INFORMATION ISSN 1343-4500
Volume 9, Number 2, pp.215-230 2006 International Information Institute

Open Issues and Challenges in Security-aware Real-Time Scheduling for Distributed Systems

Tao Xie*, Xiao Qin*, and Man Lin**

*Department of Computer Science, New Mexico Institute of Mining and Technology
801 Leroy Place, Socorro, New Mexico, 87801, USA
E-mail: {xietao, xqin}@cs.nmt.edu

** Department of Computer Science
St. Francis Xavier University
Antigonish, NS, B2G 2W5, Canada
E-mail: mlin@stfx.ca

Abstract. Task Scheduling for real-time distributed systems has been investigated extensively in the literature. However, conventional wisdom in dynamic scheduling ignores security requirements in real-time applications. As such, in addition to factoring quality of security in real-time applications running in the system, real-time scheduling algorithms need to be security-aware in nature. In this paper we first identify the open issues and challenges involved in designing and implementing security-aware real-time scheduling schemes, which are intended to consider both security and real-time constraints for distributed systems. Then five approaches towards achieving security aware in real-time scheduling are described: a security-aware architecture, a uniprocessor real-time scheduling algorithm with security awareness (EDF_OPTS) using a preliminary security overhead model and overall system performance metrics, security-aware real-time scheduling for homogeneous and heterogeneous distributed systems, and a feedback control mechanism to improve quality of security and schedulability in run time. Discussions on future security overhead models are also provided. Simulation results show that our EDF_OPTS algorithm significantly improves system performance in terms of quality of security and schedulability over three baseline algorithms under a wide range of workload characteristics.

Key Words: Security-aware, Real-time Scheduling, Distributed Systems, Quality of Security

1. Introduction

A real-time system is a system whose correctness depends not only on the results of computation, but also on time instants at which these results become available. Real-time systems are divided into two categories: *hard* real-time systems and *soft* real-time systems based on the consequences of missing deadlines. The consequences of missing deadlines of hard real-time systems may be catastrophic, whereas such consequences for soft real-time systems are relatively less damaging. Examples of hard real-time applications include aircraft control [4], radar for tracking missiles [13], and medical electronics [26]. On-line transaction processing systems are examples of soft real-time applications [15].

As real-time systems are becoming more and more complex, they are usually implemented in distributed computing platforms consisting of multiple modules interacting with one another to solve problems. Such real-time distributed systems are often heterogeneous. Growing evidence shows that scheduling is a key factor in obtaining high reliability and performance in heterogeneous distributed systems supporting real-time applications [14] [31]. A typical objective of real-time scheduling is to map tasks onto machines and order their execution in a way that task precedence requirements are satisfied and a minimum schedule length is given [17].

Due to the critical natures of applications executed in real-time systems, high security is an inherent requirement for such systems. A single design flaw with respect to security may render all security measures useless. This is especially true for distributed real-time applications. For example, in a real-time stock quote update and trading system, each incoming request from business partners and each outgoing response from an enterprise's back-end application (a request and a response are referred to as a task in this paper) have deadlines and security quality requirements, which have to be met by the server located between the business partners and enterprise back-end applications [3]. The server performs security operations on behalf of all its clients. In this case, a scheduler running on the server dynamically schedules all tasks in a way to guarantee their deadlines in addition to meeting their security requirements.

Security in distributed systems can roughly be divided into two components [27]. One part concerns communications among users or processes, possibly residing in different machines. A principal mechanism for ensuring secure communication is that of a secure channel. The second part concerns authorization, which deals with protecting access to resources. To protect a system against all possible security threats, there is a need to provide a *security policy* to precisely describe which actions entities in a system are allowed to take and which ones are prohibited. Note that entities include users, services, data, machines, and the like. Once a security policy has been laid down, security mechanisms enforcing the policy can be developed [1]. Important security mechanisms are encryption, authentication, authorization, and auditing.

Addressing security requirements in the context of real-time distributed systems poses various technical challenges. This paper aims at identifying open issues and challenges involved in providing high security for real-time distributed systems. The rest of the paper is organized as follows. Section 2 identifies open issues in real-time distributed systems with security requirements. Section 3 discusses approaches to achieve high security and schedulability for real-time distributed systems, and section 4 concludes the paper.

2. Real-time and Security Guarantees: Fundamental Challenges

The issue of scheduling on heterogeneous distributed systems was reported in the literature, and these studies addressed various aspects of a complicated problem. Ranaweera and Agrawal developed a scalable scheduling scheme for heterogeneous systems [21]. In [24] and [2], reliability cost, defined to be a product of processor failure rate and task execution time, was incorporated into scheduling algorithms for tasks with precedence constraints. However, these algorithms are unable to support real-time applications.

