An Application-Based Real-Time Scheduler for Wireless Sensor Networks

Leo Ordinez, David Donari, Rodrigo Santos, and Javier Orozco

Instituto de Investigaciones en Ingeniería Eléctrica
Universidad Nacional del Sur - CONICET
Av. Alem 1253 - (8000) Bahía Blanca
Buenos Aires - Argentina
{lordinez,ddonari,ierms,jorozco}@uns.edu.ar

Abstract. Wireless sensor networks have turned into a solution to tackle monitoring in difficult access places. In this paper, an algorithm to perform real-time information retrieval from nodes is presented. The algorithm is proposed at the application level. An early forest fire detection system is presented as case study.

1 Introduction

Wireless Sensor Networks (WSN) are becoming more and more used today because they provide an economic way of performing surveillance on difficult access places or remote locations where the presence of other technologies are prohibitively expensive. This kind of networks rely on small System-On-a-Chip (SoC) devices that sense some variables and transmit their values over an *ad-hoc* network protocol. Then, a sink node collects all that information and updates the situation in a central computer in charge of making decisions. By correlating sensor output of multiple nodes, the WSN as a whole can provide a functionality that a single node cannot.

A WSN can also be seen as a distributed client/server system where the nodes are servers and the application running in the sink contains the clients. Thus, a standard interface that sits between the application above and communication software and operating system below (*i.e.*, a middleware) is needed by the heterogeneous hardware and software platforms involved. In this paper, the Data Distribution Service for Real-Time Systems (DDS) [1] from the Object Management Group will be used to this purpose.

New devices used in WSN allow the logging of information on web sites providing a real-time monitoring for anyone that can access the data. However, it is not enough to have the information available if there is no pre-processing of it in order to detect the potentially dangerous spots. Consequently, the way in which the application requests information from the nodes is very significant to the overall performance of the system. Moreover, the great amount of nodes deployed in a field determines the need for a mechanism that offers a sound way of polling those nodes. In the case of an early forest fire detection system, weather

XXXIV Conferencia Latinoamericana de Informática (CLEI 2008), pp. 1239-1248, 2008.

information from environmental sensors is used to calculate a series of indexes that establish the danger of fire. This application will be taken, through the rest of the paper, as a clear example, about distinction of importance among sensors to be inquired and of dealing with timing constraints.

To sum up, the main contribution of this paper is the proposal of an algorithm to provide real-time information retrieval from sensors in a WSN at the application level. The algorithm provides an intelligent retrieval from nodes with a potentially developing incident by scheduling more frequently the risky ones. In this sense, the Behavioral Importance Dual-Priority Server (BIDS) [2] provides an interesting framework to decide who, when and which should be polled for data more frequently. In order to exemplify the design of a system based on BIDS, an early forest fire detection system is presented as case study.

The rest of the paper is organized as follows: in Section 2, previous work related to WSN and information retrieval (mainly in the field of forest fire detection) is analyzed and compared to the proposed algorithm. In Section 3, the system model including the usual fire indexes and their qualities for early detection are analyzed. In Section 4, a detailed architecture of the system with tasks and relationships is developed to illustrate the mechanism. Finally, in Section 5, conclusions are drawn.

2 Related Work

In recent years, WSNs have been increasingly attracting research and development efforts specially oriented to the environmental monitoring. The fact that they provide a simple and economical way to reach areas where a wired network would be infeasible is one of its main advantages.

Real-time forest fire detection by means of WSNs has been treated in the literature before [3], [4] and [5]. However, in those papers the accent is made on the deployment and configuration of the nodes into clusters. In [3], the network behaves in three possible modes: time-driven, event-driven and query-driven (a detailed explanation of these protocols can be found in [6]). Basically each node can produce a regular report periodically, an emergency report upon a special event and finally may answer to the query of the master. Once the information has been collected, a neural network processes the retrieved data and produces an output indicating the probability of a developing fire in the area. In [5], a careful analysis of the causes producing forest fires is done and the Fire Weather Index [7] is described. The paper models the problem with a k-cluster algorithm optimization. Basically, the best configuration based on the deployment and activity of the nodes is obtained. Finally, in [4] a similar approach to that of [5] is presented. The authors used a different index to measure the probability of fire. In this case, the nodes operate with a flat query protocol without distinction of priorities.

