A Task Allocation Middleware for Wireless Sensor Networks in a Multi-Agent Environment

Luca Caviglione[1], Mauro Coccoli[2] and Alberto Grosso[2]
[1]*Institute of Intelligent Systems for Automation (ISSIA), National Research Council of Italy (CNR) via de Marini 6, I-16149, Genova, Italy*
[2]*Department of Informatics, Bioengineering, Robotics and Systems Engineering (DIBRIS), University of Genoa, via Opera Pia, 13, I-16145, Genova, Italy*

Keywords: Task Allocation, Middleware, Wireless Sensor Network (WSN), Multi-Agent Systems.

Abstract: This paper presents the design of a task allocation middleware for the coordination of a Wireless Sensor Network (WSN) of embedded devices. Acquisition and distribution of new tasks are performed via a multi-agent system, while service oriented principles are used to handle data gathered from the field. Also, an ad-hoc component has been designed to reduce the impact of heterogeneous networks (e.g., long-haul satellite links). Lastly, we showcase an experimental setup to prove the effectiveness of the approach when used to enhance a forest fire prevention application.

1 INTRODUCTION

The division of work in activities, tasks and sub-tasks implies the availability of *task allocation techniques* to define specific objectives and responsibilities (Quiñonez et al., 2011). By using this design, it is possible to have parallel/specialized execution flows, a versatile resource allocation scheme, and the delegation of duties to humans. Such features also increase the robustness of the overall framework, which is critical when deployed into unpredictable or time-varying environments. To effectively implement such requirements, a *multi agent community* is one of the most suitable solutions (Shehory and Kraus, 1995). However, some deployments, like those using a Wireless Sensor Network (WSN), may lead to very large decision spaces requiring adjustable data collection disciplines, or task adaptive algorithms. In this respect, the Service Oriented Architecture (SOA) offers *loose coupling*, *statelessness*, *composability* and *discoverability*, thus making the merge between a WSN and the task allocation middleware more efficient (Ibbotson et al., 2010). Besides, the SOA enables sensors to be discovered, accessed, invoked and controlled via the Web (Delicato et al., 2005) and (Golatowski et al., 2003). Software *agents* are another important block to offer intelligent and flexible entities making possible the cooperation, competition, and coordination for specific goals. To exploit their social abilities, agents are often organized in a Multi-Agent System (MAS), which is an interesting candidate to implement middleware services (e.g., object request brokers and directory services) (Omicini and Rimassa, 2004). This paper presents a middleware based on a cooperative multi-agent society for managing the allocation of tasks in a WSN with time-varying topology and a broad set of functionalities. To prove the effectiveness of our design, we showcase the experimental testbed of a forest fire prevention application. The contribution of this work is the definition of a task allocation framework based on the SOA paradigm, making the assignment problem independent both from the evolution of the WSN and the target operations. Additionally, it enables to dynamically match jobs against resources, for instance through optimized strategies, such as the execution time, Quality of Service (QoS) requirements, and power consumption constraints.

The remainder of the paper is structured as follows: Section 2 reviews task allocation mechanisms for WSN, and Section 3 presents the architecture of the system. Section 4 outlines network design aspects and Section 5 showcases the task allocation middleware. Section 6 describes the forest fire prevention application. Lastly, Section 7 concludes the paper.

Caviglione L., Coccoli M. and Grosso A..
A Task Allocation Middleware for Wireless Sensor Networks in a Multi-Agent Environment.
DOI: 10.5220/0004634803250330
In *Proceedings of the 3rd International Conference on Sensor Networks* (SENSORNETS-2014), pages 325-330
ISBN: 978-989-758-001-7
Copyright © 2014 SCITEPRESS (Science and Technology Publications, Lda.)

