A scalable platform for safe and secure decentralized traffic management of multi-agent mobile systems

Antonio Danesi
Interdepartmental Research Center “E. Piaggio”
Faculty of Engineering
Pisa, Italy

Adriano Fagiolini
Interdepartmental Research Center “E. Piaggio”
Faculty of Engineering
Pisa, Italy

Ida M. Savino
Dipartimento di Ingegneria dell’Informazione
Faculty of Engineering
Pisa, Italy

ABSTRACT

In this paper we describe the application of wireless sensor networking techniques to address the realization of a safe and secure decentralized traffic management system.

We consider systems of many heterogeneous autonomous vehicles moving in a shared environment. Each vehicle is assumed to have different and possibly unspecified tasks, but they cooperate to avoid collisions. We are interested in designing a scalable architecture capable of accommodating a very large and dynamically changing number of vehicles, guaranteeing their safety, the achievement of their goals, and security against potential adversaries. By properly distributing and revoking cryptographic keys we are able to protect communications from an external adversary as well as to detect non-cooperative, possibly malicious vehicles and trigger suitable countermeasures. In our architecture, scalability is obtained by decentralization, i.e. each vehicle is regarded as an autonomous agent capable of processing information concerning its own state and the state of only a fixed, small number of “neighboring” agents. Ad-hoc wireless sensor networks are employed to provide support for this architecture.

Categories and Subject Descriptors

C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design; I.2.9 [**Artificial Intelligence**]: Robotics—*Autonomous vehicles*

General Terms

Real Sensor Networks

Keywords

Wireless network platform, Mobile agents, Safety, Security

1. INTRODUCTION

The convergence of communication, computing and control is considered by many the future of information technology [1–3]. This will provide the ability for large numbers of sensors, actuators, and computational units, all interconnected wirelessly or over wires, to interact with the physical environment. One of the main expected consequences of such convergence is the possibility to create large systems of many autonomous and interconnected units, which not only have capabilities of sensing [4], but also of acting in and on the environment. At this point, a critical role will be played by the architecture that will support such systems.

In this paper, we consider the development of strategies and architectures for supporting safe, secure, and scalable management of autonomous mobile systems [1, 5]. In the scenario we consider, a large group of heterogeneous vehicles move in a common workspace. Every agent of the group has a specific task to accomplish, by its own or in collaboration, such as monitoring the environment, reconstructing a map, sensing the environment, or detecting light or heat sources, etc. Agents can join and/or leave the group dynamically. Typical agents are inexpensive, unmanned vehicles equipped with embedded sensor systems, with limited on-board processing units and short-range wireless communication capabilities.

The main specification of the architecture is *safety*, i.e. to avoid collisions of the mobile agents while they attend to their tasks. The requirement on *scalability* of the architecture imposes the absence of a centralized traffic supervisor dispatching detailed instructions to all agents. Rather, the architecture should be based on a decentralized strategy hinging upon a set of “traffic rules” shared among agents. These traffic rules, or *collision avoidance protocol* (CAP), must enable each agent to autonomously make decisions about its own motion, based on exact and correct information on its own state and the state of only a fixed, small number of “neighboring” agents. Ad-hoc wireless network technologies are apparently good candidates to provide support for such a platform. Another requirement for a CAP to be useful in realistic application scenarios is the allowance for *heterogeneity* of vehicles. More generally, we are interested in protocols by which vehicles integrating hardware and software of completely different origins, can safely coexist and collaborate.

In many applications, agents can join and leave the group depending on their private tasks. To guarantee *dynamic reconfigurability* of the architecture, it is necesary to provide means of safe check-in. The CAP should describe conditions under which a candidate new member of the group can be accepted without endangering the group’s safety.

Furthermore, the group could share information related to the agents’ tasks, which might have to be protected against

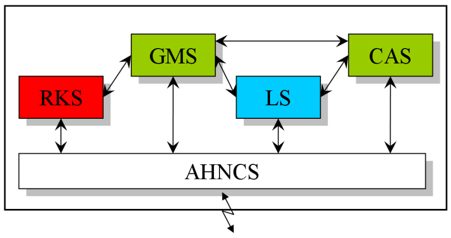

Figure 1: Software modules comprising the architecture of an agent.

eavesdropping or even malicious tampering. These requirements imply the need for *security* of communications. As is well known, protecting a wireless communication network poses unique challenges. First, unlike traditional wired networks, an external adversary with a simple radio receiver/transmitter can easily eavesdrop as well as inject/ modify packets. Second, in order to make the system economically viable, agents are limited in their on-board capabilities and this practically excludes the use of public-key cryptography.

