SAFCM: A Security-Aware Feedback Control Mechanism for Distributed Real-Time Embedded Systems

Yue Ma[†], Wei Jiang[†#*], Nan Sang[†], Paul Pop[#]

[†]School of Computer Science and Engineering
University of Electronic Science and Technology of China
yue_ma_880131@hotmail.com, {weijiang, sn}@uestc.edu.cn
*Corresponding Author

[#]Informatics and Mathematical Modeling Department
Technical University of Denmark
DK-2800 Kongens Lyngby, Denmark
paul.pop@imm.dtu.dk

***Abstract*—Distributed Real-time Embedded (DRE) systems are facing great challenges in networked, unpredictable and especially unsecured environments. In such systems, there is a strong need to enforce security on distributed computing nodes in order to guard against potential threats, while satisfying the real-time requirements. This paper proposes a Security-Aware Feedback Control Mechanism (SAFCM) which has the ability to dynamically change the security level to guarantee soft real-time requirements and make the security protection as strong as possible. In order to widely support distributed real-time systems, a multi-input multi-output feedback loop is designed and a model predictive controller is deployed based on an equation model that describes the dynamic behavior of the DRE systems. This control loop uses security level scaling to globally control the CPU utilization and security performance for the whole system. We propose a "security level" metric based on an evolution of cryptography algorithms used in embedded systems. Experimental results demonstrate that SAFCM not only has the excellent adaptivity compared to open-loop mechanism, but also has a better overall performance than PID control mechanism.**

Keywords: Real-time Embedded Systems, Security-Aware, Feedback Control,

I. INTRODUCTION

With the rapid development of embedded systems and security technologies, real-time embedded systems are widely used in security-critical application areas, such as power grid and data acquisition systems. However, the operating environments in these areas are highly dynamic. The fluctuating system workload and unpredictable security surroundings make the traditional "open loop" mechanisms no longer applicable [1]. Traditionally, the static method to meet the system requirements of security and real-time mostly depends on static analysis which requires *a priori* knowledge of the Worst Case Execution Time (WCET) of each task [2]. Thus, such static approaches are not adaptable and have poor reactions in rapidly changing environments. These limitations are particularly serious for security-critical DRE systems. Facing more dangerous and unpredictable attacks, these DRE systems need higher security protections in addition to their hard or soft timing constraints. Failing to satisfy real-time or security requirements may lead to catastrophic consequences. Therefore, the design of effective DRE systems for security-critical real-time applications is very important.

To provide critical Quality of Service (QoS) guarantees, recently, a control theory approach has been proposed [1]. This approach has shown promise in providing robust QoS and real-time guarantees in unpredictable environments. While the traditional real-time scheduling relies on accurate characterization of workloads, this control theory approach can adapt to overloads and deadline misses via on-line performance feedback loops.

Many existing works on feedback control have focused on providing guarantees on stand-alone systems based on the assumption that tasks are independent. Unfortunately, such solutions are not applicable to DRE systems, for the nodes and tasks in distributed systems are inter-independent. Therefore, we consider it is necessary to propose new security-aware scheduling mechanisms for DRE systems. Such mechanisms are required to have the ability to monitor system status or slack time, and then dynamically change the Security Level (SL) for each node to maintain the desired utilization as well as provide high-level security protection. To achieve these goals, several issues need to be addressed. The first one is to design of a feedback loop. Corresponding to the features of distributed systems, the feedback loop should be Multi-Input Multi-Out (MIMO) where the system performance on multiple nodes must be guaranteed simultaneously. The second one is controller design. Traditionally, controller designs are based on linear control techniques such as Proportional Integral Derivative (PID) controller, which cannot easily be extended to distributed systems due to the MIMO feedback loop and practical constraints. Thus, it is necessary to introduce a more effective controller for security-critical DRE systems. The final issue is to assess cryptographic algorithms. To protect information security, researchers have designed many cryptographic algorithms, including symmetric, asymmetric and hash algorithms. Although these algorithms have been widely used in many devices, their performance in embedded systems has not been systematically evaluated.

In this paper, we propose a Security-Aware Feedback Control Mechanism (SAFCM) to address these problems. SAFCM can guarantee soft real-time requirements as well as provide high-level security protection. Considering one major application of DRE systems, supervisory control

978-0-7695-4824-1/12 $26.00 © 2012 IEEE
DOI 10.1109/RTCSA.2012.31

and data acquisition system [3], the SAFCM architecture features a MIMO feedback loop which is controlled by a global controller. At every sampling window, the surveillance components, which are deployed in distributed nodes, monitor the status of every node. Based on this information, the central controller, which is deployed in the host server, makes a global decision to adjust security level for each node. The mechanism increases security level when the system has enough slack, and decreases the level when it is busy. Compared to PID controller, the advanced Model Predictive Controller (MPC) is more effective for distributed systems [4]. Hence, we design a novel MPC controller as the core part of SAFCM to obtain good performance. To make our SAFCM more realistic, we evaluated the performance of several cryptographic algorithms in real embedded testbed and utilize the obtained data to verify the advantages of SAFCM through experiments.

The primary contributions of this paper are: (1) the design and development of a MPC controller for security-critical DRE systems to guarantee the security performance and soft real-time for the first time; (2) the design and analysis of a MIMO feedback control loop that maintains good system performance even when tasks' execution times vary significantly at runtime; (3) establishing a dynamic system model that describes the dynamic behavior of a distributed real-time system based on our measured execution times of cryptographic algorithms in a real embedded testbed.

The rest of this paper is organized as follows. We review related work in Section II. Section III presents the system architecture. We model the security-critical distributed real-time systems and introduce the design problem in Section IV. A MPC controller is designed and analyzed in Section V. Extensive experiments are conducted in Section VI and conclusions are drawn in Section VII.

II. RELATED WORK

Recently, many researchers have focused on risk and security management systems and proposed many useful algorithms and models. Chopra et al. discussed some of the main security issues and proposed solutions for real-time peer-to-peer communication [5]. Although these solutions can improve the security, they are static and cannot be adaptive to the unpredictable threats. M. Lin et al. integrate the group-based security model with a traditional real-time scheduling policy, and then design two EDF based security-aware scheduling schemes [6]. Jiang et al. proposed a hardware/software co-design technique to satisfy the confidentiality requirements of DRE systems with communication security constraints [7]. Qin et al. proposed a dynamic security-aware packet scheduling mechanism for wireless networks, which can achieve high security for packets and guarantee their soft real-time requirements [8]. Due to the lack of feedback control loop, these works have limited adaptivity and are not applicable to dynamic DRE systems.

To address the limitations of static algorithms, several dynamic and adaptive QoS scheduling policies have been proposed. A survey of feedback performance control in computing system is presented by Abdelzaher [9]. The dynamic feedback-based scheduling approaches have been applied in many application areas, such as dynamic voltage scaling [10], fault-tolerance [11] and thermal control [12]. Xie et al. bring the feedback idea into the area of security-critical clusters and propose a security-aware real-time heuristic strategy. Although the adaptivity is still limited, this mechanism has been proved more efficient than the static strategies [13]. However, the controllers in these works are inappropriate for distributed MIMO systems. Following the development of feedback control theory, advanced controllers have been proposed. X. Wang et al. deployed a MPC controller in a DRE system for end-to-end tasks to control the utilization and verified this controller has better performance than a PID controller [4], [14]. However, due to not taking security factors into consideration, these approaches cannot be directly used for security-critical DRE systems.

