Information extraction from large-scale WSNs: approaches and research issues: approaches and research issues: part I: overview and agent based approaches

Gaura, E. and Daniel, T.

Sensors & Transducers Journal, Vol. 94, Issue 7, July 2008, pp. 15-33

www.sensorsportal.com

Sensors & Transducers

ISSN 1726-5479

http://www.sensorsportal.com

Information Extraction from Large-scale WSNs: Approaches and Research Issues
Part I: Overview and Agent Based Approaches

Elena GAURA, Tessa DANIEL
Cogent Computing Applied Research Centre, Coventry University, UK CV1 5FB
E-mail: e.gaura@coventry.ac.uk, www.cogentcomputing.org

Received: 29 June 2008 /Accepted: 21 July 2008 /Published: 31 July 2008

Abstract: Regardless of the application domain and deployment scope, the ability to retrieve information is critical to the successful functioning of any wireless sensor network (WSN) system. In general, information extraction procedures can be categorized into three main approaches: agent-based, query-based and macroprogramming. Whilst query-based systems are the most popular, macroprogramming techniques provide a more general-purpose approach to distributed computation. Finally, the agent-based approaches tailor the information extraction mechanism to the type of information needed and the configuration of the network it needs to be extracted from. This suite of three papers (Part I-III) offers an extensive survey of the literature in the area of WSN information extraction, covering in Part I and Part II the three main approaches above. Part III highlights the open research questions and issues faced by deployable WSN system designers and discusses the potential benefits of both in-network processing and complex querying for large scale wireless informational systems. *Copyright © 2008 IFSA.*

Keywords: Information extraction, Mobile agents, Wireless sensor network architectures

1. Introduction and Domain Overview

The increase in availability and affordability of wireless technology has led to a proliferation of large wireless sensor networks (WSNs) with increasing numbers of nodes deployed to resolve complex "informational" problems. Typically, however, nodes in these networks have limited resources including energy (given the limited battery power), memory, computing power and communication bandwidth. The coupling of high complexity global tasks and the reduced resource nodes make WSNs

a rich research domain. The use of WSNs has been explored in a number of application domains with deployments ranging from scientific research to battlefield surveillance for the military. Some examples of deployments include habitat monitoring applications [1, 2]; agricultural monitoring [3]; healthcare [4]; disaster management and detection [5] and military applications [6]. Moreover, some newer applications involve the integration of multiple WSN systems spanning a variety of disciplines. The High Performance Wireless Research and Education Network [7] and ROADNet [8] projects, both in California, are two examples of such multidisciplinary applications currently in operation.

Although specific application requirements may differ from one WSN system to another, essential to all WSN systems is the ability to acquire data and convert this to usable information in order to fulfill the application needs. Many of the WSN information extraction approaches currently in operation are based on applicative query mechanisms where requests are initiated via queries written in an appropriate declarative language, posed to the network, and data or information generated as a result. Data processing in such applications usually follows one of two approaches: centralized or distributed. For some systems, resources are not as severely constrained, hence, minimizing energy usage is not a major concern. In such unconstrained systems, a centralized approach to processing is often used where sensed values are pushed to a power-rich location for processing, which may involve cleaning and querying the data as part of more in-depth analysis.

In a large number of systems, however, constraints like computing and battery power dictate that applications be developed with energy-efficiency in mind. For many of these applications human intervention can be both time consuming and expensive, for example, in terms of changing batteries or adding new nodes. Therefore, a key design objective is to extend the lifetime of the network as much as possible. It is well-understood that power usage costs in WSNs are dominated by communication as opposed to computation [9, 10, 11]. Therefore, techniques which promote a decrease in communications while using in-network data reduction have been identified as key to creating more energy-efficient WSN applications. Distributed processing is proposed as a method that promotes processing of data on nodes in the network [10, 12]. This in-network processing takes advantage of nodes' processing power and may include techniques like data aggregation, fusion or elimination of redundant values through filtering [13, 14]. The net result is more energy-efficient applications since data transmission is reduced in exchange for in-network computation. Whilst this is apparently of obvious benefit towards long -life systems design, in-network computation is still in its infancy.

Existing information extraction procedures (whose survey is at the core of this suite of papers) can be categorized into three main approaches: agent-based, query-based and macroprogramming. Of the three, *query-based* systems are the most popular mainly because they provide a usable, high level interface to the sensor network while abstracting away some of the low level details like the network topology and radio communication. They are very useful and provide a solution in cases where data needs to be retrieved from the entire network. With this ease of use, however, come a number of limitations.

The first limitation is in terms of what queries can be posed to the network. In general, the query-based applicative systems in use allow the issuing of restricted queries, ranging from those targeting raw sensor readings on nodes in the network to those requiring the computation of simple aggregates like average, maximum, and minimum over some attribute of interest, for the entire network. Second, the query languages used cannot easily express spatio-temporal characteristics which are an important aspect of the data generated in WSNs. Third, it is quite difficult if not impossible to construct information requests that represent higher level behaviour, or involve just a "subset" of the network (whether physical or logical) or require more complex in-network interactions between "subsets" in order to generate information. Furthermore, as distributed computation is not the main focus of SQL-based query languages (which support most WSN query-based approaches) implementing arbitrary aggregation, for example, is quite difficult [15].

In contrast, *macroprogramming* has been proposed as an approach to information extraction that provides a more general-purpose approach to distributed computation compared to traditional query-based approaches. As applied to WSNs, macroprogramming approaches focus on programming the network as a whole rather than programming the individual devices that form the network. Many macroprogramming systems provide the ability to create programs that represent higher level behaviour, a level of abstraction beyond that of the more popular query-based approaches. Global behaviour can be specified, programmed and then translated to node level code. Ideally, the programmer is not concerned with low level details like network topology, radio communication or energy capacity.

Of interest with some macroprogramming systems are the application-defined, in-network abstractions (some based on local node interactions) that are used in data processing. One example is the Regiment system [16] in which a programming abstraction called a region is used. A region is described as a collection of spatially distributed signals with an example being the set of sensor readings from nodes in a geographic area. Regions as opposed to individual nodes are programmed (for example, an **rfold** operator is used to aggregate the values in a region into a single signal which can then be communicated to the user or used in further computation).

Macroprogramming, however, still presents a number of challenges. First, creating a powerful macroprogram requires a learning curve for the programmer. Expressing high-level requirements are not necessarily as intuitive to the user as perhaps SQL-based approaches are. Second, although proposed as an approach that eliminates the need for node-level programming, many of the current macroprogramming systems provide node-specific abstractions, undermining the rapid development and productivity advantages macroprogramming is meant to provide. Finally, because of the wide semantic gap that exists between the high level program and the node level code, compiler construction is quite challenging. The code generated as a result of compilation has to cater for not just computation but node-level communication as well.

A third approach to information extraction looks at the extracting of information in a network-aware manner and tailors the mechanism to the type of information needed and the configuration of the network it needs to be extracted from. These models use *agents* to perform tasks, make decisions and collaborate to achieve more complicated tasks. An agent is simply a piece of software that performs a task without the need of user invocation to function. Many agent-based models are multi-agent systems in which multiple agents sometimes coded to function in different ways, collaborate, coordinate and organize to perform complex tasks and generate information. These models therefore introduce expressiveness and flexibility in terms of what functions they can facilitate in the network as well as in facilitating the ability for agents to make decisions while in the network. Agents can act autonomously, multiple agents can run on a node at the same time, and multiple applications can co-exist in the network. Mobile agents can move, clone themselves and act to deal with unexpected changes in the environment [17, 18].

