Dynamic Task Scheduling with Security Awareness in Real-Time Systems

Tao Xie, Xiao Qin, Andrew Sung
Department of Computer Science,
New Mexico Institute of Mining and Technology, Socorro, USA
E-mail: {xietao, xqin, sung@cs.nmt.edu}

Lin Man, Laurence Yang
Department of Computer Science
St. Francis Xavier University
Antigonish, NS, B2G 2W5, Canada
{mlin, lyang@stfx.ca}

Abstract: An increasing number of real-time applications like aircraft control and medical electronics systems require high quality of security to assure confidentiality, authenticity and integrity of information. However, most existing algorithms for scheduling independent tasks in real-time systems do not adequately consider security requirements of real-time tasks. In recognition of this problem we propose a novel dynamic scheduling algorithm with security awareness, which is capable of achieving high security for real-time tasks while improving resource utilization. We conducted extensive simulation experiments to quantitatively evaluate the performance of our approach. Experimental results based on synthetic and real world traces show that compared with three baseline algorithms, the proposed algorithm can consistently improve overall system performance in terms of quality of security and guarantee ratio under a wide range of workload characteristics.

Keywords: security-aware scheduling; real-time; overall system performance; security value; guarantee ratio.

Reference to this paper should be made as follows: Xie, T.; Qin, X.; Sung, A.; Lin, M.; and Yang, L. (2006) 'Dynamic Task Scheduling with Security Awareness in Real-Time Systems', *Int. J. High Performance Computing and Networking*, Vol. 1, Nos. 1/2/3, pp.43–54.

Biographical notes: T. Xie is currently a Ph.D. Candidate at the Department of Computer Science, New Mexico Institute of Mining and Technology, New Mexico, USA. His current research interests include High Performance Computing, Cluster and Grid Computing, Distributed Systems, Parallel Processing, Real-time/Embedded Systems, Information Security, and Performance Evaluation

X. Qin is an Assistant Professor of Computer Science at the New Mexico Institute of Mining and Technology, New Mexico, USA. He received the Ph.D. degree in Computer Science from the University of Nebraska-Lincoln. His current research focuses on parallel and distributed systems, storage systems, real-time computing, performance evaluation, and fault-tolerance.

A. Sung is currently Professor and Chairman of the Computer Science Department of New Mexico Institute of Mining and Technology. He received his Ph.D. in Computer Science from the State University of New York at Stony Brook in 1984. His current research interests are computational intelligence, soft computing, and information security.

1 INTRODUCTION

Various dynamic real-time scheduling algorithms like Earliest Deadline First (EDF) (Liu and Layland, 1973) and

Spring scheduling algorithm (Ramamritham and Stankovic, 1984; Zhao et al, 1987) been developed for systems that have no complete knowledge of task sets or timing constraints. For example, new created tasks may arrive at a unknown time. Although existing dynamic real-time scheduling algorithms are effective in enhancing the performance of real-time systems, security requirements posed by scheduled tasks have not been factored in. This critical problem becomes more pronounced when all the scheduled real-time tasks require a certain level of security guarantee provided by the system. Security requirement bearing with real-time tasks is an extremely critical issue and, therefore, have to be taken into account by dynamic real-time scheduling algorithms. To tackle this important problem, we proposed four security-aware dynamic real-time scheduling algorithms, namely, EDF_MINS, EDF_MAXS, EDF_RNDS, and EDF_OPTS. The first three algorithms are variants of the earliest deadline first (EDF) scheduling algorithms with security awareness. These three algorithms intentionally choose minimal, maximal, and random security level specified by each incoming tasks, respectively. Unlike these three algorithms, the EDF_OPTS algorithm is an optimized security-aware EDF scheduling algorithm. EDF_OPTS is able to improve the quality of security of tasks admitted to a real-time system while maintaining a reasonably high level of guarantee ratio defined as the ratio of the number of tasks guaranteed to meet their specified deadlines to the total number of submitted tasks. To the best of our knowledge, our algorithms are the first algorithms for scheduling dynamic tasks with security requirements in a real-time environment. The basic idea behind our algorithms is to incorporate a security-aware scheme into the Earliest Deadline First algorithm, or EDF, to construct the four security-aware EDF scheduling algorithms. A compelling advantage of our approach is that real-time systems with high-security demands can make use of the proposed algorithms to flexibly provide the most appropriate level of security for each task arrived in the systems. Most existing dynamic scheduling algorithms for real-time systems are not security-aware and, thus, unable to be employed in security demanding real-time environments.

In a real-time system with high security demands like a real-time stock quote update and trading system (Donoho, 2004), each incoming request or task submitted from a business partner and each outgoing response from an enterprise's back-end application (the terms request and task are used interchangeably throughout this paper) has a deadline and a range of security level requirements which must be met by the server located between the business partners and enterprise back-end applications. In this case, the server performs security operations on behalf of all its clients. Each task submitted by a large group of clients explicitly specifies a range of security levels in addition to its deadline. The server facilitates required security mechanisms on top of a request or a response in an out-of-the-box integration manner. In general the server system performs the following security operations on behalf of its clients (VeriSign, 2003):

- Establishes secure connections with business partners and back-end applications
- Applies and verifies digital signatures
- Authorizes access based on digital certificates
- Validates credentials in real time using public key infrastructure
- Encrypts and decrypts requests and responses

Furthermore, the server can judiciously select a suitable security level from the range of security levels specified by each task. A security level is a predefined combination of transport and message security mechanisms. Some typical security levels in a real-time quote and trading systems are (VeriSign, 2003):

- Routing only
- Routing + message security
- Routing + SSL
- Routing + SSL + message security
- Routing + SSL + client authentication
- Routing + SSL + message security + client authentication

