This is the author's version of a work that was submitted/accepted for publication in the following source:


This file was downloaded from: http://eprints.qut.edu.au/41529/

© Copyright 2011 IEEE

Personal use of this material is permitted. However, permission to reprint/republish this material for advertising or promotional purposes or for creating new collective works for resale or redistribution to servers or lists, or to reuse any copyrighted component of this work in other works must be obtained from the IEEE.

Notice: Changes introduced as a result of publishing processes such as copy-editing and formatting may not be reflected in this document. For a definitive version of this work, please refer to the published source:

http://dx.doi.org/10.1109/TCST.2011.2141133
Local Adjustment and Global Adaptation of Control Periods for QoC Management of Control Systems

Yu-Chu Tian, Xiefu Jiang, David C. Levy and Ashok Agrawala

Abstract—Linking real-time schedulability directly to the Quality of Control (QoC), the ultimate goal of a control system, a hierarchical feedback QoC management framework with the Fixed Priority (FP) and the Earliest-Deadline-First (EDF) policies as plug-ins is proposed in this paper for real-time control systems with multiple control tasks. It uses a task decomposition model for continuous QoC evaluation even in overload conditions, and then employs heuristic rules to adjust the period of each of the control tasks for QoC improvement. If the total requested workload exceeds the desired value, global adaptation of control periods is triggered for workload maintenance. A sufficient stability condition is derived for a class of control systems with delay and period switching of the heuristic rules. Examples are given to demonstrate the proposed approach.

Index Terms—Control systems, QoC management, feedback scheduling, period switching, stability, multitasking

I. INTRODUCTION

Real-time and embedded control systems are conventionally developed in two separate phases: control design and its software implementation with real-time scheduling [1], [2], [3]. For control design, control theory has been well established for fixed sampling frequency, and the control periods and thus the computing workload of the task set are kept unchanged at runtime. This leads to poor use of the computing resources. For real-time scheduling, theory has been well developed under the known worst-case execution times, fixed periods, and hard deadlines [2], [4]. Many of such assumptions are conservative and do not reflect the real runtime system requirements.

The primary objective of a control system is to maintain satisfactory Quality of Control (QoC), which is characterized by some performance indices [5], e.g., integral of absolute error (IAE), integral of time absolute error (ITAE), quadratic cost function, etc. However, neither of the two separate design phases can provide a solution that can maximize the QoC of the control system. This demands the co-design of control and scheduling in real-time control systems [1], [2], [3].

The QoC of control systems has been indirectly addressed by reducing control latency and jitter in task scheduling. A method was developed in [6] for period and priority assignment in control systems. Reference [2] proposed a task model of control systems to reduce control action interval and data acquisition interval for a potential QoC improvement. Strategies were developed in [3] to reduce control latency and jitter in control task scheduling. The idea of the subtask partition [2], [3] was also investigated in [7], where the performance of a control system is evaluated to examine the benefits of some task partition schemes. To make the timing of the control output more predictable, a one-shot task model was developed for robust real-time control systems [8].

The QoC is closely related to the control period $p$. Conventional fixed-period control generates constant worst-case workload but prevents runtime resource re-allocation, leading to difficulties when a system has to add new tasks, delete existing tasks, and/or re-prioritize tasks at runtime. A shorter $p$ gives better QoC in general if the system is not overloaded. However, a too small $p$ may lead to QoC degradation [9] and excessive workload. Åström and Wittenmark [10] suggest that $p$ be chosen such that $0.2 \leq \omega_0 p \leq 0.6$, where $\omega_0$ is the natural frequency of the plant. Thus, $p$ can be made adjustable between its upper and lower bounds to provide satisfactory QoC while avoiding overloading the system. While relaxing the periodicity assumption is beneficial [11], [12], it also brings difficulties to control design and scheduling, motivating recent research on feedback scheduling of control periods.

This paper addresses period scheduling for QoC management of multitasking control systems. The main contributions include: (1) Linking real-time scheduling to the QoC directly, a hierarchical feedback QoC management framework is developed. Task decomposition, local adjustment and global adaptation of control periods, and event-triggering are embedded into the framework. The fixed priority (FP) and the earliest-deadline-first (EDF) policies are used as plug-ins. (2) Heuristic rules are proposed for runtime scheduling of control periods. (3) A sufficient stability condition is derived for a class of control systems with period switching of the heuristic rules.

The paper is organized as follows. Section II discusses related work. Section III proposes a hierarchical QoC management framework. Heuristic rules are developed for local adjustment and global adaptation of control periods in Sections IV and V, respectively. Section VI conducts system stability analysis under period switching. Case studies are given in Section VII. Finally, Section VIII concludes the paper.
II. Related Work

Several approaches were developed with different complexities in period scaling [13]. Elastic scheduling was proposed in [14] to adjust the periods for flexible workload management through the compressing algorithm. It was further developed in a general optimization framework [15]. Cervin [16] developed a method incorporating with the EDF to re-scale the periods in overload conditions. Considering integrated design of control and scheduling, a method was developed to find the optimal control frequencies [17]. All these methods focused on the schedulability of the control tasks. They did not address the QoC directly, and thus did not tell when period scaling should be activated. In comparison, this work addresses the QoC directly in period scheduling.