2 WSN AND TASK ALLOCATION

Usually, a WSN is composed by a large number of heterogeneous sensing devices forming highly dynamic network topologies due to impairments in the wireless link, hardware/software outages, external hazards, or battery drains. Nevertheless, to flatten costs, a WSN is required to concurrently support different *sensing tasks*, mainly by sharing a common resource (e.g., measurement devices or long-haul communication links), or competing for its exclusive control. Therefore, according to constraints imposed by the specific application scenario, each sensor needs to be associated to the best-served task. This raises the problem commonly defined as Multi-Sensor Task Allocation (MSTA). To this aim, in reference (Pizzocaro and Preece, 2009) authors discuss a framework to categorize different instances of the MSTA. They propose a domain-independent taxonomy, which considers the number of sensors, the number of tasks, the time constraints, and the nature of the scenario, i.e., collaborative versus competitive. As regards the MAS, the literature already offers a variety of solutions, especially based upon distributed task allocation techniques (Shehory and Kraus, 1995). Consequently, we decided to exploit such result to coordinate activities belonging to the WSNs, also by taking into account that auction-based systems are widely used in MAS (Faratin et al., 1998). Hence, in order to assign activities, an auction is performed, resulting in agents/sensors bidding a value in a shared currency based on their perceived fitness for that task. Applying this paradigm to the scenario of interest leads again to the MSTA problem. Specifically, the literature proposes several variations of auction and market-based mechanisms, as well as awarding tasks according to suited cost functions, e.g., to guarantee a proper degree of QoS (Pletzer and Rinner, 2010), and by explicitly considering the energy consumption (Edalata et al., 2012). Accordingly, our middleware will model multiple instances of the MSTA problem.

3 THE WSN ARCHITECTURE

The system architecture has been engineered to fit different distributed systems requiring flexibility in the execution of operations/tasks. Yet, it has to be specialized for the WSN case. We consider the reference blueprint as composed by a set of connected entities whose topology and features can dynamically change. In order to manage and orchestrate the WSN, the middleware adopts a MAS deployed among its nodes, and at least one agent has to be located in each of these with proper computational and storage capabilities. Sensors are assumed as attached to an entity offering a web service interface for data retrieval and management. As a result, agents within nodes are the *targets* of the task allocation process. To implement this software design, we use a middleware based on the AgentService framework (Vecchiola et al., 2008) and (Grosso et al., 2003) relying upon the Microsoft .NET Framework and MONO Project. We point out that AgentService already offers many critical features, such as, the support of MASs, proper tools for modeling agent societies, standard web service communication channels, and a workflow engine (Boccalatte et al., 2005). Furthermore, it can distribute agents via servers or resource-constrained embedded devices. In this case, the minimal requirement is a device running the Microsoft .NET Micro Framework (Fox and Box, 2003).

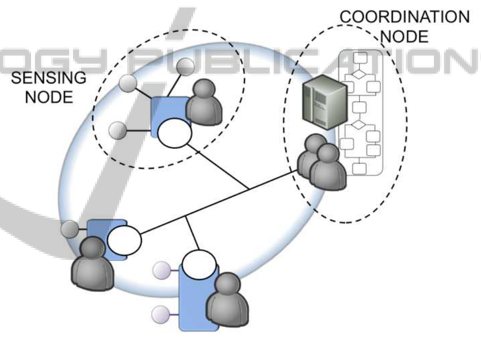

Figure 1: System architecture based on WSN and MAS.

In our approach, the middleware must be able to handle *two* kinds of entities: *sensing* and *coordination* nodes. Sensing nodes are handled in form of services, thus via a set of operations to retrieve measurements and information about their state (e.g., the assigned task). Also, they are provided with basic functions to exploit task allocation (e.g., for bidding). On the contrary, a coordination node requires more computational capabilities, since it contains the core engine in charge of composing and executing the workflow of services. The *communication infrastructure* for the agent society is based on a SOA, which can be extended via third party services. In this respect, Figure 1 depicts the WSN architecture composed by six sensors connected to three embedded devices forming the sensing nodes. Also, a coordination node equipped with the task allocation and workflow engine is present. Owing to the generality of the design, this architecture can be applied to different scenarios, ranging from small use cases requiring high level of reliability in task completion (e.g., to allocate production orders in manufacturing plants), to geographically distributed systems.

4 NETWORK DESIGN ASPECTS

This section introduces the engineering of a specific entity to prevent issues arising when in presence of long-range communication links. The terms node and sensor will be used interchangeably, except when doubts arise.

