Cooperation in Wireless Ad Hoc and Sensor Networks

J. Barbancho[1], D. Cascado[2], J. L. Sevillano[2], C. León[1], A. Linares[2] and F. J. Molina[1]

[1]Department of Electronic Technology, University of Seville, Spain
[2]Department of Computer Architecture, University of Seville, Spain

4.1 Introduction

Cooperation is a key issue in Ad Hoc and sensor networks (Wireless Ad Hoc Sensor Networks, WAdSN) for several reasons. Usually ad-hoc and sensor networks are two concepts related to networks consisting of wireless nodes. Our approach in this chapter considers that these nodes have common features such as tiny size, low computational and storage resources and low autonomy (understood as battery lifetime). Frequently, these nodes can acquire information from the environment via different sensors and can interact with it through actuators.

The term ad hoc refers to a type of network configuration associated with broad networks, such as WiFi. However, this type of network faces different challenges that are not covered in this chapter.

The rapid development of low-cost Micro-Electro-Mechanical-Systems has led to a sharp increase in the use of these devices in several fields such as home automation, industry, health, biology, and so on. The number of potential applications is huge. However, the design of these applications is subject to important constraints: power consumption, limited storage, limited computing capacity, low data rate, and so on.

A typical scenario for application could be summarized as follows: Nodes are spread out during deployment and each device has a limited radio range. Nodes have different roles assigned to them: data source, data sink and gateway.

Two different paradigms exit when trying to identify the main objective of such a network. The classical approach assumes that the main purpose of the network is to transport data from the sources to the sinks (which we could name data transport paradigm). Data often needs more than one hop to reach the sink node. As there is no network manager (ad hoc approach), nodes have to collaborate to form routes through which data can be transmitted. Once the data reaches the sinks, a base station (usually a personal computer) processes the collected information. In this paradigm, cooperation is limited to an exchange of information. In contrast, the new approach considers the network as a whole. In this way, although nodes may have low resources they can store, process and send small pieces of information. So, cooperation is understood as a distributed way of achieving an objective, like in a colony of ants. The main characteristics are: only local computations are involved, no global knowledge is assumed (except for the common objective), no centralized control is needed, and scalability is relatively easy to achieve [~~55~~]. In this second approach, cooperation becomes a much more important issue (cooperative paradigm).

Collaboration can be established in such a network in pursuit of different goals, as detailed in in the following text.

Time synchronization, calibration and localization are topics that can be studied under the new approach of cooperation in wireless sensor networks. Every node has its own clock that sets the rhythm the microprocessor has to follow. Usually these clocks have limited precision, and microprocessors tolerate changes in the time measurement to work correctly. If there is no external time reference the network has to create the rules to define a global clock, so that nodes can synchronize their clocks estimating and compensating clock skew and offset. These rules could be designed in a cooperative way, enabling all network nodes to exchange synchronization messages. Calibration could be approached in the same way. Given that a physical magnitude is measured by a network of sensors, they can cooperate themselves to reach a given calibration accuracy. In this sense, localization also uses measured data where the nodes need to match cooperatively. Synchronization, localization and calibration are discussed in Section 4.2.1.

Having to manage data routes in the whole network is a problem that can also be solved with cooperative methods. Routing can be understood from the point of view of power consumption. Nodes have to cooperate to achieve a common trade-off: data delivery versus energy expenditure. Radio transmission is the process that uses most energy in the network. Reducing the number of packets that are transmitted is one of the strategies that we can follow to extend network lifetime. The selection of the path that the packets have to follow is directly related to the number of packets that are used in the task of delivering information. In Section 4.2.2 we study an example that shows how power consumption can be reduced by managing the quality of the radio links involved in data transmissions. The quality of the radio links is estimated by the nodes using Artificial Neural Networks. This information is exchanged in a cooperative way to dynamically decide packet flow. The effect that this behaviour produces is similar to that of the blood flow in a human body. There is no central control; the decision about where the blood has to flow is taken by all the arteries and veins.

Other issues that are studied under the paradigm of cooperative networking are data aggregation and data fusion. Several examples of applications based on these topics are described in Section 4.2.3 to illustrate the usefulness of cooperative networking.

Having discussed why cooperation is needed in WAdSN, we shall then look at how cooperation can be implemented. Wireless sensor nodes can cooperate among themselves in many different ways in pursuit of the common goal. Researchers have proposed a wide array of mechanisms for implementing cooperation, such as multi-agent intelligent systems (MAS), neural networks, middleware systems, and so on. These techniques are covered in Section 4.3.

Finally, Section 4.4 presents a summary of the chapter, focusing on future lines of research into cooperation in ad hoc and sensor networks.

4.2 Why Could Cooperation in WAdSN be Useful?

4.2.1 Time Synchronization, Localization and Calibration

Observation of a physical phenomenon requires the specification of the time and location of occurrence, with a measure of how confident we can be about the observation. Therefore, if a

Wireless Ad Hoc Sensor Network (WAdSN) is used for this observation, nodes must know their own position and the time, as well as being able to refer their output to a well-defined scale. This is the main argument for providing WAdSNs with localization, synchronization and calibration techniques. However, there are many other reasons why these techniques may be useful.

For instance, location information may not only be used to stamp a given measure; it may also be a requirement in many sensor network applications: tracking of mobile vehicles or animals; monitoring of elderly and disabled people in residencies; support for navigation, logistics and inventory management, and so on. In these cases, localization can be seen as a service: some nodes provide location information and others require this service to determine their locations or in turn to provide location based services or perform location-aware functions. On the other hand, when a node provides a measure with location information, data fusion techniques can reduce traffic and energy consumption. For instance, data aggregation can be used to combine the information coming from a number of sensors in the same area (i.e. temperature) to provide more meaningful information (i.e. forest fire alarm). Data aggregation requires the nodes to be positioned with a level of accuracy that depends on the application.

Time synchronization is particularly important in wireless sensor networks. One of the most important reasons is energy saving. Data communication is the major source of energy expenditure (much higher than sensing or data processing [~~7~~]), and in data communication there are four major sources of energy waste [~~57~~]:

- **Collision:** colliding packets must be discarded and therefore waste energy. Retransmissions also increase energy consumption.
- **Overhearing:** when a node picks up packets that are destined to other nodes.
- **Control packet overhead:** as sending and receiving control packets also consumes energy.
- **Idle listening:** i.e., listening to receive possible traffic that is not sent.

The energy consumed by idle listening may be of the same order of magnitude as the energy required for receiving [~~57~~], [~~31~~]. As WAdSNs are characterized by low data rates, according to the node features we have assumed in this chapter, nodes can be in idle mode for most of the time and therefore idle listening becomes the major source of energy waste in WAdSNs. Nodes should be kept awake the minimum time required to exchange data to save energy. There is often only specific time periods of interest. Nodes should be synchronized to perform the observation within these time periods, and then switch off the radio during the inactive periods.

Collisions also waste energy, and this is one of the drawbacks of contention-based MAC (Medium Access Control) mechanisms. Usually, protocols based on reservation and scheduling are preferred when energy consumption is the main concern. For instance, a TDMA (Time Division Multiple Access)-based MAC allows energy saving because it avoids collisions, reduces overhead and eases the implementation of the duty cycle of the radio. Of course, TDMA protocols require tight time synchronization.

It is true that some WAdSNs may use contention-based MACs provided they transmit with low power and low data rate, so that the probability of collision is very low. For instance, this is the case of the beaconless mode of the IEEE 802.15.4 protocol (the basis of the ZigBee standard) which is essentially an unslotted CSMA/CA mode. However, this simple solution presents several drawbacks. For instance, when any node is mobile (i.e. not fixed), the beaconless mode is not suitable because there is no periodical communication with the coordinator node, so the mobile node may assume its association although it may have lost the link with the coordinator [~~59~~]. On the other hand, the 802.15.4 protocol includes a beacon-enabled mode where a so-called ZigBee Coordinator (ZC) periodically transmits beacon frames that establish a superframe structure. In this superframe, devices may transmit using a slotted CSMA/CA medium access, with an optional

Contention-Free Period (CFP). Even if these CFPs are not used, the periodic transmission of beacons provides a way of synchronizing all nodes. It is also possible to change the duty cycle to achieve low power consumptions: the device, upon receiving the first beacon, gets current superframe structure and thus knows when to activate its receiver for the next beacon (just a bit earlier to save power). The device periodically enables its radio to receive the beacon frame, performs the required transmissions and then switches off the radio during the inactive period. With mobile devices in the beacon-enabled mode, if a node does not receive a predetermined number of beacon frames then it considers itself as an ‘orphan’ node, beginning a realignment procedure to be associated again with a coordinator. Therefore, generally speaking, when there are energy and/or timing requirements the right choice is the beacon-enabled mode which means that nodes are time synchronized.