Feedback scheduling for general real-time systems has been adopted in control systems for dynamic workload and QoC management [18]. Cervin et al. [19] reviewed related work till 2002, and proposed a feedforward-feedback scheduling approach for control tasks by using quadratic and linear cost functions to approximate the QoC.

A feedback scheduling scheme was proposed to automatically adjust task periods without knowing the actual computation times of the tasks [20]. Following this idea, Ushio et al. [21] applied a nonlinear elastic task model to an adaptive fair sharing controller. All those developments try to formulate the problems from plant models. However, accurate plant models may not always be available [45]. Avoiding the difficulties in obtaining accurate plant models and/or analytical solutions, this work develops heuristic rules to adjust control periods to achieve satisfactory QoC.

Recently, Buttazzo et al. [5] further investigated how to manage the QoC in overload conditions. Feedback scheduling was also used to support the specified performance of dynamic systems with limited resources and unpredictable workload [22]. An integrated feedback scheduler that incorporates period adjustment with priority modification was developed for flexible QoC management [23]. A common problem in all those methods is that when the FP is used a lower-priority task with deteriorating QoC may experience a significant delay before a period adjustment and/or priority modification can be made, especially in overload conditions. As a result, the QoC of the lower-priority task may deteriorate significantly [19]. Elastic and adaptive scheduling with the EDF in [14], [16], [20], [24] will help; but they do not address the QoC directly. This work uses a task decomposition for continuous QoC evaluation, enabling quick period scaling even in overload conditions.

Effort is made to decompose real-time scheduling in several layers. A two-level pre-emptive scheduling model was described in which the global scheduler could be the EDF [25], [26]. A hierarchical scheme was proposed in [27], where an application-level feedback was used to adjust the QoC requirements of the control tasks and a system-level feedback was employed to adjust the bandwidths assigned to the tasks. Davis and Burns [28], [29] analyzed a two-level system, in which both schedulers used the FP, based on the worst-case response time. By using the same principle, the worst-case response time of tasks under a two-level EDF scheme was analyzed [30]. Hierarchical scheduling of hard real-time applications was also investigated in [31] where the local scheduler was the EDF and the global scheduler could be the FP or EDF. In hierarchical scheduling, the top-level scheduler is usually designed as a periodic task [19], wasting computing resources when there is no need to make any change. Different from the above mentioned hierarchical scheduling methods, the approach proposed in this work is even-triggered for more efficient use of computing resources. Unlike many existing task scheduling and QoC management methods which aim to maintain the processor utilization at a desired value [5], [19], it allows a much lower utilization when one or more control loops have good QoC. This has significant implications to embedded systems with limited power resources [32].

Event-triggering in control, also known as self-triggered control, is not a new idea [33] but has received increasing attention. It has been discussed for real-time scheduling of stabilizing control systems [34], stabilization [35] and robustness [36], [37] of control systems, and nonlinear control systems [38]. It has also been analyzed rigorously for a class of first-order linear stochastic systems [39]. Focusing on system stability, such rigorous analysis and development are carried out for control systems with a single control task. However, the work of this paper deals with multiple control tasks.

Most recently, effort is being made in self-triggered control of multiple control tasks. An optimization problem is formulated in [40] to deal with this problem with resource constraints. Though an analytical solution is not obtained, numerical simulation of the optimization problem showed some interesting system behaviors. Workload management in control systems with multiple control tasks is studied in [41], in which coordinated and self-triggered methods are experimentally investigated. The benefits of non-periodic control design is evaluated in [12] for networked control systems (NCSs). Further experimental results are also reported in [42] for self-triggered NCS control. For mixed types of data packets, a scheduling method was presented in [43] to stabilize an NCS. Ben Gaid et al. [44] have investigated optimal control and scheduling of NCS control tasks with limited network bandwidth. All those developments try to formulate the problems from plant models. However, as mentioned previously, accurate plant models may not always be available [45].

This paper deviates the requirements of accurate plant models and/or analytical solutions through developing heuristic rules for period switching.

Period scheduling in control systems leads to control mode switching. Even if the controller of a control loop is tuned to be able to stabilize the control loop at any fixed period within the upper and lower bounds of the control period, period switching may cause system instability. The evidence of system instability resulting from switching among stable systems is given in [46]. Despite some advances in stabilization of self-triggered control, stability analysis of control systems with period switching is still an open problem. Using general delay systems theory [47], this paper derives a sufficient stability condition for a class of control systems with input delay and period switching of the proposed heuristic rules.
III. FEEDBACK QoC MANAGEMENT ARCHITECTURE

The architecture of the proposed hierarchical feedback QoC management framework is shown in Fig. 1. It consists of two levels: QoC-driven local adjustment of control periods at the bottom level, and utilization-based global adaptation of the periods at the top level. Standard scheduling policies such as the FP and EDF can be used as plug-ins in this framework.