Different security levels impose different extra system overheads including CPU and memory usage. While a real-time system can automatically provide high quality of security for some tasks by increasing security levels at the cost of high overheads, the system can intentionally reduce the quality of security for other tasks to improve guarantee ratios. In our scheduling model, each task has a range of security requirement levels denoted by $[SL_{min}, SL_{max}]$. The goal for our security-aware scheduling algorithms for real-time tasks is to enable real-time systems to meet the deadlines of a large fraction of submitted tasks while providing a high level of system security. Unfortunately, current state-of-the-art real-time scheduling algorithms are not security-aware and thus cannot be directly deployed to security-sensitive computing environments. In addition, some straightforward security-aware real-time scheduling algorithms, including EDF_MINS, EDF_MAXS, and EDF_RNDS can not achieve the goal either. The EDF_MINS algorithm is intended to choose the minimal security level from the range of security levels specified by a task. Although EDF_MINS can inherently achieve a high guarantee ratio, it tends to present real-time tasks with the minimal level of security, making the lowest security performance of the real-time system. On the contrary, EDF_MAXS obtains the highest security performance by choosing the maximal security level for each admitted task. The disadvantage to the EDF_MAXS algorithm is that the likelihood of a task being rejected by the system is unreasonably high if the system relatively overloaded, resulting in a low guarantee ratio. The EDF_RNDS algorithm, an alternative to EDF_MINS and EDF_MAXS, randomly configures the security level for each real-time task admitted to the system. As a result, the performance of EDF_RNDS in terms of guarantee ratio and quality of security is in the range between that of EDF_MINS and EDF_MAXS.

The main contribution of this paper is to propose a novel security-aware real-time scheduling algorithm referred to as EDF_OPTS, which can be successfully applied to security demanding real-time systems such as real-time stock quote update and trading systems. The captivating characteristic of our scheme is to adaptively pick the most suitable security levels from tasks' security requirement ranges in a way to obtain high overall security performance while maximizing guarantee ratios. Experimental results show that the EDF_OPTS algorithm outperforms all three heuristic baseline security-aware EDF scheduling algorithms in terms of overall system performance in all cases. Furthermore, EDF_OPTS consistently improves the performance of EDF_MAXS and EDF_RNDS with respect to guarantee ratio and is only slightly inferior to EDF_MINS. In particular, EDF_OPTS achieves performance improvement in security over EDF_MINS by 94.41% with the marginal guarantee ratio decreasing (< 5%).

The rest of this paper is organized as follows. Related work is discussed in Section 2. Section 3 describes the design of our scheduling algorithm with security awareness. Section 4 evaluates the proposed algorithm. Section 5 concludes the paper with some comments on the future work.

2 RELATED WORK

Scheduling algorithms play an important role in achieving high performance for real-time systems. While a scheduling algorithm maps real-time tasks to processors in a system such that deadlines and response time requirements are met (Stankovic et al, 1998), the system has to guarantee its functional and timing correctness even in the presence of hardware and software faults (Qin and Jiang, 2001; Qin et al, 2002).

Many scheduling algorithms, which can be classified into two categories: static (Palencia et al, 1998; Abdelzaher and Shin, 1999; Kwok et al, 1996) and dynamic (Palis, 2002; Thomadakis and Liu, 1999; Kalogeraki et al, 2000; Manimaran et al, 1998), have been proposed in the literature to provide high performance for real-time systems. Palis proposed a task-scheduling algorithm that provides quality of service guarantees in the context of reservation-based real-time systems (Palis, 2002). While real-time tasks in Palis's scheduling framework are preemptive (Palis, 2002), it is assumed in our study that real-time tasks are non-preemptive.

Abdelzaher and Shin proposed a communication sub-system architecture for Quality of Service (QoS) adaptive real-time applications such as streaming stored video on end-hosts (Abdelzaher and Shin, 1998). They developed a dynamic QoS-optimization mechanism, where flexible QoS contracts specify multiple acceptable levels of service (or QoS levels for short) and their corresponding rewards for each client. The goal of their algorithm is to adjust each accepted task's QoS levels in a way that the system's total reward under resource constraints can be maximized. Our task model assumes that each task has multiple acceptable security levels, which is similar to the concept of QoS level introduced by Abdelzaher and Shin (Abdelzaher and Shin, 1998). However, their scheduling algorithms do not rely on QoS levels. More importantly, our study differs from that of Abdelzaher and Shin in that the goal of our algorithms is to maximize the overall quality of security (measured by security value) and guarantee ratio by dynamically changing each accepted tasks' security levels.

Azzedin and Maheswaran examined the integration of the notion of "trust" into resource management of a large-scale wide-area system like the Grid (Azzedin and Maheswaran, 2002). They argued that there is a "trust relationship" between a resource provider and a resource consumer. The hypothesis is that if a resource manager is aware of the security requirements of the resources and tasks, the resource manager can perform allocations such that security overhead can be minimized. It is worth noting that the allocation algorithm proposed by Azzedin and Maheswaran is developed in the context of non-real-time computing environments, and it is not suitable for real-time systems where tasks' deadlines have to be factored in (Azzedin and Maheswaran, 2002).

Previous research has applied control theory to dynamic real-time scheduling algorithms (Lu et al, 1999; Stankovic et al, 2001). The basic idea behind the feedback control EDF scheduling system is to construct a feedback loop where the system periodically compares controlled variables to a set point to determine errors and changes the values of the manipulated variables to control the system (Lu et al, 1999). Stankovic et al. successfully built a theoretical model to analyze the stability of distributed real-time systems, thereby demonstrating that their scheduling algorithm with feedback control is stable and superior to other algorithms. Furthermore, Marbini and Sacks proposed a closed-loop dynamic scheduling algorithm that performs better than its open-loop counterparts (Marbini and Sacks, 2002). Our scheduling approach differs from the above scheduling techniques in that ours is a security-aware scheduling scheme to which a Security Level controller is integrated.