Conventional real-time scheduling algorithms such as Rate Monotonic (RM) algorithm [11], Earliest Deadline First (EDF) [25], and Spring scheduling algorithm [20][30] have been successfully applied in real-time systems. Previous work has been done to facilitate real-time computing in heterogeneous systems. Huh *et al*. proposed an approach to dynamically managing resources in real-time heterogeneous systems [6]. Santos *et al*. developed a probabilistic model for a client/server heterogeneous multimedia system [22]. These algorithms, however, could not tolerate any permanent processor failures. We proposed both static [16][18] and dynamic [17] reliability driven real-time scheduling schemes for heterogeneous systems which are able to tolerate failures.

Although real-time scheduling algorithms for distributed systems have been extensively investigated, less attention has been paid to real-time systems with security requirements in general. Most existing real-time scheduling algorithms are not security-aware in the sense that the existing real-time scheduling algorithms ignore security overhead while scheduling real-time tasks. Next, we present three challenges of security aware real-time scheduling.

- The first fundamental challenge is to specify security requirements for a real-time task. The security requirements of a task include how to specify the security of the task, the possible range of the security quality and the overhead of achieving some particular degree of security quality. The difficulty lies in measuring the quality of security of real-time tasks. To the best of our knowledge, no existing literature has directly addressed the issue of accurately estimating security overhead experienced by security-critical real-time applications, and this becomes a significant open issue in the development of real-time security-aware scheduling schemes. To tackle this problem, we are studying the fundamental security services and their corresponding mechanisms in the literature and proposing some quantitative measure for some standard security mechanisms. This will lead to mathematical models that can approximately estimate security overhead experienced by real-time tasks.
- The second fundamental challenge is to design and implement real-time security-aware scheduling schemes, which can meet specific real-time and security requirements of applications executing in distributed systems. The ultimate goal of

security-aware scheduling is to guarantee security constraints in addition to real-time requirements of tasks running in distributed systems.

- The third challenge is to introduce a new set of performance metrics including the quality of security, deadline guarantee ratio, overall system performance and scalability. Quality of security of a system can be represented by a formal measurement model based on the quality of security of individual tasks. Deadline guarantee ratio, or guarantee ratio for short is a ratio between the number of admitted tasks whose deadline can be met and the total number of submitted tasks. The scalability can be measured by the capacity of the real-time distributed systems in the sense that how the quality of security and guarantee ratio can be scaled by adding additional nodes, memory, or processing power.

3. Scheduling Approaches to Providing Security and Real-time Guarantees

3.1 Security-aware architecture

Developing the architecture of security-aware real-time service provider is the key to building trustworthy real-time systems. As heterogeneous real-time systems are special case of distributed systems, we focus on security-aware architecture in the context of distributed computing. The following architecture is based on capabilities of distributed systems for information management, security mechanisms for protection, security-aware scheduling mechanisms for security management, and real-time mechanisms for guaranteeing timing constraints. The architecture is designed for flexible real-time tasks models including the periodic and aperiodic task models and is intended to provide a general framework to meet user needs by integrating an array of security services on various scales. The architecture makes it possible to develop a wide range of security policies and mechanisms to protect heterogeneous real-time systems at all scales from deliberate intrusions and attacks.

Our security-aware architecture, as illustrated in Figure 1, is augmented with six major

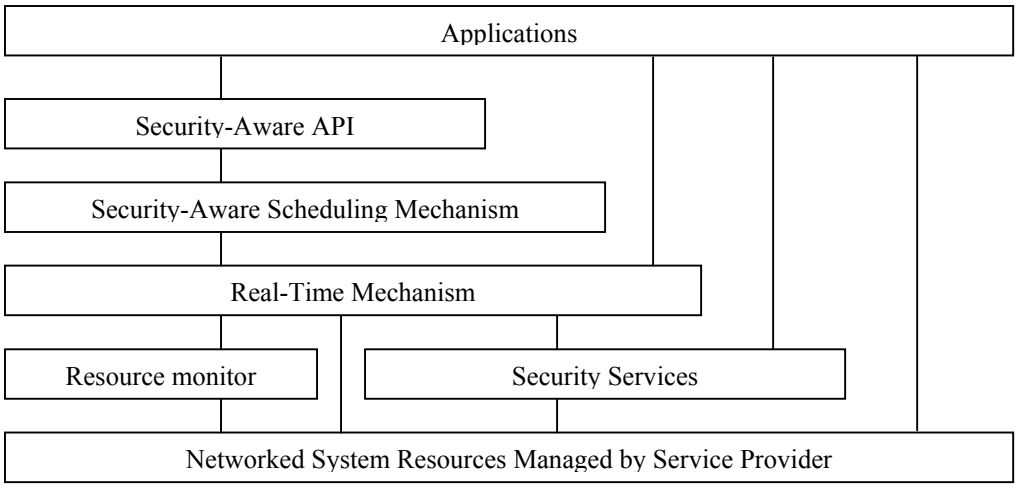

Figure 1. Security-aware architecture for distributed systems

components: security services, a security-aware scheduling mechanism, security-aware application programming interface (API), a resource management, a resource monitor, and a real-time mechanism.

