Platform-Based Design of Wireless Sensor Networks for Industrial Applications

†*Alvise Bonivento,* ‡*Luca P. Carloni, and* †*Alberto Sangiovanni-Vincentelli*
†*University of California at Berkeley,* ‡*Columbia University in the City of New York*
e-mail: †alvise,alberto@eecs.berkeley.edu ‡luca@cs.columbia.edu

***Abstract*— We present a methodology, an environment and supporting tools to map an application on a wireless sensor network (WSN). While the method is quite general, we use extensively an example in the domain of industrial control as it is one of the most promising application of WSN and yet it is largely untouched by it.**

Our design flow starts from a high level description of the control algorithm and a set of candidate hardware platforms and automatically derives an implementation that satisfies system requirements while optimizing for power consumption. To manage the heterogeneity and complexity inherent in this rather complete design flow, we identify three abstraction layers and introduce the tools to transition between different layers and obtain the final solution.

We present a case study of a control application for manufacturing plants that shows how the methodology covers all the aspects of the design process, from conceptual description to implementation.

I. Introduction

The application of WSN technology [1] to the design of field-area networks for industrial communication and control systems has the potential to provide major benefits in terms of flexible installation and maintenance of field devices, support for monitoring the operations of mobile robots, and reduction in costs and problems due to wire cabling [2], [3]

Figure 1 illustrates an example of *manufacturing cell*, i.e. a stage of an automation line in an industrial plant. The physical dimensions of this cell range between 10 and 20 meters on each side. In this area six robots cooperate to manipulate and transform the same production piece under the supervision of a process loop controller (PLC), which is placed right outside the cell. In this example the robots are equipped with drilling tips to work on metal surfaces. Careful monitoring of the state of the drilling tips is a critical task because they do wear out and they need to be replaced before damaging both the piece under construction and the robot itself. A typical way of monitoring the state of the tips is to observe the vibration pattern of the robot. Different vibration sensors are placed on the robot, and if the average vibration intensity of a robot goes above a given threshold, all the robots of the cell must be stopped so that a human operator (or another machine) can safely perform the required maintenance. The monitoring task is performed by the software running on the processor in the PLC unit, which must periodically query each cluster for information on the vibration intensity of each robot. We will consider this case study as a running example to better explain the proposed methodology.

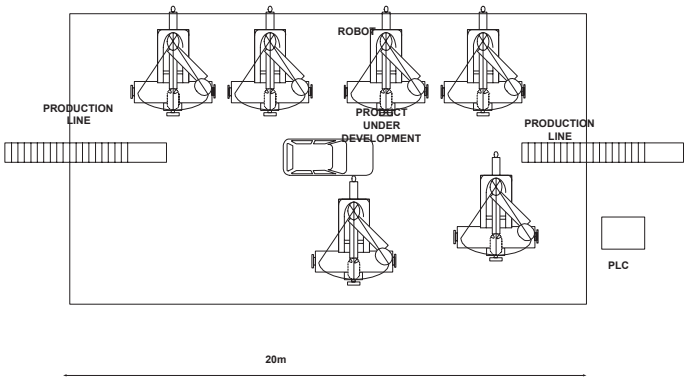

Fig. 1. Manufacturing Cell

The software for these applications is usually written by process or mechanical engineers that are expert in process control technology, but know little of the communication and sensing infrastructure that has to be deployed to support these algorithms. On the other side, the communication infrastructure is designed by communication engineers that know little about process control technology. Moreover, the adoption of wireless technology further complicates the design of these networks. Being able to satisfy high requirements on communication performance over an extremely unreliable communication channel is a difficult task. Consequently, the gap between the control algorithm designers and the network designers will inevitably increase and this phenomenon might delay the adoption of wireless sensor networks technology within manufacturing plants.

To bridge this gap and derive a correct and efficient implementation, we propose a design methodology that:

1) Allows the control algorithm designer to specify the application using a clear interface that abstracts the drudgeries of the network implementation.
2) Starting from the application description, it derives a set of constraints on the end-to-end (E2E) latency and packet error rate that the network has to satisfy.
3) Using the E2E requirements and an abstraction of the hardware platform, derives a solution for MAC and Routing that satisfies requirements and optimizes for energy consumption
4) Maps the communication protocol into the hardware nodes and the PLC.