3 SECURITY-AWARE REAL-TIME SCHEDULING

3.1 Architecture of EDF_OPTS

First, we introduce the architecture of the security-aware scheduling algorithms. Second, basic ideas behind the security-aware EDF scheduling algorithms are given. Lastly, we concentrate on the definitions of the performance metrics that we pursued.

The EDF_OPTS scheduling architecture is mainly composed of a security level controller, an admission controller (AC) and an EDF scheduler, as depicted in Figure 1. This architecture is applied to one server (or CPU) node, meaning that there is only one computer system handling all incoming real-time tasks and the EDF_OPTS scheduler is running on this node.

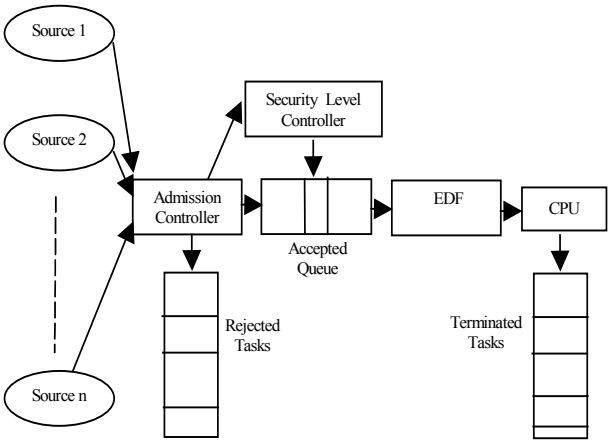

Figure 1 *System architecture*

All the tasks are independently submitted from n sources and their arrival times abide by Poisson distribution. The function of the admission controller is to determine if an incoming task can be accepted, or rejected otherwise. The security level controller is intended to maximize the security levels of current tasks residing in an accepted queue. The EDF scheduler makes use of the Earliest Deadline First policy to schedule admitted tasks, where security levels are optimized by the security level controller. The following steps depict the procedures of the EDF_OPTS scheduler where the security quality of a real-time system is improved:

Step One: Initialize the scheduler. Overall security value and number of rejected task are set to zero. Wait for any incoming tasks.

Step Two: If a task *i* arrives and it is the only task available, execute the task immediately using its highest security level. The starting time ST_i (actually ST_i is its arrival time in this case) and completion time CT_i of task *i* are calculated. The security value is increased by the security level of task *i*, which, in this case, is SL_{i_max}.

Step Three: All the tasks arriving in the system during the time period $[ST_i, CT_i]$ are stored into a waiting queue in the non-decreasing order of their deadlines. The starting time of the next task ST_{i+1} is set to CT_i.

Step Four: Admission controller is responsible of deciding whether a task in the waiting queue can be accepted by considering (1) the security overhead (extra execution time) imposed by the task's lowest security requirement, and (2) the deadline of this task. If the system can meet the task's deadline while fulfilling the security requirement, the task will be forwarded into the accepted queue for further processing. Otherwise, it will be rejected by being put into the rejected queue, and the number of rejected task is increased by one.

Step Five: SLC promotes the security levels of all the tasks in the accepted queue as high as possible, subjecting to two constraints: (1) Raising of an accepted task's security level should still guarantee its deadline. (2) Increasing security levels should not result in any rejection of currently accepted tasks. The rationale of the above policy is that guarantee ratio, a fundamental performance metric for real-time systems, has to be given the highest priority. Security level promotion can be performed only when such an adjustment leads to no rejection of any accepted tasks residing in the accepted queue. Note that the deadlines of tasks in the accepted queue can be guaranteed, although the adjustment of security levels might affect the schedulability of future coming tasks. We will demonstrate how the security level controller improves security levels shortly in section 3.3.

Step Six: At this point, the security level SL_{i+1} of the next starting task's can be optimized. Increase security value by SL_{i+1}. Its completion time CT_{i+1} is calculated. Therefore, we obtain a new time slot $[ST_{i+1}, CT_{i+1}]$ for the task. Steps 3-6 are repeatedly executed until all the arrival tasks are processed in one run.

Now we are in a position to briefly outline the ideas of the three alternative EDF security-aware scheduling algorithms that are envisioned as baseline algorithms.

EDF_MINS: The admission controller intentionally selects the lowest security level of each coming task. Therefore, the guarantee ratio is improved at the cost of reducing overall security value of the system.

EDF_MAXS: The admission controller chooses the highest security level for each accepted task. As a result, security values are increased while decreasing guarantee ratios.

EDF_RNDS: Unlike EDF_MINS and EDF_MAXS, EDF_RNDS randomly picks a value within the range of security levels specified by each arrival task. Hence, security values and guarantee ratios of the system are unlikely to be pushed to extremes, meaning that the performance of EDF_RNDS is somewhere between that of the above two algorithms.

3.2 Task model

Like many existing real-time task models presented in the literature (Lu et al, 1999; Palis, 2002; Palencia et al, 1998; Thomadakis and Liu, 1999), our task model assumes that all tasks have soft deadlines and all tasks are independent of one another. In our system model, there exists a single server handling tasks from n sources. Each task has a range of security level requirement where the server is allowed to choose a value under particular constraints. For synthetic workload conditions, it is assumed that tasks arrive at the system according to a Poisson process. Task T_i is represented as a tuple (AT_i, ET_i, SL_i, D_i), where AT_i and ET_i denote the arrival time and the worst case execution time of task *i*. SL_i and D_i represent the security level and soft deadline of task *i*. Note that given task *i*, ET_i and D_i of are known to the scheduler in advance. Without loss of generality, we assume that the worst case execution time ET_i is the execution time of task *i* without having its security overhead in place. In addition, in the task model each task is

assigned a quality of security measured as a security level SL_i that is in the range [1, 2, …, R], where 1 and R are the lowest and highest levels of security, and R is set to 10 in our experiments.

