Sensor Network Grids: Agent Environments combined with QoS in Wireless Sensor Networks

Kostas Stathis[1], Stella Kafetzoglou[2], Symeon Papavassiliou[2] and Stefano Bromuri[1]
[1] Department of Computer Science, Royal Holloway, University of London, UK
kostas@cs.rhul.ac.uk, stefano@cs.rhul.ac.uk

[2] School of Electrical and Computer Engineering, National Technical University of Athens, Greece
skafetzo@netmode.ntua.gr, papavass@mail.ntua.gr

Abstract

We present a distributed systems architecture that uses Grid computing to combine basic nodes of wireless sensor networks with complex sensor nodes of wired networks. Three kinds of complex sensor nodes are identified: objects, agents and containers. The decision making capabilities of the complex nodes are then combined with a quality of service (QoS) framework for gathering data from the wireless sensor nodes. The resulting combination provides a powerful conceptual framework for developing ambient intelligence and ubiquitous computing applications on the Grid.

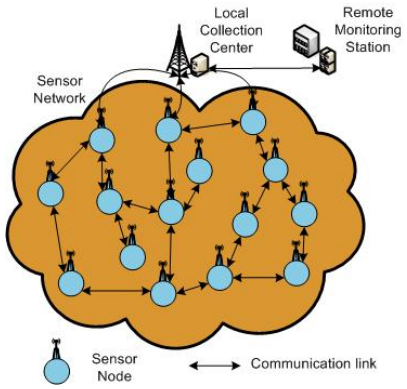

Figure 1. A wireless sensor network

1. Introduction

Sensor networks [1] are computer networks of many, spatially distributed sensor nodes that allow the network to monitor wirelessly a variety of physical conditions or parameters, possibly at different geographical locations. Nodes in a sensor network are small and inexpensive devices, so that they can be produced and deployed in large numbers. One implication of this is that their resources in terms of energy, memory, computational speed and bandwidth are typically severely constrained by the underlying hardware used. Due to power and transmission range limitations, data dissemination in sensor networks is typically carried out as a collective operation, in which sensors collaborate to get data from different parts of the sensor network to the information sinks, or collection centers [1].

A typical multi-hop sensor network is presented in Figure 1, where usually there exists a local collection center where sensors send their collected data for further processing and decision making. The collection center can communicate via a wireless or wired infrastructure with a remote monitoring station for execution of more complex tasks. Sensor networks become increasingly important in new ubiquitous computing and ambient intelligence applications such as environment monitoring, robotics, intelligent cars and traffic systems, smart homes, health monitoring and industrial automation [1].

As part of a platform we are developing in ARGUGRID [2], we investigate extensions of Grid computing that enable applications to access and affect the physical environment via the Grid. We are motivated by the fact that today's existing infrastructures have focused on computational and data Grids. These address the requirements for high computational power and data storage for applications with increasing demands in fields like high energy physics, meteorology and biomedical computations. However, recently Instrumentation Grids have emerged which focus on the creation of a coherent collection of services that can allow the remote configuration of partitions and controls of a physical instrument and a better integration with the computational Grid. Our work here mainly refers to Sensor Grids, which can be

viewed as an instance of Instrumentation Grids, whereby wireless sensor networks are integrated within a distributed computing/storage resource sharing environment. In this context we propose an architecture that integrates wireless sensor networks with wired environments of interacting cognitive agents. The emphasis in this paper, with respect to the Grid, is given to the extension of the Grid paradigm to add sensors as new type of resources in the network that are part of more complex sensor nodes called objects, agents and containers. This consideration and integration is motivated by the fact that in a sensorized universe there is a great amount of data collected and partially processed by sensors, that needs to be further analyzed, processed and stored using the computational and data storage resources of the Grid. Such a powerfull combination would enable the development and deployment of several useful applications and systems, such as early warning natural disaster systems or intelligent transportation systems. A significant integrated feature of our work, apart from the architecture of objects, agents and containers, is the inclusion of a quality of service (QoS) paradigm in support of practical ubiquitous computing and ambient intelligence applications[13].