Following the Platform Based Design (PBD) methodol-

ogy [5], [6], our system level approach is characterized by a top-down phase, where application requirements are refined in E2E network requirements, a bottom up phase, where hardware performance are abstracted, and a meet in the middle phase where the requirements and performance are used to solve a constrained optimization problem whose solution determines the policies and the parameters of the MAC and Routing layer.

The paper is organized as follows: In Section II, we offer a quick overview of the layers of abstraction we use to support our design methodology, in Section III we introduce a tool to capture application requirements, and in Section IV a methodology for protocol synthesis. In Section V we discuss how to map the solution to the given hardware platforms, and in Sections VII and VI we present our future plans and the related work. For a more detailed overview of our methodology, we refer the interested readers to [16].

II. Platform-Based Design for WSN: Abstraction Layers

Following Figure 2, at the application level we introduce the highest layer of abstraction in our methodology, the Sensor Network Service Platform (SNSP) [4]. Similar to the role played by the Socket in Internet applications, the SNSP offers an application interface that is able to support the possible services that can be used in a WSN independently of the network implementation.

To perform its functionality, a controller (algorithm) has to be able to read and modify the state of the environment. In a WSN, controllers do so by relying on communication and coordination among a set of distinct elements that are distributed in the environment in order to complete three different types of functions: sensing, control and actuation. The role of the SNSP is to provide a logical abstraction for these communication and coordination functions. The SNSP offers a *query service (QS)* used by controllers to get information from other components, a *command service (CS)* used by controllers to set the state of other components, a *timing/synchronization service (TSS)* used by components to agree on a common time, a *location service (LS)* used by components to learn their location, a *concept repository service (CRS)* which maintains a map of the capabilities of the deployed system and it is used by all the components to maintain a common consistent definition of the concepts that they agreed upon during the network operation. The CSR is quite novel in the WSN community, but is deemed essential if a true ad-hoc realization of the network is to be obtained. The repository includes definitions of relevant global concepts such as the attributes that can be queried (e.g. temperature, pressure), or the regions that define the scope of the names used for addressing. It further allows collecting information about the capabilities of the system (i.e. which services it provides and at which quality and cost) and provides the application with a sufficiently accurate description. The repository is dynamically updated during the network operations.

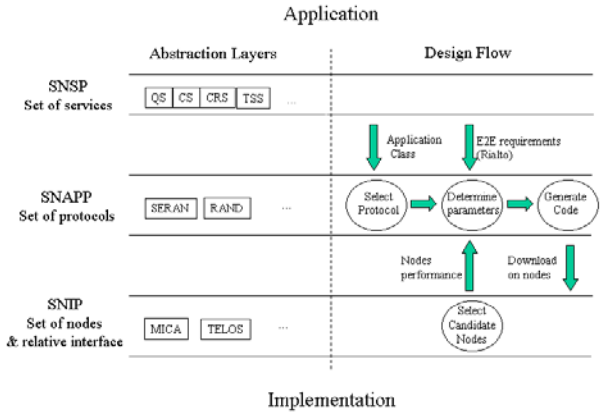

Fig. 2. Layers of abstraction and design flow

A. The Sensor Network Implementation Platform

The Sensor Network Implementation Platform (SNIP) is a network of interconnected physical nodes that implement the logical functions of the application and the SNSP. A physical node is a collection of physical resources such as clocks and energy sources, processing units, memory, communication, I/O devices, sensor and actuator devices. The main physical parameters of a node are the list of sensors and actuators attached to node, the memory available for the application, clock frequency range, the clock accuracy and stability, the level of available energy, the cost of computation (energy), the cost of communication (energy), the transmission rate (range).

B. The Sensor Network Ad-hoc Protocol Platform

To choose the architecture of the SNIP and to map the functional specification of the system onto it are critical steps in sensor network design. To facilitate the process we created an intermediate level of abstraction called Sensor Network Ad-hoc Protocol Platform (SNAPP).