This paper differs from the previous ones in the fact that a new query algorithm for the nodes is proposed. This algorithm distinguishes between risky (important) and non-risky (not important) spots. The way in which the cluster-

ing and deployment of the nodes is done, is left as an open issue and any of the mentioned proposals can be applied.

3 System Model

In this section, several aspects that influence the design of a WSN application based on BIDS are modelled. The theoretical and practical bases and assumptions of forest fire modeling, WSN architecture and real-time scheduling are presented.

3.1 Forest Fire Modeling

The beginning of forest fire is subjected to the presence of an ignition source. Once that ignition source is present, several environmental factors determine the occurrence of fire. Vegetation characteristics, weather state and topography are fundamental aspects in order to predict the behavior of fire.

Different systems have been developed to evaluate the main factors that affect occurrence, behavior and consequences of fire. In particular, the Canadian Fire Weather Index (FWI) [7] represents the work of several decades to build a complete system of indexes to determine the risk of forest fire. The FWI takes into account ignition probability, fire behavioral characteristics in case it develops, difficulties to control it and the damages it could cause. This system is being ported and adapted to the Argentinian geographical characteristics (see [8], [9]).

The FWI is composed of six modules, each of which indicates the contribution of a specific aspect to the characteristics of fire. This is a complex system that incorporates relationships between weather variables, fuel states and behavior of fire to generate indicators that provide a quantitative measure of the difficulties to control and the potential damage that fire could cause.

The architecture of the FWI is shown in Figure 1. At a first level, the system measures the moisture content of three different fuel classes using weather observations. Here, three primary codes are calculated: the Fire Fuel Moisture Code (FFMC), the Duff Moisture Code (DMC) and the Drought Code (DC). These codes, in turn, represent the moisture content of litter and fine fuels, the moisture content of loosely compacted decomposing organic matter and the moisture content of the deep layer of compacted organic matter.

In a second level, the Initial Spread Index (ISI) and the Build Up Index (BUI) are computed. The ISI index indicates the rate of fire spread immediately after ignition. It combines the FFMC and wind speed to predict the expected rate of fire spread. The BUI index is a weighted combination of the DMC and DC codes, and it indicates the total amount of fuel available for combustion.

Finally, the Fire Weather Index (FWI) is calculated from the ISI and BUI to provide an estimate of the intensity of a spreading fire. In fact, FWI indicates fire intensity by combining the rate of fire spread with the amount of fuel being consumed. Fire intensity is defined as the energy output measured in kilowatts per meter of flame length at the head of a fire. The head of a fire is the portion of a fire edge showing the greatest rate of spread and fire intensity.

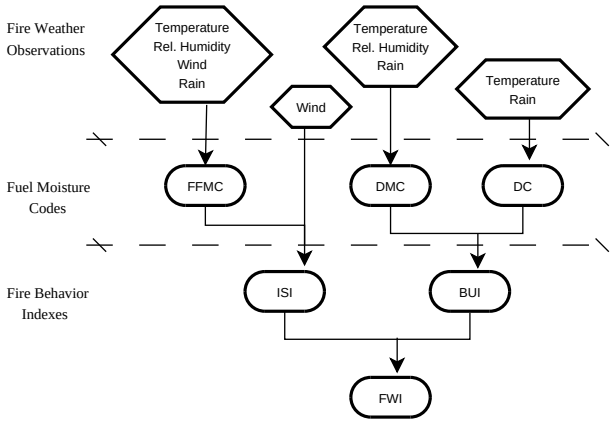

Fig. 1. The Canadian Fire Weather Index Model

3.2 Wireless Sensor Network Modeling

A WSN is a cooperating system composed of sensing nodes with a limited intelligence and a centralized sink that makes decisions based on the information reported by the nodes [10]. In this paper, the WSN for environmental monitoring considered is assumed to be so densely deployed that they often have redundant information.

Usual network protocols for WSN are based on IEEE 802.15.4 [11]. Some derivations are ZigBee [12] and Wibree [13]. They defined a set of high-level communication protocols aimed at providing long battery life and secure networking. The way in which the nodes in the WSN communicate between each other and how they are organized in clusters are beyond the scope of this work.

