Load adjustment in real-time processing systems
W. Tarng, a T.-H. Lin b

aDepartment of Mathematics and Science Education, National Hsin-Chu Teachers College, Hsin-Chu, Taiwan, Republic of China
bDepartment of Electrical and Computer Engineering, State University of New York at Buffalo, Buffalo, NY 14260, USA

Abstract

Modern process control systems are usually incorporated with high-speed computers to satisfy the requirements of real-time response and growing complexity. In such systems, the whole mission is divided into a number of tasks which are scheduled to meet their real-time constraints. In this paper, an adaptive scheduling algorithm based on current task queue length is proposed to handle the problem of transient overload in real-time processing systems due to the change of task characteristics. Our simulation result shows that the queue-length-based (QLB) algorithm can achieve better performance than traditional static scheduling algorithms.

1 Introduction

The major difference between real-time systems and general-purpose computer systems is that the correctness of a real-time system depends on not only the logical results of computation but also the time when these results are produced. Modern process control systems are usually incorporated with high-speed computers to satisfy the requirements of real-time response and growing complexity. In such systems, the whole processes are divided into a number of tasks and then scheduled to meet their real-time constraints (or deadlines). A deadline is hard if missing the deadline may result in catastrophic results, and is soft if missing the deadline only cause some degree of performance degradation.
Applications of hard real-time systems like airplane or nuclear reactor control systems are life-critical and therefore tasks must be scheduled carefully to meet their deadlines. When designing a real-time system, tasks are often categorized as periodic and aperiodic to simplify the scheduling mechanism and to guarantee the hard deadlines. Periodic tasks are invoked repeatedly by fixed time intervals while aperiodic tasks may arrive randomly. A general assumption for real-time scheduling is that periodic tasks must meet the hard deadlines equal to their periods and aperiodic tasks have soft deadlines to be satisfied. Because only periodic tasks have hard deadlines, they will be scheduled first and thus given higher priority over aperiodic tasks. However, the response times of aperiodic tasks can also be reduced if they are scheduled by the time slots allocated to a high-priority periodic server [1,2].

In [3], the period and service time of periodic server can be determined according to the deterministic characteristics of periodic tasks (computation times and periods) and the stochastic characteristics of aperiodic tasks (inter-arrival-time and computation-time distributions) to further improve the system performance. In practice, the periods and computation times of periodic tasks or even tasks themselves may change during different stages of system operation, for example, the take-off, cruise, and landing stages in a flight control system. Therefore, static scheduling algorithms can not fulfill the requirements of such systems in spite of their simplicity and low scheduling overhead. An adaptive scheduling algorithm based on dynamic period transformation mechanism [4] is proposed to handle the problem of transient overload due to the change of task characteristics in real-time systems. The adaptive algorithm defines a number of static schedules optimized for different load levels and switches among these schedules during run time to improve system performance. A real-time mission is first divided into several operational phases according to their functional requirements and estimated load conditions. In each phase, a static algorithm is used to schedule periodic tasks including the periodic server of which the service rate is determined based on the arrival rates of aperiodic tasks. To accommodate the change of load conditions, modification of task periods are performed when the system moves from one phase to another.

Using the gradient method, the optimal periods in each phase can be derived a priori from the deterministic characteristics of periodic tasks and stochastic characteristics of aperiodic tasks. However, in certain environments the stochastic characteristics of aperiodic tasks can not be estimated clearly such that it is very difficult to determine the service rate of the periodic server. One possible solution is to measure these parameters on-line and perform period transformation accordingly. In this paper, a queue-length-based (QLB) version of dynamic period transformation algorithm is proposed for dynamic load adjustment in real-time systems, and it is suitable for the applications where the stochastic characteristics of aperiodic tasks are not available in advance. The operation of the QLB algorithm is similar to the original dynamic period transformation mechanism except that the phases are specified by the number of aperiodic tasks waiting for execution. Usually, the periodic server in a phase
with more waiting aperiodic tasks has higher service rate than it does in a phase with less waiting aperiodic task. The QLB algorithm monitors the number of pending aperiodic tasks whenever the CPU is idle to decided whether a period transformation is necessary. If the queue length does not match the defined status of current phase, a period transformation is performed immediately to adapt to the load change by increasing or decreasing the service rate of the periodic server. Our simulation result shows that the QLB algorithm can provide better performance than traditional static scheduling algorithms.