The SNAPP is composed by a library of MAC and routing protocols that offer to the SNSP guarantees on latency, error rate, sensing requirements. Different protocols have been developed for different application classes. For example SERAN [12] was developed for clustered topologies, while the randomized approach of [10] (called RAND in Figure 2) was developed for uniformly distributed topologies. The appropriate protocol is selected according to the application class.

These protocols are “parametrized protocols”, meaning that their structure is specified, but their working point is determined by a set of parameters. The value of these parameters is obtained as the solution of a constrained optimization problem, where the constraints are derived from the latency, error rate, sensing requirements of the application while the cost function is the energy consumption. The energy consumption is estimated based on an abstraction of the physical properties of the candidate hardware platform. The synthesis of these parameters represents the meet-in-the-middle phase of the PBD methodology when applied to the WSN domain.

III. Mapping SNSP to SNAPP for Industrial Control: Rialto

Rialto [11] is a tool that targets WSN industrial control applications and helps the transition from the SNSP to the SNAPP. It supports those applications in which, as in our case study, the end user wants to deploy a dedicated network to support a periodic control application.

Rialto supports only the subset of the services of the SNSP that are relevant for the chosen industrial domain; specifically: the query service, the command service and the concept repository service. Rialto allows the end user to specify a loose notion of the system topology in the concept repository service and to describe the control algorithm in terms of logical components, queries and commands.

Following the approach of [4], in the proposed framework, designers describe the application in a Rialto Model in terms of Virtual Controllers, Virtual Sensors, and Virtual Actuators.

A *Virtual Controller (VC)* contains the description of the control algorithm of the application. If the application has more than one independent control algorithm, multiple Virtual Controllers have to be specified. In our case study, we have a single VC with an algorithm that needs information on robot vibrations to take its decisions.

A *Virtual Sensor (VS)* represents a sensing area. This abstraction is useful because designers know which are the areas that need to be sensed, but they generally don't know how many sensors must be placed to cover that area and how they have to placed. For example, in our application, there are six virtual sensors (one for each robot). There is not necessarily a one-to-one relationship between virtual sensors and physical sensors. The number and the type of physical sensors that will be used to implement a virtual sensor is an implementation choice. In our application, a virtual sensor will most likely be implemented with a set of multiple sensors.

A *Virtual Actuator (VA)* represents an actuation capability. Similarly to the VS, the user describes the position of the VA, but the number and type of physical actuators that will be selected to implement its functionality is an implementation choice. In our case, there are six Virtual Actuators, one for each robot.

After the virtual components are declared, the interaction among them is described using queries and commands. Rialto allows connections only between Virtual Controllers and Virtual Sensors and between Virtual Controllers and Virtual Actuators. Consequently, no connection is allowed between two Virtual Sensors, two Virtual Actuators, or a Virtual Sensor and a Virtual Actuator. This restriction makes sense because we are describing an application using logical components. Connections between two sensors (commonly referred to as multi-hopping) are an implementation option, and as such they don't belong to the application description level of abstraction. Similarly, a connection between two physical controllers is an implementation option, but at the application description level connections between two Virtual Controllers are not allowed. Hence, if a Virtual Controller needs a particular set of data, it has to send a query directly to a Virtual Sensor.

A. Scenarios exploration

After the application is described, the description is translated into an internal representation called RialtoNet.

Since we want to generate a set of requirements to design a sensing and communication infrastructure that is able to satisfy every possible request of the controlling algorithms, we need to evaluate all the various combinations of requests that Virtual Controllers could generate. The RialtoNet is created precisely for an explicit exploration of all the possible combinations of queries and commands in a given application. Since the number in a control routine has is typically limited, the number of possible combinations is often very manageable.

By analyzing the software code of every VC, we detect all the possible combinations of conditional statements involving a request, and for each of them we create a new independent component, called VC Branch (VCB). Each Virtual Sensor is modified into a Virtual Sensor Skeleton and each Virtual Actuator into a Virtual Actuator Skeleton (VAS) that are obtained from the original code modifying the read and write semantic. A RialtoNet is generated by substituting each VC with its relative VCBs, each VS with its relative VSS, and each VA with its relative VAS.