4.1 Long-range Communication Support

Since nodes could be placed in isolated or rural areas, public network infrastructures or cellular carriers could be absent. As a consequence, some form of long-range communications must be provided, e.g., to deliver measurements to a remote *sink*. Long-range links are nowadays widely supported, both from the viewpoints of hardware and availability as commercial services (Caviglione, 2009a). Though, they can introduce the following drawbacks:

1. generally, the adopted radio technology has features to compensate attenuation/fading, to provide data confidentiality through proper encryption algorithms, and to support channel reservation mechanisms. Alas, they are energy consuming tasks, thus heavily affecting battery-operated nodes;
2. sensors are usually equipped with L2 air interfaces offering short-range communication services. Then, providing medium to long-range data exchanges could require add-ons, both in terms of hardware and software. Even if this can be also guaranteed through L2 bridges, such a choice increases the number of needed devices, and the overall network complexity, e.g., it requires configuration/maintenance/power supply;
3. the standard TCP/IP protocol suite has performance issues when in presence of high $delay \cdot bandwidth$ or heavily faded channels (e.g., GEO links). Moreover, intermittent connectivity, as provided by LEO satellites, could demand for proper application layer countermeasures. A common approach uses ad-hoc protocol suites, e.g., those enabling data custody or deferred transmission. We mention, among the others, the Disruption Tolerant Networking (DTN) (see, e.g., (Mc Donald et al., 2007) for an example of the usage of DTN in a sensor network for monitoring a lake). Unfortunately, they require an implementation available for the target platform;
4. as a partial workaround to 3), many protocol modifications are available, but usually demand for "patching" low-level software components (e.g., the kernel or the drivers), which can be unfeasible for all the devices, for instance due to source-code unavailability.

To cope with issues 1) - 4) a more suitable method is to use proxy-based components, often defined as Performance Enhancing Proxies (PEPs) or "middleboxes". In this way, it is possible to reduce the impact of long-range links on the overall network architecture, while at the same time, avoid a major re-engineer of the WSN. PEPs have been already successfully utilized for Public Safety and Disaster Relief (PSDR) duties, as described in (Caviglione, 2006).

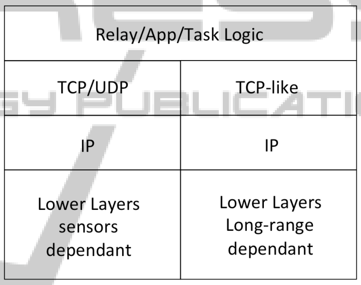
Relay/App/Task Logic
TCP/UDP
TCP-like
IP
IP
Lower Layers sensors dependant
Lower Layers Long-range dependant

Figure 2: Reference protocol of the PEP.

Figure 2 depicts the reference protocol architecture of a PEP designed to merge the local WSN (and/or devices in charge of performing task allocation), with long-range links. Specifically, it is composed by two protocol stacks: one is devoted to manage communications within the WSN (i.e., the left portion), while the other is used to transmit data over the long-range link (i.e., the right part). The ISO/OSI L7 layer is in charge of relaying data, and running local processes. Thus, it can take in custody data to cope with intermittent connectivity, or can perform local task allocation duties, also by enabling a proper feedback with the remote facility (in this case, it is assumed co-located with a coordination node). The availability of a full-featured application layer can be also exploited for dynamically turning-on/off sensors, having only the PEP to constantly communicate with the proper data collection facility, i.e., it acts as a sink. The PEP can be also used to perform data processing/compression, to save bandwidth when in presence of narrow wireless channels, or to use portion of the transmission resource for fading countermeasures, e.g., for Forward Error Correction (FEC). By decoupling the stacks, each specific protocol can be tweaked according to well-given design constraints.

As an example, the L4 of the right stack can be a TCP-like protocol optimized for satellite communications, while the L3 layer used in the WSN can have simplified routing strategies, as well as specific countermeasures to reduce the transmission power and the impact of local channel interferences.

Furthermore, the PEP entity could be standalone, or co-located within a coordination node, which can also be mobile. In this case, let us consider an individual interacting with nodes by exploiting proximity communications (e.g., the IEEE 802.15 substitutes the generic merge of ISO/OSI L1 and L2 defined as *"lower layers sensors dependent"* in Figure 2). Upon approaching a node, he/she can perform the following actions: *i)* issue/collect task-related information and *ii)* gather local measurements or disseminate previously collected data. For what concerns *i)*, the App/Task Logic is in charge of managing and allocating tasks. As regards *ii)*, a proper protocol suite supporting deferred transmissions, data bundling and intermittent connectivity (e.g., a link is only supposed to exist between the local node and the operator) has to be implemented. Owing to the decoupled design, specific choices do not propagate to the rest of the framework, especially in the middleware.

