Self-Healing Wireless Sensor Networks

Markus Lanthaler, *Department of Computer Science, University of Helsinki*

***Abstract*—Nowadays wireless sensor networks have found their way into a wide variety of applications and systems with vastly varying requirements and characteristics, but all of them have a common element: faults are a normal fact and not isolated events as in traditional networks. Thus, in order to guarantee the network quality of service, it is essential for the sensor network to be able to detect and heal failures. In this work a failure detection scheme and a service management approach using the autonomic computing paradigm and some concepts of the IT Infrastructure Library (ITIL) will be evaluated. The presented approach aims to employ self-healing services, allowing them to discover, examine, diagnose and react to malfunctions.**

***Index Terms*—Autonomic computing, fault tolerance, network architecture and design, network management, self-healing, wireless sensor networks**

I. Introduction

THE number of transistors on a cost effective chip and, therefore, the processing or storage capacity of that chip, doubles every year, following Moore's law [1]. A continuation of Moore's law until 2019 will result in transistor features just a few atoms in width and will make the vision of *smart dust* [2] to become reality.

Mark Weiser stated already in 1991 that "In the 21st century the technology revolution will move into the everyday, the small and the invisible…" and "The most profound technologies are those that disappear. They weave themselves into the fabric of everyday life until they are indistinguishable from it."

Today we are at the beginning of that *ubiquitous computing* era. The design of micropower wireless sensor systems has already gained increasing importance for a variety of civil and military applications. Recent advances in microelectro-mechanical systems (MEMS) technology and its associated interfaces, signal processing, and wireless communications, have enabled the development of low-cost, low-power, multifunctional sensor nodes that are small in size and communicate in short distances. These tiny sensor nodes, which consist of sensing, data processing, and communicating components, leverage the idea of sensor networks based on collaborative effort of a large number of nodes and so the focus has shifted away from limited macrosensors communicating with base stations.

While individual microsensor nodes are not as accurate as their macrosensor counterparts, the networking of a large number of nodes enables high quality sensing networks with the additional advantages of easy deployment and fault-tolerance. These characteristics make microsensors ideal for deployment in otherwise inaccessible environments, where maintenance would be inconvenient or impossible and represent a significant improvement over traditional sensors, which are deployed in the following two ways [3]:

- Sensors can be positioned far from the actual phenomenon, i.e., something known by sense perception. In this approach, large sensors that use some complex techniques to distinguish the targets from environmental noise are required.
- Several sensors that perform only sensing can be deployed. The positions of the sensors and communications topology are carefully engineered. They transmit time series of the sensed phenomenon to the central nodes where computations are performed and data are fused.

A common sensor network is composed of a large number of sensor nodes (in the majority of cases, these networks are composed of hundreds of thousands of nodes), which are densely deployed either inside the phenomenon or very close to it.

The position of sensor nodes need not be engineered or pre-determined. This allows random deployment in inaccessible terrains or disaster relief operations. On the other hand, this also means that sensor network protocols and algorithms must possess self-organizing capabilities because deploying and maintaining the nodes must remain inexpensive – manually configuring large networks of small devices is impractical.

The nodes are able to collect, process, disseminate and store data. They perceive the environment, monitor different parameters and collect data according to the application purpose. Another unique feature of sensor networks is the cooperative effort of sensor nodes. Sensor nodes are fitted with an on-board processor. Instead of sending the raw data to the nodes responsible for the fusion, sensor nodes use their processing abilities to locally carry out simple computations and transmit only the required and partially processed data. The reason for this is that computation is much cheaper than communication in regard to the most critical resource, the energy. Each transferred bit costs as much energy as about 1,000 instructions [4], thus, wireless sensor networks process data within the network wherever possible.

A wireless sensor network has applications in environmental and habitat monitoring, precision agriculture, indoor climate control, surveillance, treaty verification, intelligent alarms, and medical diagnostics. The most dramatic applications involve monitoring complex interactions, including wildlife habitats, disaster management, emergency response, ubiquitous computing environments, asset tracking, healthcare, and manufacturing process flow.

Markus Lanthaler was with Graz University of Technology, 8010 Graz, Austria. He is now with the Department of Computer Science, University of Helsinki, 00014 Helsinki, Finland
(e-mail: markus.lanthaler@helsinki.fi).

Some of these applications require a large number of devices making traditional methods of sensor networking impractical due to the high demand on cable installations.

To be able to manage wireless sensor networks in an efficient manner, Assunção *et al.* [5] proposed the use of the IT Infrastructure Library (ITIL) [6] and the autonomic computing paradigm [7].

This work is organized as follow. Section II presents wireless sensor networks that manage themselves without direct human intervention and presents the needed main management services to implement an autonomic wireless sensor network. Section III covers the important aspects of failure detection and fault management with the focus on event-driven networks. Section IV presents the definition of an *Autonomic Service Management System*, developed based on the autonomic computing and ITIL concepts by Assunção *et al.* [5]. In section 0 the results of experiments conducted by Assunção *et al.* [5] and Ruiz *et al.* [8] will be presented and finally, the concluding remarks are presented in section VI.