Although attractive, in theory, the agent-based approach presents a number of challenges particularly in terms of the difficulty they present for non-expert users to design and program. First, given the complexity of distributed systems, it is a challenge to model accurately what interactions will be taking place between components in the network when designing agents. Second, when implementing the agents, particularly mobile agents, the code must be able to run in unpredictable, resource-constrained environments. Forecasting accurately exactly what conditions will exist in the network is difficult and therefore difficult to program for. Third, there is difficulty in testing and debugging, given the distributed nature of the system. In mobile agent systems, tracking and understanding execution in the face of agents moving from node to node makes debugging and testing even more of a challenge. Given these difficulties, practical implementations of agent-based systems, particularly multi-agent

and mobile agent systems, are not as widespread as query-based systems are. Each of the three approaches above is suitable for a number of applications. They each have strengths but pose challenges as well. The agent-based approach provides a high degree of expressiveness and flexibility as do the macroprogramming approaches, but they both are more difficult to implement into deployable WSN systems. The query-based systems have, as their key feature, ease of use but are limited in the types of queries that can be posed to the network. Macroprogramming tries to address this by providing powerful constructs for capturing higher level behaviour through global programs but introduce a steep learning curve to the user.

The authors here argue that a hybrid approach that retains the simplicity and ease of use of the more traditional query-based approaches while allowing the inclusion of useful logical abstractions provided by macroprogramming approaches to facilitate construction and resolution of more powerful queries would be indeed desirable. Such an approach would need to incorporate some of the principles of agent-based systems, such as collaboration and decision making in the network, in assisting in query resolution. In this context it can be hypothesized that end-users of WSN applications could be provided with the ability to construct and pose higher level information requests instead of simple queries requiring the collection of raw sensed values or the calculation of simple aggregates over the entire network. Complex query type constructs that allow the expression of phenomenological spatio-temporal characteristics would provide the means for exploiting the very core of the networked sensing concept at large scale. Moreover, providing the end-user with a system that produces responses to these higher level requests for information within the network instead of as a result of post-collection analysis of all data would most certainly demolish one of the largest road-blocks of the WSN technology in its route to adoption: ensuring user acceptability.

This suite of three papers is focused as follows: Part I concentrates on Agent based approaches to information extraction; Part II surveys query based system and macroprogramming approaches and Part III looks at the open research issues the community faces with respect to information extraction, raises research question relating to the usefulness of in-network processing from an informational viewpoint and concludes the survey. In both parts I and II, example developments are identified, highlighting those that have been used in practical deployments. Wherever possible the architecture and mode of information processing (whether centralized or distributed) is identified. The key issues for discussion lay with:

- identifying which of the methods proposed in the literature have been evaluated at implementation level. (This analysis is needed as a large proportion of the research effort is at theoretical level and not readily suitable for practical deployments.)
- identifying which of the methods proposed in the literature make use of or support in-network information extraction. (This is important as approaches that utilize in-network processing like aggregation and filtering have been shown to be more energy efficient and therefore more desirable.)

2. Agent-based Approaches and Architectures

The agent-based approach to information extraction tasks a network by injecting into it a program with some type of processing or decision making function. In this approach there is no attempt to devise ways of transmitting data to ’collection’ or ’aggregation’ points for processing, rather, the *application* is sent *to* the data instead [19]. The agent is able to collect local data and perform any necessary data aggregation. Autonomous agents can make decisions without user input. In the case of an autonomous agent, which also happens to be mobile, that decision may involve whether the agent should move to a particular node, and if yes, which node it should migrate to. Autonomous mobile agents moving from one node to another can be designed to be cognizant of issues that may exist in the network [20]. For instance, they can make decisions, perhaps to clone themselves, if necessary, to avoid faulty regions or nodes in the network. If an agent detects that a node has a problem and is unreachable it can adapt its

route so that it avoids the faulty node [21]. The mobile agent therefore presents added flexibility in terms of decision making and reliability in dealing with faults in the network.

[22] report three architecture models for agent-based wireless sensor networks. The classification is based on the number and type of agents being used and the mode of operation of the agent-based system.

In the **first model**, all nodes respond to a single agent usually housed at the base station or central processor. The advantage is that there is a single control point with the benefit being the ability to access the entire network and an increase in efficiency since transmissions from this central point could be better synchronized and collision reduced as a result. Collisions during transmission require a retransmission which is quite costly. Therefore, synchronization brings with it direct energy savings. There are, however, a number of limitations to this model.

The first limitation is lack of scalability. In this model, the agent queries each node for a sensed value which forms the input to the agent's deliberation (the agent follows a perceive-deliberate-act cycle). As the number of nodes increases, however, the time taken to deliberate also increases which can lead to time delays. Also, with larger networks it is more likely that multihop communication is needed. This can lead to even more time delays during data gathering as well as increased power consumption.

A second problem is observed in cases where there is a need for more than one agent to access nodes in the network simultaneously, for example, to execute different application tasks. Here an increase in the time delay occurs because one agent would have to wait for another to complete its task before accessing that node or set of nodes. This leads to delays in agents getting the data needed for processing and acting if needed. This model can be considered to utilize a centralized processing approach as all data is transmitted to a central point (albeit an agent) for further processing. In-network processing is absent.

In the **second model**, each node in the network hosts an agent which will determine how that node will behave. Control of the network is therefore distributed and agents can coordinate and collaborate to achieve goals both at a local and global level. A main advantage in using this model is its scalability. The ability to perform local computation without needing multihop messages to a central controller results in a reduction in energy usage. A second advantage is the ability of the network to perform tasks concurrently given the presence of multiple agents. However, in practice, it is difficult to create agent-based systems that solve global tasks using only local information. The result, therefore, is an increase in resource and energy usage for the necessary inter-agent communications, a cost avoided by the centralized model presented above. This multi-agent approach clearly incorporates in-network processing techniques for information generation. The "environmental nervous system", [23] a multi-agent system, is based on this model. It is a small (three node) autonomic wireless sensor network (see Fig. 2) aimed at environmental sensing for intruder detection. This system is described in greater detail in Section 2.2.

The **third model** uses mobile agents. Here an agent is able to migrate to a node or nodes, collecting, aggregating and fusing data before it sends a message back to the central processor. A primary advantage here is a reduction in the cost of computation since only one node is active at a time. A second advantage is that communication cost is low since migration of agents only involves one *transmit* and *receive* event. The agent in this model is responsible for both collecting data and the itinerary related to the task in operation. The main disadvantage, however, is that if the node to which an agent has migrated fails then all data gathered up to that point is lost. There are, however, techniques that could be implemented to combat this problem, for instance leaving copies of the data on previously visited nodes so that the agent could restore its state if needed. A second disadvantage is the lack of concurrency this approach brings. In large networks, in particular, this could be a problem

if the agent takes too long to collect all the data required to perform the given task. The mobile agent-based deployment reported by [17] is based on this model. This network was aimed at ground vehicle classification using acoustic sensors and is described in greater detail in Section 2.1.

Each of the three models described exhibits both advantages and disadvantages. According to [22] the strengths of the three methods were simplicity in the case of the centralized approach, robustness in the case of the multi-agent approach, and efficiency in the case of the mobile agent approach. The weaknesses were identified as scalability, complexity and latency. Two hybrid approaches were further proposed and described by [22] that combined the features of the three in ways that attempted to minimize the impact of the disadvantages described.

Much of the reported research in the area of agent-based information extraction mechanisms for WSNs describes variations of the agent approach outlined above. In the following, some examples of mobile and non-mobile agent system architectures are presented.

2.1. Mobile Agent-based Distributed Sensor Networks (MADSN)

The Distributed Sensor Network (DSN) model was first proposed by [24]. In this model, referred to as a *fusion architecture*, all sensor data is sent to a central location where it is fused [20]. A number of variations followed [25, 26, 27], each making some improvement on the DSN architecture but holding to a common client/server paradigm or a centralized approach to information processing.