To simplify the security overhead model without loss of generality, we make use of Equation (1) to model the security overhead envisioned as the extra execution time experienced by task *i*. The rationale behind the overhead model is based on our observation that the security overhead of a particular application tends to be proportional to the execution time of the application. Importantly, the overhead model captures the essence of applications with security demands to provide reasonable estimates of security overheads based on particular security levels. Very recently, a sophisticated security overhead model was constructed by Xie *et al.* (Xie et al, 2005). This comprehensive model can be easily integrated into our scheduling architecture.

$$SO_i = ET_i * (SL_i / R) \quad (1)$$

where SO_i is the security overhead of task *i*; and SL_i is the security level provided to task *i*. Thus, the total execution time of task *i* can be expressed as Equation (2).

$$WL_i = ET_i + SO_i = ET_i * (1 + SL_i / R) \quad (2)$$

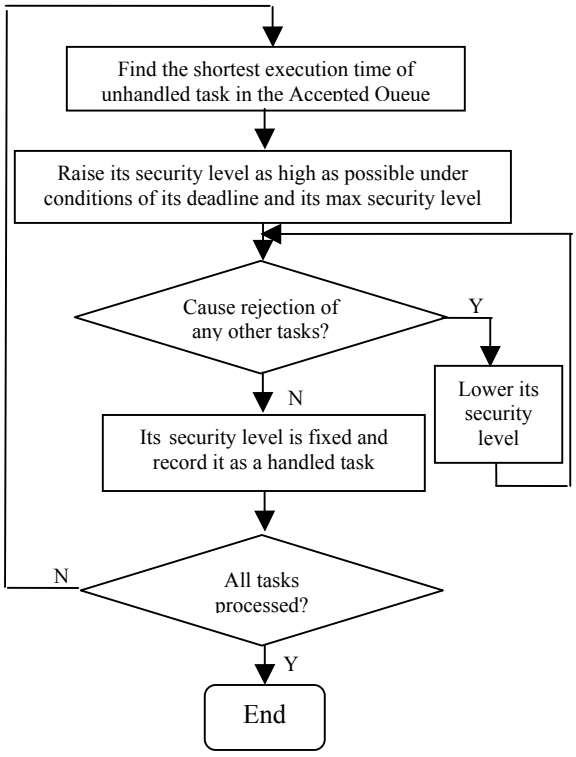

Figure 2 *Workflow of security lever controller*

3.3 Security Level Controller

The ultimate goal of this study is to maximize overall system performance (See Equation 3), which reflects both guarantee ratio and security performance (See Equation 4 and 5). To achieve the goal, we designed an optimized EDF scheduling algorithm with security awareness, or EDF_OPTS, which is capable of maintaining high guarantee ratios while maximizing security values. In particular, the EDF_OPTS algorithm strives to achieve the best trade-off between the guarantee ration and security value. It can be accomplished by applying a security level controller to our EDF_OPTS algorithm. Figure 2 shows the flow chart of the controller that adjusts security levels.

The controller is intended to assign the highest security level to an admitted task in the accepted queue while making the task schedulable under the following two constraints. First, the increase of the task's security level cannot violate its deadline constraint. Second, the security level promotion for the task has to result in no potential rejection of subsequent accepted tasks. In an effort to maximize the overall security value, we propose a strategy to give accepted tasks with short execution time (security overhead is not incorporated) a higher priority and make the best effort to boost their security levels. The reason for doing so is that tasks with short execution times impose low security overheads compared with tasks with long execution times. Consequently, the EDF_OPTS algorithm is efficient in obtaining a high overall security value by judiciously adjusting security levels for the accepted tasks under this approach.

Since this research is focused on dynamic real-time environments, the EDF_OPTS algorithm can maximize security values for tasks residing in the accepted queue, providing locally optimized security performance. The experimental results reported in the following section show that this local optimization yield an appealing overall security value.

4 SIMULATION RESULTS AND DISCUSSIONS

4.1 Performance metrics and parameters

For the sake of simplicity, throughout this section EDF_OPTS is referred to as OPTS. Similarly, the baseline algorithms are referred to as MINS, MAXS, and RNDS, respectively.

In order to evaluate the effectiveness of our approach, it is useful to compare the OPTS algorithm against three baseline algorithms, namely, MINS, MAXS, and RNDS. The following three important performance metrics for highly secure real-time systems will be used to quantitatively evaluate the proposed algorithm. Overall system performance (OSP) is measured as a product of Security Value (SV) and Guarantee Ratio (GR)

$$OSP = SV * GR . \quad (3)$$

The security value and guarantee ratio of a system are defined as follows.

$$SV = \sum_{j=1}^{N} SL_j , \quad (4)$$

where *N* is the number of accepted tasks in one run.

$$GR = N / M , \quad (5)$$

where M is the total number submitted tasks in one run.

Task arrival rate λ and execution time range (ETR) are two workload parameters. We evaluate our algorithm performance under a wide range of system workload conditions by varying λ and ETR.

In addition to synthetic simulations, we also tested the four algorithms using a real world trace. We modified the trace used in (Harchol-Balter and Downey, 1997; Zhang et al, 2000) by adding deadlines for all tasks in the trace, which was collected from one workstation in the third time interval. There are totally 572 tasks in the trace. The assignment of deadlines is controlled by the deadline base denoted as β, which sets an upper bound on tasks' slack times. We use Equation (6) to generate task i's deadline d_i.

$$d_i = a_i + e_i + SO_i^{max} + \beta \qquad (6)$$

where a_i and e_i are the arrival and execution times obtained from the real-world trace. SO_i^{max} is the maximal security overhead (measured in second), which is computed by Equation (1) using the highest security level of task i.

