

Optimizing Quality-of-Service in Adaptive Optics Systems and Other (m, k) -Firm Real-Time Spacecraft Control Systems

by Albert M. K. Cheng

ADAPTIVE OPTICS systems⁸ used in surveillance satellites/spacecraft and astronomical observatories provide high-resolution imaging but have not yet achieved their full potential. In these applications, light from an astronomical object such as a star, and, similarly, a ground object such as a car, is distorted by Earth's atmosphere. An adaptive optics system employs a high-speed control computer, with an array of digital signal processors, to periodically adjust a deformable lens or mirror based on the periodic input from a wave front sensor. The control task must calculate, a thousand times per second (i.e., period = 1ms, deadline < 1ms), how much and where to deform the lens or mirror to compensate for atmospheric distortion (i.e., wave front error). The corrected light is then passed through another lens, focusing it into a high-resolution image. It is clear that in this

application, QoS stability over time is critical since increasing the reward later following a low-reward interval is definitely not desirable. However, current adaptive optics systems, as well as many other control system applications, have not fully addressed this issue. This project, therefore, aims to optimize this QoS stability by maximizing the guaranteed performance while maintaining a schedulable task set in real-time systems such as adaptive optics systems.

Traditionally, many real-time space system applications are sets of periodic tasks in which each task of a set executes at a fixed rate and has a specific deadline. They are called "hard real-time systems" since violating task deadlines can be catastrophic. However, in firm real-time systems such as an adaptive optics system, occasionally missing task deadlines is tolerable. A number of models have been advocated for such systems. Examples are the skip-over model,² the window-constrained model,³ and



Albert M. K. Cheng

ABSTRACT—Adaptive optics systems used in surveillance satellites/air/spacecraft and astronomical observatories provide high-resolution imaging but have not yet achieved their full potential. In these applications, light from an astronomical object such as a star and, similarly, a ground object such as a car, is distorted by Earth's atmosphere. An adaptive optics system employs a high-speed control computer with an array of digital signal processors to periodically adjust a deformable lens or mirror based on the periodic input from a wave front sensor. The control task must calculate, a thousand times per second, how much and where to deform the lens or mirror to compensate for atmospheric distortion (i.e., wave front error). The corrected light is then passed through another lens, focusing it into a high-resolution image. It is clear that the Quality-of-Service (QoS) stability over time is critical since increasing the reward later following a low-reward interval is

definitely not desirable. However, current adaptive optics systems, as well as many other control system applications, have not fully addressed this issue. This project aims to optimize this QoS stability by maximizing the guaranteed performance while maintaining a schedulable task set in periodic firm real-time systems such as adaptive optics systems. We have introduced the problem of maximizing the guaranteed quality of service while maintaining the schedulability of a task set in an (m,k) -firm real-time system. Our experimental study has evaluated the solution using the Granularity of Quality of Service Rewards, effective processor utilization, total accumulated reward, and instability as performance measures. We like the simplicity of the heuristics and the good performance in regard to minimizing the instability in an overloaded system.

the (m,k) -constrained model.¹ In the (m,k) -constrained model, m out of any k consecutive jobs' deadlines must be met. This model is more realistic for capturing the temporal requirements of soft/firm periodic tasks thanks to the incorporation of granularities of quality. It is also the generalization of classic hard and firm periodic real-time tasks. For instance, $(1,1)$ -constrained represents the hard real-time requirement, and $(2,3)$ -constrained designates that, at most, one deadline can be missed in any three consecutive job releases. The (m,k) -constrained model also can increase the schedulability of tasks set under overloaded conditions. Since it does not have to require all jobs meeting their deadlines, the system may be schedulable under this constraint while otherwise it is not. In most applications using (m,k) constraints, the notion of (m,k) represents a quality of service metric. The more m out of any k consecutive jobs meeting their deadlines, the better the quality produced (more stable) by the

application/task. This project focuses on this metric in a stable-QoS-preferable real-time system. In the adaptive optics system, this would mean higher-resolution long-exposure imaging.

To exploit the (m,k) -constrained model, several static and dynamic techniques have been proposed. For dynamic methods, Hamdaoui and Ramanathan¹ present a best-effort scheduling algorithm which raises the priority of jobs that are going to violate their (m,k) -constraints. However, it cannot guarantee that these constraints are satisfied.

For static methods, to ensure the deterministic firm real time requirements, a specific pattern is used to partition jobs in each task as mandatory and optional. Mandatory jobs have their own original priority, and all optional jobs are assigned the lowest priority. The pattern is repeated within an infinite sequence of jobs. Although static methods can use the pattern to perform a schedulability test for the task set and then ensure the minimum required granularity levels of QoS, there is a gap between this lower bound and the total maximum across the system. This gap occurs because (m,k) -constraints are usually designated by system designers. In the design stage, they only care about the acceptable levels of application performance and actually do not take the issues of task scheduling into their considerations. It is unsatisfactory that the computation resource (*e.g.*, CPU time) is used up just for the mandatory jobs (guaranteed granularity of QoS) without looking into the schedulability stage. Although optional jobs can be run when no mandatory job executes or waits, the guaranteed QoS is hardly maximized because of the lack of investigation about the interactions among the (m,k) constraints, task schedulability, and performance.

