Integrating Fault Recovery and Quality of Security in Real-time Systems

Kiranmai Bellam, Ziliang Zong, Mohammed .I. Alghamdi, Mais Nijim and Xiao Qin*
Department of Computer Science
New Mexico Institute of Mining and Technology
Socorro, NM 87801
{kiran, zzong, alghamdi, mais, xqin, }@cs.nmt.edu

Abstract

In the past five years, mandatory security requirements and fault tolerance have become critical criteria for most real-time systems. Although many conventional fault-tolerant or security approaches were investigated and applied to real-time systems, most existing schemes only addressed either security demands ignoring the fault-tolerant requirements or vice versa. To bridge this technology gap in real-time systems, in this paper we propose a way of integrating fault recovery and confidentiality services. The novel integration of security and fault recovery makes it possible to implement next-generation real-time systems with high reliability and quality of security. Experimental results from real-world applications show that our approach can significantly improve security over the conventional approaches by up to 661.56% while providing an efficient means of fault tolerance.

1. Introduction

Completing tasks before their deadlines is not the only consideration in the arena of today's real-time systems because there are many real-time applications like online trading and banking systems where security requirements [7] along with fault tolerance [2] are very crucial. Our work mainly addresses the seamless integration of quality of security services and fault tolerance in the context of soft real-time systems.

Confidentiality is the security services considered in this study to ensure the required security requirements. It is to be noted that the confidentiality services are used to counter unauthorized interception. With the advancement of cryptographic algorithms, the confidentiality services become widely available.

Checkpoints are inserted into real-time tasks to recover real-time systems from software or hardware failures that occur during the course of tasks' execution. To facilitate the functionality of fault tolerance, slack times existing in schedules are used to restart the whole or part of failed tasks.

In our research, we are focused on the integration of fault recovery and security to achieve maximized overall performance of real-time tasks. Here we should note that deadlines for all the tasks are soft, thus our algorithms is suitable for the soft real-time systems. From the perspective of real-time tasks, our approach is always trying to complete tasks in a timely manner while achieving higher security and fault tolerance level. From the system's standing point, a more reliable and secure real-time system can be efficiently built by incorporating security services into fault tolerance strategies.

To clearly evaluate the performance of the proposed approach in integrating security and fault-tolerance, we investigated a checkpoint placement strategy with confidentiality.

In this study we investigate a feasible way to determine an optimal number of checkpoints so that minimum reliability level is guaranteed. We develop two optimal algorithms to implementation the integration of security and fault tolerance.

The experimental values that we observed are compared to the conventional approaches (where we assumed that minimum security is guaranteed) it has proven that our approach is up to 661.56% more secure while achieving the high reliability standards. The approach we presented in this paper is novel and is very useful for the tasks where security is crucial along with reliability.

The rest of the paper is organized as follows. In

* Corresponding author. *http://www.cs.nmt.edu/~xqin*

21st International Conference on
Advanced Information Networking and Applications Workshops (AINAW'07)
0-7695-2847-3/07 $20.00 © 2007 IEEE
Authorized licensed use limited to: Auburn University. Downloaded on November 17,2026 at 22:48:31 UTC from IEEE Xplore. Restrictions apply.

section 2, the brief outlines of the similar research done in this area are described. Section 3 describes the distribution of checkpoints to achieve fault tolerance. It details the skeleton of a real-time application and the fault tolerance model. Section 4 explains the implementation of security mechanism by briefing the confidentiality model. Section 5 addresses the combination of Fault tolerant model and security mechanism. Summary and future work will be discussed in section 6.

2. Related Work

Much attention has been paid towards fault- tolerant real-time scheduling. Most of conventional real-time scheduling algorithms used timeline or backup approaches to providing fault tolerance [4]. Such approaches with low overhead are adequate for real-time systems when storage is not a primary concern. During transient faults, re-executions of tasks are usually performed to recover faults [3][6].

Checkpoints along with log management are used to design fault-tolerant distributed shared memory systems [5]. Sultan *et al.* developed management algorithms that are efficient in bounding checkpoints and logs [5].