A sketch of the proposed WSN architecture is depicted in Figure 2. A large number of nodes are uniformly distributed across the observing field. Nodes are organized into clusters through a clustering protocol and the communication between the cluster head and the sink is done by a multihop network protocol. In the figure, three kinds of sensor nodes can be distinguished: sensors that measure temperature and relative humidity; sensors that measure accumulated rain and wind speed sensors. The sink is in charge of polling the different sensors, collect the data and respond accordingly.

In general, the chosen sensors depend on the type of application. In the case of WSN for forest fire detection based on the FWI system, the variables needed to be measured were defined in Section 3.1. Among the available sensors for this purpose [14], [15], [16], [17], [18] and [19] are the ones that best fit the requirements. It is worth pointing out, that generally temperature and relative humidity sensors are integrated in the same device.

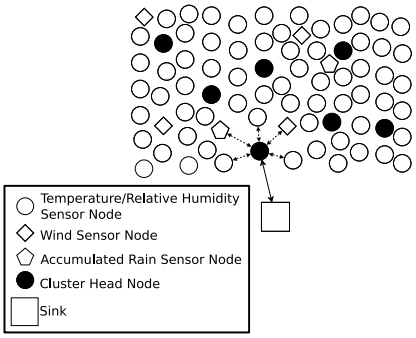

Fig. 2. Wireless sensor network organization.

3.3 Real-Time Model

The BIDS approach falls in the paradigm of Resource Reservation Mechanisms [20]. It is based on servers that adjusts the bandwidth of each server according to the last instance behavior of its encapsulated task. Thus, servers having important tasks get more bandwidth and execute more frequently than those that have not important tasks. In what follows, the real-time aspects of a system based on BIDS will be presented (a deeper explanation can be found in [2]).

An application is composed of several tasks. These tasks are assumed to be independent, periodic and preemptible. They may be hard or soft. The hard ones will be scheduled according to the Earliest Deadline First [21] scheduling policy; while soft ones will be scheduled in a hierarchical form through the use of BIDS. A BIDS server S is a software abstraction capable of encapsulating one or more tasks and is described by a tuple (Q_s,P_s,D_s,α_s) for its budget, period, relative deadline and postponement factor, respectively. Tasks running on a server are said to run on a virtual processor whose speed is $\frac{Q_s}{P_s}$ times the actual speed of the processor (the quotient $\frac{Q_s}{P_s}$ is named the bandwidth U_s of a BIDS). In addition, servers provide temporal isolation among tasks running on different servers, which means that misbehavior in one of them does not affect the rest.

Since tasks are periodic, they can be seen as a stream of jobs or instances J_{ij} where the first subindex refers to the task and the second one to the instance. Each task is characterized by its worst-case execution time, C_i, its period, T_i and its relative deadline D_i. Tasks have a behavior parameter μ_i associated that will be further explained later. Each job within the task has a release or activation time defined by $a_{ij} = a_{i(j-1)} + \gamma_i T_i$, where the value of γ_i can be either 1 or another natural number and its election corresponds to the election of α_s from the server. The last timing constraint is the absolute deadline given by $d_{ij} = a_{ij} + D_i$. In addition, each job has a parameter δ_{ij} that is related to the behavior of the information processing in the instance, its description will be made later.

To take account of the behavioral aspect of a task, a parameter μ_i, named the *threshold*, is introduced. The μ_i threshold is established by the system de-

veloper at design time and it is based on the expected values of the information processing behavior of the task. If the result obtained exceeds the threshold, the task is said to be IMPORTANT and it is associated, at least for one instance, to the set of $IMPORTANT$ tasks. On the contrary, when the result is below that threshold the task is NOT IMPORTANT and, analogously to the previous case, it will belong to the $NOT\,IMPORTANT$ set. It is worth mentioning, that the classification of a job is done once the execution is completed. Consequently, the behavior of a job will be given by the following mapping:

$$\mathcal{M} : \mathbb{R}^2 \rightarrow \{IMPORTANT,\ NOT\ IMPORTANT\}$$

$$\mathcal{M}(\delta_{i,j}, \mu_i) \in \begin{cases} IMPORTANT & \text{if } \delta_{i,j-1} \geq \mu_i \\ NOT\,IMPORTANT & \text{if } \delta_{i,j-1} < \mu_i \end{cases} \quad (1)$$

The behavior of a task is reflected in the way the server encapsulating it updates its parameters. When a job is described as NOT IMPORTANT, the bandwidth of the server is reduced accordingly to the α_s value. That is, the period of the server holding the task is adjusted by α_s. The α_s parameter is based on the dynamics of the controlled variable of the task. Thus, the selection of the proper α_s should be based on the dynamics of the system. It is worth mentioning, that α_s can be a function of the variables involved in the process.