In most cases, delivery of data in networked real-time systems has timing constraints. For example, a networked radar system can compare images of objects against a database of known aircraft type. To adequately support real-time applications, the real-time mechanism is responsible for managing networked system resources to guarantee timeliness, which is achieved by performing schedulability analysis. The underlying computing resources are dynamically monitored by a resource monitor, which provides the real-time mechanism a way of conducting feasibility assessment.

The security-aware scheduling mechanism is responsible for choosing the most appropriate security mechanism for each real-time task in a way to maximize resource utilization while guaranteeing timeliness. A security-aware API, which enables application programmers to implement security-aware real-time applications, is utilized to exchange security and performance related information among the security-aware scheduling mechanism and applications. The API provides the capability to specify security requirements imposed by applications.

The proposed architecture will provide practical framework and toolkit to seamlessly integrate security services with real-time services for distributed systems. Additionally, the framework will allow multiple security-aware scheduling algorithms to be efficiently designed for distributed computing environments.

3.2 Uniprocessor Real-time Scheduling with Security Awareness

We proposed a real-time security-aware scheduling algorithm, which is referred to as the earliest deadline first scheduling algorithm with optimized security-awareness, or *EDF_OPTS* for short. EDF_OPTS is intended to achieve a high quality of security for admitted tasks while maximizing guarantee ratio [29]. It is assumed that there is only one CPU or node in a real-time system. The cases where the real-time system consists of multiple nodes will be discussed shortly in the subsequent sections.

A real-time task is modelled by its arrival time, worst-case execution time, and deadline. It is assumed in our study that real-time tasks are indivisible, non-preemptive, and independent of one another. Note that the security overhead of the task is not factored in its worst-case execution time. We provide a simplified model to specify the security requirement of real-time tasks. Each task has a security level between 1 and R, where 1 and R are the lowest and highest security level, respectively. R is set to 10 in our simulation experiments. For the sake of simplicity, we model the security overhead of a real-time task as a linear function of its worst-case execution time. The rationale behind it is based on the observation that security

overhead of a particular application tends to be proportional to the execution time of the application and the size of data handled.

The EDF_OPTS algorithm is focused on optimizing the security levels of tasks in the priority queue. Specifically, EDF_OPTS strives to maximize the security levels of all the admitted tasks without violating real-time constraints. Under the circumstance where load is relatively high the EDF_OPTS algorithm will rejects real-time tasks whose deadlines are unable to be met. In contrast, real-time tasks that can be accomplished before their deadlines will be accepted by EDF_OPTS and placed in a priority queue, where priorities are assigned based on the tasks' deadlines in a way that tasks with the earliest deadlines can be executed first. There is a trade-off between guarantee ratio and quality of security. Generally speaking, a high guarantee ratio results in low security levels. This is because EDF_OPTS attempts to admit as many tasks as possible in a way of reducing security overhead and decreasing the security levels. The goal of the *EDF_OPTS* algorithm is to maximize the overall system performance, or *OSP,* which is defined as a product of guarantee ratio (*GR*) and security value (*SV*). The security value *SV* is calculated as a sum of security levels of all admitted tasks. Thus, the EDF_OPTS makes an effort to achieve a high guarantee ratio while optimizing security value. Specifically, EDF_OPTS maximizes the security level of an admitted task in the queue, subjecting the following two constraints. (1) Increasing a task's security level does not violate its deadline constraint. (2) The increment in the security level leads to no potential rejection of subsequent admitted real-time tasks.

To quantitatively evaluate the effectiveness of the proposed EDF_OPTS, we compared our EDF_OPTS algorithm against three baseline heuristic scheduling algorithms, namely, *EDF_MINS, EDF_MAXS, and EDF_RNDS*. The EDF_MINS algorithm selects the lowest security level of each admitted task. Therefore, EDF_MINS improves guarantee ratios at the cost of reducing overall security value. Conversely, EDF_MAXS chooses the highest security level for each accepted task. As a result, the security values are increased while decreasing guarantee ratio. Unlike EDF_MINS and EDF_MAXS, the EDF_RNDS algorithm randomly picks a value within the range of security levels specified by each task in the queue. Hence, the performance of EDF_RNDS is somewhere between that of EDF_MINS and EDF_MAXS.

