This paper appeared in the *Proceedings of the 19th International Parallel and Distributed Processing Symposium* (IPDPS'05), *the 4th Int'l Workshop on Performance Modeling, Evaluation, and Optimization of Parallel and Distributed Systems,* IEEE/ACM, April 4-8, 2005.

Dynamic Task Scheduling with Security Awareness in Real-Time Systems

Tao Xie Andrew Sung Xiao Qin
Department of Computer Science
New Mexico Institute of Mining and Technology
801 Leroy Place, Socorro, New Mexico 87801-4796
{xietao, sung, xqin}@cs.nmt.edu

Abstract

An increasing number of real-time applications, such as 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 quality of security for real-time tasks while improving resource utilization. We have conducted extensive simulation experiments to quantitatively evaluate the performance of our approach. Specifically, experimental results show that compared with three heuristic algorithms, the proposed algorithm can consistently improve overall system performance in terms of quality of security and system guarantee ratio under a wide range of workload characteristics.

1. Introduction

Many dynamic real-time scheduling algorithms such as Earliest Deadline First (EDF) [1] and Spring scheduling algorithm [2] [3] have been developed when the scheduling algorithms do not possess complete knowledge of the task set or its timing constraints. For example, new created tasks, which are not known to the algorithm when it is handling the current task set, may arrive at a future 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 are even not 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 propose four security-aware dynamic real-time scheduling algorithms, namely, EDF_MINS, EDF_MAXS, EDF_RNDS, and EDF_OPTS. The first three algorithms are earliest deadline first (EDF) scheduling algorithms with security awareness, which intentionally choose minimal, maximal, and random security level specified by the incoming tasks, respectively. Unlike these three algorithms, the last one is an optimized security-aware EDF scheduling algorithm in the sense that it is able to improve the quality of security of tasks admitted to the 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 tasks arrived in the system. To the best of our knowledge, our algorithms are the first non-trivial algorithms that are designed to dynamically schedule tasks with security requirements in a real-time environment. The basic idea behind our algorithms is to incorporate our security-aware scheme into the Earliest Deadline First algorithm, or EDF, to construct the four security-aware EDF scheduling algorithms. The advantage of our algorithms is that real-time systems with high-security demands can make use of the proposed algorithm 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 environments.

In a real-time system with high security demands such as a real-time stock quote update and trading system [4], 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 security level requirement range which must be both 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 [5]:

- 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 service levels specified by a particular 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 [5]:

- 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 the guarantee ratio. of a task varies depending on resource utilization. In our scheduling model, each task has a range of security requirement levels denoted as $[SL_{min}, SL_{max}]$. 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 transplanted in this situation. In addition, some straightforward security-aware real-time scheduling algorithms such as 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 level from a task's security requirement range 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. While a scheduling algorithm maps real-time tasks to processors in a system such that deadlines and response time requirements are met [11], the system has to guarantee its functional and timing correctness even in the presence of hardware and software faults [14][15].

Many scheduling algorithms, which can be classified into two categories: static [17][19][20] and dynamic [16][18][21][22], 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 [16]. While real-time tasks in Palis's scheduling framework are preemptive [16], it is assumed in our study that real-time tasks are non-preemptive.

Abdelzaher and Shin proposed a new communication subsystem architecture for Quality of Service (QoS) adaptive real-time applications such as streaming stored video on end-hosts [6]. 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 the algorithm devised in [6] 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 in [6]. However, the scheduling algorithms presented in this paper 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 task's security levels.

Azzedin and Maheswaran examined the integration of the notion of "trust" into resource management of a large-scale wide-area system such as the Grid [7]. They argued that there is a "trust relationship" between a resource provider (RP) and a resource consumer (RC). The hypothesis is that if the RMS (Resource Manage System) is aware of the security requirements of the resources and tasks it can perform the allocations such that the security overhead can be minimized. However, the allocation algorithm presented in [7] is developed in the context of non-real-time applications, and it is not suitable for real-time systems where tasks' deadlines have to be factored in.

Previous research has applied control theory to dynamic real-time scheduling algorithms [8][9]. The basic idea behind the feedback control EDF scheduling system presented in [8] 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. 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. Our scheduling approach differs from the above scheduling techniques in that ours is security aware in nature by making use of a Security Level controller.