III. MOTIVATING APPLICATIONS AND SYSTEM ARCHITECTURE

A. Motivating Scenario

In this section, we describe a real-time distributed system, Supervisory Control And Data Acquisition (SCADA), which serves as a case study to investigate security-aware management mechanisms in DRE systems [3]. SCADA is a combination of telemetry and data acquisition and its typical structure is depicted in Fig. 1. Distributed nodes, as Programmable Logic Controllers (PLCs) in Fig. 1, encompass the collecting of information, transferring it back to the central site, the SCADA Master. SCADA Master carries out any necessary analysis and control and then displays that information on operator screens.

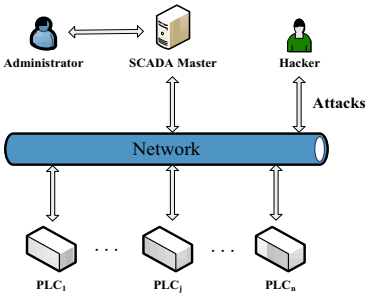


Fig. 1. Typical SCADA Network Architecture

The traditional proprietary standards of SCADA communication protocols are becoming more open, hence hackers can more easily gain in-depth knowledge of SCADA networks. Although many approaches have been proposed by researchers to improve SCADA network security, potential security risks are still very large [15]. Thus, we consider that it is necessary to improve the security for DRE systems like SCADA. Generally, deploying symmetric and hash cryptography algorithms can guarantee the integrity and confidentiality. In this work, we make efforts to design a dynamic mechanism to protect the system from intercepting and alternating attacks even when hackers enter into the network. The

DRE system is assumed clock synchronized and allows the switching between different cryptography algorithms.

B. Security-Aware Scheduling Framework

In this section, we propose architecture for SAFCM to manage security-critical DRE systems, as presented in Fig. 2. The MPC (the core of SAFCM) is deployed in the host server and makes global decisions. It collects the utilization (control variable) of each node periodically and calculates appropriate security level (control inputs). There are three major components in each distributed node, *Utilization Monitor*, *Security Actuator* and *Executor*. At each sampling window, the *Monitor* measures the utilization and feeds the samples back to MPC. Based on the control inputs conveyed from the MPC controller, the *Actuator* dynamically chooses the most appropriate security algorithm to encrypt or decrypt the sensitive data in tasks. *Executor* is in charge of executing tasks and guaranteeing their real-time requirements by special scheduling policies like RM and EDF [16]. Hence, the security algorithms and tasks can be seemed as separated. Tasks will be scheduled and executed by *Executor* in each node and the dynamic security algorithm decision is made by MPC and the *Actuator*.

For the MPC controller, the *multi-inputs* are the utilization of all the nodes and the *multi-outputs* are the integrity security level and confidentiality security level.

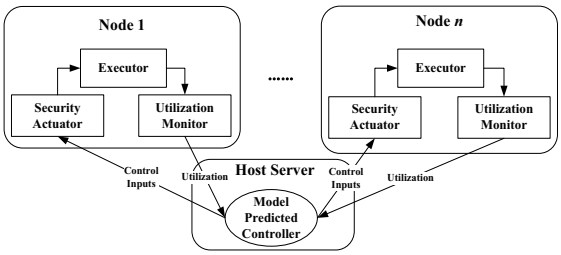


Fig. 2. The MIMO Control Loop

IV. SYSTEM MODELING AND PROBLEM FORMULATIONS

A. Security Overhead Model

To obtain the performance of cryptographic algorithms, some researchers have measured and analyzed their time executions [17]. However, many of these studies are mainly based on specific protocols which may not be suitable for real-time embedded systems. Hence, we evaluate the relative symmetric and hash cryptographic algorithms typically used in a real embedded platform. The security library we used is Cryptlib, which is open source and independent for any network protocol [18]. The target device is an embedded development board, TQ2440, shipped with an S3C2440 ARM processor and 64MB SDRAM. The measuring device is PXI 1024Q with NI 6221 data acquisition card. For symmetric algorithms, we are interested in the encryption, decryption and key initialization time; for hash algorithms, we only measure the encryption time.

Integrity and confidentiality can be accomplished by hash and symmetric algorithms, respectively. In this paper, we investigate four hash algorithms and seven symmetric algorithms for security. Fig. 3, 4 and 5 depict the results of our measurements.

From Fig. 3, we find an interesting phenomenon that with the same plaintext, the time cost of encryption approximates to the cost of decryption. Hence, we can make an average of the time costs of encryption and decryption to analyze cryptographic algorithms. Fig. 5 depicts the time cost of key initialization where the key size is a default value in Cryptlib. The execution time of key initialization is much larger than encryption and decryption. However, it is not necessary to dynamically initialize a new key at run time, and we assume all the required keys are established and distributed when system initializing and never changed during the runtime.

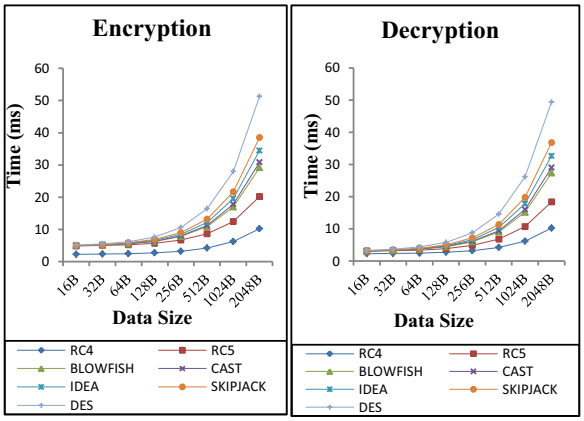


Fig. 3. Time Costs of Symmetric Algorithms.

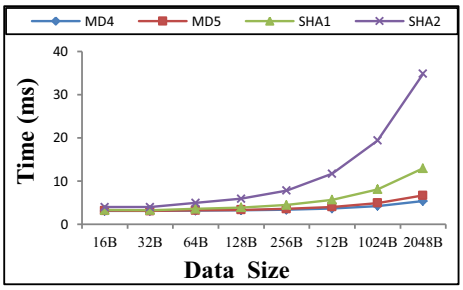


Fig. 4. Time Consumption of Hash Algorithms

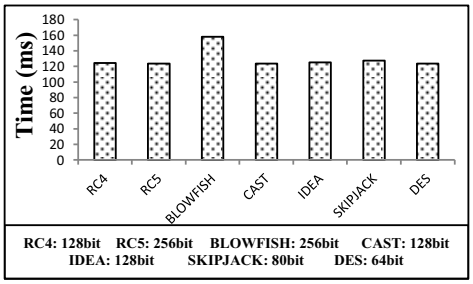


Fig. 5. Key Initialization Time of Symmetric Algorithms

Based on our measurement, we introduce the concept of security level to rank these cryptography algorithms in Table I and II. Similar to other findings, the security level of one algorithm is generally proportional to its execution

time [13]. We assign integrity security level 1 to MD4, the weakest but fastest encryption algorithm, and assign confidentiality security level 1 to RC4 for the same reason. Security levels of the remaining hash algorithms are computed by (1), where $AvgTime_i$ is the average execution time of the i^{th} cryptographic algorithm when the data size of plaintext in a range of 16B to 2048B.

$$SL_i^{int} = AvgTime_i / 3.7, 1 \le i \le 4 \quad (1)$$

Security levels of symmetric algorithms are computed by (2).

$$SL_i^{conf} = AvgTime_i / 4.2, 1 \le i \le 7 \quad (2)$$

TABLE I. Security Levels of Algorithms for Integrity

Cryptographic Algorithm	*AvgTime* (ms)	SL^{int}
MD4	3.7	1.000
MD5	4.0	1.098
SHA1	5.7	1.551
SHA2	11.6	3.169

TABLE II. Security Levels of Algorithms for Confidentiality

Cryptographic Algorithm	*AvgTime* (ms)	SL^{conf}
RC4	4.2	1.000
RC5	7.7	1.826
BLOWFISH	9.9	2.352
CAST	10.3	2.450
IDEA	11.3	2.665
SKIPJACK	12.3	2.906
DES	15.4	3.654

For the one-to-one relationship between security level and cryptography algorithm, we can use the security level of an algorithm to refer to the algorithm itself.