The rest of the paper is structured as follows. In section 2 we present the main components of a distribusted systems architecture, referred to as *Sensor Network Grid*. Such a Grid motivates the need for a QoS paradigm, the details of which are presented in section 3. The intergration of the rest of the architecture with the proposed QoS paradigm is presented in section 4, while section 5 concludes the paper.

2. Sensor Network Grids

A *Sensor Network Grid* (SNG) is a proposed Grid infrastructure [4] aiming at connecting wired computers of the kind employed in local collection points or remote base stations. The main purpose of an SNG is to interface sensor nodes communicating via a wireless network with virtual sensor nodes communicating via a wired network. The basic element of an SNG is the notion of *compound sensor nodes* (CSNs), virtual entities that are characterized by (a) a unique identifier, (b) a presence that can be sensed by other CSNs in the wired environment supported by the SNG, and (c) a specific internal organization. Three types of CSNs are proposed: *object nodes*, *agents*, and *containers*.

2.1 Object Nodes

An object node is organized as shown in Figure 2. We assume a message-based communication model between object nodes and the Grid in which these nodes are created and discovered as resources. Messages between object nodes are received by *triggers* and sent by *emitters*; these act as

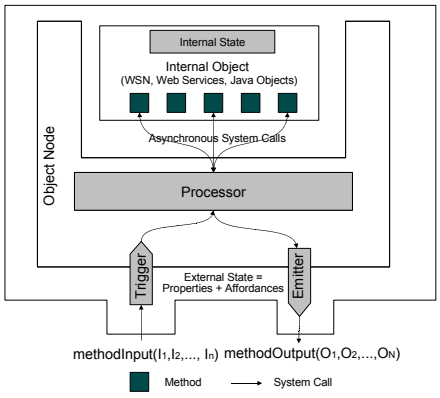

Figure 2. An object node

low-level channels that transport messages in a wired network like the internet. We also assume that the object node wraps an *internal object* whose functionality can be accessed via the object node. It is through such an internal object that the node would link with the base station of a Wireless Sensor Network (WSN), as described in [3]; we use an object node pattern to implement this. In the same way the node accesses other objects or Web Services.

To ensure asynchronous and bidirectional communication, we define a component called *processor*, that manages the system calls received by the object node (either by the trigger part of the node or by the internal object); the processor returns immediately the control to the caller. Messages received by the trigger result in the processor calling a *method* and its input parameters. The method call will typically result in the output of the call to the caller transmitted as a message via the emitter. Alternatively, the processor can receive messages by the internal object, in which case a message will be typically transmitted to the environment via the emitter.

After a method is executed the object's internal state changes. We distinguish between the *invisible* from the *visible* part of an object node, thus separating what can or cannot be sensed of the node in the environment. Changes in the internal part may cause changes to the *external state* of the node. Such a state contains a set of properties and the *affordances* of the object, a description of the perceived methods that are public to other nodes at any one time.

2.2 Agent nodes

Object nodes are not autonomous in that their processor executes methods as long as they have been called by another object in the system. In some applications, therefore,

we would need to have special kinds of nodes that can take decision for themselves, without necessarily serving all the "methods" that are requested of them. To accommodate this requirement we introduce nodes that are *agents*.

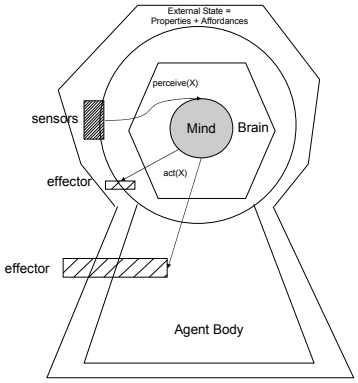

Figure 3. An agent node

An agent node is organized as an extension of the PROSOCS agent architecture [12], as shown in Figure 3. We use virtual sensors and virtual effectors as the agent interface towards an environment and we attach them in a complex component that we call the *body* of the agent. The body situates the agent within an environment and contains a *brain* to connect the various sensors attached to it. The brain also provides an interface to the *mind*, a cognitive component giving the agent the ability to reason logically and make decisions. This mind-brain separation allows different cognitive models of agency to be interfaced to the body and make this kind of node more flexible. A user can use an agent's body to access the electronic environment, in which case the brain of the agent provides simply a convenient interface for the user to select actions using his own mind.