4.2 Experiments design

There are two types of simulations conducted in this work. In Section 4.3, we designed four groups of experiments to measure the performance metrics under four workload situations using synthetic workload. Each curve in the figures is composed of 30 points, each of which is an average value computed from 30 runs. In the first three group experiments, we measure the security value, guarantee ratio, and overall system performance of the simulated real-time system when the task arrival rate is increased from 0.1 to 3.0 with the increment of 0.1. The ETR in these experiments is set to be [1, 50], [1,100] and [1,200] respectively. In the fourth group experiment, we stressed the security workload by keeping the guarantee ratio as high as 100%. This workload reflects a scenario where the deadlines of real-time tasks are not tight. In Section 4.4, we tested the four algorithms using the trace data with five different deadline bases varying from 100 seconds to 2000 seconds.

4.3 Experimental results using synthetic data

Below are figures that we obtained from the experiments designed above. Figures 3-6 depict the security values and guarantee ratios as functions of the task arrival rate λ.

As we can see from Figure 3, OPTS does not show its strength when ETR is [1, 50] if we consider SV and GR separately. It takes the second place among four scheduling algorithms in terms of security value and guarantee ratio. On average, the security value of OPTS is 7.25% lower than that of MAXS (please refer to Table 1) because MAXS attempts to accept real-time tasks using the highest security levels. Under this workload where the arrival tasks are more homogeneous in terms of execution times, security levels specified by the tasks vary slightly.

As we mentioned before, if a fixed security level is maintained, long execution times result in high security overheads, which in turn give rise to the potential rejection of submitted tasks due to the high system load. In cases where arrival tasks have similar short execution times, the security overhead of the lowest security level is very close to that of the highest security level. The implication is that assigning the highest security level for each accepted task is

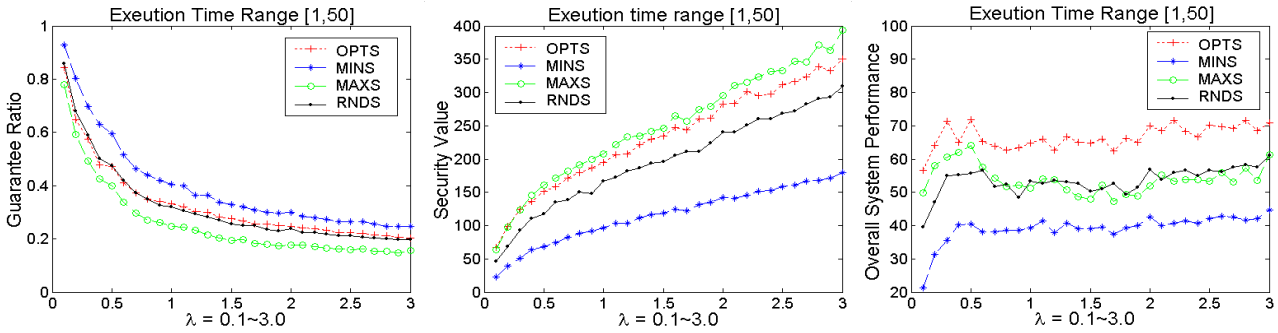

Figure 3 *Performance of four algorithms when ETR=[1, 50]*

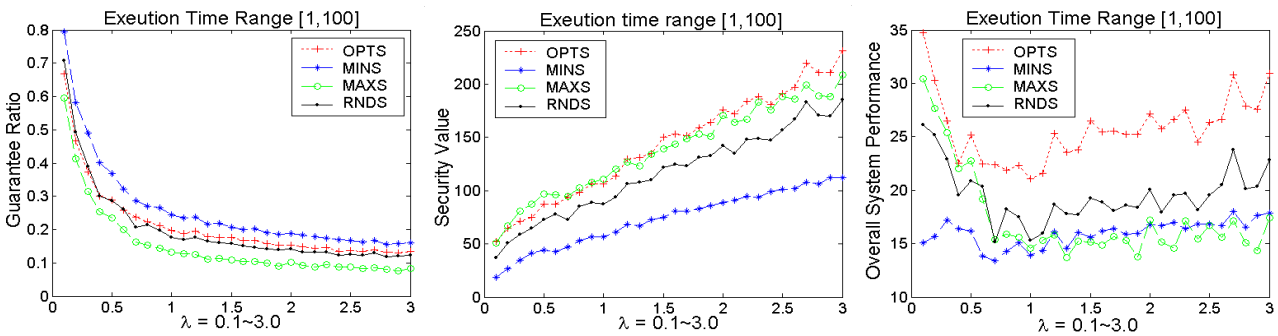

Figure 4 *Performance of four algorithms when ETR=[1, 100]*

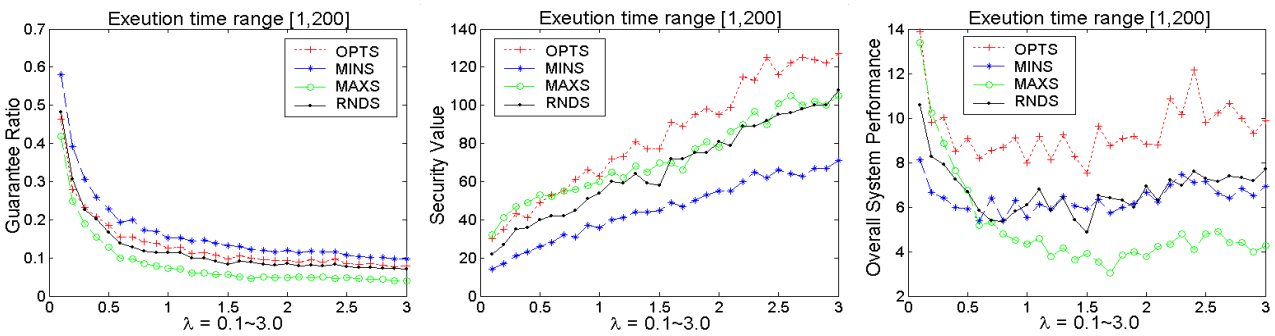

Figure 5 *Performance of four algorithms when ETR=[1, 200]*

unlikely to incur potential rejections of forthcoming tasks. Therefore, in this scenario, which largely depends on the nature of arrival tasks, MAXS has the best performance in security value.