A variety of fault-tolerant activities are incorporated in both static and dynamic scheduling without impairing the feasibility of pre-guaranteed tasks and minimizing the number of reliability activities [7]. In addition, a method – slot shifting – was used to seamlessly integrate static and dynamic scheduling. Uniform and non-uniform check-pointing policies were developed to recover from failures and reduce power consumption in real-time systems [2]. We leverage a similar approach to supporting fault tolerance. However, our work is fundamentally different from the previous studies in the sense that ours addresses the issue of improving security without adversely affecting reliability whereas the previous studies were focused on improving reliability of real-time systems.

Song *et al.* developed security driven scheduling algorithms for Grids [1]. In our previous work, we designed and implemented security-aware real-time scheduling algorithms, which make use of three types of security services to guard real-time systems from potential threats [7]. In this previous study, we investigated the integration of security services to protect against a diversity of threats and attacks in cluster computing environment.

Bertram *et al.* proposed a set of algorithms for security-constrained optimal power flow (SCOPF) and their configuration in an integrated package for real-time security enhancements [8]. An introduction of non-convex SCOPF methods for modelling utility operating policy is also given. The study discussed here relies on a security overhead model that we recently presented in [7], which depicts the overhead model for an array of security services.

3. Checkpoints for Fault Recovery

Real-time applications can be generally classified into two types: hard real-time applications and soft real-time applications. In hard real-time applications, specified deadlines have to be guaranteed, whereas deadlines in soft real-time applications are not extremely restrictive. In this research we considered soft deadlines in real-time applications. However, there is no means to restrict our approach to soft real-time applications. The basic idea behind our approach is to recover soft real-time systems from transient failures by inserting checkpoints. It should be noted that checkpoint insertions can be accomplished in a uniform or non-uniform manner.

3.1 Real-time applications

A real-time application in this study is envisioned as a set of tasks, which have to be completed before their specified deadlines to accomplish an overall mission. We assume a real-time task t has a deadline d derived from hard real-time constraints. It is assumed that task t takes C number of CPU cycles for its worst-case execution. Cycle consumption of memory references varies with processor speed. Hence C is considered as the independent of the processor speed. Abundant research has been done to prove the degree of pessimism in the definition of C [2].

The time remaining to reach deadline d after considering the worst-case execution time C of the real-time task is the slack time denoted as S_l. This slack time S_l can be used to recover from transient faults in a real-time system. Note that slacks may be static or dynamic. Specifically, a static slack time is obtained when deadlines and worst-case execution times are known a priori. In contrast, dynamic slack times can be derived only at runtime executions. Slack times considered in this work are static slacks (here after referred to as slacks).

21st International Conference on
Advanced Information Networking and Applications Workshops (AINAW'07)
0-7695-2847-3/07 $20.00 © 2007 IEEE
Authorized licensed use limited to: Auburn University. Downloaded on November 17,2026 at 22:48:31 UTC from IEEE Xplore. Restrictions apply.
IEEE COMPUTER SOCIETY

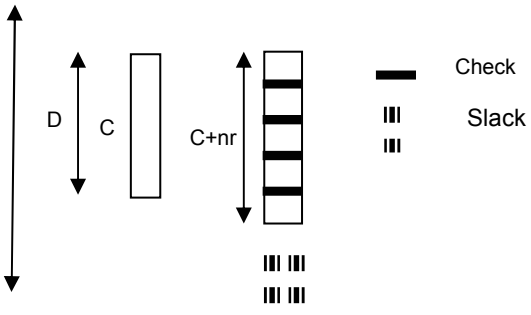


Fig. 1 Fault tolerant model

3.2 Fault tolerant model

Recall that checkpoints are inserted in a real-time task to efficiently recover a system from any transient failure occurred during the execution of a task. If there is a failure in the execution of the task then the failure is detected before the next checkpoint is placed. Hence it makes easy to re-execute the task from the previous checkpoint, where the failure occurred. Transient failures during the course of the task's execution can be readily detected by applying existing fault-detection techniques. Without loss of generality, it is assumed that no more than one transient failure might occur before the execution of the task. Similar assumptions were imposed in other studies [2]. The process we describe here (see Fig. 1.) may not be applicable to a real-time system where a real-time task must be re-executed entirely when a failure occurs. This is because available slack S_l may not be sufficient to accommodate such a re-execution.