B. Requirement generation

During the execution of the RialtoNet, we generate a set of constraints on latency, bit error rate, and sensing requirements that are the starting point for the design of the physical network. Since the distinct VC Branches are executed as independent components and each of them represents a possible combination of queries and commands, the requirements on sensing and communication infrastructure guarantee that all the possible combinations can be supported.

Consequently, the end user is able to describe the application with no knowledge of the network architecture, while Rialto provides a bridge to the implementation platform. Starting from these requirements, a communication protocol can be designed with the guarantee that, if these constraints are satisfied, the network architecture will be appropriate to support the correct functionality of the application.

IV. Protocol Synthesis

An important and usually non trivial step in the top-down refinement process associated with the move from one layer of abstraction to the next consists in analyzing application requirements on the end-to-end (E2E) delay and translating them into a hop-to-hop (H2H) delay which is simpler to handle and of direct impact to the protocol design. The ability of performing this refinement is subject to the capability of characterizing the interaction among the different layers of the protocol solution using a mathematical framework. The mathematical framework allows us to capture the requirements of the design functionality and performance as a constrained optimization problem. The solution to this problem provides the parameters to derive the final protocol implementation. Once the trade-off equations are derived and solved as an

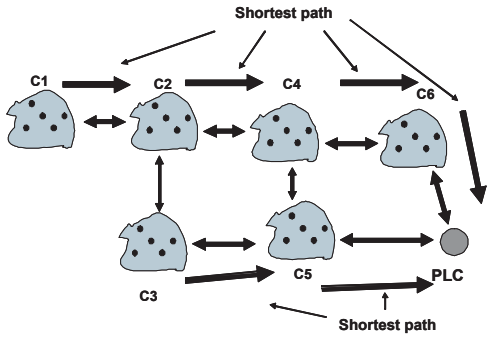

Fig. 3. Connectivity graph

optimization problem, all the protocol parameters are automatically synthesized. The formalism and the capability of offering end-to-end guarantees instead of local guarantees is what distinguish our approach from the previous protocol design for WSNs.

The use of parameterized protocols allows us to effectively restrict the large design space to a few parameters. In addition, since the protocols are developed with a specific mathematical model in mind, we can easily gouge the effects of changing these parameters on the overall network performance. This predictive ability prevents the need for extensive simulation and allows for the ease of comparison with other protocols.

A. A SEmi-RANdom Protocol for Clustered Topologies: SERAN

For naturally clustered environments, as in our example, we developed a semi-random protocol stack called SERAN, which covers two layers of a classical protocol stack: routing and MAC. In this section, we give a brief overview of SERAN, see [12] for a more detailed description and performance analysis.

1) Routing Algorithm: Routing over an unpredictable environment is notoriously a hard task. High node density makes the problem easier to solve. The idea is to have a set of nodes within transmission range that could be candidate receivers; at least one of them will offer a good link anytime a transmission is needed.

The routing solution of SERAN is based on a semi-random scheme that reduces the overhead of purely random approaches. In SERAN, the sender has knowledge of the region to which the packet will be forwarded, but the actual choice of forwarding node is made at random. This approach is motivated by the fact that the clustered topology of the sensor network for robots monitoring in a manufacturing cell is known a priori.

A connective graph like the one reported in Figure 3 can be derived from the given cluster topology [12]. In the graph, an arc between two clusters means that the nodes of the two clusters are within transmission range. We further assume that the nodes share the same communication channel. Then, the first step of the SERAN routing algorithm consists of calculating the shortest path from every cluster to the PLC and generating the minimum spanning tree as shown in Figure 3.

Assuming that a particular node in Cluster 1 must forward a packet to the PLC, the proposed routing algorithm works as follows:

- first, the node that has the packet selects randomly a node in Cluster 2 and forwards the packets to it;
- then, the chosen node determines its next hop by choosing a node randomly in Cluster 4, and so on.

In other words, packets are forwarded to a randomly chosen node within the next-hop cluster in the minimum spanning tree leading to the PLC.