B. Task Model

In this paper, we consider a security-critical DRE system with n independent embedded nodes and in each node there are m periodic tasks running on it. Each task is denoted by $T_i = \{p_i, e_i, d_i, SL_i^{int}, SL_i^{conf}\}$, where p_i is the period, e_i is the execution time and d_i is the deadline. Since alternating and snooping are two common attacks in distributed systems, it is mandatory to deploy integrity and confidentiality services for every task. SL_i^{int} and SL_i^{conf} represent the integrity security level and confidentiality security level, respectively. As our definition of security level, the SL_i^{int} and SL_i^{conf} can be dynamically adjusted within $[SL_{\min}^{int}, SL_{\max}^{int}]$ and $[SL_{\min}^{conf}, SL_{\max}^{conf}]$, respectively.

The execution time of every task, e_i, is decided by three factors, the data size of the plaintext, the security levels of integrity and confidentiality, and the execution times of other operations with no security issue. Hence, e_i can be described as,

$$e_i = \lambda_i^{int} \cdot SL_i^{int} + \lambda_i^{conf} \cdot SL_i^{conf} + e_i^o \quad (3)$$

The coefficients λ_i^{int} and λ_i^{conf} are relative the size of sensitive data. The more the sensitive data, the bigger these two coefficients are. e_i^o represents the WCETs of the other security-unrelated operations.

C. Control Model of Security-Critical DRE System

To formulate the control model, some notations need to be introduced:

- $\boldsymbol{\omega_i}$: The task set in the i^{th} node.
- $\boldsymbol{u_i^s}$: The local utilization set point in the i^{th} node.
- $\boldsymbol{u_i(k)}$: The CPU utilization of node P_i in the k^{th} sampling window. It ranges from 0% to 100%.
- $\boldsymbol{\Delta SL^{int}(k)}$: One of the control inputs to adjust security level of integrity in the k^{th} sampling window.
- $\boldsymbol{\Delta SL^{conf}(k)}$: One of the control inputs to adjust security level of confidentiality in the k^{th} sampling window.
- $\boldsymbol{SL^{int}(k)}$: Security level of integrity services in the k^{th} sampling window.
- $\boldsymbol{SL^{conf}(k)}$: Security level of confidentiality services in the k^{th} sampling window.

Note that in a very large scale distributed systems, the host server may be confused if sensitive data is encrypted by different cryptographic algorithm in different node, we assume all nodes execute the same cryptographic algorithms at the same time. Though all these nodes change their security levels to the same new level at once, the solution for one node cannot directly apply to all other nodes. In the real-world applications, such as SCADA, these nodes are heterogeneous, and the load of every node is also disparate. Hence, we still need to design a global and integrated distributed mechanism rather than simply applying the one-node solution into the whole distributed system.

As the definition in equation (3), security level is proportional to tasks' execution times and of course proportional to system utilization. Similar with [4], we define the *estimated utilization*, $u'_i(k)$, as,

$$u'_i(k) = \sum_{T_j \in \omega_i} C_j^{int} \cdot SL^{int}(k) + C_j^{conf} \cdot SL^{conf}(k) \quad (4)$$

where the security levels ($SL^{int}(k)$, $SL^{conf}(k)$) in the i^{th} node. C_j^{int} and C_j^{conf} are the *estimated* proportional factors.

Following control theory, we must establish a dynamic model that characterizes the relationship between the control inputs ($\Delta SL^{int}(k)$, $\Delta SL^{conf}(k)$) and the controlled variable ($\Delta u_i(k)$). The *estimated* change to utilization ($\Delta u'_i(k)$) can be defined as follows,

$$\Delta u'_i(k) = \sum_{T_j \in \omega_i} C_j^{int} \cdot \Delta SL^{int}(k) + C_j^{conf} \cdot \Delta SL^{conf}(k) \quad (5)$$

We introduce a new parameter (g_i) to model the actual utilization $u_i(k)$ at k^{th} sampling window.

$$u_i(k) = u_i(k-1) + g_i \cdot \Delta u'_i(k) \quad (6)$$

where the system gain g_i represents the ratio between the estimated utilization change ($\Delta u'_i(k)$) and the actual utilization change ($\Delta u_i(k)$). The smaller the g_i, the more optimistic the estimation is, and the estimation comes to real situation more closely. However, the exact value of g_i is unknown due to the unpredictability of tasks' execution times and the environment.

Thus, we have established the dynamic model to describe the relationship between the control inputs and controlled variable. The whole equation is

$$u_i(k+1) = u_i(k) + g_i \left(\sum_{T_j \in \omega_i} C_j^{int} \cdot \Delta SL^{int}(k) + C_j^{conf} \cdot \Delta SL^{conf}(k) \right) \quad (7)$$

Equations (7) are used for a single node, but SAFCM is designed for a DRE system with multiple nodes. The following MIMO model can describe a DRE system with n nodes,

$$\boldsymbol{u}(k) = \boldsymbol{u}(k-1) + \boldsymbol{G} \cdot \Delta \boldsymbol{u'}(k-1) \quad (8)$$

$\boldsymbol{G}$ is a diagonal matrix where $g_{ii} = g_i$ $(1 \le i \le n)$ and $g_{ij} = 0$ $(i \ne j)$. Both of $\boldsymbol{u}(k)$ and $\boldsymbol{u'}(k-1)$ are n-dimensional vectors.

$$\boldsymbol{u}(k) = [u_1(k), u_2(k), ..., u_{\boldsymbol{n}}(k)]^T$$

$$\Delta \boldsymbol{u}'(k-1) = [\Delta u'_1(k-1), \Delta u'_2(k-1), ..., \Delta u'_n(k-1)]^T$$

From equation (5), the relationship between the estimated utilization and the security level is characterized as

$$\Delta \boldsymbol{u'}(k) = \boldsymbol{C} \cdot \Delta \boldsymbol{SL}(k) \quad (9)$$

where $\Delta \boldsymbol{SL}(k)$ is a two-dimensional vector and equals to $[\Delta SL^{int}(k), \Delta SL^{conf}(k)]^T$ and $\boldsymbol{C}$ is a n×2 matrix defined as

$$\boldsymbol{C} = \begin{bmatrix} \sum_{T_i \in \omega_1} C_i^{int} & \sum_{T_j \in \omega_1} C_i^{conf} \\ \sum_{T_i \in \omega_2} C_i^{int} & \sum_{T_i \in \omega_2} C_i^{conf} \\ \vdots & \vdots \\ \sum_{T_i \in \omega_n} C_i^{int} & \sum_{T_i \in \omega_n} C_i^{conf} \end{bmatrix}$$

Finally, the state equation of SAFCM is

$$\boldsymbol{u}(k) = \boldsymbol{u}(k-1) + \boldsymbol{C} \cdot \boldsymbol{G} \cdot \Delta \boldsymbol{SL}(k-1) \quad (10)$$

Equation (10) cannot be directly used by the controller for the unknown parameter $\boldsymbol{G}$. According to control theory, if a feedback system is stable and controllable, this $\boldsymbol{G}$ cannot affect the final system status. Thus, we assume $\boldsymbol{G} = diag(1, 1, \cdots 1)$ and design MIMO controller based on the following system mode,

$$\boldsymbol{u}(k) = \boldsymbol{u}(k-1) + \boldsymbol{C} \cdot \Delta \boldsymbol{SL}(k-1) \quad (11)$$

D. Problem Formulation

Security-aware feedback control can be formulated as a constrained optimization problem. The goal is to minimize the difference between the utilization set points and the actual utilization,

$$\min \sum_{i=1}^{n} (u_i(k) - u_i^s) \quad (12)$$

subject to following constraints

$$SL_{\min}^{int} \le SL_i^{int} \le SL_{\max}^{int} \quad (13)$$

$$SL_{\min}^{conf} \le SL_i^{conf} \le SL_{\max}^{conf} \quad (14)$$

These two constraints are the controlled regions of SL^{int} and SL^{conf} respectively. This optimization problem concerns the system performance as an integrated one, and dismisses the status of each single node. In our method to assign security level,

$$SL_{\min}^{int} = SL_{\min}^{conf} = 1.000$$

$$SL_{\max}^{int} = 3.169$$

$$SL_{\max}^{conf} = 3.654$$

In addition, both SL^{int} and SL^{conf} are discrete and can be only enumerated in Table I and II.