To make use of the surplus computing time once the original pre-defined (m,k) constraints have been satisfied, our proposed strategy aims to complete additional mandatory jobs (thus yielding finer granularity) rather than to run optional jobs. The adaptive optics system would benefit from this strategy, which ensures that the variance of the reward is small over time, achieving a stable QoS. The more stable the QoS, the higher the benefit. Furthermore, if cost is a factor, then our strategy of extracting a higher QoS in an overloaded task system will produce the same QoS provided by today's systems but at a fraction of the current cost since lower-performance computer systems can then be used.

Methodology

Several studies relate task scheduling and system performance. Periods' selection is discussed^{5,6} as an off-line design aid for choosing the optimal frequency combination of classic periodic task sets. This project considers the problem of optimizing the trade-off between system-wide guaranteed QoS and task set schedulability off-line for the (m,k) -constrained model on a single-processor system and then on a multi-processor system. Each task T_i associates with a class of finite levels of QoS (*e.g.*, (m, k) or $(m+1, k)$ or...or (k, k)) and each level in the class has a specific reward value. We call this reward Granularity of Quality of Service reward (GQoS-reward). All of the reward values are assigned according to the relative importance of the guaranteed GQoS among all tasks within the system.

A task set of n independent periodic tasks is represented as

$T = \{T_1, \dots, T_n\}$. Each instance of a task is called a job. Each task T_i is characterized by a 5-tuple $(P_i, C_i, m_i, k_i, R_i)$ where P_i is the period of the task, C_i is the worst-case execution time (WCET), m_i and k_i ($0 \leq m_i \leq k_i$) define the (m,k) constraint for T_i , and R_i is a reward vector of size $k_i - m_i + 1$. Each element in the vector contains one version of GQoS-reward, *e.g.*, the reward for the (m_i, k_i) level, the reward for the $(m_i + 1, k_i)$ level, and so on. Without loss of generality, we define the values in R_i in the sense of monotonic increasing with their index. In other words, $r_{i(0)}$ is the smallest reward (it is 0 if the original m_i is 0) for guaranteeing the (m_i, k_i) -firm deadline, while the last element of the reward vector is the largest for satisfying a hard deadline, (k_i, k_i) , or $(1,1)$ equivalently. Before running on the system, exactly one version of GQoS for each task is to be determined. We also assume that the relative deadline of each task is the same as its period and all tasks are in phase in the preliminary experiments in order to study the feasibility of the proposed approach.

Currently, there are two main partitioning strategies to categorize jobs in a sequence. The first is the deeply-red pattern³ proposed by Koren et al.: a job T_{ij} , *i.e.*, the j -th instance of task T_i , is considered mandatory if $1 \leq \text{mod } k_i \leq m_i$ (for $m_i < k_i$. If $m_i = k_i$, T_i is a hard real-time task. If $m_i = 0$, all jobs of T_i are optional); otherwise, it is optional. The second strategy is the evenly-distributed-mandatory pattern⁴ proposed by Ramanathan et al. According to this pattern, T_{ij} is mandatory if and only if

$$j = \left\lceil \left[\frac{(j-1) * m_i}{k_i} * \frac{k_i}{m_i} \right] + 1 \right\rceil$$

($m_i > 0$. If $m_i = 0$, all jobs of T_i are optional similarly); otherwise, it is optional. Quan et al. improve the result of overload management with enhanced fixed-priority scheduling.^{4,7} We use the evenly-distributed-mandatory pattern in our preliminary experiments since it has been reported that this pattern exhibits relatively good schedulability.^{4,7}

Regarding the scheduling method, we adopt the earliest-deadline-first (EDF) algorithm because it is expected to provide good schedulability. In mandatory/optional job scheduling, the mandatory jobs of each task are assigned the priority according to the deadline-monotonic (DM) or EDF algorithm, and all optional jobs of every task are assigned the lowest priority. If two or more optional jobs compete for the processor when no mandatory jobs are eligible to run, the one with the earliest deadline executes first. If they happen to have the same deadline, any arbitrary priority assignment can be used to break the ties. West and Poellabauer³ present a necessary and sufficient condition for the deeply-red task set to determine schedulability. They claim that the worst case occurs on the first job of each task. We can use the time-demand analysis method proposed by Lehoczky⁹ to calculate the response time for each task. Please note that only mandatory workloads are incurred in the calculations.

Quan et al.¹⁷ present a sufficient and necessary condition for the evenly-distributed-mandatory pattern by using the EDF algorithm to schedule the jobs. Let R be the set of mandatory jobs. R is (m,k) -firm schedulable with EDF if and only if all the mandatory jobs within the first busy interval $[0, t]$ can meet their

deadlines. The end point of the first busy interval can be easily found by observing that it must be the smallest t such that the accumulated mandatory workload within interval $[0, t]$ equals t , that is,

$$\sum_i W_i(t) = \sum_i \left\lceil \frac{m_i}{k_i} \left\lceil \frac{t}{P_i} \right\rceil \right\rceil C_i = t$$

One can easily compute the end point for the first busy interval by using fixed-point iteration on this formula with the initial value of t set to a small number larger than 0, *e.g.*, 1. The fixed-point iteration will converge rapidly as long as the task set is schedulable. We can always set the upper bound of the busy interval length to be the least common multiple (LCM) of the tasks' periods. If the busy interval length is longer than the LCM of the tasks' periods, the task set is not schedulable.