The security values and guarantee ratios of four algorithms on various task arrival rates are respectively plotted in Figures 2 and 3. The execution time of each task is randomly chosen uniformly between 1 and 200 time units. Figure 2 clearly indicates that our EDF_OPTS consistently outperforms all the other three algorithms in security value when the arrival rate is larger than 0.75 Tasks/Time-Unit. Specifically, it is shown from Figure 2 that the security value of the proposed EDF_OPTS is 85.3%, 26.3%, and 16.6% higher than those of EDF_MINS, EDF_RNDS, and EDF_MAXS. In addition, the discrepancy in security value

between EDF_OPTS and other algorithms is more pronounced when the arrival rate increases. Interestingly, the results reveal that the security value performance of EDF_OPTS is better than that of EDF_MAXS. This is mainly because EDF_MAXS admitted some long tasks, which cause rejections of other short tasks. On the contrary, EDF_OPTS give higher priorities

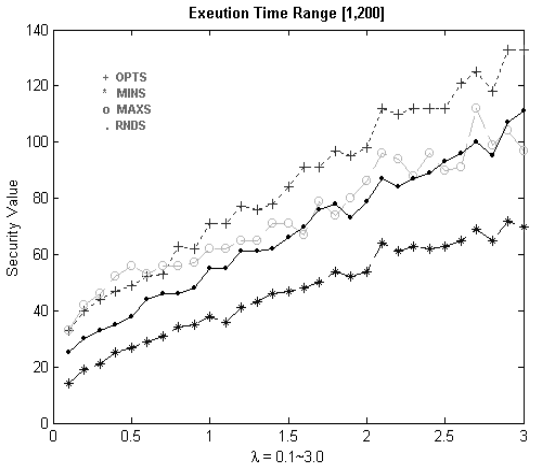

Figure 2. Security values

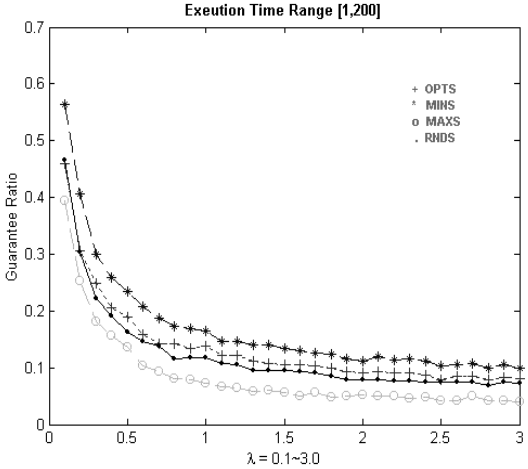

Figure 3. Guarantee ratios

to short tasks by enhancing the security levels of the short tasks, resulting in fewer rejections for other real-time tasks.

We can see from Figure 3 that when the arrival rate is 0.1 Tasks/Time-Unit, there are only few tasks submitted to the system and, thus, guarantee ratio is relatively high (from 40% to 57%). In contrast, when the system workload becomes extremely heavy, i.e., the arrival rate is 3.0 Tasks/Time-Unit, guarantee ratio becomes very low (from 5% to 25%). The guarantee ratio performance of EDF_OPTS is better than those of EDF_MAXS and EDF_RNDS. We attribute this result to the fact that EDF_MAXS and EDF_RNDS always select higher security levels for admitted tasks and, therefore, many real-time tasks are rejected due to the acceptance of tasks with high security levels. The guarantee ratio of EDF_OPTS is only 5% less than that of EDF_MINS, and such discrepancy between EDF_OPTS and EDF_MINS becomes less obvious with the increasing arrival rate. However, aforementioned results showed that EDF_OPTS improves the performance in security value over EDF_MINS by 85.3%. As such, EDF_OPTS delivers the *Overall System Performance* (*OSP*) improvement over EDF_MINS by an average of 50.6%.

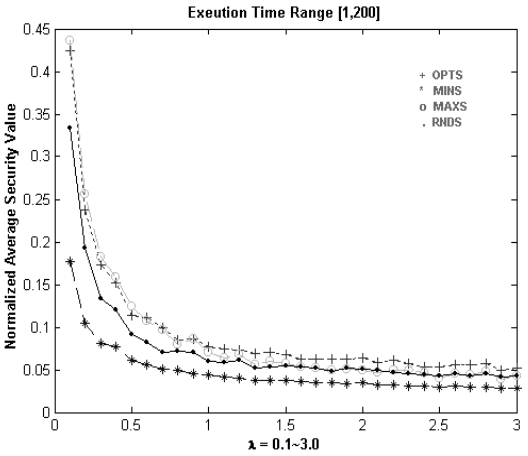

Figure 4. Normalized average security values of the EDF_OPTS algorithm and three baseline algorithms

Additionally, our EDF_OPTS improves the overall system performance (*OSP*) over EDF_MAX and EDF_RNDS by 90.7% and 36.7%, respectively.