A distributed alternative to the 802.15.4 beacon enabled mode is S-MAC [~~57~~], where instead of a central coordinator, neighbouring nodes coordinate their sleep schedules. The so-called ‘coordinated sleeping’ requires each node to maintain a schedule table that stores the schedules of its neighbours. This way they know their neighbours’ scheduled listen time and they exchange their schedules within these periods. This is a good example of how collaborative methods can be used not only to reduce energy waste but also control overhead and latency.

There are many other uses of time synchronization. For instance, at lower layers tight time synchronization may also be fundamental for increasing data rates (short bit times) or enhancing noise immunity (as in the frequency hopping mechanism). At higher layers, data fusion requires synchronization for two tasks: time scheduling and time stamping. The first is needed when the nodes coordinate to perform cooperative communications. The second is commonly used when data is fused taking into account the collecting instant; for example, to perform event detection, tracking, reconstruction of system’s state for control algorithms, off line analysis, and so on. These issues will be further discussed in the following sections.

There is a close relationship between calibration, time synchronization and localization in wireless sensor networks. Calibration can be broadly defined as the mapping of the output of a sensor to a well-defined scale, which as pointed out in [~~47~~] includes time synchronization as a special case simply considering the hardware clock as a sensor whose output has to be mapped to a timescale. Localization is also closely connected to synchronization [~~44~~] and by extension to calibration. Localization is usually performed measuring angle-of-arrival (AoA), time-difference-of-arrival (TDOA) or Received Signal Strength (RSS), and in any case it also entails mapping one physical magnitude to a well-defined scale to infer the node position. Of course, RSS based localization systems require careful calibration of the receivers. Furthermore, some of these localization methods are based on the measurement of time of flight or difference of arrival time of a signal so they require synchronized time. For instance, in [~~36~~] a location system is described where the position is obtained by measuring the time of flight of ultrasonic signals sent by the ‘beacons’. These beacons must be synchronized so that the ultrasonic signals are sent by the beacons at the same time. In a sense, the opposite is also true: synchronization accuracy depends on the relative location of the nodes. If nodes are located too far away from each other, multiple hops are needed to connect them so synchronization messages suffer from higher delays and these delays decrease synchronization accuracy [~~37~~].

Differences in time synchronization, localization and calibration (SLC for short in what follows) do not stop similar techniques and algorithms being transferred among the three. For instance, unlike other physical variables time is not bounded, so the time difference between two sensors will also grow unbounded unless periodic time synchronization is performed. However, it should be noted that this is also the case with calibration and localization in many wireless sensor networks. Obviously, if nodes are mobile or simply their position changes because of the effect of environmental changes (wind, rain, etc.), periodic location information has to be exchanged among the nodes to perform localization. Similarly, although calibration of conventional sensors is usually performed only once before operation, periodical calibration may also be required. The reason is that changes in environmental parameters during the lifetime of typically low-cost sensors may affect their operation [~~47~~]. This may be especially important in the case of mobile sensors or in safety critical applications. It is for all these reasons that we present in this chapter a unified discussion of synchronization/localization/calibration (SLC). The three share the same classes identified by [~~47~~], for instance:

- **External vs. internal:** If SLC information is provided by an external node, then we say that the SLC is external. For instance, a GPS (Global Positioning System) receiver obtains its position or absolute time from external sources. Calibration is external if the output of all sensors map to a given scale [~~47~~]. For internal calibration, the only thing that is needed is for all nodes to output the same value (at least within a given interval) if they are exposed to the same stimulus, but not necessarily equal to what an external sensor would output. Similarly, internal localization or synchronization only tries to maintain consistency among the nodes, but without conforming to an absolute position or time. In some applications, this internal SLC could be limited to a subset of the nodes (for instance, those observing a given phenomenon), or could have different precision requirements in different subsets.
- **Continuous vs. on demand:** Continuous SLC means that nodes try to maintain synchronization, localization and calibration at all times. This approach is not very efficient in WAdSN especially in terms of energy consumption. On demand SLC is much more appropriate in this case, particularly because the system does not waste energy in SLC during the long idle periods. On demand SLC may be:
  - **Event-triggered:** when the event has occurred the node is synchronized/located/calibrated.
  - **Time triggered:** periodically or when an external node sends the order to perform the SLC. Due to the uncertainty of delays in a WAdSN, especially with multi-hops, this order may be anticipated: nodes receive the order to take the sample at some future time (of course, this requires nodes to be synchronized).
- **Master-slave vs. peer to peer:** Usually, two kinds of nodes are considered; ‘masters’ that know their time and/or location (which we call ‘reference nodes’) and ‘slaves’ that must estimate their location and/or time by some algorithm (‘common nodes’). Reference nodes (sometimes called ‘anchors’ [~~35~~] or ‘beacons’ [~~14~~]) can obtain their coordinates from an external system (such as GPS) or simply act as a time or location reference for the other nodes. This would be the case of an internal SLC, where only relative time or location information is required. Some algorithm must be used for the common nodes to estimate their SLC information based on the known information from the reference nodes. However, note that sometimes reference nodes are not required, especially with internal SLC. In these cases (most of which are time synchronization protocols), peer-to-peer solutions offer more flexibility but are more difficult to control.

A recent survey of localization techniques is [~~35~~]. When all common nodes are at one-hop distance from a sufficient number of reference nodes, then localization techniques do not require nodes to cooperate. They can simply use some signal measuring system to infer the distance between the reference node(s) and the common node (or the element to be located). As mentioned before, they simply use methods based on measuring Angle Of Arrival (AoA), Time Difference Of Arrival (TDOA) or Received Signal Strength (RSS). Once the distance is obtained, multilateration is used to obtain the coordinates of the element to be located. Redundant measurements are needed to cope with difficulties such as non-line of sight errors [~~14~~]. Less accurate methods exist that are not based on these measurement techniques. For instance, in [~~12~~] a node estimates its position as the centroid of the coordinates of the reference nodes that are within its communications range. Of course, this simple solution requires high node density and regular topologies, and in any case it can only provide a coarse estimation. More complex algorithms are described in [~~10~~] and [~~35~~].

However, cooperative algorithms are more useful for common nodes that are not necessarily at one-hop distance. For instance, in [~~41~~] all reference nodes flood messages with their location, and these messages, which are propagated hop-by-hop, include a hop-count so that each receiver can record their hop-count distance to the reference nodes. When another reference node receives this message, it computes the mean one-hop distance using the known location of both reference nodes and the hop-count. This one-hop estimation is sent back to the network so that common nodes can estimate their actual distance to reference nodes from their own hop-count.

As for time synchronization, cooperation is more important as the clock skew (differences in the frequencies of the nodes' clocks) and drift (differences in the rate at which frequencies change) increase over time even in static WAdSNs. Many different techniques for clock synchronization exist, and they include all the classes discussed above: external vs. internal synchronization, continuous vs. on demand, Master-slave vs. peer-to-peer, and so on. [~~53~~], [~~47~~]. Regarding cooperation, it is also interesting to distinguish between methods where there is a sender transmitting the current clock values to the receiver as timestamps (sender-to-receiver) and methods where synchronization messages are broadcast so that all receivers receive the message at approximately the same time (receiver-to-receiver). According to Maróti et al., the most significant delays when transmitting messages over a wireless link are those from the send, receive and access processes [~~38~~]. Sender-to-receiver protocols like TPSN (Timing-sync Protocol for Sensor Networks) or the Flooding Time Synchronization Protocol (FTSP) avoid the indeterminism of these delays working at the MAC (Medium Access Control) layer to precisely timestamp messages, which is not always possible when using standard or complex protocols. In receiver-to-receiver methods, the use of reference broadcast messages eliminates the time uncertainty introduced by the send and access processes and sets a temporal reference shared by all the nodes.

One of the best known receiver-to-receiver synchronization methods is MBS [~~17~~]. Elson and Estrin propose a post-facto synchronization method; that is, a method that performs synchronization only when it is needed [~~16~~]. The scattering method they propose does not give a common time reference to the broadcast sender; it only synchronizes receivers. However, several authors point out the need for setting a network time to propagate the synchronism over a multi-hop network using broadcasts [~~17~~], [~~42~~]. Thus, nodes in broadcast domain need to cooperate by sharing timing information among them to determine the global network time (GNT). This makes them non-usable in large networks because the amount of data exchange required is huge [~~23~~]. On the other hand, Multi-hop Broadcast Synchronization protocol [~~37~~] is a receiver-to-receiver synchronization scheme that nonetheless obtains a GNT working at the application layer. In this protocol each reference broadcast informs about the timestamp of the previous one. This way message exchanging is minimized, being similar to other sender-to-receiver methods, which makes MBS suitable for large multi-hop networks.