5 MIDDLEWARE DESIGN

This section discusses the layered architecture of the middleware, which is composed by the runtime framework implementing the task allocation process, and a set of Application Programming Interfaces (APIs). To this aim, we specialized the general-purpose middleware AgentService, where each agent represents sensor(s). Also, the device on which sensors are attached/deployed is assumed as the geospatial reference.

5.1 Agents and Task Definition

In our design, agents are the only targets of the task allocation process, and can be classified in *three* categories:

- *sensor agents* – acting on behalf of a single sensor, or a device equipped with a set of sensors;
- *mobile agents* – deployed on mobile appliances, interacting with both sensors and coordination nodes. Their activities can be driven by human operators, as well as autonomous vehicles duly equipped;
- *software agents* – they only perform pure software tasks, such as data retrieval and processing.

We point out that using the network component presented in Section 4.1 prevents the need of a *network agent*. This is another benefit of having a middlebox to decouple the WSN from the rest of the network. From the viewpoint of the middleware, each agent can submit new tasks to the runtime and participate to their concurrent execution, according to the specific capabilities.

To represent a task, the following *tuple* t, is used:

$$t =< U, P, C, L >$$

where, U is the Unique Resource Identifier (URI) associated to the task, P is a set of input/output parameters, C is a collection of required capabilities, and L contains information about the area/node location where the task must be executed. If $L = \texttt{NULL}$, then no specific locations are necessary. Thus, L can be used to discriminate between *localized* and *non-localized* tasks. A possible example of non-localized task is a general weather forecast, while for the localized case could be a group of measurements coming from sensors deployed in a well-defined geographical area. Besides, if a task does not require a feedback, it can be considered terminated once allocated, otherwise results have to be sent back to the runtime engine.

5.2 Task Allocation Process

The runtime framework is organized in *three* layers:

1. *Task Decomposition Layer*: for each new task, it generates the relevant workflow of services;
2. *Workflow Engine Layer*: for each service, it executes the service flow and it submits a task request to the Task Allocation Layer;
3. *Task Allocation Layer*: it collects task requests and performs assignments to agents.

Each agent awaits for new tasks by polling the hosting device. Based on its capability, it may decide to take in charge the execution of a pending task. In this case, the agent will notify such assignment both to the device and the coordinator. Owing to task decomposition, each "simple" task corresponds to a SOA service, while complex tasks are represented with a workflow whose actions are modeled as service invocations. The *Workflow Engine Layer* creates a new instance of the given workflow also by supervising its control flow. Then, upon receiving a new task request (i.e., a service invocation), it routes proper commands to the *Task Allocation Layer* as to complete its assignment to sensor agents. Specifically, the *Task Allocation Layer* offers two functionalities: for localized tasks it arranges their deployment over specific WSN nodes/embedded devices, and it implements the auction-based strategies.

5.3 Deployment of Localized Tasks

When in presence of localized tasks, the *Task Allocation Layer* performs a *"Call for Task"* limited to a specific area. This reduces to finding a proper match among areas and nodes, and task tuples are sent to matching entities. The middleware generates a new task, which is defined as *deployment task*. In order to be accomplished, it is assigned to *mobile agents* that are placed on mobile devices. Obviously, the deployment task can be performed by human operators, or by unmanned vehicles, for instance in presence of hazardous environments.

6 FOREST FIRE PREVENTION APPLICATION

To prove the effectiveness of our design, we showcase a *forest fire prevention* application. Since areas to be monitored are often outback, the system should be disaster-resistant and remotely controllable. Figure 3 depicts the typical reference blueprint composed by different sensing nodes. To guarantee the proper network connectivity, at least one coordination node must be present.

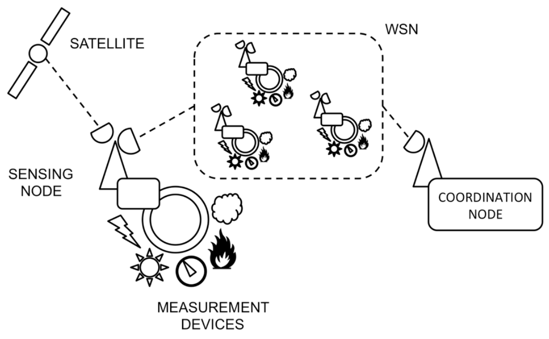

Figure 3: The configuration of a forest fire prevention system including devices and the WSN.