II. Autonomic Wireless Sensor Networks

In the majority of cases the network elements, called sensor nodes, of a wireless sensor network are deployed in remote areas where maintenance and administration by technicians are impracticable. The form factor of a single sensor node can vary, depending on the actual need of the application, from the size of a shoe box (e.g. a weather station) to a microscopically small particle (e.g. for military applications where sensor nodes should be almost invisible). Similarly, the cost of a single device may vary from hundreds of Euros (for networks of very few, but powerful nodes) to a few cents (for large-scale networks made up of very simple nodes) [9]. Each device is composed by a computational unit, a wireless communication unit, a sensing unit (one or more sensors), a logic unit (software) and a power unit. Recharging or replacing the battery is generally impracticable, since there are thousands of nodes in potential inaccessible environments. Depending on the application, the required lifetime of a sensor network may range from some hours to several years and has a high impact on the required degree of energy efficiency and robustness of the nodes [9].

Sensor nodes observe the environment, monitor different parameters and collect data according to the application purpose. In some applications, the network must collect, process, and deliver the data continuously and in real-time, in other types of applications the data is delivered to the observer only when a certain event occurs. Any missing or late information can influence the interpretation of the data and therefore a failure in sensing, processing, or delivery can disturb the network goals.

The design and development of energy efficient systems in environments that impose severe restrictions is not a trivial task. Considering the given characteristics, the system should be as autonomic as possible, that is, the wireless sensor network should manage itself with the least or no human intervention.

An autonomic system is composed of interrelated autonomic elements. Each of these elements has managed hardware or software resources that build the IT infrastructure and autonomic managers that supervise and control these resources. The autonomic manager provides self-management services using monitoring, planning, analyzing and executing modules. Fig. 1 presents the interaction between the autonomic elements.

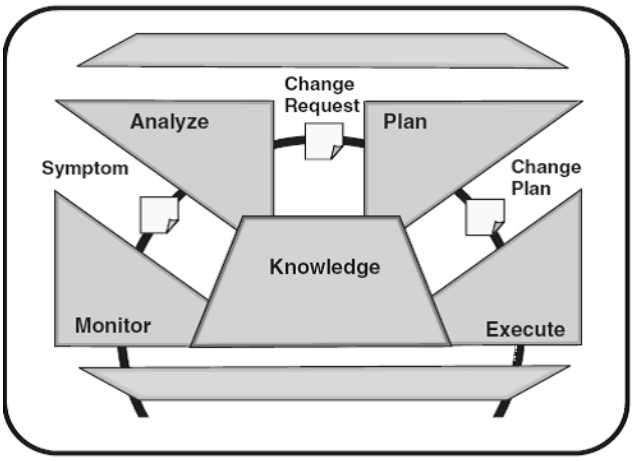

Fig. 1. Interaction between autonomic elements [7]

Regarding to autonomic wireless sensor networks, the management tasks should consider some of the following aspects ([7], [10]):

- Self-healing: discover, diagnose, and react to network disruptions. Self-healing components can detect system malfunctions or failures and start corrective actions based on defined policies to recover the network or a node. The automatic recovering from damages improves the service availability.
- Self-optimization: monitor and tune resources automatically. The management service that maximizes the resource allocation and utilization, and guarantees optimal service quality, based on policies. The tuning actions could mean reallocating resources – such as in response to dynamically changing workloads – to improve overall utilization, or ensuring that particular business transactions can be completed in a timely fashion. The automation of complex tasks and the components adjustment in response to variable workloads allows the delivery of a high-level service.
- Self-configuration: change configuration parameters to adapt dynamically under varying conditions and network states. This management service self-configures and reconfigures the network elements under varying and even unpredictable conditions. The network configuration must occur automatically, as well as dynamic adjusts to the current configuration to best handle changes in the environment.
- Self-protection: anticipate, detect, identify and protect against threats (internal or external, accidental or malicious) from anywhere. In case an attack happens, this service executes detection routines in order to reach security.
- Self-service: allow the provision of sensing, processing and dissemination services, anticipating resources and at the same time keeping the complexity hidden, in order to shrink the gap between business application

and service goals.

- Self-awareness: allow the entity to know its environment and its activities context and act accordingly. It finds and generates rules to best interact with neighbor entities.
- Self-knowledge: the management service that qualifies an entity to know itself. For example, an entity that governs itself should know its components, current state, capacity, and all the connections with other entities. It needs to know the extension of its resources that can be lent and borrowed.
- Self-maintain: allow an entity to monitor its components and fine-tune itself to achieve pre-determined goals.

According to the characteristics described above, four common functions should be implemented in the autonomic sensor nodes: a function to collect the details it needs to know from the system, a function to analyze those retrieved details to determine if something is wrong and needs to change, a function to create a plan, or sequence of actions, that specifies the necessary changes, and a function to perform those actions. These functions work together to provide the control loop functionality of an autonomic manager [7]:

Monitor: The monitor function provides the mechanisms that collect, aggregate, filter and report details that can be analyzed and collected from a managed resource. The details can include topology information, metrics, configuration, status, capacity, and throughput.

Analyze: The analyze function provides mechanisms to correlate, observe, and analyze complex situations in an effort to determine whether changes need to be implemented. This function can model complex behaviors to use prediction techniques allowing the autonomic managers to learn about the IT environment and predict future behaviors.

Plan: The plan function creates or selects a procedure to execute a desired change in the managed resource. A change plan, which represents a desired set of changes for the managed resource, is created and passed to the execute function. The planning mechanism uses policy information to guide its work.

Execute: The execute function provides the mechanisms that controls the execution of a plan with considerations for dynamic updates. This function is responsible to execute the change plan and to update the knowledge used by the autonomic manager.

These four parts communicate and collaborate with one another and exchange appropriate knowledge and data, as shown in Fig. 2.