In a WSN with a large number of deployed sensors, each with its own piece of data, it is impractical to have all that information flowing to a central store. First, it has been identified that communication dominates power usage in sensor networks and that the cost of transmission of data from individual nodes in a network to a base station or sink far exceeds the cost of in-network processing of data before transmission [28, 29, 11]. With resource constraints like limited battery power on the nodes in the network, there is a need to reduce the cost of communication if at all possible. The energy availability problem is further exacerbated by the fact that often not all data that is transmitted is critical to ensuring quality information. There is often a large quantity of redundant or erroneous data. Other problems include network bandwidth limitations, unreliable connectivity and noise in the network. Given these constraints, DSN and other client/server type architectures cannot meet all the challenges of WSNs. Qi, Iyengar and Chakrabarty proposed an improved DSN architecture that used mobile agents (MADSN) [30, 17]. Fig. 1 shows a comparison between DSN and MADSN.

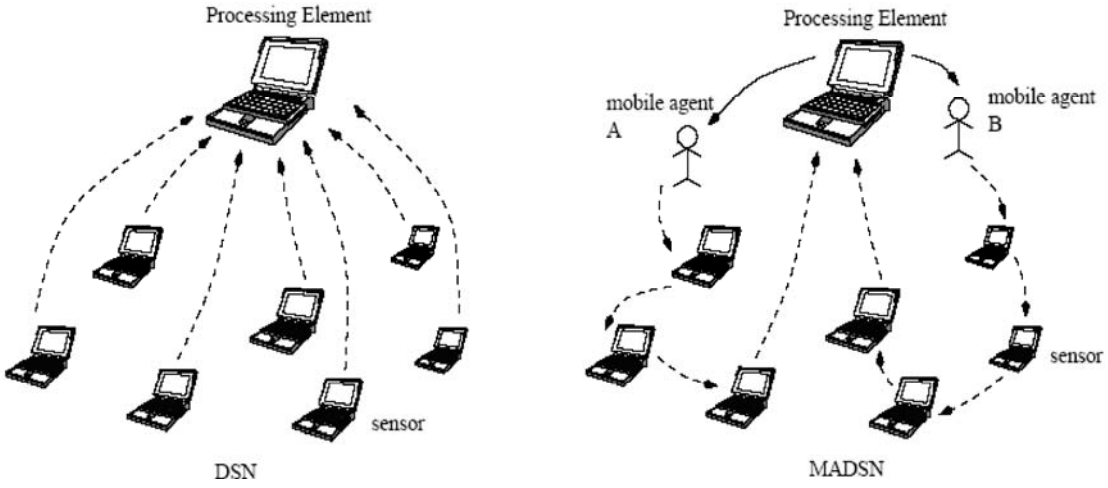

Fig. 1. Comparison between DSN and MADSN architectures taken from [30].

Whereas in DSN sensor nodes collected data and transmitted to the base station node or sink (the central processor), with MADSN the computation code is transmitted to the node where the data resides. The need to transmit masses of data is removed and replaced by transmission of only a small piece of code representing the agent. Additional benefits are that the agents can be programmed to perform fusion tasks to consolidate data increasing the extensibility of the system and the mode of information collection is more stable since it is not affected by fluctuations in connectivity. If a network connection fails, the agent can wait till it is re-established before returning its results.

A comparison between MADSN and DSN showed up to 90% savings in data transfer for MADSN over DSN due to avoiding transfer of raw data [30]. It is not clear however if this was despite the overhead introduced by having agents created and dispatched. In another study, a model that incorporated collaborative signal and information processing (CSIP) techniques was applied and an analysis of performance between MADSN and DSN conducted to measure energy consumption and execution time. Again, in that study MADSN outperformed DSN [31].

[32] studied the use of mobile agents for data fusion in a WSN running MADSN and focused on creating an optimum design of the *itinerary* component of the agent to improve performance and minimize resource usage. [33] showed that MADSN could be successfully applied to the real-world problem of vehicle classification in an unattended ground sensor system. The study showed, however, that classification accuracy was low (about 23%) when only one sensor was used but improved to about 80% when a multisensor array was used. (Target classification accuracy with a single sensor was high (about 75%) only when the target was in close proximity to the sensor.)

2.2. Autonomic Wireless Sensor Networks (AWSN)

[23] introduced the concept of the autonomic wireless sensor network. They proposed that mobile agents could be deployed in the network to facilitate autonomic computing. Autonomic computing implies the ability to self-manage, self-maintain as well as interact with a user of the network at a policy rather than hardware level. The purpose of interaction would be to allow interoperability with legacy systems if needed. Mobile agents in that context would support cooperation and negotiation in the system via relevant protocols and a suitable agent communication language. The mobile agents would introduce flexibility in terms of the ability to modify agent code through agent migration or agent adaptation. The idea here is that an agent could evolve as system policies evolved. In modelling autonomic behaviour, an AWSN also uses the self-knowledge, self-protection and self-interest characteristics inherent in mobile agents of multi agent systems.

Autonomic behaviour in Marsh's system includes the ability to handle sensors which have been rendered inoperable due to some type of damage, and the system can put into effect strategies to deal with that. For instance, the authors suggest that the system could make an estimate of data lost due to system damage and effect nearby sensors to increase their sampling rate to compensate. It could also mean a reconfiguration of the routing topology to minimize or eliminate message loss. The AWSN concept is closely related to the multi agent system framework and incorporates some of the same ideas including that of entities migrating from node to node while making 'intelligent' decisions. Fig. 2 shows the setup of the experiment.

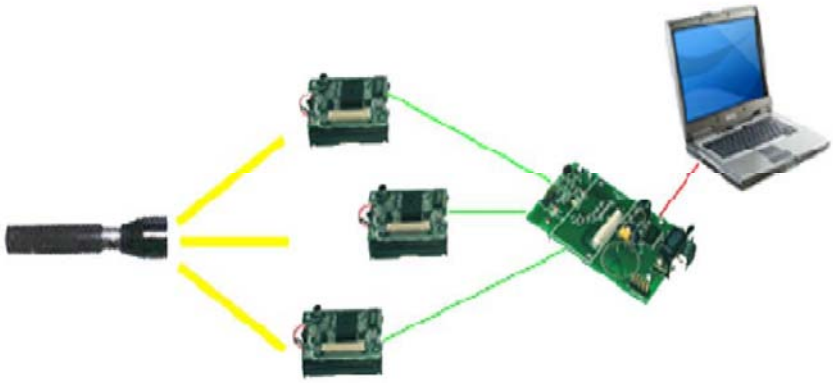

Fig. 2. Setup of the intruder detector experiment, each node houses an agent, taken from [23].

[34] and [35], building on this idea of an autonomic wireless sensor network, proposed an agent-based approach to implementing intelligent power management. In their work they promote the use of agents in intelligently activating or deactivating nodes for power conservation based on interpolation. For example, the following criteria are used to decide whether a node should be put to sleep: a node is considered redundant if the remaining nodes can interpolate the temperature at its location to a desired degree of accuracy. Redundancy in this context means that the node is not needed since the other nodes or the network can operate effectively without it.

In Marsh's scenario, a *coordinator* agent is appointed and aggregates all calculations performed by other agents in deciding whether deactivation should occur or not. The coordinator agent sends out requests to other agents to calculate the components of the interpolation function used and send that result back to the coordinator. The coordinator then calculates the actual interpolated temperature, then requests the temperature from the appropriate agent. If the two values fall within a given error range the node is put into sleep mode.

It must be noted, however, that the issue of power management is influenced by a number of other network characteristics such as coverage, latency and longevity. Any power management scheme would have to balance the tradeoffs in making decisions on how power can be conserved. In the above example, the inherent autonomic characteristics of the network itself are used for efficiently managing power usage. Although a promising approach, this concept has not yet been implemented in agent-based WSN deployments.

