Giving neurons to sensors. QoS management in wireless sensors networks.

Julio Barbancho, Carlos León, Javier Molina and Antonio Barbancho
Department of Electronic Technology, University of Seville.
C/ Virgen de Africa, 7. Seville 41011 (Spain)
Tlfno.: (+034) 954 55 71 92, Fax: (+034) 954 55 28 33.
{jbarbancho, cleon, fjmolina, ayboc}@us.es

***Abstract*— Public utilities services (gas, water and electricity) have been traditionally automated with several technologies. The main functions that these technologies must support are AMR, Automated Meter Reading, and SCADA, Supervisory Control And Data Acquisition. Most meter manufacturers provide devices with Bluetooth® or ZigBee™ communication features. This characteristic has allowed the inclusion of wireless sensor networks (WSN) in these systems. Once WSNs have appeared in such a scenario, real-time AMR and SCADA applications can be developed with low cost. Data must be routed from every meter to a base station. This paper describes the use of a novel QoS-driven routing algorithm, named SIR: Sensor Intelligence Routing, over a network of meters. An artificial neural network is introduced in every node to manage the routes that data have to follow. The resulting system is named Intelligent Wireless Sensor Network (IWSN).**

Keywords: Wireless sensor networks (WSN); Ad hoc networks, Quality of service (QoS); Artificial neural networks (ANN); Routing; Self-Organizing Map (SOM), ubiquitous computing.

I. Introduction

Wireless sensor networks (WSNs) contain hundreds or thousands of sensors nodes. Due to the sensor features (low-power consumption, low radio range, low memory, low processing capacity, and low cost), self-organizing network is the best suitable network architecture to support applications in such a scenario. Goals like efficient energy management [1], high reliability and availability, communication security, and robustness have become very important issues to be considered.

Our research group, Computer Science for Industrial Applications, from the University of Seville, is working on the development of protocols and system architectures on Wireless Sensor Networks to support Supervisory Control and Data Acquisition (SCADA) applications. We present in this paper a new routing algorithm which introduces artificial intelligence (AI) techniques to measure the QoS supported by the network.

This paper is organized as follows. In section II, we relate the main routing features we should consider in a network communication system. A description of the defined network topology is given. Section III introduces the use of neural networks in sensors for determining the quality of neighborhood links, giving a QoS model for routing protocols. The performance of the use of this technique in existing routing protocols for sensor networks is evaluated by simulation in section IV. Concluding remarks and future works are given on section V.

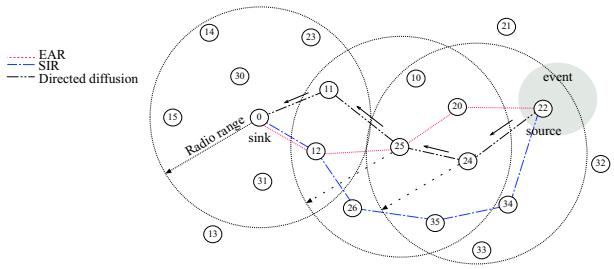

Fig. 1. Event transmission from a source to a sink.

II. Designing the network topology

The WSN architecture as a whole has to take into account different aspects, such as the protocol architecture; Quality-of-Service, dependability, redundancy and imprecision in sensor readings; addressing structures, scalability and energy requirements; geographic and data-centric addressing structures; aggregating data techniques; integration of WSNs into larger networks, bridging different communication protocols; etc.

Acknowledging the impact that route selection will have on network lifetime, we would like to determine route selection in conjunction with the sensor schedule. In general, the routes should be chosen so that nodes that are more critical for use as sensors are routed around as often as possible. In this section, we model this scenario in which sensors are working, and in section III we formalize the routing algorithm, SIR, proposed to solve this problem.

We consider a random distribution of sensors, as depicted in figure 1. Every node has a radio transmitter power and a radio receiver sensibility, which defines an average radio range. Using minimum-transmission-energy routing protocol, nodes route data to the base station through intermediate nodes. Thus nodes act as routers for others nodes' data. The problem is how to elect intermediate nodes, in which the final objective es to minimize the global energy consumption.

