Technical Report

Elements of Scalable Data Processing

Björn Andersson
Paulo Gandra de Sousa
Filipe Pacheco
Panayiotis Andreou
Pedro Jose Marron
Umer Iqbal

HURRAY-TR-100510
Version:
Date: 05-27-2010

Elements of Scalable Data Processing

Björn Andersson, Paulo Gandra de Sousa, Filipe Pacheco, Panayiotis Andreou, Pedro Jose Marron, Umer Iqbal, Vinny Reynolds

IPP-HURRAY!

Polytechnic Institute of Porto (ISEP-IPP)

Rua Dr. António Bernardino de Almeida, 431

4200-072 Porto

Portugal

Tel.: +351.22.8340509, Fax: +351.22.8340509

E-mail:

http://www.hurray.isep.ipp.pt

Abstract

Cooperating objects (COs) is a recently coined termused to signify the convergence of classical embedded computersystems, wireless sensor networks and robotics and control. Wepresent essential elements of a reference architecture for scalabledata processing for the CO paradigm.

Elements of Scalable Data Processing

Björn Andersson*, Paulo Gandra de Sousa*, Filipe Pacheco*, Panayiotis Andreou†,
Pedro José Marrón‡, Umer Iqbal‡ and Vinny Reynolds§
*CISTER Research unit, Polytechnic Institute of Porto, bandersson@dei.isep.ipp.pt
*CISTER Research unit, Polytechnic Institute of Porto, pag@isep.ipp.pt
*CISTER Research unit, Polytechnic Institute of Porto, ffp@isep.ipp.pt
†University of Cyprus, panic@cs.ucy.ac.cy
‡University of Duisburg-Essen, pjmarron@uni-due.de
‡University of Duisburg-Essen, umer.iqbal@uni-due.de
§The National University of Ireland, Galway, vinny.reynolds@deri.org

***Abstract*—Cooperating objects (COs) is a recently coined term used to signify the convergence of classical embedded computer systems, wireless sensor networks and robotics and control. We present essential elements of a reference architecture for scalable data processing for the CO paradigm.**

I. Introduction

As embedded computer systems increase in importance and become networked, new concepts emerge to emphasize different aspects of them. The concepts of pervasive and ubiquitous computing emphasize seamless deployment in everyday things and environments. The concept of wireless sensor networks (WSNs) emphasizes sensing and networking aspects, whilst the concept of cyber-physical systems (CPSs) emphasizes the interaction between information processing ("Cyber") and the physical world. The concept of cooperating objects (COs) [1] has recently gained acceptance in signifying the convergence of classical embedded computer systems, wireless sensor networks and robotics and control, whilst emphasizing ad-hoc, opportunistic cooperation between (often) autonomous entities. Furthermore, it is currently seen by the European Commission as the main research direction in networked embedded computer systems.

Formally speaking, a cooperating object (CO) [2] is a single entity or a collection of entities consisting of (i) sensors, (ii) actuators, (iii) information processors or (iv) cooperating objects. Since the definition of a CO is recursive (a CO may be a collection of COs), this definition makes it possible to form arbitrarily complex structures. The concept of a CO is quite new however and therefore, no general theory for COs is known and no model or architecture for data processing concerning COs is known either.

It is possible to co-locate a single sensor, a single information processor and a single actuator to a single computer node and let the information processor perform processing based on readings from the single sensor and let the actuator enact commands computed by the single information processor. Harnessing the power of cooperating objects requires however that information processors perform processing of sensor readings originating from different computer nodes. Sometimes this is strictly necessary. For example, most methods for estimating the most likely geographical location of a physical object depend on sensor readings originating at different computer nodes. In other cases, an information processor can obtain a much better image of the physical world, using better resolution and greater coverage, if the information processor takes as input sensor readings originating from different computer nodes. Clearly, many COs need to perform data processing based on sensor readings originating from different computer nodes.