2.3. Mobile Agent-based Wireless Sensor Networks (MAWSN)

[19] identified that the operation of MADSN was really based on three main assumptions. First, that the network had a cluster-based architecture, second, that each source node (nodes with data) was one hop away from the clusterhead, and third, that the redundant data collected could all be fused into one packet of fixed size [19, 36]. Those restrictions in a real-world context seemed to impose too strict limitations on the applications that could make use of the architecture and excluded many systems which did not boast all of these features. Hence, [19] proposed MAWSN as an alternative architecture to address networks where the features needed for the MADSN approach to work were not evident. Previous work done by [36] showed that in multi-hop environments without clusters, mobile agents could be employed to eliminate data redundancy by employing context-aware local processing techniques, use data aggregation to eliminate spatial redundancy with sensors that were of close proximity to each other, and reduce communication overhead by combining tasks. They were able to

reduce information redundancy at three levels: the node level, the task level and the combined task level.

At the node level mobile agents were assigned processing code specifically targeting the requirements of the specific application. The result was that the mobile agent only needed local processing of that data requested by the application thereby reducing the amount of data transmitted as only relevant data was extracted from nodes for transmission. At the task level, the mobile agent aggregated sensed data from individual nodes when visited. Finally, at the combined task level, the mobile agent used the packet unification technique to unify shorter data packets to one longer packet again reducing the number of transmissions. MAWSN built on the encouraging results in [36] and experimentally showed improvements in energy consumption for data packets transmission over the more traditional client-server based WSN.

Concurrently with the above work, [37] proposed a framework aimed at implementing a wide variety of sensor network applications. Their aim was to create an architecture that allows multiple applications to share a network and permits scalable deployments that can evolve over time. In their design autonomous agents are deployed in the network with a particular agent having the ability to be on one or more nodes. Their framework is built atop TinyOS and its associated virtual machine Mate [38] and agents run on Crossbow motes. The agents were shown to use resources only on those motes they visited so resources in the network were used more efficiently. Agents were able to read sensor values, start devices and sent radio packets if required.

In a data collection experiment aiming to find the maximum value in a field of sensor readings, an agent was injected into the network via broadcast. After arriving at a node, it re-broadcast itself if the reading at the current node was greater than that at the previous node. Agents stopped propagating once the region with the maximum sensor reading was reached. This was done by incrementing a count value when an agent arrived in the area. Once that value went over a pre-defined threshold the last agent to enter the area could issue a notification message to the processor.

Analysis of the experiment showed that as the size of the network increased, a smaller proportion of the network needed to be active. This meant that in large networks agents stayed on nodes only long enough to perform their task and quickly release resources for other task execution. The conclusion of the experiment was that the agent model presented was successful in showing that it could scale up without having to occupy all nodes on the network in performing tasks; however some areas for future work were identified. The first is security to ensure that unauthorized agents are not able to mount energy-depletion and denial-of-service attacks among others. The second involves giving agents the capability to query nodes for information such as processing power and energy. The third area involved giving agents the ability to cache their code on a node to reduce the number of agent transmissions.

In a similar approach [39] describe the agent oriented programming paradigm for development of intelligent sensor networks. They implemented a test application using the Java Agent Development Framework (JADE) [40] aiming to develop an *intelligent* ground sensor network. The application consisted of an Unattended Ground Sensor Network (UGSN) used to monitor moving targets with agents analyzing the data being acquired by nodes. In the experiment, information from the sensors were correlated and fused by reasoning agents using a fusing algorithm (the majority voting ordinary technique) suited for distributed information fusion. A special agent called the *Target Agent (TA)* is created on the node where the first sensor trigger occurs. The *TA* then reasons with its neighbours and after correlating their sensor data, decides what node it should migrate to next. The objective of migration is to get closer to the target being tracked. After each move the *TA* performs data fusion using the new data available on its host node. Through the *TA*, the network as a whole, decides when external applications need to be updated and what information should be sent to those applications.

The experiment was successful in its sole aim of showing that agents were suitable for distributed information fusion. Distributed information fusion is the idea, quite similar to aggregation, that data generated by nodes in a network can be efficiently combined in-network instead of transmitting all raw data to a sink for processing. The main advantages of this approach were identified as: reduction in data redundancy since we have agents deciding whether data is important enough to be used; reduction in power consumption in having all data sent directly back to a central location; conservation of communication bandwidth by limiting the exchange of data between nodes as much as possible.

The advantages of agent-based distributed information fusion parallel those exhibited by other systems, particularly query-based systems that incorporate in-network aggregation.

2.4. Multi-Agent Systems

Although the idea of using agents in wireless sensor networks is widely supported by the literature [27, 31, 41, 32, 36], the actual use of multi-agent systems in WSNs is not as widespread as one might expect [42]. There are of course the big problems of deployment, testing and debugging of such systems on what is essentially a distributed application with minimal interfaces to support user interaction. There have been real deployments using agents [20, 37, 43, 36] but multi-agent system deployments are still rare. [42] propose a methodology for multi-agent deployment. They start with a base station implementation for the agent which leads to a distributed implementation where agents are mapped to nodes, using either one-to-one, many-to-one or one-to-many mapping. This mapping is used to model the interaction expected between agents and the base station, as well as iron out communication and interaction rules between agents. These first two stages are carried out on a device with more processing power like a laptop or desktop. In the third and final stage, the statements and rules governing agent behaviour are translated to the language of the WSN system hardware that will be hosting the agent. The result of the last stage is a topology in which the load of executing an algorithm is distributed among the agents which can allow faster response times for complex algorithms. In addition, it can also result in a reduction in the number of transmissions in the network since packets which previously had to be routed to the base station for processing can now be processed by an agent on a node or on a neighbouring node.

In the examples described above, agents are pieces of code injected into the system, gathering data, making decisions and migrating between nodes in some cases. Some research, however, has investigated the concept of having nodes themselves act as intelligent agents. [44] use intelligent agent theory to model sensor behaviour, viewing the WSN as a network of distributed autonomous nodes with the capability of making decisions. They also propose using artificial intelligence strategies on nodes to enable decision making, selecting a rule-based expert system that makes use of locally collected information on a node to make step-wise decisions. The concept, however, has not yet been implemented and deployed.

3. Agent-based Middleware

In general terms, middleware sits between the operating system and the application with its main function being to provide support for development, deployment, execution and maintenance of sensing applications [45]. There are a number of categories of middleware based on the objectives of the approach and the way in which the wireless sensor network is viewed. With some, the main goal is to provide a dynamic reprogramming capability while others seek to provide a platform-independent model on which sensor network programs can be written and executed. Other approaches try to provide a cross-layer management approach and provide functionality for manipulating other services like routing, etc. Information extraction which is ultimately the goal of the sensor network applications

built using these models is supported by the services provided by the particular middleware approach. Approaches include the use of virtual machines which in many cases are application-specific but reduce the amount of code that has to be transmitted and therefore reduces the cost of communication, as well as modular programming approaches which allow easier and more efficient reprogramming of sensor nodes [46]. Some agent-based middleware will be described below.

3.1. Mate

Virtual machines (VMs) provide one middleware solution to the problems posed by wireless sensor networks. In WSNs the greatest drain on energy resources is the cost of communication, so if the traffic needed to reprogram motes could be reduced this could lead to a longer network lifetime as well as the ability to reprogram more frequently. Mate is a middleware approach that abstracts high-level operations into virtual machine bytecodes (see Fig. 3). As a result a VM program can be very short, bytes long instead of kilobytes. Mate, therefore, is a virtual machine designed for sensor networks [38] and forms the basis for many agent-based information gathering applications.