System Model and Objectives

Here is an example showing that carefully selecting the (m, k) constraint, hence the guaranteed granularity of QoS for each task, is important to the system's overall performance. Consider a task set of three tasks: $T_1 = (10, 2, 1, 2, [10, 30])$, $T_2 = (15, 6, 1, 2, [20, 50])$, $T_3 = (60, 30, 1, 1, [50])$.

Since the utilization of the task set is 1.10, this task set is not schedulable without using (m, k) constraints. With the help of (m, k) constraints, the schedule of meeting all mandatory deadlines of the jobs is shown in Fig. 1(a). Note that in this case, using either the deeply-red pattern or evenly-distributed pattern produces the same result. It is not difficult to see that all tasks in Fig. 1(a) only attain their minimum GQoS-reward, and the total reward of the system for this case is $10+20+50 = 80$. However, with a more in-depth analysis, the total GQoS-reward can be increased by adjusting the (m, k) constraints used at run-time. In fact, for this example, we can increase either m_1 to 2 or m_2 to 2. In both cases, they all raise the total GQoS-reward while satisfying the new (m, k) -related rules. Figure 1(b) shows the schedule of giving m_2 one increment and the schedule in Fig. 1(c) has m_1 increased by 1.

In addition to improving the overall granularity of QoS, several interesting phenomena can be observed in the example. First, one idle interval (from 56 to 60) is left within the schedule in Fig. 1(a). The effective processor utilization (EPU) of this schedule is the lowest among the three schedules because several optional job(s) which, in fact, can be executed as mandatory job(s) have to be blocked by other mandatory jobs, while there are some idle intervals after executing the leftover optional jobs.

Another phenomenon is well known although it is not reflected in our example: EDF performs very poorly under overload.¹⁶ Second, note that the output produced from the schedule in Fig. 1(a) is non-stable. However, after increasing m of k for some task, EPU and output stability are all enhanced for the schedules in Fig. 1(b) and Fig. 1(c). As a result of these observations, we expect that our idea has positive effects on these soft/firm real-time performance metrics. The rationale behind our conjecture is that since we reserve as much CPU time as possible for important mandatory jobs, the space for the source of the negative factor, overloaded optional jobs, is compressed.

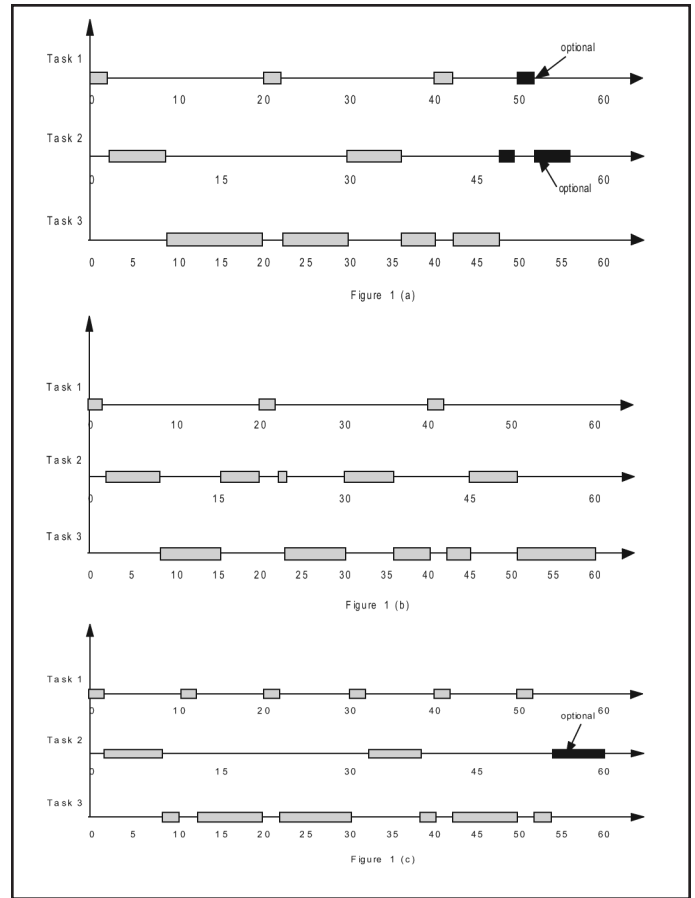


Figure 1. (a) Schedule obtained by satisfying the original (m, k) constraints, that is, $(m_1, k_1) = (1, 2)$, $(m_2, k_2) = (1, 2)$, $(m_3, k_3) = (1, 1)$. (b) Schedule obtained by increasing m_2 by 1, $(m_2, k_2) = (2, 2)$ or $(1, 1)$. (c) Schedule obtained by increasing m_1 by 1, $(m_1, k_1) = (2, 2)$ or $(1, 1)$.