The remainder of this paper is organized as follows. In Section 2, the dynamic period transformation mechanism is reviewed first and then the QLB algorithm is presented. The performance measure to determine the optimal periods of periodic tasks and the service rate for aperiodic tasks in the QLB algorithm is described in Section 3. A queueing model is created to evaluated the performance of the QLB algorithm in Section 4. Section 5 is the conclusion of this paper.

2 The QLB Dynamic Period Transformation Algorithm

The dynamic period transformation mechanism [4] can achieve load adjustment by dividing a real-time mission into several operational phases, where the periods of periodic tasks and service rate for aperiodic tasks are determined by the gradient method in advance to optimize system performance. During system operation, a dynamic period transformation is performed when the mission moves from one phase to another to change the periods of periodic tasks and service rate for aperiodic tasks according to current load condition. In each phase, the rate-monotonic (RM) algorithm [5], the optimal static scheduling algorithm, is used to schedule periodic tasks including the periodic server. The RM algorithm schedules periodic tasks according their priorities, which are predetermined according to the periods of tasks. The rule for assigning priorities is that tasks with shorter periods are given higher priorities. Usually, the period of the server is the shortest in order to receive the highest priority to provide better responsiveness for aperiodic tasks.

Because the computation resources such as CPU time and I/O channels are fixed, increasing the service rate of aperiodic tasks (by decreasing the period of server) will require increasing the periods of other tasks as compromise. An important part of the dynamic period transformation algorithm is to find the optimal periods in each phase, which are determined by the gradient method based on a pre-defined performance measure. For safety reason, the satisfaction of deadlines for periodic tasks in each phase must be examined off-line by the static checking algorithm proposed in [3]. Also, it is better to perform the dynamic period transformations when the CPU is idle, because idle times can separate the worst-case conditions in two different phases. Usually, the CPU idle times occur frequently and therefore the delay time for the request of period transformation is often very short.
Using the gradient method, the optimal periods of real-time tasks in each phase can be determined to improve the system performance. However, not all real-time missions can be clearly divided into a number of operational phases or provide stochastic characteristics of aperiodic tasks in each phase before their operation. In that case, it is very difficult to define the operational phases or to determine the service rate of periodic server. Therefore, we propose the QLB algorithm which performs period transformation based on the current workload of aperiodic tasks, i.e., the number of waiting aperiodic tasks in the task queue. In this approach, only the computation-time distribution of aperiodic tasks is required for the determination of task periods in each phase. For example, we can create a number of $n+1$ phases for a real-time mission and each phase is defined by the number of pending aperiodic tasks. The mission will stay in Phase $j$, $0 \leq j \leq n-1$, if the number of waiting aperiodic tasks equals $j$, and in Phase $n$ if there are at least $n$ aperiodic tasks pending. Usually, the service rate of the server is higher in the phase where the queue of waiting aperiodic tasks is longer. In Phase 0, because no aperiodic tasks are waiting for execution, the server can be suspended to reduce the overhead and CPU time utilization. A wider range of load conditions (queue length) can be covered by increasing the number of phases or states in each phase. The latter approach has the advantage of lower overhead since period transformations are performed less frequently, but it also has the disadvantage of longer reaction time to load changes. The operation of the QLB algorithm is very simple. It obtains the number of waiting aperiodic tasks at the beginning of idle times to decide whether a period transformation is necessary. If the current queue length does not match the specified condition of current phase, a period transformation is performed to adapt to the load change by increasing or decreasing the service rate of the server. The following example shows the operation of the QLB algorithm.

**Example 1:**
Three phases are created for a real-time mission by the QLB algorithm to schedule three periodic tasks and aperiodic tasks in Figure 1. The legitimate queue length of aperiodic tasks in each phase is defined as zero in Phase 0, one in Phase 1, and two or more in Phase 2, respectively. As described before, $\tau_1$ is used as the server for aperiodic tasks. The mission starts with Phase 0 at $t=0$. Because no aperiodic tasks have arrived, only $\tau_2$ and $\tau_3$ are invoked and $\tau_2$ is executed first. An aperiodic task $\tau_{a1}$ arrives at $t=3.5$, which requires five units of computation time. The first idle time appears at $t=5$ after the first iteration of $\tau_3$ is completed. Because the number of aperiodic tasks has been changed to one, a period transformation is performed to move the system state from Phase 0 to Phase 1. The periodic server $\tau_1$ with one unit of service time is invoked immediately to serve $\tau_{a1}$. The second idle time occurs at $t=8$ after the second iteration of $\tau_3$ is completed. Since the number of aperiodic task is not changed, the system stays in Phase 1 and the idle time can also be used to execute $\tau_{a1}$. Another aperiodic task $\tau_{a2}$ arrives at $t=10.5$ with computation time equal to two.
The third idle time occurs at $t=ll$ after the third iteration of $\tau_2$ is completed. Since the number of pending aperiodic tasks has now become two, the system switches from Phase 1 to Phase 2 to increase the service rate of $\tau_1$.