In this paper, we propose a platform for safe, secure, and scalable management of heterogeneous multi-agent mobile systems. The platform employs a CAP based on the so–called *generalized roundabout policy* (GRP), previously described in [6]. In the GRP-CAP, decisions are made based on information about a maximum of six neighbouring agents. The required information essentially consists of only the position and heading angle of the agents (referred to as their state). Communication among neighboring agents is based on ad-hoc wireless networking. The platform guarantees *security* of communications within the group with respect to an external adversary by distributing a *group-key* to be used to encrypt messages broadcast within the group. When an agent leaves the group, the current group-key must be revoked and a new one distributed to all agents but the leaving one (*forward security*). It follows that an agent failure to provide the correct group-key can be interpreted as an alarm by the system, which triggers countermeasures (not described in this paper).

As the platform must scale to a large number of agents, we adopt the rekeying protocol proposed in [7]. This protocol is efficient in terms of computation costs because it uses primitives with low-computational overhead such as symmetric cyphers and one-way hash functions. Furthermore, unlike other existing solutions, the chosen protocol improves the scalability because its communication overhead is a logarithmic function of the network size.

2. PLATFORM ARCHITECTURE

From an agent perspective, the proposed architecture for decentralized, safe, and secure traffic management of mobile robots is composed of several services (see Figure 1). The *Collision Avoidance Service* (CAS) is responsible for coordinating the motion of agents within the group. In order to avoid collisions among them, the CAS needs to know the agent localization as well as the one of the neighbors. These two functionalities are provided by the *Localization Service* (LS). Whenever a new agent joins the group, its entrance may compromise the safety of existing agents, or prevent some agents to complete their tasks. For this purpose, a *Group Membership Service* (GMS) is introduced for guaranteeing that all agents reach their own final destinations. When an agent leaves the group, the GMS prompts the *Re-Keying Service* (RKS) to distribute a new group-key to all group members. All these services use the *Ad-Hoc Networking Communication Service* (AHNCS) that provides secure communication, and scheduling of the wireless communication medium. The proposed services will now be described in detail. The reported details represent the minimal set of requirements which any agent have to fulfil in order to be consistent with the architecture.

2.1 CAS: Collision Avoidance Service

The collision avoidance service coordinates the motion of agents within the group, preventing collisions and guaranteeing that each agent eventually accomplishes its individual task. The service implementation is based on a decentralized collision avoidance protocol, called “generalized roundabout policy” (GRP), that has been recently proposed for mobile agents evolving on the plane [6]. The GRP is now briefly reported for the reader convenience. However, a complete, formal and detailed description of it can be found in the cited literature.

Consider a number of mobile agents moving on the plane at constant speed, along paths with bounded curvature. The state of each agent is represented by the coordinates (x, y) and the heading angle θ. According to the protocol a first circle is assigned to each agent, called the *safety disk*, being the circle centered at the agent position (x, y) with heading given by θ. A *collision* is said to occur whenever two or more safety disks overlap.

The protocol applies also to vehicles that can not stop their motions. For dealing with such a case, the protocol defines a *reserved disk* for each agent as the circle that contains the path traveled by the safety disk, when its associated agent turns right at the minimum allowable curvature. The center of a reserved disk can easily be obtained from its agent state, and its heading is directly inherited from that of the corresponding agent. In spite of the agent constraint, the motion of the reserved disk can be stopped at any time, by making the agent turn right at the minimum curvature rate.

Suppose that each agent has to reach a desired final position and heading to accomplish its task. The motion strategy followed by the agent is based on four distinct *modes of operation*, each assigning a suitable value to the curvature rate of the agent. Figure 2 shows these operation modes along with the corresponding switching conditions, a.k.a. *guards*. With reference to the figure, each agent enters the **straight** mode if the motion along the line directed as the agent heading is permitted, i.e. its reserved disk does not overlap with other reserved disks. During this mode, the agent's curvature rate is set to zero. Whenever its reserved disk becomes tangent to the one of another agent, a test is made based on the current motion heading θ. If a further movement in the direction specified by θ causes an overlapping, then the agent enters the **hold** mode. Otherwise, the agent is able to proceed, and remains in the straight mode. When the hold mode is entered, the agent's curvature rate is set to the minimum allowable, and the motion of its reserved disk is stopped. As soon as the agent heading is permitted but not directed towards the target destination, the agent enters the **roll** mode, and tries to go around the other reserved disk. This is achieved by selecting a suitable value for the curvature rate of the agent such that the two disks