Our goal, of course, is to efficiently leverage slack time S_l to improve quality of security and to provide fault recovery. The overhead involved with generating checkpoints is modeled as *r*. It is assumed that in task *t, n* checkpoints are inserted to recover the system from failures if there exists any. Therefore, the total time spent in creating all the checkpoints and to do the corresponding diagnosis is expressed as *n*r*. If there exists a fault, then it takes (C/n+1) time for a potential roll back. In a previous study Melhem *et al.* showed that it takes *(C/N)* time for a potential rollback to recover from a fault if there exists any [2]. They inserted the first checkpoint of *n* at the very start of the execution of the task. We argue that there is no significant use in inserting a checkpoint at the very beginning of the task; therefore, we considered another more effective strategy. Thus, the time spent in processing a potential roll back in our research is written as *(C/n+1)*. We can easily prove the correctness of this expression by showing that there can never be a failure before the execution of a task. This argument indicates that the strategy we used does not insert checkpoints at the beginning of each task. Hence a task *t* should satisfy the below equation, to execute before the deadline and to guarantee the fault tolerance.

$$d = C + nr + (C/(n+1)) \quad (1)$$

It is intuitive to illustrate that the more the number of checkpoints made in a real-time task the more the task is fault tolerant. However, it is imperative to carefully choose the number of checkpoints, because there is a chance that the overhead might exceed slack time S_l, which makes it infeasible to finish the task before its deadline.

Simple mathematical manipulation of inequality (1) yields

$$n = x \pm \sqrt{(x^2 - 4r(2c - D))} / 2r \quad (2)$$

Where $x = d - c - r$. (3)

Using the Eq. (2) we can calculate the optimal value of *n*. In our experiments, we made use of Eq. (2) to optimize *n* resulting maximal reliability of a real-time task while guaranteeing the task to be completed before its deadline.

4. Security Implementation

Security has become an important requirement for modern real-time systems. There are a wide variety of security mechanisms that can be applied to real-time applications. However, improving security inevitably leads to high overheads. Thus it is difficult if not impossible to implement all security mechanisms available to a real-time task. Therefore, vital security mechanisms have to be carefully chosen to provide sufficient level of security for real-time tasks.As confidentiality is a major security issue that real-time tasks may encounter we described the overhead model for this security service to integrate in our approach. It is worth noting that our approach can readily incorporate other types of security services like Integrity and Availability to achieve higher security.

4.1. Confidentiality Model

Confidentiality counters snooping, which is unauthorized interception. Confidentiality is achieved by the implementation of cryptographic algorithms. All the data of a real-time application, which has to be protected from unauthorized interceptions, must be enciphered in one way or the other. A wide range of cryptographic algorithms are available. The best candidate cryptographic algorithm, of course, is the one consuming less time while ensuring maximal security. There are a rich set of cryptographic algorithms that can be used for confidentiality; the most common ones

21st International Conference on
Advanced Information Networking and Applications Workshops (AINAW'07)
0-7695-2847-3/07 $20.00 © 2007 IEEE
Authorized licensed use limited to: Auburn University. Downloaded on November 17,2026 at 22:48:31 UTC from IEEE Xplore. Restrictions apply.
IEEE COMPUTER SOCIETY

are used in our research as shown in Table 1(Please refer to [7] for more details). Sl_i signifies the security level assigned to each algorithm in the range from 0.08 to 1. The strongest encryption algorithm IDEA is assigned the maximum-security level 1 and its performance is 13.5KB/ms.The security levels for the other algorithms are normalized by applying the following equation [7].

$$sl_i^c = 13.5 / \mu_i^c, 1 \le i \le 7 \quad (4)$$

Table1: Cryptographic algorithms for confidentiality

Cryptographic Algorithms	Security Level (sl_i^c)	μ_i^c (KB/ms)
SEAL	0.08	168.75
RC4	0.14	96.43
Blowfish	0.36	37.50
Knufu/Khafre	0.40	33.75
RC5	0.46	29.35
Rijndael	0.64	21.09
IDEA	1.00	13.5

Assume that l_i is the amount of data whose confidentiality has to be guaranteed for task t_i.

$$l_i / \mu_i^c = P \quad (5)$$

Where P is the amount of time the corresponding algorithm of μ^c_i takes to encrypt the data with size l_i. Hence the appropriate cryptographic algorithm can be selected based on the amount of time that can be allocated for the security mechanisms.