According to the work of Assunção *et al.* [5], some of the IT Infrastructure Library (ITIL) concepts are used to employ self-management services: for this purpose the monitor, analyze, plan and execute functions are implemented for autonomic managers defined under the ITIL paradigm.

III. Failure Detection and Fault Management

Sensor nodes have strong hardware and software restrictions in terms of processing power, memory capability, power supply, and communication throughput. The power supply is the most critical restriction, given that it is typically not rechargeable. For this reason faults are likely to occur frequently and will not be isolated events. Besides, large-scale deployment of cheap individual nodes means that node failures from fabrication defects will not be uncommon.

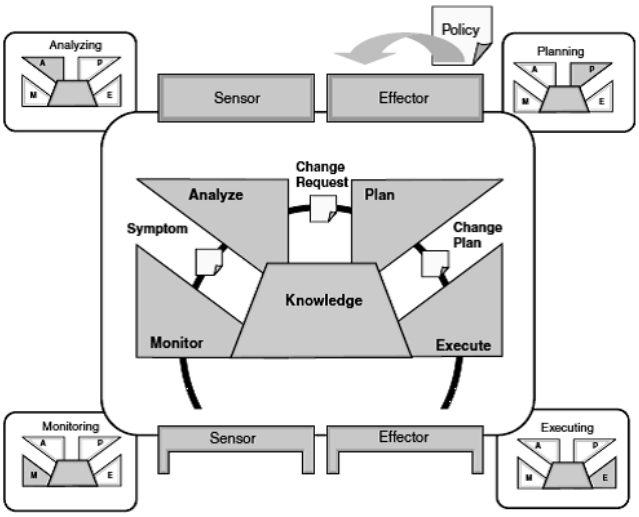

Fig. 2. Functional details of the autonomic manager [7]

In military applications, where these networks are deployed in open spaces or enemy territories, adversaries can manipulate the environment (so as disrupt communication, for example by jamming), but have also physical access to the nodes. At the same time, ad-hoc wireless communication by radio frequency means that adversaries can easily put themselves in the networks and disrupt infrastructure functions (such as routing) that are performed by the individual nodes themselves. Finally, the sensor nodes are exposed to natural phenomena like rain, fire, or even falls of trees since they are commonly used to monitor external environments. Therefore failure detection and fault management plays a crucial role in wireless sensor networks. If, in addition to detecting a failure, the management application can also determine the reasons of the failure and distinguish between malicious and non-malicious origins, it can trigger security management services or, if it is an accidental or natural failure, activate "backup nodes".

In applications interested in the conditions of the environment at all times sensor nodes will be programmed to sense and send back their measurements at regular intervals or continuously. These networks are called *programmed* and *continuous*, respectively. Other applications just need data when some "special" events occur; the networks are then called *event-driven* networks. On the other hand, when the network is able to answer to queries of the observers, it has the *on demand* property.

Configuring the network as event-driven is an attractive option for a large class of applications since it typically sends far fewer messages. This results into a significant energy saving, since message transmissions are much more energy-intensive when compared to sensing and processing. For instance, if the application is temperature monitoring, it could be possible just to report data when the temperature of the area being monitored goes above or below certain thresholds.

The drawback of such event-driven networks is, that failure detection is much harder to implement than in continuous networks since the continuous data stream can also be used as an indicative of the network operation quality. If data are received from every single node, then all is well. In event-driven networks however, if the management application stops receiving data from certain nodes or entire regions of the network, it cannot distinguish if a failure has occurred or if no application event has occurred.

Ruiz *et al.* [8] use a homogeneous hierarchical network. The nodes are grouped into clusters and there is a special node called cluster-head which has more resources and, thus, is more powerful than the common-nodes. Furthermore, cluster-heads are responsible for sending data to a base station. The base station communicates with the observer, which is a network entity or a final user that wants to have information about data collected from the sensor nodes. In their implementation, the management agents execute in the cluster-heads where aggregation of management and application data is performed. This mechanism decreases the information flow and energy consumption as well. A manager is located externally to the sensor network where it has a global vision of the network and can perform complex tasks that would not be possible inside the network.

The failure detection in such a sensor network can be done in the following way:

In the installation phase, that occurs as soon as the nodes are deployed in the network, each node finds out its position in the area and reports it and its energy level to the agent located in the cluster-head, which aggregates it and send it to the manager. With this data the management application builds a topology map model and an energy map.

In the operational phase, while the sensor nodes are performing their functions, i.e., collecting and sending data, management activities take place. Among them, energy level monitoring plays a central role. Each node checks its energy level and sends a message to the agent whenever there is a state change. This information is also transmitted to the manager, which can then recalculate the energy and topology maps, as well as the coverage area, which characterizes the coverage area maintenance service. Also, operations can be sent to the agents in order to execute the failure detection management service. The manager sends *GET* operations in order to retrieve the node state. The *GET-RESPONSEs* are used to build the network audit map. If an agent or a node does not answer to a *GET* operation, the manager consults the energy map to verify if it has residual energy. If so, the manager detects a failure and sends a notification to the observer.

IV. Service Management System for Self-healing

To extend this model of failure detection and fault management, Assunção *et al.* proposed in their work [5] the usage of some of the IT Infrastructure Library (ITIL) concepts and proposed a system whose architecture is described as follows:

An autonomic manager, located outside the network, is responsible to map the Service Level Agreement (SLA) into so called "policies" for the network nodes, to monitor the service quality and availability and, if necessary, to renegotiate the SLAs. These policies are delivered to the network nodes and stored in repositories, the knowledge bases. The autonomic managers, located in the sensor nodes, then start the monitor, analyze, plan and execute functions, based on this information.

