Multi-Agents Supporting Reflection in a Middleware for Mission-Driven Heterogeneous Sensor Networks

Edison Pignaton de Freitas
IDE – Halmstad University – Sweden - PPGC UFRGS - Brazil
PO Box 823. SE - 301 18
Halmstad - Sweden
+46 35 16 78 47
edison.pignaton@hh.se

Marco Aurélio Wehrmeister
PPGC UFRGS - Brazil
Caixa Postal 15.064 - 91.501-970 - Porto Alegre/RS
Brazil
+ 55 51 3316 3561
mawehrmeister@inf.ufrgs.br

Carlos Eduardo Pereira
PPGC UFRGS - Brazil
Caixa Postal 15.064 – 91.501-970 - Porto Alegre/ RS
Brazil
+ 55 51 3316 3561
cpereira@ece.ufrgs.br

Armando Morado Ferreira
Military Institute of Engineering
22290-270 – Rio de Janeiro/RJ – Brazil
+55 21 3873 2298
armando@ime.eb.br

Tony Larsson
IDE – Halmstad University – Sweden
PO Box 823. SE - 301 18 Halmstad - Sweden
+46 35 16 71 68
tony.larsson@hh.se

ABSTRACT

The emerging applications using sensor networks technologies constitute a new trend requiring several different devices to work together and this partly autonomously. However, the integration and coordination of heterogeneous sensors in these emerging systems is still a challenge, especially when the target application scenario is susceptible to constant changes. Such systems must adapt themselves in order to fulfill requirements that can also change during the system runtime. Due to the dynamicity of this context, system adaptations must take place very quickly, requiring system autonomous decisions to perform them without any human operator intervention, besides the first directions to the system. Thus a reflective behavior must be provided. This paper presents a reflective middleware that supports reflective behaviors to address adaptation needs of heterogeneous sensor networks deployed in dynamic scenarios. This middleware presents specific handling of users' requirements by representing them as missions that the network must accomplish with. These missions are then translated to network parameters and distributed over the network by means of the reasoning about network nodes capabilities and environment conditions. A multi-agent approach is proposed to perform this initial reasoning as well as the adaptations needed during the system runtime.

Categories and Subject Descriptors

D.2.11 [**Software Engineering**]: Software Architectures – *Domain-specific architectures and Patterns.*

I.2.9 [**Artificial Intelligence**]: Robotics – *Autonomous vehicles, Sensors.*

I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence – *Multiagent systems.*

General Terms

Management, Performance, Design.

Keywords

Sensor Networks, Heterogeneous Sensors, Dynamic Scenarios, Self-adaptation, Reflective Middleware, Multi-agents Reasoning.

1. INTRODUCTION

Sensor network applications are becoming more complex due to the use of different kinds of mobile and sophisticated sensors, which provide advanced functionalities [1] and also are deployed in dynamic scenarios where context-awareness is needed [2]. To support those emerging applications, an adaptable underlying infrastructure is necessary. Current proposals suggest the use of a middleware, for example TinyDB [3]. However, this kind of state-of-the-art middleware have important non-negligible drawbacks that make them useless in the context of such new emerging applications, because: (i) the assumption that the network is composed only by a homogeneous set of basic or very constrained low-end sensors; (ii) the lack of intelligence in such network compromises the adaptability required to deal with changing operation conditions, e.g. lack of QoS management and control.

Adaptability is a major concern that must be addressed in sensor networks due to two main factors: (a) long usage life time; and (b) deployment in highly dynamic environments. The first reason increases the probability of changes in user requirements through systems life time, which requires flexibility in order to comply with the changing demands. The second reason implies that applications have to be flexible enough in order to cope with drastic changes in the operation scenarios. In such environments, services are required in different places at different times; resources must be reallocated in order to fulfill specific requirements and also assure compliance with different constraints; and nodes that satisfy specific constraints during a certain period of time can become unable to continue working properly after changes. In addition, there are real-time requirements that are especially hard to be met. Thus, QoS management must be flexible, allowing renegotiation of required/ provided QoS among nodes during the system runtime [4].

This paper presents a reflective middleware aimed to support sophisticated sensor network applications that need to adapt its behavior according to changes in the environment and in the application demands. The idea is that the users specify missions to be accomplished by the network using a high-level Mission Description Language (MDL) in which they describe the desired data and constraints related to the gathering of them, for example space and time limits, representing mission goals. In order to promote the missions accomplishment, the concept of multi-agents is used to provide the reasoning about the network and, among other things, to decide about service, resource allocation, time-related requirements and QoS control. The reasoning by the agents, i.e. the self-reflection of middleware agents, decides about what adaptations that must take place based on the mission goals. These adaptations are tuned through the use of aspects, which weave the desired behaviors into the middleware (e.g. jitter monitoring), and also through the movement of mobile-agents that change their location in the network in order to provide different services in different places as required in a context specific moment. In this paper, the focus is to present the mission-driven approach and the multi-agent reasoning based on this approach.

The remaining text is organized as follows: Section 2 presents the ideas of nodes' heterogeneity and dynamicity of operation conditions that motivate the present work. Section 3 provides an overview of the middleware structure, while Section 4 gives a summary description of the Mission Description Language. Section 5 presents the mission parameters representation. Section 6 presents the planning-agent intern model, while the multi-agents reasoning is described in Section 7. In Section 8 some related work are shortly presented, and Section 9 concludes the paper with some final remarks and directions of future work.