2.3 Container nodes

Complex nodes such as objects and agents are deployed within compound nodes that we call *containers*. As shown in Figure 4 the container has a *state* that holds a directory of all the agents, objects, and sensors in it, including information about their topology and configuration. Interactions and communication within the container is managed by the container's *environment monitor*. This component ensures that interaction and communication within the component follows the *physical laws* specifying the way communication and coordination is implemented in the container.

Another important component is the *interface* of a container with the external environment. Firstly, this needs to

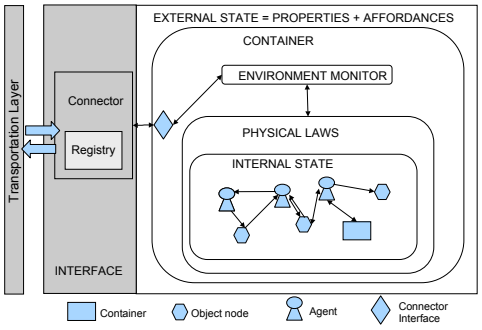

Figure 4. A container node

specify the anchoring between the external environment in which the container is situated from the *internal state* of the container. To provide this anchoring we assume a *connector* managing signals and messages in and out of the container. To achieve this we rely upon a *transportation layer* so that when something happens in the container can be sensed by the external environment and vice versa. Hence, the connector is the part of the middleware supporting agents to communicate with other agents in different containers. The connector also relies upon a *registry* with the references of other agents, object nodes and containers. Assuming a common communication framework between components, this interface can support the integration of heterogeneous components by linking them to form a more complex system. In addition, the interface is used to publicize the *external state* of the container to outside connectors. This external state too can be understood as a set of properties describing the container and the container's *affordances*. These include the ways in which an agent node can configure itself (or other basic, object, agent, and container nodes) to became part of the container's state.

2.4 The SNG Universe

The design of a complex multi-agent system environment can now be composed as a complex SNG of heterogeneous container, agent, object, and basic sensor nodes. The overall system is identified by a top-most container that we refer to as the *universe*. Figure 5 revisits the original sensor network of Figure 1 as a universe of artificial and human agents. These agents interact with other agents and objects seemlessly, while their interactions is facilitated by blending sensor nodes in a mixed (virtual and physical) global environment.

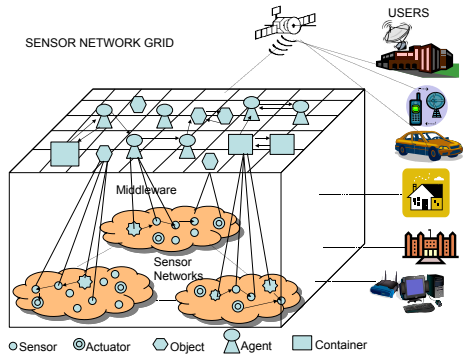

Figure 5. An SNG universe

3. A QoS paradigm for data gathering in SNGs

The combination of wireless sensor networks and of the Grid infrastructure can find many uses, particularly where real-time applications require data provided by the sensor nodes to perform complex computations and decision making, such as adjusting the parameters of an experiment, or controlling the traffic of vehicles within a specific area. However, the unique characteristics of the wireless sensor networks have to be taken into consideration on scheduling the gathering of data. Data collected by the sensor nodes are transmitted to the local collection center and then made available to agents, objects or the containers by the middleware. In some cases, mainly due to the limitations and constraints of the energy resources of sensor nodes, each sensor can either be in active mode gathering and forwarding data or in sleep mode, thus defining a measurement epoch. In that case a sensor node may be synchronized to turn on for a specific period of time (depending on the application), to collect data and send them to the base station or collection center. After this task is completed, the node may turn off for some period of time. In this scenario when an agent accesses the object node, the object node responsible to be a wrapper round the sensor node should return the last measurement epoch to the agent. By allowing that, each sensor conserves energy thus extending the overall lifetime of the sensor network. All sensor measurements are also stored into a repository, which could be either a common Grid repository or an object, so as applications have access to all data, including past gathered measurements.