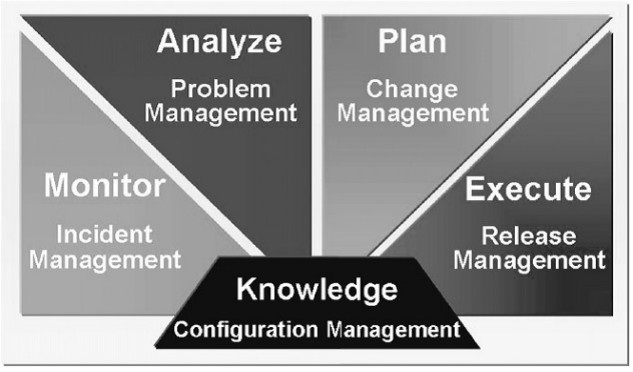

Fig. 3. Autonomic element [5]

Autonomic managers on cluster-head nodes guarantee that the service level is being attended inside the cluster and adjust the network components to attend these levels.

Common nodes' autonomic managers are responsible to monitor their resources, optimize the nodes' functioning, detect anomalous behavior, analyze events and adjust the nodes' configuration in order to diminish the risk of faults. In case any problem occur, these managers will recover the network operation as fast as possible.

Some concepts of the ITIL Service Delivery area were employed by Assunção *et al.* [5] in the definition of four autonomic managers with the purpose of creating a self-healing wireless sensor networks, namely:

- Autonomic Service Level Manager: the autonomic manager that guarantees the fulfillment of agreed service levels and eventually redefines the SLA using a manual manager or policies.
- Availability Autonomic Manager: the autonomic manager that plans and manages service availability through the monitoring of the IT service availability.
- Continuity Autonomic Manager: the autonomic manager that analyzes network risks identifying possible failures and creating a recover or risk reduction plan.
- Capacity Autonomic Manager: the autonomic manager that monitors nodes resources and identifies demands. In case of current or future insufficient capacity this manager is responsible to reallocate resources and anticipate new resources, what makes necessary the definition of a resource utilization model to determine whether the nodes are attending the defined requirements or not.

Each one of these managers employs concepts of Service Support disciplines to accomplish the monitor, analyze, plan and execute function, considering the self-healing service (see Fig. 3).

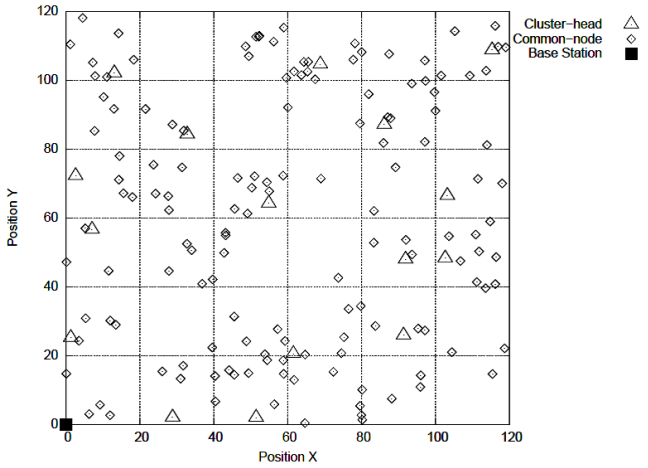

Fig. 4. Hierarchical network comprised of common-nodes, cluster-heads and a base-station. Common nodes are less powerful than cluster-heads and take part of the group that has as the leader the nearest cluster-head. Communication among nodes is single hop [7].

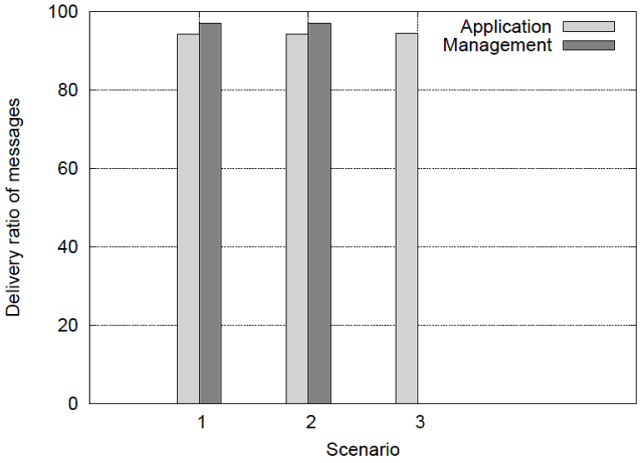

Fig. 5. Delivery rate of messages [7]

V. Experiments

The experiments presented here, are divided in two separate sections and cover two common problems in wireless sensor networks. In the first part, which covers the experiments from [8], an unexpected event puts the nodes, confined in a predefined region, out of operation, whereas the second part, which covers the experiments from [5], deals with problems caused by traffic congestion and energy loss. The aim of the experiments in the first part is to evaluate the impact of management functions over the wireless sensor network, analyzing the management costs and identify the effectiveness of the management architecture in detecting failures. The experiments in the second part are used to investigate the impact of service management regarding to power consumption and data flow.

In all experiments the implemented application accomplishes temperature monitoring which is simulated using the Network Simulator 2 (NS-2) tool and the MannaSim, which is a framework made of a set of base classes that extends NS-2 to simulate sensor networks.