2. HETEROGENEOUS AND DYNAMIC

The intention of this work is to develop a flexible middleware that can be used to support applications in heterogeneous sensor networks deployed in dynamic environments. In the context of this work, heterogeneity means that nodes in the network may have different sensing capabilities, computation power, and communication abilities, running on different hardware and operating system platforms.

Low-end sensors are those with constrained capabilities, such as piezoelectric resistive tilt sensors, with limited processing support and communication resource capabilities. Rich sensors comprehend powerful devices like radar, visible light cameras or infrared sensors that are supported by moderate to high computing and communication resources. Thus, in order to deal with these very distinct capabilities, the proposed middleware must be lightweight, while being scalable and customizable. The mobility characteristic is also related to the heterogeneity addressed by the middleware. Sensor nodes can be static placed on the ground or can move themselves on the ground or fly at some height over the target area in which observed phenomenon is occurring. Figure 1 illustrates the heterogeneity dimensions considered in this work.

The proposed middleware is aimed to support applications that deal with dynamic and changing scenarios. Consequently, the set of sensors chosen in the beginning of a mission may not be the most adequate one during the whole mission. For example, an area surveillance system receives the mission to survey an area that may not allow traffic of certain kinds of vehicles. Ground sensors are set to trigger and send an alarm in the presence of unauthorized vehicles. Then Unmanned Aerial Vehicles (UAVs) equipped with visible-light cameras are set to fly over the area where the ground sensor has issued an alarm to verify the occurrence. However, a sudden change in the weather, e.g. the area becomes foggy or cloudy, turns visible-light cameras useless. However the detection mission must still be accomplished. This type of change in the operational conditions during a mission must be handled by the middleware. It must be able to choose the best alternative of employable sensors among the set of available options. In the described situation it may choose, for instance, an infrared camera instead of the visible-light one.

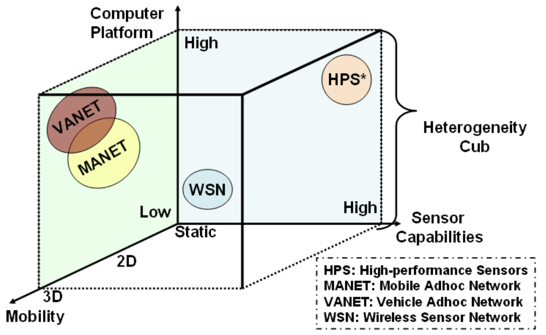
Computer Platform
High
HPS*
Heterogeneity Cub
VANET
MANET
WSN
Low
High
Sensor Capabilities
Static
2D
3D
Mobility
HPS: High-performance Sensors
MANET: Mobile Adhoc Network
VANET: Vehicle Adhoc Network
WSN: Wireless Sensor Network

Figure 1. Heterogeneity Dimensions

According to the taxonomy presented in [5], the sensor network described above can be classified as: a mix of static and dynamic configurable sensors with full self-awareness; a heterogeneous dynamic ad-hoc network with a large number of nodes; deployed in a high dynamic environment partially observable; and which achieve its goals by collectively coordinated actions with a non-local environment dependency. A sensor network with this classification requires a great flexibility in its behavior and at the same time "in-network intelligence", which is represented by the spread of intelligent capabilities over its nodes. These two features, flexibility and in-network intelligence, enables reflection about network status and environment conditions in order to adapt the network for the mission and to new demands from end-users.

3. MIDDLEWARE STRUCTURE

The main goal is that the proposed middleware fits both resource constrained and rich sensors. In order to achieve this goal, aspect- and component-oriented techniques are used in a way similar to the approaches discussed in [6], and [7] and the mobile and multi-agents approach presented in [8].

The proposed middleware is divided in three parts or layers, see also Figure 2:

Infrastructure Layer. It is responsible for the interaction with the underlying operating system and for the management of the sensor node resources, such as available communication and sensing capabilities, remaining energy, etc. This layer also helps to coordinate resource sharing according to application needs passed through the upper layers. Additionally, services provided by upper layers may also need some resource sharing support.

Common Services Layer. It provides services that are common to different kinds of applications, such as QoS negotiation and control, quality of data assurance, data compression, and the handling of real-time requirements. Other concerns such as deadline expiration alarms, timeouts for data transmissions, number of retries and delivery failure announcements, resource reservation negotiation among applications (based on priorities established by missions and operation conditions), bindings, synchronous and asynchronous concurrent requests are also handled within this layer. Readers specially interested in those concerns are referred to [9] for more details about the mechanism used in the middleware to provide these features.

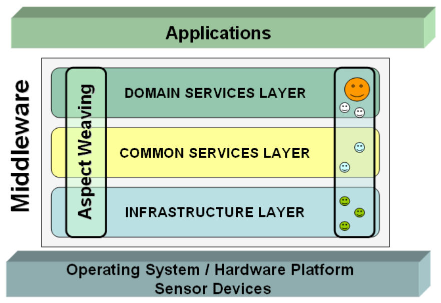

Figure 2. Overview of the Middleware Layers

Domain-Services Layer. It supports domain specific needs, such as data fusion and specific data semantic support, in order to allow the production of application-related information from raw data processing. Fuzzy classifiers, special kinds of mathematical filters (e.g. Kalman Filter) and functions that can be reused among different applications in the same domain are found in this layer.