In general, routing in WSNs can be divided into *flat-based* routing , *hierarchical-base* routing, and *location-based* routing. In this paper we study networks where all nodes are supposed to be assigned equal roles or functionalities. In

this sense, flat-based routing is best suited for this kind of networks.

Among all the existing flat routing protocols, we have chosen *directed diffusion* [2] and *Energy-Aware Routing* (*EAR*) [3] to evaluate the influence of the use of AI techniques.

III. Introducing neurons in sensor nodes

The necessity of connectivity among nodes introduces the routing problem. In a WSN we need a multi-hop scheme to travel from a source to a destiny. The paths the packets have to follow can be established based on a specific criterion. Possible criteria can be minimum number of hops, minimum latency, maximum data rate, minimum error rate, etc. For example, imagine that all the nodes desire to have a path to route data to a *base station*. In this situation, the problem is solved by the *network backbone formation*.

Our approach to enhance this solution is based on the introduction of artificial intelligence techniques in the WSNs.

A. Network backbone formation

This problem has been studied in mathematics as a particular discipline called *Graph Theory*, which studies the properties of graphs.

We propose a modification on Dijkstra's algorithm to form the network backbone, with the minimum cost paths from the base station or *root*, r, to every node in the network. We have named this algorithm Sensor Intelligence Routing, **SIR**.

B. Quality of Service in Wireless Sensor Networks

Once the backbone formation algorithm is designed, a way of measuring the edge weight parameter, w_{ij}, must be defined. On a first approach we can assume that w_{ij} can be modelled with the number of hops. According to this assumption, $w_{ij} = 1 \ \forall \ i, j \in \mathcal{R}, i \neq j$. However, imagine that we have another scenario in which the node v_j is located in a noisy environment. The collisions over v_j can introduce link failures increasing power consumption and decreasing reliability in this area. In this case, the optimal path from node v_k to the root node can be p', instead of p. It is necessary to modify w_{ij} to solve this problem. The evaluation of the QoS in a specific area can be used to modify this parameter.

The traditional view of QoS in communication networks is concerned with end-to-end delay, packet loss, delay variation and throughput. However, other performance-related features, such as network reliability, availability, communication security and robustness are often neglected in QoS research.

We use a QoS definition based on three types of QoS parameters: timeliness, precision and accuracy. Due to the distributed feature of sensor networks, our approach measures the QoS level in a spread way, instead of an end-to-end paradigm. Each node tests every neighbor link quality with the transmissions of a specific packet named *ping*. With these transmissions every node obtains mean values of latency, error rate, duty cycle and throughput. These are the four metrics we have defined to measure the related QoS parameters.

Once a node, v_i, has tested a neighbor link QoS, qos, it calculates the distance to the root, through node v_j, using the obtained QoS value, $d(v_i) = d(v_j) \cdot qos$.

qos is a variable whose value is obtained as an output of a neural network. This tool is described in section III-C.

C. SOM: Self Organizing Map

One of the most powerful mechanism developed in AI is the Self-Organizing Map (SOM) model, created by Teuvo Kohonen in 1982, at the University of Helsinky, Finland.

SOM is an unsupervised neural network. The neurons are organized in an unidirectional two layers architecture. The first one is the input or sensorial layer, formed by m neurons, one per each input variable. These neurons work as buffers distributing the information sensed in the input space. The input is formed by stochastic samples $\boldsymbol{x}(t) \in \boldsymbol{\mathcal{R}}^m$ from the sensorial space. The second layer is usually formed by a rectangular grid with nxx$n'y$ neurons. Each neuron (i, j) is represented by an m-dimensional weight or reference vector called *synapsis*, $\boldsymbol{w'}_{ij} = [w'_{ij1}, w'_{ij2}, \ldots, w'_{ijm}]$, where m is the dimension of the input vector $\boldsymbol{x}(t)$. The neurons in the output layer -also known as the competitive Kohonen layer- are fully connected to the neurons in the input layer, meaning that every neuron in the input layer is linked to every neuron in the Kohonen layer. In SOM we can distinguish two phases:

Learning phase:

In this phase, neurons from the second layer compete for the privilege of learning among each other, while the correct answer(s) is (are) not known.