<table>
<thead>
<tr>
<th>Task</th>
<th>Phase 0</th>
<th>Phase 1</th>
<th>Phase 2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$C_i$</td>
<td>$T_i$</td>
<td>$C_i$</td>
</tr>
<tr>
<td>$\tau_1$</td>
<td>Suspended</td>
<td>1.0</td>
<td>4.0</td>
</tr>
<tr>
<td>$\tau_2$</td>
<td>1.5</td>
<td>3.0</td>
<td>1st</td>
</tr>
<tr>
<td>$\tau_3$</td>
<td>2.0</td>
<td>5.5</td>
<td>2nd</td>
</tr>
</tbody>
</table>

Figure 1: The operation of the QLB algorithm.

3 The Optimal Periods of The QLB algorithm

An important part for the development of the QLB algorithm is to find the optimal periods in each operational phase. The definition of operational phases in the QLB algorithm is based on current queue length of aperiodic tasks. When the computation-time distribution of aperiodic tasks and the service rate of periodic server are given, the mean response time of aperiodic tasks can be estimated [3]. In many applications, the performances of real-time systems such as accuracy and stability are strongly related to the periods or response times of tasks. Usually, a shorter period or response time can produce better results or higher stability. However, it is not possible to schedule all tasks with very short periods since the computation resources are fixed. A systematic approach for improving system performance is to determine the periods of tasks according to their importance and timing constraints. We develop the following performance measure to determine the optimal periods of tasks in the QLB algorithm based on the gradient method.

Our performance measure is a vector comprising a number of components to evaluate each task, where the performance of a periodic tasks is evaluated by its period, and the performance of aperiodic tasks is evaluated by their mean response time. We define the longest period that a periodic tasks can be scheduled to satisfy the worst-case requirements as its boundary period. To determine the optimal periods, the performance vector has to be converted into a scalar by the mapping of the performance vector to a weight vector specifying the importance of each individual task. For a real-time system containing $m$
periodic tasks $\tau_1, ..., \tau_m$ and the aperiodic task $\tau_a$, the system cost is determined based on the following assumptions.

1. The computation time $C_i$ of periodic task $\tau_i$ is a constant and the cost of running $\tau_i$ is a non-decreasing function $\Psi_i$ of its period $T_i$.

2. The computation times of aperiodic task $\tau_a$ are exponentially distributed with mean $\mu$, and the cost of running $\tau_a$ is a non-decreasing function $\Psi_a$ of its mean response time.

According to the above assumptions, the mean response times of aperiodic tasks can be approximated by $\frac{1}{\mu}$ because $T_1$ is usually very small, and the cost of running aperiodic tasks can then be expressed as the cost of running the server, which is also a non-decreasing function of its period $T_i$ because $C_i$ and $\mu$ are all constant. Finally, the system cost is computed by $\Phi(T_1, ..., T_m) = \sum_{i=1}^{m} w_i \Psi_i(T_1)$, where $w_1, ..., w_m$ are the weights of individual tasks.

The optimization process is performed to minimize the system cost $\Phi(T_1, ..., T_m)$ using the gradient method subject to the timing constraints of periodic tasks. At the beginning of optimization process, the periods of all tasks are set to their boundary values $T_1, ..., T_m$. Then, the minimization of system cost is performed step by step by moving the period vector $(T_1, ..., T_m)$ towards the opposite direction of the gradient vector $(\frac{\partial \Phi}{\partial T_1}, ..., \frac{\partial \Phi}{\partial T_m})$. In each step, the FC algorithm [3] is used to check whether all periodic tasks can satisfy their timing constraints. If the deadlines of all periodic tasks are satisfied, we compute the gradient vector again and repeat the same procedure. Otherwise, instead of decreasing all periods at the same time, we start to reduce the periods one at a time to further minimize the system cost. Similarly, the FC algorithm has to be applied for the checking of timing constraints. The whole procedure stops when all periods can not be reduced any further. Since the optimal periods in each phase are determined by the gradient method, the QLB algorithm can always provide the best results no matter how the arrivals of aperiodic tasks change.

