-time systems,” *Real-Time Syst.*, vol. 23, November 2002.

[12] D. Marinca, P. Minet, and L. George, “Analysis of deadline assignment methods in distributed real-time systems,” *Computer Communications*, vol. 27, no. 15, pp. 1412 – 1423, 2004.

[13] S. Matic and T. Henzinger, “Trading end-to-end latency for composability,” in *Real-Time Systems Symposium, 2005. RTSS 2005. 26th IEEE International*, 2005, pp. 12 pp. –110.

[14] J. Palencia and M. G. Harbour, “Offset-based response time analysis of distributed systems scheduled under edf,” *Real-Time Systems, Euromicro Conference on*, vol. 0, p. 3, 2003.

[15] A. Rahni, E. Grolleau, and M. Richard, “Feasibility Analysis of Non-Concrete Real-Time Transactions With EDF Assignment priority,” in *16th International Conference on Real-Time and Network Systems (RTNS 2008)*, 2008.

[16] N. Serreli, G. Lipari, and E. Bini, “The demand bound function interface of distributed sporadic pipelines of tasks scheduled by edf,” in *Real-Time Systems (ECRTS), 2010 22nd Euromicro Conference on*, 2010, pp. 187 –196.

[17] ——, “The distributed deadline synchronization protocol for real-time systems scheduled by edf,” in *Emerging Technologies and Factory Automation (ETFA), 2010 IEEE Conference on*, 2010, pp. 1 –8.

[18] N. Serreli and E. Bini, “Deadline assignment for component-based analysis of real-time transactions,” in *2nd Workshop on Compositional Real-Time Systems, Washington, DC, USA*, 2009.

[19] Y. Zhang, R. West, and X. Qi, “A virtual deadline scheduler for window-constrained service guarantees,” in *Real-Time Systems Symposium, 2004. Proceedings. 25th IEEE International*, 2004, pp. 151 – 160.

[20] W. Zheng, M. Di Natale, C. Pinello, P. Giusto, and A. S. Vincentelli, “Synthesis of task and message activation models in real-time distributed automotive systems,” in *Proceedings of the conference on Design, automation and test in Europe*, ser. DATE ’07, 2007, pp. 93–98.