Because of the limited energy resources of the sensor nodes, data aggregation is performed along the transmission to the collection center. In-network processing has been considered and addressed several times in the literature in similar environments, however in most cases the QoS requirements are not taken into consideration [9], [10]. In our architecture, in order to provide an energy efficient data gathering and dissemination while satisfying QoS we propose a two phase approach. During the first phase, a leveling algorithm is used to define the different levels of the network that the sensor nodes belong to and during the second phase a modified Q-DAP algorithm [15] is used to perform the data aggregation.

3.1 Phase 1: The leveling algorithm

In order to decide which nodes are located closer to the local collection center we use the notion of network levels [11]. A simple algorithm for computing the different levels is adopted [14]. Initially the collection center has the lowest level (0), and all the other nodes infinite (unknown). Moreover, a simple rule is used to establish the definition of levels: if the lowest level of a given's node neighbor is i, then the local's node level is i+1. In this way a tree is established that is used for the collection of data in each gathering round.

At the beginning of the algorithm, each node broadcasts a HELLO message with the following fields: a) level(i) - the level of the transmitted node, b) my_ID - the local's node identification medium access control (MAC) address, and c) parent_ID - the identification (MAC) address of the designated parent of the localnode.

Level	*Neighbour*	*Child_flag*	*Parent_flag*
$i-1$	$ID_1(i-1)$	0	1
	$ID_2(i-1)$	0	0
	⋮	⋮	⋮
	$ID_j(i+1)$	0	0
i	$ID_1(i)$	0	0
	$ID_2(i)$	0	0
	⋮	⋮	⋮
	$ID_j(i)$	0	0
$i+1$	$ID_1(i+1)$	1	1
	$ID_2(i+1)$	0	0
	⋮	⋮	⋮
	$ID_j(i+1)$	1	1

Table 1. A neighboring status table.

Throughout the operation of the algorithm a local node updates its level(i) according to the level information that receives from its neighbors. The field parent_ID is the node's ID from which the local node obtained its level. If more than one node reported the same level to the local node, a node is used arbitrarily to be its parent node. In such cases the identification of the parent node could be also based on more sophisticated techniques, that take into account different parameters such as available battery or distance between the nodes.

Based on the rules described above, each node builds a neighboring status table as shown in Table 1. Note that the nodes with ID_1(i+1) and ID_j(i+1) are the local node's

children and the node with ID_1(i-1) is its parent. Moreover, the rest of the nodes at level i-1 are candidates to be its parent in case its parent node fails. Situations where the parent node fails may occur either when a sensor node has exhausted its energy, or the link is lost due to bad channel conditions, or some nodes are entering into sleep mode for energy conservation purposes. In that way, regular recreation of the tree is not necessary. The sensor nodes periodically or upon request of the collection center and/or the application transmit their sensed data by relaying it to their parents until it reaches the final destination.

3.2 Phase 2: Data gathering

The approach described in this section presents a distributed and effective method to perform in-network processing in order to conserve energy as data traverse the sensor network, while at the same time we satisfy the QoS constraints posed by the application. Data from neighboring nodes are expected to present significant correlation and therefore data aggregation is performed to reduce the amount of data traversing the network. Each sensor node is assumed to generate its own packet of data and sends this to the collection center. At the same time a node is responsible to forward the data packets of its neighbors. The choice of whether or not a node performs data aggregation or just relays the packet to the next hop neighbor, according to the QoS constraints, is made independently and in a distributed manner. The quality constraint to be met is the *end-to-end delay constraint* D, which we assume is posed by the application. Alternatively, similar delay constraints could be imposed by limitations that stem from the measuremet epoch duration. It is noted that the choice of the delay as a constraint is not exclusive and additional QoS constraints may be imposed as well (e.g. packet loss constraint).