Since COs often interact with their physical environment by giving commands to actuators (these commands are computed based on sensor readings), it is essential that the *response time* of data processing from sensing to actuation is small. A straightforward approach to perform data processing in COs is that each actuator has an associated information processor and all sensors that provide sensor readings that are needed for the data processing on this information processor send the sensor readings to this information processor. This approach is applicable in COs composed of a relatively small number of sensors. But the trend [3], [4], [5] in COs is towards a larger number of sensors because (as already stated) this provides a better image of the physical world. Unfortunately, performing data processing with the straightforward approach leads to long response times because of the large amounts of packets that must be sent from each sensor. This is particularly problematic when all sensor nodes are in a single broadcast domain because then at most one sensor can transmit at a time. Clearly, better methods for data processing are needed; data processing based on sensor readings originating from different computer nodes should be performed in a scalable manner; that is, the response time should grow slowly (or not at all) with the number of sensor nodes/readings.

Many COs are comprised of sensor nodes belonging to different organizations with no superior/subordinate relationship between these organizations. Yet, the functioning of the CO requires that data processing is performed based on these sensor readings. And this brings an additional challenge to data processing in COs. Therefore, a core problem in the design of COs is to perform data processing based on sensor readings originating from different computer nodes potentially in different organizations and do so in a scalable manner; we refer to this as the *Scalable Data Processing* problem.

In this paper, we present essential elements of a reference architecture for the Scalable Data Processing problem[1]. We believe that developing this reference architecture is significant because (i) COs with a large number of sensor nodes and which operates across a large number of organizations will be built in the future and (ii) there is (as far as we know) no reference architecture available for the Scalable Data Processing problem. Since COs and CPSs have many commonalities, we also expect our future work on this reference architecture to be useful also for CPSs.

The remainder of this document is organized as follows. Section II presents a usage scenario to motivate the need for query processing. Section III presents a system overview. Section IV presents the architecture. Section V presents related work. Finally, Section VI presents ongoing work.

II. A USAGE SCENARIO

In order to understand the scalable data processing problem, let us consider a usage scenario.

Europe food safety legislation [6] specifies that when certain foods are being transported, they must be kept within certain temperature thresholds for the safe preservation of the food's quality. For example, it is necessary to be able to verify that frozen foods must be kept frozen for the entire duration of the journey. From a European health and safety perspective, food that defrosts and is re-frozen is considered unsafe and this event must be detected and reported. In order to accomplish this, many refrigerated vehicles, "reefers", now embed wireless sensor networks in vehicles and in load containers. Coupled with on board telematics units, this set up allows the monitoring of these temperature thresholds during the entire food transportation cycle.

A. Description

Typically, a fleet management company provides clients (food companies in this scenario) with a fleet administration tool which allows clients to make a range of queries against vehicles transporting their load. Each vehicle, shown in Figure 1, may be connected to the fleet administration tool via a 3G internet connection, and each vehicle administers its own wireless sensor network, which is monitoring the vehicle's load. For larger companies, they may have several hundreds or even thousands (in the case of supermarket chains) WSN-enabled vehicles transporting loads across Europe at any one time. A simple geospatial query requesting the location of all vehicles at a single point in time, would result in a very large and costly query being disseminated to potentially all of these vehicles. In this scenario, minimizing the financial cost of these queries is very important to the fleet administrators, and doing this in the presence of large scale systems is a challenge.

[1]Due to space limitations, we do not present the entire architecture — only the essential elements of it.

Fig. 1. A typical query in the fleet management scenario

B. Scenario Characteristics

This scenario has several characteristics that are noteworthy with the scalable data processing problem. These include:

- Dynamic scale. The total number of sensors (gateways + wireless sensors) within the entire query system includes the amount of wireless sensor nodes on each individual vehicle in addition to the mobile gateway on each vehicle. Dynamic aspects of typical queries such as geospatial queries, can result in a very large number of nodes having to be queried.
- Mobility. Gateway nodes (on the vehicle) are mobile as are the wireless sensor networks themselves, but not with respect to the gateway node. The gateway node is attached to a mobile telematics unit, (equivalent to a embedded mobile phone), which can both process one shot, continuous and event triggered queries. The mobile telematics unit is equipped with a GPS, so accurate location data is usually provided.
- Expensive communication costs. The cost of an individual communication between the client and a vehicle is small but not zero, as the communication is performed over a 3G internet connection. For example, continuous queries that push data constantly upwards are cost prohibitive and may only be executed once an hour, or even less frequently. Effective management of queries should help to depress the overall cost of the communication within the system, which is an important characteristic of this scenario.