4 Performance Evaluation

In this section, an $M/M/n$ queueing model is created to analyze the performance of the QLB algorithm. We will compare the performance of the QLB algorithm with that of the RM algorithm under different load conditions. For a mission containing $n+1$ phases, the $M/M/n$ queueing model can be used to compute the probability of staying in each phase and the long-term system performance can then be evaluated by the mean of system costs in all created phases. For simplicity, the $n+1$ phases of the QLB algorithm are defined in such a way that the legitimate number of waiting aperiodic tasks in Phase $i$ is equal to $i$ when $i < n$. Also, the number of legitimate waiting aperiodic tasks can be greater than or equal to $n$ in Phase $n$. 
According to the operation of the QLB algorithm, any discrepancy in the queue length of aperiodic tasks will result in a period transformation to switch the system from current phase to one of the neighboring phases. The state transition diagram of the simplified model is shown in Figure 2, where $\lambda$ is the arrival rate of aperiodic tasks and $B_i \mu$ is the service rate of the periodic tasks in Phase $i$. The service rate is computed by $B_i \mu = \frac{c^i}{T_i^r}$ where $T_i^r$ is the optimal period of the server in Phase $i$. A question may arise for the introduction of aperiodic tasks' arrival rate $\lambda$ here because the QLB algorithm is designed specifically for the applications where the arrivals of aperiodic tasks can not be estimated. In fact, the parameter $\lambda$ in our queueing model is only used to evaluate the performance of the QLB algorithm in general environments and is not needed in the determination of task periods.

![State transition diagram of the QLB algorithm](image)

If the probability that $i$ aperiodic tasks are waiting in task queue is denoted by $p_i$, then we have

$$p_i = \begin{cases} \frac{p_0 (\frac{\lambda}{\mu})^i}{\prod_{j=1}^{i} B_j}, & 0 \leq i \leq n-1 \\ \frac{p_0 (\frac{\lambda}{\mu})^i}{B_n^{i-n} \prod_{j=1}^{n} B_j}, & i \geq n \end{cases}$$

(1)

Since $\sum_{i=0}^{\infty} p_i = 1$, we can derive $p_0$ from Equation (1) as

$$p_0 = \frac{1}{1 + \sum_{i=1}^{n-1} \frac{(\frac{\lambda}{\mu})^i}{\prod_{j=1}^{i} B_j} + \sum_{i=n}^{\infty} \frac{(\frac{\lambda}{\mu})^i}{B_n^{i-n} \prod_{j=1}^{n} B_j}}$$

(2)

All other $p_i$'s can be obtained by the substitution of $p_0$ from Equation (2) into Equation (1). The probability for the system to stay in phase $i$, denoted by $P_i$, can now be computed as
The following example shows the computation of probabilities and system performance for a real-time mission scheduled by the QLB algorithm.

Example 2:

A real-time mission containing five periodic tasks \(\tau_1, \tau_2, \tau_3, \tau_4,\) and \(\tau_5\) as listed in Table 1 are scheduled by the QLB algorithm and their optimal periods in each phase are computed by the gradient method and listed in Table 2. It is assumed that the arrival rate of aperiodic tasks \(\lambda = 0.1\) and the mean computation time of aperiodic tasks \(1/\mu = 1\). Four phases are created where the optimal periods of periodic tasks are determined by the gradient method and listed in Table 2.

\[
P_i = \begin{cases} 
p_i, & 0 \leq i \leq n - 1 \\
1 - \sum_{j=0}^{n-1} p_j, & i = n
\end{cases}
\] (3)

The probability \(P_0, P_1, P_2,\) and \(P_3\) are calculated as 0.35, 0.31, 0.23, and 0.11, respectively, and the system cost in phase \(i\) is computed by

\[
\Phi(T_1^i, T_2^i, T_3^i, T_4^i, T_5^i) = 0.5T_1^i + 4e^{0.1T_2^i} + 3e^{0.1T_3^i} + 2e^{0.1T_4^i} + e^{0.1T_5^i}
\] (4)

and also listed in Table 2. Finally, the average system cost \(E[\Phi]\) for the whole mission is computed as