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 (SLC), 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.

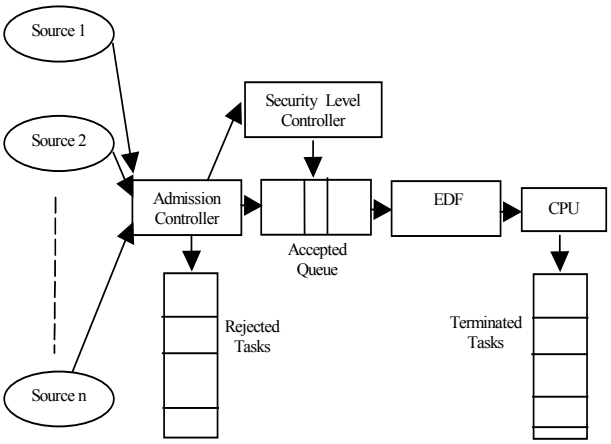

Figure 1. System architecture

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 not, whereas 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 the admitted tasks in which security levels are optimized by the Security Level Controller.

Step One: Initialize the scheduler. Overall Security Value (*SV*) and Number of Rejected Task (*NRT*) 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 (a) the security overhead (extra execution time) imposed by the task's lowest security requirement, and (b) 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: (a) Raising of an accepted task's security level should still guarantee its deadline. (b)

The promotion of 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 SLC 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 heuristic 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, the security values are increased while decreasing the guarantee ratio.

EDF_RNDS: Unlike EDF_MINS and EDF_MAXS, EDF_RNDS randomly pick a value within the range of security levels specified by each arrival task. Hence, the security value and guarantee ratio 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 [8][16][17][18], 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. 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, 3, …, 10], where 1 and 10 are the lowest and highest levels of security.

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.

$$SO_i = ET_i * (SL_i / 10), \quad (1)$$

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).

$$WL_i = ET_i + SO_i = ET_i * (1 + SL_i / 10). \quad (2)$$

3.3. EDF Scheduling with Security Awareness

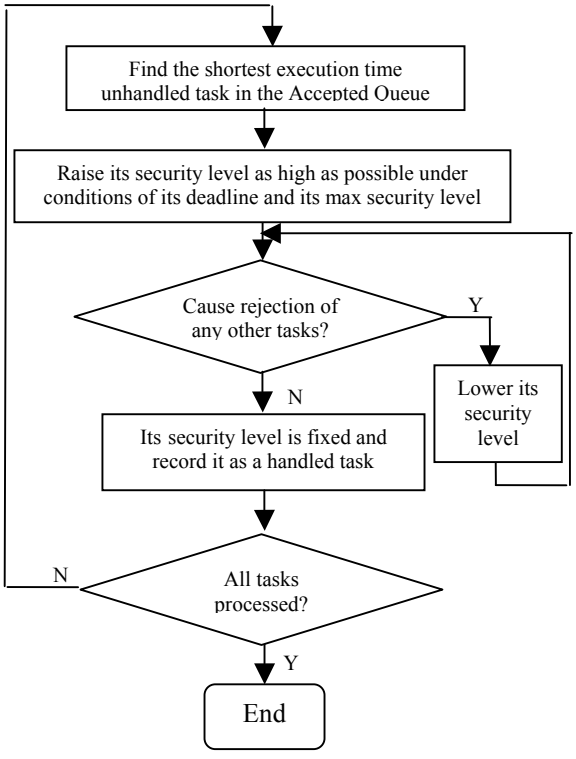

Figure 2. Workflow of security lever controller

The ultimate goal of this study is to maximize the overall system performance (See Equation 3), which reflects the 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 the Security Level Controller to our EDF_OPTS algorithm. Figure 2 shows the flow chart of the Security Level Controller. The controller is intended to assign the highest security level to an admitted task *i* 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 task impose low security overheads compared with tasks with long execution times. Consequently, the EDF_OPTS 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

In order to evaluate the effectiveness of our approach, it is useful to compare the EDF_OPTS algorithm against three baseline security-aware algorithms, namely, EDF_MINS, EDF_MAXS, and EDF_RNDS. The following three important performance metrics for highly security 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.