Hence, the maximum effect of the possible degradation is minimized, and consequently, it is very likely that the soft/firm real-time performance can be improved.

According to the relative importance among all tasks in the task set, each task T_i has a multi-version GQoS rewards. The number of versions in task T_i is dependent on the (m_i, k_i) parameter and then it equals $k_i - m_i + 1$. Our goal is to determine the version of GQoS for each task such that the total GQoS reward is maximized while satisfying the corresponding (m, k) -firm deadline requirements. Formally, the problem can be defined as:

$$\text{Maximize } \sum_{i=1}^n r_i \quad \text{subject to:}$$

- (1) the task set is schedulable by the re-defined (m, k) -firm constraints: $\{(m_1, k_1) \dots (m_n, k_n)\}$ corresponding to $\{r_1 \dots r_n\}$, and
- (2) $r_i \in \{r_{i(0)} \dots r_{i(k_i - m_i)}\}$.

Complexity of the GQoS-Reward Optimization Problem

By using patterns where the first job of every task is mandatory and has the worst-case execution time, we can use Lehoczky's time-demand analysis method⁹ to check if the task set is schedulable for those given (m, k) -patterns. However, the

problem of selecting the best version of GQoS for each task to ensure that the total reward is maximized and all mandatory jobs are schedulable is an NP-hard problem. This can be shown by using a reduction from the 0-1 constrained knapsack maximization problem,¹¹ which is defined as follows: Maximize $p_i x_i$ subject to $w_i x_i \leq M$, where $x_i \in \{0, 1\}$.

Given an instance of the 0-1 constrained knapsack problem $P = \{p_1, p_2, \dots, p_n\}$, $W = \{w_1, w_2, \dots, w_n\}$, we construct an instance of the GQoS-reward maximization problem: each task T_i has a worst-case execution time $e_i = w_i$. The period and relative deadline of each task are all equal to M . The number of granularities of QoS for each task is 2, represented as $(m_i, k_i) = (0,1)$ and $(1,1)$. The reward for the case of $(0,1)$ is trivial, which is 0. The reward for the case of $(1, 1)$ for task T_i is p_i . Therefore, if $x_i = 1$, $p_i x_i$ means the GQoS-reward of selecting the $(1, 1)$ -version of granularity of QoS for T_i . Otherwise, $(0, 1)$ is selected instead. It is not difficult to see that $p_i x_i$ is maximized in the 0-1 constrained knapsack problem if and only if the total GQoS-reward is maximized in the GQoS-reward maximization problem. Since the transformation between the given instance of 0-1 constrained knapsack problem and the instance of GQoS-reward maximization problem can be performed in polynomial time, we thus know that the GQoS-reward maximization problem is also NP-hard.

Results

If the number of tasks in a task set and the number of versions of each task are not large, an exhaustive search method similar to that in Seto, Lehoczky *et al.* can be used.⁶ First, to guarantee that each task T_i is schedulable, a set of feasible (m_j, k_j) parameters, where $P_j P_i$ is determined, *e.g.*, in the motivational example, the feasible set for task T_3 is $\{(m_1, k_1)-(m_2, k_2)-(m_3, k_3) \mid (1, 2)-(1, 2)-(1, 1), (1, 2)-(2, 2)-(1, 1), (2, 2)-(1, 2)-(1, 1)\}$. Next, we intersect the sets of feasible granularities. After this intersection, we have all feasible combinations of (m, k) parameters in the task set which can be used at run-time. Finally, we traverse these combinations to determine the optimal selection which maximizes the total GQoS-reward. Although we can use the Branch-and-Bound strategy to ease the work slightly, the number of possible combinations of GQoS increases exponentially with the number of tasks or the total number of granularity versions in all tasks. If the task set is very large, it is impossible to perform this calculation within a reasonable length of time. Thus, we have to look for some other sub-optimal solutions to solve the problem efficiently.

Our idea is to define a heuristic method based on sorting all versions of GQoS by a metric to be determined. The metric should reflect a relatively large reward and has a small negative effect on schedulability. Hence, we can solve the problem sub-optimally in polynomial time (the time for sorting plus the time for a linear search). Selecting a good heuristic is very important to the final results. A good choice is prone to a larger value-density task¹⁰ which focuses on values and execution times of the task. However, for our problem, the effects of the (m,k) parameters and task periods also need to be considered. For example, increasing different m_i by one increment has different effects on task schedulability. This project will design the heuristics and

Algorithm 1: Select the best (m,k) parameter for each task while keeping the task set schedulable.