When a sensor node receives a packet from one of its neighbors, it has the choice to perform data aggregation. The following cases may occur:
a) If the delay D can be met, with probability γ the node waits (i.e. packet is deferred) for a certain time interval τ for other packets to arrive and aggregates the data received forming one single packet. When the time interval expires it forwards the data to the next sensor node where the same procedure is followed until the data reaches the collection center. With probability $(1-\gamma)$ the node simply forwards the data.
b) The node evaluates if the delay D can be met only if the packet is not deferred, and forwards it to the next hop.
c) The node evaluates if the delay D cannot be satisfied in any case, and discards the packet.

If a packet of collected data is delivered to the collection center within the given constraint D, it is considered to be a successful delivery. Otherwise, the data is obsolete and therefore is discarded. This approach succeeds in decreasing the communication load of the network by not transmitting the obsolete data [15]. Parameter γ is a configurable parameter of the network, while τ depends on the node's placement in the network. Intuitively, the closer a sensor node is to the collection center the longer has to wait in order to gather data packets from more distant nodes, and form a single packet to transmit to the collection center.

4 Integration scenario

A Sensor Network Grid provides a platform that enables intelligent decisions concerning the sensor networks and corresponding applications, especially when the infrastructure is equipped with heterogeneous sensor nodes, while various applications with diverse and dynamic requirements are supported. According to the infrastructure introduced in this paper, the objects are virtual sensor nodes and represent the status of the actual sensors in the network. Agents can communicate with the objects, becoming aware of the sensor nodes' status and characteristics, through the object's affordances part and consequently make decisions concerning the application.

Let us consider a scenario where the user requests for the periodic collection of specific data from sensor nodes that are distributed throughout the network with a predefined QoS requirement, e.g. posing a certain delay constraint D from the time of the data generation until the point that the data reach their final destination. The QoS requirement information is passed to the agents of the network, which in turn have to assess an energy efficient data gathering plan for the sensor sub-networks. Considering the data gathering scheme described in section 3, the agents can determine an appropriate value for the τ parameter for every sensor node, depending on their placement in the sensor network (e.g. level of the sensor node within the sub-network).

More specifically, since agents can obtain global knowledge of the sensor network or sub-network, each node's level(i) as well as the depth of the tree and the average data transmission and propagation time can be obtained, and using the following relation the appropriate τ parameter for every node at a given level i can be calculated as:

$\tau = level_{max} - level(i) * s_{avg} + e$

where $level_{max}$ is the total amount of levels in the tree, $level(i)$ is the local's node level, s_{avg} represents an average value for the transmission and the propagation duration between two neighbouring nodes and e is the timing parameter to prolong the deferred period in case of collisions, so that to avoid them.

After the appropriate values of parameter τ are defined by the agents, they are transmitted to the objects that represent the given sensor sub-network, and then through the collection center of the sensor sub-network these values are

distributed to the sensor nodes. The data gathering is then accomplished following the procedure described in phase 2 of the previous section. Following this procedure, we achieve not only to transmit fewer packets to the collection center through the wireless part of the network and as a result less energy is depleted in the sensor network, but at the same time we manage to provision and apply certain QoS requirements requested by the applications. Furthermore the proposed framework provides the flexibility to support over the same infrastructure different applications and scenarios that may present diverse QoS requirements.

5. Concluding remarks and future work

We have presented a distributed systems architecture seeking to integrate a QoS paradigm in wireless sensor networks with a Grid of virtual sensor nodes called objects, agents and containers. This QoS paradigm supports suitable combination of complex nodes to access the physical environment according to constraints set by a user or agent. This in turn allows our system to have the potential to integrate heterogeneous multi-agent systems environments of increased complexity.

Our approach complements existing proposals where agents are directly related with the nodes of a sensor network (e.g. see [8], [7], and [6]). The rationale for our choice is based on pragmatic constraints that are normally imposed on a sensor node and the conflict between these and computational requirements of cognitive agents that must do symbolic processing and exhibit logical reasoning. We also differ from attempts to combine wireless sensor networks and the Grid, for example [5], in that our Grid contains agents, objects, and containers as additional nodes.