C. Queries

A client sends his query from a non-self administered website, a portal. Two typical queries would be as follows:

1) Query Q is "Display the location of all of my vehicles". Without any prior information, an uninformed query planner would have to disseminate this one shot query to all telematics devices. They would respond with their location.
2) Query Q2 is a continuous query. "Raise an event when the temperature within a vehicle is greater than -2 degrees." In this query, periodic single shot queries are ex-

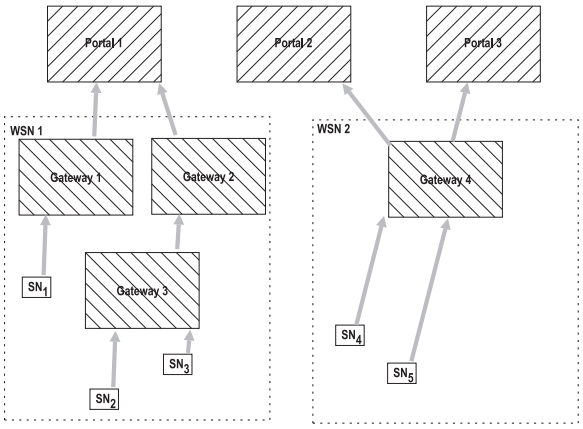

Fig. 2. An example of a CO that performs scalable data processing. There are three users (not shown). One user asks a query to Portal 1; another user asks a query to Portal 2 and a third user asks a query to Portal 3. The former query requires execution of sensor reading originating in WSN_1. The two latter queries require execution based on sensor readings in the same WSN, WSN_2. The gray arrows show the flow of data when executing the queries.

pensive and inefficient. Events that are triggered locally by the vehicle's onboard sensors push the data/event towards interested parties (the client).

III. System overview

In this section we will formalize our system model and the basic terminology which will be used in the subsequent sections.

Let $\{SN_1, SN_2, \ldots, SN_n\}$ denote a set of individual sensing devices with the ability to communicate in an adhoc manner wirelessly. These sensing devices have the ability to acquire physical attributes at discrete time instances t and then propagate them using a multihop communication mechanism to some gateway $Gateway_i$. A gateway has the purpose of retrieving critical data from individual sensor nodes and has a larger energy capacity. This setup is typically referred to as a Wireless Sensor Network (WSN) which is normally owned by a single organization. Usually, a WSN consists of a single gateway but there are often cases where more than one gateway exists to better support the operation of the WSN. For example, in Figure 2, WSN_1 consists of three gateways.

A user/application can specify a query Q directly to a WSN (i.e., through a gateway) or through a portal. In the former case, the user must first identify the proper gateway and then follow specific access routines and protocols to issue Q. On the other hand, portals offer a transparent way of accessing a WSN (or multiple WSNs) by providing a set of abstract interfaces that allow the user to query one or more WSNs easily in an identical manner. To accomplish this, portals employ: (i) WSN registries that allow applications to discover WSNs, and (ii) data transformation mechanisms that enable communication to (i.e., query) and from (i.e., data) the WSN.

In this work, queries are represented in a declarative SQL-like syntax. For instance, the following query declares that each sensing device should recursively collect the node identifier and the temperature from its children every 31 seconds and communicate the results to the gateway. `SELECT nodeid, temp FROM sensors EPOCH DURATION 31 seconds`

IV. Architecture

The reference architecture describes the main components that have to be provided to meet the requirements for scalable data processing across heterogeneous sensor networks. As an *in-node architecture*, the reference architecture omits the description of specific interfaces and implementations, but instead presents generic roles that should be met by software components on each node, regardless of whether the node is a sensor node, gateway or portal. Within these nodes, different software and hardware constraints are enforced requiring the reference architecture to be implemented to meet these constraints. Our scalable data processing architecture, shown in Figure 3, presents six main components. Due to the brevity of the paper, only those components that relate to the key elements of scalable data processing are described, and as such the descriptions of the API layer, and the system management component are omitted.