Multiple applications performing different missions can run concurrently in the network. As stated before, the middleware handles resource and data sharing among applications, which need the same type of data, allowing a better energy management in resource constrained nodes. In powerful nodes, with more energy available, the middleware can provide more complex services aiming at the handling of rich data, such as those related to image processing, and pattern matching. This also means that such nodes can take some of the burden from low-end nodes.

"Smile faces" in the Figure 2 represent agents that can provide specific services in a certain node at a certain moment of system runtime. A special region (called *agents-space*) links agents throughout layers, allowing information exchange. The *Domain-Services Layer* hosts a special agent (bigger smiling face), which is responsible receiving the mission directions from the application layer, and based on that, plan and reason about the activities related to the sensing missions. This agent is called *planning-agent*, and details about it will be presented later in the following sections. The other agents (small smiling faces) are used to provide specific services that support applications.

Non-functional concerns that affect elements in more than one layer of the middleware, such as security, are represented as cross-layer features, which are addressed, at least partly, with the aspect-oriented approach presented in [6].

4. MISSION DESCRIPTION LANGUAGE

The Mission Description Language (MDL) provides means to describe the information requested or to be monitored about certain detectable phenomena in a given time-space domains interesting to the end-users. For instance, the user may want to know about the different kinds of vehicles that pass through a given area during a certain time, or the environmental conditions during the occurrence of a pre-defined event. By using the MDL to setup a mission to the network the user "tests" the environment in order to achieve the desired information about a phenomenon or an event of interest. The idea of "test" the environment is based on the C/ATLAS test language [10], in which high-level test commands are specified in order to retrieve information about devices in a system. In a similar way, the MDL provides high-level commands to retrieve specific information about matters or changes that occur in the environment. Based on this idea, the MDL uses patterns and definitions to test the environment in order to gather information that matches with those patterns and definitions that describe the user's need for information. However, it is important to highlight that in this proposal the MDL uses just this conceptual idea behind C/ATLAS, it does not use the terminology presented by the test language, neither the same taxonomy.

By using the MDL, the user defines high-level statements, which define and describe the events of his/her interest, as well as the constraints that are linked with that specific sensing mission. For instance, the maximum tolerated delay to receive an alert or the maximum amount of energy that can be used for that mission, among others. Another important concept in the MDL is the mission priority ordination, which allows several missions to run at the same time in the network, but prioritizing those which are more important, according to the user's definitions. Linked with the former idea and the constraints enforcement is the usage of policies to govern the performance of missions. The user can order the mission accomplishment according to their priorities, selecting some constraints and also link a policy that will dictate how persistent the nodes in the network will be in order to gather the requested information. For instance, in an aggressive policy, nodes can deplete their batteries in order to assure that the requested data will be delivered to the end user (i.e., by performing several retransmissions until the end-user receives the information or the battery is depleted). On the other hand, in a less aggressive policy, nodes may preserve their batteries in spite of that they cannot assure the data delivery. The user can also use a policy and "tune" it by means of specifying specific constraints that override the general policy for those specific parameters.

The mission described as a set of MDL statements is then translated to parameters that will configure the system as to retrieve the information desired by the user. The definition of these parameters is done by the interpretation of the MDL statements, together with the analyses of the characteristics of the deployed network and the chosen policy, if any, and the priority level. A configuration console (Mission Specification Console) enables the user to enter this information, which will be translated in a tuple of parameters (the content of this tuple will be explained in Section 5) representing the mission that will be injected into the network. Figure 3 illustrates the configuration console and its components.

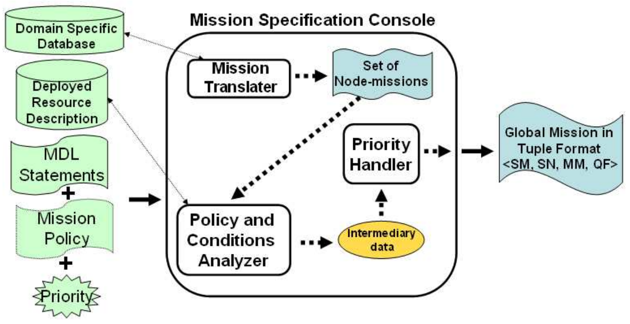
Domain Specific Database
Deployed Resource Description
MDL Statements
+
Mission Policy
+
Priority
Mission Specification Console
Mission Translater
Set of Node-missions
Priority Handler
Policy and Conditions Analyzer
Intermediary data
Global Mission in Tuple Format <SM, SN, MM, QF>

Figure 3. Mission Specification Console

The MDL statements are the essential elements used to define the overall mission (Global Mission) of the network. The Global Mission will be divided into node-missions (sub-missions), which will be executed by specific nodes or a group of nodes in the network. The subdivision of a mission in node-missions is an important part of the Global Mission translation in system parameters. It is done by a component called Mission Interpreter, which takes the MDL statements as input, consult a domain-specific database (for information about a particular domain), and translate these statements into node-missions, see Figure 4.