V. MPC CONTROLLER DESIGN

In this section, we propose a novel MPC controller to solve the problem. Firstly, we derive formulations of MPC in our SAFCM framework. Then, we transform these formulations into a quadratic programming problem which can be solved by using Matlab functions [19]. Finally, we analyze the stability and controllability of SAFCM.

A. The Formulation for MPC

A MPC controller includes a cost function, a reference trajectory, an appropriate predicted system model and a quadratic programming solver. In the end of every sampling window, the controller uses the solver to compute the control input that minimizes the cost function under the security level constraints. Referring to equation (11), the predictive model is:

$$\boldsymbol{u}(k+i+1 \mid k) = \boldsymbol{u}(k+i \mid k) + \boldsymbol{C} \cdot \Delta \boldsymbol{SL}(k+i) \quad (15)$$

The term controlled variable, $\boldsymbol{u}(k+i+1 \mid k)$, means that the CPU utilization at $(k+i+1)^{th}$ sampling window relies on the condition at k^{th} sampling window. The $\Delta \boldsymbol{SL}(k+i)$ are the control inputs which represents the change of security levels at the $(k+i)^{th}$ sampling window.

The MPC deployed in SAFCM can predict the system behavior in the next P sampling windows, called the *prediction horizon*. The control objective is to select the control inputs $\Delta \boldsymbol{SL}(k+i)$ that minimize the cost function under the security level constraints. The input trajectory includes the control input in the following M periods, called *control horizon*. The cost function to be minimized by MPC is

$$\begin{aligned} J(k) = & \sum_{i=1}^{P} \sum_{j=1}^{N} q_i [u_j(k+i \mid k) - w_j(k+i)]^2 \\ & + \sum_{i=0}^{M-1} r_i ([\Delta SL^{conf}(k+i) - \Delta SL^{int}(k+i-1)]^2) \\ & + \sum_{i=0}^{M-1} r_i ([\Delta SL^{conf}(k+i) - \Delta SL^{int}(k+i-1)]^2) \end{aligned} \quad (16)$$

where both the *error weight* q_i and the *control weight* r_i are nonnegative. We assume the *reference trajectory*, $w_j(k+i)$, is exponential,

$$w_j(k+i) = u_j^s - e^{-S_{ref} \times i} (u_j^s - u_j(k)) \quad (17)$$

where S_{ref} is a time constant to specify the speed of system to converge its set point. A larger S_{ref} can accelerate the system respond speed, as well as the possibility of oscillation. The first term in this cost function is *tracking error*, and the second and third terms are *penalty errors*, which are used to prevent system from dramatic oscillation or overstep the limited range. Comparing to the formulated optimization problem in (12), this cost function only has more coefficients and the *penalty errors*. Hence, the method to minimize the cost function can address this optimization problem [4].

For convenience, we transfer the scalar equation (16) into a matrix as follows.

$$\begin{aligned} \boldsymbol{J}(k) = & \sum_{i=1}^{P} \| \boldsymbol{u}(k+i \mid k) - \boldsymbol{w}(k+i \mid k) \|_Q^2 \\ & + \sum_{i=0}^{M-1} \| \Delta \boldsymbol{SL}(k+i) \|_R^2 \end{aligned} \quad (18)$$

The *tracking error weight* $\boldsymbol{Q}$ and the *control penalty weight* $\boldsymbol{R}$ are P-dimension and M-dimension matrix, respectively,

$$\boldsymbol{Q} = diag(q_1, q_2, ..., q_P)$$

$$\boldsymbol{R} = diag(r_1, r_2, ..., r_M)$$

In addition,

$$\boldsymbol{w}(k+i\mid k)=[w_1(k+i\mid k),w_2(k+i\mid k),...,w_N(k+i\mid k)]^T$$

Thus, we have established the formulation of MPC. The system optimization problem can now be described as finding a best input trajectory to minimize the cost function in (18) under the security level constraints in(13) and (14).

B. Solving the MPC Problem

Quadratic Programming (QP) is a classical approach to solve the mathematical optimization problem. Matlab has integrated a function, named QUADPROG, to solve this QP problem,

$$\min(\frac{1}{2}\boldsymbol{e}(k)^T\cdot\boldsymbol{H}_1\cdot\boldsymbol{e}(k)+\boldsymbol{H}_2{}^T\cdot\boldsymbol{e}(k)),\boldsymbol{LC}\le\boldsymbol{e}(k)\le\boldsymbol{RC} \quad (19)$$

$\boldsymbol{e}(k)$ is a vector of control inputs and $\boldsymbol{H_1}$, $\boldsymbol{H_2}$, $\boldsymbol{LC}$, $\boldsymbol{RC}$ are coefficient. Therefore, in order to solve the MPC problem, we only need transform the formulation of MPC to the standard QP problem.

The first step to solve the MPC problem is to create a relationship between the controlled variable $\boldsymbol{u}(k+i\mid k)$ and the current system status $\boldsymbol{u}(k)$. $\boldsymbol{u}(k+i\mid k)$ can be described as equation (20) by recursions of equation (15).

$$\begin{aligned}\boldsymbol{u}(k+i\mid k)&=\boldsymbol{u}(k+i-1\mid k)+\boldsymbol{C}\cdot\Delta\boldsymbol{SL}(k+i-1\mid k)\\&=\boldsymbol{u}(k)+\cdots+\boldsymbol{C}\cdot\Delta\boldsymbol{SL}(k+i-1\mid k)\end{aligned} \quad (20)$$

The utilization for given prediction horizon, P, can be written as,

$$\bar{\boldsymbol{u}}(k)=\boldsymbol{u}(k)+\boldsymbol{A}\cdot\boldsymbol{e}(k) \quad (21)$$

$$\boldsymbol{e}(k)=[\Delta\boldsymbol{SL}(k),\Delta\boldsymbol{SL}(k+1),...,\Delta\boldsymbol{SL}(k+P-1)]^T,$$

$$\bar{\boldsymbol{u}}(k)=\begin{bmatrix}\boldsymbol{u}(k+1\mid k)\\\boldsymbol{u}(k+2\mid k)\\\vdots\\\boldsymbol{u}(k+P\mid k)\end{bmatrix}_P,\quad \boldsymbol{u}(k)=\begin{bmatrix}\boldsymbol{u}(k)\\\boldsymbol{u}(k)\\\boldsymbol{u}(k)\\\vdots\\\boldsymbol{u}(k)\end{bmatrix}_P$$

$$\boldsymbol{A}=\begin{bmatrix}\boldsymbol{C}&0&0&\cdots&0\\\boldsymbol{C}&\boldsymbol{C}&0&\cdots&0\\\boldsymbol{C}&\boldsymbol{C}&\boldsymbol{C}&\cdots&0\\\vdots&\vdots&\vdots&\ddots&0\\\boldsymbol{C}&\boldsymbol{C}&\boldsymbol{C}&\cdots&\boldsymbol{C}\end{bmatrix}_{P\times P}$$