For a task to execute before its deadline while achieving the confidentiality, the task should satisfy the following condition.

$$d = C + (l_i^c / \mu_i^c) \quad (6)$$

The above inequality can be effectively used when the confidentiality of a task has to be guaranteed along with completing the task before the deadline.

5. Integration of Fault Recovery with Security Mechanisms

In the previous sections, we have described a way to implement security mechanism and also fault tolerance model along with their overheads. In this section, we illustrate how to integrate fault tolerant model with the security mechanisms to enhance the overall performance of real-time tasks.

Given a real-time task with its timing constraint, the aggregate of the overheads of the fault tolerant model and security mechanism implementation should be less than or equal to the slack S_l of that task. The implementation of fault recovery in addition to security mechanisms is possible only if the following inequality is satisfied

$$d = C + nr + (C/(n+1)) + (l_i^c / \mu_i^c) \quad (7)$$

The above expression guarantees both reliability and confidentiality for real-time tasks.

Simple manipulation of (7) leads to

$$\mu_i^c = \frac{l_i^c}{d - C - nr - (C/(n+1))} \quad (8)$$

Using the above formula we can straightforwardly determine an optimal security level for a real-time task while guaranteeing high reliability through fault recovery without violating the task's deadline. By mapping the value of μ_i^c to the corresponding encryption algorithm the feasible security can be implemented.

Equation (8) can be used to identify the most appropriate security levels for real-time tasks. To use the above formula, one has to predetermine n. Consequently, we have the flexibility to choose the level of reliability. If a real-time task is security critical, then we can choose a smaller value for n, which in turn results in a long slack time for high security. On the other hand, if the real-time task is not security crucial, then the task can have a high level of reliability. We can choose the minimal security level for the task in order to create a maximized number of checkpoints to improve reliability without missing the deadline. The equations below is used to handle this special case of increasing reliability

$$n = \frac{(x-r) \pm (\sqrt{(x-r)^2 - (4r(C-x))}}{2r} \quad (9)$$

Where $x = d - C - (l_i^c / \mu_i^c)$ (10)

Using Eq. (9) we can increase the reliability to the maximum possible extent provided that security levels are known a priori based on the security requirement of real-time tasks. Then the value of n can be calculated. Here the reliability is maximized while achieving the minimum security without crossing the deadline.

Now we are positioned to address the issue of having both security and reliability optimized. More specifically, the security and reliability of a real-time task can be simultaneously optimized by choosing the optimal value of security level and determining the optimal value of the number n of checkpoints. An alternative approach to achieving this goal is to first calculate the optimal number n of checkpoints followed by deciding the optimal value of security level. In

21st International Conference on
Advanced Information Networking and Applications Workshops (AINAW'07)
0-7695-2847-3/07 $20.00 © 2007 IEEE
Authorized licensed use limited to: Auburn University. Downloaded on November 17,2026 at 22:48:31 UTC from IEEE Xplore. Restrictions apply.
IEEE COMPUTER SOCIETY

doing so, both security and reliability can be efficiently optimized.

6. Performance Evaluation

In this section we evaluate the effectiveness of the proposed approach of integrating fault recovery (through checkpoints) and quality of security (measured as security levels). To demonstratively show the strength of our scheme, we conducted extensive experiments. It is to be noted that worst-case execution times are derived from real-world applications.

6.1 Confidentiality versus reliability

In Fig.2(a), the change in the security levels with the increase of checkpoints is observed for three different data sizes of 100MB, 150MB, 200MB, respectively. The results indicate that for a given task, initially when the number of checkpoints increases the security level also increases because there is a sufficient slack time available to accommodate the increasing security level and the number of check points. Once the slack is fully utilized the security level remains constant up to a certain number of checkpoint increase. Now a further increase of checkpoints with no slack available forces the security level to be compromised. As a result the security level decreases.