Finally, works on calibration in WAdSNs are very scarce. Römer et al. [~~47~~] discuss external and internal calibration methods. Collaboration among nodes is important mainly in internal methods such as Collaborative In-Place Calibration (CIC) [~~13~~]. It is interesting to note that in this method location, synchronization and calibration are again closely related. For instance, nodes need to be synchronized. Node location is also important, as CIC begins with collocated pairs of nodes that are calibrated against each other.

A final comment is that SLC in cooperative wireless sensor networks can be seen from two different points of view: as we have shown, cooperation is needed to perform SLC. Many of the previously discussed algorithms are based on collaboration among nodes. On the other hand, SLC is often needed to perform cooperation. For instance, in dense WAdSNs nodes that are close to each other may be redundant from a routing point of view. The network can then be partitioned into a grid, with some active nodes assuming the role of grid ‘leaders’ while others are kept asleep except for periodical wake-ups to communicate with the active nodes. This way nodes in sleep mode save energy. Active and asleep nodes cooperate by alternating their roles to prevent active nodes from running out of energy. Some of these algorithms need nodes to know their position to select which neighbouring nodes will take the role of grid leaders [~~56~~].

4.2.2 Routing

The need for connectivity among nodes introduces the routing problem. There are several reasons that make routing in wireless sensor networks a challenge:

- First, a global addressing scheme, as in Internet Protocol, cannot be maintained. This is because the network is formed by a huge number of nodes. Furthermore, sensors are deployed in such a way (ad-hoc) that self-configuration techniques are required.
- Second, the data usually flow from several sources to a few number of sinks.
- Third, the nodes provide highly correlated information. So, the implementation of data aggregation and fusion techniques is required. We’ll raise this problem in Section 4.2.3.

Due to these features, routing in WAdSN has traditionally been studied as a problem that could be solved by solutions based on distributed systems. In other words, collaboration among nodes is needed to route data.

In the following we shall focus our approach to the routing problem on the need for a multihop scheme to let data 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 a minimum number of hops, minimum latency, maximum data rate, minimum error rate, etc. For example, imagine that all the nodes wish to have a path to route data to the base station. In this situation, the problem could be solved by a technique called network backbone formation. In general, sensor networks do not use node addresses. In contrast, sensor locations could be used to solve the routing problem, as seen in Section 4.2.1. Another approach to routing is to query all sensor nodes matching a certain criterion instead of utilizing individual node addresses; this is discussed in Section 4.3.1.

An example to enhance this solution is based on the introduction of artificial intelligence techniques in the WAdSNs: expert systems, artificial neural networks, fuzzy logic and genetic algorithms. Although there are many authors who have proposed the introduction of different AI techniques in several applications over WAdSNs [~~51~~], [~~28~~], [~~54~~], only a few (e.g. [~~6~~]) have considered the possibility of implementing an AI technique inside a sensor node. Due to the processing constraints we have to consider in a sensor node, the best suited of all these techniques, is the self-organizing-map (SOM). This kind of artificial neural network is based on the self organization concept.

The network backbone formation has been studied in mathematics as a particular discipline called *Graph Theory*, which studies the properties of graphs. A directed graph G is an ordered pair $G := (V, A)$ with V, a set of vertices or nodes, v_i, and A, a set of ordered pairs of vertices, called directed edges, arcs, or arrows. An edge $v_{xy} = (x, y)$ is considered to be directed from x to y; where y is called the head and x is called the tail of the edge.

In 1959, E. Dijkstra proposed an algorithm that solves the single-source shortest path problem for a directed graph with nonnegative edge weights. In our wireless sensor network we assume that all the links are symmetrical, in the sense that if a node A can reach a node B, then the node

B can reach the node A. With these kinds of links, we can model our network as an undirected graph $G := (V, E)$.

Barbancho et al. [9] propose a modification of 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. The proposed algorithm is named Sensor Intelligence Routing, SIR. In Dijkstra's algorithm the graph has arrows and in SIR the graph has edges. Every edge between nodes v_i and v_j has a weight, w_{ij}, and it is easy to prove that $w_{ij} = w_{ji}$. The distance from the base station to a node v_i is named $d(v_i)$. The set of nodes which are successors or predecessors of a node v_i is denoted by $\Gamma(v_{\mathrm{i}})$, and is defined in this way: $\Gamma(v_i) = \{v_j \in V | (v_i, v_j) \in E\}$. If we denote a path from the root node to a node v_k by p, we can define $\Gamma_{\mathrm{p}}(v_j)$, if $v_j \in p$, as the subset of nodes which are predecessors or successors of node v_j. It is also assumed that $V = \{r, v_i\}_{\mathrm{i}}$ and that there is a subset of V, T, defined as $T := V - \{r\}$. With this terminology, the SIR algorithm can be described as detailed in Table 4.1.

In the first step, every node is assigned an initial cost to reach the sink. In the following steps this cost is updated depending on the neighbourhood. The algorithm ends when there are no more possible updates.

Once the backbone formation algorithm has been designed, a way of measuring the edge weight parameter, w_{ij}, must be defined. On first approach it could be assumed that w_{ij} can be modelled with the number of hops. According to this assumption, $w_{ij} = 1 \,\forall\, i,\ j \in \mathbb{R},\ i \neq j$. However, let us 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 may be p', instead of p. Modification of w_{ij} is required to solve this problem. The evaluation of the quality of service (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. Several authors have proposed architectures and integrated frameworks to achieve guaranteed levels of network performance [8], [48]. However, other performance-related features, such as network reliability, availability, communication security and robustness are often neglected in QoS research. The definition of QoS requires some enlargement if we want to use it as a criterion to support the goal of controlling the network.

Table 4.1 Network Backbone Formation Algorithm

Step 1:	Set up phase: $d(r) = 0$ $d(v_i) = \begin{cases} w_{ri} \text{ if } v_i \in \Gamma(r) \\ \infty \text{ if } v_i \notin \Gamma(r) \end{cases}$ $\Gamma_{\mathrm{p}}(v_i) = \begin{cases} r \text{ if } v_i \in \Gamma(r) \\ 0 \text{ if } v_i \notin \Gamma(r) \end{cases}$
Step 2:	Find a $v_j \in T$ such as $d(v_j) = \min\{d(v_i) | v_i \in T\}$ Do $T = T - \{v_j\}$
Step 3:	$\forall\ v_i \in T \cap \Gamma(v_i)$ calculate $t_i := d(v_j) + w_{ji}$ If $t_i < d(v_i)$ do $d(v_i) = t_i$
Step 4:	If $|T| > 0$ go to step 2 If $|T| = 0$ stop

This way, sensors participate equally in the network, conserving energy and maintaining the required application performance. What is sensor network QoS? Iyer and Kleinrock proposed in [~~25~~] a definition of sensor network QoS based on sensor network resolution. They define resolution as the optimum number of sensors sending information toward information-collecting sinks, typically base stations. Kay and Frolik defined sensor network QoS in terms of how many of the deployed sensors are active [~~30~~]. The same idea is discussed in [~~43~~] by Perillo et al., and in [~~46~~] by Rakocevic et al.

Barbancho et al. [~~9~~] uses a 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 neighbour 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 has tested a neighbour link QoS, it calculates the distance to the root using the obtained QoS value. Expression 4.1 represents the way a node v_i calculates the distance to the root through node v_j, where qos is a variable whose value is obtained as an output of a neural network.

$$d(v_i) = d(v_j) \cdot qos \tag{4.1}$$

According to this strategy, data from source nodes travels through dynamic paths, avoiding the regions with the worst quality of service levels.

4.2.3 Data Aggregation and Fusion

Data fusion is an analogous procedure to the human cognitive process. This procedure is like a set of multidisciplinary techniques with the objective of integrating data from several sensors. The main goal of this integration is to produce a relevant output that identifies the state of the phenomenon being studied. In this sense data fusion could be understood as a connection between a sensory layer and a reaction layer.