A task decomposition model is embedded into the framework to enable QoC evaluation for each control loop in every period even in overload conditions. The QoC evaluation module in the framework evaluates the QoC.

A periodic control task is denoted by \( T(c, d, p) \), where \( c, d, p \) are the worst-case execution time, deadline, and period, respectively. The utilization (or workload) of the task is \( U = c/p \). For \( n \) periodic control tasks running on a uni-processor, the \( i \)th task is denoted by \( T_i(c_i, d_i, p_i) \) with the utilization \( U_i = c_i/p_i, \ i = 1, 2, \cdots , n \). The total utilization of the \( n \) tasks is \( U = \sum_{i=1}^{n} U_i = \sum_{i=1}^{n} c_i/p_i \). A necessary condition for schedulability of those tasks on a uni-processor is \( U \leq 1 \). This is also a sufficient condition for the EDF.

Because the environment of a system changes over time, the \( n \) periodic tasks may overload the controller. When this happens, the overall QoC of the system deteriorates, and some of the control loops may even become unstable [5]. Rescaling the periods will help improve the QoC in overload conditions. However, as discussed previously, the existing methods do not address the QoC directly, and are also sluggish to respond to QoC changes especially in lower-priority tasks when the FP is employed [19]. Using similar ideas of [2], [3], a two-subtask decomposition model is developed in this work in order to respond to the QoC changes promptly for improved QoC management. While [2], [3] focus on reducing control latency and jitter, the task decomposition model proposed here aims to evaluate the QoC even in overload conditions, and then to use the QoC for control period scheduling.

The \( i \)th control task \( T_i(c_i, d_i, p_i) \) is decomposed into two subtasks \( T_{i1}(c_{i1}, d_{i1}, p_i) \) and \( T_{i2}(c_{i2}, d_{i2}, p_i) \) such that

\[
T_i = T_{i1} \cup T_{i2}, \quad c_{i1} + c_{i2} = c_i, \quad i = 1, 2, \cdots , n, \quad (1)
\]

where the deadlines \( d_{i1} \) and \( d_{i2} \) are set to be [3]

\[
d_{i1} = [d_i \cdot c_{i1}/c_i], \quad d_{i2} = d_i, \quad i = 1, 2, \cdots , n. \quad (2)
\]

It follows from the relationship \( c_i < d_i \) that

\[
d_{i1} > c_{i1}, \quad d_{i2} = d_i > c_{i1} + c_{i2}, \quad i = 1, 2, \cdots , n. \quad (3)
\]

The first subscript to \( T \), \( c \) and \( d \) is the task identifier, while the second one indicates the subtask (1 for data acquisition and QoC evaluation, and 2 for control computation and output).

As shown in Fig. 2, for \( n \) control tasks, the task decomposition gives \( 2n \) subtasks with subtask sets \( T^I \) and \( T^{II} \),

\[
T^I = \cup_{i=1}^{n} T_{i1}, \quad T^{II} = \cup_{i=1}^{n} T_{i2}. \quad (4)
\]

![Fig. 1. Hierarchical feedback QoC management.](image)

![Fig. 2. Task decomposition model (The priority levels are for the FP).](image)

The priorities are also assigned to the original tasks and the decomposed subtasks in Fig. 2 for the FP: the lower the number, the higher the priority. For the original task set, the priorities are determined using the Rate-Monotonic (RM) rule. Without loss of generality, assume that the \( n \) tasks \( T_1, \cdots , T_n \) have been arranged in the descending order of their priorities. The decomposed subtask set \( T^I \) inherits the priorities of the original task set. However, in order for each of the subtasks in \( T^I \) to have a chance to execute in every period, all subtasks in \( T^{II} \) are assigned lower priorities without changing the order of the priorities in the original task set. It follows that

\[
\text{Priority level of } T_{i1} : \quad n + i, \quad i = 1, 2, \cdots , n,
\]

\[
\text{Priority level of } T_{i2} : \quad i, \quad i = 1, 2, \cdots , n. \quad (5)
\]

The workload of the \( n \) tasks \( T_1, \cdots , T_n \) is:

\[
U = U^I + U^{II}, \quad U^I = \sum_{i=1}^{n} \frac{c_{i1}}{p_i}, \quad U^{II} = \sum_{i=1}^{n} \frac{c_{i2}}{p_i} \quad (6)
\]

Because \( U^I \) can be far below the full potential of the system capability, all subtasks in \( T^I \) can execute regularly in the FP with the priority assignment of Eq. (5), enabling re-scaling of periods quickly in overload conditions.