A. Query Manager

The Query manager is responsible for the execution of queries in our framework. Queries are posted to the query manager via the Query API. There exists three different groups of queries in our framework categorized according to the layer at which they are going to be executed. These are the (i) Portal, (ii) Gateway and (iii) Broadcast domain levels. Our framework will support different instantiations of the same components in order to cope with this diversity.

The Query Manager consists of the following components:

- **Parser:** Queries executed within a wireless sensor network can be classified using a variety of metrics, e.g. number of returned results, aggregation mechanisms used, execution frequency, etc. The Parser component is responsible for translating queries from a predefined format to a data structure. In order to do so, the Parser incorporates a grammar that checks the query syntax and looks for specific tokens that exist in the grammar. This process enables the parser to determine the query semantics and build the appropriate data structure that will be used by other components for the execution of the query. Our framework supports three query types: (i) Selection Queries, (ii) Storage/View Queries and (iii) Event based Queries.
- **Planner:** The Planner component handles the planning of the queries. Specifically, it considers all possible query plans for a query Q, calculates a "cost" for each query plan and then opts for the one with the lower cost. Typically, "cost" is calculated with regards to response time performance (I/O operations, memory usage, etc.). However, in WSNs, "cost" is evaluated with different metrics that take into account the peculiarities and limitations of WSNs (e.g., limited battery).
- **Query Optimizer:** The Query Optimizer sub-component considers all possible query plans and chooses the

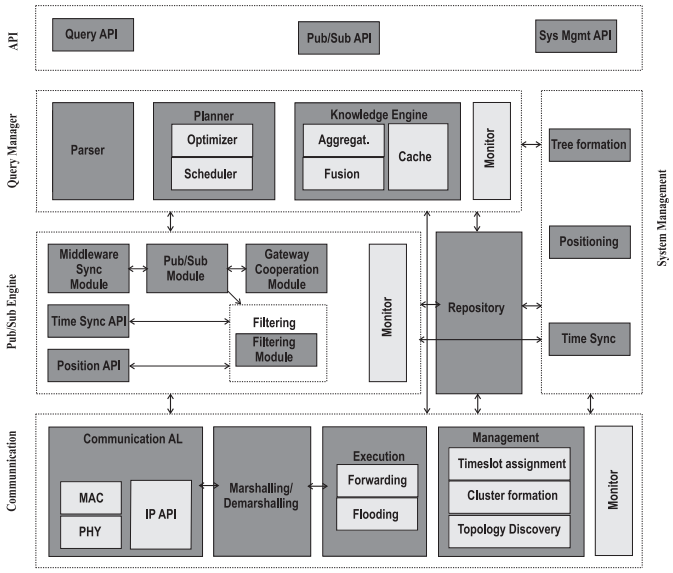

Fig. 3. The Scalable Data Processing reference architecture

most cost-efficient, whilst the scheduling sub-component schedules the execution of the query depending on the query type.

- **Knowledge Engine:** The knowledge engine is responsible for advanced topics in the query execution process like aggregation. Additionally, it utilizes a caching component that enables query instantiations to cache results locally in order to speed-up operations.

B. Publish/Subscribe

Unlike in traditional networks where the communicating entities are interested in the actual sender and receiver of data, data centric networks are less worried about this information and more concerned with the data transmitted between those entities. This means that in such networks the entities register for a particular data type in which they are interested in and not with particular nodes offering or looking for data. This is also true for the majority of wireless sensor network applications where the applications are interested in the data gathered by the sensor nodes and not in the physical sensor nodes themselves.