The Joint Directors of Laboratories (JDL), an organism that belongs to the Department of Defense of the United States of America has defined a process model to describe the data fusion process. The model has five levels [~~20~~]:

- **Level 0:** Source preprocessing. Data is standardized on this level. This process is executed on the data acquisition hardware associated with the sensor. This could be called pre-fusion. On this level every sensor is provided with independent data from other sensors.
- **Level 1:** Object refinement. The data is identified with an entity. This entity is understood as an application which is being carried out. For example: a medical diagnosis.
- **Level 2:** Situation refinement. The results obtained with the last level are evaluated according to the speed of data processing. If the speed is too low, data aggregation algorithms have to be carried out to increase the speed.
- **Level 3:** Impact assessment. The phenomenon is studied with the acquired data. This study intends to predict the performance of the phenomenon in the future.
- **Level 4:** Process refinement. General refinement in real time.
- **Level 5:** Cognitive refinement. On this last level, the human interprets the results obtained by the data fusion processing. This way possible error can be detected.

Data fusion has been considered in wireless sensor networks as a mechanism for reducing the number of packets transmitted among sensors. This reduction consequently entails lower

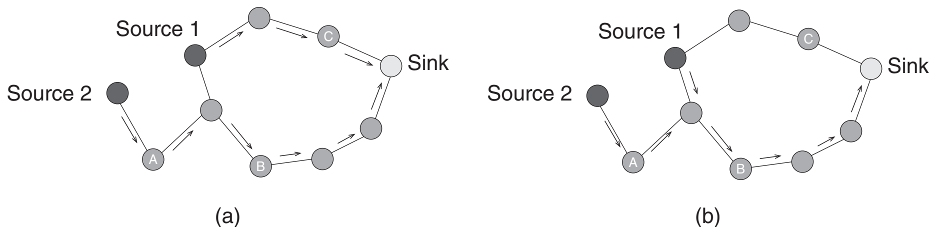

Figure 4.1 Different routing approaches: (a) address-centric routing and (b) data-centric routing.

average network power consumption. In such a network, the information obtained by a node and the nodes that belong to its neighbourhood is often redundant and highly correlated. In this scenario, it makes sense to process data locally instead of sending it to a central station for processing.

Routing is a problem that needs to be borne in mind when considering data delivery to sinks. Conventional address-centric routing finds the shortest routes from sources to sinks. However this approach does not provide an optimal solution. Better energy and bandwidth efficiency are obtained with the data-centric routing approach. It considers data aggregation along a route with multiple sources and one sink. This approach is particularly efficient in scenarios where the sources are located close to one another and far away from the sink [~~32~~]. This comparison is depicted in Figure 4.1. In the address-centric routing approach, source 1 chooses the path through node C to deliver data to the sink with the minimum numbers of hops. At the same time, source 2 chooses the path through node B following the same criterion. The total amount of hops and the number of packets travelling through the network are 9. In the data-centric routing approach, source 1 chooses the path through node B. In this approach the node between A and B relays and aggregates data from sources 1 and 2. Consequently only seven hops are required for data delivery. Research into data aggregation and fusion can be split into three categories:

- **Research based on the fusion function.** The main goals are to suppress redundancy and estimate a system parameter [~~34~~].
- **Research based the system architecture.** The main goal is to determine how many data aggregators must be used in the wireless sensor network and where.
- **Research based on trade-off and resources.** This research focuses on several trade-off, such as energy vs. estimation accuracy, energy vs. aggregation latency and bandwidth vs. aggregation [~~49~~].

The following is an example of an application in which fusion can bring these three categories into play:

The South of Spain is home to one of the largest wildlife reserves in Europe: the National Park of Doñana which covers a huge area of about 520 km^2. It has a variety of ecosystems: Mediterranean forest, marshland, wetlands, dunes, beaches, and so on. Doñana is a wildlife reserve protected by the Spanish Government; it is an area that has seen little human interference through history. Nowadays no-one can enter Doñana without special permission. As a result, it is a privileged scenario for studying populations of wild animals and plants.

The international scientific community has been conducting research in Doñana for over 50 years. Biologists have traditionally deployed different kinds of sensors to gather information about natural phenomena such as vegetation stress, water levels of the aquiferous system, growth of animal populations, etc. These sensors are usually connected to a data-logger that stores data for a period of time. Nowadays the Park has a WiFi infrastructure that provides connections to an intranet from every part of the Park. This way, it is possible to access an environmental variable in real time. For example, anyone can observe online the ultraviolet irradiation at Santa Olalla's lake through the information provided by the Doñana Biological Station, EBD [1].

The deployment of wireless sensor networks in this Park has increased the number of information sources. Consequently data fusion and aggregation techniques are needed to manage such a huge amount of data. The solution that is being implemented is based on the integration of computational intelligence in the wireless sensor nodes. Some approaches to this integration are described in Section 4.3.3.

In our approach every node executes an artificial neuronal network to estimate the water level in flood zones. This task is performed once a day. The neural network is a Self-Organized Map (SOM) trained with historical data obtained by the EBD over several years. The SOM receives data from environmental sensors installed in the node (level 0, source preprocessing according to the JDL definition of fusion) and produces an output that identifies the state of the water level with a neuron at the output layer of the SOM (level 1, object refinement). This procedure constitutes a data fusion technique that suppresses the redundancy at the estimation of the system parameter, that is, the water level in the flood zone under study (level 3, impact assessment, and level 4, process refinement).

A node processes this information in the following way. If the state is different to the state estimated the day before, the node notifies this event to a base station. If the state is equal to the state obtained the day before, the node sends no information. This way the nodes waste neither energy nor bandwidth. In this situation we have a case of research based on trade-off and resources: energy vs. estimation accuracy.

Hence the nodes are deployed over a flood zone, and every node produces its own estimation; it is clear that there will be several estimations spread over the sensor network. Some nodes will change their daily estimation with a higher frequency than others. Consequently the former will produce more traffic than the latter. As the information is sent to a base station using multihop protocols based on IEEE 802.15.4 standard, this information needs to be aggregated. At this point a second fusion technique is implemented to determine how many data aggregators must be used in the wireless sensor network and where. The use of aggregators enhances the speed of data processing (level 2, situations refinement).

Once the base station receives the aggregated data, it sends it to the EBD through the intranet network (WiFi and Internet). At the EBD the biologists interpret the results obtained by the sensor networks and evaluate the accuracy (level 5, cognitive refinement).

This application example illustrates the cooperative activities performed by all the nodes in the network towards a common goal, that is, the estimation of the water level of a flood zone. From the individual tasks developed at each node to the communications between them, all the activities constitute a cooperative networking scheme.

4.3 Research Directions for Cooperation in WAdSN

A set of network devices is a system that could be quite hard to manage if only the physical and MAC layers are implemented. As we mentioned above, aspects such as time synchronization, calibration and location are fundamental for implementation in a WAdSN. In this sense, routing, data aggregation and fusion can also be understood as fundamental. Obviously the upper layers of the network stack are the proper place for implementation. Therefore cooperation networking, as we have seen in the previous section, must be implemented in these upper layers. There are many different ways of doing this and we will discuss some of them in the following subsections. In particular, we will look at some of the current approaches that may facilitate the implementation of cooperation in WAdSN.

4.3.1 Middleware for WAdSN

All the nodes in a wireless sensor network have to run a program written in a specific programming language. Compatibility problems may arise if the executable code of this program runs in different hardware platforms to the network nodes. The use of operating systems solves part of these problems, bringing a hardware abstraction that makes it slightly easier to write programs (the program can be written in the same language for all network nodes but, if different hardware and/or operating systems are used, it would be diffcult to create a source code without incompatibilities between the different versions of hardware, compilers and operating systems). Moreover, additional problems arise when different networking technologies are used to communicate the nodes of the system (imagine for instance that one set of tiny nodes coexists with another set of more powerful nodes, all running the same program), or when one client/server architecture has to be developed in a simple way.

These problems, specific to distributed systems, can be solved by adding an additional layer between applications and operating system: the middleware (Figure 4.2). This layer provides a common interface for communicating clients and services, methods for service discovering (how to know if there are services in the network), service description (how to know what it does), service control (how to use it), and even a common language for developing applications, services and clients. Sharing of services is a good way of performing collaboration among nodes in a WAdSN. Furthermore, the unified view that the middleware provides allows the system to be considered as a single system and not just as the sum of a (high) number of sensor nodes. There are several middleware architectures that meet these requirements:

- **Distributed Tuples:** offers abstraction for distributed tuples. Examples: L2IMBO, LINDA.
- **Remote Procedure Call:** abstraction of external procedure invocation: Java Remote Method Invocation, Modula-3, XML-RPC, .NET Remoting.
- **Message-Oriented Middleware:** message queue abstraction for messages between distributed applications. Examples: Advanced Message Queuing Protocol, Java Message Service.

