siwn.org.uk

International Transactions on Systems Science and Applications
Volume 7 Number 1/2 November 2011 pp. 26-39

ITSSA

Trading Security for Control Performance in Distributed Robotic Applications

Ruslan ASAULA [a] Tommaso CUCINOTTA [b] Gianluca DINI [c,1] and Luigi PALOPOLI [a]

[a] *Dipartimento di Scienza e Ingegneria dell'Informazione, University of Trento, Italy*

[b] *Scuola Superiore S.Anna, Pisa, Italy*

[c] *Dipartimento di Ingegneria dell'Informazione, University of Pisa, Italy*

Abstract:Networked embedded and control systems are largely used in factory automation for production and logistics tasks. In this application domain, security has become a prominent issue due to the critical consequences a cyber attack may have in terms of safety and financial losses. Unfortunately security solutions compete against control applications for the often scarce resources of embedded platforms. In this paper, we show how security can be dealt with as one of different Quality of Service dimensions and traded for control performance in an adaptive QoS management scheme. The system is able to respond to increased resource requirements or to changes in the risk level by reconfiguring the application modes and the security modes. We offer an explanatory case study to show how this idea is implemented.

Keywords: Quality of Service, security, distributed control, distributed automation systems

1. Introduction

Networked Embedded Control Systems (NECS) are largely used in factory automation for production and/or logistics tasks. In this application domain, meeting performance requirements (e.g., deadlines) is essential, since a miss may cause business losses and safety infringements.

The move from proprietary technologies to more standardised and open solutions together with the increased number of connections with Internet has made NECS more vulnerable to cyber attacks. Because of the critical nature of many factory automation systems, successful attacks could cause massive financial losses [3]. The recent case of the Stuxnet worm has shown that malicious intrusions in distributed automation systems are not merely an academic speculation but are very concrete risk [5,11].

While security, Quality of Service (QoS), and real-time issues of NECS have been extensively investigated, their interplay has emerged only recently as an important problem [9,10,16,17]. Intuitively, the crucial issue is that security and performance compete for the same computational and communication resources that are often limited in embedded platforms. The quality of control experienced by control applications is heavily affected by the amount resources they can rely on [12,13]. Thereby, if we always execute the communications with the maximum level of security, control application can incur a severe performance degradation. On the other hand, using the lowest level of security can make the factory liable to external attacks when an intrusion detection system (IDS) evaluates an increased level of security risk [15]. In this case, a controlled performance degradation can be an acceptable price to pay to guarantee the overall system security and safety.

[1] Corresponding author: Via Diotisalvi 2, 56100 Pisa, Italy; email: dini@iet.unipi.it.

sai: itssa.2011.11.024 ISSN 1751-1461 (print) ISSN 1751-147X (cd-rom)

To face this challenge, we advocate an approach based on *flexible security policies and application modes*. In our idea, the system quality can be evaluated along different dimensions. One of them is the Quality delivered by the control applications (Quality of Control - QoC) and the other is the security level achieved in communications (Quality of Security -QoSec). These two different metrics are in evident tradeoff [16,10]: if we reduce the security level the saved bandwidth can be reclaimed by the control applications to increase their quality (and viceversa). Different tradeoff solution between QoC and QoSec can be sought adaptively in different moments of the system execution (e.g., based on the evolution of application load and/or of the security risks).

To make this abstract idea a viable approach, we need a “tuning knob” to change the QoC delivered by the control application and a similar “tuning knob” to change the level of security. In our framework, we assume that the control applications can operate in a discrete set of modes, each one associated with a different resource consumption and with a different QoC level. Likewise, communications can have discrete security options, each one associated with a different resource consumption and with a different level of QoSec. This way, the system adaptation problem can be set up in an optimisation framework where QoC and QoSec can be differently weighted in the cost function and under the constraint of a finite availability of resources.

In this paper, we show how this problem can be formulated and solved in mathematical terms and discuss the most important architectural issues underlying the implementation of the paradigm. What is more, we will present a concrete case study consisting of a set of mobile robots moving in the factory floor. Control applications are assumed compliant with the paradigm described above (i.e., discrete functional options) and so are the security protocols used for the communications. We have developed a simulator that captures the dynamics of the different agents, implements the adaptation algorithm of QoC and QoSec and shows how the proposed framework allows the designer to define different policies.

The paper is organized as follows. Section 2 provides a brief overview of related works. Section 3 presents our approach, whereas Section 6 presents simulations results showing the effectiveness of the proposed technique. Finally, Section 7 draws final conclusions and introduces future work.

2. Related Work

The problem of scheduling a set of independent real-time tasks with various security requirements in embedded systems has been previously faced with by both Xie and Qin [16,17] and Lin *et al.* [10]. Both scheduling schemes integrate security requirements into the real-time scheduling with the aim at optimising the security quality while guaranteeing the schedulability of the real-time tasks. Both scheduling schemes hinge on security service and service quality. A *security service* specifies the security requirements one wishes to achieve, i.e., confidentiality, integrity, authenticity, or a mix of them whereas the *service quality* specifies the actual cryptographic mechanisms used to implement a given the service. Xie and Qin face with the problem of scheduling real-time aperiodic tasks [16] and periodic real-time tasks [17] and provide a solutions based on the Earliest Deadline First (EDF) scheduling algorithm. Lin *et al.* consider the EDF scheduling algorithm too but they face with it from a static point view so reducing the scheduling problem to a combinatorial optimization problem.

Both Xie and Qin and Lin *et al.* consider the quality of a security service at a very fine-grain, i.e., at the level of cryptographic primitive. That is, they select the security service quality by choosing, dynamically in one case and statically in the other, ciphers and hash functions among a number of possible choices according to their computing overhead. In this paper we take a similar fine-grain approach except we refer to cryptographic key lengths rather than cryptographic algorithms. In other words we associate different security services to different key lengths, where longer keys define more secure but more resource demanding services.