1. **Input:** Periodic task set T with tasks having firm deadlines and GQoS-rewards assigned properly.
2. **Output:** Version of QoS represented by the (m,k) parameter used at runtime for each task to maximize the total GQoS-reward.
3. Sort all the versions according to their non-increasing values of

$$r_{ij} / \left(\frac{m_{ij} \cdot C_i}{k_i \cdot P_i} \right), \quad i = 1, \dots, n, j = 0, \dots, k_i - m_i.$$
4. $U =$ set containing the sorted versions.
5. $V_i =$ minimum required version for task T_i . Remove the minimum versions from U .
6. **while** $U \neq \emptyset$ do
7. temp = the head of U
8. use temp to replace corresponding V_i to check the feasibility of the task set
9. **if** schedulable **then**
10. $V_i =$ temp
11. remove all versions not finer than V_i in U for T_i
12. **else**
13. remove the version at the head
14. **end if**
15. **end while**
16. **Return** $V_i, i = 1, \dots, n$

Figure 2. Algorithm for Selecting the Best (m,k) Parameter

perform experiments on a variety of control, multimedia, and networking applications to evaluate different heuristic functions. In our preliminary experiments, metrics which do not consider these parameters fail to give good approximations of the optimal total rewards. Hence, we design the versions of granularity so that they are ordered non-increasingly according to the values of

$i = 1, \dots, n, j = 0, \dots, k_i - m_i$, initially. We next present the

$$r_{ij} / \left(\frac{m_{ij} \cdot C_i}{k_i \cdot P_i} \right),$$

algorithm applying this principle for selecting the best (m,k) parameter.

It works like the greedy approach, always choosing the schedulable version of GQoS which has the largest reward to resource demand ratio. If one version for a task should be chosen, the coarser versions for that task are removed from the sorted list and only finer versions are left. By accomplishing this process, the “surplus” computing resource is reserved for relatively important tasks as much as possible.

Experimental Results and Evaluation

We use simulation experiments to demonstrate the effectiveness of our heuristic method to solve the GQoS-reward optimization problem, and discover the positive effects of maximizing gran-

ularities of QoS. In the first part, we simulate our heuristic algorithm on the same task sets whose sizes are relatively small with up to eight (8) tasks each. The task sets are divided evenly into nine (9) groups by their effective utilizations in the interval 0.1–1.0. Each group contains at least 100 randomly generated task sets.

$$\sum \frac{m_i * C_i}{k_i * P_i}$$

Then we compare our solution with the optimal solution that is obtained with an exhaustive search. Figure 3 shows the optimal total GQoS-reward and our approximated total GQoS-reward normalized to the initial total GQoS-reward (the granularities of QoS over the system with the original (m, k) parameters). One can see that our heuristic solutions are very close to the optimal results. Actually, in these preliminary experiments, our algorithm finds the optimal solution most of the time.

In the second part, we want to determine the relationship between the optimization of the granularities of QoS and the performance of a soft/firm real-time system. We adopt several popular soft/firm real-time system performance metrics from the literature such as EPU¹⁹ and instability.¹⁵ In the experiments, 500 task sets are randomly generated. The periods are randomly selected between 10 and 150, and the deadlines are assumed to be the same as their periods. The worst-case execution time (WCET) of a task is uniformly distributed from 1 to its deadline. Similar to the experiments in the first part, the task sets are divided into 9 groups according to their effective utilizations. To reduce statistical errors, we ensure that each group contains at least 20 schedulable task sets.

To measure EPU and instability, the task sets are simulated within the K-LCMs (least common multiple of $P_i * k_i$). EPU is obtained by calculating the fraction of time during the interval in which the processor executes tasks that complete by their deadlines. To calculate instabilities, the idea in Brandt (2001) is used, which sees the instability of a task as a discrete sum of changes on the number of the task completions between any two consecutive distinct intervals. To accommodate the requirements of the (m, k)-constrained model, we define the length of the intervals for a task to be the same as the parameter k for that task and the intervals are overlapped. We expect that our algorithm can improve the soft/firm real-time performance on both EPU and instability. Figure 4 shows the comparison results on EPU.

Compared with the results that are not optimized, our proposed technique has significant improvements when the effective utilization of the system is relatively low. As utilization increases, improvement decreases because the space for improvement becomes less available. However, it is very attractive that our strategy always keeps the average EPU around 95 percent. For instability, an amazingly huge improvement can be seen in Fig. 5. The average instability is maintained at a very low level; hence, this response supports the fact that simulated systems are very stable in QoS using our method.

Discussion

In order to discover the effects of our optimization strategy more thoroughly, we also investigate the comparisons on accumulated value (AV) although this metric is not very good