A. Management costs and effectiveness in detecting failures

In this section, dealing with an event-driven network, the simulation model is defined as follows:

The nodes sense the temperature continuously along the time, but they send their data only when the minimum or maximum value collected differs 2% from the last data sent. In order to simulate the temperature behavior of the environment, random numbers were generated following a normal distribution, taking into consideration standard deviation of 1 °C from an average temperature of 25 °C. Fig. 4 presents the node distribution in the monitored area (120 m × 120 m) and Table I describes the networks parameters and the features of the nodes.

1) Evaluating Management Impact

In this section the impact of management functions over the wireless sensor network are evaluated according to an analysis of its costs. For this experiment three different scenarios were simulated. In scenario 1 and scenario 2 the network is simulated with all the fault management functions, but in scenario 2 the failure detection function was removed, finally in scenario 3 the network is simulated without any management functions. In the simulation, an unexpected event causes the failure of 32 nodes located at the center of the network at 45 s of simulation (100 s simulation time).

TABLE I
SIMULATION PARAMETERS

Parameter	Value
No. of nodes	160 (144 common-nodes and 16 cluster-heads)
Cluster size	Variable
Simulation time	100 s
Coverage area	120 m × 120 m
Environment conditions	Variations in the environment and noise are not considered
Initial energy available in each node	5 Joules
Network type	Heterogeneous
MAC Protocol	IEEE 802.11
Routing Algorithm	None
Propagation Model	Shadowing
Node distribution	Uniform random
Transmission power of common-nodes	9.9 mW (90% of receiving rate at a distance of 15 m)
Transmission power of cluster-heads	281.8 mW(90% of receiving rate at a distance of 80 m)
Sensing range	2 m
Node capacity	5 buffers for receiving packets
Energy spent in communication	0.66 W for node transmission, 6.0 W for cluster-head transmission, and 0.2 W for reception
Energy spent in sensing	10 mW
Energy spent in processing	Not considered
Node mobility	Stationary

Fig. 5 shows the delivery rate, which measures the ratio of messages received by the nodes in the network to the messages sent by the nodes, for sensing application and management messages. As expected, the delivery rate is

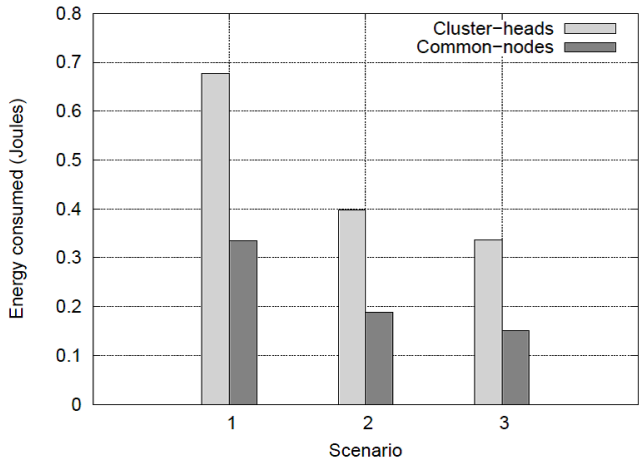

Fig. 6. Energy consumption of nodes [7]

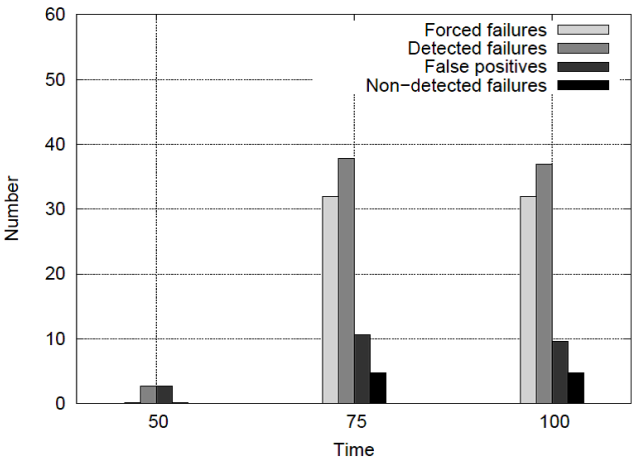

Fig. 7. Detection effectiveness for centered failures (scenario 1) [7]

very similar in all scenarios, since the messages are transmitted in the same wireless environment and to and from the same nodes. Neither the introduction of detection nor the introduction of management had any influence on the delivery rate of application messages.

Fig. 6 shows the energy consumption of common-nodes and cluster-heads. It is observed that, as far as detection is not concerned, the energy consumption increased with management in 18% for cluster-heads and 29.45% for nodes. But when the detection mechanism was taken into account, management caused an increase of 101.2% and 129.45% in the energy consumption for cluster-heads and nodes, respectively.

This result was expected since the act of transmitting and receiving messages are the most determinant activities for energy consumption according to the simulated energy model.

2) *Failure Detection Effectiveness*

This second set of experiments was conducted in order to evaluate if it is worth the increase in traffic and energy consumption.

In this simulation the region, where the ruin of nodes occurred, was modified in terms of location and also in terms of dimension. Table II presents the description of the simulated scenarios.

For these experiments an event which harms the nodes at 45 s, putting them out of operation, was simulated. The manager was programmed to start the detection mechanism at times 25, 50, and 75 s and to report the results at times 50, 75, and 100 s, respectively. So, when the unexpected event occurs, there was enough time (20 s) for the manager to come to a conclusion regarding the availability of the nodes. This means that only the reports in 75 and 100 s would have to contain any conclusion regarding this event. Thus, the report at 50 s shows the results obtained before the event occurrence.