A key difference between this line of research and ours is that we do not directly link the security choice to the scheduling mechanism. We consider the security as one of the possible metrics

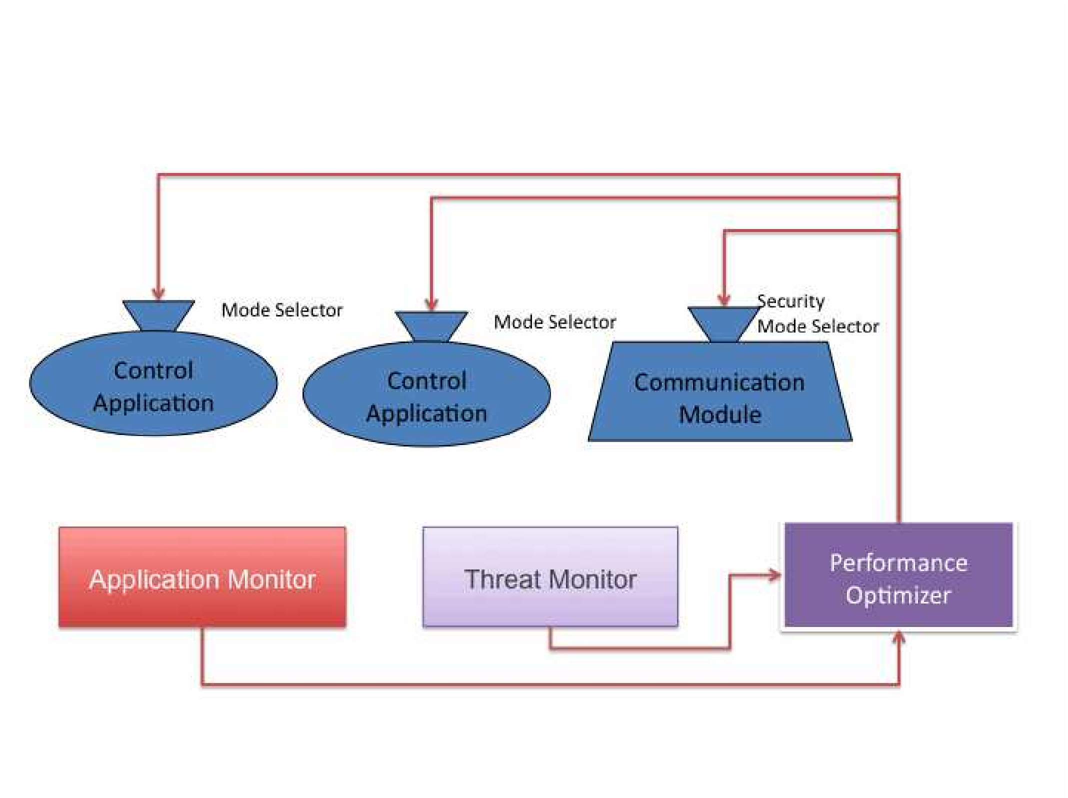

Figure 1. Block scheme of the system proposed in this paper

whereby the quality of service of the system can be quantitatively assessed. This metric is traded against other possible QoS metrics (in our case related to control performance) in an optimisation program. This idea is clearly inspired by QRAM [7], an optimisation framework in which tradeoff solution are sought between conflicting QoS metrics, under the constraints dictated by the finite availability of resource. This approach is founded on the idea that we can control the amount of resource that each application receives using an appropriate technological support in the operating system and /or in the middleware [2]. In our previous work [6], we have shown how to a resource mechanism *a la* QRAM can be implemented online using a monitoring mechanism on the resource requirements. In this work, we further develop this idea and consider the Quality of Service as a QoS dimension. In this respect, the work has evident connection with the idea of security services championed by Irvine and Levin [8]. A similar approach to the one presented in this paper has been proposed by Kang and Son [9]. They formulate the security support in real-time critical systems as a QoS optimisation problem and associate security levels to cryptographic key lengths. The authors present a very preliminary idea, which we evolve into a well founded mathematical and quantitative framework and adapt to the context of factory automation.

3. Problem description and solution overview

The basic idea that we propose in the paper is well described in the block scheme in Figure 1. We have a collection of *multi-mode* applications, which can switch between a variety of possible *operation modes*. Each mode corresponds to a different quality of control and it generates different workloads on the computation and communication resources needed for the execution. The communications required by the applications can in their turn be carried out with different *security modes*. Every mode corresponds to a different security level and introduces a different computation and communication overhead. When an application executes its associated real-time constraints should never be violated.

Our system monitors the behaviour of the application and the potential security threats. Whenever a state change is detected that could require a different balance in the allocation of the system resources, the performance optimiser adapts the configuration of application and security modes.

Our approach requires: i) an optimization framework; ii) an organisation of the applications in discrete operating modes; iii) an organisation of security in discrete security modes and a mechanism to dynamically select switch them; and, finally, iv) a low-level scheduler that ensures that each application receives the level of resources associated with the mode in which it operates. In the following sections, we will provide a detailed discussion on each of these issues.

4. The performance optimiser

The optimisation framework we propose provides a formal underpinning for our QoC/QoSec tradeoff policy. We remark that the formalisation proposed below naturally leads to a centralised solution of the optimisation problem, and hence to a centralised implementation of the performance optimiser. A distributed implementation of this component is currently being evaluated. Another remark is that the algorithm discussed below can be executed either on-line or off-line. In the latter case: 1) the problem is solved for a set of possible system configurations, 2) the solutions for the different configurations are stored in a table, 3) the performance optimiser on-line simply detects the current configuration and looks for the related entry in the table to retrieve the optimal solution.

The choice between online and off-line implementation depends on the size of the problem and on the resulting duration of the optimisation procedure. The online solution is better suited for problems of small size, while the offline solution has to be applied when the number of decision variables is in the order of hundreds.