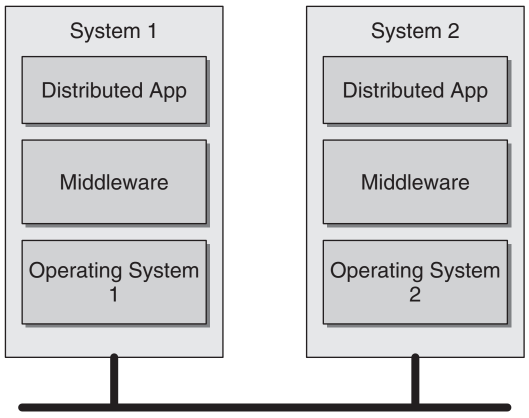

Figure 4.2 Middleware in distributed systems.

- **Object Request Broker:** remote object management abstraction allows using remote objects as if they were in the user's own space. Examples: DCOM, CORBA, COM.

But, why should we have to do these things with a WAdSN? In a way, a WAdSN can be seen as a distributed system. It has a relatively large set of computational nodes that work together to perform a specific task: for instance gathering environmental information and sending it to a central node for processing. But this vision can be interpreted in another way: each network sensor performs a service that offers actions such as reading the environmental values obtained in the sensors, reset, calibration, and an event subscription that advises clients if a sensor reading changes. At the same time, nodes like the central node mentioned above can be clients of these services and conduct operations in the service of the sensors/actuators.

The use of middleware has until now been restricted to large systems where resource restrictions have not been a problem and the networking technologies used have been very fast, homogeneous and reliable. However, in WAdSN, the resources (memory and computing power) are very limited, the small batteries of sensor nodes impose limits on data communications and operating systems and hardware platforms are very heterogeneous. This heterogeneity may simply be due to the fact that the system is made up of relatively simple sensor nodes plus a number of central nodes with larger resources and better capabilities. In addition, this new layer imposes the use of more memory for programs, more computation power requirements, and more energy wasted in communications due to the protocols used to maintain client-server architecture (for example, the protocol for service discovering requires more messages to be sent between nodes).

Thus, the design principles of a middleware oriented towards WAdSN have to be quite different to a 'standard' middleware for these reasons. We must also consider the nature of applications implemented in this kind of system [~~22~~]:

- Distributed by nature.
- Limited by low machine resources.
- Focused on energy saving.
- Dynamic availability and quality of data collected from the environment.
- Constrained applications quality of services.

So, the design principles of WAdSN middleware could be summarized as follows [~~58~~]:

- Data-centric: focused on supporting distributed data requests.
- Distributed and collaborative algorithms.
- Support of data aggregation.
- Lightweight.
- Cluster based architecture: the whole system is a set of clusters, each of which consists of several sensor nodes and one head node responsible for cluster coordination.
- Virtual-machine abstraction and (preferably) a common language to specify the applications in the whole system, regardless of the operating system and hardware platform.

Nowadays, some implementations of middleware can be accommodated in WAdSN, and they follow the above-mentioned principles.

- **Cougar [~~11~~] and SINA [~~26~~]:** both provide a distributed database interface to query WAdSN data in a SQL-like style. Energy saving policies are implemented to make the queries at the lowest power consumption. SINA has a cluster-based architecture.

- **AutoSec [~~21~~]:** provides access to sensor network's resources for control and data. This implementation takes into account the quality of service of each query launched to the system, adapting the data collecting strategies to the sensor network.
- **DSWare [~~33~~]:** very similar to AutoSec, is capable of maintaining clusters of nodes and obtaining information from them transparently to provide reliable measurements regardless of network failures.
- **IMPALA [~~33~~]:** proposes asynchronous event-based methods for WAdSN to implement a multi-agent system. Agents can move through the network depending on the system's requirements.
- **MILAN [~~22~~]:** designed to help in the development of portable applications, its applications are capable of adapting to changes in network topology and energy constraints. It supports a cluster-based architecture where the system itself identifies the clusters to meet the application's requirements, the energy consumption and the desired bandwidth. MILAN is not only an intermediate layer between applications and network/operating system layer: it extends into the network layer to control certain policies related to obtaining the optimum performance at minimum consumption.
- **Mires [~~52~~]:** implemented under TinyOS/NESC, this middleware offers a message oriented architecture (with publish/subscription to data services) that complies with the traditional middleware for large and distributed systems. Adaptations for WAdSN are made, such as routing features and data aggregation.
- **COCA [~~15~~]:** Cross-Layer Cooperative Caching is a cluster-based middleware that provides data caching and management services to the applications in Ad-Hoc networks. Although not specifically oriented towards WAdSN, it shows how the very important issue of data caching could be implemented in a cooperative way, and many of their results may also be useful for WAdSNs.

As we can see, there are several solutions for implementing a middleware in a WAdSN. However, there is still work to be done [~~29~~], [~~58~~]. Middleware solutions have no solid definition of functionality. As we saw above, the extension of the middleware layer and the functionalities of each one are different and it is not possible to extract a common set of functionalities from all of them. Support for inclusion of time and location of measurement extracted from the environment is not present in these implementations and this is a very important feature within this type of network. Proactivity has been widely considered and marks a trend to follow for future implementations. The whole system must take actions before an important event occurs. It must look for changes in the network topology, QoS, and other aspects to adapt the whole system before things go wrong.

However, one of the most interesting advances in the middleware systems of the future will be the increased capabilities of future sensor nodes. Moore's Law guarantees growths in speed, resources and cost. In the future it may be possible to apply middleware solutions currently used in large systems to these nodes, making the absolute integration of wireless networks within Internet a near reality, an idea that has been pursued for several years, since the researchers at MIT defined the term The Internet of Things.

4.3.2 Multi-Agent Systems in WAdSN

In the previous section we looked at the advantages of using a middleware framework: Easy to program in several hardware platforms at a time; abstraction of network and operating system services; easy to discover and use resources scattered in the network and (desirably for WAdSN middleware) control and automatic adaptation of the whole system to network constraints. Middleware therefore, empowers the capabilities of systems that use WAdSN together with other hardware platforms, integrating WAdSN nodes in a unique distributed system and making it possible for programs (or services) operating within the system to run, communicate and use all the system's resources regardless of which hardware is being used and which network technology has to be used to communicate with them.

The only diffculty with these frameworks is to obtain an implementation lightweight enough to be fitted into the very constrained hardware of a WAdSN node. If we may overcome this, the following step is to include certain degree of intelligence in WAdSN.

Knowing that an agent is an entity located in one environment, capable of performing flexible and autonomous actions to achieve its goals [~~27~~], how can this concept be applied to WAdSN? Initially, the application of agents was a technology designed to implement a kind of 'life' in a system to solve complex algorithms; like in a machine with lots of small programs coexisting with each other. The rules for interacting, surviving and evolving in this 'world' led to the solution of the problem in a collaborative fashion. The following evolution in this technology was to scatter agents between various machines, creating a system consisting of several machines, each containing one or more agents. In both cases, we are talking about 'multi-agent system' (MAS) but the more recent evolution allows the migration of agents between machines to meet the system's computational requirements (Figure 4.3).

If these machines all have the same hardware and software architecture, the only remaining problem for the implementation of multi-agents is the issue of establishing the migration and communication mechanisms between machines. But, when heterogeneity in hardware and software exists, the middleware framework becomes a necessary solution for multi-agent systems technology. Several middleware solutions provide a hardware abstraction and a common language for defining and describing applications regardless of hardware implementation so, with one of them, the creation of a multi-agent system is solved. However, implementing mobility in agents requires another type of capability that the majority of middleware solutions (except for IMPALA) do not have. Every agent system requires a platform providing an identification service, mobility, communication, and so on.

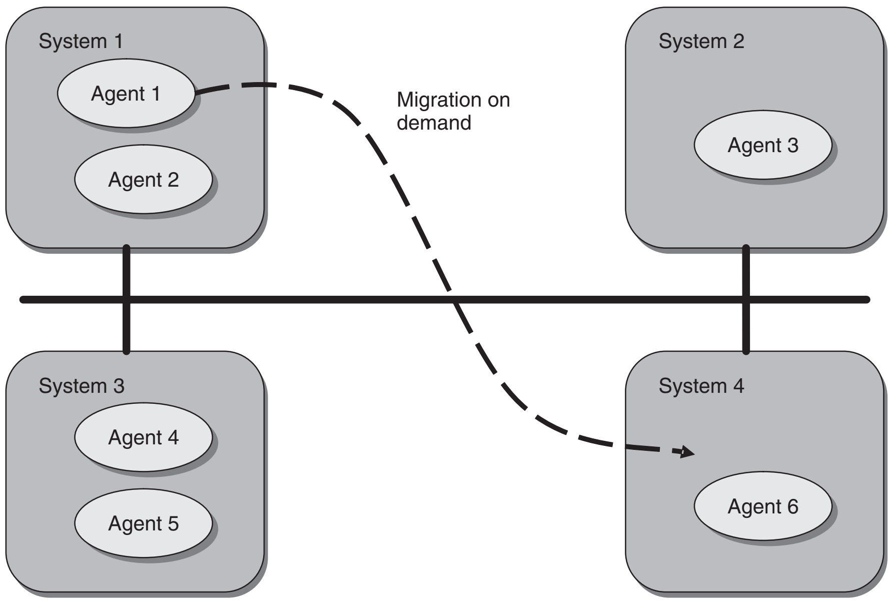

Figure 4.3 The concept of multi-agent system.