The control computation cannot start until the sampling is completed. The priority assignment in Eq. (5) for the FP reflects this constraint of task dependence and simplifies the schedulability analysis of all \( 2n \) subtasks decomposed from the original \( n \) tasks. Similar ideas are employed in [2], [3].

The schedulability of the original task set may or may not be retained in the decomposed subtask set. In Fig. 1, there may be occasions that the system becomes overloaded for both the FP and EDF due to the local period adjustment. However, the schedulability of the task set can always be achieved through global period adaptation in the framework.

IV. LOCAL ADJUSTMENT OF CONTROL PERIODS

A. QoC Characterization

Allowing evaluation of the QoC in every control period for each loop, the QoC management framework in Fig. 2
links real-time scheduling directly to the QoC. While the QoC can be evaluated by an integral form of the control error $e$, e.g., IAE, ITAE, etc., simplified QoC computation, e.g., linear approximation, is shown to be effective for real-time control [19], [48]. This work uses $e$ and its one-step difference $\delta e = e - e^{old}$ to characterize the QoC.

There are three scenarios for the QoC of a control loop:

1) If the QoC is too poor (i.e., a big $|e|$) or is deteriorating significantly (i.e., a big $|\delta e|$), more frequent control actions will help improve the QoC;

2) If the QoC is within an acceptable region (i.e., both $|e|$ and $|\delta e|$ are very small), the least frequent control can be implemented to save processor resources; and

3) Otherwise, the QoC is neither good enough nor too poor, i.e., moderate $|e|$ and $|\delta e|$. The better the QoC, the larger the control period could be set.

The following performance index captures the main features of these scenarios, and will be used to guide the development of heuristic rules for local period adjustment,

$$J = \alpha |e| + (1 - \alpha)|\delta e|, \alpha \in [0, 1],$$

(7)

where $\alpha$ is the weight of $|e|$ in the index. A stability condition will be established in Section VI for a class of control systems with period switching of the proposed heuristic rules.

B. Heuristic Rules for Local Adjustment of Control Periods

Eq. (7) shows that $J$ reaches its minimum 0 when $|e| = |\delta e| = 0$. The period of a control task is adjusted based on how far away $J$ deviates from this minimum value. Three strategies are designed for local adjustment of the period:

1) When $J$ is very close to 0, set the period $p$ of the task to its upper bound, i.e., for the $i$th task

$$p_i^{new} = p_i^{max}, \text{if } J_i \leq J_i^H, \quad i = 1, 2, \ldots, n,$$

(8)

where $J_i^H$ is a threshold, which determines how big the dead-zone is. With this strategy, when a control loop approaches its steady state, $e$ and $\delta e$ become close or equal to zero, and so does $J$. In this case, set $p = p_i^{max}$.

2) When $J$ is bigger than a threshold $J_i^H$, the QoC deteriorates significantly. Thus, set $p$ to its minimum:

$$p_i^{new} = p_i^{min}, \text{if } J_i \geq J_i^H, \quad i = 1, 2, \ldots, n,$$

(9)

3) Otherwise, i.e., when $J_i$ is between $J_i^L$ and $J_i^H$, set $p$ between its upper and lower bounds according to:

$$p_i^{new} = \min\left(p_i^{max}, \frac{p_i^{max} - p_i^{min}}{J_i^H - J_i^L}(J_i - J_i^L)\right),$$

$$\text{if } J_i^L < J_i < J_i^H, \quad i = 1, 2, \ldots, n.$$

(10)

A plot of $p^{new}$ versus $J$ is given in Fig. 3.

Moreover, the following strategy is implemented in this work to smooth out fluctuations in control periods:

$$p_i = \epsilon p^{old} + (1 - \epsilon)p^{new}, \quad \epsilon \in [0, 1],$$

(11)

where $p^{new}$ is from Eq. (8), (9) or (10); $\epsilon$ is a forgetting factor.

C. Waiting Time for Period Switching

Switching among stable systems may cause system instability [46]. As a type of control mode switching, period switching may also result in system instability. Later in Section VI, a sufficient condition will be derived which guarantees the stability of a class of control systems under period switching.

For general control systems, e.g., nonlinear systems, for which stability conditions have not been well established, the concept of waiting time for period switching is introduced. Let $t^{(wt)}$ denote the time interval from the last adjustment of $p$ to the end of the current period for the $i$th control task,

$$t_i^{(wt)} = t_i^{(wt).old} + p_i^{old}, \quad i = 1, \ldots, n,$$

(12)

where $t_i^{(wt).old}$ is the time elapsed since the last adjustment of $p$ to the beginning of the current period. According to [46], the switched system resulting from applying the strategies in Eqs. (8) through (11) is stable if $p$ is adjusted only after $t_i^{(wt)}$ becomes longer than the dwell time on average. Especially, if $p^{min} >$ the dwell time, $p$ can be adjusted in every period.