$\boldsymbol{e}(k)$ denotes the changing vector of control inputs in the control horizon. If $\boldsymbol{w}=[\boldsymbol{w}(k+1\mid k),...,\boldsymbol{w}(k+P\mid k)]^T$ and

$\boldsymbol{\Omega}=diag\left(\overset{2M}{1,...,1,0,...,0}\right)_{2P\times 2P}$, cost function (18) can be rewritten and simplified as the following equation,

$$\begin{aligned}\boldsymbol{J}(k)&=\|\boldsymbol{A}\cdot\boldsymbol{e}(k)+\boldsymbol{\Theta}(k)\|^2_{\boldsymbol{Q}}+\|\boldsymbol{\Omega e}(k)\|^2_{\boldsymbol{R}}\\&=\boldsymbol{e}(k)^T[\boldsymbol{A}^T\boldsymbol{QA}+\boldsymbol{\Omega}^T\boldsymbol{R\Omega}]\boldsymbol{e}(k)\\&\quad+2\boldsymbol{\Theta}(k)^T\boldsymbol{QAe}(k)+\boldsymbol{\Theta}(k)^T\boldsymbol{Q\Theta}(k)\end{aligned} \quad (22)$$

where $\boldsymbol{\Theta}(k)=\boldsymbol{u}(k)-\boldsymbol{w}$.

If $\boldsymbol{H}_1=2*(\boldsymbol{A}^T\boldsymbol{QA}+\boldsymbol{\Omega}^T\boldsymbol{R\Omega})$, $\boldsymbol{H}_2=(2\boldsymbol{\Theta}(k)^T\boldsymbol{QA})^T$ and $\boldsymbol{H}_3=\boldsymbol{\Theta}(k)^T\boldsymbol{Q\Theta}(k)$, the cost function can be transformed to the standard form finally as

$$\boldsymbol{J}(k)=\frac{1}{2}\boldsymbol{e}(k)^T\boldsymbol{H}_1\boldsymbol{e}(k)+\boldsymbol{H}_2{}^T\boldsymbol{e}(k)+\boldsymbol{H}_3 \quad (23)$$

In this equation, $\boldsymbol{H_1}$ and $\boldsymbol{H_2}$ are certain values and can be considered as coefficients in the k^{th} sampling window. For $\boldsymbol{H_3}$ is independent for $\boldsymbol{e}(k)$, it can be considered as a constant and ignored when solving the optimization problem.

The next step is to transform the constraints. Constraints (13) and (14) can be transformed to the linear inequality constraint form as $\boldsymbol{LC}\cdot x\le\boldsymbol{RC}$. Constraints can be described by matrix

$$\begin{bmatrix}\boldsymbol{E}&0&\cdots&0\\0&\boldsymbol{E}&\cdots&0\\0&0&\ddots&\vdots\\0&0&\cdots&\boldsymbol{E}\\-\boldsymbol{E}&0&\cdots&0\\0&-\boldsymbol{E}&\cdots&0\\0&0&\ddots&\vdots\\0&0&\cdots&-\boldsymbol{E}\end{bmatrix}_{2P\times P}\begin{bmatrix}\boldsymbol{SL}(k+1)\\\boldsymbol{SL}(k+2)\\\vdots\\\boldsymbol{SL}(k+P)\end{bmatrix}\le\begin{bmatrix}\boldsymbol{SL}_{max}\\\boldsymbol{SL}_{max}\\\vdots\\\boldsymbol{SL}_{max}\\\boldsymbol{-SL}_{min}\\\boldsymbol{-SL}_{min}\\\vdots\\\boldsymbol{-SL}_{min}\end{bmatrix}$$

where $\boldsymbol{E}$ is a unit matrix, $\boldsymbol{E}=diag(1,1)$. $\boldsymbol{SL}_{\min}$ and $\boldsymbol{SL}_{\max}$ represent the upper and lower bound.

$$\boldsymbol{SL}_{\min}=[\overset{int}{SL_{\min}}\ \overset{conf}{SL_{\min}}],\quad \boldsymbol{SL}_{\max}=[\overset{int}{SL_{\max}}\ \overset{conf}{SL_{\max}}]$$

Due to $\boldsymbol{SL}(k+P)=\boldsymbol{SL}(k)+\cdots+\Delta\boldsymbol{SL}(k+P-1)$, constraints can be described as

$$\boldsymbol{LC}\cdot\boldsymbol{e}(k)\le\boldsymbol{RC} \quad (24)$$

$$\boldsymbol{LC}=\begin{bmatrix}\boldsymbol{E}&0&\cdots&0\\0&\boldsymbol{E}&\cdots&0\\\vdots&\vdots&\ddots&\vdots\\0&0&\cdots&\boldsymbol{E}\\-\boldsymbol{E}&0&\cdots&0\\0&-\boldsymbol{E}&\cdots&0\\\vdots&\vdots&\ddots&\vdots\\0&0&\cdots&-\boldsymbol{E}\end{bmatrix}_{2P\times P}\begin{bmatrix}\boldsymbol{E}&0&\cdots&0\\\boldsymbol{E}&\boldsymbol{E}&\cdots&0\\\vdots&\vdots&\ddots&\vdots\\\boldsymbol{E}&\boldsymbol{E}&\boldsymbol{E}&\boldsymbol{E}\end{bmatrix}_{P\times P}$$

$$\boldsymbol{RC}=\begin{bmatrix}\boldsymbol{SL}_{max}\\\boldsymbol{SL}_{max}\\\vdots\\\boldsymbol{SL}_{max}\\\boldsymbol{-SL}_{min}\\\boldsymbol{-SL}_{min}\\\vdots\\\boldsymbol{-SL}_{min}\end{bmatrix}-\begin{bmatrix}\boldsymbol{E}&0&\cdots&0\\0&\boldsymbol{E}&\cdots&0\\\vdots&\vdots&\ddots&\vdots\\0&0&\cdots&\boldsymbol{E}\\-\boldsymbol{E}&0&\cdots&0\\0&-\boldsymbol{E}&\cdots&0\\\vdots&\vdots&\ddots&\vdots\\0&0&\cdots&-\boldsymbol{E}\end{bmatrix}\begin{bmatrix}\boldsymbol{SL}(k)\\\boldsymbol{SL}(k)\\\vdots\\\boldsymbol{SL}(k)\end{bmatrix}$$

Therefore, the MPC problem has been transformed to a standard QP problem. The Matlab built-in function, QUADPROG can minimize the equation (23) under the constraint in equation (24).

QUADPROG can address the MPC problem efficiently, but there is one shortage of this solution. From our method to determine the security level, the integrity level can have the following discrete values {1.000, 1.098, 1.551, 3.169}, and the confidentiality level can be in the set {1.000, 1.826, 2.352, 2.450, 2.665, 2.906, 3.654}. They are all discrete, but the control inputs

computed by MPC controller are a continuous value. Hence, we define a *minimization* disciple as: the actual security level can never be bigger than the computed security level. For example, if the computed security levels are $SL^{int} = 1.05$ and $SL^{conf} = 1.05$, the actual security levels in the next window are both 1.000, rather than $SL^{int} = 1.098$, $SL^{conf} = 1.826$. We will use this disciple in our experiments.