The Publish-Subscribe (Pub/Sub) paradigm is one such approach in which entities associate or register themselves to a type of data. It is an asynchronous messaging paradigm which provides a decoupling between the sender and receivers of data and provides one of the better abstractions for communication in mobile networks or applications. The Pub/Sub paradigm is based on the roles of publisher and subscriber, either or both of which can be assumed by a network entity. Entities which assumes the role of publishers publish data and entities subscribed to that type of data receive it. The pub/sub interaction can be either centralized or have some distributed mechanism. In the centralized approach, all publishers notify a central entity or a broker about the data which they would like to publish and all subscribers make their subscription to that central entity based on their requirements. The central entity on having a condition where a subscription request can be fulfilled by a publisher, forwards the data to the subscriber. In a distributed approach, every node can assume a role of a publisher or subscriber or both. In this case, the publish and subscription requests are sent over the network and are maintained by every node, which it can use knowledge of what types of data are available in the network and whether it is interested in it. Conversely, the type of data published locally is also known and a list of subscribers to that data within the network is also maintained.

Usually, publishing and subscription requests are additionally filtered on topics and/or content. In the former, the association is made on the topic e.g. if there is a temperature data available for the building then pass this information to the subscriber, whereas in the latter the approach, association is based on the contents i.e. if the temperature in the building is more than 25° C then pass this information to the subscriber.

In the proposed reference architecture, publish subscribe mechanism can be used for both or either of the actual sensor data and metadata regarding the sensor network. Examples of actual sensor data include temperature, humidity, light etc. whereas metadata can include network performance parameters such as number of subscriptions, position of sensor nodes, number of active nodes etc. The choice of using publish subscribe mechanism for a particular type of data depends upon the usage scenario. Depending upon usage scenario, it is also possible to use publish subscribe mechanism for regular sensor data at one part of the network and for transferring metadata for some other part of same network. An example could be a usage scenario mentioned in Section II, where publish subscribe mechanism can be used for transferring metadata information between different containers and also for transferring regular sensor data within a network in a single container.

C. Communication

In order to achieve scalable data processing, it is clearly necessary for nodes to communicate. The need for communication is however a major source of energy consumption (because transceivers are left on) and delay (because of contention for the medium) and therefore it is crucial that the architecture makes efficient use of the communication system.

Our use of the communication system is quite different from that of many typical address centric networks. For example, the pub/sub component receives a packet and its filtering module may inspect it and decide whether the packet should be forwarded. Execution of queries typically requires that in-network processing is used; typically many packets are received from child-nodes and a computation based on their data payload is performed and the result is transmitted. And query processing in a single broadcast domain may need the communication system to simply perform contention for the medium. For example MAX and MIN of sensor readings in a single broadcast domain can be computed efficiently using a prioritized medium access control protocol. Because of these reasons, the communication component must expose details rather than hiding them.

Our architecture follows these principles in the following way. The communication abstraction layer exposes primitives for other parts of the architecture to use in innovative ways in order to achieve scalable data processing. And the component "execution" provides address-centric services.

We note that the components Timeslot assignment and cluster formation configure the communication system. These can be used to optimize the execution of an ongoing continuous query, for example, if node N_1 and node N_2 transmits packets to node N_3 and N_3 aggregates this information and forwards this aggregate to N_4 then it is desirable (from the perspective of attaining low latency) that the transmissions from N_1 and N_2 has a lower timeslot id than the transmission from N_3. The timeslot assignment can also consolidate timeslots so that one node has a long wakeup time instead of two short wakeup times in a TDMA cycle. The functionality of these components can also be used by the query optimizer to setup the timeslot assignment and cluster formation when a new query (especially for a continuous query) is setup.

D. Repositories

The Cross-Layer Repository (XLR) is the location for the storage of meta data generated by local instantiations of the reference architecture and also meta data generated by remote instantiations. The purpose of maintaining this meta data is that it may be used by several other components within the architecture in order to achieve optimizations and scalability that would otherwise not be possible. For example, knowledge of the physical location of leaf instantiations of the reference architecture may be used to optimally direct certain spatial queries to some leaf nodes, and not others. It cannot be assumed that this information is known a-priori or that this information is even static. For example, node mobility may influence the outcome of certain types of queries, and being able to maintain certain types of meta-data at parent instantiations of the reference architecture may make better optimizations possible.