This approach is implemented in a WAdSN through a set of agents that reside in the nodes of the WAdSN [~~45~~]. These agents can interact with other agents and gather information about the environment (for example: try to classify the information) or another important aspect of the system (for example: they can be aware of routing tasks in the WAdSN to minimize energy consumption). They can take actions based on the information gathered, communicate with other agents in the system, and move from one sensor to another if required. These agents and their actions are supported by a middleware architecture that guarantees [~~39~~], [~~5~~]:

- Local services: agents can access local resources through the platform.
- Agent discovery and maintenance: detection of agent in the neighbourhood. Also, agents must be initiated and stopped when required.
  - Agent = code + data + state + meta-information.
- Common language: for defining and describing agents.
- Reactivity: capability to detect and react to changes in the environment.
- Asynchronous operation: agents must react to changes in the environment. This operation mode can save energy in WAdSN.
- Autonomy: agents can control their own actions.
- Communications: with other agents, regardless of where they are.
- Collaboration and cooperation: agents must be capable of processing information provided by other agents.
- Mobility: capability to migrate between nodes.

This set of features seems quite large for a WAdSN node. There are implementations for multi-agent systems that operate in medium-large machines (as in middleware, the set of protocols and functionalities required is large and so, therefore, is the footprint).

FIPA [~~2~~] defined a reference architecture for MAS but it is too large to be included in WAdSN. However, this architecture must be taken into account because it is the base of some 'standardized' frameworks of MAS. LEAP is an example of one of these frameworks. It is a lightweight version of FIPA architecture but not light enough for WAdSN because it is implemented in Java.

Another implementation is the above-mentioned IMPALA [~~33~~], which enables application modularity, adaptability to network failures and reparability in WAdSN under an asynchronous operation mode. Agilla [~~18~~] is a framework that supports code mobility of agents written in a high-level code. Data communications are made based on a shared tuple space. MASIF [~~3~~] is a set of interfaces for promoting interoperability and mobility of agents in distributed systems but it cannot be implemented in WAdSN because of its large footprint and resource consumption (it is based on CORBA).

4.3.3 Artificial Neural Networks in WAdSN

Nowadays, neural networks cover a wide spectrum of applications: they can be used when no algorithm exists to solve a specific task, but the solution can be expressed on the basis of a rich set of input examples. Another application area of artificial neural networks (ANN) is the classification of information and pattern recognition (as in weather forecasting, etc.) or when algorithms are incapable of solving a problem. There are several reasons for using ANN: high robustness against noise, malfunctions or even failure; easy to parallelize and distribute; and lightweight.

Basically, the concept of artificial neuron is an entity whose output is defined as a function of a set of weighted inputs (from each of the inputs, see Figure 4.4). The weights of each input define the interaction between neurons so, with a low weighted input coming from a neuron,

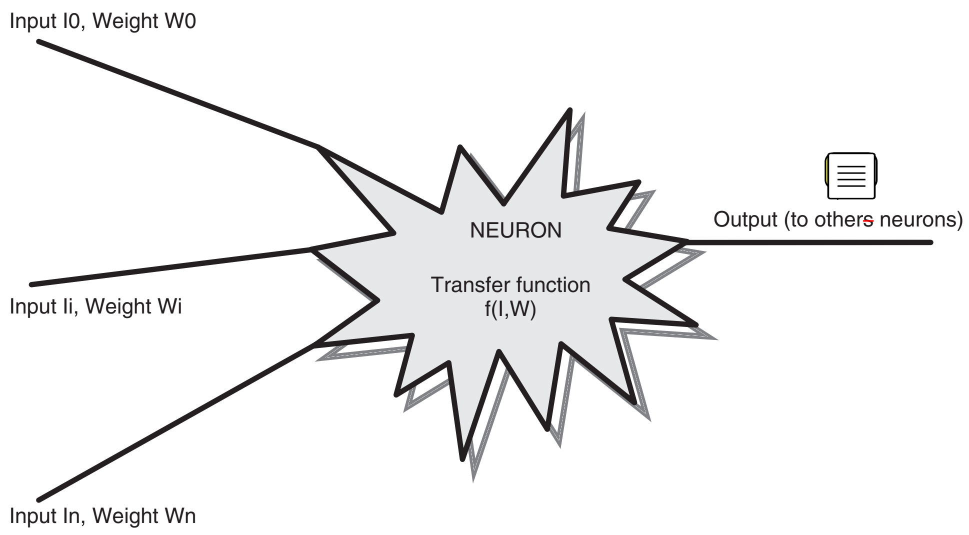

Figure 4.4 Model of an artificial neuron.

it seems that there is almost no interaction with it. This output can be codified as a number within a range (for example: 0–1) or as a series of spikes distributed in time. In the latter case, we are talking about Spiking Neurons [~~40~~], [~~19~~]. The frequency of the spike train or the space between spikes (these are two methods of coding outgoing information in this type of neural network) defines the neuron's excitation level, or state. Of course, the same coding system is also used in the neuron's inputs. Therefore, the output function is defined as a weighted function of neuron's incoming pulse trains, allowing the building of layers of interconnected neurons to work together. Spiking neural networks is a more realistic way of implementing a neural system because the way this type of neuron runs is closer to the natural entities that they try to imitate.

Neurons are organized in layers and usually, one neuron from layer i has an input to all neurons in layer $i - 1$. In this case, we can talk about feed-forward ANN (Figure 4.5), where information is spread from input to outputs. In the event of any layer i spreading information to layer $i - k$; $k > 0$, the ANN can be classified as a recurrent ANN (Figure 4.6). In an ANN, there must be an input layer and a (sometimes is the same) output layer. Multiple layers can

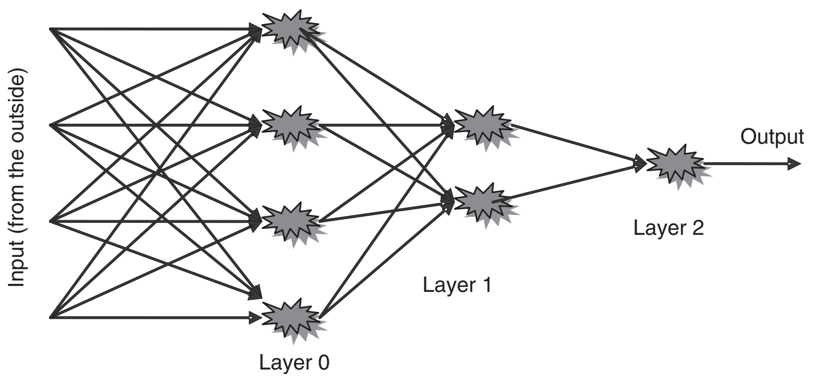

Figure 4.5 Example of a 3-layer feed-forward ANN.

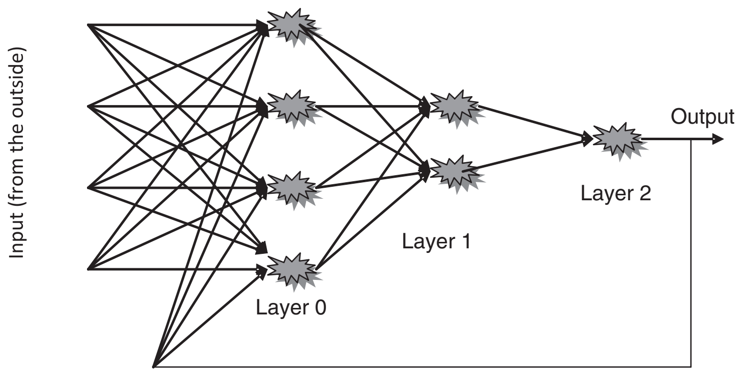

Figure 4.6 Example of a 3-layer recurrent ANN.

be placed between them, forming a multilayer ANN. The stage prior to operating with an ANN is the learning procedure. During this stage, the weight of each neuron takes its final value through a process in which a known input is presented to the ANN and the weights of neurons are modified to achieve the desired output at ANN's output layer. There are several classes of learning procedures: supervised, not supervised and reinforcement. When the ANN has finally been configured, each ANN neuron executes its output function only when there is any change in input values. This algorithm is usually simple and fast.