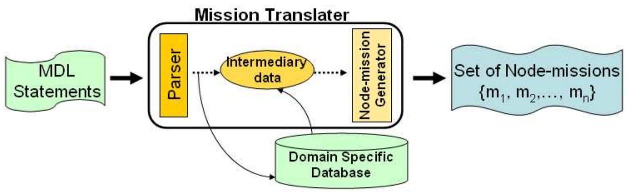

Figure 4. MDL Interpretation and Node-missions generation

The MDL is divided in two parts: (1) the kernel of the language, which defines events of interest, space and time parameters, as well as the priority of the sensing mission; and (2) the extensions, which define advanced parameters, such as accuracy, precision and constraints. It is composed by: imperative commands (i.e. SCAN and DETECT); keywords (i.e. pattern and object), which are the parameters of the commands; connectors to link commands and keywords (i.e. IF and WITH); and pre-defined patterns, which are in fact a kind of keyword that are stored in a domain library (i.e. FOG and LINEAR_MOVEMENT). As an example, the following represents a conditional MDL statement:

IF **DETECT** <DECREASE_OF <temperature>>
WITH GRANULARITY<3> **MONITOR** <FOG>
WITH ACQUISITION <period = yy>

This example statement means: if a 3ºC decrease in temperature is detected, then provide monitoring of the pre-defined pattern "FOG" with data acquisition by the sensors each "yy" time units.

5. MISSION PARAMETERIZATION

The input to the sensor network system, coordinated by the proposed middleware, is seen as a "mission" that the whole network must to accomplish in cooperation. By using a language such as MDL the user can specify necessary "data" for such a mission at a high level of abstraction. The user thus specifies the goals and priorities for the missions' directions in an MDL file, which after translation will drive the mission implementation based on reflection about the network, in order to accomplish with the users' requirements. The reflection consists of analysis and reasoning performed inside autonomous network nodes in order to allow the adaption required to face the changes in dynamic scenarios and users' requirements. Node-mission responsible agents, called **planning-agents**, reason about the network adaptations based on the mission directions and the network actual state. The former gives the requirements for data from users' point of view, while the latter is mainly characterized by nodes availability and environmental conditions. Their reasoning is made by a construction of believes about the network and its environment, which will help to achieve their goals and like this, comply with the network global mission.

The representation of the mission is provided by a set of goals that each planning-agent desires to achieve, supported by a set of "known facts" that they have about the network and the environment. Based on these facts and their goals, the planning-agents establish activity plans to achieve their goals, negotiating the best distribution of the work that must be done in order to accomplish with the global mission.

The network global mission is divided in sub-missions, called node-missions, which are assigned to planning-agents present in each individual node. Each node has just one planning-agent (placed in the *Domain Services Layer* of the middleware), thus for comprehension of the remaining text, a node-mission is assigned to a node or to the planning-agent installed on it. The accomplishment of each node-mission will corroborate for the success of the global-mission. In order to complete their node-missions, planning-agents break the node-mission into minor tasks that are related to the individual devices inside the node. A hierarchy among these concepts can be drawn: at the top is the global mission, followed by the node-missions, which is divided in several tasks in the node abstraction level. In the following, a formalization of these concepts is presented.

A **Global Mission** is represented by a tuple composed by the sets called SM and SN, and also the functions MM and QF. SM is a sub-set of all possible node-missions that could be assigned to a node; SN is a sub-set of all the nodes in the network; the mapping function MM maps elements of SM into elements of SN; while the quality function QF evaluates the mapping provided by MM. It is represented by:

$$GM = \langle SM, SN, MM, QF \rangle$$

M is the set of all possible node-missions;

SM is a set of node-missions (sub-set of ***M***) that can be assigned to the nodes members of the set ***SN***:

$$SM = \{m_i \mid m_i \in M\},\ i \in \{1,\dots,I\},$$

where ***I*** is the total number of all possible missions, i.e. the number of elements of set *M* or *SM*⊂*M*; Each **node-mission** m_i is represented by a tuple composed by a set of **measurements** that must be provided (***SME***), a set of **conditions** to the measurements (***SMC***), and a relation ***C*** that maps the set of conditions in the set of measurements:

$$m_i = \langle SME, SMC, C \rangle,$$

where ***SME*** is sub-set all possible measurements (***ME***); and ***SMC*** is the sub-set of all possible measurement conditions (***MC***),such as those related to periodicity, accuracy, time interval, range, among other. Such that SME ⊂ ME, and SMC ⊂ MC. ***C*** is the relation that maps conditions into measurements, where one measurement can be linked to none or several conditions. The opposite is also valid, i.e. one condition can be linked with none or several measurements:

$$C = \{r(mc_k) = me_j \mid mc_k \in SMC, me_j \in SME\},$$
$$k \in \{1,\dots,K\},\ j = \{1,\dots,J\},$$

where ***K*** is the total number of possible measurements conditions in the network (number of elements of the set ***MC***); and ***J*** is the total number of measurements in the network (number of elements of the set ***ME***).

N is a set of all nodes that compose the network.

SN is a sub-set of nodes in the network (a sub-set of N):

$$SN = \{n_v \mid n_v \in N\}, v \in \{1,\dots,V\},$$

where ***V*** is the total number of nodes in the network (the number of elements of the set N) or SN ⊂ N.

MM is the **mission-mapping function** that maps each node-mission to a certain node. A node in ***SN*** can perform one or more node-missions, but each node-mission is atomic (from the entire network point of view), i.e. it can be assigned to only one node:

$$MM = \{ f(m_i) = n_v \mid m_i \in SM, n_v \in SN \},\\ i \in \{1,...,I\}, v \in \{1,...,V\} \quad .$$

QF is a function that evaluates the mapping provided by ***MM***, given a grade between 0 and 10 for each par (m_i, n_i):

$$QF = \sum x_i$$

where $g(m_i, n_v) = x_i \mid m_i \in SM, n_v \in SN, x_i \in [0,10]$.