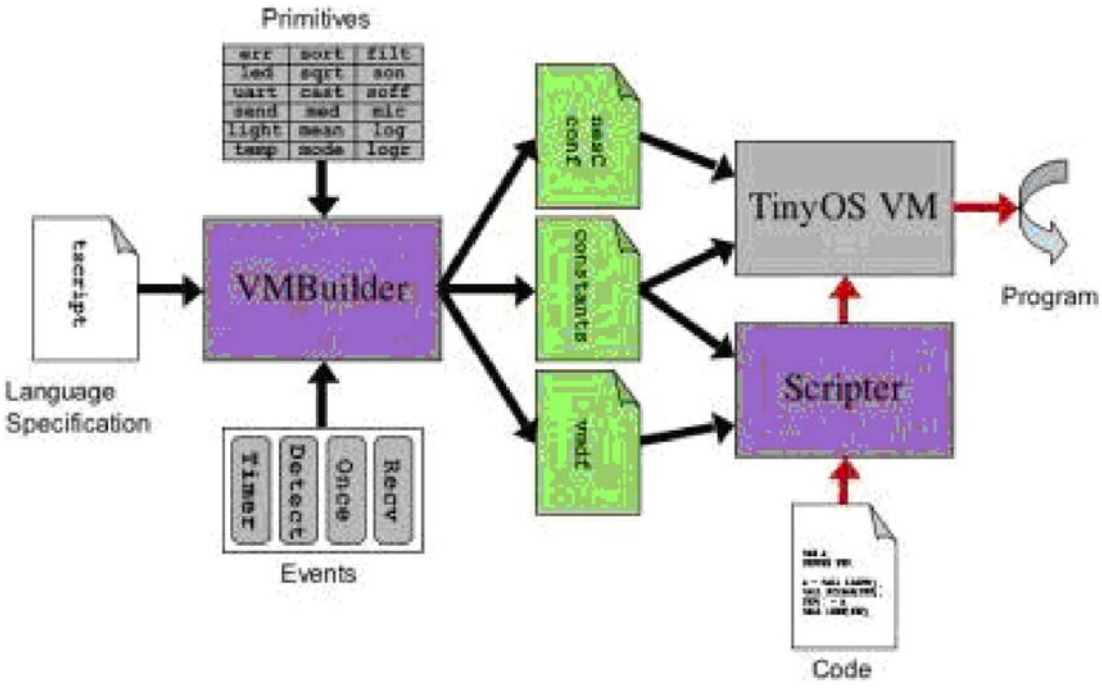

Fig. 3. Construction of a Mate virtual machine in compiling and running a generated program taken from [47].

Mate is a bytecode interpreter, running on TinyOS and provides a high-level interface for creating small (<100bytes) but complex programs. Mate is focused on energy conservation in transmission of code and breaks code into small sections called *capsules* (referred to as agents by some), each 24 bytes long. Programmers write applications and the system injects them into the network using certain algorithms that minimize energy and resource usage. The small size of the modules makes it easier to distribute into the network. Mate specifically targets the constraints of limited bandwidth and power by allowing adjustments or reprogramming of capsules before they are issued to the network.

Mate is a general architecture that allows a user to create a wide variety of virtual machines. A user builds a Mate VM in three steps [45]: a) the user selects a language that defines a set of VM bytecodes representing the basic functionality; b) the user selects the execution events. Each event has its own

execution content which runs when an event is triggered; c) the user selects the primitives to be used. These are operations that provide functionality beyond that of the selected language.

A set of files are generated after these steps have been executed. Some will build the VM which will run on the motes, others will build a scripter program with information on the language and primitives selected. The user can now construct programs in the scripter which will be compiled down to the VM-specific binary code. These bytecodes are then issued to the network and processed by a VM there which will execute it.

To reprogram motes, the user needs only add one mote with the new capsule to the network. When a mote hears a new capsule it immediately starts forwarding it until every mote in the network has been updated.

The advantages of Mate, therefore, are the simplicity of programming for the sensor network as well as the ease with which the network can be reprogrammed. Mate's authors have identified future work as being the creation of propagation algorithms for larger programs, security and scripting languages that can be compiled to a Mate VM.

3.2. Agilla

Agilla is a mobile agent middleware based on Mate that allows quick deployment of adaptive applications [18, 48]. Instead of creating capsules which are flooded throughout the network, Agilla allows the user to create mobile agents which can be injected into the network. Agents are dynamic and intelligent, and can coordinate and collaborate locally or through remote invocation. These agents can clone themselves and move from node to node as they perform tasks. Agilla affords more flexibility because it allows applications to decide how agents in the system should migrate and spread in the network while maintaining their code and state.

In the Agilla Model, each node can have a maximum of four agents and addressing is done using the geographic location of nodes instead of IDs. Agilla can therefore be used in running applications over geographic regions and to allow geographic routing for any multi-hop interactions [18]. Inter-agent coordination in Agilla is facilitated using a tuple space and an acquaintance list which are maintained on each node. Each node's tuple space is shared by local agents and remotely accessible. Global tuple spaces are not supported due to the energy and bandwidth constraints, rather separate local tuple spaces are maintained by each node and agents can access remote tuple spaces using special instructions provided in the middleware. Each node also maintains an acquaintance list containing the location of all one-hop neighbours and local agents can access it via a number of provided functions. Fig. 4 shows the programming model for the Agilla system.

[48] used Agilla to successfully deploy fire and tracker agents in a fire tracking application and demonstrated the reliability and efficiency of the exercise in this case study [18]. The scenario is as follows: a fire starts in a region of the network and as it spreads *tracker* agents crowd round it repeatedly cloning themselves until a perimeter is formed. Once that perimeter is formed a fire fighter is informed. The fire fighter injects a *guidance* agent who is responsible for guiding the fire fighter along a safe path to the fire. The study focused on the *tracker* agent, leaving the *guidance* agent and the development of a *safe-route* algorithm to future work.

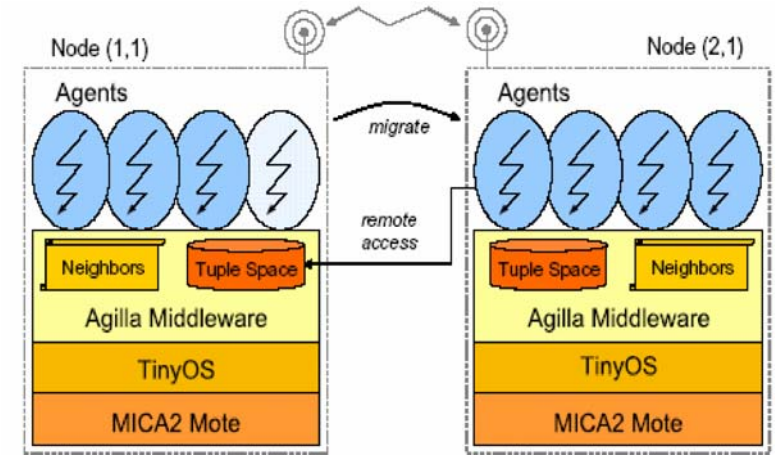

Fig. 4. The Agilla programming model taken from [48].

[48] used Agilla to successfully deploy fire and tracker agents in a fire tracking application and demonstrated the reliability and efficiency of the exercise in this case study [18]. The scenario is as follows: a fire starts in a region of the network and as it spreads *tracker* agents crowd round it repeatedly cloning themselves until a perimeter is formed. Once that perimeter is formed a fire fighter is informed. The fire fighter injects a *guidance* agent who is responsible for guiding the fire fighter along a safe path to the fire. The study focused on the *tracker* agent, leaving the *guidance* agent and the development of a *safe-route* algorithm to future work.

Two types of fire modelling agents were used, static fire agents and dynamic fire agents. Static agents simply insert a fire tuple into the local tuple space and then repeatedly blink a red LED (visual indicator of network state). These agents are used to create fires of different shapes. The dynamic agent models a spreading fire. It inserts a fire tuple upon arrival at a node, blinks the red LED a number of times, clones itself onto a random non-burning neighbour and repeats the blinking. The cloning and blinking process is repeated until every node is on fire. The rate of spread can be controlled by the number of times a node blinks between cloning operations.