4.1. Notation

A formal statement of the optimisation problem requires a notation allowing us to express constraints and utility functions. In our formalisation:

- possible QoC levels are represented by integer numbers, where greater values correspond to better control performance and are normally associated to greater resource requirements (i.e., greater computing times and/or packet sizes). Since we can host applications different from control (e.g., data-logging), *we will generally speak about Quality of Service (QoS) and specialise this notion to QoC when the application in question has to do with control.*
- possible security levels are represented as integer numbers as well, with greater values corresponding to increased security and greater resource requirements;
- for discrete variables, a vector-representation is adopted in which the j^{th} vector component is 1 (and all the others are 0) if and only if the original variable is j. For instance, if an application can run in three modes and the first mode is currently selected, we will use a vector $Q^{(1)} = \{1, 0, 0\}$.

Let:

- $\mathscr{A} = \{1, \ldots, N_A\}$ denote the set of applications;
- $\mathscr{H} = \{1, \ldots, N_H\}$ denote the set of hosts;
- $\mathscr{R} = \{1, \ldots, N_P\}$ denote the set of processors, where each processor $r \in \mathscr{R}$ has a maximum computation capacity $U_P^{(r)}$, and $\mathscr{R}_h \subset \mathscr{R}$ denote the set of processors belonging to host $h \in \mathscr{H}$;
- $\mathscr{S} = \{1, \ldots, N_S\}$ denote the set of wireless or wired subnets, where each subnet $s \in \mathscr{S}$ has a maximum traffic capacity $U_S^{(s)}$, and $\mathscr{H}_s \subset \mathscr{H}$ denote the set of hosts connected via the subnet $s \in \mathscr{S}$;

- $\mathscr{V}_Q^{(i)} = \left\{1, \ldots, V_Q^{(i)}\right\}$ denote the set of possible QoS modes for the i^{th} application;
- $\mathscr{V}_S^{(i)} = \left\{1, \ldots, V_S^{(i)}\right\}$ denote the set of possible security modes for the i^{th} application;
- $\mathscr{V}_P^{(r)} = \left\{1, \ldots, V_P^{(r)}\right\}$ denote the set of possible power modes for the r^{th} resource;
- $T^{(i,r)}$ denote the minimum activation period of the task deployed on resource r of application i;
- $\mathbf{Q}^{(i)}$ denote the vector-representation of QoS levels corresponding to each QoS mode of application i;
- $\mathbf{S}^{(i)}$ denote the vector-representation of the security level corresponding to the security mode s of application i;
- $k_{v_1,v_2}^{(i)}$ denote the QoS penalty for switching from QoS mode v_1 to v_2 for application i;
- $C_v^{(i,r)}$ denote the computation-time of $\mathscr{A}^{(i)}$ when running at QoS mode v on resource $\mathscr{R}^{(r)}$; these values do not include the computation-time needed for securing communications;
- $D_s^{(i,r)}$ denote the additional computation-time of $\mathscr{A}^{(i)}$ when running at security mode s on resource $\mathscr{R}^{(r)}$;
- $E_{v,s}^{(i,r)}$ denote the data size needed to be transmitted by application $\mathscr{A}^{(i)}$ from resource $\mathscr{R}^{(r)}$ every period $T^{(i,r)}$, when configured with QoS mode v and security mode s;
- $\mathbf{B}_Q^{(i,r)} \equiv \left[\frac{C_v^{(i,r)}}{T^{(i,r)}}\right]_v$ denote the vector of computation-bandwidth requirements for application $\mathscr{A}^{(i)}$ on resource $\mathscr{R}^{(r)}$, due to each QoS mode;
- $\mathbf{B}_S^{(i,r)} \equiv \left[\frac{D_s^{(i,r)}}{T^{(i,r)}}\right]_s$ denote the vector of additional computation-bandwidth requirements for application $\mathscr{A}^{(i)}$ on resource $\mathscr{R}^{(r)}$, due to each security mode;
- $\mathbf{B}_N^{(i,r)} \equiv \left[\frac{E_{v,s}^{(i,r)}}{T^{(i,r)}}\right]_{v,s}$ denote the network-bandwidth requirements for application $\mathscr{A}^{(i)}$ on resource $\mathscr{R}^{(r)}$, as due to each possible QoS and security mode.

We assume that security levels can be totally ordered according to the security strength they provide. If level L is stronger than L' we denote it by $L \succ L'$. If they have the same strength, we denote it by $L \equiv L'$. We assume that security modes induce a total ordering on security levels, i.e., for $\forall i, j \in \mathscr{V}_S$, then: i) $i > j$ iff $L_i \succ L_j$; ii) $i = j$ iff $L_i \equiv L_j$.

4.2. Problem Formalisation

Now we can formally define the optimisation problem. The variables of the problem are now introduced. Let:

- $\mathbf{v}_Q^{(i)} \equiv \left[v_Q^{(i,j)}\right]$ denote the vector-representation of the QoS mode to be chosen for the application $\mathscr{A}^{(i)}$; also, let $\tilde{\mathbf{v}}_Q^{(i)} \equiv \left[\tilde{v}_Q^{(i,j)}\right]$ denote the vector-representation of the current QoS mode (chosen for $\mathscr{A}^{(i)}$ at the previous optimisation step);
- $\mathbf{v}_S^{(i)} \equiv \left[v_S^{(i,j)}\right]$ denote the vector-representation of the security mode to be chosen for the application $\mathscr{A}^{(i)}$.