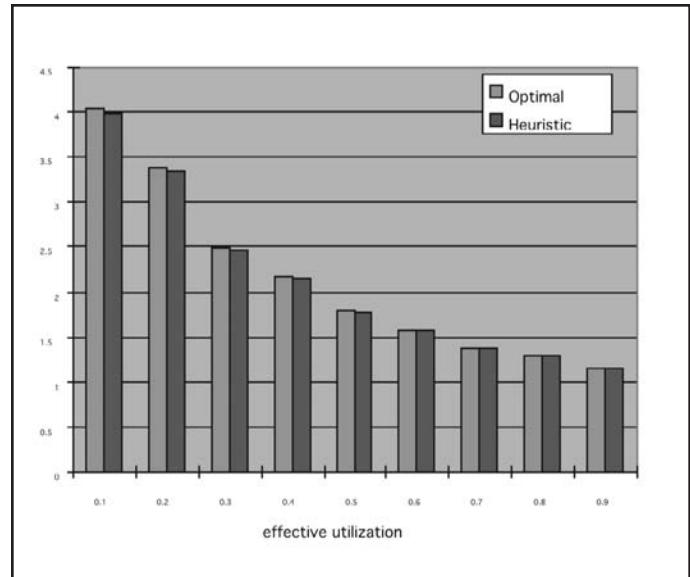


Figure 3. Total Reward Ratio

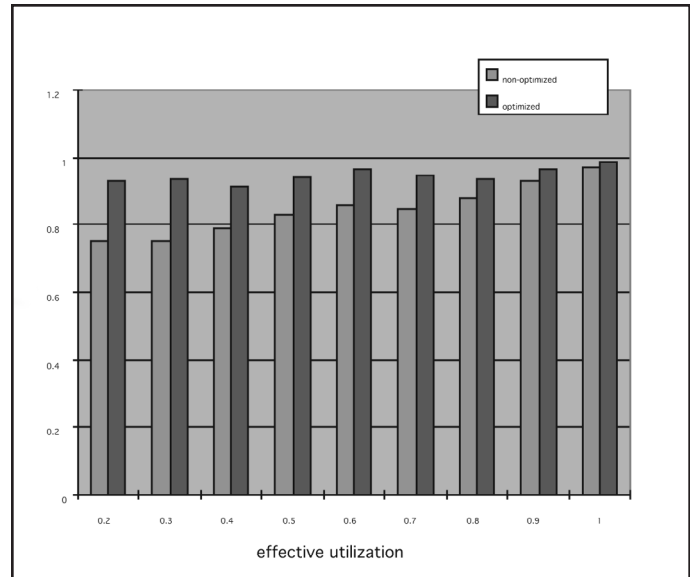


Figure 4. EPU Comparison

for the (m, k)-constrained model since it lacks enforcement on granularity. We assign each task in the experiment a value and assign the GQoS-rewards according to this value; we calculate the total accumulated value that is in the same sense of the total weighted completion count (CC). (See Baruha et al. and Ramanathan.) In our experiments, about 85 percent of the cases improve the results; consequently, the average case with our technique is slightly better than the case without using it (Table 1). The reason for the small improvement, on average, is that after we increase m of k for some relatively important tasks, we, in fact, increase the priority of some jobs in the tasks by changing them into mandatory tasks. However, since those jobs could be run even if they were optional but block some other optional jobs after being re-categorized as mandatory, the system may lose accumulated rewards in total for

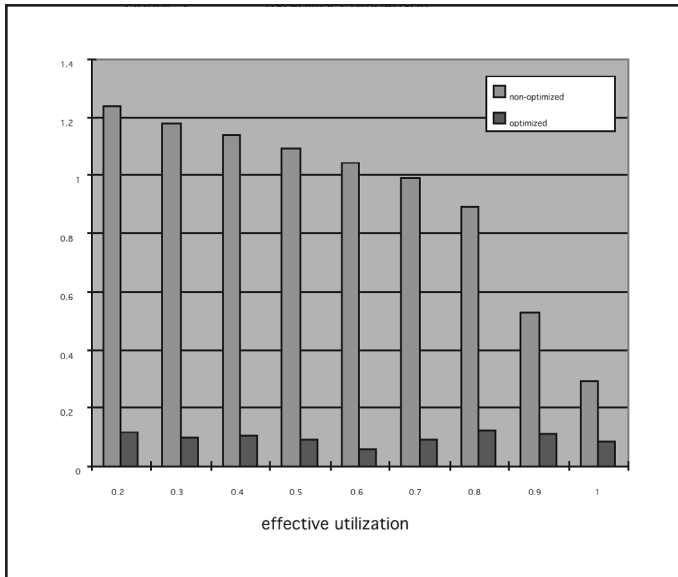


Figure 5. Instability Comparison

Effective Utilization	Average AV without our strategy	Average AV using our strategy
0.1 – 0.2	100	113
0.2 – 0.3	100	118
0.3 – 0.4	100	114
0.4 – 0.5	100	112
0.5 – 0.6	100	107
0.6 – 0.7	100	104
0.7 – 0.8	100	105
0.8 – 0.9	100	104
0.9 – 1.0	100	104

Table 1. Normalized Average Total Accumulated Values (AV)

some cases. Nonetheless, while considering this together with other performance perspectives, enhancement in the overall performance is apparent using our optimization strategy.

Conclusions

We have introduced the problem of maximizing the guaranteed quality of service while maintaining the schedulability of a task set in an (m,k) -firm real-time system. We proved that the problem is NP-hard and proposed a simple heuristic solution — by greedily increasing the QoS level of the tasks with the maximal “reward ratio” as long as all the other tasks have their minimum service level. Our preliminary study has evaluated the solution using the Granularity-of-Quality-of-Service Rewards, effective processor utilization, total accumulated reward, and instability as performance measures. We like the simplicity of the heuristics and the good performance with regard to its minimizing the instability in an overloaded system.