Query based meta-data In order to be able to plan and execute queries dynamically, it is necessary to know and store what types of queries can be executed at other instantiations of the reference architecture. To achieve effective query planning, it is also required that some additional information such as the cost of the query, be maintained, and that cost information may be dependent on how the query is implemented at that node. For example, a certain query might have one cost when the overall query is executed in such a way as to ensure a very timely response, but may have a different cost when the query is executed in such a way to minimize the amount of messages required to execute that query, i.e. optimize based on energy consumption. The cross layer repository is responsible for the consistent storage of meta-data that captures this information.

One proposed solution is that the XLR maintains a tuple set (N_i, Q_i, C_i, O_i) capturing this information, where N_i is the node where the query is to be executed at, Q_i is that query, C_i is the cost of that query given that it is optimized according to O_i.

Sensor node meta-data Additional information pertaining to the nodes themselves may also be maintained as meta-data. This would include information such as routing tables and also the physical topology of the network, for example the location of nodes.

V. Related Work

There is currently no reference architecture specifically focusing on scalable data processing. There are however related architectures.

RM-ODP [7] is a Reference Model for Open Distributed Processing. It shares our goal of addressing heterogeneity and interaction between objects in different organizations. But it does not address scalability in terms of data processing.

Ultra-Large Scale Systems (ULS) [8] is a study made by Software Engineering Institute at Carnegie-Mellon University for the US Department of Defense. The study discusses how to design software systems comprising more than one billion lines of code and where these software systems are highly distributed and deployed across different organizations. The report about ULS differs from our paper in that ULS provides a roadmap whereas we provide a reference architecture. Also, ULS focuses on interoperability and negotiation between different systems and suggests the use of mechanism design. These issues are not in the scope of our paper.

Scalable Querying of Sensor Networks from Mobile Platforms Using Tracking style Queries (SENSTRAC) proposed in [9] is an approach with some similarities to the our proposed architecture. SENSTRAC aims at querying sensors through mobile nodes (mobile phones, PDAs) using publish subscribe mechanisms. SENSTRAC assumes resource rich sensor nodes integrated in to the infrastructure. Due to the mobility aspects of applications, SENSTRAC instead of in-network aggregation of query results, suggests transformation of queries into the

subscriptions to the topics published by the sensor nodes to incorporate increasing number of querying nodes to meet the scalability requirements. This aggregation at the subscription level is proposed by a separate algorithm periodically computing sensors of interest. SENSTRAC uses leased based subscriptions in which subscription is valid for a limited time period and does not require unsubscription procedure mainly because of the mobility aspect. SENSTRAC has two types of mapping, one from query to the topics and one from topics to the sensor nodes. The sensor network architecture proposed by SENSTRAC is a broker based architecture in which subscriptions and publish offers are sent to the broker nodes in a static grid-cell network structure. In addition to intra-cell communications, the broker nodes are also used for inter-cell communications to avoid redundancy of sensed data.

Scalability, in terms of the number of nodes within a Cooperating Objects Network, has also been studied in the context of the IPAC project [10]. IPAC aims at delivering a middleware and a service creation environment for developing embedded, collaborative and context-aware services in mobile nodes equipped with sensing devices. IPAC relies on short range communications for the ad hoc realization of dialogs among collaborating nodes. The networking capabilities of IPAC are based on rumour spreading techniques, a stateless and resilient approach, and information dissemination among embedded nodes. One of the key advantages of this scalable ad-hoc network construction/communication mechanism is the ability to integrate new mobile nodes and new sensing elements in an efficient manner. However, in contrast to the reference architecture proposed in this paper, scalability with regards to multiple gateways has not been considered.

TAG [11] is a query processor for WSN. It allows the execution of restricted SQL queries and it uses in-network aggregation by creating an aggregation tree (called routing tree in [11]) to improve scalability. Our architecture differs from the one in TAG in that (i) our architecture may take advantage of query processing in a broadcast domain using the MAC to improve scalability, (ii) we consider queries that may span multiple WSNs and (iii) our architecture considers query optimization (for example by taking knowledge of the network topology into account).