The problem is formulated by defining a *utility function* for the system that the global optimisation algorithm tries to maximise. The utility is defined as a weighted sum of terms related to:

- the QoS level (positive contribution) $\mathbf{Q}^{(i)} \cdot \mathbf{v}^{(i)}$ due to the application QoS modes. Applications do not have the same importance, therefore each one of these terms is weighted by a factor of $\phi_Q^{(i)}$ in the utility function;

Maximise over the $\left\{\mathbf{v}_Q^{(i)}\right\}, \left\{\mathbf{v}_S^{(i)}\right\}$ variables the function:

$$\sum_{i=1}^{N} \phi_Q^{(i)} \mathbf{Q}^{(i)} \cdot \mathbf{v}_Q^{(i)} + \sum_{i=1}^{N} \phi_S^{(i)} \mathbf{S}^{(i)} \cdot \mathbf{v}_S^{(i)}$$

$$- \sum_{i=1}^{N} \tilde{\phi}_Q^{(i)} k_V^{(i)} \cdot \left| \mathbf{Q}^{(i)} \cdot \mathbf{v}_Q^{(i)} - \mathbf{Q}^{(i)} \cdot \tilde{\mathbf{v}}_Q^{(i)} \right|$$

subject to the constraints:

$$\begin{cases} \left(\mathbf{B}_Q^{(i,r)} \cdot \mathbf{v}_Q^{(i)} + \right. & \\ \left. \mathbf{B}_S^{(i,r)} \cdot \mathbf{v}_S^{(i)}\right) \leq U_P^{(r)}, & r = 1, \ldots, N_P \\ \sum_{i=1}^{N_A} \mathbf{v}_S^{(i)} \cdot & \\ \left(\sum_{r \in \mathscr{H}_s} \mathbf{B}_N^{(i,r)} \cdot \mathbf{v}_Q^{(i)}\right) \leq U_S^{(s)}, & s = 1, \ldots, N_S \\ \sum_{j=1}^{V_Q^{(i)}} v_Q^{(i,j)} = 1, & i = 1, \ldots, N_A \\ \sum_{j=1}^{V_S^{(i)}} v_S^{(i,j)} = 1, & i = 1, \ldots, N_A \\ v_Q^{(i,j)} \in \{0,1\}, & i = 1, \ldots, N_A, \\ & j = 1, \ldots, V_Q^{(i)} \\ v_S^{(i,j)} \in \{0,1\}, & i = 1, \ldots, N_A, \\ & j = 1, \ldots, V_S^{(i)} \end{cases}$$

Figure 2. Formal problem formulation. The product "·" among vectors and matrices is defined as the usual matrix multiplication among the first operand transposed and the second one.

- the variation in QoS level (negative contribution) $\mathbf{Q}^{(i)} \cdot \mathbf{v}_Q^{(i)} - \mathbf{Q}^{(i)} \cdot \tilde{\mathbf{v}}_Q^{(i)}$ due to the changes in QoS modes of the applications. For some application classes, such as in video streaming, it is not convenient to change too frequently the QoS operation mode (e.g. display resolution), otherwise the user perceives a QoS level that is even lower than the one that would be perceived by keeping a low QoS but constant in time operation mode; each one of these terms is weighted by a factor of $\tilde{\phi}_Q^{(i)}$;
- the security level (positive contribution) $\mathbf{S}^{(i)} \cdot \mathbf{v}_S^{(i)}$ due to the application security modes. Generally, security requirements may be quite different for different applications, therefore each one of these terms is weighted by a further factor of $\phi_S^{(i)}$.

The weights introduced above in the utility function allow one to realise the appropriate trade-offs between QoS levels and security levels at each system (re)configuration.

The problem formulation in its general form is reported in Figure 2. The set of constraints on the decision variable is explained as follows. The first N_P constraints are to ensure CPU schedulability constraints, i.e., to execute the tasks implementing the applications within the real–time constraints. The use of real–time scheduling algorithms for each processor $\mathscr{R}^{(r)}$ guarantees this property as far as the computation-load due to both the application and the security modes does not exceed the capacity $U_P^{(r)}$, which depends on the scheduling algorithm. The second group of N_S constraints are on the network capacity: the overall traffic generated by applications running on resources connected to a given subnet s must not exceed the capacity $U_S^{(s)}$. The third group of $2N_A$ constraints have a technical nature. Since we use boolean variables to encode the application and the security modes, we must ensure that they are mutually exclusive: only one application or security mode can be active at a given time. Thereby, only one of the boolean variables in the corresponding vector-notation is equal to 1 and all others to 0. The last set of constraints ensure that the introduced problem variables, $v_Q^{(i,j)}$ and $v_S^{(i,j)}$, are boolean.

The formulated problem can be easily transformed into a boolean linear programming (BLP) optimisation program, thus it can be solved by recurring to standard tools.

5. Implementation Issues

In this section, we describe our low level design choices that make for a straightforward implementation of the proposed framework on a real system. The purpose of this section is to show that the theoretical application model presented above, and the run-time environment and application model it applies to, are all elements which are concretely usable in real embedded systems. Even though in this work we have not yet come to a complete implementation of the system, we are confident that this result can be obtained following the lines described below.

5.1. The Scheduling Mechanism

The technique proposed in this paper requires a scheduling mechanism with the following features: 1) guaranteeing system schedulability with a simple test based on system utilisation (as in the first constraint of problem in Figure 2), 2) ensuring a safe transition between different modes that change the amount of CPU required by the tasks, where by “safe” we mean that the tasks should never violate their real-time constraints. Additionally, it is useful to require that each task received the amount of resources associated with its current mode regardless of the behaviour of the other tasks. In other words, the effects of temporal faults have to be contained to the application that generates the faults, according to the well known principle of temporal isolation [1].

A scheduling solution that fulfils these goals is the resource reservations, which permits to specify the amount of resource (bandwidth) that the application receives. An implementation of this scheduling algorithm that can be used for the purpose, is the one offered by the portable FRESCOR framework[2]. This framework offers real-time contracts, implementing the resource reservation model for the CPU, network access and disk access, and it is available on a number of different platforms including Linux. For Linux, it is also possible to exploit the real-time scheduler [4] developed more recently in the context of the IRMOS European project, which can also be plugged inside the FRESCOR framework thanks to the availability of the same user-space API (the AQuoSA [14] API).