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

N is the number of tasks, which are accepted in one run. *M* is the total number tasks, which are submitted in one run.

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

Task arrival rate and 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.

4.2. Experiments Design

We designed four groups of experiments to measure the performance metrics under four workload situations. . Each curve the figure is composed of 30 points, each of which is an average value computed from 30 runs. In the first three simulation experiments, we measure the security value and guarantee ratio 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 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.

4.3. Experimental Results and Analysis

Below are the seven 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, EDF_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 EDF_OPTS is 7.25% lower than that of EDF_MAXS (please refer to table 1) because EDF_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 unlikely to incur potential rejections of forthcoming tasks. Therefore, in this scenario, which largely depends on the nature of arrival tasks, EDF_MAXS has the best performance in security value.

Under the same workload condition the guarantee ratio of EDF_MINS is 6% higher than that of EDF_OPTS (please refer to table 1). The reason is self-explanatory since EDF_MINS always picks the lowest security level for an acceptable task and thus it can accept more tasks than EDF_OPTS whose security level optimization may cause rejection of later deadline tasks.

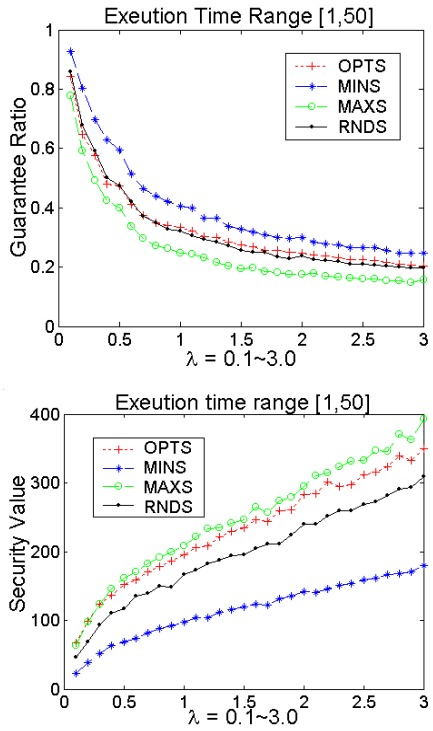

Figure 3. Guarantee ratios and security values of four algorithms when ETR=[1, 50]

Figures 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.

EDF_OPTS outperforms all the other three algorithms in security performance. The interesting results indicate that EDF_OPTS even can achieve higher security quality than EDF_MAXS, which was expected to deliver the highest security value in under various workloads. We attribute this result to the fact that because the EDF_MAXS algorithm does not incorporate a Security Level Controller and, therefore, EDF_MAXS tends to accept some tasks with long execution times, making a high likelihood to reject submitted real-time tasks. On the contrary, EDF_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 EDF_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 EDF_OPTS improves the security value over EDF_MINS by 94.41% (See table 1), and the guarantee ratio of EDF_OPTS in this case is almost the same as that of EDF_MINS (only 5% less than the guarantee ratio of EDF_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.

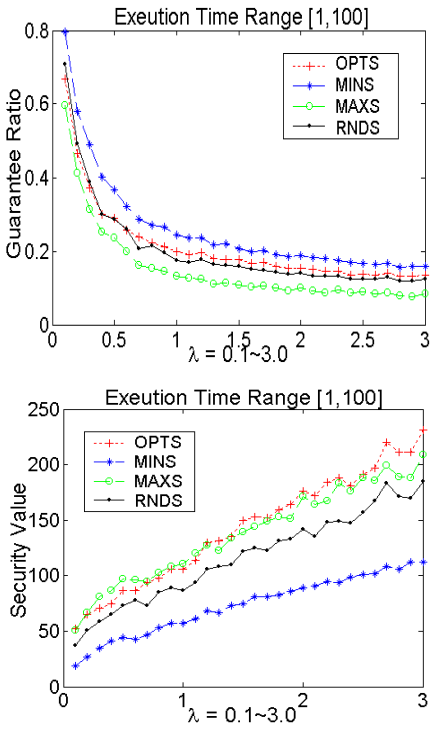

Figure 4. Guarantee ratios and security values of four algorithms when ETR=[1, 100]