This phase is executed in a central data processing unit (e.g. *offline processing*).

Execution phase:

The weights are declared fixed.

First, every neuron (i, j) calculates the similarity between the input vector $\boldsymbol{x}(t)$, $\{x_k \mid 1 \leq k \leq m\}$ and its own synaptic-weight-vector $\boldsymbol{w'}_{ij}$. This function of similarity is based on a predefined similarity criterion.

Next, it is declared a winning neuron, $\boldsymbol{g} = (g_1, g_2)$, with a synaptic-weight-vector, $\boldsymbol{w'_g}$, similar to the input $\boldsymbol{x}$. This phase is implemented in every node as a *C++* function.

SOM gives an output denoted by qos. This value is returned by a function Θ defined by the SOM user, according to its aims. Θ depends on the winning neuron: $qos = \Theta\ (\boldsymbol{g})$. In section IV-B we define this function.

This phase is executed in every node (e.g. *online processing*).

IV. Performance evaluation by simulation

Due to the desire to evaluate the SIR performance, we have created two simulation experiments running on our wireless sensor network simulator OLIMPO [4]. Every node in OLIMPO implements a neural network (SOM) running the execution phase detailed in section IV-B. We have focused our simulation on a wireless sensor network composed by 250 nodes.

A. Noise influence

Noise influence over a node has been modelled as an Additive Gaussian White Noise, (AWGN), originating at the source resistance feeding the receiver.

To evaluate the effect of noise we have defined a node state declared as *failure*. When the BER goes down below a required value (typically 10^{-3}) we assume this node has gone to a failure state. We measure this metric as a percentage of the total lifetime of a node. In section IV we describe two experiments according to different percentages of node failures.

B. SOM creation

Our SOM has a first layer formed by four input neurons, corresponding with every metric defined in section III-B (latency, throughput, error rate and duty cycle); and a second layer formed by twelve output neurons forming a 3x4 matrix.

Next, we detail our SOM implementation process.

Learning phase:

In order to organize the neurons in a two dimensional map, we need a set of input samples $\boldsymbol{x}(t)$=[latency(t), throughput(t), error-rate(t), duty-cycle(t)]. This samples should consider all the QoS environments in which a communication link between a pair of sensor nodes can work. In this sense, we have to simulate special ubiquitous computing environments. These scenarios can be implemented by different noise and data traffic simulations. In our research we create several WSNs over OLIMPO with 250 nodes and different levels of data traffic. The procedure to measure every QoS link between two neighbors is detailed as follows: every pair of nodes (eg. v_i and v_j) is exposed to a level of noise. This noise is introduced increasing the noise power density N_o in the radio channel in the proximity of a determined node. Hence, the signal-to-noise ratio at the detector input of this selected node decreases and consequently the BER related with its links with every neighbor gets worse.

In order to measure the QoS metrics related with every N_o, we run a ping application between a selected pair of nodes (eg. v_i and v_j). Node v_i sends periodically a ping message to node v_j. Because the ping requires acknowledgment (ACK), the way node v_i receives this ACK determines a specific QoS environment, expressed on the four metrics elected: latency (seconds), throughput (bits/sec), error rate (%) and duty cycle(%). This process is repeated 100 times with different N_o and d. This way, we obtain a set of samples which characterize every QoS scenario.

With this information, we construct a self-organizing map using a high performance neural network tool. This process is called *training*, and uses the learning algorithm detailed in section III-C. Because the training is not implemented by the wireless sensor network, we have called this process *offline processing*.

The following phase is considered as the most difficult one. The samples allocated in the SOM form groups, in such a way that all the samples in a group have similar characteristics (latency, throughput, error rate and duty cycle). This way, we obtain a map formed by clusters, where every cluster corresponds with a specific QoS and is assigned a neuron of the output layer. Furthermore, a synaptic-weight matrix $\boldsymbol{w'}_{ij} = [w'_{ij1}, w'_{ij2}, \ldots, w'_{ij4}]$ is formed, where every synapsis identifies a connection between input and output layer.