References

- ¹M. Hamdaoui and P. Ramanathan, “A Dynamic Priority Assignment Technique for Streams with (m,k) -Firm Deadlines,” *IEEE Trans. on Computers* 44 (1995): 1443-51.
- ²G. Koren and D. Shasha, “Skip-over: Algorithms and Complexity for Overloaded Systems That Allow Skips,” *IEEE Real-Time Systems Symposium [RTSS]* (1995): 110-17.
- ³R. West and C. Poellabauer, “Analysis of a Window-Constrained Scheduler for Real-Time and Best-Effort Packet Streams,” *Proc. IEEE RTSS* (2000): 239-48.
- ⁴P. Ramanathan, “Overload Management in Real-Time Control Applications Using (m,k) -Firm Guarantee,” *IEEE Trans. on Parallel and Distributed Systems* 10.6 (1999): 549-59.
- ⁵D. B. Seto, J. P. Lehoczky, L. Sha, and K. G. Shin, “On Task Schedulability in Real-Time Control Systems,” *IEEE RTSS* (1996): 13-21.
- ⁶D. B. Seto, J. P. Lehoczky, and L. Sha, “Task Period Selection and Schedulability in Real-Time Systems,” *IEEE RTSS* (1998): 188-98.
- ⁷Q. Quan and X. Hu, “Enhanced Fixed-Priority Scheduling with (m,k) -Firm Guarantee,” *IEEE RTSS* (2000): 79-88.
- ⁸M. Lloyd-Hart, “Taking the Twinkle out of Starlight,” *IEEE Spectrum* (2003): 22-29.
- ⁹Lehoczky, J. P., L. Sha, and Y. Ding, “The Rate-Monotonic Scheduling Algorithm: Exact Characterization and Average Case Behavior,” *IEEE RTSS* (1989): 166-71.
- ¹⁰G. Buttazzo, M. Spuri, and F. Sensini, “Value vs. Deadline Scheduling in Overload Conditions,” *IEEE RTSS* (1995): 90-99.
- ¹¹M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. New York: Freeman, 1979.
- ¹²A. M. K. Cheng, *Real-Time Systems: Scheduling, Analysis and Verification*. Wiley-Interscience, 2002.
- ¹³S. K. Baruah and Jayant R. Haritsa, “Scheduling for Overload in Real-Time Systems,” *IEEE Trans. on Computers* 46 (1997): 1034-1039.
- ¹⁴S. Baruah, J. Haritsa, and N. Sharma, “On-Line Scheduling to Maximize Task Completions,” *IEEE RTSS* (1994): 228-37.
- ¹⁵S. A. Brandt, “Performance Analysis of Dynamic Soft Real-Time Systems,” *IEEE IPCCC* (2001): 379-86.
- ¹⁶J. Haritsa, M. Carey, and M. Livny. “Earliest-Deadline Scheduling for Real-Time Database Systems,” *IEEE RTSS* (1991): 232-43.
- ¹⁷G. Quan, L. Niu and J. P. Davis, “Power Aware Scheduling for Real-Time Systems with (m,k) Guarantee,” *CNDS* (2004).
- ¹⁸J. Lin, A. M. K. Cheng, “Maximizing Guaranteed QoS in (m, k) -firm Real-time Systems,” *12th IEEE Intl. Conf. on Embedded and Real-Time Computing Systems and Applications* (2006): 402-10.
- ¹⁹P. Ramanathan, “Overload Management in Real-Time Control Applications Using (m,k) -Firm Guarantee,” *IEEE Trans. on Parallel and Distributed Systems*, 10.6 (1999): 549-59.
- ²⁰Bernat, G. and R. Cayssials, “Guaranteed on-Line Weakly-Hard Real-Time Systems,” *Proc. IEEE 22nd Real-Time Systems Symposium* (2001): 25-35.
- ²¹D. Liu, X. S. Hu, M. D. Lemmon, and Q. Ling, “Firm Real-Time

System Scheduling Based on a Novel QoS Constraint,” *IEEE Trans. on Computers*, 55.3 (2006): 320-33

Publications

Andrei, S. and A. M. K. Cheng. “Faster Verification of RTL-Specified Systems via Decomposition and Constraint Extension,” *Proc. IEEE-CS Real-Time Systems Symposium (RTSS)*, Rio de Janeiro, Brazil, December 2006.

Andrei, S. and A. M. K. Cheng. “Optimization of Real-Time Systems Timing Specifications,” *Proc. 12th IEEE-CS Intl. Conf. on Embedded and Real-Time Computing Systems and Applications*, Sydney, Australia, 2006.

Andrei, S., W.-N. Chin, A. M. K. Cheng, and M. Lupu. “Automatic Debugging of Real-Time Systems Based on Incremental Satisfiability Counting,” *IEEE Transactions on Computers* 55.7 (2006): 830-43. (Accepted February 2006, selected as this issue’s featured article.)

Aruchamy, G. and A. M. K. Cheng. “Translating Real-Time UML Timing Constraints into Real-Time Logic Formulas,” *IEEE-CS Real-Time Systems Symposium (RTSS) WIP Session*, Rio de Janeiro, Brazil, December 2006.

Cheng, A. M. K. “Intrusion Detection via Automatic Rule-Base Generation and Semantic Analysis,” *Proc. SCISS*, Houston, TX, April 2006.