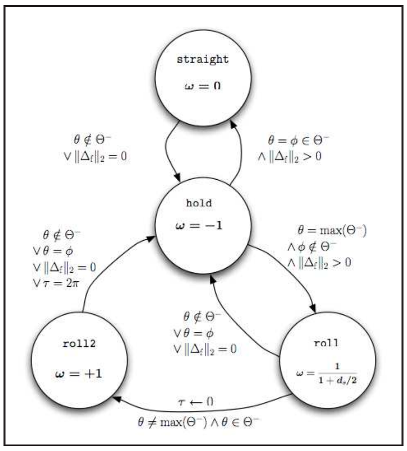
straight
$\omega = 0$
$\theta \notin \Theta^- \lor \|\Delta_f\|_2 = 0$
$\theta = \phi \in \Theta^- \land \|\Delta_f\|_2 > 0$
hold
$\omega = -1$
$\theta \notin \Theta^- \lor \theta = \phi \lor \|\Delta_f\|_2 = 0 \lor \tau = 2\pi$
$\theta = \max(\Theta^-) \land \phi \notin \Theta^- \land \|\Delta_f\|_2 > 0$
$\theta \notin \Theta^- \lor \theta = \phi \lor \|\Delta_f\|_2 = 0$
roll2
$\omega = +1$
roll
$\omega = \frac{1}{1 + d_s/2}$
$\tau \leftarrow 0$
$\theta \neq \max(\Theta^-) \land \theta \in \Theta^-$

Figure 2: Finite state automaton that summarizes the collision avoidance protocol GRP.

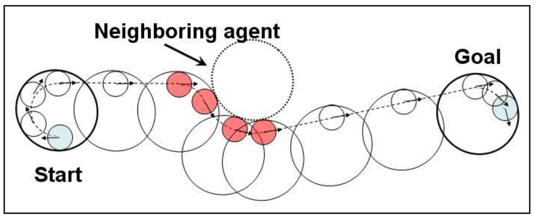

Figure 3: Example of possible trajectory of an agent applying the proposed collision avoidance protocol. Smaller circles are safety disks, larger circles are reserved disks.

never overlap. During the roll mode, the tangency of the two disks can unexpectedly be lost, then the agent enters the **roll2** mode, and the curvature rate is set to the maximum allowable in order to restore the contact. The roll2 mode can only be entered if the previous mode was roll. When the tangency is restored, the agent switches back to roll mode. An example of possible trajectory of an agent that moves according to the GRP is pictorially depicted in Figure 3.

The decentralized characteristic of the protocol allows the CAS to be implemented on board of the agent. As a matter of fact, each agent is able to make a safe decision about its motion, based only on the locally available information. This information consists of the position and orientation of agents that are within a certain sensing or communication radius. For this reason each agent communicates its state via the AHNCS, though it is not required to explicitly declare its task or goal.

2.2 GMS: Group Membership Service

The motion strategy described in Section 2.1 guarantees that no collision will occur among agents belonging to the group (safety property), and that all the agents eventually reach their final destinations (liveness property). These two properties are guaranteed provided that initial and final agents' configurations satisfy suitable conditions. In particular, safety is obtained if the agents' reserved disks do not initially overlap, whereas liveness is guaranteed if the agents' destinations are not concentrated in the plane [6].

Taking into account the fact that agents can dynamically

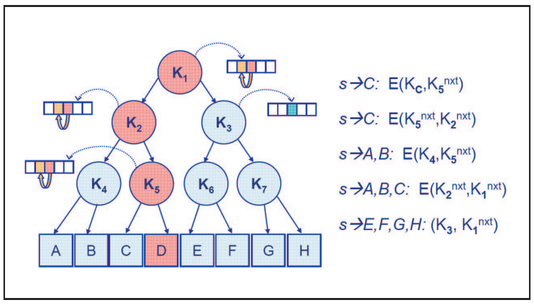
K_1
K_2
K_3
K_4
K_5
K_6
K_7
A
B
C
D
E
F
G
H
$s \rightarrow C$: $E(K_C, K_5^{nxt})$
$s \rightarrow C$: $E(K_5^{nxt}, K_2^{nxt})$
$s \rightarrow A,B$: $E(K_4, K_5^{nxt})$
$s \rightarrow A,B,C$: $E(K_2^{nxt}, K_1^{nxt})$
$s \rightarrow E,F,G,H$: (K_3, K_1^{nxt})