In order to quantify the QoS level, we study the features of every cluster and, according to the QoS obtained in the samples allocated in the cluster, we assign a value between 0 and 10. As a consequence, e define an output function $\Theta(i,j), i \in [1,3], j \in [1,4]$ with twelve values corresponding with every neuron $(i,j), i \in [1,3], j \in [1,4]$. The highest assignment (10) must correspond to that scenario in which the link measured has the worst QoS predicted. On the other hand, the lowest assignment (0) corresponds to that scenario in which the link measured has the best QoS predicted. The assignment is supervised by an engineer during the offline processing.

Execution phase:

As a consequence of the learning phase, we have declared an output function, that has to be run in every sensor node. This procedure is named the *wining neuron election algorithm*.

In the execution phase, we create a WSN with 250 nodes. Every sensor node measures the QoS periodically running a ping application with every neighbor, which determines an input sample. After a node has collected a set of input samples, it runs the wining neuron election algorithm. After the winning neuron is elected, the node uses the output function Θ to assign a QoS estimation, qos. Finally, this value is employed to modify the distance to the root. Because the execution phase is implemented by the wireless sensor network, we have called this process *online processing*.

C. Evaluating SIR performance

Our SIR algorithm has been evaluated by the realization of two experiments detailed as follows.

Experiment #1: No node failure.

The purpose of this experiment is to evaluate the introduction of AI techniques in a scenario were there is no node failure. This means that no node has gone to a failure state because of noise, collision or battery fail influence.

To simulate this scenario, a wireless sensor network with 250 nodes is created on our simulator OLIMPO. Node # 0 is declare as a sink and node # 22 is declared as a source. At a specific time, an event (eg. an alarm) is provoked in the source. Consequently, the problem now is how to route the event from the specified source to the declared sink.

As detailed in section II we solve this problem with three different routing paradigms: SIR, directed diffusion and EAR. We choose two metrics to analyze the performance of SIR and to compare it to others schemes. These metrics are the average dissipated energy and the average delay.

Average dissipated energy. This metric computes the average work done by a node a in delivering useful tracking information to the sinks. This metric also indicates the overall lifetime of sensor nodes.

Due to transmission distance from a sensor node to the base station is large on a global scale, the transmission energy is much more higher than the received energy. In this network

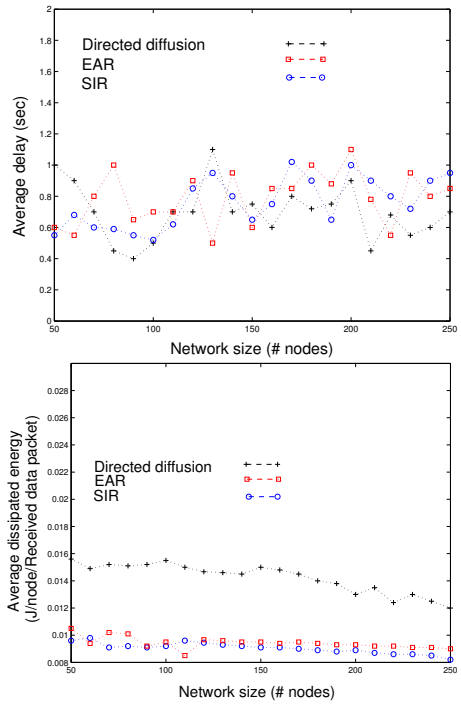

Fig. 2. Average latency and average dissipated energy in a scenario with no simultaneous node failure.