We also conducted another group of experiments using normalized average security value, or *NASV,* rather than absolute security value as a performance metric. Before introducing NASV, we define a new measurement of security level referred to as normalized security level (*NSL)*. The *NSL* value of a task is defined as a ratio between its security value and the highest security level required by the task. *NASV* can be calculated as the sum of the *NSL* values of all admitted tasks divided by the total number of submitted tasks to the system. Figure 4 plots the normalized average security values of the EDF_OPTS algorithm and three baseline algorithms. Figure 4 shows that the performance of EDF_OPTS in terms of security is better than those of the other three baseline algorithms. This result is consistent with the results shown in Figure 2.

We are investigating some other performance metrics to better reflect the quality of security and therefore, better reflect the *Overall System Performance* of a security aware real-time system.

3.3 Security-aware real-time scheduling with a feedback control mechanism

Although the EDF_OPTS algorithm is capable of improving security value of a real-time system, there exist some limitations, one of which is that EDF_OPTS is an open-loop scheduling algorithm. While EDF_OPTS performs efficiently in static systems or dynamic systems with predictable workloads, it performs poorly in unpredictable dynamic environments, i.e., workloads are unable to be accurately modelled. In other words, EDF_OPTS is not adaptive in nature in the sense that it is not robust to sudden changes of system workloads.

Since EDF_OPTS fails in detecting sudden changes in workloads, it may continue increasing security levels of tasks even under the circumstances that system guarantee ratios are extremely low. The unawareness of such changes will in return worsen the guarantee ratios, implying that more submitted tasks are likely to be rejected. To tackle this problem, a feedback control mechanism for security-aware real-time scheduling can be developed. The

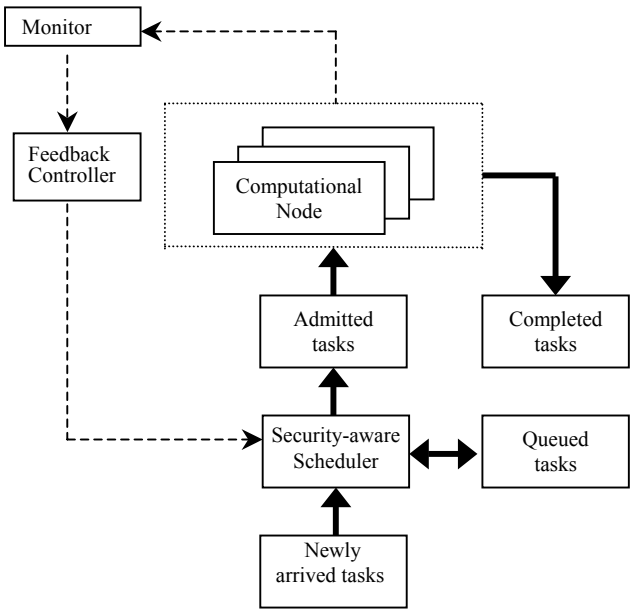

Figure 5. The feedback-control mechanism for security-aware real-time scheduling

feedback control mechanism can dynamically monitor system guarantee ratio. If the guarantee ratio is lower than a certain threshold, the mechanism will make an effort to increase the guarantee ratio by loosening security levels of currently admitted tasks, thereby accommodating more future incoming tasks.

The feedback control mechanism for security-aware real-time scheduling, or FCEDF_OPTS, is outlined in Figure 5. The mechanism consists of a security-aware scheduler, a feedback controller, a monitor, and computational nodes. The scheduler behaves as an actuator to control job admissions. For a system where real-time and security constraints both have to be factored in, accepting more jobs to the system may become counterproductive with respect to the quality of security. Thus, the feedback controller makes the best effort to dynamically maximize both the quality of security and the resource utilization and by accepting the most appropriate tasks to run in the system.

3.4 Discussions: quantitative measurement of security overheads

With a security overhead model in place, schedulers can be enabled to be aware of security overheads, thereby incorporating the overheads into the process of scheduling tasks. In previous study, we modelled the security overhead of a task as a linear function of its execution time. To make security-aware scheduling algorithms practical, we need to investigate and propose an accurate mathematical model to quantitatively measure security overheads. Unfortunately, less attention has been paid to the mathematical model used to measure security overheads imposed by tasks' security requirements. Therefore, it is desirable to develop an effective model, which is capable of approximately or reasonably measuring overheads of a collection of widely used security techniques. To achieve this goal, we can, as the first step in this study, conduct extensive experiments to glean some understanding of how much extra execution time imposed by a particular security mechanism applied to real-time applications. The second step is then to build a mathematical model used to quantitatively predict security overheads related to various security levels. Note that a particular security level consists of a collection of security techniques, i.e., SSL, digital signature, encryption and decryption methods. The accomplishment of this study will provide a solid foundation, which is expected to make security-aware real-time scheduling algorithms more practical.