Fig. 2(b) reveals the relationship between the rho(C/D) and security levels. We observe from Fig. 2(b) that the higher the amount of Rho, the less the amount of slack time is left. Fig. 2(b) shows the impact of decreasing slack time on the security level. As expected the security levels of the tasks decreases as the slack decreases. After a certain period even the minimum security can not be achieved because of the limited slack.

In general, a task which has a prolonged deadline can be made more secure than other tasks with shorter deadlines. The same phenomenon is observed for the three different tasks with data size of100MB, 150MB, 250MB. For any given task, as the deadline increases the slack required to accommodate security level also increases; hence, the security level increases. The graph proves the correctness of the above argument.

It can be observed from Fig. 2(c) that the security level can be increased or decreased by carefully choosing the number of checkpoints. Thus, if a task is very security critical, the minimum number of checkpoints can be chosen which still guarantees the basic reliability requirement. However, if security is not a major concern, the number of checkpoints can be increased for better fault recovery, which in turn leads to a higher level of reliability.

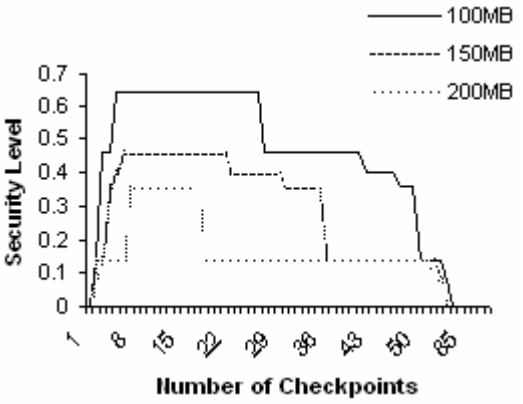


Fig.2(a) Number of checkpoints v.s security level

6.2 Security versus reliability

The Fig.3 shows the relation between reliability and security. Fig.3 shows results for three different data sizes. For the smaller data set, the levels of security and reliability are high because of large available slack times. As the data size increases the security and reliability levels are reduced. Again, we observe that higher security levels lead to low reliability with small number of checkpoints. Basic security levels give rise to high reliability with a large number of checkpoints. The optimal values for both security and reliability are the points where security level and number of checkpoints are equal. By using the Figure, one can derive the optimum values for the number of checkpoints and security levels.

7. Summary and Future Work

In this paper we studied an approach to integrating fault recovery and adaptive control of quality of security. In the proposed scheme we considered fault recovery and a security service - confidentiality. This research was undertaken mainly in the following three steps. First, the overhead models for the security service and for the fault recovery model were presented. Next, we integrated fault recovery mechanism with adaptive quality of security. To facilitate the presentation of the integration, we introduce the mathematical notations. Third, we proposed a novel way of maximizing quality of security with minimum reliability. Additionally, we described a means of optimizing reliability while providing basic quality of security. Our approach aims to maximize the security of a real-time system while providing checkpoints to efficiently support fault tolerance.

21st International Conference on
Advanced Information Networking and Applications Workshops (AINAW'07)
0-7695-2847-3/07 $20.00 © 2007 IEEE
Authorized licensed use limited to: Auburn University. Downloaded on November 17,2026 at 22:48:31 UTC from IEEE Xplore. Restrictions apply.
IEEE COMPUTER SOCIETY

Experimental results show that compared to conventional schemes where basic security is provided our approach improves the quality of security by up to 661.56%.