Fig. 7 shows the effectiveness of the detection mechanism for scenario 1. The numbers in the x-axis represent the points in time when the manager reports the availability of the nodes in the network. It can be observed that there were some failure detections in time 50 s, although at this time the destruction of the nodes could not yet be perceived. Drops of *GETs* (messages to get the current state of a node) or *GET-RESPONSEs* cause the manager to be misled and, consequently, produce false positives. This problem also occurs at points 75 and 100 for the same reason, representing 27.93% and 26.06% of the detections, respectively. The quantity of false positives at these points is considerably higher than the quantity for point 50 due to the harm of some cluster-heads where the agents run. What happens is that after the unexpected event occurs, some common-nodes, which were not harmed, lost their cluster-heads if they are located inside the damaged region. As a consequence, these common-nodes stop receiving the *GETs* from the manager, since they are sent to them through the agents. As a result, the manager does not receive answers from these common-nodes provoking false positives. In respect to scenario 1, the number of "orphan" nodes was 8.

For scenario 5, the results are similar, considering that almost twice as many nodes are harmed in this scenario.

TABLE II
DESCRIPTION OF THE SIMULATED SCENARIOS

Scenario	Description
1	32 nodes (20% of the network, composed of 3 cluster-heads and 29 common-nodes) located at the center of the network are harmed. These nodes have x and y coordinates between 30 and 90.
2	41 nodes (25.63% of the network, composed of 4 cluster-heads and 37 common-nodes) located near the base station are harmed. These nodes have x and y coordinates between 0 and 60.
3	39 nodes (24.37% of the network, composed of 4 cluster-heads and 35 common-nodes) located far from the base station are harmed. These nodes have x and y coordinates between 60 and 120.
4	14 nodes (8.75% of the network, composed of 1 cluster-head and 13 common-nodes) located at the center of the network are harmed. These nodes have x and y coordinates between 40 and 80.
5	62 nodes (38.75% of the network, composed of 6 cluster-heads and 56 common-nodes) located at the center of the network are harmed. These nodes have x and y coordinates between 20 and 100.

Fig. 8 shows the results for scenario 2. The results for point 50 are almost the same as the results for the centered region (scenario 1). As mentioned before, at that point the unexpected event had not yet been perceived, meaning that

the results seem to be independent from the region chosen. However, as far as points 75 and 100 are concerned, it is possible to observe considerable dissimilarities. The number of false positives has decreased to 10.26% (point 75) and 10.36% (point 100) of the detections. The reason is that in this experiment the number of orphan nodes is only 4, i.e., two times less than the number of orphan nodes for scenario 1.

Fig. 8 also shows the results for non-detected failures. Comparing with the results for scenario 1, the amount of non-detections is similar, representing 15.59% of the failures. This shows that the number of initial messages drops in the center is similar to the region near the base station.

The results for the scenarios 3 and 4 are very similar to these results.

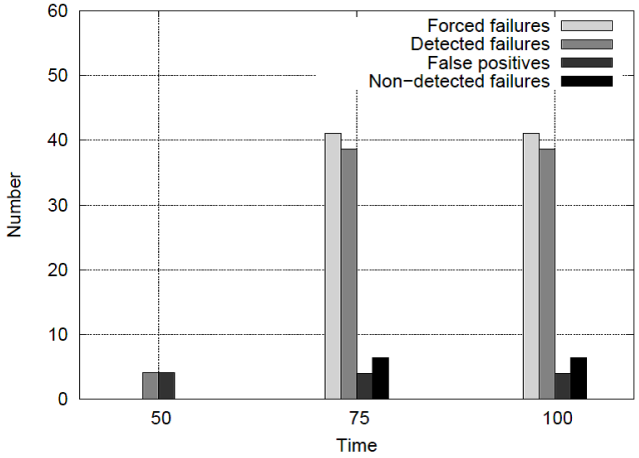

Fig. 8. Detection effectiveness for failures near the base station (scenario 2) [7]

B. *Impact of Service Management*

The second part of experiments [5] evaluates the impact of the usage of service management (based on the autonomic computing and ITIL concepts) to heal the network. The simulation will deal with communication problems due to traffic congestion and energy loss and the system has to detect the root cause of the problem and react in an adequate way. In the following the functionality of the simulated system is described.

The knowledge bases are distributed between the network nodes and all nodes store information about their own state. Cluster-head nodes store also network management information, since these nodes have more communication and storage capacity.

The main tasks defined to the autonomic managers located in common nodes are described below.

Monitor: The continuity manager detects message loss incidents. In order to detect if messages were lost, the nodes periodically check if they received an *ACK* message for each sensed data message sent. The capacity manager is responsible to monitor the sensor node residual energy. When the energy is under a certain threshold, an incident is detected and the node starts sending only high priority messages. This manager is also able to detect abnormal production increase incidents, which occurs when some event of interest is sensed making nodes to increase their production.

Analyze: The error diagnose is accomplished using information stored in the sensor nodes' knowledge sources. In case common nodes lose messages, the autonomic manager will diagnose if an abnormal increase in nodes' production is the root cause of the incident (this increase makes nodes lose their messages since the packet queue is full).

Plan: After the autonomic manager detects that messages were lost and the production has increased, it will try to decrease the node's production, until messages are no longer lost. In order to control their production the manager proposes as change the gradual increase of the sensing and sending interval. In case of a low energy problem the manager will propose to stop the node activities for some seconds.

Execute: The manager alters the sensing and sending intervals or puts the node out of service for a few seconds. While messages are being lost, the manager keeps adjusting these parameters and evaluating their impact over the network.