5.2. Multi-mode Applications

Applications with dynamically varying multi-mode capabilities are commonplace in real-time systems, both in the domain of multimedia (e.g., audio/video streamers) and industrial control. For instance, in real-time control different algorithms can be used that obtain the same control goal with a different quality. A very easy possibility is to increase the sampling period, which inevitably leads to a performance improvement. More sophisticated multi-mode behaviours can be obtained by using more sophisticated algorithms (e.g., H_∞) which have an impact on the computation time or on the sensing requirements. The different QoS can be easily mapped to system level metrics (e.g., the time required for a machinery to complete a goal).

A software infrastructure allowing multiple multi-mode applications to switch dynamically their QoS mode of operation has been developed again in the context of the FRESCOR Project, and specifically in the high-level QoS management layer [6]. The FRESCOR QoS Manager features also a

[2] More information is available at: `http://www.frescor.org`.

mode	$\mathsf{sizeof}(k)(bits)$	T_e (μs)
1	128	108.6
2	192	126.1
3	256	143.1

Table 1. Computing overhead for securing a message of 12 bytes (with an AES cipher block size of 16 bytes).

mode-change protocol for reconfiguring the whole system configuration at run-time. For example, in one of the final demonstrators of the project, we were able to easily modify the vlc multimedia player[3] for Linux so as to take advantage of the framework.

Even if the control of security modes was not in the original design, it is not difficult to modify the realised architecture to support this additional feature. More details on multi-mode security protocols are offered below.

5.3. *Security*

Security levels can be implemented in several different ways. In this work we implement security levels by means of the same cipher E and associating different levels of data transmission protection to different key lengths and thus different keys. Longer keys specify stronger security levels and, correspondingly, they are more demanding in terms of resource consumption. We denote by k_i the key associated to mode i and thus security level L_i and by $\mathsf{sizeof}(k_i)$ its length in bits. It follows that a security mode i specifies a couple $\langle L_i, k_i \rangle$ such that for $\forall i, j \in \mathcal{V}_S$, then: i) $i > j$ iff $(L_i \succ L_j) \wedge (\mathsf{sizeof}(k_i) > \mathsf{sizeof}(k_j))$; ii) $i = j$ iff $(L_i \equiv L_j) \wedge (k_i = k_j)$. When an application is in the security mode i then it uses the key k_i.

Estimation of resource consumption of every mode depends on the adopted cipher, the key length and other implementation choices. We discuss these issues in Section 5.3.1.

As each security mode is associated with a specific key, it is crucial that two agents are in the same mode to effectively communicate. Otherwise, secured messages transmitted by one would not be correctly unsecured by the other. In Section 5.3.2 we briefly discuss a mode switching protocol.

5.3.1. *Implementation issues and resource consumption*

We consider three security modes and implement them by means of the AES cipher (CBC mode) that supports a block-size $b = 128$ bits and three different key lengths, namely 128, 192 and 256 bit. In the rest of this section we estimate the resource consumption associated with each mode.

In order to implement V_S security modes, we have two choices with a different trade-off between storage and computation. The first implementation choice consists in having a single cipher instance in memory and, upon switching to the next security mode, re-initialise it dynamically with the key associated with the next mode. The other choice consists in having V_S instances of the cipher in memory, one for each mode, and initialising each of them with the corresponding keys before the mission starts. The latter solution has a larger storage overhead but prevents us from paying the computing overhead related to cipher initialisation upon every security mode switch. Such an initialisation overhead may be quite relevant depending on the cipher. For instance, in Blowfish, each new key requires a pre-processing equivalent to encrypting about four kilobytes of text.

Given the amount of RAM generally available on mid-size embedded systems that can run an OS like Linux, and the low number of different keys we are considering, we consider the memory requirements issue negligible, thus we focus on the latter implementation option.

The resource consumption of a security mode has two dimensions, computation and communication. The computation time consists in the time T_e necessary to parse and encipher (secure/unsecure) messages. As parsing is generally negligible with respect to enciphering, we neglect it. Message enciphering depends on the key length. As an example, Table 1 shows the computation overhead of

[3] More information available at: `http://www.videolan.org/vlc/`.

every security mode for a payload size equal to 12 bytes, in the case of using AES measured on an Olimex SAM9-L9260 board. The payload is the one of the case study in Section 6.

The communication overhead consists in the additional network bandwidth necessary for the transmission of a secured message due to *cipher-text expansion*. Such an expansion is due to the requirement that in block cipher the clear-text size, and thus the cipher-text size, must be multiple of the block size. Thus padding is necessary. It follows that if p if the size of the payload to be secured, the length of the cipher text is $\lfloor \frac{p+b}{b} \rfloor$. This overhead is independent of the key size. For instance, in the case of AES, a payload size equal to 12 bytes introduces a communication overhead of roughly 33.3%.

5.3.2. *Security Level Switching Protocol*

Robots must switch from one security mode to another in a coordinated way or message loss may ensue. If two robots are in different security modes they actually use two different keys. Therefore, the ciphertext produced by one cannot be correctly decrypted by the other, and vice versa. In order to avoid this problem we consider a *Security Mode Switching Protocol* that takes place under the control of a protocol Coordinator C.

We assume that robots and coordinator agree upon a given keyed-hash message authentication code *hmac*. Furthermore, we assume that robots and coordinator share a secret group key *gk*. Coordinator and robots maintain a counter, initially equal to zero, that counts the number of mode switches have occurred so far. The Security Level Switching Protocol takes the following steps.

Upon switching to the security mode i, the protocol Coordinator increases its counter cnt_C by one, computes the authenticator $h \leftarrow hmac_{gk}(i \| cnt_C)$, where $\|$ denotes the concatenation operator, and finally broadcasts the control SWITCH message $m : (i, cnt_C, h)$.

Upon receiving a SWITCH message, a robot r verifies the authenticator $m.h$ by means of the group key *gk*, and checks that its own counter cnt_r is smaller than the value of the counter field in m, $cnt_r < m.cnt$. If both checks succeed, then the robot sets cnt_r equal to $m.cnt$, and switches into the security mode $m.i$. Otherwise, the robot drops the message.