However, there is a lack of theory to analytically derive the dwell time of a general control system. Thus, a practical strategy is expected which can make the period switching slow enough for small QoC changes but sensitive enough for big QoC changes. A small change in QoC requires only a small period adjustment, and thus can be ignored to avoid frequent period switching that does not help much in QoC improvement. However, as long as the waiting time is long enough, a period switching should be activated to keep the period adjustment active. Therefore, following Eq.(11), the following strategy is designed to meet these requirements:

$$p_i = p_i^{old}, \quad \text{if } \left|\frac{p_i^{new} - p_i^{old}}{p_i^{old}}\right| < \gamma_i \quad \text{and} \quad t_i^{(wt)} < t_i^{(wt).min},$$

(13)

where $\gamma_i \geq 0$ is a threshold in relative change, $t_i^{(wt).min}$ is the minimum allowable waiting time for period switching.

D. Algorithm for Local Adjustment of Control Periods

Following the heuristic rules in Eqs. (8) through (13), Algorithm I is developed below for local adjustment of $p$.

Algorithm I: Local Adjustment of Control Periods

1: Global $c_i, p_i^{old}$; //Execution time, control period
2: Local $J_i, t_i^{(wt)}, p_i^{new}$; //QoC, period, waiting time
3: Constant $J_i^L, J_i^H, p_i^{max}, p_i^{min}, \epsilon, \gamma, t_i^{(wt).min};$
4: The $i$th subtask $T_i$;
7: if $J_i < J_i^L$ then $p_i^{new} := p_i^{max};$ //Upper bound
9: elseif $J_i \geq J_i^H$ then $p_i^{new} := p_i^{min};$ //Lower bound
V. Global Adaptation of Control Periods

When the workload $U$ is heavier than a threshold $U_d$, which is lower than but close to the total allocatable workload, the top-level utilization-based global adaptation of control periods is triggered in the proposed QoC management framework (Fig. 1). This method is different from many existing utilization-based scheduling methods in two aspects: (1) Unlike [19], it is event-triggered and thus does not run a separate periodic task at a high priority level; and (2) It is triggered only when $U > U_d$, implying that it does not globally scale down $p$ when the processor is underloaded ($U < U_d$). Thus, unlike [5], [19], it does not maintain $U$ at a desired value.

A. Event-Triggering

A separate periodic task may be used for global adaptation of the control periods [19]. However, the periodic task must execute at a high priority level if the FP is adopted. It also requires a compromise between executing the task and the promptness of the task to respond to the environmental changes. This compromise can be eliminated by using event-triggering, with which the global adaptation of the periods remains inactive when $U < U_d$ but is sensitive enough to capture the overload conditions.

Algorithm 2 shown below is for Event-Triggering for the $i$th task $T_i$.

1. Initialize $U_i := U$ for the $i$th task:

\[ U_i = U^\text{old} - U_i^\text{old} + c_i/p_i, U_i = c_i/p_i, i = 1, \ldots, n. \]  \hfill (14)

After that, it checks whether or not $U$ is too high (line 9). If not, set $N_{rq}$ (line 10), the consecutive number of requests to re-scale $p$; otherwise increment $N_{rq}$ (line 10). For smooth operation, the event-triggering will not happen until $N_{rq} \geq N_{rq}^\text{max}$ (line 11), where $N_{rq}^\text{max}$ is a threshold. After the periods are scaled up (line 12), reset $N_{rq}$ (line 13).

Algorithm 2: Event-Triggering

1: Global $U$, $U_d$; //system workload and its set-point
2: Global $c_i$, $p_i$; //Execution time, control period
3: Global $N_{rq}$; //No. of requests for increasing periods
4: The $i$th subtask $T_i$;
5: Data acquisition, and QoC evaluation;
6: Adjust $p_i$ locally if necessary; //Alg. 1
7: Update system workload $U := U^\text{old} - U_i^\text{old} + c_i/p_i$;
8: Update the $i$th task workload $U_i := c_i/p_i$;
9: if $U > U_d$ then //Load too high: event happens
   10: Increment $N_{rq}$;
11: if $N_{rq} \geq N_{rq}^\text{max}$ then //event-triggering
   12: Trigger Alg. 3 for global period adaptation;
   13: Reset $N_{rq} := 0$;
14: end if;
15: else
   16: Reset $N_{rq} := 0$;
17: end if;
18: Save results $U_i^\text{old} := U$ and $U_i^\text{old} := U_i$.