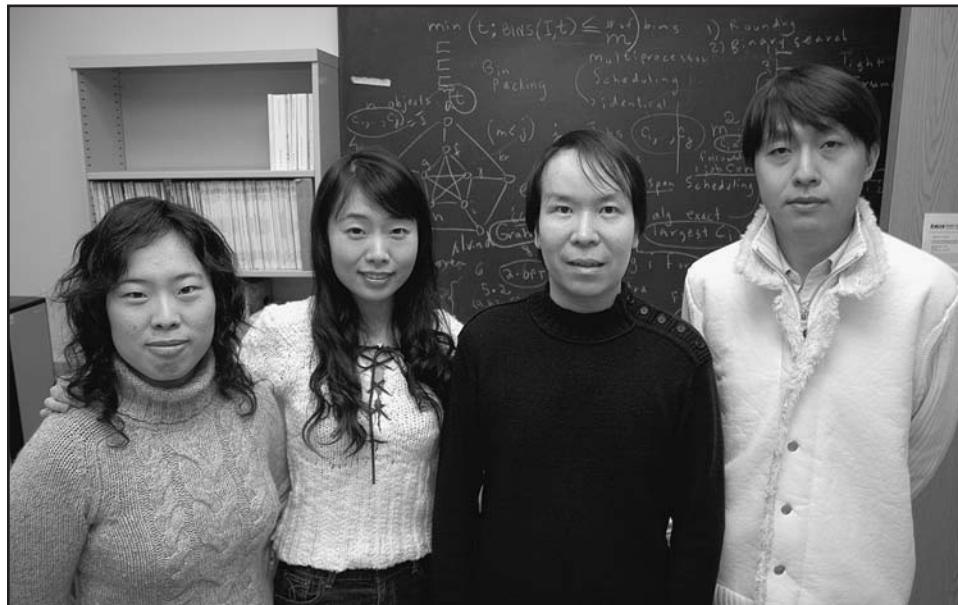
Cheng, A. M. K. “On-Time and Scalable Intrusion Detection in Embedded Systems,” *Proc. Workshop on Research Directions for Security and Networking in Critical Real-Time and Embedded Systems*, in conjunction with IEEE-CS Real-Time and Embedded Technology and Applications Symposium (RTAS), San Jose, CA, April 2006.

Cheng, A. M. K. and F. Shang. “Priority-Driven Coding and Transmission of Progressive JPEG Images for Real-Time Applications.” (Accepted May 2006; forthcoming in *J of VLSI Signal Processing - Systems for Signal, Image and Video Technology*, 2007. (Accepted May 2006.)

Cheng, A. M. K. and Z. Zhang. “Improving Web Server Performance with Adaptive Proxy Caching in Soft Real-Time Mobile Applications,” (Accepted, Dec. 2006; forthcoming in *J. of VLSI Signal Processing Systems for Signal, Image, and Video Technology*, 2007.)

Cheng, A. M. K. and Z. Zhang. “Improving Web Server Performance in Soft Real-Time Mobile Applications with Adaptive Proxy Caching,” *Proc. MobEA IV-Empowering the Mobile Web*, Collocated with ACM WWW Conference, Edinburgh, Scotland, May 2006.

Lewis, Q. S. and A. M. K. Cheng, “3D GC: Towards a Garbage



REAL-TIME TEAM—(l. to r.) Computer scientists Bin Lu, master’s student, Yingwei Kuo, Ph.D. candidate, Dr. Albert M. K. Cheng and Jian Lin, Ph.D. candidate, focused their study on optimizing real-time task processing and minimizing instability in overloaded computer systems.

Collector that Considers Time, Space, and Energy,” *IEEE-CS Real-Time Systems Symposium (RTSS) WIP Session*, Rio de Janeiro, Brazil, December 2006.

Lin, J. and A. M. K. Cheng. “Maximizing Guaranteed QoS in (m,k)-firm Real-time Systems,” *Proc. 12th IEEE-CS Intl. Conf. on Embedded and Real-Time Computing Systems and Applications*, Sydney, Australia, August 2006.

Lin, J. and Albert M. K. Cheng. “Maximizing Guaranteed QoS within (m,k)-Firm Real-time Constraints,” *Proc. IEEE-CS Real-Time and Embedded Technology and Applications Symposium (RTAS) WIP Session*, San Jose, CA, April 2006.

Zhang, W., A. M. K. Cheng, B. Fang, and M. Hu. “An Adaptive Multisite Scheduling Algorithm for Parallel Jobs in Computational Grid Environments,” *Proc. Third High-Performance Grid Computing Workshop*, in conjunction with International Parallel and Distributed Processing Symposium, Rhodes Island, Greece, April 29, 2006.

Presentations

Cheng, A. M. K., Tutorial Speaker, “Parallel and Distributed Embedded/Real-Time Systems,” *The 18th IASTED Intl. Conf. on Parallel and Distributed Computing and Systems (PDCS)*, Dallas, TX, Nov. 14, 2006.

Cheng, A. M. K., Tutorial Speaker, “Analysis and Verification of Real-Time Embedded Software and Systems,” *Formal Methods Symposium*, McMaster University, Hamilton, Canada, Aug. 21–27, 2006.

Proposals

External, NSF, “Optimizing Quality of Service in Firm Real-Time Systems,” \$325,000, June 1, 2007–May 31, 2010. (Submitted).