A fire tracking agent is responsible for discovering a fire and forming a boundary around it. The tracker agent inserts a tracker tuple when it arrives at a node. Other tracker agents will use this tuple to check whether a neighbouring tracker agent is still present on the node. If the node a tracker agent is on catches on fire the tracker agent dies. A reaction is registered when this occurs, the tracker agent will turn off all LEDs, remove its tuple and stop functioning.

The life cycle of a tracker agent is as follows: it repeatedly checks whether any of its neighbours are on fire. If there are none, it moves to a random neighbour and repeats the check. If, however, a neighbour is on fire it enters a tracking mode and lights up its green LED and repeats the following. It determines the locations of all neighbours that are on fire and for each non-burning neighbour within a certain distance of the fire clones itself to those nodes. This process is repeated until the fire dies. Periodically checking for neighbours close to the fire allows the agent to adjust the perimeter. Once a fire was started, a tracker agent was injected next to it. This allowed the investigators to focus solely on the efficiency of perimeter formation and not on fire discovery as well.

The experiment was successful in a number of ways. First, it demonstrated the use of Agilla in deploying complex applications. Second, it showed that multiple applications could share the network at the same time (fire-simulation and tracker application). Third, it showed that mobile agents can be used to successfully program WSNs. Experiments on a 26-node MICA2 network showed that tracker agents each 101 bytes long were able to form a perimeter around a static as well as a dynamic fire. It

was also evident that the efficiency of perimeter formation, in the case of static fires, depended to a great extent on the amount of agent parallelism in the system. The authors have identified areas of future work aimed at allowing new instructions to be added to the network after deployment and also allowing the instruction set to be customized depending on the agent being deployed. [18] describes the experiment and its results in greater detail.

3.3. Impala

Impala can be considered another mobile agent-based middleware approach. Its middleware architecture allows a user to create modular, adaptable and maintainable applications for WSNs. The adaptation facilities allow changes to be made at runtime in an effort to improve energy-efficiency, reliability and performance of running applications. The idea is that given the need for long-term management of a sensor application, a middleware layer that can update and adapt applications dynamically, switch to new protocols easily at runtime is a big advantage. Impala, therefore, is a middleware layer that acts as an operating system, resource manager and event filter upon which applications can be installed and executed [49]. It was specifically designed for a wildlife monitoring project, ZebraNet, also detailed in [49].

The system architecture is depicted in Fig. 5. The architecture's upper layer contains all application protocols and programs for ZebraNet. In the lower layer are three middleware agents: Application Updater (AU), Application Adapter (AA) and Event Filter (EF). The AU receives and transmits software updates to the node via the wireless transceiver, the AA adapts the protocols to different conditions at runtime in a bid to improve energy-efficiency and performance and the EF collects and sends events to the system for processing.

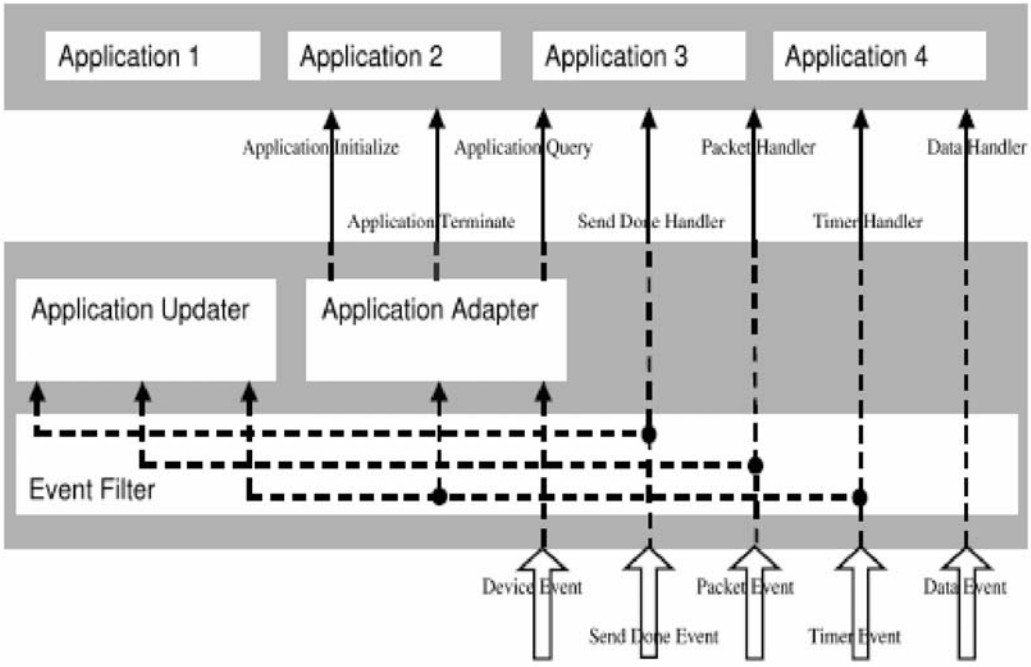

Fig. 5. Architecture of Impala system taken from [49].

A program module is compiled into binary code before it is injected into the network. Linking is performed by the updater on every node and when all modules in an update have been received the module is linked to the main program. Modules are linked independently. Agents work autonomously

making the network more fault-tolerant and better able to self-organize. It is not suitable, however, for resource-limited systems.

Impala uses an event-based programming model where the AA, AU and applications are programmed as a set of event handlers called by the EF when appropriate events occur. The EF controls various operations and starts a number of processing chains including timer, send, done data and device events. Based on different scenarios such as improving energy efficiency or performance, the AA will adapt a particular application. The AU is then responsible for effecting software updates while being cognizant of such issues as resource or bandwidth constraints and node mobility. If, for example, the radio transceiver on a node was to fail, the EF would call on an appropriate handler in the AA. The AA would determine the impact of that failure and decide whether an updated or new application that takes into account these issues should be issued to the network. The AU would then be responsible for carrying out the software update itself.

3.4. SensorWare

Although not strictly considered mobile agent architecture by some, SensorWare [50] is based on a scriptable lightweight run-time environment suited for nodes with energy and memory constraints. It is formally called an active sensor framework (ASF) where mobile scripts less than 180 Kbytes are used to make the network programmable and open to users.

The difficulty in designing an ASF has been identified as determining how to accurately define the abstraction of the run-time environment. Any abstraction would have to result in compact code, resource sharing, multiple users being able to access the system and portability to different platforms. With these in mind they designed a node abstraction that allows multiple users access to the modules on a node while still being able to create new modules.

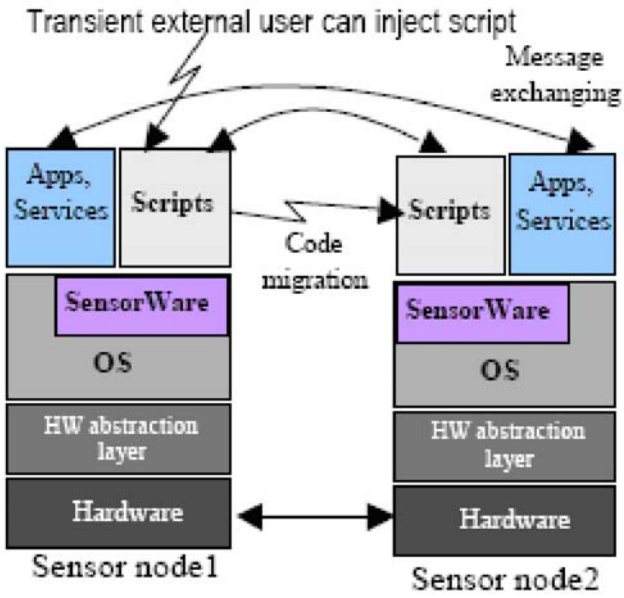

Fig. 6. Architecture of sensor nodes running SensorWare taken from [50].