We observe from Figures 5 that EDF_OPTS is in the leading position in term of security performance when ETR is set to [1, 200]. This result is consistent with that observed from the previous experiments. More importantly, the distance between EDF_OPTS and EDF_MAXS even becomes larger (EDF_OPTS's security value is 16.6% more than that of EDF_MAXS). Meanwhile, the discrepancy of guarantee ratio between EDF_OPTS and EDF_MINS becomes narrow which is

only 3% less than EDF_MINS. However, EDF_OPTS's security value is still 85.27% higher than that of EDF_MINS. We also test EDF_OPTS in an extreme case when system workload is very light. In this case, the GR of all four algorithms is from 99% to 100%.

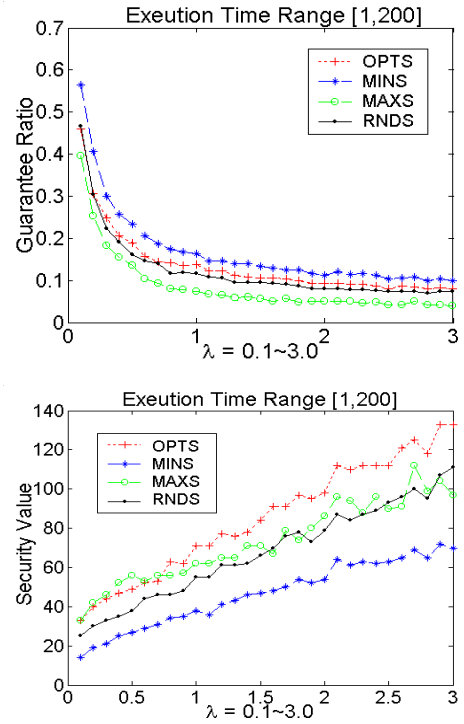

Figure 5. Guarantee ratios and security values of four algorithms when ETR=[1, 200]

Table 1. Summary of guarantee ratios and security values

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

Figure 6 plots the security values under the scenarios where the guarantee ratios are set to 100%. Figure 9 plainly shows that EDF_OPTS achieves significant performance improvements in security values over EDF_MINS and EDF_RNDS, and ties with EDF_MAXS in security performance.

Table 2 summarizes the overall performance improvements of EDF_OPTS over the other scheduling algorithms. A major observation made from Table 2 is that EDF_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 EDF_OPTS approach strives to simultaneously maximize both guarantee ratio and security value by achieving the best tradeoff between the two performance metrics. In particular, EDF_OPTS improves the overall system performance over the EDF_MINS algorithm by an average of 65.29%. Similarly, compared with EDF_MAXS and EDF_RNDS, the proposed EDF_OPTS algorithm delivers improvements in the overall system performance by averages of 32.86% and 28.04%, respectively.

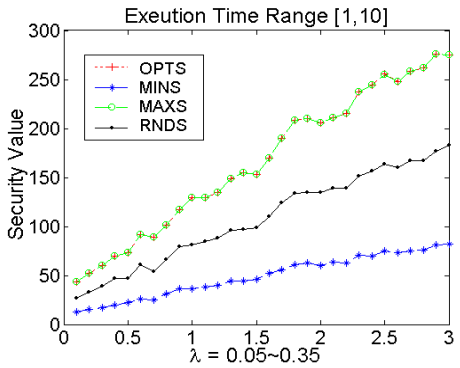

Figure 6. Security value, ETR [1, 10], λ[0.05, 0.35]

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	

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 have proposed 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 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 services. 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 [12], and (3) incorporating EDF_OPTS into our task duplication management system [13].

Acknowledgements

This work was partially supported by a DoD IASP Capacity Building grant and a start-up equipment grant (11HX11104) from the research and economic development office of New Mexico Tech.

References

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

[2] K. Ramamritham, J. A. Stankovic, “Dynamic task scheduling in distributed hard real-time system,” *IEEE Software*, Vol. 1, No. 3, July 1984.