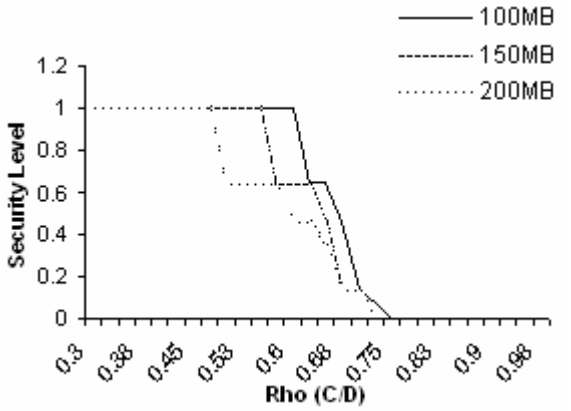


Fig.2(b) Rho V.S security level

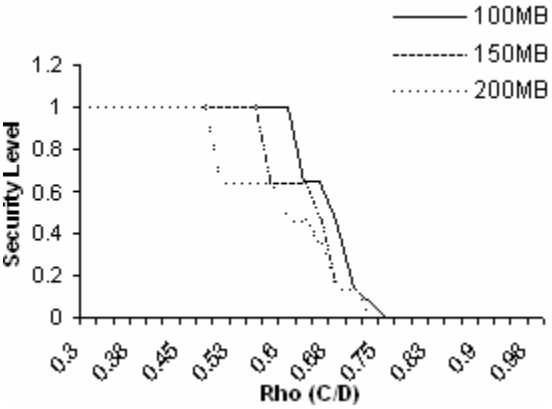


Fig.2(c) Deadline V.S Security level

Future research directions are discussed below. First, we will extend this work by including the placement of non-uniform checkpoints strategy for fault recovery. Second, we will study a way of integrating fault recovery with security for periodic real-time tasks. Last but not least we will consider a combination of multiple security mechanisms to further protect real-time systems from a variety of potential threats.

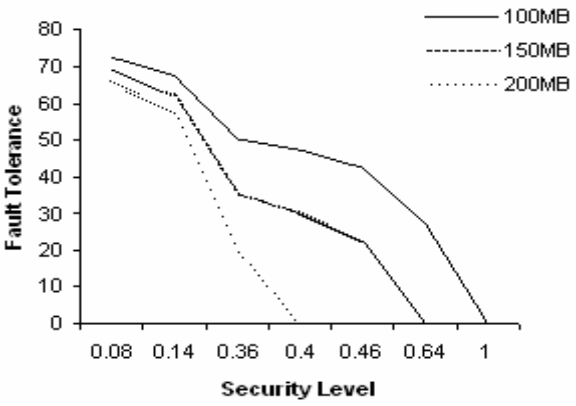


Fig.3 Security level V.S fault tolerance

Acknowledgements

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

8. References

[1] S. Song, Y. K. Kwok, and K. Hwang, "Trusted Job Scheduling in Open Computational Grids: Security-Driven Heuristics and A Fast Genetic Algorithms," Proc. Int'l Symp. Parallel and Distributed Processing, 2005

[2] Rami Melhem, Daniel Mosse, Elmootazbellah (Mootaz) Elnozahy, "The Interplay of Power Management and Fault Recovery in Real-Time Systems," IEEE Transactions on Computers, vol.53, no.2, pp. 217-231, Feb.2004.

[3] S. Ghosh, D. Mosse, and R. Melhem, "Implementation and Analysis of a Fault-Tolerant Scheduling Algorithm," IEEE Trans. Parallel and Distributed Systems, vol. 8, no. 3, Mar. 1997.

[4] Liestman and R. Campbell, "A Fault-Tolerant Scheduling Problem," IEEE Trans. Software Eng., vol. 12,no. 11,pp. 1089-1095,Nov.1986

[5] F. Sultan, T. Nguyen, L. Iftode, "Scalable Fault-Tolerant Distributed Shared Memory," Supercomputing, ACM/IEEE 2000 Conference, ISSN: 1063-9535

[6] S. Ramos-Thuel and J. Strosnider, "Scheduling Fault Recovery Operations for Time-Critical Applications," Proc. Fourth IFIP Conf. Dependable Computing for Critical Applications, Jan. 1995.

[7] T. Xie and X. Qin, "Scheduling Security-Critical Real-Time Applications on Clusters," IEEE Transactions on Computers, vol. 55, no. 7, pp. 864-879, July 2006.

[8] Bertram, T.J.; Demaree, K.D.; Dangelmaier, L.C., "An integrated package for real-time security enhancement,", IEEE Transactions on Power Systems, vol. 5, no. 2, pp.592-600, May 1990.

Authorized licensed use limited to: Auburn University. Downloaded on November 17,2026 at 22:48:31 UTC from IEEE Xplore. Restrictions apply.