B. Global Adaptation of Periods in Overload Conditions

Once the top-level global period adaptation is triggered, it will scale up the control periods bounded by their respective upper limits. A heuristic rule to enlarge $p$ is designed as

\[ p_i = p_i^\text{old}, U/U_d. \]  \hfill (15)

In this way, the system workload is brought back to its setpoint $U_d$. Similar idea has been adopted in [19], while the difference is that the scaling is used only in overload conditions in our scheme. The top-level algorithm, Global Adaptation of Control Periods, is shown below for overload conditions:

\textbf{Algorithm 3: Global Adaptation of Control Periods.}

1: Global $U$, $U_d$;
2: loop: For tasks from $i = 1$ to $n$
3: if $p_i < p_i^\text{max}$ then
4: Scale up the control period $p_i := p_i^\text{old}, U/U_d$;
5: if $p_i > p_i^\text{max}$ then $p_i := p_i^\text{max}$, //Upper bound
6: end if;
7: Set the period of the $i$th task to be $p_i$;
8: Save result $p_i^\text{old} := p_i$;
9: end if;
10: end loop.

VI. System Stability Under Period Switching

When a control system with multiple loops is designed, each controller should be tuned to ensure the control stability under fixed period. Thus, the overall control system is stable if the processor is not overloaded and if there is no period switching. However, adjustment of $p$ at runtime implies that the system becomes a switched system. As switching among stable systems may cause instability [46], the stability of the switched system should be considered carefully when implementing period switching.

Consider a class of linear control systems with input delay $\tau_p$. In digital control, the $k$th sampling and control period is denoted as $p_k$, $k = 1, 2, 3, \ldots$. A zero-order hold (ZOH) is employed to hold the sampled data in each sampling period.

The dynamics of the closed-loop control of the system is

\[ \dot{x}(t) = Ax(t) + Bx(t_{k-1} - \tau_{pk}) \text{ for } t \in [t_{k-1}, t_k), \]

\[ t_0 = 0; t_k = t_{k-1} + p_k; k = 1, 2, \ldots, \]  \hfill (16)

where variable $p_k$ represents period switching.

The main result of the stability analysis for this system with period switching is given in the following theorem, which is derived from delay systems theory [47].
Theorem 1: For given $p^{\text{max}} \geq p_k$ and $\tau^{\text{max}} \geq \tau_{pk} \forall k = 1, 2, \cdots$, if there exist matrices $P > 0, Q > 0$ and $R > 0$ such that
\[
\Phi = \begin{bmatrix}
PA + A^T P + Q - R + (\tau^{\text{max}} + p^{\text{max}})^2 A^T R A
B^T P + \tau^2_{\text{max}} B^T R A + R
PB + (\tau^{\text{max}} + p^{\text{max}})^2 A^T R B + R
(\tau^{\text{max}} + p^{\text{max}})^2 B^T R B - 2R
R & -Q - R
\end{bmatrix} < 0
\] (17)
holds, then the closed-loop control system in (16) with time delay $\tau_{pk}$ and period switching $p_k$ is asymptotically stable.

Proof: Using delay systems theory [47], we construct a Lyapunov-Krasovskii functional candidate as
\[
V(x(t)) = x^T(t) P x(t) + \int_t^{t+\tau_{\text{max}} - p^{\text{max}}} x^T(s) Q x(s) ds + (\tau^{\text{max}} + p^{\text{max}}) \int_{t - \tau_{\text{max}} - p^{\text{max}}}^{t} ds \int_{s}^{t+\tau_{\text{max}} - p^{\text{max}}} \dot{x}^T(\theta) R \dot{x}(\theta) d\theta.
\] (18)

Omitting the detailed process, we can prove $\dot{V}(x(t)) < 0$ along the trajectory of Eq. (16) with period switching. ■

Remark 1: The sufficient condition given in Theorem 1 relates system stability to variable period $p_k$ and variable time delay $\tau_{pk}$. It holds over a sequence of periods $p_0, p_1, \cdots$, which evolve with period switching. Thus, the stability of the system in Eq. (16) under period switching is guaranteed.

Remark 2: If System (16) has a constant delay, i.e., $\tau_{pk} = \tau_p$. Theorem 1 still holds with $\tau^{\text{max}}$ replaced by $\tau_p$.

VII. CASE STUDIES

A. Processes, Controllers and System Stability

Consider open-loop unstable processes governed by
\[
G_p(s) = \frac{K_p}{s(T_p s + 1)}.
\] (19)

Three such processes representing DC motors are considered, as shown in Table I. This example is taken from [49].

| Table I | Task settings with $c_i = 2$ms and $d_i = p_i$ ($i = 1, 2, 3$).