C. Stability Analysis

SAFCM is a linear time-invariant discrete system and the state equation is

$$\boldsymbol{u}(k) = \boldsymbol{u}(k-1) + \boldsymbol{C} \cdot \boldsymbol{G} \cdot \Delta \boldsymbol{SL}(k-1)$$

The operation of MPC is to find the most appropriate $\Delta \boldsymbol{SL}(k)$ to optimize the cost function $\boldsymbol{J}(k)$. Hence, we can get

$$\frac{\partial \boldsymbol{J}(k)}{\partial \Delta \boldsymbol{SL}(k)} = 0 \tag{25}$$

by derivation of $\boldsymbol{J}(k)$. Equation (25) can be described as,

$$\Delta \boldsymbol{SL}(k) = X(\boldsymbol{u}(k))$$

where $\boldsymbol{X}$ is an abstract function. Hence, the closed-loop system model is like

$$\boldsymbol{u}(k) = \boldsymbol{A}' \cdot \boldsymbol{u}(k-1) + \boldsymbol{B}' \tag{26}$$

where $\boldsymbol{A}'$ and $\boldsymbol{B}'$ are matrixes. According to control theory, a system is asymptotically stable if and only if all the eigenvalues of matrix $\boldsymbol{A}'$ are located inside the unit circle in the complex space.

The above methodology gives a general way to judge the stability. Given specific $\boldsymbol{C}$ and g_i, we can always analyze the stability. However, when $\boldsymbol{C}$ and g_i become larger, stability analysis becomes more complicated and time consuming.

D. Controllability Analysis

Controllability is a weaker concept than stability, which denotes the ability to move a system around in its entire configuration space using only certain admissible manipulations.

Controllability Condition: *the condition to guarantee controllability is*

$$rank(\boldsymbol{C} \cdot \boldsymbol{G}) = n, |\boldsymbol{C} \cdot \boldsymbol{G}| \neq 0 \tag{27}$$

or the matrix $\boldsymbol{C} \cdot \boldsymbol{G}$ *is full rank.*

Proof: If the $\boldsymbol{C} \cdot \boldsymbol{G}$ is full rank, it is,

$$rank(\boldsymbol{C} \cdot \boldsymbol{G}) = rank(\boldsymbol{C} \cdot \boldsymbol{G}, \cdots \boldsymbol{E}^{n-1} \cdot \boldsymbol{C} \cdot \boldsymbol{G}) = n$$

where the $\boldsymbol{E}$ is an unit matrix. From control theory and the system state equation in (10), the sufficient and necessary condition of controllability is,

$$rank(\boldsymbol{C} \cdot \boldsymbol{G}, \boldsymbol{E} \cdot \boldsymbol{C} \cdot \boldsymbol{G}, \cdots \boldsymbol{E}^{n-1} \cdot \boldsymbol{C} \cdot \boldsymbol{G}) = n, |\boldsymbol{C} \cdot \boldsymbol{G}| \neq 0$$

Hence the sufficient condition of controllability is equation (27).

VI. EXPERIMENTATAL EVALUATION

In this section, we conduct extensive simulations to evaluate SAFCM. Firstly, we test the system performance beyond different loads. Next, we try to confirm if SAFCM can be used in both optimistic and pessimistic estimation situations. Thirdly, we compare our proposed system with open-loop and FCS [1] mechanisms to prove the advancement of SAFCM. Finally, we measure the overhead of the MPC controller.

A. Experiment Configuration

The simulator is implemented in C# (in Visual Studio 2010) and the MPC controller is implemented in Matlab (R2009b). The system initializes at $T = 0$ and executes the feedback algorithm once every 100 time units. For every task, we assume each of the two *estimated* proportional factors, C_j^{int} and C_j^{conf}, are set to 0.2, and both λ_i^{int} and λ_i^{conf} (in equation (3)) are random in the range from 1.0 to 1.2. We assume all the security-unrelated operations are independent. They can be executed by some periphery components and do not participate in system scheduling. The periodic tasks concern only security-related operations. Hence, the e_i^o can be set to 0 in experiments. Every node executes same number of periodic tasks, and the system gain g_i is a random value ranging from 5 to 6. Parameters in the MPC controller, M and P, are 1 and 3, respectively.

In order to describe the overall performance of whole system, we introduce the concept, *global utilization*, which is defined as the average utilization of all nodes. Corresponding to this concept, *local utilization* represents the utilization of a single node. At every 100 time units, which is defined as one sampling window, we calculate this couple of utilizations. In real-time scheduling theory, the schedulable utilization bounds derive from the different scheduling polices. It has been established that the schedulable utilization bound of EDF is 100%, and the bound of RM is relied on the number of tasks [16]. For the soft real-time system, it's allowed the CPU utilization exceeds the schedulable bound sometime, but at most time, the CPU utilization must be bounded by these schedulable bounds. Similar as [1], the set point of each node, u_i^s, is better to set smaller than the schedulable bound. In addition, we assume all the u_i^s are equal to the global set point, U^S. For EDF, $U^s = 85\%$, and for RM,

$$U^s = (m \cdot (2^{1/m} - 1) - 0.10) \times 100\%$$

where m represents the number of tasks in a processor.

B. Exp. 1: Runtime Performance

We test the system performance under two different conditions (in TABLE III). The global utilization and local utilization of the 1st processor under the *Condition-A* are shown in Fig. 6.

TABLE III. LOAD CONDITIONS

Condition	Scheduling	Tasks	Nodes
Condition-A	RM	10	4
Condition-B	EDF	14	8

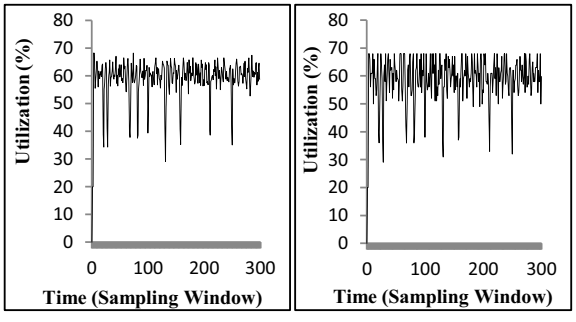


Fig. 6. Global (left) and Local (right) Utilization under *Condition-A*

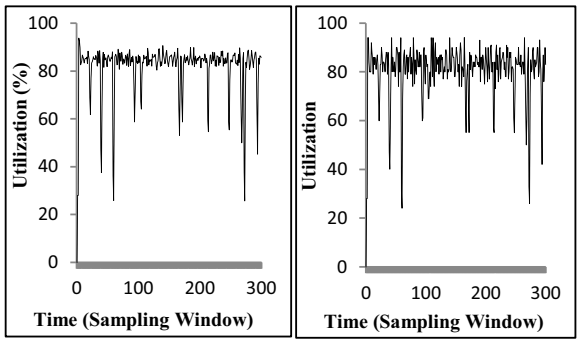


Fig. 7. Global (left) and Local (right) Utilization under *Condition-B*

From Fig. 6, we can see no matter the global or local utilization can fluctuate with the set point $U^s = 62\%$. When the utilization is higher than the set point, SAFCM will automatically decrease the security level to make the utilization back to the desired value. If the utilization is lower than the desired, SAFCM will use the available processor resources to increase security level. In the further analysis, the average security levels are $SL_{avg}^{int} = 3.13$ and $SL_{avg}^{conf} = 3.52$, which are close to the highest ($SL_{\max}^{int} = 3.169$ and $SL_{\max}^{conf} = 3.654$), and the average global utilization is 59.2%, and the peak is 68%. It can be concluded that under these conditions, SAFCM can provide high protection as well as try to control the utilization under the U^s. For the security level is discrete and limited, the system cannot completely stabilize at the desire situation.

Fig. 7 shows the global and local (at 1st processor) utilization fluctuation under the *Condition-B*. In *Condition-B*, the average security levels are $SL_{avg}^{int} = 3.03$ and $SL_{avg}^{conf} = 3.58$. The average global utilization is 81.8% and the peak is 93.9%. Hence, the performance of SAFCM in this condition is also advanced.