VI. Ongoing work

The next steps of our ongoing efforts are now to focus on the development of a prototype implementation of the complete architecture, i.e. the query manager, publish subscribe engine, communication layer, system management layer, cross layer repositories, etc. As part of the requirements specification phase, different real world scenarios were identified and for the first prototype, we will implement one of them. The first challenge in this regard is the selection of an appropriate simulation platform which can provide the required functionalities for implementing the proposed architectural components. The simulation platforms that we have investigated so far do not completely fulfill our requirements. Therefore, additional features will be added to the existing platform(s) for developing the prototype. For example, the prototype implementation will have to be executable at the different levels of the network namely the portal, gateway and broadcast domain levels. This is not supported by any existing simulator. In parallel, we will be refining the architecture with more low level details on architectural components, interaction between the different layers and addressing open issues identified in the design phase such as data models for the cross layer repository and syntax/semantics rules of query language(s). In addition to the prototype, an evaluation methodology for the reference architecture will be devised to validate that the proposed architecture achieves its goals. The validation process will ensure the conformance of the specified requirements with the prototype.

Acknowledgements

This work was partially supported by CONET, the Cooperating Objects Network of Excellence, funded by the European Commission under FP7 with contract number FP7-2007-2-224053, ARTISTDesign Network of Excellence on Embedded Systems Design, funded by the European Commission under FP7 with contract number ICT-NoE-214373 and the Portuguese Science and Technology Foundation (Fundação para Ciência e Tecnologia - FCT) and SmartSkin project supported by ISEP and by the European Union under the project IPAC (#224395) and the Cyprus national project MELCO (#TIIE/OPIZO/0308/(BIE)/14) and by Science Foundation Ireland under Grant No. SFI/08/CE/I1380 (Lion-2).

References

[1] P. Marrón, D. Minder, and The Embedded WiSeNts Consortium, "Embedded WiSeNts research roadmap," IST/FP6 (IST-004400), 2006.

[2] S. Karnouskos, "Research roadmap on cooperating objects, section 2.1: Definitions, available at http://www.cooperating-objects.eu/roadmap," 2009.

[3] A. Arora, R. Ramnath, E. Ertin, P. Sinha, S. Bapat, V. Naik, V. Kulathumani, H. Zhang, H. Cao, M. Sridharan, N. Seddon, C. Anderson, T. Herman, N. Trivedi, C. Zhang, R. Shah, S. Kulkarni, M. Aramugam, and L. Wang, "Exscal: Elements of an extreme scale wireless sensor network," in *RTCSA '05: Proc. of the 11th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications*, 2005, pp. 102–108.

[4] A. Rowe, M. Berges, G. Bhatia, E. Goldman, R. Rajkumar, L. Soibelman, J. Garrett, and J. M. F. Moura, "Sensor Andrew: Large-scale campus-wide sensing and actuation," Carnegie-Mellon University, Pittsburgh, PA, Tech. Rep., 2008, available at project website: http://www.ices.cmu.edu/censcir/resources/SensorAndrew-Tech-Report.pdf.

[5] "ARTEMIS EMMON project web page, http://www.artemis-emmon.eu/objectives.html," 2009.

[6] "Article I of Regulation (EC) No 852/2004 of 29 April 2004 on the hygiene of foodstuffs, http://www.fsai.ie/uploadedfiles/consol_reg852_2004.pdf."

[7] "RM-ODP, http://www.rm-odp.net/," 2010.

[8] P. Feiler, R. P. Gabriel, J. Goodenough, R. Linger, T. Longstaff, R. Kazman, M. Klein, L. Northrop, D. Schmidt, K. Sullivan, K. Wallnau, and B. Pollak, *Ultra-Large-Scale Systems: The Software Challenge of the Future*, 2006.

[9] S. Pleisch and K. P. Birman, "SENSTRAC: scalable querying of SENSor networks from mobile platforms using TRACking-style queries," *Int. J. Sen. Netw.*, vol. 3, no. 4, pp. 266–280, 2008.

[10] "IPAC integrate platform for autonomic computing, http://ipac.di.uoa.gr/," 2009.

[11] S. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong, "TAG: a Tiny AGgregation service for ad-hoc sensor networks," in *OSDI '02: Proceedings of the 5th symposium on Operating systems design and implementation*. New York, NY, USA: ACM, 2002, pp. 131–146.