The main tasks defined to the autonomic managers located in cluster-head nodes are described below.

Monitor: The detection of message loss is accomplished by the continuity manager using the same *ACK* mechanism as the common nodes. The capacity manager is responsible to monitor the sensor node residual energy. Cluster-head node managers, also detect the network increase of production incident and low energy level in the cluster.

Analyze: In case cluster-head nodes lose messages, these nodes analyze the knowledge source in order to diagnose if the failure is an abnormal production increase.

Plan: If the autonomic manager detected a low residual energy incident in its cluster, it proposes as change that the 20% of the cluster nodes with the smallest residual energy will stay out of service for 5 s. A similar change is performed when the network has an abnormal production increase (which is characterized by a significant raise in the number bytes sent). The cluster-head node chooses 20% of the cluster nodes with the highest production and put these nodes out of service for 5 s. These plans aim to increase the network lifetime and prevent congestion.

Execute: The autonomic manager executes the change plan putting the chosen nodes out of service for a 5 s, and then evaluates the impact of the change over the network.

The knowledge source is updated whenever messages are received or configuration parameters are changed. The cluster-head nodes store in the knowledge source their local information and some replied information of each node of the cluster. Each message received by the cluster-head from an unknown node will create an entry for this node in the knowledge source. After that, this repository will be updated by messages sent every second from the common nodes with information about their repository entries. The cluster-head knowledge source entries update the received message rate when a sensed data message is received.

The simulation was built with dimensions of 50 m × 50 m containing four cluster-head nodes and five common nodes per cluster. The cluster-head nodes were positioned in a grid and the common nodes deployed randomly. A base station was positioned in the center of the scenario. Using this

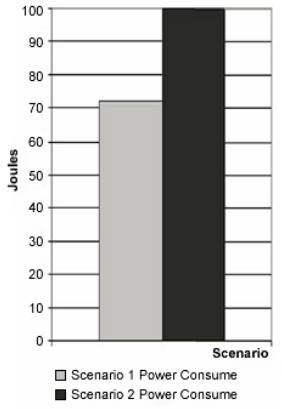

Fig. 9. Common nodes power consumption [5]

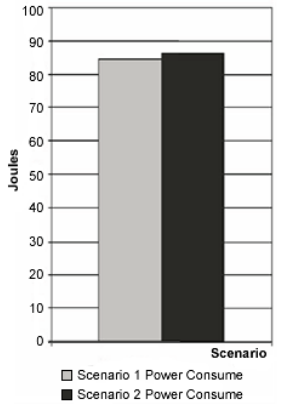

Fig. 10. Cluster-head nodes power consumption [5]

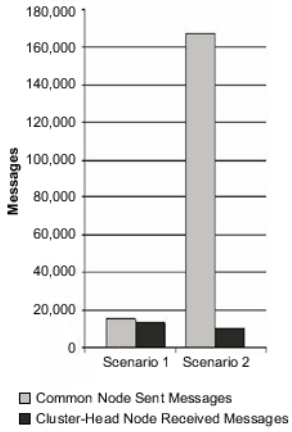

Fig. 11. Common nodes sent and cluster-head received messages [5]

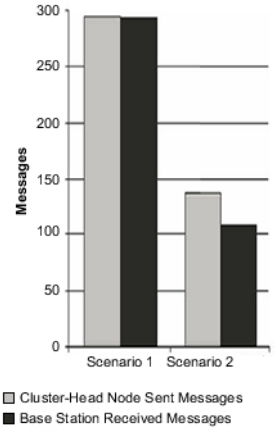

Fig. 12. Cluster-head nodes sent and base point received messages [5]

topology, two scenarios were simulated. In scenario 1 the service management system for self-healing wireless sensor networks described above was implemented and in scenario 2 the network does not implement any self-management functionalities.

The characterization of the simulation is presented in Table III. The specifications of the two kinds of sensor nodes utilized in the simulations (common and cluster-head nodes) were configured according to the ones presented by the real nodes Mica Motes [11] and WINS [12], respectively. Each scenario was executed 33 times and the results, presented in the following sections, refer to the average of the obtained values.

The modeled average temperature is, like in the previous experiments, 25 °C, but the standard deviation in these simulations is 5 °C. When the sensed temperature exceeds 28 °C, the message that contains this data is considered a high priority message.

In order to simulate communication problems, at each 20 s, common nodes decrease their sensing and disseminating interval from 0.01 and 1 s to 0.001 and 0.01 s, promoting an increase in network data production. As a consequence to that, messages are lost because of space loss at the packet queue, that admit 10 messages in common nodes and 100 messages in cluster-head nodes. The network nodes try to detect incidents at every 1 s interval.

1) Impact of Service Management in Power Consumption

In almost any application of wireless sensor networks energy is the most critical resource. In case of node failure due to energy exhaustion, the network production is reduced in an irreversible manner.

Sensor nodes from scenario 2 exhaust their energy after 75 s of simulation, supporting the communication problem event for a 25 s period. On the other hand, the common nodes of scenario 1, which implement the self-management system described above, survive during the entire simulation and support the communication problem for 60 s. According to these data, the power consumption of scenario 1 (see Fig. 9) is smaller than the power consumption in scenario 2, even though the network keeps functioning for 80 s longer than in scenario 2. This happened because the nodes detect that messages are being lost and diminish their production, and the fact, that common nodes send only high priority information in case of low energy level.