Figure 4: Hierarchical structure of key-chains for forward security.

join or leave the group, the GMS purpose is to guarantee that such conditions are never violated. Thus, upon joining, a new agent sends the configuration of its reserved disk to the GMS that verifies whether its entrance may compromise the overall system safety and liveness (join phase). Later, upon reaching its final destination, the agent leaves the group and alerts the GMS that cancels its data (leave phase). This event may allow new agents to enter the group. The GMS relies on the communication mechanism provided by the AHNCS. Moreover, it must be managed by a centralized server as the conditions guaranteeing safety and liveness are based on the information provided by all the agents.

2.3 RKS: Re-Keying Service

The Re-Keying Service is managed by a centralized *Key Distribution Server* which is responsible for guaranteeing the forward security. In fact, when an agent leaves the group communication, the server has to distribute the new group-key to all agents except the leaving one. Agents are simply responsible to verify the freshness and the authenticity of the keys coming from the server.

The key authentication mechanism levers on *key-chains*, a tecnique based on the Lamport's one-time passwords. A key-chain is a set of symmetric keys so that each key is the hash preimage of the previous one under a one-way hash function. Hence, given a key in the key-chain, anybody can compute all the previous keys, but nobody can compute any of the next keys. Keys are revealed in the reversed order with respect to creation. Given a authenticated key in the key-chain, the agents can authenticate the next keys by simply applying an hash function.

In order to reduce the communication overhead, the server maintains a tree structure of keys (Figure 4). The internal nodes are associated with key-chains, while each leaf is associated with a symmetric *agent-key*, that each group member secretly shares with the server. A group member stores the last-revealed key for every internal node belonging to the path from its leaf to the root. Hence, the key associated to the tree root is shared by all group members and it acts as the group-key. When the agent leaves the group communication, all its keys become compromised and have to be redistributed. For example, let us suppose agent D leaves the group. The server has to securely broadcast a new key for each internal node whose subtree contains the D leaf (e.g., nodes numbered with 1, 2, and 5). In case of binary tree, the server has to broadcast $2\log(n) - 1$ messages where n is the network size.

2.4 LS: Localization Service

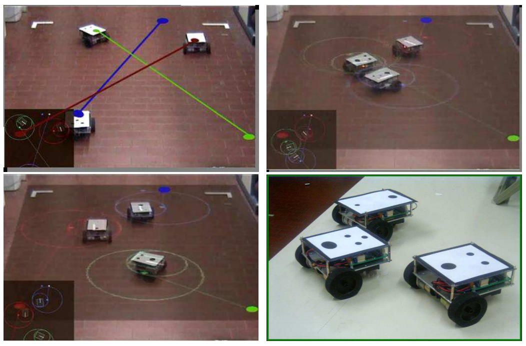

Figure 5: Three mobile robotic agents are assigned with final positions and headings which they have to reach in order to accomplish their tasks.

Two functionalities comprise the localization service of each agent: self-localization and peer-localization. The former provides the information about the state of the agent, whereas the latter is responsible for retrieving the states of neighboring agents. For self-localizing, each agent in general may take advantage of available on-board devices, or use an external localizing service. In both cases, the agent has to use the external service at least once to obtain a reference frame which is shared within the group. For the peer-localization we chose to rely on secure communication to obtain necessary data.

2.5 AHNCS: Ad-Hoc Networking Communication Service

The ad-hoc networking communication service (AHNCS) implements the secure communication mechanism, guarantees real-time bounds, and avoids starvation phenomena. The service provides periodic and aperiodic scheduling of messages, which may be useful to realize the LS. Moreover, the service itself protects communications against external adversaries by means of message encryption. Communications with the central authority are encrypted with the agent-key, whereas those among agents are encrypted with the group-key.

3. PLATFORM PROTOTYPE

This section describes a platform prototype that has been realized according to the proposed architecture. The platform is composed of a fixed main infrastructure, and a number of homogeneous mobile robotic agents. As already mentioned, our architecture implementation is tailored to a large number of low cost agents equipped with limited sensor systems. Indeed, agents prototype have been developed with such requirements. Details about agent hardware and software components are reported in the following subsections.