4 System Design

Software design for WSN, in most cases, is an *ad-hoc* job. In [22] two issues to be faced when developing applications are exposed: 1) a lack of proper abstractions, and 2) a lack of coherent tool-chains. In this sense, typically a two-layer model is used: an application layer and an operating system (OS) layer. With this model, adaptations and flexible extensions are hard to fulfill, since either OS used tend to be designed specially for the application, or an unnecessary overhead in the application is performed. Furthermore, there are cases where the layers are indistinguishable and both application and OS are designed as a single module that executes directly on the hardware. Nevertheless, recent researches are trying to revert this fact by proposing more flexible designs (even to a middleware level [10]). In this section, a layer-based approach will be proposed, pointing out the distributed nature of software components according to the hardware in which they run (*i.e.*, nodes or sink). Besides, the proposed architecture will be exemplified with an early forest fire detection system.

As was previously mentioned, this paper proposes an intelligent approach to retrieve information from the nodes with a potentially developing incident. In addition, the application domain of this kind of systems includes timing constraints that must be taken into account at the moment of designing the system. This last topic involves a real-time scheduling of the different tasks that compose an application. The timeliness of the application domain appears in WSN systems because they process real world data. This involves having in mind physical time as a fundamental variable within the system.

On the other side, the kind of system tackled in this paper is inherently modular and supported by different hardware platforms. To be able to deal with diverse hardware and OSs a set of tools that provide a uniform access medium to system resources is required. An alternative to reach an efficient distributed interface is the middleware layer Data Distribution Service for Real-Time Systems (DDS) [1]. The DDS is an Application Programming Interface (API) by which a distributed application can use *data-centric publish-subscribe* as a communication mechanism. One of the advantages of DDS is its great flexibility to adjust Quality of Service (QoS) parameters, even timing ones, to the needs of the application. Furthermore, the "data-centric communication" paradigm [10], offers a node addressing mode that focuses on inquiring those that have most relevant data. At the first time, obviously every node has to be polled.

The BIDS scheduling method was shown to be fit for adjusting QoS parameters based on the behavior of the tasks and subjected to real-time constraints. In this sense, a Real-Time Module at the application level to schedule tasks composing the application is proposed. Thus, with the inclusion of the BIDS the two major objectives of these applications are accomplished: timing constraints are met and tasks with more important information are polled more frequently.

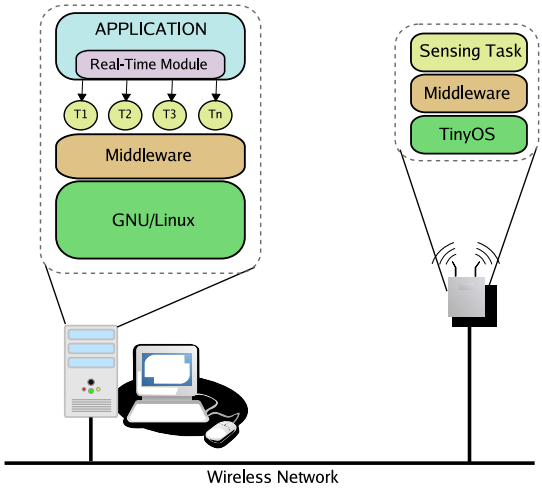

Fig. 3. System architecture.

In Figure 3 the proposed approach to the illustrative case of a single node (*e.g.*, the cluster head) connected to the sink is shown. As can be seen in the figure the application running in the sink is composed of several tasks. These tasks are scheduled by the Real-Time Module, which implements the BIDS policy. Each sensing task running in a sensor communicates with a specific task running in the sink by the DDS that is present in both sides. The OSs running in both platforms are obviously different, in the case of the sink a typical one would be

a GNU/Linux and concerning the sensor the most commonly used is TinyOS [23]. It is worth mentioning, that the proper parameter adjustments needed (done by the BIDS policy) are carried out by the Real-Time Module over the corresponding parameters of the DDS.