3.5 Security-aware real-time scheduling for homogeneous distributed systems

A homogeneous distributed system is comprised of a group of identical computers or servers loosely connected through a network and distributed middleware, which allow computers to coordinate their computation tasks and to share computing resources in a way that users perceive the system as a integrated computing platform. We are developing a security-aware real-time scheduling for distributed systems, where dynamic scheduling

algorithms are implemented as a middleware service processing requests with security and timing constraints between clients and servers.

In a distributed system, a centralized server could be used to handle the issue of real-time scheduling. However, when the centralized server becomes increasingly overloaded with the growth of system size, it will inevitably become a severe bottleneck, making the distributed system suffer a significant performance drop. To effectively alleviate the potential burden of the centralized server, our security-aware real-time scheduler, or SARED, is distributed in nature in the sense that the scheduling workload can be evenly distributed among other servers.

Application servers running on distributed systems specify an array of services by registering with security and real-time binding services to which clients make requests. The security and real-time binding services along with the security-aware real-time scheduling mechanism are responsible of the binding the most appropriate application services to clients requests to achieve high quality of security and meet timing constraints.

When client task are submitted to the distributed system, the tasks specify the quality of service including security levels and deadlines. The security-aware real-time scheduling algorithm dynamically assigns security levels to tasks running in the distributed system. As discussed in prior sections, tasks with high security levels tend to experience high CPU overhead and cost of other resources. It is noted that the security overhead of a particular security level on all the servers are identical, indicating that we can apply the same security overhead model to an array of servers in the distributed systems.

Besides deciding the security levels for real-time tasks, the scheduler is able to improve utilization of resources in distributed systems by sharing load among servers. Upon the arrival of a real-time task to a server in the system, the scheduler assigns the task to the local server if its load is not higher than a certain threshold. If the local server is overloaded, the scheduler will migrate the task to a remote server with the lightest load in the system. The local server monitors the resource utilization of the system by periodically receiving load information from other peer servers. Similarly, the local server repeatedly broadcasts its load information to other servers.

3.6 Security-aware real-time scheduling for heterogeneous distributed systems

Recent studies in the area of heterogeneous systems showed that there exist a number of challenges to be accommodated within a short span of time. One challenge is dynamic real-time scheduling for parallel tasks running in heterogeneous systems. Scheduling of parallel jobs is one of the key factors in achieving high performance in heterogeneous systems [9][19][28]. The objective of real-time scheduling is to map tasks onto multiple nodes of a system in a way that task precedence constraints are satisfied and a minimal schedule length is

generated [17]. Many scheduling schemes have been introduced for parallel computing systems [5]. Some of them assume precedence requirements among tasks being scheduled, and use directed acyclic graph (*DAG*) to model such constraints [7][14]. However, most of existing algorithms did not consider tasks' security requirements, which are essential for security-critical real-time applications running in a distributed system as heterogeneous as the Grid.

A potential solution to tackle the above problem is to develop a security-aware real-time scheduling algorithm, or *SAREH*, which is capable of dynamically scheduling parallel jobs submitted to a heterogeneous distributed system. Specifically, a scheduler running on a dedicated node is in charge of scheduling jobs and dispatching them to other nodes to execute. The SAREH algorithm has to take into account dispatch times, schedule times, and most importantly, security overheads experienced by submitted jobs with various security requirements. The SAREH scheduling algorithm will decide start times and security levels for tasks within a parallel job in addition to finding a proper group of nodes for the job.

The objective of the scheduling algorithm is twofold: enhance security values and minimize schedule lengths. To achieve high quality of security, our scheduling algorithm aims to dispatch tasks with higher security demands to more secure nodes with minimal security overheads. The quality of security of parallel jobs can be further improved by reducing data communication, because networks are insecure in nature for parallel jobs running in distributed systems.