The authors focused on defining a framework that allowed the description and deployment of distributed algorithms for wireless ad-hoc sensor networks. SensorWare's language model can

effectively express these algorithms while simultaneously shielding the user from low level details while allowing sharing of node resources among several concurrently running applications or many users. The language model focuses on the properties of efficient algorithms for sensor networks while application development for a real network is underway. Fig. 6 shows the architecture of the SensorWare system.

SensorWare is also quite effective at dynamically deploying the distributed algorithms represented in the language. Given that the nodes are memory-constrained and cannot store every application in local memory the method of dynamic deployment has to be effective. Rather than each node being programmed by the user the SensorWare approach gets the nodes themselves to program their neighbours. The user injects the program into the network and the program is autonomously distributed to the nodes that should receive it. SensorWare is described in detail in [51].

3.5. TinyLIME

Unlike many of the examples described above that use a single controlling processor or sink node for data collection and processing, TinyLIME instead promotes a distributed approach. Multiple mobile clients or agents are distributed with the ability to receive data only from the sensors that they are directly connected to. The clients can share this locally collected data through their wireless interconnections [52]. TinyLIME is built on the original Linda in a Mobile Environment (LIME) architecture [53] and deployed on Crossbow motes. All base stations run an instance of moteAgent, a LIME agent that among other functions maintains data freshness and historical information on recent sensed values. TinyLIME also supports data aggregation both over values sensed by multiple sensors and those sensed by a single sensor.

4. Concluding Remarks

The paper provided and overview of information extraction techniques for wireless sensor networks and surveyed, in some detail the agent-based approaches to information extraction.

The constraints and design challenges for information extraction systems suitable to be deployed within real-life WSN applications have been highlighted, stressing the view that no method fits all.

With respect to agent-based approaches to information extraction, it is forthcoming from the survey that many of the described techniques and architectures supporting them have not yet been sufficiently developed and evaluated other than theoretically and are, hence, far from being deployable (or even successfully implementable) on resource constrained nodes such as those found most commonly in WSN applications.

It is, however, encouraging to see the wealth of research and proposals related to agent, multi-agent and mobile agent systems, leading, no doubt, in the future, to energy efficient, hardware independent, robust methods of relaying information drawn from large scale WSNs, to the user.

References

[1]. A. Mainwaring, J. Polastre, R. Szewczyk, D. Culler, J. Anderson, Wireless sensor networks for habitat monitoring, In *Proceedings of the ACM International Workshop on Wireless Sensor Networks and Applications*, Atlanta, GA, United States, 2002, pp. 88–97.

[2]. R. Guy, B. Greenstein, J. Hicks, R. Kapur, N. Ramanathan, T. Schoellhammer, T. Stathopoulos, K. Weeks, K. Chang, L. Girod, D. Estrin, Experiences with the extensible sensing system, *Tech. Rep.* 61, CENS, 2006.

[3]. K Langendoen, A. Baggio, O. Visser, Murphy loves potatoes experiences from a pilot sensor network deployment in precision agriculture, In *Proceedings of 20th International Parallel and Distributed Processing Symposium, IPDPS 2006*, 2006.

[4]. Thaddeus R. F. Fulford-Jones, Gu-Yeon Wei, and Matt Welsh, A portable, low-power, wireless two-lead EKG system, In *Proceedings of Annual International Conference of the IEEE Engineering in Medicine and Biology - Proceedings*, San Francisco, CA, United States, Vol. 26, III, 2004, pp. 2141–2144.

[5]. Mauricio Castillo-Effen, Daniel H. Quintela, Ramiro Jordan, Wayne Westhoff, and Wilfrido Moreno, Wireless sensor networks for flash-flood alerting, In *Proceedings of the IEEE International Caracas Conference on Devices, Circuits and Systems, ICCDCS'04*, Dominican Republic, 2004, pp. 142–146.

[6]. K. Pister, The 29 palms experiment - tracking vehicles with a UAV-delivered sensor network, 2007, http://robotics.eecs.berkeley.edu/~pister/29Palms0103

[7]. HPWREN. High performance wireless research and education network, 2007, http://en.wikipedia.org/wiki/High_Performance_Wireless_Research_and_Education_Network

[8]. ROADNet, Real-time observatories, applications and data management network, 2007, http://roadnet.ucsd.edu

[9]. W. R. Heinzelman, A. Chandrakasan, H. Balakrishnan, Energy-efficient communication protocol for wireless microsensor networks, In *Proceedings of the 33rd Annual Hawaii International Conference on System Sciences*, Vol. 2, Maui, HI, USA, 2000, p. 10.

[10]. S. Madden, R. Szewczyk, M. J. Franklin, D. Culler, Supporting aggregate queries over ad-hoc wireless sensor networks, *In Proceedings of the 4th IEEE Workshop on Mobile Computing Systems and Applications*, Callicoon, NY, USA, 2002, pp. 49–58.

[11]. G. J. Pottie, W. J. Kaiser, Wireless integrated network sensors, *Communications of the ACM*, 43, 5, 2000, pp. 51–8.

[12]. P. Bonnet, J. Gehrke, P. Seshadri, Querying the physical world, *IEEE Personal Communications*, 7, 5, 2000, pp. 10–15.

[13]. Y. Yao, J. Gehrke, Query processing for sensor networks, In *Proceedings of Conference on Innovative Data Systems Research (CIDR)*, 2003.

[14]. J. Heidemann, F. Silva, C. Intanagonwiwat, R. Govindan, D. Estrin, D. Ganesan, Building efficient wireless sensor networks with low-level naming, *Operating Systems Review (ACM)*, Vol. 35, Banff, Alta., Canada, 2002, pp. 146–159.

[15]. R. Newton, Compiling Functional Reactive Macroprograms for Sensor Networks, *Masters Thesis*, Massachusetts Institute of Technology, 2005.

[16]. R. Newton, G. Morrisett, M. Welsh, The regiment macroprogramming system, In *Proceedings of the 6th International Conference on Information Processing in Sensor Networks*, 2007, Cambridge, Massachusetts, USA, pp. 489-498.

[17]. H. Qi, Xiaoling Wang, and K. Chakrabarty, Multisensor data fusion in distributed sensor networks using mobile agents, In *5th International Conference on Information Fusion*, Annapolis, MD., 2001.

[18]. C. L. Fok, G. C. Roman, and C. Lu, Mobile agent middleware for sensor networks: an application case study, In *Proceedings of Fourth International Symposium on Information Processing in Sensor Networks* (IEEE Cat. No. 05EX1086), Los Angeles, CA, USA, 2005, pp. 382–387.

[19]. M. Chen, T. Kwon, Y. Yuan, and V. C. M. Leung, Mobile agent based wireless sensor networks, *Journal of Computers*, 1, 1, 2006, pp. 14–21.

[20]. Q. Wu, N. S. V. Rao, J. Barhen, S. S. Iyenger, V. K. Vaishnavi, H. Qi, and K. Chakrabarty, On computing mobile agent routes for data fusion in distributed sensor networks, *Knowledge and Data Engineering, IEEE Transactions on*, 16, 6, 2004, pp. 740–753.

[21]. D. Massaguert, C. Fok, Nalini Venkatasubramanian, G. Roman, C. Lu, Exploring sensor networks using mobile agents, In *Proceedings of the International Conference on Autonomous Agents*, Hakodate, Japan, 2006, pp. 323–325.

[22]. R. Tynan, G. M. P. O'Hare, D. Marsh, and D. O'Kane, Multi-agent system architectures for wireless sensor networks, In *Proceedings of 5th International Conference on Computational Science*, ICCS 2005, (Lecture Notes in Computer Science Vol. 3516), Atlanta, GA, USA. *Springer-Verlag*, 2005, pp. 687–94.