3.1 Agent prototype

Mobile robotic agents have been built, consisting of a chassis of $14cm \times 13cm \times 9cm$ size that hosts motors, batteries, and electronics. The agents are also equipped with a Tmote-Sky sensory board, which enables communications with the 802.15.4 protocol, and some PSoC Mixed-Signal Array Controllers that serve as servo-driving, odometry and CAS implementation. The Tmote-Sky board has been adopted for their high compatibility with the Zigbee protocol and low power consumption. An interface between microcontrollers and Tmote-Sky has been developed. In order to take advantage of every resources offered by all the units, the load of computing algorithms has been divided among the Tmote-Sky and PSoCs CPUs. With such approach performances have been improved with respect to the performance achievable with the only Tmote board. Indeed, a 40Hz CAS computation and a 200Hz servo driver control have been obtained. Extensive tests have been done on a test bed composed of three robotic agents (see Figure 5) as reported in section 4.

A consequence of the “low cost” implementation of the agent, the peer- and self-localization have not been implemented on board. Indeed, the excessive cost and the insufficient precision of available sensor technologies have induced us to implement the LS localization service. The self-localization is achieved by means of aperiodic requests to a fixed infrastructure localization service that relies on computer vision to identify the agents’ states. Furthermore, the peer-localization module is performed by listening periodic messages of other agents communicating data about position and reserved disk radius.

From a security perspective, each agent implements an early prototype of the rekeying protocol described in 2.3 by using SkipJack or RC5 as the symmetric cipher, and SHA-1 as the hash function.

3.2 Infrastructure

First of all, the infrastructure enables the LS by detecting the states of every agents, and providing the common reference frame shared within the group. Secondly, the infrastructure enables the RKS by generating new keys and distributing them when necessary.

Off-the-shelf cameras have been exploited for monitoring the environment. Vision algorithms have been developed to identify the state of every agent by means of markers placed over the chassis. By precisely calibrating the cameras, an accurate estimate of the position and orientation of each agent has been obtained. Despite of the low-cost cameras, the chosen algorithms are robust to illumination changes in an indoor test bed. Cameras and algorithms are hosted on a system composed of three PCs, connected in a LAN.

3.3 AHNCS: Communication Protocol

An ad-hoc wireless communication protocol has been implemented. As already stated, LS requires periodic communications for the peer-localization, while aperiodic communications are used by RKS, GMS, and LS for self-localization. AHNCS realizes a time-division multiple access protocol briefly described in the following.

A central authority is responsible of the temporal synchronization, and a time slice based subdivisions. In large multi-hop wireless networks, an accurate distributed time synchronization is a nontrivial problem [8]. For the sake of simplicity, the proposed implementation deals only with one hop communication. Each time slice is composed of $2N + K$ slots, where N is the number of agents, and $K \geq 2$ is an integer value used to avoid starvation. Furthermore, in worst case each slot must allow the transmission of a maximum-length packet extimated in ≈ 10 ms. Any time an agent joins the group or the group membership changes, the AHNCS assigns a slot index and the time slice duration to each agents. A time slice is composed of two phases: Periodic Communication Phase and Aperiodic Communication Phase (see Figure 6). The Periodic Communication

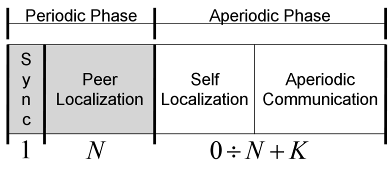

Figure 6: Time Slice structure.

Phase starts with a synchronization message, and it is composed of N slots. Every agent has its own slot to perform peer-localization (broadcast of position and reserved disk radius). Hence, at the end of this phase, all agents have collected information regarding neighboring agents. Within the same slot, an agent can request to the central authority for self-localization and permission for aperiodic communications. In the Aperiodic Communication Phase the central authority replies to requests for self-localization (in no more than N slots), and it gives acknowledgments to agents that performed request for aperiodic slot. An acknowledgment is a broadcast message that contains the identifiers of the allowed agents.

4. EXPERIMENTAL RESULTS

The effectiveness of the proposed architecture has been proved by several experiments. A scenario in which three agents were assigned with the task of reaching a final configuration has been considered. Snapshots from one experiment are shown in Figure 5.