How can an ANN be implemented in a WAdSN? Bearing in mind the similarity between neurons and sensor nodes, one may think that the only way to implement an ANN is to assign one neuron to each node in the network. This way, the neurons implemented in the WAdSN will be the neurons of the input layer. Further layers could be implemented in a more powerful network, where the information processing can be finished without exhausting the resources of the WAdSN. The ANN built in this way would be valid for pattern recognition, data classification or predictions from the input data. An example of this kind of implementation can be seen in [~~24~~].

Another type of implementation of ANN in WAdSN is to keep a complete ANN within each network node. This way, each network node has a trained ANN to solve autonomously a specific problem related to some aspect of the WAdSN. For example, [~~9~~] use this type of implementation to solve the problem of routing optimization. In [6] ANN and genetic algorithms are used to solve two important problems in WAdSN: sensor self-calibration and the insertion of new nodes in runtime. Basically, each node has a complete ANN that can be trained by a genetic algorithm once it is inserted in the network. The genetic algorithm is used to contrast the output of the ANN (that classifies the outputs of the sensor) with an ideal value extracted from a lookup table. This way, a node can be inserted at anytime in the network and the genetic algorithm will adjust the ANN of this node based on its inputs. Finally, spiking neural networks are the subject of intense research efforts to implement bio-inspired devices where information flows in spike trains transported by AER buses (Address Event Representation). One example of these kinds of devices can be found in [~~50~~]. In this project, the chain of bio-inspired devices is connected by wire and instead of focusing on low power consumption the system has been designed as a prototype to demonstrate the suitability of these devices for certain tasks. In [~~60~~], [~~4~~] neuro-inspired devices and wireless sensor networks are combined to offer a fast and low power detection device (each one implemented by spiking neural networks) and a persistent and low-cost form of communication (each WAdSN node is responsible for its sensor communicating with a central node that collects and processes all the information provided by the sensors) that enables the system to work for long periods of time without maintenance.

4.4 Final Remarks

Throughout this chapter cooperation in wireless ad-hoc and sensor networks (WAdSN) has been shown as a collaborative action which network nodes are involved in. According to the Open System Interconnection model, OSI, proposed by the International Organization for Standardization, ISO, the development of cooperation could be understood as being located between the application and low level layers (physical and link layer).

Time synchronization, localization and calibration (SLC) are services that cooperation must provide. Consequently, there needs to be some mechanism to allow the provision of these services. As explained in Section 4.3.1, this mechanism works as a container of cooperation services, allowing their implementation in a consistent manner. These services may include not only SLC, but also data caching, data management, etc. With middleware, cooperation services can be implemented and accessed by the application regardless of the hardware or network technology being used.

The use of Multi-Agent Systems (MAS) is another approach to network cooperation. Several implementations of MAS over WAdSN are being carried out by different research groups, as explained in Section 4.3.2. In some ways MAS could also be understood as a middleware.

Middleware and multi-agent systems are concepts that have been created bearing in mind other kinds of networks. For this reason current implementations of them are complex and hardly feasible over ad hoc and sensor networks. However, the adaptation of MAS and middleware to these special networks will change the current conception of cooperation in WAdSN. Conceptually it is possible to adapt them. So, we are faced with the challenge of adapting current solutions to the case of resource and energy limited nodes that communicate in an ad-hoc fashion, providing at the same time scalability, reliability and dynamic adaptation to the environment.

4.5 Acknowledgements

This work was supported by contracts TIN2006.15617.C03.03, **AmbienNet**: Ambient Intelligence Supporting Navigation for People with Disabilities; P06TIC-2298, **SemiWheelNav**: External sensing based semiautonomous wheelchair navigation; P07-TIC-02476, **ARTICA**: Wireless Sensor Networks and Computational Intelligence in Environmental Applications, as well as by the Telefonica Chair ‘Intelligence in Networks’ of the University of Sevilla (Spain).

References

[~~1~~] Doñana Biological Stations. http://www.ebd.csic.es/website1/Principal.aspx.

[~~2~~] Foundation for Intelligent Physical Agents. http://www.fipa.org.

[~~3~~] Object Management Group. http://www.omg.org.

[~~4~~] http://pantheon.yale.edu/~dk6/TDsensor.html.

[~~5~~] IEEE/FIPA WG Mobile Agents. Ulrich Pinsdorf. Presentation of WG Mobile Agents Group, in FIPA. 2005.

[~~6~~] R. Abielmona, V. Groza and W. Pretiu. Evolutionary neural network network-based sensor self-calibration scheme using IEEE 1451 and wireless sensor networks. In International Symposium on Computational Intelligence for Measurement Systems and Applications, CIMSA 2003, pages 38–43, Lugano, Switzerland, July 2003.

[~~7~~] I. Akyildiz, Y. Su, W. Sankarasubramaniam and E. Çayirci. *Wireless sensor networks: A survey. Computer Networks*, Elsevier, 38: 393–422, December 2002.

[~~8~~] C. Aurrecoechea, A. Campbell and L. Hauw. *A Survey of QoS Architectures. Multimedia Systems*, Springer-Verlag, 6: 138–151, 1998.

[~~9~~] J. Barbancho, C. León, F. Molina and A. Barbancho. Using artificial intelligence in routing schemes for wireless networks. *Computer Communications*, 3842: 2802–2811, June 2007.

[~~10~~] P. Baronti, P. Pillai, V. Chook, S. Chessa, A. Gotta, and Y. Fun Hu. Wireless sensor networks: A survey on the state of the art and the 802.15.4 and Zigbee standards. *Computer Communications*, 30(3): 1655–1695, 2007.

[~~11~~] P. Bonnet, J. Gehrke and P. Seshadri. Querying the physical world. *IEEE Personal Communication*, 7(10), October 2000.

[~~12~~] N. Bulusu, J. Heidemann and D. Estrin. GPS-less low cost outdoor localization for very small devices. *IEEE Personal Communications Magazine*, 7(5): 28–34, 2000.

[~~13~~] V. Bychkovskiy, S. Megerian, D. Estrin and M. Potkonjak. A collaborative approach to in-place sensor calibration. In Proceedings of the 2nd International Workshop on Information Processing in Sensor Networks (IPSN'03), Berkeley, CA, USA, April 2004.

[~~14~~] R. Casas, A. Marco, J. Guerrero and J. Falcón. Robust estimator for nonline-of-sight error mitigation in indoor localization. In *EURASIP Journal on Applied Signal Processing*, pp. 1–8, 2006.

[~~15~~] M. Denko, J. Tian, T. Nkwe and M. Obaidat. Cluster-based cross-layer design for cooperative caching in mobile ad hoc networks. *IEEE Systems Journal*, 3(4): 499–508, 2009.

[~~16~~] J. Elson and D. Estrin. Time synchronization for wireless sensor networks. In *International Parallel and Distrib. Processing Symp.*, San Francisco, CA, USA, 2001.

[~~17~~] J. Elson, L. Girod and D. Estrin. Fine-grained time synchronization using reference broadcasts. In *Proc. 5th Symp. Op. Sys. Design and Implementation*, pp. 147–163, Boston, MA, USA, 2002.

[~~18~~] C. Fok, G.-C. Roman, and C. Lu. Rapid development and flexible deployment of adaptive wireless sensor network applications. Technical Report WUCSE-04-59, Washington University, Department of Computer Science and Engineering, St. Louis, USA, 2004.

[~~19~~] W. Gerstner and W. Kistler. *Spiking Neurons Models. Single Neurons, Populations, Plasticity*. Cambridge University Press, 2002.

[~~20~~] D. Hall and S. McMullen. *Mathematical Techniques in Multisensor Data Fusion*. Artech House Publishers, 2004.

[~~21~~] Q. Han and N. Venkatasubramanian. Autosec: An integrated middleware framework for dynamic service brokering. *IEEE Distributed Systems Online*, 2(7), 2001.

[~~22~~] Heinzelman, W.B. Middleware to support sensor network applications. *IEEE Network*, 18(1): 6–14, 2004.

[~~23~~] Y. Hong and A. Scaglione. A scalable synchronization protocol for large scale sensor networks and its applications. *IEEE Journal on Selected Areas in Communications* (JSAC), 23(5): 1085–1099, 2005.