To be successfully adopted, the system uses standard communication infrastructures, e.g., IEEE 802.11b/g/n for medium range communications, UMTS or Ka/Ku satellite links for remote data delivery, and the TCP/IP protocol suite to support Internet services. Then, as discussed, data-handling services are exposed through a MAS-coordinated SOA infrastructure, which also allocates tasks. The coordination node collects measurements from the WSN. Such values can be used to perform analyses and predictions based on the past history.

The IP enables sending data, images, and video streams towards the Internet, constrained by the underlying resource availability. The satellite channel can be considered as a backup, or as the primary long-haul link, according to specific design constraints (e.g., the deployment of the framework in inaccessible or rural areas). We highlight that satellite bandwidth could be expensive or scarce, thus a PEP-like entity would make possible to locally implement data compression, resource reservation mechanisms and data shaping techniques. The middlebox also isolates the WSN, enabling the introduction of specific policies, such as battery preserving scheme, or communication technologies (e.g., ZigBee, IEEE 802.11 or Bluetooth). This design resembles a "mediated" peer-to-peer overlay guaranteeing the access to the WSN via a unique and coherent interface exposed by the single coordination node (Caviglione, 2009b). Also, reconfigurations of sensing nodes will not propagate across the overall system, reducing the complexities, and easing the management of the resulting data flows.

6.1 Evaluation of the Testbed

To evaluate our design, we developed a small testbed in a controlled environment. To this aim, we used 3 devices based on the Tahoe-II development platform, which features the .NET Micro Framework, a Meridian CPU based on a Freescale i.MXS ARM9 processor with 4 Mbytes of Flash and 8 Mbytes of RAM, a 3.5" touch-screen LCD, wired and wireless L2 interfaces, USB ports, and an accelerometer. The choice of the Tahoe-II platform also guarantees the needed flexibility requirements for implementing *adaptive* sensing nodes, i.e., nodes that can be replaced to fulfill specific needs/operations. Also, it enables to simulate hazardous conditions such as the presence of fire. Nevertheless, its on-board display demonstrated how humans could be effectively placed within the "decision loop". To implement the coordination node, we used a standard x86 architecture, while the network relies upon the built-in air interfaces (i.e., IEEE 802.11). Lastly, the satellite link has been emulated by adding proper transmission delays (i.e., via the *netem* tool on a Linux machine).

Trials in such a controlled environment support that: *i*) the middleware can match tasks and agents according to specific objectives; *ii*) human operators can remotely adjust functionalities of the fire prevention system; *iii*) owing to the rapid re-configurability of the system, operators can simulate specific alert/alarm conditions, resulting into an effective tool for the training of personnel; *iv*) the presence of an x86 node allows to aggregate or pre-process data gathered by sensors to save transmission resources, or to recover performance degradations due to excessive transmission delays (e.g., by properly tuning TCP parameters).

7 CONCLUSIONS

In this paper we presented a task allocation middleware for developing adaptive WSN architectures in a multi-agent environment. To prove its effectiveness, it has been used to develop a forest fire prevention application. Besides, the prototypal implementation also demonstrated that using devices also offering *human-to-machine* interaction enables individuals to easily access measurements and functionalities of the WSN.

Future work aims at enriching the middleware with semantic capabilities, for instance by using a set of tuples defined via the Resource Description Framework (RDF). This allows to develop ontology-based techniques, enabling to model the application domain, and to use automatic reasoning for the discovery and decomposition of complex tasks. Additionally, the SOA paradigm allows to easily integrate our middleware with a Geographical Information System (GIS), or to offer the built-in workflow engine as a "service" to schedule resources and personnel when in presence of alarms. Lastly, as a part of our ongoing research, we will evaluate how to merge our solution with a Decision Support System (DSS).

REFERENCES

Boccalatte, A., Coccoli, M., Grosso, A. and Vecchiola, C. (2005). A multiuser groupware calendar system based on agent tools and technology. In *Proc. of the IEEE 2005 Int. Symposium on Collaborative Technologies and Systems*, St. Louis, MO, USA, pp. 144-151.