In all the conducted experiments, agents have forward velocity of $5cm/sec$, angular velocity during the `hold` mode of $0.385rad/sec$, reserved disk radius of $13cm$ and safety disk diameter of $15cm$. On each agent a battery pack of $10.4V$, $1800mA$ has been mounted in order to provide energy to the Tmote-Sky and the PSoCs. With such power supply this kind of experiments can be conducted for at most $80min$. It is important to notice that during the experiments, intensive use of the wireless communication protocol is required by the architecture. Most part of the energy supply is used for agents motion, while the communication and the security protocols are less energy demanding.

In the proposed implementation, the time required for key authentication is $6.5ms$ with SkipJack, and $14ms$ with RC5. Moreover, confidentiality and integrity cost about 7 bytes per packet: 2 bytes are used to identify the key and to construct the Initialization Vector for encrypting/decrypting, whereas the remaining 5 bytes are employed as Cyclic Redundancy Check to grant integrity in the wireless communication.

The proposed localization service provides with resolution of $0.23cm$ and $0.03rad$ in an environment of $290cm \times 133cm$ using 2 cameras. The truncation error during the transmission process is at most of $0.5cm$ for length data and $0.025rad$ for angle. The average errors measured during experiments is around $1cm$ and $0.06rad$ for length and angle respectively.

Partial overlapping of reserved disks has occured during experiments for at most $4.4cm$ due to non exact integration of motion and noisy data communicated through the network. Indeed, as reported in section 2.1, the GRP policy ensures the safety of the system only theoretically. In the real framework, the system safety can be recovered enlarging the reserved disk dimension according to estimated errors.

5. CONCLUSIONS AND FUTURE WORK

A scalable platform for decentralized traffic management of multi-agent system has been proposed. The platform guarantees safe coordination of agents that move in a shared environment to accomplish a variety of application tasks. Furthermore, the platform guarantees security of communications among agents with respect to an external adversary. A prototypical implementation of the architecture is described, and some experimental results have been reported. Future work will be devoted to several remaining open problems, including further decentralization of the check-in and check-out and security procedures, intrusion detection, and non-collaborative collision avoidance protocols.

6. ACKNOWLEDGMENT

Authors wish to thank A. Convalle, L. Decaria, and I. S. La Porta for having contributed in the implementation of the platform prototype. The work has been done with partial support from EC project RUNES (Contract IST-2004-004536), and EC Network of Excellence HYCON (Contract IST-2004-511368).

7. ADDITIONAL AUTHORS

Additional authors: L. Pallottino, R. Schiavi, G. Dini, A. Bicchi (Interdepartmental Research Center “E. Piaggio” and Dipartimento di Ingegneria dell’Informazione, Faculty of Engineering, University of Pisa).

8. REFERENCES

[1] B. Sinopoli, C. Sharp, L. S. S. Schaffert, and S. S. Sastry, “Distributed Control Applications Within Sensor Networks,” *Proceedings of the IEEE*, vol. 91, no. 8, pp. 1235–1246, August 2003.

[2] S. Graham and P. Kumar, *Proceedings of PWC 2003: Personal Wireless Communication*, ser. Lecture Notes in Computer Science. Heidelberg: Springer-Verlag, 2003, vol. 2775, ch. Convergence of Control, Communication, and Computation, pp. 458–475.

[3] G. Baliga and P. Kumar, “Middleware for control over networks,” in *Proceedings of the 44th IEEE CDC’05*, December 2005.

[4] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, “Wireless sensor networks: a survey,” *Computer Networks*, vol. 38, no. 4, pp. 293–422, March 2002.

[5] Y. Mostofi, T. H. Chung, R. M. Murray, and J. W. Burdick, “Communication and sensing trade-offs in decentralized mobile sensor networks: A cross-layer design approach,” in *International Conference on Information Processing in Sensor Networks*, 2005.

[6] L. Pallottino, V. Scordio, E. Frazzoli, and A. Bicchi, “Probabilistic verification of a decentralized policy for conflict resolution in multi-agent systems,” in *ICRA06*, 2006.

[7] G. Dini and I. M. Savino, “An efficient key revocation protocol for wireless sensor networks,” in *Proceedings of IEEE WOWMOM’06*, Niagara-Falls, Buffalo-NY, 26–29 June 2006.

[8] V. B. R. Solis and P. R. Kumar, “A new distributed time synchronization protocol for multihop wireless networks,” in *Proceedings of the 44th IEEE CDC’05*, December 2005.