[~~24~~] Q. Huang, T. Xing and H. T. Liu. Vehicle classification in wireless sensor networks based on rough neural network. *Lecture Notes in Computer Science*. Springer-Verlag: Berlin Heidelberg, pp. 58–65, 2006.

[~~25~~] R. Iyer and L. Kleinrock. QoS control for sensor networks. In IEEE International Conference on Communications, ICC'03, volume 1, pp. 517–521. IEEE Press, May 2003.

[~~26~~] C. Jaikaeo, C. Srisathapornphat and C. C. Shen. Querying and tasking in sensor networks. In International Symposium on Aerospace/Defense Sensing, Simulation, and Control (Digitization of the Battlespace V). In SPIE's 14th Annual, pp. 24–28, Orlando, FL, USA, April 2000.

[~~27~~] N. Jennings, K. Sycara and M. Wooldridge. A roadmap of agent research and development. *Autonomous Agents and Multi-Agent Systems*. Kluwer Academic Publishers, 1: 7–38, 1998.

[~~28~~] S. Jin, M. Zhou and A. S. Wu. Sensor network optimization using a genetic algorithm. In Proceedings of the 7th World Multiconference on Systemics, Cybernetics, and Informatics, Orlando, FL, USA, July 2003.

[~~29~~] K. K. Römer, O. Kasten and F. Mattern. Middleware challenges for wireless sensor networks. In *Proceedings of the ACM SIGMOBILE Mobile Computing and Communication Review (MC2R)*, 6(4), pp. 59–61, October 2002.

[~~30~~] J. Kay and J. Frolik. Quality of service analysis and control for wireless sensor networks. In 2004 IEEE International Conference on Mobile Ad-hoc and Sensor Systems, pp. 359–368. IEEE Press, October 2004.

[~~31~~] M. Kohvakka, M. Kuorilehto, M. Hännikäinen and T. D. Hämäläinen. Performance analysis of IEEE 802.15.4 and Zigbee for large-scale wireless sensor network applications. In Proc. Third ACM International Workshop on Performance Evaluation of Wireless Ad Hoc, Sensor, and Ubiquitous Networks, pp. 48–57, Málaga, Spain, 2006.

[~~32~~] B. Krisnamachari, D. Estrin and S. Wicker. Modelling data-centric routing in wireless sensor networks. In Proceedings of INFOCOM 2002, New York, USA, June 2002.

[~~33~~] S. Li, S. Son and J. Stankovic. Event detection services using data service middleware in distributed sensor networks. In Proceedings of the 2nd International Workshop on Information Processing in Sensor Networks, April 2003.

[~~34~~] M. M. Rabbat and R. Nowak. Distributed optimization in sensor networks. In Proceedings of the 3rd International Symposium on Information Sensor Networks (IPSN), Berkeley, CA, USA, April 2004.

[~~35~~] M. Mao, B. Fidan and B. Anderson. Wireless sensor network localization techniques. *Computer Networks*, 51(4): 2529–2553, 2007.

[~~36~~] A. Marco, R. Casas, J. Falco, H. Gracia, J. Artigas and A. Roy. Location-based services for elderly and disabled people. *Computer Communications*, 31(4): 1055–1066, 2008.

[~~37~~] A. Marco, R. Casas, J. Sevillano, V. Coarasa, J. Falcón and M. Obaidat. Multi-hop synchronization at the application layer of wireless and satellite networks. In Proc. IEEE Global Communications Conference (IEEE GLOBECOM 2008), pp. 1–5, New Orleans, LA, USA, December 2008.

[~~38~~] M. Maróti, B. Kusy, G. Simon and A. Lédeczi. The flooding time synchronization protocol. In Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems, pp. 39–49, Baltimore, MD, USA, 2004.

[~~39~~] J. Martínez, A. García, A. Sanz, L. López, V. Hernández and A. Dasilva. An approach for applying multi-agent technology into wireless sensor networks. In Proceedings of the 2007 Euro American Conference on Telematics and information Systems, Faro, Portugal, Mayo 2007. EATIS, ACM.

[~~40~~] W. Mass and C. M. Bishop. *Pulsed Neural Networks*. The MIT Press, 2001.

[~~41~~] D. S. Niculescu and B. Nath. Dv based positioning in ad hoc networks. *Telecommunication Systems*, 22(1-4): 267–280, 2003.

[~~42~~] S. PalChaudhuri, A. Saha and D. Johnson. Adaptive clock synchronization in sensor networks. In Proceedings of the 3rd International Symposium on Information Sensor Networks (IPSN), pp. 340–348, Berkeley, CA, USA, April 2004.

[~~43~~] M. Perillo and W. Heinzelman. Sensor management policies to provide application QoS. *Ad Hoc Networks*, 1: 235–246, 2003.

[~~44~~] G. Pottie and W. Kaiser. *Principles of Embedded Networked Systems Design*. Cambridge University Press, 2005.

[~~45~~] H. Qi, Y. Xu, and X. Wang. Mobile-agent-based collaborative signal and information processing in sensor networks. In Proceedings of the IEEE, 8(91), August 2003.

[~~46~~] V. Rakocevic, M. Rajarajan, K. McCalla and C. Boumitri. QoS constraints in bluetooth-based wireless sensor networks. *Lecture Notes in Computer Science*, Springer Verlag, 3266: 214–223, 2004.

[~~47~~] K. Römer, P. Blum and L. Meier. *Handbook of sensor networks: algorithms and architectures*. John Wiley & Sons Ltd, 2005.

[~~48~~] B. Sabata, S. Chatterjee, M. Davis, J. Sydir and T. Lawrence. Taxonomy for QoS specifications. In Proceedings of the Third International Workshop on Object-Oriented Real-Time Dependable Systems, pp. 100–107. IEEE Press, 1997.

[~~49~~] A. Scaglione and S. Servetto. On interdependence of routing and data compression in multi-hop sensor networks. In Proceedings of the 8th Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom'02), Atlanta, Georgia, 2002.

[~~50~~] R. Serrano-Gotarredona, M. Oster and P. Lichtsteiner. Caviar: A 45kneuron, 5m-synapse, 12g-connects/sec AER hardware sensory-processing-learning-actuating system for high speed visual object recognition and tracking. *IEEE Transactions on Neural Networks*, 20(9): 1417–1438, September 2009.

[~~51~~] P. Sheu, S. Chien, C. Hu and Y. Li. An efficient genetic algorithm for the power-based QoS many-to-one routing problem for wireless sensor networks. In International Conference on Information Networking, ICOIN 2005, pp. 275–282, Jeju, Korea, February 2005.

[~~52~~] E. e. a. Souto. Mires: A publish/subscribe middleware for sensor networks. *Personal Ubiquitous Computing*, 10(1), 2005.

[~~53~~] B. Sundararaman, U. Buy and A. Kshemkalyani. Clock synchronization for wireless sensor networks: A survey. *Ad Hoc Networks*, 3: 281–323, 2005.

[~~54~~] A. Talekder, R. Bhatt, S. Chandramouli, L. Ali, R. Pidva and S. Monacos. Autonomous resource management and control algorithms for distributed wireless sensor networks. In The 3rd ACS/IEEE International Conference on Computer Systems and Applications, 2005, pp. 19–26, Cairo, Egypt, Enero 2005.

[~~55~~] T. Vercauteren, D. Guo and X. Wang. Joint multiple target tracking and classification in collaborative sensor networks. *IEEE Journal on Selected Areas in Communications*, 23(4), April 2005.

[~~56~~] Y. Xu, J. Heidemann and D. Estrin. Geography informed energy conservation for ad hoc routing. In Proc. of the 7th Int. Conf. on Mobile Computing and Networking (MobiCom 2001), pp. 70–84, Rome, Italy, July 2001.

[~~57~~] W. Ye, J. Heidemann and D. Estrin. Medium access control with coordinated adaptive sleeping for wireless sensor networks. *IEEE/ACM Transactions on Networking*, 12(3), 2004.

[~~58~~] Y. Yu, B. Krishnamachari and V. Prasanna. Issues in designing middleware for wireless sensor networks. *IEEE Network Magazine*, 12(3), January 2004.

[~~59~~] K. Zen, D. Habibi, A. Rassau and I. Ahmad. Performance evaluation of IEEE 802.15.4 for mobile sensor networks. In 5th IFIP Int. Conf. on Wireless and Optical Communications Networks, WOCN'08, pp. 1–5, May 2008.

[~~60~~] F. Zhengming, E. Culurciello, P. Lichtsteiner and T. Delbruck. Fall detection using an address-event temporal contrast vision sensor. In IEEE International Symposium on Circuits and Systems, 2008. ISCAS 2008, pp. 424–427, May 2008.