Due to communication errors, an robot may fail to receive a SWITCH message. There are several possible approaches to recover from this kind of omission failures which depend on several factors including the number of robots, their speed, and the noisy of the wireless channel. In the case study reported in Section 6 we employ the simple approach sketched in the following. If an robot fails to receive the SWITCH message, then it remains in a stale security mode and keeps using a stale key. This causes decryptions to fail. Any robot whose decryptions fail contacts the protocol coordinator to have a SWITCH message broadcast again. Evaluation of this approach and a comparison with other possible approaches is in progress.

6. A case study

To show the effectiveness of the approach, we have considered a realistic case study inspired by our industrial partner SOFIDEL s.p.a.. A group of mobile robots is used in a paper mill to move paper rolls between the working stations and the logistic areas. The robots are laser guided vehicle (LGV) and move along predefined routes in the factory plant. Whenever two robots are in proximity, a collision avoidance protocol is engaged that requires a frequent exchange of messages. Robots are also required to timely report their positions and other information that is used by the plant supervisor. A situation like this is representative of a larger class of applications that expose the critical tradeoff between performance and security. To carry out an extensive evaluation of our idea, we have implemented a co-simulator that, at the same time, simulates the physical dynamics of the robots and an abstraction of the computation and communication platform utilised by the applications.

6.1. Simulator Description

The simulation tool takes as input an XML file describing the simulation scenario. In this file, we specify the layout of the plant, the possible missions of the robots, the physical features of the robots and the parameters required to set up the optimisation program (number of applications, application levels, QoS, etc.).

A factory is a collection of rooms, connected to each other by means of *gates*. Figure 3 shows a map of factory (a paper mill) consisting of four rooms. Inside each room we define the possible paths as a graph of vertices connected by links (which correspond to physical segments inside the plant). A vertex is defined specifying its coordinates and the vertex type. The type is one of the following: 1) **Ordinary via points** - These points are used to specify the possible paths inside the factory. Robots can move along links connecting different via points. When a robot reaches a via point, it can choose any link departing from the point. In doing so, the robot can proceed in a straight direction or make a turn (to the right or to the left). 2) **Gate** - (red circles in the figure) this type of via point is used to mark a passage between different rooms; 3) **Station** (black circles) - specify parking locations of the robots when they are not executing any tasks. Each trajectory starts and ends at a station. 4) **Goal** (blue circles) - the destination of the robot. A goal is associated to a picking or storing place for the products. A goal planner in the simulator decides tasks to be accomplished and assigns them to the free robots in the stations. The task is executed selecting a trajectory from a predefined set departing from the station and linking the station to the goal.

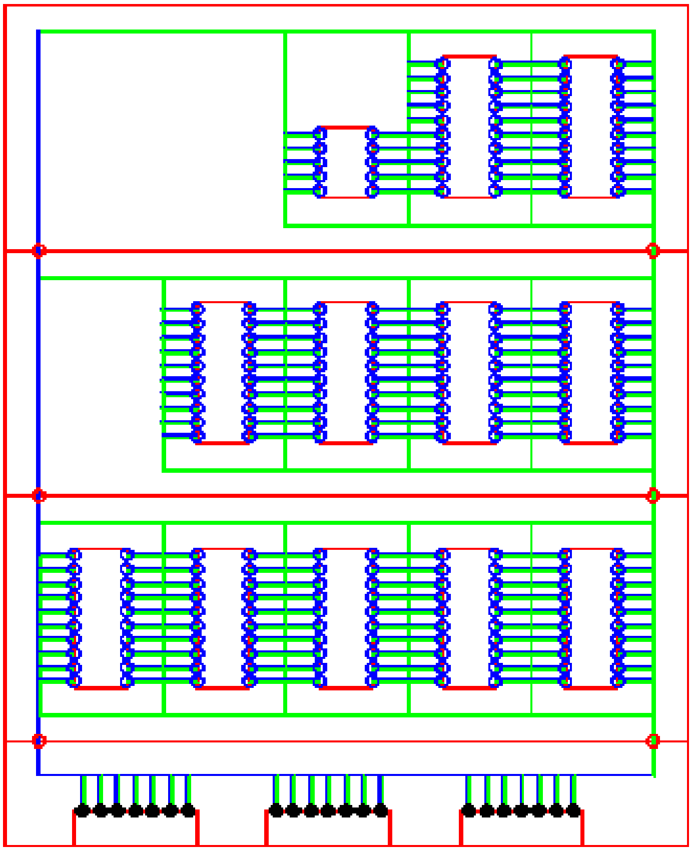

Figure 3. Factory layout with possible robot moving directions (snapshot of the simulator’s GUI).

In the XML file, we specify the different resources and their capacity (using a formalism very similar to the one described in the previous sections). A robot is specified by providing its physical parameters (e.g., minimum and maximum speed) and the applications that it has to execute. An application is for the simulator an abstract entity which is associated to a list of modes and to a list of resources that it uses. For each mode we specify the Quality of Control and the bandwidth utilised on each of the resources. If the application uses network links, we need specify the security modes. In this case, resource utilisation is associated to the different combination of application and security modes.

The performance optimiser is implemented within the simulator using the GNU Linear Programming Toolkit[4] (GLPK), which is invoked to solve the problem formalised in Section 3.

The different missions can be specified by an additional file or extracted randomly. During the simulation, the tool record execution traces from which we can evaluate the Quality of Control and the Quality of Security accumulated by the different robots.

6.2. Simulation Scenario

In our case study, the available resources are the computing boards hosted on each robot and a wireless 802.11 connection. In particular, each room hosts a 802.11e access point, which schedules the transmission slots to the different robots. The application executed by the robots (as suggested by our partners) are the following.