<table>
<thead>
<tr>
<th>Phase</th>
<th>(T_1^i)</th>
<th>(T_2^i)</th>
<th>(T_3^i)</th>
<th>(T_4^i)</th>
<th>(T_5^i)</th>
<th>(\Phi)</th>
<th>(P_i)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Phase 0</td>
<td>(\infty)</td>
<td>9.6</td>
<td>9.6</td>
<td>19.1</td>
<td>19.1</td>
<td>38.5</td>
<td>0.35</td>
</tr>
<tr>
<td>Phase 1</td>
<td>8.8</td>
<td>13.1</td>
<td>13.1</td>
<td>26.1</td>
<td>51.4</td>
<td>0.31</td>
<td></td>
</tr>
<tr>
<td>Phase 2</td>
<td>7.4</td>
<td>11.1</td>
<td>11.1</td>
<td>22.1</td>
<td>22.1</td>
<td>56.0</td>
<td>0.23</td>
</tr>
<tr>
<td>Phase 3</td>
<td>3.4</td>
<td>16.6</td>
<td>16.6</td>
<td>33.1</td>
<td>87.3</td>
<td>0.11</td>
<td></td>
</tr>
</tbody>
</table>
\[ E[\Phi] = \sum_{i=0}^{3} P_i \Phi(T_1^i, T_2^i, T_3^i, T_4^i, T_5^i) \]
\[ = 0.35 \times 38.5 + 0.31 \times 51.4 + 0.23 \times 56.0 + 0.11 \times 87.3 \]
\[ = 51.9 \]  

(5)

Under the same condition, we use the gradient method to compute the optimal periods for the RM algorithm as \( T_1^R = 4.9, \ T_2^R = 14.6, \ T_3^R = 14.6, \ T_4^R = 14.6, \) and \( T_5^R = 29.2 \) and its system cost is about 61.9, which is higher than that of the QLB algorithm. It has to be noted that there is only one set of periods for the RM algorithm and no period transformations are performed.

Figure 3 shows the average system costs for the real-time mission in Example 3 scheduled by the QLB algorithm and the RM algorithm under different values of \( \lambda. \) The optimal periods for the RM algorithms are derived under the load condition \( \lambda = 0.1 \) and then used for all \( \lambda \) ranging from 0 to 0.2. Similarly, we use the periods obtained in Table 2 for the QLB algorithm to operate under different values of \( \lambda. \) The result reveals the robustness of these two algorithms because they have to operate in the environments where the arrivals of aperiodic tasks may be different from the original estimation or even not estimated. According to this result, the QLB algorithm is always better than the RM algorithm for different values of \( \lambda. \) The QLB algorithm has a higher probability to stay in Phase 0 to reduce the system cost by suspending the server when \( \lambda \) is small. The QLB algorithm is also better than the RM algorithm when
\( \lambda \) is higher than estimated since the service rate for aperiodic tasks in the QLB algorithm can be extended to \( \frac{c_u}{\pi_1} \approx 0.294 \) in Phase 3, which is higher than that of the RM algorithm. This explains why the cost of the RM algorithm increases dramatically when \( \lambda \) is close to \( \frac{c_u}{\pi_1} \approx 0.204 \) while that of the QLB algorithm does not.

5 Conclusion

In many real-time environments, the characteristics of tasks may change during the operation of computing systems. Most static scheduling algorithms can not provide efficient mechanisms for dynamic load adjustment such that they are not suitable for such applications. The dynamic period transformation algorithm proposed in [4] can adapt to the load changes as long as the arrivals of aperiodic tasks can be estimated. In this paper, we have proposed a queue-length-based (QLB) version of the dynamic period transformation algorithm which can be specifically applied in the environments where the stochastic characteristics of aperiodic tasks are not provided beforehand. The QLB algorithm defines a number of operational phases according to the queue length of aperiodic tasks, and it performs a period transformation and switches to another phase if the number of waiting aperiodic tasks does not match the legitimate queue length. Since the optimal periods in each phase are determined by the gradient method, the QLB algorithm can always provide the best results no matter how the arrivals of aperiodic tasks change. The simulation result shows that the QLB algorithm has very high robustness in the environments where the arrivals of aperiodic tasks are different from the original estimation or even not estimated.

Reference