Under the same workload condition the guarantee ratio of MINS is 6% higher than that of OPTS (please refer to table 1). The reason is self-explanatory since MINS always picks the lowest security level for an acceptable task and thus it can accept more tasks than OPTS whose security level optimization may cause rejection of later deadline tasks. However, OPTS is in the first place in overall system performance, which is computed by Equation (3). Figure 3 (c) demonstrates that OPTS achieved the highest performance if security value and guarantee ratio are considered equally important and simultaneously.

Figure 4 plots security value and guarantee ratio of the four scheduling algorithms when the arrival tasks' execution time range is within [1,100]. This experimental setting reflects a system workload, where some tasks arriving in the system have relatively long execution times.

In general OPTS outperforms all the other three algorithms in security performance. The interesting results indicate that OPTS even can achieve higher security quality than MAXS, which was expected to deliver the highest security value in under various workloads. We attribute this result to the fact that the MAXS algorithm does not incorporate a security level controller and, therefore, MAXS tends to accept some tasks with long execution times, making a high likelihood to reject submitted real-time tasks. On the contrary, OPTS makes the best effort to elevate the security levels of tasks with short execution times, thereby resulting in low rejection rate for real-time tasks arrived in the system. Although MAXS is intended to select the highest security levels for accepted tasks, it inherently tends to reject tasks due to high system load induced by the highest security levels of accepted tasks with long execution times. In particular, the proposed OPTS improves the security value over MINS by 94.41% (See table 1), and the guarantee ratio of OPTS in this case is almost the same as that of MINS (only 5% less than the guarantee ratio of MINS). It is desirable to achieve an improvement in the quality of security by 94.41% for real-time systems with security demands at the cost of a marginal guarantee ratio loss. As for overall system performance, OPTS noticeably outperforms all the other three algorithms (Figure 4).

We observe from Figures 5 that OPTS is in the leading position in term of security performance when the execution time range is set to [1, 200]. This result is consistent with that observed from the previous experiments. More importantly, the distance between OPTS and MAXS even becomes larger (OPTS's security value is 16.6% higher than that of MAXS). Meanwhile, the discrepancy of guarantee ratio between OPTS and MINS becomes narrow which is only 3% less than MINS. However, OPTS's security value is still 85.27% higher than that of MINS. Like the two experiments mentioned above, OPTS is still in the first place in terms of overall system performance (Figure 5 c). We also test OPTS in an extreme case when system workload is very light. In this case, the guarantee ratio of all four algorithms is 100%.

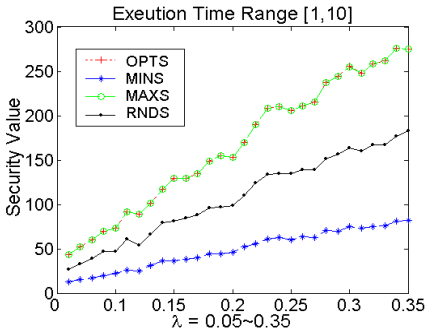

Figure 6 *Security value, ETR [1, 10], λ[0.05, 0.35]*

Figure 6 plots the security values under the scenarios where the guarantee ratios are set to 100%. Figure 6 plainly shows that OPTS achieves significant performance improvements in security values over MINS and RNDS, and ties with MAXS in security performance.

Table 1 *Summary of guarantee ratios and security values. (ETR – Execution Time Range, GR - Guarantee Ratio, SV – Security Value)*

EDF_	ETR[1,50]		ETR[1,100]		ETR[1,200]	
	GR	SV	GR	SV	GR	SV
MINS	0.39	116.13	0.26	73.57	0.16	45.77
MAXS	0.25	250.77	0.15	138.53	0.08	72.70
RNDS	0.31	197.43	0.20	116.77	0.12	67.17
OPTS	0.33	233.83	0.21	143.03	0.13	84.80

Table 2 *Summary of overall system performance improvements*

OSP	[1,50]	[1,100]	[1,200]	Gain(%)
MINS	45.29	19.13	7.32	65.29
MAXS	62.69	20.78	5.78	32.86
RNDS	61.20	23.35	8.06	28.04
OPTS	77.16	30.04	11.02	

Table 1 and 2 summarize the overall performance improvements of our OPTS over the other baseline scheduling algorithms. A major observation made from Table 2 is that OPTS is constantly the best algorithm with respect to overall system performance among all tested alternatives. This is because the overall system performance by definition is a product of guarantee ratio and security value, and the OPTS approach strives to simultaneously maximize both guarantee ratio and security value by achieving the best trade-off between the two performance metrics. In particular, OPTS improves the overall system performance over the MINS algorithm by an average of 65.29%. Similarly, compared with MAXS and RNDS, the proposed OPTS algorithm delivers improvements in the overall system performance by averages of 32.86% and 28.04%, respectively.

4.4 Trace-driven simulation

To validate the results from the synthetic experiments described in Section 4.3, we used a real world trace to drive the simulations of our OPTS algorithm. The trace was sampled from a workstation during a certain period of time (Harchol-Balter and Downey, 1997; Zhang et al, 2000). Since there were no deadlines for tasks in the original trace, we generated a deadline for each task based on Equation (6). In addition, we tested five scenarios where the deadline base varies from 100 seconds to 2000 seconds.