In order to achieve the goals of an assigned node-mission, a node must perform several different smaller tasks, called **node-tasks**. To read a value from the sensor device, or to turn a sensor device on/off are examples of **node-tasks**. At the node abstraction level, a specific **node-mission** is a sub-set of all **node-tasks** that a given node can perform, represented formally by:

$$nm = \{ t_w \mid t_w \in T \}, w \in \{1,...,W\}$$

where ***W*** is the total number of possible node-tasks that any node can perform (the number of elements of the set ***T***) or $nm \subset T$ where ***T*** is the set of all node-tasks that can be performed by any node.

6. PLANNING-AGENT MODEL

The proposed approach uses different kinds of agents; both cognitive and reactive ones, in order to perform different activities in the middleware, from the provisioning of simple services to complex reasoning about the network setup. To keep attention on the focus of this paper, only the model of the planning-agent, which is a cognitive agent, will be presented.

The model used in the present approach for the cognitive agents is based on the model of mental attitudes, known as BDI model (Believes-Desires-Intentions) presented in [11]. The BDI approach appears to suite well to the problem addressed by the current work, as some decisions that must be taken by the agents in the proposed approach require cognitive skills to "wonder" if certain actions are adequate to achieve a desired result, based on knowledge about conditions that may interfere on the performance of those actions. In the current problem formulation, what is desired is to obtain information by means of sensing activities, which are the goals of a sensing mission. Such knowledge is the "believe" that the node has about the relevant conditions and the intentions are translated into the actions need to retrieve the desired information. It thus seams that this model fits well to the goals of the proposed approach.

However, it is important to highlight that the approach used in this work is slightly different from the traditional BDI frameworks, such as [12] and [13], or more complex teamwork models, such as those presented in [14]. The major difference is that the model presented in this paper is focused on sensor networks activities, in which the network nodes do not perform any action that changes the world around them, what simplifies the model by eliminating the assumptions about this aspect. Besides, the proposal herein is simpler than those presented in the works mentioned above, as one can see in the remaining text.

The planning-agent has a complex "mental" activity, being responsible for different kinds of reasoning related to the mission accomplishment. It communicates with all other kinds of agents in the system. Besides, it negotiates with other planning-agents installed in the other nodes about the distribution of the node-missions. During these negotiations, it gathers information about the other nodes in order to achieve necessary knowledge about the network. It also has to maintain and update information about its own state, in order to inform other planning-agents and be capable to take right decisions. Environment conditions are also important in some of the deliberations taken by this agent. So this kind of information also constitutes its mental state, more precisely, as a part of its beliefs. In the following, a description about the beliefs, desires and intentions of the planning-agent, as well as a description of its plan and actions is provided.

Beliefs: Basically consisting of four groups of information: 1) background information, such as maps of the region; 2) the planning-agent's own conditions, translated in terms of the actions that it can perform and the node status (energy level, devices status, location, installed services, agents hosted in the node, etc); 3) other nodes status; 4) environment conditions.

Desires: The planning-agent has two types of desires: 1) General-Desires: which correspond to "built-in" goals, such as: distribute the node-missions in order to achieve the best overall result efficiently, and cooperate with other nodes; and 2) Specific Goals: which are related to the assumed node-missions and that come to its desires' set when it assumes the responsibility of a given node-mission ($\mathbf{m_i}$). These goals are ranked according to the related node-mission priority. It will be used to drive the construction of the plans that governs the execution of the agent's actions.

Intentions: Following the same idea of the desires, the planning-agent has two types of intentions: 1) General Intentions: which are directly related to the built-in goals, such as: to have an agreement about which that node will take the responsibility of a given node-mission after a negotiation with other nodes; to have provided the required resources to a requesting node; to have provided the correct information to other agents about data of interest; and 2) Specific Intentions: which specify intentions related to actions needed to accomplish a given node-mission, such as: to have sent the samplings with the correct accuracy within the timing constraints, which comes to the agent via the sets *SME* and *SMC* to the corresponding $\mathbf{m_i}$ that it assumes to accomplish.

Actions: Operating System or direct device drivers calls to perform commands on the underlying software and hardware platform; send and receive messages to and from other agents (request, inform, reply, notify, subscribe, publish, propose, reject, accept). Particularly in the negotiations occurring during the reasoning, the types of messages used are: inform, propose, accept or reject.

Plan: A plan is described in terms of a sequence of actions that an agent perform in order to achieve a sub-goal, decided by its deliberation and related to its intentions, and ultimately to achieve a motivational goal related to its desires. In order to accomplish with a given node-mission, the agent choose specific tasks ($\mathbf{t_w}$) such as they form a set that fulfill that node-mission. A plan will be a list of tasks that have to be done in order to accomplish with the node-missions allocated for that node. If a new node-mission is assumed and some of the tasks that are required to accomplish it are already in the plan, there is no need to insert them again in the plan, as their results are reused for this new node-mission. The planning-agent needs to construct a new plan, which can be totally new or at least a reviewed version of the current one, each time at least one of the following events occurs: it assumes a new node-mission; the conditions of the environment changes; a change in the network or in the user requirements occurs. This reflects the flexibility of the network to adapt itself according to the dynamicity of the network operation.

6.1 Architectural Structure

After the presentation of the cognitive planning-agents' internal model, its architectural structure can be described, based on the BDI architecture presented in [15], and is shown in Figure 5.