Table 1. Summary of Representative Scheduling Algorithms and Their Features

Features / Algorithms	Open Loop/ Closed Loop	Static/ Dynamic	Real-time	Homogeneous / Heterogeneous	Security-aware
CTSH [23]	Open	Static	No	Uniprocessor	No
RM [11]	Open	Static	Yes	Uniprocessor	No
EDF [25]	Open	Dynamic	Yes	Uniprocessor	No
Spring [20][30]	Open	Dynamic	Yes	Uniprocessor	No
FC_EDF [12]	Close	Dynamic	Yes	Uniprocessor	No
SQFCFS [8]	Open	Dynamic	No	Homogeneous	No
HEFT [28]	Open	Dynamic	Yes	Heterogeneous	No
EDF_OPTS [29]	Open	Dynamic	Yes	Uniprocessor	Yes
FCEDF_OPTS	Close	Dynamic	Yes	Uniprocessor	Yes
SARED	Close	Dynamic	Yes	Homogeneous	Yes
SAREH	Close	Dynamic	Yes	Heterogeneous	Yes

A second advantage of reducing communication overheads among tasks is to efficiently decrease schedule lengths. Communication overhead incurred by data transmission between two tasks allocated on the same node is negligible, because this can be performed through local memory with zero time cost. Our scheduling algorithm strives to reduce communication

overheads by allocating tasks with high communication requirements on the same node while assigning the heaviest inter-task communication to the most secure network links. As such, our SAREH scheduling algorithm is designed to make it possible to enhance schedulability and quality of security of heterogeneous distributed systems with no extra hardware cost.

To enforce security and real-time guarantees for a diverse set of parallel jobs in a heterogeneous system, the SAREH algorithm incorporates the feedback control architecture as described in Section 3.3 to provide performance guarantees to a wide variety of applications while achieving high utilization of system resources shared by these applications executing in a dynamic heterogeneous computing environment.

We summarize in Table 1 the most relevant scheduling algorithms described in the literature. It is noted from Table 1 that SAREH differs from the existing algorithms in that it is a closed-loop, dynamic, real-time, security-aware algorithm designed for heterogeneous distributed systems.

4. Conclusions

This paper identified open issues involved in providing both security and real-time guarantees to real-time applications with security requirements. We proposed five approaches to achieving high quality of security and schedulability: a security aware architecture, the uniprocessor real-time scheduling algorithm with security awareness (EDF_OPTS), the feedback control mechanism, security-aware real-time scheduling for homogeneous and heterogeneous distributed systems.

We have conducted extensive simulation experiments to show that our EDF_OPTS algorithm can consistently improve system performance in terms of quality of security and guarantee ratios over three baseline algorithms. In particular, EDF_OPTS delivers the overall system performance improvement over the three baseline algorithms by 90.7%, 50.6%, and 36.7%, respectively. We can further improve the performance of EDF_OPTS by incorporating the feedback control mechanism into the scheduling framework.

Future studies in this research can be performed in several directions. First, a systematic security overhead model for real-time tasks will be developed. Second, the feedback control mechanism will be developed to dynamically monitor and adjust security levels and guarantee ratios. It is interesting to evaluate the performance of the feedback control mechanism in a large-scale distributed system. Finally, The SARED and SAREH algorithms will be implemented in the grid environment.

Acknowledgements

The work reported in this paper was supported in part by Intel Corporation under Grant 2005-04-070 and by the New Mexico Institute of Mining and Technology under Grant 103295.

References

[1] M. Bishop, "Computer Security: Art and Science," *Addison Wesley Professional*, ISBN 0201440997; Published: Dec 2, 2002.

[2] A. Dogan, F. Ozguner, "Reliable matching and scheduling of precedence-constrained tasks in heterogeneous distributed computing," *Proc. Int'l Conf. on Parallel Processing*, pp. 307-314, 2000.

[3] G. Donoho, "Building a Web Service to Provide Real-Time Stock Quotes," *MCAD.Net*, February, 2004.

[4] W. E. Faller, S. J. Schreck, "Real-time prediction of unsteady aerodynamics: Application for aircraft control and manoeuvrability enhancement," *IEEE Transactions on Neural Networks*, Vol. 6 , No. 6 , pp 1461 – 1468, Nov. 1995.

[5] D. G. Feitelson, L. Rudolph, "Job Scheduling for Parallel Supercomputers," *In Encyclopedia of Computer Science and Technology*, Vol. 38, New York, 1998.

[6] E.N. Huh, L.R. Welch, B.A. Shirazi and C.D. Cavanaugh, "Heterogeneous Resource Management for Dynamic Real-Time Systems," *In Proc. of the 9th Heterogeneous Computing Workshop*, 2000, 287-296.

[7] M. Iverson and F. Ozguner, "Dynamic, Competitive Scheduling of Multiple DAGs in a Distributed Heterogeneous Environment," *In Proc. of the Seventh Heterogeneous Computing Workshop*, pp.70-78, Orlando, Florida, USA, 1998.

[8] H. D. Karatza, R. C. Hilzer, "Parallel Job Scheduling in Homogeneous Distributed Systems," *SIMULATION: Transactions of the Society for Modeling and Simulation*, vol. 79, issue 5, pp. 287-298(12), May 2003.