Figure 7 shows the experimental results for these four algorithms. We observe from Figure 7 (a) that OPTS and MINS exhibit similar performance in terms of guarantee ratio, whereas the OPTS noticeably outperforms the MAXS and RNDS algorithms. We attribute the performance improvement of OPTS over MAXS and RNDS to the fact that OPTS judiciously boosts the security levels of accepted tasks under the condition that the deadlines of the tasks are guaranteed, thereby maintaining relatively high guarantee ratios. Unlike OPTS, the MAXS and RNDS policies improve quality of security at the cost of missing deadlines. Figure 7 (a) illustrates that the guarantee ratios of four algorithms increase with the increasing value of the deadline base. This is because the large deadline base leads to long slack times, which in term tend to make the deadlines more likely to be guaranteed. Figure 7 (b) plots security values of the four algorithms when the deadline base is increased from 100 to 2000 seconds. A 100 second deadline base implies a very tight deadline setting because the average task execution time in the trace is 9454 seconds. Similarly, a 2000 second deadline base hints a relatively loose deadline for each task. It reveals that OPTS consistently performs better, with respect to quality of security, than all the other three approaches. When the deadlines are tight, the security values of OPTS are very close to those of MAXS. However, OPTS significantly outperforms MAXS when the deadline base becomes large. This is because that OPTS can accept more tasks than MAXS. Interestingly, when the deadlines become loose, the performance improvements of OPTS over the three competitor algorithms are more pronounced. The results clearly indicate that we can gain more performance benefits from OPTS approach under the circumstance that real-time tasks have loose deadlines.

The overall system performance improvements achieved by OPTS are plotted in Figure 7 (c). The first observation deduced from Figure 7 (c) is that the value of overall system performance increases with the deadline base. This is mainly because the overall system performance is a product of security value and guarantee ratio, which become higher when the deadlines are loose due to the high deadline base value.

A second observation made from figure 7 (c) is that the OPTS algorithm significantly outperforms all the other three alternatives. This can be explained by the fact that although the guarantee ratios of OPTS and MINS are similar, OPTS considerably improves security values over the other algorithms, while achieving higher guarantee ratio than MAXS and RNDS. This result suggests that if quality of security is the sole objective in scheduling, OPTS is more suitable for real-time tasks than the other alternatives. In contrast, if schedulability is the only performance objective, OPTS can maintain similar guarantee ratios as those of

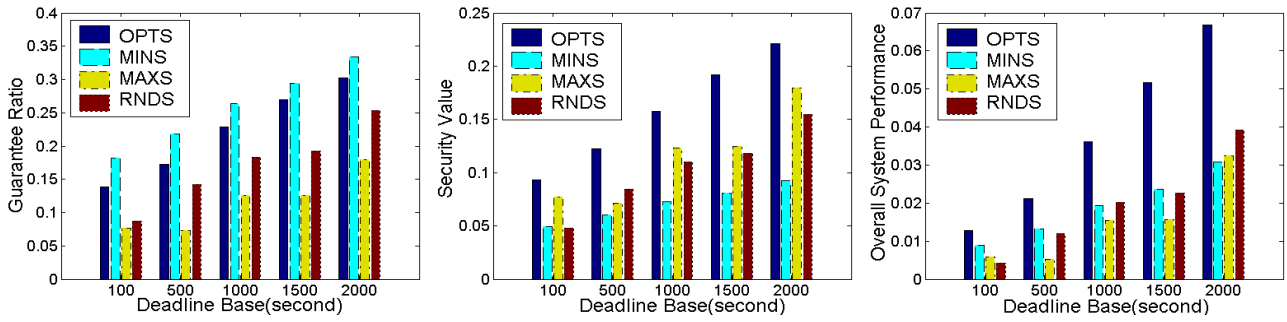

Figure 7 *Performance impact of deadline base*

MINS, whose security performance is the worst among the four algorithms.

Last but not least, Figure 7 (c) indicates that the overall performance improvement of OPTS over the other three algorithms becomes more pronounced when the deadlines are looser, implying that more performance benefits can be obtained by OPTS for real-time tasks with large slack times. This is because the OPTS approach is more sensitive to the change in deadlines than the other schemes.

5 CONCLUSIONS

In real-time systems with high security demands, schedulers not only are required to achieve high guarantee ratios to elevate system throughput and utilization, but also need to provide high quality of security for real-time applications running on the systems. To develop high security real-time systems where the above requirements are fulfilled, we proposed in this paper an optimized dynamic real-time scheduler with security–awareness (referred to as EDF_OPTS) in addition to three basic security-aware scheduling algorithms. The EDF_OPTS is designed in a way that makes it possible to achieve the best trade off between guarantee ratio and security performance. In particular, our EDF_OPTS algorithm leverages an intelligent security level controller to dynamically and adaptively boost the security levels of accepted tasks while maintaining a reasonably high guarantee ratio.

To the best of our knowledge, our algorithm is the first non-trivial algorithm designed to dynamically schedule tasks in real-time systems with security constraints. Our approach is proven to deliver significant improvements in overall system performance under a wide range of workload patterns. Specifically, our approach can provide overall performance improvement by up to 65.29%, which indicate that the EDF_OPTS algorithm is capable of simultaneously maximizing security service level and system guarantee ratio, two equally important metrics for high security real-time systems such as a real-time stock quote and trading system. More importantly, EDF_OPTS outperforms EDF_MAXS with respect to quality of security in most cases.

The drawback of our approach is that the guarantee ratio performance of EDF_OPTS is roughly 4.5% worse than that of EDF_MINS when the execution time range varies from [1, 50] to [1, 200]. Although this performance degradation mainly depends on the inherent nature of the algorithms, we are able to further improve the performance by applying a feedback control mechanism. Future studies include: (1) the design of the feedback control mechanism used to dynamically monitor and adjust guarantee ratios, (2) improving quality of security for networks by integrating EDF_OPTS into our communication-aware load balancing scheme (Qin and Jiang, 2005), and (3) incorporating EDF_OPTS into our task duplication management system (Qin 2005).