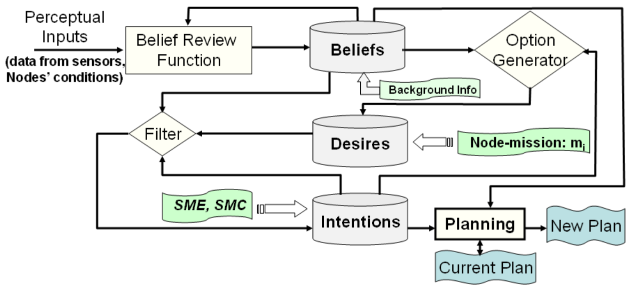

Figure 5. Planning-agent Internal Architectural Structure

In Figure 5 is shown that the agent takes the perceptual inputs (changes in the network or in the environment, data from its sensors, etc) and its current beliefs, and performs an update of their beliefs by means of the **Belief Renew Function**. After analysis of the updated beliefs and current intentions, the **Option Generator** function selects a sub-set of the desires representing the next possible goals to perform. Its beliefs, desires and intentions are then used as inputs to a **Filter** function that represents the deliberation of the agent and will provide the update of its intentions. The planning-agent constructs its own plans by reasoning about its current intentions and its beliefs. Ultimately the generated plans will fulfill with goals defined in its desire base, as the current intentions were decided based on the analysis of the set of the possible goals, by the use of the Option Generator. The result of the **Planning** is the selection of actions that are needed to perform the current intentions. It considers the current plan and beliefs, in order to select and order the execution of the actions. The current plan is used to reuse some previous decisions, and the current beliefs to evaluate which actions are more adequate to take in relation to the current conditions.

7. MULTI-AGENT REASONING

As stated above, the planning-agents construct believes that will guide their decisions based on the mission needs (the goals in their desires), which are characterized by node-missions. As the network receives a global-mission, the nodes will try to find a best fit to accomplish this mission, which characterizes the mission setup reasoning. Their decisions will influence the mission mapping function (***MM***), which may change in case of adaptation during the system runtime, due the adaptation reasoning. The mechanisms for the network setup and adaptation are described in the following.

7.1 Mission Setup

When a mission is received, the reasoning required to perform the network setup is divided in four steps, which are explained more in detail in the following.

Step One: each node performs an analysis of the elements of the set ***SM***, as well as its capabilities and the surrounding environment. If a node can provide the measurements of the set ***SME*** for a certain node-mission $\mathbf{m_i}$, satisfying the respective relation ***C***, it "declares" itself as "candidate" to perform $\mathbf{m_i}$. At this time, each node constructs a partial belief in relation to $\mathbf{m_i}$, only based on its own knowledge about the network, which is composed by the mission needs and its own capabilities.

Step Two: if a node considers itself as "candidate" to accomplish $\mathbf{m_i}$, it informs its "candidacy" to the other nodes, using an "inform message". However if the node does not consider itself as "candidate", it will just listen for the "candidacy" of other nodes.

Step Three: after a pre-established time-out, if no one considers itself as "candidate", no message will be exchanged. Thus, all nodes that can provide the data required by the measurements described in the set ***SME***, but that cannot satisfy the relation ***C***, communicate with the other nodes informing about the conditions that it can satisfy. The node, which provides the assurance closest to the desired one (specified by ***C***), takes the node-mission. This characterizes a best-effort way to solve the problem.

Step Four: nodes analyze their own conditions as well as conditions of the others, deciding which one must take $\mathbf{m_i}$. Such analysis uses the quality function ***QF*** and the node-tasks needed to perform $\mathbf{m_i}$. By maximizing $\mathbf{g(m_i,n_v)}$, nodes know which one ($\mathbf{n_v}$) will be in charge of the node-mission $\mathbf{m_i}$. If two nodes are capable to accomplish the node-mission, the one that has best conditions, e.g. remaining energy and/or other influencing parameters, takes the responsibility for that node-mission. In other words, the function **g** has a higher value for that node in comparison with the others. With this information, the nodes construct a common belief. If two nodes have the same value for the function **g** for the same node-mission, one of them is then randomly chosen.

Communication in wireless sensor networks can face problems that compromise message delivery. In case, any message of the coordination protocol is not received by any node in any of these steps, the node acts according to its belief from the last received message, if any. If it does not receive any message, it will act according to its initial partial belief. When the communication is reestablished, nodes will "listen" to the passing messages related to the same node-mission ($\mathbf{m_i}$) measurements, and then they will redo the above steps in order to achieve a new global-belief.

If compared with a centralized task distribution, the advantage in performing this reasoning in a distributed way is to avoid communication to and from the base station, which would consume more energy if compared with the local computation of the task allocation. As presented in [22], communication is the main source of energy consumption in sensor nodes, so communicating requires more energy than computation. In order to achieve a centralized task distribution that has the same quality as a distributed one can achieve; data about the current status of the nodes have to be sent often to the base station, which would increase the energy consumption. On the other hand, by the use of the distributed approach as presented, the nodes decide locally how to divide the new job, according to their status, without the need to send information through the network to the base station. This same argument holds to support the distributed way in which the adaptation is done, which is presented in the next sub section.

7.2 Mission Adaptation

During the system runtime, mission requirements and/or operational conditions can change. Nodes can perceive these changes, which induces node believes to be updated. If a change makes a node unable to proceed in the mission accomplishment, the network must adapt itself to solve the problem. The reasoning performed by the planning-agents will try to find another node that can perform node-mission $\mathbf{m_i}$ in the place of the previous node. This reasoning is similar to the one presented above, but there are two different circumstances that also must be taken into account: (1) the node simply fails; (2) the node continues to work, but is aware that it cannot continue performing the mission.