2) Hybrid MAC: The first priority in the design of our MAC is ensuring robustness against topology changes. Since nodes failure is a common phenomenon for WSN, we design a MAC that is able to support the addition of new nodes for preserving the high level of density required to ensure robustness. This flexibility is usually obtained by using random based access schemes. In the WSN domain, an interesting example of this idea is presented in BMAC [14]. High density unfortunately introduces a large number of collisions. This drawback becomes crucial in our case because we have only one channel that can be used for communication. To reduce collisions, a deterministic MAC is used. A well-known deterministic approach is SMAC [13], where the network is organized according to a clustered TDMA scheme. Our MAC solution is based on a two-level semi-random communication scheme. This offers robustness to topology changes and node failures that is typical of a random based MAC protocol and robustness to collision that is typical of a deterministic MAC protocol

a) High Level MAC: The higher level regulates channel access among clusters. A weighted TDMA scheme is used such that at any point in time, only one cluster is transmitting and only one cluster is receiving. During a TDMA cycle, each cluster is allowed to transmit for a number of TDMA-slots that is proportional to the amount of traffic it has to forward. The introduction of this high level TDMA structure has the goal of limiting interference between nodes transmitting from different clusters. The time granularity of this level is the TDMA-slot. After the two clusters terminated their transmitting TDMA-slot, another TDMA-slot (called the *actuation slot*) is reserved for the PLC. During this slot, the PLC sends a message to the actuator of each robot to continue operating or to switch the robot off.

b) Low Level MAC: The lower level regulates the communication between the nodes of the transmitting cluster and the nodes of the receiving cluster within a single TDMA-slot. It has to support the semi-random routing protocol presented in IV-A.1, and it has to offer flexibility for the introduction of new nodes. This flexibility is obtained by having the transmitting nodes access the channel in a p-persistent CSMA fashion [15]. The random selection of the receiving node is obtained by multi-casting the packet over all the nodes of the receiving cluster, and by having the receiving nodes

implement a random acknowledgment contention scheme to prevent duplication of the packets.

In this approach, nodes need to be aware only of the next-hop cluster connectivity and do not need a neighbor list of next hop nodes. We believe this is an important benefit because cluster based connectivity is very stable, while neighbor lists of nodes are usually time-varying (nodes may run out of power and other nodes may be added) and their management requires significant overhead.

c) Energy Minimization: In most of the proposed MAC algorithms for WSNs, nodes are turned off to save energy whenever their presence is not essential for the network to be operational. Following this approach, our duty-cycling algorithm leverages the MAC properties and does not require extra communication among nodes. During an entire TDMA cycle, a node has to be awake only when it is in its listening TDMA-slot or when it has a packet to send and it is in its transmitting TDMA-slot. For the remainder of the TDMA cycle, the node radio can be turned off.

3) Protocol parameter synthesis: The working point of the communication protocol is determined by tuning a set of parameters such as the TDMA schedule, the duration of the TDMA-slot, and the channel access probability p.

The number of transmitting TDMA-slots assigned to each cluster is proportional to the amount of traffic that the cluster has to forward. Consequently, clusters closer to the PLC have more transmitting slots since they have to forward the packets that they generate plus the packets coming from upstream clusters. For each path, the first cluster to transmit is the closest to the PLC (Cluster 4). Then Cluster 2 and Cluster 4 again. Then Cluster 1, 2 and 4, and similarly on the other path. This scheduling is based on the idea that evacuating the clusters closer to the PLC first, we minimize the storage requirement throughout the network.

In [12] we show that the energy consumption is monotonically decreasing with the duration of the single TDMA-slot. Consequently, using the TDMA structure and the cluster based routing, we are able to determine the maximum duration of the TDMA-slot so that the E2E requirements of the far most clusters are satisfied.

The random access parameter p needs to be set such that all the nodes in the cluster are able to forward their packets during a TDMA-slot. In [12], we model the packet transmission process as a Discrete Time Markov Chain and we show how to find the optimum p solving a convex optimization problem.

V. Mapping and Implementation

After creating the network infrastructure, the final step of the design flow consists in mapping the controlling algorithm onto the controller hardware platform, and mapping the communication protocol onto the wireless nodes.