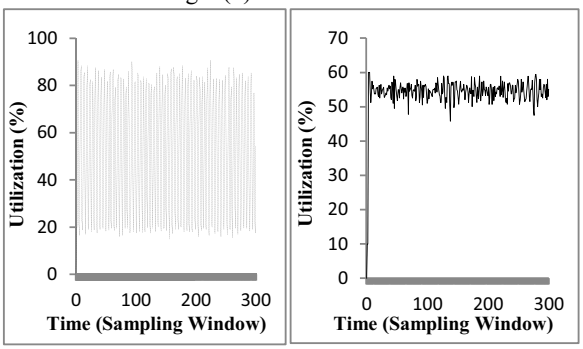
From this couple of experiments, we can observe that SAFCM can achieve the specified goals and make a good trade-off between real-time and security. On one hand, it meets the soft real-time requirement via controlling the utilization; on the other hand, it provides the security protection as strong as possible.

C. Exp. 2: Optimistic or Pessimistic Estimation

From control theory, system gain g_i can be ignored during the design of MPC controller, but it does not mean this parameter is insignificant. Actually, it represents the feature of our estimation: a small g_i means the estimation accurate and optimistic. However, it is difficult to determine whether our estimation is optimistic or pessimistic, so we want to measure the performance with different system gains in this set of experiments.

Under the *Condition-A*, we measure and analyze the system performance when g_i ranges in different regions. The statistical data for different gains is shown in Fig. 8. Based on the measured data, we find that our requirements can still be satisfied even when the estimation is pessimistic ($g \in (7.5,8)$). In this set of experiments, the average utilization is only varied in a small range around 50%, which is lower than the set point, 62%. However, following the rising g_i, the peak utilization increased, which means the possibility of more tasks might be deadline missed. At the same time, the security level of integrity and confidentiality services decreases gradually.

In addition, we compare the global utilization fluctuation when the gain ranges in different regions. In Fig. 9(a), the gain ranges from 7 to 7.5, and gain ranges from 4 to 4.5 in Fig. 9(b).



(a) (b)

Fig. 9. Global Utilization When (a) $g \in (7,7.5)$, (b) $g \in (4.5,5)$

When g_i ranges from 7.0 to 7.5, system has already become unstable and dramatically oscillates. Undoubtedly, the dramatic variations of utilization and security level are not we expected and can even make system unusable. Based on these analyses, it can be concluded the system performance can be better if g_i is small. So, careful analysis of the execution environments and optimistic estimation are still necessary to improve the performance of SAFCM.

D. Exp. 3: Steady Execution Times

We verify the advantages of SAFCM via comparing with open-loop and FCS mechanism in experiment 3 and 4. In experiment 3, we assume that the execution times of tasks do not change rapidly. In experiment 4, the execution times vary dynamically at runtime. Through these two experiments, we analyze the performance of SAFCM in the steady and changeable environments.

The open-loop mechanism is the most traditional and simplest strategy that uses fixed security level. For this mechanism, the administrator requires *a priori* knowledge of tasks' WCETs and assigns an appropriate security level before system initialization. If there are m tasks in a single node, the constraint of security level is,

$$\sum_{i=1}^{m} \frac{WCET_i}{p_i} \le U^s \tag{28}$$

where p_i is the period of task T_i. In our experiment, we assume $m = 10$, $p_i = 100$, and the basic scheduling policy is RM. Under the inequality (28), the highest security level is,

$$SL^{int} = 1.551, SL^{conf} = 3.654$$

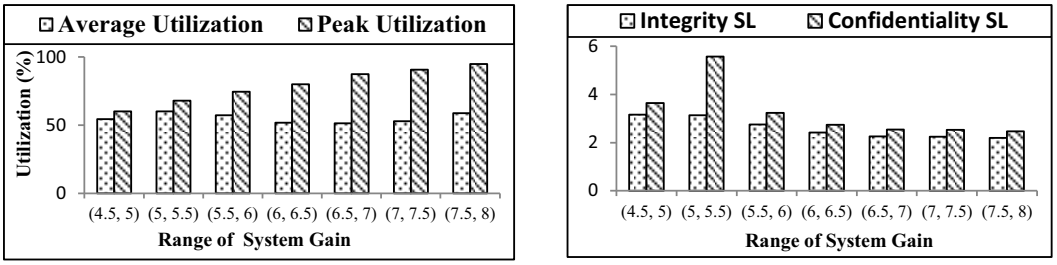
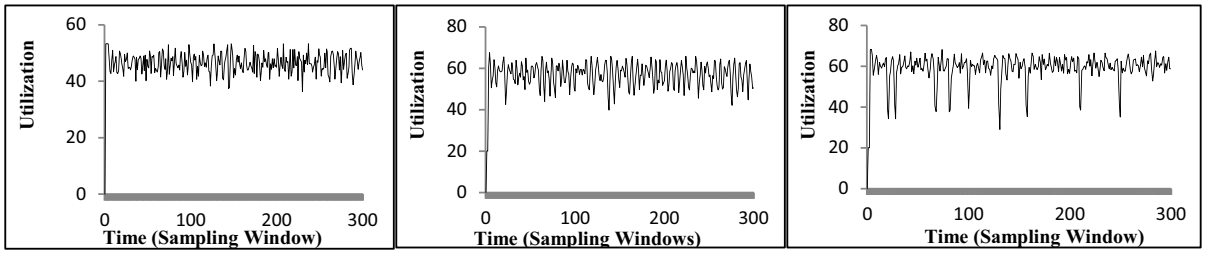


Fig. 8. System Performance When System Gain Range in Different Regions



(a) (b) (c)

Fig. 10. Global Utilization of (a) Open System. (b) FCS System. (c) SAFCM System When Execution Time is Steady

TABLE IV. Performance Comparison between Open FCS and SAFCM System When Execution Time is Steady

	Average Utilization	Peak Utilization	Integrity SL	Confidentiality SL
Open	46.5%	53.3%	1.55	3.654
FCS	56.8%	67.5%	3.162	3.34
SAFCM	59.2%	68.3%	3.13	3.52

FCS is a feedback controlled QoS management mechanism with three important components [1]. The *Monitor* measured the controlled variables and feeds the samples back to the *Controller*. Relying on these samples, *Controller* compares the performance reference with corresponding controlled variables to computes a change. The QoS *Actuator* dynamically adjusts the QoS levels of tasks according to the change computed by the Controller.

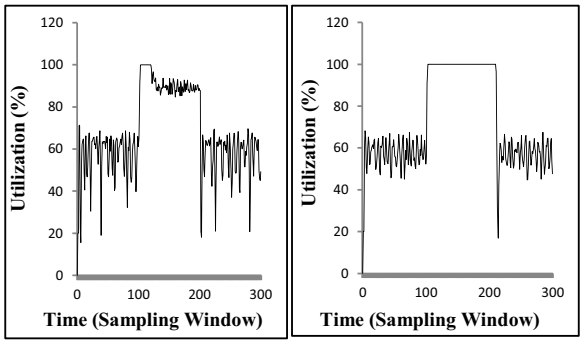
Though the FCS is designed for a stand-alone system, it can still be used in DRE system via monitoring the global utilization. The policy deployed in the *Actuator* to change the security level is similar with open-loop mechanism. We evaluate SAFCM, FCS and open-loop mechanism under the same assumptions. At the runtime, the execution times of tasks are steady

The results are shown in Fig. 10 and TABLE IV. It is obvious the FCS and SAFCM have better performance than open-loop. In open-loop, although the confidentiality level is highest, the average security level is lowest and utilization is far lower than desired, which means processor resources have been wasted. Analyzing the utilization curve, SAFCM is more stable than FCS. FCS and SAFCM have similar performance, while the security level of SAFCM is a little higher. The peak utilizations of these three systems do not exceed the schedulable utilization bound of RM, 71% which means all the systems can guarantee the real-time.