Considering the first case, faulty nodes are perceived by other healthy nodes, which have participated with the faulty one in the initial mission establishment reasoning. As nodes can perceive that the node responsible by the node-mission $\mathbf{m_i}$ is not responding during an established time-out period, they redo the reasoning to decide which one must perform $\mathbf{m_i}$. Information about the failure is added to the belief of these healthy nodes.

In the second case, the node that becomes unable to accomplish $\mathbf{m_i}$ informs this situation to the nodes that participated in the mission establishment reasoning. Further they decide which one will take the node-mission previously assigned to that node.

Another situation that requires adaptation is when changes make other nodes (more) capable to accomplish a certain node-mission $\mathbf{m_i}$. An adaptation can be triggered if the node-mission was previously assigned in a best effort way, as explained in the step three in section 3.2. The need for a best service can also trigger the adaptation, foreseeing a possible increase in the users' requirements priority. The mechanism of these changes is implemented by an exchange of "proposal" and "accept" or "reject" messages.

Adaptations decisions are also based on the quality function ***QF***. It is done during the establishment of the best mapping of node-missions to nodes as explained in the section 3.2. The target is always to maximize the value of ***QF***, what can be achieved by maximizing the function $\boldsymbol{g(m_i,n_v)}$ for each node-mission $\mathbf{m_i}$: i.e. $\boldsymbol{max(g(m_i,n_v))}$.

As result of the procedure explained above, all nodes know which node $\mathbf{n_v}$ has taken the responsibility for the node-mission $\mathbf{m_i}$, (similarly to the establishment of the node-missions mapping). Therefore, nodes' beliefs are updated with this knowledge. In the same way as explained before, if two nodes have an equal value of function **g** for a given node-mission, one of them is chosen randomly.

7.3 Considerations about Complexity

As the middleware is intended to run in a variety of nodes, from resource constrained nodes to resource rich ones, the mechanisms presented above have to be customized for each type of node.

Considering resource rich nodes, function **g** used to evaluate the quality of a given mapping may be an elaborated and computing intensive algorithm. However, when it comes to the low-end nodes, simpler functions may take place in order to perform the evaluation of the part of the mission related to them. The same way that there is a tradeoff between energy consumption and communication resource usage, there is also a tradeoff in the amount of resources that should be used and the quality of a solution provided by the used algorithm. It is possible that an optimal solution is not achieved by a simpler algorithm, but considering the resource constraints, a sub-optimal can be better than an optimal one that depletes the available resources.

The same kind of variation is present in the internal components of the planning-agent that inhabit different types of sensor nodes. The above explained Option Generator Function, Filter Function provides for that the Planning can be much richer, considering much more parameters and having more complex algorithms in the resource rich nodes, if compared with the same functions in the low-end nodes. However, it is important to highlight that every node of a given kind use the same set of functions, so the coherence is maintained.

8. RELATED WORKS

Agilla [16] is one of the precursors in the use of mobile agents in middleware for WSN. Its approach is to use agents that can move from one node to another in the network. It also allows multiple agents to run in the same node. These characteristics provide the desired features of energy saving, as the agents can run near to the data avoiding unnecessary communication. In comparison with the proposed approach, the use of agents is not restricted to moving and using services around the network but also to help in the network reflection and decision for adaptability using multi-agents.

In [17] a proposal to use a distributed mechanism to control adaptive sampling to support energy-constrained network operations is presented. In the proposal, each sensor is considered an autonomous agent, enabling decentralized control of the sampling rate of sensor nodes in the application domain of flood monitoring. Besides the contribution in the increase the efficiency of the energy consumption, the goal of this approach is also to maximize the information value of the data collected to the base-station. The major differences between our work and the one mentioned above are respectively: the consideration of a heterogeneous sensor network instead of a homogenous one; the domain independent instead of a domain-specific approach; and the direct cooperation among the agents instead of just the decentralization of the problem.

AWARE [18] is a project that proposes a middleware whose goal is to provide integration of the information gathered by different type of sensors, including WSN and mobile robots. Our proposal aims also at addressing heterogeneous sensors, but also concerns like QoS, as presented in [9], and runtime reflection to address changes in the environment and in the network. Moreover, our approach provides the capability of autonomy to the network nodes, by using an agent-orient approach. In the referred middleware, the nodes do not have the same capability.

In [19] an approach that uses Artificial Intelligence to configure an underlying middleware is presented. This approach uses the concepts of missions and goals to plan the allocation of tasks in a network of homogeneous nodes. The handling of heterogeneous nodes is one of the differences between the referred work and the one presented in this paper. Additionally, in that work, the intelligence is outside the middleware by means of just sending "commands" or adjusting its parameters. In our presented approach, agents make part of the middleware, spreading intelligence over the network.

In [20] an information processing paradigm for intelligent sensor networks is presented. Nodes in sensor networks have different levels of autonomy in terms of the signal processing, information fusion and situation assessment in order to contribute with the overall system decision making. This approach is based on the use of a genetic algorithm to provide learning features to the sensor nodes, and fuzzy cognitive maps to perform situation assessment. The sensor networks aimed by this work are those composed only by rich nodes, as the architecture and the techniques used are quite heavy to fit in low-end nodes. On the other hand, the proposal of the present paper is to address heterogeneous sensor networks that are composed by both low-end and rich nodes, allowing them cooperate in order to achieve the overall mission goals.