The cluster-head nodes from both scenarios presented similar power consumption (see Fig. 10). Although the cluster-head nodes from scenario 1 delivered more information, the size of the messages was smaller, making the power consumption comparable to the ones of scenario 2.

TABLE III
CHARACTERIZATION OF PERFORMED SIMULATIONS

Parameter	Value
No. of nodes	24 (20 common-nodes and 4 cluster-heads)
Cluster size	5 nodes
Simulation time	155 s
Number of simulations	33
Coverage area	50 m × 50 m
Environment conditions	Variations in the environment and noise are not considered
Initial energy available in each node	5 Joules in common nodes and 100 J in cluster-head nodes
Network type	Heterogeneous
MAC Protocol	IEEE 802.11
Energy spent in transmission (reception)	36 mW (24 mW) for common nodes and 600 mW (300 mW) for cluster-head nodes
Transmission range	40 m for common nodes and 250 m for cluster-head nodes
Processing consumption	24 mW in common nodes and 360 mW in cluster-head nodes
Node capacity	Space for 10 messages in common nodes and 100 messages in cluster-head nodes
Energy spent in sensing	15 mW
Sensing and disseminate type	Programmed
Node mobility	Stationary

2) Sensed Data Flow

The amount of data sent by the common nodes of scenario 1 is 10.80 times greater than the one of the scenario 2, however, the amount of data received by the cluster-head nodes from scenario 1 is 1.3 times greater than the one of scenario 2 (see Fig. 11). It means that the

scenario 1 saves energy, since it delivers fewer messages when network is congested and therefore drops less data. Nevertheless, the number of messages delivered to the cluster-head node is bigger if compared to the one of scenario 2. This demonstrates that the number of dropped messages in the network was reduced.

The amount of data sent by the cluster head nodes to the base station in scenario 1 is 2.16 times greater than in scenario 2 (see Fig. 12). This happens because the cluster-head nodes decide to decrease their dissemination intervals, being able to deliver the received messages by the common nodes of the cluster. Nevertheless, the scenario 1 presented only 0.40% of message loss, while the scenario 2 presented 21.07% of message loss.

The amount of dropped messages is in scenario 2 is 84 times greater than the one of scenario 1. That is obvious because it is not implemented any way of detecting or solving communication problems in scenario 2.

VI. Conclusions

Building and deploying networks, especially in environments where there will be tens of thousands of network elements with particular features, is very complex. The task becomes even worse due to the physical restrictions of the sensor nodes, in particular energy and bandwidth restrictions and in some applications even the environment is inaccessible. Nevertheless it is possible to operate a wireless sensor network in such circumstances. The autonomic computing approach is one of the possible solutions, because it helps to keep the network independent of human interventions.

As the results show the detection of improper operations and components failure in wireless sensor networks and the automatic recovering of problems promote a greater availability of the service and the network longevity. Analyzing all the obtained results, the proposed solution has proved to be efficient in correcting communication problems and prolonging the network lifetime even if – at least in an event-driven network – there will be an overhead, that is acceptable for mission-critical applications.

References

[1] Gordon E. More, "Cramming More Components onto Integrated Circuits" *Electronics*, volume 38, number 8, April 1965.

[2] J. Kahn, R.H. Katz, and K. Pister, "Emerging Challenges: Mobile Networking for 'Smart Dust,'" *J. Comm. Networks*, pp. 188-196, Sept. 2000.

[3] C. Intanagonwiwat, R. Govindan, D. Estrin, "Directed diffusion: a scalable and robust communication paradigm for sensor networks", Proceedings of the *ACM Mobi-Com'00*, Boston, MA, 2000, pp. 56–67.

[4] Bernhard Rinner, "Context-Aware Computing", course at Graz University of Technology, 8010 Graz, Austria, 2006. Available: http://www.iti.tugraz.at/en/teaching/echtzeit_ki/index.html

[5] Helen P. Assunção, Linnyer B. Ruiz, and Antônio A. Loureiro, *A Service Management Approach for Self-healing Wireless Sensor Networks*, LNCS 4195, pp. 215-228, 2006,

[6] *Planning to Implement Service Management Manual. The Stationary Office,* Office of Government Commerce, 2002.

[7] IBM (2005), *Autonomic computing white paper – An architectural blueprint for autonomic computing,* IBM Corp., 2005.

[8] Linnyer Beatrys Ruiz, Isabela G. Siqueira, Leonardo B. e Oliveira, Hao Chi Wong, José Marcos S. Nogueira, Antonio A. F. Loureiro, "Fault Management in Event-Driven Wireless Sensor Networks", Proceedings of the 7th ACM international symposium on Modeling, analysis and simulation of wireless and mobile systems MSWiM'04, Venice, Italy, 2004, pp. 149-156.

[9] Kay Römer and Friedemann Mattern, "The Design Space of Wireless Sensor Networks", *IEEE Wireless Communications Magazine*, volume 11, issue 6, pp. 54-61, Dec. 2004.

[10] Linnyer Beatrys Ruiz, Jose Marcos Nogueira, and Antonio A. F. Loureiro, "MANNA: A Management Architecture for Wireless Sensor Networks", *IEEE Wireless Communications Magazine*, volume 41, issue 2, pp. 116-125, Feb. 2003.

[11] *MICA Wireless Measurement System*, Crossbow Technology Inc., 2003. Available at http://www.xbow.com/

[12] WINS (2002). *Wireless Integrated Network Sensors (WINS),* Department of Electrical Engineering, UCLA, Los Angeles, CA, USA. Available: http://www.janet.ucla.edu/WINS/