As can be inferred from the variables involved in the FWI system (*i.e.*, temperature, relative humidity, accumulated rain and wind speed), there is a clear distinction between different kinds of sensors. This is because there is no sensor in the market that provides the measurement of all of them in the same device. If there were such kind of sensor, the nodes could calculate the FFMC, DMC and DC codes and send them to the sink (even they could calculate every index). As a consequence, the calculation of those codes and the indexes is done by the tasks running in the sink. To perform such calculation within the nodes they would need to communicate among them and a distributed mutual exclusion protocol would be necessary. Instead, without loss of generality, it can be considered that each soft task has three associated sensors, which are used to calculate all the codes and indexes (*i.e.*, using commercial sensors as explained in 3.2). The timebase is managed by the application through the Real-Time Module. Thus, every inquiry done to the sensors has an implicit timebase.

Concerning timing parameters of the tasks, they mainly depend on the designer criteria. However, at least for the task periods, a common range is between 10 and 30 minutes. This may seem a long time for a current computer, but considering the amount of tasks (typically thousands) running concurrently the timeliness becomes a major subject.

With all, the most sensible part of the BIDS approach has to do with the distinction of tasks between IMPORTANT and NOT IMPORTANT ones and with the election of the postponement factor α_s. In Table 1 a typical classification of FWI based on reports of the Canadian Forest Service [7] is shown.

FWI Class	Value Range	Type of Fire	Potential Danger
Low	0 - 5	Creeping surface fire	Self extinguishing fire
Moderate	5 - 10	Low vigor surface fire	Put out with hand tools
High	10 - 20	Slightly vigorous surface fire	Pumps and hoses needed
Very High	20 - 30	Very intense surface fire	Difficult to control
Extreme	30+	Developing active fire	Critical situation

Table 1. FWI classification.

From Table 1 and Equation 1 the $\mathcal{M}$ mapping is given by Equation 2:

$$\mathcal{M}(\delta_{i,j}, FWI) \in \begin{cases} NOT\ IMPORTANT & \text{if } 0 \leq \delta_{i,j-1} < 10 \\ IMPORTANT & \text{if } \delta_{i,j-1} \geq 10 \end{cases} \quad (2)$$

Where the μ_i threshold is given by the FWI index and the partition of the tasks set is as follows: $NOT\ IMPORTANT = \{LOW, MODERATE\}$ and $IMPORTANT = \{HIGH, VERY\ HIGH, EXTREME\}$.

Concerning the α_s parameter, it is also given in terms of the FWI index. With this, a flexible treatment of the different kinds of fire can be made. Equation 3 shows the way this is accomplished.

$$\alpha(FWI) \in \begin{cases} a & \text{if } 0 \leq FWI < 5 \\ b & \text{if } 5 \leq FWI < 10 \\ c & \text{if } 10 \leq FWI < 20 \\ d & \text{if } 20 \leq FWI < 30 \\ 0 & \text{if } FWI \geq 30 \end{cases} \tag{3}$$

Where a, b, c, d and e are integer positive constants properly chosen by the system designer and the following inequality holds:

$$1 \leq d < c < b < a$$

The order relationship showed in the previous inequality establishes the distinction among the importance of the different tasks in the system. In this sense, there is no priority reduction in a task with greater value of α_s, but an assignment of lower bandwidth to the server encapsulating it. This can be observed in the activation time of each task, given by $a_{ij} = a_{i(j-1)} + \gamma_i T_i$, where $\gamma_i = \alpha_s$. With this, a decrease of frequency execution is made over tasks with a FWI value that represents no risk.

5 Conclusions

In this paper, an application-based real-time scheduler for WSN was presented. The scheduler was implemented through a real-time module that performs the scheduling of the tasks composing the application based on the BIDS policy. With this, implicit timing constraints are satisfied and tasks with more important information are inquired more frequently.