As a conclusion, SAFCM is more stable and can provide stronger security protection than FCS when executing in a steady environment.

E. Exp. 4: Varying Execution Times

Besides the security level, tasks' execution times are also affected by the data size of plaintext. Hence, in this experiment, we mainly test the system performance of FCS and SAFCM under various plaintext and execution times. We assume the execution times dramatically increase k times than the original at the 100th sampling window, and back to normal at the 200th sampling window. The global utilization fluctuation when $k = 5$ is shown in Fig. 11.



(a) SAFCM (b) FCS

Fig. 11. Global Utilization When Execution Times Fluctuate at Runtime.

As shown in Fig. 11a, SAFCM makes effort to converge the set point despite the significant variations in execution times. At the 100th window, nodes are suddenly overloaded due to the execution times dramatically increase 5 times than normal, and utilization becomes maximum. Beginning at 125th sampling window, the global utilization drops for SAFCM dynamically decreases the security level to converge the set point. Since the values of tasks' execution times are so big even with lowest security level, global utilization stabilizes at

90% during the period from 125th to 200th window. At end of this period, nodes are suddenly underutilized, and then the utilization quickly returns to normal situation within 5 sampling windows. By contrast, the performance of FCS is very poor in this overloaded period (in Fig. 11b). The utilization of FCS always stabilizes at 100%, so it can be recognized as losing its adaptivity. FCS has become invalid in this situation, but SAFCM is still adaptive and enforces to avoid long time of full loaded.

F. Exp. 5: Overhead of MPC

The MPC controller has stronger ability than traditional PID controller at the cost of complexity. As discussed in Section V, the MPC problem can be transformed to a quadratic programming problem. To estimate the overhead of the MPC controller, we can measure the execution time of the quadratic programming solver function, QUADPROG in Matlab. MPC controller is generally deployed in the host server, such as SCADA Master, which has powerful processing capacity. Thus, we conduct this simulation on a 2.8GHz CPU with 2GB memory platform to test the performance of MPC. The results (in TABLE V) indicate the method to solve the MPC problem has low overhead. We consider the overhead is acceptable even in the large scale field like 200 distributed nodes.

TABLE V. MPC Controller Overhead

Nodes	10	20	50	100	200
Time (ms)	35	41	43	50	380

VII. CONCLUSION

This paper has presented a security-aware scheduling mechanism for distributed real-time embedded systems. Inspired from the feedback control theory, SAFCM is designed to dynamically manage the security level based on the handled utilization of multiple nodes to guarantee the soft real-time requirements and make security protection as strong as possible. To make our system model more practical, we measure the time costs of some cryptographic algorithms on a real embedded platform. Based on the system model, the designed MIMO feedback control loop can increase the security level and make the system more secure when system is in low workload state, or decrease the security level to satisfy the soft real-time requirement if system is busy. As the core component of SAFCM, an advanced and more suitable controller, MPC, is deployed. Through mathematic transformation and simplification, the MPC problem can be converted into a quadratic programming problem and solved using Matlab. Extensive simulations demonstrate that SAFCM can maintain the desired utilizations on multiple nodes as well as provide higher security protection than open-loop and FCS. Moreover, when the tasks' execution times change dramatically at runtime, SAFCM has also a better overall performance than FCS.

ACKNOWLEDGEMENT

This work was partly supported by the National Natural Science Foundation of China under Grant No. 61003032, the Fundamental Research Fund for the Central Universities of China under Grant No. ZYGX2011J061, and the Research Fund of National Key Laboratory of Computer Architecture under Grant No. CARCH201104.

REFERENCE

[1]. C. Lu, J. A. Stankovic, G. Tao, S. H. Son, "Feedback Control Real-Time Scheduling: Framework, Modeling, and Algorithms," Journal of Real-Time Systems, Special Issus on Control-Theoretical Approaches to Real-Time Computing, 2002.

[2]. V. Varadharajan, S. Black, "A multilevel security model for a distributed object-oriented system", Proceedings of the Sixth Annual Computer Security Applications Conference, pp. 68-78, 1990.

[3]. D. Bailey, E. Wright, "Practical SCADA for Industry," Elsevier, 2003.

[4]. C. Lu, X. Wang, X. Koutsoukos, "Feedback Utilization Control in Distributed real-Time Systems with End-to-End Tasks," IEEE Transactions on Parallel and Distributed Systems, vol. 16, Issue. 6, pp. 550 – 561, 2005.

[5]. D. Chopra, H. Schulzrinne, E. Marocco, E. Ivov, "Peer-to-Peer Overlays for Real-Time Communication: Security Issues and Solutions," IEEE Communications Surveys & Tutorials, vol. 11, Issue. 1, pp. 4 -12, 2009.

[6]. M. Lin, L. Xu, L. T. Yang, X. Qin, N. Zheng, Z. Wu, M. Qiu, "Static Security Optimization for Real-Time Systems," IEEE Transactions on Industrial Informatics, vol. 5, Issue.1, pp. 22 – 37, 2009.

[7]. K. Jang, P. Eles, Z. Peng, "Co-design techniques for distributed real-time embedded systems with communication security constraints," Design, Automation & Test in Europe Conference & Exhibition (DATE), pp. 947 – 952, 2012.

[8]. X. Qin, M. Alghamdi, M. Nijim, Z. Zong, K. Bellam, X. Ruan, "Improving Security of Real-Time Wireless Networks through Packet Scheduling," IEEE Transactions on Wireless Communications, vol. 7, issue. 9, pp. 3273 – 3279, 2008.

[9]. T. F. Abdelzaher, J. A. Stankovic, C. Lu, R. Zhang, Y. Lu, "Feedback Performance Control in Software Services," IEEE Transactions on Control Systems, vol. 23, issue. 3, 2003.

[10]. Y. Zhu, F. Mueller, "Feedback EDF Scheduling Exploiting Dynamic Voltage Scaling," Real-Time and Embedded Technology and Applications Symposium, pp. 84-93, 2004.

[11]. R. Sridharan, N. Gupta, R. Mahapatra, "Feedback-Controlled Reliability-Aware Power Management for Real-Time Embedded Systems," ACM/IEEE on Design Automation Conference, pp. 185-190, 2008.

[12]. Y. Fu, Y. Chen, C. Lu, "Feedback Thermal Control for Real-Time Systems," Real-Time and Embedded Technology and Applications Symposium (RTAS), pp. 111 - 120, 2009.

[13]. T. Xie, X. Qin, "Scheduling Security-Critical Real-Time Applications on Clusters," IEEE Transaction on Computers, vol. 55, Issus. 7, pp. 864-879, 2006.

[14]. X. Wang, X. Fu, X. Liu, Z. Gu, "Power-Aware CPU Utilization Control for Distributed Real-Time Systems," Real-Time and Embedded Technology and Applications Symposium (RTAS), pp. 233 - 242, 2009.

[15]. V. M. Igure, S. A. Laughter, R. D. Williams, "Security Issues in SCADA Networks," Elsevier Computers & Security, vol. 25, Issue. 7, pp. 498 – 506, 2006.

[16]. C. L. Liu, J. W. Layland, "Scheduling Algorithms for Multiprogramming in a Hard Real-Time Environment," Journal of ACM, vol.20, pp. 46 - 61, 1973.

[17]. O. Elkeelany, M. M. Matalgah, K. P. Sheikh, M. Thaker, "Performance Analysis of IPSec Protocol: Encryption and Authentication," IEEE International Conference on Communications, vol. 2, pp. 1164 – 1168, 2002.

[18]. Peter Gutmann, Cryptlib Security Toolkit Version 3.4.http://www.cryptlib.com.

[19]. MATLAB Manual, Version 6.5, Mathworks, Inc.