Caviglione, L. (2006). Introducing Emergent Technologies in Tactical and Disaster Recovery Networks. *Int. J. of Communication Systems*, Wiley, vol. 19, no. 9, 1045-1062.

Caviglione, L. (2009). Can Satellites Face Trends? The Case of Web 2.0. In *Proc. of Int. Workshop on Satellite and Space Communications (IWSSC'09)*, Siena, Italy, pp. 446-450.

Caviglione, L. (2009) Enabling Cooperation of Consumer Devices Through Peer-to-Peer Overlays. *IEEE Trans. on Consumer Electronics*, vol. 55, no. 2, pp. 414-421.

Delicato, F., Pires, P., Rust, L., Pirmez, L. and de Rezende, J. (2005). Reflective Middleware for Wireless Sensor Networks. In *Proc. of the 20th Annual ACM symposium on applied computing (SAC 2005)*, Santa Fe, USA, pp. 730-735.

Edalata, N., Thama, C.-K. and Xiaob, W. (2012). An Auction-based Strategy for Distributed Task Allocation in Wireless Sensor Networks. *Computer Communications*, vol. 35, no. 8, pp. 916-928.

Faratin, P., Sierra, C. and Jennings, N.R. (1998). Negotiation Decision Functions for Autonomous Agents. *Int. J of Robotics and Autonomous Systems*, vol. 24, no. 3-4, pp. 159-182.

Fox, D. and Box, J. (2003). *Building Solutions with the Microsoft.Net Compact Framework: Architecture and Best Practices for Mobile Development*. Addison-Wesley Longman Publishing Co., Boston, MA, USA.

Golatowski, F., Blumenthal, J., Handy, M. and Haase, M. (2003). Service Oriented Software Architecture for Sensor Networks. In *Proc. of the Int. Workshop on Mobile Computing (IMC)*, Rockstock, Germany, pp. 93-98.

Grosso, A., Gozzi, A., Coccoli, M. and Boccalatte, A. (2003). An Agent Programming Framework Based on the C# Language and the CLI. *Journal of .NET Technologies*, vol. 1, no. 3, pp.13-20.

Ibbotson, J., Gibson, C., Wright, J, Waggett, P., Zerfos, P., Szymanski, B.K. and Thornley D.J. (2010). Sensors as a Service Oriented Architecture: Middleware for Sensor Networks. In *Proc. of the 6th Int. Conf. on Intelligent Environments*, pp. 209-214.

Mc Donald, P., Geraghty, D., Humphreys, I., Farrell, S. and Cahill, V. (2007). Sensor Network with Delay Tolerance (SeNDT). In *Proc. of the 16th Int. Conf. on Computer Communications and Networks*, 1333-1338.

Omicini, A. and Rimassa, G. (2004). Towards Seamless Agent Middleware. In *Proc. of the IEEE 13th Int. Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises*, Modena, Italy.

Pizzocaro, D. and Preece, A. (2009). Towards a Taxonomy of Task Allocation in Sensor Networks. In *Proc. of the 28th IEEE Int. Conf. on Computer Communications Workshops*, IEEE Press Piscataway, NJ, USA, pp. 413-414.

Pletzer, F. and Rinner, B. (2010). Distributed Task Allocation for Visual Sensor Networks: a Market-based Approach. In *Proc. of the 2010 4th IEEE Int. Conf. on Self-Adaptive and Self-Organizing Systems Workshop*, Budapest, Hungary.

Quiñonez, Y. de Lope, J. and Maravall, D. (2011). Bio-inspired Decentralized Self-coordination Algorithms for Multi-heterogeneous Specialized Tasks Distribution in Multi-Robot Systems. In *Proc. of the 4th Int. Conf. on Interplay between natural and artificial computation*, Springer Berlin, Heidelberg, pp.30-39.

Shehory, O. and Kraus, S. (1995). Task Allocation via Coalition Formation Among Autonomous Agents. In *Proc. of Int. Joint Conf. on AI '95*, Montreal, pp. 655-661.

Vecchiola, C., Grosso, A. and Boccalatte, A. (2008). AgentService: a Framework to Develop Distributed Multi-agent Systems. *Int. J. of Agent-Oriented Software Engineering*, vol. 2, no. 3, pp. 290-323.