<table>
<thead>
<tr>
<th>$G_1(s)$</th>
<th>$G_2(s)$</th>
<th>$G_3(s)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Priority (for FP)</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>$p^{\text{min}}$ (ms)</td>
<td>3.6</td>
<td>4.0</td>
</tr>
<tr>
<td>$p^{\text{max}}$ (ms)</td>
<td>9.0</td>
<td>10</td>
</tr>
<tr>
<td>$U_i^{\text{nominal}}$ (GA)</td>
<td>16.67</td>
<td>15.00</td>
</tr>
<tr>
<td>$U_i^{\text{nominal}}$ (%)</td>
<td>41.67</td>
<td>37.50</td>
</tr>
<tr>
<td>Nominal $p_i$ (ms)</td>
<td>5.8</td>
<td>6.4</td>
</tr>
<tr>
<td>Nominal $U_i$ (%)</td>
<td>35.7</td>
<td>31.9</td>
</tr>
<tr>
<td>$\sum U_i^{\text{nominal}} = 91.01%$, $\sum U_i^{\text{nominal}} = 60.40%$, $\sum (\text{Nominal } U_i) = 94.4%$.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Controller: $K_c = 0.96; \varepsilon_c = 0.12; T_{cd} = 0.004$; $\beta = 0.8$; $N_d = 10$.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

A proportional-integral-derivative (PID) controller is adopted for all three processes
\[
G_c(s) = K_c \left[ 1 + \frac{1}{(T_{ci}s) + T_{cd}s} \right],
\] (20)
where $K_c$, $T_{ci}$ and $T_{cd}$ are controller gain, integral and derivative times, respectively. Shown in Table I, the controller settings are taken from [49] as well. They are tuned for $G_2$, but are also applied to $G_1$ and $G_3$ to simulate model mismatch.

To apply the stability result in Theorem 1, consider Eqs. (19) and (20) in digital control with a ZOH. For $t \in [k-1, k]$, where $t_0 = 0, t_k = k - 1 + p_k, k = 1, 2, \cdots$, we have
\[
K_p K_c \left[ r(t_{k-1}) - y(t_{k-1}) \right] + T_{ci} \left[ \dot{r}(t_{k-1}) - \dot{y}(t_{k-1}) \right] + T_{cT_{cd}} \left[ \ddot{r}(t_{k-1}) - \ddot{y}(t) \right],
\]
where $y(t)$ and $r(t)$ are process output and setpoint, respectively. Letting $x_1(t) = y(t), x_2(t) = \dot{y}(t)$ and $x_3(t) = \ddot{y}(t)$, we can obtain the state space model of Eq. (16) with
\[
x(t) = \left[ x_1(t), x_2(t), x_3(t) \right]^T,
A = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & -1/T_p \end{bmatrix},
B = -\frac{K_p K_c}{T_p T_{ci}} \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & T_{ci} & T_{cT_{cd}} \end{bmatrix}.
\]

Then, from this state space model, we can obtain that with the system settings in Table I the linear matrix inequality in Eq. (17) of Theorem 1 always holds for $p_k < 20$ms for all three control loops. Table I shows that the designed maximum periods are only 9ms, 10ms and 11ms for $G_1, G_2, G_3$, respectively. It is known from Theorem 1 that the asymptotical stability of the system with period switching is guaranteed even if period switching happens at the end of each period. Therefore, the minimum waiting time for next period switching in Algorithm 1 can be set to be the same as the current period.

For digital control, the PID controller in Eq. (20) is implemented in the following discrete-time form [50]:
\[
u(t_k) = K_c \left[ \beta r(t_k) - y(t_k) \right] + I(t_k) + D(t_k),
I(t_k) = I(t_{k-1}) + \frac{K_c}{T_p} \left[ r(t_{k-1}) - y(t_{k-1}) \right],
D(t_k) = \frac{T_{ci}}{p_{N_d} + 1} D(t_{k-1}) + \frac{N_d K_c}{p_{N_d} + 1} \left[ y(t_{k-1}) - y(t_k) \right],
\]
where $u$ represents control signal; $I$ and $D$ are integral and derivative actions, respectively; and $\beta$ and $N_d$ are filter parameters. The subscript $k$ indicates the $k$th period.

To evaluate the QoC of the system under fixed- and variable-period scheduling, step changes in setpoint are introduced into the three loops: 1) Loop $G_1(s)$: $+1$ (8s), $-1$ (1s), $+1$ (2s); 2) Loop $G_2(s)$: $+1$ (8s), $-1$ (1s); and 3) Loop $G_3(s)$: $+1$ (8s). The worst-case scenario occurs at $t = 0$ when all loops request $p^{\text{min}}$.

B. The FP and EDF Scheduling Under Fixed Periods

Under $p^{\text{min}}$ and the FP, the system is overloaded with the requested $U = 151\%$. The $G_2$ loop behaves with oscillations because it often misses its deadlines. The $G_3$ loop becomes unstable since the control task has no chance to execute.

Under $p^{\text{min}}$ and the EDF, all control tasks can execute. However, the control tasks often miss their deadlines due to the overload condition. The ITAE indices listed in Table II for MinPeriods with EDF indicates that significant improvement can be expected through better QoC management. $p^{\text{max}}$ and $p^{\text{nominal}}$ are also tested under the FP. The results are tabulated in Table II as well. They are much better than those from $p^{\text{min}}$ with either the FP or the EDF.