9. CONCLUSION AND FUTURE WORK

This paper presented the concepts of a middleware needed to address mission-driven heterogeneous sensor networks deployed in highly dynamic scenarios. These scenarios require middleware reflection to support adaptations to face constant changing conditions during runtime. Multi-agents reasoning is used in order to setup, configure and reconfigure the network. Besides, a formal definition of the mission statements and conditions (described in MDL) was presented, as well as its mapping to elements of a BDI approach that supports the proposed network wide reasoning.

The direction of the ongoing and future work includes enrichment of the Mission Description Language specification, adding abstractions that can help the user specify missions. Another ongoing work is the implementation of the simulation to provide results that validate the presented ideas. In order to do that, the adaptation of a simulator for wireless networks called Shox [21] is being done. This adaptation consists of the inclusion of the agents concepts in the simulator framework, and an interface with the Mission Specification Console.

10. ACKNOWLEDGMENTS

E. P. Freitas thanks the Brazilian Army for the given grant to follow the PhD program in Embedded Real-time Systems in Halmstad University in cooperation with UFRGS in Brazil.

11. REFERENCES

[1] Culler, D., Estrin, D. and Srivastava, M. Overview of sensor networks. *IEEE Computer*, vol. 37, no. 8, pp. 41–49, 2004.

[2] Henricksen K. and Indulska, J. A software engineering framework for context-aware pervasive computing. In *Proceedings of PerCom*, pages 77–86. IEEE Computer Society, March 2004.

[3] Madden, S., Franklin, M. J., Hellerstein, J. M. and Hong, W. TinyDB: An acquisitional query processing system for sensor networks. ACM *Transactions on Database Systems*, 30(1):122–173, 2005.

[4] Liberatore, V. Implementation challenges in real-time middleware for distributed autonomous systems. In *Proceedings of Second IEEE SMC-IT*, 2006.

[5] Vinyals, M., Rodríguez-Aguilar, J.A. and Cerquides, J. A Survey on Sensor Networks from a Multi-Agent perspective. In *Proceedings of 2nd International Workshop on Agent Technology for Sensor Networks (ATSN-08)*, 2008.

[6] Freitas, E. P., Wehrmeister, M. A., Pereira, C. E., Wagner, F. R., Silva Jr., E. T., Carvalho, F. C. DERAF: A High-Level Aspects Framework for Distributed Embedded Real-Time Systems Design. In *Proceedings of 10th International Workshop on Early Aspects*, Springer, 2007, pp. 55-74.

[7] Tesanovic, A. et al, Aspects and Components in Real-Time System Development: Towards Reconfigurable and Reusable Software. *Journal of Embedded Computing*, IOS Press, v.1, n.1, 2005.

[8] Freitas, E. P., Wehrmeister, M. A., Pereira, C. E. and Larsson, T. Reflective middleware for heterogeneous sensor networks. In *Proceedings of 7th Workshop on Adaptive and Reflective Middleware (ARM'08)*, ACM. 2008. pp. 49-50.

[9] Freitas, E. P., Wehrmeister, M. A., Pereira, C. E. and Larsson, T. Real-time support in adaptable middleware for heterogeneous sensor networks. In *Proceedings of International Workshop on Real Time Software (RTS'08)*, IEEE. 2008. pp. 593-600.

[10] IEEE Std 716-1995, 1995. IEEE standard test language for all systems-Common/Abbreviated Test Language for All Systems (C/ATLAS), IEEE, Inc.

[11] Bratman, M. E. *Intention, Plans, and Practical Reason*. Cambridge, MA, 1987.

[12] Cohen, P. R. and Levesque, H. J. Teamwork. *Nous*, 25(4):487–512, 1991.

[13] Grosz, B. and Kraus, S. Collaborative plans for complex group actions. *AIJ*, 86:269–358, 1996.

[14] Pynadath, D. V. and Tambe, M. Multiagent teamwork: analyzing the optimality and complexity of key theories and models. *Proceedings of 1st International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS-02)*. ACM. 2002. pp. 873-880.

[15] Weiss, G. *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence*. The MIT Press, 1999.

[16] Fok, C.-L., Roman, G.-C. and Lu, C. Rapid development and flexible deployment of adaptive wireless sensor network applications. *Proceedings of the 24th ICDCS'05*, 2005.

[17] Kho, J., Rogers, A. and Jennings, N. R. Decentralised Adaptive Sampling of Wireless Sensor Networks. *Proceedings of 1st International Workshop on Agent Technology for Sensor Networks (ATSN-07)*, 2007.

[18] Gil P. et al. Data centric middleware for the integration of wireless sensor networks and mobile robots. In *Proceedings of 7th ROBOTICA'07*. 2007.

[19] Schmidt D. C. et al. A Decision-Theoretic Planner with Dynamic Component Reconfiguration for Distributed Real-Time Applications. *Proceedings of 8th ISADS'07*. 2007. pp.461-472.

[20] Leung, H., Chandana, S. and Wei, S. Distributed sensing based on intelligent sensor networks. *IEEE Circuits and Systems Magazine*, 8(2). pp. 38-52, 2008.

[21] Lessmann, J., Heimfarth T. and Janacik, P. ShoX: An Easy to Use Simulation Platform for Wireless Networks. In *Proceedings of Tenth International Conference on Computer Modeling and Simulation*, 2008. pp. 410-415.

[22] Akyildiz, I. F., Weilian S., Sankarasubramaniam, Y., Cayirci, E. A survey on sensor networks. *IEEE Communications Magazine*, 40(8). pp. 102-114, 2002.