1. **Motion control** which is a line-following application used by the robot to stay on the assigned path. The application can run in three different modes, which use different algorithms and sampling rates. Depending on the choice of the modes, the robots to remain within a specified and bounded distance from the ideal path with a different velocity (the velocity can therefore be considered as a QoC metric). For our application we assumed that the velocity associated with the different modes are, respectively, given by: $2\frac{m}{s}$, $1\frac{m}{s}$ and $0.5\frac{m}{s}$. Since this application runs locally on the robot, there is no security mode associated.
2. **Tracking system** which monitors the environment to look for potential obstacles. Three discrete modes specify a different size of the monitored zone. Even in this case, there is no need for security modes since the application runs locally.
3. **Diagnostics** which checks the different subsystems of the robot. The application can be run with a different frequency and can diagnose a different number of devices. Even in this case we identified three possible modes (but the number can be in fact larger).
4. **Reporting** which sends information about the mobile robot to the server. Different modes are related to a different update frequency and amount of data to be sent. In this case the application requires the use of the communication channels. Three security modes specifies the security level (high, medium, low) of communication protocol.
5. **Motion control (in presence of obstacles)** which controls the robot motion when other robots are in the proximity. In this case, a collision avoidance protocol has to be engaged that requires a frequent exchange of messages. Depending on the number of messages that are exchanged, the robot is able to move with a different velocity. Even in this case we identified three different modes associated respectively to $2\frac{m}{s}$, $1\frac{m}{s}$, $0.5\frac{m}{s}$). Three security modes specify the security level of communication protocol and have the influence on the robot velocity, the higher communication security the lower the velocity of the robot.

Clearly, the first and the fifth applications are mutually exclusive: the first is running when robot is moving in free space, the fifth when the presence of other robot in the near-field region is detected. The simulator allows us to easily redefine the notion of the near-field region which provides more flexibility for simulations. The computation of a new configuration by the performance optimiser is

[4] More information is available at: `http://www.gnu.org/software/glpk/`.

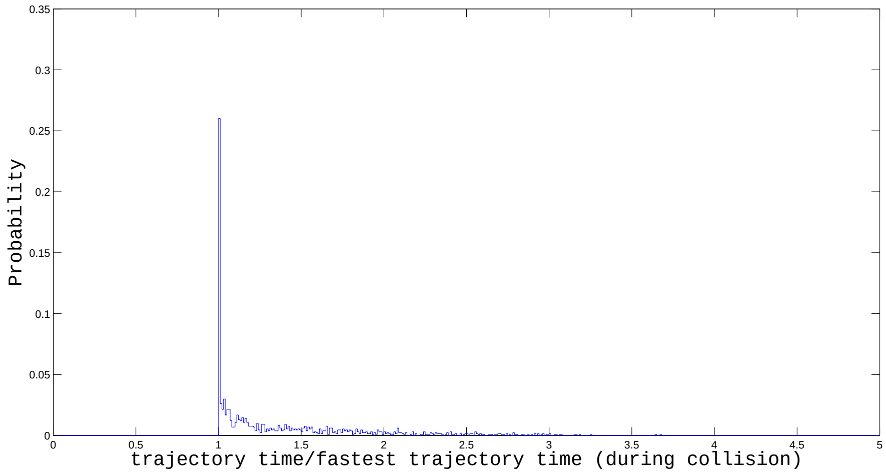

Figure 4. Probability mass function of the ratio between the real trajectory time (measured during simulations) and the fastest trajectory time. Priority is given to the speed.

Priority \ Security	High	Medium	Low
Speed	30%	3%	67%
Security	88%	7%	5%

Table 2. Probabilities of different security levels in the presence of possible collision.

required whenever the state of the simulation changes. This is the consequence of one of the following events: 1) a robot is assigned a mission and starts moving, 2) a robot changes the room in which it moves, 3) two robots enter in the near-field of each other and generate a potential conflict, 4) a conflict condition terminates because the distance between the robot becomes greater than an assigned value.

6.3. Simulation Results

In the experiments reported below, we have considered 50 robots moving in the example factory shown in Figure 3. In this application there is a clear and hardly disputable performance metric: the time required to complete a task. On the other hand, we can evaluate the security in the plant by directly associating a different value to each security level. In the specific example, we set 0, 10 and 100 for the three security levels in their turn. By using a different weight we can attach more importance to the system performance or to its security. The relative weights can even changed at run-time (e.g., if a potential intrusion is detected).

In a first set of experiments, we gave a higher priority to the QoC. Consequently, in the cost function we have chosen the weight related to the system performance was set 10 times the weight related to the communication security. As a performance indicator we chose the ratio ρ between the best possible performance (the completion time requested for the missions if the robots are always able to proceed at their maximum speed) and the one that we get in the different simulation. This is clearly a random variable whose level depends on the number of conflicts and on the levels set by the performance optimiser. Figure 4 shows the experimental probability mass function (PMF) for ρ. The very tall peak at 1 shows that in most cases the robots complete their task in the minimum time. Inverting the priority between performance and security (ratio between performance and security set equal to 0.1) resulted in the PMF shown in Figure 5. The distribution of ρ is shifted to the right, meaning that the increased resources spent for security determine longer completion time for the robot missions.

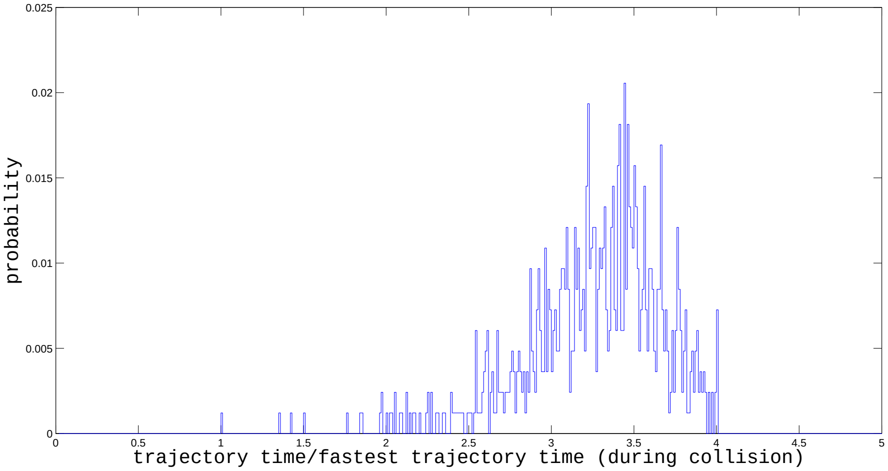

Figure 5. Probability mass function of the ratio between the real trajectory time (measured during simulations) and the fastest trajectory time. Priority is given to the security.