C. Task Decomposition and Variable-Period Scheduling

The task decomposition of the original three tasks and all settings for re-scaling of $p$ are tabulated in Table III.
Loop G1: working well; G2: oscillatory; G3: unstable

\[ \sum (\text{ITAE} \times 10^3) = 31.2; \text{U} = 63.63\% \]

VariablePeriod FP (from \( p^{max} \))

<table>
<thead>
<tr>
<th>Loop</th>
<th>0~1s</th>
<th>1~2s</th>
<th>2~3s</th>
</tr>
</thead>
<tbody>
<tr>
<td>G1</td>
<td>12.6990</td>
<td>12.8637</td>
<td>12.8018</td>
</tr>
<tr>
<td>G2</td>
<td>12.9202</td>
<td>13.0646</td>
<td>–</td>
</tr>
<tr>
<td>G3</td>
<td>12.8634</td>
<td>–</td>
<td>–</td>
</tr>
</tbody>
</table>

\[ \sum (\text{ITAE} \times 10^3) = 77.2; \text{U} = 151\% \]

VariablePeriod EDF (from \( p^{max} \))

<table>
<thead>
<tr>
<th>Loop</th>
<th>0~1s</th>
<th>1~2s</th>
<th>2~3s</th>
</tr>
</thead>
<tbody>
<tr>
<td>G1</td>
<td>5.1395</td>
<td>5.1621</td>
<td>5.3074</td>
</tr>
<tr>
<td>G2</td>
<td>5.2672</td>
<td>5.1621</td>
<td>–</td>
</tr>
<tr>
<td>G3</td>
<td>4.5906</td>
<td>–</td>
<td>–</td>
</tr>
</tbody>
</table>

\[ \sum (\text{ITAE} \times 10^3) = 29.7; \text{U} = 94.4\% \]

Start with \( p^{max} \) under either the FP or EDF with local adjustment and global adaptation of \( p \). The results are shown in Figs. 4, 5 and 6. It is seen from Figs. 4 and 5 that both the FP and EDF with variable \( p \) can stabilize the open-loop unstable processes and demonstrate good performance in setpoint tracking. They also show comparable ITAE indices (Table II under “VariablePeriod FP” and “VariablePeriod EDF”).

<table>
<thead>
<tr>
<th>Subtasks</th>
<th>Priority for FP</th>
<th>Period (ms) (Table I)</th>
</tr>
</thead>
<tbody>
<tr>
<td>G1: {T11,T12}</td>
<td>2, 5</td>
<td>3.6 - 9.0</td>
</tr>
<tr>
<td>G2: {T21,T22}</td>
<td>3, 6</td>
<td>4.0 - 10.0</td>
</tr>
<tr>
<td>G3: {T31,T32}</td>
<td>4, 7</td>
<td>1.4 - 11.0</td>
</tr>
</tbody>
</table>

Fig. 4. Dynamic adjustment of periods under the FP (plots on the left: \( p_1 \) to \( p_3 \) and \( U \); plots on the right: \( y_1 \) to \( y_3 \); average \( U = 63.63\% \)).

Figs. 4 and 5 illustrate that both the FP and EDF give similar patterns in period adjustment and workload adaptation. When the QoC of a control loop deteriorates significantly (e.g., at 1s, 2s and 3s), the corresponding \( p \) is reduced locally for more frequent control actions, resulting in an increased demand on the processor utilization \( U \). In contrast, when the QoC is improved, \( p \) is enlarged and thus the requested \( U \) is reduced without sacrifice of the QoC. Therefore, compared with the FP with fixed nominal periods (\( U = 94\% \)), the FP and EDF with variable \( p \) gives comparable ITAE indices (Table II) with their average \( U \) being as low as about 64%.

However, when \( U > U_d \), \( p \) is enlarged through global period adaptation. Fig. 6 shows that in the worst-case scenario in which a unit step setpoint change is introduced at \( t = 0s \) into all three loops, the top-level global adaptation is triggered only four times. \( U \) is well maintained under \( U_d \) almost all the time, even in the worst-case scenario.

VIII. CONCLUSION

Linking multi-tasking scheduling directly to the QoC, a hierarchical feedback QoC management framework has been developed for integrated design of control and scheduling of real-time control systems with multiple control tasks. It consists of a task decomposition model for continuous QoC monitoring, event-triggered local adjustment of control periods for QoC improvement, and event-triggered global adaptation of the periods for workload management. A set of heuristic rules have been proposed for feedback scheduling of control periods. A sufficient condition has also been derived for a class of linear control systems to guarantee the stability of the systems with period switching. Case studies have been conducted to demonstrate the developed feedback QoC management framework.

REFERENCES