An initial implementation of the proposed architecture has being developed in ARGUGRID [2]. Sensor networks will be incorporated as Grid resources where users will have access to gather data and monitor the environment. The emphasis will be mainly placed on the agents and how they use sensor networks to provide services that users have requested, taking into consideration the specific QoS posed. Internal reasoning and decision making within and between agents are envisioned to play a key role in the delivery of services.

Acknowledgements

This work was partially supported by the EU project ARGUGRID-IST-035200.

References

[1] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. Wireless sensor networks: a survey. *Computer Networks*, 38(4):393–422, 2002.

[2] ARGUGRID (ARGUmentantion as a foundation for the semantic GRID). http://www.argugrid.eu/.

[3] A. Dunkels, T. Voigt, J. Alonso, H. Ritter, and J. Schiller. Connecting Wireless Sensornets with TCP/IP Networks. In *Proceedings of the Second International Conference on Wired/Wireless Internet Communications (WWIC2004)*, Frankfurt (Oder), Germany, Feb. 2004.

[4] I. Foster. The Grid: Computing without Bounds. *Scientific American*, April 2003.

[5] M. Gaynor, S. L. Moulton, M. Welsh, E. LaCombe, A. Rowan, and J. Wynne. Integrating wireless sensor networks with the Grid. *IEEE Internet Computing*, 8(4):32–39, August 2004.

[6] D. Georgoulas and K. Blow. Making motes intelligent: An agent-based approach to wireless sensor networks. *WSEAS on Communications*, 5(3):515–522, March 2006.

[7] S. Hussain, E. Shakshuki, and A. W. Matin. Agent-based system architecture for wireless sensor networks. In *20th International Conference on Advanced Information Networking and Applications (AINA 2006)*, pages 296–302, Vienna, Austria, April 18-20 2006.

[8] B. Karlsson, O. Bckstrm, W. Kulesza, and L. Axelsson. Intelligent sensor networks - an agent-oriented approach. In *Workshop on Real-World Wireless Sensor Networks (REALWSN'05)*, Stockholm, Sweden, June 20-21 2005.

[9] B. Krishnamachari, D. Estrin, and S. Wicker. The impact of data aggregation in wireless sensor networks. In *Proceedings of thr IEEE 22nd International Conference on Distributed Computing Systems Workshop*, pages 575–578, July 2002.

[10] S. Olariu, Q. Xu, and A. Zomaya. An energy-efficient self-organization protocol for wireless sensor networks. In *Proceedings of Intelligent Sensors Sensor Networks and Information Conference*, pages 55–60, December 2004.

[11] I. Solis and K. Obraczka. The impact of timing in data aggregation for sensor networks. In *IEEE International Conference on Communications, Vol. 6*, pages 3640 – 3645, June 2004.

[12] K. Stathis, A. C. Kakas, W. Lu, N. Demetriou, U. Endriss, and A. Bracciali. PROSOCS: a platform for programming software agents in computational logic. In J. Müller and P. Petta, editors, *Proceedings of the Fourth International Symposium "From Agent Theory to Agent Implementation" (AT2AI-4)*, pages 523–528, Vienna, April 13-16 2004.

[13] K. Stathis and F. Toni. Ambient intelligence using KGP agents. In *European Symposium on Ambient Intelligence (EUSAI04)*, volume 3295 of *LNCS*, pages 351–362. Springer, September 2004.

[14] S. Xu, S. Papavassiliou, and S. Narayanan. A layer-2 multi-hop ieee 802.11 architecture: Design and performance analysis. *IEE Proceedings Communications Journal*, 151(5):460–466, October 2004.

[15] J. Zhu, S. Papavassiliou, and J. Yang. Adaptive localized qos-constrained data aggregation and processing in distributed sensor networks. *IEEE Transactions on Parallel and Distributed Systems*, 17(9):923–933, September 2006.