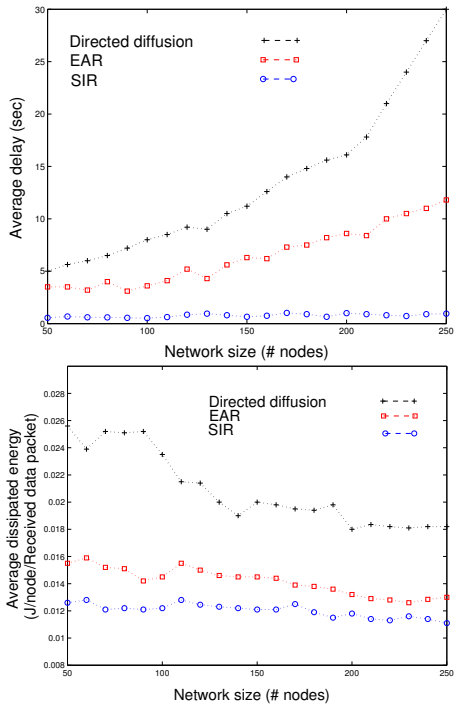

Fig. 3. Average latency and average dissipated energy in a scenario with 20 % simultaneous node failures.

topology, as detailed in section II, the most energy-efficient protocol is the minimum-transmission-energy.

Average Delay. This metric measures the average one-way latency observed between transmitting an event and receiving it at each sink.

We study these metrics as a function of sensor network size. The results are shown in figure 2.

Experiment #2: 20 % simultaneous node failures.

The purpose of this experiment is to evaluate the introduction of AI techniques in a scenario where there is a 20 % of simultaneous node failures. This means that at any instant, 20 % of the nodes in the network are unusable because of noise, collision or battery failure influence.

In these scenario we analyze the problem studied described in experiment #1 with the three paradigms related. The results are shown in figure 3.

V. Conclusion and Future Works

After comparing the results obtained with every routing paradigm, we can conclude that the differences are important when there is a significant percentage of node failures. Thus, while the average delay goes up with the number of sensors in directed diffusion and EAR, it maintains a low level of delay in SIR. The cause of this effect can be found in the fact that while directed diffusion and EAR elect the intermediate nodes using rules based on the propagation of the interest, SIR elects the intermediate nodes running an AI-algorithm. Thus, the path created by SIR avoids the election of intermediate nodes that are prone to failure because of battery draining, interference or noisy environment. Furthermore, the average dissipated energy is less in SIR when the number of nodes in the sensor goes up. We again find the reason in the effect of the election of the intermediate nodes in SIR. The use of AI in every sensor dynamically varies the assignment of this node role, distributing the energy consumption through the network. When the number of nodes is increased, the number of possible paths is increased too. Furthermore, when the percentage of node failures goes up (from 20 % to 40 %) SIR becomes the best suited protocol for these kinds of scenarios. However, although the computational payment for implementing the neural network in a sensor is inapreciable, as detailed in section IV-B, the tradeoff associated with this implementation is the increase of the overhead.

The inclusion of AI techniques (e.g. neural networks) in wireless sensor networks has been proved to be an useful tool to improve network performances.

References

[1] E. Çayirci, T. Çöplü, and O. Emiroğlu. Power aware many to many routing in wireless sensor and actuator networks. In E. Çayircy, Ş. Baydere, and P. Havinga, editors, *Proceedings of the Second Europen Workshop on Wireless Sensor Networks*, pages 236–245, Istanbul, Turkey, February 2005. IEEE, IEEE Press.

[2] C. Intanagonwiwat, R. Govindan, and D. Estrin. Directed diffusion: a scalable and robust communication paradigm for sensor networks. In *Proceedings of ACM Mobicom 2000*, pages 56–67, Boston, MA, USA, 2000.

[3] R.C. Shah and J. Rabaey. Energy aware routing for low energy ad hoc sensor networks. In *Proceeedings of IEEE WCNC*, pages 17–21, Orlando, FL, USA, 2002.

[4] J. Barbancho, F.J. Molina, D. León, J. Ropero, and A. Barbancho. OLIMPO, an ad-hoc wireless sensor network simulator for public utilities applications. In E. Çayircy, Ş. Baydere, and P. Havinga, editors, *Proceedings of the Second Europen Workshop on Wireless Sensor Networks*, pages 419–424, Istanbul, Turkey, February 2005. IEEE, IEEE Press.