The first step consists in mapping the controlling algorithm into the hardware platform of the PLC. This represents a classical embedded systems mapping problem (i.e. not specific of the WSN domain) and it can be performed with classical mapping tools. Consequently, we do not address it in this work

The second step is to map the communication protocol on the physical nodes. Since the communication protocols of the SNAPP are already described in a distributed fashion, the parametrized code for each node can be easily developed using the software interface of the nodes. Most often, this interface is given by TinyOS and the parametrized code can be written using NesC [7].

The actual setting of the parameters of the nodes to determine their working point is obtained using an initialization algorithm that kicks in when the nodes and the PLC are switched on. This algorithm allows for self-adaptation of the network to the optimal working conditions. Furthermore, to preserve the correct behavior of the communication infrastructure, network management algorithms are automatically run on the network on a periodical basis.

VI. Related work

Since we propose a design methodology that supports all the phases of the design of WSN, from application to implementation, there is quite a large body of related work on system level methods, tools and protocols. For sake of brevity, we outline only some recent works while we refer to [16] for a more detailed analysis.

A system level approach to the design of WSNs was recently proposed by Polastre *et al.* [17]. A platform called SP is proposed between the link and the network layer. The SP should provide the adequate modularity for the nodes to support different MAC and Routing layers. The philosophy is similar to the Internet "everything over IP", where in this case it would be "everything over SP". Although this is a very interesting architecture for best effort networks, we believe it is not appropriate for control applications where E2E guarantees are required. Our top-down approach and synthesis method are customized for control applications.

An attempt of raising the level of abstraction was presented in [8], where a classification for node communication mechanisms was introduced to allow for a higher level description of the network algorithms. In [9], the proposed methodology is based on a bottom-up part for the description of network algorithms, a top-down part to describe the application, and a mapping process to deploy software onto the nodes. The overall method fits with the PBD paradigm advocated in this paper but it does use different layers of abstraction. Our approach emphasizes the control based nature of WSN applications and offers a rigorous semantics and set of primitives to interpret timing issues at a very high level, hence providing a well-defined level of abstraction for the application designer.

VII. Future directions

We are currently working in different directions to improve the capabilities of our design flow in terms of supported classes of application and improved performance of the network infrastructure. In this section, we offer a quick overview of these efforts.

A. Non-Periodic control applications

Consider the case in which a mechanical engineer does not have the freedom of designing from scratch a WSN but has to use what was already deployed to implement different control applications that have to run for a limited amount of time. The problem is to reconfigure the existing network and offer E2E guarantees under certain restrictions given by the physical capabilities of the nodes already deployed. Although the levels of abstraction that we already discussed stay the same, the mapping of the application into the network infrastructure changes substantially. Specifically, this mapping will have to be performed at run time. For this to be feasible, the network infrastructure has to be flexible, easily reprogrammable, and capable of determining at run time the type of E2E guarantees it can offer given the current status of activity in the network and a projection of the utilization of the new queries. Modeling these aspects in the SNAPP is a focus of our future work.

To solve the dynamic mapping problem, we are currently developing a real time network scheduler capable of recognizing different priority classes for the incoming queries/commands, of analyzing the satisfiability of these queries/commands depending on the network status, and of deciding whether to dispatch or to stall the query/command according to a given policy.

B. Distributed Aggregation Algorithms

The case study presented here is based on a *centralized* approach where the PLC periodically receives a packet from every node in each cluster with the updated data on the vibration intensity of the corresponding robot. The efficiency of the centralized implementation, however, depends on cluster topology and the number of nodes per cluster. In fact, due to the multi-hop communication scheme used, the nodes that are closer to the PLC are required to support also the traffic due to packets coming from distant nodes. Consequently, they dissipate more energy, and statistically end up having a shorter lifetime.