[3] W. Zhao, K. Ramamritham, J.A. Stankovic, “Preemptive Scheduling Under Time and Resource Constraints,” *IEEE Transactions on Computers*, pp. 36-38, 1987.

[4] Gavin Donoho, “Building a Web Service to Provide Real-Time Stock Quotes,” *MCAD.Net*, February, 2004.

[5] VeriSign Corp., “Simplifying Application and Web Services Security - VeriSign Trust Gateway,”

[6] Tarek Abdelzaher, Kang G. Shin, “End-host Architecture for QoS-Adaptive Communication,” *Proc. IEEE Real-Time Technology and Applications Symp.*, pp. 121, June, 1998.

[7] Farag Azzedin, Muthucumaru Maheswaran, “Towards Trust-Aware Resource Management in Grid Computing Systems,” *Proc. the 2nd IEEE/ACM Int’l Symp. Cluster Computing and the Grid*, May 2002 Berlin, Germany.

[8] Chenyang Lu, John A. Stankovic, Gang Tao, Sang .H. Son, “Design and Evaluation of a Feedback Control EDF Scheduling Algorithm,” *Proc. IEEE Real-Time Systems Symp.*, Phoenix, Arizona, Dec. 1999.

[9] John A. Stankovic, Tian He, Tarek Abdelzaher, et al, “Feedback Control Scheduling in Distributed Real-Time System,” *Proc. the 22nd IEEE Real-Time Systems Symposium*, Dec. 2001 London, England.

[10] A Djafari Marbini, Lionel Sacks, “Considering Control Theory in Resource Management Scenarios,” *Proc. London Communications Symp.*, London, England, 2002.

[11] J. Stankovic, M. Spuri, K. Ramamritham and G.C. Buttazzo, “Deadline Scheduling for Real-time systems: EDF and Related Algorithms,” *Kluwer Academic Publishers*, 1998

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

[13] X. Qin, “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, 2005.

[14] X. Qin, and H. Jiang, “Dynamic, Reliability-driven Scheduling of Parallel Real-time Jobs in Heterogeneous Systems,” *Proc. Int’l Conf. on Parallel Processing*, Valencia, Spain, pp.113-122, 2001.

[15] X. Qin, H. Jiang, D. R. Swanson, "An Efficient Fault-tolerant Scheduling Algorithm for Real-time Tasks with Precedence Constraints in Heterogeneous Systems," *Proc. Int’l Conf. on Parallel Processing, British Columbia, Canada, pp.360-368, Aug. 2002*.

[16] M. A. Palis, “Online Real-Time Job Scheduling with Rate of Progress Guarantees," *Proc. the 6th Int’l Symp. Parallel Architectures, Algorithms, and Networks*, 2002, pp. 65-70.

[17] J.C. Palencia, and H.M. Gonzalez, “Schedulability analysis for tasks with static and dynamic offsets,” *Proc. the 19th IEEE Real-Time Systems Symp.*, 1998, pp.26-37.

[18] M.E. Thomadakis and Jyh-Charn Liu, “On the efficient scheduling of non-periodic tasks in hard real-time systems,” *Proc. IEEE Real-Time Systems Symp.*, 1999, pp.148-151.

[19] T.F. Abdelzaher and K.G. Shin, “Combined Task and Message Scheduling in Distributed Real-Time Systems,” *IEEE Trans. Parallel and Distributed Systems*, Vol. 10, No. 11, Nov. 1999.

[20] Y.K. Kwok, I. Ahmad, J. Gu, “FAST: A Low-Complexity Algorithm for Efficient Scheduling of DAGs on Parallel Machines,” *Proc. the 25th Int’l Conf. Parallel Processing*, 1996, pp. II:150-157.

[21] V. Kalogeraki, P.M. Melliar-Smith, L.E. Moser, “Dynamic scheduling for soft real-time distributed object systems,” *Proc. IEEE Int’l Symp. Object-Oriented Real-Time Distributed Computing*, 2000, pp.114-121.

[22] G. Manimaran and C.S.R Murthy, “An Efficient Dynamic Scheduling Algorithm for Multimachine Real-Time Systems,” *IEEE Trans. Parallel and Distributed Systems*, Vol. 9, No. 3, 1998, pp. 312-319.