Table 2 shows, for this scenario, the time spent in each of the different modes (both for the application and for security) when a conflict between two robots occurs. As we can see, in case of a conflict, most of the communications (88%) take place with the maximum security mode and in the lowest application mode (67%). This is a consequence of the higher weight attached to system security.

7. Conclusions and Future Work

In this paper, we have shown an adaptive management of computation and communication resources which enables to set different tradeoff points between security and quality of control. The idea has been displayed on a simulated industrial case study, where different robots carry goods between the different points of the factory floor using shared computation and communication resources. A fundamental requirement for our approach is that the applications be able to operate in a set of different discrete modes and the communications can be executed using a discrete set of levels. The different security levels are obtained by using different key lengths.

Our future work will be in different directions. First, we will explore different ways for building a multiple level security policy. Second, we will study distributed ways for implementing the performance optimiser (taking inspiration from the recent literature on distributed optimisation). Finally, an experimental setup is currently under construction that will allow us to study the performance of our idea in a real–life implementation.

Acknowledgements

This work has been supported by the European Community’ within the Seventh Framework Programme under grant agreements CHAT, “Control of Heterogeneous Automation Systems” (IST-2008-224428), CONET, “Cooperating Objects Network of Excellence” (FP7-2007-2-224053), and PLANET, “Platform for the Deployment and Operation of Heterogeneous Networked Cooperating Objects” (FP7-257649).

References

[1] Luca Abeni and Giorgio Buttazzo. Integrating multimedia applications in hard real-time systems. In *Proceedings of the IEEE Real-Time Systems Symposium*, Madrid, Spain, December 1998.

[2] Luca Abeni, Tommaso Cucinotta, Giuseppe Lipari, Luca Marzario, and Luigi Palopoli. Qos management through adaptive reservations. *Real-Time Systems Journal*, 29(2-3), March 2005.

[3] Alvaro A. Cardenas, Saurabh Amin, and Shankar Sastry. Research challenges for the security of control systems. In *Proceedings of the Third USENIX Workshop on Hot Topics in Security (HotSec'08)*, San Jose (CA), USA, July 28 2008.

[4] Fabio Checconi, Tommaso Cucinotta, Dario Faggioli, and Giuseppe Lipari. Hierarchical multiprocessor CPU reservations for the linux kernel. In *Proceedings of the* 5^{th} *International Workshop on Operating Systems Platforms for Embedded Real-Time Applications (OSPERT 2009)*, Dublin, Ireland, June 2009.

[5] T.M. Chen. Stuxnet, the real start of cyber warfare. *IEEE Network*, 24(6):2–3, November-December 2010.

[6] Tommaso Cucinotta, Giuseppe Lipari, Luigi Palopoli, Luca Abeni, and Rodrigo Santos. Multi-level feedback control for quality of service management. In *Proceedings of the* 14^{th} *IEEE International Conference on Emerging Technologies and Factory Automation*, Mallorca, Spain, September 2009.

[7] J.P. Hansen, J. Lehoczky, and R. Rajkumar. Optimization of quality of service in dynamic systems. In *Parallel and Distributed Processing Symposium., Proceedings 15th International*, pages 1001–1008. IEEE, 2001.

[8] C. Irvine and T. Levin. Quality of security service. In *Proceedings of the 2000 workshop on New security paradigms*, pages 91–99. ACM, 2001.

[9] Kyoung-Don Kang and Sang H. Son. Towards security and QoS optimization in real-time embedded systems. *SIGBED Rev.*, 3:29–34, January 2006.

[10] Man Lin, Li Xu, L.T. Yang, Xiao Qin, Nenggan Zheng, Zhaohui Wu, and Meikang Qiu. Static security optimization for real-time systems. *IEEE Transactions on Industrial Informatics*, 5(1):22–37, Feb. 2009.

[11] John Markoff. Malware Aimed at Iran Hit Five Sites, Report Says. The New York Times, February 11 2011.

[12] G. Nair, F. Fagnani, S. Zampieri, and R. Evans. Feedback control under data rate constraints: an overview. *Proceedings of The IEEE*, 95:108–137, 2007.

[13] L. Palopoli, C. Pinello, A. Bicchi, and A. Sangiovanni-Vincentelli. Maximizing the stability radius of a set of systems under real-time scheduling constraints. *Automatic Control, IEEE Transactions on*, 50(11):1790–1795, Nov. 2005.

[14] Luigi Palopoli, Tommaso Cucinotta, Luca Marzario, and Giuseppe Lipari. AQuoSA — adaptive quality of service architecture. *Software – Practice and Experience*, 39(1):1–31, 2009.

[15] Sooyeon Shin, Taekyoung Kwon, Gil-Yong Jo, Toungman Park, and H. Rhy. An experimental Study of Hierarchical Intrusion Detection for Wireless Industrial Sensor Networks. *IEEE Transactions on Industrial Informatics*, 6(4):744–757, November 2010.

[16] Tao Xie and Xiao Qin. Scheduling security-critical real-time applications on clusters. *IEEE Transactions on Computers*, 55(7):864–879, July 2006.

[17] Tao Xie and Xiao Qin. Improving security for periodic tasks in embedded systems through scheduling. *ACM Trans. Embed. Comput. Syst.*, 6(3), July 2007.