ACKNOWLEDGEMENT

This work was partially supported by a start-up research fund (103295) from the research and economic development office of the New Mexico Tech and a DoD IASP Capacity Building grant.

REFERENCES

Abdelzaher, T.F. and Shin, K.G. (1999) 'Combined Task and Message Scheduling in Distributed Real-Time Systems', *IEEE Trans. Parallel and Distributed Systems*, Vol. 10, No. 11, November.

Abdelzaher, T.F. and Shin, K.G. (1998) 'End-host Architecture for QoS-Adaptive Communication', *Proceedings of IEEE Real-Time Technology and Applications Symposium*, pp. 121, June.

Azzedin, F. and Maheswaran, M. (2002) 'Towards Trust-Aware Resource Management in Grid Computing Systems', *Proceedings of the 2nd IEEE/ACM Int'l Symposium on Cluster Computing and the Grid*, Berlin, Germany, May.

Donoho, G. (2004) 'Building a Web Service to Provide Real-Time Stock Quotes,' *MCAD.Net*, February

Harchol-Balter, M. and Downey, A. (1997) 'Exploiting Process Lifetime Distributions for Load Balancing', *ACM transaction on Computer Systems*, Vol. 3, No. 31, March.

Kalogeraki, V., Melliar-Smith, P.M. and Moser, L.E. (2000) 'Dynamic scheduling for soft real-time distributed object systems', *Proceedings of IEEE Int'l Symposium on Object-Oriented Real-Time Distributed Computing*, pp.114-121.

Kwok, Y.K., Ahmad, I. and Gu, J. (1996) 'FAST: A Low-Complexity Algorithm for Efficient Scheduling of DAGs on Parallel Machines', *Proceedings of the 25th Int'l Conference on Parallel Processing*, pp. II:150-157.

Liu, C.L. and Layland, J.W. (1973) 'Scheduling Algorithms for Multiprogramming in a Hard Real-Time Environment', *Journal of the ACM*, Vol. 20, No. 1, pp. 46-61,

Lu, C.Y., Stankovic, J.A., Tao, G. and Son, S.H. (1999) 'Design and Evaluation of a Feedback Control EDF Scheduling Algorithm', *Proceedings of IEEE Real-Time Systems Symposium*, Phoenix, Arizona, December.

Marbini, A.D. and Sacks, L. (2002) 'Considering Control Theory in Resource Management Scenarios', *Proceedings of London Communications Symposium*, London, England.

Manimaran, G. and Murthy, C.S.R (1998) 'An Efficient Dynamic Scheduling Algorithm for Multimachine Real-Time Systems', *IEEE Transaction on Parallel and Distributed Systems*, Vol. 9, No. 3, pp. 312-319.

Palencia, J.C. and Gonzalez, H.M. (1998) 'Schedulability analysis for tasks with static and dynamic offsets', *Proceedings of the 19th IEEE Real-Time Systems Symposium*, pp.26-37.

Palis, M.A. (2002) 'Online Real-Time Job Scheduling with Rate of Progress Guarantees', *Proceedings of the 6th Int'l Symposium on Parallel Architectures, Algorithms, and Networks*, pp. 65-70.

Qin, X. and Jiang, H. (2005) 'Improving Effective Bandwidth of Networks on Clusters using Load Balancing for Communication-Intensive Applications', *Proceedings of the 24th IEEE Int'l Performance, Computing, and Communications Conf. (IPCCC 2005),* Phoenix, Arizona, April.

Qin, X. (2005) 'Improving Network Performance through Task Duplication for Parallel Applications on Clusters', *Proc. the 24th IEEE Int'l Performance, Computing, and Communications Conference,* Phoenix, Arizona, April.

Qin, X. and Jiang, H. (2001) 'Dynamic, Reliability-driven Scheduling of Parallel Real-time Jobs in Heterogeneous

Systems', *Proc. Int'l Conf. Parallel Processing*, Valencia, Spain, pp.113-122.

Qin, X., Jiang, H. D. and Swanson, R. (2002) 'An Efficient Fault-tolerant Scheduling Algorithm for Real-time Tasks with Precedence Constraints in Heterogeneous Systems', *Proc. Int'l Conf. Parallel Processing,* British Columbia, Canada, pp.360-368*,* August.

Ramamritham, K. and Stankovic, J. A. (1984) 'Dynamic task scheduling in distributed hard real-time system', *IEEE Software*, Vol. 1, No. 3, July.

Stankovic, J.A., He, T. and Abdelzaher, T. et al, (2001) 'Feedback Control Scheduling in Distributed Real-Time System," *Proc. the 22nd IEEE Real-Time Systems Symposium*, London, England, December.

Stankovic, J., Spuri, M., Ramamritham, K. and Buttazzo, G.C. (1998) 'Deadline Scheduling for Real-time systems: EDF and Related Algorithms', *Kluwer Academic Publishers*.

Thomadakis, M.E. and Liu, J.C. (1999) 'On the efficient scheduling of non-periodic tasks in hard real-time systems', *Proc. IEEE Real-Time Systems Symposium.* pp.148-151.

VeriSign Corp., (2003) 'Simplifying Application and Web Services Security - VeriSign Trust Gateway,'

Xie, T., Qin, X., and Sung, A. (2005) 'SAREC: A Security-Aware Scheduling Strategy for Real-Time Applications on Clusters', *Proc. of the 34th Int'l Conf. Parallel Processing*, pp.5-12, Norway.

Zhang, X., Qu Y. and Xiao, L. (2000) 'Improving Distributed Workload Performance by Sharing both CPU and Memory Resources," *Proc. 20th Int'l Conf. Distributed Computing Systems (ICDCS)*, April.

Zhao, W., Ramamritham, K. and Stankovic, J.A. (1987) 'Preemptive Scheduling Under Time and Resource Constraints', *IEEE Transactions on Computers*, pp. 36-38.