[9] D. Kebbal, E.G. Talbi, J.M. Geib, "Building and Scheduling Parallel Adaptive Applications in Heterogeneous Environments," *1st IEEE Computer Society International Workshop on Cluster Computing*, Melbourne, Australia, December 02 - 03, 1999.

[10] J.-C. Laprie. "Dependable Computing: Concepts, Limits, Challenges". *Special Issue of the 25th International Symposium On Fault-Tolerant Computing,* pp. 42-54, IEEE Computer Society Press. Pasadena, CA. 1995.

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

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

[13] B. Mahafza, S. Welstead, D. Champagne, R. Manadhar, T. Worthington, and S. Campbell, "Real-time radar signal simulation for the ground based radar for national missile defense," *Proc. the 1998 IEEE Radar Conference*, pp 62 – 67, May 1998.

[14] M. Maheswaran and H. J. Siegel, "A Dynamic Matching and Scheduling Algorithm for Heterogeneous Computing Systems," *Proc. the 7th Heterogeneous Computing Workshop*, pp.57-69, 1998.

[15] J. Nilsson and F. Dahlgren, "Improving performance of load-store sequences for transaction processing workloads on multiprocessors," *Proc. International Conference on Parallel Processing*, pp. 246-255, 21-24 Sept. 1999.

[16] 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.

[17] X. Qin, H. Jiang, "Dynamic, Reliability-driven Scheduling of Parallel Real-time Jobs in Heterogeneous Systems," *In the Proceedings of the 30th International Conference on Parallel Processing (ICPP 2001),* pp.113-122, Valencia, Spain, September 3-7, 2001

[18] X. Qin, H. Jiang, C. Xie, and Z. Han, "Reliability-driven scheduling for real-time tasks with precedence constraints in heterogeneous distributed systems," *Proceedings of the International Conference Parallel and Distributed Computing and Systems 2000,* November 6-9, 2000.

[19] A. Radulescu, Arjan J.C. van Gemund, "Fast and Effective Task Scheduling in Heterogeneous Systems," *In Proc. of the 12th Euromicro conferences on Real-time Systems*, pp229-238, 2000

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

[21] S. Ranaweera, and D.P. Agrawal, "Scheduling of Periodic Time Critical Applications for Pipelined Execution on Heterogeneous Systems," *Proc. Int'l Conf. on Parallel Processing*, pp. 131-138, Sept. 2001.

[22] R. M. Santos, J. Santos, and J. Orozco, "Scheduling heterogeneous multimedia servers: different QoS for hard, soft and non real-time clients," Proc. *Euromicro Conf. on Real-Time Systems*, pp.247-253, 2000.

[23] G. C. Sih and E. A. Lee, "A Compile-Time Scheduling heuristic for Interconnection-Constrained Heterogeneous Machine Architectures," *IEEE Trans. Parallel and Distributed Systems*, 4(2), pp.175-187, 1993.

[24] S. Srinivasan, and N. K. Jha, "Safty and Reliability Driven Task Allocation in Distributed Systems," *IEEE Trans. Parallel and Distributed Systems*, 10(3), pp. 238-251, 1999.

[25] J. A. Stankovic, M. Spuri, K. Ramamritham, G.C. buttazzo, "Deadline Scheduling for Real-Time Systems – EDF and Related Algorithms*," Kluwer Academic Publishers*, 1998.

[26] S. Suzuki, T. Katane, H. Saotome, O. Saito, **"**Electric power-generating system using magnetic coupling for deeply implanted medical electronic devices," *IEEE Transactions on Magnetics*, Vol. 38 , No. 5, pp. 3006 – 3008, Sept. 2002.

[27] A. Tanenbaum, "Distributed systems: principles and paradigms," *Prentice Hall*, ISSN/ISBN: 0-13-088893-1, 2001

[28] H. Topcuoglu, S. Hariri, M.-Y. Wu, "Task Scheduling Algorithms for Heterogeneous Processors," *In Proc. of 8th Heterogeneous Computing Workshop*, pp.3-14, 1999.

[29] T. Xie, A. Sung, and X. Qin, "Dynamic Task Scheduling with Security Awareness in Real-Time Systems," Proc. of the 19th Int'l Parallel and Distributed Processing Symp., Int'l Workshop on Performance Modeling, Evaluation, and Optimization of Parallel and Distributed Systems, IEEE/ACM, April 4-8, 2005.

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

[31] Y. Zhang and A. Sivasubramaniam, "Scheduling Best-Effort and Real-Time Pipeline Application on Time-Shared Clusters," *Proc. Int'l Symp Parallel Architecture and Algorithm*, 2001.