Since WSN nodes include some computational capabilities, it is often preferable that the nodes on the same cluster locally compute the average vibration and select one among them to report the data to the PLC along the multi-hop chain. However, since node malfunctions and failures are not rare events in a WSN, fault-tolerant protocols are essential which guarantee that multiple, if not all, nodes can take over the responsibility of computing and propagating the result to the PLC. To this end, we are developing a library of fault tolerant distributed aggregation algorithms that enable nodes of the same cluster to share data locally, eventually computing aggregate functions, during their transmitting TDMA-slot so that only one packet has to be forwarded to the downstream cluster at the end of the slot. Notice however, that both the structure of the TDMA/CSMA communication protocol and the cluster-to-cluster routing algorithm that are used in the centralized solution can be seamlessly used in the distributed solution. This is an important point since it enables the separation of the mechanism for computing the aggregate function from the design of the inter-cluster communication infrastructure, thereby minimizing the extra design complexity due to the introduction of the distributed aggregation capability.

REFERENCES

[1] D. Culler, D. Estrin, and M. Srivastava, "Overview of Sensor Networks", IEEE Computer, vol. 37, num.8, pg 41-49, Aug. 2004.

[2] R. Zurawski,"Introduction to Special Issue on Industrial Communication Systems", Proc. of the IEEE, vol.9, num.6, pg. 1067-1072, Jun.2005.

[3] A. Willig, K. Matheus, and A. Wolisz, "Wireless Technology in Industrial Networks", Proc. of the IEEE, vol. 9, num.6, pg.1130-1151, Jun.2005.

[4] M. Sgroi, Adam Wolisz, Alberto Sangiovanni-Vincentelli and Jan M. Rabaey, "A Service-Based Universal Application Interface for Ad-hoc Wireless Sensor Networks", whitepaper, U.C.Berkeley 2004.

[5] A. Sangiovanni-Vincentelli, A. Ferrari, "System Design - Traditional Concepts and New Paradigms", Proceedings of ICCD 99, Austin, October, 1999, pp.2-12.

[6] A. L. Sangiovanni-Vincentelli, L. Carloni, F. De Bernardinis and M. Sgroi, "Benefits and Challenges for Platform-Based Design", Proceedings of the Design Automation Conference (DAC'04), San Diego, CA, USA, June 2004.

[7] D. Gay, P. Levis, R. von Behren, M. Welsh, E. Brewer, and D. Culler, "The nesC Language: A Holistic Approach to Networked Embedded Systems", Proceedings of Programming Language Design and Implementation (PLDI) 2003, June 2003.

[8] Y. Yu, B. Hong, V.K. Prasanna, "Communication Models for Algorithm Design in Wireless Sensor Networks", IPDPS '05.

[9] A. Bakshi, V.K. Prasanna, "Algorithm Design and Synthesis for Wireless Sensor Networks", ICPP '04.

[10] A. Bonivento, C. Fischione, A. Sangiovanni-Vincentelli "Randomized Protocol Stack for Ubiquitous Networks in Indoor Environment", CCNC 2006.

[11] A. Bonivento, L.P. Carloni, A. Sangiovanni-Vincentelli, "Rialto: a Bridge between Description and Implementation of Control Algorithms for Wireless Sensor Networks", Proc. of EMSOFT 2005, Jersey City, NJ, USA, Sept. 2005.

[12] A. Bonivento, C. Fischione, A. Sangiovanni-Vincentelli, F. Graziosi, F. Santucci, "SERAN: A Semi Random Protocol Solution for Clustered Wireless Sensor Networks", To appear in Proc. of MASS 2005, Washington D.C., USA, Nov. 2005.

[13] Wei Ye, John Heidemann and Deborah Estrin, "Medium Access Control with Coordinated Adaptive Sleeping for Wireless Sensor Networks", IEEE/ACM Transactions on Networking, Vol. 12, No. 3, pp. 493-506, June 2004.

[14] J. Polastre, J. Hill, and D. Culler, "Versatile Low Power Media Access for Wireless Sensor Networks", Sensys 2004.

[15] T.S. Rappaport, "Wireless Communications", Prentice Hall, Upper Saddle River NJ, 1996.

[16] A.Bonivento, L.P. Carloni, A. Sangiovanni-Vinctelli, "Platform-Based Design for Wireless Sensor Networks", to appear in Mobile Networks and Applications, The Journal of Special Issues on Mobility of Systems, Users, Data and Computing, 2006.

[17] J. Polastre et al., "A Unified Link Anstraction for Wireless Sensor Networks", Sensys 2005.