The system model proposed is inherently distributed. Thus, hardware and software supporting platforms may be diverse. To overcome this situation, the logical part of the wireless communication was carried out by the DDS middleware. Finally, the proposed scheme was exemplified by an early forest fire detection system.

References

1. OMG: (Data distribution service for real-time systems) `http://www.omg.org` Last Visited: 5/2008.
2. Ordinez, L., Donari, D., Santos, R., Orozco, J.: A behavior priority driven approach for resource reservation scheduling. In: Proceedings of the 2008 ACM Symposium on Applied Computing, Fortaleza, Ceará, Brazil, ACM (2008)
3. Yu, L., Wang, N., Meng, X.: Real-time forest fire detection with wireless sensor networks. In: Proceedings of the 2005 International Conference on Wireless Communications, Networking and Mobile Computing. Volume 2. (23-26 Sept. 2005) 1214–1217

4. Son, B., sork Her, Y., Kim, J.G.: A design and implementation of forest-fires surveillance system based on wireless sensor networks for south korea mountains. IJCSNS International Journal of Computer Science and Network Security **6**(9) (2006) 124–130
5. Hefeeda, M., Bagheri, M.: Wireless sensor networks for early detection of forest fires. In: Proceedings of the 4th IEEE International Conference on Mobile Adhoc and Sensor Systems, Pisa, Italy (2007)
6. Barrenetxea, G., Ingelrest, F., Schaefer, G., Vetterli, M., Couach, O., Parlange, M.: Sensorscope: Out-of-the-box environmental monitoring. In: Proceedings of the 2008 International Conference on Information Processing in Sensor Networks (ipsn 2008), Los Alamitos, CA, USA, IEEE Computer Society (2008) 332–343
7. Canadian, G.: (Canadian forest fire danger rating system) `http://www.nofc.forestry.ca/fire/` Last Visited: 5/2008.
8. Dentoni, M.C., Muñoz, M.M., Epele, F.: Implementación de un sistema nacional de evaluación de peligro de incendios: la experiencia argentina. In: Proceedings of 4th Internation Wildland Fire Conference, Sevilla, España (2007) `http://www.fire.uni-freiburg.de/sevilla-2007.html` Last Visited: 5/2008.
9. Taylor, S.W.: Considerations for applying the canadian forest fire danger rating system in argentina. Technical report, Canadian Forest Service - Pacific Forestry Centre - Natural Resources Canada (2001) `http://www.ambiente.gov.ar/archivos/web/PNMF/File/Canadian%20report.pdf` Last Visited: 5/2008.
10. Römer, K., Kasten, O., Mattern, F.: Middleware challenges for wireless sensor networks. SIGMOBILE Mob. Comput. Commun. Rev. **6**(4) (2002) 59–61
11. IEEE: (Ieee 802.15 working group for wpan) `http://www.ieee802.org/15/` Last Visited: 5/2008.
12. (ZigBee) `http://www.zigbee.org/` Last Visited: 5/2008.
13. (Wibree) `https://www.bluetooth.org/` Last Visited: 5/2008.
14. (Dynamax) `http://www.dynamax.com/` Last Visited: 5/2008.
15. (Crossbow) `http://www.xbow.com` Last Visited: 5/2008.
16. (Devices, D.) `http://www.decagon.com/` Last Visited: 5/2008.
17. (Telemetry, A.) `http://www.adcon.at/` Last Visited: 5/2008.
18. (Campbell) `http://www.campbellsci.com/` Last Visited: 5/2008.
19. (Delta-t) `http://www.delta-t.co.uk/` Last Visited: 5/2008.
20. Rajkumar, R., Juvva, K., Molano, A., Oikawa, S.: Resource kernels: a resource-centric approach to real-time and multimedia systems. (2001) 476–490
21. Liu, C.L., Layland, J.W.: Scheduling algorithms for multiprogramming in a hard-real-time environment. Journal of the ACM **20**(1) (1973) 46–61
22. Wada, H., Boonma, P., Suzuki, J., Oba, K.: Modeling and executing adaptive sensor network applications with the matilda uml virtual machine. In: Proceedings of the 11th IASTED International Conference on Software Engineering and Applications, Cambridge, MA (2007)
23. (TinyOS) `http://www.tinyos.net/` Last Visited: 5/2008.