[23]. D. Marsh, R. Tynan, D. O'Kane, and G. M. P. O'Hare, Autonomic wireless sensor networks, *Engineering Applications of Artificial Intelligence*, 17, 7, 2004, pp. 741–8.

[24]. Robert Wesson, Frederick Hayes-Roth, John W. Burge, Cathleen Stasz, and Carl A. Sunshine, Network structures for distributed situation assessment, *IEEE Transactions on Systems, Man and Cybernetics*, SMC-11, 1, 1981, pp. 5–23.

[25]. D. N. Jayasimha, S. Sitharama Iyengar, and R. L. Kashyap, Information integration and synchronization in distributed sensor networks, *IEEE Transactions on Systems, Man and Cybernetics*, 21, 5, 1991, pp. 1032–1043.

[26]. L. Prasad, S. S. Iyengar, R. L. Kashyap, and R. N. Madan, Functional characterization of sensor integration in distributed sensor networks, *In Proceedings of the Fifth International Parallel Processing Symposium* (Cat. No. 91TH0363-2), Anaheim, CA, USA, 1991, pp. 186–93.

[27]. H. Qi, S. Iyengar, and K. Chakrabarty, Multiresolution data integration using mobile agents in distributed sensor networks, *Systems, Man and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 31, 3, 2001, pp. 383–391.

[28]. Yao Yuxia, Tang Xueyan, and Lim Ee-Peng, In-network processing of nearest neighbor queries for wireless sensor networks, Database Systems for Advanced Applications, In *Proceedings of 11th International Conference, DASFAA 2006*. (Lecture Notes in Computer Science Vol. 3882), Singapore, 2006, pp. 35–49.

[29]. K. Sohrabi, J. Gao, V. Ailawadhi, and G. J. Pottie, Protocols for self-organization of a wireless sensor network, *IEEE Personal Communications*, 7, 5, 2000, pp. 16–27.

[30]. H. Qi, S. S. Iyengar, and K. Chakrabarty, Distributed multi-resolution data integration using mobile agents, In *Proceedings of IEEE Aerospace Conference 2001 (Cat. No. 01TH8542)*, Big Sky, MT, USA, 2001, pp. 3–1133.

[31]. Hairong Qi, Yingyue Xu, and Xiaoling Wang, Mobile-agent-based collaborative signal and information processing in sensor networks, In *Proceedings of the IEEE*, 91, 8, 2003, pp. 1172–1183.

[32]. Hairong Qi and Wang Feiyi, Optimal itinerary analysis for mobile agents in ad hoc wireless sensor networks, In *Proceedings 13th International Conference on Wireless Communications, Proceedings Wireless 2001.*, Calgary, Alta., Canada, 2001, pp. 147–53, TRLabs/Univ. Calgary.

[33]. H. Qi, Xiaoling Wang, S. Sitharama Iyengar, K. Chakrabarty, High performance sensor integration in distributed sensor networks using mobile agents, *International Journal of High Performance Computing Applications*, 16, 3, 2002, pp. 325–335.

[34]. G. M. P. O'Hare, D. Marsh, A. Ruzzelli, and R. Tynan, Agents for wireless sensor network power management, In *Proceedings of International Workshop on Wireless and Sensor Networks (WSNET-05)*, Oslo, Norway, 2005.

[35]. R. Tynan, D. Marsh, D. O'Kane, G. M. P. O'Hare, Intelligent agents for wireless sensor networks, In *Proceedings of the International Conference on Autonomous Agents*, Utrecht, Netherlands, 2005, pp. 1283–1284.

[36]. C. Min, K. Taekyoung, C. Yanghee, Data dissemination based on mobile agent in wireless sensor networks, In *Proceedings of Conference on Local Computer Networks, LCN*, Sydney, Australia, 2005, pp. 527–528.

[37]. L. Szumel, J. LeBrun, J. D. Owens, Towards a mobile agent framework for sensor networks, In *Proceedings of 2nd IEEE Workshop on Embedded Networked Sensors, EmNetS-II*, Sydney, Australia, 2005, pp. 79–87.

[38]. Philip Levis and David Culler, Mate: A tiny virtual machine for sensor networks, In *Proceedings of International Conference on Architectural Support for Programming Languages and Operating Systems - ASPLOS*, San Jose, CA, United States, 2002, pp. 85–95.

[39]. B. Karlsson, Oscar Bäckström, Wlodek Kulesza, and Leif Axelsson, Intelligent sensor networks: An agent-oriented approach, In *Proceedings of REALWSN*, Stockholm, Sweden, 2005.

[40]. Jade, http://jade.tilab.com, 2006.

[41]. V. K. Vaishnavi, Q. Wu, N. S. V. Rao, H. Qi, K. Chakrabarty, S. S. Iyenger, and J. Barhen, On computing mobile agent routes for data fusion in distributed sensor networks, *IEEE Transactions on Knowledge and Data Engineering*, 16, 6, 2004, pp. 740–53.

[42]. Richard Tynan, Antonio Ruzzelli, and G. M. P. O'Hare, A methodology for the deployment of multi-agent systems on wireless sensor networks, In *Proceedings of 17th International Conference on Software Engineering and Knowledge Engineering, (SEKE '07)*, Taiwan, China, *IJSEKE press*, 2005.

[43]. Kenneth N. Ross, Ronald D. Chaney, George V. Cybenko, Daniel J. Burroughs, and Alan S. Willsky, Mobile agents in adaptive hierarchical bayesian networks for global awareness, In *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*, San Diego, CA, USA, 1998, pp. 2207–2212.

[44].M. Mertsock, D. Stawski, Wireless sensor nodes as intelligent agents: Using expert systems with directed diffusion, 2005, www.mertsock.com/downloads/wsn-ia-dd.pdf
[45].K. Romer, O. Kasten, F. Mattern, Middleware challenges for wireless sensor networks, *SIGMOBILE Mob. Comput. Commun. Rev.*, 6, 4, 2002, pp. 59–61.
[46].Ryo Sugihara and Rajesh K. Gupta, Programming models for sensor networks: A survey, *ACM Transactions on Sensor Networks*, 4, 2, 2008.
[47].Mate, Mate: Building application-specific sensor network language runtimes, 2003, http://www.cs.berkeley.edu/~pal/research/mate.html
[48].Chien-Liang Fok, Gruia-Catalin Roman, and Chenyang Lu, Rapid development and flexible deployment of adaptive wireless sensor network applications, In *Proceedings of International Conference on Distributed Computing Systems*, Columbus, OH, United States, 2005, pp. 653–662.
[49].Liu Ting and M. Martonosi, Impala: a middleware system for managing autonomic parallel sensor systems, *SIGPLAN Notices*, 38, 10, 2003, pp. 107–118.
[50].A. Boulis, C. Han, R. Shea, M. B. Srivastava, Design and implementation of a framework for efficient and programmable sensor networks, In *Proceedings of the 1st International Conference on Mobile Systems, Applications and Services*, San Francisco, California, 2003, pp. 187-200.
[51].A. Boulis, C. Han, R. Shea, M. B. Srivastava, Sensorware: Programming sensor networks beyond code update and querying, *Pervasive and Mobile Computing*, 3, 4, 2007, pp. 386–412.
[52].C. Curino, M. Giani, M. Giorgetta, A. Giusti, A. Murphy, G. Pietro Picco, Mobile data collection in sensor networks: The tinylime middleware, *Pervasive and Mobile Computing*, 1, 4, 2005, pp. 446–469.
[53].A. L. Murphy, G. P. Picco, G. C. Roman, Lime: a coordination model and middleware supporting mobility of hosts and agents, *ACM Transactions on Software Engineering and Methodology*, 15, 3, 2006, pp. 279–328.
